Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Využití platformy Arduino pro realizaci úkolů v Escape Games

Bakalářská práce

Lukáš Macho

Vedoucí práce: Ing. Michal Šerý, Ph.D.

České Budějovice 2019

Macho, L., 2019: Využití platformy Arduino pro realizaci úkolů v Escape Games [Using the Arduino platform to realize tasks in Escape Games. Bc. Thesis, in Czech.] – 37 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Abstrakt

Tématem této bakalářské práce bylo zaměření na návrh a realizaci elektronických zařízení s využitím platformy Arduino, které lze využít pro realizaci herních úkolů v Escape Games. Byli zvoleny modelové úkoly použitelných v Escape Games. Tyto modelové úkoly byly navrženy, realizovány a oživeny.

Klíčová slova: Arduino, Escape Games

Abstract

The topic of this bachelor thesis was the focus on design and implementation of electronic devices using the Arduino platform, which can be used for realization of game tasks in Escape Games. Modeling tasks usable in Escape Games have been chosen. These model tasks have been designed, implemented and revived.

Keywords: Arduino, Escape Games

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 17. 4. 2019

……

Lukáš Macho

Poděkování

Chtěl bych poděkovat Ing. Michalu Šerému, Ph.D. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval. Mé poděkování patří též mé rodině za pomoc a podporu během studia.

Obsah

1. Úvod

V současné době zažívají tzv. „Únikové hry“ (anglicky Escape Games) stoupající popularitu. Proto mě napadla myšlenka spojení elektroniky a zábavy, tím vznikla myšlenka na tuto bakalářskou práci, která mi přišla velice zajímavá.

Historie únikových her sahá do roku 2007, kdy byla vytvořena první úniková hra v Japonsku. V roce 2011 se začínaly šířit po celém světe. [1]

Únikové hry jsou založené na dobrodružství ve skutečné realitě, která probíhá uzavřením určitého počtu hráčů do místnosti. V této místnosti se musí hráči dobře zorientovat, najít stopy a indicie, vyřešit úkoly, hádanky. Pokud hráči ve stanoveném čase vyřeší série hádanek, úkoly a hry podaří se jim uniknout z místnosti.

Příbuznou hru lze označit francouzskou televizní show „Fort Boyard“, která se vysílala od roku 1990. Ovšem v této televizní show byla umožněna účast pouze vybraným účastníkům. V současné době na toto téma bylo natočeno i několik filmových projektů. [2]

První únikové hry nebyly příliš sofistikované. Složeny byly z jedné nebo dvou místností s nápovědami. Hádanky a úkoly nebyly komplikované a defacto neexistoval příběh (storyline). Dekorace nebyly natolik laděné, ani technická a inženýrská zařízení. Teprve postupem času se začínala více a více zapojovat technika a elektronika. Bylo potřeba vytvářet důmyslnější příběhy a propracovanější technické zázemí těchto her. Tím se technické zázemí těchto her začalo ubírat i k využití elektroniky. Zprvu to byly jednoduché obvody, které postupem času docházely možnosti. V současné době se využívají různé elektronické platformy, jakožto kontroléry a řídící prvky, které ovládají a vyhodnocují celou dějovou linii příběhu únikové hry. [2]

Já jsem se rozhodl pro využití platformy Arduino, jelikož svůj řídící program je vyvíjen zvlášť a do Arduina následně nahrán a spuštěn. Tím pádem má tvůrce únikové hry možnost si dle vlastních možností a schopností naprogramovat různé hry, hádanky a kvízy, které by vyhodnocovaly jejich správné řešení a posouvali samotný děj celé únikové hry.

V této bakalářské práci jsem jako demonstrativní příklady zvolil několik modelových úkolů. Tyto modely jsem zrealizoval a oživil a popsal vstupní a výstupní moduly.

2. Charakteristika platformy Arduino [3]

Platforma Arduino je založena na mikrokontrolerech ATmega od firmy Atmel. Svým navržením se snaží podporovat výuky informatiky na školách a seznámit studenty, jak jsou pomocí počítačů řízena různá zařízení. Jednoduchým příkladem řízení zařízení jsou například automatická pračka, mikrovlnná trouba a další zařízení. V tomto smyslu se ovšem nejedná o počítač ve smyslu stolního počítače nebo chytrého mobilu. Proto k němu nelze přímo připojit monitor, klávesnici ani myš. Ovšem na druhou stránku je připraven na připojení LED diod, displeje, servomotory, senzory, osvětlení a mnoho dalších periferií.

Arduino je open-source platforma s grafickým vývojovým prostředím, které obsahují prostředí Wiring a Processing. Platforma může být použita pro vytváření samostatných interaktivních zapojení nebo připojeno k software na počítači. V současné době lze pořídit na trhu verze, které jsou kompletní, včetně schémat a případně návrhu plošného spoje.

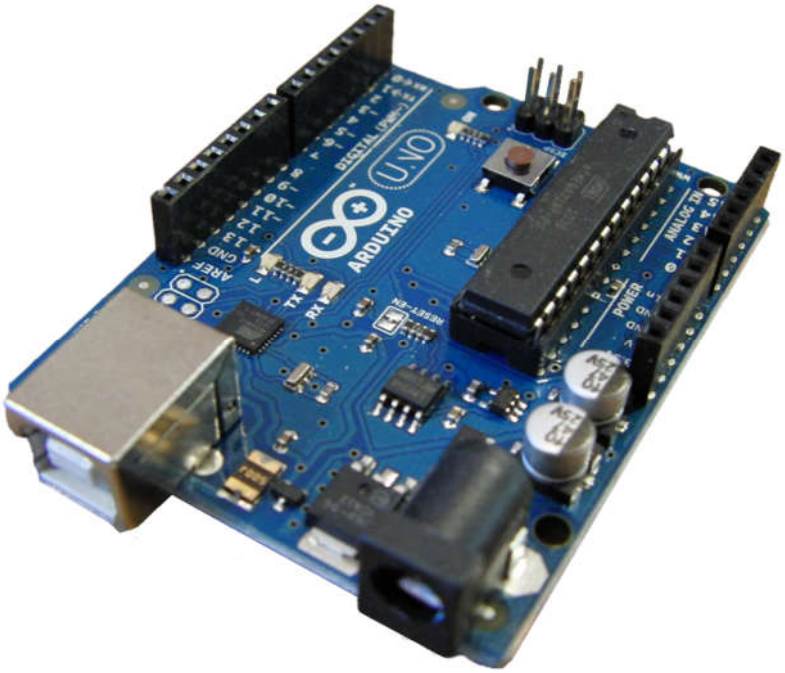
Platforma není určena jako plnohodnotný stolní počítač, ale řídící software je vyvíjen zvlášť a do platformy Arduina následně nahrán a spuštěn. Arduino opakovaně zjišťuje stav svého okolí a tím na změny reaguje. Vzhledem k tomuto faktu má nízkou spotřebu (možnost napájet baterií) a lze ho využít například pro řízení dronů, robotů a dalších zařízení.

