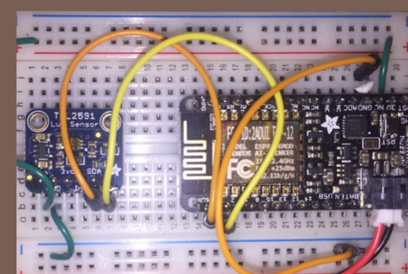
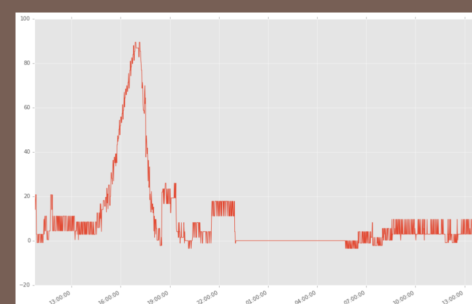
IOT, PYTHON, AND ML:

From Chips and Bits to Data Science



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



PyData Seattle July 6, 2017

Agenda

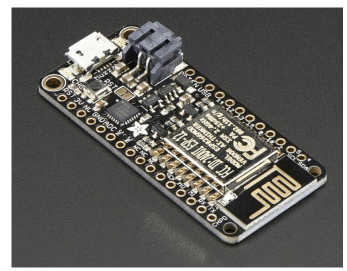
- Project overview
- Hardware
- Data capture
- Data analysis
- Player
- Parting thoughts

Why Python for IoT?

- High-level, easy to prototype ideas and explore options
- Runs on embedded devices

Raspberry Pi

- Linux “workstation”
- Can run CPython and full data science stack
- Not battery friendly



ESP8266

- System-on-a-chip with 32-bit CPU, WiFi, I/O
- Low power consumption
- Only 96K data memory!
- MicroPython to the rescure

- Python data analysis ecosystem

NumPy — Array and matrix processing

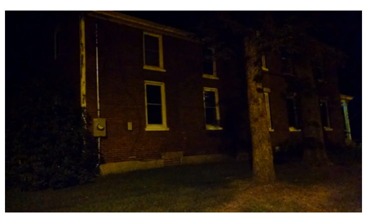
pandas — High level data analysis tools

SciPy — Numerical analysis routines

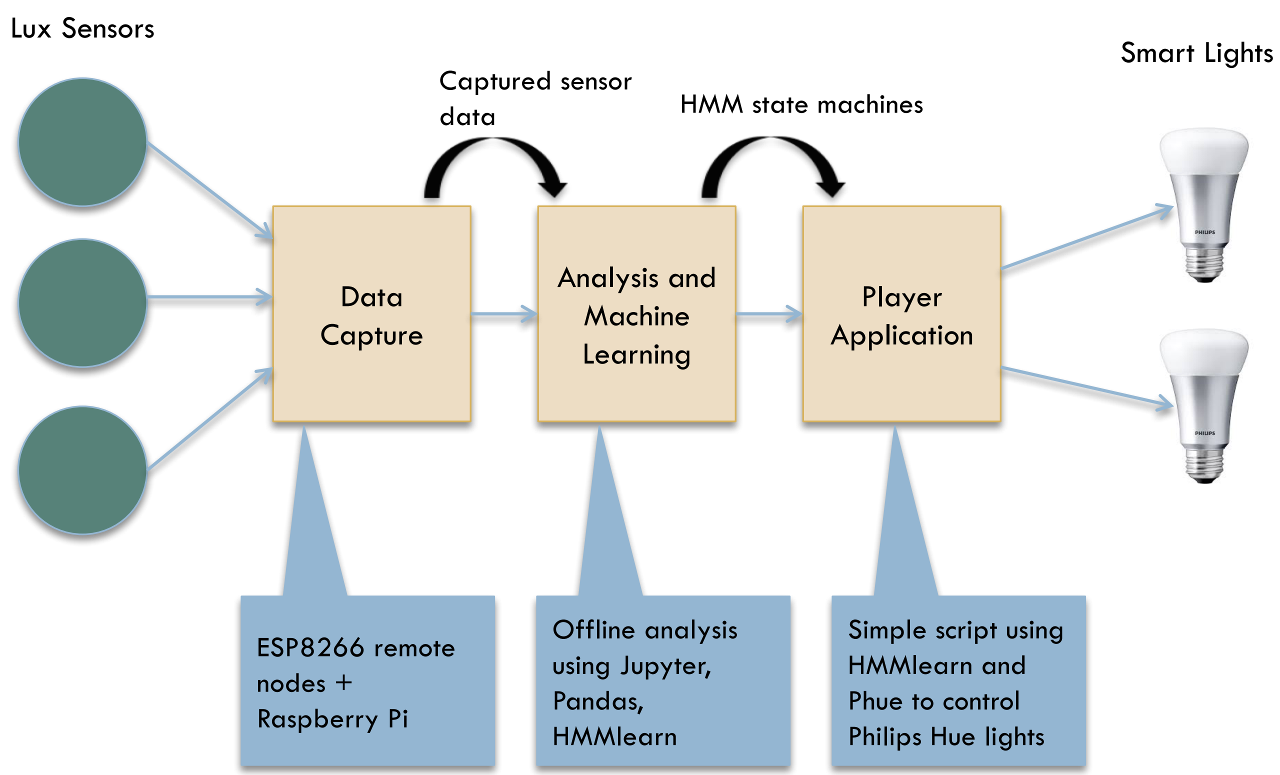
scikit learn HMMLearn — Machine learning

Project Motivation

- First thought about smart thermostat, but too dangerous
- Lighting is “safe”
- 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
- “Wouldn’t it be cool to use machine learning?”

