The Automatic Plant Watering Device

By Sara Mirza

Description of the product:

I have around 40 plants in my room. I'd like to say that I take very good care of them, but there are times where I may overlook watering a plant or two for more than intended, especially those that are out of my reach. I bought a moisture sensor to check the water content of my plant's soil, but I still have to actively go around my plants and check. I decided to use this project as a push to create Something that would assist me. I designed a device that would always be measuring the moisture level of the soil and warn me when a plant's soil is too dry. Furthermore, in the case that I am occupied and I do take more than a couple of days to water the plant, then the device would water it for me!! The only unfortunate part about this is that I needed to design a container to hold water and an accurate and durable stopper to open and close it. This was really hard to do at home without having any leakage. So I decided to just control the actuator for now and hopefully in the future I can have access to a 3D printer.

Figure 1: Most of the plants in my room. The ones I tend to leave dry for longer than intended are the ones on the top of the shelf. I have more plants on the windowsill shown in Appendix B.

For a video example of the device's operation and testing, visit

Electromechanical details:

<u>Interfacing with the device</u>: almost all of the components are neatly tucked into the control box which is the main body of the device. I used an LCD screen to show the value of the soil's current water percentage, as well as 2 LEDs to warn me when the moisture level dropped below the desired level.

The system uses a capacitive soil moisture sensor. The sensor outputs a voltage value which is converted to a water percentage.

I used a servo motor as I only need a 180-degree revolution in order to open a valve and let water out and the pole count also does not really make a difference to my use. The servo motor is the only component used that requires a 5V input, and that is what the power supply is for.

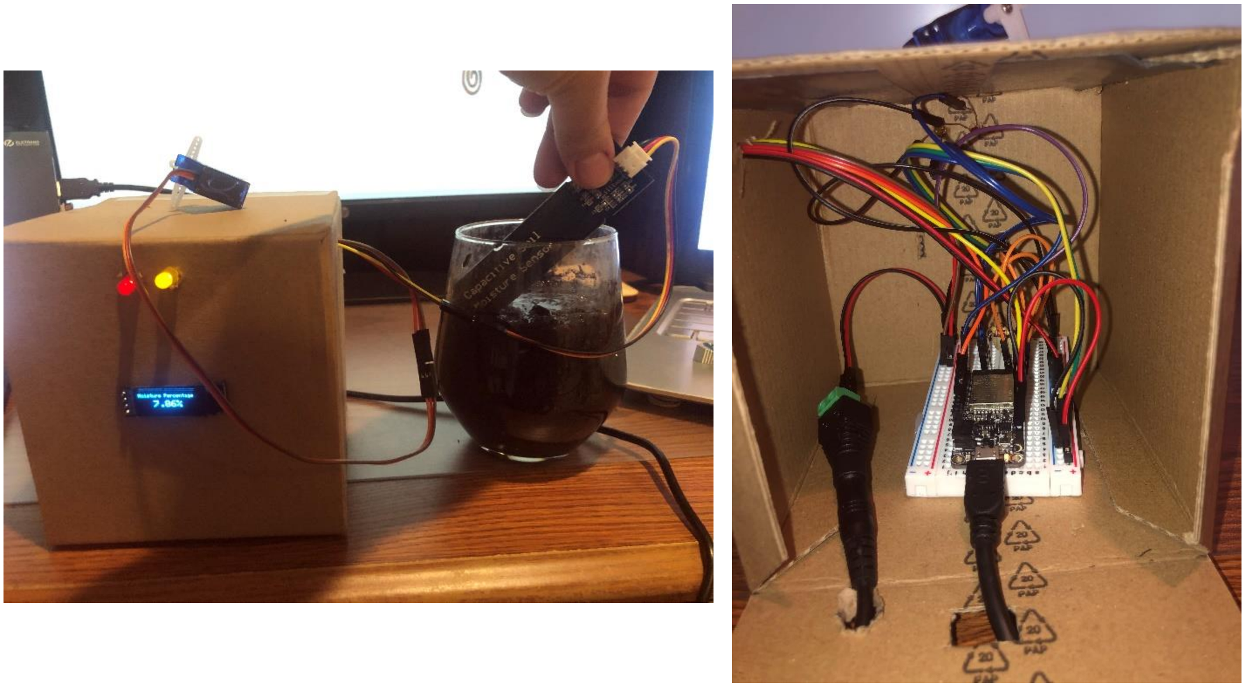


Figure 2: LEDs and an LCD screen are attached at the front of the box in the user's sight, the moisture sensor and servo motor's wires go through a hole on the side of the box, and all the cables, and resistors, as well as the microcontroller sit inside. There are two holes at the back for the USB plug and the power supply plus. The moisture sensor is inserted in the soil, and the motor would have been attached to the water container system and either be on a stand, by the plant or hung up on the wall. A cup filled with soil was used for testing and the demo for the purpose of not overwatering the plant.

