

CCA Installation Guide

Required Components

- Philips Hue light
- Philips Hue bridge
- Raspberry Pi 4 (with Raspberry Pi OS for Desktop installed)
- Sense hat for Raspberry Pi 4
- Monitor, keyboard and mouse for Pi 4 (or ssh)
- Smart phone

Installation

1. Google Maps Api

Sign into google and obtain a google maps API key.

Link to set up: <https://developers.google.com/maps/documentation/embed/get-api-key>

Store this API key somewhere safe as you will need it when setting up the application.

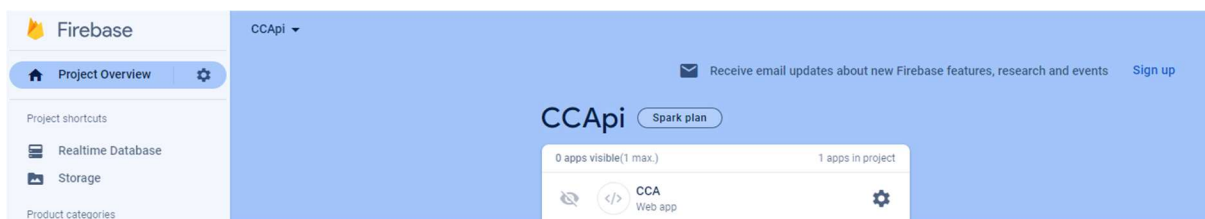
2. Firebase Realtime Database

Sign into Firebase via google and set up a Realtime database.

Link to set up: <https://firebase.google.com/docs/database/web/start>

Once you've created a project, navigate to the settings cog as seen below next to 'CCA'.






Click this and then navigate to 'Service Accounts' and then click 'Generate New Private Key'. Store this JSON file somewhere safe as it will be needed later.



3. Blynk application

Sign in to Blynk.com and set up a new device labelled CCA. Once this device is created click on it from the devices screen. Next to the name of the device where there is an 'offline' box, click the drop down to the right and then 'edit dashboard'. Once in the dashboard view, click '+ new Datastream' and then 'virtual pin'. Create two virtual pins matching the below format:

+ New Datastream

|  | Id | Name | Alias | Color | Pin | Data Type | Units | Is Raw | Min | Max | Decimals | Actions |
|---|----|----------------|----------------|---|-----|-----------|-------|--------|-----|-----|----------|---------|
|  | 2 | ETA | ETA |  | V1 | String | | false | | | -- | |
|  | 3 | Journey Status | Journey Status |  | V2 | Integer | | false | 0 | 4 | -- | |

The image displays four visual workflow diagrams, each representing a different trigger condition for a journey status. Each diagram is structured as follows:

- Trigger (When):** A colored square icon followed by the trigger text.
- Action (Do this):** A circular icon followed by the action text.
- Next Action (Add next action):** A light blue box containing a plus icon and a list of available actions.

Workflow 1: On The Way

- When:** On The Way (Dark Blue square)
- Do this:** Send In-App Notification (Mobile icon)
- Add next action:** Control Device, Forward Device Data, Send E-Mail, Send In-App Notifications, Wait, Then Do Something

Workflow 2: Leave Immediately!

- When:** Leave Immediately! (Red square)
- Do this:** Send In-App Notification (Mobile icon)
- Add next action:** Control Device, Forward Device Data, Send E-Mail, Send In-App Notifications, Wait, Then Do Something

Workflow 3: Cutting It Close

- When:** Cutting It Close (Orange square)
- Do this:** Send In-App Notification (Mobile icon)
- Add next action:** Control Device, Forward Device Data, Send E-Mail, Send In-App Notifications, Wait, Then Do Something

Workflow 4: On Time

- When:** On Time (Teal square)
- Do this:** Send In-App Notification (Mobile icon)
- Add next action:** Control Device, Forward Device Data, Send E-Mail, Send In-App Notifications, Wait, Then Do Something

Next, on your smartphone download the Blynk application from Google Play or App Store. Sign in using the same account you set up the device with and select the CCA. If you wish, you can add a widget for ETA here however it is not necessary for the functioning of the push notifications.

Finally return to the Blynk website and click 'Devices' -> 'CCA' -> 'Device Info' and click the button to the right of 'AUTHTOKEN' to copy your authentication key. Store this as you will need it when setting up the application.

4. Philips Hue Lights

Click the following link and set up a developer account: <https://developers.meethue.com/>

Once you're logged in, click 'Develop' in the navigation bar then 'Get Started'. Follow this guide to obtain your Hue Username. Store this somewhere safe on your computer as it will be needed during application setup.

