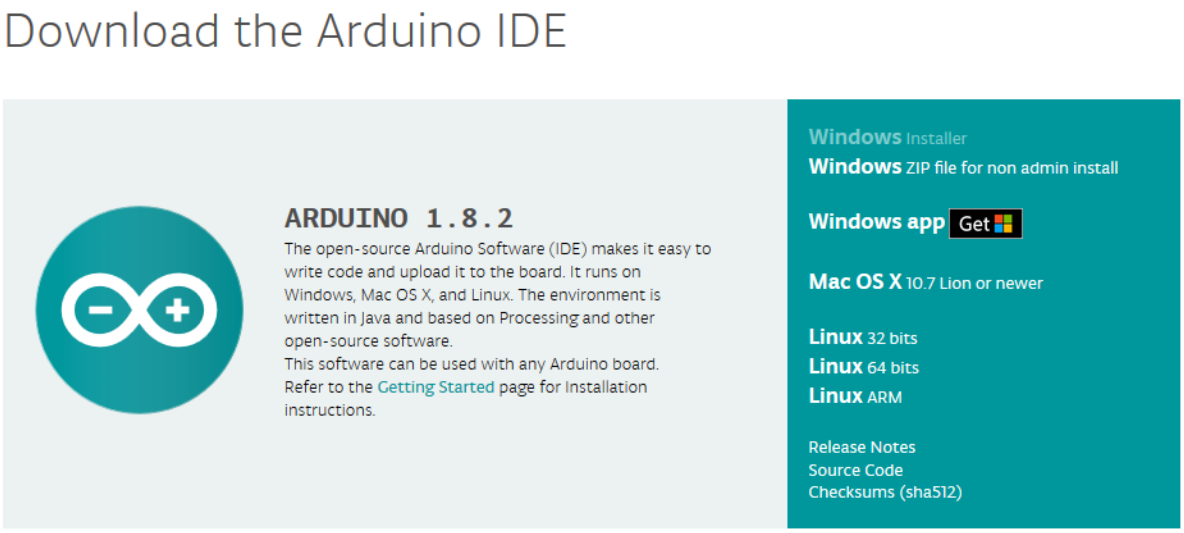
S-arduino: návod

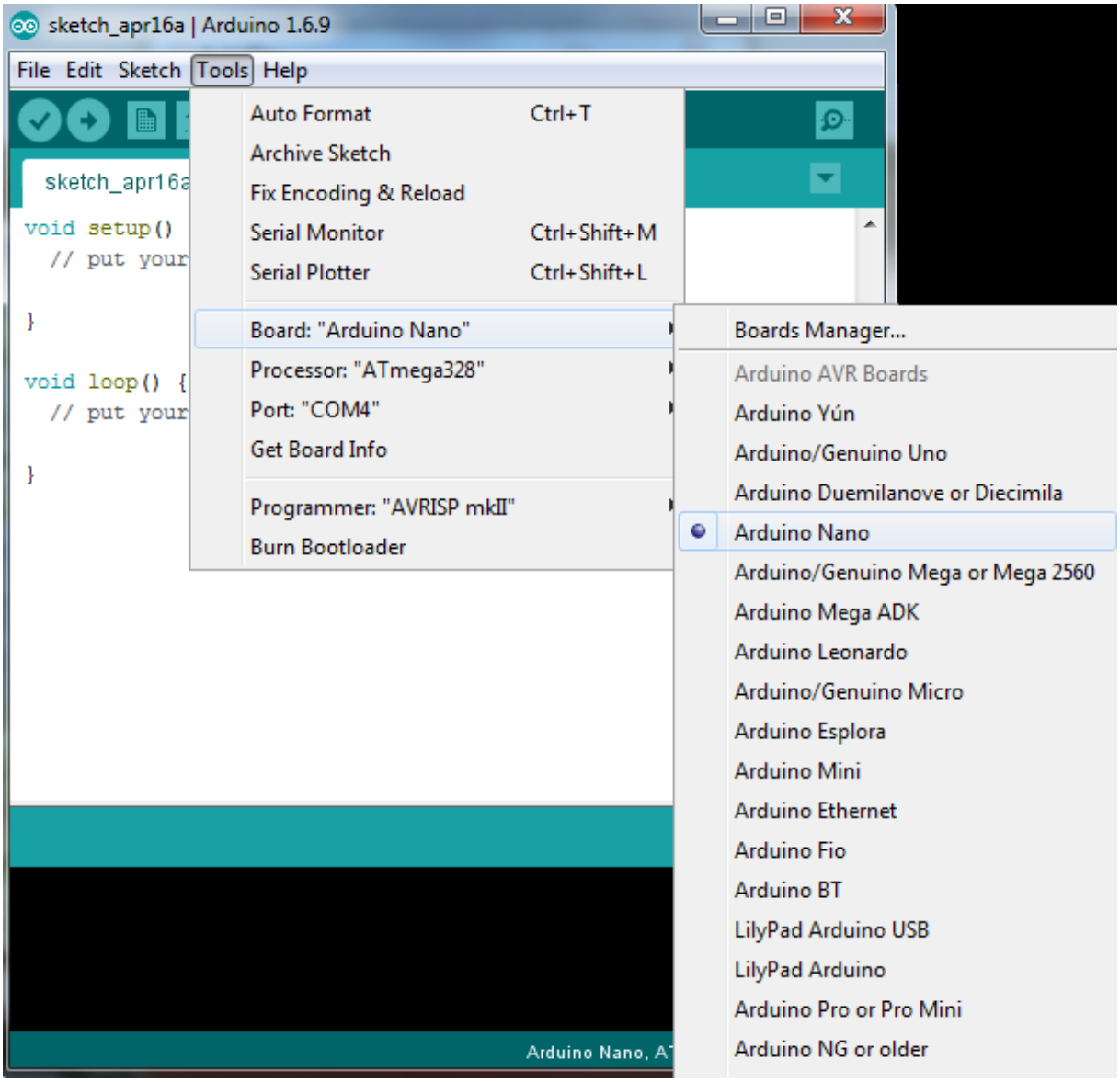
Na začátku je potřeba na počítač nainstalovat vývojové prostředí pro psaní programů. To nalezneme na webové adrese (platná při psaní návodu)

https://www.arduino.cc/en/Main/Software

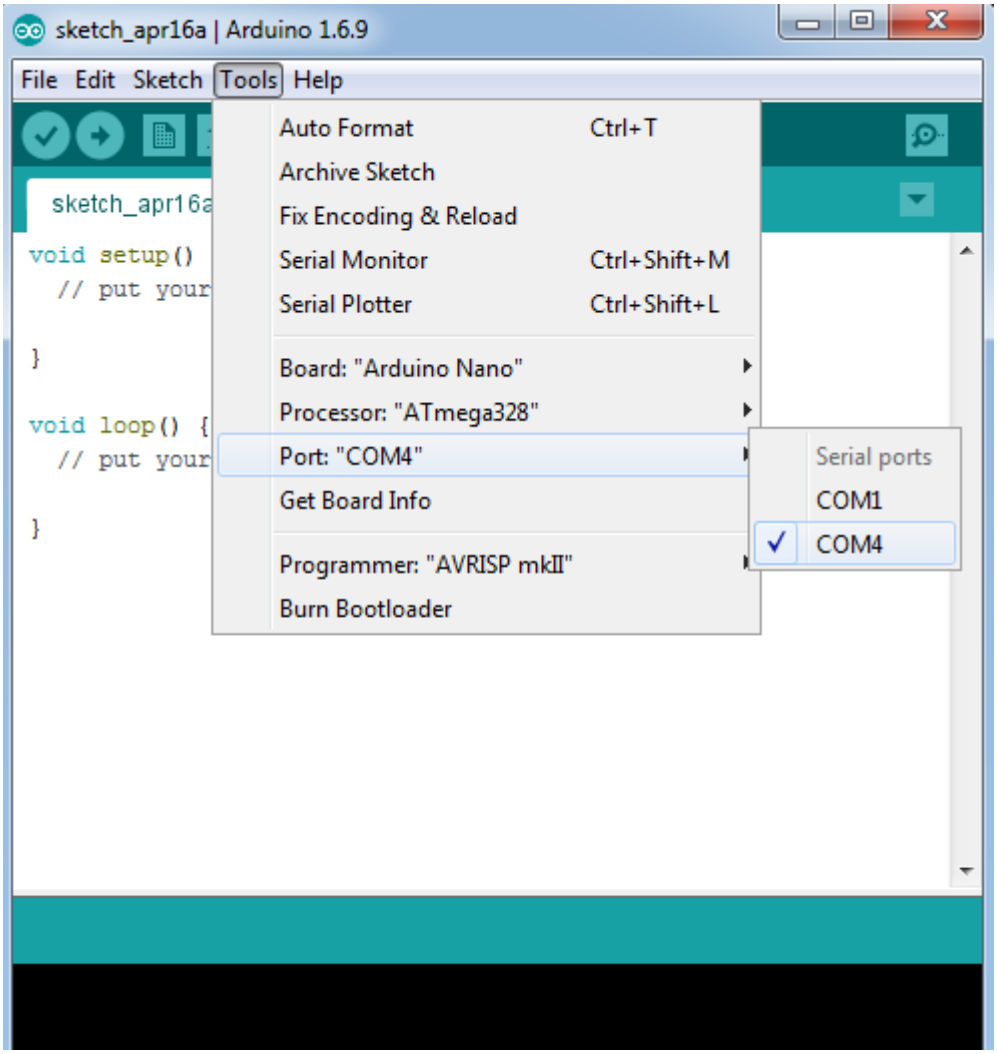
Kde si podle operačního systému stáhneme potřebný software



Když software nainstalujeme, je třeba připojit stavebnici S-arduino do USB. Pokud máte stavebnici Saimon1 + Saimon4, je potřeba připojit Saimon1 do USB a ještě připojit zdířku **NAPÁJENÍ na log.1. Zdířka je na modulu Saimon4**. Software spustíme a v nabídce TOOLS zvolíme "Board:Arduino Nano"



Jako druhou věc zvolíme správný port, na kterém je arduino připojené. V mém případě port "Port:COM4". Ve vašem počítači bude pravděpodobně jiné číslo.



