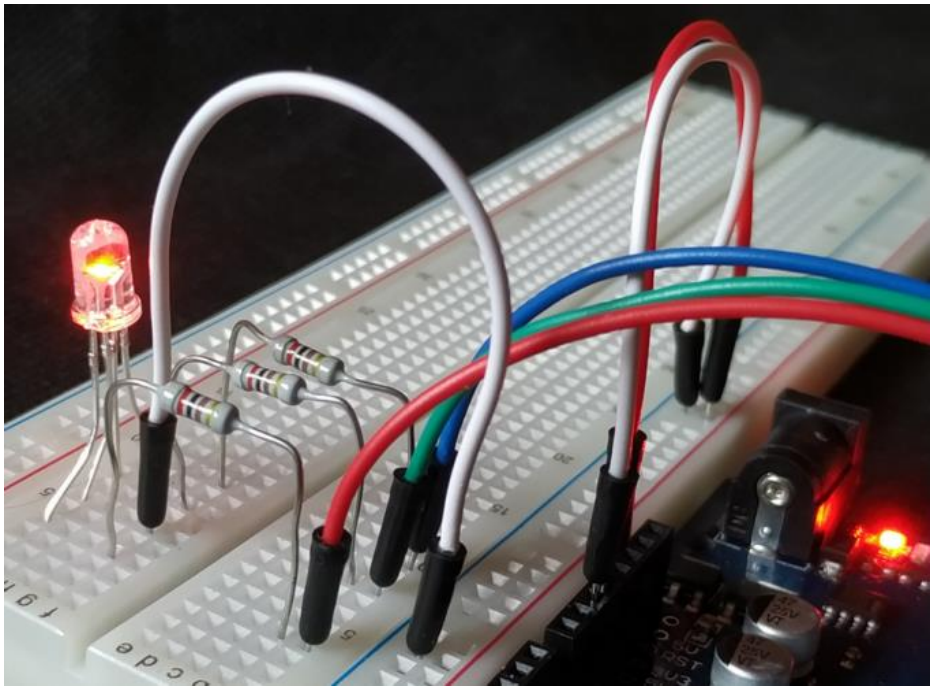
Arduino

od začátku s příklady



Martin Gembec 2020

inspirace: Elduino + moje věčné začátky

Informace zde uvedené vznikaly postupně s cílem pomoci sobě samému, jako věčnému začátečníkovi s Arduinem, abych už příště pronikal základy s Arduinem rychleji. Zároveň věřím, že se podle nich může rychle zorientovat i někdo další, třeba z kroužku, který se zabývá robotikou.

Existuje samozřejmě i česká kniha, která vás podrobně provede základy Arduina. Dá se zakoupit v tištěné verzi za 220 Kč nebo je za jistých podmínek i ke stažení zdarma: https://bastlirna.hwkitchen.cz/

Základní informace

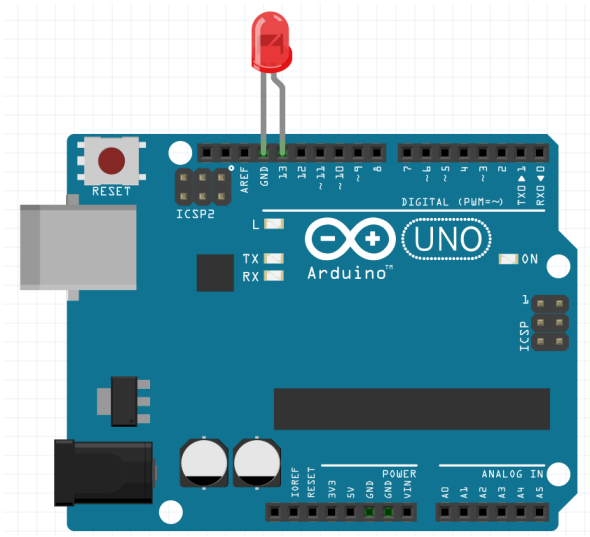
Pro nahrávání programů do Arduina potřebujeme programovací prostředí. My budeme využívat IDE. Pokyny jsou zde: https://bastlirna.hwkitchen.cz/zaciname-s-arduinem/. Pokročilejší si jistě najdou sofistikovanější prostředí než IDE, ale pro začátky nám to bude stačit. Využívat budeme desku Arduino Uno, ale existují i mnohé další, jejichž přehled je třeba na https://blog.laskarduino.cz/.

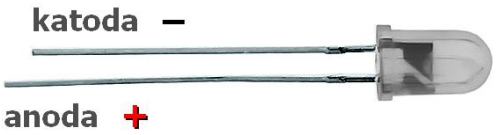
LED – Kontaktní pole – Program Blink

LED (*Light Emitting Diode*) je přímo na desce Arduina v podobě tzv. SMD součástky. Pokud jsme tedy v předchozím kroku nainstalovali IDE a vyzkoušeli program Blink, měla by na desce Arduina LEDka blikat.

Nyní si zkusíme připojit externí LED s „kloboučkem a nožičkami“. Dioda je součástka, u které není jedno, v jakém směru ji zapojíme. Svítí, když je zapojena od + k –.

Proto vždy nožičku + dáváme do očíslovaného (digitálního) pinu a nožičku – dáme do pinu GND. Správně bychom to ale podle níže uvedeného obrázku udělat neměli. LED nevydrží napětí 5 V.

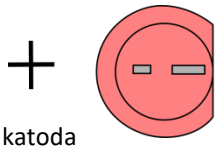




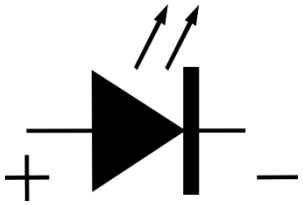
Arduino deska „dodává“ napětí 5 V. Z toho si LED vezme zhruba 2,4 V a zbylých 2,6 V se „spotřebuje“ kde? – Právě od toho slouží rezistor. Ten diodu ochrání a „spotřebuje“ přebytečné napětí.

rezistor 200 Ω ± 5 %

při pohledu shora může být LED useknutá



schematická značka LED



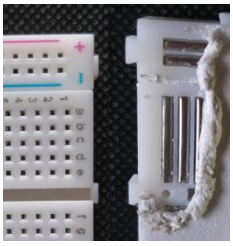
Z Ohmova zákona plyne, že odpor *R* vypočítáme tak, že vydělíme napětí *U* protékajícím proudem *I:*

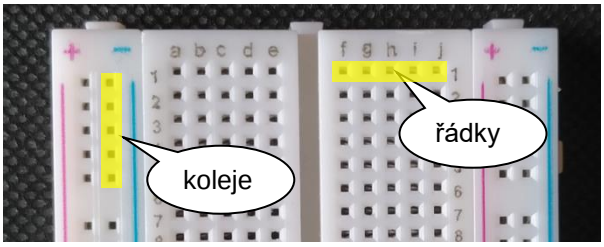
$$R = \frac{U}{I} = \frac{5\ V}{0{,}025\ A} = 200\ \Omega$$

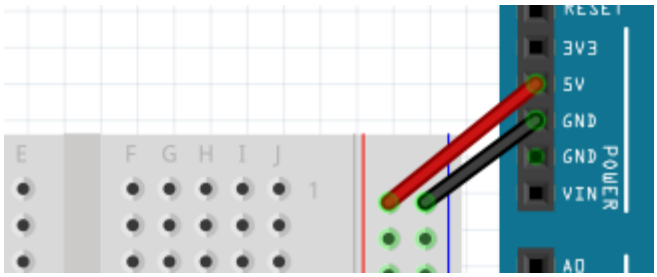
**pro menší proudový odběr doporučujeme spíše 1000 Ω*

Kontaktní (nepájivé) pole / breadboard

K pohodlnému zapojení součástek použijeme kontaktní pole. To je uspořádáno tak, že po stranách jsou navzájem spojené kontakty + a – v tzv. „kolejnicích“ a uvnitř je vždy spojeno 5 konektorů v „řádcích“.



