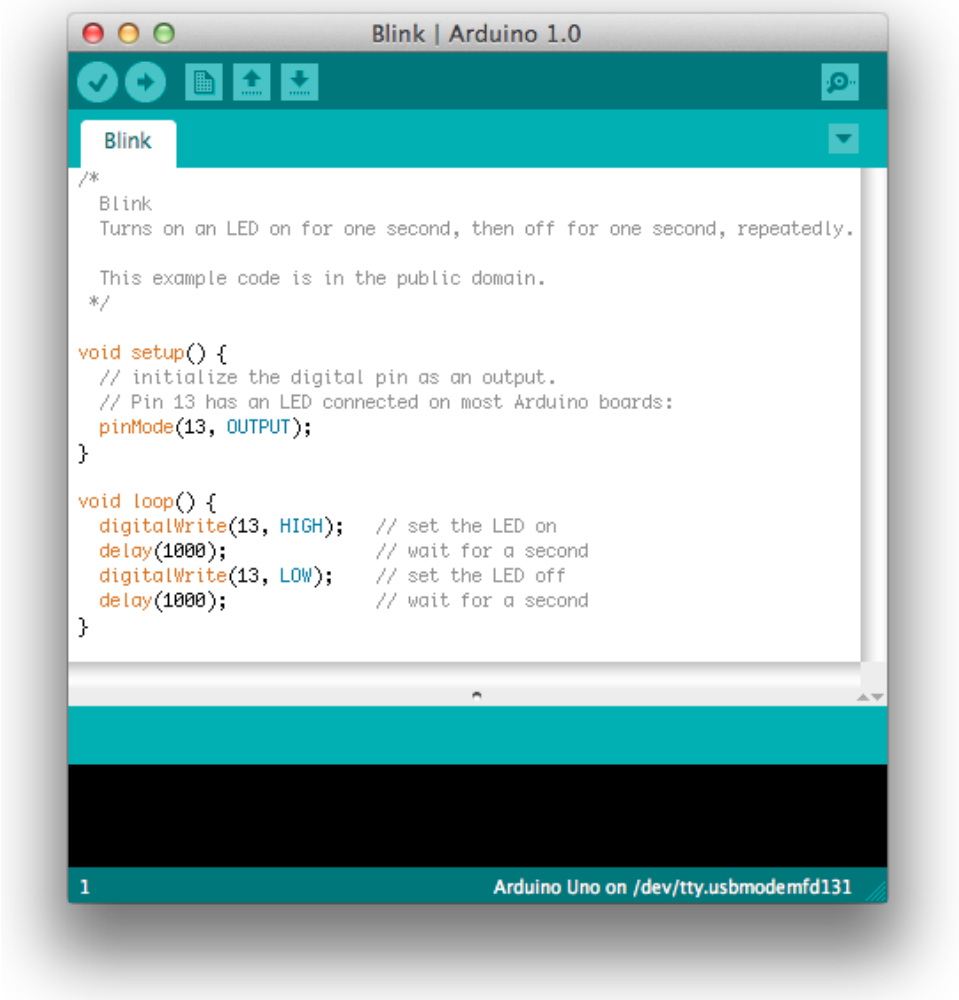
LASKKIT

Arduino MAXI Starter kit
OD BASTLÍŘŮ BASTLÍŘŮM

Ver. 1.4

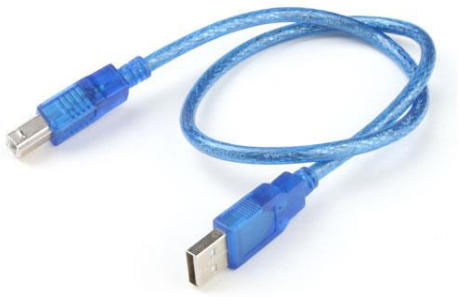
ARDUINO – ZAČÍNÁME!



Tento článek popisuje, jak připojit Tvoje Arduino k počítači a nahrát první projekt (sketch).

1. Připrav si Arduino a USB kabel

V tomto tutoriálu předpokládáme, že používáš Arduino Uno, Mega 2560. Dále budeš potřebovat standardní USB kabel (A plug to B plug).



2. Stáhni si prostředí Arduino

Stáhni si poslední verzi programu z https://arduino.cc/en/Main/Software.
Jakmile se stahování dokončí, rozbal stažený soubor. Dej si pozor, abys zachoval strukturu souborů ve složce. Dvojklikem soubor otevři. Ve složce by mělo být několik souborů a podsložek.

3. Zapoj desku

Arduino Uno či Mega mohou být napájeny z USB nebo z externího zdroje. Připoj desku Arduino ke svému počítači použitím USB kabelu. LED na zdroji (označená PWR) by se měla rozsvítit.

4. Nainstaluj ovladače

Instalace ovladačů pro Arduino Uno nebo Arduino Mega 2560 s převodníkem ATmega16U2 (Precizní klon) na Windows 7, Vista a XP (Pro macOS není potřeba).

- Zapoj svoji desku a počkej, až Windows začne instalovat ovladače.
- Klikni na nabídku „Start“ a otevři ovládací panely.
- V nabídce ovládacích panelů klikni na položku „Systém“. Jakmile se Ti otevře nabídka systému, klikni na „Správce zařízení“ (může být potřeba oprávnění správce).
- Rozbal Porty (COM a LPT). Měl bys vidět otevřený port pojmenovaný „Arduino UNO (COMxx)“. Pokud nenajdeš sekci COM a LPT, vyhledej v Ostatních zařízeních „Neznámé zařízení“.
- Pravým tlačítkem klikni na „Arduino UNO (COMx)“ (popřípadě na neznámé zařízení) a vyber možnost „Aktualizovat ovladač“.
- Dále vyber možnost „Prohledat počítač a najít ovladače“.
- Nakonec, najdi a vyber soubor ovladače pojmenovaný „arduino.inf“, který se nachází ve složce „Drivers“ staženého softwaru Arduino.
- Windows dokončí instalaci ovladače.

Instalace ovladačů pro všechny desky Arduino, WeMos atd. s převodníkem CH340 na Windows 7.
Jakmile zapojíš svoji desku do počítače, Windows by měl začít instalovat ovladače (pokud jsi již předtím na počítači nepoužíval desky s CH340).

- Jakmile se zobrazí okno dialogu „Automaticky se spojit s Windows Update a vyhledat software?“, vyber „Tentokrát ne“. Klikni na další.
- Vyber „Nainstalovat ze seznamu“, nebo z umístění (Pokročilé). Klikni na další.
- Ujisti se, že je zaškrtnuta volba „Hledat nejlepší ovladač“ v těchto umístěních; označ volbu „Prohledat výsuvná zařízení“; zaškrtni volbu „Zahrň toto umístění do hledání“ a najdi složku s ovladačem stažených z webových stránek WeMos.
- Panel začne hledat ovladače a pak zobrazí zprávu „USB–SERIAL CH340(COMxx)“ (nebo podobný) byl nalezen. Klikni na „dokončit“.
- Znovu se zobrazí panel „Přidat nové zařízení“. Projdi znovu těmi stejnými kroky a vyber ty samé volby pro umístění a hledání. Tentokrát bude nalezen „USB–SERIAL CH340(COMxx)“ (nebo podobný).

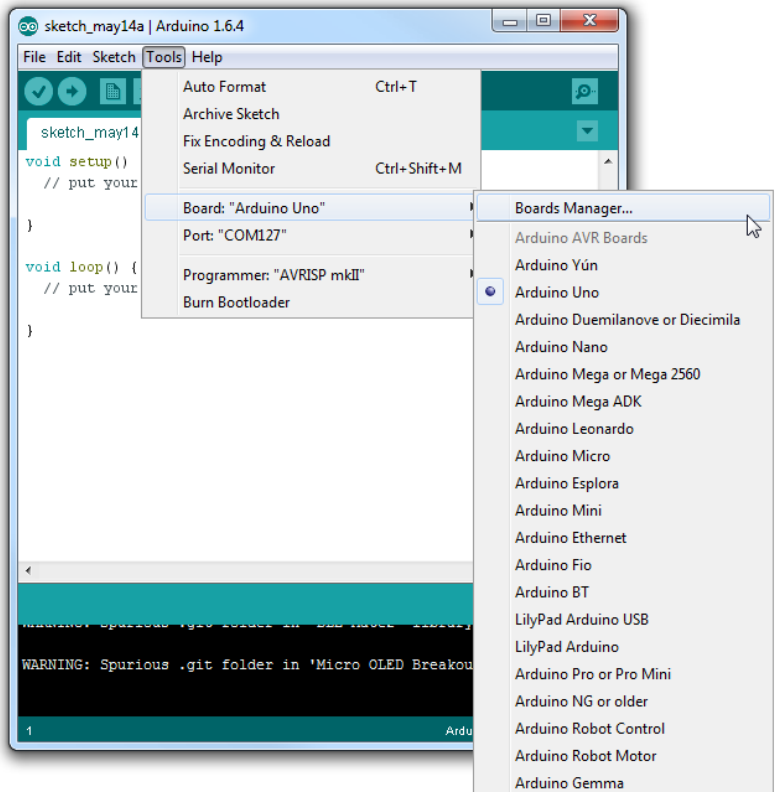
O tom, zda byly ovladače správně nainstalovány, se můžeš ujistit otevřením „Správce zařízení“ (v záložce Hardware panelu Systém). V sekci „Porty“ najdi „USB–SERIAL CH340(COMxx)“ (nebo podobný). To je deska Arduino.
Pro macOS je potřeba stáhnout a naistalovat ovladače z webových stránek WeMos!

5. Spusť aplikaci Arduino

Dvojklikem otevři aplikaci Arduino.
Otevři LED blink example sketch (ukázkový příklad): File > Examples > 1.Basics > Blink.

7. Vyber svoji desku

Vyber v menu Tools > Board (nástroje > deska) druh svého Arduino zařízení.
Vyber v menu Tools > Port (nástroje > port) port svého Arduino zařízení.



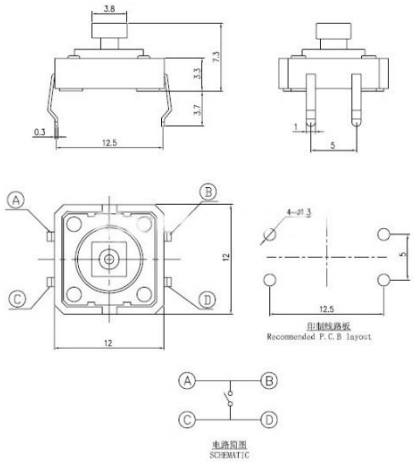
9. Nahraj program

Teď jednoduše klikni na tlačítko „Upload“ v prostředí programu. Počkej několik sekund – měl bys vidět blikat RX a TX LED diody na desce. Pokud je nahrávání dokončeno, zobrazí se na status baru zpráva „Done uploading“- nahrávání dokončeno. Několik sekund potom, co se dokončí nahrávání, bys měl vidět LED piny 13(L) na desce blikat. Pokud blikají, gratulujeme! Tvoje Arduino funguje správně.

Lekce 1 Řízení LED tlačítkem

Úvod

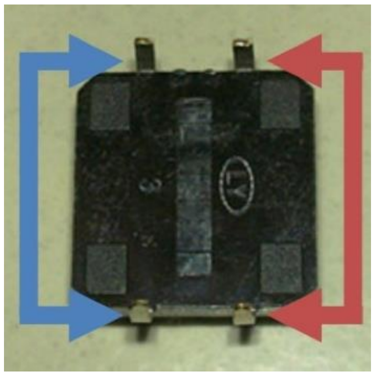
V tomto experiment se dozvíš, jak zapnout / vypnout LED pomocí I/O portu a tlačítka. "I/O port" odkazuje na výstupní a vstupní port. Zde je použit vstupní port Arduino Uno k tomu, aby přečetl výstup z externího zařízení. Vzhledem k tomu, že samostatná deska je vybavena LED (a připojena do pinu 13), můžeš tuto LED použít pro pohodlnější provedení experimentu.



Komponenty

- Arduino
- USB kabel
- Tlačítko
- Odpor (10kΩ)
- Breadboard vodiče

- Breadboard



Princip

Tlačítka jsou běžnou součástí a slouží k ovládání elektronických zařízení. Jsou obvykle používána jako přepínače pro připojení nebo odpojení obvodů. Tlačítka se vyrábí v různých tvarech a velikostech, avšak zde jsme použili 12 mm tlačítko, viz následující obrázky. Piny, na které ukazují šipky stejné barvy, jsou propojeny.

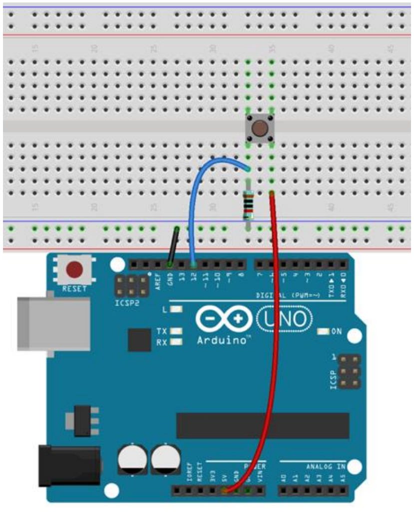
Pří stisknutí tlačítka se piny, na které ukazují modré šipky, připojí na piny označené červenou šipkou.

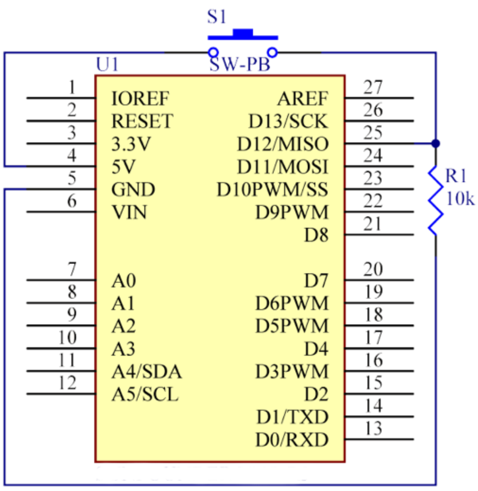
Obecně platí, že tlačítko je přímo napojeno na LED za účelem zapnutí a vypnutí LED. Toto spojení je relativně jednoduché. Někdy se ale LED rozsvítí automaticky, bez stisknutí tlačítka. To může být způsobeno různým typem rušení. Aby se zabránilo těmto vnějším zásahům, je použit pull-down rezistor pro připojení 1K–10KΩ odporu mezi tlačítkem, portem a GND. Toto se využívá k ničení vnějších zásahů, zatímco je připojeno GND tak dlouho, dokud je tlačítko vypnuté.

Toto připojení obvodu je široce používáno v řadě obvodů a elektronických zařízení. Například, pokud stiskneš jakékoli tlačítko na svém telefonu, rozsvítí se Ti podsvícení.

Postup experimentu

Krok 1: Vytvoř obvod

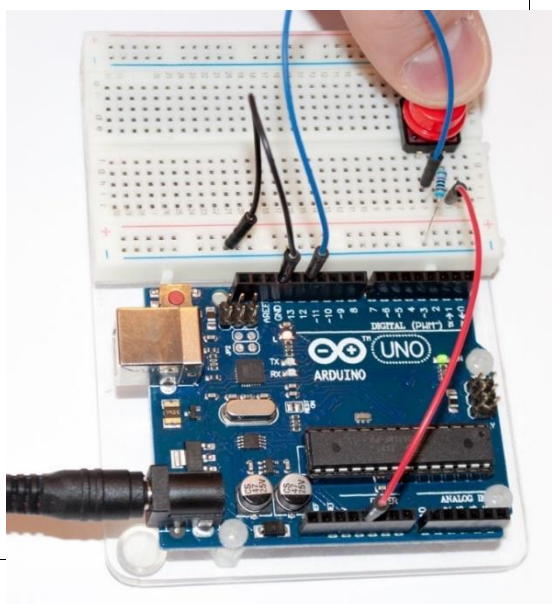




Krok 2: Napiš kód do Arduino IDE

Kód

```
// Řízení LEDky tlačítkem
// Zapíná a vypíná LEDku když stiskneš tlačítko
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

const int keyPin = 12;     // číslo pinu tlačítka
const int ledPin = 13;     // číslo pinu LEDky

void setup() {
         // nastavení pinu tlačítka jako vstupu
         pinMode(keyPin,INPUT);
         // nastavení pinu LEDky jako výstupu
         pinMode(ledPin,OUTPUT);
}

void loop() {
         // přečíst stav hodnoty pinu tlačítka
         // a zkontrolovat, jestli je tlačítko
         // stisknuté
         // pokud ano, stav pinu bude HIGH
         if(digitalRead(keyPin) == HIGH ) {
                  digitalWrite(ledPin,HIGH);   // zapnout LEDku
         } else {
```



```
                digitalWrite(ledPin,LOW);    // vypnout LEDku
        }
}
```

Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Nyní stiskni tlačítko a LED na Arduino se rozsvítí.

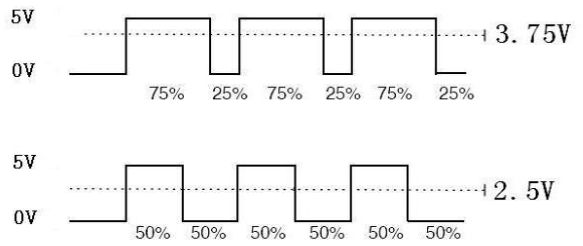
Lekce 2 Řízení LED PWMkou

Úvod

Pojďme zkusit v této lekci něco trochu jednoduššího – postupně budeme měnit jas LEDky pomocí kódů. Vzhledem k tomu, že pulzující světlo vypadá jako dýchání, dali jsme mu magické jméno – dýchající LEDka. Dosáhneme tohoto efektu pomocí pulzně šířkové modulace (PWM)

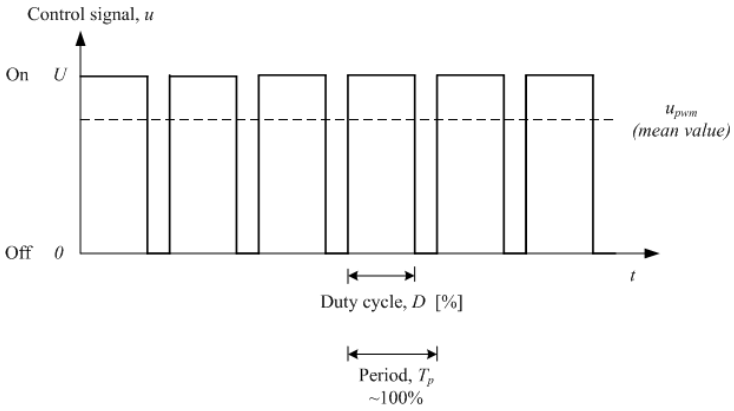
Komponenty

- Arduino
- Breadboard
- Breadboard vodiče
- 1x LED
- Odpor (220Ω)
- USB kabel



Princip

Pulzně šířková modulace neboli PWM (Pulse Width Modulation) je diskrétní modulace pro přenos analogového signálu pomocí dvouhodnotového signálu. Jako dvouhodnotová veličina může být použito např. napětí. Signál je přenášen pomocí střídy. Vzhledem ke svým vlastnostem je pulsně šířková modulace často využívána ve výkonové elektronice pro řízení velikosti napětí.



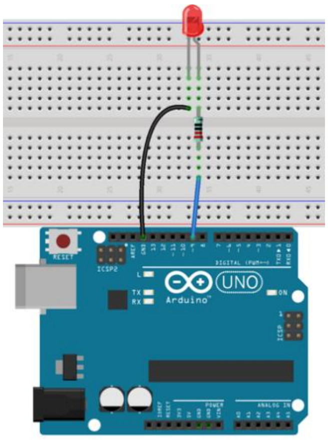
Přenosový signál, který nese informaci o přenášené hodnotě, může nabývat hodnot zapnuto/vypnuto tj. log.1/log.0 (5V a 0V v našem případě). Hodnota přenášeného signálu je v přenosu "zakódována" jako poměr mezi stavy zapnuto/vypnuto. Tomuto poměru se říká střída. Cyklu, kdy dojde k přenosu jedné střídy, se říká perioda. Omezením pro PWM je to, že přenos informace je vždy omezen na relativní vyjádření a to 0 - 100 %. To znamená, že musí být znám poměr mezi skutečnou hodnotou a procentuálním vyjádřením. Časové hodnoty střídy se pohybují v sekundách, v milisekundách pro přesnější řízení. Perioda je vždy součtem doby zapnuto a vypnuto.
Musíš zopakovat periodu dostatečně rychle s LED, aby nebylo vidět, ze LEDka bliká, ale trvale svítí.

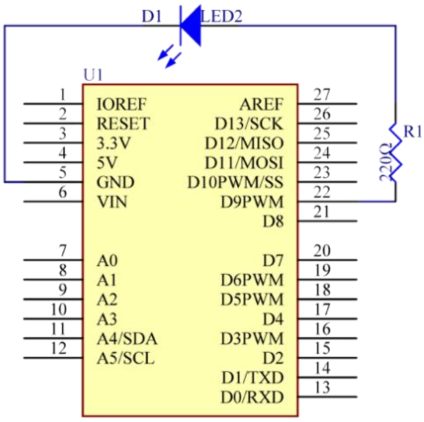
Z oscilogramu je patrné, že amplituda napětí výstupu je 5V. Skutečné výstupní napětí je pouze 3.75V pomocí PWM, protože 5V trvá pouze 75% střídy (duty cycle).
Tři základní parametry PWM:

1. Střída (duty cycle) je poměr mezi stavy zapnuto/vypnuto.
2. Perioda je součtem doby zapnuto a vypnuto.
3. Amplituda napětí je zde 0V-5V.

Postup experimentu

Krok 1: Vytvoř obvod

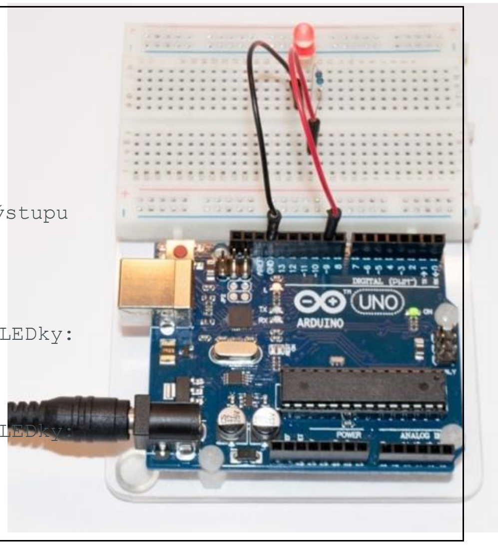




Krok 2: Napiš kód do Arduino IDE

Kód

```
// Řízení LEDky PWMkou
// LEDka se bude pomalu rozsvěcovat a pomalu zhasínat, opakovaně
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/
const int ledPin = 9;                          // číslo pinu LEDky

void setup () {
        pinMode(ledPin, OUTPUT);     // nastavení pinu LEDky jako výstupu
}

void loop() {
        for (int a=0; a<=255;a++) {            // smyčka od 0 do 255
                analogWrite(ledPin, a);        // nastavit jas pinu LEDky:
                delay(8);                      // počkat 8 ms
        }
        for (int a=255; a>=0;a--) {  // smyčka od 255 do 0
                analogWrite(ledPin, a);        // nastavit jas pinu LEDky
                delay(8);                      // počkat 8 ms
        }
        delay(800);                            // počkat 800 ms
}
```



Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Tady bys měl vidět, jak se LEDka stává jasnější a tmavší, a tak pořád dokola.

Lekce 3 LED Tekoucí potok

Úvod

V této lekci uděláme jednoduchý, ale zajímavý experiment – pomocí LED vytvořím tekoucí LED světla. Jak název napovídá, tato plynulá světla jsou tvořena osmi LED v řadě, která se postupně rozsvítí a tmavnou jeden po druhém, stejně jako tekoucí voda.

Komponenty

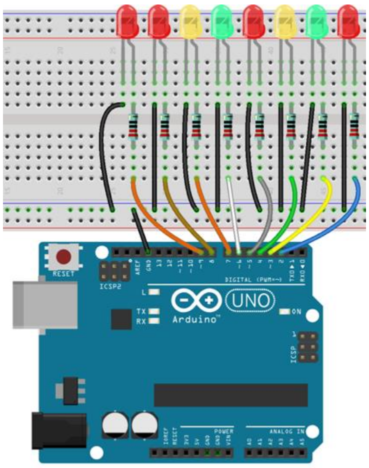
- Arduino
- Breadboard
- Breadboard vodiče
- 8x LED
- 8x Odpor (220Ω)
- USB kabel

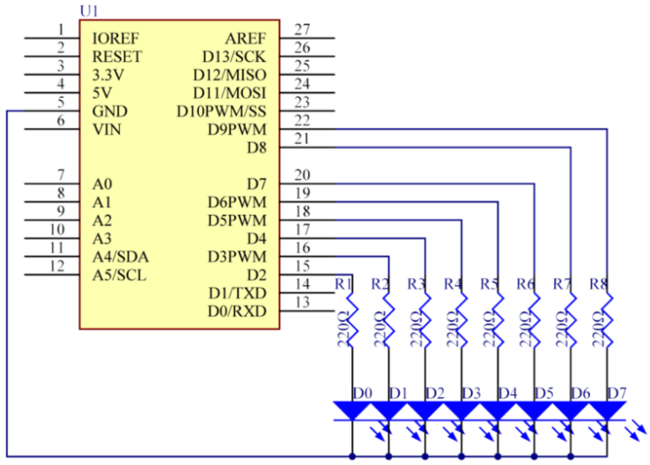
Princip

Princip tohoto experimentu je jednoduchý – zapnout osm LED za sebou.

Postup experimentu

Krok 1: Vytvoř obvod

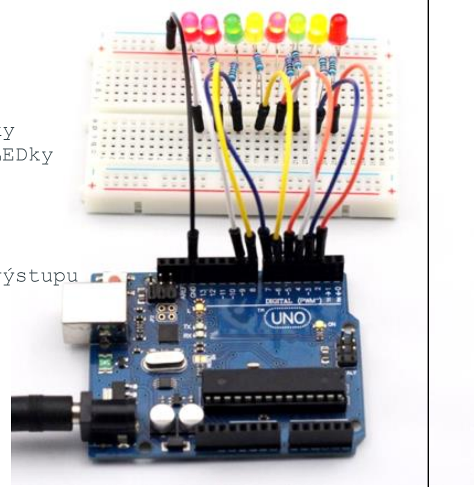




Krok 2: Napiš kód do Arduino IDE

Kód

```
// LED Tekoucí potok
// Tato plynulá světla jsou tvořena osmi LED v řadě, která se postupně rozsvítí a tmavnou jeden po druhém,
stejně jako tekoucí voda.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

const int lowestPin = 2;                          // číslo pinu první LEDky
const int highestPin = 9;                         // číslo pinu poslední LEDky

void setup() {
         // nastavení pinu od 2 do 9 LEDky jako výstupu
         for(int thisPin = lowestPin;thisPin <= highestPin;thisPin++){
                  pinMode(thisPin,OUTPUT);        // nastavení pinu jako výstupu
         }
}

void loop() {
         // iterace pinu za pinem
         // zapnout LEDky od první do poslední
         for(int thisPin = lowestPin;thisPin <= highestPin;thisPin++) {
                  digitalWrite(thisPin,HIGH);  // zapnout tuto LEDku
                  delay(100);                  // počkat 100 ms
         }
         // vypnout LEDky od poslední do první
         for(int thisPin = highestPin;thisPin>=lowestPin;thisPin--) {
                  digitalWrite(thisPin,LOW);   // vypnout tuto LEDku
                  delay(100);                                                  // počkat 100 ms
         }
         for(int thisPin = highestPin;thisPin>=lowestPin;thisPin--) {
                  digitalWrite(thisPin,HIGH);  // zapnout tuto LEDku
                  delay(100);                                                  // počkat 100 ms
         }
         for(int thisPin = lowestPin;thisPin <= highestPin;thisPin++) {
                  digitalWrite(thisPin,LOW);   // vypnout tuto LEDku
                  delay(100);                                                   // počkat 100 ms
         }
}
```



Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Nyní bys měl vidět osm LED světel, která se rozzáří jedno po druhém - zleva doprava, a potom ztlumeně postupně - zprava doleva. A zase opačně. Celý tento proces se bude opakovat, dokud bude obvod napájen.

Lekce 4 Interaktivní tekoucí LED světla

Úvod

V minulé lekci ses už naučil, jak se dělají tekoucí LED světla. V této lekci přidáme potenciometr, který bude měnit interval blikání pomocí nastavení potenciometru.

