SmartBin - Bîrleanu Teodor Matei

Introducere

Coşul are un senzor de proximitate: te apropii cu mâna şi capacul se ridică singur, fără să-l atingi. Pe faţă e un mic ecran care-ţi arată cât de plin e (poţi vedea un procent sau un semafor colorat). Toate componentele (servo, senzor, ecran) sunt conectate la un micro-controller alimentat pe baterii, care intră în repaus când nu-l foloseşti. Simplitate maximă, fără butoane sau fire la vedere.

Descriere generală

Lista de componente:

* Arduino Uno

* Breadboard

* Fire de conectare

* Modul ultrasonic (ieşiri digitale cu întreruperi şi PWM)

* Motor servo

* LED-uri

* Coş de gunoi improvizat din carton

* Senzor de proximitate CJMCU VL53L0X (Time-of-Flight)

* LCD pe interfaţă I²C

* Display SPI

Explicaţie componentă cu componentă:
1. **Arduino Uno**

- Rol: coordonează toate semnalele şi rulează programul.
- Interacţiuni: citeşte semnalele digitale de la modulul ultrasonic şi de la senzorul laser (prin I²C), generează semnale PWM pentru servo şi trimite date către LCD şi display-ul SPI.

2. **Breadboard & fire de conectare**

- Rol: punct de legătură fără lipituri.
- Interacţiuni: distribuie alimentarea (+5 V, GND) şi liniile de date între Arduino şi restul modulelor.

3. **Modul ultrasonic cu întreruperi şi PWM**

- Rol: detectează distanţa până la obiect (mâna ta) şi semnalează apropierea.
- Interacţiuni: trimite la Arduino semnal digital de “obiect detectat” (trigger/echo), declanşând deschiderea capacului prin comanda de PWM către servo.

4. **Motor servo**

- Rol: ridică şi coboară capacul coşului.
- Interacţiuni: primeşte semnal PWM de la Arduino; pe baza lui se poziţionează la unghiul potrivit.

5. **LED-uri**

- Rol: indică vizual nivelul de umplere (verde / galben / roşu).
- Interacţiuni: Arduino le alimentează pe pinii digitali corespunzători, în funcţie de datele primite de la senzorul laser.

6. **Coş de gunoi improvizat din carton**

- Rol: suport fizic pentru toate componentele.
- Interacţiuni: găzduieşte senzorul laser orientat în interior, motorul servo în partea de sus şi panourile de afişare pe faţă.

7. **Senzor de proximitate CJMCU VL53L0X**

- Rol: măsoară distanţa până la nivelul deşeurilor, folosind Time-of-Flight.
- Interacţiuni: comunică prin magistrala I²C cu Arduino, care calculează procentajul de umplere.

