Smart Irrigation System using Microcontroller (SISM)

Presented by
Ramon Zintzun, Alexander Ov, Dylan St. Laurent, Steven Do
Group 8

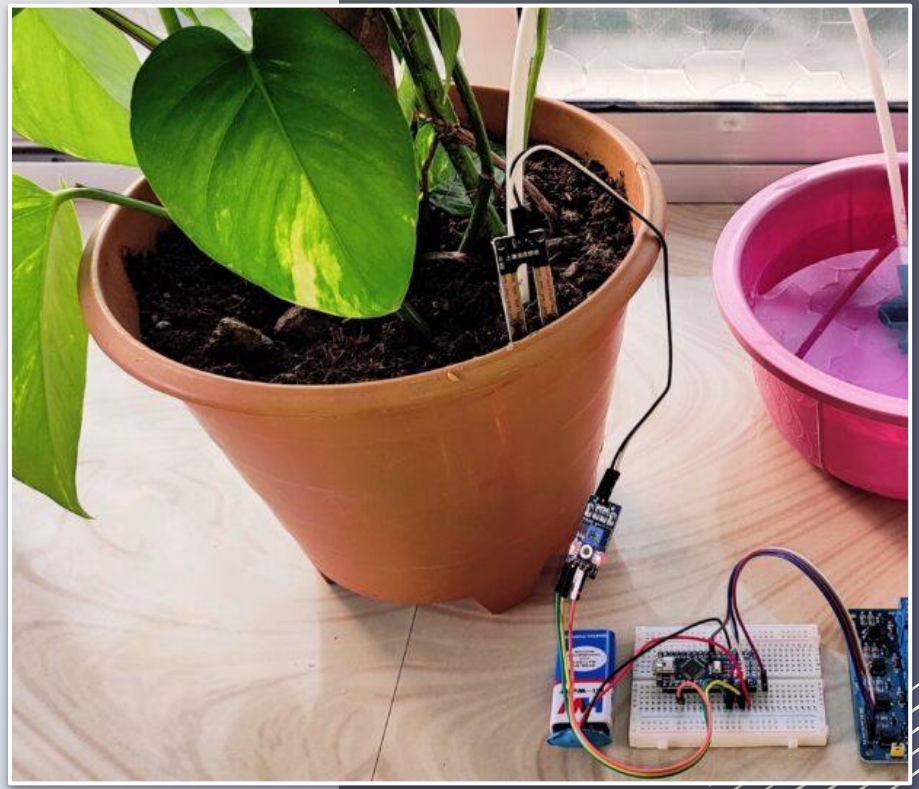
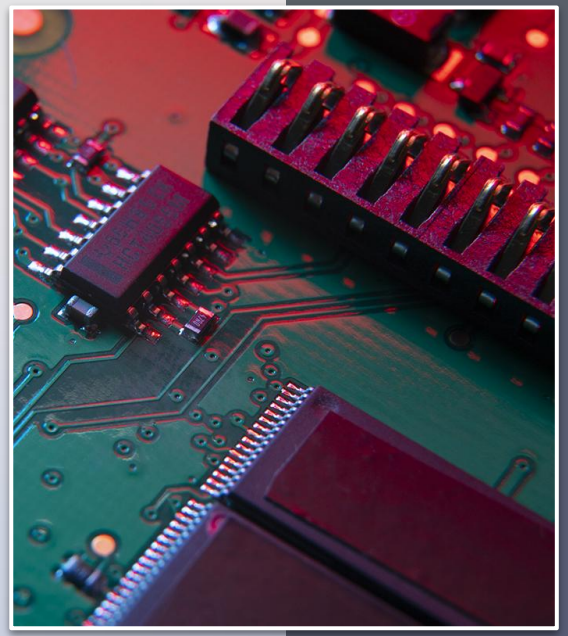
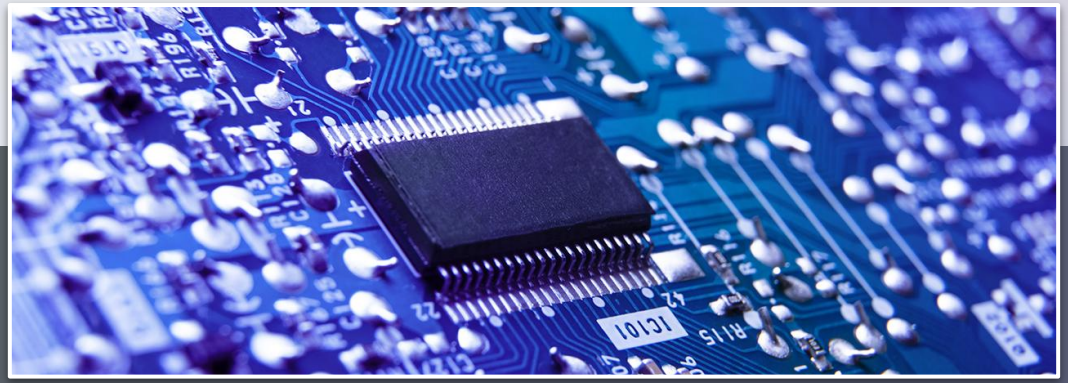


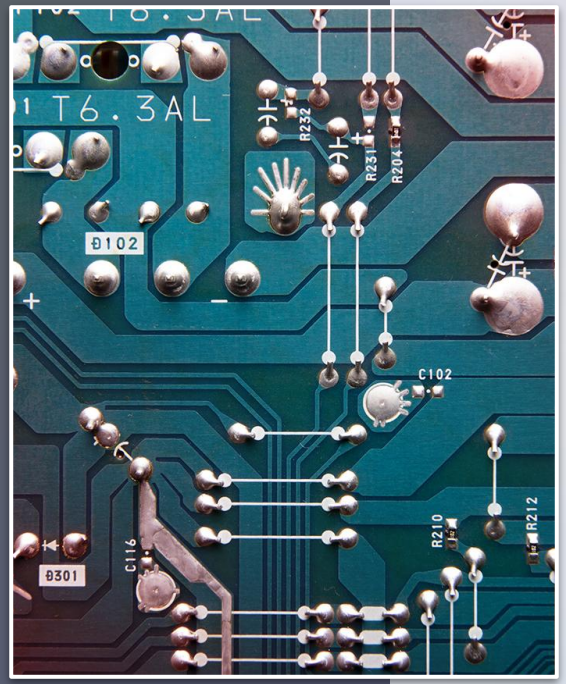
Table of contents



Whoa!

Microcontroller or World Controller?!
For now, Irrigation controller. :)





01

Introduction

The Purpose and objectives of this project is to implement the class lessons into real life applications.

What is Smart Irrigation System?

A smart irrigation system is a system that incorporates sensors that allow it to monitor real world conditions and react to them.

The system takes the data that it has collected from the sensors and then compares it to a set of parameters that are coded into the system. Based on the coded parameters the system will then respond with an appropriate reaction.

The smart irrigation system that we have implemented incorporates 4 analog sensors.

Analog Sensors Used:

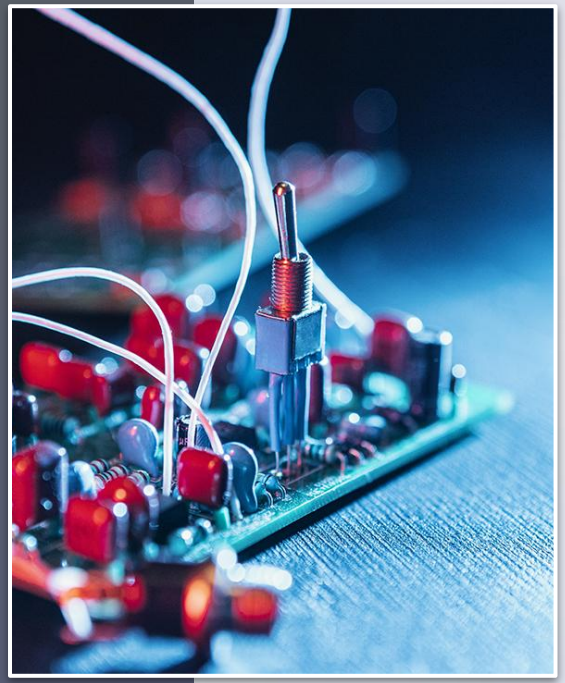
- Moisture Levels
- Temperature
- Light Levels
- Water Source Levels.

What is it practical use for?

Practical use of a smart irrigation system using a microcontroller is to automate the irrigation process. The smart irrigation system can be programmed to control the following components,

- Amount of water
- Frequency of watering
- Soil moisture level
- Temperature (Indoors or Outdoors)
- Humidity
- Weather Conditions

This ensures that the crops or plants in this situation receive the right amount of water at the right time, which can result in improved growth, reduced water consumption and lowering operation cost.



