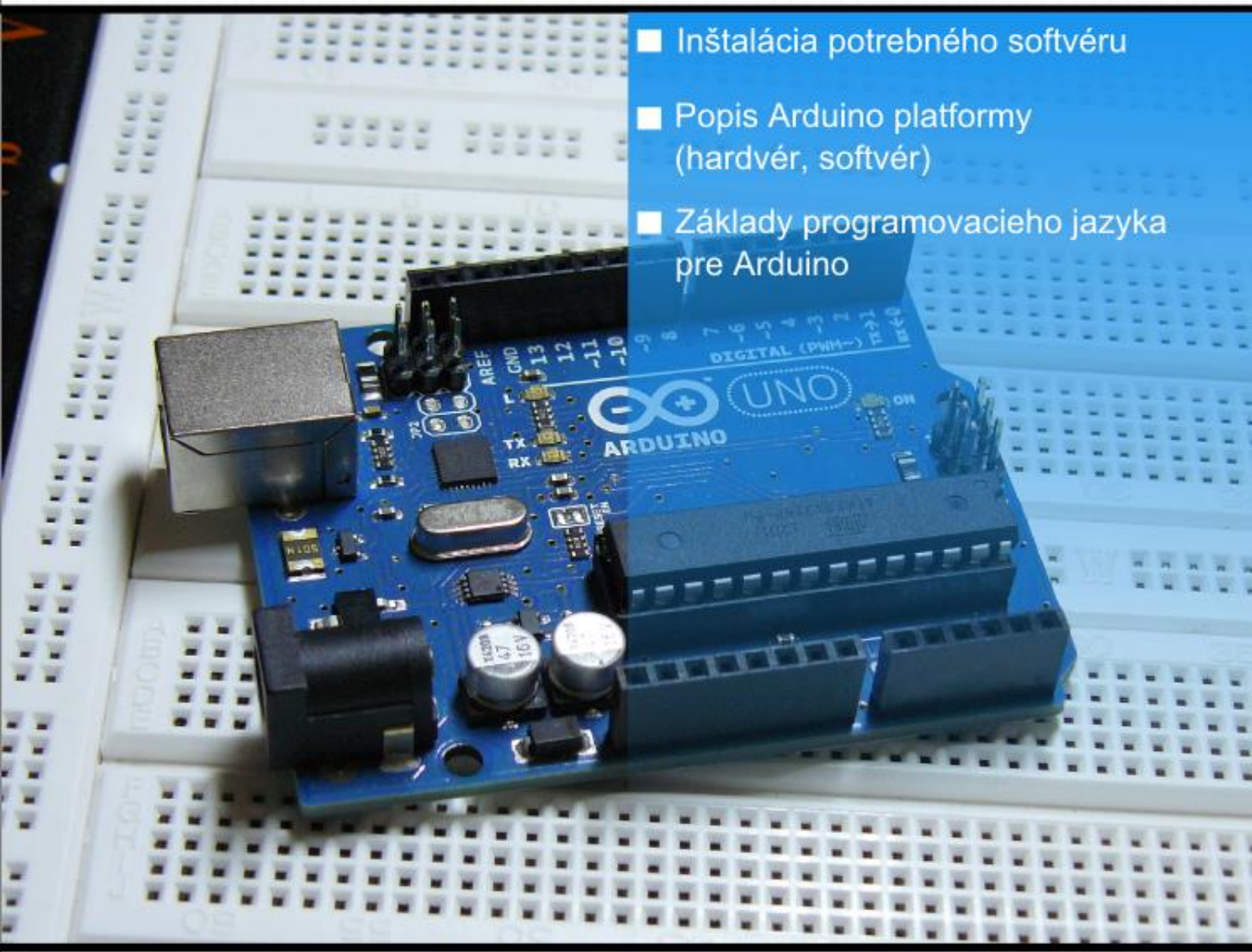
{mb();}

Miroslav Božík

Ako naprogramovať ARDUINO

bez predchádzajúcich znalostí



- Inštalácia potrebného softvéru
- Popis Arduino platformy (hardvér, softvér)
- Základy programovacieho jazyka pre Arduino

{mb();} /* Miro Božík */

Ako naprogramovať Arduino bez predchádzajúcich znalostí.

Ebook o tom ako začať programovať a vytvárať elektronické zariadenia založené na známej platforme Arduino.

Vyhlásenie

Obsah

Úvod

Nebudem vás trápiť dlhým úvodom to nie je môj štýl :). Takže v krátkosti. Dôvod na napísanie tohto ebooku bol nasledovný: Chcel som dať dokopy kvalitné a ucelené informácie pre začiatočníkov o programovaní elektronickej platformy Arduino.

Pre koho je kniha určená?

Kniha je určená predovšetkým začiatočníkom, ktorý sa chcú zoznámiť s Arduinom, naučiť sa programovať a navrhovať vlastné elektronické zariadenia.

Čo Vás čaká v knihe?

V knihe Vám bude predstavená platforma Arduino. Postupne krok za krokom vás povedie inštaláciou softvéru, zapojením a programovaním Arduina.

Čo je to Arduino?

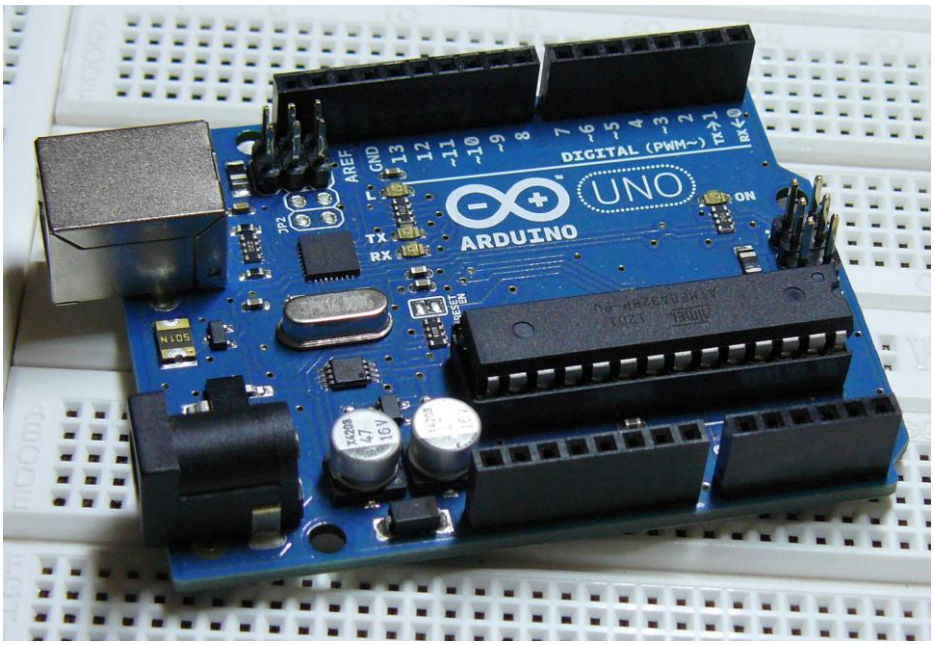
Arduino je úžasný **open-source projekt**, konkrétne sa jedná o **elektronickú platformu**. Keďže je Arduino open-source projekt sú dostupné **elektronické schémy** zapojenia Arduino dosky, taktiež zdrojové kódy knižníc a vývojového prostredia. Open-source hovorí aj o tom, že tieto zdroje môžete ďalej upravovať a šíriť alebo predávať ale už nie pod rovnakým názvom, teda už sa to nemôže volať Arduino. A čo znamená „elektronická platforma"? Platforma je niečo na čom môžete postaviť vlastný projekt, niečo čo vám dá základ, prostriedky (knižnice, vývojové prostredie), nejakú „predlohu" podľa ktorej budete postupovať. A čo všetko táto platforma zahrňuje? Základom je **vývojová doska** (plošný spoj - hardvér) a **vývojové prostredie** (teda softvér tzv. IDE - **Integrated Development Environment**).

Čo Arduino dokáže ?

Arduino môže získavať údaje z prostredia pomocou rôznych **senzorov** a na základe toho **ovládať** nejaké iné zariadenie napr. motory, osvetlenie, atď. Arduino projekt môže fungovať **samostatne** alebo môže byť **ovládaný** nejakou **aplikáciou** z vášho počítača, smartfónu alebo tabletu.
Pri jednoduchých projektoch, nie je nutné dokonca ani spájkovanie.

Čo k tomu potrebujeme ?

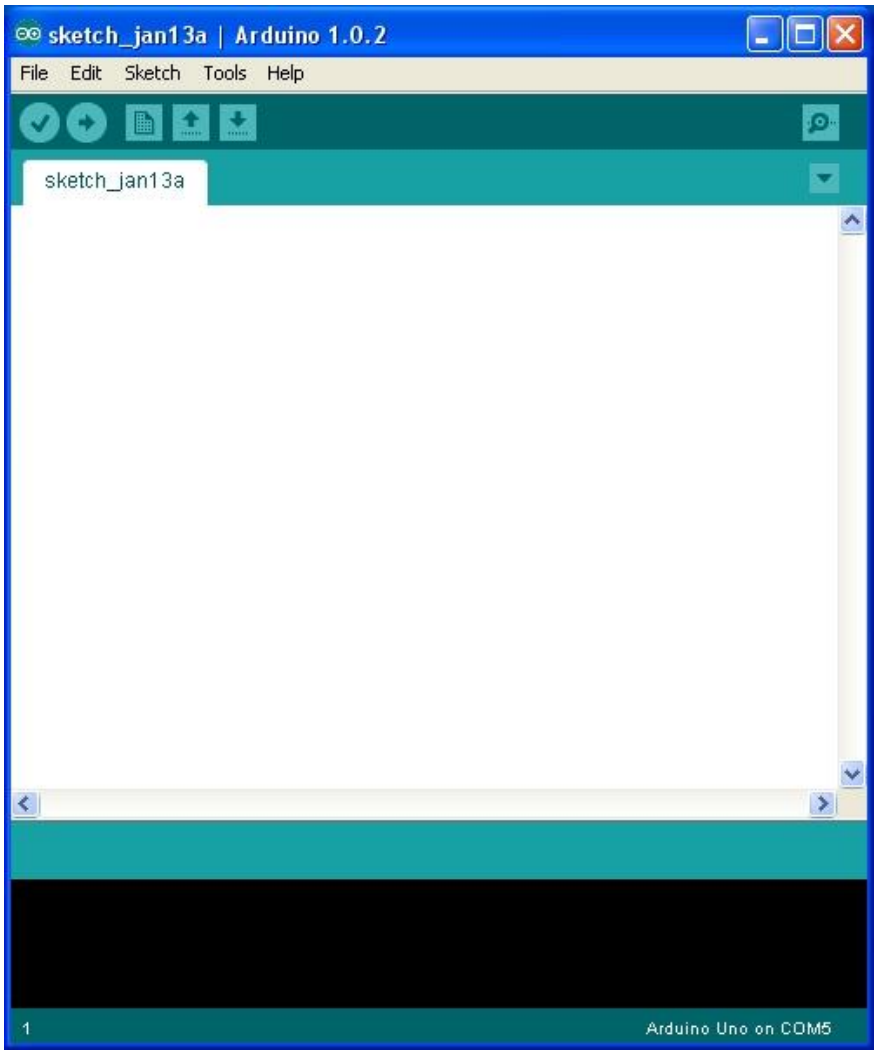
Na to aby sme mohli začať vytvárať nejaké jednoduché elektronické projekty s Arduinom potrebujeme **Arduino vývojovú dosku** (Obrázok 0-1), ktorú si je nutné zakúpiť (distribútorov pre Slovensko môžete nájsť na stránke http://arduino.cc v sekcii Buy http://arduino.cc/en/Main/Buy), **USB kábel (A-B)** na prepojenie Arduina s počítačom, **kontaktné pole** (Obrázok 0-2), nainštalované **vývojové prostredie** (obrázok 0-3, ďalej len IDE) aby sme mohli napísať a nahrať program do Arduina a samozrejme chuť sa niečo naučiť ale to vám pravdepodobne nechýba keď čítate túto knihu :). K jednotlivým projektom, ktoré budú v tejto knihe bude treba ešte zopár ďalších súčiastok.



Obrázok 0-1. Doska Arduino UNO R3

Obrázok 0-2. Kontaktné pole

Obrázok 0-3. USB kábel A-B



Obrázok 0-4. Arduino IDE - vývojové prostredie

Čítajte ďalšiu kapitolu ako nainštalovať potrebný softvér.

1. Inštalácia softvéru

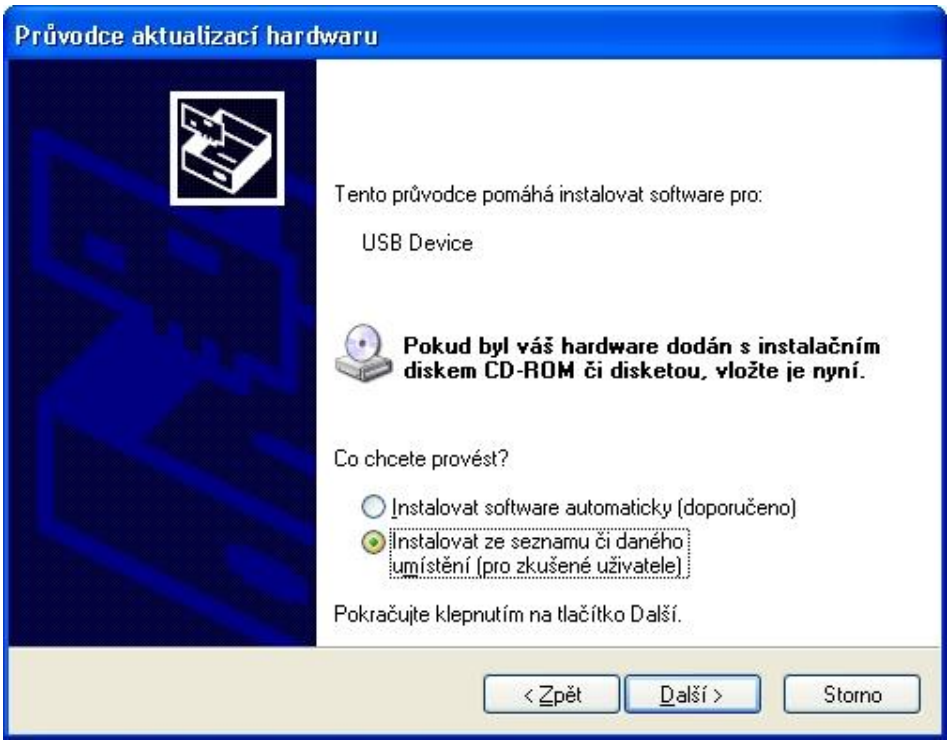
Tak konečne začíname s inštaláciou softvéru. Všetok potrebný softvér nájdeme na stránke Arduina http://arduino.cc v sekcii Download (http://arduino.cc/en/Main/Software). Na tejto stránke si môžeme vybrať typ inštalačného súboru podľa toho pod akým operačným systémom chceme Arduino programovať. Vývojové prostredie pre Arduino je multiplatformové tzn. že je ho možné spustiť na všetkých známych operačných systémoch ako Microsoft Windows, Linux alebo Mac OS.
V nasledujúcich podkapitolách si ukážeme ako nainštalovať ovládače pre Arduino UNO R3 (alebo inú verziu Arduina).

1.1. Inštalácia na Windows XP

Po stiahnutí inštalačného súboru **arduino-1.0.2-windows.zip** (v dobe písania tejto knihy bola aktuálna verzia Arduino IDE 1.0.2, vo vašom prípade sa toto číslo môže líšiť), túto následne rozbalíme (rozzipujeme). Ak otvoríme rozbalený priečinok **arduino-1.0.2/** nájdeme ďalšie priečinky a súbory, medzi ktorými nájdete aj **arduino.exe**, čo je spúšťací súbor pre IDE.

Obrázok 1-1

Ďalej pripojíme Arduino dosku pomocou USB kábla do PC. Na Arduino doske by mala svietiť **zelená LED** označená **PWR**, ktorá indikuje napájanie Arduina. Windows nájde nový hardvér (obrázok 1-1) a následne sa otvorí okno „**Sprievodca novo rozpoznaným hardvérom**“, v tomto okne uvidíte otázku či sa má Windows pripojiť k Windows update webu, na výber máme tri možnosti, my teraz vyberieme možnosť: „Nie, teraz nie“ a klikneme na tlačítko „Ďalej“. V ďalšom okne máme na výber dve možnosti, vyberieme druhú možnosť podľa obrázka 1-2 a zase klikneme na tlačítko „Ďalej“.



Obrázok 1-2. Sprievodca nainštalovaním nového hardwaru.

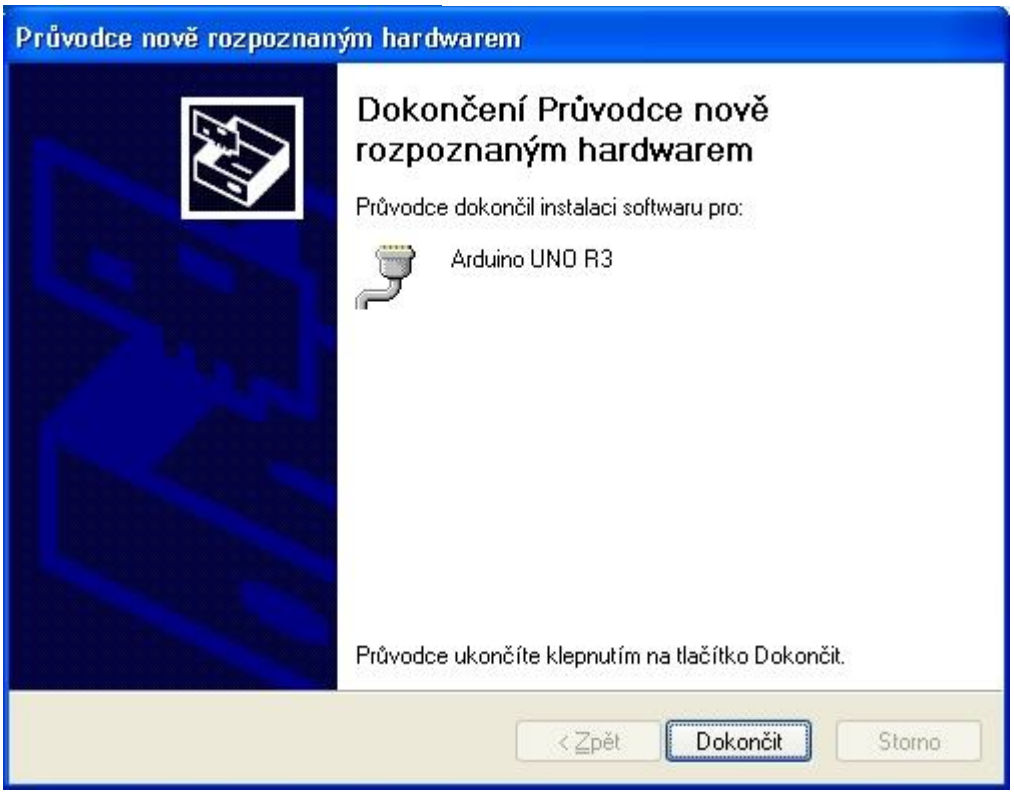
V ďalšom kroku vyberieme cestu k priečinku drivers, ktorý sa nachádza v rozbalenom Arduino IDE priečinku v našom prípade arduino-1.0.2 , viď obrázok 1-3.

Obrázok 1-3. Sprievodca inštaláciou hardvéru na Windows XP - výber umiestnenia ovládačov.

Po kliknutí na tlačítko „Ďalej“, Windows vyhľadá potrebný ovládač a nainštaluje ho. Počas inštalácie vás môže Windows upozorniť na to, že softvér pre tento hardvér neprešiel testom pre získanie loga systému Windows (viď obrázok 1-4). Ovládač nijak nepoškodí váš systém a je nutný, aby počítač vedel komunikovať s Arduinom. Preto sa nemusíte báť a môžete kliknúť na tlačítko „Pokračovať“.

Po inštalácii sa zobrazí okno sprievodcu s informáciou o úspešnej inštalácii softvéru ovládača (obrázok 1-5). Už stačí len kliknúť na „Dokončiť“ a inštaláciu máme úspešne za sebou ☺.

Obrázok 1-4. Upozornenie počas inštalácie hardvéru na Windows XP systéme

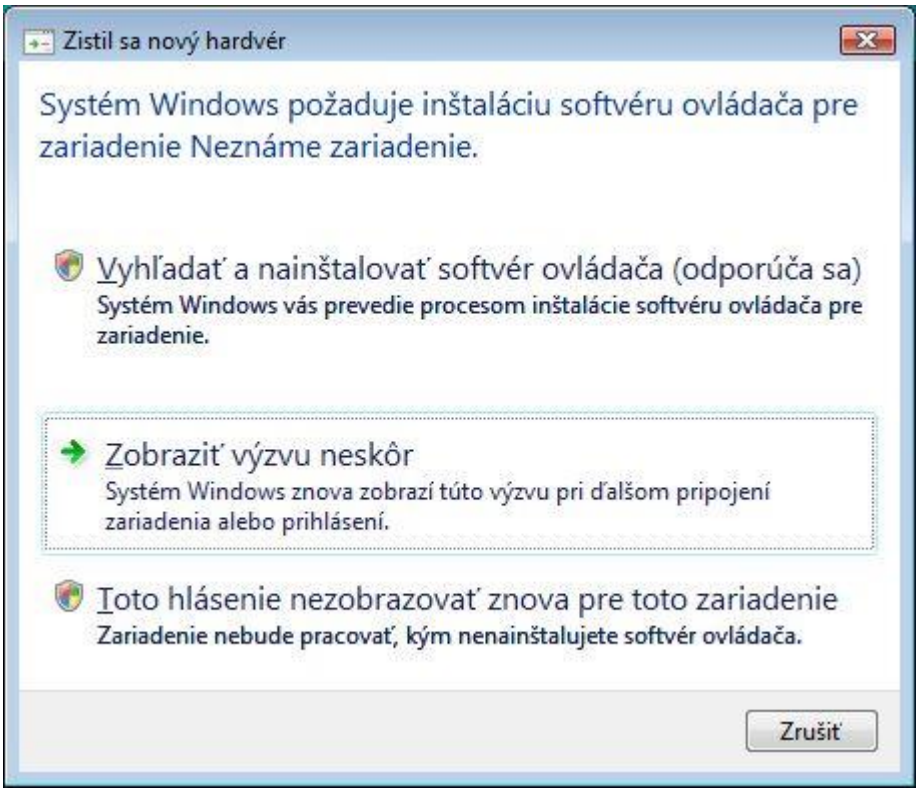


Obrázok 1-5. Sprievodca inštaláciou hardvéru - úspešné dokončenie inštalácie.

