

Summer 2020 10/09/2020

EECe 500-Approved experince report

by: Salah el dine Mneimneh

Contents

[Table of Images 1](#_Toc51839186)

[Introduction to Company 2](#_Toc51839187)

[Background on The Project 2](#_Toc51839188)

[My Work 3](#_Toc51839189)

[Tasks Assigned 4](#_Toc51839190)

[Implementing Text-to-Speech on Robot 4](#_Toc51839191)

[Implementing NLP on Robot 6](#_Toc51839192)

[Controlling a Robot from Another Robot 10](#_Toc51839193)

[Setting Up Gazebo 11](#_Toc51839194)

[Sending Commands to Multiple Robots 12](#_Toc51839195)

[Impact of The Work 12](#_Toc51839196)

[Conclusion 13](#_Toc51839197)

[Appendix 14](#_Toc51839198)

[Software Definitions 14](#_Toc51839199)

[Alexa skills kit 14](#_Toc51839200)

[AWS lambda 14](#_Toc51839201)

[AWS IOT 14](#_Toc51839202)

[ROS 14](#_Toc51839203)

[Amazon Polly 14](#_Toc51839204)

[Amazon Lex 15](#_Toc51839205)

[Gazebo 15](#_Toc51839206)

[References 15](#_Toc51839207)

# Table of Images

[Figure 1 Architecture diagram of the initial system to communicate to the robot via Alexa 2](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839208)

[Figure 2. IAM user JSON file for tts permission 4](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839209)

[Figure 3. Code for offline python tts engine 4](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839210)

[Figure 4. function i created for generating speech from text using gtts and playing it using pygame 5](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839211)

[Figure 6. text to speech using tts ros node 6](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839212)

[Figure 5. text to speech using python engine pyttsx3 6](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839213)

[Figure 7. google speech recognition code for catching audio from microphone and printing it as text 7](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839214)

[Figure 8. say function created before to output speech from text 8](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839215)

[Figure 9. ros service call for lex node 8](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839216)

[Figure 10. aws lex console 9](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839217)

[Figure 11. Final Architecture Diagram 10](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839218)

[Figure 12. the robot in the simulation environment 11](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839219)

[Figure 13. Code for setting the robot as a MQTT client 12](file:///C:\Users\Salah\Desktop\Final%20Report.docx#_Toc51839220)

# Introduction to Company

I am completing my internship with professor Imad Elhajj from the department of electrical and computer engineering in AUB. Professor Elhajj’s research interests involve cybersecurity, instrumentation and robotics, multimedia networking, and sensor and computer networks. My internship is involved with the robotics research field, more on that in the next section. I am working under the Vision and Robotics Lab (VRL) at AUB. Established in 2006 the goal of the lab is to enhance our understanding of robotics in the fields of robotic vision, human-robot interfacing, and robotic platform design to name a few.

