

To: Professor Pisano

From: Jacob Dansey, Changshuo Fu, Christine Low, Urvashi Mohnani, Neil Sanghrajka

Team: 24

Date: 11/20/16

Subject: First Deliverable Test Report



1. **Project Objective**

* 1. The Trivia Throw Toy will be a ball-shaped toy that speaks in a clear voice a trivia fact that has been intelligently parsed from Internet sources upon detecting it’s been thrown or shaken. The user will have the ability to hear facts by category and also up vote/down vote facts to enhance other user’s experience. A mobile application will be used as the interface for picking categories and voting. An accelerometer is used within the toy to detect motion. Upon detection, a fact is pulled from the proper categories cache and spoken aloud through a speaker. If the cache is near empty, a call to a cloud server is made to pull more facts in. The server handles the parsing of Wikipedia and identifies proper facts through natural language processing.

1. **Test Objective and Significance**
   1. **Detecting a Throw or Shake**

To detect a throw or shake, the MPU6050 accelerometer module is used. It has 3-axis acceleration and gyroscope detection which helps detect the shake/throw regardless of the toy’s orientation. This is significant because the toy must be able to differentiate between when it is thrown/shaken, versus when a person is carrying and moving around with it. The objective of the tests regarding the acceleration values is to determine what is the difference in values in the state of being thrown and not thrown while making sure the user does not have to shake the toy violently.

* 1. **Caching a Fact and Reading From Memory**

For the purposes of the first deliverable, the facts are being stored into the arduino’s EEPROM memory for offline fact retrieval. It must also store the next available address on EEPROM for storing a fact and the address of the next fact to be read so when the arduino starts up again, it does not repeat any facts.

* 1. **Using the Text-to-Speech Module**

The fact is parsed to make it audible. The string that is retrieved from the arduino’s EEPROM memory is passed to the emic.speak() function, which then converts it to a form that can be heard from the speaker.

* 1. **Server**

All the trivia facts are stored on a remote database. The servers acts as an interface between the database and the arduino; hence it is necessary to test whether it's configured correctly. It is important to ensure that the server is hosted remotely on AWS, so that the toy(arduino on toy) can retrieve facts at anytime using internet. Additionally, since most of the Natural Language processing will be done on the server, setting it up now lays foundation for future work.

* 1. **Retrieving Fact From Server**

In order to upload new facts to the trivia throw toy, the toy needs to be able to have Wi-Fi capability to hit a server with a request for more data. The Arduino UNO must connect to a Wi-Fi module that can send HTTP GET requests and read back data from a server. This is a test of the ESP8266 connection to the end point of the server and also the retrieval of the proper response.

1. **Equipment and Setup**
   1. **Overall Arrangement**

The hardware of the toy for the first deliverable includes an arduino, an accelerometer, a text-to-speech module, and a speaker. Figure 3.1.1 shows the configuration of the setup with the module on the breadboard being the accelerometer and other module being the text-to-speech module.

