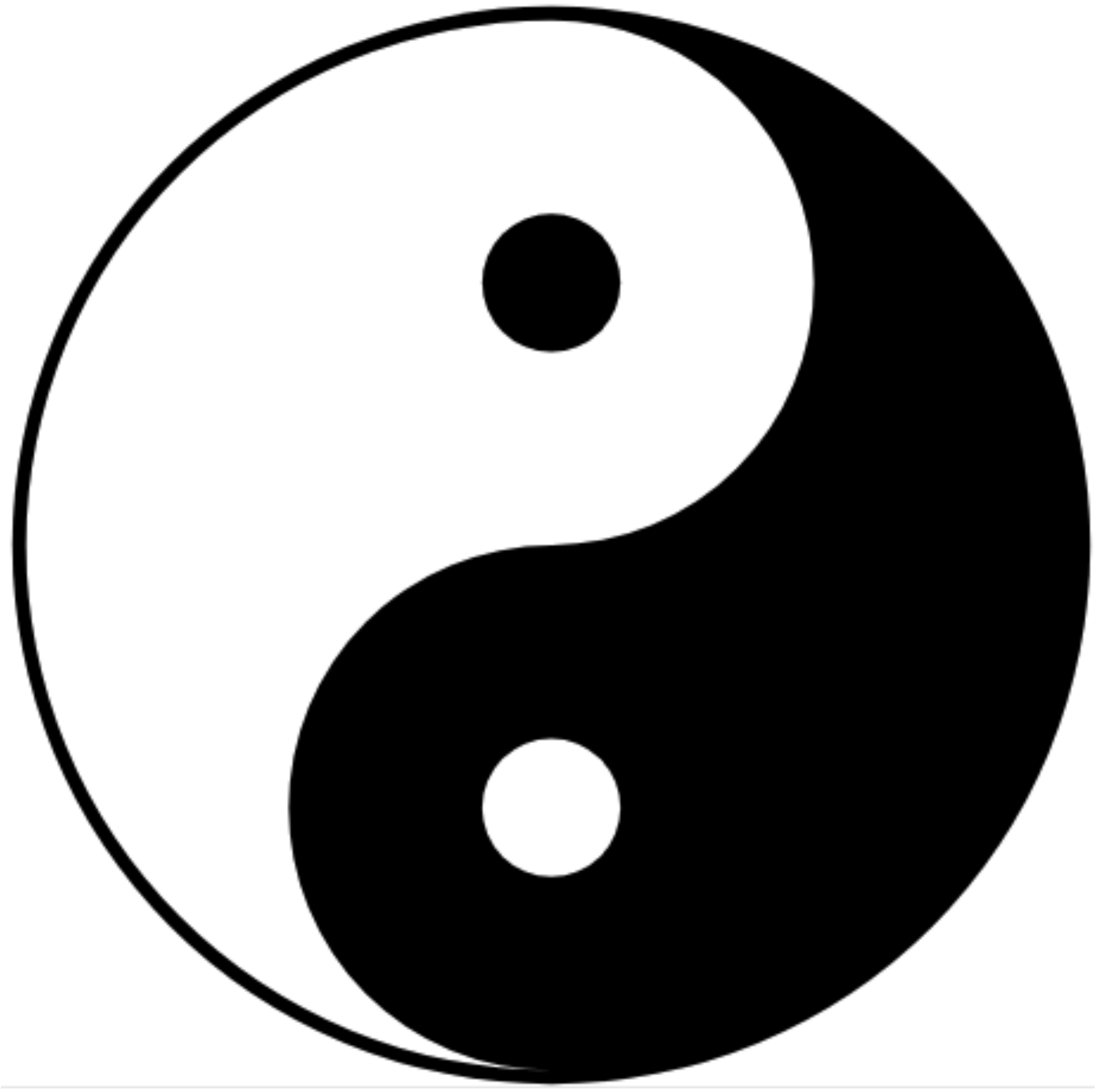
Yin Yang Ranch: A Distributed Computer Vision System with Raspberry Pi's and Macs



Jeff Bass

Building a Small Permaculture Farm in Suburbia

- Permaculture is a collection of practices to make farming more sustainable by building living soil, emulating old growth forests and recycling nutrients and water

- I am turning my 2 acre suburban lot into a small permaculture farm called Yin Yang Ranch

- It is an evolving science project and a demonstration garden; lots of shared learnings with others

- Growing a plant polyculture: figs, pears, pomegranates, plums, grapes, avocados, oranges, mulberries, blackberries…

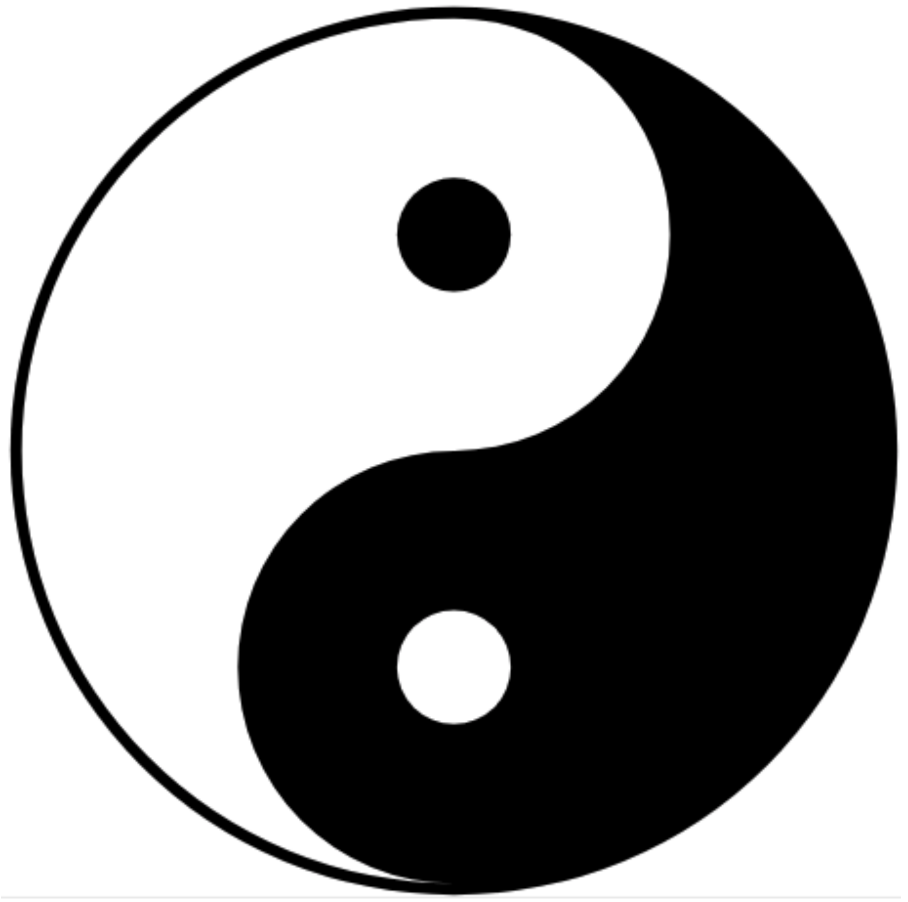
- … and a bumper crop of Raspberry Pi's.

