CS 362 - Milestone 8 Final Design Document

Names, NetID, Email

Mauricio Alvarez, malva30, malva30@uic.edu

Victoria Rossi, vross3, vross3@uic.edu

Hristian Tountchev, htoun2, htoun2@uic.edu

Shaan Kohli, shaank2, shaank2@uic.edu

FloraSense: The Intelligent Plant Care System

FloraSense is an intelligent plant care system using four Arduinos to monitor and automate indoor plant care. It tracks soil moisture, light, and temperature, triggering a gravity-fed drip system to water plants as needed. Data is displayed in real-time, allowing users to monitor plant health. FloraSense offers modular functionality, expandability, and precise automated watering, streamlining plant care for busy users.

Detailed Project Ideas

1. **Overall Description of Project Idea**

   FloraSense is an intelligent indoor plant care system designed to provide consistent and optimal care for plants without requiring constant attention from the user. The system monitors crucial environmental factors—soil moisture, temperature, and light exposure. This data will be used to trigger a gravity-fed system that will water the plant when needed. By dividing tasks among multiple Arduinos, FloraSense can provide consistent plant care with minimal user intervention. The system is designed to simplify plant maintenance, making it accessible even for users who may not have extensive knowledge of plant care.

2. **How Multiple Arduinos are used**

FloraSense uses four Arduinos, each performing a specialized role to ensure optimal plant care. Arduino 1 manages soil conditions by monitoring moisture and temperature, with an LED indicator for low moisture alerts. It also contains a speaker that goes off in tangent with the LED in order to audibly warn users that their plant is low on moisture. Arduino 2 tracks sunlight levels with a light sensor and displays real-time light exposure on a 16×2 LCD. Arduino 3 serves as the main user interface, it consolidates data from Arduinos 1 and 2 to determine watering needs. It will send a message to Arduino 4 indicating if the plant needs water or not. Also it will display all collected environmental data in real time for easy plant monitoring on a second 16×2 LCD. Arduino 4 will receive the data from Arduino 3, and handle the solenoid valve. Also has a button that overrides the input from Arduino 3 and will open and close the valve on command.

3. **Use and Communication Between Multiple Arduinos**

   FloraSense uses I2C communication in order to communicate the data and coordinate the four Arduinos, each one having a particular role and address on the I2C bus.

   Arduino 1 (Soil and Temperature Monitoring):

   Responsibilities:

   - Reads soil moisture and temperature data using respective sensors.
   - Triggers an LED and speaker when soil moisture is below the threshold.

   Communication:

   - Acts as an I2C slave that sends data containing soil moisture and temperature to the master (Arduino 3) when requested.

   Arduino 2 (Light Monitoring):

   Responsibilities:

- Monitors light intensity with a light sensor.
- Displays light data on a 16×2 LCD.

Communication:

- Acts as an I2C slave that sends light intensity data to Arduino 3 upon request.

Arduino 3 (Central Controller):

Responsibilities:

- Is the I2C master, requests and consolidates data from Arduinos 1 and 2.
- Analyses soil moisture, temperature, and light data to determine if watering is needed.
- Displays all environmental data (soil moisture, temperature, light) on a 16×2 LCD.
- Sends I2C commands to Arduino 4.

Communication:

- Requests data from Arduinos 1 and 2 at their assigned I2C addresses.
- Sends control signals to Arduino 4 depending on the analysis.

Arduino 4 (Watering Control):

Responsibilities:

- It controls the solenoid valve for a gravity-fed drip irrigation system.
- Provides a manual override button for user control of the solenoid valve.

Communication:

- It's an I2C slave, getting watering commands from Arduino 3 and carrying those commands out.

Communication Protocol:

I2C Master-Slave Architecture:

- Arduino 3 operates as the I2C master, while Arduinos 1, 2, and 4 are I2C slaves with unique addresses.

Data Exchange:

- Arduino 3 initiates requests for sensor data from Arduinos 1 and 2 and processes the data.
- Commands for the solenoid valve are sent to Arduino 4 via I2C.

4. **Expected Inputs/Outputs**

   FloraSense integrates updated input and output components to efficiently monitor and manage plant care. The system uses multiple Arduinos to process environmental data and automate plant care tasks.