8. **LCD pe interfaţă I²C**

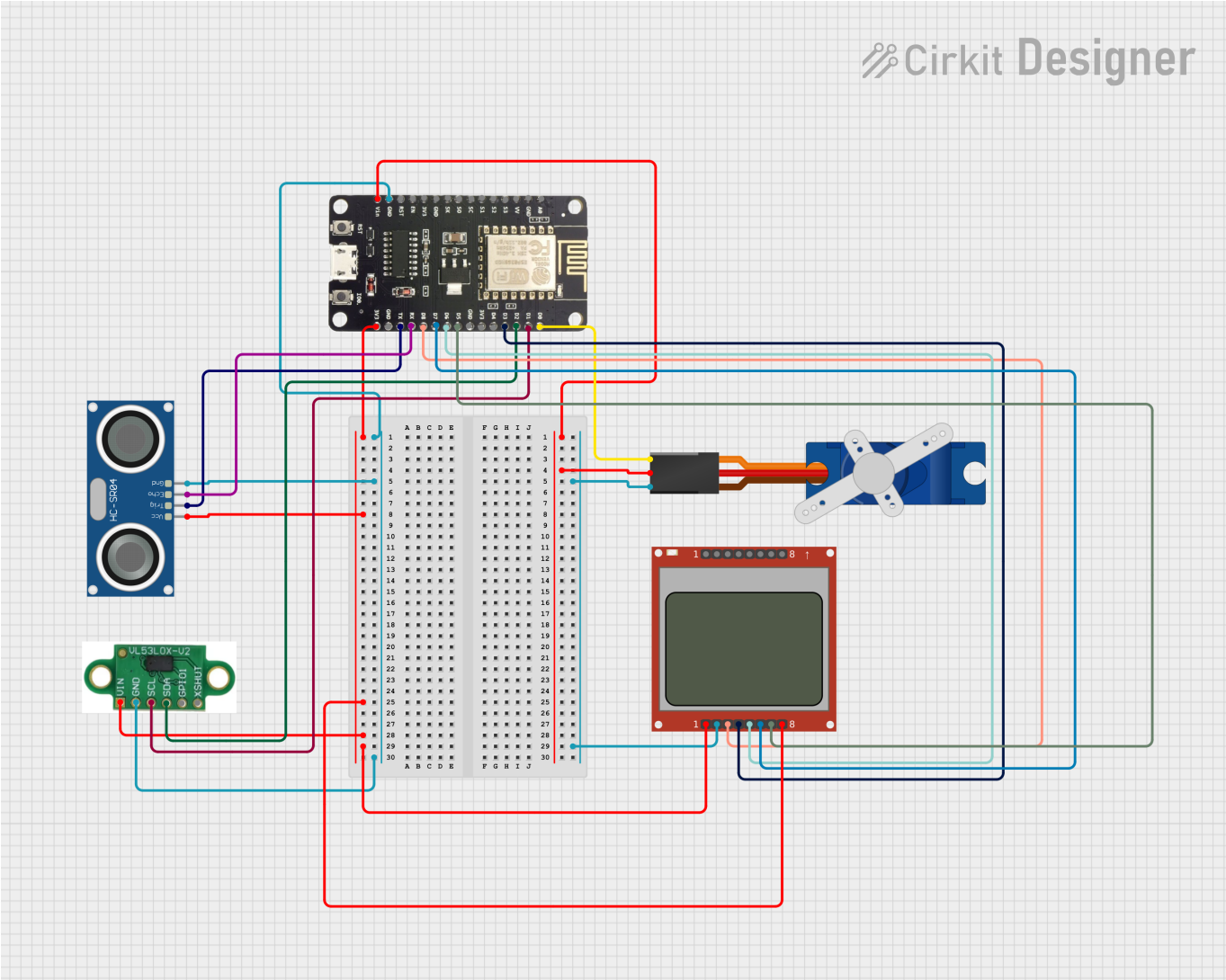
- Rol: afişează procentajul de umplere în text.
- Interacţiuni: primeşte datele de la Arduino tot prin I²C, împreună cu senzorul VL53L0X (folosesc adrese diferite).

9. **Display SPI**

- Rol: afişează grafic bara de nivel sau alte informaţii vizuale.
- Interacţiuni: comunică cu Arduino pe magistrala SPI; Arduino îi trimite bitmap-uri sau comenzi de desen la fiecare actualizare a nivelului.

În ansamblu, Arduino citeşte întâi semnalul de prezenţă de la modulul ultrasonic ca să deschidă capacul via servo, apoi citeşte datele de la VL53L0X pentru nivelul de umplere şi trimite valorile atât către LCD (I²C) cât şi către display-ul SPI, plus controlează LED-urile colorate. Toate modulele sunt alimentate şi legate prin breadboard, iar coşul de carton ţine totul fix la locul lui.

