Arduino Distance Indicator

Online Link:

This project is available on my online portfolio at: http://timothyjgow.com/

Objective:

The purpose of this project is to learn how to use an ultrasonic sensor to measure distance and using http send data to an actuator to do something based off the sensor's readings. This includes the following:

- Set up a distance sensor to work with Arduino
- Set up LED lights to work with Arduino
- Creating a http server to send data to client
- Creating a client to receive data from a server

Materials:

The materials needed for this project are the following:

- Personal Computer
- Arduino IDE
- 2x Wemos D1 Mini
- 2x MicroUSB cable
- 2x Breadboard
- HC-SR04 Ultrasonic Sensor
- 1x Red LED (I used 2, optional)
- 1x Yellow LED (I used 2, optional)
- 1x Green LED (I used 2, optional)
- 9x Jumper Wires (I used 12, optional)
- 3x 220 Ω Resistors (I used 6, optional)

References:

Below are the resources I used to build this project:

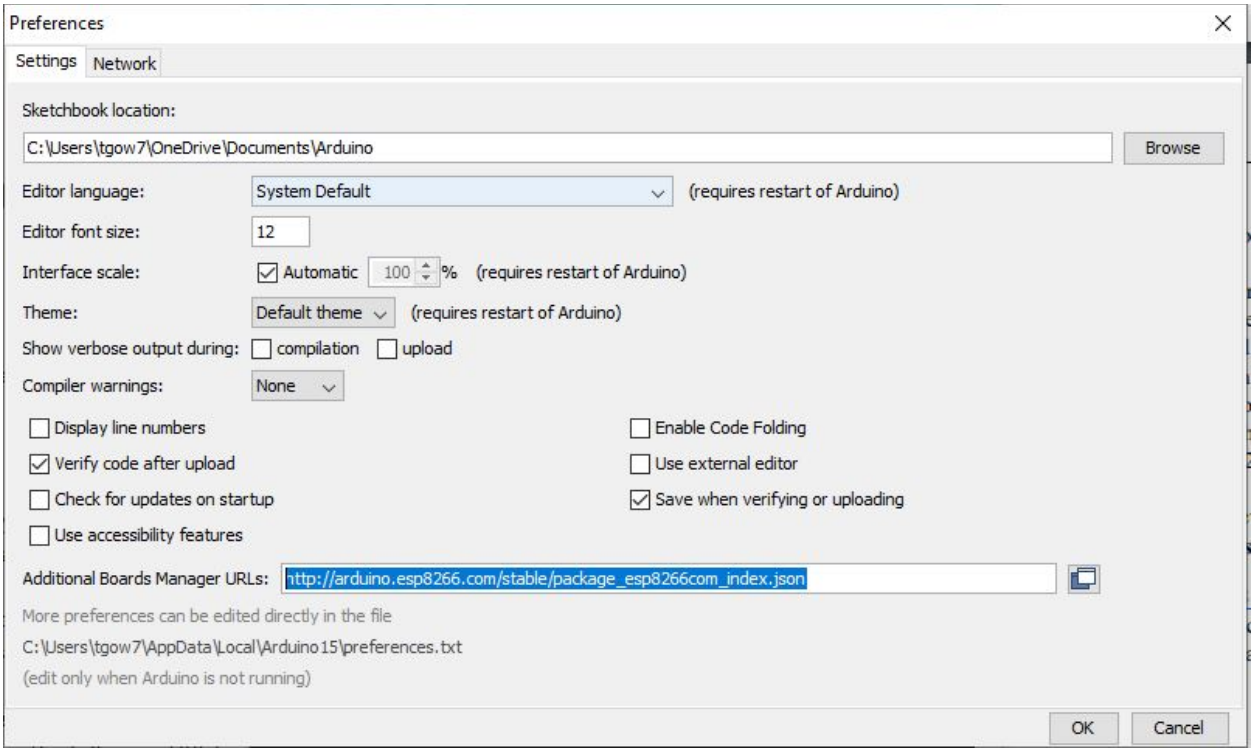
http://timothyjgow.com/ I used the Arduino Traffic Light project to help me configure the http server and the Wi-Fi network.

https://www.pinterest.com/pin/312648399135257896/?nic_v2=1a5fNjweZ I used the image here to know which pins are which.

Procedures:

We will start with the Client board first.

