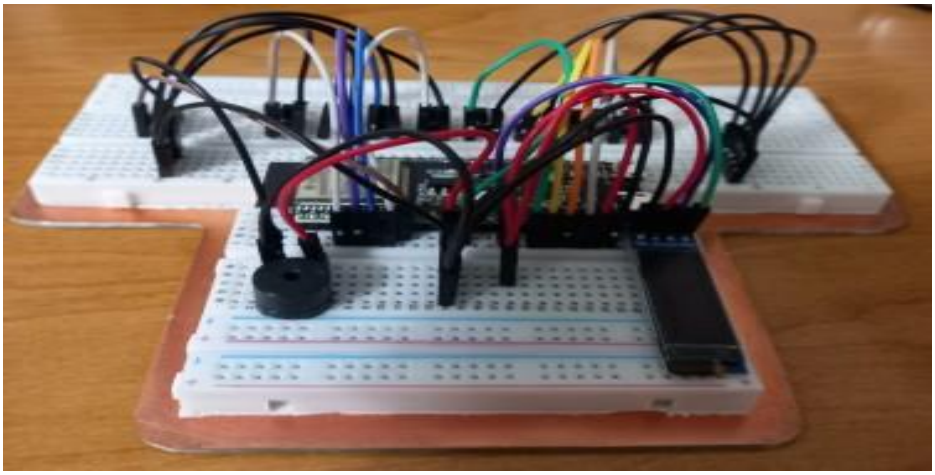
Projekt klavír hrajúci melódiu

Autor: Mgr. Marek Hogh

Ročník: 2 RŠI

Školský rok: 2024/25



Projekt je chápaný ako prototyp jednoduchého piána, ktorý kombinuje hru na klávesy sólo (8 kláves) a automatické prehratie melódie (Jingle bells) prostredníctvom deviatej klávesy. Piano je riadené mikrokontrolórom ESP 32. V systéme je použitý ako doplnok OLED displej pre IOT ARDUINO Raspbery 0.91, s rozlíšením 128 x 32, 3,3 V. Ten slúži na grafické znázornenie stlačených kláves, resp. prehrávanej melódie. Program je vytvorený v prostredí ARDUINO IDE. Ako programovací jazyk je použitý jazyk podobný jazyku C ++. Prostredie ARDUINO má svoje osobitné knižnice a štruktúry, ktoré zjednodušujú prácu s mikrokontrolérmi. Jazyk sa zakladá na C++ ale používa jednoduchšie funkcie a syntax. Program projektu je možné po upravení kódu preprogramovať z fixnej melódie na inú podľa výberu užívateľa. Projekt je možné využiť ako pomôcku, ktorá prispieva ku prepájaniu predmetov hudobná výchova a informatika, čo umožňuje jej využitie v školskom prostredí ako príklad praktického prepájania medzipredmetových vzťahov.

Špecifiká jazyka:

Ako už bolo spomenuté jazyk v prostredí ARDUINI IDE sa zakladá na C++.

Základné štruktúry v Arduino jazyku sú:

1. **setup()** – Funkcia, ktorá sa vykoná len raz pri štarte programu. Slúži na inicializáciu nastavení (napríklad nastavenie pinov, sériovej komunikácie).
2. **loop()** – Funkcia, ktorá sa vykonáva neustále v cykle, pokiaľ je program zapnutý. Slúži na opakované vykonávanie logiky.

