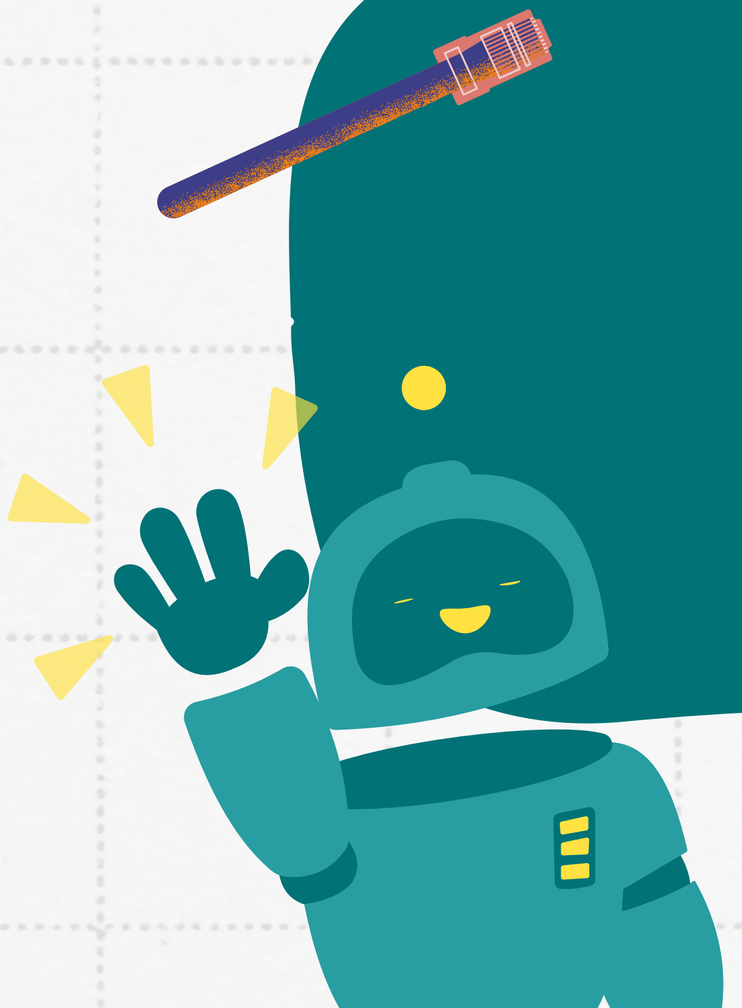
IEEE ZAGAZIG Student Branch

IEEE RAS ROBOTICS & AUTOMATION SOCIETY Zagazig student chapter

ROBOSNAIL



OUR TEAM

Elsayed Abdelhamed

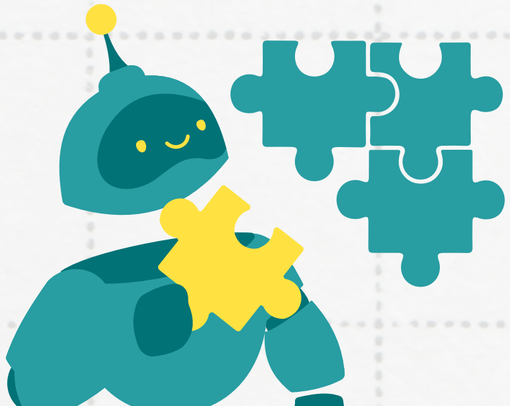
ABdaLLah HaMaDa

Yasmin Ammar

Heba Magdy

Jana Hassan

CONTENT



PROJECT OVERVIEW

We have completed a project involving a robot car named RoboSnail. The car has functions that allow it to move smoothly in all directions.

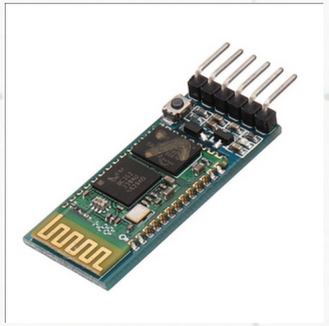
Additionally, we have incorporated an IR Sensor that detects obstacles in front of the car to prevent collisions. If an obstacle is detected, the car stops and the buzzer sounds to alert the user. The car will then reverse to avoid the obstacle.