# Background on The Project

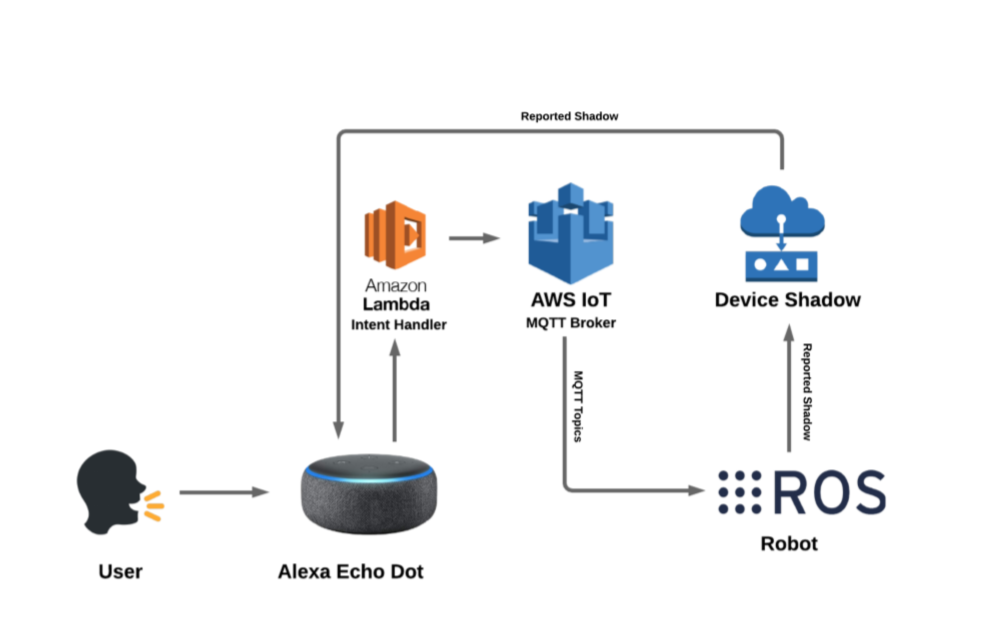
The project we are working on is controlling robots through voice commands using amazon’s Alexa. This part of the project was done previously by a graduate student (Ahmad Amine) whom I’m working with currently to add new features to the robot. The project is cloud based using Amazon web services (AWS), as well as the Alexa development console to create Alexa skills which dictate the response you get from commands. The way it works is Alexa listens to a command from the user, this command is predefined in an Alexa skill which is created in the Alexa development console. This skill is linked to back-end code in AWS lambda which has the intent handlers (an intent is what we want to do basically in this project it is moving the robot). The lambda code then communicates with an AWS IOT MQTT broker which controls sending the messages to the robot and feedback is provided through the device shadow service from AWS IOT which gives the state of the robot. This method of communication allows for a one to many communication schemes in which the project scales to handling multiple robots from one command. It also allows for asynchronous communication since through the usage of an MQTT broker we can use a pub/sub architecture in which we have publishers and subscribers on the broker sending messages and listening to them.

Figure Architecture diagram of the initial system to communicate to the robot via Alexa

# My Work

The first week involved familiarizing myself with aws services and with ros and setting up a virtual machine to run ubuntu in order to be able to use the packages and services already used in the project. I went through [ros wiki](http://wiki.ros.org/ROS/Tutorials) which had plenty of tutorials which helped me understand the project more , and I went through [alexa skills kit development tutorials](http://www.youtube.com/watch?v=CzTKDu7Qgjs) to understand creating an alexa skill and using aws lambda. My general workflow is basically researching ideas to implement the task and then learning the services needed watching tutorials to be able to implement the tasks at hand.

## Tasks Assigned

### Implementing Text-to-Speech on Robot

The first task I had was to implement text-to-speech on the robot which allows the robot to speak to Alexa to issue commands to another robot. I found plenty of packages that are able to achieve this goal the first one I used was a ROS node called tts(text to speech) which uses the aws polly amazon service which basically takes any text whether in normal or ssml format and creates an audio file from it. To do this first we create an IAM user on our aws account with the following permission file (Figure 2.). After this we just call the following 2 commands on the terminal :

Figure . IAM user JSON file for tts permission

roslaunch tts sample\_application.launch

rosrun tts voicer.py ‘text to turn to speech’

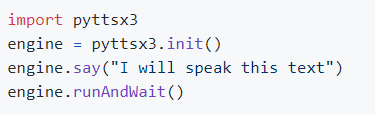
 this basically achieved the task however I faced some issues with the python version since it doesn’t match the one used in the project, and we also wanted to make the final project into an API therefore we needed to avoid commands to the terminal. Later on I found a text to speech engine running on python 3.x which allows to take a text with 3 lines of code and produce the output audio.

