MEMORANDUM

TO: Prof. Hossein Asghari, PhD
FROM: Nicole Levine, Makenna Golding, Mike Hennessy
DATE: November 29, 2021
SUBJECT: Smart Water Design

Introduction

The objective of this model was to create a smart house plant system that tracks moisture levels, water levels, and sunlight of a plant and alerts a user when those levels are not within thresholds. The materials used included an Arduino, water level sensor, LCD, buzzer, LED lights, photoresistor, moisture level sensor, various wires, and a circuit breadboard. There were two aspects to the project: wire the sensors correctly to the breadboard and Arduino and code the Arduino correctly to achieve the desired output. The thresholds used were calibrated using sample code that was given. The final system, as seen in figure 1, worked successfully, through much trial and error, using ground dirt and tap water. The code is referenced throughout and the full code is attached at the end of the document.

Deliverable 1

The first sensor put into place was the moisture level sensor. The sensor includes three ports: “GND” (black), “VCC” (red), and “AOUT” (yellow). From figure 4, the “AOUT” port got connected to the “A0” port on the Arduino, with the “GND” port being connected to the “GND” on the Arduino, and the “VCC” port was connected to the “5V” (5 volts) power port on the Arduino. After connecting the jumper wires, the moisture sensor then outputs the moisture level of the product into the Arduino.

The moisture sensor was tested using a cup of fresh dirt from outside. The front end of it was placed into the dirt and the arduino then gave an output for the moisture level.

Deliverable 2

The moisture level outputs were represented with an LCD. In terms of hardware, the LCD included three ports: “GND” (blue), “VCC” (red), “SDA” (orange), and “SCL” (green). For this to be accomplished, as seen in figure 4, the “GND”, “SDA”, and “SCL” ports of the LCD were connected to the corresponding ports on the Arduino with the same names, while the power output, “VCC”, was connected to the “5V” port of the Arduino.

The following code was used to print the values of the moisture sensor onto the LCD, where “ML” stands for “Moisture Level”:

```
{
lcd.setCursor(0,0);
lcd.print("ML:" + moistureLevel + "   WL:" + waterLevel)
```

```
    lcd.println("LL: " + lightLevel)
    }
```
The following statement was used to update the system on the current moisture level:
```
    moistureLevel = analogRead(moistureSensor);
```
And finally, the following statement was used to define the moisture sensor as the port “A0” in the Arduino:
```
    #define moistureSensor A0
```
Through this code, the system updated the moisture level each time it ran and displayed the number on the LCD screen for the user to see.

Deliverable 3

Aside from the LCD, the system showed the user the moisture level status using three LED lights: green, yellow, and red. The green LED turned on if the moisture level was in a saturated range, above the threshold of 553. The yellow LED turned on if the moisture level was between a dry and saturated range, between the thresholds of 365 and 553. Finally, the red LED turned on if the moisture level was in a dry range, under the threshold of 365.

As seen in figure 6, the long ends of the LED lights were placed in the “12” (green), “13” (yellow), and “2” (red) power inputs of the Arduino. The short ends were placed in the “GND” ground output of the Arduino. One resistor went through each LED light.

The thresholds were calibrated using a cup of dirt from outside. Saturated dirt was tested to see the moisture level using our moisture sensor and the same was done with dry dirt. Using these numbers, the thresholds were found to be 365 on the low end and 553 on the high end.

The following code set the thresholds for the system and defined the LED lights within the Arduino:
```
    int lowMoistureThreshold = 365;
    int highMoistureThreshold = 553;

    #define green1 12
    #define yellow1 13
    #define red1 2
```

After checking and updating the moisture level, the following code turned on the corresponding LED light based on the level:
```
     if(moistureLevel > highMoistureThreshold){
       digitalWrite(green1, HIGH);
       digitalWrite(yellow1, LOW);
       digitalWrite(red1, LOW);
```

```
} else if(moistureLevel < lowMoistureThreshold){
  digitalWrite(red1, HIGH);
  digitalWrite(green1, LOW);
  digitalWrite(yellow1, LOW);
  tone(buzzer, 10, 100);
} else{
  digitalWrite(yellow1, HIGH);
  digitalWrite(green1, LOW);
  digitalWrite(red1, LOW);
}
```

Deliverable 4
Using a buzzer, the system alerted the user when the plant was too dry. When the moisture level got too low, the buzzer went off. The threshold for low moisture was set at 365, as explained in deliverable 3.