Komponenty

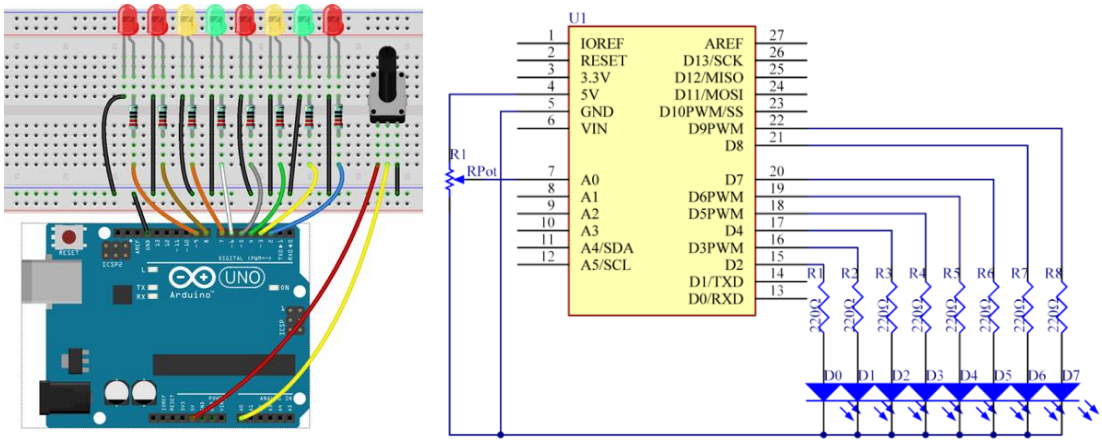
- Arduino
- Breadboard
- 8x LED
- 8x Odpor (220Ω)
- Potentiometer
- USB kabel
- Breadboard vodiče

Princip

Princip je poměrně jednoduchý. Zapne se 8 LEDek podle pořadí. A pak se změní časový interval zapnutí a vypnutí LED pomocí nastavení potenciometru.

Postup experimentu

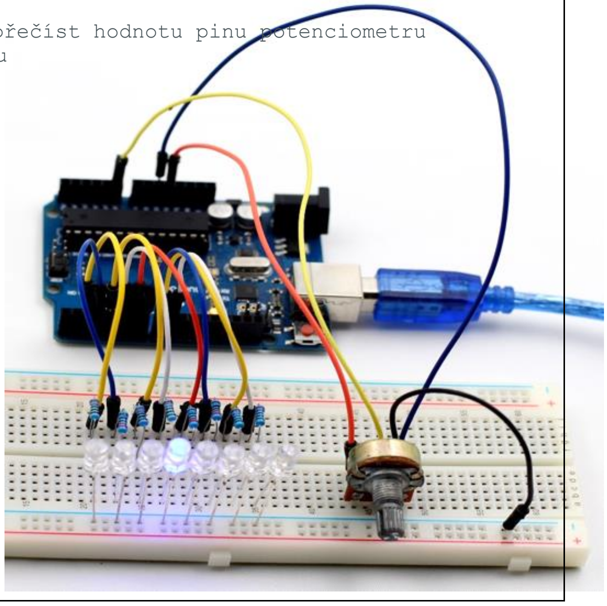
Krok 1: Vytvoř obvod



Krok 2: Napiš kód do Arduino IDE

Kód

```
// Interaktivní LED tekoucí potok
// Zde bys měl vidět osm LEDek, které se budou rozsvěcovat jedna po druhé.
// Nastav potenciometr a uvidíš, že se interval rozsvěcování LEDek změnil.
// Email:support@sunfounder.com
// Web:www.sunfounder.com
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

int ledNum = 8;                                 // počet LEDek
byte ledPin[] = { 2, 3, 4, 5, 6, 7, 8, 9};      // vytvoříme pole pro LED piny
int ledDelay;                                   // pauza mezi změnami
int direction = 1;
int currentLED = 0;
unsigned long changeTime;
int potPin = A0;                                // nastavení pinu potentiometru jako vstupu

void setup() {
        for (int x = 0; x < ledNum; x++) {
                // nastavení pinu od 2 do 9 LEDky jako výstupu
                pinMode(ledPin[x], OUTPUT);
        }
        changeTime = millis();
}

void loop() {
        ledDelay = analogRead(potPin);                  // přečíst hodnotu pinu potenciometru
        // tady se změní interval po nastavení potenciometru
        if ((millis() - changeTime) > ledDelay) {
                changeLED();
                changeTime = millis();
        }
}

void changeLED() {
        for (int x=0; x < ledNum; x++) {
                // vypnout všechny LEDky
                digitalWrite(ledPin[x], LOW);
        }
        // zapnout tuto LEDku
        digitalWrite(ledPin[currentLED], HIGH);
        currentLED += direction;
        // změnit směr, pokud jsme dosáhli konce
        if (currentLED == ledNum-1) {
                direction = -1;
        }
        if (currentLED == 0) {
                direction = 1;
        }
}
```



Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Nyní bys měl vidět osm LED světel, která se rozzáří jedno po druhém, podle pořadí. Nastav potenciometr a přijdeš na to, že časový interval zapnuti a vypnuti LEDek se změní.

Lekce 5 Používaní skeneru sběrnicí I²C

Úvod

Dozvěděli jsme se od našeho zákazníka, že existují převodníky, který mají jinou I2C adresu, než popisujeme v další lekci (prý dostal převodník s adresou 0x3F, což se samozřejmě může stát). Tak jsme se rozhodli přidat tuhle lekci a naučit Tě skenovat adresy na sběrnici I2C.

Komponenty

- Arduino
- USB kabel
- I²C zařízení (I2C LCD displej nebo Hodiny reálného času)
- Breadboard vodiče

Princip

Každé zařízení s I2C sběrnicí má adresu I2C, kterou používá k přijímání příkazů nebo odesílání zpráv. Bohužel je to jedna z oblastí, kde dokumentace často chybí. Tak se pojďme podívat a najít adresu na příkladu I2C převodníku pro LCD displej.

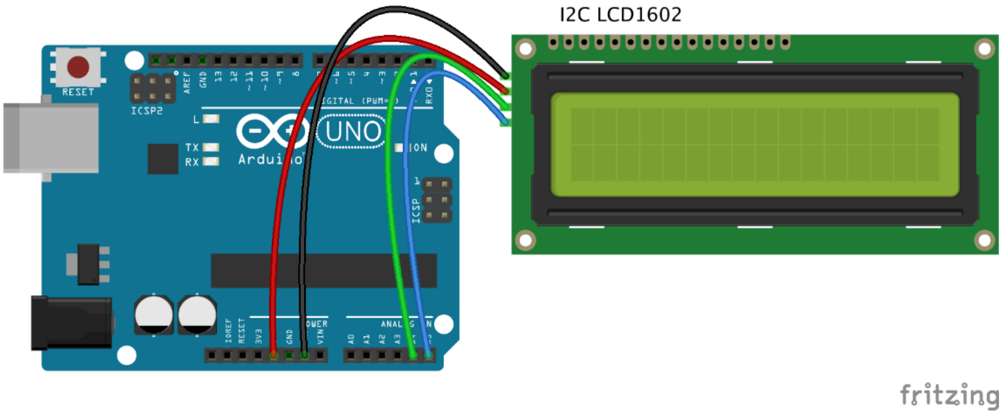
Tento velmi jednoduchý sketch skenuje sběrnici I2C pro zařízení. Pokud je zařízení nalezeno, je hlášeno v sériovém monitoru Arduino.
Sketch zobrazuje 7-bitové adresy nalezených zařízení jako hexadecimální hodnoty. Tato hodnota může být použita pro funkci "Wire.begin", která používá 7-bitovou adresu. Každé nalezené zařízení na sběrnici I2C je hlášeno. Zapojovat a odpojovat I2C zařízení můžeš za běhu aplikace.

Postup experimentu

Krok 1: Vytvoř obvod

Spojení mezi I2C LCD1602 a Arduino:

I2C LCD1602	Arduino Uno	Arduino Mega
GND	GND	GND
VCC	5V	5V
SDA	A4	20
SCL	A5	21



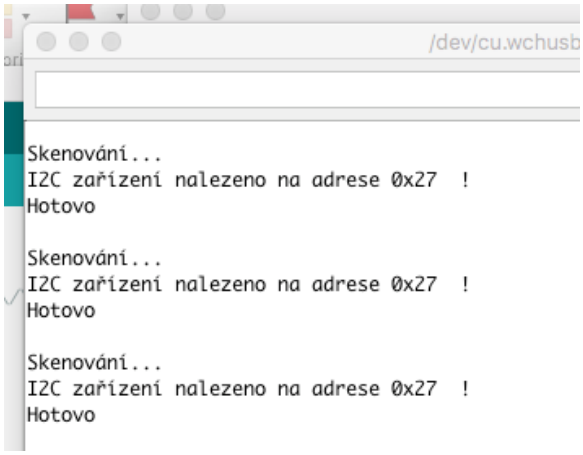
Krok 2: Napiš kód do Arduino IDE

Kód

```
// Používaní skeneru sběrnicí I²C
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

// připoj knihovny
#include <Wire.h>

void setup() {
        Wire.begin();
        Serial.begin(9600);
        while (!Serial);              // Leonardo: počkat na sériový monitor
        Serial.println("\nI2C Skener");
}

void loop() {
        byte error, address;
        int nDevices;

        Serial.println("Skenování...");

        nDevices = 0;
        for(address = 1; address < 127; address++ )     {
                // i2c_skener používá vrácenou hodnotu
                // Write.endTransmisstion pro zjištění,
                // zda zařízení potvrdilo adresu.
                Wire.beginTransmission(address);
                error = Wire.endTransmission();

                if (error == 0)     {
```

```
                        Serial.print("I2C zařízení nalezeno na adrese 0x");
                        if (address<16)
                                Serial.print("0");
                        Serial.print(address,HEX);
                        Serial.println("  !");

                        nDevices++;
                } else if (error==4) {
                        Serial.print("Neznámá chyba na adrese 0x");
                        if (address<16)
                                Serial.print("0");
                        Serial.println(address,HEX);
                }
        }
        if (nDevices == 0)
        Serial.println("Žádné I2C zařízení nenalezeno\n");
        else
        Serial.println("Hotovo\n");

        delay(5000);              // počkat 5 vteřin
}
```

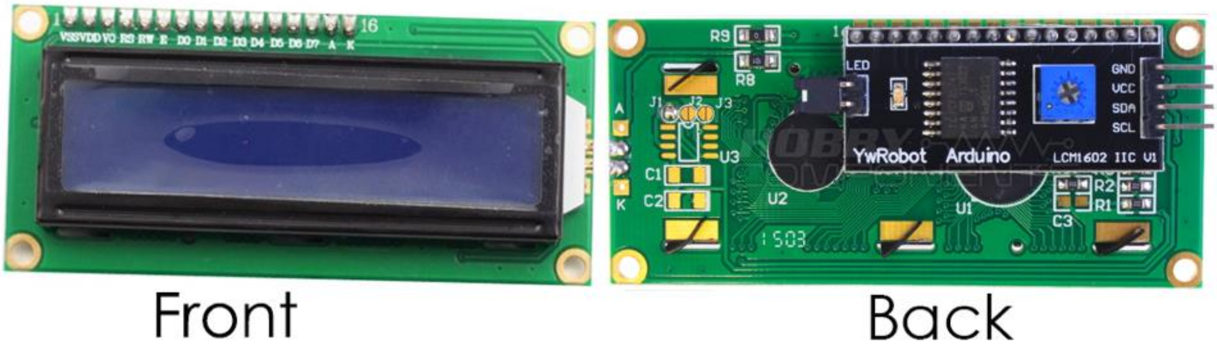
Krok 4: Nahraj sketch do Arduino
Nyní bys na svém displeji měl vidět zařízení na sběrnici I2C s adresou.



Lekce 6 I2C LCD displej

Úvod

Jak jistě všichni víme, LCD a další obrazovky výrazně obohacují naší interakci s přístroji, avšak mají i jednu nevýhodu. Pokud jsou připojeny ke kontroleru, bude obsazeno několik IO portů, kterých je omezený počet. A právě z těchto důvodů byl vyvinut převodník se I2C sběrnicí pro 1602 a 2004 LCD displej, který daný problém vyřešil.
I²C (anglicky Inter-Integrated Circuit, čteme I-squared-C, nesprávně I-two-C) je multi-masterová počítačová sériová sběrnice vyvinutá firmou Philips, která je používána k připojování nízkorychlostních periferií k základní desce, mikrokontroléru, aj. Praktická identická sběrnice se skrývá i pod zkratkou TWI (Two Wire Interface - dvoudrátové rozhraní), kterou používá firma Atmel a další výrobci, namísto chráněné značky I2C. Modrý potenciometr na převodníky se používá k nastavení podsvícení. I²C používá pouze dva obousměrné kanály - datový kanál SDA a hodinový signál SCL s pull-up rezistory. Používá se napětí +5 V nebo +3,3 V, povoleny jsou však i systémy s jiným napětím.



Komponenty

- Arduino
- LCD displej
- USB kabel
- Kabely Dupont samec-samice

Princip

V tomto pokuse necháme I2C LCD1602 displej zobrazit “Laskarduino” a “Zdravim bastlire!" pomocí programování.

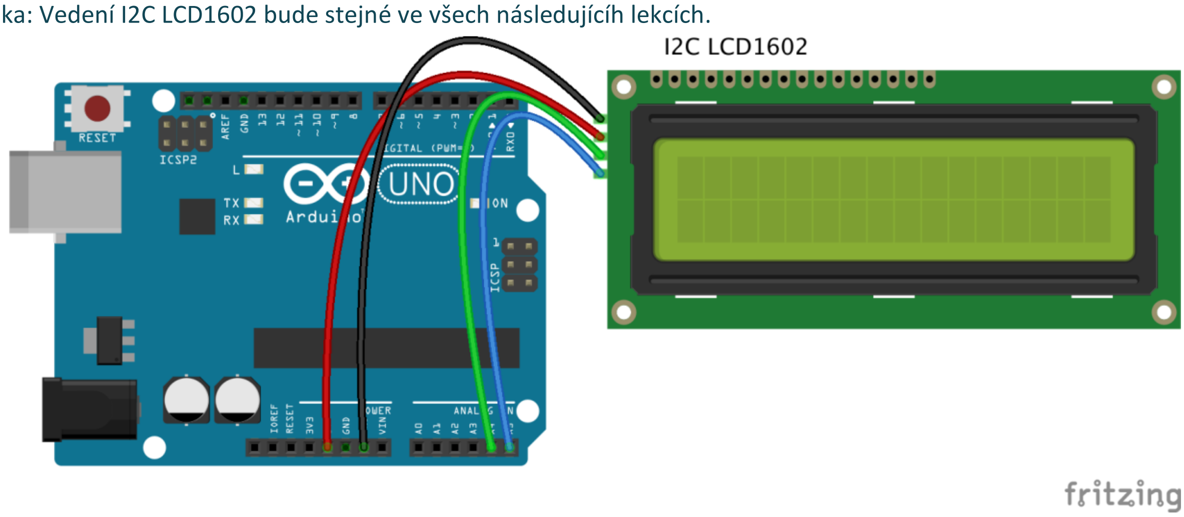
Postup experimentu

Krok 1: Vytvoř obvod

Spojení mezi I2C LCD1602 a Arduino:

I2C LCD1602	Arduino Uno	Arduino Mega
GND	GND	GND
VCC	5V	5V
SDA	A4	20
SCL	A5	21

Poznámka: Vedení I2C LCD1602 bude stejné ve všech následujícíh lekcích.



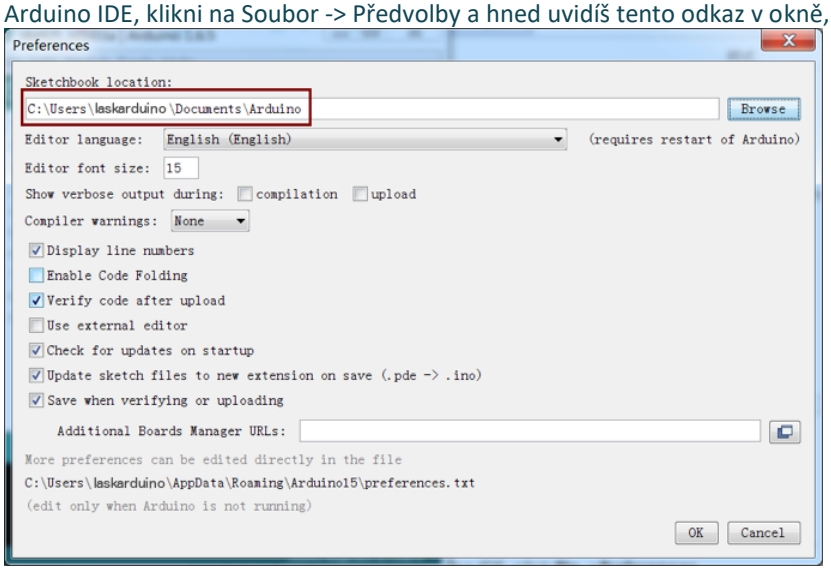
Krok 2: Napiš kód do Arduino IDE

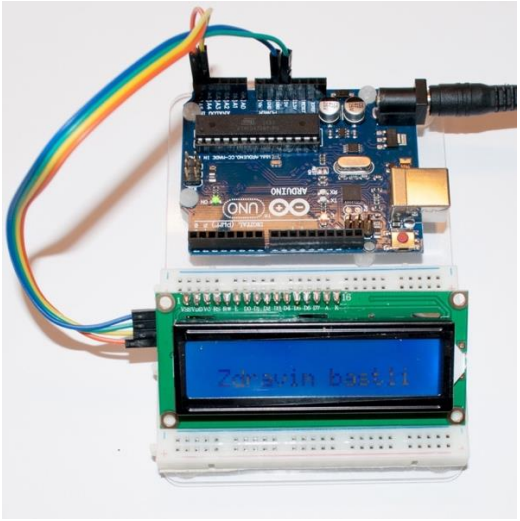
Kód

```
// I2C LCD displej
// Teď by jsi měl na svém displeji vidět běžící text:“Laskarduino” a "Zdravim bastlire!".
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

// připoj knihovny
#include <Wire.h>
```

```
#include <LiquidCrystal_I2C.h>

char array1[]=" Laskarduino              ";     // první řádek displeje (26 znaků!)
char array2[]=" Zdravim bastlire!        ";     // druhý řádek displeje (26 znaků!)
int tim = 500;  //délka pauzy
// inicializace knihovny displeje
LiquidCrystal_I2C lcd(0x27,16,2);               // nastavit adresu displeje na 0x27 pro 1602 displej

void setup() {
          lcd.init();
                    // inicializace lcd
          lcd.backlight();                      // zapnout podsvícení
}

void loop() {
          lcd.setCursor(15,0);                  // nastavení kurzoru na sloupec 15, řádek 0
          for (int positionCounter1 = 0; positionCounter1 < 26; positionCounter1++) {
                    lcd.scrollDisplayLeft();    // Protáhnout obsah zprávy na displeji zprava do leva.
                    lcd.print(array1[positionCounter1]); // Zobrazit zprávu na displeji.
                    delay(tim);                 // počkat 500 ms
          }
          lcd.clear();                          // vyčistit displej a přesunout kurzor na začátek.
          lcd.setCursor(15,1);                  // nastavení kurzoru na sloupec 15, řádek 1
          for (int positionCounter = 0; positionCounter < 26; positionCounter++) {
                    lcd.scrollDisplayLeft();    // protáhnout obsah zprávy na displeji zprava do leva.
                    lcd.print(array2[positionCounter]);   // zobrazit zprávu na displeji.
                    delay(tim);
                     // počkat 500 ms
          }
          lcd.clear();                          // vyčistit displej a přesunout kurzor na začátek.
}
```

Krok 3 Některé lekce potřebují knihovny, které nejsou součástí Arduino IDE, takže je potřeba je přidat před kompilací. Rozbal stažený soubor. Zkopíruj složky ve složce Library do složky Libraries v Arduino (pokud nemůžeš najít adresu v Arduino, otevři Arduino IDE, klikni na Soubor -> Předvolby a hned uvidíš tento odkaz v okně, viz obrázek. Zkompiluj kód.





Krok 4: Nahraj sketch do Arduino
Nyní bys na svém displeji měl vidět “Laskarduino” a “Zdravim bastlire!".

Pokud na displeji nic nevidíš, zkontroluj nastavení odporového trimru pro řízení kontrastu na převodníku. Může se stát, že je nastavený na minimum a tudíž na displeji není nic vidět. Otáčením si nastav správný kontrsat, tak aby byl nápis na displeji co nejlépe čitelný.

Lekce 7 Bzučák

Úvod

Bzučák můžeš využít kdykoli, pokud budeš chtít použít nějaký zvuk.

Komponenty

- Arduino
- Breadboard
- USB kabel
- Aktivní bzučak
- Breadboard vodiče

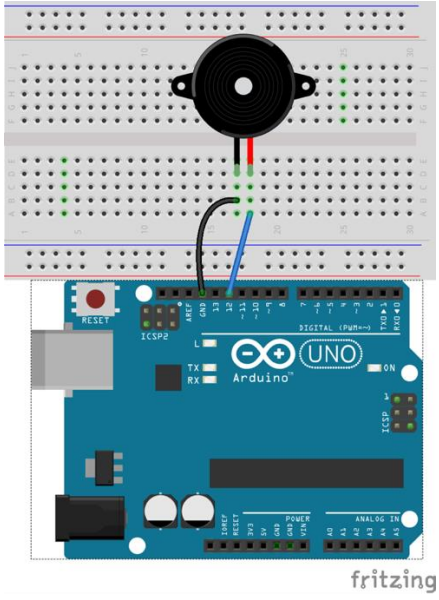
Princip

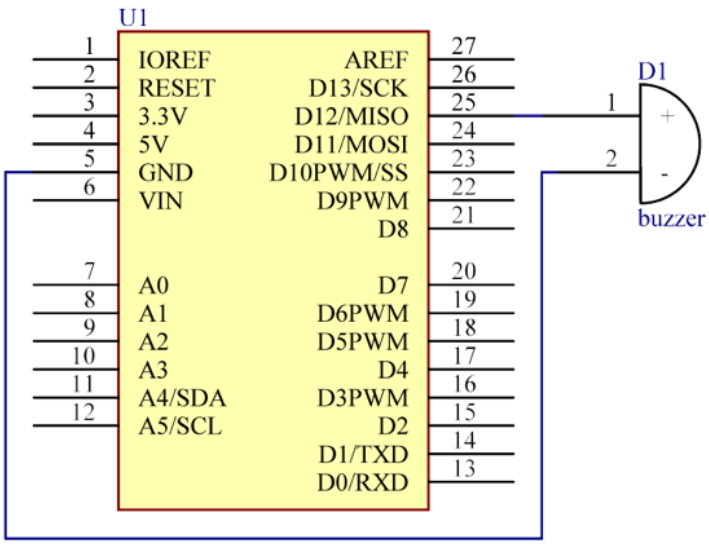
Jedná se o elektrický bzučák s integrovanou strukturou, a proto jsou bzučáky, jež jsou napájeny stejnosměrným napětím, hojně využívány v počítačích, tiskárnách, kopírkách, alarmech, elektrických hračkách, automobilových elektronických zařízeních, telefonech, časovačích a jiných elektronických výrobcích s hlasovým zařízením. Bzučáky můžeme rozdělit na aktivní nebo pasivní (viz následující obrázek). Otoč kolíčky obou bzučáků lícem nahoru - ten se zelenou kulatou destičkou je pasivní, zatímco ten zalitý černým lepidlem (polymerem) je bzučák aktivní.

Rozdíl mezi aktivním a pasivním bzučákem:
Aktivní bzučák má v sobě zabudovaný oscilační zdroj, takže bude vydávat zvuky, pokud bude pod proudem. Pasivní bzučák ale takový zdroj nemá, proto při použití stejnosměrného napětí pípat nebude – místo toho je potřeba použít čtvercové vlny s frekvencí mezi 2K a 5K. Aktivní bzučák bývá často nákladnější a to právě z důvodu několika zabudovaných oscilátorů. V následujícím pokusu použijeme aktivní bzučák.

Postup experimentu

Krok 1: Vytvoř obvod
Schéma zapojení





Krok 2: Napiš kód do Arduino IDE

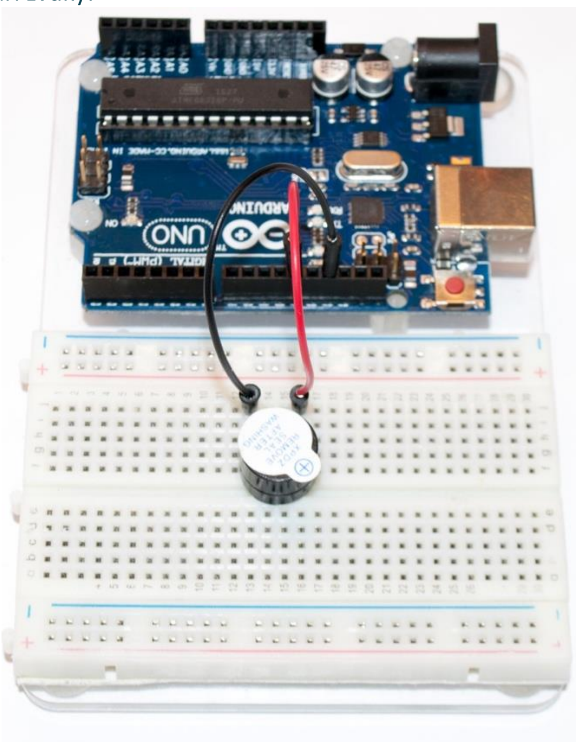
Kód

```
// Bzučák
// Bzučák můžeš využít kdykoli, pokud budeš chtít použít nějaký zvuk.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

int buzzer = 12;                          // číslo pinu bzučáku
void setup() {
         pinMode(buzzer,OUTPUT);          // nastavení pinu bzučáku jako výstupu
}
void loop() {
         unsigned char i;                  // definuj proměnnou
         while(1) {
                  // generace frekvence
                  for(i=0;i<80;i++) {
                           digitalWrite(buzzer,HIGH);
                           delay(1);              // počkat 1ms
                           digitalWrite(buzzer,LOW);
                           delay(1);              // počkat 1ms
                  }
                  //generace jiné frekvence
                  for(i=0;i<100;i++) {
                           digitalWrite(buzzer,HIGH);
                           delay(2);              // počkat 2ms
                           digitalWrite(buzzer,LOW);
                           delay(2);              // počkat 2ms
                  }
         }
}
```

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Nyní bys měl slyšet, jak bzučák vytváří zvuky.



Lekce 8 Otřesové čidlo

Úvod

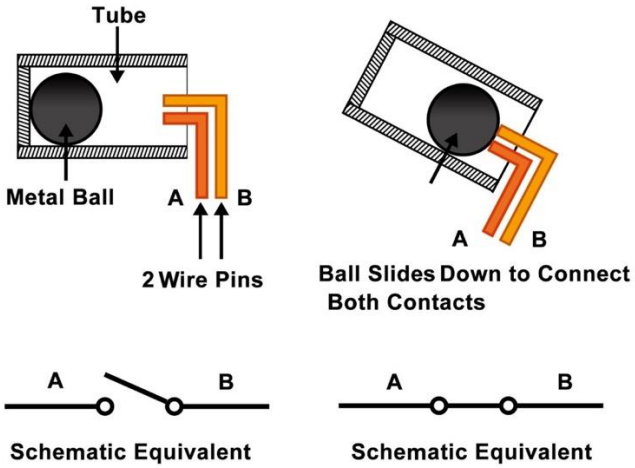
Jde o kulový spínač s kovovou kuličkou uvnitř. Používá se k detekci malého úhlu sklonu.

Komponenty

- Arduino
- USB kabel
- Otřesové čidlo
- Breadboard vodiče

Princip

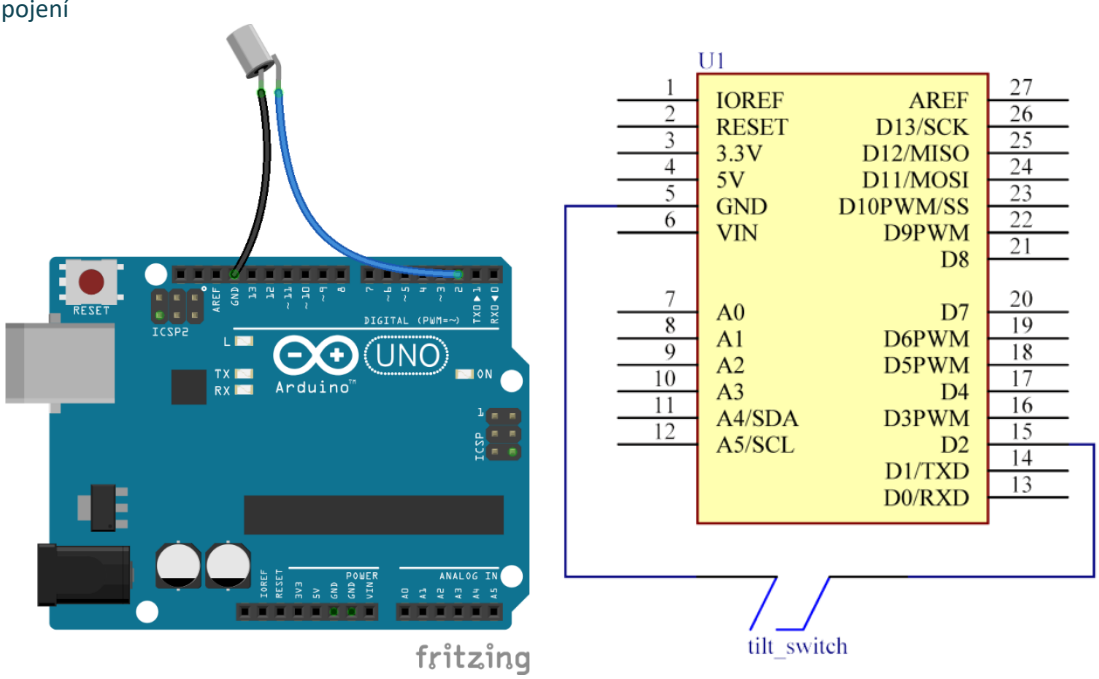
Princip je velice jednoduchý. Je-li spínač nakloněn v určitém úhlu, kulička uvnitř sjede dolů a dotkne se dvou kontaktů připojených ke vnějším kolíčkům. Tím propojí kontakty. Pokud zůstane kulička mimo kontakty, rozpojí je.



