AutoBin

Zeitouni Aila Grupa: 333CA

Introducere

Prezentarea pe scurt a proiectului:

AutoBin este un coș de gunoi inteligent care se deschide automat atunci când detectează prezența unei persoane sau a unui obiect în apropiere. Proiectul folosește un senzor cu ultrasunete pentru detecție și un servomotor pentru a acționa capacul.

Ce face:

Sistemul detectează apropierea unei persoane și acționează automat servomotorul pentru a deschide capacul. După ce persoana se îndepărtează, capacul se închide. În plus, datele despre distanță și acțiuni sunt transmise prin UART pentru monitorizare și depanare.

Care este scopul lui:

Scopul AutoBin este de a oferi o soluție igienică și practică pentru colectarea deșeurilor, eliminând nevoia de a atinge capacul manual. Astfel, se reduce riscul de contaminare și se îmbunătățește experiența utilizatorului.

Care a fost ideea de la care ați pornit:

Ideea a pornit de la nevoia de a evita contactul cu suprafețe potențial murdare în spațiile publice sau acasă. Am vrut să creăm un sistem simplu, accesibil și eficient, bazat pe tehnologii învățate în laborator, cum ar fi controlul servomotoarelor, senzorii de distanță și comunicația UART.

Care a fost ideea de la care ați pornit:

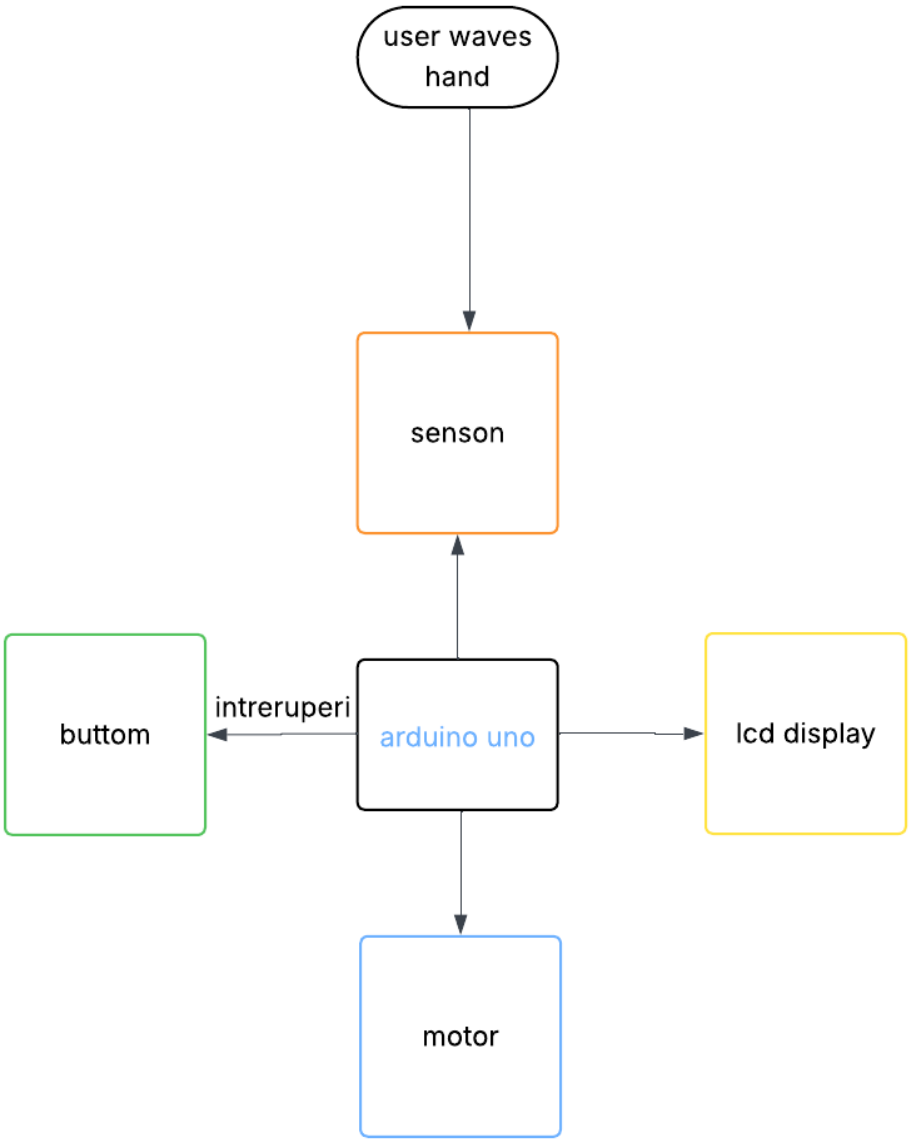
Ideea a pornit de la nevoia de a evita contactul cu suprafețe potențial murdare în spațiile publice sau acasă. Am vrut să creăm un sistem simplu, accesibil și eficient, bazat pe tehnologii învățate în laborator, cum ar fi controlul servomotoarelor, senzorii de distanță și comunicația UART.

