PŠP III

ARDUINO

ÚVOD

Milí studenti, následující text vznikl především jako doplňkový materiál k předmětu *Praktikum školních pokusů III*, kde je část cvičení vyhrazena seznámení s platformou *Arduino* a jejími základními možnostmi využití při výuce fyziky na středních školách. Jde především o základní měření, která lze realizovat se senzory ovládanými jednodeskovým počítačem Arduino. Takováto měření lze realizovat pomocí Arduina výrazně levněji, než by bylo možné při nákupu odpovídajících, komerčně vyráběných pomůcek, a pokud se učitel se studenty pustí i do programování desek, přirozeně se objeví mezipředmětové vztahy do matematiky a informatiky a u pokročilejších obvodů je nutné řešit i volbu vhodného algoritmu. Nevýhodou je ovšem velká časová náročnost (oproti alternativním, ale dražším platformám) a náchylnost desek a senzorů k hrubšímu zacházení, a především statické elektřině. I přes tyto nevýhody ale považujeme za přínosné studenty s platformou Arduino seznámit a naznačit směr, kterým by se mohli i při svém samostudiu ubírat (ať s Arduinem nebo s obdobnými platformami, jako je *Raspberry Pi*, *Micro:bit* apod.) a obohatit tak výuku fyziky na středních školách.

Tento text je rozdělen na dvě hlavní části. První část je zaměřena na úplné začátečníky ve světě Arduino, kde se seznámíte s deskou Arduino UNO, základy programování desky a zapojením základních obvodů s ledkami a mikrospínači. Druhá část textu je zaměřena na měření, kde si ukážeme senzory na měření veličin z různých částí středoškolské fyziky.

V textu se předpokládá, že máte k ruce níže uvedený seznam pomůcek a že máte zkušenosti se zapojováním obvodů. Nebudeme tedy vždy uvádět fotografie reálných komponent a nebudeme ani připomínat základy zapojování obvodů, jako je sériové a paralelní zapojení, způsob připojení voltmetru, nebezpečí zkratu apod.

SEZNAM POMŮCEK:

1) počítač s nainstalovaným Arduino IDE (https://www.arduino.cc/en/software)
2) deska Arduino UNO Rev3
3) kabel USB 2.0 A-B
4) nepájivé pole
5) sada propojovacích vodičů (např. DuPont)
6) elektroluminiscenční dioda RED (630 nm; 2,2 V; 20 mA)
7) elektroluminiscenční dioda YELLOW (590 nm; 2,1 V; 20 mA)
8) elektroluminiscenční dioda GREEN (570 nm; 2,2 V; 20 mA)
9) 3krát rezistor 220 Ω
10) rezistor 10 kΩ
11) mikrospínač

OBSAH

1. SEZNÁMENÍ S ARDUINEM
1.1. Deska Arduino UNO Rev3
1.2. Deska jako „obyčejný“ zdroj napětí
1.3. Deska jako „chytrý“ zdroj napětí
1.4. Příklady s blikající ledkou
1.5. Přepínání módů s mikrospínačem
1.6.

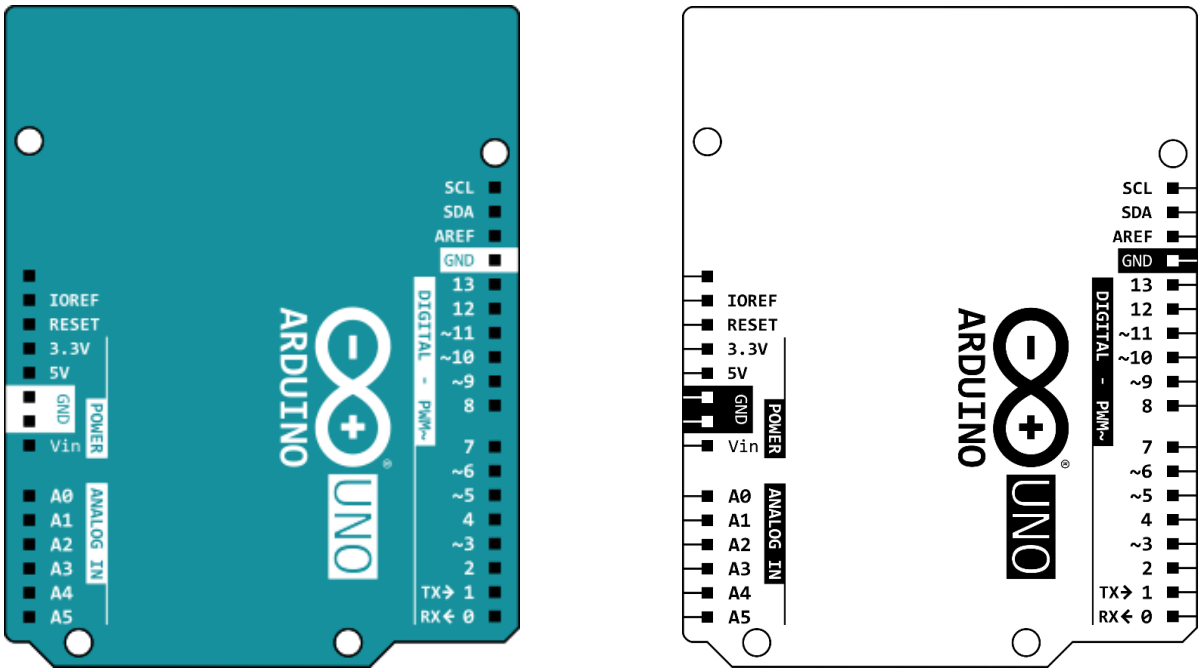
2. MĚŘENÍ S ARDUINEM
2.1. Elektrické napětí
2.2. Teplota
2.3.

1. SEZNÁMENÍ S ARDUINEM

V následujících pododdílech se pokusíme zjednodušeně naznačit, jak funguje deska *Arduino UNO Rev3*, seznámíme se s prostředím *Arduino IDE*, ve kterém budeme desku programovat a zkusíme deskou napájet elektrické obvody, anebo obráceně – snímat, co se v obvodech děje, a reagovat na to.