Postup experimentu

Krok 1: Vytvoř obvod

Schéma zapojení



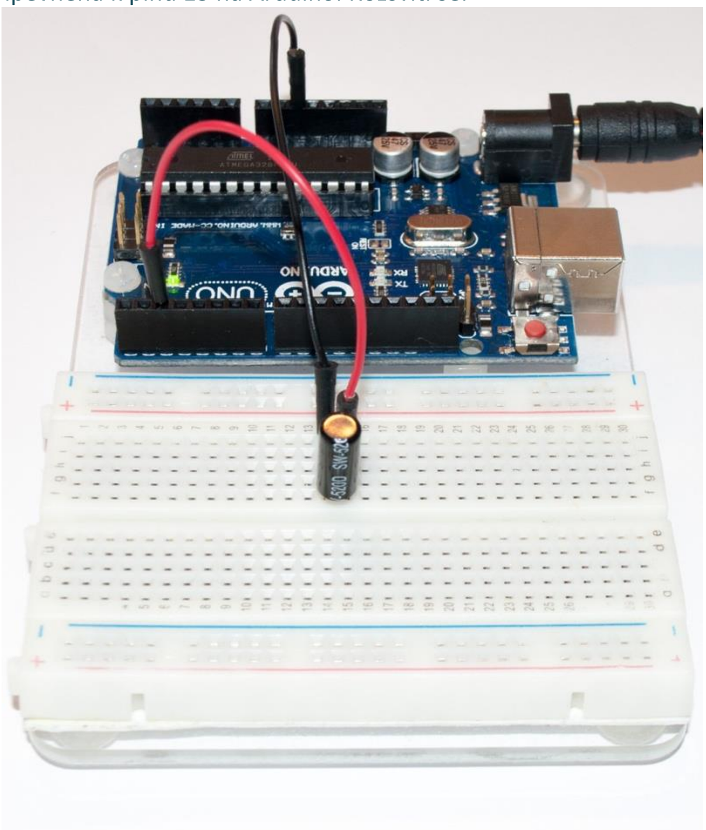
Krok 2: Napiš kód do Arduino IDE

Kód

```
// Otřesové čidlo
// Zapíná a vypíná LEDku když nakloniš čidlo.
// Email:laskarduino@gmail.com
```

```
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

const int ledPin = 13;                                    // číslo pinu LEDky
const int tiltPin = 2;                                    // číslo pinu otřesového čidla

void setup() {
         pinMode(ledPin,OUTPUT);                 // nastavení pinu LEDky jako výstupu
         pinMode(tiltPin,INPUT_PULLUP);          // nastavení pinu otřesového čidla jako vstupu
}
void loop() {
         int digitalVal = digitalRead(tiltPin);// přečíst stav hodnoty pinu otřesového čidla
         delay(300);                             // počkat 300ms
         if(HIGH == digitalVal) {                // pokud není otřesové čidlo nakloněno
                  digitalWrite(ledPin,LOW);      // vypnout LEDku
         } else {                                // pokud je otřesové čidlo nakloněno
                  digitalWrite(ledPin,HIGH);     // zapnout LEDku
         }
}
```

Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Pohněte čidlem a LED, jež je připevněna k pinu 13 na Arduino. Rozsvítí se.



Lekce 9 Kvízový bzučák

Úvod

Ve vědomostních soutěžích, především těch zábavních (např. soutěže v odpovídání na otázky), organizátoři často používají bzučákový systém k přesnému a spravedlivému určení čísla odpovídajícího. Nyní je systém schopný objektivně ilustrovat přesnost a spravedlivost soutěže, čímž dělá pořad zábavnější. V této lekci budeme používat tlačítka, bzučáky a LED a vytvoříme kvízový bzučák.

Komponenty

- Arduino

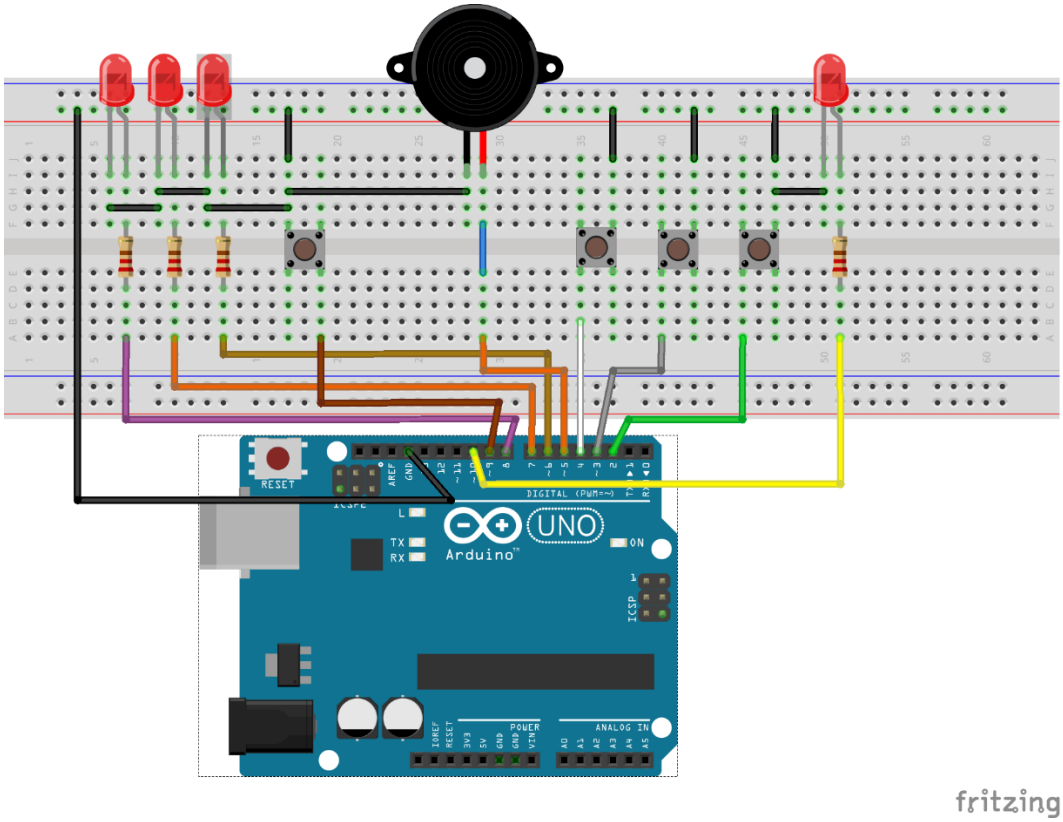
- USB kabel
- 4 Tlačítka
- 4x LED
- 4x Odpor (220Ω)
- Aktivní bzučak
- Breadboard vodiče
- Breadboard

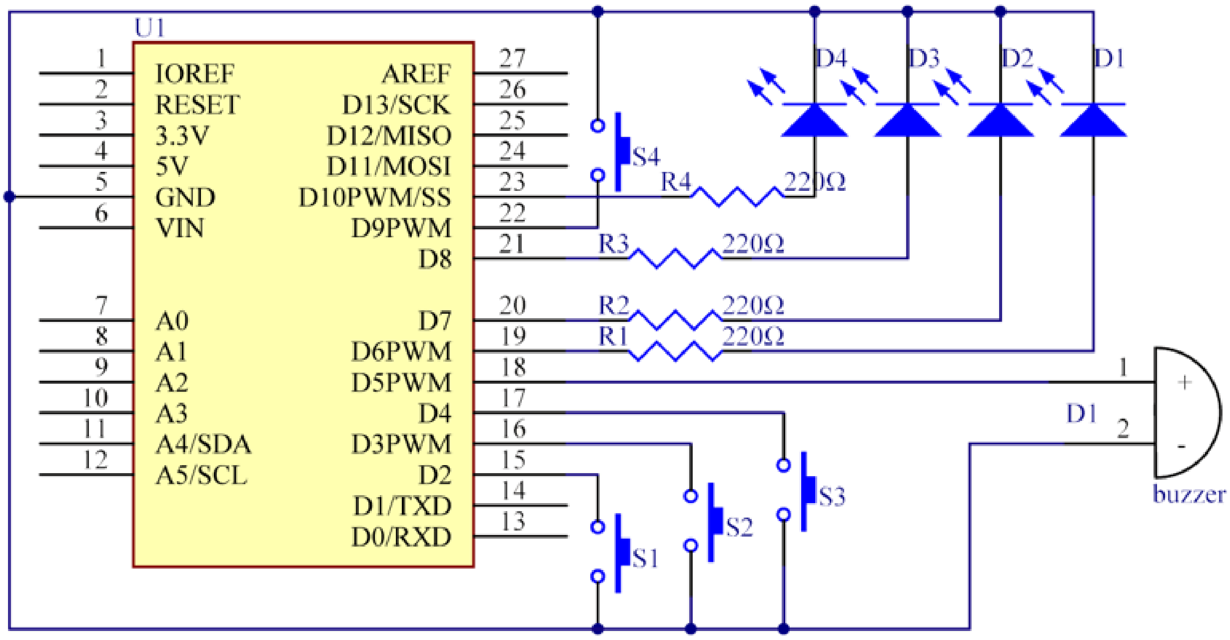
Princip

Tlačítka 1, 2 a 3 slouží k odpovídání a tlačítkem 4 se resetuje. Pokud se nejprve zmáčke tlačítko 1, bzučák pípne, odpovídající LED se rozsvítí a všechny další LED zhasnou. Chceme-li začít další kolo, jednoduše zmáčkneme tlačítko 4.

Postup experimentu

Krok 1: Vytvoř obvod





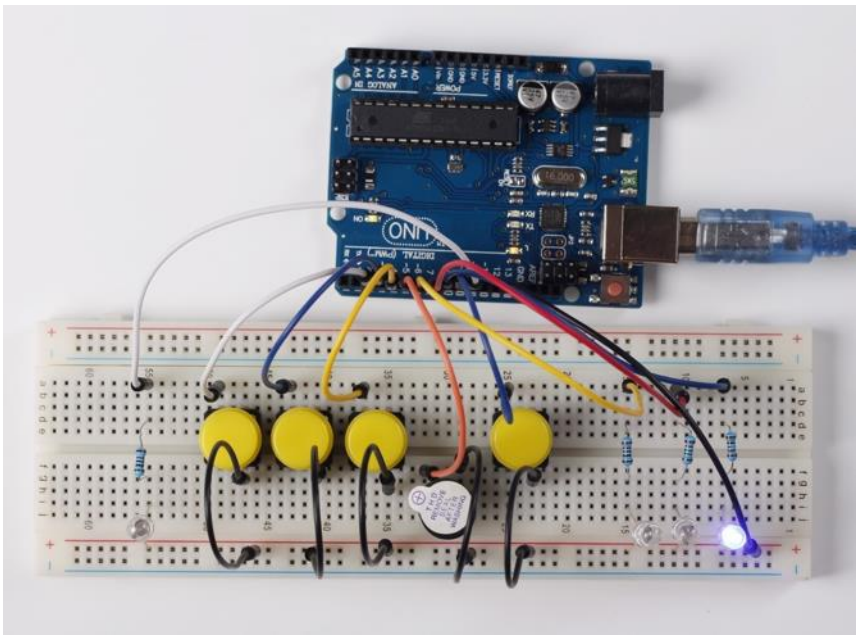
Krok 2: Napiš kód do Arduino IDE

Kód

```
// Kvízový bzučák
// Nejprve zmáčkni tlačítko 4 pro nastartování.
// Pokud nejprve stiskneš tlačítko 1, uvidíš, jak se odpovídající LEDka rozvítí a bzučák zabzučí.
// Poté zmáčkni znovu tlačítko 4 pro restart (toto udělej dřív, než zmáčkneš nějaké jiné tlačítko).
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

#define button1 2             // číslo pinu 1 tlačítka
#define button2 3             // číslo pinu 2 tlačítka
#define button3 4             // číslo pinu 3 tlačítka
#define button4 9             // číslo pinu 4 tlačítka
#define buzzerPin 5           // číslo pinu bzučáku
#define LED1 6                // číslo pinu 1 LEDky
#define LED2 7                // číslo pinu 2 LEDky
#define LED3 8                // číslo pinu 3 LEDky
#define LED4 10               // číslo pinu 4 LEDky
#define uint8 unsigned char
uint8 flag = 0;               // indikace stavu tlačítka 4
uint8 b1State,b2State,b3State,b4State = 0;

void setup() {
         // nastavení pinů LEDek a bzučáku jako výstupu
         pinMode(buzzerPin, OUTPUT);
         pinMode(LED1, OUTPUT);
         pinMode(LED2, OUTPUT);
         pinMode(LED3, OUTPUT);
         pinMode(LED4, OUTPUT);
         // nastavení pinů tlačítek jako vstupu s pullup odporem
         pinMode(button1, INPUT_PULLUP);
         pinMode(button2, INPUT_PULLUP);
         pinMode(button3, INPUT_PULLUP);
         pinMode(button4, INPUT_PULLUP);
         // vypnout všechny LEDky
         digitalWrite(LED1, LOW);
         digitalWrite(LED2, LOW);
         digitalWrite(LED3, LOW);
         digitalWrite(LED4, LOW);
}

void loop() {
         // vypnout všechny LEDky
         digitalWrite(LED1, LOW);
         digitalWrite(LED2, LOW);
```

```
        digitalWrite(LED3, LOW);
        digitalWrite(LED4, LOW);
        // přečíst stav pinu tlačítka 4
        b4State = digitalRead(button4);
        // pokud je tlačítko 4 stisknuté
        if(b4State == 0) {
                flag = 1;  // nastavím flag na 1
                digitalWrite(LED4, HIGH);              // zapnout 4 LEDku
                delay(200);
        }
        if(1 == flag) {
                // přečíst stav pinů tlačítek 1 až 3
                b1State = digitalRead(button1);
                b2State = digitalRead(button2);
                b3State = digitalRead(button3);
                // pokud bylo tlačítko 1 stisknuto jako první
                if(b1State == 0) {
                        flag = 0;
                        digitalWrite(LED4, LOW);
                        Alarm();                       // zvuk bzučáku
                        digitalWrite(LED1,HIGH);       // zapnout jen 1 LEDku
                        digitalWrite(LED2,LOW);
                        digitalWrite(LED3,LOW);
                        // čekání na stisknutí tlačítka 4
                        while(digitalRead(button4));
                }
                // pokud bylo tlačítko 2 stisknuto jako první
                if(b2State == 0) {
                        flag = 0;
                        digitalWrite(LED4, LOW);
                        Alarm();
                        digitalWrite(LED1,LOW);
                        digitalWrite(LED2,HIGH);
                        digitalWrite(LED3,LOW);
                        while(digitalRead(button4));
                }
                // pokud bylo tlačítko 3 stisknuto jako první
                if(b3State == 0) {
                        flag = 0;
                        digitalWrite(LED4, LOW);
                        Alarm();
                        digitalWrite(LED1,LOW);
                        digitalWrite(LED2,LOW);
                        digitalWrite(LED3,HIGH);
                        while(digitalRead(button4));
                }
        }
}

// zvuk bzučáku
void Alarm() {
        for(int i=0;i<300;i++){
                digitalWrite(buzzerPin,HIGH);          // zapnout bzučák
                delay(1);                              // pauza nastaví frekvenci
                digitalWrite(buzzerPin,LOW);           // vypnout bzučák
                delay(1);                              // pauza nastaví frekvenci
        }
}
```

Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Nyní nejprve zmáčkni tlačítko 4. Pokud prvně zmáčkneš tlačítko 1, rozsvítí se odpovídající LED a pípne bzučák. Poté opět zmáčkni tlačítko 4 k resetování, než začneš s dalšími tlačítky.



Lekce 10 Sériový monitor

Úvod

V této lekci se naučíme, jak zapnout a vypnout LED pomocí počítače a seriového monioru. Sériový monitor se používá ke komunikaci mezi Arduino deskou a počítačem nebo jinými přístroji. Jedná se o zabudovaný software v Arduino prostředí a otevře se kliknutím na tlačítko v pravém horním rohu. Můžeš posílat a získávat data pomocí sériového portu na Arduino desce a ovládat desku pomocí klávesnice. V této lekci používáme 3 LED, a proto můžeš vložit červenou, zelenou a žlutou barvu na sériový monitor v IDE. Odpovídající LED na Arduino se tímto rozsvítí.

Komponenty

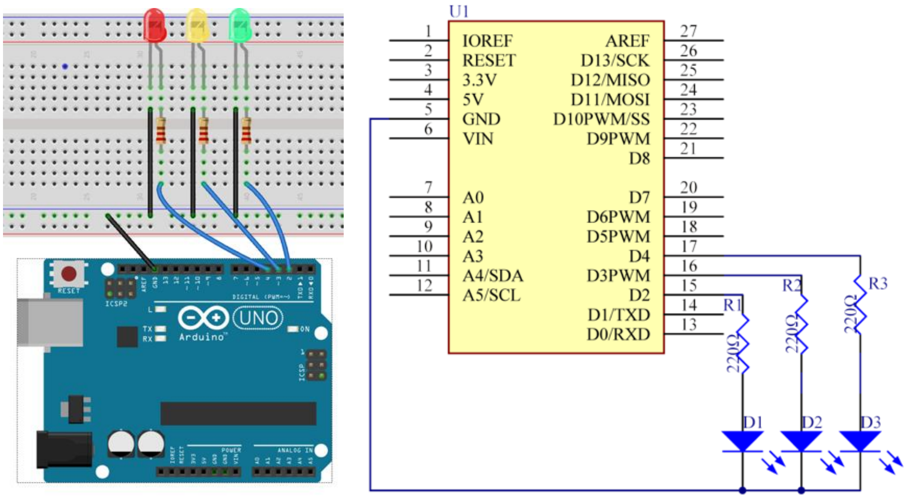
- Arduino
- Breadboard
- 3x LED
- 3x Odpor (220Ω)
- Breadboard vodiče
- USB kabel

Princip

Zde slouží sériový monitor jako přestupní stanice pro komunikaci mezi tvým počítačem a Arduino. Nejprve převede počítač data do sériového monitoru a ta jsou poté přečtena pomocí Arduino Uno (nebo Mega). Poté provede Uno další operace.

Postup experimentu

Krok 1: Vytvoř obvod

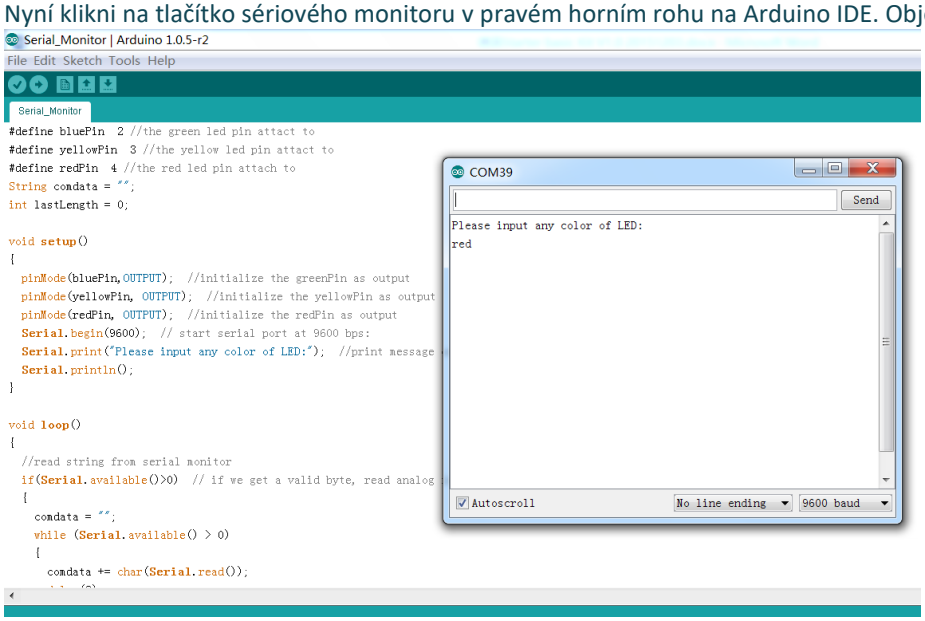


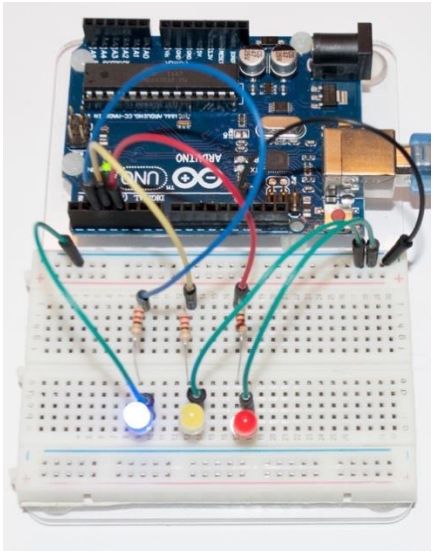
Krok 2: Napiš kód do Arduino IDE

Kód

```
// Sériový monitor
// Otevři sériový monitor. Pokud do monitoru napíšeš "cervena" a stiskneš Enter,
// rozsvítí se červené LED světlo. Zkus další barvy!
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

const int modraPin= 2;       // číslo pinu modré LEDky
const int zlutaPin= 3;       // číslo pinu žluté LEDky
const int cervenaPin= 4;     // číslo pinu červené LEDky
String comdata = "";
int lastLength = 0;

void setup() {
         pinMode(modraPin,OUTPUT);                   // nastavení pinu modré LEDky jako výstupu
         pinMode(zlutaPin, OUTPUT);                  // nastavení pinu žluté LEDky jako výstupu
         pinMode(cervenaPin, OUTPUT);                // nastavení pinu červené LEDky jako výstupu
         Serial.begin(9600);                         // spustit sériový monitor na 9600 bps
         Serial.print("Please input any color of LED:");  // tisknout zprávu do sériového monitoru
}

void loop() {
         // jestli jsi neco napsal do monitoru
         if(Serial.available()>0) {
                  comdata = "";
                  while (Serial.available() > 0) {
                           comdata += char(Serial.read());// sbírame "char" do "stringu"
                           delay(2);
                  }
                  Serial.println(comdata);
         }
         if(comdata == "cervena") {
                  digitalWrite(cervenaPin, HIGH);        // zapnout červenu LEDku
                  digitalWrite(modraPin, LOW);           // vypnout modrou LEDku
                  digitalWrite(zlutaPin, LOW);           // vypnout žlutou LEDku
         } else if(comdata == "zluta") {
                  digitalWrite(cervenaPin, LOW);         // vypnout červenu LEDku
                  digitalWrite(modraPin, LOW);           // vypnout modrou LEDku
                  digitalWrite(zlutaPin, HIGH);          // zapnout žlutou LEDku
         } else if(comdata == "modra") {
                  digitalWrite(cervenaPin, LOW);         // vypnout červenu LEDku
                  digitalWrite(modraPin, HIGH);          // zapnout modrou LEDku
                  digitalWrite(zlutaPin, LOW);           // vypnout žlutou LEDku
         } else {
                  digitalWrite(cervenaPin, LOW);         // vypnout červenu LEDku
                  digitalWrite(modraPin, LOW);           // vypnout modrou LEDku
                  digitalWrite(zlutaPin, LOW);           // vypnout žlutou LEDku
         }
}
```

Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Nyní klikni na tlačítko sériového monitoru v pravém horním rohu na Arduino IDE. Objeví se okénko sériového monitoru.





Pomocí tohoto okénka můžeš posílat data z počítače do Arduino, ale také přijímat data a zobrazit si je na obrazovce. Když okno otevřeš, objeví se “Vlož prosím jakoukoli LED barvu.” Zde můžeš vložit barvu. Když vložíš červenou, zelenou nebo žlutou, klikni na Poslat a odpovídající LED se rozsvítí. Pokud ale přidáš jinou barvu, nerozsvítí se žádná LED. Například, pokud přidáš červenou, rozsvítí se červené LED.

Lekce 11 Fotorezistor

Úvod

Fotorezistor či photocell je světlem ovládaný proměnný odpor. Odpor fotorezistoru se snižuje s rostoucí intenzitou dopadajícího světla; jinými slovy, je fotovodivý. Fotorezistor můžeme využít pro na světlo citlivé detektory, jež se aktivují světlem či tmou.

Komponenty

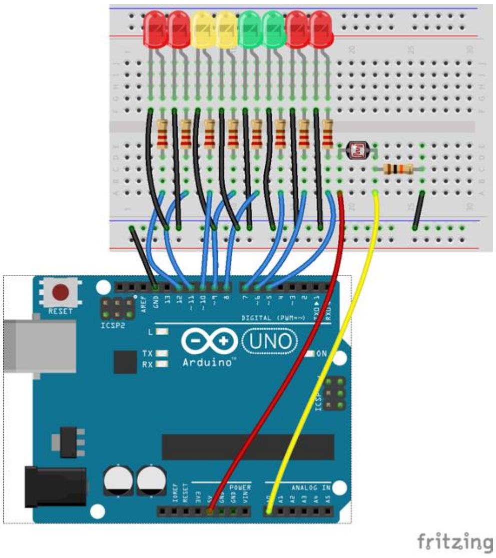
- Arduino
- USB kabel
- Fotorezistor
- Odpor (10KΩ)
- 8x LED
- 8x Odpor (220Ω)
- Breadboard vodiče
- Breadboard

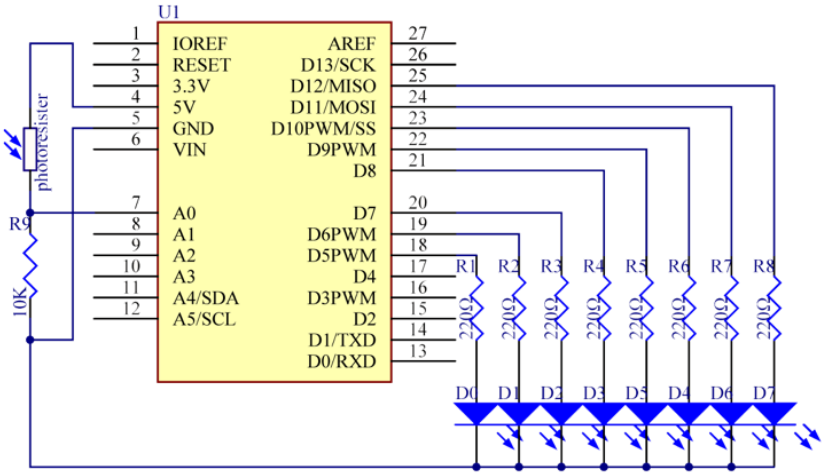
Princip

Odpor ve fotorezistoru se mění na základě intenzity dopadajícího světla. Když se intenzita tohoto světla zvýší, odpor se sníží a naopak. V tomto experimentu použijeme 8 LED. Čím větší bude intenzita světla, tím více LED se rozsvítí. Až dosáhneme určité intenzity světla, rozsvítí se všechny LED. V případě žádného světla se nerozsvítí nic.

Postup experimentu

Krok 1: Vytvoř obvod





Krok 2: Napiš kód do Arduino IDE

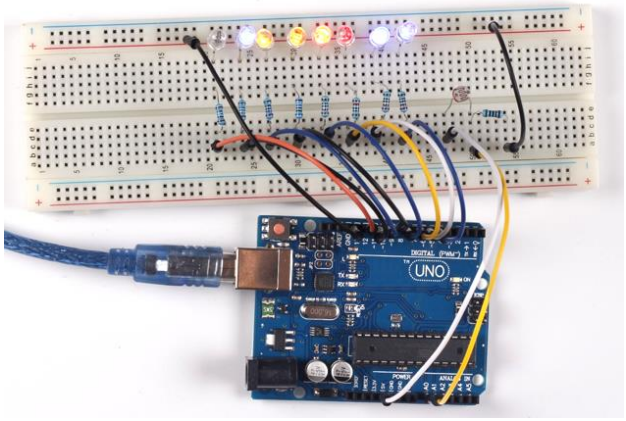
Kód

```
// Fotorezistor
// Pokud bude na fotorezistor svítit světlo určité intenzity, uvidíš určitý počet svítících LEDek.
// Pokud tuto intenzitu zvětšíš, uvidíš víc svítících LEDek.
// Pokud to dáš do tmy, žádná LEDka svítit nebude.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

const int NbrLEDs = 8;                               // počet LEDek
const int ledPins[] = {5, 6, 7, 8, 9, 10, 11, 12};   // vytvoříme pole pro LED piny
const int photocellPin = A0;                         // číslo pinu
fotorezistoru
int sensorValue = 0;                                 // přečteny hodnoty z čidla
int ledLevel = 0;                                    // hodnota čidla převedená do LED řádku

void setup() {
         for (int led = 0; led < NbrLEDs; led++) {
                  pinMode(ledPins[led], OUTPUT);      // nastavím všechny LED piny jako výstup
         }
}

void loop() {
         sensorValue = analogRead(photocellPin);      // přečíst hodnotu pinu A0
         ledLevel = map(sensorValue, 300, 1023, 0, NbrLEDs); // převedeme hodnotu čidla do LED řádku
         for (int led = 0; led < NbrLEDs; led++) {    // půjdeme po každé LEDce
                  if (led < ledLevel ) {             // Jestli je číslo LED méně, než ledLevel
                           digitalWrite(ledPins[led], HIGH);     // zapneme tuto LEDku
                  } else {
                           digitalWrite(ledPins[led],LOW);       // jinak vypneme tuto LEDku
                  }
         }
}
```

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Posviť na fotorezistor a uvidíš několik LED, jak se rozsvítí. Posviť více a rozsvítí se LED o to víc. Jakmile fotorezistor postavíš do tmy, všechna LED zhasnou.



