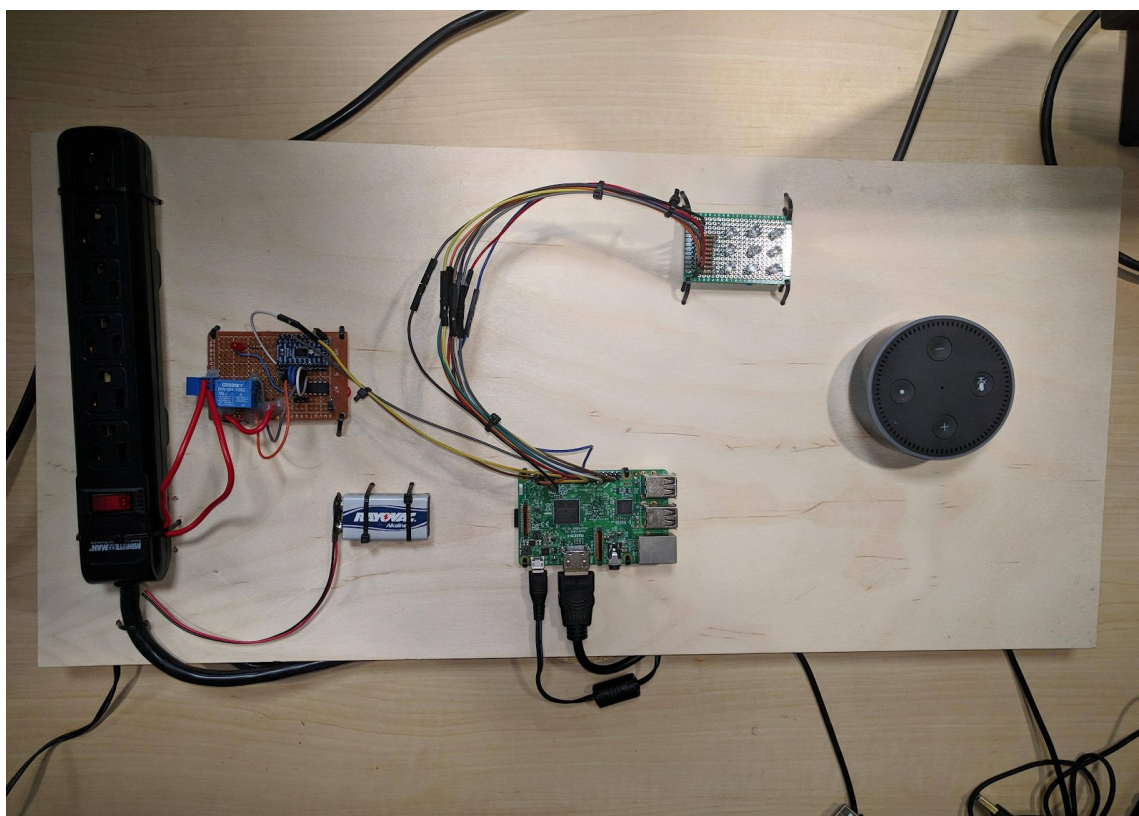




Pi Whisperer's Speech Recognition Control System



Developers: Samuel Chora, Mike Ranis, Victor Wu, Kenil Vora

Last Updated: 12/13/17



REQUIRED DOWNLOADS / INSTALLATION

1. <https://github.com/kvora97/DIYAlexa.git>

A NOTE FROM THE DEVELOPERS

From the developers, we want to give a genuine thank you to the ECE 196 course for fostering a community of innovation. Welcome to the Pi Whisperer project, which explores the potential of voice controlled hardware through an Amazon Alexa. In this project, students will learn how to program in several languages, and implement software into hardware through the Raspberry GPIO pinouts. Good luck on optimizing different objects to respond with Alexa, improving this documentation, and making the project personally yours.

As always, have fun!

-Pi Whisperers



PROJECT DESCRIPTION

Our project is a Custom Speech Controlled System that can turn devices on or off as well as control various modes on those devices. This project was made so that one can expand upon it to control devices around the home.

PROJECT ORIGINALITY

Created a DIY Alexa on a raspberry Pi and configured it to control various modes on an LED matrix we created controlled by a separate Raspberry Pi. Also created an android application which controls LED through a bluetooth connection. The originality of this project stems from the fact that we found very little projects online that set out to have voice controlled hardware in the manner in which we achieved during this project. Voice recognition activated hardware is not a new concept that has never been done before, but we are very proud of the uniqueness in the manner in which went about doing it.

OVERALL LEARNING OBJECTIVES

- Programming w/ Raspberry Pi in terminal window, Python, and Java
- Configuring Amazon Alexa on Raspberry Pi
- Local Host Tunneling for Transmission Protocol (NGROK)
- Setting up Alexa Voice Control with Raspberry Pi
- Setting up an array of LEDs that can be independently controlled by the GPIO pins of the Raspberry Pi



REQUIRED PROJECT PARTS (ALL)

Product Name	Vendor URL	Quantity	Cost	Total Cost
Raspberry Pi 3	https://www.amazon.com	2	\$34.88	\$75.76
1.5k Ohm Resistors	https://www.amazon.com	9	\$0.71	\$6.39
LED's	https://www.amazon.com	9	\$0.35	\$3.15
16 GB SD Cards	https://www.amazon.com	2	\$11.84	\$23.86
Amazon Libraries	N/A	1	N/A	
Relay	https://www.amazon.com	1	\$7.88	\$7.88
Logic Level Shifter	https://www.amazon.com	1	\$8.00	\$8.00
				\$115.98

REQUIRED PROJECT TOOLS/EQUIPMENT

- Computer Monitor
- Soldering Iron



PROJECT BUILD STEPS

Install Raspbian Operating System onto each Raspberry Pi

Visit <https://www.raspberrypi.org/learning/software-guide/quickstart/> and read the Raspberry Pi Software Guide section titled 'Install Rasbian with NOOBS'. This guide will link you to a download page for an SD memory card formatter (https://www.sdcard.org/downloads/formatter_4/index.html).

Follow the instructions to install the SD memory card formatter software on your computer.

Insert the SD card into your computer's SD card reader slot. (If your computer doesn't have an SD card reader slot then you will need to use a USB SD card reader). For our project we used a micro SD to SD card adapter that came with the micro SD cards we bought.

When you plug in the SD card, take note of the drive letter of the SD card as you will need it later for formatting your SD card. Format the SD card using Fat32, as this is necessary for getting your SD card to run the Raspbian operating system on. Also make sure that the SD card is at least 16GB or more in order to have enough storage capacity.

Next, we will need to download NOOBS (New Out Of the Box Software) from the downloads page on the official Raspberry Pi website. You can download NOOBS directly from <https://www.raspberrypi.org/downloads/noobs/>. Make sure to download it to the same computer that has the SD card reader slot and SD card formatter software installed on.