1.1. Deska Arduino UNO Rev3

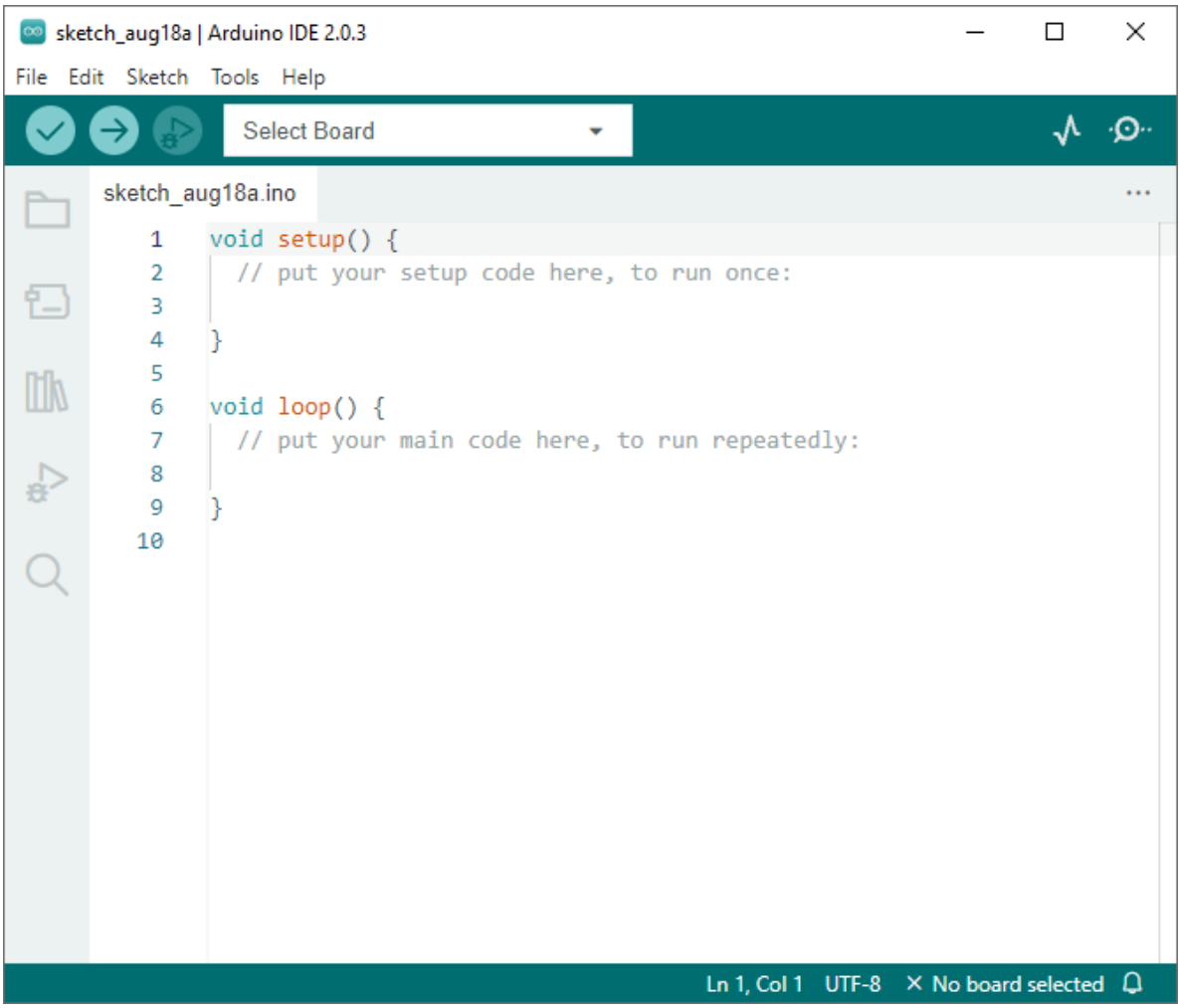
Na obrázku 1.1 je zjednodušené znázornění desky Arduino UNO Rev3, se kterou budeme nadále pracovat. Na levé části obrázku je deska vyobrazena ve věrných barvách, na pravé části černobíle, což budeme používat ve schématech elektrických obvodů. Na obrázku jsme zachovali především analogové, digitální a další piny (po stranách desky), ke kterým budeme připojovat naše obvody. Ostatní hardware zde vyobrazen není, ale za zmínku stojí především USB konektor, kterým budeme desku pomocí USB A-B kabelu připojovat k počítači nebo k powerbance, a tlačítko RESET (vedle USB konektoru), kterým můžeme resetovat běžící sketch (termín používaný pro program, který běží na desce).



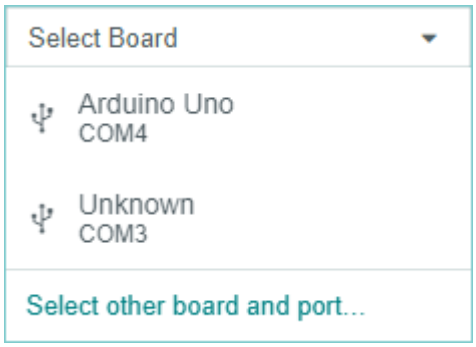
OBRÁZEK 1.1: Zjednodušené znázornění desky Arduino UNO Rev3

Ze začátku budeme používat především digitální piny, které mohou nabývat pouze dvou diskrétních hodnot – 1 nebo 0, resp. TRUE nebo FALSE, resp. HIGH nebo LOW. Oproti tomu analogové piny budou moci nabývat, zjednodušeně řečeno, „spojitých“ hodnot (později bude uvedeno na pravou míru). Digitální piny se tedy budou hodit pro zapínání či vypínání obvodu, resp. pro detekci, jestli je něco zapnuté nebo vypnuté. Analogové piny se budou hodit například pro měření, kdy nás nebude zajímat pouze informace o tom, že je někde napětí (resp. že je něco zapnuté), ale budeme chtít znát i konkrétní hodnotu daného napětí.