Circuit:

In my system, including the calibration portion, there were 4 main components: 1) the servo motor, 2) the OLED screen, 3) the capacitive soil moisture sensor and 4) the load cell and the hx711 amplifier. The challenge in this project for me was that with the exception of the servo motor, all the other main components were new to me, and I had to find out which libraries I need to install and what commands I needed to use to properly get the result I need. Having all the components work together at the same time also had its complexity.

1) Servo motor (SG90 9g Micro Servo, 4 for ~$9). The motor operates at voltages between 4.8-6V. It comes with an encoder that I used to send signals to the motor. I only move it from 0 to 180 degrees and back.
2) OLED screen: it is a small screen with a resolution of 128x32 screen that can operate at DC voltages between 3.3-5V but I decided to use my microcontroller to power it. I bought a bigger one as I thought it would fit more nicely and also be easier to read out from but I needed to solder connections to it and I don't have such equipment so I continued to use the smaller one.
3) Capacitive soil moisture sensor: this kind of sensor works by measuring the changes in the capacitance cause by the changes in the dielectric. Instead of measuring moisture, it measures the ions that are dissolved in the moisture, and thus it produces a voltage output. This output is then taken and is converted to a water percentage using an equation I obtained during the calibration process which I will discuss.
4) The load cell and hx711 amplifier: There are 4 strain gages that are attached to the load cell that form a Wheatstone bridge. As force is applied to the load cell, it temporarily deforms, causing some of the strain gages to compress and others to stretch, this changes the resistance of the

circuit. The output of the load cell is the potential difference at the center of the Wheatstone bridge, which changes as a result to the resistance changing.

In addition to these components, I added two LEDs and current limiting resistors that I used to give me a visual signal when the water percentage dropped too low.

When the USB cable is unplugged, I can use the 5V power supply to power it as the ESP32 has an on-board regulator that converts the voltage to 3.3V.