After the download is done, you will want to unzip the file onto your computer. Then, to get the files onto your recently formatted SD card, you will want to open the folder that contains the unzipped NOOBS files and simply drag and drop the entire folder onto the recently formatted SD card drive. Patiently wait for all of the files to over, and eject the SD card drive from your computer once it is complete. Remove the micro SD card from the adapter and place it into your Raspberry Pi.



Connect the power cable, HDMI cable to a monitor, a USB mouse and USB keyboard to your Raspberry Pi. When the Pi boots up, you will see a screen that has installation options for the software on your SD card. Click the white box to the left of the Raspbian software and then click the install button in the top left of the window. Once you click yes on the next window that pops up, the Raspbian Operating System will begin to properly configure itself onto your Raspberry Pi's SD card. Once this is done then you have successfully loaded the Raspbian OS onto your Raspberry Pi.

We did this process twice; once for the Pi we used as the https endpoint for the Alexa commands, and once for the Pi that we used as the DIY Alexa.

Connecting to the Internet:

Open the Raspberry Pi's terminal window and type in the following command

```
sudo nano /etc/network/interfaces
```

The file will have an information header that is commented out with this symbol: #

Change the file, so that it looks like this:

```
1 # configure the file to look like this
2 # ignore the '#' symbols, because they are commented phrases
3
4 source-directory /etc/network/interfaces.d
5
6 auto lo
7 iface lo inet loopback
8
9 iface etho0 inet dhcp
10
11 allow-hotplug wlan0
12 iface wlan0 inet dhcp
13     pre-up wpa_supplicant -B -Dwext -i wlan0 -c/etc/wpa-supPLICANT/wpa_supplicant.conf
14     post-down killall -q wpa_supplicant
15
16
17 allow-hotplug wlan1
18 iface wlan1 inet manual
19     wpa-conf /etc/wpa-supPLICANT/wpa_supplicant.conf
```



This script prompts the mini computer to access `/etc/wpa_supplicant/wpa_supplicant.conf` file for login credentials

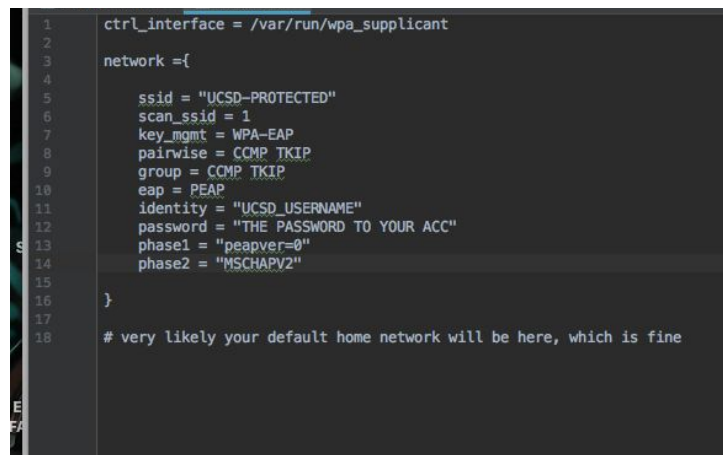
Save and exit the file by entering in the command line of nano :

ctrl O (on prompt, type 'y' to overwrite), ctrl X

Now type the following command line into terminal:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

In that file, add this text, ignoring the comment:

A screenshot of a terminal window showing the nano text editor editing the file `/etc/wpa_supplicant/wpa_supplicant.conf`. The file content is as follows:

```
1 ctrl_interface = /var/run/wpa_supplicant
2
3 network = {
4     ssid = "UCSD-PROTECTED"
5     scan_ssid = 1
6     key_mgmt = WPA-EAP
7     pairwise = CCMP TKIP
8     group = CCMP TKIP
9     eap = PEAP
10    identity = "UCSD_USERNAME"
11    password = "THE PASSWORD TO YOUR ACC"
12    phase1 = "peapver=0"
13    phase2 = "MSCHAPV2"
14 }
15
16 # very likely your default home network will be here, which is fine
17
18
```

Note: There may be a default home network set-up. For our purposes, we chose to comment the home block of code.

Adding this line will give the login credentials the UCSD-PROTECTED wifi is looking for. Once finished, type the following command to reboot the pi, and connect to UCSD wifi:

```
sudo reboot
```

Congrats, you are now connected to UCSD's wifi. If there were any errors, it was very likely that the script was typed incorrectly.



Integrate Alexa Onto a Raspberry Pi:

Making a developer account

Before you connect to the internet, create an amazon developer account.

1. Create an account according to your credentials, and follow through security agreements and details.

2. Click on the Alexa tab, and register a product with any information. Make sure to click no on commercially using the product.

3. Once it is created, click on the manage tab and change the Allowed origins and Allowed returns to the following:

Origin: <https://localhost:3000>

Return: <https://localhost:3000/authresponse>



Products

Product name	Product ID	Amazon ID	Security profile	Category		
PI Whisper	pi	A2RM7YINYWA7TQ	pi	Whisper	Metrics	Manage



Client secret ?

f3eb71a9acf19bbfb40deca9aa935f5d8e96822b43b5961c588216ced50ed4dd

COPY

Allowed origins ?

https://www.example.com

ADD

https://localhost:3000

×

Allowed return URLs ?

https://www.example.com/login.php

ADD

https://localhost:3000/authresponse

×

UPDATE

Enter in the https origin and return as such

And we are done with creating and configuring the Amazon Developer Account. Make sure to take note of the Client/Product/Secret IDs.

ECE 196 | FALL 2017

9



Alexa on Pi:

Once that is created, open a terminal window on the raspberry pi and type the following command:

```
cd Desktop
```

This will bring you into the Desktop directory. This will allow easy access to necessary scripts, and will be the cloning location of the amazon scripts.

Type in the following command to clone the necessary scripts onto the raspberry pi:

```
git clone https://github.com/kvora97/DIYAlexa
```

Once that is complete, type the following command to enter the folder downloaded:

```
cd ~/Desktop/kvora97/DIYAlexa
```

Then type the following command to open the automated installation:

```
nano automated_install.sh
```

This will open the file with nano. Using the arrow keys to navigate, enter the “Product ID, Client ID, and Client Secret. In the document, the location to enter it in will be denoted by “YOUR_”. Type in appropriately after the = sign, and save and exit the file.

Now type in the following command to run the automated installation script.

```
. automated_install.sh
```

While running, it will prompt the user with several questions. Answer “Y” for the yes/no questions; for the remainder, we answered according to our desired configuration. Feel free to customize.

Note: This configuration process may take up to 30 minutes, so do not stop any processes.

