Artificial Intelligence and Robotics Driven Moving Telescope for Astronomy Observations

Mr. Fakruddin Mohammed

Contents

[1 Motivation 4](#_Toc9658749)

[2 Problem Statement 4](#_Toc9658750)

[3 Technology 4](#_Toc9658751)

[3.1 Hardware 4](#_Toc9658752)

[3.2 Software 4](#_Toc9658753)

[4 Architecture 5](#_Toc9658754)

[5 Implementation 5](#_Toc9658755)

[5.1 Google Login 5](#_Toc9658756)

[5.2 Dialogflow Project Setup 6](#_Toc9658757)

[5.2.1 Create Agent 7](#_Toc9658758)

[5.2.2 Define Intentions 7](#_Toc9658759)

[5.2.3 Define Entities 8](#_Toc9658760)

[5.2.4 Configure Webhooks 8](#_Toc9658761)

[5.2.5 Testing 8](#_Toc9658762)

[5.3 Google Cloud Function Project Setup 9](#_Toc9658763)

[5.3.1 Create Project 9](#_Toc9658764)

[5.3.2 Install Google Cloud Platform SDK 11](#_Toc9658765)

[5.3.3 Initialize Local Laptop & Authenticate 12](#_Toc9658766)

[5.3.4 Function 12](#_Toc9658767)

[5.3.5 Deployment 12](#_Toc9658768)

[5.3.6 Console Admin to View Log Files 13](#_Toc9658769)

[5.4 Firebase Project Setup 14](#_Toc9658770)

[5.4.1 Create Project 14](#_Toc9658771)

[5.4.2 Install Firebase SDK 14](#_Toc9658772)

[5.4.3 Create Database 14](#_Toc9658773)

[5.4.4 Service Account and Secret Key 16](#_Toc9658774)

[5.4.5 Test 17](#_Toc9658775)

[5.5 Robotic Car 18](#_Toc9658776)

[5.5.1 Parts 18](#_Toc9658777)

[5.5.2 Wiring Diagram 19](#_Toc9658778)

[5.5.3 Python – Driver Controller Code. 19](#_Toc9658779)

[6 Results & Demo 20](#_Toc9658780)

[6.1 Drive the Car 20](#_Toc9658781)

[6.2 Celestial Object Details 20](#_Toc9658782)

[6.3 Sky Maps 22](#_Toc9658783)

[7 Conclusions 25](#_Toc9658784)

[8 Next Steps 25](#_Toc9658785)

[9 References 25](#_Toc9658786)

[10 YouTube Tutorial URLs 26](#_Toc9658787)

[10.1 Summary – 2 Minutes Video 26](#_Toc9658788)

[10.2 Girhub Repository 26](#_Toc9658789)

# Motivation

It was April 2019, peak Indian summer, I took my family to a 3 nights astronomy & night sky star gazing holidays in remote holiday resort in the Kavalur forest region of Tamil Nadu state. We stayed in a small holiday resort which is very popular amongst keen astronomers because the resort has a powerful electronic telescope for night sky star gazing. For first time my life I seen all three big planets: MARS, Jupiter and Saturn. It was a thrilling experience to me and to my family which spurred interest in the Astronomy.

# Problem Statement

The telescope at holiday resort was electronic; which means, once a planet position is locked in Telescope eyepiece, both telescope body and planets movement is synchronised and for hours that planet is visible. But on the last night of our holiday trip there was a power cut in the holiday resort, therefore, we had to operate the telescope in manual/mechanical mode which was very painful as : (a) spotting the celestial object requires precision calculations and (b) the celestial objects were moving at high very speed and keeping them in telescope view requires constant adjustments of the eyepiece and focus tube. This is when I thought wouldn't be nice if we had a moving telescope which moves and keep tracks of celestial object mechanically. Here is the small prototype of what I dreamt that night.

As a prototype I built a small robot car which has GPS, Raspberry PI mounted on it and it listens for voice commands received from Google Assistant and serves: (a) the details of celestial object, (b) gives the co-ordinates of celestial object and (c) the car position itself at a given GPS location in the direction of the celestial object.

# Technology

## Hardware

The following hardware components are used to build the robotic model car.

* 1/18th Two Wheeled Drive Model Car
* Raspberry Pi 3B+ Controller
* GPS
* L298N – Motor Driver
* Connecting wires, batteries, soldering kits and various tools

## Software

The following software platforms & libraries are used for this implementation.

Python System Libraries

* os
* requests
* beatifulsoup

Dialoglfow (NLP)

* A online plorform to build & train the Natural Langauge Processing (NLP) models.

GCP – Functions (python backend)

* google-cloud-sdk
* gcloud

Firebase Database

* firbase
* firebase-admin
* firebase-sdk
* python\_jwt
* sseclient

# Architecture

The end to end architecture diagram is shown below. The end-to-end architecture, data flow and user interaction is described below.

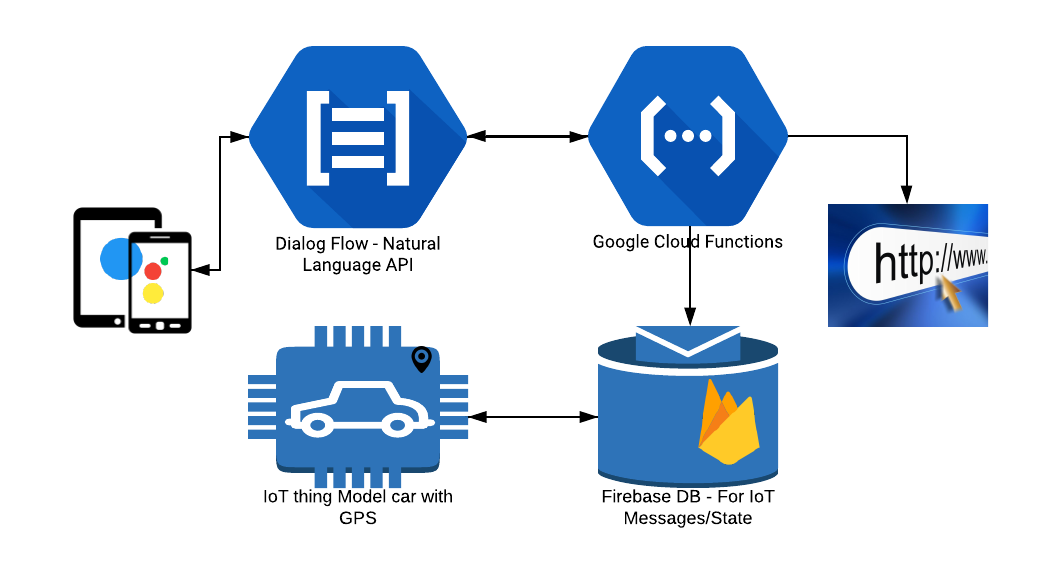
**Google Assistant:** User speaks into the Google Assistant application installed on Android/IOS enabled device. The user can ask the platform for night sky maps, celestial object details or driving the car remotely.

**Dialogflow Platform**: Receives the user speech transmitted by Google Assistant app, then interprets the speech and routes the extracted entities (in programming terms variables) from the speech to the backend function implemented and deployed on Google Cloud Platform.

