EMBEDDED SYSTEM AND DESIGN
EXPERIMENT NO. 7

Watering Plant System

Name: NAVARRO, ROD GERYK C.

Course/Section: CPE160P-4/A1

Group No.: N/A

Date of Performance: 10/16/2024
Date of Submission: 11/03/2024

CYREL O. MANLISES, PH.D.
Instructor

DISCUSSION

In the experiment titled "Watering Plant System," I focused on building an automated plant irrigation setup using an Arduino UNO, an LCD display, a soil moisture sensor, and a servo motor. The experiment aimed to monitor soil moisture levels and activate a watering mechanism when necessary. The setup involved testing five soil conditions: very low, low, mid, normal, and high moisture.

The first part of the experiment involved constructing the circuit. I connected all components to the Arduino UNO and carefully arranged them on the breadboard to ensure correct wiring. This included connecting the LCD display, servo motor, and soil moisture sensor to their appropriate pins on the microcontroller. Once the circuit was complete, I wrote the code to operate the system.

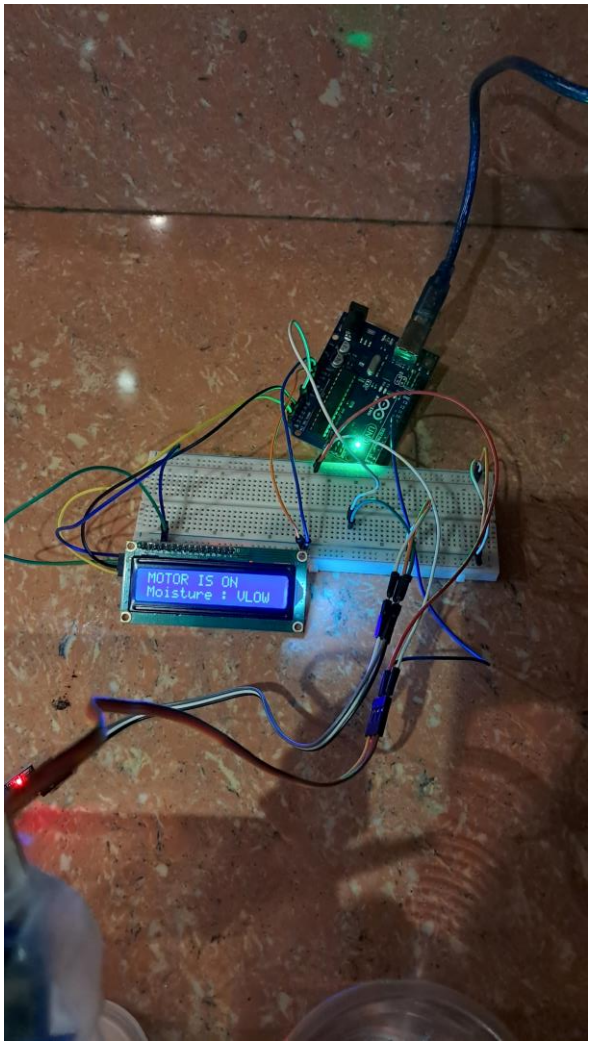

Figure 1: *The connection of the LCD, servo motor, and soil moisture sensor.*

The code begins by including the necessary libraries for the LCD display (“LiquidCrystal_I2C.hβ) and servo motor. The display is initialized with an I2C address and set up for 16x2 characters, while the servo motor is attached to pin 7. The setup() function initializes the components, displays a startup message, and simulates a loading animation.

Navarro_EXP7.ino

```
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
LiquidCrystal_I2C dis(0x27, 16, 2);
Servo myservo1;
int pos = 0;
int cm = 0;



void setup() {
myservo1.attach(7);
Serial.begin(9600);
dis.init();
dis.backlight();
dis.clear();
pinMode(2, OUTPUT);
digitalWrite(2, HIGH);
delay(1000);
dis.setCursor(0, 0);
dis.print(" MY IRRIGRATION");
dis.setCursor(0, 1);
dis.print("SYSTEM IS ON ");
for (int a = 12; a <= 15; a++) {
  dis.setCursor(a, 1);
  dis.print(".");
  delay(1500);
}
dis.clear();

}

void loop() {
int value = analogRead(A0);
```

Figure 2: The initialization of variables and components as well as the acquisition of sensor reading, and motor arm positioning.

