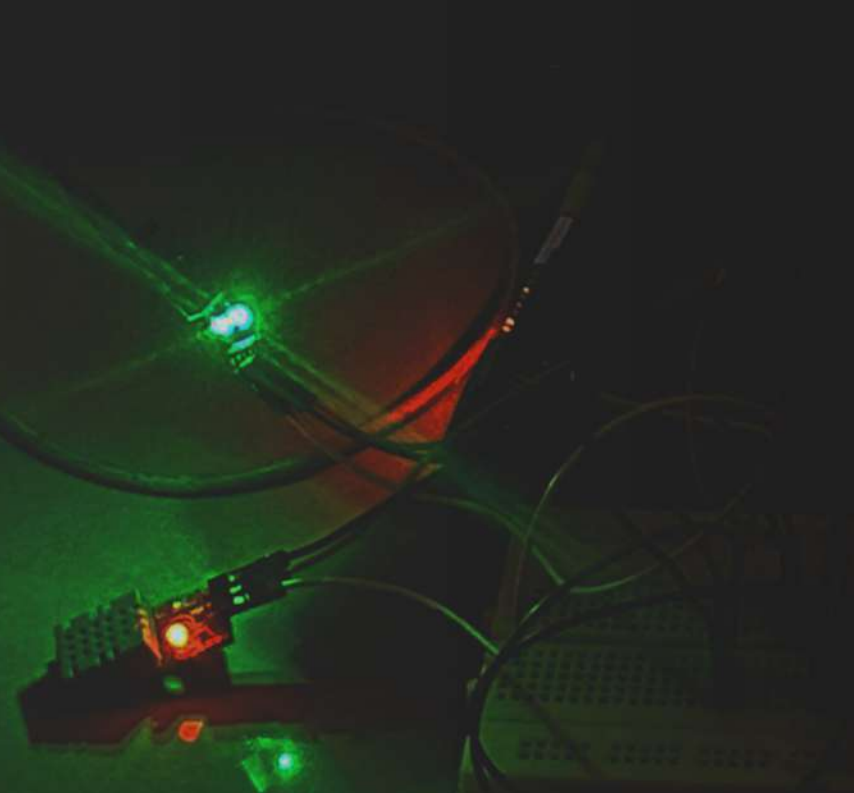


```
1 #include <stdio.h>
2 #include "thingy.h"
3 #include "lib.h"
4
5 // WiFi credentials
6 const char* ssid = "ssid";
7 const char* password = "password";
8
9 // Thingy pin info
10 const int ledPin = 13;
11 const int pushPin = 2;
12
13 // Pins
14 #define LED_PIN 13 // LED connected to GND
15 #define PUSH_PIN 2 // Push button to GND (VCC)
16 #define GND_PIN 5 // GND sensor to GND (VCC)
17
18 // Pin select
19 int ledPin = LED_PIN;
20 int pushPin = PUSH_PIN;
21 int gndPin = GND_PIN;
22
23 // WiFi client
24 WiFiClient client;
25
26 void setup() {
27   Serial.begin(115200);
28   pinMode(ledPin, OUTPUT);
29   pinMode(pushPin, INPUT_PULLUP);
30   pinMode(gndPin, INPUT_PULLUP);
31   while (!client.connectToWiFi()) {
32     delay(100);
33     Serial.print(".");
34   }
35   Serial.println("WiFi connected.");
36 }
```

# MECHATRONICS

ANUSHREE CHAUHAN  
2023UG009

RASHMI RAY  
2023UG047



# FlowIQ – airflow with intelligence.

---

## RELEVANCE :

Students in hostels often endure hot, humid rooms during intense study sessions or at night. Traditional coolers demand frequent water refilling, manual pump management, and constant adjustment of airflow direction—tasks that become burdensome amid their busy routines. As a result, coolers are used inefficiently or simply neglected, leading to both discomfort and unnecessary water wastage.

## OBJECTIVE :

To design and prototype a smart cooler integrating sensors, actuators, and microcontroller programming.

## PROJECT DESCRIPTION :

The smart cooler prototype consists of three major functions:

1. Humidity Control: Activates the water pump when humidity rises above the threshold, with hysteresis to avoid rapid switching.
2. User Detection & Directional Cooling: Uses an ultrasonic sensor to detect a person's presence and rotates the cooler grill via a servo motor to direct airflow.
3. Water Level Monitoring: Continuously measures the water level and indicates how much water needs to be filled for optimal operation, preventing sudden dry-outs.

All feedback is displayed on an LCD screen, making the system user-friendly and autonomous.

# Humidity control and water level monitoring

---

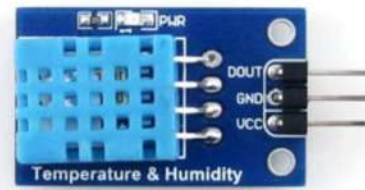
## COMPONENTS :

Microcontroller:



1.Arduino Uno

Sensors::



1. DHT Sensor



2. LCD 16x2 I<sup>2</sup>C

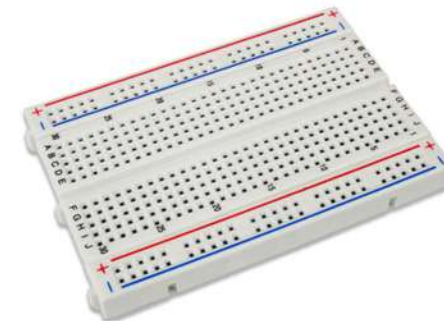
Actuators:



1.Relay module



2. Pump



3. Breadboard



4. 5V adapter



5. jJump wires

# Circuit Design

## Circuit Connections

### 1. DHT Sensor

VCC → 5V

GND → GND

DATA → Digital pin 7

### 2. I2C LCD (16x2)

VCC → 5V

GND → GND

SDA → A4

SCL → A5

### 3. Relay Module + Pump

Relay VCC → 5V

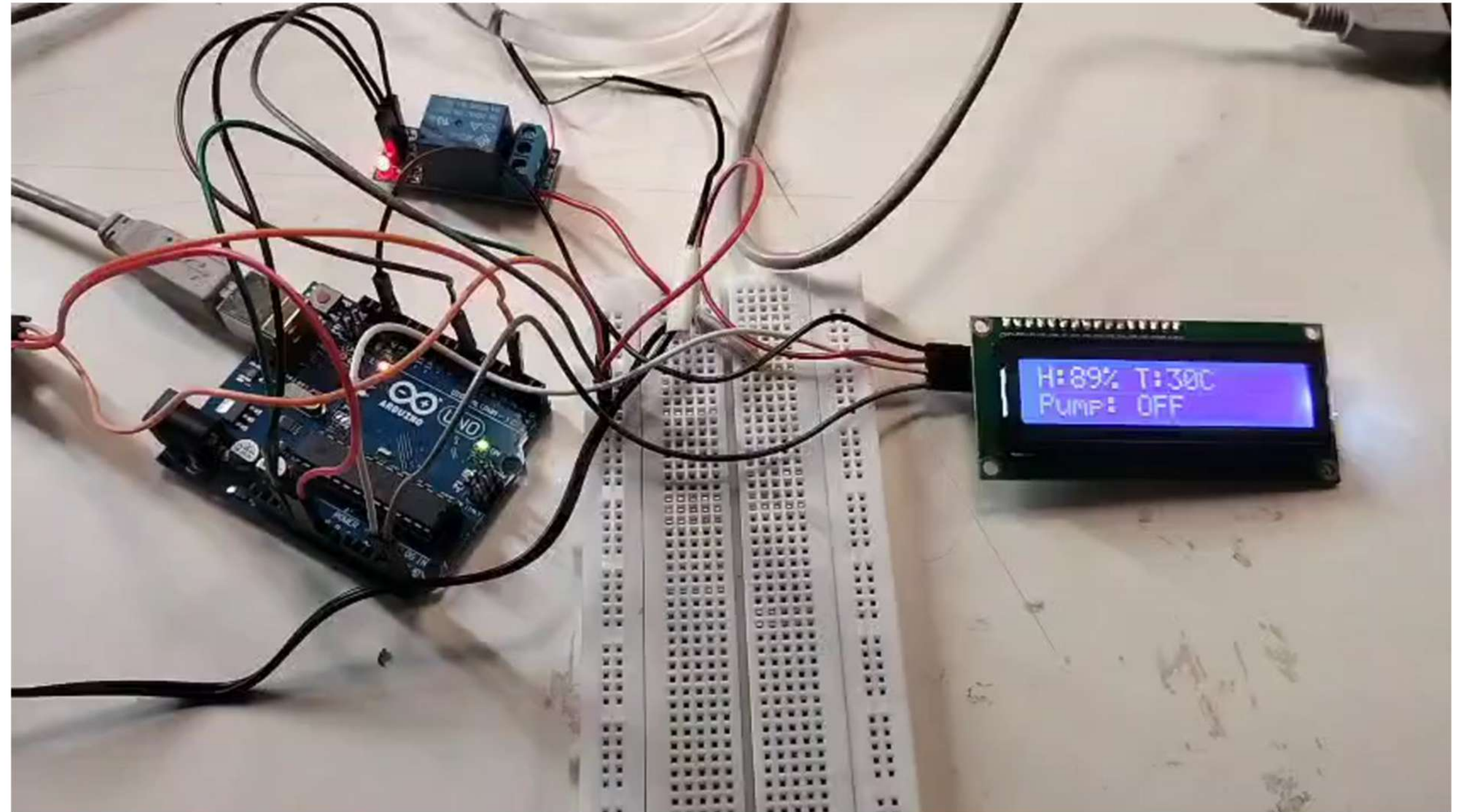
Relay GND → GND

Relay IN → Digital pin 8

Relay COM → 5V

Relay NO → Pump +ve

Pump -ve → GND



# Programming

## Key Features of the Code:

### 1. DHT11 sensor readings (humidity & temperature)

- Takes 5 readings (NUM\_READINGS) with delays in between.
- Averages them for more stable results.