   Inputs:

   1. **Soil Moisture Sensor** (Arduino 1): Measures soil moisture levels and provides data for watering decisions.
   2. **Temperature Sensor** (Arduino 1): Tracks the ambient temperature around the plant.
   3. **Light Sensor** (Arduino 2): Measures light intensity to assess plant exposure.
   4. **Button** (Arduino 4): Allows users to manually control the solenoid valve, overriding automated commands.

   Outputs:

   1. **LED Alert and Speaker** (Arduino 1): Activate when soil moisture falls below the threshold to alert the user.
   2. **16×2 LCD Displays**:
      - **Arduino 2**: Displays real-time light intensity data.

- **Arduino 3**: Displays consolidated environmental data, including soil moisture, temperature, and light levels, for user monitoring.

3. **Solenoid Valve** (Arduino 4): Controls water flow in the gravity-fed drip irrigation system, triggered by commands from Arduino 3 or the manual override button.

5. **Original Work**

   FloraSense is an innovative plant care system designed to automate and simplify indoor plant maintenance. Unlike traditional systems, FloraSense uses four Arduinos, each with a specific task, to monitor soil moisture, temperature, and light levels, and control watering through a gravity-fed drip system.

   Key Features Include:

   - **Multi-Arduino Design:** The tasks are distributed among the multiple Arduinos to achieve better efficiency, reliability, and scalability.
   - **Gravity-Fed Irrigation:** a simple, energy-saving alternative to automate watering.
   - **User-Friendly Features:** Real-time data display, alert for low moisture, and manual override button provide control and visibility.
   - **Customizability:** Adjustable thresholds allow the system to adapt to the different needs of plants.

   FloraSense is a modular and expandable solution, easing users' care for plants and solving many of the common issues in automated gardening.

6. **How to build the project**

   - **Libraries used:**
     - Dallas Temperature by Miles Burton
     - One Wire by Jim Studt

- **Arduino 1:**
  - **Materials:**
    - Main breadboard
    - Soil sensor
    - Temperature sensor
    - Speaker
    - LED
    - Wires
    - Arduino
    - 5K ohm resistor
    - 220 ohm resistor
  - **Wiring:**
    - Connect the temperature resistor into a breadboard.
    - DAT of the temperature resistor goes to digital pin 4.
    - GND of the temperature resistor goes to ground on the breadboard.
    - UCO of the temperature resistor goes to 5V on the breadboard and to the 5K resistor on and the other end of the resistor goes to 5V as well.
    - GND of the soil sensor goes to ground on the breadboard.
    - VCC of the soil sensor goes to 5V on the breadboard.
    - AOUT of the soil sensor goes to analog pin A0.
    - Place the LED on the breadboard. Connect the anode to digital pin 8. Connect the cathode to the 220 resistor. Connect the other end of the resistor to GND on the breadboard.
    - Connect the black wire of the speaker to GND on the breadboard. Connect the red wire to digital pin 13.

        - Connect the GND pin into the GND on the breadboard.
        - Connect the 5V pin into the positive end of the breadboard.
- **Arduino 2:**
    - **Materials:**
        - Main breadboard
        - Second breadboard
        - Photoresistor
        - LCD
        - Wires
        - 220 ohm resistor
        - 10K ohm resistor
        - Potentiometer
    - **Wiring:**
        - Connect GND to the negative end in the main breadboard.
        - Connect 5V to the positive end in the main breadboard.
        - Add the potentiometer to the second breadboard.
        - From now on, use the second breadboard.
        - Connect the middle leg of the potentiometer to V0 if the LCD. Connect the right leg to GND and the left leg to 5V.
        - Add the 220 ohm resistor connected to 5V on one end and to LCD LED+ .
        - Connect LCD LED- to GND.
        - Connect LCD GND to GND.
        - Connect LCD VCC to 5V.
        - Connect LCD RW to GND.
        - Connect LCD RS pin to digital pin 12.

        - Connect LCD Enable pin to digital pin 11.
        - Connect LCD D4 pin to digital pin 10.
        - Connect LCD D5 pin to digital pin 9.
        - Connect LCD D6 pin to digital pin 8.
        - Connect LCD D7 pin to digital pin 7.
        - Add photoresistor into C2 and C3.
        - Add 10K ohm resistor on B3 and GND.
        - Connect photoresistor (D2) into 5V.
        - Connect other end of photoresistor (A3) into the arduino analog pin A0.
- **Arduino 3:**
    - **Materials:**
        - Main breadboard
        - Secondary breadboard
        - LCD
        - Wires
        - 220 ohm resistor
        - Potentiometer
    - **Wiring:**
        - Connect GND to the negative end in the main breadboard.
        - Connect 5V to the positive end in the main breadboard.
        - Add the potentiometer to the secondary breadboard.
        - From now on, use the secondary breadboard.
        - Connect the middle leg of the potentiometer to V0 if the LCD. Connect the right leg to GND and the left leg to 5V.