Vyzkoušej!
Kromě výše uvedeného experimentu můžeš také vyměnit fotorezistor za mikrofon a pozorovat, jak LED signalizuje sílu zvuku. Čím vyšší bude síla zvuku, tím vice LED se rozsvítí. Vyzkoušej si tento pokus na vlastní kůži a uvidíš.

Lekce 12 Ovládání zvuku pomocí světla

Úvod

Už ses naučil, jak pracovat s fotorezistorem a nyní se dozvíš, jak ovládat bzučák pomocí fotorezistoru tak, aby pípal v různých frekvencích.

Komponenty

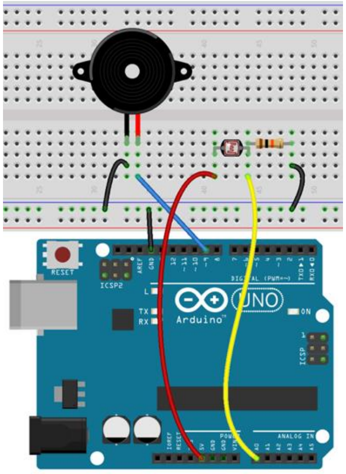
- Arduino
- USB kabel
- Fotorezistor
- Aktivní bzučak
- 1x Odpor (10KΩ)
- Breadboard vodiče
- Breadboard

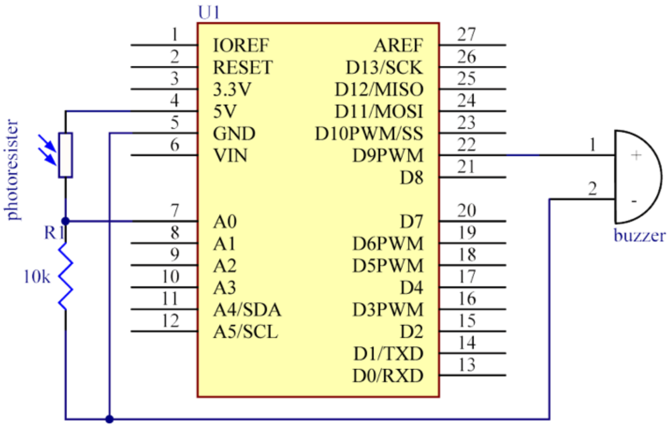
Princip

Když nasvítíte fotorezistor a intenzita dopadajícího světla bude vyšší, odpor fotorezistoru se sníží; a naopak.
V tomto pokusu je výstup fotorezistoru vyslán do pinu A0 na Arduino a poté zpracován ADC na desce, kde vytvoří digitální signál. Tento signál používáme jako parametr funkce delay() v kódu, než se ozve bzučák. Pokud je dopadající světlo silné, je výsledná hodnota vyšší, což znamená, že bude bzučák pípat pomalu; je-li světlo slabé, výsledná hodnota je nižší a bzučák bude pípat rychleji.

Postup experimentu

Krok 1: Vytvoř obvod
Schéma zapojení





Krok 2: Napiš kód do Arduino IDE

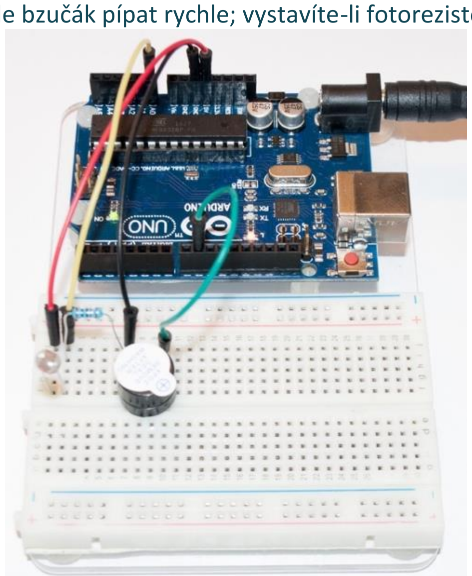
Kód

```
// Ovládání zvuku pomocí světla
// Pokud dáš fotorezistor do tmavého prostředí, bzučák bude intenzivně pípat.
// Pokud dáš fotorezistor do světlého prostředí, bzučák bude pípat pomalu.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

const int photocellPin = A0;                    // číslo pinu fotorezistoru
int sensorValue = 0;                            // přečteny hodnoty z čidla
const int buzzerPin = 9;                        // číslo pinu bzučáku

void setup() {
         pinMode(buzzerPin, OUTPUT);            // nastavení pinů bzučáku jako výstupu
}
void loop() {
         sensorValue = analogRead(photocellPin);// přečíst hodnotu pinu A0
         digitalWrite(buzzerPin, HIGH);         // zapnout bzučák
         delay(sensorValue);                    // pauza nastaví frekvenci
         digitalWrite(buzzerPin, LOW);          // vypnout bzučák
         delay(sensorValue);                    // pauza nastaví frekvenci
}
```

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Pokud postavíte fotorezistor do tmy, bude bzučák pípat rychle; vystavíte-li fotorezistor světlu, bude pípat pomalu.



Lekce 13 Senzor plamene

Úvod

Senzor plamene funguje tak, že zachytí infračervené světlo z plamene. Může být využit k detekci a varování před požárem.

Komponenty

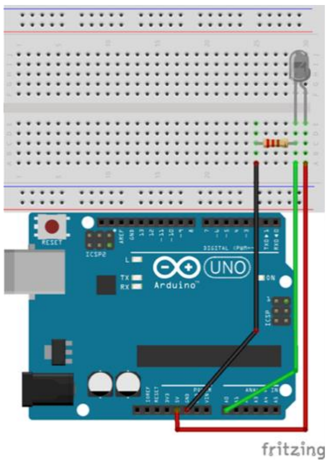
- Arduino
- USB kabel
- Senzor plamene
- Breadboard vodiče

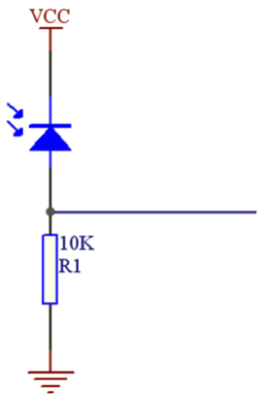
Princip

Existuje několik typů senzoru plamene. V tomto experimentu budeme využívat infračervené čidlo plamene, které dokáže rozpoznat infračervené světlo o vlnové délce od 700nm do 1000nm. Infračervená sonda převede změny vnějšího infračerveného světla do současných změn. Poté převede analogovou kvantitu do digitální kvantity. Kratší kolík senzoru je katoda a ten druhý je anoda. Spoj katodu s 5V pinem a anodu s rezistorem; připoj druhý konec rezistoru ke GND, připoj jeden konec propojovacího vodiče na anodu, druhý konec k analogovému pinu, viz níže.

Postup experimentu

Krok 1: Vytvoř obvod





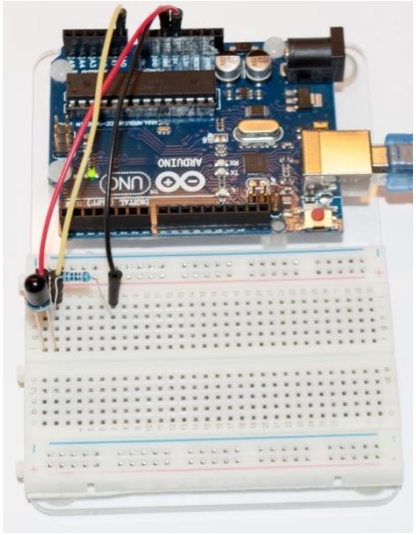
Krok 2: Napiš kód do Arduino IDE

Kód

```
// Senzor plamene
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

int flame = A0;                                 // číslo pinu senzoru
int LED = 13;                                   // číslo pinu LEDky
int val = 0;                                    // inicializace proměny

void setup() {
         pinMode(LED,OUTPUT);                   // nastavení pinu LEDky jako výstupu
         pinMode(flame,INPUT);                  // nastavení pinu senzoru jako vstupu
         Serial.begin(9600);                    // spoustit sériový monitor na 9600 bps
}

void loop() {
         val = analogRead(flame);               // přečíst hodnotu pinu senzoru
         Serial.println(val);                   // tisknout hodnotu pinu senzoru do sériového monitoru
         if(val > 30) {                         // jestli je hodnota > 30
                  digitalWrite(LED,HIGH);       // zapnout LEDku
         } else {
                  digitalWrite(LED,LOW);        // jinak vypnout LEDku
         }
```

```
        delay(500);
}
```

Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Když nyní rozsvítíš zapalovač nedaleko čidla, LED na pinu 13 na Arduino se rozsvítí.



Lekce 14 Voltmetr

Úvod

V této lekci vytvoříme voltmetr s pomocí LCD displeje 1602 a potenciometru.

Komponenty

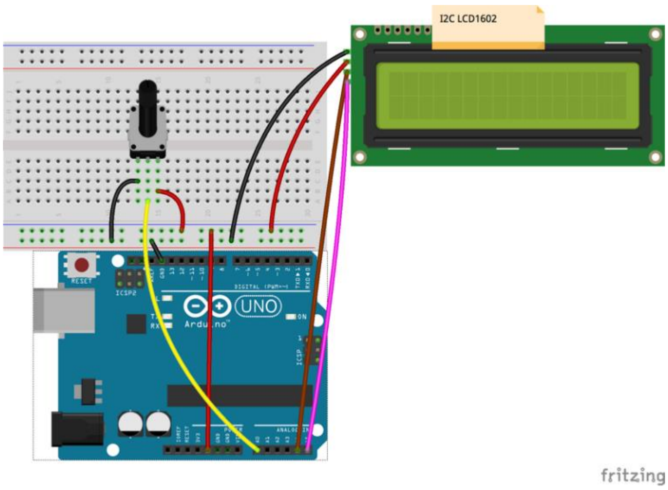
- Arduino
- USB kabel
- Potenciometr
- LCD displej
- Breadboard vodiče
- Breadboard

Princip

V tomto experimentu využijeme potenciometr k dělení napětí. Vzhledem k tomu, že Arduino umí číst pouze digitální signály, ale na výstupech posuvných pinů potenciometru je analogový signál, musíme převést tento analogový signál do digitálního s analogově-digitálním konvertorem (ADC). Naštěstí Arduino samo o sobě je dodáváno s 10-bitovým ADC, které můžeme použít k provedení této konverze.
Poté se zobrazí toto napětí na LCD displeji.

Postup experimentu

Krok 1: Vytvoř obvod



Krok 2: Napiš kód do Arduino IDE

Kód

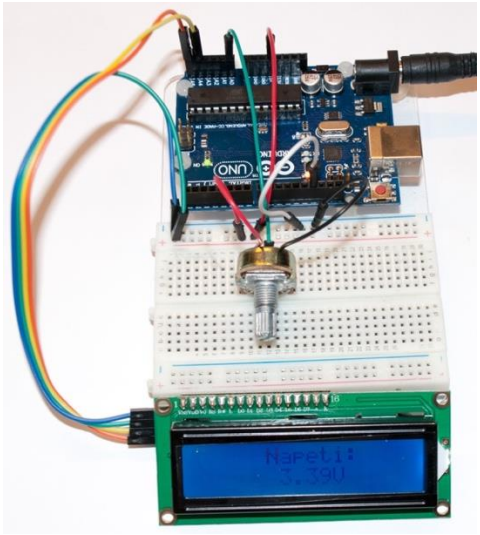
```
// Voltmetr
// Nastav potenciometr a uvidíš, že napětí zobrazené na displeji I2C LCD1602 se změnilo odpovídajícím způsobem.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

// připoj knihovny
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);              // nastavit adresu displeje na 0x27 pro 1602 displej
float val = 0;
int volt = A0;                                 // číslo pinu pro meření napětí

void setup() {
         Serial.begin(9600);                    // spustit sériový monitor na 9600 bps
         lcd.init();                           // inicializace lcd
         lcd.backlight();                      // zapnout podsvícení
         lcd.setCursor(5,0);                   // nastavení kurzoru na sloupec 5, řádek 0
         lcd.print("Napeti:");                 // zobrazit "Napeti:" na displeji
}

void loop() {
         val = analogRead(volt);               // přečíst hodnotu pinu A0
         val = val/1024*5.0;                    // konvertování hodnoty pinu do napětí
         Serial.print(val);                     // tisknout napětí do sériového monitoru
         Serial.println("V");          // tisknout "V" do sériového monitoru a přejít na nový řádek (ln)
         lcd.setCursor(6,1);                    // nastavení kurzoru na sloupec 6, řádek 1
         lcd.print(val);                       // zobrazit napětí na LCD
         lcd.print("V");                       // zobrazit "V" na LCD
         delay(300);                           // počkat 300 ms
}
```

Poznámka: Sem potřebuješ přidat knihovnu. Mrkni se do popisu v lekci 6.

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Nyní otoč potenciometr a uvidíš na LCD displeji napětí pinů A0 na Arduino. Toto napětí bude odpovídat změně provedené potenciometrem.



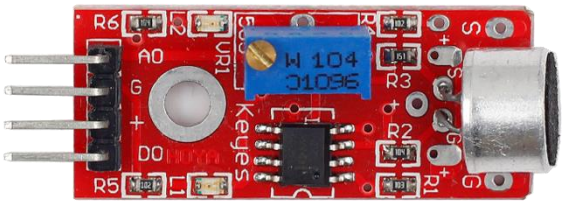
Lekce 15 Zvukový senzor

Úvod

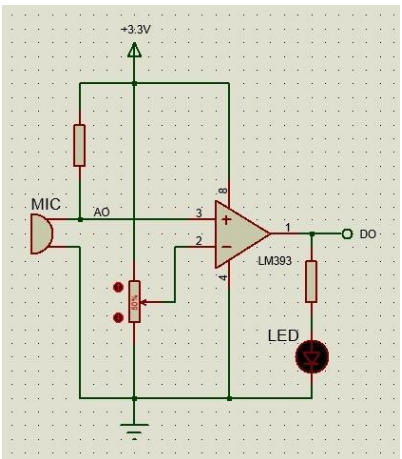
Modul mikrofonu má dva výstupy:
A0: Analogový výstup, který se používá pro výstup napěťových signálů z mikrofonu v reálném čase.
D0: Dosáhne-li intenzita zvuku určité prahové hodnoty (nastavena pomocí potenciometru), bude na výstupu snímače vysoké nebo nízké úrovně.

Komponenty

- Arduino
- USB kabel
- Zvukový senzor
- Breadboard vodiče

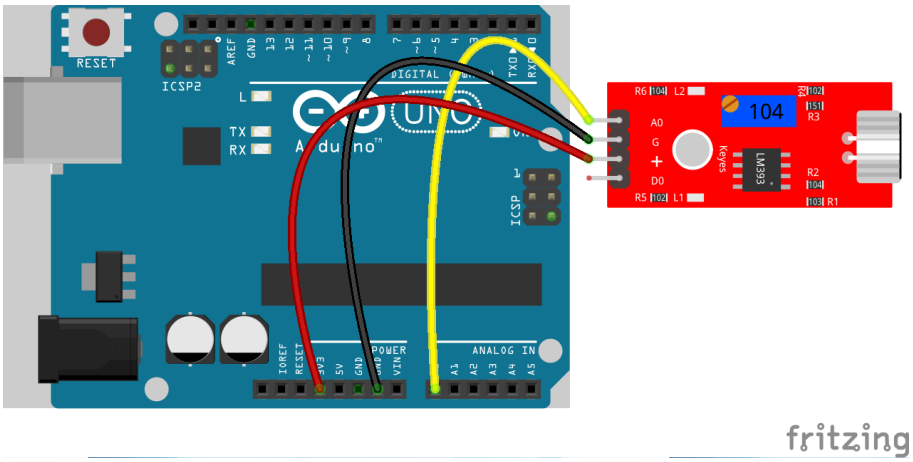


Princip



Mikrofon umí konvertovat audio signál na elektrický signál (analogový), pak převést analogový signál na digitální hodnotu s pomocí ADC, přenést ji na kontrolér a zpracovat. Podívej se na následující obrázek, je na něm schematicky zobrazen modul mikrofonu.
LM393 je komparátor napětí. Pokud je napětí na pinu 3 vyšší, než je invertující pin 2, výstupní pin 1 bude log. 1. V opačném případě log. 0. Za prvé - nastav potenciometr, aby napětí na pinu 2 LM393 bylo méně než 5V. Pokud neslyšíš žádný zvuk, odpor mikrofonu je velmi velký. Napětí na pinu 3 LM393 je blízko k napájecímu napětí (5V). Na pinu DO je log.1 a LEDka svítí. Pokud slyšíš hluk, odpor mikrofonu klesne, pin DO modulu bude mít log.0 a LEDka se vypne.
Postup experimentu
Krok 1: Vytvoř obvod

Voice Sensor	Arduino
AO	A0
G	GND
+	3,3V
DO	8



Krok 2: Napiš kód do Arduino IDE

Kód

```
// Zvukový senzor
// Hladinu intenzity zvuku můžeš vidět na Sériovém monitoru.
// Pokud dosáhne hlasitost určité hodnoty, LEDka na Arduino se rozsvítí.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\*/

const int ledPin = 13;                          // číslo pinu LEDky (integrovaná LED)
const int soundPin = A0;                        // číslo pinu zvukového senzoru

void setup() {
          pinMode(ledPin,OUTPUT);               // nastavení pinu LEDky jako výstupu
          Serial.begin(9600);                   // spustit sériový monitor na 9600 bps
}

void loop() {
          int value = analogRead(soundPin);     // přečíst hodnotu pinu zvukového senzoru
          Serial.println(value);                // tisknout hodnotu do sériového monitoru
          if(value > 25) {                      // jestli je hodnota > 25
                    digitalWrite(ledPin,HIGH);  // zapnout LEDku
                    delay(2000);                // počkat 2 ms
          } else {
                    digitalWrite(ledPin,LOW);   // vypnout LEDku
          }
}
```

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Teďka, pokud budeš mluvit nebo řvát do mikrofonu, LEDky připojené k pinu 13 na Arduino se rozsvítí.

Lekce 16 Senzor hladiny vody

Úvod

V této lekci budeme používat snímač hladiny vody na měření hloubky vody. Výsledek zobrazíme na LCD displeji.

Komponenty

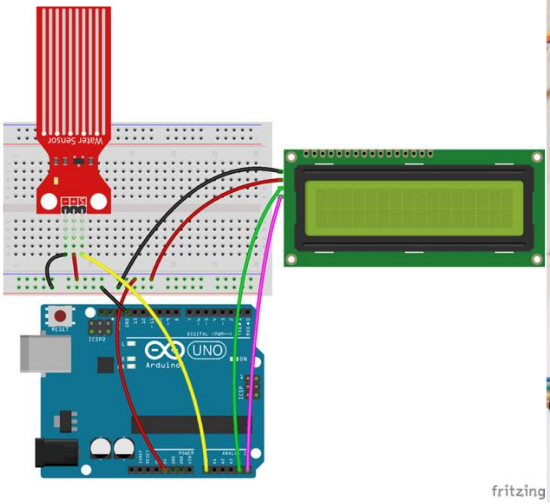
- Arduino
- Breadboard
- USB kabel
- Senzor hladiny vody
- 1x I2C LCD1602
- Breadboard vodiče

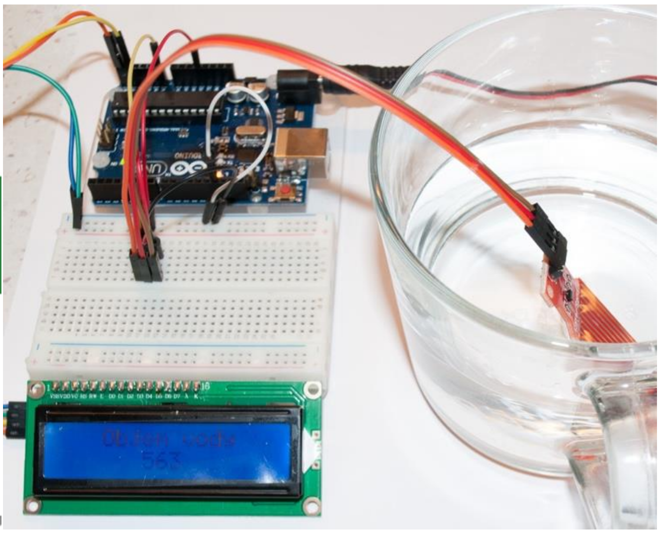
Princip

Senzor hladiny vody je modul, který může snímat hloubku vody, jejíž jádrová část je zesílený obvod, složený z tranzistoru a několika kabelů hřebenového tvaru DSP. Při umístění do vody budou tyto kabely měnit svůj odpor s hloubkou vody a převádět hloubkový signál na elektrický signál. S ADC převodníkem na Arduino konvertujeme tento analogový signál do digitálního a můžeme poznat změny hloubky vody.

Postup experimentu

Krok 1: Vytvoř obvod





Krok 2: Napiš kód do Arduino IDE

Kód

```
// Senzor hladiny vody
// Pokud ponoříš senzor do vody, uvidíš na displeji hodnotu, která bude závislá na hloubce ponoření.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

// připoj knihovny
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);               // nastavit adresu displeje na 0x27 pro 1602 displej
const int waterSensor = 0;
int waterValue = 0;

void setup() {
        lcd.init();                             // inicializace lcd
        lcd.backlight();                        // zapnout podsvícení
        lcd.setCursor(0,0);                     // nastavení kurzoru na sloupec 0, řádek 0
        lcd.print("   Objem vody   ");          // zobrazit "Objem vody" na displeji
}

void loop() {
        int waterValue = analogRead(waterSensor); // přečíst hodnotu pinu senzoru hladiny vody
        lcd.setCursor(6,1);                     // nastavení kurzoru na sloupec 6, řádek 1
        lcd.print(waterValue);                  // zobrazit hodnotu na lcd
        delay(300);
            // počkat 300 ms
        lcd.setCursor(0,1);                     // nastavení kurzoru na sloupec 0, řádek 1
        lcd.print("                ");          // zobrazit 16 mezer na lcd (vymazat celý řádek)
}
```

Poznámka: Sem potřebuješ přidat knihovnu. Mrkni se do popisu v lekci 6.

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Nyní, když se ponoří čidlo do vody, můžeš vidět hloubku zobrazenou (v neurčitých hodnotách) na LCD displeji.

Lekce 17 Teplotní čidlo LM-35

Úvod

LM35 je integrovaný obvod, převodník teploty na napětí, s maximální nelinearitou ±0,75 °C v rozsahu teplot -55 °C až 150 °C. Často používaná varianta obvodu je zapouzdřena v malém plastovém pouzdře TO-92, ale existuje i verze ve větším plastovém pouzdře TO-220 a verze v kovovém pouzdře TO-46. Převodník se vyrábí i v provedení SMD pro montáž přímo na povrch plošného spoje (SMT).
Integrovaný obvod LM35 vyvinula americká společnost National Semiconductor. Je určen pro měření teploty ve stupních Celsia. Obvod má malý odběr proudu; napájecí proud je menší než 60 µA. Vzhledem k nízkému příkonu převodníku dochází jen k malému vlastnímu ohřevu součástky, což je důležité z hlediska přesnosti měření. Výstupní impedance je typicky 0,1 Ω a nelinearita typicky ±0,25 °C.
Napájecí napětí převodníku – pokud se pohybuje v rozsahu 4 V až 20 V – nemusí být stabilizované a převodník lze tudíž napájet přímo z baterie. V základním zapojení pro měření teploty je převodník připojen ke zdroji napětí, což může být např. 9V baterie, a na výstupu převodníku je voltmetr.
Pokud se mají měřit kladné i záporné teploty, tak zapojení vychází o něco složitěji. LM35 se používá i jako převodník teplota/napětí pro Arduino.

Komponenty

- Arduino
- Breadboard
- USB kabel
- Teplotní čidlo LM-35
- I2C LCD1602
- Breadboard vodiče

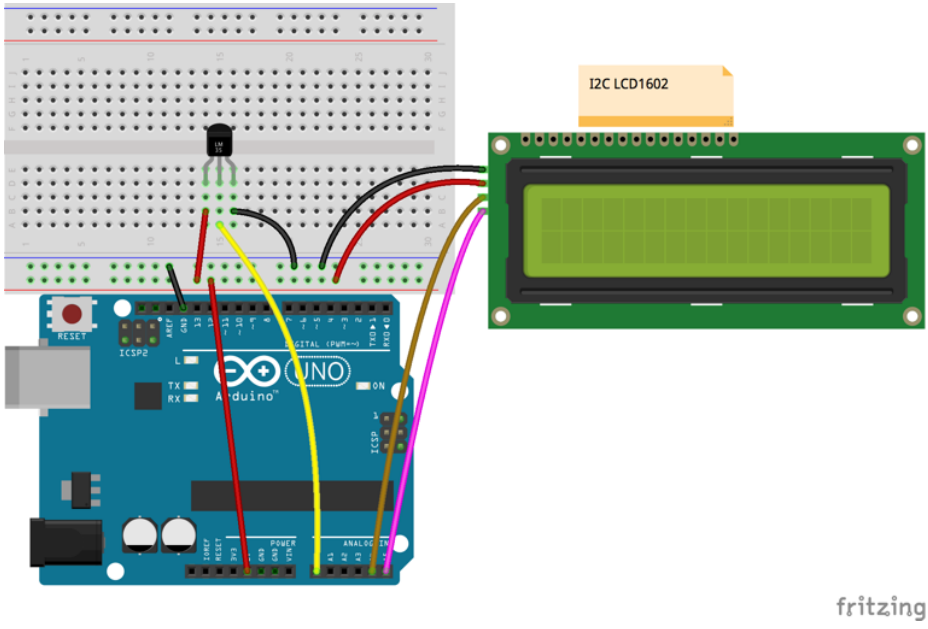
Princip

Protože převodní koeficient činí 10 mV/°C, objeví se např. při teplotě 150 °C na výstupu napětí převodníku napětí 1500 mV. Bude-li měřená teplota záporná, např. −40 °C, bude i výstupní napětí záporné (−400 mV).
Vzorec výpočtu je následující:

$$V_{out\_LM35}(T) = 10\,\mathrm{mV/^{\circ}C} \times T\,^{\circ}\mathrm{C}$$

Postup experimentu

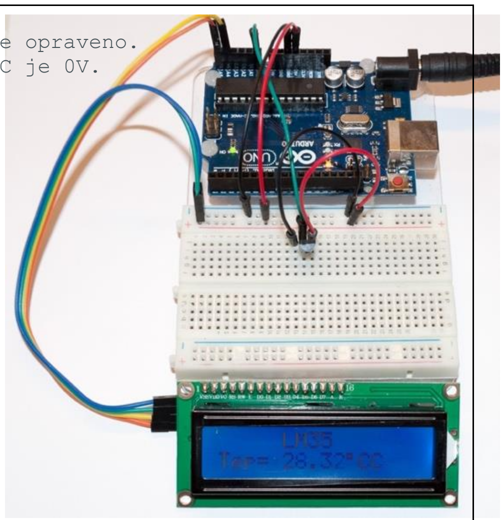
Krok 1: Vytvoř obvod



Krok 2: Napiš kód do Arduino IDE

Kód

```
// Teplotní čidlo LM-35
// Čidlo zatím funguje nestabilně, teplota skáče. V příští verzi bude opraveno.
// Napěťový výstup čidla LM35 je lineárně závislý na teplotě. Při 0°C je 0V.
// Při zvětšení teploty na 1°C se napětí zvětší na 10 mV.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

#define lmPin A0                              // číslo pinu LM35
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);

float tem = 0;
long lmVal = 0;

void setup() {
         lcd.init();
         lcd.backlight();
}
void loop() {
         lmVal = analogRead(lmPin);   // přečíst hodnotu pinu lmPin
         tem = (lmVal * 0.0048828125 * 100); // Převedeme hodnotu do voltu (5/1024=0.0048828125) a pak
mV
                                      // mV Převedeme do °C (z datasheetu LM35 Vout = 10mV/°C × T)
         lcd.setCursor(5,0);          // nastavení kurzoru na sloupec 5, řádek 0
         lcd.print("LM35");           // zobrazit "LM35"
         lcd.setCursor(0,1);          // nastavení kurzoru na sloupec 0, řádek 1
         lcd.print("Teplota= ");      // zobrazit "Tepl= "
         lcd.setCursor(5,1);          // nastavení kurzoru na sloupec 5, řádek 1
         lcd.print(tem);              // zobrazit teplotu
         lcd.print(char(223));        // zobrazit znak "°"
         lcd.print("C");
         delay(300);                  // počkat 300 ms
}
```



Poznámka: Sem potřebuješ přidat knihovnu. Mrkni se do popisu v lekci 6.

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Teď uvidíš na displeji teplotu.

Lekce 18 Sedmisegmentový displej

Úvod

Sedmisegmentovka obsahuje sedm políček (segmentů), které je možné individuálně rozsvěcet a zhasínat. Jednotlivé segmenty mohou být zapnuty nebo vypnuty. Kombinací vypnutých a zapnutých segmentů můžeme docílit zobrazení arabských číslic, hexadecimálních číslic, případně i dalších písmen a znaků.