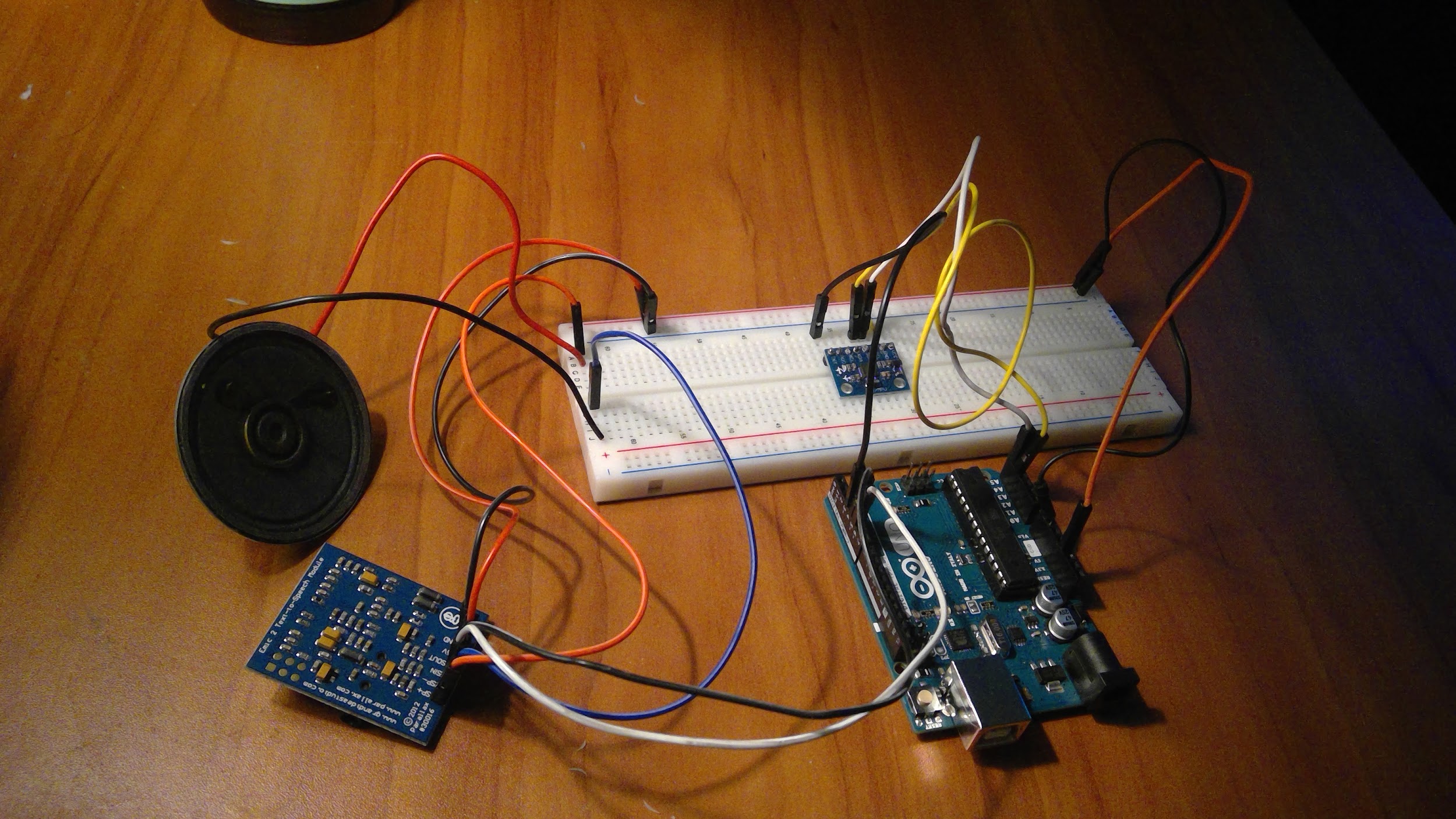
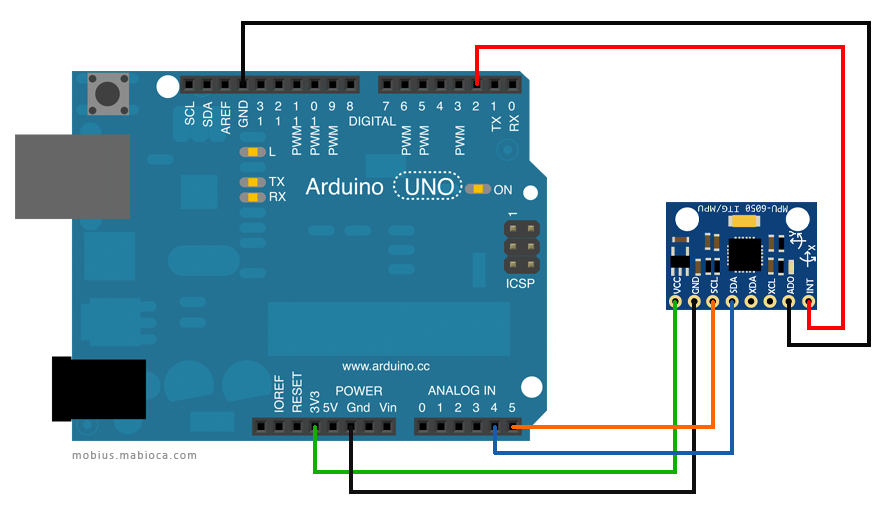


Figure 3.1.1

* 1. **Accelerometer**

The arduino feeds 3.3v to the accelerometer. The MPU6050’s SDA and SCL pins are connected to the arduino’s I2C bus by analog pins A4 and A5, respectively (Figure 3.2.1). This establishes a master-slave relationship with the MPU6050 being a slave to the arduino and allows it to give the raw values for the acceleration measurements of the x,y, and z axis.

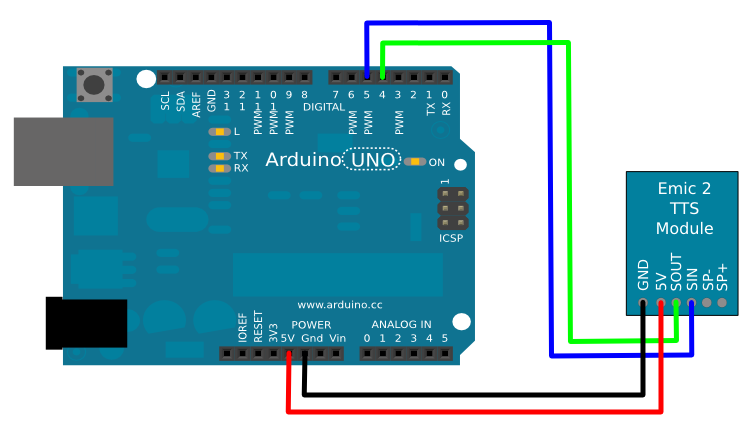
Figure 3.2.1



* 1. **Text-to-Speech and Speaker**

The Text-To-Speech (TTS) module is powered by the arduino’s 3.3V pin. The TTS module’s SOUT pin is connected to pin 4 on the arduino and the SIN pin is connected to pin 5. These pins allow for serial communication to be established between the TTS module and arduino. The speaker is connected to pins SP+ and SP- on the TTS module. The EMIC2 has a library that lets the user change the volume, voice type and pace of the audio.

