# PROGRAMOVÁNÍ MODULU

# ARDUINO

# V PROSTŘEDÍ MBLOCK


## POMOCÍ SADY

## „ARDUINO MAXI STARTER KIT“

# PROGRAMOVÁNÍ MODULU ARDUINO V PROSTŘEDÍ MBLOCK


## pomocí sady „Arduino MAXI Starter kit“

# Obsah

Úvodem… 3

mBlock + ARDUINO = ZAČÍNÁME! 5

Sériový monitor 11
Externí sériový monitor 12

Rozšíření „MAXI Starter kit“ 15

Přidání rozšíření MAXI Starter kit do prostředí mBlock 15

Příkazové bloky rozšíření MAXI Starter kit (ver. 1.0.3) 16

Jak používat modul ARDUINO 25

Vstupně/výstupní piny modulu Arduino 25
Digitální vstupy a výstupy 25
Analogové vstupy 28
Výstupy PWM 28

Speciální funkce některých pinů 30
Sériová komunikace 30
SPI komunikace 30
I²C komunikace 31

Arduino MAXI Starter kit 33

Využití digitálních vstupů a výstupů 39

Lekce 1 – Řízení LED tlačítkem 41

Lekce 2 – Otřesové čidlo 47

Lekce 3 – Modul relé 51

Lekce 4 – Bzučák 55

Lekce 5 – LED tekoucí potok 59
Použití proměnných 60

Lekce 6 – Kvízový bzučák 65
Použití podprogramu 66

Využití analogových vstupů (*analogově-digitální převodník*) 71

Lekce 7 – Interaktivní tekoucí LED světla 73

Lekce 8 – Fotorezistor 77

Lekce 9 – Senzor plamene 83

Lekce 10 – Zvukový senzor ..... 87
Lekce 11 – Ovládání zvuku pomocí světla ..... 91
Využití PWM výstupů ..... 95
Lekce 12 – Řízení LED pomocí PWM ..... 97
Lekce 13 – RGB LED ..... 101
Podprogram se vstupními parametry ..... 102
Lekce 14 – Servo ..... 107
Práce se sériovým monitorem ..... 111
Lekce 15 – Sériový monitor ..... 113
Lekce 16 – Joystick PS2 ..... 117
Lekce 17 – Infračervený přijímač ..... 121
Připojení periférií ..... 125
Lekce 18 – Krokový motor ..... 127
Lekce 19 – Sedmisegmentový LED zobrazovač ..... 131
Logická výstavba programu, přehlednost a rozsah kódu ..... 141
Lekce 20 –Čtyřmístný LED zobrazovač ..... 153
Lekce 21 – LED matice 8×8 ..... 163
Lekce 22 – Řízení LED zobrazovače pomocí 74HC595 ..... 177
Využití I²C displeje ..... 185
Lekce 23 – Používaní skeneru sběrnice I²C ..... 187
Lekce 24 – I²C LCD displej ..... 191
Lekce 25 – Voltmetr ..... 195
Lekce 26 – Senzor hladiny vody ..... 199
Lekce 27 – Teplotní čidlo LM35 ..... 203
Lekce 28 – Hodiny reálného času ..... 207
Lekce 29 – Senzor teploty a vlhkosti vzduchu DHT11 ..... 213
Lekce 30 – Vstupní ochranný systém s heslem ..... 217
Lekce 31 – RFID vstupní ochranný systém ..... 225
Něco navíc… ..... 231
BONUS – Sonarové čidlo HC-SR04 ..... 233

# Úvodem…

Dostupnost modulů Arduino i řady jejich doplňků dává příležitost vstoupit do světa elektroniky i lidem, kterým byla tato možnost doposud uzavřena. Stejně tak i různé didaktické tendence o začlenění programování do výuky relativně malých dětí nás staví do situace, kdy sice chceme cosi ovládat, ale nemůžeme očekávat ani elementární znalost potřebného programovacího jazyka. Jistým řešením takové situace může být použití některých z grafických programovacích prostředků. Naštěstí v tomto směru nám trochu štěstí přeje! Ze všech možností, které lze zvolit, jsem si pro tento tutoriál nakonec vybral prostředí mBlock, které mi přišlo poměrně hezké a intuitivní.

Společně s modulem Arduino se objevují různé stavebnice a sady zaměřené jak pro výuku na školách, tak jen „pro takové to domácí bastlení“. Obsahy i ceny těchto sad jsou různé. Když jsme na naší škole začali řešit, jak začlenit modul Arduino do výuky, začal jsem řešit, kterou z nabízených sad použít. Jelikož si myslím, že není nutné rovnou zakoupit celý balík několika výukových sad, aniž bych si je sám nevyzkoušel, rozhodl jsem se první sadu koupit domů pro své soukromé hraní. A jelikož to byl nákup „pěkně za své“, rázem jsem měl postaráno o motivaci, jak z široké nabídky výukových sad vybrat tu s nejlepším poměrem výkon/cena.

Nakonec jsem zvolil sadu LASKKIT Arduino MAXI Starter kit, která mě oslovila nejen velkým počtem komponent, ale především i určitou podporou. Nejde jen o krabici nacpanou součástkami, je zde k dispozici i hezky zpracovaný návod v podobě jakési učebnice a pochopitelně je možné stáhnout i zdrojové kódy pro prostředí Arduino IDE. Pokud s modulem Arduino začínáte, jistě se hodí jakýkoliv podpůrný materiál. A návod s třiceti na sebe postupně navazujícími a stále obtížnějšími zapojeními jednotlivých funkčních bloků a součástek byl skutečně fajn!

Spojení blokového prostředí mBlock a sady Arduino MAXI Starter kit mě nakonec natolik zaujalo svým širokým záběrem a přesto určitou jednoduchostí, že jsem pro další nadšence vytvořil tento text. Lze jej využít pro první seznámení s modulem Arduino, či jako inspiraci do zájmového kroužku. Možná jej někdo dokonce použije jako výukový text pro formování nových kormidelníků Průmyslu 4.xx. To již záleží na čtenáři samotném.

*V každém případě doufám, že Vám tento text bude aspoň částečnou inspirací pro další hrátky s modulem Arduino.*

Autor

**PODĚKOVÁNÍ:**
Text vznikl s laskavým svolením společnosti LaskaKit přepracováním a rozšířením jejich návodu ke studijní experimentální sadě „LaskaKit Arduino MAXI Starter kit, RFID“ (dříve označované jako „Arduino MAXI Starter kit“) – https://www.laskakit.cz/laskkit-arduino-maxi-starter-kit--rfid/.

**POZNÁMKA:**
Dále prezentované úlohy jednotlivých lekcí tohoto tutoriálu je možné použít i s jinou experimentální sadou, která obsahuje potřebné prvky. Funkčnost zdrojových kódů (zejména těch s použitím rozšířením „MAXI Starter kit“) byla však ověřena se sadou od společnosti LaskaKit. Pro správnou funkčnost je tedy nutné použít stejný typ periférie, jaký je v sadě LaskaKit Arduino MAXI Starter kit.

# mBlock + ARDUINO = ZAČÍNÁME!

Na úplný začátek si ukážeme, jak připojit modul Arduino k počítači, nainstalovat prostředí mBlock a nahrát první projekt. Zatím si nebudeme moc vysvětlovat, jak Arduino pracuje, ale jen zkusíme pomocí krátkého ukázkového programu, zda vše funguje, jak má. Na vlastnosti modulu Arduino se podrobněji podíváme v další kapitole. Nyní musíme především nainstalovat a nastavit prostředí mBlock, ve kterém budeme nadále programovat.


## 1. Připravíme si Arduino a USB kabel

V tomto tutoriálu předpokládáme, že budeme používat Arduino Uno. Dále budeme potřebovat standardní USB kabel (A plug to B plug). Jak modul Arduino, tak i USB kabel je součástí výukového setu Arduino MAXI Starter kit.


## 2. Stažení instalace prostředí mBlock (ver 5 a vyšší)

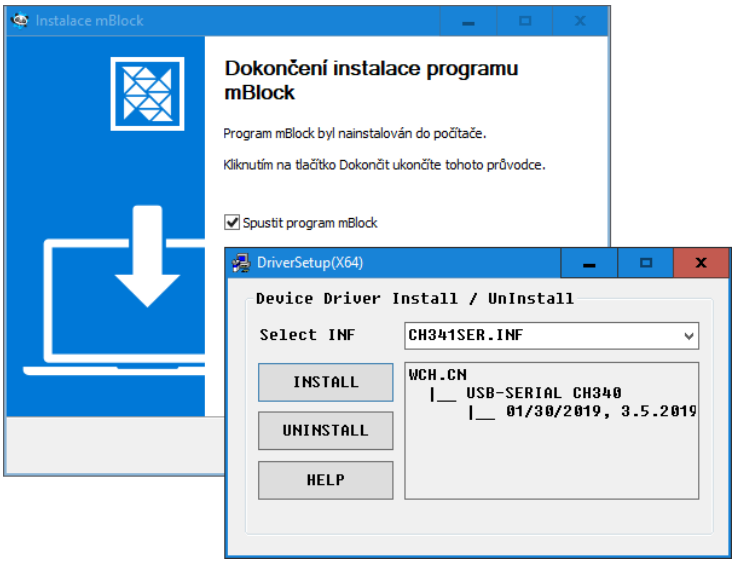
Stáhneme si poslední verzi prostředí mBlock ze stránek vývojářů: https://mblock.makeblock.com/*)

Jakmile se stahování instalačního programu dokončí, stažený soubor spustíme. Dále pokračujeme dle pokynů instalačního programu.

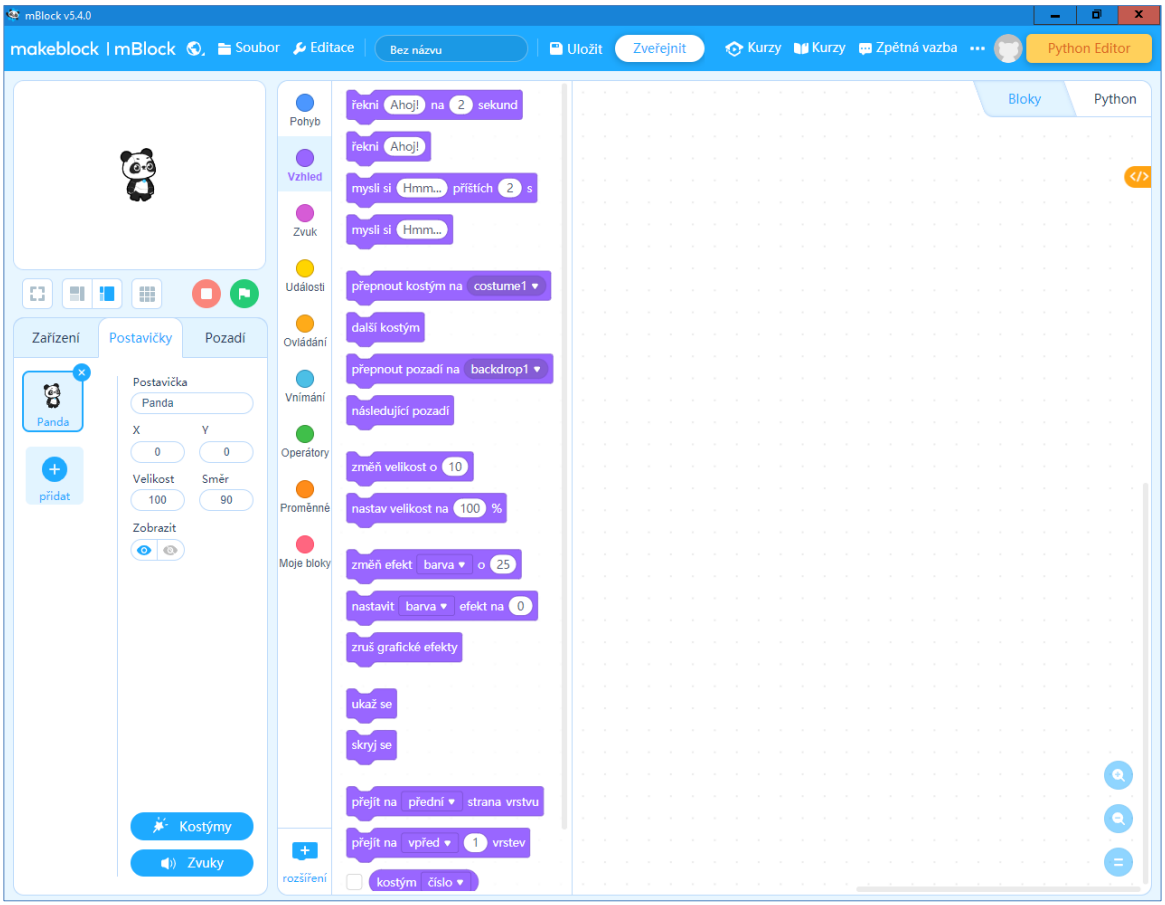
---

*) 1. Prostředí mBlock lze ve Windows 10 a 11 stáhnout i jako aplikaci přímo z Microsoft Store, tam ale občas bývá poslední verze přidána s určitým zpožděním.
2. V prostředí mBlock lze pracovat i v online verzi pomocí webového prohlížeče na adrese https://ide.mblock.cc, pak je ale třeba ještě doinstalovat rozšíření mLink (*po spuštění online prostředí mBlock se stačí řídit instrukcemi*)

Posledním krokem je instalace potřebných ovladačů. Na otevřeném novém okně klikneme na tlačítko INSTALL (viz následující obrázek). Pak okno zavřeme a můžeme spustit prostředí mBlock.




Pokud vše proběhlo bez problémů, můžeme říci: „Vítejte v prostředí mBlock!“

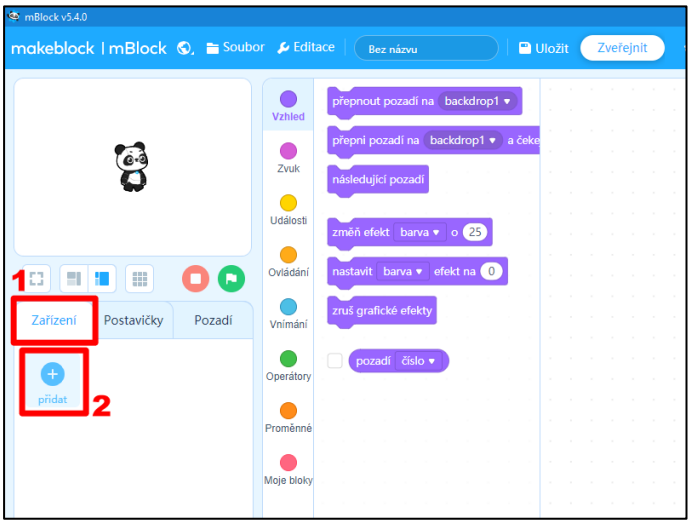


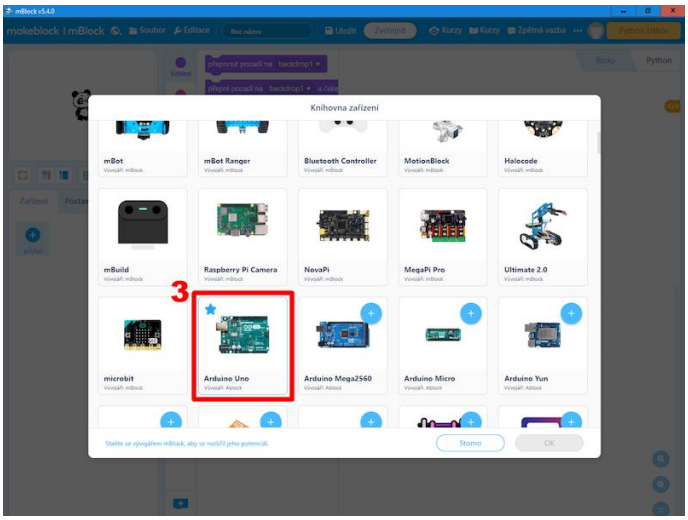
**ALE:**
Ještě než se pustíme do práce s modulem Arduino je třeba nastavit prostředí mBlock pro použití s modulem Arduino. Ale nebojte se, na to nám budou stačit následující tři kroky.

1. Klikneme na záložku **Zařízení**.

2. Pak klikneme na ikonu **přidat**.

3. A nakonec je třeba v seznamu najít a zvolit typ svého modulu Arduino – my zvolíme typ UNO (od vývojáře Ablock), se kterým budeme nadále pracovat. Na následujícím obrázku ale vidíme, že jsou zde k dispozici i typy Mega2560, Micro, Nano, Yun a Leonardo a další.




**Po stránce softwarové máme hotovo!** Možná ještě budeme časem potřebovat vědět, jak připojit do prostředí mBlock některá rozšíření pro některé použité moduly. Ale na to je zatím času dost.

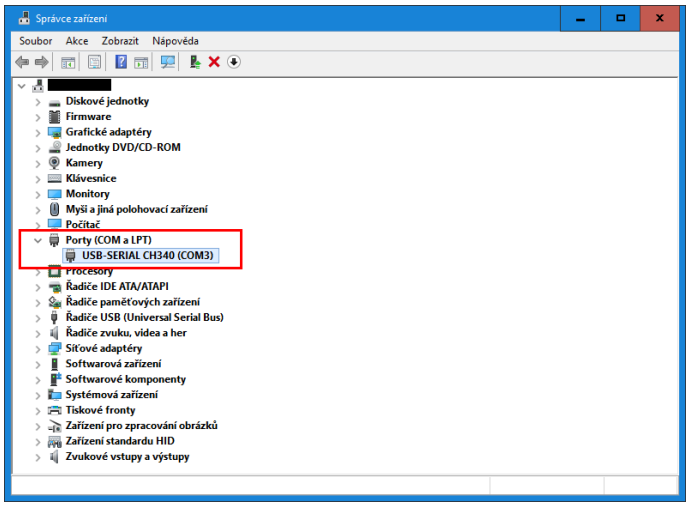
## 3. Připojení Arduino k USB portu PC

Moduly Arduino mohou být napájeny z USB nebo z externího zdroje. Pro naše prvotní zapojení stačí napájení z USB portu. Připojíme tedy modul Arduino ke svému počítači použitím USB kabelu. LED na desce modulu Arduino (označená PWR) by se měla rozsvítit.

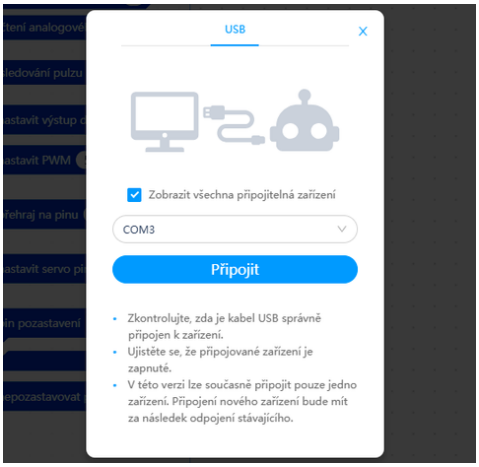
## 4. Instalace ovladače

Pokud jsme při instalaci prostředí mBlock potvrdil i instalaci ovladačů, máme hotovo! Jakmile zapojíme své Arduino k počítače, měl by se začít používat již instalovaný ovladač. O tom, zda byly ovladače správně

nainstalovány, se můžeme ujistit v operačním systému Windows otevřením „Správce zařízení“. V sekci „Porty COM a LPT“ by měla být položka „USB–SERIAL CH340(COMxx)“ (*nebo podobný*). To je deska Arduino.



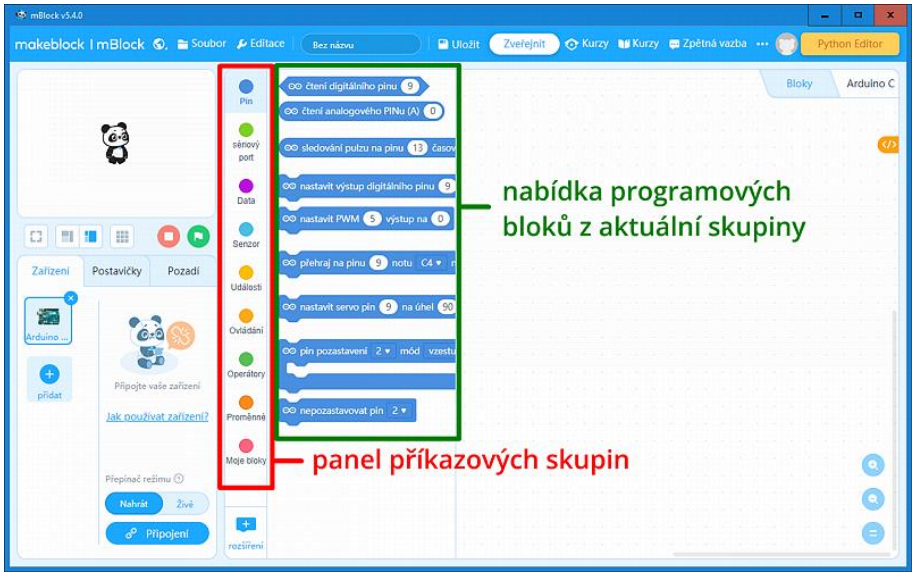

Propojení prostředí mBlock s modulem Arduino zařídíme v prostředí mBlock kliknutím na tlačítko Připojení a zvolením použitého USB portu. Z předchozího obrázku vidíme, že jde o port COM3, jeho nastavení v prostředí mBlock vidíme na následujícím obrázku. Pokud se při propojování prostředí mBlock s modulem Arduino v daném poli neobjeví číslo USB portu modulu Arduino, zaškrtneme volbu: „Zobrazit všechna připojitelná zařízení“. Připojení dokončíme volnou USB portu a tlačítkem Připojit.




## 5. Seznámení s pracovní plochou

Dobře se na hlavní obrazovce prostředí mBlock podívejme na pruh svislého panelu se skupinami příkazů – `Pin`; `Sériový port`; `Data`; `Senzor`… atd. Z těchto skupin budeme vybírat bloky příkazů a na pracovní ploše (vpravo) budeme sestavovat bloky do podoby programu. Klidně si zkusme nějaké bloky metodou Drag&Drop „natahat“ na pracovní plochu. Vyzkoušejme si, jak se některé bloky dají spojovat nebo vkládat do sebe.

Zkusme bloky příkazů i odstranit přetažením zpět nad příkazovou nabídku. Ovládání prostředí mBlock je opravdu intuitivní!




## 6. Náš první program

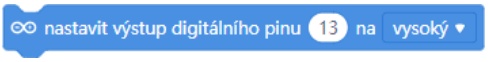
Program vždy začínáme blokem 

, který najdeme v sekci 

. Blok nekonečné smyčky 

nalezneme v příkazové skupině 

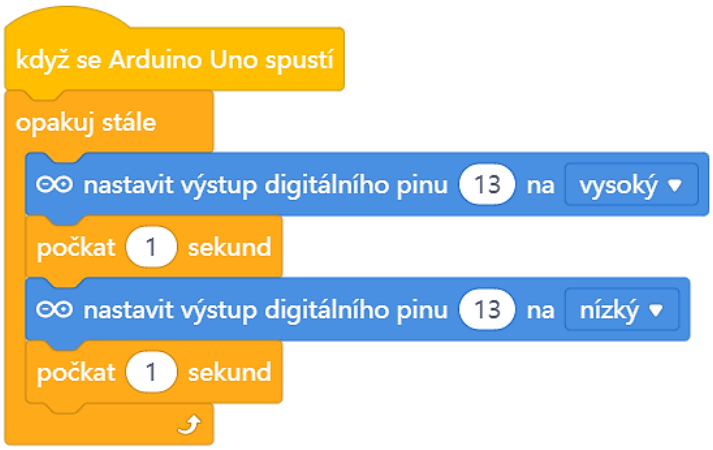
. Z této skupiny si můžeme na pracovní plochu hned vytáhnout i příkazový blok 

. Modrý příkaz 

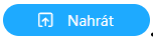
najdeme ve skupině 

. Hodnoty čísla pinu a doby čekání se zadají kliknutím na patřičné pole a zadáním čísla z klávesnice. Pro výběr výstupní úrovně digitálního pinu slouží rozbalovací nabídka.

Postupně k sobě naskládáme všechny potřebné bloky, až získáme následující náš první program.



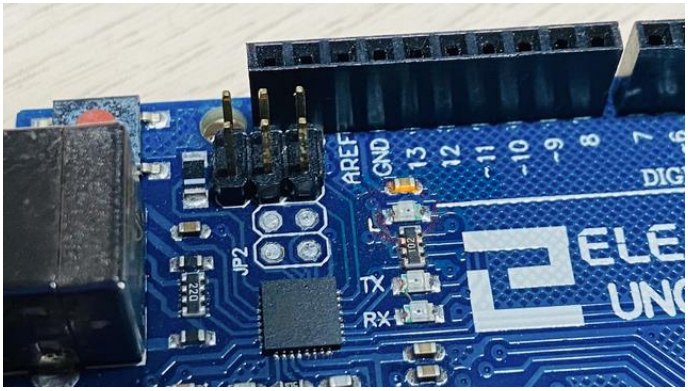
## 7. Zápis program do modulu Arduino

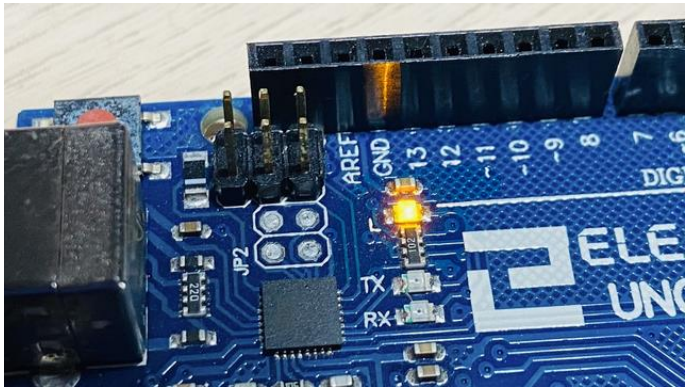
Je-li program hotový, stačí v prostředí programu mBlock jednoduše kliknout na tlačítko 

. Musíme počkat několik sekund – měli bychom vidět na modulu Arduino blikat LED označené RX a TX.

Pokud je nahrávání úspěšné, zobrazí se o tom hláška. Tím je nahrávání definitivně dokončeno! Několik sekund poté bychom měli pozorovat, jak vestavěná LED (pin 13) na desce Arduino začala blikat.

Pokud LED bliká, gratulujeme! Modul Arduino s prostředím mBlock funguje správně!



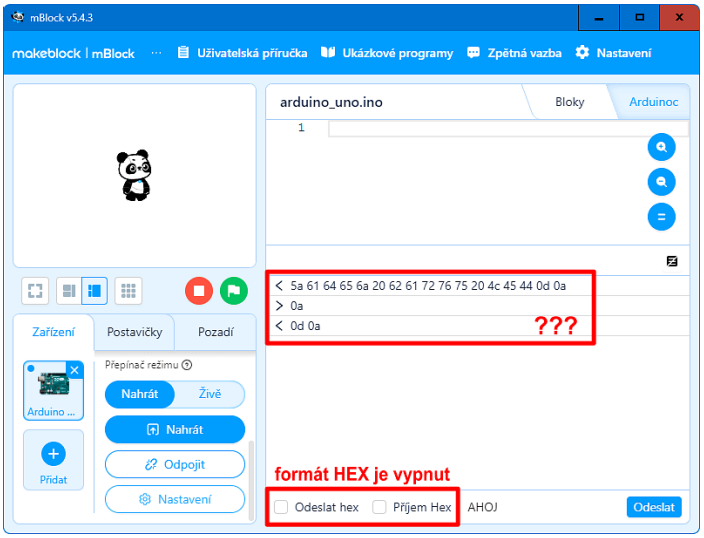
## Sériový monitor

Pokud jsme opravdu nedočkaví, mohli bychom již nyní začít programovat modul Arduino pomocí prostředí mBlock. Naše možnosti by však byly poněkud omezené. Určitě je dobré se ještě zaměřit na několik drobností, aby vše bylo kompletní a nemuseli jsme se k nim pak vracet později. Jednou z nich je sériový monitor.

Sériový monitor (Serial Monitor) je program, který budeme pro některé z dalších úkolů potřebovat a který sleduje sériový port počítače (v případě připojeného modulu Arduino USB port). Modul Arduino na tento port může vypisovat různé zprávy nebo naměřené hodnoty nebo z něj načítat hodnoty, které lze pomocí Serial Monitoru zadávat.

Dlouhou dobu byla absence sériového monitoru v prostředí mBlock určitým handicapem. Od verze 5.4.3 vývojáři prostředí mBlock sice zareagovali na hlasy volající po zavedení sériového monitoru do prostředí, ale je třeba říci, že (zatím) se tato snaha příliš nepovedla. Na sériový monitor můžeme přejít pomocí záložky Arduinoc, která se od zmíněné verze objevila v pravém horním rohu pracovní plochy. V dolní části této karty je i okno pro sériovou komunikaci. Větší ono zobrazuje zprávy, které jsou zasílané z/do modulu Arduino. Pod tímto oknem je řádek, na který lze psát zprávy pro odeslání do modulu pomocí tlačítka Odeslat.

A proč lze tedy současné pojetí integrovaného sériového monitoru považovat za nepovedené? Sériový monitor prostředí mBlock umožňuje zobrazovat vzájemnou komunikaci v hexadecimální podobě. To by mohlo být za určitých okolností přidanou hodnotou. Problém ale nastává v okamžiku, kdy tuto funkci vypneme. Místo toho, aby sériový monitor začal komunikovat v textové podobě, jak by se dalo očekávat, pokračuje tvrdošíjně v komunikaci formátu hexadecimálního (HEX). To například vidíme na následujícím obrázku. Přestože formát HEX je vypnut jak pro odesílání, tak i pro příjem, zobrazované zprávy v komunikačním okně jsou stále v hexadecimální podobě.

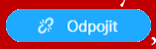



To je opravdu velká škoda! Snad se časem tato chyba podaří vývojářům opravit a prostředí mBlock bude konečně disponovat použitelným sériovým monitorem. Zatím si budeme muset stále pomoci, stejně jako bylo potřeba před verzí 5.4.3.

## Externí sériový monitor

Abychom mohli s modulem Arduino komunikovat po sériové lince (číst nebo zadávat zprávy), je stále potřeba Serial Monitor doinstalovat. Například prostředí Arduino IDE použitelný sériový monitor obsahuje, prostředí mBlok (ver. 5.4.3) zatím stále nikoliv. Je to škoda, možná se v další verzi dočkáme, ale, jak bylo dříve zmíněno, zatím si pomůžeme externím programem.

Lekce, ve kterých budeme potřebovat sledovat výstup z modulu Arduino na Serial Monitoru, budou vpravo vedle nadpisu značeny touto ikonou s lupou a USB konektorem (zde znázorněno vlevo). Program v modulu Arduino běží i bez připojeného Serial Monitoru, jen není vidět zvolený textový výstup.

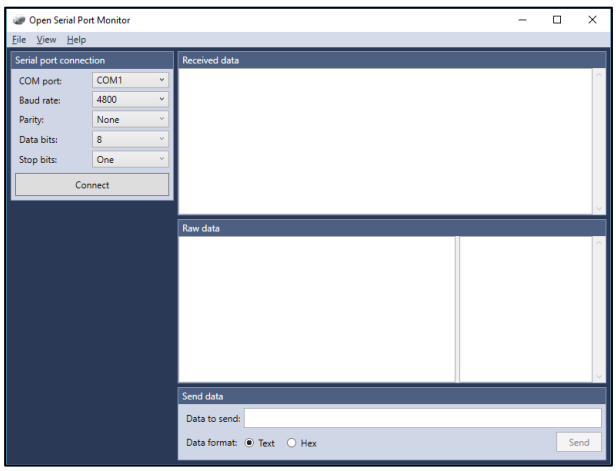
**Upozornění:**
Je třeba upozornit, že pokud budeme chtít sledovat výstup z modulu Arduino na kterémkoliv ze sériových monitorů, je potřeba předtím modul Arduino v prostředí mBlock odpojit pomocí tlačítka 

, jinak prostředí mBlock sériový monitor k USB portu modulu Arduino nepustí. K USB portu modulu Arduino může vždy přistupovat jen jedna aplikace.

Pokud hledáme nějakou jednoúčelovou aplikaci, například zde sériový monitor, stačí se podívat, zda tento problém již někdo nevyřešil před námi. Zpravidla se v podobných situacích říká: „Je možné využít několik možných aplikací, stačí jen trochu zagooglit“. I když… Pokud v tomto případě hledáme nějakou nejlépe bezplatnou aplikaci, která je navíc jednoduchá, výběr nakonec nebude bůhvíjak široký.

Abychom si ale ušetřili čas při hledání, podíváme se na některé, které jsou zdarma ke stažení a které dostatečně splní naše požadavky.

1.  **Open Serial Port Monitor** od vývojáře G. T. Hvidstena je pro milovníky „portable“ aplikací, které není třeba nijak instalovat. Stačí jen stáhnout EXE soubor, který se pak rovnou spustí. Výhodou této aplikace je především její jednoduchost. Aplikace zobrazuje výstup jak v podobě textového okna, tak v okně s hexadecimálním zobrazením.

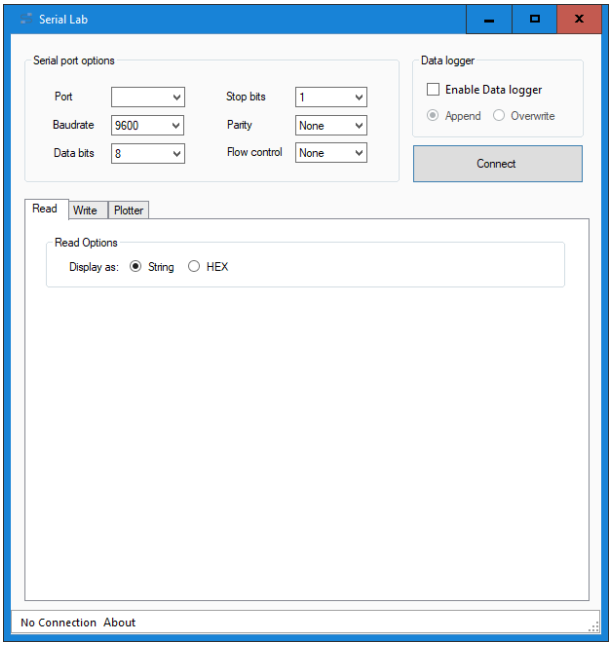


Je třeba jen dodat, že aplikace Open Serial Port Monitor využívá pro svůj běh aplikační rámec Microsoft .NET 4.5. Ten je ale již od Windows 8 běžně předinstalovanou součástí systému, takže by běh aplikace neměl být problém.

Popřípadě je třeba tento aplikační rámec doinstalovat ze stránek společnosti Microsoft.

**Odkaz pro stažení:**
https://github.com/gthvidsten/open-serial-port-monitor

2. **Serial Lab** od vývojáře Ahmeda El-Sayeda je pro milovníky normálních aplikací, u kterých je třeba nejdříve stáhnout instalační soubor, který se pak spustí. I tato aplikace je poměrně jednoduchý jednoúčelový program, který může výborně posloužit pro naše účely. Přidanou hodnotou této konkrétní aplikace oproti předchozí je to, že kromě textového výstupu obsahuje i možnost vynášení hodnot do grafu (tzv. Serial Plotter). Pro dále prezentované lekce to asi není potřeba, ale později při vývoji svých vlastních a složitějších projektů se to může hodit. Serial Lab lze stáhnout z níže uvedeného odkazu na GitHub.




**Odkaz pro stažení:**
https://github.com/ahsayde/Serial-Lab

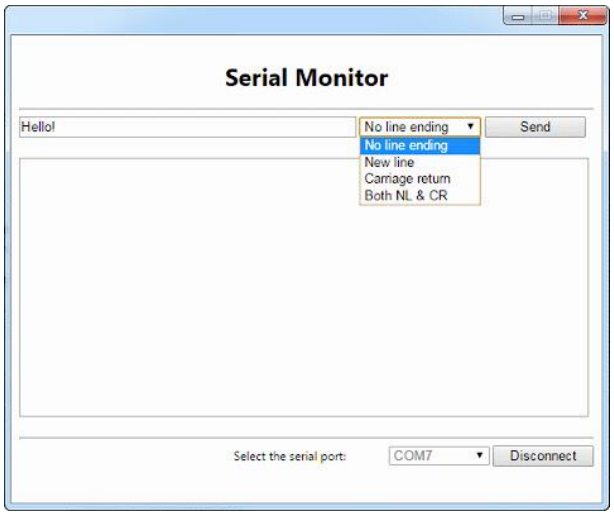
Následující dva odkazy na aplikace sériových monitorů nejsou odkazy na klasické aplikace, které se spouští v operačním systému, ale jedná se o doplňky webového prohlížeče Google Chrome. Proto také budeme tyto aplikace instalovat z internetového obchodu Chrome. Ale pozor! Nejedná se o klasické rozšíření webového prohlížeče, které se spustí z lišty prohlížeče. Na první pohled se chovají jako samostatné aplikace, které však pro svůj běh potřebují prostředí webového prohlížeče Chrome na pozadí. Takové aplikace by mohly být volbou pro majitele operačního systému, který není Windows, ale umožňuje běh prohlížeče Google Chrome včetně instalovaných rozšíření a jeho aplikací.

**Upozornění:**
Společnost Google od ledna 2024 všechna rozšíření a aplikace pracující na platformě Manifest V2 zcela odstranění z internetového obchodu Chrome. Bohužel se to zatím týká i obou dále zmíněných aplikací sériových monitorů. Takže pokud by se vývojáři Glen Arrowsmith a Luca Dentella nerozhodli povýšit své sériové monitory na novou platformu Manifest V3, nebudeme moci následující aplikace již nadále používat! 😒

3.  **Serial Port** Monitor od vývojáře Glena Arrowsmitha je, jak bylo naznačeno, aplikací pro prohlížeč Google Chrome. Najdeme ji proto na níže uvedeném odkazu v internetovém obchodě Chrome. Instalaci provedeme kliknutím na tlačítko „Přidat do Chromu“, ale pozor Serial Port Monitor se neobjeví jako rozšíření prohlížeče, je to samostatná aplikace, bude uvedená normálně mezi ostatními programy.


**Odkaz pro stažení:**
https://chrome.google.com/webstore/detail/serial-monitor/ohncdkkhephpakbbecnkclhjkmbjnmlo

4.  **Serial Port Monitor** od vývojáře Luca Dentella je také aplikací webového prohlížeče pro Google Chrome, takže ji opět získáme v internetovém obchodě Chrome. Instalaci opět provedeme kliknutím na tlačítko „Přidat do Chromu“ a opět ji ale nehledejme v prohlížeči, ale jako samostatnou aplikace mezi ostatními nainstalovanými programy.




**Odkaz pro stažení:**
https://chrome.google.com/webstore/detail/serial-monitor/aegabhjcjoamaadpaegobifbdodfecil

# Rozšíření „MAXI Starter kit“

Jelikož sada Arduino MAXI Starter kit je doslova „nabušena“ mnoha součástkami, jejichž ovládání není vždy jednoduché, vytvořili jsme pro prostředí mBlock rozšíření nazvané MAXI Starter kit, které by ovládání těchto součástek mělo výrazným způsobem zjednodušit. Rozšíření totiž přidá do prostředí mBlock na svislý sloupec příkazů další skupiny příkazových bloků, které jsou tříděné do tematických kategorií – např. příkazy pro LCD displej, servo, krokový motor apod.

Rozšíření MAXI Starter kit je nutné jen pro některé z dále uvedených lekcí. Tam, kde bude třeba použít některé jeho funkce, upozorníme na to u zápisu programu následující ikonou (zobrazena vpravo). Pokud nyní nechceme nějaké rozšíření řešit a chceme se okamžitě pustit do programování, musíme si vybrat lekce bez tohoto označení.

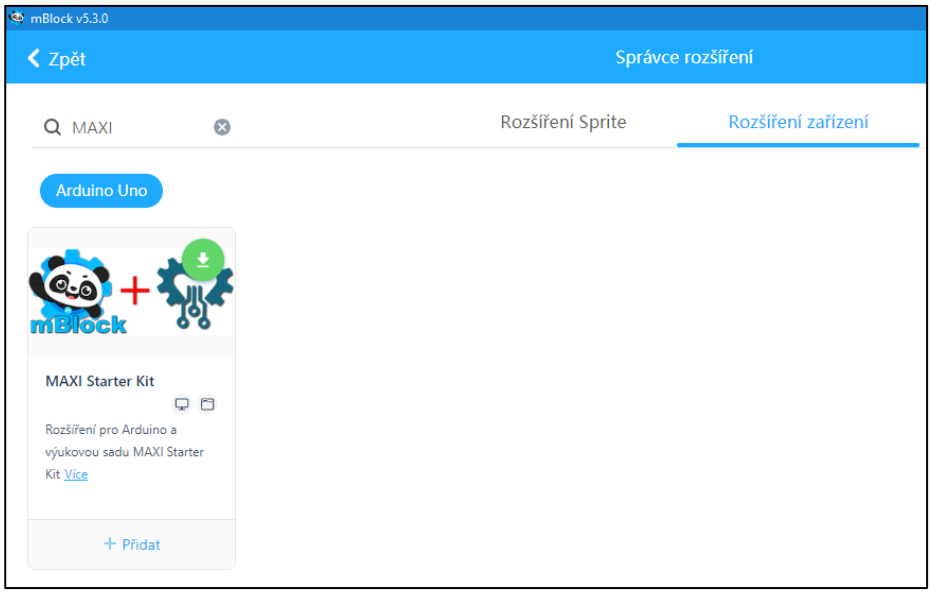

## Přidání rozšíření MAXI Starter kit do prostředí mBlock

S těmi, kteří se nenechali vylekat tím, že nás ještě čeká nějaké nastavení a chtějí se pustit i do lekcí s rozšířením MAXI Starter kit, si společně ukážeme, jak toto rozšíření do prostředí mBlock přidat.

Na svislé liště příkazů je dole i ikona nazvaná „rozšíření“  . Když na tuto ikonku klikneme, otevře se průvodce „Správce rozšíření“, který slouží k získání a přidání dalších rozšíření do aktuálního projektu. Jsou to vlastně knihovny, které nám přidají do sloupce příkazů další příkazové bloky. Najdeme tu všechno možné od knihoven pro ovládání displejů, tepelných čidel, ale i pro celé shieldy nebo roboty. To vše je k dispozici nejen od vývojáře prostředí mBlock, ale i od ostatních bastlířů, třeba jako jsme my.

My však nyní potřebujeme najít naše rozšíření MAXI Starter kit. Nejrychlejší volba je pomocí vyhledávacího pole. Stačí nahoře do vyhledávacího pole napsat „MAXI“ a je-li v tu chvíli počítač připojen k internetu, zobrazí se seznam všech dostupných rozšíření obsahující v názvu výraz MAXI. To je zatím jen to naše, takže se nám zobrazí jen požadované MAXI Starter kit. Zelená ikona v pravém horním rohu ukazuje, že toto rozšíření je zatím k dispozici na internetu a je třeba jej nejdříve do počítače stáhnout (viz následující obrázek).

Po prvním kliknutí na volbu `+Přidat` se rozšíření stáhne do počítače. Jakmile se jednou do počítače stáhne, již se může vkládat a používat i bez připojení k internetu.




Po stažení, přidání rozšíření do projektu a zavření Správce rozšíření se na liště příkazů objeví nové skupiny příkazů. Všechny tyto skupiny příkazových bloků jsou pojmenovány dle svého použití a jsou uvozeny různobarevnou ikonou symbolu firmy LASKARDIUNO .

**POZNÁMKA:**
Jen je potřeba upozornit, že při novém otevření prázdného projektu jsou na listě příkazů zase jen standardní příkazy a jakékoliv rozšíření je třeba opět přidat výše popsaným způsobem prostřednictvím Správce rozšíření. Pokud je však v projektu použit jakýkoliv příkaz daného rozšíření, načte se na svislou lištu příkazů již toto rozšíření automaticky.

## Příkazové bloky rozšíření MAXI Starter kit (ver. 1.0.3)

Celé rozšíření je rozděleno do třinácti tematických celků, které respektují jednotlivá zaměření na danou problematiku nebo funkční blok. Kategorie rozšíření se svými příkazovými bloky se v prostředí mBlock zobrazují na střední svislé příkazové liště. Na následujících řádcích si představíme jednotlivé kategorie a jejich příkazové bloky. Podrobněji se s nimi pochopitelně seznámíme v jednotlivých praktických lekcích v druhé polovině tohoto návodu.

1. Nastavení pinů

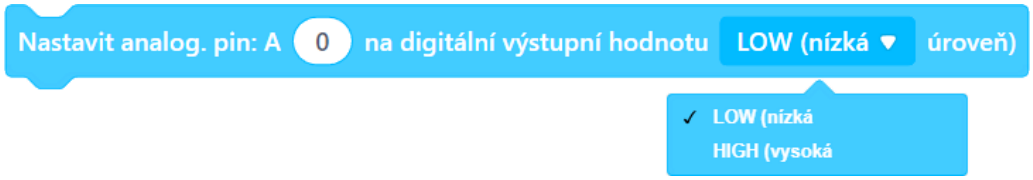
   Nastavení digitálního pinu do režimu PULLUP a jeho následné čtení.

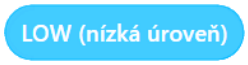
   Nastavení pinu A (analogový) do režimu PULLUP a jeho následné čtení.

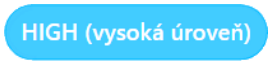
Nastavení pinu A (analogový) do digitálního režimu a nastavení vysoké/nízké úrovně (HIGH/LOW).




digitální nízká úroveň (LOW)

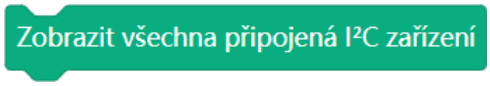



digitální vysoká úroveň (HIGH)



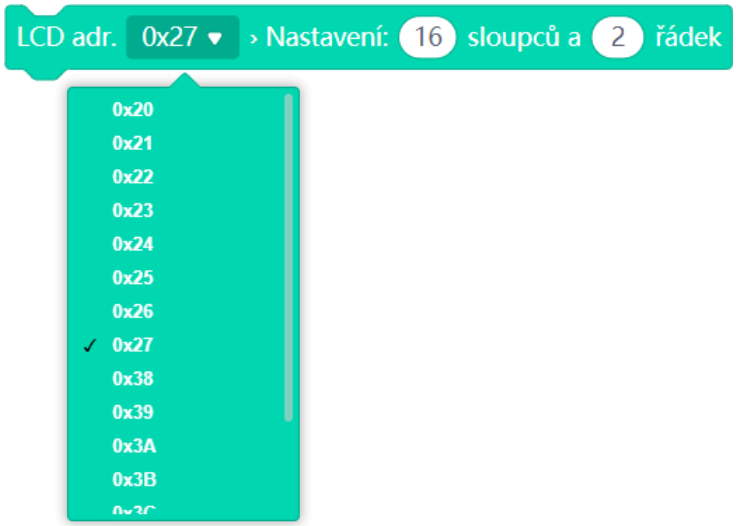

## 2. I²C skener

Na sériovém monitoru bude uveden seznam všech I²C zařízení připojených k modulu Arduino.

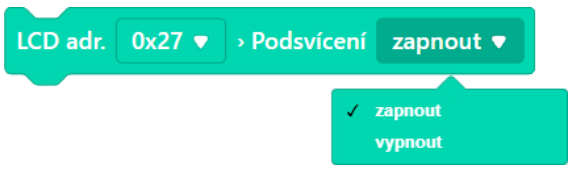



## 3. LCD I²C displej

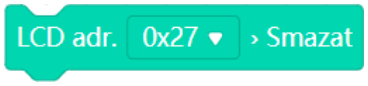
Nastavení dimenze LCD displeje (sloupce/řádky) zadané I²C adresy.



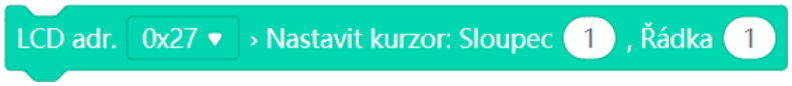

Zapnutí/vypnutí podsvícení LCD displeje zadané I²C adresy.



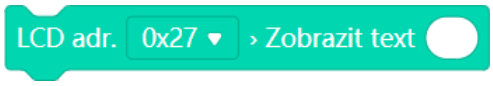
Smazání obsahu LCD displeje zadané adresy I²C.




Nastavení kurzoru na zadaný sloupec, pozice řádku na LCD displeji zadané I²C adresy.

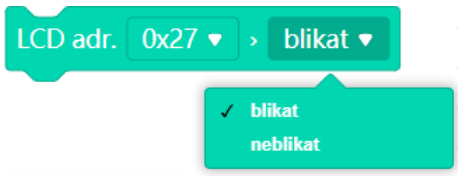



Zobrazení zadaného textu na aktuální pozici kurzoru na LCD zadané I²C adresy.

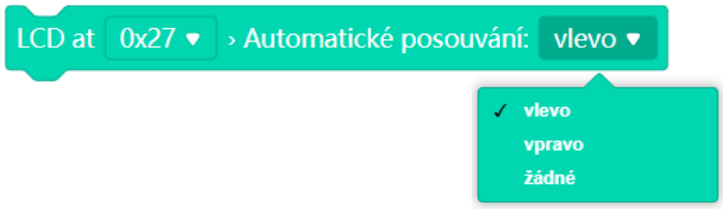



Nastavení směru zobrazeného textu (zprava doleva nebo zleva doprava) LCD displeje zadané I²C adresy.

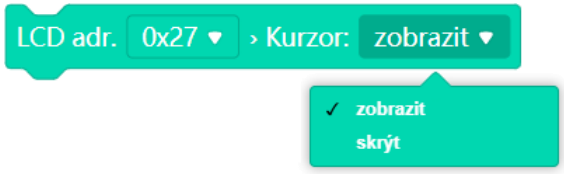

Nastavení blikání LCD displeje zadané I²C adresy.




Nastavení možnosti rolování textu (vlevo, vpravo, žádné) na LCD displeji zadané I²C adresy.




Nastavení zobrazení kurzoru (zobrazit/skrýt) na LCD displeji zadané I²C adresy.




### 4. Textové proměnné

Nastavení obsahu textové proměnné (pro odlišení jsou proměnné číslovány).

Získání obsahu dané (zadané číslem) textové proměnné.


Nahrazení výrazu „text1“ výrazem „text2“ v zadané textové proměnné.

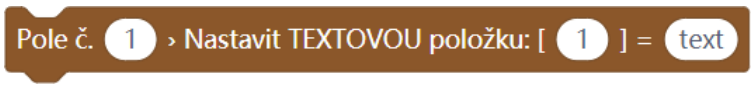

Získání podřetězce ze zadané textové proměnné. Podřetězec je určen pořadovým číslem prvního znaku a jeho celkovou délkou.


## 5. Pole proměnných

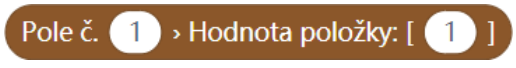
Deklarace pole číselných/textových proměnných (pro odlišení jsou pole číslovány). V této deklaraci musí být specifikován počet proměnných. Je zadán počet proměnných pole, nikoli nejvyšší index pole! První položka pole má index 0 (*pole s indexy 0 a 1 již má dva prvky!*)


Vloží zadanou číselnou hodnotu do zvolené proměnné v daném poli.


Vloží zadanou textovou hodnotu do zvolené proměnné v daném poli.




Získání obsahu dané položky vybraného pole.




## 6. Posuvný registr

Nastavení ovládacích pinů (Latch, Data, Clock) posuvného registru.

Volba směru odesílání bajtu do posuvného registru (MSBFIRST – *první nejvýznamnější bit*, LSBFIRST – *první nejméně významný bit*).

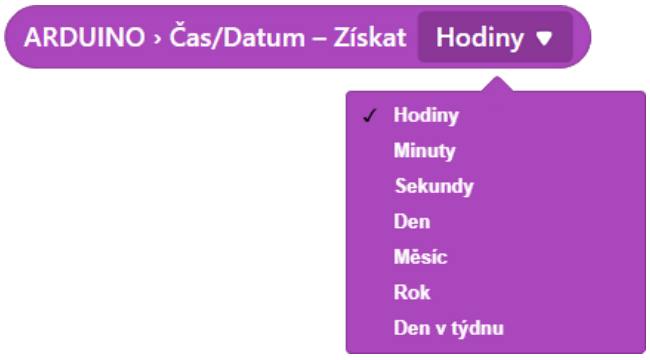

7. ## Čas/Datum DS1302RTC

Nastavení ovládacích pinů modulu reálných hodin DS1302RTC.

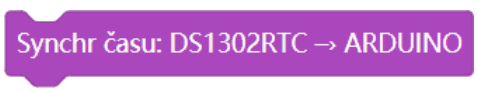

Nastavení času a data na modulu DS1302 RTC. Formát dat je: HH:MM:SS/DD-MM-20YY.


Nastavení času a data na modulu Arduino. Formát dat je: HH:MM:SS/DD-MM-20YY.


Získání jednoho z vybraných údajů (hod, min, ..., rok, den v týdnu). Hodnoty jsou čteny z modulu Arduino.

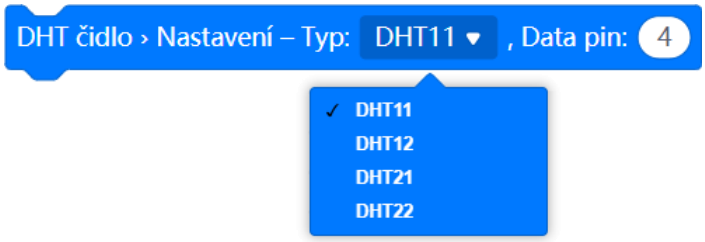



Synchronizace časových údajů z modulu DS1302RTC do modulu Arduino. Lze použít například po spuštění modulu Arduino pro načtení aktuálních hodnot z modulu reálných hodin (RTC), který je zálohován baterií.

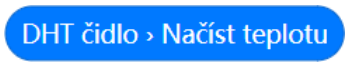


## 8. DTH čidlo (čidlo teploty a vlhkosti vzduchu)

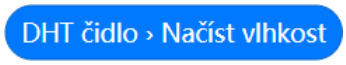
Nastavení typu DHT čidla a volba jeho datového pinu.




Načtení aktuální hodnoty teploty z DHT modulu.



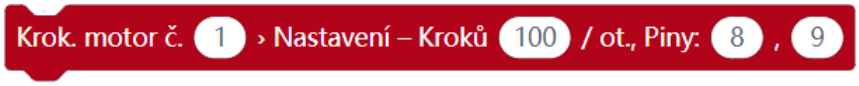

Načtení aktuální hodnoty relativní vlhkosti z DHT modulu.




## 9. Krokový motor

Inicializace 4-vodičového krokového motoru – nastavení identifikačního čísla motoru, počtu kroků na otáčku a nastavení ovládacích pinů.


Inicializace 2-vodičového krokového motoru – nastavení identifikačního čísla motoru, počtu kroků na otáčku a nastavení ovládacích pinů.




Inicializace 5-vodičového krokového motoru – nastavení identifikačního čísla motoru, počtu kroků na otáčku a nastavení ovládacích pinů.

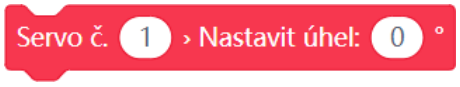

Nastavení rychlosti otáčení (ot/min) daného krokového motoru (*POZOR! Nastavuje se rychlost otáčení kotvy krokového motoru – některé krokové motory jsou osazeny převodovkou, která dle zadaného poměru snižuje rychlost otáčení výstupního hřídele*).

Vykonání zadaného počtu kroků daného krokového motor. Kladný počet kroků udává jeden směr otáčení, zadáním záporného počtu kroků se vykoná daný počet kroků v opačném směru otáčení.

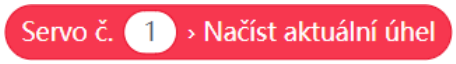

### 10. Servo

Inicializace serva – nastavení identifikačního čísla serva, volba ovládacího pinu.


Nastavení úhlu natočení pro servo daného čísla.

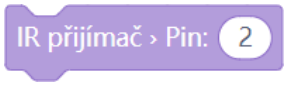



Načtení aktuální polohy natočení daného serva.

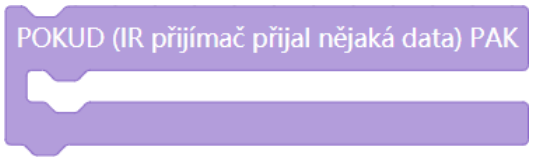



### 11. IR přijímač

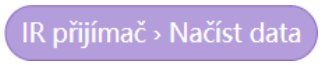
Inicializace infračerveného přijímače – nastavení datového pinu.




Programový blok, který se provede v okamžiku, kdy IR přijímač přijme nějaká data.




Čtení dat přijatých IR přijímačem.

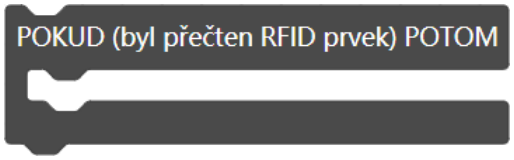



### 12. RFID čtečka

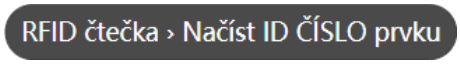
Inicializace RFID modulu – nastavení ovládacích pinů (IRQ, SCK, MOSI, MISO, SDA, RTS).

Programový blok, který se spustí, když modul RFID načte RFID prvek.


Programový blok, který se spustí, když modul RFID načte RFID prvek.

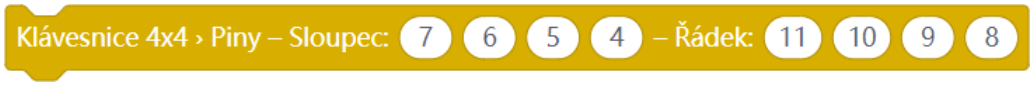



Načtení identifikačního čísla čteného RFID prvku.




## 13. Klávesnice

Nastavení ovládacích pinů klávesnice 4×4 (sloupce × řádky).

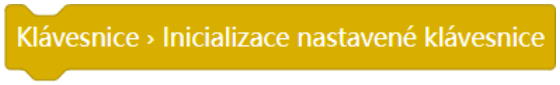



Nastavení ovládacích pinů klávesnice 3×4 (sloupce × řádky).


Nastavení ovládacích pinů klávesnice 3×3 (sloupce × řádky).


Předefinování znaku klávesnice na pozici sloupce/řádku.

Začátek používání nastavené a definované klávesnice.




Skenování stavu klávesnice. Po tomto příkazu je možné načíst, zda byla stisknuta klávesa, popřípadě která klávesa byla stisknuta.


Logický výraz, který říká, zda byla stisknuta nějaká klávesa.

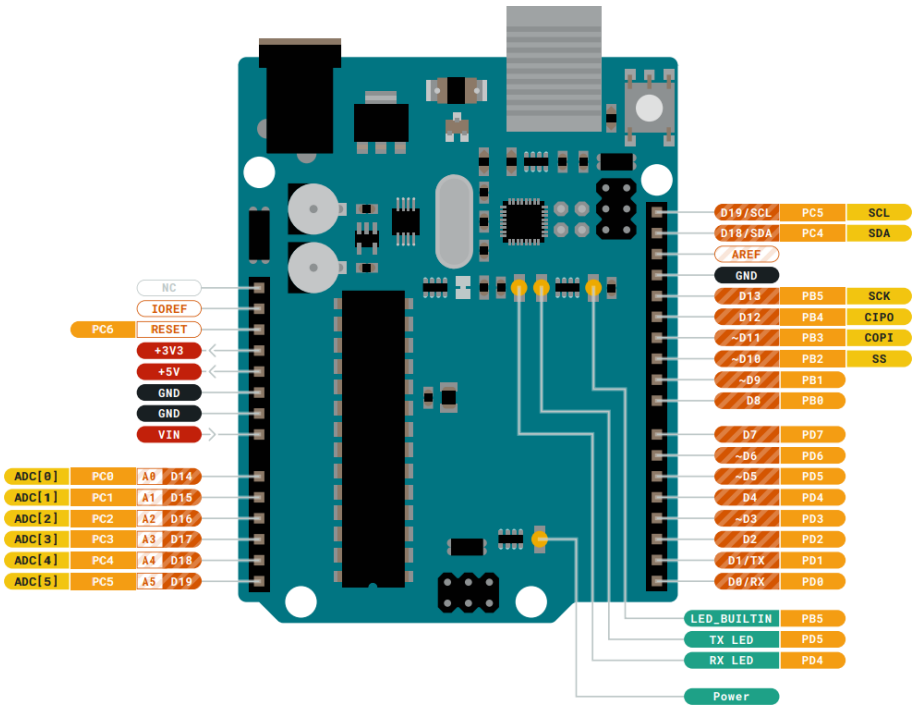

Načtení symbolu klávesy, která byla stisknuta.

# Jak používat modul ARDUINO

I když jsme se v předchozí kapitole tvářili, že se co nejdříve pustíme do „užitečné“ práce, je třeba si uvědomit, že ne každý bastlíř, který si pořídil sadu Arduino MAXI Starter kit, je ostříleným uživatelem modulu Arduino. Takže pokud již máme s modulem Arduino nějaké zkušenosti, známe například rozložení a funkce jednotlivých pinů na konektorech modulu Arduino, nedělá nám problém chápat rozdíl mezi funkcemi digitálních a analogových vstupů a výstupů, je možné následující kapitolu klidně přeskočit.

## Vstupně/výstupní piny modulu Arduino

Modul Arduino nabízí několik digitálních i analogových pinů, které můžeme ve svých projektech použít k libovolnému účelu. Většinu těchto pinů lze použít jak pro vstup, tak i pro výstup. Kromě tohoto univerzálního použití mají některé z nich i další speciální funkce, které lze jednoduše aktivovat ve zdrojovém kódu. Postupně si ukážeme, jak se s jednotlivými piny pracuje a zkusíme si ukázat i jejich další funkce. Následující obrázek ukazuje rozložení pinů na konektorech modulu Arduino UNO, které je s největší pravděpodobností součástí sady Arduino MAXI Starter kit.




## Digitální vstupy a výstupy

Jak už bylo naznačeno, téměř všechny piny modulu Arduino lze použít jako digitální vstup nebo výstup. Jejich směr lze přepnout zavoláním potřebné funkce. Jelikož se jedná o digitální piny, napětí na nich je modulem Arduino interpretováno pomocí dvou logických hodnot – HIGH a LOW (vysoká/nízká úroveň), nebo i „Pravda“ a „Nepravda“ (nebo také uváděno jako „logická 1“ a „logická 0“).

Modul Arduino pracuje s tzv. *pozitivní TTL logikou*, což znamená, že vysoká úroveň (HIGH) je reprezentována napětím blízkým napájecímu napětí (v případě verze UNO to je 5 V) a nízká úroveň (LOW) je napětí blízké 0 V. Tyto hodnoty napětí lze na výstup nastavit, nebo je naopak na vstupu číst.

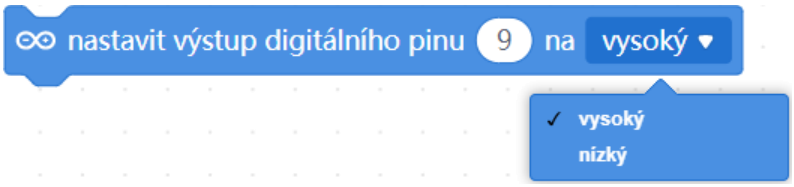
**POZNÁMKA:**
Maximální provozní (vstupní/výstupní) proud digitálních pinů je 40 mA. Doporučuje se však udržovat tento proud pod hranicí 20 mA!

Obecně se u modulu Arduino uvádí, že je možné dokonce jeden pin chvíli používat jako vstup a poté i jako výstup. Upřímně řečeno, to zrovna není standardní a často používaná technika, ale občas by se to třeba mohlo hodit. Na druhou stranu si ale okamžitě musíme říci, že zrovna tuto věc si v prostředí mBlock bohužel musíme odpustit! V případě našeho blokového programování nelze za běhu programu měnit nastavení funkce pinů. Jakmile se na pin dotážeme, bude pro celý program nastaven jako vstup. Pokud na něj zapíšeme, bude nastaven (opět pro celý program!) jako výstup. Pokud bychom se v jednom programu pokusili některý pin použít jako vstup i jako výstup, skončí program chybou. To je bohužel omezení, které nám přináší pohodlí blokového programování modulu Arduino v mBlocku.

### *Výstupy*

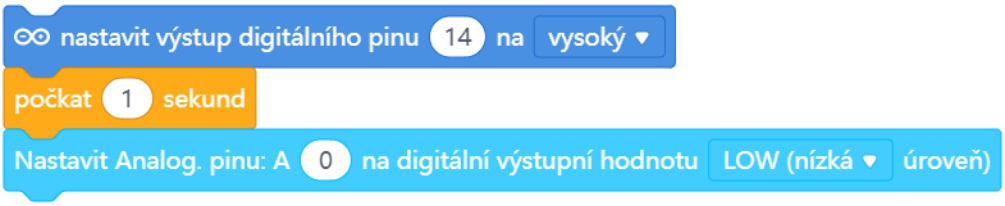
Pro digitální výstupy lze běžně používat piny z pravého konektoru modulu Arduino, které jsou na předchozím obrázku rozložení pinů na konektorech modulu Arduino označeny písmenkem D. Pro zápis digitálních výstupů je v prostředí mBlock pochopitelně připraven příkazový blok.

Pokud bychom v mBlock chtěli na digitální pin číslo 9 zapsat výstupní digitální hodnotu HIGH/LOW (vysoká/nízká úroveň), provedli bychom to takto:




V případě, že by bylo třeba více digitálních výstupních pinů, než nabízí pravý konektor modulu Arduino, je možné použít i analogové vstupy A0–A5 (levý konektor). Práce s nimi je úplně stejná jako s digitálními vstupy/výstupy, pouze s tím rozdílem, že se adresují čísly 14–19. Druhou možnost, jak nakonfigurovat vstupní analogové piny A0–A5 jako digitální výstupy a zapisovat na ně, nabízí programový blok z našeho rozšíření MAXI Starter kit.

Následující obrázek ukazuje obě možnosti ovládání pinu A0 jako digitálního výstupu. První (tmavomodrý) blok ukazuje standardní blok prostředí mBlock, který nastavuje pin A0 (čili 14) na hodnotu vysoké úrovně (HIGH). Následující blok (světle modrý) je programovým blokem z rozšíření MAXI Starter kit a nastavuje stejný pin na výstupní digitální úroveň LOW (nízká úroveň).




### *Vstupy*

Obdobně předchozímu můžeme digitální informaci též načítat. Příkladem nám může být test stisknutého tlačítka, přerušení světelné závory apod. Pro čtení digitálních výstupů je opět v prostředí mBlock příkazový blok.

Například digitální pin číslo 9 načteme v prostředí mBlock následujícím blokem:

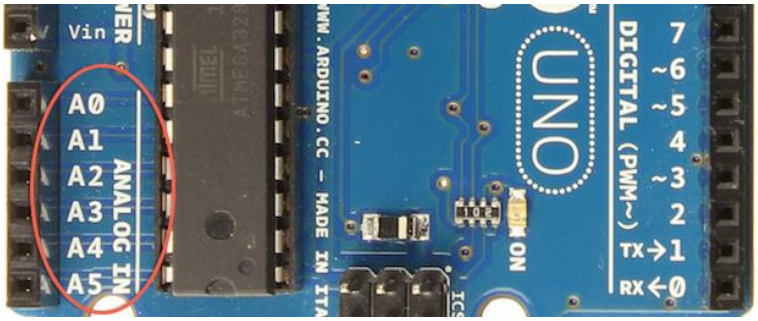

Je však třeba upozornit, že u modulu Arduino mohou být k dispozici při nastavení digitálního vstupu dva základní režimy:

- **INPUT** – vstup detekuje takovou hodnotu, která je na příslušný pin připojena. Jinými slovy, logická úroveň na nezapojeném vstupu není definována. Pokud chceme pomocí tohoto režimu načítat stav tlačítka, musíme při jeho daných stavech (stisknuto/nestisknuto) nějak elektricky vyřešit správnou hodnotu napětí na kontrolním vstupu.
- **INPUT_PULLUP** – vstup je „držen“ interním pull-up rezistorem na hodnotě 5 V. Toto nastavení způsobí, že uvnitř modulu Arduino se zapojí rezistor mezi napájecí napětí a zadaný pin. Není-li tedy na vstupu nic připojeno, je vstup nastaven na vysoké úrovni (HIGH). Při přivedení nízké úrovně (0 V) na vstup, je vstup „stažen“ na úroveň 0 V a vstupní hodnota je detekována jako nízká úroveň, čili hodnota LOW. Zároveň vnitřní pull-up rezistor zabraňuje zkratu, ke kterému by jinak při přímém setkání úrovní HIGH (5 V) a LOW (0 V) došlo.

V případě standardního příkazového bloku pro načítání digitálního pinu v prostředí mBlock pracuje jen v režimu INPUT, ale opět naše rozšíření MAXI Starter kit přináší programové bloky pro nastavení digitálních (a i analogových pinů A0–A5) do režimu INPUT_PULLUP a jejich načítání. Rozdíl mezi těmito dvěma nastaveními a možnost zjednodušení elektronického zapojení díky konfiguraci INPUT_PULLUP si ukážeme hned v první lekci: Řízení LED tlačítkem.

## Analogové vstupy

Ne vždy si v elektronice vystačíme jen s digitálními signály. Relativně často je potřeba pracovat s analogovým (spojitým) signálem. Pro analogový vstup lze využít piny označené jako A, kterých je celkem šest (A0–A5). Na těchto vstupech je 10bitový analogově digitální převodník, který dokáže převádět napětí v rozsahu 0 až 5 V na digitální číslo (0–1023) se kterým pak můžeme v našem projektu dále pracovat.

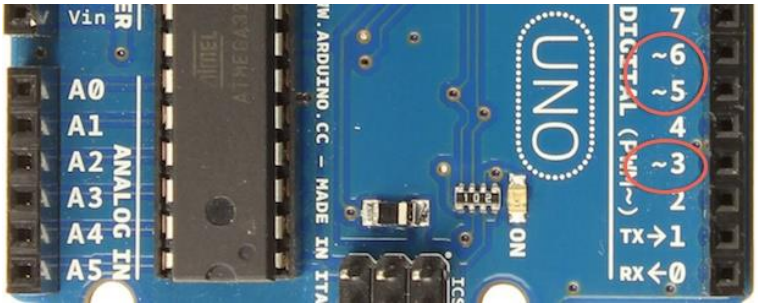



Čtení analogových vstupů pomocí prostředí mBlock je obdobné jako načítání vstupů digitálních. Jediným rozdílem je to, že získaná hodnota nebude mít jen dvojici možných stavů LOW a HIGH, ale celou množinu možných čísel z daného rozsahu. Takže načtením analogové hodnoty lze měřit různé úrovně napětí mezi 0 V a 5 V. Pomocí standardního příkazového bloku prostředí mBlock to provedeme takto:

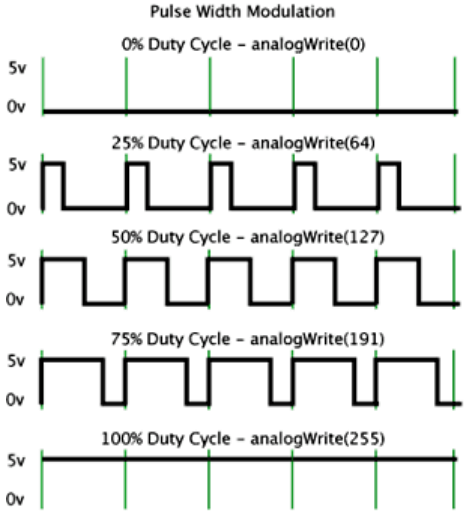

Načítání analogového vstupu si ukážeme v lekcích: Fotorezistor, Ovládání zvuku pomocí světla, Voltmetr, ale i v dalších.

## Výstupy PWM

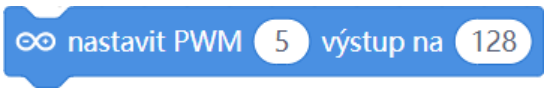
PWM (tedy „pulzně šířková modulace“) výstup je jakýmsi přechodem mezi digitálním a analogovým výstupem. Modul Arduino bohužel žádným čistě analogovým výstupem nedisponuje, ale PWM jej občas může nahradit. V případě PWM jde o cyklické přepínání digitálního výstupu mezi stavy LOW a HIGH. Podle toho, jaká je „šířka“ stavu vysoké úrovně (HIGH) ku šířce nízkého stavu (LOW), dochází k výslednému stavu na daném výstupu. I když se tedy jedná o digitální výstup, kdy na výstupu jsou jen stavy LOW (0 V) a HIGH (5 V), ve svém důsledku se výstup chová, jakoby se na něm měnilo napětí analogově. U modulu Arduino UNO může fungovat jako PWM výstup šest digitálních výstupů, jsou to piny D3, D5, D6, D9, D10 a D11. Na obrázku zachycujícím jednotlivé piny modulu (nebo na jeho potisku) tyto piny poznáme podle toho, že je před jejich označením uvedena značka vlnovky (~).




K PWM výstupu je třeba ještě říci, že jeho rozlišení (tedy rozsah výstupních hodnot) je 0–255. Následující obrázek ukazuje průběh výstupních HIGH/LOW stavů na některém z PWM výstupů pro různé výstupní hodnoty.

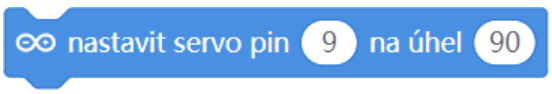



Standardní frekvence generovaného PWM signálu modulu Arduino UNO je 490 Hz (na pinech 5 a 6 je frekvence 980 Hz). Pro nastavení PWM výstupu je v prostředí mBlock následující příkazový blok:

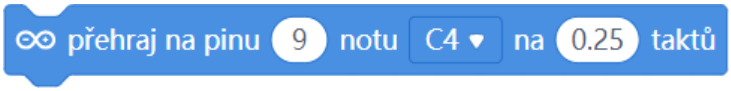



Ukázku PWM výstupu si ukážeme například v lekci nazvané: „Řízení LED pomocí PWM“.

Je třeba zmínit, že PWM signálu se také využívá při řízení serva (uvidíme v lekci Servo). I když tam si tohoto jako programátoři ani moc nevšimneme, protože PWM signál pro nás zůstane skryt za příkazovým blokem rozšíření MAXI Starter kit. I tak je ale asi dobré aspoň trochu tušit, jak se PWM dá využít.




A nakonec tu máme ještě jedno využití PWM, a to v případě generování tónů. Prostředí mBlock, které je primárně určeno pro ovládání robotů mBot, potřebuje ovládat i zvukové výstupy těchto robotů. Je zde tedy i programový blok pro generování potřebného tónu po zadanou dobu:




V několika následujících lekcích se zabýváme bzučáky, které jsou v sadě Arduino MAXI Starter kit obsaženy. Zpravidla budeme používat aktivní bzučák, který generuje svůj tón sám, ale jakmile budeme potřebovat změnit tón generovaného tónu, pomůže nám PWM a bzučák pasivní. To kupříkladu uvidíme v naší bonusové lekci.

## Speciální funkce některých pinů

Kromě výše popsaných nastavení a funkcí vstupně/výstupních pinů mají některé piny modulu Arduino své speciální funkce. Některé budeme využívat, například pro připojení displeje, další zatím nevyužijeme, ale je dobré o nich vědět, abychom občas předešli problémům.

### Sériová komunikace

Přestože v našich lekcích budeme vždy využívat komunikaci mezi modulem Arduino a připojený počítačem jedině prostřednictvím kabelu USB, je dobré vědět, že modul Arduino může sériově komunikovat i prostřednictvím svých digitálních pinů. V případě modulu Arduino UNO to jsou digitální piny 0 (RX) a 1 (TX). Piny 0 a 1 můžeme používat pro sériovou komunikaci s dalším sériovým zařízením.