02

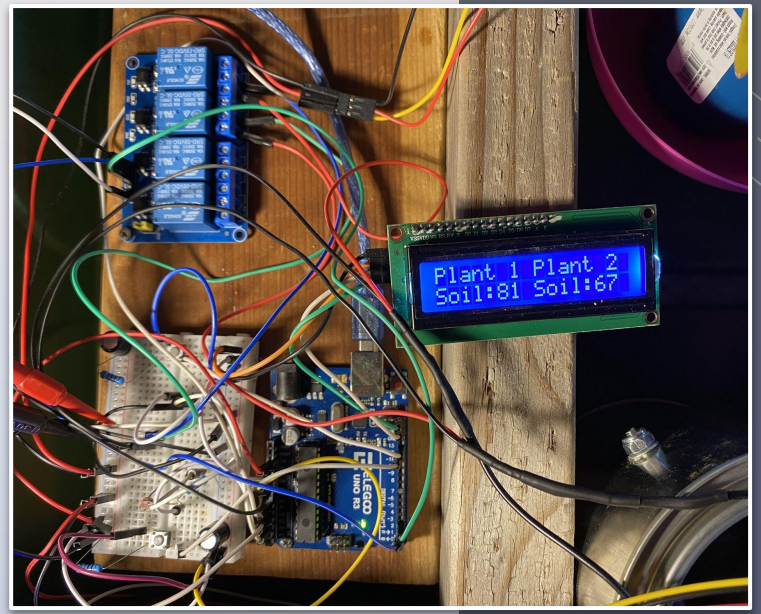
Getting Started

Research, materials/components, sensors and code that went into the creation of SISM

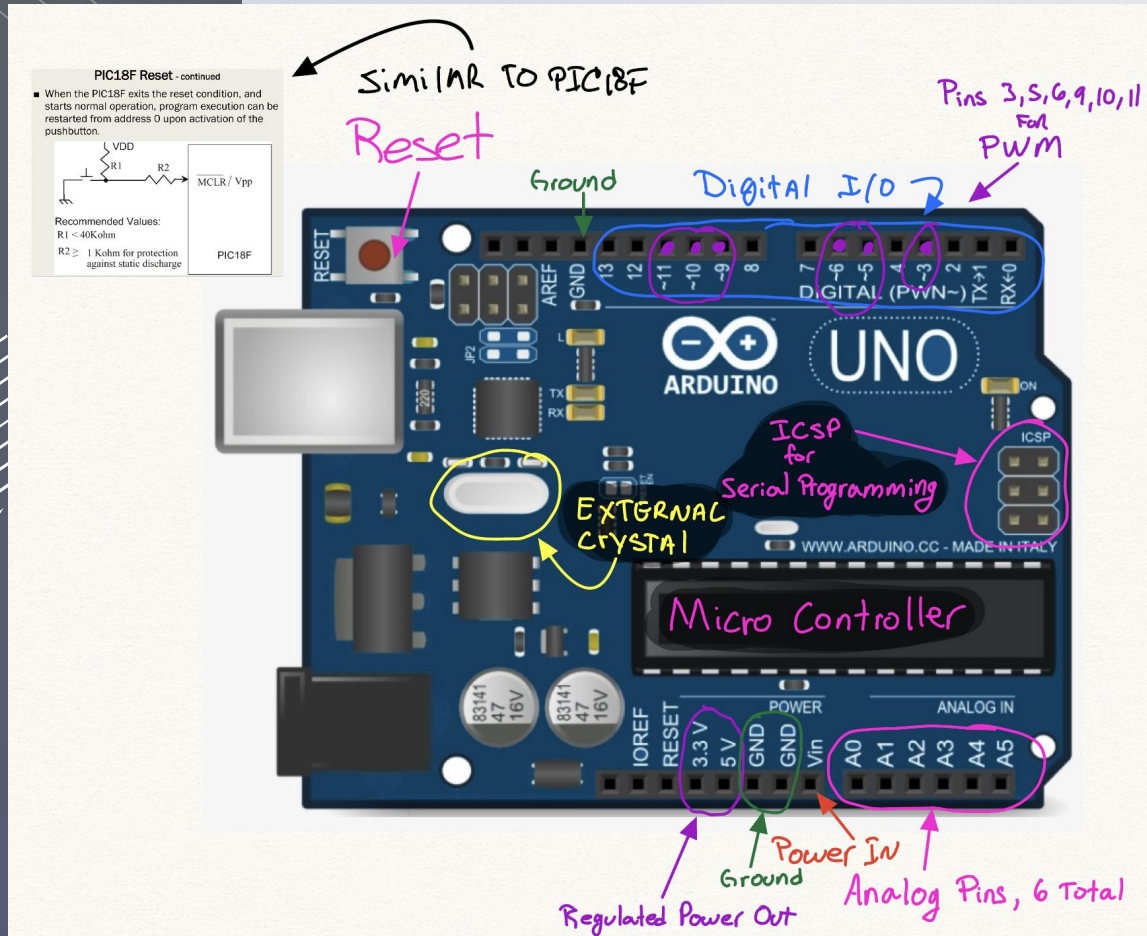
Building Blocks of a Smart Irrigation System

Components:

- Arduino Uno development Board
- 4-Channel 5V DC Relay
- LCD Display
- 3-4.5V DC Water Pump
- Tubing to transfer water
- Capacitive Moisture Sensor
- Light Sensor (photoresistor)
- Temperature Sensor
- Water Level Sensor
- Regulated Power Supply
- Water Container
- Plants
- Code



Arduino Uno Development Board

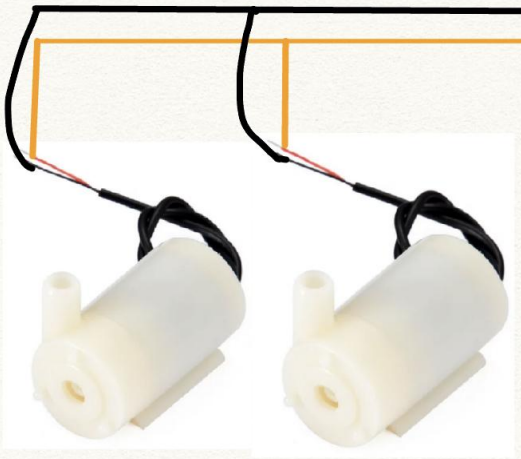


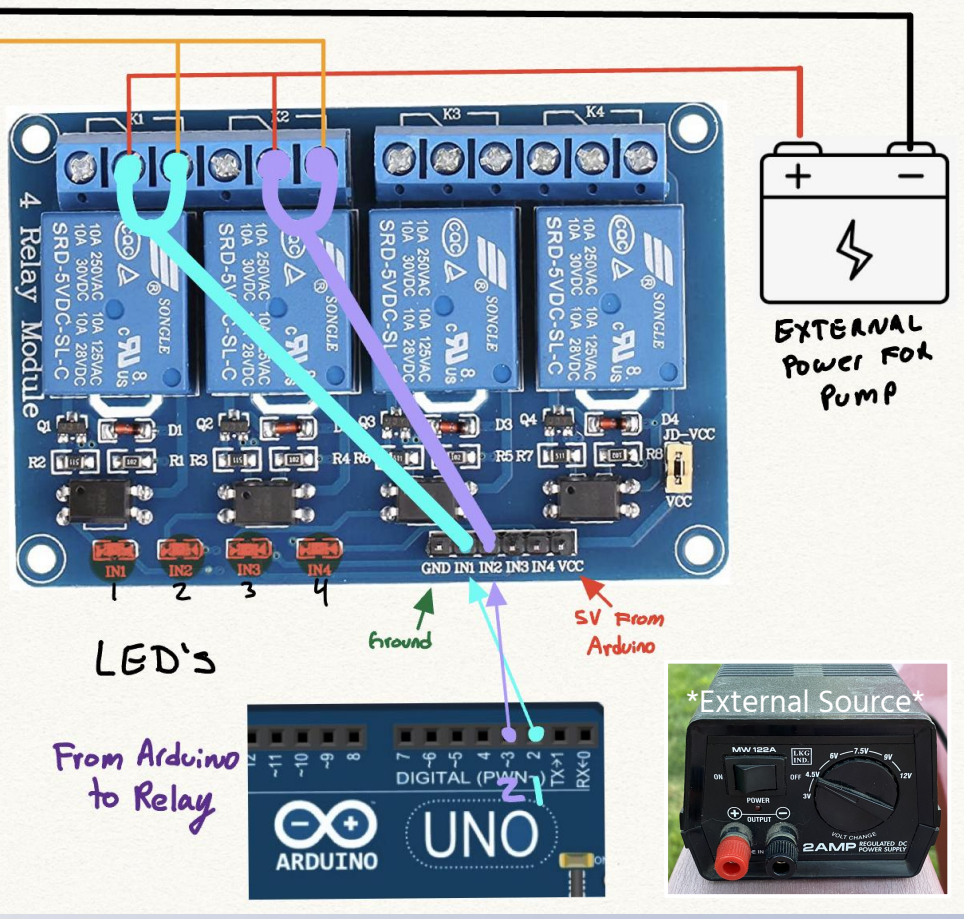
The Arduino Uno Development Board is very user friendly since it has all the basic needs of a microcontroller incorporated into it.

Things that we have studied can be seen incorporated into the development board.