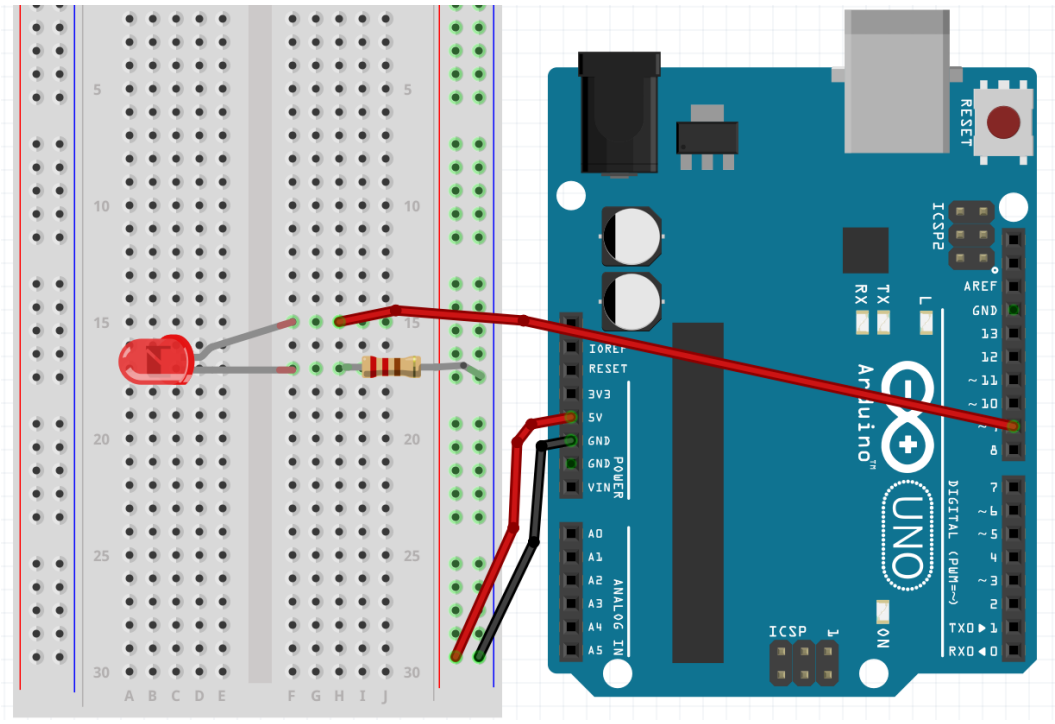


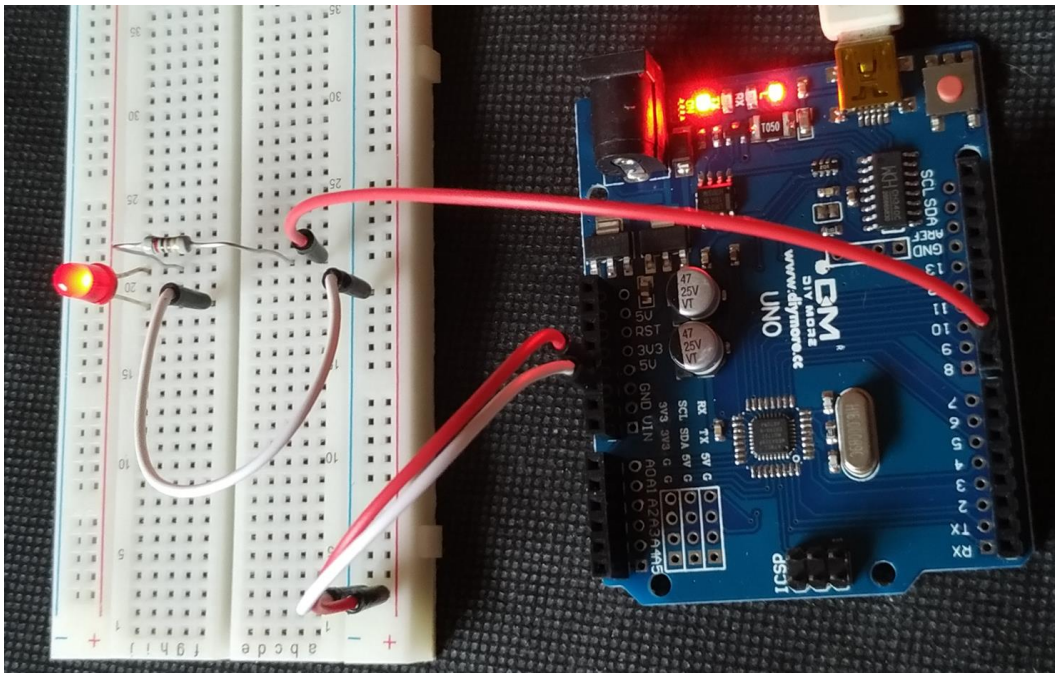


K napájení pole můžeme využít místo baterie přímo Arduino. Jeden vodič spojíme s pinem +5V a druhý s pinem GND. Propojení červeným vodičem ke kladnému pólu není podmínkou, ale je vhodné pro přehlednost používat červenou jako +.

LED na kontaktním poli

Na obrázcích je příklad schematického i skutečného zapojení LED. Jak vidíme, je jedno, zda dáme rezistor ke kladnému, nebo zápornému pólu diody. Důležité je dodržet + / −, jinak dioda nebude svítit.





Všimněte si, že ne vždy je nutno zapojit + kolejnici kontaktního pole (plus není využito pro žádnou součástku). Ale je dobré to dělat, abychom na něj nezapomněli, až bude třeba.

Program Blink

Nejprve si prohlédneme výchozí program Blink (*otevřeme přes nabídku Soubor – Příklady – 01.Basics*).

```
/*
  Blink

  Zapne opakovaně LED na jednu sekundu, pak na sekundu vypne.

  Většina Arduin má na desce napájenou LED na pin 13.
  Číslo pinu ovlivní příkaz LED_BUILTIN.

  modified 8 May 2014, by S. Fitzgerald, 2 Sep 2016, by A. Guadalupi, 8 Sep 2016, by C. Newman

  Uvedený příklad je šířen jako public domain.
*/

// funkce setup se spustí jen jednou, když Arduino resetujete, nebo připojíte k napájení
void setup() {
  // nastaví digitální pin LED_BUILTIN jako výstupní (output).
  pinMode(LED_BUILTIN, OUTPUT);
}

// smyčka (loop) je funkce, která se vykonává stále dokola
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // rozsvítí LED (HIGH = přivede na LED napětí)
  delay(1000);                       // počká jednu sekundu
  digitalWrite(LED_BUILTIN, LOW);    // zhasne LED (LOW = napětí bude nulové)
  delay(1000);                       // počká jednu sekundu
}
```

Vidíme, že je napsaný pro vestavěnou LED. Jestliže chceme, aby stejně fungovala naše LED na kontaktním poli, zapojená do pinu 9, stačí pozměnit kód:

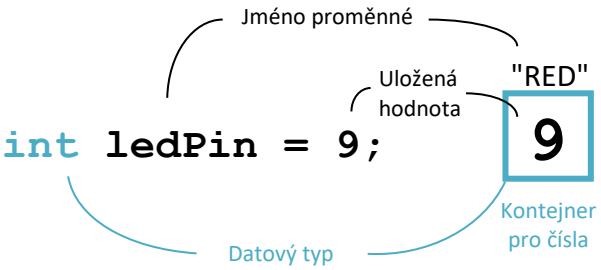
```
void setup() {
  pinMode(9, OUTPUT);
}

void loop() {
  digitalWrite(9, HIGH);
  delay(1000);
  digitalWrite(9, LOW);
  delay(1000);
}
```

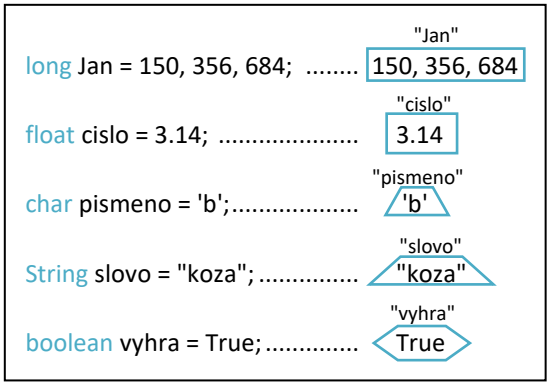
Alternativou je použít proměnné, protože u složitějších programů se v nich lépe vyznáme a ušetří nám čas:

```
int RED = 9;

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

void loop() {
  digitalWrite(RED, HIGH);
  delay(1000);
  digitalWrite(RED, LOW);
  delay(1000);
}
```

Představ si, že se najednou rozhodneš, že červenou LED přepojíš z pinu 9 na pin 3. To je další výhoda proměnné, nemusíš přepisovat číslo pinu všude v kódu, jen změníš hodnotu nahoře.