Figure 3.3.1

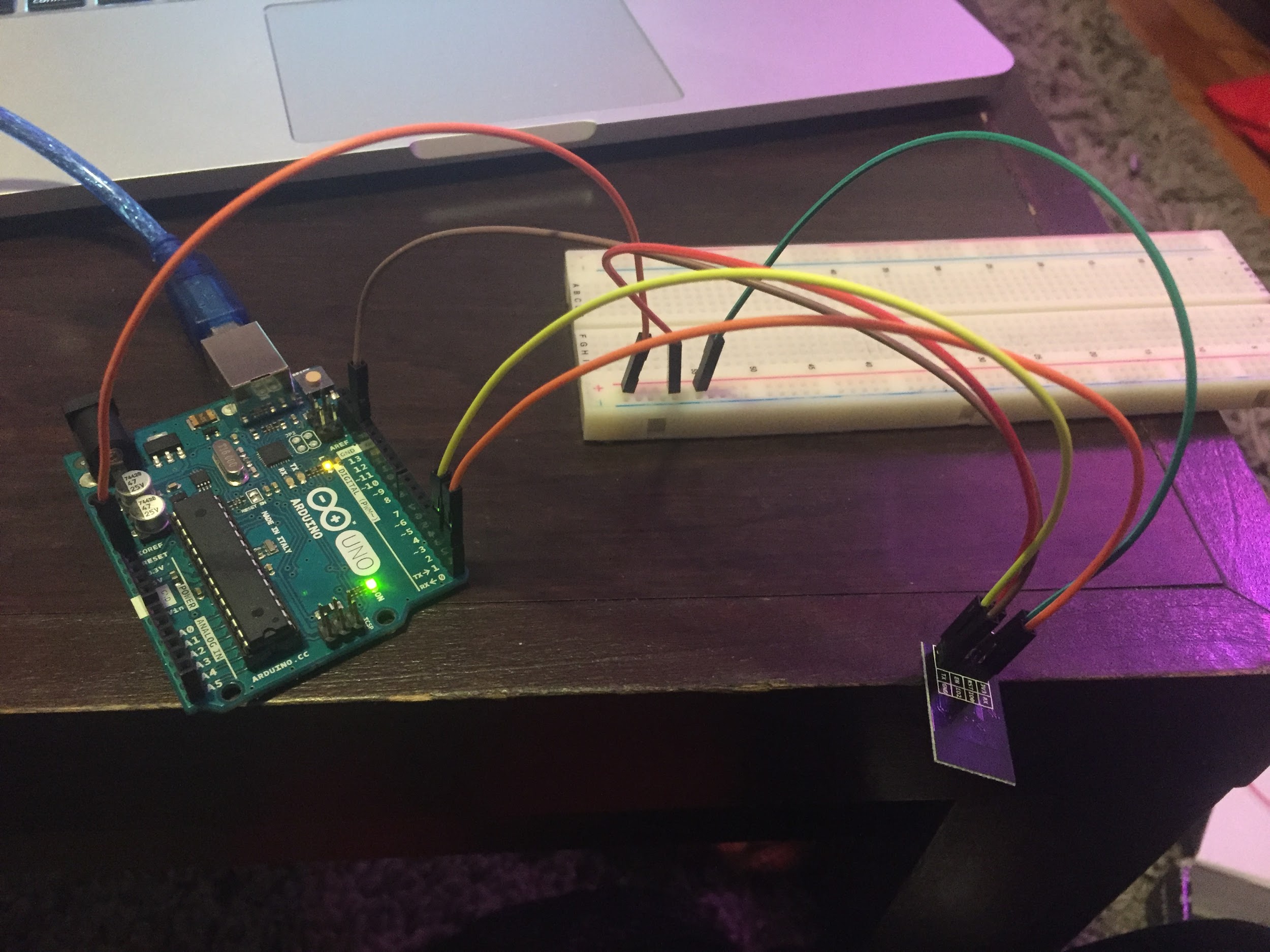


* 1. **Server**

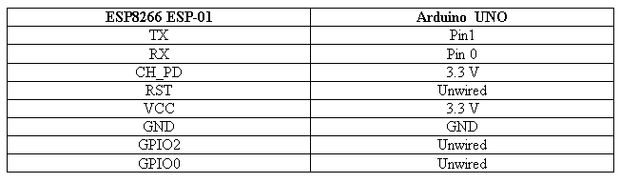
The server infrastructure consists of a PostgreSQL database and code written in Python based Flask micro-framework. This code serves two functions, first it is used to interface with the database, secondly it makes a REST API available to the Arduino for it to be be able to request for facts. All of this is hosted in the Amazon Web Services(AWS) cloud for remote access.