Next, we will run several terminal windows to set up a connection with Amazon, an interactive java client, and keyword recognizing agent. Make sure to keep all windows open!



Open a new terminal window, and get into the DIYAlexa folder with this command:

```
cd ~/Desktop/kvora97/DIYAlexa/samples/companionService
```

Next, run the companion service which opens a port to communicate with Amazon:

```
npm start
```

Now that we have established a connection with Amazon, we need to run an interactive Java client and setup the raspberry pi device so that it can communicate to Amazon.

Type in the following commands:

```
cd ~/Desktop/kvora97/DIYAlexa/samples/javaclient
```

```
mvn exec:exec
```

Note: Do not click ok on the Java app until instructed to do so. First, the device authentication needs to be in place.

This will open a window stating that we need to authenticate our device. Click “yes” and in the new browser, and log into your Amazon Developer account. Once there, you will see an authentication screen for the raspberry pi device; click ok, and the browser should display “device tokens ready.”

Now click ok on the Java app. This application is an interactive Alexa model, which will allow the user to manually begin recording. The next agent we will be setting up will allow the Alexa keyword to start the audio recording.

Note: Keep all terminal windows running

Open the third and last terminal window. This is where we will be running the wakeWordAgent software so that audio can be recorded and sent to Amazon. These scripts will search for the Alexa key word, and record afterwards.

Type in the following command, which will navigate us to the required software:



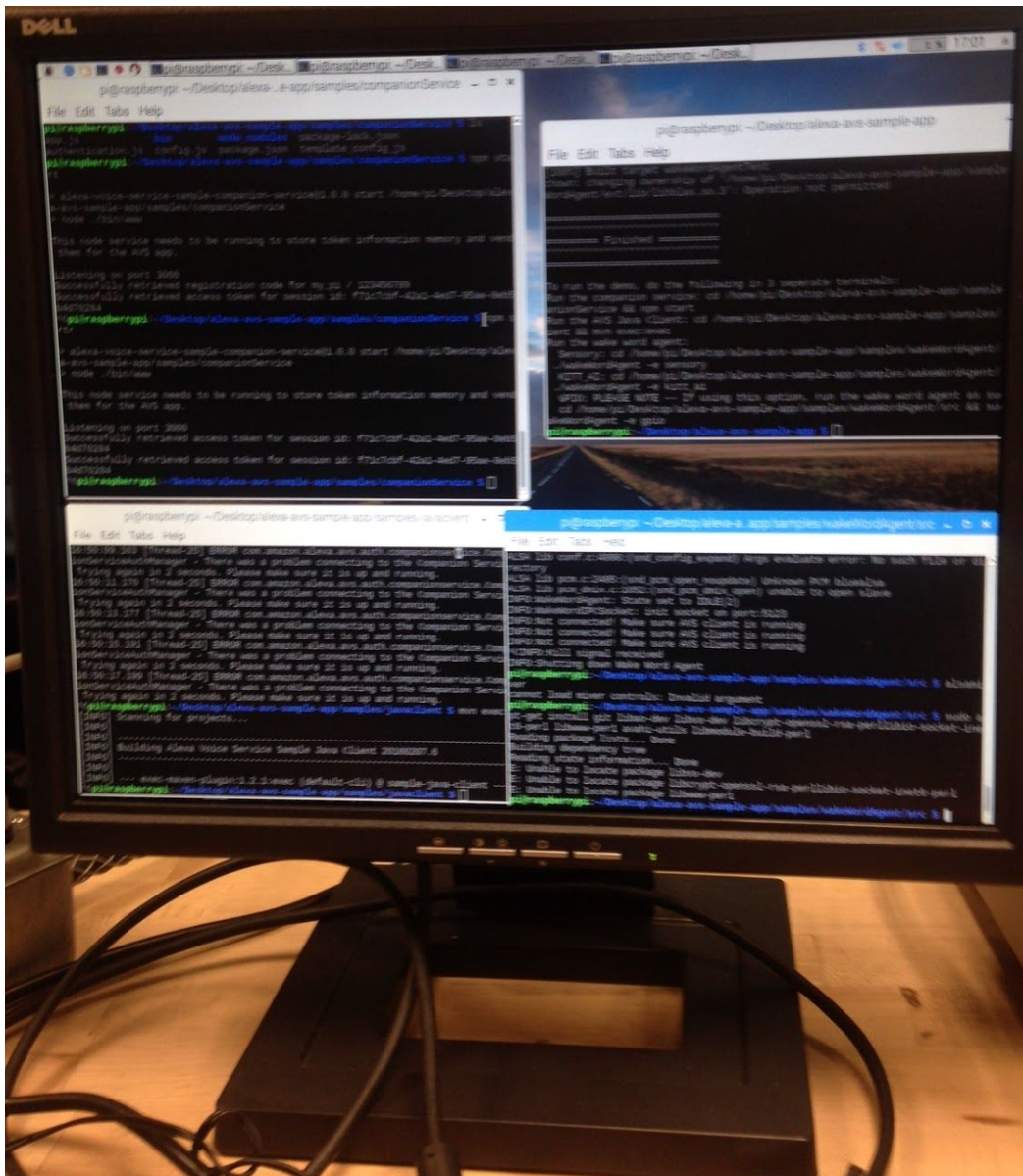
```
cd ~/Desktop/kvora97/DIYAlexa/samples/wakeWordAgent/src
```

Next, enter the command which will run the DIYAlexa :

```
./wakeWordAgent -e kitt_ai
```

The Alexa should be up and running now. When it hears Alexa, it should return a beep to notify that it is recording. If that functions, feel free to ask it anything you would ask an Alexa. The javaclient is an alternative method of usage; instead of using the keyword 'Alexa', click the play/stop button to begin and stop recording. This offers a more controlled method of voice input.

Keep in mind, this is an Alexa with a database. If one wants to add information to Alexa, just download the phone application, and it will add the library to your Alexa.



The display should look something like above. We have an extra terminal to run other extensions.

Note: Questions about the time may return different values depending on the developer account. If anything is different from expected, it can be changed in the settings. Otherwise, you can ask “what is the time in ‘place’” to get an accurate answer.



Extension: Airplay Support

Although our Raspberry Pi is running as an Amazon echo, we can add several extensions that an echo cannot support. For the next part of this project, we will be implementing “Airplay Support” to the raspberry pi, so that it can stream music from different audio outputs, such as speakers.

Type in the following command to download the software we need:

```
sudo apt-get install git libao-dev libssl-dev libcrypt-openssl-rsa-perl libio-socket-inet6-perl libwww-perl avahi-utils  
libmodule-build-perl
```

Note: This is all in one line

When prompted, type ‘y’ and wait for everything to install.