- Using Computer Vision as a tool to help manage the farm

Jeff Bass Bio Bullets (how each ties in to my project)

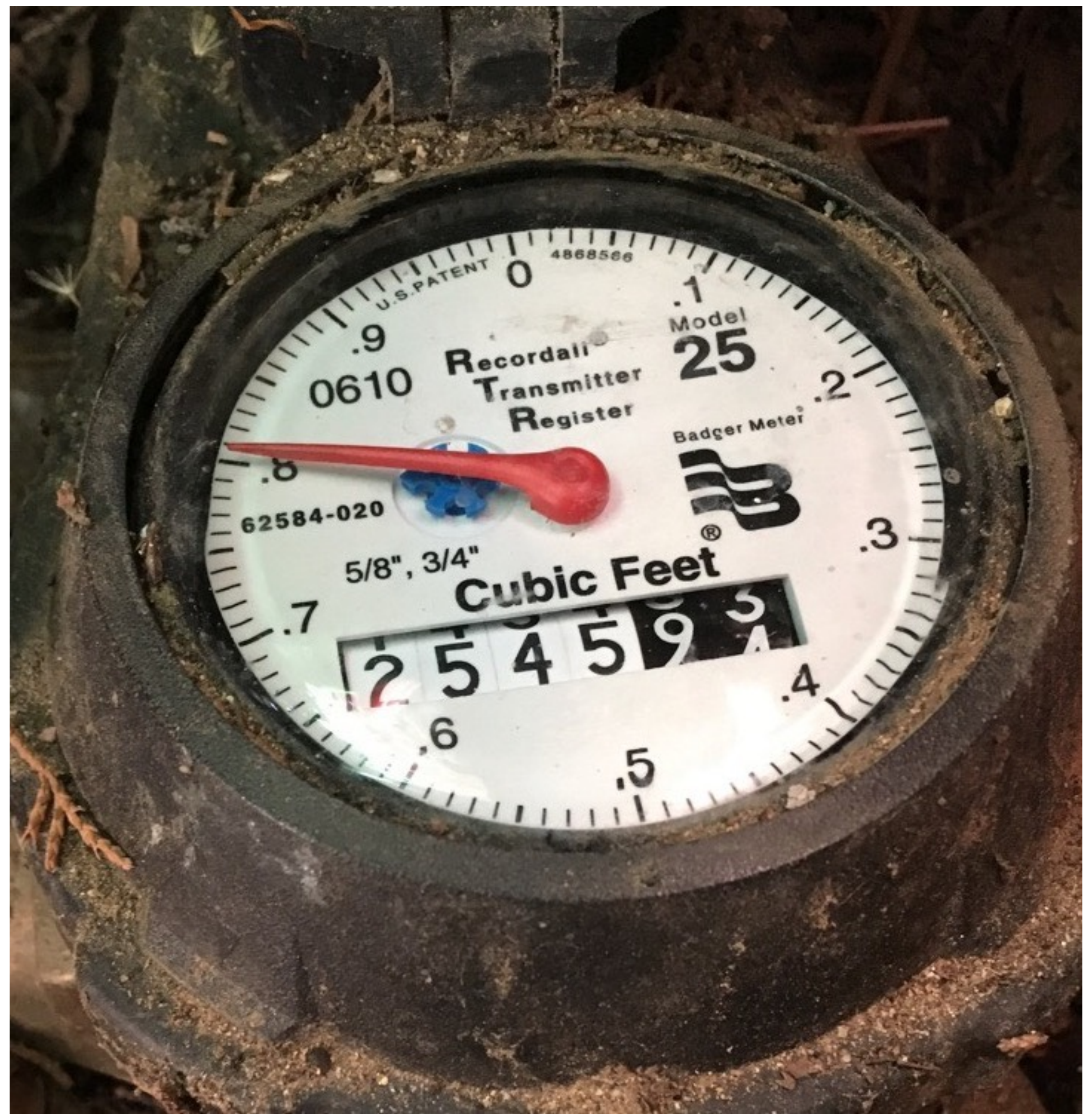
- Started college pursuing EE, got drafted, fixed radars and meteorological systems in the army (electronics)
- Went to grad school on GI bill; dissertation morphed from econometrics to computer science to a software company (CS)
- Spent 20 years doing stats / data science at big biotech; datasets from genomics to Medicare claims ("Big" data)
- Science: gene sequencing, cell receptor pathway mapping, microbiology, protein chemistry, health care economics (42)
- Small plane pilot; owned a Cessna for a while ("voice ZMQ")
- Currently retired from income producing endeavors, building the farm, bike touring the California Coast, etc. (lots of time)
- Learning about computer vision, sensors, Raspberry Pi as tools to better manage a permaculture farm (lack of focus)

What this talk will cover



- Multi-CPU computer vision pipelines
- Practical applications of CV techniques
- Project evolution and iterative design using Raspberry Pi’s (RPi’s)
- Yin and Yang of my design decisions; why “this” and “not that”
- But **NOT**:
  - Advanced computer vision or deep learning techniques
  - Deep dive into Permaculture (I can get way too didactic about it)

How Computer Vision helps manage my small permaculture farm



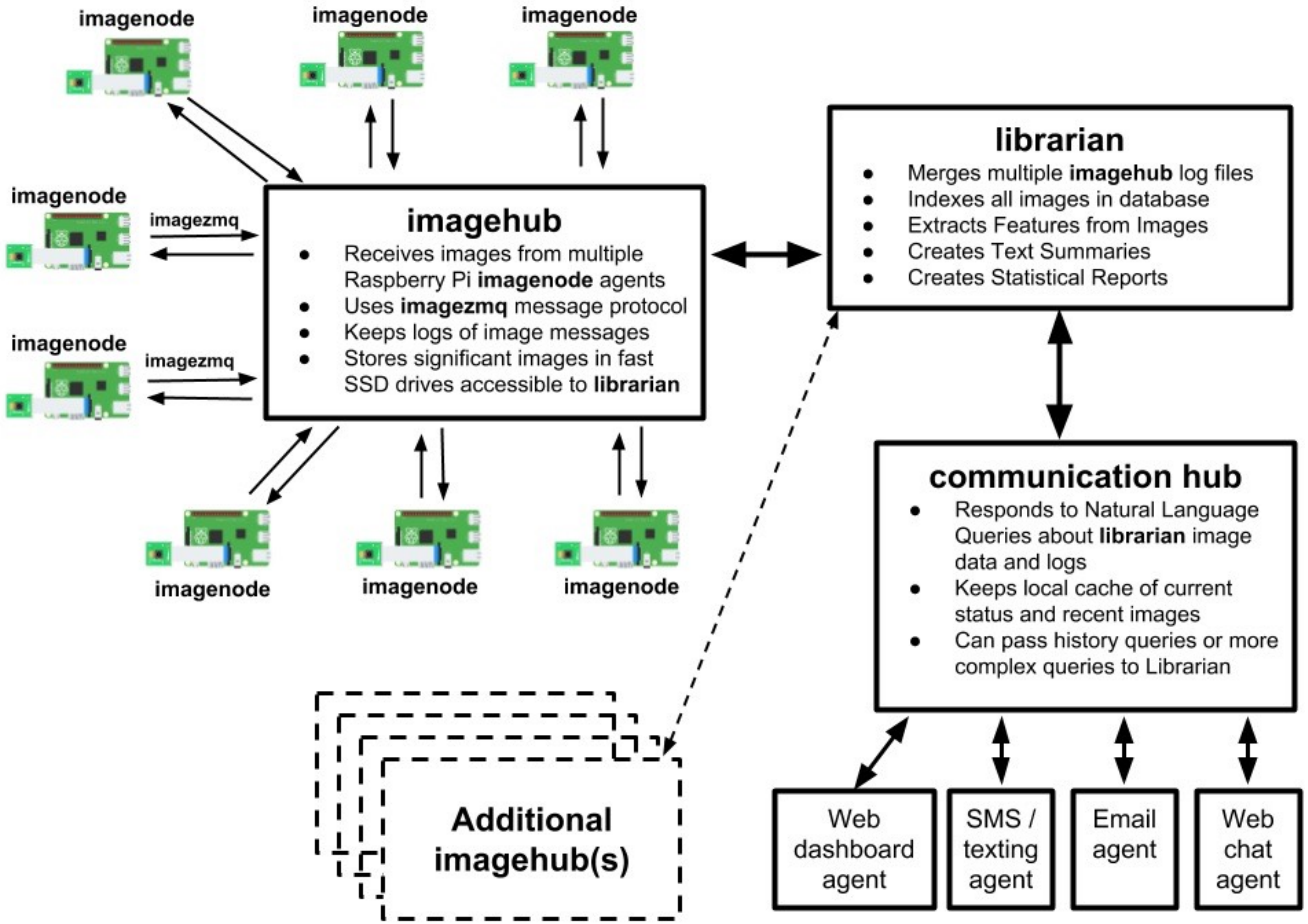
- Reading the water meter; optimizing water use
- Counting bees, butterflies and other pollinators
- Tracking coyotes, rabbits, raccoons and other critters
- Monitoring when garage and barn doors and gates open or close
- Tracking sunlight hours, sunlight intensity, clouds, shadows (photosynthesis strength and availability)
- Monitoring non-camera sensors: temperature, humidity, PIR motion sensors, solar panel power output (large 21KW array)