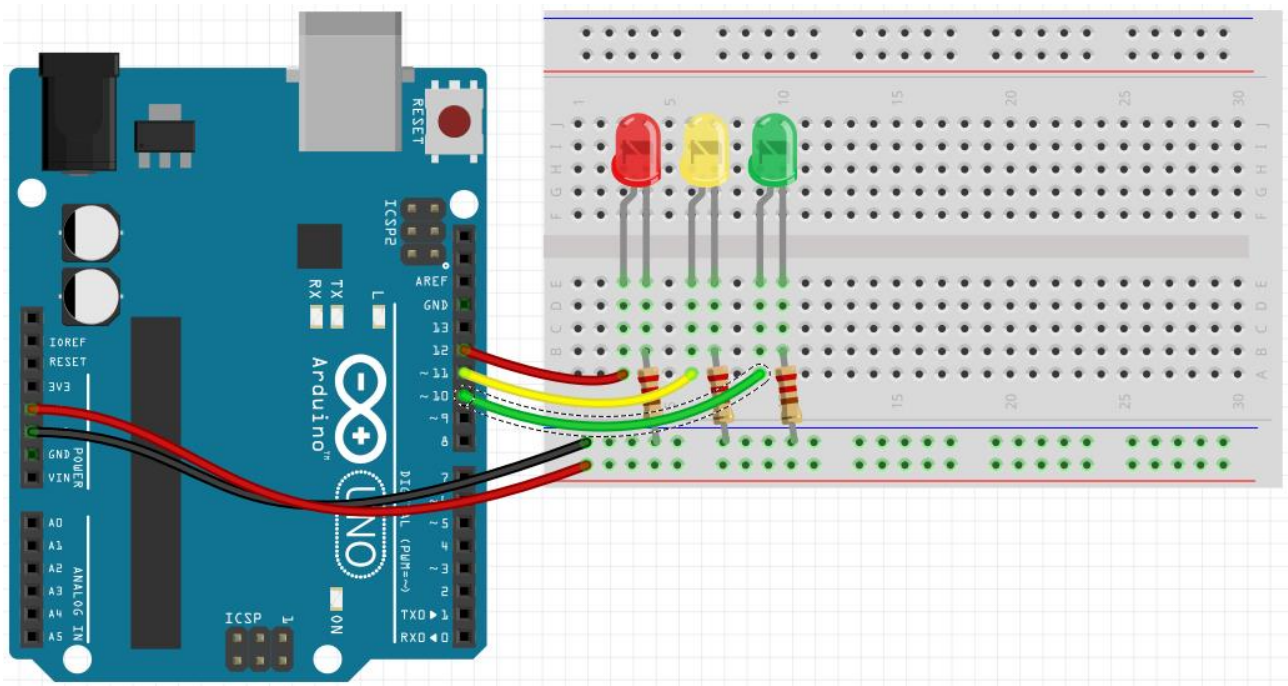
Proměnné





Semafor

Nyní se nabízí vytvořit z LEDek funkční semafor. Víš, jak semafor ve skutečnosti svítí? V jedné situaci svítí současně oranžová i červená (tipni si, řešení je pod obrázkem). Naprogramuj sekvenci od svítící červené, přes svítící zelenou, pořád dokola.



```
int Rpin = 12;
int Ypin = 11;
int Gpin = 10;

void setup() {
  pinMode(Rpin, OUTPUT);
  pinMode(Ypin, OUTPUT);
  pinMode(Gpin, OUTPUT);
}

void loop() {
  digitalWrite(Rpin, HIGH);
  delay(5000);
  digitalWrite(Ypin, HIGH);
  delay(2000);
  digitalWrite(Rpin, LOW);
  digitalWrite(Ypin, LOW);
  digitalWrite(Gpin, HIGH);
  delay(5000);
  digitalWrite(Gpin, LOW);
  digitalWrite(Ypin, HIGH);
  delay(3000);
  digitalWrite(Ypin, LOW);
  digitalWrite(Rpin, HIGH);
  delay(5000);
}
```

Než padne červená, rozsvítí se pouze oranžová a to většinou na dobu 3 sekundy. Naopak než naskočí zelená, svítí současně červená a oranžová po dobu 2 sekund.

LED a tlačítko

Sériový monitor

Jak vlastně tlačítko zapojit? Už víme, že digitální piny fungují jako výstupní pomocí digitalWrite. Nyní je využijeme jako vstupní pomocí digitalRead.

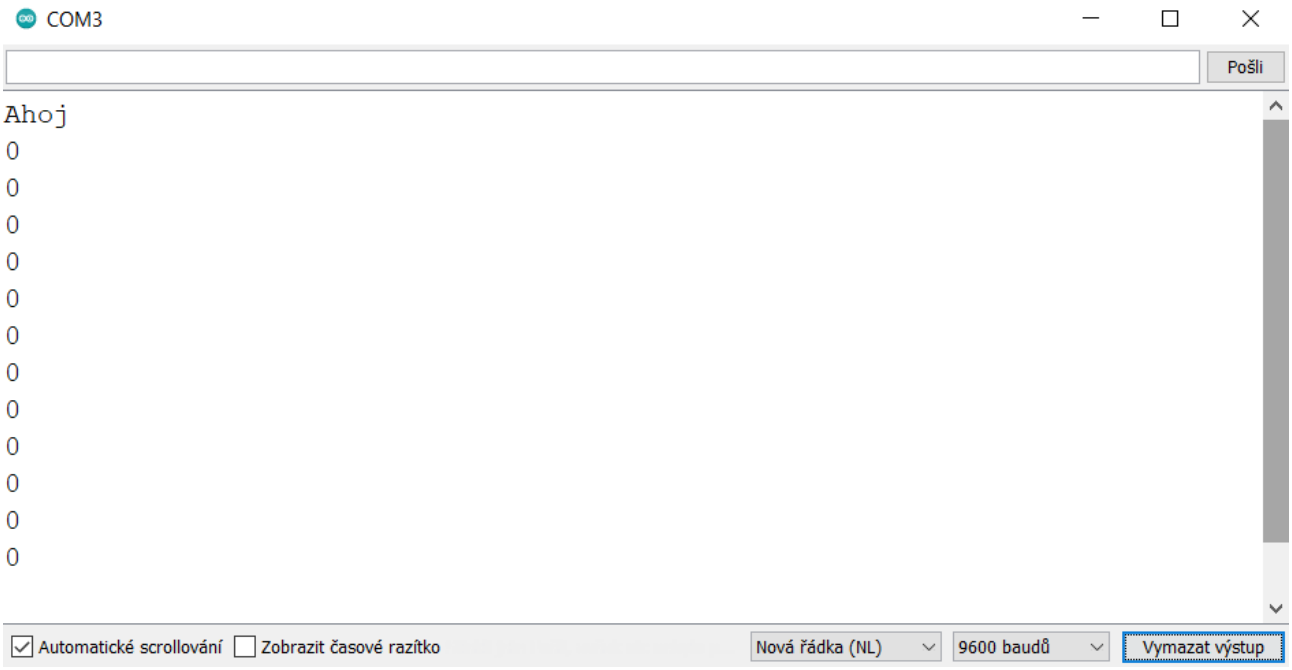
Pojďme se podívat, co „vnímá“ Arduino – využijeme *Sériový monitor*:

```
void setup() {
  Serial.begin(9600);       //9600 udává rychlost komunikace 9600 baudů*
  pinMode(7, INPUT);        //nastavuje pin 7 jako vstupní
  Serial.println("Ahoj");   //pošle do počítače text v uvozovkách; ln znamená piš pak na nový řádek
}

void loop() {
  Serial.println(digitalRead(7));
  delay(50);  //výpis v Sériovém monitoru se obnovuje každých 50 milisekund
}
```

**bity vs. baudy*, Jiří Peterka, 1992*

Nahrajeme program a pustíme *Sériový monitor* z nabídky *Nástroje*, nebo přes *Ctrl+Shift+M*.

K tomu, aby nám vše běželo, musí být deska Arduino samozřejmě připojena k počítači přes USB, které nám emuluje nějaký COM port (zde COM3). Podrobnosti k sériové komunikaci: https://robotika.vosrk.cz/.



Nahoře problikl pozdrav *Ahoj* a nyní vidíme, že na pinu 7 je hodnota 0. Zkusme propojit pin 7 s pinem +5V. Co ukáže Sériový monitor? A co, když drát zapojíme do pinu GND? Nebojte se to zkusit!

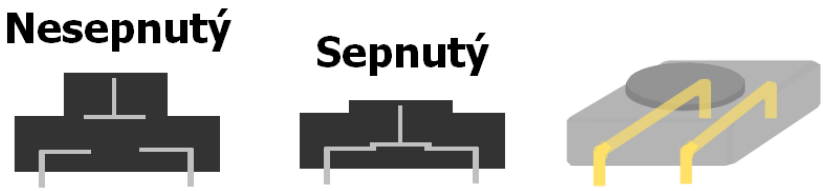
Ano, ukazuje nám to, že vstupní napětí na pinu 7 je *0*, když tam není žádný drát, hodnotu *1*, když je přítomno napětí z pinu +5V a *0* z pinu GND.

Užitečné vědět: Vstupní piny jsou velice citlivé na malé proudy. To znamená, že dokážou zachytit i elektrický šum ze vzduchu. Jak ošetřit tuto situaci si ukážeme později.

Tlačítko

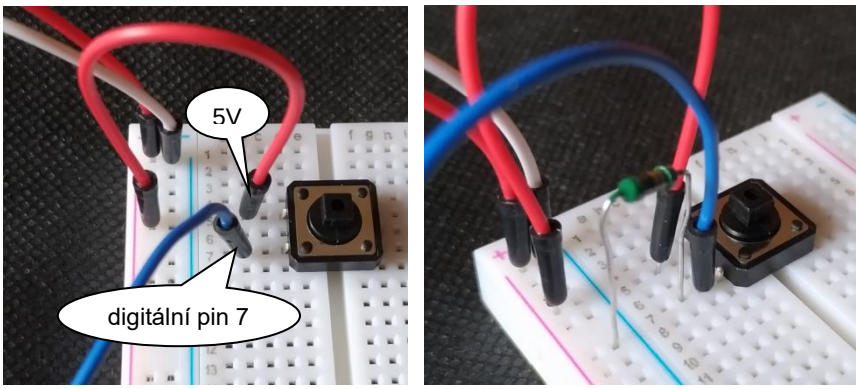
Pojďme zapojit tlačítko – nemusíme pak připojovat a odpojovat dráty.

Spínačem vedou skrz dva kousky kovu. Stiskem tlačítka vodivě spojíme jeden s druhým.

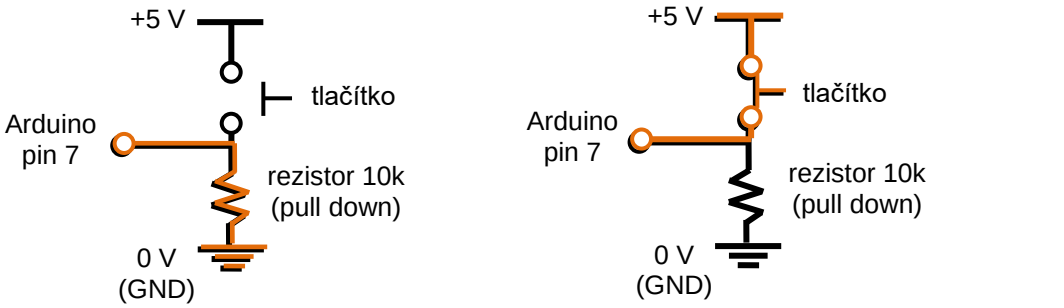


Zapojíme tlačítko, a když ho sepneme, Sériový monitor píše *1*, když není sepnuté, měl by psát *0*. -> Měl by, ale nepíše!

