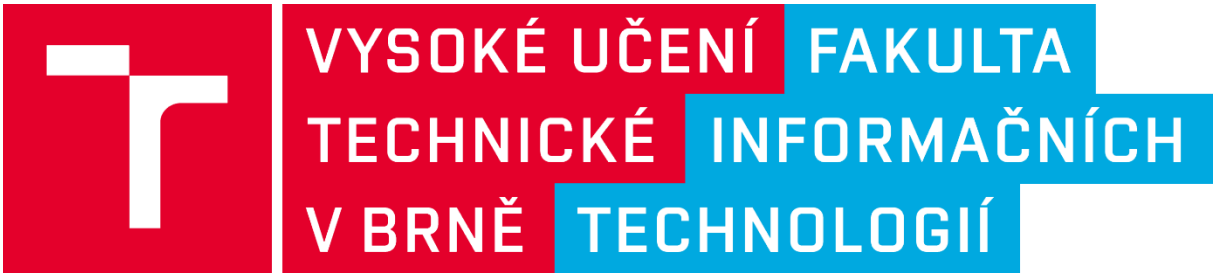
Vysoké učení technické v Brně
Fakulta informačních technologií



Mikroprocesorové a vestavěné systémy
Projekt – Hra na displeji

Meteorits

Dalibor Kalina (xkalin16)
14. prosince 2023

Obsah

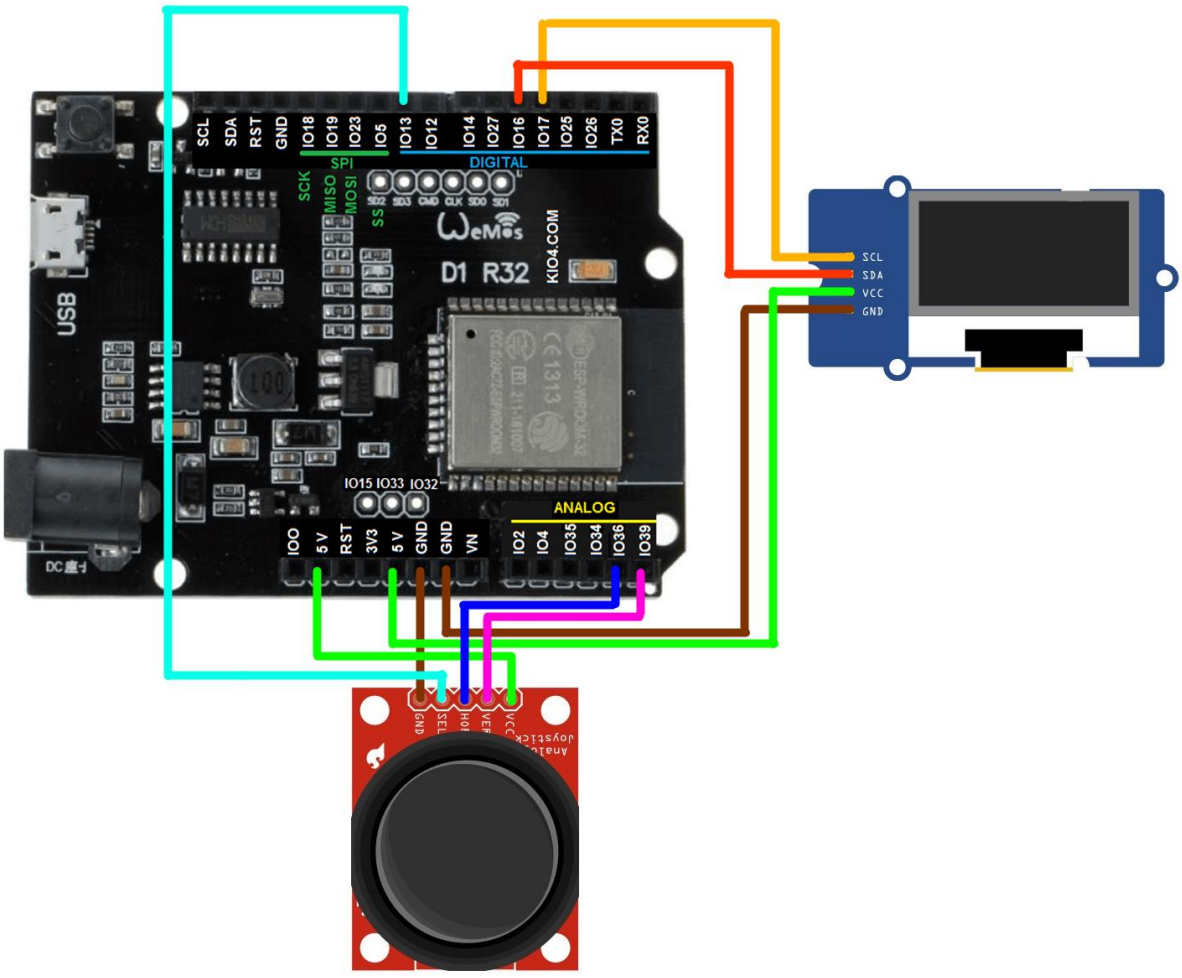
Úvod

Cílem projektu bylo realizovat jednoduchou hru pomocí analogového joysticku a OLED displeje SSD1306 na vývojové desce ESP32. Finální zpracovaná hra byla zjednodušená varianta staré arkádové hry Asteroids[1]. Aplikace byla vyvíjena v jazyce Arduino v Arduino IDE.

Původní plán byl dle rad a návodů na e-learningu[2] využít Visual Studio Code a jeho stažitelné rozšíření PlatformIO. Jelikož se mi však nepodařilo najít vhodný návod na zapojení joysticku, neboť tento postup v e-learningových návrzích chyběl, musel jsem nakonec přejít do Arduina, pro které byly všechny online návody a rady jednodušeji dohledatelné. Samotná instalace a zprovoznění Arduino IDE pro ESP32 následovala webový tutoriál[3].

Zapojení hardware

OLED display byl na desku ESP32 připojen pomocí pinů GIOP16 a GIOP17 dle návodu na e-learningu[2]. Joystick měl své analogové vstupy přiveden na piny GIOP36 a GIOP39 a jeho digitální vstup (resp. tlačítko) na pin GIOP13, zejména s využitím online návodu[4]. Tlačítko bylo původně zapojeno analogově, avšak vytvářelo neuvěřitelně nekonzistentní data na jeho pinu, kdy napětí v čase náhodně skákalo z minima na maximum.



Obr. č. 1: Schéma zapojení

Způsob řešení

Každý Arduino projekt se skládá ze dvou hlavních funkcí – *setup()* a *loop()*. První z nich se provádí jen jednou při spuštění aplikace, druhá opakovaně dokola (obdoba *while(1)*). V mém programu se v *setup()* nastaví správné fungování displeje a tlačítka a proběhne jedno zavolání funkce *reset()*, která je popsána níže. *loop()* má následující strukturu (zjednodušeně):

