

PODROBNÝ PRŮVODCE TEORIÍ

PODROBNĚ ROZEPSANÉ PŘÍKLADY S POPISEM FUNKCIONALIT OBVODŮ A PROGRAMOVÉHO KÓDU, KTERÝ JE ZAMĚŘEN NA POUŽÍVÁNÍ MATICOVÉHO LED DISPLEJE A AKCELEROMETRU. ŘEŠENÉ ÚKOLY ZOHLEDŇUJÍ NOVĚ PROBRANÉ TÉMA A STAVÝ NA PŘEDCHOZÍCH ZNALOSTECH.

OBSAH PRŮVODCE

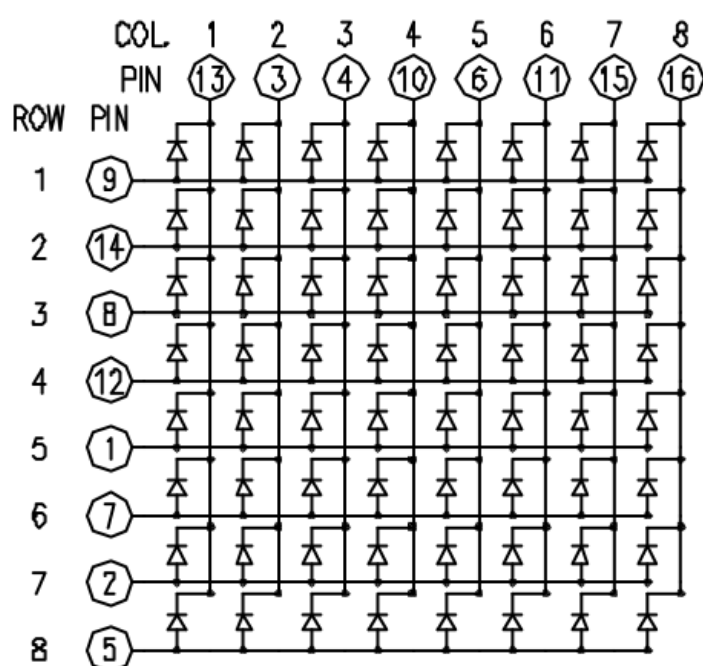
- ① Získání dovedností při zapojování maticového LED displeje.
- ② Naučení se zobrazovat symbolů na maticovém LED displeji.
- ③ Využití analogových hodnot pro ovládání diod maticového displeje.
- ④ Získání dovedností při zapojování akcelerometru.
- ⑤ Programování ovládání akcelerometru.
- ⑥ Získání dovedností při propojení maticového LED displeje a akcelerometru.

ZOBRAZOVÁNÍ SYMBOLŮ NA MATICOVÉM LED DISPLEJI

MATICOVÝ LED DISPLEJ

Maticové LED displeje, i když mají jen malé rozlišení, se používají hlavně proto, že lze s nimi dosáhnout efektního vzhledu a zobrazení jednoduché grafiky, jsou relativně velké a čitelné z větší vzdálenosti, v šeru aktivním svitem poutají pozornost.

LED displeje jsou nejčastěji reprezentovány jako matice LED diod, uspořádaných v řadách a sloupcích. Řady představují běžné anody a sloupce společné katody nebo naopak.



Obr. 1 - Schéma zapojení diod maticového displeje

Chceme-li ovládat matici, musí se propojit její řady i sloupce s mikrokontrolérem. Sloupce jsou připojeny ke katodám LED (viz Obr. 1), takže pokud mají být zapnuty LED diody v konkrétním sloupci, musí být hodnota pro každou z diod nastavena na **LOW**. Řádky jsou připojeny k anodám LED diod, takže pro jejich zapnutí musí být nastavena hodnota na **HIGH**. Pokud jsou hodnoty pro řádky i sloupce nastaveny stejně na **LOW** nebo **HIGH**, dioda se na displeji nerozsvítí.

Chcete-li ovládat jednotlivé LED diody displeje, musí se nastavit ve jejím sloupci hodnota **LOW** a řádek na **HIGH**. Má-li být ovládáno několik LED diod za sebou, musí se nastavit

řádek na **HIGH**, poté konkrétní sloupec na **LOW** nebo **HIGH**; sloupec **LOW** zapne odpovídající LED a ve sloupci **HIGH** jej vypne.