Next, we need to build and install the latest software necessary for Apple Airplay support. Execute the following commands to do so:

```
cd ~/Desktop/kvora97/DIYAlexa/perl-net-sdp  
  
perl Build.PL  
  
sudo ./Build  
  
sudo ./Build test  
  
sudo ./Build install
```

Once everything is installed, you have the necessary software to support Airplay. Now you need to access shairport, the airplay software. Type in the following commands:

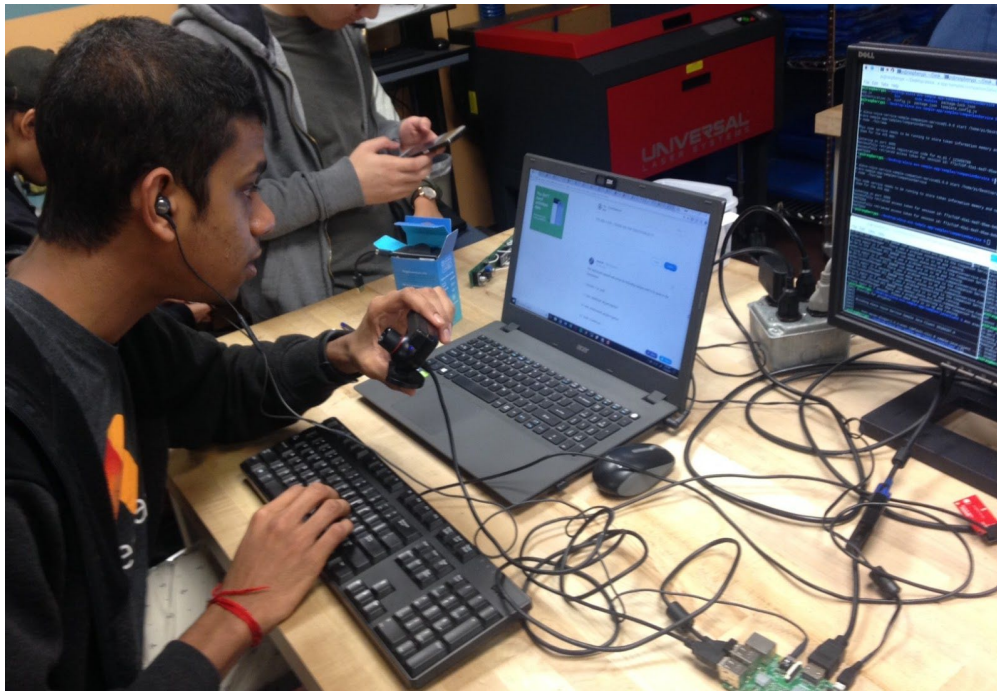
```
cd ~/Desktop/kvora97/DIYAlexa/shairport  
  
make  
  
./shairport.pl -a AlexaPi
```

Now we can Airplay music from our phones, and have it output from the raspberry pi audio port. Simply connect to the same wifi, and airplay from “pi”.



Note: If the raspberry pi is not outputting any audio through the AUX, it most likely is outputting from HDMI. To change this, type in the following into a terminal window.

```
amixer cset numid=3 1
```



Debugging the audio output



Setting up a Raspberry Pi as an https Endpoint Using ngrok:

Connect the Raspberry Pi (not the one that the DIY Alexa was installed onto) to the internet and then open a terminal and type the following commands into the terminal.

```
sudo apt-get update && sudo apt-get upgrade -y
sudo apt-get install python2.7-dev python-dev python-pip
sudo pip install Flask flask-ask
```

On the pi, open the web browser and go to <https://ngrok.com/download> and download ngrok for linux ARM. Unzip the download by typing the following command into the terminal.

```
unzip /home/pi/Downloads/ngrok-stable-linux-arm.zip
```

Next we will use ngrok to create an https endpoint for the pi using the following command. This will use port 5000, and you will need to make sure that this port is open on your router. (Don't worry about it at first, check to see if everything works, and go back to check the port on your router settings if it does not work. At school the port is open).

```
sudo ./ngrok http 5000
```

Once you type in the command, you will see a window similar to the one shown below. Take note of the Forwarding https web domain, as you will need to copy this into your Alexa Skill each time you activate your https endpoint using ngrok because it changes each time you open it up. In this example, our unique https endpoint is <https://fcb917e0.ngrok.io>.

A screenshot of a terminal window titled 'pi@raspberrypi: ~'. The window shows the output of the 'ngrok' command. The output includes session status, version, region, web interface, forwarding URLs, and a table of connections.

```
pi@raspberrypi: ~
File Edit Tabs Help
ngrok by @inconshreveable (Ctrl+C to quit)

Session Status      online
Version             2.2.8
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding            http://fcb917e0.ngrok.io -> localhost:5000
Forwarding            https://fcb917e0.ngrok.io -> localhost:5000

Connections          ttl    opn    rt1    rt5    p50    p90
0                  0      0.00   0.00   0.00   0.00
```



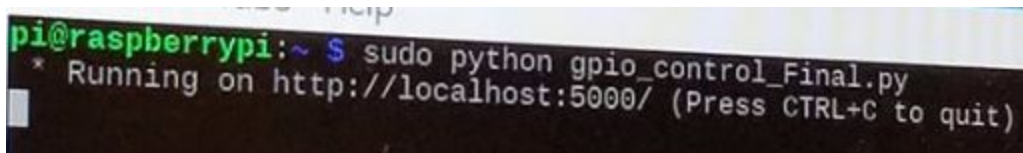

Python Script that Converts the Commands into GPIO Outputs

Open a new terminal session and create a new python file named `gpio_control.py` by typing the following command.

```
nano gpio_control_Final.py
```

Get the code from our GitHub https://github.com/kvora97/DIYAlexa/blob/master/gpio_control_Final.py and copy and paste it into your python script. Save the file with `cntrl+O` then `ENTER` then `cntrl+X` to exit. Next, you will need to have the script running while the ngrok https endpoint is running. To Run this python script, type the following command into the terminal window. If your script is running properly, then you will see a response like the one shown in the image below.

```
sudo python gpio_control_Final.py
```



Note: you will have to run the ngrok and the python script together each time you wish to use the Raspberry Pi as an https endpoint for your Amazon Alexa Skill. If your Alexa skill is already set up then you should be able to communicate with the Raspberry Pi from your Alexa or Echo dot now. If you need to set up the Raspberry Pi GPIO outputs and/or the Amazon Alexa Skill please proceed to the next sections of build steps.