COMPONENTS

Arduino Board

Arduino is an open-source electronics platform known for its versatility in creating interactive projects

bluetooth module

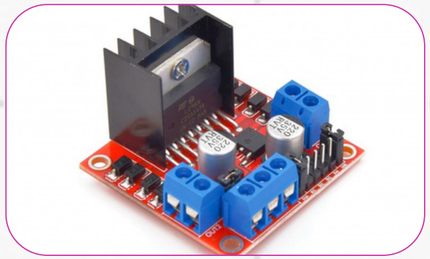


Bluetooth is a type of wireless technology that uses low-energy radio waves to send and receive information between devices.

DC Motors

Responsible for wheel movement

H-Bridge



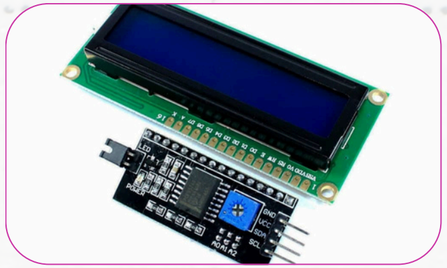
is a circuit that allows control of the direction and speed of a motor, commonly used with Arduino for robotics

COMPONENTS

Sensor

IR sensors are used to detect obstacles and measure distances by emitting infrared light and Detects obstacles in the robot’s path

LCD Display



It’s commonly used in electronics projects for displaying information.

