Capability Brown: Smart Irrigation System

James Martin

Introduction:

With the rise of urbanization and the need for sustainable living, the demand for efficient and environmentally friendly gardening solutions has increased in recent years. In turn, many gardeners have started to use smart gardening systems to make their gardens more productive and manageable. Because the prices on these systems can range from $400 to $1300, I've decided to make one of my own. In this project, you will see just how easy it can be to monitor the moisture levels, temperature, humidity, and lighting of plants remotely using a NodeMCU and Adafruit IO. Adafruit IO is an easy to use Internet of Things (IoT) interface that allows you to monitor this project remotely and has extremely good documentation about its use via the Adafruit website.

Parts List:

Software:
Arduino IDE
Adafruit IO

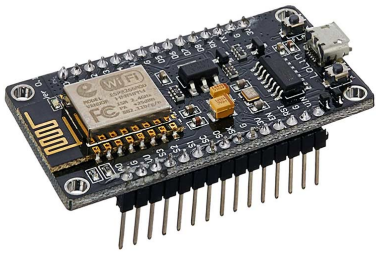
Hardware:
NodeMCU 1.0 (ESP-12E)
ADS1115 Analog to Digital Converter Breakout Board
Capacitive Soil Moisture Sensor
LDR (Light Dependent Resistor, 50Ω)
BSS138 Logic Level Converter Breakout Board
(2) 5V Relay
5V UV lamp
5V Water Pump
Plastic Tubing
DHT11 Temperature and Humidity Sensor
12V DC Brushless Fan

Hardware (con't):
(3) 18650 battery with Holder
Water Container
Plant with Pot / Soil
(2) LCD screens (optional)

Small Parts:
(2) 4.7kΩ Resistor
10kΩ Resistor
(2) 1nF Capacitor
10uF Capacitor
100uF Capacitor
Veroboard (Stripboard)
Standoffs
Hot Glue
Jumper Wires (DuPont and Breadboard)
LM7805 Voltage Regulator with Heatsink

Hardware and Theory:

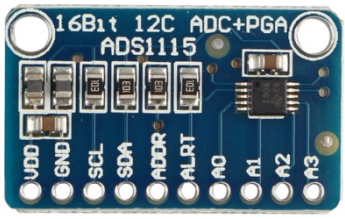
NodeMCU (WiFi)



The NodeMCU is a cheap and easy to use development board. It uses the ESP8266 WiFi chip and will serve as the main networking feature of this project. Connecting to any WiFi network is extremely simple and will be done in the program automatically. The NodeMCU has 11 digital pins and 1 analog pin with an operating voltage of 3.3V. This project uses pins D1 and D2 for I2C communication with the analog to digital convertor and pins D4, D5, and D8 for other sensors. The power to the circuit also comes from the Node. It uses the 3.3V pin for 3.3V rails and the VIN pin for 5V rails. To set up and install the NodeMCU for use with Arduino IDE, please go to:
https://learn.adafruit.com/adafruit-io-basics-esp8266-arduino/using-arduino-ide?gclid=CjwKCAjwl6OiBhA2EiwAuUwWZVUOMzOE8dk3LLDtBcGaKA4SKSFyCl8pEGwtS6EwrJgCVz1UTcA7IhoC1dsQAvD_BwE

It works exactly like the Adafruit Feather HUZZAH ESP8266. We will use Arduino's C based language instead of the NodeMCU's native Lua language.

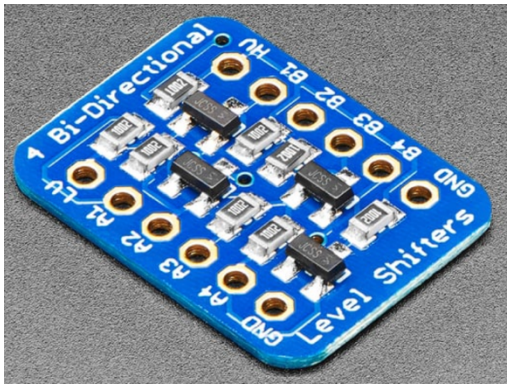
ADS1115 ADC (I2C)



