Group Member Names: Robert English, Anh Bui, Peter Polyakov, Kendra Young

Course and Quarter: ENGR114, Winter 2018

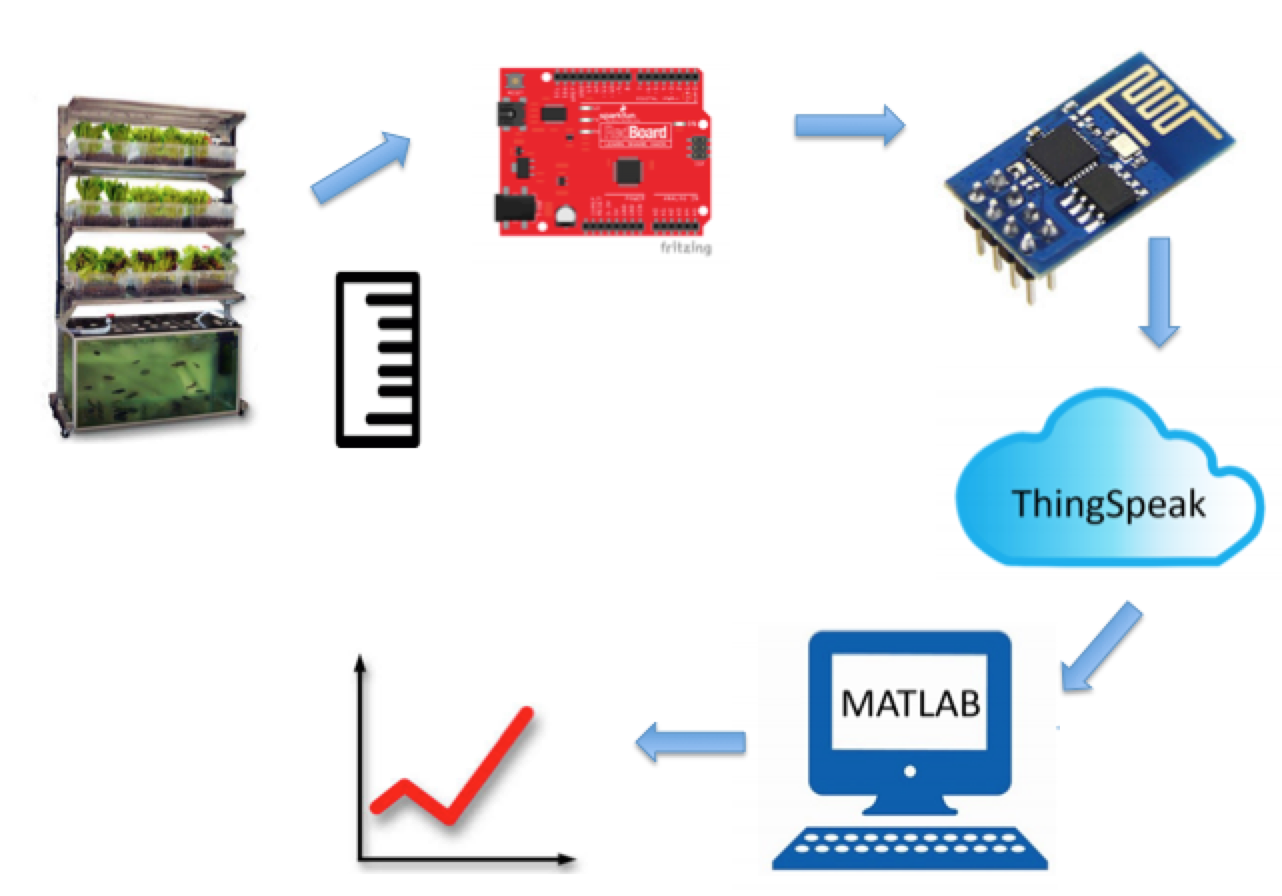
Date: March 20, 2018

Rev. 2

MATLAB-Arduino Water Level Sensor with Wireless Data Upload

**Problem Statement**:

Our group was tasked with building upon a previous group’s work of measuring water level. The previous group used MATLAB and an Arduino red board that was coded to interface with the water level sensor. They built Arduino code that collected the data from the water level sensor, which then transferred the data to MATLAB to be analyzed. Building upon this group’s work, our group wanted to create an Arduino code that would gather water level measurements and wirelessly upload the data to ThingSpeak using a wifi module. Then our MATLAB program would take the data from ThingSpeak and then analyze it.