Facts are retrieved from theserver by calling HTTP GET request to the following url: <http://api-env.ptt9xgaruh.us-west-2.elasticbeanstalk.com/><fact id>. Here <fact id> refers to the unique id associated with each fact in the database(also used as primary key). First to test if the API returns facts, copy the link into any modern browser and hit enter. Doing so by default makes a HTTP GET request to the URL. We need to ensure that the fact id is a number greater than 0. Secondly, to test that each fact is associated with a unique id,change the <fact id> parameter in the URL. If the <fact id> is greater than the number of facts, the server must provide the fact associated with id= (<fact id> % no of facts).

* 1. **Wi-Fi Module**

The Wi-Fi module we use in tandem with the Arduino UNO is the ESP8266-01. The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware. 

AT commands are instructions used to control a modem. AT is the abbreviation of ATtention. Every command line starts with “AT” or “at”, hence the name AT commands. These instructions are used to configure the ESP8266 as well as send it commands to perform server or client instructions.



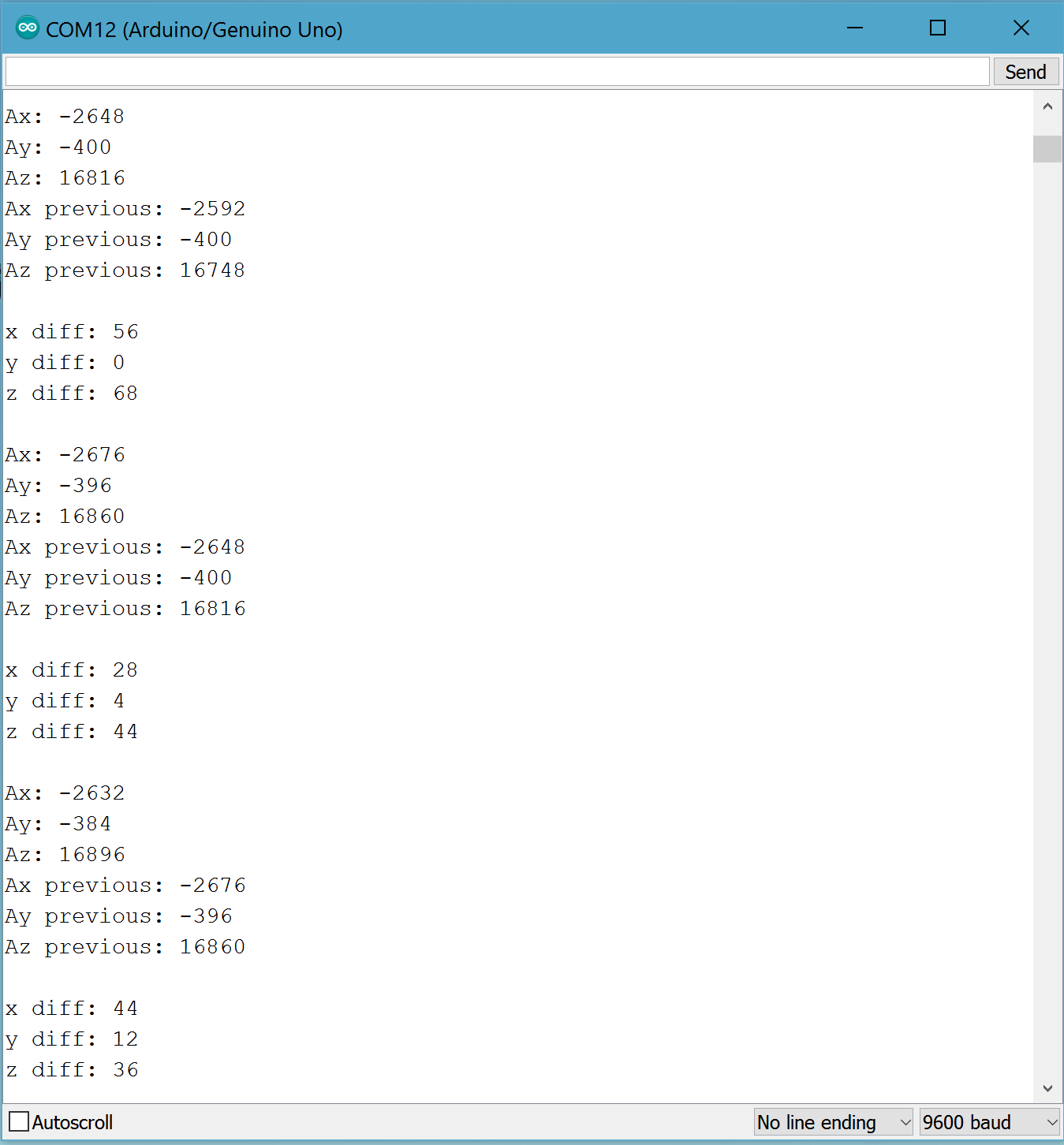
*AT+CWJAP=“ssid”,“password”* is the command to connect the Wi-Fi module to the existing Wi-Fi network. *AT+CIPSTART=“TCP”,“IP address”,port #* is the command to begin a Transmission Control Protocol (TCP) connection with the host server and perform the three-way handshake that is required of TCP. After a TCP connection is started, data can be sent across it using the *AT+CIPSEND= datalength*, followed by the data to be sent.