Desky Arduino obsahují 8bitové mikrokontrolery AVR od firmy Atmel a množství dalších obvodů. Všechny desky Arduino mají většinu I/O pinů přístupné přes standardizované patice, do kterých se jednoduše zapojují další obvody. Těmto dalším obvodům se říká „Shieldy“. Desky Arduina obsahují několik diod, resetovací tlačítko, konektory pro ICSP programování, napájecí konektor, oscilátor a obvod zprostředkovávající komunikaci po USB. Základní sestava Arduino Uno obsahuje 14 I/O digitálních pinů a 6 pinů analogových. Šest z digitálních pinů je možné také použít na softwarově řízený PWM výstup.

Hlavní mikrokontroler, který je uživatelsky programovatelný, obsahuje bootloader. Bootloader je kód, který se po spuštění stará o základní nastavení mikrokontroleru. Tím jsou myšleny interní časovače, nastavení rozhraní USART a další. Dalším nastavením, který má je potřebný fuses bajty. Těmi se na nízkoúrovňové hladině nastavují určité vlastnosti čipu. Vzhledem k těmto vlastnostem se uživatel nemusí starat o detaily a svůj kód píše v jazyce, který je podobný jazyku C/C++.

I když je Arduino k počítači fyzicky připojeno přes rozhraní USB, je softwarově simulovaná komunikace přes linku RS-232.

„Shieldy“ (již výše zmíněny) jsou označovány jako rozšiřující moduly pro platformu Arduino, které se standardně připojují na dodávané piny desky Arduina. Některé „shieldy“ používají všechny piny Arduina, jiné mohou využívat jen některé.

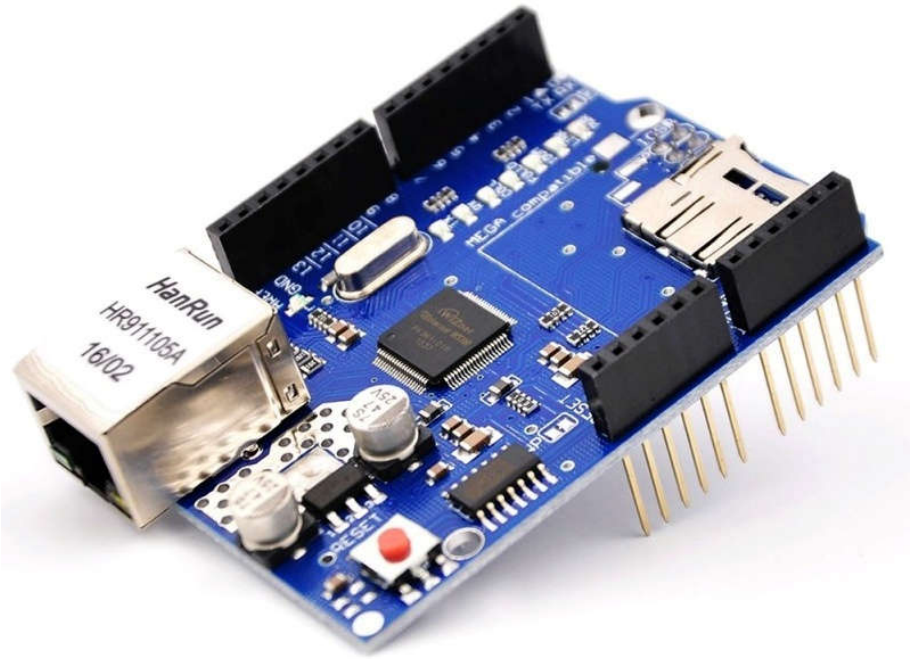


Obrázek 1: Deska Arduino UNO

2.1. Nejznámější Arduino „shieldy“

Ethernet Shield

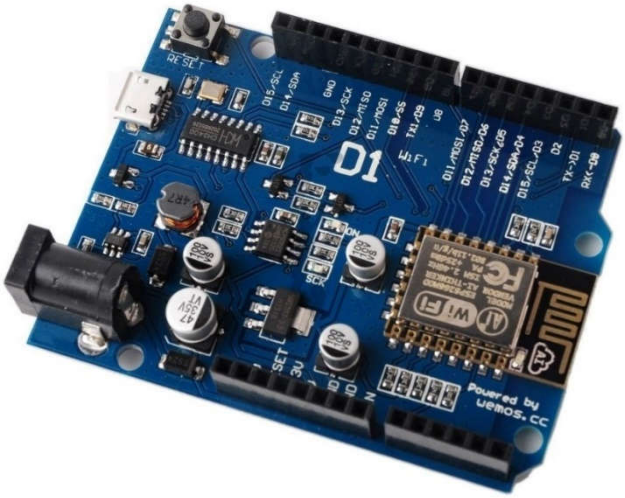
Tento modul umožňuje připojení Arduina k internetu. Obsahuje integrovaný obvod W5100, konektor RJ-45 pro připojení síťového kabelu, slot pro paměťové karty typu microSD, resetovací tlačítko a indikační LEDdiody, které signalizují činnost LAN portu. Modul využívá pro komunikaci s Arduinem piny 4, 10, 11, 12 a 13. Komunikace probíhá pomocí komunikačního protokolu SPI.



Obrázek 2: Ethernet Shield

Wifi Shield

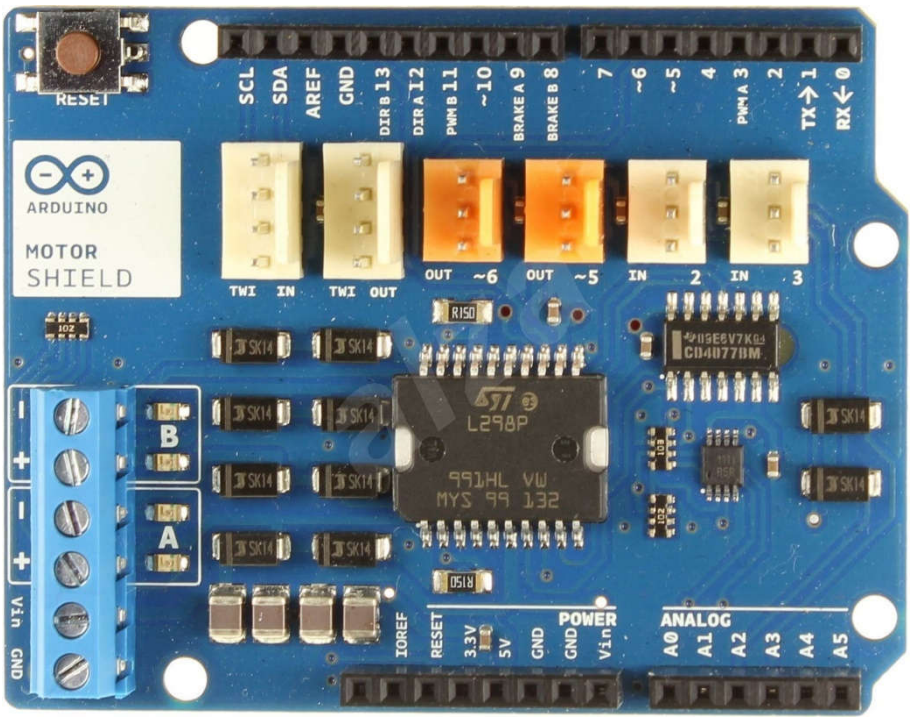
Wifi Shield modul umožňuje připojení Arduino k internetu pomocí bezdrátové sitě Wi-Fi. Jako standard se používá 802.11b a 802.11g se zabezpečením WEP a WPA2. Modul obsahuje slot pro paměťové karty typu microSD, mini USB port primárně určený pro aktualizaci firmwaru „shieldu“. Dále několik signalizačních LED diod a resetovací tlačítko. Wi-Fi modul využívá stejně jako Ethernet Shield, komunikační protokol SPI.



