Texas Tech University

CS-5331-003

Special Problems in Computer Science:

Embedded Systems

A Report on "**CRAP BAG**"

Cyber-Physical System

Submitted by:

Group Number 28

Group Members:

Raj Dhaubanjar (R11854011)
Simran Thapa (R11823462)
Sumit Shrestha (R11836842)

Date of Submission:

30th April, 2024

Introduction

In the modern era of rapid urbanization, effective waste management has become a global issue. As science and technology has been emerging at a rapid rate, traditional methods of waste management have been turning into adequate leading to overflow of trash bins, pollution, and compromised public hygiene. The conventional methodologies of waste collection and management typically require rudimentary infrastructure, manual operation and hence are not well suited for the global world. To overcome these problems, we came up with a transformative solution of creating a smart trash bin, that solves the ineffective waste management in the global world.

Problem Statement

The current situation of garbage management in this global civilization is too traditional and with inefficiencies and flaws that jeopardize the public health and the environment as well. Traditional waste collection techniques use manual interventions like opening and closing the lid of the dustbin, timely checking if the trash bin is full or not as traditional method of garbage collection do not offer real time visibility of bin fill levels. This allows the public to keep filling the trashcans even after they are full, making them spill on the ground making the surroundings more polluted. Moreover, manual closing and opening of the lids with hand makes using the bin more unhygienic as the transfer of the germs from the bin to hands is unavoidable, making public health riskier to maintain.

Let us take a simple scenario as an example of every household. We have a regular dustbin in our home. Sometimes we are too lazy to empty the trash bin even after the bin is full and we keep dumping more and more into the bin until the bin overflows leading to spilling of the waste in the ground and making it more polluted. We also use our hand to open the lid of the bin, this may make the germs spread into our hands putting our health at risk of spread of various fatal diseases compromising our health and hygiene. The other issue with waste management in dumpsters and large garbage disposal areas is not knowing the level of waste inside the garbage bins. So, the garbage is always filled over its maximum capacity leading to polluted surrounding and dirty.

There has always been an issue with waste management with the growing population making the traditional method of garbage disposal unsuited in the modern world of science of technology where every step of human life is dependent on technology. Every action in human life is dependent on one touch either with smart phones or any tech devices. Hence it is important to upgrade waste management process as well with the use of emerging technologies making human life simpler and convenient and at the same time encouraging sustainable and clean environment.

Objective

To solve the issue of inefficient waste management, unhygienic approach of dealing waste, we came up with the solution of a smart trash bin named as Crap Bag that uses cutting edge technology and fits well with the growing world of technology and civilization. The aspiration is to develop and deploy a smart thrash bin system that revolutionizes waste management practices in the modern world.

The major objectives include:

Develop a smart bin

The main objective of the project is to design and implement smart bin equipped with Arduino, sensor and actuators that detect any human nearby it, opening the lid in human presence, continuous monitoring of the waste levels and giving a warning message when the bin has reached its maximum capacity.

Real-Time Monitoring and Analysis:

Using sensors and Arduino libraries, we have aimed to provide continuous monitoring of the waste level, the sensor inside the bin keeps checking the level of waste filled and displays the result in the LED.

This process ensures the user is aware of the space available for more waste to be dumped into the bin. Moreover, when the sensors detect that the waste level has reached its maximum capacity and display a warning message as Bin s full of a red led turned on in the hardware and preventing the door lid of the bin from opening. Hence, the system prevents any overflow of the trash making the surroundings more hygienic and cleaner.

Enhancement of Public Health and Hygiene

Our project prioritizes public health and hygiene, making sure that the bin opens automatically for detection of any human in front of the bin without any physical touch. Since no physical connection is used to open the lid, the spreading of germs from the bin to humans is minimized, proving a safe and healthy method of waste collection.

Promotion of Environmental Sustainability

By mitigating the problem of overflow of bins and hence keeping the surrounding clean, the project promotes responsible waste disposal behaviors and leads us one step closer to a sustainable environment. As the sensor implemented inside of the bin continually monitors the waste level. It prevents the overflowing of bins as it won't let the lid of the bin open with warning message displayed int he screens.

Our project aims to provide a modern method of waste management solving the common problem of every household of waste management using Arduino, sensors, and actuators, proving real time monitoring and data driven decision making the waste management practices more environment friendly and hygienic uplifting the living style in every household.

Hardware Used

We used different types of hardware to create the prototype of our project. Below are the lists of hardware we used:

1. Arduino UNO x 1
2. Servo Motor x 1
3. LCD x 1
4. Ultrasound Sensor x 2
5. Potentiometer x 1
6. 10mm LED x 1
7. Jumper Cables x 25
8. Resistors x 1