The data that is sent across the TCP connection must conform to Hypertext Transfer Protocol (HTTP) in order to have the proper data sent back from the server with the 200 status response. HTTP is designed to enable communications between clients and servers.The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data. For our deliverable, the GET request returns the HTTP response with a 200 OK status as well as print the string to the the serial monitor.

1. **Measurements and Data**
   1. **Accelerometer**

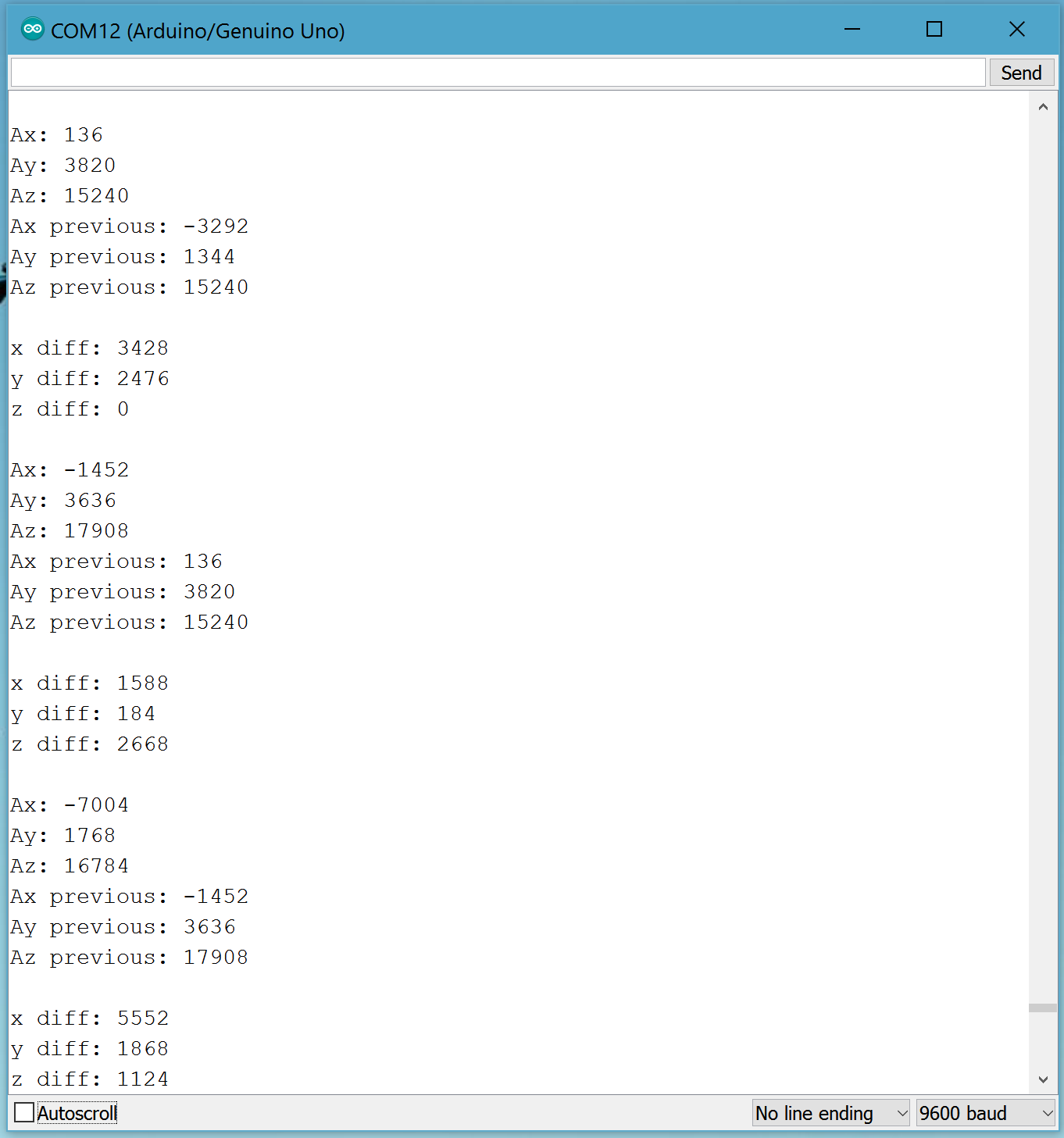
The arduino samples the acceleration values of the three axes every 500ms and compares it to the previous values. In the figure below (Figure 4.1.1), the serial monitor prints out the raw data taken from the MPU6050 and the difference between the current and previous values while the accelerometer is on the table, motionless. The data shows that even though the accelerometer may not be moving, the values still vary, therefore requiring a certain threshold in difference in triggering a detection to prevent it from falsely detecting a resting state as a throw.