Figure . Code for offline python tts engine

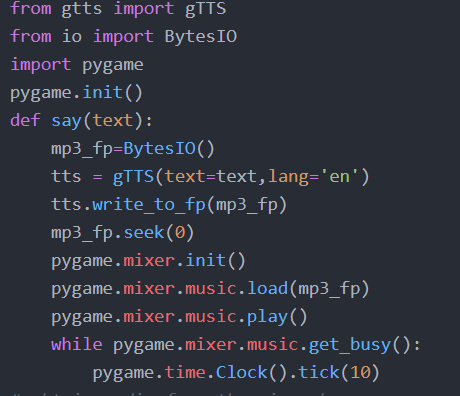
However this engine produced the output in a very robotic voice to which Alexa was only able to comprehend 75% of the time so we needed to change it into something more comprehensible by Alexa since the final goal of this task was to allow the robot to respond back to Alexa on its own creating a user free environment between them. After plenty of searching for open source packages I found a package offered by google called google text to speech or gtts for short. This package creates an mp3 file or streams the audio so in order to play the audio I had to use the pygame package audio player in python in order to play the audio generated from gtts.

Figure . function i created for generating speech from text using gtts and playing it using pygame

for every software attempt I created an architecture diagram and these are presented in the figures below

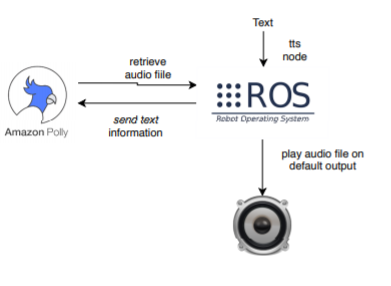
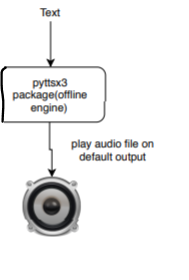


Figure . text to speech using tts ros node

Figure . text to speech using python engine pyttsx3

for the final package used however you can refer to Figure 11 for the image of the final architecture diagram which includes the text to speech packages used.

### Implementing NLP on Robot

Another task I had was implementing natural language processing(NLP) on the robot for this I also found another amazon service called amazon lex it is used to create chat bots. These chat bots have the functionality of talking in either text or voice command and then through an intent you create (similar to alexa intent) it is able to understand the utterance(words spoken by user) and send back text to respond to the user which is basically the functionality I need since I want the robot to take a voice command and be able to analyse it and understand what to do however the package I found only takes an audio file or text as input and therefore to achieve voice command I found a project on python called speech recognition which uses google’s speech recognition to get the audio from the mic and transform it to text.

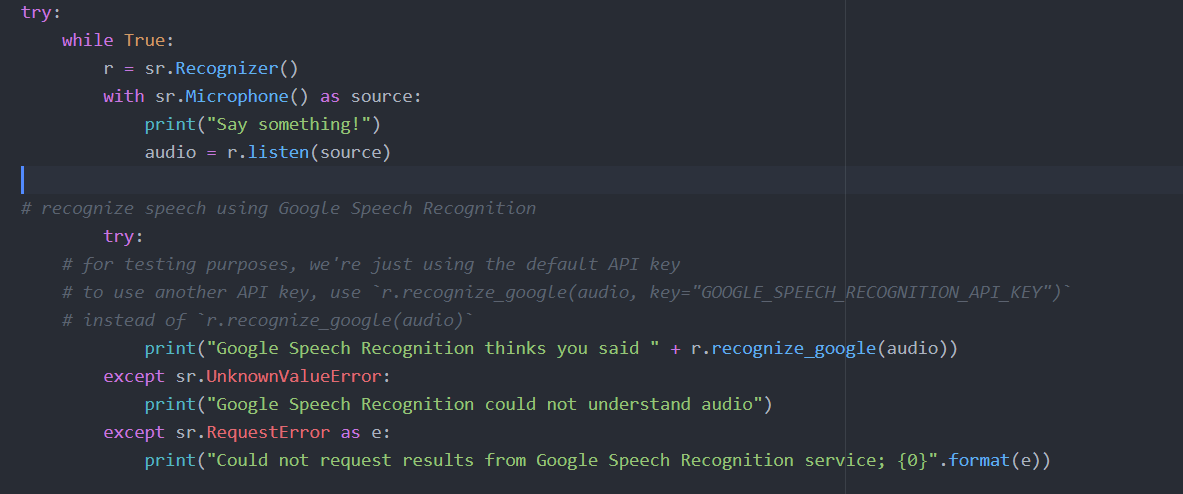


Figure . google speech recognition code for catching audio from microphone and printing it as text

