Retro arcade games console - Rusu Bogdan, 332CA

Introducere

Proiectul meu consta in realizarea unei console de jocuri arcade retro (Tetris, Pacman si altele) din dorinta de a face un proiect pe care sa il pot folosi si dupa PM.

Consola va fi encapsulata intr-o carcasa printata 3d si va avea un ecran color pe care se desfasoara jocurile, incluzand un meniu de selectie. Pe langa acestea, vor fi amplasate butoane pentru controlul in cadrul jocurilor, pentru controlul sunetului sau a statusul consolei (on, sound, sleep).

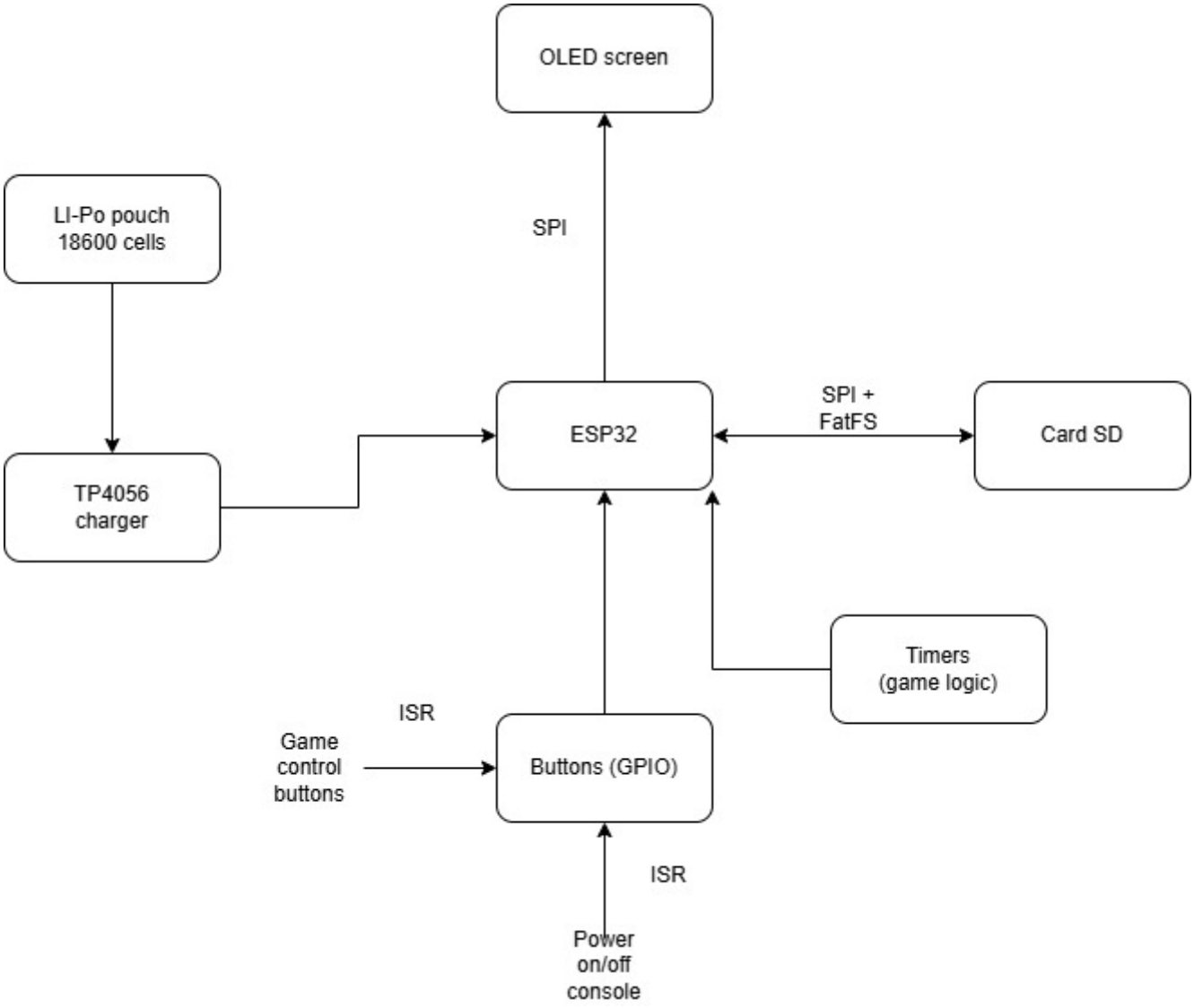
Ceea ce doresc sa realizez este si functionalitatea de power management a consolei pentru conservarea energiei cand nu se joaca nimeni pe ea.

