Multiplayer game (intre 2 placute ESP32) (Battleship)
Introducere

* **Ce face?**

Proiectul consta in realizarea unui joc multiplayer intre doua placute ESP32. Jocul ales este Battleship iar comunicarea intre placi se face prin EspNOW sau Bluetooth.

* **Care este scopul lui?**

Scopul acestui multiplayer este in primul rand entertainment-ul utilizatorului. Acesta poate sa joace multiplayer cu un alt utilizator ce detine o alta placuta ESP32.

* **Care a fost ideea de la care aţi pornit?**

Ideea a fost cea de comunicare intre doua placute. Am vrut sa explorez posibilitatile de transmitere de mesaje de la placuta la placuta.

* **De ce credeţi că este util pentru alţii şi pentru voi?**

Proiectul este util pentru ca aduce un joc clasic pe placutele ESP32. E o ocazie buna sa arat cum functioneaza comunicarea intre ESP32-uri.

Descriere generală

* **TTGO T-Display ESP32**: cele doua placute ce interactioneaza prin ESPNow

* **Display LCD**: prin protocolul SPI ESP-ul comunica cu Displayul. Acesta va arata starea jocului

* **5 mm Red LED with Diffused Len**: Ledul care va arata ca nava nu are o pozitie valida.

* **5 mm Green LED with Diffused Len**: Ledul care va arata cand este randul playerului curent.

* **3 butoane** pt fiecare placuta (6 total) pentru controlul utilizatorului cand acesta plaseaza nava (2 butoane (x,y) unul pentru rotatie)

Hardware Design

Componentă	Descriere
TTGO T-Display ESP32	Două plăcuțe ESP32 ce comunică între ele folosind protocolul **ESP-NOW**
Display LCD (SPI)	Afișează starea jocului; comunică cu ESP-ul prin protocolul **SPI**
LED roșu 5mm difuz	Indică faptul că nava **nu are o poziție validă**
LED verde 5mm difuz	Indică faptul că este **rândul jucătorului curent**
3 butoane / placă (6 total)	Control utilizator pentru **plasarea navei**: două butoane pentru coordonate *(x, y)* și unul pentru **rotație**

Aici puneţi tot ce ţine de hardware design:

- listă de piese
- scheme electrice (se pot lua şi de pe Internet şi din datasheet-uri, e.g. http://www.captain.at/electronic-atmega16-mmc-schematic.png)
- diagrame de semnal
- rezultatele simulării

Software Design

Descrierea codului aplicaţiei (firmware):

- Mediu de dezvoltare: *vim* (i'm serious).
- Am folosit:
- WiFi.h - pentru protocolul esp_now
- esp_now.h - pentru protocolul esp_now
- TFT_eSPI.h - pentru driverele de display
- Am implementat un protocol de comunicatie care arata asa:

#define MOVE_SHIP 'm'

#define ADD_SHIP 'a'

#define PASTE_SHIP 'p'

#define ADD_BOMB 'b'

#define MOVE_BOMB 'B'

#define PASTE_BOMB 'P'

protocolul de comunicatii arata asa: “header|int|int|int”. Header-ul poate fi unul din cele 6, sau 's' sau 'w'. s inseamna ca se schimba modul - din ships in bombs. w inseamna ca s a terminat jocul.

- Functii implementate:

```
  void IRAM_ATTR onMoveYPlayer1() {
if (!validate()) {
        return;
}
if(millis() - last_movey_time > debounce_delay) {
        last_movey_time = millis();
} else { return; }
if (mode == PUTTING_SHIPS) {
        const char *msg = "m|0|1|0";
        esp_err_t result =
                        esp_now_send(peer_mac, (uint8_t *)msg, strlen(msg));
        do_received_command(msg);
} else {
        const char *msg = "B|0|1|0";
        esp_err_t result =
                        esp_now_send(peer_mac, (uint8_t *)msg, strlen(msg));
        do_received_command(msg);
```

```
}
  }
```

Asa arata o functie de intrerupere^, de exemplu cea ce misca nava / bomba pe pozitia Y. Si sursa si destinatarul vor executa comanda.

```
 void OnDataRecv(const uint8_t *mac, const uint8_t *incomingData, int len)
      {
Serial.print("Received from: ");
for (int i = 0; i < 6; i++) {
        Serial.print(mac[i], HEX);
        if (i < 5)
                Serial.print(":");
}
Serial.print(" | Data: ");
Serial.write(incomingData, len);
Serial.print("Finished_data");
Serial.print(len);
Serial.print("Finished len");
char* incoming_data_str = (char*)calloc(len + 1, sizeof(char));
strncpy(incoming_data_str, (char*)incomingData, len);
int r = do_received_command(incoming_data_str);
Serial.println(incoming_data_str);
free(incoming_data_str);
 }
```

Asa arata functia (intreruperea) care handle-uieste primirea comenzii.

Am doua arrayuri de date globale: ships[SHIPS_NR] si bombs[BOMB_NR]. Am un pending_ship si un pending_bomb, precum si doua variabile: mode - SHIPS/BOMBS si modulus, care este ID-ul fiecarui controller

* Flow:

Este mereu randul playerului cu nava albastra. Primul buton din dreapta muta pe x, al doilea muta pe y, al treilea schimba orientarea navei si da toggle intre x negativ si pozitiv Ultimul buton pune nava/bomba definitiv pe tabla.

Github

* https://github.com/Barosandu/pmpriject

Video

* https://youtube.com/shorts/LX7CNFMTghA?si=4i6bsephvSB5epEF

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.
Concluzii

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2025/cmoarcas/alexandru.ariton04

Last update: **2025/05/28 17:05**