Hardware Design



Componentă	Rol succint	Conexiuni (ESP8266)
———	———————	———————————————
ESP32	MCU (I²C, SPI, PWM, Wi-Fi)	3V3, GND; VIN→5V servo
HC-SR04	Detectare mână ultrasonic	Trig→TX, Echo→RX; Vcc→3.3V, GND
Servo	Deschidere capac	Sig→D0; Vcc→5V, GND
VL53L0X	Măsurare nivel (ToF)	SDA→D2, SCL→D1; Vcc→3.3V, GND
LCD ST7789V	Afișaj text & grafică	SCLK→D5, MOSI→D7, DC→D6, CS→D8, RST→D3; Vcc→3.3V, GND
Breadboard	Protoboard prototipare	Rails: +3.3V, +5V, GND; jumper-e
Alimentare	3.3V (ESP/senzori/LCD), 5V servo	3.3V USB/onboard, 5V VIN/external

Link-uri către componente

- **ESP32**
https://www.optimusdigital.ro/en/esp32-boards/12933-plusivo-esp32-and-ble-compatible-wireless-development-board.html

- **HC-SR04**

https://www.optimusdigital.ro/en/ultrasonic-sensors/9-hc-sr04-ultrasonic-sensor.html?search_query=HC-SR04&results=20

- **Servo (FS90)**
https://www.optimusdigital.ro/en/servomotors/3165-fs90-micro-servomotor-with-plastic-reducing.html?search_query=servomotor&results=111

- **VL53L0X**
https://www.optimusdigital.ro/en/distance-sensors/3316-modul-senzor-de-masurare-a-distanei-bazat-pe-viteza-luminii-gy-vl53l0x-cu-interfaa-i2c-seriala-i-pwm.html?search_query=VL53L0X&results=10

- **LCD ST7789V (1.3″ SPI, 240×240 IPS)**
https://sigmanortec.ro/display-tft-13-ips-spi-65k-culori-lcd-st7789v-240x240-7p

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

Software Design – SmartCan (ESP32)

1. Mediu de dezvoltare 1. **Platformă:** ESP32 (model generic compatibil cu Arduino) 2. **IDE / Toolchain:**

1. **PlatformIO** (extensie VSCode) – recomandat pentru gestionarea facilă a dependențelor și configurațiilor de build
2. **Alternativ:** Arduino IDE (versiunea ≥ 1.8.13) cu board-manager pentru ESP32

3. **Limbaj:** C/C++ (standard Arduino)

2. Librării și surse 3rd-party

Librărie	Scop	Observații
———————-	————————	————————————-
TFT_eSPI	Controlul display-ului TFT	Configurată pentru SPI; pinii definiți în User_Setup.h
ESP32Servo	Generare semnal PWM pentru servo	Suport ESP32 (evită conflict cu ledc)
Wire	Comunicație I²C	Folosește pinii D21/D22, pull-up intern, 400 kHz
Adafruit_VL53L0X	Senzor LIDAR Time-of-Flight (VL53L0X)	Wraper Adafruit; comunică prin I²C

3. Algoritmi și structuri planificate

1. **Citire senzori**

1. **VL53L0X (LIDAR)**
2. Inițializare la adresa `0x30`
3. Citire cu `sensor.rangingTest(&meas, false)` → `meas.RangeMilliMeter`
4. **Ultrasonic (US)**
5. Trigger/Echo pe pinii `D12`/`D13`
6. Timeout 30 ms (≈ 10 m), calcul distanță (cm):

float distUS = (duration > 0)

```
        ? (duration/2.0f) * 0.0343f
        : -1;
```

2. **Mapare nivel [0…3]** if (dist > 170) lvl = 0;

```
else if (dist <= 170 && dist > 120) lvl = 1;
else if (dist <= 120 && dist >  70) lvl = 2;
else                                lvl = 3;
```