**ALE POZOR:** Připojení čehokoli k těmto pinům (pin 0 a 1) může narušit sériovou komunikaci, včetně způsobení neúspěšného nahrávání programu do desky. To je dobré vědět, takže pokud to půjde, využijeme pro zapojení k Arduino jiné piny!

I když se v našem tutoriálu sériovou komunikací zabývat nebudeme, třeba na ni někdy narazíme později v jiných projektech. Tak zde jen pro zajímavost uvedeme následující: Chcete-li použít sériový port modulu Arduino pro komunikaci s externím sériovým zařízením TTL, připojuje se pin TX (pin 1 modulu Arduino) k pinu RX připojeného zařízení, pin RX (pin 0 modulu Arduino) ke kolíku TX připojeného zařízení. Pochopitelně je třeba ještě propojit kolíky GND (zem) obou zařízení.

### SPI komunikace

SPI (Serial Peripheral Interface) je synchronní sériový datový protokol používaný mikrokontroléry pro rychlou komunikaci s jedním nebo více periferními zařízeními na krátké vzdálenosti. I modul Arduino má možnost této komunikace. Značnou nevýhodou SPI komunikace je to, že její standard je značně volný, takže každé zařízení jej může implementovat trochu jinak. To znamená, že při psaní kódu rozhraní musíte věnovat zvláštní pozornost datovému listu připojeného zařízení. Obecně řečeno, existují tři režimy přenosu. Tyto režimy určují, zda se data posouvají dovnitř/ven při vzestupné nebo sestupné hraně hodinového signálu, nebo zda jsou hodiny nečinné, když jsou na vysoké nebo nízké úrovni.

Výhodou je, že pro většinu standardních periférií jsou již připravené knihovny, které jsou implementovány do programových bloků prostředí mBlock nebo nějakého jeho rozšíření. Kromě řídicích příkazových bloků by při této komunikaci měl být k dispozici i blok konfigurační, který nastavuje piny pro jednotlivé role komunikačních linek – například jako vidíme na následujícím obrázku, kde je konfigurační blok pro komunikaci s posuvným registrem.


Pokud pro danou komunikaci takové nastavení není, zpravidla se počítá s určitým standardním nastavení pinů modulu Arduino, což bývá:

| pin Arduino UNO | Označení | Popis |
|---|---|---|
| 13 | SCK<br>(Serial Clock) | Hodinový signál |
| 12 | CIPO<br>(Controller In Peripheral Out) | Vstup řadiče, výstup periférie |
| 11 | COPI<br>(Controller Out Peripheral In) | Výstup řadiče, vstup periférie |
| 10 | CS<br>(Chip Select) | Výběr (zapnutí) periférie |

A aby toho nebylo málo, je tu ještě jedna komplikace. Dříve se tyto vodiče označovaly jinými názvy, takže u některých periférií se ještě můžeme setkat s jinými zkratkami. Tak, abychom v tom měli jasno, v následující tabulce je srovnání staršího a dnešního označení.

| Dříve | Nyní |
|---|---|
| MISO<br>(Master In Slave Out) | CIPO<br>(Controller In Peripheral Out) |
| MOSI<br>(Master Out Slave In) | COPI<br>(Controller Out Peripheral In) |
| SS<br>(Slave Select pin) | CS<br>(Chip Select) |

## I²C komunikace

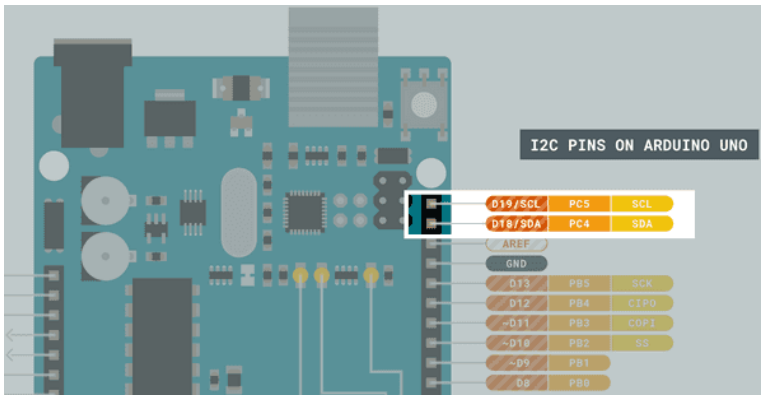
Nejdříve se podíváme, co to ta komunikace I²C vlastně je. Označení pochází z názvu „Inter-Integrated Circuit“ a tajemnou značku I²C čteme: „I-squared-C“. Mimochodem, prakticky identická sběrnice se skrývá i pod zkratkou TWI (Two Wire Interface – dvoudrátové rozhraní), kterou používá firma Atmel a další výrobci namísto chráněné značky I²C, kterou vyvinula firma Philips. Sběrnice je sériová a umožňuje propojení až 128 různých zařízení jen pomocí dvou obousměrných vodičů. Jeden tvoří hodinový signál SCL (Synchronous Clock) a druhý datový kanál SDA (Synchronous Data). Zařízení připojené na sběrnici se rozdělují na řídicí (tzv. Master – zahajuje a ukončuje komunikaci; generuje hodinový signál SCL) a řízené (tzv. Slave – zařízení adresované masterem). Sběrnice I²C neumožňuje duplexní přenos, v jednom okamžiku vysílá jen jedno zařízení. Všechna zařízení připojená na

sběrnici musí mít individuální adresu o délce 7 nebo 10 bitů a implementovaný mechanizmus komunikace pomocí I²C sběrnice.

Z elektrického hlediska jsou kanály SCL a SDA zapojeny jako otevřený kolektor. Každý tento vodič musí být připojen jedním pull-up rezistorem ke kladnému napětí, což zajistí vysokou úroveň (HIGH) v klidovém stavu sběrnice. Během vysílání musí periférie neustále porovnávat vysílané bity se skutečným stavem SDA. Vzhledem k charakteru sběrnice může dojít k situaci, že určitá stanice vysílá úroveň HIGH, zatímco jiná stanice vysílá úroveň LOW. Stanice, která na lince zjistí úroveň LOW, zatímco sama vysílá HIGH, musí vysílání okamžitě ukončit. Tím je zabráněno kolizím na této sériové lince.

**POZNÁMKA:**
Opět je tu trochu zmatek se značením! Na některých zařízeních s rozhraním I²C se můžeme též setkat s označením vodičů: SDI místo SDA a SCK místo SCL. Nic se však neděje, jedná se o totéž!

Modul Arduino UNO má pro I²C komunikaci speciálně vyhrazené dva piny, na které musíme všechny I²C periférie připojovat. Jedná se o analogové piny A4 (role vodiče SDA) a A5 (role vodiče SCL), které se při této komunikaci přepínají do digitálního režimu. Z obrázku znázorňující rozložení pinů modulu Arduino by se mohlo zdát, že tento modul disponuje ještě jednou dvojicí těchto vodičů, a to piny 18 (jako SDA) a 19 (SCL).



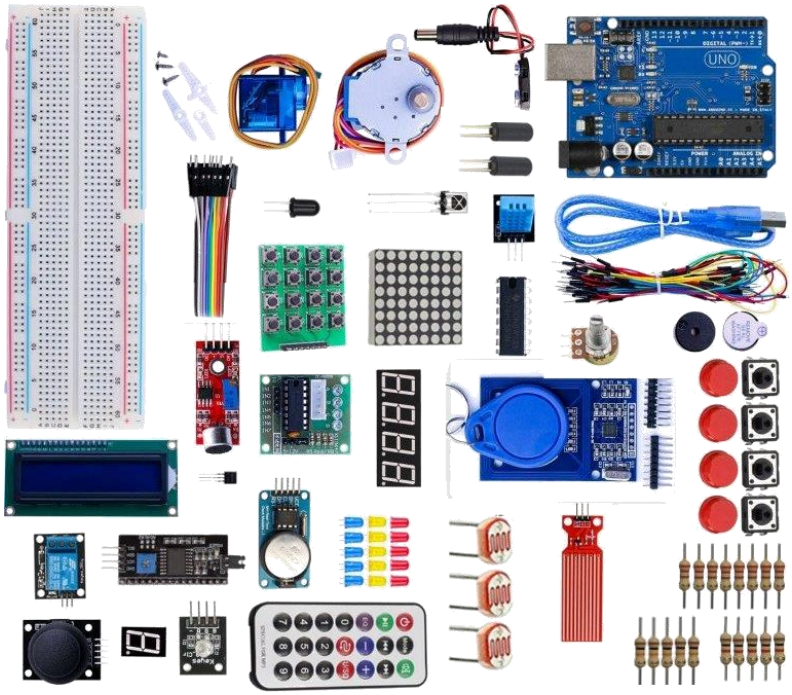

Zdání však klame, pokud si vzpomeneme, že analogové piny A0–A6 lze digitálně adresovat od čísla 14, tak vidíme, že pinům A4 odpovídá význam digitálního pinu 18 a pinu A5 obdobně pin 19. V případě I²C komunikace se tedy jedná o shodné piny. Občas se to může využít pro lepší připojení dvou I²C periférií, ale stále (jak bylo řečeno) to je stejná komunikační linka.

Ke komunikaci I²C je ještě třeba připomenout to, co sice již bylo řečeno, ale občas to může pozlobit. A to, že každé zařízení má svou vlastní adresu, které musí být na jedné komunikační lince unikátní. U některých čidel se tato adresa dá v určitém rozsahu nastavit (například pomocí propojení pinů jumperem nebo propájením pájecích bodů), jinde je však nastavena napevno přímo v čidle. Takže pokud budeme chtít nějaké I²C zařízení s modulem Arduino použít, je nutné si tuto adresu zjistit – více si ukážeme lekci: Používaní skeneru sběrnice I²C.

*To by k modulu Arduino mohlo být pro začátek všechno.*

# Arduino MAXI Starter kit

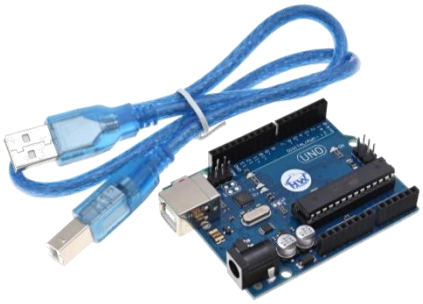
Když jsme si představili prostředí mBlock a modul Arduino, zbývá už jen jediné – podívat se na tu hromadu komponent, modulů součástek a drátků, která si říká „Arduino MAXI Starter kit“.

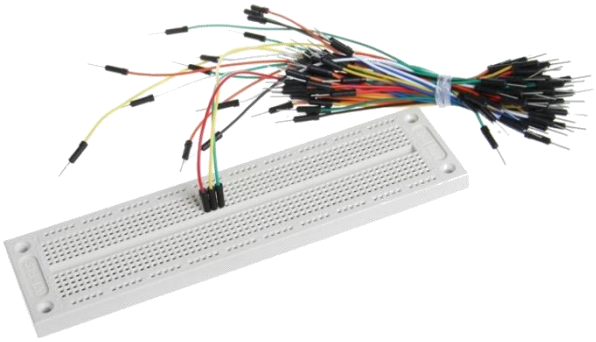
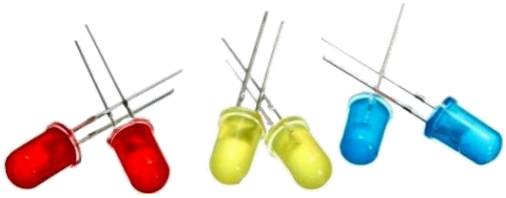
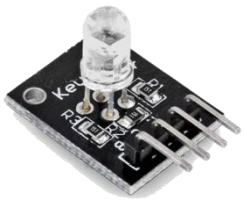
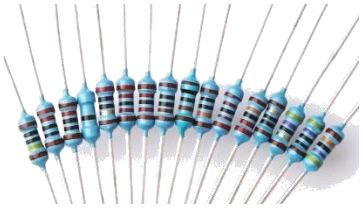
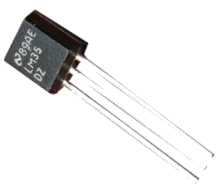


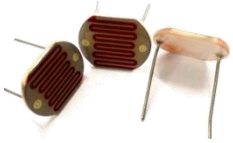
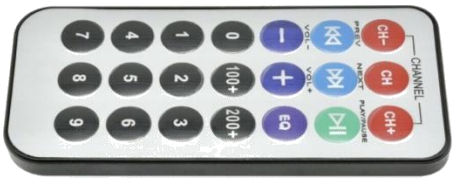
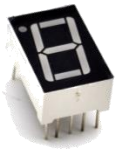
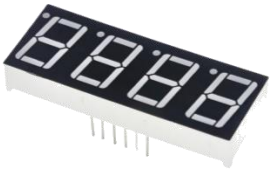
Tato MAXI studijní sada Arduino nás provede praktickými základy používání modulu Arduino. Naučíme se vytvořit několik kreativních projektů. Sada obsahuje nejběžnější a nejužitečnější elektronické součástky, díky kterým sestavíme na tři desítky projektů. Sada je vhodná jak pro začátky s elektronikou, tak pro komplexnější projekty. Tento kit nám pomůže ovládnout okolní svět s pomocí senzorů a pohonů.

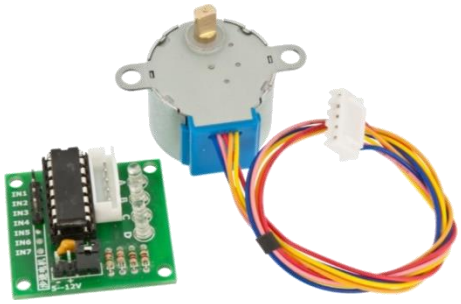
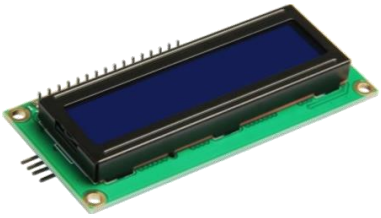
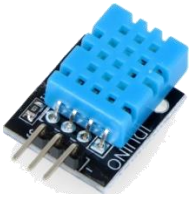
Prostřednictvím těchto experimentů bychom si měli osvojit běžné jevy a naučit se je dobře aplikovat do svých dalších samostatných projektů, ke kterým by nás měly následující ukázkové lekce inspirovat.

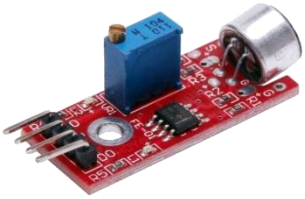
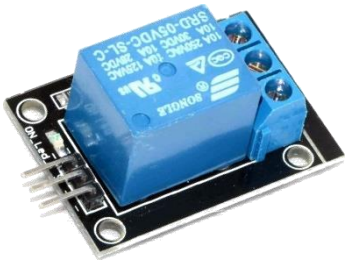
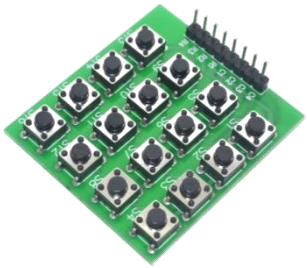
Součástí tohoto kitu jsou následující komponenty:

| | |
|---|---|
| Modul Arduino + USB kabel |  |

| | |
|---|---|
| Nepájivé pole<br>+<br>standardní vodiče typu<br>„samec-samec“<br>+<br>Doplňkové vodiče typu<br>„samec-samice“ |  |
| LED 5 mm<br>(5× červená, 5× žlutá, 5× modrá) |  |
| Modul RGB LED |  |
| Sada rezistorů<br>(8× 220 Ω, 5× 1 kΩ, 5× 10 kΩ) |  |
| Potenciometr 50 kΩ |  |
| Aktivní a pasivní bzučák |  |
| Teplotní čidlo LM-35 |  |

| | |
|---|---|
| Senzor plamene |  |
| Otřesové čidlo<br>(2×) |  |
| Fotorezistor<br>(3×) |  |
| Tlačítka + hmatníky<br>(4×) |  |
| IR přijímač HX1838 |  |
| IR ovladač |  |
| Sedmisegmentový LED zobrazovač |  |
| 4-místný sedmisegmentový<br>LED zobrazovač |  |

| | |
|---|---|
| 8×8 LED matice |  |
| Krokový motor 28BYJ-48 + řadič ULN2003 |  |
| Plastové servo 9g |  |
| LCD displej + I²C převodník |  |
| Joystick PS2 |  |
| Senzor teploty a vlhkosti vzduchu DHT11 |  |
| Senzor hladiny vody |  |

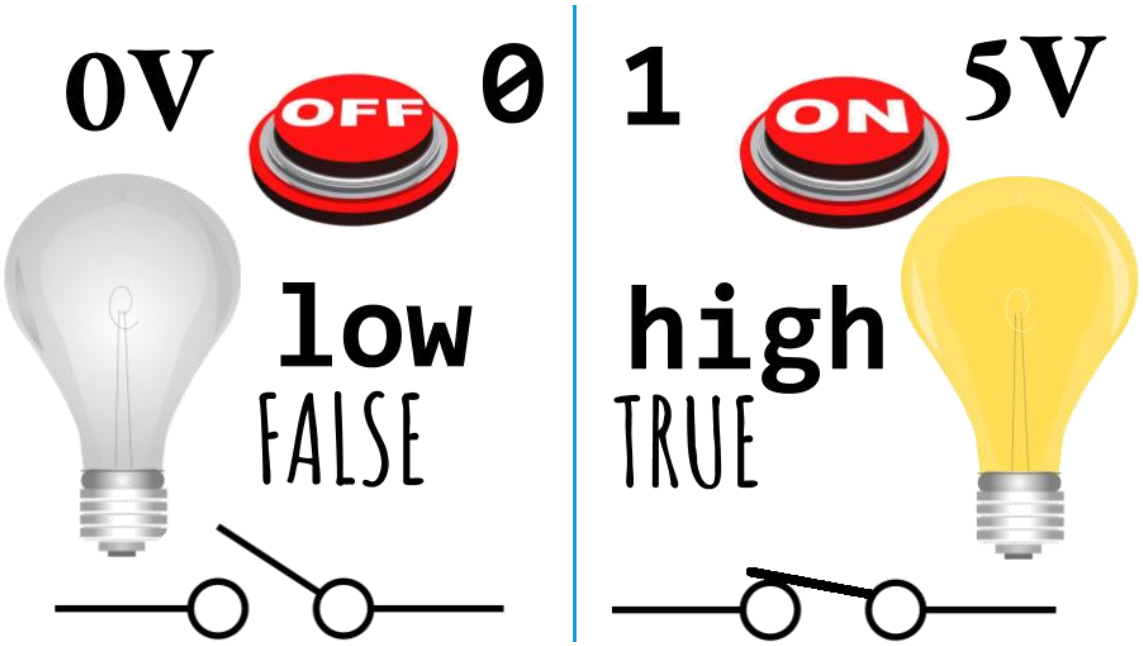
| | |
|---|---|
| RFID Sada<br>(čtečka, přívěsek, čipová karta) |  |
| Zvukový senzor |  |
| Modul relé |  |
| RTC modul<br>(hodiny reálného času DS1302) |  |
| Maticová klávesnice 4×4 |  |
| Připojovací kabel pro 9V baterii |  |

S pomocí Arduino MAXI Starter kitu si v následujících lekcích ukážeme, jak pracovat s digitálními vstupy a výstupy modulu Arduino. Dále bychom měli zvládnout načítání analogových veličin, jako je poloha potenciometru, výstup z odporového teplotního čidla a dalších. I když modul Arduino není osazeno analogovým výstupem, měla by nám tato sada názorně ukázat možnou náhradu s pomocí PWM výstupu. Ukážeme si to na možné regulaci svitu LED nebo na míchání barevných komponent pro RGB LED. Součástí sady je i několik modulů, které využívají základní sběrnice modulu Arduino. Díky tomu můžeme zcela pohodlně pracovat například s LCD displejem, RFID čtečkou a dalšími. Všechny zde uvedené komponenty sady Arduino MAXI Starter kit postupně prezentujeme v jednotlivých lekcích, jejichž obtížnost postupně roste. Společně s tím jsou představovány i další vlastnosti modulu Arduino, na kterých jsou stavěny další lekce. V závěrečných dvou lekcích je možné ze sady sestavit jednoduchý elektronický přístupový systém odemykaný buď číselným kódem, nebo pomocí RFID čipu.

Všechny zde prezentované experimenty jsou doplněny o blokové a elektronické diagramy a uvedeny s reálně vyzkoušenými programovými kódy.

**Dobrodružství nazvané „LaskaKit Arduino + mBlock“ tedy právě začíná!**

# Využití digitálních vstupů a výstupů



# Lekce 1 – Řízení LED tlačítkem

## Úvod

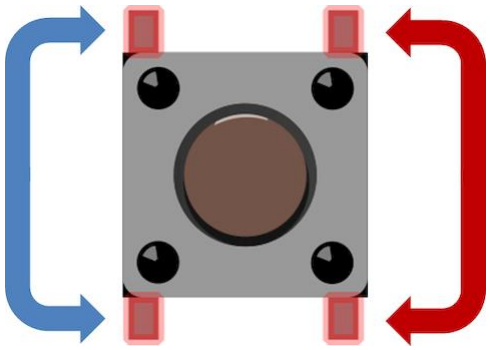
V tomto základním příkladu si ukážeme, jak zapnout/vypnout vestavěnou LED na modulu Arduino pomocí vstupního digitálního portu a tlačítka. Využijeme vstupní digitální pin, na kterém přečteme stav spínače, který má jen dva stavy – sepnutý nebo vypnutý. Pro rozsvícení LED naopak využijeme výstupní digitální pin. Vzhledem k tomu, že deska Arduino je vybavena vestavěnou LED, která je připojena k pinu 13, můžeme tuto LED rovnou použít, aniž bychom museli připojovat LED externě.

## Použité komponenty

- modul Arduino
- USB kabel
- tlačítko
- rezistor (10 kΩ)
- nepájivé pole
- vodiče pro nepájivé pole

## Princip

Tlačítka jsou běžnou součástí a slouží k ovládání elektronických zařízení. Jsou obvykle používána jako spínače nebo dokonce jako přepínače. Tlačítka se vyrábí v různých tvarech a velikostech, avšak zde jsme použili 12 mm tlačítko. U těchto spínačů je třeba si dát pozor, protože mají sice čtyři piny, ale vždy dva jsou propojené. Piny, na které ukazují šipky stejné barvy (viz obrázek), jsou propojeny. Při stisknutí tlačítka se piny, na které ukazují modré šipky, připojí na piny označené červenou šipkou.



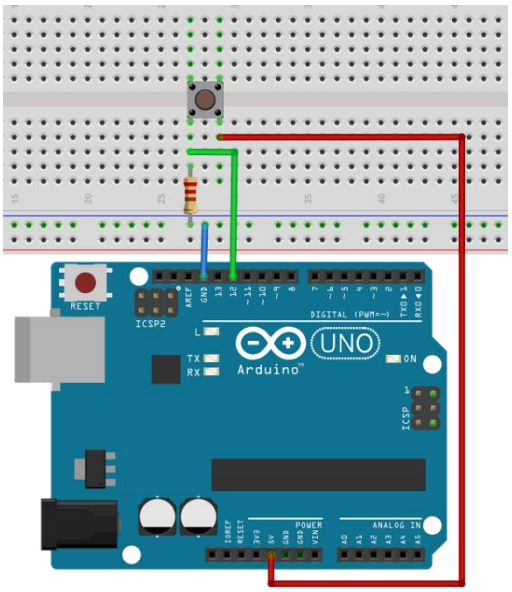
Obecně by stačilo tlačítko přímo připojit na vstupní digitální pin a s jeho pomocí připojovat požadovanou napěťovou úroveň. To by bylo relativně jednoduché. Problém je v takovém případě s napěťovými hladinami. Musíme si uvědomit, že pro digitální vstupní úroveň LOW musí být napětí na vstupu menší než 0,5 V, pro vstupní úroveň HIGH naopak vyšší než 2,5 V. Pokud tyto úrovně vstupu nějak „nevnutíme“ může se volný vstupní pin chovat různě – například se mohou projevit různé typy rušení. Aby se zabránilo těmto vnějším zásahům, které by negativně ovlivňovaly vstupní signál, je v našem zapojení použit tzv. pull-down rezistor cca 1 k–10 kΩ, který je zapojen mezi tlačítkem, vstupním pinem a zemí (GND). Díky tomuto zapojení je vstup buď přes rezistor připojen k zemi (napětí odpovídá úrovni LOW), nebo po stisknutí tlačítka k napájecímu napětí +5 V (úroveň

HIGH). Připojený rezistor zároveň zabraňuje zkratu mezi póly +5 V a GND při stisku tlačítka. Toto připojení obvodu je široce používáno v řadě obvodů a elektronických zařízení.

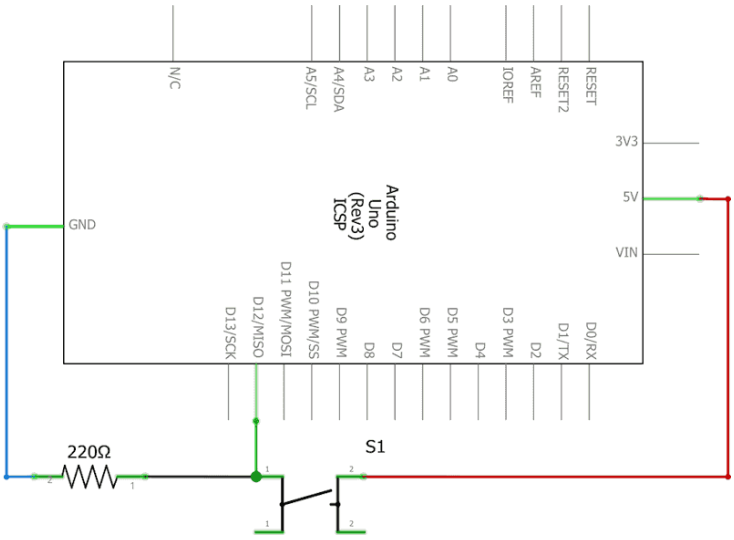
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu.

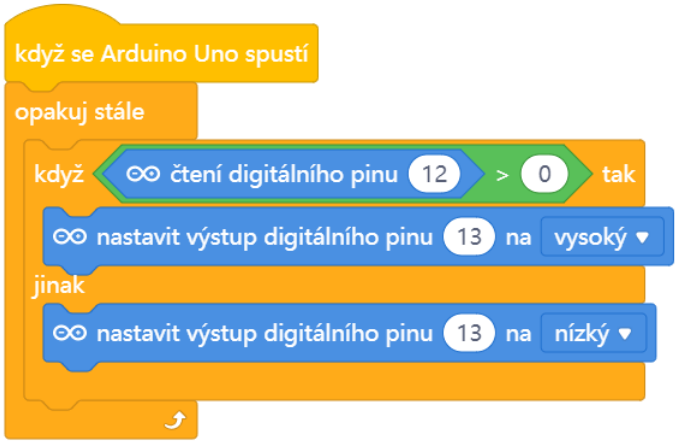
**Blokové schéma**



**Elektronické schéma**




**Krok 2:** V prostředí mBlock sestavíme následující program.

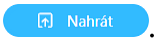



Program lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-01/01-Rizeni-LED-tlacitkem.mblock.

### Vysvětlení kódu

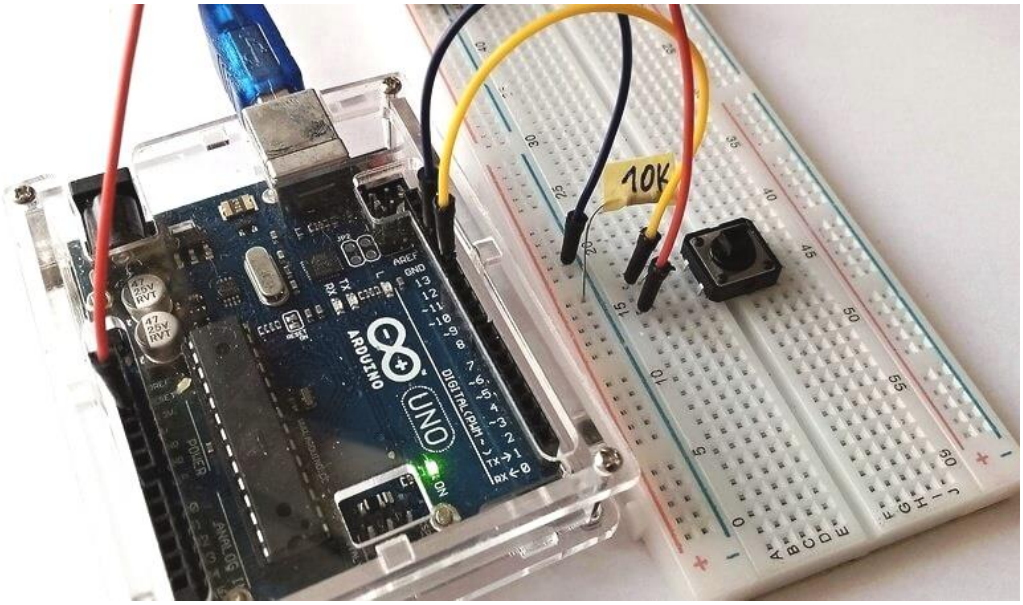
Základní nekonečná smyčka „opakuj stále“ opakovaně načítá stav digitálního pinu 12, ke kterému je připojeno tlačítko společně s pull-down rezistorem. Stav pinu je při nestisknutém tlačítku LOW (nízká

úroveň), což též odpovídá číselné podobě 0, nebo při stisknutém tlačítku hodnotě HIGH (vysoká úroveň). Tento stav testuje blok podmínky. Je-li stav pinu větší než 0, je spínač stisknutý, následuje tedy část kódy „tak“, ve kterém se výstupní digitální pin 13 nastaví do stavu HIGH (vysoká úroveň). V opačném případě je provedena alternativní část kódu „jinak“, ve které je výstupní digitální pin 13 nastaven na úroveň LOW (nízká úroveň). Vestavěná LED modulu Arduino, která je připojena přímo na pin 13 se rozsvěcí/zhasíná dle zadané výstupní úrovně.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

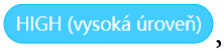
.

**Krok 4:** Nyní stiskneme tlačítko a vestavěná LED na modulu Arduino se rozsvítí. Po uvolnění tlačítka LED opět zhasne.




**POZNÁMKA:**
Výše uvedený kód je sestavený jen ze standardních příkazů prostředí mBlock. Bohužel toto standardní prostředí moc nemá vyřešenou práci se vstupními úrovněmi digitálních vstupů. Proto je načtená úroveň digitálního vstupu 12 testována na úroveň HIGH (vysoká úroveň) pomocí nerovnosti „větší než nula“. Pokud bychom chtěli použít korektnější testování, tedy skutečně testovat logickou úroveň HIGH, musíme využít rozšíření MAXI Starter kit. Jak vidíme, lze program řešit i bez něj, ale nyní ukážeme možnosti, které nám toto rozšíření přináší.

**Využití hodnoty HIGH (vysoká úroveň)**

Hned v první kategorii příkazů nazvané „Nastavení pinů“ je blok 

, který odpovídá vysoké logické vstupní úrovni, kterou nyní můžeme testovat v podmínce jako rovnost. Předchozí kód tedy upravíme podle následujícího obrázku. Funkčnost kódu je naprosto stejná, ale testování digitálního vstupu 12 je mnohem korektnější.


Program lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-01/01-Rizeni-LED-tlacitkem-ver2.mblock.

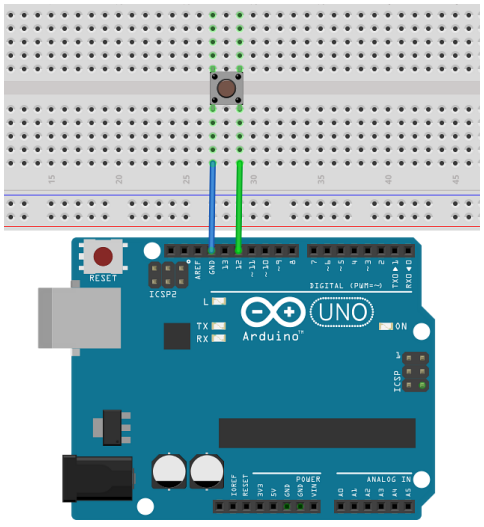
**Využití PULLUP nastavení digitálního pinu**

Rozšíření pro MAXI Starter kit nám ale otevírá i další možnost, jak realizovat načítání stisknutého tlačítka. Jak jsme si říkali v seznámení s modulem Arduino, je možné nastavit digitální vstup tak, že vstup je „držen" interním pull-up rezistorem na hodnotě +5 V (odpovídá úrovni HIGH). Je to vlastně dost podobné tomu, jak jsme pomocí externího pull-down rezistoru drželi stav pinu na úrovni LOW.

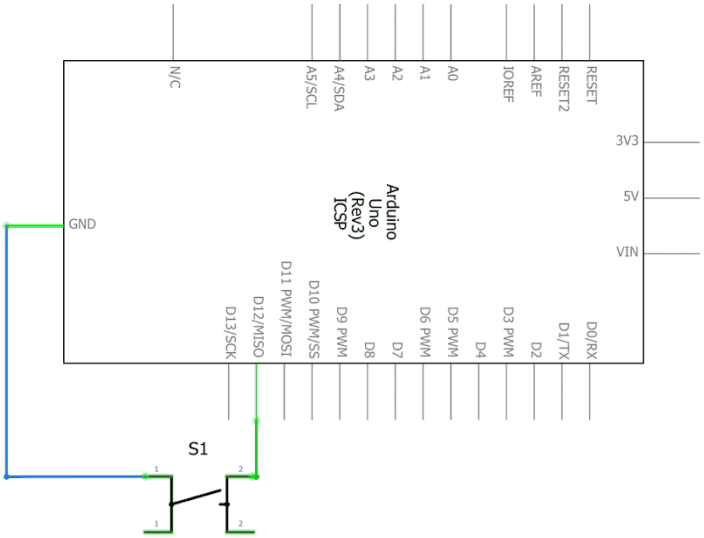
Pro nastavení a čtení digitálního pinu s připojeným interním pull-up rezistorem slouží následující programový blok:


Díky tomuto řešení s interním pull-up rezistorem modulu Arduino je možné v zapojení zcela vynechat zapojený rezistor a přes tlačítko propojit digitální pin k zemnění (GND). Zapojení tlačítka a modulu Arduino lze tedy výrazně zjednodušit, jak vidíme na následujícím obrázku a schématu.

**Blokové schéma**



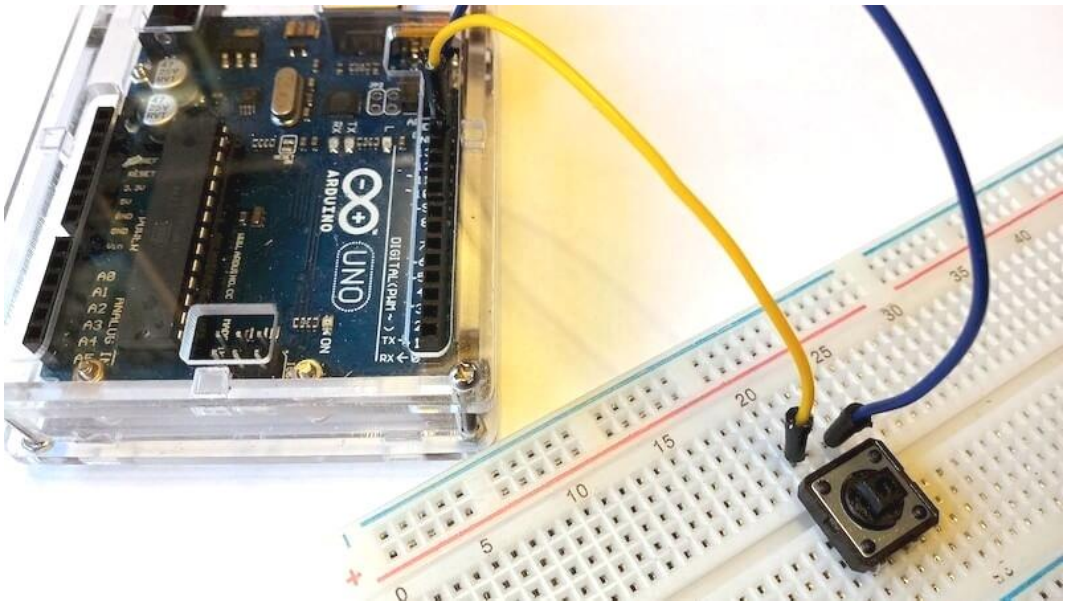
**Elektronické schéma**



Pochopitelně je třeba upravit i programový kód. Vidíme však, že se od předešlých kódů zase tak moc nezměnil. Jedinou změnou jsou programové bloky v argumentu rozhodující podmínky.


Program lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-01/01-Rizeni-LED-tlacitkem-ver3.mblock.

Jak vidíme, opět došlo jen k nepatrné změně původního kódu. V základním principu se kód vůbec nezměnil, jen jsme změnili způsob nastavení vstupního digitálního pinu a jeho načítání. Kód je ale korektnější a elektronická část celého zapojení jednodušší, a to při zachování stejné funkčnosti. To vše má na svědomí použití rozšíření MAXI Starter kit!

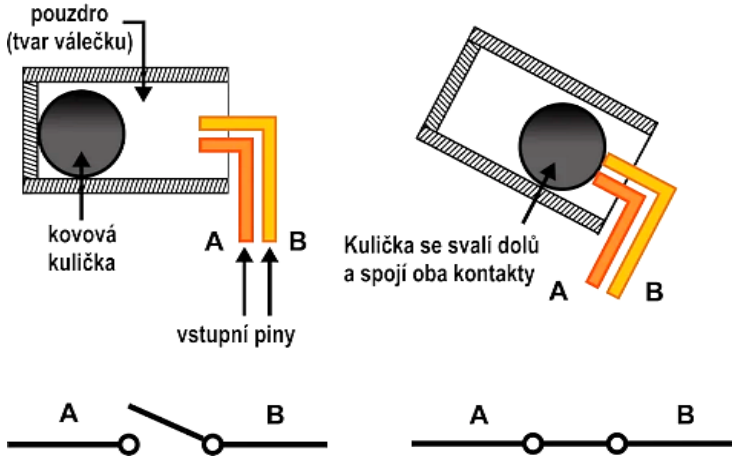


*Tak co… Ještě nás to nepřesvědčilo o tom, že se docela vyplatí přidat rozšíření MAXI Starter kit do prostředí mBlock? Ne? To nevadí! Snad to přijde později.*

# Lekce 2 – Otřesové čidlo

## Úvod

Následující zapojení s otřesovým čidlem je dosti podobné tomu předchozímu se spínačem. Především si chceme ukázat, že na principu spínače může vlastně fungovat i nějaké smysluplnější zařízení. V případě otřesového čidla jde o spínač s kovovou kuličkou uvnitř, která se při změně polohy „převalí“ na kontakty a tím je spojí. Takový polohový spínač se používá k detekci malého úhlu sklonu a můžeme ho použít jako čidlo proti manipulaci (odcizení) věci nebo třeba jako obvod signalizující naklonění terénního automobilu nad přípustnou mez.




## Použité komponenty

- modul Arduino
- USB kabel
- otřesové čidlo
- nepájivé pole
- vodiče pro nepájivé pole

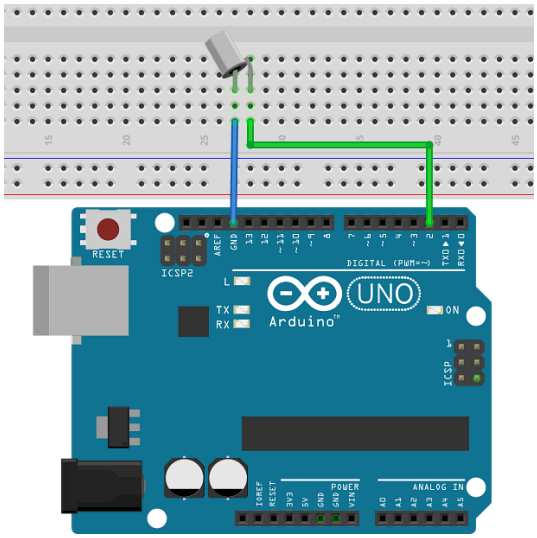
## Princip

Jak bylo řečeno v úvodu, princip je velice jednoduchý. Je-li spínač nakloněn v určitém úhlu, kulička uvnitř sjede dolů a dotkne se dvou kontaktů připojených k vnějším vývodům. Tím se kontakty propojí. Pokud zůstane kulička mimo kontakty, je elektrický obvod rozpojen. Takže jakékoliv testování stavu tohoto čidla bude stejné, jako bylo v případě spínače. Opět budeme načítat vstupní digitální pin.

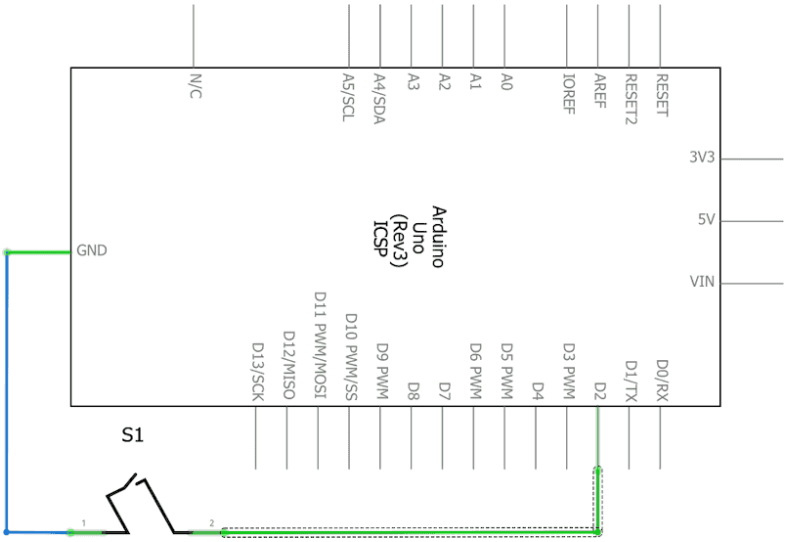
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Pro připojení čidla si můžeme vybrat libovolný digitální pin, my jsme si v tomto případu vybrali pin 2.

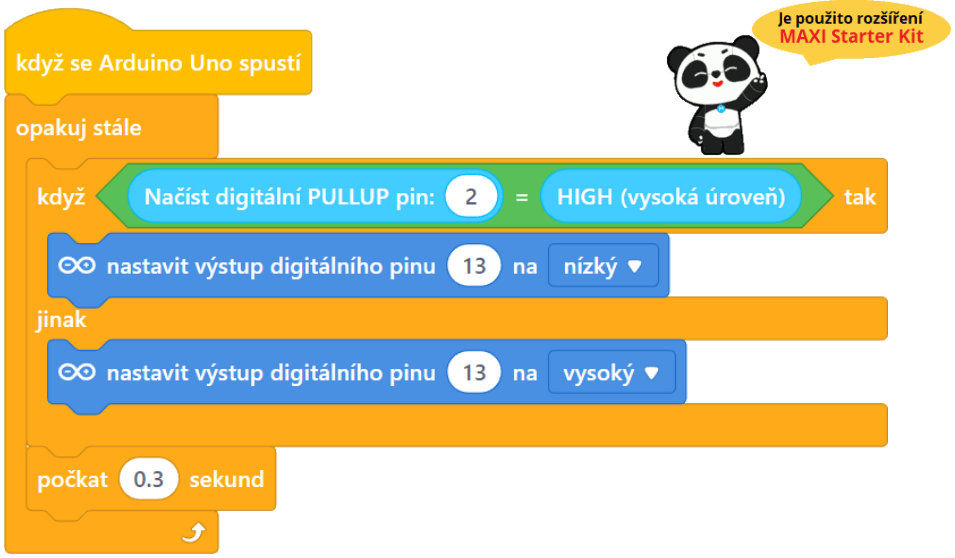
**Blokové schéma**




**Elektronické schéma**




**Krok 2:** Sestavíme v prostředí mBlock následující program:




Program lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-02/02-Otresove-cidlo.mblock.

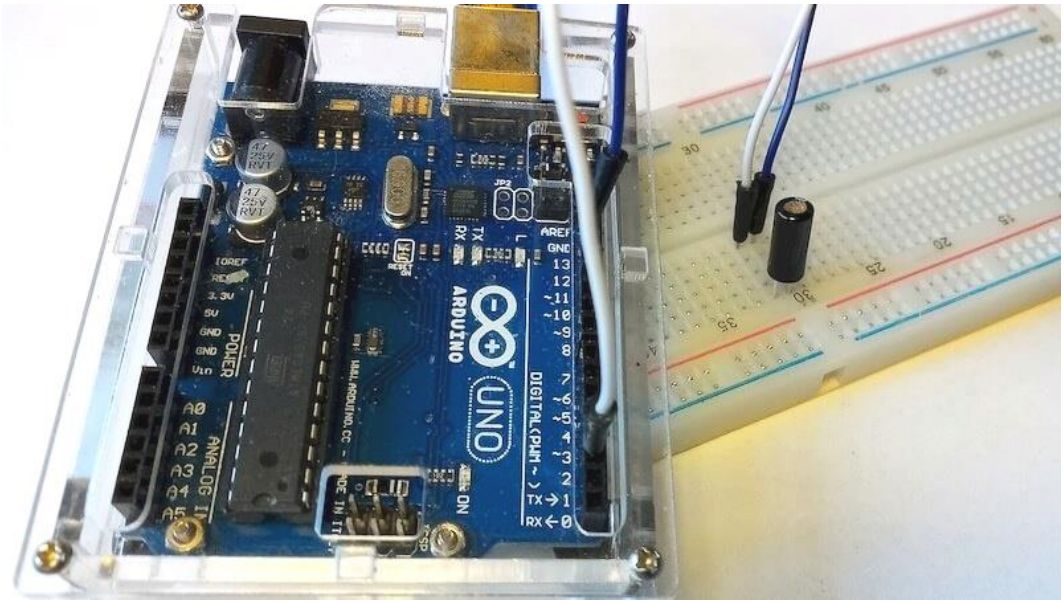
### Vysvětlení kódu

Porovnejme tento kód s programem z lekce 1. Nejsou si podobné? Liší se jen v čísle načítaného digitálního pinu. Vysvětlení kódu lze tedy najít v lekci 1.

Takže pokud se stále ještě někdo děsí rozšíření MAXI Starter kit, může tento kód předělat bez tohoto rozšíření pomocí prvotního kódu z lekce 1. Ale pak nezapomeňme, že bude třeba použít externí pull-down rezistor.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Pohneme-li výrazněji polohovým čidlem, měla by se na modulu Arudino rozsvítit vestavená LED, která je připojena k pinu 13. Později uvidíme, jak k modulu Arduino připojit bzučák nebo relé. To by třeba mohla být cesta ke konstrukci polohového signalizačního systému.



# Lekce 3 – Modul relé

## Úvod

Elektromagnetické relé je elektrotechnická součástka, která obsahuje elektromagneticky ovládané kontakty (spínač). Relé bylo vynalezeno roku 1835 Josefem Henrym a původně bylo využíváno jako mechanický zesilovač na telegrafních linkách. Jeho název se odvozuje z přepřahacích stanic na kurýrních cestách, protože v původním určení relé tak trochu „přepřahalo“ pro delší šíření telegrafního signálu.

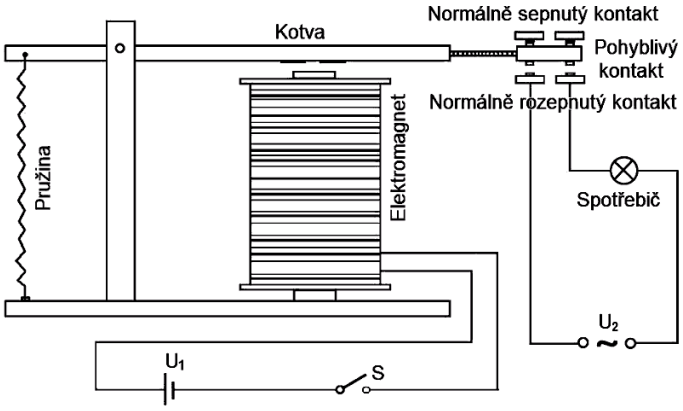
Relé se používá v mnoha aplikacích, byť jeho funkci dnes v mnoha případech přebírají obvody založené na polovodičích. Na rozdíl od polovodičů relé obvykle zajišťuje galvanické (izolační) oddělení řídícího a řízeného obvodu. Elektromagnetické relé je příkladem využití elektromagnetu v zařízení, které je důležitým funkčním prvkem v automatizovaných soustavách a řídicích systémech.


## Použité komponenty

- modul Arduino
- USB kabel
- modul relé
- vodiče „samec-samice“

## Princip

Relé se v základním provedení skládá z cívky (elektromagnetu) navinutém na jádru z magneticky měkkého feromagnetického materiálu. Magnetický obvod je doplněn pohyblivou kotvou. Kotva je pružinou uváděna do klidové polohy a současně se opírá o pohyblivý kontakt. Po připojení cívky k elektrickému zdroji vyvolá proud cívkou v magnetickém obvodu magnetický tok, který způsobí přitažlivou sílu na kotvu. Ta přemůže sílu pružiny a překlopí se – tím se sepne kontakt. Po odpojení elektrického proudu se kotva vrátí do předchozího (klidového) stavu – kontakt se rozepne. V relé často bývá dvojice spínacích kontaktů, které jsou zapojeny tak, že když se jedeny kontakty sepnou, druhé se rozepnout. Tak lze relé používat ve spínacím i rozpínacím režimu.

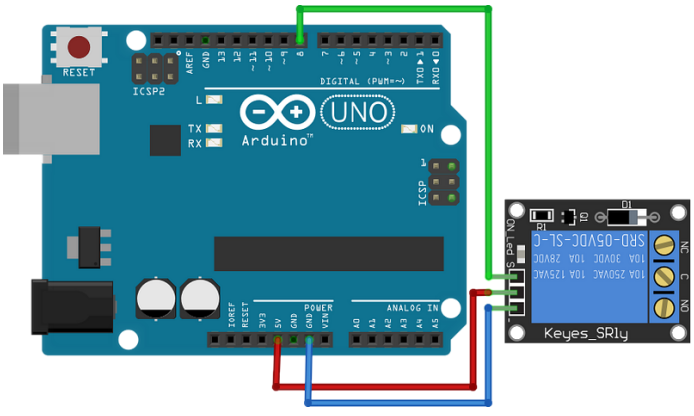



Když pošleme z modulu Arduino úroveň LOW (nízká úroveň) do vstupu modulu relé, vstupní tranzistor se otevře. Přes cívku relé bude protékat proud, spínací kontakt relé se uzavře, zatímco rozpínací kontakt relé se přeruší. Když pošleme z modulu Arduino úroveň HIGH (vysoká úroveň), vstupní tranzistor se zavře a relé se vrátí do původního stavu. Tímto způsobem můžeme pomocí relé spínat další funkční celky – např. rozsvěcet žárovku, zapnout/vypnout ventilátor, spustit alarm, roztočit motor… apod. Díky tomu, že spínací kontakty relé mohou být uzpůsobeny na poměrně velké proudy, lze pomocí relativně malého proudu z modulu Arduino do cívky relé spínat značně silové spotřebiče.

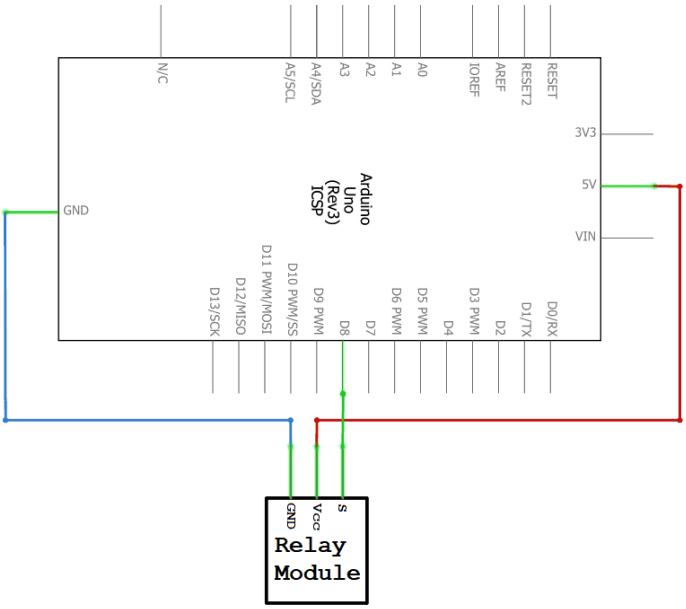
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku a schématu. Zapojení pinů modulu Arduino a pinů modulu relé znázorňuje pro přehlednost připojená tabulka.

**Blokové schéma**

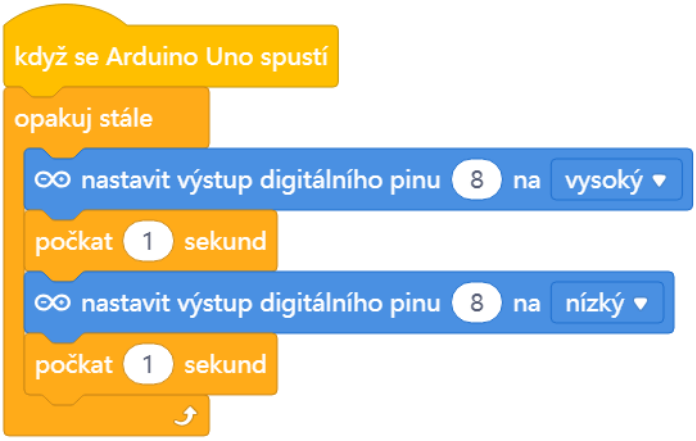



**Elektronické schéma**



| modul Relé | modul Arduino |
|---|---|
| S | 8 |
| $V_{CC}$ (+) | 5V |
| GND (–) | GND |

**Krok 2:** V prostředí mBlock vytvoříme následující program:



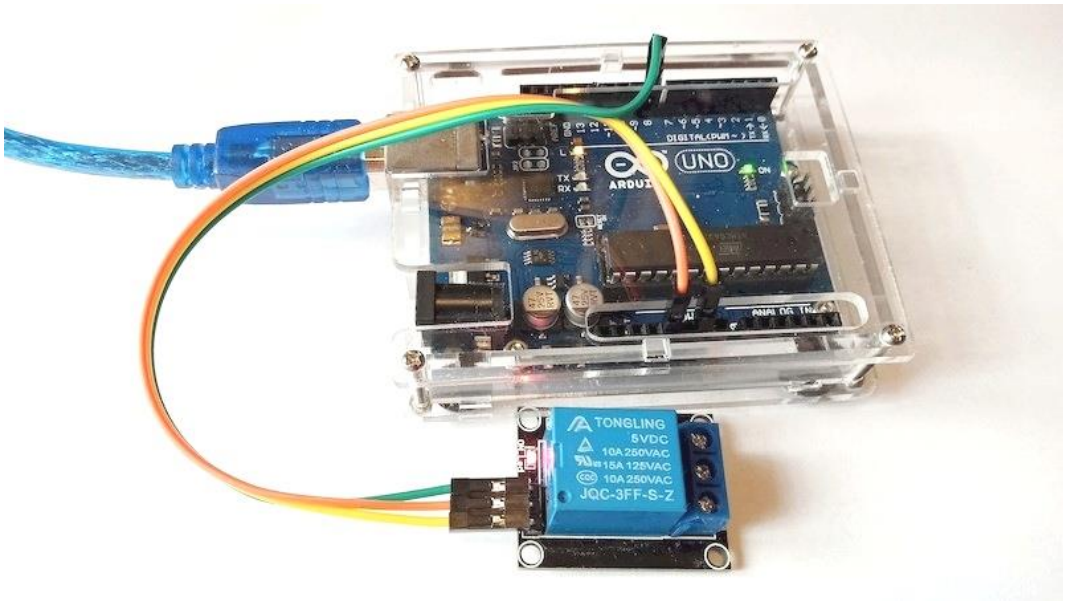

Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-03/03-Modul-rele.mblock.

### Vysvětlení kódu

Základní nekonečná smyčka „opakuj stále“ nejdříve nastaví výstupní digitální pin 8 modulu Arduino, ke kterému je připojen řídicí signál relé, do stavu HIGH (vysoká úroveň). Tím se, jak bylo dříve vysvětleno, relé přepne do jednoho stavu sepnutých/rozepnutých kontaktů. Pak kód počká jednu vteřinu. Po uplynutí čekacího času, je na digitální výstup odeslán stav LOW (nízká úroveň), který přepne relé do opačného stavu. V tomto stavu relé opět vydrží jednu vteřinu. Pak se nekonečná smyčka začíná opakovat a vše nastává znova a znova.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Nyní můžeme slyšet známé „tik-tak, tik-tak“. To se ozývají kontakty relé, jak se pravidelně spínají a naopak rozpínají. Stav relé nám též signalizuje LED, která je zapojena na modulu relé.


UNO
TONGLING
5VDC
10A 250VAC
15A 125VAC
10A 250VAC
JQC-3FF-S-Z

# Lekce 4 – Bzučák

## Úvod

Bzučák můžeme využít kdykoli, pokud budeme chtít vytvořit nějaký zvuk. Už jsme si v předešlých lekcích řekli, že by použití zvuku bylo možné použít třeba v alarmu spínaném pomocí pohybového čidla. Ale klidně by šel použít v kombinaci s tlačítkem jako jednoduchý zvonek. Jsme teprve u lekce 4 a již vidíme, že se předešlé získané znalosti dají docela dobře kombinovat.

## Použité komponenty

- modul Arduino
- USB kabel
- aktivní bzučák
- nepájivé pole
- vodiče pro nepájivé pole

## Princip

V případě elektronického bzučáku jde již o poměrně složitý obvod s integrovanou strukturou. Bzučáky, které jsou napájeny stejnosměrným napětím, jsou hojně využívány v počítačích, alarmech, elektrických hračkách, automobilových elektronických zařízeních, telefonech, časovačích… zkrátka všude tam, kde potřebujeme zvukový výstup. Bzučáky můžeme rozdělit na aktivní nebo pasivní. Sada Arduino MAXI kit obsahuje oba tyto typy. Jak je od sebe rozeznáme? Otočíme bzučáky tak, aby měly kolíčky nahoru (*viz následující obrázky*) – ten se zelenou kulatou destičkou je pasivní, zatímco ten zalitý černým lepidlem (polymerem) je bzučák aktivní.

Aktivní bzučák

Pasivní bzučák

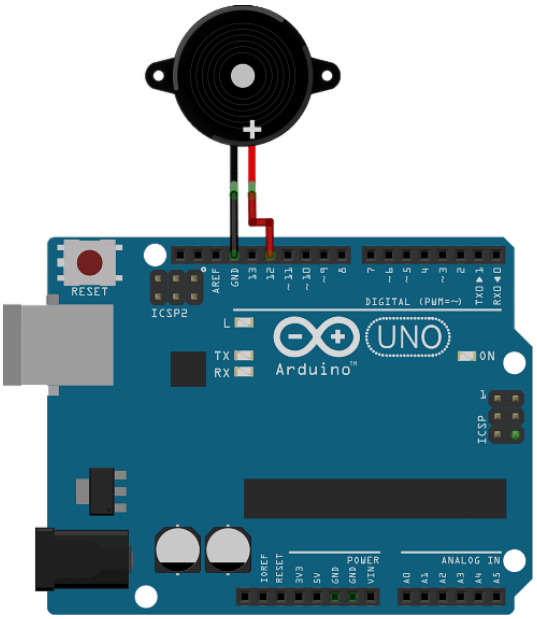
**Rozdíl mezi aktivním a pasivním bzučákem:**

Aktivní bzučák má v sobě zabudovaný oscilační zdroj, takže bude vydávat předem definovaný zvuk, jakmile bude pod proudem. Pasivní bzučák ale takový zdroj nemá, proto při použití stejnosměrného napětí pípat nebude – místo toho je potřeba použít čtvercové budící vlny s frekvencí mezi 2 kHz a 5 kHz. Zvuk tohoto bzučáku pak bude odpovídat frekvenci budícího signálu. To je jeho výhoda. Aktivní bzučák bývá obecně nákladnější a to právě z důvodu zabudovaného oscilátoru. V následujícím pokusu použijeme aktivní bzučák. Zkusíme si ukázat, že i když má aktivní bzučák jasně definovanou frekvenci, můžeme z něj vyloudit několik různých zvuků.

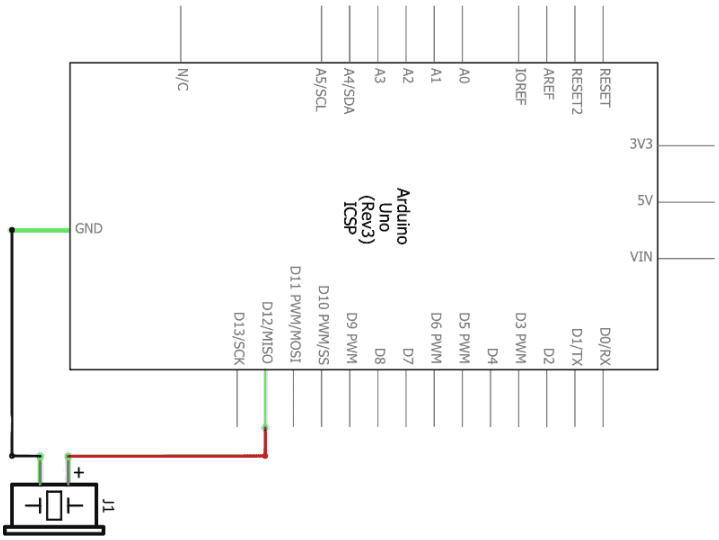
## Postup experimentu

**Krok 1:** Sestavíme zapojení podle následujícího obrázku a schématu.

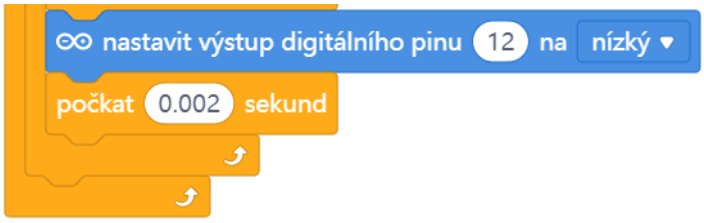
**Blokové schéma**




**Elektronické schéma**



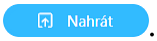

**Krok 2:** V prostředí mBlock vytvoříme následující program:




Program lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-04/04-Bzucak.mblock.

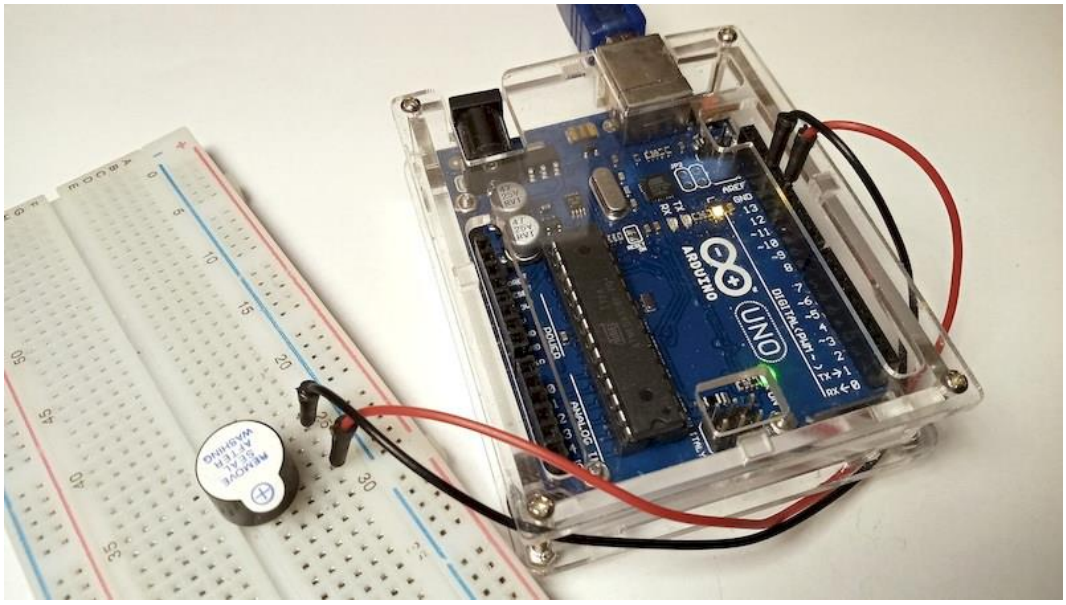
### Vysvětlení kódu

V základní nekonečné smyčce „opakuj stále“ si vytvoříme další dvě smyčky s pevným počtem opakování, které zopakují zadaný kód vždy stokrát. V obou těchto smyčkách s pevným počtem opakování neustále nastavujeme výstupní digitální pin 12, ke kterému je připojen aktivní bzučák, na úroveň HIGH (vysoká úroveň) a LOW (nízká úroveň). Díky tomu se bzučák neustále zapíná (generuje základní tón) a utichá. Rozdíl mezi první a druhou smyčkou je v časových intervalech, po kterou je bzučák zapnut/vypnut. Jelikož jsou tyto úseky velice krátké 0,001 a 0,002 vteřiny dochází v celkovém výsledku k určité zvukové modulaci, jejímž výsledkem je změna vysílaného tónu. Tímto zajímavým trikem lze „donutit“ aktivní bzučák ke změně generovaného tónu.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

.

**Krok 4:** Po spuštění modulu Arduino bychom měli slyšet, jak bzučák vytváří zvuky.



# Lekce 5 – LED tekoucí potok

V této lekci uděláme jednoduchý, ale zajímavý experiment – pomocí LED vytvoříme efekt tekoucího LED světla. Toto tekoucí světlo bude tvořeno pomocí osmi LED v řadě, které se postupně rozsvítí a zhasnou – jedna po druhé, stejně jako tekoucí voda.

## Použité komponenty

- modul Arduino
- USB kabel
- 8× LED
- 8× rezistor (220 Ω)
- nepájivé pole
- vodiče pro nepájivé pole

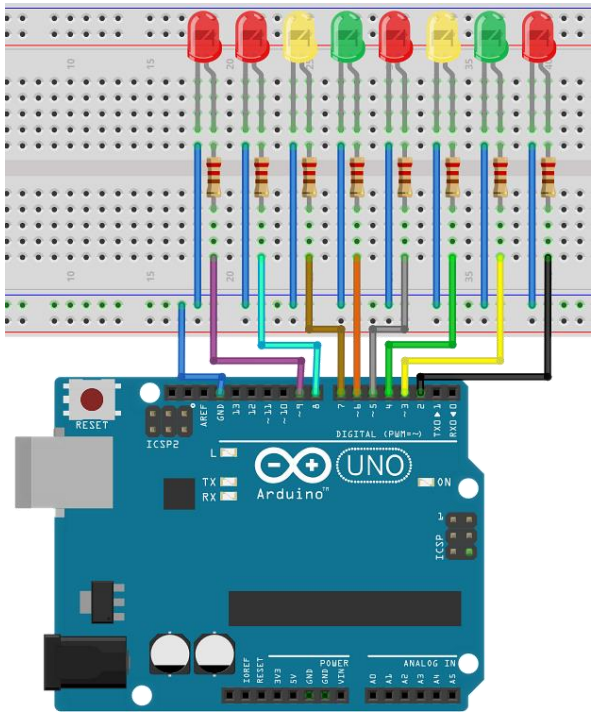
## Princip

Princip tohoto experimentu je jednoduchý – postupně zapnout a zhasnout osm LED za sebou pomocí digitálních výstupů. Jak ale dále uvidíme, při tvorbě programu budeme muset využít tzv. proměnných.

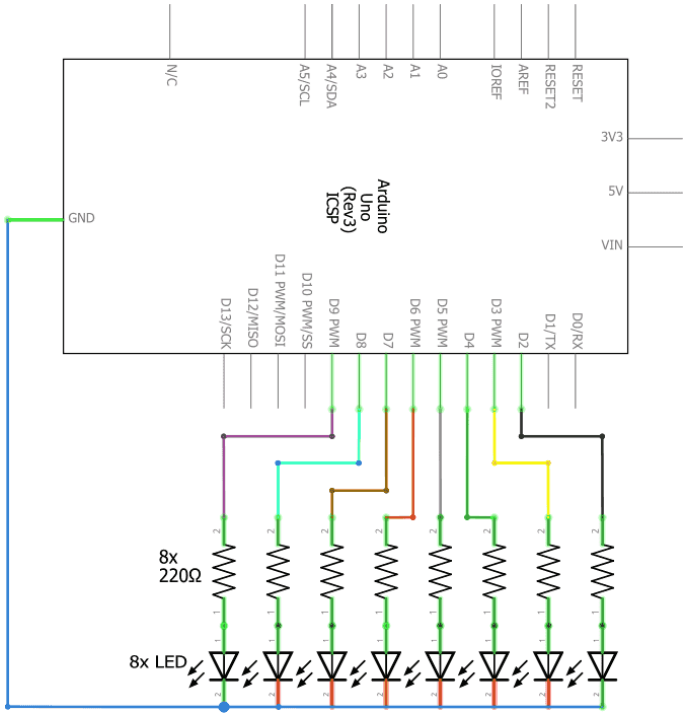
## Postup experimentu

**Krok 1:** Vytvoř obvod podle následujícího obrázku. LED zapojíme katodou (*kratší vývod*) k zemi GND a anodou (*delší vývod*) přes rezistor 220 Ω k digitálním pinům modulu Arduino.

**Blokové schéma**




**Elektronické schéma**



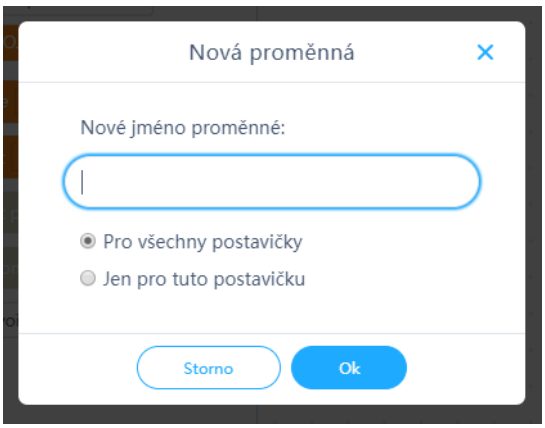
## Použití proměnných

Abychom v programu mohli postupně „adresovat“ jednotlivé LED, využijeme k tomu práci s proměnnými. Práce s proměnnými je jednou ze základních technik programování, takže čím dříve se ji naučíme, tím dříve se můžeme pustit i do pořádných složitějších projektů.

Proměnné jsou paměťová místa, do kterých si můžeme ukládat různé hodnoty, se kterými pak můžeme dále pracovat. Můžeme si tak třeba ukládat stavy jednotlivých tlačítek, nebo kupříkladu celkový stav programu, ve kterém se právě nalézá. My zde proměnné budeme využívat pro číslo LED, kterou budeme chtít rozsvítit.

Proměnné vytvoříme a později pro následné použití i najdeme v příkazové skupině 

na středním svislém příkazovém panelu. Proměnnou vytváříme pomocí tlačítka Vytvoř proměnnou. Následuje poněkud zvláštní dialog (viz následující obrázek):




Na zadání jména proměnné asi není nic zvláštního, naopak možnost volby typu „Pro všechny postavičky“ a „Jen pro tuto postavičku“ může někoho vyděsit. Musíme si uvědomit, že prostředí mBlock je mimo jiného určeno i pro programování spritových postaviček. Tam toto nastavení má svůj význam a opodstatnění. My necháme zvolenou standardní volbu „Pro všechny postavičky“ a klikneme na tlačítko 

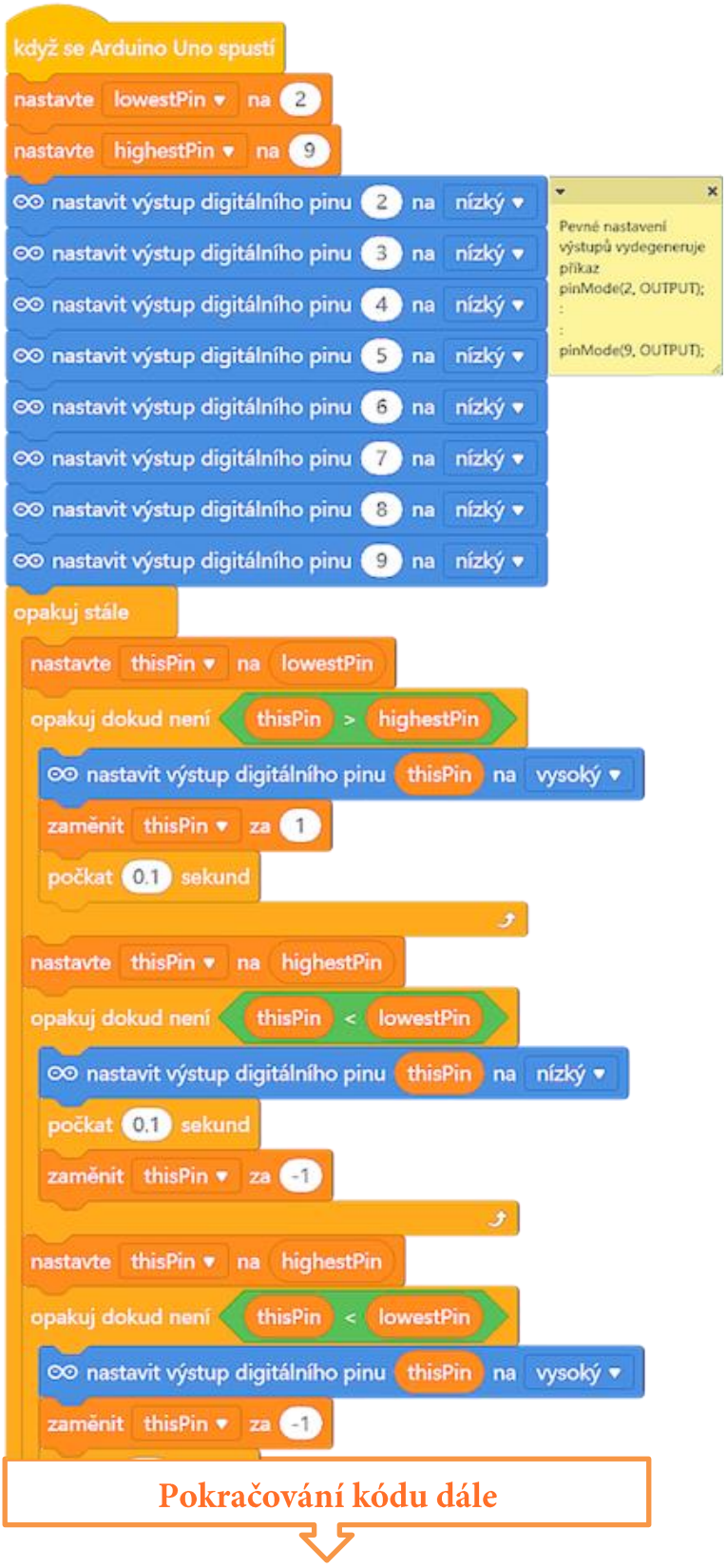
. Rázem máme k dispozici oranžový programový blok pro zvolenou proměnnou, se kterým se může dále pracovat.

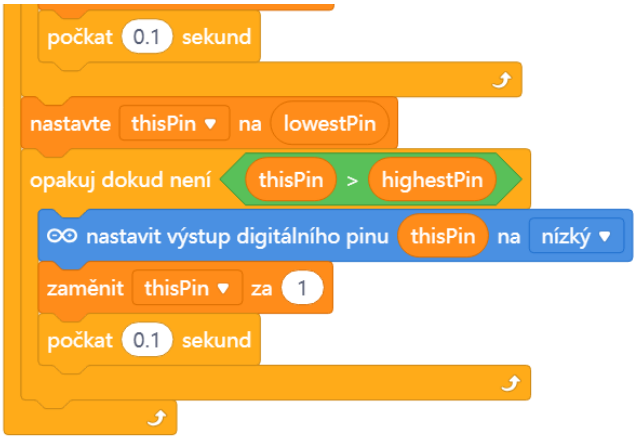
**POZNÁMKA:**
Pozor na chybný překlad bloku pojmenovaného „zaměnit“ u proměnných (*viz následující ukázka*):


Nejde o blok nějaké záměny obsahu proměnné, ale o aritmetické přičtení zvoleného čísla k obsahu zadané proměnné. Zde tedy k obsahu proměnné **AHOJ** bude přičtena 1.

