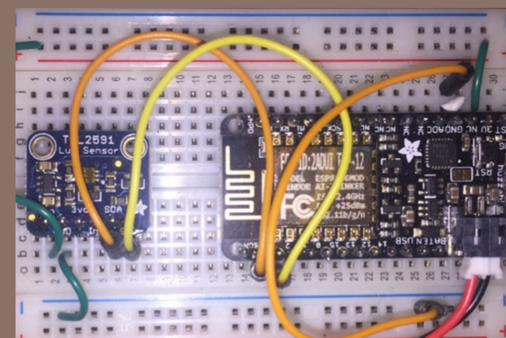
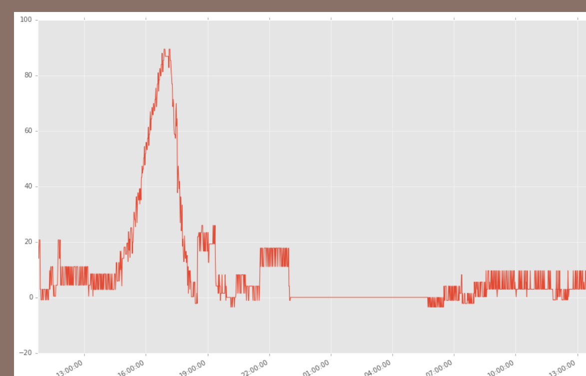
PYTHON AND IOT:

From Chips and Bits to Data Science



Jeff Fischer
Data-Ken Research
jeff@data-ken.org
https://data-ken.org
Sunnyvale, California, USA



BayPiggies October 2016

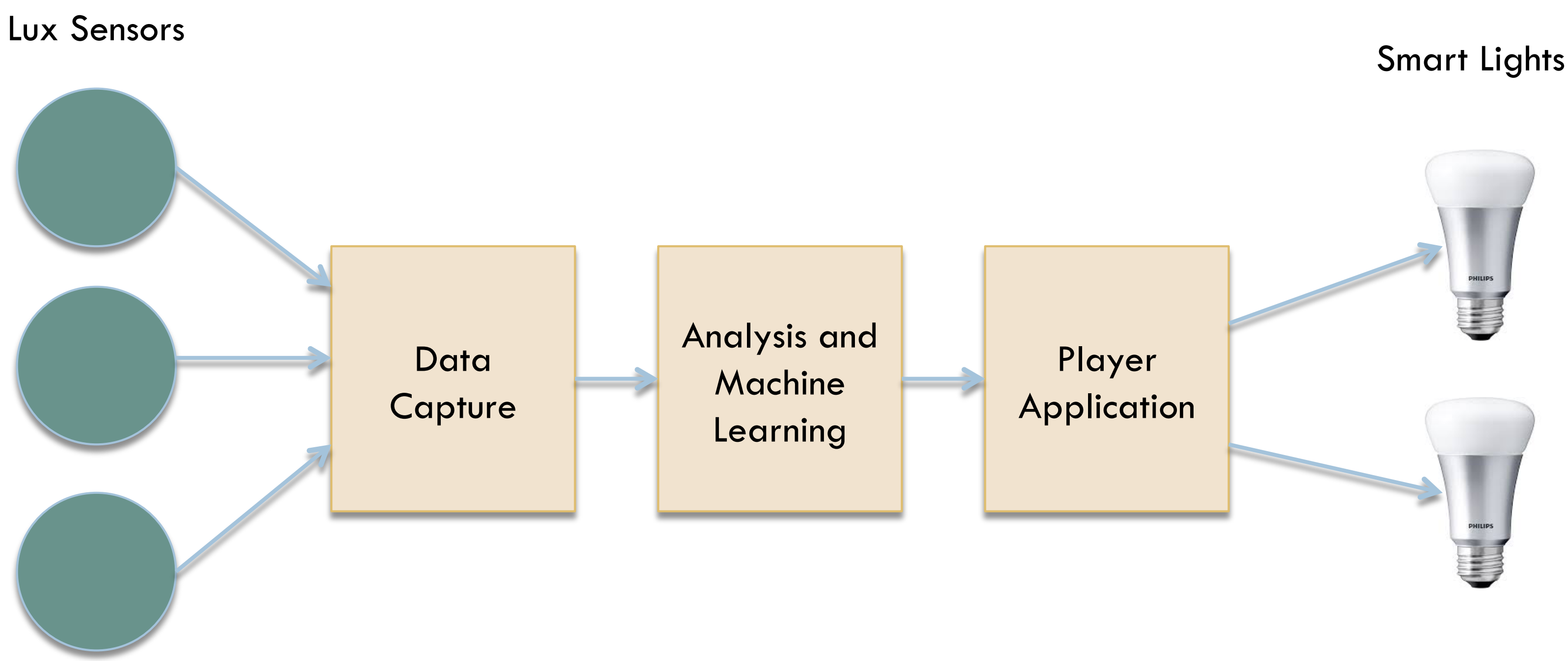
Agenda

- Project overview
- Hardware
- Data capture
- Data analysis
- Player
- Parting thoughts

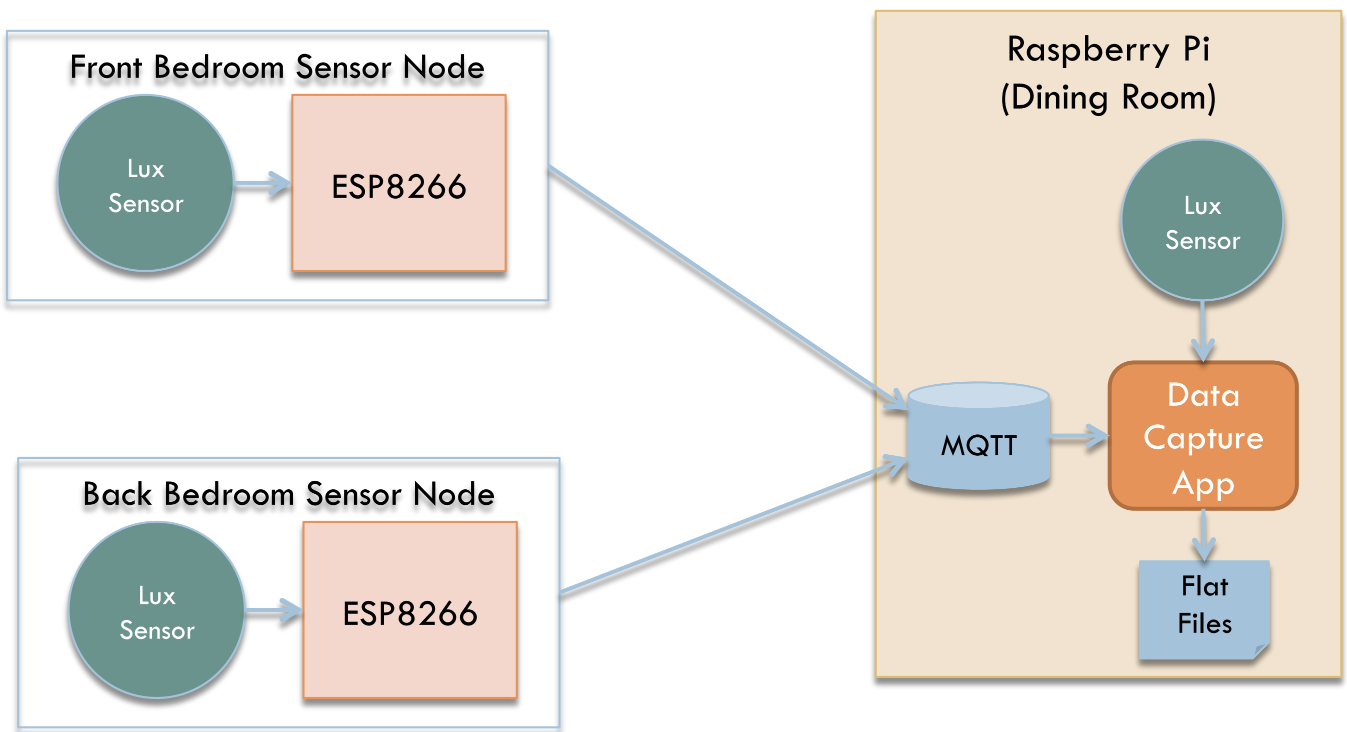
Project Motivation

- If out of town for the weekend, don't want to leave the house dark
- Timers are flakey and predictable
- Would like a self-contained solution
  - Avoid security issues with cloud solutions
- "Wouldn't be cool to use machine learning?"