5. Phone MAC Address

Lastly, use the following link to find your smartphone's MAC address:

https://support.sasktel.com/app/answers/detail/a_id/17317/~/finding-the-mac-%28or-wi-fi%29-address-of-a-mobile-device-or-computer#:~:text=Android%20phone%20or%20tablet%3A%20Go,About%20%3E%20More%20Info%20%3E%20MAC%20address

Once again, store this somewhere safe on your computer for future use.

Application Setup

Download the GitHub repo (<https://github.com/chipspeak/compSysAssignment2>) and extract all the files. Open your terminal and navigate to the directory in which you've extracted the files. Run the install script using the command: `./cca_install`

Once the script has run and the requirements have been successfully installed, move your 'serviceAccountKey.json' that you downloaded from firebase into the same directory as the extracted files. Next, create a blank .env file in the directory and place the other sensitive information in it in the below format:

```
#google maps api key
apiKey=YOUR_GOOGLEMAPS_API

#phone details
phone=YOUR_PHONES_MAC_ADDRESS

#hue details ,manually obtained username from the Philips Hue Developer portal
hueUsername=YOUR_HUE_USERNAME

#authorisation key for Blynk
blynkAuth=YOUR_BLYNK_AUTHKEY
```

Open a text editor of your choice and open the file 'main.py'. Find the below section and change the origin and destination to zip codes of your choice (include the whole zip code in one sequence with no spaces) and save the file.

```
#declarations for use with api call and user checking functionality
origin = 'ORIGIN_ZIP_CODE'
destination = 'DESTINATION_ZIP_CODE'
```

Open 'time_checks.py' and navigate to the below section and change the start time to your own following the same format where you only adjust the numeric values for 'hour' and 'minute'.

```
# Create a time object that will function as our work start time
startTime = time(hour=18, minute=45)
```

Open 'hue_integration.py' and scroll to the below section and change the hue bridge IP address to the ip address of your own hue bridge and the lamp to the name you gave your own light on hue setup (you will find this during the hue dev guide linked in the previous section).

```
# initialising hue bridge via its local ip
hue = Hue(bridge_ip='192.168.1.1', username=hueUsername)
# initialising the lamp
light = hue.get_light(name='C.C.A Lamp')
```

Open 'user_detection.py' and find the below section and change the IP address to that of your own home network address on which the pi and hue lights are connected.

```
# Returns the list of known devices found on the network
def find_devices():
    # calling the subprocess and piping the local network results to grep
    output = subprocess.check_output('sudo nmap -sn 192.168.1.0/24 | grep
MAC', shell=True)
```

Finally, open 'firebase_integration.py' and change the database URL in the below example to the URL from your own Realtime database.

```
cred=credentials.Certificate('./serviceAccountKey.json')
firebase_admin.initialize_app(cred,
    {
        'databaseURL': 'https://ccapi-42492-default-
rtdb.europe-west1.firebaseio.com/'
```

If you wish to avail of the website functionality continue with the steps below but otherwise you should now be able to run the software either via a text editor like Visual Studio or by simply running the main script in your terminal. To do this, navigate to the project directory in the terminal and then type:

```
python main.py
```

Hit 'Enter' and the application should run. If there are any errors, consult the errors in the terminal and check that you have everything from the requirements.txt file installed. Additionally, you can use crontab -e in your terminal to schedule an execution of the script at set times.

Please follow this link to video explaining this in greater detail:

https://www.youtube.com/watch?app=desktop&v=EgrpfvBc7ks&ab_channel=TonyTeachesTech

Website Setup

If you wish to utilise the website functionality of the application, please follow the below steps.

1. In a web browser, go to <https://glitch.com/>
2. Once there, click 'new project' → 'import from GitHub'.
3. Enter this url: https://github.com/chipspeak/cca_website
4. When navigating the files in the leftmost tab you should see an empty .env file. Note this.
5. Go to the same place on the firebase website that you obtained your serviceAccounts json
6. On the 'Project Settings' page, under 'General' scroll down to 'Your Apps'.
7. Navigate to the end of the code block containing 'const firebaseConfig = {'
8. Copy the contents of this block to a safe location on your computer.
9. Open glitch.com again and click on the empty .env file.
10. Paste the following into this, replacing the templates with your own firebase data.

```
FIREBASE_API_KEY=  
FIREBASE_AUTH_DOMAIN=  
FIREBASE_DATABASE_URL=  
FIREBASE_PROJECT_ID=  
FIREBASE_MESSAGING_SENDER_ID=  
FIREBASE_APP_ID=
```

The website should now function as intended. It records the four most recent journeys taken via the firebase Realtime database.