**Functional Testing Report**

To: Professor Pisano

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

Team: 24

Date: 4/4/2017

Subject: Functional Test Report



|  |  |
| --- | --- |
|  | **Functional Deliverable Test Report** |

**Project Overview**

The Trivia Throw Toy will be a ball-shaped toy that speaks in a clear voice a trivia fact that has been intelligently pulled 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, managing Wi-Fi 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 pulling in of “fun” facts from various Internet sources.

1. **Equipment and Setup**
   1. **“Toy”**

The schematic of the toy is shown in Figures 2.1 and 2.2 below.

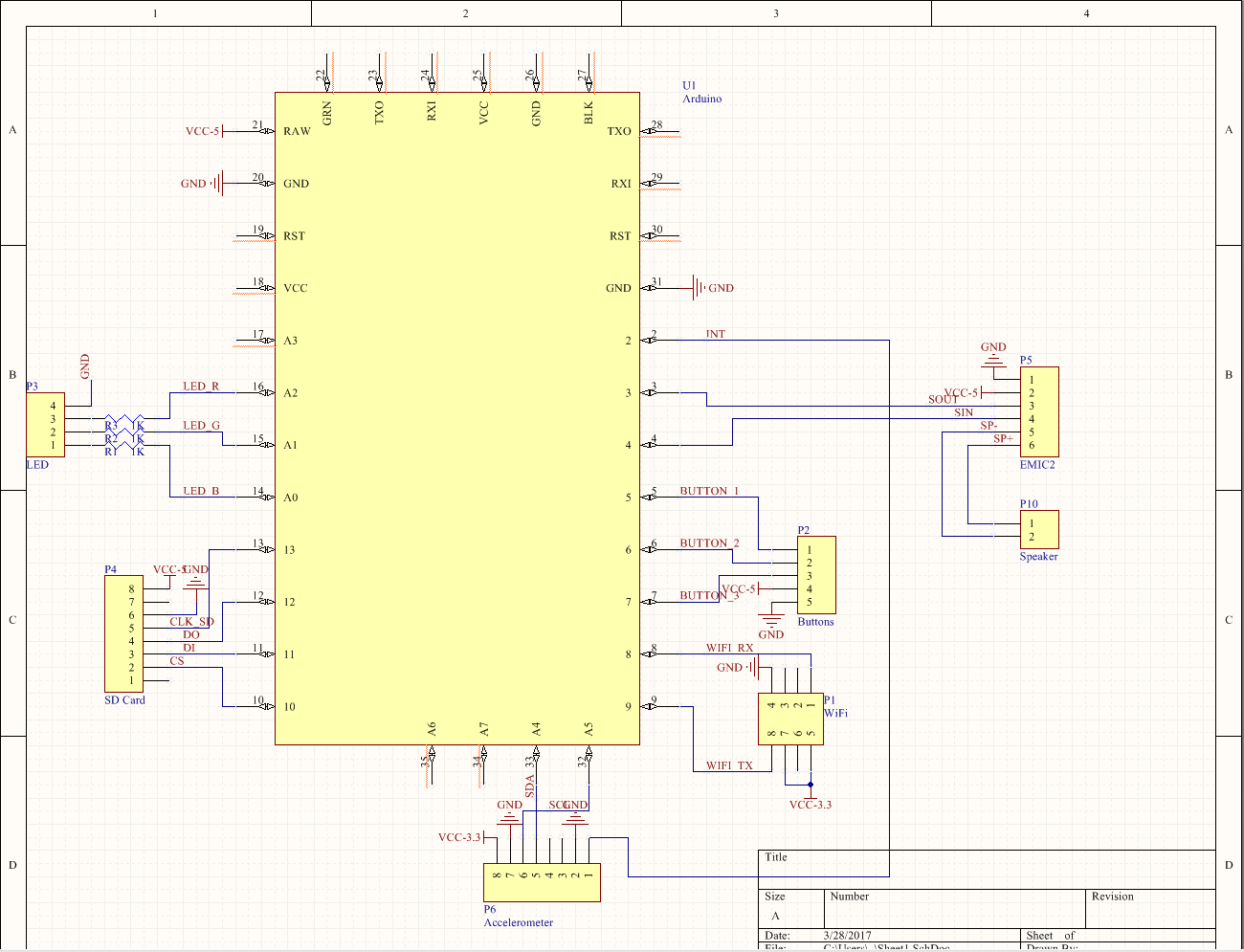


Figure 2.1: Part 1 of the overall schematic

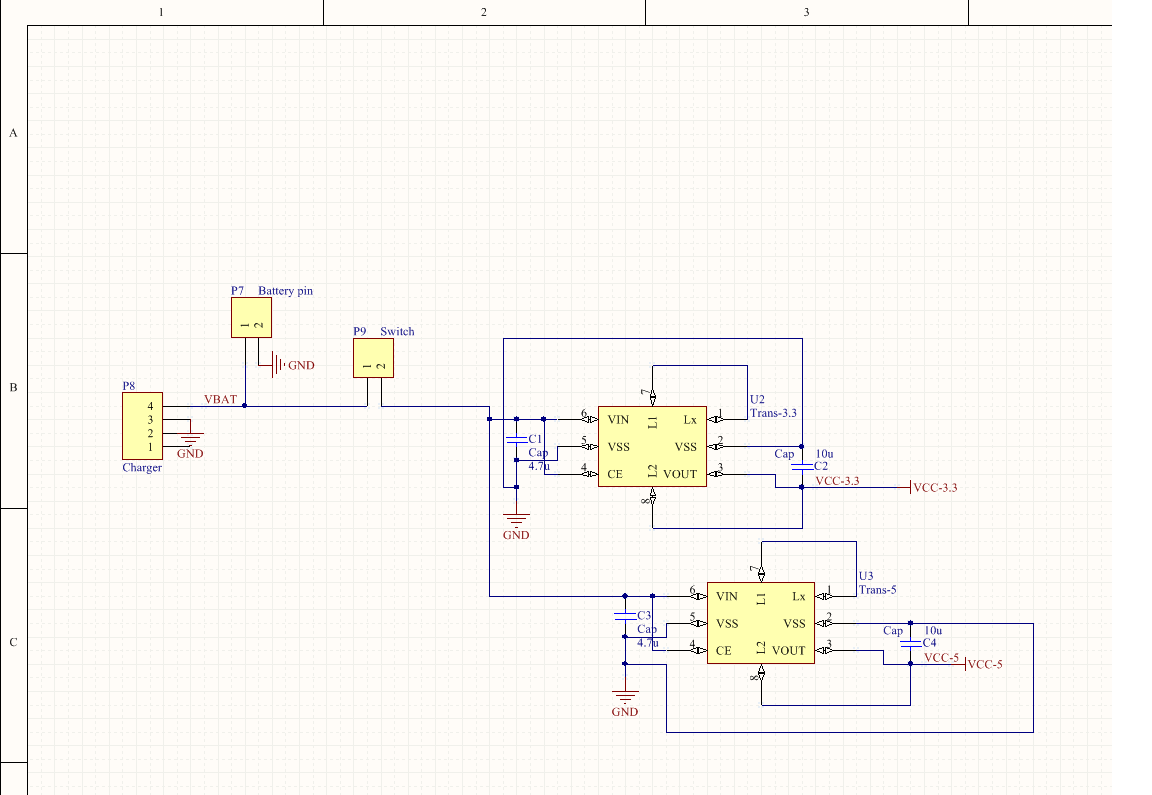


Figure 2.2: Part 2 of the overall schematic

* + 1. **WiFi**

The ESP8266 WiFi module is used with the Arduino for our purposes. The WiFi communicates with the Arduino over UART and requires 3.3V input power. The set up for the WiFi module is shown in Figure 2.1.1 below.