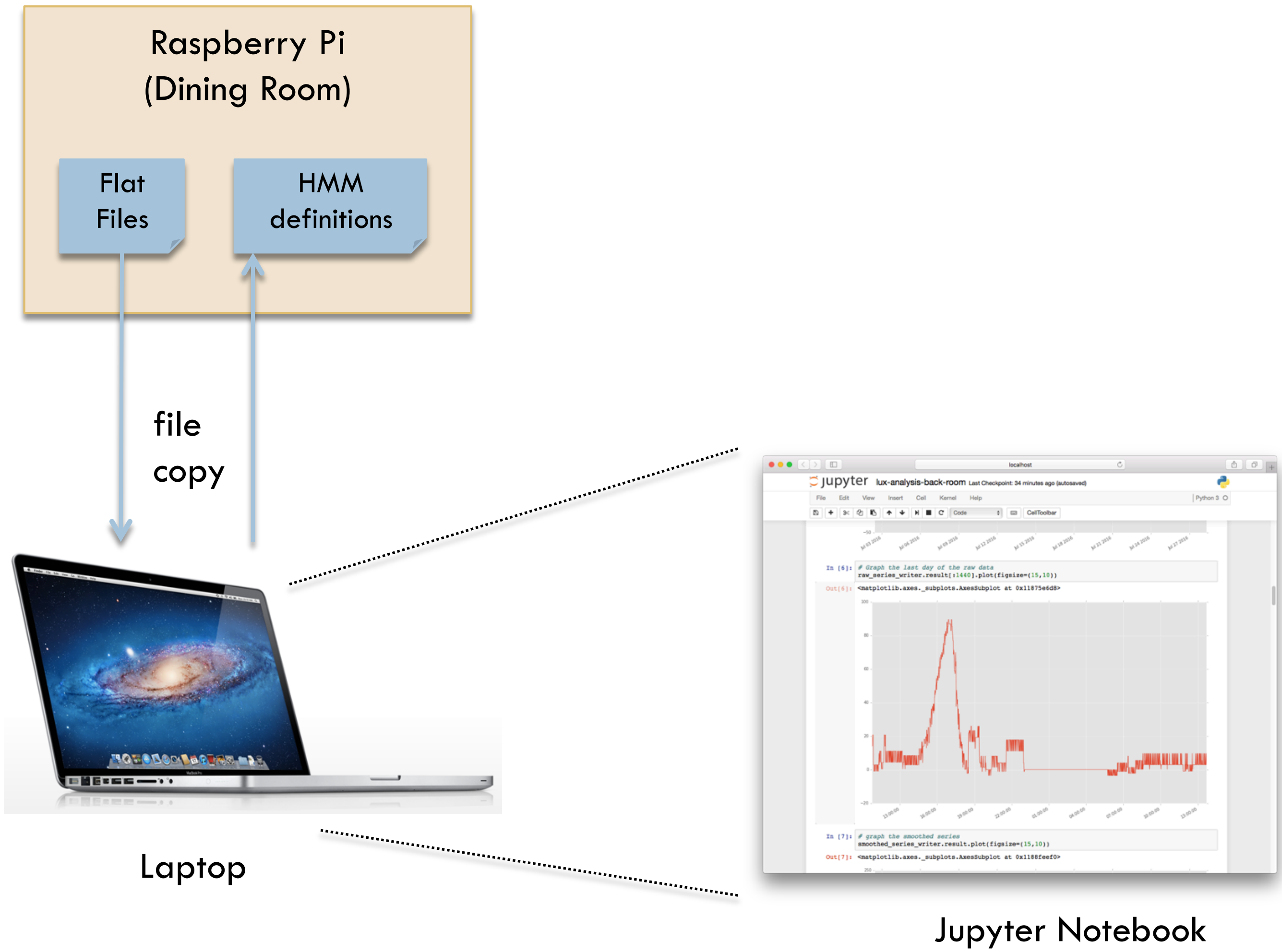
Lighting Replay Application



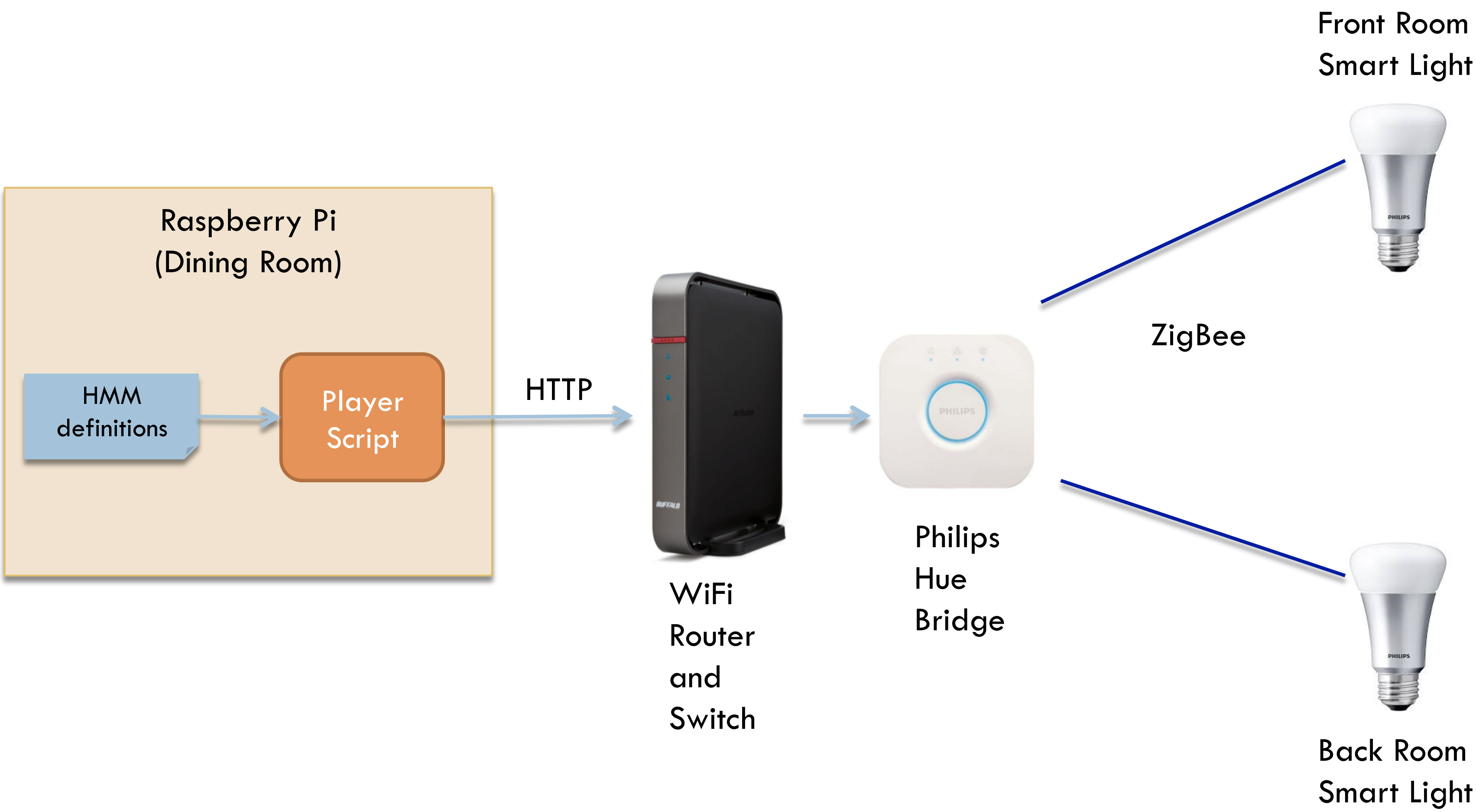
Lighting Replay Application: Capture



Lighting Replay Application: Analysis



Lighting Replay Application: Replay

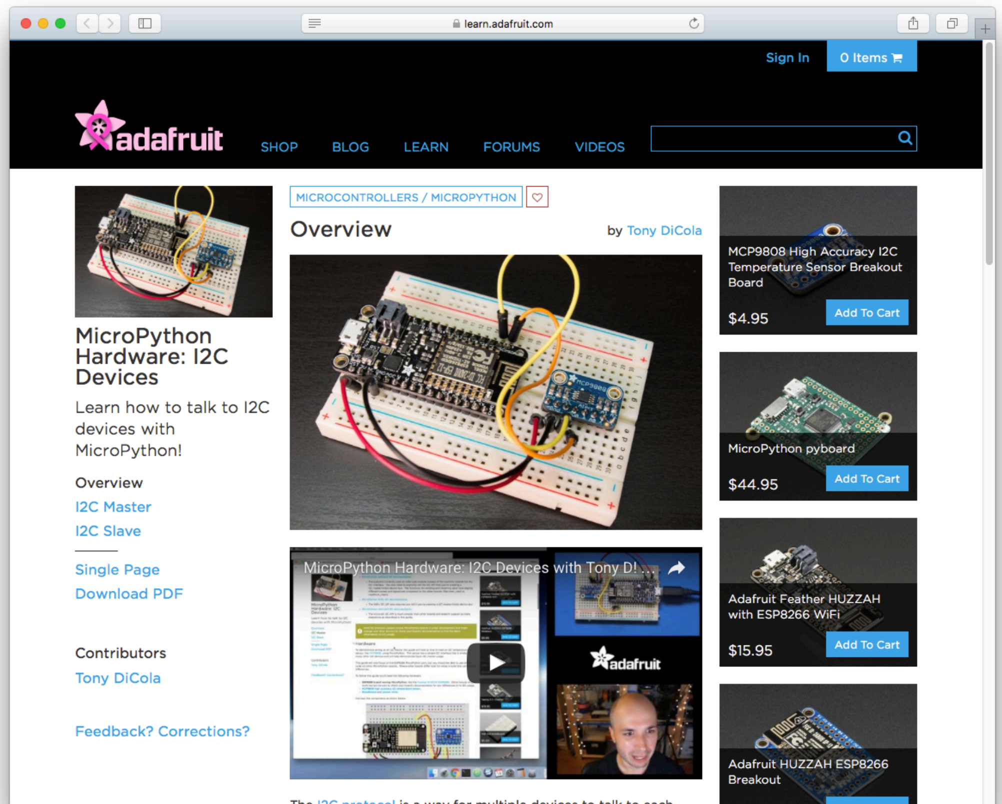


8

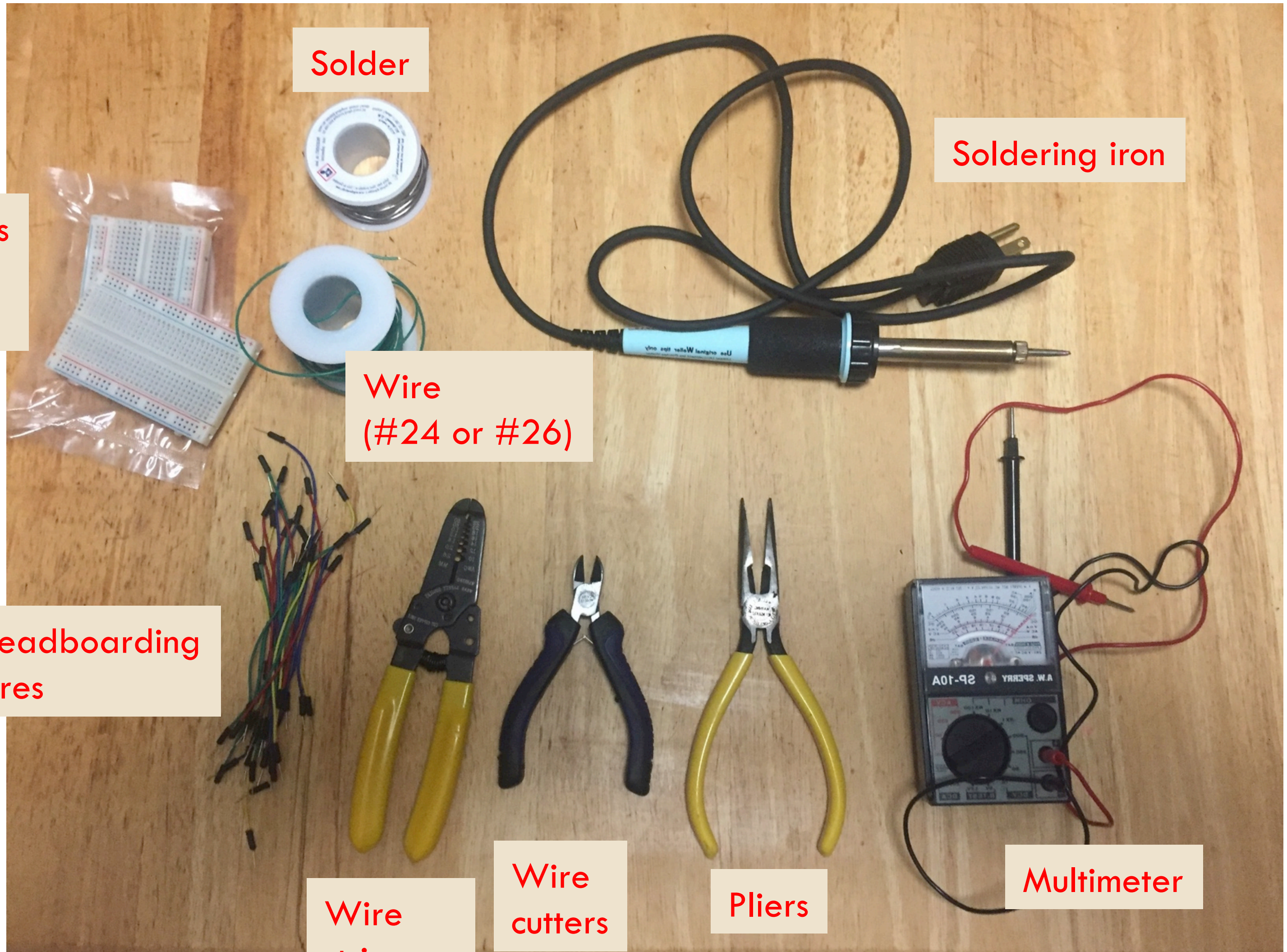
Hardware

Recommended Hardware Supplier: Adafruit

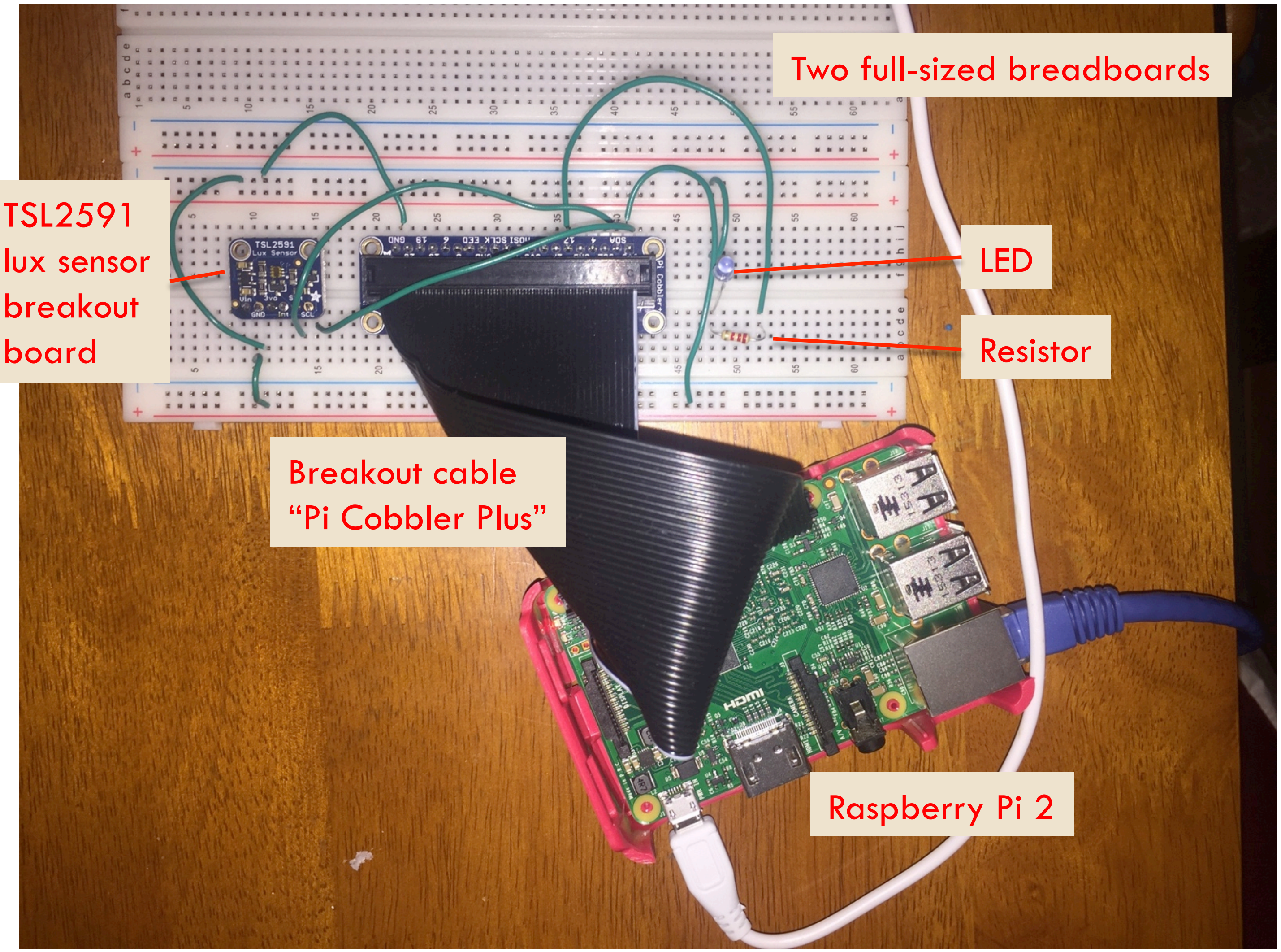
- Focused on the hobbyist
- Plenty of documentation and examples
- Breakout boards make it easy to work with peripheral ICs