*Na tuto chybu v překladu jsme vývojáře mBlok upozornili, ale doposud jsme se nedočkali nápravy.* ☹

**Krok 2:** Sestavíme v prostředí mBlock následující program. Pro potřeby našeho programu si vytvoříme proměnné nazvané `lowestPin`, `highestPin` a `thisPin`.






Program lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-05/05-LED-Tekouci-potok.mblock.

### Vysvětlení kódu

První část programu nastavuje do proměnných `lowestPin` a `highestPin` rozsah výstupních pinů, na kterých jsou připojeny jednotlivé LED. Velice zvláštní částí programu je úvodní nastavení všech výstupních pinů na nízkou úroveň. Nejde ani tak o nastavení této úrovně, jako o deklaraci, která se díky tomu v programu vytvoří. Prostředí mBlock totiž umí nastavit typ pinu (digitální/analogový, vstup/výstup) jen podle pevně zadaného čísla pinu. Bohužel v mBlock není možná dynamická deklarace pinů pomocí proměnné, která by se kupříkladu zvyšovala v nějakém úvodním cyklu od hodnoty proměnné `lowestPin` do hodnoty `highestPin`. Budeme si tedy pamatovat, že piny, které budeme chtít později dynamicky nastavovat pomocí proměnné, což je pochopitelně možné, je třeba na začátku programu pevně deklarovat trikem s přiřazením libovolné vstupní nebo výstupní hodnoty. Tím dojde v programu k deklaraci a tedy i určení, zda pin bude vstupní nebo výstupní a zda bude digitální, či analogový.

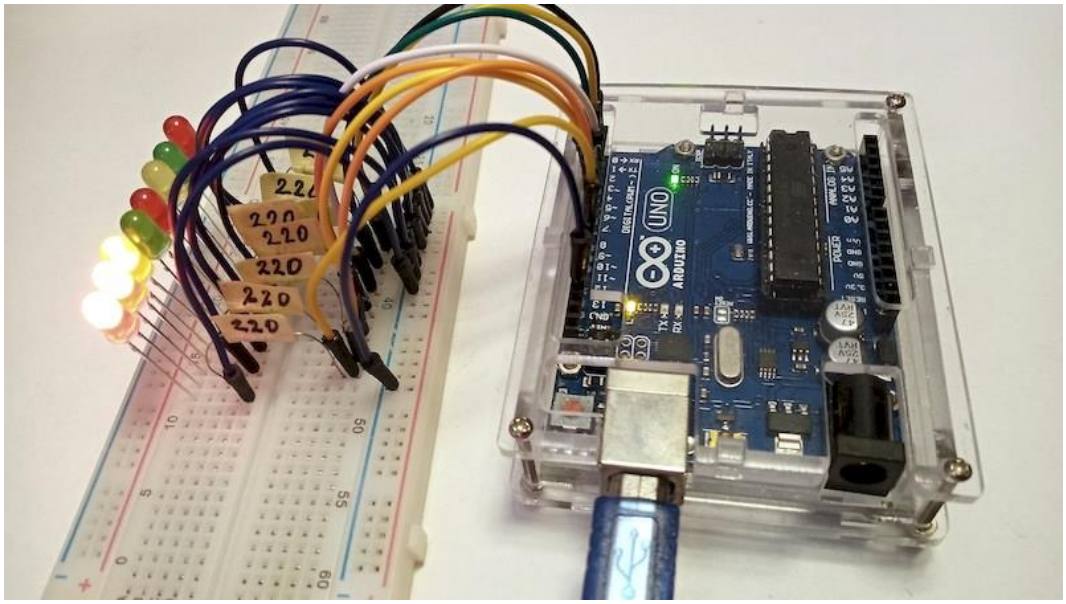
V hlavní nekonečné smyčce „opakuj stále“ je proměnná `thisPin` nastavena na adresu nejnižšího pinu s LED. Následuje podmíněná smyčka, ve které se hodnota proměnné `thisPin` postupně zvyšuje až do hodnoty nejvyššího čísla pinu s LED. Zároveň je v této smyčce nastavován příslušný pin na vysokou úroveň (se zdržením jedné desetiny vteřiny na každý pin). Díky tomu se postupně jedním směrem rozsvěcí řada LED připojených na výstupní piny. Po skončení první podmíněné smyčky jsou všechny LED rozsvícené.

Následující podmíněné smyčky fungují obdobným způsobem. Rozdíl je jen ve směru adresace – tedy zda se proměnná `thisPin` zvyšuje od proměnné `lowestPin` do hodnoty proměnné `highestPin`, nebo naopak snižuje od `highestPin` do `lowestPin`. To určuje, která LED se má rozsvítit (vysoká úroveň HIGH) nebo zhasnout (nízká úroveň LOW).

Určitě stojí na pokus, zkusit si trochu pohrát s pořadím jednotlivých podmíněných smyček nebo se směry adresace při rozsvícení/zhasnutí LED. Výsledný efekt postupného „tečení“ světla se tak dá krásně ovlivnit.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Nyní bychom měli vidět osm LED, které se postupně rozsvítí jedna po druhé – zleva doprava. Potom stejně postupně by měly zase zhasnout – zprava doleva. Následovat by mělo rozsvícení zprava doleva a zhasnutí zleva doprava. Celý tento proces se bude stále opakovat.



# Lekce 6 – Kvízový bzučák

## Úvod

Ve vědomostních soutěžích, především v těch zábavních, organizátoři často používají bzučákový systém, který slouží k přesnému a spravedlivému určení čísla odpovídajícího. Zkusíme si takový systém také vytvořit. Jelikož se jedná již o složitější projekt, budeme potřebovat větší počet součástek. V této lekci budeme používat tlačítka, bzučák a několik LED s rezistory.

## Použité komponenty

- modul Arduino
- USB kabel
- 4× tlačítko
- 4× LED
- 4× rezistor (220 Ω)
- aktivní bzučák
- nepájivé pole
- vodiče pro nepájivé pole + vodiče typu „samec-samice“

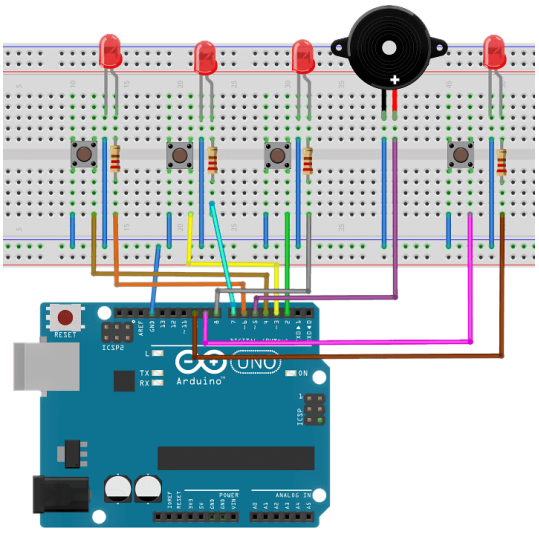
## Princip

Tlačítka 1, 2 a 3 budou sloužit pro odpovídání soutěžících, tlačítkem 4 budeme celé zařízení resetovat, aby bylo připraveno pro další odpověď. Zapojení bude testovat stisknutí jednotlivých tlačítek soutěžících a dle jejich stavu vyhodnotí výsledek vizuálně pomocí zhasnutí odpovídající LED a i zvukově pomocí signálu. Chceme-li začít další kolo, jednoduše zmáčkneme tlačítko 4 a opět vše bude připraveno.

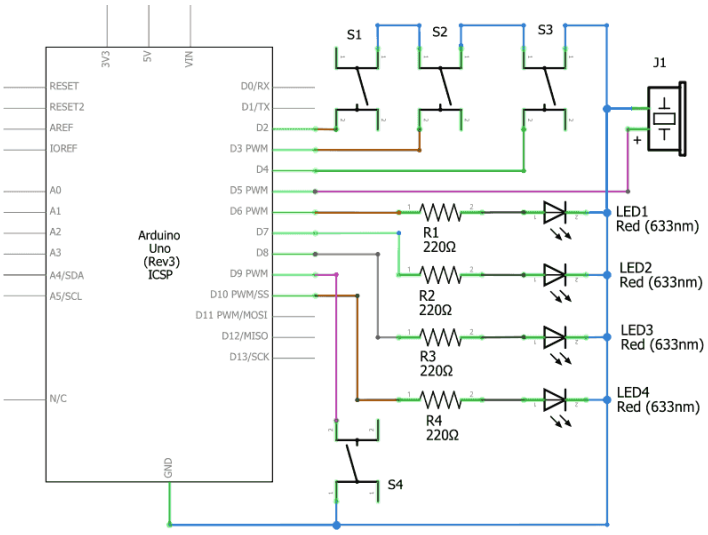
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku a schématu. LED zapojíme katodou (*kratší vývod*) k zemi GND a anodou (*delší vývod*) přes rezistor 220 Ω k digitálním pinům modulu Arduino.

**Blokové schéma**



**Elektronické schéma**

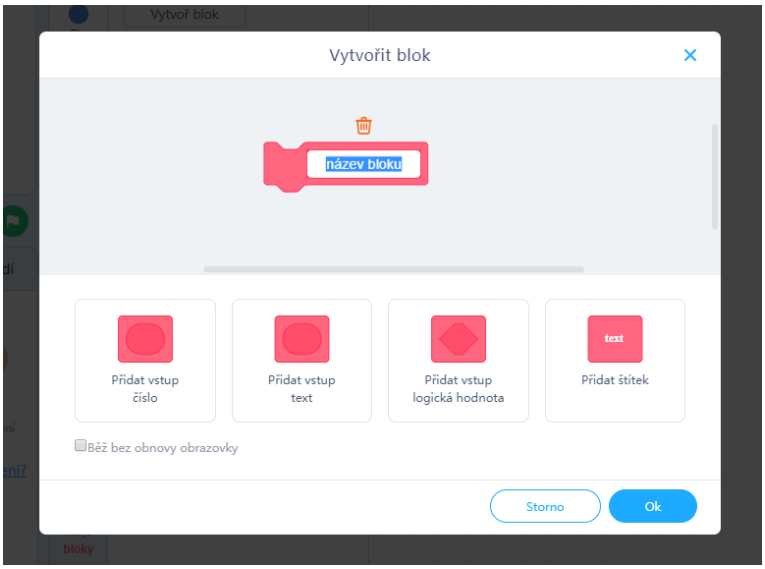


## Použití podprogramu

Jak jsme viděli již při zapojování elektronické části této lekce, je lekce kvízového bzučáku již jedním z relativně složitějších zapojení. Ne jinak tomu bude i v případě řídicího programu. V programu budeme na několika místech potřebovat spustit zvukový alarm, abychom nemuseli na toto místo neustále psát část kódu, která je vlastně stejná, využijeme tzv. podprogram, což je část kódu, kterou budeme volat mocí jejího programového bloku. Vlastně si tak trochu vytvoříme takový svůj vlastní programový blok.

Nezačneme tedy tvorbou hlavního programu, ale právě tímto podprogramem. Vytváříme jej normálně na pracovní ploše prostředí mBlock – třeba hned vedle bloků hlavního programu.

Vytvoření vlastního podprogramu začneme kliknutím na růžovou ikonu Moje bloky na svislé střední liště příkazových bloků. V místech, kde obvykle bývá nabídka programových bloků je nyní jen tlačítko Vytvoř blok. Pro vytvoření podprogramu (svého bloku) na tlačítko klikneme a otevře se nám okno průvodce vytvoření vlastního bloku.




Nejdříve je třeba zvolit jméno bloku podprogramu, my si zvolíme název `Alarm`. Další částí tvorby je pak nastavení vstupních parametrů, se kterými může tento blok pracovat. Jelikož se nyní s tvorbou podprogramu seznamujeme, je náš vlastní blok bez jakýchkoliv dalších vstupních parametrů. Později si ukážeme, jak vytvořit podprogram se vstupními parametry a uvidíme, že se to občas opravdu hodí. V tuto chvíli se vytvoření vlastního bloku zatím skutečně skládá jen z volby jména. Klikneme tedy na tlačítko Ok.

Po potvrzení názvu vlastního bloku se nám nejen objeví v příkazové kategorii „Moje bloky“ programový blok se zvoleným názvem Alarm, ale i na pracovní ploše se objeví hlavička definuj Alarm. K této hlavičce můžeme

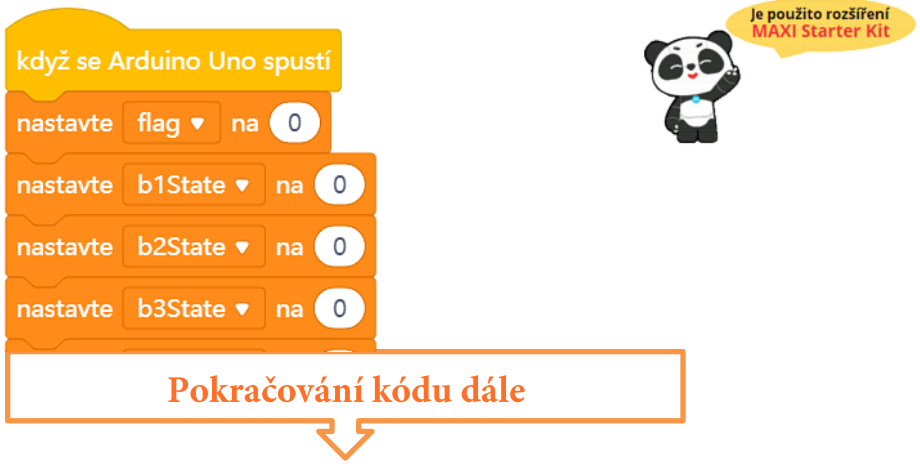
přichycovat příkazové bloky, které se mají vykonávat při zavolání programového bloku `Alarm`. Je to úplně stejné jako při tvorbě běžného kódu, kdy příkazy vkládáme pod hlavičku 

.

**Krok 2:** V prostředí mBlock je třeba vytvořit následující podprogram. Pod hlavičku vlastního bloku `Alarm` tedy vytvoříme následující kód podprogramu, který bude ovládat aktivní bzučák. Tento kód vychází z předchozího kódu pro aktivní bzučák.


Jakmile máme definovanou „pracovní náplň“ vlastního programového bloku `Alarm`, můžeme tento programový blok použít pro vytvoření následujícího hlavního kódu programu. Pochopitelně, jakmile budeme potřebovat do programu vložit růžový programový blok `Alarm`, najdeme jej v příkazové kategorii „Moje bloky“ a pracujeme s ním stejně jako s jakýmkoliv jiným blokem.

Podobně jako jsme si definovali vlastní blok podprogramu, musíme definovat jednotlivé proměnné. Jak dále v programu uvidíme, budeme potřebovat dokonce pět proměnných. První tři, které nazveme `b1State`, `b2State` a `b3State`, budeme používat pro uložení stavu tlačítek soutěžících. Proměnnou `b4State` použijeme pro testování stisknutého čtvrtého (resetovacího) tlačítka. Pátá proměnná, nazvaná `flag`, bude sloužit pro uložení stavu, ve kterém se kvízový bzučák nachází (to si podrobněji vysvětlíme v popisu programu).




nastavte b4State na 0
opakuj stále
nastavit výstup digitálního pinu 6 na nízký
nastavit výstup digitálního pinu 7 na nízký
nastavit výstup digitálního pinu 8 na nízký
nastavit výstup digitálního pinu 10 na nízký
nastavte b4State na Načtení digitálního PULLUP pinu: 9
když b4State = 0 tak
nastavte flag na 1
nastavit výstup digitálního pinu 10 na vysoký
počkat 0.2 sekund
když flag = 1 tak
nastavte b1State na Načtení digitálního PULLUP pinu: 2
nastavte b2State na Načtení digitálního PULLUP pinu: 3
nastavte b3State na Načtení digitálního PULLUP pinu: 4
když b1State = 0 tak
nastavte flag na 0
Alarm
nastavit výstup digitálního pinu 6 na nízký
nastavit výstup digitálního pinu 7 na nízký
nastavit výstup digitálního pinu 8 na vysoký
čekej dokud není Načtení digitálního PULLUP pinu: 9 = 0
když b2State = 0 tak
nastavte flag na 0
Alarm
nastavit výstup digitálního pinu 6 na nízký
Pokračování kódu dále


*Pochopitelně nezapomeňme k tomuto hlavnímu programu přidat již dříve zmíněný podprogram Alarm ☺*

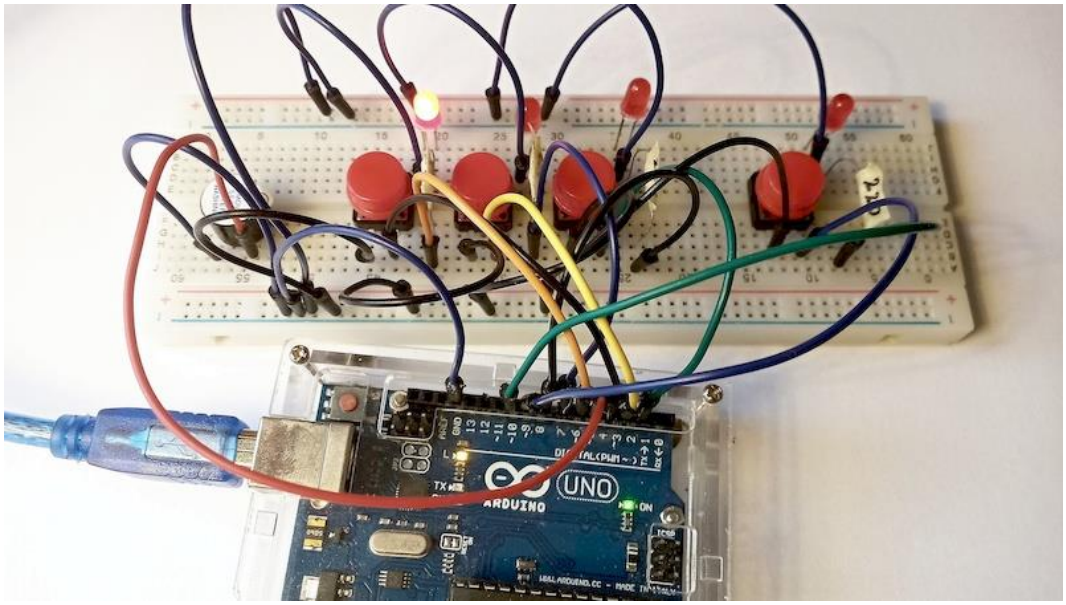
Celý program včetně podprogramu lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-06/06-Kvizovy-bzucak.mblock.

### Vysvětlení kódu

Oproti předchozím programům není úplně celý kód zanořen do nekonečné smyčky „opakuj stále“. Dříve než začne tato smyčka, jsou v programu pracovní proměnné nastaveny na výchozí hodnoty. V tomto případě to je u všech proměnných hodnota 0. V nekonečné smyčce se pak odehrává zbytek programu. Prvním krokem je nastavení výstupních digitálních pinů 6, 7, 8 a 10, na kterých jsou připojeny signalizační LED na výchozí hodnotu LOW (nízká úroveň). Pokud je stisknuto resetovací tlačítko na pinu 9, je stav čekání na odpověď signalizován rozsvícením signální čtvrté LED (pin 10) a stavová proměnná `flag` je nastavena na hodnotu 1. Pokud je proměnná `flag` rovna 1, dochází k cyklickému načítání digitálních vstupů 2, 3 a 4, na kterých jsou připojena tlačítka soutěžících, do proměnných `b1State`–`b4State`. Následuje postupné testování těchto proměnných. Pokud je některé z tlačítek stisknuto dojde ke zvukovému signálu (podprogram `Alarm`) a k rozsvícení příslušné LED a ke zhasnutí LED soupeřů. Zároveň je změněna hodnota proměnné `flag`, aby nadále již nedocházelo k načítání tlačítek soutěžících. Program tedy opět čeká na stisknutí čtvrtého (resetovacího) tlačítka, kdy dojde k rozsvícení všech LED soutěžících a hlavně k nastavení proměnné `flag` na hodnotu 0, která povolí načítání ostatních tlačítek.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Nejprve zmáčkneme tlačítko 4, tím začne soutěž. Pokud prvně zmáčkneme některé z tlačítek soutěžících, rozsvítí se odpovídající LED a pípne bzučák. Poté je třeba opět zmáčknout tlačítko 4 pro resetování.



# Využití analogových vstupů

## (*analogově-digitální převodník*)

# Lekce 7 – Interaktivní tekoucí LED světla

## Úvod

V minulých lekcích jsme se naučili ovládat digitální výstupy. V následujících lekcích si ukážeme možnost načítání analogového vstupu – kupříkladu v této lekci využijeme potenciometr, kterým budeme měnit interval blikání LED.

Když už jsme se příliš nevytáhli v originalitě hardwarové části zapojení, zkusíme to dohnat v softwarové části. V obslužném programu využijeme nejen znalost proměnných, ale program si zpřehledníme pomocí techniky využití podprogramu. Tato lekce by nám tedy měla ukázat nejen možnost načítání analogového vstupu modulu Arduino, ale i možnost, jak vhodným způsobem výrazně zkrátit blokový program. I když se tento program bude chovat trochu jinak, než se choval program v lekci „LED tekoucí potok“, i zde se bude světlo pohybovat po LED tam a zpět. Rozdílem zde jen bude to, že vždy svítí jen jedna LED a světlo tedy bude po sloupci LED přebíhat nahoru a dolů, zatímco předtím se postupně rozsvěcela a zhasínala celá řada.

Kdybychom chtěli dosáhnout stejného efektu, jako byl v lekci LED tekoucí potok, bylo by to možné poměrně jednoduchou úpravou hlavního podprogramu. Takže bychom dokázali napsat předchozí program poměrně jednodušším způsobem. Následující program by tedy měl i ukázat, že tvorba jednoduchého a funkčního kódu není jen o znalosti daného programovacího jazyka, ale i na určitém citu pro tvorbu algoritmu.

## Použité komponenty

- modul Arduino
- USB kabel
- 8× LED
- 8× rezistor (220 Ω)
- potenciometer
- nepájivé pole
- vodiče pro nepájivé pole

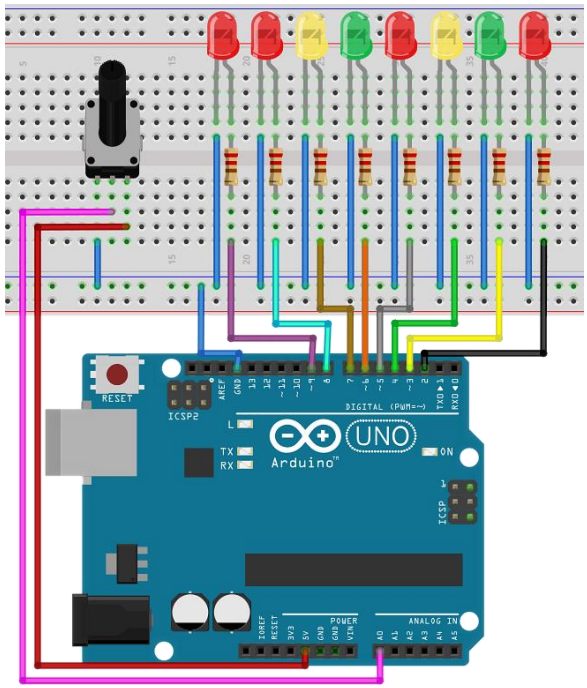
## Princip

Podobně jako v lekci „LED tekoucí potok“ použijeme osm LED, které budeme postupně rozsvěcet a zhasínat. Doplněním dříve použitého zapojení je použití potenciometru, který připojíme mezi napájecí napětí a jeho jezdec připojíme k analogovému pinu `A0` modulu Arduino. Potenciometr pak slouží jako dělič napětí, napěťovou hodnotu načtenou analogovým vstupem využijeme pro určení rychlosti běhu programu. Ve výsledku tedy budeme pomocí natáčení potenciometru nastavovat celkovou rychlost světelného efektu.

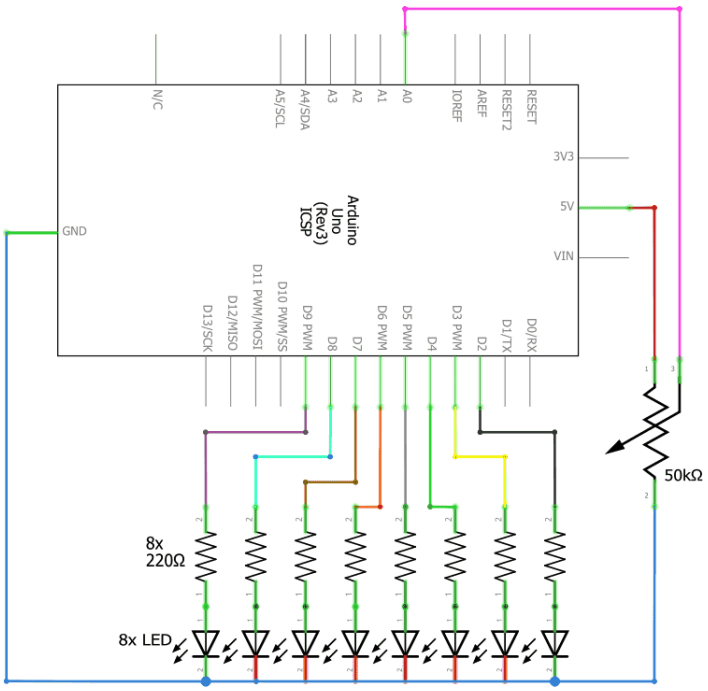
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Potenciometr zapojíme krajními piny k napájecímu napětí +5 V a zemi GND. Prostřední pin (jezdec) připojíme k analogovému pinu `A0`. LED zapojíme katodou (*kratší vývod*) k zemi GND a anodou (*delší vývod*) přes rezistor 220 Ω k digitálním pinům `2–9` modulu Arduino.

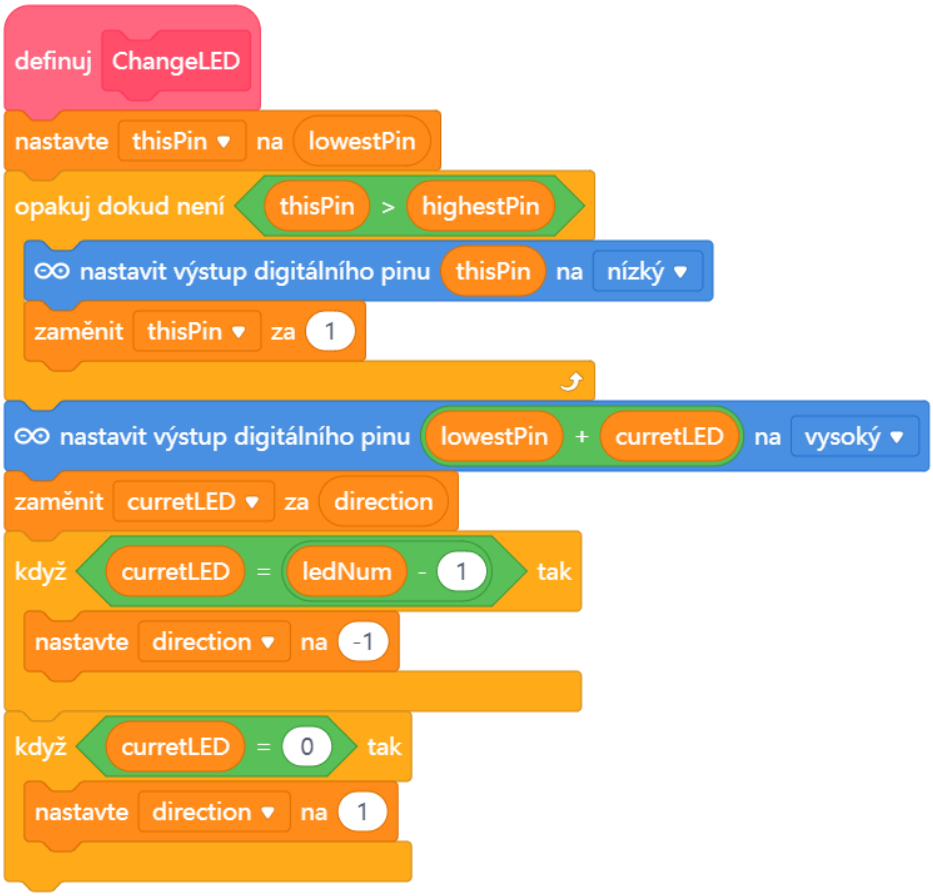
**Blokové schéma**



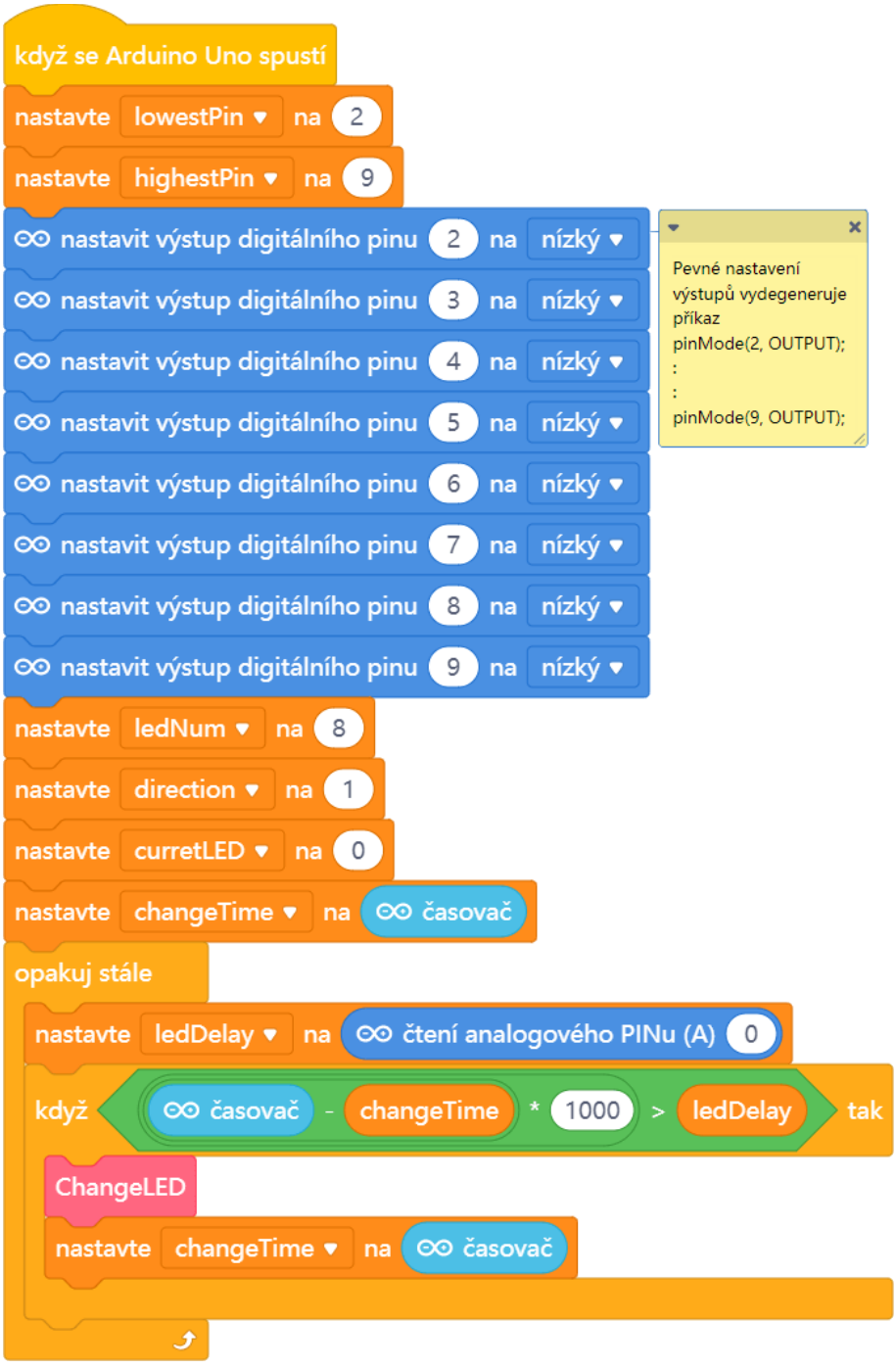
**Elektronické schéma**




**Krok 2:** V prostředí mBlock vytvoříme následující program. Pochopitelně nejdříve musíme definovat proceduru `ChangeLED`, která se stará o rozsvěcení připojených LED.



Následuje hlavní program:




Celý program včetně podprogramu lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-07/07-Interaktivni-tekouci-LED-svetla.mblock.

## Vysvětlení kódu

Proměnné `lowestPin` a `highestPin` určují rozmezí čísel pinů s připojenými LED. Proměnná `ledNum` pak určuje celkový počet připojených LED. Proměnná `direction` určuje směr, ve kterém budeme LED

rozsvěcet a zhasínat. Proměnné `currentLED` a `thisPin` určují číslo LED/pinu, kde bude LED aktuálně rozsvícena. Rychlost efektu bude nastavována pomocí proměnných `changeTime` a `ledDelay`.

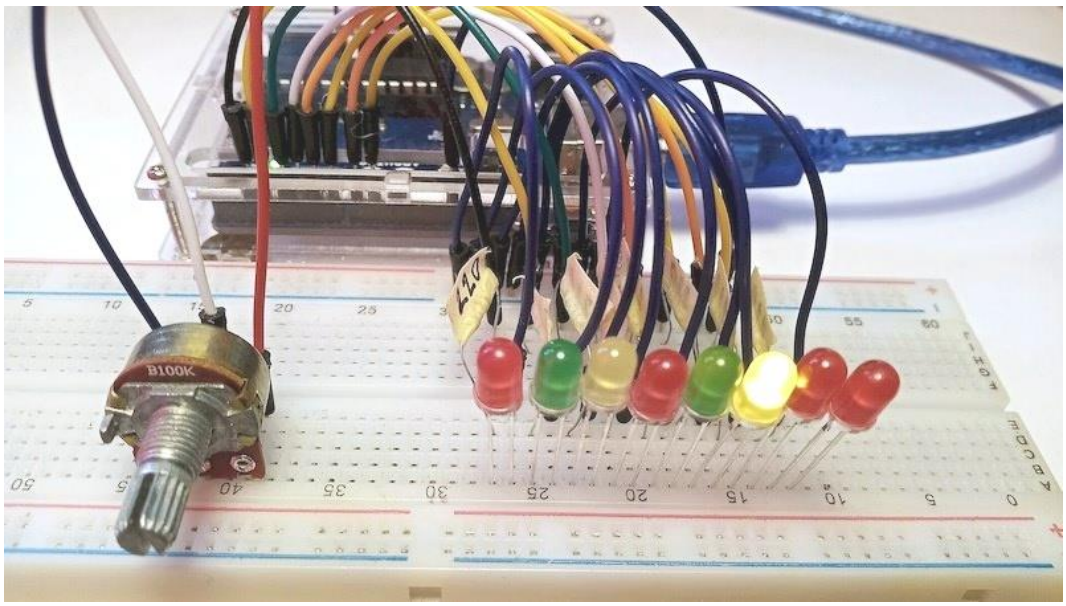
Úvodní část programu nastavuje použité proměnné na výchozí hodnoty, také jsou zde nastavené jednotlivé digitální výstupy s LED pomocí triku se zápisem výstupní hodnoty (*popsáno dříve*). Za zmínku stojí použití časovače, jehož hodnotu (*počet sekund od zapnutí modulu Arduino*) uchováváme v proměnné `changeTime`.

V hlavní nekonečné smyčce načítáme do proměnné `ledDelay` hodnotu napětí na pinu `A0` (*rozsah 0–1023*). Hodnotu využijeme v podmínce testu přírůstku časovače. Pokud je přírůstek časovače větší (rozdíl hodnoty časovače a proměnné `changeTime` je větší než `ledDelay`), provede se podprogram `ChangeLED` a nastaví se nová hodnota proměnné `changeTime`, aby mohl být opět testován přírůstek časovače.

V podprogramu `ChangeLED` jsou nejdříve nastavením digitálních pinů na nízkou hodnotu zhasnuty všechny LED. Aktuální LED pak bude rozsvícena pomocí proměnné `currentLED`, jejíž hodnota je pak buď zvýšena, nebo snížena o jedničku. Zda bude hodnota proměnné `currentLED` zvýšena, nebo snížena, rozhoduje proměnná `direction`, která má hodnotu buď +1, nebo –1. V podprogramu se podmínkami řeší testování krajních hodnot proměnné `currentLED`, pak je změněna hodnota proměnné `direction`.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Měli bychom vidět osm LED, které se postupně zleva doprava a zpět rozsvěcí. Nastavení potenciometru ovládá rychlost celého efektu.



# Lekce 8 – Fotorezistor

## Úvod

Fotorezistor je proměnný rezistor, jehož odpor je závislý na míře jeho osvětlení. Odpor fotorezistoru se snižuje s rostoucí intenzitou dopadajícího světla, takže jej můžeme využít pro měření intenzity světla nebo osvětlení. Stejně tak fotorezistor můžeme využít při stavbě světlocitlivého detektoru, který se aktivuje světlem či tmou.

## Použité komponenty

- modul Arduino
- USB kabel
- fotorezistor
- rezistor (10 kΩ)
- 8× LED
- 8× rezistor (220 Ω)
- nepájivé pole
- vodiče pro nepájivé pole

## Princip

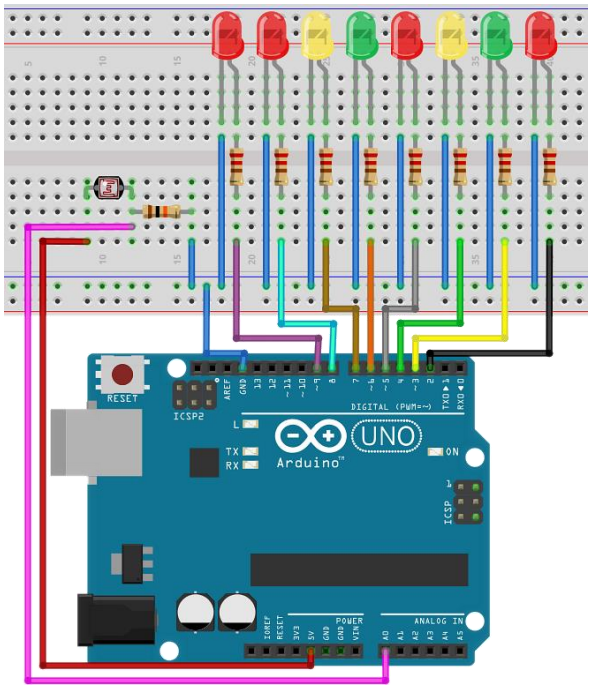
Jak bylo uvedeno v úvodu, odpor fotorezistoru se mění na základě intenzity dopadajícího světla. Když se intenzita tohoto světla zvýší, odpor se sníží a naopak. V tomto experimentu zapojíme fotorezistor společně s pevným rezistorem tak, abychom vytvořili tzv. dělič napětí. Podobné zapojení jsme kupříkladu viděli při použití potenciometru nebo ještě uvidíme u detektoru plamene. Díky změně odporu fotorezistoru se bude měnit napětí na analogovém pinu `A0`. Toto napětí budeme načítat vestaveným AD převodníkem modulu Arduino. Pro zobrazení změny velikosti vstupního napětí na fotorezistoru, které odpovídá míře osvětlení, použijeme výstup na osmici LED. Čím větší bude intenzita světla, tím více LED se rozsvítí. Při dosažení maximální intenzity světla, se rozsvítí všechny LED. V případě nízké úrovně světla se nerozsvítí nic.

Jelikož vstupní interval načtených hodnotu potřebujeme rozdělit do osmi stupňů, které budou rozsvěcet patřičné LED, můžeme buď použít systém několika podmínek, ale zde lépe využijeme příkazový blok mapování zadané hodnoty na předem stanovený interval (více v popisu kódu). Využití tohoto příkazového bloku se velmi často využívá nejen pro převod vstupní hodnoty analogových vstupů (*interval 0–1023*) na „užitečný“ rozsah.

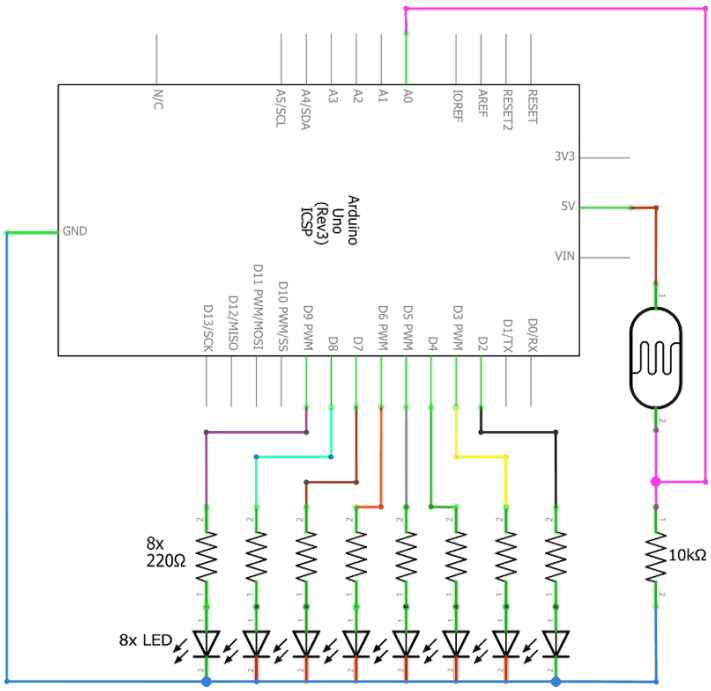
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Fotorezistor zapojíme k napájecímu napětí +5 V a k pinu analogovému pinu `A0`, který přes rezistor 10 kΩ připojíme k zemi GND. LED zapojíme katodou (*kratší vývod*) k zemi GND a anodou (*delší vývod*) přes rezistor 220 Ω k digitálním pinům `2–9` modulu Arduino.

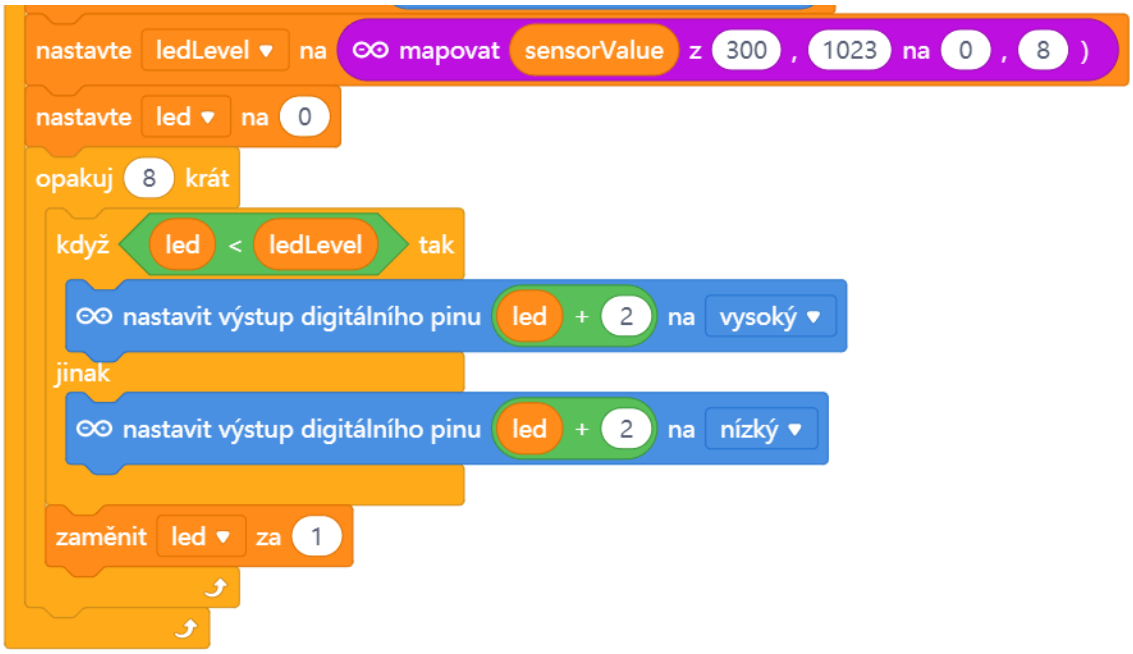
**Blokové schéma**



**Elektronické schéma**




**Krok 2:** V prostředí mBlock vytvoříme následující program:




Program lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-08/08-Fotorezistor.mblock.

## Vysvětlení kódu

V první části jsou jednotlivé piny s připojenými LED nastaveny na nízkou hodnotu (LOW), čímž jsou nejen všechny LED zhasnuty, ale zároveň deklarovány příslušné digitální piny do režimu digitálních výstupů. Po tomto kroku je již možné se na tyto výstupy odkazovat z hodnot proměnných (*vysvětleno v předešlé lekci: LED tekoucí potok*).

V hlavní nekonečné smyčce „opakuj stále“ je nedříve načten analogový vstup `A0` do proměnné `sensorValue`. Proměnná `sensorValue` je nadále „přemapována“ na osmistupňovou hodnotu (*0–7*), která je potřeba rozsvícení daných LED. Tato hodnota je uložena do proměnné `ledLevel`. Všimněme si, že hodnoty `sensorValue` nejsou mapovány od nuly (viz fialový blok „mapovat“), ale až od hodnoty 300. Důvod nalezneme v elektronickém schématu. Zatímco v jedné z předchozích lekcí potenciometr reguloval vstupní napětí na vstupu `A0` v rozmezí 0–5 V, zde je minimální hodnota dána maximálním odporem fotorezistoru a zároveň pevným rezistorem 10 kΩ zapojeným proti zemi GND.

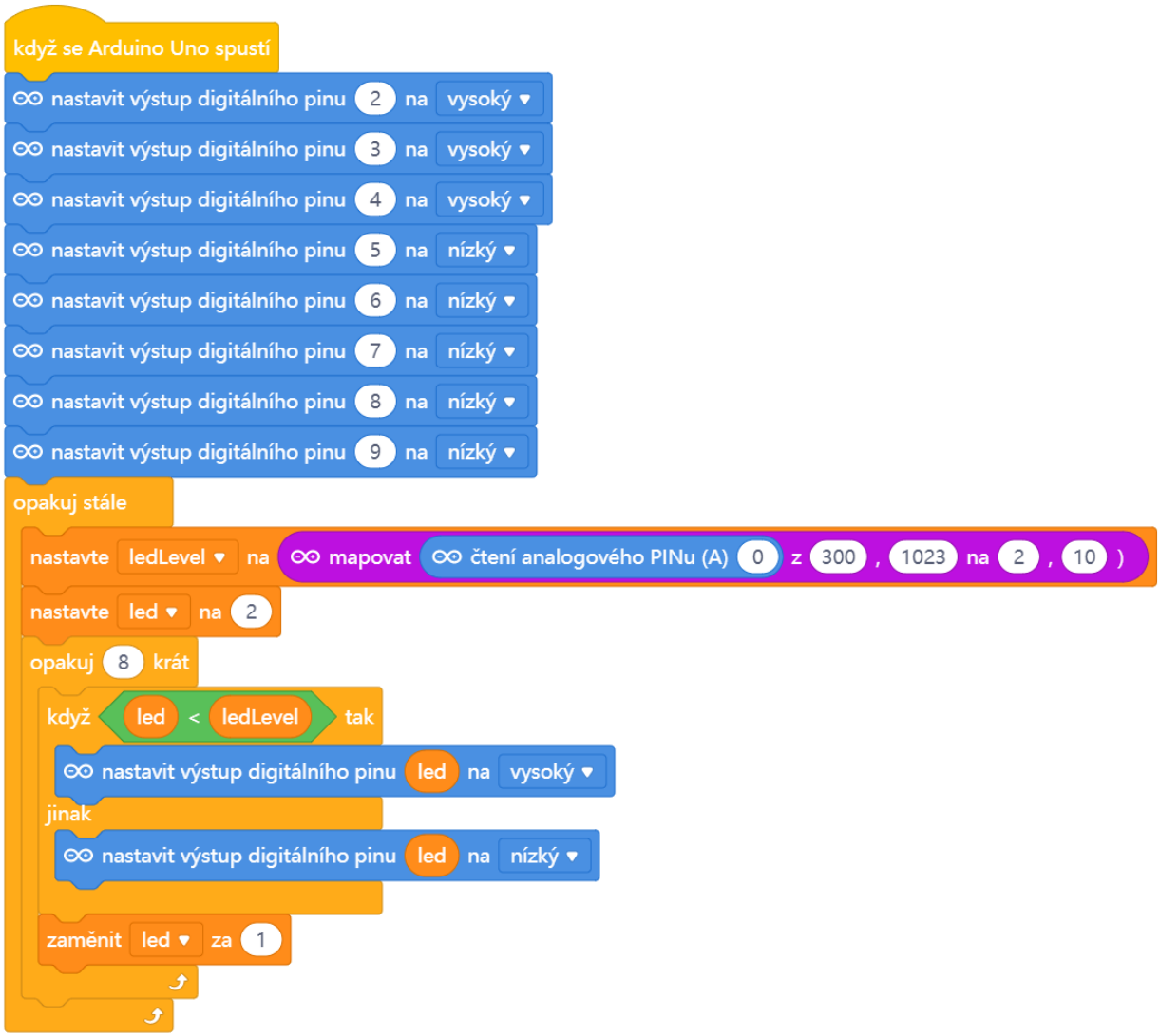
Proměnná `led` je využita pro postupné rozsvěcení LED. V cyklu, který se opakuje 8×, je postupně zvyšována proměnná `led` od 0 do 7. Podmínka testuje, zda je proměnná `led` (*adresa aktuální LED*) vyšší/nižší než hodnota proměnná `ledLevel` (*přemapovaná hodnota vstupu*). Pokud je hodnota proměnné `led` nižší, dojde k rozsvícení LED nastavením daného pinu na vysokou hodnotu HIGH, jinak je LED zhasnuta nízkou úrovní LOW. Tím dojde k efektu rozsvěcení LED stupnice podle osvícení fotorezistoru.

**TIP:**

Kdybychom chtěli v tomto kódu snížit počet proměnných, mohli bychom vypustit proměnnou `sensorValue` tím, že bychom příkazový blok načtení analogové hodnoty pinu `A0` rovnou vložili do bloku „mapování“ – viz následující ukázka:

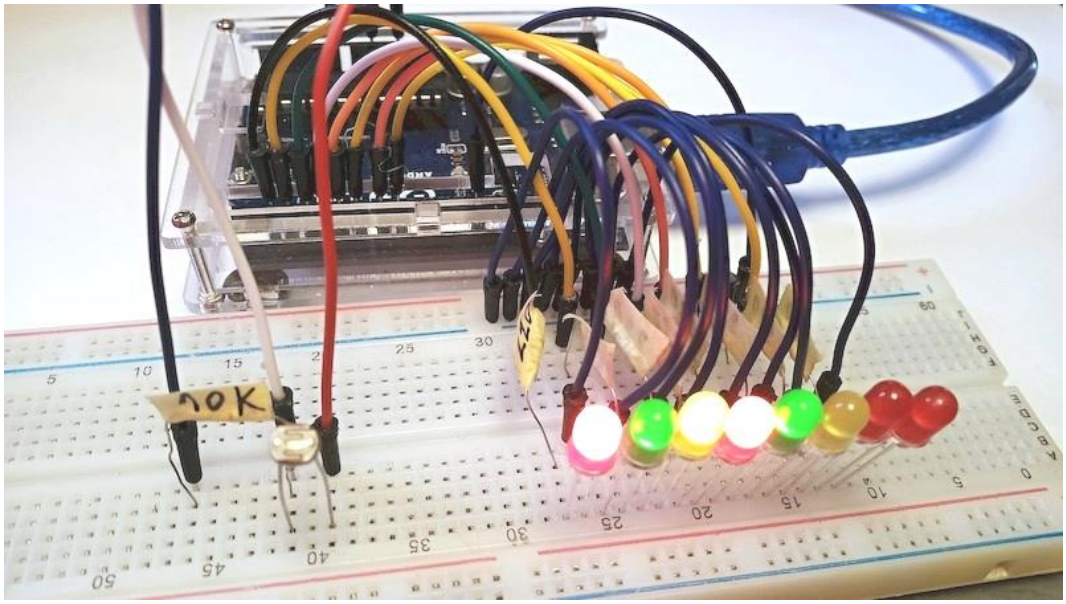

A když už optimalizujeme kód, mohli bychom udělat další úpravu. Nemapovat vstup na interval 0–7, ale na rozsah 2–9. Tím pádem by hodnoty ze senzoru rovnou odpovídaly hodnotám odpovídajícím číslům pinů jednotlivých LED. Proměnná `led` by tak v podmínce také musela odpovídat pinům a nikoliv pořadí LED. Ale to není problém – před cyklem osminásobného opakování prostě proměnnou `led` nastavíme na hodnotu 2 (*číslo pinu nejnižší LED*). Tím pádem si ušetříme přičítání dvojky v příkazech ovládání výstupů s LED.

Celkové upravený kód by pak mohl vypadat následujícím způsobem:



**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Postupně budeme měnit intenzitu osvícení fotorezistoru a měli bychom sledovat, jak se LED postupně rozsvěcí. Čím více posvítíme na fotorezistor, tím více větší počet LED se rozsvítí. Jakmile fotorezistor zatemníme, všechny LED zhasnou.




**POZNÁMKA:**
Kromě výše uvedeného experimentu můžeme vyměnit fotorezistor za mikrofon a pozorovat, jak nám budou LED signalizovat sílu zvuku. Čím větší bude síla zvuku, tím více LED se rozsvítí.

# Lekce 9 – Senzor plamene

## Úvod

Senzor plamene funguje podobně jako předchozí fotorezistor jen s tím rozdílem, že reaguje na infračervené světlo emitované z plamene. Pochopitelně fyzikální princip obou součástek je trochu jiný, ale pro základní pochopení nám tato představa stačí. Důležitou informací, kterou ale musíme k rozdílu mezi fotorezistorem a senzorem plamene vědět, je to, že fotorezistor můžeme zapojit libovolně, zatímco u senzoru plamene je třeba správně dodržet polaritu. My si v naší lekci využijeme senzor plamene k vytvoření jednoduchého detektoru plamene, tedy pro základ případného systému varování před požárem.

## Použité komponenty

- modul Arduino
- USB kabel
- rezistor 10 kΩ
- senzor plamene
- nepájivé pole
- vodiče pro nepájivé pole

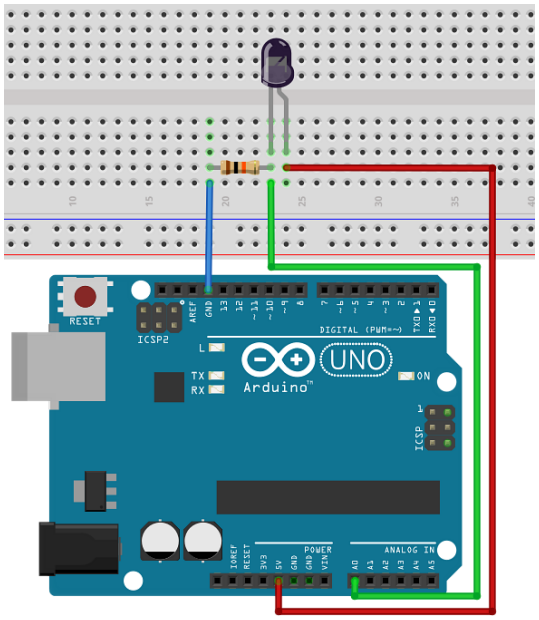
## Princip

Existuje několik typů senzoru plamene. V tomto experimentu budeme využívat infračervené čidlo plamene, které dokáže rozpoznat infračervené světlo o vlnové délce od 700 nm do 1000 nm. Tato infračervená sonda převede změny vnějšího infračerveného světla do elektrické podoby. Kratší vývod senzoru je katoda, ten druhý je anoda. Katodu připojíme k napájení 5 V a anodu spojíme přes rezistor 10 kΩ s pinu zemnění GND. Anodu ještě jedním vodičem připojíme k analogovému pinu A0 modulu Arduino, viz schéma dále. Tím rezistor a senzor plamene opět vytvoří dělič napětí, jehož výstup bude přes analogový pin modul Arduino načítat.

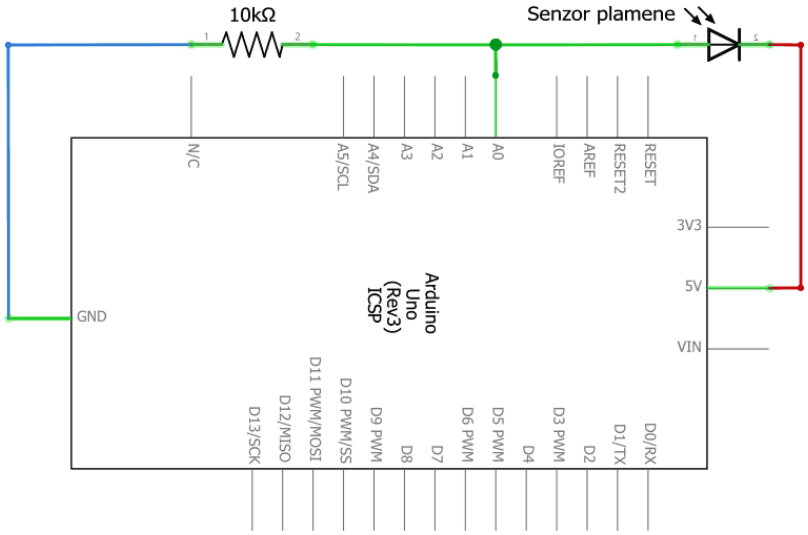
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Senzor plamene zapojíme katodou (*kratší vývod*) k napájecímu napětí +5 V a anodou (*delší vývod*) k pinu analogovému pinu `A0`, který přes rezistor 10 kΩ připojíme k zemi GND.

**Blokové schéma**



**Elektronické schéma**




**Krok 2:** V prostředí mBlock vytvoříme následující program:


Program lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-08/08-Fotorezistor.mblock.

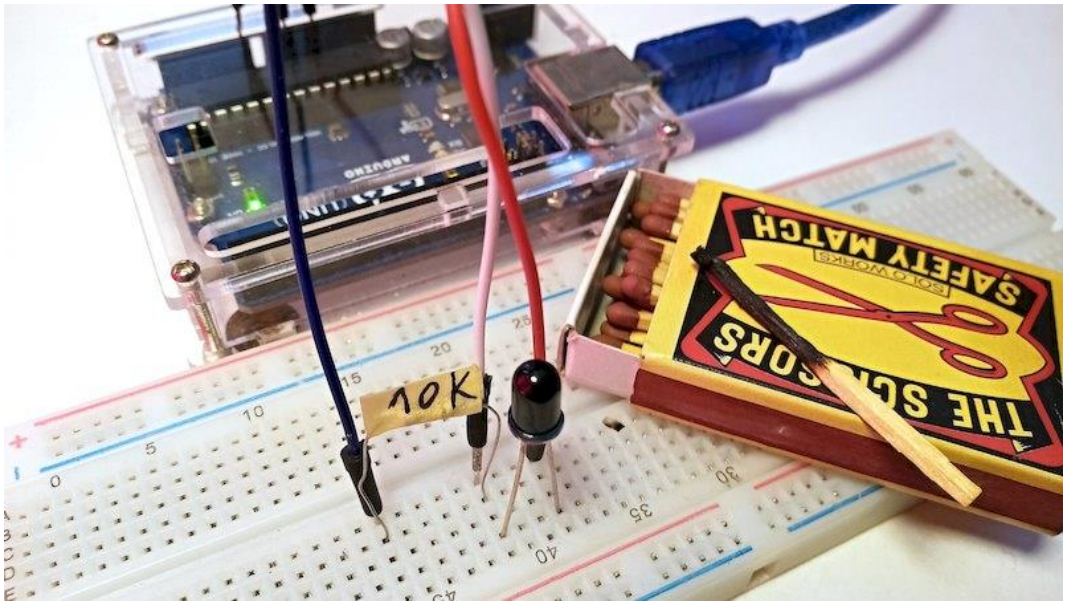
### Vysvětlení kódu

Na rozdíl od předchozích programů zde se opět dostáváme k jednodušším programům. V hlavní nekonečné smyčce „opakuj stále“ nejdříve načteme hodnotu analogového pinu `A0` (*hodnoty 0–1023*) do proměnné `val`. Následuje podmínka, která testuje načtenou tuto hodnotu. Je-li hodnota v proměnné `val` vyšší než zvolená

hladina (*zde zvoleno 30*), dojde k nastavení digitálního pinu 13 na vysokou úroveň HIGH, jinak je nastavena na nízkou úroveň LOW. Protože k pinu 13 je na modulu Arduino připojena vestavená LED, bude se podle zvolené výstupní úrovně rozsvěcet/zhasínat.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Když nyní přiblížíme zapálený zapalovač poblíž čidla *(POZOR, ať se čidlo teplem neroztaví!)*, zabudovaná LED spojená s pinem 13 na modulu Arduino se rozsvítí. Po oddálení plamene LED zhasne. Už známe modul relé, takže bylo by možné jej k pinu 13 připojit a tak začít tak spínat třeba nějaký poplašný systém. Nebo již také zvládáme pomocí modulu Arduino ovládat bzučák, takže případnému rozšíření zapojení z této lekce se meze nekladou!



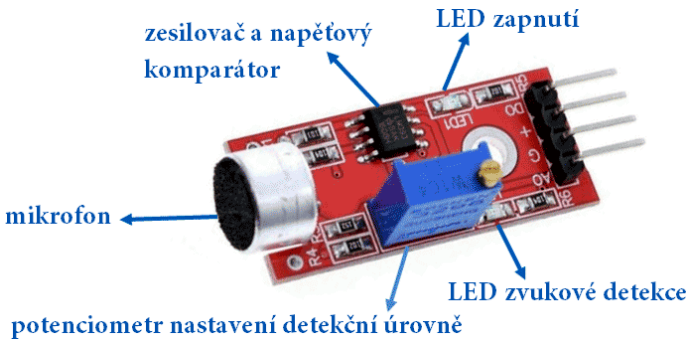
# Lekce 10 – Zvukový senzor

## Úvod

V závěru lekce věnované fotorezistoru jsme se zmínili, že by bylo možné načítat hladinu zvuku, tak si to nyní vyzkoušíme. Pro tuto lekci ale nepoužijeme obyčejný mikrofon, ale použijeme celý zvukový senzor. Zvukový senzor můžeme využít jak v režimu načítání analogového signálu, tak i případě načítání digitálního vstupu.

Náš zvukový senzor má základní dva výstupy:

1. **AO**: Analogový výstup, který se používá pro výstup napěťových signálů z mikrofonu v reálném čase. Bylo by možné jej kupříkladu používat pro digitalizaci zvuku.
2. **DO**: Digitální výstup, jehož stav je určen podle určité prahové hodnoty, tu můžeme nastavit pomocí potenciometru. Bude-li zesílená úroveň signálu z mikrofonu vyšší než zvolená hladina, bude výstup DO na vysoké úrovni HIGH, jinak bude na nízké úrovni LOW. To by se dalo kupříkladu využít jako detektor hlasitého hluku – např. tříštění skla apod.

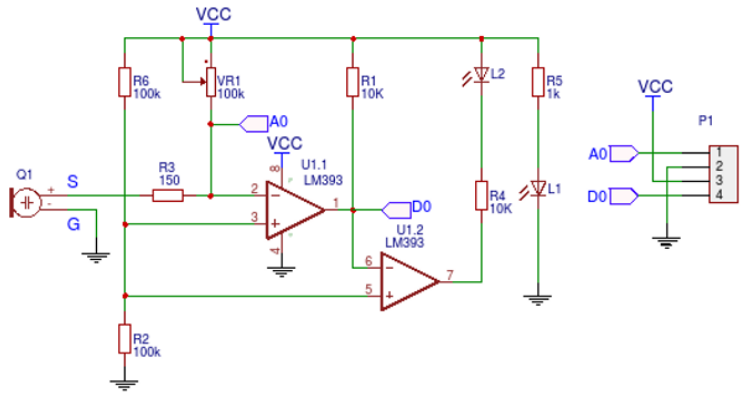



## Použité komponenty

- modul Arduino
- USB kabel
- zvukový senzor
- vodiče typu „samec-samice“

## Princip

Modul mikrofonu umí vstupní audiosignál převést na analogový elektrický signál. Tento signál je současně srovnáván s nastavenou detekční hodnotou. Při překročení detekční hodnoty, komparátor nastaví digitální výstup do vysoké úrovně HIGH. Tento princip můžeme vidět na následujícím obrázku, na kterém je elektrické schéma zvukového modul




Signál ze vstupního mikrofonu je nejen rovnou vyveden na výstup „Analog Out“ (AO), ale zároveň vstupuje i na obvod LM393, který slouží jako komparátor napětí. Pokud je ticho, odpor mikrofonu je velmi vysoký a napětí na vstupu 2. je blízko k napájecímu napětí (Ucc), na výstupním pinu „Digital Out“ (DO) je nízká úroveň LOW a LED (L2) nesvítí. Nastane-li nějaký hluk, odpor mikrofonu klesne, tím klesne napětí na vstupu 2 a ve chvíli, kdy bude napětí na vstupu 2 nižší než na vstupu 3, výstupní digitální pin DO se změní na vysokou úroveň HIGH a LED (L2) se rozsvítí.

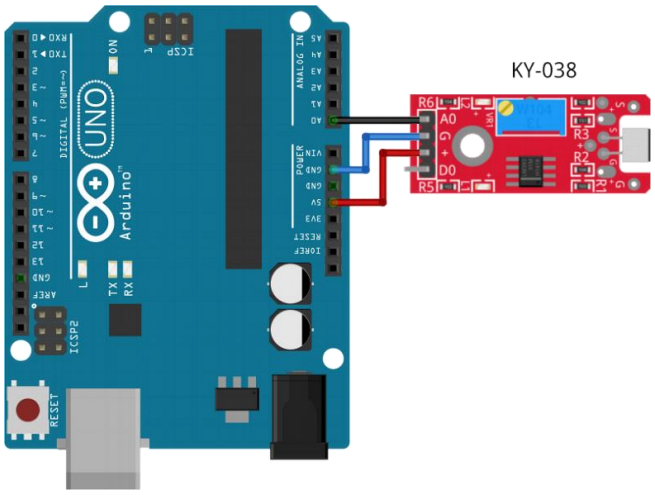
Pomocí potenciometru lze posouvat napěťovou úroveň na vstupu 2 a tedy i úroveň zvukového signálu, kdy dojde k „překlopení“ digitálního výstupu.

V této lekci budeme používat jen analogový výstup tohoto obvodu. Na druhou stranu asi nebylo špatné si vysvětlit jeho funkci podrobněji, protože možnosti modulu Arduino nekončí jen u lekcí této experimentální sady. Fantazii vývojářů se meze nekladou!

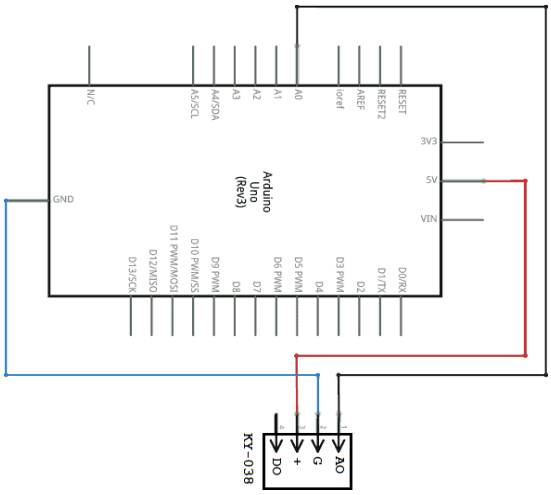
## Postup experimentu

**Krok 1:** Vytvoř obvod podle následujícího obrázku. Spojení mezi zvukovým senzorem a modulem Arduino zachycuje následující tabulka:

**Blokové schéma**




**Elektronické schéma**



| zvukový senzor | modul Arduino |
|---|---|
| AO | A0 |
| G | GND |
| + | 5 V |

**Krok 2:** V prostředí mBlock vytvoříme následující program:

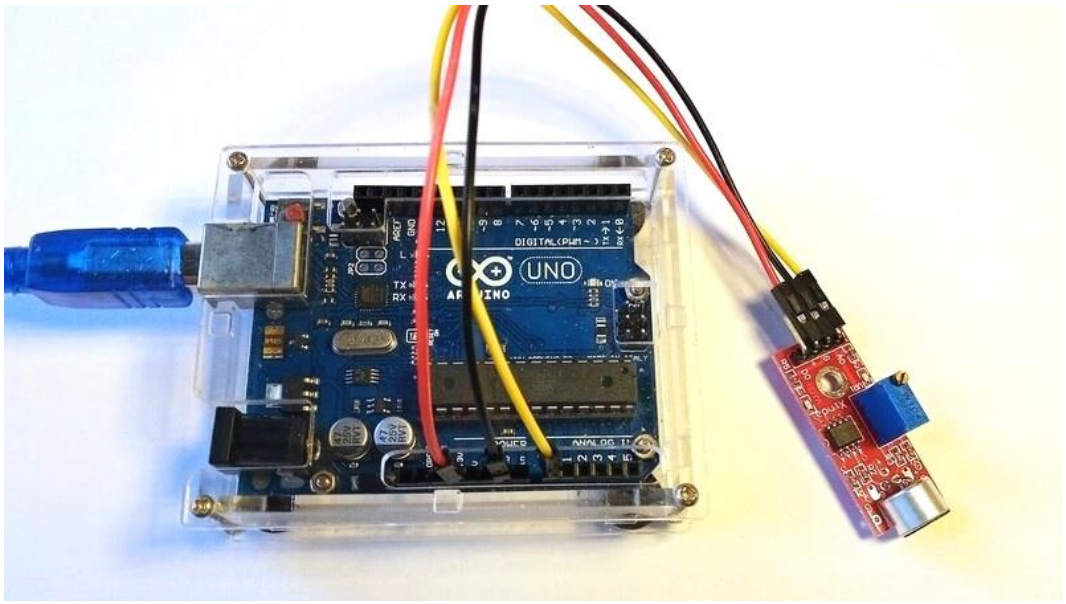

Program lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-10/10-Zvukovy-senzor.mblock.

### Vysvětlení kódu

Přestože je zvukový senzor již senzor s poměrně složitým vnitřním zapojením, z hlediska programového kódu k němu přistupujeme úplně stejně jako k předešlým senzorům. Opět v nekonečné smyčce „opakuj stále“ načítáme analogový vstup `A0` a získanou hodnotu testujeme v podmínce, která pak následuje. V této podmínce je porovnána získaná hodnota (uložená v proměnné `value`) s mezní hodnotou, při které má modul Arduino vyhodnotit hladinu zvuku jako vysokou a sepnout digitální výstup `13`. Pokud je tedy hodnota proměnné `value` nižší než zvolených 25 je výstupní digitální pin `13` na nízké úrovni (LOW) a vnitřní LED modulu Arduino je zhasnutá. Při překročení hodnoty 25 se výstup `13` nastaví do vysoké úrovně (HIGH), tím se vestavěná LED modulu Arduino rozsvítí.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Pokud budeme mluvit hlasitě nebo dokonce řvát do mikrofonu, LED vestavěná na modulu Arduino (připojená k pinu `13`) se rozsvítí.



# Lekce 11 – Ovládání zvuku pomocí světla

## Úvod

Již jsme se naučili, jak pracovat s fotorezistorem, stejně tak jsme zvládli práci s bzučákem. Abychom si ukázali, že všechny zde uvedené lekce je možné kombinovat a být tedy inspirací pro vymýšlení dalších zapojení, zkusíme nyní lekci fotorezistoru a bzučáku vzájemně zkombinovat. Ukážeme si, jak lze ovládat bzučák pomocí fotorezistoru tak, aby pípal v různých frekvencích.

## Použité komponenty

- modul Arduino
- USB kabel
- fotorezistor
- aktivní bzučák
- rezistor (10 kΩ)
- nepájivé pole
- vodiče pro nepájivé pole

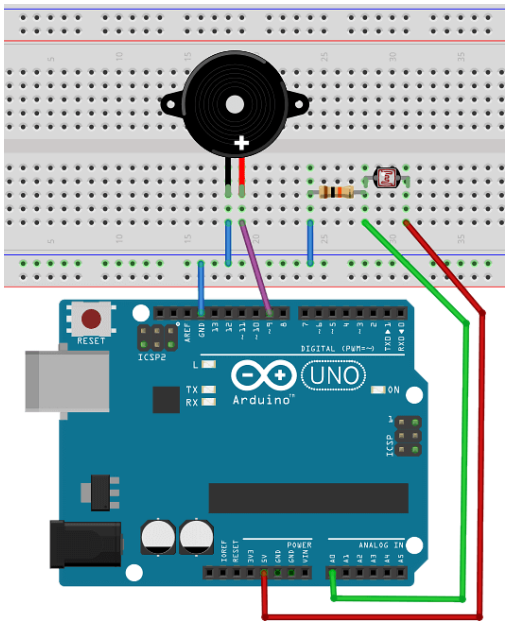
## Princip

Když posvítíme na fotorezistor, bude intenzita dopadajícího světla větší, odpor fotorezistoru se sníží a naopak. V tomto pokusu je výstup fotorezistoru připojen na pin A0 modulu Arduino, kde je zpracován ADC převodníkem. Získanou hodnotu použijeme jako parametr čekací funkce v kódu, než se ozve bzučák. Pokud je dopadající světlo silné, je výsledná hodnota vyšší, což znamená, že bude bzučák pípat pomalu. Je-li světlo slabé, výsledná hodnota je nižší a bzučák bude pípat rychleji.

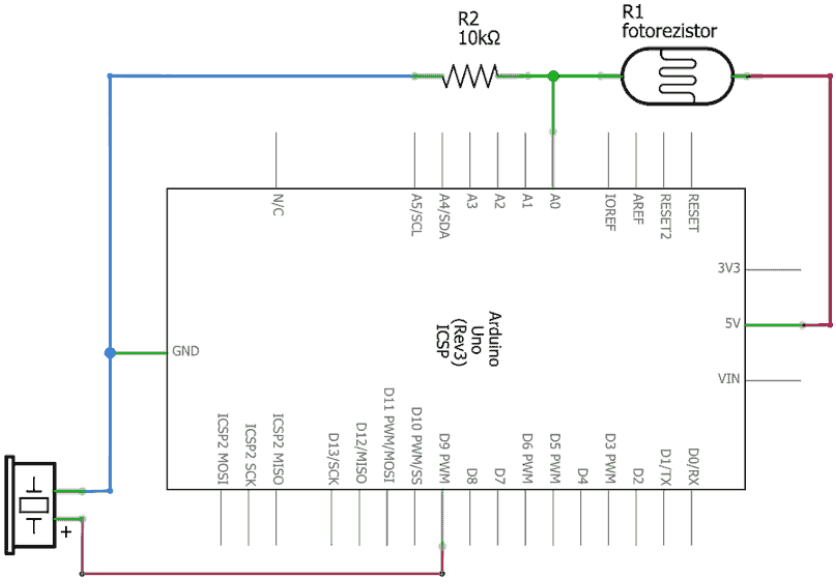
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu.

