Dude! – the Smart Speaker
Final Report

Team Members

Jing Huang

JingTao Xu

An Wu

HingOn Miu

Prefatory Information

Our speakers are connected through a central server so that information can be shared remotely. Furthermore, each speaker has its own local computation for basic functionality to provide services like personal virtual assistant. Our device essentially contains a Raspberry Pi 2, a microphone, a speaker, a camera, a temperature sensor and a light sensor. Obviously, the Raspberry Pi 2 is responsible for local computation and upload/download information to server. The microphone and the speaker are used for language processing. Basically, user commands are recorded by the microphone and the audio files are uploaded to Google server for speech-to-text translation. Then, our speaker's text responses are again uploaded to Google server for text-to-speech audio synthesis and the audio files returned are played back to the user. The camera is responsible for facial recognition so that the device is able to authenticate user identity. Last but not least, the temperature sensor and the light sensor are used to provide precise quantitative recordings when user feels cold or dark. The central server stores a user profile database so that each user is able to interact with our speaker on personal matters.

1. Project description

The goal of our project is to develop a smart speaker placed in common household rooms that can listen to human voice 24/7, and respond to commands like Siri on Apple iPhone and Echo developed by Amazon. Basically, it is able to perform tasks like searching Wikipedia and weather on the internet to respond to human voice. Moreover, the speaker should also be able to "talk to" other speakers remotely. In other words, the speaker can share messages and information like temperature, brightness and user profiles with other speakers through WiFi. In addition, our speaker is able to use facial and speaker recognition to detect different users, thus it can set up different profiles for different users.

The speaker is composed of multiple essential sensors (i.e microphone, temperature sensor, and brightness sensor), an actuator (i.e speaker), and a data centre processing unit on the hardware side. We will also need to write proper software to enable functionalities like speech recognition and data sharing.

2. Design requirements

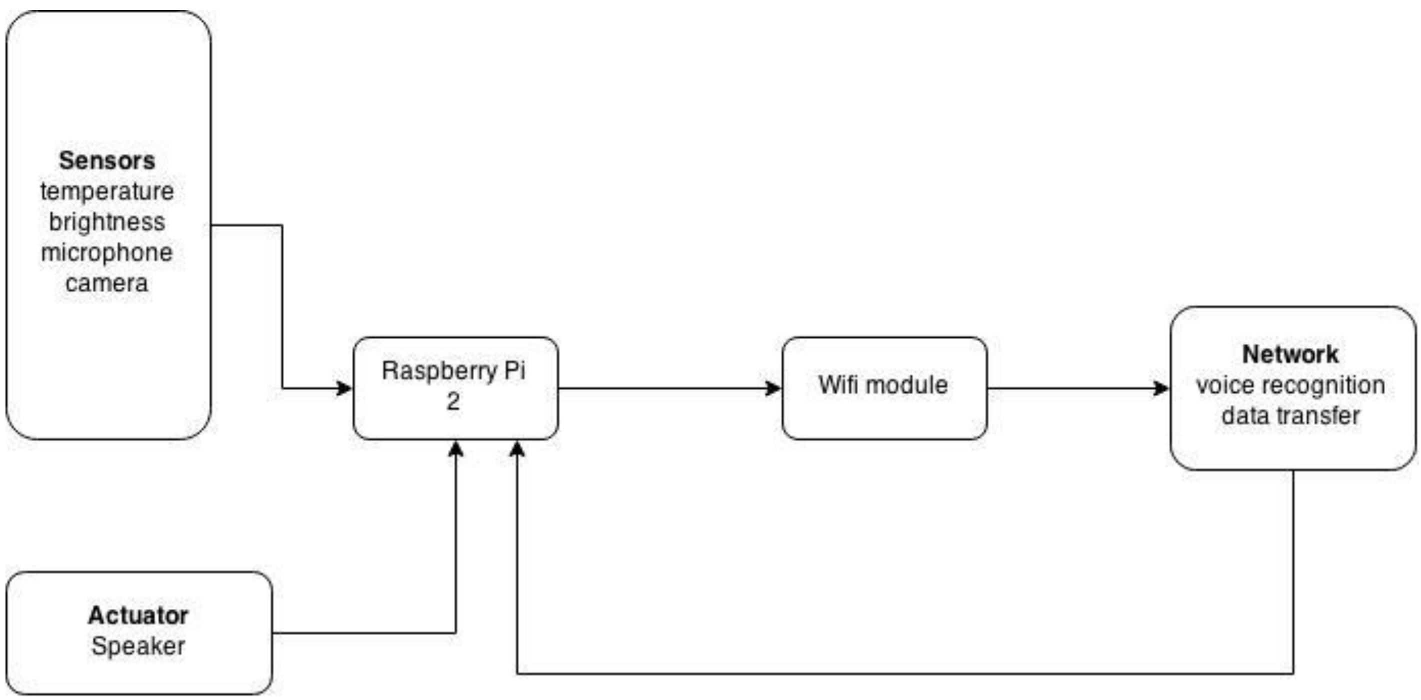
2.1. Explicit requirements from the project specifications

