**Memo**

To: Professor Pisano

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

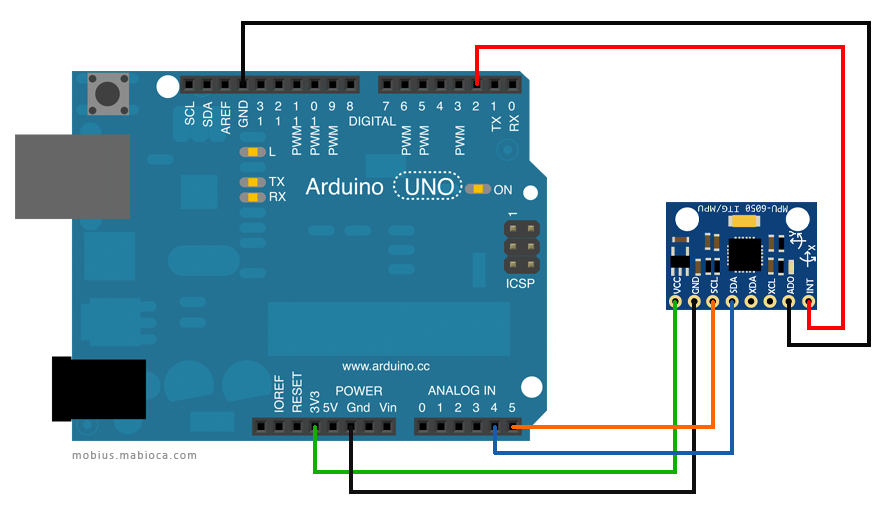
Team: 24

Date: 11/16/16

Subject: First Deliverable Test Plan



1. **Detecting a Throw or Shake**
   1. **Description & Goal**

The purpose of the trivia throw toy is to state a fact when it is shaken or thrown. To do this, we needed to implement a way of detecting a shake or throw by using an accelerometer. Not only did it need to detect these conditions, but it must also ignore scenarios which it may mistake the toy is being thrown, such as when a person is carrying the ball and walking with it. The toy must also ignore any detection of a shake/throw when it is already stating a fact to avoid it from interrupting the current fact with a new one.

* 1. **Procedure**

We are using an Arduino Uno as the microprocessor and the MPU6050 as the accelerometer. The MPU6050 was chosen because it contains a 3-axis acceleration sensor and a 3-axis gyroscope, allowing the detection of a throw or shake regardless of the toy’s orientation. The MPU6050’s SDA and SCL pins are connected to the Arduino’s I2C bus (analog pins 4 and 5 respectively) with the accelerometer acting as a slave.

Using the MPU6050 library provided with Arduino, the Arduino polls the connection of the accelerometer until the connection is successful. Once a connection is established, the input of the x, y, and z axis values of the accelerometer are read every 500 ms and updated in a variable. The difference between the current values and the previous values are computed and if the difference for any one of the axes is greater than the threshold, then a throw/shake is detected. The threshold value was determined by observing how much the three axes values changed in the cases of the accelerometer staying still, moving but not being shaken or thrown, and being shaken/thrown. This required multiple testing to make sure the accelerometer was not too sensitive to movement while not requiring any violent movement to trigger the detection.

When a shake/throw is detected, instead of storing the acceleration values to be compared to for the next measurements, the sampling of the acceleration is delayed for a few seconds until stating the fact has been completed. It then samples the acceleration for a few seconds to ensure there are no drastic differences in values, which basically means that the toy has stopped shaking or had finished its throw. It then resumes with the above mentioned procedure of detecting a shake or throw.

**1.3 Verifiable Results**

This test ensures that only a throw or shake is detected. If the toy is being carried while the user is walking then it should not consider that as a change in acceleration to state a fact. It also makes sure that the toy does not get interrupted if it is shaken while a fact is being said.

**2.0 Caching a Fact and Reading it from Memory**

**2.1 Description & Goal**

Since the trivia throw toy relies on wifi for parsing and obtaining facts, the toy must also cache facts for offline use in situations where there is no wifi. For the purposes of the first deliverable, the facts are being stored in the Arduino’s EEPROM but in the future will be stored in an external memory which allows for more storage size. When a throw or shake has been detected, the Arduino reads from EEPROM the next fact, keeping track of which facts it has already stated.

* 1. **Procedure**

The Arduino uno’s EEPROM first two addresses holds the value of the next available address (in little endian format) in the memory to store a fact. The setup code for the Arduino reads this value and updates a global variable accordingly. When the function cacheFactLocally() is called, a char array of the fact as the parameter, the fact is written to EEPROM byte by byte, incrementing the address variable along the way. At the end of each fact, a null terminator is stored to signify the end of the fact. The value of the next available address then overwrites the current one in EEPROM.