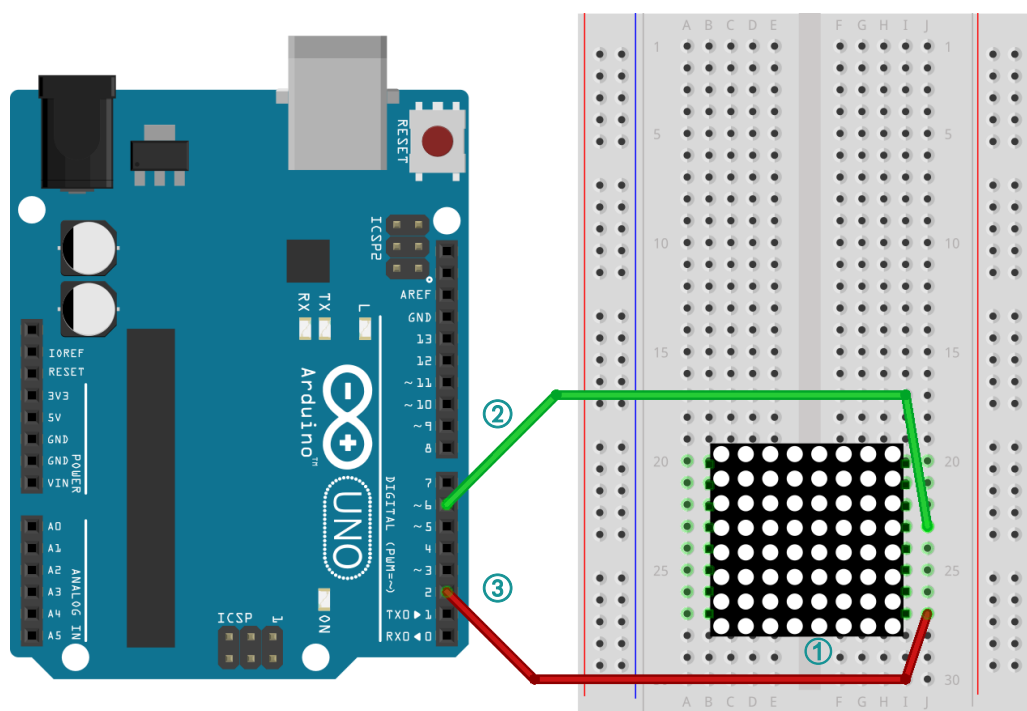


Pokud chceme rozsvítit LED na pozici [1,1], musíme připojit pin displeje 13 na + a 9 na GND. Pokud bychom ale chtěli rozsvítit zároveň bod [1,1] a [2,2], je situace trochu komplikovanější. Kdybychom totiž připojili 13 a 3 na + a 14 i 9 na GND současně, rozsvítí by se celý čtverec ([1,1], [2,1], [1,2], [2,2]). Z tohoto důvodu se vždy pracuje jen s jednou řadou (ať už jde o řádek, nebo o sloupec), rozsvítí se všechny body, které se mají zobrazit, poté se napájení řady vypne, a to samé se opakuje se všemi dalšími řadami. Pokud toto „překreslování“ probíhá dostatečně rychle, lidské oko si ničeho nevšimne (kvůli jeho setrvačnosti). Anody jsou vypnuté, pokud je na jejich pinu stav LOW, u katod je tomu naopak – vypnuté jsou při stavu HIGH. Kombinace pinů a zapojení displeje je v následující tabulce.

Matice pin	Řádek	Sloupec	Arduino pin
1	5	-	13
2	7	-	12
3	-	2	11
4	-	3	10
5	8	-	A2
6	-	5	A3
7	6	-	A4
8	3	-	A5
9	1	-	2
10	-	4	3
11	-	6	4
12	4	-	5
13	-	1	6
14	2	-	7
15	-	7	8
16	-	8	9

Tab. 2 - Rozložení pinů

ZAPOJENÍ DISPLEJE PRO JEHO OTESTOVÁNÍ



Obr. 2 – Základní zapojení LED matice

- ① Maticový displej je umístěn v kontaktním poli, tím se bude lépe propojovat s deskou Arduino.
- ② Zelený vodič připojte k pinu displeje číslo 13 a druhý konec vodiče k desce Arduino na pin 6.
- ③ Červený vodič připojte k pinu displeje číslo 9 a druhý konec vodiče k desce Arduino na pin 2.



Princip zapojení maticového displeje je patrný z výše uvedené tabulky kombinace pinů.

PROGRAMOVÝ KÓD

Programový kód je velmi jednoduchý a je podobný jako při zapojení obyčejné LED diody.

```
1  int pinA=2;
2  int pinB=6;
3
4  void setup() {
5      pinMode(pinA,OUTPUT);
6      pinMode(pinB,OUTPUT);
7      digitalWrite(pinA,HIGH);
8      digitalWrite(pinB,HIGH);
9  }
10
11 void loop() {
12     digitalWrite(pinB,LOW);
13     delay(200);
14     digitalWrite(pinB,HIGH);
15     delay(200);
16 }
```



- ① Deklarace proměnné **pinA** definuje číslo pinu na desce Arduino. V maticovém displeji je připojen na katodu diody.
- ② Deklarace proměnné **pinB** definuje číslo pinu na desce Arduino. V maticovém displeji je připojen na anodu diody.
- ③ Vyhrazení pinu pro katodu.
- ④ Vyhrazení pinu pro anodu.
- ⑤ Nastavení hodnoty na **HIGH**.
- ⑥ Nastavení hodnoty na **HIGH**. Pokud je na anodě a katodě stejná hodnota, tak dioda nesvítí.
- ⑦ Na anodu je přivedena hodnota **LOW**, tím dojde k rozsvícení diody.
- ⑧ Dioda svítí 200ms.
- ⑨ Na diodu je přivedena hodnota **HIGH**, tím dioda opět zhasne.
- ⑩ Dioda zhasne na 200ms.



Nezapomeňte program zkompileovat a nahrát do desky Arduino. Pokud je vše v pořádku, měla by blikat první dioda, tj. v levém horním rohu.



NEJDE NAHRÁT KÓD DO DESKY

USB kabel – ujistěte se, že máte desku Arduino připojenou k počítači.

Správný port – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu. **Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.

DIODA NA DISPLEJI NEBLIKÁ

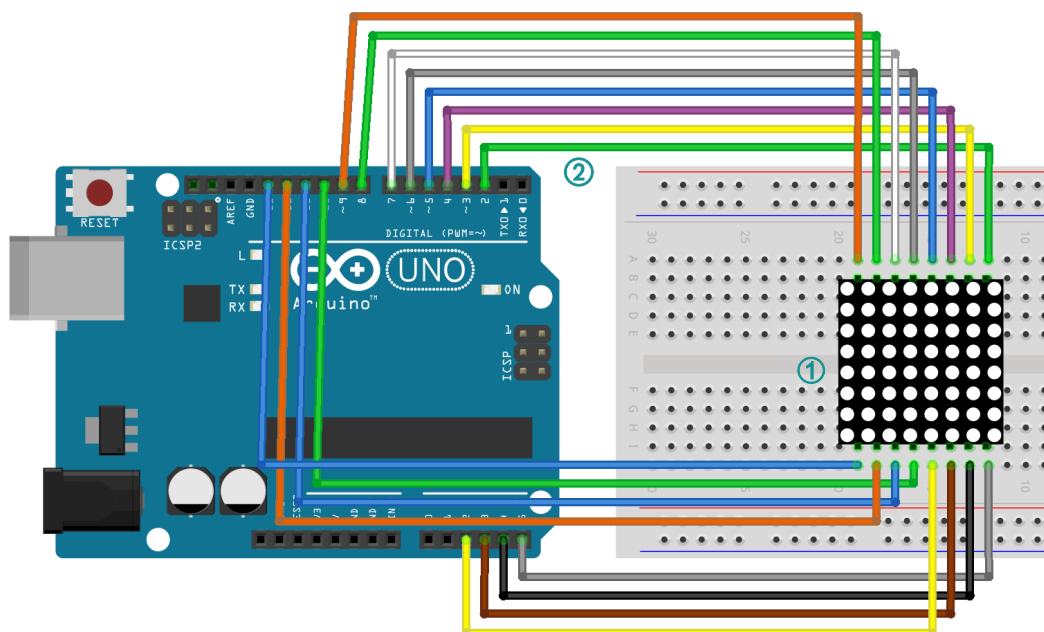
Kontrola zapojení – ujistěte se, že jsou vodiče opravdu zapojeny do správných pinů na kontaktním poli a desce Arduino.



