AUTOMATED PLANT WATERING SYSTEM

Project by:

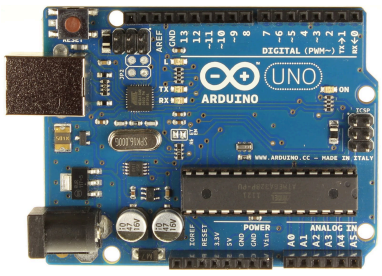
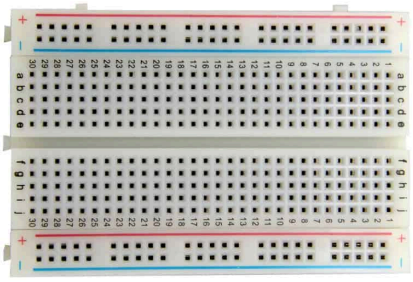
2222620260	2221610039	2221610237	2221610145	2221610187	2221610336
Suvapat Pimklang	Arinpas Jotivej	Paristemi Roe Din	Joseph Miles Nozal	Jirat Kiatsiri	Sora Ito

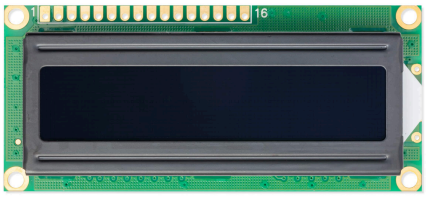
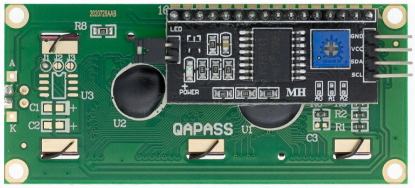
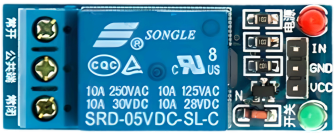
Introduction

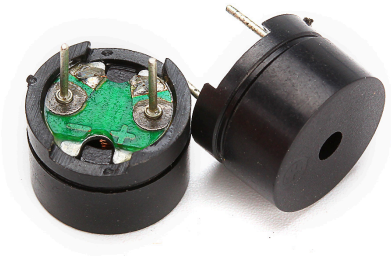
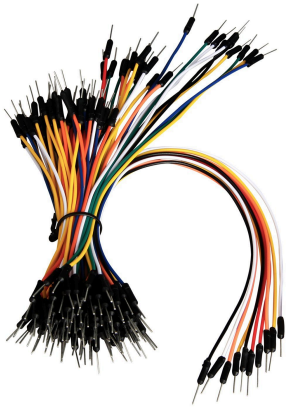
In this project, we will build an automated water planting system using IoT components to monitor and manage soil moisture levels effectively. The system leverages a soil moisture sensor to detect the moisture content in soil, ensuring plants receive water when needed. The sensor's data is displayed on an LCD screen connected via an I2C module, providing clear, real-time feedback on soil conditions.

To enhance functionality, an RGB LED provides visual indicators of moisture levels—changing colour based on dryness or excess water—and a buzzer alerts when immediate attention is required. Through this experiment, we'll learn how to integrate sensors, displays, and alert mechanisms to create a fully functional and interactive plant monitoring system.

Components

No.	Name	Component	Quantity
1.	Arduino Board		**1**
2.	Small Breadboard		**1**

3.	Capacitive Soil Moisture Sensor v2.0		**1**
4.	2*16 LCD Display with I2C Module	 	**1**
5.	5v Relay		**1**
6.	Water Pump Motor		**1**

