Smart Mirror

IoT Project Document

12/12/2015

TABLE OF CONTENTS

TABLE OF FIGURES

1 OVERVIEW

1.1 THE IDEA & THE MIRROR

A few weeks ago I've seen an article on the TechInsider.io about a developer named Evan Cohen who built a smart mirror. Watching his demonstration video, I found it's a very interesting project and the fact that my room still lacks a mirror so I decided to go on an adventure of building my own smart mirror.

1.2 INITIAL TECHNICAL CONSIDERATIONS

Hardware

I've started a little research about the mirror and figuring out what was needed:

- a 2-way mirror
- an old LCD
- a Raspberry Pi with WiFi connectivity
- a distance sensor
- some power cable, HDMI cable
- and **lots of spare time**…

Software

The mirror requires a speech recognition capability and at first I was thinking of using Sirius or Jasper, both are popular open source alternatives to Siri or "OK Google" in voice recognition field.

However, after playing around with the deployment and configurations of these 2 solutions, I've found that the 2 software is not as simple to use and configure. Both installers depend on several other open source software such as Kaldi, Pocketsphinx, or Sphinx4, or SURF (image matching), etc. which makes the installation and tuning quite a challenge. The quality of the speech recognition therefore suffers. I've even had to mess around with the kernel recompilation to get

through some of the issues during installation – which I don't think it's feasible for a hobby project in long term maintenance.
After several researches, I've chosen **HTML5 as the main platform** for the mirror. The system will be fully written in NodeJS, with server side code uses Python. This choice makes hosting and deployment simple and easy. One thing to note, the oroject Chromium already supported Speech Recognition and even Speech Synthesizing since version 33 Beta (48 as of current) so that is also a big plus.

2 HIGH LEVEL REQUIREMENTS

The main feature of a common mirror is to be able to reflect a person's image through the glass. The smart one will have some "smart" functionalities that value add to the user, such as:

- Able to retrieve and display **weather information**.
- Able to show basic **mapping information** for reference.
- Able to **control some IoT devices** in the house, such as turning ON or OFF light bulbs, air conditioner, etc.
- Able to **detect nearby user** using proximity distance check to automate the lighting or screen information.
- Able to **play music** from Spotify, YouTube.
- Read news.

2.1 USE CASE

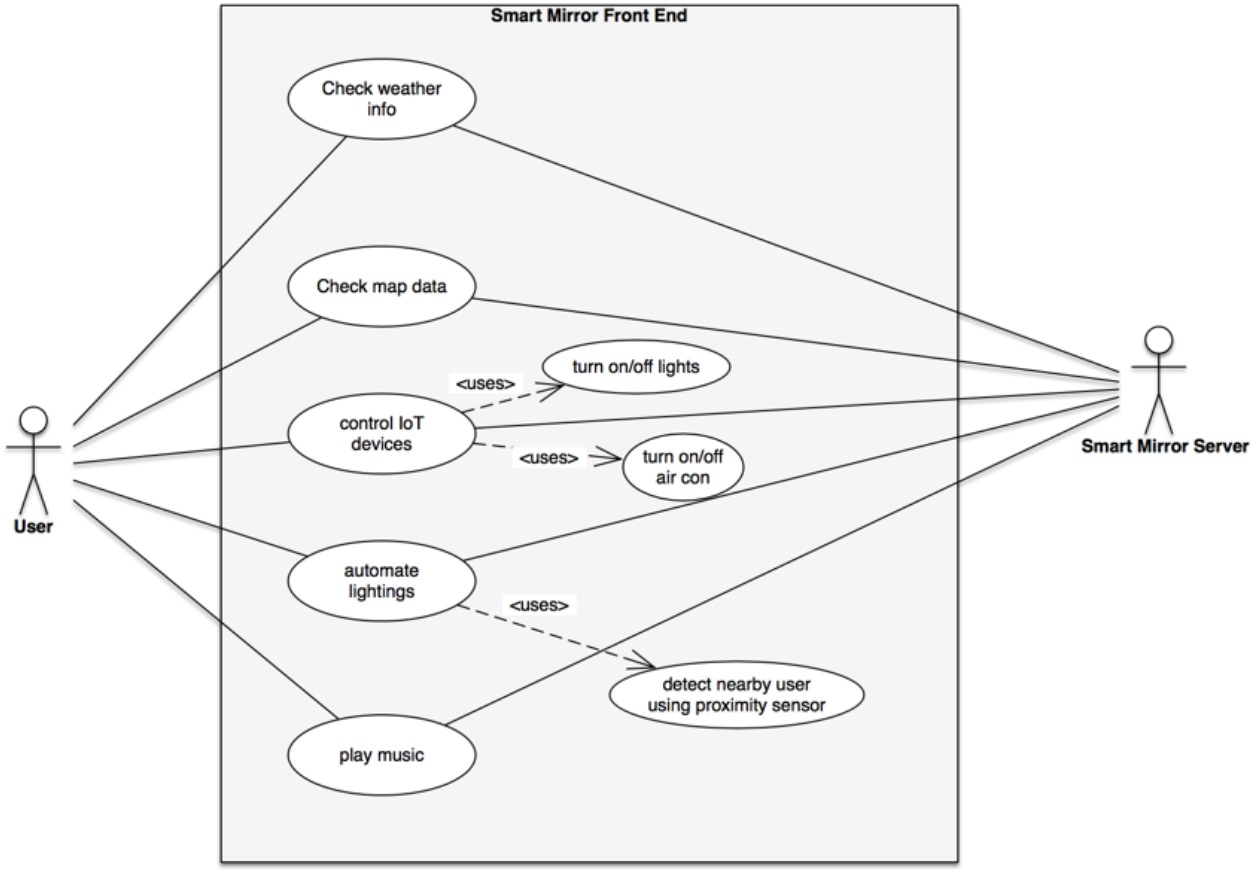

Figure 1 – Use Case Diagram

The use case diagram can be extended in the future to have more use cases when required.

3 PRODUCT DESIGN

3.1 USER FLOW DESIGN

These are the UX prototype design & the User Flow screens of the Smart Mirror. The final product and functionalities may slightly deviate from this design; however, the general idea should be the same. For example, the product will still show a home page when first boot up, but the listing and layout may change.

