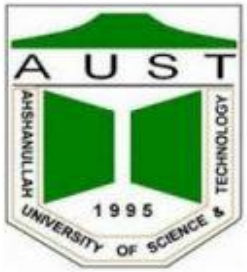
Ahsanullah University of Science & Technology

Department of Computer Science & Engineering



Facilitated Smart Dustbin

CSE 3216

Microcontroller Based System Design Lab

Submitted By:

Solayman Hossain Emon	16.01.04.091
Lamia Nazrin	16.01.04.103
Nahid Hasan	16.01.04.106
Md Toasin Habib	16.01.04.107

Date of Submission : **15 April, 2019**

INTRODUCTION

Environmental pollution manly caused for littering is as old as the civilization itself. It has become a major concern in the last few decades. It is the by product of the development of civilization and in fact a price for the progress. It is more prone in case of Bangladesh. People throw trash everywhere and this tendency is very harmful. So we have decided to make a smart dustbin which people can easily use to throw trash and that will help to reduce the harm. Basically it will be an easily accessible dustbin.

EQUIPMENT

Hardware Components:

Name	Working Procedures for the Project
Dustbin Box	Our project is develop a user friendly smart dustbin. So at the beginning a dustbin box is needed.
Servo Motor SG90	Servo motor is used for opening the cover of the bucket automatically when any garbage is detected in a specific range.
Sonar Sensor (HC-SR04)	Sonar sensor is used for detecting the trash when user take them towards the dustbin. Sonar sensor is working with emit the ultrasonic wave and absorb them.
Battery	For power up the hardware components.

LCD	For displaying the information to the users the LCD is used. In our project the object distance is displayed in LCD. When the bucket is open and closed the related message and time is shown in LCD. The temperature of the weather and some welcome message also shown in the LCD.
Arduino-Uno	Arduino-Uno micro-controller is used for driving and controlling the whole project. The functional code is also uploaded on this controller.
Temperature Sensor (LM-35)	Temperature senor is used for sensing the current temperature of the weather and display this as an user beneficial message.
Gear Motor	To move the wheel of the dustbin box.
Wheel	Move the bucket towards the users.
Motor Driver	Drive the gear motors.
Bluetooth Module	For driving the bucket using the smart phone Bluetooth control application.

Software Components:

- Arduino IDE
- Proteus
- Bluetooth Controller Application

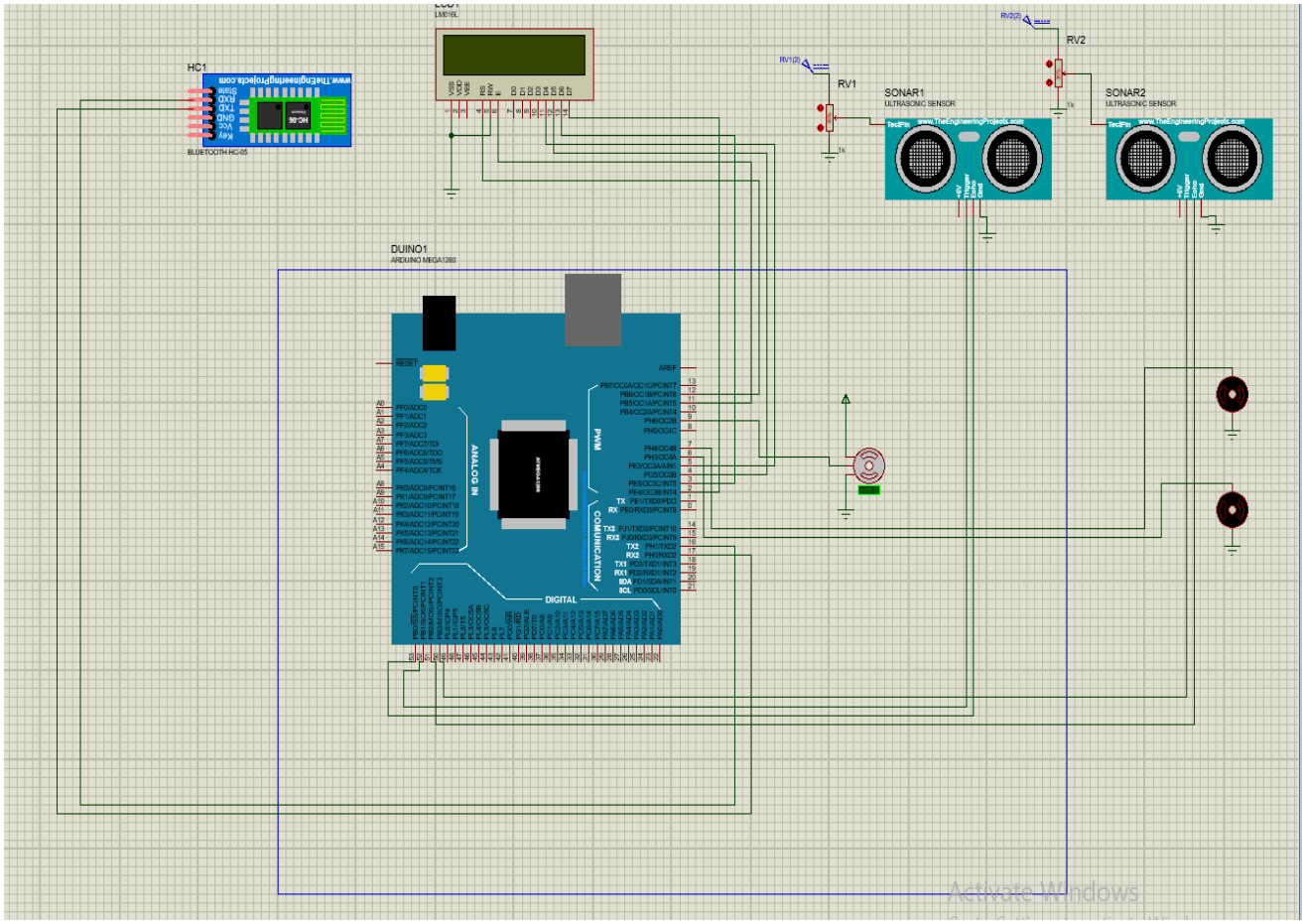
FEATURES

- It can be moved and controlled in a room or place by using a Bluetooth controller.
- When someone will hold the trash in front of the bin, the cover of the bin will open automatically.
- It will show an user beneficial message after throwing the trash inside it.
- When the bucket is filled with trash, the cover of the bucket will not opened.

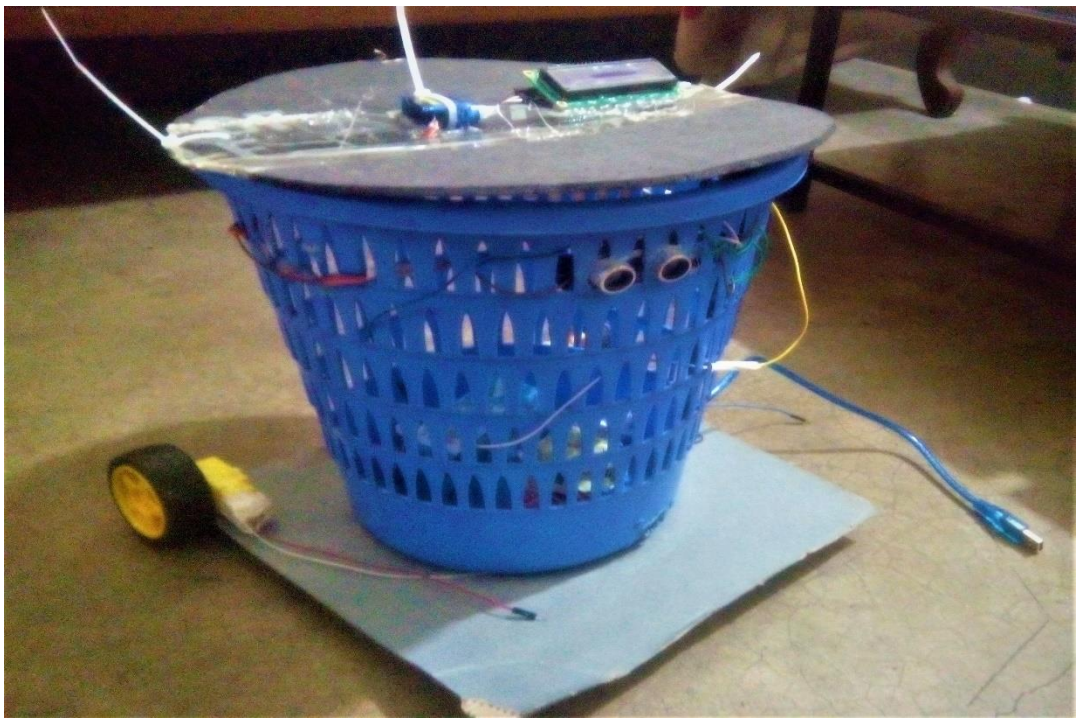
WORKING PRINCIPLE

- **Bluetooth Control**: The dustbin can be move by controlling the smart phone Bluetooth application. For connecting with the Bluetooth, a Bluetooth module is used.
- **Automatic Cover Opened**: By using the Sonar Sensor, the existence of any trash is detect by it. After detecting the trash on a specific range, the cover of the bucket will automatically opened by the Servo motor.
- **Display Message**: When the cover of the bucket is opened and closed, related message and timer will be shown in a LCD. The LCD will display the temperature of the weather as an user beneficial message.

