GARBAGE SEGREGATOR AND BIN LEVEL INDICATOR

A PROJECT REPOR

TABLE OF CONTENT

1. INTRODUCTION

As cities continue to grow in population and industrialization, the amount of waste produced also increases, leading to a strain on existing waste management systems. The traditional method of disposing of all types of waste in the bin has proven to be unsustainable and harmful to the environment.

Dustbin level indicator technology provides a real-time monitoring system for the level of waste in a dustbin. This helps waste management authorities to plan and manage the collection process more effectively, reducing the risk of overflowing bins and littering in public places.

2. OBJECTIVE

- The primary objective is to provide a cost-effective and efficient solution for monitoring and managing bin levels, particularly in industrial or municipal settings.
- It also significantly aims to improve waste management systems, reduce environmental pollution, and promote sustainable living.

3.DESIGN FLOW

PROBLEM:

There are several issues faced by the residents of the flats. One of them is disposal of solid waste. Unlike private houses, the residents of all the apartments use a common dustbin, which tends to fill up very quickly. This overflowing of garbage is a sanitary issue which might cause diseases like cholera and dengue.

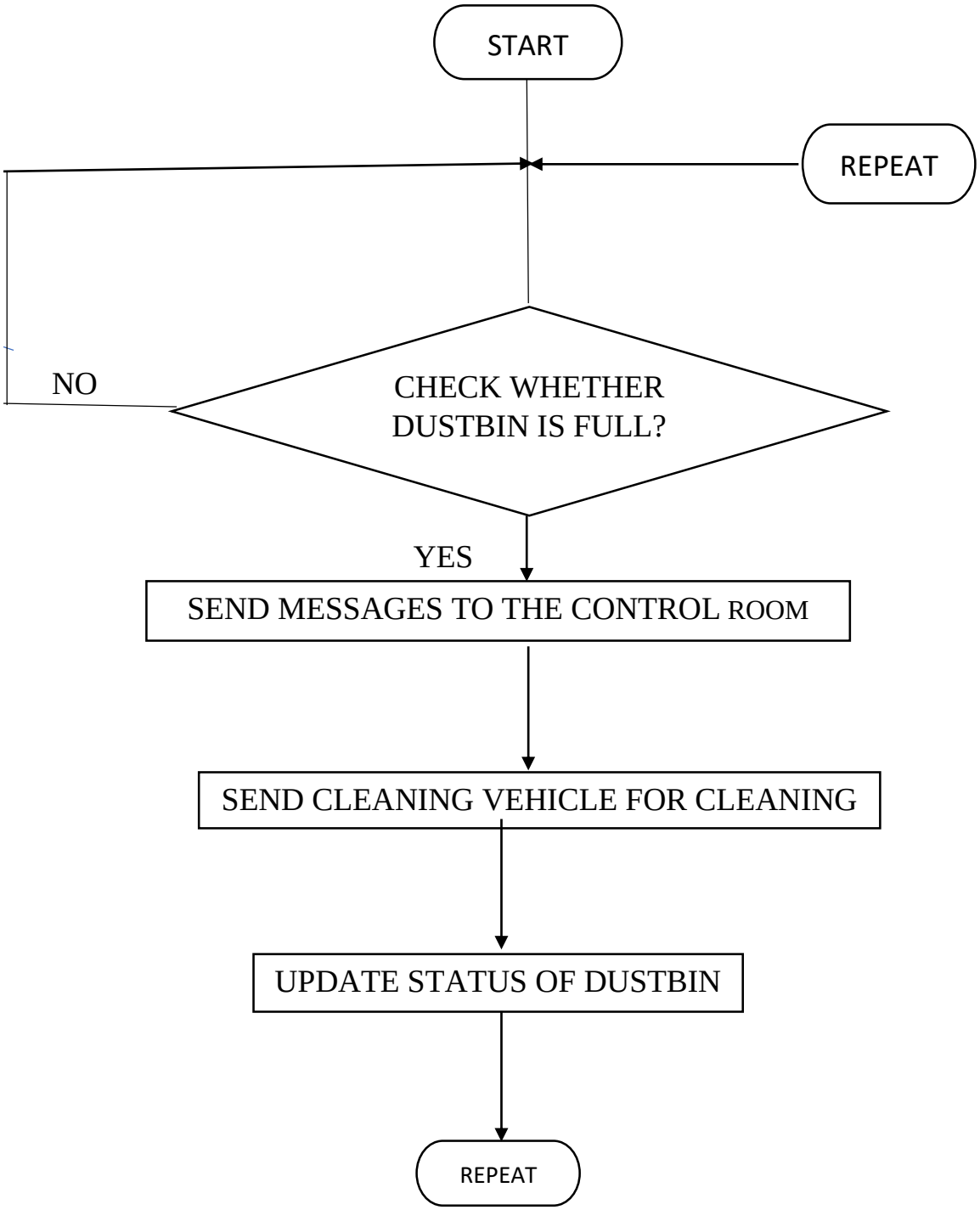


Fig: 3.1 Flow chart

4. HARDWARE REQUIREMENT

We need the following hardware to accomplish our project.