De ce credeți că este util pentru alții și pentru voi:

Pentru alții, AutoBin oferă o modalitate igienică și practică de a arunca gunoiul, ceea ce este esențial în școli, spitale, birouri sau bucătării. Pentru mine, proiectul este o oportunitate de a aplica cunoștințele dobândite la laborator într-o soluție concretă, utilă și relevantă din punct de vedere social.

Descriere generală

schema bloc



List of Hardware Components

Component Function Arduino Uno Central controller, reads sensor data, controls motor/display/button Sensor Ultrasonic sensor to detect hand motion Motor Executes a physical task (e.g., servo motor for rotation or DC motor) LCD Display Shows information like “Motor On”, “Ready”, or sensor readings buzzer Used when sensor detects something fire mama-tata Connects components to Arduino + breadboard

Hardware Design

diagramă bloc

schema electrica

bom

Descriere proiect

Proiectul este construit în jurul plăcii **Arduino Uno**, care controlează și comunică cu mai multe componente externe pentru a forma un sistem interactiv cu funcții de măsurare, afișare și alertă.

Componente utilizate

- **Arduino Uno** – microcontroller principal (ATmega328P), controlează logica și comunică cu componentele prin Digital I/O, PWM și I2C.
- **Senzor HC-SR04** – măsoară distanța. Trig (D9), Echo (D8), comunicare digitală.
- **Buzzer pasiv** – semnal sonor de alertă, controlat prin D5 cu funcția tone().
- **Servo motor SG90** – controlat prin semnal PWM de la pinul D10, alimentare 5V.
- **LCD 16×2 cu I2C** – afișează valori de la senzori, conectat pe A4 (SDA) și A5 (SCL).

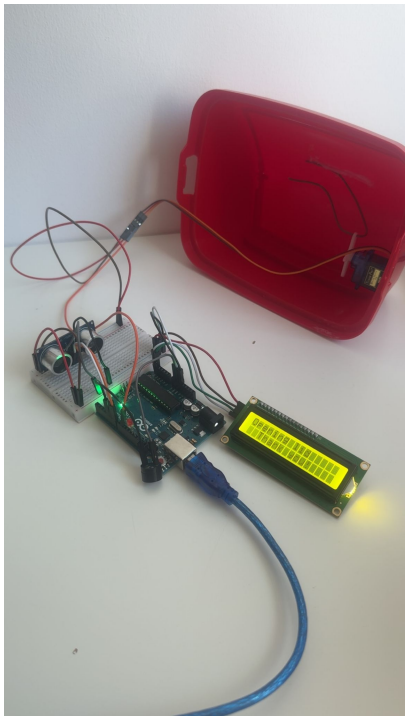
Alimentare

Arduino este alimentat prin USB (5V), iar toate componentele primesc alimentare prin pinii de 5V și GND, distribuiți prin breadboard. Toate componentele împart GND comun pentru referință corectă. Legăturile sunt realizate pe breadboard, cu grijă pentru alimentare și protecție. Comunicarea I2C permite extinderea ușoară a proiectului.

Interfețe și pini utilizați

Componentă	Interfață	Pini Arduino
HC-SR04	Digital	D9 (Trig), D8 (Echo)
Buzzer	Digital	D5
Servo	PWM	D10
LCD I2C	I2C	A4 (SDA), A5 (SCL)

hardware



Rolul fiecărei componente

- **Arduino Uno** – Acționează ca unitate centrală de control. Primește informații de la senzori, procesează datele și trimite comenzi către afișaj și actuatori.
- **Senzor HC-SR04** – Măsoară distanța până la un obiect aflat în fața sistemului folosind unde ultrasonice. Poate fi utilizat pentru declanșarea buzzer-ului sau pentru activarea servo-motorului când un obiect este detectat la o distanță prestabilită.
- **Buzzer pasiv** – Emite sunete ca semnal de alertă atunci când o condiție este îndeplinită (ex: distanță mai mică de 10 cm). Este controlat digital de Arduino prin funcția `tone()`.
- **Servo motor SG90** – Se rotește într-un anumit unghi (0°–180°) în funcție de comanda primită de la Arduino. Util în aplicații precum deschiderea unei uși automate, ridicarea unei bariere etc.
- **LCD 16×2 cu interfață I2C** – Afișează valori precum distanța măsurată sau starea senzorilor. Conectarea prin I2C permite economisirea pinilor digitali și oferă comunicare eficientă pe doar două fire (SDA și SCL).