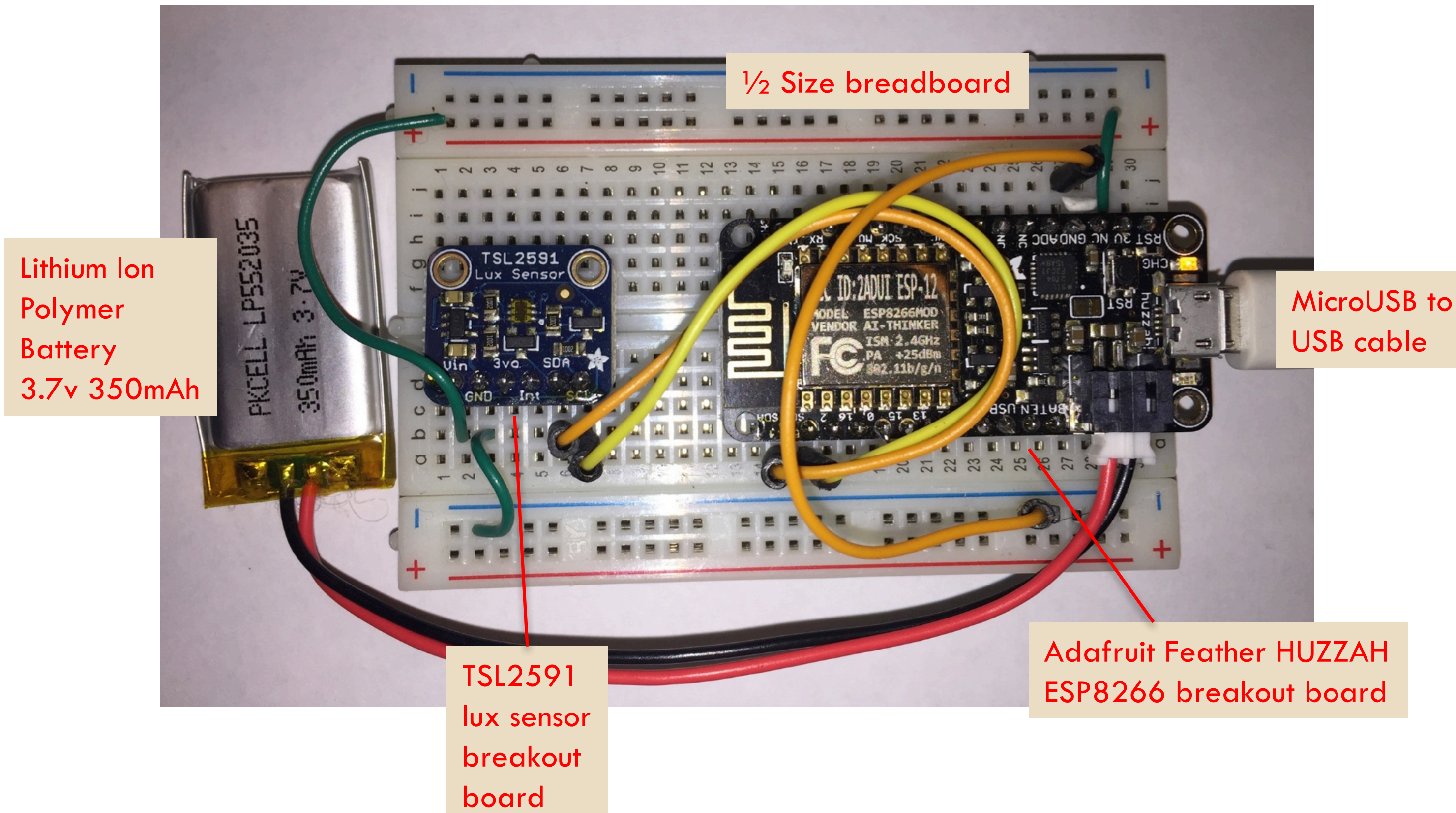


Lighting Replay Application

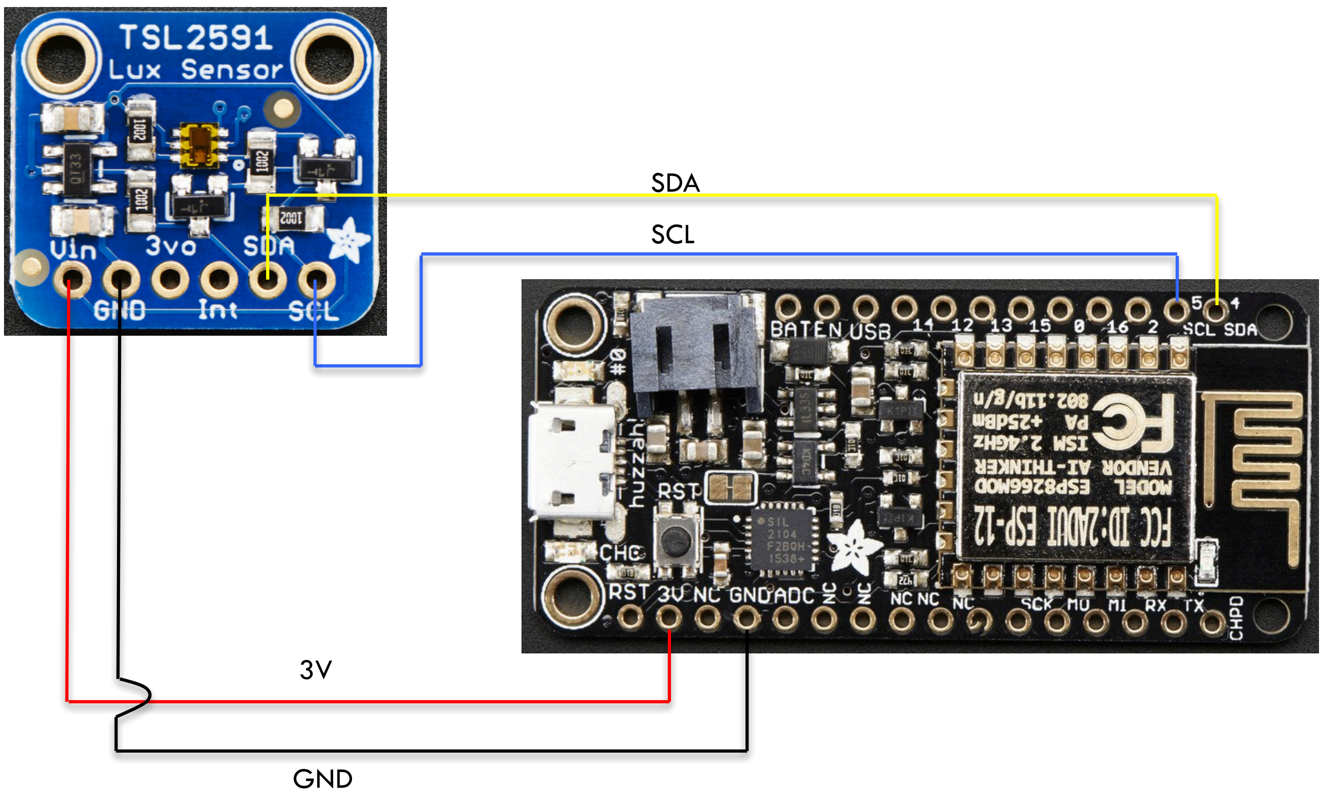


Hardware

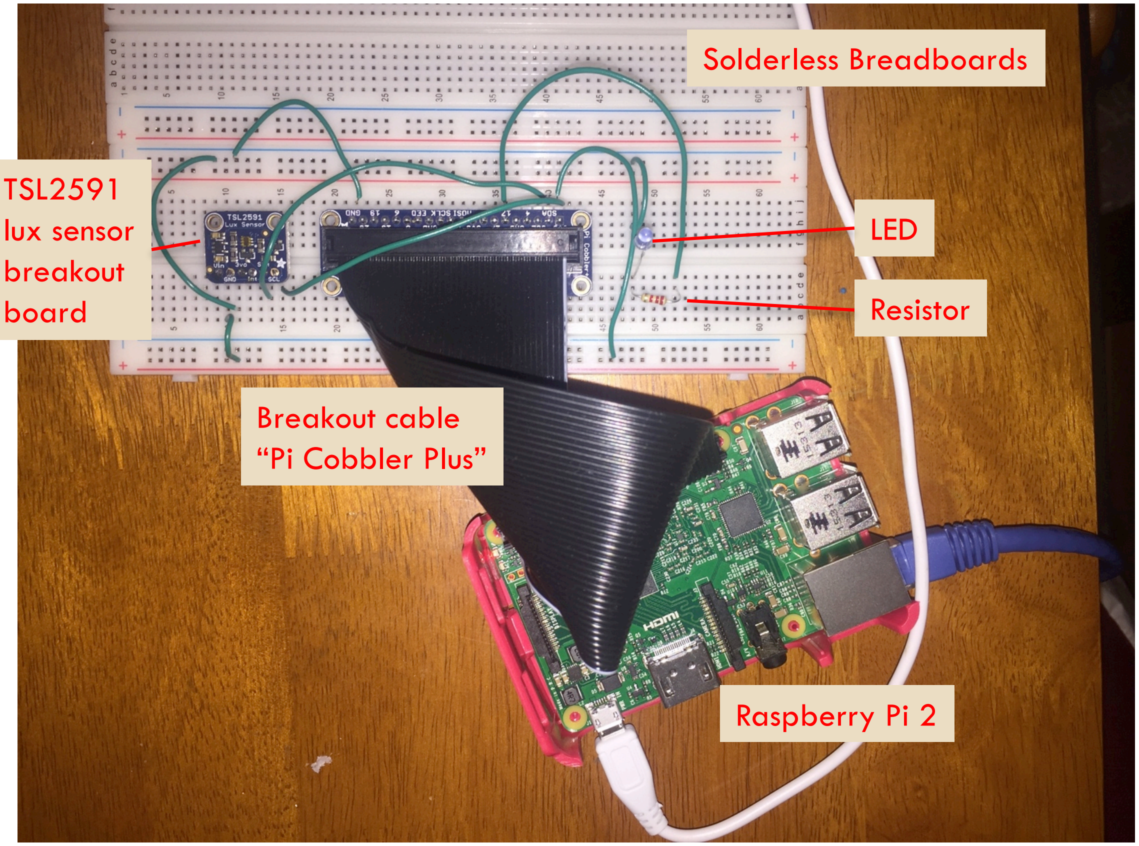
ESP8266


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

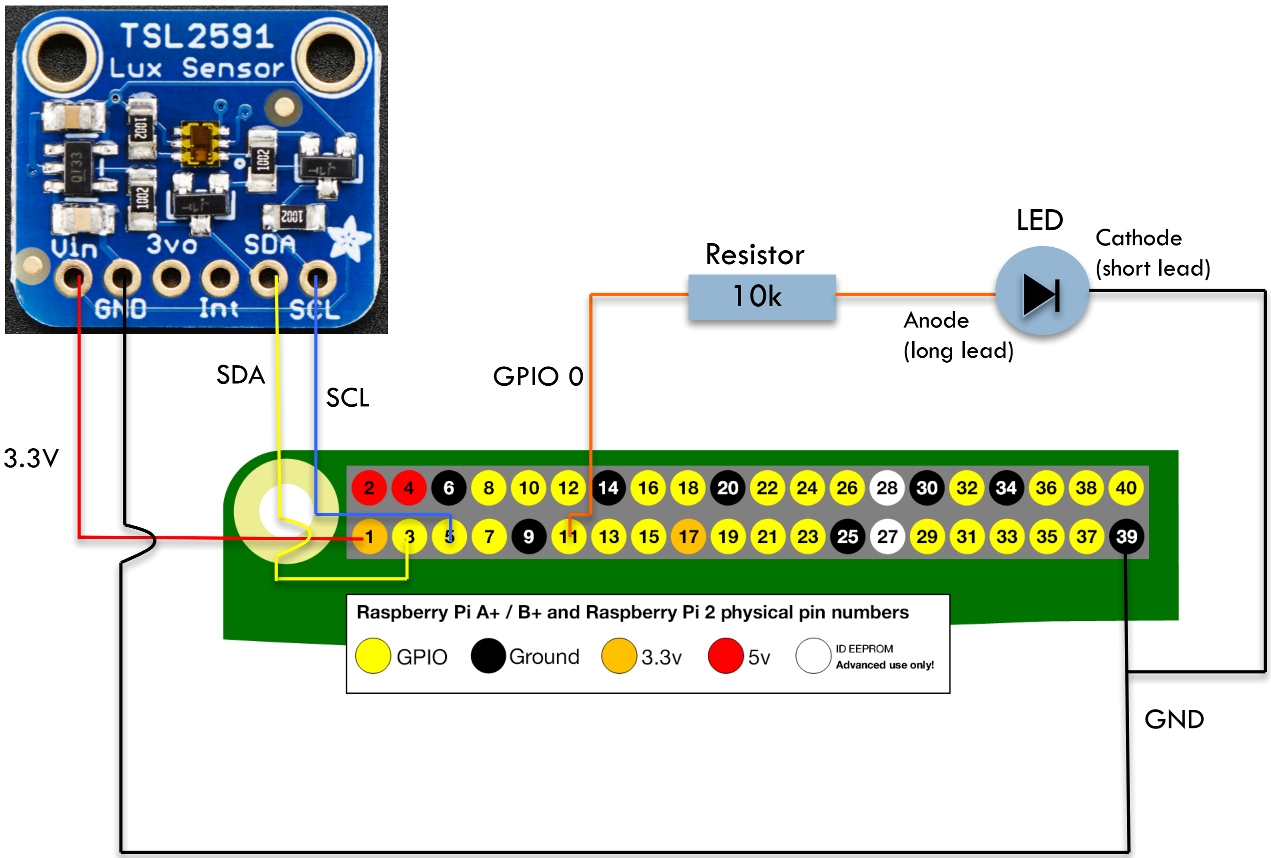
ESP8266: Wiring Diagram



Raspberry Pi

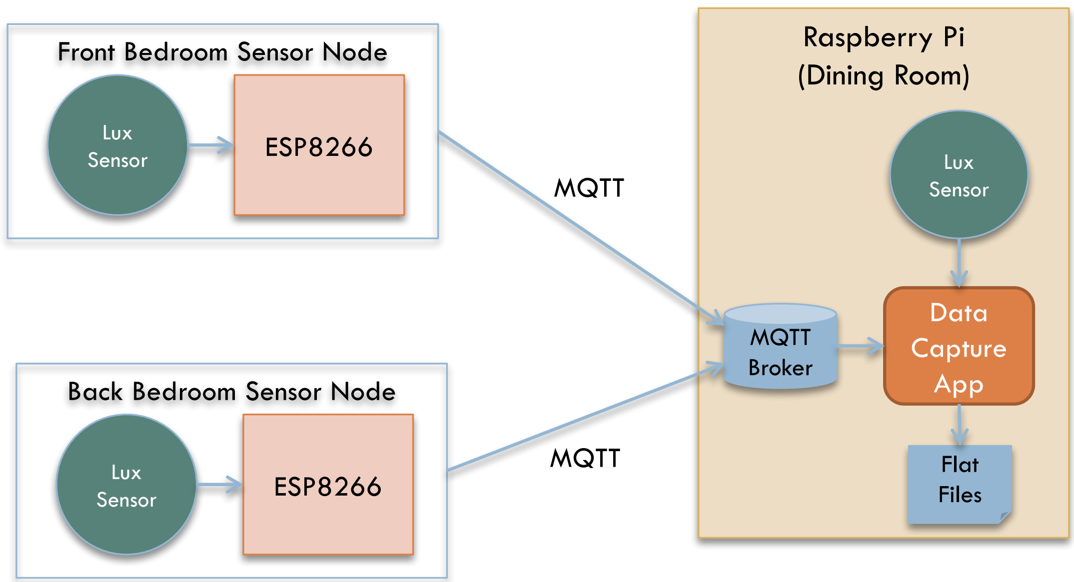


Raspberry Pi: Wiring Diagram



Data Capture

Lighting Replay Application: Capture



Event-driven IoT Code Can Be Ugly

```
def sample_and_process(sensor, mqtt_writer, xducer, completion_cb, error_cb):
    try:
        sample = sensor.sample()
    except StopIteration:
        final_event = xducer.complete()
        if final_event:
            mqtt_writer.send(final_event,
                             lambda: mqtt_writer.disconnect(lambda: completion_cb(False), error_cb), error_cb)
        else:
            mqtt_writer.disconnect(lambda: completion_cb(False), error_cb)
        return
    except Exception as e:
        error_cb(e)
        mqtt_writer.disconnect(lambda: pass, error_cb)
        return
    event = SensorEvent(sensor_id=sensor.sensor_id, ts=time.time(), val=
    csv_writer(event)
    median_event = xducer.step(event)
    if median_event:
        mqtt_writer.send(median_event,
                         lambda: completion_cb(True), error_cb)
    else:
        completion_cb(True)

def loop():
    def completion_cb(more):
        if more:
            event_loop.call_later(0.5, loop)
        else:
            print("all done, no more callbacks to schedule")
            event_loop.stop()
    def error_cb(e):
        print("Got error: %s" % e)
        event_loop.stop()
    event_loop.call_soon(lambda: sample_and_process(sensor, mqtt_writer,
```