Then this text is used to be sent as a ROS service call input to the amazon lex node to achieve NLP on the robot. The feedback text received from the chatbot is then sent to the text to speech package to be spoken out by the robot. This approach requires also giving the IAM user created before the following permission : AmazonLexRunBotsOnly. However since I don’t use the tts ros package in the final product so the IAM user need only have this final policy. Initially this would work only in the command line by sending a ros service call however after delving into the ros tutorials I was able to create the ros service call in python code and add it to the overall python file for text to speech and nlp purposes.

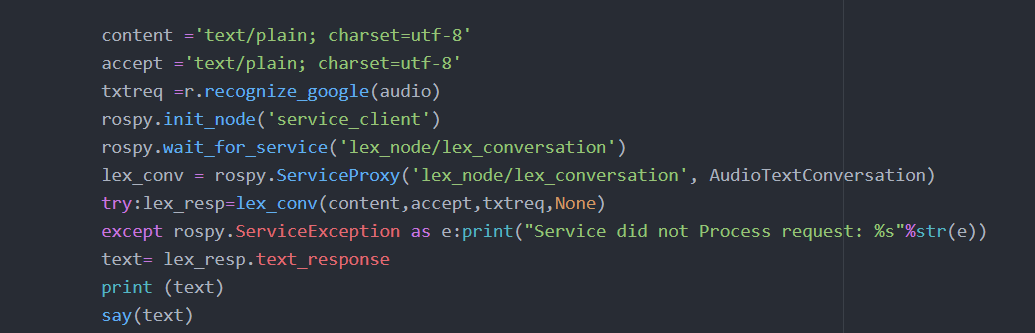
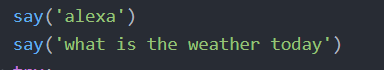
The ros service call requires 4 parameters: the content which is the type of input ,accept which is the type of output, text request which is the text to be interpreted by the lex bot, and finally audio request (not used so none in code). After the bot receives the text request it understands it similar to how Alexa interprets intents and responds accordingly. For demo purposes I created a Weather bot that will ask Alexa what the weather is using text to speech