Dá se to nazvat *problém pinu ve vzduchu* – přidáním rezistoru (10 kΩ) se problém vyřeší.



Tomuto zapojení říkáme *pull down* (tažení k hodnotě nula (GND)). Pin nikdy není ve vzduchu, je na něm buď 5V (hodnota *1*) nebo GND (*0*).



NESEPNUTO
Na pinu 7 čteme LOW (0).
Cesta k 5 V není propojena.

SEPNUTO
Proud teče cestou s nejmenším odporem.
Na pinu 7 tedy čteme HIGH (1)

LED a tlačítko

Nyní konečně můžeme rozsvítit LED stiskem tlačítka:

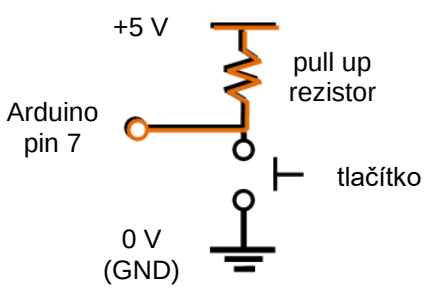
```
void setup() {
  pinMode(3, OUTPUT);    // LED na pinu číslo 3
  pinMode(7, INPUT);     // tlačítko na pinu 7
}

void loop() {
  if (digitalRead(7) == 1) {          // je tlačítko sepnuté?
       digitalWrite(3, HIGH);        // Ano -> rozsviť LED
  } else {                            // jinak, když není sepnuté…
       digitalWrite(3, LOW);          // zhasni LED
  }
}
```

Užitečné vědět: Není jedno, zda použijeme = nebo ==. Jedno = přiřazuje hodnotu do proměnné. Dvě == zjišťují v otázce, zda je splněna.

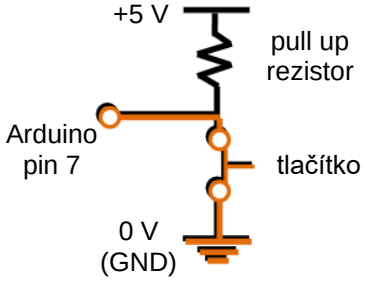
Opačné zapojení tlačítka

Zkusme nyní tlačítko zapojit obráceně – přesuňme nožičku rezistoru na + a vodič od tlačítka na −.

Tomuto zapojení říkáme *pull up* (tažení rezistoru k 5V). Po sepnutí tlačítka je obvod připojen na zem (GND).



NESEPNUTO
Na pin 7 je přivedeno 5 V.



SEPNUTO
Na pinu 7 je 0 V.

Tlačítko se chová nelogicky opačně – když není sepnuté, je na pinu 7 hodnota *1*, když je sepnuté, je tam *0*.

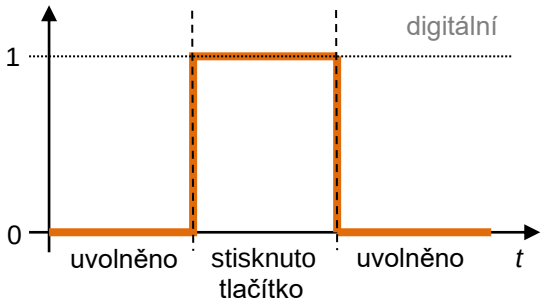
Dobrá zpráva je, že procesor Atmega328 v Arduinu je takto už zapojen – to znamená, že má rezistory integrovány na všechny digitální piny. My tak nemusíme zapojit k tlačítku rezistor. To se ale bude chovat opačně. Stisk dolů (*down*) lze vnímat jako hodnotu DOWN (*0*) a nestisknuté jako HIGH (*1*), jako u LED.

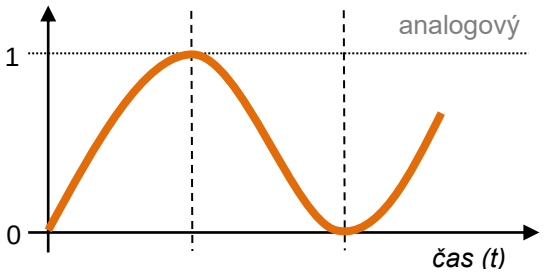
Semafor s tlačítkem pro chodce

Zkus přidat ke klasickému semaforu i semafor pro chodce ovládaný tlačítkem. Tedy chodec stiskne, tak na semaforu aut naskočí červená, a pak na chvíli svítí zelená pro chodce. Po chvíli zase autům naskočí zelená.

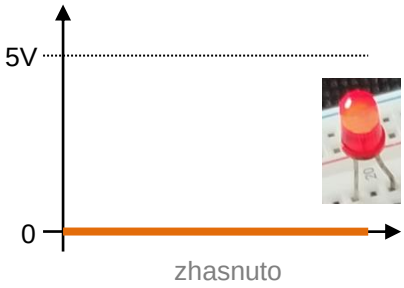
PWM modulace / AnalogWrite

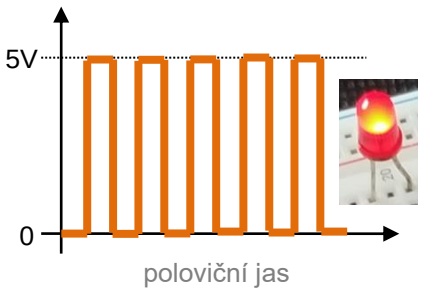
Doposud jsme LEDku jen rozsvítili nebo zhasli. Na to nám stačily digitální piny, které vysílaly hodnotu HIGH nebo LOW. Někdy ale potřebujeme proměnlivý jas, jako je vidět na druhém grafu.

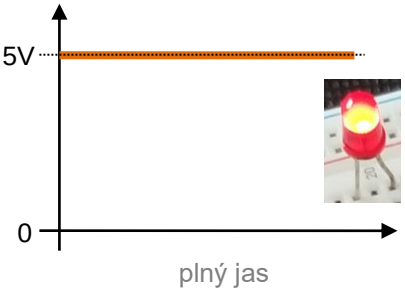




Arduino ale neumí vysílat analogový signál. Řeší to ale podobně, jako když převádíme analogový signál na digitální. Použije tzv. PWM modulaci (převod pomocí šířky pulzu). Když budeme diodou dostatečně rychle blikat, bude to vypadat, že svítí trvale, ale menším jasem.







Změna jasu LED

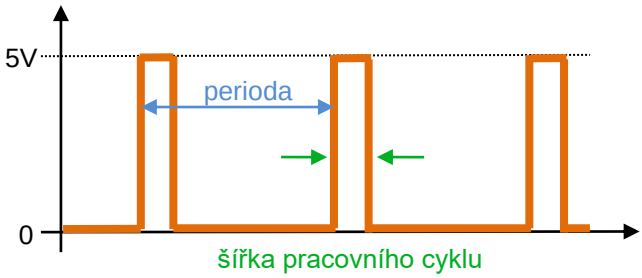
Zapoj dvě LED – jednu do pinu 3, druhou do pinu 4. LED v pinu 4 prostě rozsvítíme naplno, zatímco LED v pinu 3 necháme velmi rychle blikat – budeme ale měnit šířku pulzu a porovnávat její jas s tou plně svítící.

```
void setup() {
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
}

void loop() {
  digitalWrite(4, HIGH);

  digitalWrite(3, HIGH);
  delayMicroseconds(200);
  digitalWrite(3, LOW);
  delayMicroseconds(800);
}
```

Pozor – používáme nyní delayMicroseconds

1 milisekunda = 1000 mikrosekund (1000 µs)
1 sekunda = 1000 milisekund (1000 ms)

LED je 20 % času zapnutá a 80 % vypnutá. To je tzv. pracovní cyklus 20 % (duty cycle 20).



Proto se to nazývá

pulzně šířková modulace

pulse wide modulation (PWM)

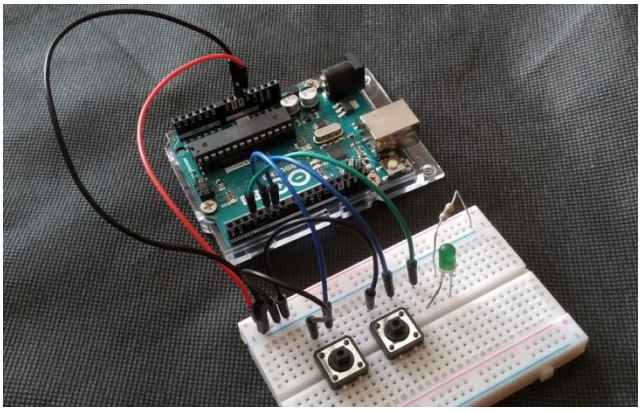
Tak složitý kód, jako je výše, se samozřejmě nemusí využívat. Nahradíme ho příkazem analogWrite().