1.2. Inštalácia na Windows Vista/7

Po stiahnutí inštalačného súboru **arduino-1.0.2-windows.zip** (v dobe písania tejto knihy bola aktuálna verzia Arduino IDE 1.0.2, vo vašom prípade sa toto číslo môže líšiť), túto následne rozbalíme (rozzipujeme). Ak otvoríme rozbalený priečinok **arduino-1.0.2/** nájdeme ďalšie priečinky a súbory, medzi ktorými nájdete aj **arduino.exe**, čo je spúšťací súbor pre IDE.

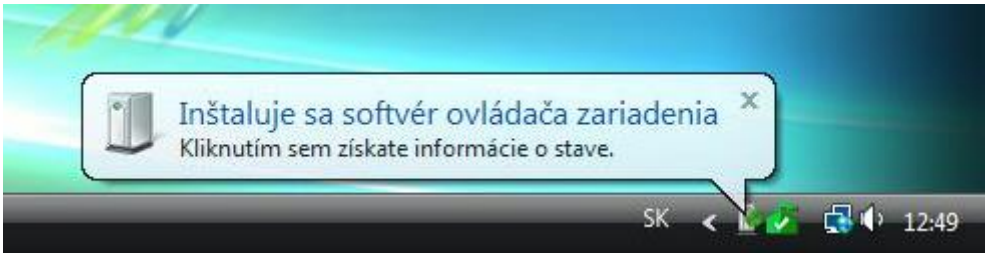
Ďalej pripojíme Arduino dosku pomocou USB kábla do PC. Na Arduino doske by mala svietiť zelená **PWR LED**, ktorá indikuje napájanie Arduina. Vo Windows sa otvorí nové okno „Zistil s nový hardware" na výber máte tri možnosti ako vidieť na obrázku 1-6.



Obrázok 1-6. Zistil sa nový hardware v MS Windows Vista / 7

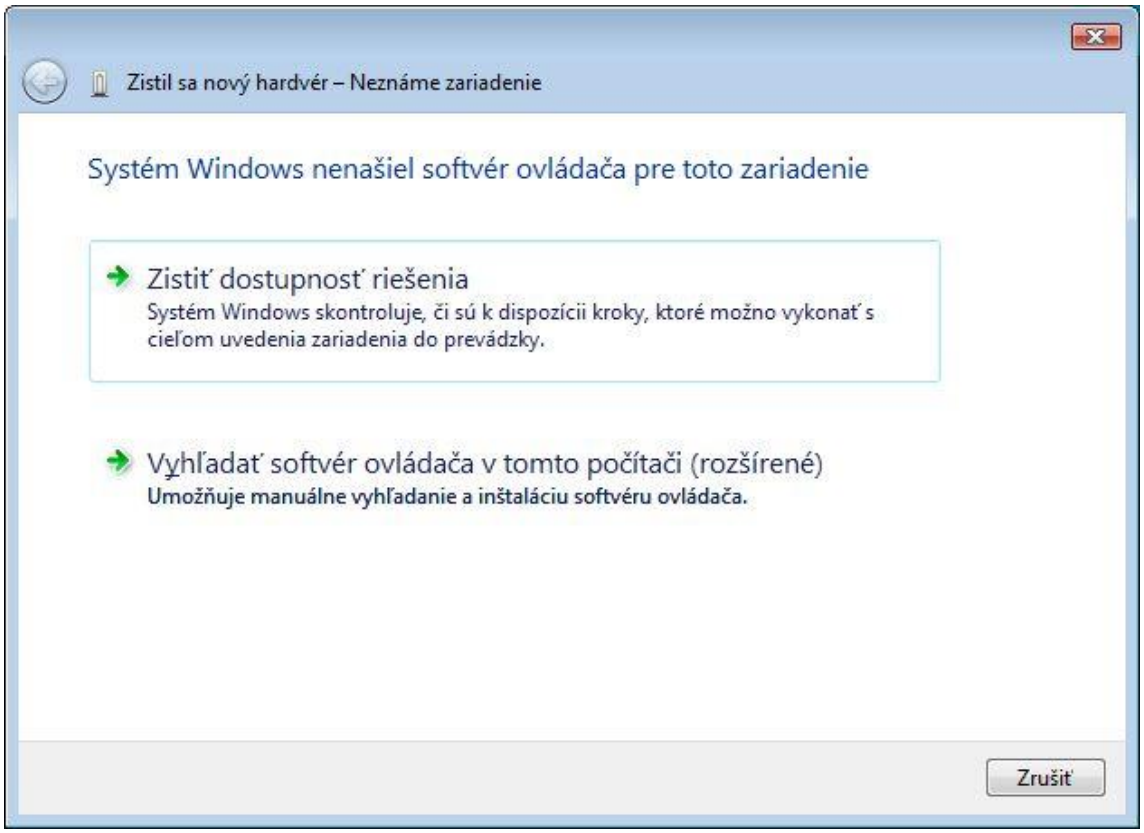
Vyberiete **prvú** možnosť **„Vyhľadať a nainštalovať softvér ovládača (odporúča sa)".**

Windows sa teraz pokúsi nájsť potrebný ovládač a nainštalovať ho (obrázok 1-7).



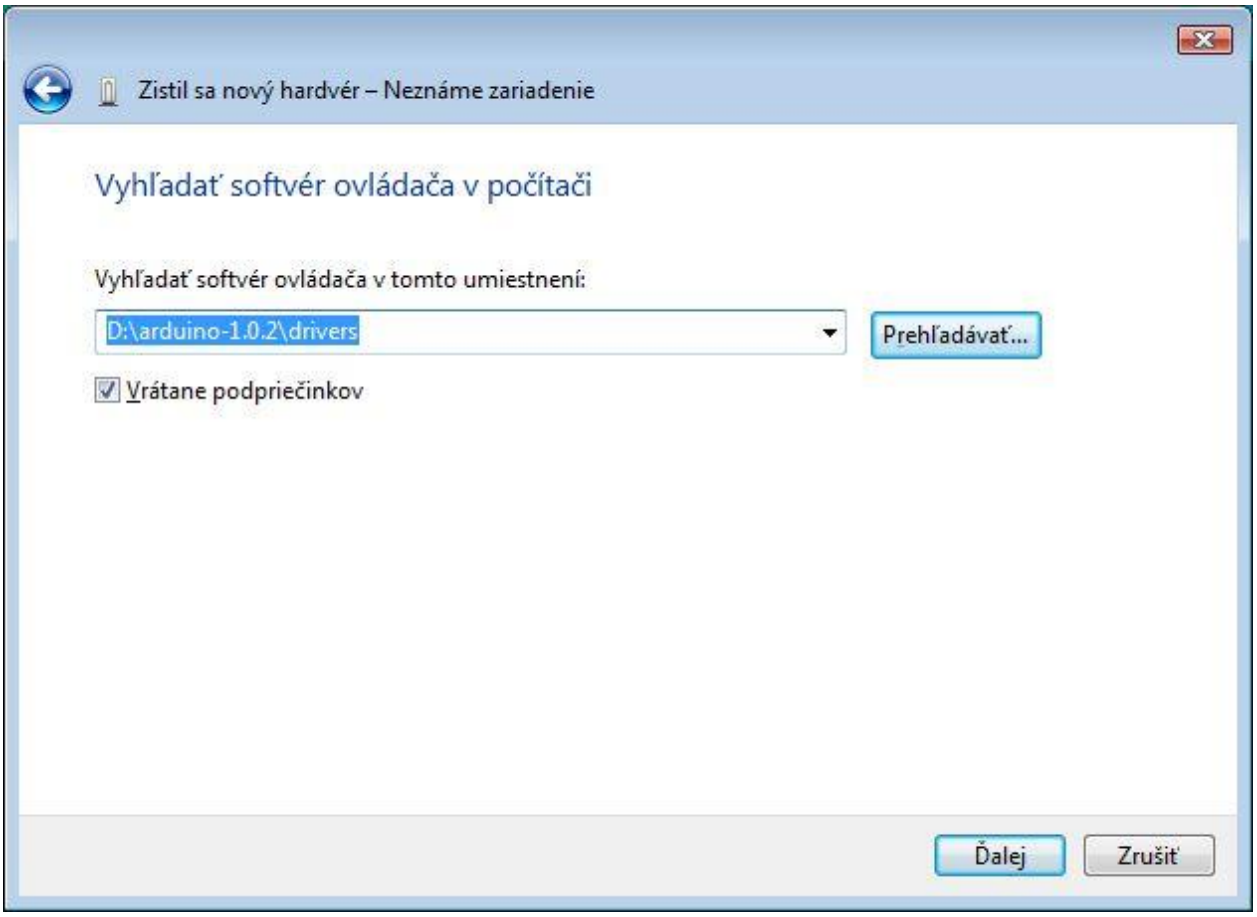
Obrázok 1-7. Pokus o nainštalovanie ovládača na Windows Vista / 7

Po chvíli Windows nahlási upozornenie, že sa nepodarilo nájsť potrebný ovládač(obrázok 1-8). Nebojte sa to je v poriadku.



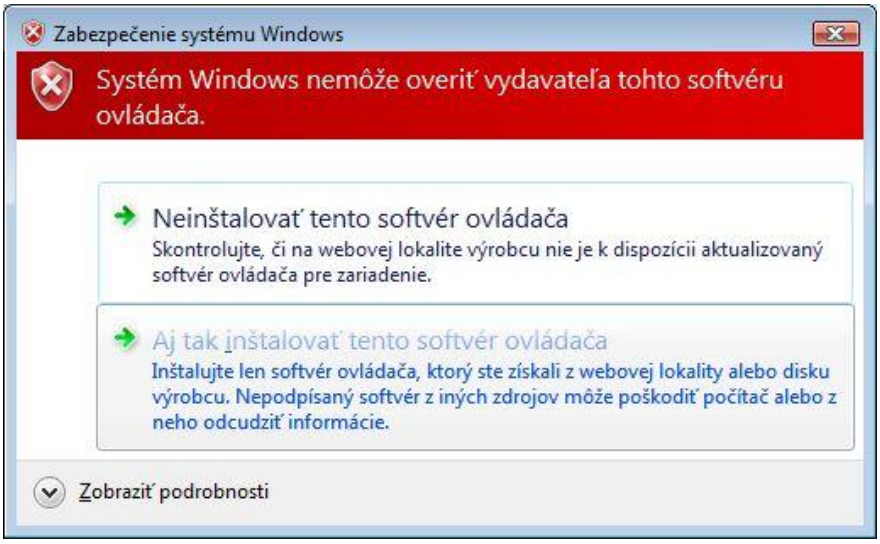
Obrázok 1-8. Upozornenie na Windows Vista / 7 , softvér ovládača sa nepodarilo nájsť.

Teraz vyberiete druhu možnosť **„Vyhľadať softvér ovládača v tomto počítači (rozšírené)“.** Zobrazí sa vám nové okno kde zadáte cestu k priečinku **drivers** v rozbalenom Arduino IDE priečinku (v našom prípade arduino-1.0.2) viď obrázok 1-9.



Obrázok 1-9. Vyhľadanie ovládača v počítači, v konkrétnom umiestnení.

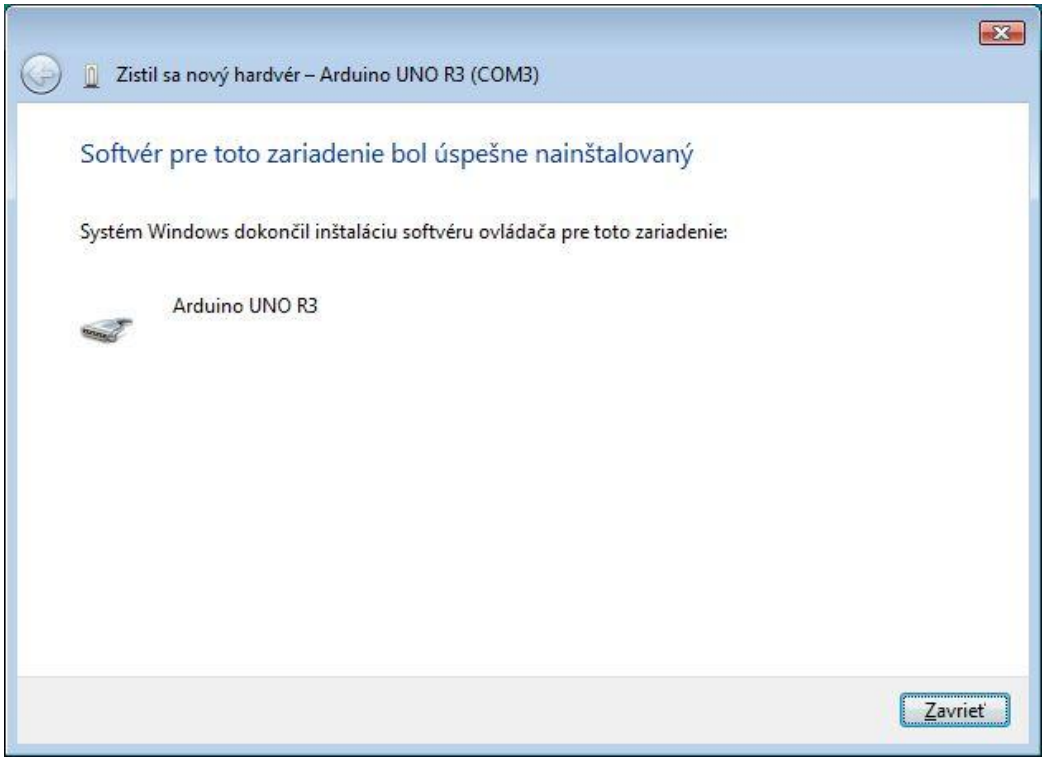
Klikneme na tlačítko „Ďalej“ a Windows začne vyhľadávať ovládač v zadanom priečinku. Ak bude ovládač nájdený otvorí sa okno s upozorneným viď obrázok 1-10.



Obrázok 1-10. Bezpečnostné upozornenie pred inštaláciou softvéru ovládača vo Windows Vista / 7

Nemusíte sa báť nejde o žiadny škodlivý softvér, Windows len nemôže/nevie overiť vydavateľa tohto softvéru. Vyberieme možnosť **„Aj tak inštalovať tento softvér ovládača“**.

Po úspešnej inštalácii (viď obrázok 1-11) , klikneme na tlačítko „zavrieť“ a inštaláciu máme úspešne za sebou ☺.



Obrázok 1-11. Úspešné nainštalovanie softvéru ovládača.

1.3. Inštalácia na Windows 8

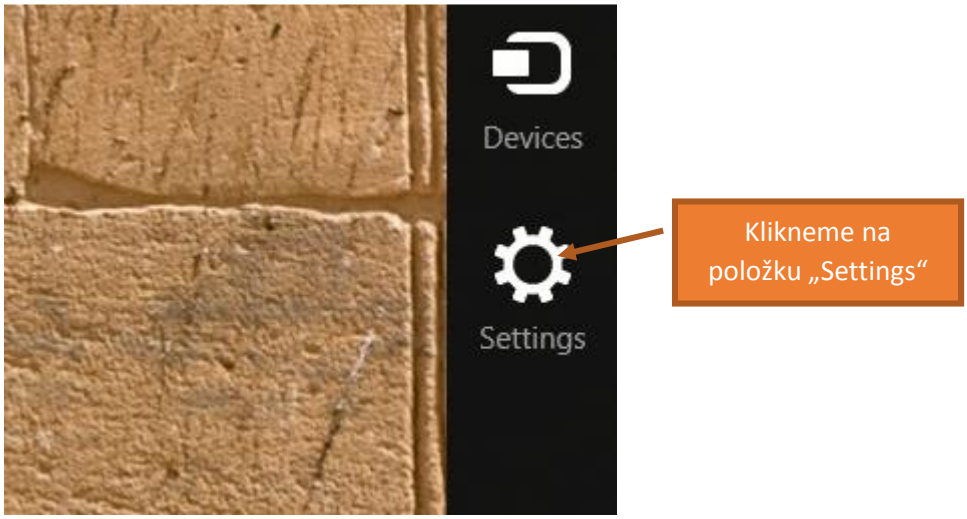
Po stiahnutí inštalačného súboru **arduino-1.0.2-windows.zip** (v dobe písania tejto knihy bola aktuálna verzia Arduino IDE 1.0.2, vo vašom prípade sa toto číslo môže líšiť), túto následne rozbalíme (rozzipujeme). Ak otvoríme rozbalený priečinok **arduino-1.0.2/** nájdeme ďalšie priečinky a súbory, medzi ktorými nájdete aj **arduino.exe**, čo je spúšťací súbor pre IDE, o ktorom si povieme v ďalšej kapitole.

Ďalej pripojíme Arduino dosku pomocou USB kábla k PC. Na doske by sa mala rozsvietiť zelená **PWR LED**, ktorá indikuje napájanie Arduina.

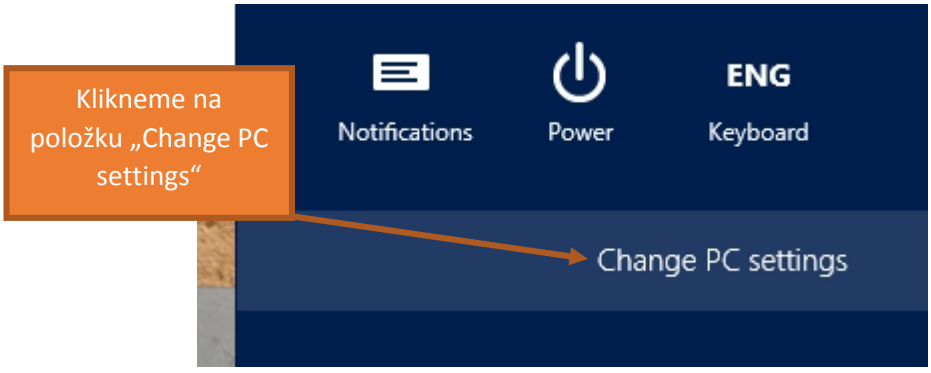
Inštalácia pod Windows 8 je veľmi podobná ako vo Windows Vista alebo 7, ale je tu jeden zásadný rozdiel. Windows 8 má štandardne zakázané inštalovať ovládače, ktoré nie sú podpísané, tzn. budeme musieť najprv povoliť v nastaveniach aj inštaláciu takýchto nepodpísaných ovládačov.

Takže ideme nato:

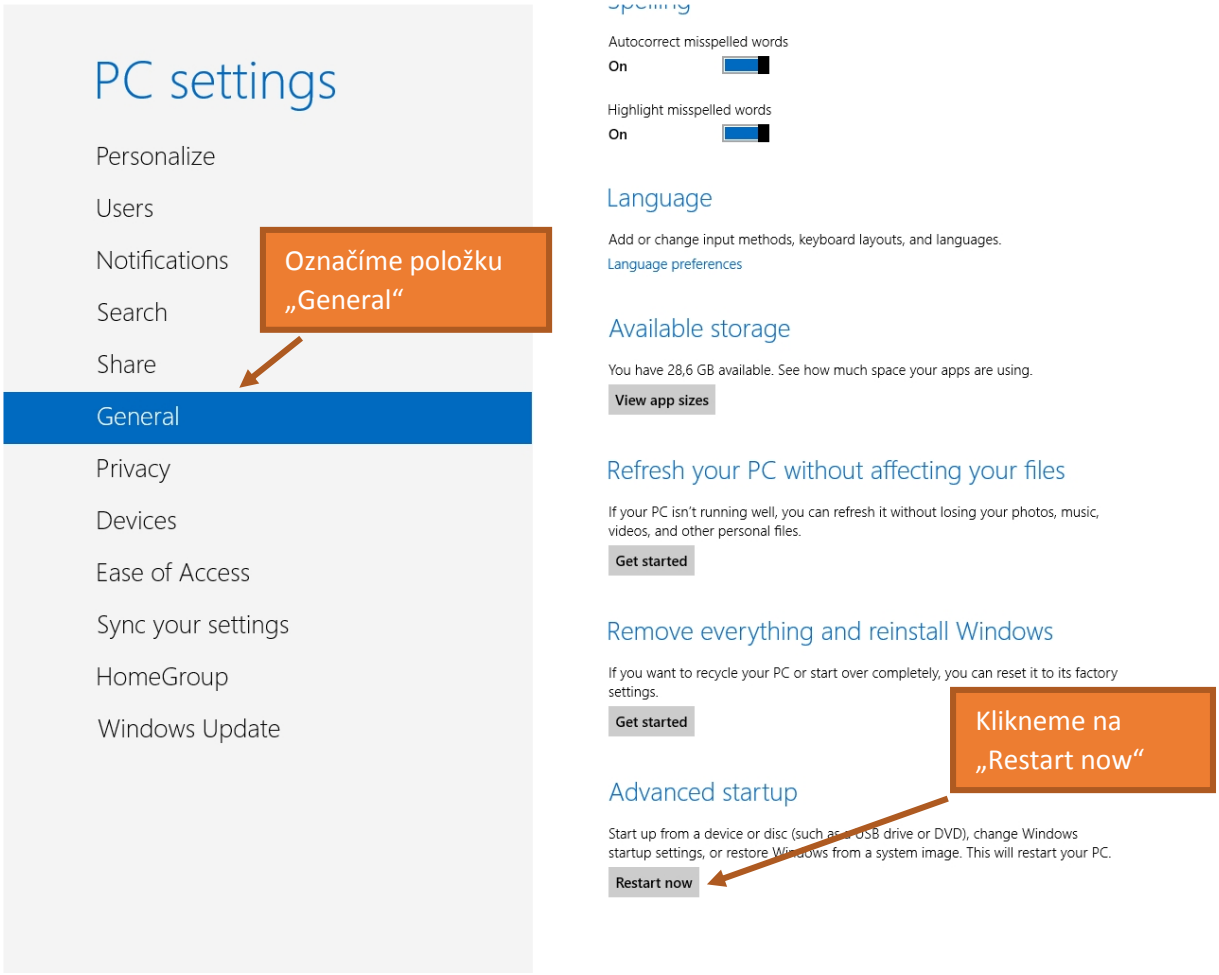
1. **krok** – ak prejdeme myšou do pravého horného rohu, objavíme sa nám menu, v ktorom klikneme na položku **Settings** (ak máte iný jazyk vo Windows bude názov položky samozrejme v inom jazyku).