1. ***Dude*** should detect and listen to human voice in a room 24/7.
2. ***Dude*** should use speech recognition to understand commands.
3. ***Dude*** should use facial recognition to recognize people.
4. ***Dude*** should use text-dependent speaker recognition to differentiate users.
5. ***Dude*** should use web APIs to respond to voice query.
6. ***Dude*** should detect environments using various sensors.
    a. Temperature
    b. Brightness
    c. Sound
7. Multiple ***Dude*** should be able to communicate over WiFi.

2.2. Implied design requirements

1. ***Dude*** should use Google Speech API to convert voice into text.
2. ***Dude*** should use Google Translate API to convert text into human voice.
3. ***Dude*** should use Wikipedia API and weather API to respond to queries.
4. ***Dude*** should use OpenCV library to perform facial recognition.
5. ***Dude*** should use Fast Fourier Transform and other speech algorithm to perform speaker authentication.

3. Architecture



4. Design Trade-off Studies

4.1. Platform

4.1.1. Microprocessor vs. Mini PC

In terms of hardware platform, we can either work on a microprocessor or a mini PC. Here is a comparison between the two platforms:

	Microprocessor	**Mini PC**
Computation Power (Processor, RAM, Memory, etc.)	Low	Relatively High
Interfacing with Sensors	Easy	Need Additional Tools
Installation of Softwares	Need to install operating system	Easy
Price	$35 to $45, can have multiple	$100 to $200
Available Systems	Linux, some with Windows but need additional package	Linux or Windows

Using a mini PC makes it easier to integrate functionalities such as motion sensing with Kinect and projecting data/pictures on the wall. Kinect works on Windows machine and most microprocessors support Linux. Also, mini PC can do more complex computation and has larger memory. Those features are preferable for speech recognition and interpretable applications. But we eventually decide to work with microprocessors because they are more accessible. And for our project, we need to demo at least two in a household environment. Second, microprocessors have easy interfaces to communicate with sensors and actuators, as we have at least 5 components to connect with the core hardware platform including Wifi module, microphone, speaker, brightness sensor and temperature sensor. Lastly, if our product is going to be promoted on the market, a microprocessor is much more portable and affordable.

To address the problem that a microprocessor not being able to do very heavy computations, we decided to send data to a remote server or use cloud computation whenever possible. This design trade-off will be covered in section 4.3.

4.1.2. BeagleBone Black vs. Raspberry Pi

We have two common microprocessors to pick from: Raspberry Pi and Beagle Bone Black. A comparison between the two is provided here.

	BeagleBone Black	**Raspberry Pi**
Base Price	45	35
Processor	1GHz TI Sitara AM3359 ARM Cortex A8	700 MHz ARM1176JZFS
RAM	512 MB DDR3L @ 400 MHz	512 MB SDRAM @ 400 MHz
Storage	2 GB on-board eMMC, MicroSD	SD
Video Connections	1 Micro-HDMI	1 HDMI, 1 Composite
Supported Resolutions	1280×1024 (5:4), 1024×768 (4:3), 1280×720 (16:9), 1440×900 (16:10) all at 16 bit	Extensive from 640×350 up to 1920×1200, this includes 1080p
Audio	Stereo over HDMI	Stereo over HDMI, Stereo from 3.5 mm jack