A můžeme začít psát programy.
Poznámka: Pokud vidíte pouze COM1, tak je problém s připojením arduina k počítači. Mám vyzkoušené, že ve Windows XP to funguje blbě, ve vyšších verzích by neměl být problém.

Pojďme na úlohy. Chci ještě říct, že toto není učebnice programování v jazyce C, ale návod, jak si hrát se stavebnicí S-arduino. Takže uvádím jen málo z toho, co programování opravdu nabízí. Dále se nebudu držet přesných názvů a programátorské terminologie. Návod jsem napsal tak, aby si majitel stavebnice mohl hrát :-).

Úloha 1. Blikající LED

V první úloze se seznámíme s tím, jak vypadá program pro arduino.
Má dvě základní části - dvě funkce. SETUP a LOOP. Tělo funkce, tedy prostor, kam píšeme názvy jiných funkcí (příkazů), začíná vždy složenou závorkou otevřenou, tedy {, a končí složenou závorkou uzavřenou, tedy }. Do funkce SETUP se píší příkazy, které se v programu mají provést jen jednou. Funkce LOOP opakuje všechny příkazy stále dokolečka. Ukážeme si první tři - **pinMode(), digitalWrite(), delay()** - a řekneme si, co dělají a jak se používají.
Program pro arduino může vypadat třeba takto.

```
void setup() {
  // Příkazy zde se spustí jen jednou (různá nastavení a tak)
  pinMode (2,OUTPUT); //Nastavíme zdířku D02 jako výstup. To je nutné.
}

void loop() {
  // Příkazy zde se budou opakovat pořád dokolečka
  digitalWrite (2,1);     //Zapíšeme na zdířku D02 log 1. Tím rozsvítíme LEDku
  delay (1000);           //Tohle zastaví celý program na 1s (tedy 1000 milisekund)
  digitalWrite (2,0);     //Zapíšeme na zdířku D02 log 0. Tím zhasneme LEDku
  delay (1000);           //Tohle zastaví celý program na 1s (tedy 1000 milisekund)

}
```

Zapojení drátků na Saimon: D02-93 , 92-log.0
Zapojení drátků na S-arduino: drátek ze zdířky D02 do LED1. Zkráceně budeme psát D02-LED1.

Funkce **pinMode** se používá pro nastavení zdířek. Když napíšeme **pinMode (2,OUTPUT)**, tak se zdířka D02 nastaví jako **výstup**. To je nutné, když na ni připojujeme LEDku. Funkce **delay (1000)** znamená příkaz čekej 1000 milisekund, tedy 1 sekundu. Funkce **digitalWrite je** příkaz, který zapíše na zvolenou zdířku logickou 1 nebo logickou 0.
Tedy v programu dvojka v závorce za příkazem **digitalWrite** znamená zdířku, na kterou se má zapsat. A jednička znamená, že zapíšeme logickou 1. Logická 1 pro nás znamená, že se na zdířce objeví napětí, a pokud na ni bude správně zapojená LEDka, tak se rozsvítí. Takže program nám bude dělat to, že LED1 bude pravidelně blikat.

Ještě chci napsat, že za každou funkcí (příkazem) musí být **středník**. Nejčastější chyba všech dětí na kroužku :). A to, co je napsáno za znaky **//,** tomu se říká **komentář**. Pro počítač to není důležité, nemusí se to pro správnou funkci programu vůbec psát, ale pro člověka je to velmi dobré, protože se v kódu dobře orientuje. Příkazy, které jsou důležité, jsem zvýraznil šedivě.

Úloha 2: Rozblikejte postupně všechny LED. Použijte zdířky D03, D04, D05 a LED2, LED3, LED4. Opište a upravte kód tak, aby blikaly postupně všechny.

Řešení:

```
void setup() { //ve funkci setup() se všechny příkazy provedou jen jednou
  pinMode (2,OUTPUT);
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);

}

void loop() { // ve funkci loop () se příkazy provádějí stále dokola, bude to stále blikat
  digitalWrite (2,1);
  delay (1000);
  digitalWrite (2,0);
  delay (1000);
  digitalWrite (3,1);
  delay (1000);
  digitalWrite (3,0);
  delay (1000);
  digitalWrite (4,1);
  delay (1000);
  digitalWrite (4,0);
  delay (1000);
  digitalWrite (5,1);
  delay (1000);
  digitalWrite (5,0);

}
```

Zapojení na Saimon: D02-93 , D03-95 , D04-97, D05-99, 92-log0, 94-log0 , 96-log0, 98-log0
Zapojení na S-arduino: D02-LED1 , D03-LED2 , D04-LED3 , D04-LED4

Úloha 3: Proměnná
Dále si ukážeme, že do parametrů funkcí nemusíme psát jen číslo. Můžeme tam psát i proměnnou. Co to je a jaké jsou její výhody si ukážeme na příkladu.

```
int A; //takhle řekneme počítači, že A je proměnná. Tomu se říká deklarace proměnné
void setup() {
  // Kód zde se spustí jen jednou (různá nastavení a tak)
  pinMode (2,OUTPUT); //Nastavíme zdířky D02, D03, D04, D05 jako výstupy
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);
}

void loop() {
  // Hlavní kód, to co je zde se bude opakovat pořád dokolečka
  A=2; //Tady sdělíme počítači, že chceme, aby se A rovnalo číslu 2.
        //Tomu se říká inicializace hodnoty proměnné
  digitalWrite (A,0); //Zapíšeme na zdířku D02 log 0.
  A=3; //Tady sdělíme počítači, že chceme, aby se A rovnalo číslu 3
  digitalWrite (A,1); //Zapíšeme na zdířku D03 log 1.
  A=4; //Tady sdělíme počítači, že chceme, aby se A rovnalo číslu 4
  digitalWrite (A,0); //Zapíšeme na zdířku D04 log 0.
  A=5; //Tady sdělíme počítači, že chceme, aby se A rovnalo číslu 5
  digitalWrite (A,1); //Zapíšeme na zdířku D05 log 1.

}
```

Zapojení na Saimon: D02-93 , D03-95 , D04-97, D05-99, 92-log0, 94-log0 , 96-log0, 98-log0
Zapojení na S-arduino: D02-LED1 , D03-LED2 , D04-LED3 , D04-LED4

Proměnná je tedy písmeno, nebo slovo.
Například to může vypadat i takto.

```
int LEDka;
void setup ()
{
  pinMode (2,OUTPUT);
}
void loop ()
{
  LEDka=2;
digitalWrite (LEDka,1);
}
```

Výhoda je ta, že hodnotu proměnné můžeme během programu měnit. Třeba přičítat a odečítat od ní jiné číslo, nebo jinou proměnnou.

LEDka=LEDka+1;

Úloha 4: Napište program, který pomocí proměnné nejprve všechny LED rozsvítí, rozsvícení bude po 1 sekundě, a pak je zase po jedné sekundě zhasne. Parametr funkce **delay** bude také proměnná, nazveme ji třeba *pauza*.
Řešení:

```
int LEDka; //deklarujeme proměnné
int pauza;
void setup ()
{
  pinMode (2,OUTPUT);
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);
  pauza=1000; //zde řekneme, kolik bude čas pro delay
}
void loop ()
{
  LEDka=2; //všechny LEDky postupně rozsvítíme
  digitalWrite (LEDka,1);
  delay (pauza);
  LEDka=3;
  digitalWrite (LEDka,1);
  delay (pauza);
  LEDka=4;
  digitalWrite (LEDka,1);
  delay (pauza);
  LEDka=5;
  digitalWrite (LEDka,1);
  delay (pauza);

  LEDka=2; //a pak je zase zhasneme
  digitalWrite (LEDka,0);
  delay (pauza);
  LEDka=3;
  digitalWrite (LEDka,0);
  delay (pauza);
  LEDka=4;
  digitalWrite (LEDka,0);
  delay (pauza);
  LEDka=5;
  digitalWrite (LEDka,0);
  delay (pauza);


}
```

Úloha 5: Naprogramujte semafor.
S vědomostmi, které máme, už můžeme naprogramovat semafor. Na semaforu svítí zelená. Když se má auto zastavit, tak se rozsvítí oranžová a pak červená. Když se má auto rozjet, tak se rozsvítí červená s oranžovou a pak zelená. Červenou na semaforu nechte 3 sekundy. Zelenou 2 sekundy a oranžovou 1sekundu.