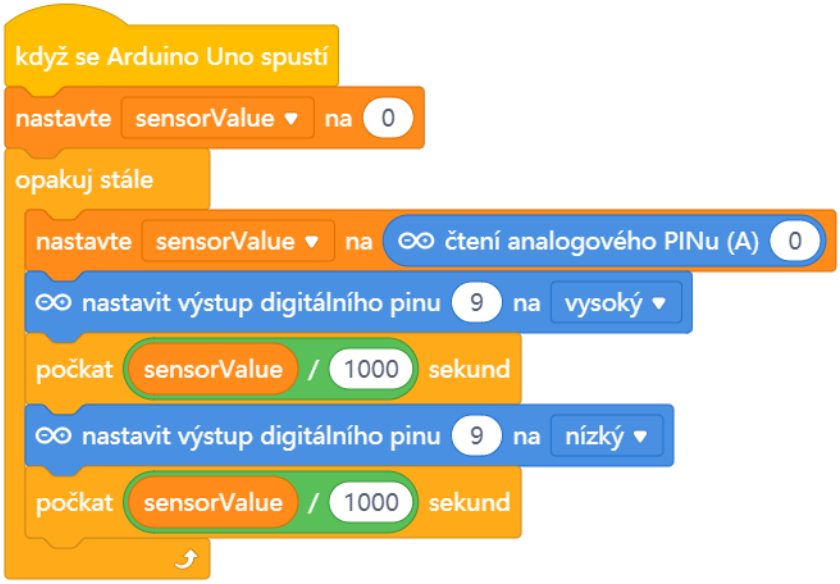
**Blokové schéma**



**Elektronické schéma**



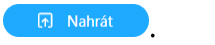
**Krok 2:** V prostředí mBlock sestavíme následující program:




Program lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-11/11-Ovladani-zvuku-pomoci-svetla.mblock.

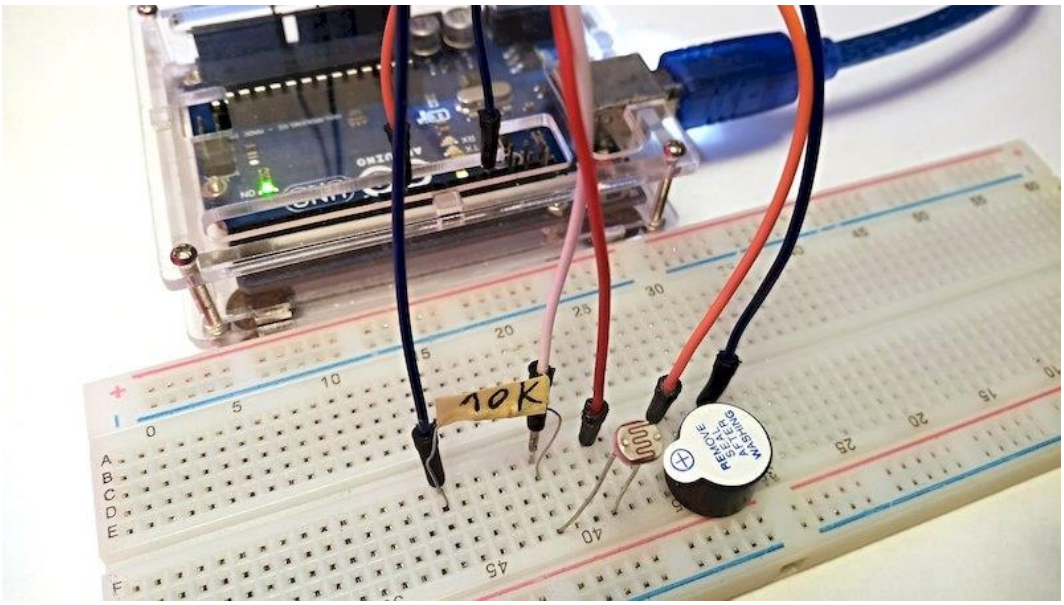
### Vysvětlení kódu

Jak již naznačují jednotlivé bloky tohoto kódu je jeho princip poměrně pochopitelný. V nekonečné smyčce „opakuj stále“ je načítána analogová hodnota na pinu A0, která odpovídá osvitu fotorezistoru. Čím je fotorezistor více osvícen, tím je díky jeho zapojení v tzv. děliči napětí na pinu A0 vyšší, tedy je vyšší i vstupní hodnota. Tato vstupní hodnota je uložena do proměnné `sensorValue` a následně použita jako čekací doba pro generování zvuku. To vidíme v následujících blocích, kde nejdříve nastavíme výstupní digitální pin 9 na vysokou úroveň (HIGH). Tím zapneme připojený aktivní bzučák. Následuje blok čekání – jelikož vstupní hodnota pinu A0 může být v rozmezí 0–1023, ale blok čekání vyžaduje vstupní hodnotu v sekundách, je hodnota proměnné `sensorValue` vydělena tisícem. Po uplynutí této čekací doby je výstupní pin 9 nastaven na nízkou úroveň (LOW), čímž je aktivní bzučák umlčen. Opět se čeká dobu určenou tisícinou hodnoty proměnné `sensorValue`. Zapínáním a vypínáním bzučáku a setrváním v tomto stavu dle načtené vstupní hodnoty tak generuje pípání, jehož četnost je odvislá od osvětlení fotorezistoru.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

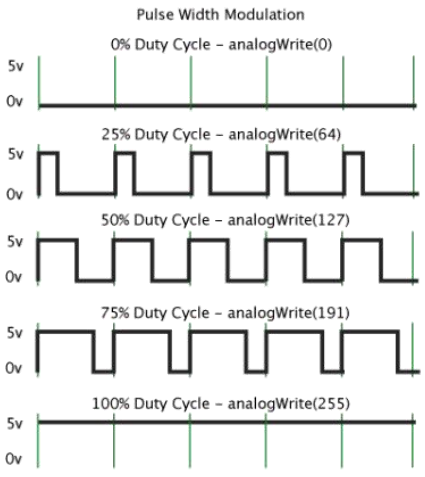
.

**Krok 4:** Pokud postavíte fotorezistor do tmy, bude bzučák pípat rychle. Osvětlíme-li fotorezistor, bude pípat pomalu. Funkčnost zařízení můžeme ověřit zakrýváním fotorezistoru rukou.



# Využití PWM výstupů

+



# Lekce 12 – Řízení LED pomocí PWM

Pojďme zkusit v této lekci opět něco trochu jednoduššího – postupně budeme měnit jas připojené LED. Vzhledem k tomu, že pulzující světlo vypadá trochu jako dýchání, můžeme výslednému zapojení dát kouzelné označení: „dýchající LEDka“. Požadovaného efektu dosáhneme pomocí pulzně šířkové modulace (PWM).

## Použité komponenty

- modul Arduino
- USB kabel
- LED (libovolné barvy)
- rezistor (220 Ω)
- nepájivé pole
- vodiče pro nepájivé pole

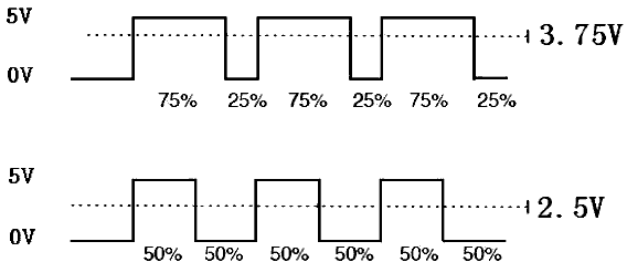
## Princip

O pulzně šířkové modulaci neboli PWM (Pulse Width Modulation) jsme si říkali v kapitole představující modul Arduino. V rychlosti si však základní informace zopakujeme. V případě PWM se jedná o přenášení signálu pomocí tzv. střídy. Střída je určena jako poměr mezi dobami stavů zapnuto/vypnuto. Přenosový signál, který nese informaci o přenášené hodnotě, může tedy nabývat hodnot zapnuto/vypnuto, což v našem případě odpovídá napěťovým hodnotám 5 V a 0 V. Omezením pro PWM je to, že přenos informace je vždy omezen na relativní vyjádření a to 0–100 %.

Tři základní parametry PWM:

1. Střída (duty cycle) – poměr mezi stavy zapnuto/vypnuto.
2. Perioda – součet jedné doby zapnuto a vypnuto.
3. Amplituda – rozsah napětí (zde 0–5 V).

Vzhledem ke svým vlastnostem je pulsně šířková modulace často využívána ve výkonové elektronice pro řízení velikosti napětí nebo výstupního výkonu. Na následujícím obrázku vidíme srovnání dvou signálů s různou střídou a výslednou hodnotou napětí, které takový signál odpovídá.

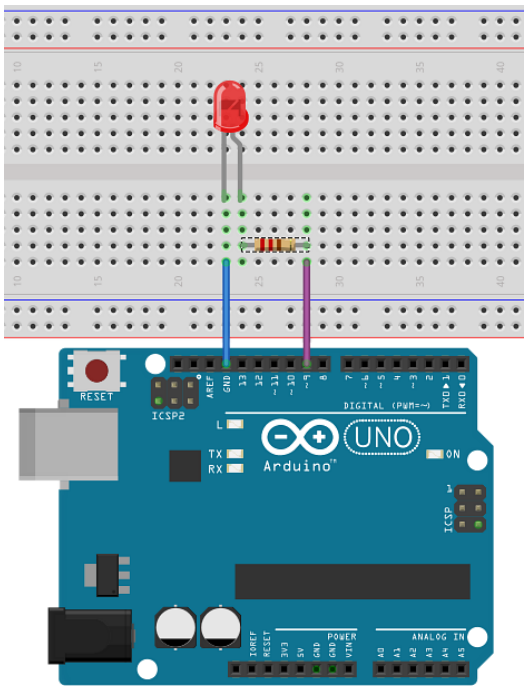



Je třeba ještě připomenou, že pokud nechceme vidět, jak LED díky zapínání/vypínání signálu bliká, musí mít PWM signál dostatečně velkou frekvenci. V případě modulu Arduino UNO je tato frekvence 490 Hz (na pinech 5 a 6 dokonce 980 Hz), což je hodnota pro náš experiment více jak dostačující.

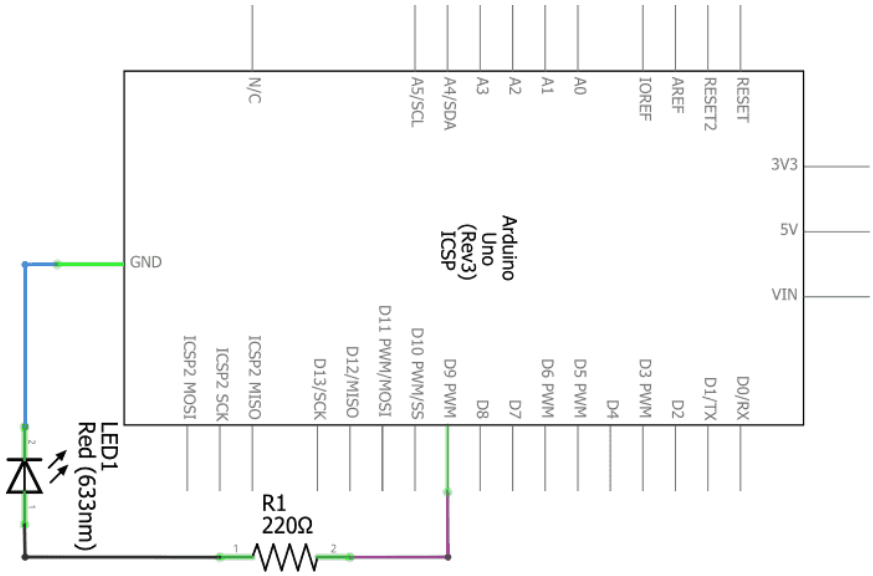
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. LED zapojíme katodou (*kratší vývod*) k zemi GND a anodou (*delší vývod*) přes rezistor 220 Ω k digitálním pinům modulu Arduino.

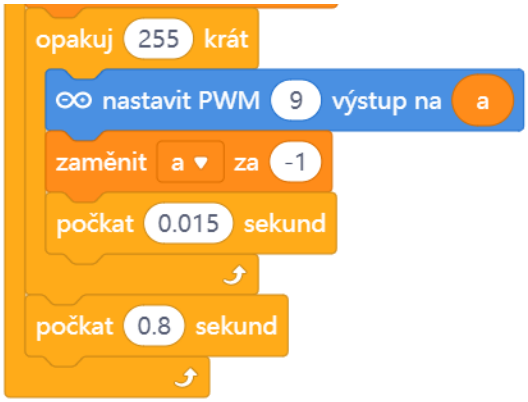
**Blokové schéma**




**Elektronické schéma**




**Krok 2:** V prostředí mBlock sestavíme následující program.



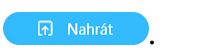

Program lze stáhnout z:
http://mBlock.fyzika.net/zdrojove-kody/lekce-12/12-Rizeni-LED-pomoci-PWM.mblock.

### Vysvětlení kódu

Hlavní nekonečná smyčka „opakuj stále“ obsahuje dva cykly s konečným počtem opakování. V prvním cyklu chceme připojenou LED postupně rozsvítit. Aby její svit postupně narůstal, využijeme výstupu PWM, na kterém budeme postupně zvyšovat výstupní hodnotu z počáteční 0 na maximálních 255. A to právě dělá první cyklus. Výstupní hodnota pro PWM výstup je uložena v proměnné `a`, kterou zapíšeme na výstup a pak zvýšíme o jedničku. Aby se tato změna mohla projevit na svitu připojené LED, je v cyklu zařazen ještě blok čekání (8 ms). Druhý cyklus funguje přesně obráceně. Hodnota proměnné `a` začíná na hodnotě 255 a následující cyklus s 255 opakováními ji postupně snižuje na nulu. I v tomto cyklu je hodnota proměnná `a` zapisována na PWM výstup a je zde i stejná čekací doba. Právě stejná hodnota čekání v obou cyklech zajistí stejnou dynamiku postupného rozsvěcení a zhasínání LED. Klidně si můžeme zkusit tyto čekací doby změnit (zvýšit, snížit, nastavit každou jinou…).

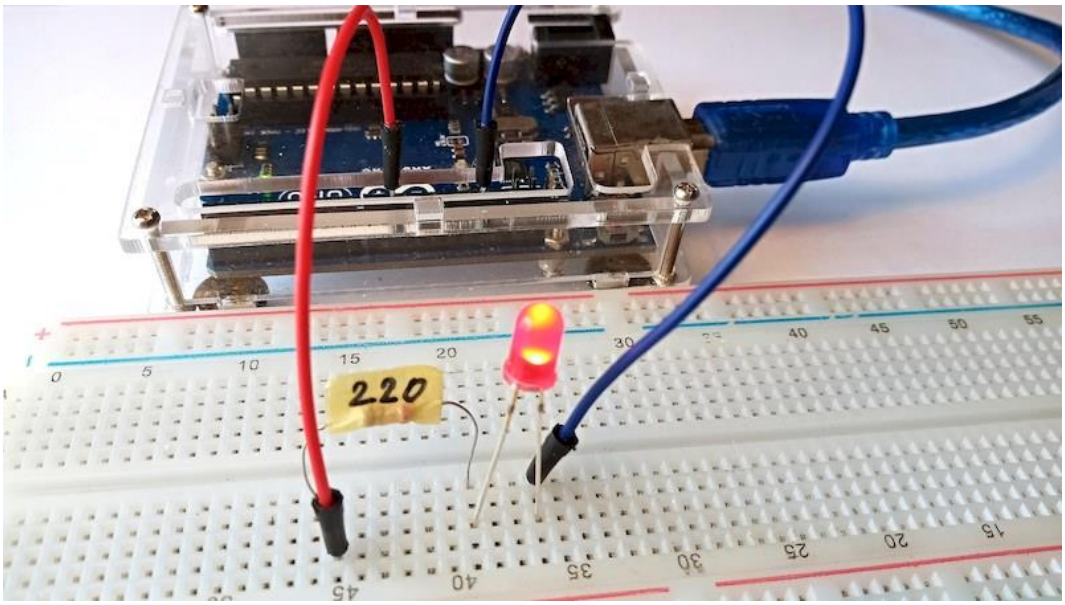
**POZNÁMKA:**

Při pohledu na kód by si někdo mohl říci, že jsou tam zbytečné bloky, které nastavují hodnotu proměnné `a` na počáteční hodnoty před jednotlivými cykly. Pokud bychom nastavili hodnotu proměnné `a` na 0 ještě před nekonečnou hlavní smyčkou, pak by se hodnota na konci prvního cyklu měla rovna hodnotě 255. Stejně tak po druhém cyklu by se měla hodnota snížit na počáteční 0. A pak se vše bude postupně opakovat. To je pravda, na druhou stranu takto může každá smyčka pracovat se svým prvotním nastavením zcela samostatně a nemusí „spoléhat“ na to, jaká hodnota na ní „zbyde“ z předešlého cyklu. Z hlediska přehlednosti a spolehlivosti kódu je někdy jistější si potřebné hodnoty znova (*i když někdy asi zbytečně*) nastavit.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

.

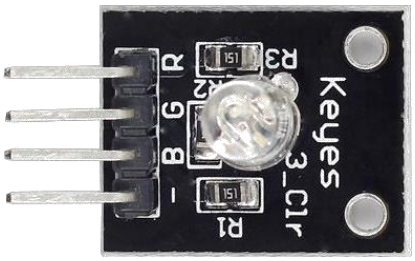
**Krok 4:** Po spuštění kódu bychom měli vidět, jak se LED postupně rozjasňuje a postupně zhasíná. To vše stále dokola. Zajímavé je experimentovat v programu s délkami čekacích intervalů ve smyčkách pro rozsvícení/zhasínání LED a sledovat vliv na výsledný světelný efekt.



# Lekce 13 – RGB LED

## Úvod

RGB LED jsou elektronické součástky, které mohou vyzařovat různě barevné světlo. V principu jde o tři různobarevné LED (červená, zelená a modrá), které jsou umístěny do společného průhledného pouzdra se čtyřmi napájecími kolíky. Ze třech základních barev (červená, zelená a modrá) lze pomocí jejich různých jasů namíchat všechny možné barvy. V námi použité experimentální sadě je k dispozici RGB modul, který kromě RGB LED obsahuje i trojici ochranných rezistorů, takže jej lze připojit přímo na PWM výstupy modulu Arduino. RGB modul se připojuje pomocí trojice pinů: R, G, B, kde vývody R/G/B jsou anody jednotlivých barevných LED, a čtvrtého vývodu, který je společnou katodou.




## Použité komponenty

- Arduino
- USB kabel
- RGB LED modul
- vodiče typu „samec-samice“

## Princip

V tomto experimentu budeme opět využívat technologii PWM, tentokráte pro ovládání jasu jednotlivých barevných LED obsažených v RGB LED. Ovládání jasu jedné LED jsme zvládli v předešlé lekci, takže nyní to využijeme postupně třikrát. Každou ze tří barevných LED budeme řídit jedním PWM kanálem, který budeme nastavovat na 255 stupňů jasu. Pokud všechny tři kanály základních barev jsou všechny nastaveny na 0, LED zhasne. Pokud jsou všechny nastaveny na maximální hodnotu 255, LED bude zářit nejjasněji a výsledná barva bude bílá. Nastavováním různých hodnot barevných komponent získáme různé barvy a odstíny svitu RGB LED. Například trojice čísel R:128, G:255 a B:16 vygeneruje světle zelenou barvu.

Podobně jako u LED segmentových zobrazovačů (*blíže je poznáme v pozdějších lekcích*) můžeme RGB LED dělit na dva základní typy:

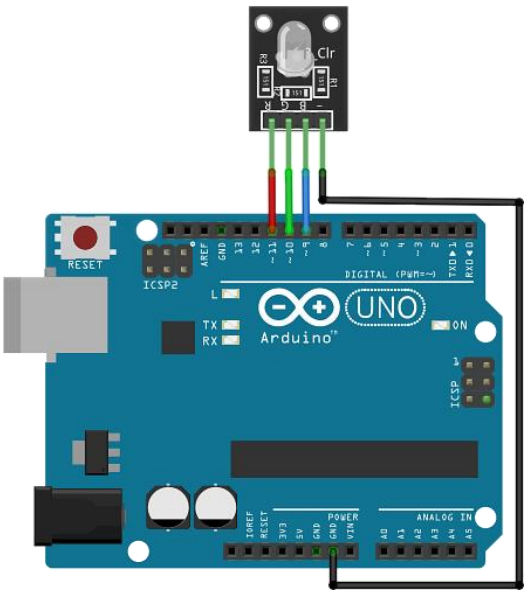
1. typ se společnou anodou,
2. typ se společnou katodou.

Vždy je potřeba vědět, s jakým typem pracujeme, neboť se oba typy liší základním elektronické zapojení. V tomto experimentu budeme používat RGB LED se společnou katodou.

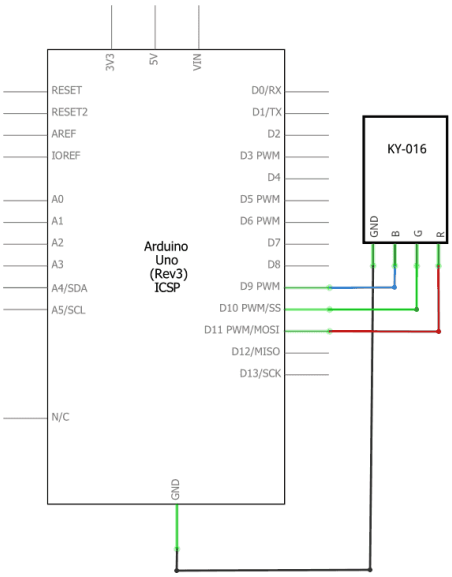
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Propojení modulu RGB LED a modulu Arduino zobrazuje následující tabulka.

**Blokové schéma**




**Elektronické schéma**




| RGB LED modul | Modul Arduino UNO |
|---|---|
| R | 11 |
| G | 10 |
| B | 9 |
| – | GND |

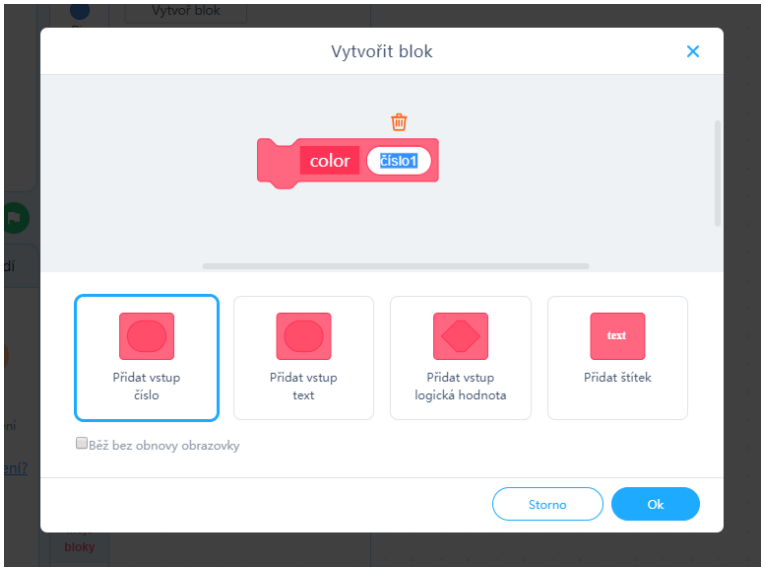
## Podprogram se vstupními parametry

Pro nastavení jednotlivých PWM výstupů barevných komponent RGB LED využijeme podprogram. Použití podprogramu jsme již několikrát zvládli. Novinkou v této lekci bude použití podprogramu se vstupními parametry. Budeme tedy chtít vytvořit programový blok podprogramu, do kterého bude možné vkládat vstupní hodnoty, se kterými tento blok bude pracovat. V našem případě budou vstupní parametry čísla pro rozsvícení jednotlivých barevných složek.

Vytvoření tohoto bloku podprogramu bude v prostředí mBlock stejné jako ve všech předešlých případech, tedy pomocí ikony „Moje bloky“ a kliknutím na volbu Vytvoř blok. Po pojmenování podprogramu musíme přidat vstupní parametry pomocí ikon v dolní části okna průvodce. Kliknutím na jednotlivé nabídky Přidat vstup číslo, Přidat vstup text atd. budou do programového bloku podprogramu přidány

zvolené vstupní parametry. Tyto parametry můžeme pojmenovat, čímž vlastně založíme stejnojmenné lokální proměnné, které pak můžeme v rámci podprogramu používat.

Parametry v deklaraci podprogramu můžeme nejen přidávat, ale i odebírat. V okamžiku, kdy klikneme na název parametru, jako bychom ho chtěli přejmenovat, se nad parametrem objeví ikona koše. Právě pomocí této ikony můžeme zvolený parametr z deklarace odstranit.

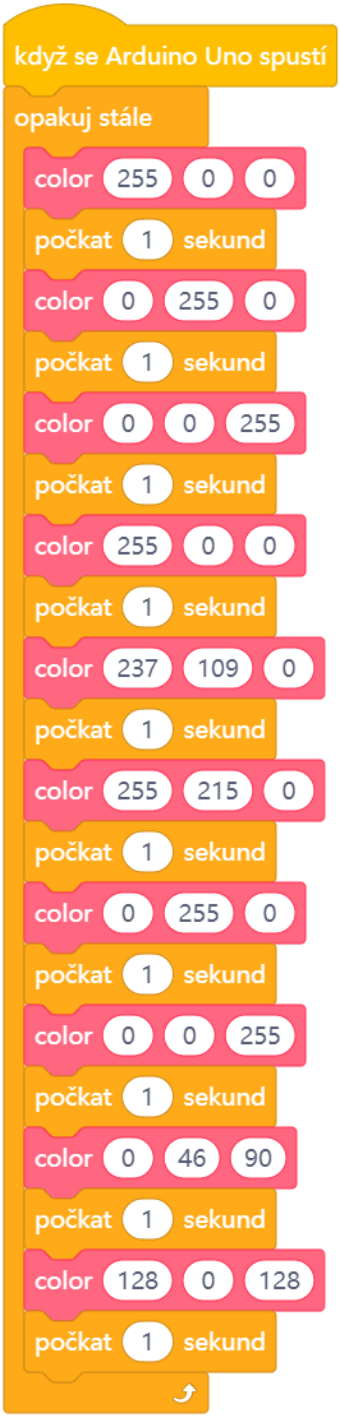



Postupně vytvoříme podprogram nazvaný `color`, který bude mít tři vstupní číselné parametry (`red`, `green` a `blue`).

Po vytvoření zvoleného bloku podprogramu včetně potřebných vstupních parametrů, potvrdíme deklaraci podprogramu pomocí tlačítka Ok

**Krok 2:** V prostředí mBlock je třeba vytvořit následující program. Nejdříve založíme podprogram nazvaný `color` se vstupními číselnými parametry `red`, `green`, `blue` (*odpovídajících barevným složkám*). Po definici hlavičky podprogramu vytvoříme kód podprogramu, který bude ovládat RGB LED, tedy vysílat PWM signál na piny `9`, `10` a `11`.

Jakmile máme definovanou „pracovní náplň“ vlastního programového bloku `color`, můžeme tento programový blok použít pro vytvoření následujícího hlavního kódu programu. Opět připomeneme, že jakmile budeme potřebovat do programu vložit růžový programový blok `color`, najdeme jej v příkazové kategorii „Moje bloky“ a pracujeme s ním stejně jako s jakýmkoliv jiným blokem.




Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-13/13-RGB-LED.mblock.

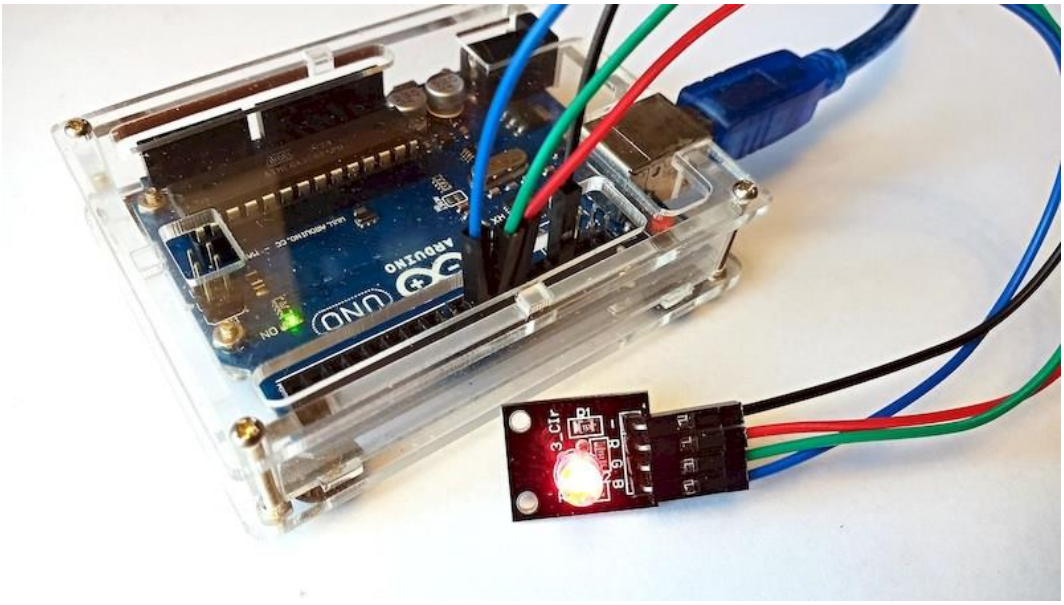
### Vysvětlení kódu

Pro pochopení celého programu je klíčové pochopení podprogramu `color`. Naštěstí jeho část není nikterak složitá. Do podprogramu vstupují tři hodnoty určující intenzity jednotlivých barev – vstupní číselné proměnné `red`, `green` a `blue`. Tyto vstupní hodnoty jsou použity jako výstupní hodnoty pro PWM výstupy jednotlivých barevných složek připojené RGB LED. Podobně jako v předešlé lekci výstupní hodnoty PWM určují míru rozsvícení dané LED. Tím dosahujeme různých světelných intenzit červené, zelené a modré složky výsledného emitovaného světla.

Pokud chápeme, jak funguje podprogram `color` se zadanými vstupními parametry `red`, `green`, `blue`, je tělo hlavního programu již jen opakovaným voláním této procedury s různými vstupními parametry, které určují RGB kód zadané barvy. Díky tomu je nejdříve nastavena červená barva (R:255, G:0, B:0), po vteřinové pauze následuje rozsvícení RGB LED v zelené barvě (R:0, G:255, B:0) a pak v modré (R:0, G:0, B:255). Po dalších sekundových prodlevách následují základní barvy duhy: červená (R:255, G:0, B:0), oranžová (R:237, G:109, B:0), žlutá (R:255, G:215, B:0), zelená (R:0, G:255, B:0), modrá (R:0, G:0, B:255), indigo (R:255, G:46, B:90), purpurová (R:128, G:0, B:128). Vše se neustále opakuje, neboť příkazy pro nastavení jednotlivých barev jsou uzavřeny v základní nekonečné smyčce „opakuj stále“.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Nyní můžeme vidět, jak RGB LED postupně mění barvy a svítí červeně, zeleně, modře, oranžově, žlutě, fialově, indigo atd. Jednotlivé barvy se střídají ve vteřinových intervalech.



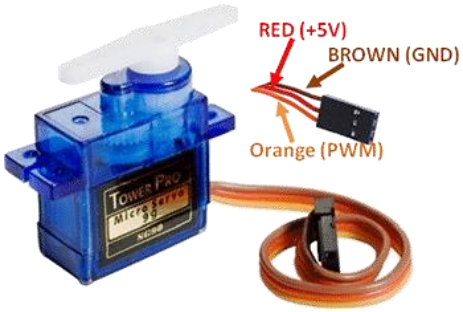
# Lekce 14 – Servo

## Úvod

Servomotor (zkráceně servo) je motor určený pro pohony, u kterých lze nastavit přesnou polohu natočení osy. V tom se podstatně liší od standardního motoru, jehož účelem je zpravidla se jen otáčet a pohánět nějaké zařízení nebo vozidlo. Pomocí serva se ovládají mechanismy, u kterých potřebujeme přesně vědět, jak jsou nastaveny nebo natočeny, například jsou tak ovládány posuvy CNC strojů, nastavení čtecí hlavičky u pevného disku a podobně. Serva ale můžeme také potkat u RC modelů, kde se používají modelářská serva pro natáčení řídicí nápravy, řídicích klapek u letadel nebo kormidla u lodí.

Servomotor je zpravidla řízen sérií pulzů na ovládacím pinu. Tyto impulsy „řeknou" servu, do jaké pozice se má otočit. Servo též obsahuje prvek zpětné vazby, který řídicí jednotce dává zpětnou informaci o tom, jak je osa serva skutečně natočena. Téměř všechna serva používaná pro hobby projekty pracují s řídicími pulzy o základní frekvenci 50 Hz (doba mezi dvěma pulzy 20 ms). Poloha hřídele je určena šířkou řídicího signálu, tzv. PWM signálu (Pulse Width Modulation), jehož šířka se tak pohybuje v rozmezí 1 ms pro krajní levou polohu do 2 ms pro krajní pravou polohu (1,5 ms odpovídá střední poloze). Servo se obvykle může otáčet od 0 do 180 stupňů (některá i víc).

V našem experimentu budeme servo řídit modelářské servo 9g pomocí modulu Ardiuno. Námi použité servo má tři vodiče. Hnědý vodič je zem (GND), červený vodič je napájecí napětí ($V_{CC}$) a oranžový vodič je určen pro řídicí PWM signál.




## Použité komponenty

- modul Arduino
- USB kabel
- servo 9g
- vodiče pro nepájivé pole *(typ „samec-samec")*

## Princip

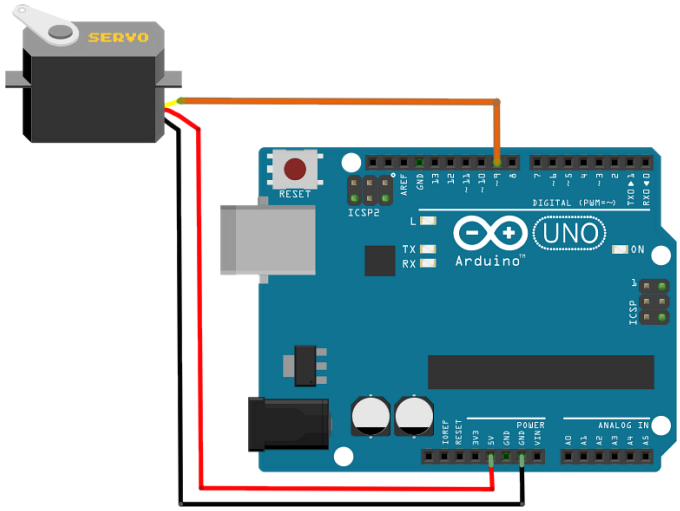
Jak již bylo řečeno, servo se skládá z motoru, převodovky a zpětnovazební detekce polohy. Naším úkolem bude modulem Arduino vysílat PWM signál, který pak bude v řídicí elektronice serva zpracováván pro výpočet směru

otáčení a k řízení motoru. Přesné nastavené řídicího PWM signálu, včetně nastavení jeho střídy, je již poměrně složité, proto využijeme již hotové knihovny pro řízení serva, která je součástí rozšíření „MAXI Starter kit“. Servo tedy bude sice řízeno PWM signálem, ale z pohledu programování, již to vlastně ani nemusíme poznat, neboť princip ovládání nám zůstane skryt za funkcí ovládací knihovny.

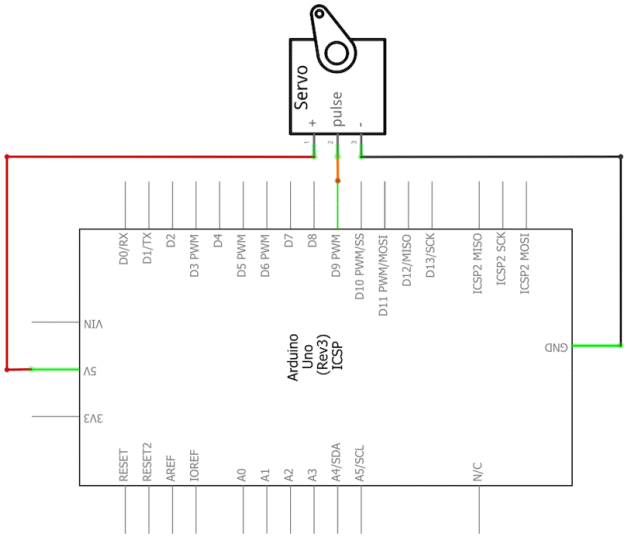
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Signálový vodič serva připojíme do pinu 9 modulu Arduino (viz tabulka).

**Blokové schéma**




**Elektronické schéma**




| Servo *(vodič)* | Modul Arduino |
|---|---|
| oranžový | 9 |
| červený | 5V |
| černý | GND |

**Krok 2:** V prostředí mBlock je třeba vytvořit následující program.

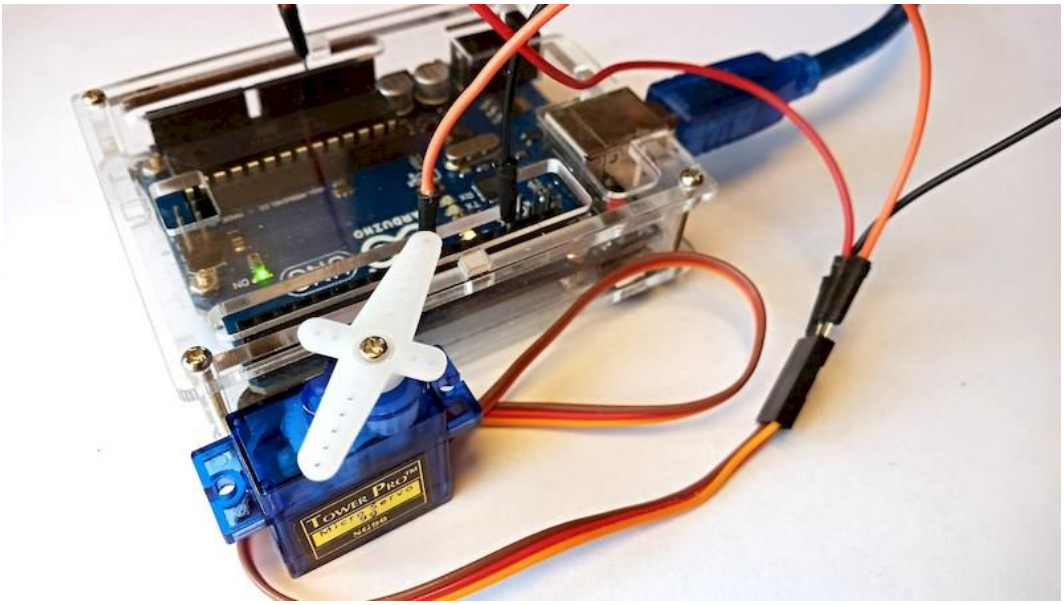

Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-14/14-Servo.mblock.

### Vysvětlení kódu

Po úvodní inicializaci serva, kde programu určíme, který z digitálních pinů podporujících výstup PWM bude řídicím signálem, následuje hlavní nekonečná smyčka „opakuj stále“. V této smyčce jsou dva cykly s pevným počtem opakování. První cyklus postupně natáčí servo v kladném směru, druhý cyklus vrací servo do výchozí polohy. Programový blok pro ovládání natočení serva má jako svůj vstupní parametr úhel, do kterého se má servo natočit. Z tohoto důvodu se v prvním cyklu postupně zvyšuje hodnota proměnné `pos`, která určuje úhel natočení. Z výchozí hodnoty 0° se postupně zvyšuje na hodnotu 160°. Po vychýlení serva do této polohy, následuje druhý cyklus, kdy je hodnota proměnné `pos` postupně zmenšována zpět na hodnotu 0. Snižování hodnoty `pos` je prováděno blokem „`změnit pos za -1`“, což může být z důvodu chybného překladu tohoto bloku (*zmíněno dříve*) trochu matoucí.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Po spuštění kódu uvidíme, jak se servomotor postupně natáčí na jednu stranu, poté se v opačném směru otočí zpět. A to vše se bude opakovat stále dokola.


TOWER PRO

# Práce se sériovým monitorem

# Lekce 15 – Sériový monitor

## Úvod

V této lekci se naučíme, jak zapnout a vypnout LED pomocí připojeného počítače a sériového monitoru. Sériový monitor se používá ke komunikaci mezi modulem Arduino a počítačem nebo jinými přístroji. Pomocí sériového portu je možné posílat data na modul Arduino, nebo naopak data načítat a tak modul Arduino ovládat – třeba pomocí klávesnice počítače. Právě možnost řízení desky Arduino pomocí počítače si ukážeme v této lekci. V dalších lekcích budeme naopak zasílat data z modulu Arduino na připojený počítač. V ovládání modulu Arduino pomocí sériového monitoru budeme ilustrovat na příkladu tří různobarevných LED (červená, zelená a žlutá), které budeme moci příkazem v sériovém monitoru rozsvěcet.

**POZNÁMKA:**
Jelikož prostředí mBlock neobsahuje sériový monitor, budeme muset použít externí aplikaci. Pokud ve svém počítači ještě aplikaci sériového monitoru nemáme, musíme ji doinstalovat. O instalaci a používání sériového monitoru (Serial Monitor) jsme si říkali v úvodní kapitole Sériový monitor.

## Použité komponenty

- modul Arduino
- USB kabel
- 3× LED (červená, zelená, žlutá)
- 3× rezistor (220 Ω)
- nepájivé pole
- vodiče pro nepájivé pole

## Princip

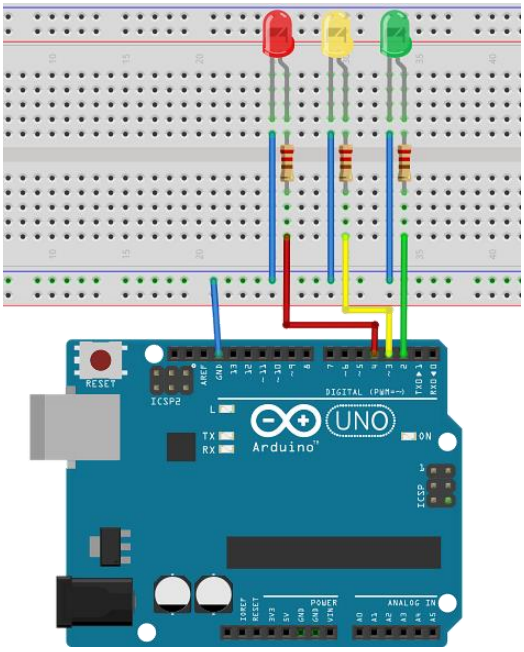
Ať již vybereme jakoukoliv aplikaci pro sériový monitor, je třeba si uvědomit, že sériový monitor slouží jako „přestupní stanice“ pro komunikaci mezi počítačem a modulem Arduino. Komunikace probíhá po sériové lince (piny 0 a 1), ale my budeme využívat případ komunikace prostřednictvím USB portu modulu Arduino. Proto je třeba si uvědomit, že pokud chceme používat sériový monitor, musí být modul Arduino s počítačem propojen USB kabelem. Zároveň je třeba si uvědomit, že pro práci sériového monitoru je třeba modul Arduino odpojit od prostředí mBlock, neboť na sériové lince může komunikovat jen jedna aplikace.

Pro odesílání dat do modulu Arduino je v sériovém monitoru určeno pole pro vepsání příkazu. Následuje tlačítko Send pro odeslání. Data odeslané od modulu Arduino (odpověď) se zobrazí v hlavním okně sériového portu. Asi nejlépe princip této komunikace uvidíme až při zprovoznění ukázkového programu.

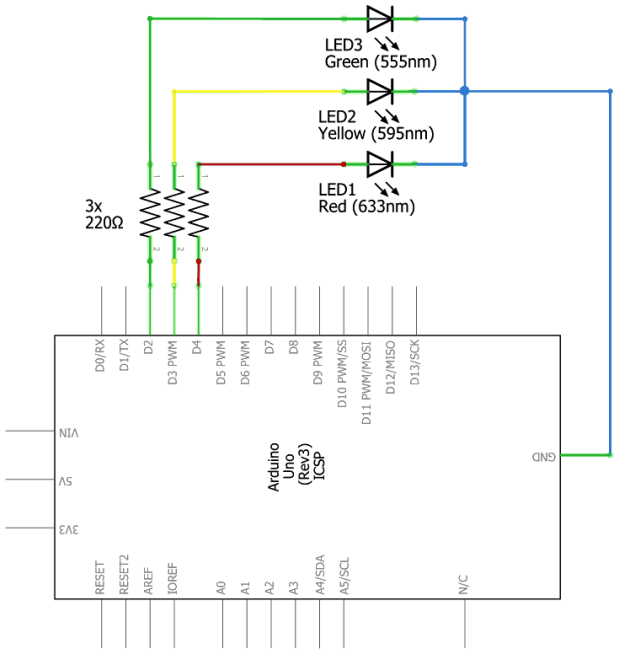
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu.

**Blokové schéma**



**Elektronické schéma**




**Krok 2:** V prostředí mBlock vytvoříme následující program:


Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-15/15-Seriovy-monitor.mblock.

## Vysvětlení kódu

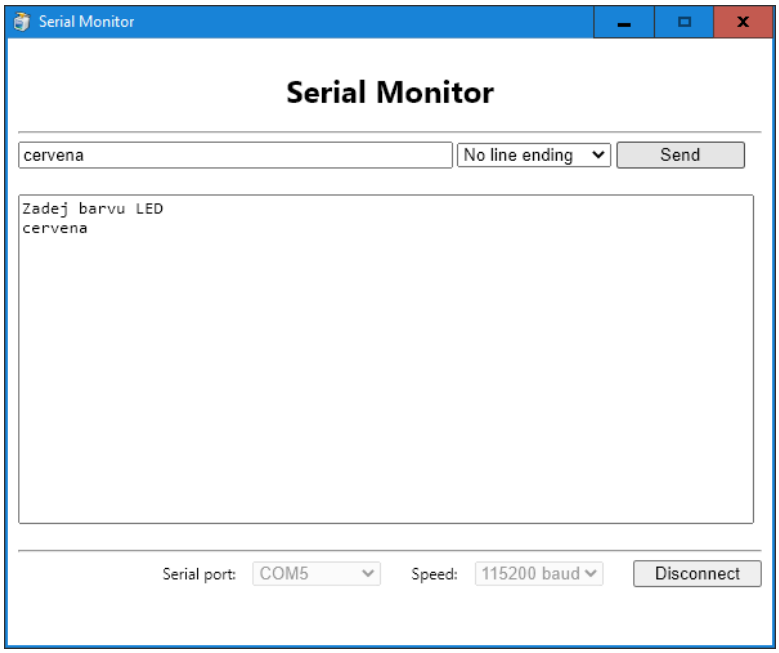
Celý program lze rozdělit na dva funkční celky. Úkolem první části je načtení sériového portu. Program je pozastaven na bloku „čekej, dokud není“, kde se čeká na okamžik, kdy se objeví na sériovém portu nějaká vstupní hodnota – kupříkladu zadaná barva odeslaná z počítače. Následuje podmíněná smyčka, která načítá ze sériového portu data, dokud tam nějaké vstupní hodnoty jsou. Načtené hodnoty jsou číselného formátu, proto jsou převedeny na znaky ASCII. Po vytvoření přijatého textového řetězce následuje druhá část programu.

Z textové proměnné obsahující přijatý textový řetězec je nedříve odstraněn znak odřádkování. To se udělá nahrazením znaku, jehož ASCII kód je 10, za prázdný znak. Pak je přijatý řetězec vypsán na sériový port, aby se objevil v okně sériového monitoru. Následuje kaskáda tří podmínek, které testují, zda se přijatý text shoduje s některou z požadovaných variant (`cervena`, `zluta` nebo `zelena`). Dle toho jsou rozsvíceny a zhasnuty jednotlivé LED. Pokud přijatý text neodpovídá žádné z požadovaných barev, jsou všechny LED zhasnuty.

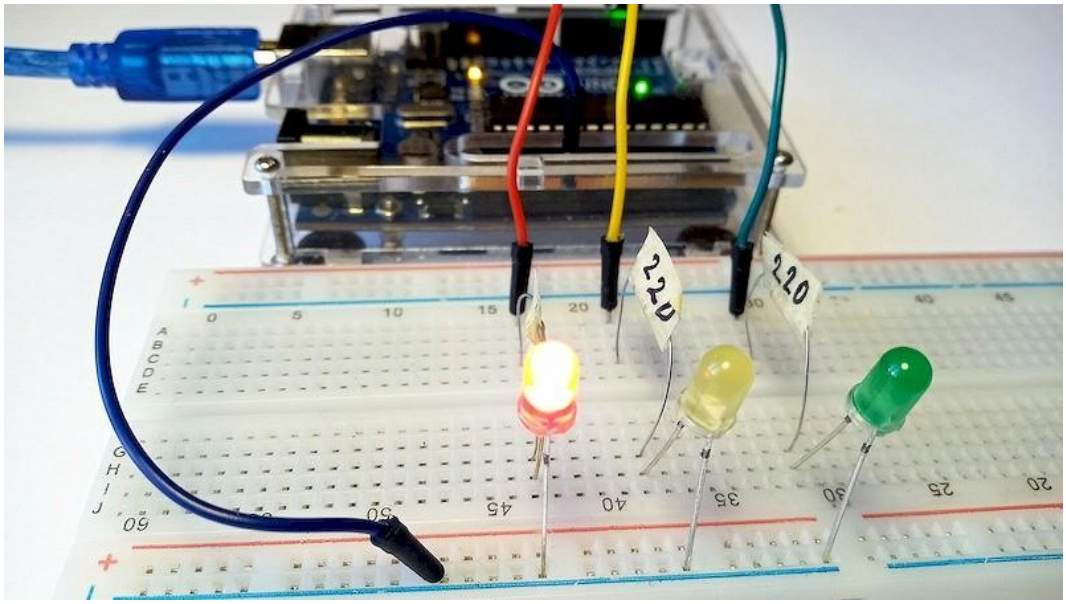
**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Nyní modul Arduino odpojíme od prostředí mBlock pomocí tlačítkem Odpojit.
**USB kabel modulu Arduino ale od počítače neodpojujeme!**

**Krok 5:** Spustíme na počítači aplikaci Serial Monitor. V Serial Monitoru vybereme port, na které je náš modul Arduino (např. COM5) připojen a nastavíme komunikační rychlost na hodnotu 115200 baudů. Na závěr se tlačítkem Connect připojíme k modulu Arduino. Objeví se následující okno sériového monitoru.



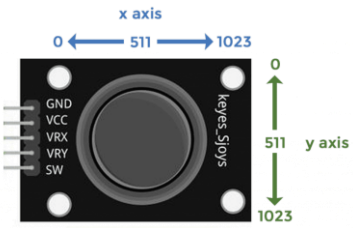

**Krok 6:** Pomocí okénka Serial Monitoru můžeme posílat data z počítače do modulu Arduino. Modul Arduino mám ale též zasílá data – například v okně sériového monitoru vidíme výzvu „Zadej barvu LED“. Do okénka pro odeslání dat můžeme vložit barvu. Když vložíme některou z frází: „cervena“, „zelena“ nebo „zluta“ a klikneme na tlačítko Send, jako odpověď se rozsvítí odpovídající LED. Pokud ale zadáme jinou než požadovanou barvu (např. „ruzova“), všechny LED zhasnou.



# Lekce 16 – Joystick PS2

## Úvod

Kdo by neznal joystick? Joystick je vstupní zařízení, které je jedním z hlavních druhů herních ovladačů. Pracuje na principu páčky, která se naklápí na své základně a tím udává úhel a směr. Kromě analogového signálu určeného náklonem páčky jsou na joysticku často i tlačítka, která svým digitálním signálem doplňují řídicí informace. Joysticky jsou často používány nejen pro ovládání videoher, ale i pro ovládání robotů a jiných manipulátorů. V naší lekci použijeme jednoduchý PS2 joystick.




## Použité komponenty

- modul Arduino
- USB kabel
- joystick PS2
- vodiče typu „samec-samice“

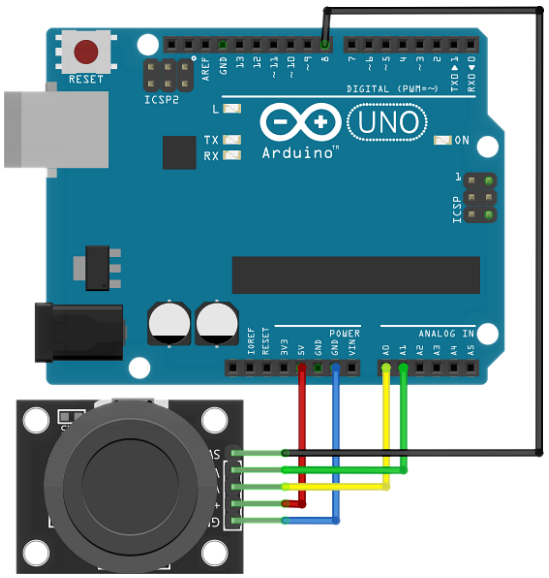
## Princip

Námi použitý modul joysticku je vytvořen dvojicí potenciometrů, které budou generovat dva analogové výstupy (odpovídající náklonu v osách X a Y). Digitální výstup je zastoupen jediným tlačítkem, které stiskneme zatlačením ovládací páčky ve směru osy Z. Náklon páčky budeme načítat jako dva nezávislé analogové signály, naopak stisknutí páčky budeme načítat některým z digitálních vstupů modulu Arduino. Získané údaje budeme vypisovat na sériový port a načítat sériovým monitorem spuštěným na připojeném počítači.

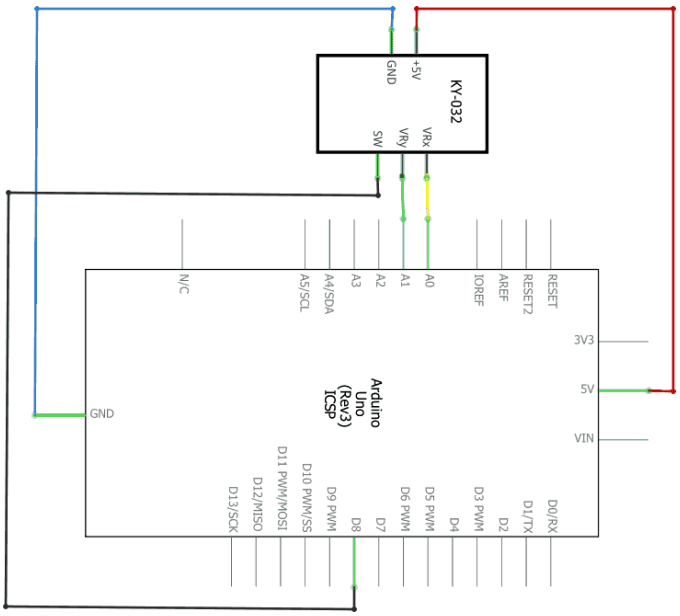
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Jednotlivé vodiče propojení modulu Arduino a modulu joysticku zachycuje tabulka.

**Blokové schéma**



**Elektronické schéma**



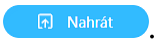

| Joystick PS2 | Modul Arduino |
|---|---|
| VRX | A0 |
| VRY | A1 |
| SW | 8 |
| $V_{CC}$ | 5V |
| GND | GND |

**Krok 2:** V prostředí mBlock vytvoříme následující program:


Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-16/16-Joystick-PS2.mblock.

### Vysvětlení kódu

Princip programu je asi již jasný při prvním pohledu na kód – to je skutečná výhoda blokového programování. V hlavní nekonečné smyčce „opakuj stále“ se neustále načítají stavy analogových pinů `A0` a `A1` a načtená hodnota je rovnou odesílána na výstup sériového portu. Stejně tak je tomu i v případě načítání digitálního pinu 8, který je však nastaven do režimu INPUT_PULLUP (vstup je „držen“ interním pull-up rezistorem na hodnotě 5 V, odpovídající logické úrovni HIGH). Stisknutím tlačítka vstup přizemníme na hodnotu LOW. Před odesláním údajů o směrech náklonu jsou k nim připojeny popisky, aby po vytištění v okně sériového monitoru dávaly smysl.

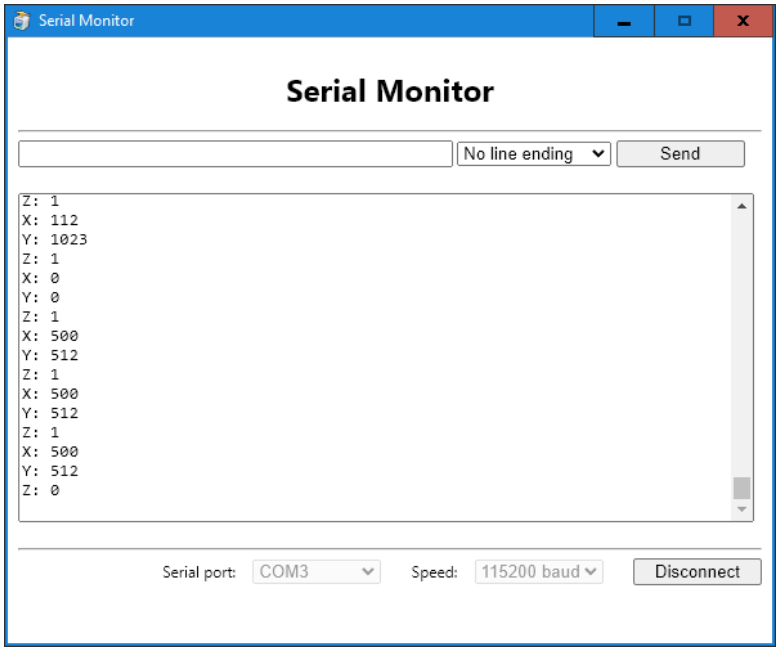
**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

.

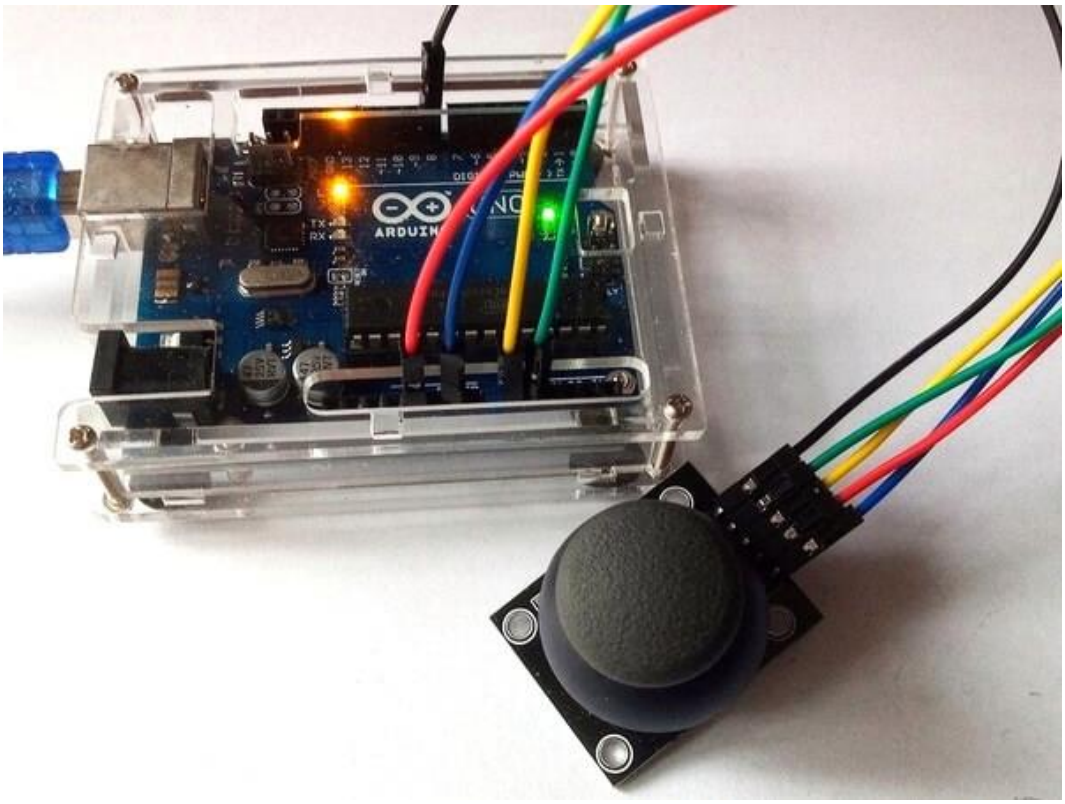
**Krok 4:** Nyní modul Arduino odpojíme od prostředí mBlock pomocí tlačítkem 

.
**USB kabel modulu Arduino ale od počítače neodpojujeme!**

**Krok 5:** Spustíme na počítači aplikaci Serial Monitor. V Serial Monitoru vybereme port, na které je náš modul Arduino (např. COM5) připojen a nastavíme komunikační rychlost na hodnotu 115200 baudů. Na závěr se tlačítkem Connect připojíme k modulu Arduino. Objeví se následující okno sériového monitoru.



**Krok 6:** Nyní budeme hýbat joystickem a v okně sériového monitoru můžeme sledovat, jak se vypisují postupně se měnící údaje o náklonu ve směru os X a Y. Když joystick stiskneme, uvidíme, že se zobrazí v okně sériového monitoru souřadnice Z: 0.



# Lekce 17 – Infračervený přijímač

## Úvod

Hlavním prvkem infračerveného dálkového ovládání, které je v sadě Arduino MAXI Starter kit, je infračervený přijímač HX1838. Tento infračervený přijímač umožňuje detekovat signály z infračervených dálkových ovladačů, jako je kupříkladu dálkový ovladač televize. Při připojení čidla HX1838 k modulu Arduino dokážeme řídicí signály převést na hodnoty proměnných a dále využít. Součástí experimentální sady Arduino MAXI Starter kit je kromě přijímače i dálkový ovladač. Tento univerzální ovladač se naučíme načítat, tím pádem jej můžeme někdy využít pro ovládání své budoucí aplikaci. Bude záležet jen na nás, jaké přiřadíme v ovládacím programu klávesám funkce. Nemusíme používat jen ovladač, který je v experimentální sadě, zkusme využít i jiné dálkové ovladače, které doma najdeme – třeba od televize nebo HiFi věže.


## Použité komponenty

- modul Arduino
- USB kabel
- IR přijímač
- IR dálkový ovladač
- nepájivé pole
- vodiče pro nepájivé pole

## Princip

Jako světelné zdroje se v dálkových ovladačích používají infračervené luminiscenční diody (LED), emitující záření o vlnové délce kolem 950 nm. Jelikož při způsobu zasílání informace způsobem log. 1 = „infra LED svítí“, log. 0 = „nesvítí“ by bylo obtížné zajistit bezchybný přenos datového rámce z důvodu vysokého rušení (zdrojem takového infračerveného záření je i slunce či obyčejná žárovka), využívá se pro přenos modulace.

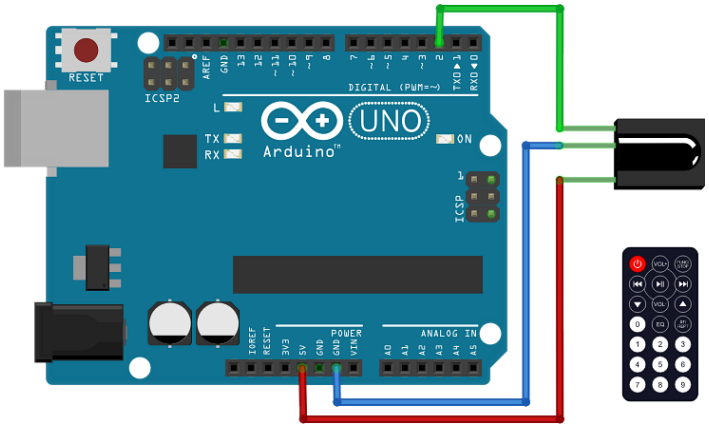
Při modulaci vysoké logické úrovně (HIGH) se infra LED budí obdélníkovým nosným signálem o dané střídě a frekvenci, typicky mezi 30–56 kHz (v našem případě 38 kHz). Tento stav se nazývá značka. Po dobu logické nízké úrovně (LOW) není dioda aktivní, nastává mezera. Dobu trvání značky a mezery, stejně tak jako jejich význam, specifikuje použitý protokol. Přijímač signálu je nastaven na použitý nosný kmitočet. Tím, že ostatní kmitočty odfiltruje, účinně zamezuje možnému rušení okolním světlem.

V našem ukázkovém příkladu budeme chtít vytvořit ovládání vestavěné LED modulu Arduino pomocí tlačítka ▶/⏸ na dálkovém ovladači. Když stiskneme na dálkovém ovladači tlačítko ▶/⏸, infračervené paprsky vyslané z dálkového ovladače budou přijaty IR přijímačem. Pokud odeslaný kód tlačítka bude odpovídat zadané hodnotě (16761405), budeme signalizovat tento stav rozsvícením vestavěné LED na modulu Arduino.

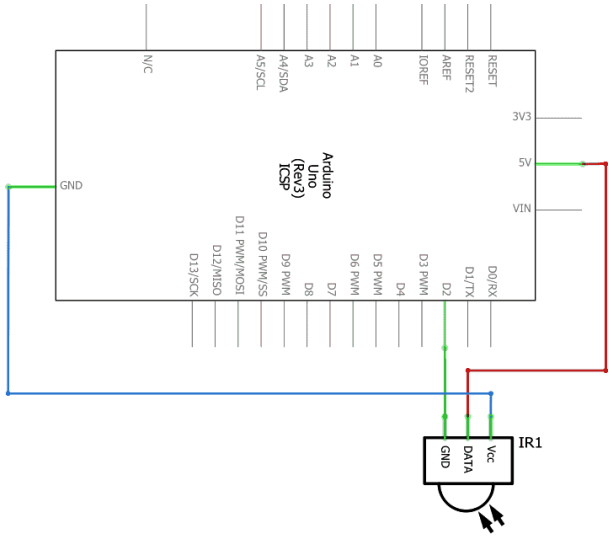
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Zapojení jednotlivých vodičů mezi modulem Arduino a IR přijímače zachycuje tabulka.

**Blokové schéma**

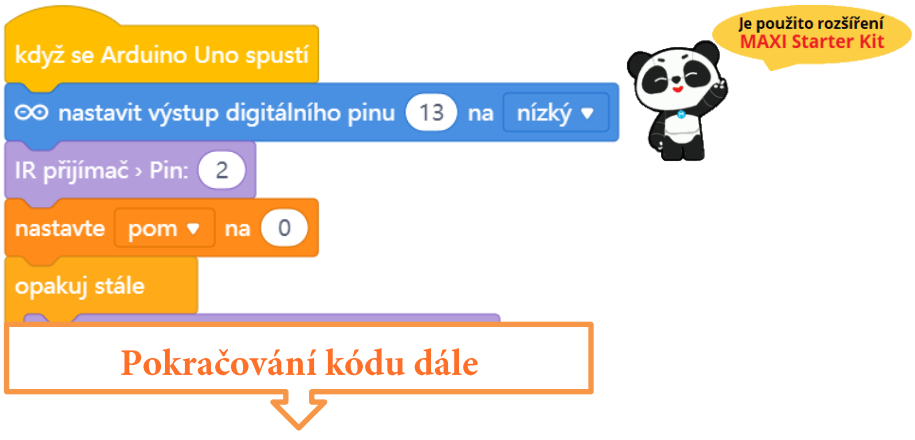



**Elektronické schéma**




| IR přijímač | modul Arduino |
|---|---|
| S (signal out) | 2 |
| GND (–) | GND |
| $U_{CC}$ (+) | 5V |

**Krok 2:** V prostředí mBlock vytvoříme následující program:




Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-17/17-Infracerveny-prijimac.mblock.

### Vysvětlení kódu

První částí programu je obvyklé nastavení pinů modulu Arduino a deklarace použitých proměnných, zde jde o nastavení pinu 13, ke kterému je připojena vestavěna LED modulu Arduino, jako digitální výstup a nastavení čísla pinu, na kterém bude přijímána informace od IR přijímače. Proměnná `pom` bude sloužit pro uložení přijaté hodnoty.

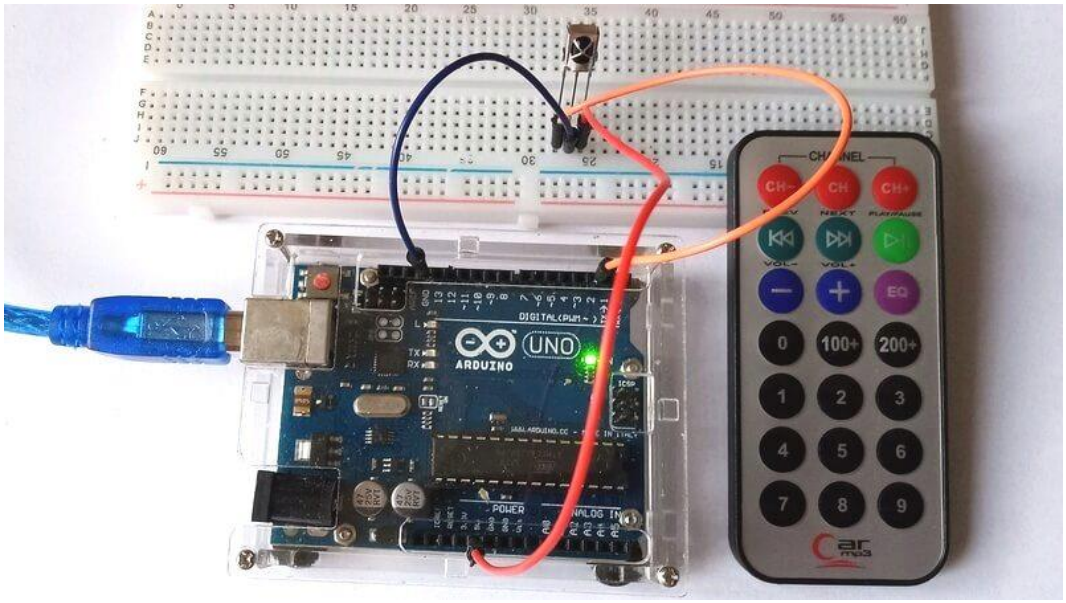
V hlavní nekonečné smyčce „opakuj stále“ programu je načítán IR přijímač. Programové bloky, které se provedou jen při přijetí nějaké zprávy, jsou uzavřeny v sekci „Pokud IR přijal data…“. V takovém případě se uloží načtená hodnota do proměnné `hod` a vypíše se i na sériový port. Vypsání přijaté hodnoty na sériový port se hodí v případě, že bychom chtěli naprogramovat i jiné tlačítko, než zde stanovené tlačítko ▶/⏸. Lze tak v sériovém monitoru vidět kódy všech zachycených tlačítek. Pokud se hodnota přijatého signálu rovná kódu zvoleného tlačítka (zde 16761405) dojde k rozsvícení vestavění LED modulu Arduino (pin 13 je nastaven na vysokou úroveň HIGH), jinak je LED zhasnuta (pin 13 má nízkou úroveň LOW).

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Nyní stiskni tlačítko ▶/⏸ na dálkovém ovladači. Vestavěná LED modulu Arduino, která je připojená k vývodu 13, se rozsvítí. Stiskni libovolnou jinou klávesu na dálkovém ovladači a LED zhasne.

**POZNÁMKA:**
Programový kód obsahuje i zobrazení načteného kódu stisknutého tlačítka dálkového ovladače na sériový port modulu Arduino. Pro samotnou funkci naší aplikace to nyní nemá vliv. Výpisy na sériový

port zde spíše slouží jako určitá kontrolní hlášení. Často se tohoto způsobu využívá při vývoji programu nebo při diagnostice problému. Pokud tedy výstup na sériový port nepotřebujeme, lze daný příkazový blok z programu zcela vynechat.




Pokud chceme výpisy na sériový port sledovat, je opět třeba modul Arduino odpojit od prostředí mBlock a připojit k aplikaci Serial Monitor.

**Krok 5:** Nyní modul Arduino odpojíme od prostředí mBlock pomocí tlačítkem Odpojit.
**USB kabel modulu Arduino ale od počítače neodpojujeme!**

**Krok 6:** Spustíme na počítači aplikaci Serial Monitor. V Serial Monitoru vybereme port, na které je náš modul Arduino (např. COM5) připojen a nastavíme komunikační rychlost na hodnotu 115200 baudů. Na závěr se tlačítkem Connect připojíme k modulu Arduino. Objeví se okno sériového monitoru, ve kterém se budou zobrazovat přijaté kódy stisknutých tlačítek na dálkovém ovladači.

# Připojení periférií

# Lekce 18 – Krokový motor

## Úvod

Občas, jako jsme kupříkladu viděli u serva, se může stát, že nám princip ovládání připojeného zařízení zůstane skryt díky vyžití ovládací knihovny. U připojeného zařízení tak nemusíme řešit, zda je ovládáno digitálními výstupy nebo třeba pomocí PWM signálu. Zkrátka jen načítáme nebo nastavujeme požadované hodnoty pomocí předem připravených programových bloků. Následující kapitola je právě věnována připojování takových zařízení ze sady Arduino MAXI Starter kit, kde již nebudeme řešit konkrétní ovládání pinů, ale využijeme sofistikovanější řízení pomocí knihoven. První takovou periférií modulu Arduino bude krokový motor.

Krokové motory, díky své zvláštní konstrukci, mohou být ovládány s vysokou přesností a bez jakýchkoli mechanismů zpětné vazby. Motor je tvořen několika elektromagnetickými cívkami, které jsou v určitém pořadí zapínány sérií elektrických impulsů a tím dokáží velmi přesně a v definovaných „krocích“ otáčet rotorem s hřídelí ve zvoleném směru. Existují dva typy krokových motorů – unipolární a bipolární. Je velmi důležité, abychom věděli, s jakým typem pracujeme. V následujícím experimentu použijeme krokový motor unipolární.

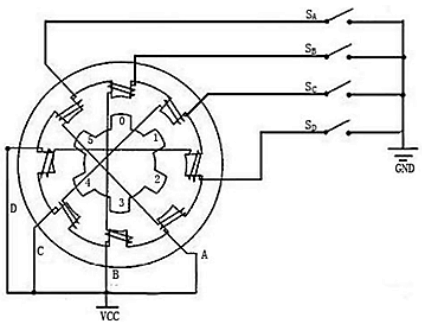
Modul Arduino nebo podobné kontroléry nemohou přímo řídit krokové motory. Na to je požadovaný proud pro ovládací cívky příliš veliký. Je tedy nezbytné připojit řídicí obvod (tzv. řadič nebo též driver), který proud do cívek posílí z externího zdroje. I v tomto experimentu tedy kromě krokového motoru budeme používat i modul řadiče krokového motoru.

## Použité komponenty

- modul Arduino
- USB kabel
- potenciometr 50 kΩ
- krokový motor
- modul řadiče krokového motoru
- nepájivé pole
- vodiče pro nepájivé pole + vodiče typu „samec-samice“

## Princip

Námi použitý krokový motor je čtyřfázový a používá stejnosměrné napájecí napětí. Při postupném napájení jednotlivých cívek motoru (dle příslušného pořadí) se začne motor otáčet krok za krokem. Schematický diagram čtyřfázového krokového motoru je znázorněn na obrázku níže.

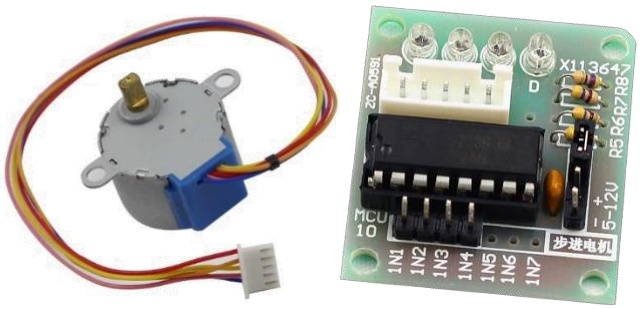


Uprostřed motoru je rotor vytvořený z permanentního magnetu ve tvaru ozubeného kola (např. 6 zubů, označíme je 0–5). Okolo rotoru je více magnetických pólů (zde např. 8). Každé dva protilehlé póly jsou vytvořené spojenými cívkami. Takže tento stator motoru tvoří čtyři páry cívek – označme je od A do D. Tyto cívky se nazývají fáze a mají čtyři vodiče připojené ke spínačům $S_A$, $S_B$, $S_C$ a $S_D$. Na začátku je spínač $S_B$ zapnutý, spínače $S_A$, $S_C$ a $S_D$ jsou vypnuty a magnetické póly ve fázi B jsou srovnány s ozubením 0 a 3 rotoru. Současně zuby 1 a 4 vytváří částečně střídavé uspořádaní zubů cívek C a D fází. Zuby 2 a 5 vytvářejí plně střídavé uspořádání zubů s póly D a A fáze. Když je spínač $S_C$ zapnutý, spínače $S_B$, $S_A$ a $S_D$ jsou vypnuty, rotor se otáčí pod magnetickým polem C vinutí a mezi zuby 1 a 4. Potom se zub 1 a 4 se vyrovná s magnetickými póly vinutí fáze C. Pak zuby 0 a 3 odjíždí mezi póly fází A a B, zuby 2 a 5 se naopak přibližují k magnetickým pólům A až D. Postupným zapínáním jednotlivých fází se tyto situace neustále opakují. Zapínáním fází A, B, C a D se rotor začne jednotlivými kroky otáčet v pořadí A, B, C a D.