Ideea a pornit mai mult din dorinta de a face ceva fun si interactiv si sa invat mai mult despre cum se poate construi un proiect practic pana la stadiul de a fi chiar utilizabil.

Pentru salvarea high scores sau a nivelurilor la care jucatorul a ajuns voi folosi un card micro sd conectat la placuta care sa memoreze aceste date.

Descriere generală

Placuta de ESP32 va actiona drept 'creierul' consolei de joc. De la acesta se vor trimite comenzile de desenare prin SPI catre ecran, se vor salva si respectiv retrage datele despre highscores si nivelurile la care a ajuns jucatorul si va interpreta comenzile de la butoane prin intreruperi. De asemenea prin timerele interne ale placutei se va monitoriza perioada de inactivitate pentru a trece consola intr-un sistem de low power pentru conservarea bateriei.



Hardware Design

Placa de dezvoltare ESP32

Este “creierul” consolei: un microcontroller dual-core Tensilica LX6 la 240 MHz, cu 520 KB SRAM și Wi-Fi/Bluetooth integrate. Primește alimentare prin pinul VIN (5 V) sau prin USB, iar regulatorul onboard îi trimite stabil 3,3 V la nucleu. Toate perifericele (SPI, ADC, GPIO, PWM, Deep-Sleep) sunt gestionate de ESP32.

TFT 2.2″ ILI9341 cu microSD

Un display color 320×240 px care comunică prin SPI. Pe aceeași magistrală împarte datele între ecran și cardul microSD, având două CS-uri separate. Backlight-ul are un pin marcat “BL” cu “PWM SAFE” – silkscreen-ul indică faptul că placa conține intern tranzistor și rezistență de limitare, astfel încât poți aplica direct un semnal PWM din ESP32 pentru reglaj de luminozitate.

Modul Pin	ESP32 Pin	Protocol	Funcție
VCC	3V3	Power	Alimentare 3.3 V
GND	GND	Power	Masă comună
SCK	GPIO18	SPI_CLK	Ceas SPI
MISO	GPIO19	SPI_MISO	Date SPI din SD
MOSI	GPIO23	SPI_MOSI	Date SPI către TFT/SD
CS	GPIO5	SPI_CS	Chip-select pentru TFT
DC	GPIO2	GPIO/SPI	Data/Command TFT
RST	GPIO16	GPIO	Reset hardware TFT
SDCS	GPIO4	SPI_CS	Chip-select pentru microSD
BL	GPIO15	PWM	Reglaj backlight (PWM SAFE)

Card microSD

Se introduce direct în slotul modulului TFT. Servește la stocarea high-score-urilor, configurațiilor și eventual a altor asset-uri (hărţi de nivel, fonturi). Accesul se face cu biblioteca FatFS prin API-ul SD.begin(SD_CS) și operaţiuni standard SD.open()

Elemente de control (9 butoane INPUT_PULLUP + întreruperi)

Butoane (9×, INPUT_PULLUP + întreruperi)

- 4× direcţionale (D-pad) pentru navigare în joc.
- 2× A/B pentru acţiuni principale şi secundare.
- Start/Select pentru control de meniu și pauză.
- Power pentru deep-sleep/wake.
- Fiecare e configurat ca pinMode(..., INPUT_PULLUP) și declanșează ISR pe front FALLING, asigurând reacţie instantă fără polling.