Máme-li před sebou desku a počítač s nainstalovaným programem Arduino IDE, můžeme rovnou vyzkoušet, jestli nám deska s počítačem komunikuje. Připojme desku USB A-B kabelem k počítači a spusťme Arduino IDE. Mělo by se objevit rozhraní jako na obrázku 1.2 (může se lišit podle nainstalované verze). Komunikuje-li deska s počítačem, mělo by být možné vybrat desku Arduino UNO z rozbalovací nabídky „Select Board“ (viz obrázek 1.3). V případě, že desku nevidíte, zkuste jiný USB port, nebo bývá častou příčinou chybné komunikace desky a počítače USB kabel, kdy je potřeba jej vyměnit.



OBRÁZEK 1.2: Rozhraní Arduino IDE



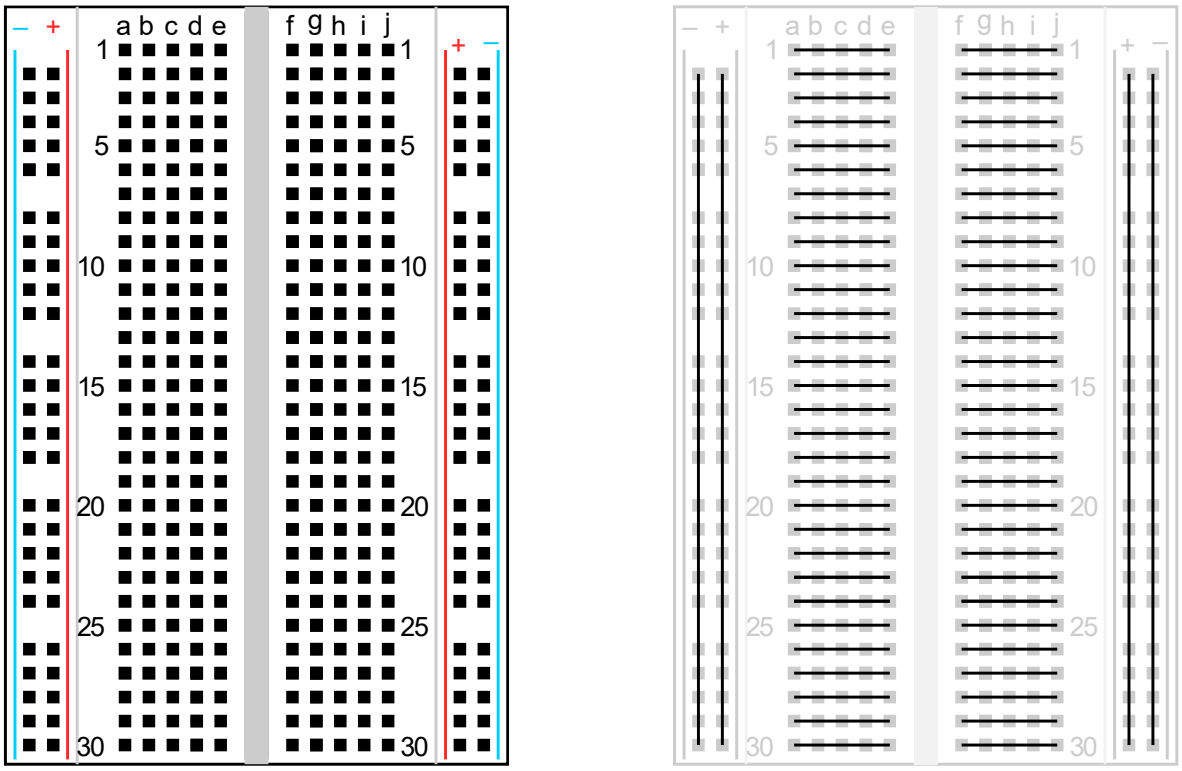
OBRÁZEK 1.3: Rozbalovací nabídka připojených desek k počítači v programu Arduino IDE

1.2. Deska jako „obyčejný“ zdroj napětí

Máme-li desku připojenou k počítači, je napájena 5 V z USB portu a sama se stává 5V zdrojem napětí – toho využijeme v prvním obvodu, kdy desku použijeme jen jako obyčejný zdroj stejnosměrného

napětí. K tomu budeme ještě potřebovat nepájivé pole, elektroluminiscenční diodu (např. RED 630 nm; 2,2 V; 20 mA), 220Ω rezistor a propojovací vodiče.

Nepájivé pole je pomůcka, která nám umožní rychlé zapojování součástek do obvodů, aniž by bylo nutné pájet nebo používat krokosvorky. Na druhou stranu se ale může nepájivé pole ze začátku zdát nepřehledné a komplikované. Budeme muset být opatrní, abychom nezpůsobili zkrat. Na obrázku 1.4 je znázorněno běžné nepájivé pole s řadami pinů (dostupná pole se mohou mírně lišit, ale principiálně by měla fungovat stejným způsobem). Běžně bývá pole rozděleno na dvě symetrické poloviny, přičemž se každá skládá ze dvou sloupců pinů označených + a - a z pole pinů se sloupci rozlišenými pomocí písmen a řádky rozlišenými pomocí čísel. Na obrázku 1.4 vpravo je znázorněno, jak jsou jednotlivé piny propojené. Jeden z pinů sloupce + a - bývá obyčejně připojen ke zdroji napětí, a tudíž se celý sloupec + chová jako kladný pól zdroje a sloupec - jako záporný pól. Součástky se potom typicky zapojují k pinům rozlišeným písmeny a čísly. Je ale nutné, aby byla každá součástka připojena svými vývody k různým řádkům. Připojíme-li například rezistor oběma jeho vývody do jednoho řádku, byl by zkratován.