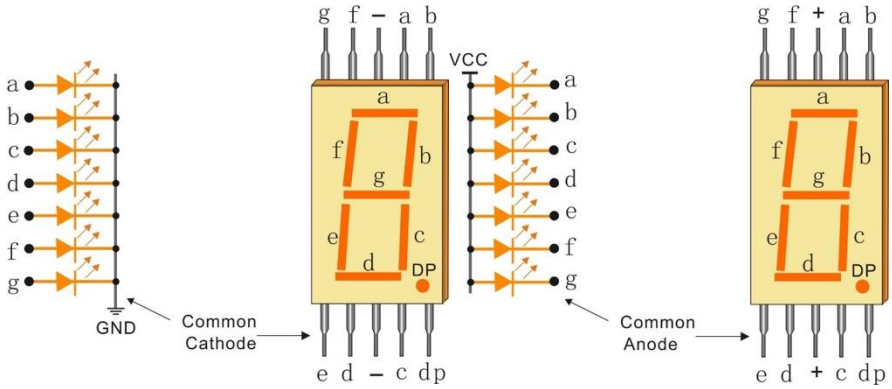
Komponenty

- Arduino
- Sedmisegmentový displej
- 8x Odpor (220Ω)
- USB kabel
- Breadboard vodiče
- Breadboard

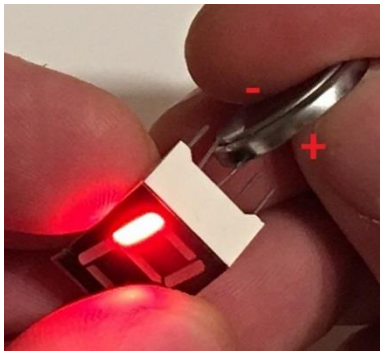
Princip

Sedmisegmentový displej je nejpoužívanější případ segmentového displeje. Je vhodný pouze pro zobrazování číslic, maximálně hexadecimálních číslic a až f. Pro číslicový indikátor je minimální počet segmentů právě sedm. Jako ostatní segmentové displeje

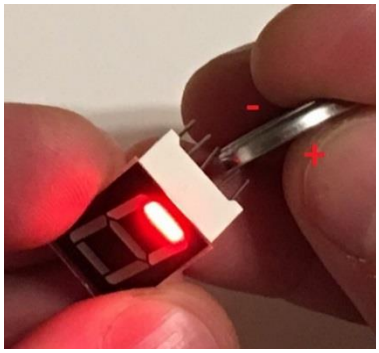
existují různé technologie zobrazení segmentů – LED, LCD, žárovková vlákna, dokonce i mechanické ovládání u největších displejů. Nejlevnější a nejpoužívanější z nich jsou sedmisegmentovky tvořené světelnými diodami. Ty se též vyrábějí sériově pro průmyslové použití, jsou ale i dostupné pro nadšence do elektroniky a dají se za sebou modulárně skládat do libovolně dlouhého displeje. Takový modul se familiérně nazývá sedmisegmentovka.
Diodové sedmisegmentovky mají relativně rychlou odezvu, přibližně 10 nanosekund, a spotřebu od 0,5 až 1 mA proudu na jeden segment u těch nejmenších (tzn. celá sedmisegmentovka 3,5–7 mA). Napětí anody je závislé na barvě – 1,5 až 2,5 V. Aby se ovládání diod zjednodušilo, mají diody navzájem propojené anody či katody.
Společná katoda (CC) a společná anoda (CA).
Rozdíl mezi těmito dvěma displeji, jak jejich název napovídá, spočívá v tom, že společná katoda má všechny katody 7-segmentů spojených dohromady a společná anoda má všechny anody 7-segmentů spojených dohromady.
Sedmisegmentový displej se společnou katodou
Společná katoda (Common Cathode, CC) – V displeji se společnou katodou jsou všechny katody LED segmentů spojeny dohromady na log. 0. Jednotlivé segmenty (a-g) jsou osvětleny tak, že anoda segmentu je zapojena na log. 1 přes omezovací odpor.



Sedmisegmentový displej se společnou anodou
Společná anoda (Common Anode, CA) – V displeji se společnou anodou jsou všechny anody LED segmentů spojeny dohromady na log. 1. Jednotlivé segmenty (a-g) jsou osvětleny tak, že katoda segmentu je zapojena na log. 0 přes omezovací odpor.

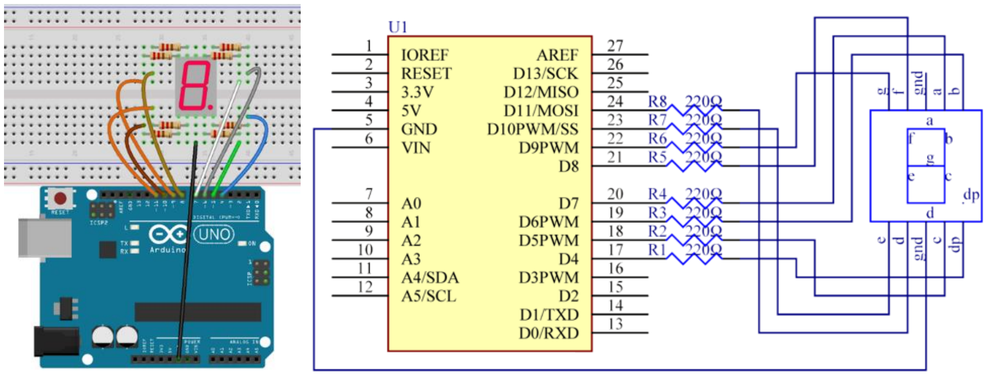


Ted je potřeba zjistit, jaký displej máš, CC nebo CA. Nejednoduší způsob je pří pomoci 3V baterie CR2032. Prostřední pin na displeji, je společná katoda nebo anoda. Zkus rychle se dotknout „+“ baterie do společného pinu a „-“ do pinu vedle. Svítí? Máš displej se společnou anodou. Nesvití? Skus otočit baterie a dotknout „-“ do společného pinu a „+“ do pinu vedle. Sviti? Máš displej se společnou katodou. Segmentem nesviť, ai vteřina trvalého svícení diodu spálí. 3V z baterie je příliš moc pro červenou LED, s napětím 1.9-2V.



Postup experimentu

Krok 1: Vytvoř obvod
Tohle je příklad zapojení pro CC displej. U CA displeje zapoj společný pin na +5V místo GND.

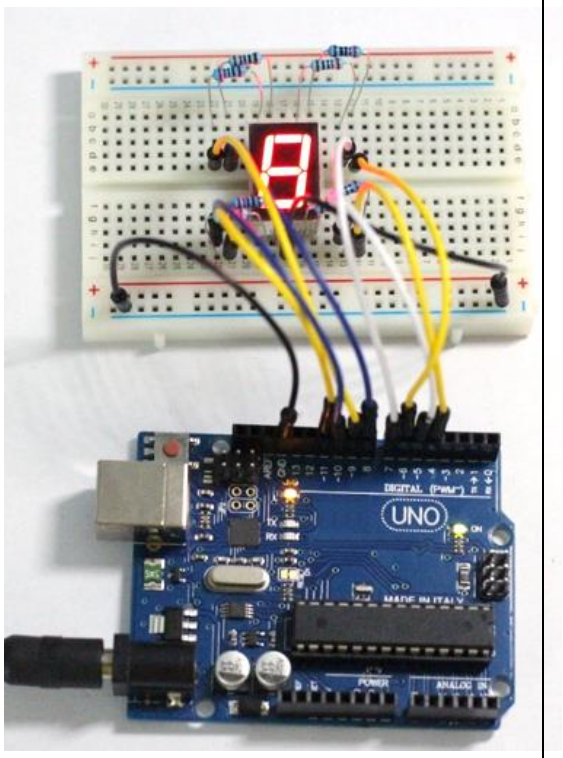


Krok 2: Napiš kód do Arduino IDE

Kód

```
// Sedmisegmentový displej
// Teď by jsi měl vidět cyklus sedmisegmentového displeje, od 1 po 0.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

//#define ON HIGH                       // Jestli máme CC (společná katoda) displej
#define ON LOW                          // Jestli máme CA (společná anoda) displej

const int a = 7;            // a číslo pinu segmentu "a"
const int b = 6;            // a číslo pinu segmentu "b"
const int c = 5;            // a číslo pinu segmentu "c"
const int d = 11;           // a číslo pinu segmentu "d"
const int e = 10;           // a číslo pinu segmentu "e"
const int f = 8;            // a číslo pinu segmentu "f"
const int g = 9;            // a číslo pinu segmentu "g"
const int dp =4;            // a číslo pinu segmentu "dp"

void setup() {
        // nastavení pinu každého segmentu jako výstupu
        for(int thisPin = 4; thisPin <= 11; thisPin++) {
                pinMode(thisPin,OUTPUT);
        }
}

void loop() {
        digital_1();        // zobrazit 1 na displeji
        delay(1000);        // počkat 1000 ms (1s)
        digital_2();        // zobrazit 2 na displeji
        delay(1000);        // počkat 1000 ms (1s)
        digital_3();        // zobrazit 3 na displeji
        delay(1000);        // počkat 1000 ms (1s)
        digital_4();        // zobrazit 4 na displeji
        delay(1000);        // počkat 1000 ms (1s)
        digital_5();        // zobrazit 5 na displeji
        delay(1000);        // počkat 1000 ms (1s)
        digital_6();        // zobrazit 6 na displeji
        delay(1000);        // počkat 1000 ms (1s)
        digital_7();        // zobrazit 7 na displeji
        delay(1000);        // počkat 1000 ms (1s)
        digital_8();        // zobrazit 8 na displeji
        delay(1000);        // počkat 1000 ms (1s)
        digital_9();        // zobrazit 9 na displeji
        delay(1000);        // počkat 1000 ms (1s)
        digital_0();        // zobrazit 0 na displeji
        delay(1000);        // počkat 1000 ms (1s)
}

void digital_1() {                      // zobrazit 1 na displeji
        // vypnout všechny segmenty
        clear();
        digitalWrite(c, ON);            // zapnout segment "c"
        digitalWrite(b, ON);            // zapnout segment "b"
}
void digital_2() {                 // zobrazit 2 na displeji
        // vypnout všechny segmenty
```

```
          clear();
          digitalWrite(a, ON);          // zapnout segment "a"
          digitalWrite(b, ON);          // zapnout segment "b"
          digitalWrite(g, ON);          // zapnout segment "g"
          digitalWrite(e, ON);          // zapnout segment "e"
          digitalWrite(d, ON);          // zapnout segment "d"
}
void digital_3() {                      // zobrazit 3 na displeji
          clear();
          digitalWrite(a, ON);
          digitalWrite(b, ON);
          digitalWrite(g, ON);
          digitalWrite(c, ON);
          digitalWrite(d, ON);
}
void digital_4() {                      // zobrazit 4 na displeji
          clear();
          digitalWrite(f, ON);
          digitalWrite(g, ON);
          digitalWrite(b, ON);
          digitalWrite(c, ON);
}
void digital_5() {                      // zobrazit 5 na displeji
          clear();
          digitalWrite(f, ON);
          digitalWrite(g, ON);
          digitalWrite(c, ON);
          digitalWrite(d, ON);
          digitalWrite(a, ON);
}
void digital_6() {                      // zobrazit 6 na displeji
          clear();
          digitalWrite(a, ON);
          digitalWrite(f, ON);
          digitalWrite(e, ON);
          digitalWrite(d, ON);
          digitalWrite(c, ON);
          digitalWrite(g, ON);
}
void digital_7() {                      // zobrazit 7 na displeji
          clear();
          digitalWrite(a, ON);
          digitalWrite(b, ON);
          digitalWrite(c, ON);
}
void digital_8() {                      // zobrazit 8 na displeji
          clear();
          digitalWrite(a, ON);
          digitalWrite(f, ON);
          digitalWrite(e, ON);
          digitalWrite(d, ON);
          digitalWrite(c, ON);
          digitalWrite(g, ON);
          digitalWrite(b, ON);
}
void digital_9() {                      // zobrazit 9 na displeji
          clear();
          digitalWrite(a, ON);
          digitalWrite(f, ON);
          digitalWrite(g, ON);
          digitalWrite(b, ON);
          digitalWrite(c, ON);
}
void digital_0() {                      // zobrazit 0 na displeji
          clear();
          digitalWrite(a, ON);
          digitalWrite(f, ON);
          digitalWrite(e, ON);
          digitalWrite(d, ON);
          digitalWrite(c, ON);
          digitalWrite(b, ON);
}
void clear () {                                 // vypnout všechny segmenty
          for(int thisPin = 4; thisPin <= 11; thisPin++) {
                    digitalWrite(thisPin, !ON);
```



```
            }
}
```

Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Nyní bys měl vidět na displeji blikat znaky od 0 do F, tam a zpět.

Lekce 19 Stopky - 4-místný sedmisegmentový displej

Úvod

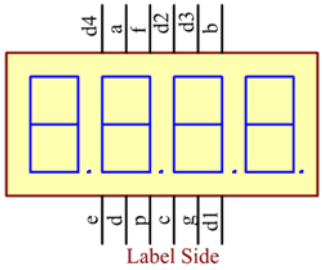
V této lekci použijeme 4-místný sedmisegmentový displej pro vytvoření stopek.

Komponenty

- Arduino
- USB kabel
- 4-místný sedmisegmentový displej
- Breadboard vodiče
- Breadboard
- 8x Odpor (220Ω)

Princip

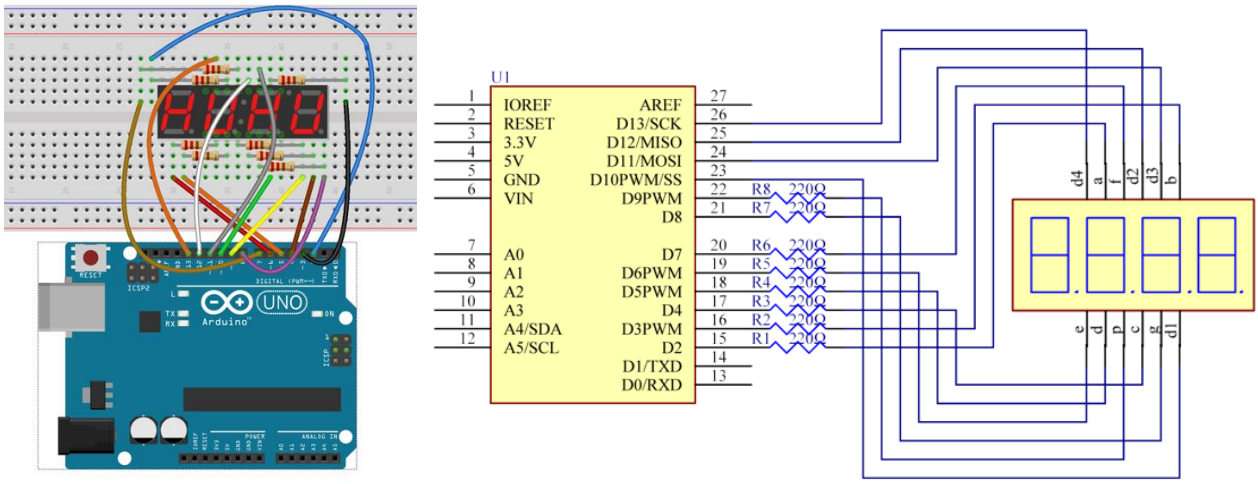
Pokud je použit 7- segmentový LED displej a jestli se jedná o displej se společnou anodou, připoj anodový pin do zdroje. Pokud se jedná o displej se společnou katodou, připoj katodu do GND. Pokud se použije 4-místný 7-segmentový LED displej, pin "společná anoda nebo katoda" se používá pro kontrolu zobrazené číslice. Může být tedy zobrazena jenom jedna číslice. Nicméně, protože je to založeno na efektu Persistence of Vision, můžeme vidět čtyři 7-segmentové displeje a všechny budou zobrazovat nějaká čísla. Důvodem je, že elektronická rychlost skenování je příliš vysoká a my tedy nestihneme zaznamenat blikání.



Postup experimentu

Ted je potřeba zjistit, jaký displej máš, CC nebo CA. Postup je stejný jako v lekce 18.
Krok 1: Vytvoř obvod
Zapojení mezi 4x7 segment LED displej a Arduino viz níže:

4-místný sedmisegmentový displej	Arduino
a	2
b	3
c	4
d	5
e	6
f	7
g	8
p	9
D1	10
D2	11
D3	12
D4	13



Krok 2: Napiš kód do Arduino IDE

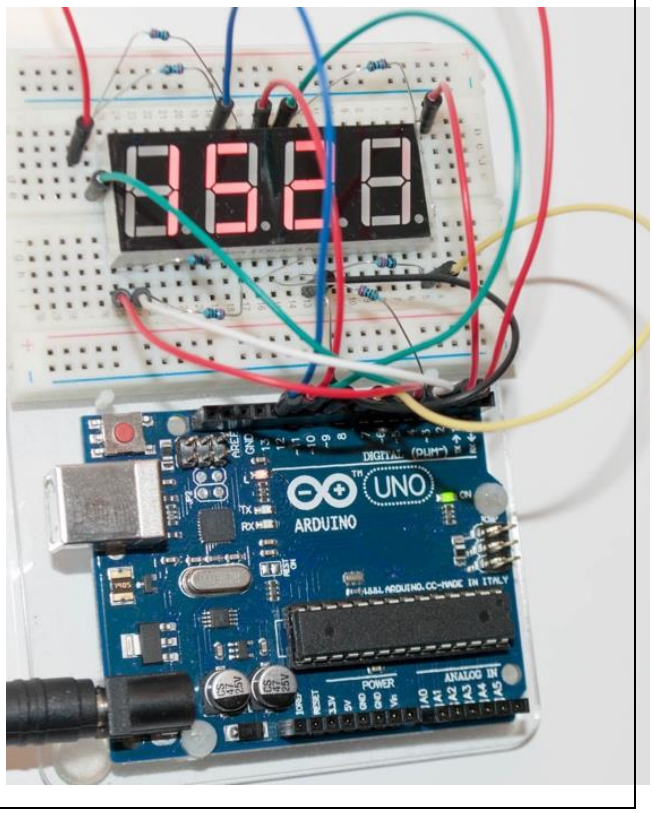
Kód

```
// Stopky - 4-místný sedmisegmentový displej
// Na displeji uvidíš číslo, které se zvyšuje o jednu každou vteřinu.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/
#include <TimerOne.h>

//#define CC                            // Jestli máme CC (společná katoda) displej
#define CA                              // Jestli máme CA (společná anoda) displej


#ifdef CA
          #define ON_NUMBER LOW         //vymení v kódu před kompilací ON_NUMBER za LOW
          #define OFF_NUMBER HIGH
          #define ON_DIGIT HIGH
          #define OFF_DIGIT LOW
#else
          #define ON_NUMBER HIGH
          #define OFF_NUMBER LOW
          #define ON_DIGIT LOW
          #define OFF_DIGIT HIGH
#endif

// Připojení displeje k pinům Arduino
const int a = 2;
const int b = 3;
const int c = 4;
const int d = 5;
const int e = 6;
const int f = 7;
const int g = 8;
const int p = 9;
const int d1 = 10;
const int d2 = 11;
const int d3 = 12;
const int d4 = 13;
// n představuje hodnotu zobrazenou na displeji.
// například při zobrazení n = 0 se zobrazí 0000. Maximální hodnota je 9999.
long n = 0;
int x = 100;
// del je pauza mezi spínanými číslovkami, 5ms je dobrá hodnota pro hodiny.
int del = 5;
// nastavit count = 0. Zde je count hodnota počítání, která se zvyšuje o 1 každých 0,1 sekundy,
// znamená, že 1 sekunda se počítá, když je hodnota 10
int count = 0;

void setup() {
          // nastavení pinů displeje jako výstupu
          pinMode(d1, OUTPUT);
          pinMode(d2, OUTPUT);
          pinMode(d3, OUTPUT);
```

```
          pinMode(d4, OUTPUT);
          pinMode(a, OUTPUT);
          pinMode(b, OUTPUT);
          pinMode(c, OUTPUT);
          pinMode(d, OUTPUT);
          pinMode(e, OUTPUT);
          pinMode(f, OUTPUT);
          pinMode(g, OUTPUT);
          pinMode(p, OUTPUT);

          // nastavime časovač o délce 100000 mikrosekund
          // (nebo 0,1s - nebo 10Hz => led 5x bliká,
          // 5 cyklů zapnutí a vypnutí, za sekundu)
          Timer1.initialize(100000);
          Timer1.attachInterrupt( add ); // připojme servisní rutinu
}

void loop() {
          clearLEDs();                              // vypnout všechny segmenty
          pickDigit(0);                             // rozsvítit displej d1
          pickNumber(n%10);  // získat a ukázat hodnotu tisíce
          delay(del);                                          // pauza 5ms

          clearLEDs();
          pickDigit(1);                             // rozsvítit displej d2
          pickNumber(n%100/10);// získat a ukázat hodnotu sto
          delay(del);

          clearLEDs();
          pickDigit(2);                             // rozsvítit displej d3
          pickNumber((n%1000)/100);    // získat a ukázat hodnotu deset
          delay(del);

          clearLEDs();
          pickDigit(3);                           // rozsvítit displej d4
          pickNumber((n/1000));                 // získat a ukázat hodnotu do 9
          delay(del);
}

void pickDigit(int x) {       // vybrat displej
          // Tento 7 segmentový displej je CA (se společnou anodou) nebo CC (se společnou katodou)
          // Takže také pomocí digitalWrite nastavíme všechny anody na LOW (jestli displej je CA )
          // nebo na HIGH (jestli displej je CC) a celý displej vypneme
          digitalWrite(d1, OFF_DIGIT);
          digitalWrite(d2, OFF_DIGIT);
          digitalWrite(d3, OFF_DIGIT);
          digitalWrite(d4, OFF_DIGIT);

          switch(x){
                    case 0:
                              digitalWrite(d1, ON_DIGIT); // zapnout displej d1
                              break;
                    case 1:
                              digitalWrite(d2, ON_DIGIT); // zapnout displej d2
                              break;
                    case 2:
                              digitalWrite(d3, ON_DIGIT); // zapnout displej d3
                              break;
                    default:
                              digitalWrite(d4, ON_DIGIT); // zapnout displej d4
                              break;
          }
}
// Funkce je pro ovládání 7 segmentového displeje pro zobrazení čísel
// Zde "x" je číslo, které chceš zobrazit. Je to celé číslo od 0 do 9
void pickNumber(int x) {
          switch(x) {
                    default:
                              zero();
                              break;
                    case 1:
                              one();
                              break;
                    case 2:
                              two();
```

```
                        break;
                case 3:
                        three();
                        break;
                case 4:
                        four();
                        break;
                case 5:
                        five();
                        break;
                case 6:
                        six();
                        break;
                case 7:
                        seven();
                        break;
                case 8:
                        eight();
                        break;
                case 9:
                        nine();
                        break;
        }
}
// vypnout všechny segmenty
void clearLEDs() {
        digitalWrite(a, OFF_NUMBER);
        digitalWrite(b, OFF_NUMBER);
        digitalWrite(c, OFF_NUMBER);
        digitalWrite(d, OFF_NUMBER);
        digitalWrite(e, OFF_NUMBER);
        digitalWrite(f, OFF_NUMBER);
        digitalWrite(g, OFF_NUMBER);
        digitalWrite(p, OFF_NUMBER);
}
// zobrazit 0 na displeji
void zero() {
        digitalWrite(a, ON_NUMBER);
        digitalWrite(b, ON_NUMBER);
        digitalWrite(c, ON_NUMBER);
        digitalWrite(d, ON_NUMBER);
        digitalWrite(e, ON_NUMBER);
        digitalWrite(f, ON_NUMBER);
        digitalWrite(g, OFF_NUMBER);
}
// zobrazit 1 na displeji
void one() {
        digitalWrite(a, OFF_NUMBER);
        digitalWrite(b, ON_NUMBER);
        digitalWrite(c, ON_NUMBER);
        digitalWrite(d, OFF_NUMBER);
        digitalWrite(e, OFF_NUMBER);
        digitalWrite(f, OFF_NUMBER);
        digitalWrite(g, OFF_NUMBER);
}
// zobrazit 2 na displeji
void two() {
        digitalWrite(a, ON_NUMBER);
        digitalWrite(b, ON_NUMBER);
        digitalWrite(c, OFF_NUMBER);
        digitalWrite(d, ON_NUMBER);
        digitalWrite(e, ON_NUMBER);
        digitalWrite(f, OFF_NUMBER);
        digitalWrite(g, ON_NUMBER);
}
// zobrazit 3 na displeji
void three() {
        digitalWrite(a, ON_NUMBER);
        digitalWrite(b, ON_NUMBER);
        digitalWrite(c, ON_NUMBER);
        digitalWrite(d, ON_NUMBER);
        digitalWrite(e, OFF_NUMBER);
        digitalWrite(f, OFF_NUMBER);
        digitalWrite(g, ON_NUMBER);
}
```

```
// zobrazit 4 na displeji
void four() {
          digitalWrite(a, OFF_NUMBER);
          digitalWrite(b, ON_NUMBER);
          digitalWrite(c, ON_NUMBER);
          digitalWrite(d, OFF_NUMBER);
          digitalWrite(e, OFF_NUMBER);
          digitalWrite(f, ON_NUMBER);
          digitalWrite(g, ON_NUMBER);
}
// zobrazit 5 na displeji
void five() {
          digitalWrite(a, ON_NUMBER);
          digitalWrite(b, OFF_NUMBER);
          digitalWrite(c, ON_NUMBER);
          digitalWrite(d, ON_NUMBER);
          digitalWrite(e, OFF_NUMBER);
          digitalWrite(f, ON_NUMBER);
          digitalWrite(g, ON_NUMBER);
}
// zobrazit 6 na displeji
void six() {
          digitalWrite(a, ON_NUMBER);
          digitalWrite(b, OFF_NUMBER);
          digitalWrite(c, ON_NUMBER);
          digitalWrite(d, ON_NUMBER);
          digitalWrite(e, ON_NUMBER);
          digitalWrite(f, ON_NUMBER);
          digitalWrite(g, ON_NUMBER);
}
// zobrazit 7 na displeji
void seven() {
          digitalWrite(a, ON_NUMBER);
          digitalWrite(b, ON_NUMBER);
          digitalWrite(c, ON_NUMBER);
          digitalWrite(d, OFF_NUMBER);
          digitalWrite(e, OFF_NUMBER);
          digitalWrite(f, OFF_NUMBER);
          digitalWrite(g, OFF_NUMBER);
}
// zobrazit 8 na displeji
void eight() {
          digitalWrite(a, ON_NUMBER);
          digitalWrite(b, ON_NUMBER);
          digitalWrite(c, ON_NUMBER);
          digitalWrite(d, ON_NUMBER);
          digitalWrite(e, ON_NUMBER);
          digitalWrite(f, ON_NUMBER);
          digitalWrite(g, ON_NUMBER);
}
// zobrazit 9 na displeji
void nine() {
          digitalWrite(a, ON_NUMBER);
          digitalWrite(b, ON_NUMBER);
          digitalWrite(c, ON_NUMBER);
          digitalWrite(d, ON_NUMBER);
          digitalWrite(e, OFF_NUMBER);
          digitalWrite(f, ON_NUMBER);
          digitalWrite(g, ON_NUMBER);
}

void add() {
          // přepnout LED
          count ++;
          if(count == 10) {
                    count = 0;
                    n++;
                    if(n == 10000) {
                              n = 0;
                    }
          }
}
```



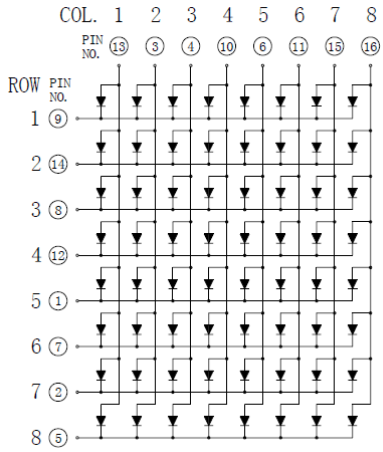
Poznámka: Sem potřebuješ přidat knihovnu. Mrkni se do popisu v lekci 6.

Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Nyní můžeš vidět číslo, které se zvyšuje o" jednu" za sekundu na displeji.

Lekce 20 8x8 LED matice

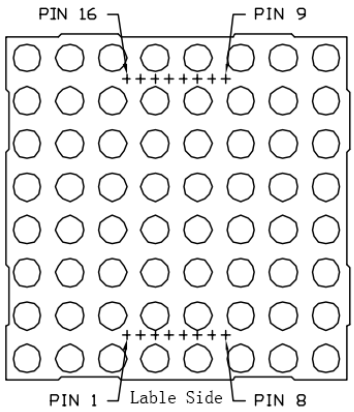
Úvod

Díky low-voltage scanning metodě mají maticové LED displeje své výhody, kterými jsou: úspora energie, dlouhá životnost, nízké náklady, vysoký jas, dlouhý vizuální dosah, nepromokavost, aj. Maticové LED displeje mohou uspokojit potřeby různých aplikací. I díky tomu mají širokou perspektivu rozvoje.
Provedeme experiment s LED maticí. Budeš si tedy moci vyzkoušet její kouzlo na vlastní kůži...