Interlude: What the heck is permaculture?

- Designing for long-term sustainability (centuries rather than decades)
- Creating deep living soil with diverse microbiology (emulating an old growth forest)
- Design choices to prioritize sustainability over efficiency (plant polyculture vs. Big Ag)
- Top down designs for land use, water capture and retention, carbon cycling
- Science based; careful observation; repeatable experiments; open publishing
- Open source seeds, hardware, techniques, designs for disease management
- Farms that are smaller and include native plants, insects, worms and mammals
- Permaculture is “A Revolution Disguised as Organic Gardening”

A Distributed Computer Vision Pipeline

- Raspberry Pi's as nodes
- 1 or 2 cameras per RPi
- 8 to 12 RPi's per Hub
- ZMQ Messaging passes images from RPi's to Hub
- Multiple Hubs feed the Librarian
- Communications Hub



imagenode <-> imagezmq <-> imagehub

Pseudocode for imagenode

```
ForeverEventLoop:
    grab camera image
    put image in queue
    analyze queue
    if SomethingHappened:
        send event message
        send images
        process hub response
```

Pseudocode for imagehub

```
ForeverEventLoop:
    receive (message, image)
    store event message
    store images
    send response
```

Simple Code — Multiple RPi's to a Single Mac

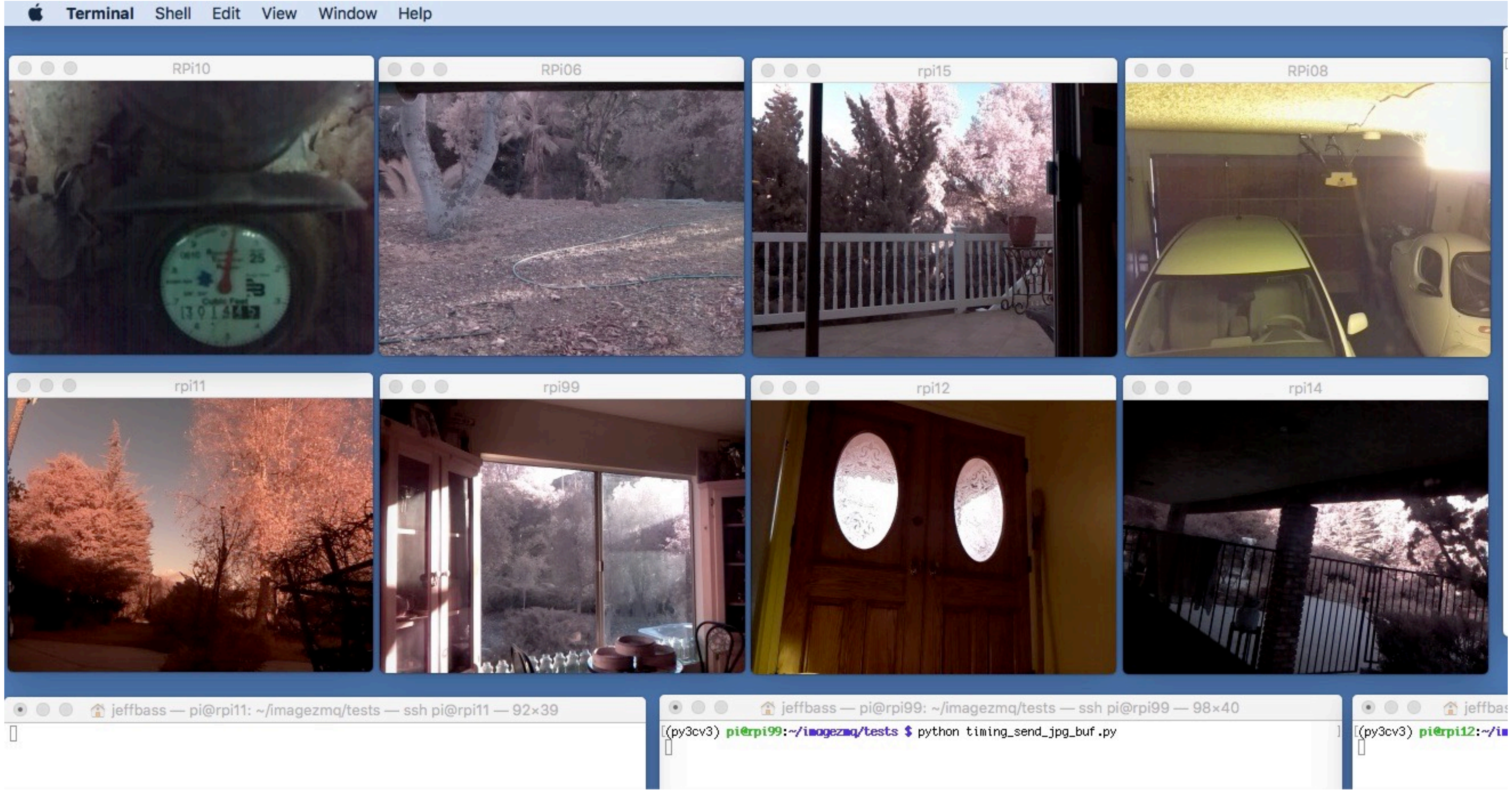
```python
# run this program on each RPi to send a labelled image stream
import socket
import time
from imutils.video import VideoStream
import imagezmq

sender = imagezmq.ImageSender(connect_to='tcp://jeff-macbook:5555')

rpi_name = socket.gethostname() # send RPi hostname with each image
picam = VideoStream(usePiCamera=True).start()
time.sleep(2.0)  # allow camera sensor to warm up
while True:  # send images as stream until Ctrl-C
    image = picam.read()
    sender.send_image(rpi_name, image)
```

Node Code (RPi)

```python
# run this program on the Mac to display image streams from multiple RPis
import cv2
import imagezmq
image_hub = imagezmq.ImageHub()
while True:  # show streamed images until Ctrl-C
    rpi_name, image = image_hub.recv_image()
    cv2.imshow(rpi_name, image) # 1 window for each RPi
    cv2.waitKey(1)
    image_hub.send_reply(b'OK')
```

Hub Code (Mac)

Simple RPi -> Mac Pipeline: Video Surveillance



An 8 RPi Simulcast in 19 lines of Python (11 per RPi; 8 on a Mac)