OBRÁZEK 1.4: Nepájivé pole

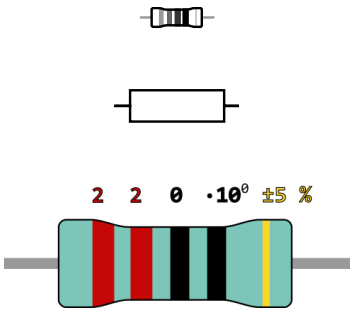
Na obrázku 1.5 je porovnání schématické značky LEDky a jejího relativně věrného vyobrazení. Důležité pro nás bude, že dioda propouští elektrický proud pouze v jednom směru. Na obrázku je vyznačeno, jakým způsobem má být dioda připojena ke zdroji. Kladný pól zdroje by měl být přiveden na anodu, která bývá delší než katoda. Případně se lze orientovat i podle pouzdra, ve kterém je dioda uzavřena – prstýnek ve spodní části pouzdra bývá u katody zploštělý, resp. seříznutý.

Na obrázku 1.6 je rezistor s jedním příkladem, jak lze pomocí barev značit jeho odpor. Pro rychlejší orientaci ale doporučujeme používat multimetr s ohmmetrem, kterým si lze snadno odpor daného rezistoru rychle změřit – je to jistější než orientace podle barev, které mohou být obtížně rozpoznatelné.

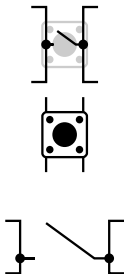
Přehled součástek doplníme i o mikrospínač, i když ho v prvním zapojení nebudeme potřebovat. Mikrospínač je znázorněn na obrázku 1.7. Uvedená schematická značka se mírně odlišuje od běžného spínače, a proto je na obrázku porovnání schématické značky mikrospínače a jeho věrného znázornění.

Z obrázku je patrné, že vývody mikrospínače jsou vždy po dvojicích propojené a budeme muset být pozorní, jakým způsobem mikrospínač zapojujeme, aby vůbec jako spínač fungoval.

OBRÁZEK 1.5: Elektroluminiscenční dioda



OBRÁZEK 1.6: Rezistor

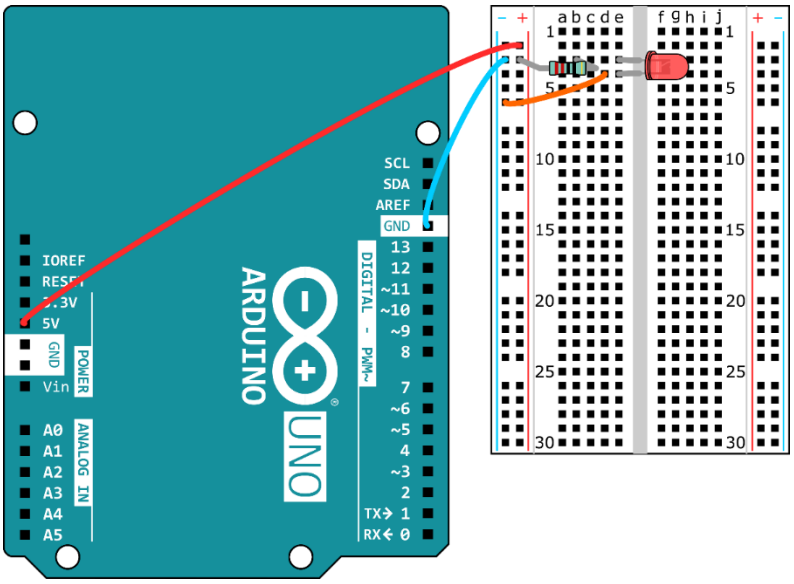


OBRÁZEK 1.7: Mikrospínač

Nyní zapojíme obvod jako na obrázku 1.8 a rozsvítíme červenou ledku. Pin 5V zapojte jako poslední, až si budete jisti, že máte obvod správně sestavený. Obvod samotný lze zapojit mnoha různými způsoby. Není vyloženě nutné využívat sloupců + a -, ale protože se běžně jako „zdrojové“ používají, využili jsme je i v ukázkovém zapojení. Také nezáleží na pořadí rezistoru a ledky.

Všimněme si, že ledka je připojena na obrázku 1.8 ke dvěma různým řádkům nepájivého pole, konkrétně k řádku 3 a řádku 4. Kdybychom ledku zapojili oběma vývody do jediného řádku, byla by nepájivým polem zkratována. Rezistor je v našem zapojení v jednom řádku, ale zkratován není, protože je sloupec + od ostatních řádků pole oddělený (viz obrázek 1.4).

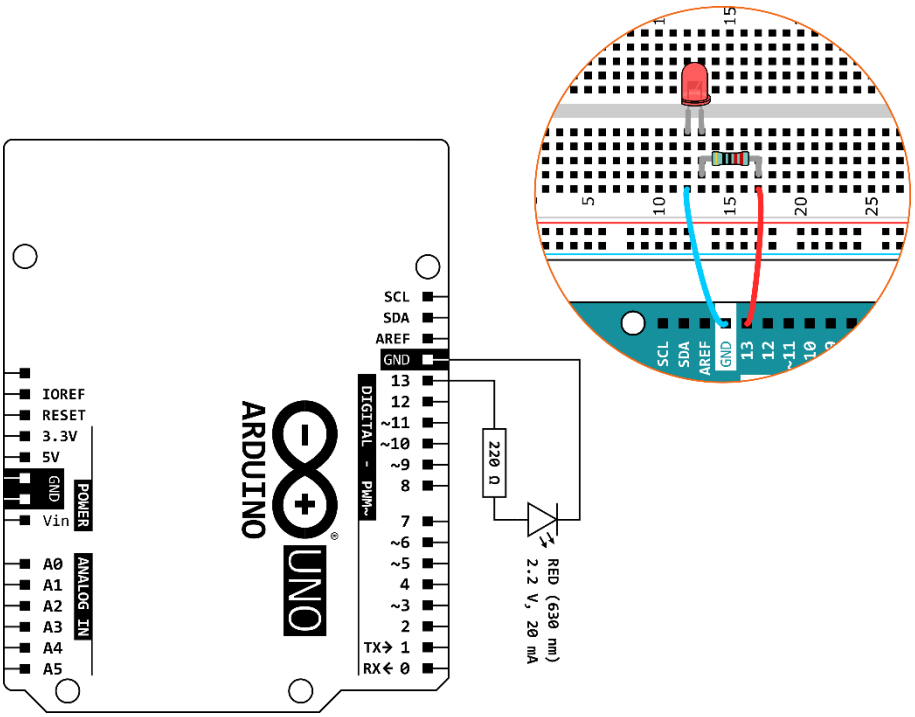
Je-li deska Arduino UNO připojena ke zdroji a obvod je správně zapojený, měla by ledka nepřerušovaně svítit. Kdybychom na ní změřili napětí a procházející proud, měli bychom získat hodnoty přibližně 1.9 V a 12 mA. Jestliže dioda nesvítí, je možné, že je zapojena v závěrném směru.