LEDs

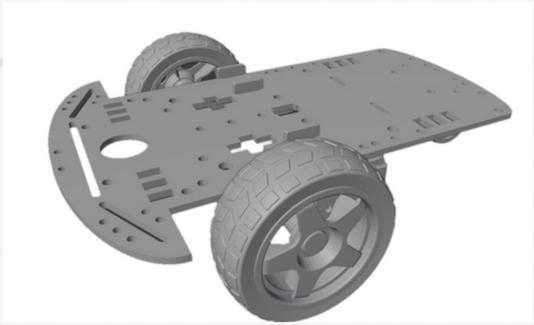
Indicate movement direction and status

Buzzer

Provides auditory feedback when obstacles are detected

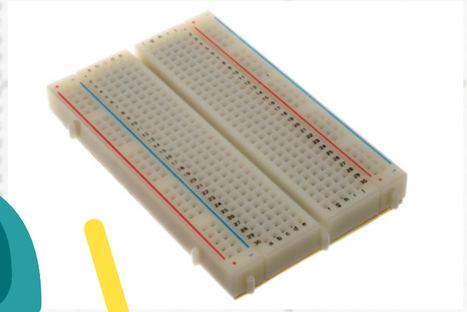
COMPONENTS

Car body

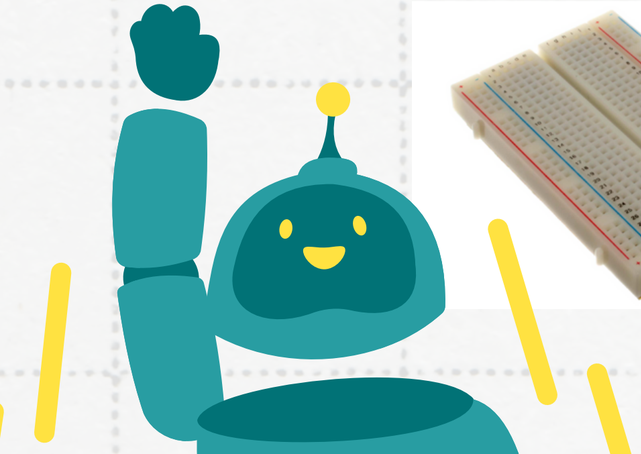


Car wheels

Bread Board



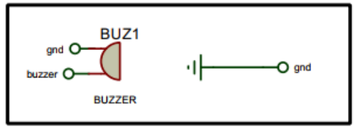
connector



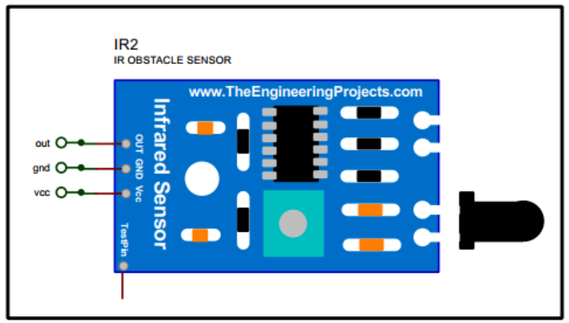
CIRCUIT DIAGRAM

ROBOSNAIL - TEAM 2

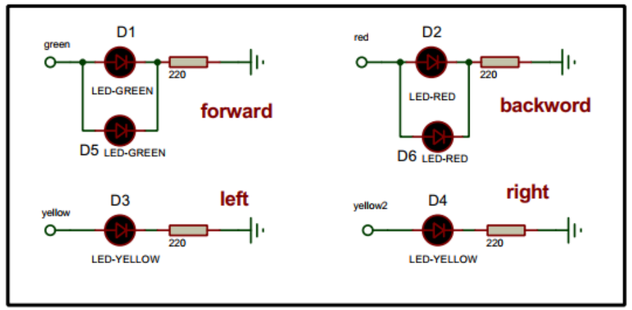
BUZZER



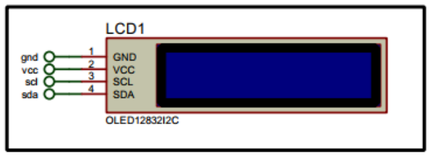
IR SENSOR



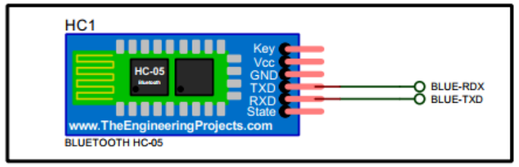
LIGHTS SYSTEM



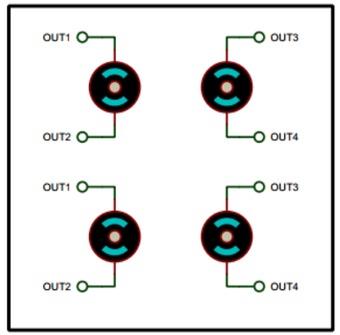
ICD



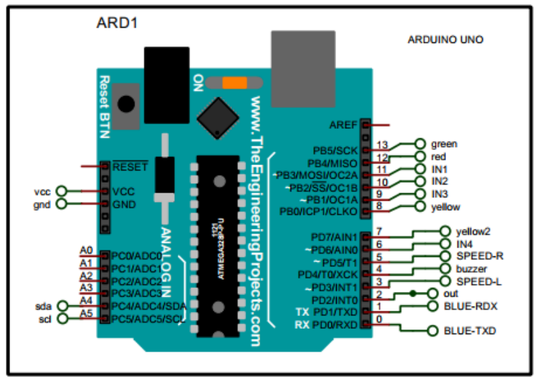
BLUETOOTH (HC-05)



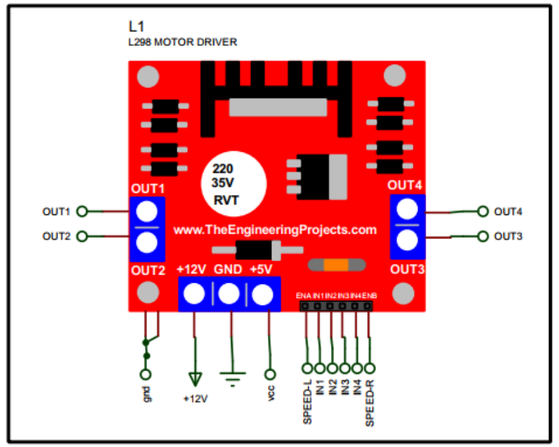
MOTORS



ARDUINO (UNO)