**Google Cloud Platform (GCP)**: The heart of the python code doing implementing the core functionality is deployed on the Google Cloud Platform. If the user requests for sky map or celestial object details then it web scrapes for the information and sends the response back to the dialogflow and subsequently to the google assistant app. On the other hand, if the user has asked for controlling the car navigation, then requested car directional movement state is written to the Firebase database.

**Firebase**: is a real-time database provided by Google to build IOT applications. The functional deployed on GCP writes the directional state of the model car to the Firebase DB, which is then read by the Raspberry-pi controller installed on top of the model car to move the car in that direction. The model car will also be writing the current vehicle’s GPS locations to the Firebase DB, so that GCP can read the GPS co-ordinates and respond to the user with the night sky maps of the current location.

**HTTP Websites: The function deployed on GCP platform web scrapes for sky maps (sky, horizon and telescopic) and for celestial object details and sends the response back to the user.**



# Implementation

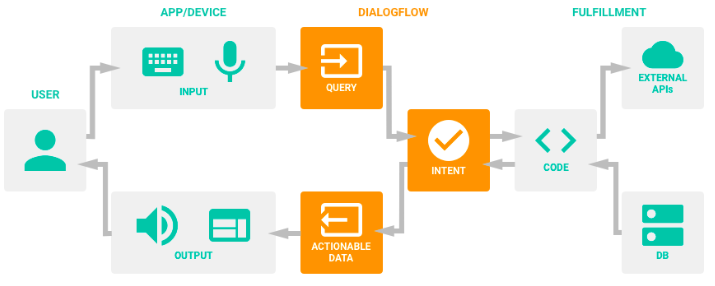
## Google Login

## Dialogflow Project Setup

Dialogflow is a technology backed by the Natural Language Processing Deep Learning models running on google infrastructure. Dialogflow enables the user to intereacts with the final product in most humanly conversation. The block diagram of the Dialogflow conversation and how it fulfils the user request is shown below.

In one sentence, it can be summarised as follows:

*The user speaks, the dialogflow backed by the Google’s Deep Learning NLP model interprets the speech, if requires talks to the backend system and responds to the user. In essence it adds human voice to the final product.*



The sub-blocks of the Diaglflow platform are as follows:

Agent: It is essentially a name of the application or module trying to achieve some objective.

Intent: When user interacts with the product, he has some aim or purpose for which the user is seeking information or wants to act on. The aim or purpose or action is an intention. The examples of intentions are: drive the car, where my order, show me the image of mars etc.

Entity: The dialogflow tries to interprets what the user is asking for so it can respond is more useful way. The interpretation of speech into piece of data are called entities. For example: drive the car forward, here the ‘forward’ is an entity; similarly switch on the light, here ‘on’ is an entity.

Fulfilment: If the user request is complex where it needs to be looked into for information in the backend databases, or compute something or do some custom work, then such user requests are routed to the piece of code that fulfils the user request. The custom code written to fulfil the user request is called fulfilment.

Context: helps the Google Assitant to have human like conversation with the user. Diagflow uses the context to remember what the user has asked for and replying in the context to the end users.

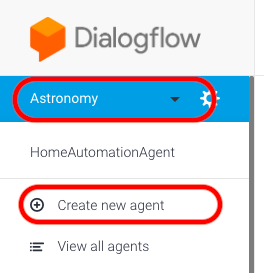
Integration: once the agent, intents and fulfilments are defined on the Dialogflow application, then it can be deployed to number of social media platform such as google assitant, facebook, twitter, slack etc.

Lets see below how the intents, entities, fulfilments are create on the Dialogflow platform.

**Note:** You can skip the sections 5.2.1 to 5.2.5 by simply importing the ‘Astronomy.zip’ into the dialogflow which will create the agent, intentions, entities and fulfilment webhooks.

### Create Agent

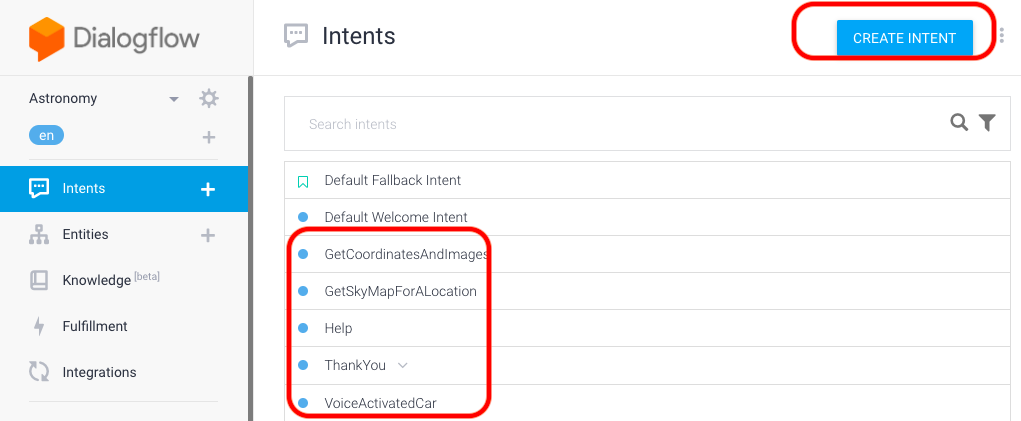
* Goto : <https://dialogflow.com/>
* Sign-in with your gmail account, and “agree” to the terms & conditions.
* Click on ‘Goto Console’ (<https://console.dialogflow.com>)
* The first step is to create agent and call is as “Astronomy”



### Define Intentions

The next step is to define the intentions and click on Create Intent.

For our application, the following intentions are created.

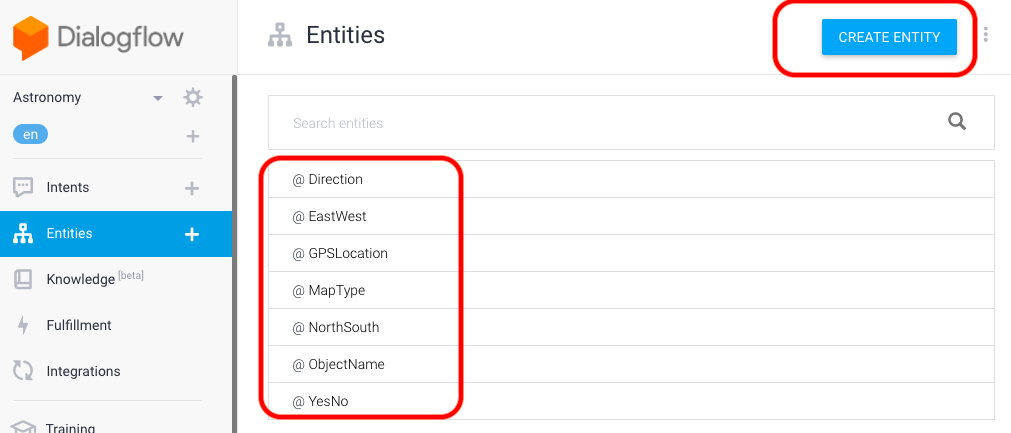


### Define Entities

Entities are like data variables where the user speech is mapped to the defined entities so that backend end code can extract or take the correct action.