Procedure

Simulation

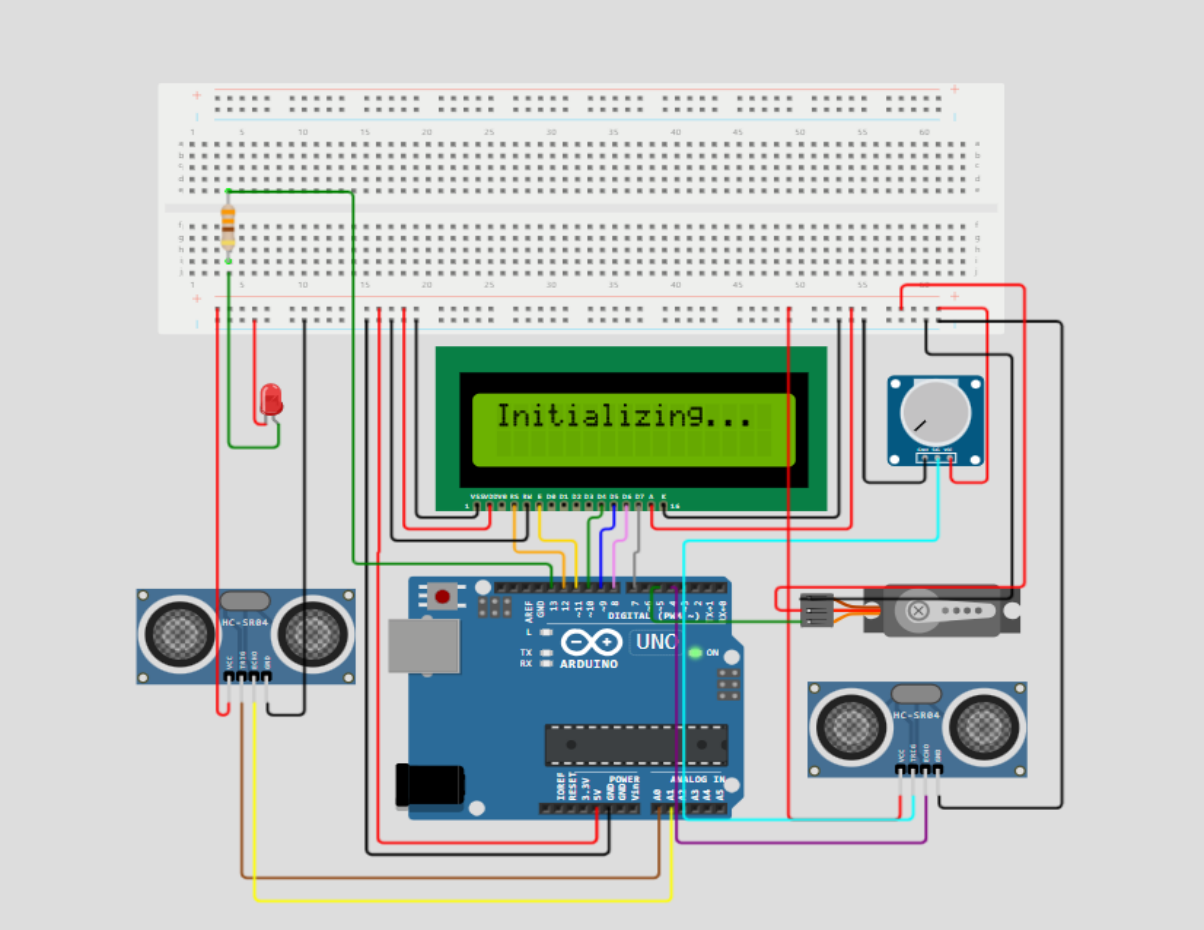

Figure 1: Simulating hardware connections

Hardware Description

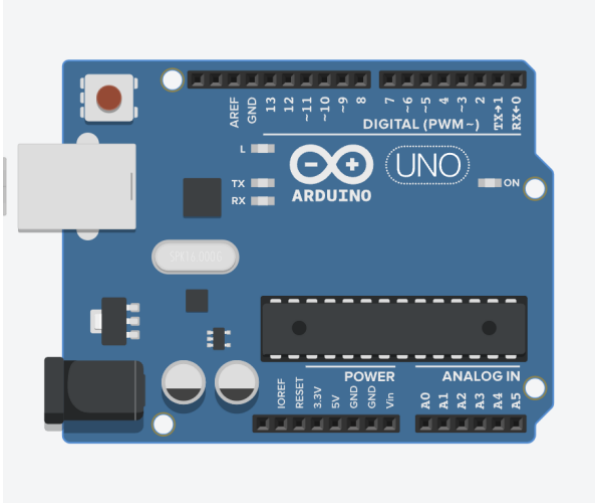

Figure 2: Arduino Uno

For the system to be a CPS, it must have sensors that detect and collect the data from the physical environment, an actuator to effect changes in that environment, and embedded computational units to process information and make decisions, enabling real-time interaction and feedback between the two domains. We created a smart system that responds to its surroundings and provides feedback accordingly. As a central processing unit of the project, we used the Arduino UNO, to facilitate the control and coordination of various hardware we used. All the functionalities of the hardware run through this microcontroller. For the sensor to collect the data from the physical environment, we used the two ultrasound sensors. And as the actuator, to respond according to the data from the sensors, we used the servo motor.

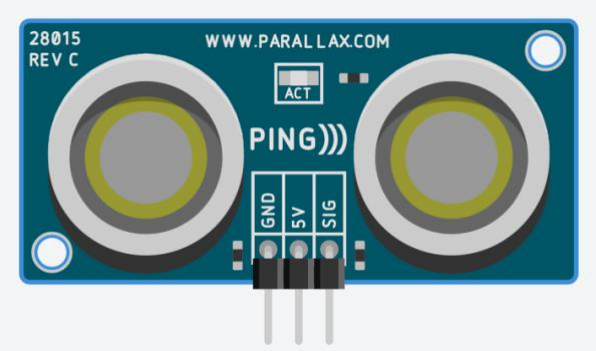

Figure 3: Ultrasound Sensor

First, we attached first ultrasound sensor outside of the trash can so that it can detect the outer environment's data i.e., the movement of the user. With this sensor the system can recognize if someone is close to the trash can. We hardcoded this sensor with the distance of 30cm. That means, the sensor will detect any movement in this range and passes the data to the Arduino UNO. We setup the second ultrasound inside the trash can on the top side. For this sensor also we hardcoded the distance of 24cm and delay of 500ms so that it will continuously check the distance. This sensor will also collect the trash level inside the can and passes the data to the Arduino UNO.

Figure 4: Servo Motor

Second, after the input data from the sensors and decisions made by the Arduino, to stimulate the decision of opening the lid of the trash can to allow more trash inside, we used the servo motor as an actuator. It helps in opening and closing the lid of the system. The servo motor only opens the lid only if the trash can is not full. Else the servo motor won't open the lid to protect the overflowing of the trash.

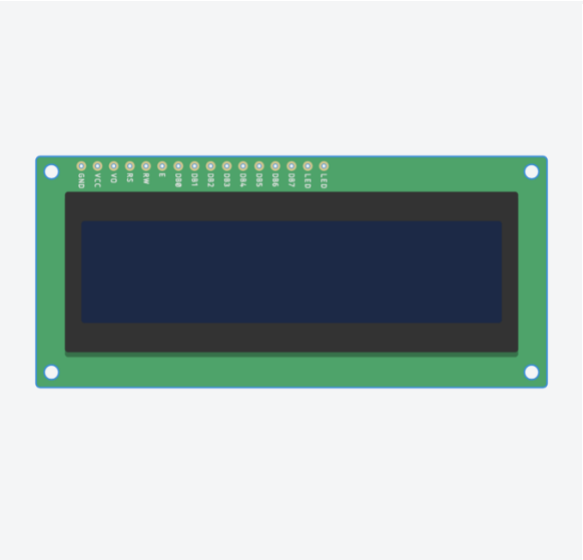

Figure 5: LED

Figure 6: LCD

Third, to make the system not only the CPS but also smart and user-friendly, we added the LCD. This will help the user to view the trash level information. By displaying the real-time data from sensors, the LCD provides immediate feedback to users about the state of the system, which will prevent the user from manually opening the lid to check level of waste inside. We used the potentiometer to control the brightness of the LCD.