### 2. Pump control with hysteresis

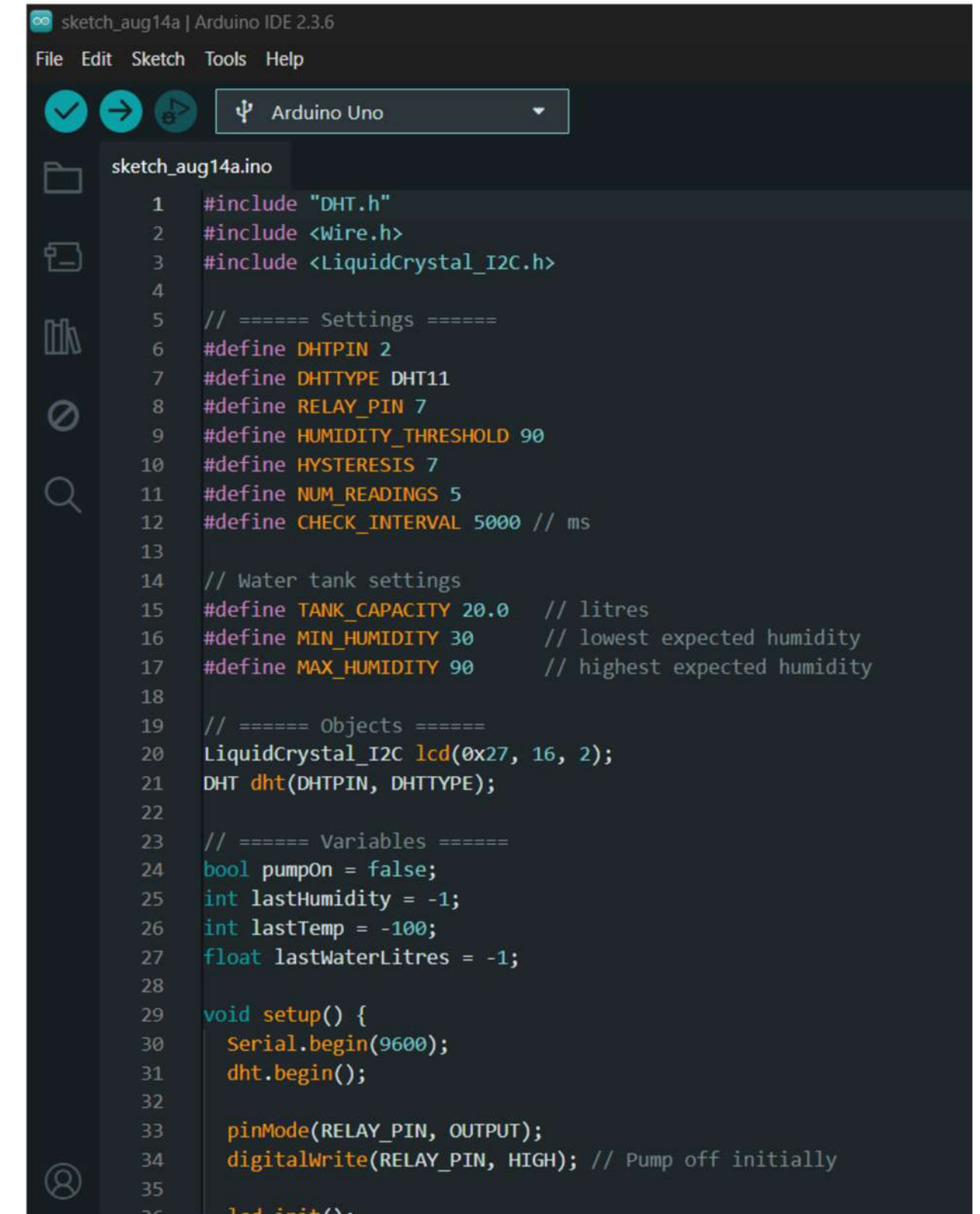
- Pump turns ON when humidity  $\leq$  (threshold - hysteresis)  $\rightarrow$  65% in your case.
- Pump turns OFF when humidity  $\geq$  threshold  $\rightarrow$  72%.
- This avoids rapid ON/OFF switching (relay chattering).

### 3. Relay handling

- Active LOW relay logic (digitalWrite(RELAY\_PIN, pumpOn ? LOW : HIGH)).
- Ensures the pump is safely off by default.

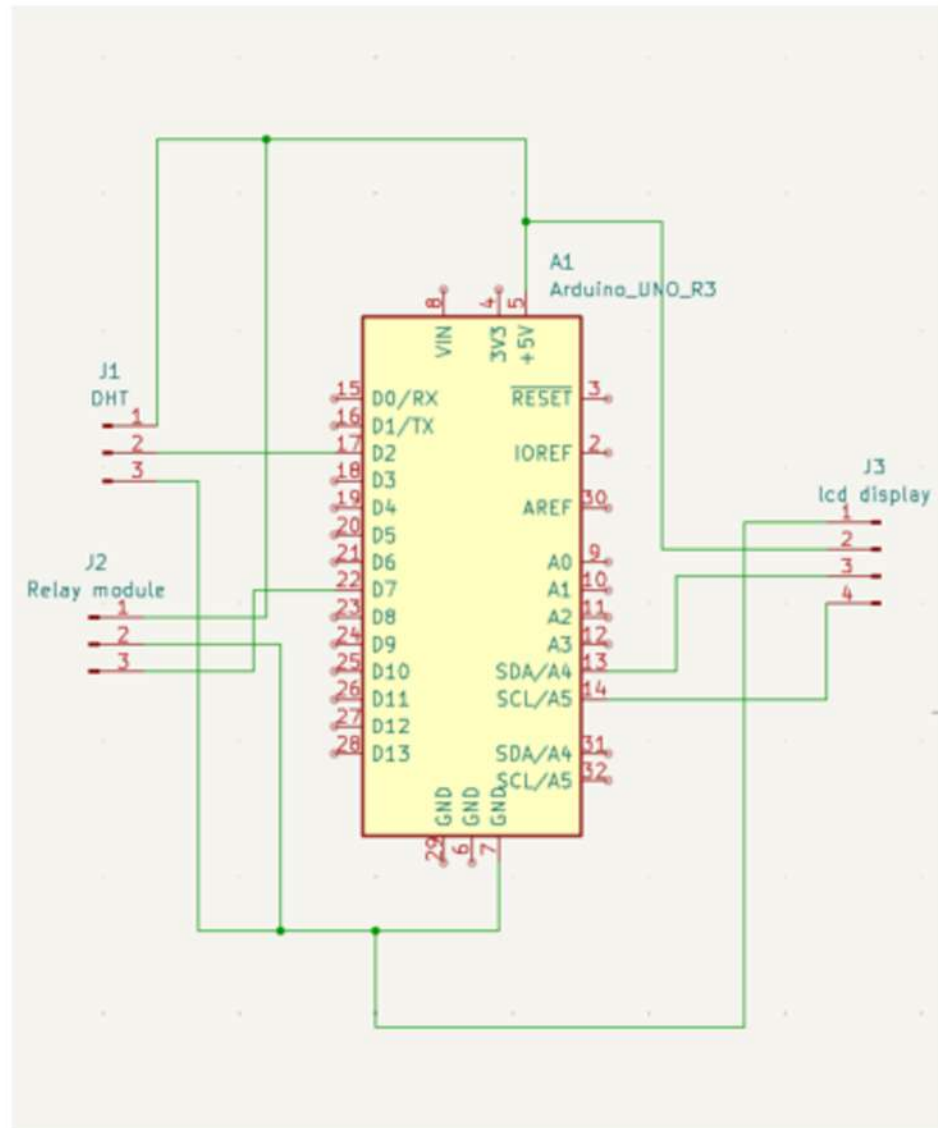
### 4. LCD display optimization

- Updates LCD only if humidity/temp changes  $\rightarrow$  prevents flickering.
- Shows Humidity, Temp, and Pump status.