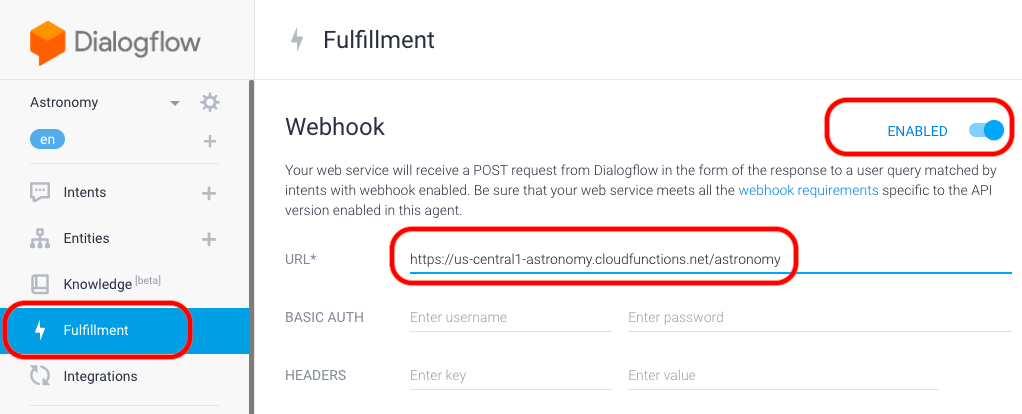
Click on entities button to create the new entities

For our application, the following entities are created.



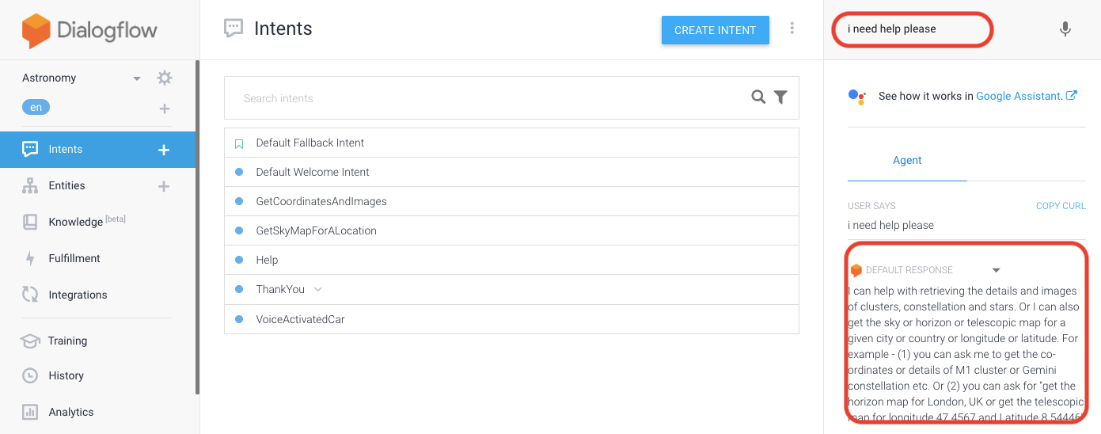
### Configure Webhooks

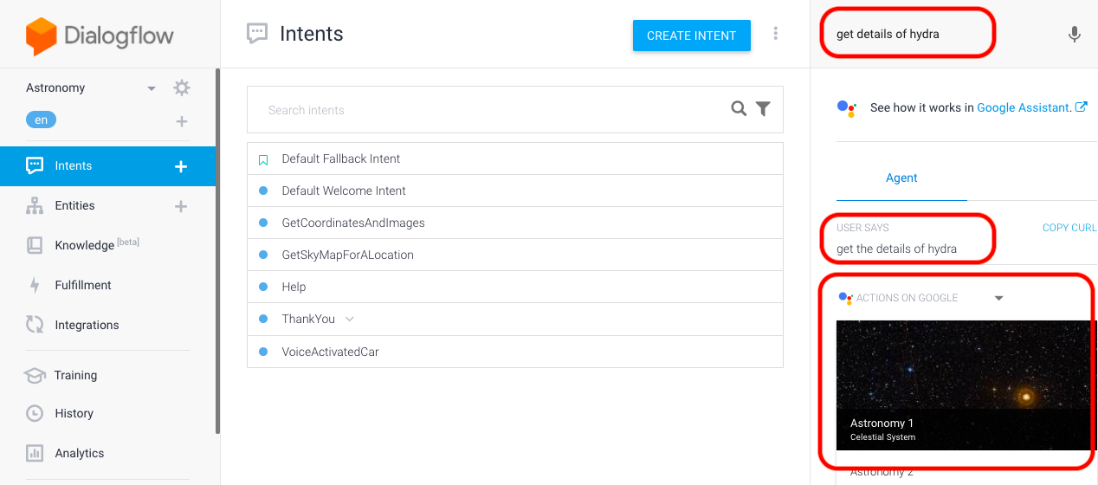
The next step is defining the fulfilment web URL where the backend code is running. The URL it is being referred is coming from the Google Cloud Platform (GCP) which is talked in more detail on how to get this URL.



### Testing

There are multiple options to test intentions but one of the simplest and easiest way is to use the simulation that comes with the dialogflow as shown below.





## Google Cloud Function Project Setup

**Google Cloud Platform (GCP)**, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search and YouTube. Alongside a set of management tools, it provides a series of modular cloud services including computing, data storage, data analytics and machine learning. Registration requires a credit card or bank account details.

For this project I am going to leverage the Google Cloud Platform (GCP), which provides serverless computing environment so I do not have worry on getting the server up & running, the network and security issues. Instead my focus will be on writing a simple python google cloud function (GCF) deploy on to the GCP platform.

**Google Cloud Functions (GCF)** are small pieces of code that execute in an event-driven manner. They do one small thing very efficiently in reaction to a trigger — usually an HTTP request. The neat thing is you manage zero infrastructure and usually only pay for the execution of your function and a few seconds of compute. You may have heard of other Functions-as-a-Service offerings such as AWS Lambda

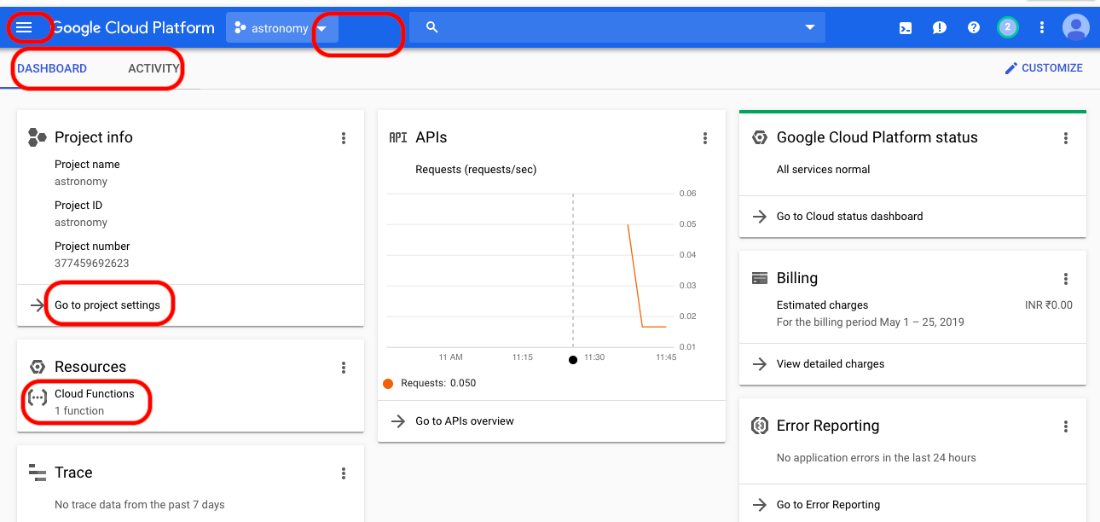
Let’s go through the step to define the google cloud function, deploy and get the HTTPS url so that it can be used with the Dialogflow to fulfil the user requests.