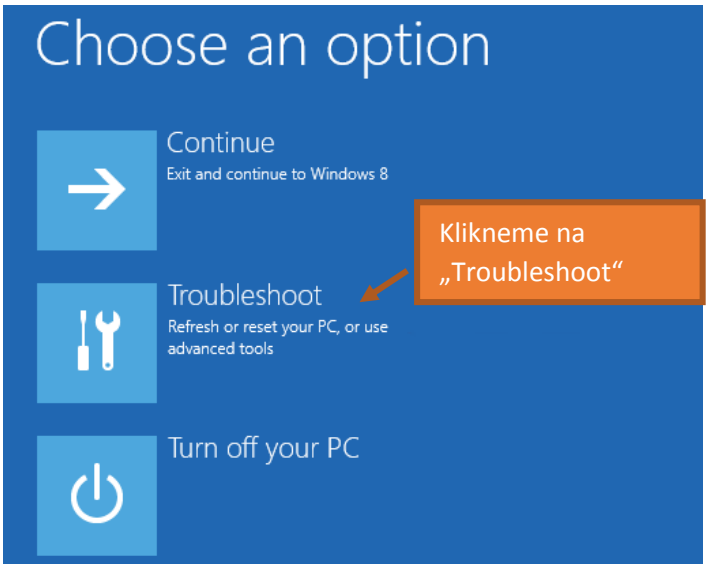
2. **krok** – objaví sa nám ďalšie menu v ktorom klikneme na **„Change PC settings“**.

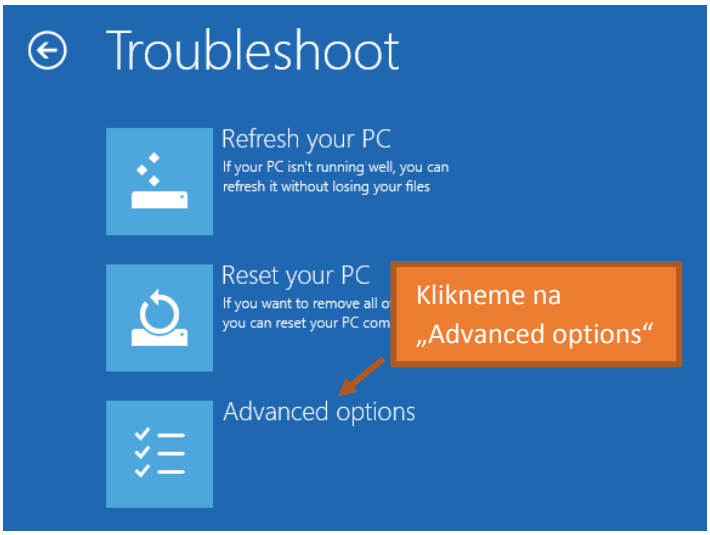


3. **krok** – teraz sme sa dostali do nastavení počítača (Windows). Vyberieme položku **General** a následne klikneme na **Restart now** v submenu **Advanced startup**. Počítač sa nám teraz bude reštartovať, preto je lepšie sa uistiť, že máte svoju prácu na PC uloženú.



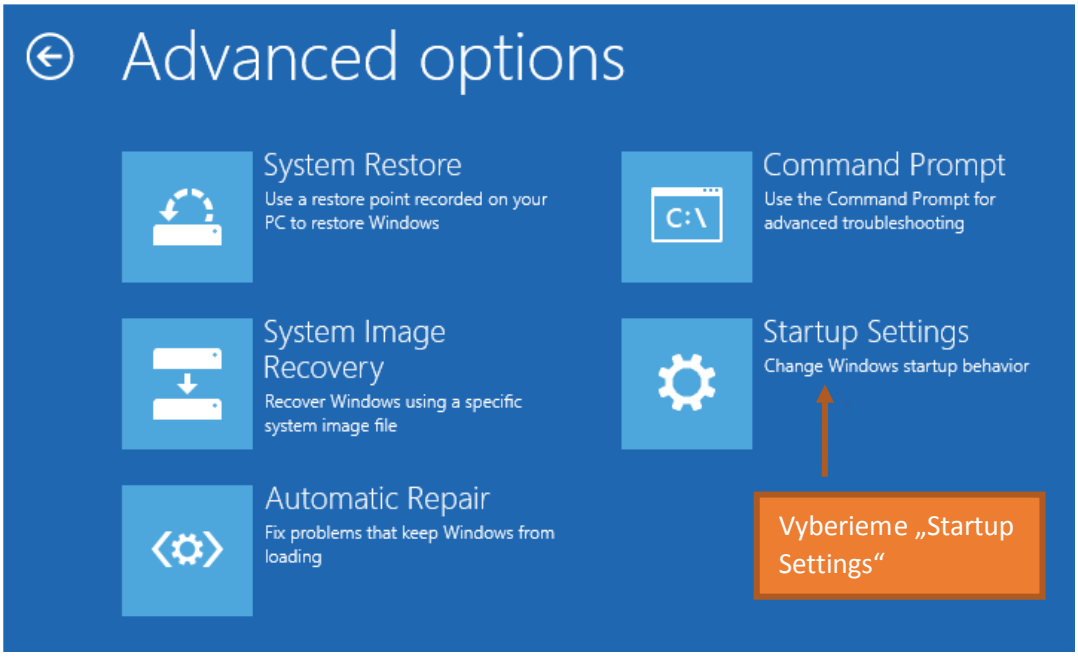
4. krok – pri spúšťaní PC vyberte teraz z menu položku **Troubleshoot** (Riešenie

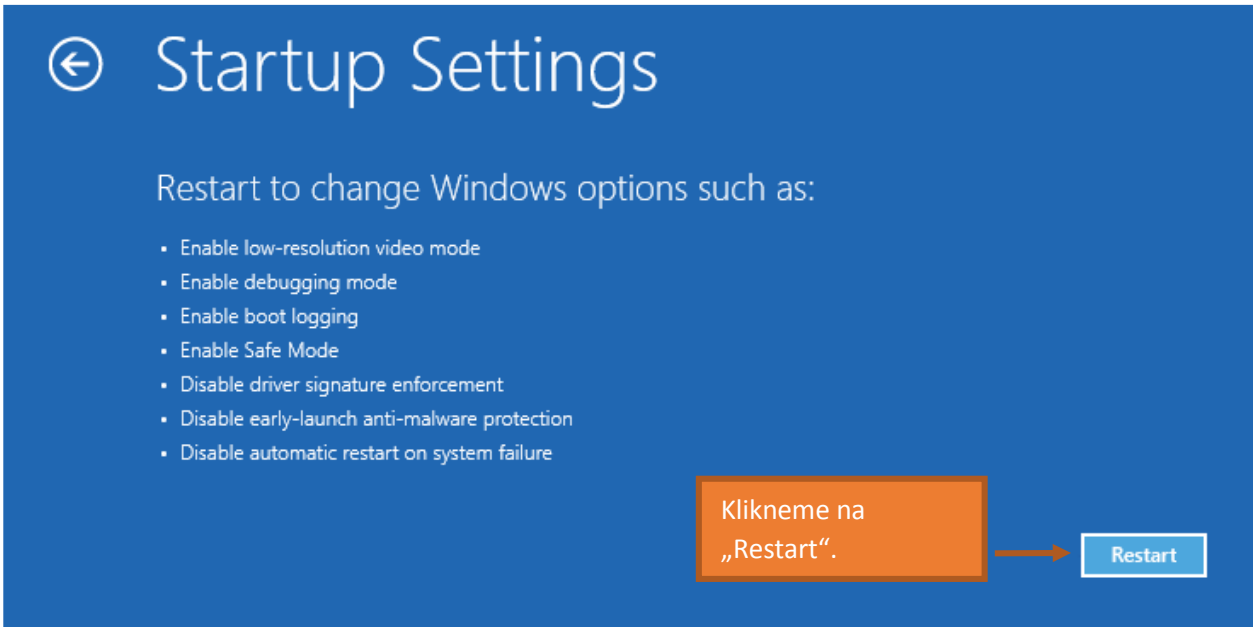




problémov) na obrazovke „**Choose an option**“. Na obrazovke **Troubleshoot** vyberieme **Advanced options**.

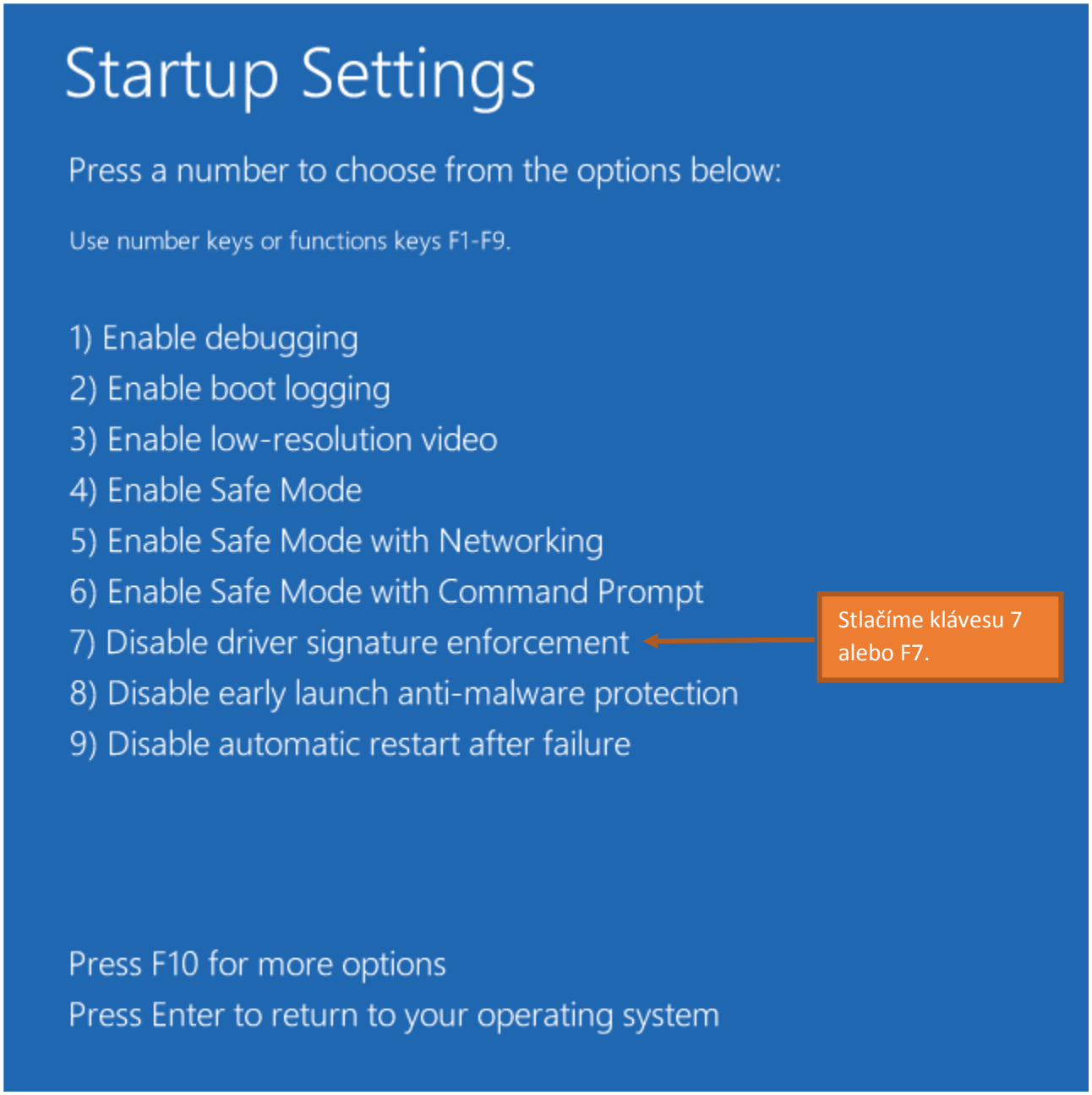
5. krok – na obrazovke **Advanced options** vyberieme **Startup settings** a potom **Restart**.





Pokračujeme na ďalšej strane..

6. krok - Teraz máme na obrazovke Startup settings, rôzne položky očíslované od 1 po 9, stlačte klávesu **7 alebo F7**, čím vyberiete možnosť **„Disable driver signature enforcement“** zo zoznamu.



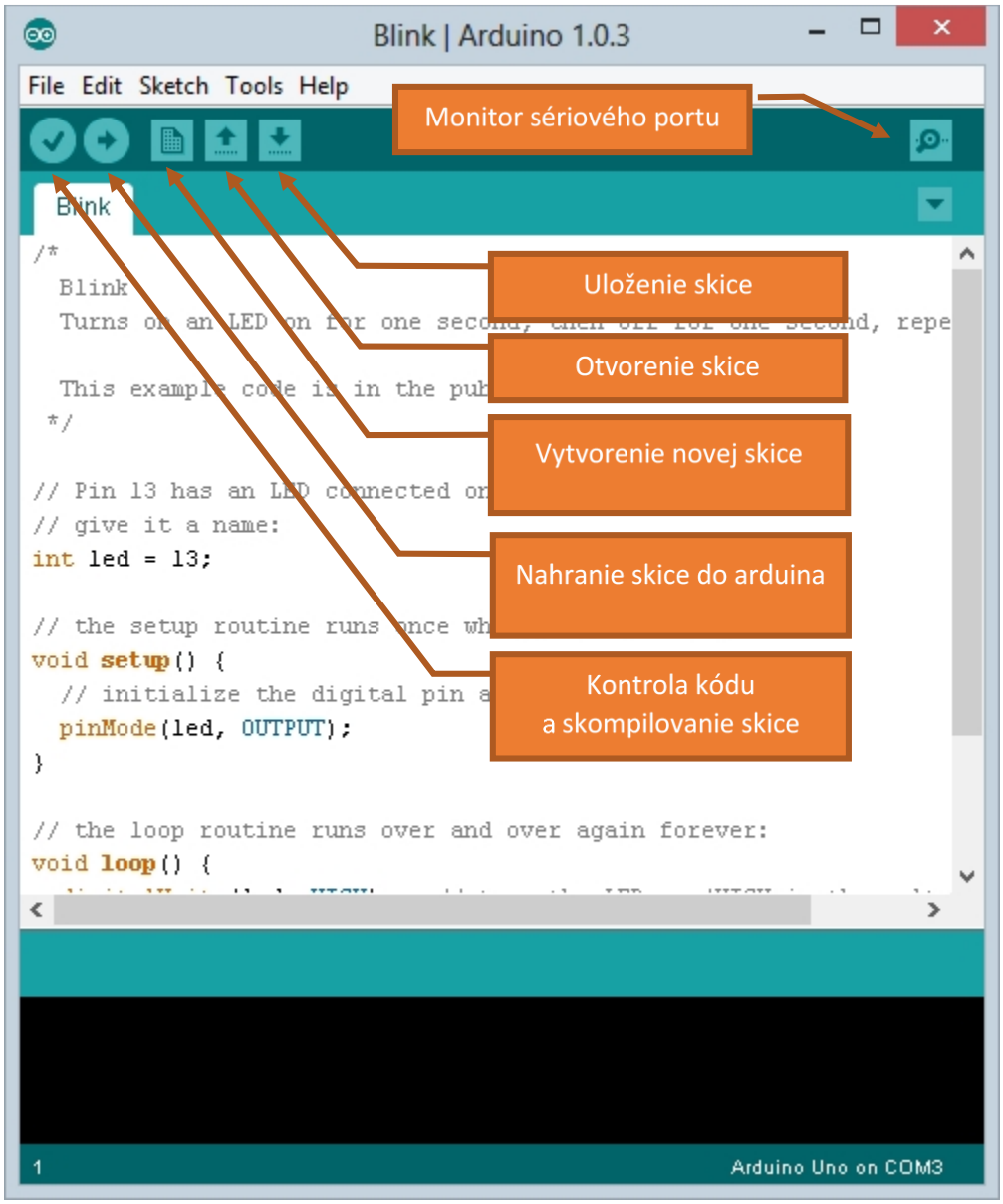
Počítač sa teraz reštartne. Po spustení Windows môžete nainštalovať ovládače pre Arduino UNO R3 rovnakým spôsobom ako pre Windows Vista/7.

2. Arduino – prehľad softvéru a hardvéru

V tejto kapitole si popíšeme Arduino IDE. Povieme niečo o programovacom jazyku pre Arduino. Vysvetlíme si tiež štruktúru programu pre Arduino.

2.1. Vývojové prostredie pre Arduino

Ak spustíme Arduino IDE (súbor arduino.exe v rozbalenom priečinku arduino-1.0.2) naskytne sa nám pohľad podobný ako na obrázku 2-1.

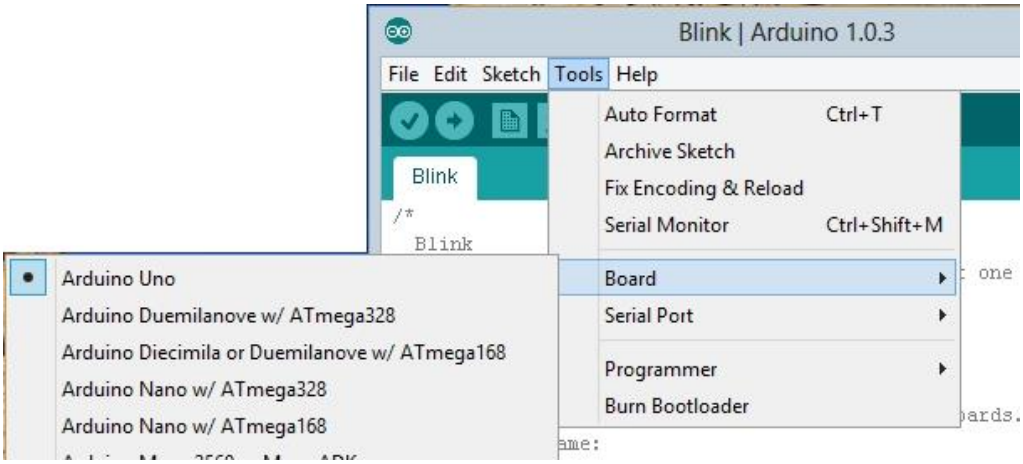


Obrázok 2-1. Arduino IDE, verzia 1.0.2

Na obrázku sú popísané jednotlivé tlačítka (zľava doprava) kontrola a kompilovanie (Verify), nahranie (Upload), Nová skica (New), Otvoriť (Open) a Uložiť (Save).

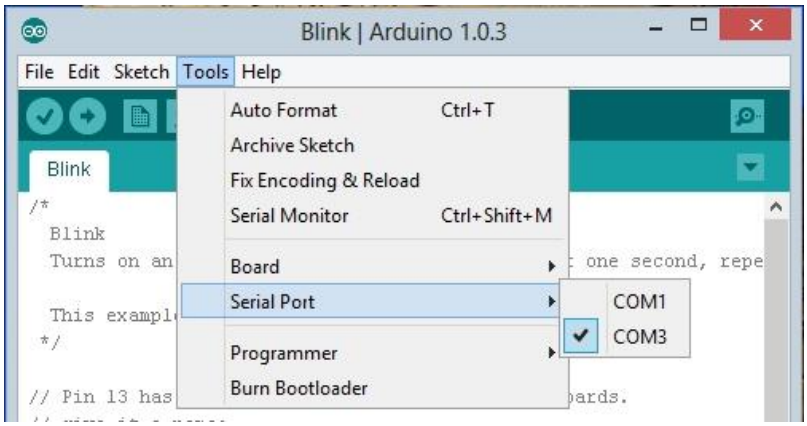
2.2. Nastavenie vývojového prostredia

Aby sme mohli program nahrať do Arduina, musíme správne nastaviť vývojové prostredie. Na obrázku 2-2 je znázornené nastavenie správnej verzie Arduino dosky, tzn. vyberieme „Arduino UNO“, ktoré budeme v projektoch používať.



Obrázok 2-2. Nastavenie správnej verzie Arduino dosky.

Obrázok 2-3 ukazuje nastavenie sériového portu na ktorý máme Arduino pripojené. Vo vašom prípade sa môže číslo COM portu líšiť.



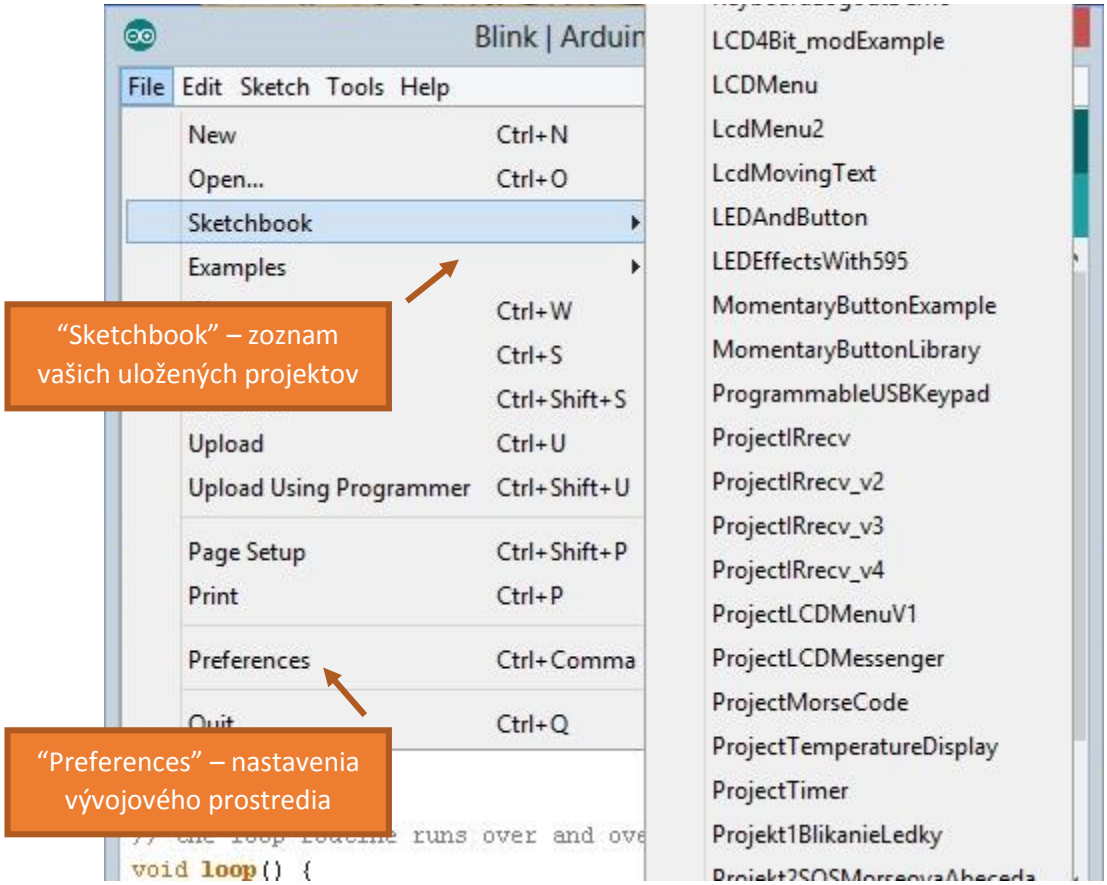
Obrázok 2-3.

2.3. Podrobnejší popis vývojového prostredia

V tejto časti sa pozrieme podrobnejšie na vývojové prostredie pre Arduino. Popíšme si menu a ďalšie možnosti v nastaveniach.