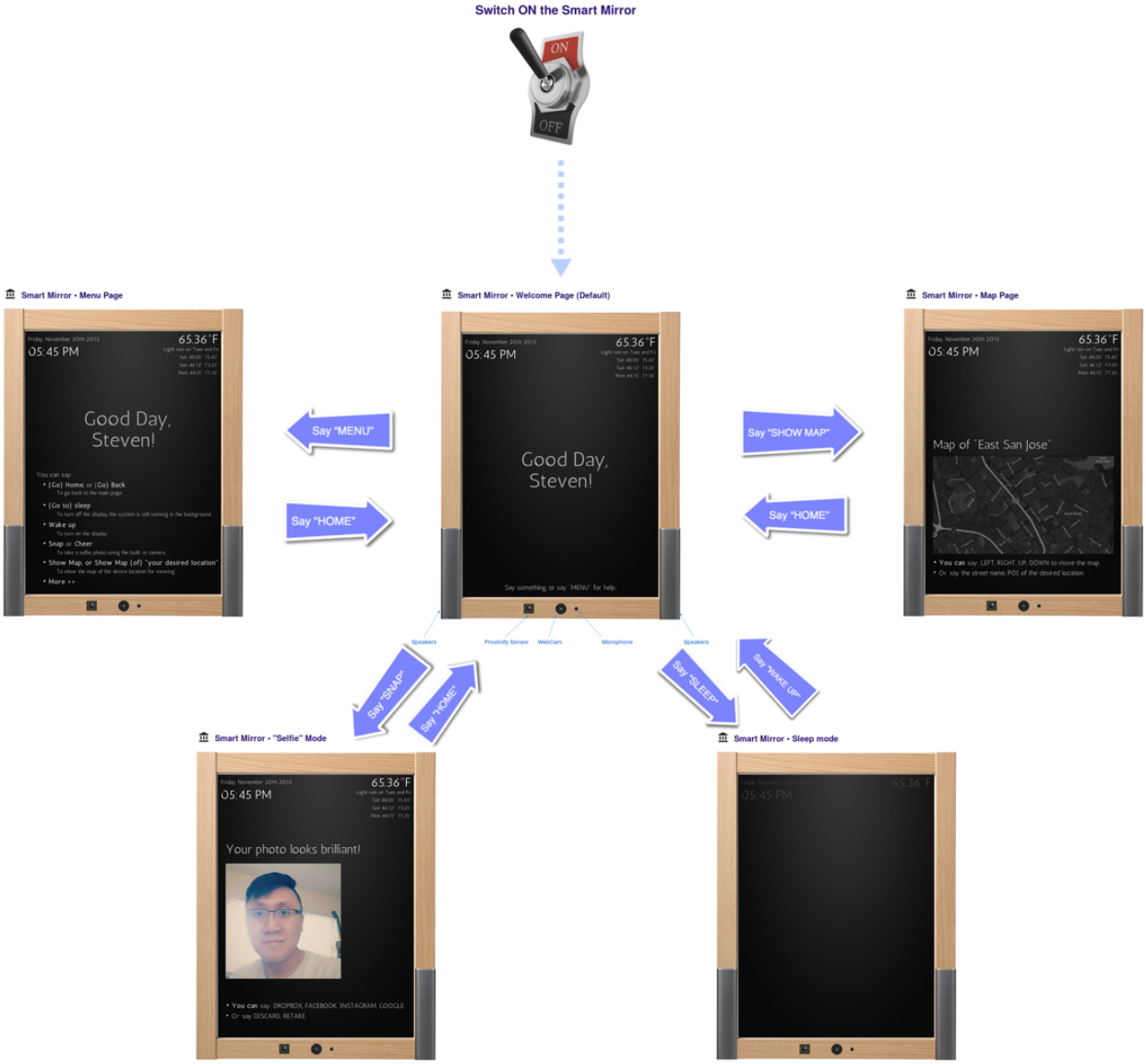

Figure 2 – User Flow

3.1.1 UX Design

3.1.1.1 Home Screen

Figure 3 – UX Design – Home Screen

3.1.1.2 Event Screens

Figure 4 – UX Design – Map

Figure 5 – UX Design – Menu

3.2 HARDWARE

The project requires these hardware:

- A mirror
- An LCD
- A Raspberry Pi, wirings and shields
- Proximity sensor
- Philips Hue Bridge and Light bulbs, for demonstration of controlling IoT devices.

3.2.1 Create the mirror

This is probably one of the hardest parts. In order to to show the information from behind the mirror, the mirror must be a two-way mirror (or observation mirror). It means on one side of the mirror; the glass will reflect (almost all) light rays coming into it. The user will see his image from this side. And on the other side, it should be transparent to allow the the light from the LCD to come through and shown on the mirror.

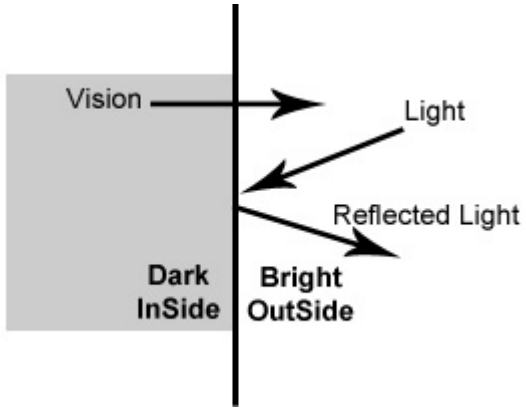

Figure 6 – Principle of a two-way mirror, left is where the LCD is places, right is the user side

There are 2 ways do this, the convenient way is to purchase an (expensive) two-way mirror off the shelf. The more tedious (and probably cheaper) way is to apply a privacy window film onto the transparent glass. This requires some practice and the quality is usually not as good as the two-way mirror you buy from the shop. It's easy to leave some bubbles between the film, or the film may come off after a week or two.

Figure 7 – two-way mirror

3.2.2 The monitor

A Hannspree 17" LCD monitor with HDMI input is used to illuminate the information onto the mirror. This is the LCD after being torn apart: the LCD panel, the I/O and signal PCB board (in green), and the power converter board (yellow).

Figure 8 – the LCD and its parts

- **Project Logs:** The challenge of this part is to stack the panel and the boards into the back of the mirror and keep them stable.