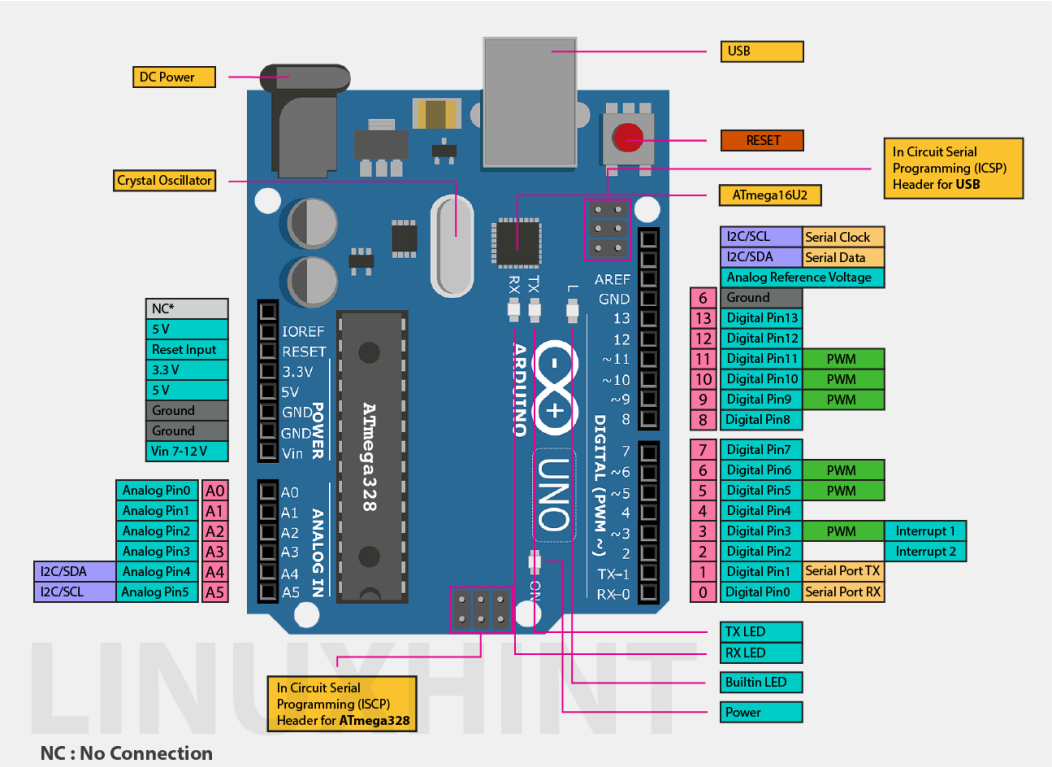
7.	Tactile Push Button		**1**
8.	4 Pin RGB LED		**1**
9.	Passive Buzzer		**1**
10.	Jumper Wires		**23**

Principles

Arduino Board

The Arduino board serves as the central microcontroller, executing code to control and communicate with connected components. It reads input from the soil moisture sensor, processes this data, and manages output to the LCD display, RGB LED, buzzer, and relay. Arduino's versatility and GPIO pins make it suitable for interfacing with various analog and digital components in IoT projects.

Pin Out:



Small Breadboard

A small breadboard is used to arrange and connect components without soldering. It provides a temporary setup that allows easy connections of components, resistors, and wires to the Arduino. Each row of the breadboard is interconnected, allowing components to share power, ground, and data lines.

Capacitive Soil Moisture Sensor v2.0

This sensor detects the moisture level of soil by measuring changes in capacitance as water levels vary. Unlike resistive sensors, which can corrode over time, the capacitive sensor is more durable for long-term soil applications. It outputs analog data to the Arduino, which translates these values into moisture readings displayed on the LCD.

2*16 LCD Display

The 2*16 LCD display is a character display capable of showing two lines of 16 characters. In this project, it displays the soil moisture levels and alerts, helping users monitor plant health. The display connects via the I2C module for simplified wiring and control.

I2C Module

The I2C module is an add-on for the LCD that reduces the number of pins required to control it. It enables communication over

the I2C protocol, which only requires two Arduino pins (SDA and SCL) for data transmission. This setup leaves more pins available for other components.

5V Relay

The relay functions as a switch that enables the Arduino to control high-power devices, such as the water pump. When the soil moisture level is low, the relay activates, allowing current to flow to the water pump to irrigate the soil. Its 5V rating makes it suitable for control by the Arduino's digital output pins.

Motor Water Pump

The water pump is responsible for delivering water to the plant when activated by the relay. This small motorised pump draws water from a reservoir, providing an automated watering system controlled by the soil moisture sensor readings.