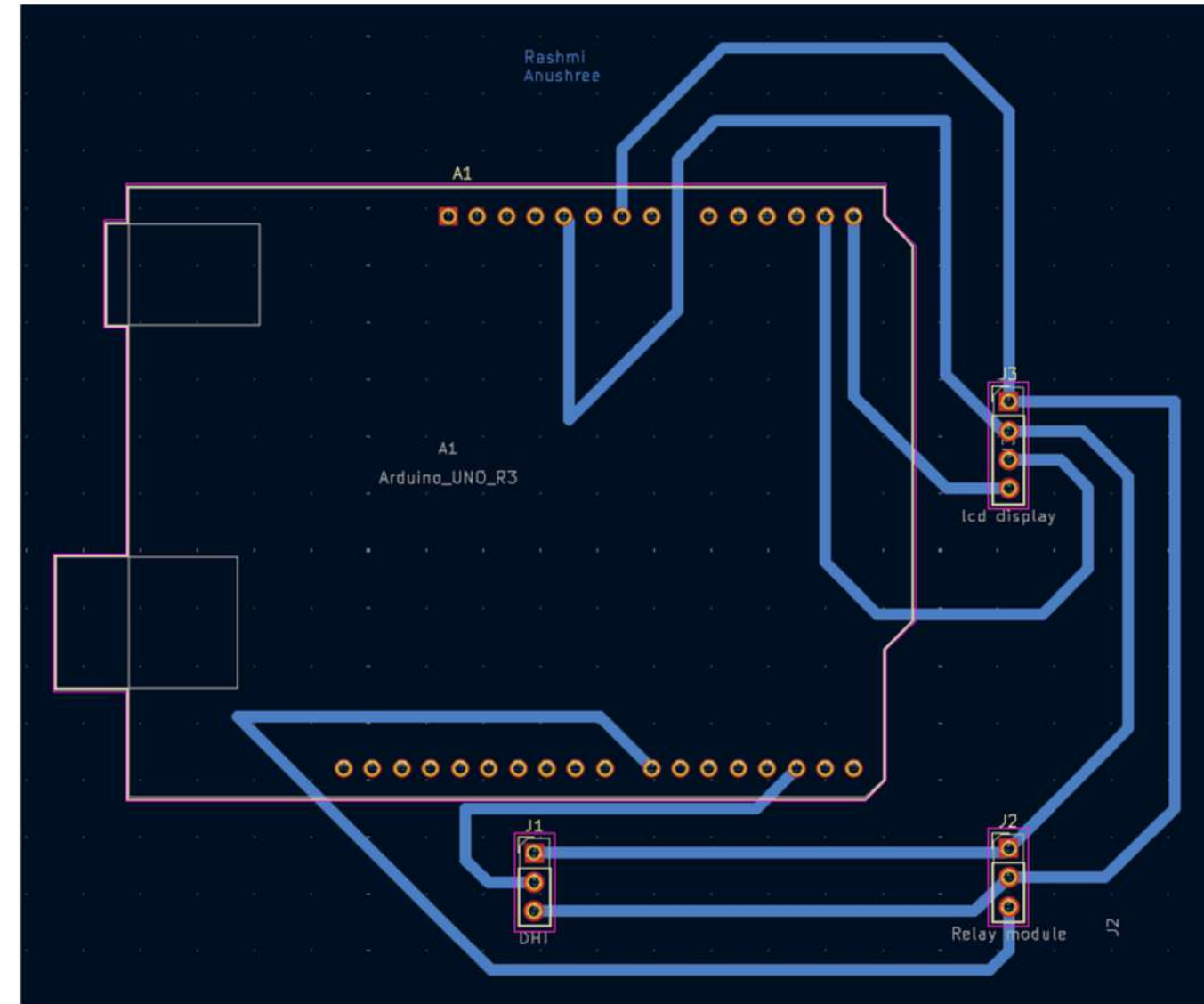


```
sketch_aug14a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
[Icons] Arduino Uno
sketch_aug14a.ino
1  #include "DHT.h"
2  #include <Wire.h>
3  #include <LiquidCrystal_I2C.h>
4
5  // ===== Settings =====
6  #define DHTPIN 2
7  #define DHTTYPE DHT11
8  #define RELAY_PIN 7
9  #define HUMIDITY_THRESHOLD 90
10 #define HYSTERESIS 7
11 #define NUM_READINGS 5
12 #define CHECK_INTERVAL 5000 // ms
13
14 // Water tank settings
15 #define TANK_CAPACITY 20.0 // litres
16 #define MIN_HUMIDITY 30 // lowest expected humidity
17 #define MAX_HUMIDITY 90 // highest expected humidity
18
19 // ===== Objects =====
20 LiquidCrystal_I2C lcd(0x27, 16, 2);
21 DHT dht(DHTPIN, DHTTYPE);
22
23 // ===== Variables =====
24 bool pumpOn = false;
25 int lastHumidity = -1;
26 int lastTemp = -100;
27 float lastWaterLitres = -1;
28
29 void setup() {
30     Serial.begin(9600);
31     dht.begin();
32
33     pinMode(RELAY_PIN, OUTPUT);
34     digitalWrite(RELAY_PIN, HIGH); // Pump off initially
35
36     lcd.init();
```

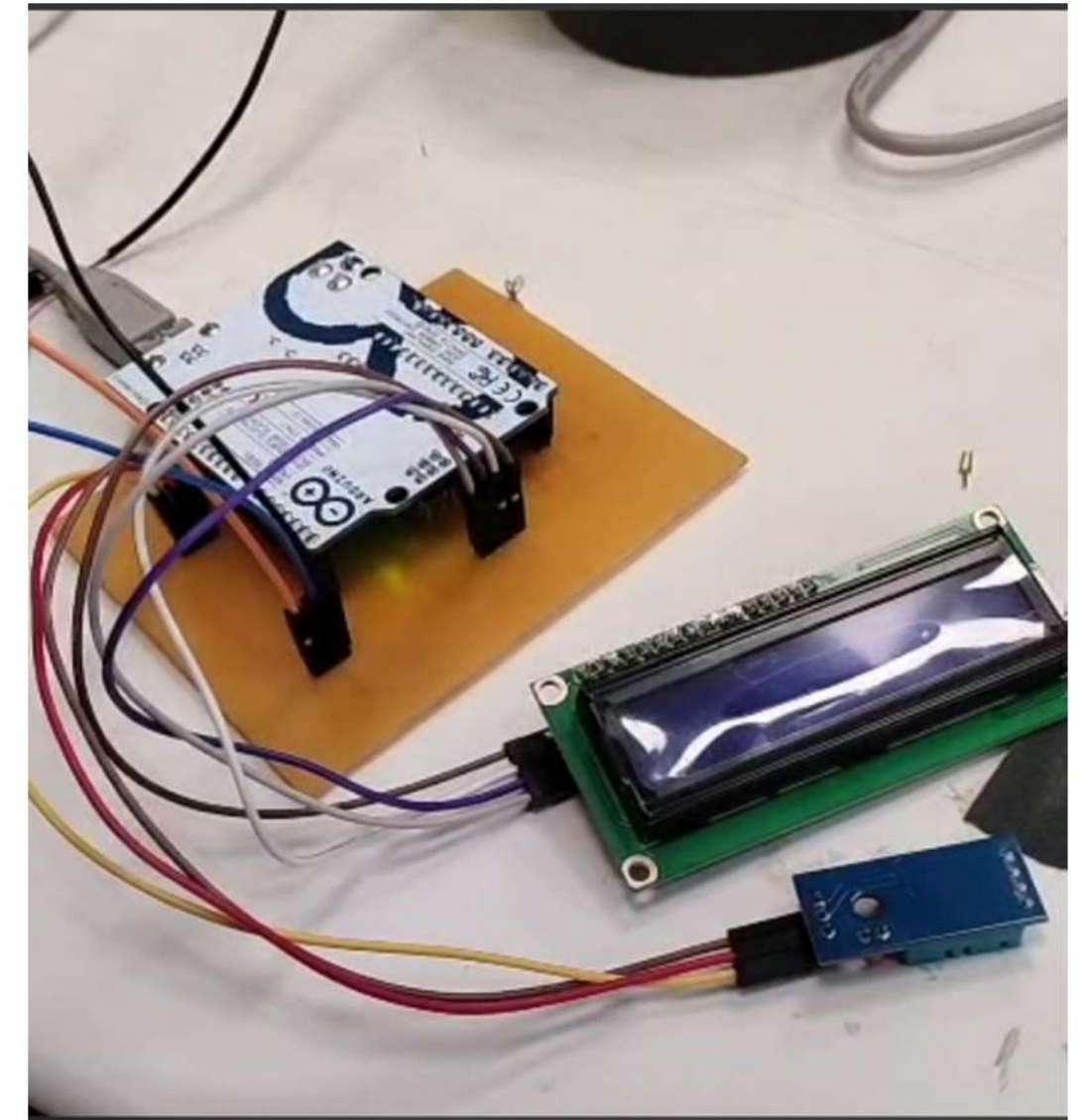
# PCB Circuit Design & Fabrication



Schematic Deisgn



PCB Layout



Assembly

# Process of Pcb making

01.