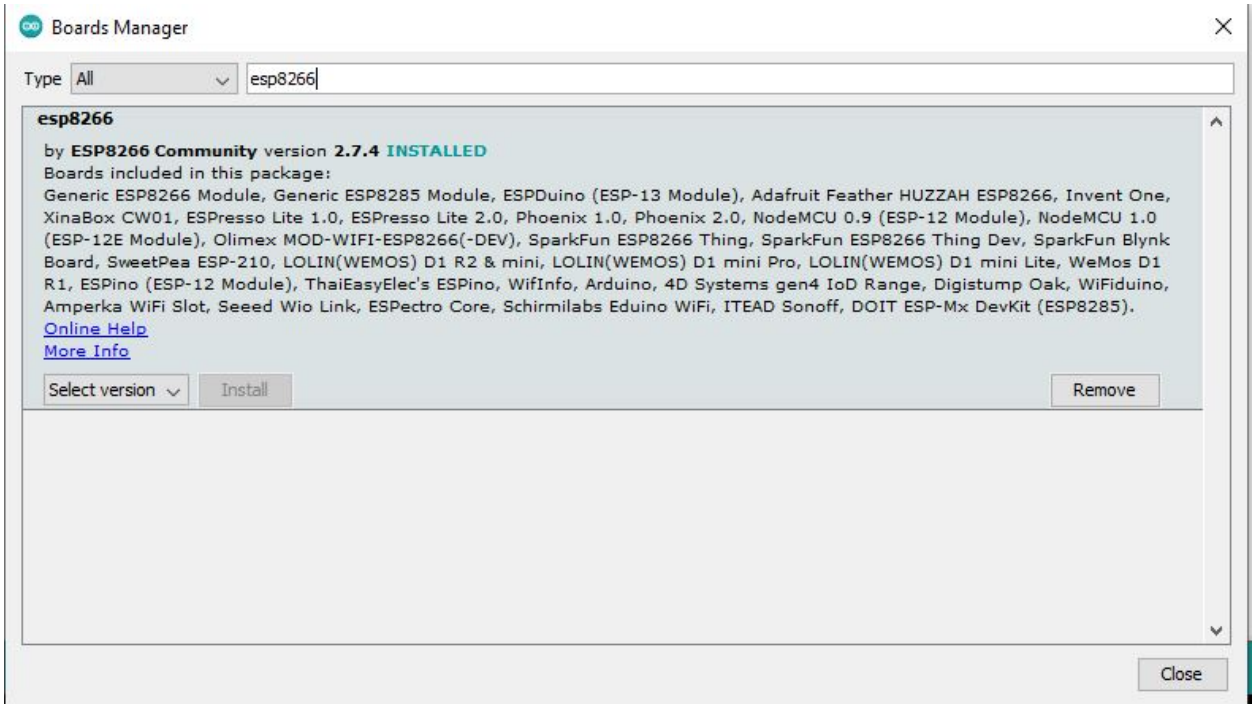
1. Download and install the Arduino IDE from https://www.arduino.cc/en/main/software
2. Install the CH340 USB to Serial controller (if not automatically installed) on the computer with the Arduino IDE.
   a. The Arduino IDE needs to be able to upload and program the D1 mini. The USB to serial chip can be installed in Windows Device Manager.
3. Add the Wemos D1 Library to the Arduino IDE
   a. In the Arduino IDE, Select **File** and then **Preferences.**
   b. In the Additional Boards Manager URLs field, paste in the following URL for the Wemos D1 Mini: http://arduino.esp8266.com/stable/package_esp8266com_index.json