```
Vykresli hlavní menu
Čekej na volbu uživatele
    if (volba == spustitHru)
        Započti hru
        if (konecHry)
            Ulož skóre
            Vrať se na začátek
    else if (volba == ukazatSkore)
        Ukaž skóre
        Vrať se na začátek
    else if (volba == ukonciAplikaci)
        Ukonči aplikaci
```

Použité knihovny

SPI.h – sériová komunikace pro výpis na terminál Arduino IDE.
Wire.h – funkce pro displej.
Adafruit_GFX.h a *Adafruit_SSD1306.h* – funkce pro joystick.
limits.h – knihovna pro zpřístupnění definovaných proměnných jako např. *INT_MAX*.

Příklad spuštění a průběhu hry

Ihned po spuštění se uživatel nachází na hlavní obrazovce. Jeho aktuální zvolená možnost je vykreslena v invertovaných barvách, aby šlo vidět, co má právě zvoleno. Stisknutí tlačítka vybere zvolenou možnost a přesune se na další obrazovku po přehrání krátké animace.

Při volbě začátku hry se po vykreslení herní arény, a ještě jednomu zmáčknutí tlačítka uživateli zpřístupní pohyb postavy. Pohybem joysticku se postava pohybuje po obrazovce. Rychlost pohybu je řešená analogově, tedy v závislosti na tom, jak moc je joystick nakloněn vlevo-vpravo či nahoru-dolů, určuje hodnotu změny pozice postavy. Ku příkladu: Joystick natočen úplně doleva (= napětí na pinu je 0V) změní *x_pos* (x pozice hráče) o *-4* každý cyklus. Natočení úplně doprava (= napětí na pinu se pohybuje okolo 4100V) provede změnu o *-4*. Všechny napěťové hodnoty mezi těmito natočeními jsou přímo úměrné změně (zaokrouhleno na celé číslo). Pozice postavy nemůže překročit hranice vykreslené arény.

Zmáčknutí tlačítka vystřelí jednu střelu, která se každý cyklus posune (vykreslí) o pevný daný počet pixelů směrem doprava. Střela může buď narazit na konec arény – v tu chvíli zmizí – nebo na meteor.