**Hardware Setup:**

Bill of Materials:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part Name | Purpose | Item Name | URL | Price |
| Arduino | a programmable microcontroller circuit board and a piece of [software](http://arduino.cc/en/Main/Software)(IDE) that runs on your computer, used to write and upload computer code to the physical board | SparkFun RedBoard-Programmed with Arduino | https://www.sparkfun.com/products/13975 | $19.95 |
| Water level sensor | Measures the water level - a solid-state sensor with a resistive output that varies with the level of the fluid | 12” eTape Liquid Level Sensor | https://www.adafruit.com/product/464 | $39.95 |
| 470 Ω resistor | Provides an additional 470Ω of electrical resistance to the circuit | 470-Ohm 1/2-Watt 5% Carbon Film Resistor | https://www.amazon.com/Projects-100EP512470R-470-Resistors-Pack/dp/B0185FIDF0/ref=sr\_1\_3?ie=UTF8&qid=1521524413&sr=8-3&keywords=470+ohm+resistor | $ 0.07 |
| Breadboard | Solderless breadboard used as a construction base in developing an electronic circuit | Breadboard - Self-Adhesive (White) | https://www.sparkfun.com/products/12002 | $4.95 |
| Wifi Adapter | Give any microcontroller access to your WiFi network | WiFi Module - ESP8266 | https://www.sparkfun.com/products/13678 | $ 6.95 |
| Jumper Wires | 155mm long jumpers, interconnect the Arduino and breadboard components | Jumper Wires Premium 6" M/M Pack of 10 | https://www.sparkfun.com/products/8431 | $(3.95)\*2 =7.90 |
| Breadboard Base | Plastic plate with space to  Hold the Arduino and breadboard in place | Arduino and Breadboard Holder | https://www.sparkfun.com/products/11235 | $3.95 |
| Mini\_B USB cable | USB 2.0 type A to Mini-B 5-pin cable-Connects arduino to computer | SparkFun Mini\_B USB cable-6ft | https://www.sparkfun.com/products/11301 | $3.95 |

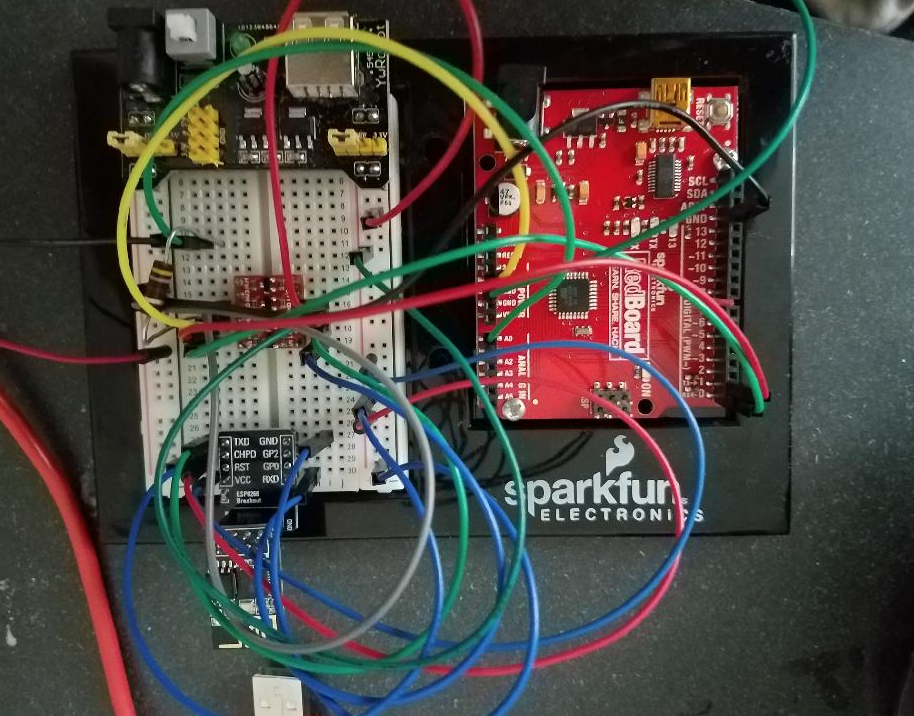
Hardware Schematic:

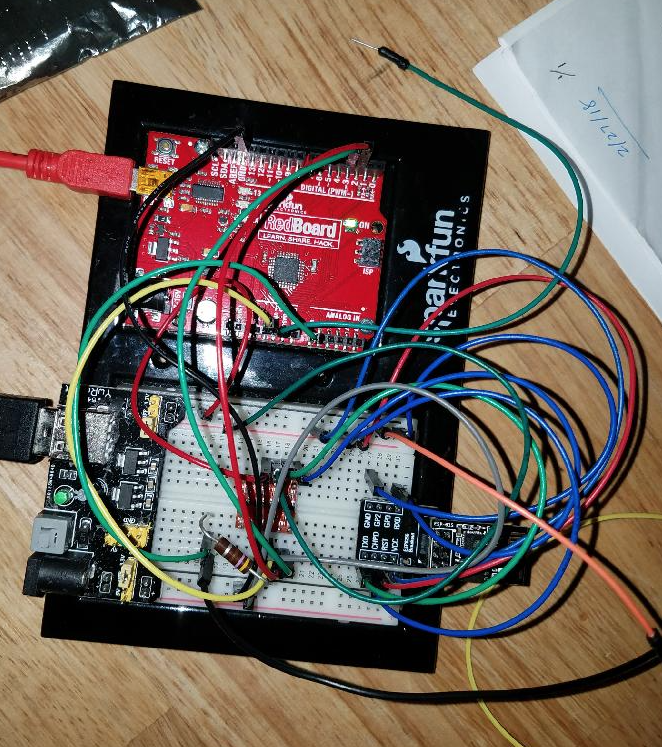
Hookup Guide:

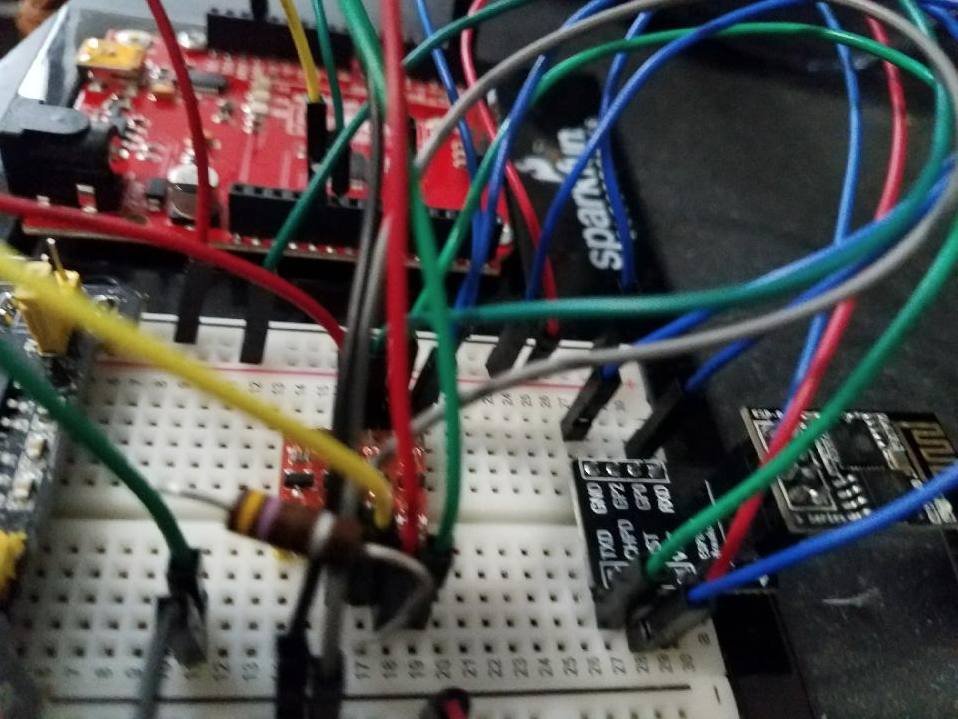
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part | Pin | Connector | Pin | Part |
| Arduino | 5v | M2M | in | 470 ohm resistor |
| 470 ohm resistor | out | M2M | A0 | Arduino |
| Wire | 2 | M2M | HV2 | Logic board |
| Wire | 3 | M2M | HV1 | Logic board |
| Wire | GND | M2M | GND | Breadboard |
| Wire | A0 | M2M | Breadboard | Resistor |
| Wire | 5V | M2M | HV | Logic board |
| Wire | GND | M2M | GND | ESP |
| Wire | 5V | M2M | RST | ESP |
| Wire | 5V | M2M | CHPD | ESP |
| Wire | 5V | M2M | VCC | ESP |
| Wire | LV1 | M2M | RDX | ESP |
| Wire | LV2 | M2M | TDX | EPS |
| Wire | LV | M2M | 5V | Breadboard |
| Wire | GND (3V) | M2M | GND | Logic board |
| Wire | GND | M2F | GND | Water level sensor |
| Wire | 5V | M2F | A0 | Water level sensor |

