Work in Progress: Adding the Internet of Things to a Freshman-level Engineering Course

Dr. W. Davis Harbour, Louisiana Tech University

Dr. Davis Harbour is a Senior Lecturer and Program Chair for Electrical Engineering at Louisiana Tech University. He earned his BS and MS degrees at the University of Oklahoma and he earned his PhD degree at the University of Arkansas. His primary teaching responsibilities are in the freshman and sophomore engineering courses, and his interests include microcontrollers, data acquisition systems, control systems, and engineering education. He is a member of ASEE and IEEE.

Dr. Stan Cronk, Louisiana Tech University

Dr. Stan Cronk is a Senior Lecturer in Industrial Engineering at Louisiana Tech University. He earned his BS and PhD degrees at Louisiana Tech University in Biomedical Engineering. His primary teaching responsibilities are in Industrial Engineering as well as the general freshman and sophomore engineering courses. He is a registered Professional Engineer in Louisiana for Electrical and Computer Engineering. He is a member of ASEE and IISE.

Mr. Nishant Shakya, Louisiana Tech University

Nishant Shakya is a second year Ph. D. student in Computational Analysis and Modeling (CAM) at Louisiana Tech University. He holds a BE degree in Electronics and Communication Engineering from Institute of Engineering, Tribhuvan University in Nepal. His research interest is centered around the intersection of information theory and computer science. He takes a multidisciplinary approach in his research that encompasses the fields of electrical engineering, mathematics, computer science, and statistics. He is a registered engineer of NEC (Nepal Engineering Council) and a member of AMS (American Mathematical Society).

Work in Progress: Adding the Internet of Things to a Freshman-level Engineering Course

Abstract

The Internet of Things is a current phenomenon that all engineering students need to have an awareness of and an appreciation for. Regardless of their chosen field of study and ultimate job upon graduation, it is almost certain that they will encounter the Internet of Things during their engineering careers. This paper describes an effort to introduce the Internet of Things to a freshman-level engineering course at Louisiana Tech University that allows the students to understand and experience some of the technology involved in this phenomenon. By adding a small amount of additional hardware to the existing Arduino microcontroller platform already in use in our year-long engineering course sequence, students are able to apply techniques they have already learned to implement a WiFi module and send data to a database. They are also able to open a web browser and see their data displayed on graphical objects in real time.

Introduction

The Internet of Things is the connection of billions of devices to each other through the Internet. Conservative estimates suggest that 10-20 billion devices could be part of the Internet of Things by 2020, while other estimates suggest the number could be closer to 100 billion. Regardless of the number of devices connected, the Internet of Things will be a part of the lives of our students and it is a very timely topic to include in a freshman-level engineering course. Even more so if that course, or course sequence, is built around a microcontroller platform and already includes topics such as programming, sensor data acquisition, data analysis and control systems.

The term "Internet of Things" covers a wide range of topics. On one end of the spectrum it includes sensors measuring values such as temperature, pressure, heartrate, liquid levels in tanks and so on. On this end of the spectrum are also sensors that monitor the status of systems such as whether or not doors are open or closed, whether or not a room is occupied, whether or not lights are on in a room or house, and so on. The data from these sensors is collected by a microcontroller-based device that has the capability to connect to the Internet. This device can be connected to just one sensor or to a group of sensors, and it can connect to the Internet either wirelessly or through a wired connection. Since the Internet is used, the middle area of the Internet of Things spectrum includes all of the current technology involved with the existing Internet, including wide area networks, local area networks, Internet Protocol (IP) addresses (both IPv4 and IPv6 versions), gateways, routers, subnet masks, and so on. The opposite end of the spectrum from the sensors includes the storage of the data and the analysis of the data, and therefore includes devices such as servers, databases, and software to display and process all of the data. For this reason, some refer to this end of the Internet of Things spectrum as the "big data" end.