As seen in figure 6, the buzzer had two wires to be connected. One side of the buzzer was connected to the “GND” ground output of the Arduino. The other end was placed in the “7” power input of the Arduino. It took two buzzers because the buzzers were fragile and the first one broke when attempting to attach it to the breadboard. The buzzer did not fit cleanly into the breadboard so it needed to be pushed into exactly diagonal ports.

The buzzer was defined and named in the following line of code:

```
#define buzzer 7
```

After the moisture level was checked, a function was created to have the buzzer go off for 100 milliseconds if the moisture level was below the desired threshold and the plant was dry:

```
else if(moistureLevel < lowMoistureThreshold){
   digitalWrite(red1, HIGH);
   digitalWrite(green1, LOW);
   digitalWrite(yellow1, LOW);
   tone(buzzer, 10, 100);
}
```

Deliverable 5
Using a red, yellow, and green LED, along with a buzzer was how the system alerts the user of the water level in the tank. The green light indicates that the water level was above a threshold value, meaning that the tank was full. Yellow was used to show the inbetween. The water was not too full, nor was it too low. Finally, the red LED was used to indicate not enough water in the

plant. If ever a red light illuminated, a buzzer would also sound, letting the user know that the water level was too low and it has to be refilled.

The code as follows is for each of the three colored lights to indicate water level:

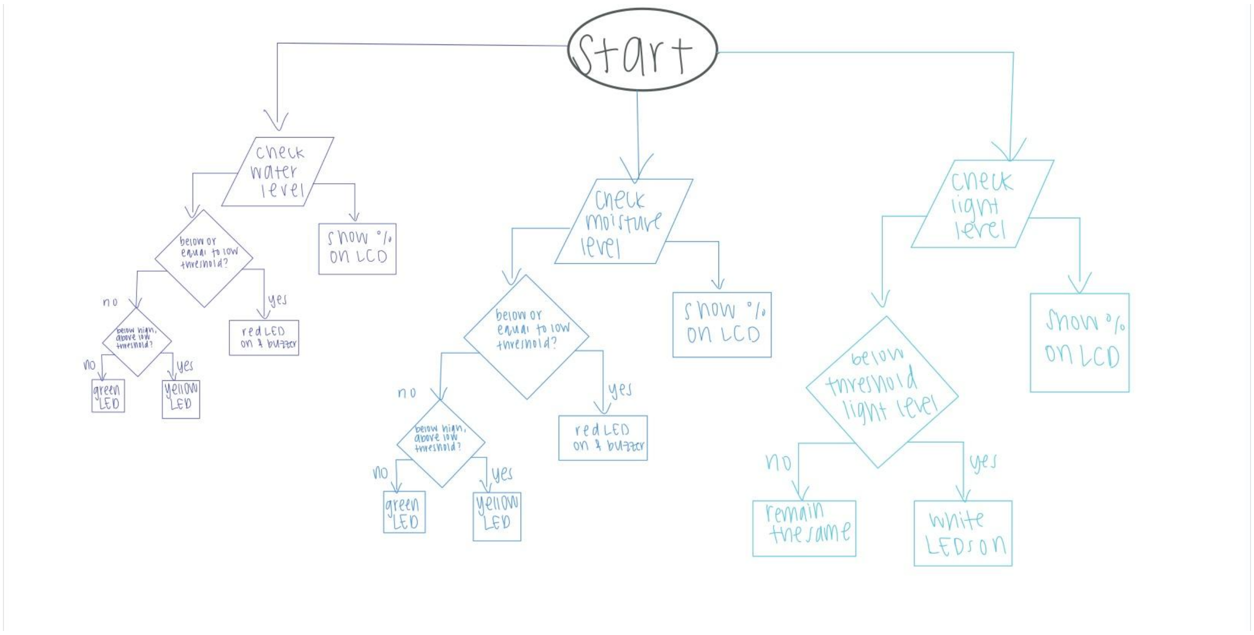
```
//check water level
  if(waterLevel > highWaterThreshold){
    digitalWrite(green2, HIGH);
    digitalWrite(yellow1, LOW);
    digitalWrite(red1, LOW);
  } else if(waterLevel < lowWaterThreshold){
    digitalWrite(red2, HIGH);
    digitalWrite(green1, LOW);
    digitalWrite(yellow1, LOW);
    tone(buzzer, 10, 100);
  }
  else{
    digitalWrite(yellow2, HIGH);
    digitalWrite(green1, LOW);
    digitalWrite(red1, LOW);
  }
```