2.3.1. „Sketchbook" – kniha projektov

Na obrázku nižšie môžete vidieť položku z menu **File** a to „**Sketchbook**" – tu sa vám budú zobrazovať všetky projekty, ktoré ste vytvorili a uložili.

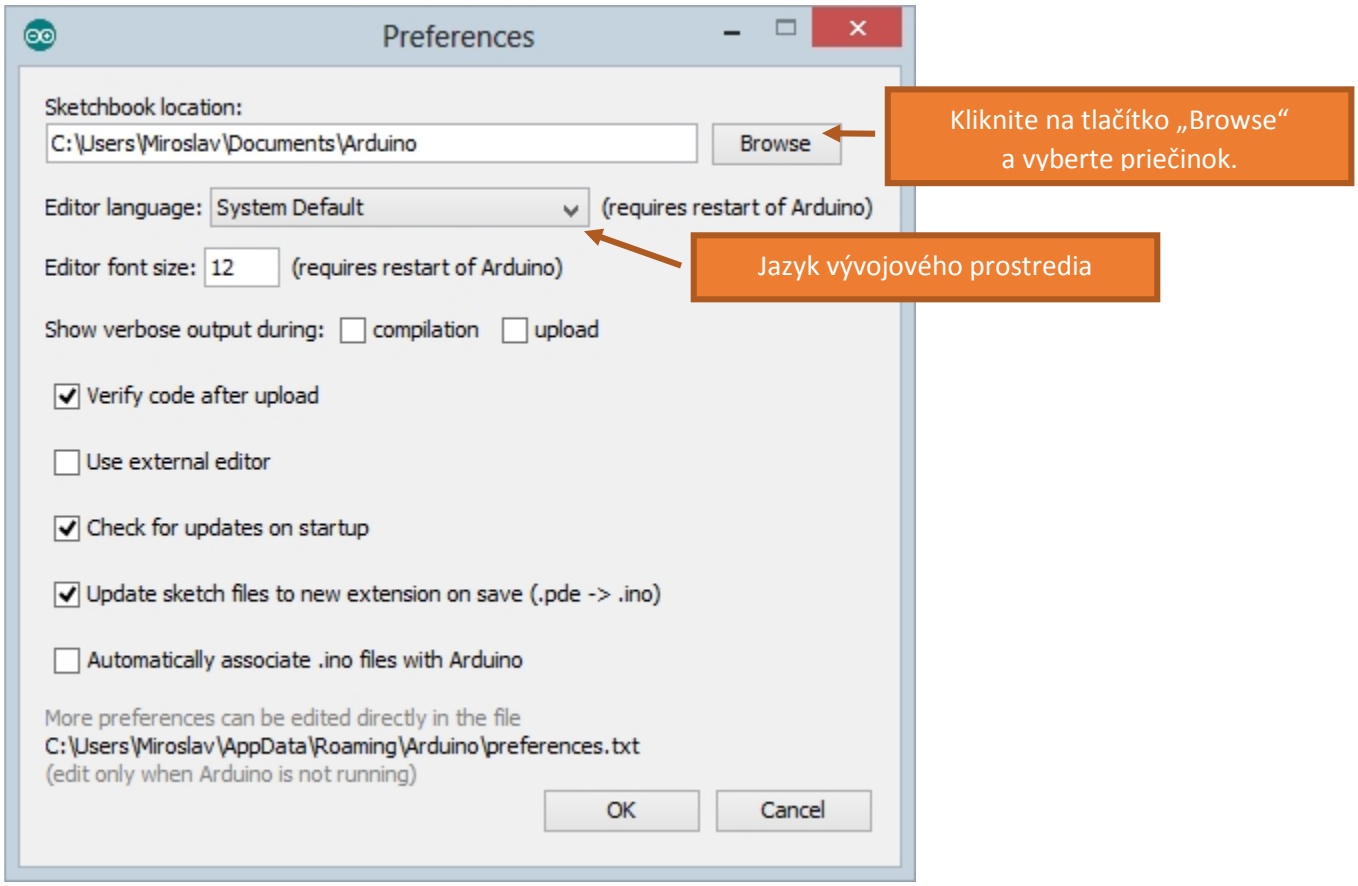


Štandardná cesta, kde sa ukladajú projekty je nasledovná:

Pre Windows Vista/7/8 **c:\Users\<User>\Documents\Arduino** alebo pre Windows XP **c:\Documents and settings\<User>\My Documents\Arduino**

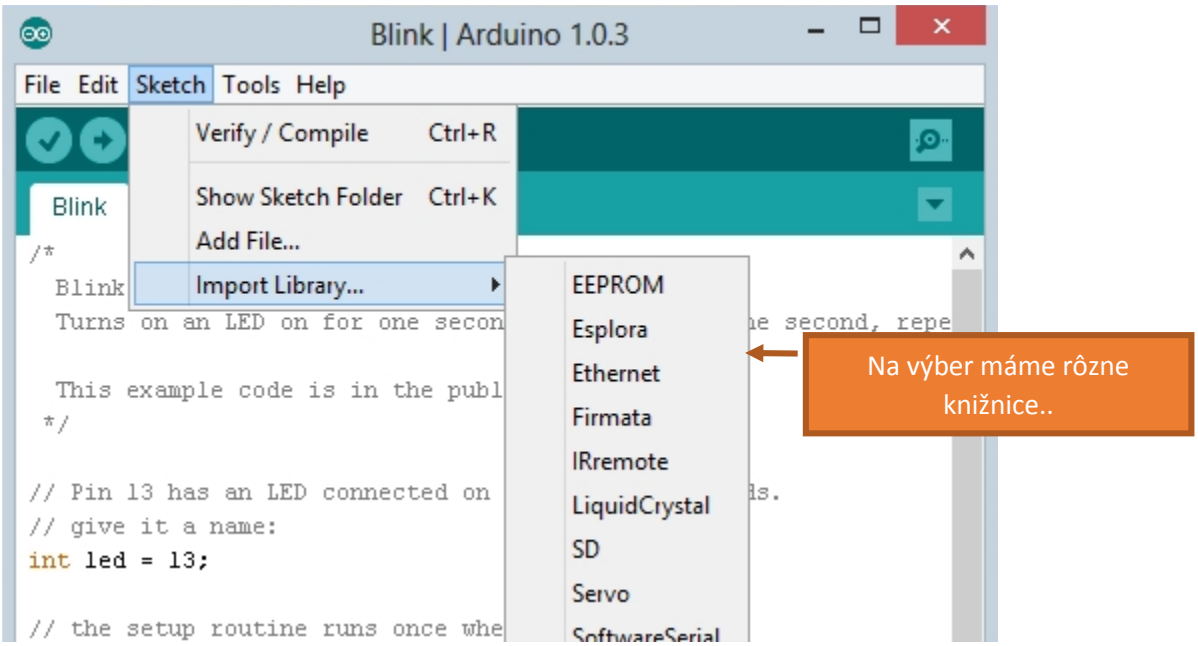
Ak si chcete nastaviť inú cestu pre ukladanie projektov, môžete tak urobiť v nastaveniach vývojového prostredia **File->Preferences.**

Vyberieme tlačítkom „Browse" priečinok, kde chceme aby sa nám ukladali projekty. Po nastavení novej cesty klikneme na tlačítko OK aby sa nastavenia uložili.



2.3.2. Import knižníc

Do skice môžeme naimportovať rôzne knižnice na prácu s LCD displejom, krokovými motormi alebo servo motormi, atď. Import urobíme veľmi jednoducho v menu vyberieme **Sketch -> Import Library.. -> ..a vyberieme požadovanú knižnicu**, viď obrázok nižšie.



2.4. Arduino programovací jazyk

Programovací jazyk pre Arduino vychádza z jazyka C/C++. Je to vlastne céčko obohatené o knižnice pre Arduino. Arduino IDE obsahuje knižnice pre prácu so sériovým portom, LCD displejmi, SPI, WiFi, SD kartami, krokovými a servo motormi a veľa ďalšími.

2.5. Arduino „Sketch“

Ako už tušíte každý program v Arduino IDE sa volá **sketch**, po slovensky teda **skica** alebo náčrt. Teraz si popíšeme štruktúru jednej takej skice:

Každá skica by mala mať minimálne dve metódy **setup** a **loop**. Metóda **setup** sa vykoná **iba raz** pri štarte programu alebo po stlačení tlačítka reset na Arduino doske.
Do tejto metódy sa vpisujú počiatočné nastavenia programu, napr. nastavenie vstupných a výstupných pinov, nastavenie sériového portu a iné (záleží na konkrétnom programe).

Metóda **loop** sa vykonáva **neustále** dokola pokiaľ je samozrejme Arduino pripojené k zdroju napätia. Do tejto metódy sa píše všetko ostatné čo má program vykonávať.

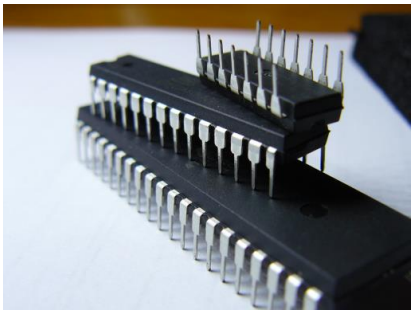
```
void setup() {

  // nastavenie pinov, seriového portu, atď

}

void loop() {

  // sem sa vpíše hlavný kód programu, ktorý sa bude vykonávať dookola

}
```

Skica 1. Základná štruktúra Arduino programu – skice.

2.6. Prehľad hardvéru

Táto podkapitola bude venovaná hardvérovej časti Arduino platformy.

2.6.1. Na akom princípe funguje Arduino ?



Základ Arduina tvorí **8 bitový mikrokontrolér s AVR architektúrou** od firmy Atmel. Takéto mikrokontroléri disponujú rôznymi perifériami, napríklad čítač/časovač , ktorý môže slúžiť na rôzne účely ako presné časovanie udalostí, generovanie PWM, čítanie frekvencii, atď. ďalšou perifériou môže byť AD prevodník (prevod analógovej hodnoty na digitálnu t.j. prevod napätia na číslo). Aby sme mohli takéto periférie použiť musíme poznať patričné registre (každá periféria môže mať vlastné). Správnym nastavením registrov tak docielime správnu funkcionalitu periférie.

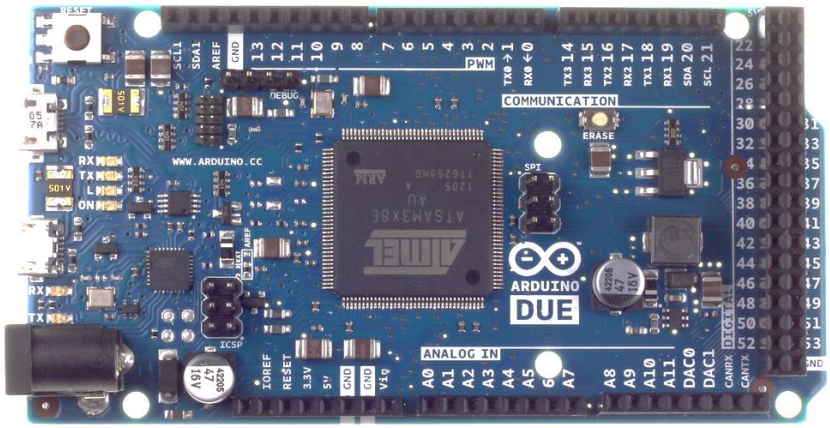
Teraz sa ukáže jedna veľká výhoda Arduina – na to aby sme mohli niečo vytvoriť, naprogramovať nemusíme poznať žiadny register, nemusíte sa učiť architektúru žiadneho mikrokontroléra (ak nechcete). Samozrejme **vedieť viac** o architektúre **nie je na škodu**, ale nie je to nutné na to aby ste niečo vytvorili**.**

Ďalšou výhodou je spôsob akým sa Arduino programuje, nie je totižto nutné použiť programátor pre mikrokontroléri. Mikrokontrolér na plošnom spoji Arduina obsahuje **bootloader**, ktorý nám zabezpečuje naprogramovanie Arduina po sériovej linke a následné spustenie programu v mikrokontroléri. Programovanie je jednoduché stačí pripojiť Arduino pomocou USB kábla k PC vo vývojovom prostredí vybrať správny port, verziu Arduina a kliknúť na tlačítko **„Upload“** alebo klávesy **Ctrl+U**.

2.6.2. Verzie Arduina

Arduino existuje vo viacerých verziách. Existujú verzie v tzv. štandardnej veľkosti ako napríklad **Arduino UNO**, **Leonardo**, alebo zo starších **Duemilanove**.

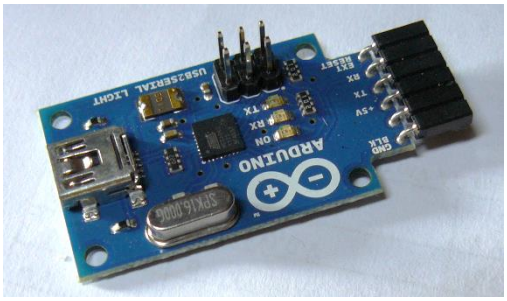
Potom sú verzie „veľké“ a to Arduino Mega 2560, Arduino Mega ADK a Arduino Due.



Do tretice sú aj malé verzie ako Arduino Pro Mini, Arduino Micro, Arduino Nano a Lilypad Arduino.

Všetky verzie si môžete pozrieť na stránke projektu http://arduino.cc/en/Main/Products.

Je teda z čoho vyberať. Každá verzia sa hodí na niečo iné. Ja si Arduino vyberám podľa projektu, ak potrebujem malé rozmery, vyberám z tých malých verzii, záleží ešte na tom či budem potrebovať s USB konektorom alebo bude stačiť bez. Ak robím samostatný projekt, ktorý nebude komunikovať s PC, USB v tomto prípade potrebovať nebudem, respektíve USB bude nutné iba pri programovaní Arduina. Verzie Arduina bez USB portu sa programujú pomocou USB to Serial adaptéra.



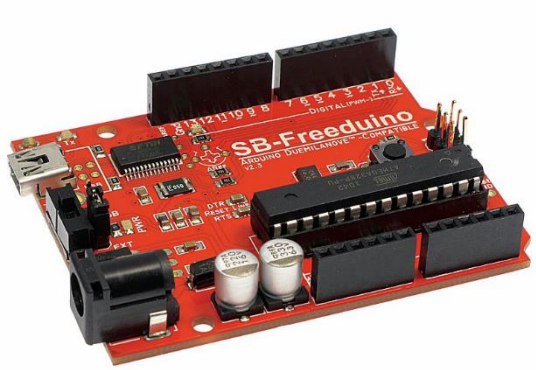
2.6.3. Klony Arduina

Ako to už býva zvykom, na svete sa dosť veľa vecí kopíruje a väčšinou tie, ktoré sú úspešné. Tak takémuto kopírovaniu sa nevyhlo ani Arduino. Existuje viac než dosť klonov ako napríklad Funduino, AVRduino, Sainsmart, Brasuíno, Diavolino, Freeduino, Illuminato Genesis, Seeeduino, Sunduino a takto by sme mohli pokračovať asi ďalšou stranou ☺.

Dokonca vznikli aj rôzne mutácie, ktoré sú určené napríklad na riadenie motorov, takéto "Arduino" sa nazýva Motoruino. Taktiež existuje verzia ArduPilot pre autonómne riadenie rôznych "hračiek".

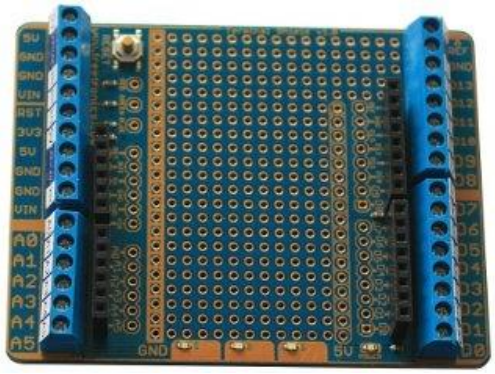
Na obrázkoch môžete vidieť klony **Freeduino** a **AVRduino**.

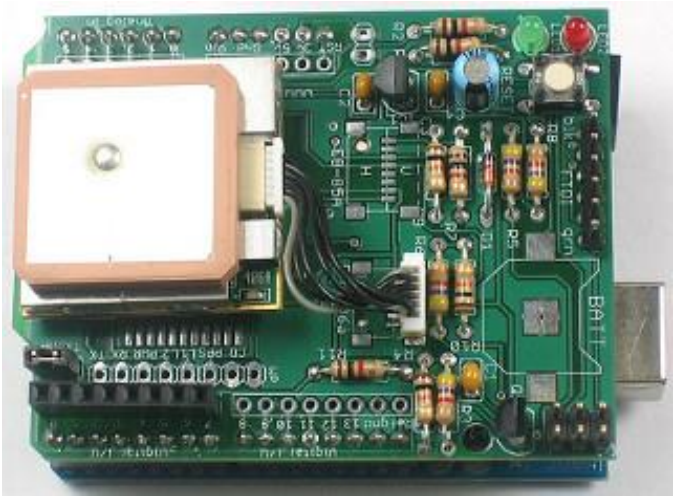
Niektorý výrobcovia klonov úvádzajú, že sú ich výrobky kompatibilné s originálnym Arduinom.



2.6.4. Rozširujúce moduly - SHIELDY

Shield – takto sa nazývajú rozširujúce dosky k Arduinu. Takýchto shieldov existuje veľké množstvo a to na všetko na čo si len spomeniete. Napríklad Arduino Ethernet shield – veľmi zaujímavý shield na komunikáciu po sieti, Arduino GSM shield – pripojenie Arduina na internet pomocou GPRS, rôzne „proto" shieldy – určené na prototypovanie. Veľké množstvo shieldov si môžete pozrieť aj na stránke http://shieldlist.org/.

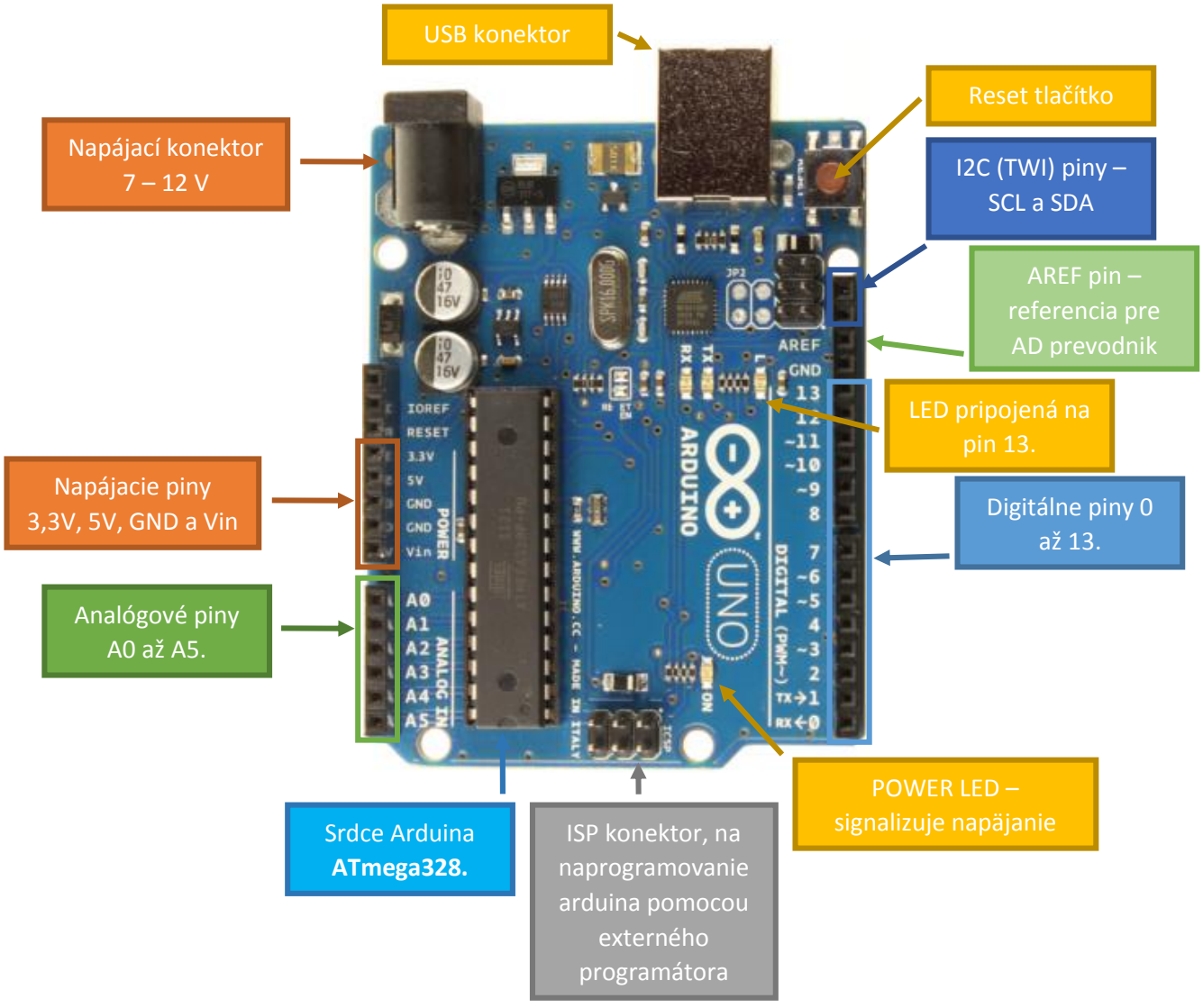




2.6.5. Arduino UNO R3

V tejto časti knihy si popíšeme dosku Arduino UNO R3. Základom Arduina je mikrokontrolér **Atmega328**, je to 8 bitový mikrokontrolér s architektúrou AVR od firmy Atmel, ktorý v tejto verzii Arduina pracuje na frekvencii 16MHz. Napájacie napätie celej dosky je 5V. Môže byť napájané buď s USB portu alebo cez jack konektor z externého zdroja – takéto napätie by malo byť v rozsahu 7 - 12V.

Každý pin na Arduinu je nejak označený ako je vidieť na obrázku, alebo ak sa pozriete na vlastné Arduino.



Arduino má **14 digitálnych vstupno-výstupných** pinov označených číslicami 0 až 13. Z týchto je **6 PWM** výstupných pinov označených PWM alebo vlnovkou (~).

