eYSIP2018

Eyantra IoT Smart Assistant & IFTTT for IoT

Onkar J. Sathe
Rohit G. Rathi
Mentors : Omkar Manjrekar, Vikrant Fernandes, Deepa Avudiappan
Duration of Internship: 21/05/2018 – 06/07/2018

2018, e-Yantra Publication

eYantra IoT Smart Assistant & IFTTT for IoT

Abstract

A web or an app interface for IoT may not always be easy for people to work with, especially if they arent used to such kinds of applications. But what if you could tell someone to do things on your behalf? eYantra IoT Smart Assistant (eYISA) is inspired from the goal to bring the magic of IoT to the fingertips of users and accessible over voice commands. In AWS IoT writing rules is hard and not that much intuitive you also need to know SQL (for AWS IoT Rule Engine). And its di fficult to select devices over which you want to write rules. A simple GUI needed to write rules to invoke the actions on devices.

Completion status

The project is successfully completed, all deliverables are achieved. We have successfully created Smart Assistant using dialogflow engine which able to perform all the frequent queries that take place on IoT platform.
And also we have devloped flexible block based GUI for IFTTT rules.

Hardware parts

- ESP DHT sensor
- 5 day workshop kit
- For connection diagram and setup refer this repository Interfacing with AWS-IoT

Software used

- Dialogflow
- Mongoose OS
- PageKite or ngrok
- Cloud Functions Local Emulator
- blockly

Assembly of hardware

For circuit diagram and steps of assembly of hardware check this repository Interfacing with AWS-IoT

Setup Guide

To start follow this instruction of to install iot-platform and setup for AWS-IoT Installation Guide

Step 1

After completing setup of iot-platform and AWS-IoT then run server of iot-platform

Step 2 - For Development Setup

Configure your PageKite account (its free of one month) then follow this quickstart.

1. Tunnel iot-platform server to public ip using this command

```
pagekite.py 8002 iotplatform-eyantra.pagekite.me
```

2. To deploy webhook function for fullfillment of dialogflow in Cloud Functions Local Emulator follow this guide
3. Tunnel webhook function deployed in Cloud Functions Local Emulator to public ip using this command

```
pagekite.py 8010 webhook-eyantra.pagekite.me
```

4. To setup Dialogflow follow this guide upto 4th instruction.
5. Then after configuring all setting of dialogflow you will get Google Project Id in setting of your Agent.
6. Then enter the fullfillment URL as

```
https://webhook-eyantra.pagekite.me/<YOUR_PROJECT_ID>/<YOUR_REGION>/eYantraWebhook
```

Step 2 - For Production Setup

1. It is assumed that iot-platform server is hosted on cloud server like EC2(Amazon).
2.To deploy webhook function for fullfillment of dialogflow in Google Cloud Functions follow this guide 3. Then enter the fullfillment URL as

```
https://<YOUR_REGION>-<YOUR_PROJECT_ID>.cloudfunctions.net/eYantraWebhook
```

Software and Code

Github Repository with code and documentation: Link

Webhook(fulfillment) program:
JavaScript program that recieves user query data from DialogFlow engine, calls corresponding rest APIs of iot-platform and returns the response to DialogFlow.

natural-cron readableToCron.js:
JavaScript Library to convert normal English phrases into "Cron expressions". Visit here for more details.

Use and Demo

1. Connect the hardware device to power supply and make sure it is switched on.

2. Verify that device has active internet connection.

3. Make sure all the servers are started and working as explained in installation guide.

4. Use the assistant(Web assistant on iot-platform, Google actions simulator, DialogFlow simulator or Google assistant on handheld devie) to fire queries.

Future Work

- After increasing the e fficiency of the assistant through more training, especially working with Hindi phrases then it can be used anywhere
- Assistants can be made dynamic enough to adapt to other APIs, So APIs are accessible over Natural Language of users even in form of text or voice.

Bug report and Challenges

Efficiency of Assistant:
Though efficiency of assistant has become quite good over the course of development, still the efficiency can be improved with more training.

Efficiency of natural-cron.js:
Building a program that will convert english phrases to desired cron expressions was a major challenge. Efficiency of natural-cron.js library can be consistantly improved over the course of time to handle complex phrases.

Challenges:

- Implementing OAuth2 token based authentication for login via Google Assistant
- Making the assistant dynamic to handle all devices and sensors
- Improving accuracy of dialogflow agent
- Implementing save and restore for blockly blocks

Bibliography

[1] Justin Jose, *Building Chatbots - A comparison of Rasa-NLU and Dialogflow*, https://www.linkedin.com/pulse/building-chatbots-comparison-rasa-nlu-dialogflow-justin-jose/

[2] DialogFlow, *Reference documentation for Google's DialogFlow engine*, https://dialogflow.com/docs/getting-started

[3] Blockly, *Reference documentation for Google's Blockly library*, https://developers.google.com/blockly/reference/overview