Deliverable 6
The following code is what was used in order to check the light level of the plant.

```
//check light
  if(light < lightThreshold){
    digitalWrite(white1, HIGH);
    digitalWrite(white2, HIGH);
    digitalWrite(white3, HIGH);
  }
  else{
    digitalWrite(white1, LOW);
    digitalWrite(white2, LOW);
    digitalWrite(white3, LOW);
  }
```

If the data of the light near the plant was reported to be lower than the threshold value, then 5 white LED lights will illuminate in order to give the plant extra light until the designated light levels return to the threshold value.

Deliverable 7



Conclusion

By using the given materials and procedure, a smart water housing system was made. Figures 4-7 helped to put the arduino together, which ultimately would hold each part of the system. Prior knowledge as well as learning new codes in Intro to Engineering were used in order to create the final code to run the system. Problems such as a broken buzzer and too messy of a breadboard resulted in starting over or adapting to what was available. The biggest problem was that the first arduino given to us overheated and the whole set-up had to be put into a new board. This project included a lot of attention to detail due to the use of the arduino and the breadboard.

Recommendation

It is very important to keep a clean and organized breadboard. There are many wires to attend to, so having them color coded and spaced out evenly makes it much easier to find errors. This is the same for the code. Keep a clean and organized code in order to find errors within the code and exactly where they occur. Also, place comments in the code that explain what the following lines are doing, and that way it is also easier to find errors and understand what the code is doing. As another recommendation, go through each part of the design in order. For example, get the moisture sensor working before starting on the water sensor. On the same note, write all of the code for the moisture sensor before starting the code for the water sensor. By doing the parts chronologically, it is easier to find errors within each part of the project and get the arduino working most efficiently.

References

Figures 4-7 from the Slideshow in the Electrical Engineering Session
Dr. Asghari's Slides from the Electrical Engineering Sessions

Appendix

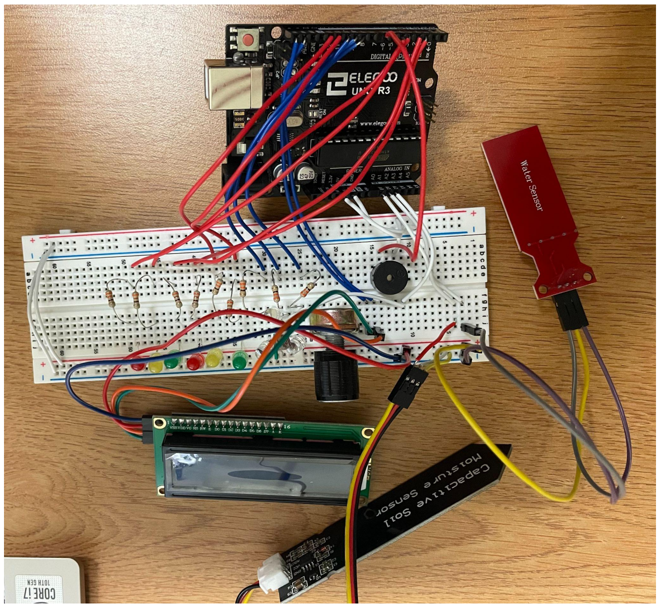


Figure 1: Working setup of Arduino and with the LCD, moisture sensor, water level sensor, and circuit breadboard.

