Team IOT - Final Project Report

Team IOT : Jack Kovach - jkovach2 , Toby Liang - tobyzl2 , Rahul Javalagi - rahulaj2
Video Link:
https://drive.google.com/file/d/1be21DOL0aPcoJ9TJmPJQXZC5uysFtQ9m/view?usp=sharing
GitHub Link: https://github.com/rxj171630/SmartHub

Motivation:

Things in life can get very disorganized. Between work, education, family, and regular errands, things can get incredibly busy and stressful. Our Smart Hub can help organize and alleviate some of these stressors and save time by putting useful information about your home, daily events, and personal notes all in one place with several different user interface types such as hand gestures and speech. This allows users to interact with the application in multiple ways for easiest usage. The user can easily automate various day-to-day interactions with other devices in your home from any location as long as the Raspberry Pi is connected. This allows the user to overall be more efficient in their daily routine.

Technical approach:

For the overall architecture of the system, the Smart Hub uses a Raspberry Pi as the server to the web client interface. The Raspberry Pi has the hardware needed for the application such as the microphone and camera. The Raspberry Pi makes calls to IFTTT and other APIs, runs the python programs, uses the microphone to record the audio samples used for text to speech, and opens the camera stream for the gesture recognition settings used to turn lights on and off. The web interface runs on the React app on the client screen for the user interface. This way, all of the recordings and information will take place on the pi and if the user needed to view this information away from the home, they would have the ability to do so.

Implementation Details:

Image Recognition:

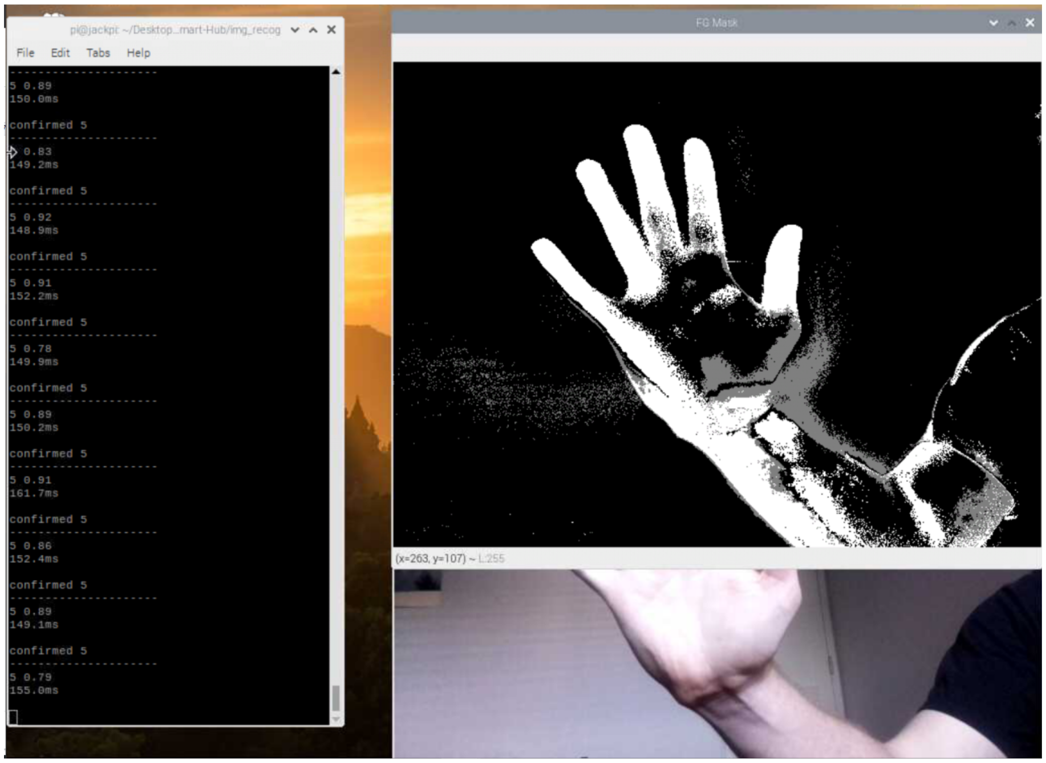
For image recognition, several iterations were needed to train a model using the kaggle fingers dataset. The first attempt at recognizing the images used TensorFlow lite APIs to train the tflite model with pretrained. The idea was to use the guide for Transfer Learning for Object Detection similar to how the PiCar project used and expanded the Coco model for the detection of stop signs and cones. The images were labeled with left and right hand, followed by the number of fingers held up (0L-5R).