CIRCUIT DIAGRAM



FIGURES OF THE PROJECT



CONSTRAINS

Temperature Sensor: Although we take the average value (take 10 value) from the temperature sensor, the value is not so much correct.

Bluetooth Control: Sometimes the Bluetooth control system is not worked properly for the functional faulting of gear motor and motor driver.

Sensing: If very fast trash movement is occurred in front of the sonar senor, sometimes detection can be delayed.

DOS AND DON'TS

Proposed features	Implement Status
Automatic Cover Opened	Successfully Implemented.
Bluetooth Control	Not implemented properly because of the hardware instrumental fault (gear motor)
Open Cover Display	Successfully Implemented.
Close Cover Display	Successfully Implemented.
User Beneficial Message	Successfully Implemented.
Time with Custom Icon	Successfully Implemented.

CONCLUSION

Basically the main theme of this project is to stop people from littering everywhere and make our country as well as the world clean and beautiful.

APPENDIX

```
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <AFMotor.h>
Servo ServoMotor;
LiquidCrystal_I2C lcd(0x27, 16, 2);   // Set the LCD address to 0x27
AF_DCMotor Left_Motor(2, MOTOR12_64KHZ);// create motor #2, 64KHz pwm
AF_DCMotor Right_Motor(3 , MOTOR34_64KHZ);

//Initialize the Variables
byte SmileIcon[] = {
  B00000,
  B01010,
  B00000,
  B00000,
  B10001,
  B01110,
  B00000,
  B00000
};
byte SadIcon[] = {
  B00000,
  B01010,
  B00000,
  B00000,
  B01110,
  B10001,
  B00000,
  B00000
};
byte DegreeIcon[] = {
  B00000,
  B01110,
  B01010,
```

```
  B01110,
  B00000,
  B00000,
  B00000,
  B00000
};
const int servoPin = 10;
const int trigPin = 52;
const int echoPin = 53;
const int trigPin1 = 50;
const int echoPin1 = 51;
const int Lm35Pin = A8;
long duration, distance, average_Distance = 0, average_Distance1 = 0;
long temp_Distance[3];
int CloseCoverDelay = 7;                // Open the Cover of the bucket for 7 seconds
float temp_Val, temperature = 0;
float temp_Arr[5];
int state = 0;
void setup() {

  pinMode(trigPin, OUTPUT);      // Sets the trigPin as an Output
  pinMode(echoPin, INPUT);       // Sets the echoPin as an Input
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
   ServoMotor.attach(servoPin);   // Attaches the servo on pin 4
  ServoMotor.write(100);         // Initially Closed the cover of the bucket
  delay(100);
  ServoMotor.detach();
  //Serial.begin(9600);            // Starts the serial communication
  Serial1.begin(9600);
  lcd.begin();
  lcd.backlight();
  lcd.createChar(0, SmileIcon);
  lcd.createChar(1, SadIcon);
  lcd.createChar(2, DegreeIcon);
}
```

```
void temp_Measure(){
    temp_Val = analogRead(Lm35Pin);        // Read Temperature
  temp_Val = (temp_Val * 0.48828125);    // Convert adc value to equivalent voltage
   }
float avg_Temperature(){
 float avg_Temp;
   for(int i = 0; i < 10; i++)
   {
     temp_Measure();
     temp_Arr[i] = temp_Val;
     delay(10);
   }
   avg_Temp = (temp_Arr[0] + temp_Arr[1] + temp_Arr[2] + temp_Arr[3] + temp_Arr[4] + temp_Arr[5] + temp_Arr[6] +
temp_Arr[7] + temp_Arr[8] + temp_Arr[9]);
   return avg_Temp;
}
void distance_Measure(const int trigPin, const int echoPin) {
   digitalWrite(trigPin, LOW);         // Clears the trigPin
  delayMicroseconds(2);
digitalWrite(trigPin, HIGH);        // Sets the trigPin on HIGH state for 10 micro seconds
  delayMicroseconds(10);
digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);  // Reads the echoPin, returns the sound wave travel time in microseconds
  distance = duration*0.034/2;        // Calculating the distance
}
long long int avg_Distance_Measure(const int trigPin, const int echoPin){
 long long int avg_Distance;
 for (int i = 0; i < 3; i++)
 {
   distance_Measure(trigPin,echoPin);
   temp_Distance[i] = distance;
   delay(10);
 }
  avg_Distance = (temp_Distance[0] + temp_Distance[1] + temp_Distance[2])/3;   //calculate the average distance