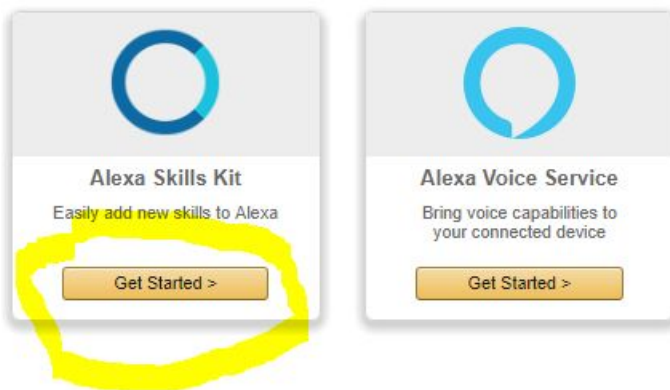


Setting up Amazon Alexa Skill

First, you will need to create or login to an Amazon Developer Account <https://developer.amazon.com/>. Once logged in, you will need to make sure that this is the same account you login to you DIY Alexa or Echo Dot with in order to make sure that the Alexa Skill you are about to create gets onto the desired devices. Click on Alexa and you should be routed to this website, <https://developer.amazon.com/edw/home.html#/> , and see the options below. Click get started under the Alexa Skills Kit option.

Get started with Alexa

Add new voice-enabled capabilities using the Alexa Skills Kit, or add voice-powered experiences to your connected devices with the Alexa Voice Service.



In the top right of the screen you will see a button that says 'Add a New Skill', click it and it will send you into the next stage of the skill setup. Here you can see our page already has the skill GPIO_Control , but we will be creating that skill right now.

Building Alexa Skills with the Alexa Skills Kit

Add a New Skill

To learn more about building Alexa skills, see [Getting Started with the Alexa Skills Kit](#). To start building an Alexa skill for free using AWS Lambda, see [Creating an AWS Lambda Function for a Custom Skill](#). We encourage you to visit the [Alexa Developer Forum](#) to collaborate with Alexa team members and fellow Alexa developers.

Good news! Developers can earn money for the most engaging skills

We're rewarding developers who design Alexa skills that customers love most! Developers can earn money each month for eligible skills that have the highest customer engagement in eligible skill categories. What's your next big idea? [Learn more](#).

Name	Language	Type	Modified	Status	Actions
 GPIO_Control View Skill ID & Client Secret	English (U.S.)	Custom	12/6/17	Development ⓘ	Metrics Edit Delete



Name the skill 'GPIO_Control', and set the Invocation name to 'Pi'. You can leave the Global Field options as all No and then click Save. Go to the next step of the setup which is the Interaction Model.

Create a New Alexa Skill

Skill Information ✓	Skill Type Define a custom interaction model or use one of the predefined skill APIs. Learn more	<input checked="" type="radio"/> Custom Interaction Model <input type="radio"/> Smart Home Skill API <input type="radio"/> Flash Briefing Skill API <input type="radio"/> Video Skill API
Interaction Model ✓		
Configuration ✓		
SSL Certificate ✓	Language Language of your skill	English (U.S.) ▼
Test ✓	Name Name of the skill that is displayed to customers in the Alexa app. Must be between 2-50 characters.	GPIO_Control
Publishing Information ✓	Invocation Name The name customers use to activate the skill. For example, "Alexa ask Tide Pooler...".	Pi
Privacy & Compliance ✓		

i For successful Alexa Skills Certification, please review and follow our [Invocation Name Guidelines](#) as well as our [Certification Requirements](#).

Global Fields

These fields apply to all languages supported by the skill.

Audio Player
Does this skill use the audio player directives? ☐ Yes ☒ No
[Learn more](#)

Video App
Does this skill use the video app directives? [Learn more](#) ☐ Yes ☒ No

Render Template
Does this skill use the Render Template directives? ☐ Yes ☒ No
[Learn more](#)

Save



Skill Information ✓

Interaction Model ✓

Configuration ✓

Test ✓

Publishing Information ✓

Privacy & Compliance ✓

Skills Beta Testing NEW

Status Not yet eligible ⓘ

Try the skill builder (beta), an intuitive interface for building your interaction model and creating dialog prompts

Launch Skill Builder BETA

Intent Schema

The schema of user intents in JSON format. For more information, see [Intent Schema](#). Also see [built-in slots](#) and [built-in intents](#).

✕ 1

Custom Slot Types (Optional)

Custom slot types to be referenced by the Intent Schema and Sample Utterances. For general information about custom slots, see [Custom Slot Types](#).

Enter Type

Type

Enter Values

Values must be line-separated

1

Cancel

Add

Copy and paste the following code into the Intent Schema section of the Interaction Model.

```
{
  "intents": [
    {
      "slots": [
        {
          "name": "status",
          "type": "GPIO_CONTROL"
        },
        {
```



```
"name": "location",  
  "type": "LOCATION"  
},  
],  
  "intent": "LocationControlIntent"  
}  
]  
}
```

Next we setup our Custom Slot Types. We set the type to LOCATION and then we entered the following values into the Enter values section. Enter the values in one per line, and then click add when you are done. Here we included options for the Nato code to represent letters (one A could also be controlled by voice as one alpha etc.), this greatly improved the recognition accuracy when trying to control the individual LEDs.

```
light  
one A  
one B  
one C  
two A  
two B  
two C  
three A  
three B  
three C  
one alpha  
one bravo  
one charlie  
two alpha  
two bravo  
two charlie
```



three alpha
three bravo
three charlie
grid
cross
ex
square
diamond
row one
row two
row three
column one
column two
column three
corners

Custom Slot Types (Optional)

Custom slot types to be referenced by the Intent Schema and Sample Utterances. For general information about custom slots, see [Custom Slot Types](#).

Enter Type

LOCATION

Enter Values

Values must be line-separated

```
21 cross
22 ex
23 square
24 diamond
25 row one
26 row two
27 row three
28 column one
29 column two
30 column three
31 corners|
```

Cancel

Update

Add one more custom type called GPIO_CONTROL as seen below.



Enter Type

GPIO_CONTROL

Enter Values

Values must be line-separated

```
1 on
2 off
3 high
4 low
```

Cancel

Update

Click Update and then fill in Sample Utterances with the following lines. Then click Save. Then click Next.

LocationControlIntent to turn {location} {status}

LocationControlIntent to change the {location} to {status}

Sample Utterances

These are what people say to interact with your skill. Type or paste in all the ways that people can invoke the intents. [Learn more](#)

Up to 3 of these will be used as Example Phrases, which are hints to users.

```
1 LocationControlIntent to turn {location} {status}
2 LocationControlIntent to change the {location} to {status}
```

See [Certification Requirements](#) in our technical documentation as you develop your skills and prepare to submit to Amazon.

Save

Submit for certification

Next



In the Configuration Section, Set the Service Endpoint Type to HTTPS and you will see something like the following image.

Skill Information ✓

Interaction Model ✓

Configuration ✓

SSL Certificate ✓

Test ✓

Publishing Information ✓

Privacy & Compliance ✓

Skills Beta Testing NEW

Status Not yet eligible ⓘ

Global Fields

These fields apply to all languages supported by the skill.