Problems

1. Callback hell
2. Connecting of event streams intermixed with handling of runtime situations: normal flow, error, and end-of-stream conditions.
3. Low-level scheduling
4. async/await helps, but not much

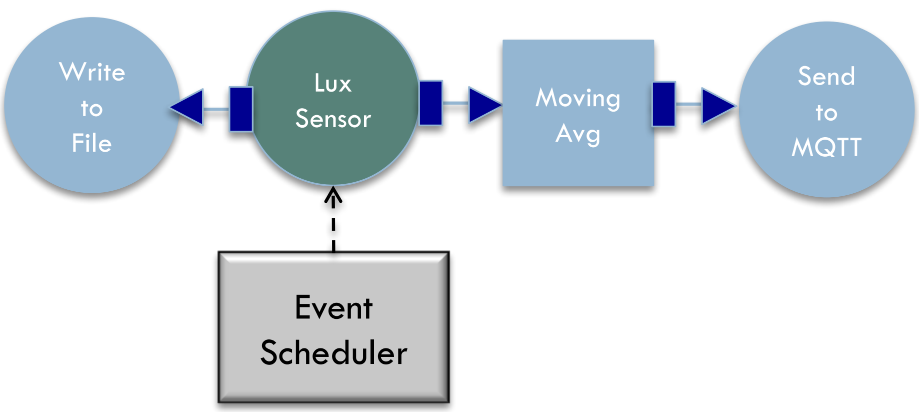
My Solution: ThingFlow

- What is ThingFlow?
  - A Domain Specific Language for IoT event processing
  - Runs on Python3 and MicroPython
- Co-creator
  - Rupak Majumdar, Scientific Director at Max Planck Institute for Software Systems
- Why did we create ThingFlow?
  - IoT event processing code can be very convoluted
  - No standardization of sensors, adapters, and transformations
  - Different frameworks for microcontrollers, edge processing, analytics

Simple ThingFlow Example

- Periodically sample a light sensor
- Write the sensed value to a local file
- Every 5 samples, send the moving average to MQTT Broker

Graphical Representation



Code

```
sensor.connect(file_writer('file'))
sensor.transduce(MovingAvg(5)).connect(mqtt_writer)
scheduler.schedule_periodic(sensor, 5)
```

ESP8266 ThingFlow Code

```
from thingflow import Scheduler, SensorAsOutputThing
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 = SensorAsOutputThing(Tsl2591())
writer = MQTTWriter(SENSOR_ID, BROKER, 1883,
                    'remote-sensors')
sched = Scheduler()
sched.schedule_sensor(sensor, SENSOR_ID, 60, writer)
sched.run_forever()
```

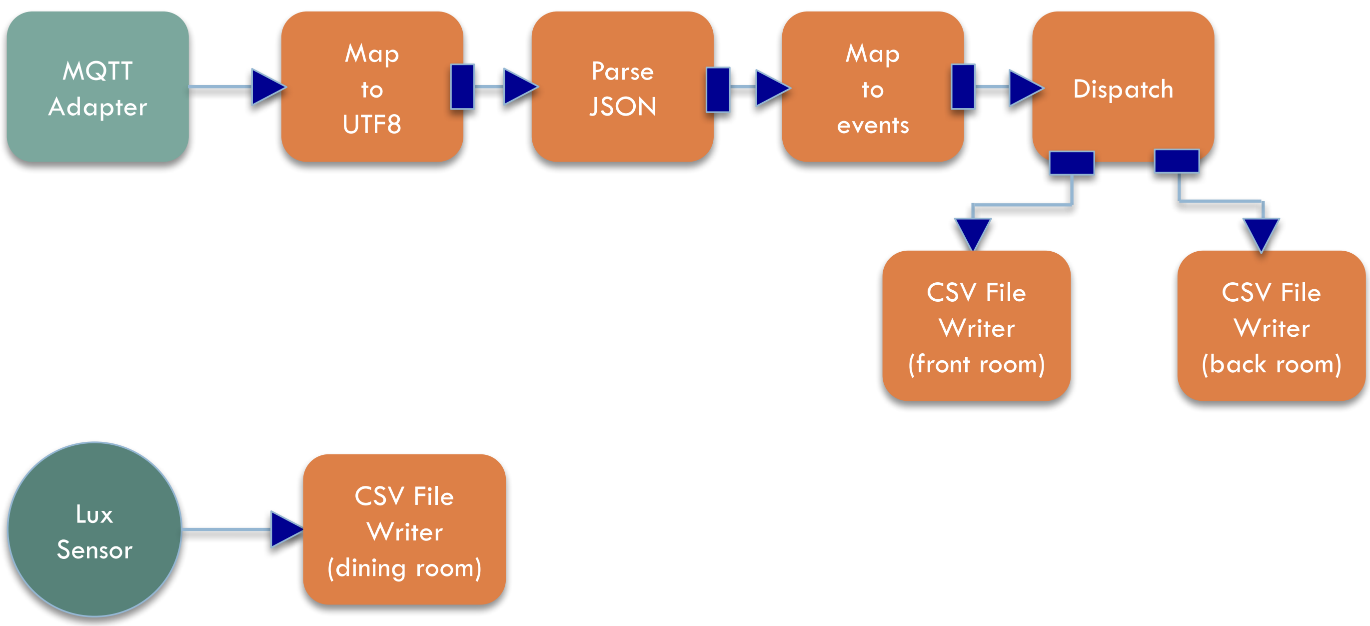
https://github.com/jfischer/micropython-tsl2591

Sample at 60 second intervals

The MQTT writer is connected to the lux sensor.