Tactile Push Button

The push button serves as a manual input, allowing users to reset or override the watering system if necessary. It works as a simple switch that, when pressed, closes the circuit and sends a signal to the Arduino to initiate a specific function, such as turning off the buzzer or halting the pump.

4-Pin RGB LED

The RGB LED provides visual feedback based on soil moisture levels. It contains three colour channels (Red, Green, and Blue) that can combine to create various colours. Each colour can represent a specific soil condition, helping users visually monitor the plant's status at a glance.

Passive Buzzer

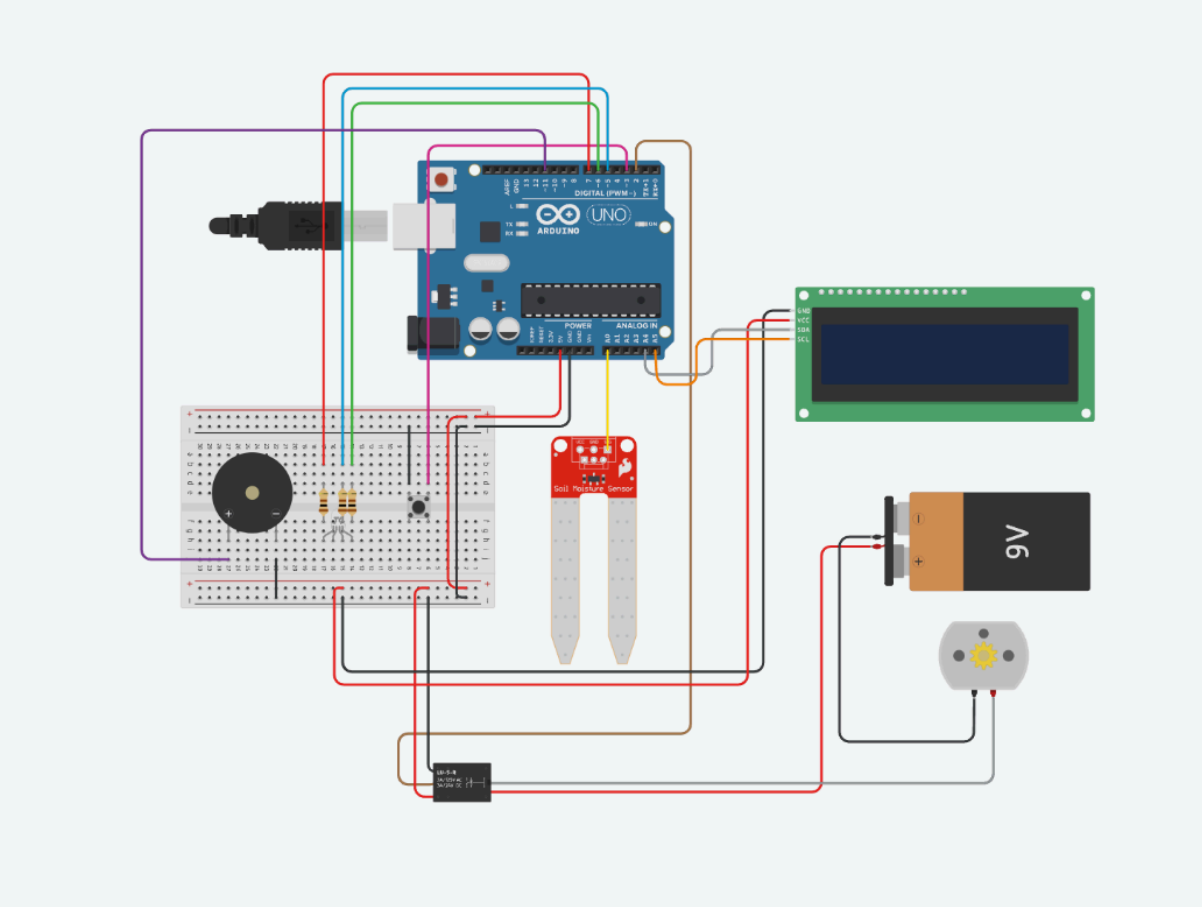
The passive buzzer emits sound alerts based on soil conditions. Unlike an active buzzer, a passive buzzer requires external signals (frequency) from the Arduino to produce sound. In this project, it serves as an auditory alert, signalling when soil is too dry or too moist.