Figure . say function created before to output speech from text

Figure . ros service call for lex node

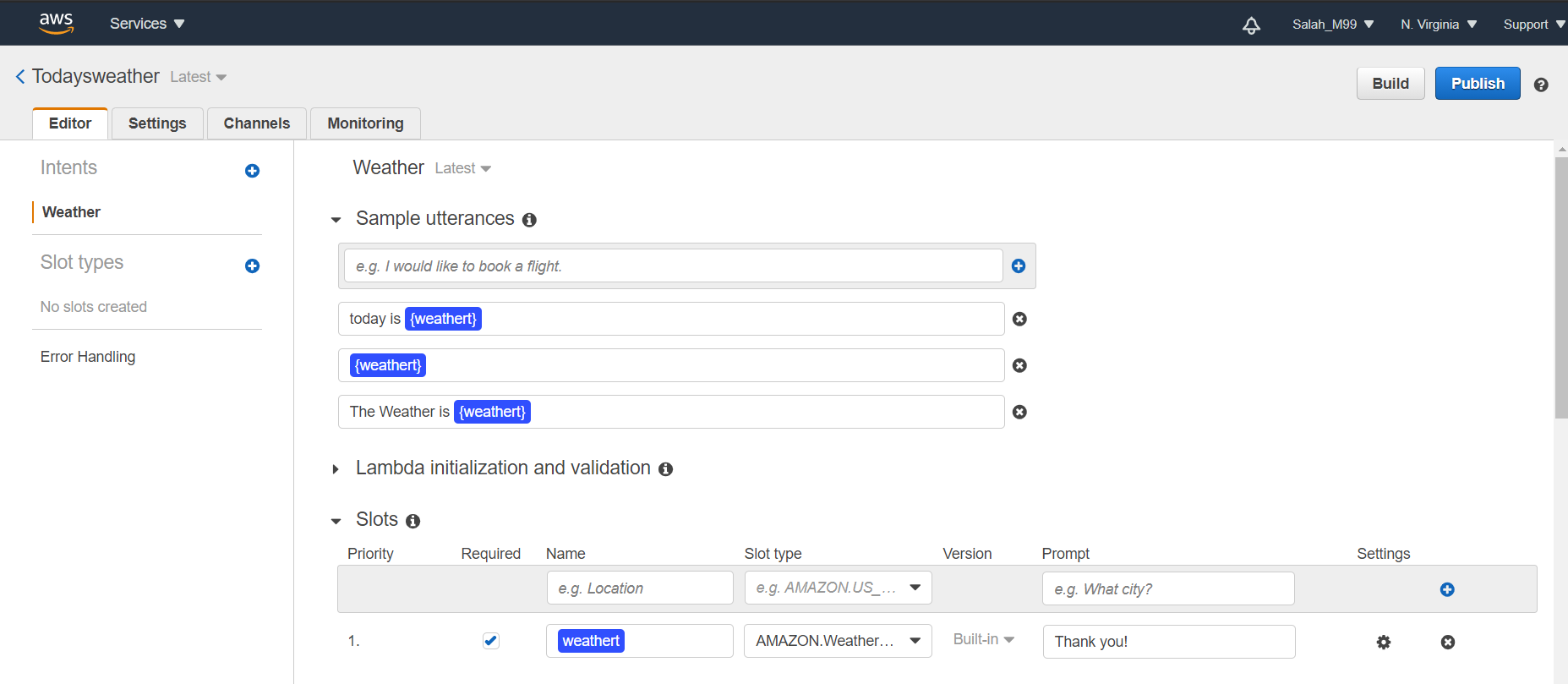
and then when Alexa responds with lets say “The Weather is sunny” the bot will reply with “Thank you!”. This interaction was programmed in the amazon lex console seen in the below image

Figure . aws lex console

where the weather slot includes all weather types uttered by Alexa such as sunny, rainy, etc…

There are two ways to get back the response either returned directly back to the user which is the method we use since it is easier to implement. The other one is through using aws lambda to define a specific end point to return to. It allows for more functionality however we didn’t require it for this application.

### Controlling a Robot from Another Robot

After implementing the NLP and the text to speech these 2 tasks combined allow us to control a robot from another robot via Alexa. The project previously was able to control the robot via Alexa through the user by sending specific commands such as “move forward”. So now we are able to make the second robot tell Alexa to control the other robot to move forward and when Alexa responds to it the amazon lex bot will understand the interaction and respond accordingly to send another command such as stopping the robot.