Endpoint

Service Endpoint Type: ☐ AWS Lambda ARN (Amazon Resource Name) ⓘ ☒ HTTPS

Recommended
AWS Lambda is a server-less compute service that runs your code in response to events and automatically manages the underlying compute resources for you.
[More info about AWS Lambda](#)
[How to integrate AWS Lambda with Alexa](#)

Default

Provide geographical region endpoints? ☐ Yes ☒ No ⓘ
(Optional)

Account Linking

Do you allow users to create an account or link to an existing account with you? ☐ Yes ☒ No

[More info about Account Linking](#)
[Tips for successful Account Linking](#)

Permissions

Request users to access resources and capabilities
Please request permissions to resources and capabilities that are absolutely core to the customer experience delivered by the skill.
[Learn More](#)

☐ Device Address
☐ Full Address ⓘ
☐ Country & Postal Code Only ⓘ

☐ Lists Read ⓘ
☐ Lists Write ⓘ

Here is how we will link the Alexa skill to the https endpoint on the Raspberry Pi that is using ngrok. Look at the Default https endpoint and put the https address from the ngrok service that you have running on the Raspberry Pi into this line.

Default

PUT THE HTTPS FORWARDING NGROK DOMAIN HERE!!!



For this session our https endpoint was <https://fcb917e0.ngrok.io> , thus your properly configured Configuration should look like the following image. Once you type in your https endpoint, click Save and then Next.

Skill Information ✓

Interaction Model ✓

Configuration ✓

SSL Certificate ✓

Test ✓

Publishing Information ✓

Privacy & Compliance ✓

Skills Beta Testing ^{NEW}

Status Not yet eligible ⓘ

Global Fields

These fields apply to all languages supported by the skill.

Endpoint

Service Endpoint Type: ☐ AWS Lambda ARN (Amazon Resource Name) ⓘ ☒ HTTPS

Recommended
AWS Lambda is a server-less compute service that runs your code in response to events and automatically manages the underlying compute resources for you.
[More info about AWS Lambda](#)
[How to integrate AWS Lambda with Alexa](#)

Default

Provide geographical region endpoints? (Optional) ⓘ ☐ Yes ☒ No

Account Linking

Do you allow users to create an account or link to an existing account with you? ☐ Yes ☒ No

[More info about Account Linking](#)
[Tips for successful Account Linking](#)

Permissions

Request users to access resources and capabilities
Please request permissions to resources and capabilities that are absolutely core to the customer experience delivered by the skill.
[Learn More](#)

☐ Device Address
☐ Full Address ⓘ
☐ Country & Postal Code Only ⓘ

☐ Lists Read ⓘ
☐ Lists Write ⓘ

In the SSL Certificate section, you will want to click the option for “My development endpoint is a sub-domain of a domain that has a wildcard certificate from a certificate authority”.



Skill Information	✓
Interaction Model	✓
Configuration	✓
SSL Certificate	⌵
Test	✓
Publishing Information	⌵
Privacy & Compliance	⌵

Skills Beta Testing NEW
Status Not yet eligible ⓘ

Global Fields

These fields apply to all languages supported by the skill.

To protect your security and the security of end users, we require that you use a certificate while developing an Alexa skill. For more information, see [Registering and Managing Alexa Skills - About SSL Options](#).

Certificate for DEFAULT Endpoint:

Please select one of the three methods below for the web service:

- ☐ My development endpoint has a certificate from a trusted certificate authority
- ☒ My development endpoint is a sub-domain of a domain that has a wildcard certificate from a certificate authority
- ☐ I will upload a self-signed certificate in X.509 format. [Learn how to create a self signed certificate.](#)

See [Certification Requirements](#) in our technical documentation as you develop your skills and prepare to submit to Amazon.

Save

Submit for certification

Next

Testing the Skill and seeing if it works and properly communicates with your Raspberry Pi can be done from the Test section of your skill in the Enter Utterance section of Service Simulator in the Test section of the skill setup. If everything in the setup was done properly, then you will see the response show up on your Pi. (If you run into any 502 bad gateway errors then you will need to check your python script and make sure the syntax is proper, and that your port traffic is open on port 5000. It took us a like 7 hours to figure out how to get it working properly. That is why we recommend copying our python script from our GitHub because it to so long to get it working properly.)



Test

Publishing Information

Privacy & Compliance

Skills Beta Testing ^{NEW}

Status Not yet eligible

The skill is available in "Skills > Your Skills" page of the Alexa App when you select 'Yes' above. You can then enable the skill and test its functionality on your device by asking Alexa, ask pi

For successful Alexa Skills Certification, please test for our requirements on [Session Management](#) and [Error Handling](#).

Voice Simulator

Hear how Alexa will speak a response entered in plain text or SSML. [Learn more about supported SSML tags.](#)

For example: Here is a word spelled out: <say-as interpret-as="spell-out">hello</say-as>.

Listen

Service Simulator

Use Service Simulator to test your HTTPS endpoint: <https://fcb917e0.ngrok.io> ▼

Note: Service Simulator does not currently support testing audio player directives, dialog model, customer permissions and customer account linking. Text mode does not support launch intents and single interaction phrases.

Text

JSON

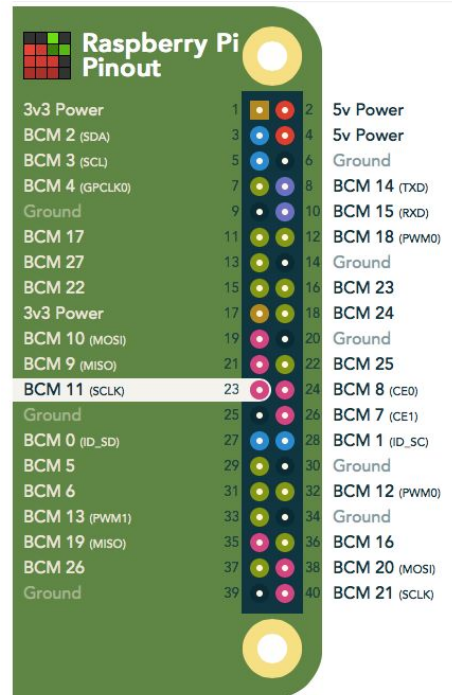
Enter Utterance

Ask GPIO_Control

Reset

Setting up Raspberry Pi GPIO Outputs & Making the LED Grid

Using the following GPIO pin layout we found from <https://pinout.xyz> we developed the GPIO pin logic to implement the different modes of our LED matrix.