Řešení:

```
int cervena=2, oranzova=3, zelena=4; //deklarujeme proměnné a rovnou přiřadíme hodnotu.
void setup ()
{
  pinMode (2,OUTPUT);
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
}
void loop ()
{
  digitalWrite (zelena,1);
  delay (2000);
  digitalWrite (zelena,0);
  digitalWrite (oranzova,1);
  delay (1000);
  digitalWrite (oranzova,0);
  digitalWrite (cervena,1);
  delay (3000);
  digitalWrite (oranzova,1);
  delay (1000);
  digitalWrite (cervena,0);
  digitalWrite (oranzova,0);
}
```

Zapojení na Saimon: D02-93 , D03-95 , D04-97, D05-99, 92-log0, 94-log0 , 96-log0
Zapojení na S-arduino: D02-LED1 , D03-LED2 , D04-LED3

Úloha 6: Cyklus

Funkce loop() se opakuje pořád dokola. V programu ale můžeme vytvořit **cyklus**, který se bude také opakovat, ale bude mít určitě podmínky jako kolikrát se bude opakovat, za jakých podmínek skončí a tak podobně. K vytvoření cyklu budeme používat klíčové slovo **do.** Pak bude následovat složená závorka otevřená {. Pak napíšeme seznam funkcí, neboli příkazů, které chceme v cyklu opakovat, a potom napíšeme složenou závornu uzavřenou }. Za tím vším napíšeme klíčové slovo **while(podmínka)**. Místo slova *podmínka* napíšeme nějakou podmínku, která **když je pravda**, tak se cyklus opakuje. Jakmile přestane být pravda, cyklus skončí. Schválně cyklus umístím do funkce setup(), i když ho můžeme umístit i do funkce loop(). Program bude vypadat takto:

```
int LEDka; // deklarujeme proměnné
int pauza;
void setup ()
{
  pinMode (2,OUTPUT);
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);
  pauza=1000; //zde řekneme, kolik bude čas pro delay

  LEDka=2;   //před cyklem nastavíme hodnotu promenné
  do  //zde začíná cyklus
   {  //příkazy mezi těmito závorkami se budou opakovat, dokud cyklus neskončí
       digitalWrite (LEDka,1);  //po prvním průchodu cyklem rozsvítíme LED na zdířce D02
       delay (pauza);
       LEDka= LEDka+1;  // do proměnné přičteme 1
   } while (LEDka<6); //opakuj dokud je LEDka < 6.
                      //tím cyklus končí
}
void loop ()
{
}
```

Zapojení na Saimon: D02-93 , D03-95 , D04-97, D05-99, 92-log0, 94-log0 , 96-log0, 98-log0
Zapojení na S-arduino: D02-LED1 , D03-LED2 , D04-LED3 , D04-LED4

Poznámka k podmínce - pokud chceme **porovnat**, jestli je proměnná rovna nějakému číslu, nepíšeme jedno =, ale **píšeme dvě** !!!!. Jedno = píšeme jen tehdy, když do proměnné chceme hodnotu přiřadit. Takže třeba A=1 přiřadí do proměnné A jedničku, a A==1 testuje, zda je a rovno jedné.

Úloha 7: Zkuste dopsat kód tak, aby všechny LED zase postupně zhasly.

Řešení: přidáme další cyklus. Nesmíme zapomenout na všechny potřebné podmínky a nastavení.

```
int LEDka, pauza; // deklarujeme proměnné
void setup ()
{
 pinMode (2,OUTPUT);
 pinMode (3,OUTPUT);
 pinMode (4,OUTPUT);
 pinMode (5,OUTPUT);
 pauza=1000; //zde řekneme, kolik bude čas pro delay
 LEDka=2;   //před cyklem nastavíme hodnotu promenné
 do  //zde začíná cyklus
  {  //funkce mezi těmito závorkami se budou opakovat, dokud cyklus neskončí
      digitalWrite (LEDka,1);  //po prvním průchodu cyklem rozsvítíme LED na zdířce D02
      delay (pauza);
      LEDka= LEDka+1;  // do proměnné přičteme 1
  } while (LEDka<6); //opakuj dokud je LEDka < 6.

 LEDka=2;   //před cyklem nastavíme hodnotu promenné
 do  //zde začíná cyklus
  {  //funkce mezi těmito závorkami se budou opakovat, dokud cyklus neskončí
      digitalWrite (LEDka,0);  //a zase od první LED zhasneme
      delay (pauza);
      LEDka= LEDka+1;  // do proměnné přičteme 1
  } while (LEDka<6); // opakuj dokud je LEDka < 6.
}
void loop ()
{
}
```

Nebo část funkce setup() může vypadat i takto:

```
 LEDka=2;   //před cyklem nastavíme hodnotu promenné
 do  //zde začíná cyklus
  {  //funkce mezi těmito závorkami se budou opakovat, dokud cyklus neskončí
      digitalWrite (LEDka,1);  //po prvním průchodu cyklem rozsvítíme LED na zdířce D02
      delay (pauza);
      LEDka= LEDka+1;  // do proměnné přičteme 1
  } while (LEDka<6); //// opakuj dokud je LEDka < 6.

 do  //zde začíná cyklus, hodnota promenné LEDka je 6.
  {  //funkce mezi těmito závorkami se budou opakovat, dokud cyklus neskončí
      digitalWrite (LEDka,0);  //tentokrát zhasínáme od poslední LED
      delay (pauza);
      LEDka= LEDka-1;  // z proměnné odečteme 1
  } while (LEDka>1); //// opakuj dokud je LEDka > 1.
}
```

Úloha 8: Použijte testovací LED T1 až T9 (můžete klidně až TC). Pomocí cyklu je postupně všechny rozsviťte a zase zhasněte. Nezapomeňte na funkci **pinMode** pro všechny zdířky. Abychom nemuseli **pinMode** opisovat do kódu tolikrát, použijeme pro její volání také cyklus.

Řešení:

```
int LEDka, pauza;
void setup ()
{
 LEDka=2; //začneme od zdířky 2
 do
 {  //v tomto cyklu nastavíme všechny zdířky jako výstupy
   pinMode (LEDka,OUTPUT);
   LEDka=LEDka+1;
 } while (LEDka<14); //opakuj, dokud LEDka<14

 pauza=1000; //zde řekneme, kolik bude čas pro delay
 LEDka=2;   //před cyklem nastavíme hodnotu promenné
 do  //zde začíná cyklus
  {  //funkce mezi těmito závorkami se budou opakovat, dokud cyklus neskončí
      digitalWrite (LEDka,1);  //po prvním průchodu cyklem rozsvítíme LED na zdířce D02
      delay (pauza);
      LEDka= LEDka+1;  // do proměnné přičteme 1
  } while (LEDka<14); // opakuj, dokud LEDka<14

 LEDka=2;   //před cyklem nastavíme hodnotu promenné
 do  //zde začíná cyklus
  {  //funkce mezi těmito závorkami se budou opakovat, dokud cyklus neskončí
      digitalWrite (LEDka,0);  //a zase od první LED zhasneme
      delay (pauza);
      LEDka= LEDka+1;  // do proměnné přičteme 1
  } while (LEDka<14); // opakuj, dokud LEDka<14.
}
void loop ()
{
}
```

Zapojení drátků:
D02-T1 , D03-T2 , D04-T3 , D05-T4 , D06-T5 , D07-T6 , D08-T7 , D09-T8 , D10-T9 , D11-TA , D12-TB , D13-TC

Úloha 9: Vytvořte program, který rozsvítí postupně LED1, LED2, LED3, LED4 a pak je postupně zhasne. A to celé se bude opakovat 4 krát.

Řešení:
K tomu potřebujeme cykly dva stejně jako v úloze 6. A ty dva cykly vnoříme do dalšího cyklu, který se bude opakovat 4 krát. Potřebujeme další proměnnou, která bude říkat, kolikrát se rozsvítilo a zhasnulo.

```
int LEDka, pauza, pocetOpakovani;

void setup ()
{
  pinMode (2,OUTPUT);
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);
  pauza=1000; //zde řekneme, kolik bude čas pro delay


  pocetOpakovani=4;
  do  //tenhle cyklus říká, kolikrát se bude rozsvecení opakovat
  {
     LEDka=2;
     do  //tenhle cyklus rozsvítí LEDky
      {
          digitalWrite (LEDka,1);
          delay (pauza);
          LEDka= LEDka+1;
      } while (LEDka<6);

     LEDka=2;
     do  //tenhle cyklus zhasne LEDky
      {
          digitalWrite (LEDka,0);
          delay (pauza);
          LEDka= LEDka+1;
      } while (LEDka<6);
      pocetOpakovani= pocetOpakovani-1;
  } while (pocetOpakovani>0);
}
void loop ()
{
}
```

Úloha 10: Naučíme používat podmíněný příkaz. K tomu slouží klíčové slovo **if**.