- Reset Button
- Digital I/O
- Analog Inputs
- Serial Programming Pins
- PWM Pins
- Power pins
- Ground Pins
- External Crystal

4-Channel 5V DC Relay



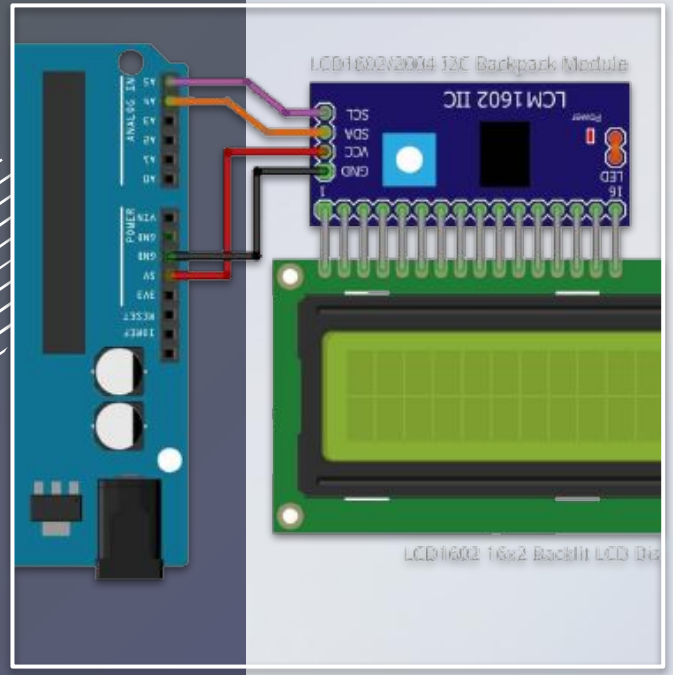


The Relay is used so that an outside power source can be used to power the water pumps since the Arduino would not provide enough power to run the pumps.

The relay acts like a switch connecting or disconnecting a wire.

Relay will wait for a signal from the controller to connect or disconnect the wire.

I2C LCD Display



I2C Protocol

The I2C or Inter-Integrated circuit is a chip to chip protocol that is used when communicating with low-speed peripherals using 2 lines: a serial clock (SCL) and a serial data (SDA). The SDA stores the data being transferred and the SCL clock signal synchronizes the data transfer between the devices on the I2C bus.

Project Application

Using the LiquidCrystal_I2C library we are able to use the SCL to synchronize the LCD display with the Arduino UNO based on the microcontroller clock speed. Then using the libraries commands such as print() we are able to send data from the Arduino UNO to the LCD display through the SDA.

Capacitive Moisture Soil Sensor

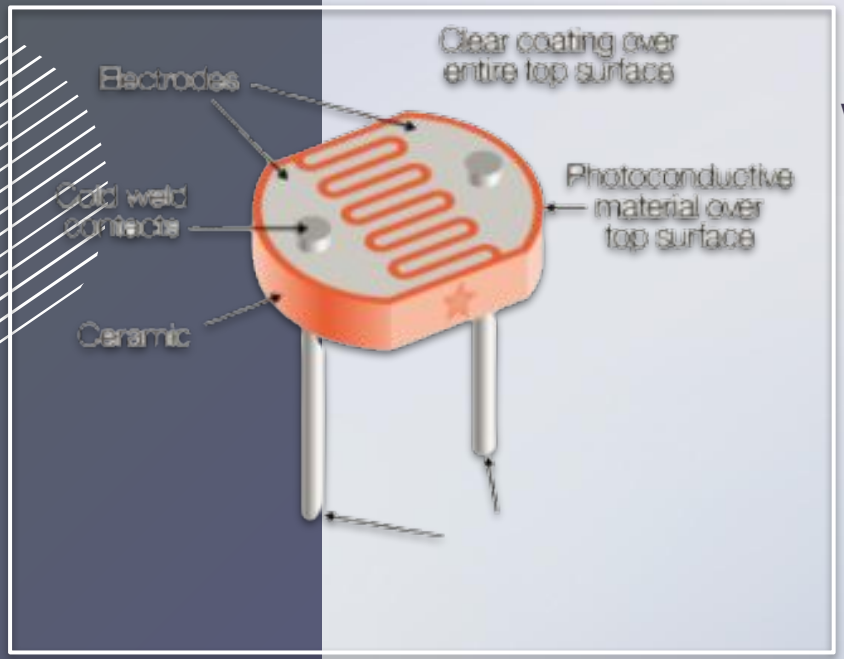
Background

A capacitive soil sensor is an electronic device that measures the capacitance of the soil, which varies with the water content of the soil.

Project Application

In the SISM, we used two sensor is used to measure the soil moisture level and send the data to the microcontroller. The microcontroller can then use the data to determine whether the soil needs watering and active the irrigation system accordingly.

Photoresistor



Photoresistor

We determined the light level of the environment using a photoresistor or a photocell which is a variable resistor where the resistance is changed through a change in light level.

Project Application

Photoresistor information was utilized to provide more awareness to the user about the environment. This was accomplished by broadcasting the analog readout to the LCD screen

Temperature Sensor (LMT84)

Background

The LMT84 is a precision analog temperature sensor that is used in electronic systems to measure temperature. It operates between the temps of .-50°C to 150°C with an accuracy of ±0.4°C at room temperature.

Project Application

In the SISM, the LMT84 temperature sensor can be used to measure the temperature of the soil or water to determine the optimal time for irrigation. The temperature of the soil or water can affect the growth of plants and crops, and by monitoring the temperature using the LMT84 sensor, the irrigation system can be programmed to only water when the temperature is within the desired range.

LMT84 Temperature Ranging

Supplied Data Sheet Equation

$$V - V_1 = \left(\frac{V_2 - V_1}{T_2 - T_1}\right)(T - T_1)$$

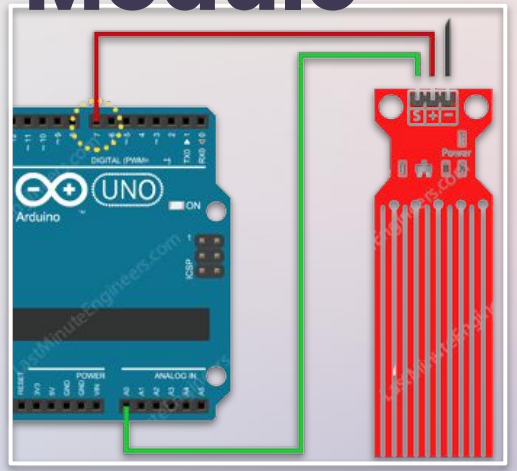
Solving for a range of 0℃ - 37℃ yields:

$$\frac{1012 - V_{mV}}{5.46} = Temp$$

TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)
–50	1299	-10	1088	30	871	70	647	110	419
–49	1294	-9	1082	31	865	71	642	111	413
–48	1289	-8	1077	32	860	72	636	112	407
–47	1284	-7	1072	33	854	73	630	113	401
–46	1278	-6	1066	34	849	74	625	114	396
–45	1273	-5	1061	35	843	75	619	115	390
–44	1268	-4	1055	36	838	76	613	116	384
–43	1263	-3	1050	37	832	77	608	117	378
–42	1257	-2	1044	38	827	78	602	118	372
–41	1252	-1	1039	39	821	79	596	119	367
–40	1247	0	1034	40	816	80	591	120	361
–39	1242	1	1028	41	810	81	585	121	355
–38	1236	2	1023	42	804	82	579	122	349

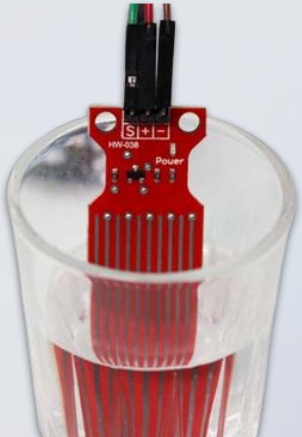
TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)
–37	1231	3	1017	43	799	83	574	123	343
–36	1226	4	1012	44	793	84	568	124	337
–35	1221	5	1007	45	788	85	562	125	332
–34	1215	6	1001	46	782	86	557	126	326
–33	1210	7	996	47	777	87	551	127	320
–32	1205	8	990	48	771	88	545	128	314
–31	1200	9	985	49	766	89	539	129	308
–30	1194	10	980	50	760	90	534	130	302
–29	1189	11	974	51	754	91	528	131	296
–28	1184	12	969	52	749	92	522	132	291
–27	1178	13	963	53	743	93	517	133	285
–26	1173	14	958	54	738	94	511	134	279
–25	1168	15	952	55	732	95	505	135	273
–24	1162	16	947	56	726	96	499	136	267
–23	1157	17	941	57	721	97	494	137	261
–22	1152	18	936	58	715	98	488	138	255
–21	1146	19	931	59	710	99	482	139	249
–20	1141	20	925	60	704	100	476	140	243
–19	1136	21	920	61	698	101	471	141	237
–18	1130	22	914	62	693	102	465	142	231
–17	1125	23	909	63	687	103	459	143	225
–16	1120	24	903	64	681	104	453	144	219
–15	1114	25	898	65	676	105	448	145	213
–14	1109	26	892	66	670	106	442	146	207
–13	1104	27	887	67	664	107	436	147	201
–12	1098	28	882	68	659	108	430	148	195
–11	1093	29	876	69	653	109	425	149	189
								150	183

Water Level Sensor Module



Background

A water level sensor module is an electronic device used to measure the level of water in a container or a tank. The sensor module typically consist a series of probes that are placed inside the container or tank, which detects the level of water by measuring the electrical conductivity of water.



Project Application

In SISM, the water level sensor module is used to help prevent over-or under-watering of plants, which can lead to either waste of water or damage to the plants. By monitoring the water level in real-time, the system can adjust the watering schedule to ensure the plants receive the optimal amount of water.

Fabrication of Environment

Water Level Sensor Mount

To store the water used to upkeep the plant life in our system, a small 1 quart paint can was utilized to show functionality of water pumps. The water level sensor was fashioned to a thin slice of wood that was mounted on the paint can to provide a solid mount surface for our sensor.