S.NO	COMPONENTS	RANGE	QUANTITY
1	Ultrasonic Sensor	HC-SR04	1
2	Arduino Uno	SMD R4 CH340	1
3	Jumper wires	-	As Required
4	Buzzer	12 volt DC	1
5	Light Emitting Diode	Red,yellow,green	3(Each 1)
6	Resistor	270 ohm	3
7	PCB Board	-	1

4.1. HC-SR04 ULTRASONIC SENSOR:

Ultrasonic Sensor HC-SR04 is a sensor that can measure distance. It emits an ultrasound at 40,000 Hz (40 kHz) which travels through the air and if there is an object or obstacle on its path. It will bounce back to the module.

Fig: 4.1 Ultrasonic sensor

HC-SR04 is an ultrasonic sensor which is used for measuring the distance between the top of the lid to the top of the garbage.

4.1.1. HC-04 Pin out:

The HC-SR04 ultrasonic sensor module typically has four pins:

PIN0	PIN NAME	DESCRIPTION
1	VCC	The Vcc pin powers the sensor, typically with +5V
2	Trigger	Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave
3	Echo	Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
4	GND	This pin is connected to the Ground of the system.

4.1.2. HC-SR04 Sensor Features:

- Operating voltage: +5V
- Theoretical Measuring Distance: 2cm to 450cm
- Accuracy: 3mm
- Measuring angle covered: <15°
- Operating Current: <15ma
- Operating Frequency: 40Hz

4.1.3. Ultrasonic sensor Working:

The HC-SR04 Ultrasonic (US) sensor is a 4 pin module, the pins are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor and it is used in many applications such as measuring distance or sensing objects. The module has

two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. Simply calculate the distance using a microcontroller or microprocessor

Distance = Speed × Time

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module. Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken.

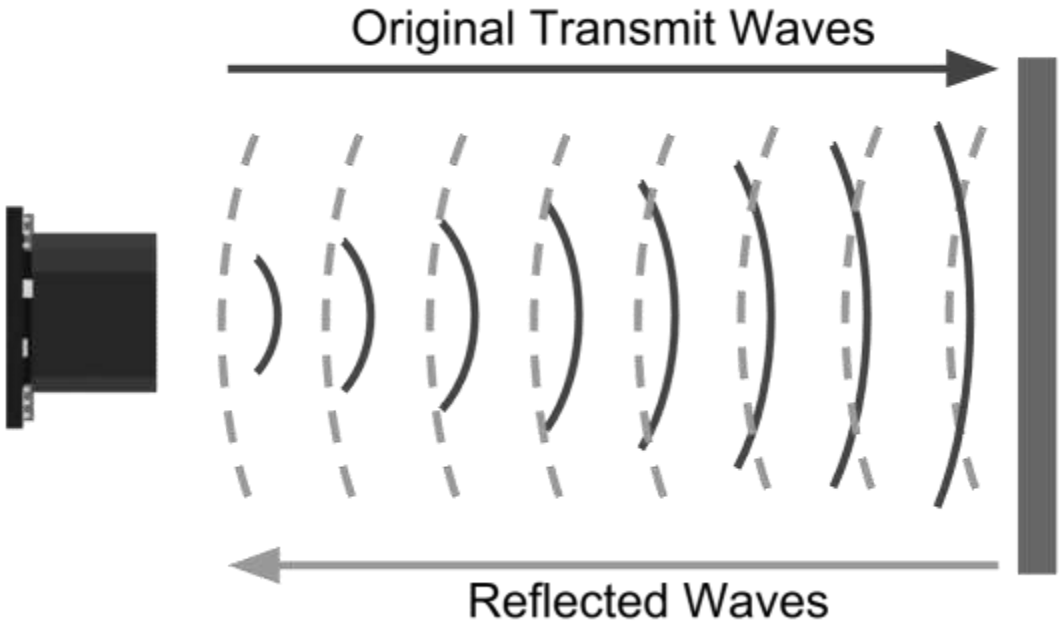


Fig: 4.2 Ultrasonic transmitter and Receiver

4.1.4. Connection of Ultrasonic Sensor with Arduino:

- Sensor has four pins -GND, VCC, trig, echo
- It works under 5v. Connect Vcc pin of sensor with Arduino 5v
- GND pin of ultrasonic sensor is connected to GND of Arduino

- Interface trig pin and echo pin of sensor with any digital pins of Arduino. Here we are connected with digital pin 2 and 3 of Arduino.