OBRÁZEK 1.7: Obvod pro rozsvícení červené ledky

1.3. Deska jako „chytrý“ zdroj napětí

Nyní se pokusíme využít potenciálu desky více a necháme ledku blikat (nejenom „hloupě“ svítit). Předěláme obvod tak, aby odpovídal schématu na obrázku 1.8, kde je i pro ukázku zobrazena jedna možnost reálného zapojení.



OBRÁZEK 1.8: Obvod pro rozblikání červené ledky

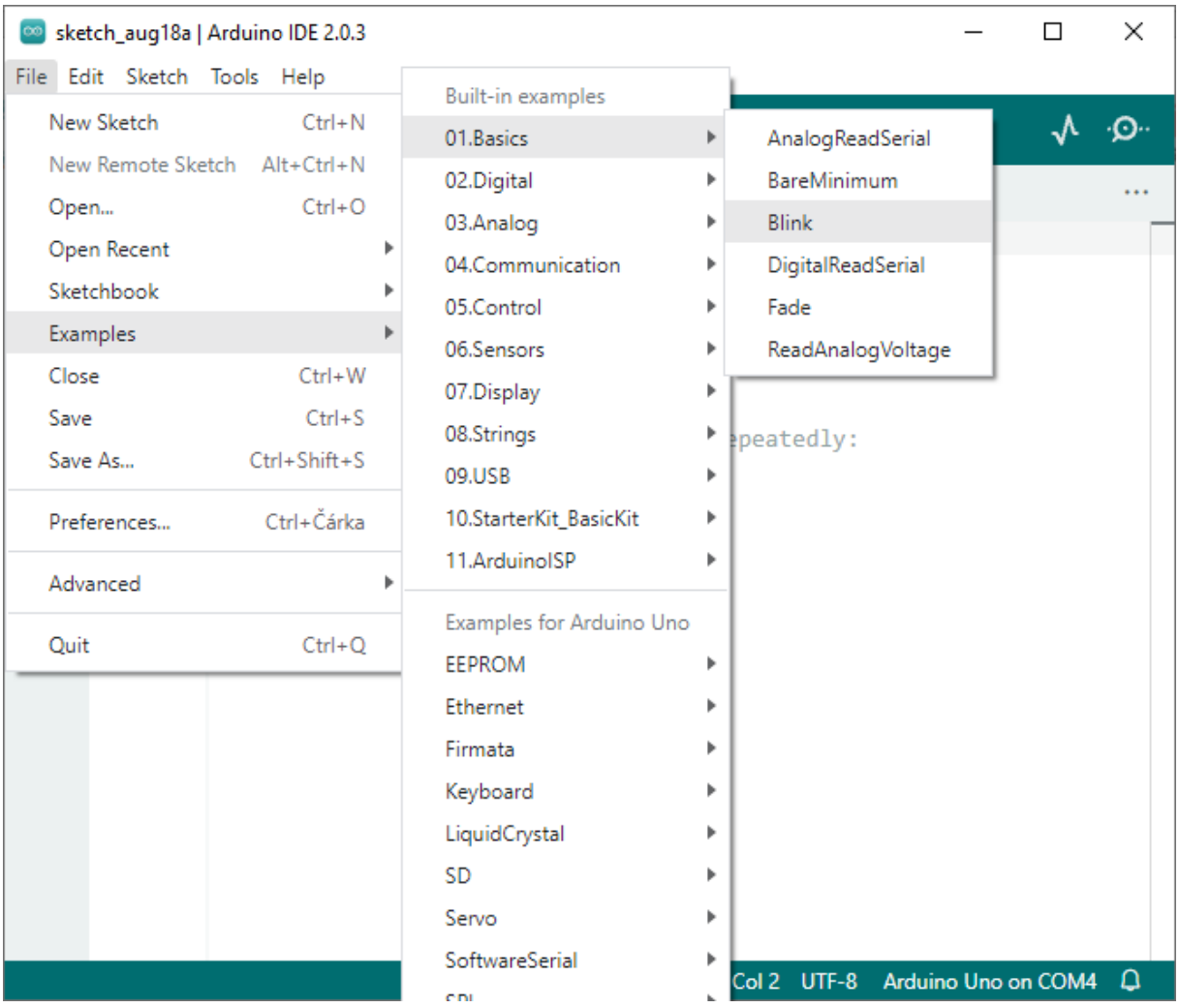
Spustíme Arduino IDE a podrobněji si vysvětlíme, jak fungují Arduino-sketche. Jak jsme si mohli všimnou již dříve (viz obrázek 1.2), Arduino IDE nám hned nabízí dvě hlavní funkce, void setup a void

loop, ze kterých se bude skládat každý program, který napíšeme. Příkazy, které napíšeme v rámci funkce void setup, se provedou jednou po spuštění programu a následně budou opakovaně v nekonečné smyčce probíhat příkazy, které napíšeme v rámci funkce void loop. Stejnou informaci uvádějí i komentáře za dvojitým lomítkem //. Vše, co napíšeme za // jsou pouze naše pomocné poznámky pro lepší orientaci v kódu, ale na vlastní program nemají vliv (při kompilaci programu se vše za // vymaže).