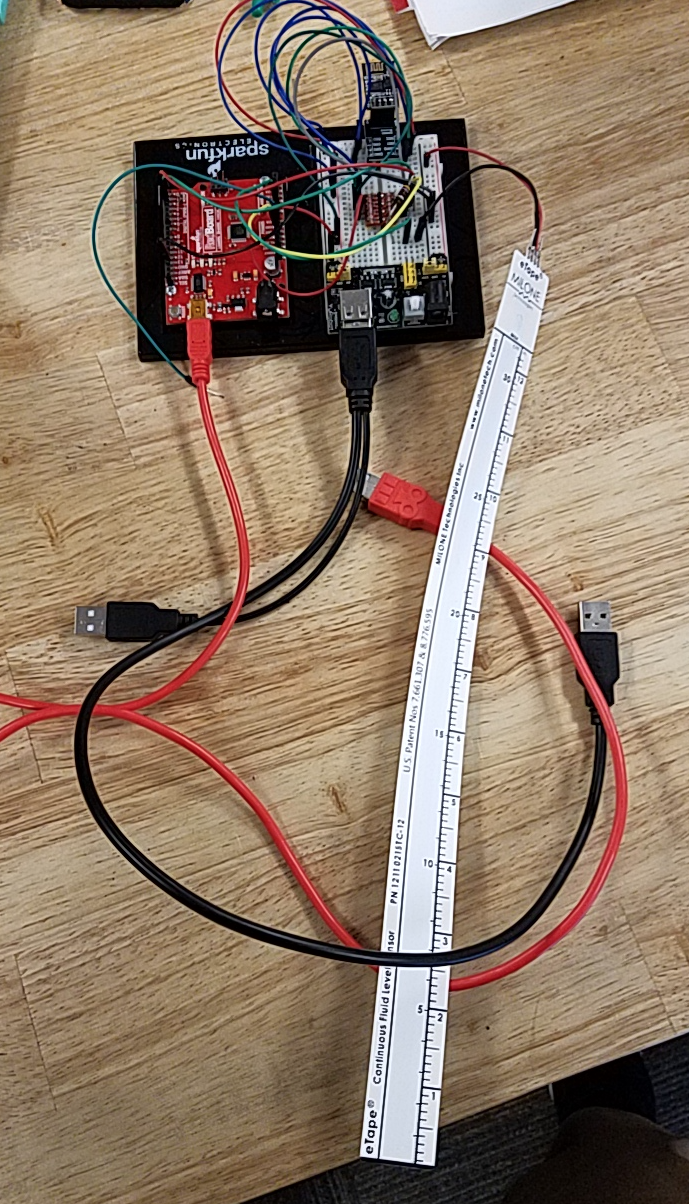
The power board and arduino are powered through chords connecting to the computer. The arduino connects to the logic board through the 0 to the HV1, the 1 to the HV2 and the 5V to the HV. The arduino also connects the A0 to the power after it goes through the 470 ohm resistor and connect the GND to the GND on the 3.3V side of the breadboard. The logic board connects the ESP through the LV1 to the RXD and the LV2 to the TXD. This connection helps in the receiving and outputting of the ESP. The ESP connects the GND to the breadboard GND and the 5V to the RST, VCC, and the CHPD. The water level sensor would then connect to the GND with one wire and the voltage from the 5V passing through the resistor on the other side. The water level sensor piece then uses this to output different ohm values to us based on the level of the water.

Images:









**Code**:

Arduino:

The arduino code for reading the sensor data was already written by a previous engineering team, and the ESP8266 code was fairly easy to find on the internet. The latter made up a far greater portion of the completed code. The complicated part was implementing the code on the hardware.

The code does not make use of any libraries other than SoftwareSerial which is built-in to the arduino IDE. It’s a fairly basic sketch that should be highly accessible to those who wish to build upon it.