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

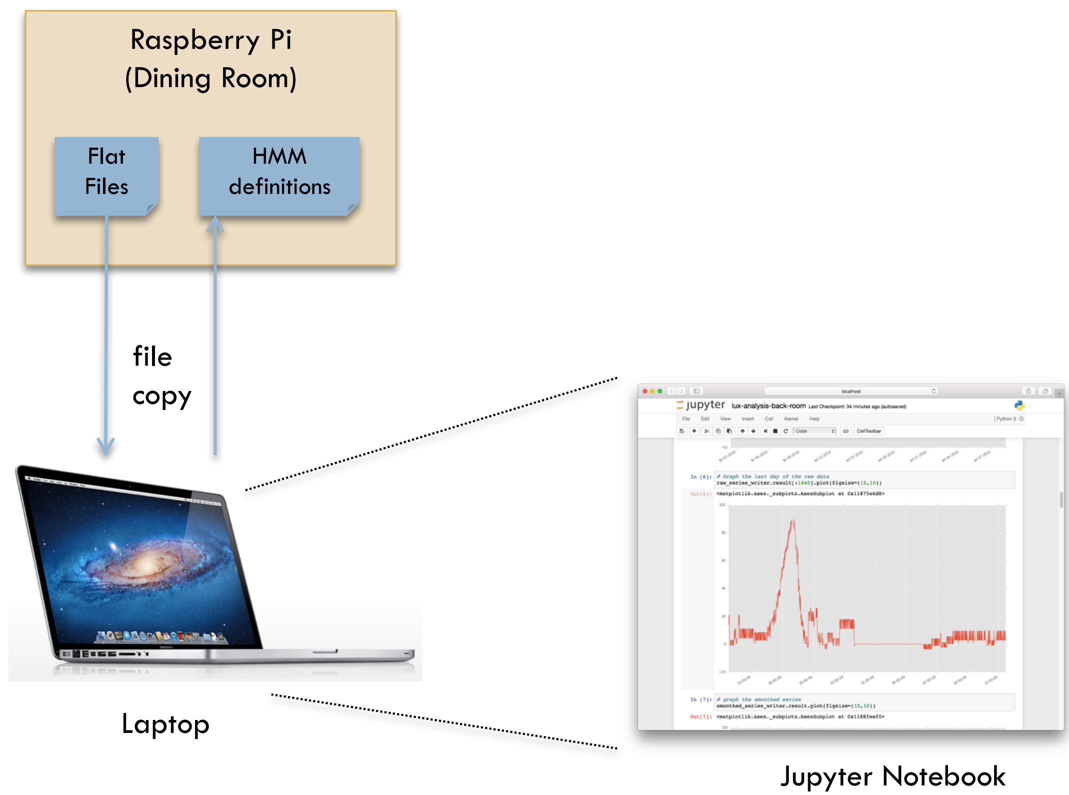
Raspberry Pi Code



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

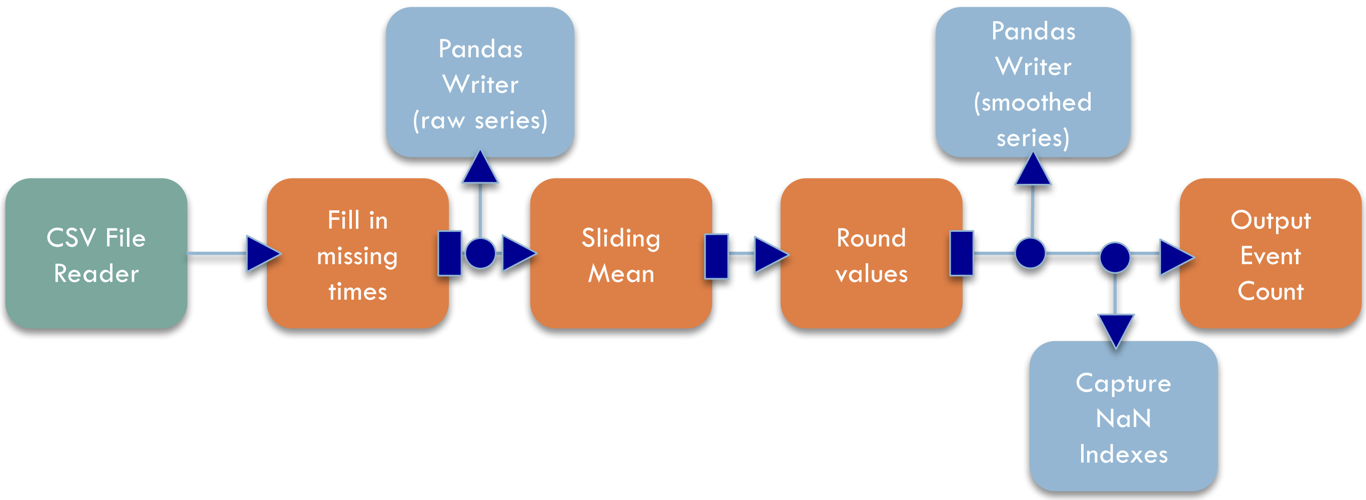
Data Analysis

Lighting Replay Application: Analysis



Preprocessing the Data

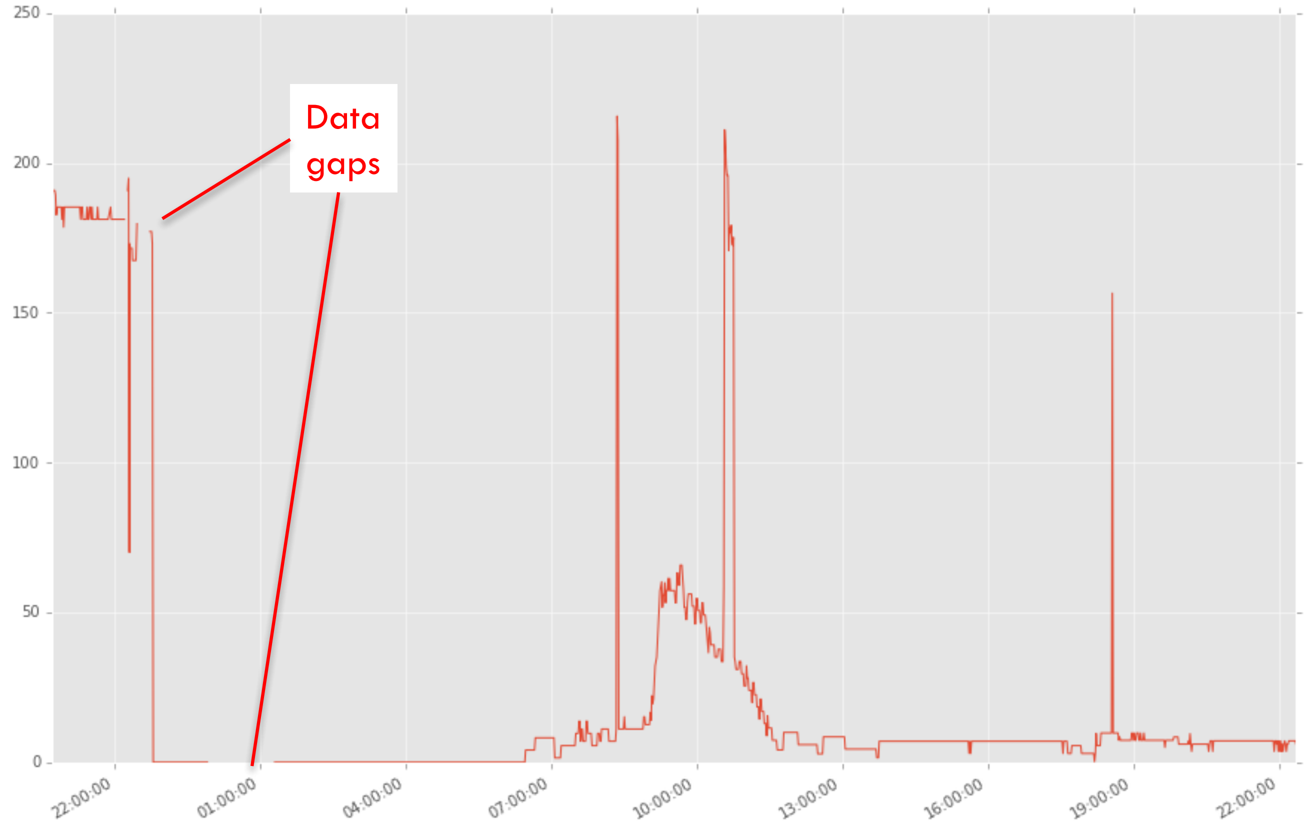
(ThingFlow running in a Jupyter Notebook)




```
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()
```