Using no bounding box information provided to the csv file that typically covers the location of the hands, several different models were created. There were many hurdles to creating these modules. The RAM required exceeded what could be run on a local computer and they took several hours to train on even on Google CoLab’s Pro Tier. Several different training parameters were used with the largest batch sizes Google CoLab’s TPU would allow without crashing due to RAM restrictions. To maximize time during training of these models, object classification models were also trained using the TensorFlow Lite image classification guide. Bounding boxes were not required in this instance.

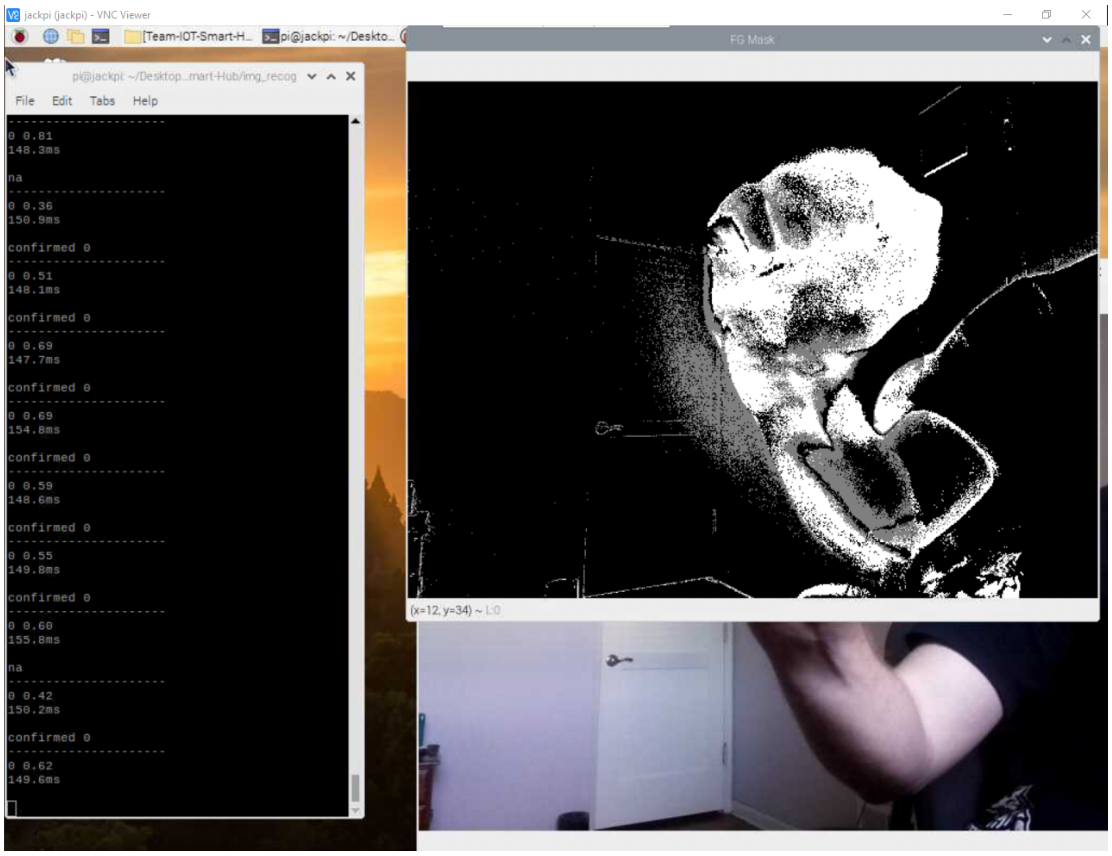
While some of the models had better performance than others for object detection none had consistently good performance for all 5 fingers. Parameters altered included the number of epochs, model type (EfficientDet-Lite0, EfficientDet-Lite1), and batch size - training outcomes can be found in linked Colab Notebooks. Overall, the object detection tflite models did not have as good performance during tests on-camera as the image classification tflite models. Still, the models had issues that are suspected to be caused by the black background in the training images.