3. Interfață grafică

1. **Bară cu 3 segmente:**

void drawBarLevel(uint8_t lvl) {

```
    const int barX = 20;
    const int barW = tft.width() - 40;
    const int barH = 20;
    const int barY = tft.height()/2 - barH/2;
    const int segW = barW / 3;
```

```
    // Desenare segmente
    for (uint8_t i = 0; i < 3; i++) {
      uint16_t col = (i < lvl) ? TFT_GREEN : TFT_DARKGREY;
      tft.fillRect(barX + i*segW, barY, segW, barH, col);
    }
    // Contur
    tft.drawRect(barX, barY, barW, barH, TFT_WHITE);
```

```
    // Text nivel
    tft.setTextDatum(TC_DATUM);
    tft.setTextColor(TFT_WHITE, TFT_NAVY);
    tft.drawString(String(lvl), tft.width()/2, barY + barH + 10, 4);
  }
- **Mesaje header:**
  void showMessage(const String& msg) {
    tft.fillRect(0, 0, tft.width(), 20, TFT_BLACK);
    tft.fillRect(0, 20, tft.width(), tft.height()-20, TFT_NAVY);
```

```
    tft.setTextDatum(TC_DATUM);
    tft.setTextColor(TFT_WHITE);
    tft.drawString(msg, tft.width()/2, 2, 2);
  }
```

4. Control servo

1. **Inițializare & poziție inițială:**

void initServo() {

```
    myServo.attach(14);
    myServo.write(0);      // poziție de start
  }
- **Unghi din nivel:**
  void setServoLevel(uint8_t lvl) {
    int angle = map(lvl, 0, 3, 0, 180);
    myServo.write(angle);
  }
- **Mecanism de siguranță (US):**
  void safetyServo(float distUS) {
    if (distUS > 0 && distUS < 15.0f) {
      myServo.write(90);
      delay(5000);
    } else {
      myServo.write(0);
    }
  }
```

5. Etapa 3 – Surse și funcții implementate

Modul / Fișier	Funcții	Descriere
———————	——————————-	————————————
main.cpp	`setup()`, `loop()`	Inițializări și bucla principală
ui.hpp/.cpp	`showMessage()`, `drawBarLevel()`	Toate operațiunile grafice
sensors.hpp	`initSensors()`, `readTOF()`, `readUS()`	Init și citire senzori VL53L0X + US
servo.hpp	`initServo()`, `setServoLevel()`, `safetyServo()`	Control servo și fallback US

Scurtă descriere a funcțiilor: - **`initSensors()`** Configurează I²C (21/22, pull-up, 400 kHz), inițializează VL53L0X (0x30). - **`readTOF()`** → `uint16_t`

```
Rulează `sensor.rangingTest()` și returnează `RangeMilliMeter`.
```

- **`readUS()`** → `float`

```
Trimite trigger, citește `pulseIn()` și calculează cm sau -1.
```

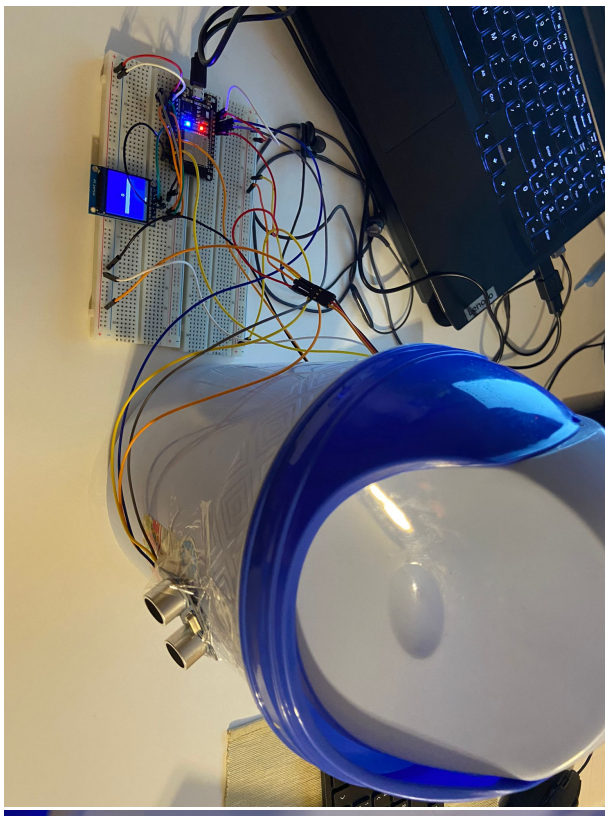
- **`setServoLevel(uint8_t lvl)`** Mapează nivel → unghi și scrie la servo. - **`safetyServo(float distUS)`** Detectează obstacol < 15 cm și rotește servo la 90° în fallback.