DRIVER



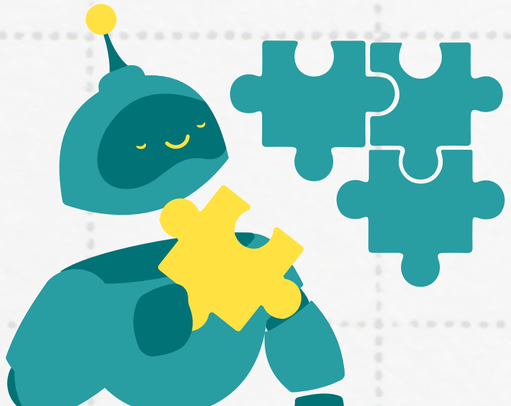
CODE OVERVIEW

LiquidCrystal_I2C: For controlling the LCD display

Pin Definitions: Configuration for motors, LEDs, and speed controllers

Variables: Managing speed and commands

```
#include <LiquidCrystal_I2C.h>  // Include the LiquidCrystal_I2C library

/* Create an instance of the LCD display */
LiquidCrystal_I2C lcd(0x27, 16, 2);

/* Motors Driver */
#define MOTOR_L_IN1       11
#define MOTOR_L_IN2       10
#define MOTOR_R_IN1       9
#define MOTOR_R_IN2       6
#define speedControlerR   5
#define speedControlerL   3
#define buzzerPin         4
#define ir_model          2

int speedValue = 120;
int holder = 20;
char theOrder = ' ';
int sensorRead=digitalRead(ir_model);

/* LED Pins */
#define LEDF 13    // Green LED for Forward
#define LEDR 12    // Red LED for Backward
#define LEDO1 8    // Orange LED for Left Turn
#define LEDO2 7    // Orange LED for Right Turn

// LED and motor pin arrays for initialization
const int motorPins[] = {MOTOR_L_IN1, MOTOR_L_IN2, MOTOR_R_IN1, MOTOR_R_IN2, speedControlerR, speedControlerL};
const int ledPins[] = {LEDF, LEDR, LEDO1, LEDO2};
```



INITIALIZATION

Configure motor and LED pins.

Initialize LCD display.

Start Serial communication.

Set initial speed and display on LCD.

```
void setup() {
  // MOTORS
  pinMode(MOTOR_L_IN1, OUTPUT);
  pinMode(MOTOR_L_IN2, OUTPUT);
  pinMode(MOTOR_R_IN1, OUTPUT);
  pinMode(MOTOR_R_IN2, OUTPUT);
  pinMode(speedControlerR, OUTPUT);
  pinMode(speedControlerL, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(ir_model, INPUT);

  // Initialize LED pins
  pinMode(LEDF, OUTPUT);
  pinMode(LEDR, OUTPUT);
  pinMode(LEDO1, OUTPUT);
  pinMode(LEDO2, OUTPUT);

  Serial.begin(9600);
  Serial.println("Motor Control Program Initialized")
  Serial.print("Initial speed: ");
  Serial.println(speedValue);

  // Initialize LCD
  lcd.init();
  lcd.backlight();   // Turn on the LCD backlight
  lcd.setCursor(0, 0);
  lcd.print("RoboSnail Ready");
}
```

LOOP FUNCTION

Read commands from Serial input

```
void loop() {

  sensorRead=digitalRead(ir_model);
    IR_buzzer(sensorRead);

  // Continuously read the sensor and control the buzzer/motor
  if (Serial.available()) {
    theOrder = Serial.read();
    // Print the received command
    Serial.print("Command received: ");
    Serial.println(theOrder);
    // Only process new commands
    if (theOrder) {
```

Execute commands: Forward (F),Backward (B), Left (L),Right (R), Brake (K), Speed Adjustment (I/D), Little Right (E)

```
// Process commands
switch (theOrder) {
  case 'F': MovingForward(); lcdMessage("Moving Forward"); break;
  case 'B': MovingBack(); lcdMessage("Moving Backward"); break;
  case 'L': TurningLeft(); lcdMessage("Turning Left"); break;
  case 'R': TurningRight(); lcdMessage("Turning Right"); break;
  case 'K': Brake(); lcdMessage("Stop"); break;
  case 'I': updateSpeed(2); lcdSpeed(); break;
  case 'D': updateSpeed(-2); lcdSpeed(); break;
  case 'E': forwardRight(max(speedValue - holder, 0)); lcdMessage("Little Forward Right"); break;
  case 'Q': forwardLeft(max(speedValue - holder, 0)); lcdMessage("Little Forward Left"); break;
  case 'C': backwardRight(max(speedValue - holder, 0)); lcdMessage("Little Backward Right"); break;
  case 'Z': backwardLeft(max(speedValue - holder, 0)); lcdMessage("Little  Backward Left"); break;
}
```

Ensure speedValue is within the 0-255 range.

```
// Limit speed value between 0 and 255
speedValue = constrain(speedValue, 0, 255);
```

Update motor speed based on speedValue

```
// Update the speed of the motors
analogWrite(speedControlerR, speedValue);
analogWrite(speedControlerL, speedValue);
}
```