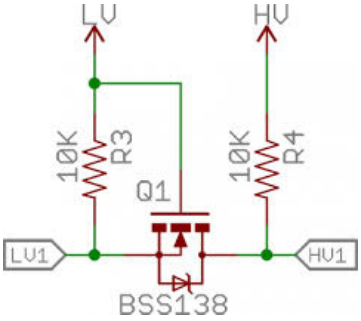
Since the NodeMCU only has one analog pin and we will be using two analog sensors in this project, we will be utilizing pins D1 and D2 of the NodeMCU instead. These are the SDA (data line) and SCL (clock line) pins for I2C. I2C is a versatile two wire communication protocol that is very common when interfacing multiple sensors with a microcontroller. The ADC takes data from an analog sensor and converts it into binary (1s and 0s) for use by a digital pin. The soil moisture sensor and light sensor will be connected to the ADC.

There are two 4.7KΩ resistors on the SDA and SCL lines to tie them high and a 1nF capacitor on the DC power lines for AC action and noise rejection.

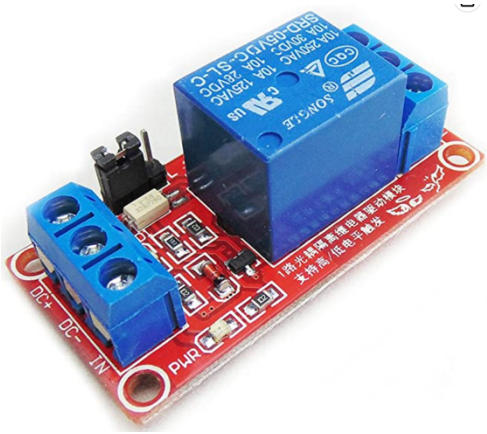
BSS138 Logic Level Converter



The logic levels from the NodeMCU are 3.3V. This is a problem because the relays we are using need 5V logic levels. The breakout board features four channels of bi-directional I2C safe BSS138 logic level converters to shift the 3.3V levels to 5V levels. These circuits use an n-channel MOSFET as a switch.



When the low logic side is low, the MOSFETs gate has no voltage because no voltage is coming from the low voltage and is off. When the low logic side is high, voltage will be allowed into the gate, turning the MOSFET on, and allowing voltage to travel through the high voltage and into the device that needs a 5V logic level. A single channel example is pictured to the left.

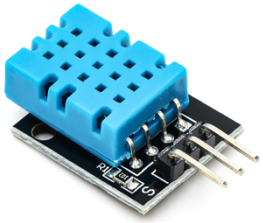


5V Relay

The two 5V relays will be used as switches to turn on and off the UV lamp and water pump. Electrical relays are switches that open and close based on a high or low signal. For this project, the relays will switch on the lamp or pump whenever they receive a high signal from the NodeMCU.

Relays also allow us to connect devices to voltages higher than allowed by the NodeMCU via an external power supply. Currently, this project uses the NodeMCU VIN pin as a 5V power supply.

DHT11 / UV Lamp / Water Pump / LDR



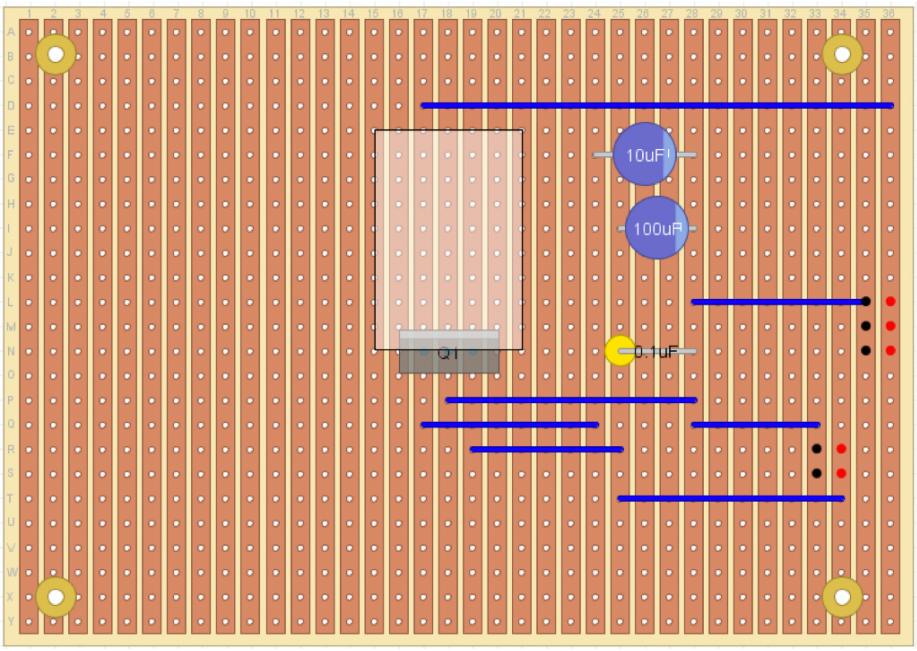
The DHT11 temperature and humidity sensor is a basic three pin DHT11 breakout board. It has a VCC, GND, and DATA pin. The data pin connects to the D5 pin of the NodeMCU.