The work in progress discussed in this paper focuses primarily on the sensor data end of the spectrum, but touches on aspects from across the spectrum. We discuss how the microcontroller platform already in use in our freshman-level introduction to engineering course sequence is enhanced to allow that device to connect wirelessly to the Internet. We discuss how data from a sensor is acquired by this microcontroller and sent to a server via that wireless Internet connection. And we then discuss how that data is stored in a database and made available for viewing through graphical objects on a webpage.

Freshman-level engineering course

At Louisiana Tech we operate on a quarter system, and all incoming freshmen engineering students are required to take a three-course introduction to engineering sequence that spans their entire freshman year. The course sequence is ENGR 120, ENGR 121 and ENGR 122 and each course is a 2 semester-credit-hour (SCH) course that meets twice a week for 110 minutes. Each course is a very hands-on, project-based course, and each lecture within each course includes a combination of material delivered via lecture and material delivered through hands-on activities. On the first day of the first course, students receive a kit that includes most of the items that they will use in all three courses for the rest of the year. These kits include, amongst other things: an Arduino microcontroller platform, a multimeter, a breadboard, a switch, a vibrating DC motor, a dial caliper, a wire stripper, a wire crimper, a battery pack, two servo motors, two wheels with two tires, a 12 VDC power supply, safety glasses, several screwdrivers, a pocket knife, needle nose pliers, wire, Teflon tape, and numerous resistors, transistors, LEDs, capacitors, relays and diodes.

In the first course, ENGR 120, the first topic covered is the fundamental engineering topic of electricity. Students learn about Ohm's Law, Kirchhoff's Voltage Law (KVL), Kirchhoff's Current Law (KCL), how to measure resistance, how to measure voltage, how to measure current, how to build a circuit to light up an LED, and how to build a circuit to monitor a switch. They learn how to build circuits that allow them to prove KVL and KCL, and they learn how to use these circuits to see and measure resistors in series and in parallel. They learn how to construct a robot platform consisting of the Arduino microcontroller, a breadboard, two servo motors, two wheels and tires, a battery pack and a switch. They learn C programming in this class, they learn to control their servo motors which allows them to move their robot platform, and this course concludes with a robot challenge exercise. In this first course the students also learn how to design an impeller for a centrifugal pump using 3D modeling software. The impellers are printed on a 3D printer and the students insert them into pump bodies that they fabricate using milling machines. The students collect data as they use their pumps to pump water at varying heights and from that data they are able to compute the efficiency of their pumps.

In the second course, ENGR 121, the students use the pumps they designed and fabricated in ENGR 120 to construct a model of a salt water aquarium. They construct a wooden platform to hold a reservoir, and they connect tubing to provide a flow loop from the reservoir through their pump and back to the reservoir. In this flow loop they insert a conductivity sensor (that they fabricate) to measure the salinity of the water in the system. They also add to the platform two tanks, one containing very salty water to raise the salinity in the system and the other containing deionized water to lower the salinity of the water in the system. Each tank is connected to the reservoir via a solenoid valve that can be opened and closed by the Arduino microcontroller platform. They learn the basic fundamentals of control systems in this course and over the duration of this course they learn to program their microcontrollers to maintain a desired salinity value in the model of the aquarium. They also learn how to measure temperature using a thermistor connected to their microcontroller and how to turn a heater on and off. By inserting the heater and the thermistor into the reservoir, they also learn to control the temperature of the water in the system as well as the salinity. Other topics covered in this second course include salt water chemistry, conservation of mass and conservation of energy.