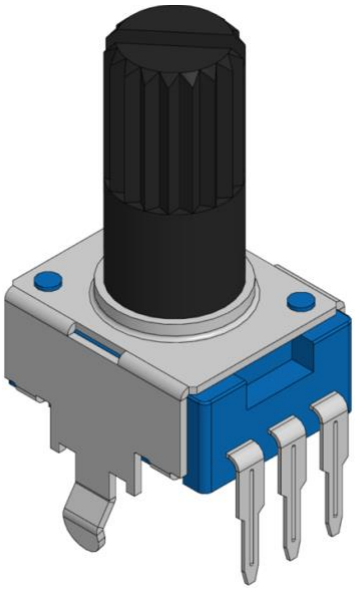

Figure 7: Potentiometer

Figure 8: Jumper Cable

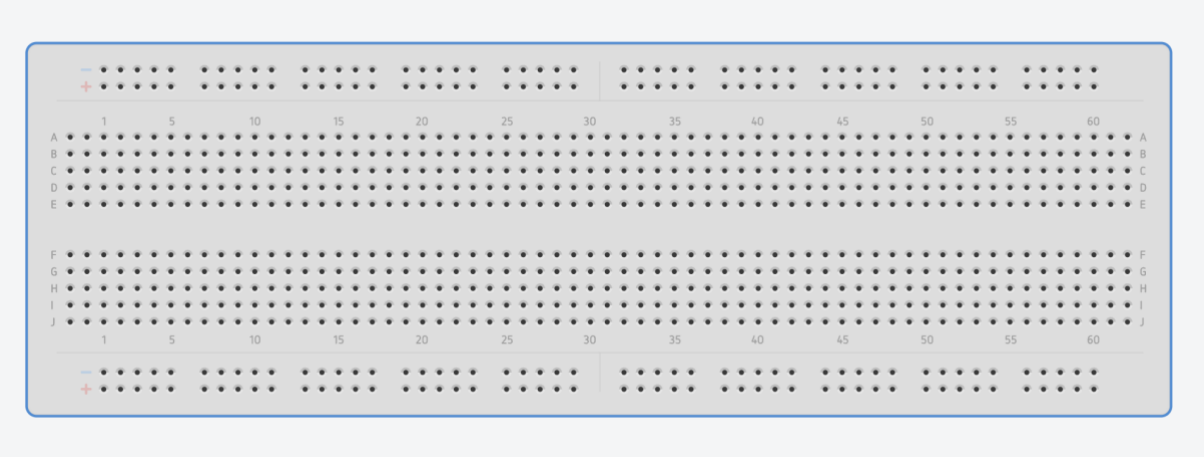

Figure 9: Breadboard

And as to warn the user of the trash can getting full and change the trash bag, we have added a 10mm LED light in front face of the can. The system turns on the LED when the trash is full and turns it off when the trash is not full. To protect the LED against voltage spike, we used the 220-ohm resistor. Lastly, we used the jumper cables and breadboard for the connections between the sensors, actuators, and Arduino UNO.