The UV lamp and water pump can by any 5V lamp and pump.

The LDR is a variable resistor that changes resistance based on light. It's connected to a 10kΩ resistor as a voltage divider that outputs to the ADC.

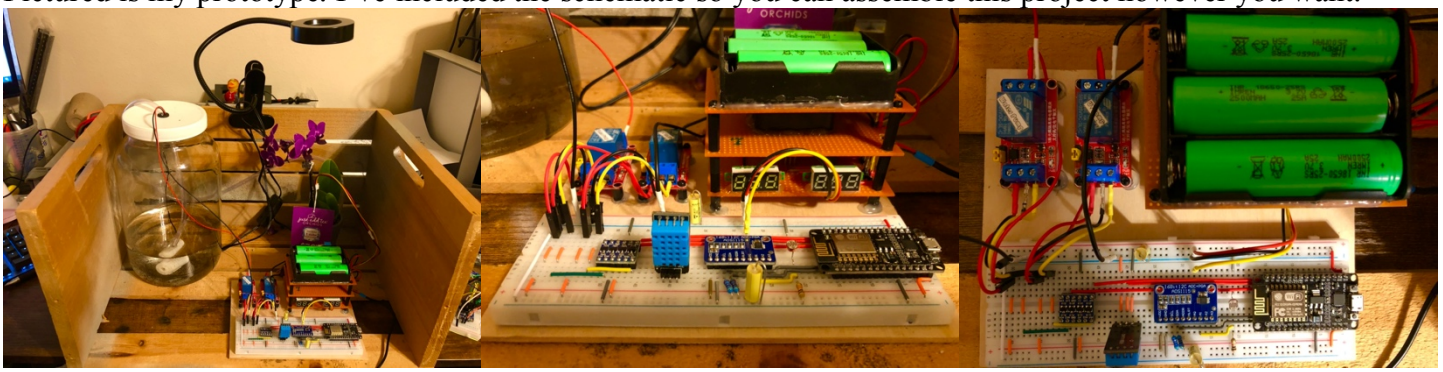
Power

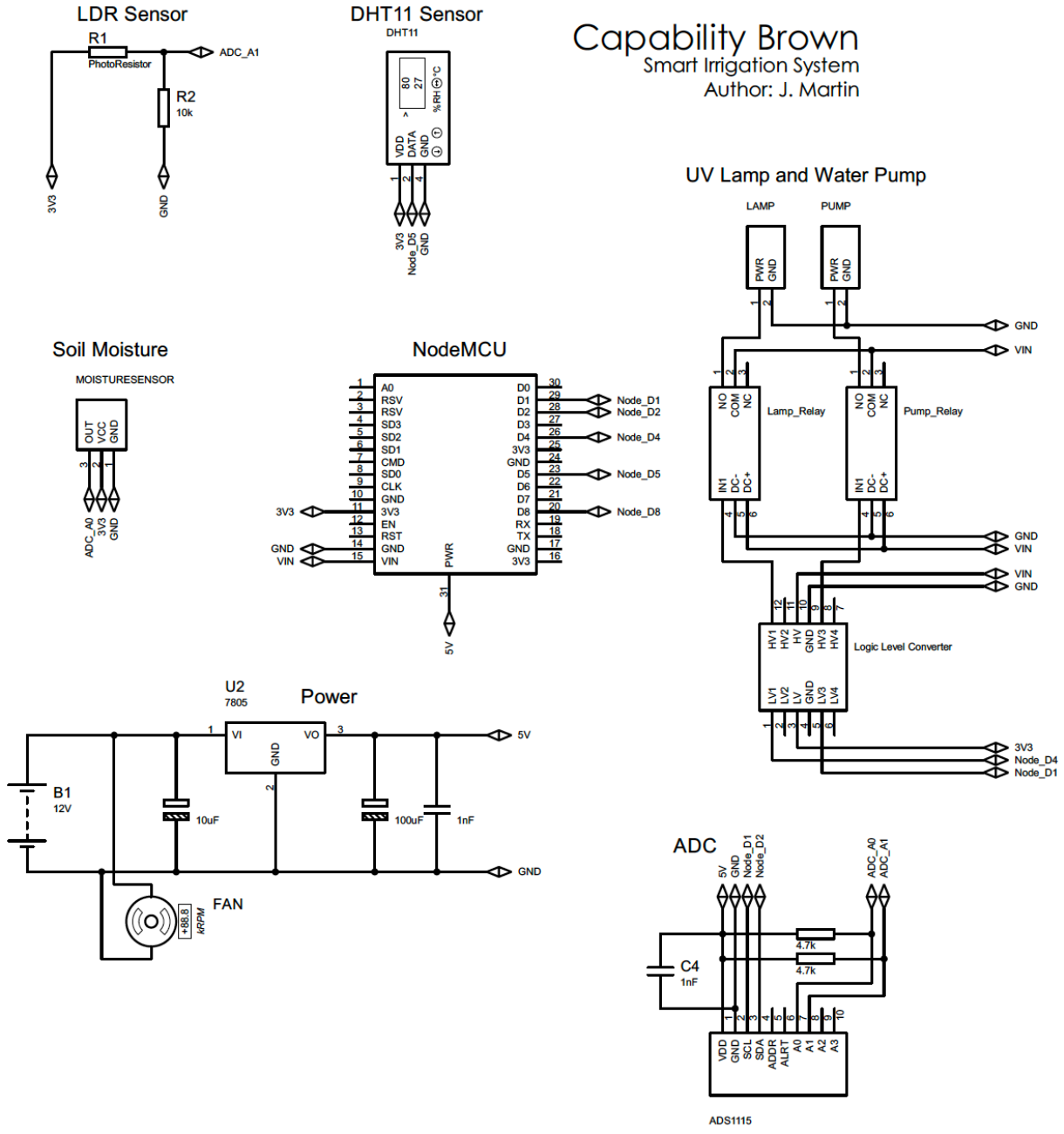
Connecting the NodeMCU to a computer or 5V wall adapter will power this project just fine. However, to add mobility, we will create a 5V power supply with three 18650 batteries, an LM7805 voltage regulator, a fan, and some capacitors. 12V will flow into the circuit, go through the 7805, and come out as 5V. The extra voltage is dissipated via heat. This is why we MUST attach it to a heatsink and position a fan next to it. The capacitors are there to filter any transient voltages during the conversion. The 7805 circuit for this project was built on veroboard and the layout is to the left. Two LCDs are included in the final setup to monitor input and output voltage. A fan is then stacked on top of it with some standoffs. It's just a large chunk of veroboard with a hole cut in it for air to flow. Next, the 18650 battery pack is stacked on top of the fan and power and ground are connected via DuPont wires.



Construction

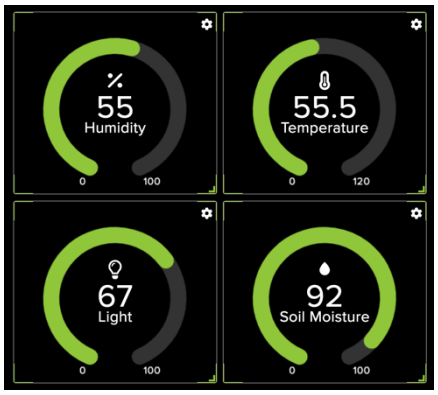
Pictured is my prototype. I've included the schematic so you can assemble this project however you want.