Imagenode Code Snippet

```
# imagenode.py snippet
#     (all the imports, logging setup hidden to fit slide)

settings = Settings()  # get settings for node cameras, ROIs, GPIO
node = ImageNode(settings)  # start ZMQ, cameras and other sensors

# forever event loop
while True:
    node.get_sensor_data()  # grab camera images and other sensor data
    node.process_sensor_data()  # detect motion, etc.
    while len(node.msg_q) > 0:
        try:
            with Patience(settings.patience):  # wait for hub response
                text, image = node.msg_q.popleft()
                hub_reply = node.send_frame(text, image)
        except Patience.Timeout:  # except no timely response from hub
            hub_reply = node.fix_comm_link()
        node.process_hub_reply(hub_reply)
#  ……
```

Example: Reading a Water Meter

What the RPi does:

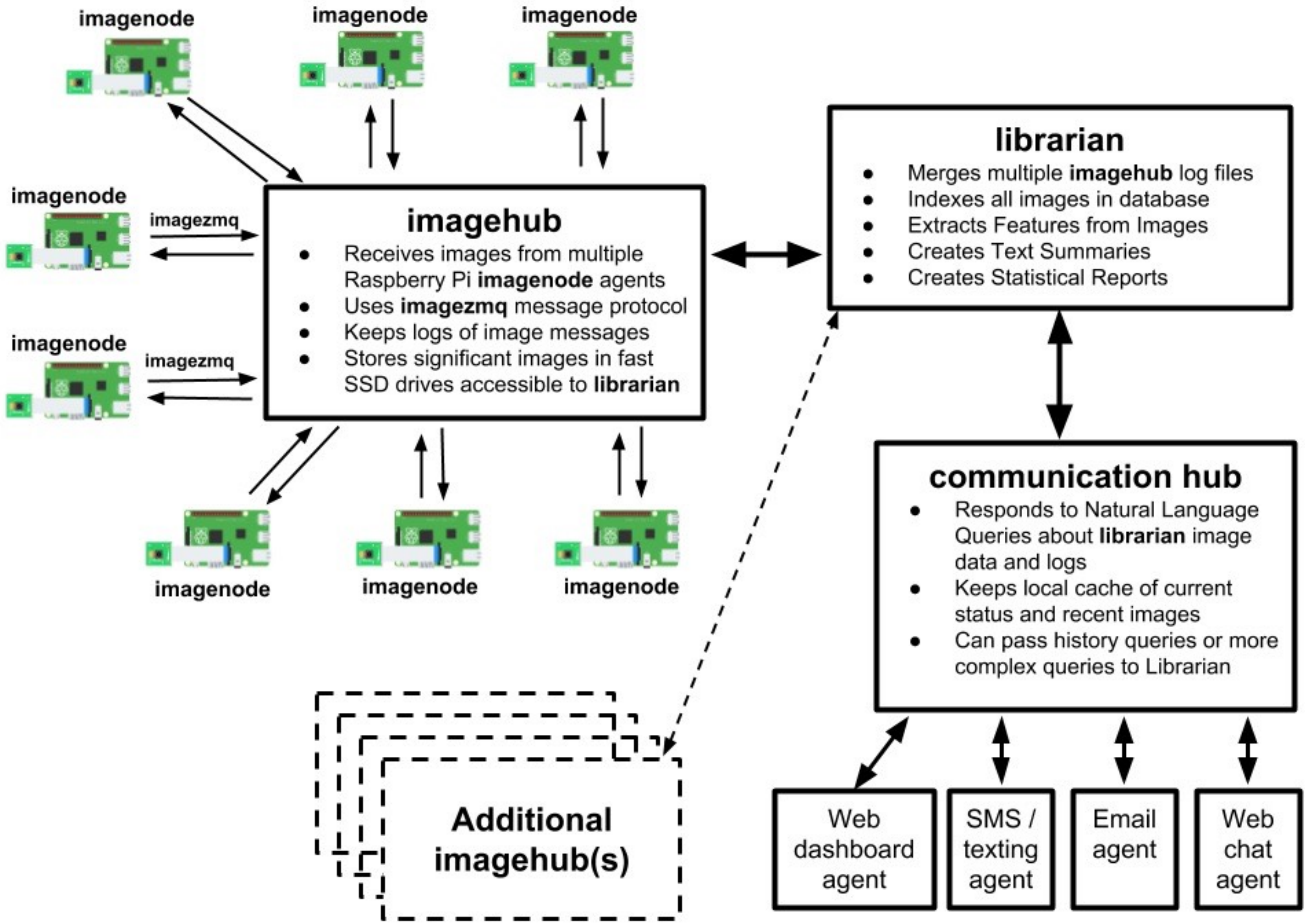
- Turns on LEDs to light the meter
- Continuously grabs frames
- Does ROI cropping, conversion to grayscale, thresholding, dilation
- Detects needle & leak spinner motion
  - Sends event msg (“water flowing”)
  - Sends images of event
- Xfers images & messages via ZMQ

What the Mac does:

- Acts as ZMQ hub for multiple RPi’s
- Receives and acknowledges images and messages from multiple RPi’s
- Adds events messages to events log
- Stores and indexes images
- Extracts digits to read the meter
- Keeps history of events; does stats
- Responds to queries about status

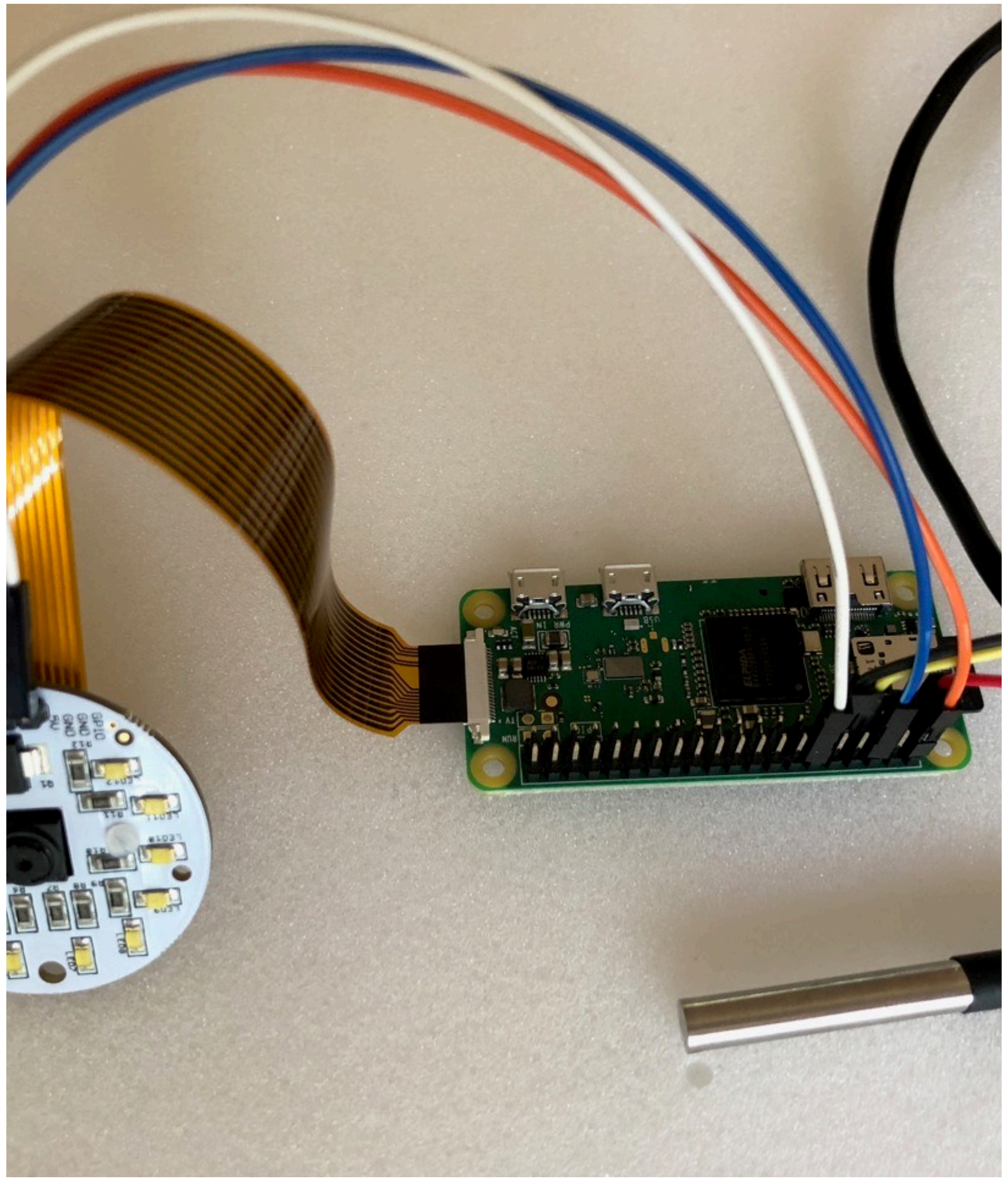
Distributed Computer Vision Pipeline (revisited)