        - Add the 220 ohm resistor connected to 5V on one end and to LCD LED+.
        - Connect LCD LED- to GND.
        - Connect LCD GND to GND.
        - Connect LCD VCC to 5V.
        - Connect LCD RW to GND.
        - Connect LCD RS pin to digital pin 12.
        - Connect LCD Enable pin to digital pin 11.
        - Connect LCD D4 pin to digital pin 10.
        - Connect LCD D5 pin to digital pin 9.
        - Connect LCD D6 pin to digital pin 8.
        - Connect LCD D7 pin to digital pin 7.
- **Arduino 4:**
    - **Materials:**
        - Solenoid valve
        - Relay Module
        - 1 Pushbutton
        - 10K ohm resistor
        - Power jack plug adapter (female)
        - 1/4 inch tube
        - Wires
    - **Wiring:**
        - Connect DC+ from the Relay into 5V on the breadboard.
        - Connect DC- from the Relay into GND on the breadboard.
        - Connect IN from the Relay into digital pin 5.
        - Connect ON from the Relay into + on the adapter.

        - Connect COM from the Relay with the input wire (yellow) from the valve.
        - Connect the power of the valve (blue) into the - on the adapter.
        - Connect the button to the breadboard.
        - Connect one leg of the button to digital pin 7.
        - Connect a 10k ohm resistor between the same button leg and **GND** to create a pull-down resistor, which ensures the pin reads LOW when the button isn't pressed.
        - Connect the opposite leg of the button to the **5V** rail on the breadboard.
        - Connect the tube to the solenoid valve.
    - **Serial communication between Arduinos:**
        - Connect SCL from each arduino to the breadboard on the same rail.
        - Connect SDA from each arduino to the breadboard on the same rail (different from SCL).
    - **Power and Ground Connections**
        - Connect the **GND** pin on each Arduino to the ground rail on the main breadboard.
        - Connect the **5V** pin on each Arduino to the power rail on the main breadboard.

7. **How the project is to be used**

    Setup Instructions:

    Hardware Setup:

        - Place the soil moisture sensor securely into the plant's soil near the roots.
        - Position the light sensor where it can accurately measure the light exposure for the plant.

- Ensure the temperature sensor is in the same environment as the plant.
- Connect the water reservoir to the gravity-fed drip system and attach the solenoid valve. Ensure proper alignment for smooth water flow.

Power On the System:

- Connect all Arduinos to a power source via USB or a wall adapter.
- Ensure that all components, including sensors, the LED, the speaker, and the LCDs, are properly connected and powered.

Calibrate Thresholds:

- Adjust soil moisture and light thresholds in the Arduino code or using potentiometers (if included) based on your plant's needs.
- Ensure that the system displays correct baseline data on the LCDs after startup.

Operating the System:

Monitoring Plant Health:

- View real-time soil moisture, temperature, and light intensity data on the LCD displays.
- The system will automatically water the plant when the soil moisture drops below the threshold.

Alerts:

- If soil moisture is critically low, the LED and speaker on Arduino 1 will activate, alerting you to take action if needed.

Manual Override:

- Press the manual override button on Arduino 4 to open or close the solenoid valve manually. This feature is useful if you want to water the plant manually or stop watering in emergencies.

Maintenance:

Refill the Water Reservoir:

- Regularly check the water level in the reservoir and refill it to maintain the gravity-fed drip system.

Check Sensors:

- Periodically clean the soil moisture sensor to ensure accurate readings.
- Verify that the light and temperature sensors are unobstructed and functional.

Inspect Connections:

- Ensure that all Arduino connections, wires, and components remain secure and undamaged.

Troubleshooting:

No Data Displayed on LCD:

- Check connections between the LCD and the corresponding Arduino.
- Ensure the Arduino is powered and functional.

No Watering Despite Low Soil Moisture:

- Verify that the solenoid valve is connected and powered.
- Ensure Arduino 3 is communicating properly with Arduino 4.