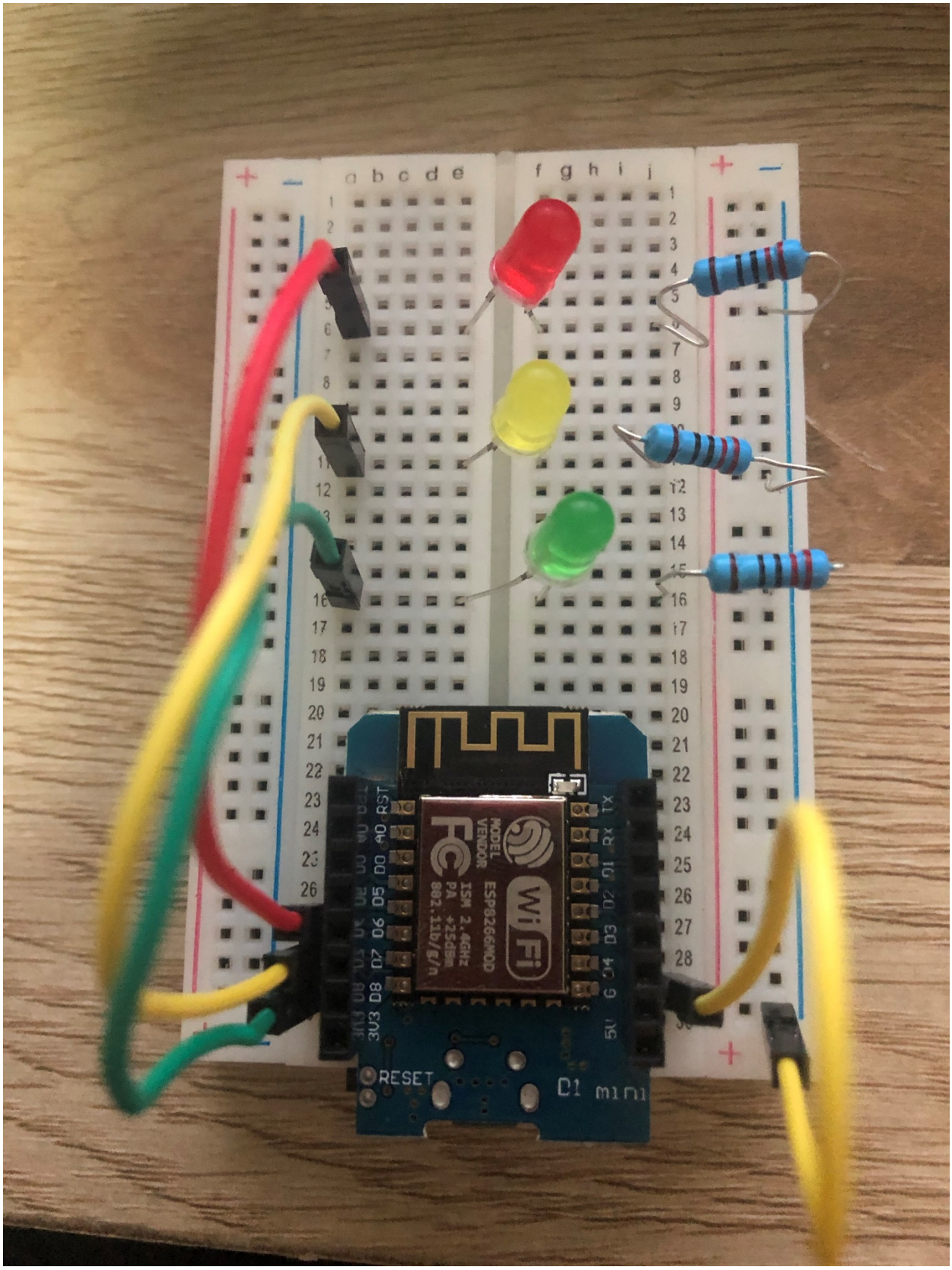
4. Install the ESP8266 Library:
   a. **Go to Sketch** / **Include Library** / **Manage Libraries,** in the search box enter **ESP8266 Platform**

b. Install the Library called **ESP8266 Microgear**

5. Next step is to install the board so that it shows up in the tools menu alongside the other Arduino boards.
   a. Go to **Tools** / **Board** / **Boards Manager**, then search for **ESP8266**. Install the first option.



6. Connect the Wemos board into your pc with the MicroUSB cable.
   a. Select which board to use. This could be the Generic ESP8266 module. **Tools** / **Boards** / **ESP8266 Boards** / **Generic ESP8266 Module**
   b. Next select the correct COM port to communicate to the board. Mine for example was COM3. **Tools** / **Port:**
7. Now it is time to wire the LEDs to the Breadboard so we can program them **Make sure** the USB is disconnected from the Board.
   a. Connect the Wemos board into the breadboard so that the female microUSB port is sticking out at the end edge of the breadboard.
   b. Connect a jumper to the G pin on the 5v side. Connect the other end on the negative far column.
   c. Connect the resistors one after another on the negative column with the other connection in the connecting to the middle.
   d. Connect the Anode ends of each LED to each of their corresponding resistors.
   e. Use jumper wires to connect the Cathode end of each LEDs to pins D8, D7, and D6. Which in my case is Green = D8, Yellow = D7, and Red = D6.


