Flashback photo frame

Nume: Craciun Alexandra-Georgiana

Grupa: 331CB

Introducere

O ramă foto digitală este un mijloc modern și elegant de a păstra momentele frumoase și de a ne reconecta cu amintirile noastre și cu cei dragi. Poate conține o diversitate de poze pline de nostalgie, ce îți vor lumina ziua în funcție de cât de înnorat e afară, prin intermediul unui senzor aferent, si poate oferi un ton mai dramatic, prin functionalitatea de a transforma pozele si in format alb-negru. Poate fi un cadou prețios, sau un element central al decorului de acasă, aducând un plus de căldură și personalitate oricărui spațiu. Motivația din spatele dezvoltării acestui proiect vine din dorința de a combina tehnologia modernă cu nevoia umană de a celebra momentele importante din viață.

Descriere generală

Placuta Arduino va fi alimentata prin cablu USB de la laptop. Proiectul va citi din cardul SD pozele si le va afisa la un interval de 5s pe ecranul LCD. Senzorul de luminozitate trimite informatie la placuta Arduino, care mai apoi o va folosi pentru a adapta luminozitatea pozelor de pe ecran. Utilizatorul are posibilitatea de a apasa un buton, ce va transforma pozele color in poze alb-negru. Proiectul dispune de alte 2 butoane, prin care se poate derula prin poze inainte si inapoi.

Celule roz, dispozitive de intrare, trimit informatie la Arduino:

- Senzor intensitate: masoara intensitatea luminii din camera
- Buton Filtru: odata apasat, pozele vor fi in format alb-negru
- Buton Next: odata apasat, se va afisa urmatoarea poza; daca e apasat de x ori sare peste x poze
- Buton Prev: asemanator cu Next, doar ca deruleaza pozele inapoi
- Card SD: se vor citi pozele puse pe card, ulterior din afisate pe ecran

Celula verde, dispozitiv de iesire, ecranul LCD pe care se vor afisa pozele

Hardware Design

Lista de piese:

1. placa Arduino
2. breadboard
3. display LCD SPI 1.8”
4. senzor luminos
5. card SD + adaptor
6. butoane
7. fire
8. rezistente

Conectare piese:

- Ecran LCD - Arduino:

1. GND pe GND
2. VCC pe 5V
3. RESET pe pinul 8
4. A0 pe pinul 9
5. SDA pe pinul 11
6. SCL pe pinul 13
7. CS pe pinul 10
8. SCK pe pinul 13
9. MISO pe pinul 12
10. MOSI pe pinul 11
11. SD_CS pe pinul 4
12. LED+ pe pinul 5
13. LED- pe GND

