Miguel Augusto Tapia

Professor Sabal

Computer Org. and Arc.

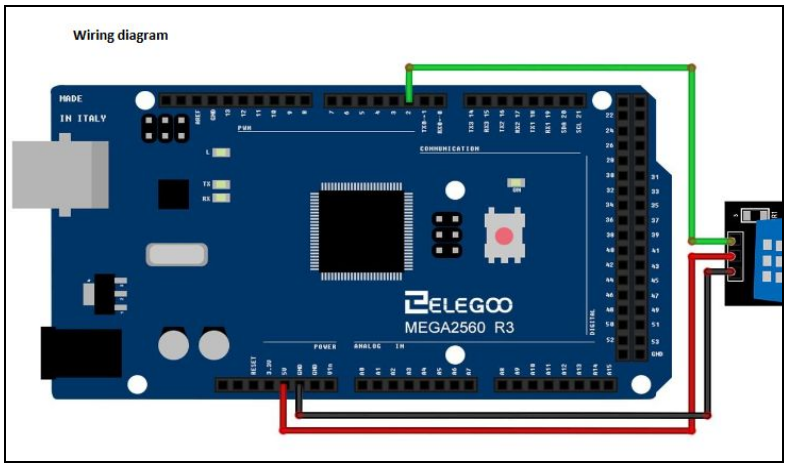
26 November 2019

Technical Document: Sensing Plant Health (ArduAgriSense)

Before trying to replicate this prototype, the ArduAgriSense, you must have an overview of what the prototype is trying to accomplish. The prototype's purpose is to provide farmers and everyday, common individuals with real-time, reliable data concerning their plants. Certainly, you can change the specific information reported by the prototype to suit your needs; however, for the purposes of this technical document, the device will report data concerning the temperatures surrounding the plants and leaf coloration. In an effort to simplify the user's experience, the device communicates over wifi. This removes the necessity for physical movement and decreases the maintenance of the devices. The prototype ensures plants' success and health by communicating data accurately and reliably in real time.

In order to successfully implement this prototype, you need to acquire some devices and modules. Undoubtedly, the prototype requires a microcontroller board like the Mega 2560 or Arduino Uno. Additionally, in order to capture data, the prototype needs sensors. The temperature data surrounding the device comes from a DHT module. In addition to the DHT module, the prototype will need a color sensor in order to read leaf coloration. Finally, there needs to be some type of wifi module so that data can be communicated to a server that can process the data. You can purchase these 4 products by navigating to amazon.com and searching the following amazon ids: B01CZTLHGE, B010O1G1ES, B07D55NSTV.

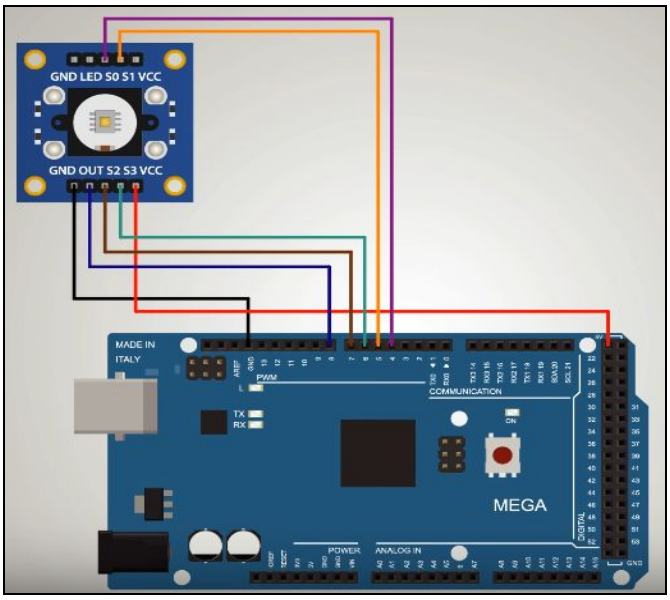
Now that you have gathered up all the components needed to build the ArduAgriSense, the first step is to getting temperature from a DHT module. Temperature data, including humidity, is crucial to your operations, because plants simply cannot survive in extreme conditions. By incorporating a DHT sensor, you can also categorize changes in temperature over time and see how plants are responding. The kit listed above includes a DHT11 sensor. You must connect GND to GND on the arduino, VCC to 5V on the arduino, and DATA to a DATA pin ("Lesson 12"). An external breadboard is certainly an option as well. There are supplemental pictures, outlining the code per implementation step, listed below in the section called additional resources. Once you wire up the code, as explained below, and it is getting data, this implementation step is complete as you will append the data in implementation step 3 before sending a POST to the server.