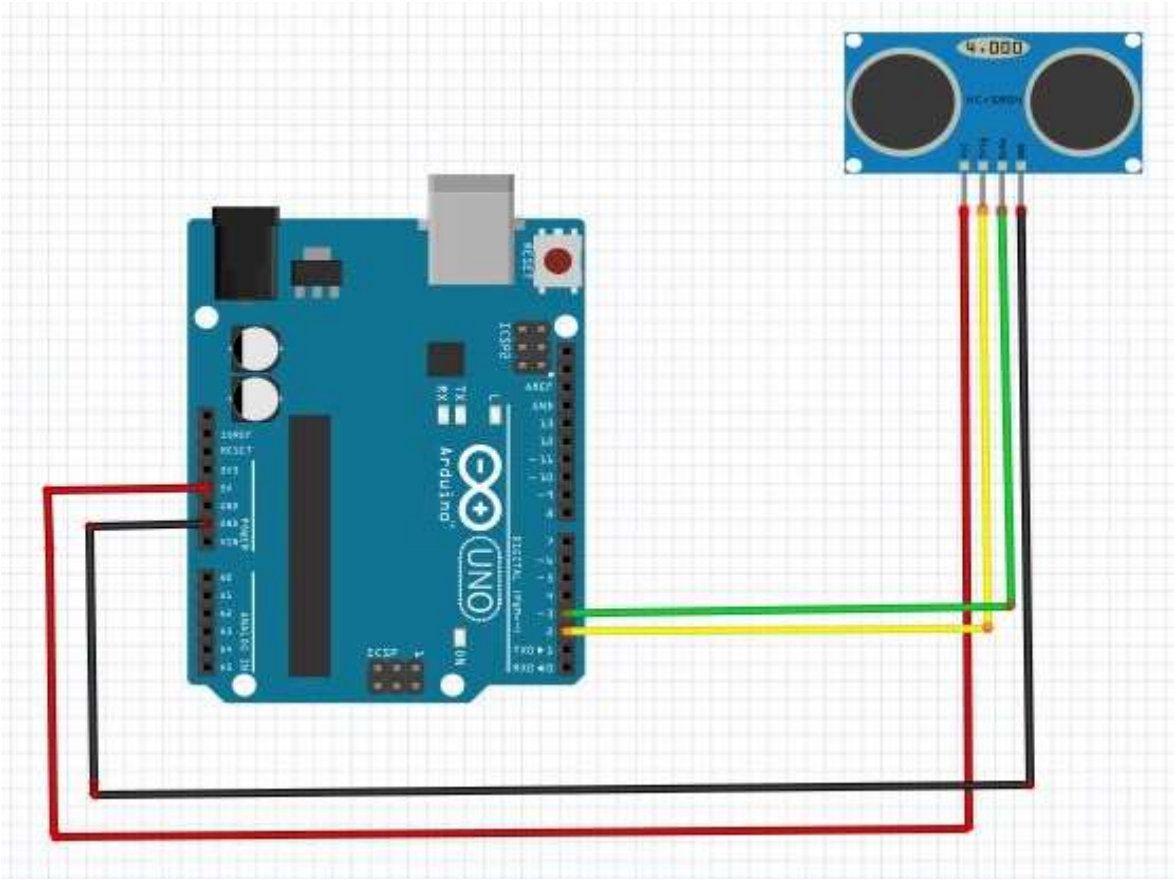


Fig: 4.3.Connection of ultrasonic sensor with Arduino

4.2. ARDUINO UNO

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board. You can simply use

a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.

4.2.1 Arduino Uno SMD CH340:

Fig: 4.2.1 Arduino Uno

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator , a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. This is UNO R3 CH340C ATMega328P compatible with Arduino + Cable for Uno R3. A basic Arduino kit with all the components included eliminating the time our customers for finding and buying Arduino and its basic accessories. Uno R3 CH340C ATmega328p Development Board is the low-cost version of the popular Uno R3 Arduino. It is assembled with the CH340 USB to Serial converter chip, instead of using an Atmega16U2 chip. We have used plenty of these low-cost Arduino boards with CH340 chips, and have found them to work perfectly. The only time the CH340 chip is used is during programming and when using the serial output of the USB port.

During normal operation, this board is identical to the more expensive version without CH340 chip. The pack includes good quality USB-A to USB-B type Arduino connecting cable. It is a Plug-N-Play provision

Specifications and Features: UNO R3 Arduino CH340C

- Microcontroller ATmega328 (SMD) – Interface CH340C
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limits): 5-20V
- Digital I / O Pins 14 (of which 6 provide PWM output)
- DC Current per I Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB (ATmega328) of which 0.5 KB used by boot loader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)

PIN FUNCTION:

Power: The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the God and VIN pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

VIN: The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V: The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

3V3: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

Memory: The ATmega328 has 32 KB (with 0.5 KB used for the boot loader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output: Each of the 14 digital pins on the Uno can be used as an input or output, using pin Mode (), digital Write (), and digital Read () functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kohms.