Čtyřfázový krokový motor má tři provozní režimy:

- jednoduchý čtyřkrokový (zapnutá vždy jen jedna fáze: A–B–C–D–A–…)
- dvojitý čtyřkrokový (zapnuty vždy dvě následující fáze: AB–BC–CD–DA–AB–…)
- osmikrokový (zapnutá jedna fáze, pak dvě následující a jedna fáze: A–AB–B–BC–C–CD–D–DA–A–…)

Krokový úhel pro oba čtyřkrokové režimy je stejný, ale hnací moment je pro jednoduchý (samostatný) čtyřkrokový režim menší. Krokový úhel pro osmikrokový režim je poloviční, než u režimů čtyřkrokových. Osmistupňový provozní režim tak může udržet vysoký hnací moment a zlepšit přesnost řízení. V tomto experimentu si vyzkoušíme osmikrokový režimu.

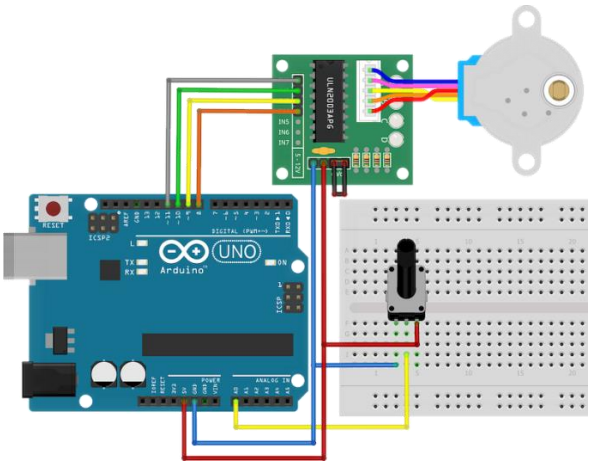



Pro řízení našeho krokového motoru použijeme řadič krokového motoru ULN2003. Jedná se o invertorový obvod, což znamená, že když je vstup na vysoké úrovni (HIGH), odpovídající výstup obvodu ULN2003 je na nízké úrovni (LOW), a naopak. Tedy pokud je na vstupu IN1 vysoká úroveň a nízká úroveň na vstupech IN2, IN3 a IN4, pak je výstup OUT1 na nízké úrovni a všechny ostatní výstupy (OUT2–OUT4) jsou na úrovni vysoké. Konkrétně to bude znamenat, že spínač $S_A$ se zapne a krokový motor se otočí do dané polohy. Pro řízení tedy budeme dávat krokovému motoru určitou časovou posloupnost sepnutí spínačů $S_A$–$S_D$, které nám zastupuje obvod ULN2003.

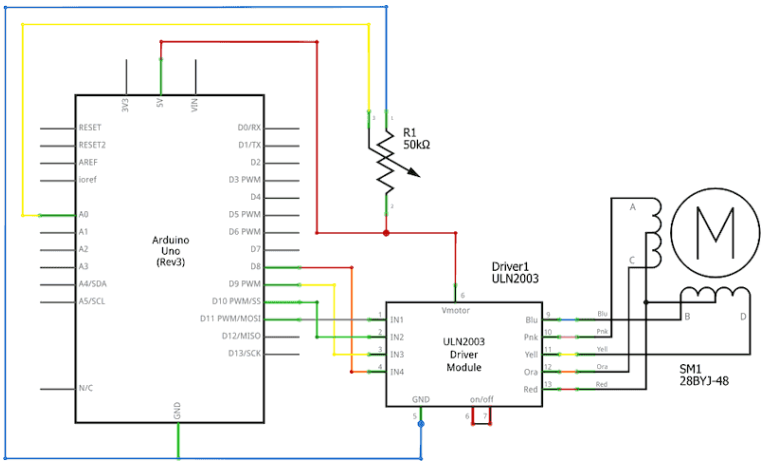
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Propojení modulu Arduino a modulu ovladače krokového motoru uvádí tabulka.

**Blokové schéma**

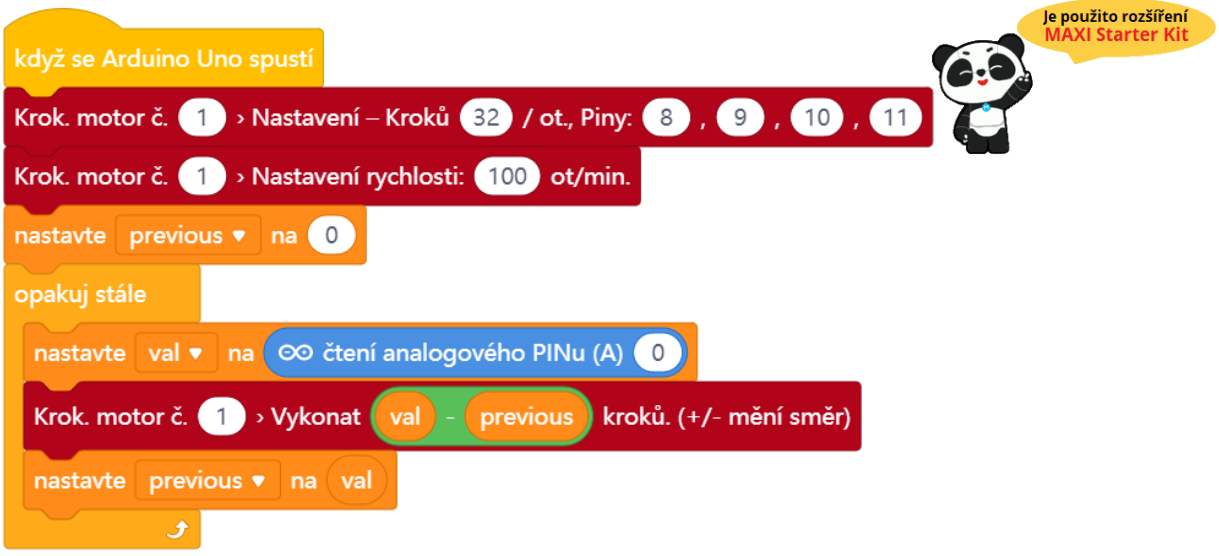


**Elektronické schéma**




| Modul ULN2003 | Modul Arduino |
|---|---|
| IN1 | 8 |
| IN2 | 9 |
| IN3 | 10 |
| IN4 | 11 |
| Ucc | 5 V |
| GND | GND |

**Krok 2:** V prostředí mBlock vytvoříme následující program:




Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-18/18-Krokovy-motor.mblock.

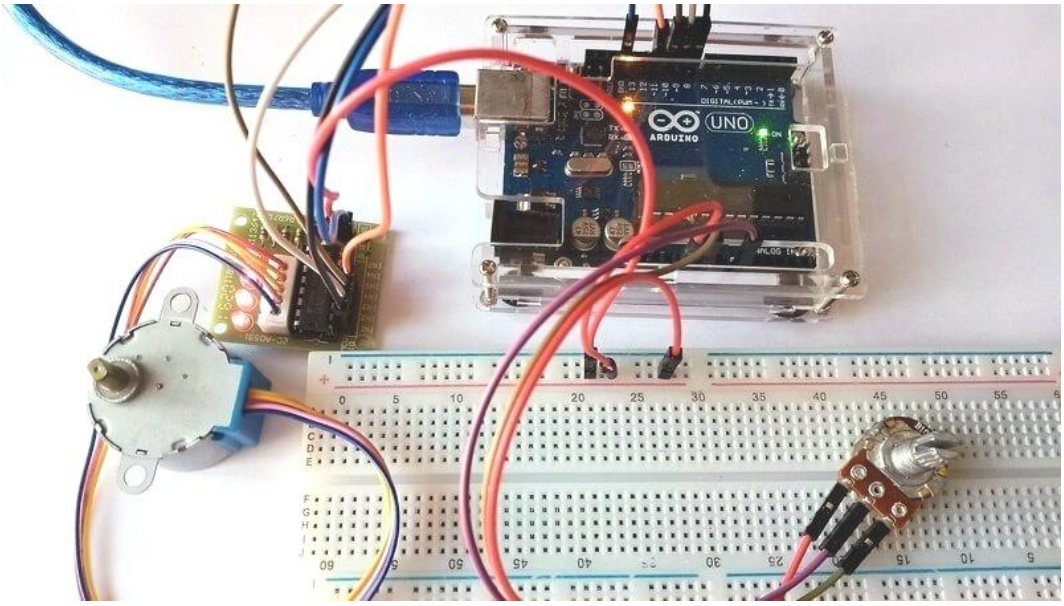
### Vysvětlení kódu

První částí, ještě před hlavní nekonečnou smyčkou „opakuj stále“, je nastavení parametrů použitého krokového motoru. Jsou zde nastaveny výstupní digitální piny jednotlivých cívek krokového motoru a počet kroků na otáčku. Zde je třeba si uvědomit, že údaj o počtu kroků na otáčku se vztahuje na počet kroků na jednu otáčku rotoru motoru a nikoliv osy, která z něj vychází. Některé krokové motory bývají ve své konstrukci doplněny převodovkou, která je jejich velikost kroku redukuje. Takový případ je i u námi použitého krokového motoru 28BYJ-48. I když to na první pohled není poznat, je tento krokový motor osazen redukční převodovkou 1/64. Proto na jednu otáčku motoru (dle katalogového listu) připadá jen 32 kroků, což odpovídá i námi zadanému parametru v programu. Pokud bychom ale počítali kroky na výstupní ose motoru (tedy až za převodovkou), bylo by jich na jednu otáčku 2048. Obdobně musíme uvažovat při zadávání parametru počtu otáček za minutu.

Hlavní část programu načítá analogovou hodnotu z potenciometru a vytváří rozdíl s předchozí hodnotou, která je uchována v proměnné `previous`. Rozdíl těchto hodnot určuje počet kroků, které má motor vykonat. Pootočíme-li potenciometr směrem k vyšším hodnotám je rozdíl kladný a krokový motor se otočí jedním směrem. Čím větší je rozdíl, tedy čím více otočíme potenciometrem, tím více se motor otočí. Pokud otočíme potenciometr na druhou stranu, je rozdíl hodnot záporný, motor má vykonat záporný počet kroků, což odpovídá opačnému směru otáčení. Pokud se potenciometrem neotáčí, je rozdíl nulový a motor se nikam neotáčí.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Po spuštění kódu můžeme otáčet potenciometrem a vidíme, že tím ovládáme natočení krokového motoru. Nastavení polohy potenciometru určuje, kam se bude otáčet hřídel krokového motoru.



# Lekce 19 – Sedmisegmentový LED zobrazovač

## Úvod

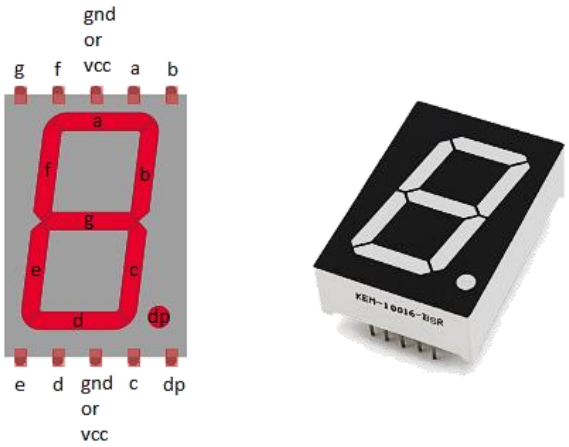
Sedmisegmentový LED zobrazovač představuje soustavu sedmi LED vytvarovaných do tvaru segmentů, ze kterých lze jejich rozsvěcením zobrazovat určité symboly. Kombinací vypnutých a zapnutých segmentů můžeme docílit zobrazení arabských číslic, hexadecimálních číslic, případně i dalších písmen a znaků. Máme-li být přesní, tak je třeba upozornit, že se sice říká „sedmisegmentový" LED zobrazovač, ale ve skutečnosti tento modul obsahuje LED osm. Kromě sedmi segmentů pro zobrazení znaků je zde ještě kulatá LED, která představuje desetinnou tečku.

## Použité komponenty

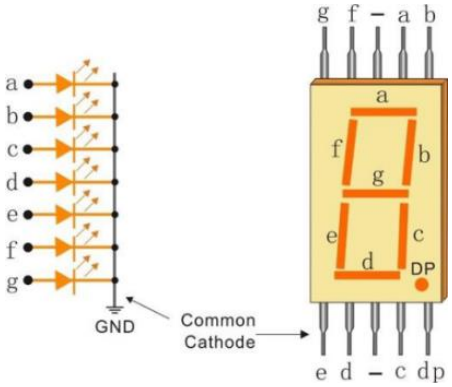
- modul Arduino
- USB kabel
- sedmisegmentový LED zobrazovač
- 8× rezistor (220 Ω)
- nepájivé pole
- vodiče pro nepájivé pole

## Princip

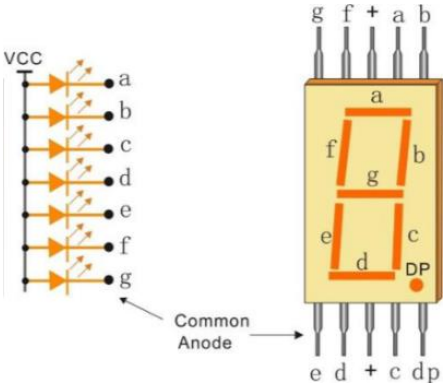
Sedmisegmentový LED zobrazovač je nejpoužívanějším případem segmentového displeje. Je vhodný pouze pro zobrazování číslic, maximálně hexadecimálních číslic A až F. Pro číslicový indikátor je minimální počet segmentů právě sedm. Takový zobrazovací modul se familiárně nazývá „sedmisegmentovka". Na obrázku vidíme rozložení a označení jednotlivých segmentů včetně LED pro signalizaci desetinné čárky.




Diodové sedmisegmentovky mají relativně rychlou odezvu, přibližně 10 nanosekund, a spotřebu proudu od 0,5 až 1 mA na jeden segment u těch nejmenších (tzn. celá sedmisegmentovka 3,5–7 mA). Napětí anody je závislé na barvě, tedy 1,5 až 2,5 V. Aby se ovládání diod zjednodušilo, mají diody navzájem propojené anody či katody. Společná katoda (CC) a společná anoda (CA).



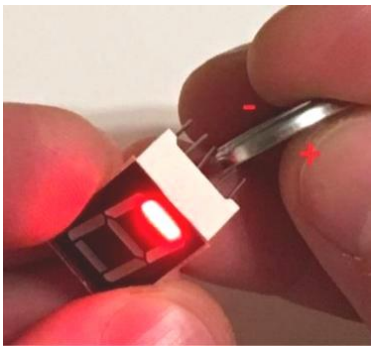

**Sedmisegmentový displej se společnou katodou**
Společná katoda (Common Cathode, CC) – všechny katody LED segmentů jsou spojeny dohromady na nízkou úroveň LOW (GND). Jednotlivé segmenty (a–g) jsou rozsvíceny tak, že anoda segmentu je zapojena na vysokou úroveň HIGH (+5 V) přes omezovací odpory.



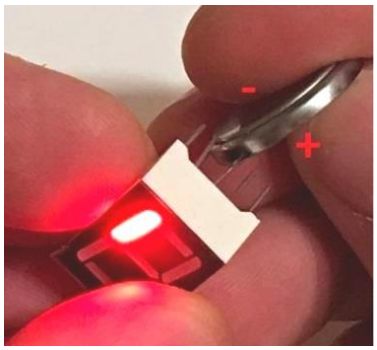

**Sedmisegmentový displej se společnou anodou**
Společná anoda (Common Anode, CA) – všechny anody LED segmentů jsou spojeny dohromady na vysokou úroveň HIGH (+5 V). Jednotlivé segmenty (a–g) jsou rozsvíceny tak, že katoda segmentu je zapojena na nízkou úroveň LOW (GND) přes omezovací odpory.

Rozdíl mezi těmito dvěma displeji, jak jejich název napovídá, spočívá v tom, že společná katoda má všechny katody 7-segmentů spojených dohromady a společná anoda má všechny anody 7-segmentů spojených dohromady.

Především je potřeba zjistit, jaký displej máme ve své experimentální sadě, zda CC nebo CA? Nejednoduší způsob je pomocí 3 V baterie CR2032, která je k dispozici v sadě Arduino MAXI Starter kit. Prostřední pin na displeji, je společný vývod – tedy buď společná katoda, nebo anoda. Zkusíme se rychle dotknout pólem „+“ baterie do společného pinu a „–“ do pinu vedle. Svítí? Pokud ano, máme zobrazovač se společnou anodou. Jestliže nesvítí, zkusíme otočit baterii a dotknout „–“ do společného pinu a „+“ do pinu vedle. Sviti? Máme tedy zobrazovač se společnou katodou.




LED zobrazovač se společnou katodou CC



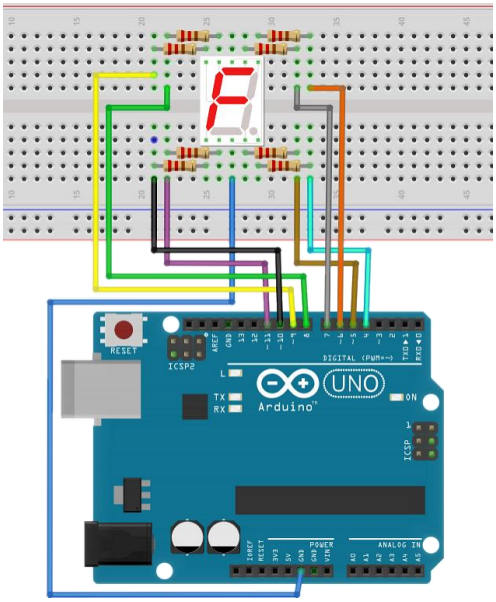

LED zobrazovač se společnou anodou CA

**POZNÁMKA:**
Segmentem nesmíme svítit příliš dlouho, už asi vteřina trvalého svícení může diodu segmentu spálit. Napětí 3 V z baterie je příliš moc pro červenou LED, která má předepsané napětím 1,9–2 V!

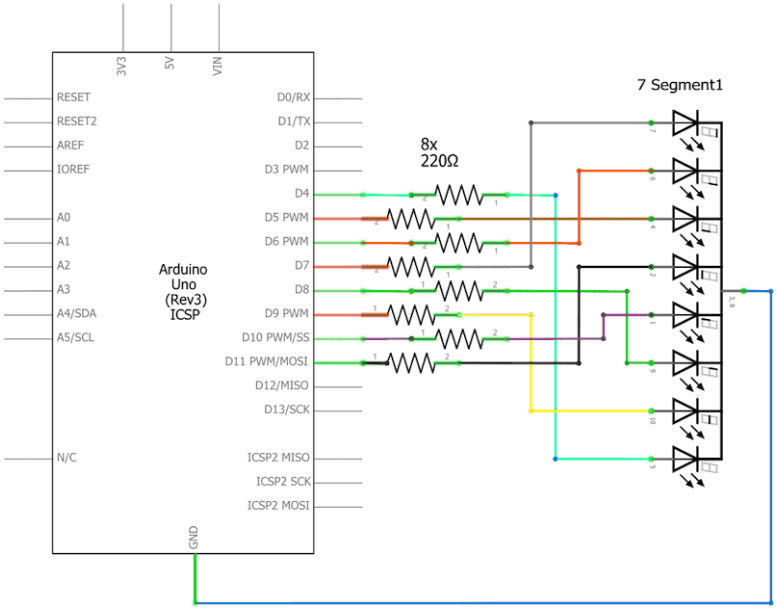
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Následující zapojení je příklad zapojení pro CC zobrazovač. V případě CA zobrazovače je třeba zapojit společný pin na +5 V. Propojení pinů modulu Arduino a pinů LED zobrazovače uvádí následující tabulka (propojení je zrealizováno přes rezistory 220 Ω).

**Blokové schéma**



**Elektronické schéma**

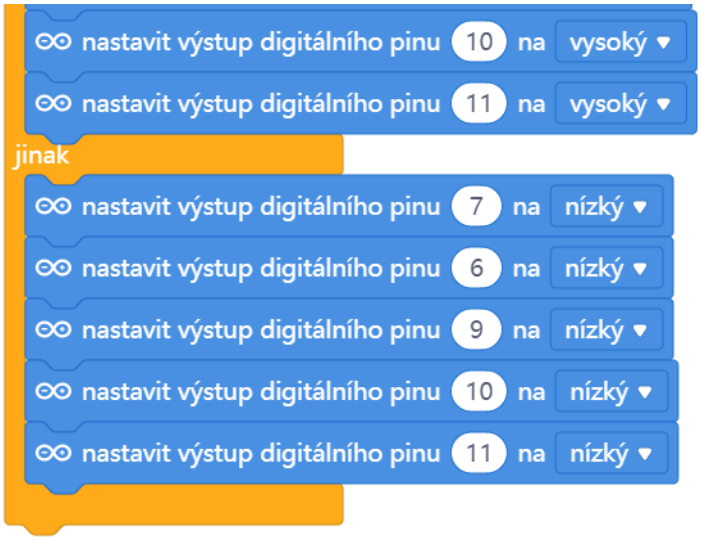



| LED zobrazovač CC | Modul Arduino |
|---|---|
| a | 7 |
| b | 6 |
| c | 5 |
| d | 11 |
| e | 10 |
| f | 8 |
| g | 9 |
| dp | 4 |
| GND | GND |

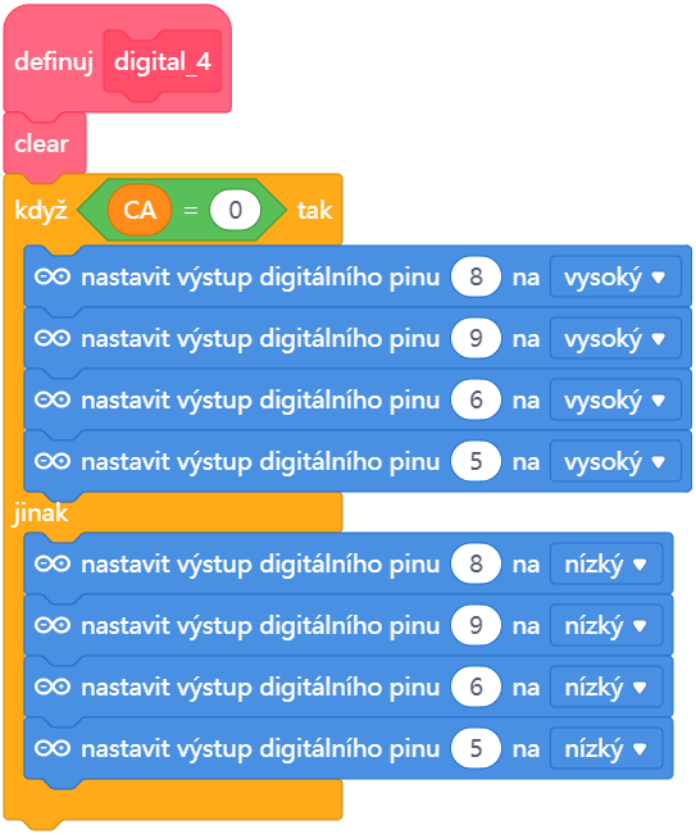
**Krok 2:** V prostředí mBlock vytvoříme následující program. Asi nejlepší postup pro tvorbu tohoto programu s tolika podprogramy je začít „tak trochu pozpátku“. Tedy nejdříve vytvořit podprogram `clear`, který zhasíná celý zobrazovač, pak postupně podprogramy `digital_0`, `digital_1`, …, `digital_9` pro zobrazení jednotlivých číslic. Podprogram `clear` je volán ve všech podprogramech pro jednotlivé číslice, proto je třeba jen definovat dříve, než začneme vytvářet podprogramy pro rozsvícení segmentů dané číslice. Tvorbu hlavní část programu si tedy paradoxně musíme nechat až na finále.

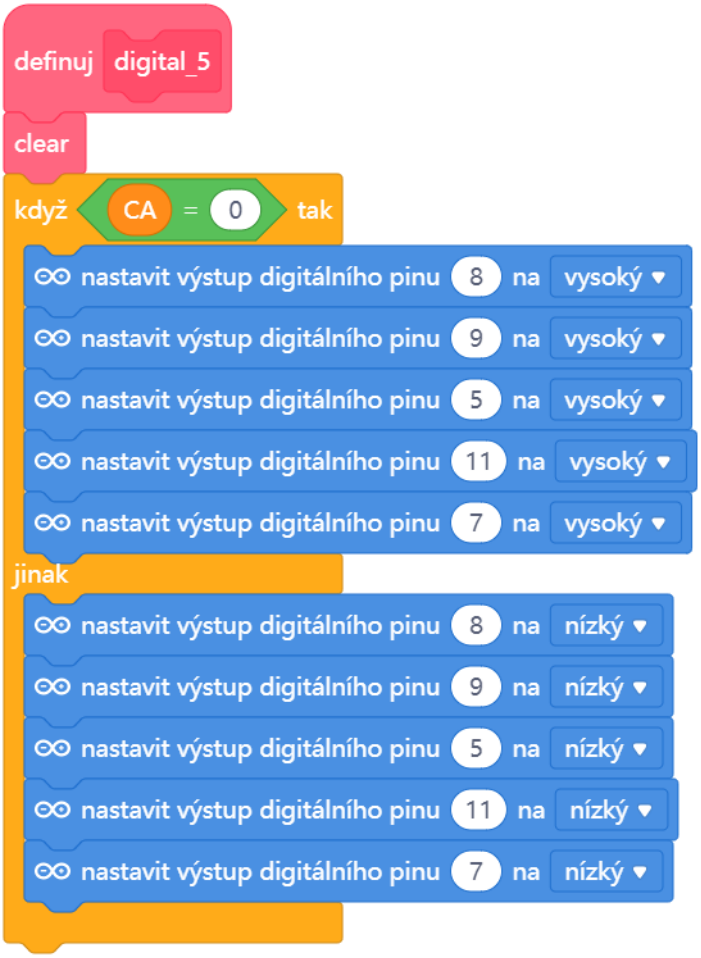

Nyní je třeba definovat jednotlivé procedury `digital_0`, `digital_1`, …, `digital_9` pro zobrazení jednotlivých číslic 0–9. Pracovní plocha určená pro kód v prostředí mBlock se začne poměrně rychle plnit, je třeba umisťovat jednotlivé bloky podprogramů s citem pro přehlednost.

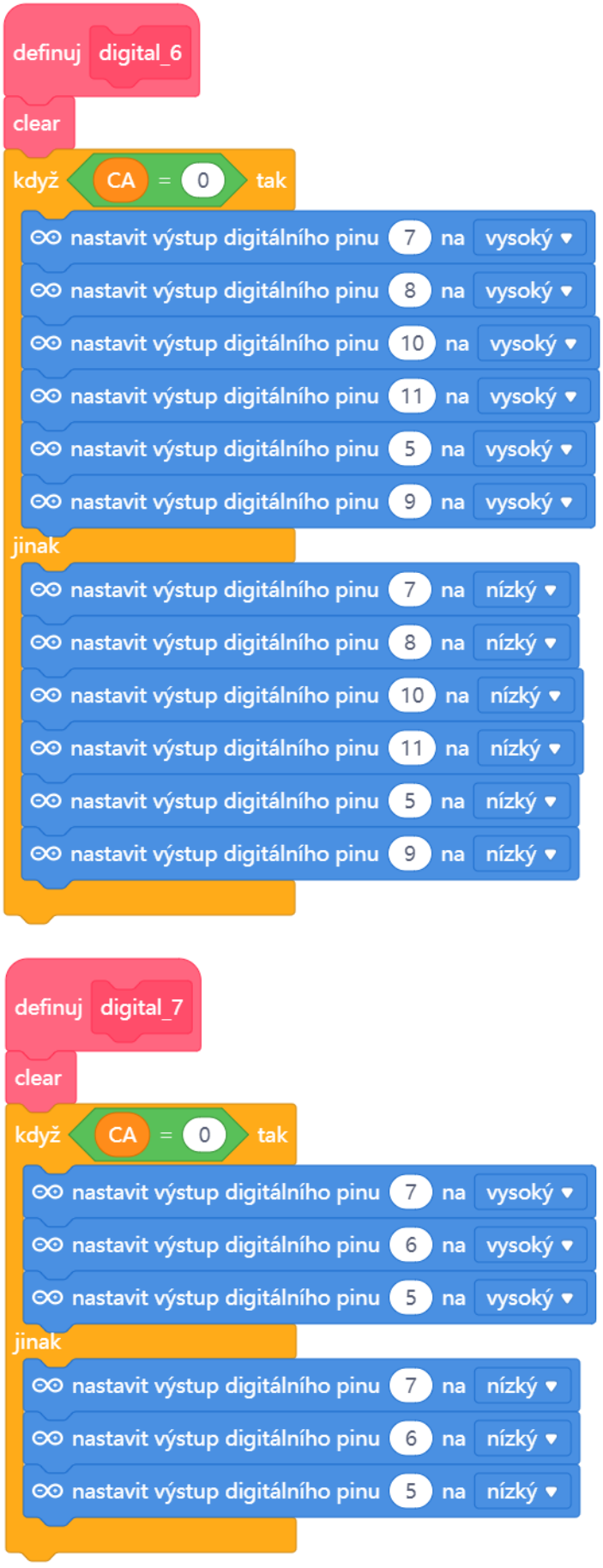

nastavit výstup digitálního pinu 11 na vysoký
nastavit výstup digitálního pinu 5 na vysoký
nastavit výstup digitálního pinu 6 na vysoký
jinak
nastavit výstup digitálního pinu 7 na nízký
nastavit výstup digitálního pinu 8 na nízký
nastavit výstup digitálního pinu 10 na nízký
nastavit výstup digitálního pinu 11 na nízký
nastavit výstup digitálního pinu 5 na nízký
nastavit výstup digitálního pinu 6 na nízký
definuj digital_1
clear
když CA = 0 tak
nastavit výstup digitálního pinu 5 na vysoký
nastavit výstup digitálního pinu 6 na vysoký
jinak
nastavit výstup digitálního pinu 5 na nízký
nastavit výstup digitálního pinu 6 na nízký
definuj digital_2
clear
když CA = 0 tak
nastavit výstup digitálního pinu 7 na vysoký
nastavit výstup digitálního pinu 6 na vysoký
nastavit výstup digitálního pinu 9 na vysoký
Pokračování kódu dále

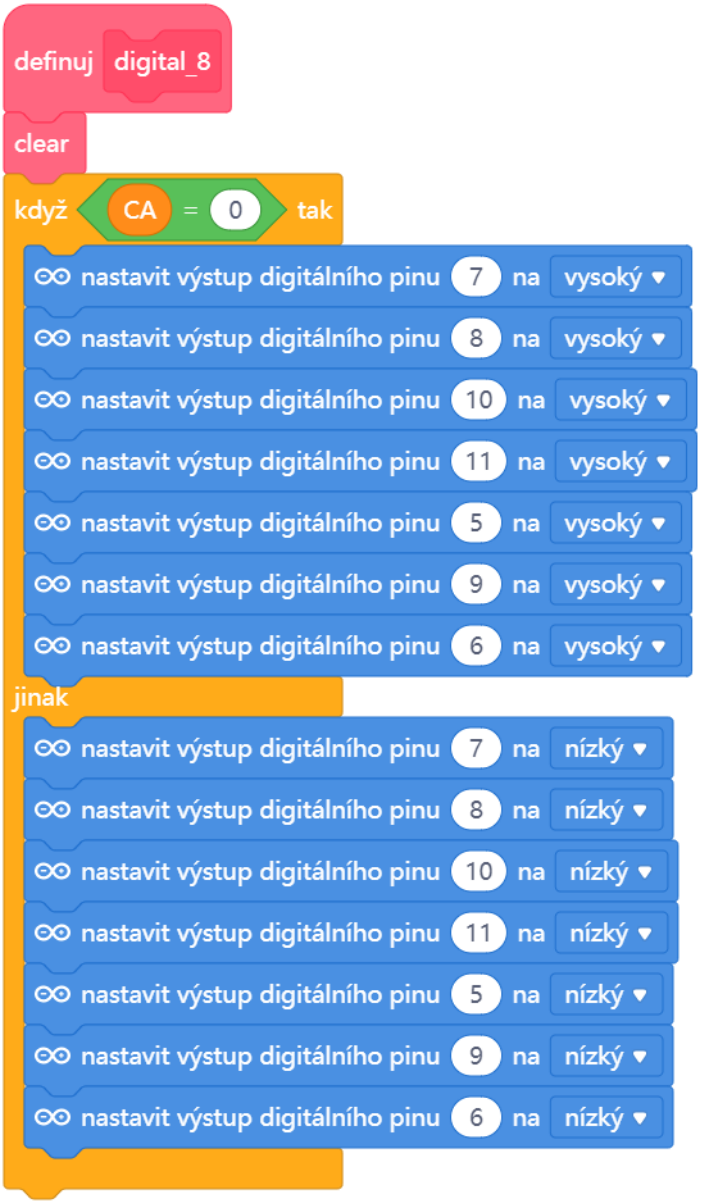

nastavit výstup digitálního pinu 10 na vysoký
nastavit výstup digitálního pinu 11 na vysoký
jinak
nastavit výstup digitálního pinu 7 na nízký
nastavit výstup digitálního pinu 6 na nízký
nastavit výstup digitálního pinu 9 na nízký
nastavit výstup digitálního pinu 10 na nízký
nastavit výstup digitálního pinu 11 na nízký

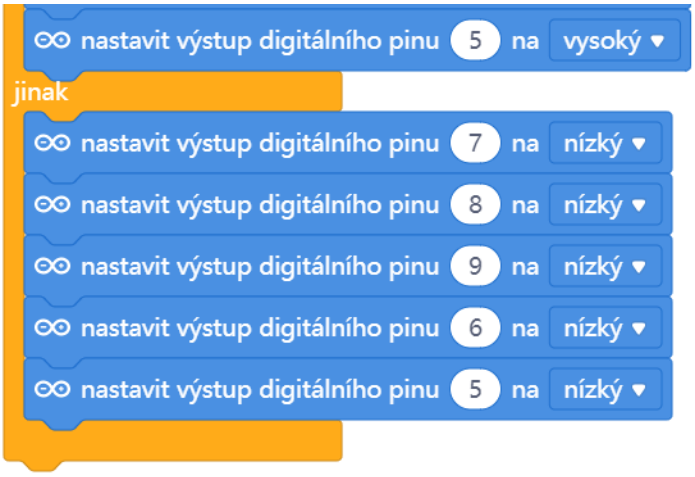

definuj digital_3
clear
když CA = 0 tak
nastavit výstup digitálního pinu 7 na vysoký
nastavit výstup digitálního pinu 6 na vysoký
nastavit výstup digitálního pinu 9 na vysoký
nastavit výstup digitálního pinu 5 na vysoký
nastavit výstup digitálního pinu 11 na vysoký
jinak
nastavit výstup digitálního pinu 7 na nízký
nastavit výstup digitálního pinu 6 na nízký
nastavit výstup digitálního pinu 9 na nízký
nastavit výstup digitálního pinu 5 na nízký
nastavit výstup digitálního pinu 11 na nízký


definuj digital_4
clear
když CA = 0 tak
nastavit výstup digitálního pinu 8 na vysoký
nastavit výstup digitálního pinu 9 na vysoký
nastavit výstup digitálního pinu 6 na vysoký
nastavit výstup digitálního pinu 5 na vysoký
jinak
nastavit výstup digitálního pinu 8 na nízký
nastavit výstup digitálního pinu 9 na nízký
nastavit výstup digitálního pinu 6 na nízký
nastavit výstup digitálního pinu 5 na nízký

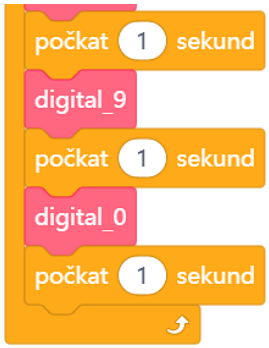

definuj digital_5
clear
když CA = 0 tak
nastavit výstup digitálního pinu 8 na vysoký
nastavit výstup digitálního pinu 9 na vysoký
nastavit výstup digitálního pinu 5 na vysoký
nastavit výstup digitálního pinu 11 na vysoký
nastavit výstup digitálního pinu 7 na vysoký
jinak
nastavit výstup digitálního pinu 8 na nízký
nastavit výstup digitálního pinu 9 na nízký
nastavit výstup digitálního pinu 5 na nízký
nastavit výstup digitálního pinu 11 na nízký
nastavit výstup digitálního pinu 7 na nízký


definuj digital_6
clear
když CA = 0 tak
nastavit výstup digitálního pinu 7 na vysoký
nastavit výstup digitálního pinu 8 na vysoký
nastavit výstup digitálního pinu 10 na vysoký
nastavit výstup digitálního pinu 11 na vysoký
nastavit výstup digitálního pinu 5 na vysoký
nastavit výstup digitálního pinu 9 na vysoký
jinak
nastavit výstup digitálního pinu 7 na nízký
nastavit výstup digitálního pinu 8 na nízký
nastavit výstup digitálního pinu 10 na nízký
nastavit výstup digitálního pinu 11 na nízký
nastavit výstup digitálního pinu 5 na nízký
nastavit výstup digitálního pinu 9 na nízký
definuj digital_7
clear
když CA = 0 tak
nastavit výstup digitálního pinu 7 na vysoký
nastavit výstup digitálního pinu 6 na vysoký
nastavit výstup digitálního pinu 5 na vysoký
jinak
nastavit výstup digitálního pinu 7 na nízký
nastavit výstup digitálního pinu 6 na nízký
nastavit výstup digitálního pinu 5 na nízký


definuj digital_8
clear
když CA = 0 tak
nastavit výstup digitálního pinu 7 na vysoký
nastavit výstup digitálního pinu 8 na vysoký
nastavit výstup digitálního pinu 10 na vysoký
nastavit výstup digitálního pinu 11 na vysoký
nastavit výstup digitálního pinu 5 na vysoký
nastavit výstup digitálního pinu 9 na vysoký
nastavit výstup digitálního pinu 6 na vysoký
jinak
nastavit výstup digitálního pinu 7 na nízký
nastavit výstup digitálního pinu 8 na nízký
nastavit výstup digitálního pinu 10 na nízký
nastavit výstup digitálního pinu 11 na nízký
nastavit výstup digitálního pinu 5 na nízký
nastavit výstup digitálního pinu 9 na nízký
nastavit výstup digitálního pinu 6 na nízký
definuj digital_9
clear
když CA = 0 tak
nastavit výstup digitálního pinu 7 na vysoký
nastavit výstup digitálního pinu 8 na vysoký
nastavit výstup digitálního pinu 9 na vysoký
nastavit výstup digitálního pinu 6 na vysoký
Pokračování kódu dále




A na závěr konečně hlavní program:




Celý program včetně podprogramu lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-19/19-Sedmisegmentovy-LED-zobrazovac.mblock.

*Jak vidno, nakonec celý program včetně všech podprogramů získal skutečně na úctyhodných rozměrech!*

### Vysvětlení kódu

Tento program vychází asi z prvního nápadu, podle kterého bychom program vytvořili. Je zde hlavní program, který postupně rozsvěcí čísla 0, 1, …, 9. Pro rozsvícení potřebných segmentů daného čísla je speciálně vytvořen podprogram – například pro rozsvícení číslice 0 to je podprogram `digital_0`. V každém podprogramu je nejdříve celý zobrazovač zhasnut pomocí podprogramu `clear`, pak jsou rozsvíceny jen ty segmenty zobrazovače, které zobrazí požadovanou číslici. Jelikož je celý program vytvořen pro oba typy LED zobrazovače, tedy se společnou anodou CA a i se společnou katodou CC. Typ zobrazovače se nastavuje pomocí proměnné CA v úvodu hlavního programu. Všechny podprogramy proto obsahují úvodní podmínku, která tyto případy rozděluje a dle potřeby nastavuje výstupní digitální piny modulu Arduino buď na vysokou (HIGH), nebo nízkou (LOW) úroveň.

## Logická výstavba programu, přehlednost a rozsah kódu

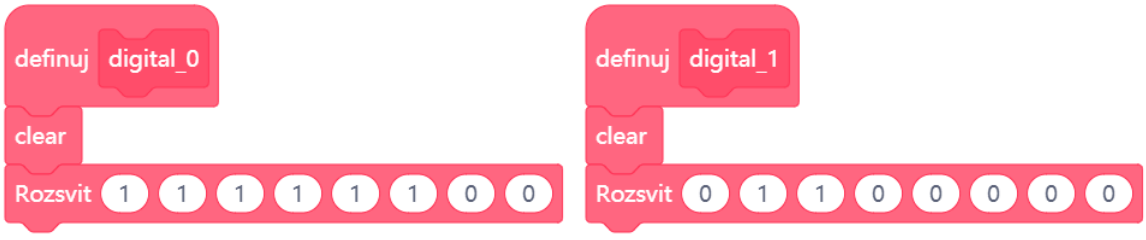
Výše uvedený kód funguje, to je asi to, co by nás jako programátory mělo v prvé řadě zajímat. Otázkou je, zda výše uvedený kód programu je skutečně tím nejlepším řešením. Konkrétně při pohledu na předešlý kód nás určitě musí ohromit jeho neskutečný rozsah. Je to způsobeno tím, že program je napsán pro oba typy LED zobrazovačů, tedy CC i CA, tedy všechny podprogramy `digital_0`, `digital_1`, …, `digital_9` obsluhují veškeré nastavení LED segmentů dvakrát (jednou pro CA, podruhé pro CC). Navíc je zde pro každou zobrazovanou číslici založen speciální podprogram. To také celkový rozsah kódu zvětší. Protože tvorba programů není jen o vytvoření funkčního kódu, což se nám teď již povedlo, ale i o jeho jednoduchosti a určité univerzálnosti, zkusíme původní rozsáhlý kód programu předělat.

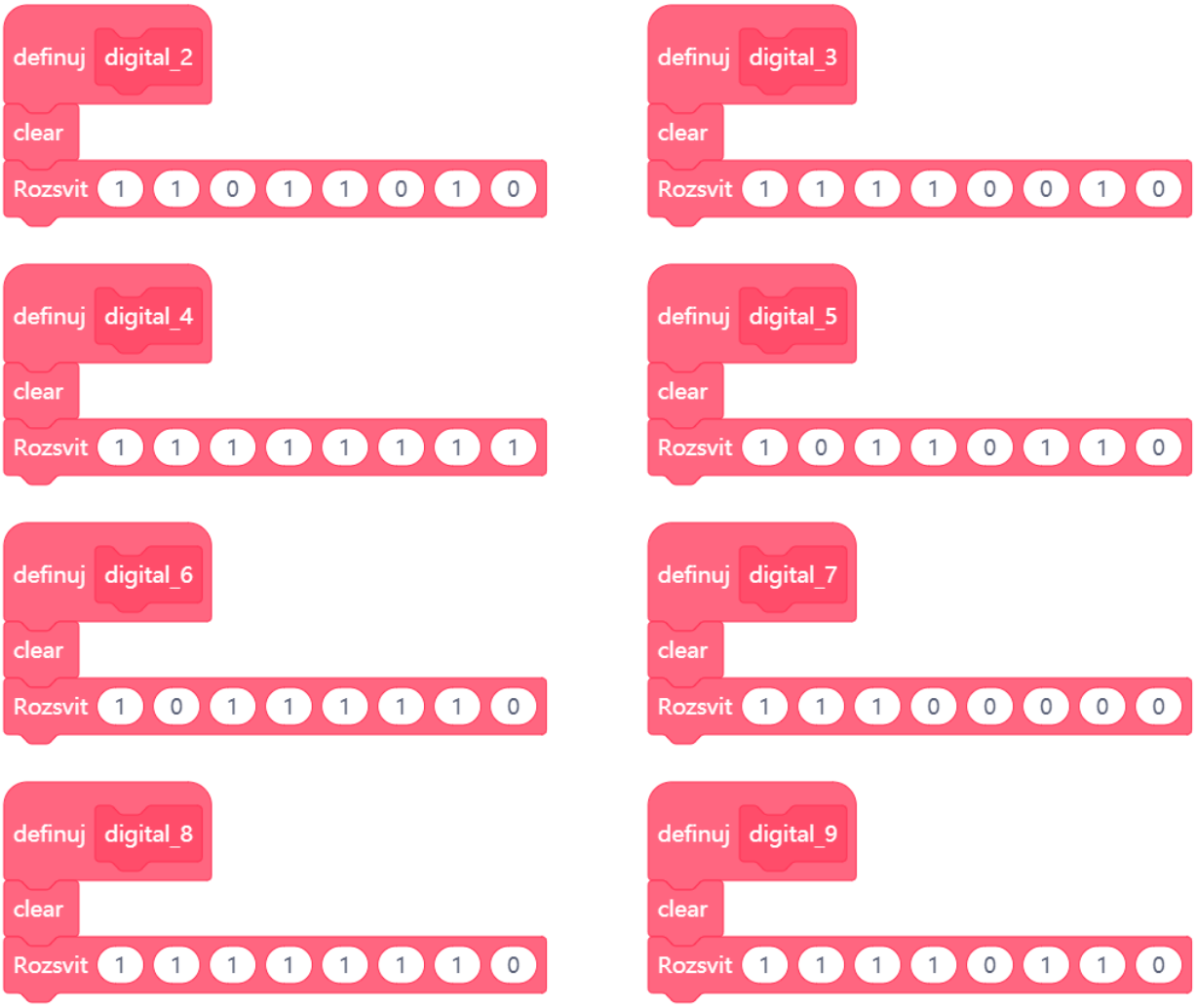
Kdyby se nám podařilo vyřešit odsluhu jednotlivých segmentů LED zobrazovače jedním obecným podprogramem, mohli bychom kód podprogramů `digital_0`, `digital_1`, …, `digital_9` buď výrazně zjednodušil, nebo dokonce zcela vynechat. Prvním pokusem o zjednodušení programu tedy bude

redukce příkazových bloků v podprogramech `digital_0`, `digital_1`, …, `digital_9`. Vytvoříme si univerzální podprogram `Rozsvit`, který bude mít vstupní parametry `a`, `b`, `c`, `d`, `e`, `f`, `g` a `dp`, do kterých se bude zadávat, který ze segmentů (a–dp) má být rozsvícen. Pochopitelně podprogram `Rozsvit` bude řešit i to, zda je použit zobrazovač CC nebo CA. Tím nebude třeba testovat typ zobrazovače v podprogramu každé zobrazené číslice. Úsporu kódu, zejména v podprogramech `digital_0`, `digital_1`, …, `digital_9`, uvidíme v následujících ukázkách kódu. Jako první je pochopitelně uveden kód podprogramu `Rozvit`, ve kterém dle vstupních parametrů `a–dp` dojde k rozsvícení daných LED segmentů (a–dp).

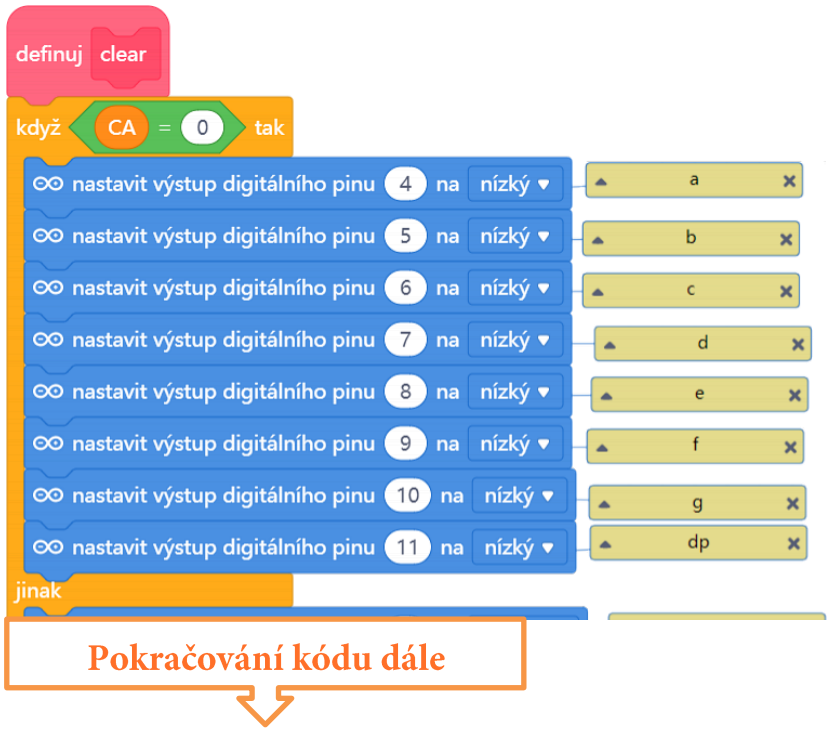

Pochopitelně je v kódu podprogramu `Rozvit` rozdělen podle typu nastaveného zobrazovače. Je škoda, že prostředí mBlock ve svých standardních programových blocích nemá možnost zapsat na digitální výstup hodnotu proměnné, pak by bylo možné kód podprogramu `Rozvit` zredukovat ještě na polovinu.

Máme-li podprogram `Rozsvit`, můžeme všechny podprogramy `digital_0`, `digital_1`, …, `digital_9` nyní napsat jen pomocí podprogramu `Rozsvit` s příslušnými parametry. Jejich kód se tím výrazně zjednoduší.






A na závěr opět hlavní program a podprogram `clear`, které zatím nezaznamenaly žádnou změnu:




nastavit výstup digitálního pinu 4 na vysoký
a
nastavit výstup digitálního pinu 5 na vysoký
b
nastavit výstup digitálního pinu 6 na vysoký
c
nastavit výstup digitálního pinu 7 na vysoký
d
nastavit výstup digitálního pinu 8 na vysoký
e
nastavit výstup digitálního pinu 9 na vysoký
f
nastavit výstup digitálního pinu 10 na vysoký
g
nastavit výstup digitálního pinu 11 na vysoký
dp

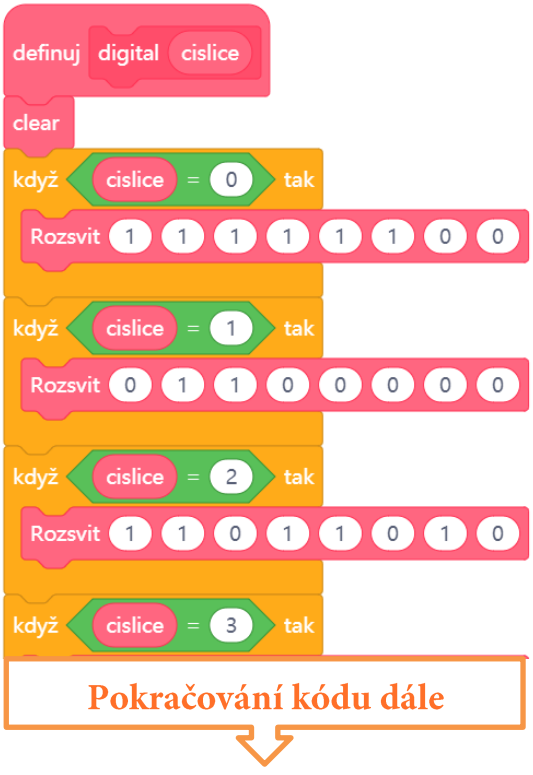

když se Arduino Uno spustí
nastavte CA na 0
CA = 1 (společná ANODA)
CA = 0 (společná KATODA)
opakuj stále
digital_1
počkat 1 sekund
digital_2
počkat 1 sekund
digital_3
počkat 1 sekund
digital_4
počkat 1 sekund
digital_5
počkat 1 sekund
digital_6
počkat 1 sekund
digital_7
počkat 1 sekund
digital_8
počkat 1 sekund
digital_9
Pokračování kódu dále

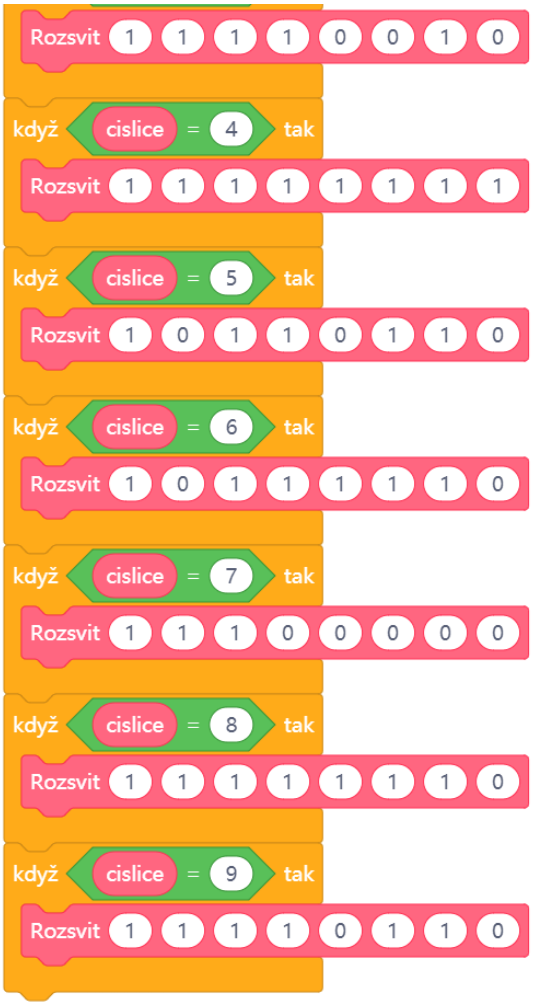

Celý program včetně podprogramu lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-19/19-Sedmisegmentovy-LED-zobrazovac-ver2.mblock.

### 2. ÚPRAVA

V porovnání s předchozí verzí programu je jasně vidět, jak se těla podprogramů `digital_0` až `digital_9` díky použití podprogramu `Rozsvit` výrazně zredukovaly. Skoro by se dalo říci, že je otázkou, zda je nutné pro tak jednoduché operace (*smaž segmentovku a rozsviť zadané segmenty*) mít kód rozdělen na deset podprogramů. Toto je jistě správná úvaha a ještě ji využijeme nadále. Abychom tuto optimalizaci tedy posunuli dále, zkusíme všechny tyto podprogramy nahradit jedním společným. Vytvoříme si podprogram, který by měl nahradit všech deset podprogramů `digital_0` až `digital_9`. Tento „univerzální" podprogram, pojmenujme ho například `digital`, bude mít vstupní parametr určující číslici, která se má zobrazit a dle toho bude volat podprogram `Rozsvit` – stejně jako to doteď dělaly podprogramy `digital_0` až `digital_9`.

Kód podprogramu `digital` bude vypadat následujícím způsobem. Vstupní parametr `cislice` určuje číslici, která má být zobrazena – vlastně tím říká, zda má podprogram `digital` být použit jako `digital_0` … nebo `digital_9`.

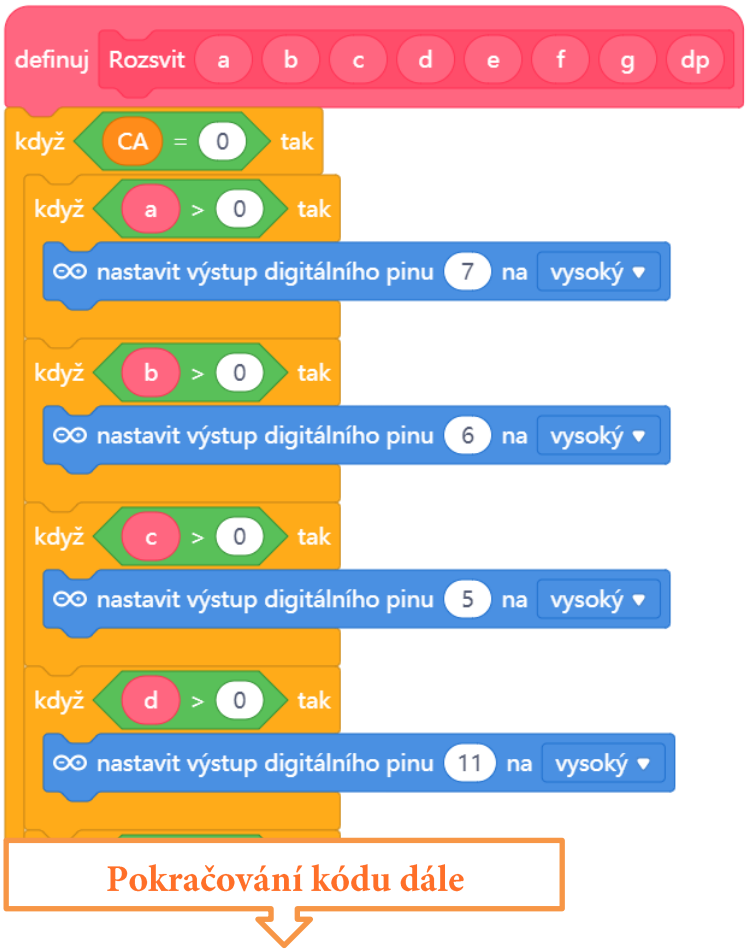


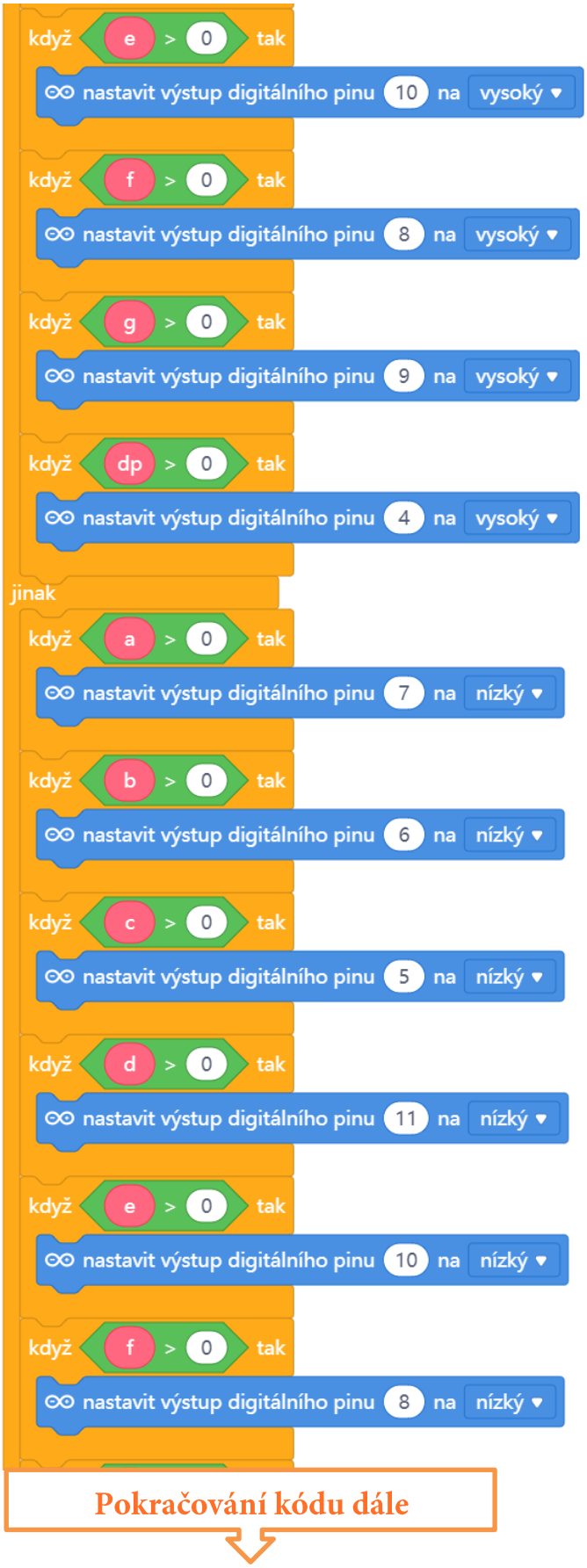



Výhodou podprogramu `digital` je i to, že díky jeho vstupnímu parametru jej můžeme volat pomocí proměnné. Není třeba ji tedy v programovém kódu volat desetkrát za sebou, jako to bylo při volání podprogramů `digital_0` … nebo `digital_9`. Tím, že je možné vstupní parametr předávat pomocí proměnné `hod`, jejíž hodnota se může postupně zvyšovat z hodnoty 0 na hodnotu 9, je možné tento podprogram volat opakovaně v programové smyčce. Celkový program lze tedy tímto způsobem přepsat. Tím se tedy zredukuje i hlavní kód.

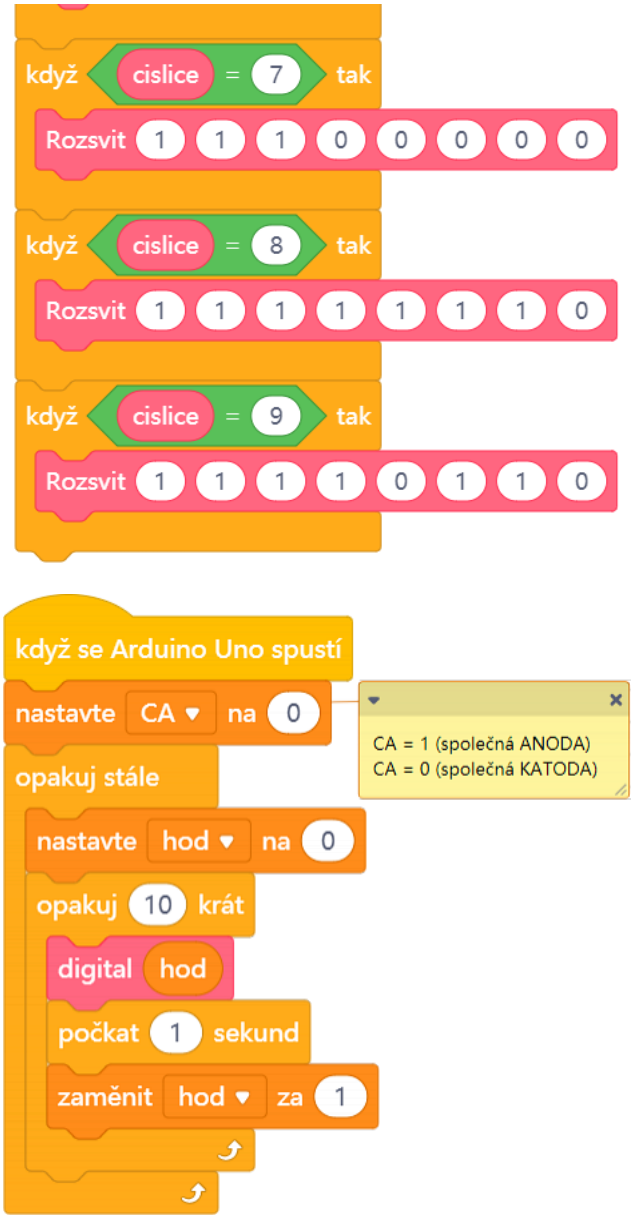
Celkový program (včetně všech podprogramů) bude vypadat následujícím způsobem:


nastavit výstup digitálního pinu 6 na nízký
c
nastavit výstup digitálního pinu 7 na nízký
d
nastavit výstup digitálního pinu 8 na nízký
e
nastavit výstup digitálního pinu 9 na nízký
f
nastavit výstup digitálního pinu 10 na nízký
g
nastavit výstup digitálního pinu 11 na nízký
dp
jinak
nastavit výstup digitálního pinu 4 na vysoký
a
nastavit výstup digitálního pinu 5 na vysoký
b
nastavit výstup digitálního pinu 6 na vysoký
c
nastavit výstup digitálního pinu 7 na vysoký
d
nastavit výstup digitálního pinu 8 na vysoký
e
nastavit výstup digitálního pinu 9 na vysoký
f
nastavit výstup digitálního pinu 10 na vysoký
g
nastavit výstup digitálního pinu 11 na vysoký
dp


definuj Rozsvit a b c d e f g dp
když CA = 0 tak
když a > 0 tak
nastavit výstup digitálního pinu 7 na vysoký
když b > 0 tak
nastavit výstup digitálního pinu 6 na vysoký
když c > 0 tak
nastavit výstup digitálního pinu 5 na vysoký
když d > 0 tak
nastavit výstup digitálního pinu 11 na vysoký
Pokračování kódu dále


když e > 0 tak
nastavit výstup digitálního pinu 10 na vysoký
když f > 0 tak
nastavit výstup digitálního pinu 8 na vysoký
když g > 0 tak
nastavit výstup digitálního pinu 9 na vysoký
když dp > 0 tak
nastavit výstup digitálního pinu 4 na vysoký
jinak
když a > 0 tak
nastavit výstup digitálního pinu 7 na nízký
když b > 0 tak
nastavit výstup digitálního pinu 6 na nízký
když c > 0 tak
nastavit výstup digitálního pinu 5 na nízký
když d > 0 tak
nastavit výstup digitálního pinu 11 na nízký
když e > 0 tak
nastavit výstup digitálního pinu 10 na nízký
když f > 0 tak
nastavit výstup digitálního pinu 8 na nízký
Pokračování kódu dále


když g > 0 tak
nastavit výstup digitálního pinu 9 na nízký
když dp > 0 tak
nastavit výstup digitálního pinu 4 na nízký
definuj digital cislice
clear
když cislice = 0 tak
Rozsvit 1 1 1 1 1 1 0 0
když cislice = 1 tak
Rozsvit 0 1 1 0 0 0 0 0
když cislice = 2 tak
Rozsvit 1 1 0 1 1 0 1 0
když cislice = 3 tak
Rozsvit 1 1 1 1 0 0 1 0
když cislice = 4 tak
Rozsvit 1 1 1 1 1 1 1 1
když cislice = 5 tak
Rozsvit 1 0 1 1 0 1 1 0
když cislice = 6 tak
Rozsvit 1 0 1 1 1 1 1 0
Pokračování kódu dále




Celý program včetně podprogramu lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-19/19-Sedmisegmentovy-LED-zobrazovac-ver3.mblock.

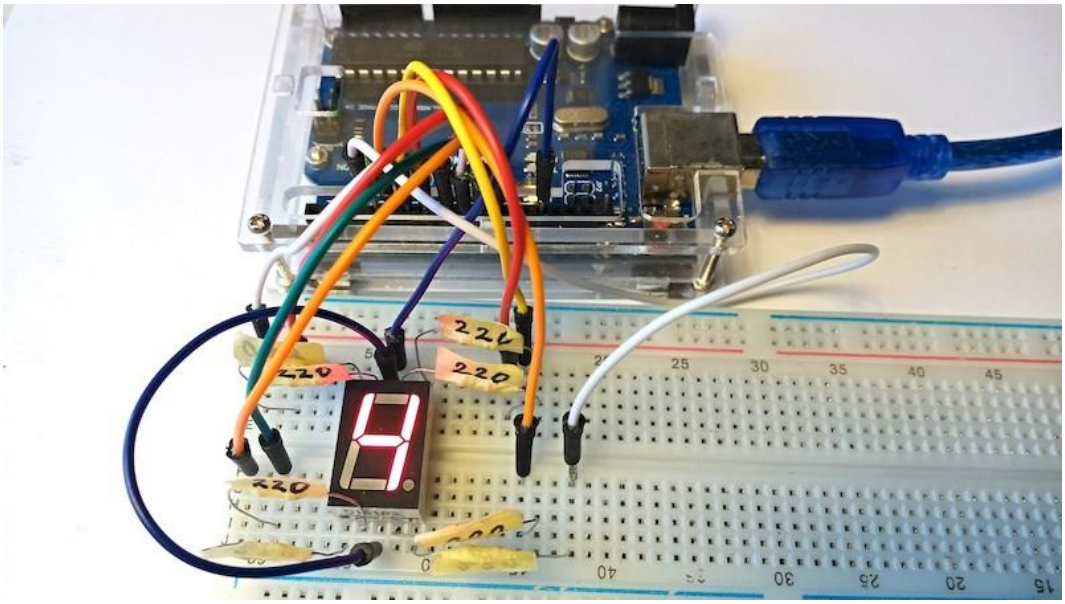
Porovnejme rozsah původního kódu programu, kde jsme postupně volali deset různých podprogramů, s poslední verzí, kde je v cyklu s deseti kroky opakovaně volán univerzální podprogram s rozdílným parametrem. Je pravda, že i nyní je rozsah upraveného programu poměrně veliký, ale v porovnání s první verzí je zde výrazný rozdíl.

*Tento příklad optimalizace programového kódu nám v tomto konkrétním případě měl ukázat, jak je vhodné u složitějších projektů nejdříve celý program řádně promyslet, pak zvolit „taktiku“ obecnějšího využití podprogramů a nakonec stejně celý program několikrát předělat. ☺ Nevěříte? Tak se podívejte na popis*

*programu v následující lekci, tam je popsána ještě jedna úprava zde použitých dvou základních podprogramů („digital“ a „Rozvit“) pro obsluhu LED zobrazovače. A jistě by se našlo další vylepšení i tam.*

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Nyní bychom měli vidět, jak na zobrazovači v sekundových intervalech postupně přeblikávají znaky 1, 2, …, 9 a 0.



# Lekce 20 –Čtyřmístný LED zobrazovač

## Úvod

V minulé lekci jsme poznali řízení LED zobrazovače. Jelikož se tyto zobrazovače používají poměrně často v různých aplikacích, nevyrábějí se jen jednočíslicové, ale lze použít i vícemístný LED zobrazovač. V této lekci si tedy ukážeme, jak ovládat čtyřmístný (sedmisegmentový) LED zobrazovač. Výsledkem našeho snažení by měly být jednoduché stopky, které budou postupně počítat vteřiny.

## Použité komponenty

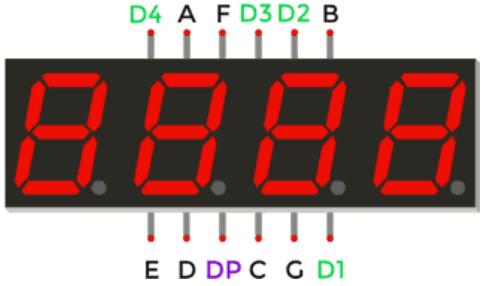
- modul Arduino
- USB kabel
- čtyřmístný LED zobrazovač
- 8× rezistor (220 Ω)
- vodiče pro nepájivé pole
- nepájivé pole

## Princip

Čtyřmístný LED zobrazovač, který nyní budeme používat, vlastně odpovídá předchozímu sedmisegmentovému zobrazovači, jen obsahuje čtveřici společných vstupů (společná anoda nebo katoda – dle typu), kterými se nastavuje jeden ze čtyř LED zobrazovačů, která má aktuálně svítit. V daný okamžik se tedy může zobrazovat jenom jedna číslice. Jak tedy na čtyřmístném LED zobrazovači zobrazit čtyři číslice najednou? Jelikož je lidské oko poměrně pomalé, můžeme při velmi rychlém střídání zobrazených čísel na jednotlivých místech zobrazovače vnímat zároveň rozsvícené všechny čtyři zobrazovače. A všechny při tom mohou zobrazovat zcela jinou číslici.

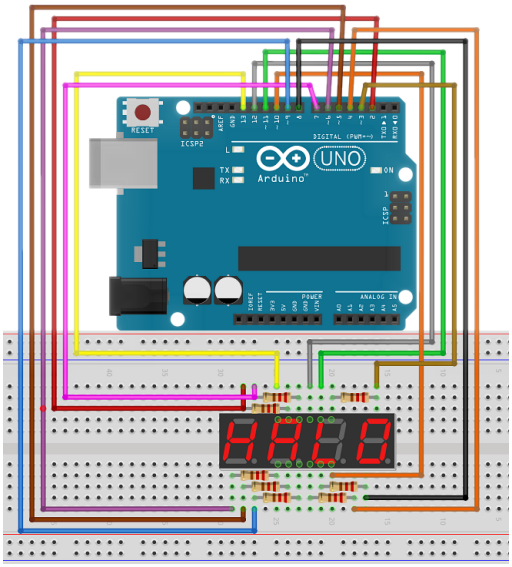
## Postup experimentu

Opět je třeba zjistit, jaký typ zobrazovače je v naší sadě (zda CC nebo CA). Postup je stejný jako v předešlé lekci. Následující obrázek ukazuje rozložení vývodů zobrazovače, vývody d1–d4 jsou společné vývody (tedy buď společná anoda, nebo katoda). Proti jednomu z těchto vývodů je třeba zkoušet rozsvítiti některý ze segmentů pomocí knoflíkové 3 V baterie. **Opět je třeba si dát pozor na zbytečně dlouhé rozsvícení LED segmentu!**

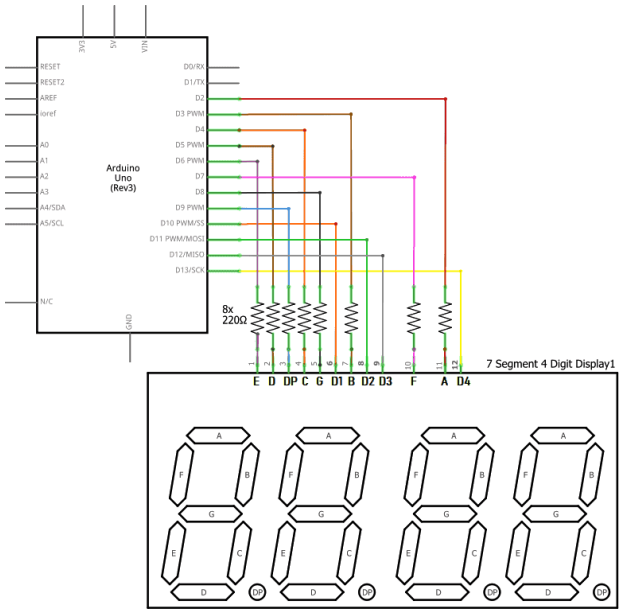


**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Oproti předchozí lekci zde není třeba rozdělovat zapojení podle použitého typu zobrazovače (CC nebo CA). Společné vývody d1–d4 jsou připojeny k digitálním výstupům modulu Arduino, typ zobrazovače tedy ošetříme softwarově nastavenou výstupní úrovní na těchto pinech. Zapojení mezi 4× LED zobrazovačem a modulem Arduino uvádí tabulka.