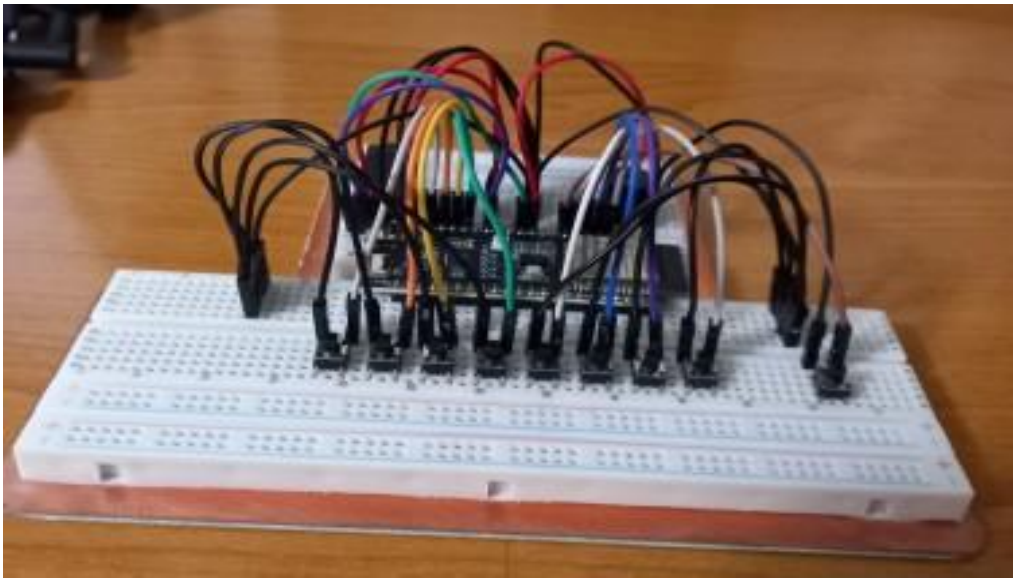
Tento jazyk podporuje aj ďalšie funkcie C++ ako podmienky (if, else), cykly (for, while), premenné, funkcie, objekty a pod.

Použité komponenty:

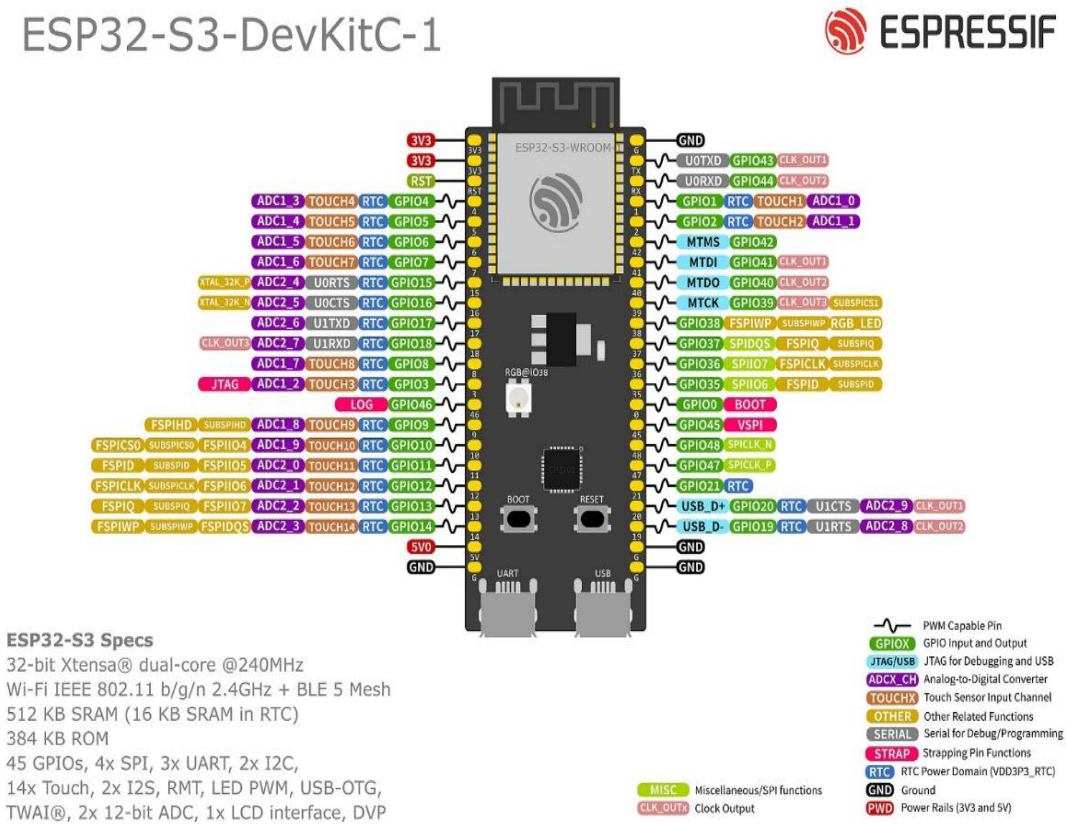
1. ESP 32 S 3DEVKC – 8R8 vývojová sada: Wifi, prototypová doska,
2. Du Pont kábel M – 40 x 10 cm,
3. Nepálivé pole,
4. Mikrospínač TC – 1212T, 12x12x7,3 mm SMD,
5. Piezoelektrický menič 5V,
6. OLED displej s rozlíšením 128 x 32 a ochrane I2C alebo IIC,
7. USB kábel.