Figure 4.1.1



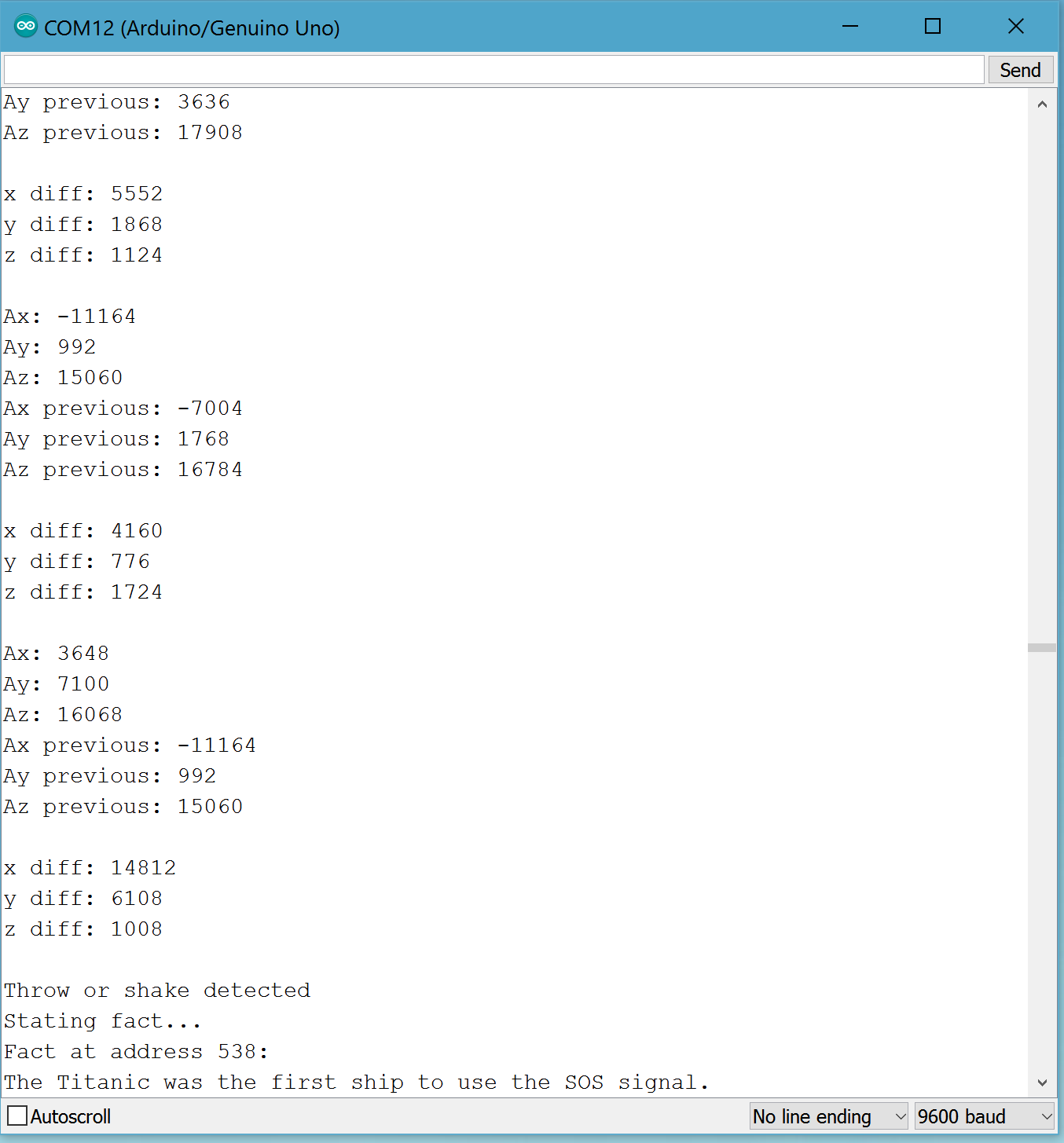
The next testing involved analyzing the difference in values for the accelerometer is moving, such as being carried around, but not being thrown or shaken. The figure below (Figure 4.1.2) shows the difference in the previous and current values being increased, as expected. Repeating this multiple times to get an accurate understanding in the range of values, we were able to increase the threshold of what differences to ignore regarding the accelerometer moving.