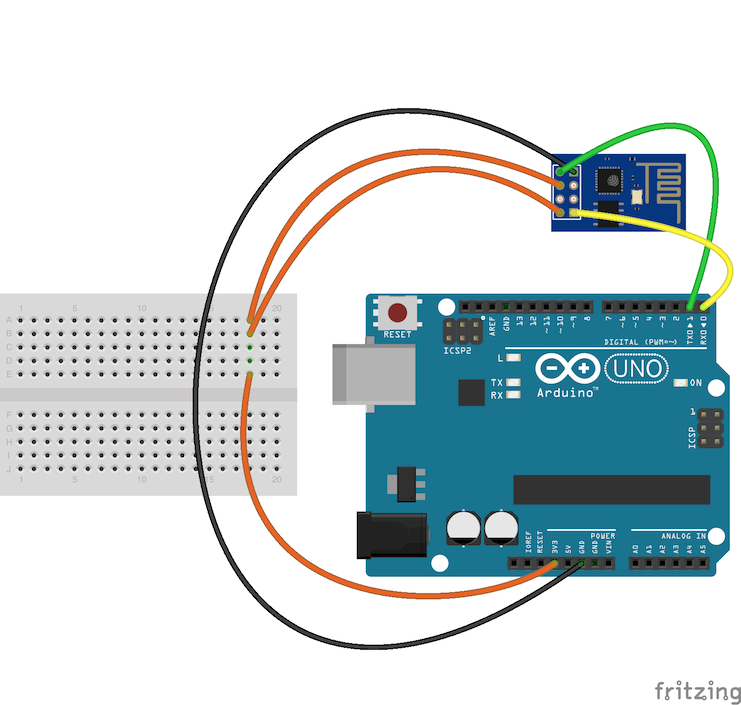


Figure 2.1.1: WiFi module connection with the Arduino

* + 1. **TTS and Speaker**

The TTS module used is called the EMIC2. It is the best string parser compatible with the arduino that offers many features as volume set, voice change etc. The emic2 connects to a speaker for the facts to be vocalized and the speaker used is a 3W and 8 ohms speaker, which gives us a better vocal clarity. The EMIC2 connects to the speaker from SP+ and SP-. The schematic of this connection is shown in Figure 2.1.2 below.

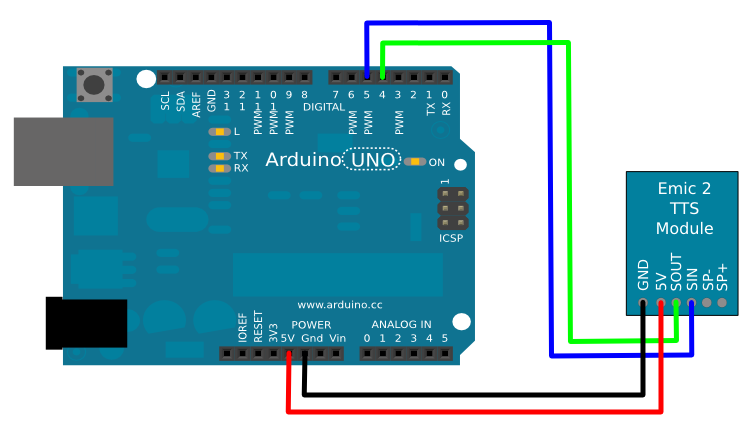


Figure 2.1.2: TTS module and Arduino connection

* + 1. **Accelerometer**

The Accelerometer used is the MPU-6050 with 3-axis detection feature. This feature is essential since the ball can be thrown in any direction by the user to trigger a fact. The accelerometer is set such that only a hard shake or throw triggers the fact vocalization to prevent it from starting up while walking with it.

The schematic for the accelerometer with the Arduino is shown in Figure 2.1.3 below.

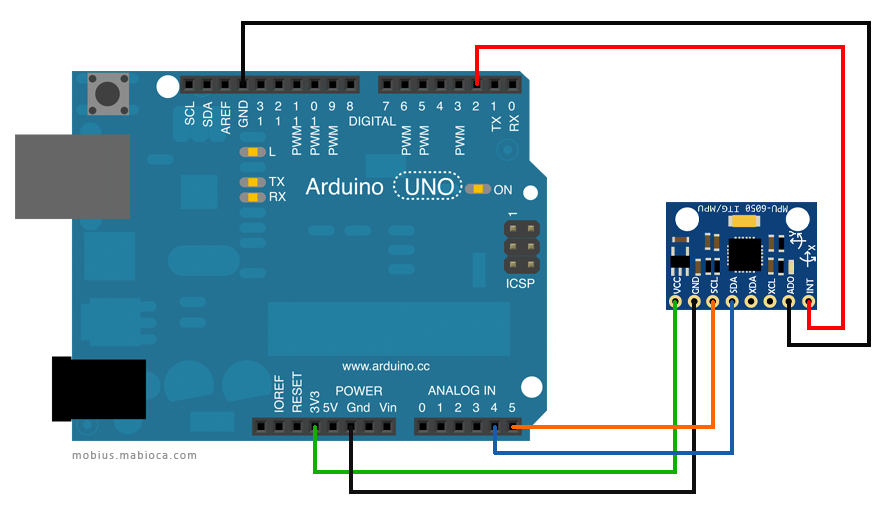
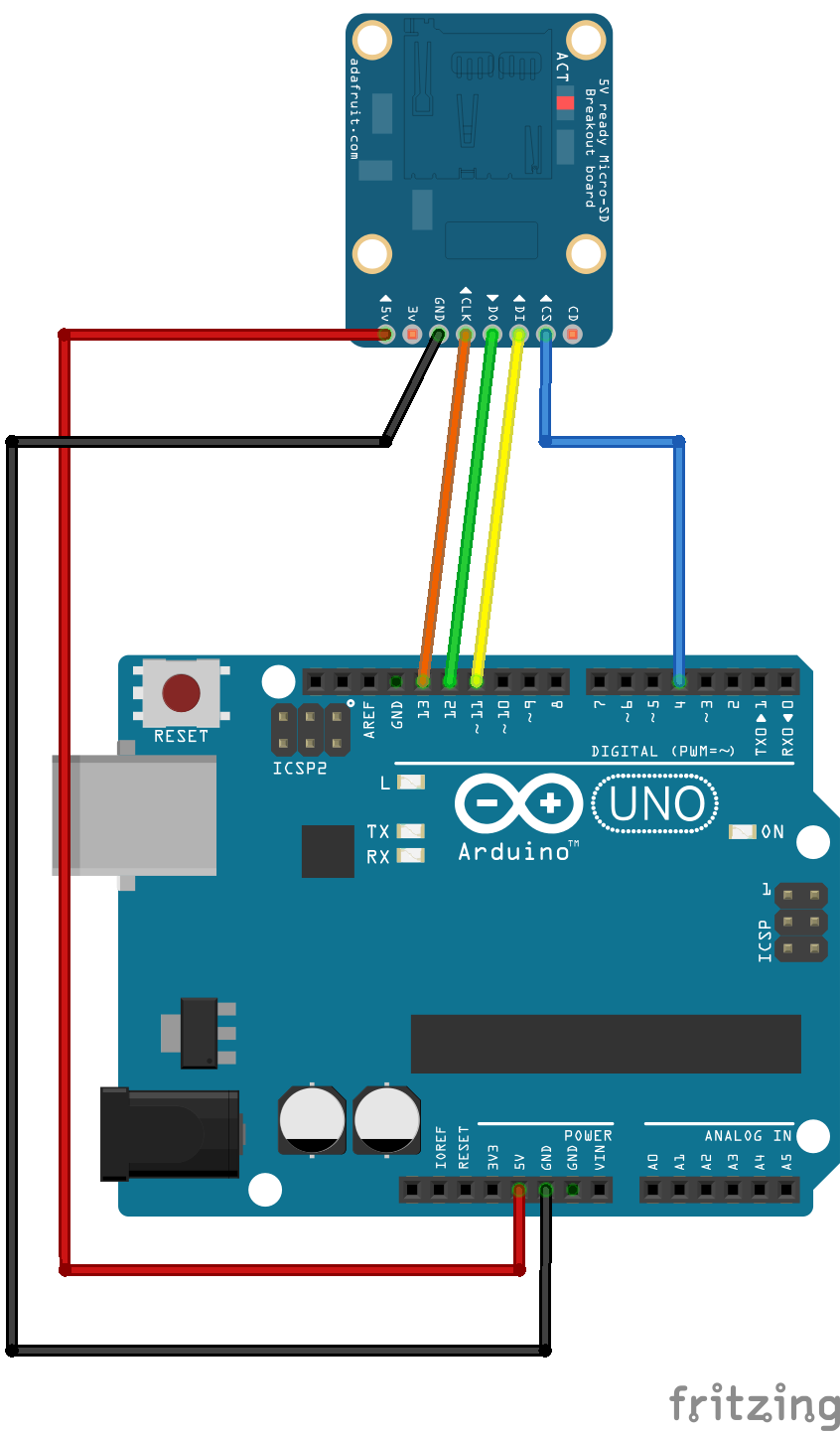


Figure 2.1.3: Schematic of accelerometer and Arduino

* + 1. **SD card**

The SD Card module is used to read and write the facts to the local storage on the toy. This is done so that the fact is vocalized within 0.5seconds of detecting a throw or shake. The toy connects to the server and downloads facts to ensure that it always has 50 facts in local storage. The external storage was necessary since the arduino does not have much EEPROM memory, only about 2kB. The schematic for this part is shown in Figure 2.1.4 below.