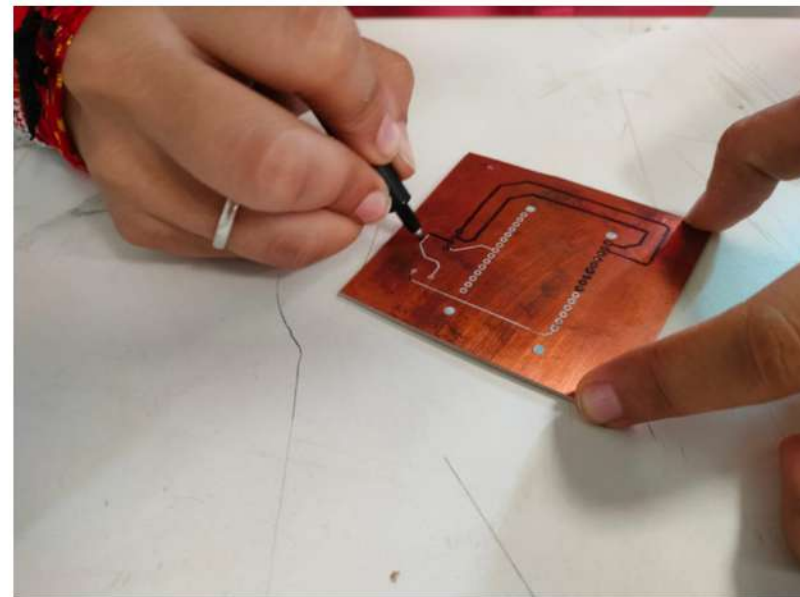
Dip the board in Ferric Chloride solution

02.



Cleaning with thinner

03.



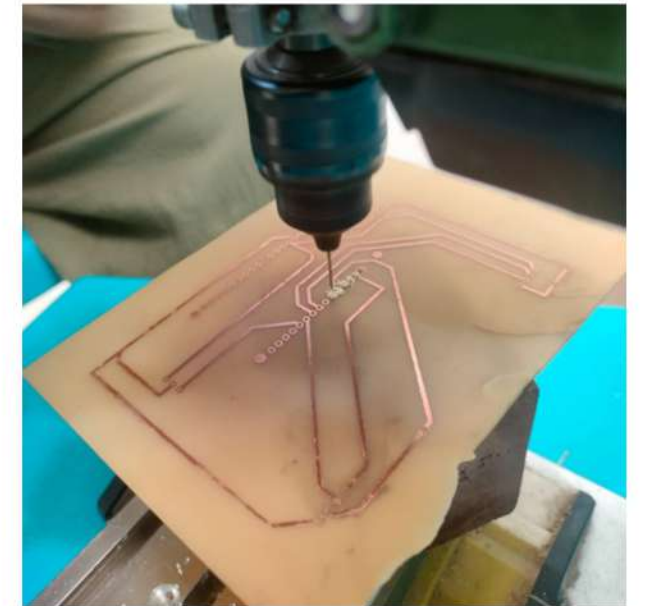
Reconnecting the broken Connection

04.



Soldering with the led wire for better connectivity

05.



Drilling Holes to mount the components

# User detection and direction

---

## COMPONENTS :

Microcontroller:



Esp 32

Sensors::

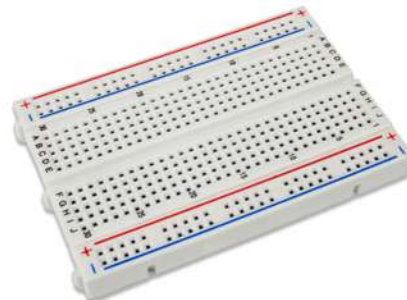


Ultrasonic Sensor

Actuators:



Servo Motor



Bread Board

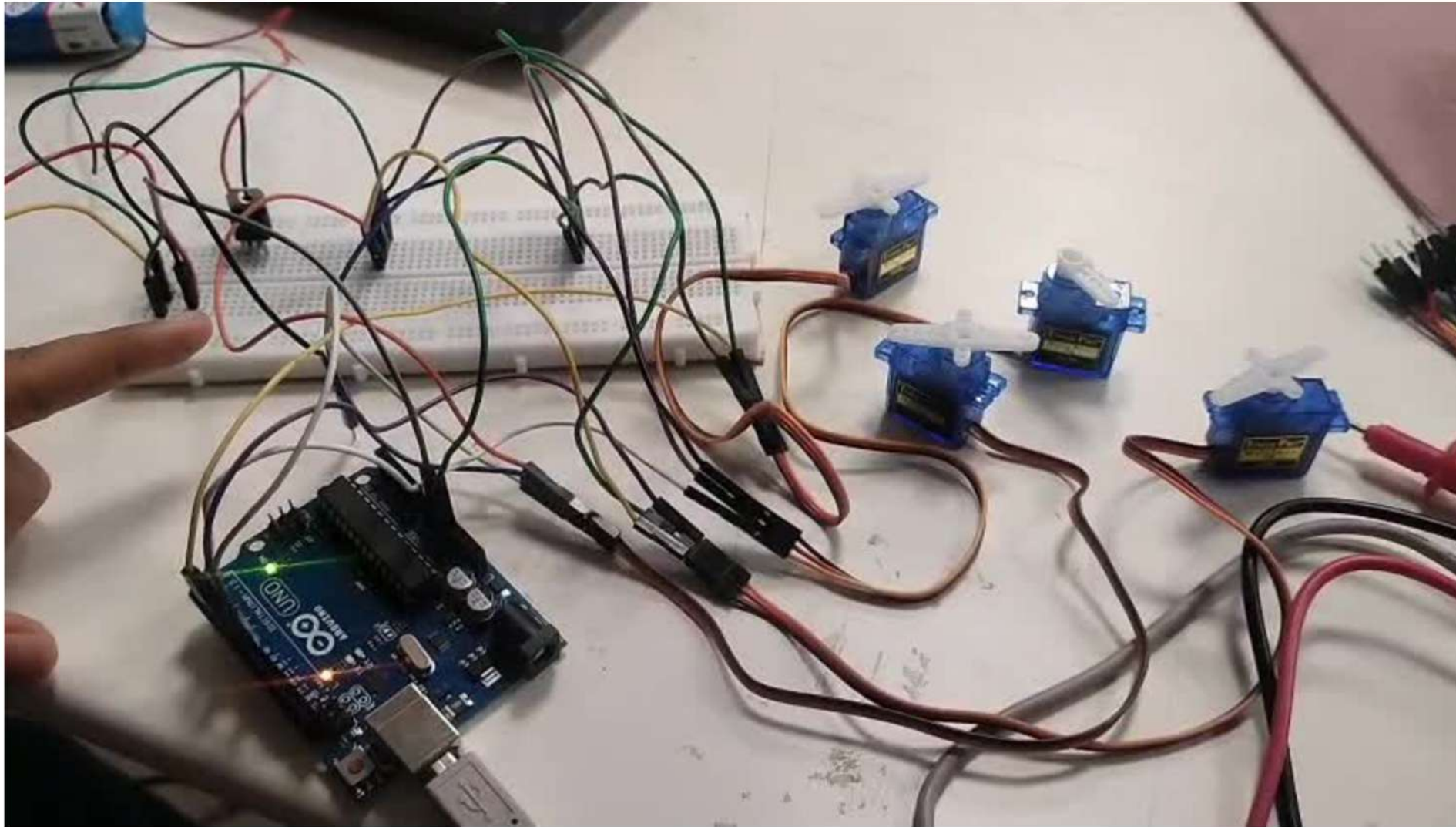


Jumper wire



5v dc Adapter

# 1<sup>st</sup> Attempt

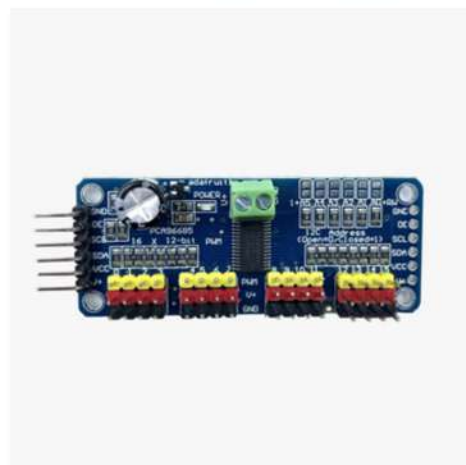


## PROBLEM

- Initially, four servos powered from a single source—only 1-2 moved.
- Added a relay driver, but issue persisted.
- Root cause: power supply couldn't deliver enough current for all servos, especially at startup.
- Relay solved control, not power deficiency.

