UNIVERSITATEA POLITEHNICA TIMIŞOARA

Facultatea de Automatică şi Calculatoare

Sisteme încorporate

Proiect:
Da-i bice roulette

Costolas Darius-Flavius
An III CTI
Grupa 2.1

An universitar 2018-2019

Cuprins:

Tema proiectului:

Lucrarea curenta isi propune realizarea unei rulete rudimentare care va expune un access point si o interfata http cu care un client, in cazul nostru mobil, se va putea sincroniza cu aceasta

Acest sistem va dispune de niste display-uri care vor informatii ca: ultimele 8 numere “trase” la ruleta, al catelea spin este, daca ruleta este in starea de acceptare de beturi sau de spinning si cat timp a mai ramas pana la schimbarea starii curente.

Pentru realizarea acestui proiect au fost necesare cunostinte despre:

- ➔ microcontrollere (în cazul din proiect fiind vorba despre un microcontroler ESP32)
- ➔ servere web si protocoale de comunicare
- ➔ cunostiinte despre React Native

1.1 Descrierea Hardware

Partea hardware a proiectul consta in primul rand din microcontroller-ul ESP32. Acesta este un chip low cost si low power care vine cu un chip de wifi incorporat in sistemul acestuia.

In cadrul proiectului curent am folosit un microcontroller care vine cu programator lipit care are un port micro usb, si picioare lipite pe pini, gata de pus in breadboard.

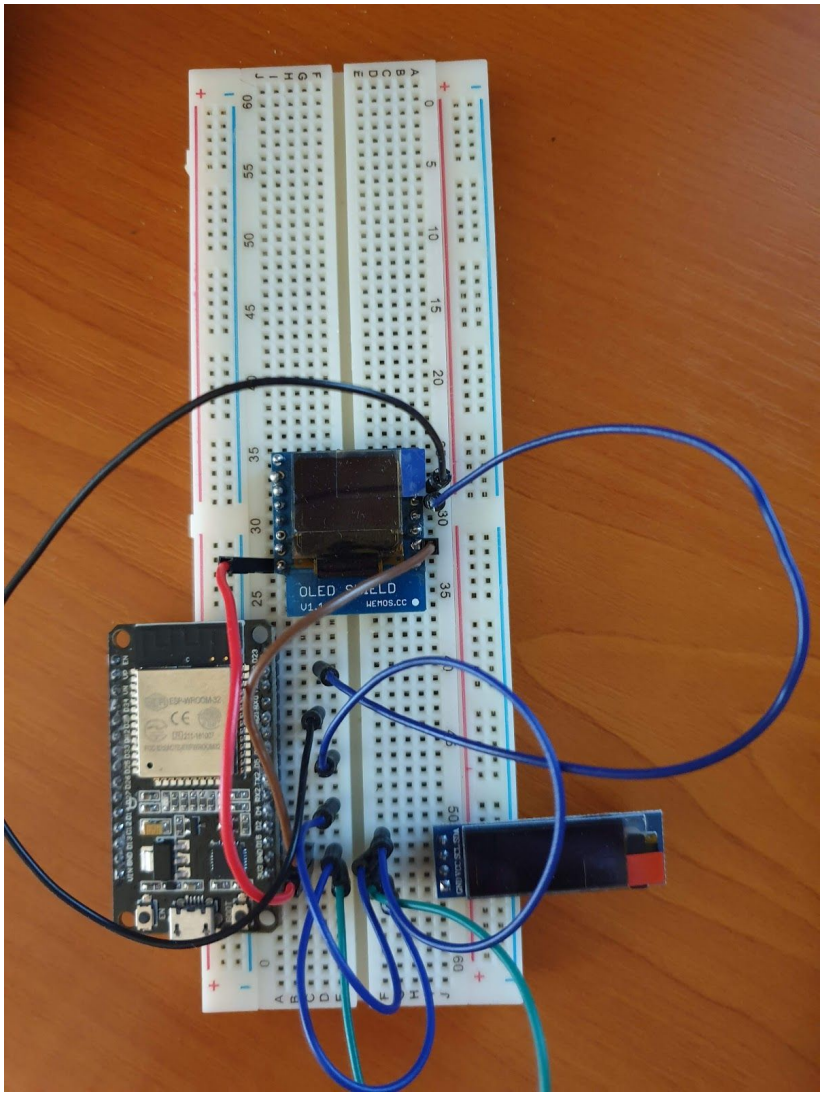
2 Display-uri au fost folosite pentru a expune informatiile de pe ruleta. Amandoua aceste display-uri sunt de tip SSD1306 OLED si se comunica cu microcontroller-ul folosind protocolul I2C.

Primul display are dimensiunile 64x48 si pe acesta avem afisate numarul de spin, starea curenta a ruletei (WaitingForSpin/Spinning) si timpul cat a ramas pana se schimba starea. Acesta display provine dintr-un Arduino OLED Shield V1.1.0 si astfel ca are pinii specifici de conectarea pe un Arduino, de aici vom folosi doar pinii de 3V3, GND si D1 care este de fapt

SCL si D2 care este SDA. Acesta are 2 adrese de I2C respectiv 0x3C si 0x3D. Noi vom folosi adresa 0x3C.