[Github link to code here](https://github.com/roburpo/PCC-Water-Level-WiFi)

|  |  |
| --- | --- |
| /\* Water Level Sensor System |  |
|  | Author: PCC ENGR114 Student Spring 2017 |
|  | Date: 06/01/2017 |
|  | Version: 2.0 |
|  | Description: This program reads the analog voltage signal from the eTape |
|  | and displays the measurements digitally in Arduino. |
|  | Connect pin #2 of the sensor to ground, then pin #3 to a 560 ohm resistor. |
|  | The other side of the 560 ohm resistor to VCC 5V to create a resistor |
|  | divider. The ADC pin connects to the point between the resistor and sensor. |
|  | \*/ |
|  | // the value of the resistor (not the eTape) |
|  | #define SERIESRESISTOR 470 |
|  |  |
|  | // Sensor pin input |
|  | #define SENSORPIN A0 |
|  |  |
|  | // resistance values |
|  | #define zero\_depth\_resistance 550 // Resistance value when no liquid is present. |
|  | #define max\_resistance 60 // Resistance when liquid is at max line. |
|  |  |
|  | #include <SoftwareSerial.h> |
|  | #define RX 2 |
|  | #define TX 3 |
|  | SoftwareSerial esp8266(RX,TX); |
|  |  |
|  | String AP="Samsung Galaxy S7 6335"; |
|  | String PASS="A1234567"; |
|  |  |
|  | String API = "R03IAJF3KWV0HC4M"; |
|  | String HOST = "api.thingspeak.com"; |
|  | String PORT = "80"; |
|  | String field = "field1"; |
|  |  |
|  | int countTrueCommand; |
|  | int countTimeCommand; |
|  | boolean found = false; |
|  |  |
|  | void sendCommand(String command, int maxTime, char readReplay[]) { |
|  | Serial.print(countTrueCommand); |
|  | Serial.print(". at command => "); |
|  | Serial.print(command); |
|  | Serial.print(" "); |
|  | while(countTimeCommand < (maxTime\*1)) |
|  | { |
|  | esp8266.println(command);//at+cipsend |
|  | if(esp8266.find(readReplay))//ok |
|  | { |
|  | found = true; |
|  | break; |
|  | } |
|  |  |
|  | countTimeCommand++; |
|  | } |
|  |  |
|  | if(found == true) |
|  | { |
|  | Serial.println("OYI"); |
|  | countTrueCommand++; |
|  | countTimeCommand = 0; |
|  | } |
|  |  |
|  | if(found == false) |
|  | { |
|  | Serial.println("Fail"); |
|  | countTrueCommand = 0; |
|  | countTimeCommand = 0; |
|  | } |
|  |  |
|  | found = false; |
|  | } |
|  |  |
|  | void setup(void) { |
|  | Serial.begin(9600); |
|  | esp8266.begin(9600); |
|  | sendCommand("AT",5,"OK"); |
|  | sendCommand("AT+CWMODE=1",5,"OK"); |
|  | sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",20,"OK"); |
|  | } |
|  |  |
|  | void loop(void) { |
|  |  |
|  | float reading; |
|  |  |
|  | reading = analogRead(SENSORPIN); // voltage reading from eTape |
|  |  |
|  |  |
|  | // convert the value to resistance |
|  |  |
|  | reading = (1023 / reading) - 1; |
|  | reading = SERIESRESISTOR / reading; |
|  | char buffer[8]; |
|  | String resistance = dtostrf(reading, 1, 4, buffer); |
|  | //Serial.println(reading); |
|  |  |
|  |  |
|  | // connect and send data to ThingSpeak |
|  | String getData = "GET /update?api\_key="+ API +"&"+ field +"="+resistance; |
|  |  |
|  | sendCommand("AT+CIPMUX=1",5,"OK"); |
|  | sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+ PORT,15,"OK"); |
|  | sendCommand("AT+CIPSEND=0," +String(getData.length()+4),4,">"); |
|  | esp8266.println(getData);delay(1500);countTrueCommand++; |
|  | sendCommand("AT+CIPCLOSE=0",5,"OK"); |
|  |  |
|  | delay(16000); |
|  |  |
|  |  |
|  |  |
|  | } |

Matlab:

# Group Project

* Authors: ENGR114 Winter 2018 Student Group
* Group Members: Robert English, Anh Bui, Peter Polyakov, Kendra Young.
* Date: Mar 20, 2018
* Descriptive: This MATLAb code to pull down the water level data from ThinkSpeak.com, clean up and analyze the data.