Obrázek 3: Wifi Shield

Motor Shield

Motor shield se používá pro pohodlné a bezpečné ovládání stejnosměrných motorů, krokových motorů a servomotorů. Pokud bychom chtěli připojit krokový motor, musíme použít dvojici konektorů vedle sebe, jsou to M1 + M2, nebo M3 + M4. S ohledem na proudovou náročnost, která je nutná pro řízení motorů, se na modulu nachází i konektor pro externí napájení. Velikost tohoto externího napájení souvisí s daným typem motoru. Rozsah externího napájecího napětí je od 4,5 V do 25 V. Maximální velikost výstupního proudu je 0,6 A, špičkově až 1,2 A.



Obrázek 4: Motor Shield

GPS Shield

Tento modul rozšiřuje platformu Arduino o GPS modul, pomocí kterého můžeme zjišťovat aktuální polohu, nadmořskou výšku a rychlost pohybu. Je velmi snadno použitelný, jelikož stačí jen přečíst data ze sériové linky a získáme aktuální GPS data. Jako většina shieldů obsahuje další možnosti a to například slot pro připojení microSD karty. Na tuto microSD kartu můžeme například uložit získaná GPS data. GPS shield podporuje obě úrovně napětí, a to 3,3 V i 5 V.

Obrázek 5: GPS Shield

3. Modelové úkoly únikových her

Jak již bylo výše zmíněno, únikové hry mohou mít různou podobu, funkčnost, dobový styl, počet místností, atd… Pro tuto práci jsem vymyslel několik modelových úkolů, které by se daly prakticky využít v dané situaci pro danou hru.

Při zapojování obvodů jednotlivých praktických úkolů jsem využíval platformu Arduino UNO a místo reálných komponent, jako jsou různá světelná zařízení, magnetických kontaktů, atd…, jsem jako demonstrativní příklad nahradil tato zařízení například LED diodami, nebo magnetické kontakty jsem nahradil tlačítkem.

3.1. Přesýpací hodiny

Myšlenku na tento úkol mě přivedl syn, když si hrál s přesýpacími hodinami a přemýšlel jsem, jakým způsobem tuto myšlenku rozvinout do reality. Celé zadání tohoto úkolu není nijak složité, naopak prosté, a přitom může plnit důležitou úlohu během hraní únikových her.

3.1.1. Teorie přesýpacích hodin

Přesýpací hodiny budou plnit úlohu po celou dobu trvání hry. Na matrici LED displeje jsou rozsvíceny horní body do tvaru trojúhelníku, který napodobuje obsah a tvar přesýpacích hodin. Pokud obsluha vpustí tým hráčů do místnosti a zavře za nimi místnost, dojde ke spuštění odpočítávání času, za který je potřeba, aby hráči vyřešily všechny úkoly v dané hře a unikly.

Jak bude plynout čas, tak budou jednotlivé body z horní poloviny matrice LED displeje zhasínat a naopak rozsvěcovat na spodní polovině matrice LED displeje. Tím bude demonstrován smysl přesýpacích hodin.

Pokud se hráčům podaří vyřešit všechny úkoly a získat všechny potřebné indicie pro nalezení klíče, který by je osvobodil v čase, tak se hodiny zastaví a ukončí. Jestliže nastane situace, kdy tým hráčů nedokáže v požadovaném čase hry projít všemi úkoly a získat potřebné indicie, přesýpací hodiny se přesypou na spodní polovinu a tím dojde k vypršení času a tým hráčů se nestává úspěšným, a tudíž se jim nemohlo podařit uniknout z místnosti.

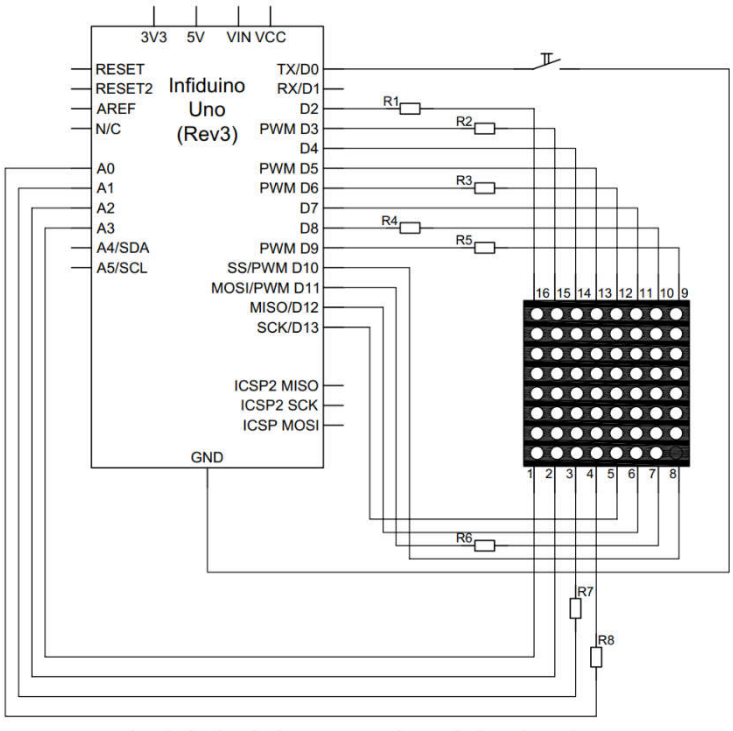
3.1.2. Technické řešení úkolu přesýpacích hodin

Abych mohl správně navrhnout obvodové schéma a zapojení komponent, následně napsat zdrojový kód, musel jsem si vhodně zvolit komponenty. Pro tuto demonstrativní verzi zapojení jsem použil desku Arduino UNO, matrice 8x8 LED displej, tlačítko, osm rezistorů 330 Ω.

Jako demonstrace přesýpacích hodin jsem pro tuto práci zvolil matrici 8x8 LED displeje, která pro názornost postačí. Ovšem pro reálné využití v únikové hře by bylo vhodnější zvolit větší matrici LED displeje pro přehlednost na větší vzdálenost.

Zároveň by bylo vhodnější pro zapnutí a vypnutí přesýpacích hodin použití magnetického kontaktu na dveřích, který by zajišťoval po zavření dveří do hrací místnosti kontakt sepnutý – tím by se začaly přesýpat hodiny. Pokud by se dveře otevřely a kontakt by se stal rozepnutý, tak by přesýpací hodiny přestaly pracovat (přesýpat). Ovšem pro naši prezentaci v této práci jsem se rozhodl magnety nahradit klasickým tlačítkem, kterým bych zajišťoval jedním stisknutím start přesýpacích hodin a opětovným stisknutím tlačítka zastavení přesýpání hodin.