In addition, some pins have specialized functions:

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analog Write() function.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog Reference() function. Additionally, some pins have specialized functionality:

I2C: A4 (SDA) and A5 (SCL). Support I2C (TWI) communication using the Wire library. There are a couple of other pins on the board:

AREF: Reference voltage (0 to 5V only) for the analog inputs. Used with analog Reference(). **Reset**: Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

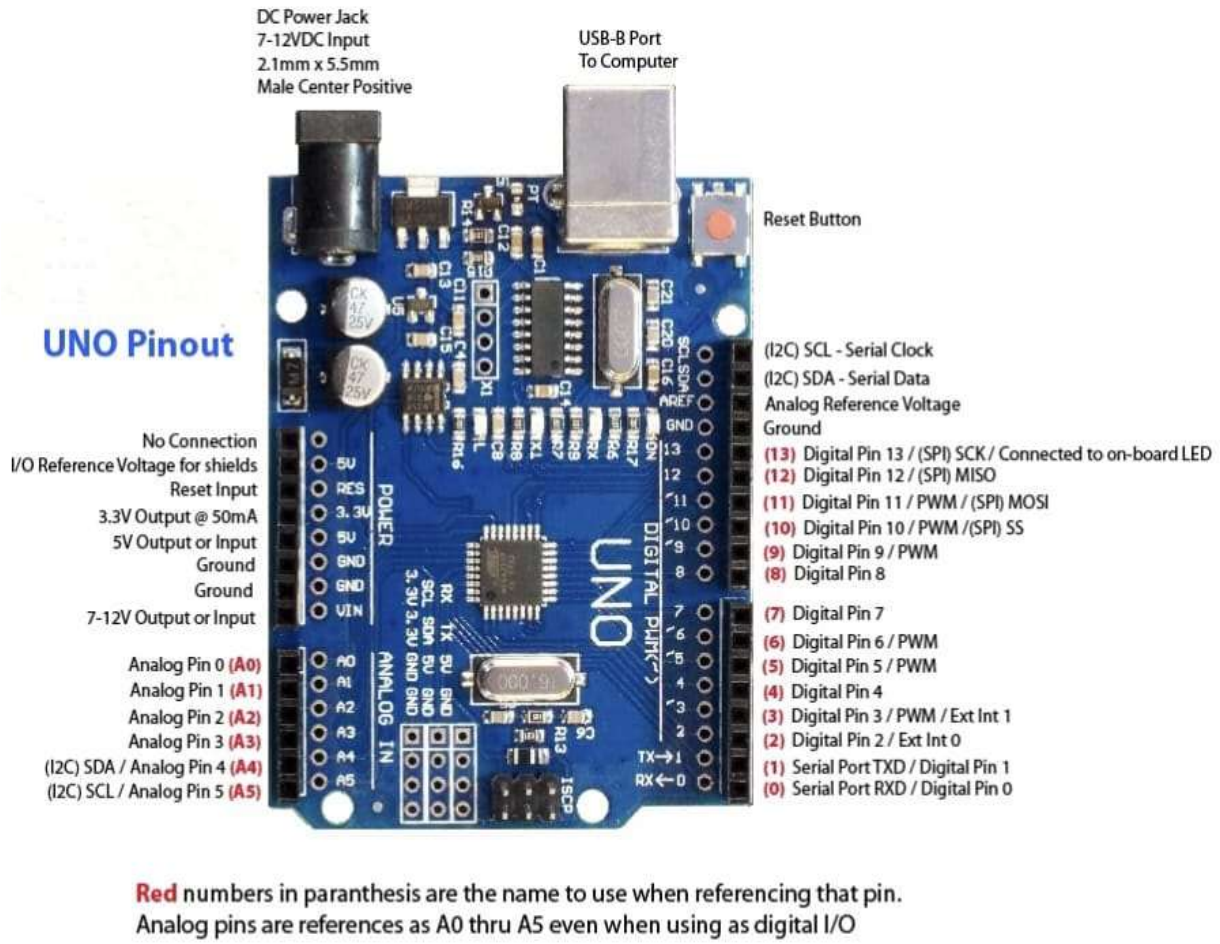


Fig: 4.2 Arduino UNO port specification

4.3JUMPER WIRES:

Jumper wires are wires with a connector at both ends and are suitable for interconnecting components in a breadboard, prototype, or test circuit. They connect components without soldering. If you want to build your own circuits, jumper wires are the way to go.

Fig: 4.3.1.Jumper wire

The three types of jumper wires: male-to-male, male-to-female and female-to-female. Both are ideal for interconnecting two electrical or electronic components. Despite their names, male-to-male jumper wires serve a crucial role in electronics. These wires are ideal for breadboards, prototyping units, and other applications.