3.2.3 Raspberry Pi

As the project requires several complex computational tasks, such as voice recognition, client-server communication, data visualization. A Raspberry Pi therefore fits the purposes perfectly. RPi v2 is used in this project. However, the RPi v2 is short of audio input (MIC) so I've added a Suptronics X300 extension board, which provides Audio I/O, Wifi a/b/g/n capability, SATA I/O, Bluetooth connectivity, IR sensor, plus extended 40 GPIO header pins.
The expansion board requires a customized build of the kernel that includes some drivers for it work. Luckily, SupTronics provided a Raspbian distro image so we do not waste time building the drivers from scratch.
The image includes:

- Based on RASPBIAN (Release date: 2015-01-30)
- Rpi-update is included
- Updated to the latest Raspberry Pi firmware and kernel as of Feb 15th 2015
- Alsamixer is included and configured
- RTC time is configured
- IR remote is configured

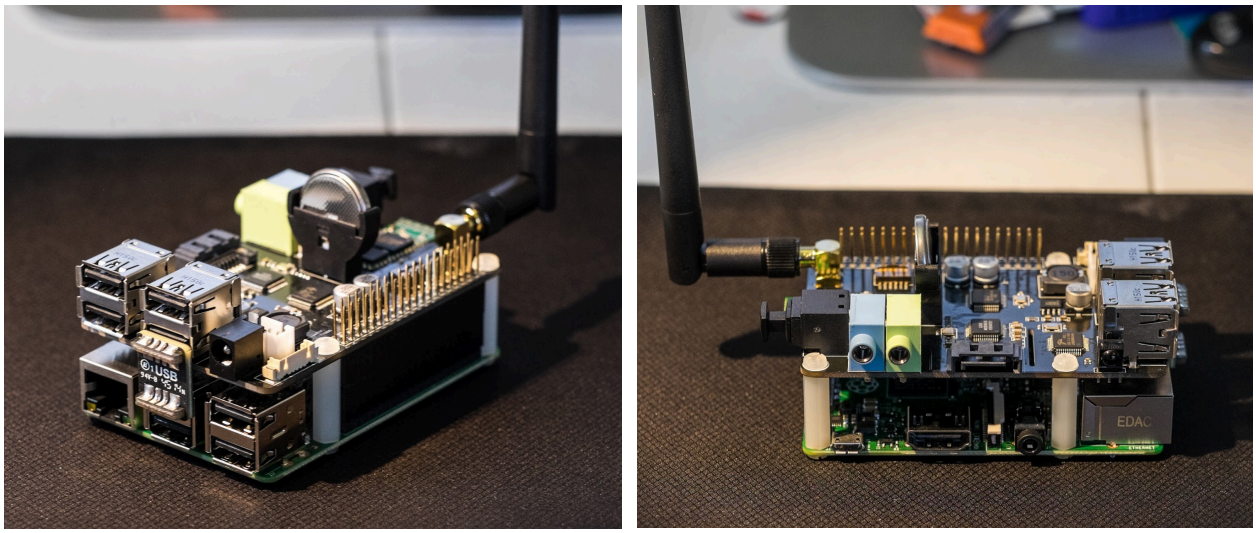

Figure 9 – the Raspberry Pi 2 with the X300 shield on top

Raspberry Pi 2 Technical Specifications

- Broadcom BCM2836 Arm7 Quad Core Processor powered Single Board Computer running at 900MHz
- 1GB RAM
- 40pin extended GPIO
- x USB 2 ports
- pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Micro USB power source

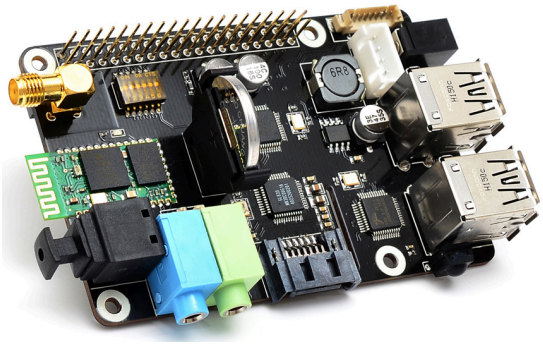

Figure 10 – the X300 expansion board

X300 Expansion Board Technical Specifications

Input Voltage	- 6V to 18Vdc converted to 5V, 4A via step-down DC/DC converter to power the Raspberry Pi
SATA	- Allows you to connect SATA devices to your Raspberry Pi
Audio	- 3.5mm MIC in jack - 3.5mm stereo audio jack - SPDIF output - Audio IO connector (Microphone input and stereo audio amplifier 3.3Wx2)

Wireless	- WiFi (IEEE 802.11b/g/n) with external antenna - IR sensor (38KHz) - Bluetooth serial communication
USB Storage	- Self-powered USB hub with 3 ports
Real-time clock (RTC)	- Based on DS3231SN with included CR2032 battery
Misc	- Power output socket - Camera flex slot so camera can still be used with the expansion board attached - DIP switch to remove connection from RPi's pin header - Directly connected on top of the Raspberry Pi using the board GPIO header pins - No wiring nor soldering is required - Duplicated the 40-pin header of the R-Pi in order to support existing expansion boards - Suitable for Raspberry Pi Model B+ and Raspberry Pi 2 Model B
Dimensions	- 85 x 56mm (Same size as Raspberry Pi)

3.2.4 Proximity Sensor

To implement the proximity check, a Sharp GP2Y0A21YK0F analog distance sensor is used. The sensor can measure an object accurately between 10 cm to 80 cm.

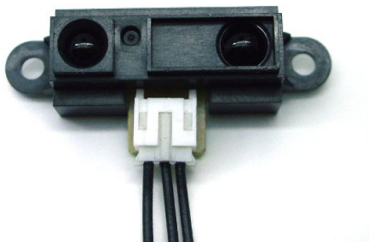

Figure 11 – Sharp GP2Y0A21YK0F IR Distance Sensor

The sensor is connected to the GPIO headers on the Raspberry Pi via a breadboard, which also holds a 10K Ohm (pull up) resistor and an ADC to convert the analog signal to digital distance metrics.

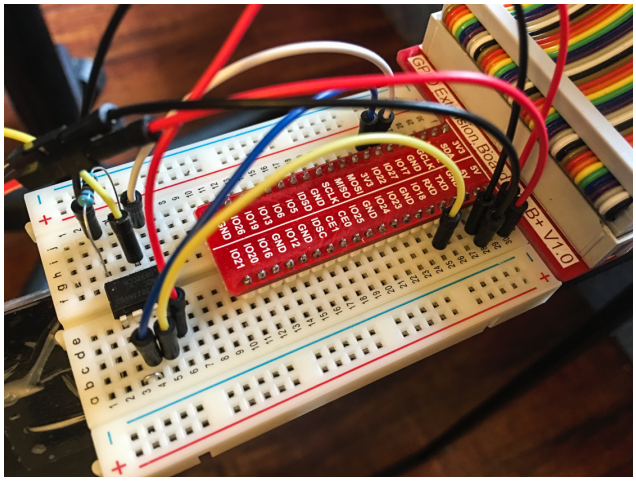

Figure 12 – Breadboard with wiring, resistor and an ADC

- **Project Logs:** There was one issue that I ran into when connecting the sensor to the Pi. It's quite strange that the sensor maximum read is only up to 50 centimeters, whilst testing separately it yielded up to 1000 centimeters. After some troubleshooting with the wires and the software, I discovered the sensor may not function properly with insufficient power. This is because the RPi is connected to a built-in USB power plug on the monitor. This power source is weak and thus unable to power the sensor to reach its full range. The solution is to power the RPi directly from a power adapter instead.