**Blokové schéma**

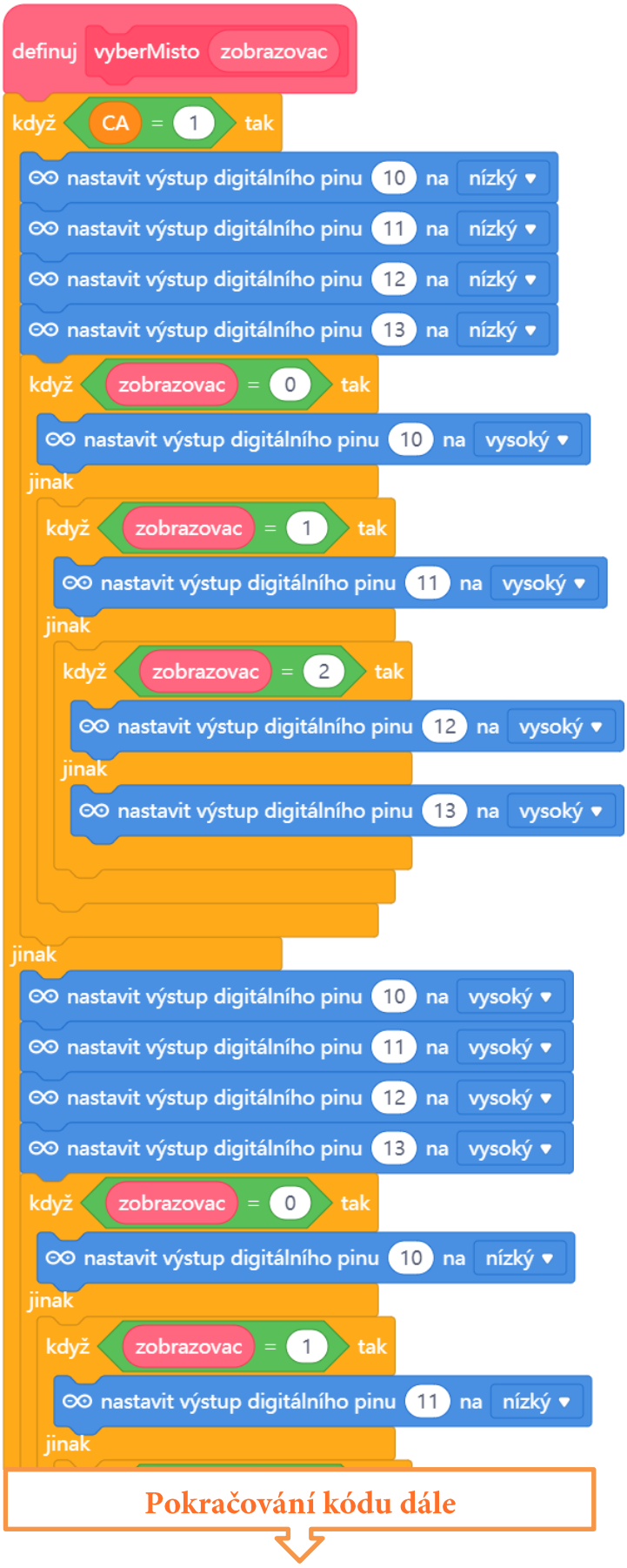


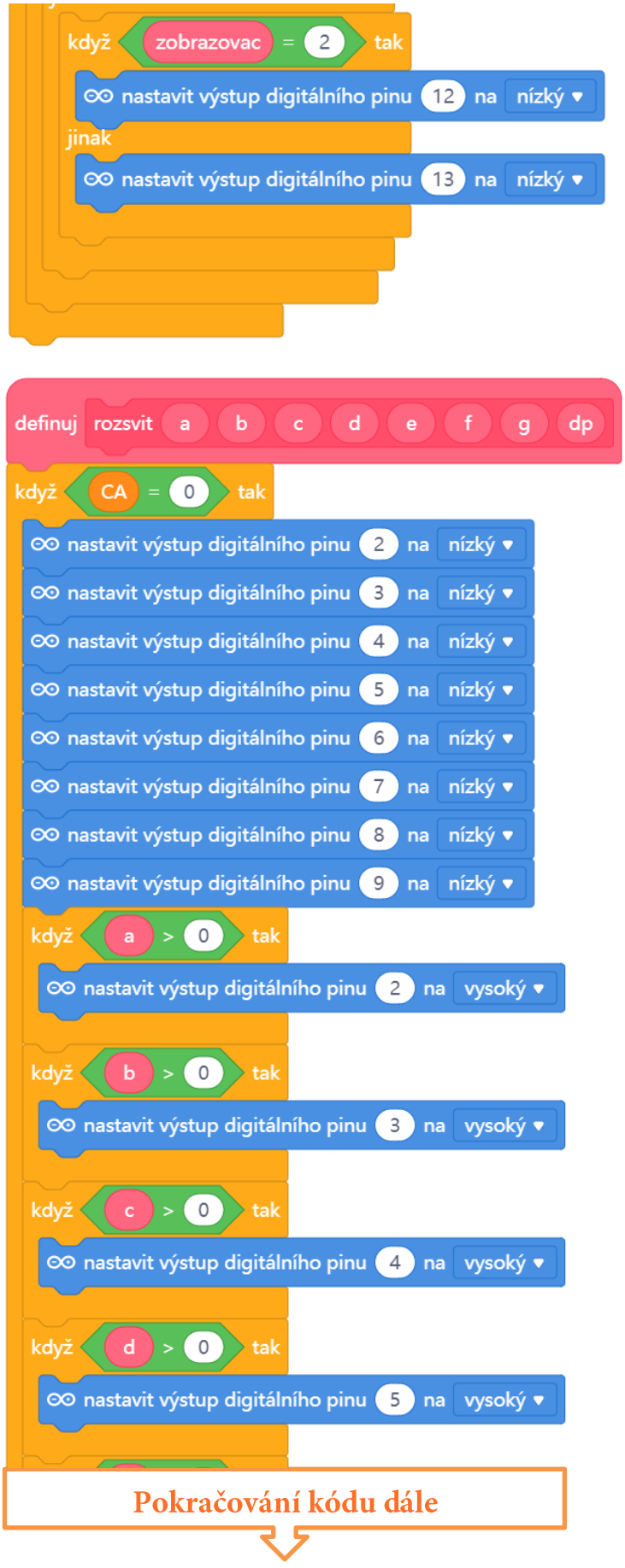
**Elektronické schéma**

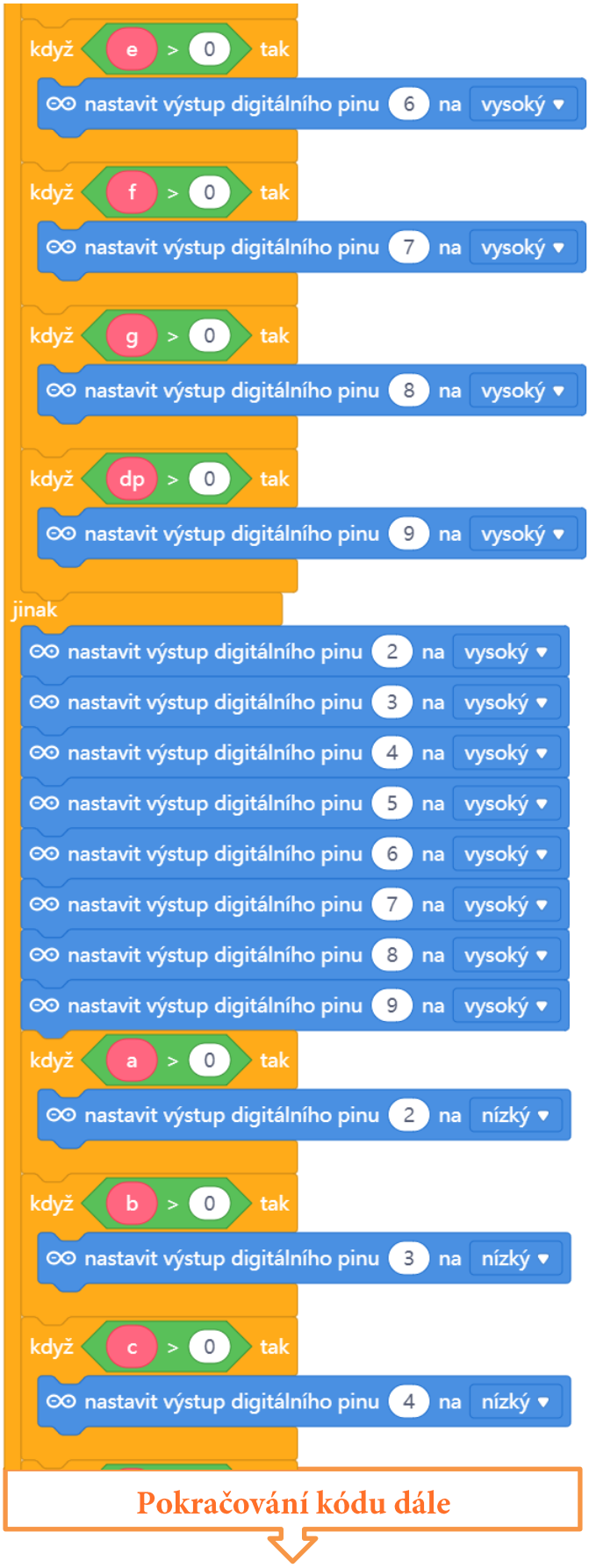



| 4× LED zobrazovač | modul Arduino |
|---|---|
| A | 2 |
| B | 3 |
| C | 4 |
| D | 5 |
| E | 6 |
| F | 7 |
| G | 8 |
| DP | 9 |
| D1 | 10 |
| D2 | 11 |
| D3 | 12 |
| D4 | 13 |

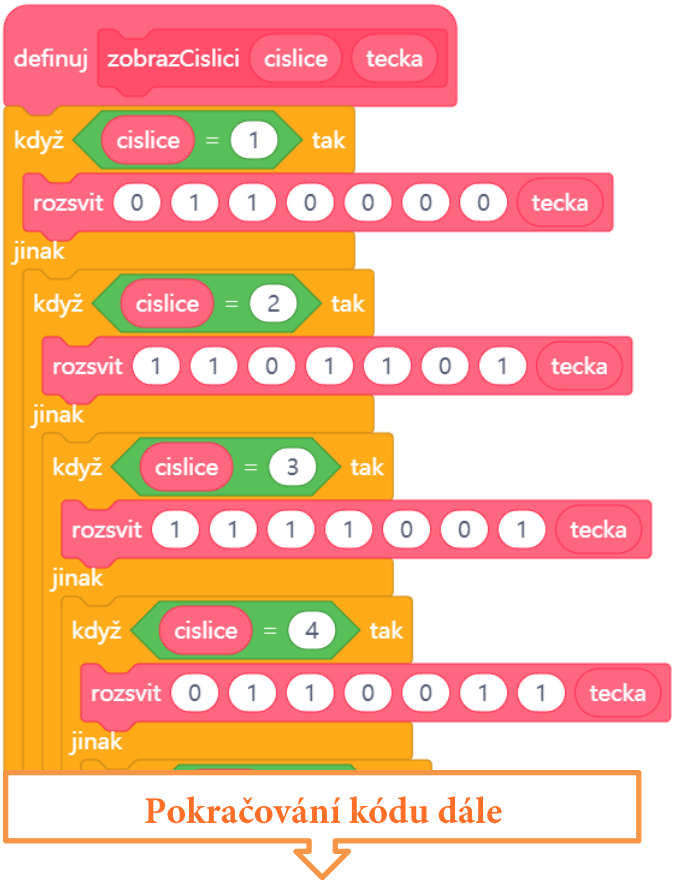
**Krok 2:** V prostředí mBlock vytvoříme následující program. Asi bude dobré opět program vytvořit „odzadu“, tedy nejdříve od jednotlivých podprogramů (`vyberMisto`, `rozsvit` a `zobrazCislici`), pak dokončit hlavní program.

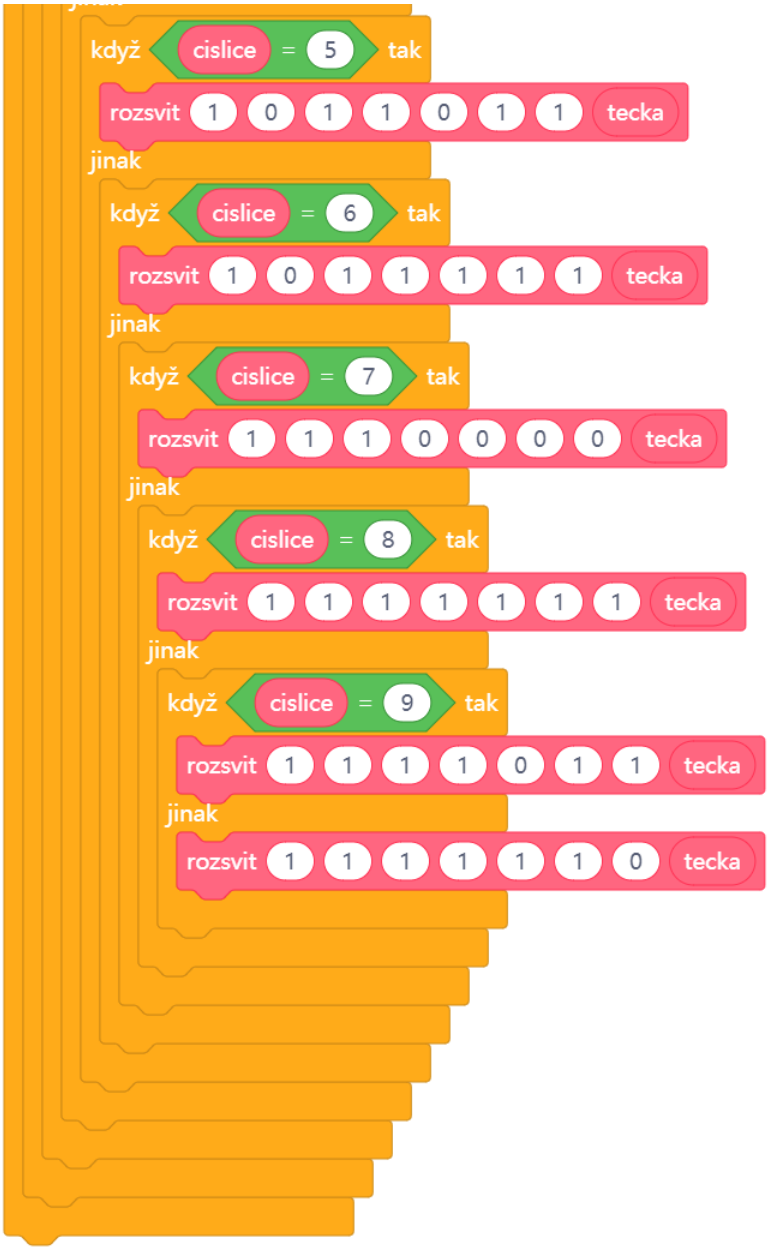

definuj vyberMisto zobrazovac
když CA = 1 tak
nastavit výstup digitálního pinu 10 na nízký
nastavit výstup digitálního pinu 11 na nízký
nastavit výstup digitálního pinu 12 na nízký
nastavit výstup digitálního pinu 13 na nízký
když zobrazovac = 0 tak
nastavit výstup digitálního pinu 10 na vysoký
jinak
když zobrazovac = 1 tak
nastavit výstup digitálního pinu 11 na vysoký
jinak
když zobrazovac = 2 tak
nastavit výstup digitálního pinu 12 na vysoký
jinak
nastavit výstup digitálního pinu 13 na vysoký
jinak
nastavit výstup digitálního pinu 10 na vysoký
nastavit výstup digitálního pinu 11 na vysoký
nastavit výstup digitálního pinu 12 na vysoký
nastavit výstup digitálního pinu 13 na vysoký
když zobrazovac = 0 tak
nastavit výstup digitálního pinu 10 na nízký
jinak
když zobrazovac = 1 tak
nastavit výstup digitálního pinu 11 na nízký
jinak
Pokračování kódu dále


když zobrazovac = 2 tak
nastavit výstup digitálního pinu 12 na nízký
jinak
nastavit výstup digitálního pinu 13 na nízký
definuj rozsvit a b c d e f g dp
když CA = 0 tak
nastavit výstup digitálního pinu 2 na nízký
nastavit výstup digitálního pinu 3 na nízký
nastavit výstup digitálního pinu 4 na nízký
nastavit výstup digitálního pinu 5 na nízký
nastavit výstup digitálního pinu 6 na nízký
nastavit výstup digitálního pinu 7 na nízký
nastavit výstup digitálního pinu 8 na nízký
nastavit výstup digitálního pinu 9 na nízký
když a > 0 tak
nastavit výstup digitálního pinu 2 na vysoký
když b > 0 tak
nastavit výstup digitálního pinu 3 na vysoký
když c > 0 tak
nastavit výstup digitálního pinu 4 na vysoký
když d > 0 tak
nastavit výstup digitálního pinu 5 na vysoký
Pokračování kódu dále


když e > 0 tak
nastavit výstup digitálního pinu 6 na vysoký
když f > 0 tak
nastavit výstup digitálního pinu 7 na vysoký
když g > 0 tak
nastavit výstup digitálního pinu 8 na vysoký
když dp > 0 tak
nastavit výstup digitálního pinu 9 na vysoký
jinak
nastavit výstup digitálního pinu 2 na vysoký
nastavit výstup digitálního pinu 3 na vysoký
nastavit výstup digitálního pinu 4 na vysoký
nastavit výstup digitálního pinu 5 na vysoký
nastavit výstup digitálního pinu 6 na vysoký
nastavit výstup digitálního pinu 7 na vysoký
nastavit výstup digitálního pinu 8 na vysoký
nastavit výstup digitálního pinu 9 na vysoký
když a > 0 tak
nastavit výstup digitálního pinu 2 na nízký
když b > 0 tak
nastavit výstup digitálního pinu 3 na nízký
když c > 0 tak
nastavit výstup digitálního pinu 4 na nízký
Pokračování kódu dále

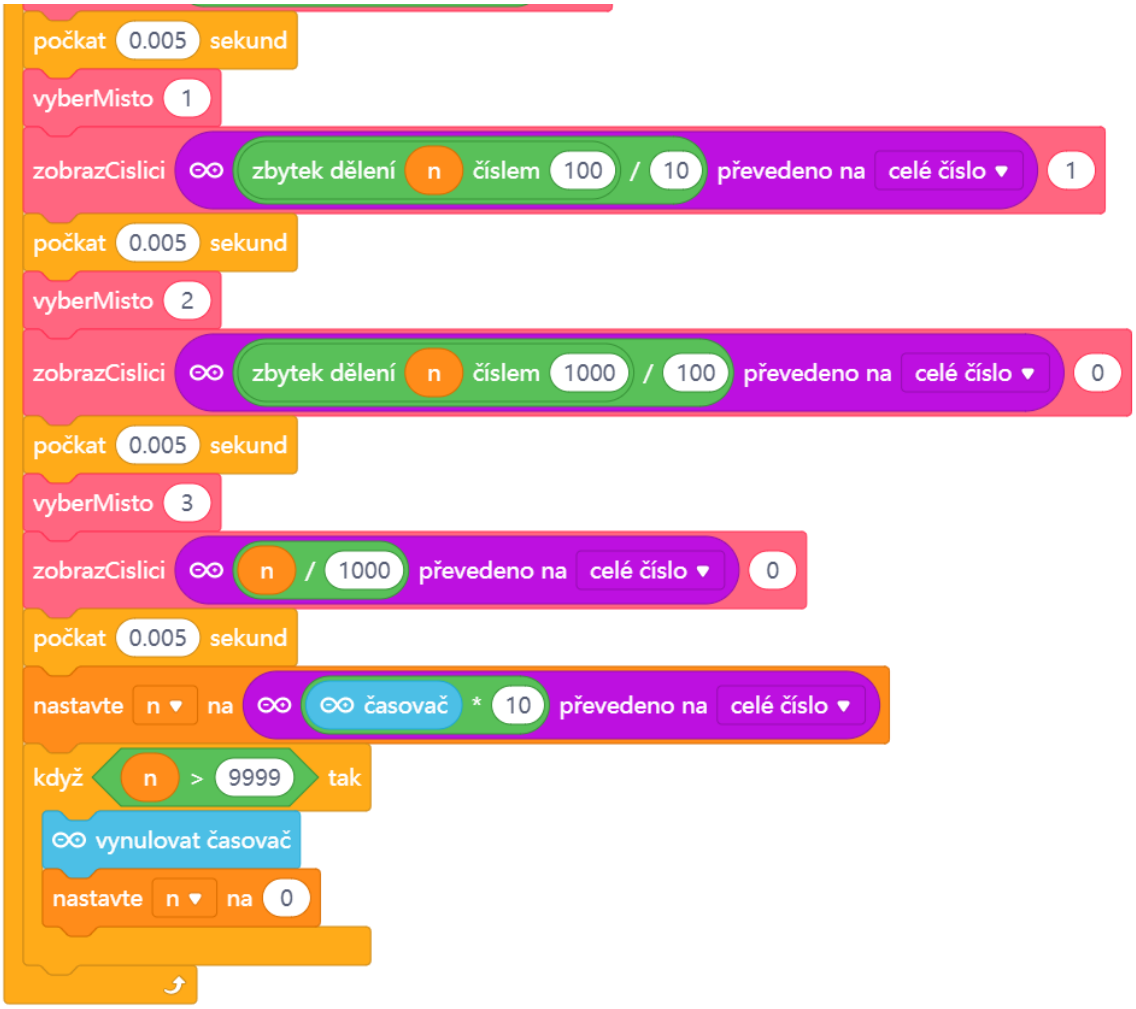

když d > 0 tak
nastavit výstup digitálního pinu 5 na nízký
když e > 0 tak
nastavit výstup digitálního pinu 6 na nízký
když f > 0 tak
nastavit výstup digitálního pinu 7 na nízký
když g > 0 tak
nastavit výstup digitálního pinu 8 na nízký
když dp > 0 tak
nastavit výstup digitálního pinu 9 na nízký


definuj zobrazCislici cislice tecka
když cislice = 1 tak
rozsvit 0 1 1 0 0 0 0 tecka
jinak
když cislice = 2 tak
rozsvit 1 1 0 1 1 0 1 tecka
jinak
když cislice = 3 tak
rozsvit 1 1 1 1 0 0 1 tecka
jinak
když cislice = 4 tak
rozsvit 0 1 1 0 0 1 1 tecka
jinak
Pokračování kódu dále





Pokračování kódu dále




Celý program včetně podprogramu lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-20/20-4mistny-LED-zobrazovac.mblock.

## Vysvětlení kódu

Obecná obsluha sedmisegmentového LED zobrazovače vychází z principu, který jsme si ukázali v minulé lekci. Tedy ze dvou podprogramů – jeden slouží k rozsvícení zadaných segmentů LED zobrazovače a druhý pro zobrazení zadané číslice. Oproti předchozímu programu zde ale došlo k určitému vylepšení, aby i tato část kódu ukázala, že je stále co zlepšovat.

Podprogram `rozsvit`, který má za úkol dle zadaných vstupních parametrů rozsvítit jednotlivé segmenty LED zobrazovače, byl nyní na svém začátku rozšířen o bloky zhasnutí všech segmentů (modré bloky nastavení digitálních výstupů na začátku každé sekce typu zobrazovače). Díky tomu, můžeme nyní tento podprogram používat bez podprogramu pro úplné zhasnutí zobrazovače – v předchozí lekci jsme kvůli tomu měli speciální podprogram `clear` (*zde již není třeba*). Jinak se podprogram `rozsvit` nezměnil. Stále je to jen systém vzájemně navazujících podmínek, které testují vstupní parametry odpovídající daným segmentům (`A`, `B`, …, `DP`) zobrazovače a které pak rozsvěcejí dané segmenty.

Druhá změna proběhla v podprogramu pro zobrazení číslice na zvoleném zobrazovači. Tento podprogram se nyní jmenuje `zobrazCislici` (oproti názvu `digital` z minulé lekce), což ale není tak důležité. Především je třeba se podívat na funkční strukturu podprogramu. Kromě absence příkazů pro zhasnutí zobrazovače (viz předešlý odstavec) je vidět, že tento podprogram je nyní tvořen sérií vzájemně vnořených podmínek. Uvědomíme-li si, jak pracoval v předchozí lekci podprogram `digital`, vidíme, zde určitou neefektivitu. Vstupní parametr určoval, která číslice se má zobrazit. Následovala série na sebe navazujících podmínek, kde se testoval tento parametr. Pokud došlo ke shodě, došlo k zobrazení. Struktura podmínek, které stále na sebe navazovaly, však zapříčinily, že docházelo k testování všech deseti znaků (0–9), i když logicky byl zobrazen jen jeden. Například pokud se zobrazila číslice 1, nadále byl zbytečně testován případ pro číslice 2, 3, …, 9. Přitom bylo jasné, že tyto podmínky už prostě nemohou být splněny. To nutí procesor dělat zbytečnou práci a zbytečně snižuje výpočetní výkon systému. Pokud však podmínky do sebe vnoříme tak, že každá následující je ve větvi „jinak“, je kaskáda následujících podmínek prováděna jen, pokud byla předchozí podmínka nesplněna. Takže, pokud je vstupním parametrem zadáno, že se má zobrazit číslice 1, případy 2, 3, … 9 již testovány nebudou (leží v jiné větvi programu).

Pozorný čtenář se ale jistě všiml dalšího rozdílu podprogramu `zobrazCislici` oproti podprogramu `digital` (z minulé lekce), a to je druhý vstupní parametr – `tecka`. Jelikož v tomto programu, kde chceme simulovat stopky, chceme zobrazovat nejen počet sekund, ale i jejich desetin, je třeba zobrazit za některou z číslic desetinnou tečku. To právě zařizuje vstupní parametr `tecka`. Parametr `tecka` je v podprogramu `zobrazCislici` předán podprogramu `rozsvit` jako parametr pro rozsvícení desetinné tečky. Je-li tedy parametr `tecka` nastaven na 0, dojde k zobrazení číslice bez desetinné tečky (stejně, jako v předchozí lekci). Naopak pro vstupní hodnotu vyšší než 0 (např. 1) bude zadaná číslice na daném místě LED zobrazovače rozsvícena s desetinnou tečkou.

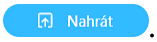
*Čistě pro zájemce doporučujeme se vrátit k předešlé lekci a zkusit si podle výše popsaných principů předělat podprogramy „digit“ a „rozsvit“. Získáme tím univerzálnější podobu programu a ušetříme proceduru „clear“.*

Hlavní program má dva základní úkoly: zaprvé měřit uběhlý čas, zadruhé obsluhovat všechny čtyři LED zobrazovače a zobrazovat na ně příslušné symboly. Jelikož pracujeme se čtyřnásobným zobrazovačem, musí být před zobrazením výstupu zvoleno místo, na kterém se má zadaný symbol zobrazit. K tomuto účelu máme podprogram `vyberMisto`, jehož vstupní parametr `zobrazovac` určí, který společný vývod daného zobrazovače má být připojen. Jedná se tedy o sérii podmínek, které dle vstupního parametru a typu zobrazovače zapojí na výstupní digitální pin společných vývodů jednotlivých zobrazovačů potřebnou výstupní úroveň. Například pro zobrazení číslice na prvním zobrazovači (typ CC) je potřeba jeho společný vývod nastavit na nízkou úroveň (LOW) a společné vývody ostatních tří zobrazovačů nastavit na vysokou úroveň (HIGH). Tím máme vyřešené zobrazování hodnoty času. Teď jej ještě musíme změřit.

V hlavním programu využijeme pro měření času programového bloku obsluhující interní časovač modulu Arduino s pomocí pracovní proměnné `n`. Obě tyto hodnoty na začátku programu nastavíme na hodnotu 0. Do proměnné `n` je postupně dosazován desetinásobek hodnoty interního časovače (jako celé číslo), čímž získáváme počet desetin sekund uplynulých od spuštění programu. Z této hodnoty je postupně získáván

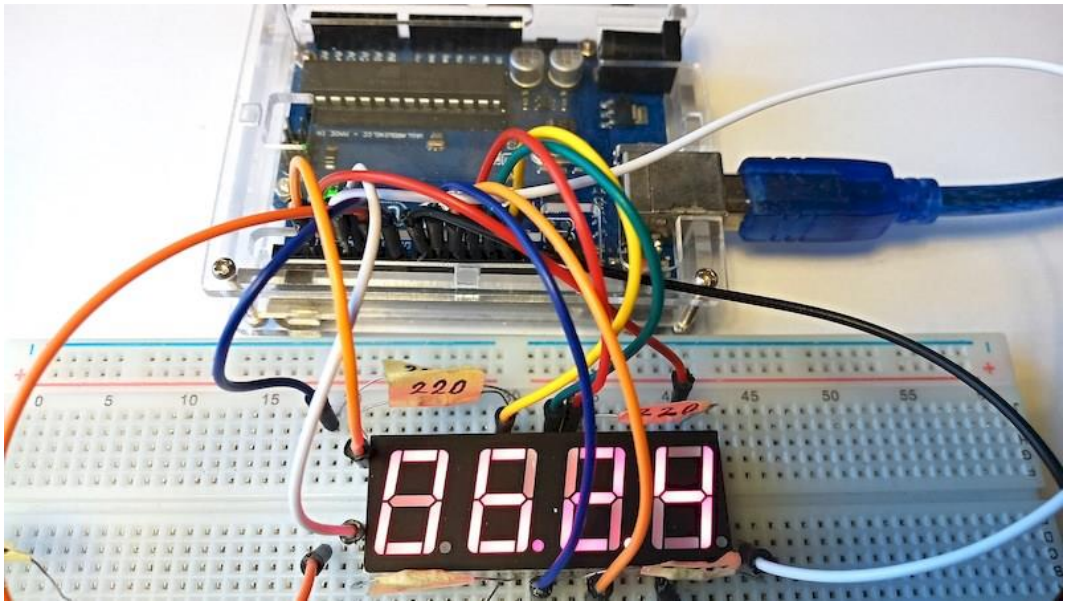
počet desetin sekundy (zbytek po dělení 10), počet sekund (zbytek po dělení 100) vydělený deseti a převeden na celé číslo, počet desítek sekund (zbytek po dělení 1000) vydělená deseti a převeden na celé číslo a pochopitelně počet stovek sekund (proměnná `n` vydělená 1000 a převedeno na celé číslo). A to jsou právě ty číslice, které budeme podprogramem `zobrazCislici` postupně vypisovat na LED zobrazovače. Nesmíme zapomenout za zobrazovačem odpovídajícímu počtu sekund rozsvítit desetinnou tečku hodnotou 1 vstupního parametru `tecka`. Po zobrazení každé číslice program počká na 5 ms, aby jej mohlo oko dobře zaznamenat, pak následuje zobrazení další číslice. Výsledným efektem je zobrazení všech 4 číslic odpovídajícím uplynulému času.

Poslední podmínka v nekonečné smyčce „opakuj stále“ hlavního programu testuje hodnotu proměnné na překročení hodnoty 9999 (čas 999,9 s). Při překročení tohoto maximálního času dojde k vynulování interního časovače modulu Arduino i proměnné `n`. Čas pak běží opět od nuly.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

.

**Krok 4:** Nyní můžeme vidět, jak na všech čtyřech místech LED zobrazovače se postupně zobrazuje hodnota běžícího času, včetně desetin vteřiny. Také ale můžeme vidět, že na prvních místech se zobrazuje číslice 0 – například 002.3. To není moc hezké. Což takhle se zamyslet a upravit programový kód, aby zobrazovaný formát času byl bez těch nul na začátku!



# Lekce 21 – LED matice 8×8

## Úvod

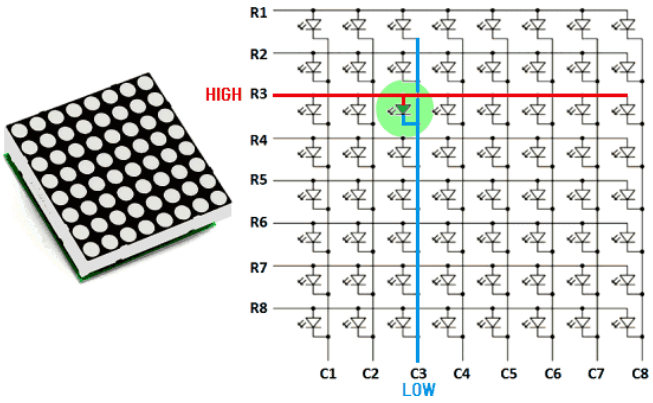
V předešlých lekcích jsme se seznámili se segmentovými LED zobrazovači. Viděli jsme, že jeho nabídka zobrazovaných symbolů byla však poměrně omezená. Nyní se podíváme na další typ zobrazovače, který by nám mohl umožnit zobrazit trochu více – například písmena celé abecedy nebo jednoduché obrázky. Součástkou, kterou budeme používat, je LED matnice o rozměrech 8×8. Maticové LED zobrazovače mohou uspokojit potřeby různých aplikací. Je to především díky jejich dlouhé životnosti, relativně nízkým nákladům na nákup i provoz, vysokému jasu, ale třeba i určité „nepromokavosti“. Zkrátka tyto zobrazovače mají širokou perspektivu rozvoje.

## Použité komponenty

- Modul Arduino
- USB kabel
- LED matice 8×8
- 8× rezistor (220 Ω)
- nepájivé pole
- vodiče pro nepájivé pole + vodiče typy „samec-samice“

## Princip

LED matice 8×8 se skládá z 64 LED, kde každá LED je umístěna na průsečíku řádku a sloupce. Když je elektrická úroveň určité řady vysoká a elektrická úroveň určitého sloupce nízká, pak se LED, která je jejich průsečíkem, rozsvítí. Chceme-li rozsvítit LED na první pozici, měli bychom nastavit řádek 1 na vysokou úroveň (HIGH) a sloupec 1 na nízkou úroveň (LOW). Chceme-li rozsvítit LED na prvním řádku, měli bychom nastavit řádek 1 na vysokou úroveň (HIGH) a sloupce (1, 2, 3, 4, 5, 6, 7, 8) na nízkou úroveň (LOW). Jedině tak se všechny LED diody na prvním řádku rozsvítí.



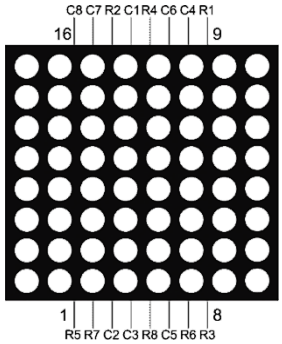

Základním principem při použití LED zobrazovačů je jejich postupné rozsvěcení po řádcích řádků. Již víme, že vývody LED matice tvoří anodovou a katodovou skupinu. Data tvořící budoucí obrázek (snímek) jsou postupně přiváděna na jednotlivé anody, kdy vždy jedna z katod je na nízké úrovni. Tím se rozsvítí první řádek. Následuje obdobné rozsvícení druhého řádku, pak třetího… Postupným vysvícením všech dat na všech katodách dojde k vytvoření tzv. snímku. Princip je obdobný s funkcí obrazovky televize nebo monitoru. Snímková frekvence,

tedy počet zobrazených snímků za sekundu, musí být tak vysoká, aby nepůsobila rušivě na lidské oko. V praxi se používají frekvence od stovek hertzů do jednotek nebo desítek kilohertzů. Neustálé rozsvěcení snímků je poměrně náročná záležitost, jejíž složitost stoupá s počtem prvků matice. Pro matici 8×8 LED (bodů) je zapotřebí 8 bitů pro řízení anodové a dalších 8 bitů pro řízení katodové skupiny. Tímto je zařízeno ovládání pouhých 64 bodů. Přirozeným řešením tohoto problému je využití jednočipového mikropočítače (například Arduino) s dostatečnou kapacitou vstupních a výstupních pinů a výpočetním výkonem.

## Postup experimentu

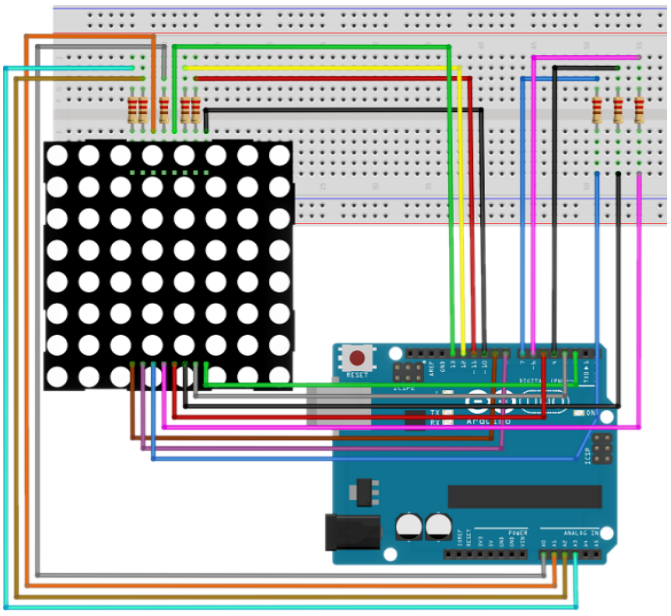
**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Než se pustíme do zapojování obvodu, je třeba se podívat na rozložení pinů s ohledem na číslování řádků (R) a sloupců (C) Rozložení pinů na maticovém displeji versus významy jednotlivých pinů (sloupec/řádek) ukazuje následující tabulka.

| **sloupec (C_)** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| **č. pinu** | 13 | 3 | 4 | 10 | 6 | 11 | 15 | 16 |
| **řádek (R_)** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| **č. pinu** | 9 | 14 | 8 | 12 | 1 | 7 | 2 | 5 |

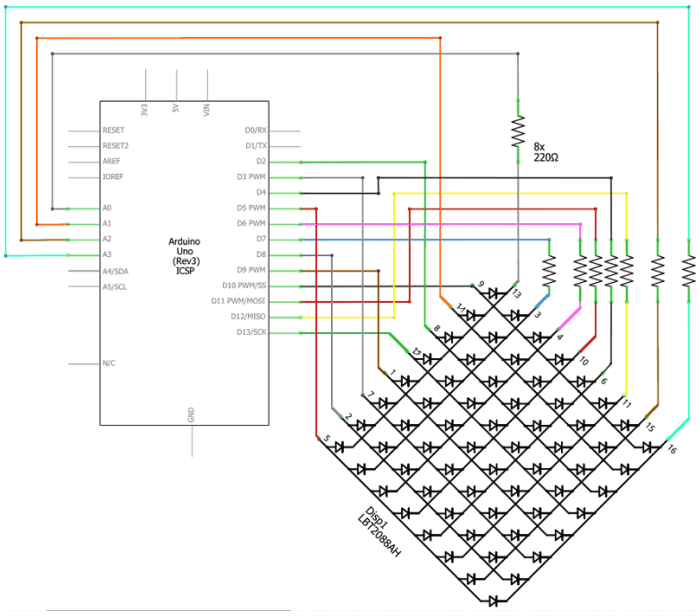



*POZOR, uvedený popis platí pouze pro maticový LED displej, jehož číslo modelu končí BS!*

**Blokové schéma**

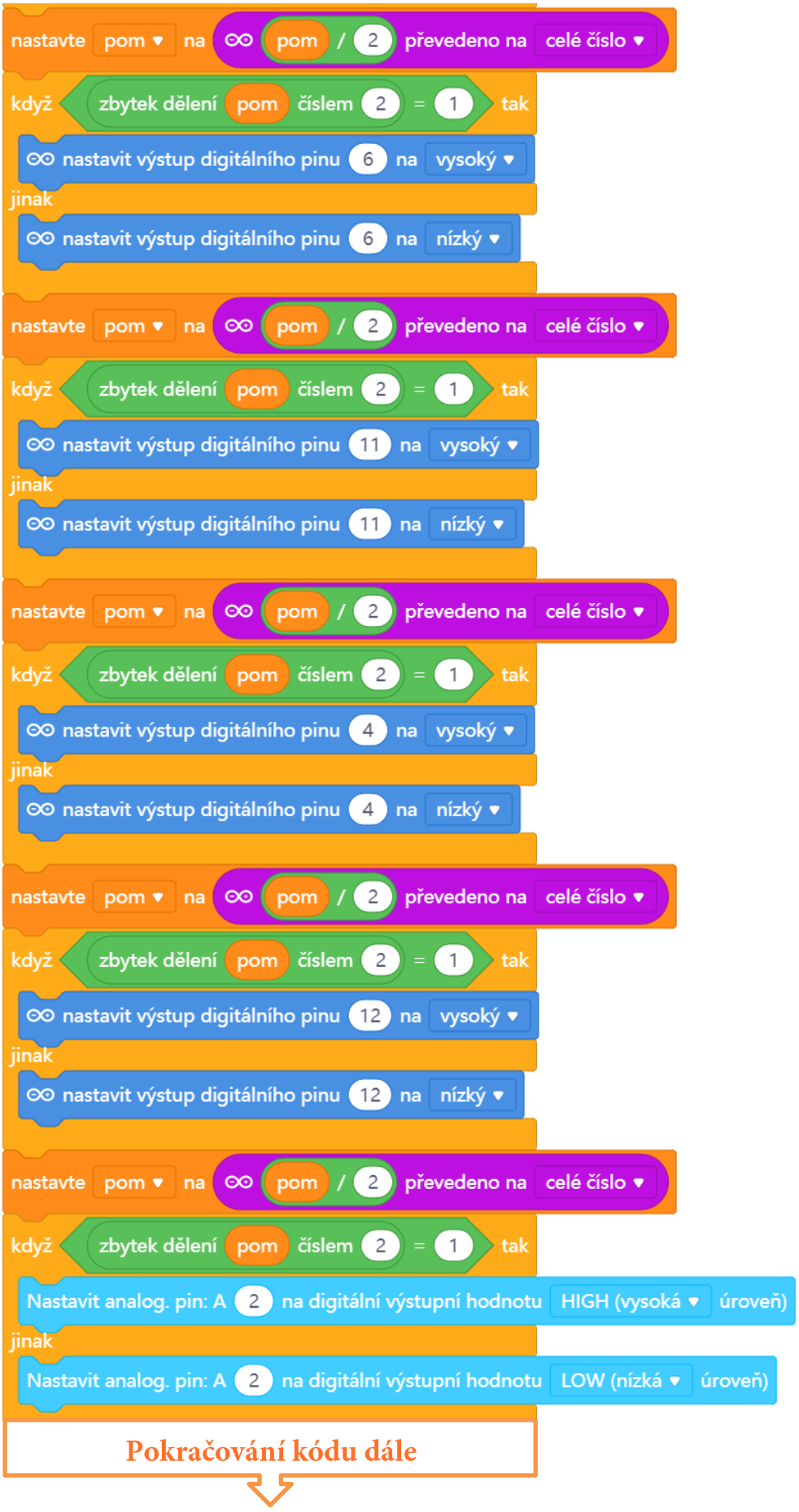


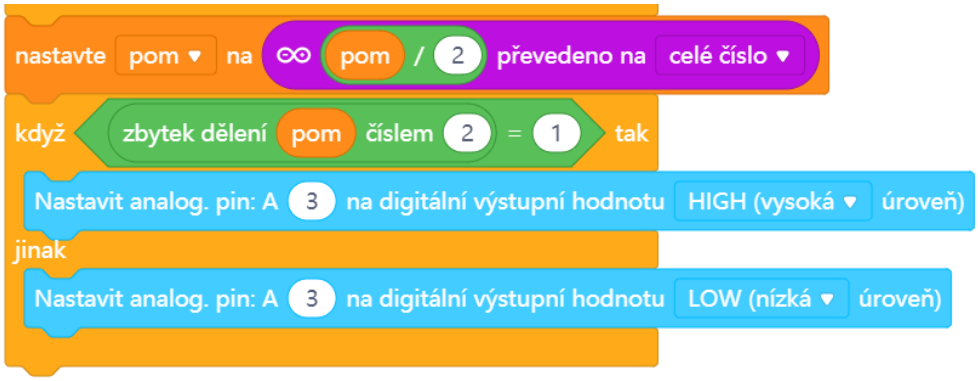
**Elektronické schéma**

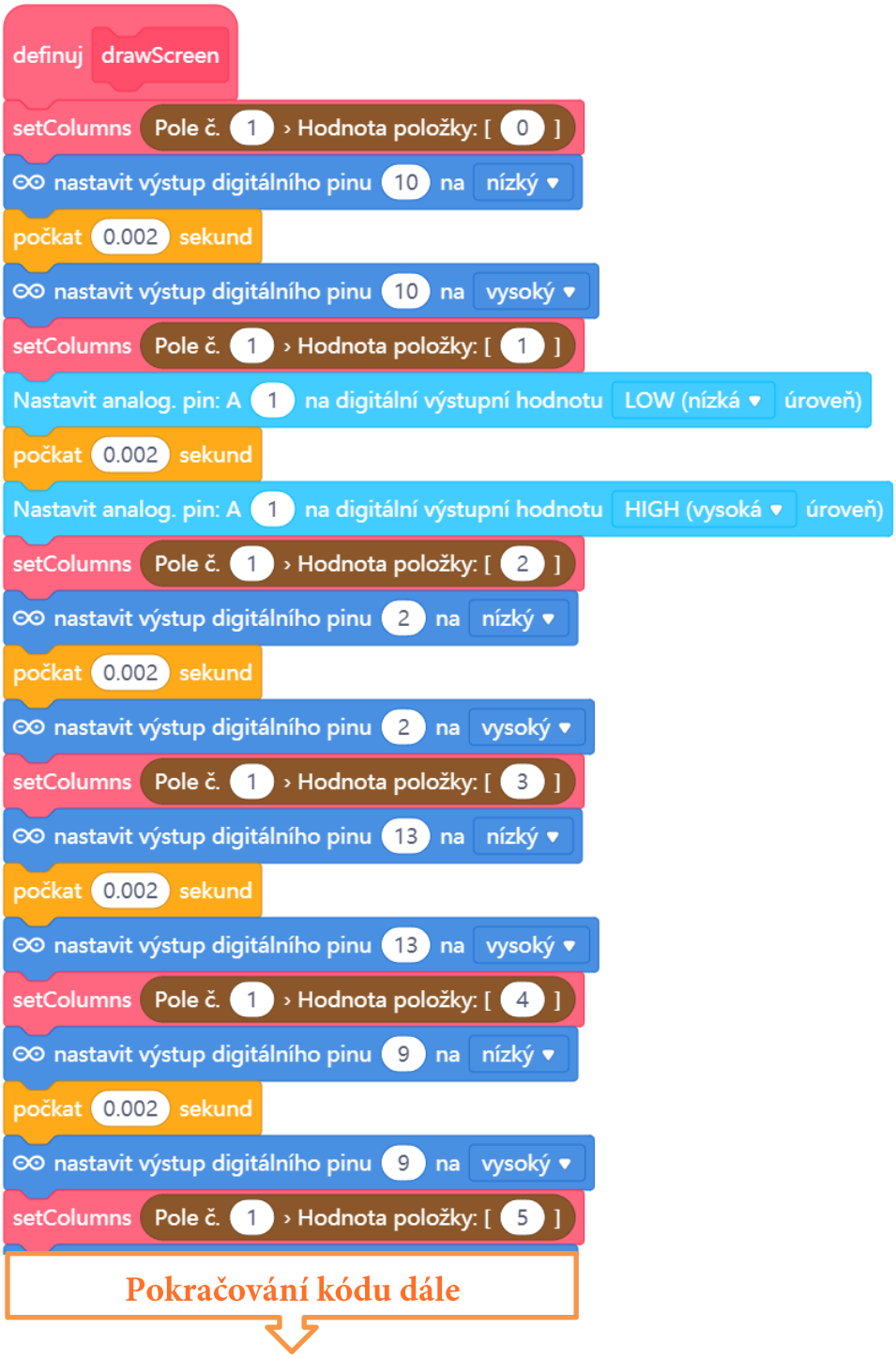


| LED matice 8×8 | modul Arduino | způsob propojení |
|---|---|---|
| 1 | 9 | přímo |
| 2 | 8 | přímo |
| 3 | 7 | přes rezistor |
| 4 | 6 | přes rezistor |
| 5 | 5 | přímo |
| 6 | 4 | přes rezistor |
| 7 | 3 | přímo |
| 8 | 2 | přímo |
| 9 | 10 | přímo |
| 10 | 11 | přes rezistor |
| 11 | 12 | přes rezistor |
| 12 | 13 | přímo |
| 13 | A0 | přes rezistor |
| 14 | A1 | přímo |
| 15 | A2 | přes rezistor |
| 16 | A3 | přes rezistor |

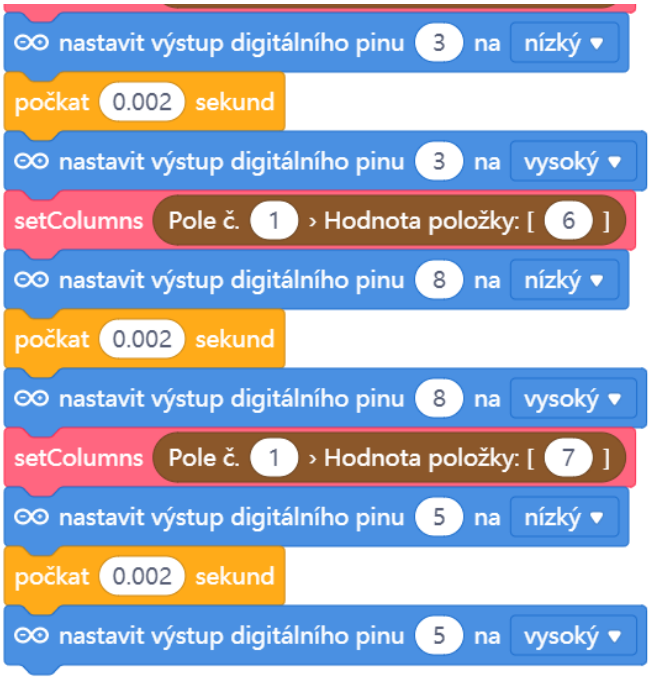
**Krok 2:** V prostředí mBlock vytvoříme následující program. I v tomto programu je několik podprogramů a proto začneme nejdříve s nimi, hlavní program vznikne až na závěr.


nastavte pom na pom / 2 převedeno na celé číslo
když zbytek dělení pom číslem 2 = 1 tak
nastavit výstup digitálního pinu 6 na vysoký
jinak
nastavit výstup digitálního pinu 6 na nízký
nastavte pom na pom / 2 převedeno na celé číslo
když zbytek dělení pom číslem 2 = 1 tak
nastavit výstup digitálního pinu 11 na vysoký
jinak
nastavit výstup digitálního pinu 11 na nízký
nastavte pom na pom / 2 převedeno na celé číslo
když zbytek dělení pom číslem 2 = 1 tak
nastavit výstup digitálního pinu 4 na vysoký
jinak
nastavit výstup digitálního pinu 4 na nízký
nastavte pom na pom / 2 převedeno na celé číslo
když zbytek dělení pom číslem 2 = 1 tak
nastavit výstup digitálního pinu 12 na vysoký
jinak
nastavit výstup digitálního pinu 12 na nízký
nastavte pom na pom / 2 převedeno na celé číslo
když zbytek dělení pom číslem 2 = 1 tak
Nastavit analog. pin: A 2 na digitální výstupní hodnotu HIGH (vysoká úroveň)
jinak
Nastavit analog. pin: A 2 na digitální výstupní hodnotu LOW (nízká úroveň)
Pokračování kódu dále



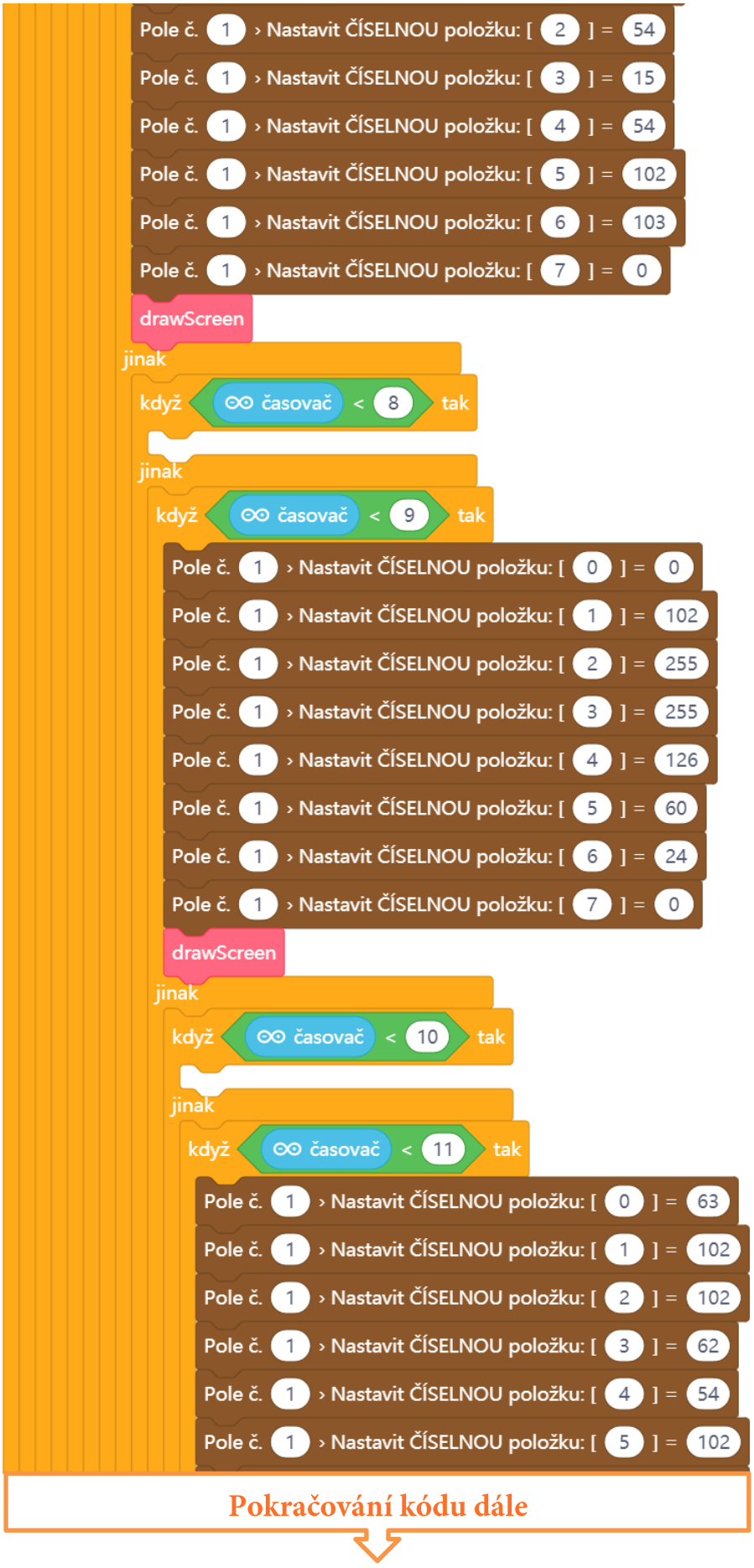

Pokračování kódu dále

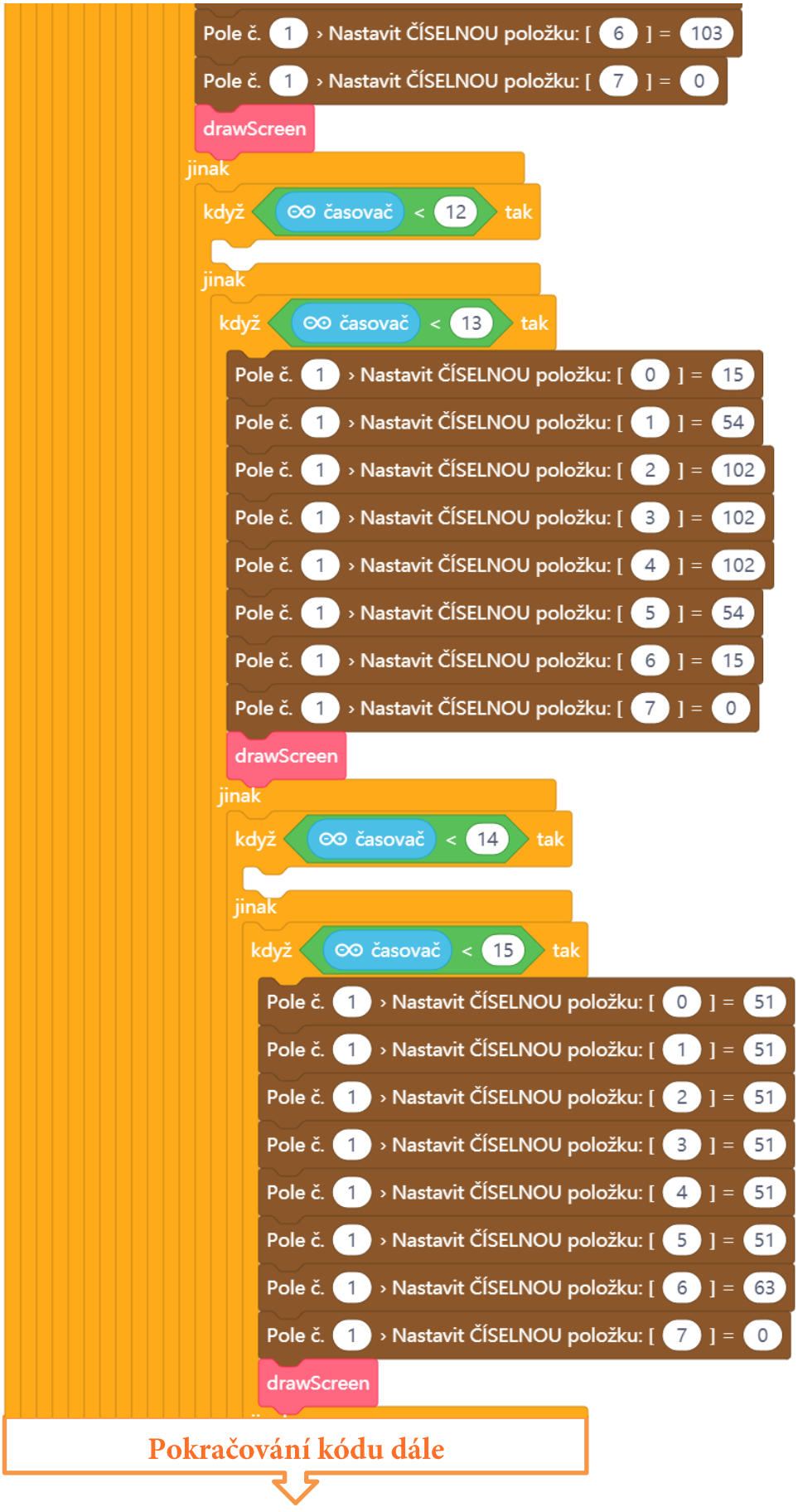


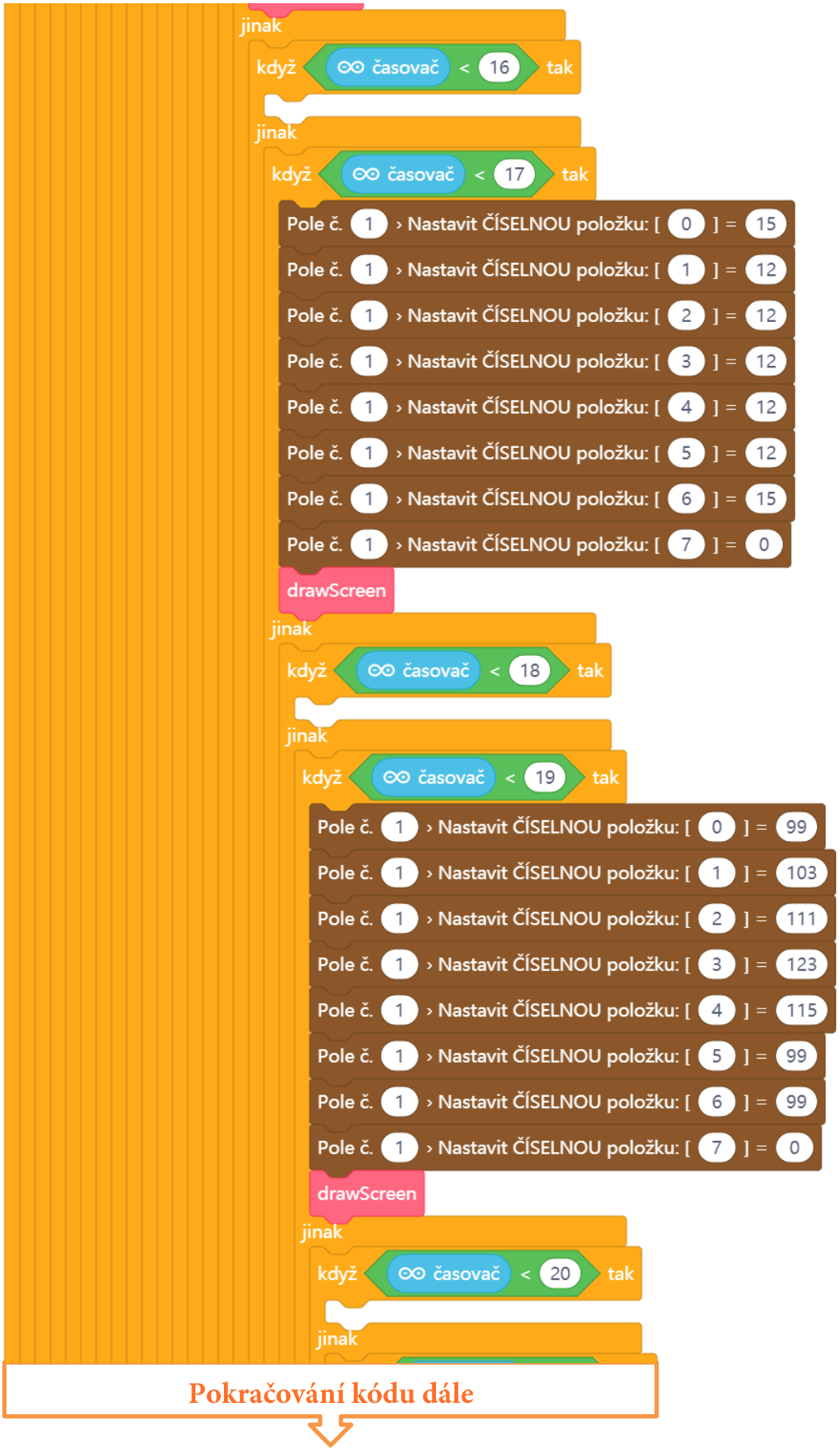


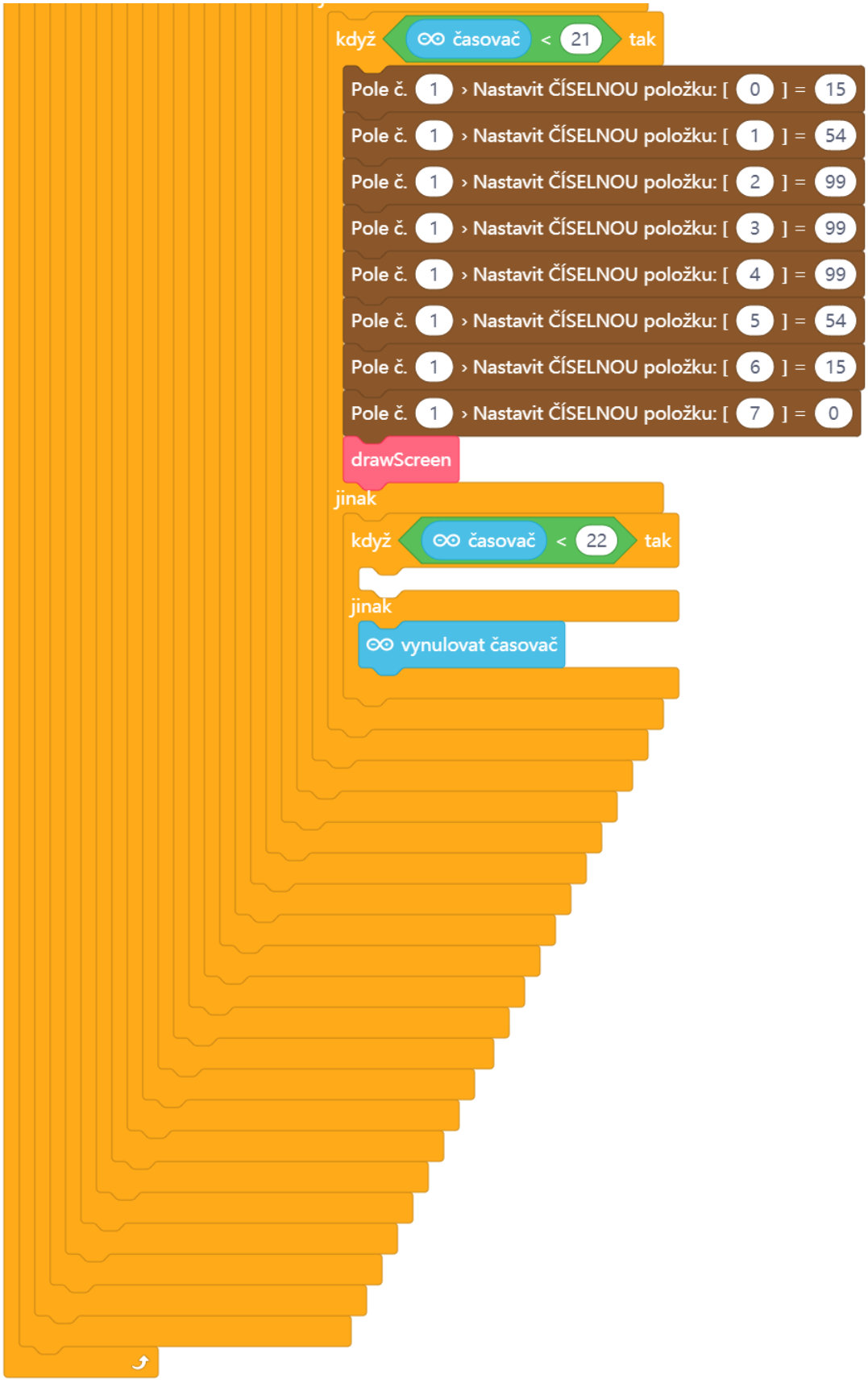
Pokračování kódu dále


Pokračování kódu dále


Pole č. 1 › Nastavit ČÍSELNOU položku: [ 2 ] = 54
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 3 ] = 15
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 4 ] = 54
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 5 ] = 102
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 6 ] = 103
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 7 ] = 0
drawScreen
jinak
když časovač < 8 tak
jinak
když časovač < 9 tak
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 0 ] = 0
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 1 ] = 102
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 2 ] = 255
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 3 ] = 255
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 4 ] = 126
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 5 ] = 60
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 6 ] = 24
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 7 ] = 0
drawScreen
jinak
když časovač < 10 tak
jinak
když časovač < 11 tak
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 0 ] = 63
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 1 ] = 102
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 2 ] = 102
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 3 ] = 62
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 4 ] = 54
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 5 ] = 102
Pokračování kódu dále


Pole č. 1 › Nastavit ČÍSELNOU položku: [ 6 ] = 103
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 7 ] = 0
drawScreen
jinak
když časovač < 12 tak
jinak
když časovač < 13 tak
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 0 ] = 15
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 1 ] = 54
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 2 ] = 102
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 3 ] = 102
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 4 ] = 102
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 5 ] = 54
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 6 ] = 15
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 7 ] = 0
drawScreen
jinak
když časovač < 14 tak
jinak
když časovač < 15 tak
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 0 ] = 51
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 1 ] = 51
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 2 ] = 51
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 3 ] = 51
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 4 ] = 51
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 5 ] = 51
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 6 ] = 63
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 7 ] = 0
drawScreen
Pokračování kódu dále


jinak
když časovač < 16 tak
jinak
když časovač < 17 tak
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 0 ] = 15
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 1 ] = 12
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 2 ] = 12
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 3 ] = 12
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 4 ] = 12
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 5 ] = 12
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 6 ] = 15
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 7 ] = 0
drawScreen
jinak
když časovač < 18 tak
jinak
když časovač < 19 tak
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 0 ] = 99
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 1 ] = 103
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 2 ] = 111
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 3 ] = 123
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 4 ] = 115
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 5 ] = 99
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 6 ] = 99
Pole č. 1 › Nastavit ČÍSELNOU položku: [ 7 ] = 0
drawScreen
jinak
když časovač < 20 tak
jinak
Pokračování kódu dále




Celý program včetně podprogramu lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-21/21-8x8-LED-matice.mblock.

### Vysvětlení kódu

Nejdříve si vysvětlíme funkci hlavního programu, kde se odkážeme na pomocné podprogramy, u kterých si řekneme, co dělají, ale jejich kód si podrobněji rozebereme později. Určitou „novinkou“ tohoto programu jsou hnědé programové bloky, které nám poskytlo rozšíření MAXI Starter kit. Jedná se o proměnné typu pole. Pod jedním názvem se vlastně skrývá několik proměnných, které se vzájemně jen odlišují svým indexem. Na samém začátku musíme proměnné typu pole definovat (tzv. deklarovat). Je třeba nastavit typ ukládaných hodnot (lze nastavit číslo nebo text), dále je pak třeba zadat rozsah (počet položek pole). A jelikož v programu takových polí můžeme použít několik, je třeba pro pole zvolit identifikační číslo. V tomto programu bude pole jen jedno, tak bude mít číslo 1. Každá položka našeho takto deklarovaného číselného pole bude odpovídat jedné řádce, která se zobrazí na LED matnici. Ukažme si na následující tabulce, jak tyto hodnoty získáme pomocí dvojkové soustavy.

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | výpočet | SOUČET |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 0+0+0+0+0+0+0+0 | **0** |
| | ■ | ■ | | | ■ | ■ | | 0+64+32+0+0+4+2+0 | **102** |
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | 128+64+32+16+8+4+2+1 | **255** |
| ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | 128+64+32+16+8+4+2+1 | **255** |
| | ■ | ■ | ■ | ■ | ■ | ■ | | 0+64+32+16+8+4+2+0 | **126** |
| | | ■ | ■ | ■ | ■ | | | 0+0+32+16+8+4+0+0 | **60** |
| | | | ■ | ■ | | | | 0+0+0+16+8+0+0+0 | **24** |
| | | | | | | | | 0+0+0+0+0+0+0+0 | **0** |

V části 8×8 jsme si vykreslili znak, který budeme chtít zobrazit – třeba srdíčko. Každé políčko této části tabulky odpovídá jedné LED ze zobrazovací matice. V daném řádku tabulky dané LED odpovídá bit, který bude určovat, zda má být daná LED rozsvícena (*bit bude mít hodnotu vysoké úrovně HIGH*), nebo zda má být zhasnuta (*bit bude nízké úrovně LOW*). Osmice těchto řádkových bitů tvoří jeden byte, který může nabývat hodnoty 0–255. Hodnoty pro rozsvícení zadané kombinace získáme tak, že bity nízké úrovně nepočítáme (mají pro nás hodnotu 0) a bity vysoké úrovně budeme sčítat dle jejich „váhy“ je uvedena nahoře v tabulce). Například v druhém řádku mají být rozsvíceny LED určené bity váhy 64, 32, 4 a 2. Celková hodnota součtu je 102. Podle této hodnoty pak budou nastaveny digitální výstupy modulu Arduino ve chvíli, kdy budeme potřebovat rozsvítit druhý řádek (viz princip ovládání LED matice). Jakmile bude osmice proměnných pole nastavena na zadané hodnoty odpovídající znaku určenému k vykreslení, je zavolán podprogram „drawScreen“, který se postará o vykreslení obrázku na LED matici.

Aby se na LED matici postupně zobrazovaly znaky se vteřinovými intervaly a se vteřinovými prodlevami, opět se pracuje s interním časovačem modulu Arduino. Na začátku je hodnota časovače vynulována, pak následuje kaskáda asi dvaceti dvou vzájemně vnořených podmínek. Každá z podmínek testuje hodnotu časovače. Jakmile hodnota časovače odpovídá zadanému času 1, 3, 5, …, 21 je zobrazen zadaný znak. Pokud

je hodnota 2, 4, 6, …, 22 je oblast kódu dané podmínky prázdná, to znamená, že se nic zobrazovat nebude – LED matice bude na tuto vteřinu zhasnutá.

Protože je třeba LED matici neustále řádkově rozsvěcovat, je nutné, aby se znaky rozsvěcovaly během celé vteřiny. Z tohoto důvodu nejsou podmínky vytvořeny jako rovnost konkrétního času, ale jako nerovnost, tedy je znak vždy vykreslován, dokud časovač nepřekročí následující hodnotu pro zobrazení dalšího znaku. Při proběhnutí celého nápisu 23 vteřin, dojde k vynulování hodnoty časovače (poslední podmínka) a díky nekonečné smyčce „opakuj stále“ hlavního programu se vše opakuje znova od začátku.

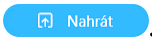
Zobrazení zadaného symbolu na LED matici zajišťuje podprogram „drawScreen“, který se stará o řádkové zobrazení. Vždy vezme výstupní hodnotu pro první, druhý, …, osmý řádek, která je uložená v proměnné pole – první položka pro 1. řádek, druhá pro 2. … atd. Samotné nastavení digitálních výstupů pro daný řádek vykonává podprogram „setColumns“, který je pro každý řádek volán podprogramem „drawScreen“.

Úkolem podprogramu „setColumns“ je rozložení vstupního čísla (předané vstupním parametrem `b`) na jednotlivé bity a dle jejich hodnoty (HIGH/LOW) nastavení digitálních výstupů modulu daných LED. Pro získávání hodnot jednotlivých bitů čísla se používá jednoduchý trik. Zadané číslo celočíselně děleno dvěma a je testován zbytek po tomto dělení. Pokud je zbytek 1, má být LED rozsvícena, pokud je zbytek 0, bude zhasnuta. Ukážeme si to na dříve uvedené hodnotě 102, která odpovídá horní části srdíčka (viz tabulka dříve).

Číslo 102 celočíselně vydělíme 2, získáme 51. Zbytek po tomto dělení je 0, tedy první LED zprava svítit nebude. Po dalším celočíselném vydělení získáme 25 se zbytkem 1 – druhá LED zprava bude svítit. Po dělení 25 je zbytek 1 – třetí LED svítí. Naopak další dělení mají zbytky 0, 0, a 1. Výsledná binární podoba čísla 102 bude `01100110`. Jedničky odpovídají rozsvícení LED. A teď se podívejme do tabulky dříve, jaká konfigurace LED tam odpovídá hodnotě 102.

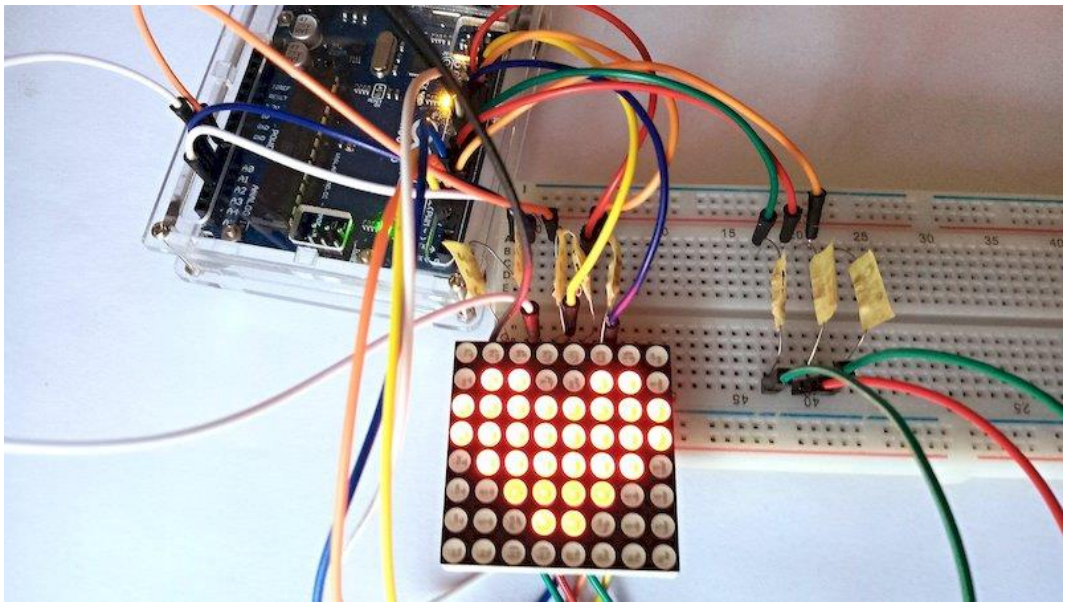
Když se podíváme na kód podprogramu „setColumns“, tak přesně dělá totéž, co jsme si zde podepsali. Proměnná `b` je postupně osmkrát celočíselně vydělena dvěma a podmínkami je testován zbytek po tomto dělení. Dle zbytku jsou nastavovány digitální výstupu modulu Arduino. Je třeba jen zdůraznit, že první zbytek po dělení dvěma říká, zda má být rozsvícena první LED bráno **zprava doleva**.

*Při pohledu zejména na část hlavního programu se určitě opět otevírá možnost nějaké optimalizace kódu nebo dokonce celkového algoritmu. Ale to je již prostor pro samostatné zkoumání.*

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

.

**Krok 4:** Po spuštění programu bychom měli vidět na LED matici postupné rozsvěcení (po jednotlivých znacích a se vteřinovými prodlevami) nápisu „LASK♥RDUINO“.