- Senzor lumnozitate - Arduino

1. GND pe GND
2. VCC pe 5V
3. DO pe pinul 3

- Buton Filtru- Arduino

1. Un brat la GND cu o rezistenta de 10k
2. Un brat la 5V
3. Un brat la pinul 2

- Buton Next- Arduino

1. Un brat la GND cu o rezistenta de 10k
2. Un brat la 5V

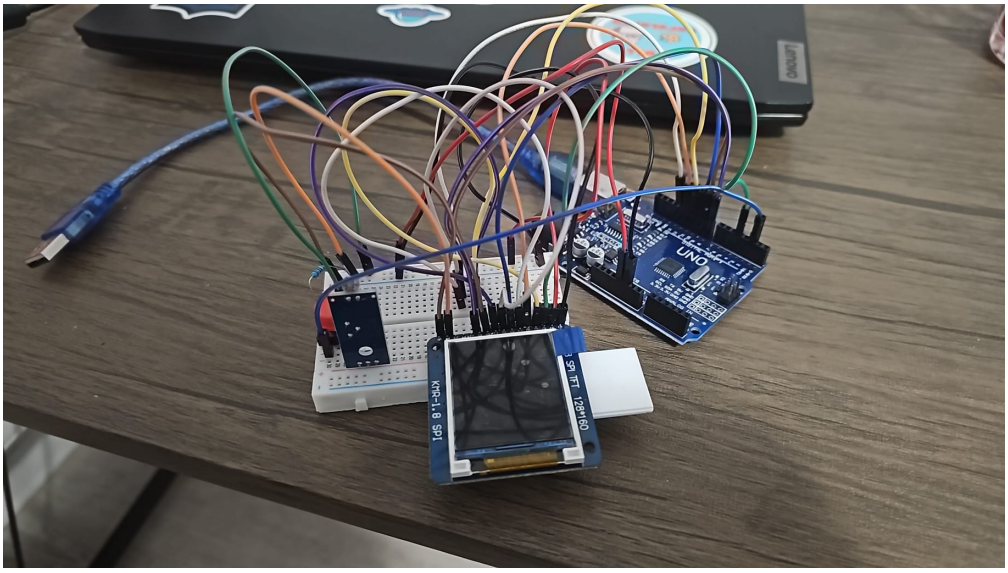
3. Un brat la pinul 6

- Buton Prev- Arduino

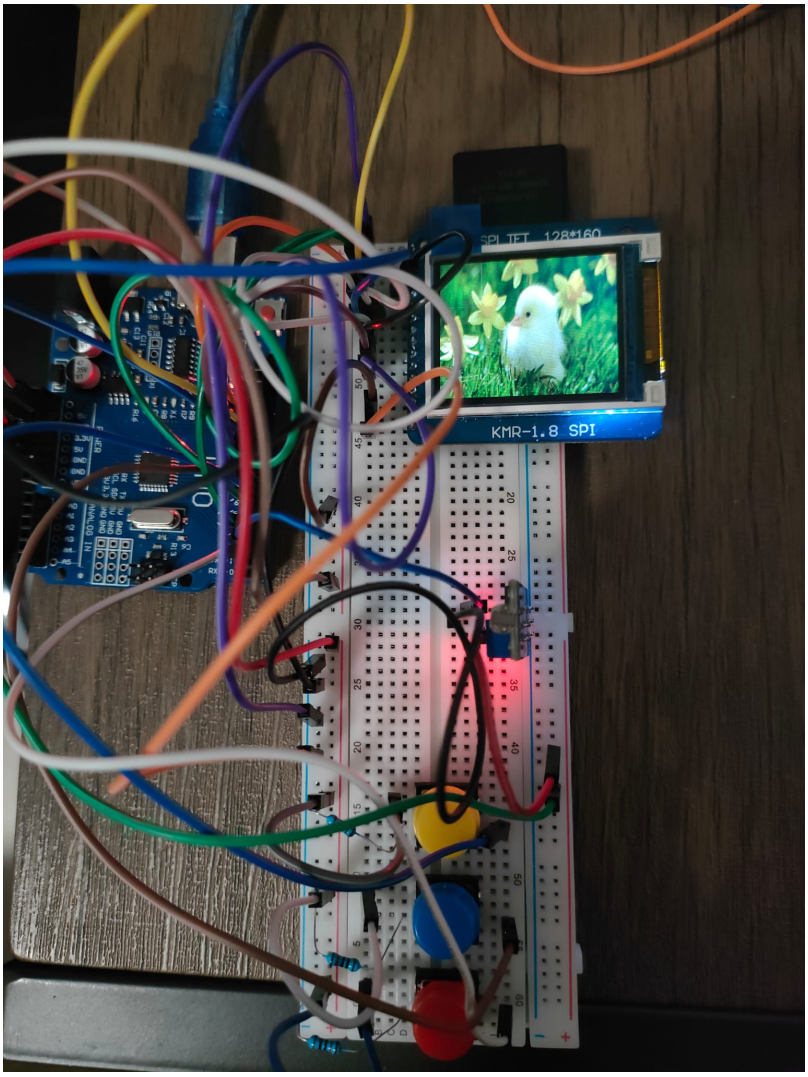
1. Un brat la GND cu o rezistenta de 10k
2. Un brat la 5V
3. Un brat la pinul 7

Asamblare hardware:

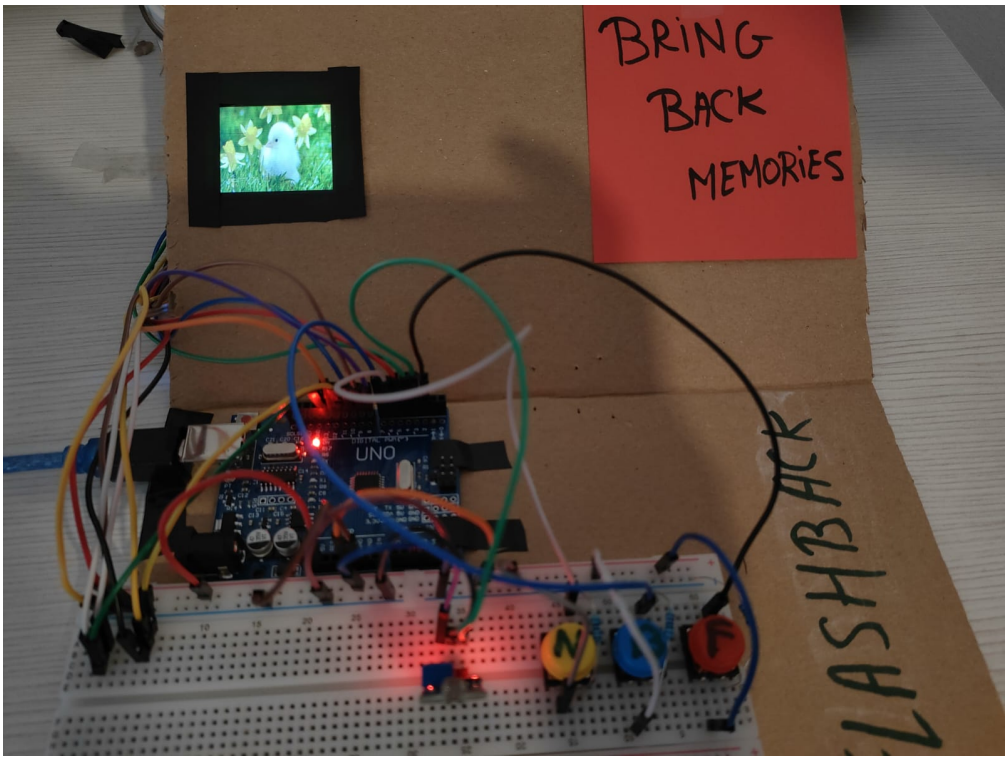
Inainte de cod



Varianta intermediara



Varianta finala



Software Design

Am scris codul in Arduino IDE si am utilizat bibliotecile:

- SPI.h: pentru comunicarea prin protocolul SPI cu ecranul LCD și cardul SD.
- SD.h: pentru accesul și citirea fișierelor BMP de pe cardul SD.
- TFT.h: pentru interacțiunea și desenarea pe ecranul TFT.
- LinkedList.h: pentru gestionarea listei de fișiere BMP.

Functionalitate

În codul dat, inițializarea sistemului începe prin configurarea piniilor și inițializarea componentelor hardware, precum ecranul TFT și cardul SD. Gestionarea întreruperilor permite detectarea rapidă a apăsării butoanelor, a acțiunilor utilizatorului si a schimbarilor de luminozitate. Principala funcție de execuție, loop(), coordonează afișarea imaginilor BMP pe ecranul TFT, încărcând și afișând fiecare imagine în conformitate cu interacțiunile utilizatorului. Funcția bmpDraw() este responsabilă de deschiderea și afișarea fiecărei imagini BMP, inclusiv aplicarea unui filtru alb-negru opțional.

Variabile globale

```
#define PIN_SD_CS 4
#define PIN_TFT_CS 10
#define PIN_DC 9
#define PIN_RST 8
#define BUTTON_PIN 2
#define BACKLIGHT_PIN 5
#define SENSOR_PIN 3
#define BUTTON_PIN_NEXT 6
#define BUTTON_PIN_BACK 7

#define DELAY_IMAGE_SWAP 5000 // each image is shown for 5 seconds
#define BUFFPIXEL 20 // Number of pixels to buffer

int filtru = 0;
String currentFile;
unsigned long lastDebounceTime = 0;  // Last pressed time
unsigned long debounceDelay = 50;    // press interval

TFT TFTscreen = TFT(PIN_TFT_CS, PIN_DC, PIN_RST); //init screen

LinkedList<String> bmpFiles; // List to store all BMP files
int currentFileIndex = -1; // Index of the current file being displayed
```