Al doilea display are dimensiunile de 128x32 si acesta va avea afisat pe 2 linii, ultimele 8 numere “trase” de ruleta. Acesta are direct 4 pini: GND, VCC(3.3v), SCL si SDA. Problema la acest display este ca amandoua folosesc aceasi adresa si astfel nu le-am putut pune pe aceleasi linii asa ca fiecare e display e pe alti pini. O solutie mai buna era implementarea multiplexor I2C ca de ex. TCA9548A.

Montajul schemei este



I2C (Inter-Integrated Circuit) este un protocol serial sincron de comunicare cu care poti conecta mai multi slaves la un singur master. Acesta foloseste 2 linii ca se transmita/primeasca date.

- SDA, pe care se transmit/primesc date

- SCL, linia de clock

Fiecare device e adresabil dupa o adresa unica si astfel mai multe dispozitive pot fi puse pe aceasi linie. Pentru a incepe comunicarea cu un slave respectiv pentru a incheia-o se folosesc niste conditii de START/STOP care sunt SDA-H,SCL-H respectiv SDA-L,SCL-H. Fiecare pachet de date contine 8 biti, dupa fiecare urmand un bit ACK care anunta ca pachetul a fost primit. Primul pachet trimis spre un slave contine adresa lui(7/10 biti) + 1 bit care specifica daca se pune pe modul R/W.

SSD1306 este un chip care este driver pentru un OLED de diferite dimensiuni. Ca acesta sa functioneze, acesta are un Graphic Display Data RAM (GDDRAM) care contine informatiile ce trebuie afisate. Acesta este format dintr-un anumit numar de pagini/coloane, in functie de dimensiunea fiecarui display. Acesta este alimnentat de o sursa de 3.3V si are un consum maxim de 100uA / segment. Acesta dispune de comenzi de control cu care se aprinde fiecare segment in parte.

1.2 Descrierea Software

Partea software consta in 2 parti

- terminale mobile (client)
- montajul de de ruleta (ESP32) (descris la pagina 2)

1.2.a) Aplicaţie nativă React Native

Partea de primire si procesare a inforamtiei se realizeaza astfel:

React Native este un framework open source dezvoltat de facebook, in care poti scrie un singur cod si apoi distribui pe mai multe sisteme de operare in sistem nativ.

Aplicatia are un ecran principal. In cadrul acestui ecran, se poate observa ruleta impreuna cu toate tipurile de pariere, impreuna cu numarul de credite curente, butoane de reset/undo nmarul spinului curent si un bara de progres pentru a vedea cat timp mai e pana se incepe un nou spin

La intrarea in aplicatie, primul lucru ce se intampla este ca aplicatia incearca sa faca request-uri de HTTP de tip GET la un ip prestabilit (192.168.4.1) pe portul 80 si pe ruta status. Acest ip este ip-ul ruletei pe care aceasta si-l atribuie ca si client in reteaua de AP pe care o creaza.

Pe acest request vor veni informatii despre starea ruletei: numarul curent de spin, care este statusul (WaitingForSpin/Spinning), cat timp mai e pana se trece la urmatoarea stare, cat timp a fost initial, si ultimul numar castigator.

Pe baza acestor informatii, aplicatia de pe telefon ia decizii si lasa jucatorul sa faca anumite actiuni.

Cat timp ruleta este in starea de WaitingForSpin, jucatorul poate plasa pariuri pe telefon in limita creditului pe care il are. In momentul in care starea se schimba in Spinning, jucatorul nu mai poate plasa pariuri si asteapta sa vina un numar castigator. In momentul in care starea ruletei se schimba inapoi in WaitingForSpinning, aplicatia de pe telefon calculeaza castigul in functie de numarul castigator primit de la ruleta.

(Figura 1)

1.2.b) Montajul ESP32

Codul ruletei a fost scris folosind arduino compiling/linking/deploying cu un driver si librarii speciale pentru ESP32.

Cateva constante definesc care sunt parametrii de pornire si apoi se initializeaza modulele cheie.

Modului de wifi se configureaza ca fiind un Access Point pentru clientii cu aplicatiile mobile. Mai apoi se porneste un WebServer pe portul 80 si se adauga un handler pentru ruta GET /status, pentru a putea returna statusul ruletei mentionat mai sus.

Modulele de display sunt initializate folosind libraria SSD1306Wire, aceasta librarie foloseste niste variabile globale asa ca de fiecare data cand vreau sa scriu pe un display, acesta trebuie reinitializat pentru ca altfel s-ar scrie pe ultimul display initializat.

Codul consta intr-un automat cu stari finite care la fiecare secunda scade din timpul curent ramas iar daca acesta ajunge la 0, schimba starea ruletei. Si executa logica de alegere a numarului si apoi afisarea ultimelor numere primite.

Alegerea numarului se face folosind random() care are nevoie de un seed initial pentru algoritmul acesteia care este initializat in setup cu valoarea de pe pinul analog 0 care este destul de impredictibila.