In the third course, ENGR 122, the first half of the quarter focuses on the fundamental engineering topics of statics and engineering economics. For the topic of statics, students learn to draw free body diagrams, they learn about static equilibrium, they learn about moments and torque, they learn about non-concurrent forces, they learn about the torque and rpm of gears and gear trains, and they learn to measure and compute the efficiency of a DC gearmotor. For the topic of engineering economics, students learn about simple and compound interest, they learn about the present value and future value of money, they learn about cash flow diagrams and they learn about uniform series. In the first half of the quarter time is also allocated in each class to learn about different sensors that can be connected to the Arduino microcontroller platform to measure or monitor some physical process. In the first half of the quarter the students also learn about the product design process, and then the entire second half of the quarter is devoted to a product that they themselves design. The design requirements are very open ended, with the only requirements being that they must include the Arduino microcontroller platform, they must include one or more of the sensors discussed in class, and their product solve some problem that they have identified in their lives as college freshmen. Examples of products designed by students in this course include: a TV tracking system that follows the motion of people in a room, a system to automatically lock and unlock a door based on a person approaching or moving away from the door, a system to automatically inflate an air bag inside and around a bicycle helmet when a fall is detected, a system to automatically repair leaks in irrigation pipes, and a system to send out GPS data via text message when sensors detect that a person has fallen and has not moved in a given period of time (with the intent of being used by firefighters to find and rescue colleagues who have fallen in a building).

Hardware

With all of the material already being covered in the ENGR 120, 121 and 122 courses it was difficult to determine where to add the concept of the Internet of Things. After much deliberation we decided to add this topic midway through the third course, ENGR 122, to coincide with all of the lectures on different sensors and data acquisition. To add this topic, some additional hardware was required that was not included in the kits they received in ENGR 120. This additional hardware includes: a small breadboard (1.8" x 1.4" x 0.4") to hold the new circuitry, a 3.3 V linear regulator to provide additional power, two resistors to form a voltage divider circuit, a potentiometer to provide a simulated sensor signal, and a WiFi module that would allow the Arduino microcontroller platform to connect to the Internet wirelessly.

The WiFi module selected to provide wireless Internet connectivity was the ESP8266-01 module. This module is very low cost at approximately $3.00 per unit, available from a wide range of suppliers, and provides all of the functionality needed for this project. The first issue to be addressed when connecting this module to the Arduino microcontroller platform is the issue of voltage differences. The Arduino is powered by a 5 V regulator, while all ESP8266 modules are powered by a 3.3 V regulator. There is a 3.3 V regulator on board the Arduino microcontroller platform, but it provides very low current and much less than the ESP8266 requires when transmitting. Thus, the need to add a separate 3.3 V linear regulator to the small breadboard to be powered by the 5 V regulator on the Arduino microcontroller platform. The communication interface between the Arduino microcontroller and the ESP8266 is RS-232 serial communication at 9600 baud. This interface utilizes a wire to transmit information from the Arduino to the ESP8266, labeled as TX on the Arduino and as RX on the ESP8266, and a wire to receive information by the Arduino from the ESP8266, labeled as RX on the Arduino and as TX on the ESP8266. Since the ESP8266 is a 3.3 V device, the signal coming from TX on the ESP8266 to RX on the Arduino will never be above this voltage, so this signal is safe to connect to the Arduino powered at 5 V. However, since the Arduino is powered at 5 V, the signal coming from TX on the Arduino to RX on the ESP8266 will be at this voltage, and this signal is too high for the ESP8266 powered at 3.3 V. So, a voltage divider must be employed to lower the voltage of this signal to a value that is safe for the ESP8266 to process. Connecting the TX signal from the Arduino to a circuit consisting of a 1 kΩ resistor in series with a 2 kΩ resistor will divide this 5 V signal. The value of the voltage across the 1 kΩ resistor will be one-third of 5 V and the value of the voltage across the 2 kΩ resistor will be two-thirds of 5 V. It is then safe to connect the TX signal from between the 1 kΩ resistor and the 2 kΩ resistor to the RX pin on the ESP8266. The topics of resistors in series and voltage division are covered in ENGR 120, so the students are already familiar with these concepts. The final connection on the ESP8266 is that the CH_PD pin must be tied to 3.3 V.

The last bit of hardware needed on the small breadboard is the potentiometer to provide a simulated sensor signal. This signal is connected to one of the analog input pins on the Arduino microcontroller platform, and this allows the potentiometer to be connected to 5 V. One wire brings 5 V from the Arduino platform to the potentiometer, one wire brings ground from the Arduino platform to the potentiometer, and one wire sends the voltage signal from the potentiometer back to the analog input on the Arduino platform. All of the circuitry that is placed on the small breadboard is shown in Figure 1 below.