Software Preparation

Head to https://www.arduino.cc/en/software and download according to your operating system. There are options for Windows, Linux, and macOS. After downloading, run the app and follow the instructions shown on the screen to finish setting up. There's no need to donate but if you'd like to go ahead. If not keep clicking "JUST DOWNLOAD"

Experimental Procedure

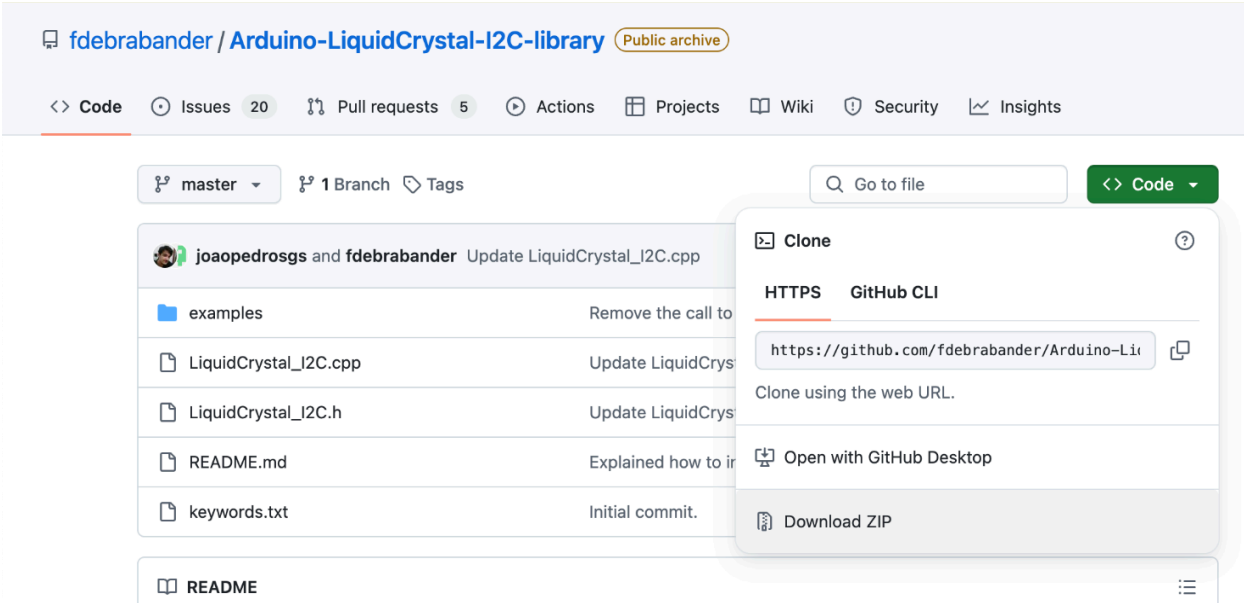
Step 1: Build the circuit

Arduino	
| 5v | - - - - - > [Positive Pin Breadboard] | GND | - - - - - >[Negative Pin Breadboard] | A0 | - - - - - >[Moisture Sensor] | A4 | - - - - - >[LCD SDA pin] | A5 | - - - - - >[LCD SCL pin]	| ~11 | - - - - - >[Buzzer] | 7 | - - - - - >[RGB LED – Blue] | 6 | - - - - - >[RGB LED – Green] | 5 | - - - - - >[RGB LED – Red] | ~3 | - - - - - >[Tactile Push Button] | 2 | - - - - - >[5v Relay]
Circuit Diagram	
	