Systems	Angstrom (Default), Ubuntu, Android, ArchLinux, Gentoo, Minix, RISC OS, others…	Raspbian (Recommended), Ubuntu, Android, ArchLinux, FreeBSD, Fedora, RISC OS, others…
Power Draw	210-460 mA @ 5V under varying conditions	150-350 mA @ 5V under varying conditions
GPIO Capability	65 Pins	8 Pins
Peripherals	1 USB Host, 1 Mini-USB Client, 1 10/100 Mbps Ethernet	2 USB Hosts, 1 Micro-USB Power, 1 10/100 Mbps Ethernet, RPi camera connector

Overall, BeagleBone Black has a faster processor (1 GHz vs. 700 MHz) and more storage (2GB and SD vs. SD). While both microprocessors support Ubuntu, BeagleBone black also support Windows with a Windows package. Both microprocessors are relatively cheap, but BeagleBone Black costs ten extra dollars. While Raspberry Pi has been in the market for a much longer time, so there are better documentation and support from the online community. In the end, we decided to go with Raspberry Pi, considering that there are plenty relevant projects done by other developers.

4.2.Voice Recognition: Google Voice API vs. Dragon SDK vs. CMU Sphinx

Implementing speech recognition is a big software decision. It requires a well-supported library to build our own application so that it can provide higher accuracy rate in recognition. In considering this problem, the library needs to be user-friendly and easily expendable, along with providing fast processing speed.

There are two open-sourced APIs for Google voice that use Java (Google-Voice-Java) and Python (PolyGoogleVoice). However, both of them have limited functionality that are only usable in operations with phone communication such as placing calls, sending SMS and searching for files within Google accounts. Another library is called Google Web Speech API, which is more suitable for our project. It uses Google Voice's voice recognition engine and it allows programmers to build websites that allow voice input for forms and sending emails. Through experimenting with Web Speech API, we discovered that it has relatively high accuracy in English vocabularies and the service responds reasonably fast.

Dragon NaturallySpeaking is a speech recognition software package that only runs on Windows PC (7 or 8) or Mac. This software is commonly used in product for voice control computer operations such as opening file, searching internet, and checking the calendar, etc. Dragon NaturallySpeaking Software developer kit (SDK) can also be used to add speech recognition to applications. In addition to the SDK, there is a Python framework that uses Dragon NatuallySpeaking as speech recognition engine and it allows developers to write scripts and macros. This software is very frequently adopted by programmers and it has a fairly good accuracy in speech recognition. However, the license comes with a cost of at least $99.99, and it requires a minimum of 2.2 GHz processing speed, 4GB memory space and 2GB RAM for designated hardware platform.

Another option we have is CMU Sphinx. It is an well-known open source toolkit with various models to build speech recognition applications. For our use, it is better to go with its Pocketsphinx package, since it is much faster in reaction time and much smaller in installation size for higher portability. CMU Sphinx has a support community for developers. There are plenty documentations of how to use Pocketsphinx across various platforms, especially for Linux. Although CMU Sphinx supports multiple operating systems, we think it is best to go with Linux to get better support online.

With these three options, we ruled out Dragon NaturallySpeaking because it is costly and it has stricter requirements on the hardware platform. Ultimately, we decided to use Google Web Speech API because it is fairly simple to use. Then, we have to implement exactly how to record and upload audio recordings as HTTP web requests. We also have to implement software logics for translating texts into commands. There are some other potential audio problems we need to consider such as filtering out background noises from audio microphone input.