(A==1) digitalWrite (3,1); //pokud A se rová 1, zapiš na zdířku D03 logickou 1

Zápis je:
if (podmínka) příkaz**;**

Příkaz, který se provede při pravdivé podmínce, nemusí být jen jeden. Pak se musí napsat mezi složené závorky, stejně jako u cyklu **do**. Pro podmínku platí stejná pravidla, jako u cyklu do - když chceme zjistit, zda je proměnná rovna hodnotě, píšeme **dvakrát** rovná se.

```
if (podmínka)
{
   příkaz1;
   příkaz2;
   příkaz3;
}
```

Ještě je možný tento zápis - pokud je podmínka splněná, provede se příkaz1, pokud splněná není, provede se příkaz2.

if (podmínka) příkaz1**;**
else příkaz2**;**

Začneme s úplně jednoduchým úkolem. Vytvoříme cyklus, který čtyřikrát blikne LED1 připojenou na zdířce 2. Použijeme proměnnou A. Na zdířku D03 připojíme LED2, kterou rozsvítíme, když A bude rovno 3.

```
int A;
void setup ()
{
 pinMode (2,OUTPUT);
 pinMode (3,OUTPUT);
 pauza=1000; //zde řekneme, kolik bude čas pro delay

 A=0;
 do  //zde začíná cyklus
  {
      digitalWrite (2,1);
      delay (pauza);
      digitalWrite (2,0);
      delay (pauza);
      if (A==3) digitalWrite (3,1);  //pokud je A rovno 3, rozsviť LED2
      A++;  // do proměnné přičteme 1
  } while (A<4); //opakuj dokud je A< 4.
                          //tím cyklus končí
}
void loop ()
{
}
```

Úloha 11:

V programu máme proměnnou LEDka a měníme její hodnotu, například pomocí cyklu. Chceme, aby když LEDka = 1, se rozsvítila LED3. Když LEDka = 4, se rozsvítila LED1. pro LEDka = 2 se rozsvítí LED4 a pro LEDka = 3 se rozsvítila LED2. Pak je všechny zhasneme. A to celé se bude stále opakovat (napíšeme to do funkce loop).

```
int LEDka;
void setup ()
{
  pinMode (2,OUTPUT); //nastavíme zdířky
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);
}
void loop ()  //tahle funkce opakuje vše dokolečka
{
  LEDka=1;
  do
   {  //zde se testují podmínky. LED1 je na zdířce D02 atd.
       if (LEDka==1) digitalWrite (4,1);
       if (LEDka==2) digitalWrite (5,1);
       if (LEDka==3) digitalWrite (3,1);
       if (LEDka==4) digitalWrite (2,1);
       delay (1000);
       LEDka= LEDka+1;
   } while (LEDka<4);
  LEDka=2;
  do
   {  // zde zhasneme
      digitalWrite (LEDka,0);
      delay (1000);
      LEDka= LEDka+1;
   } while (LEDka<6);
}
```

Zapojení na Saimon: D02-93 , D03-95 , D04-97, D05-99, 92-log0, 94-log0 , 96-log0, 98-log0
Zapojení na S-arduino: D02-LED1 , D03-LED2 , D04-LED3 , D04-LED4

Úloha 12: Tlačítka

Už umíme rozsvítit LED, čili zapsat na zdířku logickou 0 nebo 1. Teď se naučíme ze zdířek také číst. Pro zápis jsme používali funkci **digitalWrite ();** a pro čtení budeme používat funkci **digitalRead ();**. Abychom měli ze zdířky co číst, musíme k ní připojit spínač.

Pamatujete na funkci digitalWrite()? Ta zapisovala na zvolenou zdířku buď 1 nebo 0. Připojená LED pak buď svítila nebo ne. Funkce digitalRead() dělá opak. Ta čte ze zdířky. Pokud chceme ze zdířky číst, tak ji musíme nastavit jako vstup. To uděláme příkazem **pinMode(číslo zdířky, INPUT).** Následující program čte na zdířce D06, do které připojíme tlačítko. Pokud je stisknuté, rozbliká LED1.

```
int stiskTlacitkaS3;

void setup() {
  pinMode (2, OUTPUT); //zdířku D02 nastavíme jako výstup
  pinMode (6,INPUT);  //zdířku D06 nastavíme jako vstup
}

void loop() {
  stiskTlacitkaS3 = digitalRead (6); // do proměnné stiskTlacitka načteme zdířku D06
  if (stiskTlacitkaS3 == 1) //porovnáme, zda jsme načetli 1, čili je stisknuté tlačítko
  {
    digitalWrite (2,1);   //tohle je asi jasné
    delay (100);
    digitalWrite (2,0);
    delay (100);
  }
}
```

Zapojení na Saimon: D02-93 , 92-log0, 122-X3(tohle X3 je na modulu Saimon1, nad S3) , X3-D06 (tohle X3 je na modulu Saimon4), 123-log1, 124-log0.

Zapojení na S-arduino: D02-LED1, S3-D06

Co dělá následující program?

```
int stiskTlacitkaS3;
void setup() {
  pinMode (2, OUTPUT); //zdířku D02 nastavíme jako výstup
  pinMode (6,INPUT);  //zdířku D06 nastavíme jako vstup
}

void loop() {

  stiskTlacitkaS3 = digitalRead (6);
  if (stiskTlacitkaS3 == 1) digitalWrite (2,0);
  else digitalWrite (2,1)

}
```

Úloha 13: Dopište program tak, aby stisk tlačítka S4 rozblikal LED2.
Řešení:

```
int stiskTlacitkaS3, stiskTlacitkaS4;

void setup() {

  pinMode (2, OUTPUT);
  pinMode (3, OUTPUT);

  pinMode (6,INPUT);
  pinMode (7,INPUT);

}

void loop() {

  stiskTlacitkaS3 = digitalRead (6);
  stiskTlacitkaS4 = digitalRead (7);
  if (stiskTlacitkaS3 == 1)
  {
    digitalWrite (2,1);
    delay (100);
    digitalWrite (2,0);
    delay (100);
  }
  if (stiskTlacitkaS4 == 1)
  {
    digitalWrite (3,1);
    delay (100);
    digitalWrite (3,0);
    delay (100);
  }

}
```

Zapojení na Saimon: D02-93 , D03-95 , 92-log0, 94-log0 , 122-X3, X3-D06, 123-log1 , 124-log0 , 125-X4 , X4-D07 , 126-log1 , 127-log0

Zapojení na S-arduino: D02-LED1 , D03-LED2 , S3-D06, S4-D07

Dobrovolně můžete dopsat program tak, aby navíc tlačítko S5 rozblikalo LED3 a S6 rozblikalo LED4

Úloha 14: Zvuk

Zvuk tvoříme příkazem **tone ()**. Píše se takto:

tone (9,1000,50);

9 znamená, že reproduktor je připojený na zdířku D09. 1000 znamená výšku tónu. A 50 znamená délku tónu, takže 50 milisekund.
Chtěl bych ještě zmínit, že v případě zvuku nemusíme používat pro zdířku funkci pinMode.

Program vypadá takhle

```
void setup() {

   tone (9,1000,50);  //uděláme zvuk
   delay (50);  //musíme počkat až zvuk dozní
}

void loop() {
}
```

No, tohle bylo moc jednoduché. Uděláme to trochu složitější.

Úloha 15:
Pomocí cyklu uděláme sirénu. Potřebujeme proměnnou, která bude obsahovat hodnotu výšky zvyku, kterou budeme v cyklu měnit. Nazveme ji **vyskaZvuku** Začneme na hodnotě 200 a budeme postupně zvyšovat, až dosáhne výšky například 1000. Pak jí budeme zase snižovat. K tomu prozatím vytvoříme druhý cyklus.

```
int vyskaZvuku;
void setup() {

}

void loop() {
vyskaZvuku =200; //zde je počáteční hodnota výšku zvuku
do
{  //tento cyklus zvuk zvyšuje
  tone (9, vyskaZvuku,10);
  delay (10);
  vyskaZvuku ++;
} while (vyskaZvuku <1000);

do
{ //z tenhle zase snižuje
  tone (9, vyskaZvuku,10);
  delay (10);
  vyskaZvuku --;
} while (vyskaZvuku >200);
}
```

První cyklus zvyšuje hodnotu proměnné A, pro výšku zvuku. Druhý cyklus ji zase snižuje. Díky tomu, že oba cykly jsou ve funkci **loop ()**, tak se to celé opakuje a vznikne siréna.

Úloha 16:
Uměli bychom sirénu naprogramovat pouze pomocí jednoho cyklu? Nebo bez cyklu ve funkci **loop()**?

Nápověda - bude potřeba druhá proměnná. A příkaz **if**. Podle hodnoty proměnné **vyskaZvuku** budeme hodnotu buď navyšovat, nebo snižovat. To rozhodneme příkazem **if**.

Řešení:

```
int vyskaZvuku, zmenaZvuku;
void setup() {
vyskaZvuku=500;
zmenaZvuku=5;
}

void loop() {
tone (9,vyskaZvuku,10);
delay (10);
vyskaZvuku=vyskaZvuku+zmenaZvuku;

if (vyskaZvuku>1000) zmenaZvuku=-5;
if (vyskaZvuku<500) zmenaZvuku=5;
}
```

Hodnotu výšky zvuku tentokrát měníme o 5, nikoliv o 1 jako v úloze 15.

Úloha 17:
Naprogramujeme malé piano. Po stisku klávesy se ozve tón. Spínače S3 až S6 napojíme na zdířky D02 až D05. Pomocí příkazu **if ()** budeme testovat, zda jsou stisknuté a v případě, že ano, uděláme tón.
Řešení:

```
void setup() {
  pinMode (2,INPUT); //nastavíme zdířky jako vstupy
  pinMode (3,INPUT);
  pinMode (4,INPUT);
  pinMode (5,INPUT);
}

void loop() {
  int S3,S4,S5,S6; //Zde je deklarace proměnných uvnitř funkce loop
  S3=digitalRead (2);// načteme do proměnných stavy tlačítek
  S4=digitalRead (3);
  S5=digitalRead (4);
  S6=digitalRead (5);
  if (S3) // otestujeme, zda je stačítko zmáčknuté
  {
    tone (9,261,10);
    delay (20);
  }
  if (S4)
  {
    tone (9,294,10);
    delay (20);
  }
  if (S5)
  {
    tone (9,329,10);
    delay (20);
  }
  if (S6)
  {
    tone (9,349,10);
    delay (20);
  }
 }
```

Všimněte si, že proměnné tentokrát deklarujeme uvnitř funkce loop(). Tím chci jen ukázat, že to jde. Jaký je rozdíl mezi touto deklarací a deklarací na začátku programu? Pokud je proměnná deklarována uvnitř funkce, říká se jí lokální proměnná a počítač o ní "ví" jen uvnitř dané funkce. Pokud ji deklarujeme na začátku programu, počítač o ní "ví" všude a všude je přístupná její hodnota.