Každý cyklus se vygeneruje náhodné číslo a s určitou procentuální šancí, která se mění na obtížnosti (ta je spjata se skóre – vyšší skóre = větší obtížnost), vytvoří meteorit o náhodné velikosti (kruh o poloměru 4-6) s náhodnou y pozicí. Meteorit se následně pohybuje stálou rychlostí zprava doleva a mohou nastat tři situace. Buď dorazí na konec obrazovky –

v takovém případě zmizí. Nebo narazí na náboj. Pokud se pozice náboje rovná pozici meteoritu (hitbox meteoritu je čtverec), oba zmizí a skóre je navýšeno o hodnotu v závislosti na velikosti a rychlosti meteoritu (menší rychlejší meteorit je těžší na trefení než větší pomalejší). Poslední možnost, co se s meteorem může stát, je, že se potká s postavou. Uživatel v tu chvíli ztratí jeden ze svých životů. Spustí se animace zásahu, aby si byl uživatel plně vědom toho, že byl zasáhnut. Tato animace trvá necelou sekundu a následně vyčistí vše, co se dělo uvnitř herní arény (tedy všechny meteority a náboje), aby nedošlo k opakovaným zásahům postavy meteority při nadměrném počtu meteoritů.

Obr. č. 2: Zadávání hráčova jména

Uživatel má tři životy znázorněné na pravé straně obrazovky obrázkem plného či prázdného srdíčka. Při obdržení zásahu počet plných srdíček klesne o jedna. Pokud uživatel obdržel celkově tři zásahy, postava je mrtvá a hra končí. Na obrazovku se nejprve vypíše „GAME OVER“ a následně po krátké animaci bude uživatel vyzván k zadání svého herního jména pro uložení získaného skóre. Zadávání jména je intuitivní a nejlépe pochopeno na *obr. č. 2*. Zadávání je dokončeno kliknutím na tlačítko „OK“. V tu chvíli proběhne uložení skóre do tabulky. Tabulka má fixní velikost a v případě, že v ní již není místo, smaže se nejnižší získané skóre. Uživatel je pak vrácen na hlavní obrazovku a může začít další pokus hry.

Pokud si uživatel z hlavního menu vybere možnost zobrazení tabulky skóre, tak se po krátké animaci zobrazí obrazovka, na které jsou na levé půlce zobrazeny zadaná jména a vpravo od nich získané skóre. Uživatel si nimi může procházet pomocí joysticku a po stisku tlačítka se vrátit na hlavní obrazovku. Pokud žádné skóre ještě nebylo získáno (tedy hráč nehrál hru ani jednou), žádné skóre se v tabulce neukážou a hráč je informován o tom, že si musí nejprve hru zahrát.

Poslední možností na hlavním menu je tlačítko pro ukončení hry. To vyčistí obrazovku a program skočí do nekonečné smyčky. Pro opětovné spuštění musí uživatel restartovat desku ESP32 nebo znovu nahrát aplikaci na desku.

Způsob realizace pohybu

Tomuto bodu bych chtěl věnovat celou jednu kapitolu dokumentace. Jednalo se o první velký problém (hned po zprovoznění desky a Arduino IDE), nad kterým jsem se musel pozastavit. Celý popis problému je následovný: „Hráč se pohybuje v závislosti na uživatelském vstupu. Střely hráče se pohybují konstantní rychlostí doprava. Meteority se pohybují konstantí rychlostí doleva. Jak správně vykreslit pohyb střel a meteoritů?“

Střela vzniká z hráčovy pozice při stisku tlačítka. Jelikož se má pohybovat směrem doprava, tak v prvotních verzích jsem konstantně posouval celou obrazovku směrem doprava, což vytvořilo iluzi pohybu střely. To však mělo za následek to, že jsem nemohl využít *display.clearDisplay()*, neboli knihovní funkci pro smazání všeho na displeji a hráčův pohyb tvořil prapodivné vzorce na obrazovce, neboť se při každém jeho pohybu jeho předchozí pozice nesmazala. I kdybych vyřešil tento problém (a jakože jsem ho i řešit začal pomocí překreslení předchozí pozice invertovanou barvou = z bílé na černou), nastal by druhý a větší

– pohyb meteoritu. Ty se musí pohybovat přesně opačným směrem, avšak obrazovka se nyní pohybuje doprava.