False Alerts from LED/Speaker:

- Check if the soil moisture sensor is properly inserted into the soil and providing accurate readings.

REQUIRED Supporting Materials

1. **Timeline of Development given in a week-by-week format of completed items**

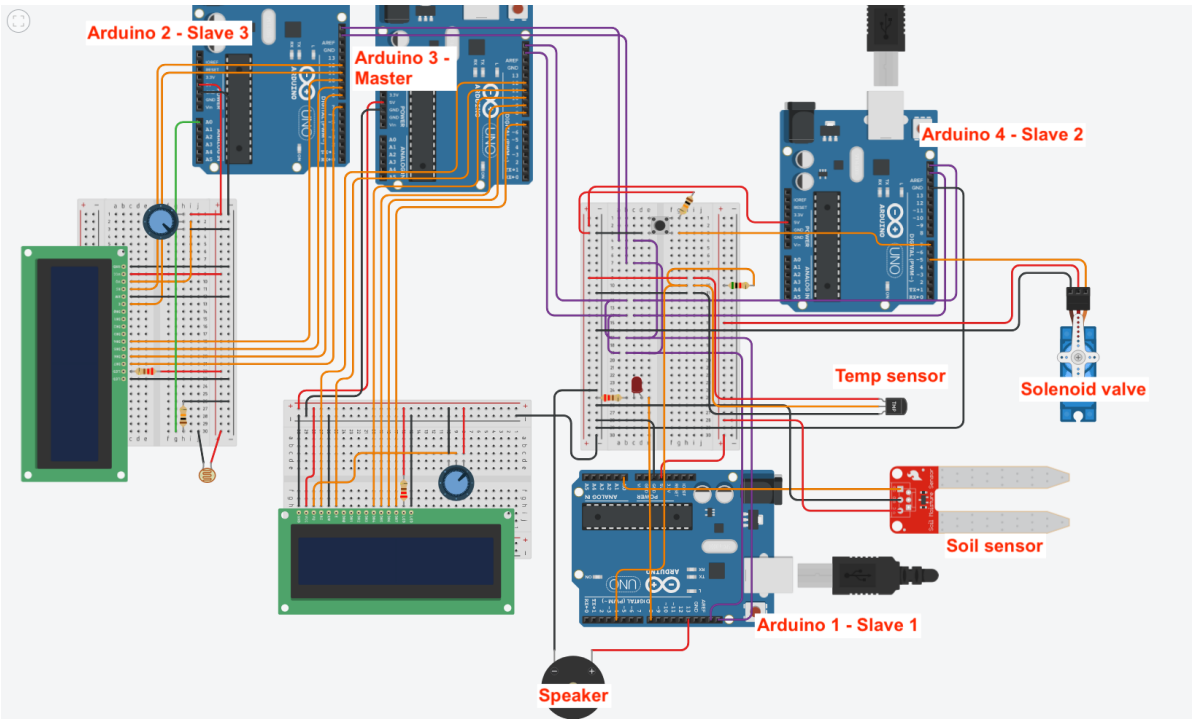
Week	Tasks
Week 1 (11/01 - 11/07)	Gathered materials, finalized the component list and assigned roles for the arduinos.

Week 2 (11/08 - 11/14)	Began hardware setup; configured individual sensors (soil moisture, temperature, light) on test Arduinos. Start basic code sketches for sensor data collection.
Week 3 (11/15 - 11/21)	Developed wired serial communication setup between 2 Arduinos and tested data transmission. Finalized the individual sensors code.
Design Presentation (11/22)	Present preliminary designs, initial data, and expected functionality.
Week 4 (11/22 - 11/28)	Integrated and tested data flow between all of the Arduinos; tested solenoid valve activation with Arduino 3 based on moisture threshold. Adjust thresholds as needed.
Week 5 (11/29 - 12/05)	Finalize integration of all components; troubleshoot system operation and data display. Complete functionality of valve with the water bottle and the valve.
Project Demonstration (12/06)	Demonstrate FloraSense’s capabilities, showing how sensors trigger the drip system.