Pozor, tuto funkci mají jen některé piny – jsou to ty, u kterých je namalovaná vlnovka => 3, 5, 6, 9, 10 a 11.

A ještě jedna věc – piny mají pevně nastavenou periodu 2040 µs, ale piny 5 a 6 jen 1020 µs.

Změna jasu LED pomocí dvou tlačítek

Zapojme nyní dvě tlačítka a LED. Stiskem prvního tlačítka se bude jas plynule zvyšovat, druhým snižovat.

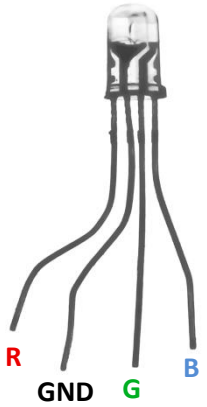


```
int ledPin = 3;
int jas = 0;
int tlac1 = 5;
int tlac2 = 6;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(tlac1, INPUT_PULLUP);
  pinMode(tlac2, INPUT_PULLUP);
}

void loop() {
  analogWrite(ledPin, jas);                          // procesor použije hodnotu ze setup (nyní 0)
  if (digitalRead(tlac1) == LOW && jas < 255) {      // tlač1 stisknuto a zároveň jas menší než max
       jas += 1;                                     // zkrácený zápis verze jas = jas + 1
  }
  if (digitalRead(tlac2) == LOW && jas > 0) {        // když dioda nesvítí, tlač2 její jas zvyšuje
       jas -= 1;
  }

  delay(10);                                         // chvilku počkej, ať se jas nemění tak rychle
}
```

RGB LED

Teď použijeme PWM modulaci pro nastavení barvy na RGB LED. Ta má totiž pro každou barvu svou nožičku, a tak nastavením jasu svitu dané barvy nastavíme nejen základní barvy, ale můžeme je i míchat.

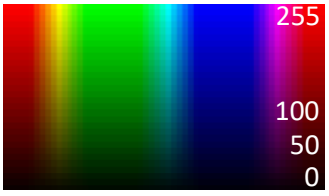
RGB LED má buď společnou anodu, nebo katodu. Zde verze se společnou katodou. R, G, B piny jsou tedy +.



V principu je u nejdelší nožičky R pin. Další pak v pořadí RGB. Pokud si to ale chceš ověřit, stačí připojit nejdelší nožičku na GND a rozsvítit připojením jednotlivých nožiček k napájení (samozřejmě přes rezistor).

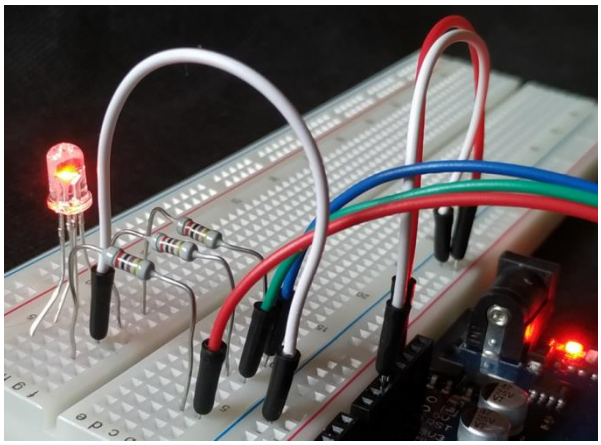
R	G	B	barva
0	0	0	černá
255	0	0	červená
0	255	0	zelená
0	0	255	modrá
255	255	0	žlutá
255	0	255	purpurová
0	255	255	azurová
255	255	255	bílá

Vlevo je tabulka základních barev, ale smíchat lze libovolnou a i různého jasu. Např. hodnotou 50, 0, 50 vznikne slabá purpurová.

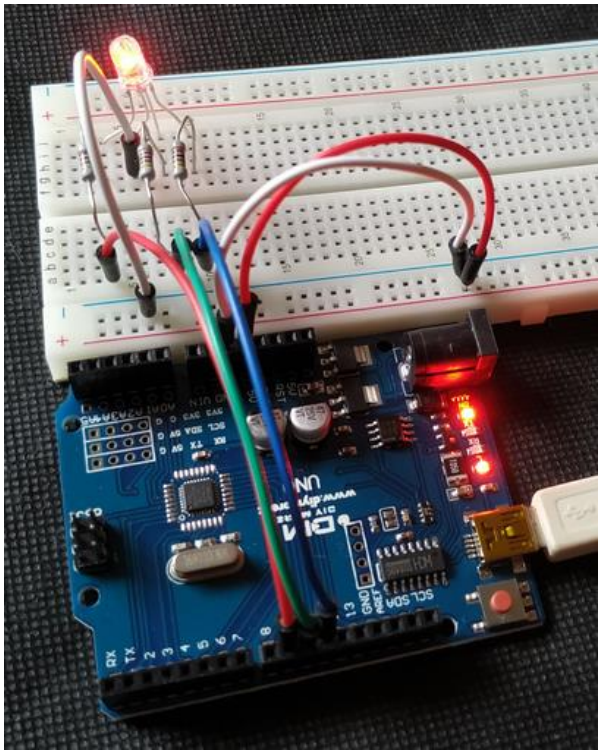


Příklad programu, kde RGB LED se rozsvítí červeně a tlačítko změní barvu na purpurovou.

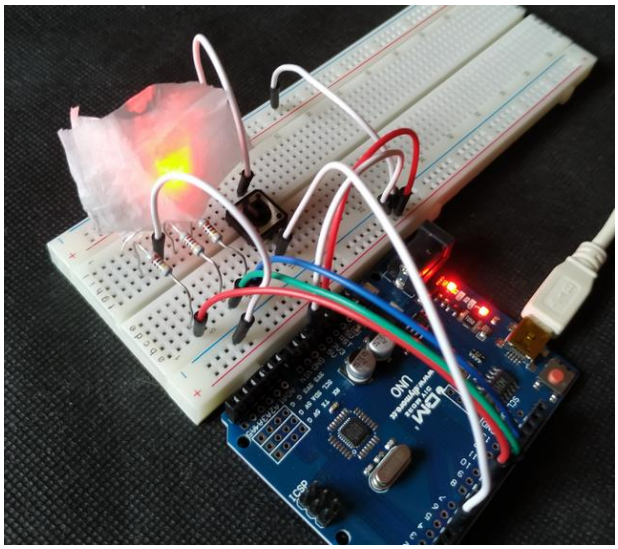
```
int RPin = 9;
int GPin = 10;
int BPin = 11;

int RValue = 0;
int GValue = 0;
int BValue = 0;

int tlac1 = 5;

void setup() {
  pinMode(RPin, OUTPUT);
  pinMode(GPin, OUTPUT);
  pinMode(BPin, OUTPUT);
  pinMode(tlac1, INPUT_PULLUP);
}

void loop() {

  analogWrite(RPin, RValue);
  analogWrite(GPin, GValue);
  analogWrite(BPin, BValue);

  if (digitalRead(tlac1) == LOW) {
    RValue = 50;
    GValue = 00;
    BValue = 50;
  }else{
    RValue = 50;
    GValue = 0;
    BValue = 0;
  }

  delay(10);

}
```

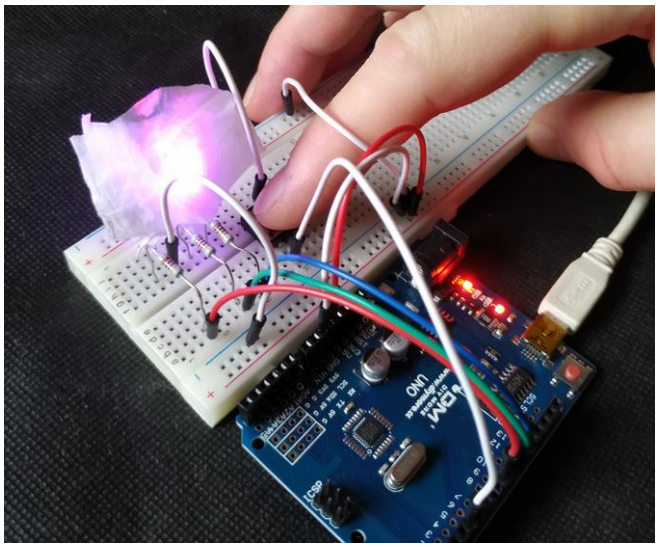


Zapojení RGB LED na kontaktním poli.



Pokud tě ruší, že jsou vidět obě barvy (zde červená a modrá odděleně), zkus přes LED přehodit kus papíru.