Kompilaci programu a ověření, že v něm nejsou syntaktické chyby budeme provádět tlačítkem se symbolem ✓ v levém horním rohu prostředí. Když bude vše v pořádku, nahrajeme kód do desky tlačítkem se symbolem →. Není vyloženě nutné nejdříve program kontrolovat tlačítkem ✓ a program lze rovnou nahrát.

Máme-li vybránu desku a port, můžeme vyzkoušet předpřipravený příklad. Otevřete program *Blink* v kartě *File–Examples–01.Basics*, viz obrázek 1.9. Proběhne-li následné nahrání kódu do desky v pořádku, měla by začít červená ledka blikat s dvousekundovou periodou (jednu sekundu bude svítit, jednu sekundu bude vypnutá).

Všimněme si také, že bliká i vestavěná ledka přímo do desky Arduino UNO (je označená písmenem L a je situována vedle digitálního pinu 13).



OBRÁZEK 1.9: Spuštění sketche *Blink*

Nyní si projdeme celý kód, abychom se naučili první příkazy, kterými budeme desku Arduino UNO programovat. Níže je kód programu Blink uvedený a pracuje se v něm právě s vestavěnou ledkou L. Zkusíme tedy i vysvětlit, proč nám bliká i naše červená dioda.

Blink | Arduino IDE 2.0.3 | deska Arduino UNO Rev3

```
1
2  void setup() {
3    pinMode(LED_BUILTIN, OUTPUT);
4  }
5
6  void loop() {
7    digitalWrite(LED_BUILTIN, HIGH);
8    delay(1000);
9    digitalWrite(LED_BUILTIN, LOW);
10   delay(1000);
11 }
12
```

Jak je vidět z kódu, pro vestavěnou ledku má Arduino speciální označení LED_BUILTIN. Tato ledka je ale také "na pevno" spojena s pinem 13 a přepíšeme-li v kódu všechna LED_BUILTIN za 13 a tento nový kód nahrajeme do desky, měly by obě ledky (vestavěná i červená) blikat stále stejným způsobem.

Kód z programu Blink tedy pracuje s digitálním pinem 13 a nastavuje ho jako *výstup* (resp. OUTPUT). Tím říkáme desce, že pin 13 má fungovat jako zdroj a jednu sekundu být v podstatě kladným pólem 5V zdroje a jednu sekundu být vypnutý.

Vezmeme-li příkazy popořadě, v řádku 3 je příkaz:

```
pinMode(LED_BUILTIN, OUTPUT),
```

kterým na desce nastavujeme funkci digitálního pinu. Příkaz pracuje se dvěma argumenty – prvním říkáme, jakého pinu (LED_BUILTIN je označení pinu 13) se má příkaz týkat; druhým definujeme vybraný pin jako *výstup* (OUTPUT), anebo *vstup* (INPUT). Výstup deska "chápe" tak, že má být daný pin kladným pólem zdroje, tj. má z něj vystupovat proud. Dalšími příkazy potom můžeme ovlivňovat, kdy má být tento zdroj zapnutý a kdy naopak vypnutý. Bude-li daný pin vstupem, bude reagovat na napětí, které se něm objeví (jako by deska byla voltmetrem se zdířkami GND a příslušným digitálním pinem). Na použití pinu jako vstupu se podíváme později, prozatím si vystačíme pouze s výstupy. Příkaz pinMode si můžeme shrnout následovně:

```
pinMode(označení pinu, INPUT/OUTPUT).
```

V řádku 7 máme příkaz:

```
digitalWrite(LED_BUILTIN, HIGH).
```

Když už deska ví, že má být pin 13 výstupem (resp. zdrojem), můžeme příkazem digitalWrite pin "zapínat", anebo "vypínat" (viz příkaz v řádku 9). Opět pracujeme se dvěma argumenty, přičemž první je opět označení pinu, kterého se má příkaz týkat. Druhým argumentem je v podstatě *bit*, tj. může nabývat pouze dvou hodnot – např. HIGH nebo LOW, 1 nebo 0, true nebo false. Argument HIGH, 1 a true deska interpretuje tak, že má pin "zapnout", argumentem LOW, 0 a false pin "vypne". Můžeme zkusit v programu Blink přepsat argumenty v příkazech digitalWrite z HIGH a LOW na 1 a 0 nebo true a false a program opět nahrát – měl by fungovat pořád stejným způsobem. Příkaz digitalWrite tedy shrneme následovně:

```
digitalWrite(označení pinu, HIGH/LOW nebo 1/0 nebo true/false).
```

Poslední příkaz z programu Blink, který jsme ještě nerozebrali, je delay:

```
delay(1000),
```

který je sám o sobě relativně intuitivní. Znamená, že se má program zastavit na dobu, která je uvedena v argumentu příkazu. Defaultně příkaz delay počítá s milisekundami, tj. $1\,000$ v argumentu znamená $1\,000\ \mathrm{ms} = 1\ \mathrm{s}$. Teprve po uplynutí zadané doby se začnou vykonávat příkazy, které za příkazem delay následují. Shrnout to tedy lze následovně:

```
delay(doba čekání než se přejde na další příkazy).
```

Nakonec ještě upozorněme, že každý příkaz musí být ukončený středníkem, tj. symbolem ;. Také je nutné rozlišovat velká a malá písmena – syntaxe je na ně citlivá. A příkazy, které tvoří funkci (zatím jenom void setup nebo void loop) musí být uzavřeny ve složených závorkách, tj. { a }.