- Raspberry Pi's as nodes
- 1 or 2 cameras per RPi
- 8 to 12 RPi's per Hub
- ZMQ Messaging passes images from RPi's to Hub
- Multiple Hubs feed the Librarian
- Communications Hub

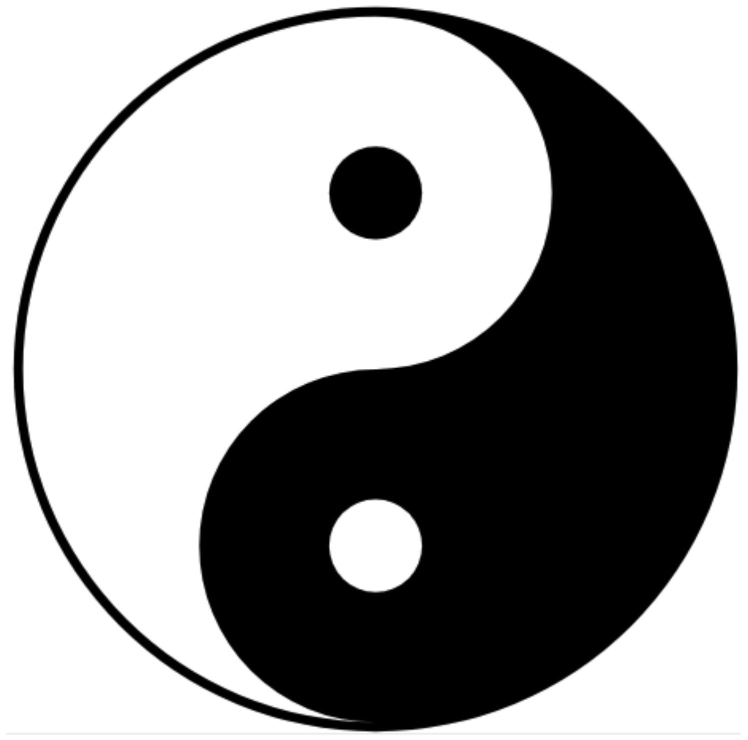


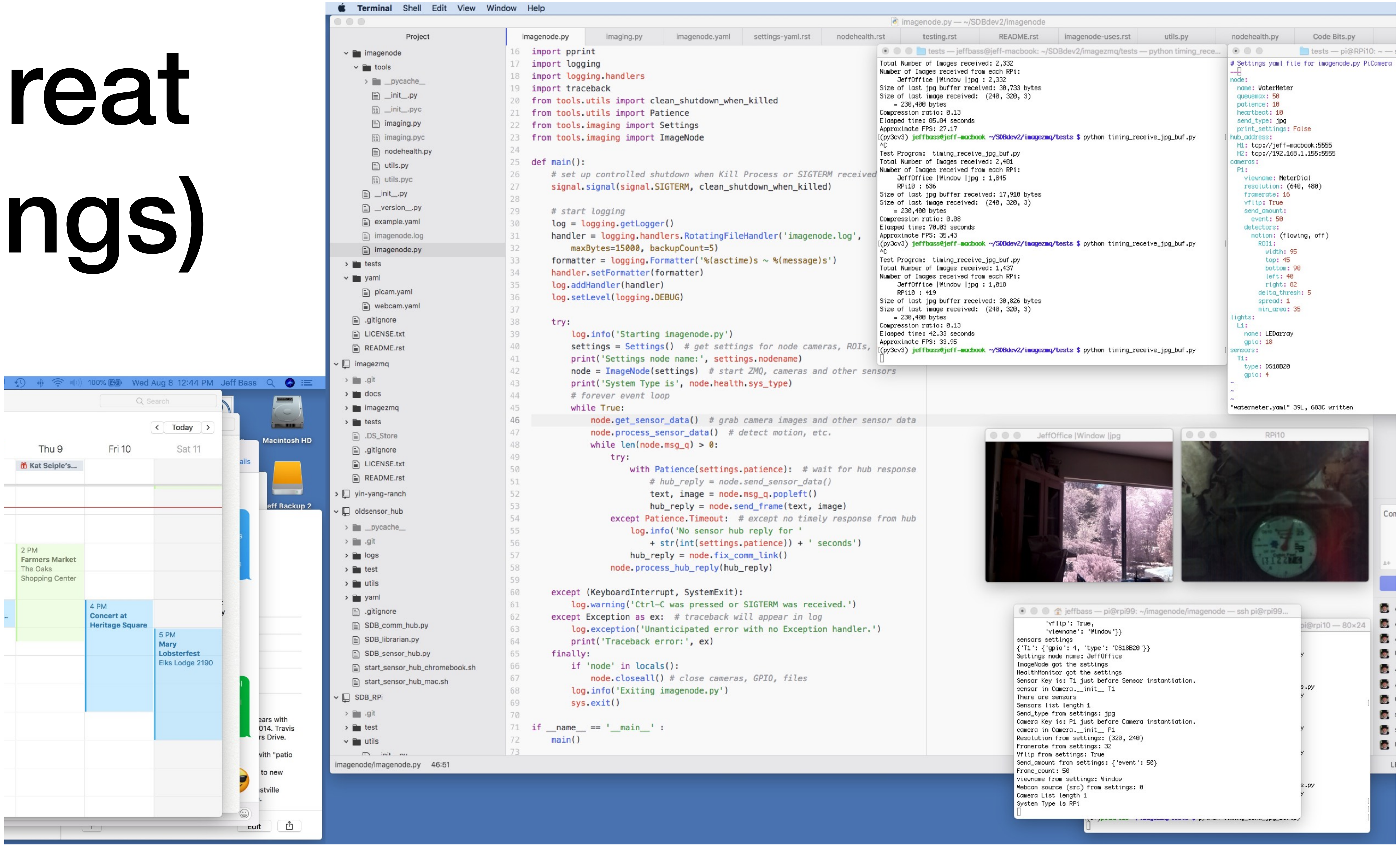
Raspberry Pi's are Great (at some things)

- Cheap, reliable, **fanless**, dust tolerant
- Physically small (Pi Zero) for sensor enclosures
- Very low power; quick restart after power failures
- PiCamera has helpful adjustable settings
- **BUT…**
  - SD Cards not reliable for writing large image files
  - SD Card writing is VERY slow
  - Limited memory and limited processing capacity
  - Slower USB bus: slow Ethernet; slow WiFi
  - USB webcams on RPi are slow and cranky