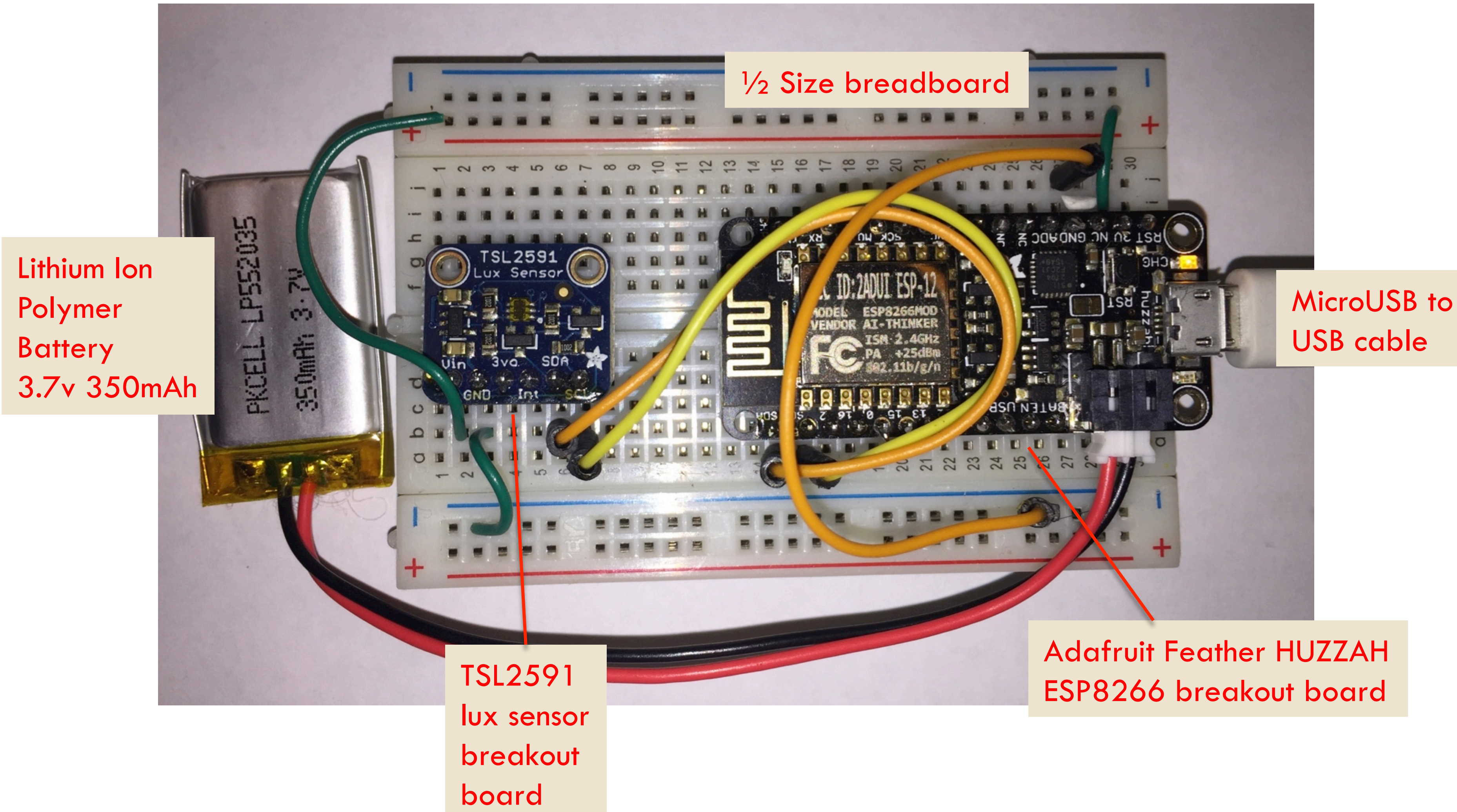
Recommended Tools



Raspberry Pi

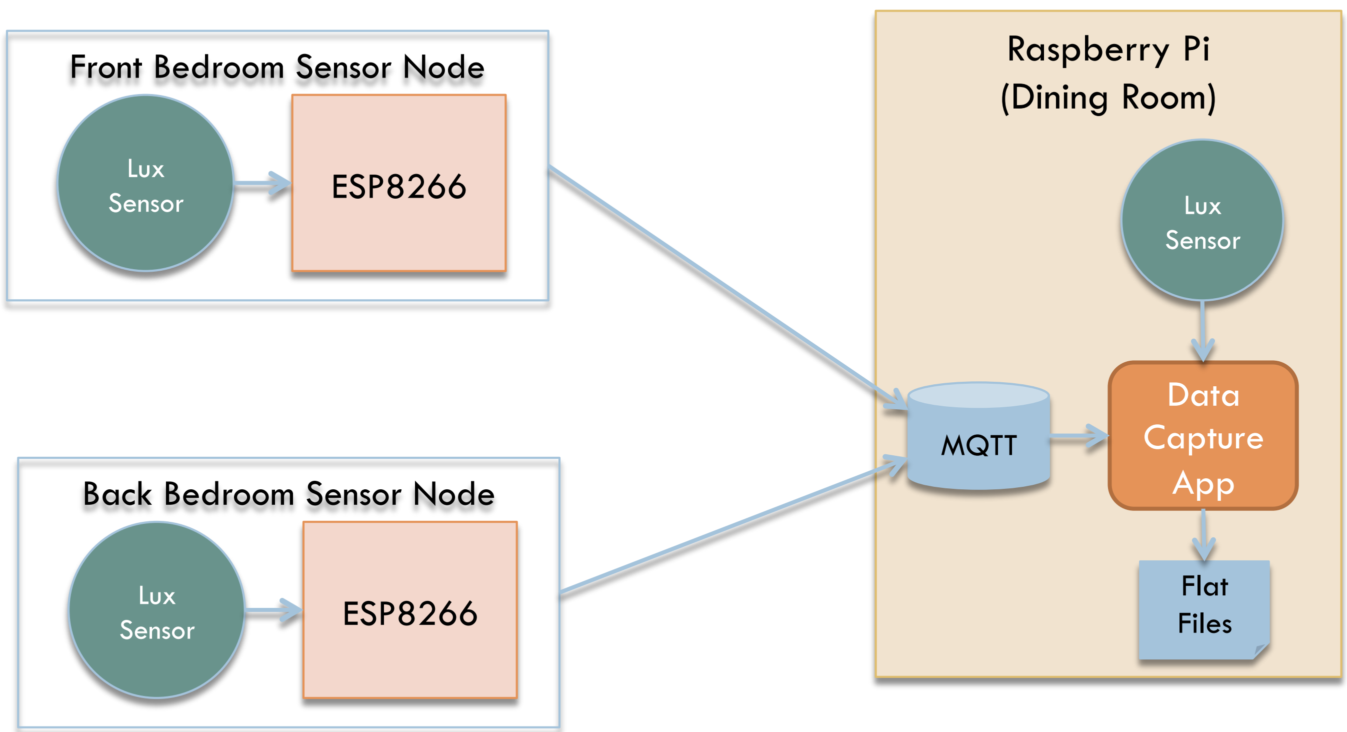


ESP8266


½ Size breadboard
Lithium Ion Polymer Battery 3.7v 350mAh
MicroUSB to USB cable
TSL2591 lux sensor breakout board
Adafruit Feather HUZZAH ESP8266 breakout board
TSL2591
Lux Sensor
ESP-12
MODEL ESP8266MOD
VENDOR AI-THINKER

Data Capture

Lighting Replay Application: Capture



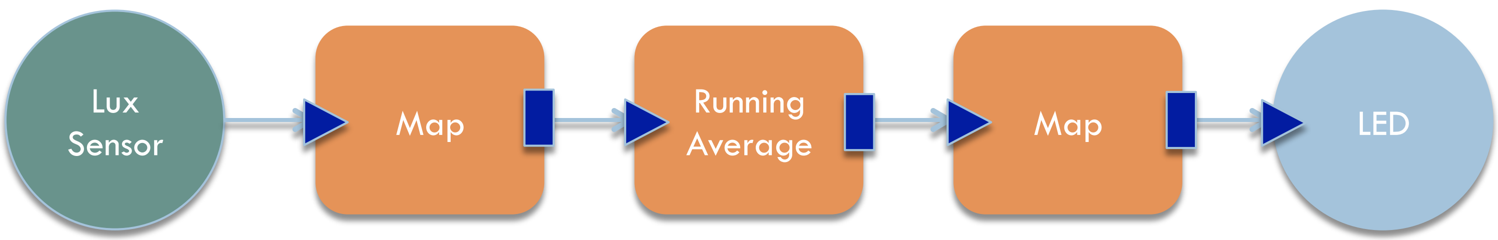
AntEvents

- Python3 library for processing IoT event streams
  - Built on Python 3.4's asyncio module
  - Port to Micropython, which runs on the ESP8266
- Key library features:
  - Push-style streams of events
  - Assemble "elements" into a DAG
    - Fine-grained pub/sub model: an element is a publisher, a subscriber, or both
    - Special support for pipelines of stateful filters
    - Elements can be proxies for external systems
  - Event-driven scheduling, with separate threads for blocking elements
- https://github.com/mpi-sws-rse/antevents-python

Simple AntEvents Example

- Sample a light sensor every two seconds and turn on an LED if the average of the last 5 samples exceeds a threshold

```
lux = LuxSensor()
Lux.map(lambda e: e.val).running_avg(5) \
   .map(lambda v: v > threshold).GpioPinOut()
scheduler.schedule_recurring(lux, 2.0)
```



ESP8266 Code


```
from antevents import Scheduler
from tsl2591 import Tsl2591
from mqtt_writer import MQTTWriter
from wifi import wifi_connect
import os

# Params to set
WIFI_SID= …
WIFI_PW= …
SENSOR_ID="front-room"
BROKER='192.168.11.153'

wifi_connect(WIFI_SID, WIFI_PW)
sensor = Tsl2591()
writer = MQTTWriter(SENSOR_ID, BROKER, 1883,
                    'remote-sensors')
sched = Scheduler()
sched.schedule_sensor(sensor, SENSOR_ID, 60, writer)
sched.run_forever()
```

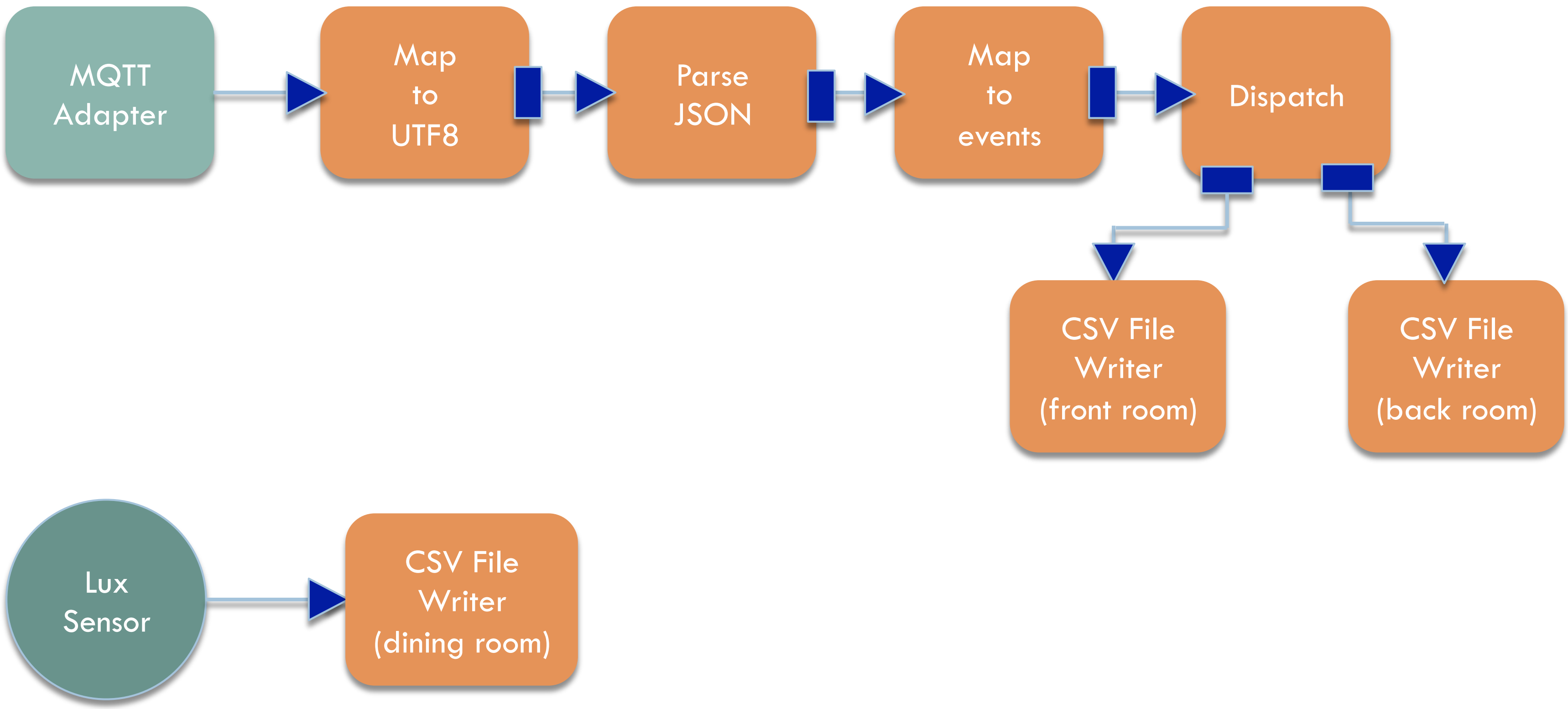
https://github.com/jfischer/micropython-tsl2591

Sample at 60 second intervals

The MQTT writer *subscribes* to events from
The lux sensor.

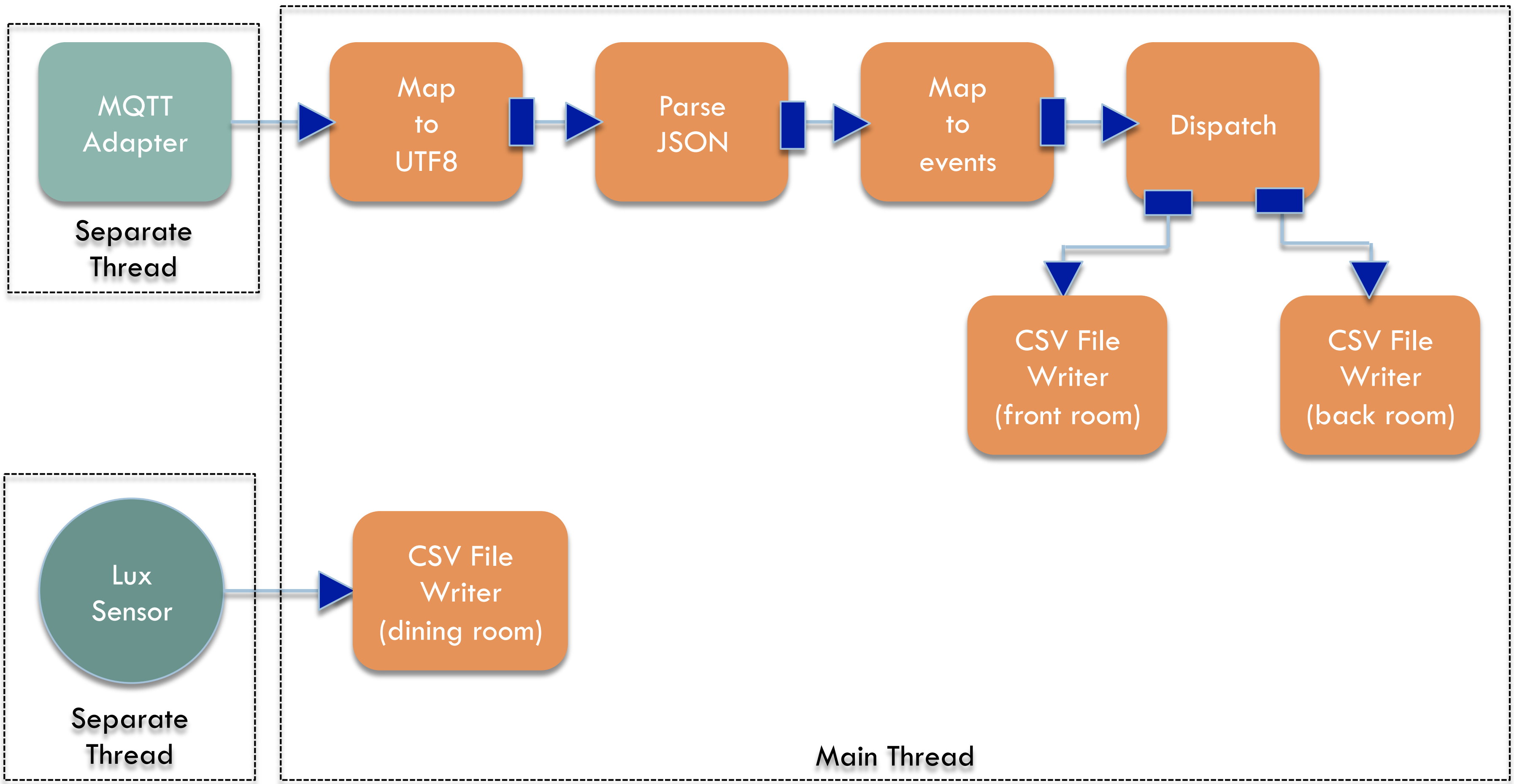
See https://github.com/mpi-sws-rse/antevents-examples/blob/master/lighting_replay_app/capture/esp8266_main.py