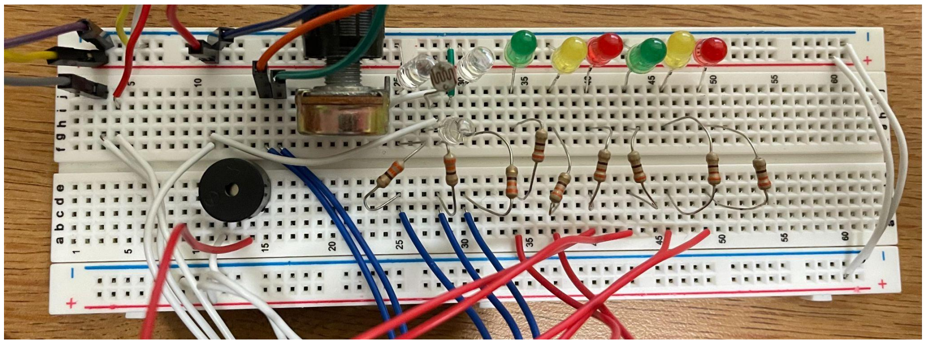


Figure 2: Working setup of the breadboard with the lights, resistors, buzzer, and photoresistor.

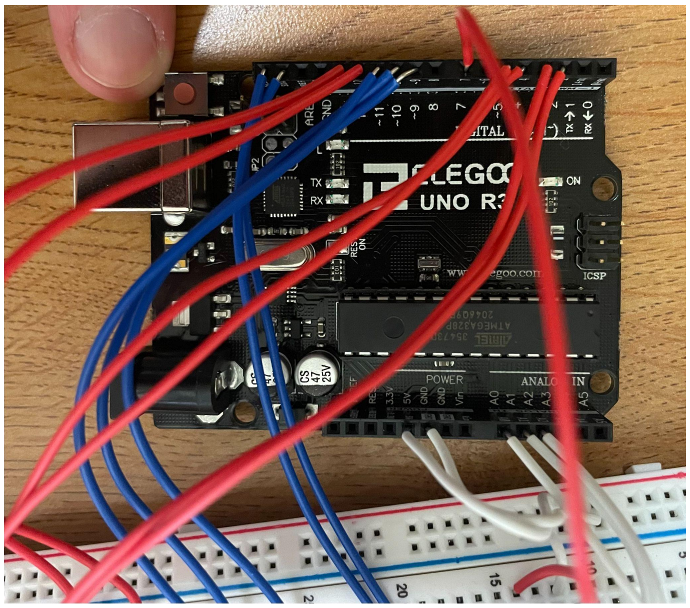


Figure 3: Working set up of the Arduino with color-coordinated blue, white, and red wires.