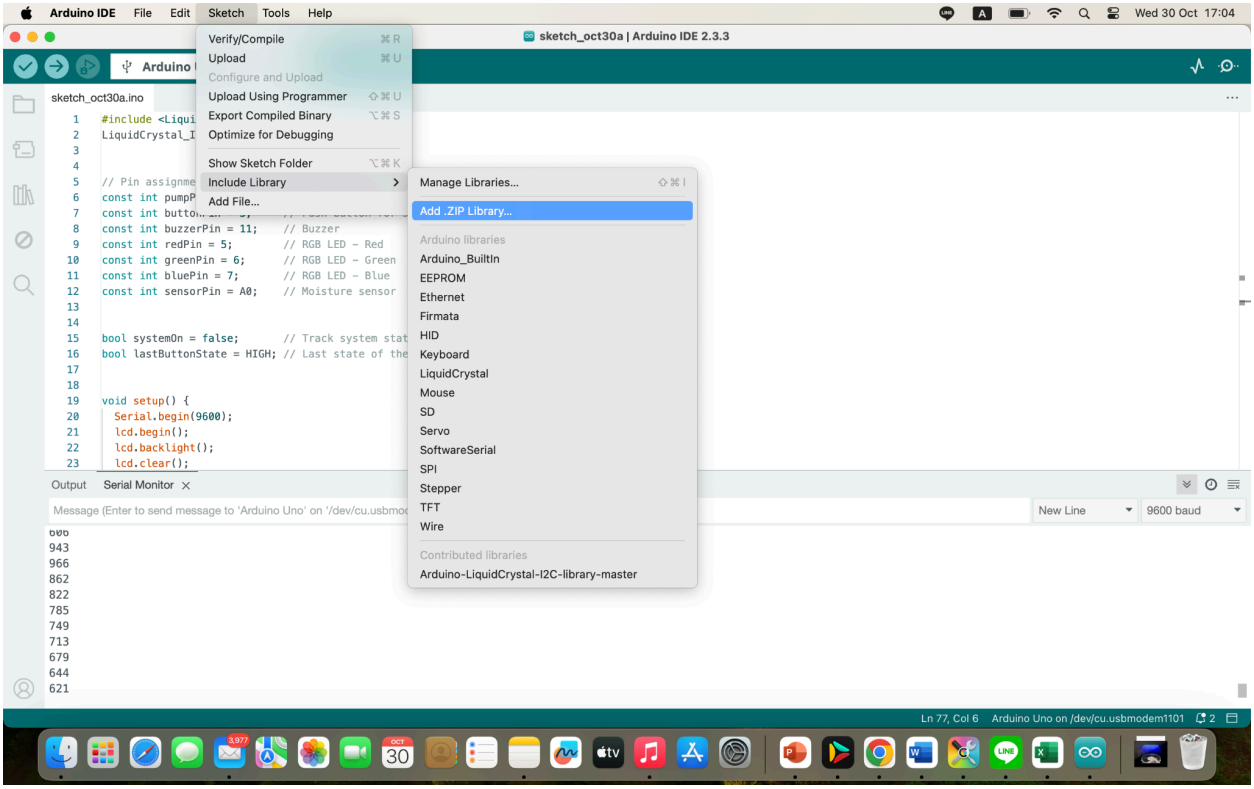
The visual circuit uses rgb led with no module therefore using resistors. Make sure to connect the right pins to the board
*The Relay SPDT represents the 5v Relay, The DC Motor represents the water pump***

Step 2: Download Library

Open https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library and download the zip file.



Open up Arduino IDE. Go to Sketch > Include Libraries > Add .ZIP Library... Search for the downloaded zip file and select.



Step 3: Copy code + Explanation

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Pin assignments
const int pumpPin = 2;
const int buttonPin = 3;
const int buzzerPin = 11;
const int redPin = 5;
const int greenPin = 6;
const int bluePin = 7;
const int sensorPin = A0;

bool systemOn = false; // System OFF initially
unsigned long lastMoistureCheck = 0; // Time of last moisture check
const unsigned long moistureCheckInterval = 5000; // 5-second interval for moisture
check
bool lastButtonState = HIGH; // Last button state for debouncing

void setup() {
 Serial.begin(9600);
 lcd.begin();
 lcd.backlight();
 lcd.clear();