(Př. 1) Upravte obvod zapojení displeje a programový kód předchozího příkladu tak, aby blikaly i diody ve všech rozích stejně jako dioda první.

(Př. 2) Změňte programový kód předchozího příkladu tak, aby diody v protilehlých rozích blikaly střídavě.

ÚPLNÉ ZAPOJENÍ DISPLEJE



Obr. 3 – Základní zapojení LED matice

- ① Maticový displej je umístěn v kontaktním poli, tím se bude lépe propojovat s deskou Arduino.
- ② Pro propojení displeje s deskou Arduino je potřeba 16 vodičů. Zapojení je velmi jednoduché. Při zapojení si lze vzít na pomoc rozložení pinů v tabulce Tab. 2 - Rozložení pinů.



Na uvedeném zapojení lze provést celou řadu příkladů. Proto by bylo dobré, aby tento obvod mohl být složen i do příštích hodin a vy jste se věnovali pouze programování.

PROGRAMOVÝ KÓD – POSTUPNÉ ROZSVĚCOVÁNÍ DIOD DISPLEJE

Rozsvěcování jednotlivých diod na maticovém displeji, lze udělat několika různými způsoby. Resp. lze vytvořit mnoho různých světelných kombinací. Zde je uvedena varianta, kdy se v každém řádku postupně rozsvěčují diody.

```
1  const int row[8] = {                               ①
2      2, 7, 19, 5, 13, 18, 12, 16
3  };
4
5  const int col[8] = {                                 ②
6      6, 11, 10, 3, 17, 4, 8, 9
7  };
8
9  void setup(){                                       ③
10     for(int i = 0; i < 8; i++){
11         pinMode(col[i], OUTPUT);
12         pinMode(row[i], OUTPUT);
13         digitalWrite(col[i], HIGH);                 ④
14         digitalWrite(row[i], LOW);                  ⑤
15     }
16 }
17
18 void loop(){
19     for(int j = 0; j<8;j++) {
20         digitalWrite(col[j],LOW);                   ⑥
21         for(int k = 0;k<8;k++){
22             digitalWrite(row[k],HIGH);              ⑦
23             delay(200);                             ⑧
24         }
25         for(int i = 0;i<8;i++){                      ⑨
26             digitalWrite(row[i],LOW);
27             digitalWrite(col[i],HIGH);
28         }
29     }
30 }
```

- ① Deklarace pole **row[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro řádky displeje.
- ② Deklarace pole **col[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro sloupce displeje.

- ③ Využití cyklu **for** k nastavení pinů pro sloupce a řádky jako výstupních.
- ④ Vymazání (zhasnutí) sloupců displeje.
- ⑤ Vymazání (zhasnutí) řádků displeje.
- ⑥ V cyklu **for** se postupně nastavují jednotlivé sloupce na hodnotu **LOW**.
- ⑦ V následujícím cyklu **for** se pro každý řádek v aktuálním sloupci nastavuje hodnota **HIGH**, čímž dojde k rozsvícení diody.
- ⑧ Aby bylo vidět postupné rozsvěcování diod je nastavena pauza na 200ms.
- ⑨ Ostatní sloupce a řádky se vypnou (zhasnou).



Nezapomeňte program zkompileovat a nahrát do desky Arduino. Pokud je vše v pořádku, měla by blikat první dioda, tj. v levém horním rohu.



(Př. 3) Upravte (optimalizujte) programový kód tak, aby se aktualizace a mazání displeje prováděla ve dvou vámi deklarovaných funkcích.

(Př. 4) Upravte programový kód tak, aby se v celém, rozsvíceném displeji postupně posouval vypnutý sloupec a při tomto vypnutém sloupci projížděl vypnutý řádek.

PROGRAMOVÝ KÓD – ZOBRAZOVÁNÍ SYMBOLŮ

Maticový displej, při vhodné kombinaci rozsvícených diod, může zobrazovat jednoduché symboly.

```
1  const int row[8] = {
2      2, 7, 19, 5, 13, 18, 12, 16
3  };
4
5  const int col[8] = {
6      6, 11, 10, 3, 17, 4, 8, 9
7  };
8
9  byte image[8][8] = {
10     {0,0,0,0,0,0,0,0},
11     {0,1,1,0,0,1,1,0},
12     {1,0,0,1,1,0,0,1},
13     {1,0,0,0,0,0,0,1},
14     {1,0,0,0,0,0,0,1},
15     {0,1,0,0,0,0,1,0},
16     {0,0,1,0,0,1,0,0},
17     {0,0,0,1,1,0,0,0}};
18
19  void setup(){
20      for(int i = 0; i < 8; i++){
21          pinMode(col[i], OUTPUT);
22          pinMode(row[i], OUTPUT);
23          digitalWrite(col[i], HIGH);
24          digitalWrite(row[i], LOW);
25      }
26  }
27
28  void loop(){
29      refreshScreen();
30  }
31
32  void refreshScreen(){
33      for(int j = 0; j<8;j++){
34          digitalWrite(col[j], LOW);
35          for(int k = 0; k<8; k++){
36              digitalWrite(row[k], image[k][j]);
37          }
38          Clear();
39      }
40  }
```

①

②

③

④

⑤

⑥

⑦

⑧

⑨

⑩

```

41
42 void Clear(){
43     for(int i = 0; i<8; i++){
44         digitalWrite(row[i],LOW);
45         digitalWrite(col[i],HIGH);
46     }
47 }
48

```

⑪

- ① Deklarace pole **row[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro řádky displeje.
- ② Deklarace pole **col[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro sloupce displeje.
- ③ Deklarace dvourozměrného pole **image[8][8]**, které obsahuje definici řádků a sloupců displeje. Rozsvícené diody jsou reprezentovány hodnotou 1 a zhasnuté hodnotou 0.
- ④ Využití cyklu **for** k nastavení pinů pro sloupce a řádky jako výstupních.
- ⑤ Vymazání (zhasnutí) sloupců a řádků displeje.
- ⑥ Volání funkce **refreshScreen**, která vykresluje symbol na displej.
- ⑦ V cyklu **for** se postupně nastavují jednotlivé sloupce na hodnotu **LOW**.
- ⑧ V následujícím cyklu **for** se pro každý řádek v aktuálním sloupci.
- ⑨ Nastavuje se hodnota z pole **image[k][j]**, kde hodnota **HIGH = 1** a **LOW = 0**. Tím dojde k rozsvícení/zhasnutí aktuální diody.
- ⑩ Volání funkce **Clear**, která zhasne diody mimo symbol.
- ⑪ Deklarace funkce **Clear**.