2. **Final List of Materials Needed**
    - 4 Arduinos
    - 2 LCD’s
    - 2 Potentiometers
    - 1 Speaker
    - 1 Solenoid valve
    - 1 Relay Module
    - 1 Pushdown button
    - 1 Female power jack plug adapter
    - 1 red LED
    - 1 Temperature sensor
    - 1 Soil Temperature
    - 220 ohm resistors
    - 10K ohm resistors

- 1/4 inch tube
- 1 Photoresistor
- Plant
- Wires

3. **Final List of References**
    - LED wiring and initial code → https://docs.arduino.cc/built-in-examples/basics/Blink/
    - LCD wiring and intial code → https://docs.arduino.cc/learn/electronics/lcd-displays/
    - Speaker wiring and intial code → https://www.hackster.io/blackpanda856/play-music-using-arduino-uno-and-a-speaker-b94e4a
    - Button wiring and intial code → https://docs.arduino.cc/built-in-examples/digital/Button/
    - Photoresistor wiring and initial code → https://www.instructables.com/Light-Sensor-Photoresistor-Arduino-Tinkercad/
    - Solenoid Valve → https://arduinogetstarted.com/tutorials/arduino-water-liquid-valve
    - Temperature Sensor → https://randomnerdtutorials.com/guide-for-ds18b20-temperature-sensor-with-arduino/
    - Soil Moisture Sensor → https://www.instructables.com/Arduino-Soil-Moisture-Sensor/
4. **Inclusion of final diagrams to identify your project's hardware is REQUIRED (diagrams should use Fritzing diagrams)**



5. **Inclusion of final code sketches to identify your project's software is REQUIRED**
   (include all of the .ino code in the PDF document)

   Arduino 1 Code - Slave 1

```
#include <Wire.h> // For I2C communication
#include <OneWire.h> // For interfacing with 1-Wire devices
#include <DallasTemperature.h> // For temperature sensors
#include <stdint.h>

#define ONE_WIRE_BUS 4 // Pin for 1-Wire temperature sensor
#define LED_PIN 8 // Pin connected to LED

OneWire oneWire(ONE_WIRE_BUS); // Initializes OneWire instanc
DallasTemperature sensors(&oneWire); // Initializes DallasTer

volatile float temperature = 0.0; // Store temperature value
volatile uint16_t sensorValue = 0; // Store moisture sensor v
int sensorPin = A0; // Analog pin for moisture sensor
```

```
const int speakerPin = 13; // Pin connected to speaker
const int toneFreq = 1000; // Frequency for tone
const int toneDur = 500; // Duration of tone

void setup() {
  Serial.begin(9600);
  Wire.begin(8);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  Wire.onRequest(requestEvent);
  sensors.begin();
  pinMode(speakerPin, OUTPUT);
}

void loop() {
  // Request temperature readings anmd get first reading in F
  sensors.requestTemperatures();
  temperature = sensors.getTempFByIndex(0);
  sensorValue = analogRead(sensorPin);

  // Debugging statement
  Serial.print("Moisture Value: ");
  Serial.println(sensorValue);
  Serial.print("Raw Temp: ");
  Serial.println(temperature);

  // Checks if moisture is low
  if(sensorValue > 400) {
    digitalWrite(LED_PIN, HIGH);
    tone(speakerPin, toneFreq, toneDur);
  }
  // Moisture is good
  else {
    digitalWrite(LED_PIN, LOW);
    noTone(13);
  }
```

```
}

void requestEvent() {
  Serial.println("Entering requestEvent");
  // Store data to send I2C
  uint8_t dataBuffer[6];

  // Copies temperature and sensor value to buffer and writes
  memcpy(dataBuffer, &temperature, sizeof(temperature));
  memcpy(dataBuffer + sizeof(temperature), &sensorValue, size
  Wire.write(dataBuffer, sizeof(dataBuffer));
  Serial.println("Sent");
}
```

