Smart Pin Trash: Automatic Waste Classification System

Embedded Systems Project Report

Team members:

Team 17
عمر محمد محمد حامد
محمد ايمن فتحي عبد الغني
محمد خالد عكاشه
عمر محمد فتح الله
بيشوي خير
ايليا ايهاب عبد الله إبراهيم
دانيال اسامه حليم عزيز
احمد فوزي البيومي فوزى

Under supervision of:

Dr.Hossam El-Din ibrahim

Abstract

This project presents the development of "Smart Pin Trash," an innovative waste classification system that automatically sorts waste into metal and non-metal categories. The system utilizes sensors to detect waste type and proximity, facilitating efficient waste management. The project implements embedded systems principles using multiple ATmega32 microcontrollers, ultrasonic sensors, and inductive sensors to create an intelligent classification mechanism. This report outlines the system architecture, hardware components, implementation methodology, and functional analysis of the Smart Pin Trash system.

1. Introduction

Waste management presents significant environmental challenges globally. Effective sorting of waste at the source can substantially improve recycling efficiency and reduce environmental impact. The Smart Pin Trash system addresses this challenge by automatically categorizing waste into metal and non-metal components, eliminating the need for manual sorting and reducing contamination between different waste types.

1.1 Project Objectives

- Design and implement an automatic waste classification system using embedded systems
- Distinguish between metallic and non-metallic waste materials
- Provide real-time waste level monitoring for each compartment
- Create a user-friendly interface with automatic opening mechanism
- Optimize the classification accuracy using sensor fusion techniques

2. System Architecture

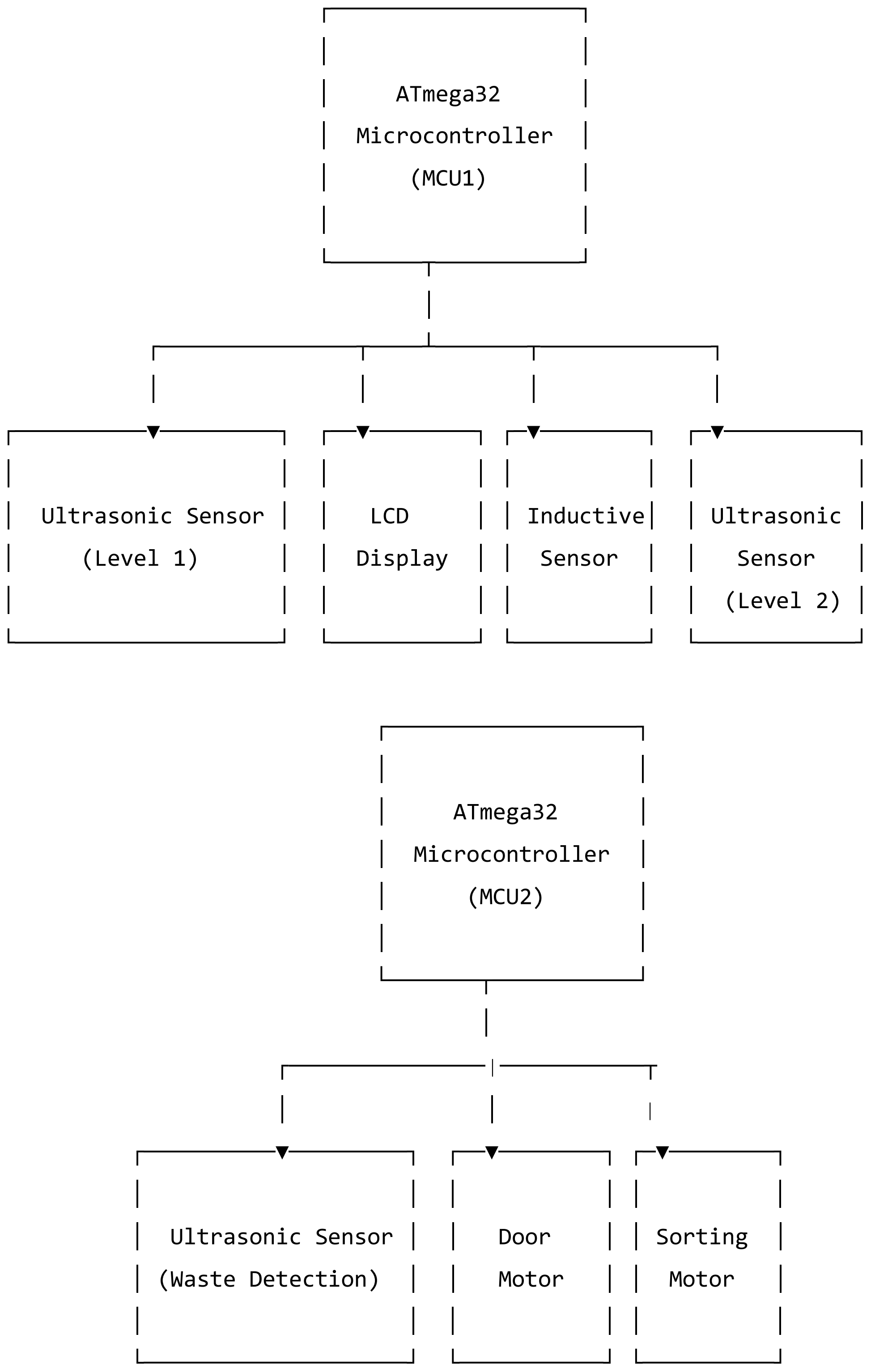
The Smart Pin Trash system employs a modular architecture with multiple microcontrollers working in coordination to achieve the classification objectives. The system is divided into sensing, processing, actuation, and interface modules.