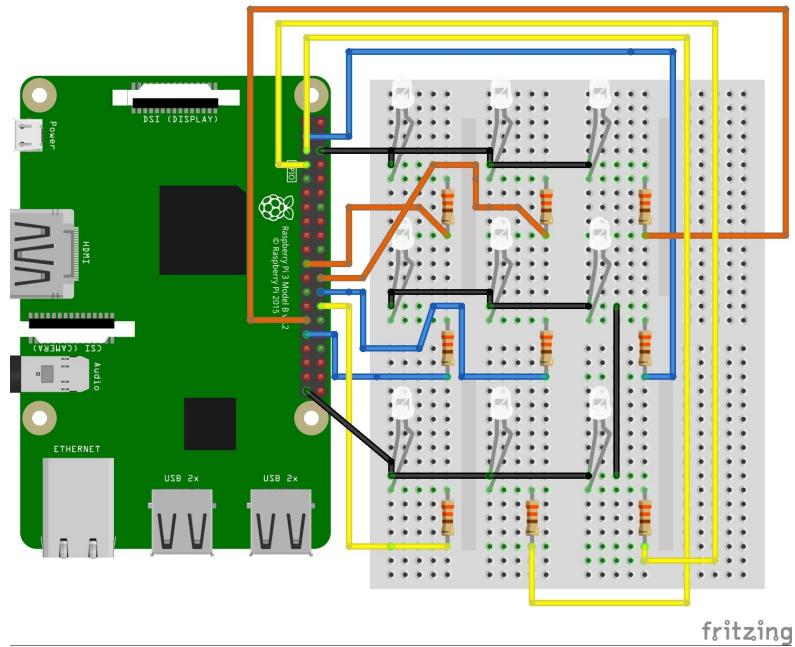


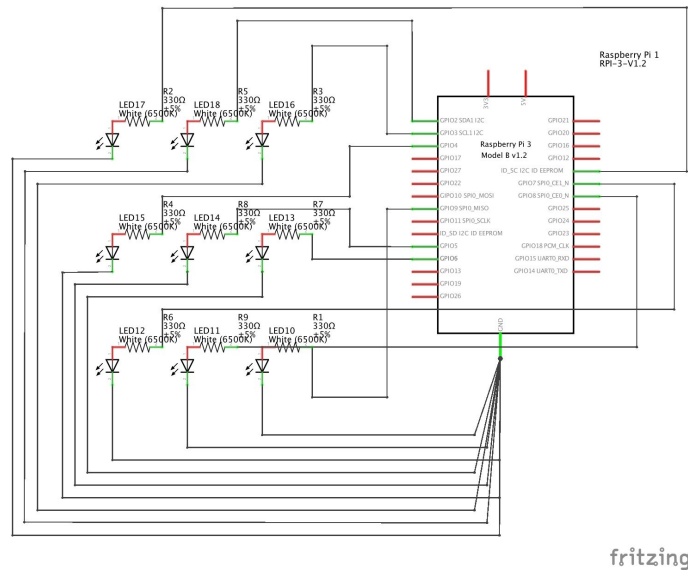
For implementation purposes we then named each LED and have shown which pin corresponds to which LED as well as the name of that LED so that we can control the LED using speech control below:

- Pin 28 - BCM 1 - 1 alpha
- Pin 3 - BCM 2 - 1 bravo
- Pin 5 - BCM 3 - 1 charlie
- Pin 7 - BCM 4 - 2 alpha
- Pin 29 - BCM 5 - 2 bravo
- Pin 31 - BCM 6 - 2 charlie
- Pin 26 - BCM 7 - 3 alpha
- Pin 24 - BCM 8 - 3 bravo
- Pin 21 - BCM 9 - 3 charlie
- Pin 6 - Ground - no command

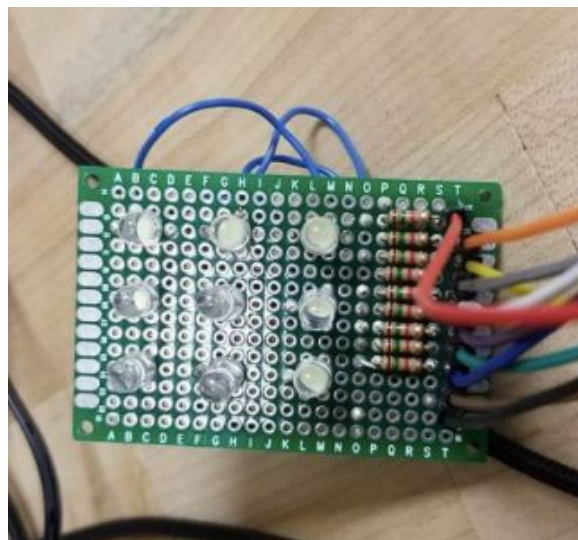
We then configured the layout of the of our matrix in the format shown on our schematic using 9 LED's and 9 resistors:

PROJECT SCHEMATICS





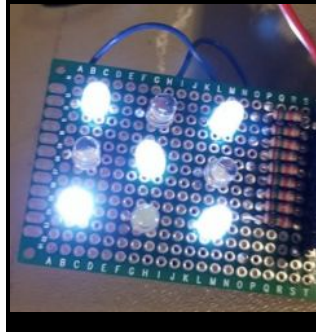
Next we proceeded to solder our formatted matrix onto a protoboard as shown below:



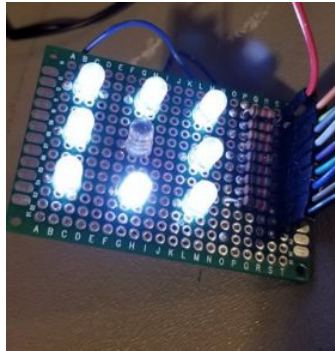


The demonstrations to our configured modes which can be controlled are shown below:

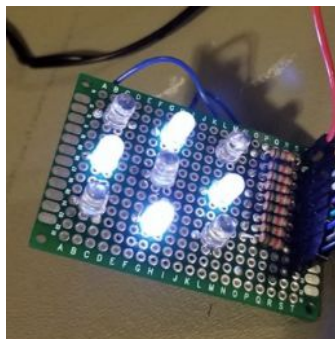
“X”



“Square”

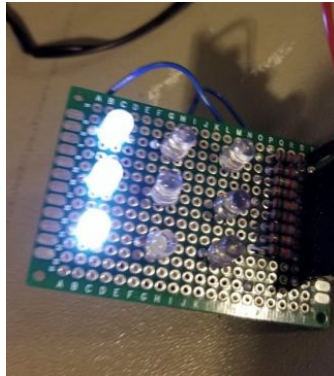


“Diamond”