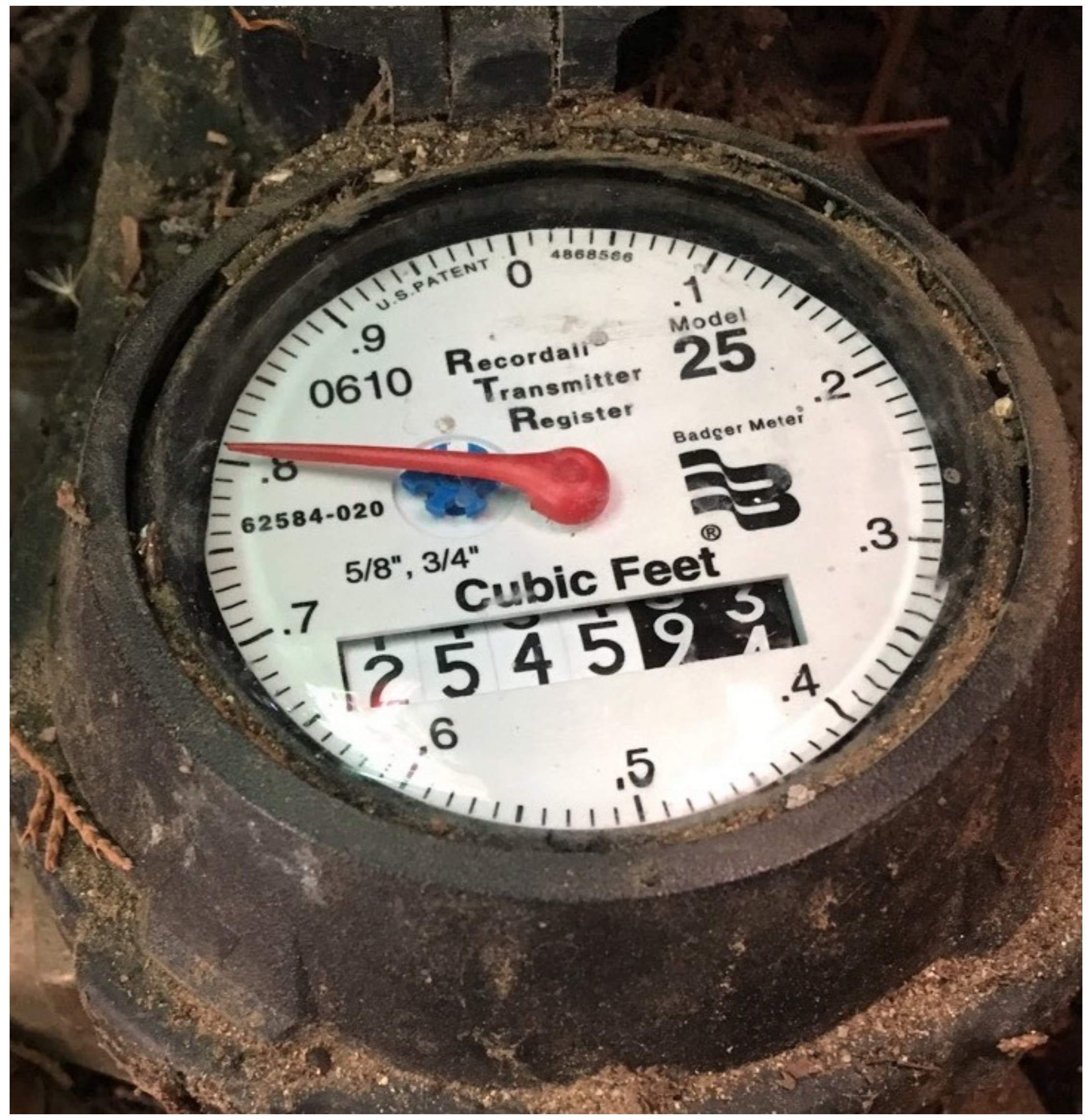
Macs are Great (at some things)





- Fast WiFi & Ethernet
- Fast SSD storage
  (for lots of images)
- More computing power for elaborate processing of images
- Digits recognition; feature extraction; classification
- **But…**one Mac laptop uses as much power as 8 or 10 RPis
- Macs are expensive and not suitable for outdoors

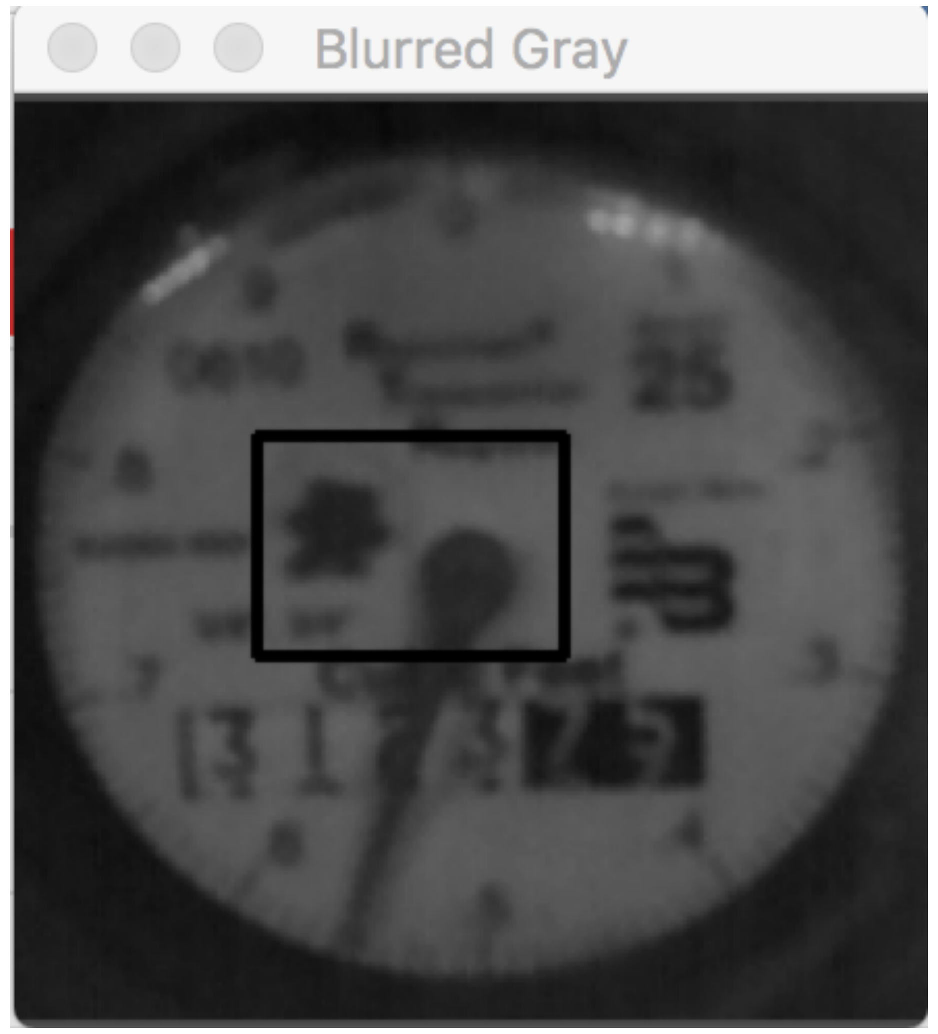
Image Pipeline for Reading a Water Meter



Daylight view with the water meter cover off

Full Frame from RPi PiCamera. Lit with 24 low power LEDs to minimize reflections and glare

(Gotcha: rotating needle can occlude digits)



RPi does frame to frame differencing to detect motion of spinning bits in the rectangle