Figure 2.1.4: Schematic of SD card and Arduino

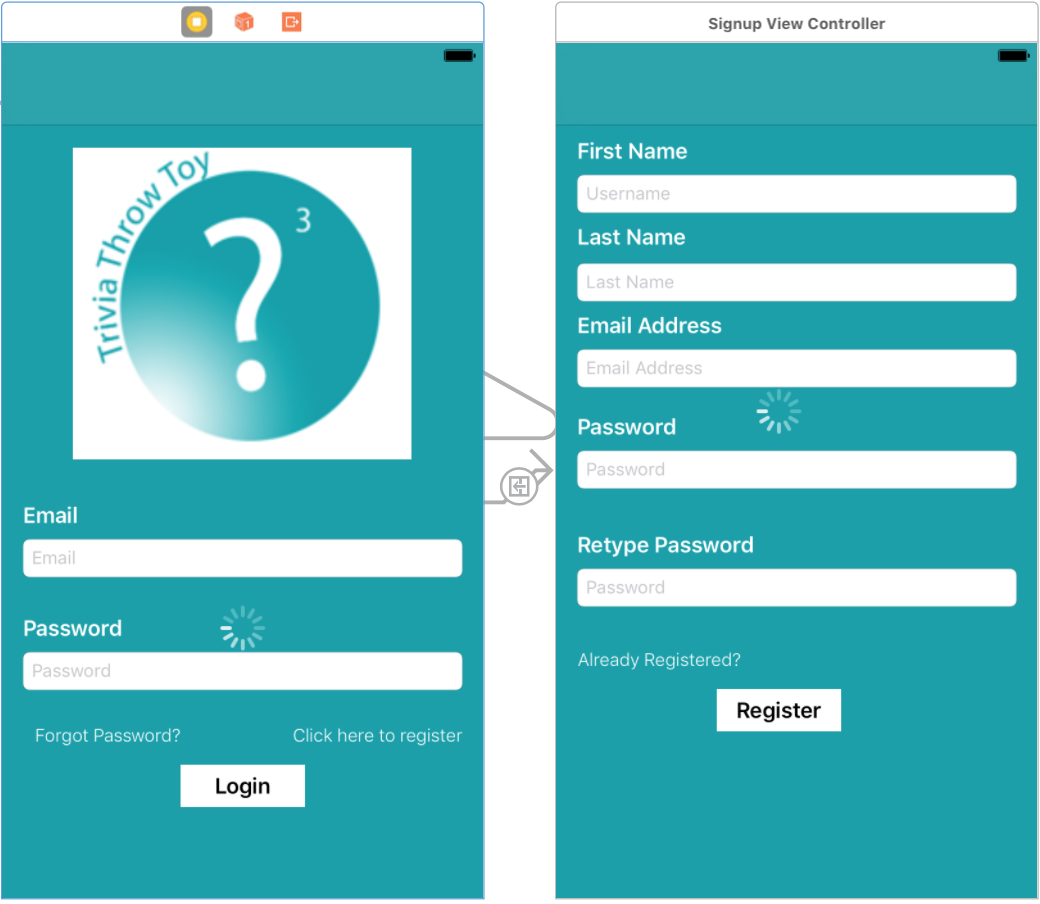
* + 1. **LEDs and Buttons**

There are three buttons in the circuit. One button is volume up, the other is volume down, and the third is to reset the WiFi. There is a switch that creates the connection between the battery and circuit to power on and off the circuit. There is one RGB LED that changes colour based on the status of the toy. For example, red is for low battery, blue is updating from server etc.

* 1. **Smartphone Application**
     1. **Login/Registration**

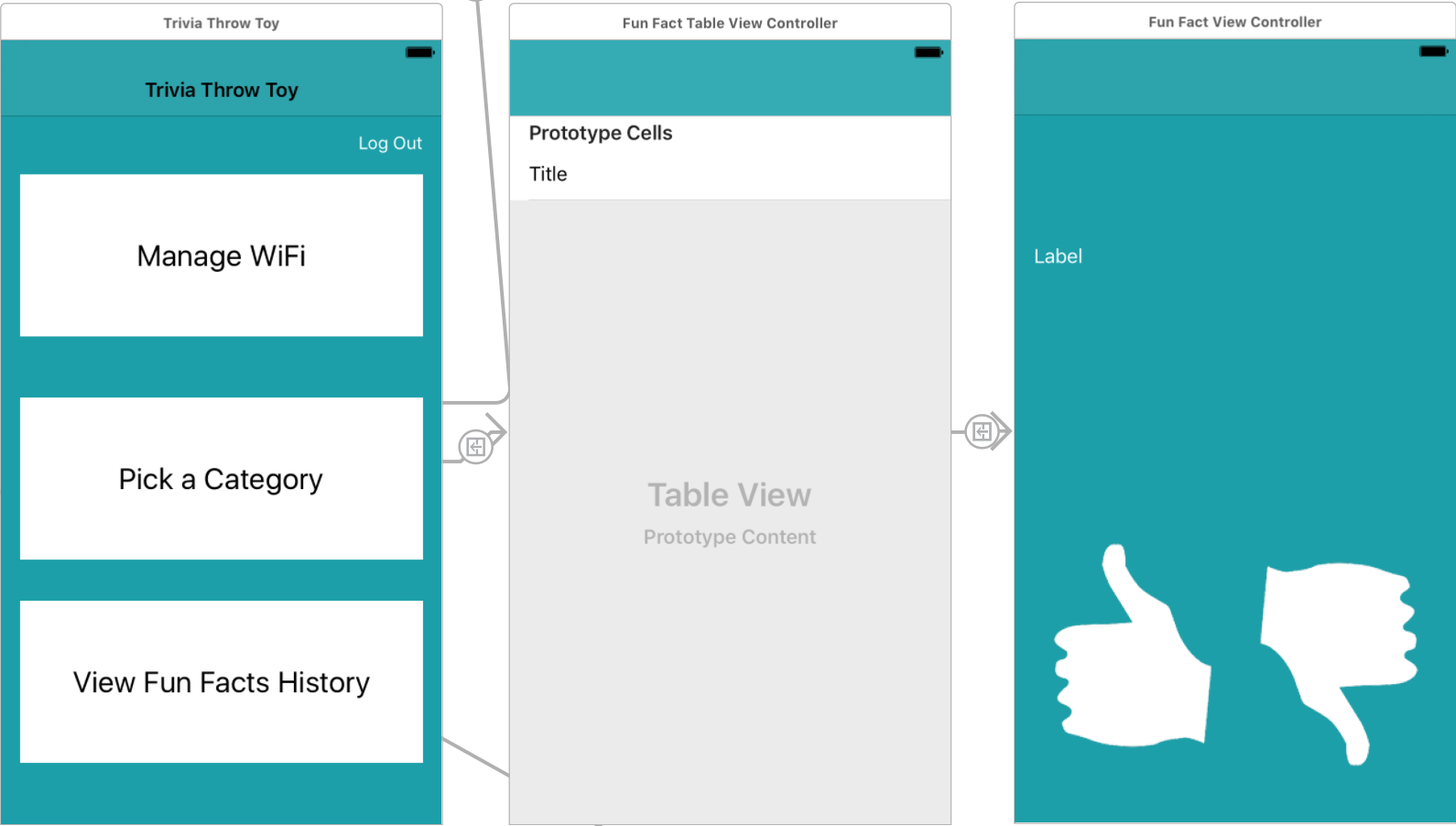
When a user registers with the application, they input their first name, last name, email and password as unique credentials in the *SignupViewController*. This information is then sent as an asynchronous POST request to the endpoint /Users/Register with a unique userID returned in a successful response JSON. The unique userID is stored on the device in the user defaults location with the value “*userID*”, and a session is initiated. When a unique userID is detected on the login screen, the user is automatically logged in to the application.