Definice tvaru symbolů je velmi snadné. Můžete využít nástroj, pomocí něhož si symbol „naklikáte“ a následně použijete vygenerované dvourozměrné pole vypnutých/zapnutých diod, které vložíte do programového kódu.

Odkaz: <https://www.prf.jcu.cz/generator-led-matrix/index.htm>



Nezapomeňte program zkompileovat a nahrát do desky Arduino. Pokud je vše v pořádku, zobrazí se na displeji symbol srdce.

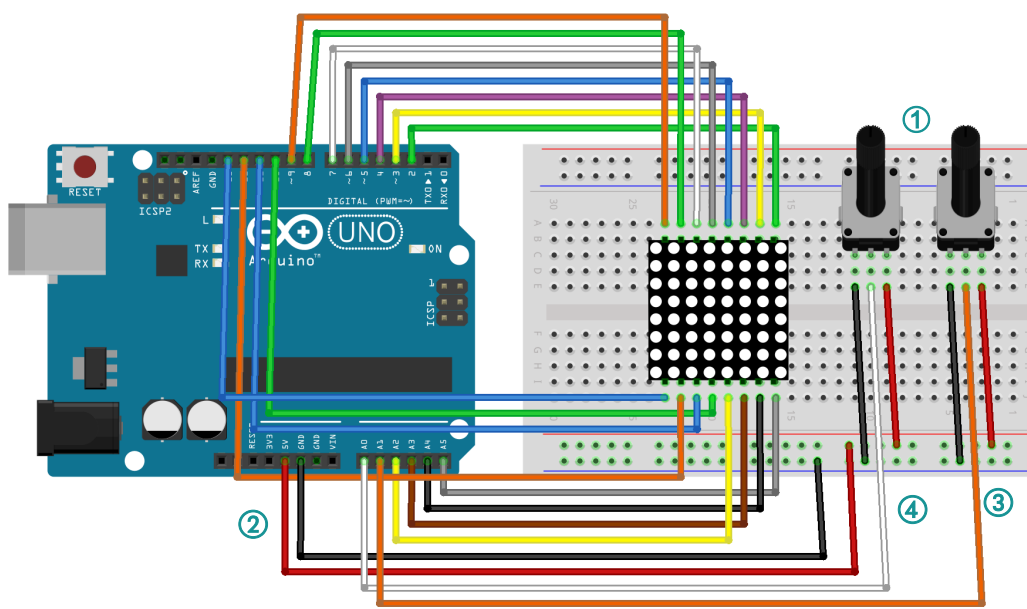


(Př. 5) Vyzkoušejte si zobrazit další symboly. Může to být smajlík, trojúhelník, dva kruhy v sobě atd.

(Př. 6) Změňte programový kód tak, aby se střídavě zobrazovalo velké a malé srdce.

OVLÁDÁNÍ DIOD MATICOVÉHO DISPLEJE POMOCÍ HODNOT Z ANALOGOVÉHO VSTUPU

Pro vstup se použijí dva potenciometry. Jeden bude ovládat pohyb rozsvícené diody v řádcích a druhý ve sloupcích.



Obr. 4 – Základní zapojení LED matice

- ① Potenciometry jsou připojeny přímo do kontaktní desky.
- ② Na kontaktní desku je přivedeno napájení **5V** a zemní vodič **GND** přímo z desky Arduino.
- ③ Signál z prvního potenciometru je přiveden na analogový vstup desky Arduino – **A1**.
- ④ Signál z druhého potenciometru je přiveden na analogový vstup desky Arduino – **A0**.



Zapojení maticového displeje je stejné, jako u předchozích příkladů, proto lze využít již zapojený obvod maticového displeje a přidat pouze potenciometry, což je již opakování. Studenti zapojení mohou provést v rámci samostatné úlohy.

PROGRAMOVÝ KÓD – ANALOGOVÉ VSTUPY

Při programování ovládání maticového displeje pomocí analogových vstupů, lze opět vyjít z předchozích programů. Samotné doprogramování ovládání je pak velice jednoduché.

```
1  const int row[8] = {
2      2, 7, 19, 5, 13, 18, 12, 16
3  };
4
5  const int col[8] = {
6      6, 11, 10, 3, 17, 4, 8, 9
7  };
8
9  int pixels[8][8];
10
11 int x = 5;
12 int y = 5;
13
14 void setup(){
15     for(int i = 0; i < 8; i++){
16         pinMode(col[i], OUTPUT);
17         pinMode(row[i], OUTPUT);
18         digitalWrite(row[i], LOW);
19     }
20
21     for(int x = 0; x < 8; x++) {
22         for(int y = 0; y < 8; y++) {
23             pixels[x][y] = HIGH;
24         }
25     }
26 }
27
28 void loop(){
29     readSensors();
30     refreshScreen();
31 }
32
33 void readSensors(){
34     pixels[x][y] = HIGH;
35     x = 7 - map(analogRead(A0), 0, 1023, 0, 7);
36     y = map(analogRead(A1), 0, 1023, 0, 7);
37     pixels[x][y] = LOW;
38 }
39
40
```

Diagrammatic annotations on the right side of the code:

- ① points to line 1.
- ② points to line 5.
- ③ points to line 9.
- ④ points to line 11.
- ⑤ points to line 23.
- ⑥ points to line 29.
- ⑦ points to line 33.
- ⑧ points to line 35.
- ⑨ points to line 36.
- ⑩ points to line 37.

```

41 void refreshScreen(){
42     for(int j = 0; j<8;j++){
43         digitalWrite(row[j], HIGH);
44         for(int k = 0; k<8; k++){
45             int thisPixel = pixels[j][k];
46             digitalWrite(col[k], thisPixel);
47             if (thisPixel == LOW) {
48                 digitalWrite(col[k], HIGH);
49             }
50         }
51         digitalWrite(row[j], LOW);
52     }
53 }

```

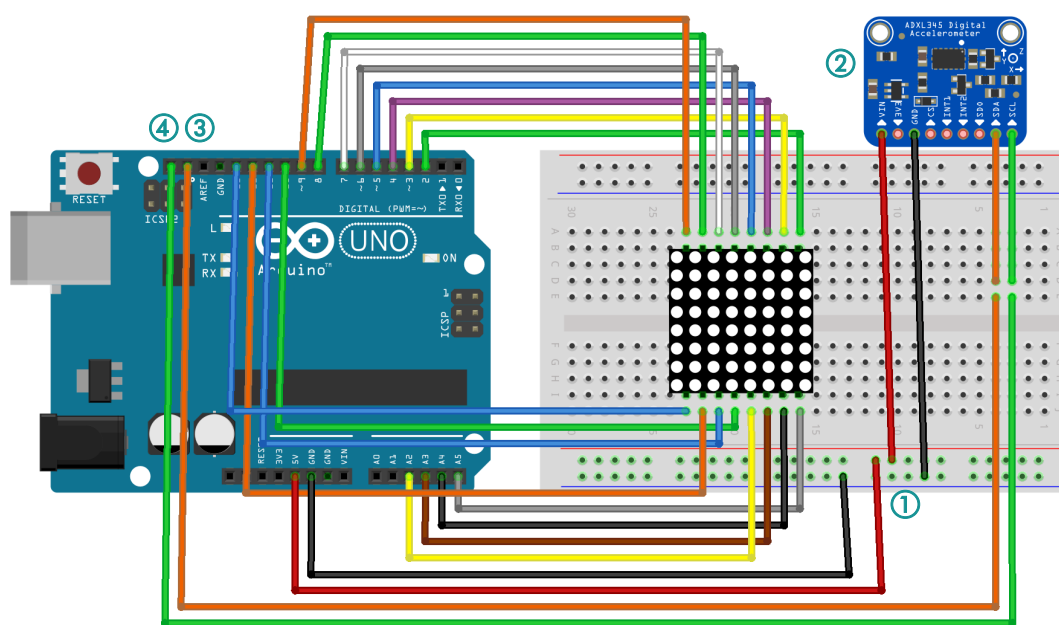
- ① Deklarace pole **row[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro řádky displeje.
- ② Deklarace pole **col[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro sloupce displeje.
- ③ Deklarace dvourozměrného pole **pixels[8][8]**, které obsahuje pozici rozsvícené diody.
- ④ Výchozí pozice rozsvícené diody při spuštění programu. Následuje již známá inicializace pinů pro výstup a vypnutí všech diod.
- ⑤ Inicializace matice **pixels**.
- ⑥ Volání funkce **readSensors()**, která čte hodnoty z potenciometrů.
- ⑦ Deklarace funkce **readSensors()**.
- ⑧ Namapování hodnoty z potenciometru pro souřadnici **x**.
- ⑨ Namapování hodnoty z potenciometru pro souřadnici **y**.
- ⑩ Nastavení nové pozice bodu tak, aby se LED dioda rozsvítila.
- ⑪ Projít přes řádky displeje.
- ⑫ Nastavení konkrétního bodu v řádku na hodnotu **HIGH**.
- ⑬ Získat hodnotu aktuálního bodu.
- ⑭ Pokud je hodnota aktuálního bodu pro řádek **HIGH** a pro sloupec **LOW**, dioda se rozsvítí.
- ⑮ Vypnutí bodu.
- ⑯ Vypnutí celého řádku.



(Př. 7) Dokázali byste pomocí potenciometrů a maticového displeje vytvořit klasickou hru Ping Pong? Pokud byste si nevěděli rady, tak vzorové řešení je k dispozici na GitHub.

SPOJENÍ MATICOVÉHO DISPLEJE A AKCELEROMETRU

Pro spojení maticového displeje a akcelerometru lze vyjít z předchozího příkladu. Na základě předchozích kapitol již víte, jak maticový displej používat. Největším problémem je tak zapojení a programování akcelerometru, jako zdroje signálu.



Obr. 5 – Zapojení LED matice a akcelerometru

- ① Na kontaktní desku je přivedeno napájení **5V** a zemní vodič **GND** přímo z desky Arduino.
- ② Akcelerometr je připojen pouze na vodiče, aby se sním dalo snadno pohybovat. Z akcelerometru se připojí pin **VIN** (tento pin může být také značen jako VCC, VCC_IN) do kontaktního pole k napájení. Pin **GND** se připojí do kontaktního pole na zem.
- ③ Z akcelerometru se dále připojí pin **SDA** do desky Arduino. Na desce Arduino tento pin bývá označen stejným názvem, na spodní straně desky.
- ④ Stejně se do desky Arduino připojí i druhý datový pin **SCL**.



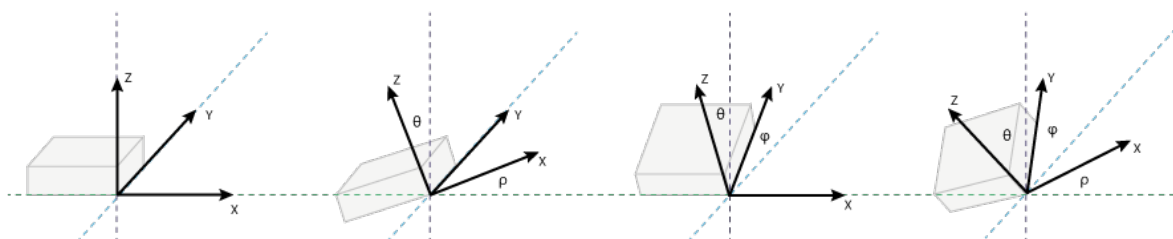
Zapojení maticového displeje je stejné, jako u předchozích příkladů, proto lze využít již zapojený obvod maticového displeje a přidat pouze akcelerometr.