Schéma zapojenia (zapájanie celého obvodu):

1. Na pin 5V a PIN G sme zapojili napájanie vývojovej dosky.
2. Na piny 14, 13, 12, 11 a 7, 6, 5, 4 sme zapojili tlačidlá. Obvod pre zapnutie tlačidiel sa uzatvára voči G.
3. Na pin 15 sme pripojili tlačidlo určené pre spustenie melódie. Jeden pól tlačidla je pripojený na G.
4. Na pin 18 (GPIO 18) sme pripojili kladný pól piezoelektrického bzučiaku. Obvod je znovu zakruhovaný voči G (zemi).
5. OLED dispej je pripájaný 4 pinmi. Piny označené VCC a GND sú prepojené s príslušnými pinmi vývojovej dosky (VCC na pin 5V). GND displeja je prepojený s GND pinom vývojovej dosky. Komunikačné piny displeja sú označené SDA (Dataline) a SCL (Clockline).
   Pin SDA (pin displeja) sme pripojili na pin číslo 8 (GPIO 8) vývojovej dosky. SCL sme pripojili na pin 9 (GPIO 9) vývojovej dosky.



Rozloženie pinov na ARDUINO IDE:



Po správnom napojení ďalej nasleduje práca v programovacom prostredí ARDUINO IDE.

Vysvetlenie kódu ARDUINO IDE

Tento kód je určený pre mikrokontrolér ESP32 a riadi interakciu s niekoľkými tlačidlami, ktoré spúšťajú tóny na piezo bzučiaku a zobrazenia na OLED displeji. Tiež zahŕňa prehrávanie melódie „Jingle Bells“ pri stlačení tlačidla. Umožňuje používateľovi interagovať s viacerými tlačidlami, pričom každé tlačidlo prehráva určitý tón a zobrazuje názov tónu na OLED displeji. Okrem toho je tu špeciálne tlačidlo, ktoré spustí prehrávanie melódie „Jingle Bells“. Celý systém je doplnený o debounce čas na tlačidlá, aby sa zabránilo falošným stlačeniam, a všetky akcie sa zobrazujú na displeji.

Kód využíva rôzne knižnice na prácu s tlačidlami, displejom a piezzo buzzerom. Uvádzam analýzu kódu:

Knižnice:

- **#include “ pitches.h“** Uvedený súbor obsahuje popis frekvencii pre rôzne tóny, ktoré budú prehrávané cez piezo buzzer.
- **#incliude <ezButton.h>:** uvedená knižnica umožňuje jednoduché ovládanie tlačidiel s funkciou proti šumu (debouncing).
- **#include <Wire.h>:** predstavuje knižnicu pre komunikáciu po 12C (ide o integrovaný obvod pre sériovú komunikáciu).

- **#include <Adafruit_GFX.h> a #include <Adafruit_SSD1306.h>:** dané knižnice umožňujú prácu s OLED displejom (typicky s rozlíšením 128x32 px).