Figure 4.1.2



The last testing was to determine the difference in values from going from a resting state to a throwing/shaken state. Since the toy could be shaken/thrown at any orientation, it is important to note that only one of the differences in the previous value of the axis compared to the current needs to be above the threshold. The figure below (Figure 4.1.3) shows the values of going from a resting state to a thrown state, which triggers a fact retrieval. After multiple trials, the threshold value that worked best for detecting a throw while not mistaking it for the accelerometer being moved casually, was a difference of 10000.

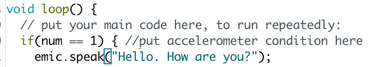
Figure 4.1.3



* 1. **Text-to-Speech and Speaker**

After a fact is retrieved from the arduino’s EEPROM memory, it is passed into the speak() function, which sends it over to the TTS module to be vocalized and heard on the speaker. Initially, the communication between the TTS module and arduino was set up and a random string was passed to the speak() function to test whether it could be heard on the speaker.

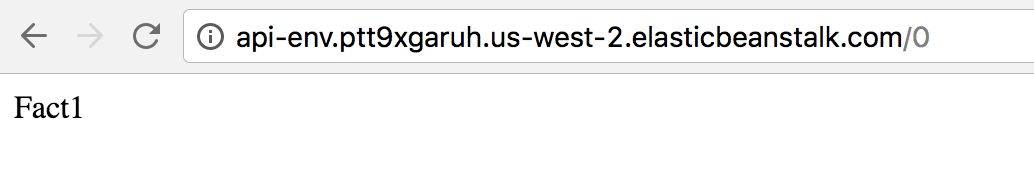
Figure 4.2.1

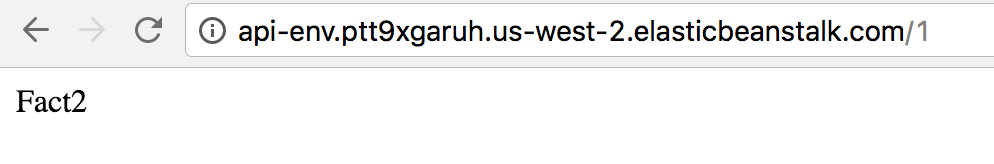


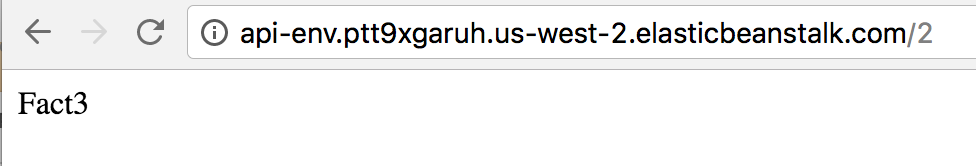
Since the fact has to be retrieved from the arduino’s EEPROM memory, the next test was to see if it could convert a string stored in a variable. The variable was then passed to the speak() function and it was heard on the speaker. There are 9 different options for the type of voice, which vary based on accent, pitch, tone and gender. The volume has a range of -58dB to 18dB and can be changed anytime in the program.

Once this was working, the TTS module was combined with the accelerometer so that when a shake or throw is detected, a fact is retrieved from the EEPROM memory and made audible. The arduino’s serial monitor prints out the acceleration as well as when a fact is being stated due to a change in the acceleration. It also prints out the fact being stated as shown in Figure 4.1.3 above.