To read a fact from EEPROM, the variable factIndex stores the address in EEPROM of the end of the last fact that was read. It then forms a string by reading byte by byte the characters in next addresses until it reached a null termination character while incrementing factIndex to keep track of the last read address. It then writes the address for the next fact to be played to EEPROM. The fact string is passed to the function playFact() which will play the fact out loud on the speaker. Upon setup, when the Arduino turns on, it reads from EEPROM the address of the next fact to be played so it remembers where it left off at.

**2.3 Verifiable Results**

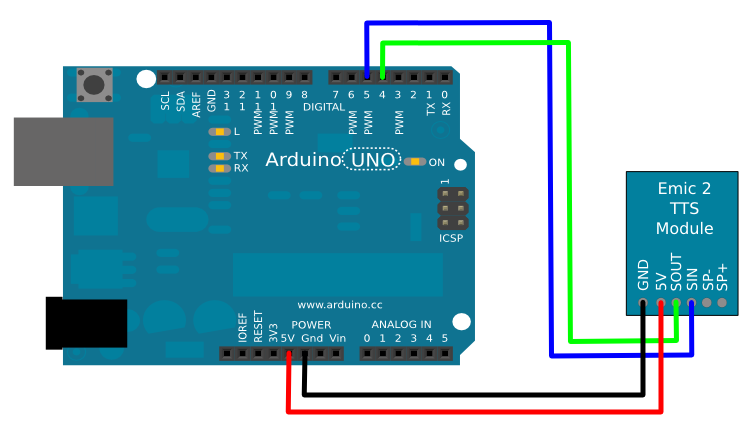
We test the offline fact retrieval process. The toy stores some facts on the device itself for fast fact retrieval as well as fact retrieval when the toy is not connected to the internet. It keeps track of the facts that have been already stated to ensure that the same fact is not heard twice.

**3.0 Using the Text-To-Speech Module**

**3.1 Description & Goal**

Another purpose of the trivia throw toy is to be able to vocalize the facts after fetching it from memory. The volume of the speaker can be adjusted so that the fact can be easily heard in a quiet or crowded area. The facts should be understood easily by humans and should go at an average speaking rate. The user also has the freedom to select the type of voice based on gender and accent.

**3.2 Procedure**

We are using the Arduino UNO as the microprocessor, the EMIC2 text-to-speech module and a speaker to make the facts audible. The EMIC2 is powered by the 5V pin on the arduino and the SOUT and SIN pins on the EMIC2 connects pins 4 and 5 on the arduino which allows for serial communication to take place. The speaker is connected to SP- and SP+ but is not shown on the diagram. The speaker allows for the fact to be heard after the TTS module has parsed it. 

The EMIC2 has a library called EMIC2.h that has functions to set up a connection between the module and the arduino as well as converting a string to its audible version. We first call emic.begin() to set up a serial connection between the arduino and TTS module. Using the setVoice() and setVolume() function we set the voice style and volume of the speaker. There are 9 options for the voice style and the volume can range from -48 dB to 18 dB. The volume can be adjusted using the setVoice() function or using the + and - operators to increment or decrement the volume. When the fact is fetched from the EEPROM, it is passed onto the speak() function and the TTS module then parses it and plays the fact out loud on the speaker. The speak() function is called when the accelerometer detects a change in the acceleration as per the requirements of the toy.

**3.3 Verifiable Results**

This test ensures that the fact is parsed correctly and can be easily heard and understood by the user. It also tests the quality of the speaker and whether the speak() function is called when a throw or shake is detected.

**4.0 Server**

**4.1 Description & Goal**

All the trivia facts are stored on a remote database. The arduino has to be able to access this database in a secure and simple manner, in order to cache trivia to the toy. This is done by setting up a python server, hosted on AWS. The server makes an API available to retrieve facts. Currently, HTTP requests to the API do not require any form of authentication. GET requests take in one parameter: the fact id and returns a string to the arduino. For this deliverable, the database is populated with dummy data.

**4.2 Procedure**

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). The test must run on local and AWS server.