4.3.Computation Location: On Chip vs. Cloud

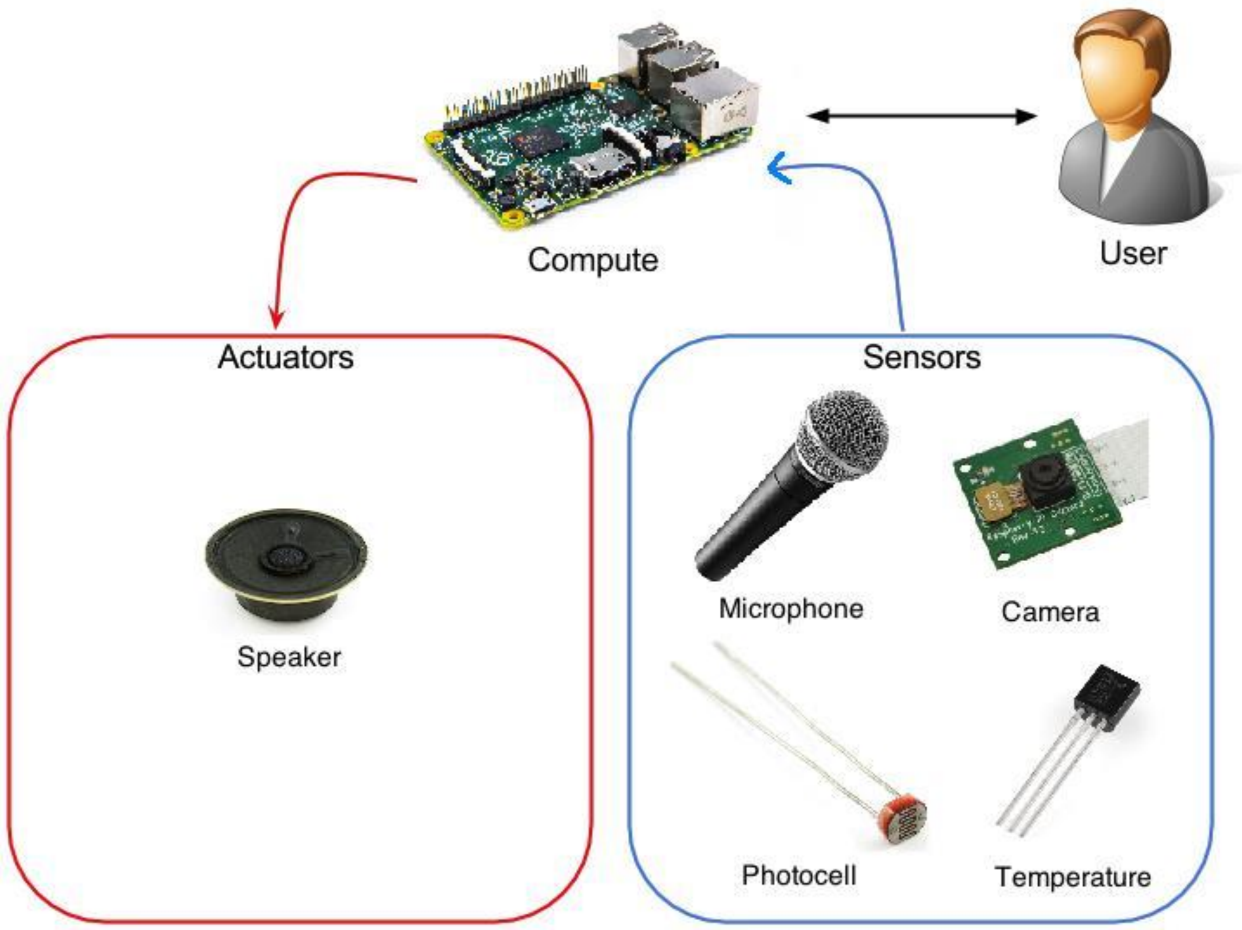
To implement speech recognition and language processing, we can either run these programs locally in the microprocessor processing unit, or we can upload audio recordings to an online service or a remote machine for processing. Running applications locally requires installation of multiple dependent libraries on our device. Anyhow, Raspberry Pi is still not an ideal platform for running large computations and installing large libraries with many dependencies.

Uploading audio recordings to an online service can solve our problems. Amazon Echo adopts this method. It sends audio recordings to Amazon Web Service (AWS) and fetches the interpreted audio texts. We could do the same but the service cost is relatively high. It costs about $0.5 per hour to use an AWS EC2 instance. Since affording to rent an AWS EC2 machine 24/7 is not ideal, it could take long time to set up the speech recognition application each time an audio processing request presents. This is quite detrimental for our speech application, as it always requires immediate response.

Using a remote machine to do such processing is a better option. Therefore we decide to upload audio recordings and user data to a remote computer. However, using full online application is still a long way to go if our product is ultimately being widely manufactured.