# Lekce 22 – Řízení LED zobrazovače pomocí 74HC595

## Úvod

Opět se vrátíme k sedmisegmentovému LED zobrazovači, se kterým jsme pracovali jedné z minulých lekcí, kde jsme si ukázali, jak tento zobrazovač řídit pomocí modulu Arduino – zatím „holým“ způsobem bez jakýchkoliv speciálních obvodů. Viděli jsme, že modulem Arduino musíme řídit osm zobrazovacích LED (*sedm pro zobrazení znaku, jednu pro zobrazení desetinné tečky*), což obnášelo obsazení osmi výstupních digitálních výstupů. Množství vstupních/výstupních pinů je ale modulu Arduino jen omezené množství. Nebylo by možné ovládat sedmisegmentový LED zobrazovač s využitím výrazně nižšího počtu propojovacích vodičů? Ano!

Další možností, jak řídit LED zobrazovače, je použití tzv. posuvných registrů, které pracují pomocí sériového přenosu dat. V tomto experimentu si právě takové použití ukážeme. Pro řízení LED zobrazovače použijeme integrovaný obvod 74HC595 s osmibitovým posuvným registrem. Obvod 74HC595 se skládá ze tří částí: posuvný registr, paměťový registr a třístavové výstupy. Obvod je určen pro převod dat ze sériového vstupu (v nejjednodušší variantě jeden datový a jeden hodinový signál) na osm paralelních výstupů. Takže právě tím ušetříme vstupně-výstupní piny modulu Arduino. Z tohoto důvodu je obvod 74HC595 velmi často a široce používán k řízení segmentových displejů. Jeho výhodou je i možnost zapojování několika těchto obvodů do kaskády, takže lze po sériovém rozhraní „nasypat“ poměrně velké množství dat pro paralelní výstupy.

## Použité komponenty

- modul Arduino
- USB kabel
- posuvný registr 74HC595
- sedmisegmentový LED zobrazovač
- 8× rezistor (220 Ω)
- nepájivé pole
- vodiče pro nepájivé pole

## Princip

Pojďme se podrobněji podívat, co tedy posuvný registr dělá? V zásadě je to soustava klopných obvodů, kterými se logická informace přicházející sériově posouvá pomocí hodinového impulsu dále po paralelních výstupech (Q0–Q7). Informace jednoho bitu (vysoká nebo nízká úroveň) na sériovém vstupu je při prvním tiku hodinového signálu „vtažena“ do registru a zapsána na výstup Q0. Při dalším tiku hodinového signálu se bit z výstupu Q0 posune na výstup Q1 a na uvolněné místo výstupu Q0 je načten bit ze sériového výstupu. Po uplynutí osmi hodinových pulzů tak máme v registru načteno osm bitů, které mohou být prezentovány na osmi paralelních výstupech. Jelikož paralelní výstupy posuvného registru nabývají nízké a vysoké úrovně, stejně jako by nabývaly výstupy modulu Arduino, připojený segmentový LED zobrazovač se chová stejně, jako kdyby byl připojen přímo k modulu Arduino.

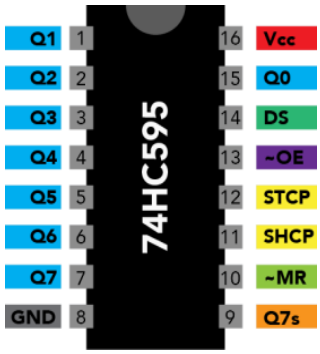
Posuvný registr 74HC595, se kterým budeme dále pracovat, bude pro své ovládání potřebovat tři signály, kterými jej budeme ovládat. Prvním bude datový signál se sériovými vstupními daty, druhým pak musí pochopitelně být

synchronizační hodinový signál. Třetí signál bude přesouvat načtená sériová data do paměťového registru paralelních výstupů. Jestliže při plnění posuvného registru se data postupně posouvají přes jednotlivé bity, bylo by nevhodné, aby se jejich stav okamžitě projevoval na výstupu. Z tohoto důvodu neobsahuje obvod 74HC595 jen posuvný registr, ale i paměťový registr. Až teprve zapsáním dat do paměťového registru se tyto data mohou začít projevovat na výstupech. Výhodou je i to, že zatímco paměťový registr drží platná data na výstupu, může se posuvný registr sériově plnit novou osmicí dat, která je pak okamžitě a najednou zapsána na výstup.

Tak je možné pomocí tří pinů modulu Arduino zaslat do obvodu 74HC595 osm bitů data určující osm stavů pro jednotlivé segmenty zobrazovače. Obvod 74HC595 tedy zapojíme mezi modul Arduino a LED zobrazovač.

Následující obrázek a tabulka ukazuje rozmístění a popis jednotlivých vývodů obvodu 74HC595:

| **piny 74HC595** | **Popis funkce vývodu** |
|---|---|
| Q0–Q7 | paralelní bitové výstupy, které jsou schopné řídit jednotlivé segmenty LED zobrazovače |
| Q7’ (Serial data output) | pin, který slouží k propojení s dalším obvodem 74HCT595 |
| MR (Master Reclear) | pin vynulování výstupů – aktivní při nízké úrovni (log. 0); zapojíme ho na vysokou úroveň (+5 V) |
| SHcp (Shift register clock input) | vstup určený pro řídicí hodinový signál |
| STcp (Storage register clock input) | při na náběžné hraně na tomto vstupu se údaje v posuvném registru přesunou do paměťového registru |
| OE (Output enable) | pin „zapnutí“ výstupů čipu – aktivní nízké úrovni (log. 0); obvod chceme aktivovat, zapojíme ho tedy na GND |
| DS (Serial data input pin) | vstup dat, kam se zapisují vstupní data, která budou s pomocí hodinového signálu „vtažena“ do posuvného registru |
| Vcc | napájecí napětí (+5 V); připojíme na pin 5V modulu Arduino. |
| GND | vývod zemnění (GND); propojíme s pinem GND modulu Arduino. |



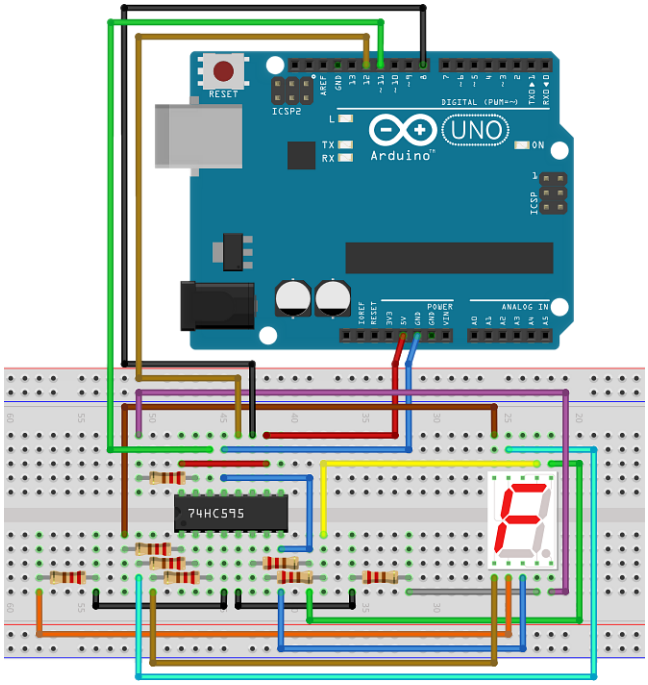

Přestože samotný princip posuvného registru možná vypadá jednoduše, samotné řešení z hlediska signálů na řídicích pinech vyžaduje přesné ovládání. Určitým úskalím je nutnost „rozkrájení“ výstupních dat na jednotlivé

bity. Naštěstí tento proces již někdo vyřešil za nás a můžeme využít rozšíření MAXI Starter kit, které nám rovnou přidá programové bloky, které jsou speciálně pro tuto lekci připravené.

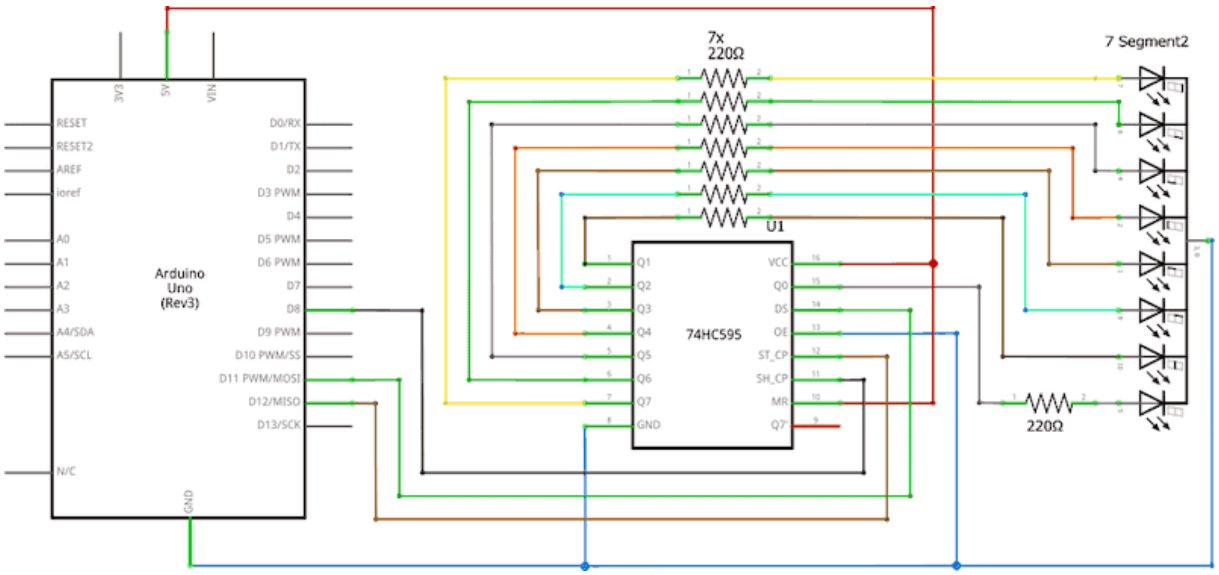
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Zde uvedený obvod, stejně jako následný program je určen jen pro LED zobrazovač se společnou katodou. Pokud jsme však pochopili elektronický rozdíl mezi typem zobrazovače CC a CA (stejně tak mezi způsoby jejich ovládání), měli bychom již dokázat schéma a program upravit i pro typ se společnou anodou.

### Blokové schéma




### Elektronické schéma

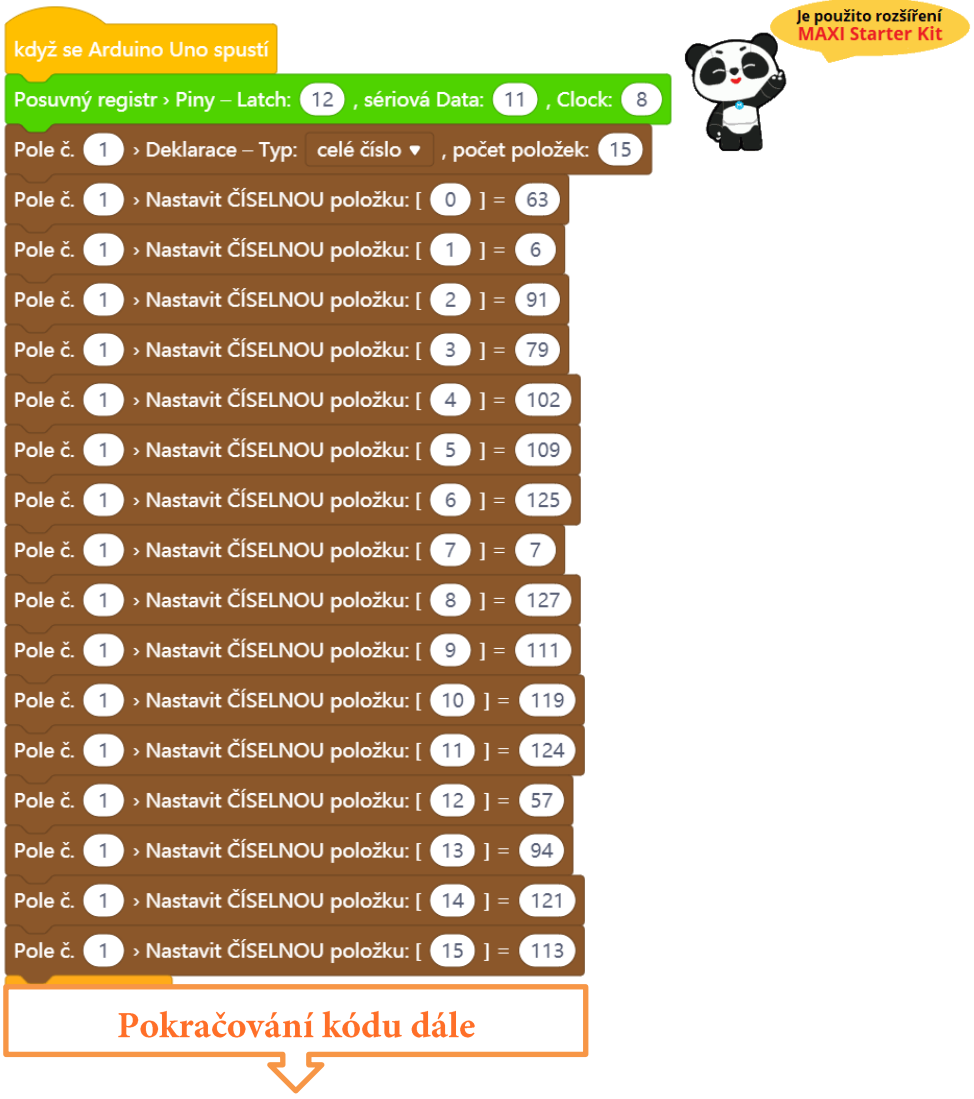


Následující tabulky znázorňují propojení modulu Arduino, obvodu 74HC595 a LED zobrazovače.

| obvod 74HC595 | modul Arduino |
|---|---|
| DS (14) | 11 |
| STcp (12) | 12 |
| SHcp (8) | 8 |
| OE (13) | GND |
| MR (10) | 5V |
| $V_{CC}$ (16) | 5V |

| obvod 74HC595 | LED zobrazovač |
|---|---|
| Q7 (7) | a |
| Q6 (6) | b |
| Q5 (5) | c |
| Q4 (4) | d |
| Q3 (3) | e |
| Q2 (2) | f |
| Q1 (1) | g |
| Q0 (15) | dp |
| GND (8) | GND |

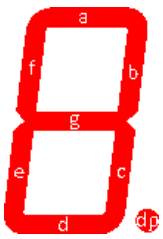
**Krok 2:** V prostředí mBlock vytvoříme následující program.




Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-22/22-Rizeni-LED-zobrazovace-pomoci-74HC595.mblock.

## Vysvětlení kódu

I v tomto programu využijeme pole proměnných a obdobně do nich zapíšeme hodnoty, které budeme chtít zobrazit na připojeném LED zobrazovači. Tentokráte pole proměnných využijeme tak, jak by se mělo využívat a jaká je vlastně jeho výhoda. A to tak, že budeme jeho jednotlivé položky volat indexem – to uvidíme v hlavní nekonečné smyčce „opakuj stále“. První část kódu, která proběhne jen jednou je deklarace pole proměnných a naplnění jednotlivých jeho položek. Každé číslo uvedené v dané položce odpovídá rozsvícení daných LED segmentů zobrazovače. Ukážeme si to na první hodnotě 63, která by měla zobrazit číslici 0.



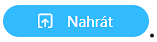

| segment | stav | váha bitu | hodnota |
|---|---|---|---|
| a | 1 *(svítí)* | 1 | 1 |
| b | 1 *(svítí)* | 2 | 2 |
| c | 1 *(svítí)* | 4 | 4 |
| d | 1 *(svítí)* | 8 | 8 |
| e | 1 *(svítí)* | 16 | 16 |
| f | 1 *(svítí)* | 32 | 32 |
| g | 0 *(nesvítí)* | 64 | 0 |
| dp | 0 *(nesvítí)* | 128 | 0 |
| | | **SOUČET:** | **63** |

Rozsvícení LED segmentů opět udávají jednotlivé bity binární podoby dané hodnoty. V naší tabulce odpovídá segment `a` bitu s nejnižší váhou (tzv. LSB), naopak desetinná tečka odpovídá bitu s nejvyšší váhou (tzv. MSB). Hodnotu, kterou budeme odesílat na výstup, opět udává součet vah bitů, které odpovídají rozsvícenému segmentům. Zhasnuté segmenty (zde `g` a `dp`) se nezapočítávají. Po sečtení všech příspěvků získáme hodnotu 63, což je přesně hodnota uložená v nulté položce proměnné typu pole.

Víme-li, že k položkám pole můžeme přistupovat pomocí jejich indexu – např. k hodnotě, která má zobrazit číslici 0 přistupujeme s indexem 0 – můžeme toho využít. V hlavní části programu máme cyklus, který

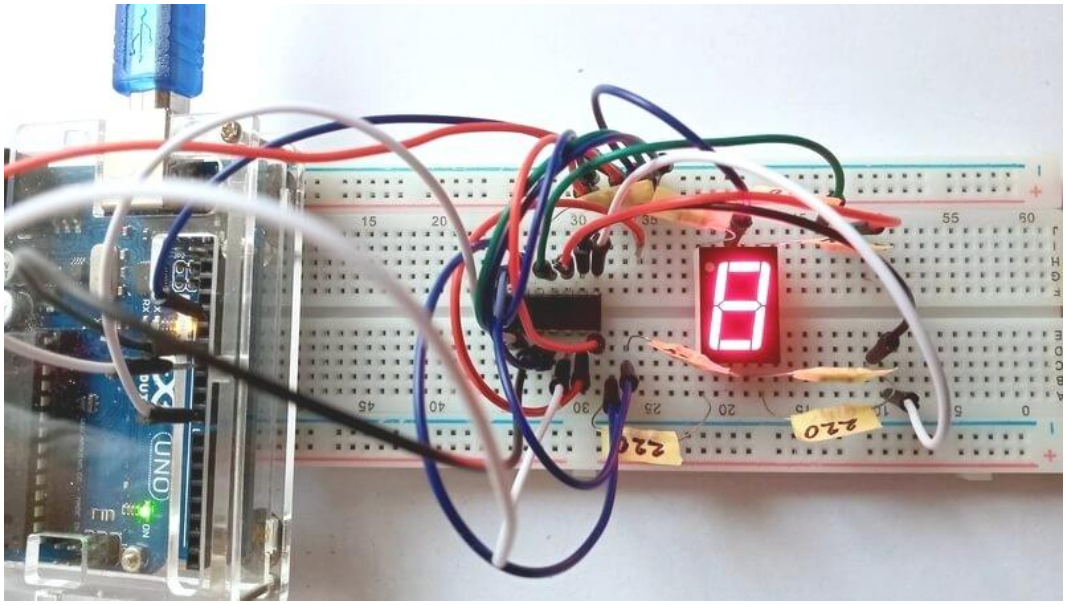
proběhne šestnáctkrát a proměnnou `num`, která díky tomu bude postupně nabývat hodnot 0–15. Tuto proměnnou použijeme jako index proměnné pole a tím postupně můžeme získat předem nadefinované hodnoty pro zobrazení jednotlivých 16 znaků. Zde je dobré si všimnou trochu podivné konstrukce v programovém bloku, kde je proměnná `num` použita jako index pole. Prostředí mBlock standardně používá proměnné jako reálné čísla, zatímco indexy pole musí být celá čísla. I když je hodnota proměnné `num` třeba rovna 1 (*což je celé číslo*), stále ji prostředí mBlock považuje za reálné číslo. Takže si budeme pamatovat, že než použijeme v prostředí mBlock některou proměnnou jako index pole, musíme ji pomocí převodního bloku převést na celé číslo. Jinak přeložení a zápis programu skončí chybou. Díky tedy jednomu cyklu můžeme do posuvného registru postupně „nasypat" všech 16 hodnot. To výrazně zkracuje zápis celého programu.

Na závěr vysvětlení programového kódu je třeba ještě upozornit na jednu věc, která souvisí s obsluhou a vlastně i principem posuvného registru. Když se podíváme na elektronické propojení LED segmentů a digitálních výstupů posuvného registru, vidíme, že segment `a` je zapojen na bitu nejvyšším váhy (Q7) a desetinná tečka `dp` na nejnižším (Q0). To je přesně obráceně, než jsme si ukazovali při odvození hodnoty pro zobrazení! Vysvětlení najdeme na zeleném programovém bloku obsluhy posuvného registru. Konkrétně nás zajímá volba „Pořadí bitů", kde lze nastavit hodnotu MSBFIRST nebo LSBFIRST. Tyto hodnoty určují, jakým způsobem budou jednotlivé bity odesílány na posuvný registr. V našem programu je nastavena hodnota LSBFIRST, což znamená, že jako první má být odeslán bit nejnižší váhy. Při našem vytváření hodnoty pro odeslání to odpovídá bitu odpovídajícímu segmentu `a`. Jenže, jak jsme si říkali, při zápisu dalšího bitu na posuvný registr se první bit posouvá na druhé místo atd. Tedy první zapsaný bit (bit segmentu `a`) se posune až na nejvyšší místo. Proto musí v této konfiguraci být segment elektronicky připojen na nejvyšším bitu Q7. Pochopitelně by bylo možné zvolit i pořadí odesílání MSBFIRST, pak by byl nejdříve odesílán bit odpovídající desetinné čárce, a tedy bit segmentu a by řídil výstup Q0 s nejnižší váhou. Tož by pak ale vyžadovalo změnit pořadí připojených LED segmentů. Ale opět se experimentům meze nekladou.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

.

**Krok 4:** Výsledkem našeho zapojení a programu by mělo být postupné zobrazování číslic 0–9 a pak i znaků A–F na připojeném LED zobrazovači.


UNO
220

# Využití
# $I^2C$ displeje

# Lekce 23 – Používaní skeneru sběrnice I²C

## Úvod

Jak jsme popsali v části věnované I²C komunikaci, musí mít každé zařízení připojené ke sběrnici s tímto způsobem komunikace, určenou jednoznačnou adresu. Bohužel je to jedna z oblastí, kde občas dokumentace k některým čidlům chybí. Abychom tuto adresu mohli u jednotlivých periférií zjistit nebo jen ověřit, ukážeme si, jak naskenovat a zobrazit adresy všech zařízení připojených na sběrnici I²C.

## Použité komponenty

- modul Arduino
- USB kabel
- libovolné I²C zařízení *(např. LCD displej s I²C řadičem)*
- vodiče typu „samec-samice“

## Princip

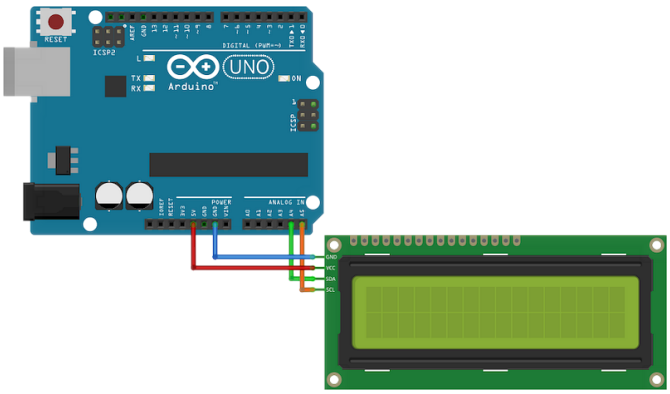
Jak bylo řečeno, každé zařízení s I²C sběrnicí má adresu I²C, kterou používá k přijímání příkazů nebo odesílání zpráv. Pojďme si ukázat, jak najít adresu na příkladu I²C převodníku pro LCD displej. Princip hledání je poměrně jednoduchý, ale zato účinný. Postupně jsou na sběrnici I²C oslovovány adresy z daného rozsahu. Pokud je na sběrnici přítomna nějaká periférie dané adresy, odpoví. Pokud volání skončí chybou, periférie dané adresy na sběrnici není.

Dále vytvořený program bude zobrazovat 7-bitové adresy nalezených zařízení v hexadecimální podobě. Každé nalezené zařízení na sběrnici I²C bude vypsáno na sériový monitor. Pro možné vyzkoušení funkčnosti programu je možné jednotlivé I²C zařízení zapojovat a odpojovat i za běhu programu.

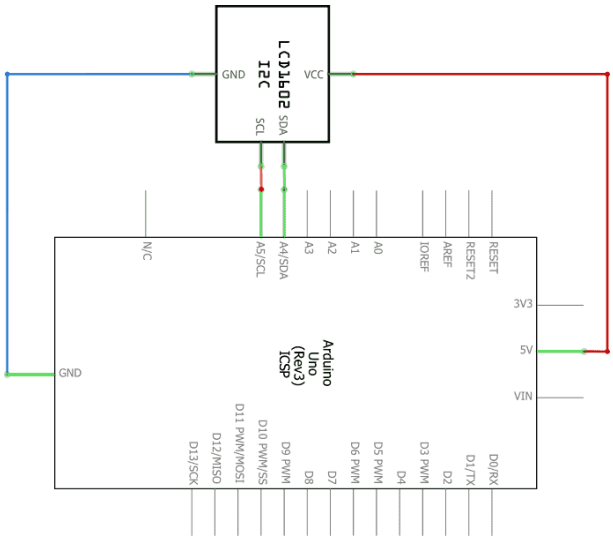
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Spojení mezi displejem I²C LCD1602 a modulem Arduino znázorňuje následující tabulka. V tomto příkladu je sice použit displej s I²C převodníkem, ale jak jsme si říkali v popisu I²C sběrnice, vývody `SDA` a `SCL` jsou podstatou komunikace na všech I²C periférií ch. A protože modul Arduino UNO využívá komunikace I²C jen na pinech A4 a A5, je i připojení na straně modulu Arduino vždy shodné s následující tabulkou.

**Blokové schéma**



**Elektronické schéma**



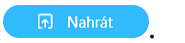

| I²C LCD1602 | modul Arduino | |
|---|---|---|
| | UNO | Mega |
| SDA | A4 | 20 |
| SCL | A5 | 21 |
| $V_{CC}$ | 5V | 5V |
| GND | GND | GND |

**Krok 2:** V prostředí mBlock vytvoříme následující program. Zatímco v předchozích programech byly programy někdy i vcelku rozsáhlé, zde díky rozšíření MAXI Starter kit jde dva programové bloky.


Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-23/23-Pouzivani-skeneru-sbernice-I2C.mblock.

### Vysvětlení kódu

V programu jsme využili programový blok, který je do prostředí mBlock přidán díky rozšíření MAXI Starter kit. Tento jediný programový blok skenuje sběrnici I²C a hledá připojená zařízení, pokud nějaké zařízení nalezeno, je jeho adresa vypsána v sériovém monitoru. To je vše! ☺

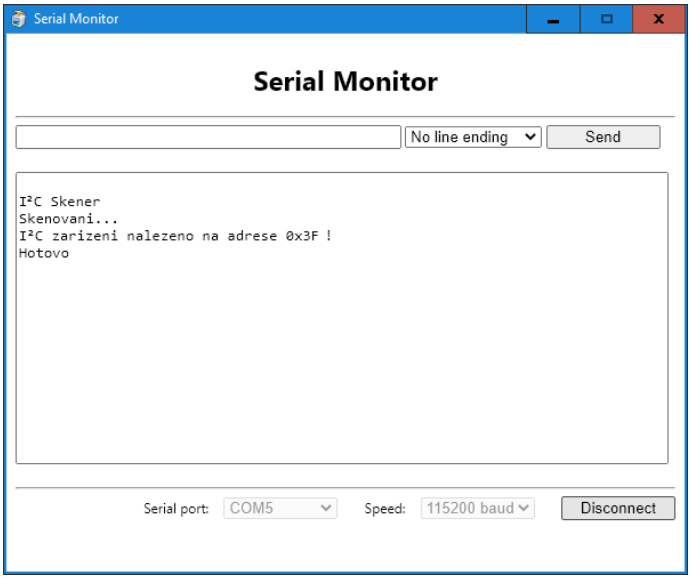
**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

.

**Krok 4:** Nyní modul Arduino odpojíme od prostředí mBlock pomocí tlačítkem 

.
**USB kabel modulu Arduino ale od počítače neodpojujeme!**

**Krok 5:** Spustíme na počítači aplikaci Serial Monitor. V Serial Monitoru vybereme port, na které je náš modul Arduino (např. COM5) připojen a nastavíme komunikační rychlost na hodnotu 115200 baudů. Na závěr se tlačítkem Connect připojíme k modulu Arduino. Objeví se okno sériového monitoru, ve kterém bychom měli vidět seznam všech zařízení připojených ke sběrnici $I^2C$ i s jejich adresou v hexadecimální podobě.

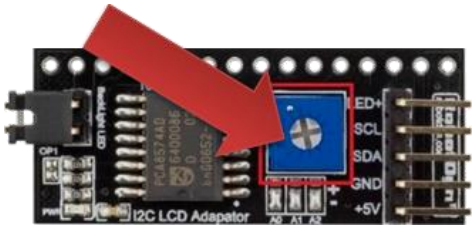


# Lekce 24 – I²C LCD displej

## Úvod

Jak už jsme viděli některých předešlých v lekcích, jakékoliv zobrazovače výrazně obohacují naší interakci s řídicím modulem. Zároveň jsme ale narazili i na jednu nevýhodu. Pokud připojíme nějaký zobrazovač ke kontroléru, může tím být obsazeno poměrně velké množství vstupně-výstupních portů, kterých je zpravidla na řídicím modulu omezený počet. Na druhu stranu jsme ale pak viděli, že lze zobrazovače kombinovat s pomocnými obvody a náročnost na počet řídicích portů se tím může výrazně zredukovat. V případě ovládání našeho LCD displeje je potřeba řídicích pinů hned 16. A právě důvodů snížení počtu ovládacích vodičů a pinů byl vyvinut převodník se I²C sběrnicí, který použijeme pro řízení LCD displeje a který daný problém vyřeší.

Přední část displeje
(nahoře vidíme 16 ovládacích pinů)




I²C převodník, který lze připojit na řídicí piny, tak je zpravidla rovnou připájen na rubové straně displeje

Sada Arduino MAXI Starter kit obsahuje LCD displej, ke kterému je na zadní straně připojený I²C převodník, takže lze displej řídit pomocí jen dvou datových vodičů. Modrý potenciometr (viz obrázek) na převodníky se používá k nastavení podsvícení.

## Použité komponenty

- modul Arduino
- USB kabel
- LCD displej s I²C převodníkem
- vodiče typu „samec-samice“

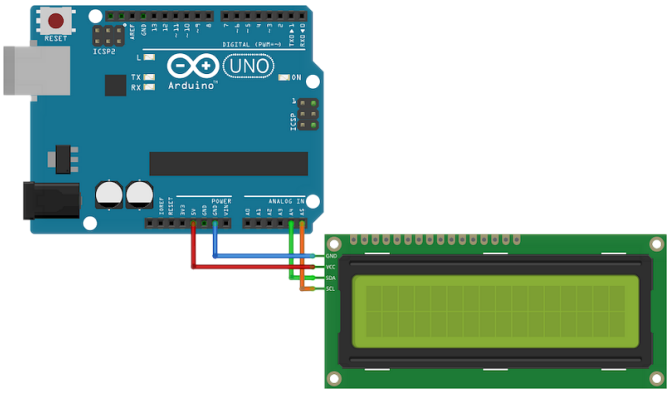
## Princip

V této lekci si zkusíme základní ovládání LCD displeje. Řízení nám značným způsobem ulehčí programové bloky, které do prostředí mBlock opět přidá rozšíření MAXI Starter kit. Pro začátek v tomto experimentu necháme na displeji zobrazit následující texty: „Laskarduino“ a „Zdravim bastlire!“. Použitá LCD displej LCD1602 je doplněný I²C převodníkem. U displeje nebudeme řídit jednotlivé pixely, ale odešleme na něj text, jehož zobrazení zařídí podpůrné obvody. Displej pro nás bude tedy prostorem, kde budeme mít možnost zobrazit dva řádky textu po šestnácti znacích (odtud i označení displeje LCD1602).

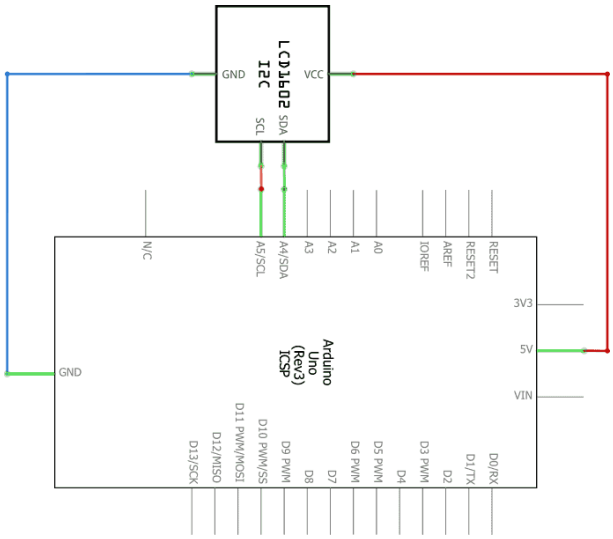
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Spojení mezi displejem I²C LCD1602 a modulem Arduino znázorňuje následující tabulka.

**Blokové schéma**



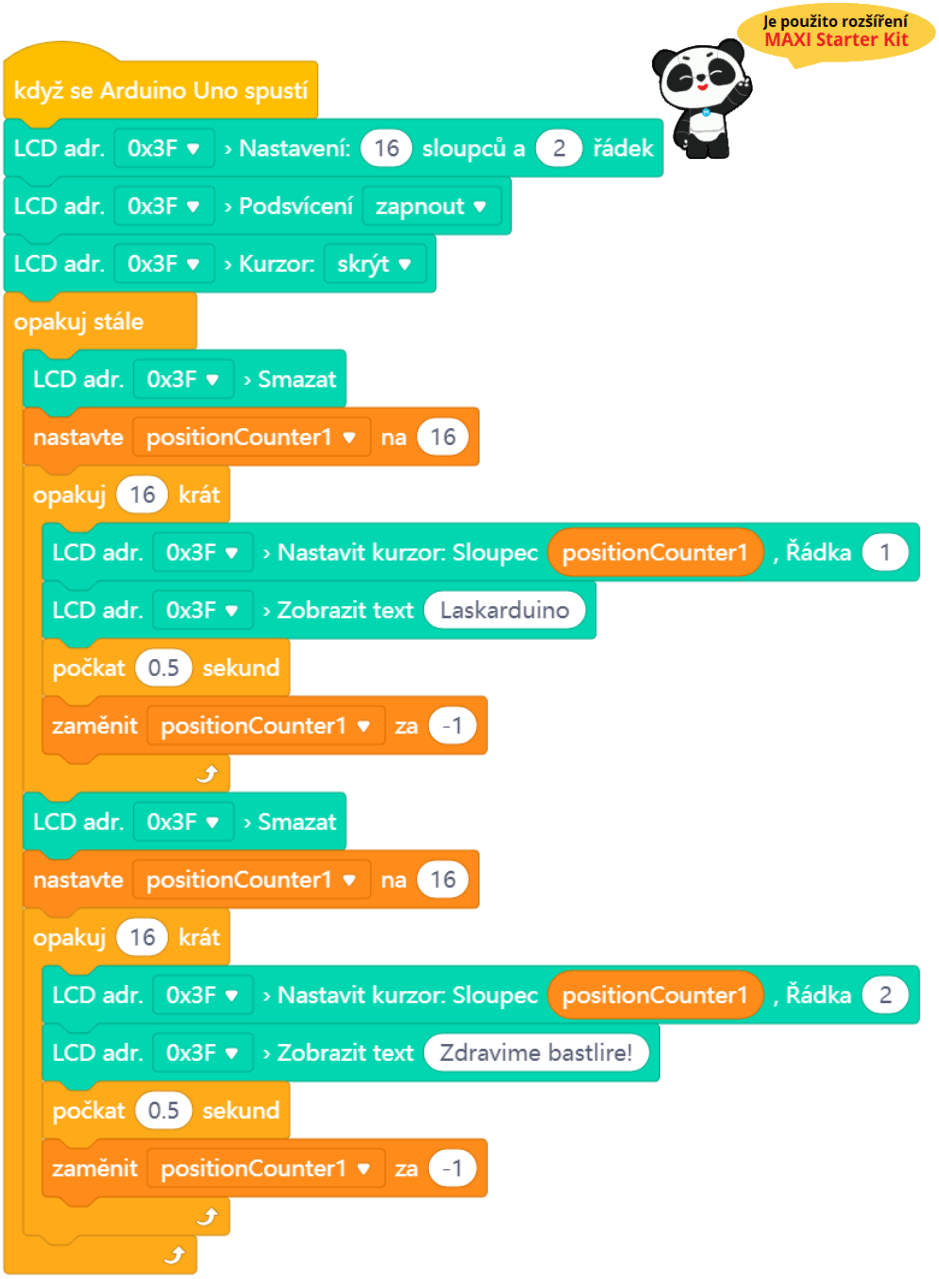
**Elektronické schéma**




| I²C LCD1602 | modul Arduino | |
|---|---|---|
| | UNO | Mega |
| SDA | A4 | 20 |
| SCL | A5 | 21 |
| $V_{CC}$ | 5V | 5V |
| GND | GND | GND |

**POZNÁMKA:**
Propojení I²C LCD1602 a modulu Arduino bude stejné ve všech následujících lekcích, kde bude displej použit.

**Krok 2:** V prostředí mBlock vytvoříme následující program. V programu jsou použity příkazové bloky rozšíření MAXI Starter kit pro obsluhu I²C LCD (modrozelené bloky), které mají jako jeden ze svých vstupních parametrů adresu použitého I²C převodníku pro displej. Každý příkaz provedený na I²C periférii musí být vždy řádně adresován. Proto je potřeba při tvorbě programu znát a přesně zadat adresu použitého I²C LCD displeje. Obvykle bývá adresa převodníku pro displej `0x27`, ale v sadě, podle které vznikal tento text, byla zrovna adresa převodníku `0x3F`. Každý uživatel musí u příkazů pro LCD nastavit aktuální adresu dle své situace, a to u všech bloků. (viz rozbalovací pole „adr.“).




Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-24/24-I2C-LCD-displej.mblock.

## Vysvětlení kódu

V úvodní části programu jsou nastaveny parametry použitého LCD displeje, tedy rozměr 16 sloupců a 2 řádky. Dále je nastaveno zapnutí podsvícení. Pro fungující LED podsvícení displeje musí být na displeji propojena zkratovací spojka (jumper). Dále je nastaven skrytý kurzor. V hlavní nekonečné smyčce „opakuj stále“ jsou za sebou zařazeny dva cykly, s pevně daným opakováním (16×). Oba cykly dělají de facto to samé. Na začátku je nastavena ho proměnné `positionCounter1` hodnota 16, která odpovídá poslednímu sloupci displeje. Pak je na tuto pozici vyprán zadaný nápis. V prvním cyklu jde o text „Laskarduino_“ *(podtržítko znázorňuje mezeru)* a je vypisován na první řádku. V druhém cyklu je zobrazován text „Zdravime

bastliře!_“ *(podtržítko opět znázorňuje mezeru)* a je text vypisován na druhou řádku. Daný text je zobrazen na půl vteřiny, pak je proměnná `positionCounter1` zmenšena o jedničku a cyklus se začíná opakovat. Text je tedy vypsán o jednu pozici vlevo, což vyvolává efekt „přijíždějícího“ textu směrem zprava. Po šestnácti krocích text dorazí až na levý okraj displeje. Mezera na konci textů je důležitá, slouží jako mazač posledních znaků posouvajícího se textu.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Nyní bychom na svém LCD displeji měli postupně vidět nápisy „Laskarduino“ a „Zdravim bastlire!“ Pokud na displeji není nic vidět, je třeba zkontrolovat nastavení odporového trimru (zadní strana I²C převodníku) pro řízení kontrastu na převodníku. Může se stát, že je nastavený na minimum a tudíž na displeji není nic vidět. Otáčením si tedy nastavíme správný kontrast, tak aby byl nápis na displeji co nejlépe čitelný.

# Lekce 25 – Voltmetr

## Úvod

Když již zvládáme načítání analogových vstupů modulu Arduino a nyní jsme si osvojili ovládání LCD displeje, můžeme si kupříkladu vytvořit digitální voltmetr s rozsahem 0–5 V. Proměnné vstupní napětí budeme simulovat připojeným potenciometrem.

## Použité komponenty

- modul Arduino
- USB kabel
- potenciometr 50 kΩ
- LCD displej s I²C převodníkem
- nepájivé pole
- vodiče pro nepájivé pole + vodiče typu „samec-samice“

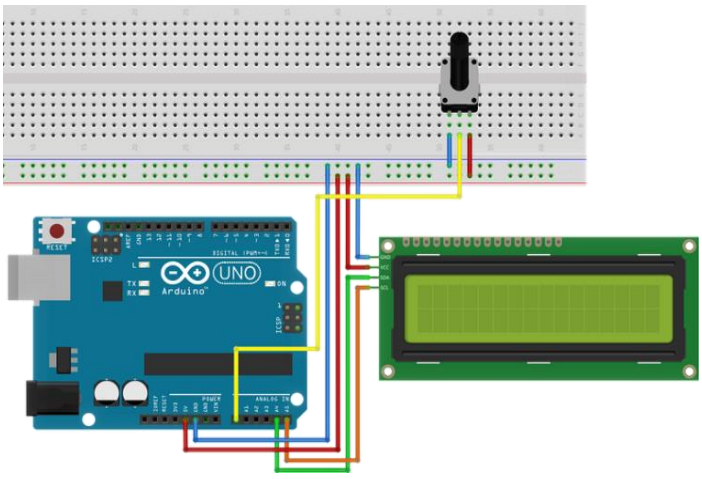
## Princip

V tomto experimentu využijeme potenciometr k dělení napětí. Výhodou modulu Arduino je, že neumí číst pouze digitální signály, ale umí načítat i analogový signál na výstupu jezdce potenciometru. Na analogových vstupech, které jsou osazeny 10-bitovými analogově-digitálním převodníkem (ADC), dokáže převést tento analogový signál na číselnou podobu. Po přepočítání vstupní hodnoty z rozsahu 0–1023 na interval 0–5 V je získaná hodnota vstupního napětí zobrazena na LCD displeji.

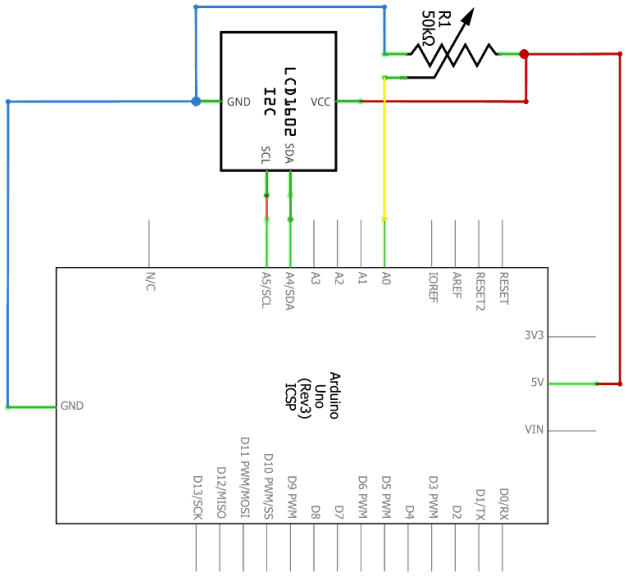
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Spojení mezi displejem I²C LCD1602 a modulem Arduino znázorňuje následující tabulka.

**Blokové schéma**

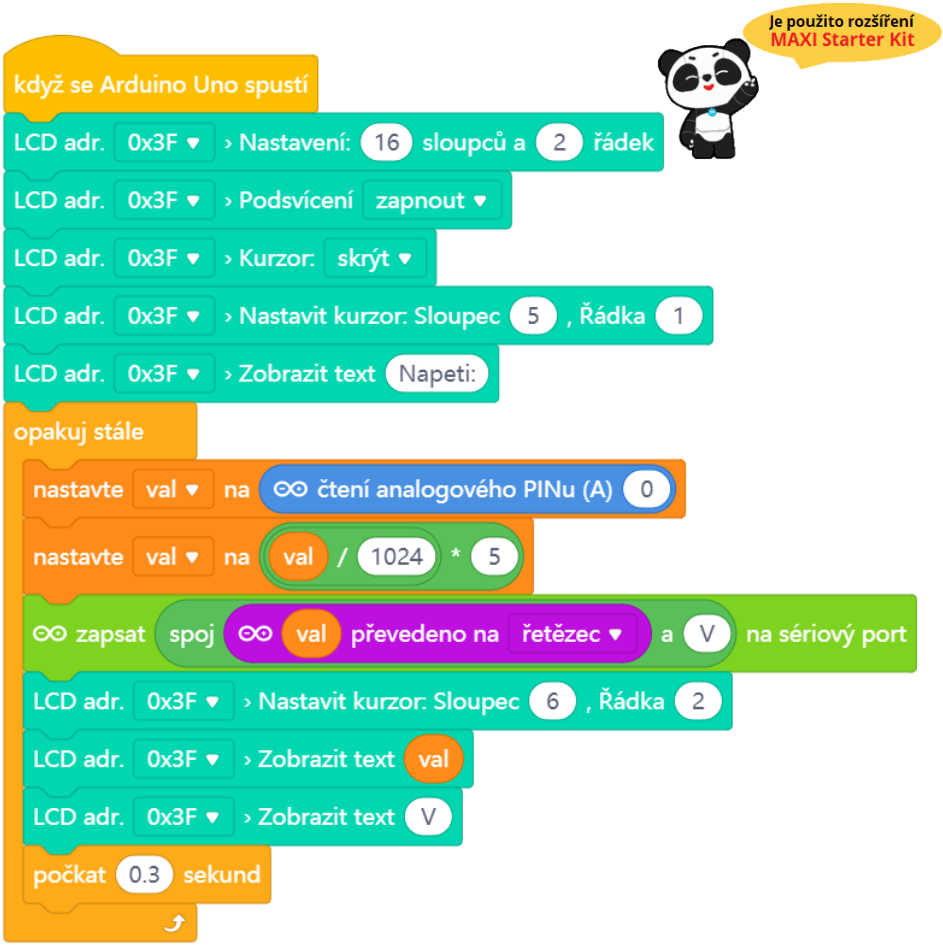


**Elektronické schéma**



| I²C LCD1602 | modul Arduino | |
|---|---|---|
| | UNO | Mega |
| SDA | A4 | 20 |
| SCL | A5 | 21 |
| $V_{CC}$ | 5V | 5V |
| GND | GND | GND |

**Krok 2:** V prostředí mBlock vytvoříme následující program. Při tvorbě programu je třeba znát adresu použitého I²C LCD displeje, neboť u každého příkazů pro LCD je třeba nastavit aktuální adresu. (viz rozbalovací pole „adr.“).




Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-25/25-Voltmetr.mblock.

### Vysvětlení kódu

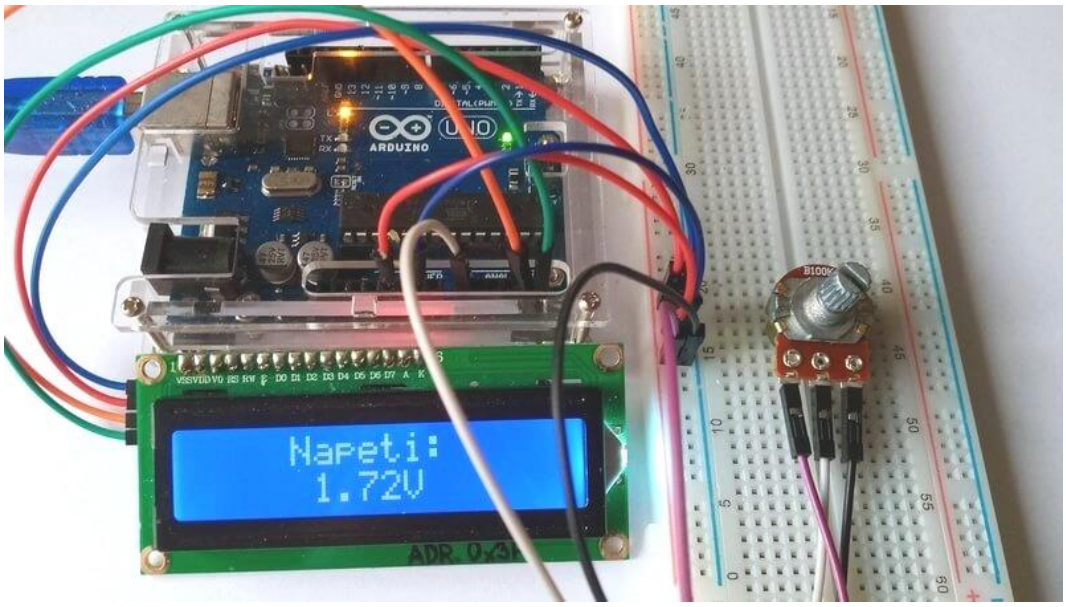
V počáteční části programu nastavíme parametry použitého LCD, jako jsou jeho rozměry (sloupce, řádky) a zapnutí podsvícení. Je třeba se zaměřit na správné nastavení adresy použitého I²C převodníku! Před spuštění

hlavní nekonečné smyčky „opakuj stále“ je ještě na střed první řádky vypsán text „Napeti:“. Hlavní nekonečná smyčka pak již pracuje jen s druhou řádkou displeje. Nejdříve je načtena analogová hodnota napětí na vstupním pinu A0, pak je přepočtena ze vstupního rozmezí 0–1023 na interval 0–5. Tím získáme hodnotu vstupního elektrického napětí. Tato hodnota je uložena do proměnné `hod`. Následuje výpis této hodnoty na druhý řádek LCD. Jelikož při konverzi reálného čísla (hodnota napětí) na text, který bude na LCD vypsán, je číslo vždy uvedeno s dvěma desetinnými místy, není třeba řešit různou délku výstupního řetězce na displeji. Displej tedy není třeba mazat, neboť hodnoty se přepisují a díky stejné délce vypsaných textů se nemůže stát, že by na daném řádku LCD zbyly znaky z předešlého delšího textu.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Při otáčení potenciometrem můžeme na LCD displeji vidět velikost naměřeného elektrického napětí na pinu `A0` modulu Arduino.

**POZNÁMKA:**
Programový kód obsahuje i zobrazení naměřeného napětí na sériový monitor (světle zelený příkazový blok). Pro sledování výstupu na sériovém portu, je třeba modul Arduino odpojit od prostředí mBlock a naopak připojit k aplikaci Serial Monitor (viz dále). Pokud výstup na sériový port nepotřebujeme, neboť se naměřené napětí zobrazuje na LCD displeji, lze daný příkazový blok zcela vynechat.




Pokud chceme výpisy na sériový port sledovat, je třeba modul Arduino odpojit od prostředí mBlock a připojit k aplikaci Serial Monitor.

**Krok 5:** Nyní modul Arduino odpojíme od prostředí mBlock pomocí tlačítkem Odpojit.
**USB kabel modulu Arduino ale od počítače neodpojujeme!**

**Krok 6:** Spustíme na počítači aplikaci Serial Monitor. V Serial Monitoru vybereme port, na které je náš modul Arduino (např. COM5) připojen a nastavíme komunikační rychlost na hodnotu 115200 baudů. Na závěr se tlačítkem Connect připojíme k modulu Arduino. Objeví se okno sériového monitoru, ve kterém budou vypisovány výstupní hodnoty naměřeného elektrického napětí.

# Lekce 26 – Senzor hladiny vody

## Úvod

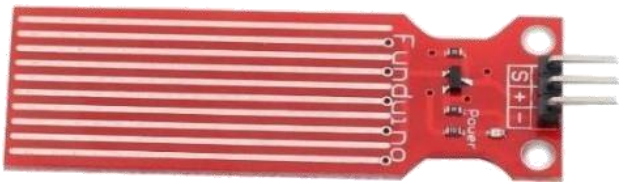
Potenciometr nemusí být jediným čidlem, které nám poskytuje výstupní napětí úměrné nějaké měřené veličině. V této lekci si ukážeme, že zcela obdobně může fungovat snímač hladiny vody, který můžeme použít buď na měření hloubky vody, nebo při znalosti rozměrů nádoby jako měřič obejmu. Naměřený výsledek opět zobrazíme na LCD displeji.

## Použité komponenty

- modul Arduino
- USB kabel
- senzor hladiny vody
- LCD displej s I²C převodníkem
- nepájivé pole
- vodiče pro nepájivé pole + vodiče typu „samec-samice“

## Princip

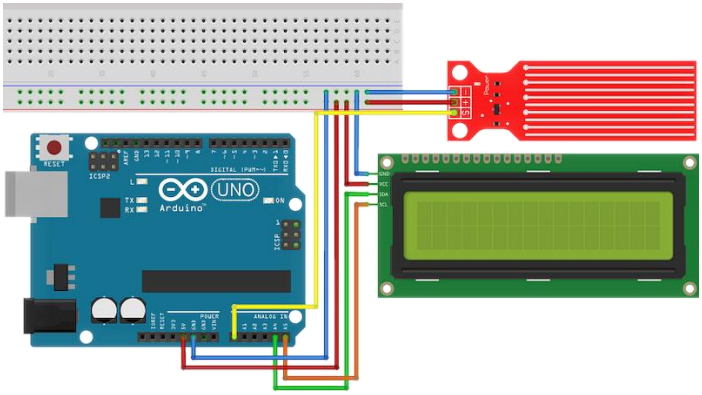
Senzor hladiny vody je modul, který ve své podstatě může snímat výšku hladiny vody, je fyzikálně řešen jako odporové čidlo doplněné zesilovačem. Základní částí jsou vodivé kontakty hřebenového tvaru vytvoření přímo na desce DPS. Při umístění do vody budou tyto kontakty měnit svůj odpor (v závislosti na hloubce ponoření) a tím převádět „hloubkový“ signál na signál elektrický. Když si toto uvědomíme, vidíme, že v principu jde vlastně o úplně stejné zapojení, jako bylo zapojení s potenciometrem. I zpracování získaného elektrického signálu bude stejné. Na modulu Arduino převedeme analogový signál do digitální podoby, na které můžeme kupříkladu sledovat změny výšky hladiny nebo objemu vody apod.



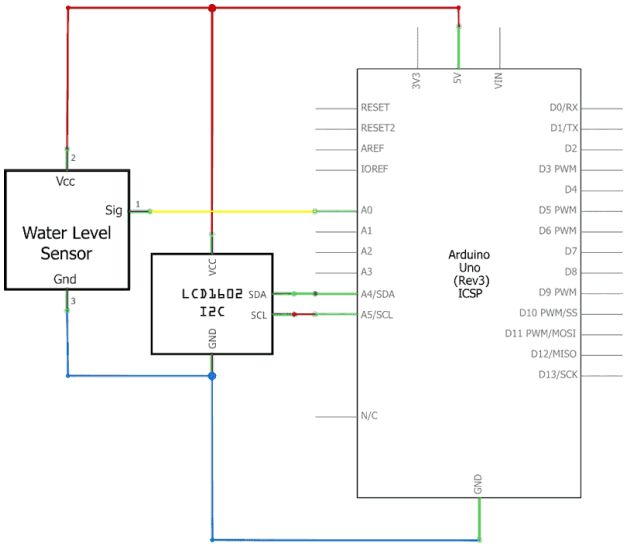

## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Spojení mezi displejem I²C LCD1602 a modulem Arduino, stejně tak jako propojení modulu Arduino a čidla výšky hladiny znázorňují následující tabulky.

**Blokové schéma**



**Elektronické schéma**




| I²C LCD1602 | modul Arduino | |
|---|---|---|
| | UNO | Mega |
| SDA | A4 | 20 |
| SCL | A5 | 21 |
| $V_{CC}$ | 5V | 5V |
| GND | GND | GND |

| Senzor hladiny | modul Arduino |
|---|---|
| S | A0 |
| + | 5V |
| – | GND |

**Krok 2:** V prostředí mBlock vytvoříme následující program. Při tvorbě programu je třeba znát adresu použitého I²C LCD displeje, neboť u každého příkazů pro LCD je třeba nastavit aktuální adresu. (viz rozbalovací pole „adr.“).

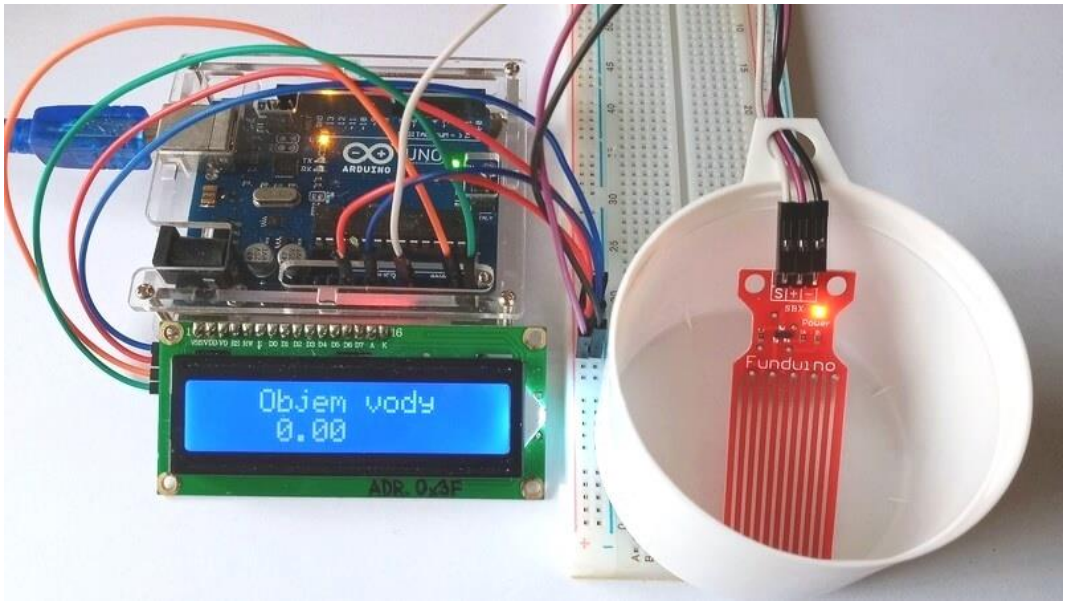

Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-26/26-Senzor-hladiny-vody.mblock.

### Vysvětlení kódu

V počáteční části programu nastavíme parametry použitého LCD, jako jsou jeho rozměry (sloupce, řádky) a zapnutí podsvícení. Je třeba se zaměřit na správné nastavení adresy použitého I²C převodníku! Před spuštění hlavní nekonečné smyčky „opakuj stále“ je ještě na střed první řádky vypsán text „Objem vody“. Hlavní nekonečná smyčka pak již pracuje jen s druhou řádkou displeje, kam vypisuje získané hodnoty z analogového vstupu A0, které se načetly do proměnné `waterValue`. V tomto případě tato proměnné nabývá hodnot od 0 do 1023. Pro korektní prezentaci naměřené hodnoty, by bylo dobré tuto hodnotu ještě zpracovat – např. přepočítat na mililitry dle použité nádoby. Protože zde není jasné, jak bude zobrazený text dlouhý, nedochází zde k pouhému přepisování zobrazeného textu, jako bylo kupříkladu v předešlé lekci. Z důvodů smazání druhé řádky je tato řádka vždy přepsána textem složeným z šestnácti mezer.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Když se ponoří senzor hladiny do vody, můžeme na displeji vidět hodnotu objemu (*odpovídá hloubce ponoření*) zobrazenou v „neurčitých“ jednotkách na LCD displeji. Pokud bychom chtěli dát zobrazeným číslům nějaký smysl (jednotku), musíme provést kalibraci, při které nalijeme známé množství vody do nádoby a zapíšeme, jaká hodnota na LCD tomu odpovídá. Následně převedeme rozsah měřených jednotek na objem vody – podobně jako jsme v předešlé lekci spočítali hodnotu napětí.


Objem vody
0.00
ADR 0x3F
Funduino

# Lekce 27 – Teplotní čidlo LM35

## Úvod

Dalším senzorem, který převádí nějakou neelektrickou veličinu na elektrický analogový signál je teplotní čidlo LM35. Jedná se o integrovaný obvod, obsahující převodník teploty na napětí, s maximální nelinearitou ±0,75 °C v rozsahu teplot –55 °C až 150 °C. Často používaná varianta obvodu je zapouzdřena v malém plastovém pouzdře TO-92, ale existuje i verze ve větším plastovém pouzdře TO-220 a i verze v kovovém pouzdře TO-46.


Integrovaný obvod LM35 vyvinula americká společnost National Semiconductor. Je určen pro měření teploty ve stupních Celsia. Obvod má malý odběr proudu; napájecí proud je menší než 60 µA. Vzhledem k nízkému příkonu převodníku dochází jen k malému vlastnímu ohřevu součástky, což je důležité z hlediska přesnosti měření. Napájecí napětí převodníku – pokud se pohybuje v rozsahu 3 V až 5,5 V – nemusí být stabilizované a převodník lze tudíž napájet přímo z baterie, která se časem může částečně vybíjet. Pokud se mají měřit kladné i záporné teploty, tak zapojení je o něco složitější.

## Použité komponenty

- modul Arduino
- USB kabel
- teplotní čidlo LM35
- LCD displej s I²C převodníkem
- nepájivé pole
- vodiče pro nepájivé pole + vodiče typu „samec-samice“

## Princip

Abychom mohli získat poměrně přesný teploměr, je třeba si uvědomit, jak jsou výstupní hodnoty použitého čidla prezentovány ve stupních Celsia. Protože převodní koeficient činí 10 mV/°C, objeví se např. při teplotě 150 °C na výstupu napětí převodníku napětí 1500 mV. Mimochodem, v předešlém odstavci byla zmínka o měření záporných hodnot, tak jen pro zajímavost zde uvedeme, že bude-li měřená teplota záporná, např. –40 °C, bude i výstupní napětí záporné (–400 mV). Modul Arduino neumí na analogových pinech měřit záporná napětí, proto se v dalším zaměříme jen na měření kladných teplot.

Pro přepočet teploty $t$ na napětí $U_{OUT}$ platí následující vzorec:

$$U_{OUT} = 10 \cdot t$$

kde teplota se dosazuje ve stupních Celsia a výsledné napětí vychází v milivoltech.

Modul Arduino načítá vstupní napětí svým 10-bitovým AD převodníkem v rozmezí 0–1023, pro určení teploty `temp` z hodnoty AD převodníku (`imVal`) tedy použijeme následující vzorec:

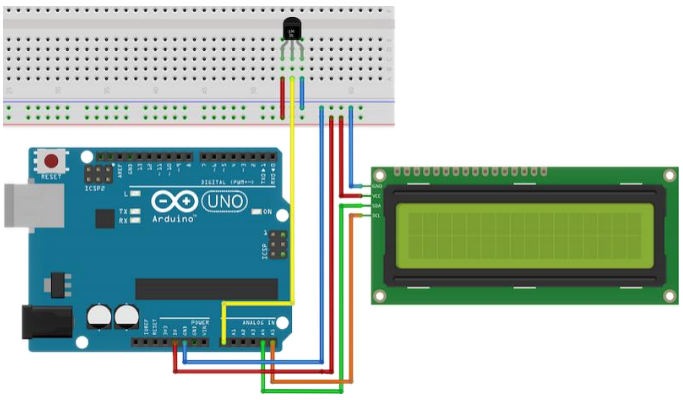
$$\texttt{temp} = \underbrace{\frac{5}{1024}}_{\substack{\text{převod mezi}\\ \text{napětím a}\\ \text{rozsahem ADC}}} \cdot \underbrace{\frac{1000}{10}}_{\substack{\text{milivolty}\\ \text{vs.}\\ \text{koef. čidla}}} \cdot \texttt{imVal} = 0{,}48828125 \cdot \texttt{imVal}$$

Načtené hodnoty z pinu A0 budeme v programu násobit konstantou 0,48828125 a výsledná hodnota bude odpovídat naměřené teplotě.

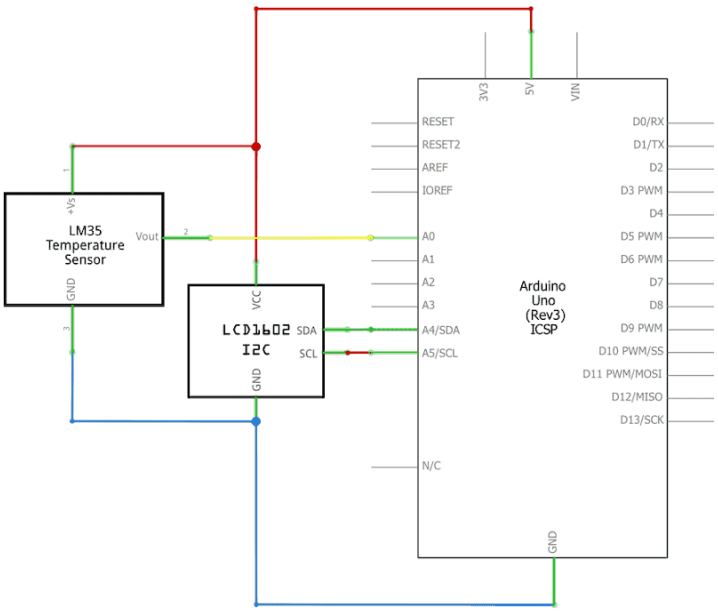
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Spojení mezi displejem I²C LCD1602 a modulem Arduino, stejně tak jako propojení modulu Arduino a čidla LM35 znázorňují následující tabulky.

**Blokové schéma**



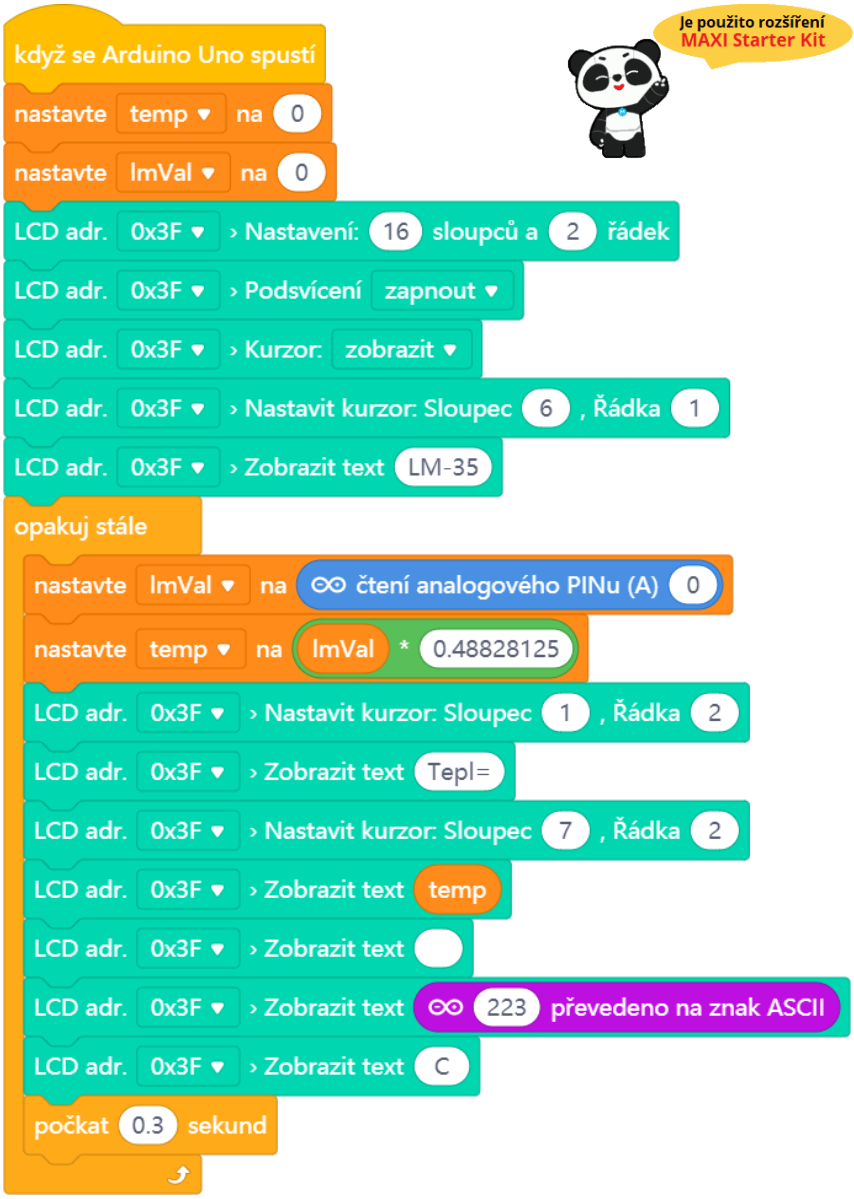
**Elektronické schéma**




| I²C LCD1602 | modul Arduino | |
|---|---|---|
| | UNO | Mega |
| SDA | A4 | 20 |
| SCL | A5 | 21 |
| $V_{CC}$ | 5V | 5V |
| GND | GND | GND |

| LM35 | modul Arduino |
|---|---|
| Analog Out | A0 |
| $V_{CC}$ | 5V |
| GND | GND |

**Krok 2:** V prostředí mBlock vytvoříme následující program. Při tvorbě programu je třeba znát adresu použitého I²C LCD displeje, neboť u každého příkazů pro LCD je třeba nastavit aktuální adresu. (viz rozbalovací pole „adr.“).

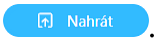



Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-27/27-Teplotni-cidlo-LM35.mblock.

## Vysvětlení kódu

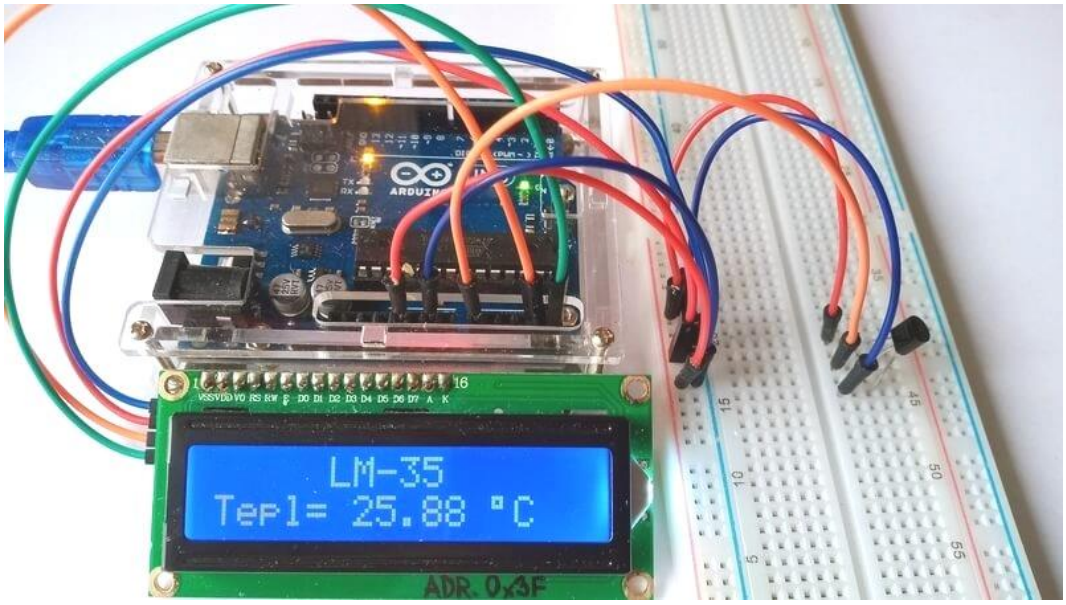
V počáteční části programu nastavíme parametry použitého LCD, jako jsou jeho rozměry (sloupce, řádky) a zapnutí podsvícení. Je třeba se zaměřit na správné nastavení adresy použitého I²C převodníku! Před spuštění hlavní nekonečné smyčky „opakuj stále“ jsou ještě nataveny proměnné `temp` a `imVal`. Jedna bude odpovídat vypočítané teplotě, druhá slouží pro načtení hodnoty z AD převodníku modulu Arduino. V hlavní nekonečné smyčce je načítáno teplotní čidlo, které je připojené na pin A0. Získaná hodnota je následně vynásobena konstantou 0,48828125, abychom získali hodnotu teploty ve stupních Celsia. Získaná hodnota

teploty je vypsána na druhý řádek displeje. V minulém programu, kde jsme nevěděli, jak bude zobrazovaný text dlouhý, jsme tuto řádku mazali přepsání několika mezer. Zde je situace podobná, neboť nevíme, zda bude teplota řádově jednotky nebo desítky (popř. stovka) stupňů. Možná jsme si v minulé lekci všimli, že při přepisování mezerami docházelo k blikání. Proto zde zkusíme opět text přepisovat, ale za posledním znakem přidáme dvě mezery, kdyby došlo, že se zobrazí kratší text. Tyto mezery jsou přidány za znak C, který se zobrazuje na konec řádky. Za zmínku ještě stojí vložení symbolu stupně. Jelikož není dobré tyto speciální symboly vkládat pomocí znaku, je zde využito vložení pomocí ASCII kódu. Podobně lze vložit i další nestandardní znaky z ASCII tabulky.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

.

**Krok 4:** Po spuštění programu uvidíme na LCD displeji hodnotu aktuální teploty. Pro sledování směny teploty je možné třeba stisknout čidlo LM35 mezi prsty.



# Lekce 28 – Hodiny reálného času

## Úvod

V jedné z předešlých lekcí jsme postavili stopky, tak proč se nyní nepustit do celých hodin. Modul Arduino umí pracovat s časovými údaji. Problém ale nastane, když se Arduino odpojí od elektřiny. Po opětovném zapojení ke zdroji rázem zapomene časové údaje a začíná vše od začátku. Právě pro zachování časových údajů používají tzv. moduly reálného času. Jsou to vlastně hodiny zálohované svou vlastní baterií a tak po výpadku napájení nebo restartu modulu Arduino mohou potřebné „ztracené“ časové údaje poskytnout. Existuje mnoho různých modulů hodin reálného času (RTC), jako je např. DS1302, DS1307, PCF8485 atd. Jsou široce používány vzhledem k jejich nízké ceně a jednoduchému použití. V této lekci budeme používat DS1302 modul hodin reálného času.

## Použité komponenty

- modul Arduino
- USB kabel
- DS1302 (modul reálného času)
- LCD displej s I²C převodníkem
- nepájivé pole
- vodiče pro nepájivé pole + vodiče typu „samec-samice“

## Princip

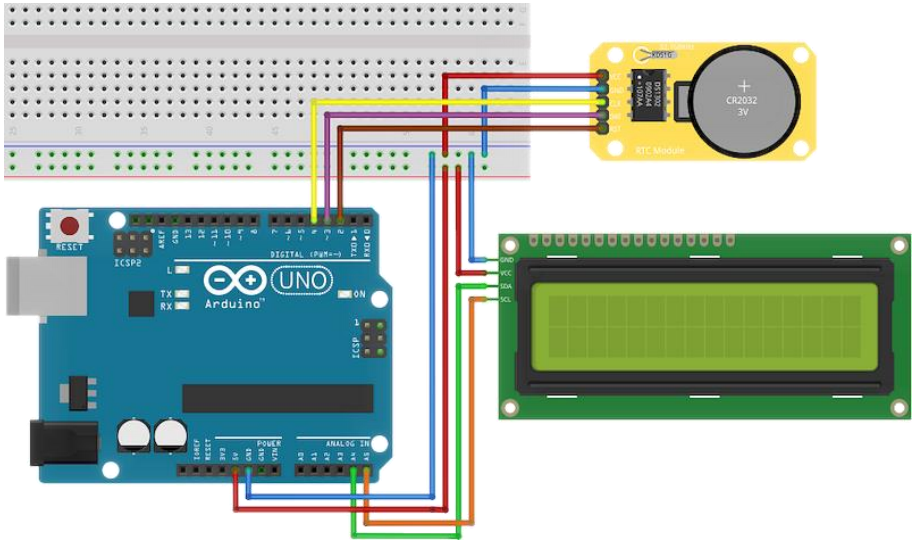
Obvod udržovacích hodin DS1302 obsahuje hodiny reálného času, kalendáře a 31 B statickou paměť RAM. Záložní běh hodin zajišťuje 3 V baterie, přesnost je dána použitým krystalem. RTC modul komunikuje s okolím přes jednoduché sériové rozhraní. Hodiny reálného času poskytují sekundy, minuty, hodiny, den v týdnu, den v měsíci, měsíc a rok. Konec měsíce je automaticky přizpůsobován u měsíců s méně než 31 dny, včetně korekce přestupného roku. Hodiny pracují v 24-hodinovém nebo 12-hodinovém režimu s indikátorem AM/PM.