Descriere Funcții

- Setup Inițial: **setup():**

Configurează pini pentru butoane, lumină de fundal și senzor de lumină. Inițializează ecranul TFT, cardul SD și setează întreruperi pentru butoanele de control.

```
 void setup() {
    // initialize default serial connection to send debug information
    Serial.begin(9600);
    while (!Serial) {
        // wait until default serial connection is fully set up
    }
    // Initialize button filter pin
    pinMode(BUTTON_PIN, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(BUTTON_PIN), buttonInterrupt,
FALLING); // Attach interrupt to the button pin
    // Initialize backlight pin and sensor pin
    pinMode(BACKLIGHT_PIN, OUTPUT);
    pinMode(SENSOR_PIN, INPUT);
    attachInterrupt(digitalPinToInterrupt(SENSOR_PIN), senzorIntrerrupt,
CHANGE); // Attach interrupt to the button pin

    // Initialize next and back pin
    pinMode(BUTTON_PIN_NEXT, INPUT_PULLUP);
    pinMode(BUTTON_PIN_BACK, INPUT_PULLUP);

    PCICR |= (1 << PCIE2);  // Enable Pin Change Interrupts for D0 to D7
(PCINT16 to PCINT23)
    PCMSK2 |= (1 << PCINT22);  // Enable pin change interrupt on pin 6
(PCINT22)
    PCMSK2 |= (1 << PCINT23);  // Enable pin change interrupt on pin 7
(PCINT23)

    sei();  // Enable global interrupts

    // The following two lines replace "TFTscreen.begin();" => avoids that
red and blue (?) are swapped/interchanged
    TFTscreen.initR(INITR_BLACKTAB);
    TFTscreen.setRotation(1);

    TFTscreen.background(255, 255, 255); // prints black screen to TFT
display
    analogWrite(BACKLIGHT_PIN, 250);

    init_SD(); // function call that initializes SD card

}
```

- Gestionarea Întreruperilor: **buttonInterrupt() & ISR(PCINT2_vect) && senzorIntrerrupt():**

1. Această funcție este apelată la apăsarea butonului pentru aplicarea filtrului alb-negru. Folosește o verificare anti-debounce pentru a evita multiple declanșări dintr-o singură apăsare. Updateaza filtru si apeleaza functia de draw cu noul filtru.

```
 void buttonInterrupt() {
    unsigned long currentMillis = millis();
```

```
    if (currentMillis - lastDebounceTime > debounceDelay) {
        if (digitalRead(BUTTON_PIN) == LOW) {
            filtru = (filtru + 1) % 2;
            bmpDraw((char*)currentFile.c_str(), 0, 0, filtru); // Redraw
with filter
        }
        lastDebounceTime = currentMillis; // Update last
    }
}
```

2. Gestionează întreruperile pentru butoanele de next și back. Verifică starea butoanelor și updateaza indexul fișierului afisat în funcție de apăsările detectate. Apeleaza functia de draw cu noua imagine.

```
 ISR(PCINT2_vect) {
    unsigned long currentMillis = millis();
    if (currentMillis - lastDebounceTime > debounceDelay) {
        bool nextPressed = digitalRead(BUTTON_PIN_NEXT) == HIGH;
        bool backPressed = digitalRead(BUTTON_PIN_BACK) == HIGH;

        if (nextPressed && !backPressed) {
           currentFileIndex = (currentFileIndex + 1) % bmpFiles.size();
        }
        if (backPressed && !nextPressed) {
             if (currentFileIndex == 0) {
                 currentFileIndex = bmpFiles.size() - 1; // Wrap to last
image
              } else {
                 currentFileIndex--;
              }
        }

        currentFile = bmpFiles.get(currentFileIndex);
        // Draw the current image
        bmpDraw((char*)currentFile.c_str(), 0, 0, filtru);
        lastDebounceTime = millis();
    }
}
```

3. Gestionează întreruperile de la senzorul de lumină. La detectarea unei schimbări în starea senzorului, aceasta ajustează luminozitatea ecranului TFT, astfel incat la intuneric luminozitatea va fi redusă, iar la lumină, ridicată.

```
void senzorIntrerrupt() {

  int senzor = digitalRead(SENSOR_PIN);
  if (!senzor)
      analogWrite(BACKLIGHT_PIN, 250);
  else
      analogWrite(BACKLIGHT_PIN, 100);
}
```