RESET
D1 mini

8. Now we will get started with the code for the client board. Plug in your wemos board to your pc using the microUSB cable.
   a. The first part of the code is to define the LED variables, password and ssid, include wifi library set the IP address we will connect to (note this will most likely be different for you), and declare the client object.

```
#include <ESP8266WiFi.h>

const char* ssid = "your wifi network";           // SSID of your home WiFi
const char* password = "your wifi password";            // password of your home
WiFi
int GreenLED = 15;
int YellowLED = 13;
int RedLED = 12;
int flashing = 0;


IPAddress server(192,168,0,4);       // the fix IP address of the server
WiFiClient client;
```

   b. Next we will make our setup loop which will connect to the wifi network and set the LEDs to output.

```
void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, password);             // connects to the WiFi router
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
 Serial.println("Connected to wifi");
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
  pinMode(GreenLED, OUTPUT);
  pinMode(YellowLED, OUTPUT);
  pinMode(RedLED, OUTPUT);
}
```

   c. The last part of the code is to write the main loop. In this loop we will connect to our server on port 80 we will define a answer for any incoming data the server gives

us. And from there we will do turn on different lights depending on the answer's value.

```
void loop () {
  client.connect(server, 80);   // Connection to the server
  String answer = client.readStringUntil('\r');   // receives the answer from the
 sever
  if (answer == "greenON"){
    digitalWrite(GreenLED, HIGH);
    digitalWrite(YellowLED, LOW);
    digitalWrite(RedLED, LOW);
    flashing = 0;
  }
  else if (answer == "yellowON") {
    digitalWrite(GreenLED, LOW);
    digitalWrite(YellowLED, HIGH);
    digitalWrite(RedLED, LOW);
    flashing = 0;
  }
  else if (answer == "redON") {
    digitalWrite(GreenLED, LOW);
    digitalWrite(YellowLED, LOW);
    digitalWrite(RedLED, HIGH);
    flashing = 0;
  }
  else if (answer == "redFLASH") {
    flashing = 1;
    digitalWrite(GreenLED, LOW);
    digitalWrite(YellowLED, LOW);
    digitalWrite(RedLED, LOW);
  }
  if (flashing == 1) {
    digitalWrite(RedLED, HIGH);
    delay(800);
    digitalWrite(RedLED, LOW);
    delay(300);
    digitalWrite(RedLED, HIGH);
    delay(800);
    digitalWrite(RedLED, LOW);
    delay(300);
    digitalWrite(RedLED, HIGH);
    delay(800);
    digitalWrite(RedLED, LOW);
    delay(300);
```

```
  }
  client.flush();
}
```