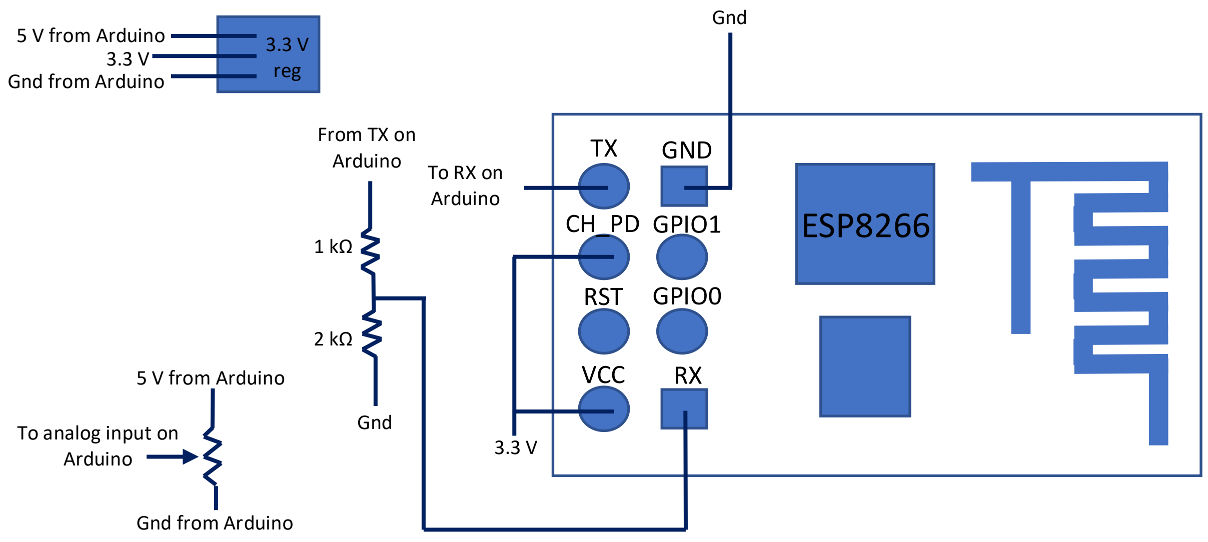

Figure 1. Circuitry required on small breadboard

The last remaining hardware change concerns the wireless network used in our facility. All of the existing wireless networks are secure networks requiring some form of authentication in order to connect to the network. The ESP8266-01 modules that we selected do not have the capability to provide this authenticity as they can only provide a simple network name and password in order to connect. After discussing this situation with our IT department, they determined that they could modify the wireless routers in the classrooms where we teach ENGR 122 and create a wireless network that only required a network name and password in order to log in. They created this network as a single Class C network, thereby allowing for 254 devices to be connected and assigned unique IP addresses. Based on the enrollment in our ENGR 122 courses in the spring, when the enrollment is at its highest, this would allow more than enough connections as we intend to have the students work in teams of two on this particular exercise.

Software

The software program that is executed on the Arduino microcontroller is shown in the flowchart in Figure 2 below. A complete listing of the code is included at the end of this paper in Appendix A.