Possible Larger Supply

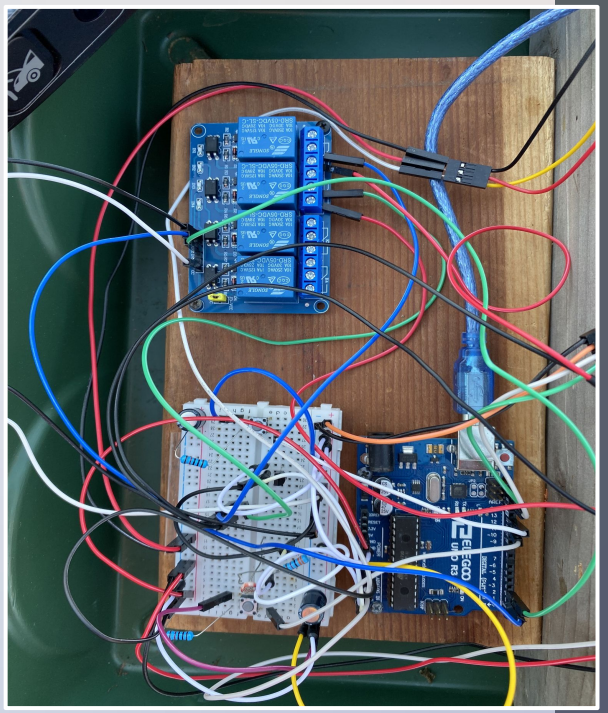
Due to portability being prioritized in the project design, a larger source container was not desired. In a more realistic application a larger 5 gallon container would be utilized to provide for longer operation time without human intervention.

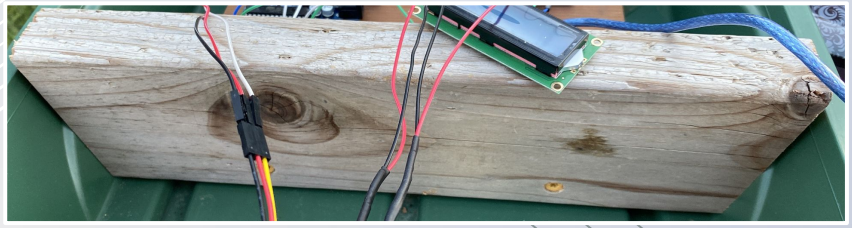
Fabrication of Environment

Main Mounting Platform

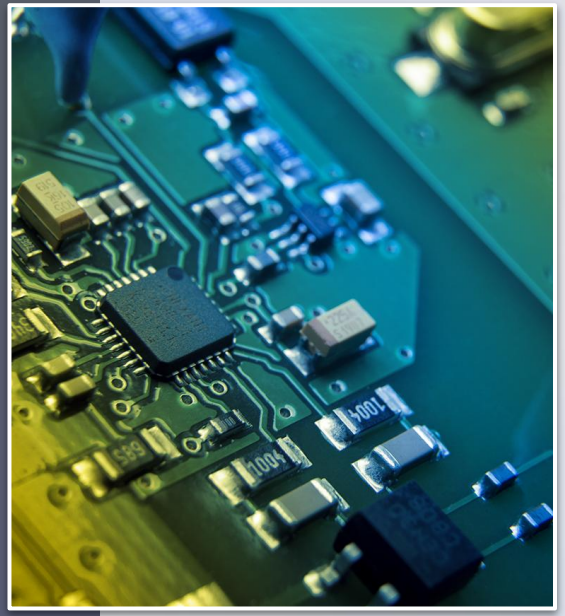
To protect the fragile components of the system such as the relay and arduino, a master mounting surface was required. A 2 inch thick board was used as a platform to mount to with screws ensuring stability amongst parts as well as protection from water.

A splitting board to generate a compartment like section was obtained by measuring the 13 degree angle in the tub and mounted to the main platform.





Closed Loop Control System & Code



Microcontroller

Here the controller reacts to input from the sensors and responds with new instructions.

Plant/Process

The water pump and power source act as the power plant for the system.

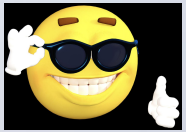
Sensors

The System uses 4 analog sensors to analyze outside conditions so that data can be fed to the controller.

Input

The data from the sensors is fed to the controller so that it can give compensated instructions to the plant (water pumps).

A nod to ECE 3709



Code Implementation for Smart Irrigation System using Microcontroller

Coding Language

For this project, we wrote our code for SISM in C along with including comments.

Integrated development environment (IDE)

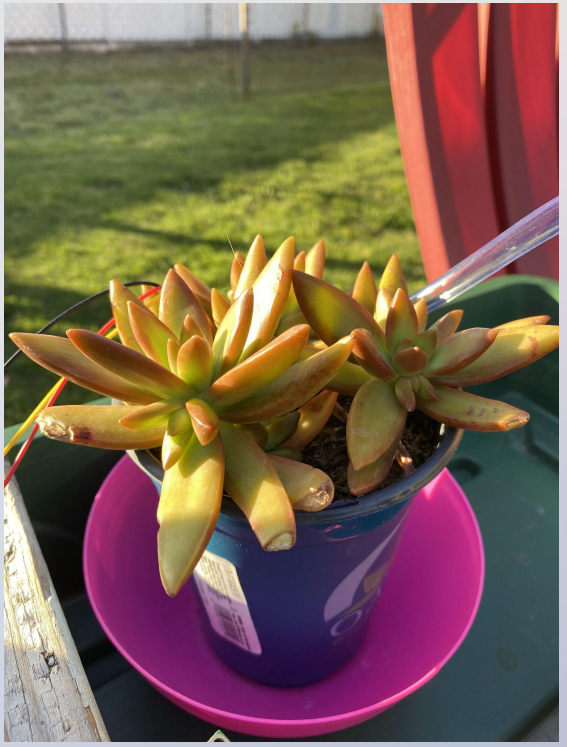
To program the Arduino Uno Development Board, we choose to write it in the Arduino IDE. The Arduino IDE allows for its variety of built-in-library, like the LiquidCrystal_I2C library, to interface with the common sensors, led, water pumps, and temperature sensor. In addition for the compatibility of it being cross-platform.

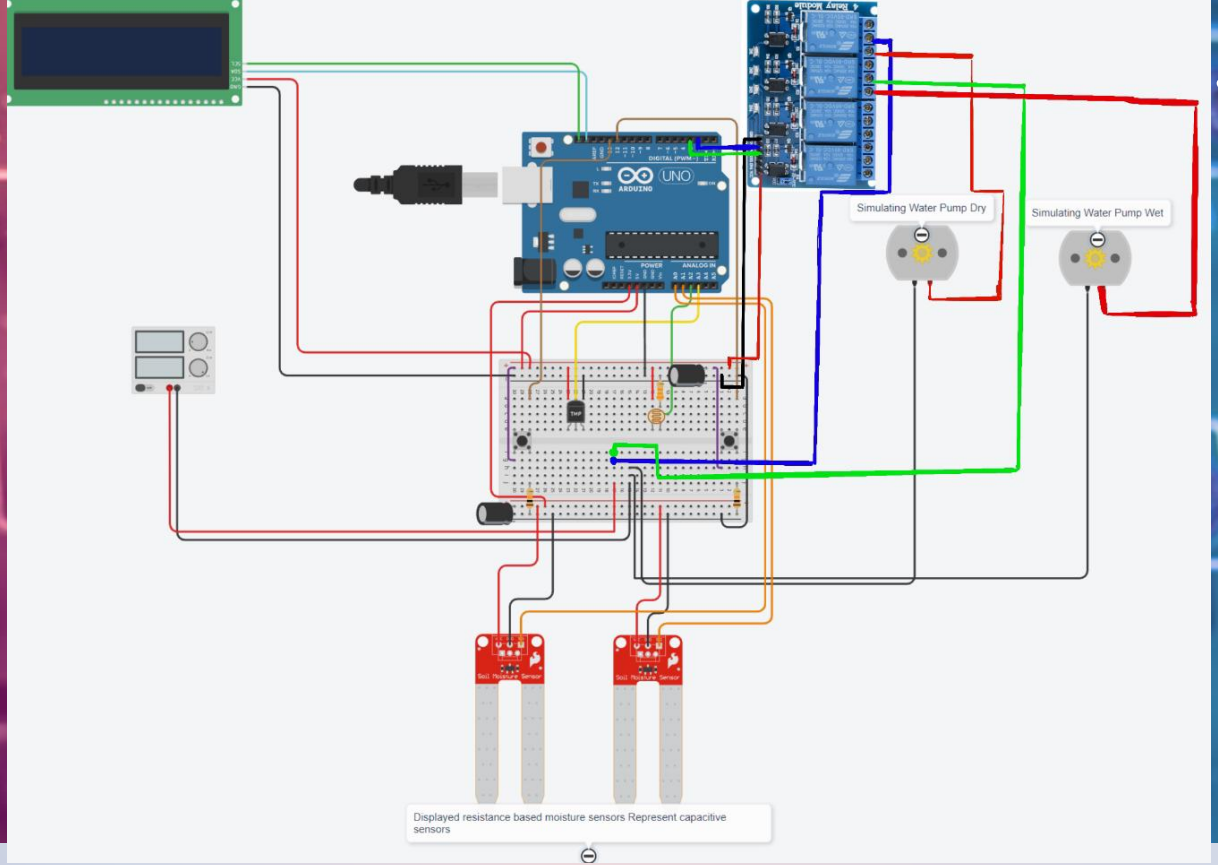
03

Results

As you can see the plants are still ALIVE!!