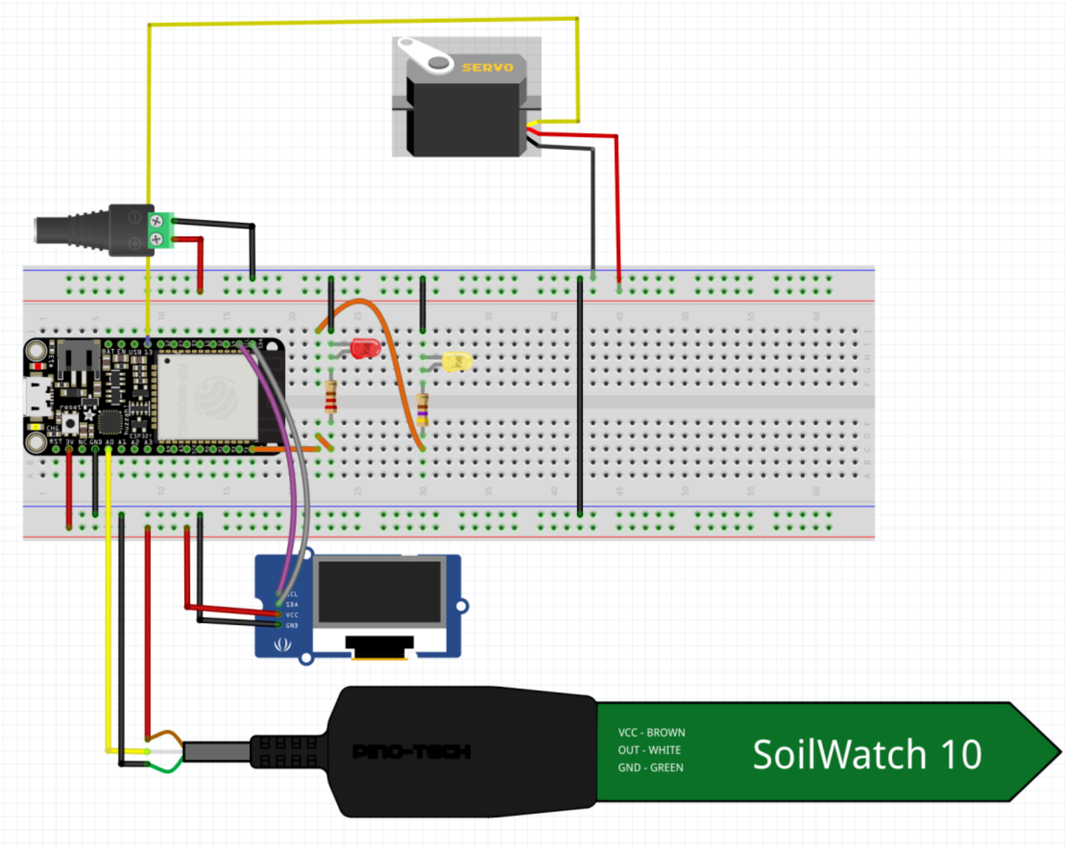
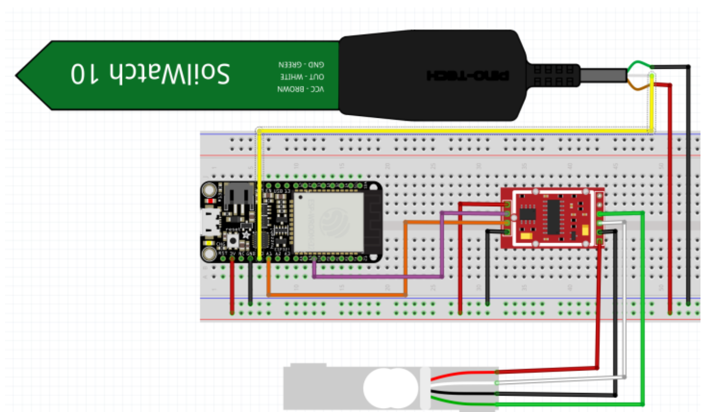
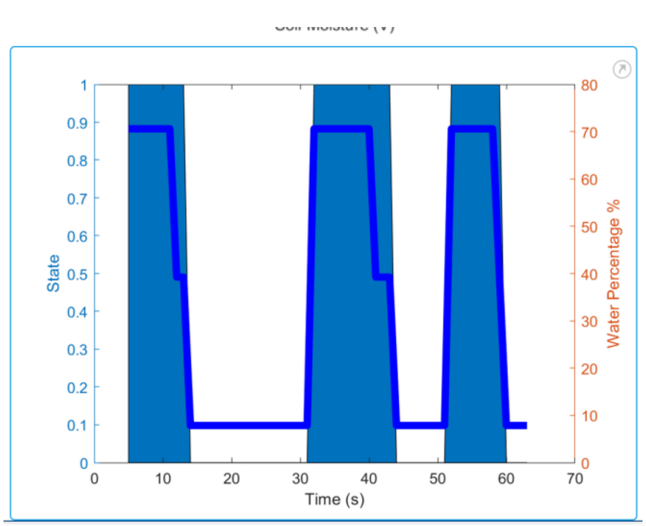


Figure 3: The circuit diagram for this machine

Calibration of moisture sensor:

I used a kitchen scale to obtain the mass of an object that I used to calibrate the load cell.


SoilWatch 10


State
Water Percentage %
Time (s)
0
10
20
30
40
50
60
70
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
80