Komponenty

- Arduino
- USB kabel
- LED matice 8x8
- 8x Odpor (220Ω)
- Breadboard
- Breadboard vodiče



Princip

Vnější pohled na maticový LED displej:

Definice pinů:
Nejprve definuj číslování řádků a sloupců (pouze pro maticový LED displej, jehož číslo modelu končí BS).
Pin číslování odpovídá výše uvedenému řádku a sloupci:

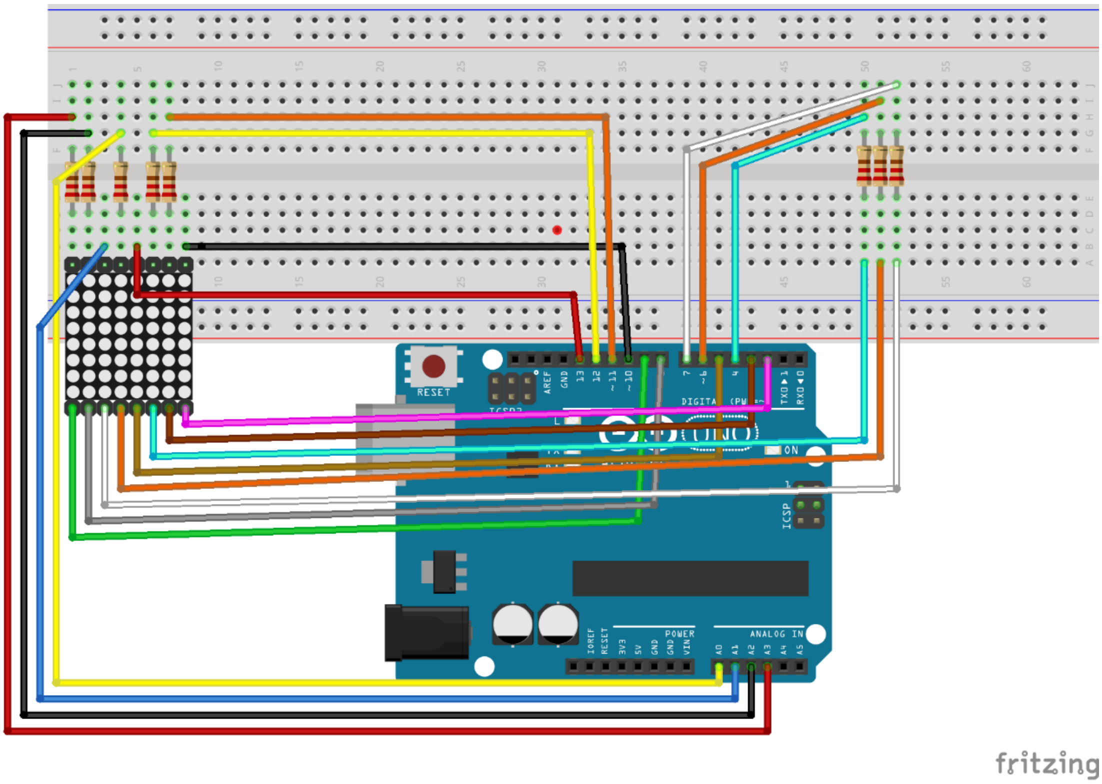
Sloupec (COL)	1	2	3	4	5	6	7	8
č. pinu matice	13	3	4	10	6	11	15	16
Řádek (ROW)	1	2	3	4	5	6	7	8
č. pinu matice	9	14	8	12	1	7	2	5

Princip 8x8 maticového§ LED displeje:
8x8 LED matice se skládá z 64 LED a každá LEDka je umístěna na průsečíku řádku a sloupce. Když je elektrická úroveň určité řady vysoká a elektrická úroveň určitého sloupce nízká, pak se odpovídající LED rozsvítí. Chceš-li rozsvítit LED na první pokus, měl bys nastavit řádek 1 na vysokou úroveň (log. 1) a sloupec 1 na nízkou úroveň (log. 0). Pak se LED na prvním bodě rozsvítí. Chceš-li rozsvítit světlo LED na prvním řádku, měl bys nastavit řádek 1 na log. 1 a sloupce (1, 2, 3, 4, 5, 6, 7, 8) na log. 0. Pak se všechny LED diody na prvním řádku rozsvítí.

Základním principem při použití LED zobrazovačů je tzv. multiplikované vysvěcování, jehož podstata spočívá v zapojení LED do matic. Tato matice je tvořena anodovou a katodovou skupinou. Data tvořící snímek jsou postupně přiváděna na anody, kdy vždy jedna z katod je sepnuta. Postupným vysvícením všech dat na všech katodách dojde k vysvícení tzv. snímku. Princip je obdobný s funkcí obrazovky televize nebo monitoru. Snímková frekvence, tedy počet zobrazených snímků za sekundu, musí být tak vysoká, aby nepůsobila rušivě na lidské oko. V praxi se používají frekvence od stovek Hz do jednotek nebo desítek kHz. Neustálé vysvěcování snímků je poměrně náročná záležitost, jejíž složitost stoupá s počtem prvků matice. Pro matici 8x8 bodů je zapotřebí 8 bitů pro řízení anodové a dalších 8 pro řízení katodové skupiny. Tímto je zařízeno ovládání pouhých 64 bodů. Přirozeným řešením tohoto problému je využití jednočipového mikropočítače (například Arduino) s dostatečnou kapacitou vstupních a výstupních pinů a výpočetním výkonem.

Postup experimentu

Krok 1: Vytvoř obvod

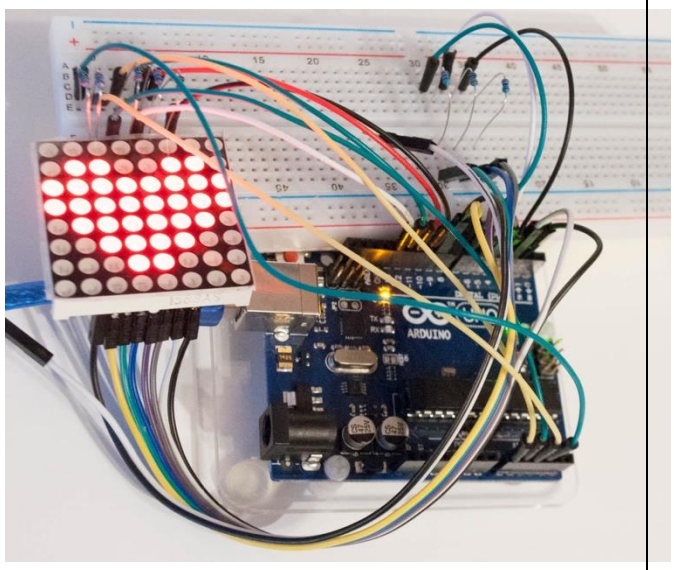


Krok 2: Napiš kód do Arduino IDE

Kód

```
// 8x8 LED matice
// Každý znak "LASK♥RDUINO" postupně blikne na displeji, jeden za druhým.
// Generuj svůj symbol zde http://embed.plnkr.co/3VUsekP3jC5xwSIQDVHx/preview
// Písma pro 8x8 maticový displej zde https://github.com/dhepper/font8x8
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

#define ROW_1 10     // pin matice 9
#define ROW_2 A1     // pin 14
#define ROW_3 2      // pin 8
#define ROW_4 13     // pin 12
#define ROW_5 9      // pin 1
#define ROW_6 3      // pin 7
#define ROW_7 8      // pin 2
#define ROW_8 5      // pin 5

#define COL_1 A0     // pin 13
#define COL_2 7      // pin 3
#define COL_3 6      // pin 4
#define COL_4 11     // pin 10
#define COL_5 4      // pin 6
#define COL_6 12     // pin 11
#define COL_7 A2     // pin 15
#define COL_8 A3     // pin 16

const byte rows[] = {
    ROW_1, ROW_2, ROW_3, ROW_4, ROW_5, ROW_6, ROW_7, ROW_8
};

// kódy káždého znaku, zobrazeného na displeji
// jako Hexadecimal
byte l[] = { 0x0F, 0x06, 0x06, 0x06, 0x46, 0x66, 0x7F, 0x00};
byte a[] = { 0x0C, 0x1E, 0x33, 0x33, 0x3F, 0x33, 0x33, 0x00};
byte s[] = { 0x1E, 0x33, 0x07, 0x0E, 0x38, 0x33, 0x1E, 0x00};
byte k[] = { 0x67, 0x66, 0x36, 0x1E, 0x36, 0x66, 0x67, 0x00};
byte r[] = { 0x3F, 0x66, 0x66, 0x3E, 0x36, 0x66, 0x67, 0x00};
byte d[] = { 0x1F, 0x36, 0x66, 0x66, 0x66, 0x36, 0x1F, 0x00};
byte u[] = { 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x3F, 0x00};
byte i[] = { 0x1E, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x1E, 0x00};
```

```
byte n[] = { 0x63, 0x67, 0x6F, 0x7B, 0x73, 0x63, 0x63, 0x00};
byte o[] = { 0x1C, 0x36, 0x63, 0x63, 0x63, 0x36, 0x1C, 0x00};
// jako binární
byte heart[] = {
B00000000,
B01100110,
B11111111,
B11111111,
B01111110,
B00111100,
B00011000,
B00000000};

float timeCount = 0;

void setup() {
          // nastavení pinů displeje jako výstupu
          for (byte i = 2; i <= 13; i++)
                    pinMode(i, OUTPUT);

          pinMode(A0, OUTPUT);
          pinMode(A1, OUTPUT);
          pinMode(A2, OUTPUT);
          pinMode(A3, OUTPUT);
}

void loop() {
          // Pauzu můžeš odstranit.
          // tím pádem bude displej jasnější.
          delay(2);
          timeCount += 1;
          if(timeCount <  100) {
                    drawScreen(l);
          } else if (timeCount <   210) {
                    // nic neděláme -> pauza mezi znaky
          } else if (timeCount <   300) {
                    drawScreen(a);
          } else if (timeCount <   410) {
                    // nic neděláme -> pauza mezi znaky
          } else if (timeCount <   500) {
                    drawScreen(s);
          } else if (timeCount <   610) {
                    // nic neděláme -> pauza mezi znaky
          } else if (timeCount <   700) {
                    drawScreen(k);
          } else if (timeCount <   810) {
                    // nic neděláme -> pauza mezi znaky
          } else if (timeCount <   900) {
                    drawScreen(heart);
          } else if (timeCount <  1010) {
                    // nic neděláme -> pauza mezi znaky
          } else if (timeCount <  1100) {
                    drawScreen(d);
          } else if (timeCount <  1210) {
                    // nic neděláme -> pauza mezi znaky
          } else if (timeCount <  1300) {
                    drawScreen(u);
          } else if (timeCount <  1410) {
                    // nic neděláme -> pauza mezi znaky
          } else if (timeCount <  1500) {
                    drawScreen(i);
          } else if (timeCount <  1610) {
                    // nic neděláme -> pauza mezi znaky
          } else if (timeCount <  1700) {
          drawScreen(n);
          } else if (timeCount <  1810) {
                    // nic neděláme -> pauza mezi znaky
          } else if (timeCount <  1900) {
                    drawScreen(o);
          } else if (timeCount <  2010) {
                    // nic neděláme -> pauza mezi znaky
          } else {
                    // a zase od začátku...
                    timeCount = 0;
          }
```



```
}
 void  drawScreen(byte buffer2[]){


   // zapínáme každý řádek v sérii
    for (byte i = 0; i < 8; i++) {
        setColumns(buffer2[i]); // nastav sloupce pro tento konkrétní řádek

        digitalWrite(rows[i], LOW);
        delay(2); // nastav na 50 až 100, chceš-li vidět multiplexní efekt!!!
        digitalWrite(rows[i], HIGH);

    }
}
void setColumns(byte b) {
    digitalWrite(COL_1, (b >> 0) & 0x01); // získat 1 bit: 10000000
    digitalWrite(COL_2, (b >> 1) & 0x01); // získat 2 bit: 01000000
    digitalWrite(COL_3, (b >> 2) & 0x01); // získat 3 bit: 00100000
    digitalWrite(COL_4, (b >> 3) & 0x01); // získat 4 bit: 00010000
    digitalWrite(COL_5, (b >> 4) & 0x01); // získat 5 bit: 00001000
    digitalWrite(COL_6, (b >> 5) & 0x01); // získat 6 bit: 00000100
    digitalWrite(COL_7, (b >> 6) & 0x01); // získat 7 bit: 00000010
    digitalWrite(COL_8, (b >> 7) & 0x01); // získat 8 bit: 00000001
}
```

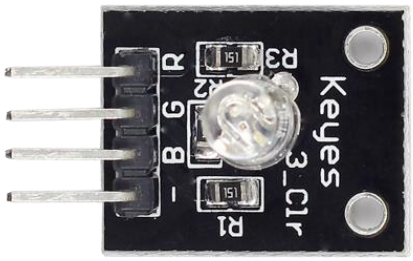
Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Zde bys měl vidět obrázek s písmenem“A”.

Lekce 21 RGB LED

Úvod

RGB LED mohou vyzařovat různé barvy světla. Tři LED: červená, zelená a modrá jsou zabaleny do průhledného nebo poloprůhledného plastového pláště se čtyřmi kolíky. Ze třech základních barev: červená, zelená a modrá lze míchat a vytvářet všechny druhy barev podle jasu, takže si můžeš udělat RGB LED vyzařující barevné světlo.

Komponenty



- Arduino (or ArduinoMega2560 board)
- USB kabel
- RGB LED modul
- Breadboard vodiče

Princip

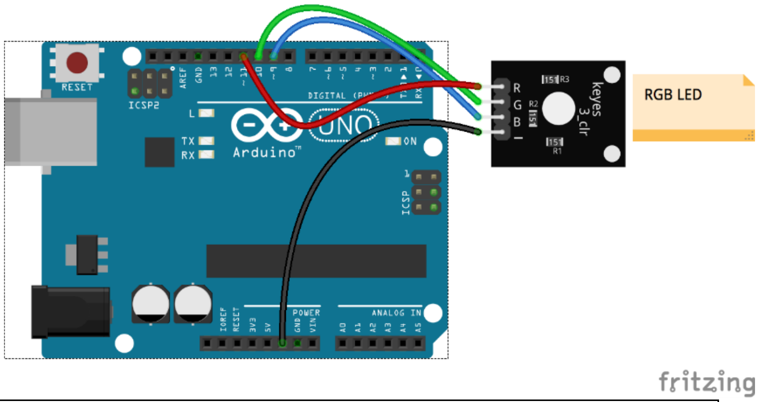
V tomto experimentu budeme používat technologii PWM pro ovládání jasu RGB LED. Už jsme to popsali v lekci 2 tohoto kitu. Pokud je to pro Tebe nezbytné, zkontroluj si to.

Každá ze tří barevných kanálů: červená, zelená a modrá, má 255 stupňů jasu. Pokud ze tří základních barev jsou všechny 0, LED zhasne. Pokud jsou všechny 255, LED bude nejjasnější. Pokud do všech kolíků pustíme hodnoty mezi 0 a 255, RGB LED bude zobrazovat různé barvy.

RGB LED lze rozdělit na následující typy: společná anoda a společná katoda. V tomto experimentu budeme používat společnou katodu RGB LED.

Postup experimentu

Krok 1: Vytvoř obvod



RGB LED module	Arduino
R	11
G	10
B	9
-	GND

Krok 2: Napiš kód do Arduino IDE

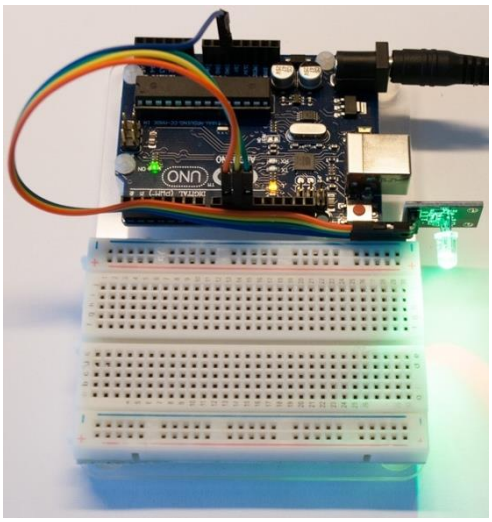
Kód

```
// RGB LED
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

const int redPin   = 11;     // číslo pinu červené barvy
const int greenPin = 10;     // číslo pinu zelené barvy
const int bluePin  = 9;               // číslo pinu modré barvy

void setup() {
          pinMode(redPin, OUTPUT);     // nastavení pinu redPin jako výstupu
          pinMode(greenPin, OUTPUT);   // nastavení pinu greenPin jako výstupu
          pinMode(bluePin, OUTPUT);    // nastavení pinu bluePin jako výstupu
}

void loop() {
          // Zakladní barvy:
          color(255, 0, 0); // zapnout červenou
          delay(1000);      // počkat 1s
          color(0,255, 0);  // zapnout zelenou
          delay(1000);      // počkat 1s
          color(0, 0, 255); // zapnout modrou
          delay(1000);      // počkat 1s
          // Michaný barvy:
          color(255,0,0);   // červená
          delay(1000);
          color(237,109,0); // oranžová
          delay(1000);
          color(255,215,0); // žlutá
          delay(1000);
          color(0,255,0);    // zelená
          delay(1000);
          color(0,0,255);    // modrá
          delay(1000);
          color(0,46,90);    // indigo
          delay(1000);
          color(128,0,128); // purpurová
          delay(1000);
}
// generace barvy
void color (unsigned char red, unsigned char green, unsigned char blue)
{
          analogWrite(redPin, red);
          analogWrite(bluePin, blue);
          analogWrite(greenPin, green);
}
```

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Nyní můžeš vidět, že RGB LED bliká červeně, zeleně, modře, oranžově, žlutě, fialově, indigo atd.



Lekce 22 Řízení sedmisegmentového displeje s 74HC595

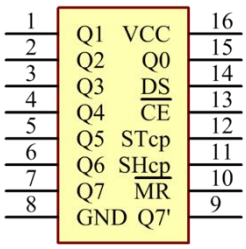
Úvod

V minulé lekci jsme si ukázali, jak řídit sedmisegmentové displeje s pomocí Arduino, bez speciálních obvodů. Další možností, jak řídit několik sedmisegmentových displejů pomocí Arduino, je použití posuvných registrů, které pracují pomocí sériového přenosu dat. Pro řízení použijeme osmibitové posuvné registry 74HCT595.

Komponenty

- Arduino
- 8x Odpor (220Ω)
- Posuvný registr 74HC595
- Breadboard vodiče
- Breadboard
- USB kabel
- Sedmisegmentový LED displej

Princip



Co vůbec posuvný registr dělá? V zásadě je to soustava klopných obvodů, kterými se logická informace posouvá dále pomocí hodinového impulsu. K vysvětlení má posuvný regist celkem pro nás potřebné 3 vstupy (SER, SRCLK, SCLK) a celkem v našem případě 8 výstupů (Q0 – Q7). K ovládání 8 výstupů jsou zapotřebí pouze 3 piny na Arduino. Zapojíme tedy 74HC595 k Arduino a sedmisegmentovku k 74HC595.

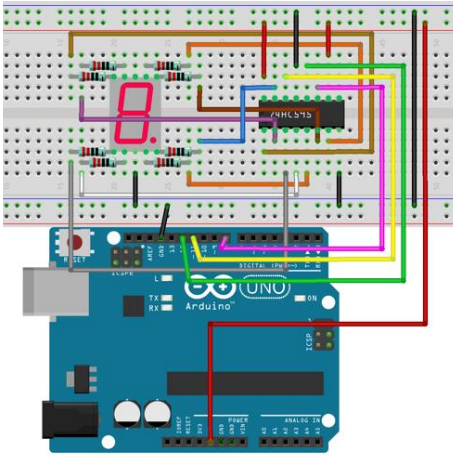
74HC595 se skládá ze tří částí: posuvný registr, záchytný registr a třístavové výstupy. Převádí sériový vstup do paralelního výstupu, takže můžeš ušetřit IO porty Arduino. 74HC595 je široce používán k řízení např. segmentových displejů. Můžeme nastavit buď výstupní piny log.1, log.0 nebo stav vysoké impedance. Je možné zapojovat do kaskády několik 74HC595.

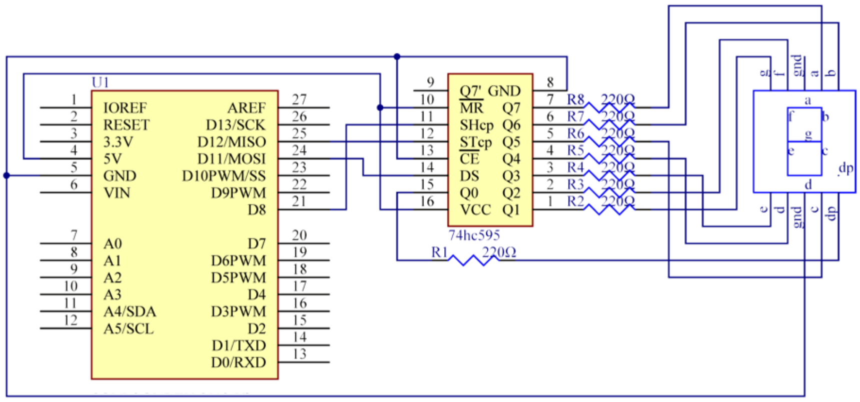
Piny 74HC595 a jeich funkce:
Q0-Q7: 8-bitové výstupy, které jsou schopné řídit 8 LED nebo 8 pinů sedmisegmentového displeje
Q7S: Serial data output pin, který slouží k propojení s dalším obvodem 74HCT595
MR: Reset pin, aktivní na log.0; zapojíme ho na +5V
SH: Shift register clock input je určen pro hodinový signál
ST: Storage register clock input, na náběžné hraně, údaje v posuvném registru přesune do paměťového
OE : Output enable pin, aktivní na log.0; zapojíme ho na GND
Ds : Serial data input pin, sem se zapisují data

Zde se používá funkce shiftout (), která je dodávána s Arduino IDE. Jednoduše zadejte číslo mezi 0 a 255 a registr úložiště ho může převést na 8bitové binární číslo a paralelně jej vyvést. Toto ti umožní snadno ovládat 8 pinů sedmisegmentového displeje a vytvořit libovolné schéma.

Postup experimentu

Krok 1: Vytvoř obvod (Zatím jen pro CC displej)





Krok 2: Napiš kód do Arduino IDE

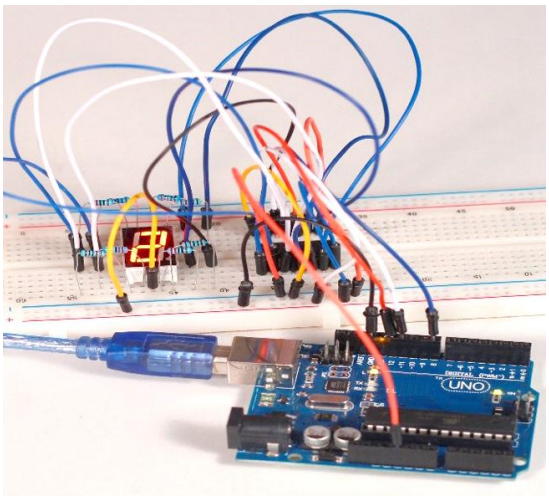
Kód

```
// Řízení sedmisegmentového displeje s 74HC595
// Teď by jsi měl na sedmisegmentovém displeji vidět zobrazené symboly od 0 do F.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

const int latchPin = 12;              // číslo pinu ST_CP 74HC595
const int clockPin = 8;               // číslo pinu SH_CP 74HC595
const int dataPin = 11;               // číslo pinu DS 74HC595
//kódy pro 0,1,2,3,4,5,6,7,8,9,A,b,C,d,E,F
int datArray[16] = {252, 96, 218, 242, 102, 182, 190, 224, 254, 246, 238, 62, 156, 122, 158, 142};

void setup () {
          // nastavení pinu jako výstupu
          pinMode(latchPin,OUTPUT);
          pinMode(clockPin,OUTPUT);
          pinMode(dataPin,OUTPUT);
}

void loop() {
          //cyklus od 0 do 16
          for(int num = 0; num < 16; num++) {
                    // latcPin do stavu LOW pro start zapisování dat do 74HC595
                    digitalWrite(latchPin,LOW);
                    shiftOut(dataPin, clockPin, MSBFIRST, datArray[num]);
                    // latcPin do stavu HIGH pro konec zapisování dat a uložení do 74HC595
                    digitalWrite(latchPin,HIGH);
                    delay(1000);             // počkat 1s
          }
}
```

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Nyní bys měl vidět na displeji blikat znaky od 0 do F, tam a zpět.



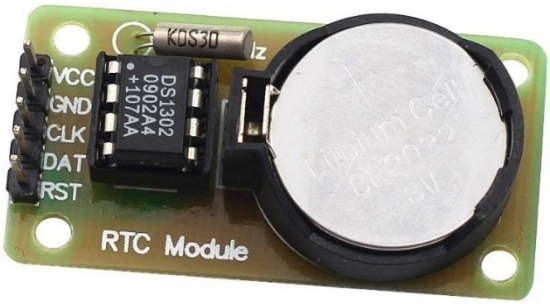
Lekce 23 Hodiny reálného času

Úvod

Existuje mnoho populárních modulů hodin reálného času, jako je DS1302, DS1307, PCF8485 atd. Jsou široce používány vzhledem k jejich jednoduchému "interfejsu", nízké ceně a snadnosti použití. V této lekci budeme používat DS1302 modul hodin reálného času.

Komponenty

- Arduino
- USB kabel
- DS1302 Hodiny reálného času
- LCD displej
- Breadboard vodiče
- Breadboard



Princip

Obvod udržovacích hodin DS1302 obsahuje hodiny reálného času, kalendář a 31B statickou paměť RAM. Zařízení komunikuje s mikroprocesorem přes jednoduché sériové rozhraní. Hodiny reálného času poskytují sekundy, minuty, hodiny, den v týdnu, den v měsíci, měsíc a rok. Konec měsíce je automaticky přizpůsobován u měsíců s méně než 31 dny, včetně korekce u přestupného roku. Hodiny pracují v 24 hodinovém nebo 12 hodinovém režimu s indikátorem AM/PM.

Propojení DS1302 s mikroprocesorem je zjednodušeno užitím synchronního sériového přenosu. Pro komunikaci s hodinami/pamětí jsou použity pouze tři vodiče: CE, I/O (data) a SCLK (časovač). Data mohou být přenesena z/do hodin nebo paměti buď po jednom bajtu nebo až 31 bajtů v burst módu. DS1302 je navržen pro velmi malou spotřebu a udržení dat při příkonu menším, než 1µW.

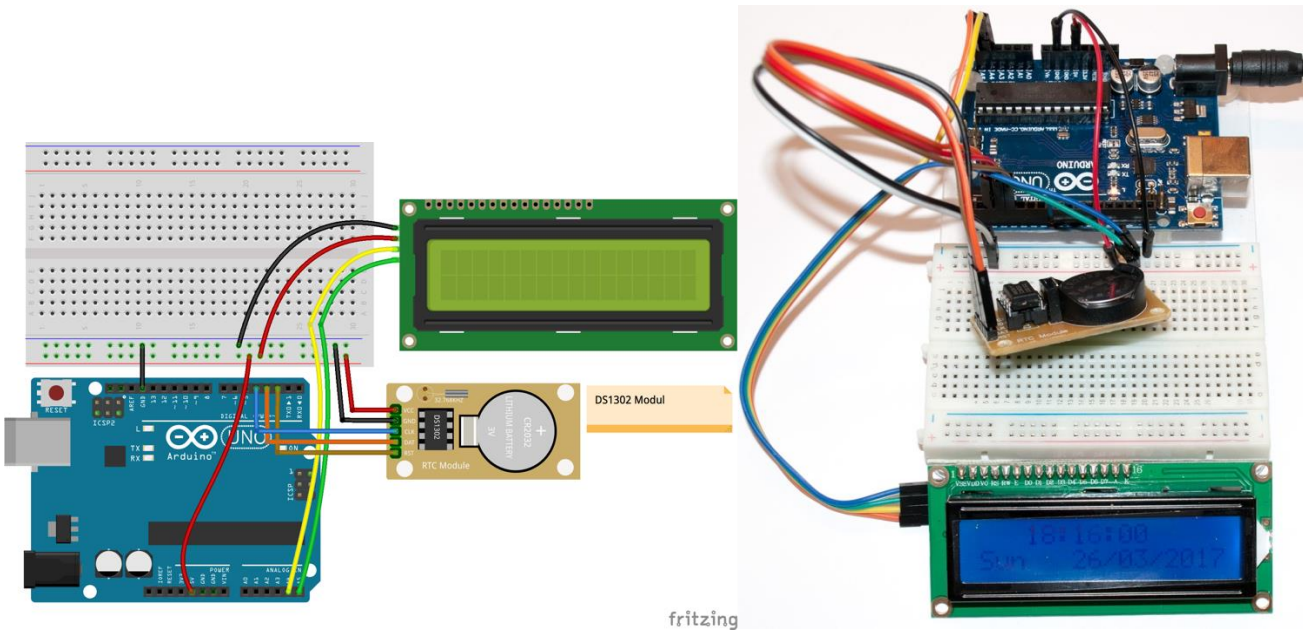
Postup experimentu

Krok 1: Vytvoř obvod

Zapojení mezi I2C LCD1602, DS1302 a Arduino je znázorněno níže:

I2C LCD1602	Arduino
GND	GND
VCC	5V
SDA	A4
SCL	A5

DS1302	Arduino
VCC	5V
GND	GND
CLK	4
DAT	3
RST	2



Krok 2: Napiš kód do Arduino IDE

Poznámka: Sem potřebuješ přidat knihovnu. Mrkni se do popisu v lekci 6.