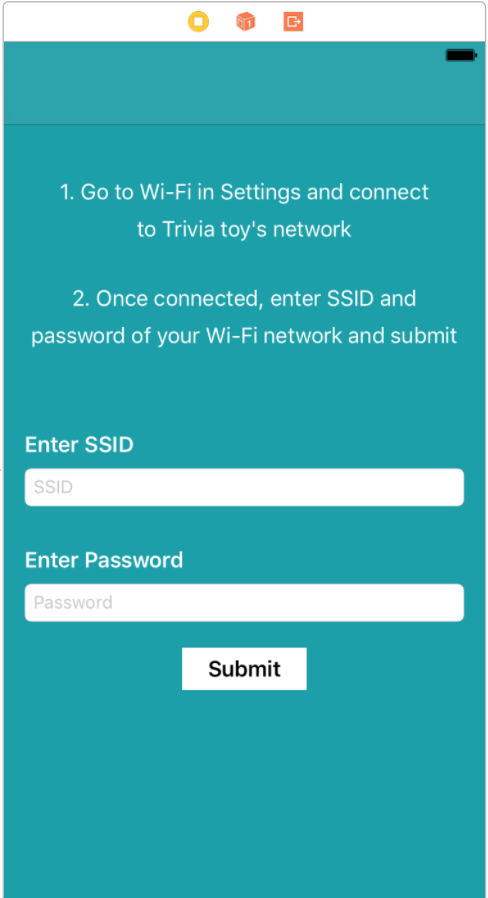
If their is no current session going, the user must login to the application with their email and password. The email and password are sent as as asynchronous POST request to the endpoint /Users/Login with their uniqueID returned in a successful response JSON in the *LoginViewController*. This ID is then stored back into the user defaults “*userID*” value to begin a new session and the user is entered into the application. Within the application the unique ID is used in HTTP requests to identify the proper fact history and category selection. When the logout button is pressed, the user defaults for “*userID*” are cleared to end the session and the user is brought to the main login screen.



* + 1. **Fun Fact History and Voting**

The fact history of a user is displayed in a tableview in the *FunFactTableViewController.* Upon entering the table view controller, the userID of the user is pulled from the default preferences on the device. An asynchronous GET request is then sent to the endpoint User/{userID}/history and a JSON response is returned with the user’s fact history. Once the JSON is returned, the tableview has to reload the displayed data due to the nature of asynchronous calls. The JSON is then parsed and displayed in the proper cell of the table view in reverse chronological order. This is accomplished through the use of the function *tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell*. The index path of a cell is related to its location in the tableview and the return value is the cell of that specific index with the information it is displaying. Once a fact cell is selected from the tableview,the current view is segued to the *FunFactViewController* that displays the fact information as well as the ability to up vote/down vote said fact. Prior to this segue, the *FunFactTableViewController* must pass along the fun fact string and unique ID to the next view.





* + 1. **WiFi Management**

Managing the Wi-Fi connection is handled by the *WiFiViewController*. Upon entering the view, the user is prompted to go to the settings of the phone and connect to the wi-fi network being emitted by the toy. Doing this allows the phone application and the toy to communicate through a socket connection. It checks the connection by attempting to start the socket connection and then receives the toy’s socket emitting message that confirms it is the Trivio toy. After connecting to the proper network, the submit button is enabled and the user is prompted enter the network credentials of their Wi-Fi network. If the toy successfully connects, it will emit a confirmation message that it is connected and the app will display that it was successful.

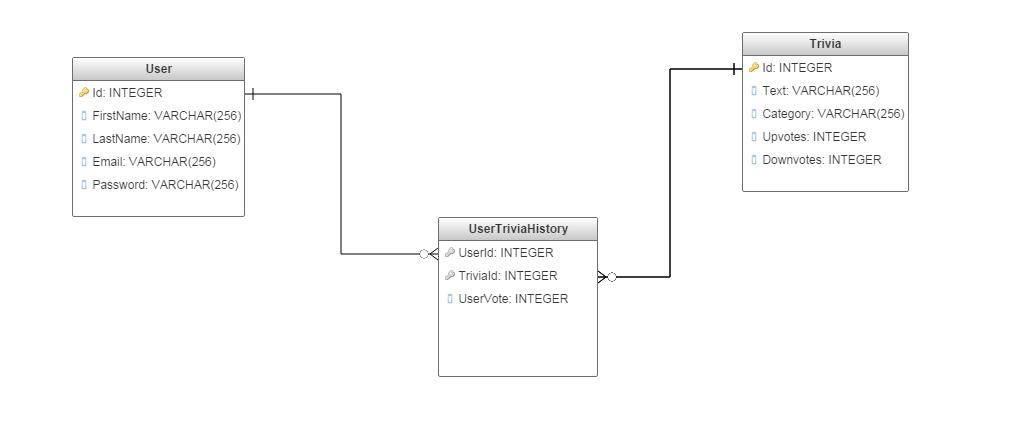
* 1. **Server**
     1. **Database Setup**

The SQL database is hosted on Microsoft Azure and contains 3 tables:

* **User**: Information about all user accounts
* **Trivia**: The collection of trivias, and the number of upvotes/downvotes on that trivia.
* **UserTriviaHistory**: This is a many-many relation table, created by joining the Id from Trivia and User table. It allows us to track the trivia history for a particular user.

To connect to the database using ADO.NET the following connection string is used: *Server=tcp:triviotoyserver.database.windows.net,1433;Initial Catalog=TrivioDb;Persist Security Info=False;User ID={your\_username};Password={your\_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;*

The diagram below shows the Database schema.



* + 1. **Server Setup**

The server is implemented using ASP.NET framework. It performs multiple functions to access the database found primarily in TriviaController.cs and UserController.cs. The server exposes two broad categories of Web API that follow the REST convention. These are:

* User: These endpoints deal with tasks leading to management of user accounts. This API is primarily used by the smartphone application.
* Trivia: These endpoints deal with modifying and accessing the trivia stored in the database.

Detailed API documentation: <https://app.swaggerhub.com/api/neils95/trivio-api/1.0.0>

This table below provides a summary of API endpoints and expected outcome:

|  |  |  |  |
| --- | --- | --- | --- |
| **Endpoint** | **Description** | **Parameters** | **Return values** |
| GET:  Trivia/ | Returns JSON collection of all facts stored in database. | None | Status code : 200   * JSON list of Trivia   Status Code: 400 |
| GET:  Trivia/{UserId} | Returns a single Trivia string for user account linked to {UserId} | {UserId:int} in Path | Status code : 200   * Trivia string   Status Code: 400   * Error string |
| GET:  Users/ | Returns JSON collection of all Users stored in database | None | Status code : 200   * JSON list of users   Status Code: 400 |
| POST:  Users/Register | Creates new user account with unique userid. | JSON in request body {password,username,name} | Status code : 200  - Created user object(JSON)  Status Code: 400   * Preexisting account |
| POST: Users/Login | Log in existing user. | JSON in request body {password,username} | Status code : 200   * User object(JSON)   Status Code: 400   * Invalid Login detail |
| GET: User/History/{UserId} | Return array of TriviaId of a particular user’s Trivia history | {UserId:int} in Path | Status code:200   * Array of TriviaId (int)   Status code:400   * Invalid UserId |

1. **Test Objective and Significance**
   1. **Register/Login users using smartphone app**
2. Register Users: Creating new users is necessary for keeping track of the fact history as well as voting on facts for crowd sourced optimization. User IDs are required for most of the API calls on the toy and ensure that the the user never receives the same fact twice. This User ID is created when a new
3. Login Users: Being able to login as an existing user allows for a full mobile and toy user experience. Logging in to the application allows the user to view all of the fun facts they have heard, vote on the facts, and send their userID to the phone via the wifi management.
   1. **Connect toy to smartphone app**

Once the smartphone is connected to the toy’s AP it is able to transmit data to the toy. Connecting the toy to the smartphone app is important for 2 reasons.