Řízení jasu LED pomocí potenciometru

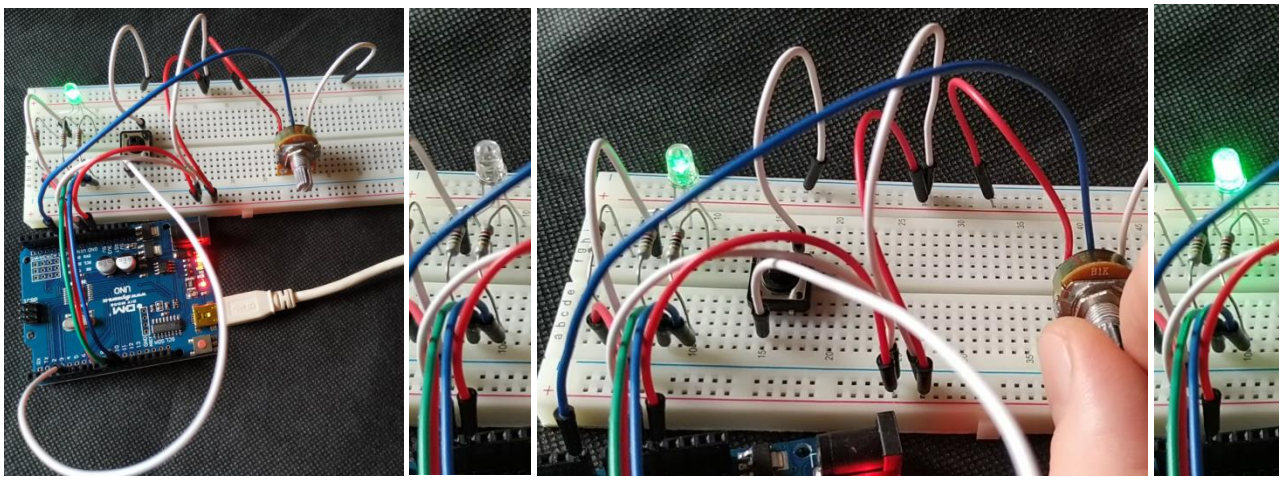
Potenciometr je součástka podobná rezistoru, ale při vhodném zapojení můžeme pomocí jezdce měnit její odpor, a tím i proud protékající obvodem. Máme potenciometry, které slouží pro řízení obvodu člověkem a mají hřídelku pro nasazení hmatníku. A pak máme trimry, které slouží pro jemné donastavení nějaké hodnoty, a pak už se na ně nesahá a ty se ovládají šroubovákem. Trimry jsou menší, jednodušší a mají výrazně nižší mechanickou životnost, protože se neočekává že hodnota se na nich bude často měnit.

Zkusme nyní k předchozímu obvodu přidat potenciometr a přimíchat pomocí něj do jedné barvy druhou. Vidíme, že jezdec má prostřední pin a zapojíme jej do Arduina do pinu A0, krajní připojíme na + a –.

```
int RPin = 9;
int GPin = 10;
int BPin = 11;
int tlac1 = 5;
int potPin = A0;                        // pro vstup z jezdce použijeme analogový pin Arduina!
int RValue = 0;
int GValue = 0;
int BValue = 0;
int potValue = 0;

void setup() {
  pinMode(RPin, OUTPUT);
  pinMode(GPin, OUTPUT);
  pinMode(BPin, OUTPUT);
  pinMode(tlac1, INPUT_PULLUP);
  pinMode(potPin, INPUT);

  Serial.begin(9600);          // spustíme i Sériový monitor pro zobrazení hodnot na potenciometru
}

void loop() {

  analogWrite(RPin, RValue);
  analogWrite(GPin, GValue);
  analogWrite(BPin, BValue);

  if (digitalRead(tlac1) == LOW && RValue <255) {    // zajistíme plynulý nárůst R stiskem tlačítka
    GValue =0;                                       // zelená bude 0, modrá podle potenciometru
    RValue +=1;                                      // tedy výsledná barva bude červená
  }

  potValue = analogRead(potPin);             // načteme hodnotu z potenciometru
  GValue = map(potValue, 0, 1023, 0, 255);   // nastavíme hodnotu na Gpinu RGB LED, ale trikem, kdy
  analogWrite(GPin, GValue);                 // namapujeme hodnoty z rozsahu 0-1023 na rozsah 0-255*

  Serial.println(potValue);1

  delay(10);
}
```

*analogový pin čte hodnoty v rozsahu 0-1023, ale jak už víme, zápis pomocí digitální PWM modulace je v rozsahu 0-255, proto musíme hodnoty z analogového pinu přepočítat na menší rozsah – to umí *map*.

Všimni si, že k analogovému pinu musí být v procesoru Atmega328 připojen A/D převodník (analog/digital). Ten z námi nastavené hodnoty rozsahu potenciometru vytvoří omezený rozsah hodnot – zde 0 až 1023.



Plynulá změna jasu pomocí PWM

Nyní zkusíme RGB LED nastavit, aby svítila nejprve červeně, pak se její barva plynule změní v zelenou, potom v modrou, a tak stále dokola.

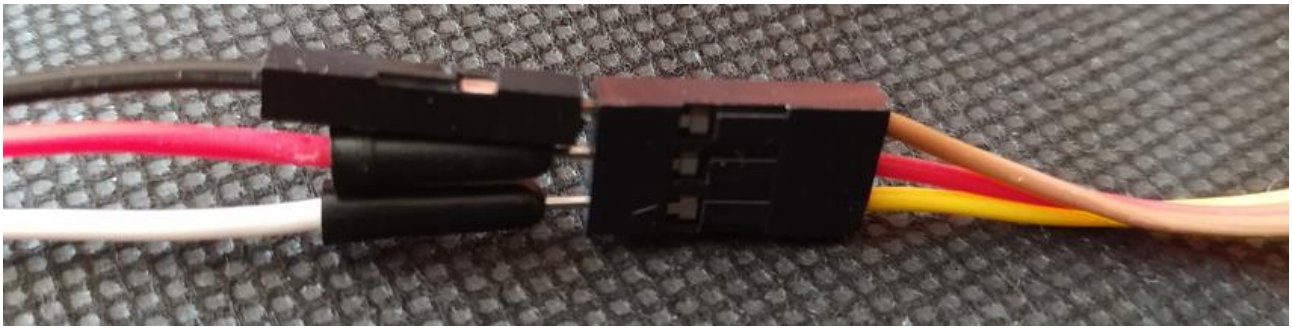
```
#define rPin 11        // naučíme se nový způsob nastavení názvu konstant
#define gPin 10        // #define totiž nevyužívá paměť Arduina
#define bPin 9         // a načte hodnoty až při kompilaci programu do Arduina

void setup() {
  pinMode(rPin, OUTPUT);
  pinMode(gPin, OUTPUT);
  pinMode(bPin, OUTPUT);
}

void loop() {
                                          // červená se prolne do zelené
  for (int i=0; i<256;i++) {              // int i=0: zde zavádíme proměnnou i a nastavujeme jí 0
    analogWrite(rPin, 255-i);             // for i<256;i++ znamená: pro i<256 zvětši i o 1
    analogWrite(gPin, i);                 // s každým krokem cyklu for se hodnota na R zmenší o 1
    analogWrite(bPin, 0);                 // B zůstává vypnutá
    delay(50);
  }

  for (int i=0; i<256; i++) {             // zelená se prolne do modré
    analogWrite(rPin, 0);
    analogWrite(gPin, 255-i);
    analogWrite(bPin, i);
    delay(50);
  }

  for (int i=0; i<256; i++) {             // a modrá zpět do červené
    analogWrite(rPin, i);
    analogWrite(gPin, 0);
    analogWrite(bPin, 255-i);
    delay(50);
  }
}
```

Servomotor

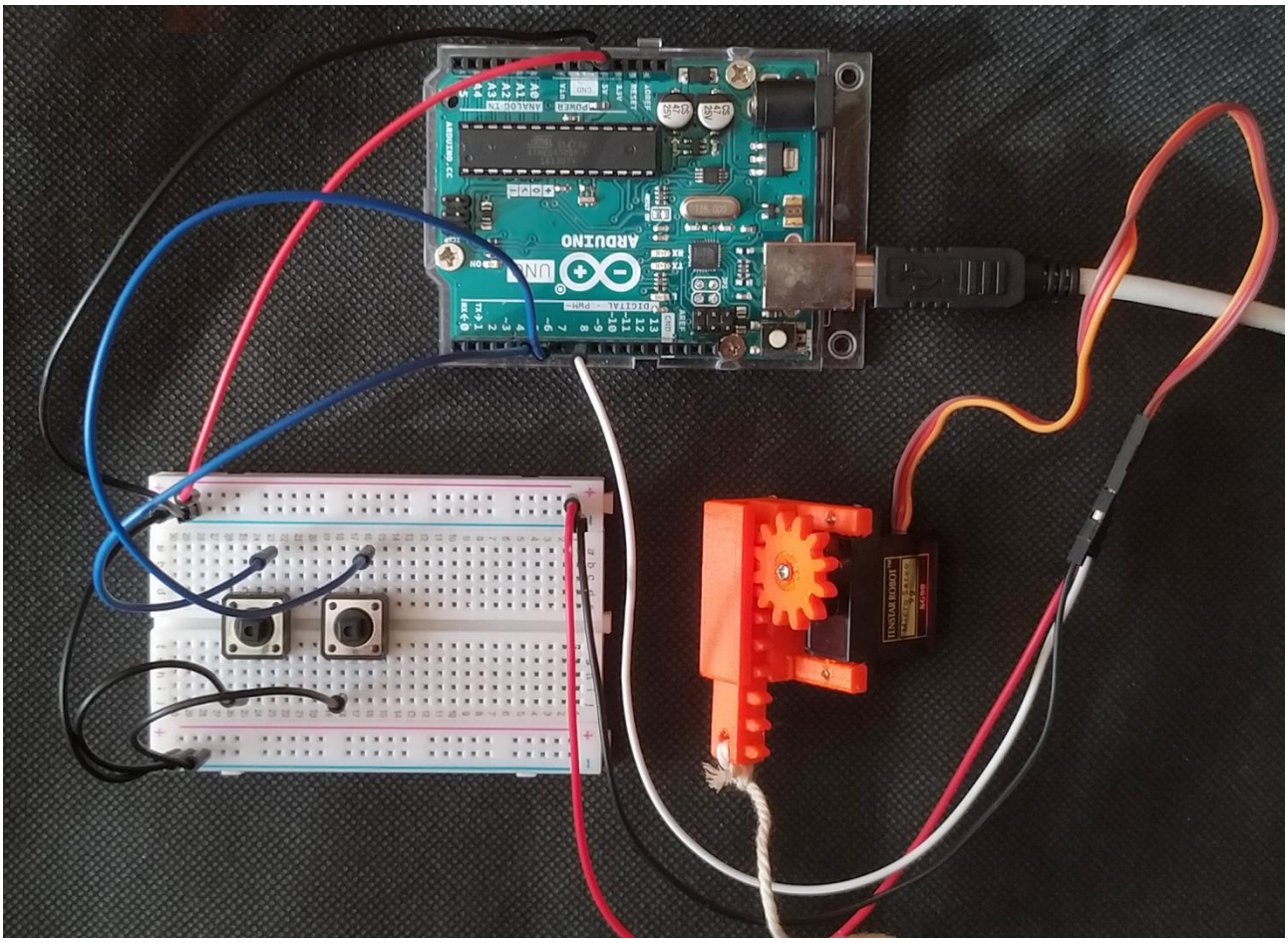
Servo motory slouží pro nastavení určité polohy ovládaného mechanizmu a následné držení v této poloze. Stejnosměrné servo motory se využívají například pro ovládání robotické paže nebo pro nastavení kormidla u leteckých modelů. Jejich hlavní výhodou je malý rozměr a malá hmotnost s relativně velkou silou.