Funcție	GPIO Pin	Config	Interrupție
D-pad Sus	GPIO32	INPUT_PULLUP	FALLING (apăsare)
D-pad Jos	GPIO33	INPUT_PULLUP	FALLING (apăsare)
D-pad Stânga	GPIO25	INPUT_PULLUP	FALLING (apăsare)
D-pad Dreapta	GPIO26	INPUT_PULLUP	FALLING (apăsare)
Buton A	GPIO27	INPUT_PULLUP	FALLING (apăsare)

Buton B	GPIO14	INPUT_PULLUP	FALLING (apăsare)
Sound	GPIO21	INPUT_PULLUP	FALLING (apăsare)
Menu	GPIO22	INPUT_PULLUP	FALLING (apăsare)
Power	GPIO12	INPUT_PULLUP + WAKE	FALLING (deep-sleep wake)

Monitorizare nivel baterie

Două rezistoare (100kΩ/100 kΩ) formează un divizor care aduce maxim 4,2 V de la BAT+ la ~2,1 V, sigur pentru ADC-ul ESP32. Se conecteaza punctul de măsură la GPIO35 (ADC1_CH7) cu atenuare de 11 dB și măsori nivelul bateriei în cod, transformând valoarea ADC invers la tensiunea reală a pachetului.

Divizor node	ESP32 Pin	Protocol	Descriere
Între R1 și R2	GPIO35	ADC1_CH7 (ADC_11db)	Măsurare tensiune baterie (factor 2×)

Alimentare și încărcare baterie

Două celule Li-Ion Samsung 25R (3,7 V nominal, 2 500 mAh fiecare) montate paralel într-un holder “go-bare”. În paralel se obtin ~5 000 mAh și tensiune constantă de 3,7 V, cu curent disponibil de zeci de amperi pentru perioade scurte.

Încărcător CC/CV pentru o singură celulă Li-Ion, alimentat la 5 V (micro-USB sau boost). Conține circuit de protecție la supraîncărcare, scurt-circuit și sub-tensiune. Pinii B+/B– se conectează la baterii, iar OUT+/OUT– livrează tensiunea protejată către consolă.

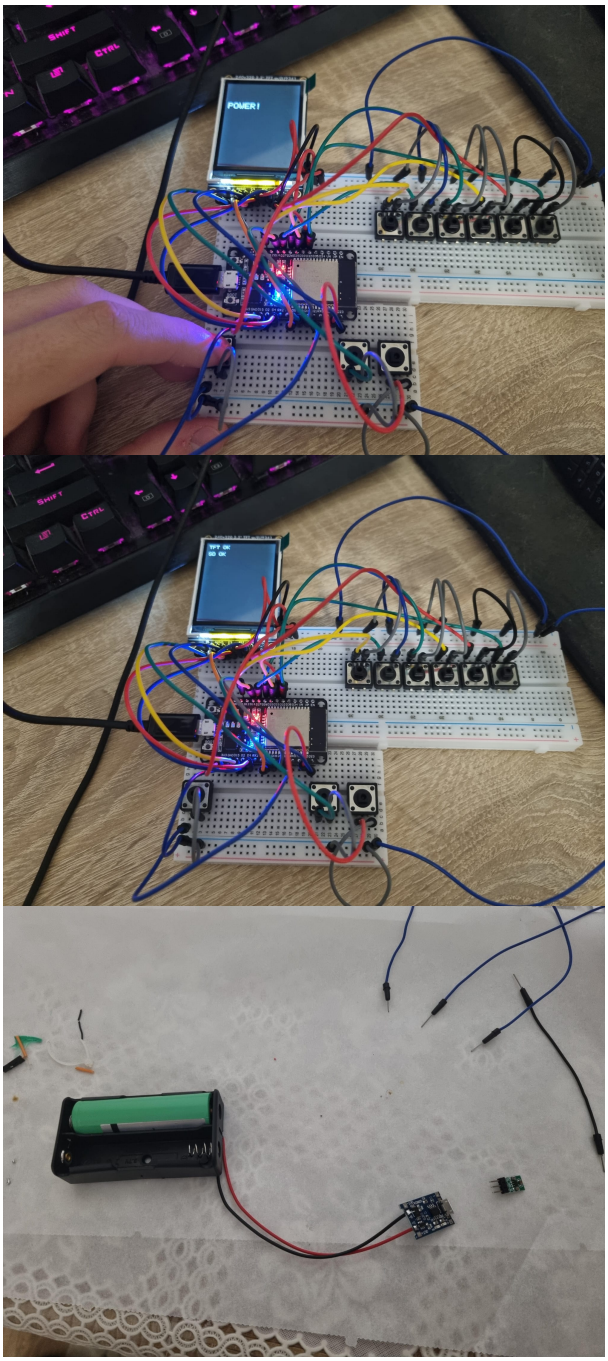
- TP4056 VIN (+) → 5 V de la boost/USB
- TP4056 GND (–) → GND comun
- TP4056 B+ / B– → suportul 2×18650 (celule în paralel)
- TP4056 OUT+ / OUT– → rail de alimentare (spre VIN al ESP32)