In the loop(), the system reads the soil moisture level using analogRead(A0) and displays the value on the serial monitor for observation. The code then evaluates the moisture reading and decides the action based on set thresholds:

Navarro_EXP7.ino

```
31
32  void loop() {
33  int value = analogRead(A0);
34  Serial.println(value);
35
36  if (value >= 900){
37    dis.setCursor(0, 1);
38    dis.print("Moisture : VLOW ");
39    digitalWrite(2, LOW);
40    dis.setCursor(0, 0);
41    dis.print("MOTOR IS ON ");
42
43    for (pos = 0; pos <= 120; pos += 1){
44      myservo1.write(pos);
45      delay(19);
46    }
47    delay(1000);
48
49    for (pos = 120; pos >= 0; pos -= 1){
50      myservo1.write(pos);
51      delay(5);
52    }
53    delay(1000);
54  } else if (value >= 700) {
55    dis.setCursor(0, 1);
56    dis.print("MOISTURE : LOW ");
57    digitalWrite(2, LOW);
58    dis.setCursor(0, 0);
59    dis.print("MOTOR IS ON ");
60
61    for (pos = 0; pos <= 120; pos += 1){
62      myservo1.write(pos);
63      delay(19);
```

Output

Figure 3: The continuation of the code showing the conditions that evaluates the moisture reading and the action based on set thresholds.

Navarro_EXP7.ino

```
    dis.print("MOISTURE : MID  ");
    digitalWrite(2, LOW);
    dis.setCursor(0, 0);
    dis.print("MOTOR IS ON ");

    for (pos = 0; pos <= 120; pos += 1) {
      myservo1.write(pos);
      delay(19);
    }
    delay(1000);

    for (pos = 120; pos >= 0; pos -= 1) {
      myservo1.write(pos);
      delay(5);
    }
    delay(1000);
  }
    else if (value >= 310){
      dis.setCursor(0, 1);
      dis.print("MOISTURE : NORMAL ");
      digitalWrite(2, HIGH);
      dis.setCursor(0, 0);
      dis.print("MOTOR IS OFF ");
    }

    else {
      dis.setCursor(0, 1);
      dis.print("MOISTURE : HIGH ");
      digitalWrite(2, HIGH);
      dis.setCursor(0, 0);
      dis.print("MOTOR IS OFF ");
    }

```

Figure 4: The continuation of the code showing the rest of the conditions and thresholds.

Very Low (β= 900): The LCD displays "Moisture: VLOW," the motor is activated, and water is dispensed. Low (β= 700): The LCD displays "MOISTURE: LOW," and the motor is similarly activated to water the soil. Mid (β= 500): The LCD shows "MOISTURE: MID," with the motor turning on to irrigate. Normal (β= 310): The LCD displays "MOISTURE: NORMAL," indicating that no watering is needed, and the motor remains off. High (≥ 310): The LCD shows "MOISTURE: HIGH," signaling that the soil has enough moisture and the motor stays off.

The code controls the servo motor to rotate from 0 to 120 degrees to simulate pouring water, followed by a return to its initial position. Delays are included to ensure the servo moves smoothly.

After writing and uploading the code, I attached the servo motor to a container and affixed a straw to the servo arm to create a basic watering mechanism. This setup allowed for water to be directed to the soil samples. I tested the system with the five soil conditions to verify if the motor activated appropriately when the soil moisture was below normal, ensuring effective irrigation. The experiment successfully demonstrated an automated watering system that responded to different soil moisture levels, with the servo motor functioning as the actuator for water distribution.