## SOLUTION

- Replaced 9 g servos with MG996R (large) servos.
- Added 5 V adapter to supply sufficient current.



Servo Motor driver



Servo Motor



MG996R



5v dc Adapter

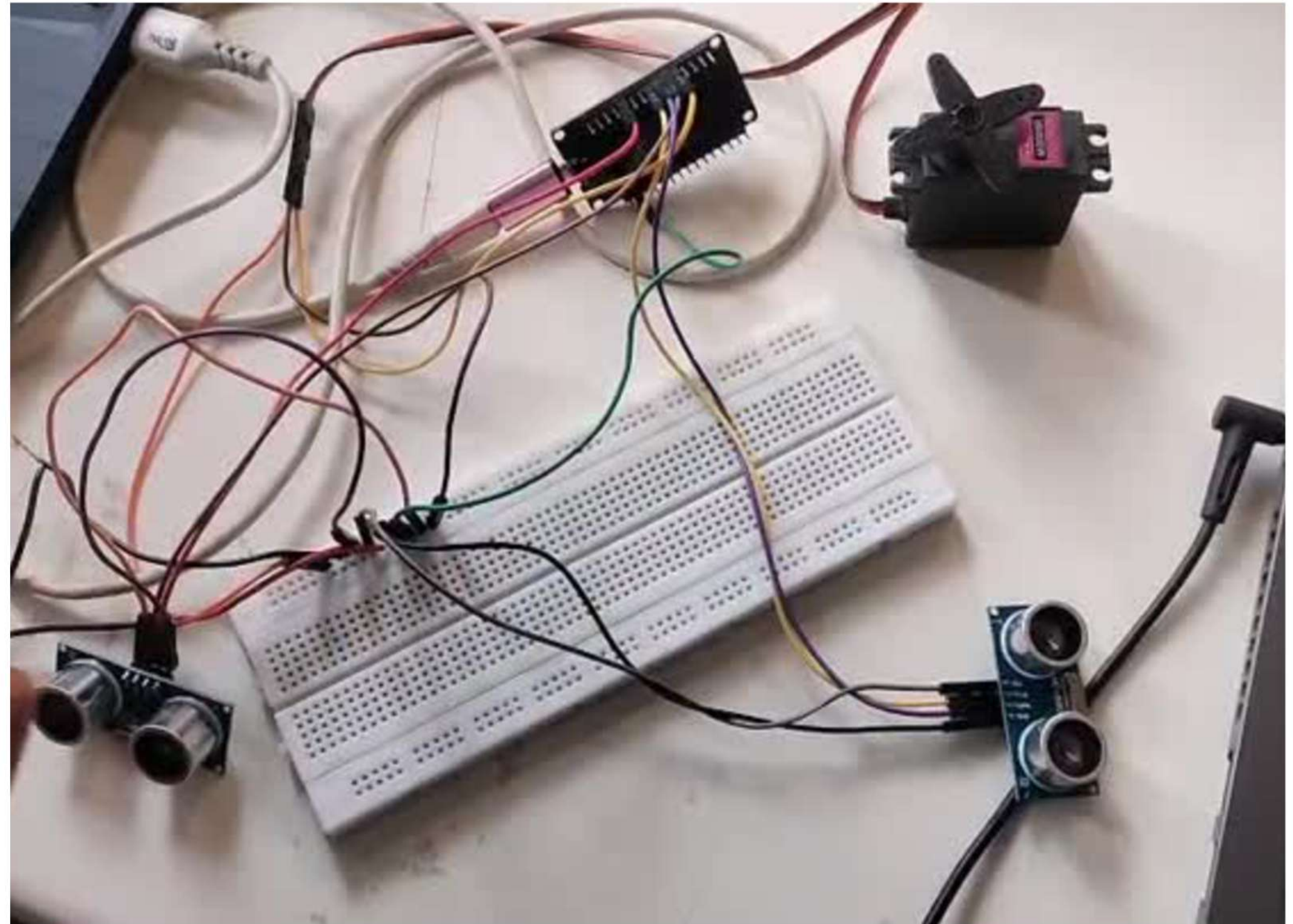
# Circuit Design

## Circuit Connections

1. Ultrasonic Sensor (HC-SR04) →  
ESP32  
VCC → 5V (from ESP32 or adapter)  
GND → GND  
Trig → GPIO 5  
Echo → GPIO 18

2. Servo Motor → ESP32  
VCC (Red) → 5V external supply  
GND (Brown/Black) → Common GND  
(ESP32 + Adapter)  
Signal (Orange/Yellow) → GPIO 19

3. Power (5V Adapter)  
Adapter +5V → Breadboard + Rail  
Adapter GND → Breadboard - Rail  
ESP32 Vin → Breadboard + Rail (if  
powering via adapter)  
ESP32 GND → Breadboard - Rail

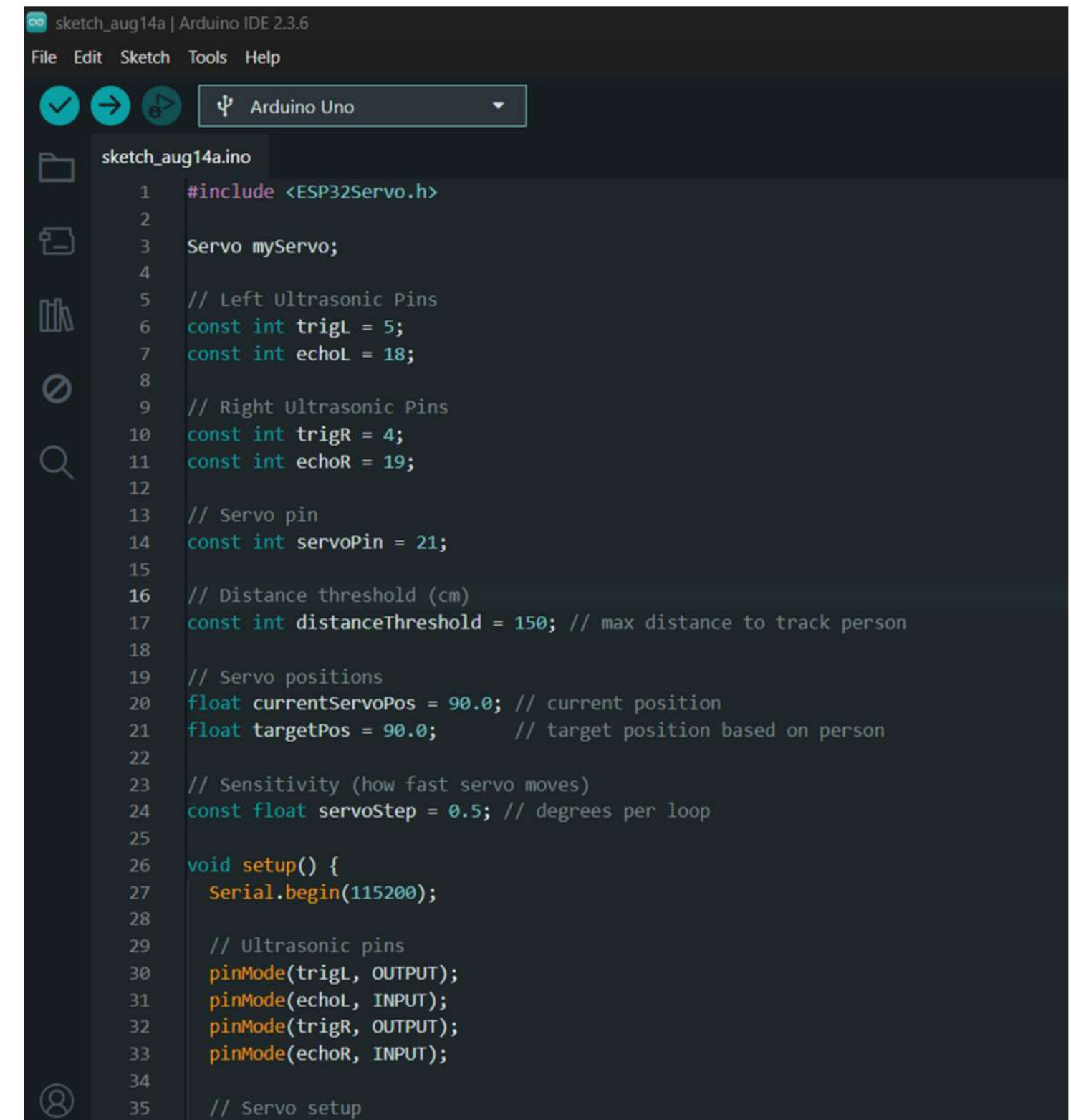


# Programming

## The key feature of code is:

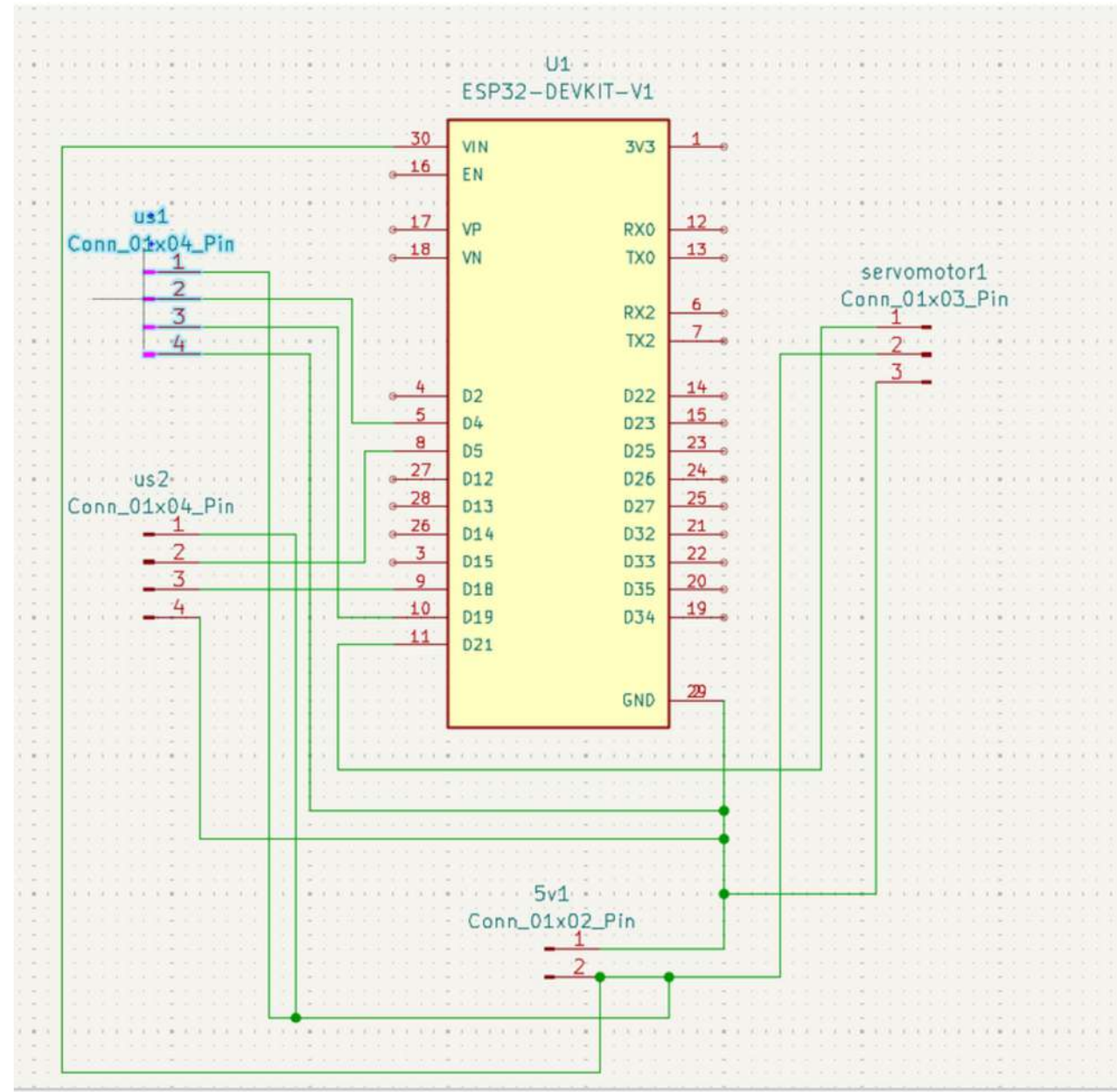
Real-time servo-based tracking of a person using two ultrasonic sensors.

1. Dual Ultrasonic Sensing – Measures distance on the left and right to detect a person's position relative to the servo.
2. Proportional Servo Control – Calculates a target servo angle based on the difference in distances, allowing the servo to “follow” the person smoothly.
3. Smooth Movement – Uses incremental steps (servoStep) to move the servo gradually, avoiding abrupt or jerky motion.
4. Distance Thresholding – Only reacts if a person is within a defined maximum distance (distanceThreshold), preventing unnecessary servo movement.

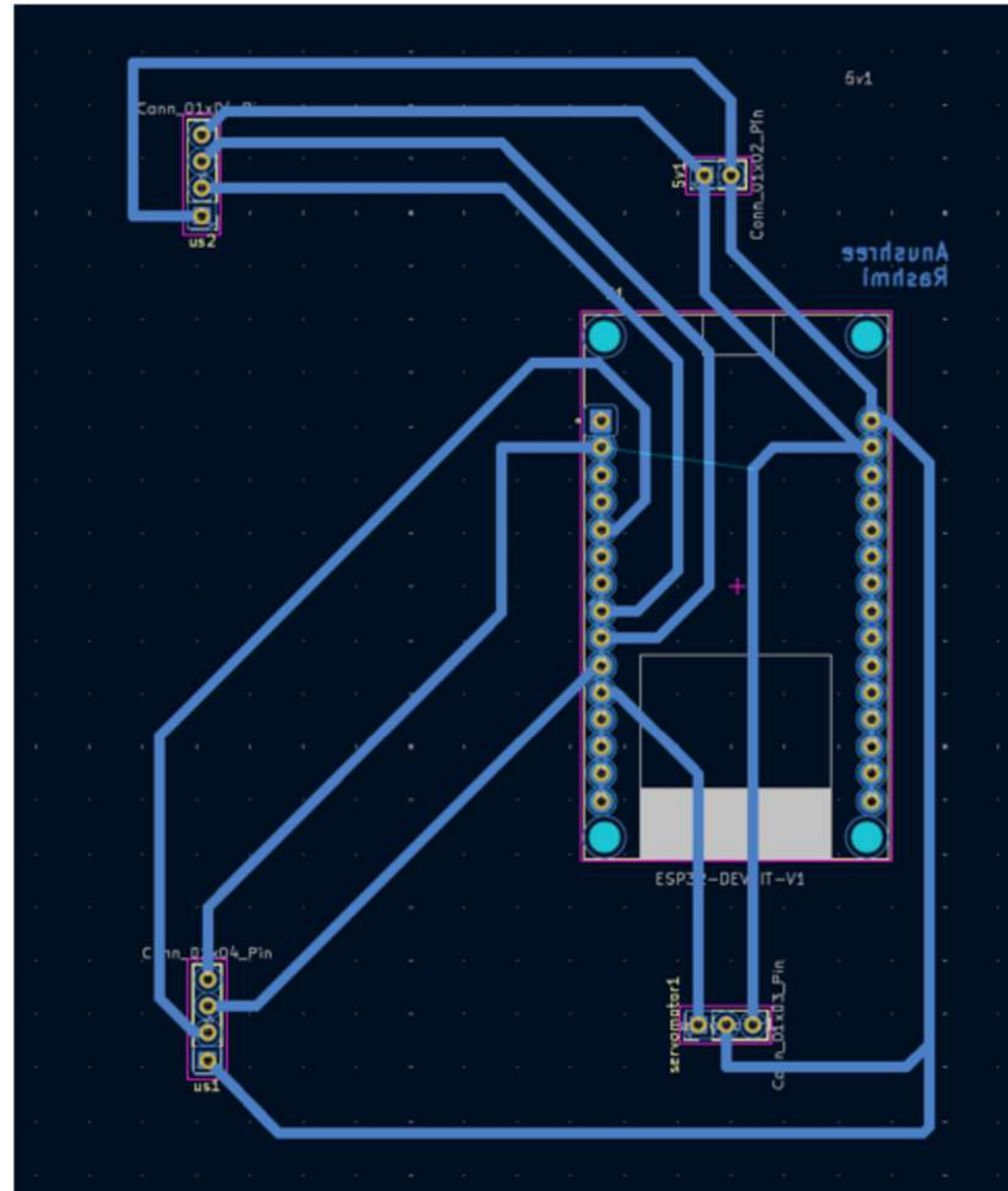


```
sketch_aug14a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
[Icons] Arduino Uno
sketch_aug14a.ino
1  #include <ESP32Servo.h>
2
3  Servo myServo;
4
5  // Left Ultrasonic Pins
6  const int trigL = 5;
7  const int echoL = 18;
8
9  // Right Ultrasonic Pins
10 const int trigR = 4;
11 const int echoR = 19;
12
13 // Servo pin
14 const int servoPin = 21;
15
16 // Distance threshold (cm)
17 const int distanceThreshold = 150; // max distance to track person
18
19 // Servo positions
20 float currentServoPos = 90.0; // current position
21 float targetPos = 90.0; // target position based on person
22
23 // Sensitivity (how fast servo moves)
24 const float servoStep = 0.5; // degrees per loop
25
26 void setup() {
27     Serial.begin(115200);
28
29     // Ultrasonic pins
30     pinMode(trigL, OUTPUT);
31     pinMode(echoL, INPUT);
32     pinMode(trigR, OUTPUT);
33     pinMode(echoR, INPUT);
34
35     // Servo setup
```