In momentul in care un client face request pe ip-ul, portul si ruta specificata, functia de handle al request-ului creaza un string in format JSON unde adauga informatiile despre starea ruletei si numarul castigator pentru a putea fi citita mai usor de aplictia mobila.

2.2) Exemple de cod

```
boolean checkSecondPassed()
{
    elapsedTime = millis();
    if (elapsedTime - startTime >= 1000)
    {
        startTime = elapsedTime;
        return true;
    }
    return false;
}
```

(Figura 2)

In figura 2, putem observa cum se verifica daca a trecut o secunda. functia millis() returneaza numarul de milisecunde de cand a fost pornit microcontroller-ul. Astfel daca pastram la inceput un startTime=millis() si la

fiecare loop verificam daca timpul curent - timpul de inceput > 1000 ms(1s), putem vedea daca intradevar a trecut o secunda sau nu.

```
void drawLastNumbers()
{
    beginLastNumbersDisplay();
    lastNumbersDisplay.clear();
    std::stringstream firstRow;
    for (unsigned short i = 0; i < min((unsigned short)4, lastNumberIndex); i++)
    {
        firstRow << lastNumbers[i];
        if (i < 3)
            firstRow << " ";
    }
    lastNumbersDisplay.drawString(0, 8, firstRow.str().c_str());
    if (lastNumberIndex >= 4)
    {
        std::stringstream secondRow;
        for (unsigned short i = 4; i < min((unsigned short)8, lastNumberIndex); i++)
        {
            secondRow << lastNumbers[i];
            if (i < 7)
                secondRow << " ";
        }
        lastNumbersDisplay.drawString(0, 40, secondRow.str().c_str());
    }
    lastNumbersDisplay.display();
    beginMainDisplay();
}
```

(Figura 3)

In figura de mai sus (figura 3) putem observa cum se face desenarea ultimelor 8 numere alese. Inainte ca sa putem sa desenam, trebuie sa reapelam functiile de intializare a display-ului care se ocupa de afisarea ultimelor numere

Fiecare numar ales este adaugat intr-un array lastNumbers si pastrat un index la cat s-a ajuns in caz ca inca suntem la un spin number la care nu avem istoricul al 8 numere.

Prima data, desenam primul rand de 4 numere, inainte de care construim un sir de caractere ca sa contina toate numerele si apoi le desenam pe display-ul corect, folosind functia drawString la coordonatele x:0, y:8

Pentru al doilea rand de 4 numere vom construi din nou un sir de caractere care sa contina numere si vom desena din nou, de data aceasta la coordonatele x:0, y:40

```
server.on("/status", handleGetStatus);

void handleGetStatus()
{
    char response[200];
    sprintf(response,
            "{ \"status\": \"%s\", \"spinNumber\": %d, \"timeLeft\": %d, \"totalTime\": %d, \"lastWinningNumber\": %d }",
            gameState == WaitingToSpin ? "WaitingForSpin" : "Spinning",
            spinNumber,
            gameState == WaitingToSpin ? waitTimeToSpin : timeToSpin,
            gameState == WaitingToSpin ? WaitToSpinTime : SpinTime,
            winningNumber);
    server.send(200, "application/json", response);
}
```

(Figura 4)

In figura 4 putem observa ca se adauga un event listener pe ruta /status al serverului si cu functia handleGetStatus ca handler al event-ului.

Primul lucru care se face, este sa se construiasca raspunsul in format JSON pentru a putea fi citit de client, mai apoi se apeleaza functia send pentru a raspunde inapoi clientului care a facut request-ul. Se trimit ca parametrii functiei send, 200 ca fiind codul de status http, tipul de encoding al raspunsului si apoi raspunsul in sine. Header-ele aferente protocolului http sunt adaugate de librarie WebServer.

2.3) Concluzii

In concluzie se poate observa ca un sistem in care ruleta sa fie conectata la internet si jucatorii sa foloseasca aplicatia pentru a se juca ar fii ceva trivial, mai ales daca se foloseste microcontroller-ul ESP32 care are destula memorie, spatiu si pe deasupra chip-uri pentru diferite tipuri de comunicari (WIFI, Bluetooth).

Lucruri ce ar mai putea fi de imbunatatit ar fii un I2C switch pentru a folosi aceleasi linii pentru amandoua display-uri, implementarea unor leduri sau ceva vizual pentru a vedea mai bine tot ansamblul de ruleta de joc si mai apoi functionalitati de securitate ca de ex, validarea de castig pe montaj si nu pe client sau folosirea protocolul HTTPS si nu HTTP pentru a opri atacuri de tipul “man in the middle”.

Bibliografie:

[1]https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[2]https://i2c.info/i2c-bus-specification

[3]https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf

[4]https://www.instructables.com/id/ESP32-and-OLED-Display-Internet-Clock-DHT22/?fbclid=IwAR3T1julWr7EvymKfEDhs4CvSJhKeXWI4ruiBeHW29krSaGxFnxcb1a9Qac

[5]https://facebook.github.io/react-native/