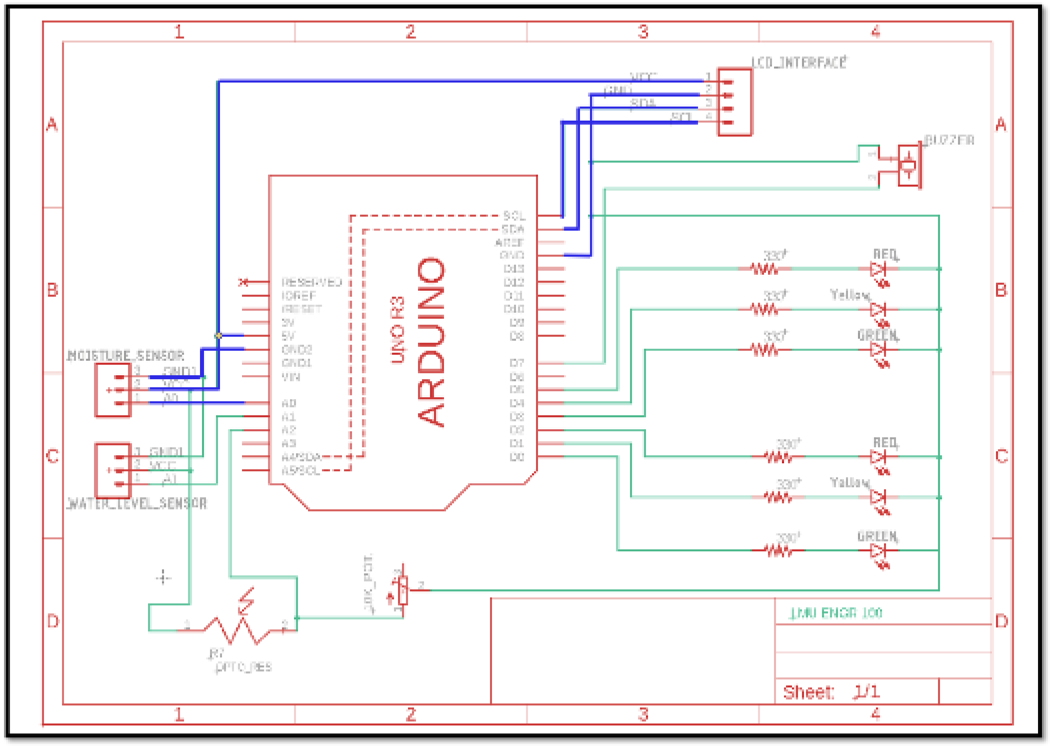


Figure 4: Set up of the moisture sensor on the Arduino.

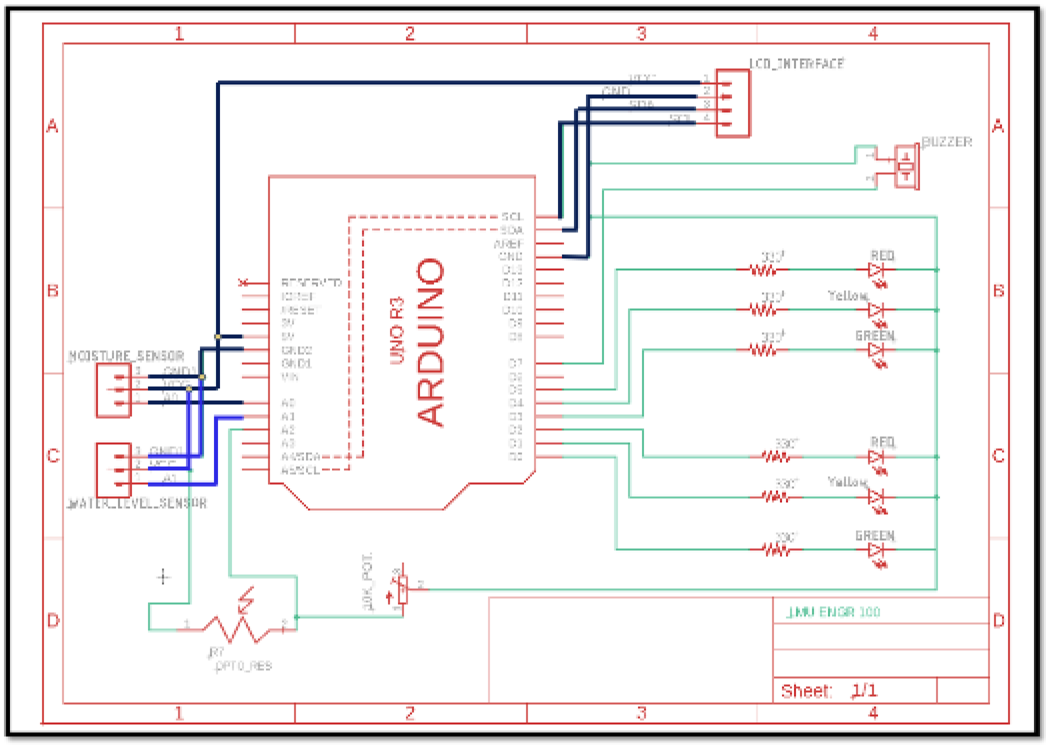


Figure 5: Set up of the water level sensor on the Arduino.