Zkuste přemístit **int S3,S4,S5,S6;** do funkce **setup()** a sledujte chybové hlášení.

Dál si zkuste změnit slovo int u deklarace proměnných na bool. Takto:

```
bool S3,S4,S5,S6;
```

Proměnná typu int totiž může obsahovat hodnotu od −32768 až do 32767. Nám ale stačí vědět, jestli je tlačítko zmáčknuté, nebo ne. Čili stačí nám ukládat hodnotu 0 nebo 1, čili pravda, nebo nepravda. K tomu slouží **datový typ** *bool.* Ten může nabývat pouze hodnot 0 a 1, neboli TRUE a FALSE (pravda a nepravda).

Úloha 18: Měníme jas u LEDek - Analogový výstup

Funkce **digitalWrite ();** zapíše na danou zdířku hodnotu 0 nebo 1. Když máme na zdířku připojenou LEDku, tak se buď rozsvítí, nebo ne. Když chceme měnit jas, tedy jak moc LEDka bude svítit, použijeme funkci **analogWrite ();** která nám umožní nejen rozsvítit nebo zhasnout, ale určíme si hodnotu jasu. Následující program postupně rozsvítí LED na zdířce D03. V programu je naznačený druhý cyklus, který budete mít za úkol dopsat tak, aby LED zase pomalu zhasla. Bude podobný prvnímu cyklu, ale je potřeba udělat správné úpravy.

```
void setup() {

}

void loop() {
int jas;
  do
  {
    analogWrite (3, jas); // na zdířku D03 zapíšeme hodnotu jasu, od 0 do 255
    jas = jas + 1;
    delay (10);
  } while (jas <255);
  do
  {

  }

}
```

Zapojení na Saimon: D03-93 , 92-log0

Zapojení na S-arduino: D03-LED1

Pokud budete zkoušet zapojit více LEDek, tak chci upozornit, že **analogWrite()**; funguje pouze na zdířkách, kde je u čísla napsáno PWM. Schválně vyzkoušejte připojit LEDku na zdířku D02, upravte program a sledujte, co se stane.

Úloha 19: Potenciometr - Analogový vstup

Jas dokážeme také měnit součástkou, která se jmenuje **potenciometr**. Je na stavebnici vlevo dole. Zapojte drátek do zdířky u potenciometru a druhý konec zapojte do libovolné LED. Zakruťte s kloboučkem sem a tam. Pokud je vše správně zapojené, měl by se měnit jas.

Hodnotu potenciometru můžeme samozřejmě programem také číst. K tomu slouží funkce **analogRead ()** a zdířky **A00** až **A07**. Funkce vrací hodnotu od 0 (zhasnuto) do 1023 (nejvyšší jas). Připojte výstup z potenciometru na zdířku A07. Následující program přečte hodnotu potenciometru do proměnné **potenciometr**, která bude mezi 0 a 1023, podle toho, jak s potenciometrem otáčíme. Proměnnou použijeme jako argument funkce **tone ()**, a vytvoříme bzučák.

```
int potenciometr;
void setup() {
}

void loop() {
  potenciometr=analogRead(A7);
  tone (8,potenciometr, 10);
  delay (10);
}
```

Zapojení na Saimon: 24-A07 , 23-log.0 , 25-log.1 , D08-108 , 109-log.0

Zapojení na S-arduino: potenciometr-A07 , D08-reproduktor

Zkuste změnit program tak, aby zároveň se změnou zvuku měnil jas na nějaké LED.

Úloha 20: Další cyklus

Cyklus můžeme tvořit i příkazem **for**. Správný zápis vypadá následovně

for (A=2; A<=13; A++)

Tento cyklus říká, že hodnota proměnné **A** bude na začátku cyklu rovna 2. Cyklus pokračuje, když je splněna podmínka, že **A<=13**. A v každém průchodu cyklem se A zvýší o 1. Čili obecně vypadá náš cyklus takto:

for (počáteční hodnota proměnné, podmínka pokračování, změna proměnné)

Změna jasu ledky se dá udělat i takto.

```
void setup() {

}

void loop() {
int A;
for (A=0;A<255;A++) //tento cyklus rozsvítí
{
  analogWrite (3,A);
  delay (10);
}
for (A=255;A>0;A--) //tento cyklus zhasne
{
  analogWrite (3,A);
  delay (10);
}
}
```

Zapojení na Saimon: D03-93 , 92-log0

Zapojení na S-arduino: D03-LED1

Úloha 21. Sloupec LEDek

Další ukázka bude postupně rozsvěcet testovací LEDky podle hodnoty potenciometru. Dokážete sami přijít na to, jak program funguje?

```
int MaxLED, A, potenciometr;

void setup() { //pomocí cyklu for nastavíme OUTPUT
  for (A=2;A<=13;A++)
  {
    pinMode(A,OUTPUT);
  }
}

void loop() {
  potenciometr = analogRead(A7);
  MaxLED = map (potenciometr, 0,1023,2,13);
  for (A=2;A<=13;A++)
  {
    if (A<=MaxLED) digitalWrite (A,1);
    else digitalWrite (A,0);
    delay (10);
  }
}
```

Zapojení na Saimon: 24-A07 , 23-log.0 , 25-log.1 , D02-T7, D03-T8, D04-T9, D05-TA, D06-TB, D07-TC, D08-T1, D09-T2, D10-T3, D11-T4, D12-T5, D13-T6

Zapojení na S-arduino: potenciometr-A07 , D02-T7, D03-T8, D04-T9, D05-TA, D06-TB, D07-TC, D08-T1, D09-T2, D10-T3, D11-T4, D12-T5, D13-T6

Nejdříve si ho zkuste naprogramovat a nahrát do arduina. Zapojte drátky.

Nápovědu dám akorát k funkci **map ()**; Funguje tak, že změní rozsah hodnot 0 až 1023 na rozsah 2 až 13. Takže pokud hodnota proměnné potenciometr bude 1023, hodnota MaxLED bude 13. Tím se určí nejvyšší LED, která se má v cyklu rozsvítit, ostatní se zhasnou.

Úloha 22: Číslicové displeje

V této úloze se naučíme zobrazovat čísla na displejích pomocí programu. Nejprve to ale zkusíme pomocí spínačů. U displeje jsou zdířky s digitálními číslicemi 8,4,2,1. Zapojte drátky následovně:

Zapojení na S-arduino: S3-8, S4-4, S5-2, S6-1, VCC (displej) - log.1
Zapojení na Saimon: 122-88, 125-89, 128-90, 131-91, 123-log.1, 124-log.0, 126-log.1, 127-log.0, 129-log.1, 130-log.0, 132-log.1, 133-log.0, napájení číslicovky - log.1

Stiskem spínačů se na displeji zobrazují číslice. Stiskem například S3 se zobrazí na displeji číslo 8 atd. Pokud stiskneme spínačů více, zobrazí se součet čísel. Stisknutý spínač připojí na zdířku logickou 1. Nestisknutý připojí logickou 0. Dá se tedy zapsat kombinace stisknutých a nestisknutých spínačů například takto: S3 S4 S5 S6 = 0101. A to zobrazí číslo 5. Tomuhle říkáme **binární zápis čísel** (v návodu pro Saimon 1 je to také podrobně vysvětlené). Na displeji se kromě čísel mohou zobrazovat i znaky **A, b, C, d, E, F**. To proto, že například kombinace spínačů 1011 by měla zobrazit číslo 11, ale to na jednociferném displeji nejde. Takže se místo toho zobrazí znak **b**. Tabulka všech možných kombinací je uvedena zde:

S3	S4	S5	S6	ČÍSLO
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

S3	S4	S5	S6	ČÍSLO
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	b
1	1	0	0	C
1	1	0	1	d
1	1	1	0	E
1	1	1	1	F

Zápisu čísla pomocí znaků 0,1,2,3,4,5,6,7,8,9,A,b,C,d,E,F říkáme **hexadecimální** zápis čísla.

Nyní přepojíme drátky takto:
Zapojení na S-arduino: D02-8, D03-4, D04-2, D05-1
Zapojení na Saimon: D02-88, D03-89, D04-90, D05-91

Program pro zobrazování čísel by mohl vypadat například takto:

```
void setup() {
  pinMode (2,OUTPUT);
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);

  digitalWrite (2,0);
  digitalWrite (3,1);
  digitalWrite (4,0);
  digitalWrite (5,1);
}

void loop() {

}
```

Program zapíše jedničky na zdířky D03 a D05, čili na displeji se zobrazí číslo 5.