2.1 Hardware Components

The project utilizes the following hardware components:

- 2 × ATmega32 microcontrollers
- 2 × Custom PCBs
- 4 × Ultrasonic sensors (HC-SR04)
- 1 × Inductive proximity sensor
- 1 × LCD display module
- 2 × Servo motors (for door opening and classification mechanism)
- Power supply circuit
- Supporting electronic components (resistors, capacitors, etc.)

2.2 System Block Diagram


ATmega32 Microcontroller (MCU1)
Ultrasonic Sensor (Level 1)
LCD Display
Inductive Sensor
Ultrasonic Sensor (Level 2)
ATmega32 Microcontroller (MCU2)
Ultrasonic Sensor (Waste Detection)
Door Motor
Sorting Motor

3. Working Principle

The Smart Pin Trash system operates according to the following principles:

3.1 User Detection and Door Operation

- The first ultrasonic sensor detects when a user approaches the trash system
- Upon detection, a signal is sent to activate the door motor
- The door automatically opens, allowing the user to dispose of waste
- After a predefined period of inactivity, the door closes automatically

3.2 Waste Classification Mechanism

- When waste is inserted, it passes through a detection zone containing both an ultrasonic sensor and an inductive sensor
- The ultrasonic sensor detects the presence of any object
- The inductive sensor specifically detects metallic objects
- The classification algorithm uses sensor fusion to determine the waste type:
  - If both sensors detect an object simultaneously, the waste is classified as metal
  - If only the ultrasonic sensor detects an object, the waste is classified as non-metal

3.3 Sorting Mechanism Operation

- Based on the classification result, the sorting motor rotates in the appropriate direction:
  - Clockwise rotation (Right) for non-metallic waste
  - Counter-clockwise rotation (Left) for metallic waste
- The waste is directed to the corresponding compartment

3.4 Fill Level Monitoring

- Two ultrasonic sensors continuously monitor the fill level in each compartment
- The current fill levels are displayed on the LCD screen
- This allows for timely emptying of the compartments when they reach capacity

4. Hardware Implementation

4.1 Microcontroller Configuration

The system utilizes two ATmega32 microcontrollers operating at 16MHz, programmed using the Atmel Studio environment with C programming language. The microcontrollers are configured as follows:

MCU1 Configuration:

- Controls the LCD display
- Manages the inductive proximity sensor
- Monitors the fill levels of both compartments via two ultrasonic sensors
- Processes the classification algorithm based on sensor inputs

MCU2 Configuration:

- Handles the waste detection ultrasonic sensor
- Controls both motors:
  - The door opening/closing motor
  - The classification sorting motor
- Receives classification decisions from MCU1

4.2 Sensor Integration

Ultrasonic Sensors (HC-SR04):

The ultrasonic sensors are configured to operate in ranging mode with the following specifications:

- Supply voltage: 5V
- Trigger pulse: 10μs
- Echo pulse: proportional to distance
- Range: 2cm to 30cm
- Resolution: 0.3cm

The time-of-flight principle is used to calculate distance:

```
Distance (cm) = (Echo pulse duration × Speed of sound) / 2
```

Inductive Proximity Sensor:

- Operating voltage: 5V
- Detection range: 4mm (adjustable)
- Output: Digital (High when metal is detected)
- Response time: <10ms

4.3 PCB Design

Two custom PCBs were designed to house the microcontrollers and supporting components:

- PCB1: Contains MCU1 and interface components
- PCB2: Contains MCU2 and sensor signal conditioning circuits

The PCBs were designed with the following considerations:

- Power distribution with proper decoupling
- Signal integrity for sensor inputs
- Motor driver integration
- Inter-board communication interfaces

5. Software Implementation

5.1 Software Architecture

The software is structured in a modular fashion with separate modules for:

- Sensor reading and processing
- Motor control
- User interface
- Classification algorithm
- System coordination

5.2 Classification main Algorithm

The core classification algorithm implements sensor fusion as follows:

```
while (1) {
        distance1 = measure_distance(US1_TRIG_PIN_NUM, US1_ECHO_PIN_NUM);
        _delay_ms(20);
        distance2 = measure_distance(US2_TRIG_PIN_NUM, US2_ECHO_PIN_NUM);
        _delay_ms(20);
        distance3 = measure_distance(US3_TRIG_PIN_NUM, US3_ECHO_PIN_NUM);
        fill1_percentage = calculate_fill_percentage(distance1);
        fill2_percentage = calculate_fill_percentage(distance2);
        lcd_clear();
        display_bin_status(fill1_percentage, fill2_percentage, distance1, distance2);
        if (distance3 < PERSON_DISTANCE_CM) {
            lcd_set_cursor(0, 14);
            lcd_string("P!");
        }
        _delay_ms(MEASUREMENT_DELAY);
    }

    return 0;

}
```