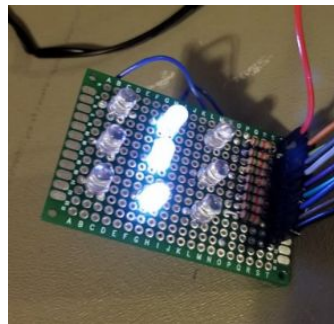




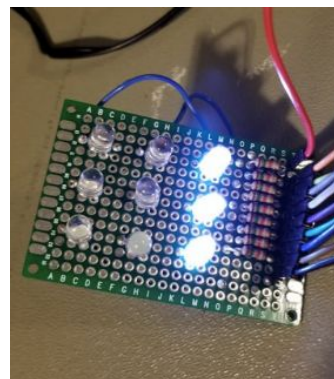
"Row 1"



"Row 2"

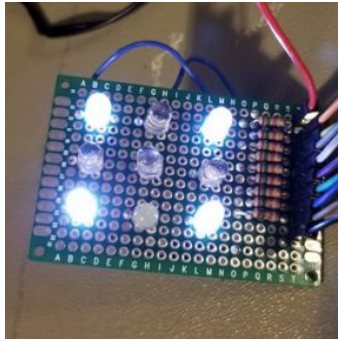


"Row 3"





"Corners"



PROJECT TIMELINE

Week 6: Brainstorm ideas and Project Outline (#6)

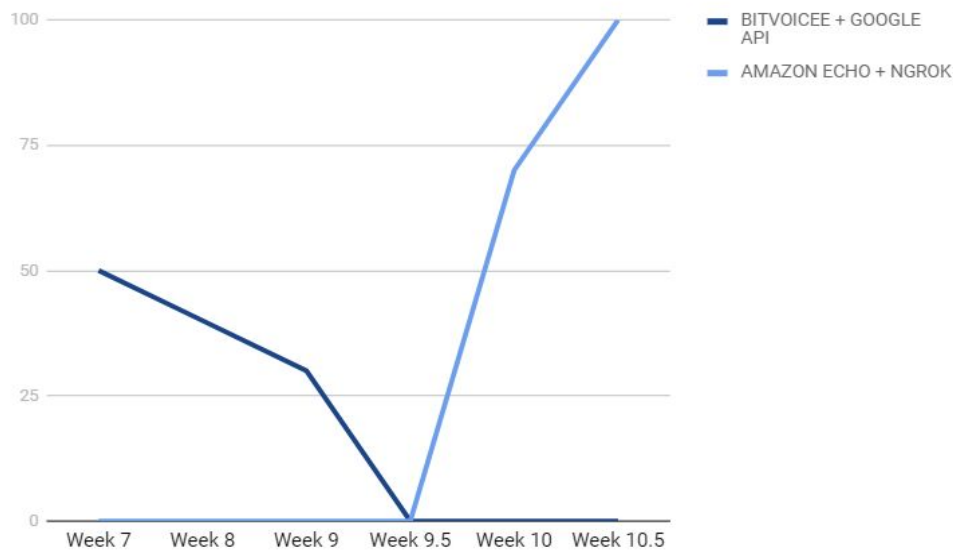
= hours

Week 7: Put together Hardware and Start Documentation (#15)

Week 8: Dependency Failures due to outdated Libraries (#15)

Week 9: Pivot to Amazon Libraries & Ngrok Client (#40)

Week 10: Final Testing, Documentation and Mock Presentations (#25)



TEAM MEMBER RESPONSIBILITIES

Mike Ranis: For this project, I set up and soldered the LED grid. I also hooked up the LED grid to the GPIO pins on the Raspberry Pi. I had to make sure that these connections corresponded directly to the python script that we ran on the Raspberry Pi that communicated with the ngrok client. I also coded the python script that processed the incoming voice commands for the Alexa servers and coded all oth the logic responsible for converting those into specific GPIO output patterns,of which I also came up with the design on how we were going to control them. I then tested every pattern we coded for and made sure that it worked perfectly as designed. I set up the Amazon Alexa skill that we used and loaded it onto the Amazon Echo dot that we demoed with. I also hooked up the Relay circuit that we used to demo the light turning on and off. I also set up the presentation board that kept everything nice and neat.

Kenil Vora: My contribution towards this project was working on integrating Alexa on the Raspberry Pi so that it can communicate with the ngrok server and run the custom alexa skills that were added using my Amazon Developer Account. There were also a lot of dependency issues for the node package manager that we faced and I debugged all that so that the Alexa on Pi can efficiently communicate with the server. Furthermore, I spent time on maintaining our code on my Github account so that we can efficiently keep track of all the gradual progress that we made as we proceeded and also helped with the documentation, taking photos of team project to keep track of team progress. Furthermore I also helped to set up the custom alexa skills that are used by Alexa to recognize the input commands and take corresponding action. before we moved on to the Alexa project, I spent time with Ammar and Samuel to figure out the problems we were initially having with the BitVoicer Server in the Arduino IDE with the 'port-not-found' errors. The idea to move on from the BitVoicer project to the Alexa on Pi was my idea and I recommended the team to take on that project because I had found a lot of useful documentation online that would help us to finish the project in time and I have had some experience configuring that in the past. Finally I helped Mike to setup the GPIO logic and some of the Python and Unix Scripting functionality too.



Victor Wu - For this project, I worked with Kenil on integrating Alexa onto the Raspberry pi. When downloading the code, some of the software was outdated. We pushed working versions that used the correct pi software so that the wakeWordAgent could function properly. After Alexa worked properly, I integrated Airplay to expand upon the amazon frame. I also brainstormed ways of accessing Amazon's database of information; when we realized it was inaccessible, I worked with the team on manipulating skills to reach the ngrok client endpoint solution. Before we switched projects, I had written code to access the Google Speech recognition server; it also included preliminary string processing logic. This was to be implemented with our previous pi project, and when the bugs were too much, Kenil and I pivoted to the Amazon project, which had more updated libraries. I attempted 3 different audio saving library implementations, however they all lead to dependency failures. Lastly, I documented our project and helped make the presentation.

Samuel Chora - For this project I helped with the conceptualizing of the hardware response that we were going to use as the proof of concept for the speech control system, which was the LED matrix. I also was checking out other speech control software as part of the brainstorming and trying to see which one would be good for what we wanted to do. One of these alternatives was building an android app on MIT app inventor to control LED's using a bluetooth module and an android phone. I was also part of the implementation of the hardware response as well as helping to look for some of the solutions to some of the issues we had when trying to tunnel to our local host.



REFERENCES

Shoutout to RJ for helping us with the python script that communicated with the ngrok host tunneling

- GPIO Pin Input/Output
 - <https://pinout.xyz/>
- Controlling Pi GPIO using Alexa:
 - <https://www.instructables.com>
- Adding & Testing custom Alexa skills:
 - <https://developer.amazon.com/docs/custom-skills>
- Configuring Alexa on a Raspberry Pi
 - <https://lifehacker.com/how-to-build-your-own-amazon-echo-with-a-raspberry-pi-1787726931>