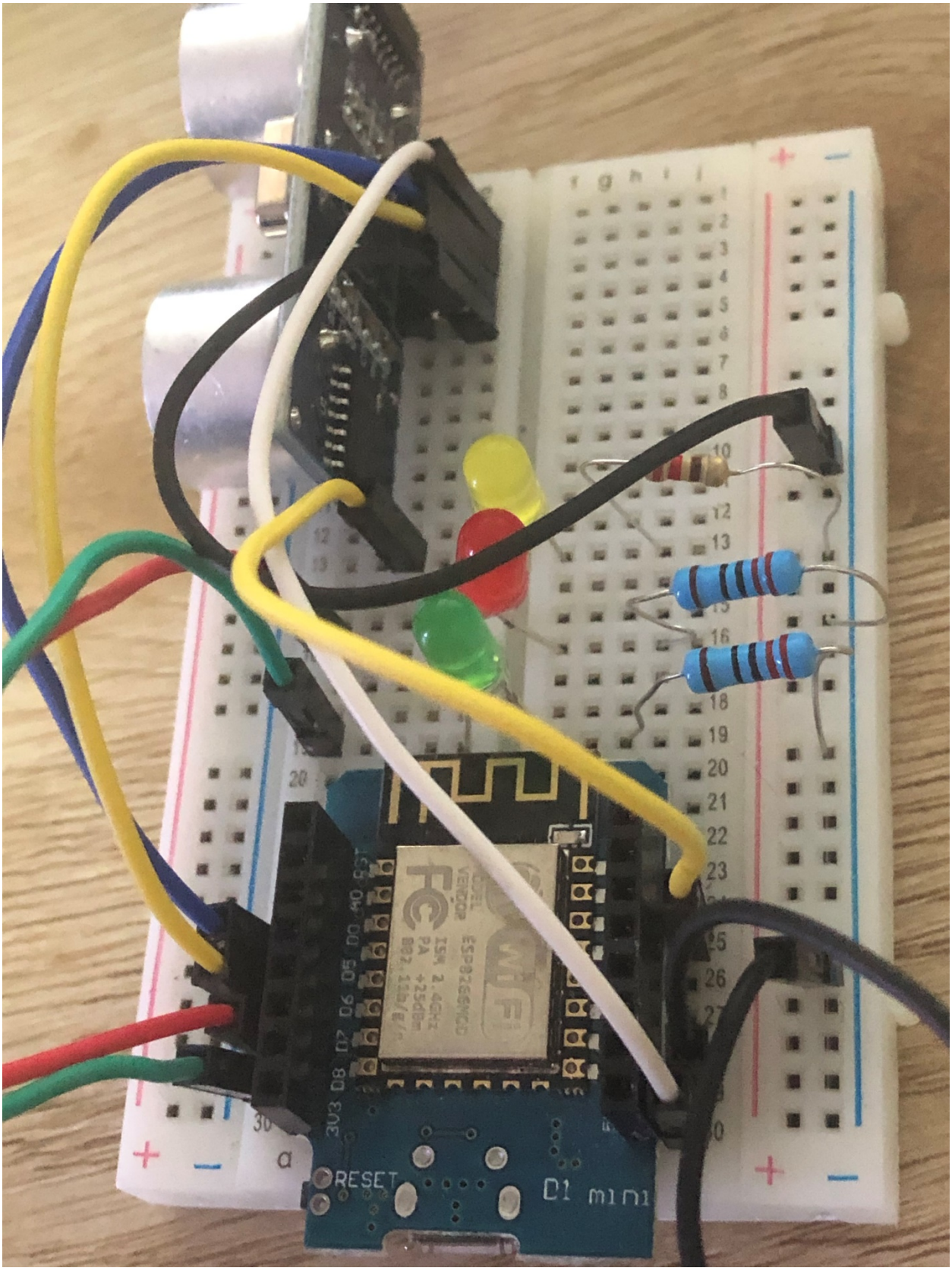
9. The last step for the client side is to upload the code to the board.
   a. In the Arduino IDE click on the arrow pointing to the right. This will compile and upload the code to the Arduino.

10. We will now start working on the client device. This will have the optional items in it. You can choose to add those in if you want. In the code you will just comment out or not include the information for the LED pins.

11. The bare minimum wiring for this is 4 LED jumpers, ultrasonic sensor, and the Wemos board.

    a. Connect the VCC on the Sensor to the 5v pin on the Wemos board.
    b. Connect G on the Sensor to G on the Wemos board. Should be right next to the 5v pin.
    c. Connect Echo to D6 and connect Trig to D5
    d. Optionally you can add in three LEDs. Green, Yellow, Red. They will need three resistors and three jumper wires. Connect Green to D8, Connect Red to D7 and connect Yellow to D1.

    Image for wiring is below.


RESET
D1 mini

12. Now we will add the code. Plug in the wemos board (one with sensor) to your pc.

    a. First we will define the pins we are using, include the wifi library, define our network and password and define the server object.

```
#include <ESP8266WiFi.h>

int GreenLED = 15; //optional
int RedLED = 13;  //optional
int YellowLED = 5; //optional
int Echo = 12;
int Trig = 14;
long duration;
int distance;
int flashing = 0; //optional

const char* ssid = "your network";
const char* password = "your password";

WiFiServer server(80);
```

    b. Next is to define the setup loop. This will define our pins to output or input, connect to the wifi network. Make sure to open the serial monitor to get the IP address that will be printed there on setup. You will put that as the connection point in the client code so you can connect to the server. You will have to reupload the code to the client board. `IPAddress server(The server's IP address here!);`

```
void setup() {
  // put your setup code here, to run once:
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  pinMode(RedLED, OUTPUT);  //optional
  pinMode(GreenLED, OUTPUT);  //optional
  pinMode(YellowLED, OUTPUT);  //optional
  Serial.begin(9600);
   // Connect to Wifi Network
 Serial.println();
 Serial.println();
 Serial.print("Connecting to ");
 Serial.print(ssid);
 WiFi.begin(ssid, password);
```

```
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}

 Serial.println("\n");
 Serial.println("connected to ");
 Serial.println(ssid);
 //Start server
 server.begin();
 Serial.println("Server has started");
 //Print IP
 Serial.println("Server can be reached at: ");
 Serial.println("http://");
 Serial.println(WiFi.localIP());
 Serial.println("/");

}
```

c. The last step for the code is to make the main loop. This will wait for a client to connect. (optional) Next will ask if the flashing flag has been set, if so then it will flash. After that we get the distance reading. We will ask if the distance is greater or equal to 100cm, between 60 and 100 cm, between 40 and 60 cm and less than 40 cm. For each one the server will send a command to the client side to turn on the correct LED.