Sharp GP2Y0A21YK0F analog distance sensor technical specs:

Maximum range:	80 cm
Minimum range:	10 cm
Sampling rate:	26 Hz
Minimum operating voltage:	4.5 V
Maximum operating voltage:	5.5 V
Supply current:	30 mA
Output type:	analog voltage
Output voltage differential:	1.9 V

3.2.5 IoT Devices

To demonstrate the ability to control IoT devices from the smart mirror, a Starter Kit of Philips Hue lighting, including the Bridge and the soft light A19 White Bulbs, are used.

Figure 13 – Philips Hue Starter Kit

Connecting the kit

- Connect the Philips Hue Bridge to the Local Area Network using RJ45 network cable, the setup is automatic.
- Replace the room light bulbs with the provided A19 Soft White.
- Install Philips Hue app on iPhone or Android phone to complete the setup.

If everything is set, the 3 blue lights are ON on the bridge. And the API is accessible through local interface provided by the bridge (no outgoing API call to the cloud is required). The bridge IP address is discoverable when accessing this URL www.meethue.com/api/nupnp.

- **Project Logs:** At first, I've actually tried the GE Link Starter Kit. This kit is almost half the price of the Philips Hue. However, the APIs to call the GE Link devices is powered by

Quirky platform, i.e. “the cloud”. That means any API call has to go through the cloud first then come back in. The delay in each action to turn ON/OFF the light is quite noticeable due to this setup. Another major issue is Quirky does not have the API key and secret ready in the kit. The developer has to send them a request via email and will be told to wait for 3-4 weeks (unbelievable!) for account provisioning. So I’ve dropped GE Link and go with Philips Hue Kit eventually.

3.2.6 The Casing

I’ve re-used part of the mirror frame as the base for the casing. After measuring the dimensions of the mirror and the LCD panel, the DIY is quite simple with some paper carton stuffed between the panel and the frame to keep them steady.

3.3 SYSTEM MODEL

The below diagram describes the components used in the system, and its interraction.

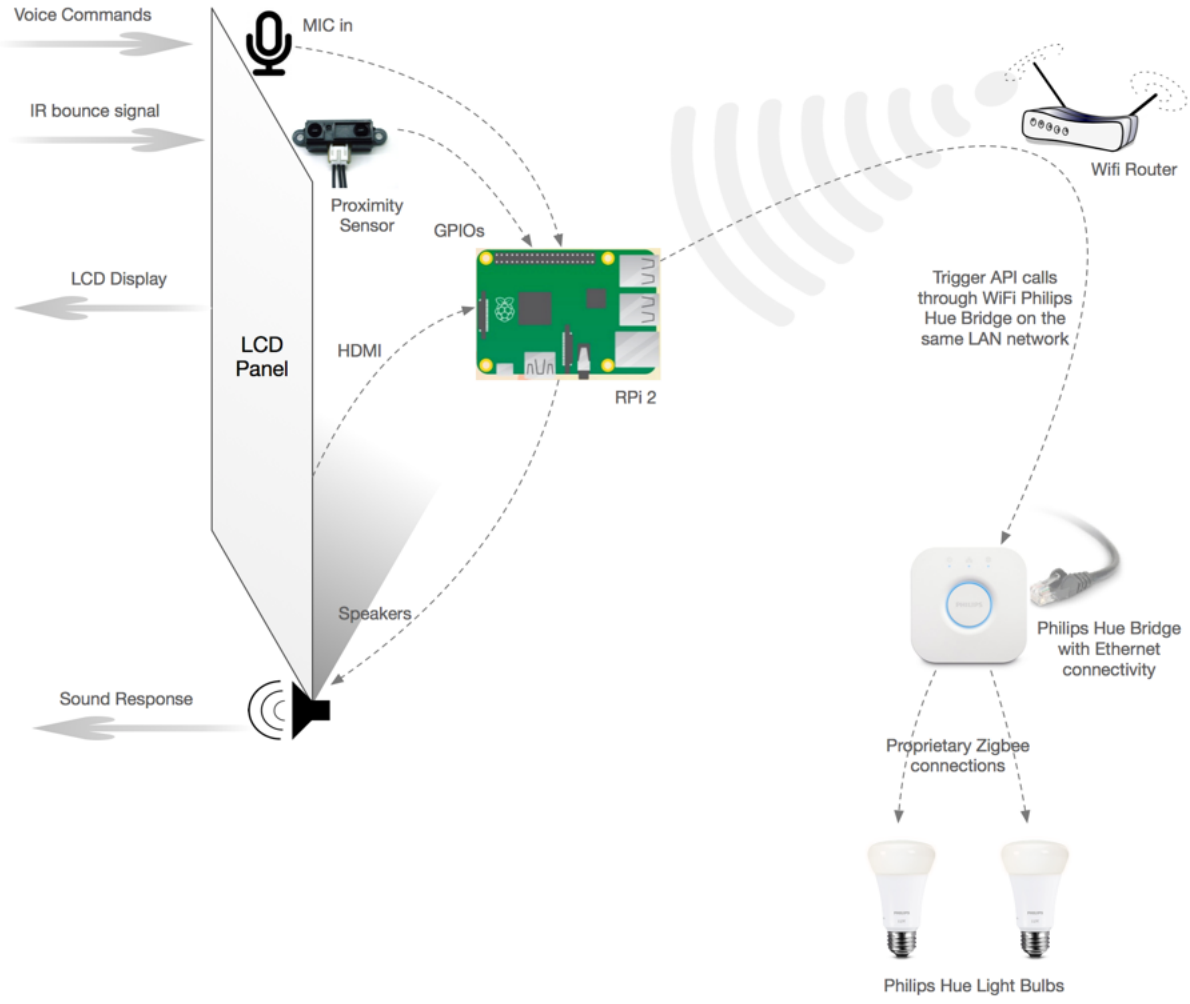

Figure 14 – System Model & Connectivity Diagram

3.4 SOFTWARE DESIGN

This diagram describes the end to end software logical design of the solution.