 pinMode(pumpPin, OUTPUT);
 digitalWrite(pumpPin, HIGH);  // Pump OFF initially

 pinMode(buttonPin, INPUT_PULLUP); // Button with pull-up resistor
 pinMode(buzzerPin, OUTPUT);
 pinMode(redPin, OUTPUT);
 pinMode(greenPin, OUTPUT);
 pinMode(bluePin, OUTPUT);

 lcd.setCursor(0, 0);
 lcd.print("IRRIGATION");
 lcd.setCursor(0, 1);
 lcd.print("SYSTEM READY");
 delay(3000);
 lcd.clear();
}
```

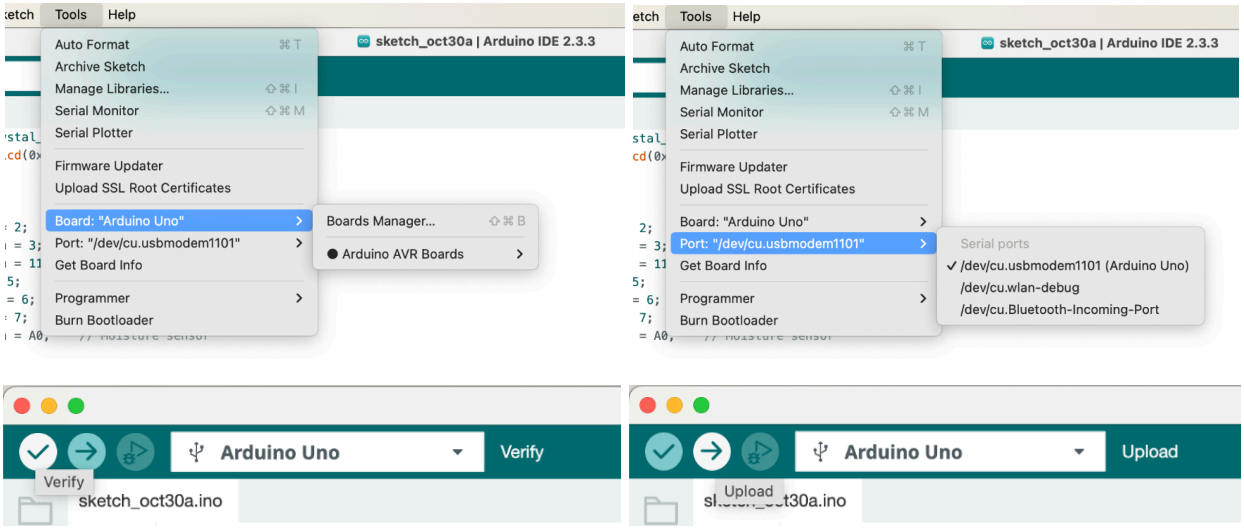
```
void loop() {
 // Button toggle handling
 bool buttonState = digitalRead(buttonPin);
 if (buttonState == LOW && lastButtonState == HIGH) {
   systemOn = !systemOn; // Toggle system state
   Serial.println(systemOn ? "System ON" : "System OFF");
   lcd.clear();
   delay(500); // Simple debounce delay
 }
 lastButtonState = buttonState;

 if (systemOn) {
   lcd.setCursor(0, 0);
   lcd.print("System ON");

   // Check moisture level at intervals
   if (millis() - lastMoistureCheck >= moistureCheckInterval) {
     lastMoistureCheck = millis(); // Reset the moisture check timer

     int moistureValue = analogRead(sensorPin);
     Serial.println(moistureValue);

     if (moistureValue < 290) { // Too much water
       digitalWrite(pumpPin, HIGH);  // Ensure pump is OFF
       lcd.setCursor(0, 1);
       lcd.print("Too Much Water");
       analogWrite(redPin, 255);
       analogWrite(greenPin, 0);
       analogWrite(bluePin, 0);

       for (int i = 0; i < 2; i++) {
         tone(buzzerPin, 1000);
         delay(200);
         noTone(buzzerPin);
         delay(200);
       }
     }
     else if (moistureValue >= 290 && moistureValue <= 400) { // Well hydrated
       digitalWrite(pumpPin, HIGH); // Turn OFF pump
       lcd.setCursor(0, 1);
```