To combat the issue of black background and learn if this would improve the model or a different direction altogether would need to be taken. To remove the background from the camera image, the OpenCV docs reference on Background Subtraction was used. This worked better than without any background removal but still had difficulty with the intermediate fingers 1,2,3,4. A retrained model with 0 and 5 fingers caused us to change functionality to just be a binary message to the IFTTT application which would be used as open-hand for light on and closed-hand for light off. A program that duplicated many images of black squares was added to the training model so that the empty background would not be represented as 0L or 0R label.

For determining if the image is a “confirmed” image that we want to use to make a decision in the application - the labels for 5L and 5R were combined into a single 5 and same with 0L or 0R so either hand can make that decision. The logic for the output confirmation was based on the label output, its prediction probability from the model being over 0.5 and it having the same label as the previous capture of the image on the stream. Images below with printed output show what the live image and filtered image look like with the image recognition program.



```
---------------------
5 0.89
150.0ms

confirmed 5
---------------------
5 0.83
149.2ms

confirmed 5
---------------------
5 0.92
148.9ms

confirmed 5
---------------------
5 0.91
152.2ms

confirmed 5
---------------------
5 0.78
149.9ms

confirmed 5
---------------------
5 0.89
150.2ms

confirmed 5
---------------------
5 0.91
161.7ms

confirmed 5
---------------------
5 0.86
152.4ms

confirmed 5
---------------------
5 0.89
149.1ms

confirmed 5
---------------------
5 0.79
155.0ms
```