Bill of materials

Nume piesă	Cantitate	Link
Placă de dezvoltare ESP32 cu WiFi & Bluetooth 4.2	1	ESP32
Display LCD TFT 320×240 2.2″ cu slot microSD	1	Display TFT 2.2″
Modul DC-DC 1.5–5 V → 3.3 V	1	DC-DC 1.5–5 V→3.3 V
Modul încărcător TP4056 LiPo 1 A cu protecție	1	TP4056 charger

Card de memorie MicroSDHC 4 GB, Class 10	1	MicroSD card 4 GB
Suport baterii 2×18650	1	Battery holder 2×18650
Celule Li-Ion 18650 3.7 V	2	18650 Li-Ion 3.7 V
Buzzer pasiv	1	–
Push-buttons tactile (momentary)	9	–
Fire jumper (duse-întoarse)	10	–
Rezistoare 100 kΩ	2	–

Proof of concept



Software Design

Link proiect Github: https://github.com/Bogdan-Rusu17/Retro_Games_Console

Link video cu demo: https://youtu.be/d9HI2PYjlhI

Cateva fragmente de cod interesante:

Cum se reprezinta piesele de tetris?

```
const int8_t shapes[7][4][4][2] = {
  // I-piece
  { // 0°        90°          180°        270°
    {{-2,0},{-1,0},{0,0},{1,0}},    //orizontal
    {{0,-2},{0,-1},{0,0},{0,1}},    // vertical
    {{-2,-1},{-1,-1},{0,-1},{1,-1}},// orizontal
    {{-1,-2},{-1,-1},{-1,0},{-1,1}} // vertical
  },
  //...
 };
```

- formele in coordonate carteziene pentru piesele de tetris
- se porneste de la o forma fixa pentru fiecare piesa
- si cel mai simplu mod de a obtine coordonatele pentru o rotatie
- e sa inmultim cu matricea de rotatie 90 de grade:

{ {cos(pi / 2), sin(pi / 2)}, {cos(pi / 2), -sin(pi / 2)} }

Intreruperi pe butoane

```
void IRAM_ATTR onBtn(volatile uint32_t &lastTime, uint32_t debounce,
volatile bool &flag) {
  uint32_t now = micros();
  if (now - lastTime > debounce) flag = true;
  lastTime = now;
}
```

- helper de ISR pt butoane ca sa nu am cod duplicat
- verific la debounce sa fi trecut un numar de secunde ca sa iau apasarea in considerare

```
void IRAM_ATTR onUp() {
  if (state == TETRIS) {
    onBtn(lastUp, DEB_SOUND, btnUpFlag);
  } else if (state == SNAKE) {
    onBtn(lastUp, DEB_FAST, btnUpFlag);
  }

  onBtn(lastUp, DEB_SOUND, btnUpFlag);
}

// alte butoane
```

- interuperi pentru butoane cu rate custom de debounce dependent de status consola
- de exemplu la butonul UP cand suntem in jocul de tetris cu el rotim piesa si nu
- vrem sa se roteasca foarte repede ca sa controlam precizia
- cand suntem pe meniu vrem cam tot acelasi debounce rate
- dar cand suntem pe snake vrem sa avem rapiditate

PWM pentru controlul backlight ecran

```
  pinMode(TFT_BL, OUTPUT);
  analogWrite(TFT_BL, brightness);

  ledcAttachChannel(BUZZER_PIN, 2000, 8, 2);
  ledcWriteTone(BUZZER_PIN, 0);
  //...
```