Tyto motory obvykle neumožňují otáčení neustále dokola, ale udržují nastavený úhel natočení. Úhel se pohybuje nejčastěji v rozsahu 0° až 180°. Nastavení tohoto úhlu se provádí zasláním impulsu o určité délce. Neutrální poloha (90°) odpovídá obvykle délce impulsu 1,5ms. Délka 0,5ms odpovídá úhlu 0° a impuls délky 2,5ms nastavuje úhel 180°. Impulsy se posílají motoru pravidelně každých 20ms.

Jak vidíme, servomotor má tři konektory, červený je obvykle +, hnědý nebo černý – a žlutý nebo oranžový posílá příkazy. (*Podobně to bývá i u motorků větráků v PC, kde žlutý řídí rychlost otáček*).



Pojďme si tedy rozhýbat mikro servo SG90 pomocí dvou tlačítek. Na fotce je servo zabudované do vytištěných dílů, které simulují hřebenový převod.



Jak je vidět, program se v zásadě nemusí lišit od toho, kdy jsme tlačítky plynule rozsvěceli a zhasínali LED.

```
#include <Servo.h>            // zahrnutí knihovny pro ovládání servo motoru

Servo servo;                  // každé servo má svou instanci třídy Servo
int uhel = 0;                 // výchozí pozice
int tlac1 = 5;
int tlac2 = 6;

void setup() {
  servo.attach(8);
  servo.write(uhel);
  pinMode (tlac1, INPUT_PULLUP);
  pinMode (tlac2, INPUT_PULLUP);
}

void loop() {
  if (digitalRead (tlac1) == LOW && uhel < 255){
      uhel += 1;
  }
  if (digitalRead (tlac2) == LOW && uhel > 0){
      uhel -= 1;
  }
servo.write(uhel);
delay(15);
}
```

Alternativně bez tlačítek by program vypadal takto

```
#include <Servo.h>

Servo servo;
int uhel = 0;

void setup() {
  servo.attach(8);
}

void loop()
{
  for(uhel = 0; uhel <= 180; uhel += 1)      //je od úhlu 0 do úhlu 180
  {
    servo.write(uhel);                       //natočení motoru na aktuální úhel
    delay(15);                               //chvilka čekání než se motor natočí
  }
  for(uhel = 180; uhel >= 0; uhel -= 1)      //je od úhlu 180 zpět do úhlu 0
  {
    servo.write(uhel);                       //natočení motoru na aktuální úhel
    delay(15);                               //chvilka čekání než se motor natočí
  }
}
```

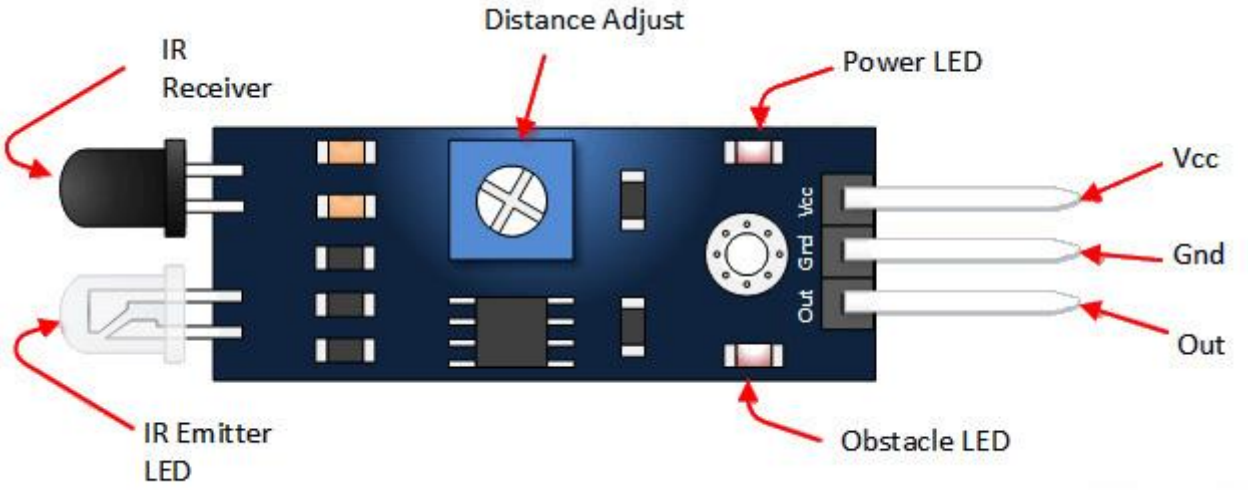
Servo motor má uvnitř potenciometr, který měří jeho natočení – to je také důvodem, proč se neotočí víc než o 180°. Natočení serva můžeme tedy ovlivnit ručně také pomocí potenciometru, podobně jsme ovládali jas LED.

```
#include <Servo.h>     //zahrnutí knihovny pro ovládání servo motoru
Servo servo;           //každý motor má svou proměnnou typu Servo
int potPin = 0;        //pin ke kterému je připojen potenciometr
int poloha;            //proměnná pro načtení a nastavení úhlu

void setup()
{
  servo.attach(9);     //tento motor je připojen na pin 9
}

void loop()
{
  poloha = analogRead(potPin);               //napětí na potenciometru (0 až 1023)
  poloha = map(poloha, 0, 1023, 0, 180);     //převod z 0 až 1023 na 0 až 180
  servo.write(poloha);                       //nastavení polohy podle potenciometru
  delay(15);                                 //chvilka čekání než se motor natočí
}
```

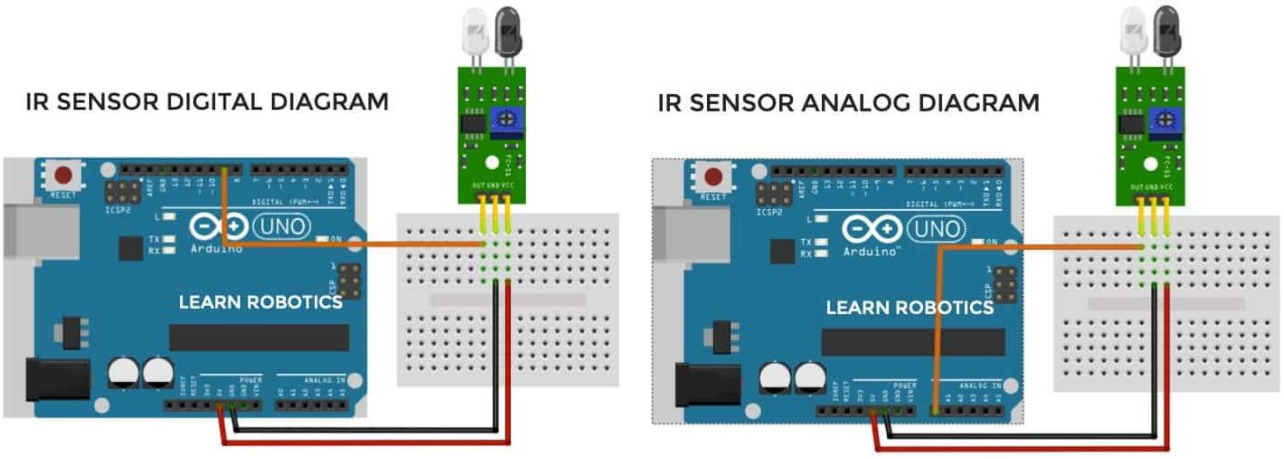
Infračervený senzor

IR senzor používá jako vysílač a přijímač speciální LED. Neměří, jak daleko objekt je, ale dá se potenciometrem nastavovat citlivost, kam až „vidí“. Jeho použití je jednodušší a v případech, kde nám jde jen o to, zda překážka přítomna je, či nikoliv, to stačí.