**4.3 Verifiable result**

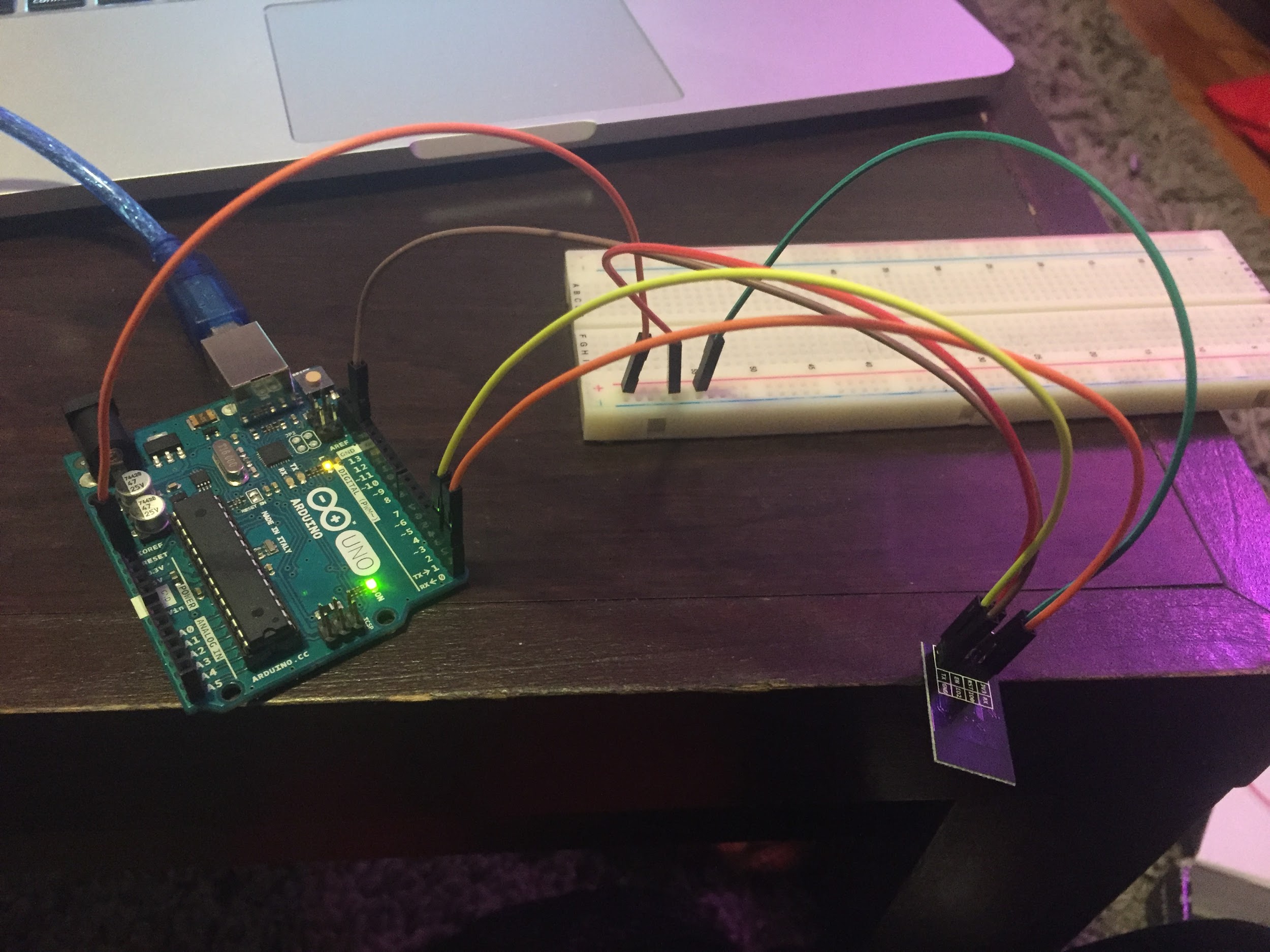
A single string of text appears in the browser window containing a fact associated with that fact id. Changing fact id in the URL returns a new fact. The maximum number of facts currently stored is 5. So entering fact id = 6, must return the fact which has id 1.

**5.0 Retrieving Fact from Server**

**5.1 Description & Goal**

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.

**5.2 Procedure**

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 shall return the HTTP response with a 200 OK status as well as print the string to the the serial monitor.

**5.3 Verifiable Results**

Enter the appropriate AT commands to send an HTTP GET request and read back the appropriate response with data.

**6.0 Battery**

**6.1 Description & Goal**

All hardware within trivia throw toy require lots of charge. In order to make sure every part in the toy works successfully (to detect the motion, the accelerometer must always be on; a high quality speaker is required to ensure outputted sounds are audible and articulate.) a high quality battery is needed. The battery life must last at least 3 hours and support charging from standard 120V and 60Hz AC power output. Also, the size of the battery must be fit in the ball and light enough.

**6.2 Procedure**

For battery life that at least can be used for 3 hours and rechargeable without memory effect, we think cell-phone lithium batteries are the best options. Because these kinds of batteries are thin and light. They can fit well in the ball and won’t make the toy too heavy. Anker AK-70SMI9300-S12P1NA is one of the options. The size of Anker AK-70SMI9300 is 2.4x2x0.2 inches; weight: 2.6 ounces. Small and light enough to fit in the ball. The voltage is 3.8V; capacity is 2200mAh. And built-in IC chip to prevents overcharging. Another thought is using iphone’s battery. This is because the universality of apple equipments use in daily life. Users can use same way to charge the toy just like charging their phones. Scandi tech B011WYNW is a good option. Size is 4.8x3.5x0.8 inches and 3.2 ounces. The voltage is 3.8V and capacity is 1440mAh. This battery is little bigger than Anker AK-70SMI9300. All these two kinds of batteries can provide the power for the toy. Besides, we want to make our toy can be charged wirelessly if we can.

**6.3 Verifiable result**

With fully charged battery, the toy should work at least 3 hours continuously and will shut down when the battery is lower than 5%. The battery should be charged normally when it is connected to the power output.