Úloha 23:

Vytvořte pomocí vzoru v předchozí úloze a tabulky takový program, který zobrazí po sobě čísla 0,1,2,3,4,5,6,7,8,9

Řešení:

Pro zobrazení každého čísla potřebujeme 4 příkazy **digitalWrite();**, tedy 4 řádky. To je dost nepraktické. Navíc pokud budeme mít hodnotu v nějaké proměnné, tak tímto způsobem zobrazit nepůjde. Použijeme operaci, které se říká **bitový součin** a značí se **&**. V tomto návodu se nebudu věnovat tomu, co to bitový součin je a jak funguje. Kdo má zájem se to dozvědět, nechť si to najde na internetu. Program pro zobrazení čísla pak vypadá takto:

```
void setup() {
  int A=6;
  pinMode (2,OUTPUT);
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);

  digitalWrite (2,A&8); //test, zda se má zobrazit 8
  digitalWrite (3,A&4); //test, zda se má zobrazit 4
  digitalWrite (4,A&2); //test, zda se má zobrazit 2
  digitalWrite (5,A&1); //test, zda se má zobrazit 1
}

void loop() {

}
```

Když budeme chtít zobrazit více po sobě jdoucích čísel, opět to ale znamená pro každé číslo čtyři řádky. Proto se naučíme tvořit si vlastní funkce, vlastní příkazy.

Úloha 24: Vytvoření vlastní funkce
Zatím jsme se v programu setkali s dvěmi funkcemi. A to void **loop ()** a void **setup ()**. Teď se naučíme vytvořit si vlastní. Funkci si nazveme třeba **zobrazCislo ().** Funkce se definuje takto:

```
void zobrazCislo (int cislo) //definice nové funkce
{ //příkazy, které se mají provést, pokud funkci voláme
  digitalWrite (2,cislo&8);  //zobrazí se cislo na číslicovém displeji
  digitalWrite (3,cislo&4);
  digitalWrite (4,cislo&2);
  digitalWrite (5,cislo&1);
}

void setup() {

  int A;
  pinMode (2,OUTPUT);
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);
  A=6;
  zobrazCislo (5); //voláni naší funkce, zobrazí 5
  delay (1000);
  zobrazCislo (7); //voláni naší funkce, zobrazí 7
  delay (1000);
  zobrazCislo (A); //voláni naší funkce, zobrazí 6 (hodnota proměnné)
  delay (1000);

}

void loop() {

}
```

Zapojení drátků je stejné, jako v předchozí úloze.

Ted si to trochu přiblížíme. funkce se tedy jmenuje **zobrazCislo**. Slovo **void** před funkcí znamená, že funkce nevrací žádnou hodnotu. Co to znamená, že funkce nevrací hodnotu? Vzpomínáte na funkci **analogRead()**? Tak třeba ta vrací hodnotu, použili jsme jí třeba takto:

```
potenciometr=analogRead(A7);
```

Hodnota, kterou funkce vrací, se přiřadí do proměnné **potenciometr**. Naše funkce ale žádnou hodnotu nevrací, proto při definici napíšeme před funkci slovo **void**. Pak následuje jméno funkce, které jsme si zvolili a jednoduché závorky. Do závorek můžeme napsat typ a název proměnné, kterou má funkce vyžadovat při jejím volání. Tomu říkáme parametr funkce. My chceme zobrazit číslo, takže jako parametr napíšeme **int cislo**. Následují složené závorky, mezi které napíšeme seznam příkazů, které se mají provést. Tím jsme napsali definici funkce. K tomu, aby funkce opravdu něco udělala, jí musíme ještě zavolat. To provedeme v těle funkce setup() nebo loop (). Jak se to dělá je v příkladu nahoře.

Více o definici funkcí v tomto návodu uvádět nebudu, zvědavé čtenáře opět odkážu na internet.

Úloha 25: Zobrazte po sobě znaky 0 až F

Použijeme cyklus a námi definovanou funkci **zobrazCislo()**. Tentokrát použijeme cyklus **do** i **for**.

Varianta 1: s cyklem do

```
int A;

void zobrazCislo (int cislo) //deklarace nové funkce
{
  digitalWrite (2,cislo&8);
  digitalWrite (3,cislo&4);
  digitalWrite (4,cislo&2);
  digitalWrite (5,cislo&1);
}

void setup() {
  pinMode (2,OUTPUT);
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);
  A=0;
  do {
    zobrazCislo (A);
    delay (1000);
    A++;
  } while (A<16);

}

void loop() {

}
```

Zapojení drátků je stejné, jako v předchozí úloze.

Varianta 2: s cyklem for

```
int A;

void zobrazCislo (int cislo) //deklarace nové funkce
{
  digitalWrite (2,cislo&8);
  digitalWrite (3,cislo&4);
  digitalWrite (4,cislo&2);
  digitalWrite (5,cislo&1);
}

void setup() {
  pinMode (2,OUTPUT);
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);
  for (A=0;A<16;A++)
  {
    zobrazCislo (A);
    delay (1000);
  }
}

void loop() {

}
```

Zapojení drátků je stejné, jako v předchozí úloze.

Úloha 26: Zobrazení hodnoty potenciometru na číslicovém displeji

Víme, že analogové vstupy nám dávají hodnoty nikoliv 0 nebo 1, ale v rozsahu 0 až 1023. Číslicový displej je schopný zobrazit 16 znaků, a to čísla 0 až F. Pokud program dobře upravíme, můžeme znaky 0 až F měnit pomocí otáčení potenciometru. Potřebujeme jen informaci, že 1023 děleno 64 se rovná 16. Přečtenou hodnotu z analogového vstupu vydělíme 64 a dostaneme požadovaný rozsah hodnot pro zobrazení na číslicovém displeji. Tu zobrazíme pomocí naší funkce **void zobrazCislo (int cislo)**

Řešení:

```
int A;

void zobrazCislo (int cislo) //deklarace nové funkce
{
  digitalWrite (2,cislo&8);
  digitalWrite (3,cislo&4);
  digitalWrite (4,cislo&2);
  digitalWrite (5,cislo&1);
}

void setup() {
  pinMode (2,OUTPUT);
  pinMode (3,OUTPUT);
  pinMode (4,OUTPUT);
  pinMode (5,OUTPUT);

}

void loop() {
    A=analogRead (A7)/64;
    zobrazCislo (A);
    delay (10);
}
```

Zapojení na S-arduino: D02-8, D03-4, D04-2, D05-1, potenciometr-A07

Zapojení na Saimon: D02-88, D03-89, D04-90, D05-91, 23-log.1, 25-log.0,
24-X1(na Saimon1), X1(na Saimon4)-A07

Úloha 27: LED displej
K ovládání LED displeje potřebujeme funkce, které nejsou součástí vývojového prostředí. Jsou k dispozici v jiném souboru, kterému říkáme knihovna. Knihovnu musíme připojit k programu, abychom funkce pro ovládání LED displeje mohli používat. Knihoven je na internetu veliká spousta, já jsem si vybral knihovnu s názvem **ledcontrol.h**. Knihovnu připojíme příkazem
#include <LedControl.h>

Jelikož knihovna není součástí vývojového prostředí, je třeba ji nahrát do složky:

dokumenty\arduino\libraries\

tam je třeba rozbalit obsah souboru **ledcontrol.zip**.
Nejnovější verze (v době psaní manuálu) se nachází na adrese

https://github.com/wayoda/LedControl/releases

Další návody pro instalaci knihoven včetně obrázků jsou zde:

http://www.arduino.cc/en/Guide/Libraries

Jak tedy napsat program pro ovládání LED displeje. Displej má 8 sloupců a 8 řádků LED diod. Začneme s něčím jednoduchým. Rozblikáme jeden bod na displeji, tedy jednu LEDku. K tomu slouží speciální příkazy. Abychom je mohli používat, musíme nejdříve vytvořit instanci třídy (něco jako proměnná), ke které přiřadíme displej. To uděláme zápisem

LedControl Displej=LedControl(DIN,CLK,CS,1);

kde DIN, CLK a CS jsou zdířky LED displeje, které připojíme na zdířky arduina. Například takto:

LedControl Displej=LedControl(12,10,11,1);

Pak propojíme drátky DIN-D12, CS-D11, CLK-D10. K tomu je ještě potřeba displej připojit na napájení, VCC-LOG.1.

Příkazy pro ovládání displeje se zde nazývají **metody**. Ukážeme si několik prvních. nejlépe to bude vidět v programu.

```
#include <LedControl.h> //  Tímto připojíme knihovnu
                        //  s funkcemi pro ovládání
                        //  LED matice

                        //  musíme vytvořit instanci třidy
                        //  představující displej

LedControl Displej=LedControl(12,10,11,1);

// zdířka D12 je připojena na DIN
// zdířka D11 je připojena na CS
// zdířka D10 je připojena na CLK
// 1 znamená, že máme jen jeden displej

void setup()
{
  // nula znamená právě to, že máme jeden displej
  // (číslo prvního displeje je 0)
  Displej.shutdown(0,false);// tohle zapne displej
  Displej.setIntensity(0,4);// nastavíme jas displeje na 4 (možné je 0-15)
  Displej.clearDisplay(0);// tahle funkce smaže displej
}
void loop()
{
  int radek, sloupec;
  radek = 2;
  sloupec = 3; //Zadáme číslo řádku a sloupce, kde se
             //má rozsvítit LEDka. Čísla začínají od nuly
  Displej.setLed(0, radek, sloupec, 1); // Zapne LEDku
  delay (1000);
  Displej.setLed(0, radek, sloupec, 0); // Vypne LEDku
  delay (1000);
}
```

Úloha 28: Pohyb na LED displeji

Pomocí cyklu udělejte pohyb jednoho bodu na displeji shora dolů.