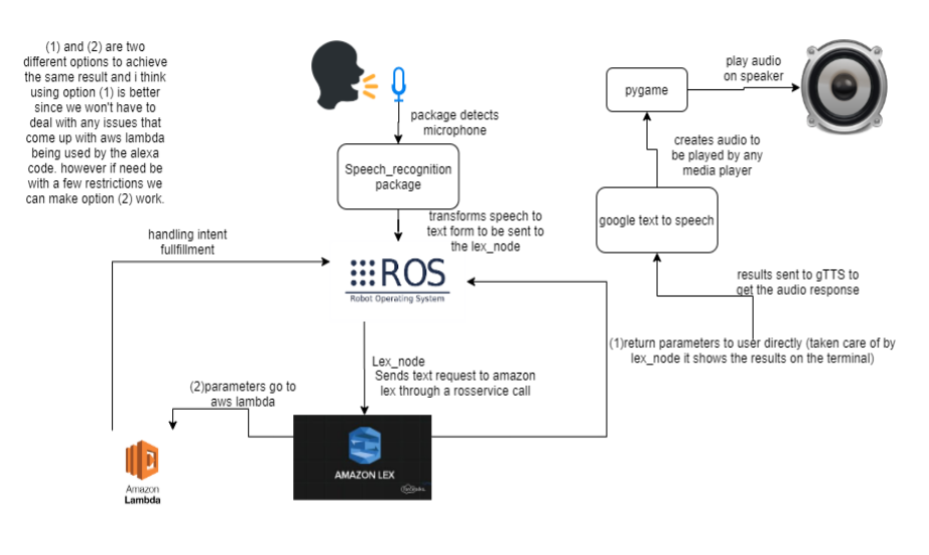


Figure . Final Architecture Diagram

### Setting Up Gazebo

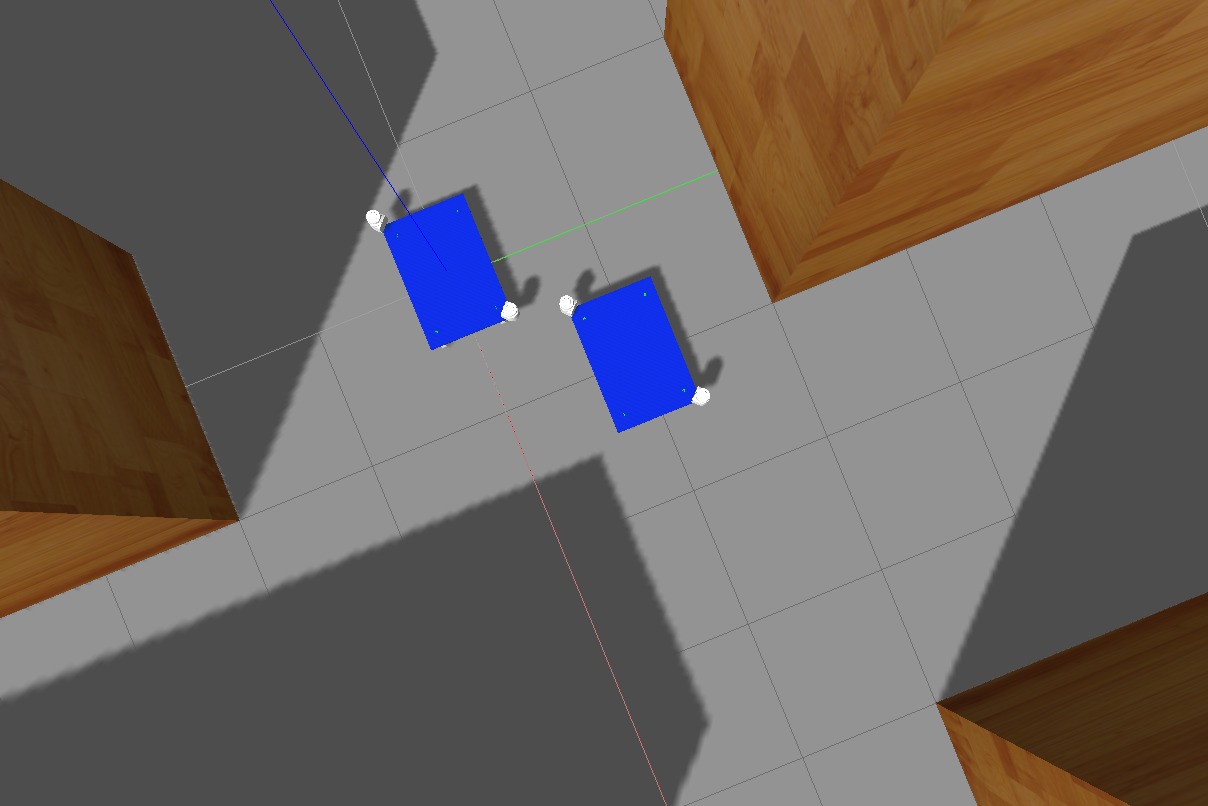
We also worked on setting the simulation environment with gazebo in order to be able to test on the robot from our machine however plenty of errors were faced in this process and it hasn’t been completed eventually since we decided not to use it on our machines and rely on the computers available in the lab already set up. Personally I couldn’t access the campus during the current situation however I was working with the graduate assistant on receiving the results from the simulation and further testing.