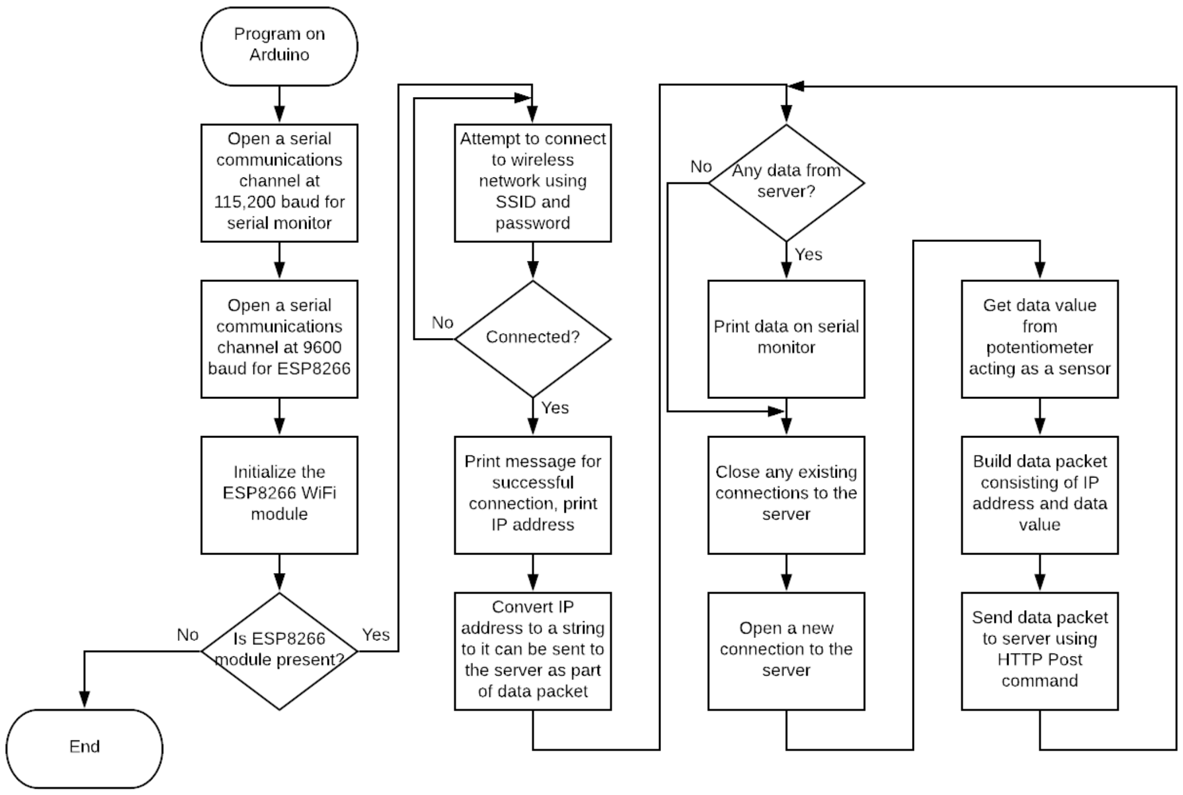
Program on Arduino
Open a serial communications channel at 115,200 baud for serial monitor
Open a serial communications channel at 9600 baud for ESP8266
Initialize the ESP8266 WiFi module
Is ESP8266 module present?
No
Yes
End
Attempt to connect to wireless network using SSID and password
Connected?
No
Yes
Print message for successful connection, print IP address
Convert IP address to a string to it can be sent to the server as part of data packet
Any data from server?
No
Yes
Print data on serial monitor
Close any existing connections to the server
Open a new connection to the server
Get data value from potentiometer acting as a sensor
Build data packet consisting of IP address and data value
Send data packet to server using HTTP Post command

Figure 2. Flowchart of program to be executed on Arduino microcontroller

The program starts by opening a serial communications channel at a baud rate of 115,200 that is to be used to communicate with any serial monitor program. A second serial communications channel is opened at a baud rate of 9600 and this channel is used to communicate with the ESP8266 module. The ESP8266 module is then initialized with this serial channel. The program then checks for the presence of the WiFi module and if it is not found, the program halts. When the module is detected, an attempt is made to connect to the WiFi network using the supplied Service Set Identifier (SSID), which is just the name of the network, and the network password. Once connected, the information about the network is displayed on the serial monitor along with the IP address that is assigned to the module by the Dynamic Host Configuration Protocol (DHCP) server. The students are asked to record this number in their notes as they will need this IP address in order to retrieve their data from the server. The program then converts the IP address assigned to the module to a string so that it can be included in the data packet that will be sent to the server later in the program. At this point all of the setup code (i.e. code that only needs to be executed once) has been executed and the program is now ready to enter into the repeating portion of the code.