Řešení: Když umíme rozsvítit jeden bod, tak jich můžeme rozsvítit i víc. Pohyb se dělá tak, že nejdříve rozsvítíme jeden bod. Chvíli počkáme, pak ho zhasneme a rozsvítíme další bod, hned vedle. A tak pokračujeme stále dál. Více je vidět opět v programu

```
#include <LedControl.h> //  Tímto připojíme knihovnu
                        //  s funkcemi pro ovládání
                        //  LED matice

                        //  musíme vytvořit instanci třídy
                        //  představující displej

LedControl Displej=LedControl(12,10,11,1);

// zdířka D12 je připojena na DIN
// zdířka D11 je připojena na CS
// zdířka D10 je připojena na CLK
// 1 znamená, že máme jen jeden displej

void setup()
{
  // nula znamená právě to, že máme jeden displej
  // (číslo prvního displeje je 0)
  Displej.shutdown(0,false);// tohle zapne displej
  Displej.setIntensity(0,4);// nastavíme jas displeje na 4 (možné je 0-15)
  Displej.clearDisplay(0);// tahle funkce smaže displej
}
void loop()
{
  int radek, sloupec;

  sloupec = 3; //Zadáme číslo sloupce, kde se
             //má pohybovat bod. Čísla začínají od nuly

  for ( radek=0; radek<8; radek++)
  {
     Displej.setLed(0,radek,sloupec,1); // rozsvítí LEdku
     delay(100);
     Displej.setLed(0,radek,sloupec,0); // před pohybem ji zhasne
  }


}
```

Úloha 29: Pohyb po celém displeji

Napište program, který bude pohybovat bodem zleva doprava. Začne vlevo nahoře. Až dojde na konec řádku, bude pokračovat o řádek níž. Až dojde do pravého spodního rohu, bude se to celé opakovat.

Řešení:
v programu budeme potřebovat dva cykly. Jeden bude měnit číslo řádku a druhý číslo sloupce. Druhý cyklus bude vnořený do prvního.

```
#include <LedControl.h> //  Tímto připojíme knihovnu
                        //  s funkcemi pro ovládání
                        //  LED matice

                        //  musíme vytvořit instanci třídy
                        //  představující displej

LedControl Displej=LedControl(12,10,11,1);

// zdířka D12 je připojena na DIN
// zdířka D11 je připojena na CS
// zdířka D10 je připojena na CLK
// 1 znamená, že máme jen jeden displej

void setup()
{
  // nula znamená právě to, že máme jeden displej
  // (číslo prvního displeje je 0)
  Displej.shutdown(0,false);// tohle zapne displej
  Displej.setIntensity(0,4);// nastavíme jas displeje na 4 (možné je 0-15)
  Displej.clearDisplay(0);// tahle funkce smaže displej
}
void loop()
{
  int radek, sloupec;

  for ( radek=0; radek<8; radek++)
  {
    for ( sloupec=0; sloupec<8; sloupec++)
    {
      Displej.setLed(0,radek,sloupec,1); // rozsvítí LEdku
      delay(100);
      Displej.setLed(0,radek,sloupec,0); //  a zhasne
    }
  }
}
```

Úloha 30: Zobrazení obrázku.

Další metoda pro ovládání displeje umí zobrazit celý řádek. Pomocí jedniček a nul můžeme říci, které LEDky v řádku budou svítit a které ne. Metoda se jmenuje **setRow**. Jak se volá je vidět opět v programu.

```
#include <LedControl.h> //  Tímto připojíme knihovnu
                        //  s funkcemi pro ovládání
                        //  LED matice

                        //  musíme vytvořit instanci třidy
                        //  představující displej

LedControl Displej=LedControl(12,10,11,1);

// zdířka D12 je připojena na DIN
// zdířka D11 je připojena na CS
// zdířka D10 je připojena na CLK
// 1 znamená, že máme jen jeden displej

void setup()
{
  // nula znamená právě to, že máme jeden displej
  // (číslo prvního displeje je 0)
  Displej.shutdown(0,false);// tohle zapne displej
  Displej.setIntensity(0,4);// nastavíme jas displeje na 4 (možné je 0-15)
  Displej.clearDisplay(0);// tahle funkce smaže displej

  Displej.setRow ( 0,0,B00111100);
  Displej.setRow ( 0,1,B01111110);
  Displej.setRow ( 0,2,B11011100);
  Displej.setRow ( 0,3,B11111000);
  Displej.setRow ( 0,4,B11111000);
  Displej.setRow ( 0,5,B11111100);
  Displej.setRow ( 0,6,B01111110);
  Displej.setRow ( 0,7,B00111100);
}
void loop()
{
}
```

Kde jsou jedničky, tam svítí LEDka. Písmeno B v zápisu B00111100 znamená, že se jedná o binární zápis, to znamená pomocí jedniček a nul. Stejného výsledku bychom dosáhli, kdybychom na první řádek napsali Displej.setRow (0,0,60);, protože 60 je stejné číslo jako B00111100. Ale v binárním zápisu mnohem lépe vidíme, jak bude obrázek asi vypadat.

Úloha 31: Animace

Pokud bychom chtěli zobrazit nějakou animaci, je nejlepší udělat něco podobného, jako s blikající LEDkou. Zobrazit jeden obrázek, chvíli počkat a zobrazit další.

```
#include <LedControl.h>

LedControl Displej=LedControl(12,10,11,1);

// zdířka D12 je připojena na DIN
// zdířka D11 je připojena na CS
// zdířka D10 je připojena na CLK
// 1 znamená, že máme jen jeden displej

void setup()
{
  // nula znamená právě to, že máme jeden displej
  // (číslo prvního displeje je 0)
  Displej.shutdown(0,false);// tohle zapne displej
  Displej.setIntensity(0,4);// nastavíme jas displeje na 4 (možné je 0-15)
  Displej.clearDisplay(0);// tahle funkce smaže displej
}
void loop()
{
  Displej.setRow ( 0,0,B00111100);
  Displej.setRow ( 0,1,B01111110);
  Displej.setRow ( 0,2,B11011100);
  Displej.setRow ( 0,3,B11111000);
  Displej.setRow ( 0,4,B11111000);
  Displej.setRow ( 0,5,B11111100);
  Displej.setRow ( 0,6,B01111110);
  Displej.setRow ( 0,7,B00111100);

  delay (500);

  Displej.setRow ( 0,0,B00111100);
  Displej.setRow ( 0,1,B01111110);
  Displej.setRow ( 0,2,B11011111);
  Displej.setRow ( 0,3,B11111111);
  Displej.setRow ( 0,4,B11111000);
  Displej.setRow ( 0,5,B11111111);
  Displej.setRow ( 0,6,B01111110);
  Displej.setRow ( 0,7,B00111100);

  delay (500);
}
```

Pokud jste vše opsali správně, měla by se zobrazit animace Pacmana otevírajícího pusu.

Úloha 32: Vesmírný koráb

Tahle úloha už bude pro pokročilejší. Zobrazíme útvar, kterému budeme říkat vesmírný koráb. Budeme s ním pohybovat doleva a doprava pomocí potenciometru. A nakonec pomocí tlačítka budeme i střílet. Koráb bude vypadat na displeji takto:

01000000
11100000

Budeme ho zobrazovat pomocí metody **setLed**, která rozsvítí pouze jednu LEDku. My budeme chtít rozsvítit 4 LEDky (všude, kde je jednička). K tomu musíme určit souřadnice všech bodů, které mají svítit. Víme, že vlevo nahoře je bod s číslem řádku nula a číslem sloupce také nula. Zkráceně píšeme, že má souřadnice 0,0. Bod vlevo od něj má souřadnice 0,1 (řádek 0, sloupec 1).
Pokud bychom napsali, že bod vlevo nahoře má souřadnice **radek=0, sloupec=0**, tak bod vlevo od něj má souřadnice **radek, sloupec+1**. Bod na řádku pod ním má souřadnice **radek+1, sloupec**. A tak dále. Následující program zobrazí koráb.

```
#include <LedControl.h>

LedControl Displej=LedControl(12,10,11,1);

void setup()
{
  Displej.shutdown(0,false);  // tohle zapne displej
  Displej.setIntensity(0,4);    // nastavíme jas displeje na 4 (možné je 0-15)
  Displej.clearDisplay(0);     // tahle funkce smaže displej
}
void loop()
{
  int radek, sloupec;
  radek = 0;
  sloupec = 0;

  Displej.setLed(0, radek, sloupec+1, 1);
  Displej.setLed(0, radek+1, sloupec, 1);
  Displej.setLed(0, radek+1, sloupec+1, 1);
  Displej.setLed(0, radek+1, sloupec+2, 1);
  delay (1000);
}
```

Koráb se nám zobrazil vlevo nahoře. Když bychom chtěli, aby se zobrazil dole, musíme posunou souřadnici řádku.
radek = 6;

Teď musíme vyřešit pohyb. Chceme, aby se koráb pohyboval zleva doprava, tedy aby se měnila souřadnice sloupec. Ta se může měnit od 0 do 7. Pokud bude sloupec=0, koráb se zobrazí vlevo. Pokud by byl sloupec=7, bude skoro celý koráb mimo obrazovku. Aby byla vidět alespoň hlaveň, musí být sloupec maximálně roven 6. Naopak vlevo může být souřadnice sloupec nejen rovna 0, ale můžeme ji přiřadit i hodnotu -1, aby koráb mohl hlavní až k levému okraji. Čili souřadnici sloupec můžeme měnit v intervalu <-1,6>.

Teď připojíme potenciometr na zdířku A07.