Definície:

- Rozmery displeja:
  #defice SCREEN_WIDTH 128 a #define SCREEN_HEIGHT 32 charakterizujú veľkosť OLED displeja (128x32).
  #define OLED_RESET 3 popisuje pin na obnovenie displeja.
  #define SCREEN_ADDRESS 0x3C je adresa 12C pre OLED displej.
- **Pin pre piezzo buzzer:**
  #define Buzzer_PIN 18 – definuje pin, na ktorom je pripojený piezo buzzer.
- **Pin pre tlačidlá:**
  #define BUTTON_PIN_1 14, ... , BUTTON_PIN_8 4 sú definované piny pre 8 tlačidiel pripojených ku ESP 32.
- **Frekvencie tlačidiel:**
  Int buttonFrequencies[]: pole obsahujúce frekvencie pre každý tón, ktoré sú zodpovedajúce tlačidlám (napr. tlačidlo 1 hrá tó C4, tlačidlo 2 hrá D4, atď.
- **Popisky tlačidiel:**

  Const char* buttonLabels[]: Polia s názvami tónov, ktoré sa zobrazujú na displeji.

- **Inicializácia:**

1. **Serial Monitor** je nastavený na rýchlosť 9600 baudov.
2. **OLED displej** je inicializovcaný pomocou knižnice Adafruit_SSD1306. Ak inicializácia zlyhá, program sa zastaví a vypíše chybovú správu na Serial Monitor.
3. **Debounce čas** pre každé tlačidlo je nastavený na 100ms, aby sa zabránilo falošným stlačeniam tlačidiel.

Hlavná funkcia (loop):

- **Pre každé tlačidlo:**
  Buttons[i].loop(): kontroluje stav každého tlačidla a zabezpečuje správnu funkciu debouncingu.
  Ak je tlačidlo stlačené (buttons[i].isPressed()):
  Displej sa vymaže, zobrazí sa názov tónu podľa zodpovedajúceho tlačidla a prehrá sa príslušný tón cez **tone(BUZZER_PIN,buttonFrequencies[i].**
  Ak je tlačidlo uvoľnené(buttons[i].isReleased()):
  Prehrávanie tónu sa zastaví pomocou **noTone(BUZZER_PIN).**
  Displej sa vymaže a aktualizuje.

- **Dôležité súvislosti:**
  Debounce tlačidiel: Vďaka knižnici ezButton a nastavenému debounce času (100ms) sa znižuje šanca na náhodné alebo falošné detekcie stlačenia tlačidla.
- **Displej:** Pri stlačení tlačidla na displeji zobrazuje názov tónu, ktorý je aktuálne prehrávaný.
- **Buzzer**: Tón sa prehráva len počas stlačenia a po jeho uvoľnení sa prehrávanie zastaví.

Možné vylepšenia:

1. **Optimalizácia displeja:** momentálne sa pri každom stlačení tlačidla displej vymaže a znova zobrazuje nový text. Uvedený proces by mohol byť optimalizovaný, aby sa vyhli zbytočným zápisom na displej, čo by mohlo znížiť opotrebovanie OLED displeja.
2. **Vylepšenie debouncingu:** Knižnica ezButton už poskytuje debouncing ale pokiaľ je potrebné ešte jemnejšie nastavenie, nožno by bolo vhodné aj experimentovanie s časovými intervalmi.

Uvedený kód je funkčný na prehrávanie tónov cez piezzo buzzer pri stlačení tlačidiel a vizualizáciou zodpovedajúcich tónov na OLED dispeji.