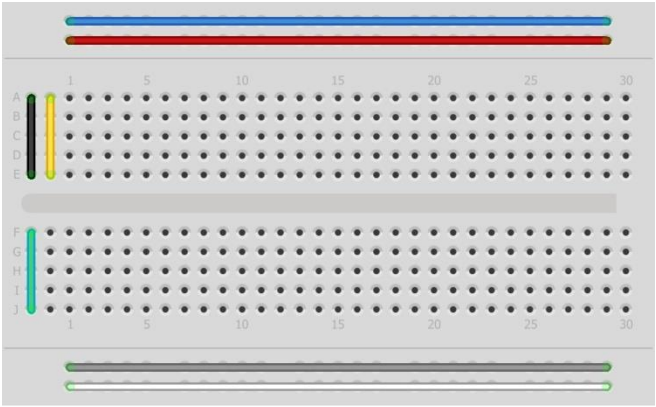
Ďalej má **6 analógových** vstupných pinov označených **A0** až **A5**. Piny označené GND (Ground=>Zem), 5V, 3V3(3,3V) hovoria sami zaseba. Pin **Vin** je prepojený s jack konektorom na externé napájanie, tzn. že ho tiež môžeme použiť pre napájanie na externý zdroj. Nemusíme tak zhánať konektor, stačí nám jeden obyčajný vodič, ktorý

zasunieme do tochto pinu. **RESET** pin slúži ak chceme arduino vyresetovať externe, napr. vo vlastnom projekte.

2.6.6. Kontaktné pole „Breadboard“

Ak sa nestretávate s kontaktným poľom po prvýkrát môžete túto časť preskočiť, ale ak áno čítajte ďalej a zistíte ako kontaktné pole funguje.

Prepojenie jednotlivých dier v kontaktom poli ilustruje obrázok 2-4. Vodiče vo vodorovnom smere (modrý, červený, šedý a biely) znázorňujú prepojenie kontaktov v rovnakom smere a to na krajoch kontaktného poľa. Prepojenie kontaktov v zvislom smere znázorňujú vodiče čiernej, žltej a zelenej farby.



Obrázok 2-4. Prepojenie dier na kontaktnom poli, znázornené vodorovnými a zvislými vodičmi.

3. Projekt 1: „Hello world“ v elektronike

V tejto kapitole si ukážeme ako pracovať s LEDkami a napíšeme jednoduchý program na rozblikanie LEDky.

3.1. Zoznam súčiastok

Tak a konečne prvý projekt, v ktorom si rozblikáme LEDku. Čo k tomu budeme potrebovať je v tabuľke „Zoznam súčiastok 1“. Na napájanie Arduina bude stačiť napájanie z USB portu.

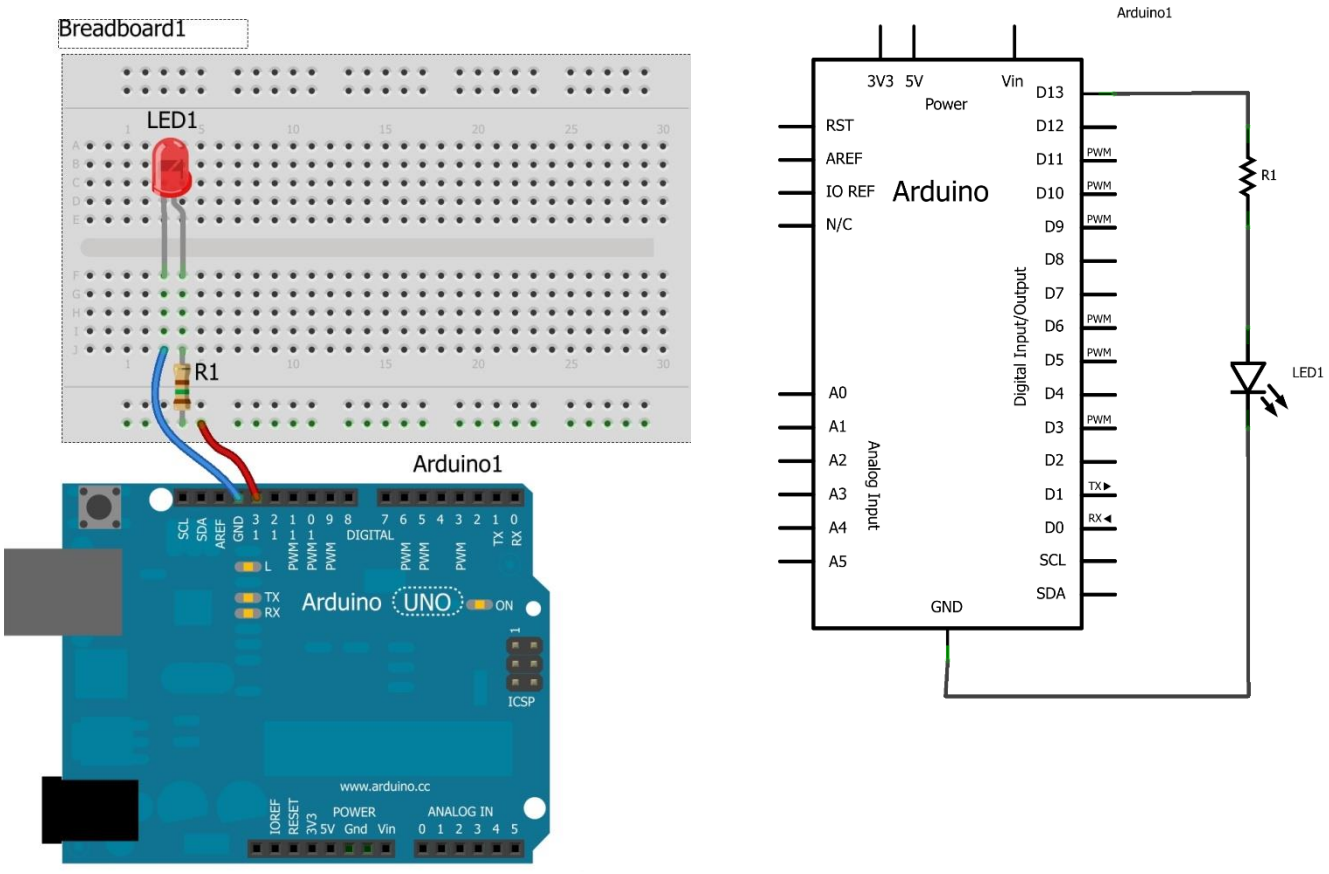
Zoznam súčiastok 1

Označenie	*Hodnota*	*Počet kusov*
LED1	Červená 5mm LED dióda	1
R1	Rezistor 150R(150 Ohmov)	1
Breadboard1	Kontaktné pole	1
Arduino1	Arduino UNO R3	1

Pri tomto zapojení je možné využiť aj LED (L) pripojenú na pin číslo 13.

3.2. Zapojenie

Všetko pozapájame podľa obrázka 3-1. Ak pracujeme s Arduino doskou, zapájame alebo odpájame nejaké súčiastky, tak odporúčam Arduino vytiahnuť zo zdroja napätia v tomto prípade z USB portu.



Obrázok 3-1. Zapojenie LED diódy.

3.3. Popis zapojenia

Zapojenie je veľmi jednoduché, obsahuje dve súčiastky (okrem Arduina a kont. poľa), LEDku LED1 a rezistor R1. Rezistor R1 slúži ako predradný rezistor pre LED diódu. Napájacie napätie LED diód je rôzne, ak si zoberieme len červené, tak ich napájacie napätie sa pohybuje od 1,8 do 2 voltov (záleží na výrobcovi). Ak by sme LED diódu pripojili na 5V, diódou by vtedy pretekal veľký prúd, ktorý by ju mohol zničiť, preto sa pred LEDkami dáva tzv. predradný rezistor. Hodnotu takéhoto rezistora (odporu) vypočítame nasledovne:

$$R = \frac{U_{zdroja} - U_{LED}}{I_{LED}}$$

Potrebujeme poznať nasledovné hodnoty:

1. U_{zdroja} - **napájacie napätie zdroja** ku ktorému chceme LEDku pripojiť,
2. U_{LED} - **napájacie napätie LEDky**, najlepšie zistíme s katalógu alebo s datasheetu,
3. I_{LED} - **prúd LEDky**, takisto zistíme z katalógu alebo s datasheetu

Všetky neznáme vo vzorci poznáme. U_{zdroja} je 5V, pretože LEDka je pripojené k pinu ktorý bude spínaný k napájaciemu napätiu arduina tzn. 5V. U_{LED} , môžeme zistiť, ako bolo spomenuté vyššie alebo podľa tabuľky:

Farba	*Napätie*
Infračervená	1.6 V
Červená	1.9 V
Oranžová	2.2 V
Žltá	2.4 V
Zelená	2.6 V
Modrá	3.5 V
Biela	3.5 V
Ultrafialová	3.5 V

Tabuľka 1. Napájacie napätie LED diód podľa farieb.

I_{LED} , použitej LEDky je 20mA, teda 0,02A. Ak tieto hodnoty dosadíme do vzorca.

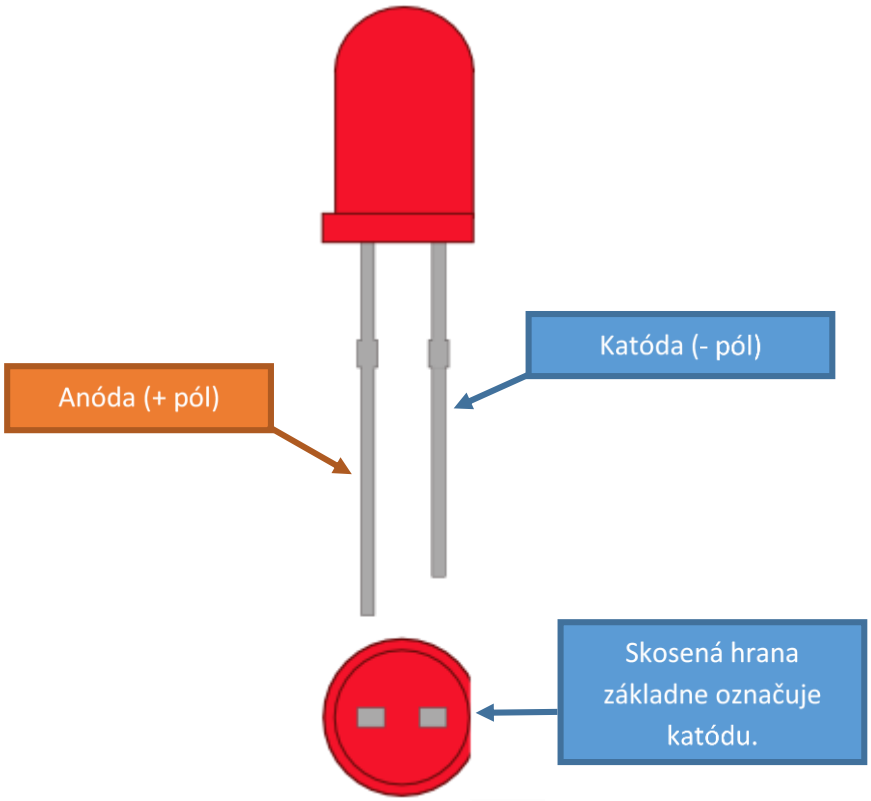
$$R = \frac{5 - 1{,}9}{0{,}02} = 155\Omega$$

Vyšlo nám, že predradný rezistor by mal mať odpor 155Ω. Najbližšia hodnota rezistora je teda 150R ako je vidieť aj v zozname súčiastok.

3.3.1. Ako zapojiť LED diódu?

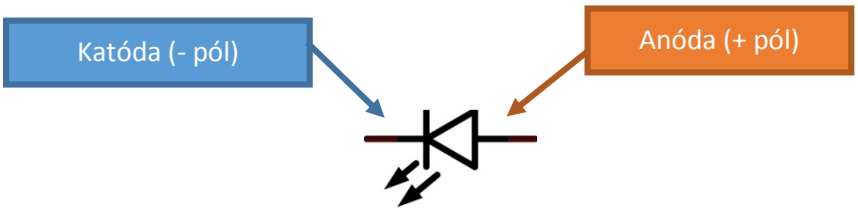
Nasledujúcej časti si povieme niečo o LED diódach. **Light-emitting diode** v skratke teda **LED** je polovodičová súčiastka, ktorá vyžaruje svetlo, má dva vývody. Tieto vývody sa nazývajú **katóda** (záporný pól) a **anóda** (kladný pól). Katóda sa pripája na záporný pól a anóda na kladný pól napätia. Anódu od katódy rozoznáme tak, že anóda má dlhší vývod (viď obrázok 3-2). Toto platí pri nových diódach, ktoré si kúpime, ale ak diódu

„vypájkujeme“ z nejakého plošného spoja vývody sú vtedy rovnaké. Ako potom rozoznať vývody ? Jednoducho, LED dióda by mala mať na jednej strane skosenú (upiľovanú) hranu základne, na tejto strane je vždy katóda. LED diódy sa dnes vyrábajú v rôznych prevedeniach, ktoré sa líšia rôznymi parametrami, napr.: rozmermi, vyžarovaným svetlom (farbou a intenzitou), napájacím napätím a prúdom.



Obrázok 3-2. LED dióda, popis vývodov.

Schematická značka LED diódy je na obrázku 3-3.



Obrázok 3-3. Schematická značka LED diódy.

3.4. Kód programu

Do Arduino IDE skopírujeme nasledovný kód (Skica 2):

```
// výstupný pin pre LEDku
int ledPin = 13;

/* setup metóda sa vykoná iba raz a to pri prvom štarte alebo pri
stlačení tlačitka reset na Arduino doske */
void setup() {
  pinMode(ledPin, OUTPUT);      // Nastavíme pin ako výstupný
}

/*
loop metóda sa vykonáva neustále dokola, samozrejme pokiaľ je Arduino
pripojené do USB portu alebo na iný zdroj napätia
 */
void loop() {

  digitalWrite(ledPin, HIGH);  /* pošleme na ledPin hodnotu HIGH
(čo je vlastne hodnota 1), teda 5 voltov tzn. LEDka svieti */

  delay(1000);                 // čakáme 1000 milisekund (1 sekundu)

  digitalWrite(ledPin, LOW);   /* pošleme na ledPin hodnotu LOW
(čo je vlastne hodnota 0), teda 0 voltov tzn. LEDka nesvieti */

  delay(1000);          // zase čakáme 1000 milisekund (1 sekundu)
}
```

Skica 2. Blikajúca LEDka

Ak kód skompilujeme, pripojíme Arduino a nahráme program do Arduina. LEDka by mala začať blikať.

3.5. Popis programu

Teraz si rozoberieme kód skice 2. Riadok `int ledPin = 13;` vytvorí premennú typu **int** (integer) s názvom **ledPin** a hodnotou **13**. Čo to vlastne znamená ? Určili sme si číslo pinu pre LEDku a toto číslo sme vložili do premennej ledPin, aby sme v kóde nemuseli pokaždé písať číslo 13, tzn. namiesto toho použijeme túto premennú. Ak budeme chcieť v budúcnosti zmeniť toto číslo, bude stačiť len prepísať číslo 13 na iné. Určite je lepšie zmeniť hodnotu na jednom mieste ako keby sme mali hľadať konkrétne číslo v celom kóde. Zatiaľ je skica síce len niekoľko riadková, no v budúcnosti budete možno vytvárať aj zložitejšie, kde bude niekoľko desiatok riadkov, preto je lepšie mať kód prehľadný.

Ďalej sa dostávame do metódy setup, kde máme riadok `pinMode(ledPin, OUTPUT);`. Metóda pinMode nastavuje tzv. mód pinu. Digitálne piny na Arduinu môžu byť totiž vstupné alebo výstupné, podľa toho ako si určíme metódou pinMode. Metóda má dva parametre: číslo pinu a konštantu OUTPUT (Výstupný) alebo INPUT (vstupný). V našom prípade sa vkladajú do metódy parametre – ledPin (s hodnotou 13) a konštanta OUTPUT tzn. týmto zápisom sme určili, že číslo pinu 13 je nastavený ako výstupný.

V metóde loop máme niekoľko riadkov.
Riadok s kódom `digitalWrite(ledPin, HIGH);` urobí to že zapíše na výstupný pin 13 hodnotu HIGH (teda 1) tzn. na pine 13 sa objaví 5V, LEDka je tak pripojená k zdroju napätia a preto svieti. V ďalšom riadku sa volá metóda delay: `delay(1000);`. Táto metóda slúži na pozastavenie vykonávania programu určitý čas. Čas sa určuje vstupným parametrom v milisekundách, v našom prípade je to 1000 milisekúnd teda 1 sekunda. Ďalej sa zase volá metóda digitalWrite, no tentokrát s odlišným druhým parametrom LOW - `digitalWrite(ledPin, LOW);`. Metóda digitalWrite nám teraz zapíše na výstupný pin hodnotu LOW (teda 0V), LEDka nám teraz nesvieti.
V nasledujúcom riadku voláme opäť metódu delay s rovnakým parametrom, tzn. vykonávanie programu sa opäť pozastaví na 1 sekundu. A celý cyklus sa opakuje (až kým neodpojíme zdroj napätia), pretože tento kód sa nachádza v metóde loop.

3.6. Na záver projektu

Túto skicu môžeme ďalej upravovať, napríklad môžeme zmeniť pin na iný, alebo upraviť intenzitu blikania. Ak napríklad chceme zvýšiť intenzitu blikania stačí v metóde delay znížiť čas z 1000 na 500 milisekúnd. LEDka nám teraz bude blikať dvakrát rýchlejšie.
Čo sa vlastne stalo zmenou tohto parametra? Tým, že sme znížili túto hodnotu 1000 na 500. Program teraz čaká len 500 milisekúnd, tzn. že LEDka svieti kratšiu dobu ako predtým a celý proces zasvietenia a zhasnutia sa zrýchlil.

4. Projekt 2: Neblikám pre srandu, volám S.O.S

V tejto kapitole rozblikáme LEDku trochu inak. LEDka bude blikať podľa Morseovej abecedy, konkrétne „vyblikáme“ medzinárodnú skratku **SOS**, teda volanie o pomoc.

4.1. Kód programu

V tomto projekte využijeme zapojenie z predchádzajúceho projektu, zmeníme len program. Nasledujúcu skicu skopírujte alebo prepíšte do vývojového prostredia.

```
int ledPin = 13;

void setup() {
  // nastavíme ledPin (pin číslo 13) ako výstupný
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // 3 krátke '...' v morseovej abecede písmeno 'S'
  for(int i=0; i<3; i++){
    digitalWrite(ledPin, HIGH);
    delay(120);
    digitalWrite(ledPin, LOW);
    delay(120);
  }

  delay(100); //čakáme 100 milisekúnd

  // 3 dlhé '---' v morseovej abecede písmeno 'O'
  for(int i=0; i<3; i++){
    digitalWrite(ledPin, HIGH);
    delay(350);
    digitalWrite(ledPin, LOW);
    delay(350);
  }

  delay(100);//zase čakáme 100 milisekúnd

  //a zase 3 krátke '...' teda 'S'
  for(int i=0; i<3; i++){
    digitalWrite(ledPin, HIGH);
    delay(120);
    digitalWrite(ledPin, LOW);
    delay(120);
  }

  delay(5000);//celý cyklus sa spustí znova po 5 sekundách
}
```

Skica 3. Kód skice pre SOS projekt.

Po skompilovaní a nahratí programu do Arduina, LEDka zabliká trikrát rýchlo, trikrát pomaly a zase trikrát rýchlo tzn. v Morseovej abecede tri krátke ako „S“ (...) , tri dlhé ako „O“ (---) a tri krátke ako „S“ (spolu ...---...). Po piatich sekundách sa toto blikanie opakuje.

4.2. Popis programu

Teraz si popíšeme kód skice 3. Inicializa premennej ledPin sa nelíši od predchádzajúcej skice, taktiež v setup metóde sa nič nezmenilo. Zmenu môžeme vidieť v loop metóde, kde máme tri cykly **for .** Cyklus for funguje nasledovne:

```
for(inicializa premennej; podmienka; zmena premennej){

    Kód vo vnútri sa vykonáva dovtedy, pokiaľ platí podmienka.
}
```

V našom prípade má cyklus for v sebe premennú **i** typu **int**, s počiatočnou hodnotou **0**. Podmienka v cykle je **i < 3**, čo znamená, že cyklus sa bude vykonávať pokiaľ premenná i bude menšia ako 3. Premenná sa mení inkrementáciou **i++**, tzn. hodnota sa zvyšuje o jedna.

```
for(int i=0; i<3; i++){
  ...
}
```

Každým cyklom sa teda zvýši hodnota premennej o jedna a v nasledujúcom cykle sa v podmienke kontroluje. Ak hodnota vyhovuje, hovoríme, že podmienka bola splnená - cyklus pokračuje, ale ak hodnota premennej i nevyhovuje podmienke = podmienka nebola splnená = cyklus sa ukončí a beh programu pokračuje ďalej za cyklom for (výsledok podmienky môže byť **true = pravda**, alebo **false = nepravda**).

Vysvetlili sme si ako funguje cyklus for. Teraz sa pozrieme dovnútra cyklu, máme tam nasledujúci kód:

```
digitalWrite(ledPin, HIGH);
delay(120);
digitalWrite(ledPin, LOW);
delay(120);
```

Čo je vlastne rovnaký kód ako v predchádzajúcom projekte, len je tu nastavená kratšia doba svietenia LEDky na 120 ms. Podľa podmienky v cykle sa kód vo vnútri vykoná trikrát, tzn. LEDka zasvieti a zhasne trikrát za sebou. Ostatné cykly fungujú analogicky, s tým, že v strednom cykle je nastavené oneskorenie o niečo väčšie(350 ms).
Medzi cyklami je tiež nastavené oneskorenie na 100 ms. Lepšie tak uvidíme zmenu v blikaní LEDky = lepšie rozoznáme znaky Morseovej abecedy.

Na konci vidíme volanie metódy delay s hodnotou parametra 5000, teda oneskorenie 5s, tzn. celý proces blikania sa opakuje od začiatku po piatich sekundách.

4.3. Na záver projektu

V tomto projekte bolo zapojenie rovnaké ako v predchádzajúcom. Kód programu je veľmi jednoduchý a je ho možné upraviť napríklad tak aby „vyblikal“ aj iné písmená z Morseovej abecedy.

5. Projekt 3: Pulzar (Pulzujúca LED)

V tomto projekte zostaneme ešte u jednej LEDky. Povieme si niečo o pulzne šírkovej modulácii čiže PWM (Pulse Width Modulation). Využijeme PWM na plynulé rozsvecovanie a zhasínanie LEDky.