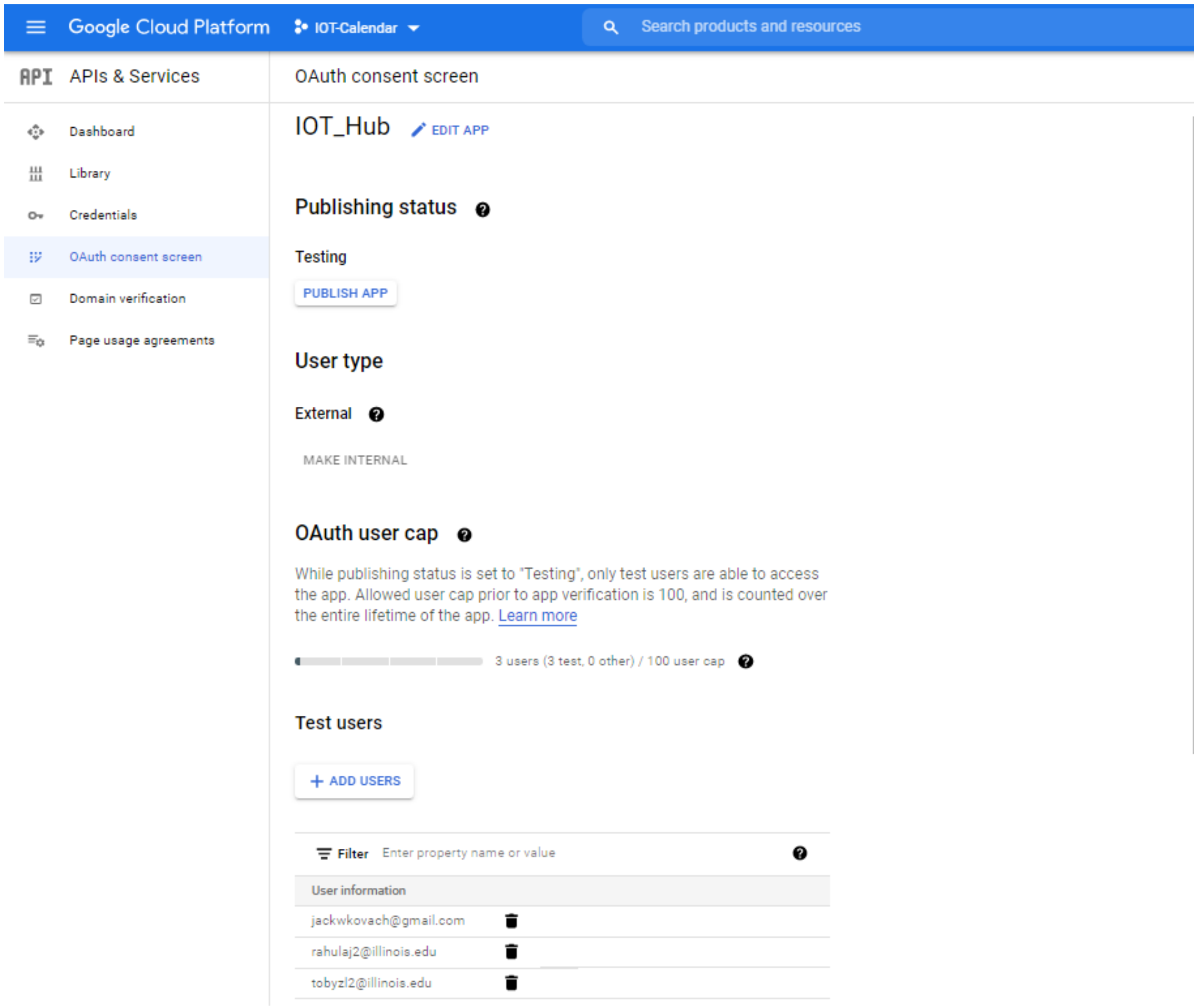
```
---------------------
0 0.81
148.3ms

na
---------------------
0 0.36
150.9ms

confirmed 0
---------------------
0 0.51
148.1ms

confirmed 0
---------------------
0 0.69
147.7ms

confirmed 0
---------------------
0 0.69
154.8ms

confirmed 0
---------------------
0 0.59
148.6ms

confirmed 0
---------------------
0 0.55
149.8ms

confirmed 0
---------------------
0 0.60
155.8ms

na
---------------------
0 0.42
150.2ms

confirmed 0
---------------------
0 0.62
149.6ms
```

Program Details - several helper functions were created to get the images to where they needed to be for the training.
Include name of actual raspi program
convert_to_JPEG.py - On Google CoLab, the original PNG images were not working properly in the notebook. This converted to JPEG format that was functional.
to_csv.py - creates csv formatted to include all filenames for the object detection models with corresponding comma suffixes for bounding boxes and labels
strip_csv.py - uses pandas dataframes to split up only the 5 or 0 finger datasets to move into csv

Speech-to-Text:
For the Speech-to-Text recognition on the Raspberry Pi, an additional USB Microphone is used with the Google Speech-To-Text API. In the first test of the Speech-To-Text API, the python call to r.recognize_google would be limited in time and would stop recording when the user speaks. We wanted the ability to start and stop recording on a button press on the user interface with a stretch goal of having the hand gestures be compatible with the speech-to-text in addition to the light interface. For this the program, the portaudio19-dev packages and pyaudio packages were installed and the audio stream sample code from makersportal was referenced to generate a .wav file the a length that could be determined by a button press/user input. This .wav file then is fed through the same r.recognize_google command to convert the speech to text.

Travel Times:
For the travel times information Google's Distance Matrix API was used with a generated API Key in the same Google Cloud Platform Console. From the React UI, the user can pre-set source and destination addresses to send to the python code to run on the Raspberry Pi to get the information on travel time to return back to the User Interface. As of right now, the travel time is default to driving, not public transit or walking.

Calendar Integration:
For the calendar integration, the Google Cloud Platform Python Calendar API was used referencing the Python Quickstart guide. An OAuth key was added and all of us are users who have access to the pings of the calendar from the application. The quickstart reference was modified to only list the events within the next 7 days with a maximum of 10 days. This could be changeable in future versions of the application if users want more flexibility on the amount of calendar events.