* 1. **Server**



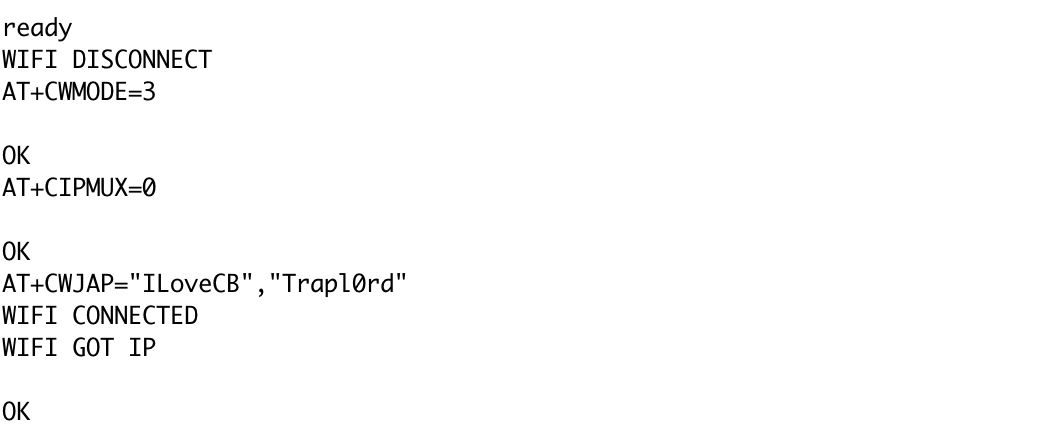




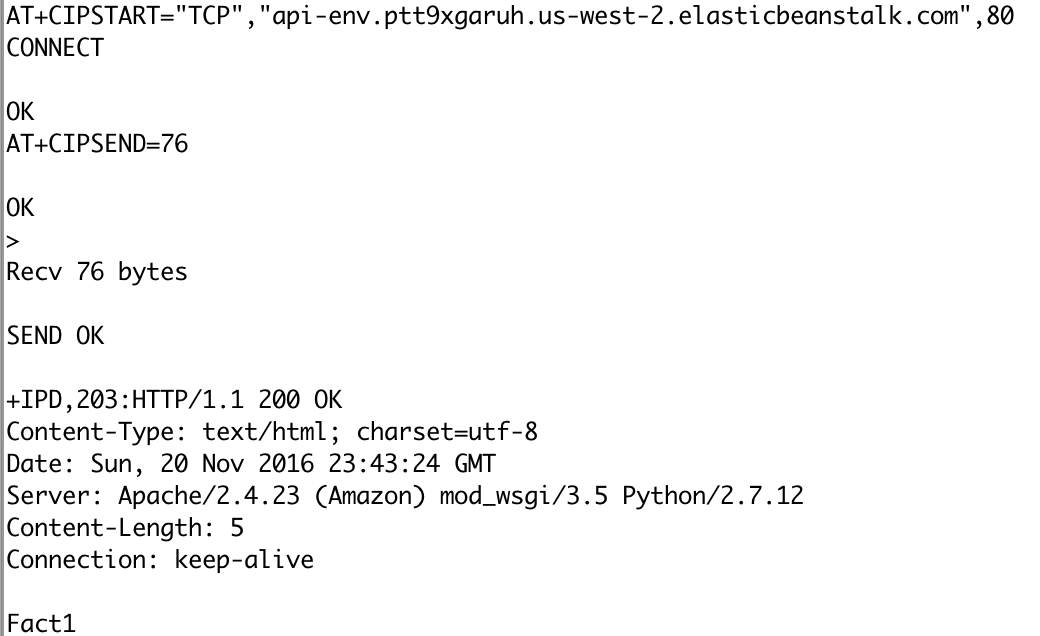
Above is the results of a browser request to the URL of the server. Entering the URL in a browser runs the same operation as a HTTP GET request for /<fact id>. By changing the fact id, the associated fact returned changes accordingly.

* 1. **Wi-Fi Module**

Below is the response from the Serial Monitor of the Arduino IDE.

*AT+CWJAP=“ILoveCB”,“Trapl0rd”* connects the ESP8266 to the Wi-Fi with the given ssid and

password. A TCP connection is then formed with port 80 of the cloud server endpoint “api-env.ptt9xgaruh.us-west-2.elasticbeanstalk.com”. Port 80 is chosen because it is the well known port of HTTP*. AT+CIPSEND=76* prepares the connection for a 76 byte message. Not shown is the HTTP message sent to the server endpoint. The message is:



GET /0 HTTP/1.1

Host: api-env.ptt9xgaruh.us-west-2.elasticbeanstalk.com

The message is received and a response is outputted with the 200 OK status that everything went smoothly. The data *Fact1* is sent back which is the string retrieved from <http://api-env.ptt9xgaruh.us-west-2.elasticbeanstalk.com/0>.

1. **Conclusions**

**5.1 Accelerometer**

After testing multiple trials of throws and shakes, the arduino will consider an acceleration difference of greater than 10000 between each samples to be a throw or shake. This ideal value prevents any false detection that may come from the accelerometer staying still or being moved.

**5.2 Text-to-Speech and Speaker**

In conclusion, the TTS module correctly parses the fact that is retrieved from the EEPROM memory. This fact is then heard clearly and loudly on the speaker. The quality of the speaker determines the clarity and we are researching on the best possible speaker that takes in less power.

**5.3 Server**

A fact is successfully retrieved from the database hosted on the Amazon Web Server. The fact is retrieved by calling a HTTP GET request for the specific url with the fact number after it. Since the server is live, this test works at all times.

**5.4 Wi-Fi Module**

The ESP8266 successfully sends HTTP requests and retrieves responses from the server. The next step is to cache the string in the EEPROM memory to be retrieved from the TTS module.