Kód v ARDUINO IDE:

```
#include "pitches.h"
#include <ezButton.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // Šírka OLED displeja v pixeloch
#define SCREEN_HEIGHT 32 // Výška OLED displeja v pixeloch
#define OLED_RESET     3 // Reset pin (alebo -1, ak sa zdieľa reset pin s Arduinom)
#define SCREEN_ADDRESS 0x3C

#define BUZZER_PIN  18 // Pin GPIO18 na ESP32 pripojený k piezo bzučiaku

// Definícia pinov tlačidiel
#define BUTTON_PIN_1 14
#define BUTTON_PIN_2 13
#define BUTTON_PIN_3 12
#define BUTTON_PIN_4 11
#define BUTTON_PIN_5 7
#define BUTTON_PIN_6 6
#define BUTTON_PIN_7 5
#define BUTTON_PIN_8 4
```

```
#define BUTTON_PIN_15 15 // Pridané tlačidlo na pin 15

// Mapa frekvencií tlačidiel
int buttonFrequencies[] = {
  NOTE_C4, NOTE_D4, NOTE_E4, NOTE_F4,
  NOTE_G4, NOTE_A4, NOTE_B4, NOTE_C5
};

// Názvy tlačidiel na zobrazenie na displeji
const char* buttonLabels[] = {
  "C4", "D4", "E4", "F4", "G4", "A4", "B4", "C5"
};

// Melódia Jingle Bells (frekvencie a trvanie)
int melody[] = {
    NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_G4, NOTE_C4, NOTE_D4, NOTE_E4,
  NOTE_F4, NOTE_F4, NOTE_F4, NOTE_F4, NOTE_F4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4,
  NOTE_E4, NOTE_D4, NOTE_D4, NOTE_E4, NOTE_D4, NOTE_E4, NOTE_C4
};

int noteDurations[] = {
  4, 4, 4, 4, 4, 4, 4, 8, 8, 4, 4,
  4, 4, 4, 4, 4, 4, 4, 8, 8, 4, 4, 4,
  4, 8, 8, 4, 8, 8, 4
};

ezButton buttons[] = {
  ezButton(BUTTON_PIN_1),
  ezButton(BUTTON_PIN_2),
```

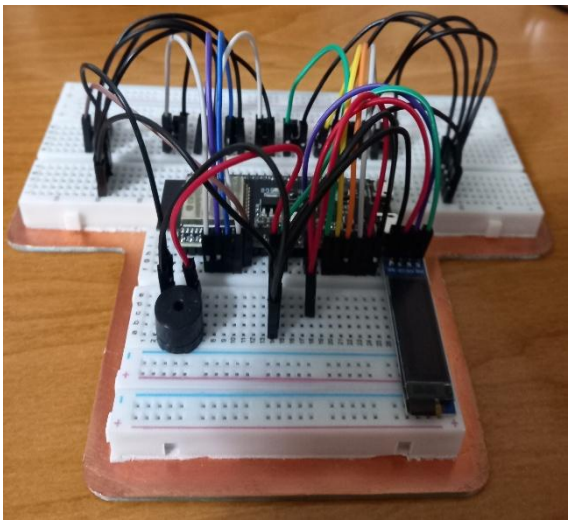
```
  ezButton(BUTTON_PIN_3),
  ezButton(BUTTON_PIN_4),
  ezButton(BUTTON_PIN_5),
  ezButton(BUTTON_PIN_6),
  ezButton(BUTTON_PIN_7),
  ezButton(BUTTON_PIN_8),
  ezButton(BUTTON_PIN_15) // Pridané tlačidlo pre Jingle Bells
};

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

bool isMelodyPlaying = false; // Premenná na kontrolu, či melódia hrá

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

  // Inicializácia displeja
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("Nezískaná alokácia SSD1306"));
    for(;;);                    // Ak displej nefunguje, program sa zastaví
  }

  delay(2000);                  // Pauza na 2 sekundy
  display.clearDisplay();
  display.setTextSize(2);           // Normálny 1:1 pixelový škálovanie
  display.setTextColor(SSD1306_WHITE);// Biela farba textu
  display.setRotation(1); // Otočenie textu na OLED (1=90 stupňov, 2=180 stupňov)
  display.setCursor(7,0);           // Začiatok v ľavom hornom rohu
  display.println(F("MH"));
  display.display();
```

```
  // Nastavenie debounce času pre všetky tlačidlá
  for (int i = 0; i < 9; i++) { // Teraz 9 tlačidiel (8 pôvodných + nové tlačidlo na pin 15)
    buttons[i].setDebounceTime(100);  // 100ms debounce čas pre každé tlačidlo
  }
}

void playJingleBells() {
  int melodySize = sizeof(melody) / sizeof(melody[0]);
  for (int thisNote = 0; thisNote < melodySize; thisNote++) {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(BUZZER_PIN, melody[thisNote], noteDuration);

    // Pauza medzi notami
    int pauseDuration = noteDuration * 1.30;
    delay(pauseDuration);

    // Zastaviť zvuk po každej note
    noTone(BUZZER_PIN);
  }
}

void loop() {
  for (int i = 0; i < 9; i++) { // Teraz 9 tlačidiel
    buttons[i].loop();  // Spracovanie debounce pre všetky tlačidlá

    // Kontrola stlačenia bežných tlačidiel
    if (i < 8 && buttons[i].isPressed()) {
      display.clearDisplay();             // Vymaže predchádzajúci obsah na displeji
      display.setCursor(7, 0);            // Začiatok v ľavom hornom rohu
```