Schéma zapojení:



Obrázek 6: Schéma zapojení obvodu – přesýpací hodiny

Zdrojový kód:

```
/*********************************************************
 * Bakalarska prace – Presypaci hodiny                   *
 * Autor: Lukas Macho                                    *
 * Merici a Vypocetni technika - Kombinovana forma studia *
 *********************************************************/

  #include <Arduino.h>
  #include <avr/pgmspace.h>

   unsigned long aktual_milis;
   unsigned long old_milis;
   byte citac_obr = 0;

   //cas prepinani v msec, normal = 2800msec x 20 stavu = cca 60 sec
   int cas_cykl = 2800;

    byte sloupce[] = {13,7,6,A2,4,A1,11,10};
    byte radky[] =   {A3,12,2,A0,9,3,8,5};

    //dvojrozměrné pole pro obrázek
    byte obrazek[8][8] = {  //uvodni logo , je zobrazeno po startu
    {0,0,0,0,0,0,0,0},
    {1,1,1,1,1,0,0,1},
    {0,0,1,0,1,0,0,1},
    {0,0,1,0,1,1,1,1},
    {0,0,1,0,1,0,0,1},
    {0,0,1,0,1,0,0,1},
    {0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0}
    };

    byte obrazek1[8][8]  = {  //zakladni stav pro standartni stav
    {1,1,1,1,1,1,1,1},
    {0,1,1,1,1,1,1,0},
    {0,0,1,1,1,1,0,0},
    {0,0,0,1,1,0,0,0},
    {0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0},
    {0,0,0,0,0,0,0,0}
    };
    byte obrazek21[8][8] = {   //zaverecny smile pro obraceny stav
    {0,0,1,1,1,1,0,0},
    {0,1,0,0,0,0,1,0},
    {1,0,0,1,1,0,0,1},
    {1,0,1,0,0,1,0,1},
    {1,0,0,0,0,0,0,1},
    {1,0,1,0,0,1,0,1},
    {0,1,0,0,0,0,1,0},
    {0,0,1,1,1,1,0,0}
    };
    byte obrazek22[8][8] = { //posledni stav po presypani/první pro
obraceny stav
```

```
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0},
{0,0,0,1,1,0,0,0},
{0,0,1,1,1,1,0,0},
{0,1,1,1,1,1,1,0},
{1,1,1,1,1,1,1,1}
};
byte obrazek23[8][8] = { //zaverecny smile -standartni stav
{1,1,1,1,1,1,1,1},
{1,0,1,0,0,0,0,1},
{1,1,1,1,0,1,0,1},
{1,0,1,0,0,0,0,1},
{1,1,0,0,0,0,1,1},
{1,0,1,1,1,1,0,1},
{1,0,0,0,0,0,0,1},
{1,1,1,1,1,1,1,1}
};
byte obrazek24[8][8] = { //zaverecny smile02 - standartni stav
{0,0,1,1,1,1,0,0},
{0,1,0,0,0,0,1,0},
{1,0,1,0,0,1,0,1},
{1,0,0,0,0,0,0,1},
{1,0,1,0,0,1,0,1},
{1,0,0,1,1,0,0,1},
{0,1,0,0,0,0,1,0},
{0,0,1,1,1,1,0,0}
};

void setup()
{
  for(int i = 0; i < 8; i++)
  {
    //nastavíme piny
    pinMode(sloupce[i], OUTPUT);
    pinMode(radky[i], OUTPUT);

    //zajistíme vypnutí displeje
    digitalWrite(sloupce[i], HIGH);
    digitalWrite(radky[i], LOW);

    //tlacitko - prepinac pro obrácený směr
    pinMode(0,INPUT_PULLUP);
    digitalWrite(0,HIGH);
  }
}

void loop()
{
  refresh();    // prekresleni obrázku

  /*zobrazeni obrazku--------*/
  for(int i = 0; i < 8; i++)
  {
    //zapneme řádek i
```

```
        digitalWrite(radky[i], HIGH);

        //dále pracujeme s jednotlivými sloupci
        for(int j = 0; j < 8; j++)
        {
          //pokud je ve vybraném políčku 1, rozsvítí se LED
          if(obrazek[i][j] == 1)
          {
            digitalWrite(sloupce[j], LOW);
          }
        }

        delay(2); //chvíli počkáme, aby byl obraz dostatečně jasný
(zmensuje kmitocet..)

        //vypneme všechny sloupce
        for(int j = 0; j < 8; j++)
        {
          digitalWrite(sloupce[j], HIGH);
        }

        //vypneme řádek i
        digitalWrite(radky[i], LOW);
      }
    }

    void refresh(){

      aktual_milis =  millis() - old_milis;

      if(aktual_milis <= cas_cykl)
      return; //pokud se prekroci cas=dalsi obrazek
      old_milis = millis();
      citac_obr++;

      //ob obrazku
      for(int x = 0; x < 8; x++)
      {
        for(int y = 0; y < 8; y++)
        {
          if (citac_obr == 0)
          citac_obr = 1;

          if (citac_obr ==1 && digitalRead(0)== LOW)
          citac_obr = 50;  //pokracovani pri prevraceni

           //prvni cely obrazek
          if (citac_obr == 1) obrazek[x][y] = obrazek1[x][y];

          if (citac_obr == 2)
          {
            obrazek[0][4] = 0;
            obrazek[7][4] = 1;
          }

          if (citac_obr == 3)
```

```
{
  obrazek[0][3] = 0;    //zhasnuty bit
  obrazek[6][4] = 1;    //svitici bit
}

if (citac_obr == 4)
{
  //tady se nastavuji jen zmeny v obrazku po bitech
  obrazek[1][4] = 0;
  obrazek[7][3] = 1;
}

if (citac_obr == 5)
{
  obrazek[0][5] = 0;
  obrazek[7][5] = 1;
}

if (citac_obr == 6)
{
  obrazek[0][2] = 0;
  obrazek[6][3] = 1;
}

if (citac_obr == 7)
{
  obrazek[1][3] = 0;
  obrazek[7][2] = 1;
}

if (citac_obr == 8)
{
  obrazek[0][6] = 0;
  obrazek[5][4] = 1;
}

if (citac_obr == 9)
{
  obrazek[0][1] = 0;
  obrazek[7][6] = 1;
}

if (citac_obr == 10)
{
  obrazek[1][5] = 0;
  obrazek[6][5] = 1;
}

if (citac_obr == 11)
{
  obrazek[1][2] = 0;
  obrazek[7][1] = 1;
}

if (citac_obr == 12)
{
```

```
  obrazek[0][0] = 0;
  obrazek[6][2] = 1;
}

if (citac_obr == 13)
{
  obrazek[0][7] = 0;
  obrazek[5][3] = 1;
}

if (citac_obr == 14)
{
  obrazek[1][1] = 0;
  obrazek[4][4] = 1;
}

if (citac_obr == 15)
{
  obrazek[1][6] = 0;
  obrazek[6][6] = 1;
}

if (citac_obr == 16)
{
  obrazek[2][3] = 0;
  obrazek[7][7] = 1;
}

if (citac_obr == 17)
{
  obrazek[2][4] = 0;
  obrazek[5][5] = 1;
}

if (citac_obr == 18)
{
  obrazek[2][5] = 0;
  obrazek[5][2] = 1;
}

if (citac_obr == 19)
{
  obrazek[2][2] = 0;
  obrazek[5][2] = 0;  // zhasnout minuly bod
  obrazek[7][0] = 1;
  obrazek[6][1] = 1;
}

if (citac_obr == 20)
{
  obrazek[3][3] = 0;
  obrazek[5][2] = 1;
}

 //cela pyramida
if (citac_obr == 21) obrazek[x][y] = obrazek22[x][y];
```

```
 //smile 1
if (citac_obr == 22) obrazek[x][y] = obrazek23[x][y];
 //smile 2
if (citac_obr == 23) obrazek[x][y] = obrazek24[x][y];
if (citac_obr == 24) citac_obr = 23;
 //zastaveni po prvni strane
if (citac_obr == 23 && digitalRead(0)==HIGH) citac_obr=23;
 //pokracovani pri prevraceni
if (citac_obr == 23 && digitalRead(0)==LOW) citac_obr=50;
 //prvni prevraceny - cely
if (citac_obr == 50) obrazek[x][y] = obrazek22[x][y];

if (citac_obr == 51)
{
  obrazek[0][4] = 1;
  obrazek[7][4] = 0;
}

if (citac_obr == 52)
{
  obrazek[0][3] = 1;    //zhasnuty bit
  obrazek[7][3] = 0;    //svitici  bit
}

if (citac_obr == 53)
{
  obrazek[1][4] = 1;
  obrazek[6][4] = 0;
}

if (citac_obr == 54)
{
 obrazek[0][5] = 1;
 obrazek[7][5] = 0;
}

if (citac_obr == 55)
{
  obrazek[2][4] = 1;
  obrazek[7][2] = 0;
}

if (citac_obr == 56)
{
  obrazek[1][3] = 1;
  obrazek[6][3] = 0;
}

if (citac_obr == 57)
{
  obrazek[1][5] = 1;
  obrazek[7][6] = 0;
}

if (citac_obr == 58)
{
```

```
  obrazek[0][2] = 1;
  obrazek[7][1] = 0;
}

if (citac_obr == 59)
{
  obrazek[0][6] = 1;
  obrazek[6][5] = 0;
}

if (citac_obr == 60)
{
  obrazek[1][2] = 1;
  obrazek[6][2] = 0;
}

if (citac_obr == 61)
{
  obrazek[0][1] = 1;
  obrazek[7][0] = 0;
}

if (citac_obr == 62)
{
  obrazek[0][7] = 1;
  obrazek[7][7] = 0;
}

if (citac_obr == 63)
{
  obrazek[3][4] = 1;
  obrazek[6][1] = 0;
}

if (citac_obr == 64)
{
  obrazek[1][6] = 1;
  obrazek[6][6] = 0;
}

if (citac_obr == 65)
{
  obrazek[2][5] = 1;
  obrazek[5][3] = 0;
}

if (citac_obr == 66)
{
  obrazek[1][1] = 1;
  obrazek[5][4] = 0;
}

if (citac_obr == 67)
{
  obrazek[0][0] = 1;
  obrazek[5][5] = 0;
```

```
        }

        if (citac_obr == 68)
        {
          obrazek[2][3] = 1;
          obrazek[5][2] = 0;
        }

        if (citac_obr == 69)
        {
          obrazek[2][2] = 1;
          obrazek[4][4] = 0;
        }

         //posledni cely obrazek = prvni cely
        if (citac_obr == 70) obrazek[x][y] = obrazek1[x][y];
         // smile 2 - otocenyq
        if (citac_obr == 71) obrazek[x][y] = obrazek21[x][y];
        if (citac_obr == 73) citac_obr = 72;
         // zastaveni po  strane
        if (citac_obr == 72 && digitalRead(0)==LOW) citac_obr=72;
         // pokracovani pri prevraceni
        if (citac_obr == 72 && digitalRead(0)==HIGH) citac_obr=1;
      }
    }
  }
```