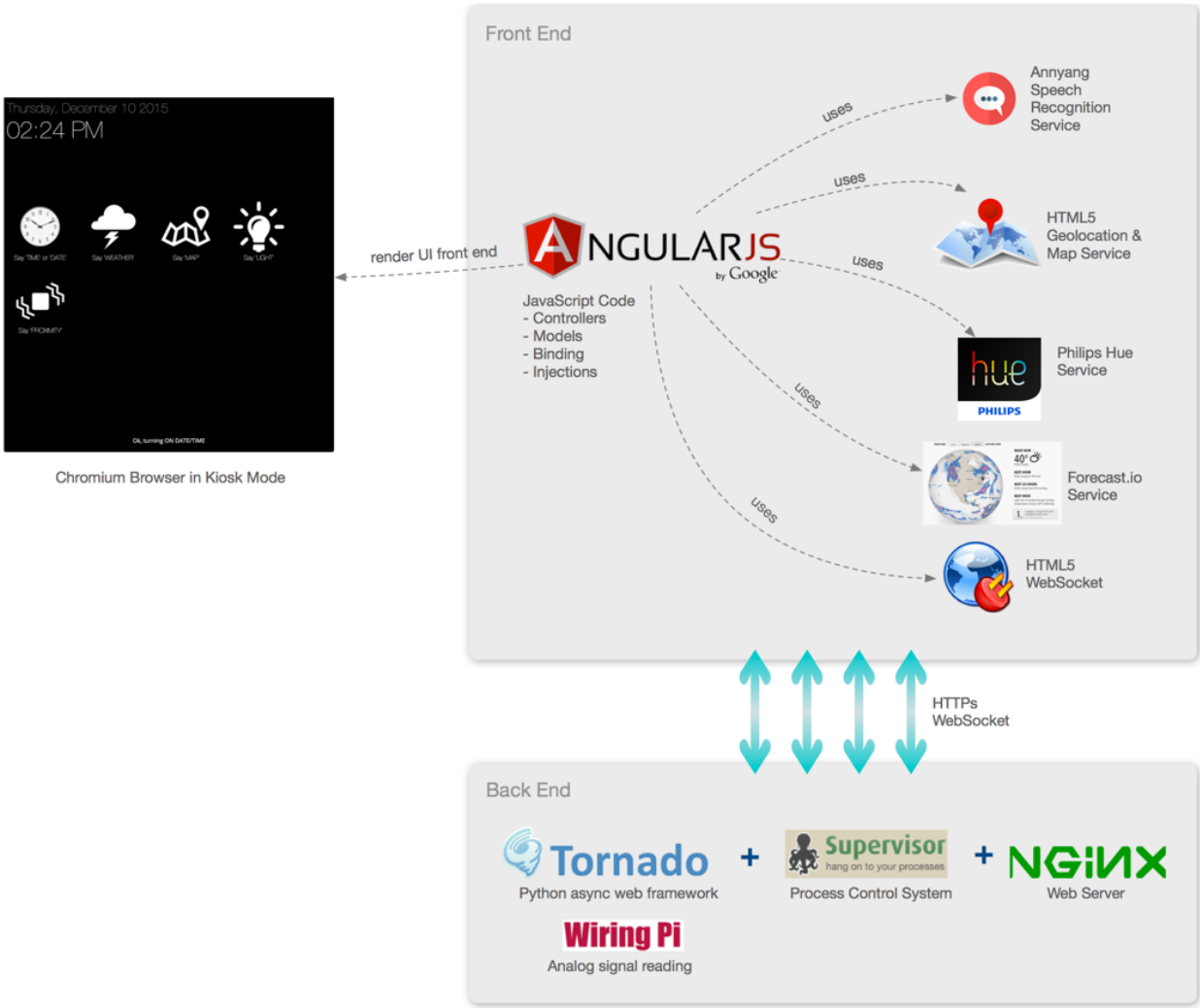

Figure 15 – Software Design

The front end and back end software are both hosted on the Raspberry Pi.

- The **Chromium browser** is configured to run full screen in Kiosk Mode upon the RPi boots up.
- The **front end** is written in JavaScript, using AngularJS framework as the backbone. All other services are injected into the AngularJS to serve different purposes:
  - Annyang service for Speech Recognition on top of Chromium engine.
  - HTML5 Geolocation and Map service from Google.
  - Philips Hue Service for API call to turning on/off IoT devices.
  - Forecast.io Service for retrieving weather information.
  - HTML5 WebSocket for interacting with the Tornado async server.
- The **back end** is written in Python, using Tornado web framework.
  - Tornado async web framework serves content through WebSockets.
  - SupervisorD is used to launch Tornado Python code as a system process (start|stop|restart|alwayson).
  - Nginx is used a reverse SSL proxy serving all connections coming from the front end.

- WiringPi is used to read analog signal from the GPIO headers connected to the sensors.

- **Notes:** all connections to the back end must be SSL-enabled because Chromium browser only persist the user's permission to access Microphone/Geolocation data if the website is on HTTPS. Otherwise, it will keep asking for approval of sharing voice or geolocation data whenever the website is loaded.

3.4.1 Sequence Diagram

This sequence diagram describes the sequential events happened to the Smart Mirror.