return avg_Distance;
 }
```

```
// Displays the object distance in LCD
void display_Distance(String msg1, String msg2, int distance) {
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(msg1);              // Prints the given custom String
  lcd.setCursor(0,1);
  lcd.print(msg2);              // Prints the given custom String
  lcd.print(distance);          // Prints the distance on the LCD
  lcd.print(" cm");
}
// Displays message on LCD After Opening the Cover of the bucket
void OpenBucket_Display(String msg1, String msg2, String msg3){
   lcd.clear();
  lcd.setCursor(0,0);
  lcd.home();
  lcd.write(0);
  lcd.setCursor(2,0);
  lcd.print(msg1);
  lcd.setCursor(15,0);
  lcd.write(0);
  lcd.setCursor(2,1);
  lcd.print(msg2);
   // Seven(7) Seconds delay before Closing the Cover
for(int i = CloseCoverDelay; i >= 0; i--)
{
   lcd.setCursor(6,1);
   lcd.print(i);
   lcd.setCursor(8,1);
   lcd.print(msg3);
   delay(1000);
   lcd.print(" ");
}
  }
// Displays message on LCD After Closing the Cover of the bucket
void CloseBucket_Display(String msg1, String msg2, String msg3){
lcd.clear();
```

```
  lcd.setCursor(0,0);
  lcd.home();
  lcd.write(1);
  lcd.setCursor(2,0);
  lcd.print(msg1);
  lcd.setCursor(15,0);
  lcd.write(1);
  lcd.setCursor(0,1);
  lcd.print(msg2);
  delay(3000);
  temperature = avg_Temperature();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(msg3);
  lcd.setCursor(0,1);
  lcd.print(temperature);
  lcd.setCursor(6,1);
  lcd.write(2);
  lcd.print("C");
  delay(5000);
  }
// Display message When the bucket is Full
void FullBucket_Display(String msg1){
   lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(msg1);
  lcd.setCursor(0,1);
  lcd.write(1);
  lcd.setCursor(1,1);
  lcd.write(1);
  lcd.setCursor(2,1);
  lcd.write(1);
  lcd.setCursor(3,1);
  lcd.write(1);
  lcd.setCursor(4,1);
  lcd.write(1);
```

```
  lcd.setCursor(5,1);
  lcd.write(1);
}
void Forward()
{
  Left_Motor.setSpeed(255); //Define maximum velocity
  Left_Motor.run(FORWARD); //rotate the motor clockwise
  Right_Motor.setSpeed(200);
  Right_Motor.run(FORWARD);
}
void Backward()
{
  Left_Motor.setSpeed(255);
  Left_Motor.run(BACKWARD); //rotate the motor counterclockwise
  Right_Motor.setSpeed(200);
  Right_Motor.run(BACKWARD);
}
void Right(){
  Left_Motor.setSpeed(255);
  Left_Motor.run(FORWARD);
  Right_Motor.setSpeed(200);
  Right_Motor.run(BACKWARD);
}
void Left(){
  Left_Motor.setSpeed(255);
  Left_Motor.run(BACKWARD);
  Right_Motor.setSpeed(200);
  Right_Motor.run(FORWARD);
}
 void loop() {
// Measure the distance of the object using Sonar Sensor
  average_Distance = avg_Distance_Measure(trigPin, echoPin);
  average_Distance1 = avg_Distance_Measure(trigPin1, echoPin1);
  //Serial.println(average_Distance1);
    display_Distance("Measuring ...","Distance :", average_Distance);   // Diplay the average distance of the object on LCD
  // Condition for Open the Cover of the bucket
```

```
if (average_Distance < 50)
{
if(average_Distance1 <= 4)
  {
     FullBucket_Display("Bucket Full!!!");
     delay(4000);
  }
  else
  {
     ServoMotor.attach(servoPin);
    delay(1);
    ServoMotor.write(0);   // Open the Cover of the Bucket
  OpenBucket_Display("Opened Cover","For","Seconds");  // Display message on LCD after the Cover Open
   ServoMotor.write(100);  // Close the Cover of the Bucket
    CloseBucket_Display("Cover Closed","Thank You !!","The Temperature is :");  // Display message on LCD after Closed the Cover
ServoMotor.detach();
}
}
    if(Serial1.available() > 0){ // Checks whether data is comming from the serial port
    state = Serial1.read(); // Reads the data from the serial port
    }
  if (state == '1') {
   Forward();
  }
else if(state == '2'){
   Backward();
  }
else if(state == '3'){
   Right();
  }
 else if(state == '4'){
   Left();
  }
}
```