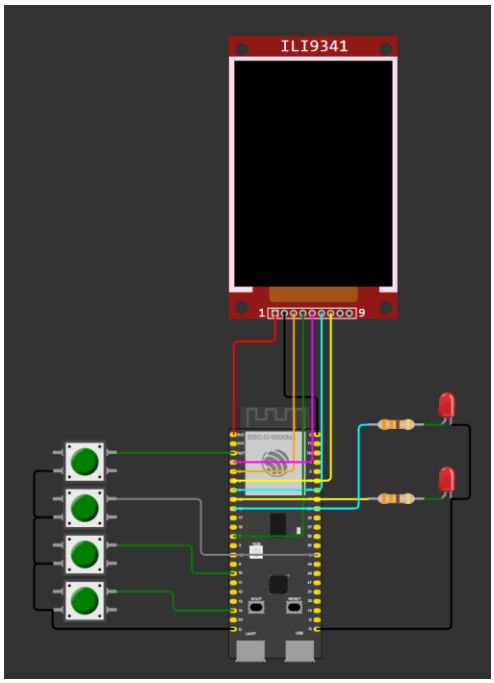
Protokol o semestrálním projektu z předmětu Elektronika a komunikace 2024

Název projektu: Flappy EK

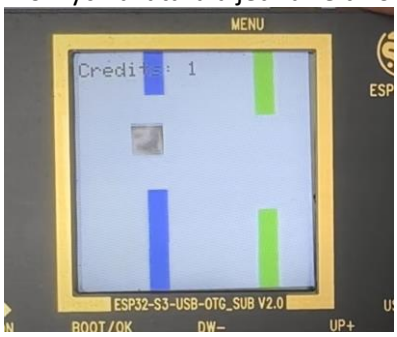
Autor: Adam Paleček

Popis zapojení:
Zapojení slouží zároveň jako výstup informací pro uživatele, tak i jako způsob, jak uživatel může zařízení ovládat. Hlavním prvkem je mikrokontroler ESP32-S3, ke kterému je připojen display ST7789, který komunikuje po sběrnici SPI. Uživatel zařízení ovládá pomocí čtyř tlačítek, první je resetovací tlačítko, které je připojeno rovnou na enable pin esp, a při stisknutí zařízení hardwarově resetuje. Následující 3 tlačítka už jsou připojena pouze na standardní GPIO piny s využitím integrovaného pull up rezistoru. Nakonec jsou k ESP připojeny pouze dvě LED, pro signalizaci opakování LOOP smyčky a indikaci vstupu od uživatele.

Schéma zapojení:



Popis Funkčnosti:
Cílem hry je dostat co nejvyšší skóre, které představuje vysokoškolské kredity, tím, že se hráč trefuje do mezer mezi překážkami, jediné ovládací tlačítko hry je UP, díky kterému hráč poskočí výše. Pokud hráč narazí do překážky, tak je to ihned konec hry a přehraje se animace výbuchu. Pokud hráč dosáhl nového nejvyššího skóre, tak se skóre ukládá do flash paměti, a tedy si ho hra pamatuje i po vypnutí, každá obtížnost si udržuje svoje vlastní nejvyšší skóre. Funkcí navíc je možnost zvolit si jednoho z pěti možných avatarů a jednu ze tří obtížností, které byli pojmenovány dle předmětů na FELu.