Celý program Blink tedy můžeme popsat následovně:

```
// příkazy, které se vykonají jednou na začátku programu
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);      // nastav pin 13, resp. diodu
                                     L, jako výstup
} // konec jednorázových příkazů

// příkazy, které se budou vykonávat pořád dokola v nekonečné smyčce
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // zapni pin 13, resp. diodu
                                     L (voltage high)
  delay(1000);                       // počkej 1 000 ms, tj. 1 s
  digitalWrite(LED_BUILTIN, LOW);    // vypni pin 13, resp. diodu
                                     L (voltage low)
  delay(1000);                       // počkej 1 000 ms, tj. 1 s
} // konec smyčky, vrať se na její začátek
```

Také, když nyní rozumíme jednotlivým příkazům, můžeme změnit způsob, jakým dioda připojená k pinu 13 (dioda L) bliká. Můžeme zkusit periodu blikaní zkrátit a dobu svícení a nesvícení zadat nesymetricky. Nastavme dobu svícení na třičtvrtě sekundy ($750\ \mathrm{ms}$) a dobu nesvícení na čtvrt sekundy ($250\ \mathrm{ms}$). Po nahrání nového kódu bychom měli pozorovat změnu ve způsobu blikání.

Než přejdeme dále, upozorněme ještě na to, že perioda blikání neodpovídá přesně součtu času, které jsme zadali do příkazů delay. Všechny příkazy totiž potřebují na vykonání nějaký čas, i když třeba velmi krátký. Takže perioda blikání je ve skutečnosti delší – prodlužuje se o čas nutný na vykonání příkazů digitalWrite, o návrat na začátek smyčky apod.

1.4. Příklady s blikající ledkou

Níže je uvedeno několik příkladů, které lze realizovat s ledkou, resp. ledkami.

Jako první je příklad s vysláním zprávy SOS pomocí Morseovy abecedy. V ukázkovém kódu vysíláme v nekonečné smyčce. Pokud bychom chtěli vyslat signál pouze jednou, stačí veškerý obsah funkce void loop přesunout do funkce void setup pod příkaz pinMode na řádku 3 (ten nesmíme přepsat). Protože budeme vysílat “SOS” pouze jednou, můžeme nyní poslední příkaz delay smazat (není nutné oddělovat smyčky).

SOS | Arduino IDE 2.0.3 | deska Arduino UNO Rev3

```
1
2  void setup() {
3    pinMode(13, OUTPUT);
4  }
5
6  void loop() {
7    digitalWrite(13, HIGH); // písmeno S
8    delay(500);             // tečka
9    digitalWrite(13, LOW);
10   delay(500);
11   digitalWrite(13, HIGH);
12   delay(500);             // tečka
13   digitalWrite(13, LOW);
14   delay(500);
15   digitalWrite(13, HIGH);
16   delay(500);             // tečka
17   digitalWrite(13, LOW);
18
19   delay(2000);  // lomítko
20
21   digitalWrite(13, HIGH); // písmeno O
22   delay(1000);            // čárka
23   digitalWrite(13, LOW);
24   delay(500);
25   digitalWrite(13, HIGH);
26   delay(1000);            // čárka
27   digitalWrite(13, LOW);
28   delay(500);
29   digitalWrite(13, HIGH);
30   delay(1000);            // čárka
31   digitalWrite(13, LOW);
32
33   delay(2000);  // lomítko
34
35   digitalWrite(13, HIGH); // písmeno S
36   delay(500);             // tečka
37   digitalWrite(13, LOW);
38   delay(500);
39   digitalWrite(13, HIGH);
40   delay(500);             // tečka
41   digitalWrite(13, LOW);
42   delay(500);
43   digitalWrite(13, HIGH);
44   delay(500);             // tečka
45   digitalWrite(13, LOW);
46
47   delay(5000);  // prodleva, před opětovným vysláním zprávy
```

```
48  }
49
```

Jako druhý příklad uvedeme semafor, jehož schéma je na obrázku 1.10. Nyní budeme pracovat s více digitálními piny naráz, a abychom se v nich lépe orientovali, není nutné je označovat jejich číslem, ale můžeme si je na začátku programu přejmenovat. Např. pin 13 můžeme označovat i LED_BUILTIN.

Pojmenování pinu je v podstatě pojmenování určité konstanty – konstanta 13 je defaultně pojmenována LED_BUILTIN. Konstantu si můžeme pojmenovat dvěma způsoby. První způsob je pomocí komponenty #define z programovacího jazyka C++. Uvažujme, že budeme červenou LEDku na semaforu ovládat pinem 12, potom si můžeme pin 12 pojmenovat konstantou RED následujícím způsobem:

```
#define RED 12.
```