3.2. Váha – určení hmotnosti

Jako zajímavý úkol mi přišlo pro únikovou hru využití váhového senzoru. Úkolem hráčů bude podle indicií a poskytnutých nápověd, najít daný předmět, který postaví na váhu. Tým hráčů ovšem nebude vědět, jaký konkrétní předmět má hledat a ani nebudou tušit kolik má vážit. Pokud budou pozorní, tak budou mít různě po místnosti umístěné nápovědi, které by je měli směřovat k danému předmětu. Hledaný předmět by byla bysta T. G. Masaryka, vážící přibližně 5kg.

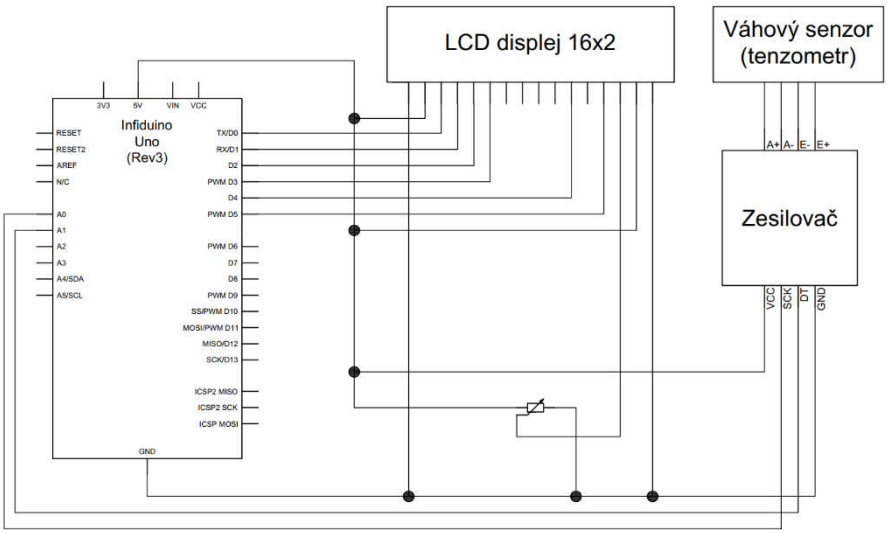
Pokud hráči bystu T. G. Masaryka najdou a postaví ji na váhu, tak se na LCD displeji vypíše text T. G. Masaryk. Tento text může znamenat pro hráče zmatenou indicii. Zvlášť, když našli bystu T. G. Masaryka a jako odpověď na úkol jim byla poskytnuta s textem T. G. Masaryk. Tím pro hráče začíná další úkol v hledání další indicie. Například by se mohla skrývat v knihovně kniha s názvem T. G. Masaryk.

3.2.1. Technické řešení úkolu s váhovým senzorem

Z technického směru mi přišlo toto zapojení obvodu velice zajímavé. Byla to moje první zkušenost s váhovým senzorem a tak trošku výzva.

Jako řídící hardware jsem použil opětovně desku Arduino UNO, váhový senzor společně se speciálním modulem zesilovače, který následně umožňuje připojení na analogové piny desky Arduina UNO.