CO TO JE AKCELEROMETR A JAK PRACUJE

Akcelerometr je malé pohybové čidlo, které měří pohybové zrychlení a to nejlépe ve všech třech osách. Ze znalosti zrychlení a hmotnosti lze zjistit sílu působící na těleso.

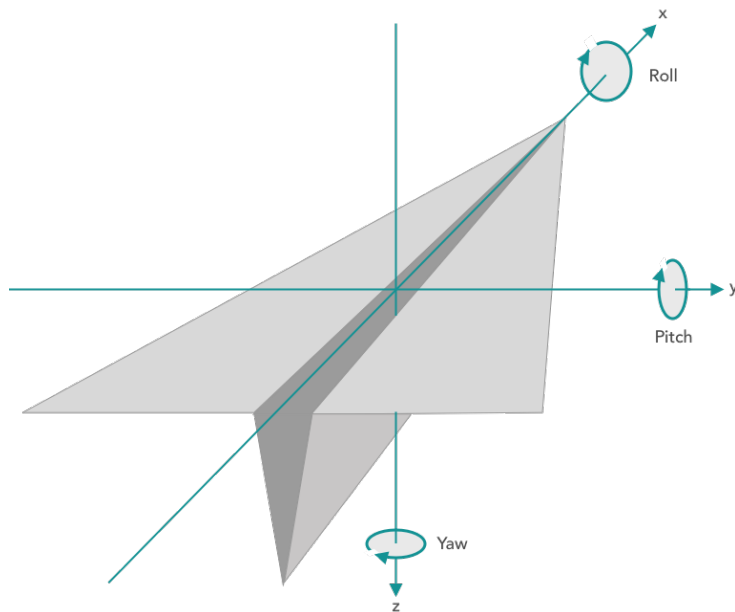
Akcelerometry jsou vhodné nejen pro měření odstředivých a setrvačných sil, ale i pro určování pozice tělesa, jeho náklon nebo vibrace. Akcelerometry jsou dnes i v mobilních telefonech a využívají se v leteckém a automobilovém průmyslu.

Aby bylo možné definovat úhly akcelerometru ve třech rozměrech **pitch**, **roll** a **theta**, využívají se všechny tři výstupy akcelerometru. **Pitch** (ró), je definováno jako úhel vzhledem k ose X a země. **Roll** (fí) je definováno jako úhel vzhledem k ose Y a země. **Theta** je úhel vzhledem k ose Z - gravitace.



$$\rho = \arctan\left(\frac{Ax}{\sqrt{Ay^2 + Az^2}}\right) \quad \varphi = \arctan\left(\frac{Ay}{\sqrt{Ax^2 + Az^2}}\right) \quad \theta = \arctan\left(\frac{\sqrt{Ax^2 + Ay^2}}{Az}\right)$$

Je důležité poznamenat, že akcelerometr poskytuje poměrně přesné údaje úhlové orientace za předpokladu, že gravitace je jediná síla působící na snímač. Nicméně, při pohybu a otáčení senzoru, mohou působit jiné síly a dochází ke kolísání přesnosti. Výsledkem potom jsou údaje obsahující šum, který způsobuje sice krátkodobé, ale významné odchylky.



Pro uvedený příklad nás budou zajímat úhly **Pitch** a **Roll**. Zobecněný vzorec pro zrychlení z naměřených hodnot akcelerometru je:

$$G_{Accel} = Raw_{Accel} \frac{Range}{2^{Resolution-1}}$$

Jakmile jsou k dispozici odpovídající hodnoty z akcelerometru, lze pokračovat ve výpočtu úhlů pomocí následujících rovnic:¹

$$pitch = \arctan\left(\frac{G_y}{\sqrt{G_x^2 + G_z^2}}\right) \quad roll = \arctan\left(\frac{-G_x}{G_z}\right)$$

Tyto vzorce lze v Arduino kódu dají přepsat v následujícím tvaru:

```
1 roll = (atan2(-Yg, Zg)*180.0)/M_PI;
2 pitch = (atan2(Xg, sqrt(Yg*Yg + Zg*Zg))*180.0)/M_PI;
```

¹ NXP: Freescale Semiconductor [online]. 2013, 2013(6) [cit. 2018-11-15]. Dostupné z: https://cache.freescale.com/files/sensors/doc/app_note/AN3461.pdf

PROGRAMOVÝ KÓD – ČTENÍ HODNOT Z AKCELEROMETRU

Programové propojení akcelerometru a maticového displeje je velmi podobné jako při zapojení s potenciometry. Složitější je pouze v tom, že se musí provést čtení dat z akcelerometru. Složitost tohoto programování je závislá na zvolené knihovně, která se použije pro spojení akcelerometru a desky Arduino.

KNIHOVNA ADXL345

→ Pro jednodušší práci a správnou funkcionalitu akcelerometru, musí být nainstalovaná podpůrná knihovna, kterou naleznete na [GitHub](#).



```
1  #include <Wire.h>
2  #include <ADXL345.h>
3
4  ADXL345 acc;
5
6  const int row[8] = {
7      2, 7, 19, 5, 13, 18, 12, 16
8  };
9
10 const int col[8] = {
11     6, 11, 10, 3, 17, 4, 8, 9
12 };
13
14 int pixels[8][8];
15
16 int x = 5;
17 int y = 5;
18
19 void setup(){
20     acc.begin();
21
22     for(int i = 0; i < 8; i++){
23         pinMode(col[i], OUTPUT);
24         pinMode(row[i], OUTPUT);
25         digitalWrite(row[i], LOW);
26     }
27
28
```

①
②
③

④

```

29 for(int x = 0; x < 8; x++) {
30     for(int y = 0; y < 8; y++) {
31         pixels[x][y] = HIGH;
32     }
33 }
34 }
35
36 void loop(){
37     readSensors();
38     refreshScreen();
39 }
40
41 void readSensors(){
42     double pitch, roll, Xg, Yg, Zg;
43     acc.read(&Xg, &Yg, &Zg);
44
45     roll = (atan2(-Yg, Zg)*180.0)/M_PI;
46     pitch = (atan2(Xg, sqrt(Yg*Yg + Zg*Zg))*180.0)/M_PI;
47
48     pixels[x][y] = HIGH;
49     x = 7 - map(roll, -20, 20, 0, 7);
50     y = map(pitch, -20, 20, 0, 7);
51     pixels[x][y] = LOW;
52 }
53
54 void refreshScreen(){
55     for(int j = 0; j<8;j++){
56         digitalWrite(row[j], HIGH);
57         for(int k = 0; k<8; k++){
58             int thisPixel = pixels[j][k];
59             digitalWrite(col[k], thisPixel);
60             if (thisPixel == LOW) {
61                 digitalWrite(col[k], HIGH);
62             }
63         }
64         digitalWrite(row[j], LOW);
65     }
66 }