Programming

Follow the link to make an Adafruit IO account and view the dashboard and feed tutorials:
https://learn.adafruit.com/welcome-to-adafruit-io/what-is-adafruit-io



Open up the Arduino IDE and enter the program as follows:

Begin by including your dependent libraries. To install AdafruitIO_WiFi.h, ADS1X15.h, and DHT.h you will have to search for them under ***Sketch >> Include Library >> Manage Libraries***.

```
#include <ESP8266wifi.h>
#include "AdafruitIO_WiFi.h"
#include "ADS1X15.h"
#include <DHT.h>
```

Now, give the sensors reasonable names and define their pin numbers. Keep in mind, the moisture sensor and LDR are connected via the ADC. You'll be calling for them later.

```
// Define ADC addr.
ADS1115 ADS( 0x48 );

// Define DHT11 temperature / humidity sensor
#define DHTPin  D5
#define DHTType DHT11
DHT dht(DHTPin, DHTType);

// Define UV-Lamp
#define lamp D7

// Define water pump
#define pump D4
```

Next, enter in the information that the NodeMCU needs to be able to connect to the internet via WiFi and upload things to your Adafruit IO dashboard. You'll have to enter your own SSID and password. Your SSID is your WiFi routers name. You can find your Adafruit IO keys by clicking the yellow keys in the top right corner of the screen in your Adafruit IO account.

```
// Set user WiFi login information
#define WIFI_SSID       "[redacted]"
#define WIFI_PASS       "[redacted]"

// Set user Adafruit IO login information
#define IO_USERNAME     "[redacted]"
#define IO_KEY          "[redacted]"

// Connect to the Adafruit IO server
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);

// Create feed objects
AdafruitIO_Feed *temperatureFeed = io.feed("Temperature");
AdafruitIO_Feed *humidityFeed = io.feed("Humidity");
AdafruitIO_Feed *lightFeed = io.feed("Light");
AdafruitIO_Feed *moistureFeed = io.feed("Moisture");
```

In the setup section, start by beginning serial output at a baud rate of 115200. Since the UV lamp and water pump pins will be used to send a logic level to the relays, make them outputs using the pinMode() function and make sure they start in an off state by making those pins low with the digitalWrite() function. Next, connect to Adafruit IO, and begin the ADC and DHT11. The while loop is used to print dots while its connecting so you know it didn't crash.

```
void setup() {

  // Begin serial output at 115200 baud rate
  Serial.begin(115200);

  // Set lamp and pump pins as outputs
  pinMode( lamp, OUTPUT );
  pinMode( pump, OUTPUT );

  // Set lamp and pump intial states to off
  digitalWrite( lamp, LOW );
  digitalWrite( pump, LOW );

  // Connect to Adafruit IO
  io.connect();

  // Start ADC and set gain
  ADS.begin( );
  ADS.setGain( 0 );

  // Start DHT11 sensor
  dht.begin();

  // Wait for connection
  while(io.status() < AIO_CONNECTED)
  {
    Serial.print(".");
    delay(500);
  }
}
```

In loop(), start your dashboard with the io.run() function and get temperature and humidity readings from the DHT11 sensor. It reads temperature in Celsius so you need to convert it to Fahrenheit with the following formula: $\left(\frac{C*9}{5}\right) + 32$

```
void loop() {
  io.run( );

  // Read temp / humidity from sensors
  float temperature = ( dht.readTemperature( ) * ( 9 / 5 ) ) + 32;
  Serial.print("Temperature: ");
  Serial.println( temperature );
  float humidity = dht.readHumidity( );
  Serial.print("Humidity: ");
  Serial.println( humidity );
```

Reading the ADC sensor values is more involved. First, we need to establish a voltage factor of 2. Next, we take the analog reading from the ADC pin with ADS.readADC(). Remember the LDR is ADC pin A1 and the moisture sensor is ADC pin A0. Now, we convert the analog data into a percentage using the map() function. For light, 3 is 100% light. For the moisture sensor, 13.8 is 100% wet. The 0 and 100 are used to map the analog data from 0 to 100 percent.

```
  // Read light / moisture from ADC
  float voltFactor = ADS.toVoltage( 2 );
  int lightRead = ADS.readADC( 1 );
  float light = map( ( lightRead * voltFactor ), 0, 3, 0, 100 );
  Serial.print("Light: ");
  Serial.println( light );
  int moistRead = ADS.readADC( 0 );
  float moisture = map( ( moistRead * voltFactor ), 13.8, 0, 0, 100 );
  Serial.print("Moisture: ");
  Serial.println( moisture );
```