In the repeating portion of the code, or the infinite loop, the program first checks to see if any data has been received from the server and if so, that data is printed on the serial monitor. After that, all connections are closed with the server so that a new connection can be made. The data value is obtained from the potentiometer using the "analogRead()" function, and that function will return an integer value between 0 and 1023. A data packet is then created that includes the IP address of this module and the data value just obtained. Text is sent to the ESP8266 module that includes the Hypertext Transfer Protocol (HTTP) POST command, the IP address of the server, the number of bytes in the data packet, instructions to close the connection once the data packet has been received, and the data packet itself. This section of the code repeats until power is removed from the Arduino microcontroller platform.

Server (Database)

To receive the data sent from each of the Arduino/ESP8266 module pairs, a server with a database must be established somewhere on the local area network at a location that allows connectivity between the Arduino/ESP8266 units and the server. In our case, a server was created running a version of Linux from Red Hat, Inc., and a database was established on that server using MySQL. The database was configured to store the data received from each Arduino/ESP8266 unit in separate tables based on the IP address sent from each unit in each data packet. Storing the data in this manner allows the students to later retrieve their data by just knowing the IP address they were assigned in class.

Server (Webserver)

On the same server as the MySQL database, a webserver was also established that pulls the data from a particular table in the database and uses that data to populate graphical objects such as gauges and scrolling line charts. To see the data, students simply open any web browser using the IP address of the server and a web page appears prompting them for the IP address of the data they wish to see. Once that is provided, then the student sees a screen with their data. If their Arduino/ESP8266 module pair is running and sending data to the server at the same time they are looking at the data, then they will see the most recent value on the gauge and a history of their data, including the most recent value, on the scrolling line chart. A screen shot of the web page showing data for a particular Arduino/ESP8266 unit is shown below in Figure 3.

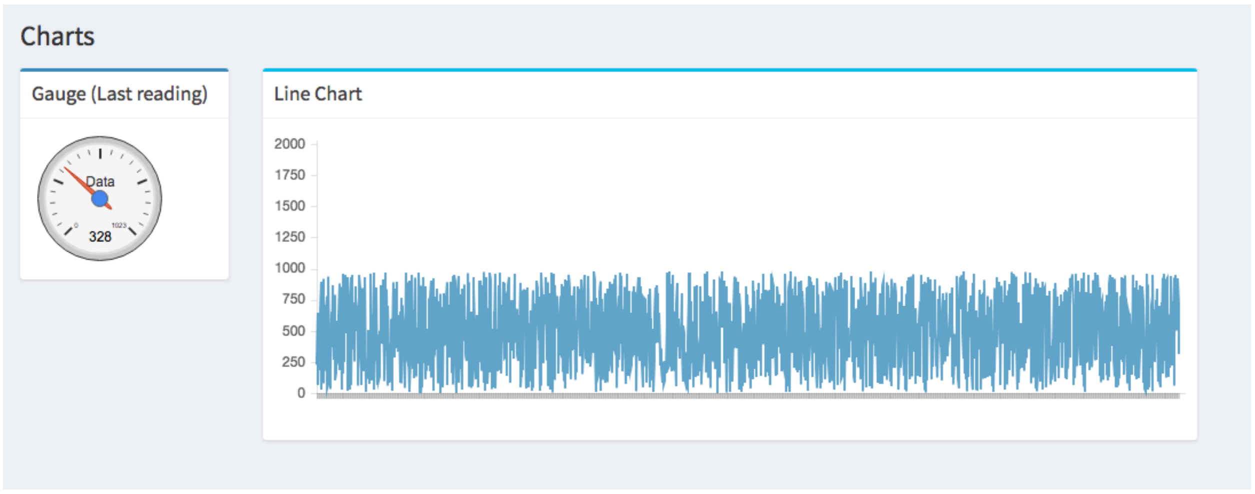

Figure 3. Web page with gauge showing most recent data value and scrolling line chart showing historical data values.