Raspberry Pi Code



https://github.com/mpi-sws-rse/antevents-examples/blob/master/lighting_replay_app/capture/sensor_capture.py

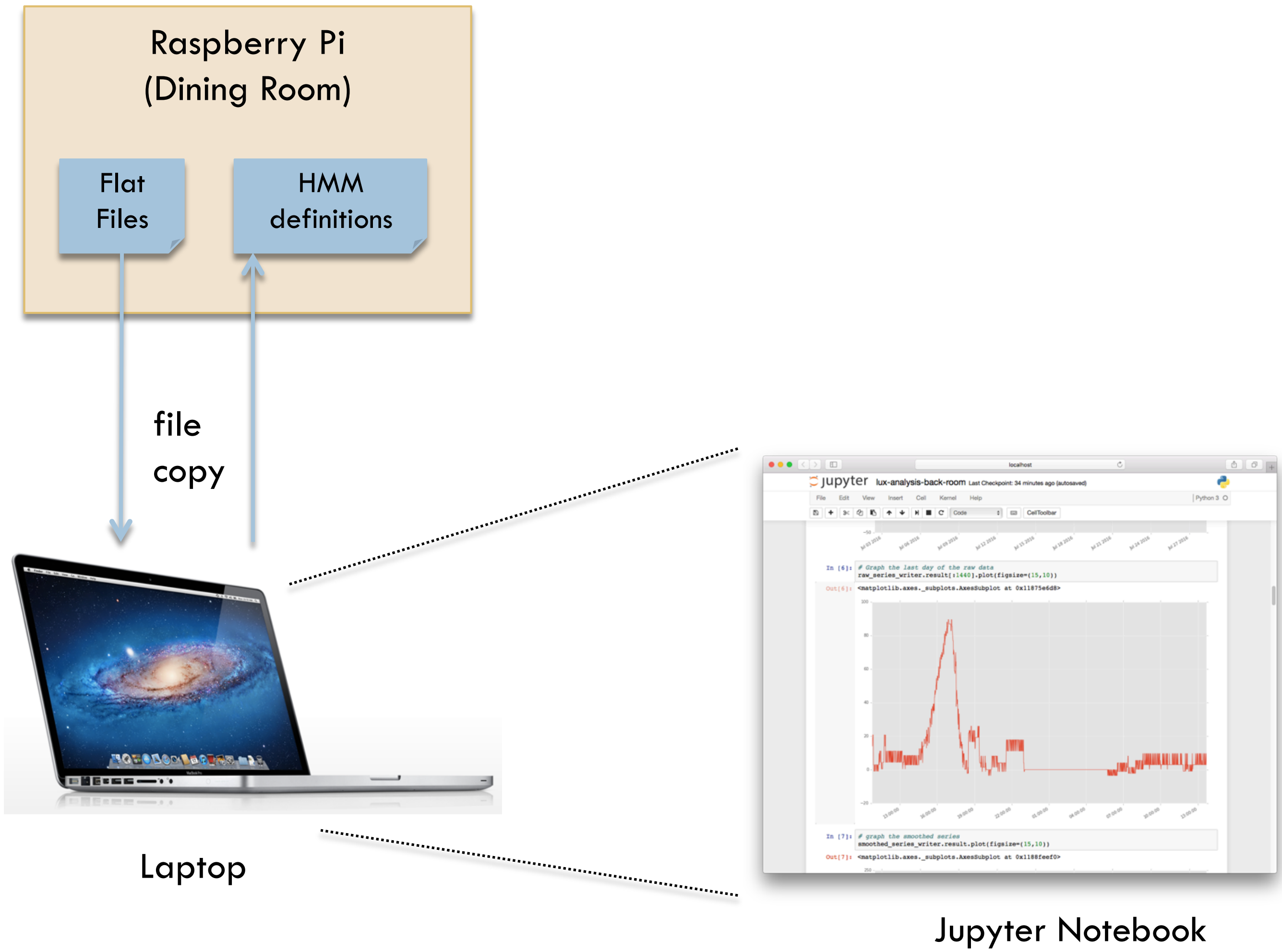
Raspberry Pi Code: Threading Model



https://github.com/mpi-sws-rse/antevents-examples/blob/master/lighting_replay_app/capture/sensor_capture.py

Data Analysis

Lighting Replay Application: Analysis

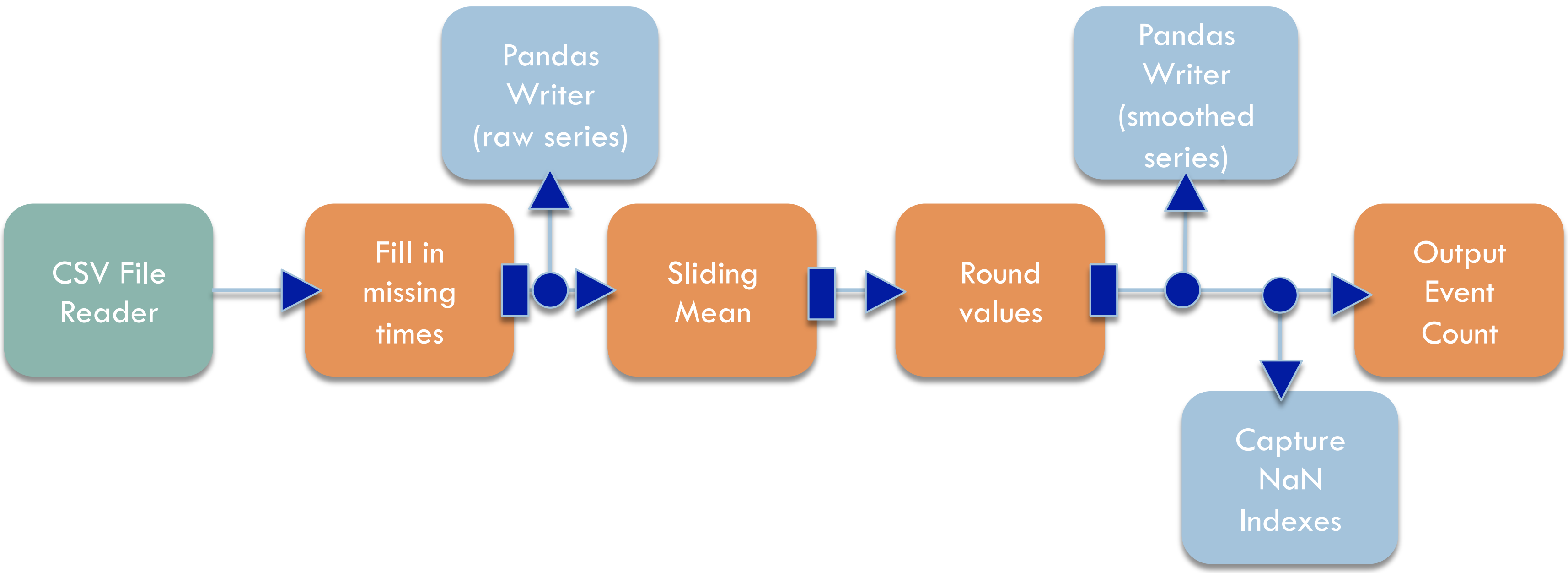


Steps in Data Analysis

1. Read and preprocess data files
2. Convert to discrete levels using K-means clustering
3. Map to on-off values
4. Train Hidden Markov Models (HMMs) on data
5. Validate predictions
6. Export HMM definitions for player

https://github.com/mpi-sws-rse/antevents-examples/tree/master/lighting_replay_app/analysis

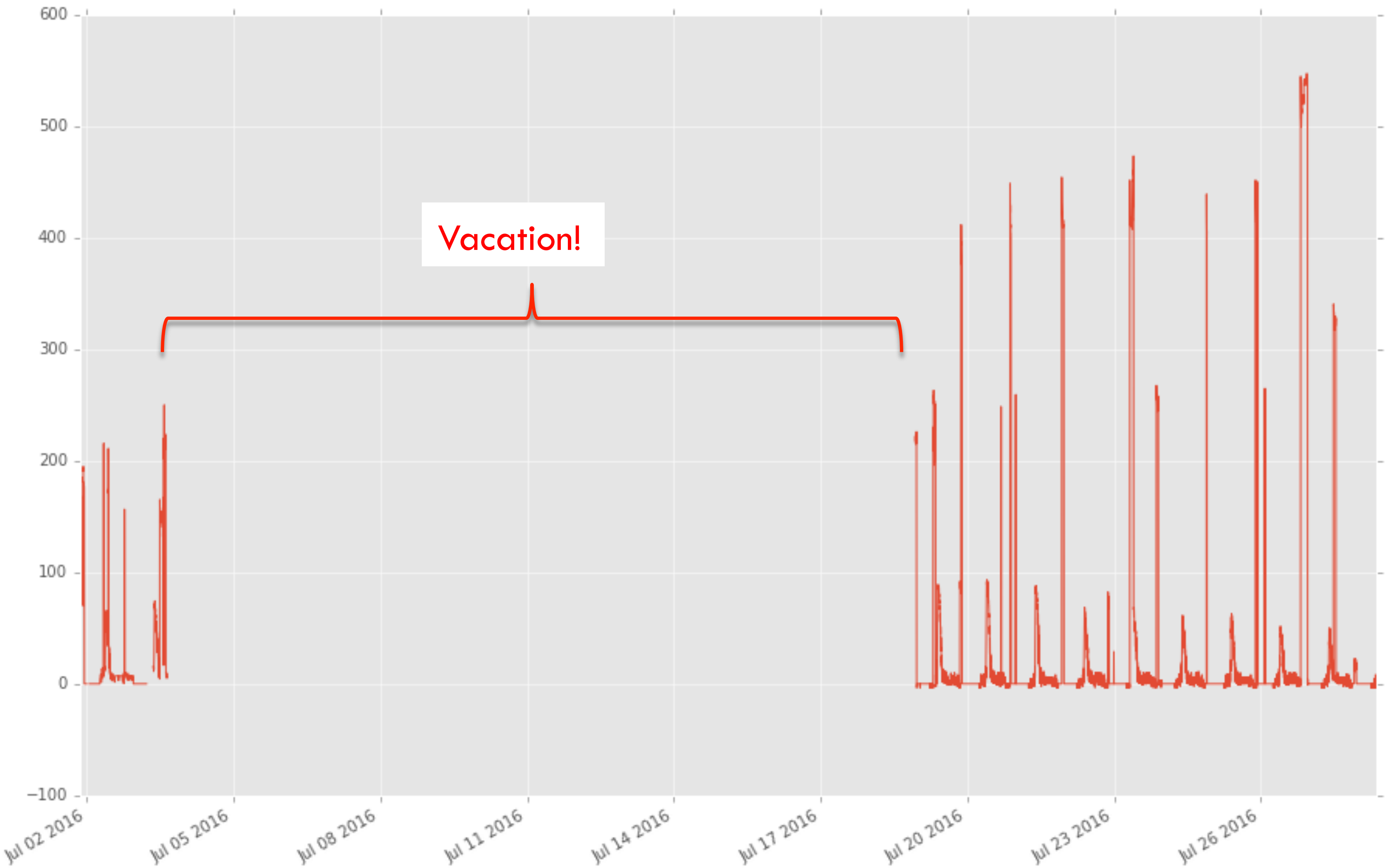
Read and Process CSV Files
(AntEvents running in a Jupyter Notebook)




```
reader.fill_in_missing_times()\
    .passthrough(raw_series_writer)\
    .transduce(SensorSlidingMeanPassNaNs(5)).select(round_event_val).passthrough(smoothed_series_writer)\
    .passthrough(capture_nan_indexes).output_count()
```

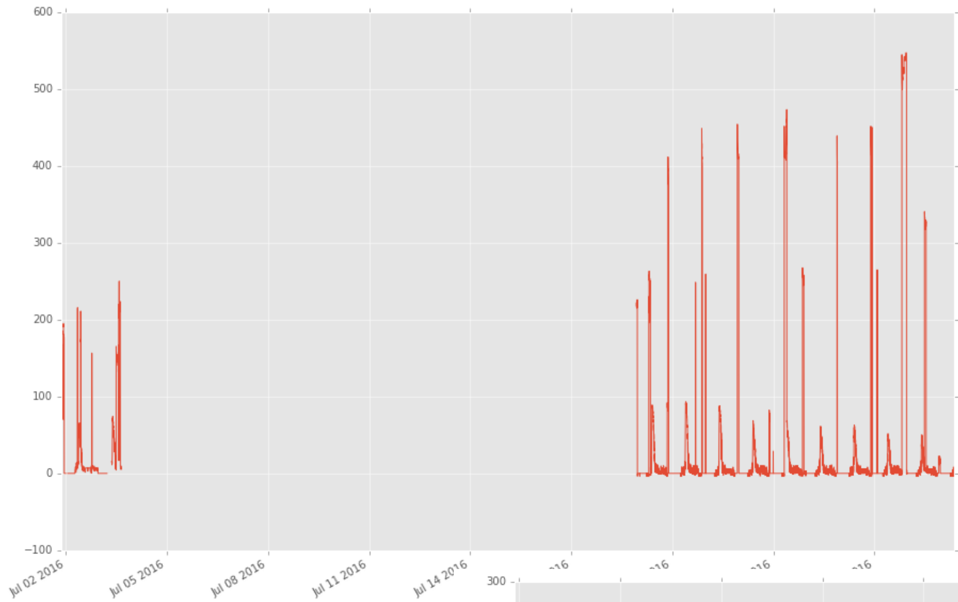
Raw Sensor Data: Entire Set

Front room

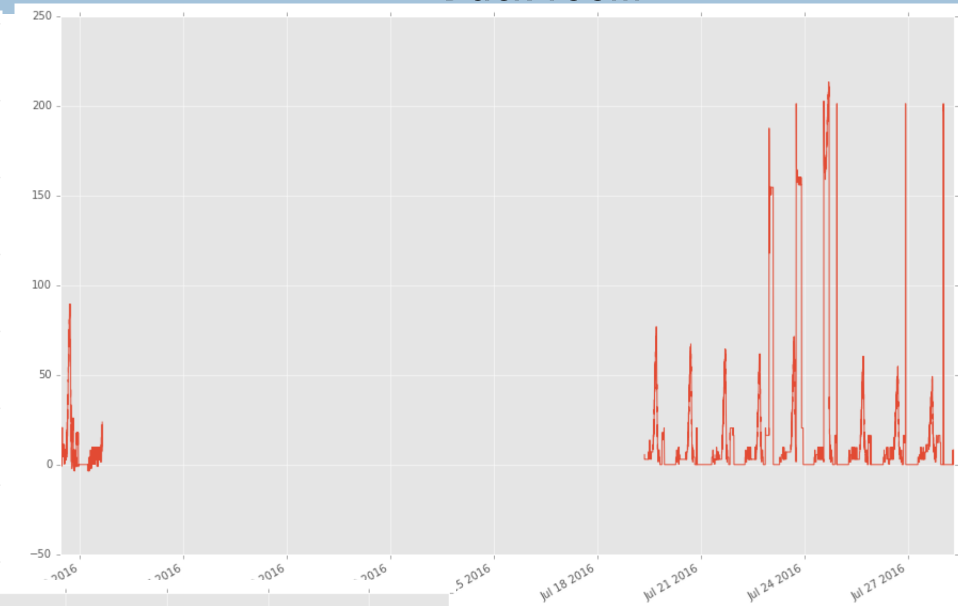


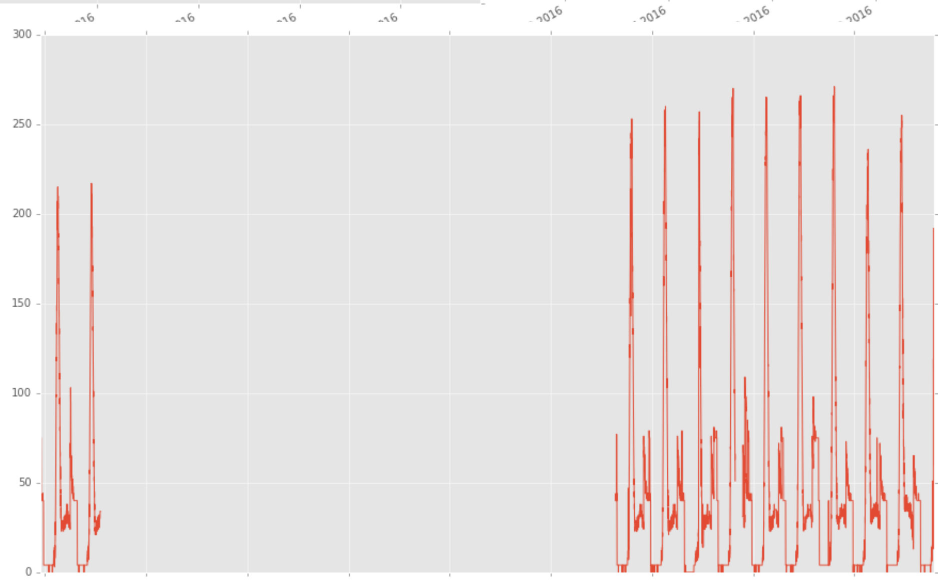
Raw Sensor Data: Entire Set

Front room



Back room

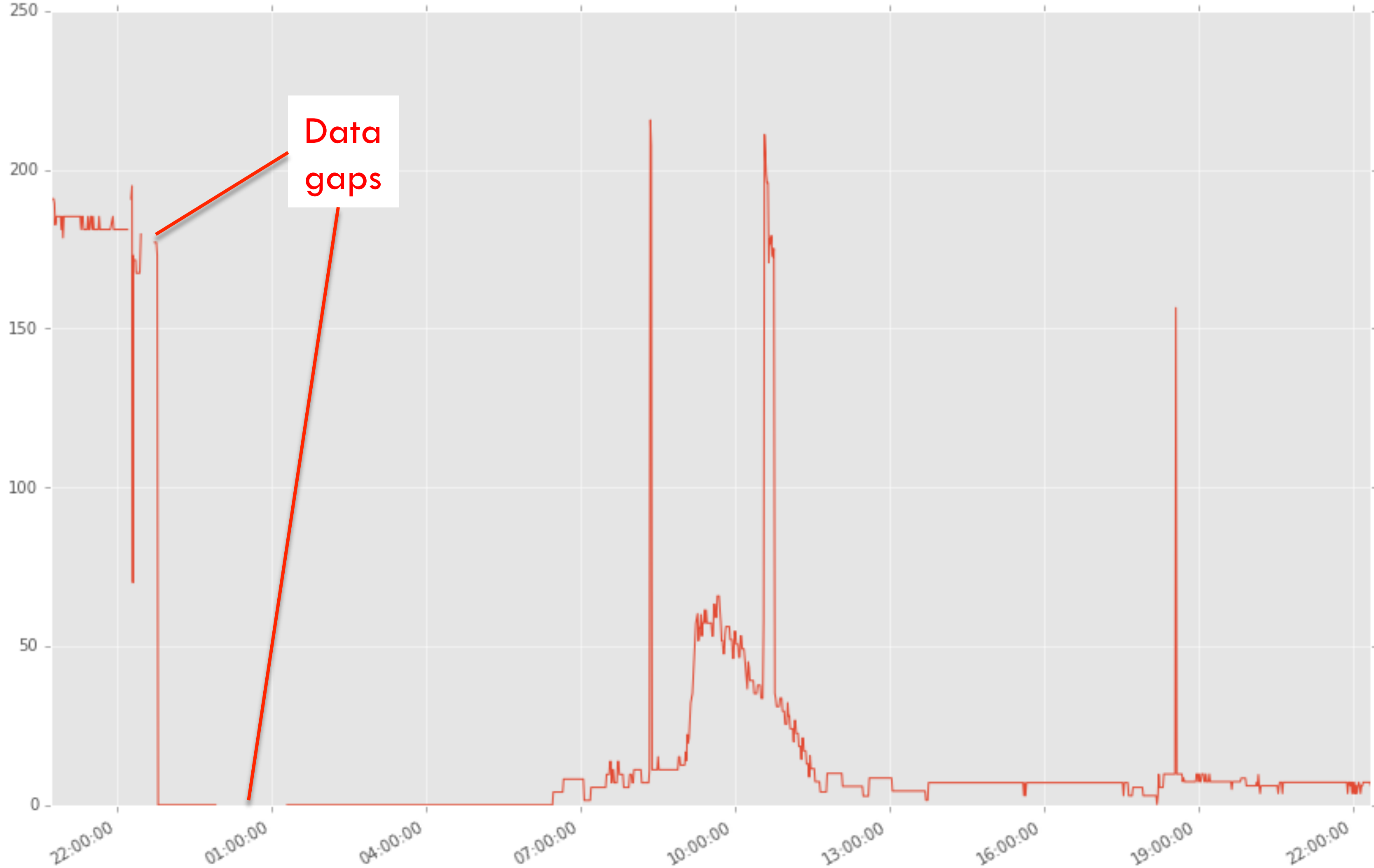




Dining room

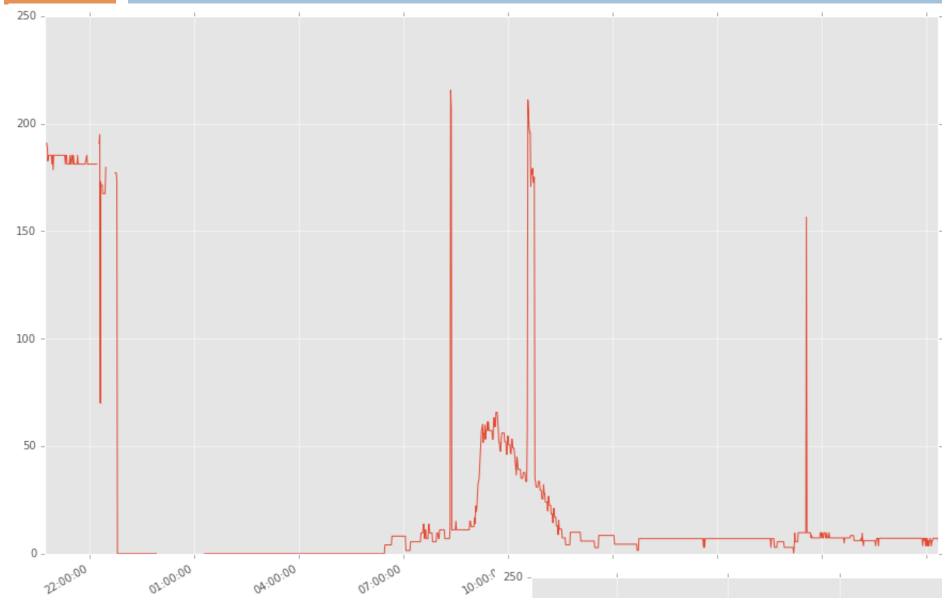
Raw Sensor Data: Last Day Only

Front room

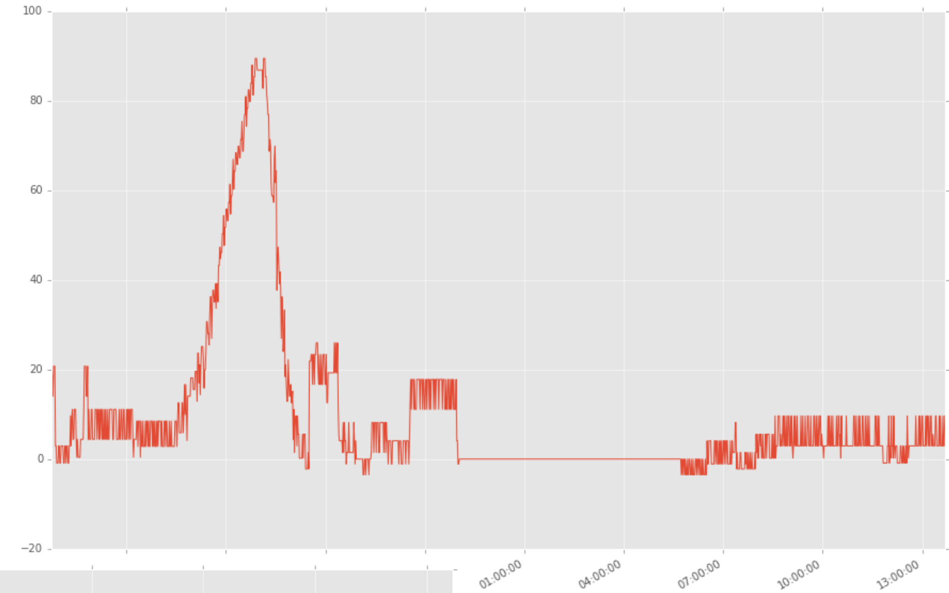


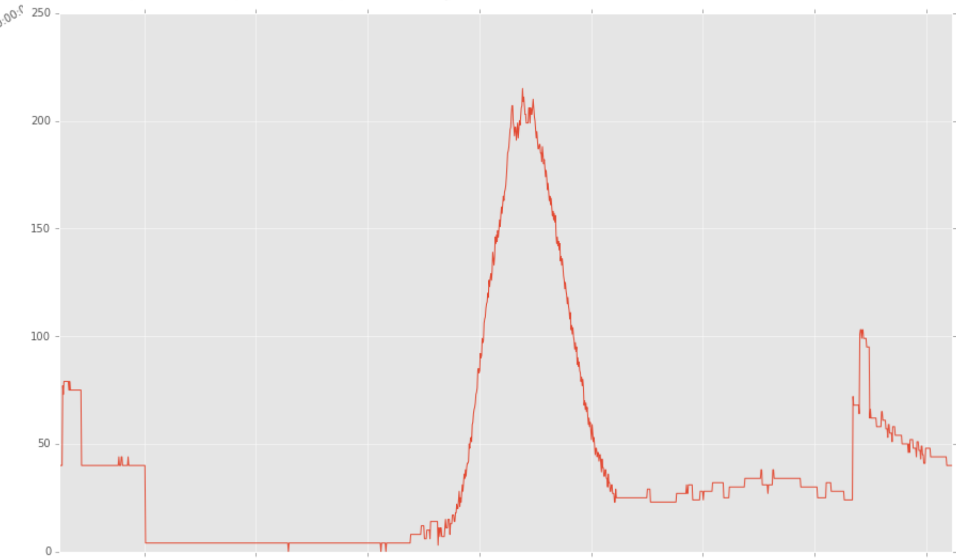
Raw Sensor Data: Last Day Only

Front room



Back room

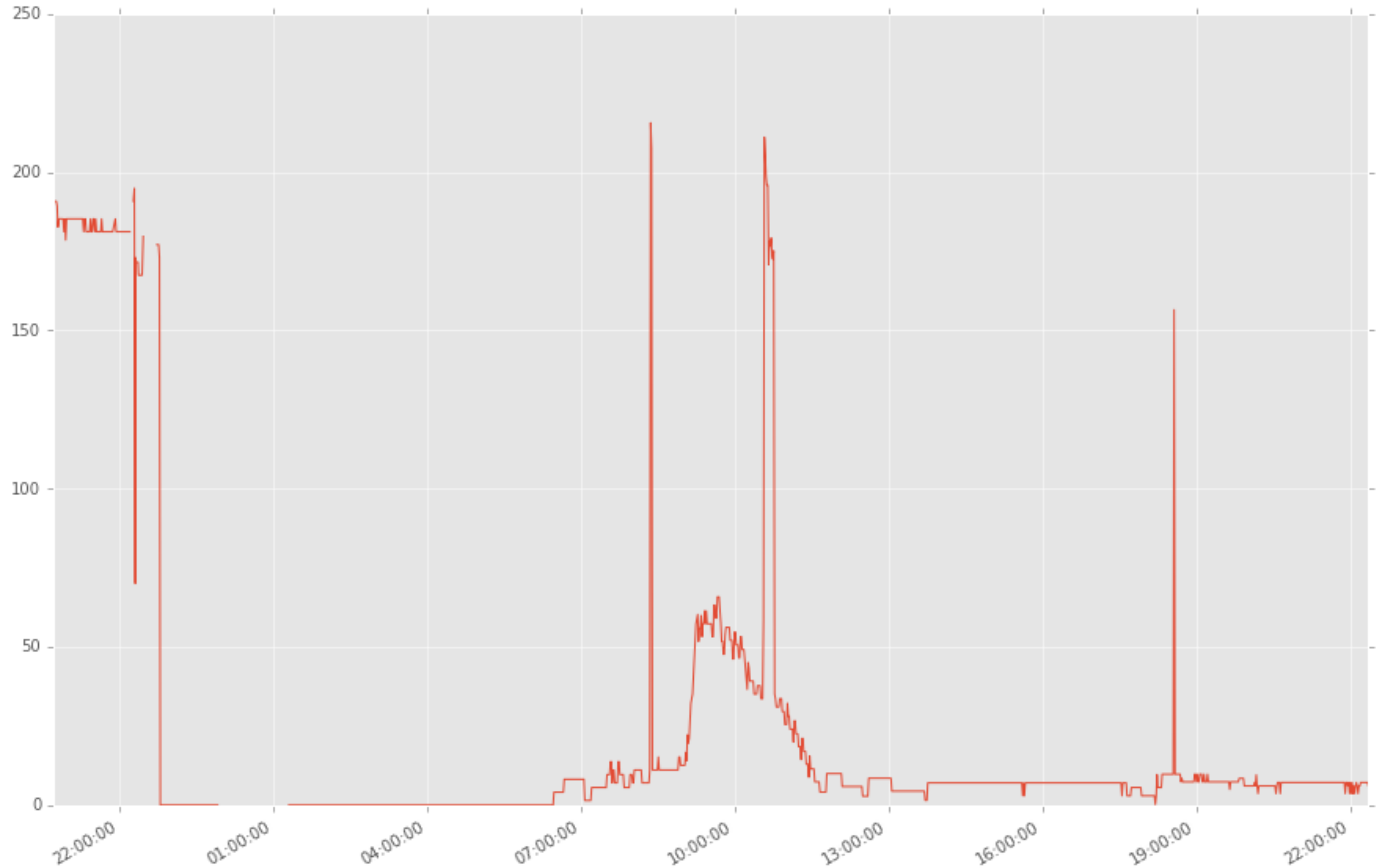




Dining room

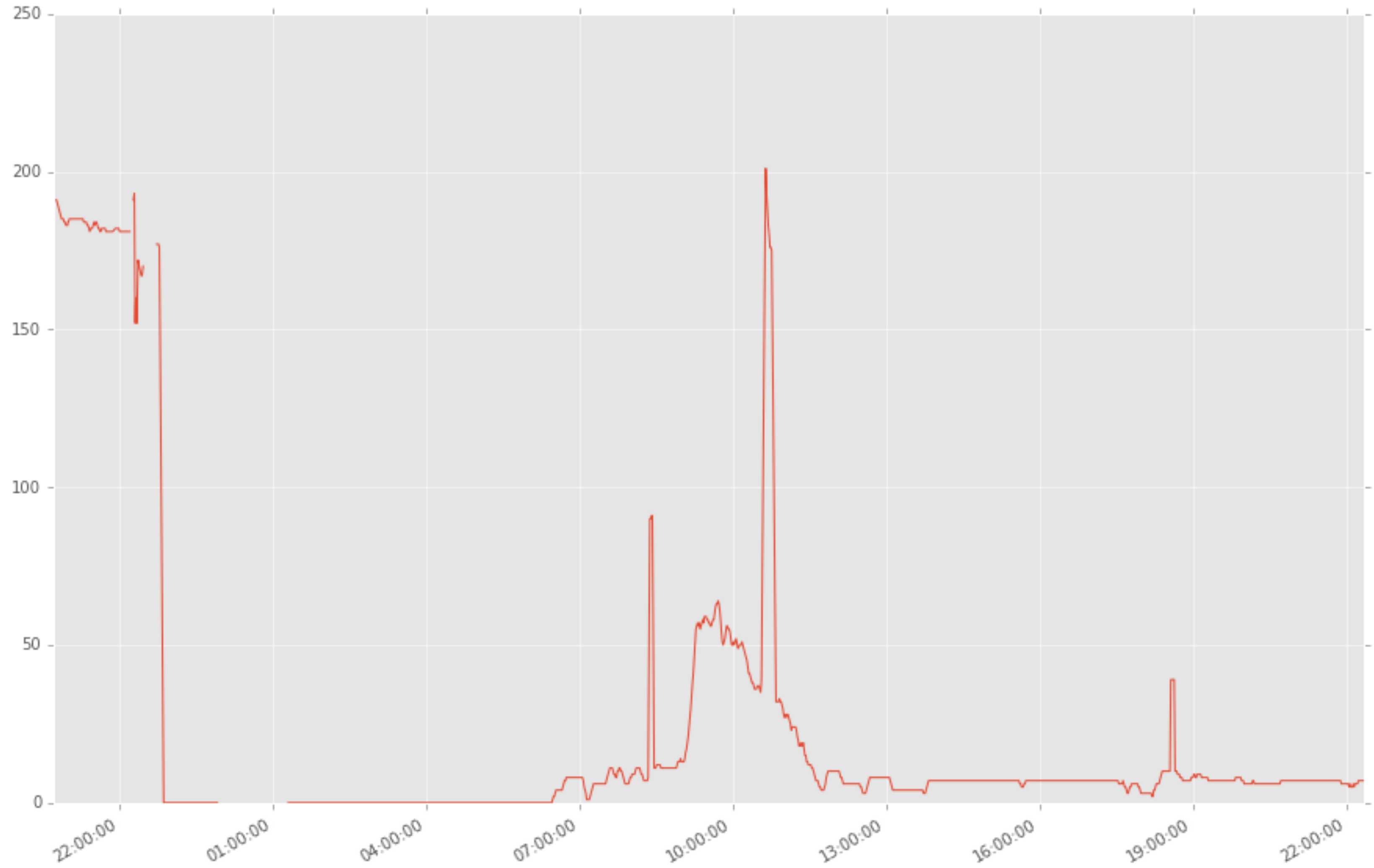
Data Processing: Raw Data

Front room, last day



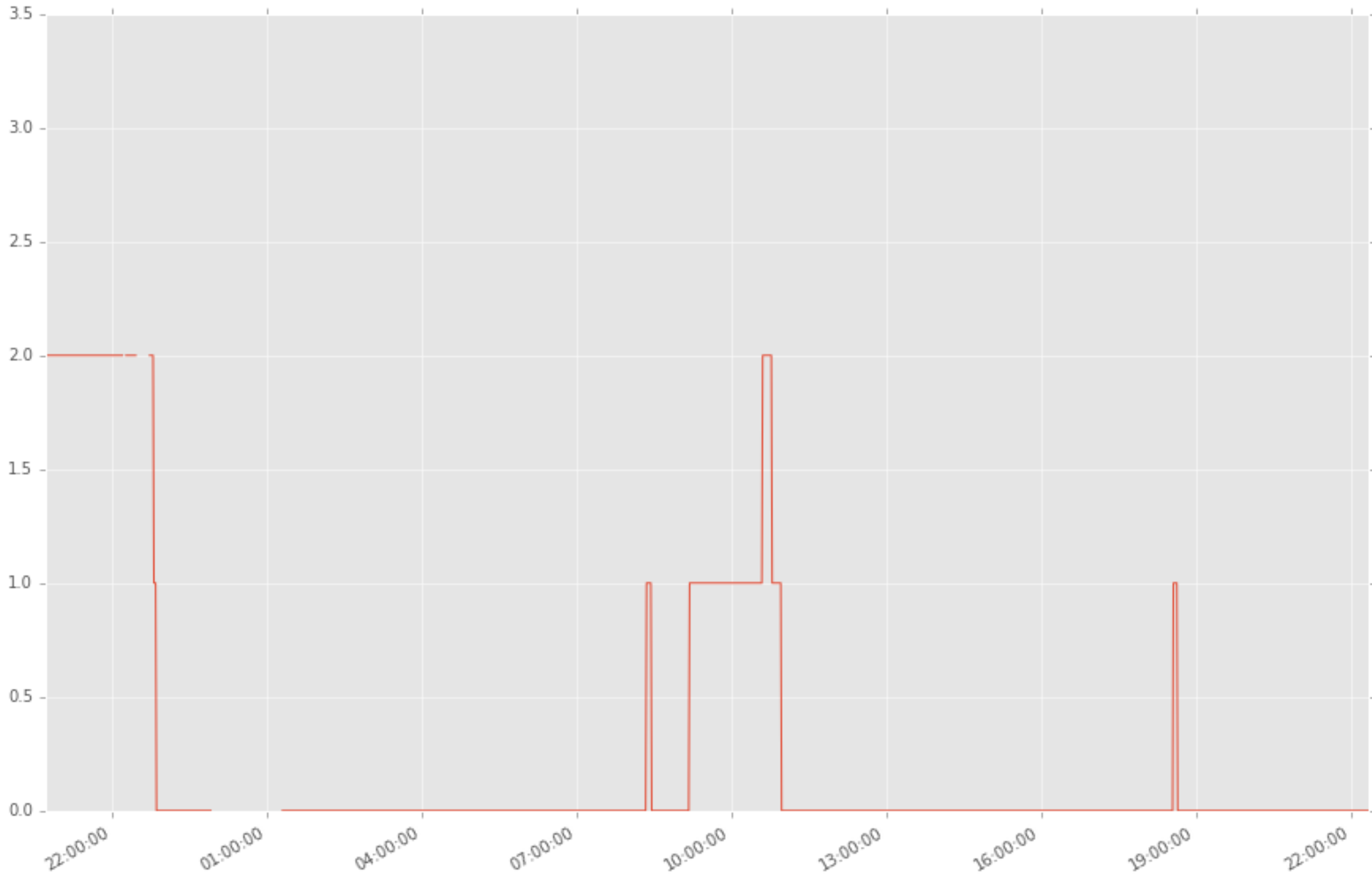
Data Processing: Smoothed Data

Front room, last day



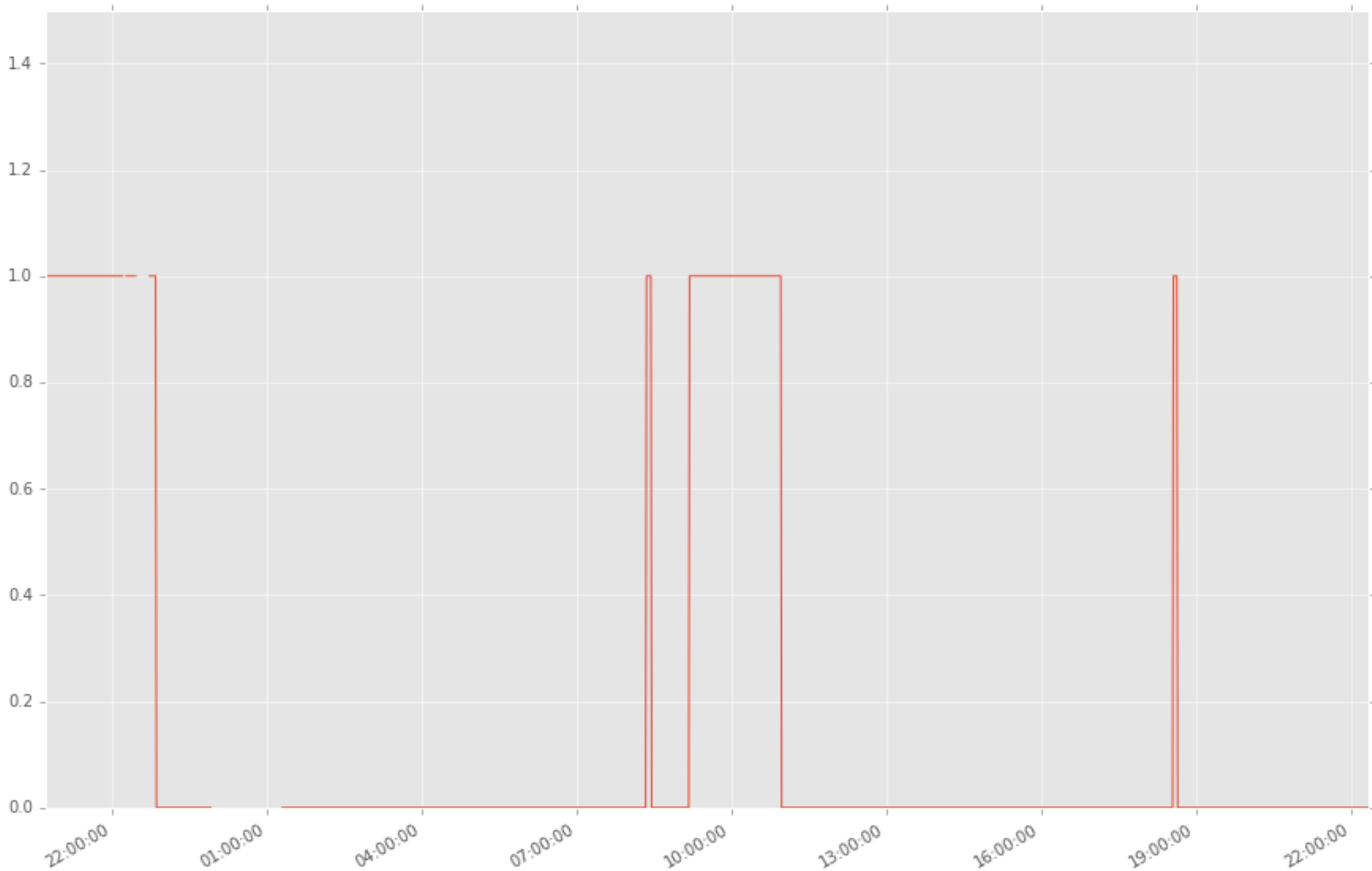
Data Processing: K-Means Clustering

Front room, last day



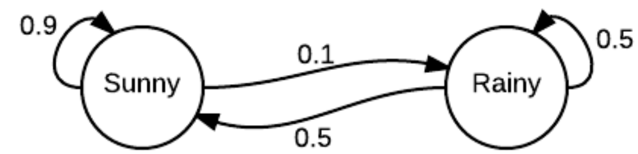
Data Processing: Mapping to on-off values

Front room, last day



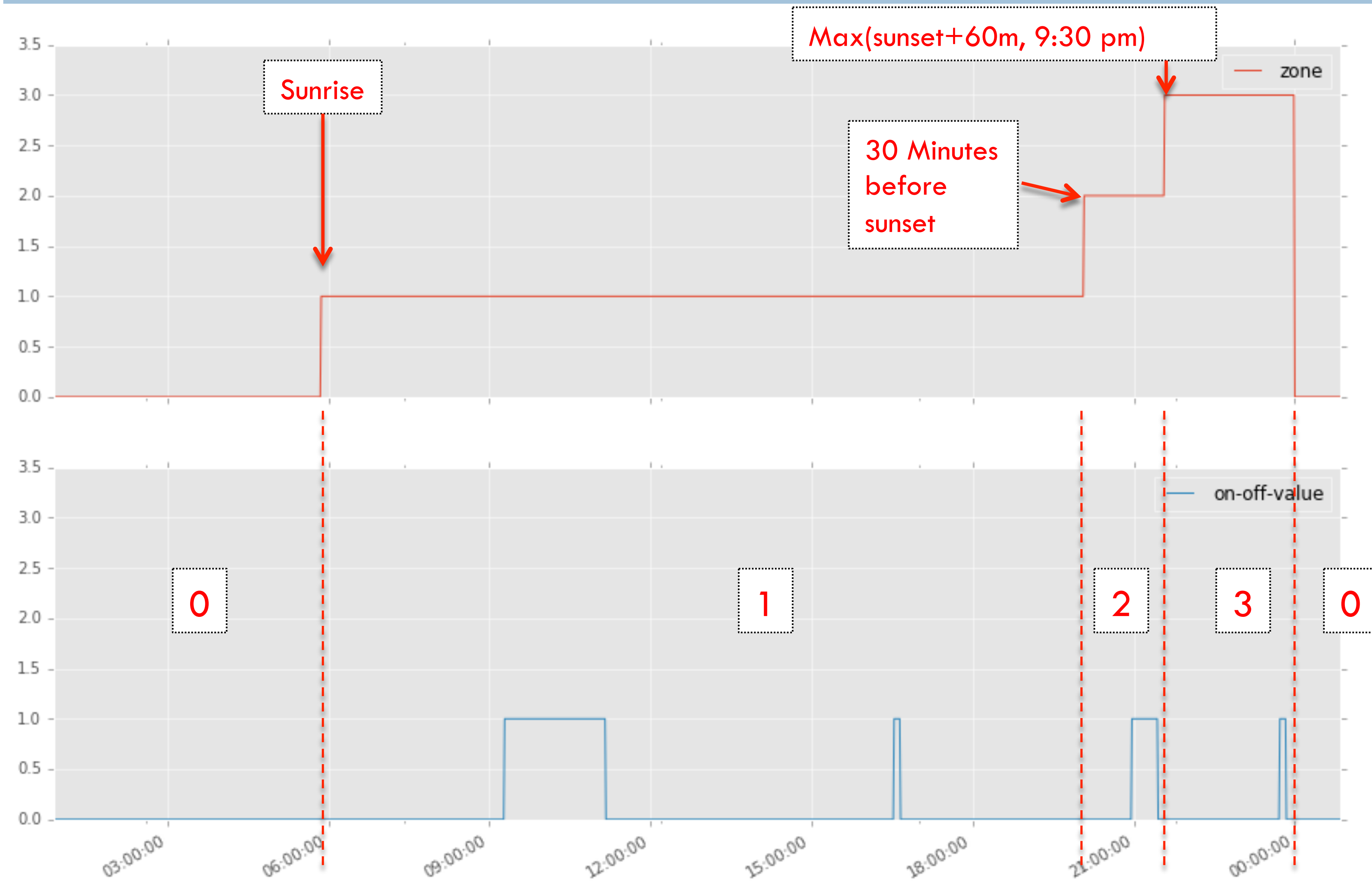
Hidden Markov Models (HMMs)

- In a *Markov process*, the probability distribution of future states is determined only by the current state, not on the sequence of events that preceded it.
- In a HMM, the states are not visible to the observer, only the outputs ("emissions").
- In a machine learning context, we are given a sequence of emissions and a number of states. We want to infer the state machine.
- The `hmmlearn` library will do this for us.
  - https://github.com/hmmlearn/hmmlearn



Example Markov process (from Wikipedia)

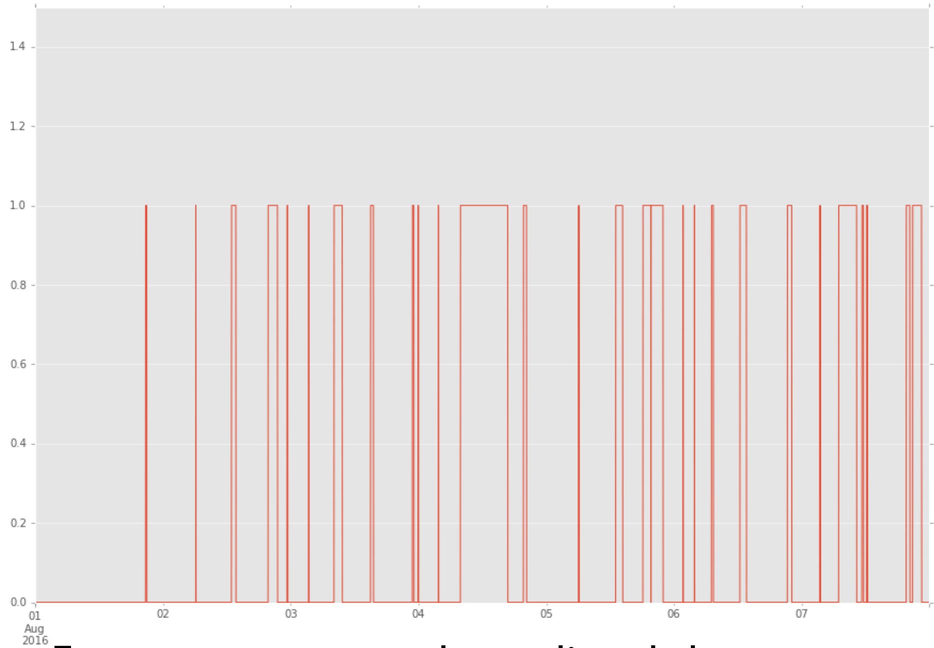
Slicing Data into Time-based "Zones"



HMM Training and Prediction Process

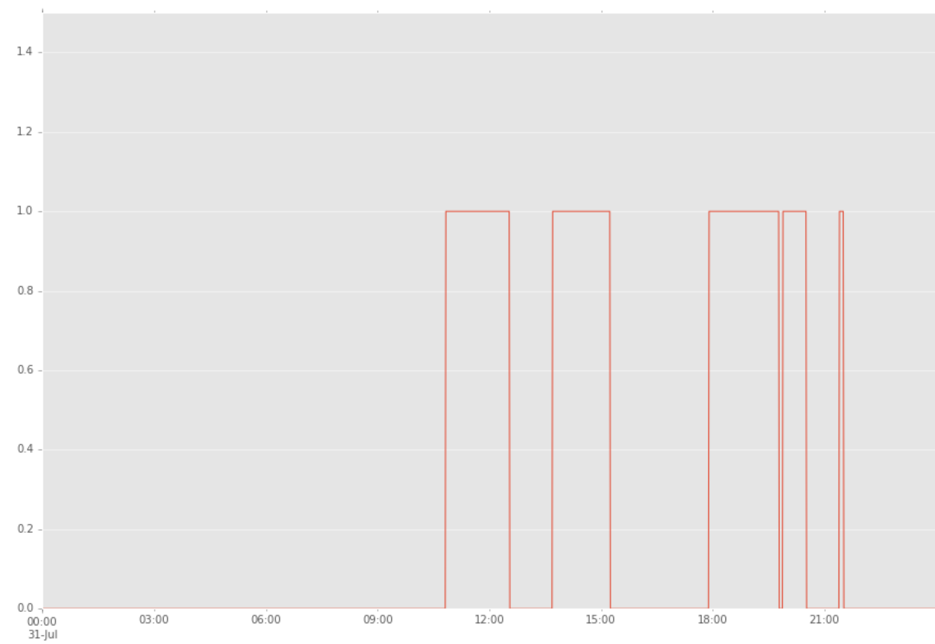
1. Build a list of sample subsequences for each zone
   - Drop the timestamps
   - Beak into separate sequences at zone boundaries and NaNs
2. Guess a number of states (e.g. 5)
3. For each zone, create an HMM and call `fit()` with the subsequences
4. For each zone of a given day:
   - Run the associated HMM to generate N samples for an N minute zone duration
   - Associated a computed timestamp with each sample

HMM Predicted Data



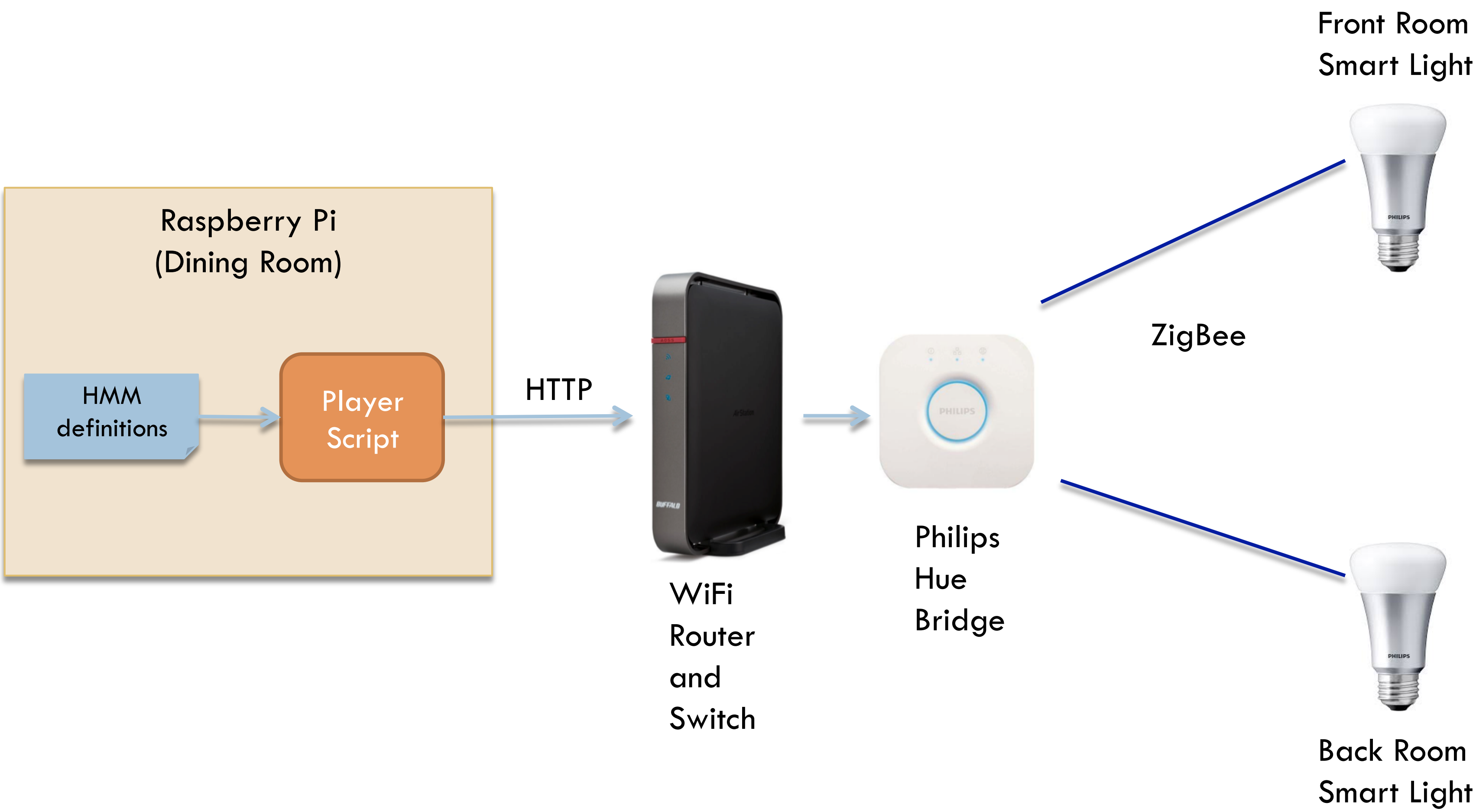
Front room, one week predicted data

Front room, one day predicted data



Replaying the Lights

Lighting Replay Application: Replay



Logic of the Replay Script

- Use phue library to control lights
- Reuse time zone logic and HMMs from analysis
- Pseudo-code:

```
Initial testing of lights
while True:
    compute predicted values for rest of day
    organize predictions into a time-sorted list of on/off events
    for each event:
        sleep until event time
        send control message for event
    wait until next day
```

https://github.com/mpi-sws-rse/antevents-examples/blob/master/lighting_replay_app/player/lux_player.py

Parting Thoughts

Acknowledgements

- Rupak Majumdar, Max Planck Institute for Software Systems
  - Co-designer of AntEvents
- Sze Ning Chng, Cambridge University
  - First user of AntEvents while interning at MPI
- Dmitrill Lourovitski, BayPiggies
  - Gave me advice regarding machine learning techniques

Lessons Learned

- An end-to-end project like this is a great way to learn a new area
- Applying machine learning to a problem can be very much a trial-and-error process
- Visualization is key to understanding/debugging these systems
- The Python ecosystem is great for both runtime IoT and offline analytics

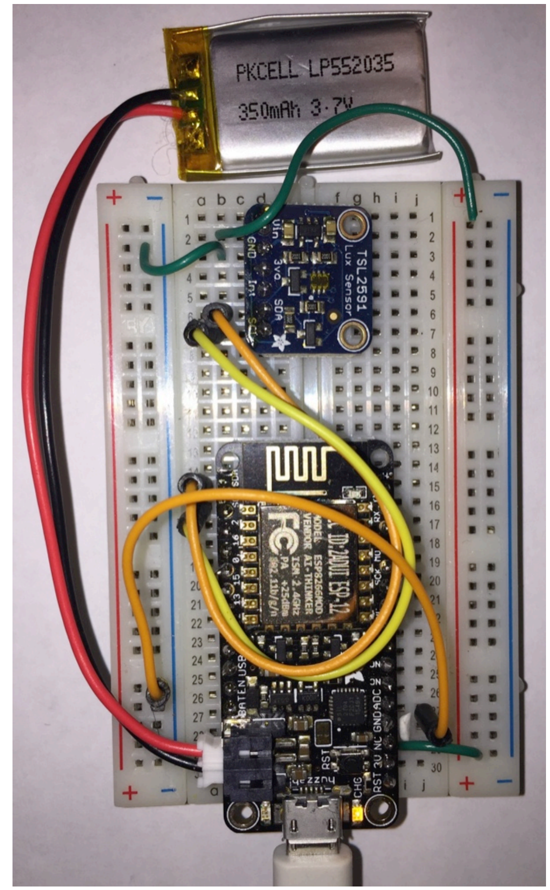
Future Work

- Gather more data and re-try other machine learning algorithms
- Integrate AntEvents with visualization (looking at Bokeh)
- What are the right abstractions for IoT analytics?

ESP8266 Demo

1. python3.5

```
MicroPython v1.8.5-10-g0e69e6b on 2016-10-17; ESP module with ESP8266
Type "help()" for more information.
>>> from antevents import *
>>> from tsl2591 import Tsl2591
>>> tsl = Tsl2591('lux-1')
>>> tsl.sample()
170.0544
>>> sched = Scheduler()
>>>
>>> class Output:
...     def on_next(self, x):
...         print(x)
...     def on_completed(self):
...         pass
...     def on_error(self, e):
...         pass
...
>>> sched.schedule_sensor(tsl, 2.0, Output())
<closure>
>>> sched.run_forever()
('lux-1', 89, 170.0544)
('lux-1', 91, 170.0544)
('lux-1', 93, 170.0544)
('lux-1', 95, 170.0544)
```



Thank You

Questions?

More information

Website and blog: https://data-ken.org

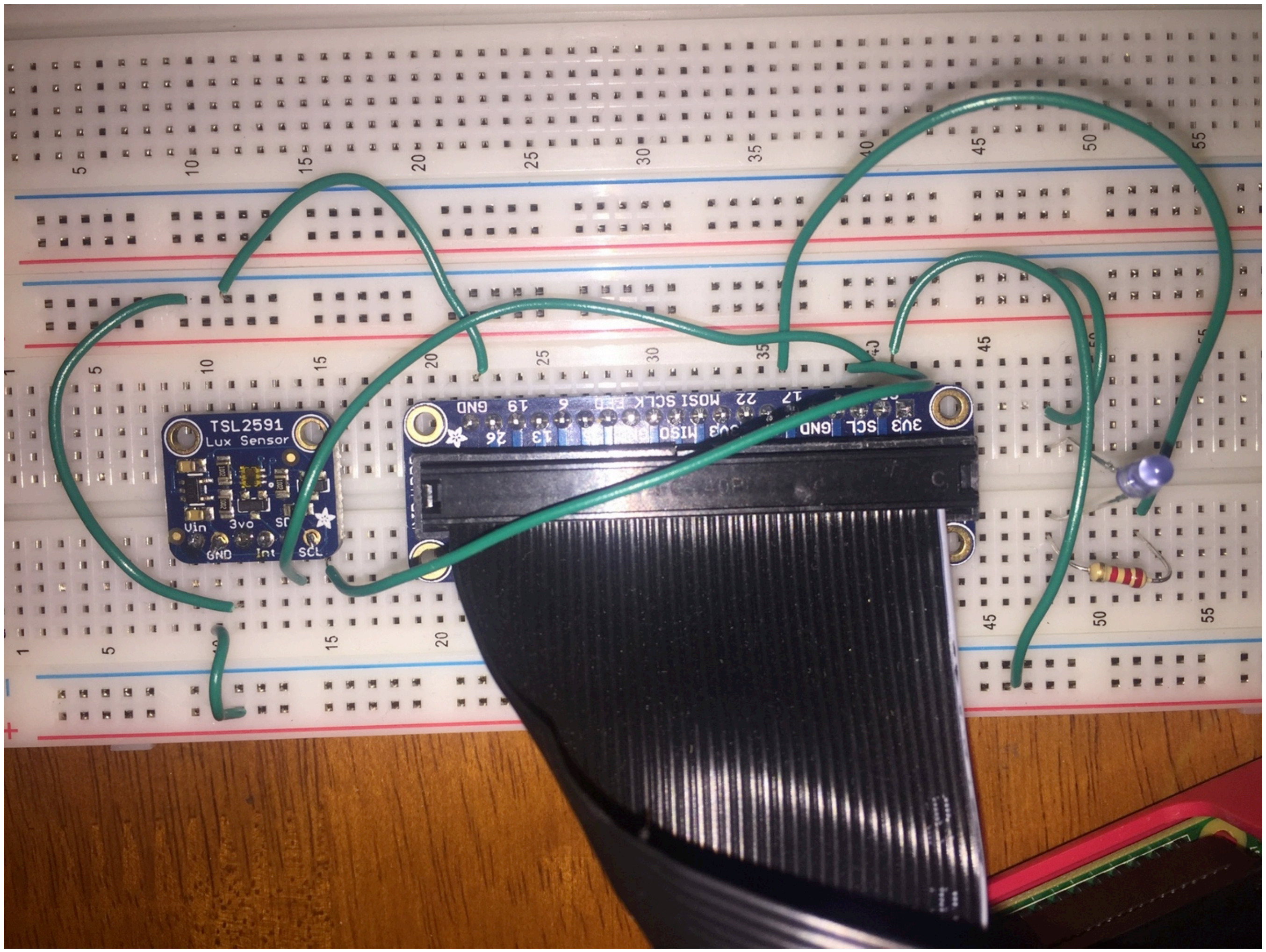
AntEvents: https://github.com/mpi-sws-rse/antevents-python

Examples (including lighting replay app): https://github.com/mpi-sws-rse/antevents-examples

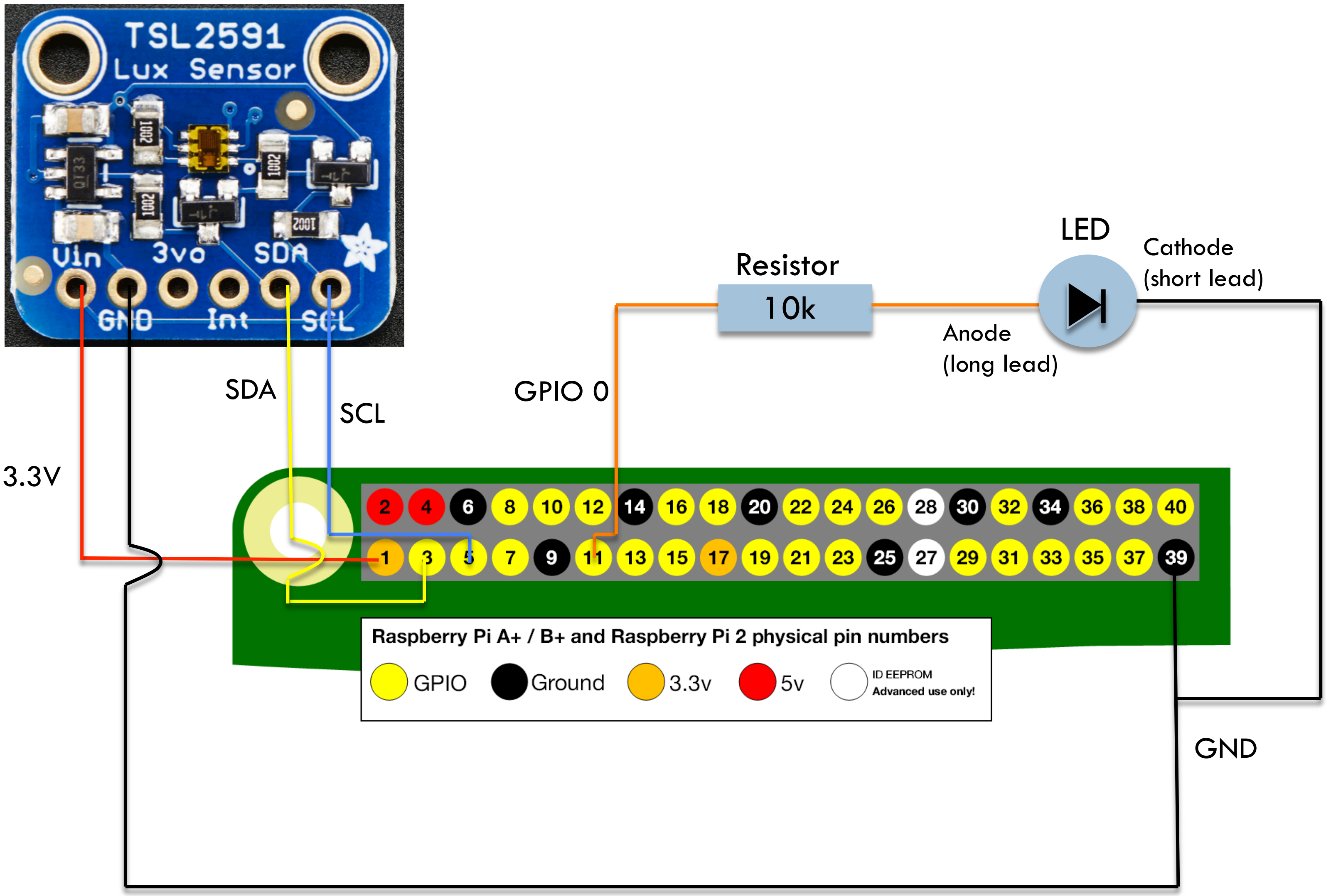
45

Additional Details

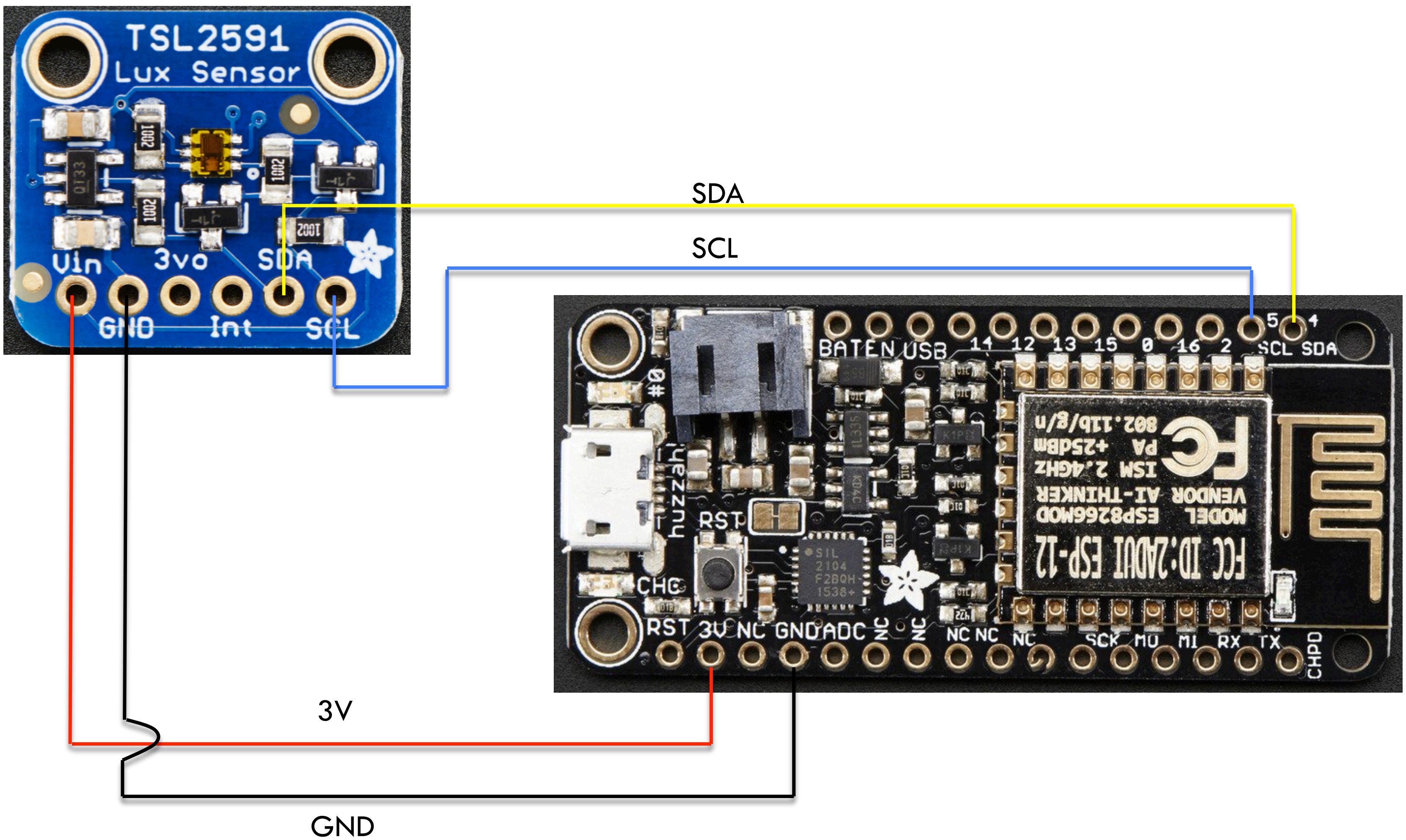
Raspberry Pi 2: Wiring Detail



Raspberry Pi 2: Wiring Diagram



ESP8266: Wiring Diagram



Third-party Resources

- Adafruit TSL2591 Lux Sensor tutorial

  https://learn.adafruit.com/adafruit-tsl2591
- Adafruit ESP8266 tutorial

  https://learn.adafruit.com/adafruit-feather-huzzah-esp8266
- LED tutorials
  - https://learn.adafruit.com/all-about-leds/overview
  - https://thepihut.com/blogs/raspberry-pi-tutorials/27968772-turning-on-an-led-with-your-raspberry-pis-gpio-pins
  - https://projects.drogon.net/raspberry-pi/gpio-examples/tux-crossing/gpio-examples-1-a-single-led/
- Micropython Getting Started on ESP8266

  https://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/intro.html

Machine Learning: Other Approaches Tried

- Feature data
  - Time of day, zone, on-off value N-samples back
  - Also tried the number of samples since the last value change – made results worse
- Algorithms tried
  - K-nearest neighbors
  - Logistic Regression
  - Decision Tree (classifier, probabilistic classifier, regressor)
- Pure probability approach
  - Build a probability distribution based on length of time at current value – worked fairly well
- Conclusion: need more sample data