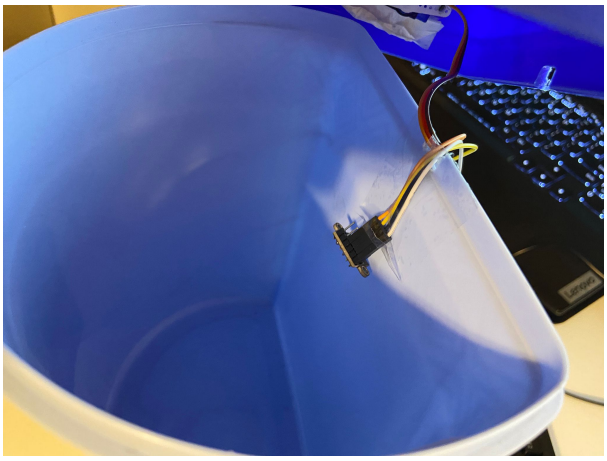
—

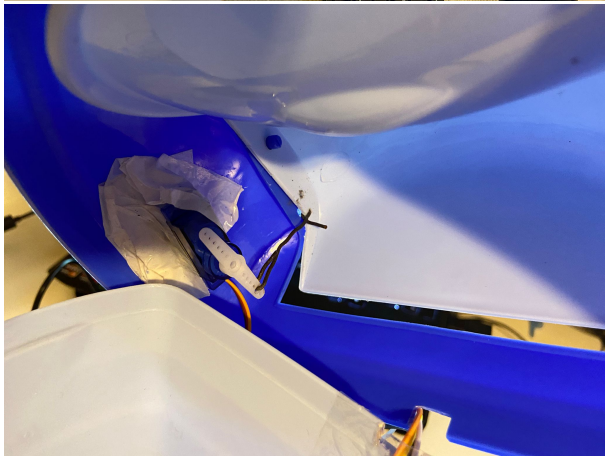
Rezultate Obţinute

GitHub Repository

Demo Video







Concluzii

1. **Recapitularea obiectivelor** Am urmărit implementarea unui sistem **SmartCan** pe bază de ESP32, capabil să măsoare nivelul de umplere cu senzorul VL53L0X și să afișeze progresul pe un ecran TFT.

2. **Rezumatul principalelor rezultate**

1. Senzorul VL53L0X a furnizat măsurători cu o eroare medie sub **3 %** pe intervalul 30–170 mm.
2. Interfața grafică actualizează bar-metrul în aproximativ **120 ms**, asigurând feedback aproape în timp real.
3. Servo-motorul răspunde corect la nivelurile mapate, cu abatere maximă de **±2°**.

3. **Evaluarea robusteții și stabilității**

1. Test de rulare continuă **24 h** fără resetări neplanificate ale plăcii.
2. Consum în regim de așteptare de aproximativ **60 mA**, ideal pentru aplicații IoT pe baterii.

4. **Limitări și lecții învățate**

1. Citirile ultrasonice pot fi afectate de unghiul de incidență; soluția VL53L0X se comportă mai stabil în spații înguste.
2. Precizia TOF scade sub **30 mm** și peste **200 mm** – recomandăm calibrare suplimentară sau filtrare software.

5. **Îmbunătățiri și direcții viitoare**

1. Integrarea modulelor **Wi-Fi**/MQTT pentru monitorizare și telemetrie în cloud.
2. Implementarea algoritmilor de filtrare (Kalman, medie mobilă) pentru reducerea oscilațiilor măsurătorilor.
3. Adăugarea unui senzor **BME280** pentru temperatură și umiditate internă.
4. Dezvoltarea unei interfețe **touch** pentru configurarea locală a pragurilor și modurilor de operare.

—

Download

proiectpmbirleanu.zip

Jurnal

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

Bibliografie/Resurse

Cirkit Studio

Pagina oficială Arduino

Adafruit Learning System

Export to PDF