5.3 Level Monitoring

The fill level monitoring system calculates the percentage of filling based on the compartment depth:

```
    distance1 = measure_distance(US1_TRIG_PIN_NUM, US1_ECHO_PIN_NUM);
        _delay_ms(20); // Pequeño retraso entre mediciones (aumentado)
```

```
// Mide distancia con sensor 2 (nivel de basura 2)
distance2 = measure_distance(US2_TRIG_PIN_NUM, US2_ECHO_PIN_NUM);
_delay_ms(20); // Pequeño retraso entre mediciones

// Mide distancia con sensor 3 (detección de persona)
distance3 = measure_distance(US3_TRIG_PIN_NUM, US3_ECHO_PIN_NUM);

// Calcula porcentajes de llenado
fill1_percentage = calculate_fill_percentage(distance1);
fill2_percentage = calculate_fill_percentage(distance2);
```

5.4 User Interface

The LCD interface provides real-time information to users:

- Current fill level of both compartments
- System status indicators
- Error messages when applicable

5.5 Complete Embedded C Code with Explanations

The following code is responsible for controlling the servo motors, detecting the type of waste using an inductive sensor, and measuring distances using two ultrasonic sensors.

```
#include <avr/io.h>

#include <util/delay.h>

#define F_CPU 16000000UL

// Pin Definitions

#define SERVO_PIN PD5        // SG90 Servo connected to PD5

#define METAL_SENSOR_PIN PD2 // Inductive Proximity Sensor
connected to PD2

#define TRIG_PIN PD6         // Ultrasonic 1 TRIG

#define ECHO_PIN PD0         // Ultrasonic 1 ECHO

#define SERVO2_PIN PD3       // MG995 Servo connected to PD3

#define TRIG2_PIN PC0        // Ultrasonic 2 TRIG (A0)

#define ECHO2_PIN PC1        // Ultrasonic 2 ECHO (A1)
```

- **Servo Control Functions**

These functions generate the PWM signal required to control standard hobby servos.

```
// Move SG90 servo to right
void servo_right() {
        for (int i = 0; i < 50; i++) {
                PORTD |= (1 << SERVO_PIN);
                _delay_us(2300);
                PORTD &= ~(1 << SERVO_PIN);
                _delay_ms(17.7);
        }
}

// Move SG90 servo to left
void servo_left() {
        for (int i = 0; i < 50; i++) {
                PORTD |= (1 << SERVO_PIN);
                _delay_us(700);
                PORTD &= ~(1 << SERVO_PIN);
                _delay_ms(19.3);
        }
}

// Center SG90 servo
void servo_center() {
        for (int i = 0; i < 50; i++) {
                PORTD |= (1 << SERVO_PIN);
                _delay_us(1500);
                PORTD &= ~(1 << SERVO_PIN);
                _delay_ms(18.5);
        }
}
```

- **Ultrasonic Sensor Measurement Function**

```
uint16_t measure_distance() {
 PORTD |= (1 << TRIG_PIN);
 _delay_us(10);
 PORTD &= ~(1 << TRIG_PIN);

 while (!(PIND & (1 << ECHO_PIN))) {}
 uint16_t start_time = 0;
 while (PIND & (1 << ECHO_PIN)) {
       start_time++;
       _delay_us(1);
 }
 uint16_t distance = start_time * 0.0343 / 2;
 return distance; }
```

- **Servo Motor 2:**

```
void servo2_right() {
 for (int i = 0; i < 50; i++) {
       PORTD |= (1 << SERVO2_PIN);
       _delay_us(2400);
       PORTD &= ~(1 << SERVO2_PIN);
       _delay_ms(17.6);
 }
}

void servo2_left() {
 for (int i = 0; i < 50; i++) {
       PORTD |= (1 << SERVO2_PIN);
       _delay_us(600);
       PORTD &= ~(1 << SERVO2_PIN);
       _delay_ms(19.4);
 }
}

void servo2_center() {
 for (int i = 0; i < 50; i++) {
       PORTD |= (1 << SERVO2_PIN);
       _delay_us(1500);
       PORTD &= ~(1 << SERVO2_PIN);
       _delay_ms(18.5);
 }
}
```

- **Second Ultrasonic Sensor Measurement**

```
uint16_t measure_distance2() {
        PORTC |= (1 << TRIG2_PIN);
        _delay_us(10);
        PORTC &= ~(1 << TRIG2_PIN);
        while (!(PINC & (1 << ECHO2_PIN))) {}
        uint16_t duration = 0;
        while (PINC & (1 << ECHO2_PIN)) {
                duration++;
                _delay_us(1);
        }

        uint16_t distance = duration * 0.0343 / 2;
        return distance;
}
```