```
      lcd.print("Well Hydrated");
      analogWrite(redPin, 0);
      analogWrite(greenPin, 255);
      analogWrite(bluePin, 0);
    }
    else if (moistureValue > 400 && moistureValue <= 599) { // Medium range
      digitalWrite(pumpPin, HIGH);
      lcd.setCursor(0, 1);
      lcd.print("Hydrated");
      analogWrite(redPin, 0);
      analogWrite(greenPin, 0);
      analogWrite(bluePin, 255);
    }
    else { // Low moisture
      digitalWrite(pumpPin, LOW); // Turn ON pump
      lcd.setCursor(0, 1);
      lcd.print("Dehydrated!");
      lcd.setCursor(0, 2);
      lcd.print("Watering...");

      for (int i = 0; i < 5; i++) {  // Blink 5 times
        analogWrite(redPin, 0);
        analogWrite(greenPin, 0);
        analogWrite(bluePin, 255);
        delay(200);
        analogWrite(bluePin, 0);
        delay(200);
      }

      analogWrite(redPin, 255);
      analogWrite(greenPin, 0);
      analogWrite(bluePin, 0);

      for (int i = 0; i < 2; i++) {
        tone(buzzerPin, 1000);
        delay(100);
        noTone(buzzerPin);
        delay(100);
      }
    }
  }
```

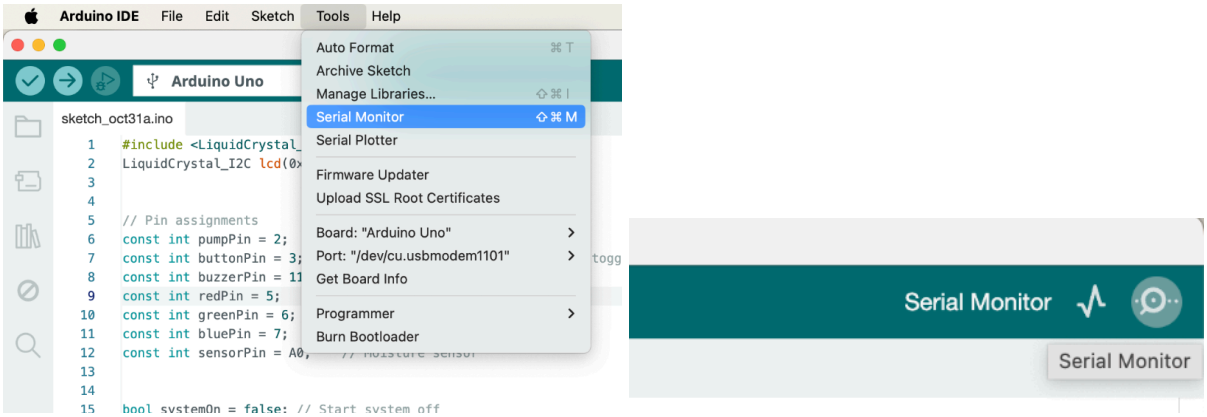
```
  }
  else {
    lcd.setCursor(0, 0);
    lcd.print("System OFF");
    digitalWrite(pumpPin, HIGH);
    noTone(buzzerPin);
    analogWrite(redPin, 0);
    analogWrite(greenPin, 0);
    analogWrite(bluePin, 0);
    delay(1000);
  }
}
```

Step 3: Configuration

Check that your Board and Port are properly connected. Verify your code, then upload.

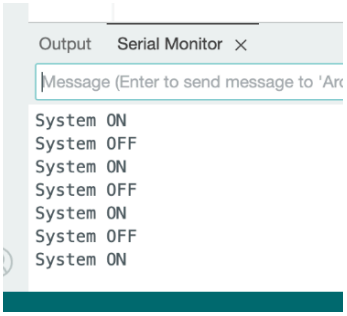
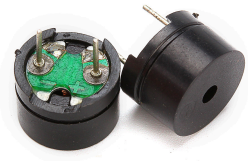


To see the serial monitor, open Tools > Serial Monitor or Click the top right icon



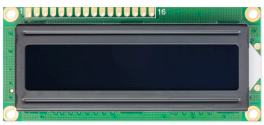
Testing and Troubleshooting

Before building the circuit, testing each component ensures there's no issue after building the circuit.

Components	Basic Code Testing	Intended Output
	`const int buttonPin = 3;` `bool systemOn = false;` `void setup() {` `Serial.begin(9600);` `pinMode(buttonPin, INPUT_PULLUP);` `}` `void loop() {` `if (digitalRead(buttonPin) == LOW)` `{` `systemOn = !systemOn; // Toggle system state` `Serial.println(systemOn ? "System ON" : "System OFF");` `delay(500);` `}` `}`	 When the system starts in the "OFF" state, pressing the button should switch it to "ON." Each press alternates between "System ON" and "System OFF" messages in the Serial Monitor, confirming the toggle.
	`const int buzzerPin = 11;` `void setup() {` `pinMode(buzzerPin, OUTPUT);` `}` `void loop() {` `for (int i = 0; i < 2; i++) {` `tone(buzzerPin, 1000);` `delay(100);` `noTone(buzzerPin);` `delay(100);` `}` `delay(2000);` `}`	The buzzer emits two quick beeps, pauses, and repeats the sequence, serving as an alert signal. This cycle provides a clear auditory indicator for specific conditions.

<table>
<tr>
<td>  </td>
<td>