These are actual Pictures of our Live Plants! :)





A picture is worth a thousand words

THE CODE:

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

LiquidCrystal_I2C lcd(0x27,16,2); // initialize the lcd display with 16 columns and 2 rows

int lightsensorValue = 0; // initialize the variable for the light sensor

const int tempPin = A3; // define the analog input pin for the temperature sensor (LMT84)
float tempC = 0; // initialize the temperature value in Celsius
float tempF = 0; // initialize the temperature value in Fahrenheit

const int btnPin = 12; // define the digital input pin for button 1
const int btn2Pin = 13; // define the digital input pin for button 2
int btnState = 0; // initialize the state of button 1
int btn2State = 0; // initialize the state of button 2
int lcdState = 0; // initialize the state of the lcd display

int OutToRelay1DRY = 2; // define the digital output pin for the dry pump relay
int OutToRelay2Wet = 3; // define the digital output pin for the wet pump relay
int InMoistureSensor1 = A0; // define the analog input pin for the first moisture sensor
int InMoistureSensor2 = A1; // define the analog input pin for the second moisture sensor
float MoistureSensorValue1 = 0; // initialize the value of the first moisture sensor
float MoistureSensorValue2 = 0; // initialize the value of the second moisture sensor
int Moist1Percentage = 0; // initialize the percentage of moisture for the first sensor
int Moist2Percentage = 0; // initialize the percentage of moisture for the second sensor
```

THE CODE:

```
void setup()
{
  Serial.begin(9600); // initialize serial communication

  lcd.init(); // initialize the lcd display
  lcd.clear(); // clear the lcd display
  lcd.backlight(); // turn on the backlight for the lcd display
  lcd.setCursor(0,0); // set the cursor to the top left corner of the lcd display

  pinMode(A1, INPUT); // set pin A1 as an input
  pinMode(A2, INPUT); // set pin A2 as an input
  pinMode(9, OUTPUT); // set pin 9 as an output
  pinMode(btnPin, INPUT); // set pin btnPin as an input
  pinMode(btn2Pin, INPUT); // set pin btn2Pin as an input
  pinMode(OutToRelay1DRY, OUTPUT); // set pin OutToRelay1DRY as an output
  pinMode(OutToRelay2Wet, OUTPUT); // set pin OutToRelay2Wet as an output
  pinMode(InMoistureSensor1, INPUT); // set pin InMoistureSensor1 as an input
  pinMode(InMoistureSensor2, INPUT); // set pin InMoistureSensor2 as an input

  digitalWrite(OutToRelay1DRY, HIGH); // set the initial state of the dry pump relay to off
  digitalWrite(OutToRelay2Wet, HIGH); // set the initial state of the wet pump relay to off
}
```

THE CODE:

```
void loop()
{
  // Read temperature from analog pin and convert to Fahrenheit
  int tempRead = analogRead(tempPin);
  float tempmV = ((float) tempRead)*4.9;
  float volt_v = tempmV/1000.0;
  float temp_C = ((1012.0 - tempmV)/5.45);
  float temp_F = 1.80 * temp_C + 32.0;

  // Read state of two buttons
  btnState = digitalRead(btnPin);
  btn2State = digitalRead(btn2Pin);

  // Print temperature to serial monitor
  Serial.println("LMT84 Temp Sensor: ");
  Serial.print("Analog: "); Serial.println(tempRead);
  Serial.print("mVolts: "); Serial.println(tempmV,4);
  Serial.print("Temp: "); Serial.println(temp_F, 4);

  // Read sensor data and update moisture percentage values
  sensorRead(lightsensorValue, InMoistureSensor1, InMoistureSensor2);

  // Determine LCD state based on button press
  if (btnState == HIGH) {
    if(lcdState<1) lcdState++;
    else if(lcdState>0) lcdState--;
  }
  if (btn2State == HIGH) {
    if(lcdState>0) lcdState--;
    else if(lcdState<1) lcdState++;
  }
```

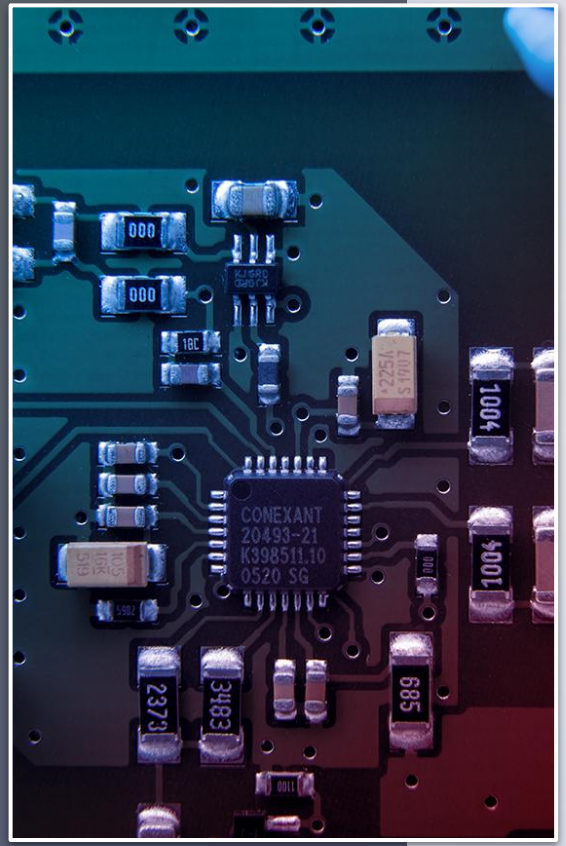
```
  // Control water pump based on moisture sensor readings
  if(MoistureSensorValue1 > 400) {
    digitalWrite(OutToRelay1DRY,LOW);
  }
  else {
    digitalWrite(OutToRelay1DRY,HIGH);
  }

  if(MoistureSensorValue2 > 300) {
    digitalWrite(OutToRelay2Wet,LOW);
  }
  else {
    digitalWrite(OutToRelay2Wet,HIGH);
  }

  // Update LCD display with sensor data
  printPage(lcdState, temp_F, lightsensorValue, MoistureSensorValue1/10, MoistureSensorValue2/10);

  // Delay for 500 milliseconds before looping again
  delay(500);
}
```

THE CODE:

```
// Read moisture sensor data
void sensorRead( int lighsensorValue, int InMoistureSensor1, int InMoistureSensor2){
  // Read light sensor data
  lightsensorValue = analogRead(A2);
  Serial.println(lightsensorValue);

  // Read moisture sensor 1 and print value
  MoistureSensorValue1= analogRead(InMoistureSensor1);
  Moist1Percentage = map(MoistureSensorValue1,199,514,100,0);
  Serial.print("Moisture Sensor 1 Value: ");
  Serial.print(MoistureSensorValue1);

  // Read moisture sensor 2 and print value
  MoistureSensorValue2= analogRead(InMoistureSensor2);
  Moist2Percentage = map(MoistureSensorValue2,186,512,100,0);
  Serial.print("   Moisture Sensor 2 Value: ");
  Serial.println(MoistureSensorValue2);

  // Delay for 1 second before continuing
  delay(1000);
}
```

THE CODE:

```
void printPage(int lcdState, int temp_F, int lightsensorValue, int MoistureSensorValue1, int MoistureSensorValue2){

  // Check if the lcdState is 0
  if(lcdState == 0){
    // Clear the LCD screen
    lcd.clear();

    // Set the cursor position to display "Plant 1" on the first row
    lcd.setCursor(0, 0);
    lcd.print("Plant 1");

    // Set the cursor position to display "Plant 2" on the first row, 8 characters from the left
    lcd.setCursor(8, 0);
    lcd.print("Plant 2");

    // Set the cursor position to display "Soil:" on the second row, first column
    lcd.setCursor(0, 1);
    lcd.print("Soil:");

    // Display the moisture level of the first plant on the second row, after "Soil:"
    lcd.print(Moist1Percentage);