Process

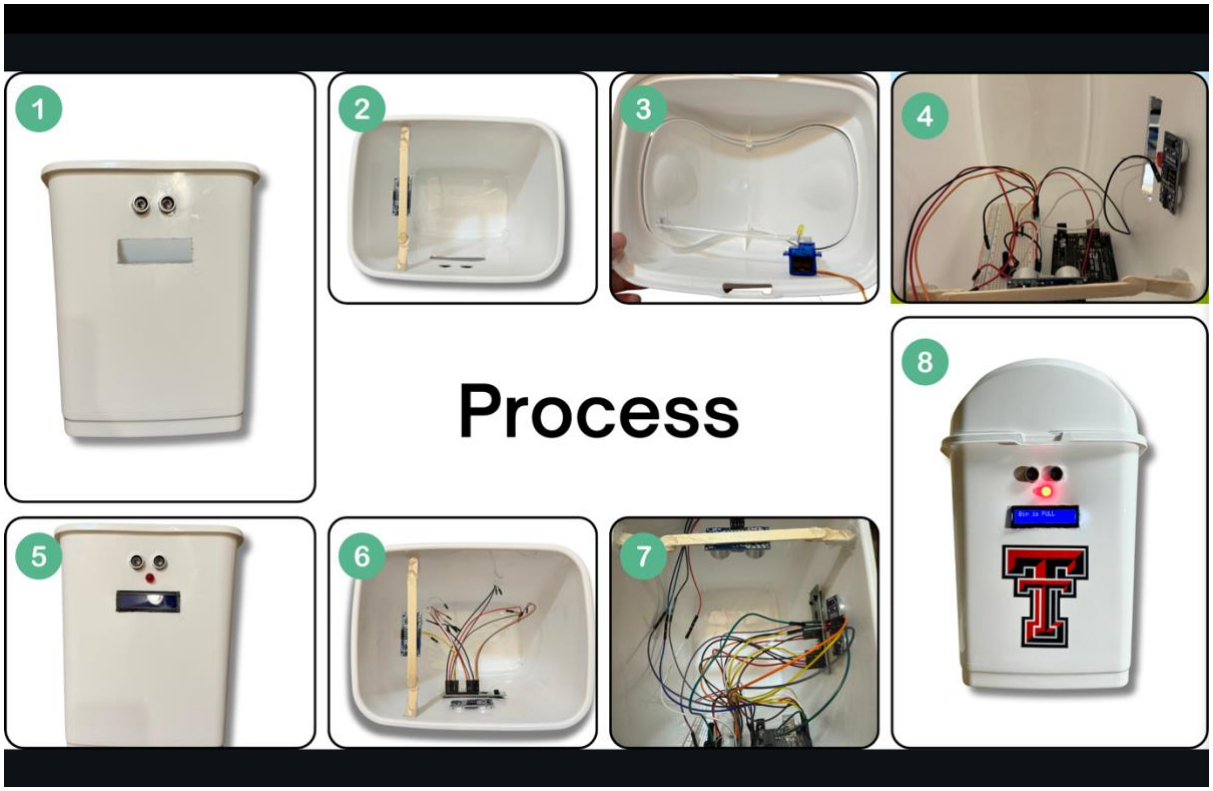

Figure 10: Stepwise hardware implementation

Flowchart

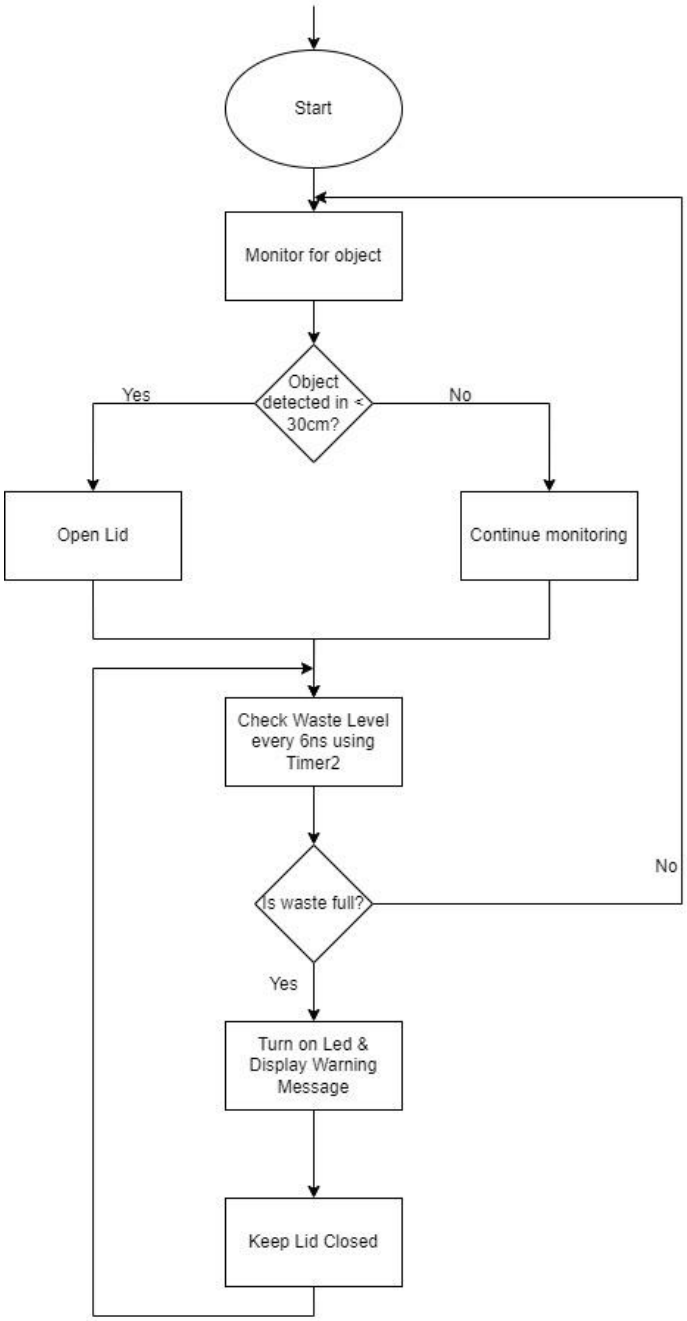

Figure 11: Flowchart of Crap Bag

State Diagram

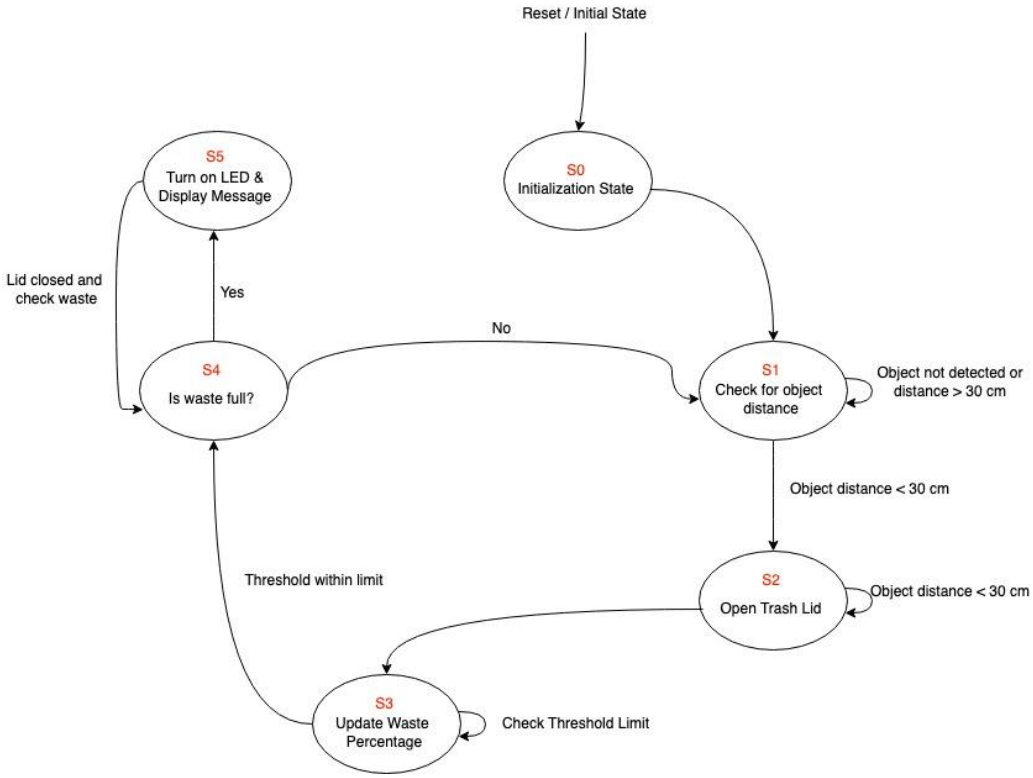

Figure 12: State Diagram of Crap Bag

Code Description

Code Partition

```
#include <Servo.h>          // Include Servo library for controlling servo motor
#include <LiquidCrystal.h>  // Include LiquidCrystal library for LCD display
#include <NewPing.h>        // Include NewPing library for ultrasonic sensor

// Constants declaration
const int maxWastePercentage = 96;  // Maximum waste percentage before indicating bin is full
const int trigPin = 3;              // Pin for triggering ultrasonic sensor
const int echoPin = 4;              // Pin for receiving echo from ultrasonic sensor
const int servoPin = 5;             // Pin for controlling servo motor

// Variables declaration
int wastePercentage;                // Current waste percentage
float durationWaste, distanceWaste; // Duration and distance variables for waste level
float maxLevel = 23.0;              // Maximum level of waste before indicating bin is full

// LCD pins and maximum distance for ultrasonic sensor
#define wasteTrig A0      // Ultrasonic sensor trigger pin
#define wasteEcho A1      // Ultrasonic sensor echo pin
#define MAX_DISTANCE 200  // Maximum distance for ultrasonic sensor in centimeters

NewPing sonar(wasteTrig, wasteEcho, MAX_DISTANCE);  // NewPing object for ultrasonic sensor
Servo servo;                                        // Servo motor object
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);             // LCD object
```

The above code shows the initialization of the variables, libraries and the objects that we used while implementing the crap bag.The of libraries expands the functionality and capabilities of our Arduino-based smart trash bin system. We utilize the Servo library, accessed through <Servo.h>, to control servo motors, which are crucial for opening and closing the can's door. Similarly, with the <LiquidCrystal.h> library, we can interact with character-based LCD displays, we used this for displaying text and symbols on the LCD screen, allowing us to show the waste percentage and if the can is full. Also, we used <NewPing.h> library for simplifying the use of ultrasonic sensors for distance measurement. With functions like ping() and convert_cm(), this library streamlines the process of sending ultrasonic pulses, measuring echo return times, and calculating distances accurately, enabling us to assess the level of waste in the bin precisely.