5. System description

5.1. High-level Description



5.2. Raspberry Pi

This is the central hardware platform in our design. It costs about only thirty five dollars, but it is able to perform most computation tasks in our design. Raspberry Pi is a mini computer of pocket size but it is quite powerful. It supports Linux and has 8 GPIO pins, UART, I²C bus, SPI bus with two chip selects, and I²S. Generic USB keyboards and mice are also compatible with the Raspberry Pi, and this makes development much easier. The video controller is capable of standard modern TV resolutions, such as HD and Full HD. With Ethernet port, we can easily upload the data collected from different sensors to the server as well as get feedback from the server, and then process the fetched data to control various actuators to perform tasks accordingly.

5.3. Sensors

5.3.1. Microphone

We use a microphone to collect sound input from user, and transfer the audio signal to our Raspberry Pi for speech processing. Also, we need to connect the microphone to an opamp so that our device can detect soft sounds.

5.3.2. Brightness and Temperature sensor

We plan to use humidity and temperature sensors to capture the relative room humidity and temperature. The sensor uses a digital 2-wire interface and it offers high precision with excellent long term stability. The sensor collects data continuously and sends the data to the central server. While a user query our speaker for temperature or humidity, our chip converts the related sensor data into either sound or picture and sends it through an actuator (i.e. speaker or projector).

5.3.3 Raspberry Camera

We plan to use a Raspberry Pi camera to capture the images of users and perform facial recognition accordingly. It is a customized camera for Raspberry Pi. When a user uses our speaker for the first time, Raspberry Pi will take a few photos and run face detection algorithm on the photos to isolate the facial regions and set up new profile for the user. When running facial recognition, it takes the captured photo for comparing to the other photos of all previously built user profiles in our central database. A high confidence score resulted from the recognition algorithm alerts our speaker that a good user profile match is found.

5.4. Actuators

5.4.1. Speaker

Raspberry Pi gathers information about the user's query and stores the result into audio forms using cloud computing. Those audio signals are going to be transmitting to all other speakers.

5.4.2. Network & Server

After the user makes a query, the central microprocessor chip sends the recoded audio data to Google voice server, and it waits for the response. Then, the returned translated information is parsed into strings. The central chip either process the query locally (i.e. the room temperature and humidity) or search the query keywords on the internet through WiFi connection. When search result is found, it is once again passed to Google voice server for audio synthesis. Finally, the synthesized audio of search result is transmitted to user by our speaker.

6. Project management

6.1. Schedule

Week	Hardware	Software
	Tasks to Complete	
1	• Familiarize with components' datasheets • Build schematics	• Put proposal information on website
2	• Build schematics • Order components	• Web Speech API
3	• Wire components	• Web Speech API
4	• Debug	• Write device logic to test functionality
5	• Order components for second unit	• Add functionalities

6	• Wire components	• Setup server
7	• Add customized PCB	• Make sensors work
8		• Install OpenCV
9	• Include camera	• Add facial recognition
10		• Debug facial recognition