Kód

```
// Hodiny reálného času
// Teď by jsi na displeji I2C LCD1602 měl vidět správný datum a čas.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

// připoj knihovny
#include <LiquidCrystal_I2C.h>
#include <Time.h>
#include <TimeLib.h>
#include <DS1302RTC.h>
#include <Wire.h>


// inicializace DS1302
// nastavení pinů: CE, IO, CLK
DS1302RTC RTC (2, 3, 4);


// Inicializovat LCD
LiquidCrystal_I2C lcd(0x27,16,2);

void setup () {
          Serial.begin(9600);          // spustit sériový monitor na 9600 bps
          lcd.init();                  // inicializace lcd
          lcd.backlight();             // zapnout podsvícení

          // aktivace RTC
          lcd.print("RTC aktivovan");
          delay(500);

          // kontrola oscilace
          lcd.clear();
          if (RTC.haltRTC())
                    lcd.print("Hodiny zastaveny");
          else
                    lcd.print("Hodiny pracujou");

          // kontrola write-protection
          lcd.setCursor(0,1);
          if (RTC.writeEN())
                    lcd.print("Zapis povolen");
          else
                    lcd.print("Zapis zakazan");
```

```
          delay ( 1000 );

          // nastavení knihovny
          lcd.clear();
          lcd.print("RTC Sync");
          setSyncProvider(RTC.get);    // získat čas z RTC
          if(timeStatus() == timeSet)
                    lcd.print("OK!");
          else
                    lcd.print("SELHALO!");

          delay ( 1000 );
          lcd.clear();

          // nastavit čas a datum
          // (hod,min,sec,den,mesic,rok);
          //setTime(18,15,44,5,6,2017);
}

void loop() {

          // zobrazit čas na displeji
          lcd.setCursor(3, 0);
          print2digits(hour());
          lcd.print(":");
          print2digits(minute());
          lcd.print(":");
          print2digits(second());

          // zobrazit den v týdnu
          lcd.setCursor(0, 1);
          lcd.print(dayShortStr(weekday()));

          // zobrazit datum
          lcd.setCursor(5,1);
          lcd.print(" ");
          print2digits(day());
          lcd.print(".");
          print2digits(month());
          lcd.print(".");
          lcd.print(year());

          // pozor!
          if(timeStatus() != timeSet) {
                    lcd.setCursor(0, 1);
                    lcd.print(F("CHYBA RTC: SYNC!"));
          }
          delay ( 1000 ); // počkej přibližně 1 sec
}

void print2digits(int number ) {
          // přidaní "0" u čísel do 10
          if (number >= 0 && number < 10) {
                    lcd.write('0');
          }
          lcd.print(number);
}
```

Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Nyní můžete vidět datum a čas na LCD displeji.

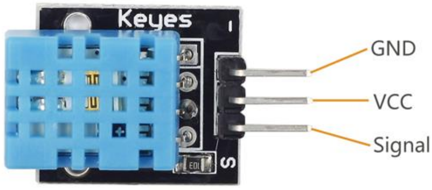
Lekce 24 Senzor teploty a vlhkosti vzduchu DHT11

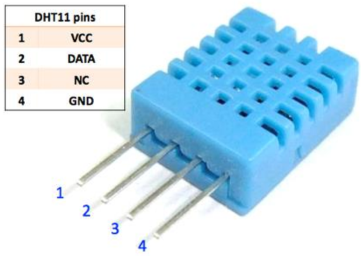
Úvod

Digitální čidlo teploty a vlhkosti DHT11 je kompozitní senzor, který obsahuje kalibrovaný digitální výstup. Snímač obsahuje rezistivní čidlo vlhkosti a NTC teplotní senzor, a je spojen s vysoce výkonným 8-bitovým převodnikem.

Komponenty

- Arduino
- USB kabel
- Teplotní modul DHT11
- LCD displej
- Breadboard vodiče





Princip

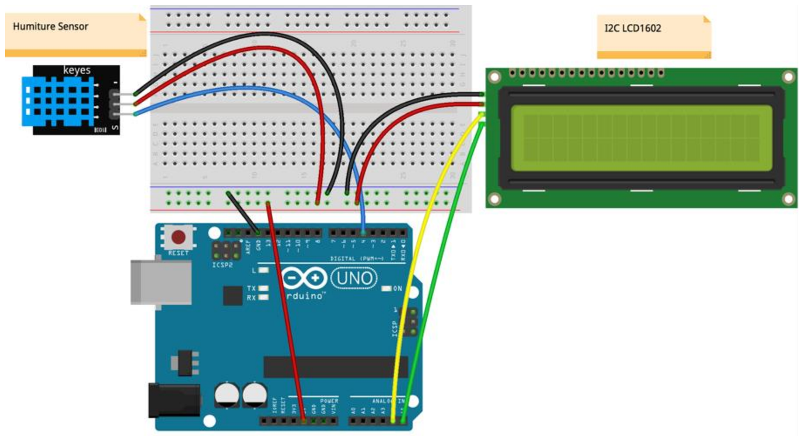
Pouze tři piny jsou k dispozici pro použití: VCC, GND, a DATA. Komunikační proces začíná posláním startového signálu do DHT11. Pak DHT11 přijímá signál a vrací odpověď. Poté hostitel přijme odpověď a začne přijímat 40bitová data (8-bitové integer vlhkost + 8-bitové decimal vlhkost + 8-bitové integer teplota + 8-bitové decimal teplota + 8 bitů checksum).

Postup experimentu

Krok 1: Vytvoř obvod

Humiture Sensor	Arduino
"-"	GND
+	5V
S	4

I2C LCD1602	Arduino
GND	GND
VCC	5V
SDA	A4
SCL	A5



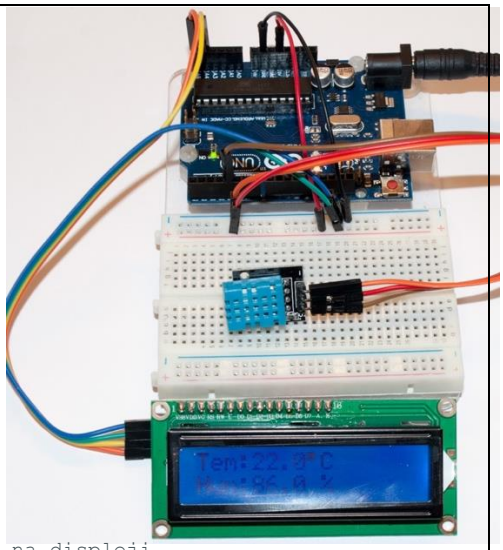
Krok 2: Napiš kód do Arduino IDE

Poznámka: Sem potřebuješ přidat knihovnu. Mrkni se do popisu v lekci 6.

Kód

```
// Senzor teploty a vlhkosti vzduchu DHT11
// Na displeji I2C LCD1602 uvidíš vlhkost a teplotu vzduchu.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/


#include <dht.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27,16,2);

dht DHT; //vytvořit objekt třídy dht

const int DHT11_PIN= 4;

void setup() {
         Serial.begin(9600);          // spustit sériový monitor na 9600 bps
         lcd.init();                  // inicializace lcd
         lcd.backlight();             // zapnout podsvícení
}

void loop() {
```

```
            lcd.setCursor(0, 0);
            // přečíst hodnoty vrácené z čidla
            int chk = DHT.read11(DHT11_PIN);
            switch (chk) {
                        case DHTLIB_OK:
                                    lcd.print("DHT11: OK!");
                                    break;
                        case DHTLIB_ERROR_CHECKSUM:
                                    lcd.print("Checksum chyba");
                                    break;
                        case DHTLIB_ERROR_TIMEOUT:
                                    lcd.print("Time out chyba");
                                    break;
                        default:
                                    lcd.print("Neznama chyba");
                                    break;
            }
            delay(500);
            // zobrazit data
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("Teplota:");
            lcd.print(DHT.temperature,1);          // zobrazit teplotu na displeji
            lcd.print(char(223));                  // zobrazit znak "°"
            lcd.print("C");
            lcd.setCursor(0, 1);
            lcd.print("Vlhkost:");
            lcd.print(DHT.humidity,1);             //zobrazit vlhkost na displeji
            lcd.print(" %");
            delay(300);                            // počkat 300 ms
}
```



Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Teď uvidíš aktuální vlhkost a teplotu na displeji.

Lekce 25 Modul relé

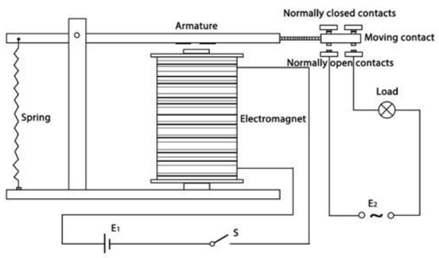
Úvod

Elektromagnetické relé je elektrotechnická součástka, která obsahuje elektromagneticky ovládané kontakty. Relé bylo vynalezeno roku 1835 Josefem Henrym a původně bylo využíváno jako mechanický zesilovač na telegrafních linkách. Jeho název pochází z přepřahacích stanic na kurýrních cestách.
Relé se používá v mnoha aplikacích, byť jeho funkci v mnoha případech přebírají obvody založené na polovodičích. Na rozdíl od polovodičů, relé obvykle zajišťuje galvanické - izolační oddělení řídícího a řízeného obvodu.
Elektromagnetické relé je příkladem využití elektromagnetu v zařízení, které je důležitým funkčním prvkem v automatizovaných soustavách a řídících systémech.

Komponenty

- Arduino (or ArduinoMega2560 board)
- USB kabel
- 1-kanál relé modul
- Dupont kabely samec-samice

Princip

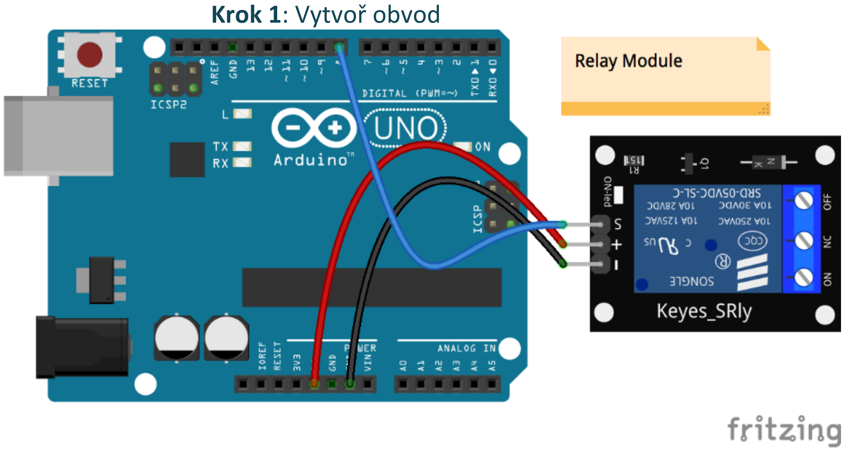


Relé se v základním provedení skládá z cívky (elektromagnetu) navinuté na jádru z měkkého feromagnetického materiálu. Magnetický obvod je uzavřen pohyblivou kotvou. Kotva je pružinou uváděna do klidové polohy a současně se opírá o pohyblivý kontakt. Po připojení cívky na elektrický zdroj vyvolá proud cívkou v magnetickém obvodu magnetický tok. Magnetický tok vyvolá přitažlivou sílu na kotvu, která přemůže sílu v pružině a překlopí kontakt. Po odpojení el. proudu se kotva a kontakt vrátí do předchozího, klidového stavu.

Když pošleme z Arduino log.0 (0V) do vstupu modulu relé, tranzistor se otevře. Spínací kontakt relé bude uzavřen, zatímco rozpínací kontakt relé se přeruší. Když pošleme z Arduino log. 1 (5V), tranzistor se zavře a relé se vrátí do původního stavu.

Postup experimentu

Krok 1: Vytvoř obvod



Krok 2: Napiš kód do Arduino IDE

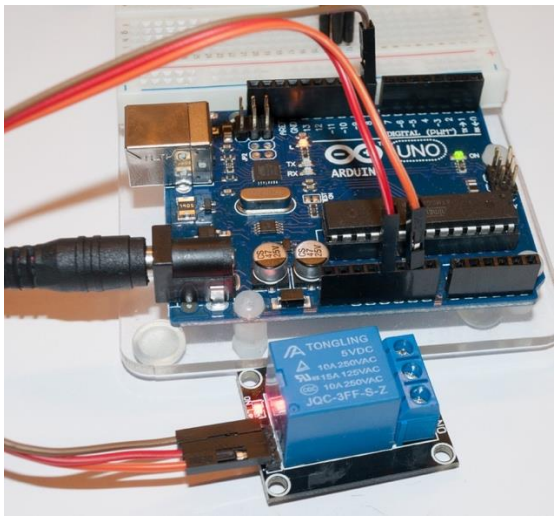
Kód

```
// Modul relé
// Každou minutu zavírej a otevírej relé.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

const int relayPin = 8;                    // číslo pinu rele

void setup() {
        pinMode(relayPin, OUTPUT);        // nastavení pinu relé jako výstupu
}

void loop() {
        digitalWrite(relayPin, HIGH);     // vypnut relé
        delay(1000);                      // počkat 1s
        digitalWrite(relayPin, LOW);      // zapnut relé
        delay(1000);                      // počkat 1s
}
```

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Teď můžete slyšet známé „tik-tak, tik-tak“. To znamená, že normálně uzavřený kontakt se otevře a naopak, normálně otevřený kontakt se uzavře.



Lekce 26 Krokový motor

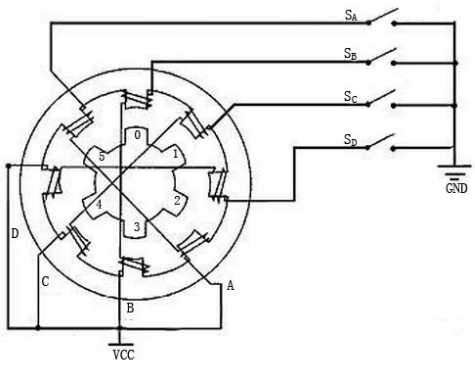
Úvod

Krokové motory, díky své jedinečné konstrukci, mohou být ovládány s vysokou přesností a bez jakýchkoli mechanismů zpětné vazby. Na hřídel krokového motoru je namontována série magnetů. Ta je řízena sérií elektromagnetických cívek, které jsou v určitém pořadí řízeny sérií elektrických impulsů a dokáží se přesně otáčet dopředu a dozadu v pomalých „krocích“.
Existují dva typy krokových motorů – unipolar a bipolar. Je velmi důležité, abys věděl, s jakým typem pracuješ. V tomto experimentu použijeme krokový motor unipolar.
Arduino desky nebo kontrolery nemohou přímo řídit krokové motory. Je tedy nezbytný řídící obvod, proto používáme řadič krokového motoru (jako je ukázáno na následujícím obrázku) k pohonu krokového motoru.

Komponenty

- Arduino
- USB kabel
- Potenciometr
- Krokový motor
- Řadič krokový motor
- Breadboard vodiče
- Breadboard

Princip



Krokový motor je čtyřfázový a používá stejnosměrný napájecí zdroj. Po napájení všech fází motoru dle příslušného pořadí se začne otáčet krok za krokem. Schematický diagram čtyřfázového krokového motoru je znázorněn níže:
Na obrázku je uprostřed motoru rotor – permanentní magnet ve tvaru ozubeného kola. Okolo rotoru je 0 až 5 zubů. Více jich je venku. Tam je 8 magnetických pólů, každý se dvěma protilehlými a ty jsou spojené vinutou cívkou. Takže tvoří čtyři páry od A do D. Ty se nazývají fáze. Mají čtyři vodiče připojené ke spínačům SA, SB, SC a SD. Proto jsou čtyři fáze paralelně v obvodu a dva magnetické póly v jedné fázi jsou v sérii.
Níže se dočteš, jak 4-fázový krokový motor funguje:
Na začátku je spínač SB zapnutý, spínače SA, SC a SD jsou vypnuty a magnetické póly ve fázi B jsou zarovnány s ozubením 0 a 3 rotoru. Současně 1 a 4 vytváří střídavě uspořádané zuby s C- a D-fází pólu. Zuby 2 a 5 vytvářejí odstupňované zuby s póly D a A fáze.

Když je spínač SC zapnutý, spínače SB, SA a SD jsou vypnuty, rotor se otáčí pod magnetickým polem C vinutí a mezi zubem 1 a 4. Potom se zub 1 a 4 vyrovná s magnetickými póly C-fázového vinutí. Když zub 0 a 3 vytváří stoupavé zuby s póly fáze A a B, zuby 2 a 5 vytváří stoupavé zuby s magnetickými póly A až D. Podobné situace se neustále opakují. Zapněte fáze A, B, C a D, rotor se začne otáčet v pořadí of A, B, C a D.

Čtyřfázový krokový motor má tři provozní režimy: samostatný čtyř-krokový, dvojitý čtyř-krokový a osmi-krokový. Krokový úhel pro samostatný čtyř-krokový a dvojitý čtyř-krokový režim je stejný, ale hnací moment pro samostatný čtyř-krokový režim je menší. Krokový úhel pro osmi-krokový režim je poloviční, než u režimu samostatného čtyř-krokového a dvojitého čtyř-krokového. Osmistupňový provozní režim tak může udržet vysoký hnací moment a zlepšit přesnost řízení. V tomto experimentu necháváme krokový motor pracovat v osmi-krokovém režimu.

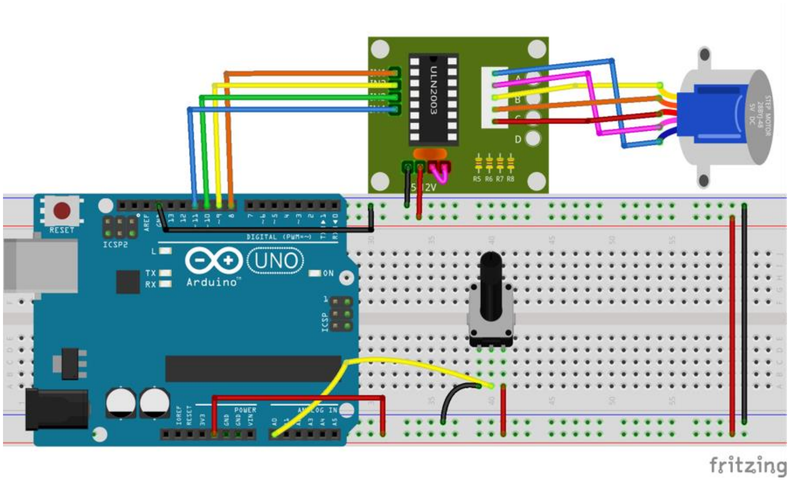
Pokud chcete použít motor v obvodu, musíte použít řídící desku. Řadič krokového motoru ULN2003 je 7-kanálový invertorový obvod. To znamená, že když je konec vstupu na vysoké úrovni, konec výstupu ULN2003 je na nízké úrovni, a naopak. Pokud je na IN1 vysoká úroveň a nízká úroveň na IN2, IN3 a IN4, pak je konec výstupu OUT1 na nízké úrovni a všechny ostatní konce jsou na úrovni vysoké. To tedy znamená, že D1 se rozsvítí, spínač SA se zapne a krokový motor se začne otáčet. Tato situace se bude stále opakovat. Proto stačí dát krokovému motoru určitou časovou posloupnost. Ten se pak bude postupně otáčet. ULN2003 se zde používá k poskytnutí konkrétních časových sekvencí pro krokový motor.

Postup experimentu

Krok 1: Vytvoř obvod

Zapojení mezi Krokovým motorem-řadičem a Arduino:

Stepper Motor Driver	Arduino
IN1	2
IN2	4
IN3	3
IN4	5
GND	GND
VCC	5v



Krok 2: Napiš kód do Arduino IDE

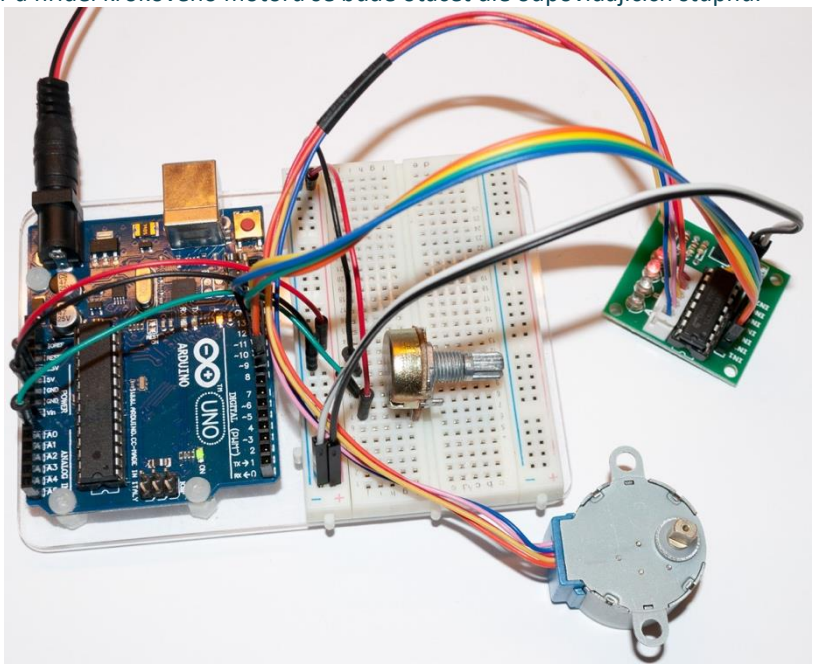
Poznámka: Sem potřebuješ přidat knihovnu. Mrkni se do popisu v lekci 6.

Kód

```
// Krokový motor
// Krokový motor následuje potenciometr.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

#include <Stepper.h>

// počet kroků motoru
#define STEPS 100

// Vytvoříme objekt třídy Stepper, specifikujeme
// počet kroků motoru a piny, ke kterým je řadič připojen
Stepper stepper(STEPS, 8, 9, 10, 11);

// předchozi stav analogového vstupu
int previous = 0;

void setup() {
        // nastavit rychlost motoru na 60 ot/m
        stepper.setSpeed(60);
}

void loop() {
        // přečíst hodnotu pinu A0
        int val = analogRead(0);
```

```
        // otočit motor na počet kroků = rozdilu
        //předchozího stavu potenciometru a aktualního
        stepper.step(val - previous);

        // zapamatovat stav pinu A0
        previous = val;
}
```

Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Nyní nastav potenciometr a hřídel krokového motoru se bude otáčet dle odpovídajících stupňů.



Lekce 27 Servo

Úvod

Servomotor (Servo) je typ motoru s převodovkou, který se může otáčet jen o 180 stupňů (některý víc) a je řízen impulsy z Arduino. Tyto impulsy „řeknou“ servu, do jaké pozice se má otočit.
Servo má tři vodiče. Hnědý vodič je GND, červený vodič je VCC a oranžový vodič je signál.

Komponenty

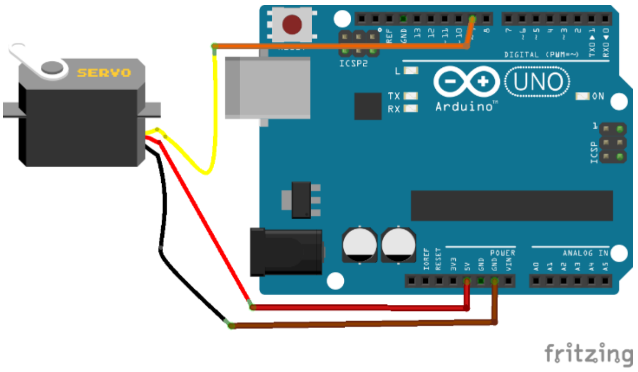
- Arduino
- USB kabel
- Servomotor
- Breadboard vodiče

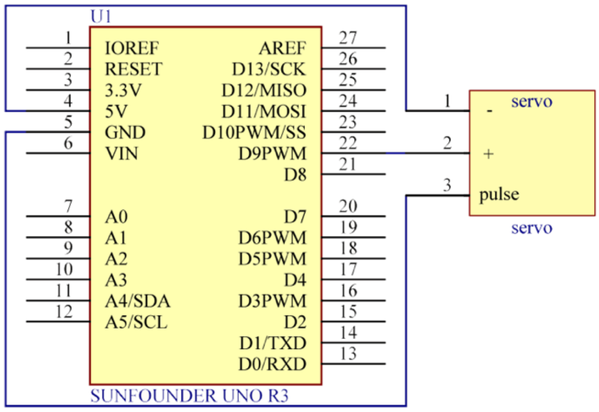
Princip

Servo se skládá z pláště, obvodové desky, non-jádra motoru, převodovky a detekce polohy. Jeho pracovní princip je následující: Arduino vysílá signál PWM do serva. Tam je tento signál zpracováván IO pro výpočet směru otáčení k řízení motoru, a pak je tato hnací síla přenášena na otočné rameno pomocí ozubeného převodu. Ve stejné době vrací detektor signál pozici, jestli je dosaženo nastavení polohy nebo ne.

Postup experimentu

Krok 1: Vytvoř obvod





Krok 2: Napiš kód do Arduino IDE

Kód

```
// Servo
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

#include <Servo.h>

Servo myservo;                        // vytvořit objekt pro řízení serva
const int servoPin = 9;               // číslo pinu serva

int pos = 0;                          // proměnna pro ukládání pozice serva

void setup() {
  myservo.attach(servoPin);           // nastavení pinu serva
}

void loop() {
  for (pos = 0; pos <= 160; pos += 1) {// točení servem od 0 do 160 stupnů
    // in steps of 1 degree
    myservo.write(pos);               // nastavit servo do pozice "pos"
    delay(15);                        // počkat 15ms než servo dosáhne pozice
  }
  for (pos = 160; pos >= 0; pos -= 1) { // točení servem od 160 do 0 stupnů
    myservo.write(pos);               // nastavit servo do pozice "pos"
    delay(15);                        // počkat 15ms než servo dosáhne pozice
  }
}
```

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

Nyní uvidíš servomotor otáčet se o 90 stupňů (otáčí se jedenkrát, každých 15 stupňů). A poté se otáčí v opačném směru.

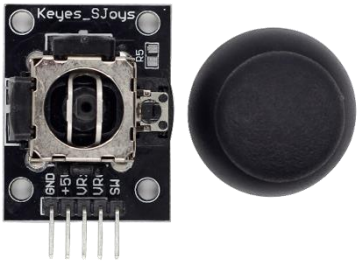
Lekce 28 Joystick PS2

Úvod

Joystick je vstupní zařízení, sestávající z tyče, která se otáčí na základně a udává úhel nebo směr k ovládacímu systému zařízení. Joysticky jsou často používány pro ovládání videoher a robotů. Zde je použit PS2 joystick.

Komponenty

- Arduino
- USB kabel
- Joystick PS2
- Breadboard vodiče

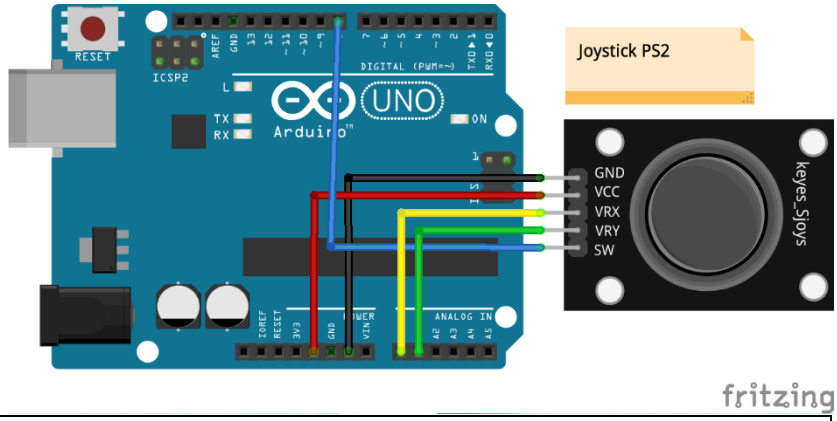


Princip

Tento modul má dva analogové výstupy (odpovídající souřadnicím X a Y) a jeden digitální výstup, představující tlačítko na ose Z.