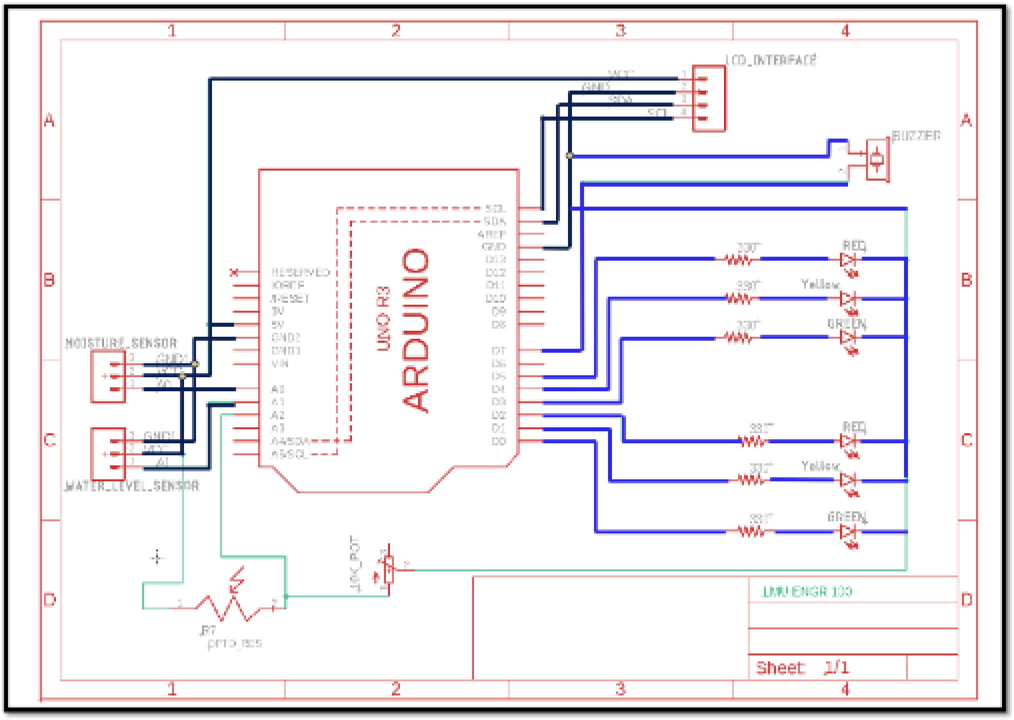


Figure 6: Set up of the LEDs and buzzers on the Arduino.