```
const int redPin = 7;
const int greenPin = 6;
const int bluePin = 5;

void setup() {
 pinMode(redPin, OUTPUT);
 pinMode(greenPin, OUTPUT);
 pinMode(bluePin, OUTPUT);
}

void loop() {

 analogWrite(redPin, 255);
 analogWrite(greenPin, 0);
 analogWrite(bluePin, 0);
 delay(1000);
 analogWrite(redPin, 0);
 analogWrite(greenPin, 255);
 analogWrite(bluePin, 0);
 delay(1000);

 analogWrite(redPin, 0);
 analogWrite(greenPin, 0);
 analogWrite(bluePin, 255);
 delay(1000);
}
```

</td>
<td>The RGB LED lights up in red, green, and blue sequentially, with each colour staying on for one second before cycling again. This colour-changing pattern confirms the LED's functionality.</td>
</tr>
<tr>
<td> 
  </td>
<td>

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
 lcd.begin();
 lcd.backlight();
 lcd.setCursor(0, 0);
 lcd.print("Hello, World!");
}

void loop() {
}
```

</td>
<td> 

The LCD screen displays "Hello, World!" on the first row, verifying that the LCD and I2C module are correctly configured.</td>
</tr>
</table>

	``` const int relayPin = 2; void setup() { pinMode(relayPin, OUTPUT); } void loop() { digitalWrite(relayPin, LOW); delay(2000); digitalWrite(relayPin, HIGH); delay(2000); } ```	The relay toggles on for two seconds, then off for two seconds, looping continuously. This ensures the relay is prepared to control the pump as expected.
	Carefully connect the wires to the negative and positive terminals of the 9V battery, with the red wire connected to positive and the black wire to negative.	When connected to a 9V battery, the water pump should start vibrating, indicating it is ready to provide water flow upon relay activation.

Conclusion

In this project, we successfully designed an automated irrigation system using IoT components to monitor and manage soil moisture. By integrating a soil moisture sensor, RGB LED, buzzer, relay, and LCD display, we developed a system that efficiently maintains plant health with minimal human intervention. The project demonstrated essential skills in circuit design, sensor integration, and programming for real-time environmental monitoring as well as enhanced our skills in circuit design, sensor application, and programming, highlighting the potential of IoT for resource-efficient, scalable agricultural management. This system provides a practical and effective foundation for broader applications in automated environmental monitoring.

Note

There is a drip bowl located under the plant, inside the mountain to collect excess water from the pot drainage holes.

After every use, ensure that the drip bowl is empty. In the case that there is water, utilise the water pump motor and moisture sensor to drain it out. Use a paper towel to completely dry the bowl.