Pro komunikaci s hodinami/pamětí jsou použity pouze tři vodiče: RST, DAT (sériová data) a CLK (hodinový signál). Data mohou být přenesena z/do modulu hodin nebo paměti. DS1302 je navržen pro velmi malou spotřebu a udržení dat při příkonu menším, než 1 μW.

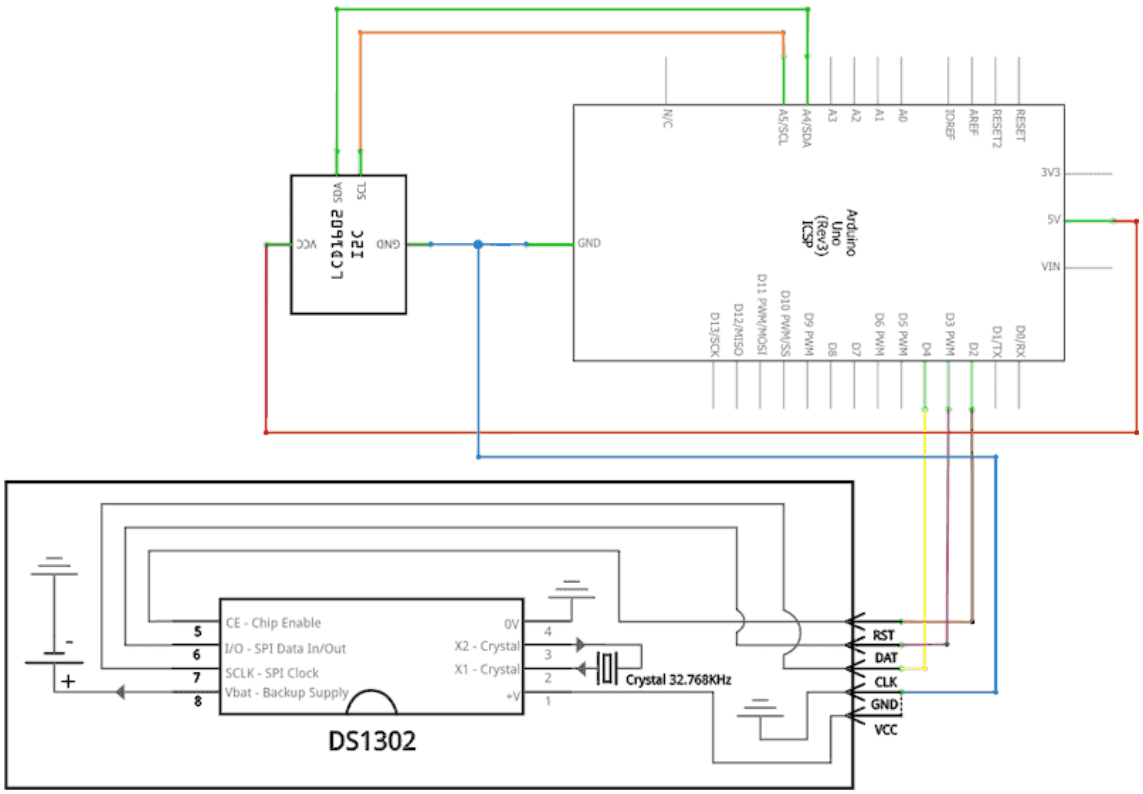
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. I²C LCD displej připojíme stejně, jako v předešlých lekcích. Níže uvedené tabulky uvádějí propojení modulu Arduino, I²C LCD displeje a modulu reálných hodin DS1302.

**Blokové schéma**



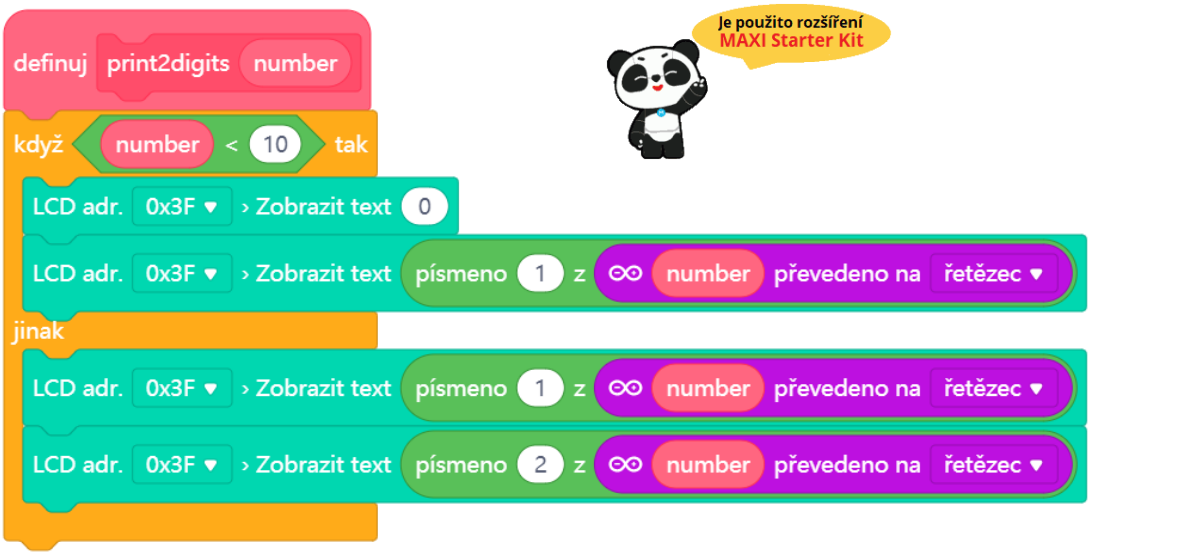

**Elektronické schéma**

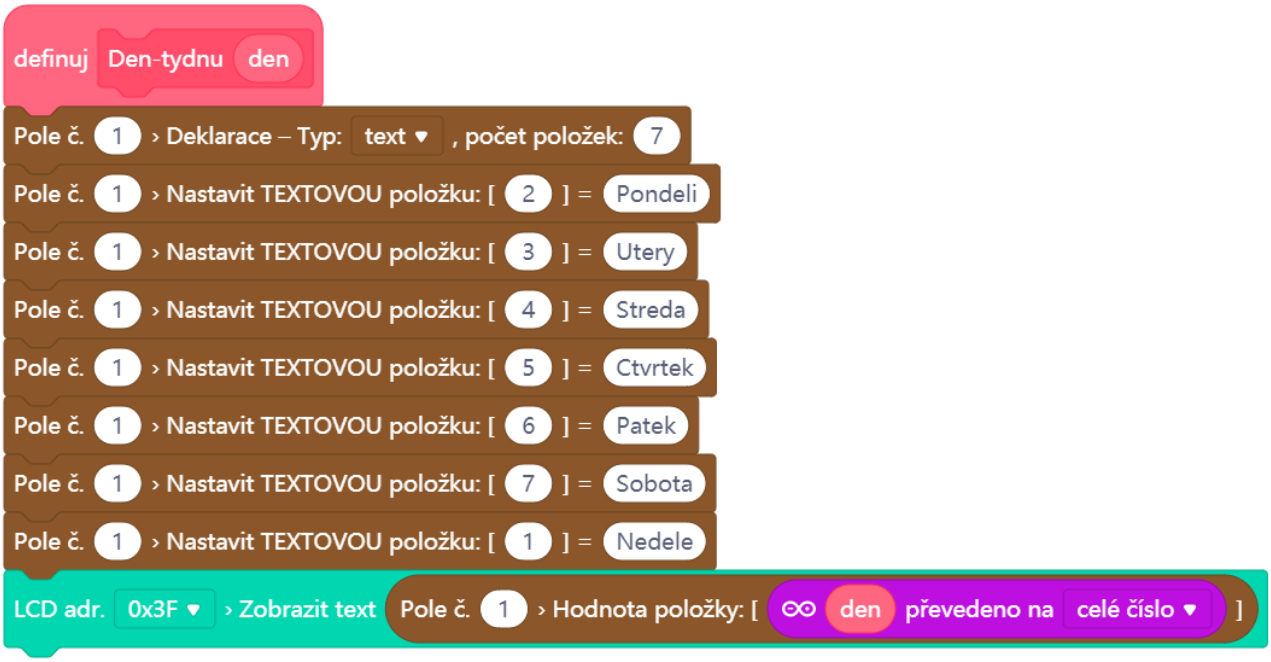


| I²C LCD1602 | modul Arduino | |
|---|---|---|
| | UNO | Mega |
| SDA | A4 | 20 |
| SCL | A5 | 21 |
| $V_{CC}$ | 5V | 5V |
| GND | GND | GND |

| DS1302 | modul Arduino |
|---|---|
| RST | 2 |
| DAT | 3 |
| CLK | 4 |
| $V_{CC}$ | 5 V |
| GND | GND |

**Krok 2:** V prostředí mBlock vytvoříme následující program. Při tvorbě programu je třeba znát adresu použitého I²C LCD displeje, neboť u každého příkazů pro LCD je třeba nastavit aktuální adresu. (viz rozbalovací pole „adr.“). V tomto programu se opět použije několik podprogramů, proto začneme nejdříve s nimi, hlavní program je uveden až na závěr.




Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-28/28-Hodiny-realneho-casu.mblock.

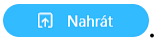
### Vysvětlení kódu

Podprogram „print2digits“ se vstupním parametrem `number` vypisuje na aktuální místo na LCD displeji (tam, kde je právě nastavena pozice kurzoru) zadané číslo ve dvouznakovém formátu. To znamená, že dvouciferné číslo vypíše normálně, jednocifernému přidá na začátek 0. Provedení tohoto konkrétního podprogramu je takové, že kdyby náhodu bylo zadané troj a víceciferné číslo, budou vždy zobrazeny jen jeho první dvě číslice. Na první pohled poněkud zvláštní řešení pomocí převodu čísla na text vychází z problému, že prostředí mBlock považuje všechna čísla jako reálná a na LCD displeji se všechna reálná čísla vypisují s dvěma desetinnými místy.

Druhý podprogram „Den-týdnu“ se vstupním parametrem `den` dělá přesně to, co jeho název napovídá. Dle zadaného pořadového čísla dne vypíše jeho název. Jen je třeba si upozornit, že podle anglické normy, ve které se v časových údajích modulu Arduino pracuje, je prvním dnem v týdnu neděle. Pondělí má tedy číslo 2, úterý číslo 3… až sobota číslo 7. Podprogram „Den-týdnu“ nejen převádí anglické pořadové číslo na český název, ale hned jej i vypisuje na danou na aktuální místo na LCD displeji.

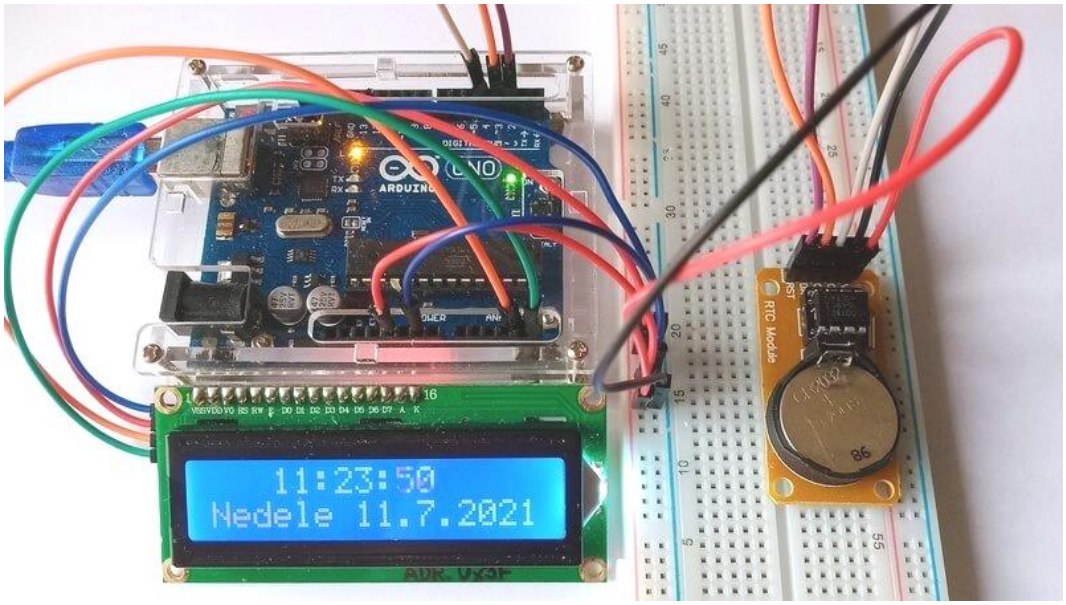
Hlavní program ve své první části nastavuje řídicí piny RTC modulu reálného času, stejně jako nastavuje potřebné nastavení LCD displeje. Důležitým příkazem je v této části načtení časových údajů z RTC modulu do modulu Arduino. V hlavní nekonečné smyčce „opakuj stále“ je pracováno s potřebnými časovými údaji v modulu Arduino. První podmínka testuje, zda je počet hodin větší než devět (tedy dvojciferný), pokud ano, je počet hodin vytištěn od prvního sloupce. Pokud je aktuální počet hodin jednociferný, je počet hodin vytištěn až do druhého sloupce. Toto řešení dělá přesný opak toho, co podprogram „print2digits“. Komu by se totiž líbil aktuální čas třeba 08:23:10? Takto číslovka počtu hodin zabírá dvě místa, ale u jednociferného počtu je na začátku mezera, tedy _8:23:10. Na zbylé číslovky, tedy na počet minut a vteřin je naopak podprogram „print2digits“ aplikován. Zbytek programu je na principu získání časové informace a jejího výpisu na LCD displej. Údaje jsou doplňovány dvojtečkami a tečkami, aby byl celkový výsledek na LCD displeji co nejvíce podobný skutečným digitálním hodinám.

*V lekci, kde jsme konstruovali stopky, byla o problému nuly na začátku zmínka. Tento problém tam ale byl ponechán k samostatnému bádání. Tak nyní je zde uvedeno jedno z jeho možných řešení v podobě úvodní podmínky testující počet hodin (v případě stopek by se stejným způsobem testoval počet uplynulých vteřin).*

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

.

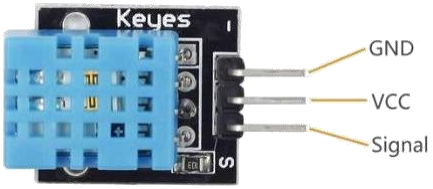
**Krok 4:** Po spuštění programu funguje modul Arduino jako hodiny, dokonce i s datem. Potřebné údaje se zobrazují na LCD displeji. Při vypnutí zdroje se sice modul Arduino vynuluje, ale čas běží v RTC modulu (reálných hodin) díky vestavěné baterii dále. Po opětovném zapnutí napájení modulu Arduino se na displeji opět objeví správný čas.


11:23:50
Nedele 11.7.2021
RTC Module

# Lekce 29 – Senzor teploty a vlhkosti vzduchu DHT11

## Úvod

Když se nám podařilo postavit skutečné hodiny, co takhle zkusit jednoduchou domácí meteorologickou stanici? Experimentální sada Arduino MAXI Starter kit obsahuje mimo jiného digitální čidlo teploty a vlhkosti DHT11. Tento senzor, který měří teplotu a relativní vlhkost, zapisuje obě tyto hodnoty v digitální podobě na svůj digitální výstup. Údaje jsou na výstupu sice zakódovány do zvoleného protokolu, ale o jeho načtení se opět postarají programové bloky rozšíření MAXI Starter kit. Takže se s tímto čidlem bude pracovat stejně jednoduše jako kupříkladu s předchozím modulem reálného času.

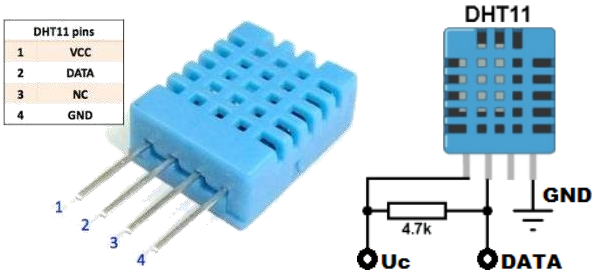



## Použité komponenty

- modul Arduino
- USB kabel
- modul senzoru DHT11
- LCD displej s I²C převodníkem
- nepájivé pole
- vodiče pro nepájivé pole + vodiče typu „samec-samice“

## Princip

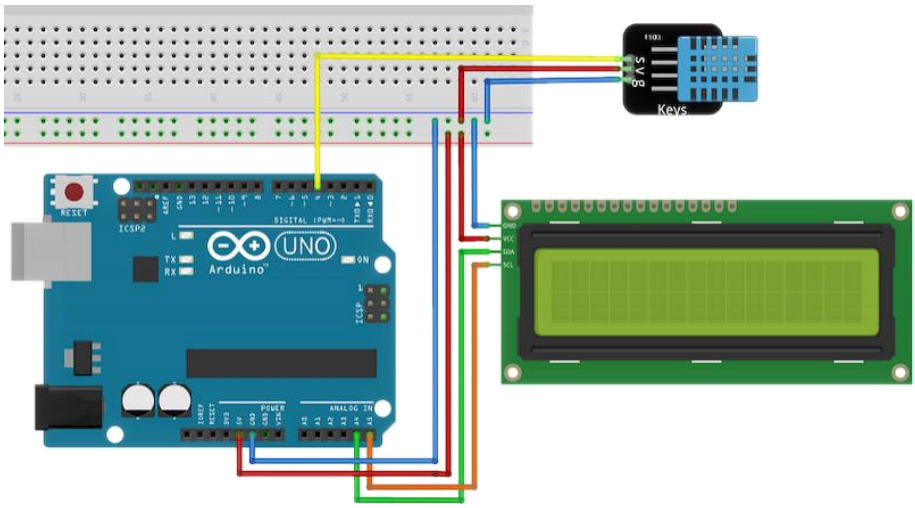
Digitální čidlo teploty a vlhkosti DHT11 obsahuje odporové čidlo vlhkosti a polovodičový teplotní senzor. Obě čidla jsou doplněna osmibitovým analogově-digitálním převodníkem. Na destičce čidla DHT11 je kromě samotného čidla ještě i pomocný pull-up rezistor. Pokud používáme čidlo v podobě modulu (na destičce) jsou pro připojení k dispozici tři piny: $V_{CC}$, GND, a S (DATA nebo Signal). Komunikační proces začíná posláním startovního signálu do čidla DHT11, jakmile čidlo změří potřebné údaje, čidlo vrací odpověď, což je 40bitová sekvence dat (16 bitů hodnoty vlhkosti + 16 bitů hodnoty teploty + 8 bitů kontrolního součtu). Čidlo DHT11 lze zakoupit i v „holé“ verzi, kdy má samotné čidlo vývody čtyři. Následující obrázek ukazuje nejen zapojení jednotlivých vývodů (NC – znamená „nezapojeno“) samotného čidla DHT11, ale především i připojení pull-up rezistoru pro jeho provoz.



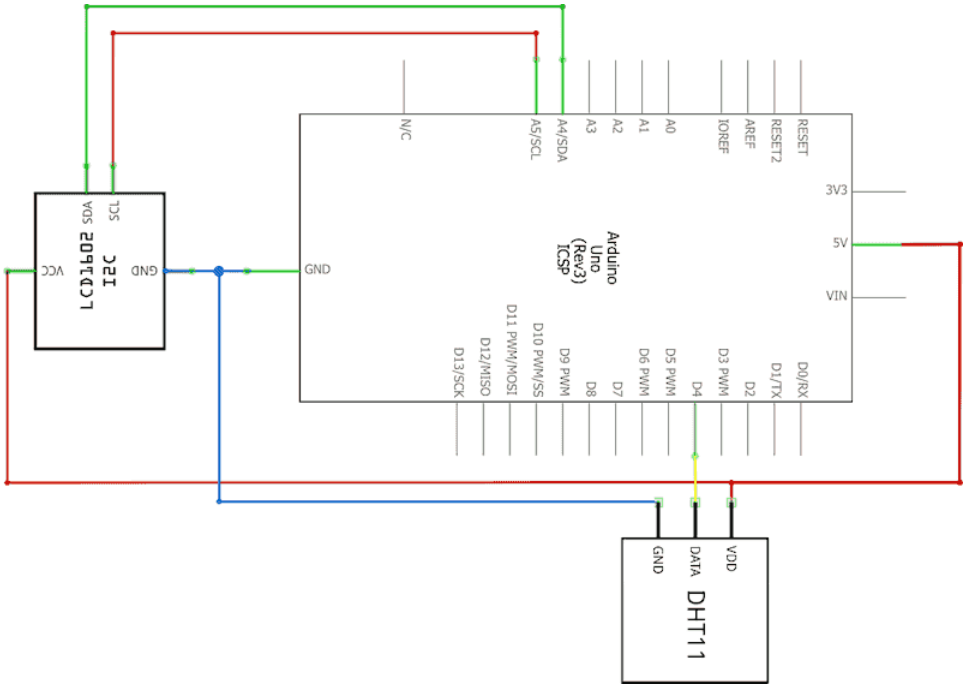
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. I²C LCD displej připojíme stejně, jako v předešlých lekcích. Čidlo DHT11 a LCD displej připojíme dle níže uvedených tabulek.

**Blokové schéma**




**Elektronické schéma**




| I²C LCD1602 | modul Arduino | |
|---|---|---|
| | UNO | Mega |
| SDA | A4 | 20 |
| SCL | A5 | 21 |
| $V_{CC}$ | 5V | 5V |
| GND | GND | GND |

| DHT11 | modul Arduino |
|---|---|
| S | 4 |
| + | 5 V |
| – | GND |

**Krok 2:** V prostředí mBlock vytvoříme následující program. Při tvorbě programu je třeba znát adresu použitého I²C LCD displeje, neboť u každého příkazů pro LCD je třeba nastavit aktuální adresu. (viz rozbalovací pole „adr.“).

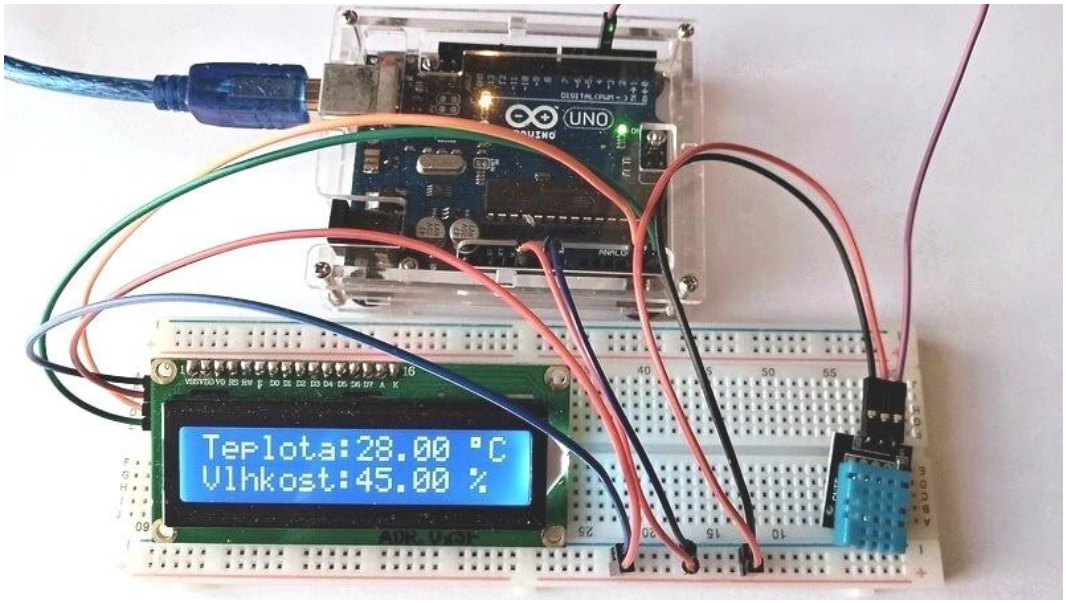

Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-29/29-Senzor-teploty-a-vlhkosti-vzduchu-DHT11.mblock.

### Vysvětlení kódu

První část programu nastavuje údaje pro ovládání LCD displeje, též deklaruje typ použitého senzoru teploty a vlhkosti a komunikační pin, které kterému je na modulu Arduino čidlo připojeno. V hlavní nekonečné smyčce „opakuj stále“ je postupně z čidla načítána hodnota teploty a vlhkosti a zapisována na LCD displej. Díky poměrně značné pomalosti DHT čidla (doba převodu asi 1 s) program ani nepotřebuje čekací příkaz pro zpomalení. Díky tomu, že hodnota vlhkosti a teploty budou zpravidla dvouciferné hodnoty, není v tomto programu řešeno mazání předchozích hodnot, které jsou na displeji prostě jen přepisovány.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Po spuštění programu na modulu Arduino můžeme vidět na LCD displeji aktuální hodnotu teploty a vlhkosti vzduchu.

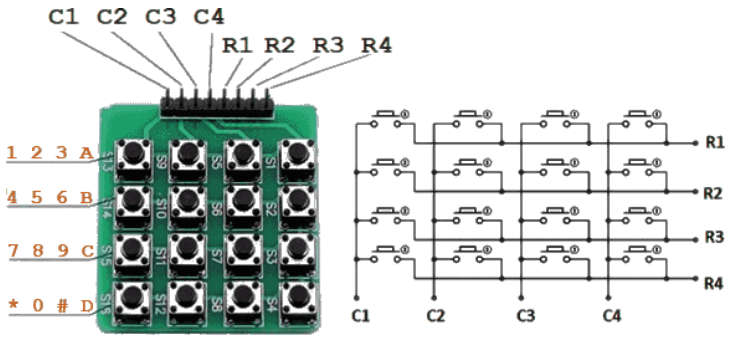


# Lekce 30 – Vstupní ochranný systém s heslem

## Úvod

Poté, co jsme si ukázali funkčnost několik samostatných modulů, zkusíme je opět zkombinovat a vytvořit něco trochu interaktivního a praktického. V této lekci použijeme I²C LCD displej, modul relé a klávesnici. Naším cílem bude sestavit jednoduchý číselný zámek, který by bylo možné použít pro bezpečnostní dveře.

Novým prvkem této lekce je maticová klávesnice 4×4. Princip její funkce je naznačena na následujícím obrázku. V principu jde o „šikovně" zapojené spínače mezi vodiči R1–R4, které odpovídají řádkám a vodiči C1–C4, které naopak odpovídají sloupcům. Je to podobný způsob propojení, jako bylo v případě zobrazovací LED matice 8×8, kterou jsme poznali v jedné z předešlých lekcí. Každý spínač spojuje právě jeden vodič R s vodičem C. Podle toho, které konkrétní vodiče R a C stisknutím spínače propojíme, poznáme stisknuté tlačítko.




## Použité komponenty

- modul Arduino
- USB kabel
- maticová klávesnice 4×4
- modul relé
- LCD displej s I²C převodníkem
- nepájivé pole
- vodiče pro nepájivé pole + vodiče typu „samec-samice"

## Princip

Jak jsme naznačili, naším úkolem je vyrobit číselný (kódový) zámek. V programu tedy nastavíme heslo (např. 123456). Pomocí zadávání na klávesnici zadáme vstupní kód. Pokud se bude shodovat s naším heslem, sepne relé, jinak zůstane zámek zamčený. Samotné zadávání kódu bude probíhat tak, že jej spustíme stiskem hvězdičky *. Naopak konec pro zadání hesla potvrdíme křížkem #.

Pochopitelně chceme, aby zámek s uživatelem komunikoval, k tomu použijeme LCD displej. Na začátku je na displeji úvodní hláška „Vitejte!". Po zadání hesla se na displeji kupříkladu objeví „Spravne zadano!" „Prosim, vstupte!", nebo se naopak zobrazí „Vstupní chyba!" „Prosim, zkus znovu!". Po uplynutí dvou sekund se na displeji objeví úvodní hláška „Vitejte!"

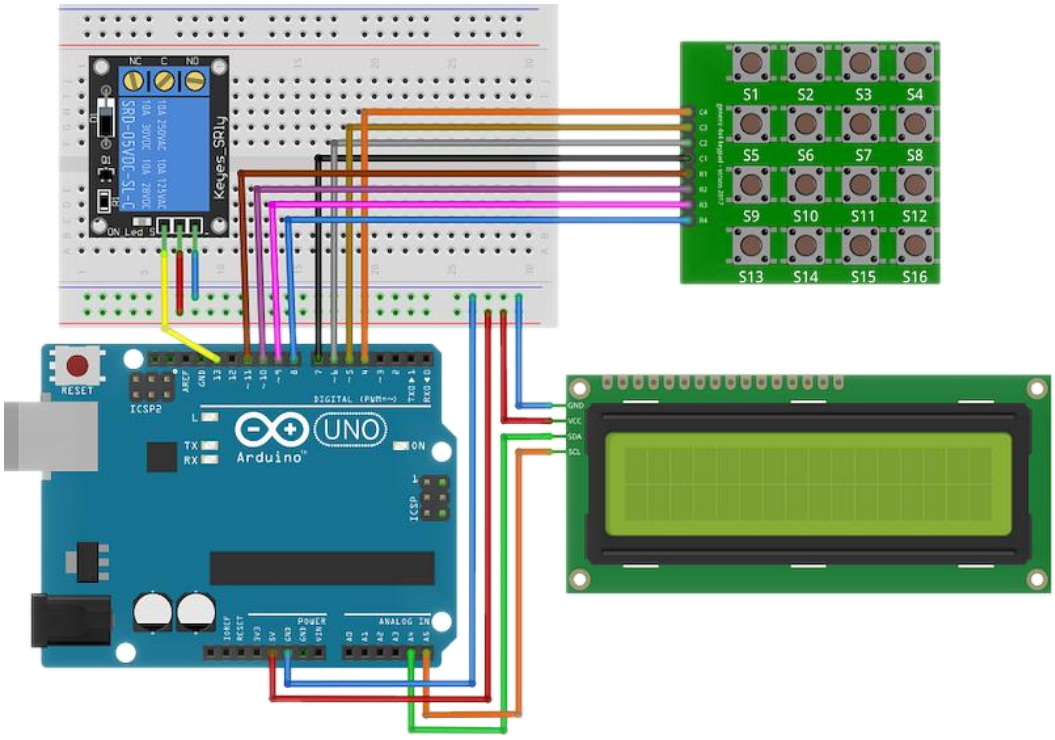
**POZNÁMKA:**
Protože naše klávesnice je složena z tlačítek bez jakéhokoliv popisu a vrácené kódy odpovídají jen stisknutému sloupci a řádce, musíme si nastavení kláves nastavit v programu. Rozšíření MAXI Starter kit předpokládá standardní rozložení klávesnice podle dnes dostupných klávesnic 4×4. Toto rozložení je uvedeno v následující obrázek.


Rozložení klávesnice si ale můžeme zvolit, jaké chceme – například následující rozložení: 1, 2, 3, 4 v první řadě; ve druhé řadě 5, 6, 7, 8; ve třetí řadě 9, A, B a C a ve čtvrté řadě to je D, *, 0, #. I s tímto rozšíření MAXI Starter kit počítá. V sekci vysvětlení kódu si ukážeme, jak bychom tuto definici změnili v programu.

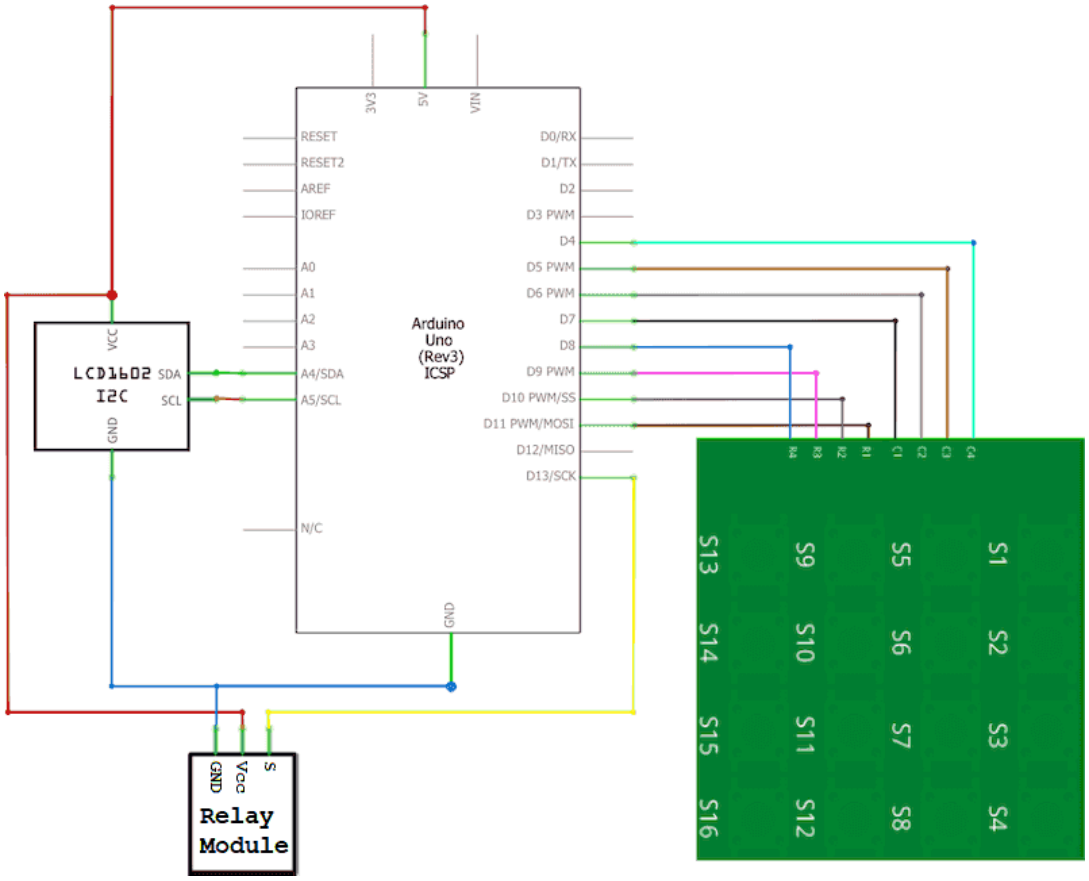
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. I²C LCD displej připojíme stejně, jako v předešlých lekcích. Propojení modulu Arduino s ostatními komponentami je uvedeno v tabulkách.

**Blokové schéma**



**Elektronické schéma**

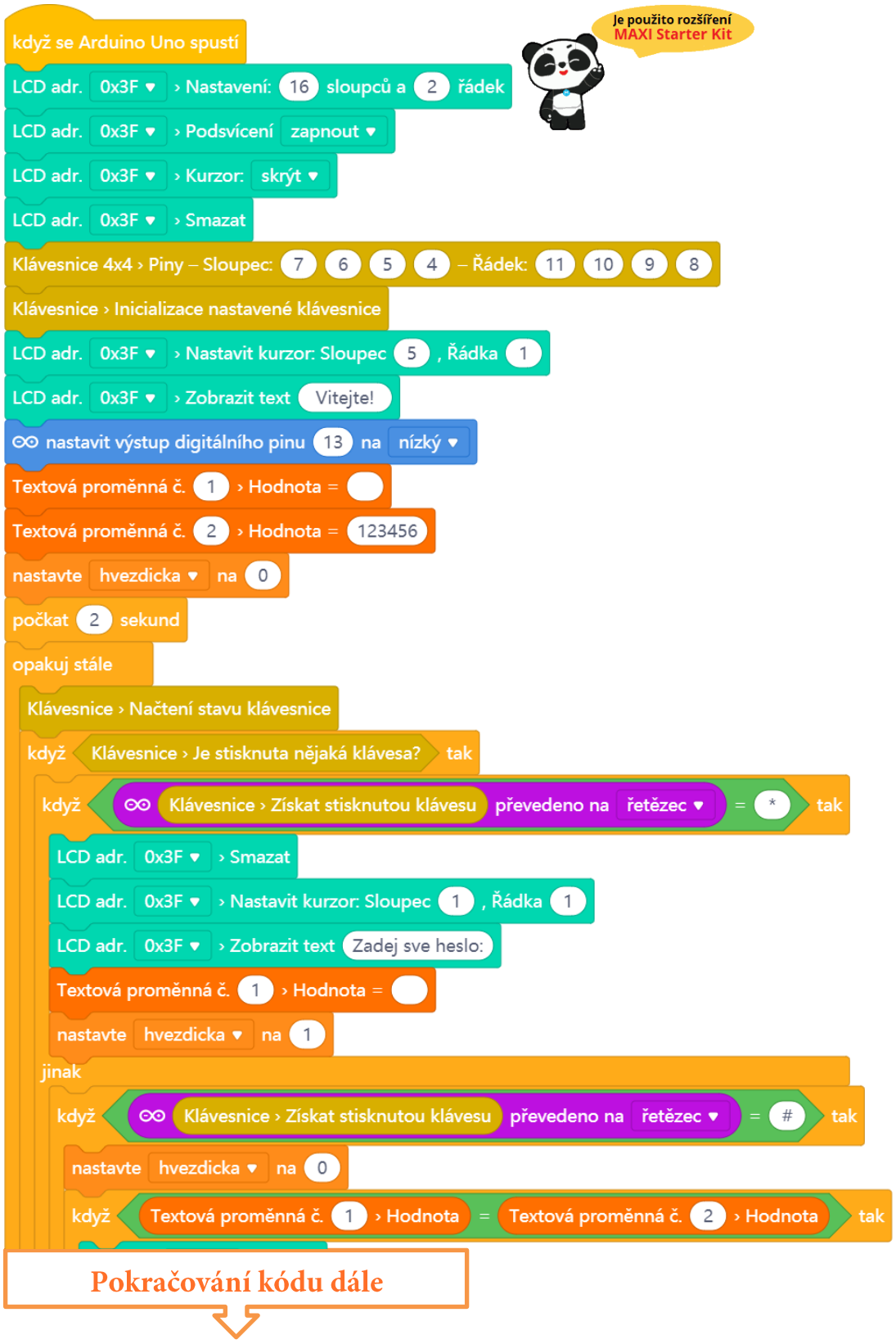


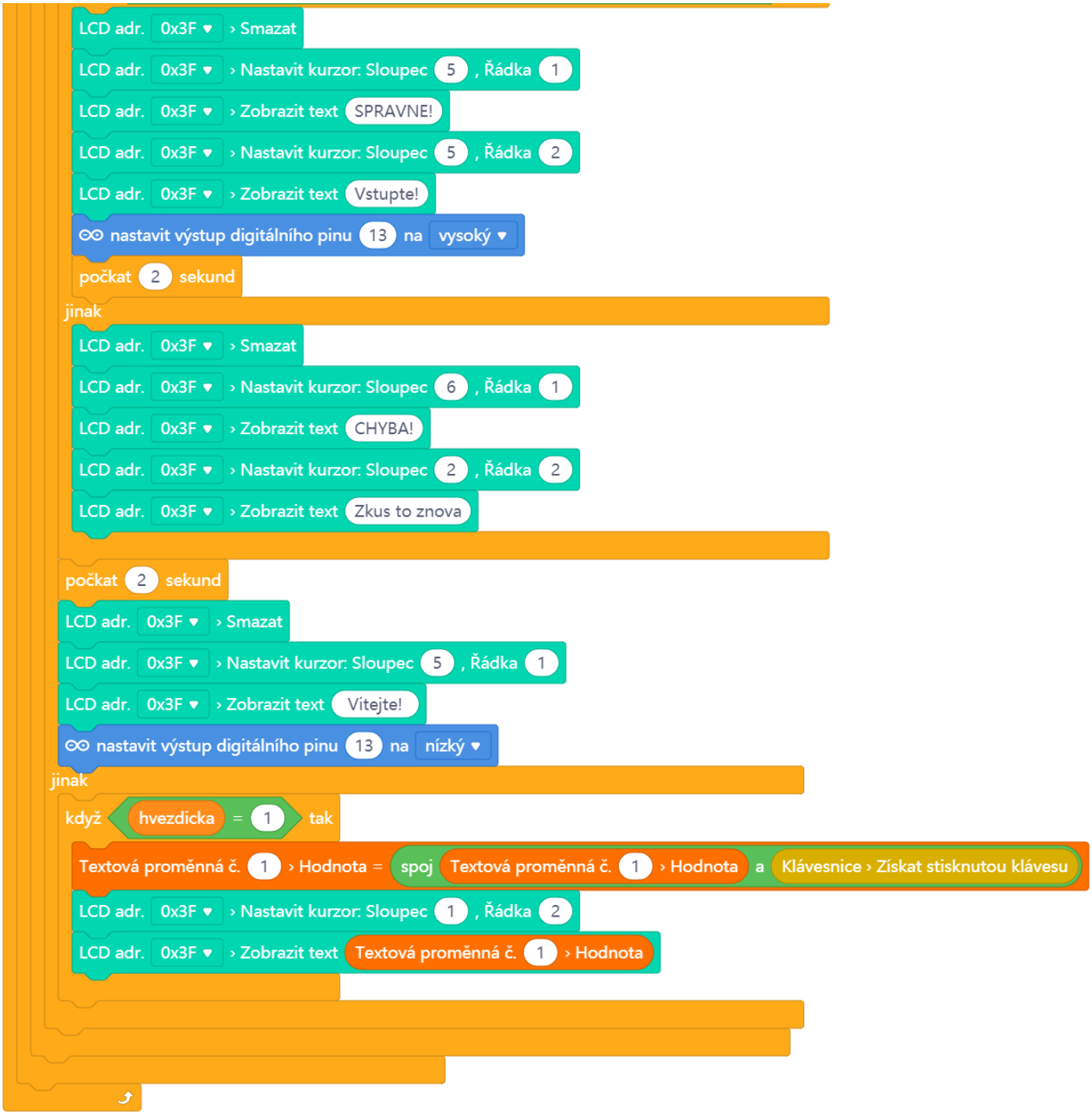

| klávesnice 4×4 | modul Arduino |
|---|---|
| C1 | 7 |
| C2 | 6 |
| C3 | 5 |
| C4 | 4 |
| R1 | 11 |
| R2 | 10 |
| R3 | 9 |
| R4 | 8 |

| modul Relé | modul Arduino |
|---|---|
| S | 13 |
| $V_{CC}$ (+) | 5V |
| GND (–) | GND |

| $I^2C$ LCD1602 | modul Arduino | |
|---|---|---|
| | UNO | Mega |
| SDA | A4 | 20 |
| SCL | A5 | 21 |
| $V_{CC}$ | 5V | 5V |
| GND | GND | GND |

**Krok 2:** V prostředí mBlock vytvoříme následující program. Při tvorbě programu je třeba znát adresu použitého I²C LCD displeje, neboť u každého příkazů pro LCD je třeba nastavit aktuální adresu. (viz rozbalovací pole „adr.“).






Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-30/30-Vstupni-ochranny-system-s-heslem.mblock.

## Vysvětlení kódu

První část programu je opět věnována prvotnímu nastavení, jako je nastavení parametrů použitého LCD displeje. Za zmínku stojí dva okrové programové bloky, které se zabývají inicializací klávesnice. První blok slouží pro nastavení ovládacích pinů při řádky a sloupce, druhý provede inicializaci klávesnice, čímž se zadané piny nastaví tak, aby mohla být klávesnice vyhodnocována. Pokud bychom se rozhodli změnit

rozložení klávesnice, jak jsme se zmínili dříve, musíme to učinit ještě před inicializací klávesnice. Proto mezi bloky pro nastavení pinů a blok inicializace bychom museli vložit následující bloky (červeně orámováno) s předefinováním jednotlivých míst – sloupec, řádek.


V hlavní nekonečné smyčce „opakuj stále“ se nejdříve načte stav klávesnice. Pokud je nějaká klávesa stisknuta, je její hodnota načtena. Programový blok načítání klávesnice nevrací souřadnici zmáčknuté klávesy (sloupec, řádek), ale rovnou její hodnotu (dle standardního rozložení, nebo předefinování). Získaná hodnota stisknuté klávesy je vypsána na LDC displej, aby byla vidět odezva pro uživatele. Následuje série podmínek, která testuje hned několik možností reakcí na stisknutou klávesu.

Zaprvé, pokud je stisknuta klávesa *, dojde ke smazání displeje a k výpisu uvítacího dialogu, taktéž dojde k vynulování textové proměnné pro uložení zadaného hesla a stav „probuzení“ zámku se v programu signalizuje uložením hodnoty 1 do proměnné `hvězdička`.

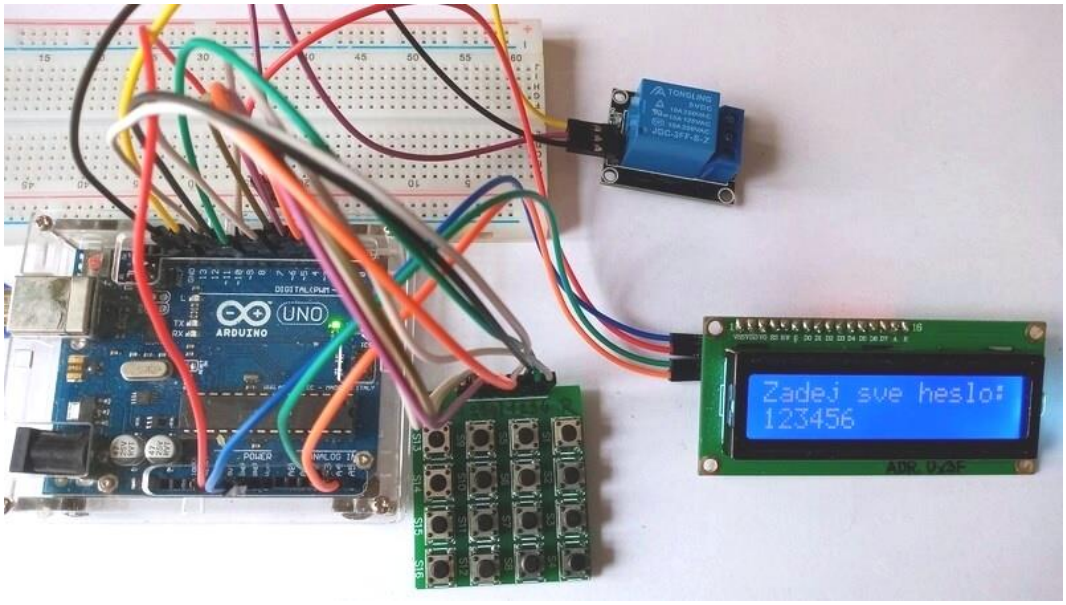
Pokud byla stisknuta klávesa #, došlo k potvrzení vstupního hesla. Dojde k porovnání proměnné s hodnotou zadaného hesla s proměnnou obsahující platné heslo. Pokud dojde ke shodě, následuje nastavení digitálního výstupu pro sepnutí relé a výpis hlášky na LCD. Pokud se hesla neshodují, dojde jen k výpisu patřičné hlášky

na LCD. Zároveň se v obou případech vynuluje proměnné `hvezdicka`, čímž se zámek opět uvede do standardního vyčkávacího režimu.

Třetí podmínka nastává, pokud byla stisknutá jiná klávesa než * nebo #. Podle stavu hodnoty proměnné `hvezdicka` se buď neprovede nic (nebyla stisknuta počáteční hvězdička, zámek tedy nereaguje), nebo se zadaná klávesa připojí k textové proměnné pro zadané heslo. A to dokud nedojde k potvrzení hesla klávesou #.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Po spuštění programu se na displeji objeví „Vitejte!“. Relé (zámek) je pochopitelně zavřený, což poznáme, protože vestavěná LED modulu Arduino zůstane zhaslá. Pokud stiskneme na klávesnici tlačítko *, objeví se pokyn „Zadej sve heslo“. Když zadáme 123456 a potvrdíme #, rozsvítí se vestavěná LED, relé sepne a na displeji se objeví: „SPRAVNE!“ „Vstupte!“. Při zadání nesprávného hesla se objeví na displeji „CHYBA!“ „Zkus to znova“ a relé zůstane v původním stavu. O dvě vteřiny později se na displeji opět objeví úvodní hláška „Vitej!“ a zámek čeká na nové zadání kódu.



# Lekce 31 – RFID vstupní ochranný systém

## Úvod

V závěrečné lekci navážeme na předchozí zapojení. Kódový zámek se zadáváním hesla vylepšíme tím, že místo vstupní klávesnice použijeme RFID modul. RFID je zkratka pro radiofrekvenční identifikaci. Jedná se o bezdrátovou aplikaci pro přenos dat za účelem identifikace a sledování identifikačních prvků. Takže nám odpadne zadávání hesla, ale budeme ovládat zámek přiložením RFID prvkem (přístupová karta nebo přívěšek). Výsledkem našeho zapojení tedy bude RFID vstupní ochranný systém.


## Použité komponenty

- modul Arduino
- USB kabel
- RFID čtečka
- RFID čipový přívěsek na klíče
- RFID čipová karta
- modul relé
- LCD displej s I²C převodníkem
- nepájivé pole
- vodiče pro nepájivé pole + vodiče typu „samec-samice“

## Princip

Především potřebujeme znát identifikační čísla našich RFID „klíčů“ (přívěšek, čipová karta). Každý RFID prvek musí mít toto identifikační číslo unikátní. Pro zjištění tohoto čísla použijeme kód prvního pomocného programu. Jakmile budeme znát konkrétní čísla svého RFID přívěšku nebo RFID karty, použijeme tento údaj do druhého programu, který už bude sloužit jako přístupový systém. Přístupový systém je navržen jako elektronický zámek, který odemkne (sepne relé), jen pokud je přiložen správná RFID prvek.

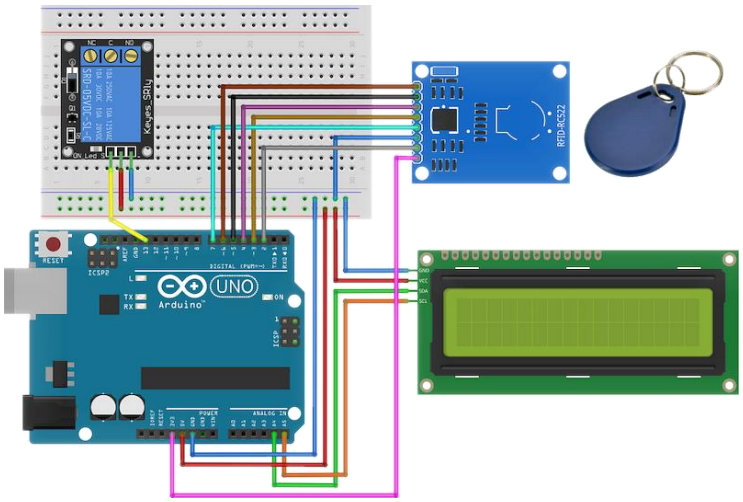
Na začátku na LCD displeji bude zobrazen text „Vitejte“. Přiložíme RFID prvek (přívěšek, kartu) k modulu RFID čtečky. Pokud je heslo správné, kontakt relé se sepne a na displeji v první řádce se zobrazí identifikační číslo RFID prvku (např. „ID:5AE4C955“). Pokud je přiložen RFID přívěšek je na druhém řádku displeje zobrazen text: „Ahoj Kosto!“. Pokud je přiložena RFID karta, dojde také k sepnutí relé a je zobrazen text „Zdravime bastlire“. V případě přiložení neznámého RFID prvku, zůstane kontakt relé rozpojeno a na displeji se zobrazí text „Ahoj, cizince!“. Po chvíli se přístupový systém vrátí to výchozího stavu s hláškou „Vitejte“.

## Postup experimentu

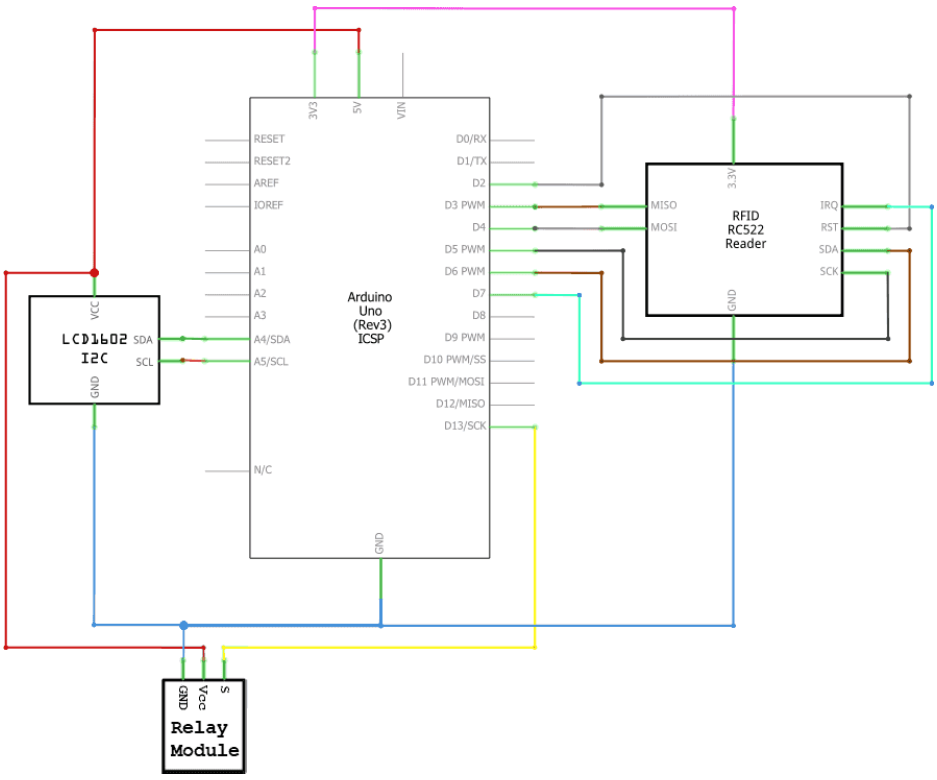
**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. I²C LCD displej připojíme stejně, jako v předešlých lekcích. Propojení modulu Arduino a ostatních komponent (RFID modulu, modulu relé a LCD displej s I²C převodníkem) zachycují následující tabulky.

**POZNÁMKA:**
Pro napájení modulu RFID čtečky musíme použít napájecí napětí 3,3 V, jinak se modul nenávratně zničí!

**Blokové schéma**



**Elektronické schéma**



| RFID čtečka | modul Arduino |
|---|---|
| $U_{CC}$ | 3,3V |
| RST | 2 |
| GND | GND |
| MISO | 3 |
| MOSI | 4 |
| SCK | 5 |
| SDA | 6 |
| IRQ | 7 |

| modul Relé | modul Arduino |
|---|---|
| S | 13 |
| $V_{CC}$ (+) | 5V |
| GND (–) | GND |

| $I^2C$ LCD1602 | modul Arduino | |
|---|---|---|
| | UNO | Mega |
| SDA | A4 | 20 |
| SCL | A5 | 21 |
| $V_{CC}$ | 5V | 5V |
| GND | GND | GND |

Nejdříve potřebujeme získat identifikační údaje svých RFID prvků, abychom je pak mohli dosadit do hlavního programu. Pro tento účel si vytvoříme následující pomocný program.

**Krok 2:** V prostředí mBlock vytvoříme následující program.


Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-31/31-RFID-vstupni-ochranny-system-getID.mblock.

### Vysvětlení kódu

Nejdříve je potřeba v programu definovat jednotlivé piny modulu Arduino, ke kterým je modul RFID čtečky připojen. V hlavní nekonečné smyčce „opakuj stále“ se neustále testuje, zda je ke čtečce přiložen RFID prvek. Pokud ano je jeho identifikační číslo vypsáno na sériový port. Číslo daného prvku se tedy přečteme aplikací Serial Monitor na počítači připojeném k modulu Arduino.

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Nyní modul Arduino odpojíme od prostředí mBlock pomocí tlačítkem Odpojit.
**USB kabel modulu Arduino ale od počítače neodpojujeme!**

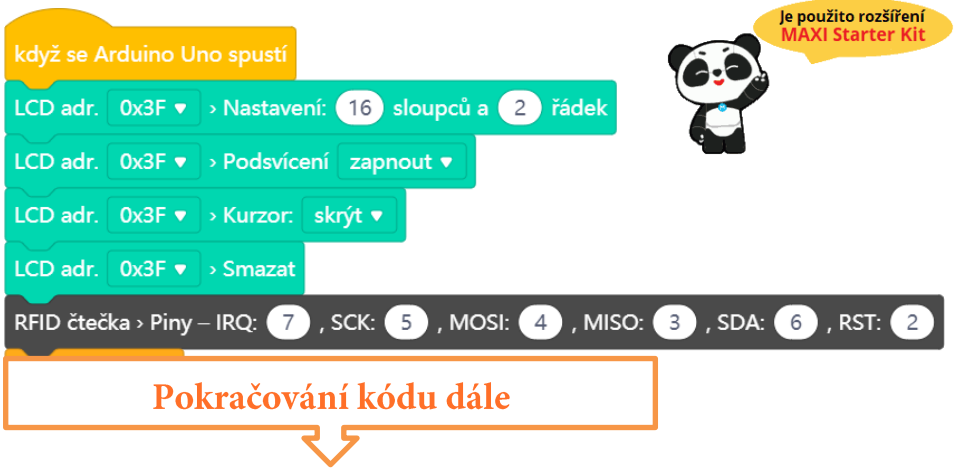
**Krok 5:** Spustíme na počítači aplikaci Serial Monitor. V Serial Monitoru vybereme port, na které je náš modul Arduino (např. COM5) připojen a nastavíme komunikační rychlost na hodnotu 115200 baudů. Na závěr se tlačítkem Connect připojíme k modulu Arduino. Objeví se okno sériového monitoru.

**Krok 6:** Postupně umístíme všechny své RFID prvky do indukční zóny modulu RFID čtečky. V okně Serial Monitoru se vždy zobrazí číslo přiloženého čipu. Všechna identifikační čísla si postupně poznamenáme, budeme je později potřebovat. V tomto okamžiku bychom tedy měli znát ID všech svých RFID klíčů. Můžeme začít stavět RFID přístupový systém!

**Krok 7:** Pomocí tlačítka Disconnect odpojíme modul Arduino od aplikace Serial Monitor. Aplikaci Serial Monitor ukončíme, již ji nebudeme potřebovat.

**Krok 8:** V prostředí mBlock se opět připojíme k modulu Arduino tlačítkem Připojení.

**Krok 9:** V prostředí mBlock sestavíme nový program. Nezapomeňme do podmínek identifikace (v kódu označeno červeně) vložit právě získané identifikační číslo RFID přívěšku a RFID karty.



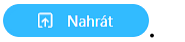

Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-31/31-RFID-vstupni-ochranny-system.mblock.

## Vysvětlení kódu

Na začátku programu je obvyklé nastavení LCD displeje a RFID čtečky. V hlavní nekonečné smyčce „opakuj stále“ je podobně jako v předešlém programu je testována RFID čtečka. Akce nastává až v případě přiložení

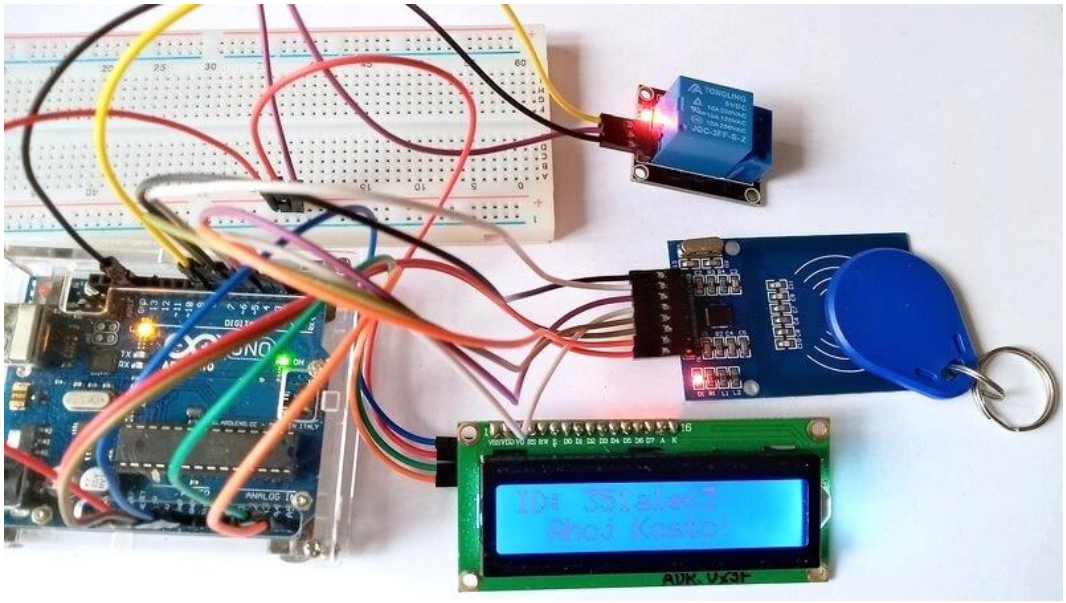
RFID prvku. Je načteno identifikační číslo přiloženého prvku a vypsáno na první řádek LCD displeje. Získané identifikační číslo RFID prvku je v následné kaskádě dvou podmínek postupně porovnáno s dvěma přednastavenými přístupovými kódy (přívěšek a karta). V případě shody s jednou nebo druhou možností je vypsána příslušná hláška a je sepnuto relé. Při přiložení RFID prvku, který nesplnil ani jednu z podmínek nastavených známých identifikačních čísel, je zobrazena hláška pro cizího uživatele. Pochopitelně se relé v tomto případě nespíná.

Tím, že se tyto příkazy vykonávají jen v případě přiložení RFID je i automaticky vyřešena nečinnost systému v době, kdy je v pohotovostním režimu a čeká na přiložení RFID klíče. To je signalizováno vypsáním textu „Vítejte“ první řádce LCD displeje.

**Krok 11:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem 

.

**Krok 12:** Můžeme postupně k RFID čtečce přikládat různé RFID prvku. Pokud je heslo správné, relé sepne a LCD zobrazí příslušnou hlášku – buď „Ahoj Kosto!“ (přiložen přívěšek na klíče), nebo „Zdravime bastlire“ (přiložena karta), relé ovládající zámek sepne. Pokud je přiložen neznámý RFID prvek, systém také zareaguje. Zobrazí se „Ahoj, cizince!“, avšak elektronický zámek, který by byl připojený k relé, zůstane zamčený. Ať již pokus o přístup skončí jakkoliv, o dvě vteřiny později se na displeji opět zobrazí úvodní „Vitejte!“ a přístupový systém je ve výchozím „čekacím“ režimu.



# Něco navíc…

# BONUS – Sonarové čidlo HC-SR04

## Úvod

Experimentální sada Arduino MAXI Starter kit obsahuje spoustu modulů, ze kterých se dá vyrobit mnoho zajímavých zapojení, jedna věc však této sadě chybí a to je ultrazvukový měřič vzdálenosti HC-SR04. Jeho cena se pohybuje kdesi pod hranicí padesáti korun (dle prodejce), takže ho určitě doporučuji dokoupit a do sady si přidat. Jde o modul pro poměrně přesné měření vzdálenosti (s přesností až 3 mm) v rozmezí asi 4 cm až do 4 metrů. Dá se použít téměř ke všemu, funguje na principu odrazu ultrazvuku od překážky.


V této lekci sonarové čidlo HC-SR04 využijeme na postavení jednoduchého modelu hudebního nástroje zvaného theremin. Že nevíte, co to je? Theremin (také těremin) je elektronický hudební nástroj, který vynalezl v roce 1920 Lev Sergejevič Těrmen. Je to hudební nástroj, na který se hraje, aniž by se ho hráč jakkoli dotýkal. Theremin je ovládán pouze pohybem paží, dlaní a prstů. Vzdálenost ruky od příslušné antény ovlivňuje vlastnost zvuku. Hra na theremin je proto výjimečně náročná. Theremin byl kupříkladu použit ve znělce seriálu Vraždy v Midsomeru, kde hraje hlavní melodii.

My jsme si stavbu thereminu vybrali nejen proto, že by s ním mohla být poměrně legrace, ale také abychom kromě ultrazvukového měřiče vzdálenosti HC-SR04 využili i součástku, která sice v sadě Arduino MAXI Starter kit je, ale ani jednou jsme ji v předchozích lekcích nepoužili – pasivní bzučák.

## Použité komponenty

- modul Arduino
- USB kabel
- ultrazvukový měřič vzdálenosti HC-SR04
- pasivní bzučák
- nepájivé pole
- vodiče pro nepájivé pole + vodiče typu „samec-samice“

## Princip

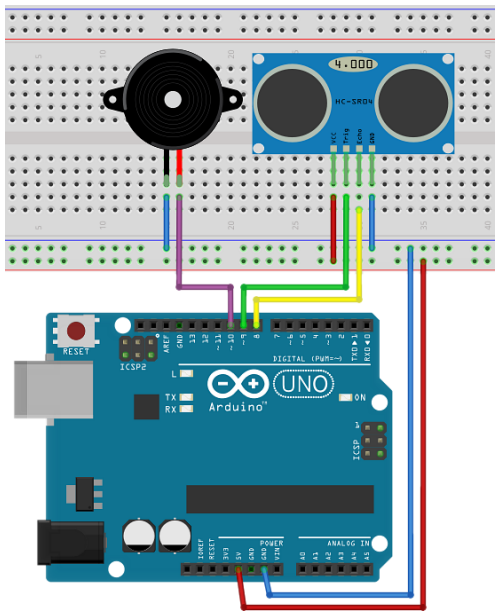
Chceme-li generovat tóny podle toho, jak bude vzdálena ruka od našeho hudebního nástroje, je přesné ultrazvukové čidlo přesně to, co potřebujeme. Samotný princip bude tedy poměrně jednoduchý – změříme vzdálenost překážky od ultrazvukového čidla. Bude-li vzdálenost relativně velká, bude to znamenat, že nad čidlem není žádná ruka a hudebník tedy nehraje. Začne-li se vzdálenost ruky pohybovat v nějaké „rozumné“ vzdálenosti, začneme generovat tóny podle naměřené vzdálenosti. Asi nechceme generovat libovolné frekvence, ale chceme generovat klasické tóny stupnice. Takže si zvolíme určitý interval vzdálenosti (například 10–15 cm) kterému

přisoudíme určitý tón. Pro generování melodie tedy bude stačit vsouvat ruce nad čidlo ve správných vzdálenostech.

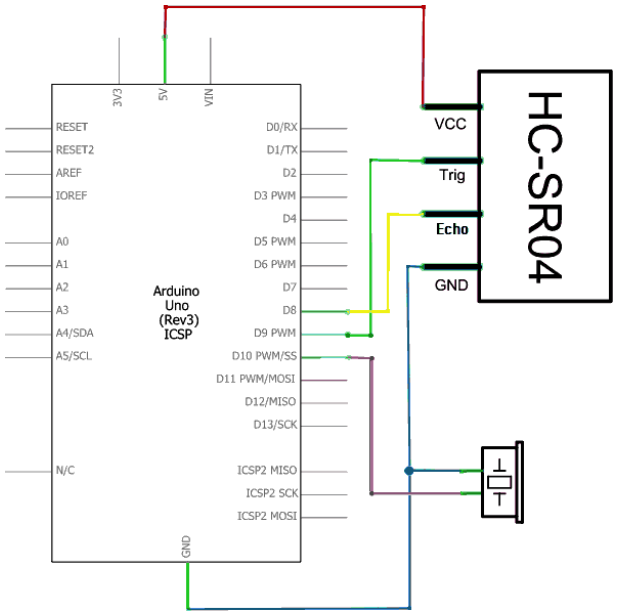
## Postup experimentu

**Krok 1:** Sestavíme si obvod podle následujícího obrázku nebo schématu. Propojení modulu Arduino a ultrazvukového měřiče vzdálenosti HC-SR04 zachycuje tabulka.

**Blokové schéma**

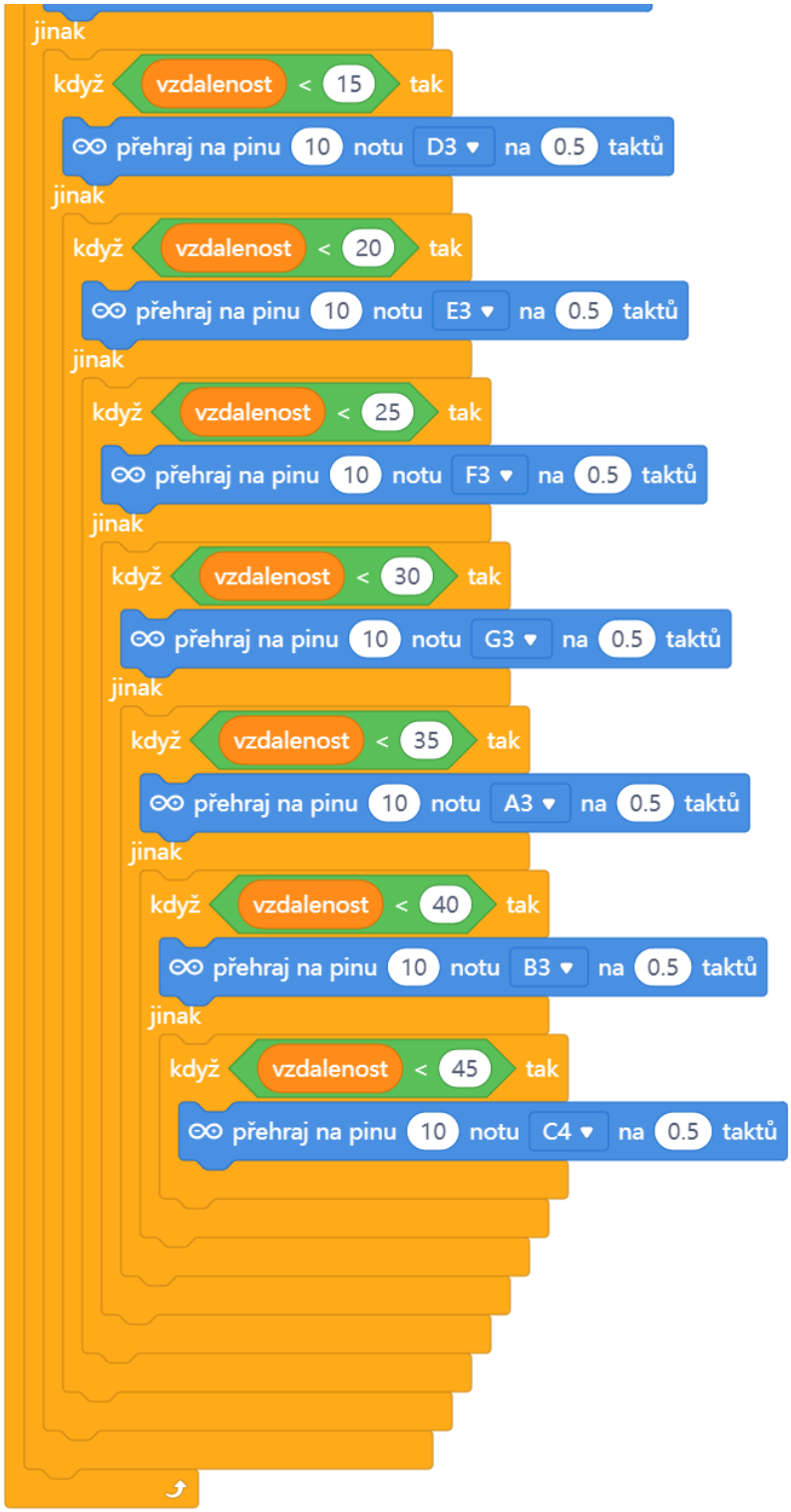


**Elektronické schéma**




| čidlo HC-SR04 | modul Arduino |
|---|---|
| $U_{CC}$ | 5V |
| Trig | 9 |
| Echo | 8 |
| GND | GND |

**Krok 2:** V prostředí mBlock vytvoříme následující program.




Program lze stáhnout z:
http://mblock.fyzika.net/zdrojove-kody/lekce-32/32-sonarove-cidlo-HC-SR04.mblock.

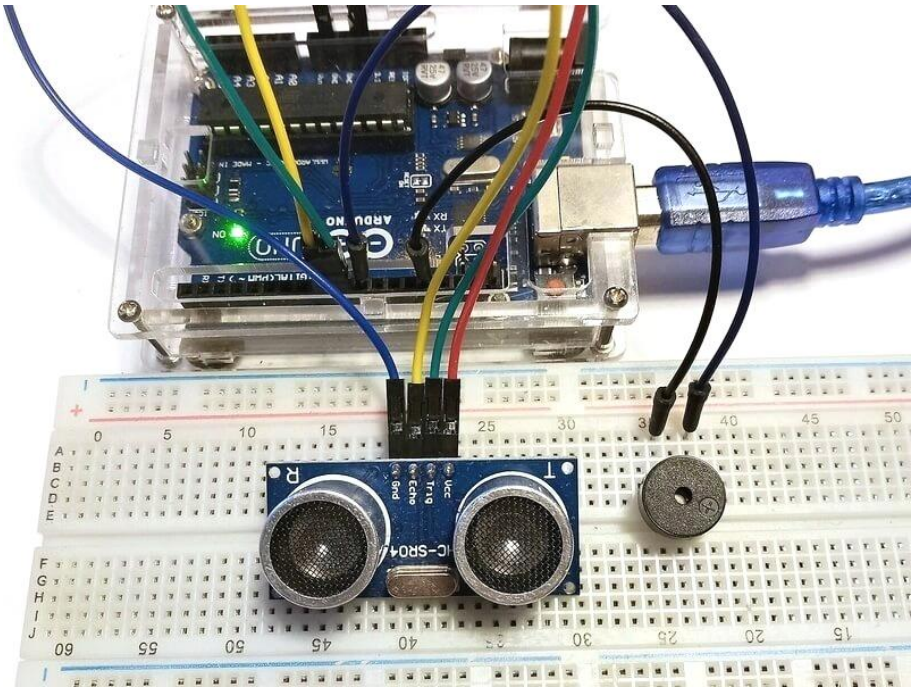
### Vysvětlení kódu

Krása tohoto programu je i ta, že sonarové čidlo a pasivní bzučák jsou součástí robota mBot, pro kterého je prostředí mBlok primárně vyvinuto. Z tohoto důvodu jsou pro obě tyto součástky již z prostředí mBlock

zabudované standardní programové bloky. Obsluha jak sonaru, tak pasivního bzučáku je díky tomu prostě jednoduchá. V nekonečné hlavní smyčce „opakuj stále“ se nejdříve načte vzdálenost od překážky. Následuje kaskáda podmínek, kdy je postupně získaná vzdálenost testována. Například pokud je menší než 10 cm, je na pasivním bzučáku vygenerována nota C3. Pokud je vzdálenost větší následuje další podmínka, která testuje, zda je vzdálenost menší než 15 cm. Jelikož jde o větev podmínky, ve které je jasné, že vzdálenost musela být větší, je při splnění podmínky vzdálenost z rozmezí 10–15 cm. Pro rozmezí těchto vzdáleností je nastaveno pípnutí s frekvencí odpovídající notě D3. Tímto způsobem jsou vzdálenost rozděleny do pěticentimetrových intervalů až po notu C4 (maximální vzdálenost 45 cm).

**Krok 3:** Zkompilujeme kód a nahrajeme do modulu Arduino tlačítkem Nahrát.

**Krok 4:** Pokud začneme pohybovat dlaní před sonarovým čidlem, měly by se ozývat jednotlivé tóny. Z počátku to asi nebude žádná serenáda, ale po troše tréninku by se z toho mohli třeba „Ovčáci čteráci“ podařit. A kdoví, třeba časem se to dá i poslouchat.



**Programování modulu Arduino v prostředí mBlock**
*pomocí sady „Arduino MAXI Starter kit“*

Vydaly webové stránky *Fyzikální kabinet FyzKAB* – http://kabinet.fyzika.net/
Text, fotografie a schémata: společnost LaskaKit a RNDr. Miroslav Panoš, Ph.D.
Zdrojové kódy v prostředí mBlock: RNDr. Miroslav Panoš, Ph.D.
Grafická úprava: FyzKAB
Jazyková korektura: nikdo ☹

236 stran



---

*aktuální verze*: β_6-2 (2024-01-20)