- **Main Function**

```
int main(void) {
        // Set pin directions
        DDRD |= (1 << SERVO_PIN) | (1 << SERVO2_PIN) | (1 << TRIG_PIN);
        DDRD &= ~(1 << ECHO_PIN) & ~(1 << METAL_SENSOR_PIN);
        PORTD |= (1 << METAL_SENSOR_PIN);  // Enable pull-up

        DDRC |= (1 << TRIG2_PIN);
        DDRC &= ~(1 << ECHO2_PIN);

        while (1) {
                // First Ultrasonic + Metal sensor classification
                uint16_t distance1 = measure_distance();
                if (distance1 < 3) {
                        if (PIND & (1 << METAL_SENSOR_PIN)) {
                                servo_right();      // Metal
                        } else {
                                servo_left();       // Non-metal
                        }
                        servo_center();
                        _delay_ms(500);
                }

                // Second Ultrasonic for approach detection
                uint16_t distance2 = measure_distance2();
                if (distance2 < 5) {
                        servo2_right();        // User detected, open lid
                        _delay_ms(100);
                } else {
                        servo2_center();       // No user, close lid
                        _delay_ms(250);
                }
        }
        return 0;
}
```

- ## Second ATMEGA32 :

```
/*
 * main.h
 *
 * Created: 4/30/2025 8:50:42 PM
 * Author: huawei
 */

#ifndef MAIN_H_
#define MAIN_H_

#include <avr/io.h>
#include <util/delay.h>
#include <stdint.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

// ==============================
//  Microcontroller Settings
// ==============================
#define F_CPU 16000000UL  // Define CPU frequency (16 MHz for ATmega32)

// ==============================
// Trash Level Thresholds
// ==============================
#define LEVEL_LOW_THRESHOLD   30    // Below 30% → low level
#define LEVEL_MID_THRESHOLD   70    // 30–70% → mid, above 70% → high

// ==============================
//  LCD Pin Configuration
// ==============================
#define LCD_RS      PD0
#define LCD_EN      PD1
#define LCD_D4      PD2
#define LCD_D5      PD3
#define LCD_D6      PD4
#define LCD_D7      PD5
#define LCD_PORT    PORTD
#define LCD_DDR     DDRD
#define LCD_PIN     PIND

// LCD Commands for 4-bit Mode
#define LCD_CLEAR           0x01
#define LCD_HOME            0x02
#define LCD_ENTRY_MODE      0x06  // Increment cursor
#define LCD_DISPLAY_ON      0x0C  // Display ON, cursor OFF
#define LCD_FUNCTION_4BIT   0x28  // 4-bit interface, 2-line mode
#define LCD_FUNCTION_RESET  0x30
#define LCD_SET_CGRAM       0x40
#define LCD_SET_DDRAM       0x80
```

```
// =============================
//  ADC Configuration (Unused Pot 2)
// =============================
#define POT1_CHANNEL 1
#define POT2_CHANNEL 0

// =============================
//  Ultrasonic Sensor Pin Assignments
// =============================
#define US_DDR      DDRA
#define US_PORT     PORTA
#define US_PIN      PINA

#define US1_TRIG_PIN PA3
#define US1_ECHO_PIN PA0
#define US2_TRIG_PIN PA4
#define US2_ECHO_PIN PA2
#define US3_TRIG_PIN PA5  // Person detection
#define US3_ECHO_PIN PA6

// Pin number macros for generic functions
#define US1_TRIG_PIN_NUM    3
#define US1_ECHO_PIN_NUM    0
#define US2_TRIG_PIN_NUM    4
#define US2_ECHO_PIN_NUM    2
#define US3_TRIG_PIN_NUM    5
#define US3_ECHO_PIN_NUM    6

// =============================
//  Servo Motor Configuration
// =============================
#define SERVO_DDR    DDRD
#define SERVO_PORT   PORTD
#define SERVO_PIN    PD7
#define SERVO_OPEN   2500   // µs pulse width to open
#define SERVO_CLOSED 1250   // µs pulse width to close

// =============================
//  System Operation Parameters
// =============================
#define BIN_DEPTH_CM        50    // Total depth of bin (cm)
#define MIN_DISTANCE_CM     5     // Minimum distance (interpreted as full)
#define PERSON_DISTANCE_CM  30    // Trigger distance for person detection
#define MAX_SENSOR_TIMEOUT  50000 // Timeout to avoid sensor lock
#define MEASUREMENT_DELAY   500   // Delay between measurements (ms)
#define ULTRASONIC_TIMEOUT  20    // Timeout for ultrasonic signal (ms)
#define SERVO_OPEN_TIME     3000  // Lid open duration (ms)

// =============================
//  LCD Function Prototypes
// =============================
void lcd_init(void);
void lcd_send_command(uint8_t cmd);
void lcd_send_data(uint8_t data);
void lcd_send_byte(uint8_t byte);
void lcd_send_nibble(uint8_t nibble);
void lcd_clear(void);
void lcd_home(void);
void lcd_set_cursor(uint8_t row, uint8_t col);
void lcd_string(const char *str);
void lcd_create_char(uint8_t location, const uint8_t *charmap);

// ==============================
// ADC Function Prototypes
// ==============================
void adc_init(void);
uint16_t adc_read(uint8_t channel);

// ==============================
//  Ultrasonic Sensor Prototypes
// ==============================
void ultrasonic_init(void);
uint16_t measure_distance(uint8_t trig_pin, uint8_t echo_pin);
void trig_pulse(uint8_t trig_pin);

// ==============================
//  Servo Motor Function Prototypes
// ==============================
void servo_init(void);
void servo_position(uint16_t position);
void servo_open(void);
void servo_close(void);

// ==============================
//  Utility Function Prototypes
// ==============================
void system_init(void);
uint8_t calculate_fill_percentage(uint16_t distance);
void display_bin_status(uint8_t fill1, uint8_t fill2, uint16_t dist1, uint16_t
dist2);
void handle_error(uint8_t error_code);

#endif /* MAIN_H_ */
```

Main System Implementation (LCD_2_ULTRASONIC.c)

This file contains the core logic of the Smart Pin Trash system. It integrates multiple modules such as the LCD interface, ultrasonic distance sensors, servo motors for lid actuation, and bin fill level calculation. The system is implemented in Embedded C and compiled using Atmel Studio for the ATmega32 microcontroller.