1. Every IoT device needs a way of receiving WiFi credentials in order to access the Internet. However, the problem oftentimes is that the device often needs access to the Internet in order to receive the necessary data to access the Internet. Once connected, the WiFi username and password that the toy should connect to are transmitted to the toy.
2. To enable a user’s fact history, the toy must somehow be aware of which user is using it. Each user account is associated with a unique user Id. Once a user logs in on the smartphone app, this user id is sent to the toy which uses it to download new trivia. Every time the server responds to a GET:/Trivia/{userId} request from the toy, this is added to the user’s history.
   1. **Connect toy to WiFi**

The toy must be able to connect and reset the WiFi in order to make the proper connection to the server. This connection is needed in order to pull new facts in as well as tell the server which facts have been listened to.

* 1. **Speak trivia when thrown**

The main attraction of the Trivia Throw Toy is the ability to throw a fun fact to a friend. This requires the device to be able to detect a throw/shake and almost immediately begin speaking the fact out loud. A fact must be spoken in a timely enough manner so that users are not waiting for a considerable amount of time.

* 1. **Download new trivia on toy**

Without the ability to connect to the Internet, the toy will eventually run out of the facts that are stored on the toy. Our project must be able to pull in a nearly endless supply of new facts for continuous enjoyment.

* 1. **Update/View User’s trivia history**

Every time a user play’s a fact, it automatically shows up in the smartphone application. The fact history functionality of the application is an integral part of the toy experience. If a user was unable to hear a fact on the toy, they can simply check the application to identify what the fact was. The fact history is also the interface that allows users to vote on the facts. If they view their history and particularly like or dislike a fact they can vote accordingly.

* 1. **Vote on Trivia**

Allowing users to vote on trivia helps solve some of the pitfalls of random fact scraping from the Internet. Since there is no one moderating every single fact, crowdsourcing the overall opinion of each fact allows the database to optimize the experience over time.

* 1. **Power**

To ensure reusability of the toy, the battery must be rechargeable using a standard micro usb cable. Additionally, so that the toy can be taken out of home for extended period it must last at least 6 hours on a single charge. Further, the toy must be able to power on and off in order to operate and conserve battery life.

* 1. **Status LED**

The LED status indicators provide outputs for the user to know what is going on with the system. The LED lights indicate connectivity to WiFi, low battery life and retrieval of a fact. These external outputs allow the user to understand issues with the toy and what actions need to take place in order for the toy to operate correctly.

* 1. **Buttons**

1. Volume Button: The volume button is essential to allow for different styles of play and comfort sound levels. When playing outdoors, users will want more volume in order to compensate for the larger playing space and difficulty hearing.
2. Power On/Power Off button: Enables the user to preserve charge.
3. WiFi reset button: Since the toy needs to go into a special AP mode to be able to receive data from the smartphone, this button is important. Pressing this allows the user to change the WiFi network the toy is connected to.
   1. **Outer casing for toy**

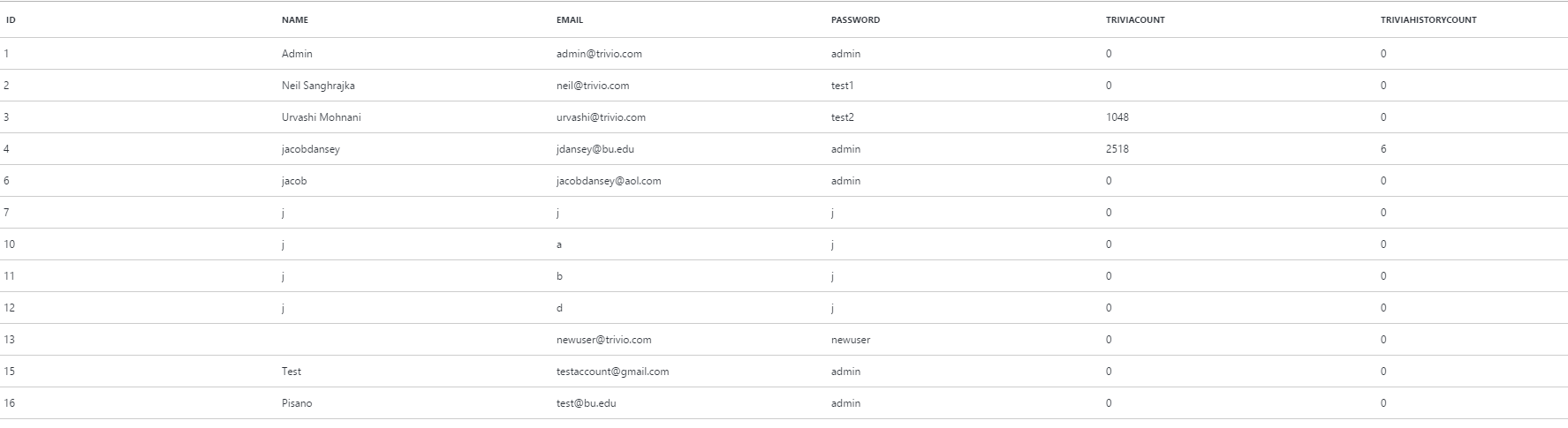
The toy needs a shell casing to hold the PCB in place, as well as the button's, LED, and speaker. This casing forms the basic structure of the toy and all components can be fit accordingly. It prevents components from moving when the toy is thrown or shaken Additionally, it forms the inner layer for our shock absorbent shell.

1. **Measurements and Data**
   1. **Register/Login users using smartphone app**

Registration: After entering name, password and email(unique) on the signup page of the smartphone app and pressing sign up button, a new user account should be created in the online database.







The figure above shows the contents of the DB after the testing, A new user account for Professor Pisano was created with ID=16. Once successfully registered, the options page should load on the smartphone which is depicted under section 2.2.1.

Login: After successfully entering email and password of an account already registered, on the login page of the smartphone app the user fact history page(Under section 2.2.1) shows up. This page must show all the facts heard by the user previously. The unique userID is sent back from the server and stored on the device to allow for immediate logins. The user can also log out of the application to reset the session.

If invalid credentials are entered, an error message shows up “ User account does not exist”.

* 1. **Connect toy to smartphone app**

On pressing the WiFi button, the LED changes color to indicate it is looking for WiFi and the toy becomes an access point.

Connect the smartphone to the toy’s WiFi named “WiFi ES8266”. The phone displays message: “Connected to WiFi ESP8266”. The phone app successfully starts a TCP server and the WiFi module successfully creates a TCP connection with it. The acknowledgment messages are transferred between the two successfully.

In the smartphone app, under Manage WiFi, a list of all the WiFi networks available at that location is available. After clicking the WiFi network that the user wants the toy to connect to, the credentials are transferred to the toy. The toy is then able to break the string received and try connecting the toy to the internet. On a success, a “yes” message is successfully sent and on a failure, a “no” message is sent. These messages are received by the phone app and used to notify the user the status of the WiFi connection.

* 1. **Connect toy to WiFi**

After a successful transfer of credentials as mentioned in Section 3.2. On successful connection to WiFI the toy led color should change. Also when the toy is shaken, if it can make an HTTP request, it will download trivia from the server and say it out loud. This is also visible in the arduino serial monitor. For testing purposes, we used a smartphone’s hotspot which allows us to view when a device is connected to it. The moment the toy connects to WiFi successfully, the iphone displays a message “1 device connected to your personal hotspot”.

* 1. **Speak trivia when thrown**

The toy successfully detects the acceleration of a throw or shake. The arduino serial monitor prints a message “Throw detected.” The next fact from the SD card, (which can be viewed by opening it on a computer and reading its contents) fed into the text to speech module. The speaker then audibly verbalized the fact within half a second of the detected shake. Throwing toy while it is speaking a fact, does nothing as throw detection is deactivated during speech.