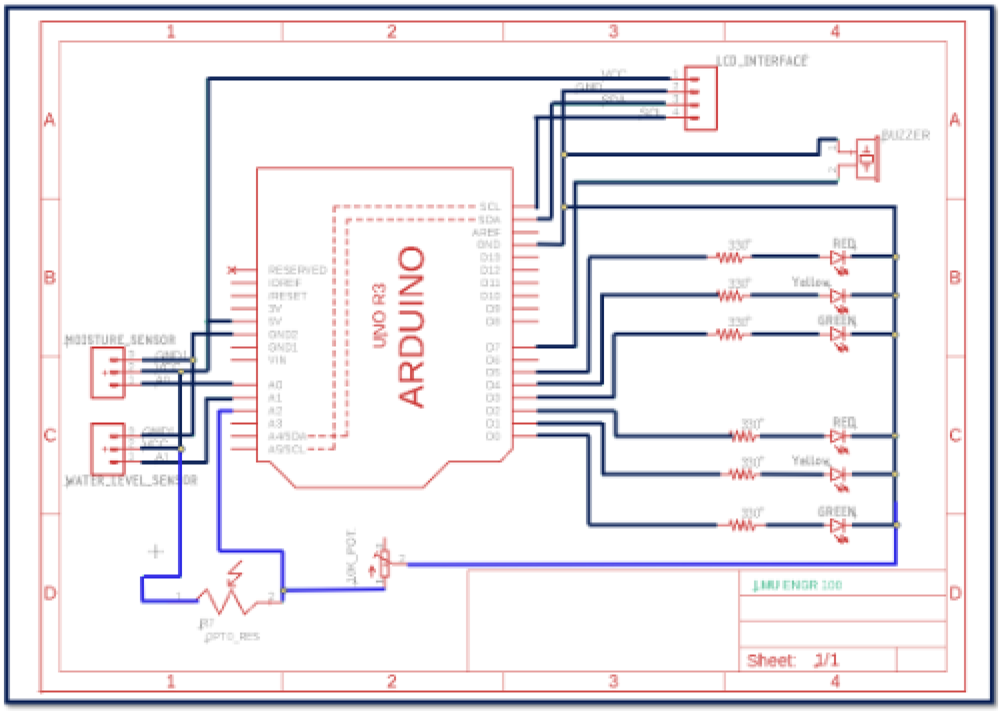


Figure 7: Set up of the photoresistor sensor on the Arduino.

Full Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

float waterLevel;
float moistureLevel;
float light;
int lowWaterThreshold = 1;
int highWaterThreshold = 414;
int lowMoistureThreshold = 365;
int highMoistureThreshold = 553;
int lightThreshold = 500;

#define green1 12
#define yellow1 13
#define red1 2
#define green2 3
#define yellow2 4
#define red2 5
#define white1 10
```

```
#define white2 11
#define white3 12
#define waterLevelSensor A1
#define moistureSensor A0
#define lightSensor A2
#define buzzer 7

LiquidCrystal_I2C lcd(0x27,20,4);

void setup() {
  // put your setup code here, to run once:
  pinMode(moistureSensor, INPUT);
  pinMode(waterLevelSensor, INPUT);
  pinMode(lightSensor, INPUT);
  pinMode(green1, OUTPUT);
  pinMode(yellow1, OUTPUT);
  pinMode(red1, OUTPUT);
  pinMode(green2, OUTPUT);
  pinMode(yellow2, OUTPUT);
  pinMode(red2, OUTPUT);
  pinMode(white1, OUTPUT);
  pinMode(white2, OUTPUT);
  pinMode(white3, OUTPUT);

  lcd.init();
  lcd.backlight();
}

void loop() {
  // put your main code here, to run repeatedly:
  updateSensors();

 //check moisture
  if(moistureLevel > highMoistureThreshold){
    digitalWrite(green1, HIGH);
    digitalWrite(yellow1, LOW);
    digitalWrite(red1, LOW);
  } else if(moistureLevel < lowMoistureThreshold){
    digitalWrite(red1, HIGH);
    digitalWrite(green1, LOW);
    digitalWrite(yellow1, LOW);
    tone(buzzer, 10, 100);
  } else{
    digitalWrite(yellow1, HIGH);
    digitalWrite(green1, LOW);
    digitalWrite(red1, LOW);
  }

//check water level
  if(waterLevel > highWaterThreshold){
    digitalWrite(green2, HIGH);
    digitalWrite(yellow1, LOW);
    digitalWrite(red1, LOW);
  } else if(waterLevel < lowWaterThreshold){
    digitalWrite(red2, HIGH);
    digitalWrite(green1, LOW);
    digitalWrite(yellow1, LOW);
    tone(buzzer, 10, 100);
  }
  else{
    digitalWrite(yellow2, HIGH);
    digitalWrite(green1, LOW);
    digitalWrite(red1, LOW);
  }

//check light
  if(light < lightThreshold){
    digitalWrite(white1, HIGH);
    digitalWrite(white2, HIGH);
    digitalWrite(white3, HIGH);
  }
  else{
    digitalWrite(white1, LOW);
    digitalWrite(white2, LOW);
    digitalWrite(white3, LOW);
  }

//print sensor findings on LCD
  lcd.setCursor(0,0);
  lcd.print("ML:" + moistureLevel + "   WL:" + waterLevel)
  lcd.println("LL: " + lightLevel)
}

void updateSensors() {
// updates the value from each sensor
  moistureLevel = analogRead(moistureSensor);
  light = analogRead(lightSensor);
  waterLevel = analogRead(waterLevelSensor);
}
```