Global Variables and Enums

```
volatile uint16_t timer_overflow_count = 0;
volatile uint8_t servo_state = 0;
volatile uint32_t servo_open_timestamp = 0;
```

```
enum ErrorCodes {
    ERR_NONE = 0,
    ERR_US_TIMEOUT = 1,
    ERR_LCD_INIT = 2,
    ERR_ADC_READ = 3
};
```

- `timer_overflow_count`: Counts timer overflows to measure pulse duration.
- `servo_state`: Tracks whether the lid is open or closed.
- `ErrorCodes`: Helps handle various system errors (timeout, LCD failure, etc.).

LCD Display Functions (4-bit Mode)

These functions handle initialization and interaction with the 16x2 LCD screen.

```
void lcd_send_nibble(uint8_t nibble);
void lcd_send_byte(uint8_t byte);
void lcd_send_command(uint8_t cmd);
void lcd_send_data(uint8_t data);
void lcd_init(void);
void lcd_clear(void);
void lcd_home(void);
void lcd_set_cursor(uint8_t row, uint8_t col);
void lcd_string(const char *str);
void lcd_create_char(uint8_t location, const uint8_t *charmap);
```

Explanation:

- Initializes the LCD in 4-bit mode.
- Displays strings, numbers, and system statuses.
- Supports custom character creation and cursor movement.

ADC Functions

```
void adc_init(void);
uint16_t adc_read(uint8_t channel);
```

Explanation:

- Initializes the ADC module.
- Reads analog input values. (Currently used for potential future expansion.)

Servo Motor Control Functions

```
void servo_init(void);
void servo_position(uint16_t position);
void servo_open(void);
void servo_close(void);
```

Explanation:

- Controls the servo motor to open/close the bin lid.
- Uses PWM to position the servo at specific angles.

Ultrasonic Sensor Functions

```
void ultrasonic_init(void);
void trig_pulse(uint8_t trig_pin);
uint16_t measure_distance(uint8_t trig_pin, uint8_t echo_pin);
```

Explanation:

- Sends a trigger pulse.
- Measures distance by calculating the echo pulse duration.
- Detects waste level in bins and user proximity.

Utility Functions

```
void system_init(void);
uint8_t calculate_fill_percentage(uint16_t distance);
void display_bin_status(uint8_t fill1, uint8_t fill2, uint16_t dist1,
uint16_t dist2);
void handle_error(uint8_t error_code);
```

Explanation:

- Initializes all modules (LCD, ADC, Servo, Ultrasonic).
- Converts distance measurements into fill percentages.
- Displays bin status (percentage and classification: low, mid, high).
- Handles system errors (e.g., timeout).

Fill Level Status Logic

```
const char* get_level_text(uint8_t fill_percentage) {
    if (fill_percentage < LEVEL_LOW_THRESHOLD) return "low ";
    else if (fill_percentage < LEVEL_MID_THRESHOLD) return "mid ";
    else return "high";
}
```

Explanation:

- Translates numerical fill levels into readable text for display.