6.3. Budget

Picked Up	Raspberry Pi2 Project Kit	http://www.mcmelectronics.com/product/83-16551RK	$138.99	1	$19.99	$158.98
Picked Up	USB Microphone	http://www.amazon.com/gp/product/B001AIQGUO?ref_	$44.99	1		$44.99
Picked Up	USB Speaker	http://www.amazon.com/Altec-Lansing-IML227-Speaker	$15.99	1		$15.99
						$0.00
Picked Up	Raspberry Pi Wifi USB	http://www.adafruit.com/products/1030	$19.95	1	$3.49	$23.44
Picked Up	Raspberry Pi 5" Display	http://www.adafruit.com/products/2260	$74.95	1		$74.95
						$0.00
Picked Up	Audio Jack Speaker	http://www.amazon.com/XBOOM-Portable-Rechargeabl	$14.99	1		$14.99
						$0.00
Picked Up	Raspberry Pi 2	http://www.mcmelectronics.com/product/RASPBERRY-	$39.99	1	$9.99	$49.98
Picked Up	Raspberry Pi Wifi USB	http://www.adafruit.com/products/1030	$19.95	1	$6.97	$26.92
Picked Up	Audio Jack Speaker	http://www.amazon.com/XBOOM-Portable-Rechargeabl	$14.99	1		$14.99
Picked Up	USB Microphone	www.amazon.com/Blue-Microphones-Snowball-Conden:	$44.96	1		$44.96
						$0.00
Picked Up	Temperature Sensor	https://www.adafruit.com/product/1782	$4.95	1	$3.49	$8.44
Picked Up	Wireless Travel Router	http://www.amazon.com/gp/product/B007PTCFFW	$18.80	1		$18.80
						$0.00
Picked Up	Raspberry Pi Camera	http://www.amazon.com/gp/product/B00E1GGE40/ref=(	$26.95	1		$26.95
Picked Up	Raspberry Pi Wifi USB	http://www.adafruit.com/products/1030	$19.95	1		$19.95
Picked Up	Raspberry Pi 5" Display	http://www.adafruit.com/products/2260	$74.95	1		$74.95
Picked Up	PCB Order		$43.52	1	$3.57	$47.09
Picked Up	PCB Order		$68.52	1	$8.33	$76.85
Picked Up	MCP3008	http://www.adafruit.com/products/856	$3.75	2		$7.50
Picked Up	Photocell	https://www.sparkfun.com/products/9088	$1.50	2		$3.00
Picked Up	TMP36	https://www.sparkfun.com/products/10988	$1.50	2		$3.00
Picked Up	RESISTOR1206 (10k Ohm)	http://www.digikey.com/product-detail/en/RC1206FR-07	$0.03	10		$0.32
Picked Up	CAPACITOR1206 (0.1 uF)	http://www.digikey.com/product-detail/en/12065C104MA	$0.17	10		$1.65
Picked Up	Break away headers	https://www.sparkfun.com/products/116	$1.50	1		$1.50

6.3. Risk management

6.3.1. Design risks

According to our professor, most microphones are proximity based, so it takes some efforts to find a proper microphone sensor that can listen across a room.

6.3.2. Resource risks

Since it requires at least 2 units of our speakers to demo in a household setting, the first prototype may fail due to unpredictable hardware issues. So, we plan to implement only the hardware resources for the first unit from the beginning, and use a server to simulate the second unit for development purpose. If the first prototype is successful, then we would move on and develop more units.

7. Evaluation

To better test the accuracy of our speaker's speech recognition, we invite random testers to speak the hotword "dude" in turns, and count the number of successful hotword activation. Also, we have to fairly evaluate if the speaker authenticates a new user as "unknown" and attempts to build a new user profile for each tester. The result we collected is quite pleasing, as 4 out of 5 random testers are recognized as "unknown". Though one of the testers is improperly recognized as an another tester, we speculate this voice authentication failure is due to these two testers are both males and so have similar voice features.

Next, we invite the same testers, who have successfully built their user profiles, to repeat the hotword in turn. The result we collected is quite satisfactory, since 3 out of 5 testers are successfully authenticated. Although one tester is authenticated as “unknown”, we speculate this authentication failure is due to that this tester speaks in an drastically different tone. Considering that this is not a signal related project, we are fairly satisfied with the speaker authentication results.

8. Conclusions

8.1. Lessons Learned

The first important lesson is to always backup our software and keep notes of the hardware configuration steps. Our Raspberry Pi kernel had multiple times software panics throughout the project, and all hardware configurations on the device were lost. It took at least two hours to re-install the operating system and reconfigure everything properly for each hardware failure. Fortunately, we backup our software regularly with Git, and so we did not have to rewrite any software even with all those hardware crashes.

Another important lesson is to design a better user interface. Although we tested our device repeatedly and we are very familiar with how to use the device, we did not give enough thoughts on how to make our device a lot more user friendly such that any new user can learn to use our device quickly. So, instead of spending more time on software functionalities, we should spend time to redesign the user interface of the device to make it more accessible for common people.

The last lesson is that the product package presentation is very important. Initially, we did not have any extra aesthetic design for our device. Our speaker was just a bunch of PCB hardware with wires all over one another, so the appearance was not very attractive. Then, one of our team members suggested we could do something creative and so we decided to put our speaker in a giant Teddy bear. The final presentation was pretty good, since our speaker successfully attracted many audience in our public demo as many of them are young ladies.

8.2. What would we do differently?

We would have use the central server to do facial recognition instead of wasting computation power of our Raspberry Pi. The library we use for facial recognition is called OpenCV. It could take up to 5 hours to install on Raspberry Pi. OpenCV generally requires lots of hardware resources (memory, processors, etc.). During our experiments, it is the facial recognition algorithm that causes kernel panics. And so, we would have to reinstall OpenCV after each panic. It should run better on server, and its computation time would be shortened as well.