Hlavní. ino soubor:
vše ostatní zde: https://github.com/AdamPalecek/Flappy_EK

```
#include "bitmaps.h" //obsahuje všechny obrázky

#include <TFT_eSPI.h> //knihovna ovládající display

#include "SPI.h"  //knihovna zajistujici komunikaci přes spi

#include <Preferences.h>  //ukladani dat pri resetu


// zakladni barvy

#define BLACK    0x0000

#define BLUE     0x001F

#define RED      0xF800

#define GREEN    0x07E0

#define CYAN     0x07FF

#define MAGENTA  0xF81F

#define YELLOW   0xFFE0

#define WHITE    0xFFFF


//piny LED

#define LED_GREEN 15

#define LED_YELLOW 16


//piny tlacitka

#define OK 0

#define DOWN 11

#define UP 10

#define MENU 14


//vnitrni promenne hry

int break_height = 75;

int speed = 30;

int jump_pixels = 35;

int fall_pixels = 2;

int wall_speed = 1;


//zakladni nastaveni hry

#define hit_box 10

#define player_pixels 32


//pohyb avatar
```

```
int up_count = 0;
int fall_count = 0;

//vyber avatara
int avatar = 0;

//rozmery avatar
int EK_height = 90;
int EK_height_old = 90;

//prekazky
bool first_obstacle = 0;
int obstacle1_break = 0;
int obstacle2_break = 0;
int obstacle1_pos = 0;
int obstacle2_pos = 130;
int obstacle1_old = 0;
int obstacle2_old = 130-wall_speed;
int score = -1;

//score
int high_score_1;
int high_score_2;
int high_score_3;

//menu promenna
int menu = 0;

//inicializace knihoven
TFT_eSPI tft = TFT_eSPI();
Preferences preferences;

//funkce start
void start(){
  tft.unloadFont(); //vymazani fontu
  tft.fillScreen(CYAN); //vymazani cele obrazovky
  tft.setTextColor(TFT_BLACK, CYAN);
  //vypisovani informaci na obrazovku
  tft.setTextSize(3);
```

```
tft.drawString("Flappy EK", 45, 40);
tft.setTextSize(2);
tft.drawString("set difficulty: ", 10, 120);
tft.drawString("High score: ", 40, 150);
tft.drawString("press OK", 80, 200);

//vyber obtiznosti
while(digitalRead(OK)){

  if(digitalRead(MENU) == LOW){
    delay(200);
    tft.fillRect(175, 150, 100, 40, CYAN);
    menu++;
    if (menu > 2) menu = 0;
    digitalWrite(LED_GREEN, HIGH);
  }

  switch (menu) {
case 0:
  tft.drawString("BEZB", 190, 120);
      speed = 40;
      jump_pixels = 35;
      fall_pixels = 2;
      wall_speed = 1;
      break_height = 95;
  tft.drawString(String(high_score_1), 175, 150);
  break;

case 1:
  tft.drawString("EKP ", 190, 120);
      speed = 26;
      jump_pixels = 37;
      fall_pixels = 2;
      wall_speed = 1;
      break_height = 75;
  tft.drawString(String(high_score_2), 175, 150);
  break;
case 2:
  tft.drawString("LAGA", 190, 120);
```

```
        speed = 20;
        jump_pixels = 50;
        fall_pixels = 4;
        wall_speed = 2;
        break_height = 75;
    tft.drawString(String(high_score_3), 175, 150);
    break;
}
  }

  //signalizace led
  digitalWrite(LED_GREEN, LOW);

  //premazani displeje
  tft.setTextColor(TFT_BLACK, TFT_WHITE);
  tft.fillScreen(TFT_WHITE);
  delay(1000);

}

//reset funkce
void reset(){

  //nebude vypisovat score, kdyz neni
  if(score != -1){
  if(EK_height > 180) EK_height = 180;

  //animace
  tft.pushImage(60, EK_height, 120, 120, bum_1);
  delay(50);
  tft.pushImage(60, EK_height, 120, 120, bum_2);
  delay(50);
  tft.pushImage(60, EK_height, 120, 120, bum_3);