4.4 BUZZER 12V DC:

A buzzer that operates on 12V DC is a common electrical component used in various applications such as alarms, notifications, and signalling devices. These buzzers typically have two wires, one for positive and one for negative connection, and they produce a sound when powered by a 12V direct current source. They are available in different sizes, shapes, and sound levels to suit different needs and preferences.

Fig: 4.4.1.Buzzer

5. HARDWARE IMPLEMENTATION

- Connections of the ultrasonic sensor with the Arduino is very simple. Connect the VCC and the ground of the ultrasonic sensor to the 5V and the ground of the Arduino. Then connect the TRIG and ECHO pin of ultrasonic sensor to the pin 2 and 3 of the Arduino respectively (you can use any other pin as well).
- Three LED'S (green, yellow, red) are connected to the resistor of 270ohm separately which the supply voltage from Arduino of 5v dc.
- Each of one LED'S leg is connected with Arduino pins of 7, 8, 9 respectively and the other legs are connected to ground.
- Buzzer is connected to the pin 10 of Arduino (you can use other pins as well) and also give connection to the ground.

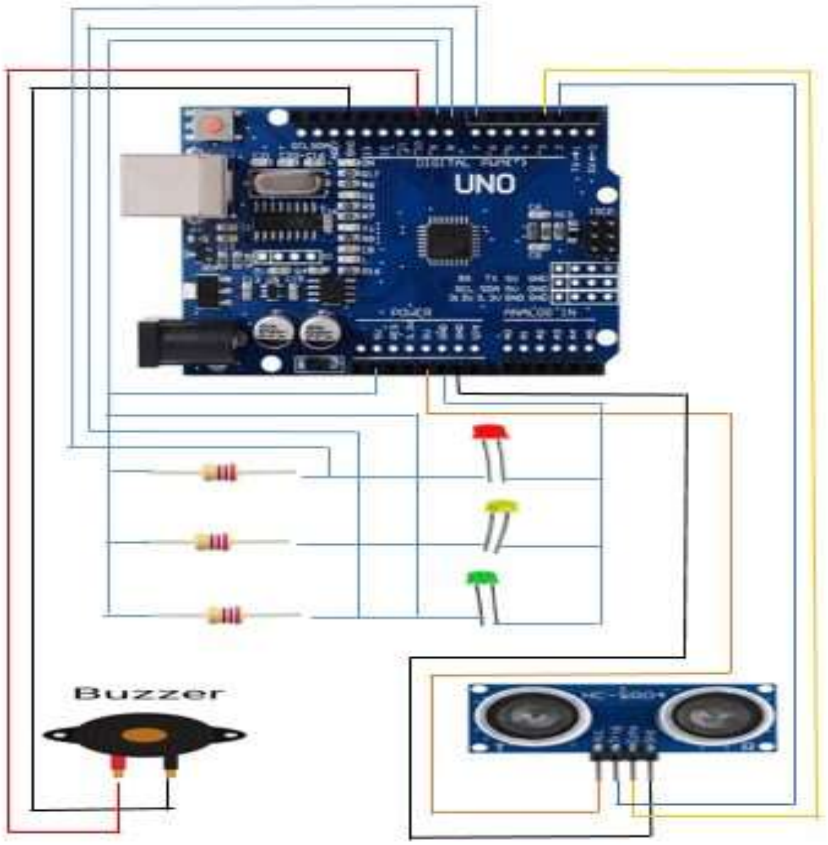


Fig5.1: Block diagram

6. SOFTWARE IMPLEMENTATION

The software required is

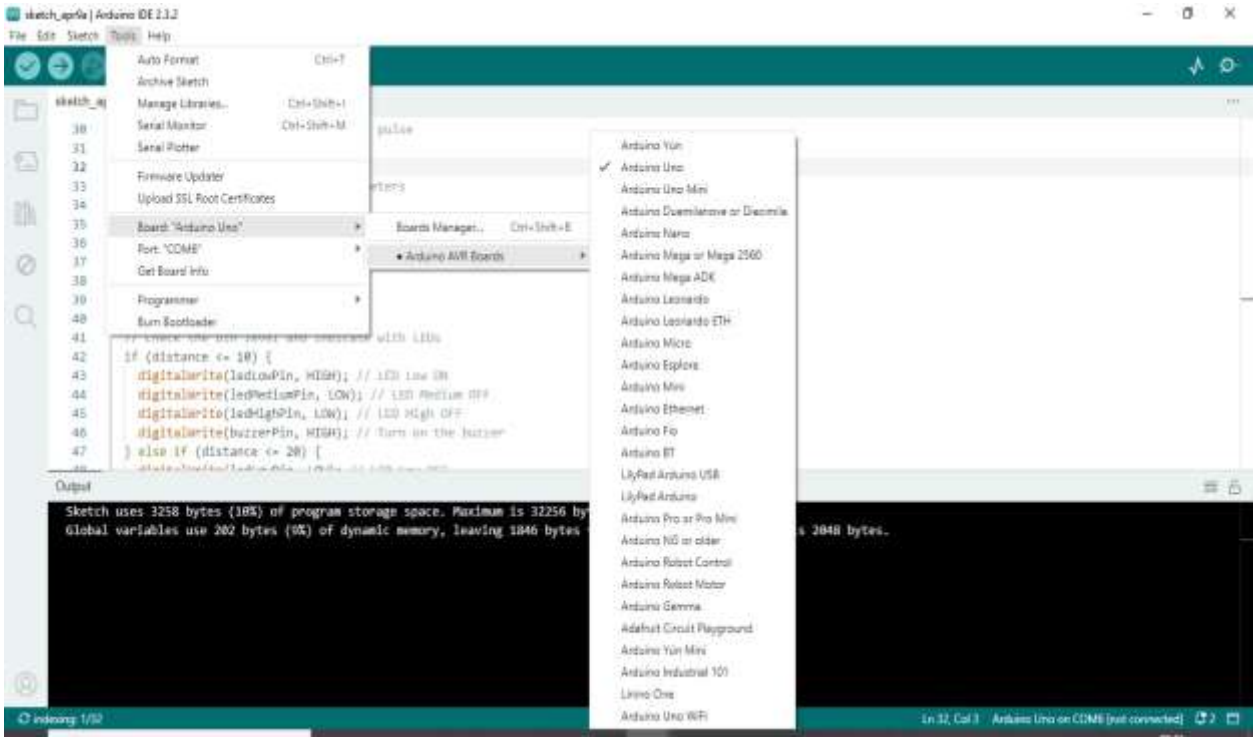
- Arduino IDE

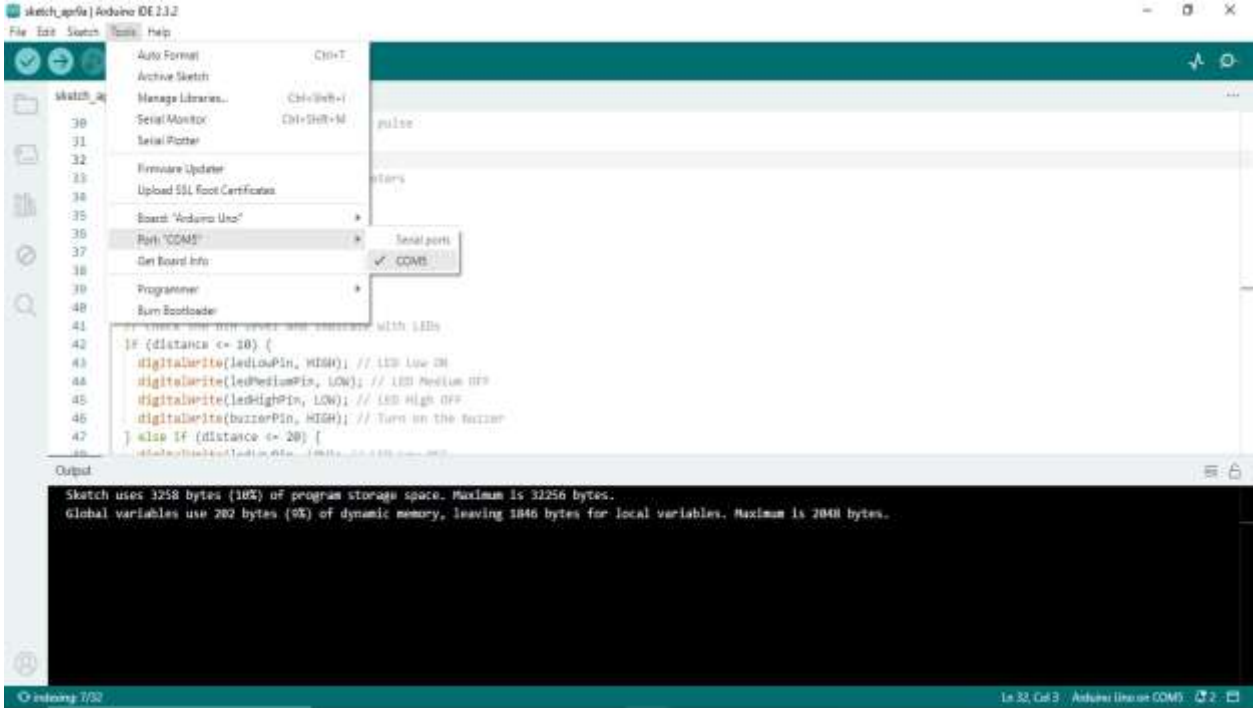
6.1. ARDUINO IDE 2.3.2:

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. The Arduino development environment contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. Software written using Arduino are called sketches. These sketches are written in the text editor. Sketches are saved with the file extension .ion. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino environment including complete error messages and other information. The bottom right-hand corner of the window displays the current board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

Feature:

- Modern, fully featured development environment
- New Board Manager
- New Library Manager
- Board List
- Basic Auto-Completion
- Serial Monitor
- Dark Mode


sketch_apr9a | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Arduino Uno"
Port: "COM8"
Get Board Info
Programmer
Burn Bootloader
Boards Manager... Ctrl+Shift+B
Arduino AVR Boards
Arduino Yún
Arduino Uno
Arduino Uno Mini
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH
Arduino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
Arduino Robot Motor
Arduino Gemma
Adafruit Circuit Playground
Arduino Yún Mini
Arduino Industrial 101
Linino One
Arduino Uno WiFi
if (distance <= 10) {
digitalWrite(ledLowPin, HIGH); // LED Low ON
digitalWrite(ledMediumPin, LOW); // LED Medium OFF
digitalWrite(ledHighPin, LOW); // LED High OFF
digitalWrite(buzzerPin, HIGH); // Turn on the buzzer
} else if (distance <= 20) {
Output
Sketch uses 3258 bytes (10%) of program storage space. Maximum is 32256 by
Global variables use 202 bytes (9%) of dynamic memory, leaving 1846 bytes
s 2048 bytes.
Ln 32, Col 3 Arduino Uno on COM8 (not connected)


sketch_apr9a | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Arduino Uno"
Port: "COM5"
Get Board Info
Programmer
Burn Bootloader
Serial ports
COM5
if (distance <= 10) {
digitalWrite(ledLowPin, HIGH); // LED Low ON
digitalWrite(ledMediumPin, LOW); // LED Medium OFF
digitalWrite(ledHighPin, LOW); // LED High OFF
digitalWrite(buzzerPin, HIGH); // Turn on the buzzer
} else if (distance <= 20) {
Output
Sketch uses 3258 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 202 bytes (9%) of dynamic memory, leaving 1846 bytes for local variables. Maximum is 2048 bytes.
Ln 32, Col 3 Arduino Uno on COM5

7. SOURCE CODE

```
#define trigPin 2
#define echoPin 3
#define ledLowPin 7
#define ledMediumPin 8
#define ledHighPin 9
#define buzzerPin 10 // Define the buzzer pin

void setup()
{
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledLowPin, OUTPUT);
  pinMode(ledMediumPin, OUTPUT);
  pinMode(ledHighPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT); // Set the buzzer pin as output
  Serial.begin(9600);
}

void loop() {
  long duration, distance;

  // Clear the trigger pin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Set the trigger pin HIGH for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Measure the duration of the echo pulse
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance in centimeters
  distance = duration * 0.034 / 2;

  // Print the distance to the serial monitor
  Serial.print("Distance: ");
  Serial.print(distance);
```

```
  Serial.println(" cm");

  // Check the bin level and indicate with LEDs
  if (distance <= 5) {
    digitalWrite(ledLowPin, HIGH); // LED Low ON
    digitalWrite(ledMediumPin, LOW); // LED Medium OFF
    digitalWrite(ledHighPin, LOW); // LED High OFF
    digitalWrite(buzzerPin, HIGH); // Turn on the buzzer
  } else if (distance <= 15) {
    digitalWrite(ledLowPin, LOW); // LED Low OFF
    digitalWrite(ledMediumPin, HIGH); // LED Medium ON
    digitalWrite(ledHighPin, LOW); // LED High OFF
    digitalWrite(buzzerPin, LOW); // Turn off the buzzer
  } else {
    digitalWrite(ledLowPin, LOW); // LED Low OFF
    digitalWrite(ledMediumPin, LOW); // LED Medium OFF
    digitalWrite(ledHighPin, HIGH); // LED High ON
    digitalWrite(buzzerPin, LOW); // Turn off the buzzer
  }

  delay(1000); // Wait for 1 second before next reading
}
```

7.1.CODE EXPLANATION:

Setup Function:

•pinMode(trigPin, OUTPUT); and pinMode(echoPin, INPUT);: These lines configure the pins connected to the ultrasonic sensor. trigPin is set as an output to send trigger signals, while echoPin is set as an input to receive echo signals.

•pinMode(ledLowPin, OUTPUT);, pinMode(ledMediumPin, OUTPUT);, pinMode(ledHighPin, OUTPUT);, pinMode(buzzerPin, OUTPUT);: These lines configure the pins connected to the LEDs and the buzzer.

•Serial.begin(9600);: Initializes serial communication with a baud rate of 9600 for debugging purposes.