Conclusions and Future Work

To date, the work on this project has included the creation of the server with the database and the webserver, and the simultaneous transmission of data from four Arduino/ESP8266 module pairs. Work continues on fine tuning the database and on expanding the number and type of graphical objects that can be used to display the data from the database. It is our intention to introduce this project into one or more sections of our ENGR 122 courses in the upcoming spring quarter that runs from March 12, 2019 through May 24, 2019. We have identified a place in the curriculum for this course that we believe will be suitable and that occurs just after the midpoint of the quarter. There are several mechanisms that we plan to use to assess the outcomes of this project. Since one of the goals is to increase the student's awareness of the Internet of Things, one mechanism of assessment will be to see if the students are able to identify potential applications of this technology, either in their own student designed product in ENGR 122 or in any of the other products designed by their peers. Another means of assessment will be to see if any of the students in ENGR 122 this spring actually choose to include this technology in their product and send one, or more, data values from their product to the server. Yet another means of assessment will be to wait until these freshmen are seniors and see if any of them incorporate any part of the Internet of Things into their senior design projects.

Appendix A – Source code listing for Arduino microcontroller program

```
#include "WiFiEsp.h"

// Emulate Serial1 on pins 2/3 if not present
#ifndef HAVE_HWSERIAL1
#include "SoftwareSerial.h"
SoftwareSerial Serial1(2, 3);           // RX, TX
#endif

char ssid[] = "ENGR122";                // the network SSID (name)
char pass[] = "ENGR122Arduino";         // the network password

int status = WL_IDLE_STATUS;            // the Wifi radio's status

char server[] = "138.47.102.40";        // the IP address of the server
IPAddress ip;                           // the IP address assigned to this module
String ip_addr;                         // the string value of the IP address
int data_value = 0;                     // the data to send to the server

WiFiEspClient client;                   // Initialize the Ethernet client object

void setup()
{
  Serial.begin(115200);                 // initialize channel for serial monitor
  Serial1.begin(9600);                  // initialize channel for ESP module
  WiFi.init(&Serial1);                  // initialize ESP module

  // check for the presence of the ESP8266 WiFi module
  if (WiFi.status() == WL_NO_SHIELD) {
        Serial.println("WiFi module not present");
        // halt the program if not WiFi module
        while (true);
  }

  // attempt to connect to WiFi network
  while ( status != WL_CONNECTED) {
        Serial.print("Attempting to connect to WPA SSID: ");
        Serial.println(ssid);
        // Connect to WPA/WPA2 network
        status = WiFi.begin(ssid, pass);
  }

  // we connected now, so print out the data about our network
  Serial.println("We are connected to the network!");
  printWifiStatus();
  // Convert the IP address assigned by the DHCP server to a string
  ip_addr = String(ip[0]) + String(".") +\
                String(ip[1]) + String(".") +\
                String(ip[2]) + String(".") +\
                String(ip[3]);

  // attempt a connection, report back if successful
  if (client.connect(server, 80)) {
          Serial.println("Connected to server");
  }
}

void loop()
{
  // if there are incoming bytes available
  // from the server, read them and print them
  while (client.available()) {
          char c = client.read();
          Serial.write(c);
  }

  // Close any existing connections before opening a new connection
  client.stop();

  // connect to the server again
  if (client.connect(server, 80)) {

          // obtain a data value from potentiometer
          data_value = analogRead(0);

          // build data packet consisting of IP address and data value
          String content ="{\"userId\": \"";
          content = content + ip_addr;
          content = content + "\",\"value\": ";
          content = content + String(data_value);
          content = content + "\r\n}";

          // send text to ESP8266 module to post this data to the server
          client.print("POST /api/v1/save HTTP/1.1\r\n");
          client.print("Host: 138.47.102.40\r\n");
          client.print("Content-Length: ");
          client.print(content.length());
          client.print("\r\n");
          client.print("Content-Type: application/json;\r\n");
          client.print("Connection: close\r\n");
```

```
            client.print("\r\n");
            client.print(content);
  }
}

void printWifiStatus()
{
  // print the SSID of the network we are attached to
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print our WiFi module’s IP address
  ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength
  long rssi = WiFi.RSSI();
  Serial.print("Signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}
```