Figure . the robot in the simulation environment

### Sending Commands to Multiple Robots

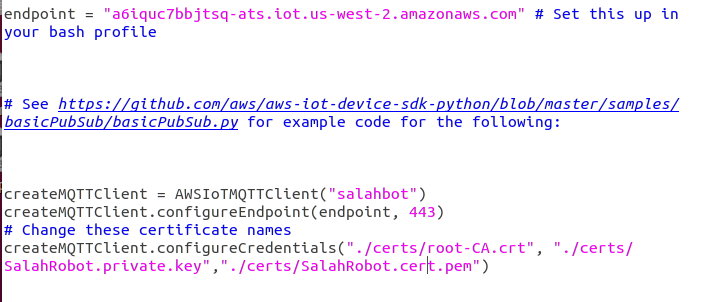
 This is achievable due to using the MQTT broker which allows to publish the data on topics which the robots can subscribe to. The process has 2 levels of security the first one being the aws account used to create the robots named things in aws iot core and the second level being the certificate files generated for each thing. So, we created 3 robots each ran on a different machine and after sending the move forward command all the robots received the command.

Figure . Code for setting the robot as a MQTT client

# Impact of The Work

From an economic standpoint, the project is based on amazon cloud services therefore it is cheaper to implement and scalable. Especially since most of the software used other than amazon services are open source so for now there is no cost of production.

This project also helps enhance human-robot interaction and furthers the integration of robots into our daily lives. This has a huge impact on society since controlling robots from other robots and voice controlling robots through Alexa allows for plenty of application from the industrial level to the household level.

# Conclusion

From this internship I gained plenty of knowledge in cloud-based programming as well as working with ROS and amazon services specifically. All of these and especially the usage of amazon services enhanced my technical knowledge and allowed me to gain experience in fields I haven’t had prior expertise in. I also was able to grow my abilities of team work specifically in using GIT for version control between myself and the graduate assistant whom I worked with on this project. This skill is extremely necessary and I believe that GIT is an essential tool for managing projects and working in a team. Along with all this technical experience I also gained a network of colleagues whom I had the pleasure to work with in this internship. I believe this internship was a great opportunity even with all the hardships faced due the current situation however it still was able to sharpen and expand my knowledge in these new fields I tackled.

# Appendix

## Software Definitions

### Alexa skills kit

The alexa skills kit is a set of services used to develop new alexa skills which allow alexa to understand a certain phrase and link it to a function on aws lambda through intents.

### AWS lambda

runs the back-end code for an alexa skills and contains the intent handlers required to send the commands to the robot.

### AWS IOT

This is a host of multiple services for monitoring ,controlling, and connecting multiple iot devices using the cloud. You create a thing (the robot for example) and this thing will have a shadow that stores its states which we can refer to for feedback.

### ROS

The robot operating system which is basically a bunch of libraries and tools and conventions that help simplify creating complex and powerful robot behavior that can be portable to multiple types of robots.

### Amazon Polly

This service takes plain text ot text in SSML format and transforms it to speech. it allows for customization of the voice and even using multiple langauges.

### Amazon Lex

This service is used for building chat bots using voice and text it provides automatic speech recognition functionalities using deep learning and natural language understanding to recognize the intent.

### Gazebo

A well-designed simulator for simulating robots in environments whether indoor or outdoor to have a better understanding in designing the robot.

# References

* <http://wiki.ros.org/ROS/Tutorials>
* <https://github.com/aws-robotics/tts-ros1>
* <https://github.com/aws-robotics/lex-ros1>
* <https://pyttsx3.readthedocs.io/>
* <https://gtts.readthedocs.io/en/latest/>
* <https://aws.amazon.com/>