Next, program the UV lamp and water pump actions. If the light level gets below 35%, the UV light will turn on. If the soil moisture level gets below 50%, the pump will turn on for 250mS.

```
  // Lamp actions
  if ( light <= 35 ) {
    digitalWrite( lamp, HIGH );
  }
  else {
    digitalWrite( lamp, LOW );
  }

  // Pump actions
  if ( moisture <= 50 ) {
    digitalWrite( pump, HIGH );
    delay( 250 );
    digitalWrite( pump, LOW );
  }
  else {
    digitalWrite( pump, LOW );
  }
```

Finally, upload your variables to your Adafruit IO feeds so your gauges have data to display. I've found that a delay of under 200mS results in a NaN (not a number) on the dashboard. The final delay is to unsure you don't get a throttle warning from Adafruit IO. If you use the free version you can only upload 30 data points per minute.

```
  // Save variables to their feeds on Adafruit IO
  temperatureFeed -> save( temperature );
  delay( 200 );
  humidityFeed -> save( humidity );
  delay( 200 );
  lightFeed -> save( light );
  delay( 200 );
  moistureFeed -> save( moisture );

  // Delay for Adafruit IO uploads per minute ( 30/min )
  delay( 8000 );
}
```

Now, upload the program to your NodeMCU using the instructions in the link at the start of the programming section.

Conclusion

This project is the third attempt at making a WiFi smart garden. First, I was using a MKR WiFi 1010 and MKR IoT Carrier with Arduino IO. It was a convenient setup because all the relays and sensors were embedded in the IoT Carrier. The problem was the connectivity. Arduino IO doesn't have a limit on the amount of data you can upload to it causing a problem when a lot of people are using its servers. The dashboard updates took 7-10 minutes sometimes. The second attempt was using an Arduino UNO with a simple ESP8266 module and Adafruit IO. Even with Adafruit IO's upload restrictions this setup produced similar results and my setup became a mess of jumper wires rather quickly. Keeping wiring clean was a crucial thing for me because I wanted to be able to work with the setup and not accidentally rip any wired connections out. Using the NodeMCU is really a space and time saver. It was made specifically for these types of projects and, even though it doesn't have the convenient embedded relays and sensors, it was much more intuitive for me to setup and manipulate. Having many input / output pins also allows for a wider range of sensors and lines of communication (like I2C and UART). Think of it as a "ESP8266 WiFi microcontroller".

The power supply as it stands is an issue. The 7805 regulator gets uncomfortably hot. There are two fixes I'd like to try in the future. First, using a smaller battery pack. Since the regulator uses heat dissipation to step down the voltage, a smaller battery pack would lessen the amount of voltage it needs to dissipate and it wouldn't get so hot. Second, I could use a larger fan with better air flow. The current fan is a rather small fan to save space but it doesn't move enough air around the regulator to be truly effective. This is fine for now as we can use a 5V/1A wall adaptor or just plug it into a computers USB port. The NodeMCU has a 1A onboard limiter.

Useful Links

Learn about Adafruit IO:
https://learn.adafruit.com/welcome-to-adafruit-io/what-is-adafruit-io

Learn about the NodeMCU:
https://en.wikipedia.org/wiki/NodeMCU

Installing Arduino IDE:
https://docs.arduino.cc/software/ide-v1/tutorials/Windows

Using NodeMCU with Adafruit IO:
https://learn.adafruit.com/adafruit-io-basics-esp8266-arduino/arduino-io-library

Using the DHT11 temperature / humidity sensor with NodeMCU and Adafruit IO:
https://electricdiylab.com/esp8266-nodemcu-and-dht11-sensor/

Programming an LDR ambient light sensor:
https://maker.pro/arduino/tutorial/how-to-use-an-ldr-sensor-with-arduino

Soil moisture sensor calibration:
https://makersportal.com/blog/2020/5/26/capacitive-soil-moisture-calibration-with-arduino

Programming a soil moisture sensor:
https://lastminuteengineers.com/soil-moisture-sensor-arduino-tutorial/

Using relays with Arduino IDE:
https://littlebirdelectronics.com.au/guides/56/use-an-arduino-to-control-a-relay

The original idea for this project:
https://docs.arduino.cc/tutorials/mkr-iot-carrier/smart-garden-project