Constants and Variables

We define the maximum waste percentage constant maxWastePercentage to be 96% before indicating the bin is full. Also, trigPin and echoPin are the pins we use for triggering and receiving signals from the ultrasonic sensor. Meanwhile servoPin is used for controlling the servo motor. Also, the wastePercentage holds the current waste percentage whereas durationWaste and distanceWaste are for measuring duration and distance related to waste level. The max level maxLevel is set to 23.0 since the size of the crap bag is 23.0 cm which helps to calculate the waste percentage. Also, we define the pin for detecting waste percentage which has wasteTrig as trigger pin and wasteEcho as echo pin. The maximum distance for the ultrasonic is set to 200cm.

Object Instantiation

We instantiate a NewPing object as sonar with trigger and echo pins for the ultrasonic sensor and maximum distance. The servo object is for controlling the servo motor. The lcd object is used to configure pin for the LCD display.

```
void setup() {
  lcd.begin(16, 2);               // Initialize LCD with 16 columns and 2 rows
  lcd.print("...Welcome to...");  // Display welcome message
  lcd.setCursor(0, 1);            // Move cursor to the second row
  lcd.print("    Crap Bag    ");  // Print device name
  Serial.begin(9600);             // Initialize serial communication
  servo.attach(servoPin);         // Attach servo motor to its pin
  DDRB |= B100000;                // Set LED pin as output
  pinMode(trigPin, OUTPUT);       // Set trigger pin of ultrasonic sensor as output
  pinMode(echoPin, INPUT);        // Set echo pin of ultrasonic sensor as input
  servo.write(0);                 // Set servo motor to initial position (0 degrees)
  servo.detach();                 // Detach the servo after setting initial position
  cli();                          // Disable global interrupt

  // Configure Timer2
  TCNT2 = 0;           // Reset Timer2 counter
  TCCR2A = B00000000;  // Set Timer2 control register A to 0
  TCCR2B = B00000000;  // Set Timer2 control register B to 0

  OCR2A = 252;          // Set compare match register for 16ns
  TCCR2A |= B00000100;  // Set Timer2 to CTC mode
  TCCR2B |= B00000111;  // Set prescaler of 1024
  TIMSK2 |= B00000010;  // Enable Timer2 compare match interrupt
  sei();                // Enable global interrupt
}
```

The above code shows the initial setup of the code that we do while initializing the project. It initializes the various components and configurations required for the proper functioning of the crap bag.

LCD Initialization

lcd.begin(16, 2), it initializes the LCD display with 16 columns and 2 rows. lcd.print("...Welcome to..."), it prints the welcome message on the LCD whereas lcd.setCursor(0, 1) moves the cursor to the beginning of the second row.

Serial Communication Initialization

We used Serial.begin(9600) to initializes serial communication with a baud rate of 9600 which allows communication between the Arduino board and a connected computer for debugging purposes. We used this serial communication to debug different features and timer.

Servo Motor Initialization

servo.attach(servoPin), it attaches the servo motor to the specified pin (servoPin) which prepares the servo motor for control whereas servo.write(0) sets the initial position of the servo motor to 0 degrees, effectively closing the bin's door and the servo.detach() detaches the servo motor after setting its initial position. This reduces power consumption and prevents interference with other components.

LED Pin Configuration

We have used the LED to determine if the trash is full or not. So we initialized it by setting DDRB |= B100000 which sets the specified pin (LED pin) as an output pin. This prepares the pin to drive an LED indicator.

Ultrasonic Sensor Pin Configuration

We initialize two pins of the ultrasonic sensor pinMode(trigPin, OUTPUT) which sets the trigger pin as an output pin and used to send the ultrasonic pulses and pinMode(echoPin, INPUT) which is sets the echo pin of the ultrasonic sensor as an input pin and receives the echo signals.

Timer Configuration

The final thing we setup here is the Timer2 configuration. First, we disabled the global interrupts to prevent if there is any interruption during timer configuration. Then, we configured Timer 2 which is 8-bit timer. First, we set TCNT2 = 0 which resets the counter of Timer 2 to 0. Then, we set TCCR2A and TCCR2B to 0 to clear control registers A and B of Timer 2. After that we set the value OCR2A = 252 which compare match register to trigger an interrupt when the timer value reaches 252. Since are using Timer2, which is an 8-bit timer so we cannot exceed the timer value greater than 256. Then, we setup the CTC mode TCCR2A |= B00000100 which sets Timer 2 to operate in Clear Timer on Compare Match (CTC) mode. Then, we use pre-scaler of 1024 by TCCR2B |= B00000111 and enable ISR(TIMER2_COMPA_vect) interrupt service routine by setting TIMSK2 |= B00000010. Finally, we enable global interrupt to resume normal operation after the timer configuration by sei().

```
void loop() {
  float distdoor = measureSensorDoor();  // Measure distance to door
  servo.attach(servoPin);                // Reattach servo motor
  checkDoor(distdoor, wastePercentage);  // Check door position based on waste percentage
}
```

In our operational loop defined within the loop() function, we execute the main cycle of our smart trash can system. Initially, we measure the distance to the door by invoking the measureSensorDoor() function, which utilizes an ultrasonic sensor to determine the distance. Following this, we ensure the servo motor is properly connected and ready for operation by reattaching it to its designated pin using servo.attach(servoPin). Finally, we assess the position of the door based on the current waste percentage in the bin. This is achieved by calling the checkDoor() function with the parameters distdoor and wastePercentage(), which handles the logic of opening or closing the door as necessary. Through this iterative process, our smart trash can system continuously monitors waste levels and autonomously manages waste disposal, ensuring efficient and seamless operation.

```
// Measure distance from sensor to door
long measureSensorDoor() {
  digitalWrite(trigPin, LOW);                 // Set trigger pin to LOW
  delayMicroseconds(2);                       // Delay for stabilization
  digitalWrite(trigPin, HIGH);                // Set trigger pin to HIGH
  delayMicroseconds(10);                      // Delay for pulse duration
  digitalWrite(trigPin, LOW);                 // Set trigger pin back to LOW
  float duration = pulseIn(echoPin, HIGH);    // Measure pulse duration
  float distdoor = (duration * 0.034) / 2.0;  // Convert pulse duration to distance
  return distdoor;                            // Return distance
}
```