Finálním řešením bylo vytvořit si pole o pevné velikosti (maximální počet střel a meteoritů je tedy omezen na *MAX_BULLETS* a *MAX_METEOR*), do kterého si budu ukládat existenci a stav všech střel a meteoritů. Těchto polí je vícero a při zpětném zamýšlení jsem je rozhodně měl dát do jedné struktury. Jak střely, tak meteority mají tři základní pole: *exist, posX, posY*. První pole bool hodnot udává, zda se objekt nachází na displeji. Při tvorbě nového objektu se nejprve najde první *false* hodnota v tomto poli, přepíše se na *true* a její index se zapamatuje pro práci s dalšími poli. Na stejný index se do *posX* a *posY* zapíše pozice. Ta je v případě střely dána pozicí hráče při stisknutí tlačítka. V případě meteoritu je x pozice rovna nejpravější straně obrazovky (viz *GAME_WIN_MAX_X*) a y pozice je náhodně vygenerovaná v rozmezí dané herní arénou (viz *GAME_WIN_MIN_Y* a *GAME_WIN_MAX_Y*). Poslední dvě pole, které už má pouze jen meteorit, si pamatují a udávají jeho rychlost (*speed*) a velikost (*size*).

S těmito všemi údaji se obrazovko nyní maže a znovu vykresluje každý cyklus s tím, že se pozice střel a meteoritu s každým cyklem mění v *posX* (y pozice zůstává stejná napříč celou existencí objektu).

Popis funkcí

Většina programu probíhá přímo ve funkci *loop()*, ale přesto jsou nějaké části odděleny pro lepši přehlednost do vlastních funkcí.

mainMenuDisplay(int input)

Na obrazovku vykreslí 3 možnosti (*PLAY, SCORES, EXIT)*. Právě zvolená možnost *input* je vykreslena v invertovaných barvách, tedy černé písmo na bílém pozadí.

reset()

Nastaví všechny globální proměnné do stavu připraveném na spuštění hry. Tohle zahrnuje mimo jiné například pozici hráče, počet životů, skóre či pole pro meteority a střely.

checkButtonPress(int error_value)

Přečte a porovná hodnotu na pinu GIOP13 (tedy tlačítko joysticku) vůči hodnotě LOW, neboli stav „zmáčknuto“. Hodnota musí být na pinu přivedena alespoň tolik cyklů, kolik je hodnota *error_value.*

gameMainWindow()

Vykreslí hlavní hrací okno kromě hráčovy postavy, všech meteoritů a střel.

clearGameWindowOnHit()

Při zásahu a po krátké animaci znázorňující zásah vymaže všechno, co se děje na obrazovce, aby měl uživatel čas se vzpamatovat a nebyl zahlcen zásahy v případě nadměrného množství na něj letících meteoritů.

scoreWindowLettersDisplay(int opt, int A, int B, int C)

Vykreslí 4 „tlačítka“. Tři z nich obsahují písmeno abecedy v závislosti na hodnotě *A, B* a *C* (tyto proměnné nabývají hodnot 0-26). Čtvrté tlačítko slouží jako potvrzovací tlačítko. Právě zvolené tlačítko (dle *opt*) je vykresleno v invertovaných barvách. Tato funkce slouží k uživatelskému zadání jména pro uložení skóre.

saveNameToScores(int A, int B, int C)

Po zvolení čtvrtého tlačítka (resp. kliknutí na něj) uloží jméno a získané skóre do tabulky všech předchozích skóre.

Video ukázka

Krátké mnou natočené video průběhu jedné hry a ukázky hlavního menu je dostupné na mém google drive pod tímto odkazem. Odkaz na video je zároveň přiložen v textovém dokumentu, který je součástí odevzdání.

Závěr

Aplikace by se určitě dala vylepšit, co se efektivity a přehlednosti kódu týče. Je použito nadměrné množství globálních proměnných a malé množství struktur. Některé části kódu se mohli též volat pomocí funkce, aby se zvýšila úhlednost *loop()*. Krom těchto vad je však finální projekt kompletní a hra plně funkční.

Zdroje

[1] *Asteroids (video game)* [online], poslední aktualizace 25. listopadu 2023 19:29 [cit. 14. 12. 2023], Wikipedie. Dostupné z WWW: https://en.wikipedia.org/wiki/Asteroids_%28video_game%29

[2] *Projekt – ESP32 (M)* [online], poslední aktualizace 10. listopadu 2023 13:40 [cit. 14. 12. 2023], Moodle Informačního systému VUT, Dostupné z WWW: https://moodle.vut.cz/mod/page/view.php?id=338880

[3] *Vývojová deska ESP32* [online], Luboš M. [cit. 14. 12. 2023], Drátek návody, Dostupné z WWW: https://navody.dratek.cz/navody-k-produktum/vyvojova-deska-esp32.html

[4] *ESP32 – Joystick* [online], [cit. 14. 12. 2023], ESP32 I/O, Dostupné z WWW: https://esp32io.com/tutorials/esp32-joystick