Loop Function:

- digitalWrite(trigPin, LOW); and delayMicroseconds(2);: Clears the trigger pin and waits for a short duration.
- digitalWrite(trigPin, HIGH);, delayMicroseconds(10);, and digitalWrite(trigPin, LOW);: Sends a 10-microsecond pulse to the ultrasonic sensor to trigger it.
- duration = pulseIn(echoPin, HIGH);: Measures the duration of the echo pulse (the time taken for the signal to return).
- distance = duration * 0.034 / 2;: Calculates the distance based on the duration of the echo pulse (since sound travels at approximately 34 centimeters per millisecond).
- Serial.print("Distance: ");, Serial.print(distance);, Serial.println(" cm");: Prints the measured distance to the serial monitor.
- Distance Analysis:
- The code checks the measured distance against predefined thresholds to determine the LED and buzzer states:
- If the distance is less than or equal to 5 cm, it turns on the low-level LED and activates the buzzer.
- If the distance is between 5 cm and 15 cm, it activates the medium-level LED.
- If the distance is greater than 15 cm, it activates the high-level LED.
- delay(1000);: Pauses execution for 1 second before taking the next reading

8. TESTING METHODOLOGY

We will test the project in two stages: software and hardware. The software part is to be tested via the Arduino IDE, whereas the hardware part has to be tested physically. It is necessary to check whether the system is working properly or not. To check whether the readings are accurate, we will check the distance pointed out by the sensor by a centimeter tape.

TESTING CRITERIA:

The code should be tested for its basic functionality. This includes verifying that the ultrasonic sensor is correctly connected to the designated pins and that the code accurately measures distances. During this testing phase, it's essential to observe the serial monitor output to confirm that the distance readings are reasonable and correspond to physical distances from obstacles.

The behavior of the LEDs and the buzzer should be evaluated. The LEDs should illuminate according to the predefined distance thresholds (low, medium, high), and the buzzer should sound appropriately when the distance falls within the critical range. Each LED and the buzzer should activate and deactivate as expected, providing clear feedback about the detected distance level.

The code should undergo validation against predefined requirements and specifications. This includes comparing its actual performance against the intended functionality outlined in the project requirements. By ensuring alignment between the code's behavior and the project objectives, any discrepancies or deviations can be identified and addressed accordingly.

In summary, testing the Arduino code involves accessing its functionality, LED and buzzer behavior, robustness, efficiency, and alignment with project requirements. By systematically evaluating these criteria, the code can be validated for deployment in practical applications, ensuring reliable performance and user satisfaction.

9. EXPERIMENTAL OUTPUT

Fig 9.1: Low bin output

Fig9.2: Medium bin output

Fig9.3: High bin output

10. FUTURE SCOPE

The current project is designed for home or apartment use, focusing on monitoring bin levels within residential settings. However, there is significant potential for expansion to include alarming the station and enabling station personnel to monitor bin levels remotely using a mobile application. This report outlines the proposed expansion and the steps required to implement it.

Key Objectives:

1. Integrate the bin monitoring system with the station's alarm system to trigger alerts when the bin reaches a specified level.
2. Develop a mobile application using the Blynk platform to allow station personnel to monitor bin levels remotely in real-time.
3. Ensure the system is user-friendly, reliable, and capable of accurately detecting bin levels.

11. CONCLUSION

We built an efficient garbage monitoring system which can be used to monitor the level of garbage in the dump. This data can be further used to plan garbage collection trips more efficiently, ultimately reducing overflowing bins and helping have better public sanitation.

Bin level indication using LEDs and a buzzer provides a comprehensive solution for monitoring and managing waste levels efficiently. The combination of visual cues through LEDs and auditory alerts via the buzzer ensures both immediate and accessible notifications. By integrating these indicators, users can easily determine when bins require attention, optimizing waste management processes and promoting timely interventions to prevent overflows or operational disruptions. This system not

only enhances operational efficiency but also contributes to environmental sustainability by promoting responsible waste management practices.

12. REFERENCES

- https://youtu.be/ZP0wLe3jIXk?si=DgPQ4Jl0TMatXREh
- https://youtu.be/_KKtbCJt1F4?si=3q6ZbL4QF9pOvEzY
- https://github.com/clarencedelgado/arduino-smart-trashbin-with-indicator/tree/main
- https://youtu.be/agVuYHe0-vI?si=8JoHm0cFk7ReBAnr
- https://github.com/sourabhdeshmukh/Smart-Dustbin
  https://create.arduino.cc/projecthub/Technovation/smart-garbage-monitoring-systemusing-arduino-101-3b813c
- https://www.instructables.com/Smart-Garbage-Monitoring-System-Using-Internet-of-/
- Navghane S S, Killedar M S and Rohokale D V 2016 IoT Based Smart Garbage and waste collection, International Journal of Advanced Research in Electronics And Communication.
- Monika K A, Rao N, Prapulla S B and Shobha G 2016 Smart Dustbin-An EfficientGarbage Monitoring System International Journal of Engineering Science andComputing 6 7113-16.
- Medvedev A, Fedchenkov P, Zaslavsky A, Anagnostopoulos T and Khoruzhnikov S,2015 Waste management as an IoT-enabled service in smart cities In Conference on Smart Spaces Springer International Publishing 104-15