IOT Based Garbage Monitoring System

Trends in the area of Waste Management

Current Technology:

- Rapid development in the field of smart home.
- Concentrated at home level
- Device manufacture: SimpleHuman.
- Features: Voice and Motion control.

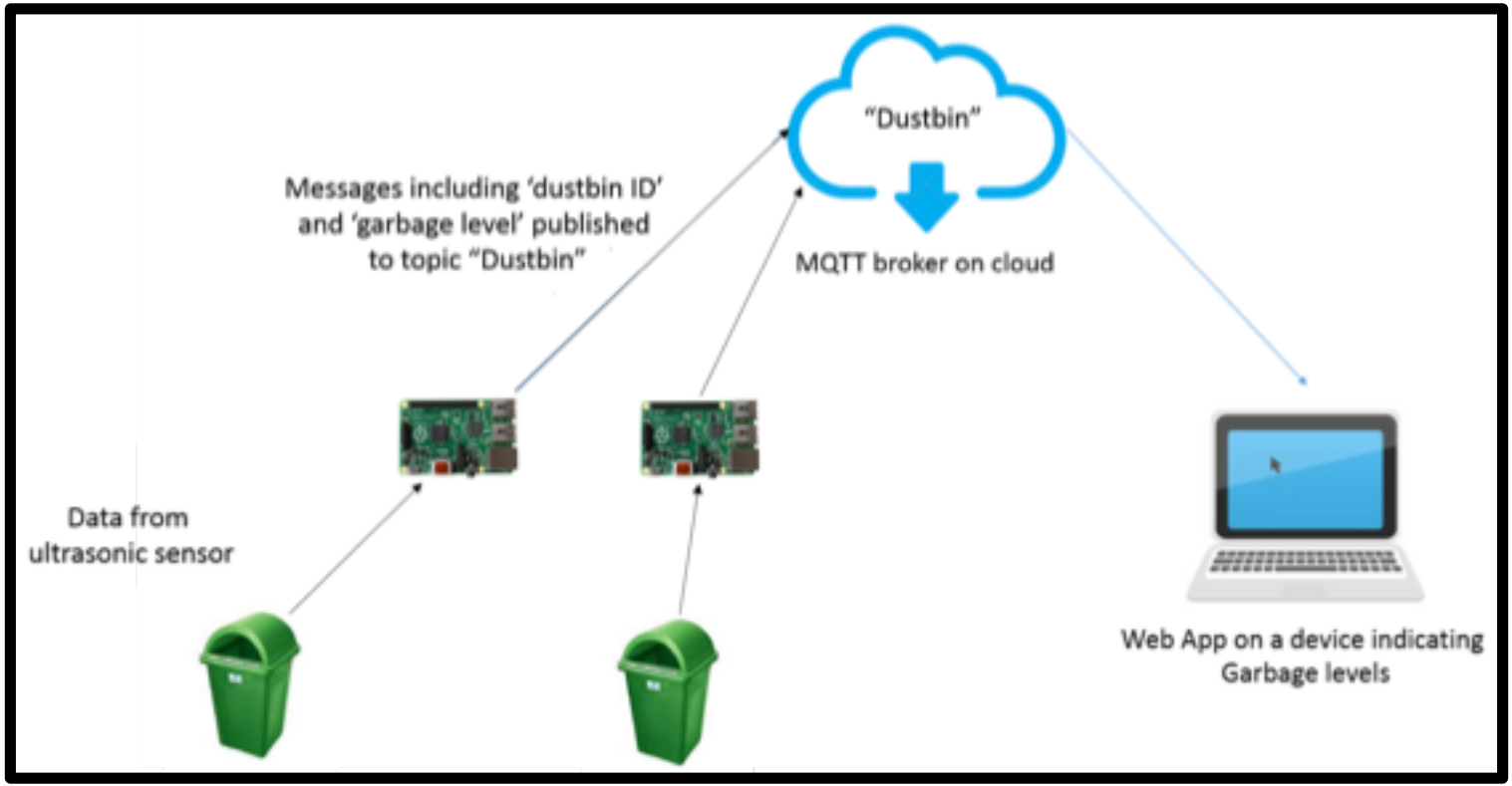
Need:

- Concentration at the public places
- Severe problem at the collection of trash
- Preventing health diseases in the Neighbourhood
- Efficient and managing the resources

Goals

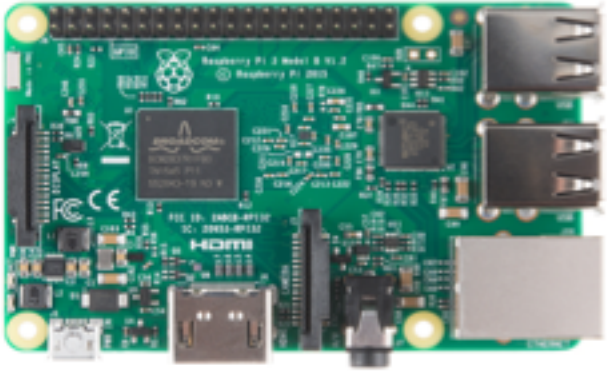
- Need a monitoring system at city level for the garbage collection
- Collection of the waste at a time when the dustbin is full.
- Closing the lid when the dustbin is full, thus preventing overflowing.
- Easy interface for the user to operate
- Create rich datasets for city planners

Overall Architecture



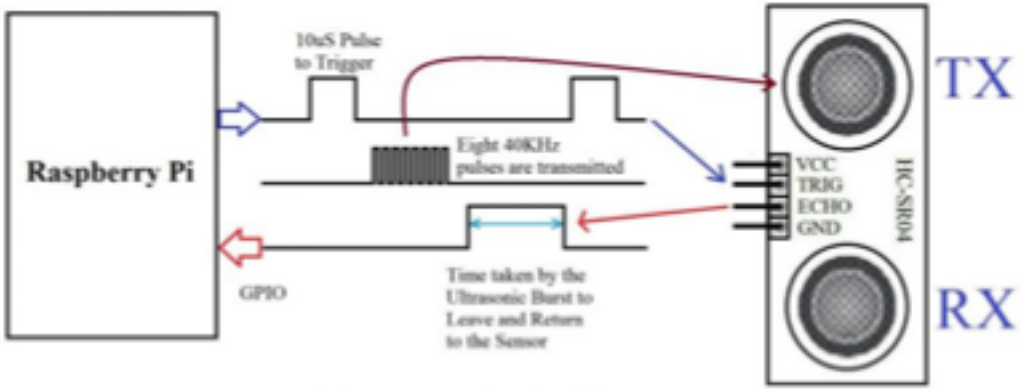
Hardware Development

Controller



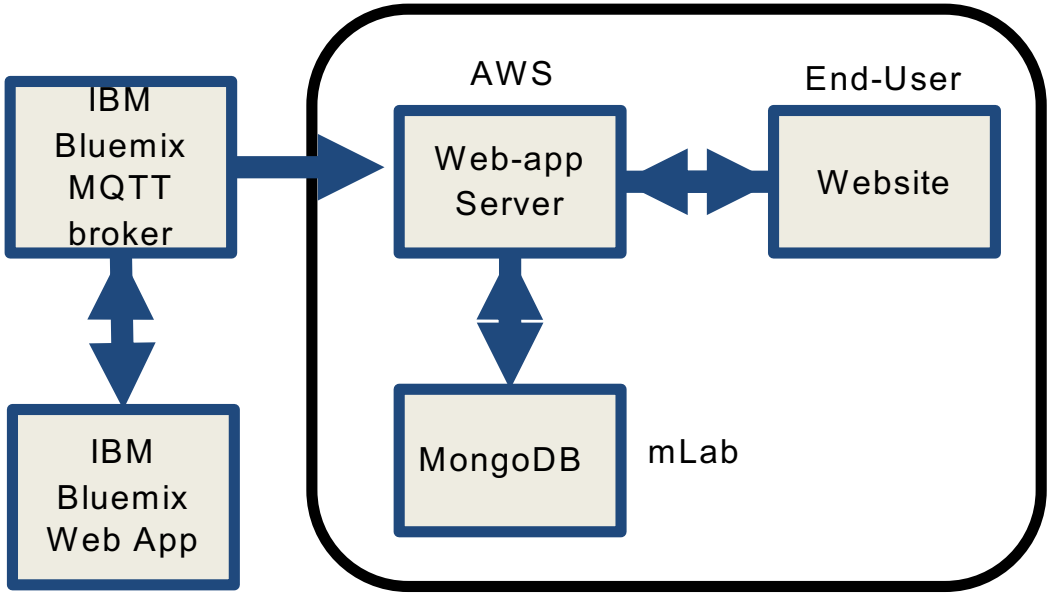
HC-SR04 Ultrasonic Sensor

Interface



Software Development

Software Architecture Continued



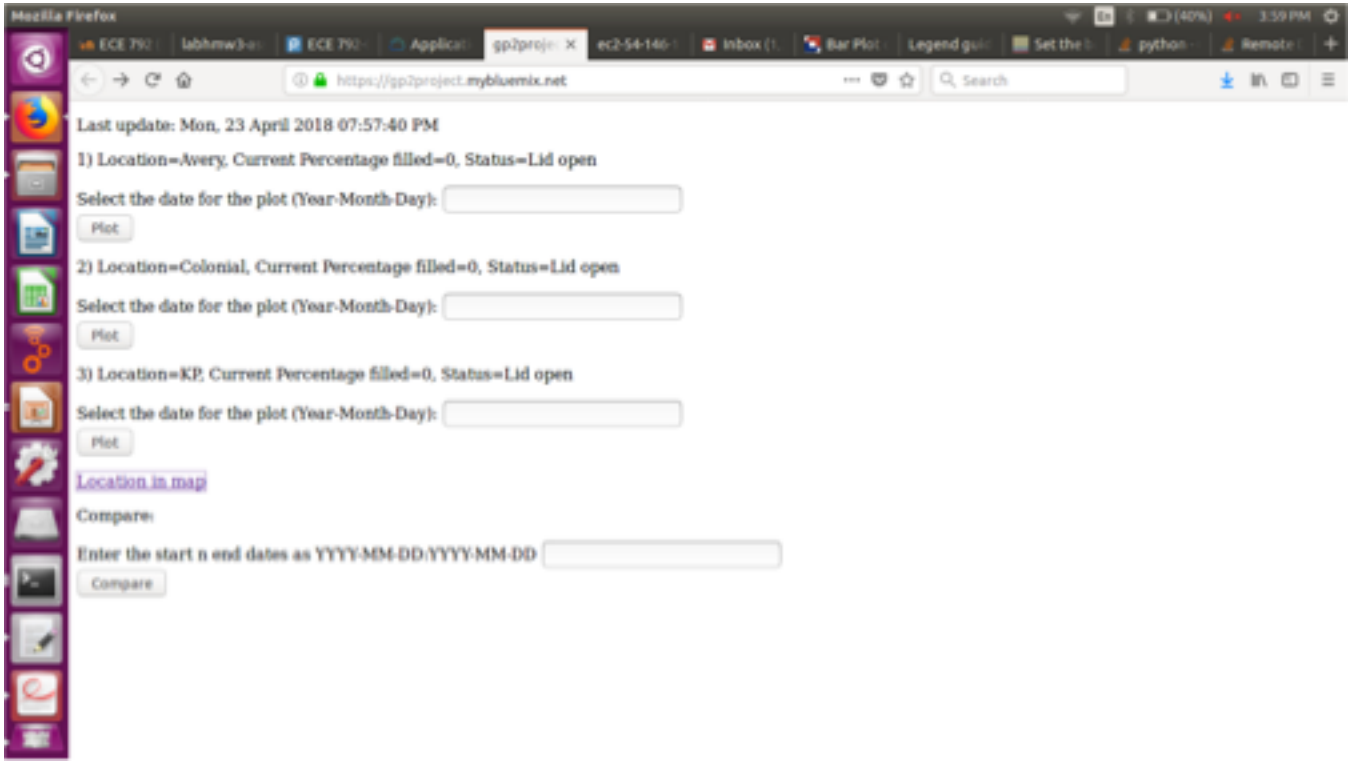
Web Application for google API

Software Stack
Backend - Node.js, Express, MongoDB
Frontend - React.js / Next.js

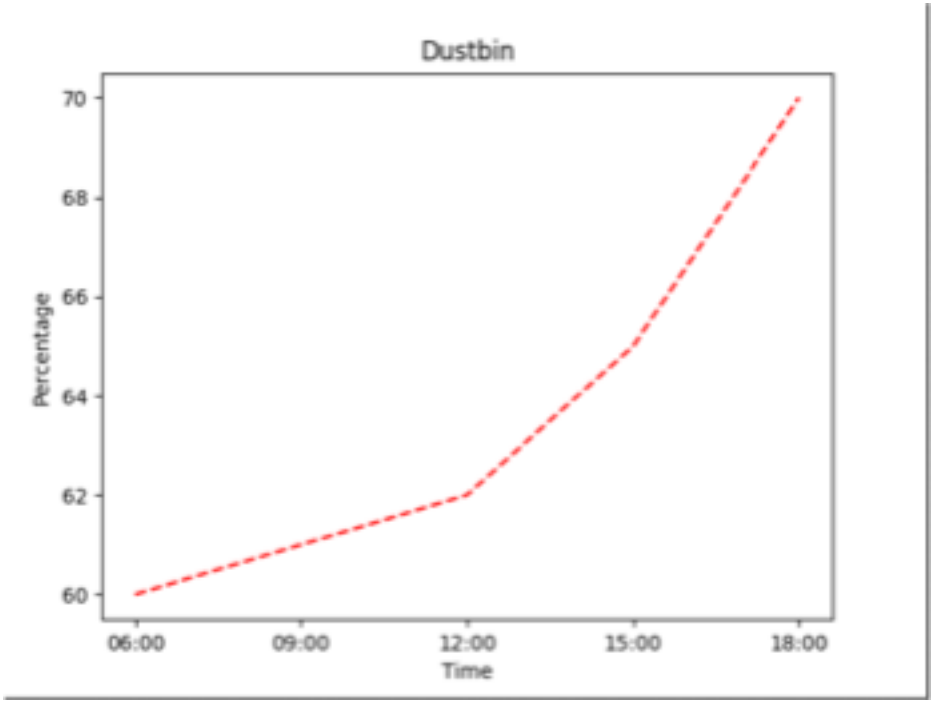
API's Used
Google Maps
mLab - MongoDB
IBM Bluemix - Node.js MQTT Client

IDE:
Cloud9 on AWS

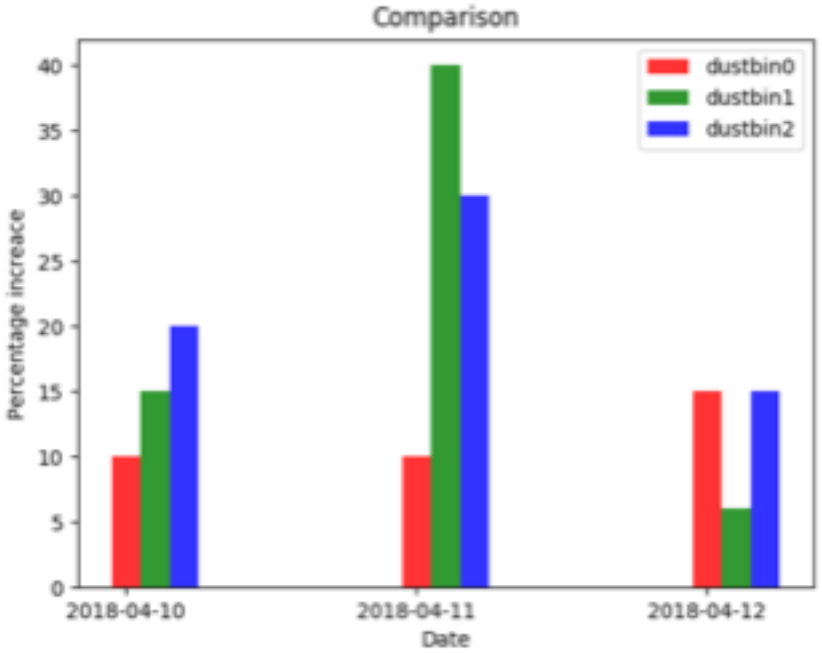
Bluemix Web App



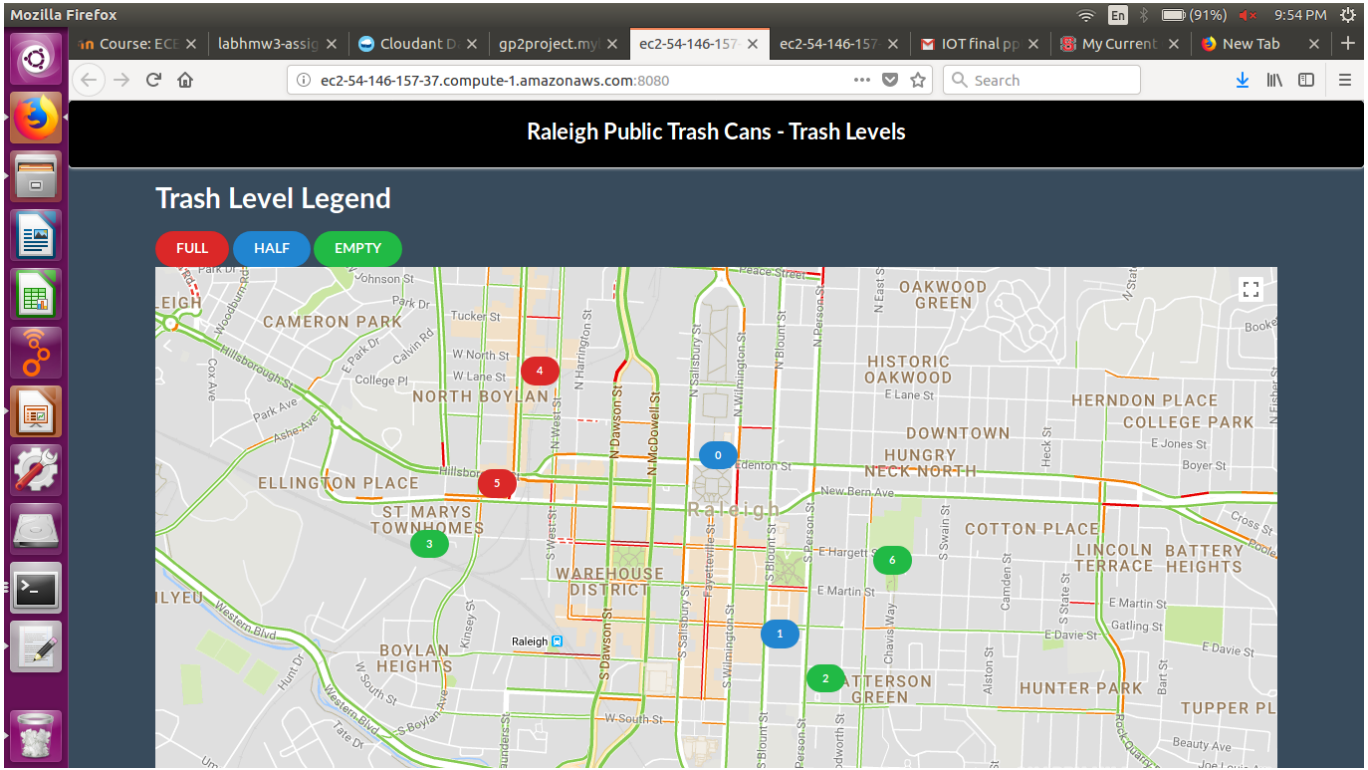
Bluemix Web App



Bluemix Web App



Web Application for google API



Demo

- Initially levels is 0%.
- When you start adding garbage the level should be reflected on ‘web app’ and ‘ibm bluemix dashboard’.
- Red LED should glow when the dustbin is full.
- Color of dustbin icons on the Google map API should change according to the levels.

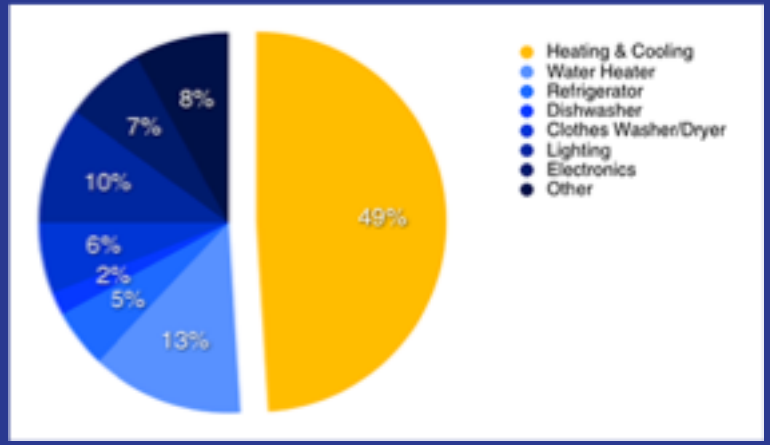

Thank you!

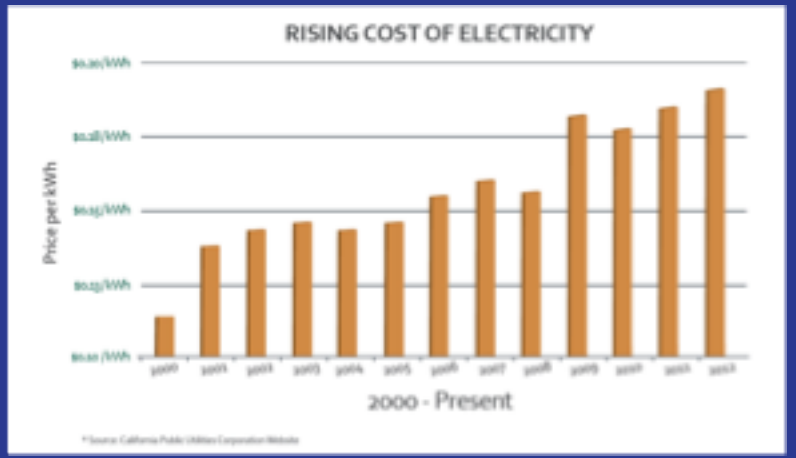
Smart Air Handling

Driving Issues

Cost Considerations:

- Heating and Cooling consumes 49% of household power
- Energy Costs are Rising

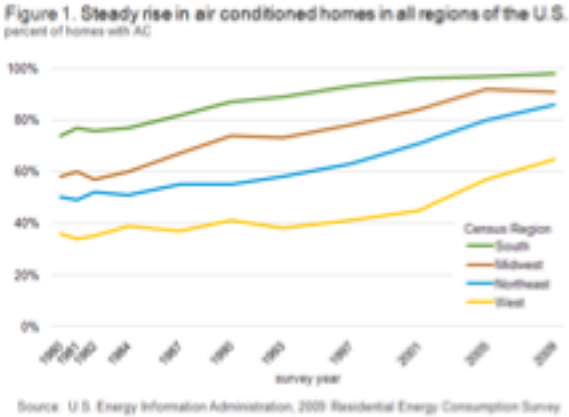


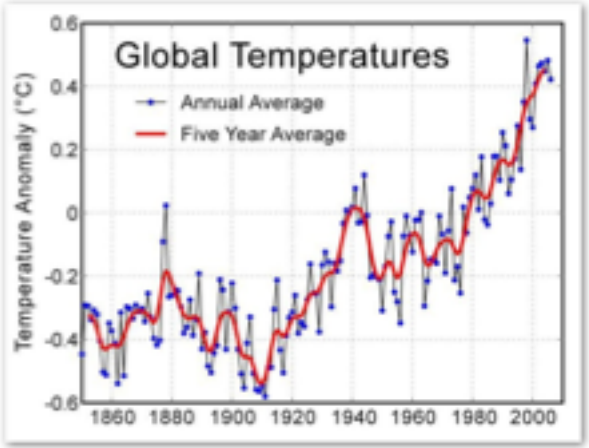


Driving Issues

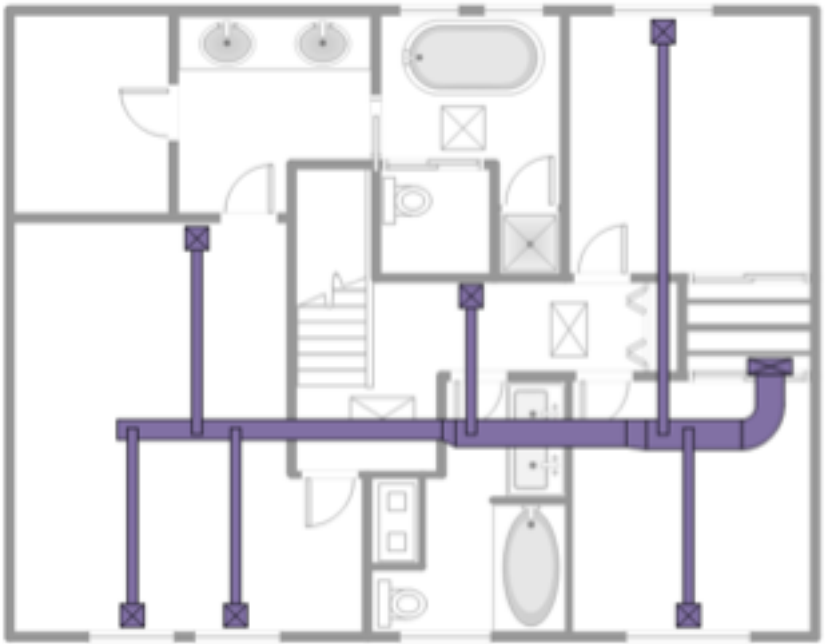
Global Considerations:

- More homes are getting HVAC
- Global temperatures are rising





Driving Issues



Comfort Considerations:

- HVAC is not “Smart”
- Rooms are often different temperatures

Solution

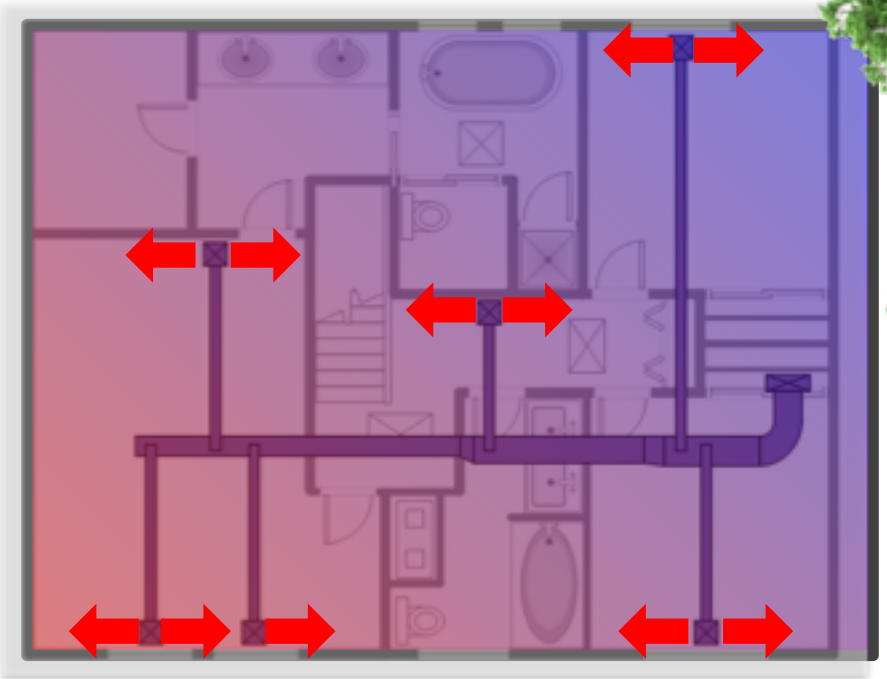
- Intelligent Thermostat
  - Use external temperature to Heat/Cool
  - Heat/Cool only when needed (Range)
- Per-Room Control
  - Don’t heat hot rooms
  - Don’t Cool Cold rooms
  - Focus Energy Where Needed
- Analytics
  - Understand where energy goes
  - React to data

Previous System

Expensive

Inefficient

Uncomfortable

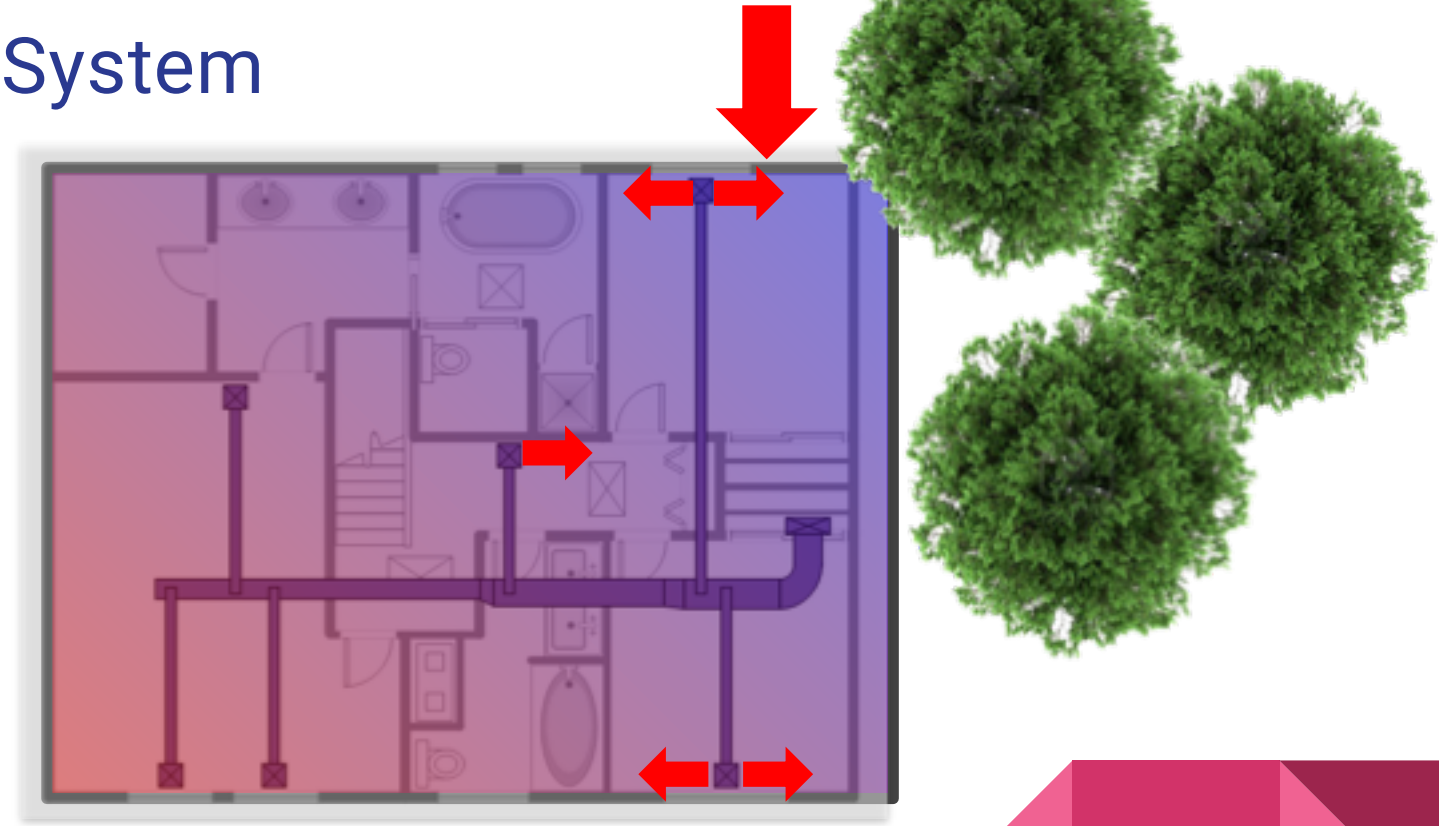


Proposed System

Reduced Cost

Efficient

Optimal Comfort

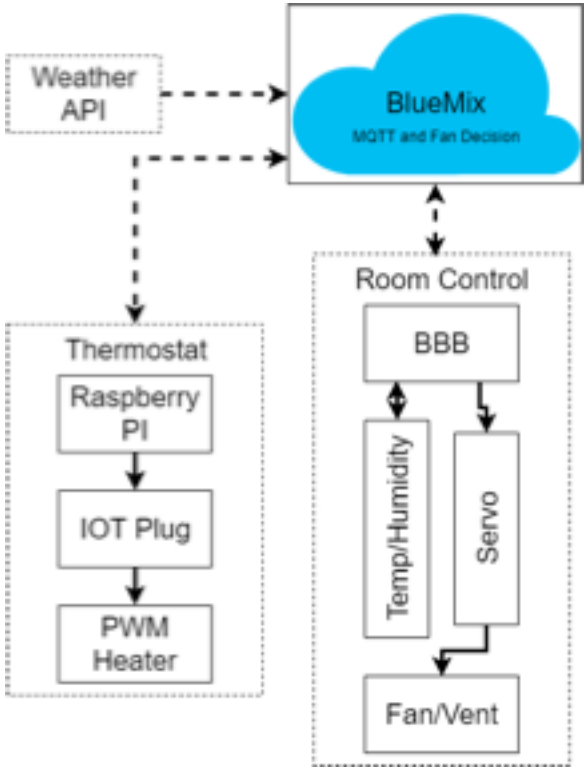


Our Solution

Goals:

- Individualized Sensing and Control
  - Open/Close Vent
  - Sense temperature in each room
- HVAC Control
  - Control Heater/Cooling
  - Use external air when possible
  - Customizable Comfort Range
- Data Logging
  - Store records of heating/cooling

System Architecture



- Server
  - Bluemix
  - MQTT
  - OpenWeatherMap.org API
- Thermostat
  - MQTT
  - IOT Plug
  - Heater
- Room Control
  - Temp Sensor
  - Servo Vent control

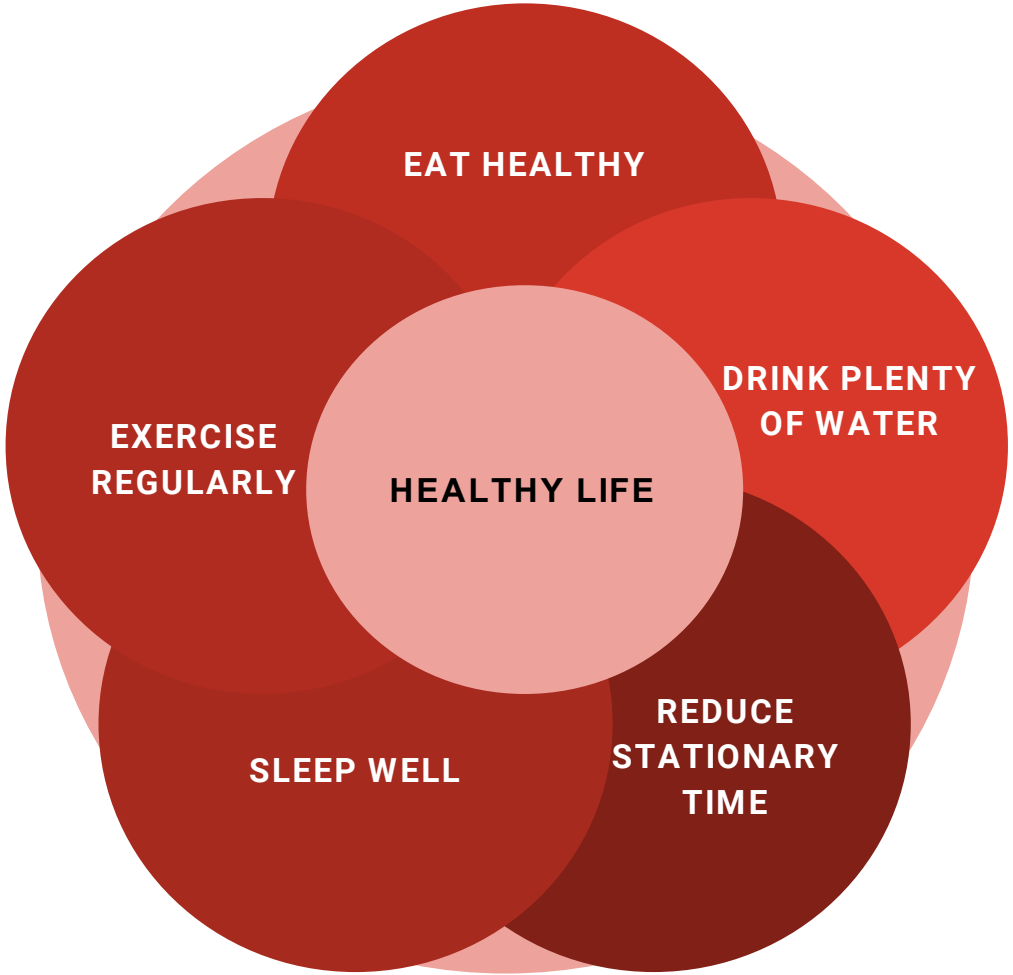
Demo

- MQTT Stream
- Comfort Range Setting
  - User Interface
  - Temperature Sensing
- Vent Response
- Heater Response

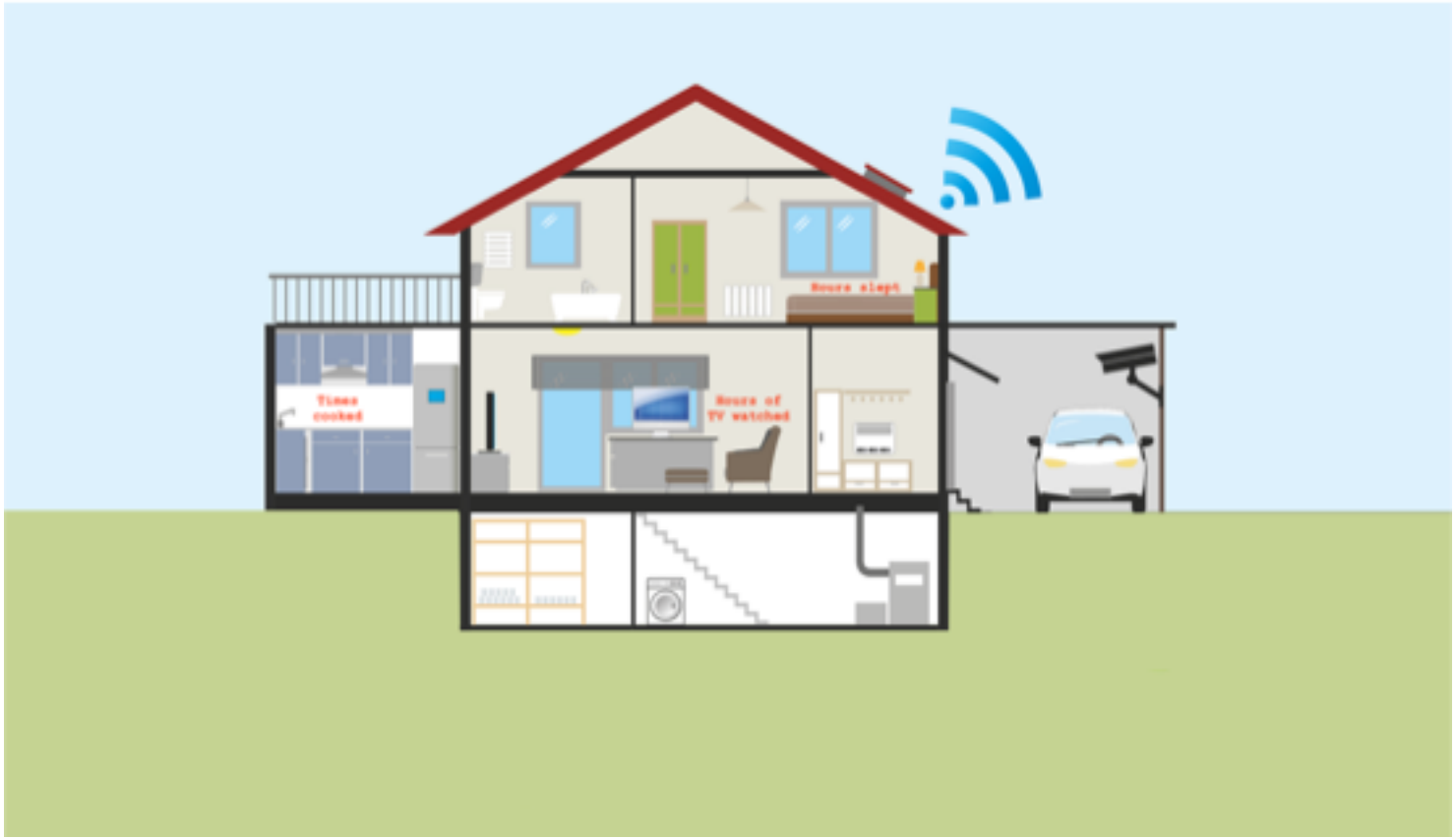
Healthy Habits Monitor

Goals

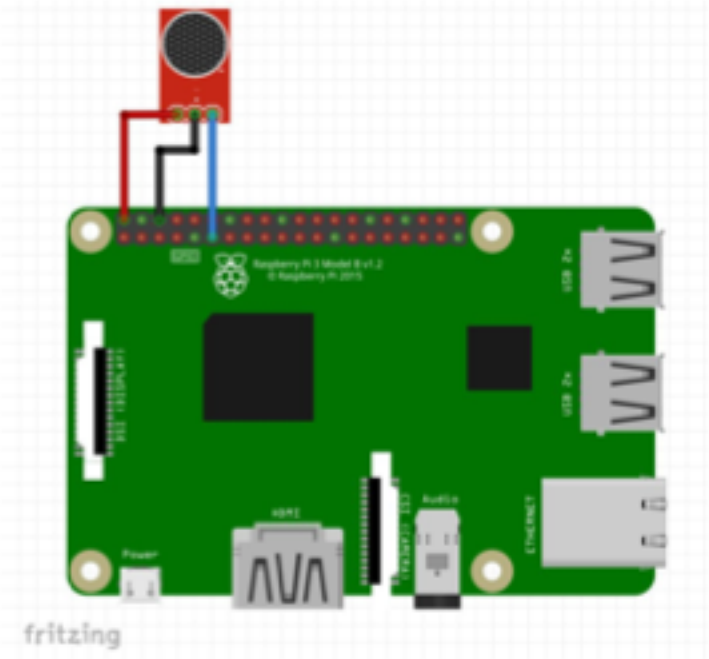
- Automated non-invasive system that monitors health habits in the home.
- Classify based on training data gathered by us and our peers against various studies in the field of health.
- Provide feedback to users as to how healthy their habits are.



Smart home

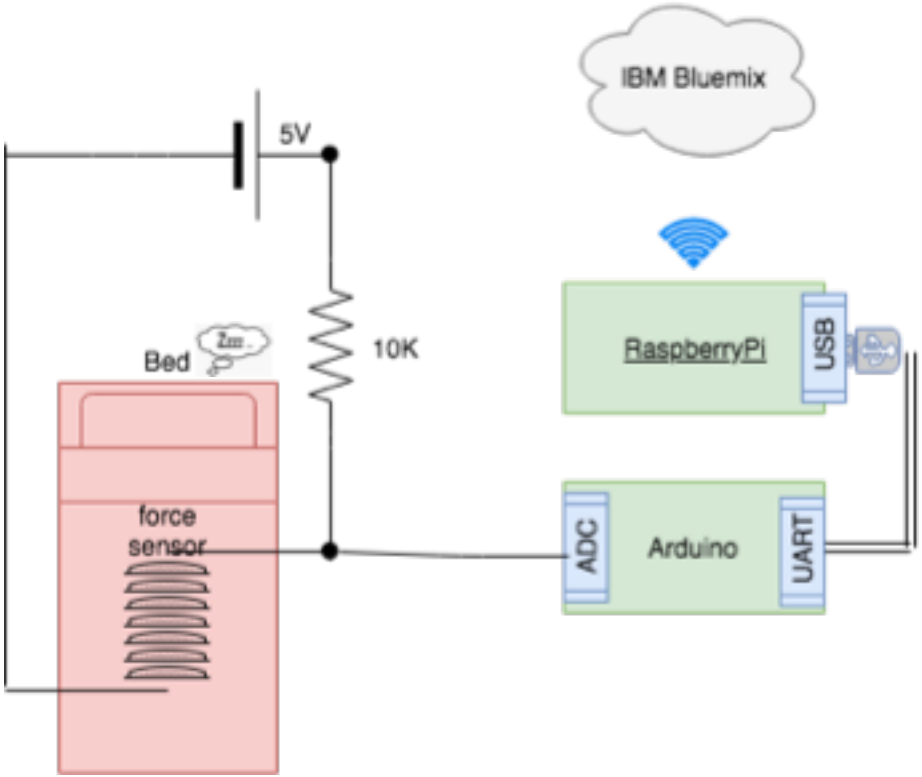


TV Tracker



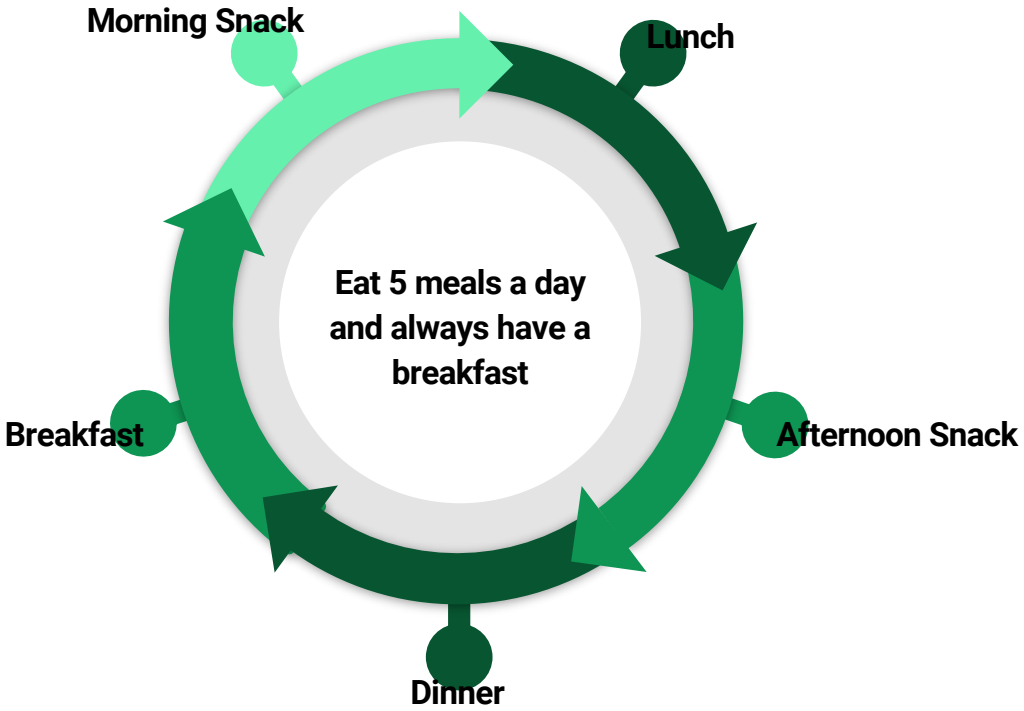
- 5V microphone with adjustable sensitivity connected to Raspberry Pi
- Sound detection triggers a timer, which will calculate the total TV time if it expires before another sound is detected
- Normalizes values and sends total hours watching TV to Bluemix once a day

Sleep tracker



- Resistive force sensor in voltage divider configuration
- Arduino serving as ADC
- Raspberry PI implements a moving average filter.
- Runs sleep detection algorithm.
- Publishes hours slept to IBM bluemix cloud once a day.

Sleep tracker data example


Morning Snack
Lunch
Eat 5 meals a day and always have a breakfast
Breakfast
Afternoon Snack
Dinner

What is a Smart Stove ?

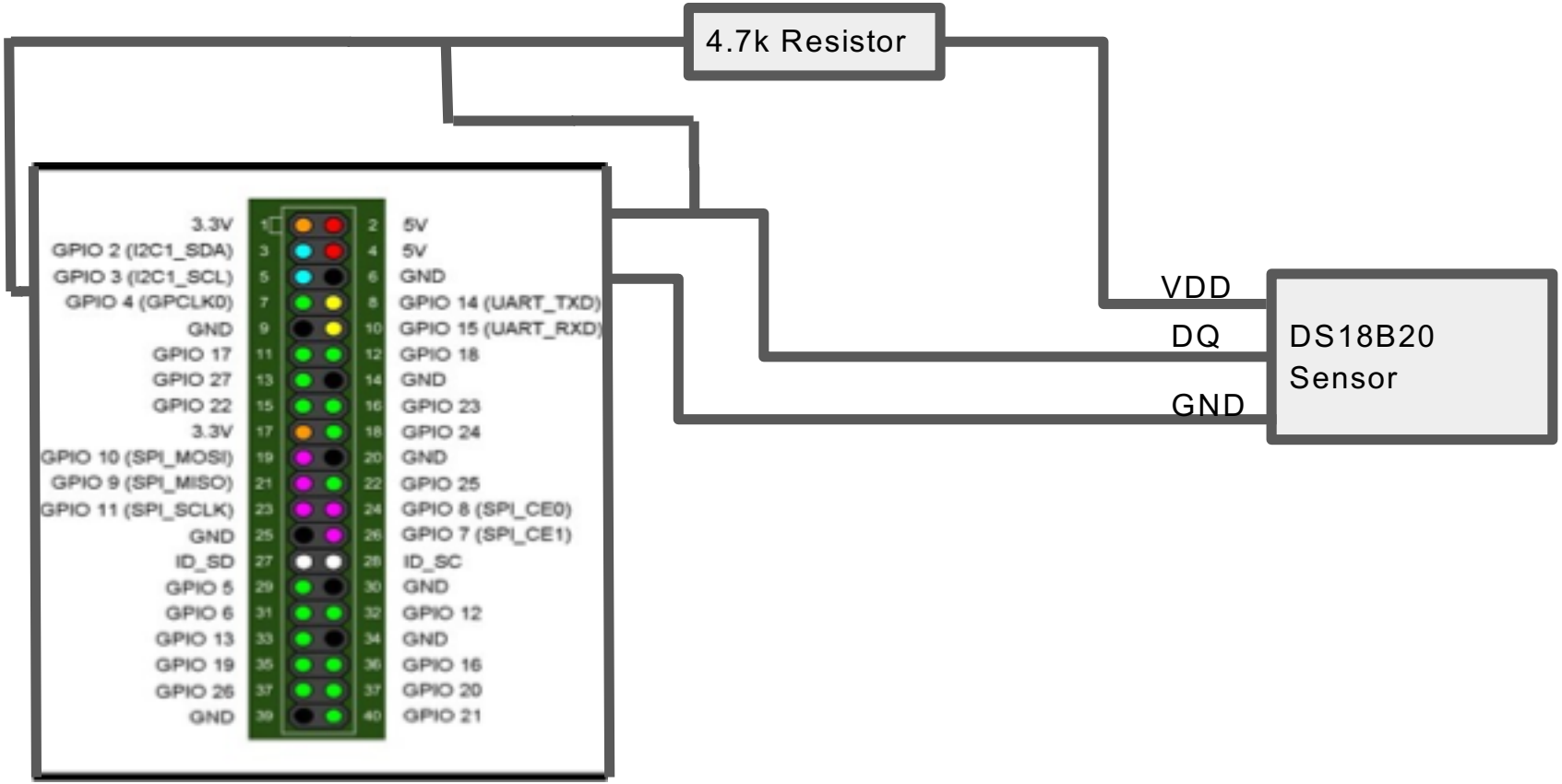
What we achieved ?

Temperature :

1. Stove On/Off (Used for Project)
   a. On : Temp >130
   b. Off : Temp <130

1. Temperature of the stove(Not in scope but displays in Pi Console)
   a. Low : 110 to 130
   b. Low Medium : 130 to 150
   c. Medium : 150 to 180
   d. High Medium : 180 to 200
   e. High : >200

Cooking times counter



Automated tracking:

Sleep Tracker

1. Monitors Sleep
2. Normalise values
3. Publishes the hours of sleep per day

+

Meals Tracker

1. Monitors cooking habits
2. Normalise values
3. Publishes number of cooked meals per day

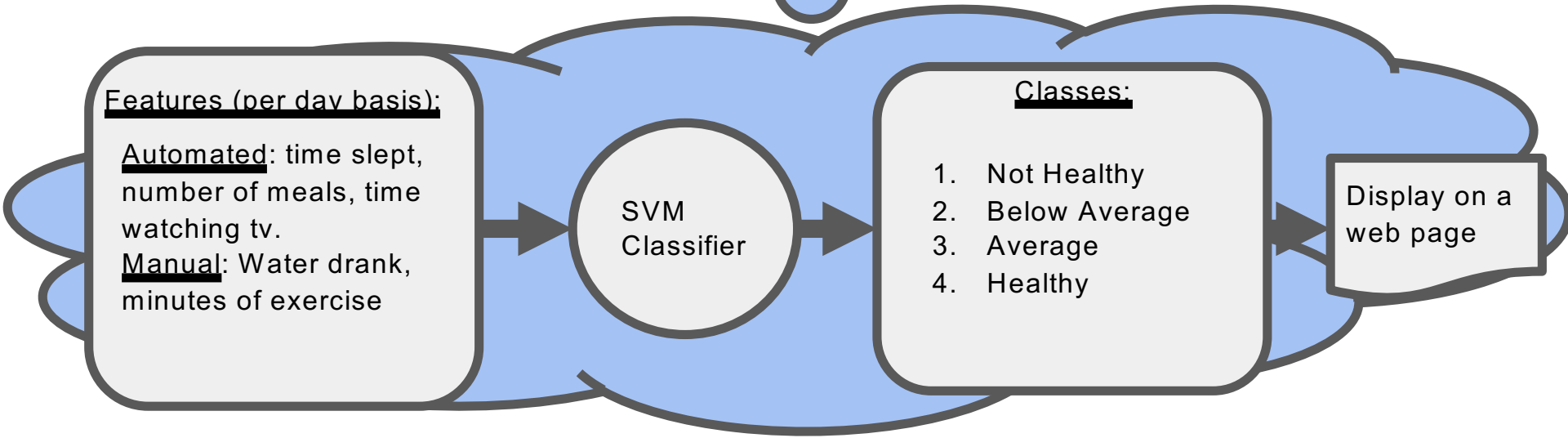
+

TV Tracker

1. Monitors watching TV
2. Normalise values
3. Publishes number of hours of entertainment per day

Subscribe to events from raspberry pi (sleep,meals,tv)

IBM Bluemix:



GROUP 7 : HEALTHY HABITS MONITOR

Hours of sleep =8

Minutes of exercise = 20

Cups of water = 8

Number of cooked meals = 2

Watching TV = 2.5

You are AVERAGE

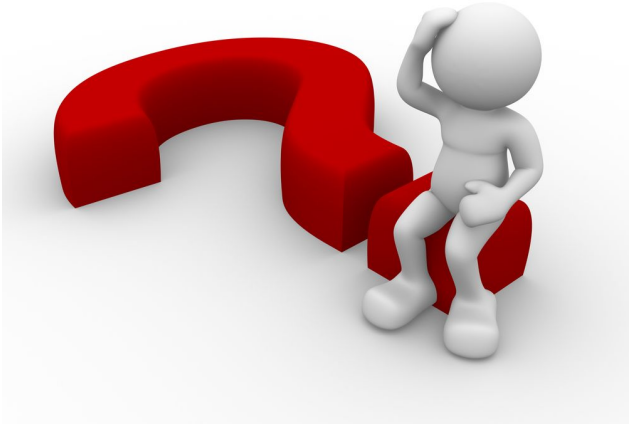
Current Time :2018-04-24 02:09:54

IntelliClock

Outline

- Introduction
- Objectives
- System Architecture
  - Front end
  - Back end
  - MQTT + Pis
- Demo Overview

Have you ever…



…Been late to work b/c of inclement weather?

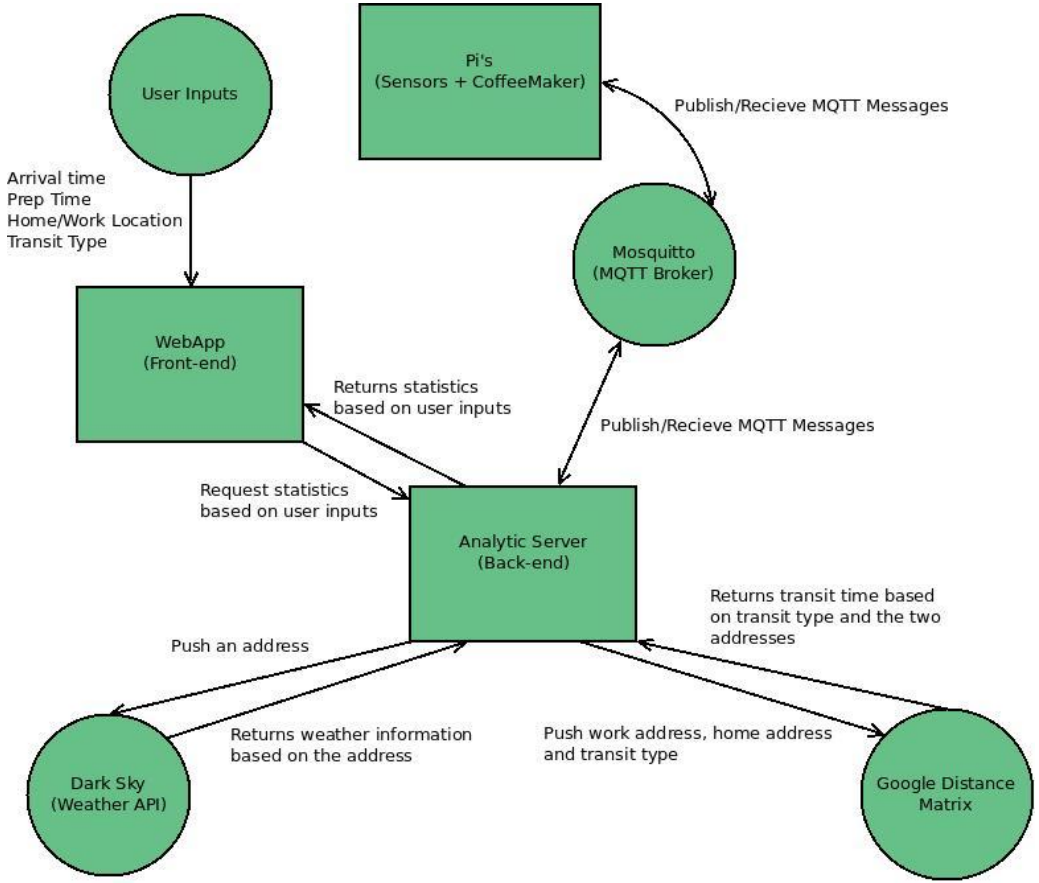
…Had your alarm go off after you already woke up?

…Been groggy in the morning b/c you didn't have time to make your coffee?

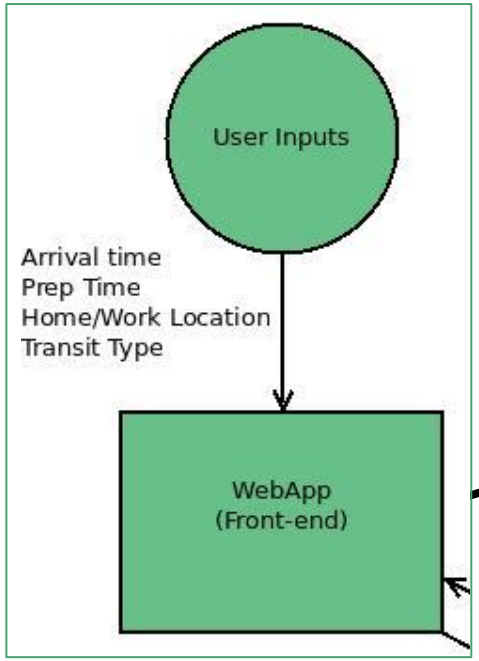
Objectives

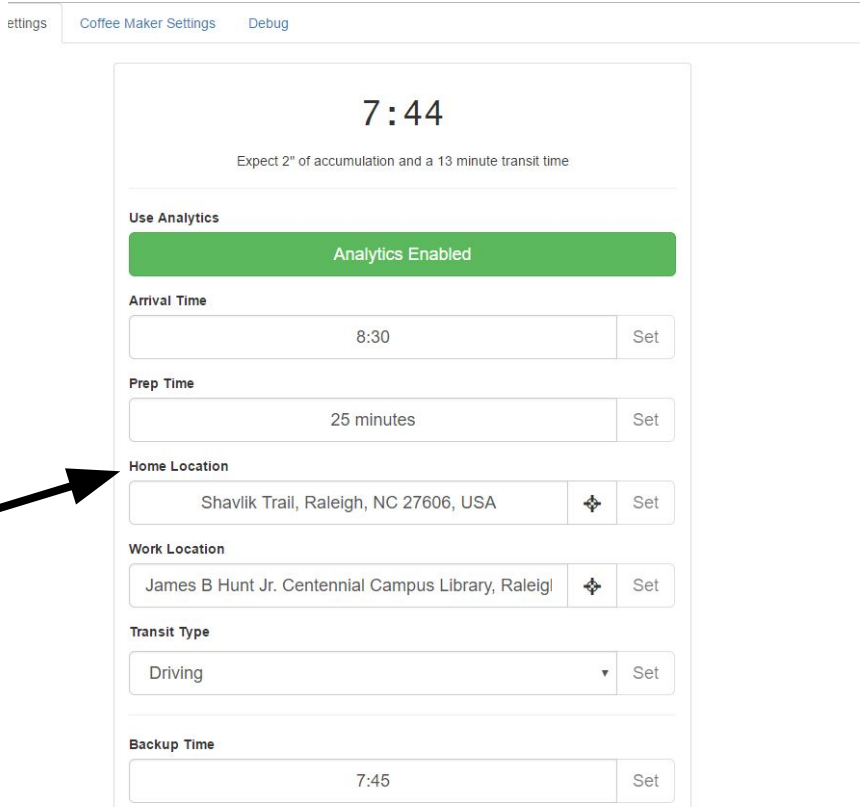
- Create an IoT Alarm Clock that can be configured online
- Set a work and home address
- Use analytics to determine how long it will take to get to work and dynamically change wake up time
  - Google Maps
  - DarkSky (weather)
- Determine if the user is already awake using a photoresistor

System Architecture

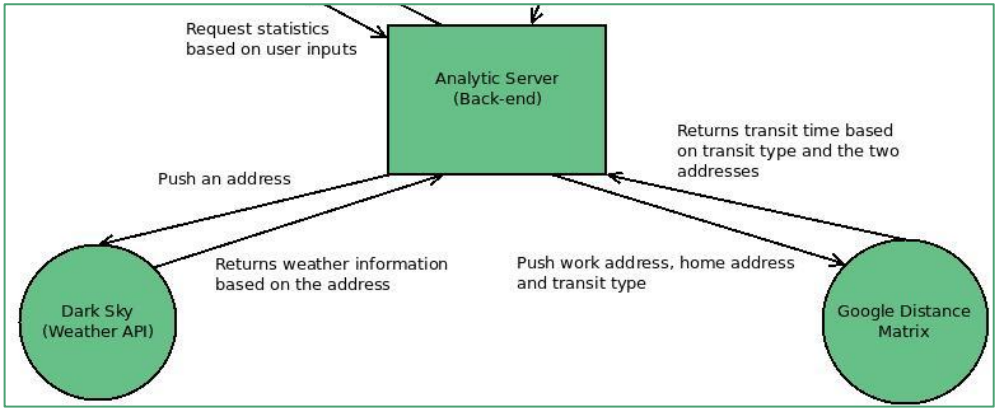


System Architecture - Front end





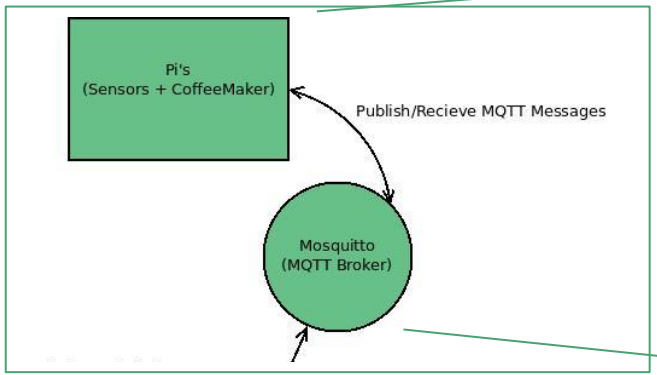
System Architecture - Back end

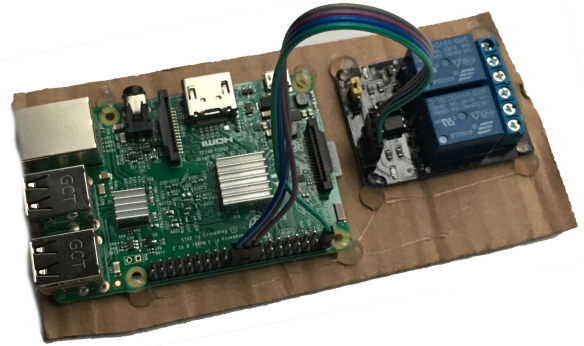


IBM Bluemix

Powered by Dark Sky

System Architecture - MQTT + Pis

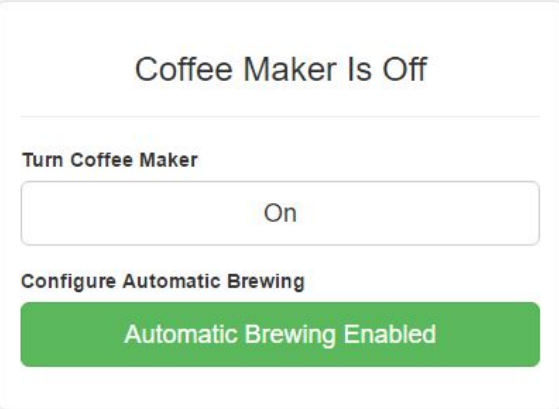


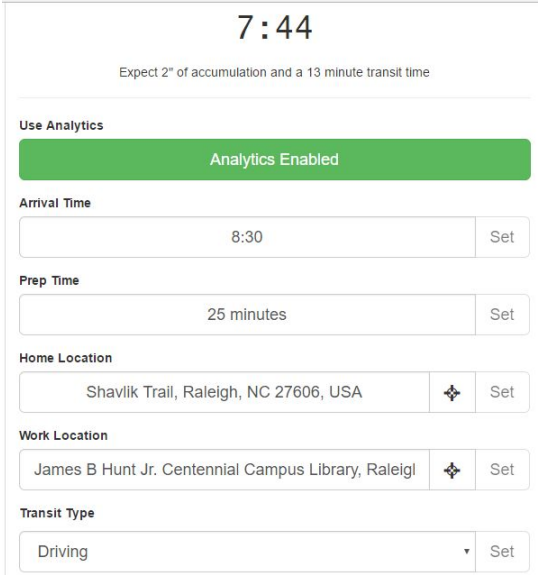


mosquitto

Demo Overview

- Allow the user to set their home address and work address
- Show the backend has pulled the correct amount of snow accumulation
- Show the backend has calculated the transit time
- Show the ability to change the “Prep Time”
- Show the ability to have the system “Start Coffee”





Demo Overview (continued)

- Show that the alarm will update every minute to account for weather/transit time changes
- Show the manual alarm clock can be set
- Show that if the light is turned on before the alarm time the alarm doesn't turn on
- If the "Make Coffee" setting is on, show that coffee begins brewing 5 minutes before the alarm turns on

Thanks for watching!

Raspbeery Pi

(Or, No One Likes Running Out of Beer)

Introduction

- Automatic beverage monitoring system
- Detects when (and how much) beverage is dispensed
- Detects when containers are low and informs a user
- Detects when stock is low (based on refills), and orders more

Goals

1. Monitor the dispensing of a beverage
2. Know when a container has been refilled
3. Keep running track of stock
4. Order new stock when low

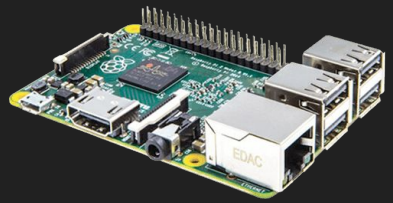
Design

Refill update pi

Refill notifications

- Manual stock adjustment
- Order prediction adjustment

Flowmeter

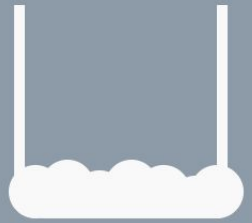


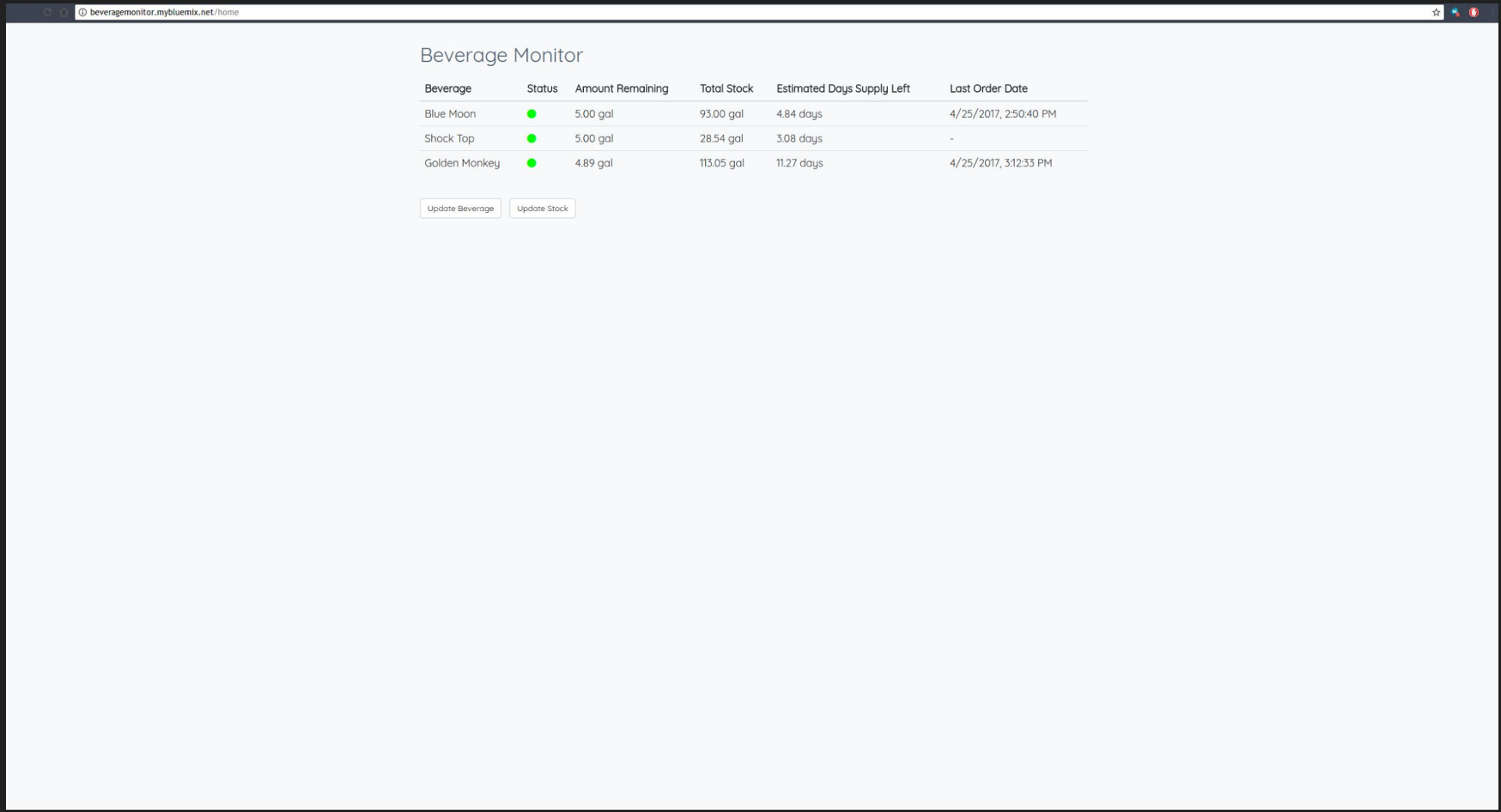
Flowmeter pi

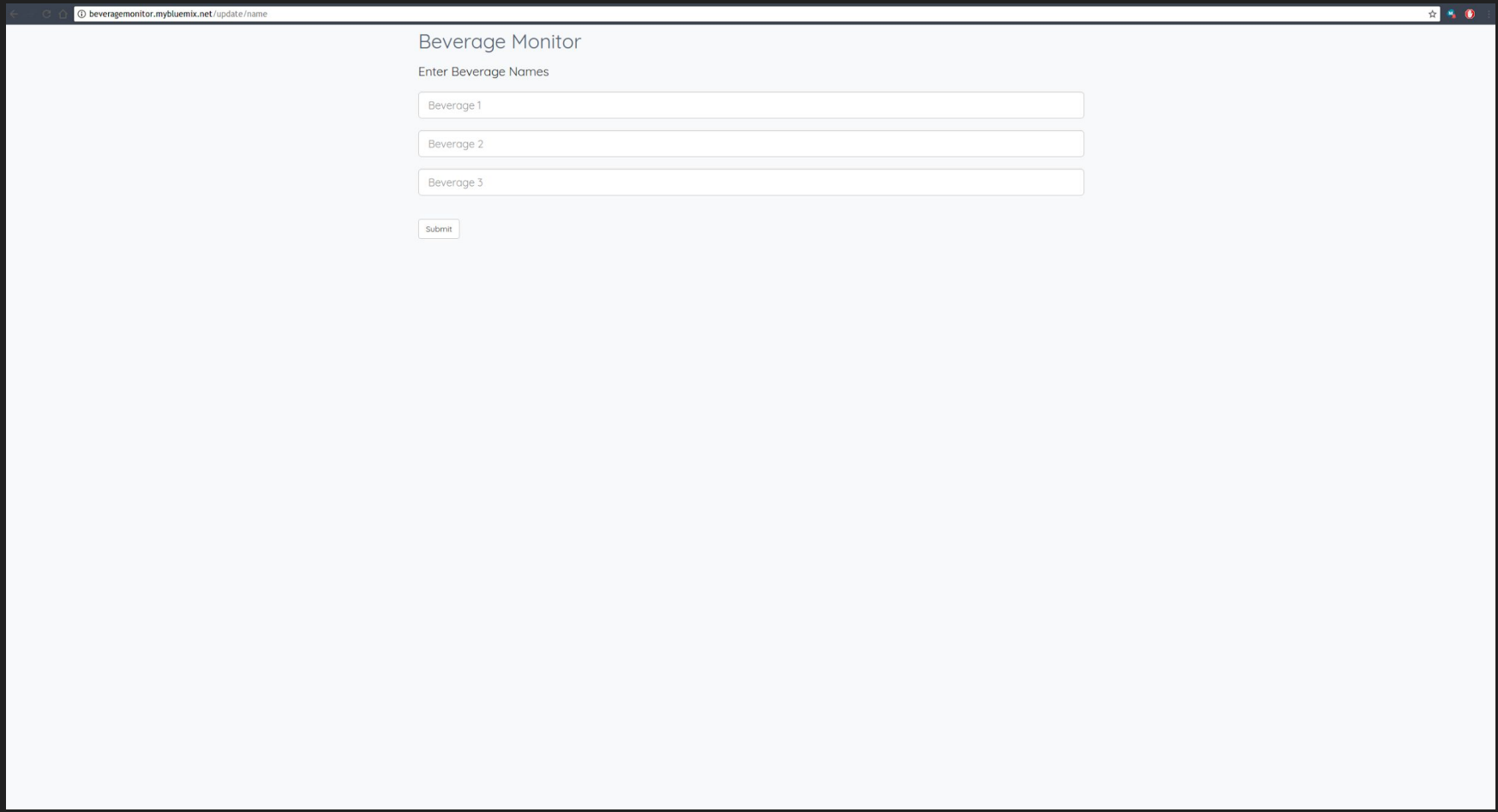
Pour updates

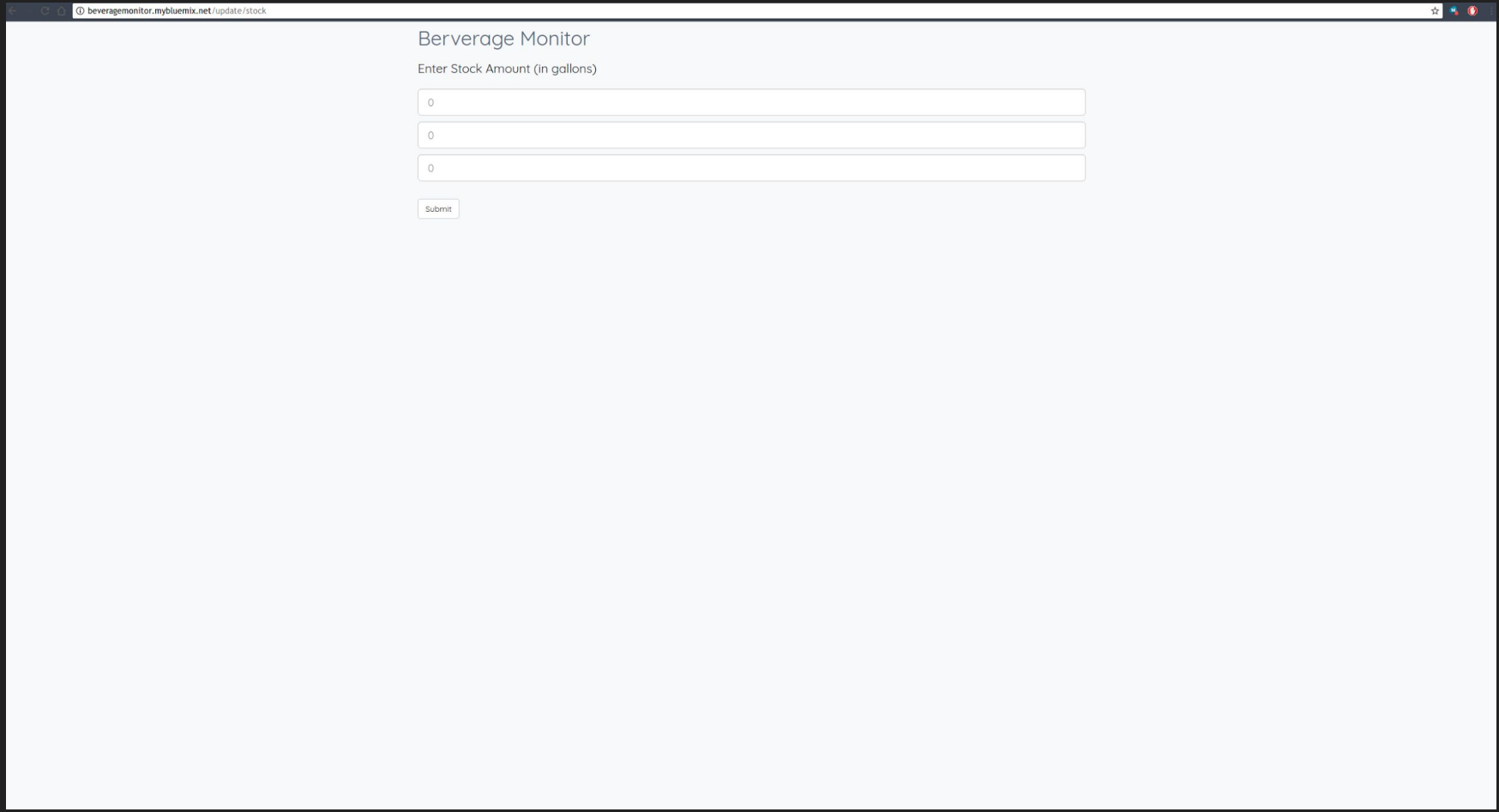
IBM BLUEMIX

- Reorder email
- Status
- Daily use tracking
- Pretty animations




beveragemonitor.mybluemix.net/home
Beverage Monitor
Beverage
Status
Amount Remaining
Total Stock
Estimated Days Supply Left
Last Order Date
Blue Moon
5.00 gal
93.00 gal
4.84 days
4/25/2017, 2:50:40 PM
Shock Top
5.00 gal
28.54 gal
3.08 days
-
Golden Monkey
4.89 gal
113.05 gal
11.27 days
4/25/2017, 3:12:33 PM
Update Beverage
Update Stock

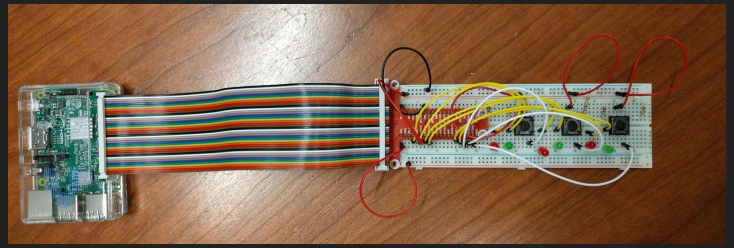

beveragemonitor.mybluemix.net/update/name
Beverage Monitor
Enter Beverage Names
Beverage 1
Beverage 2
Beverage 3
Submit

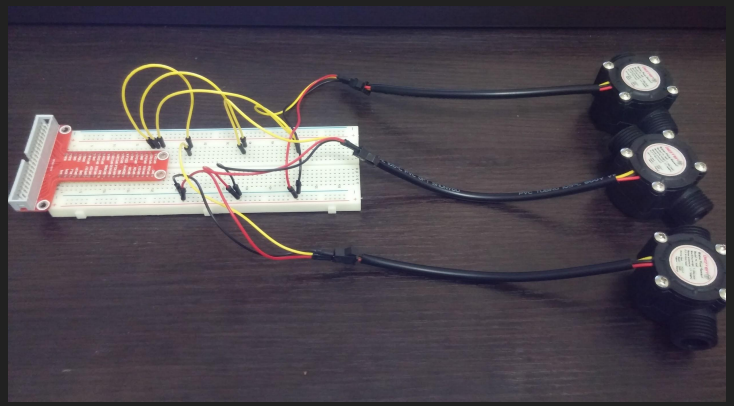

beveragemonitor.mybluemix.net/update/stock
Berverage Monitor
Enter Stock Amount (in gallons)
0
0
0
Submit

Implementation - Code

- HTML
  - Web page for stock reporting and reordering
- JavaScript
  - Parse through JSON data for displaying on webpage
- Python
  - client_amount.py
    - Reports when containers have been refilled
  - client_dispense.py
    - Reports when and how much beverage is dispensed
  - webapp.py
    - Module for data, interacts with Pi and web interface
  - web_backing.py
    - Controller for web interface

Implementation - Physical





Results

Goal: Estimate the number of days left given the amount of beer left
Goal Formula: T=mA+b
T=Days until stock is depleted
A=Amount of beer in stock

150 Customers/Day * 2 Beers/Customer * 0.125 Gallons/Beer = 37.5 Gallons/Day

Simulated mean values over a month:

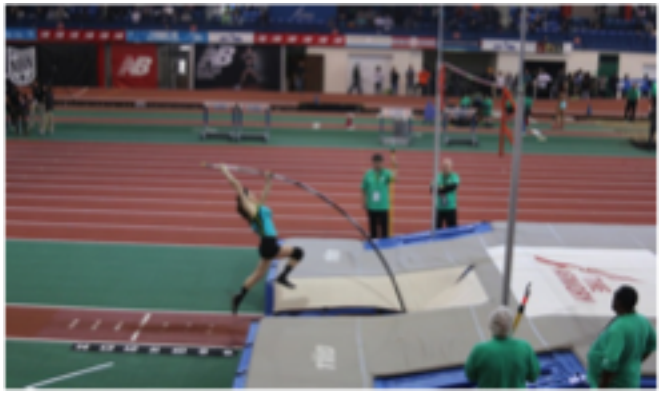
Beer #1: μ_1=19.233 T_1=0.0520 * A_1
Beer #2: μ_2=9.2667 T_2=0.1079 * A_2
Beer #3: μ_3=10.033 T_3=0.0997 * A_3

Recommendation:

- Purchase 3 barrel kegs (31 gallons per keg) for each order.
- An order of this size will last approximately one to two weeks.

Questions?

Smart Pole Vault Runway



Project Goals

- Use IR sensors to track athlete performance metrics
- Measures:
  - Stride Length - distance between 2 steps
  - Speed of athlete
- Future Work:
  - Takeoff point
  - Takeoff force

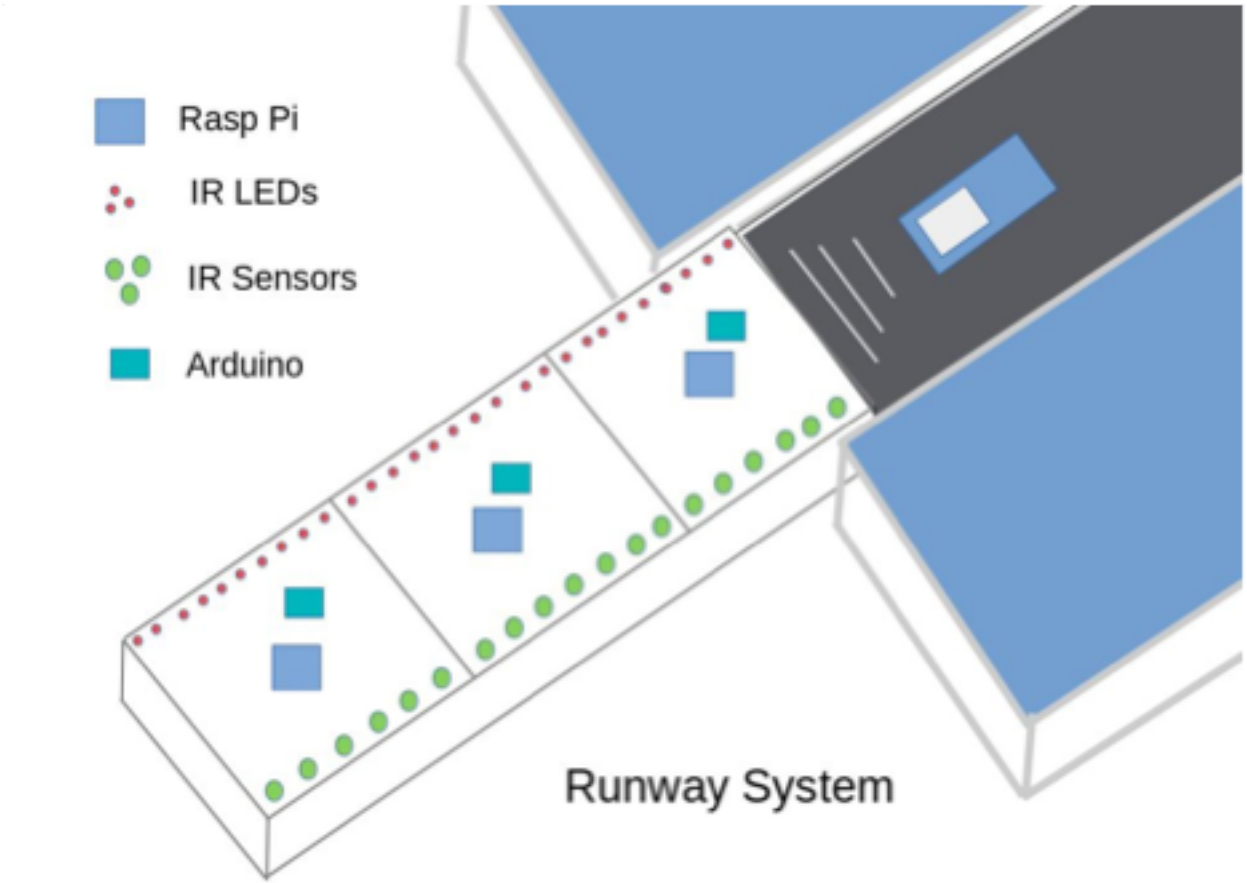
Motivation

- “A little faster at the center” and “start a little back” are not as useful as “start your run up twelve inches further” and “don’t slow down when you reach your second stride.”
- Helps athletes offload the concern about positioning and focus on nailing the jump.
- Improve Athlete performance and safety margins.

How we're doing it

- IR Sensor and LED Pair along opposite sides of the runway.
- Each LED emits a distinct pulse at known times.
- With knowledge of positions of sensors and LEDs, we can calculate the position of any blockage.

Installation Schematic



What we can interpret

- We can identify when a sensor has been blocked. This gives us:
  - Location of interruption along X axis (parallel to runway) = position
  - Time of interruption = instantaneous speed
  - Distance between interruptions = stride length

Information Delivery

- Allows for immediate coaching feedback
- Ability to save individual jumps to Athlete History
- Expansion capability to examine performance trends over time
- Quantifiable performance metrics aid in college recruiting

DriverMonitor: Young Driver Behavior Monitoring System Using SmartWatch and OBD II Sensor

Background

- Crash rates are particularly high early in licensure, due mainly to inexperience and risky driving behavior
- Relative to older drivers, crashes among teenagers are more likely to involve speeding and hitting stationary objects, which could reflect both risk-taking and poor judgment
- Nearly 8,000 people are killed in crashes involving drivers ages 16-20

Objective

- We proposed the DriverMonitor system to keep track of young driver's performance through his/her physical activity and vehicle's sensors.
- The purpose of the project was to examine over time novice teenage driving performance and risk, including kinematic risky driving and speeding.

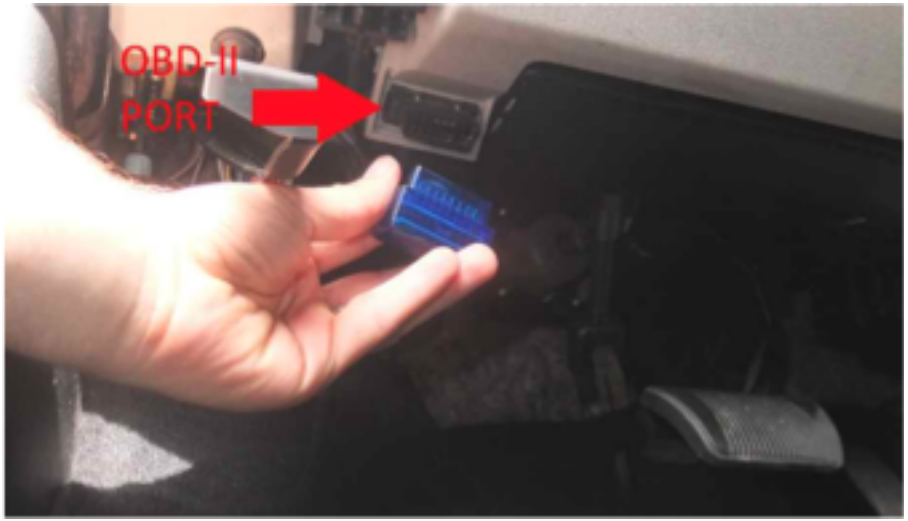
System Design

Speed
Acceleration
RPM
Fuel Level
etc.

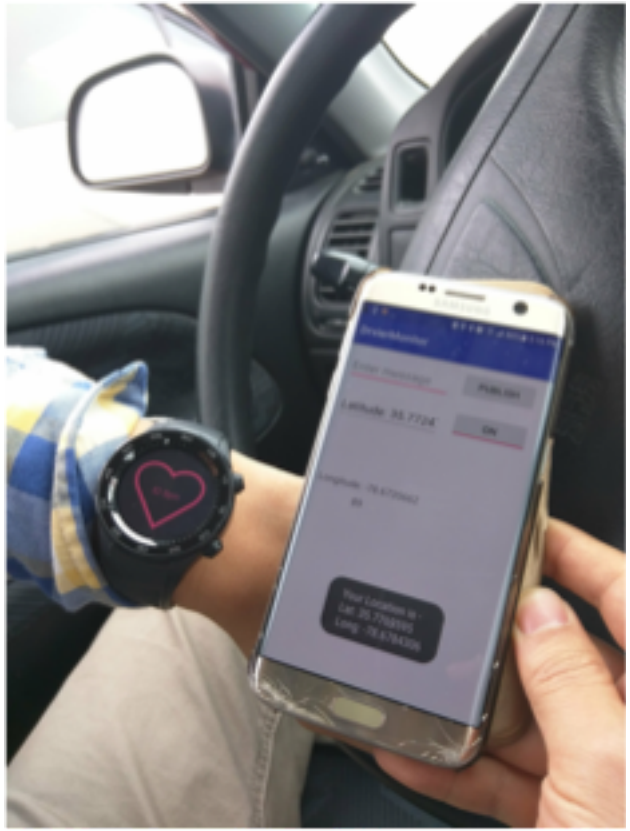
Heart rate

GPS locations

Module 1. Vehicle Sensors



Module 2. Mobile App



Module 3. Web Application

Web link

Thank you!

Smart Coffeemaker

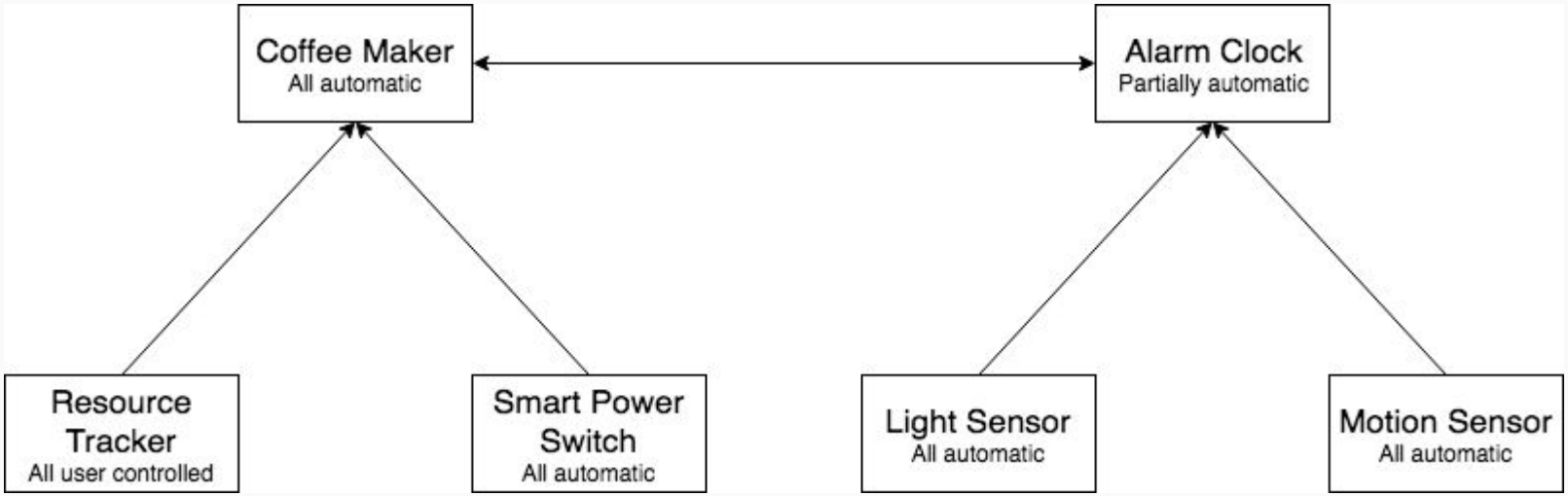
Overview

- Project
- Goals
- Technical Explanation
- Demo Explanation

Project

- Coffeemaker that starts automatically
- Connected to alarm clock
- Can be started manually

Design


Coffee Maker
All automatic
Alarm Clock
Partially automatic
Resource Tracker
All user controlled
Smart Power Switch
All automatic
Light Sensor
All automatic
Motion Sensor
All automatic

Goals

- Starts in a time window when sun rises
- Only starts if it detects user is moving
- Protects against false positives
  - Must detect motion multiple times
  - Light level must be stable to activate alarm
- Wakes the user
  - Can be snoozed
  - If snoozed multiple times, must be manually activated
- User has ability to manually override device

Technical Details

- User manually sets time threshold
- User manually sets resource levels of coffeemaker
- Light, motion, detected automatically
  - Light sensor and motion sensor attached to alarm clock
  - Alarm clock is wireless enabled (in our case Raspberry Pi)
- Spikes in light level are ignored
  - Light must maintain intensity for X minutes to be considered a positive reading
- If resources are not present, they must be added to start

Demo Explanation

- We will shine a light on the light sensor
- After X minutes the alarm clock will activate
- We will ensure the motion detector senses us
- After X minutes the coffee maker will activate
- Demonstration of resource levels


NOX™

What Is NOX™?

- NOX™ is a new, easy light toggling and house monitoring system.
- Current prototype monitors two lights, the sun itself, and a door.
- NOX™ takes sound into account for detailed audio analysis.
- Each monitored light can be turned on or off with the push of button using our intuitive web application.

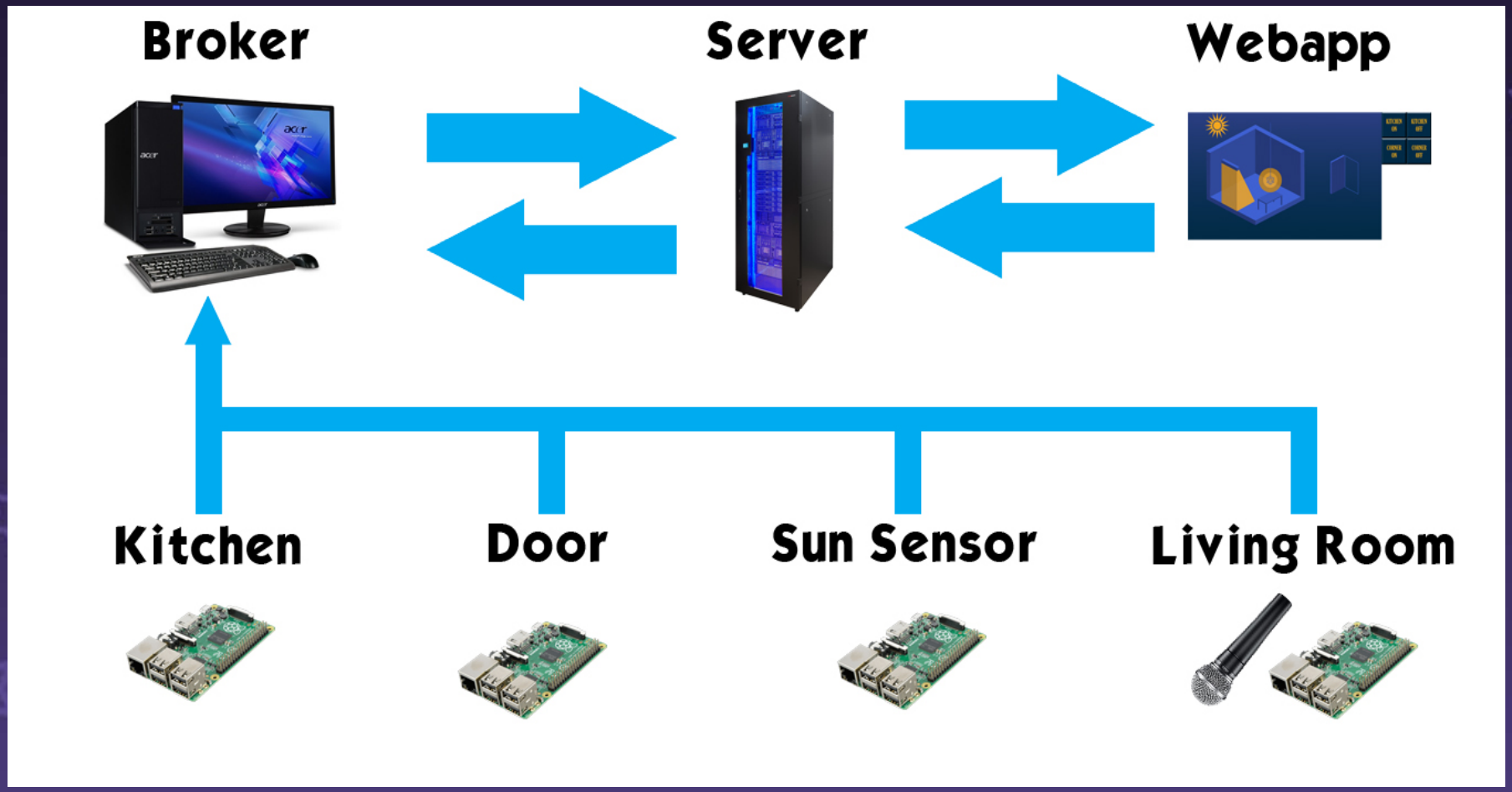
What's The Goal Of NOX™?

- The original goal of NOX™ was to provide a quick way to check the state of common household items.
- Did you forget to close your door?
- Did someone come in your door?
- Did you forget to turn off your lights?
- Which of your lights are used most often?
- Are your lights used actively (someone present)?

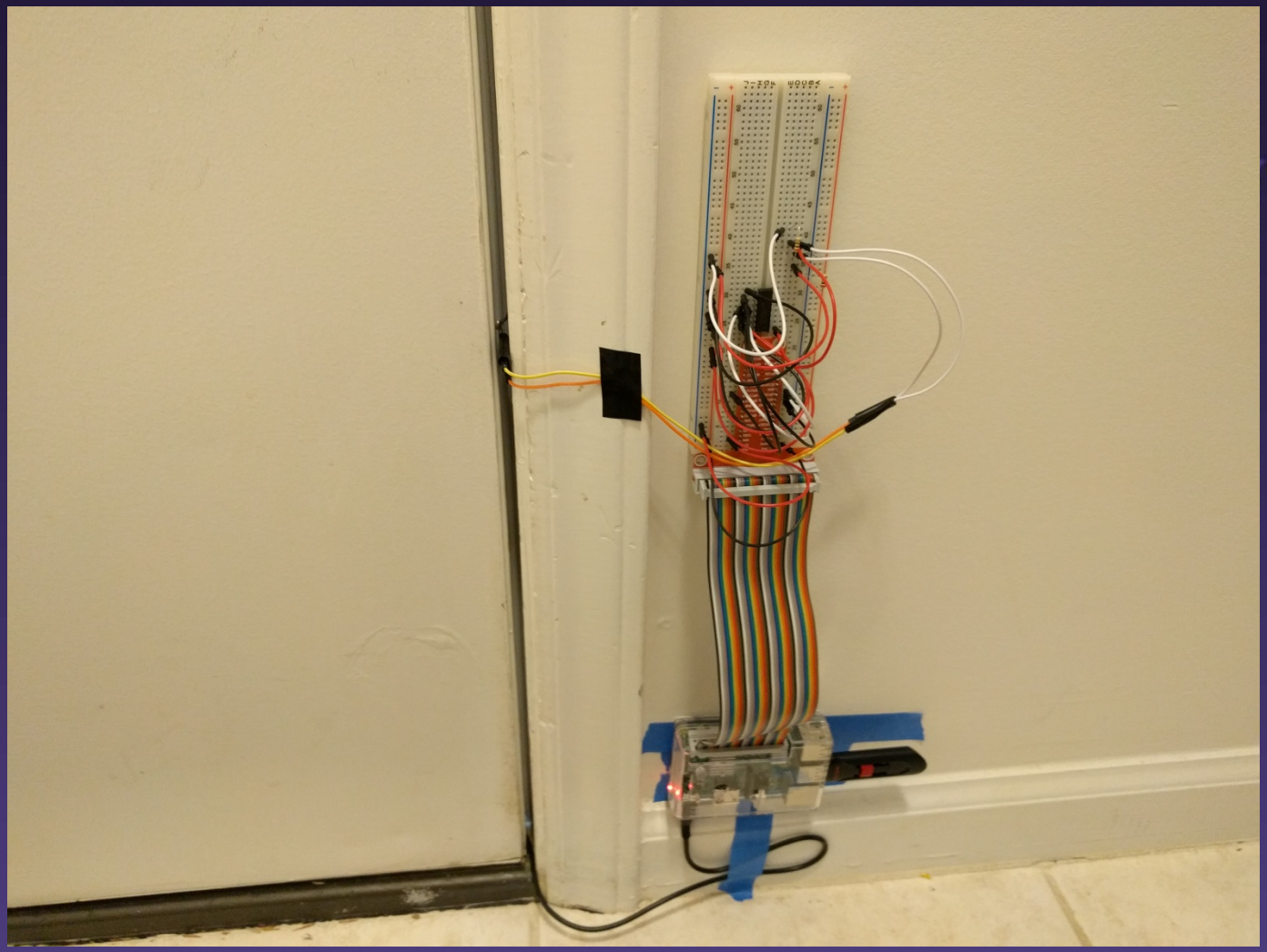
Goal Achievement

- Wired LDR under each light and created a classification model of light.
- Wired a microphone in the main room to create an audio classification model.
- Created an easy-to-use web application that can toggle the lights on and off.
- Used the data produced from the software to create powerful and understandable graphs.

Setup

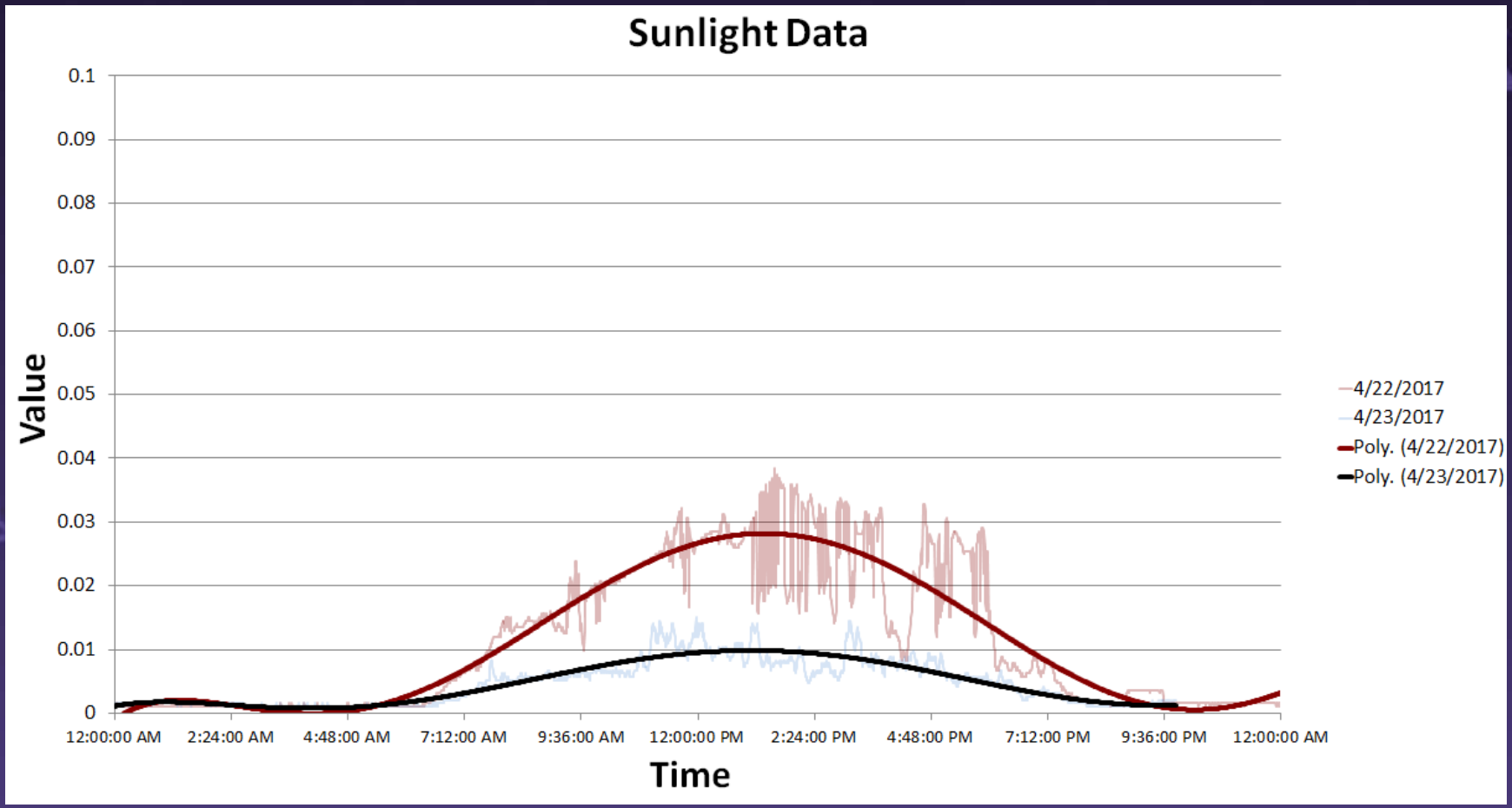


Door Monitor Prototype V1

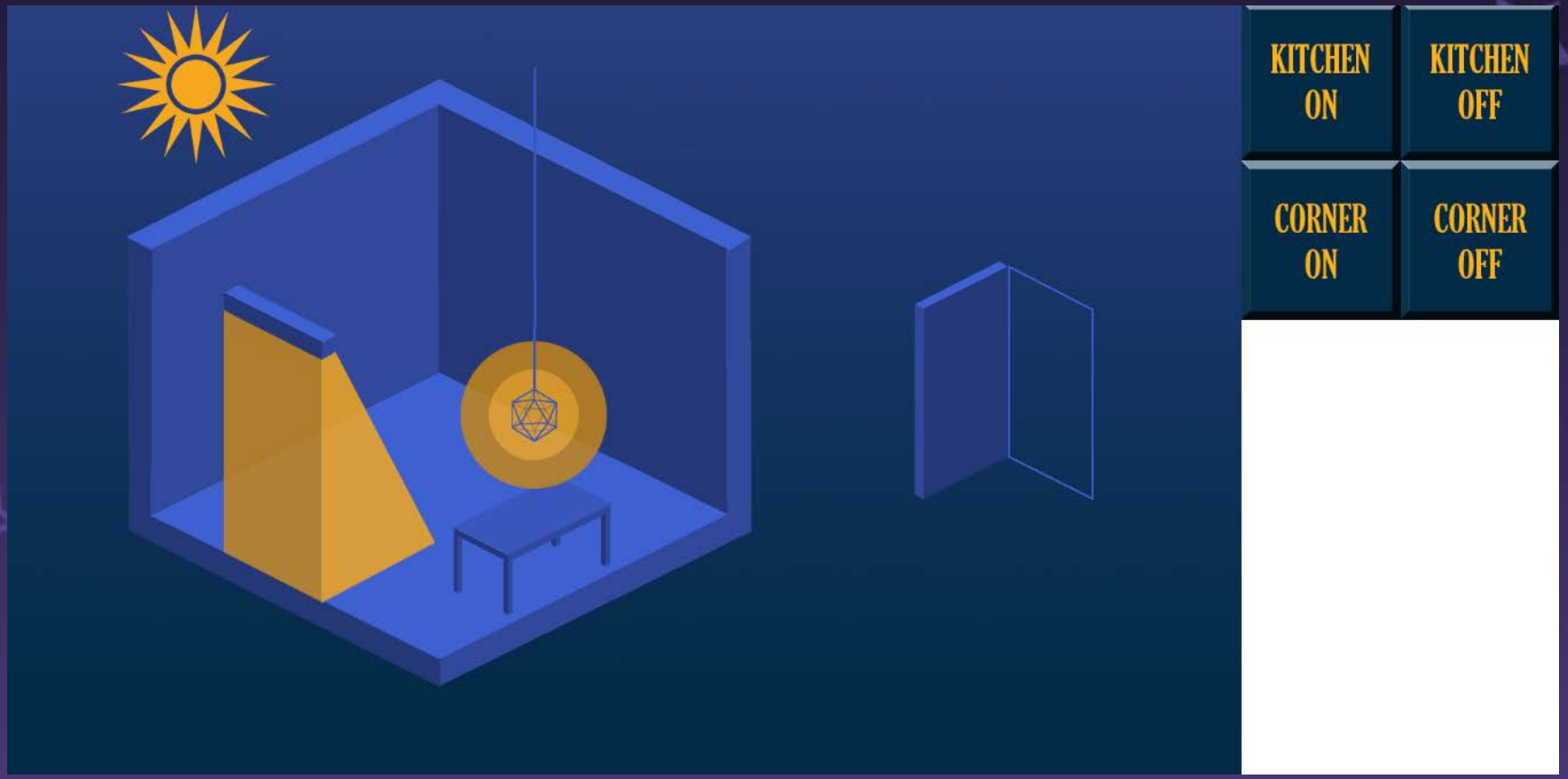


Light Monitor and Advanced Audio System

Sunlight Data Sample



Web Application



Demonstration

- Available soon after May 3rd!
- A quick look at our complex classification system.
- Introduction to the web application and its features.
- Demonstration of light toggling and real-time updates.
- Review of data output for further analysis.

THANK YOU

People Counter

...

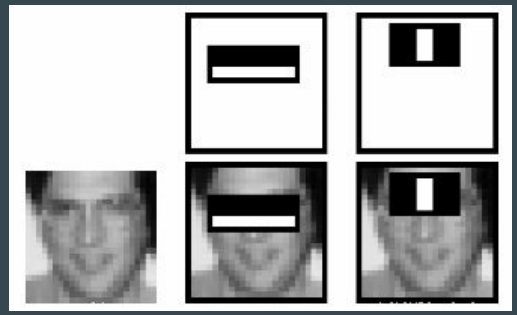
Objectives

We wanted to create a system that would let a user know how many people are in a room and the rate at which the come in, in real-time. More specifically:

1. **Accurate detection of people**
2. **Real-time data**
3. **Data visualization**

How We Met Our Objectives

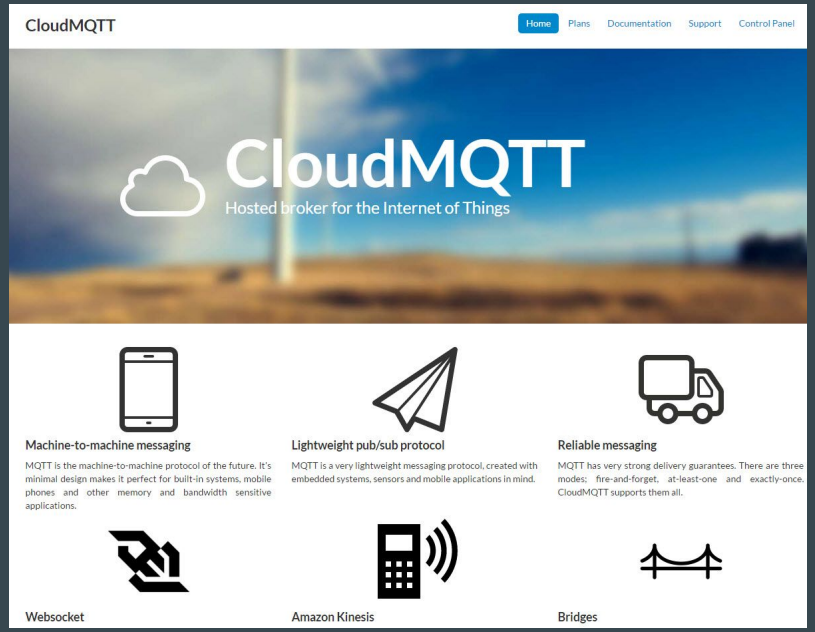
1. **Accurate detection of people**
    a. Pi Camera
        i. Easy installation
        ii. Video/images provide valuable information about the room
    b. OpenCV - Haar Cascades detector to detect faces
        i. Detects real people, rather than moving objects
        ii. High accuracy
        iii. Works seamlessly with Pi Camera
    c. Deduplication on web service pre-trained neural network as feature descriptor



How We Met Our Objectives (cont.)

2. Real-time Data

a. CloudMQTT
   i. Secure and convenient MQTT Broker for pub/sub
b. Clever programming
   i. Processes, logs, calculates and publishes in real-time
   ii. Removes duplicates



How We Met Our Objectives (cont.)

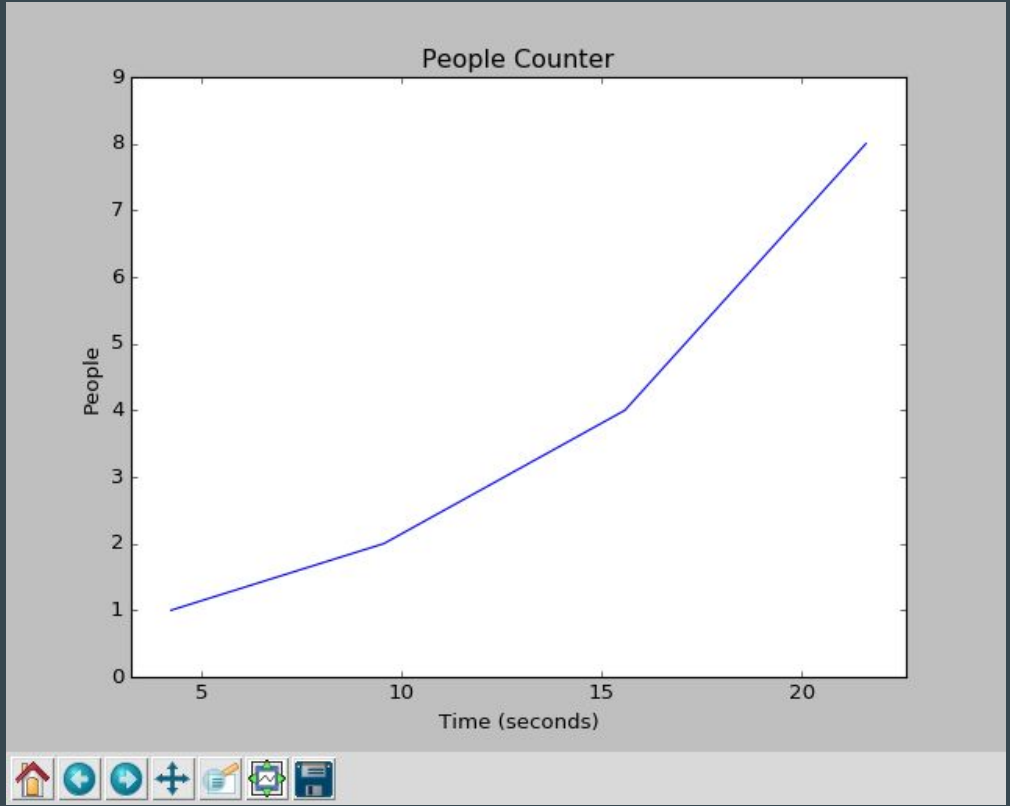
3. Data Visualization

a. Matplotlib

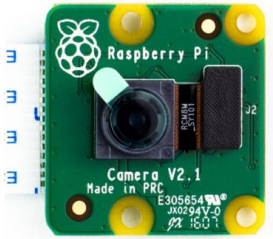
b. SQLite 3

c. Dynamic Graph Updates

d. The rate of people entering the slope



Bringing Everything Together...



Camera Counter

Face upload

Image Processing Service

Subscribe on camera1/count

CloudMQTT - MQTT Broker

Subscribe on camera1/count

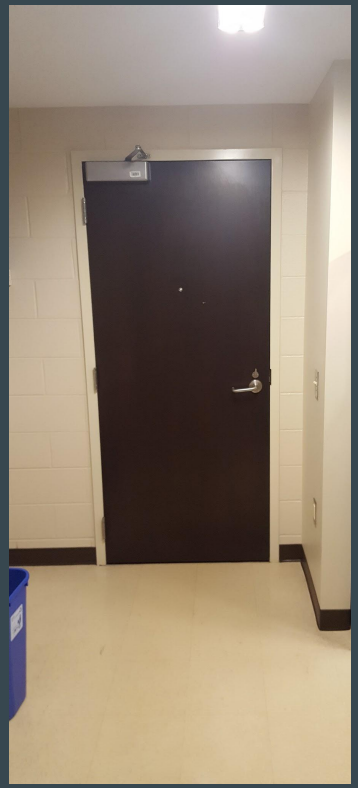
Logger

Visualization App

Persist entry logs

SQLite DB

Our Project



1

2

Peopl

e 2

1

Publish(1) cameral/count

Publish(2) cameral/count

Time (s)

Demo Steps

Step 1: Secure the Pi to a wall approximately 6-12 feet away from the entryway to the room. Start the person counter program and successfully connect the Pi to the MQTT broker. **(15% of total grade)**

Step 2: Three group members will exit the room (or FOV of camera). The first person will walk in and the counter displays a total count of 1 individual. **(8% of total grade)**

Step 3: The second person will walk in and the counter displays a total count of 2 individuals. **(8% of total grade)**

Step 4: The third person will walk in and the counter displays a total count of 3 individuals. **(8% of total grade)**

Step 5: Ensure that the data visualization of the people count changes in near real time. Ensure that each individual is logged with a timestamp and a picture is stored locally in the deduplication service. **(15% of total grade)**

Step 6: All 3 members who have previously entered will re-enter the frame of the camera. The camera will not publish any new people and the count will remain 3. **(25% of total grade)**

Step 7: The group member that has not previously been logged, will enter the frame and the count will be displayed as 4. **(21% of total grade)**

Questions?

Bus Capacity Counter

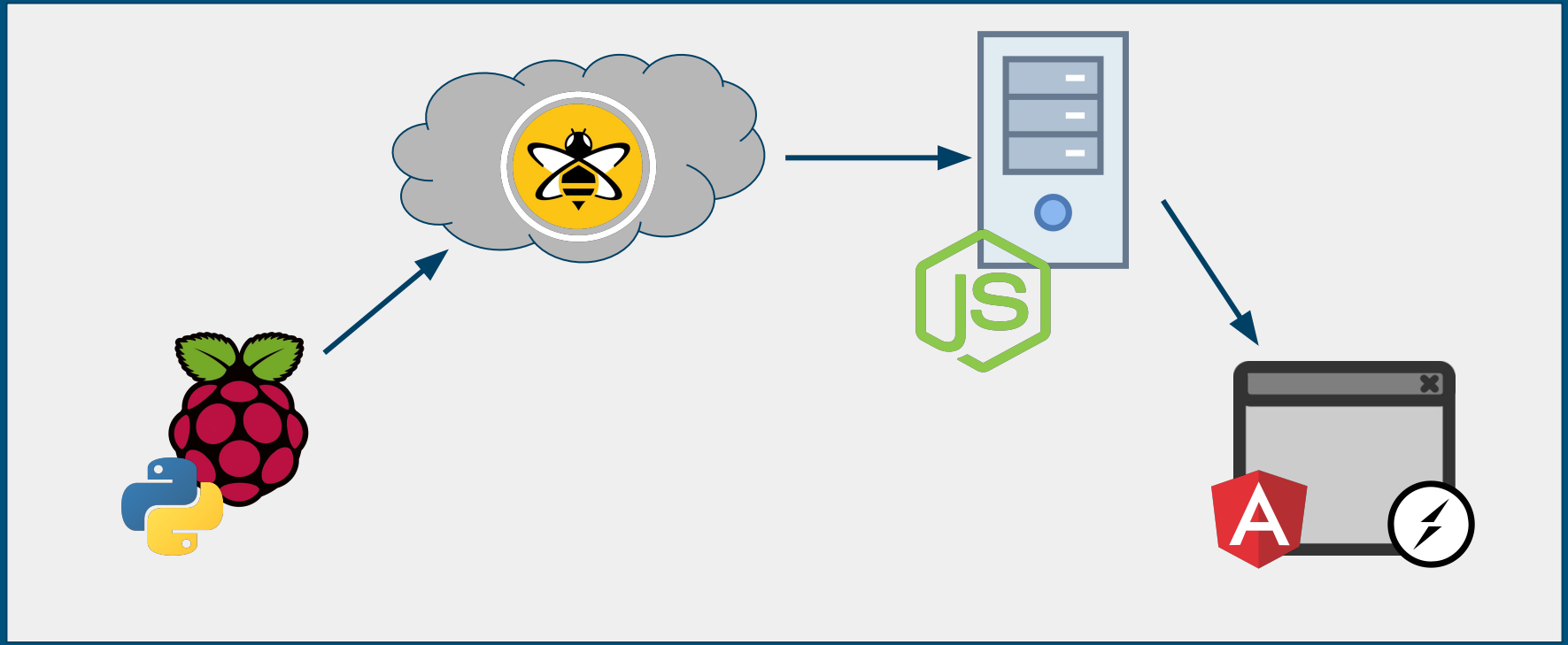
Goals:

- Accurate percentage of the capacity of a bus
- Update in real-time as students get on and off the bus
- Simple and accessible UI to view the information

Achieving Those Goals:

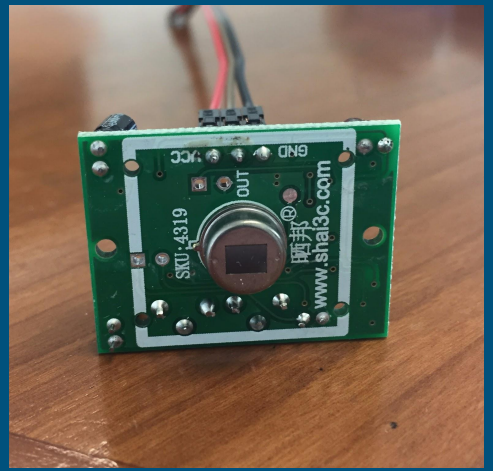
- Use four motion sensors at the door of the “bus”
- Algorithm to determine which direction a student is moving in
- HiveMQ as an online broker to post our capacity to

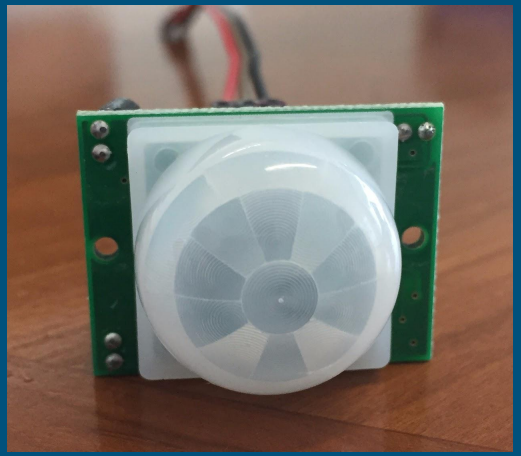
Architecture and Design:

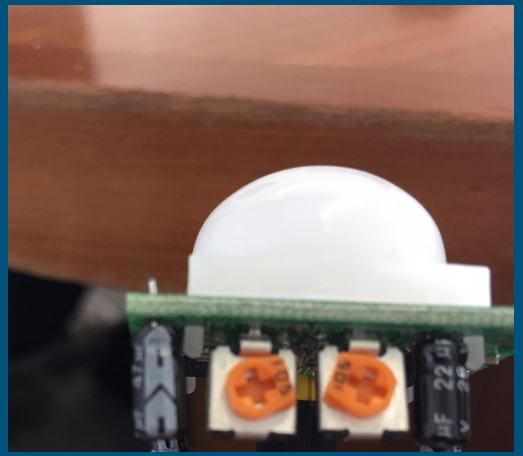


Sensors

- For monitoring individuals walking on and off the bus we used 4 HC-SR501 PIR motion sensors:

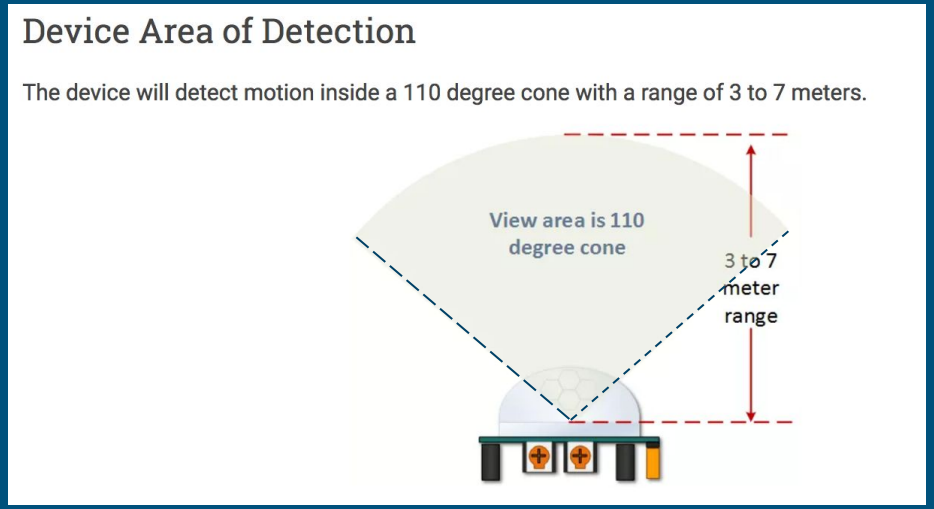






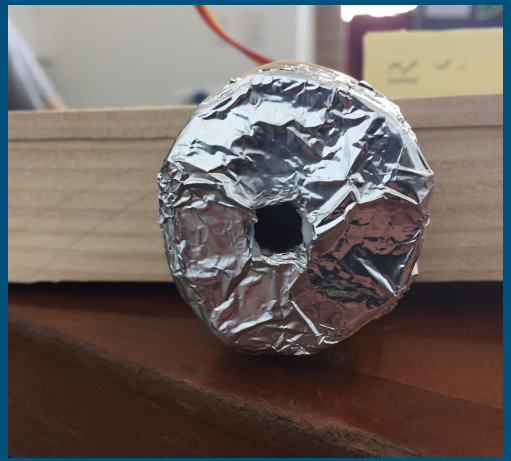
Issues with the Sensors

- Our major issue with the sensors was their viewing angle
  - Caused us issues with our algorithm since the 110 degree viewing angle caused multiple sensors to detect at once



Fixing Sensor Issues

- To counteract and adjust for the wide detection angle, we modified our sensors with paper-towel rolls, white paper, and aluminum foil



Circuit Schematic

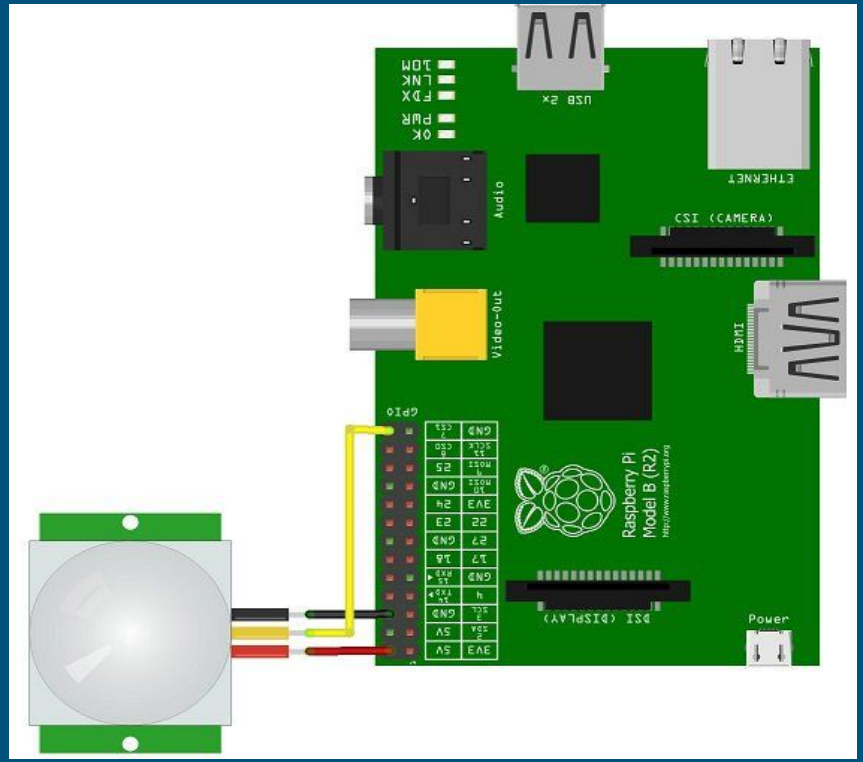


Image source: https://electrosome.com/pir-motion-sensor-hc-sr501-raspberry-pi/

Final Prototype

- Represents the railing on a bus where you enter and exit

Demo

- Get on the bus
- Get off the bus
- Get half on/off and turn around

The End

Any questions?

CSC453 - Group 9
Automatic DJ

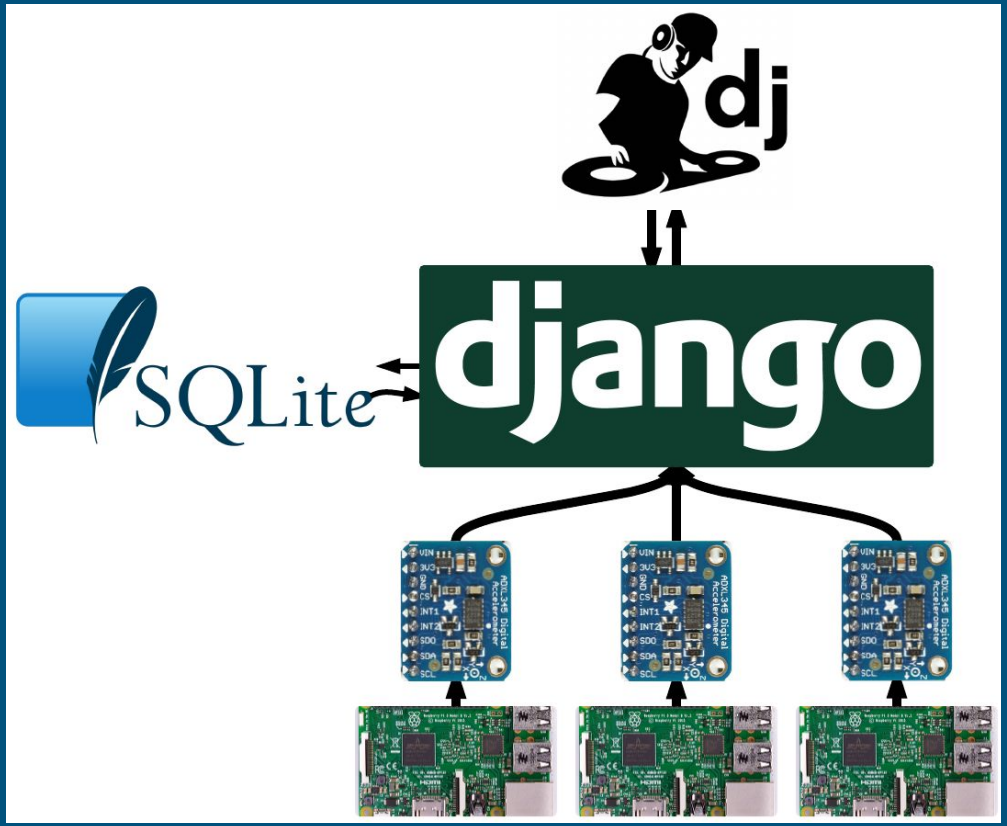
Project Introduction

- People in social settings don’t want to worry about fitting music to match the mood and movement of a room
- People at a party would have accelerometers attached to them
- Movement readings would be sent to a server
- Server takes in accelerometer readings and calculates danceability of the currently playing song
- Danceability and song metrics are used by server to determine the next best song to play
- A model is developed for predicting danceabilities and songs

Project Goals

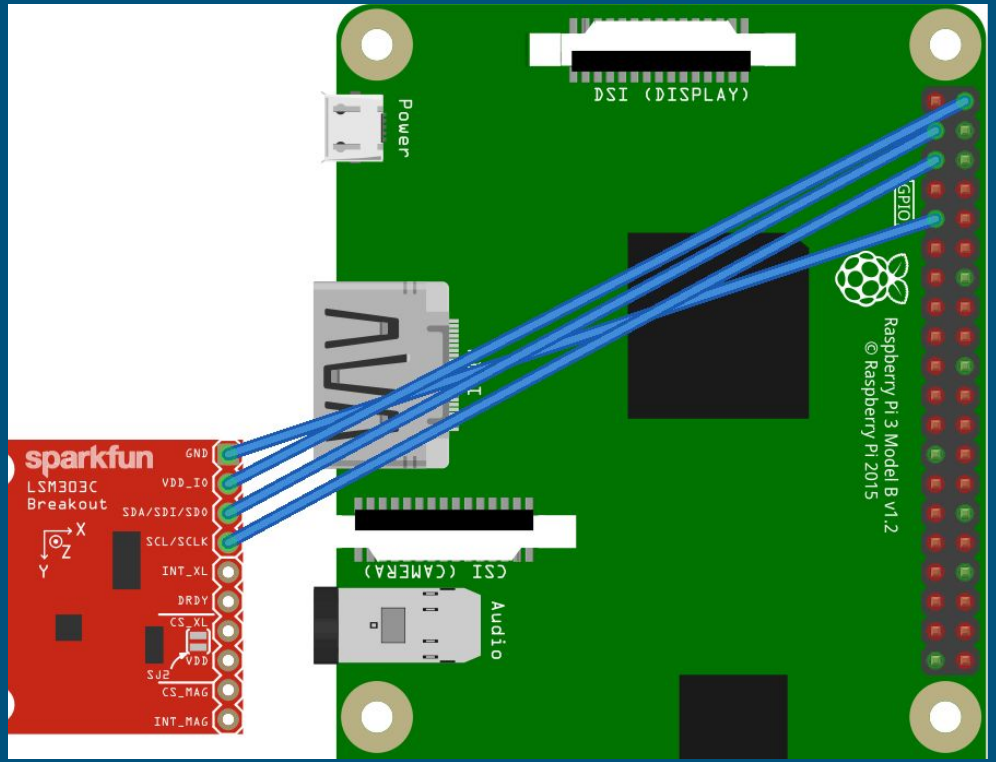
- Read accelerometer data into individual Raspberry Pi clients during song playback
- Post readings to server periodically
- Interpret readings to calculate a danceability score for the song
- Through regression across recorded data, use song feature data from Spotify to predict danceability scores for similar songs in the playlist
- Automatically decide order of playback based on continuous data input

System Architecture



Project Implementation

- Attach accelerometers to each pi using four pins
  - Power
  - Ground
  - SDA
  - SDL
- Acceleration in X, Y, and Z directions read 40 times a second
- Use changes from positive to negative acceleration to tally a “shake” count
- Danceability = shake count / song length


DSI (DISPLAY)
Power
GPIO
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
CSI (CAMERA)
Audio
sparkfun
LSM303C
Breakout
GND
VDD_IO
SDA/SDI/SDO
SCL/SCLK
INT_XL
DRDY
CS_XL
VDD
SJ2
CS_MAG
INT_MAG

Project Implementation (cont.)

- Server receives playlist and gathers eight song metrics from Spotify for each song
- Server tells DJ what song to play first (picked at random)
- At the end of each song:
  - Calculate danceability for the song that just ended
  - Generate a decision tree regression
  - Use regression to update our model for predicting danceability
  - Return the song to be played next with the highest danceability
- Works continuously, the more data the better we can predict

Our Demo

1. Run server and DJ programs on laptop
2. The server will tell the DJ what song to start playing
3. Power on each Pi and start accelerometer client
4. “Dance” by moving the Pis around and verify the server is receiving data
5. After the song finishes, a danceability score should be calculated and displayed on the server’s dashboard
6. The server will then use data pulled from Spotify to predict danceability scores for songs with similar values from Spotify (visible on dashboard)
7. Server selects next song based on predicted danceability
8. Rinse and repeat (steps 4-7), varying “dance” intensity

Design Decisions

- Decided to use django REST framework to store data and offload calculation workload from lightweight clients
- Call and response client/server interaction => no need for MQTT
- Calculating danceability
- Choosing a regression model
  - Collected a set of training data
  - Used 6 fold validation on Weka to find a regression with lower error values
  - Proved difficult as it’s hard to get true training data without hosting a party