8.3. Future Works

If we were to proceed from our current state of development and use our device as the basis for a start-up company, we would have to implement more basic functionalities so that our speaker can do a better job serving as a virtual assistant. Furthermore, we would package our device better so that it has a more attractive appearance. Currently, the wirings of our hardware are exposed, so the speaker does not look pretty enough to be sold in open market. With a better packaging and more well-rounded software functionalities, we believe our speaker is a great deal.

9. Related Work

Our idea is inspired by Amazon Echo. Amazon Echo is designed to interact with user information, online music, news, weather, and much more. It begins listening to commands as soon as it detects the hotword Alexa. As Echo detects the hotword, its LED lights up and streams audio commands to its cloud, in which the Amazon Web Services recognize and respond to user's query. For example, Amazon Echo can also play songs and manage everyday To-Do lists.

Unlike Amazon Echo, our speaker has a variety of customized sensors and actuators. Moreover, it builds personal profiles for different users so that it authenticates users using voice and facial recognition.

10. References

I. "BeagleBone Black." Black. N.p., n.d. Web. 08 Feb. 2015. <http://beagleboard.org/BLACK>.

II. "Beaglebone Black." - DEV-12076. N.p., n.d. Web. 08 Feb. 2015. <https://www.sparkfun.com/products/12076>.

III. "Humidity and Temperature Sensor - SHT15." - COM-08227. N.p., n.d. Web. 09 Feb. 2015. <https://www.sparkfun.com/products/8227>.

IV. "Kinect." Wikipedia. Wikimedia Foundation, n.d. Web. 09 Feb. 2015. <http://en.wikipedia.org/wiki/Kinect>.

V. "Speaker - 0.5W (8 Ohm)." - COM-09151. N.p., n.d. Web. 09 Feb. 2015. <https://www.sparkfun.com/products/9151>.

VI. "AAXA P2 Pico Projector - LCoS Based Hand-held Pocket Projector - LED Micro Projector." N.p., n.d. Web. 09 Feb. 2015. <http://www.aaxatech.com/products/p2_pico_projector.htm>.

VII. "How to Choose the Right Platform: Raspberry Pi or BeagleBone Black?" Make. N.p., 25 Feb. 2014. Web. 09 Feb. 2015. <http://makezine.com/magazine/how-to-choose-the-right-platform-raspberry-pi-or-beaglebone-black/>.

VIII. "Dragon Software Developer Kits." Dragon. N.p., n.d. Web. 09 Feb. 2015. <http://www.nuance.com/for-developers/dragon/index.htm>.

IX. "T4ngo/dragonfly." GitHub. N.p., n.d. Web. 09 Feb. 2015. <https://github.com/t4ngo/dragonfly>.

X. 12, Dragon Naturallyspeaking. DATASHEET (n.d.): n. pag. Web. <http://www.nuance.com/ucmprod/groups/dragon/@web-enus/documents/webasset/nc_008811.pdf>.

XI. "ILA Voice Assistant." CMU Sphinx. N.p., n.d. Web. 09 Feb. 2015. <http://cmusphinx.sourceforge.net/>.

XII. "Amazon Echo FAQs." Amazon.com Help:. N.p., n.d. Web. 09 Feb. 2015. <http://www.amazon.com/gp/help/customer/display.html?nodeId=201602230>.

XIII. "AWS | Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting." Amazon Web Services, Inc. N.p., n.d. Web. 09 Feb. 2015. <http://aws.amazon.com/ec2/>.

XIV. "Speaker - 0.5W (8 Ohm)." - COM-09151. N.p., n.d. Web. 11 Feb. 2015. <https://www.sparkfun.com/products/9151>.

XV. "Humidity and Temperature Sensor - SHT15." - COM-08227. N.p., n.d. Web. 11 Feb. 2015. <https://www.sparkfun.com/products/8227>.

XVI. "Electret Microphone." - COM-08635. N.p., n.d. Web. 11 Feb. 2015. <https://www.sparkfun.com/products/8635>.

11. Appendices

Brainstorming Diagrams: