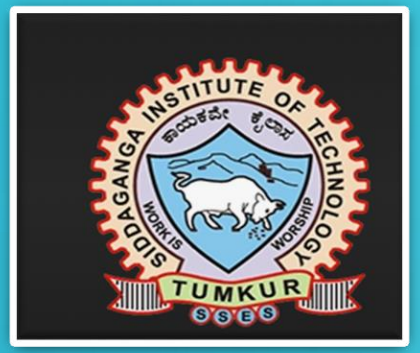


24-HOUR HAXLR8 HACKATHON

WATER-EFFICIENT IRRIGATION MANAGEMENT

❖ INDEX

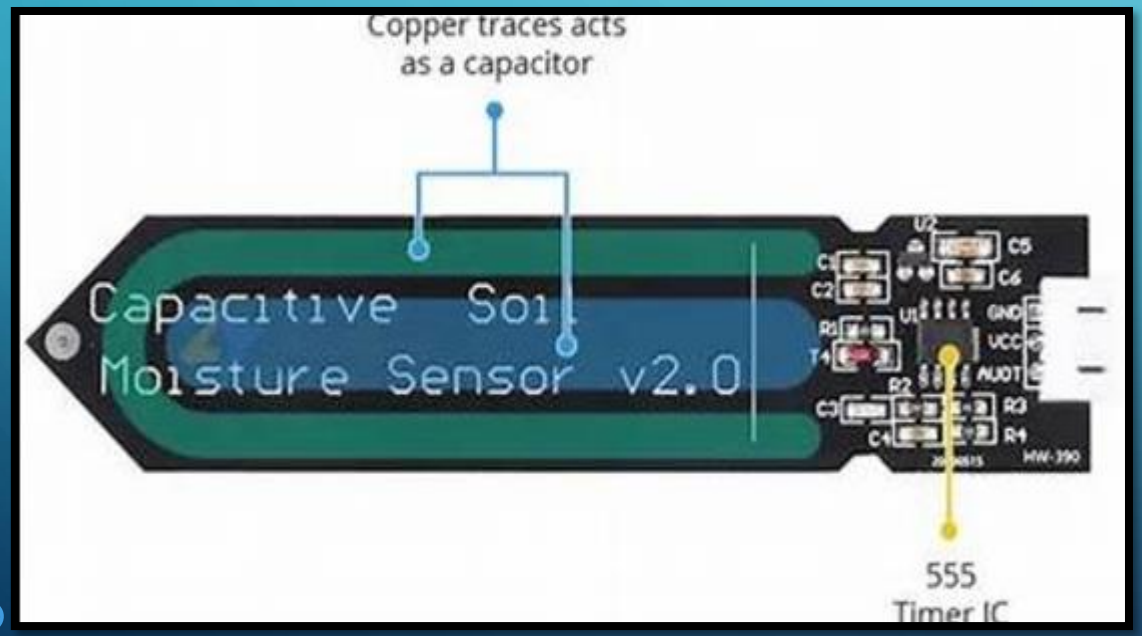
INTRODUCTION

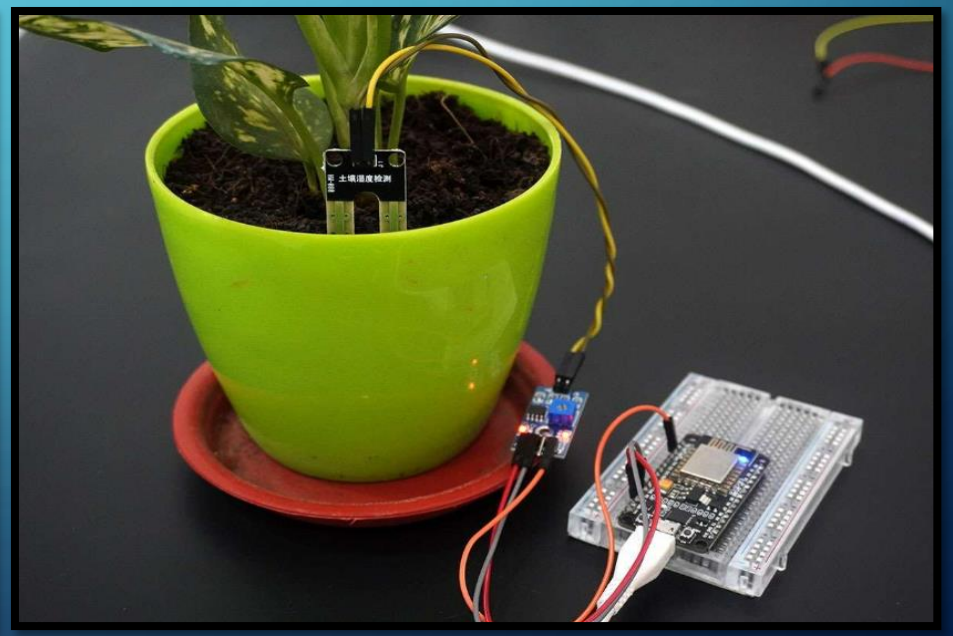
- This project presents an intelligent irrigation management system that optimizes water usage by monitoring soil moisture levels.
- The system consists of 2 Arduino-based transmitter nodes, each integrated with a moisture sensor, and a central receiver node connected to a motor and 2 solenoid valves. The transmitters wirelessly transmit soil moisture data to the receiver, which activates the motor and solenoid valves based on predefined threshold values.
- When soil moisture levels fall below the threshold, the system irrigates the land; otherwise, it remains off. The 24V motor and solenoid valves are controlled using relays, ensuring efficient water distribution to 2 separate lands. This automated system reduces water waste, conserves energy, and promotes sustainable agriculture practices.

OVERVIEW OF MOISTURE SENSORS AND THEIR FUNCTIONALITY

- **Importance of Soil Moisture Monitoring**

Accurate soil moisture monitoring is essential for optimizing irrigation practices, enhancing crop health, and promoting sustainable water usage in agriculture.





DESCRIPTION & OBJECTIVES

❖**Description**:

- This system manages irrigation based on soil moisture levels to minimize
- water waste and improve crop field.

❖**Objectives**:

- Monitor soil moisture levels in real-time.
- Automate irrigation based on threshold values.
- Optimize water usage and reduce waste.
- Promote energy efficiency and sustainable agriculture practices.

UNDERSTANDING THE COMPONENTS OF THE SYSTEM

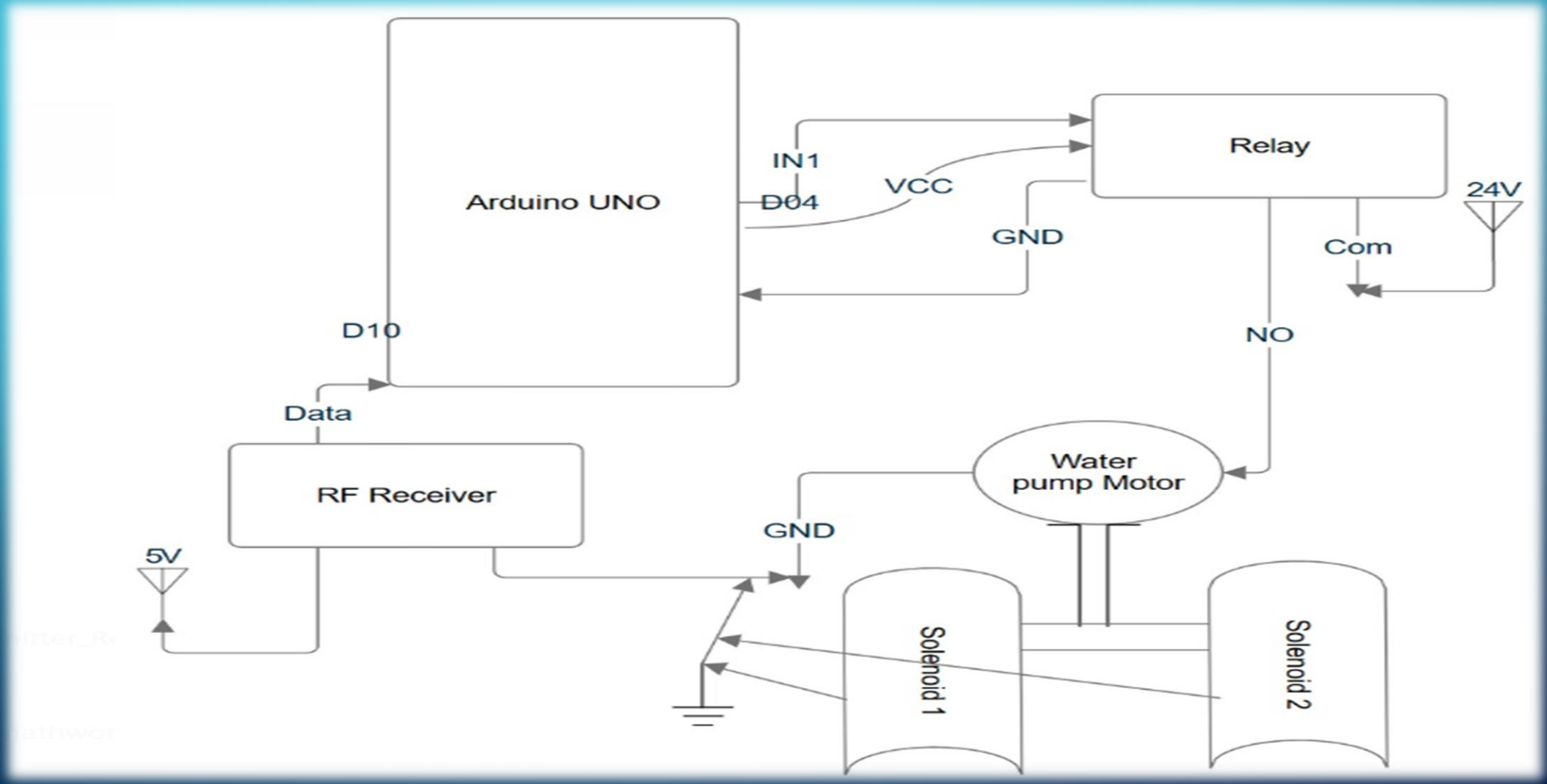
HARDWARE COMPONENTS

- Arduino uno microcontroller
- RF 433Mhz module
- 24V motor
- Solenoid valve(24v 1.4A)
- Moisture sensor
- Relay module
- 24V adapter
- LCD display
- Pipes
- Connectors

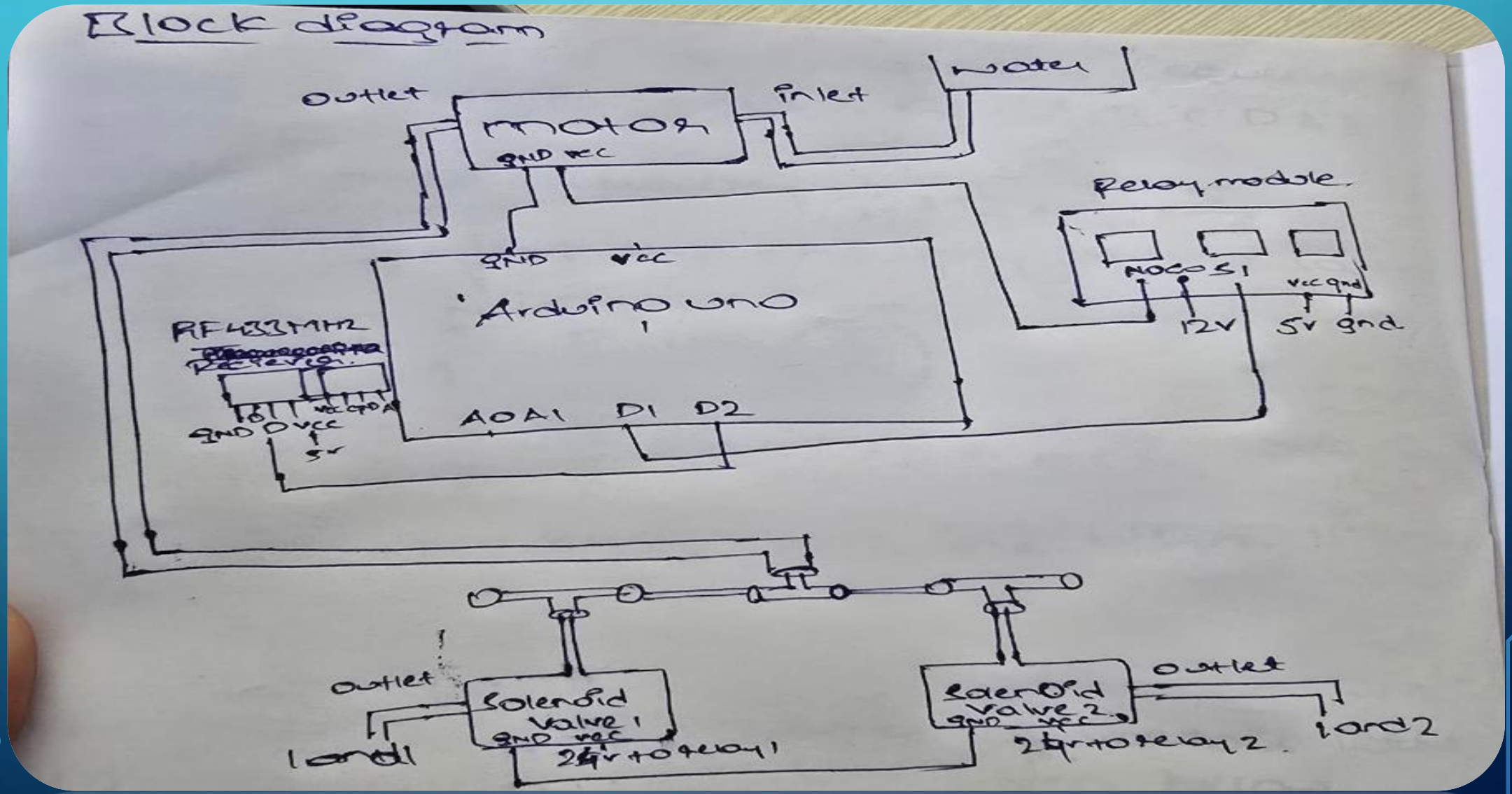
BENEFITS OF AUTOMATED IRRIGATION SYSTEMS