MOVEMENT FUNCTIONS:

```
// Movement functions
void MovingForward() {
  setMotorState(HIGH, LOW, HIGH, LOW);
  digitalWrite(LEDF, HIGH);  // Turn on GREEN LED for
  digitalWrite(LEDR, LOW);   // Turn off RED LED
  digitalWrite(LEDO1, LOW);  // Turn off LEFT LED
  digitalWrite(LEDO2, LOW);  // Turn off RIGHT LED
}
```

```
void TurningRight() {
  setMotorState(HIGH, LOW, LOW, HIGH);
  digitalWrite(LEDF, LOW);   // Turn off GREEN LED
  digitalWrite(LEDR, LOW);   // Turn off RED LED
  digitalWrite(LEDO1, LOW);  // Turn off LEFT LED
  digitalWrite(LEDO2, HIGH); // Turn on RIGHT LED
}
```

```
void MovingBack() {
  setMotorState(LOW, HIGH, LOW, HIGH);
  digitalWrite(LEDF, LOW);   // Turn off GREEN LED
  digitalWrite(LEDR, HIGH);  // Turn on RED LED for BACKWARD
  digitalWrite(LEDO1, LOW);  // Turn off LEFT LED
  digitalWrite(LEDO2, LOW);  // Turn off RIGHT LED
}
void TurningLeft() {
  setMotorState(LOW, HIGH, HIGH, LOW);
  digitalWrite(LEDF, LOW);   // Turn off GREEN LED
  digitalWrite(LEDR, LOW);   // Turn off RED LED
  digitalWrite(LEDO1, HIGH); // Turn on LEFT LED
  digitalWrite(LEDO2, LOW);  // Turn off RIGHT LED
}
```

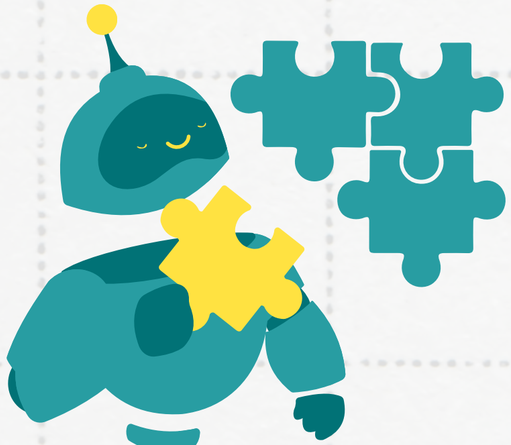
```
  setMotorState(LOW, LOW, LOW, LOW);
  digitalWrite(LEDF, LOW);   // Turn off all LEDs
  digitalWrite(LEDR, LOW);
  digitalWrite(LEDO1, LOW);
  digitalWrite(LEDO2, LOW);
}
```

SPECIAL MOVEMENT:

```
void forwardRight(int speed) {
  analogWrite(MOTOR_R_IN1, speed);
  digitalWrite(MOTOR_L_IN1, HIGH);
  digitalWrite(MOTOR_L_IN2, LOW);
  digitalWrite(MOTOR_R_IN2, LOW);
  digitalWrite(LEDO2, HIGH);  // Turn on RIGHT LED
}
```

IR SENSOR AND BUZZER

- Detects obstacles using the IR sensor.
- Activates the buzzer and applies the brake if an obstacle is detected.



```
void IR_buzzer(int sensor) {

  if (sensor == LOW) {  // If object detected (sensor value below threshold)
    digitalWrite(buzzerPin, HIGH);;  // Turn on buzzer
    Brake();
    delay(1000);
    MovingBack();
    delay(500);
    Brake();



    // Apply brake
  }
  else
  {
    digitalWrite(buzzerPin, LOW);;  // Turn on buzzer
```