```

- ① Připojení knihovny **Wire.h**, která umožňuje komunikaci se zařízeními s I2C/TWI.
- ② Připojení knihovny **ADXL345.h**, zajišťuje jednodušší komunikaci mezi deskou Arduino a akcelerometrem.
- ③ Vytvoření instance třídy ADXL 345 pro práci s akcelerometrem.
- ④ Zahájení komunikace akcelerometru s deskou Arduino.
- ⑤ Deklarace funkce **readSensor**, která provádí čtení a výpočet dat z akcelerometru.

- ⑥ Deklarace proměnných pro výpočet klopení a klonění.
- ⑦ Čtení hodnot z akcelerometru.
- ⑧ Výpočet klopení.
- ⑨ Výpočet klonění.
- ⑩ Namapování hodnoty z akcelerometru pro **klopení**.
- ⑪ Namapování hodnoty z akcelerometru pro **klonění**.



Z programového kódu je patrné, že při využití funkcí je změna kódu minimální. Prakticky došlo pouze ke změně ve funkci pro čtení ze senzorů, která spočívala v převedení získaných hodnot z akcelerometru na hodnoty použitelné k namapování pro maticový displej.

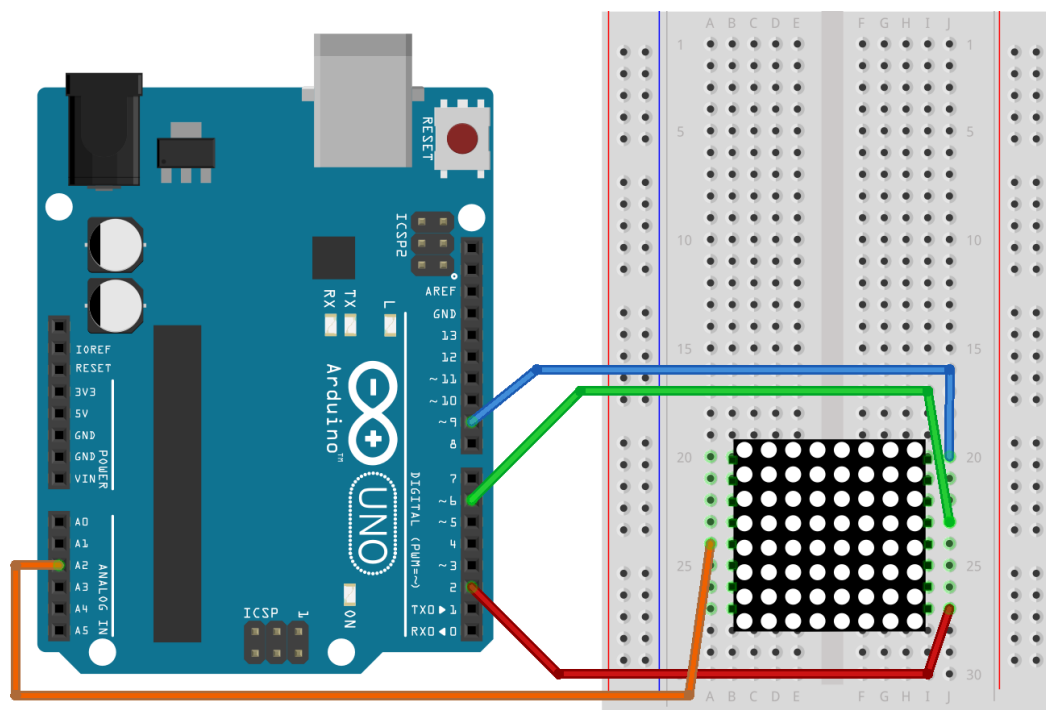


(Př. 8) Předchozí schéma a programový kód rozšiřte o zapojení dvou servomotorů, které budou reagovat na polohu akcelerometru.

ŘEŠENÍ PŘÍKLADŮ

CHYBA! NENALEZEN ZDROJ ODKAZŮ.

Změna zapojení obvodu má vést k ujasnění si principu zapojení maticového displeje. Při řešení příkladu stačí využít tabulku pinů Tab. 2 - Rozložení pinů.



Obr. 6 – Základní zapojení LED matice

```
1  int pinA=2;
2  int pinB=6;
3
4  int pinC=9;
5  int pinD=A2;
6
7  void setup() {
8      pinMode(pinA,OUTPUT);
9      pinMode(pinB,OUTPUT);
10     digitalWrite(pinA,HIGH);
11     digitalWrite(pinB,HIGH);
12
13     pinMode(pinC,OUTPUT);
```



```
14     pinMode(pinD,OUTPUT);
15     digitalWrite(pinC,HIGH);
16     digitalWrite(pinD,HIGH);
17
18 }
19
20 void loop() {
21     digitalWrite(pinB,LOW);
22     digitalWrite(pinD,HIGH);
23     delay(200);
24
25     digitalWrite(pinB,LOW);
26     digitalWrite(pinD,HIGH);
27     delay(200);
28 }
```

(PŘ. 1

K vyřešení příkladu se využije stejné zapojení jako u příkladu 1. Programový kód bude takřka stejný. Pouze se změní pořadí zapínání diod.

```
1  int pinA=2;
2  int pinB=6;
3
4  int pinC=9;
5  int pinD=A2;
6
7  void setup() {
8      pinMode(pinA,OUTPUT);
9      pinMode(pinB,OUTPUT);
10     digitalWrite(pinA,HIGH);
11     digitalWrite(pinB,HIGH);
12
13     pinMode(pinC,OUTPUT);
14     pinMode(pinD,OUTPUT);
15     digitalWrite(pinC,HIGH);
16     digitalWrite(pinD,HIGH);
17
18 }
19
20 void loop() {
21     digitalWrite(pinB,HIGH); // změna na HIGH
22     digitalWrite(pinD,LOW);  // změna na LOW
23     delay(200);
24
25     digitalWrite(pinB,LOW);
26     digitalWrite(pinD,HIGH);
27     delay(200);
28 }
```