- **Enhanced Resource Management**
- **Improved Crop Resilience**
- **Sustainable Agricultural Practices**

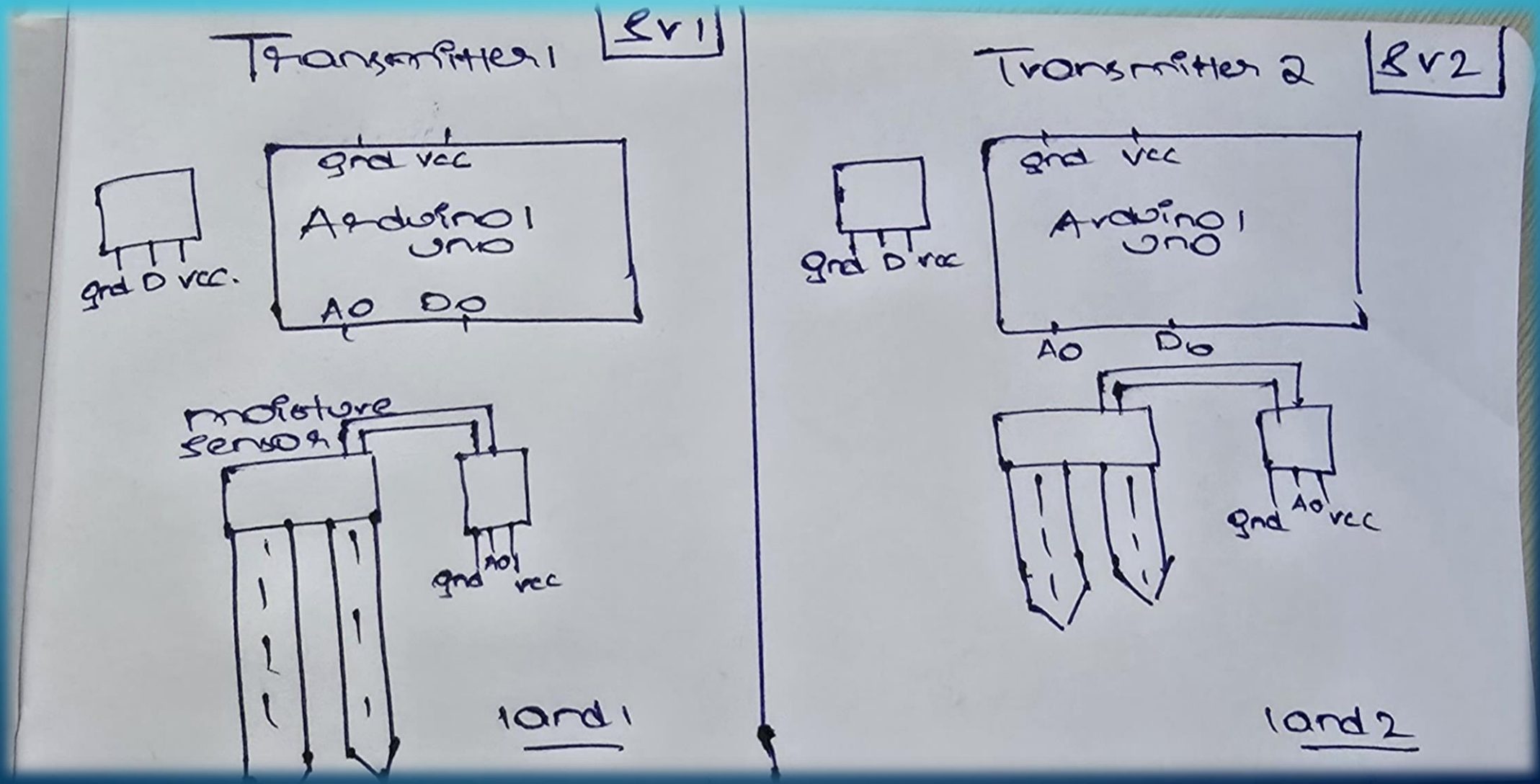
CIRCUIT DIAGRAM & CONNECTIONS-01



CIRCUIT DIAGRAM & CONNECTIONS-2



CIRCUIT DIAGRAM & CONNECTIONS-03



rftrans.ino

```
#include <SPI.h>
#include <RH_ASK.h>
// Set the transmission speed and the pin number
RH_ASK rf_driver(2000, 13);
#define motor 2
#define solen1 3
#define solen2 4

void setup() {
  Serial.begin(9600);

  // Set pins as output
  pinMode(motor, OUTPUT);
  pinMode(solen1, OUTPUT);
  pinMode(solen2, OUTPUT);
  Serial.println("Waiting for data");

  // Initialize RF driver
  if (!rf_driver.init()) {
    Serial.println("RF driver init failed");
    lcd.clear();
    lcd.print("RF init failed");
  }
}
void loop() {
  uint8_t buf[20];
  uint8_t len = sizeof(buf);
  digitalWrite(motor, HIGH);+
  digitalWrite(solen1, HIGH);
  digitalWrite(solen2, HIGH);
  // Check if a message is received
  if (rf_driver.recv(buf, &len)) {
    buf[len] = '\0';  // Null-terminate the string
    String received=String((char*)buf);
    // Debug: Print the received message
    Serial.print("Message received: ");
    Serial.println(received);
    // Process commands based on received data
    if (received=="ON1") {
      Serial.println("Turning ON solenoid 1 and motor");
      lcd.clear();
      lcd.print("ON1 - Solenoid 1");
      // Ensure solenoid 2 is OFF
      digitalWrite(solen2, HIGH);
      // Activate motor and solenoid 1
      digitalWrite(motor, LOW);     // Turn motor ON
      digitalWrite(solen1, LOW);   // Turn solenoid 1 ON
      delay(10000);                 // Keep ON for 10 seconds
      digitalWrite(solen1, HIGH); // Turn solenoid 1 OFF