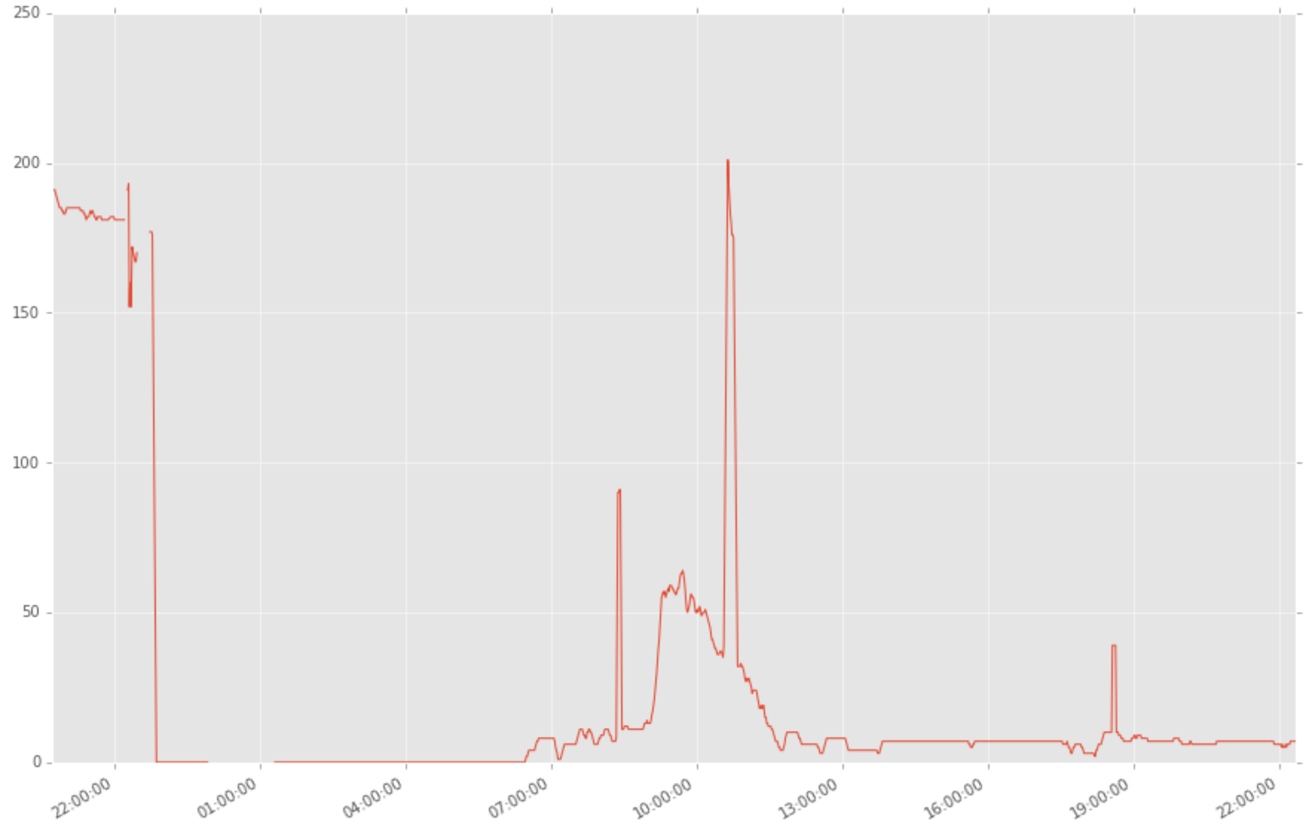
Data Processing: Raw Data

Front room, last day



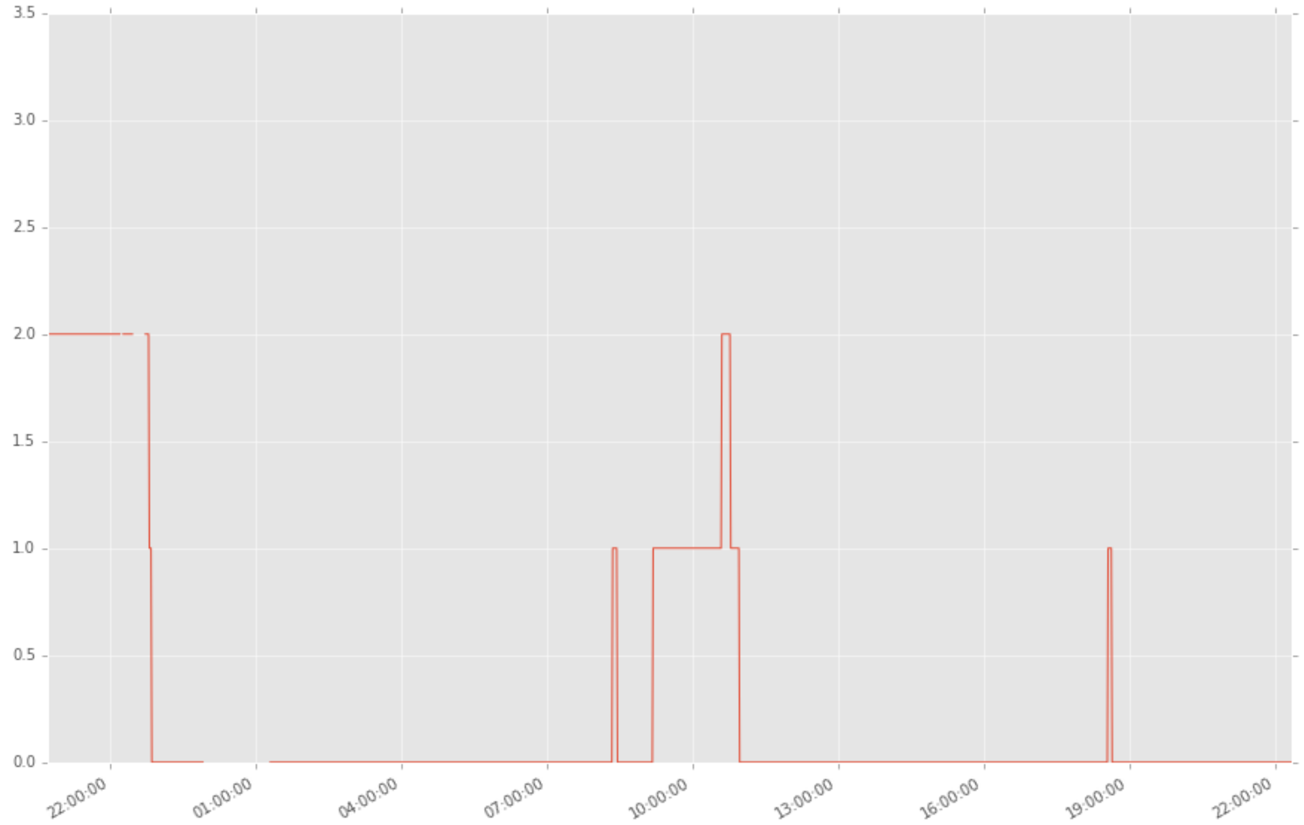
Data Processing: Smoothed Data

Front room, last day



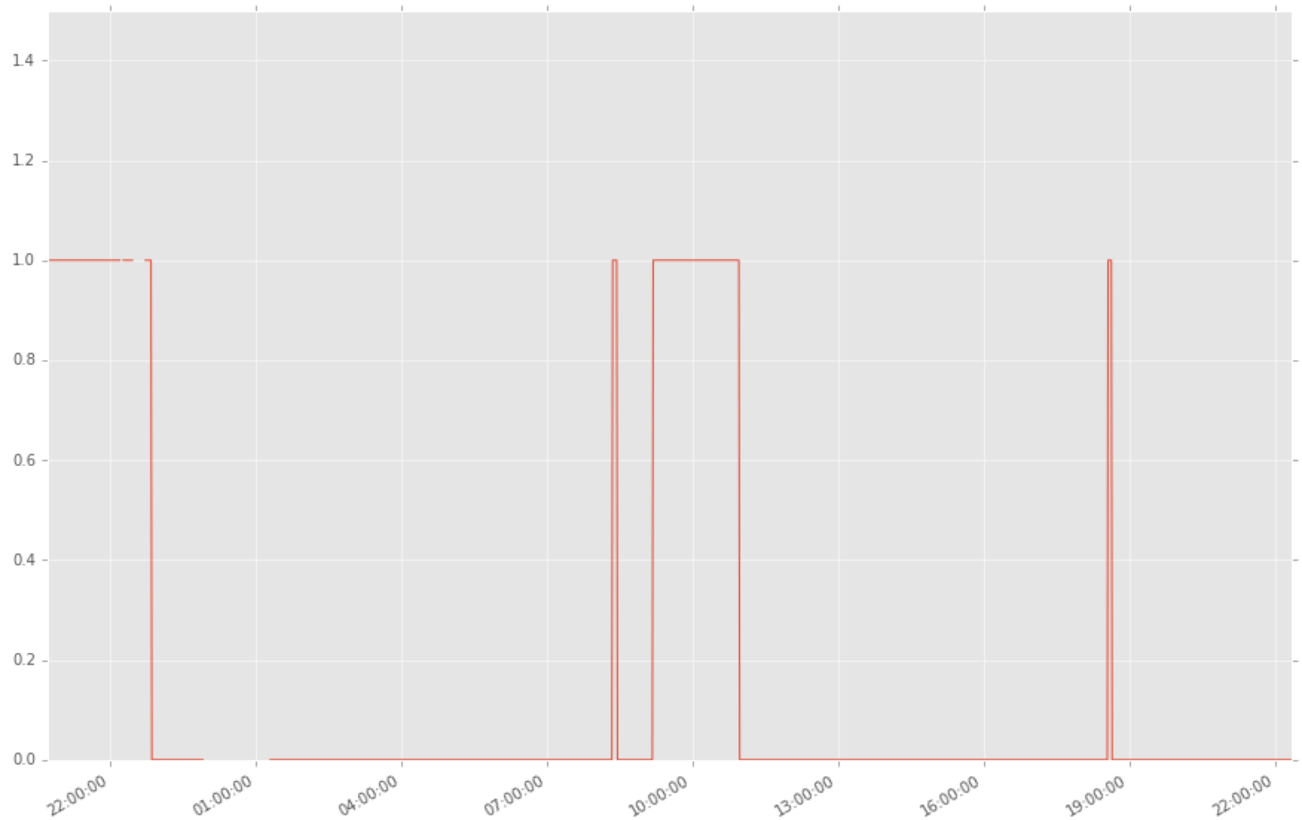
Data Processing: K-Means Clustering

Front room, last day



Data Processing: Mapping to on-off values

Front room, last day

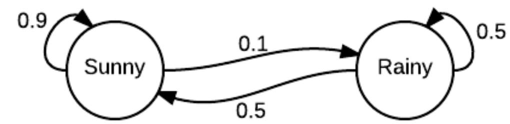


Applying "Machine Learning"

- Apply a supervised learning to create predictions for the light
  - Regression => predict light value
  - Classification => Light "on" or "off"
  - Features = time of day; time relative to sunrise, sunset; history
- Challenges
  - Transitions more important than individual samples (200 vs. 25,000)