Postup experimentu

Krok 1: Vytvoř obvod



Krok 2: Napiš kód do Arduino IDE

Joystick PS2	Arduino
GND	GND
VCC	5V
X	A0
Y	A1
SW	8

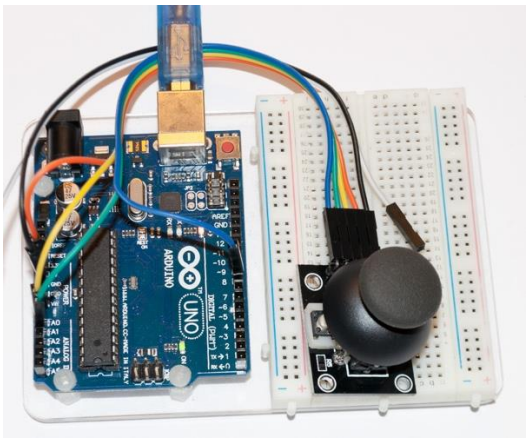
Kód

```
// Joystick PS2
// Pohybuj s joystickem, stiskni joystick a souřadnice X, Y a Z sériového monitoru se budou měnit.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

const int xPin = A0;          // číslo pinu osy X
const int yPin = A1;          // číslo pinu osy Y
const int swPin = 8;          // číslo pinu tlačítka

void setup() {
          pinMode(swPin,INPUT);                 // nastavení pinu tlačítka jako vstupu
          digitalWrite(swPin, HIGH);            // "HIGH" zapne pullup odpor zabudovaný do Atmelu
          Serial.begin(9600);                   // spustit sériový monitor na 9600 bps
}

void loop() {
          Serial.print("X: ");
          // přečíst hodnotu pinu xPin a tisknout do sériového monitoru, jak decimal
          Serial.print(analogRead(xPin),DEC);
          Serial.print("|Y: ");
          // přečíst hodnotu pinu yPin a tisknout do sériového monitoru, jak decimal
          Serial.print(analogRead(yPin),DEC);
          Serial.print("|Z: ");
          // přečíst stav pinu tlačítka a tisknout do sériového monitoru
          Serial.println(digitalRead(swPin));
          delay(500);                           // počkat 100 ms
}
```

Krok 3: Zkompiluj kód

Krok 4: Nahraj sketch do Arduino

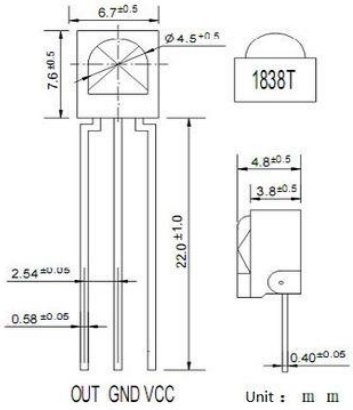
Nyní hýbej s joystickem a souřadnice X a Y na monitoru se změní. Stiskni joystick a uvidíš, že se současně zobrazí souřadnice Z = 0.



Lekce 29 Infračervený přijímač

Úvod

Hlavním prvkem tohoto infračerveného dálkového ovládání je přijímač HX1838. Tento infračervený přijímač umožňuje ve svém okolí detekovat signály z IR dálkových ovladačů a při jeho připojení k Arduino můžeme tyto signály převést na proměnné a dále využít. V našem Kitu se společně s přijímačem nachází dálkový ovladač, kterým můžeme přiřadit libovolné funkce v programu. Můžeš ale využít i jiné dálkové ovladače, které máš doma, třeba od televize či HIFI věže.



Komponenty

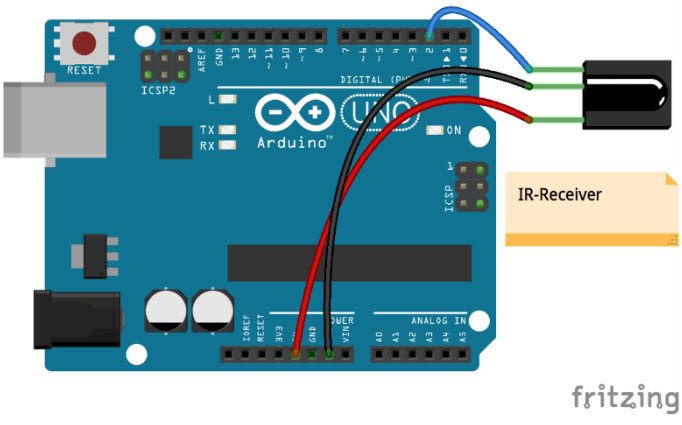
- Arduino
- USB kabel
- IR přijímač
- IR ovladač
- Breadboard vodiče

Princip

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

Při modulaci logickou úrovní 1 se infra-LED budí obdélníkovým nosným signálem o dané střídě a frekvenci, typicky mezi 30-56kHz (38kHz v našem případě). Tento stav se nazývá značka. Po dobu logické úrovně 0 není dioda aktivní, nastává mezera. Dobu trvání značky a mezery, stejně tak jako jejich význam, specifikuje použitý protokol. V klidu, tj. když neprobíhá přenos rámce, setrvává vysílač ve stavu log.0. Přijímač signálu je nastaven na použitý nosný kmitočet. Tím, že ostatní kmitočety odfiltruje, účinně zamezuje možnému rušení okolním světlem.

Když stiskneš klávesu Play / Pause, infračervené paprsky budou emitovány z dálkového ovladače a přijímány infračerveným přijímačem a LED na Arduino se rozsvítí.



Postup experimentu

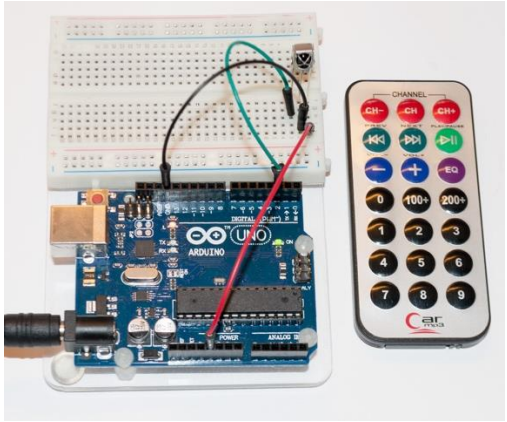
Krok 1: Vytvoř obvod

Infrared-receiver module	Arduino
S	2
"-"	GND
+	5V

Krok 2: Napiš kód do Arduino IDE
Kód

```
// Infračervený přijímač
// Stiskni tlačítko „Play“ na dálkovém ovladači a LED,
// připojená k vývodu 13 na Arduino, se rozsvítí.
// Stiskni libovolnou jinou klávesu a LED zhasne.
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

#include <IRremote.h>

const int IRPin = 2;                            // číslo pinu tlačítka přijímače
const int ledPin = 13;                          // číslo pinu zabudované LEDky

IRrecv irrecv(IRPin);                           // vytvořeme objekt typu IRrecv
decode_results results;                         // definujeme proměnnou

void setup() {
          pinMode(ledPin,OUTPUT);               // nastavení pinu LEDky jako výstupu
          Serial.begin(9600);                   // spustit sériový monitor na 9600 bps
          irrecv.enableIRIn();                  // zapnout infračervený přijímač
}

void loop() {
          if (irrecv.decode(&results)) {        // jestli přijímač přijal data
                    Serial.print("IR Kod: ");    // tisknout "irCode: "
                    Serial.print(results.value, HEX);  // tisknout data jako hexdecimal
                    Serial.print(",  bity: ");   // tisknout " ,  bity: "
                    Serial.println(results.bits);    // tisknout bity
                    irrecv.resume();            // přijat další data
          }
          delay(600);  //počkat 600ms
          if(results.value == 0xFFC23D) {       // Kód tlačitka "Play"
                    digitalWrite(ledPin,HIGH);  // zapnout LEDku
          } else {
                    digitalWrite(ledPin,LOW);   // vypnout LEDku
          }
}
```

Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Nyní stiskni tlačítko „Play“ na dálkovém ovladači a LED, připojená k vývodu 13 na Arduino, se rozsvítí. Stiskni libovolnou jinou klávesu a LED zhasne.



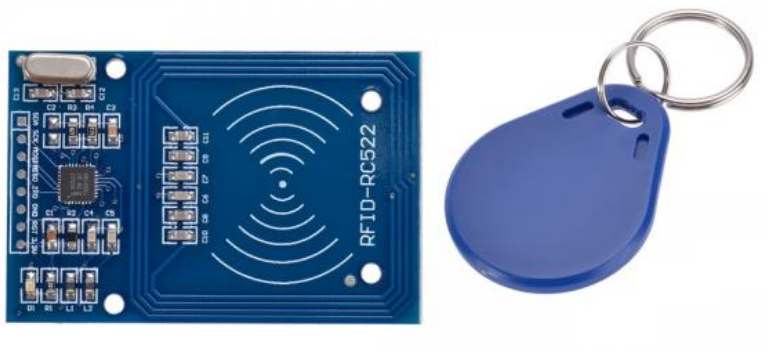
Lekce 30 RFID vstupní ochranný systém

Úvod

RFID je zkratka pro radiofrekvenční identifikaci. Jedná se o bezdrátovou aplikaci pro přenos dat za účelem identifikace a sledování značek. V tomto experimentu budeme používat modul RFID - relé, a I2C LCD1602 pro sestavení vstupního ochranného systému.

Komponenty

- Arduino
- USB kabel
- RFID čtečka
- RFID čipový přívěsek na klíče
- RFID čipová karta
- 1-kanál relé modul
- I2C LCD1602
- Breadboard vodiče
- Breadboard



Princip

Za prvé, potřebuješ znát ID klíče RFID tagu a napsat toto ID do rfidTest souboru. Zkompiluj kód. Uvidíš zobrazené slovo „Vítej“ na I2C LCD1602. Protáhni RFID kód-kroužek na modul RFID. Pokud je heslo správné, normálně otevřený kontakt relé se uzavře a na displeji se zobrazí řetězec “ID:5AE4C955” se slovy“ Ahoj LaskArduino“, hned poté „Vítej“. Pokud je heslo nesprávné, bude otevřený kontakt relé odpojen a na displeji se zobrazí text „Ahoj, neznámý chlapíčku“, hned poté „Vítej“.

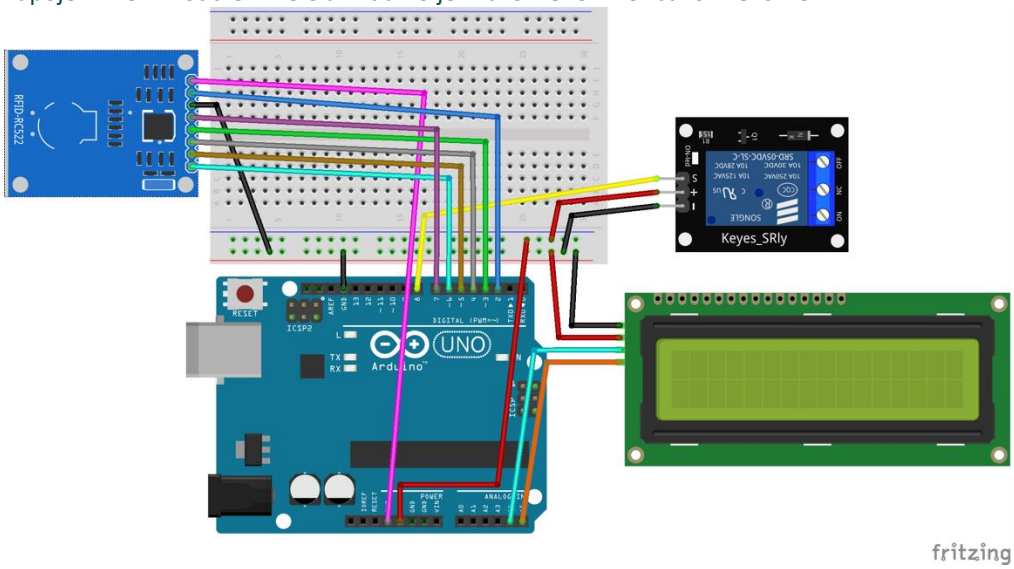
Poznámka: Pro tento modul použij, prosím, 3.3V napájení nebo Ti to shoří.

Postup experimentu

Krok 1: Vytvoř obvod

RFID	Arduino
VCC	3.3V
RST	2
GND	GND
MISO	3
MOSI	4
SCK	5
SDA	6
IRQ	7

Zapojení mezi I2C LCD1602 a Arduino je znázorněno níže: Mrkni se do popisu v lekci 6.
Zapojení mezi Modulem Relé a Arduino je znázorněno níže: Jako v lekci 25.



Krok 2: Napiš kód do Arduino IDE

Poznámka: Sem potřebuješ přidat knihovnu. Mrkni se do popisu v lekci 6.

Kód getId.ino

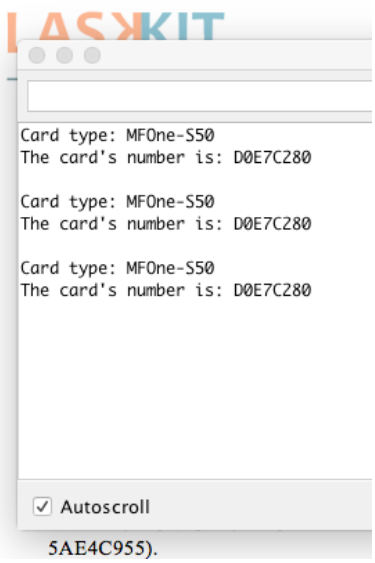
```
// RFID vstupní ochranný systém -> getId.ino
// Umísti klíč RFID tagu do indukční zóny modulu RFID.
// Uvidíš následující hodnoty vytištěné na seriálovém monitoru:
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

#include"rfid1.h"
RFID1 rfid;                    // vytvoříme objekt typu RFID1

uchar serNum[5];               // pole pro ukládání ID

void setup() {
          Serial.begin(9600);// spustit sériový monitor
          //rfid.begin(IRQ_PIN,SCK_PIN,MOSI_PIN,MISO_PIN,NSS_PIN,RST_PIN) -IRQ_PIN není potřeba
zapojovat
          rfid.begin(7, 5, 4, 3, 6, 2);
          delay(100);
          rfid.init();         //inicializace RFID
}
void loop() {
          uchar status;
          uchar str[MAX_LEN];
          // Hledání čipu
          status = rfid.request(PICC_REQIDL, str);
          if (status != MI_OK) {
                    return;
          }
          // Ukázat druh čipu
          rfid.showCardType(str);
          //Předejdi konfliktu a vrať 4 bajty sériového čísla karty
          status = rfid.anticoll(str);

          if (status == MI_OK) {
                    Serial.print("The card's number is: ");
                    memcpy(serNum, str, 5);
                    rfid.showCardID(serNum);     // Ukázat ID čipu
                    Serial.println();
                    Serial.println();
          }
          delay(500);
          rfid.halt();                           // uspat modul
}
```

Kód rfid.ino

```
// RFID vstupní ochranný systém -> rfid.ino
// Teď projeď klíč RFID tagu přes RFID modul. Pokud je heslo správné, LCD zobrazí „ID:
// D0E7C280“ „Zdravím Bastliře“. O dvě vteřiny později se na dipleji zobrazí „Vítej!“.
// Pokud je heslo nesprávné, LCD zobrazí: „Ahoj cizinče!“, a poté opět po dvou vteřinách „Vítej!“..
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

#include"rfid.h"
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27,16,2);
RFID rfid;                             // vytvořeme objekt typu RFID1
#define relayPin 8                     // číslo pinu relé

uchar serNum[5];                       // pole pro ukladaní ID

void setup() {
          lcd.init();
          lcd.backlight();
          Serial.begin(9600);
          //rfid.begin(IRQ_PIN,SCK_PIN,MOSI_PIN,MISO_PIN,NSS_PIN,RST_PIN) - IRQ_PIN není potřeba
zapojovat
          rfid.begin(7, 5, 4, 3, 6, 2);
          delay(100);
```

```
        rfid.init();                // inicializace RFID
        pinMode(relayPin, OUTPUT);  // nastavení pinu relé jako výstupu
        digitalWrite(relayPin,LOW); // zapnut relé

        lcd.setCursor(0,0);
        lcd.print("     Vitej!    ");// zobrazit "    Vitej!    "
        delay(2000);
}
void loop() {
        uchar status;
        uchar str[MAX_LEN];
        // Hledání čipu
        status = rfid.request(PICC_REQIDL, str);
        if (status != MI_OK) {
                return;
        }
        // Ukázat druh čipu
        rfid.showCardType(str);
        // Předejdi problému a vrať 4 bajty sériového čísla karty.
        status = rfid.anticoll(str);

        if (status == MI_OK) {
                lcd.setCursor(0,0);
                lcd.print(" ID: ");
                memcpy(serNum, str, 5);
                rfid.showCardID(serNum);  // Ukázat ID čipu

                // Jestli ID čipu je 4BE6D13B, zapni relé
                uchar* id = serNum;
                if( id[0]==0x4B && id[1]==0xE6 && id[2]==0xD1 && id[3]==0x3B ) {
                        digitalWrite(relayPin,HIGH);
                        lcd.setCursor(0,1);
                        lcd.print("  Ahoj Kosto!   ");
                        delay(2000);
                        lcd.clear();
                        digitalWrite(relayPin,LOW);
                }
                // Jestli ID čipu je D0E7C280, zapni relé
                else if(id[0]==0xD0 && id[1]==0xE7 && id[2]==0xC2 && id[3]==0x80) { // D0-E7-C2-80
                        digitalWrite(relayPin,HIGH);
                        lcd.setCursor(0,1);
                        lcd.print("Zdravim Bastlire");
                        delay(2000);
                        lcd.clear();
                        digitalWrite(relayPin,LOW);
                } else {
                        lcd.setCursor(0,1);
                        lcd.print(" Ahoj cizinec!");
                        delay(2000);
                        lcd.clear();
                }
        }
        lcd.setCursor(0,0);
        lcd.print("     Vitej!    ");
        delay(2000);
        rfid.halt(); // uspat modul
}
```

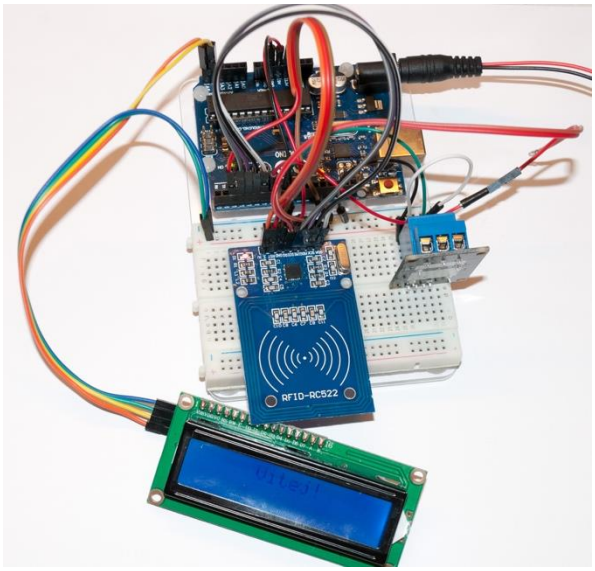
Krok 3: Nahraj sketch do Arduino
Umísti klíč RFID tagu do indukční zóny modulu RFID. Uvidíš následující hodnoty vytištěné na Seriál monitoru:



Krok 4: V tomto okamžiku bys už měl znát ID svého klíče RFID tagu (např.: ID mé magnetické karty je D0E7C280).
Otevři rfidTest a nahraď ID v náčrtku číslem ID, které sis právě zapsal (rozděl toto ID do čtyř částí a vyplň je podle následujícího formátu), takto:

```
else if(id[0]==0xD0 && id[1]==0xE7 && id[2]==0xC2 && id[3]==0x80) { // D0-E7-C2-80
  digitalWrite(relayPin,HIGH);
  lcd.setCursor(0,1);
  lcd.print("Zdravim Bastlire");
  delay(2000);
  lcd.clear();
  digitalWrite(relayPin,LOW);
```

Krok 5:
Teď projeď klíč RFID tagu přes RFID modul. Pokud je heslo správné, LCD zobrazí „ID: D0E7C280“ „Zdravím Bastlíře“. O dvě vteřiny později se na dipleji zobrazí „Vítej!“. Pokud je heslo nesprávné, LCD zobrazí: „Ahoj cizinče!“, a poté opět po dvou vteřinách „Vítej!“..



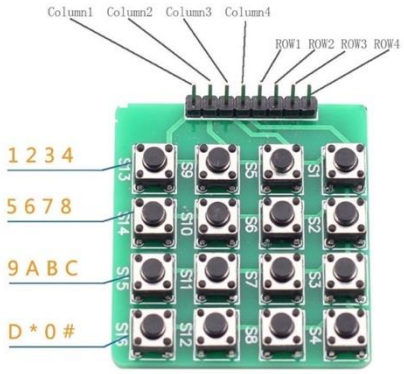
Lekce 31 Vstupní ochranný systém s heslem

Úvod

Poté, co ses už naučil tolik samostatných modulů, pojď si zkombinovat tyto moduly dohromady a vytvořit něco zábavného a interaktivního. V této lekci použijeme I2C LCD1602, modul relé, potenciometr a klávesnici k sestavení jednoduchého hesla k zámku. To je založeno na Arduino-mikrokontroleru a obecně je využito hlavně pro bezpečnostní dveře.

Komponenty

- Arduino
- USB kabel
- 1-kanál relé modul I2C LCD1602
- 4x4 Maticová tlačítková klávesnice
- Breadboard vodiče



Princip

Nastav heslo (např.: 123456) pomocí programování. Po zapnutí se pak na I2C LCD1602 displeji objeví „Vítej!“. V této fázi se normálně otevřený kontakt relé odpojí a indikátor LED na relé modulu zůstane ve vypnutém režimu. Stiskni klíč označený hvězdičkou "*" pro zadání hesla, poté stiskni "#". Pokud je heslo správné, normálně otevřený kontakt relé se uzavře a kontrolka se rozsvítí. Na displeji I2C LCD1602 se objeví „Správně zadáno!“ „Prosím, vstupte!“. Pokud zadáš jiné heslo, na displeji I2C LCD1602 se objeví „Vstupní chyba!“ „Prosím, zkus znovu!“ a relé zůstane v původním stavu. Po uplynutí dvou sekund se na displeji I2C LCD1602 objeví „Vítej!“.

POZNÁMKA: Prostřednictvím programování nastav čtyři klíče v první řadě (s piny v horní části, jak vidíš na obrázku), jako 1,2,3,4; ve druhé řadě jako 5,6,7,8; ve třetí řadě jako 9,A,B a C; ve čtvrté řadě jako D, *, 0, #.

Postup experimentu

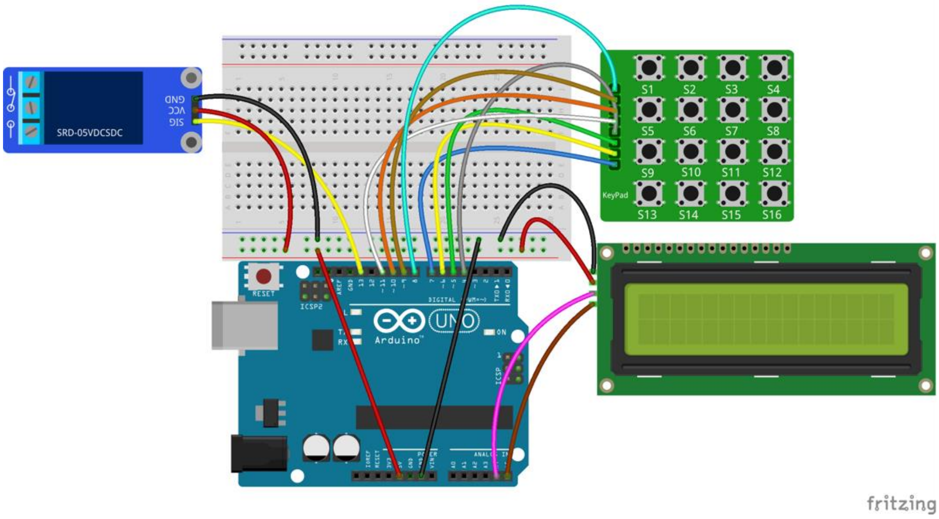
Krok 1: Vytvoř obvod

I2C LCD1602	Arduino
GND	GND
VCC	5v
SDA	A4
SCL	A5

Zapojení mezi Modulem Relé a Arduino: Mrkni se do popisu v lekci 25.

Zapojení mezi klávesnicí a Arduino:

Keypad	Arduino
Column 1	7
Column 2	6
Column 3	5
Column 4	4
Row 1	11
Row 2	10
Row 3	9
Row 4	8



Krok 2: Napiš kód do Arduino IDE

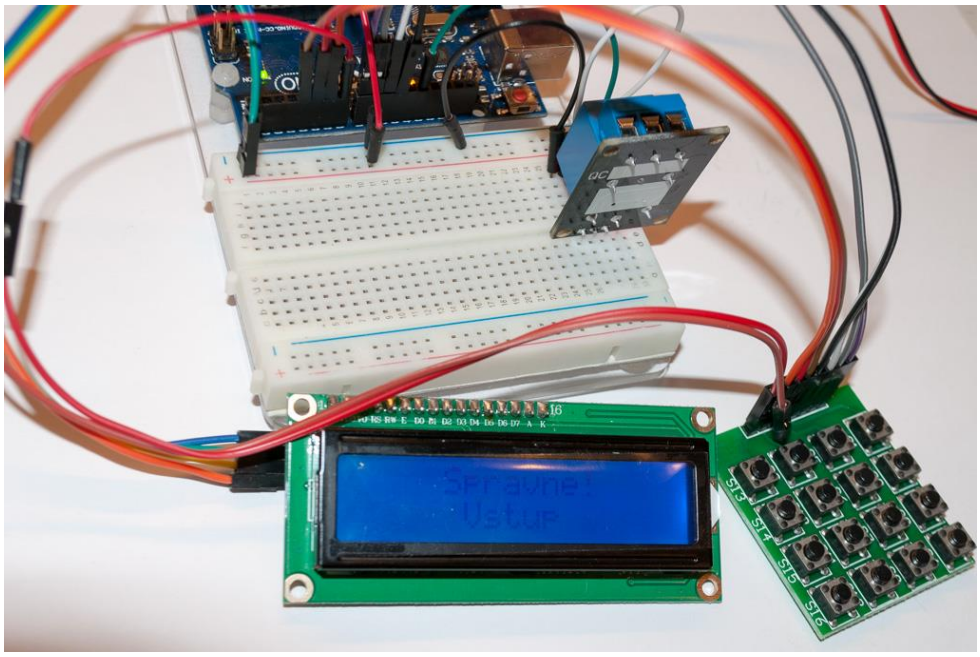
Poznámka: Sem potřebuješ přidat knihovnu. Mrkni se do popisu v lekci 6.

Kód

```
// Vstupní systém pomocí hesla
// Email:laskarduino@gmail.com
// Web:laskarduino.cz
/*/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/*/

#include <Keypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>


LiquidCrystal_I2C lcd(0x27,16,2);


#define relayPin 13          // číslo pinu relé


const byte ROWS = 4;         // 4 řádky
const byte COLS = 4;         // 4 sloupce
// definujeme  znaky tlačitek klávesnice
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','4'},
  {'5','6','7','8'},
  {'9','A','B','C'},
  {'D','*','0','#'}
};
byte rowPins[ROWS] = {11, 10, 9, 8};   // číslo pinu řádků
byte colPins[COLS] = { 7, 6, 5, 4};    // číslo pinu sloupců


int pos = 0;
char secretCode[6] = {'1', '2', '3', '4', '5', '6'};
char inputCode[6] = {'0', '0', '0', '0', '0', '0'};

//vytvoříme objekt typu NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup() {
         lcd.init();
         lcd.backlight();
         pinMode(relayPin, OUTPUT);
         lcd.setCursor(0,0);
         lcd.print("     Vitej!     ");
         delay(2000);
}

void loop() {
         readKey();
}
```

```
void readKey() {
          int correct = 0;
          int i;
          char customKey = customKeypad.getKey();
          if (customKey) {
                    switch(customKey) {
                              case '*':
                                        pos = 0;
                                        lcd.clear();
                                        lcd.setCursor(0,0);
                                        lcd.print("Zadej sve heslo:");
                                        break;
                              case '#':
                                        for(i = 0; i < 6; i++) {
                                                  if(inputCode[i] == secretCode[i]) {
                                                            correct ++;
                                                  }
                                        }
                                        if(correct == 6) {
                                                  lcd.clear();
                                                  lcd.setCursor(0, 0);
                                                  lcd.print("    Spravne!    ");
                                                  lcd.setCursor(0, 1);
                                                  lcd.print("     Vstup      ");
                                                  digitalWrite(relayPin, HIGH);
                                        } else {
                                                  lcd.clear();
                                                  lcd.setCursor(0, 0);
                                                  lcd.print("     Chyba!     ");
                                                  lcd.setCursor(0, 1);
                                                  lcd.print("  Zkus to znova ");
                                                  digitalWrite(relayPin, LOW);
                                                  delay(2000);
                                                  lcd.clear();
                                                  lcd.setCursor(0,0);
                                                  lcd.print("     Vitej!     ");
                                        }
                                        break;
                              default:
                                        inputCode[pos] = customKey;
                                        lcd.setCursor(pos,1);
                                        lcd.print(inputCode[pos]);
                                        pos ++;
                    }
          }
}
```

Krok 3: Zkompiluj kód
Krok 4: Nahraj sketch do Arduino
Nyní se na displeji I2C LCD1602, po zapnutí, objeví „Vítej!“. V tomto bodě zůstane indikátor LED ve vypnutém režimu. Pokud stiskneš tlačítko "*" klíče, objeví se „Zadej své heslo“. Pokud zadáš „123456“ a stiskneš “#” klíč pro potvrzení, rozsvítí se indikátor LED. Na displeji I2C LCD1602 se objeví: „Správně!“ „Vstup!“. O dvě vteřiny později se objeví „Vítej!“ na displeji I2C LCD1602. Pokud ale zadáš něco jiného, objeví se na displeji „Chyba!“ „Zkus to znova!“ a relé zůstane v původním stavu. O dvě vteřiny později se na displeji I2C LCD1602 objeví „Vítej!“.


Spravne!
Ustup