### Create Project

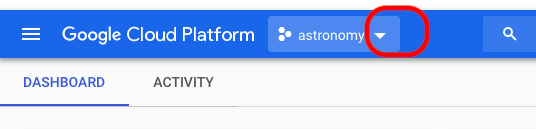
First you need to sign-up using the same google email account as it was used with Dialogflow.

* Goto: <https://cloud.google.com/free/>
* Sign-in using the email google account
* Once you login, your landing page might looks like the below as the google is constantly changing this home page by adding more and more features. But conceptually it should look the same

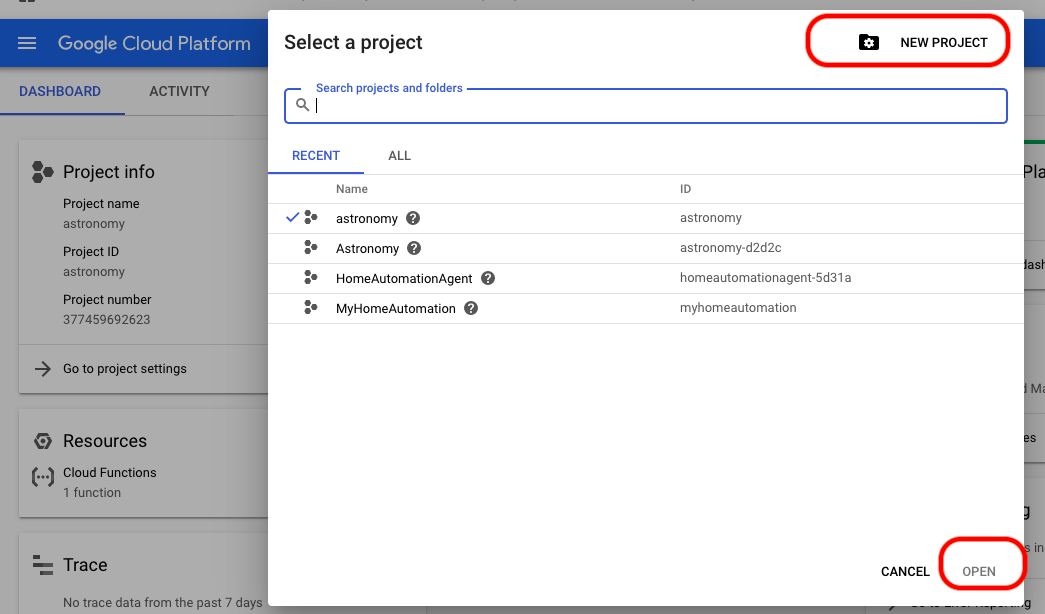
https://console.cloud.google.com/home/



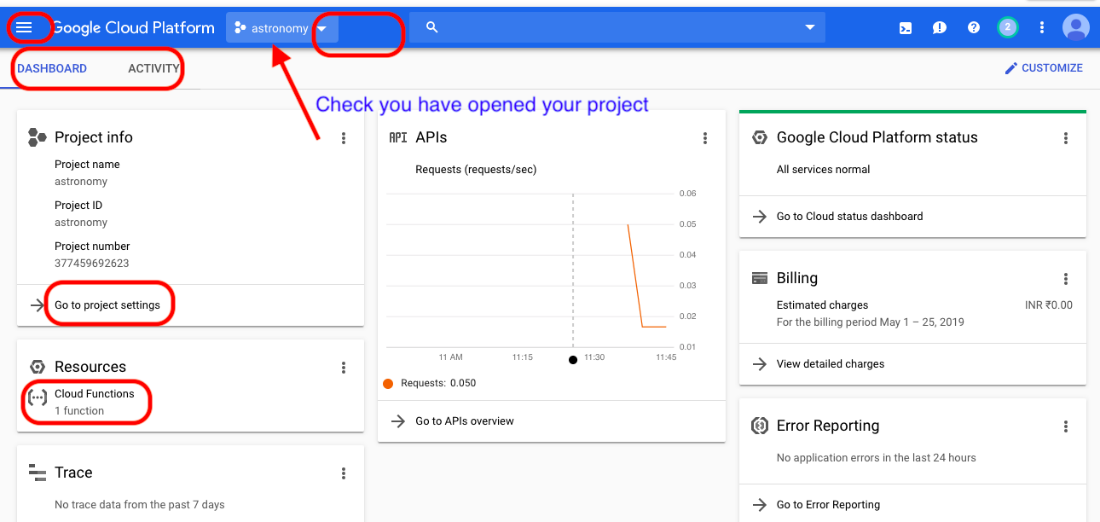
* Click on the projects list drop down menu as shown below.



* It will open the Create a project overlay as shown below and click on new project and name it as ‘astronomy’



Once the project is created successfully, it will present the project dashboard as shown below. Ensure that your project name is listed as shown below.



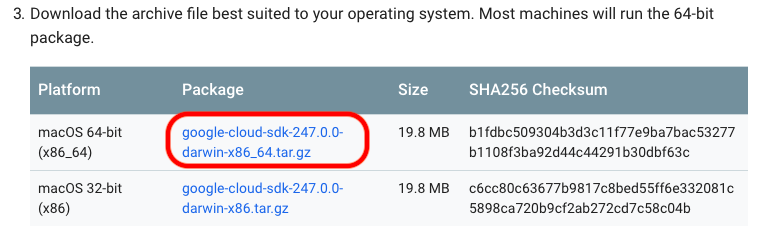
### Install Google Cloud Platform SDK

Now the next step is, installing the google cloud platform SDK. Follow the steps described below.

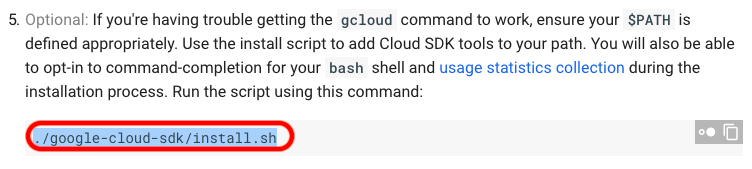
* Goto: <https://cloud.google.com/sdk/>
* Choose the platform (windows or mac)
* Since mine is mac computer, I chose mac and it takes me to the following page.

<https://cloud.google.com/sdk/docs/quickstart-macos>

* And scroll down below, you will see the download files. Click on



* Copy the download file and extract into a folder called: ‘google-cloud-sdk'
* Change directory into the directory where you extracted the downloaded the zip file. You see a directory called ‘google-cloud-sdk' and then run the following install command (shown in the image below).