Typicky se senzor používá u detekce překážek roboty, jízdu po čáře, detekci plamene nebo pohybu (PIR). Pro detekci plamene a detekci pohybu se kupuje specifický senzor plamene a PIR senzor.



IR senzor má tři piny: GND, Vcc a Signal (datový) – ten lze připojit k digitálnímu (hodnota 0 nebo 1) i analogovému pinu (hodnoty 0 až 1023).

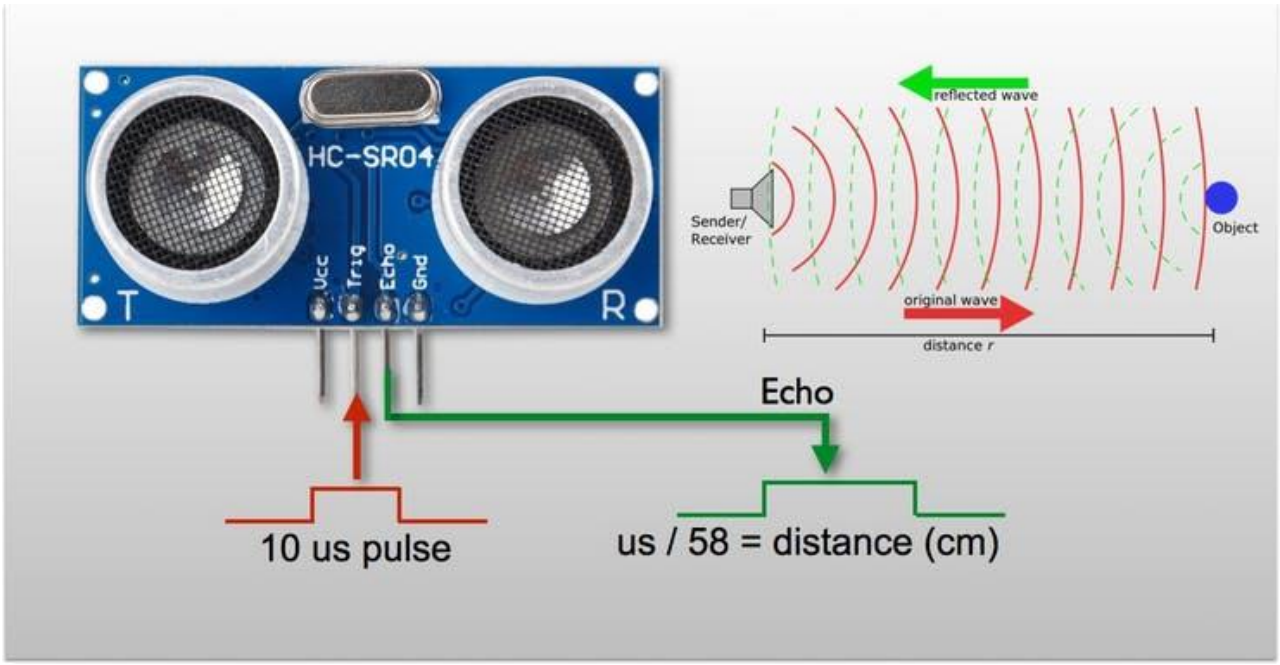


Program pro senzor může vypadat takto:

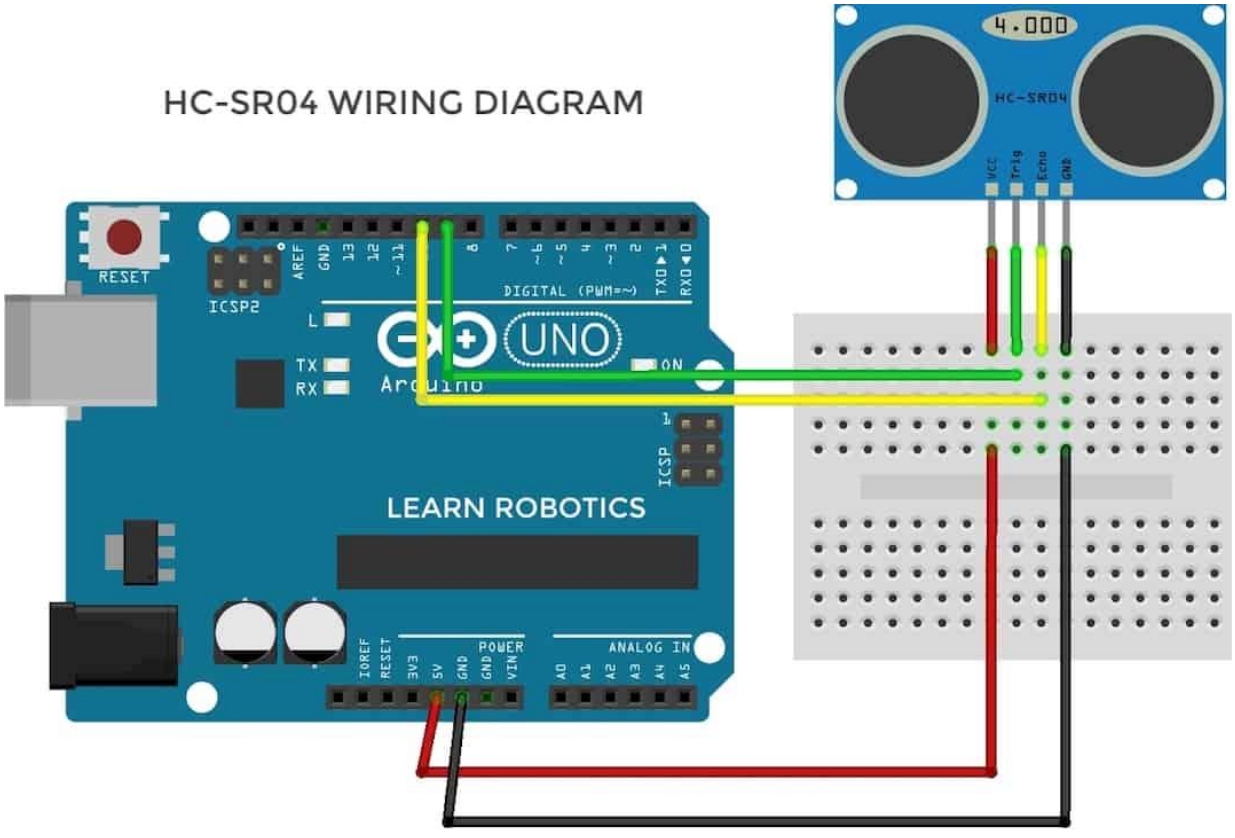
```
int IR = 9;

void setup() {
  pinMode(IR, INPUT);
  Serial.begin(9600); //inicializace sériového monitoru
}

void loop() {
  digitalRead(IR);
  Serial.print("IR hodnota = ");
  Serial.println(IR);
  delay (2000);
}
```

Ultrazvukový senzor

Senzor vysílá zvukové vlny a po odrazu od překážky je opět přijímá. Měříme tedy čas a ze známé rychlosti zvuku 340 m/s ve vzduchu počítáme vzdálenost překážky. Pro běžné projekty můžeme použít například senzor HC-SR04.



Senzory mohou být i třípinové, ale HC-SR04 má čtyři piny. Vždy obsahují Gnd a Vcc pin. Třípinový senzor, např. PING, má signálový pin funkční jako vstupní i výstupní (INPUT, OUTPUT). HC-SR04 má zvlášť výstupní (vysílač, Trig) a vstupní (přijímač, Echo). Připojujeme je k digitálním pinům Arduina.



```
/*
 KÓD PRO ULTRAZVUKOVÝ SENZOR HC-SR04
 */
#define trigPin 13      // Trigger Pin
#define echoPin 12      // Echo Pin
#define ledMin 11     // Červena LED na pinu 11
#define ledMax 10     // Zelena LED na pinu 10
int maximumRange = 200;       // maximální vzdálenost
int minimumRange = 0;       // minimální vzdálenost
long duration, distance;    // Vypočítání vzdálenosti
void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledMin, OUTPUT);
  pinMode(ledMax, OUTPUT);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration/2) / 29.1;   //vypočítání v cm

  // To je místo, kde LED rozsvítíme a zhasneme.
if (distance < 4) {                 // rozsvítíme červenou LED, vzdálenost do 4 cm
  digitalWrite(ledMin,HIGH);        // Když rozsvítíme červenou, zelenou musíme zhasnout
  digitalWrite(ledMax,LOW);
} else if (distance > 200) {        // rozsvítíme zelenou LED, vzdálenost > 200 cm
       digitalWrite(ledMax,HIGH);   // Když rozsvítíme zelenou, červenou musíme zhasnout
       digitalWrite(ledMin,LOW);
}
  else {          // normální provoz, vzdálenost od 4 do 200 cm, nesviti nic
       digitalWrite(ledMin,LOW);
       digitalWrite(ledMax,LOW);
  }
  if (distance >= maximumRange  || distance <= minimumRange){
  Serial.println("Mimo dosah");
  }
  else {
    Serial.print(distance);
    Serial.println(" cm");
  }
  //počkání 500ms
  delay(500);
}
```

Ultrazvukový senzor je často používaný v robotice k vyhnutí se překážkám. Jiným zajímavým využitím je čtení výšky hladiny v nádobě (např. množství srážek) nebo výšky sněhu.