LCD FEEDBACK

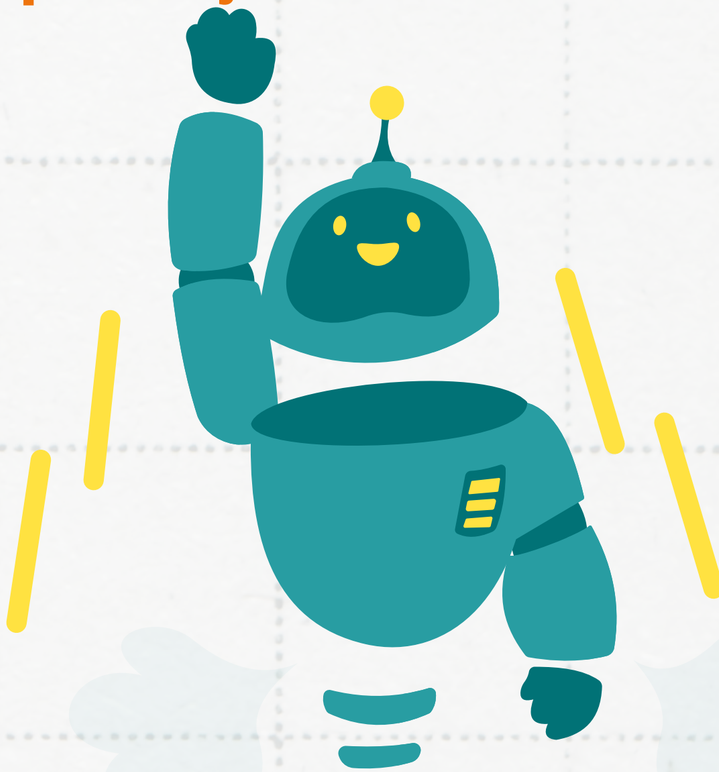
Displaying Commands:

- Show current status on LCD.
- Update messages for movement direction, braking, and speed adjustments.

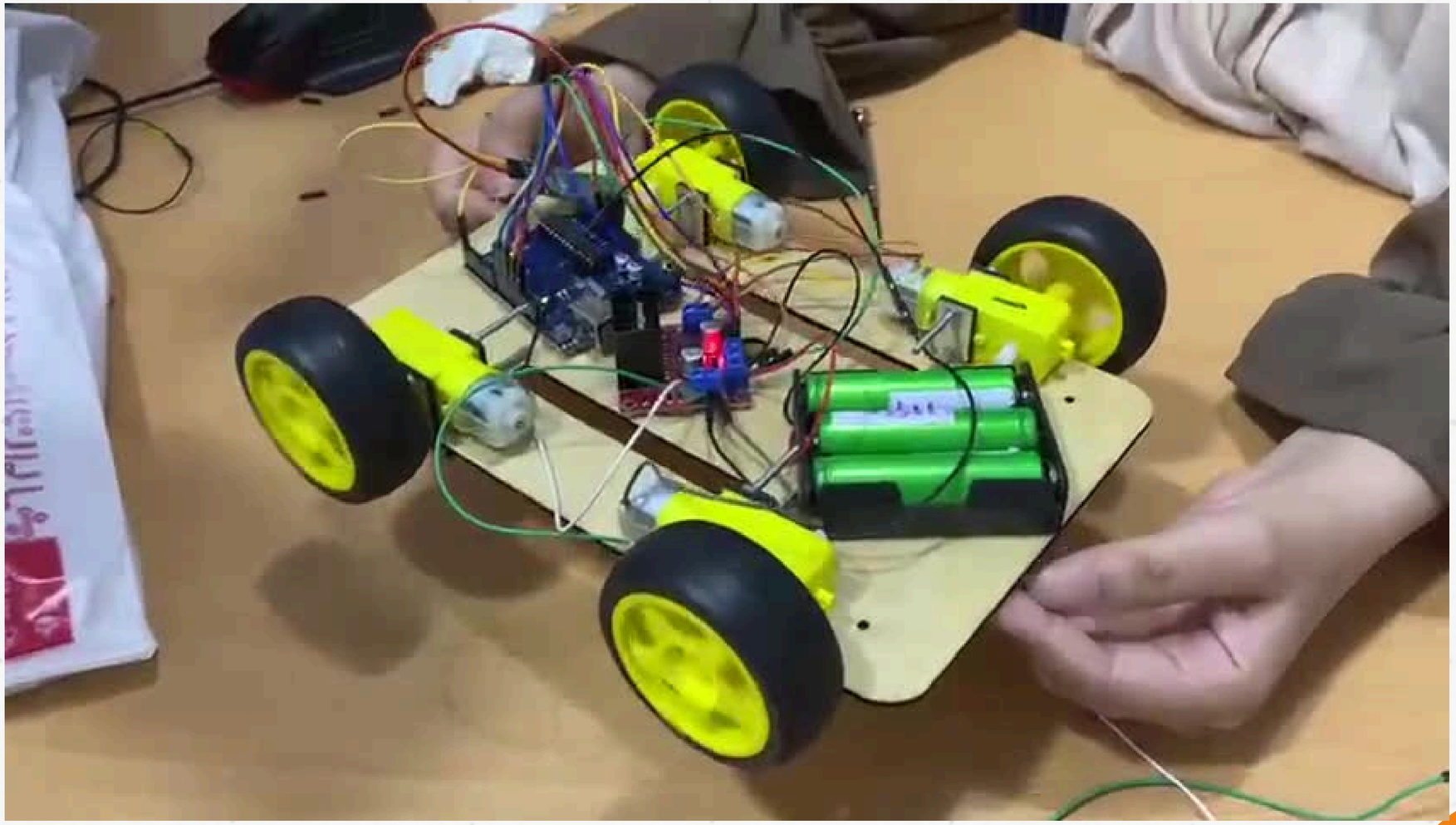
SYSTEM BEHAVIOR

How It Works:

- Commands are received via serial input.
- LEDs provide visual feedback on movement direction.
- LCD displays current status and speed.
- Motors adjust speed and direction based on commands.

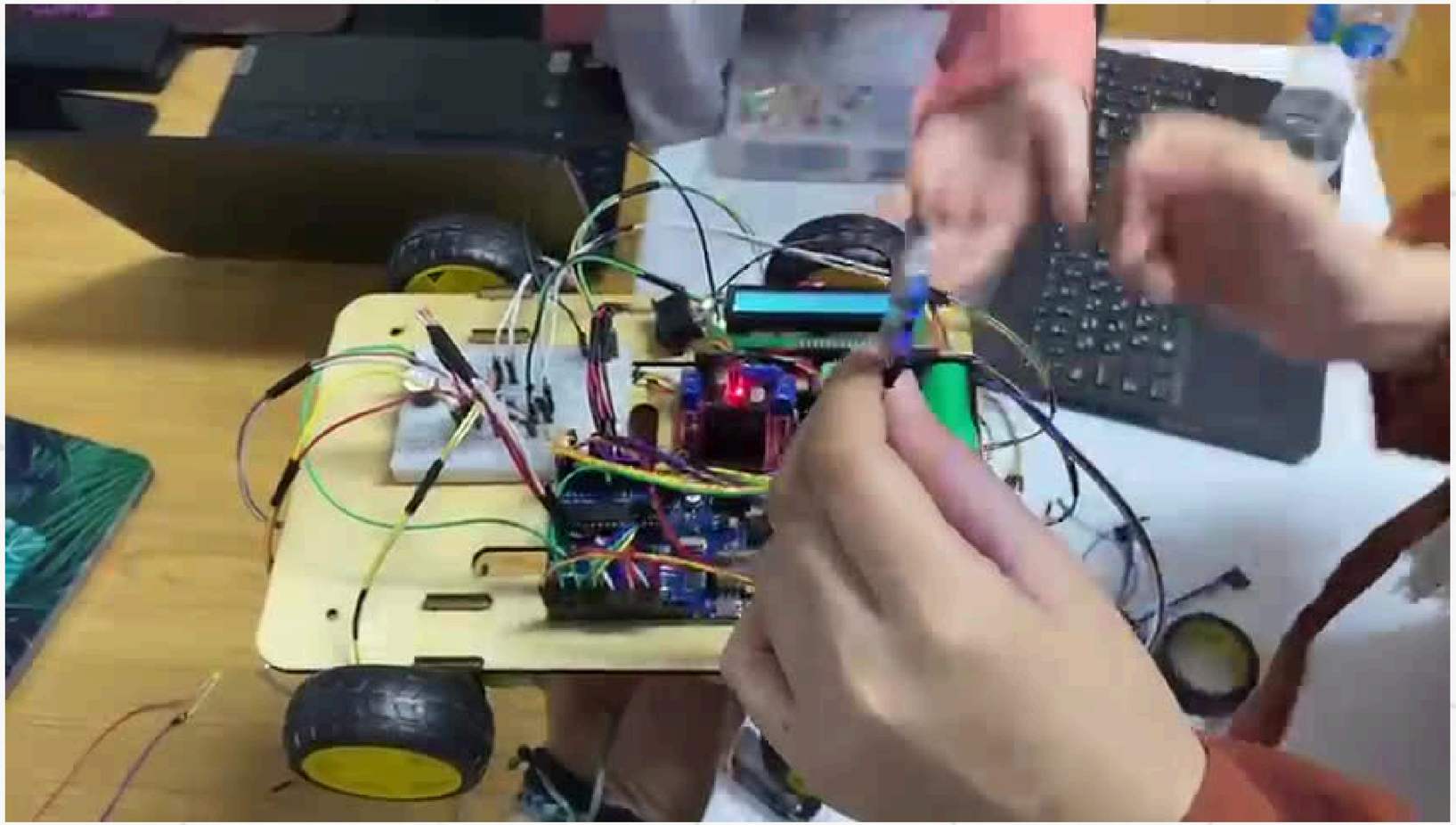


FIRST TEST

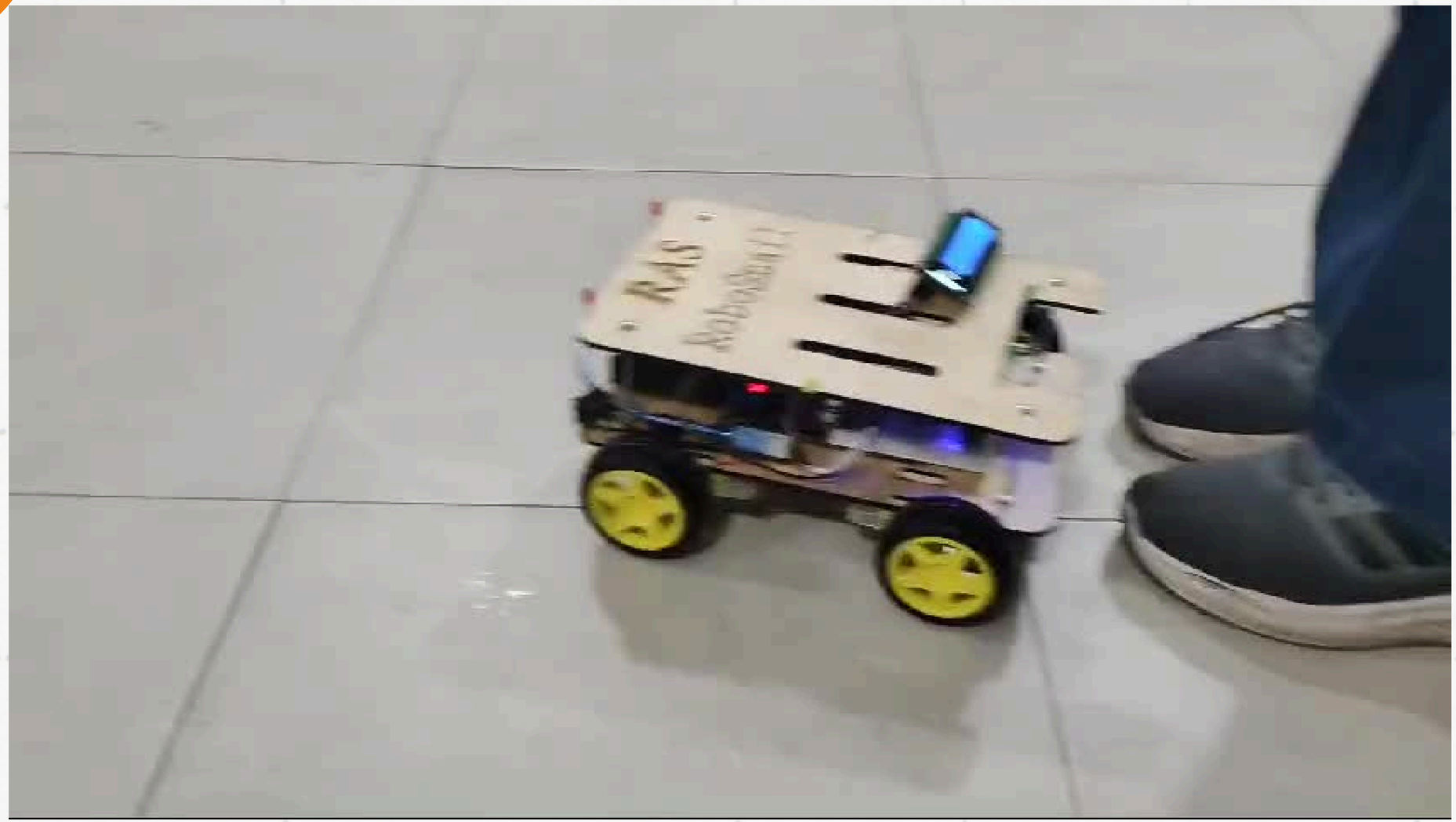


MOTORS TEST

SYSTEM TEST



IR SENSOR TEST



FINAL TEST

THANKS

ANY QUESTIONS?