EMBEDDED SYSTEMS

SMART CLEANING BOT

Vaddepally Sai Krushik Reddy

EC22B1062

G Lokesh Naidu

EC22B1025

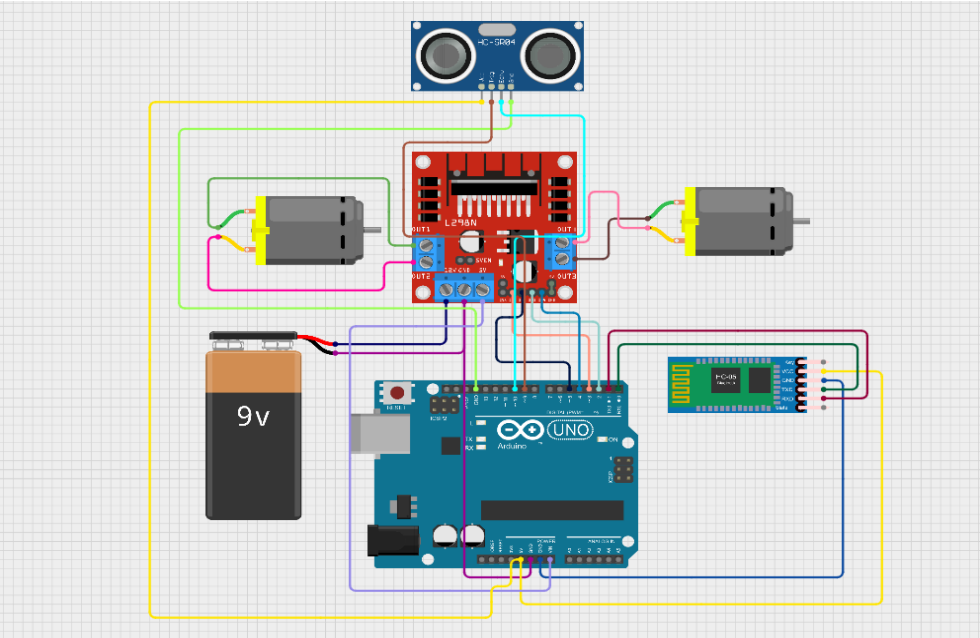
OVERVIEW:

CleanBot, a cleaning robot, aims to transform household cleaning. Unlike traditional methods, CleanBot follows paths set by users, offering a personalized cleaning experience. Equipped with advanced sensors and mapping tech, it navigates paths efficiently, ensuring thorough cleaning while avoiding obstacles. With its compact design and user-friendly interface, CleanBot is a convenient solution for modern homes, promising to save time and enhance cleanliness.

Introduction:

CleanBot introduces a new era in home cleaning, addressing traditional methods' limitations with its innovative features. Users can customize their cleaning by specifying precise paths, ensuring thorough cleaning in any space. CleanBot's adaptability suits various environments, from apartments to commercial areas. Its advanced navigation and attention to detail set it apart from standard cleaners, promising a more efficient cleaning experience. Embrace the future of cleanliness with CleanBot, offering a smarter and cleaner solution to household chores.

Circuit Diagram:



Working Principle:

DC Motor:

A DC motor converts electrical energy into mechanical energy through electromagnetic interaction. Current flows through the armature winding, creating a magnetic field that interacts with the motor's field, causing rotational motion

L298N DC motor driver:

The L298N Dc motor driver operates by controlling the direction and speed of DC motors. It sues H-bridge circuitry to switch the polarity of motor terminals, allowing for forward, reverse and speed control.

HC-05 Bluetooth Module:

The HC-05 Bluetooth module facilitates wireless communication between devices. It pairs with a smartphone or controller and transmits/receives data over Bluetooth. It enables remote control or data exchange between devices in various applications such as robotics and IoT.

Arduino UNO:

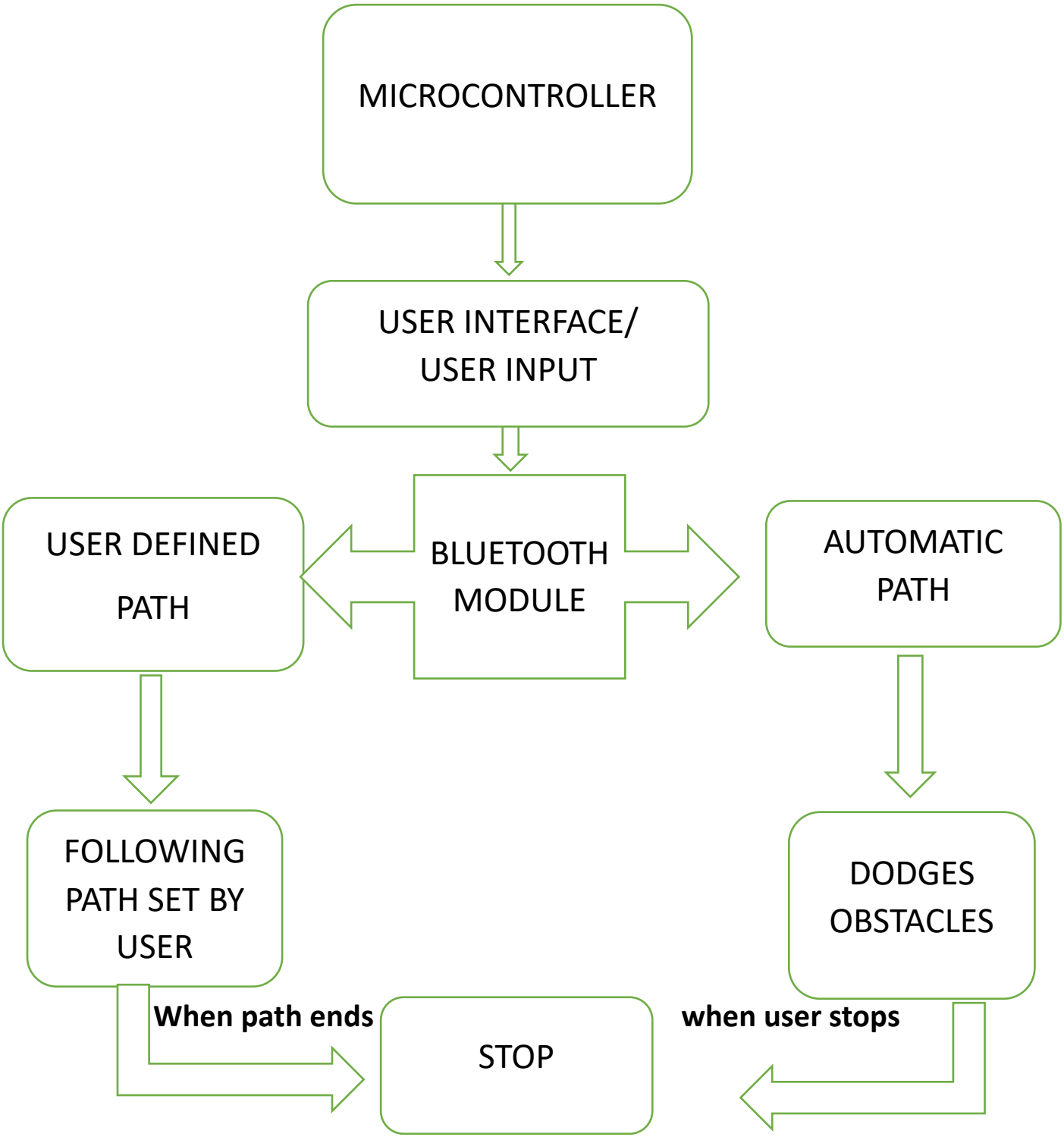
The Arduino UNO isa microcontroller board that executes programmed instructions. It interfaces with sensors, actuators, and other devices to perform tasks. It processes inputs, runs code, and produces outputs, enabling a wide range of electronic projects.

Advantages of CleanBot:

1. Customizable Cleaning: One of the most significant advantages of CleanBot is its ability to clean according to user-defined paths. This feature allows users to target specific areas for cleaning, ensuring that no space is overlooked or unnecessarily cleaned. Whether it's a high-traffic hallway, a cluttered living room, or a hard-to-reach corner, CleanBot can adapt to any cleaning task with precision.
2. Time-Saving: By automating the cleaning process and allowing users to designate cleaning paths, CleanBot saves valuable time. Instead of spending hours manually cleaning floors, users can simply input the desired path, and CleanBot will efficiently navigate and clean the area, freeing up time for other tasks or leisure activities.
3. Efficient Cleaning: CleanBot utilizes advanced sensors and mapping technology to navigate along designated paths while avoiding obstacles. This ensures thorough cleaning without the risk of missing spots or causing damage to furniture or objects in its path. The result is a consistently clean and well-maintained environment with minimal effort required from the user.
4. Convenience: CleanBot offers unparalleled convenience with its user-friendly interface and remote control capabilities. Users can control CleanBot via the dedicated app or control panel, allowing for effortless operation from anywhere in

the home or office. Additionally, CleanBot's compact design and rechargeable battery make it easy to store and use whenever needed.

BLOCK DIAGRAM:



Code:

```
#define IN1 3
#define IN2 5
#define IN3 2
#define IN4 4

#define TRIG_PIN 9 // Ultrasonic sensor trigger pin
#define ECHO_PIN 10 // Ultrasonic sensor echo pin
#define MAX_DISTANCE 35 // Maximum distance to detect obstacles (in centimeters)

char path[100]; // Array to store the path
int pathIndex = 0; // Index to keep track of the path
bool automaticMode = false; // Flag to indicate automatic mode
bool stopRequested = false; // Flag to indicate stop request
char stop;
char value;

void setup() {
  Serial.begin(9600);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

void loop() {
  if (Serial.available() > 0) {
     value = Serial.read();
    Serial.println(value);

    if (value == 'F' && !automaticMode) {
      Forward();
      delay(500);
      Stop();
      storePath('F'); // Store the command
    } else if (value == 'B' && !automaticMode) {
      Backward();
       delay(500);
      Stop();
      storePath('B');
    } else if (value == '0' && !automaticMode) {
      Stop();
      storePath('0');
    } else if(value == 'L' && !automaticMode){
```

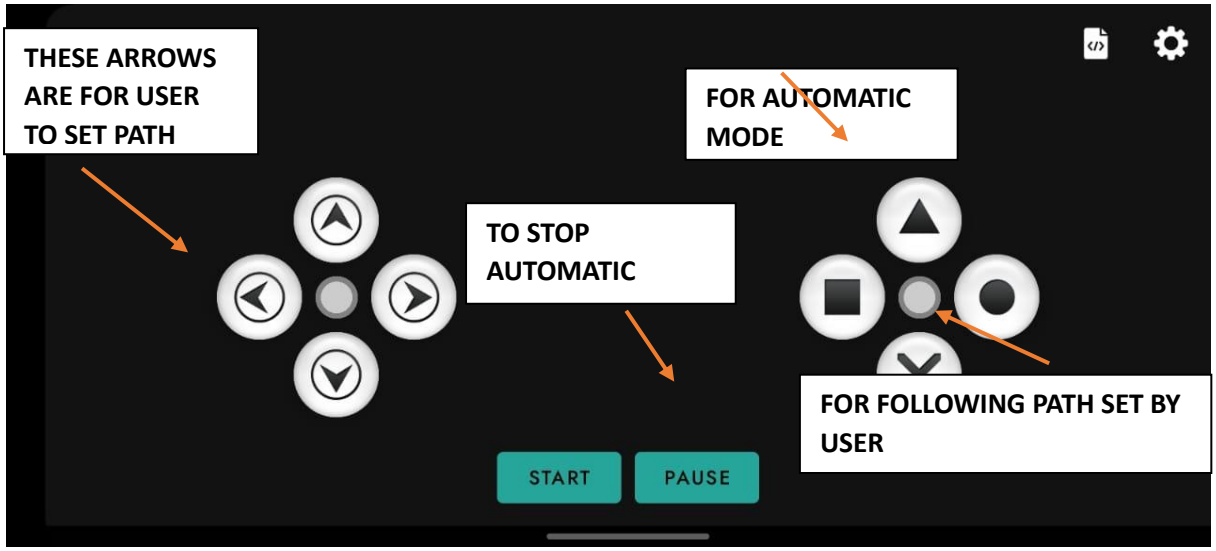
```
      Left();
       delay(200);
      Stop();
      storePath('L');
    } else if(value == 'R' && !automaticMode){
      Right();
       delay(200);
      Stop();
      storePath('R');
    } else if (value == 'X' && !automaticMode) {
      // Follow the reverse path to return to the original position
      followReversePath();
    } else if (value == 'T') {
      // Enter automatic mode
      automaticMode = true;
    } else if (value == 'S') {
      // Enter manual mode
      automaticMode = false;
    }
  }

  while (automaticMode) {
    if (Serial.available() > 0)
    {
    stop = Serial.read();
    Serial.println(stop);
    }
    // If in automatic mode and not stop requested, perform obstacle avoidance
    if (stop!='P')
    {

      dodgeObstacle();
    }
    else
    Stop();
  }
}

void storePath(char command) {
  path[pathIndex++] = command; // Store the command in the path array
}

void followReversePath() {
  for (int i = pathIndex - 1; i >= 0; i--) {
    // Follow the reverse of the stored path
    char command = path[i];
    switch (command) {
      case 'F':
```

```
        {
        Backward();
        delay(500);
         Stop();
        }
        break;
      case 'B':

        {
          Forward();
           delay(500);
         Stop();

        }
        break;
      case 'L':
      {
        Right();
         delay(200);
      Stop();
      }
        break;
      case 'R':
      {
        Left();
         delay(200);
      Stop();
      }
        break;
      case '0':
        Stop(); // No need to move, just stop
        break;
      default:
        // Invalid command
        break;
    }
    delay(500); // Adjust delay as per your requirement
  }
}

void dodgeObstacle() {
  long duration, distance;
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
```

```
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  if (distance < MAX_DISTANCE && distance>0) {
    // Obstacle detected, dodge it
     // Reverse for a second
     Forward();
     delay(1000);
    Left(); // Turn left
    delay(500);
    Backward();
    delay(1000);
    Right();
    delay(500);

     // Adjust the delay according to your robot's speed and turning radius
    stopRequested = true; // Set stop requested flag
  } else {
    // No obstacle, continue forward
    Backward();
    stopRequested = false; // Reset stop requested flag
  }
}

void Forward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

void Backward() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}

void Stop() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}

void Left() {
```

```
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}

void Right() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}
```

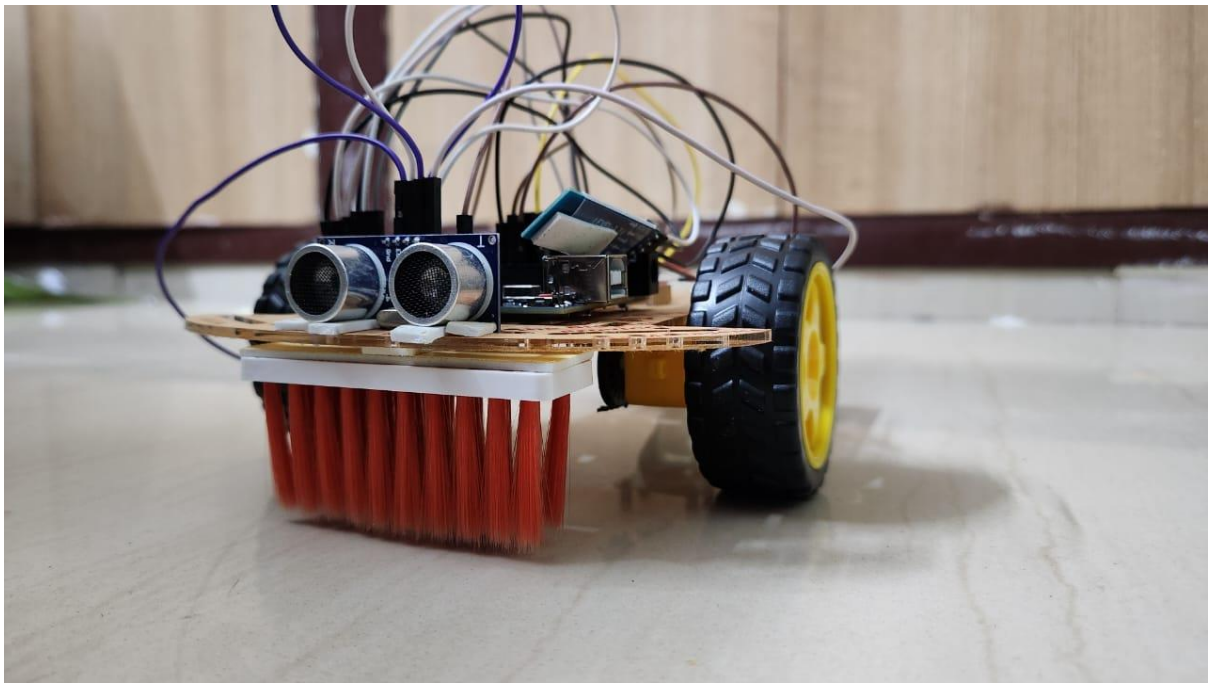
USER INTERFACE



Conclusion:

CleanBot is not just a cleaning device; it's a game-changer in the world of household chores. By allowing users to customize cleaning paths and automate the cleaning process, CleanBot streamlines and simplifies cleaning routines, freeing up time for more important tasks or leisure activities. Its efficiency, convenience, and versatility make it a valuable asset for any home or business looking to maintain a clean and hygienic environment with minimal effort. With CleanBot, the future of home cleaning is cleaner, smarter, and more efficient than ever before.

HARDWARE OUTPUT:



Result:

CleanBot is a groundbreaking cleaning technology that allows users to define specific cleaning paths, saving time and effort. Equipped with advanced sensors and mapping technology, it navigates designated paths while avoiding obstacles. Its versatility makes it ideal for residential and commercial use, promising a new era of efficient cleaning solutions.