Mac does digit masking, digit classification to read the value of the digits

There are many, many CV settings…

Need to iteratively optimize all settings

- Settings for:
  - Node name, queue sizes, wait times
  - Hub addresses for multiple hubs
  - Cameras
    - Viewname, resolution, framerate, send quantity
    - Detectors (motion, light, color, etc)
      - ROI dimensions and corners
      - Detection parameters (delta_threshold, area)
  - Lights (under GPIO control)
  - Sensors (temperature, humidity, PIR)
- Using YAML files for settings
  - Feels more Pythonic than json or configs
  - Large settings file becomes a nested Python dict

tests — pi@RPi10: ~ — ssh pi@rpi10 — 66×42


```yaml
# Settings yaml file for imagenode.py PiCamera
---
node:
  name: WaterMeter
  queuemax: 50
  patience: 10
  heartbeat: 10
  send_type: jpg
  print_settings: False
hub_address:
  H1: tcp://jeff-macbook:5555
  H2: tcp://192.168.1.155:5555
cameras:
  P1:
    viewname: MeterDial
    resolution: (640, 480)
    framerate: 16
    vflip: True
    send_amount:
      event: 50
    detectors:
      motion: (flowing, off)
        ROI1:
          width: 95
          top: 45
          bottom: 90
          left: 40
          right: 82
        delta_thresh: 5
        spread: 1
        min_area: 35
lights:
  L1:
    name: LEDarray
    gpio: 18
sensors:
  T1:
    type: DS18B20
    gpio: 4
~
~
"watermeter.yaml" 39L, 683C written            39,1          All
```

Communications: ZMQ

Distributed Messaging

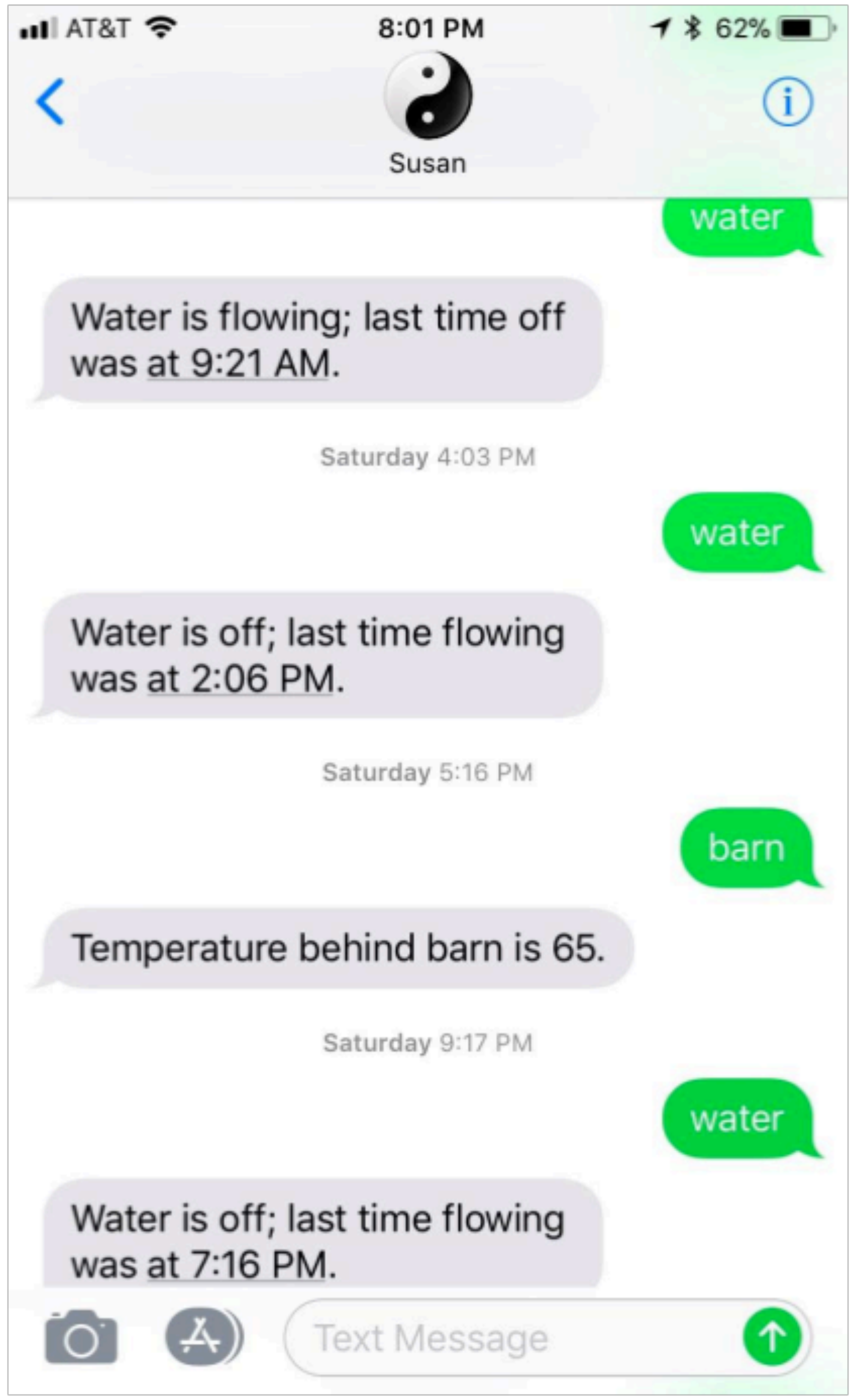
- Fast, memory efficient
- No central server or broker process needed
- ZMQ makes a great concurrency manager for multiple RPis (using synchronous REQ / REP)
- Communications protocol is flexible, but messaging protocol must be well designed
- Alternatives considered: RabbitMQ, ROS, AMQP, MQTT, others

Communications: Some Learnings

- Modeled on FAA protocol between ATC and pilots (pilot geekiness)
  - Who you are, where you are, what you want; predictable pattern
  - No unnecessary chatter; keep channel clear
  - Backup protocol if ATC is unavailable
- Use multiple communications channels
  - Transfer via ZMQ for some things: images and messages from RPis
  - Use scp & rsync for others: moving images around; backing things up
  - Dropbox and Google Drive are helpful in multiple ways

Communicating to End Users (me ;-)