  delay(50);
  tft.pushImage(60, EK_height, 120, 120, bum_4);
  delay(50);
  tft.pushImage(60, EK_height, 120, 120, bum_5);
  delay(50);
  tft.pushImage(60, EK_height, 120, 120, bum_6);
```

```
  delay(50);
  tft.pushImage(60, EK_height, 120, 120, bum_7);
  delay(45);
  tft.pushImage(60, EK_height, 120, 120, bum_8);
  delay(40);
  tft.pushImage(60, EK_height, 120, 120, bum_9);
  delay(30);
  tft.drawString(String(score), 185, 150);
  tft.drawString("Final score: ", 40, 150);
  delay(2000); //cas na precteni
  }



  //premazani displeje
  tft.fillScreen(TFT_WHITE);

  //zapise nejvyssi score do pameti
  switch (menu) {
  case 0:
      if (score > high_score_1){
        high_score_1 = score;
        preferences.putUInt("high_score_1", high_score_1);
      }
    break;
  case 1:
      if (score > high_score_2){
        high_score_2 = score;
        preferences.putUInt("high_score_2", high_score_2);
      }
    break;
  case 2:
      if (score > high_score_3){
        high_score_3 = score;
        preferences.putUInt("high_score_3", high_score_3);
      }
    break;
  }

//resetuje zacinajici promenne pro hru
```

```
EK_height = 90;
EK_height_old = 90;

up_count = 0;
fall_count = 0;

first_obstacle = 0;
obstacle1_break = 0;
obstacle2_break = 0;
obstacle1_pos = 0;
obstacle2_pos = 130;
obstacle1_old = 0;
obstacle2_old = 130-wall_speed;
score = -1;

Serial.println("init");
//premazani displeje
tft.fillScreen(TFT_WHITE);
delay(200);
}

//interupt funkce
void IRAM_ATTR isr() {
  //zjistuje, kdyz se zmackne tlacitko pro skok
  if(up_count < 3){
    up_count = up_count + 3;
  }
  else{
    up_count = up_count + 2;
  }
  fall_count = 0;
  digitalWrite(LED_YELLOW, HIGH);
}

//setup funkce
void setup() {
  preferences.begin("score", false);

  //nacte stara data
```

```
  high_score_1 = preferences.getUInt("high_score_1", 0);
  high_score_2 = preferences.getUInt("high_score_2", 0);
  high_score_3 = preferences.getUInt("high_score_3", 0);

  Serial.begin(9600); //zapne komunikaci

  //nastavi vystupy a vstupy
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_YELLOW, OUTPUT);

  pinMode(OK, INPUT_PULLUP);
  pinMode(DOWN, INPUT_PULLUP);
  pinMode(UP,INPUT_PULLUP);
  attachInterrupt(UP, isr, FALLING);
  pinMode(MENU, INPUT_PULLUP);

  //zapne displej
  tft.begin();
  tft.setRotation(0);
  tft.setTextWrap(true, true);


  //vypise zpravu
  Serial.println("init");

  //spusti start funkci
  start();
}

void loop() {

  //hlida tlacitko pro zmenu avatara
  if(digitalRead(MENU) == LOW){
    avatar++;
    if(avatar > 5) avatar = 0;
  }

  //postupne zrychluje padani
  fall_count++;
```

```
  //resi skakani
  if (up_count > 0){
    up_count = up_count - 1;
    EK_height = EK_height - jump_pixels/3 - fall_pixels;
    if(EK_height < 0-player_pixels/2) EK_height = player_pixels/2;
  }

  //prepne ledku
digitalWrite(LED_GREEN, !digitalRead(LED_GREEN));

  //posouva prekazku 1
  if (obstacle1_pos < 10){
    obstacle1_break = random(0,240-break_height);
    obstacle1_pos = 240;
    score++;
      //premaze kus obrazovky
  tft.fillRect(105, 10, 30, 20, WHITE);
  tft.fillRect(5, 0, 25, 240, WHITE);
  }

  //posouva prekazku 2
  if (obstacle2_pos < 10){
    obstacle2_break = random(0,240-break_height);
    obstacle2_pos = 240;
    if(first_obstacle == 1) score++;
    first_obstacle = 1;
      //premaze kus obrazovky
  tft.fillRect(105, 10, 30, 20, WHITE);
  tft.fillRect(5, 0, 25, 240, WHITE);
  }



  //nakresli avatara podle aktualniho vyberu
  tft.fillRect(60, EK_height_old, player_pixels, player_pixels, WHITE);
  switch (avatar) {
  case 0:
    tft.pushImage(60, EK_height, player_pixels, player_pixels, EK);
```

```
    break;
  case 1:
    tft.pushImage(60, EK_height, player_pixels, player_pixels, avatar_1);
    break;
  case 2:
    tft.pushImage(60, EK_height, player_pixels, player_pixels, avatar_2);
    break;
  case 3:
    tft.pushImage(60, EK_height, player_pixels, player_pixels, avatar_3);
  break;
  case 4:
    tft.pushImage(60, EK_height, player_pixels, player_pixels, avatar_4);
  break;
  case 5:
    tft.pushImage(60, EK_height, player_pixels, player_pixels, avatar_5);
  break;

}


  //vykresli prekazku 1
  tft.fillRect(obstacle1_old+20-wall_speed, 0, wall_speed, obstacle1_break, WHITE);
  tft.fillRect(obstacle1_old+20-wall_speed, obstacle1_break + break_height, wall_speed, 240-
obstacle1_break-break_height, WHITE);
  tft.fillRect(obstacle1_pos, 0, wall_speed, obstacle1_break, GREEN);
  tft.fillRect(obstacle1_pos, obstacle1_break + break_height, wall_speed, 240-obstacle1_break-
break_height, GREEN);


  //zamaze stare misto prekazky
  obstacle1_old = obstacle1_pos;
  obstacle1_pos = obstacle1_pos - wall_speed;


  //vykresli prekazku 2
  if(first_obstacle == 1){
  tft.fillRect(obstacle2_old+20-wall_speed, 0, wall_speed, obstacle2_break, WHITE);
  tft.fillRect(obstacle2_old+20-wall_speed, obstacle2_break + break_height, wall_speed, 240-
obstacle2_break-break_height, WHITE);
  tft.fillRect(obstacle2_pos, 0, 20, obstacle2_break, BLUE);
  tft.fillRect(obstacle2_pos, obstacle2_break + break_height, wall_speed, 240-obstacle2_break-
break_height, BLUE);
  }
```

```
  //zamaze stare misto prekazky
  obstacle2_old = obstacle2_pos;
  obstacle2_pos = obstacle2_pos - wall_speed;

  //detekuje, zda se stala kolize hrace s prekazkou
  if(obstacle1_pos >= 60 - player_pixels/2 && obstacle1_pos <= 60 + 20 + player_pixels/3){
    if(EK_height + hit_box <= obstacle1_break){
        reset();
    }
    if(EK_height + player_pixels - hit_box >= obstacle1_break + break_height){
        reset();
    }
  }

  if(obstacle2_pos >= 60 - player_pixels/2 && obstacle2_pos <= 60 + 20 + player_pixels/3
&&  first_obstacle == 1){
    if(EK_height + hit_box  <= obstacle2_break){
        reset();
    }
    if(EK_height + player_pixels - hit_box >= obstacle2_break + break_height){
        reset();
    }
  }

  //zapamatuje si novou hodnotu lokace hrace, pro nasledne premazani
  EK_height_old = EK_height;
  EK_height = EK_height + fall_pixels + fall_count/7;
  if(EK_height > 240-player_pixels/2) EK_height = 240-player_pixels/2;

  //vypise aktualni score
  tft.setCursor(10, 10);
  tft.setTextColor(TFT_BLACK);
  tft.print("Credits: ");
  tft.print(score);

  //delay pro hezci animace
  delay(20);
  //blikne ledkou
```

```
  digitalWrite(LED_YELLOW, LOW);

}
```