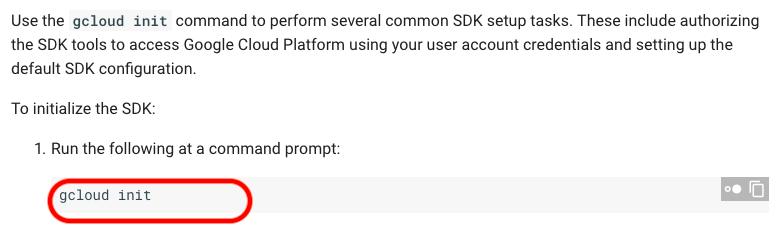
./google-cloud-sdk/install.sh or

./install.sh

### Initialize Local Laptop & Authenticate

Now the next step is initialising the project on your local laptop so that we can write a function and deploy to the GCP.

* Run the following init command - ‘gcloud init’

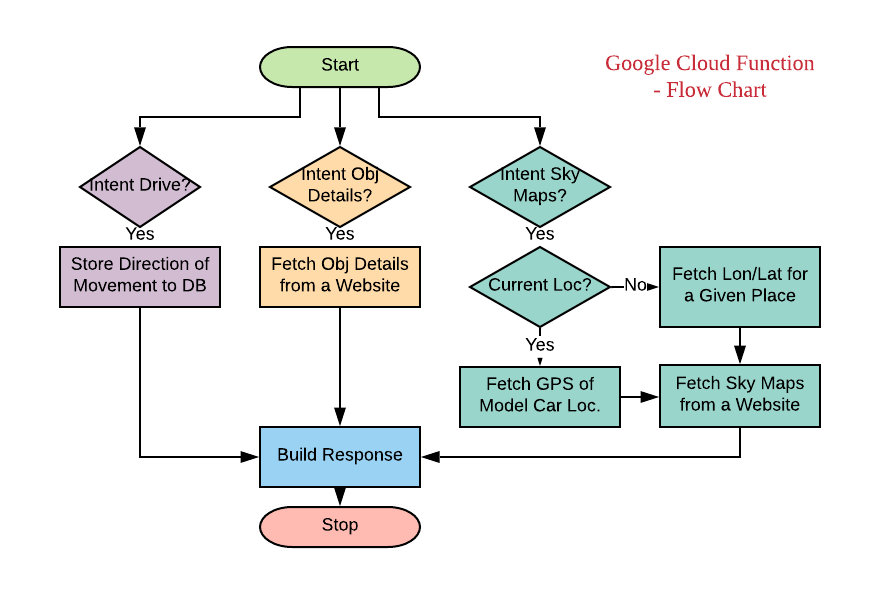


* The above action will lanuch a browser and it will ask you to select the gmail account. It will also show you a key, which you will have to copy and paste in the terminal window. This step is essentially authenticating and preparing your laptop for project deplyment.

### Function

Now create a folder with the same name as project name you created in step 5.3.1. In our case the project name is ‘astronomy’, therefore, I created a folder called ‘astronomy.

* Change directory into ‘astronomy’ folder and copy the following two files
  + main.py
  + requirements.txt
* The main.py is the file which has the core programming logic which fetches the data from the various URL and builds the response messages back to the Dialogflow. The ‘requirements.txt’ file has list of python libraries which our project is dependent on.
* The flowchart describing the programming logic is shown below.

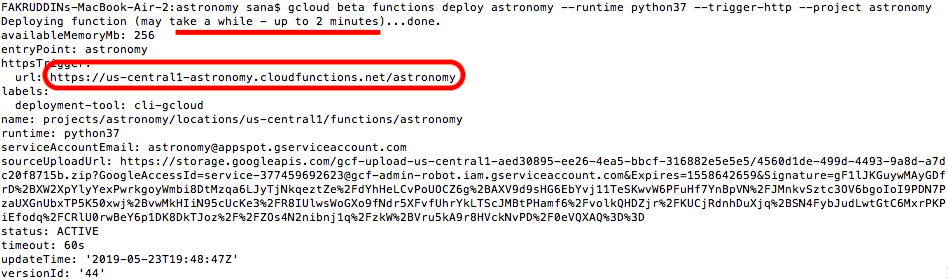


### Deployment

* Run the following command to deploy the main.py to the google cloud platform.

gcloud beta functions deploy astronomy --runtime python37 --trigger-http --project astronomy

* Once you run the above command, your terminal output should like the below.



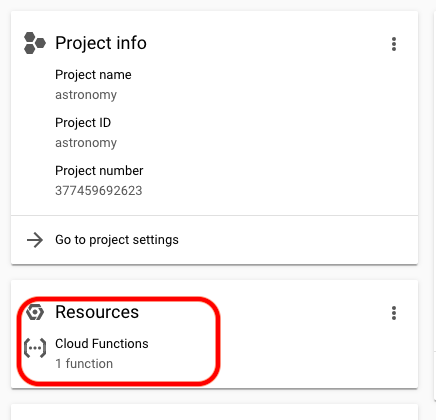
* If the deployment is successful, then copy the URL from terminal output and goto the Dialogflow fulfilment section 5.2.4 and paste the URL for webhook.

### Console Admin to View Log Files

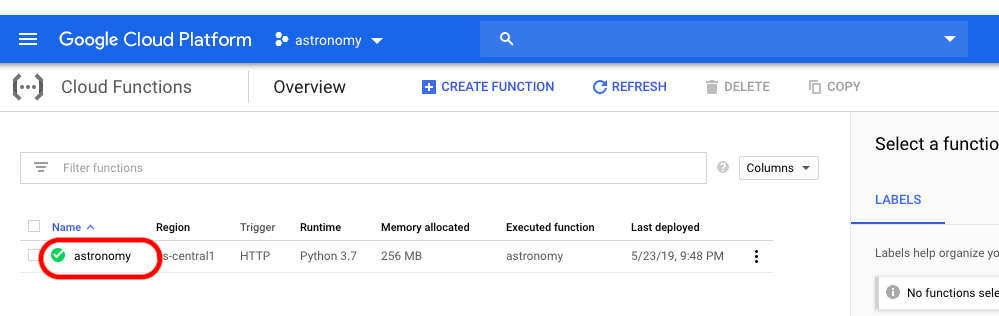
Once deployment is done and to view the log files, you can go back to the GCP console home page

<https://console.cloud.google.com/home/dashboard?project=astronomy>

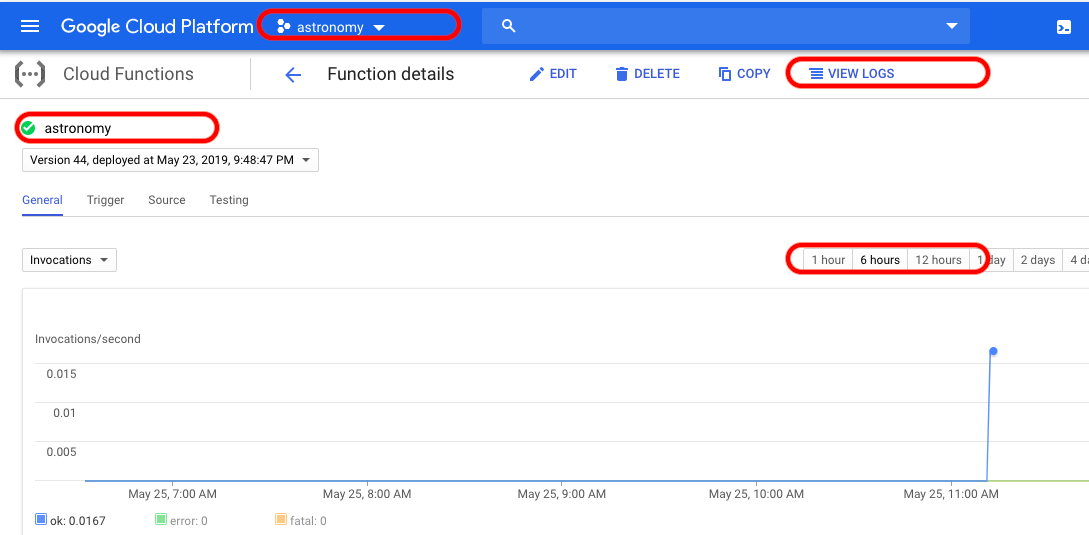
* Click on cloud functions



* This will take you the list functions list page, click on the ‘astronomy’ function to goto the logs page.