("Lesson 12")

The second implementation step focuses on fetching data from the color sensor. A plant's coloration is critical, because climate changes rapidly. The idea of utilizing the color sensor to predict a plant's behavior works through the "sensor-to-plant principle [which] allows for plants to be monitored" (Bao). There are many ways to get data concerning color. One "monitoring capability [is] the use of digital RGB imaging" (Bao). You will be setting this form of color detection in this prototype. The module listed in the products needed will assist you with this, because digital RGB imaging requires photodetectors; these are "sensors which respond to the

electromagnetic radiation of the spectrum" (Botero). Through this response, the application will have information about the color it is detecting. The entire premise of these detectors, "made of semiconductors that are responsive to photo-excitation," "is to show a particular spectral region as output" (Botero). Once you have the color sensor, you must wire it. This data enables plant understanding.



(Dejan)

As stated above, you can connect it to the microcontroller itself; however, not everything is going to fit, so an external breadboard is a fantastic option. The schematic, attached to the right, shows how to connect this particular module. The code that allows data extraction from the module is at the end of this document. You must begin by defining the pins that the sensor communicates with and set a frequency (Dejan). When the initial defines are complete, the setup code includes a pinMode command for the 4 pins receiving data as output, and the sensorOut on the module becomes the input to the arduino (Dejan). Finally, the loop part begins with setting a particular pin's section to low in order to receive a read command; pulseIn will send output frequency from sensorOut with the particular data (Dejan). You should repeat this code snippet for red, green and blue.

Finally, the gathered data must communicate to a server of sorts. For this prototype, you will create a PHP server that receives data communicated by the ESP8266 WiFi Module. The data will be sent to the following server: csinfo.cairn.edu/~mt815/arduiagrisense. If the plant is

experiencing unfavorable conditions, regarding leaf coloration or temperature, the server will update a dashboard with an alert and send an email to users users. In order to achieve this, you must connect the wifi module, in this case the ESP8266, to wifi. The connection for this model is in the additional resources below. Once a connection is made, you must send all the gathered data from the DHT and Color Sensor Module as a payload within the POST body. Above this text is an example of how the send would work. The server is the server you set aside and the POST variable can be anything. For this demo, the server code uses *temp.* The final *client.print*(data) sends the payload to the server. You can outline the server in any way to suit your needs; however, the server code below just outputs data to an html page.

```
void loop() {
  int reading = analogRead(sensorPin);
  float voltage = reading * 5.0;
  voltage /= 1024.0;
  float temperatureC = (voltage - 0.5) * 100 ;
  float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;

  postData = postVariable + temperatureF;

  if (client.connect(server, 80)) {
    client.println("POST /test/post.php HTTP/1.1");
    client.println("Host: www.elithecomputerguy.com");
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.print("Content-Length: ");
    client.println(postData.length());
    client.println();
    client.print(postData);
  }
```

("How to Post")

Thus, the ArduAgriSense enables your success through effective and reliable communication concerning a plant's health. Without a doubt, the possibilities, regrading the data sent from the ArduAgriSense, are endless. Likewise, the server has infinite capabilities. For example, the database could store the data in a database, and then process the data through SQL in order to predict behavior and better understand the plants. As a whole, the ArduAgriSense has incredible potential, because it can determine changes in temperature and leaf coloration. These are crucial because they determine a plant's health.

Additional Resources

Sample Code: DHT Sensor

```
// DHT INCLUDE AND CONSTANTS
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
```

```
  // Begin DHT Module and LCD Display
  dht.begin();
  lcd.begin(16,2);
}

// GetValsAndCheck() - Gets DHT Data And Che
int getValsAndCheck() {
  // Read Celcius
  tCelcius = dht.readTemperature();

  // Read Fahrenheit
  tFahrenheit = dht.readTemperature(true);

  // Read Humidity
  humidity = dht.readHumidity();
```

When you write this code, your DHT module will initialize, so it can read incoming data.