  - Different class sizes: light is mostly off
  - Really a random process
- Solution: Hidden Markov Models

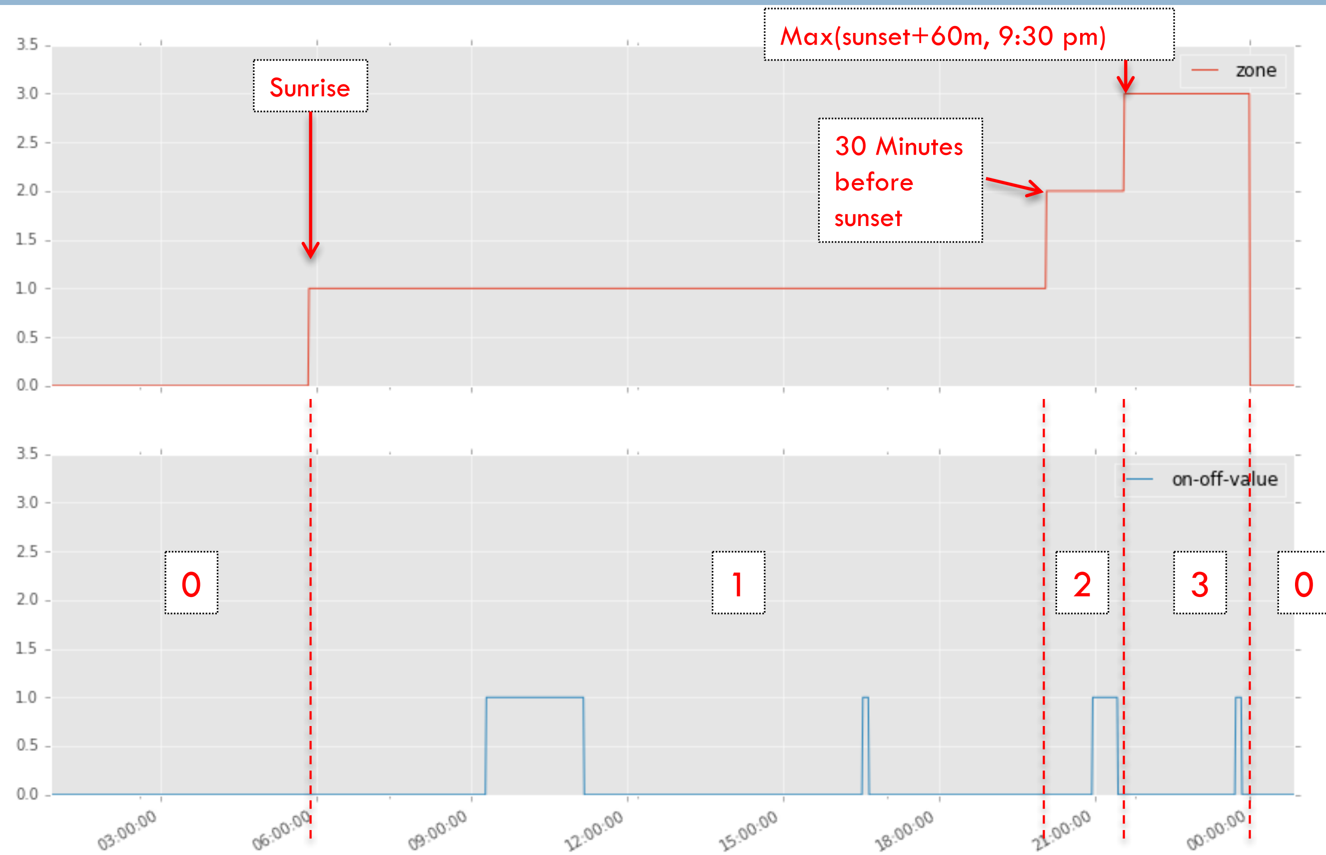
Hidden Markov Models (HMMs)

- *Markov process*
  - State machine with probability associated with each outgoing transition
  - Probabilities determined only by the current state, not on history
- Hidden Markov Model
  - The states are not visible to the observer, only the outputs ("emissions").
- In a machine learning context:
  - (Sequence of emissions, # states) => inferred HMM
- The `hmmlearn` library will do this for us.
  - https://github.com/hmmlearn/hmmlearn
- But, no way to account for time of day, etc.



Example Markov process (from Wikipedia)

Slicing Data into Time-based "Zones"



HMM Training and Prediction Process

<u>Training</u>

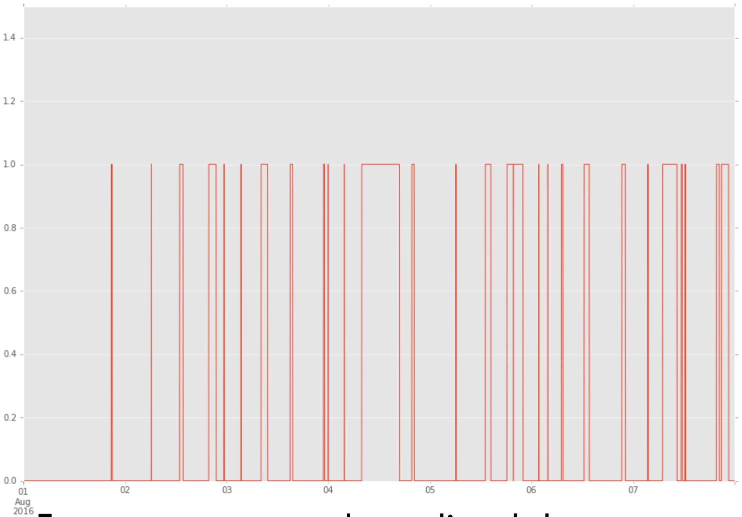
1. Build a list of sample subsequences for each zone
2. Guess a number of states (e.g. 5)
3. For each zone, create an HMM and call `fit()` with the subsequences

<u>Prediction</u>

For each zone of a given day:

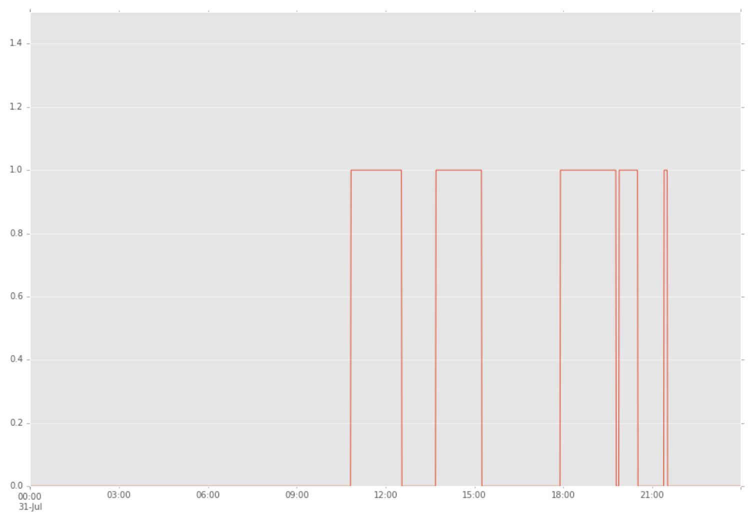
- Run the associated HMM to generate N samples for an N minute zone duration
- Associated a computed timestamp with each sample

HMM Predicted Data



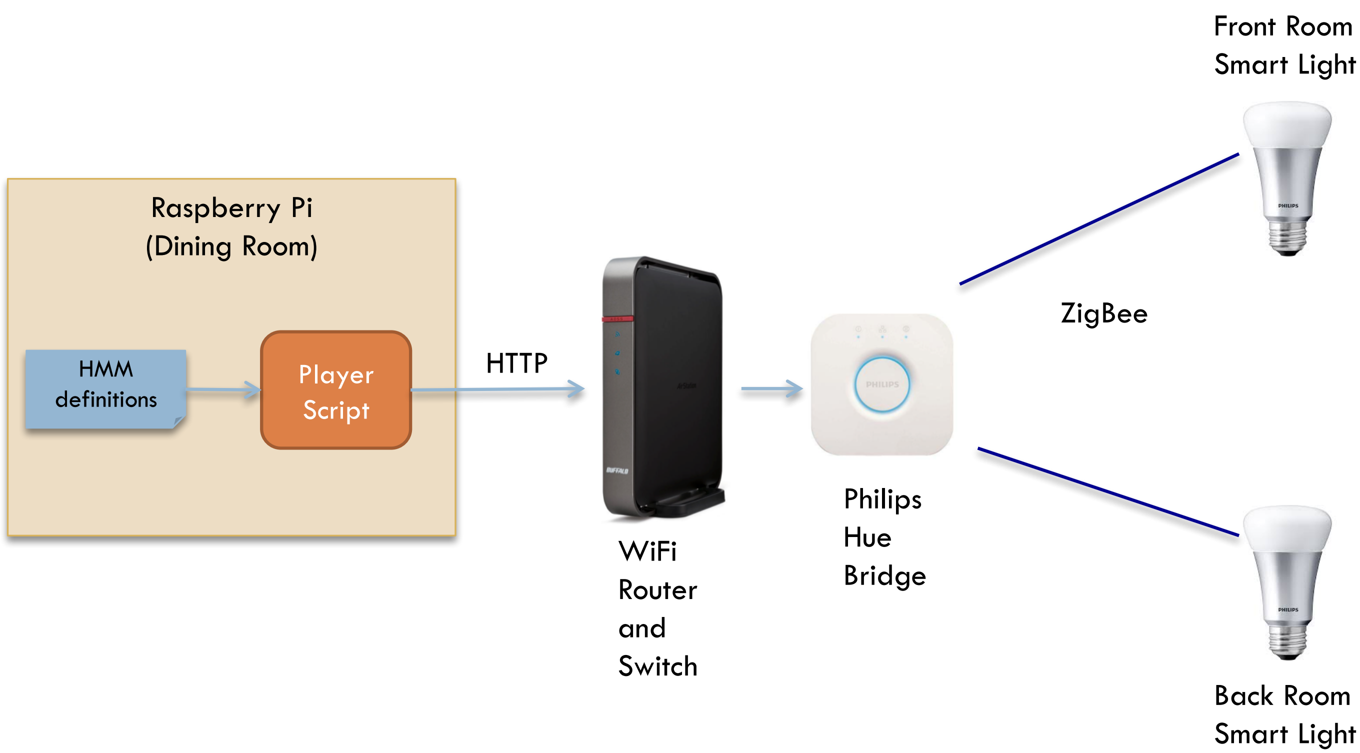
Front room, one week predicted data

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/thingflow-examples/blob/master/lighting_replay_app/player/lux_player.py

Parting Thoughts

Lessons Learned

- End-to-end projects great for learning
- Machine learning involves trial-and-error
- Visualization is key
- Python ecosystem is great for both runtime IoT and offline analytics

Thank You

Contact Me

Email: jeff@data-ken.org
Twitter: @fischer_jeff
Website and blog: https://data-ken.org

More Information

ThingFlow: https://thingflow.io

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

Hardware tutorial: http://micropython-iot-hackathon.readthedocs.io/en/latest/