OAuth consent screen

IOT_Hub

Publishing status

Testing

User type

External

OAuth user cap

While publishing status is set to "Testing", only test users are able to access the app. Allowed user cap prior to app verification is 100, and is counted over the entire lifetime of the app. Learn more

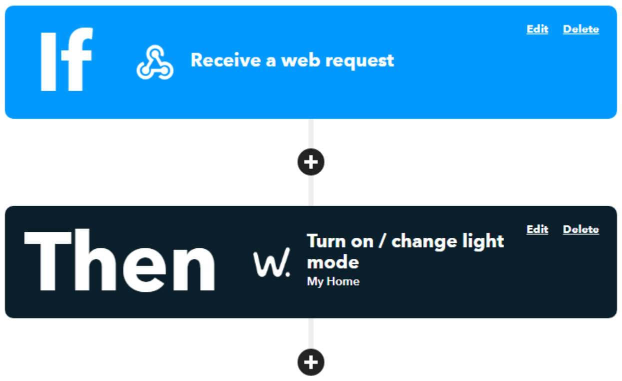
3 users (3 test, 0 other) / 100 user cap

Test users

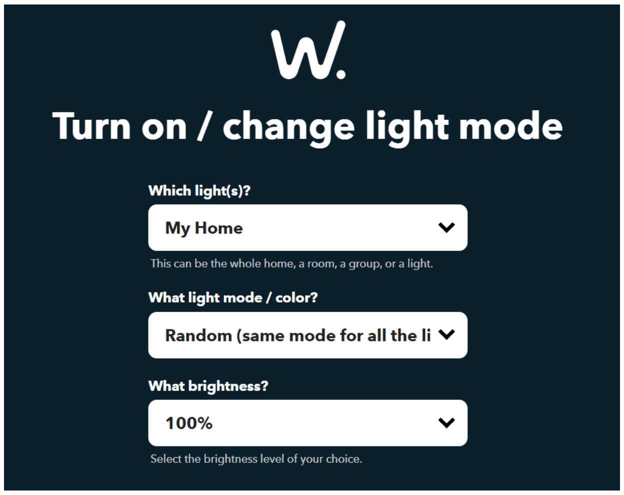
User information
jackwkovach@gmail.com
rahulaj2@illinois.edu
tobyzl2@illinois.edu

Stocks and News Data:
For stocks data, we utilize the yahoo finance API to query stock names, ticker symbol, bid, ask, and market cap of 5 popular stocks in the stock market. We also utilize a news api to retrieve headlines and summaries of trending articles, which we limit to 4 for displaying.

Alarm:
Our smart hub also includes an alarm feature that allows the user to specify the time that the alarm is triggered (Ex. 12:51) and the alarm will turn on the lights to a random color at 100% brightness and call the user’s phone number.


If Receive a web request

Then Turn on / change light mode
My Home


Turn on / change light mode

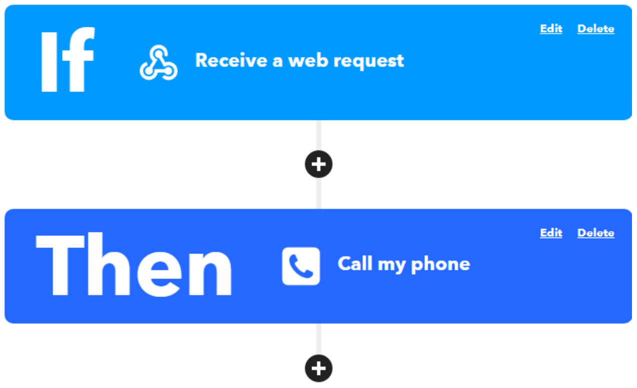
Which light(s)? My Home

This can be the whole home, a room, a group, or a light.