Arduino 2 Code - Slave 3

```
#include <LiquidCrystal.h>
#include <Wire.h>

// Pin configuration for the photoresistor and LCD
int photoPin = A0; // Photoresistor is connected to analog pi
const int rs = 12, en = 11, d4 = 10, d5 = 9, d6 = 8, d7 = 7;

// Initialize the LCD (16x2) with the appropriate pins
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// Variables for light sensor and time tracking
int lightRaw; // Stores the raw reading from the photoresisto
long previousMillis = 0; // Tracks time for display updates
const long interval = 1000; // Interval to update display (1
const int MASTER_ADDRESS = 7; // Master device I2C address

void setup() {
  // Initialize the LCD
  lcd.begin(16, 2);
```

```
  // Start serial communication for debugging
  Serial.begin(9600);

  // Initialize I2C as a slave device
  Wire.begin(10); // Set slave address to 10

  // Define the behavior when the master requests data
  Wire.onRequest(requestEvent);
}

void loop() {
  // Read the value from the photoresistor
  lightRaw = analogRead(photoPin);

  // Debugging: Print the raw light value to the Serial monit
  Serial.print("Light raw value: ");
  Serial.println(lightRaw);

  // Display the light level text on the LCD
  displayLightLevel(lightRaw);

}

// Function to display the light level on the first line of t
void displayLightLevel(int lightRaw) {
  lcd.setCursor(0, 0); // Set cursor to the top line of the l

  // Determine the light level based on thresholds and displa
  if (lightRaw <= 200) {
    lcd.print("Dark           "); // Pad with spaces to clear
  } else if (lightRaw > 200 && lightRaw <= 400) {
    lcd.print("Partially Dark ");
  } else if (lightRaw > 400 && lightRaw <= 600) {
    lcd.print("Medium         ");
  } else if (lightRaw > 600 && lightRaw <= 720) {
    lcd.print("Fully Lit      ");
```

```
  } else {
    lcd.print("Brightly Lit   ");
  }
}

// I2C request event handler: sends the lightRaw value to the
void requestEvent() {
  Serial.println("Got request"); // Debugging: log that a req

  uint8_t dataBuffer[sizeof(lightRaw)]; // Create a buffer to
  memcpy(dataBuffer, &lightRaw, sizeof(lightRaw)); // Copy li

  Wire.write(dataBuffer, sizeof(dataBuffer)); // Send the buf

  // Debugging: confirm the sent value
  Serial.print("Sent lightRaw value: ");
  Serial.println(lightRaw);
}
```

Arduino 3 Code → master arduino

```
#include <Wire.h>                  // Include Wire library fo
#include <LiquidCrystal.h>         // Include LiquidCrystal l
#include <stdint.h>                // Include standard intege

// Initialize LCD with pin connections
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);

// Define slave device addresses
const uint8_t SENSOR_SLAVE_ADDRESS = 8;
const uint8_t RELAY_SLAVE_ADDRESS = 9;
const uint8_t PHOTORESISTOR_ADRESS = 10;

// Define commands for relay control
volatile const char OPEN_VALVE = 'O';
volatile const char CLOSE_VALVE = 'C';
```

```
// Initialize sensor variables
float temperature = 0.0;           // Variable to store tempe
uint16_t soilMoisture = 0;         // Variable to store soil
int photoresistor = 0;             // Variable to store photo

void setup() {
  Wire.begin(7);                  // Initialize I2C with addr
  lcd.begin(16, 2);               // Set up the LCD with 16 c
  Serial.begin(9600);             // Initialize serial commun
}

void loop() {
  // Define the number of bytes to request from the sensor si
  const uint8_t numBytes = (uint8_t)(sizeof(float) + sizeof(u

  // Request temperature and soil moisture data from the sens
  Wire.requestFrom((uint8_t)SENSOR_SLAVE_ADDRESS, (uint8_t)nu

  uint8_t dataBuffer[6];          // Buffer to store received
  uint8_t index = 0;              // Index for filling the bu

  // Read data into the buffer while available and within the
  while (Wire.available() && index < numBytes) {
    dataBuffer[index++] = Wire.read();
  }

  // Extract temperature and soil moisture values from the da
  memcpy(&temperature, dataBuffer, sizeof(temperature));
  memcpy(&soilMoisture, dataBuffer + sizeof(temperature), siz

  // Print temperature and soil moisture readings to the seri
  Serial.print("Received Temp: ");
  Serial.println(temperature);

  Serial.print("Received Moisture: ");
```

```
  Serial.println(soilMoisture);

  // Display temperature and soil moisture readings on the LC
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("T:");
  lcd.print(temperature, 2);
  lcd.setCursor(0, 1);
  lcd.print("M:");
  lcd.print(soilMoisture);

  // Request light level data from the photoresistor slave
  Wire.requestFrom((uint8_t)PHOTORESISTOR_ADRESS, (uint8_t)si

  // Check if enough data is available to process
  if (Wire.available() >= sizeof(int)) {
    uint8_t dataBufferPhotoresistor[sizeof(int)];  // Buffer
    for (uint8_t i = 0; i < sizeof(int); i++) {
      dataBufferPhotoresistor[i] = Wire.read();
    }

    // Extract photoresistor value from the data buffer
    memcpy(&photoresistor, dataBufferPhotoresistor, sizeof(ph

    // Print photoresistor value to the serial monitor
    Serial.print("Photoresistor: ");
    Serial.println(photoresistor);
  }

  // Begin communication with the relay slave
  Wire.beginTransmission(RELAY_SLAVE_ADDRESS);

  // Check if watering is needed and send the appropriate com
  if (checkWatering()) {
    Wire.write(OPEN_VALVE);       // Command to open the valv
    Serial.println("Command Sent: Open Valve");
```

```
  } else {
    Wire.write(CLOSE_VALVE);      // Command to close the val
    Serial.println("Command Sent: Close Valve");
  }

  Wire.endTransmission();        // End communication with th
}

// Function to determine if watering is needed
bool checkWatering() {
  int soil_threshold = 400;      // Default soil moisture thr

  // Adjust threshold based on light and temperature conditio
  if (photoresistor > 700 && temperature > 80) {
    soil_threshold -= 20;        // Lower threshold for high
  } else if (photoresistor < 300 && temperature < 65) {
    soil_threshold += 20;        // Raise threshold for low l
  }

  // Return true if soil moisture exceeds the threshold, fals
  return soilMoisture > soil_threshold;
}
```