- initializez PWM-ul pe pinul capabil PWM TFT_BL pe canalul 0 pe care si-l ia implicit
- initializez buzzer ul pe canalul 2 de PWM ca sa nu intre in conflict cu backlight-ul

```
if (digitalRead(BTN_A)==LOW && now - lastBright > BRIGHT_DEBOUNCE_MS) {
    brightness = min(brightness + 16, 255);
    analogWrite(TFT_BL, brightness);
    lastBright = now;
}
if (digitalRead(BTN_B)==LOW && now - lastBright > BRIGHT_DEBOUNCE_MS) {
    brightness = max(brightness - 16, 0);
    analogWrite(TFT_BL, brightness);
    lastBright = now;
}
```

Prin functia `analogWrite` cresc sau descresc factorul de umplere ca sa maresc/micsorez luminozitatea ecranului in termeni discreti pana la rezolutia maxima a timer-ului 0 pe care se bazeaza acest PWM.

SPI pentru comunicatie eficienta prin API-ul ecranului

```
#define TFT_DC       2
#define TFT_RST     -1  // pus la 3v3
#define TFT_BL      15  // pwm pt luminozitate ecran
#define SD_CS        4   // select la card

// instantiaza obiectul tft cu care am acces la functiile ecranului
Adafruit_ILI9341 tft(TFT_CS, TFT_DC, TFT_RST);
/* {...} */
SPI.begin(18, 19, 23);
tft.begin();
tft.setRotation(0);
```

- initializam liniile protocolului SPI specificand in ordine SCLK, MISO si MOSI
- setam ecranul sa functioneze in regimul de rotatie obisnuita adica cu verticala mai lunga decat orizontala

Cum folosesc ADC pentru a determina nivelul bateriei?

```
analogReadResolution(12);
analogSetAttenuation(ADC_11db);
pinMode(BAT_ADC_PIN, INPUT);
```

- ADC pentru baterie, 12 biti rezolutie, 11 db pt zgomot atenuare

```
void drawBattery() {
  // updateaza bateria doar odata la 1s ca sa nu fie flicker
  int now = millis();
  if (now - lastBatUpdate < 1000) {
    return;
  }

  lastBatUpdate = now;

  int raw = analogRead(BAT_ADC_PIN);
  float m = (raw / 4095.0f) * 3.3f, v = m * 2.0f * BAT_CAL_FACTOR;
  int pct = constrain(int((v - BAT_EMPTY_VOLTAGE)
            / (BAT_FULL_VOLTAGE - BAT_EMPTY_VOLTAGE) * 100 + 0.5), 0, 100);
```

```
  tft.fillRect(0, 0, tft.width(), 20, ILI9341_BLACK);
  tft.setCursor(2,2);
  tft.setTextSize(2);
  tft.setTextColor(ILI9341_WHITE);
  tft.print(v,2);
  tft.print("V");
  tft.setCursor(tft.width()-50, 2);
  tft.print(pct);
  tft.print("%");
}
```

- citeste valoarea pe 12 biti de pe pinul de ADC
- obtinem tensiunea la pinul ADC inmultind cu valoarea de referinta
- apoi obtinem valoarea reala a tensiunii din BAT+ a TP4056 inmultind cu 2
- pt ca avem divizor de tensiune cu 2 rezistenta egale
- calculam procentul de baterie rotunjit la cel mai apropiat int

Rezultate Obţinute

Link varianta finala

Concluzii

Per total a fost un proiect destul de interesant de realizat in care am imbinat si cunostinte de electronica si de programare intr-un mod destul de placut si sunt destul de multumit de cum a functionat totul intr-un final.

Mi-ar fi placut in schimb sa gasesc niste fire mai bune de lipit cu care sa realizez proiectul pe perfboard intrucat ar fi aratat mult mai clean..., dar a fost interesant sa imi reamintesc cum se fac lipituri, pentru partea de baterii si la ecran (nu avea pinii pusi).

Jurnal

1. 29 Aprilie - Alegere tema proiect
2. 11 Mai - Realizare schema bloc si schema electrica in Fritzing
3. 18 Mai - Realizarea hardware-ului in prima etapa
4. 25 Mai - Realizarea software in prima etapa
5. 29 Mai - Definitivarea hardware, software si a aspectului final al proiectului

Bibliografie/Resurse

Datasheet ESP32

Ledc library for PWM

Inspiratie pentru reprezentarea de tetromino-uri

Tetromino

Adafruit library for gfx

Adafruit manual pentru ecran