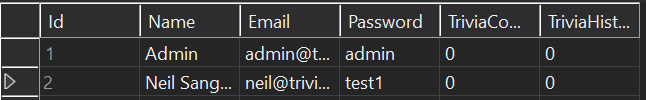
* 1. **Download new trivia on toy**

Once a fact is played from the SD card, the arduino makes a get request to the server for a new fact. This can be viewed in the serial : “GET:/triviotoy.azurewebsites.com/Trivia/4”.

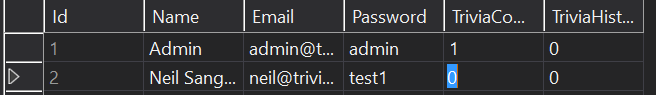
Additionally, this fact is also written to the SD card and opening the SD card on a pc allows us to view this. The new downloaded fact, is the corresponding next fact in the database(which can be viewed using SQL server object explorer in Visual Studios).

* 1. **Update/View User’s trivia history**

After a trivia is played on the toy, the trivia history count for that user is updated on the server. For instance consider user 1: Admin



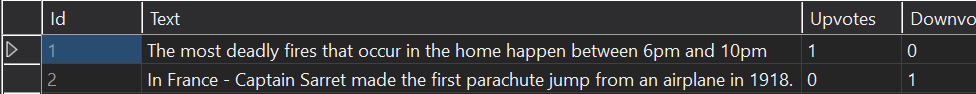
If the toy is setup using Admin’s account, his current TriviaCount is 0. Implying the user has heard no facts yet. After playing trivia, the toy makes an HTTP request and the TriviaCount is updated in the Users table of the DB:



This change is also visible in the smartphone app in the User History page as shown in section 2.2.1.

* 1. **Vote on Trivia**

Once a trivia is present in the User’s Trivia history, they can upvote/downvote it using the smartphone app. This action is visible in two tables in the database. Firstly, the trivia table. After clicking upvote for the first fact, the trivia upvote column is incremented.



Secondly, in the user trivia history table. When admin(user id=1) votes on trivia with (id =1), the user vote column in the database is updated to +1. (-1 represents downvote).



Finally, if a user changes their vote, the appropriate changes are visible in both tables of the database.

* 1. **Power**

Charging the toy: After charging the battery, every part in the Trivia throw toy can work functionally and whole system can work continuously at least 3 hours. When the battery is fully charged, green LED will light up to indicate user the toy is charged and stop charging. While charging, it is red. A multimeter is used to show no current flowing to the battery once it is fully charged.

Power On: Once the power switch is turned on, all components on the arduino are on. This can be tested by viewing the power led on each component. The toy functions as desired. The converse is true for power down.

Battery Level: When the battery level is below 15% the LED color changes to red. First the battery was drained by connecting 4 motors to it, and then it’s battery level is calculated using a multimeter. When it fall’s below 15%, the arduino code changes the LED to red.

* 1. **Status LED**

The LED status is initially set upon startup of the toy with the function setColor() which has a switch statement to change the output of the RGB pins, taking into consideration that a common cathode LED is being used. The toy’s LED status can initially be started as either low battery, connected to wifi, or no wifi connection, with low battery taking priority in displaying its status. When a throw/shake is detected and the toy is connected to wifi, it will call setColor() to display the state of “updating from server”. When the request is complete, it will call setColor again and change it to one of the initial 3 states mentioned above again. When the button for wifi configuration mode is pressed, the function setColor() is called to change the LED status to that mode and when it exits configuration mode, the color is set to one of the 3 initial states again.

The different colours that correspond to the different statuses are shown in the table below.

|  |  |
| --- | --- |
| **Colour** | **Status** |
| Red | Battery is <= 15% |
| Green | Connected to the Internet (WiFi) |
| Purple | Not connected to the Internet (WiFi) |
| Blue | Updating from the server |
| White | In WiFi setup mode |

* 1. **Buttons**

The code for checking if a button has been pressed and carrying out its function is called depending on the status of the toy. The three buttons will only be detected when the toy is in the state of checking the acceleration values. This prevents the toy from adjusting its volume or entering wifi configuration mode while it is playing a fact, making fact requests to the server, or already in wifi configuration mode.

* 1. **Outer casing for toy**

The dimension of the PCB was 7.2 x 12.8 cm so in designing a tray to hold the PCB in place in the toy, I gave the tray the dimensions with +0.5cm tolerance so the PCB can fit snugly in. The buttons were 0.5 x 0.5 cm so square holes were made for it with 0.1cm tolerance. The LED has a diameter of 0.5 cm so a hole was created for that with a 0.1cm tolerance. The usb charging port for the batter was 0.5 x 0.25 cm so a rectangular hole was created for that. The speaker has a diameter of 4.5 cm so a circle was created for it with the same dimensions with tiny 0.25cm holes in a circular pattern to allow the sound to be heard.

Figure 3.11.1 Top half design of the shell with speaker holes

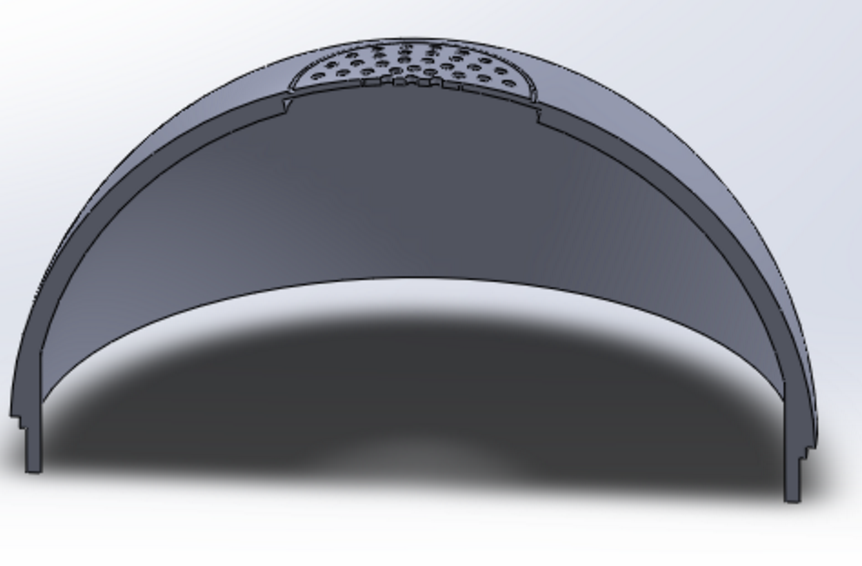
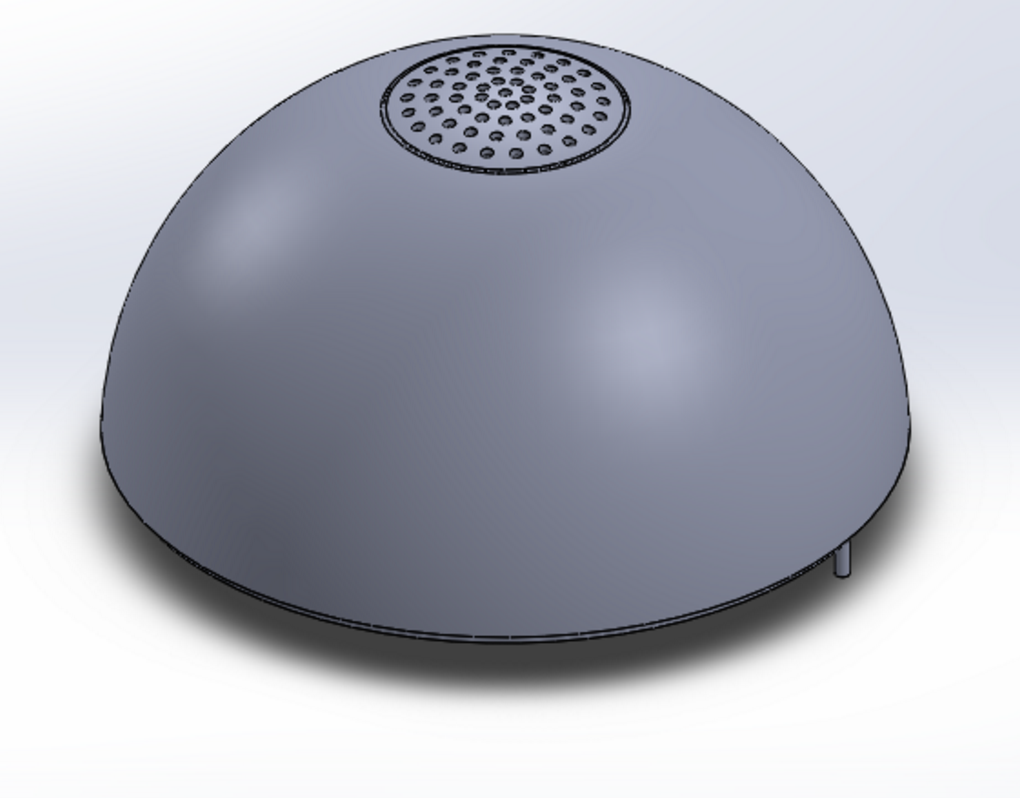