Project_Script

```
/* ME 102 Project Script
   By Sara Hassan
   12/8/2020
*/

#include <ESP32Servo.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// CONSTANTS / VARS %%%%%%%%%%%%%%%%%%%%%%

//INPUTS
#define SOIL_PIN A0 //AOUT pin on sensor

//OUTPUTS
Servo myservo;  // create servo object to control a servo
          // 16 servo objects can be created on the ESP32
#define SERVO_PIN 13
#define LEDS 21

float sensor_val;
float water_per = 0;
const float per_ref = 32.47;
boolean state = 1;

long initial_time;
long current_time;
long time_lapse;
const long waiting_period = 10;//172800 // in seconds (2 days)

//Displaying Intervals
long D_I_T;
long D_C_T;
float D_T_L;
const float D_W_P = 1000; //2.5 seconds

// OLED SCREEN
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
#define NUMFLAKES     10 // Number of snowflakes in the animation example
#define LOGO_HEIGHT   16
#define LOGO_WIDTH    16
static const unsigned char PROGMEM logo_bmp[] =
```

```
static const unsigned char PROGMEM logo_bmp[] =
{ B00000000, B11000000,
  B00000001, B11000000,
  B00000001, B11000000,
  B00000011, B11100000,
  B11110011, B11100000,
  B11111110, B11111000,
  B01111110, B11111111,
  B00110011, B10011111,
  B00011111, B11111100,
  B00001101, B01110000,
  B00011011, B10100000,
  B00111111, B11100000,
  B00111111, B11110000,
  B01111100, B11110000,
  B01110000, B01110000,
  B00000000, B00110000 };

//INITIALIZE TIMER
hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
int timer_counter=0;
int old_timer_counter=0;

void IRAM_ATTR isrTimer(){
  printing();
  //testdrawstyles();
  portENTER_CRITICAL_ISR(&timerMux);

  //Serial.println(timer_counter);
  timer_counter++;
  portEXIT_CRITICAL_ISR(&timerMux);
  //Serial.println(long(millis()));
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); // serial port setup

  // SERVO MOTOR SET UP %%%%%%%%
  // Allow allocation of all timers
  ESP32PWM::allocateTimer(0);
  ESP32PWM::allocateTimer(1);
  ESP32PWM::allocateTimer(2);
  ESP32PWM::allocateTimer(3);
  myservo.setPeriodHertz(50);    // standard 50 hz servo
  myservo.attach(SERVO_PIN, 500, 2400); // attaches the servo on pin 13 to the servo object
        // different servos may require different min/max settings
```

```
      // for an accurate 0 to 180 sweep

  //OLED SCREEN %%%%%%
  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C for 128x32
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }
  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();
  delay(2000); // Pause for 2 seconds
  // Clear the buffer
  display.clearDisplay();
  // Draw a single pixel in white
  display.drawPixel(10, 10, SSD1306_WHITE);
  // Show the display buffer on the screen. You MUST call display() after
  // drawing commands to make them visible on screen!
  display.display();
  delay(2000);

  //TIMER %%%%%%%%%%%
  timer = timerBegin(0,80,true); //Timer triggering 80 million times a second so when we set 80 we are saying one second is 1000000
  timerAttachInterrupt(timer,&isrTimer, true);
  timerAlarmWrite(timer, 1000000,true);//Runs one second at a time
  timerAlarmEnable(timer);

  pinMode(LEDS, OUTPUT);

}

void loop() {
  current_time = millis();
  time_lapse = ((current_time - initial_time)/1000);
  sensor_val = (analogRead(SOIL_PIN)/1023); //read sensor
  water_per = -31.3774*sensor_val + 101.9908;
  if(water_per < per_ref && state==1){
    state = !state;
    alert();
    initial_time = millis();
    //Serial.println("111111111111111111111111111111111");
  }
  else if(water_per < per_ref && state==0 && time_lapse < waiting_period){
    alert();
    //Serial.println("22222222222222222222222222222222");
  }
  else if(water_per < per_ref && state==0 && time_lapse >= waiting_period){
    alert();
```

```
    state=1;
    myservo.write(90);
    delay(2000);
    myservo.write(0);
    initial_time = millis();
    //Serial.println("33333333333333333333333333333333");
  }
  if(water_per > 32 && state==0){
    state = !state;
    initial_time = millis();
    //Serial.println("44444444444444444444444444444444");
  }

D_C_T = millis();
D_T_L = float(D_C_T - D_I_T);
if (D_T_L > D_W_P){
  testdrawstyles();
  D_I_T = millis();
}


}
void alert(){
  digitalWrite(LEDS, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LEDS, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}

//Printing for trouble shooting (every 4s)
//void printing(){
//  Serial.print("Soil Moisture Sensor Voltage: ");
//  Serial.print(sensor_val); // read sensor
//  Serial.println(" V");
//  Serial.print("Water Percentage: ");
//  Serial.println(water_per);
//  Serial.print("state: ");
//  Serial.println(state);
//  Serial.print("time_lapse: ");
//  Serial.println(time_lapse);
//  //delay(100); // slight delay between readings
//}

//printing for plotting (every 0.5s)
void printing(){
  Serial.print(millis()/1000);
  Serial.print(",");
  Serial.print(water_per);
```

```

}
void alert(){
  digitalWrite(LEDS, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LEDS, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}

//Printing for trouble shooting (every 4s)
//void printing(){
//  Serial.print("Soil Moisture Sensor Voltage: ");
//  Serial.print(sensor_val); // read sensor
//  Serial.println(" V");
//  Serial.print("Water Percentage: ");
//  Serial.println(water_per);
//  Serial.print("state: ");
//  Serial.println(state);
//  Serial.print("time_lapse: ");
//  Serial.println(time_lapse);
//  //delay(100); // slight delay between readings
//}

//printing for plotting (every 0.5s)
void printing(){
  Serial.print(millis()/1000);
  Serial.print(",");
  Serial.print(water_per);
  Serial.print(",");
  Serial.print(state);
  Serial.println(";");
  //delay(100); // slight delay between readings
}

void testdrawstyles() {
  display.clearDisplay();
  display.setTextSize(1);             // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE);        // Draw white text
  display.setCursor(0,0);             // Start at top-left corner
  display.println(F("Moisture Percentage"));
  display.setTextSize(2);             // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(30,15);
  display.print(water_per);
  display.println(F("%"));
  display.display();
}

```