    // Set the cursor position to display "Soil:" on the second row, eighth column
    lcd.setCursor(8, 1);
    lcd.print("Soil:");
```



Thanks!

Does anyone have any questions?
If you do, please address them to the professor!
JUST KIDDING

NOW CLAP :)

California State Polytechnic University Pomona
Department of Electrical and Computer Engineering
ECE 3301-01

Smart Irrigation System using Microcontroller (SISM)

Presented By
Ramon Zintzun
Alexander Ov
Dylan St Laurent
Steven Do
May 7, 2023

Table of Contents:

Abstract:

The Smart Irrigation System using Microcontroller (SISM) is a cutting-edge technology that holds immense potential in revolutionizing plant irrigation practices. In today's society, where water scarcity and environmental sustainability are pressing concerns, SISM offers a practical and efficient solution for optimizing water usage in plant irrigation. SISM leverages the power of microcontrollers to gather real-time data from various sources such as weather conditions, soil moisture levels, and plant-specific requirements. By analyzing this data, the system intelligently determines the precise amount of water needed by each plant, ensuring optimal hydration without any water wastage. This targeted approach not only conserves water but also minimizes the risk of overwatering, which can be detrimental to plant health. In today's society, where efficient resource management is crucial, SISM plays a pivotal role in conserving water resources. By significantly reducing water consumption in plant irrigation, the system helps address water scarcity issues and promotes sustainable water usage practices. Additionally, SISM enables farmers and gardeners to optimize their irrigation processes, resulting in healthier plants, improved crop yields, and reduced maintenance costs.

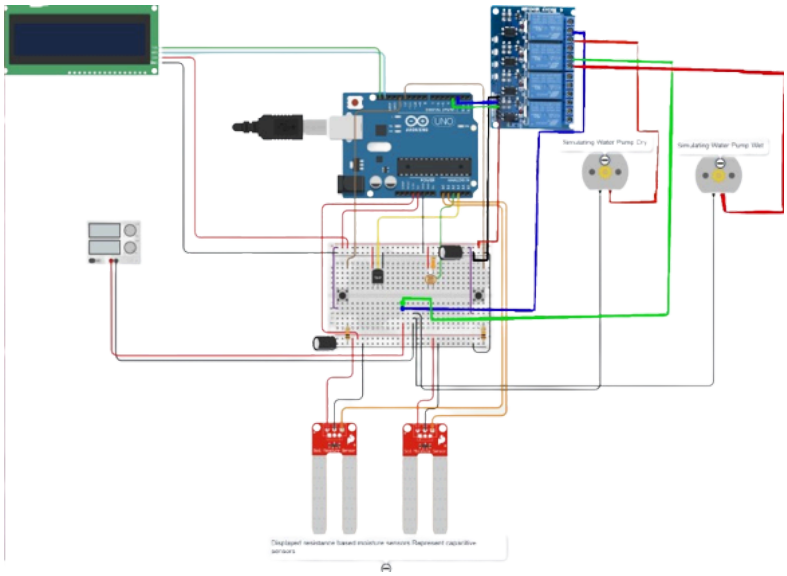


Figure 1: Concept of Smart Irrigation System using an Microcontroller

Introduction:

Smart irrigation systems have gained significant attention in recent years due to their potential to optimize water usage and improve crop yields. These systems leverage advanced technologies, such as microcontrollers, to monitor and control irrigation processes automatically. In this research paper, we focus on the implementation of a Smart Irrigation System using a Microcontroller (SISM) for plants.

The use of microcontrollers in smart irrigation systems offers numerous advantages, including precise control over watering schedules, real-time monitoring of environmental parameters, and efficient water resource management. By integrating sensors and actuators with a microcontroller-based control system, the SISM can collect data from various sources, such as soil moisture sensors, temperature sensors, and light sensors, to make informed decisions about irrigation requirements.

The context of this project revolves around the increasing need for sustainable agricultural practices in the face of growing water scarcity and climate change. Traditional irrigation methods often result in excessive water consumption and suboptimal plant growth, leading to water waste and reduced crop productivity. By developing and implementing smart irrigation systems, we can address these challenges by optimizing water usage, minimizing resource wastage, and enhancing the overall efficiency of agricultural practices.

The importance of this project lies in its potential to revolutionize irrigation practices in the agriculture industry. By combining the capabilities of microcontrollers and sensor technologies, the SISM can provide precise and timely irrigation, tailored to the specific needs of plants. This approach ensures that plants receive adequate water while avoiding over-or under-watering, resulting in improved crop quality and yield. Moreover, the efficient utilization of water resources contributes to environmental sustainability and the conservation of this vital resource.

In this paper, we present a detailed analysis of the Smart Irrigation System using a Microcontroller (SISM), focusing on its hardware and software components. We will discuss the design and implementation of the system, including the integration of various sensors and actuators.

Through this paper, we aim to contribute to the ongoing efforts in developing sustainable agricultural practices and promoting water conservation through the adoption of smart irrigation systems. By highlighting the capabilities and benefits of the Smart Irrigation System using a Microcontroller (SISM), we envision a future where precision agriculture becomes the norm, ensuring optimal plant growth while preserving our precious water resources.

Experimental Methodology:

This code is written in the Arduino programming language and includes the use of various sensors, an LCD display, and relay modules to control outputs based on input readings. It can be used in several real-life applications, such as plant watering systems, home automation systems, and temperature monitoring systems.

The code starts with including the LiquidCrystal_I2C library, which enables the code to communicate with an LCD display module. It initializes the LCD display module and sets the backlight on. Then it defines various pins for sensors, buttons, and relay modules.

The first sensor used in this code is the LMT84 temperature sensor, which is connected to pin A3. The code reads the analog value from the sensor, converts it to temperature in Fahrenheit and prints the values to the Serial Monitor. The temperature sensor can be used in applications such as a temperature monitoring system for a greenhouse or a room.

Next, the code reads the state of two buttons connected to pins 12 and 13. The button states are used to control the LCD display to show different readings from sensors. This feature can be used in home automation systems, where users can control the displays based on their needs.

Third, in the code, the photoresistor is connected to analog pin A2 (lightsensorValue). The sensorRead function reads the analog value from the photoresistor using analogRead(A2), and the obtained value is stored in the lightsensorValue variable. The lightsensorValue is then used in the printPage function, where it can be displayed on the LCD screen along with other sensor data. The value can also be printed to the serial monitor for debugging purposes using Serial.println(lightsensorValue). Based on the light intensity measured by the photoresistor, the Smart Irrigation System can make decisions about when to water the plants. For example, if the light intensity is low, indicating that it is dark or there is little sunlight, the system may delay or reduce the watering to avoid over-watering the plants. Conversely, if the light intensity is high, indicating that it is bright and sunny, the system may increase the watering frequency to compensate for increased evaporation and plant water requirements. While such a system is not present within the operation of our SISM, it could be easily added as an operational feature in an updated model.

The code then reads the analog value from two moisture sensors connected to pins A0 and A1, respectively. The MoistureSensorValue1 and MoistureSensorValue2 are threshold values set to decide if the relay modules should be turned on or off. If the moisture value is above the threshold, the relay module connected to the corresponding moisture sensor will be turned on, which is used in the plant watering system.

Finally, the code sets the pins for relay modules as outputs and sends a high signal to them to turn them off initially. Then the code sets the pins for moisture sensors as inputs. The relay modules' pins are set as outputs to control the water pump's power supply. This allows for users to turn on or off the water pump based on moisture levels in the soil.

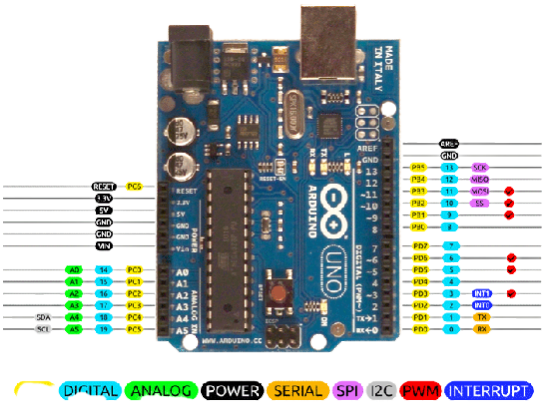


Figure 2: Diagram of Arduino Development Board

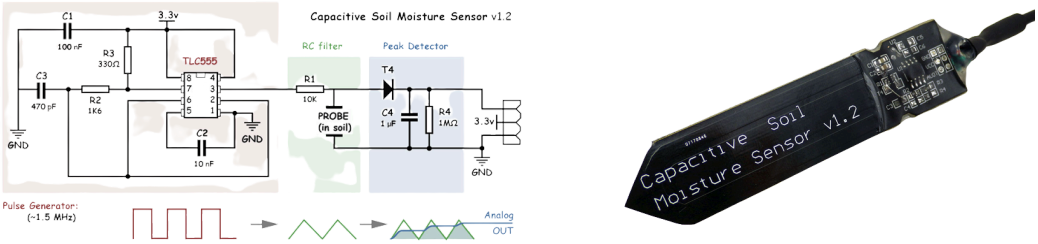


Figure 3: Internal schematic of an Capacitive Soil Moisture Sensor v1.2

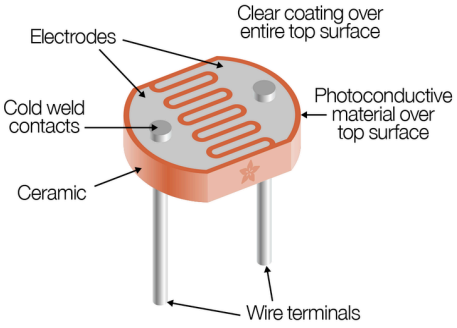


Figure 4: Diagram of an Photoresistor

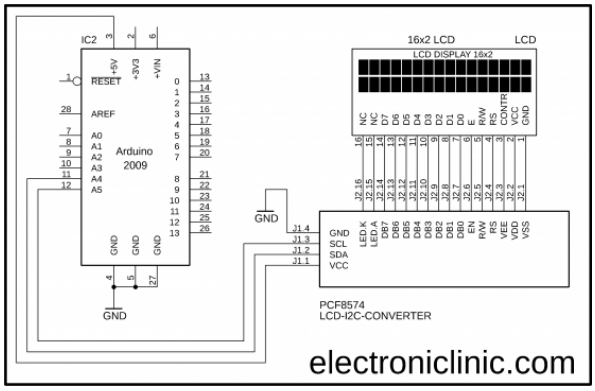


Figure 5: Diagram of an I2C LCD Display

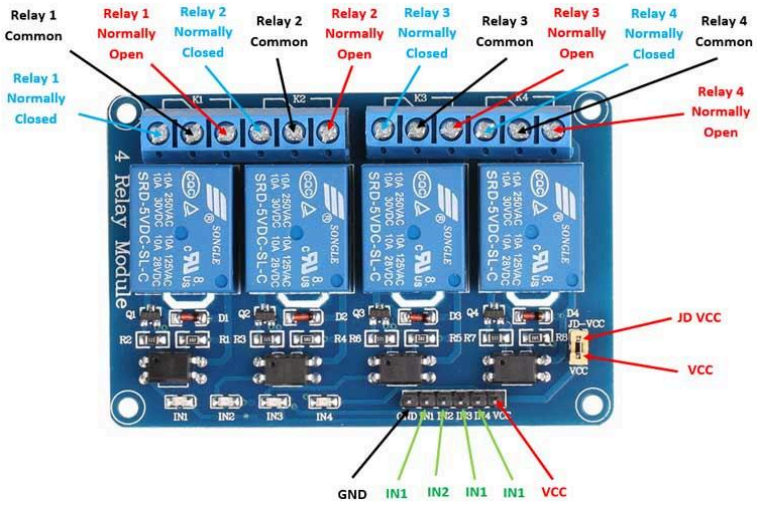


Figure 6: Diagram of an 4-Channel 5V

Experimental Results:

The Smart Irrigation System using Microcontroller (SISM) was implemented and tested to evaluate its performance in monitoring and controlling soil moisture levels for two plants. The following results were obtained:

1. Temperature Monitoring:
    a. The LMT84 temperature sensor was used to measure the temperature in Fahrenheit. The analog readings from the sensor were converted to temperature values. The temperature values were printed to theI2C LCD Display. No significant issues or deviations were observed during temperature monitoring.
2. Moisture Sensor Readings:
    a. Two moisture sensors (Moisture Sensor 1 and Moisture Sensor 2) were utilized to measure the moisture levels in the soil for Plant 1 and Plant 2, respectively. The moisture sensor values were obtained through analog readings and converted to percentage values. The moisture sensor readings were printed to the I2C LCD Display.