5.1. Zoznam súčiastok

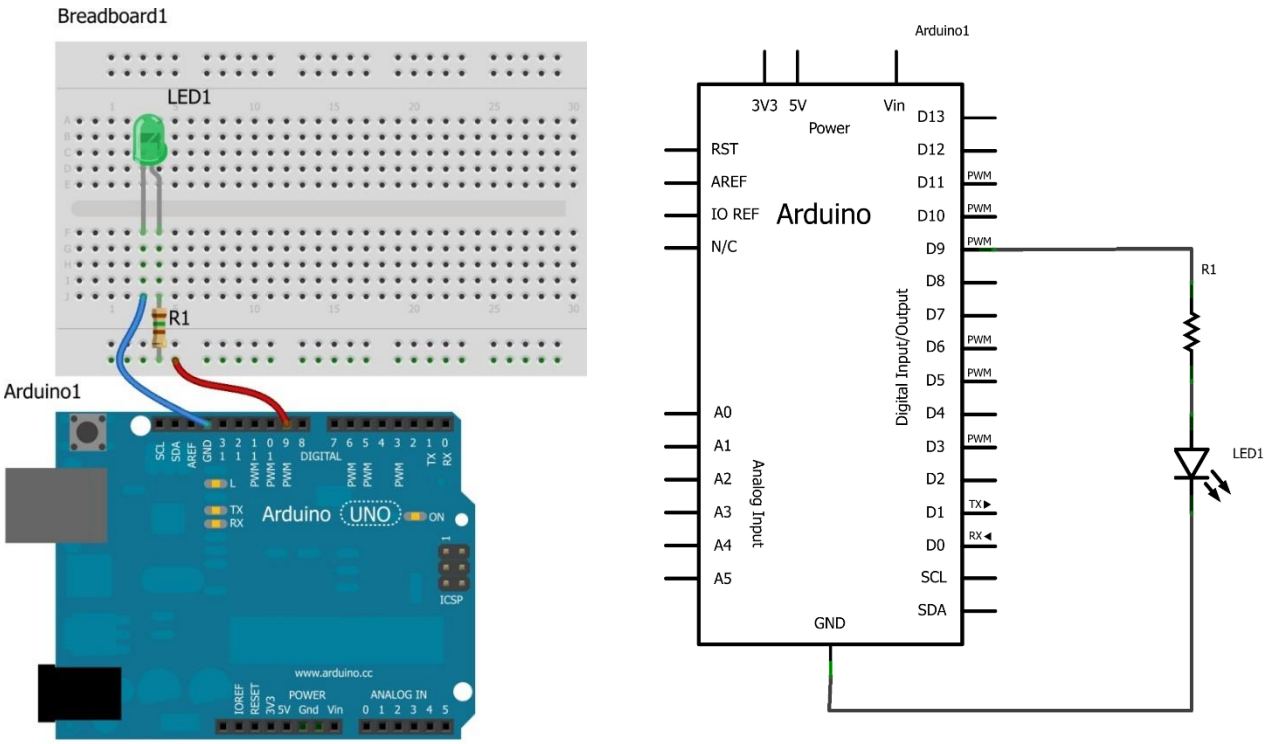
Zoznam súčiastok pre tento projekt máme v tabuľke 2. Je podobný ako v projektoch 1 a 2. Zamenili sme tentokrát farbu LEDdky za zelenú.

Zoznam súčiastok 2

Označenie	*Hodnota*	*Počet kusov*
LED1	Zelená 5mm LED dióda	1
R1	Rezistor 150R(150 Ohmov)	1
Breadboard1	Kontaktné pole	1
Arduino1	Arduino UNO R3	1
Prepojovacie káble		

5.2. Zapojenie

Zmenilo sa nám trochu zapojenie. LEDku tentokrát zapojíme do pinu číslo 9, podľa obrázka 5-1.



Obrázok 5-1. Zapojenie pulzujúcej LED diódy.

5.3. Popis zapojenia

Zapojenie sa veľmi nezmenilo. Zostal rovnaký aj počet súčiastok, len sme zmenili farbu LEDky, ktorú sme pripojili k pinu D9.

5.4. Kód programu

Zdrojový kód skice pre tento projekt môžeme vidieť nižšie. Ak ho skompilujeme a nahráme do Arduina, LED dióda sa bude pozvoľna rozsvecovať a zhasínať (preto pulzujúca LED). Postupne sa teda bude zvyšovať jas a keď dosiahne maxima začne jas klesať až LEDka úplne zhasne. Celý proces sa opakuje dookola.

```
int ledPin = 9;             // číslo pinu na ktorý je pripojená LEDka
int brightness = 0;         // jas LEDky
int stepValue = 5;          // veľkosť kroku na nastavenie jasu LEDky
int direction = 1;          // Smer zvyšovanie alebo znižovanie jasu

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // nastavenie jasu LEDky
  analogWrite(ledPin, brightness);

  // ak jas dosiahne maximálnej hodnoty, zmeníme smer na znižovanie
  if(brightness == 255){
    direction = 2;
  }
  // ak jas dosiahne minimálnej hodnoty, zmeníme smer na zvyšovanie
  if(brightness == 0) {
    direction = 1;
  }

  switch(direction){
  case 1:// zvyšovanie jasu
    brightness = brightness + stepValue;
    break;
  case 2:// znižovanie jasu
    brightness = brightness - stepValue;
    break;
  }

  delay(40);
}
```

Skica 4. Zdrojový kód skice pre pulzujúcu LED diódu.

5.5. Popis programu

V skice sa nám tentokrát objavili ďalšie premenné okrem ledPin, ktorá má tentokrát hodnotu 9, pretože LEDku máme zapojenú v pine číslo 9.

```
int brightness = 0;
```

Táto premenná slúži ako aktuálna hodnota jasu. Môže nadobúdať hodnoty od 0 po 255.

```
int stepValue = 5;
```

Premenná stepValue určuje veľkosť o koľko sa zvýši alebo zníži jas každým cyklom (teda hodnota premennéj brightness) . Hodnota tejto premennej je 5 a v programe sa nemení.

```
int direction = 1;
```

A nakoniec máme premennú direction, ktorá určuje či sa hodnota jasu bude zvyšovať alebo znižovať. Premenná direction môže mať hodnotu 1 alebo 2. Hodnota 1 určuje zvyšovanie jasu a hodnota 2 zase určuje znižovanie.

Setup metóda sa nám opäť nezmenila, preto si popíšeme metódu loop.

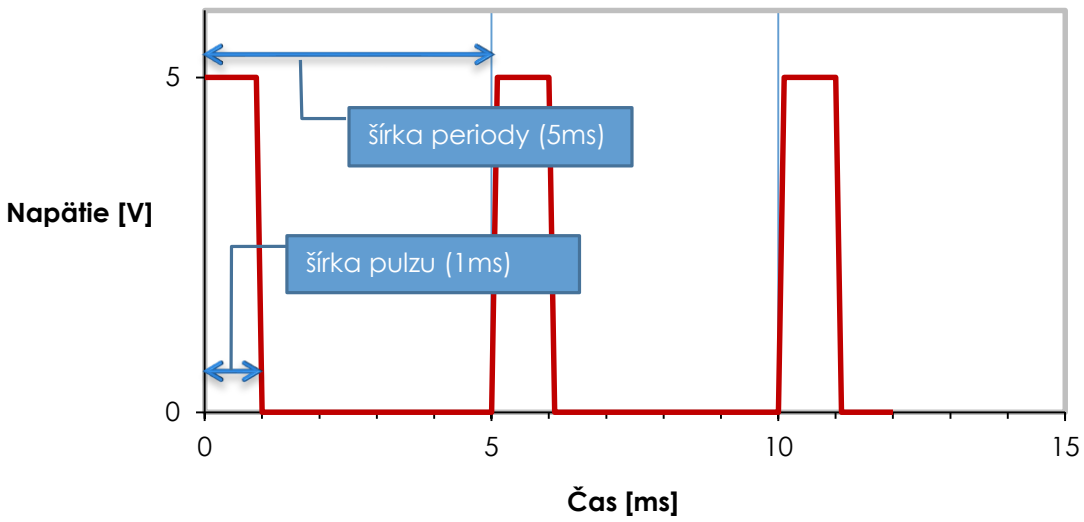
```
analogWrite(ledPin, brightness);
```

Metóda analogWrite má dva parametre. Prvý parameter je číslo pinu a druhý parameter je hodnota z rozsahu 0 až 255. Táto metóda urobí to, že nám na pin číslo 9 zapíše hodnotu z premennej brightness, táto hodnota sa na výstupnom pine objaví ako napätie v rozsahu 0 – 5 voltov. Napríklad ak chceme mať na výstupnom pine 2.5V, tak druhý parameter musí mať hodnotu 127. Metóda analogWrite generuje tzv. PWM signál, pomocou ktorého je možné regulovať výkon, či už jas pri LEDkách alebo otáčky pri motoroch.

5.5.1. Čo je to PWM?

Čo je to PWM a ako to vlastne funguje? **PWM**(Pulse Width Modulation – Pulzne šírková modulácia) je spôsob ako digitálnymi prostriedkami získať analógový výsledok, odborne povedané ☺. A teraz si to rozoberieme. Už vieme, že Arduino dokáže spínať niektoré svoje piny k zemi (0V) alebo k napätiu 5V. Tím vznikne na výstupe stále napätie 0V alebo 5V. Ako je teda možné, že dokáže aj napätia v rozsahu 0 - 5V? Jednoducho, Arduino rýchlo strieda stavy LOW a HIGH (teda 0 a 5V) tzn. generuje signál štvorcového tvaru (Graf 1). Veľkosť výstupného napätia určuje **šírka pulzu**, teda doba počas ktorej bolo na výstupe 5V oproti celkovému **času (šírky) periódy**.

Napríklad ak chceme aby sme na výstupnom pine mali napätie 1V, čo je 1/5 z 5V (z nap. napätia) => 20%, musí byť šírka pulzu taktiež 1/5 celkovej periody v našom prípade 1ms (1/5 z 5ms, viď graf 1)



Graf 1. Priebeh PWM signálu.

Ak sa pozrieme na graf 1, tak vidíme, že šírka pulzu je 1ms. Šírka celej periódy je 5ms.

```
if(brightness == 255){
    direction = 2;
}
```

Príkaz **if** sme tu ešte nemali, tak si ho popíšeme. Slúži k vetveniu programu. Čo znamená, že ak je splnená určitá logická podmienka, zmení sa postupnosť vykonávania príkazov.

```
if(podmienka){
  príkaz
}
```

Podmienka je tzn. booleovský výraz, ktorý môže nadobúdať iba dve hodnoty a to **true** – pravda alebo **false** – nepravda. Príkazy vo vnútri sa vykonajú iba vtedy ak výraz má hodnotu true. Vo výraze môžu byť použité nasledovné porovnávacie operátory:

==	Rovný
>	Väčší
>=	Väčší alebo rovný
<	Menší
<=	Menší alebo rovný
!=	Nerovná sa

Takže v našom prípade by sme podmienku prečítali takto: Ak je premenná brightness rovná 255, nastav premennú **direction** na 2, tzn. ak výraz `brightness == 255` vráti true, vykoná sa kód `direction = 2`.

```
if(brightness == 0) {
    direction = 1;
}
```

Ďalší if, tentokrát sa nastavuje premenná na 1, ak sa premenná brightness rovná nule.

Príkaz if môže mať ešte nasledovnú variantu:

```
if(podmienka) {
    príkaz1;
} else {
    príkaz2;
}
```

Už asi tušíte ako to bude fungovať. Ak je podmienka splnená, teda je true, vykoná sa príkaz1. Ale ak nie je splnená (je false) vykoná sa príkaz2 v **else** bloku.

Ďalej máme príkaz **switch.** Funguje úplne jednoducho, premenná direction je porovnávaná s konštantami v klauzulách **case.**

```
switch(direction){
  case 1:// zvyšovanie jasu
    brightness = brightness + stepValue;
    break;
  case 2:// znižovanie jasu
    brightness = brightness - stepValue;
    break;
}
```

Ak sa zhoduje hodnota premennej direction napr. s hodnotu 1 vykonajú sa príkazy:

```
brightness = brightness + stepValue;
break;
```

Zvýši sa hodnota premennej brightness o hodnotu premennej stepValue a následne sa porovnávanie ďalších konštánt ukončí príkazom **break**.

Ak je hodnota premennej direction rovná konštante 2, tak sa vykonajú príkazy za klauzulou case 2: - analogicky ako v predchádzajúcom prípade, s tím rozdielom, že sa hodnota premennej brightness zníži o hodnotu premennej stepValue.

5.6. Na záver projektu

V tomto projekte sme si povedali čo je to PWM. Ukázali sme na jednoduchom príklade ako riadiť jas LEDky pomocou PWM. Aj tento projekt je možné samozrejme upravovať a rozširovať podľa potreby.

6. Projekt 4: Svetelný efekt – Knight Rider

Knight rider, kto by ho nepoznal ☺, že. V tejto kapitole si spravíme svetelný efekt z lediek ako mal kitt na prednej maske.

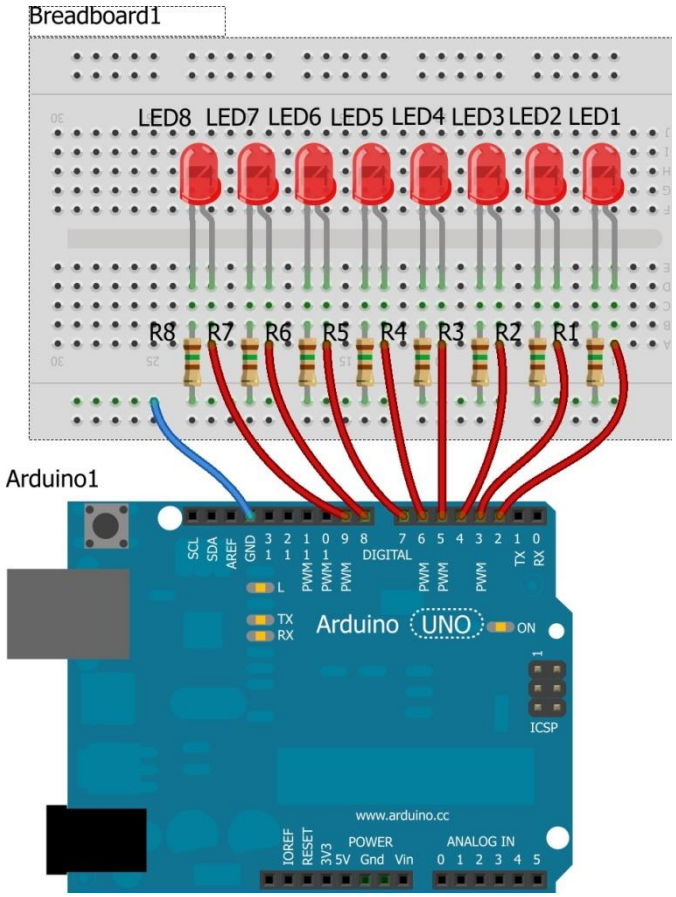
6.1. Zoznam súčiastok

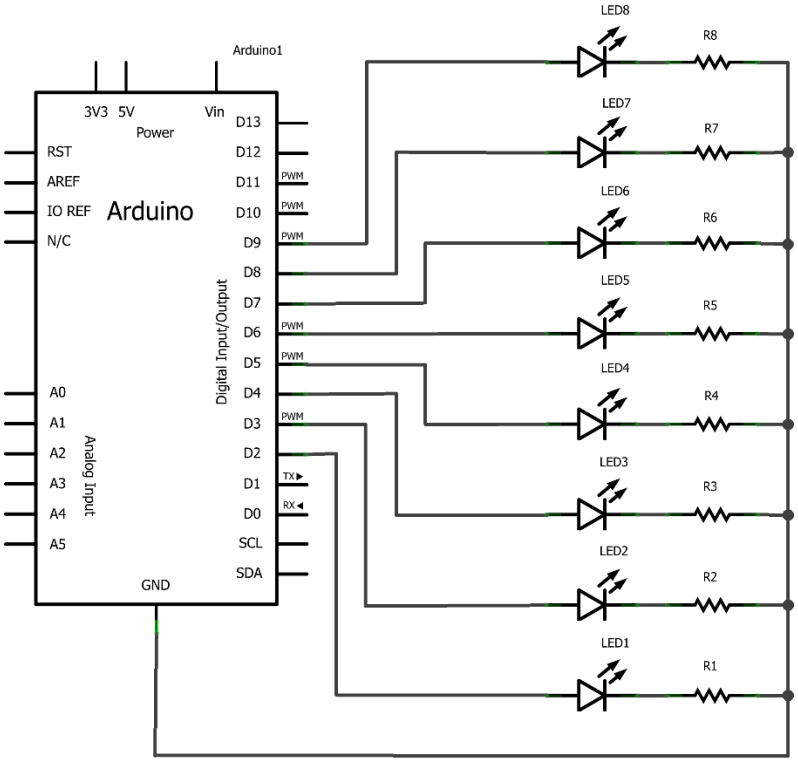
Pre toto zapojenie budeme potrebovať viac LED diód, ako v predchádzajúcich projektoch. Zoznam potrebných súčiastok je v tabuľke.

Zoznam súčiastok 3

Označenie	*Hodnota*	*Počet kusov*
LED1-LED8	červená 5mm LED dióda	8
R1-R8	Rezistor 150R(150 Ohmov)	8
Breadboard1	Kontaktné pole	1
Arduino1	Arduino UNO R3	1
Prepojovacie káble		

6.2. Zapojenie





Obrázok 6-1. Zapojenie súčiastok svetelného efektu Kinght Rider.

Na obrázku 6-1 máme zapojenie svetelného efektu. Všetko pekne pozapájame a pripojíme Arduino k PC.

6.3. Popis zapojenia

Toto zapojenie nie je o nič zložitejšie ako to predchádzajúce. V tomto máme len viac súčiastok. Všetky LEDky LED1 až LED8 sú rovnakého typu. Ak chcete môžete použiť inú farbu. Taktiež rezistory R1 až R8 majú rovnakú hodnotu. Každá jedna LEDka je cez anódu pripojená k samostatnému pinu na Arduine. Na katódach lediek sú predradné rezistory s odporom 150Ω. Druhé konce rezistorov sú pripojené na zem. Tak to by bolo k popisu zapojenia asi všetko.

6.4. Kód programu

Zdrojový kód skice 5 skopírujeme do vývojového prostredia a uploadneme do Arduina. Postupne by sa mala spínať vždy jedna LEDka v poradí LED1, LED2 až LED8 a opačne LED7,LED6, atď.

```
/* čísla výstupných pinov */
byte outputPins[] = {2, 3, 4, 5, 6, 7, 8, 9};

void setup()
{
  for(int i=0; i<8;i++)
  {
    pinMode(outputPins[i], OUTPUT);
    digitalWrite(outputPins[i], LOW);
  }
}

void loop()
{
  for(int i=0; i<7;i++)
  {
    digitalWrite(outputPins[i], HIGH);
    delay(75);
    digitalWrite(outputPins[i], LOW);
  }
  for(int i=7; i>0;i--)
  {
    digitalWrite(outputPins[i], HIGH);
    delay(75);
    digitalWrite(outputPins[i], LOW);
  }
}
```

Skica 5. Zdrojový kód skice pre projekt „Knight Rider“.

6.5. Popis programu

A teraz si popíšeme zdrojový kód skice.

```
byte outputPins[] = {2, 3, 4, 5, 6, 7, 8, 9};
```

Tu máme inicializovanú premennú outpuPins, čo je vlastne pole typu byte. V tomto poli sú zadefinované čísla výstupných pinov 2 až 9. Polia sú indexované od nuly, čo znamená, že prvky v poli začínajú poradovým číslom 0. Napr. ak chceme získať prvý prvok, tak ho z poľa vyberieme nasledovne:

```
byte firstItem = outputPins[0];
```

Druhý prvok: `byte secondItem = outputPins[1];`. Ostatné prvky získame analogicky.

V setup metóde máme cyklus for, ktorý nám nastavuje piny na výstupné. Tiež nastavuje na výstupné piny hodnotu LOW, čo znamená, že všetky LEDky sú na začiatku programu vypnuté.

V loop metóde máme zase cyklus for a to dvakrát. Prvý cyklus spína LEDky v poradí od LED1 po LED7.

```
for(int i=0; i<7;i++)
{
```

```
    digitalWrite(outputPins[i], HIGH);
    delay(75);
    digitalWrite(outputPins[i], LOW);
}
```

Premenná i v cykle, nadobúda hodnoty 0 až 6 a vo vnútri cyklu sa tak postupne nastavujú piny, ktorých čísla sa vyberajú z poľa outputPins podľa premennej i.

Druhý cyklus spína LEDky v opačnom poradí od LED8 po LED2. Cyklus tak pracuje analogicky ako prvý.

6.6. Na záver projektu

Na záver projektu môžeme povedať, že je veľmi inšpiratívny. Určite vás napadnú nejaké úpravy, ktoré by ste mohli urobiť. Napríklad zvýšiť alebo znížiť rýchlosť rozsvecovania LEDiek, pripojiť ďalšie LEDky a využiť tak viac pinov na Arduine. Fantázii sa medze nekladú, to platí aj u projektovania s Arduinom.

7. Projekt 5: Interaktívna LED

V tejto kapitole sa vrátime ešte k jednej LEDke, ale tentokrát si k nej zapojíme aj tlačítko, ktoré nám bude LEDku ovládať. Vysvetlíme si pojmy ako pull up a pull down.

7.1. Zoznam súčiastok

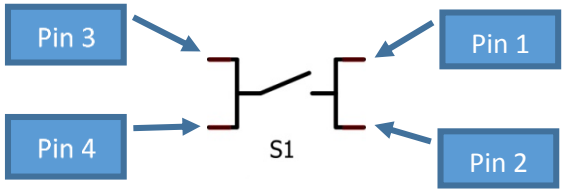
Tento projekt bude prvý, ktorý je interaktívny, čo znamená, že na základe vstupných dát sa bude meniť chod programu. Vstupným dátam sa rozumie stav tlačítka. Arduino bude sledovať stav tlačítka a na základe tohto stavu bude Arduino spínať LEDku. Zoznam potrebných súčiastok je v tabuľke.