Schéma zapojení:



Obrázek 7: Schéma zapojení obvodu s váhovým senzorem

Zdrojový kód:

```
/*****************************************************
 * Bakalarska prace - Vahovy senzor                  *
 * Autor: Lukas Macho                                *
 * Merici a Vypocetni technika - Kombinovana forma studia *
 *****************************************************/

#include <LiquidCrystal.h>
#include "HX711.h"

#define DOUT  3
#define CLK  2

float calibration_factor = -7050;

LiquidCrystal lcd(12,11,10,9,8,7,6,5,4,3,2);
```

```
void setup()
{
 scale.set_scale();
 scale.tare(); //vyresetovani na nulu

 long zero_factor = scale.read_average(); //nacist zakladni
udaje
 Serial.print("Zero factor: ");
 Serial.println(zero_factor);

 lcd.begin(16,2);      // nastaveni poctu sloupcu a radku
 lcd.setCursor(0,0);   // nastavit kurzor na sloupec 0, radek
0
}

void loop()
{
  scale.set_scale(calibration_factor); //nastevni kalibracniho
faktoru

  Serial.print("Reading: ");
  Serial.print(scale.get_units(), 1);
  Serial.print(" lbs"); //zmenit na jendotku kg a nastavit
kalibracni faktor
  Serial.print(" calibration_factor: ");
  Serial.print(calibration_factor);
  Serial.println();

  if(Serial.available())
  {
    char temp = Serial.read();

    if(temp == '+' || temp == 'a')
    calibration_factor += 10;
    else if(temp == '-' || temp == 'z')
    calibration_factor -= 10;
  }
}
```

3.3. Morseova abeceda

Využití Morseovy abecedy byla jedna z několika myšlenek, která směřovala k sestrojení dalšího modelového příkladu. Napadalo mě několik různých variant, jak využít Morseovu abecedu pro danou situaci v únikové hře a zároveň ji využitelně sestrojit pomocí platformy Arduino.

3.3.1. Teorie Morseovy abecedy

Historie Morseovy abecedy sahá do období roku 1844, kdy Samuela Morse poprvé uskutečnil telegrafické spojení mezi Washingtonem a Baltimorem.

Morseova abeceda (lidově používanější název – morseovka) je skupina kódů, která se používala v telegrafii. Jednotlivé kódy představují písmena latinky, čísel a určité specifické znaky. Tyto kódy se používají v kombinacích krátkých a dlouhých signálů. Signály mohou představoval zvukovou podobu, elektrický signál (telegraf), či světelná signalizace. Pokud se používá zvuková signalizace, je určitým zvukovým signálem – tzv. „pípáním“ krátkým (tečka), či dlouhým (čárka) sdílen její kód. Na podobném stylu funguje i světelná signalizace. Ovšem při tomto sdílení kódu se využívá světlo, které podle daného kódu se buď rozsvítí na krátkou chvíli – znamená kód tečka, či na delší chvíli – znamená kód „čárka“.

Pro správné pochopení Morseovy abecedy se používá mnemotechnická pomocná slova, která určují počáteční písmeno a délku slabik a pomáhají si zapamatovat kód písmene. V dnešní době existuje celá řada verzí Morseovy abecedy – mezinárodní kódy Morseovy abecedy, dalšími jsou americká a kontinentální verze. V této práci budu prezentovat verzi mezinárodní Morseovy abecedy.

Tabulka č.1.: Morseova abeceda

Písmeno	Kód	Pomocní slovo	Číslice	Kód	Zkrácený kód
A	.-	Akát	0	-----	-
B	-...	Blýskavice	1	.----	.-
C	-.-.	Cílovníci	2	..---	..-
D	-..	Dálava	3	...--	...-

E	.	Erb	4	....-	Není
F	..-.	Filipíny	5	.....	.
G	--.	Grónská zem	6	-....	Není
H	....	Hrachovina	7	--...	-...
CH	----	Chvátá k nám sám	8	---..	-..
I	..	Ibis	9	----.	-.
J	.---	Jasmín bílý	**Speciální znaky**		
K	-.-	Krákora	**Znak**	**Název**	**Kód**
L	.-..	Lupíneček	,	Čárka	--..--
M	--	Mává	.	Tečka	.-.-.-
N	-.	Nástup	:	Dvojtečka	---...
O	---	Ó náš pán	;	Středník	-.-.-.
P	.--.	Papírníci	?	Otazník	..--..
Q	--.-	Kvílí orkán	!	Vykřičník	--...-
R	.-.	Rarášek	-	Pomlčka	-....-
S	...	Sekera	/	Lomítko	-..-.
T	-	Trám	=	Rovnítko	-...-
U	..-	Uličník	_	Podtržítko	..--.-
V	...-	Vyvolený	„“	Uvozovka	.-..-.
W	.--	Vagón klád	(	Kulatá závorka otevírací	-.--.
X	-..-	Xénokratés	)	Kulatá závorka zavírací	-.--.-
Y	-.--	Ý se krátí	<->	Tabulátor	.----.
Z	--..	Známá žena	@	Zavináč	.--.-.

3.3.2. Úkol s Morseovou abecedou

Tento úkol v únikové hře je založení na smyslu rozluštit zašifrovanou indicii v Morseově abecedě. V místnosti se začne projevovat zvuková a světelná signalizace. Hráči musí přijít na to, že se jedná o morseovku a začít pozorně naslouchat zvukové signalizaci nebo světlené signalizaci, která jim bude šifrovaně pomocí morseovky napovídat potřebnou indicii. Pokud se stane, že v týmu není hráč, který by ovládal Morseovu abecedu, nezbývá než začít v místnosti hledat, zda se někde neskrývá nějaká nápověda k rozluštění morseovky.

Cílem je rozluštit kompletní šifrovanou zprávu v morseovce, ve které bude indicie ve tvaru číselného kódu. Pokud tento číselný kód rozšifrují a správně, tak hráči získají indicii pro další úkol v únikové hře. Ovšem k čemu a zda jim tato indicie k něčemu poslouží hráči musí zjistit sami.

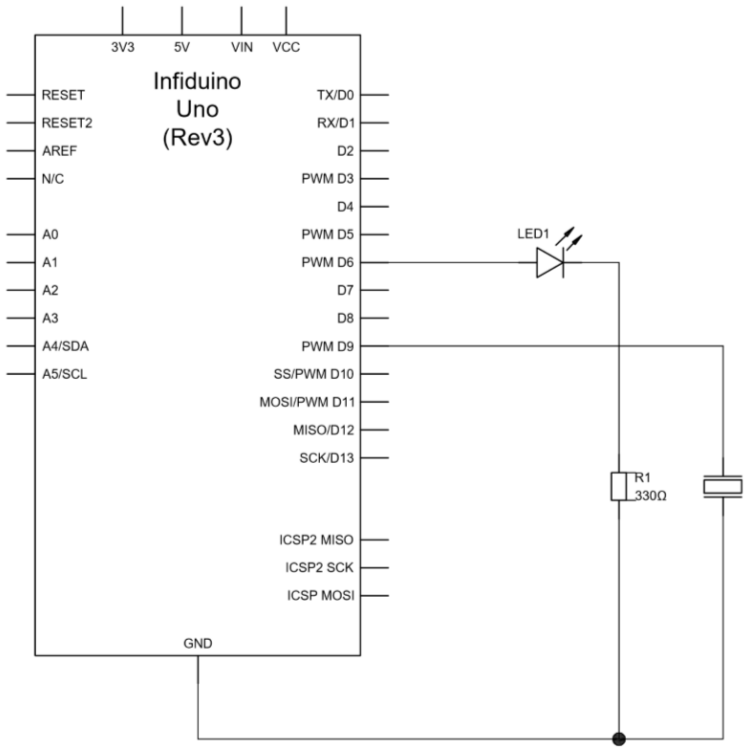
3.3.3. Technické řešení úkolu s Morseovou abecedou

V této praktické části jsem navrhl schéma zapojení komponent a připojení na jednotlivé piny k desce Arduina a zdrojový kód.

Elektronický obvod se skládá s LED diody, piezo bzučák, rezistoru 330 Ω a kontroléru Arduino UNO.

Funkce návrhu obvodu a zdrojového kódu se zakládá na principu vložení požadované indicie do zdrojového kódu, která bude převedena na morseovku. Výstupní signalizace bude řešena zvukovou (piezo bzučák) a světelnou (LED dioda) signalizací.

Schéma zapojení:



Obrázek 8: Schéma zapojení obvodu – Morseova abeceda

Zdrojový kód:

```
/*********************************************************
 * Bakalarska prace - Morseova abeceda                   *
 * Autor: Lukas Macho                                    *
 * Merici a Vypocetni technika - Kombinovana forma studia *
 *********************************************************/

const int piezoPin = 9;   //definovani cisla pinu pro piezo bzucak
const int ledPin = 6;     //definovani cisla pinu pro LED diodu
const int tonFrekvence = 523;   //definovani tonu 523

//konstanty delek tonu
const int teckaDelka = 100;
const int pomlckaDelka = teckaDelka * 3;

//casove pomlky mezi teckami, nebo pomlckami
const int meziPomlka = teckaDelka;

//casova pomlka mezi pismeny
const int mezeraP = teckaDelka * 2;

//mezera slova
const int mezeraS = teckaDelka * 4;
```

```
void setup()
{
  pinMode(piezoPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  char znak;
  if (Serial.available())
  {
    //cteni jednoho znaku
    znak = Serial.read();
    Serial.print(znak);
    if (znak>='a' && znak<='z')
    {
      //prevod na velka pismena
      znak = znak -32;
    }
    //vymena, pokud neni v A-Z
    if(znak<65 || znak>90)
    {
      znak=' ';
    }
    tonPismene(znak);
    delay(mezeraP);
  }
}
void tecka()
{
  //ton tecky
  tone(piezoPin, tonFrekvence);
  //LED dioda
  digitalWrite(ledPin, HIGH);
  delay(teckaDelka);
  noTone(piezoPin);
  //LED dioda
  digitalWrite(ledPin, LOW);
  delay(meziPomlka);
}

void pomlcka()
{
  //ton pomlcky
  tone(piezoPin, tonFrekvence);
  //LED dioda
  digitalWrite(ledPin, HIGH);
  delay(pomlckaDelka);
  noTone(piezoPin);
  //LED dioda
  digitalWrite(ledPin, LOW);
  delay(meziPomlka);
}
```

```
void tonPismene(char pismeno)
{
  //pismena jsou v poradi podle frekvence
  switch(pismeno)
  {
  case 'E':
    tecka();
    return;
  case 'T':
    pomlcka();
    return;
  case 'A':
    tecka();
    pomlcka();
    return;
  case 'O':
    tecka();
    tecka();
    pomlcka();
    return;
  case 'I':
    tecka();
    tecka();
    return;
  case 'N':
    pomlcka();
    tecka();
    return;
  case 'S':
    tecka();
    tecka();
    tecka();
    return;
  case 'H':
    tecka();
    tecka();
    tecka();
    tecka();
    return;
  case 'R':
    tecka();
    pomlcka();
    tecka();
    return;
  case 'D':
    pomlcka();
    tecka();
    tecka();
    return;
  case 'L':
    tecka();
    pomlcka();
    tecka();
    tecka();
    return;
  case 'C':
```

```
  pomlcka();
  tecka();
  pomlcka();
  tecka();
  return;
case 'U':
  tecka();
  tecka();
  pomlcka();
  return;
case 'M':
  pomlcka();
  pomlcka();
  return;
case 'W':
  tecka();
  pomlcka();
  pomlcka();
  return;
case 'F':
  tecka();
  tecka();
  pomlcka();
  tecka();
  return;
case 'G':
  pomlcka();
  pomlcka();
  tecka();
  return;
case 'Y':
  pomlcka();
  tecka();
  pomlcka();
  pomlcka();
  return;
case 'P':
  tecka();
  pomlcka();
  pomlcka();
  tecka();
  return;
case 'B':
  pomlcka();
  tecka();
  tecka();
  tecka();
  return;
case 'V':
  tecka();
  tecka();
  tecka();
  pomlcka();
  return;
case 'K':
  pomlcka();
```

```
    tecka();
    pomlcka();
    return;
  case 'J':
    tecka();
    pomlcka();
    pomlcka();
    pomlcka();
    return;
  case 'X':
    pomlcka();
    tecka();
    tecka();
    pomlcka();
    return;
  case 'Q':
    pomlcka();
    pomlcka();
    tecka();
    pomlcka();
    return;
  case 'Z':
    pomlcka();
    pomlcka();
    tecka();
    tecka();
    return;
  case '0':
    pomlcka();
    pomlcka();
    pomlcka();
    pomlcka();
    pomlcka();
    return;
case '1':
    tecka();
    pomlcka();
    pomlcka();
    pomlcka();
    pomlcka();
    return;
case '2':
    tecka();
    tecka();
    pomlcka();
    pomlcka();
    pomlcka();
    return;
case '3':
    tecka();
    tecka();
    tecka();
    pomlcka();
    pomlcka();
    return;
case '4':
```

```
    tecka();
    tecka();
    tecka();
    tecka();
    pomlcka();
    return;
case '5':
    tecka();
    tecka();
    tecka();
    tecka();
    tecka();
    return;
case '6':
    pomlcka();
    tecka();
    tecka();
    tecka();
    tecka();
    return;
case '7':
    pomlcka();
    pomlcka();
    tecka();
    tecka();
    tecka();
    return;
case '8':
    pomlcka();
    pomlcka();
    pomlcka();
    tecka();
    tecka();
    return;
case '9':
    pomlcka();
    pomlcka();
    pomlcka();
    pomlcka();
    tecka();
    return;
case ' ':
    delay(mezeraS);
    return; } }
```

3.4. Číselný kód

V kapitole 3.4. měl tým hráčů dešifrovat z Morseovy abecedy indicii. Pokud ji správně rozluštili, tak získali indicii v podobě číselného kódu, který budou muset použít pro tento další herní úkol. Ovšem to hráči netuší, k čemu získanou indicii v podobě číselného kódu mají a jak s ní naložit. Na to musí přijít…

Cílem tohoto úkolu je, že získaný číselný kód bude potřeba použit a tím způsobem, že hráči budou muset někde v místnosti najít číselnou klávesnici s LCD displejem. Pokud ji najdou a dovtípí se, jak správně zadat číselnou indicii z předchozího úkolu, získají indicii, která by je mohla osvobodit z místnosti.

Zadání číselného kódu není funkční tím způsobem, že hráči zadají čísla z indicie. Správné zadání číselného kódu je na principu psaní SMS na tlačítkovém mobilu. Pokud takto budou zadávat číselnou sekvenci, tak se jim na displeji bude postupně vypisovat textová indicie. Tato indicie by měla hráčům definitivně napovědět, kde se skrývá klíč k odemknutí místnosti a tím pádem zdolání únikové hry.

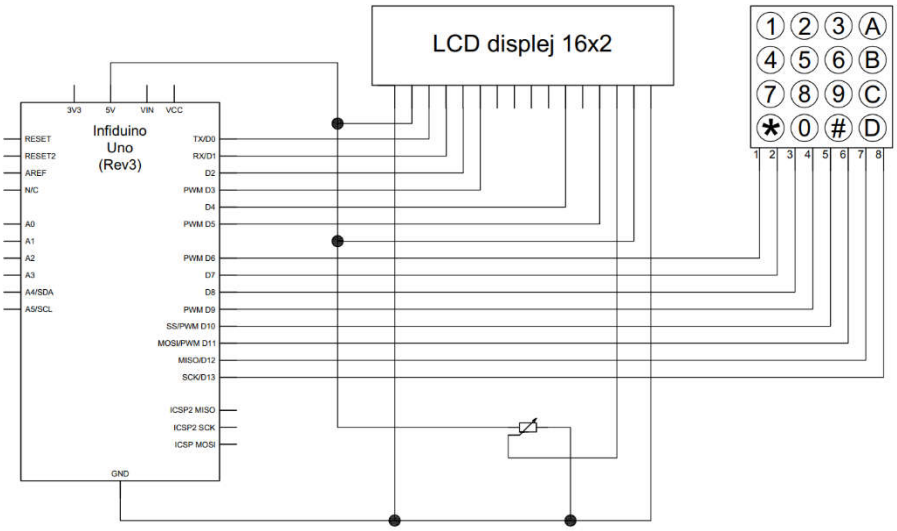
3.4.1. Technické řešení úkolu s číselným kódem

Technické řešení vychází z návrhu využitelnosti číselné klávesnice a LCD displeje. Jako řídící jednotka je použita deska Arduino UNO. Toto jsou klíčové komponenty zapojení.

Hráč bude zadávat číselnou sekvenci na číselnou klávesnici. Deska Arduino UNO bude „zachytávat“ stisknutí tlačítek číselné klávesnice. Jaké tlačítko hráč stiskl a kolikrát po sobě. Arduino UNO bude následně sledovat jaká číselná klávesa byla stisknuta a počítat kolikrát byla číselná klávesa stisknuta. Výstupem této konverze budou vypisovány jednotlivé znaky na LCD displej. Jednotlivé znaky budou postupně skládat slova a následně případně i větu.

Příklad zadání: 2-44-666-5 konverze: ahoj

Schéma zapojení:



Obrázek 9: Schéma zapojení obvodu – číselný kód

Zdrojový kód:

```
/*************************************************************
 * Bakalarska prace - Ciselny kod                            *
 * Autor: Lukas Macho                                        *
 * Merici a Vypocetni technika - Kombinovana forma studia    *
 *************************************************************/

#include <Keypad.h> //knihovnu Keypad
#include <LiquidCrystal.h>

const byte ROWS = 4; //řádky
const byte COLS = 3; //sloupce

char hexaKeys[ROWS][COLS] = {  //3D pole s klávesami
  {'1','2','3'},  //první řádek klávesnice
  {'4','5','6'} , //druhý řádek
  {'7','8','9'},  //třetí
  {'A','0','B'}   //a čtvrtý
};

byte rowPins[ROWS] = {30,31,32,33}; //piny radku
byte colPins[COLS] = {40,41,42};    //piny sloupcu

Keypad klavesnice = Keypad(makeKeymap(hexaKeys), rowPins, colPins,
ROWS, COLS);
//vytvoreni objektu klavesnice          znaky |pin
radky|p.sloupce|radku|sloupcu

String vstup; //promenna pro ukladani vstupu z klávesnice

LiquidCrystal lcd(12,11,10,9,8,7,6,5,4,3,2);
```

```
void setup()
{
  Serial.begin(9600);

  lcd.begin(16,2);      // nastaveni poctu sloupcu a radku
  lcd.setCursor(0,0);   // nastavit kurzor na sloupec 0, radek 0
}

void loop()
{
  // nastavení výpisu na první znak, první řádek
  lcd.setCursor(0, 0);
  lcd.print("Stisknuto:      ");
  lcd.print(analogData);
  // vytisteni textu na displej pomoci zavolani funkce
  // s predanim aktualne zmerene hodnoty na vstupu
  lcd.print(nactiTlacitka(analogData));
  delay(1000);
  // pauza před koncem smyčky
  delay(100);
    }
    else
    {
      vstup += stisknutyznak;
    }
  }

  String nactiTlacitka(int analog)
  {
    // promenna pro ulozeni textu pro vypis
    String text;
    if (analog < 50) text = "Vpravo(RIGHT)";
    if ((analog > 700) && (analog < 1024)) text = "              ";
    if ( (analog > 95) && (analog < 150) ) text = "Nahoru(UP)";
    if ( (analog > 250) && (analog < 350) ) text = "Dolu(DOWN)";
    if ( (analog > 400) && (analog < 500) ) text = "Vlevo(LEFT)";
    if ( (analog > 600) && (analog < 750) ) text = "Vyber(SELECT)";
    // vraceni textu jako vystup funkce
    return text;
  }
}
```

4. Závěr

Hlavním cílem této bakalářské práce byl záměr na návrh a realizaci elektronických zařízení, které lze využít pro realizaci herních úkolů v Escape Games (česky Únikové hry).

I přes to, že jsem nerealizoval návrh elektronického zařízení na skutečných elektronických zařízení v reálné únikové hře, ale pouze tvořil prototyp se stejnou funkcionalitou, tak mi tato práce výrazně rozšířila vědomosti z oblasti aplikace mikroprocesorových systémů a doplnila teoretickou výuku.

Tato práce může být inspirací pro tvořitele tzv. Escape Games, jak si více z automatizovat jejich herní místnosti v únikových hrách. Tato práce byla založena na jednoduchých příkladech a sloužila jako ukázka možností, jakým dalším způsobem lze únikové hry (místnosti) obohacovat.

5. Seznam obrázků

6. Citovaná literatura

[1] „QUESTERLAND“, [Online]. Available: https://questerland.cz/

[2] „Wikipedia“, [Online]. Available: https://cs.wikipedia.org/wiki/Úniková_hra

[3] „Wikipedia“, [Online]. Available: https://cs.wikipedia.org/wiki/Arduino#Charakteristika