Click on the function name to open the log files.



## Firebase Project Setup

### Create Project

### Install Firebase SDK

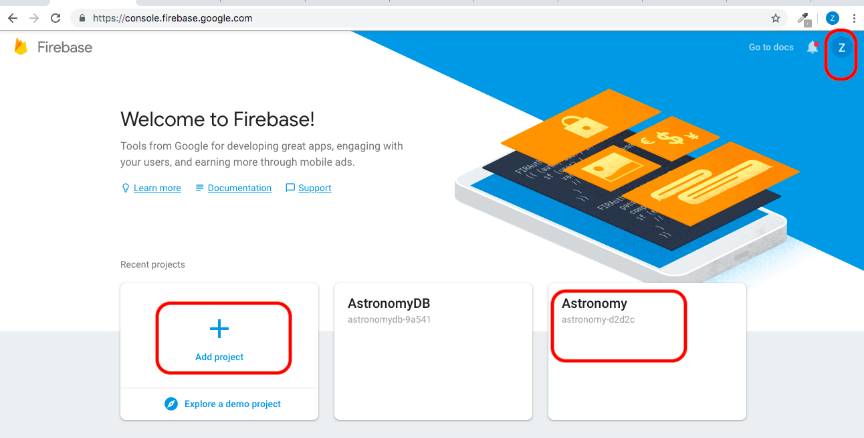
Follow the steps described below to install the Firebase SDK by running the following commands.

* Run the following command
* pip install firebase
* pip install firebase-admin
* pip install python\_jwt
* pip install gcloud
* pip install sseclient

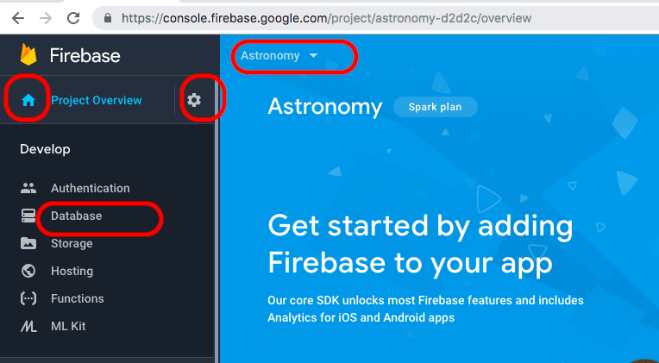
### Create Database

The first step in creating the firebase database is signing into the firebase console and creating project. Please follow the steps described below.

* Goto : <https://console.firebase.google.com/>
* Signin with using the same google-email account which you used with the Dialogflow and Google Cloud Platform.
* Once you sign-in, your firebase console login page should look like the below.

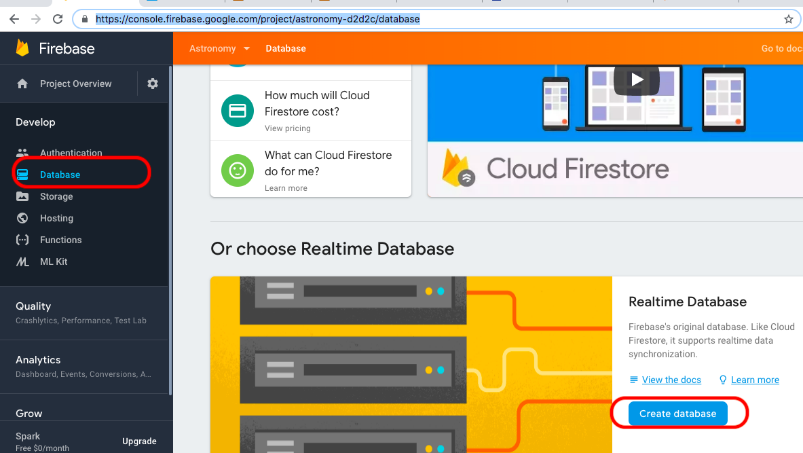


* Click on ‘Add Project’ to create a project called ‘astronomy’.
* Once you created page, the project home page should look like the below.

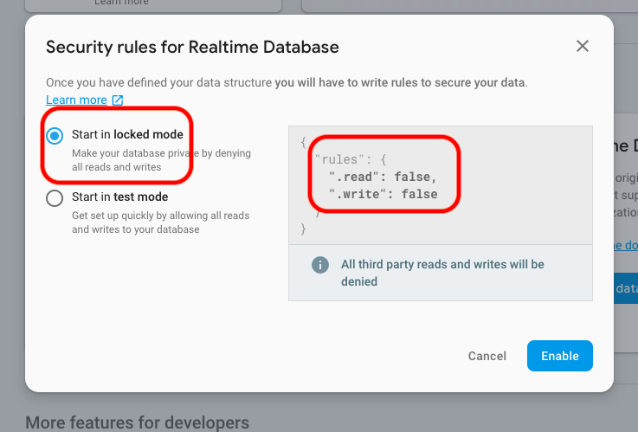


* Create a real-time database, by clicking on the ‘Databases’ and scrolling down on the page below.

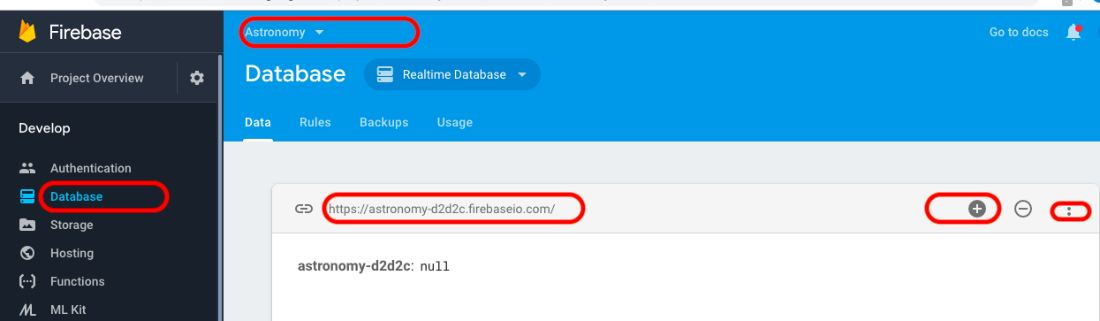
<https://console.firebase.google.com/project/astronomy-d2d2c/database>



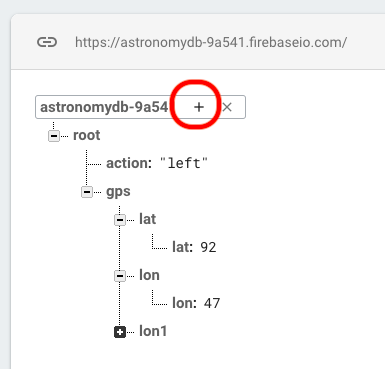
* Once you on ‘Create database’, it will show you a pop-up window asking whether to create DB in secure mode or test mode. You can choose either mode, but please be aware that if it is test mode then anyone can read/write database so I recommend choosing the secured/locked mode.