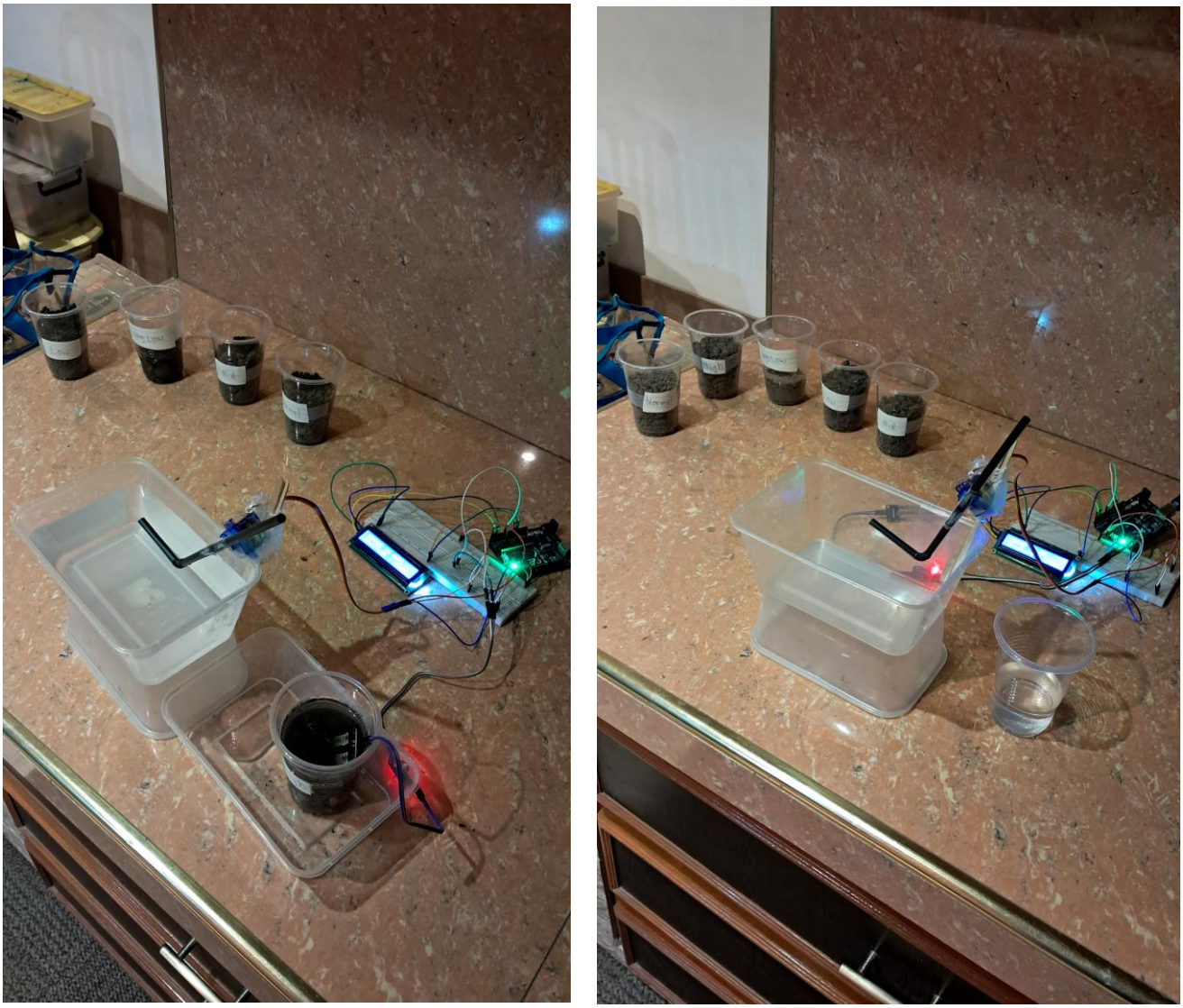

Figure 5: *The setup of the system as well as the watering mechanism for the soil.*

CONCLUSION

To conclude, the "Watering Plant System" experiment 7, effectively showcased how automation can simplify plant irrigation. The system was able to monitor different levels of soil moisture and respond by activating the servo motor to water the soil only when needed. This experiment highlighted the importance of precise coding and wiring to ensure the system worked as intended. The LCD display provided clear feedback on the soil's moisture condition, making it easy to understand when watering was necessary or when the soil was sufficiently hydrated. The experiment demonstrated that an Arduino-based system could successfully manage plant watering automatically, reducing the need for manual intervention. This project serves as a useful prototype for more complex irrigation systems, showing how simple components and coding can be used to create practical solutions for everyday problems like plant care.