This function is used to determine the distance of the object or actor from the sensor of the can by using ultrasonic sensor of the door. We start by setting the trigger pin (trigPin) of the ultrasonic sensor to LOW to prepare for the measurement. After a brief delay of 2 microseconds for stabilization, we set the trigger pin to HIGH for 10 microseconds to generate an ultrasonic pulse. Subsequently, we set the trigger pin back to LOW to conclude the pulse transmission. Following this, we utilize the pulseIn() function to measure the duration of the echo pulse received by the echo pin (echoPin) of the sensor when it detects the reflected pulse. We then convert this duration into distance using the formula: distance = (duration × 0.034)/ 2.0 considering the speed of sound (approximately 0.034 centimeters per microsecond) and accounting for the two-way travel time of the pulse. Finally, we return the calculated distance value as distdoor. This function facilitates accurate distance measurements, essential for determining the position of the can's door relative to the sensor.

```
// Check door position based on waste percentage
void checkDoor(long distdoor, int wastePercentage) {
  if (wastePercentage < maxWastePercentage) {  // Check if waste percentage is below threshold
    if (distdoor < 30 && distdoor > 0) {       // Check if door is close to sensor
      servo.write(360);                        // Open door (360 degrees)
      delay(1000);                             // Delay for door opening
    } else {
      servo.write(0);  // Close door (0 degrees)
    }
  }
}
```

This function is used to check the door position. It will initially be zero. Basically, we assess the appropriate position for the door based on the current waste percentage and the distance to the door from the sensor. We begin by verifying if the waste percentage is below a predefined threshold maxWastePercentage. If it is, we proceed to evaluate the distance to the door. If the door is within a close range of the sensor (less than 30 centimeters) and is positioned at a positive distance (indicating detection), we instruct the servo motor to rotate to 360 degrees, effectively opening the door. Also, we added a delay of 1000 milliseconds to ensure the door has sufficient time to open completely. Conversely, if the door is not within the specified range, we command the servo motor to rotate to 0 degrees, closing the door shut. In a nutshell, the function allows us to dynamically adjust the door position based on the waste percentage and the proximity of the door to the sensor, ensuring efficient waste management and containment within the bin.

```
// Interrupt service routine for Timer2
ISR(TIMER2_COMPA_vect) {
  int wastePercentange = checkWasteDistance();  // Measure waste percentage
  if (wastePercentage >= 0) {                   // Check if waste percentage is valid
    printWastePercentage(wastePercentage);      // Print waste percentage on LCD
  }
}
```

In our interrupt service routine (ISR) for Timer2, we handle the periodic measurement and display of the waste percentage on the LCD screen. When Timer2 triggers a compare match interrupt (TIMER2_COMPA_vect), the ISR is executed. Within the ISR, we call the checkWasteDistance() function to measure the current waste percentage. If the measured waste percentage is valid (i.e., greater than or equal to 0), we proceed to print it on the LCD screen using the printWastePercentage() function. This ISR allows us to update the displayed waste percentage at regular intervals, ensuring real-time monitoring and display of the bin's fill level without interrupting the main program flow. This interrupt service routine will be triggered when the timer value reaches 252 which is every 16ns.

```
// Measure distance of waste level
int checkWasteDistance() {
  unsigned int wastePing = sonar.ping();                  // Perform ultrasonic ping measurement
  Serial.print("Ping: ");                                 // Print ping value
  unsigned int wasteInCm = sonar.convert_cm(wastePing);   // Convert ping to distance in cm
  while (wasteInCm < 0) {                                 // Ensure valid distance measurement
    unsigned int wastePing = sonar.ping();                // Perform ultrasonic ping measurement
    unsigned int wasteInCm = sonar.convert_cm(wastePing); // Convert ping to distance in cm
  }
  wastePercentage = 100 - ((wasteInCm / maxLevel) * 100); // Calculate waste percentage
  delay(1000);                                            // Delay for stability
  checkWasteThreshold(wastePercentage);                   // Check if waste percentage exceeds threshold
  Serial.print("Waste Percentage: ");                     // Print waste percentage
  Serial.println(wastePercentage);                        // Print waste percentage
  return wastePercentage;                                 // Return waste percentage
}
```

This function is used to measure the distance of the waste level with the trash can which utilizes an ultrasonic sensor to perform the distance measurements. Initially, we trigger an ultrasonic ping using the ping() function provided by the sonar object. We then convert the obtained ping value to a distance measurement in centimeters using the convert_cm() function. During this process, we ensure that the distance measurement is valid by checking if it is greater than or equal to 0. If not, we repeat the ultrasonic ping measurement until a valid distance is obtained.

Once a valid distance measurement is obtained, we calculate the waste percentage by subtracting the ratio of the waste level distance to the maximum level from 100. This calculation is based on the assumption that the waste level occupies a certain portion of the maximum level of the trash bin. After calculating the waste percentage, we introduce a delay of 1000 milliseconds for stability. Subsequently, we check if the waste percentage exceeds a predefined threshold using the checkWasteThreshold() function. Finally, we print the waste percentage to the serial monitor for debugging

purposes and return the calculated waste percentage. This function allows us to accurately measure and monitor the waste level within the trash bin, facilitating efficient waste management.

```
// Check if waste percentage exceeds threshold limits
void checkWasteThreshold(int wastePercentage) {
  if (wastePercentage <= 0) {          // Check if waste percentage is below lower limit
    wastePercentage = 1;               // Set waste percentage to minimum value
  } else if (wastePercentage >= 99) {  // Check if waste percentage is above upper limit
    wastePercentage = 100;             // Set waste percentage to maximum value
  }
  ::wastePercentage = wastePercentage;  // Update global waste percentage variable
}
```

This function manages the threshold of the waste percentage and checks if the waste percentage exceeds predefined threshold limits, we ensure that the waste percentage value remains within acceptable bounds. We start by evaluating if the waste percentage is below the lower limit. If so, indicating an invalid or negligible waste level, we set the waste percentage to a minimum value of 1 to maintain a non-zero value for accuracy. Conversely, if the waste percentage exceeds the upper limit, suggesting that the bin is full or close to full capacity, we set the waste percentage to the maximum value of 100. This adjustment ensures that the waste percentage remains within a meaningful range for monitoring purposes. Finally, we update the global waste percentage variable to reflect the adjusted value. This function guarantees that the waste percentage value is appropriately bounded, facilitating accurate monitoring and management of waste levels within the trash can.

```
// Print waste percentage on LCD
void printWastePercentage(int wastePercentage) {
  if (wastePercentage < maxWastePercentage) {  // Check if waste percentage is below threshold
    lcd.clear();                               // Clear LCD display
    lcd.print("Waste percentage:");            // Print label
    lcd.setCursor(0, 1);                       // Move cursor to second row
    lcd.print(wastePercentage);                // Print waste percentage
    lcd.print("%");                            // Print percentage symbol
    PORTB &= B011111;                          // Turn off LED indicating bin full
  } else {
    PORTB |= B100000;             // Turn on LED indicating bin full
    servo.write(0);               // Close door
    lcd.clear();                  // Clear LCD display
    lcd.print("Bin is FULL!!!");  // Print warning message
  }
}
```

This function prints the waste percentage, or the bin is full in the LCD. We start by checking if the waste percentage is below the maximum threshold (maxWastePercentage). If it is, indicating that the bin is not yet full, we proceed to update the LCD display. We first clear the display to remove any previous content, then print a label indicating "Waste percentage:" to denote the nature of the displayed value. We move the cursor to the second row of the display and print the actual waste percentage value, followed by the percentage symbol ("%").