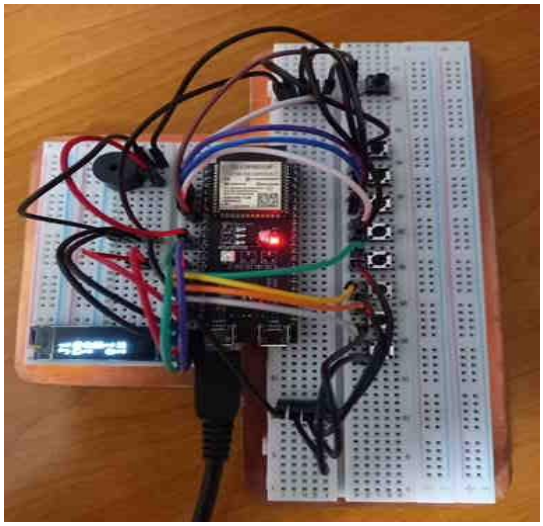
```
      display.print(F(""));                // Text pred názvom tónu
      display.println(buttonLabels[i]);    // Zobrazenie názvu tónu
      display.display();                   // Aktualizácia displeja
      tone(BUZZER_PIN, buttonFrequencies[i]);  // Prehrá zvuk podľa tlačidla
    }

    // Kontrola stlačenia tlačidla pre Jingle Bells (pin 15)
    if (i == 8 && buttons[i].isPressed() && !isMelodyPlaying) {
      display.clearDisplay();
      display.setCursor(0, 0);
      display.println(F("Jingle Bells"));
      display.display();
      isMelodyPlaying = true; // Označíme, že melódia hrá
      playJingleBells();  // Spustí melódiu
      noTone(BUZZER_PIN);  // Zastaví prehrávanie po skončení melódie
      isMelodyPlaying = false; // Resetujeme flag, melódia skončila
    }

    // Uvoľnenie tlačidiel
    if (buttons[i].isReleased()) {
      noTone(BUZZER_PIN);  // Zastaví sa prehrávanie tónu po uvoľnení tlačidla
      delay(100);                      // Pauza
      display.clearDisplay();              // Ak tlačidlo nie je stlačené, vymaže displej
      display.display(); // Aktualizácia displeja
    }
  }
}
```

Po zadaní kódu som prepojil dané zariadenie s Arduiono cloudom. Prekontroloval som funkčnosť celého prepojenia. Na obrázkoch nižšie je zobrazené piano vo vypnutom stave bez znázorneného textu na displeji – obr. č. 1 a piano hrajúce melódiu Jingle bells, ktorú signalizuje grafické zobrazenie názvu melódie na displeji – obr. č. 2.

obr. 1



obr. 2



Video piana hrajúceho melódiu aj samostatné tóny je v prílohe Githubu.

Ďalšie prílohy spolu s arduino súborom sú na linku tu: Na stiahnutie – Všeobecné – UNIoT

Zdroje:

https://randomnerdtutorials.com/esp32-s3-devkitc-pinout-guide/

https://esp32io.com/tutorials/esp32-piezo-buzzer

https://docs.espressif.com/projects/esp-idf/en/v5.2.2/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html

https://www.phippselectronics.com/the-esp32-s3-devkitc-1-pinouts/?srsltid=AfmBOoqTAGM-12ESOWcICEtOq7-uGw5D9Uw2GnkkOf9ZfYX09kqQdMNV

https://esp32io.com/tutorials/esp32-button

https://esp32io.com/tutorials/esp32-button-debounce

https://esp32io.com/tutorials/esp32-oled

https://wokwi.com/projects/291958456169005577