Zoznam súčiastok 4

Označenie	*Hodnota*	*Počet kusov*
LED1	červená 5mm LED dióda	1
R1	Rezistor 150R	1
R2	Rezistor 10k	1
S1	Tlačítko	1
Breadboard1	Kontaktné pole	1
Arduino1	Arduino UNO R3	1
Prepojovacie káble		

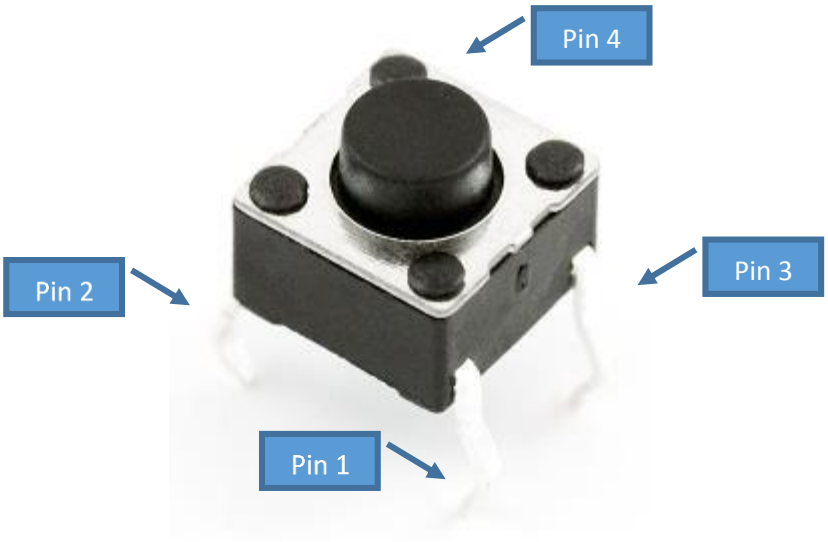
V zozname súčiastok (Zoznam súčiastok 4) nám pribudlo jedno tlačítko. Toto tlačítko je normálnom stave vypnuté. Keď ho stlačíme kontakty sa zopnú a prúd nim môže pretekať. Takéto tlačítko môžeme „odpájkovať“ zo starej nefunkčnej elektroniky, alebo keď nemáte nič na „rozobratie“ je ho možné kúpiť buď v kamennom obchode s elektro súčiastkami alebo v niektorom z eshopov.
Ďalej si popíšeme niečo o tlačítkách. Tlačítko samo o sebe nie je vôbec zložitá súčiastka, jeho funkciou je spínať alebo rozpínať 1 pár alebo viac párov kontaktov v elektrickom obvode. Na obrázku 7-1 máme schematickú značku jednoduchého tlačítka (spínača).

Piny na tlačítku sú prepojené nasledovne: Pin 1 s pinom 2 a Pin 3 s pinom 4, ako je vidieť na obrázku 7-1.



Obrázok 7-1. Schematická značka tlačítka.

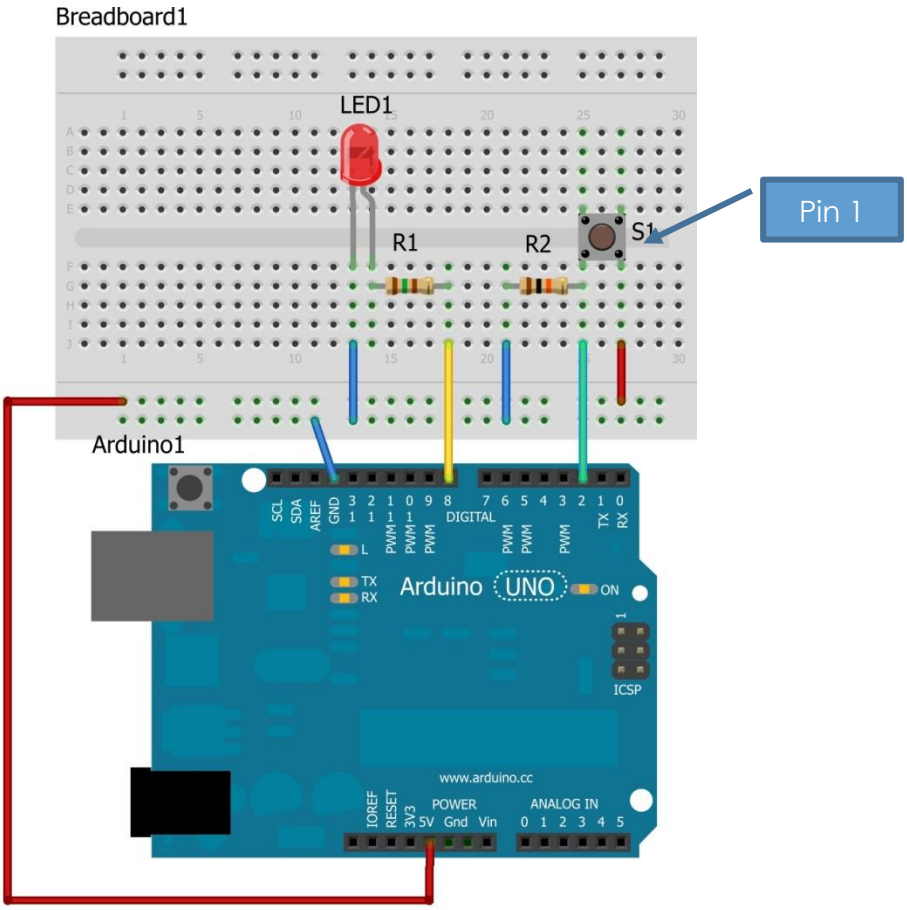
Na obrázku 7-2 je možné vidieť piny na reálnom tlačítku.

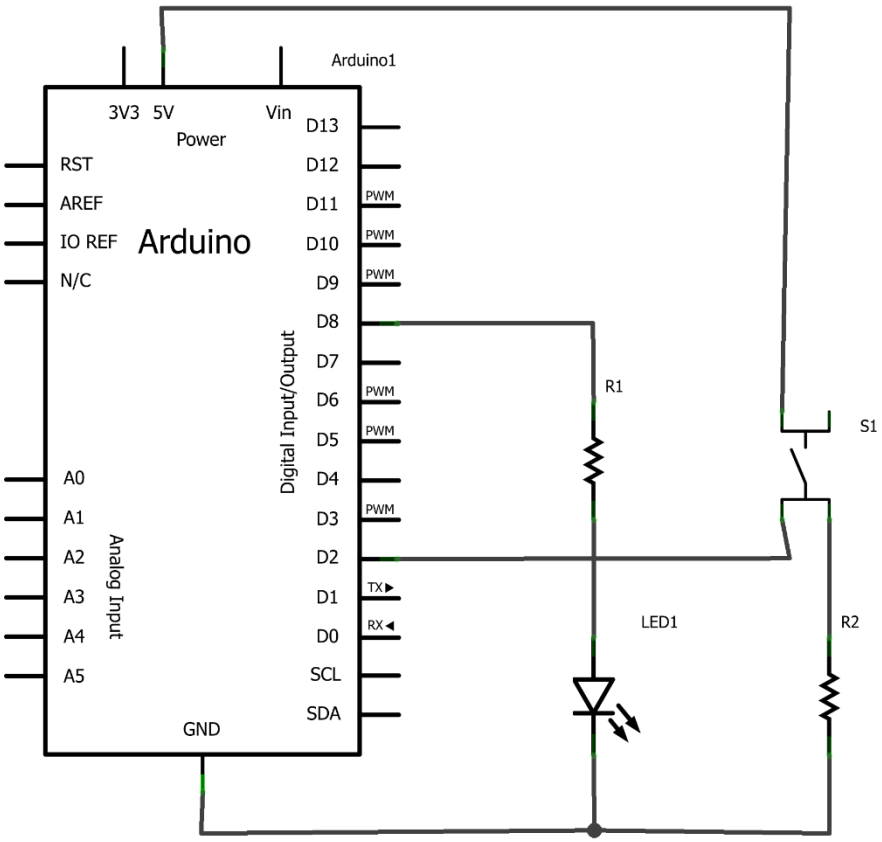


Obrázok 7-2. Obrázok tlačítka a popis jeho vývodov.

7.2. Zapojenie

Na obrázku 7-3 je zapojenie súčiastok pre interaktívnu LEDku.





Obrázok 7-3. Zapojenie súčiastok pre interaktívnu LEDku.

Arduino odpojíme od zdroja napätia (ak používate ako zdroj napätia USB port, vytiahneme Arduino z USB portu). Všetky súčiastky pozapájame podľa obrázka 26.

7.3. Popis zapojenia

Zapojenie je jednoduché, samozrejme treba si dať pozor na správnu polaritu LED diódy a tiež správne zasunúť tlačítko S1. Aby ste to mali jednoduchšie je na obrázku naznačené kde sa nachádza pin 1.

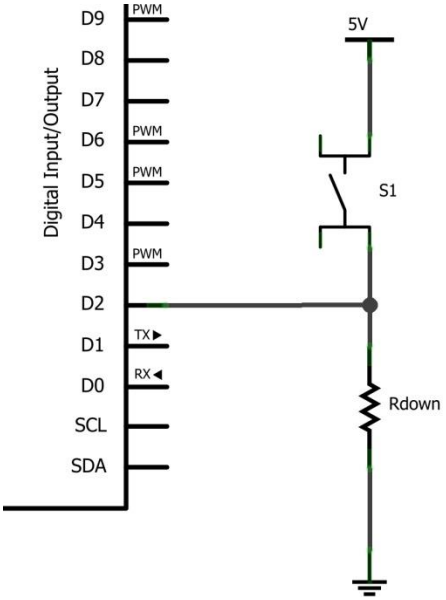
7.3.1. Pull down a pull up rezistory

Možno si kladiete otázku: „Načo slúži rezistor R2 ?“. Rezistor R2 plní funkciu tzv. pull down rezistora. Okrem pull down existuje aj pull up rezistor, a načo vlastne sú?

Ak je digitálny pin nastavený ako vstupný, tak na vstupe tohto pinu sa očakávajú hodnoty HIGH (1) alebo LOW (0), tzn. musí byť určený (počiatočný) stav. Ak by sme na tento pin pripojili len tlačítko, je možné že na vstupe sa môže vyskytnúť nerozlíšiteľný stav. A aký to je ten nerozlíšiteľný stav? Je to napätie v intervale od 0,8 do 2 voltov. Teda ak máme na vstupe takéto napätie nevieme určiť či ide o logickú 1 alebo logickú 0. Arduino by nevedelo určiť stav na pine a tím by nepracovalo korektne. V takomto prípade by sa náš projekt správal inak ako by sme chceli.

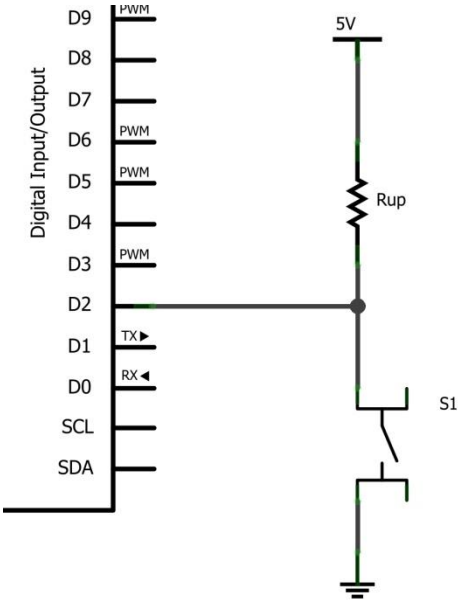
Aby sme zabezpečili na vstupoch rozlíšiteľný stav (1 alebo 0), tak na vstup zapojíme rezistor nasledovne:

Ešte predtým si musíme určiť, ktorý zo stavov (1 alebo 0) bude pre nás aktívny, teda na základe ktorého vykonáva Arduino nejaké príkazy, podľa programu ktorý sme doňho nahrali. Ak si zoberieme náš projekt, tak pre nás je aktívna hodnota HIGH teda 1 (tlačítko S1 pripojí vstupný pin na napätie 5V). Teda logicky ak tlačítko nie je zopnuté musí byť na vstupe logická 0 a to zabezpečíme tak, že vstupný pin pripojíme cez rezistor na zem(0V). Takýto rezistor sa nazýva **pull down** rezistor viď obrázok 7-4.



Obrázok 7-4. Zapojenie pull down rezistora na vstupe.

V opačnom prípade, ak si zvolíme ako aktívnu hodnotu logickú 0, teda stav LOW. Budeme musieť vstup pripojiť cez rezistor na napätie 5V, aby sme dosiahli toho, že ak tlačítko nie je zopnuté bude na vstupe stav HIGH (logická 1). Tento rezistor sa nazýva **pull up,** viď obrázok 7-5.



Obrázok 7-5. Zapojenie pull up rezistora na vstupe.

Hodnota pull up a pull down rezistorov by nemala byť príliš malá aby nimi nepretekal zbytočne veľký prúd. Zvyčajne sa volí v intervale od 5kΩ až 10kΩ. V našom prípade sme zvolili hodnotu pull down rezistora 10kΩ.

7.4. Kód programu

Ak máme všetko pozapájané, môžeme sa pustiť do programovania Arduina. Do vývojového prostredia vložíme kód skice 6. Skicu skompilujeme a nahráme do Arduina.

```
//číslo pinu pre tlačítko
int buttonPin = 2;
//číslo pinu pre LEDku
int ledPin =  8;
//premenná uchovávajuca stav tlačítka
int buttonState = 0;

void setup() {
  // nastavenie pinu pre LEDku ako výstupný
  pinMode(ledPin, OUTPUT);
  // nastavenie pinu pre tlačítko ako vstupný
  pinMode(buttonPin, INPUT);
}

void loop(){
  // načítame stav tlačítka
  buttonState = digitalRead(buttonPin);

  // skontrolujeme či je tlačitko stlačené alebo nie
  // ak je, tzn. premenná buttonState má hodnotu HIGH
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH); // zapneme LEDku
  } else {
    // ak tlačítko nie je zopnuté (stlačené), LEDku vypneme
    digitalWrite(ledPin, LOW);
  }
}
```

Skica 6. Zdrojový kód skice pre interaktívnu LEDku.

Ak stlačíme tlačítko S1, LED 1 by mala zasvietiť. Ak tlačítko pustíme LEDka zhasne.

7.5. Popis programu

Na začiatku sa nám inicializujú premenné buttonPin, ledPin a buttonState.

```
//číslo pinu pre tlačítko
int buttonPin = 2;
//číslo pinu pre LEDku
int ledPin =  8;
//premenná uchovávajuca stav tlačítka
int buttonState = 0;
```

Popis premenných je možné vidieť v komentároch nad nimi. Túto časť kódu už dobre poznáte z predchádzajúcich projektov, tak sa ďalším popisom nemusíme zaoberať.

V setup metóde ako zvyčajne nastavujeme mód jednotlivých pinov.

```
void setup() {
  // nastavenie pinu pre LEDku ako výstupný
  pinMode(ledPin, OUTPUT);
  // nastavenie pinu pre tlačítko ako vstupný
  pinMode(buttonPin, INPUT);
}
```

V tomto projekte máme malú zmenu a to tím, že nastavujeme pin 2 ako vstupný. Na tento pin je zapojené totiž tlačítko, ktorého stav budeme sledovať.

V loop metóde máme použitú novú metódu digitalRead. Táto metóda nám vracia stav na určitom pine, ktorý zadáme ako jediný parameter.

```
int pinState = digitalRead(pinNumber);
```

V našej skice máme ako parameter zadanú premennú buttonPin, teda pin číslo 2.

```
buttonState = digitalRead(buttonPin);
```

Stav pinu, ktorý nám vráti metóda digitalRead sa nám uloží do premennej buttonState.V ďalšej časti kódu porovnávame tento stav, tzn. zisťujeme či bolo tlačítko stlačené alebo nie.

```
if (buttonState == HIGH) {
  digitalWrite(ledPin, HIGH); // zapneme LEDku
} else {
   // ak tlačítko nie je zopnuté (stlačené), LEDku vypneme
   digitalWrite(ledPin, LOW);
}
```

Ak je premenná buttonState rovná HIGH znamená to, že tlačítko je stlačené a tak sa vykoná kód:

```
digitalWrite(ledPin, HIGH);
```

A vy už viete, že táto časť kódu spôsobí to že nám LEDka bude svietiť. Ale ak sa stav tlačítka nezhoduje z HIGH stavom, čo znamená, že tlačítko nie je zopnuté, vykoná sa kód v bloku **else**:

```
digitalWrite(ledPin, LOW);
```

Čo spôsobí, že LEDka nebude svietiť, resp. zhasne, ak sme tlačítko pustili.

7.6. Na záver projektu

V tejto kapitole sme sa naučili správne pripojiť tlačítko k Arduinu. Vysvetlili sme si čo je to pull down a pull up rezistor a akú plní funkciu v číslicových obvodoch.

8. Projekt 6: Arduino ako generátor zvuku

S Arduinom je možné jednoducho generovať rôzne zvuky a tóny. My si v tejto kapitole vytvoríme taký jednoduchý generátor tónu.

8.1. Zoznam súčiastok

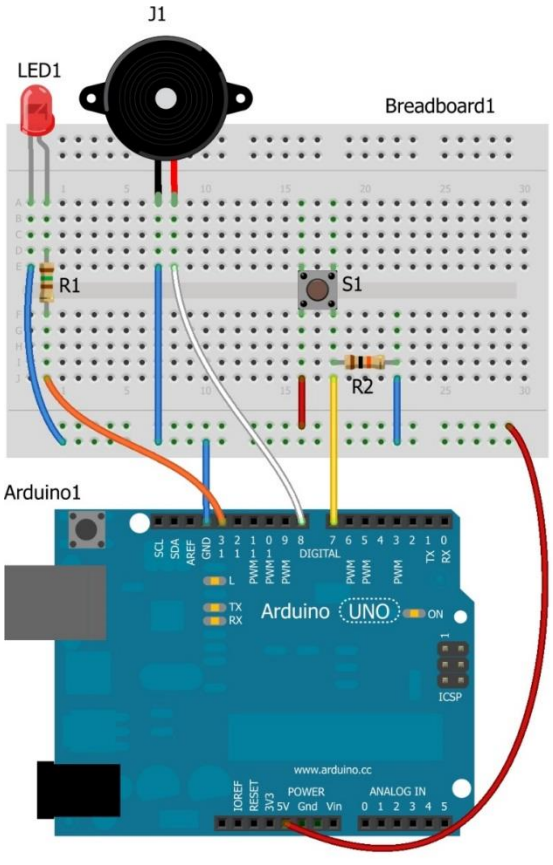
V tomto projekte si k Arduinu pripojíme piezo (alebo reproduktor), LEDku a tlačítko. Po stlačení tlačítka zasvieti LEDka a piezo vydá jednoduchý tón o frekvencii 1kHz. Zoznam všetkých súčiastok máme v tabuľke „Zoznam súčiastok 5".

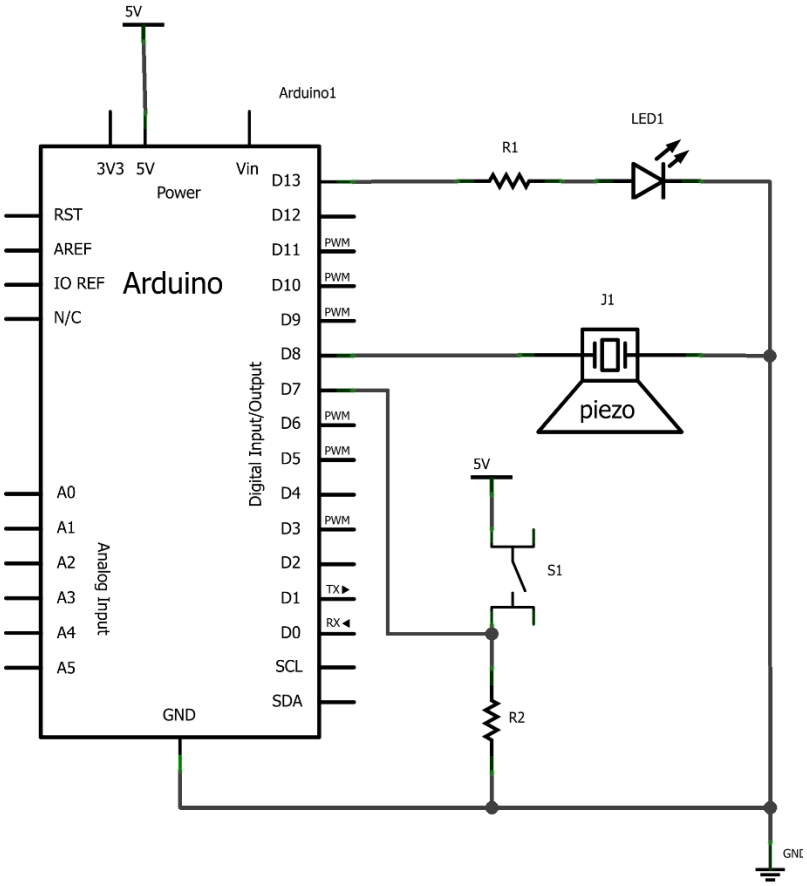
Zoznam súčiastok 5

Označenie	*Hodnota*	*Počet kusov*
LED1	červená 5mm LED dióda	1
R1	Rezistor 150R	1
R2	Rezistor 10k	1
S1	Tlačítko	1
J1	Piezo menič, alebo reproduktor	1
Arduino1	Arduino UNO R3	1
Prepojovacie káble		

8.2. Zapojenie

Ako v predchádzajúcich projektoch najprv odpojíme Arduino od zdroja napätia a pozapájame súčiastky podľa nasledujúceho obrázka.





Obrázok 8-1. Zapojenie súčiastok jednoduchého generátora.

8.3. Popis zapojenia

Toto zapojenie je už o niečo zložitejšie, ako predchádzajúce. Zapojenie LEDky a tlačítka je v podstate rovnaké ako v predchádzajúcom projekte. Pribudlo nám tu tentokrát piezo, označené J1. Piezo J1 má dva vývody, jeden je označený čiernou a druhý červenou farbou. Vývod s čiernou farbou pripojíme na zem a vývod s červenou farbou na Arduino, na pin číslo 8. Ak vývody vášho piezomeniča nie sú farebne odlíšené, zapojte vývody ľubovoľne (nebojte sa nič tým nepokazíte).

8.4. Kód programu

Po dokončení zapojenia, môžeme začať programovať. Nasledujúcu skicu vložíme do vývojového prostredia, skompilujeme a nahráme do Arduina.

```
// číslo pinu pre LED
int ledPin = 13;
// číslo pinu pre tlačítko
int buttonPin = 7;
// číslo pinu pre piezo
int piezoPin =  8;
// premenná uchovávajuca stav tlačítka
int buttonState = 0;

void setup() {
  // nastavenie pinu pre LEDku ako výstupný
  pinMode(ledPin, OUTPUT);
  // nastavenie pinu pre piezo ako výstupný
  pinMode(piezoPin, OUTPUT);
  // nastavenie pinu pre tlačítko ako vstupný
  pinMode(buttonPin, INPUT);
}

void loop(){
  // načítame stav tlačítka
  buttonState = digitalRead(buttonPin);

  // skontrolujeme či je tlačitko stlačené
  // ak je, tzn. premenná buttonState má hodnotu HIGH
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH); // zapneme LEDku
    tone(piezoPin, 1000, 100);
    delay(100);
  } else { // ak nie je, tzn. premenná buttonState má hodnotu LOW
    digitalWrite(ledPin, LOW); // vypneme LEDku
    noTone(piezoPin); // vypneme generovanie tónu
  }
}
```

Skica 7. Zdrojový kód skice pre generátor tónu.