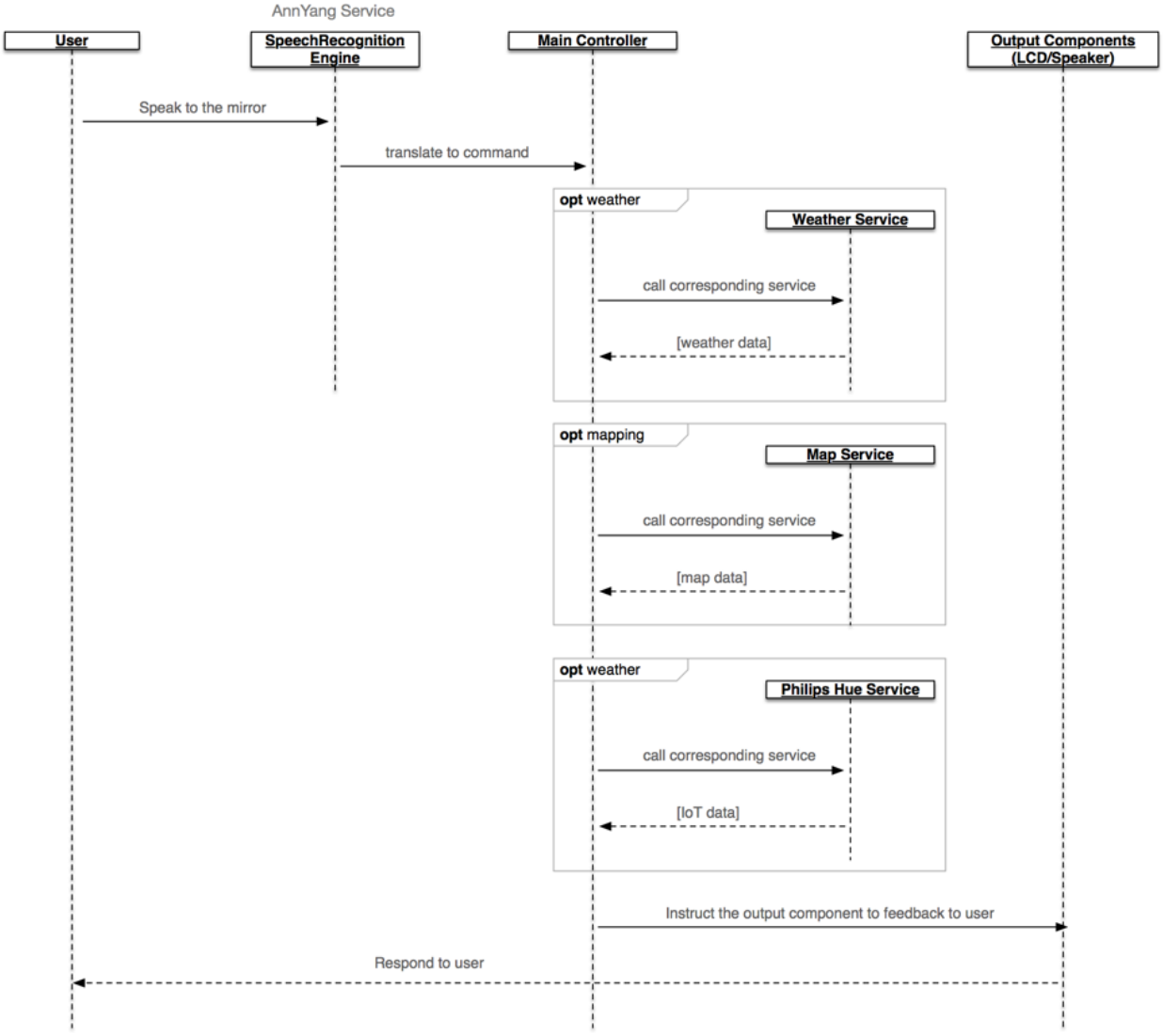

Figure 16 – Sequence Diagram

- **Note:** Philips Hue Service, or Map Service, or Weather Service will have their own procedures to obtain data or trigger some subsequent actions on behalf of the main controller. Those sequences will not be described in this diagram. The main sequence diagram is meant for describing the overall flow of the Smart Mirror only.

3.4.2 Code Structure & Practices

Disclaimer: The original framework and structure of this project is heavily inspired by the Evan Cohen's code (Microsoft guy) at https://github.com/evancohen/smart-mirror. However, I've rewritten almost every unit.

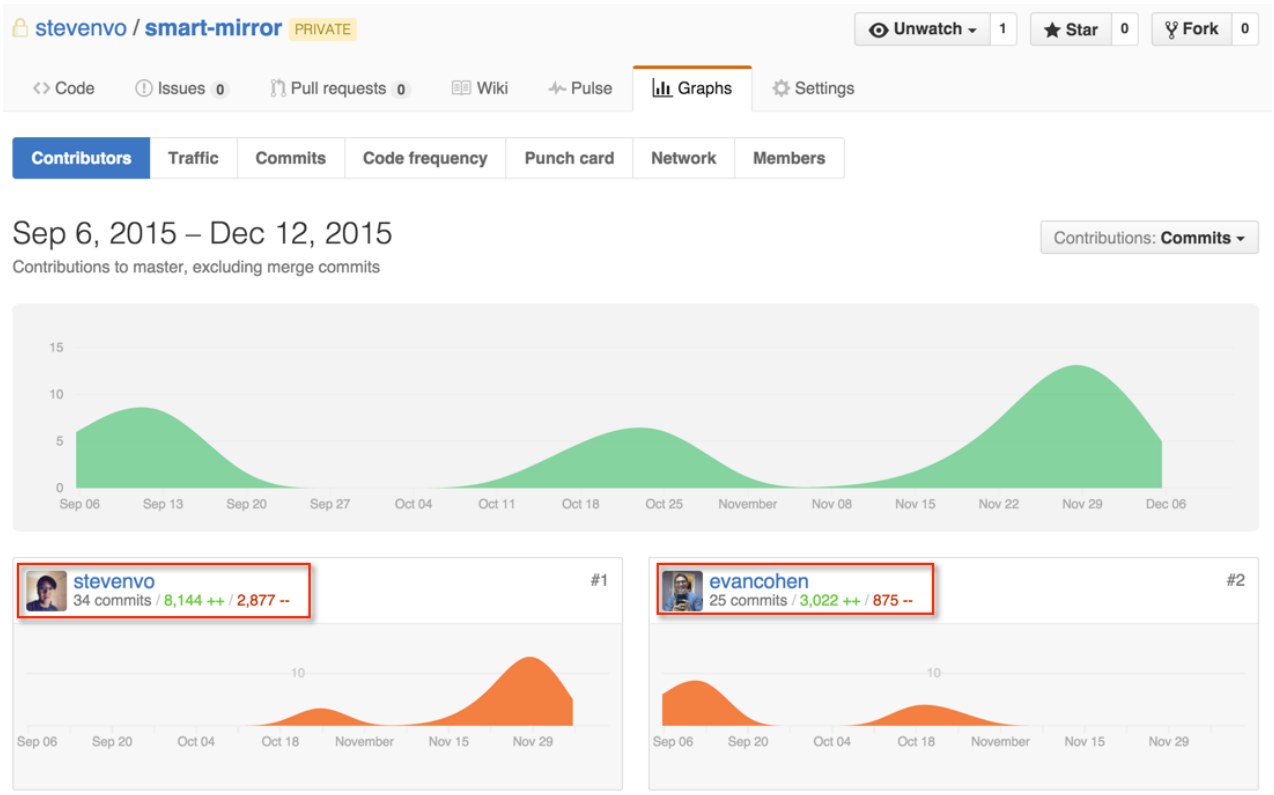

Figure 17 – Github Code Contribution Report (stevenvo v/s evancohen)

The source code is hosted on Github at https://github.com/stevenvo/smart-mirror.
This is the project structure:

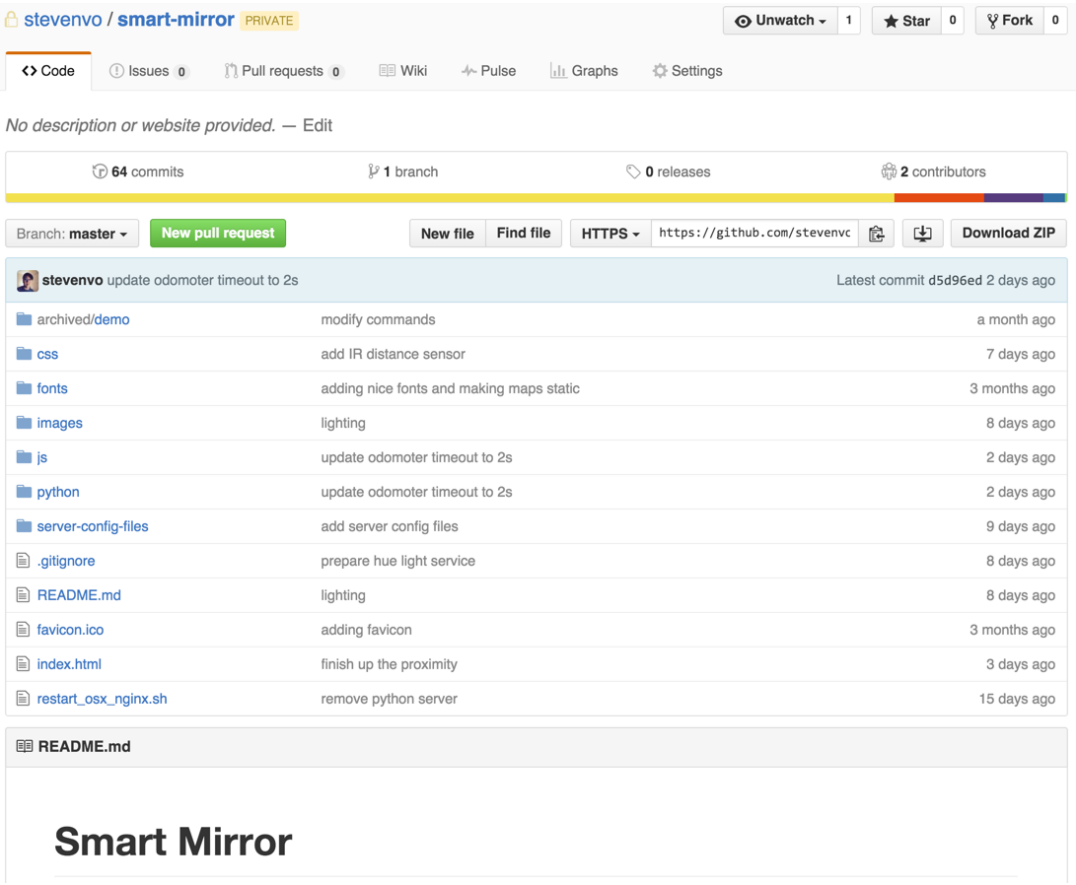

Figure 18 – Source Code Repository

Note:

- /js, /css, /fonts, /images: web resources of the mirror interface.
- /python: where the server side code locates
- /server-config-files: backup of nginx, tornado, supervisord configurations

config.js | controller.js | index.html

```
(function(angular) {
   'use strict';

   function MirrorCtrl(AnnyangService, hue, InfraDistanceService, VoiceSynthesisService, GeolocationService, WeatherService, MapService, $scope, $timeout, $window, $log) {
      var _this = this;

      $scope.listening  = false;
      $scope.debug      = false;
      $scope.complement = "Good Day!"
      $scope.user       = {};

      $scope.colors           = ["#6ed3cf", "#9068be", "#e1e8f0", "#e62739"];
      $scope.mirror_response  = "Say something, or say 'MENU' for help...";

      $scope.last_distance       = 0;
      $scope.last_distance_count = 0;
      $scope.distance            = 0;

      $window.odometerOptions = {
         auto: false, // Don't automatically initialize everything with class 'odometer'
         // selector: '.my-numbers', // Change the selector used to automatically find things to be animated
         // format: '(,ddd).dd', // Change how digit groups are formatted, and how many digits are shown after the decimal point
         duration: 2, // Change how long the javascript expects the CSS animation to take
         // theme: 'car', // Specify the theme (if you have more than one theme css file on the page)
         animation: 'count' // Count is a simpler animation method which just increments the value,
      };

      /**** Light states ****/
      $scope.lights = [];

      /*****************************************/
      /* Default display flag for each section */
      /*****************************************/
      $scope.sections = {
         display_complement   : true,
         display_time         : false,
         display_menu         : false,
         display_weather      : false,
         display_map          : false,
         display_sleep        : false,
         display_light        : false,
         display_infrared     : false
      };
```

Figure 19 – Controller structure in AngularJS Framework

Note:

- AngularJS is a very strong framework for incremental development. It has the concept of service injection which allows the developer to incrementally add more services into the same class. For example, “hue”, “InfraDistanceService”, “VoiceSynthesis”, etc. are injected into the main controller. The service logics will be kept at the Service itself not in the main controller for better abstraction and easier maintenance.

3.5 THIRD-PARTY SERVICES IN USE

3.5.1 Weather Service

The weather forecast service is from **forecast.io**.
The API call is very simple, with below format: https://api.forecast.io/forecast/<API_KEY>/<LAT>,<LON>
The response data is in JSON format, example:


```
{
  "latitude":37.3603917,
  "longitude":-121.83585679999999,
  "timezone":"America/Los_Angeles",
  "offset":-8,
  "currently":{
    "time":1449892930,
    "summary":"Partly Cloudy",
    "icon":"partly-cloudy-night",
    "nearestStormDistance":1,
    "nearestStormBearing":163,
    "precipIntensity":0,
    "precipProbability":0,
    "temperature":48.95,
    "apparentTemperature":46.47,
    "dewPoint":43.92,
    "humidity":0.83,
    "windSpeed":5.9,
    "windBearing":308,
    "visibility":9.08,
    "cloudCover":0.31,
    "pressure":1015.89,
    "ozone":360.79
  },
  "minutely":{
    "summary":"Partly cloudy for the hour.",
    "icon":"partly-cloudy-night",
    "data":[
      {
        "time":1449892920,
        "precipIntensity":0,
        "precipProbability":0
      },
      {
        "time":1449892980,
        "precipIntensity":0,
        "precipProbability":0
      },
```