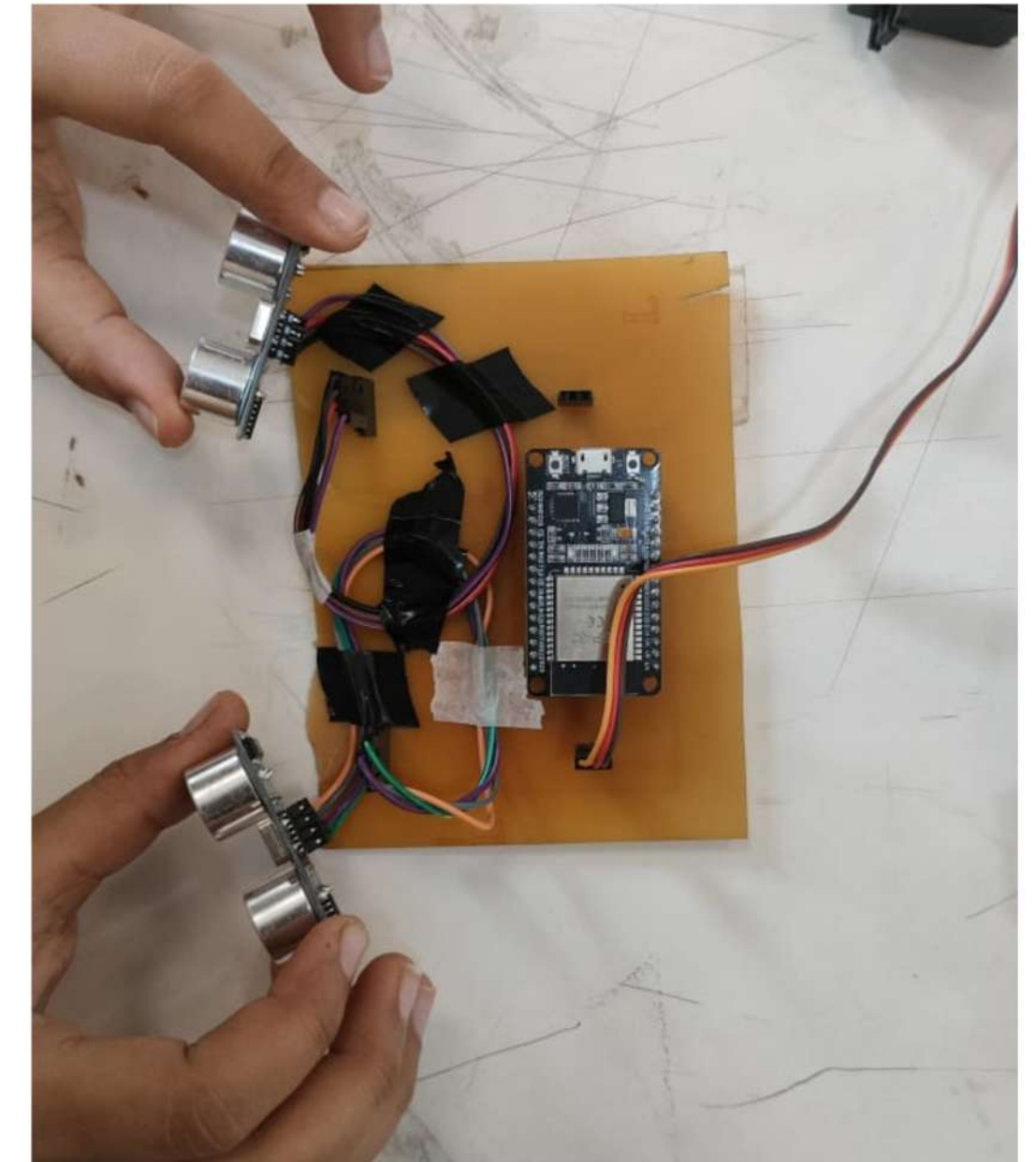
# PCB Circuit Design & Fabrication



Schematic Deisgn



PCB layout



Assembly

## App design

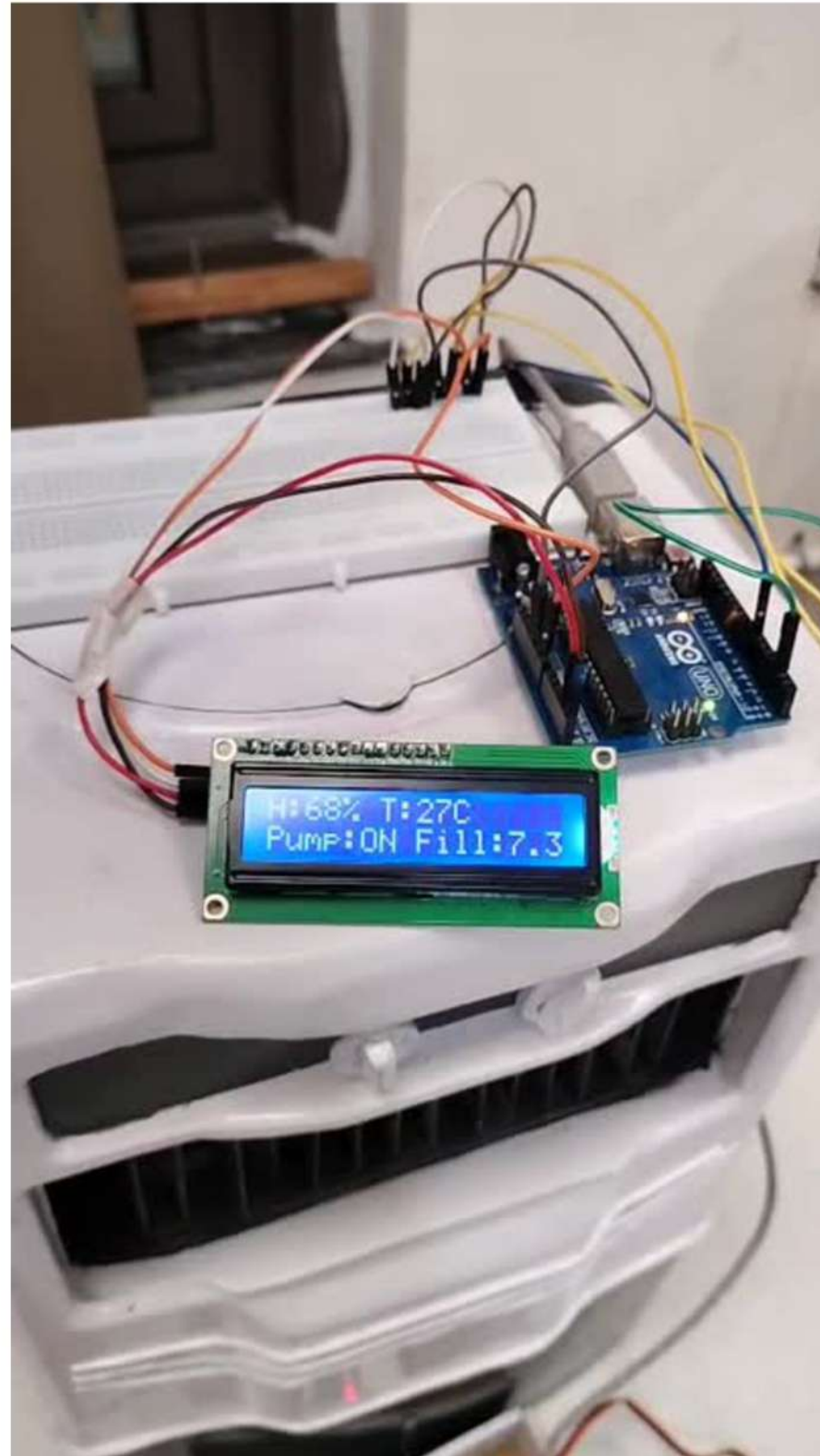
---

This app was created to automate the operation of a cooler, enabling smart control based on environmental conditions. It simplifies user interaction while optimizing energy efficiency.