Conversely, if the waste percentage exceeds the maximum threshold, suggesting that the bin is full or close to full capacity, we take appropriate actions to notify the user. We turn on an LED indicator (if present) to signal that the bin is full and may require attention. Additionally, we close the bin's door using the servo motor to prevent further waste deposition. Furthermore, we clear the LCD display and print a warning message indicating "Bin is FULL!!!" to alert the user of the situation. This function enables real-time monitoring of the waste level and provides timely notifications to ensure effective waste management within the trash bin.

Final Code

```
#include <Servo.h> // Include Servo library for controlling servo motor
#include <LiquidCrystal.h> // Include LiquidCrystal library for LCD display
#include <NewPing.h> // Include NewPing library for ultrasonic sensor

// Constants declaration
const int maxWastePercentage = 96; // Maximum waste percentage before indicating bin is full
const int trigPin = 3; // Pin for triggering ultrasonic sensor
const int echoPin = 4; // Pin for receiving echo from ultrasonic sensor
const int servoPin = 5; // Pin for controlling servo motor

// Variables declaration
int wastePercentage; // Current waste percentage
float durationWaste, distanceWaste; // Duration and distance variables for waste level
float maxLevel = 23.0; // Maximum level of waste before indicating bin is full

// LCD pins and maximum distance for ultrasonic sensor
#define wasteTrig A0 // Ultrasonic sensor trigger pin
#define wasteEcho A1 // Ultrasonic sensor echo pin
#define MAX_DISTANCE 200 // Maximum distance for ultrasonic sensor in centimeters

NewPing sonar(wasteTrig, wasteEcho, MAX_DISTANCE); // NewPing object for ultrasonic sensor
Servo servo; // Servo motor object
LiquidCrystal lcd(12, 11, 10, 9, 8, 7); // LCD object

void setup() {
        lcd.begin(16, 2); // Initialize LCD with 16 columns and 2 rows
        lcd.print("...Welcome to..."); // Display welcome message
        lcd.setCursor(0, 1); // Move cursor to the second row
```

```
        lcd.print(" Crap Bag "); // Print device name
        Serial.begin(9600); // Initialize serial communication
        servo.attach(servoPin); // Attach servo motor to its pin
        DDRB |= B100000; // Set LED pin as output
        pinMode(trigPin, OUTPUT); // Set trigger pin of ultrasonic sensor as output
        pinMode(echoPin, INPUT); // Set echo pin of ultrasonic sensor as input
        servo.write(0); // Set servo motor to initial position (0 degrees)
        servo.detach(); // Detach the servo after setting initial position
        cli(); // Disable global interrupt

        // Configure Timer2
        TCNT2 = 0; // Reset Timer2 counter
        TCCR2A = B00000000; // Set Timer2 control register A to 0
        TCCR2B = B00000000; // Set Timer2 control register B to 0

        OCR2A = 252; // Set compare match register for 16ns
        TCCR2A |= B00000100; // Set Timer2 to CTC mode
        TCCR2B |= B00000111; // Set prescaler of 1024
        TIMSK2 |= B00000010; // Enable Timer2 compare match interrupt
        sei(); // Enable global interrupt
}

// Measure distance from sensor to door
long measureSensorDoor() {
        digitalWrite(trigPin, LOW); // Set trigger pin to LOW
        delayMicroseconds(2); // Delay for stabilization
        digitalWrite(trigPin, HIGH); // Set trigger pin to HIGH
        delayMicroseconds(10); // Delay for pulse duration
        digitalWrite(trigPin, LOW); // Set trigger pin back to LOW
        float duration = pulseIn(echoPin, HIGH); // Measure pulse duration
        float distdoor = (duration * 0.034) / 2.0; // Convert pulse duration to distance
        return distdoor; // Return distance
}

// Check door position based on waste percentage
void checkDoor(long distdoor, int wastePercentage) {
```

```
        if (wastePercentage < maxWastePercentage) { // Check if waste percentage is below threshold
                if (distdoor < 30 && distdoor > 0) { // Check if door is close to sensor
                        servo.write(360); // Open door (360 degrees)
                        delay(1000); // Delay for door opening
                } else {
                        servo.write(0); // Close door (0 degrees)
                }
        }
}

// Measure distance of waste level
int checkWasteDistance() {
        unsigned int wastePing = sonar.ping(); // Perform ultrasonic ping measurement
        Serial.print("Ping: "); // Print ping value
        unsigned int wasteInCm = sonar.convert_cm(wastePing); // Convert ping to distance in cm
        while (wasteInCm < 0) { // Ensure valid distance measurement
                unsigned int wastePing = sonar.ping(); // Perform ultrasonic ping measurement
                unsigned int wasteInCm = sonar.convert_cm(wastePing); // Convert ping to distance in cm
        }
        wastePercentage = 100 - ((wasteInCm / maxLevel) * 100); // Calculate waste percentage
        delay(1000); // Delay for stability
        checkWasteThreshold(wastePercentage); // Check if waste percentage exceeds threshold
        Serial.print("Waste Percentage: "); // Print waste percentage
        Serial.println(wastePercentage); // Print waste percentage
        return wastePercentage; // Return waste percentage
}

// Check if waste percentage exceeds threshold limits
void checkWasteThreshold(int wastePercentage) {
        if (wastePercentage <= 0) { // Check if waste percentage is below lower limit
                wastePercentage = 1; // Set waste percentage to minimum value
        } else if (wastePercentage >= 99) { // Check if waste percentage is above upper limit
                wastePercentage = 100; // Set waste percentage to maximum value
        }
        ::wastePercentage = wastePercentage; // Update global waste percentage variable
}
```

```
// Print waste percentage on LCD
void printWastePercentage(int wastePercentage) {
        if (wastePercentage < maxWastePercentage) { // Check if waste percentage is below threshold
                lcd.clear(); // Clear LCD display
                lcd.print("Waste percentage:"); // Print label
                lcd.setCursor(0, 1); // Move cursor to second row
                lcd.print(wastePercentage); // Print waste percentage
                lcd.print("%"); // Print percentage symbol
                PORTB &= B011111; // Turn off LED indicating bin full
        } else {
                PORTB |= B100000; // Turn on LED indicating bin full
                servo.write(0); // Close door
                lcd.clear(); // Clear LCD display
                lcd.print("Bin is FULL!!!"); // Print warning message
  }
}

void loop() {
        float distdoor = measureSensorDoor(); // Measure distance to door
        servo.attach(servoPin); // Reattach servo motor
        checkDoor(distdoor, wastePercentage); // Check door position based on waste percentage
}

// Interrupt service routine for Timer2
ISR(TIMER2_COMPA_vect) {
        int wastePercentange = checkWasteDistance(); // Measure waste percentage
        if (wastePercentage >= 0) { // Check if waste percentage is valid
        printWastePercentage(wastePercentage); // Print waste percentage on LCD
        }
}
```