(PŘ. 3

Původní programový kód stačí rozdělit do dvou funkcí. Např. **refreshScreen** a **Clear**.

```
1  const int row[8] = {
2      2, 7, 19, 5, 13, 18, 12, 16
3  };
4
5  const int col[8] = {
6      6, 11, 10, 3, 17, 4, 8, 9
7  };
8
9  void setup(){
10     for(int i = 0; i < 8; i++){
11         pinMode(col[i], OUTPUT);
12         pinMode(row[i], OUTPUT);
13         digitalWrite(col[i], HIGH);
14         digitalWrite(row[i], LOW);
15     }
16 }
17
18 void loop(){
19     refreshScreen();
20 }
21
22 void refreshScreen(){
23     for(int j = 0; j<8;j++){
24         digitalWrite(col[j], LOW);
25         for(int k = 0; k<8; k++){
26             digitalWrite(row[k], HIGH);
27         }
28         Clear();
29     }
30 }
31
32 void Clear(){
33     for(int i = 0; i<8; i++){
34         digitalWrite(row[i],LOW);
35         digitalWrite(col[i],HIGH);
36     }
37 }
```

(PŘ. 4

Postačí upravit pořadí zapínání diod ve funkci **refreshScreen**.

```
1
2  const int row[8] = {
3      2, 7, 19, 5, 13, 18, 12, 16
4  };
5
6  const int col[8] = {
7      6, 11, 10, 3, 17, 4, 8, 9
8  };
9
10 void setup(){
11     for(int i = 0; i < 8; i++){
12         pinMode(col[i], OUTPUT);
13         pinMode(row[i], OUTPUT);
14         digitalWrite(col[i], HIGH);
15         digitalWrite(row[i], LOW);
16     }
17 }
18
19 void loop(){
20     refreshScreen();
21 }
22
23 void refreshScreen(){
24     for(int j = 0; j<8;j++){
25         digitalWrite(row[j], LOW);
26         for(int k = 0; k<8; k++){
27             digitalWrite(col[k], HIGH);
28             delay(100);
29             digitalWrite(col[k], LOW);
30         }
31         digitalWrite(row[j], HIGH);
32     }
33 }
```

(PŘ. 5

V původním programovém kódu postačí změnit dvourozměrné pole image.



Definice tvaru symbolů je velmi snadné. Můžete využít nástroj, pomocí něhož si symbol „naklikáte“ a následně použijete vygenerované dvourozměrné pole vypnutých/zapnutých diod, které vložíte do programového kódu.

Odkaz: <https://www.prf.jcu.cz/generator-led-matrix/index.htm>

```
1 // Srdce
2 byte image[8][8] = {
3     {0,0,0,0,0,0,0,0},
4     {0,1,1,0,0,1,1,0},
5     {1,0,0,1,1,0,0,1},
6     {1,0,0,0,0,0,0,1},
7     {1,0,0,0,0,0,0,1},
8     {0,1,0,0,0,0,1,0},
9     {0,0,1,0,0,1,0,0},
10    {0,0,0,1,1,0,0,0}};
11
12 // Smajlík
13 byte image[8][8] = {
14     {0,0,1,1,1,1,0,0},
15     {0,1,0,0,0,0,1,0},
16     {1,0,1,0,0,1,0,1},
17     {1,0,0,0,0,0,0,1},
18     {1,0,1,0,0,1,0,1},
19     {1,0,0,1,1,0,0,1},
20     {0,1,0,0,0,0,1,0},
21     {0,0,1,1,1,1,0,0}};
```

(PŘ. 6

V příkladu je jednoduchá inovace, a to v podobě předávání parametru funkce **refreshScreen**. Tímto parametrem je pole s různým obrazcem.

```
1
2  const int row[8] = {
3      2, 7, 19, 5, 13, 18, 12, 16
4  };
5
6  const int col[8] = {
7      6, 11, 10, 3, 17, 4, 8, 9
8  };
9
10 // Velke srdce
11 byte image[8][8] = {
12     {0,0,0,0,0,0,0,0},
13     {0,1,1,0,0,1,1,0},
14     {1,0,0,1,1,0,0,1},
15     {1,0,0,0,0,0,0,1},
16     {1,0,0,0,0,0,0,1},
17     {0,1,0,0,0,0,1,0},
18     {0,0,1,0,0,1,0,0},
19     {0,0,0,1,1,0,0,0}};
20
21 // Male srdce
22 byte imageS[8][8] = {
23     {0,0,0,0,0,0,0,0},
24     {0,0,0,0,0,0,0,0},
25     {0,0,0,1,0,1,0,0},
26     {0,0,1,0,1,0,1,0},
27     {0,0,1,0,0,0,1,0},
28     {0,0,0,1,0,1,0,0},
29     {0,0,0,0,1,0,0,0},
30     {0,0,0,0,0,0,0,0}};
31
32 void setup(){
33     for(int i = 0; i < 8; i++){
34         pinMode(col[i], OUTPUT);
35         pinMode(row[i], OUTPUT);
36         digitalWrite(col[i], HIGH);
37         digitalWrite(row[i], LOW);
38     }
39 }
40
```

```
41 void loop(){
42     // Zobrazeni vždy po dobu 100 iteraci
43     for(int i = 0; i < 100; i++){
44         refreshScreen(image);
45     }
46
47     for(int i = 0; i < 100; i++){
48         refreshScreen(imageS);
49     }
50 }
51
52
53 void refreshScreen(unsigned char dat[8][8]){
54     for(int j = 0; j<8;j++){
55         digitalWrite(col[j], LOW);
56         for(int k = 0; k<8; k++){
57             digitalWrite(row[k], dat[k][j]);
58         }
59         delay(1);
60         Clear();
61     }
62 }
63
64 void Clear(){
65     for(int i = 0; i<8; i++){
66         digitalWrite(row[i],LOW);
67         digitalWrite(col[i],HIGH);
68     }
69 }
```