* Once you click on ‘Enable’ button, you will be taken to the empty database page as shown below.



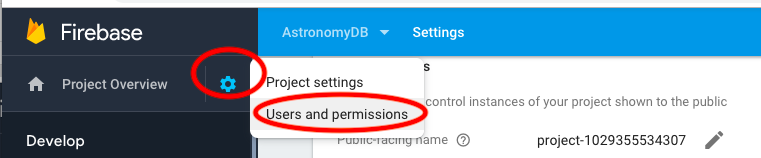
* Create a root element by clicking on the + sign and subsequently create the following tree structure database.



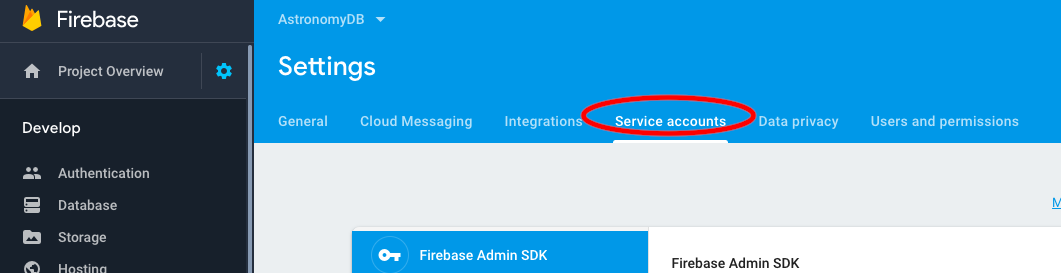
### Service Account and Secret Key

Now that you have created a database, the next step is to create a service account and a secret key so that the google cloud function can talk to the firebase database.

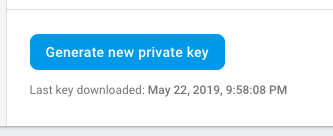
* Click on project settings icon

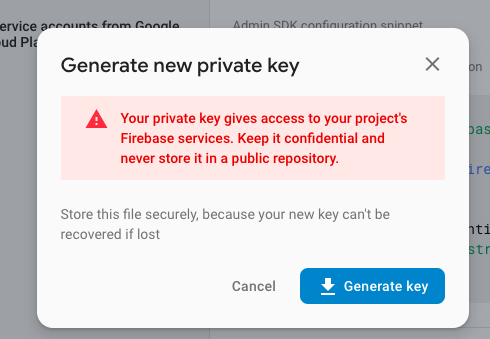


* Click on Service Accounts tab and scroll down to see the generate new private keu



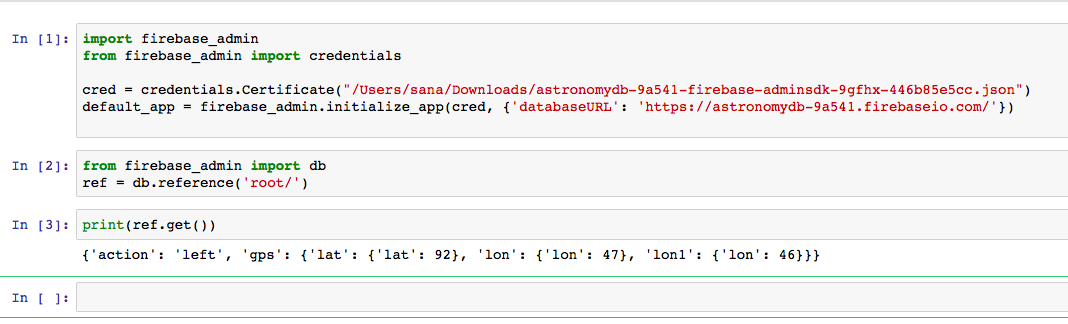
* Click on generate private key, this action will store a json file on your computer. This json key file will be used as authentication to connect to the firebase database.





### Test

* To test the firbase database connectivity, you can run the following command or script on jupyter notebook and if you see the output as shown then the firebase installation is successful.



* Code to test the firebase connectivity.

*import firebase\_admin*

*from firebase\_admin import credentials*

*cred = credentials.Certificate("/Users/sana/Downloads/astronomydb-9a541-firebase-adminsdk-9gfhx-446b85e5cc.json")*

*default\_app = firebase\_admin.initialize\_app(cred, {'databaseURL': '*[*https://astronomydb-9a541.firebaseio.com/*](https://astronomydb-9a541.firebaseio.com/)*'})*

*from firebase\_admin import db*

*ref = db.reference('root/')*

*print(ref.get())*

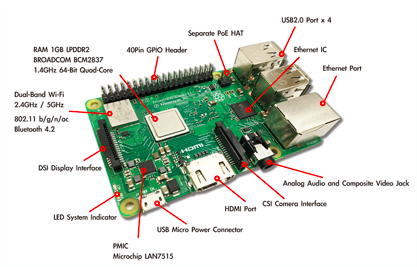
## Robotic Car

### Parts

The following hardware components are used to build the model robotic car.

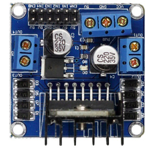
#### Raspberry Pi 3B+

Raspberry Pi small size computer which has its OS (Operating System) called Raspbian. It can be used to do multiple function simultaneously. It has 1.2 Ghz quad core ARM cortex A53 and RAM 1 GB. It requires power upto to 5V, 2.5 amp which is negligible with respect to PC [4]. The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins that allows us to control electronic components for physical computing and explore the Internet of Things (IoT).



#### Motor Driver

The L298N is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver de-signed to accept standard TTL logic level sand drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the in-put signals. The emitters of the lower transistors of each bridge are connected and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional Supply input is provided so that the logic works at a lower voltage.



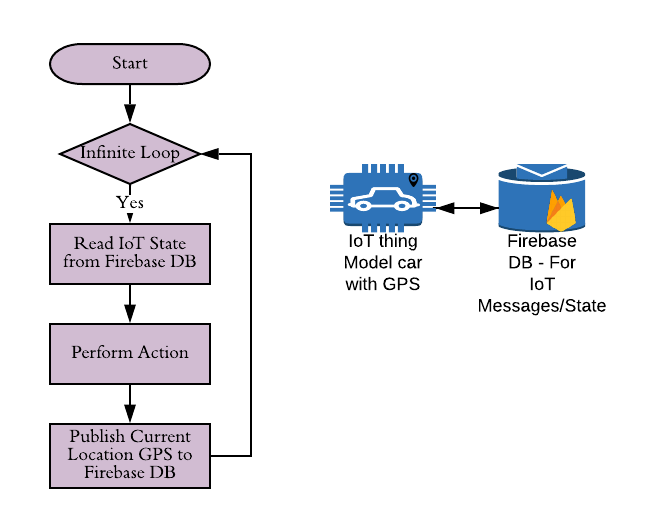
### Wiring Diagram

…...