What light mode / color? Random (same mode for all the li

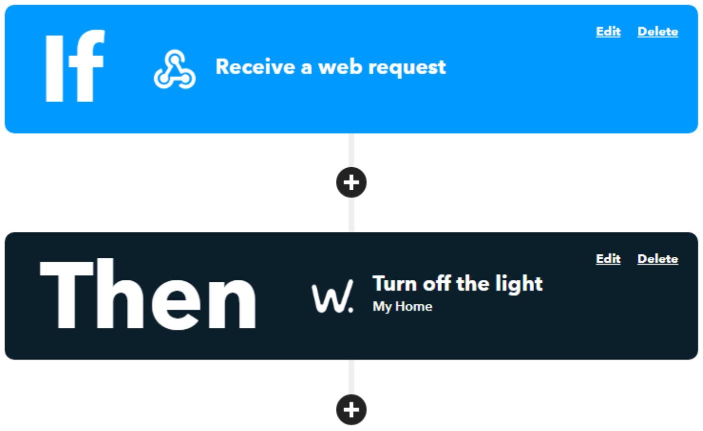
What brightness? 100%

Select the brightness level of your choice.

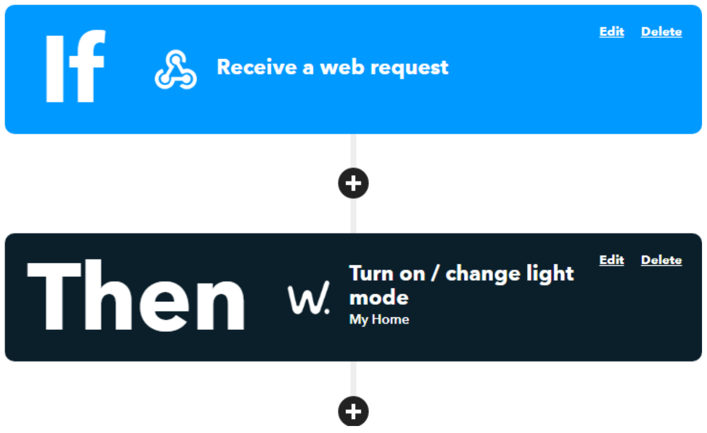

If Receive a web request

Then Call my phone

Gesture-Based Light controls:
Our client supports turning smart lights on and off using hand gestures in front of the raspberry pi camera. We utilized OpenCV along with the IFTTT api so that when the program recognizes a closed hand, it turns off the lights, and when it recognizes an open hand, it turns on the lights to a desired color and brightness.