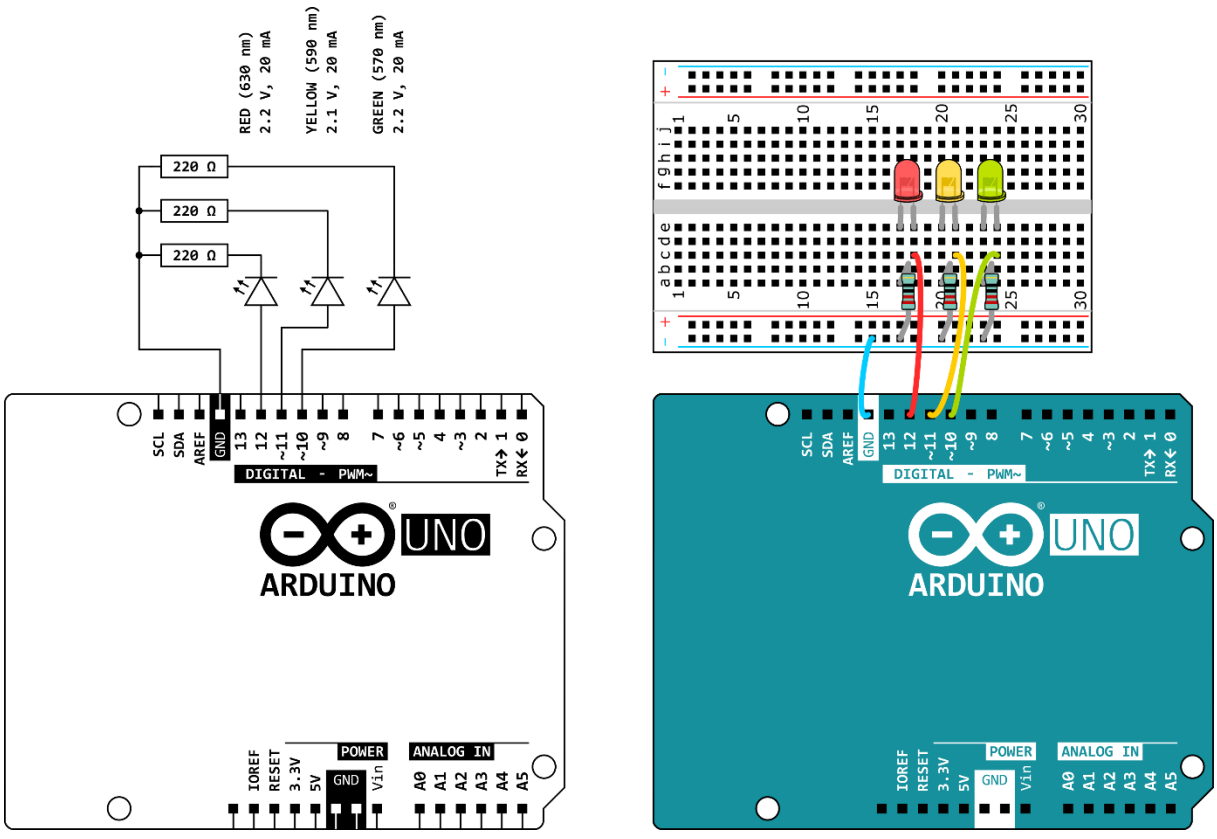
Potom můžeme v celém programu místo 12 psát RED a při programování si nemůžeme splést pin, kterým budeme ovládat červenou. Všechna RED se v programu sama přepíšou na 12 při kompilaci. Nakonec ještě upozorněme, že se na konci řádku s #define se nepíše středník. Syntaxe je tedy:

```
#define název konstanty hodnota konstanty.
```

Druhý způsob definování konstanty je pomocí klíčového slova const. Tento způsob si vysvětlíme, až se budeme zabývat proměnnými. Na ukázku si ale uveďme, jak bychom pin 12 pojmenovali RED:

```
const int RED = 12;
```

(tentokrát je ukončení středníkem nutné a také znaménko =).

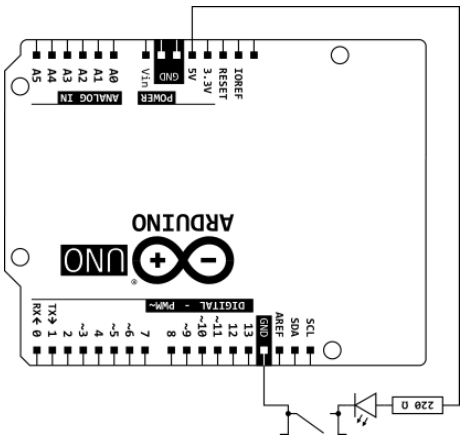


OBRÁZEK 1. 10: Semafor

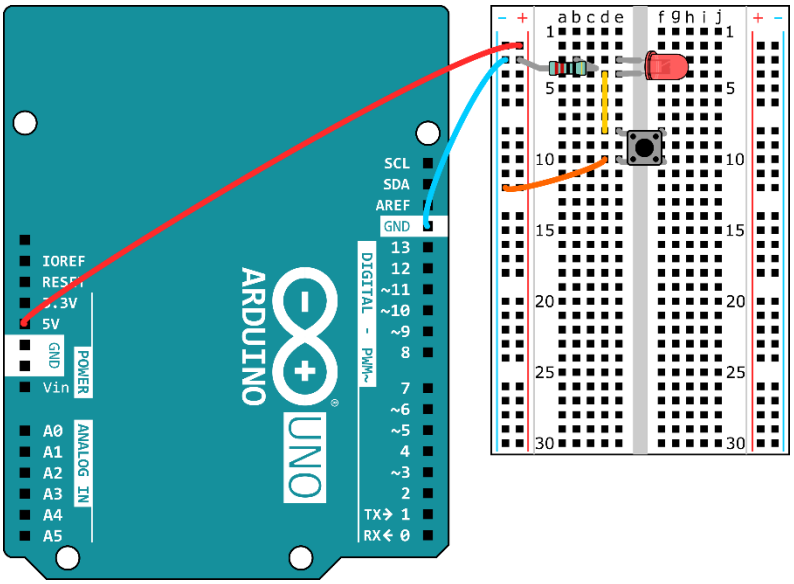
Semafor | Arduino IDE 2.0.3 | deska Arduino UNO Rev3

```
#define RED 12
#define YELLOW 11
#define GREEN 10

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

void loop() {
  digitalWrite(RED, HIGH);
  delay(10000);

  digitalWrite(YELLOW, HIGH);
  delay(1000);

  digitalWrite(RED, LOW);
  digitalWrite(YELLOW, LOW);
  digitalWrite(GREEN, HIGH);
  delay(10000);

  digitalWrite(GREEN, LOW);
  digitalWrite(YELLOW, HIGH);
  delay(2000);

  digitalWrite(YELLOW, LOW);
}

```

V následujícím příkladu použijeme k rozsvícení ledky mikrospínač. Opět použijeme desku pouze jako 5V baterii, není tedy třeba nic programovat. Schéma je na obrázku 1.11. Jelikož je první zapojování s mikrospínačem často problematické, na obrázku 1.12 je možné reálné zapojení.



OBRÁZEK 1.11: Schéma k rozsvícení ledky mikrospínačem



OBRÁZEK 1. 12: Zapojení mikrospínače

1.5. Přepínání módů mikrospínačem

1.6.

2. MĚŘENÍ S ARDUINEM

2.1. Elektrické napětí

2.2. Teplota