## **Contents**

* Clear all the command window, variable workspace, and close all the window
* Get the water level data from ThinkSpeak.com
* Clean up the data to get the water level
* Plot the water level vs. time recorded

## **Clear all the command window, variable workspace, and close all the window**

clc; clear; close all;

## **Get the water level data from ThinkSpeak.com**

data\_num = input('\n Enter how many data you want to measure: ')  
  
raw\_data = webread(sprintf('https://api.thingspeak.com/channels/448325/fields/1.csv?results=%d', data\_num));

Error using input  
Cannot call INPUT from EVALC.  
  
Error in Water\_Wifiless\_Project (line 14)  
data\_num = input('\n Enter how many data you want to measure: ')

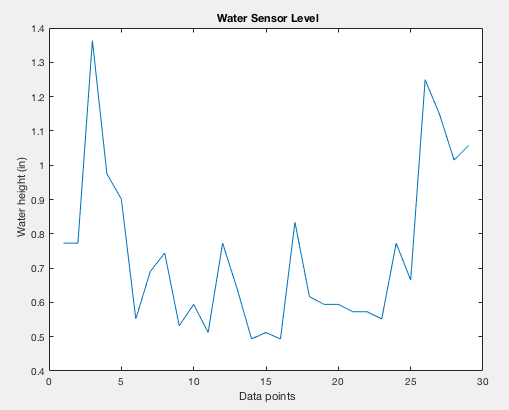
## **Clean up the data to get the water level**

data = table2array(raw\_data(:,3)) % pull out the third column from the data  
  
x\_values = [1:numel(data)]'; % determine the values for x-axis  
  
y\_str = ' '; % create an empty string with 1 row x 16 cols  
  
for i=1 : numel(data) % use the for loop to run the data order  
  
 if numel(data{i}) == 17 % use the if function to check if the y values is over 10000  
  
 y\_add = data{i}(1:16); % truncate the last digit data  
  
 y\_str = [y\_str; y\_add]; % show the table of the strings containing the y values  
  
 y\_values = str2num(y\_str(:, 8:end)); % adding y\_value at the index to the y\_values matrix  
  
 else % if else the y values is less than 10000k  
 y\_add = data{i}(1:16); % adding one row for the string after the index ran  
  
 y\_str = [y\_str; y\_add]; % show the table of the strings containing the y values  
  
 y\_values = str2num(y\_str(:, 8:end)); % adding y\_value at the index to the y\_values matrix  
 end  
  
end  
 y\_values = y\_values; % combine all the y values  
  
 y\_values = (y\_values - 4000)/7742; % convert the resistance values to the water height values  
  
 table = [x\_values y\_values] % create a table of x and y values

## **Plot the water level vs. time recorded**

figure(1) % create a figure 1  
  
plot(x\_values, y\_values) % plot the x and y values  
  
ylabel(' Water height (in)') % add name to the y-axis  
  
xlabel(' Data points') % add name to the x-axis  
  
title(' Water Sensor Level') % add title to the plot

**Results**:



Challenges:

Our first issue was that the arduino board provided was presumably dead. It would not respond to uploading code from the arduino IDE. However, looking back, I believe I know what was wrong. If the 0 (RX) pin of the arduino is populated, then the serial connection will override the USB connection. Thus, SoftwareSerial should be used, unless the arduino code will never have to be changed. Then, it is important to mention that AT commands sent through serial monitor will not work if the esp is not connected to hardware serial on the arduino.

Connecting the esp to your home WiFi is totally painless, but connecting to college WiFi is an entirely different story. Although WPA2 is supported by the esp, PEAP is not. One could use the MAC address of the esp to allow access to an access point with such an authentication protocol.

**Future Work**:

With wireless ability, there are limitless possibilities. Future work could include optimizing the code and also working to build a code that when the water level goes out of the optimal level range, it can interact with the water pump and manage the levels. This idea of measuring a water level and adjusting it when it becomes lowers or rises above the optimal level could also be applied to other mediums. For example, a program could be created to measure the level of flour in a flour container and when it is almost empty, a code could be written so that the person is notified and/or more flour is ordered automatically (like an Amazon dash button).

**License - MIT**

Copyright 2018 Kendra Young

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

**References**

1. Worksheet *ENGR114\_Arduino\_IoT\_Project\_Description*.
2. Example of MATLAB\_Water\_Sensor from previous ENGR114 Project on github.