### Python – Driver Controller Code.

The flowchart of the python code running on Raspberry pi controller is shown below. As you see, the code is continuously running in the infinite look and polling the firebase database field (root/action). Depending on the action, the controller issues or sets the digital pins on the Raspberry pi so that the car can move in that specified direction.

At the same time, the Raspberry Pi controller is also measuring the GPS co-ordinates using the GPS sensor and writing the values to the firebase database under the following nodes: root/gps/lon and root/gps/lat.



# Results & Demo

## Drive the Car

The user can talk into Google assistant to drive the model car remotely from anywhere in the world by speaking the following sentences.

* Go **forward**
* Please drive the vehicle **backward**
* Please take a **left**
* Can you please take a right **right**
* **Freeze**
* **Halt** the vehicle
* Stand **still**

The Dialogflow NLP models backed by the Google Deep Learning infrastructure successfully interprets what the user is saying and accordingly it was successfully extracting the data entities (Forward/Backward/Left/Right/Halt/Still/Freeze) from the speech and persisting the desired action to the firebase database.

The Raspberry Pi 3B+ controller running on top of the model car is continuously polling the firebase database and as per the requested state the controller is driving the car.

Please see the demo video link provided in the references section.

## Celestial Object Details

The user can utter into the Google assistant to get the co-ordinate and image details of the 500+ celestial objects such as constellation, stars and clusters. The full list of the celestial objects recognized by the current implementation is provided in the references section.

The following are few sample sentences, the user can utter to get the coordinates and image details.

* Can you please get the co-ordinates of M33 cluster
* Can please show me the details Gemini
* details of Hydra, please
* Castor co-ordinates
* Hi, there can you fetch the details of Leo

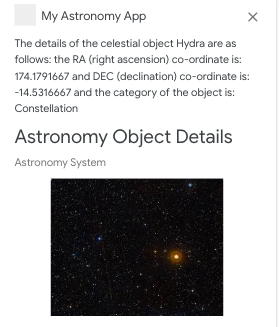
The Dialogflow model is successful able to interpret the speech, extract the entities and google cloud function after scraping the results from the web and responding back to the user with the details and image.

Please see the demo video and for reference screen prints of the google assistant is shown below.

**Example-1 (Cluster)**

**User utterance:** please get the details of Hydra

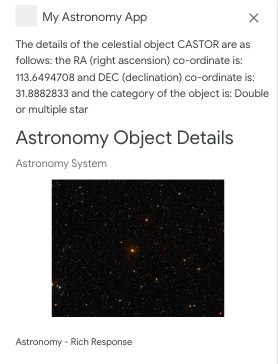
**Google Assistant Response:**



**Example-2 (Star)**

**User utterance:** can you show me the images of Castor

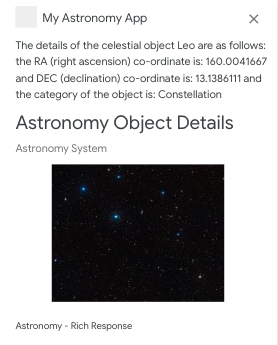
**Google Assistant Response:**



**Example-3 (Constellation)**

**User utterance:** fetch the images of Leo constellation

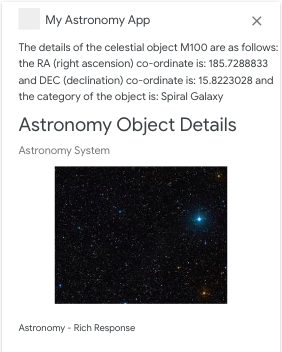
**Google Assistant Response:**



**Example-4 (Galaxy)**

**User utterance:** M100, please

**Google Assistant Response:**



## Sky Maps

Similar to the celestial object details, the user can utter into the Google assistant to get the sku, telescopic or horizon maps for a given location. The location can be a specific longitude/latitude GPS co-ordinates or country or city or city/country together. The user can also ask for Horizon or Telescopic or Sky maps

The following are few sample sentences, the user can utter to get the coordinates and image details.

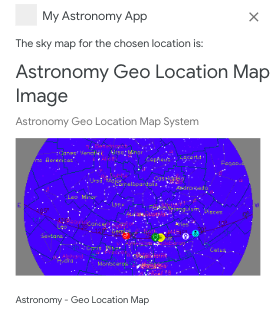
* Show me the sky map for London, UK
* Present me the horizon map for Mumbai
* Telescopic map for longitude 47.8767 and latitude –12.4567
* Please shoe me the sky map for greenland
* Get the me telescopic map for Argentina

Please see the demo video and for reference screen prints of the google assistant is shown below.

**Example-1 (City & Country)**

**User: please show me the sky map for London, UK**

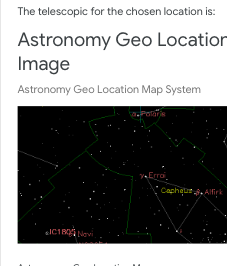
**Google assistant response:**



**Example-2 (City)**

**User: Mumbai telescopic map please**

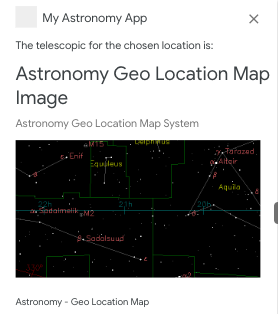
**Google assistant response:**



**Example-3 (GPS Co-ordinates)**

**User: please retrieve the telescopic map for longitude 47.7684 and latitude –0.1254**

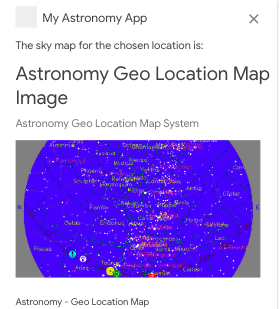
**Google assistant response:**



**Example-4 (Country)**

**User: Argentina sky map please**

**Google assistant response:**



# Conclusions

Using the Software (AI-NLP, Google Cloud Platform) and hardware (2 wheeled driven model car equipped with GPS module) a low cost one-stop platform is successfully built where the user can ask for moving telescope platform to retrieve the details of celestial and localised maps.

# Next Steps

Enhance the platform to add more celestial object catalogues.

# References

[1] Dialogflow tutorial: - <https://codeburst.io/2-how-assistant-work-introduction-to-dialogflow-319a72ba2db>

[2] Dialogflow tutorial: <https://medium.com/swlh/how-to-build-a-chatbot-with-dialog-flow-chapter-1-introduction-ab880c3428b5>

[3] Dialoglfow Documentation: <https://cloud.google.com/dialogflow-enterprise/docs/>

[4] Google Cloud SDK installation: <https://cloud.google.com/sdk/docs/quickstart-macos>

[5] Stuart Lowe's Website: http://www.strudel.org.uk/lookUP, this website provides all the details, categories and image of stars, constellation and clusters (2017).

[6] John Walker's Website: http://www.fourmilab.ch/, this website provides the sky, horizon and telescopic map for a given location (2019).

[7] Firebase SDK installation: <https://rakibul.net/fb-realtime-db-python>

# YouTube Tutorial URLs

## Summary – 2 Minutes Video

## Girhub Repository