# FlowIQ

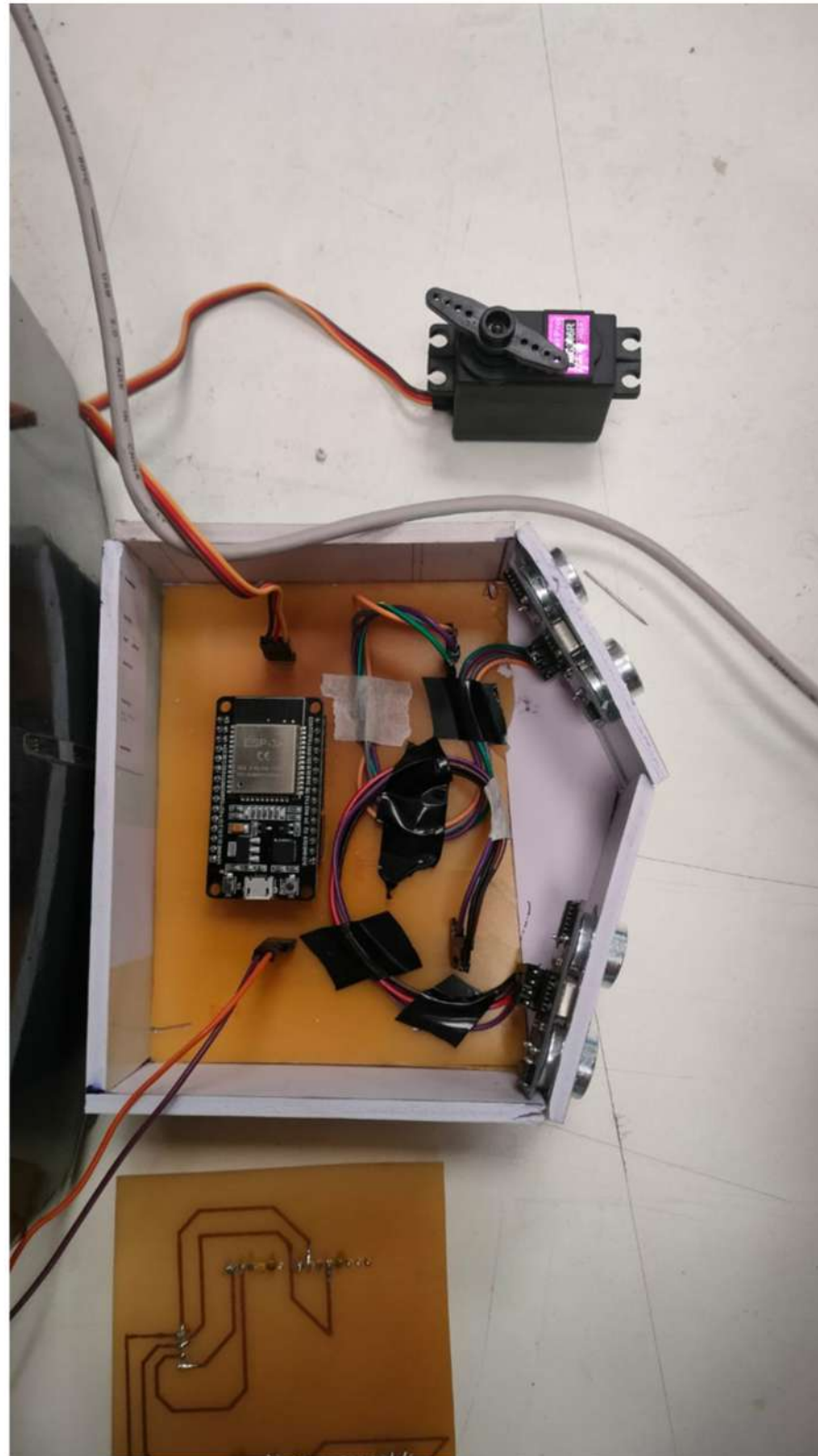
Our smart cooler redefines comfort by automatically tracking the user and angling airflow accordingly, while an integrated humidity sensor intelligently controls the water pump—turning it on only when needed and monitoring water levels to prompt refills. It ensures cooling is efficient, adaptive, and hassle-free, reducing both effort and water waste



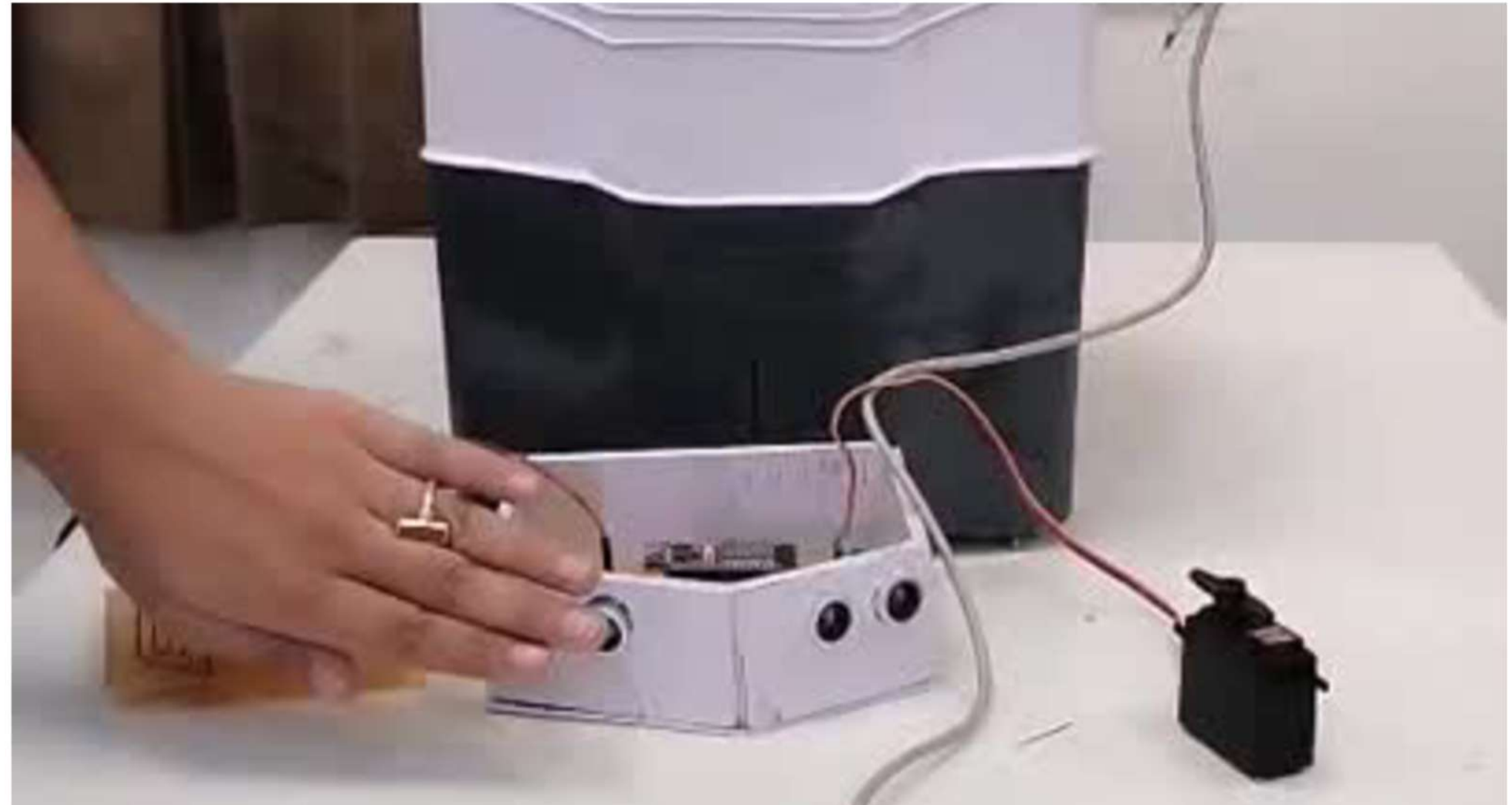
Displays real-time information from the system.

Performs action in response to sensor detection





The servo motor rotates to follow the position of a person or object, based on distance readings from the left and right ultrasonic sensors.



**THANK YOU**