Estimare consum energie

Componentă	Curent estimat
Arduino Uno	~50 mA
HC-SR04	~15 mA
Buzzer	~30–50 mA
Servo SG90	~100–250 mA
LCD I2C	~20 mA
Total estimat	**~300–400 mA**

Arduino poate alimenta toate componentele prin portul USB. Dacă servo-ul va fi utilizat intens, este recomandată o sursă externă de 5V pentru siguranță.

Stadiul actual și pașii următori

Montajul hardware a fost realizat și testat în mediul virtual (Tinkercad), cu rezultate funcționale. Conexiunile au fost validate, iar codul de control este pregătit. Urmează achiziționarea componentelor fizice, montarea pe breadboard real și testarea sistemului complet.

Software Design

#include <avr/io.h> #include <util/delay.h> #include <stdint.h> #include <stdbool.h> #include “lcd_i2c.h” #include “twi.h” *Pin defines matching your Arduino sketch #define TRIG_PIN PD6* D6 #define ECHO_PIN PD7 *D7 #define SERVO_PIN PB1* D9 (OC1A) #define BUZZER_PIN PB0 *D8 #define DISTANCE_OPEN_THRESHOLD 15 #define DISTANCE_TRASH_THRESHOLD 8 #define TRASH_DETECT_DELAY_MS 3000 #define SERVO_OPEN_ANGLE 0 #define SERVO_CLOSE_ANGLE 180 void setup_servo_pwm() { DDRB |= (1 « SERVO_PIN);* PB1 as output

```
  TCCR1A |= (1 << COM1A1) | (1 << WGM11);
  TCCR1B |= (1 << WGM12) | (1 << WGM13) | (1 << CS11); // prescaler 8
  ICR1 = 19999; // 20 ms PWM
```

}

void set_servo_angle(uint8_t angle) {

```
  OCR1A = (ICR1 * (1000 + ((uint32_t)angle * 1000 / 180))) / 20000;
```

}

uint16_t measure_distance_cm() {

```
  // Send trigger pulse
  PORTD &= ~(1 << TRIG_PIN);
  _delay_us(2);
  PORTD |= (1 << TRIG_PIN);
  _delay_us(10);
  PORTD &= ~(1 << TRIG_PIN);
```

```
  // Wait for echo to go high
  uint16_t timeout = 30000;
  while (!(PIND & (1 << ECHO_PIN)) && timeout--) _delay_us(1);
  if (timeout == 0) return 0;
```

```
  // Measure how long echo is high
  uint32_t count = 0;
  while ((PIND & (1 << ECHO_PIN)) && count < 30000) {
      _delay_us(1);
      count++;
  }
```

```
  // Convert to cm
  return (uint16_t)((count * 0.0343) / 2);
```

```
}
```

void buzz(uint8_t times) {

```
  for (uint8_t i = 0; i < times; i++) {
      PORTB |= (1 << BUZZER_PIN);
      _delay_ms(300);
      PORTB &= ~(1 << BUZZER_PIN);
      _delay_ms(200);
  }
```

} #include “lcd_i2c.h” #include “twi.h”

int main(void) {

```
  // Setup TRIG as output, ECHO as input
  DDRD |= (1 << TRIG_PIN);
  DDRD &= ~(1 << ECHO_PIN);
```

```
  // Setup buzzer
  DDRB |= (1 << BUZZER_PIN);
  PORTB &= ~(1 << BUZZER_PIN);
```

```
  // Init I2C and LCD
  twi_init();
  lcd_init();
```

```
  lcd_clear();
```

```
  lcd_set_cursor(0, 0);
  lcd_print("Smart Stash Can");
  lcd_set_cursor(0, 1);
  lcd_print("   Initializing ");
  _delay_ms(2000);
  lcd_clear();
```

```
  // Init servo
  setup_servo_pwm();
  set_servo_angle(SERVO_CLOSE_ANGLE);
```

```
  uint32_t last_thank_time = 0;
```

```
  while (1) {
      uint16_t distance = measure_distance_cm();
```

```
      if (distance > 0 && distance < DISTANCE_OPEN_THRESHOLD) {
          set_servo_angle(SERVO_OPEN_ANGLE);
          lcd_set_cursor(0, 0);
          lcd_print("Opening lid     ");
          buzz(1);
      } else {
          set_servo_angle(SERVO_CLOSE_ANGLE);
          lcd_set_cursor(0, 0);
          lcd_print("Lid closed      ");
      }
```

```
      if (distance > 0 && distance < DISTANCE_TRASH_THRESHOLD) {
          if ((last_thank_time + TRASH_DETECT_DELAY_MS) < 60000) {
              lcd_set_cursor(0, 1);
              lcd_print("  Thank you!    ");
              buzz(2);
              last_thank_time = 0;
          }
      } else {
          lcd_set_cursor(0, 1);
          lcd_print("Scanning area   ");
          last_thank_time += 300;
      }
```

```
      _delay_ms(300);
  }
```

```
}
```