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


  // Check to see if client connection
 WiFiClient client = server.available();
 if (!client) {
   return;
 }


 Serial.println("new client connected");


   if (flashing == 1) { //optional
```

```
    digitalWrite(RedLED, HIGH);
    delay(800);
    digitalWrite(RedLED, LOW);
    delay(300);
    digitalWrite(RedLED, HIGH);
    delay(800);
    digitalWrite(RedLED, LOW);
    delay(300);
    digitalWrite(RedLED, HIGH);
    delay(800);
    digitalWrite(RedLED, LOW);
    delay(300);
  } //optional

  digitalWrite(Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);

  duration = pulseIn(Echo, HIGH);

  distance = duration * 0.034/2;  //0.034 is for the speed of sound divided by 2
//because it is there and back multipled by the time.  Distance = rate * time
  if (distance >= 100 ) { //far away still
    digitalWrite(GreenLED, HIGH); //optional
    digitalWrite(YellowLED, LOW); //optional
    digitalWrite(RedLED, LOW); //optional
    flashing = 0; //optional
    client.println("greenON\r"); // tell client to turn on its green led.
  }
  else if (distance > 60 && distance < 100) { //getting close but still not there
    digitalWrite(GreenLED, LOW); //optional
    digitalWrite(RedLED, LOW); //optional
    digitalWrite(YellowLED, HIGH); //optional
    flashing = 0; //optional
    client.println("yellowON\r"); // tell client to turn on its yellow led.
  }
  else if (distance > 40 && distance <= 60) { // good distance
    digitalWrite(GreenLED, LOW); //optional
    digitalWrite(YellowLED, LOW); //optional
    digitalWrite(RedLED, HIGH); //optional
    flashing = 0; //optional
    client.println("redON\r"); // tell clien to turn on its yellow led.
  }
```

```
  else if (distance <= 40) { // too close back up;
    digitalWrite(GreenLED, LOW); //optional
    digitalWrite(YellowLED, LOW); //optional
    digitalWrite(RedLED, LOW); //optional
    flashing = 1; //optional
    client.println("redFLASH\r");
  }

delay(1);
 Serial.println("Client disconnected");
 Serial.println("");

}
```

13. Now you can upload this code to the your wemos sensor board. Make sure to open the serial monitor to see what the IP address is and input that in the client's code. Make sure to reupload the client's code to the client's board. Serial monitor can be accessed from **Tools / Serial Monitor.**

Thought Questions:

1. Think of the interaction between your devices. How well would this scale to multiple devices? Is it easy to add another sensor? Another actuator?

This could easily scale to multiple device since the multiple clients could react to the sensors reading's. It would not be to hard to add another actuator.

2. What are strengths and weaknesses of the tennis-ball-on-a-string system that Don had originally? What are strengths and weaknesses of the IoT system that he developed? What enhancements would you suggest?

The problem with the tennis ball is that it can be knocked down and it is annoying to have hanging in your garage. The IOT system could be well installed and out of the way it also tells you information in a clear informative way. The weakness that I saw is that the distance can be off and the sensor can have some weird readings, also the system relies on internet and power. Internet can be known to be unreliable at times. So the tennis ball would always work in all conditions. Maybe if the sensor and actuator used batteries for backup if the power went out and used another form of communication it would be better since, both power and internet can go out.

3. What was the biggest challenge you overcame in this lab?

Figuring out how to get the sensor to turn on lights accurately for different distances.

4. Please estimate the total time you spent on this lab and report.

About 5 hrs in total.

Certification of Work:

I certify that the solution presented in this lab represents my own work. In the case where I have borrowed code or ideas from another person, I have provided a link to the author's work in the references, and included a citation in the comments of my code.

--Timothy Gow

Appendix:

Code can be found at my github here at:
https://github.com/Choyero/Arduino_distance_indicator.git