8.5. Popis programu

Kód skice sa veľmi nelíši od toho predchádzajúceho. Inicializácia premenných je skoro rovnaká, akurát nám pribudla premenná piezoPin, ktorá má hodnotu 8. V setup metóde tak následne pribudlo nastavenie pinu pre piezo:

```
// nastavenie pinu pre piezo ako výstupný
pinMode(piezoPin, OUTPUT);
```

V loop metóde je to tiež veľmi podobné, máme tu ešte dve nové metódy a to **tone** a **noTone**. Metóda tone vygeneruje jednoduchý tón za pomoci PWM. Má tri vstupné parametre z toho je posledný nepovinný.

```
tone(pin, frekvencia, trvanie);
```

alebo

```
tone(pin, frekvencia);
```

Prvý parameter **pin**, je číslo výstupného pinu na ktorom sa bude generovať tón. Druhý parameter **frekvencia**, je frekvencia tónu v Hz (Hertz), v našej skice máme frekvenciu 1000Hz (1kHz). Posledný parameter ktorý je nepovinný je **trvanie** tónu v milisekundách, v našom prípade 100ms.

```
tone(piezoPin, 1000, 100);
```

Metóda noTone ukončí generovanie tónu na pine, ktorý zadáme ako vstupný parameter do tejto metódy.

```
noTone(pin);
```

V našom projekte voláme metódu noTone, kde ako hodnotu parametra predávame hodnotu z premennej piezoPin.

```
noTone(piezoPin);
```

Čo znamená, že ak tlačítko nie je zopnuté generovanie tónu sa ukončí.

8.6. Na záver projektu

V tejto kapitole sme si ukázali ako jednoducho sa dá s Arduinom generovať ľubovoľný tón. Projekt je samozrejme ukážkový, je ho možné ďalej upravovať a prispôsobovať vašim potrebám.

9. Projekt 7: Arduino a LED displej

V tejto kapitole si vysvetlíme ako funguje sedemsegmentovka, teda LED Display. Ukážeme si na jednoduchom príklade ako pripojiť LED display priamo k Arduinu.

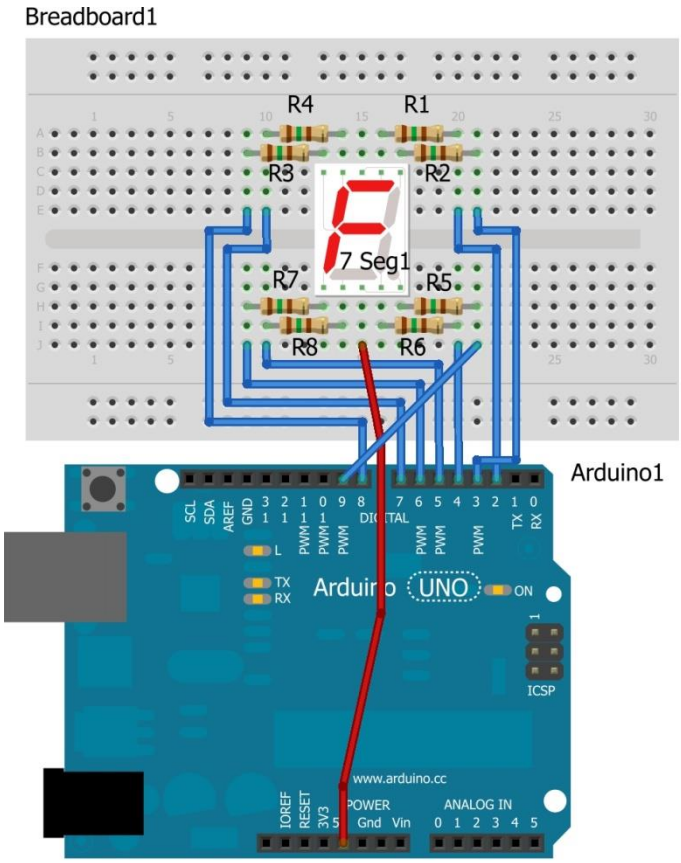
9.1. Zoznam súčiastok

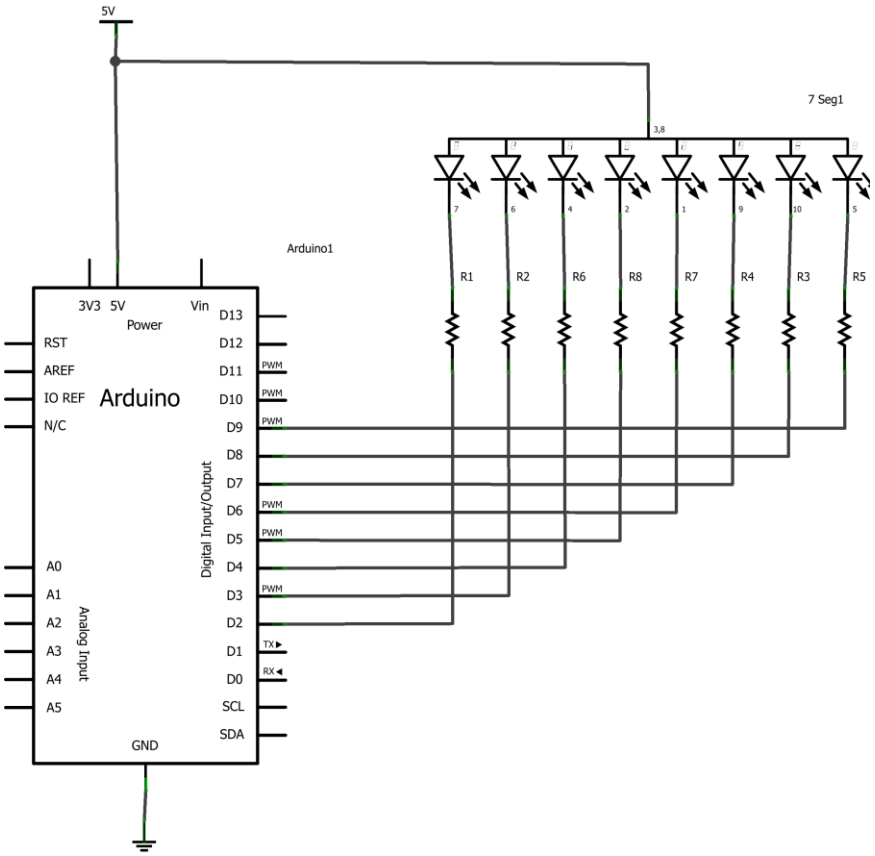
V tomto projekte pripojíme sedemsegmentovku priamo k arduinu. Napíšeme jednoduchý program, ktorý na displayi zobrazí číslice 0 až 9. Zoznam potrebných súčiastok je v tabuľke "Zoznam súčiastok 6".

Zoznam súčiastok 6

Označenie	*Hodnota*	*Počet kusov*
7 Segment1	LED display (7 segmentovka, so spoločnou anódou)	1
R1 až R8	Rezistor 150R	8
Breadboard1	Kontaktné pole	1
Arduino1	Arduino UNO R3	1
Prepojovacie káble		

9.2. Zapojenie





Obrázok 9-1. Zapojenie LED displeja k Arduinu.

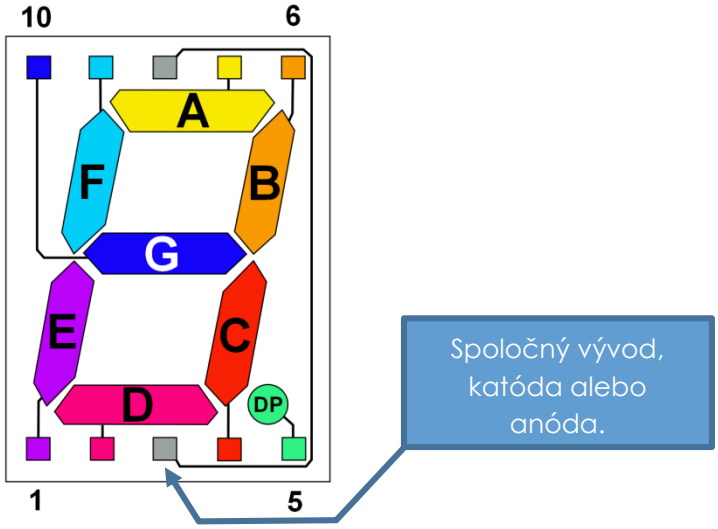
9.3. Popis zapojenia

Zoberieme si súčiastky a pozapájame všetko podľa obrázka 9-1. Komplikácie môžu nastať pri zapájaní LED displeja. Musíme poznať jednotlivé vývody displeja, ktorý vývod patrí ku ktorému segmentu.

9.3.1. Ako funguje LED displej

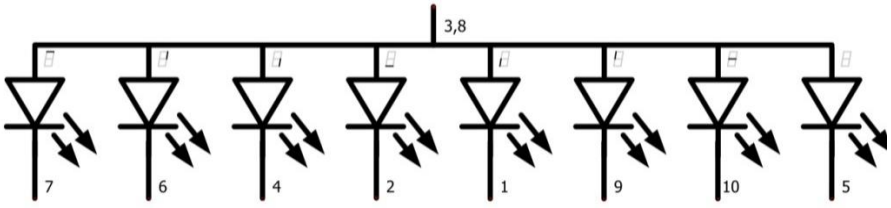
Je čas si teda povedať niečo o sedem segmentových LED displejoch. Sedem segmentový LED display alebo slangovo nazývaná sedemsegmentovka je skupina siedmych LED diód zoskupených určitým spôsobom do jedného celku. Pomocou takéhoto displeja je možné zobraziť číslice 0 až 9. Na obrázku 9-2 sú znázornené jednotlivé segmenty displeja. Každý segment označený písmenom je vlastne jedna LEDka. Často býva v takomto displeji aj

ôsmy segment v tvare bodky označovaný DP (Dot Point) - tento segment sa používa ako desatinná bodka.

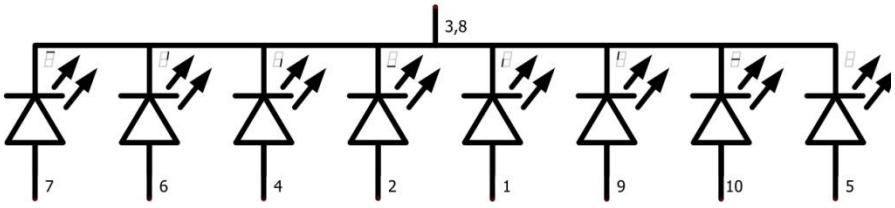


Obrázok 9-2. Popis segmentov a vývodov jednomiestneho LED displeja.

Na obrázku 9-2 sú jednotlivé segmenty označené rôznymi farbami a písmenami **A, B, C, D, E, F a G** (s písmenovým označeným segmentov sa stretnete aj v ďalších odborných textoch alebo datasheetoch, je to **štandartné označenie segmentov**). Na obrázku je tiež vidieť prepojenie medzi segmentami a vývodmi displeja. Aby takýto display nemal priveľa vývodov, tak sa určí **spoločný vývod** pre anódu alebo katódu, tzn. spoja sa buď všetky anódy segmentov alebo všetky katódy, podľa toho môžeme LED displeje rozdeliť: so **spoločnou anódou** alebo so **spoločnou katódou**. Na obrázkoch 9-3 a 9-4 sú znázornené vnútorné zapojenia jednomiestnych LED displejov.

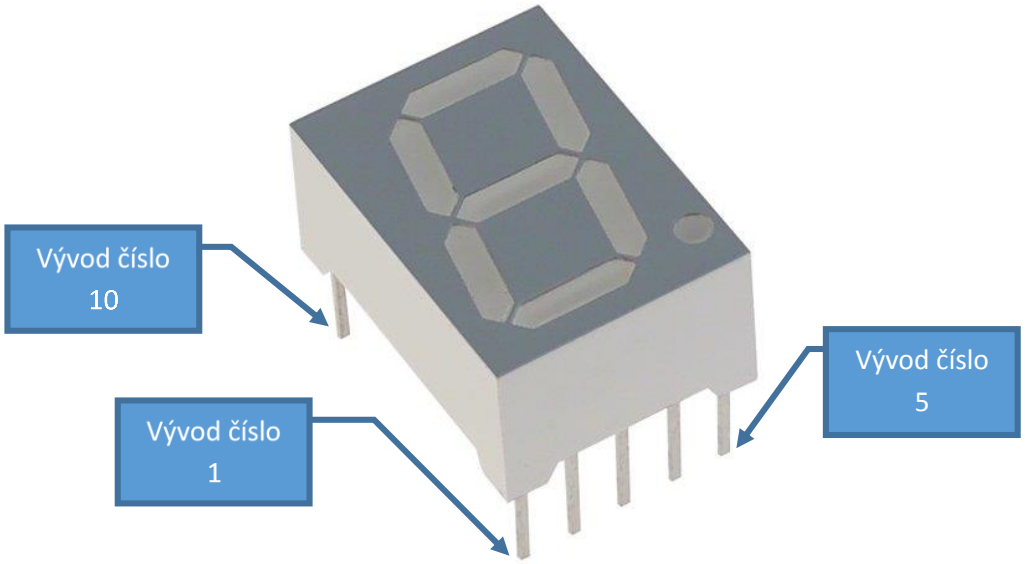


Obrázok 9-3. Vnútorné zapojenie jednomiestneho LED displeja so spoločnou anódou.



Obrázok 9-4. Vnútorné zapojenie jednomiestneho LED displeja so spoločnou katódou.

Čísla pri katódach alebo na anódach LED diód na obrázkoch 9-3 a 9-4, sedia s číslami vývodov na obrázku 9-5.



Obrázok 9-5. Jednomiestny 7 segmentový LED display.

Na obrázku 9-5 je znázornenie vývodov na reálnom displeji.

Ešte sa vrátim k zapojeniu, rezistory R1 až R8 plnia funkciu predradných rezistorov pre jednotlivé segmenty (teda LEDky) v LED displeji.

9.4. Kód programu

Myslím, že o sedem segmentovke sme si povedali dosť na to aby sme ju vedeli zapojiť v našom projekte, preto by sme mohli prejsť k programovaniu.

```
 // čísla pinov segmentov
byte segmentPins[] =
{2/*A*/,3/*B*/,4/*C*/,5/*D*/,6/*E*/,7/*F*/,8/*G*/,9/*BODKA*/};
// Pole aktívnych segmentov pre jednotlivé číslice, LOW = svieti, HIGH = ne
svieti
byte segmentDigits[10][8] = {
//{ A  , B , C , D , E , F , G , BODKA }
  {LOW, LOW, LOW, LOW, LOW, LOW, HIGH, HIGH},     // číslica 0
  {HIGH, LOW, LOW, HIGH, HIGH, HIGH, HIGH, HIGH}, // číslica 1
  {LOW, LOW, HIGH, LOW, LOW, HIGH, LOW, HIGH},    // číslica 2
  {LOW, LOW, LOW, LOW, HIGH, HIGH, LOW, HIGH},    // číslica 3
  {HIGH, LOW, LOW, HIGH, HIGH, LOW, LOW, HIGH},   // číslica 4
  {LOW, HIGH, LOW, LOW, HIGH, LOW, LOW, HIGH},    // číslica 5
  {LOW, HIGH, LOW, LOW, LOW, LOW, LOW, HIGH},     // číslica 6
  {LOW, LOW, LOW, HIGH, HIGH, HIGH, HIGH, HIGH},  // číslica 7
  {LOW, LOW, LOW, LOW, LOW, LOW, LOW, HIGH},      // číslica 8
  {LOW, LOW, LOW, LOW, HIGH, LOW, LOW, HIGH}      // číslica 9
};

/* Metóda pre zobrazenie číslice */
void segmentDisplay(int digit);

void setup(){
  // nastavíme porty ako vystupne
  for(int i=0;i<8;i++){
    pinMode(segmentPins[i], OUTPUT);
  }
  // nastavíme počiatočné hodnoty, nesvieti žiadna číslica
  for(int i=0;i<8;i++){
    digitalWrite(segmentPins[i], HIGH);
  }
}

void loop(){
// Postupne zobrazíme čísla od 0 po 9
  for(int n=0; n<10; n++){
    segmentDisplay(n);
    delay(500);
  }
// Opačný postup, zobrazíme čísla od 9 po 0
  for(int m=9; m>=0; m--){
    segmentDisplay(m);
    delay(500);
  }
}

void segmentDisplay(int digit){
    for(int s=0; s<8; s++){
      digitalWrite(segmentPins[s], segmentDigits[digit][s]);
    }
}
```

Skica 8. Kód skice pre ovládanie jednomiestneho sedem segmentového LED displeja.

Skicu 8 skopírujeme do vývojového prostredia a nahráme do Arduina. Ak máme všetko správne zapojené na displeji by sa mali postupne zobrazovať číslice 0 až 9 a následne opačne 9 až 0. Celý cyklus sa neustále opakuje.

9.5. Popis programu

Teraz si rozoberieme kód skice 8. Na začiatku máme inicializované dve premenné segmentPins a segmentDigits. Premenná segmentPins je pole typu byte a uchováva čísla výstupných pinov na ktoré máme pripojenú sedem segmentovku.

```
byte segmentPins[] =
{2/*A*/,3/*B*/,4/*C*/,5/*D*/,6/*E*/,7/*F*/,8/*G*/,9/*BODKA*/};
```

V komentároch za číslom je segment ku ktorému je pin pripojený.
Druhá premenná segmentDigits je tiež pole typu byte ale dvojrozmerné, je to vlastne pole polí, v ktorých sú zadefinované hodnoty HIGH a LOW pre jednotlivé piny ku každej číslice.

```
byte segmentDigits[10][8] = {
//{ A   , B , C , D , E , F , G , BODKA }
  {LOW, LOW, LOW, LOW, LOW, LOW, HIGH, HIGH},      // číslica 0
  {HIGH, LOW, LOW, HIGH, HIGH, HIGH, HIGH, HIGH}, // číslica 1
  {LOW, LOW, HIGH, LOW, LOW, HIGH, LOW, HIGH},     // číslica 2
  {LOW, LOW, LOW, LOW, HIGH, HIGH, LOW, HIGH},     // číslica 3
  {HIGH, LOW, LOW, HIGH, HIGH, LOW, LOW, HIGH},    // číslica 4
  {LOW, HIGH, LOW, LOW, HIGH, LOW, LOW, HIGH},     // číslica 5
  {LOW, HIGH, LOW, LOW, LOW, LOW, LOW, HIGH},      // číslica 6
  {LOW, LOW, LOW, HIGH, HIGH, HIGH, HIGH, HIGH},   // číslica 7
  {LOW, LOW, LOW, LOW, LOW, LOW, LOW, HIGH},       // číslica 8
  {LOW, LOW, LOW, LOW, HIGH, LOW, LOW, HIGH}       // číslica 9
};
```

Toto pole má 10 riadkov (teda polí) pre každú jednu číslicu. Každé toto pole má zase 8 prvkov (teda toľko, koľko je pinov), prvky môžu mať hodnotu LOW alebo HIGH.
Rozoberme si prvý riadok v poli:

```
//{ A   , B , C , D , E , F , G , BODKA }
  {LOW, LOW, LOW, LOW, LOW, LOW, HIGH, HIGH},      // číslica 0
```

Prvý riadok určuje nastavenie pinov pre číslicu 0 (nula). Aby sa segment pripojený k pinu rozsvietil musí byť pripojený k zemi teda na 0V. Pretože máme použitý LED display so spoločnou anódou, tak vývody segmentov sú vlastne katódy LED diód v segmentoch, tzn. tieto katódy musia byť pripojené na záporný pól napätia a to docielime tým, že na výstupný pin zapíšeme hodnotu LOW. Teda bude platiť LOW = svieti, HIGH = nesvieti. Podľa tohto vidíme, že by mali svietiť segmenty A, B, C, D, E, a F. Segmenty G a BODKA majú nastavenú hodnotu HIGH teda nesvietia.
Ďalšie riadky v poli, resp. ďalšie polia určujú nastavenie pinov analogicky pre ostatné číslice 1, 2 až 9.

Ďalej máme definovanú metódu segmentDisplay:

```
void segmentDisplay(int digit);
```

Takýmto zápisom sme si zadefinovali novú metódu, ktorú môžeme neskoršie zavolať (použiť) v hocikde v programe. Ešte predtým je nutné urobiť nasledujúce:

```
void segmentDisplay(int digit){
    for(int s=0; s<8; s++){
      digitalWrite(segmentPins[s], segmentDigits[digit][s]);
    }
}
```

A to, zapísať telo metódy tzv. implementáciu. Implementácia je zapísaná za metódou loop. Je to vlastne to, čo má metóda vlastne robiť. V krátkosti metóda nastaví piny tak aby sa rozsvietila konkrétna číslica na displeji. Metóda má jeden parameter typu int (integer). Týmto parametrom udávame, ktorá z číslic (0-9) sa má na displeji zobraziť.

A teraz sa vrátime ešte k metóde setup, kde sa nám nastavujú výstupné piny pre display a tiež nastavujeme počiatočné hodnoty pinov, tak aby nám na displeji nesvietilo žiadne číslo.

```
// nastavíme porty ako výstupné
for(int i=0;i<8;i++){
  pinMode(segmentPins[i], OUTPUT);
}
// nastavíme počiatočné hodnoty, nesvieti žiadna číslica
for(int i=0;i<8;i++){
  digitalWrite(segmentPins[i], HIGH);
}
```

V loop metóde máme dva cykly for. Prvý cyklus zobrazí postupne na displeji číslice od 0 po 9.

```
// Postupne zobrazíme čísla od 0 po 9
for(int n=0; n<10; n++){
  segmentDisplay(n);
  delay(500);
}
```

Druhý cyklus spraví analogicky to isté len opačným smerom zobrazí číslice od 9 po 0.

```
// Opačný postup, zobrazíme čísla od 9 po 0
for(int m=9; m>=0; m--){
  segmentDisplay(m);
  delay(500);
}
```

Zobrazenie každej číslice v oboch cykloch trvá pol sekundy, čo je spôsobené volaním delay metódy vo vnútri cyklu (`delay(500)`).

9.6. Na záver projektu

V tejto kapitole sme si ukázali ako jednoducho sa dá pripojiť jednomiestny sedem segmentový LED display k Arduinu. Napísali sme si jednoduchý program, ktorý nám na displeji zobrazil číslice 0 až 9. Zapojenie ako aj program je ukážkový a je možné ďalej zapojenie ako aj skicu upravovať podľa potrieb.

Zhrnutie a záver

Tak a je tu koniec knihy. Ak ste sa dočítali až sem, tak je to super, pretože už teraz viete viac ako ostatný. Ako ste mohli vidieť s Arduinom je možné veľa vecí. V tejto knihe sú však popísané len niektoré, preto sa budem snažiť vám priniesť ešte viac kvalitných informácii a skúseností o tejto platforme.

Ak máte nejaké otázky alebo problémy s programovaním rád vám poradím. Budem tiež rád ak mi dáte nejakú spätnú väzbu, či sa vám ebook páčil alebo máte nejaké výhrady. Môžete ma kontaktovať na info@mirobozik.sk.

Nemilovať knihy znamená nemilovať múdrosť. Nemilovať múdrosť však znamená stať sa hlupákom.

Ján Amos Komenský