Milestone 3 - Raport asupra implementării software

Stadiul actual al implementării

Software-ul pentru proiectul Smart Stash Can este funcțional și a fost testat cu succes pe hardware-ul final. Funcționalitățile implementate includ:

- Detectarea distanței utilizând senzorul ultrasonic HC-SR04 (TRIG pe PD6, ECHO pe PD7)
- Controlul capacului prin servomotor acționat PWM (OC1A - PB1)
- Feedback sonor cu ajutorul unui buzzer (PB0)
- Afișaj LCD I2C (cu PCF8574, adresă 0x27) pentru interfața cu utilizatorul
- Cod modular scris în C bare-metal cu acces direct la registrele perifericelor
- Demonstrație completă validată experimental în laborator

—

Biblioteci și drivere utilizate

Componentă	Bibliotecă / Driver	Justificare
Afișaj LCD	lcd_i2c.c / lcd_i2c.h (scris de noi)	Implementare ușoară și eficientă pentru PCF8574, fără dependențe de Arduino
Magistrală I2C	twi.c / twi.h (din laborator)	Permite control complet asupra comunicației TWI
UART (debug)	usart.c / usart.h	Folosit pentru mesaje de test și jurnalizare serială
Control servomotor	PWM cu Timer1	Precizie ridicată folosind OCR1A și ICR1 fără delay-uri software

Fiecare bibliotecă a fost selectată pentru simplitate, eficiență și integrare directă cu perifericele platformei ATmega328P.

—

Element de noutate al proiectului

Acest proiect se diferențiază prin abordarea complet bare-metal, fără utilizarea bibliotecilor Arduino. Interacțiunea între senzor, servomotor, buzzer și afișaj este realizată exclusiv prin programare directă a registrelor. Se obține astfel un control total asupra funcționalității, temporizării și performanței sistemului, într-un format educativ și extensibil.

—

Justificarea utilizării funcționalităților din laborator

Laborator	Funcționalitate utilizată	Aplicare în proiect
Lab 0	GPIO	TRIG și ECHO pentru senzor, ieșire pentru buzzer
Lab 1	UART	Trimiterea mesajelor debug către terminal serial
Lab 3	Timere și PWM	Control precis al servomotorului cu Timer1, canal OC1A
Lab 6	I2C (TWI)	Comunicație cu afișajul LCD prin interfața I2C (PCF8574)

Fiecare funcționalitate din laborator a fost integrată într-un mod practic pentru a forma un sistem embedded complet funcțional.

—

Structura proiectului și validare

Fișiere principale:

- `main.c` – logica principală, secvența de control
- `servo.c/h` – control PWM pentru servomotor
- `twi.c/h` – comunicație I2C (TWI)
- `lcd_i2c.c/h` – afișare pe LCD prin PCF8574
- `usart.c/h` – trimiterea mesajelor seriale

Validare:

- Testare modulară: fiecare componentă a fost verificată individual
- Testare integrată: comportamentul întregului sistem a fost observat în timp real
- Compararea distanțelor măsurate cu o riglă fizică
- Verificarea reacției servo în funcție de distanță
- Confirmarea clarității mesajelor LCD și a actualizării cursorului

—

Calibrarea senzorului de distanță

Senzorul ultrasonic HC-SR04 a fost calibrat folosind măsurători directe:

- Formula de conversie: `(durata în µs × 0.0343) / 2`
- Pulsurile TRIG și ECHO au fost gestionate cu `_delay_us()` pentru precizie
- Praguri stabilite experimental:
  - Deschiderea capacului la < 15 cm
  - Mesaj de mulțumire la < 8 cm
- Zgomotul de măsurare a fost redus prin cicluri scurte și perioade de eșantionare stabile (300 ms)

—

Optimizări realizate

- PWM hardware în loc de temporizări software pentru controlul servomotorului
- I2C implementat low-level fără librării externe, pentru control precis
- Suprimarea “ghosting”-ului pe LCD prin scriere completă a liniilor cu spații
- Utilizarea exclusivă a variabilelor statice, fără alocare dinamică
- Funcția de măsurare a distanței optimizată pentru precizie și timeout sigur
- Structura buclei principale minimizează întârzierile și optimizează reacția sistemului

—

Video demonstrativ

O demonstrație video a întregului proiect, cu explicații și comportament funcțional, poate fi vizualizată la:

—

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