If Receive a web request

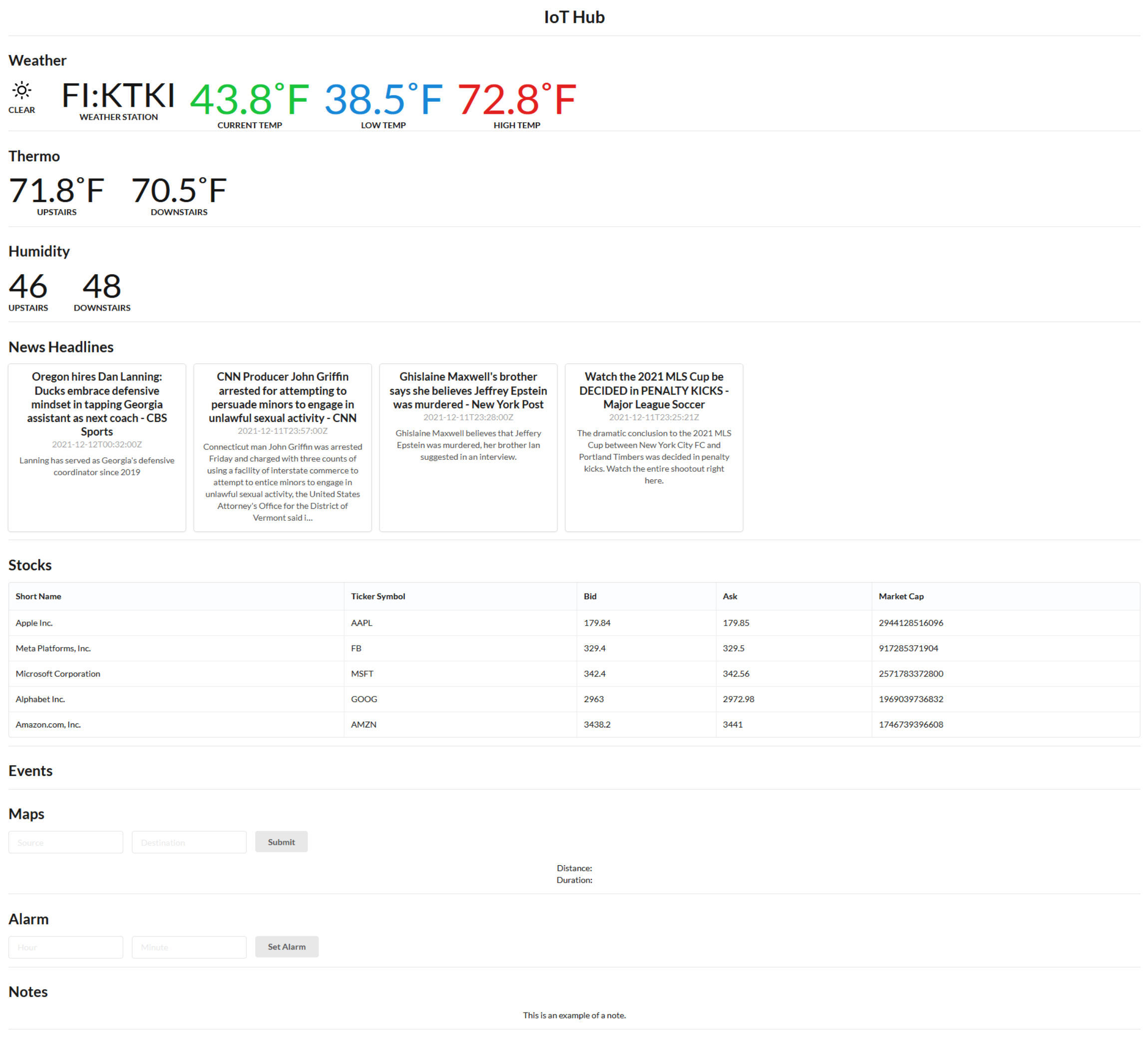
Then Turn off the light
My Home


If Receive a web request

Then Turn on / change light mode
My Home

Server:
Instead of calling each of the APIs separately for all of the different functionalities that our client app supports, we instead run a web server on the raspberry pi that loads sensor data including thermostat/humidity data along with other useful information such as stocks, news, calendar, and travel. Our web server aggregates all of the information into a single central hub and provides eight different endpoints to allow any client app to easily query the data. The server is implemented using Flask and can be run as a Python script on the raspberry pi.

Front-End Interface:
For our front-end interface, we designed our client so that it can be run on any browser and can interface with the web server that runs on the raspberry pi. Our client utilizes the React framework to easily update states of different components to read in information such as form fields and keep track of results from API calls. We create a separate react component for each feature that we are supporting and maintain separate states for each feature. Further, we utilize the semantic-UI library for our widgets.



IoT Hub

Weather

CLEAR — FI:KTKI WEATHER STATION — 43.8˚F CURRENT TEMP — 38.5˚F LOW TEMP — 72.8˚F HIGH TEMP

Thermo

71.8˚F UPSTAIRS — 70.5˚F DOWNSTAIRS

Humidity

46 UPSTAIRS — 48 DOWNSTAIRS

News Headlines

Oregon hires Dan Lanning: Ducks embrace defensive mindset in tapping Georgia assistant as next coach - CBS Sports
2021-12-12T00:32:00Z
Lanning has served as Georgia's defensive coordinator since 2019

CNN Producer John Griffin arrested for attempting to persuade minors to engage in unlawful sexual activity - CNN
2021-12-11T23:57:00Z
Connecticut man John Griffin was arrested Friday and charged with three counts of using a facility of interstate commerce to attempt to entice minors to engage in unlawful sexual activity, the United States Attorney's Office for the District of Vermont said i…

Ghislaine Maxwell's brother says she believes Jeffrey Epstein was murdered - New York Post
2021-12-11T23:28:00Z
Ghislaine Maxwell believes that Jeffery Epstein was murdered, her brother Ian suggested in an interview.