(Tapia)

Sample Code: Color Sensor

```
#define S0 4
#define S1 5
#define S2 6
#define S3 7
#define sensorOut 8

int frequency = 0;

void setup() {
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  pinMode(sensorOut, INPUT);

  // Setting frequency-scaling to 20%
  digitalWrite(S0,HIGH);
  digitalWrite(S1,LOW);

  Serial.begin(9600);
}

void loop() {
  // Setting red filtered photodiodes to be read
  digitalWrite(S2,LOW);
  digitalWrite(S3,LOW);
  // Reading the output frequency
  frequency = pulseIn(sensorOut, LOW);
  //Remaping the value of the frequency to the RGB Model of 0 to 255
  frequency = map(frequency, 25,72,255,0);
  // Printing the value on the serial monitor
  Serial.print("R= ");//printing name
  Serial.print(frequency);//printing RED color frequency
  Serial.print("  ");
  delay(100);
```

You first initialize the pins used by the color sensor, these must then be set to output. You set the sensorOut to input, for data will come from there. The snippet under loop is what gets the data for the red photodiodes. The code here repeats for green and blue. In sum, when you write the code above, red (r), green (g), and blue (b) values return.

(Dejan).

Sample Code: Wifi Module

```
#include "ESP8266WiFi.h"

const char* ssid = "ssid"; //Enter SSID
const char* password = "password"; //Enter Password

void setup(void)
{
  Serial.begin(115200);
  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
     delay(500);
     Serial.print("*");
  }

  Serial.println("");
  Serial.println("WiFi connection Successful");
  Serial.print("The IP Address of ESP8266 Module is: ");
  Serial.print(WiFi.localIP());// Print the IP address
}
```

```
<?php
$time = time();
$tempF = $_POST["temp"];
$file = 'temp.html';
$data = $time."  -  ".$tempF;
file_put_contents($file, $data);
?>
```

The code here is simple. You define the ssid for the network and password. In the setup section, your code will continue to run until the status changed to connected. The server code simply takes the time and outputs your incoming “temp” data with a timestamp to an html file. Excitingly, though some additional code and research, you can make the server send an email with the data or store the data in a database.

(“How to Connect”) (“Write POST Data”).

Works Cited

Bao, Yin, et al. "Assessing plant performance in the Enviratron." *Plant Methods*, vol. 15, no. 1, 2019, p. NA. *Gale OneFile: Agriculture*, https://link.gale.com/apps/doc/A604540072/PPAG?u=philbibu&sid=PPAG&xid=de13b62d. Accessed 23 Nov. 2019.

Botero V., J.-S., et al. "CHARACTERIZATION OF PHOTODECTORS USING A MONOCHROMATOR AND A BROADBAND LIGHT SOURCE IN THE XYZ COLOR SPACE." *International Journal on Smart Sensing and Intelligent Systems*, vol. 9, no. 2, 2016, p. 752+. *Gale OneFile: Computer Science*, https://link.gale.com/apps/doc/A544247573/CDB?u=philbibu&sid=CDB&xid=b048c37c. Accessed 23 Nov. 2019.

Dejan. “Arduino Color Sensing Tutorial - TCS230 TCS3200 Color Sensor.” *HowToMechatronics*, HowToMechatronics.com, 4 Aug. 2018, https://howtomechatronics.com/tutorials/arduino/arduino-color-sensing-tutorial-tcs230-tcs3200-color-sensor/.

“How to Connect ESP8266 to WiFi: A Beginner's Guide.” *Electronics Hub*, Electronicshub.org, 30 Mar. 2018, https://www.electronicshub.org/connect-esp8266-to-wifi/.

"Lesson 12 DHT11 Temperature and HumiditySensor." Elegoo - The Most Complete Starter Kit Tutorial for MEGA2560, V1.0.17.7.9, 91-96.

“Write POST Data to Server with Arduino Uno with WiFi.” *Eli the Computer Guy*, 18 July 2019, https://www.elithecomputerguy.com/2019/07/write-post-data-to-server-with-arduino-uno-with-wifi/.