- Inițializarea Cardului SD: **init_SD()**

Inițializează cardul SD și verifică dacă a fost montat corect. Parcurge directorul rădăcină al cardului SD și adaugă toate fișierele BMP găsite în lista bmpFiles.

```
 void init_SD() {
    // try to init SD card
    Serial.print(F("SD card init..."));
    if (!SD.begin(PIN_SD_CS)) {
        Serial.println(F("ERROR")); // failed
        return;
    }
    Serial.println(F("SUCCESS")); // ok

    // Store all BMP files in the bmpFiles list
    File dir = SD.open("/");
    File entry;
    while ((entry = dir.openNextFile())) {
        if (!entry.isDirectory() && String(entry.name()).endsWith(".BMP")) {
            bmpFiles.add(String(entry.name()));
        }
        entry.close();
    }
    dir.close();

    if (bmpFiles.size() > 0) {
        currentFileIndex = 0; // Start with the first file
    }
}
```

- Funcția Principală de Executare: **loop()**

Încarcă și afișează imaginea BMP curentă de pe cardul SD.

```
 void loop() {
  currentFile = bmpFiles.get(currentFileIndex);
  // Draw the current image
  bmpDraw((char*)currentFile.c_str(), 0, 0, filtru);
  delay(DELAY_IMAGE_SWAP);
  currentFileIndex= (currentFileIndex + 1) % bmpFiles.size(); //go default
to new image
}
```

- Afișarea Imaginilor BMP: **bmpDraw()**

Are ca parametrii numele file ului, coordonatele de pe ecranul TFT unde dorim să începem afișarea imaginii si un int ce reprezinta filtrul. Deschide fișierul BMP specificat și citește headerul pentru a obține informații despre dimensiuni și format. Parcurge pixelii imaginii și îi afișează pe ecranul TFT, aplicând filtrul alb-negru dacă este activat.

- Funcții Helper pentru Citirea Datelor: **read16() & read32()**

Citește 16 și, respectiv, 32 de biți din fișierul BMP pentru a obține informații necesare despre headerul imaginilor.

Download

pm_photo_frame.rar

Rezultate Obţinute

Rezultatul a fost unul satisfacator, intrucat toate functionalitatile dorite au fost implementate

Link catre demo pe Youtube: https://youtu.be/lRo6TX4jXq0?si=5vUYiCinLNkvlKRF

Concluzii

Pe parcursul acestui proiect, am intampinat diverse dificulatati, cum ar fi fire ce nu faceau bine contact, folosirea gresita a butoanelor sau a intreruperilor, formatul gresit al pozelor de pe card. Consider ca cea mai grea parte a fost software, nu hardware, intrucat am avut destul de putine piese care insa faceau multe instructiuni ce trebuiau sincronizate Au fost necesare documentatie extra, incercari destul de multe, emotii cand dadea black screen sau cand nu mai citea USB -ul. Iar tot procesul a fost usor diferit de ceea ce eram obisnuita, implicand acum si parte hardware si rezultat real-time, dar si frica ca poti sa arzi lucruri si nu mai ai timp sa ti vina alta comanda de piese

Totusi, a fost o experienta interesanta, iar satisfactia rezultatului final a fost uriasa. Pe masura ce mai reuseam sa fac o functionalitate, prindeam si mai mult drag de proiectul pe care il voi pastra, cu siguranta, ca amintire.

Jurnal

- 05.05.2024: Creare pagina wiki
- 13.05.2024: Comanda piese
- 16.05.2024: Verificarea pieselor si asamblarea lor
- 17.05.2024: Adaugare scheme hardware
- 19.05.2024: Scriere cod pentru ecran si buton + citire card SD
- 20.05.2024: Adaugarea senzorului si a filtrului alb-negru fara intreruperi, ci cu iterare prin for

- 21.05.2024: Adaugare intreruperi
- 22.05.2024: Fixed a bug cu intreruperi; am crezut ca am ars ceva
- 23.05.2024: Adaugare butoane next/prev + mutat tot pe breadboard mai mare

Bibliografie

Resurse Hardware:

- https://mschoeffler.com/2019/06/20/arduino-tutorial-making-the-kmr-1-8-spi-tft-display-work/
- https://docs.arduino.cc/built-in-examples/digital/Button/

Resurse Software:

- https://www.electronics-lab.com/project/arduino-diy-photo-frame/
- https://www.arduino.cc/reference/en/libraries/tft/
- https://github.com/sparkfun/SparkFun-Eagle-Libraries

Export to PDF