Watch the 2021 MLS Cup be DECIDED in PENALTY KICKS - Major League Soccer
2021-12-11T23:25:21Z
The dramatic conclusion to the 2021 MLS Cup between New York City FC and Portland Timbers was decided in penalty kicks. Watch the entire shootout right here.

Stocks

Short Name	Ticker Symbol	Bid	Ask	Market Cap
Apple Inc.	AAPL	179.84	179.85	2944128516096
Meta Platforms, Inc.	FB	329.4	329.5	917285371904
Microsoft Corporation	MSFT	342.4	342.56	2571783372800
Alphabet Inc.	GOOG	2963	2972.98	1969039736832
Amazon.com, Inc.	AMZN	3438.2	3441	1746739396608

Events

Maps

Distance:
Duration:

Alarm

Notes

This is an example of a note.

Results:

Overall, all items were able to integrate all of our Raspberry Pi features onto the User Interface for the user to interact with and use to get their daily information.

The image recognition models were difficult to train and can still be improved upon for the future - ideally, the model could recognize the difference between all fingers - 0,1,2,3,4,5 but the poor performance required adaptation to the binary 0 or 5 fingers. There are different ML techniques that allow for the fingers to be detected by anchorpoint that could be further explored. It was definitely a great area to experiment with and allowed for a lot of learning with both TensorFlow for the models and OpenCV for the background removal.

The free tier of IFTTT posed some limitations for our implementation. Each integration could only have one action, so we had to create multiple triggers to execute one function of our hub. For example, the alarm required one integration for changing the lights, and a separate integration for calling the phone. Fortunately, IFTTT's webhook trigger allows using custom keywords, so we were able to have multiple integrations execute their actions simultaneously with just one API call from our hub. In addition, IFTTT actions are not instant, there is often a delay, and it can be inconsistent. However, for our proof of concept, we were still successful in achieving our goals.

The Ecobee API was very beneficial to the functionality of our smart hub. Although the documentation for fetching data was not very descriptive, we were able to successfully fetch data for a pair of thermostats. Each thermostat has its own individual properties that we were able to query and display on our interface. Specifically, we display the in-home temperature that each thermostat recorded, one for upstairs, and one for downstairs. In addition, we also displayed the humidity recorded by each thermostat. In addition, Ecobee's API also provided a weather forecast. Since this data was identical between both thermostats, we only fetch this for one of them and display it on the interface.

One area that we struggled with was the speech-to-text implementation. This feature gave us a lot of trouble on the back-end. Although we were able to eventually have the feature working and displaying onto the front-end, we were not able to iron out all of the bugs. Oftentimes, if there is a lot of background noise, the speech recognition will cause the server to crash. If it is successful, it terminates the server as well. Since we want our hub to always be connected to the server, this is not ideal. We were unfortunately unable to come up with a solution to this issue in time.

Overall this project gave us a new way to interact with smart devices we already own, and increased our understanding of APIs. We definitely struggled quite a bit with our initial aspirations and had to dial back some of our intended goals, but in the end, we were able to create not only a dashboard client, but also a server. In addition to the skills obtained throughout the semester, we were also pushed to do additional research and experimentation, especially with the various APIs that were available and how the different frameworks interact with each other. We took the concepts that we had learned throughout the semester and added onto them to achieve our goals.

References:

Ecobee API:
  Getting Started with the ecobee API
Google Cloud Platform Python Calendar API:
  Python Quickstart | Calendar API
IFTTT:
  Connect API - IFTTT
Kaggle Fingers Dataset:
  Fingers
Makersportal Speech-To-Text:
  Recording Audio on the Raspberry Pi with Python and a USB Microphone — Maker Portal
News API:
  Documentation
Open CV:
  OpenCV: How to Use Background Subtraction Methods
TensorFlow: Image classification
  Image classification with TensorFlow Lite Model Maker
Weather icons:
  erikflowers/weather-icons: 215 Weather Themed Icons and CSS
YFAPI:
  API,Documentation | Stock Prices | Quote Comparison