Figure 20 – ForeCast.io API Response Data

3.5.2 Philips Hue Lighting Service

The Philips Hue service is served by the local bridge, with simple API call as below:
https://<bridge-ip-address>/api/<api-key>/lights

```
{
   "1":{
      "state":{
         "on":true,
         "bri":191,
         "alert":"none",
         "reachable":true
      },
      "type":"Dimmable light",
      "name":"Study light",
      "modelid":"LWB006",
      "manufacturername":"Philips",
      "uniqueid":"00:17:88:01:10:5b:9e:7f-0b",
      "swversion":"66015095"
   },
   "2":{
      "state":{
         "on":false,
         "bri":254,
         "alert":"none",
         "reachable":true
      },
      "type":"Dimmable light",
      "name":"Dining light",
      "modelid":"LWB006",
      "manufacturername":"Philips",
      "uniqueid":"00:17:88:01:10:5b:a8:c8-0b",
      "swversion":"66015095"
   }
}
```


Figure 21 – Philips Hue API sample response

3.5.3 Calculate distance value from analog proximity sensor

Since the project uses an analog proximity sensor, we have to convert it to distance value.

```
analogVal = ADC.getResult(0)
distance = (6762/(analogVal-9))-4
```

Figure 22 – Convert analog value to digital distance

4 BUSINESS MODEL

This diagram describes major concepts of the business model, including resources, value proposition, partners, offering channels, etc.

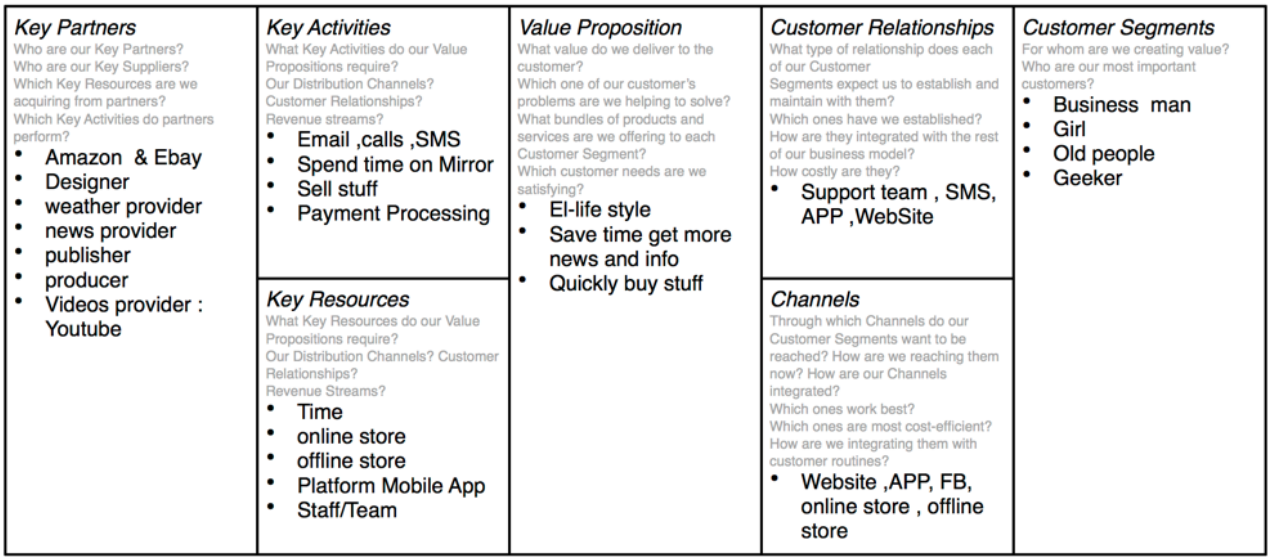

Figure 23 – High-level Business Model

5 PROJECT MANAGEMENT

5.1 PROJECT TASKS

Scope of work only include documentations. Development has been completed.

- Prepare high level requirement items
- Prepare Use Case diagram
- Prepare Sequence diagram
- Prepare Business Model chart
- Prepare System Model diagram
- Merge and review the documentation.

Tasks allocation is assigned by Steven during meeting or via email.

5.2 TEAM ORGANIZATION

Project Team Role	Project Team Member & Student ID	Responsibilities
Team Lead	Doan Quoc Sy Vo (Steven) 150201133	Development & documentation of all sections.
Team Member	Sen zheng 140301103	Prepare diagram, section write-up.
Team Member	ji li 1503070002	Prepare diagram, section write-up.
Team Member	Fey 150101006	Prepare diagram, section write-up.
Team Member	Jizhou Wu 140301122	Prepare diagram, section write-up.
Team Member	Alan 140301021	Prepare diagram, section write-up.
Team Member	Hanjing 130303007	Prepare diagram, section write-up.

6 CONCLUSION

Imagine a fitting room with a “smart” mirror having a “fashionista” taste that suggests a pair of blue jeans to go with the white shirt you just brought in without much hassle trying. Or a

bathroom mirror that can detect your skin condition and advise you to see your dermatologist (with your skin data sent to your doctor's computer immediately with your approval). That's still a long way to go from the "smart mirror" we have here. But we know it is a very near feature where the mirror or any personal belonging will be connected and sending live data to improve the human's lives and health.

Future Prospective

The project just came with a quick implementation so It has some inevitable shortcomings, for example the microphone can be improved further.

- Voice recognition plays a very critical part in this type of single-input-channel product. Improving the speech recognition involves in improving the accuracy of voice matching (i.e. the software, the machine learning) and also depends on the hardware (i.e. the microphone). That's the reason why the Amazon Echo has to install an array of 7 microphone in its device (ifixit).
- The mirror can be more transparent and durable if we can find a good two-way mirror. Unfortunately, most glass shops do not sell this type of mirror any more and we have to make a mass order if we need them. This is more of a logistics challenge that the business needs to tackle if we push forward for production.
- Besides, finding the market will be a challenge for the team as the mirror seems to be a very common household item that nobody really cares about. However, if we do it right, the "already-popularity" factor will be a great push for a fatter bottom line.

Although this is a just a small school project, we've managed to achieve quite substantial experience such as:

- Learn how a two-way mirror works and to create it from scratch.
- Learn how to use HTML5 Speech Recognition and Speech Synthesizing.
- Build an end to end product that collects 4 types of data:
    - The date/time coming from the OS.
    - The weather data coming from 3rd party API such as ForeCast.io
    - The IoT data coming from a commercial Smart-Home IoT provider such as Philips.
    - The raw data coming from an IR proximity sensor device, anticipate from schematic design to analog signal reading, then exposing the data over web service with WebSocket/HTTPS protocol and presenting to the web interface.
- Master a popular web development framework such as AngularJS.
- Design a "workable" infrastructure with Nginx, Tornado, SupervisorD, WebSocket.