Figure 3.11.2 Bottom half design of the shell with button, switch, led holes and a holder for the PCB

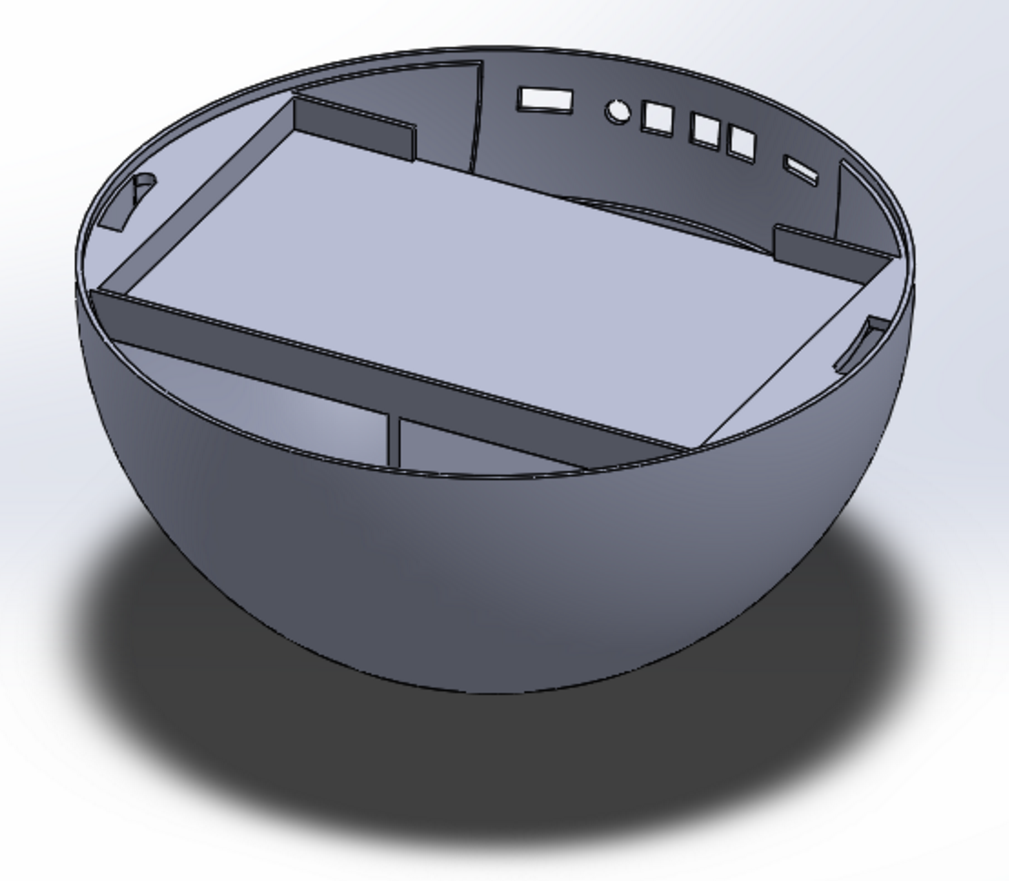
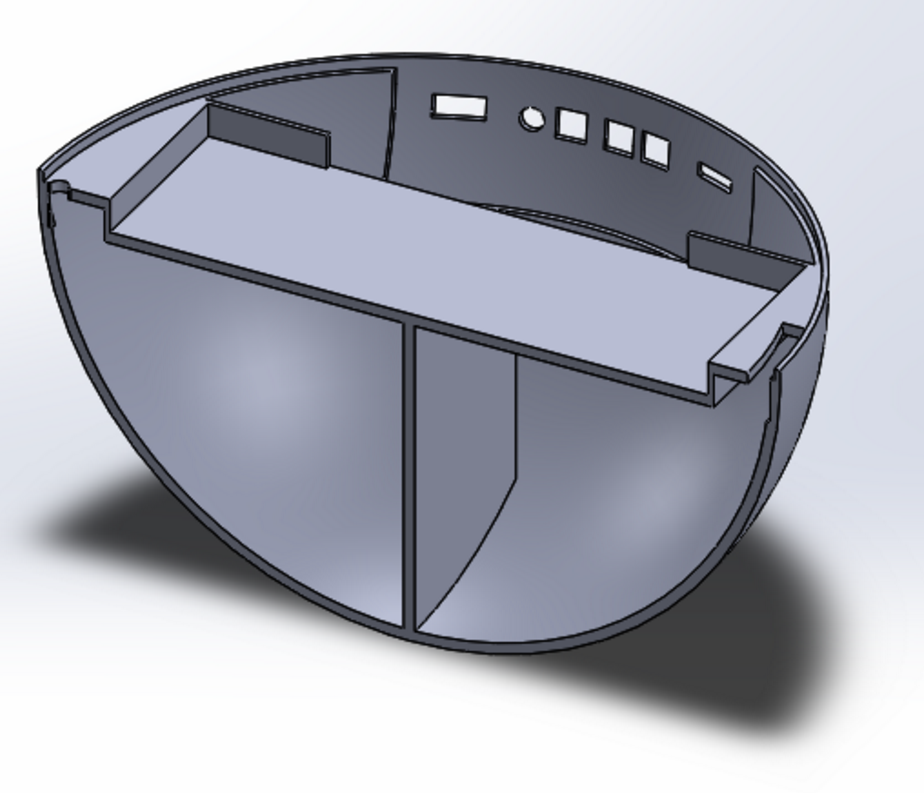
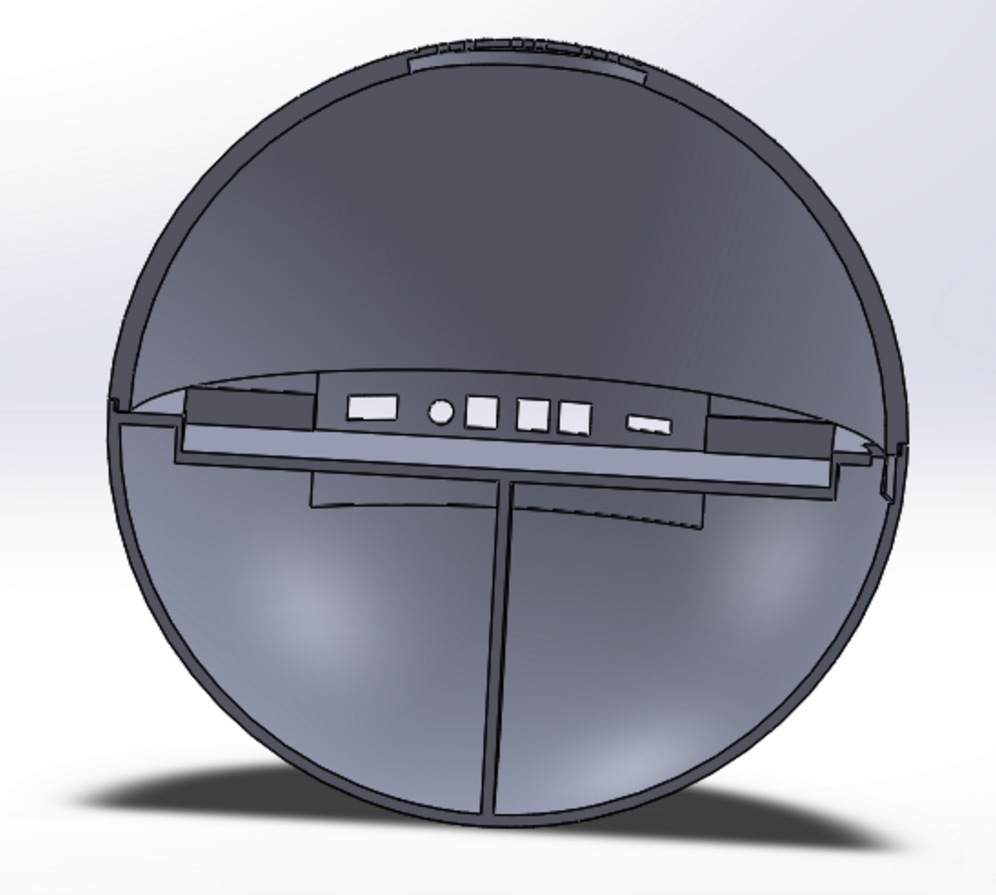
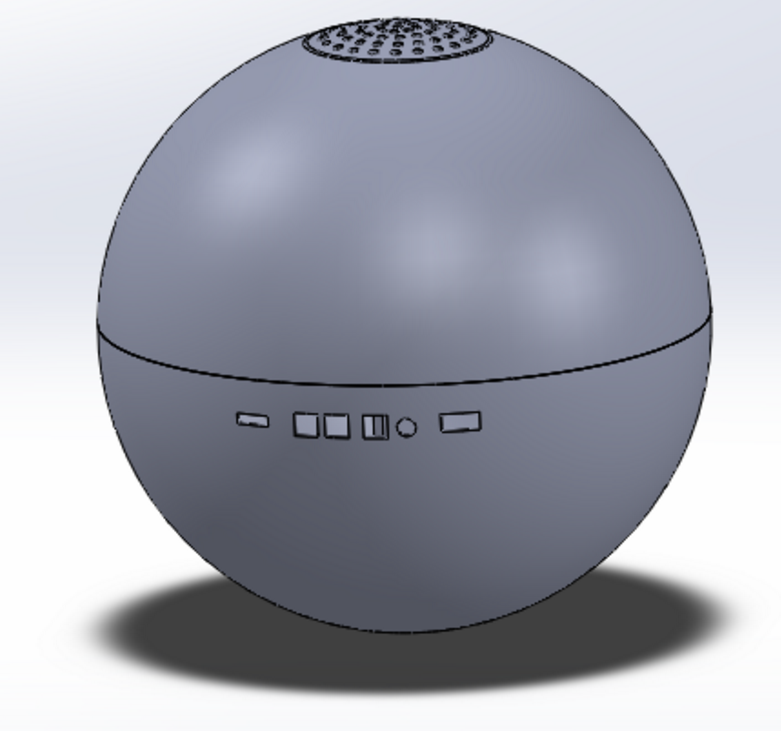