Technical Difficulties

As we developed the Crap Bag - A Smart Trash Can project, we faced several challenges that required us to think creatively and persistently troubleshoot. One recurring issue was with the LCD hardware, which sometimes displayed strange characters when connected to the Arduino. This disrupted the functionality of the trash can. However, we found a workaround by unplugging the Arduino and letting it rest for a while. After this, the LCD would function normally again.

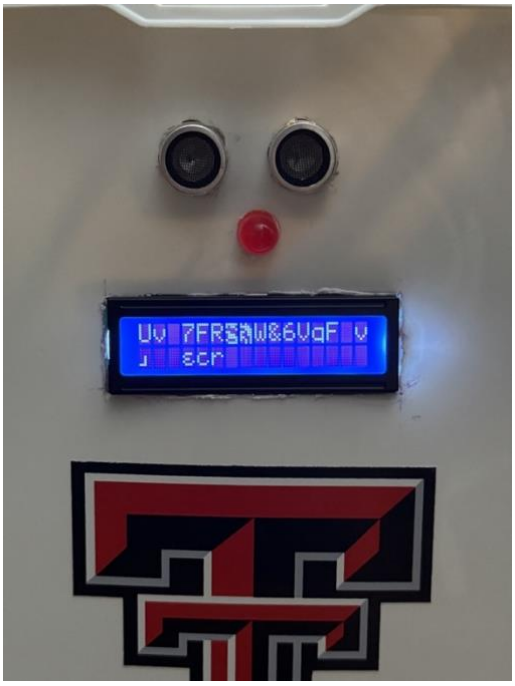

Figure 13: Technical Difficulties - Weird Characters

Another difficulty we encountered was managing the numerous wires needed to connect all the components. With so many wires, it became tricky to keep track of everything, and it was easy for things to get messy. Plus, some wires had loose pins, which made the problem even worse. We had to carefully check and reconnect these wires to make sure everything worked as it should.

Additionally, when we tried to use Timer1 for tasks like printing waste percentage on the LCD, we ran into problems because Timer1 was already being used by the Servo library. This caused conflicts and prevented us from using Timer1 for our purposes. To solve this, we had to switch to using Timer2 instead.

But even Timer2 had its challenges, especially when we tried to use it alongside the NewPing library for measuring waste distance with the ultrasonic sensor. Initially, there were conflicts because the NewPing library was using Timer2 in a way that clashed with our implementation. Thankfully, we discovered a setting in the NewPing library that allowed us to disable the timer functionality we didn't need. Once we did that, everything worked smoothly.

```
159   #define TIMER_ENABLED false      // Set to "false" to disable the timer ISR (if getting "__vector_7"
```

Despite these challenges, we persisted, learned, and found solutions to each problem we encountered. Through teamwork and determination, we successfully built and implemented the Crap Bag - A Smart Trash Can project, making waste management a little bit smarter and easier.

Work Distribution

Raj Dhaubanjar - Hardware and Testing

Simran Thapa – Integrated Servo Motor for Lid Open and Close, Object Detection by Sensor, and LED functionality

Sumit Shrestha – Implemented monitoring waste levels by Ultrasonic Sensor, Managing LCD display and Timer2 functionality

All Collaborated – Idea Generation, Presentation Development and Report Writing

Result

Our implementation of the cyber-physical system (CPS) with Arduino Uno and associated components demonstrated successful integration and functionality. Through rigorous testing, we confirmed the seamless operation of the Arduino Uno alongside the ultrasound sensors, LCD, potentiometer, LED, Servo Motor, and other peripherals. We demonstrated the proper responsiveness of the sensor readings, accurate trash level reading and real-time opening and closing of the lid with the help of servo motor. The sensor outside was able to detect the person in its closed proximity and was able to work every time seamlessly. We were able to show the exact reading of the level of the waste inside the can when some trash is added. We also demonstrated that the lid remains closed after the trash can get full inside. The LCD worked perfectly and displayed greetings when the system is turned on and the trash level when we put some trash inside.

Conclusion

Despite encountering minor challenges during testing, we were able to demonstrate implementation and flawless functioning of our system, the Crap Bag. This prototype has proven to be an innovative solution for modern waste management needs. With its "No physical contact" design, the system ensures user hygiene, cleanliness, and safety by eliminating the need for direct contact with the bin. With real-time updates on trash levels, automated lid control, and user-friendly interface, our prototype demonstrates significant potential for advanced waste management practices in the near future.

Future Work

We implemented a basic Smart Trash Can for this project. But we can obviously implement more than that. Here are some of the future works that can be done and make the Crap bag more appealing and more usable.

Mobile App Integration for Monitoring

We can use mobile apps like Blynk which is a popular platform that enables to create a IoT (Internet of Things) projects easily and efficiently. It serves as the interface through which we can control and monitor the IoT Projects from our smartphones. Since it was a premium version, we couldn't implement it at this time due to lack of money as well as time constraints. Integrating the Crap Bag smart trash can with the Blynk app can provide users with convenient access to real-time monitoring and management of waste levels, enhancing the efficiency and usability of the device. Here's how the Blynk app integration could work, including various features for monitoring waste percentage and other relevant information like Dashboard Display, Waste Percentage Widget or Level, Historical Data Logging, Remote Control and many more.

Cooling and Air Freshener System for Bad Odor

Our prototype intends to improve the efficiency and adaptability by expanding the functionalities including the cooling and air freshener system to keep down the bad odor and decelerate the decaying process of the degradable waste by adding smart sensors that can detect the temperature and odor.

Adding Wheels for Movement

Another huge feature that we will introduce within our system is the wheels for movement. This addition will take this system to a new dimension of convenience, allowing the trash bin to freely roam around the house or apartment or even the office. When provided some commands like sound or hand gesture, the smart bin will move toward the user. This adds flexibility and comfort will further enhance the prototype's accessibility for individuals of all ages and abilities.