      digitalWrite(motor, HIGH);  // Turn motor OFF
    }
    delay(100);
    if(received=="ON2") {
      Serial.println("Turning ON solenoid 2 and motor");
      lcd.clear();
      lcd.print("ON2 - Solenoid 2");
      // Ensure solenoid 1 is OFF
      digitalWrite(solen1, HIGH);
      // Activate motor and solenoid 2
      digitalWrite(motor, LOW);     // Turn motor ON
      digitalWrite(solen2, LOW);   // Turn solenoid 2 ON
      delay(10000);                 // Keep ON for 10 seconds
      digitalWrite(solen2, HIGH); // Turn solenoid 2 OFF
      digitalWrite(motor, HIGH);  // Turn motor OFF
    }
    else {
      Serial.println("Invalid command received");
      lcd.clear();
      lcd.print("Invalid command");
      // Reset all relays to OFF state
      digitalWrite(motor, HIGH);
      digitalWrite(solen1, HIGH);
      digitalWrite(solen2, HIGH);
    }
  }
  delay(0);
}
```

File Edit Sketch Tools Help

Arduino Nano

rfre.ino

```
#include <SPI.h>
#include <RH_ASK.h>
// Set the transmission speed and the pin number
RH_ASK rf_driver(2000, 12);
const int moisturesen=A0;

void setup() {
    Serial.begin(9600);
    if (!rf_driver.init()) {
        Serial.println("RF driver init failed");
    }
}

void loop() {
    float voltlevel=analogRead(moisturesen);
    voltlevel=voltlevel*(5.0/1023.0);
    Serial.println(voltlevel);
    float percent=(voltlevel/5)*100;
    String valuestr;
    if(percent>=75)
   {
    valuestr="ON2";
    const char *msg= valuestr.c_str();
    rf_driver.send((uint8_t *)msg, strlen(msg)); // Send the message
    rf_driver.waitPacketSent(); // Wait for the message to be sent
    Serial.print("Message sent: ");
    Serial.println(valuestr);
    delay(1);
    }
}
```