Figure 3.11.3 Design of the shell with both halves fitted together



1. **Conclusion**
   1. **Register/Login users using smartphone app**
      1. **Completed**

In our testing phase, we were able to successfully register and login users accounts using the smartphone application. Consequently, we were able to view a user’s trivia history and allow them to upvote/downvote on the application. User’s cannot log in with invalid credentials, and registration must be completed with a unique email id, hence preventing conflict between multiple users.

* + 1. **Pending**

During testing we found two issues with our account creation process. Firstly, which email id’s had to be unique, there was no code in place to validate whether the string entered is actually an email id. So creating an account with the email id set to “j” would be valid. We have corrected this on the server side, and making an account with an invalid email returns an error message. Secondly, the passwords are currently stored in the database unhashed. While this is fine for testing purposes, before ECE day we would like to encrypt them for better security.

* 1. **Connect toy to smartphone app**
     1. **Completed**

The smartphone app is successfully able to create a socket server which the toy can connect to if it is in wifi configuration mode and the phone is connected the the toy’s access point. The app is able to scan for available wifi networks and allow the user to pick which one to connect to, enter in the password, and send it over to the toy. The toy is able to send a confirmation over the socket server of whether the connection was successful or not for the phone to be notified of the status. Depending on the result, the application displays to the user if the toy is connected to wifi or not.

* 1. **Connect toy to WiFi** 
     1. **Completed**

The toy is able to successfully connect to any smartphone’s hotspot and other wireless networks except for 802.11x. This enables the toy to work with full functionality like download new trivia and update a user’s trivia history.

* + 1. **Pending**

We were unable to configure the toy’s WiFi module to connect to BU 802.11x WiFi network as it uses multiple credentials. While this is beyond the scope of our project, if time permits we would like to be able to make our toy compatible with different types of WiFi network.

* 1. **Speak trivia when thrown**
     1. **Completed**

Initially the toy would open a txt file from the SD card and write the contents of the file to a string to be passed to the EMIC text-to-speech module. However, upon further testing of the code, we were able to use the EMIC’s own built in function to speak out a fact directly from the SD card when specified a filename. This worked successfully and we were able to reduce the memory the arduino used as well.

* + 1. **Pending**

During testing, Professor Pisano told us that he was not satisfied with the volume of the toy. Because, the room was noisy during the testing, he was unable to clearly understand what was being said. Our PDRR mentions that the toy must be audible and have a range of up to 80Db. Anticipating that ECE day will be equally noisy, we are working on improving the audio quality of the toy.

* 1. **Download new trivia on toy**
     1. **Completed**

The toy successfully makes HTTP requests to the server to download facts. This is viewed in the serial monitor.

* + 1. **Pending**

During our demo, we had WiFi timeout issues when combining the code with all other modules. We are working on fixing this bug.

* 1. **Update/View User’s trivia history**
     1. **Completed**

The server code supports API calls for both updating a user’s trivia history and downloading all the history. The smartphone application currently is able to display all trivia heard by a user. For testing purposes, we made changes to the DB manually to add trivia to a user’s account.

* + 1. **Pending**

During testing, we were unable to demo automatic updating of trivia history when a fact is played. Since then, we have added code to the arduino which makes an HTTP PUT request to User/History/{UserId} and the change is visible in the database.

* 1. **Vote on Trivia**
     1. **Completed**

The phone application is successfully able to make a PUT request to the server when a thumbs up/thumbs down button is pressed in the GUI. Storing the user’s vote on the server allows the mobile app to fetch the vote status of the fact the next time the app is open to prevent a user from voting on a fact more than once or change his or her vote.

* 1. **Power**
     1. **Completed**

The Battery provides voltage in the range of 3.7V to 4.2V. A DC step up booster is used to step the voltage up to about 9V. This is then fed into the VIN pin of the arduino and all the other modules are powered using the arduino’s 3.3V and 5V pins as the arduino already has these voltage regulators on board. A switch is connected in series with the battery and voltage booster, which allows the toy to be powered on and off.

The battery is also connected to the A0 pin on the arduino for battery voltage measurement. The A0 pin reads in the amount of voltage being produced by the battery and calculates the percentage left based on the minimum and maximum voltage produced by the battery. When the battery falls below 15%, the LED becomes red informing the user that it needs to be charged.

* 1. **Status LED**
     1. **Completed**

The LED is able to display the appropriate status colors when needed.

* 1. **Buttons**
     1. **Completed**

The buttons are able to detect the presses when we want it to and trigger the corresponding event designated to it. It also includes code for debouncing the button to prevent any incorrect detection.

* 1. **Outer casing for toy**
     1. **Completed:**

For our first PCB, we successfully 3D printed a sphere with a speaker vent and holes for button/LED. Additionally we have the shock absorbing material, EVA foam, required to coat the shell and were able to successfully mold it into the shape of the sphere.

* + 1. **Pending**

In our first prototype shell, we found issues with the the locking mechanism between the two halves of the shell. Additionally, since we will be using a newer PCB for our final project, we have also changed the dimensions of the shell. We will incorporate both these changes in our next 3D printed shell. We also have to mold the EVA foam for the new shell and cut out appropriate holes in it for the speaker, buttons, charging port, and LED indicator. We are currently waiting for our new PCB to arrive so we can solder on the modules.