Zapojení na Saimon: 23-log.1, 24-X1(na Saimon1), X1(Na Saimon4)-A07, 25-log.0
Zapojení na S-arduino: potenciometr-A07.

Čtení hodnoty potenciometru ze zdířky A07 nám dává hodnoty 0 až 1023. My potřebujeme hodnoty -1 až 6. Je více způsobů, jak hodnotu převést, třeba pomocí již zmíněné funkce **map()**. Já to ale udělám trochu více průhledné, aby bylo vidět, co děláme. Nejdříve přečtenou hodnotu vydělíme číslem 100. Tím dostaneme hodnotu přibližně 0 až 10. My ale potřebujeme -1 až 6. Jednoduše odečteme od proměnné sloupec jedničku a dostaneme hodnoty -1 až 9. Jelikož maximálně potřebujeme číslo 6, dáme za čtení podmínku, která bude testovat, zda sloupec nepřesáhl hodnotu 6. Pokud ji přesáhl, hodnota se nastaví na 6.

```
#include <LedControl.h>

LedControl Displej=LedControl(12,10,11,1);

void setup()
{
  Displej.shutdown(0,false);// tohle zapne displej
  Displej.setIntensity(0,4);// nastavíme jas displeje na 4 (možné je 0-15)
  Displej.clearDisplay(0);// tahle funkce smaže displej
}
void loop()
{
  int radek, sloupec;
  radek = 6;
  sloupec = analogRead(A7)/100;
  sloupec--;
  if (sloupec>6) sloupec=6;

    //nakreslíme koráb na souřadnice
  Displej.setLed(0, radek, sloupec+1, 1);
  Displej.setLed(0, radek+1, sloupec, 1);
  Displej.setLed(0, radek+1, sloupec+1, 1);
  Displej.setLed(0, radek+1, sloupec+2, 1);
  delay (100);
  //a smažeme koráb, aby mohl vzniknout pohyb
  Displej.setLed(0, radek, sloupec+1, 0);
  Displej.setLed(0, radek+1, sloupec, 0);
  Displej.setLed(0, radek+1, sloupec+1, 0);
  Displej.setLed(0, radek+1, sloupec+2, 0);

}
```

Nakonec doděláme jednoduchou střelbu. Připojíme tlačítko. A reproduktor.

Zapojení na Saimon: 120-log.1, 119-X2(na Saimon1), X2(Na Saimon4)-D02, 121-log.0, 109-log0, 108(na Saimon4)-D03.
Zapojení na S-arduino: S3-D02, reproduktor-D03

Při stisku tlačítka se aktivuje cyklus, který vytvoří pohyb bodu (střely) od hlavně až na horní část displeje. Přitom zazní zvuk.

Až si opíšete kód na následující stránce, zjistíte, že se například nedá pohybovat korábem během střelby. Navíc by to chtělo nějaké nepřátele. Tomu už se v tomto návodu věnovat nebudeme, cílem návodu je předat základní informace o ovládání stavebnic. Hra, která obsahuje nepřítele a počítání skóre sice na kroužku vytvořena byla, ale není součástí návodu. Měla by být součástí souborů s příklady.

```
#include <LedControl.h>

LedControl Displej=LedControl(12,10,11,1);

void setup()
{
  Displej.shutdown(0,false);// tohle zapne displej
  Displej.setIntensity(0,4);// nastavíme jas displeje na 4 (možné je 0-15)
  Displej.clearDisplay(0);// tahle funkce smaže displej
}
void loop()
{
  int radek, sloupec;
  radek = 6;

  int strelaR, strelaS; //souřadnice pro pohyb střely
  int zvukStrely; //promenná pro zvuk výstřelu
  //zde se čte hodnota potenciometru pro pohyb
  sloupec = analogRead(A7)/100;
  sloupec--;
  if (sloupec>6) sloupec=6;
  //nakreslíme koráb na souřadnice
  Displej.setLed(0, radek, sloupec+1, 1);
  Displej.setLed(0, radek+1, sloupec, 1);
  Displej.setLed(0, radek+1, sloupec+1, 1);
  Displej.setLed(0, radek+1, sloupec+2, 1);
//zde testujeme stisk tlačítka
  if (digitalRead(2)==1)
  {
    strelaS=sloupec+1; //číslo sloupce, kde je hlaveň
    strelaR=5; //začínáme odspodu
    zvukStrely=800; //nastavíme výšku zvuku střely
    do
    {
      Displej.setLed(0,strelaR, strelaS, 1);
      tone (3,zvukStrely,25);
      delay(50);
      Displej.setLed(0,strelaR, strelaS, 0);
      strelaR--;//pohyb střely
      zvukStrely=zvukStrely-50;//změna zvuku střely
    } while (strelaR>=0); //opakujeme, dokud střela nedosáhne vršku
  }
  delay (100);//chvili čekáme
  //a smažeme koráb, aby mohl vzniknout pohyb
  Displej.setLed(0, radek, sloupec+1, 0);
  Displej.setLed(0, radek+1, sloupec, 0);
  Displej.setLed(0, radek+1, sloupec+1, 0);
  Displej.setLed(0, radek+1, sloupec+2, 0);
}
```

Hra časovaná bomba

Tahle úloha je již pro pokročilé programátory. Ostatní si ji mohou opsat.

Pomocí úpravy úlohy 26 můžeme vytvořit jednoduchou hru. Na stavebnici S-arduino máme dva číselné displeje, se stavebnicí Saimon1 bychom museli použít ještě modul Saimon2 s druhým číselným displejem.

Bomba by měla vypadat takto: Na jednom číslicovém displeji bude probíhat odpočítávání od F do 0. Jako správná bomba by měla také tikat, takže zapojíme reproduktor a při každém odečtu tikneme. Bombu deaktivuje zadání správné hodnoty na druhém displeji (hodnotu můžeme v programu určit náhodně). Zvolení hodnoty pro deaktivaci provedeme pomocí potenciometru a stisknutím tlačítka potvrdíme. Pokud bude správná, odpočet skončí (může se ozvat nějaký potvrzující zvuk) a pokud bude špatná, tak se zrychlí odpočet. Pokud počítadlo dojde až na 0, tak se ozve zvuk napodobující výbuch a program skončí.

Začneme tím, že naprogramujeme jednoduché počítadlo od F do 0. Mezitím budeme číst hodnotu potenciometru, kterou zobrazíme na druhém číslicovém displeji. Ten připojíme na zdířky D06 až D09. K tomu musíme vytvořit funkci **void zobrazCislo2 (int cislo2),** která bude zobrazovat na těchto zdířkách. Na zdířku D10 připojíme reproduktor a Na D11 tlačítko, kterým budeme testovat, zda je zadaná kombinace správná. Celý kód hry je napsaný níže.

```
void zobrazCislo (int cislo)
{ //Funkce zobrazí číslo na Saimon1
  digitalWrite(2,cislo&1);
  digitalWrite(3,cislo&2);
  digitalWrite(4,cislo&4);
  digitalWrite(5,cislo&8);
}
void zobrazCisloSaimon2 (int cislo2)
{ //Funkce zobrazí číslo na Saimon1
  digitalWrite(6,cislo2&1);
  digitalWrite(7,cislo2&2);
  digitalWrite(8,cislo2&4);
  digitalWrite(9,cislo2&8);
}

void setup() {
    int krok;
  int odpocet, rychlostOdpoctu, potenciometr, kodBomby;
  int pin = 2;
  do
  { //Zde se nastaví zdířky D02 až D09 jako OUTPUT
    pinMode(pin++,OUTPUT);
  } while (pin<=9);
  pinMode (10,INPUT); //zde je připojené tlačítko pro deaktivaci

  randomSeed (analogRead(A0)); //zapne generátor náhodných čísel
  kodBomby=random (0,16) ; //vygeneruje náhodný kod pro deaktivaci
  odpocet=15;
  rychlostOdpoctu=10; //defunuje, jak rychle bomba odpočítává
```

```
 do
   {
    zobrazCislo(odpocet);
    tone(10,500,10); //tikání
    for (krok=0;krok<100;krok++)
    { //v tomto cyklu stále čteme hodnotu
     //potenciometru a porovnáváme, zda
     //bylo stisknuté tlačítko
     potenciometr=analogRead (A7)/64;
     zobrazCisloSaimon2 (potenciometr);
     if (digitalRead (11))
     { //pokud bylo stisknuté
      if (potenciometr==kodBomby)
      { //a našli jsme správný kod, hra končí
       tone (10,500,100);
       delay(100);
       tone (10,750,100);
       delay(100);
       exit(0);
      }
      else
      { //pokud jsme zadali neslrávný kod, zrychlí se odpočet
       //který ale nesmí být roven nule
       if ( rychlostOdpoctu > 1 )rychlostOdpoctu--;
       delay(100);
      }
     }
    delay(rychlostOdpoctu);
    }
    odpocet=odpocet-1;
   } while(odpocet>=0);

   //pokud je odpočet na nule, uděláme zvuk výbuchu
   //a hra také končí
   for (krok=2000;krok>100;krok=krok-10)
   {
    tone (10,krok,10);
    delay (20);
   }
}


void loop() {
}
```

A co dál? Pusťte uzdu vlastní fantazii. Pomocí znalostí z těchto příkladů budete jistě umět udělat různé jednoduché hry a programy. A pokud ne dnes, tak časem jistě určitě ano.

Typy na pokročilé hry (které napadly kluky na kroužku):

Pexeso
Videostop
Had
Tetris