        i. Moisture Sensor 1: The readings from Moisture Sensor 1 were in the range of 199 to 514. These values were mapped to a moisture percentage range of 0 to 100. The moisture level of Plant 1 was displayed on the LCD screen.
        ii. Moisture Sensor 2: The readings from Moisture Sensor 2 ranged from 186 to 512. These values were mapped to a moisture percentage range of 0 to 100. The moisture level of Plant 2 was displayed on the LCD screen.
3. Photoresistor:
    a. A light sensor was used to measure the light intensity. The analog reading from the light sensor was obtained and printed to the I2C LCD Display. The light intensity monitoring was performed successfully.
4. LCD Display:
    a. The LCD display was utilized to present the sensor data and system status. The LCD screen showed two rows of information. The first row displayed the labels "Plant 1" and "Plant 2" for the respective plants. The second row showed the moisture levels of the plants labeled as "Soil:" followed by the percentage values. The LCD display was updated based on the state of two buttons (btnPin and btn2Pin). Pressing btnPin cycled through different display states, while btn2Pin reversed the cycling direction. The LCD state change was synchronized with the button presses.
5. Water Pump Control:
    a. The SISM system controlled two water pumps (dry pump and wet pump) based on the moisture sensor readings. If the moisture sensor reading for a

plant exceeded a specific threshold, the respective water pump was activated. Otherwise, the water pump remained off. The system successfully controlled the water pumps based on the moisture sensor data.

Overall, the SISM system demonstrated effective monitoring of temperature, moisture levels, and light intensity. The LCD display provided clear and concise information about the plants' moisture levels. The water pump control mechanism operated as intended, ensuring efficient irrigation based on the plants' needs. These results indicate that the SISM system can effectively regulate irrigation for plants in an automated and intelligent manner.

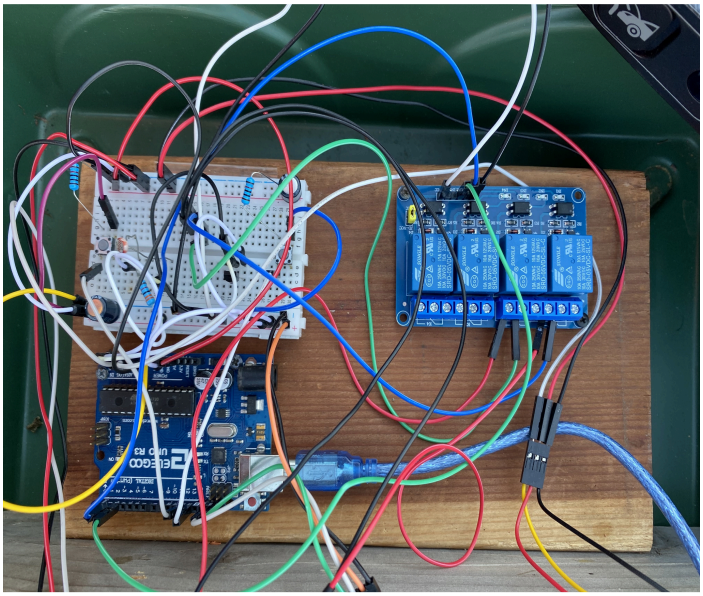


Figure 7: Visual Assembly of the SISM

Figure 8: Outputs of the Plant 1's & Plant 2's Moisture Sensor Readings on LCD Display

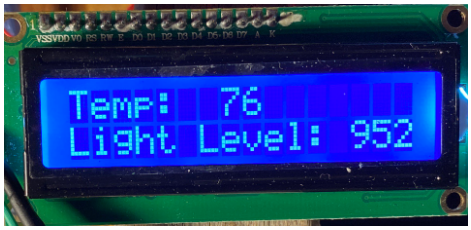


Figure 9: Outputs of the Plant 1 & Plant 2's Temperature Sensor Readings on LCD Display

Figure 10: Visual Photos of the Live Plants getting water from the Water Pump

Summary of Challenges:

Development of the Smart Irrigation System did not come without its various trials. One of the first issues was a display error in the LCD display. More specifically, when outputting to the LCD an error was observed that only the first character of the transmitted data would display on the screen. In an attempt to correct this error, we first scoured through the code ensuring that we didn't make a mistake with placements of other text and title locations. After it was clear that there were no errors in printing to the screen in code syntax, we attempted to see if the display itself was damaged by printing a test code to the screen from a separate project file. The issue persisted through to the new file which indicated that the error was outside of the syntax of the code itself.

The next easiest solution to attempt was to transfer the code to a separate computer. Upon changing the computer that was connecting to the display, the expected output was observed on the Liquid Crystal Display. This isolated the error to the original machine that the code was being developed from. Originally, when installing the library for connecting to the LCD multiple libraries were offered. An incorrect liquid crystal library was first downloaded to the machine and promptly changed to the correct version. However, because both libraries existed on the machine, the two libraries conflicted with each other when being called. To rectify the mistake, both libraries were uninstalled and only the correct library was installed and included.

The next issue encountered in the design process pertained to the water level sensor. When we first assembled all of the sensors with the arduino, we had excluded the water level sensor as we did not plan on utilizing it in the project. Multiple, separate instances of the code were run with the light, temperature, and moisture sensors ensuring that the SISM was functional in this test stage before the introduction of the water level sensor. Upon connecting the peripheral to port A4 and launching the program with an updated version of the source code, an error would occur resulting in the program hanging. No sensor readings were observed in the arduino output window and the LCD did not make it through its initialization stage, resulting in the 2x16 grid display of the LCD being filled.

It was apparent that there was an issue with the addition of the sensor here so we progressed to searching through the code implementation for mistakes. After we verified that there was no issue in the syntax of the code, we delved through the implementation and also found it to be free of errors. This moved the search to the hardware realm and resulted in us first testing the functionality of the wires between the devices. The errors persisted through with the replacement wires and so we suspected a problem with the sensor. After running multiple tests with the level sensor utilizing a separate source code we determined that there was an issue with the sensor itself. To fix this problem we have obtained a new water level sensor and extras to ensure at least one functional sensor is acquired. Due to issues with the shipping of the water sensors, their arrival was pushed back past the first finalized design of the project, resulting in their removal from the current design phase.

Finally, one of our most key issues was the quality of the moisture sensors. As it turns out, there are many capacitive moisture sensors that are shipped out that dont work. Considering that the moisture sensor was a key element in our project this was very problematic. As it turned

out, the first set of 4 moisture sensors were bad and had to be replaced with another 4 that were ordered later which caused a serious set back.

Starting off with bad moisture sensors was a very big headache especially since the first set of 4 were bad. This was bad because at first we did not know why are project was not working and did not know if it was our code, we did not know if it was software or hardware related, and since we expected the equipment to work we thought it was in our software at first, and then perhaps the way we had it connected. After painstakingly going through the logic and code, we figured we should order a second set of equipment including sensors and microcontroller. We then started with the microcontroller by verifying power and logic with a multimeter. The code and microcontroller (Arduino-Uno) seemed to be working fine. A high gave 5 volts, and low gave 0 volts. So the next logical step was to check the sensor.

After going through the microcontroller and code with a multimeter and verifying that everything seemed to be ok, it was now time to test and troubleshoot the sensors. So a new batch of 4 sensors were used and as it turned out 2 of the four sensors were working. We then decided to see what was wrong with the other 6 sensors because originally we were going to use 4 of them. So after some research and going through a schematic we found online for the capacitive moisture sensors, it turns out that there were several issues. To break it down there were 3 main issues. One was that some were missing a voltage regulator, the second was that some used the wrong timer chip, and the third and biggest issue was that there was a 1Mega ohm resistor that was not grounded properly and was left open. It was the ungrounded resistor that was causing the biggest issue with the sensor.

Problem 1 with the capacitive moisture sensor (missing voltage regulator) would cause the sensor to read funny when powered by a battery. This happened because the battery was not a constant voltage and would change over time as it depleted or with temperature (cold temperature in particular). A fix was to use a constant regulated voltage source.

Problem 2 with the capacitive moisture sensor was that some used the wrong timer chip. The correct timer chip was either the “TLC55C or TLC555I” which used 2-15 volts or 3-15 volts which was good since the sensor needed 3.3 volts to be powered. The problem was that on some of the sensors the timer chip used were different models that required 5-15 volts. That of course would cause the moisture sensor to not read or read funny since the timer chip was used to turn a square wave into an analog reading which if not working would cause the issue of the sensor not working.