Main Function Logic

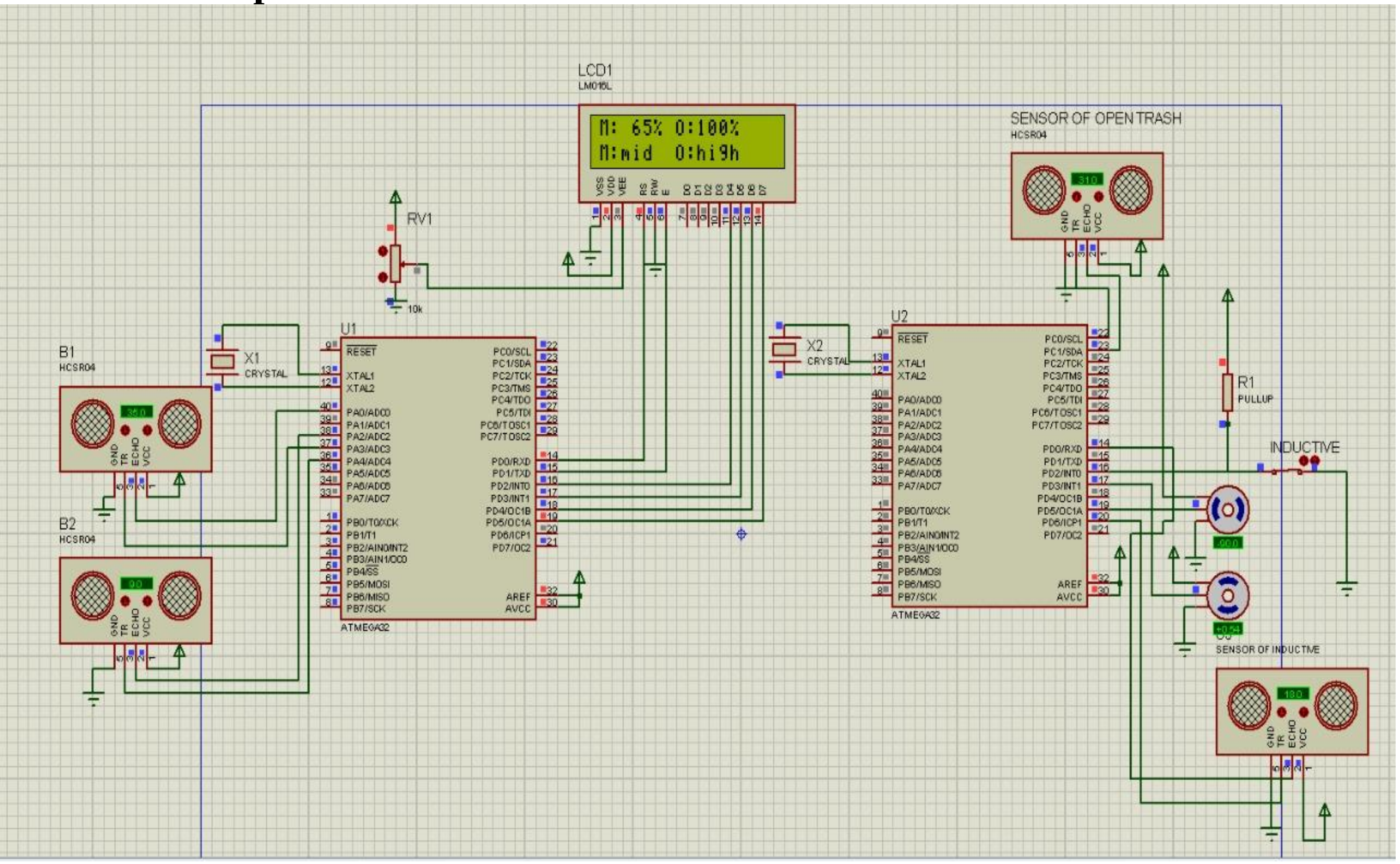
```
int main(void) {
    uint16_t distance1 = 0, distance2 = 0, distance3 = 0;
    uint8_t fill1_percentage = 0, fill2_percentage = 0;
    uint8_t error_status = ERR_NONE;
    uint16_t servo_timer = 0;

    system_init();

    lcd_clear();
    lcd_string("Smart Trash Bin");
    lcd_set_cursor(1, 0);
    lcd_string("Initializing...");
    _delay_ms(1000);

    while (1) {
        // Measure distances from ultrasonic sensors
        distance1 = measure_distance(US1_TRIG_PIN_NUM, US1_ECHO_PIN_NUM);
        _delay_ms(20);
        distance2 = measure_distance(US2_TRIG_PIN_NUM, US2_ECHO_PIN_NUM);
        _delay_ms(20);
        distance3 = measure_distance(US3_TRIG_PIN_NUM, US3_ECHO_PIN_NUM);

        // Calculate fill percentages
        fill1_percentage = calculate_fill_percentage(distance1);
        fill2_percentage = calculate_fill_percentage(distance2);

        // Display fill levels
        lcd_clear();
        display_bin_status(fill1_percentage, fill2_percentage, distance1,
distance2);

        // Detect person presence and operate lid
        if (distance3 < PERSON_DISTANCE_CM) {
            if (servo_state == 0) {
                servo_open();
                lcd_set_cursor(0, 14);
                lcd_string("OP");
            }
            servo_timer = 0;
        } else {
            if (servo_state == 1) {
                servo_timer++;
                if (servo_timer >= (SERVO_OPEN_TIME / MEASUREMENT_DELAY)) {
                    servo_close();
                    lcd_set_cursor(0, 14);
                    lcd_string("CL");
                }
            }
        }

        _delay_ms(MEASUREMENT_DELAY);
    }

    return 0;
}
```

Explanation:

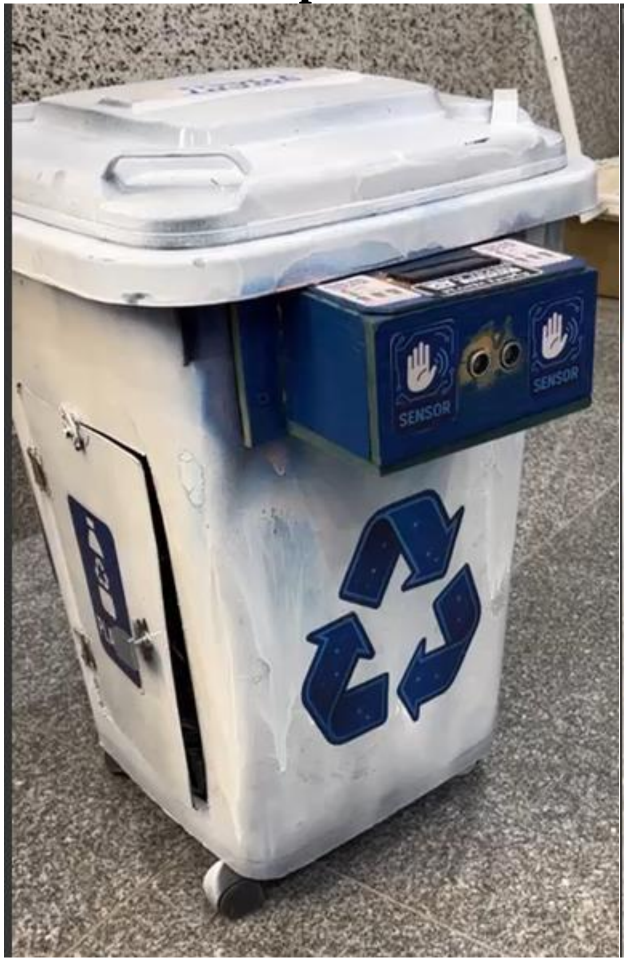
- Continuously measures bin levels and detects user presence.

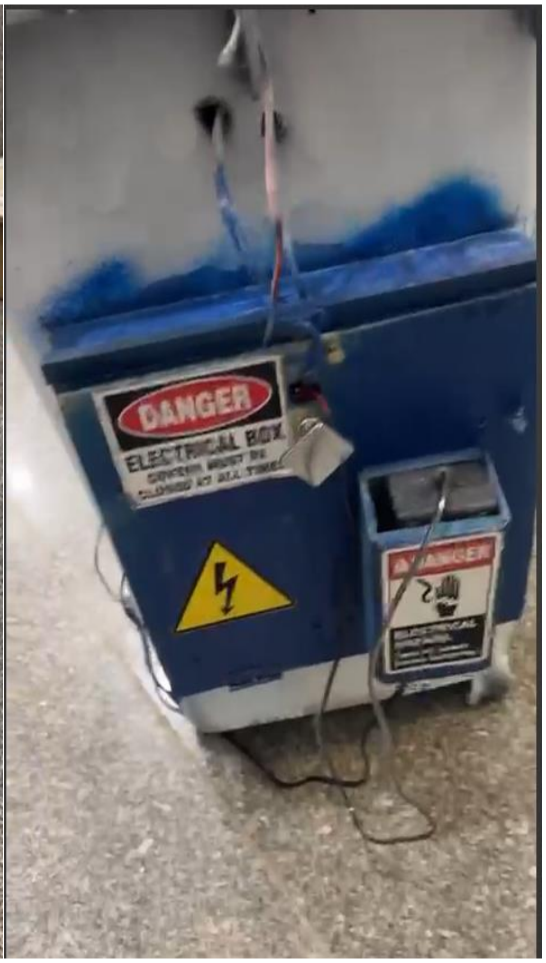
- Opens the lid if a user is near.
- Displays real-time information on the LCD screen.
- Closes the lid after a timeout if no user is detected.

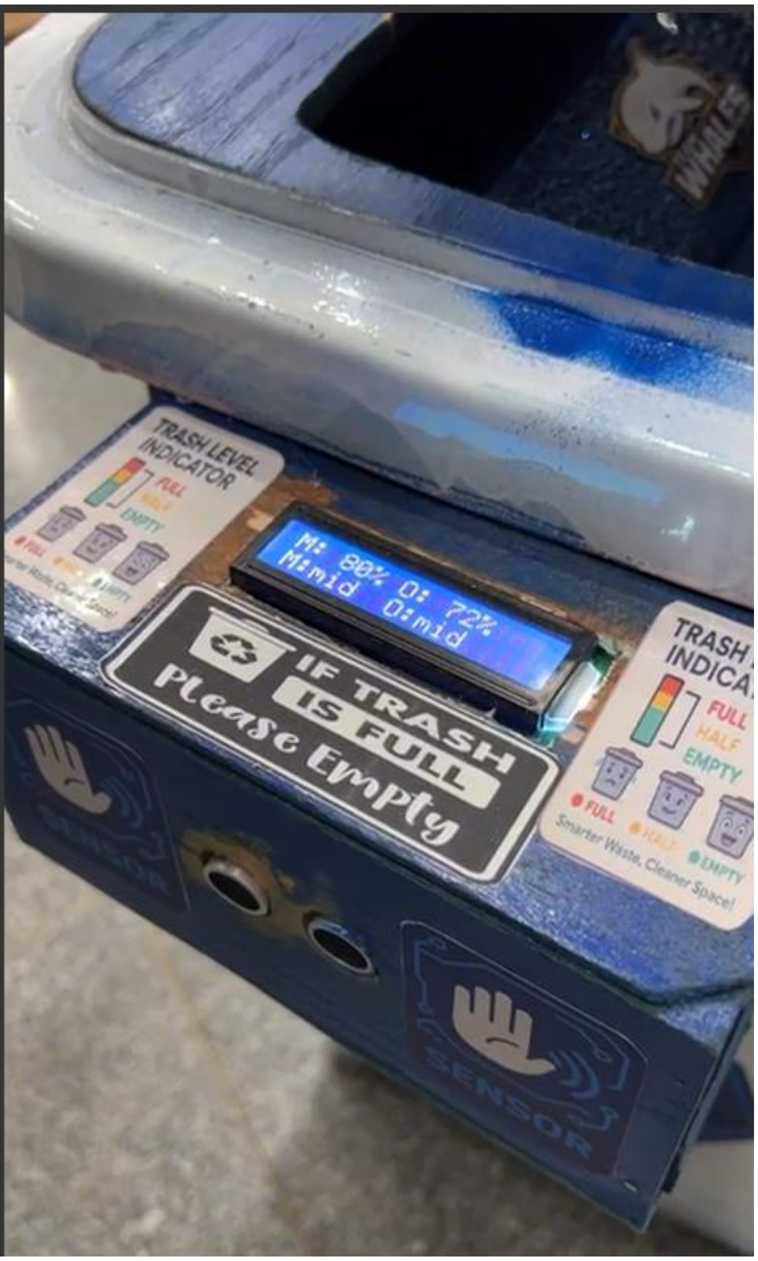
5.5 Schematic proteus Circuit:



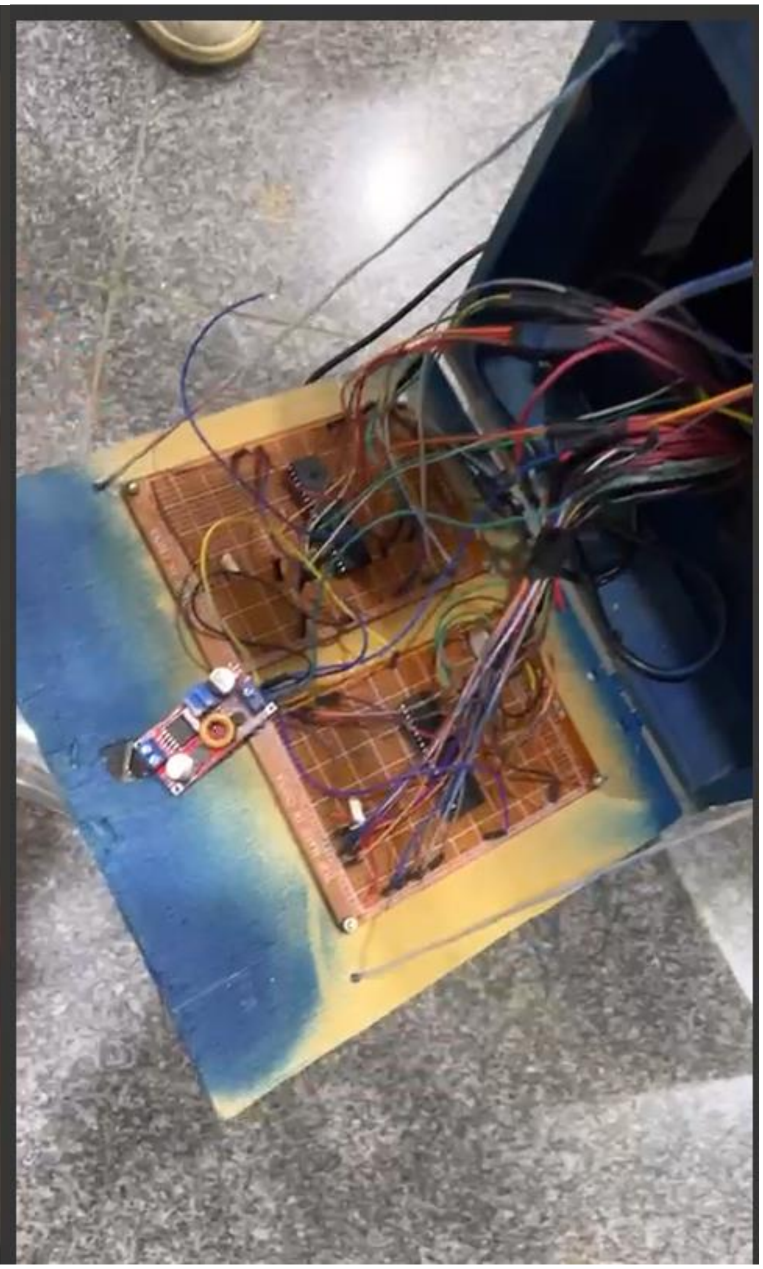
5.6 Hardware pictures:







```
M: 80% O: 72%
M:mid  O:mid
```

WHALES
WHALES


6. Future Improvements

Potential enhancements for future iterations include:

- Integration of additional sensor types for more precise classification
- Implementing machine learning algorithms for improved accuracy
- Adding wireless connectivity for remote monitoring
- Incorporating solar power for energy autonomy
- Expanding classification categories (plastic, paper, glass, etc.)

7. Conclusion

The Smart Pin Trash system successfully demonstrates the application of embedded systems concepts to create an effective waste classification solution. By combining multiple sensors and microcontrollers, the project achieves reliable waste sorting with minimal user intervention. The system's modular design allows for future enhancements and adaptations to different waste management scenarios.

Through this project, we have developed practical skills in sensor integration, embedded programming, PCB design, and system integration while addressing a significant environmental challenge. The Smart Pin Trash system represents a viable approach to improving waste sorting efficiency at the source.