Arduino 4 Code - Slave 2

```
#include <Wire.h>

// Pin and variable definitions
const int RELAY_PIN = 5;  // Relay connected to digital pin D
char receivedCommand = 'N'; // Holds the command received fro
const int interval = 5000;  // Interval for timing-related fe
unsigned long prevMilli = 0; // Previous time marker for timi
const int Button = 7;  // Button connected to digital pin D7
int lastButtonState = HIGH; // Last known state of the buttor
bool relayState = LOW; // Current state of the relay (LOW = o
long lastDebounceTime = 0; // Last time the button state char
```

```
const long debounceDelay = 50; // Debounce delay to avoid fal
bool close = true; // Tracks whether the relay should close (
bool readCommand = true; // Determines if the relay should ac

void setup() {
  pinMode(RELAY_PIN, OUTPUT); // Set relay pin as output
  digitalWrite(RELAY_PIN, LOW); // Ensure the relay is initia
  Wire.begin(9); // Initialize the I2C slave with address 9
  Serial.begin(9600); // Start Serial communication for debug
  pinMode(Button, INPUT_PULLUP); // Configure button pin with
}

void loop() {
  // Get the current time in milliseconds
  unsigned long currMilli = millis();

  // Read the current state of the button
  int buttonState = digitalRead(Button);

  // Request a command from the master device
  const uint8_t numBytes = (uint8_t)(sizeof(char)); // Number
  Wire.requestFrom((uint8_t)7, numBytes); // Request data fro
  while (Wire.available()) {
    Serial.println("Wire available"); // Debugging: confirm d
    receivedCommand = Wire.read(); // Read the received comma
  }

  // Debugging: Print the received command to the Serial moni
  Serial.print("Received Command: ");
  Serial.println(receivedCommand);

  // Handle button press logic with debounce
  if (buttonState != lastButtonState) {
    if (buttonState == LOW) {  // Button was pressed
      Serial.print("READ COMMAND: ");
      Serial.println(readCommand);
```

```
      readCommand = false; // Temporarily disable command rea
      if (close) {
        digitalWrite(RELAY_PIN, HIGH); // Turn the relay on
        close = !close; // Toggle the close state
      } else {
        digitalWrite(RELAY_PIN, LOW); // Turn the relay off
        close = !close; // Toggle the close state
        readCommand = true; // Re-enable command reading
      }
    }
    lastButtonState = buttonState; // Update the last known b
  }

  // Handle commands from the master device if readCommand is
  if (readCommand) {
    if (receivedCommand == 'O') { // 'O' command: Open the va
      digitalWrite(RELAY_PIN, HIGH); // Turn the relay on
      Serial.println("OPEN VALVE"); // Debugging: Log action
    } else { // Any other command: Close the valve
      digitalWrite(RELAY_PIN, LOW); // Turn the relay off
      Serial.println("CLOSE VALVE"); // Debugging: Log actio
    }
  }
}
```