Problem 3 was the biggest but also had a solution that could fix it. The last problem with the capacitive sensor was that it had a resistor that was in parallel with another capacitor that tied the analog out wire to ground which was left open. This was identified with a continuity check with a multimeter. So one fix was to solder a wire from the resistor on the sensor to the ground pin, or to solder a 1Mega ohm resistor from the ground pin to the analog out pin. We decided to just solder a wire, to resolve the issue.

In the end, we ended up just using 2 of the original working capacitive moisture sensors because we had already used the rest of the analog ports for other sensors and also found out that

the Arduino could not provide enough current for all the sensors and it would cause the program to hang.

Conclusion:

The Smart Irrigation System using a Microcontroller is a successful application of advanced technologies to the area of agriculture. We accomplished the goal stated of creating a system that can monitor and upkeep the plants and environment that they are in. Over a test period of a few weeks the first plant, a Viola, was growing strong. Some ends had to be trimmed due to the container roof it was housed in being restrictive, however new ends are in the process of sprouting. The second plant, a succulent, was seen to be in a healthy state with SISM only needing to water the succulent 1-2 times based upon the determined ratio that was seen by mirroring the values observed from the plant soil moisture directly from the store.

The capability of keeping both plant specimen alive demonstrates that the SISM is able to account for the difference in water intake from both plants and without the necessity of human input, maintaining the specimen group. The completion of the current design phase marks a system that is both complete in its goals as well as a good base for implementing a further specialized system capable of more precise control over the environment. With monitoring devices already in place for the temperature and light, these peripheral units can be utilized to ensure that new additions to the system such as heaters or uv growing lights can be controlled based on the inputs from these sensors.

Development of this system presented challenges that we were able to solve and learn from. Our first finalized design has taught us many skills necessary to be able to work with and design agriculture based microcontroller technologies. In the future, microcontroller based systems will be a backbone in farming technology due to their capability to effectively supply farmers with hands off solutions allowing them to manage the bigger picture of their crop environment. These technologies will be essential in the progression of the human population making the early finding of this project highly important to our engineering education.

References:

Rafiquzzaman, M. *Microcontroller Theory and Applications With the PIC18F*. John Wiley and Sons, 2018.

"In-Depth: Interfacing an I2C LCD with Arduino." *Last Minute Engineers*, 12 Feb. 2023, lastminuteengineers.com/i2c-lcd-arduino-tutorial/.

Admin. "Interface Capacitive Soil Moisture Sensor V1.2 with Arduino." *How To Electronics*, 20 Nov. 2022, how2electronics.com/interface-capacitive-soil-moisture-sensor-arduino/.

"Arduino - 4-Channel Relay Module: Arduino Tutorial." *Arduino Getting Started*, arduinogetstarted.com/tutorials/arduino-4-channel-relay-module. Accessed 1 May 2023.

Code Addendum:

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

LiquidCrystal_I2C lcd(0x27,16,2); // initialize the lcd display with 16 columns and 2 rows

int lightsensorValue = 0; // initialize the variable for the light sensor

const int tempPin = A3; // define the analog input pin for the temperature sensor (LMT84)
float tempC = 0; // initialize the temperature value in Celsius
float tempF = 0; // initialize the temperature value in Fahrenheit

const int btnPin = 12; // define the digital input pin for button 1
const int btn2Pin = 13; // define the digital input pin for button 2
int btnState = 0; // initialize the state of button 1
int btn2State = 0; // initialize the state of button 2
int lcdState = 0; // initialize the state of the lcd display

int OutToRelay1DRY = 2; // define the digital output pin for the dry pump relay
int OutToRelay2Wet = 3; // define the digital output pin for the wet pump relay
int InMoistureSensor1 = A0; // define the analog input pin for the first moisture sensor
int InMoistureSensor2 = A1; // define the analog input pin for the second moisture sensor
float MoistureSensorValue1 = 0; // initialize the value of the first moisture sensor
float MoistureSensorValue2 = 0; // initialize the value of the second moisture sensor
int Moist1Percentage = 0; // initialize the percentage of moisture for the first sensor
int Moist2Percentage = 0; // initialize the percentage of moisture for the second sensor

void setup()
{
  Serial.begin(9600); // initialize serial communication

  lcd.init(); // initialize the lcd display
  lcd.clear(); // clear the lcd display
  lcd.backlight(); // turn on the backlight for the lcd display
  lcd.setCursor(0,0); // set the cursor to the top left corner of the lcd display

  pinMode(A1, INPUT); // set pin A1 as an input
  pinMode(A2, INPUT); // set pin A2 as an input
  pinMode(9, OUTPUT); // set pin 9 as an output
  pinMode(btnPin, INPUT); // set pin btnPin as an input
  pinMode(btn2Pin, INPUT); // set pin btn2Pin as an input
  pinMode(OutToRelay1DRY, OUTPUT); // set pin OutToRelay1DRY as an output
  pinMode(OutToRelay2Wet, OUTPUT); // set pin OutToRelay2Wet as an output
  pinMode(InMoistureSensor1, INPUT); // set pin InMoistureSensor1 as an input
  pinMode(InMoistureSensor2, INPUT); // set pin InMoistureSensor2 as an input

  digitalWrite(OutToRelay1DRY, HIGH); // set the initial state of the dry pump relay to off
  digitalWrite(OutToRelay2Wet, HIGH); // set the initial state of the wet pump relay to off
}
```

```
void loop() {
  // Read temperature from analog pin and convert to Fahrenheit
  int tempRead = analogRead(tempPin);
  float tempmV = ((float) tempRead)*4.9;
  float volt_v = tempmV/1000.0;
  float temp_C = (1035.0 - tempmV)/5.55;
  float temp_F = 1.80 * temp_C + 32.0;

  // Read state of two buttons
  btnState = digitalRead(btnPin);
  btn2State = digitalRead(btn2Pin);

  // Print temperature to serial monitor
  Serial.println("LMT84 Temp Sensor: ");
  Serial.print("Analog: "); Serial.println(tempRead);
  Serial.print("mVolts: "); Serial.println(tempmV,4);
  Serial.print("Temp: "); Serial.println(temp_F, 4);

  // Read sensor data and update moisture percentage values
  sensorRead(lightsensorValue, InMoistureSensor1, InMoistureSensor2);

  // Determine LCD state based on button press
  if (btnState == HIGH) {
    if(lcdState<1) lcdState++;
    else if(lcdState>0) lcdState--;
  }
  if (btn2State == HIGH) {
    if(lcdState>0) lcdState--;
    else if(lcdState<1) lcdState++;
  }

  // Control water pump based on moisture sensor readings
  if(MoistureSensorValue1 > 400) {
    digitalWrite(OutToRelay1DRY,LOW);
  }
  else {
    digitalWrite(OutToRelay1DRY,HIGH);
  }

  if(MoistureSensorValue2 > 300) {
    digitalWrite(OutToRelay2Wet,LOW);
  }
  else {
    digitalWrite(OutToRelay2Wet,HIGH);
  }

  // Update LCD display with sensor data
  printPage(lcdState, temp_F, lightsensorValue, MoistureSensorValue1/10, MoistureSensorValue2/10);

  // Delay for 500 milliseconds before looping again
  delay(500);
}
```

```
// Read moisture sensor data
void sensorRead( int lighsensorValue, int InMoistureSensor1, int InMoistureSensor2){
  // Read light sensor data
  lightsensorValue = analogRead(A2);
  Serial.println(lightsensorValue);

  // Read moisture sensor 1 and print value
  MoistureSensorValue1= analogRead(InMoistureSensor1);
  Moist1Percentage = map(MoistureSensorValue1,199,514,100,0);
  Serial.print("Moisture Sensor 1 Value: ");
  Serial.print(MoistureSensorValue1);

  // Read moisture sensor 2 and print value
  MoistureSensorValue2= analogRead(InMoistureSensor2);
  Moist2Percentage = map(MoistureSensorValue2,186,512,100,0);
  Serial.print("   Moisture Sensor 2 Value: ");
  Serial.println(MoistureSensorValue2);

  // Delay for 1 second before continuing
  delay(1000);
}

void printPage(int lcdState, int temp_F, int lightsensorValue, int MoistureSensorValue1, int MoistureSensorValue2){

  // Check if the lcdState is 0
  if(lcdState == 0){
    // Clear the LCD screen
    lcd.clear();

    // Set the cursor position to display "Plant 1" on the first row
    lcd.setCursor(0, 0);
    lcd.print("Plant 1");

    // Set the cursor position to display "Plant 2" on the first row, 8 characters from the left
    lcd.setCursor(8, 0);
    lcd.print("Plant 2");

    // Set the cursor position to display "Soil:" on the second row, first column
    lcd.setCursor(0, 1);
    lcd.print("Soil:");

    // Display the moisture level of the first plant on the second row, after "Soil:"
    lcd.print(Moist1Percentage);

    // Set the cursor position to display "Soil:" on the second row, eighth column
    lcd.setCursor(8, 1);
    lcd.print("Soil:");
```