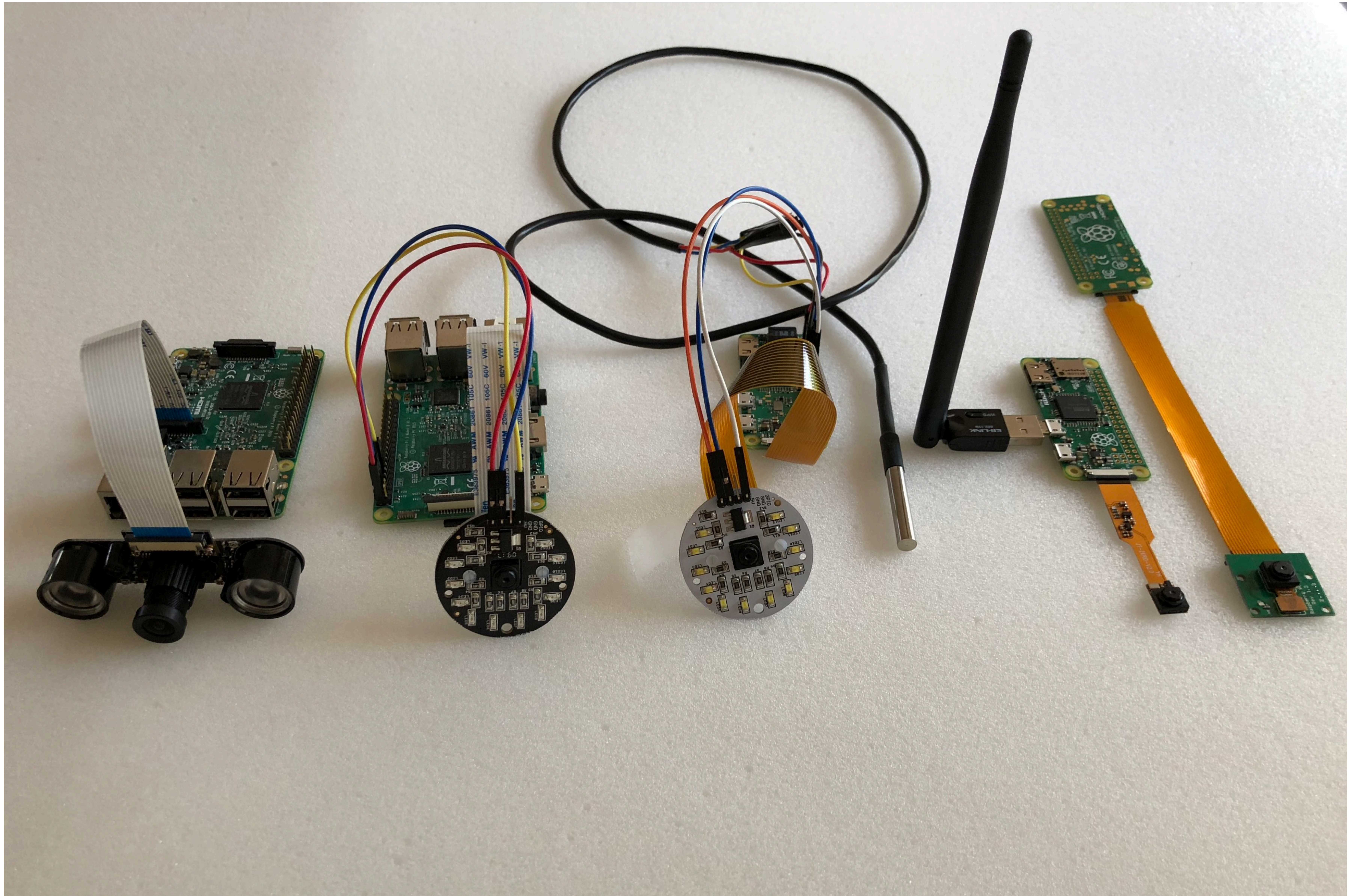
- Communicating “conclusions" from computer vision observations:
  - “Water is off; last time flowing at 7:15am”
  - “Garage is closed; last open at 8:20pm yesterday”
  - “Animal spotted at 3:15am; coyote (77%)”
- Communicating via multiple channels
  - Simple CLI text chat interface for testing
  - SMS texting interface has proved most useful so far
  - Website “dashboard” for images and data charts



Learnings: Hardware Tips and Tricks

- PiCams vs. WebCams: setting exposure mode, iso, contrast, brightness and saturation adjustment
- Enclosures, e.g. water meter cam, critter cam: mason jars and old shingles are my DIY favorites
- Cost of different cams: the variation is large; the variation in quality is even larger
- InfraRed lighting for critters: multiple choices for size and on/off control, including PIR sensors
- LED lighting under RPi control: MOSFETs rule!
- Temperature / humidity sensors: DS18B20 & W1ThermSensor; DHT &
- Networking via power-line modem ethernet as well as WiFI and wired ethernet
- Laptops keep working during power failures; RPi’s don’t; design considerations
- Power: advantages of 12v over 5v: cheap reliable converters (yes, those car chargers for your phone)

Hardware: Lights, Camera, Action!



Left to Right:

RPi with Waveshare Combo IR lens and dual IR floodlights

PiNoir IR Camera with IR “Ring Light” floodlight

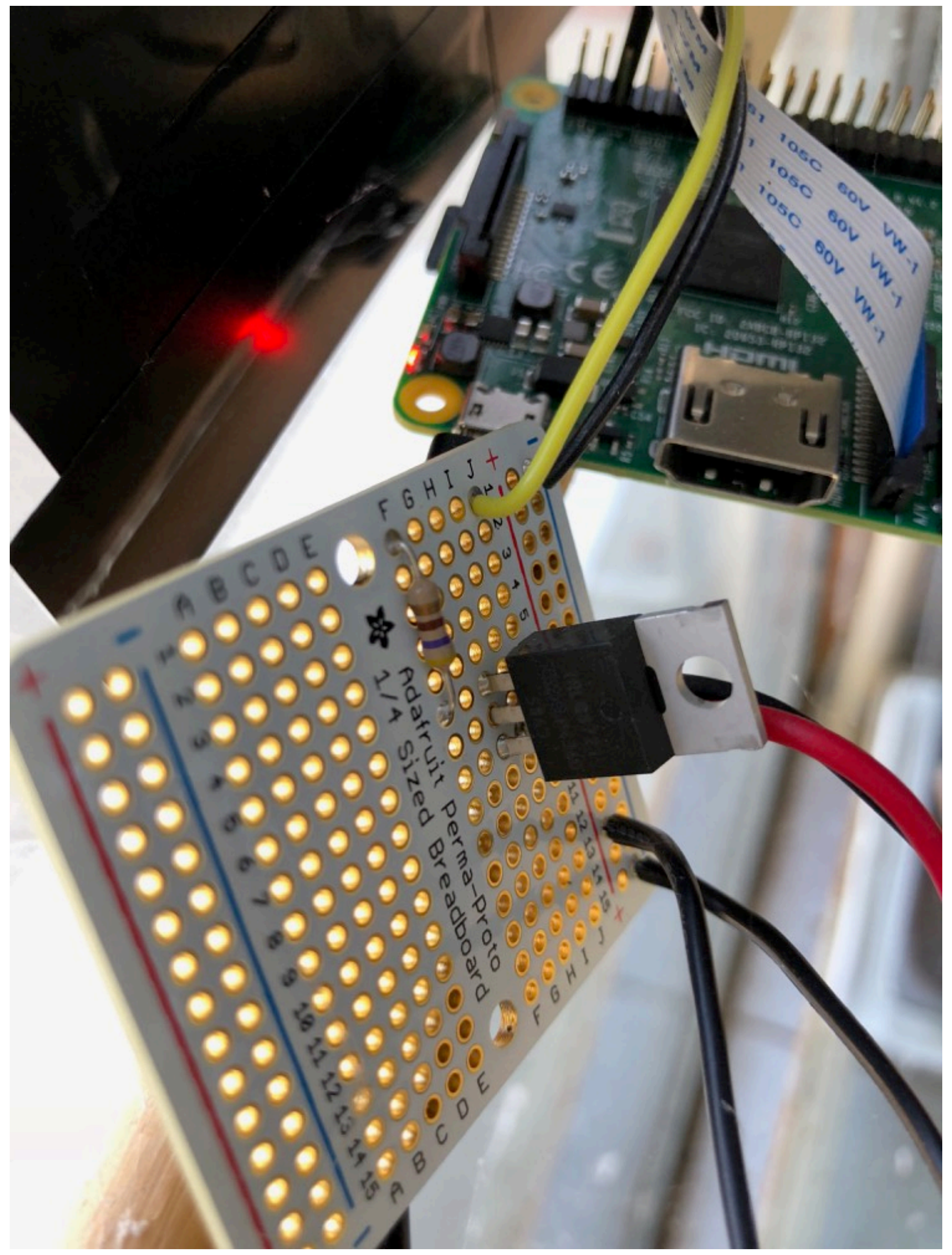
RPi Zero with PiCamera in white light “Ring Light” with DS18B20 temperature probe

RPi Zero with “Spy Cam” and longer range WiFi

RPi Zero with older model (and half price) PiCamera

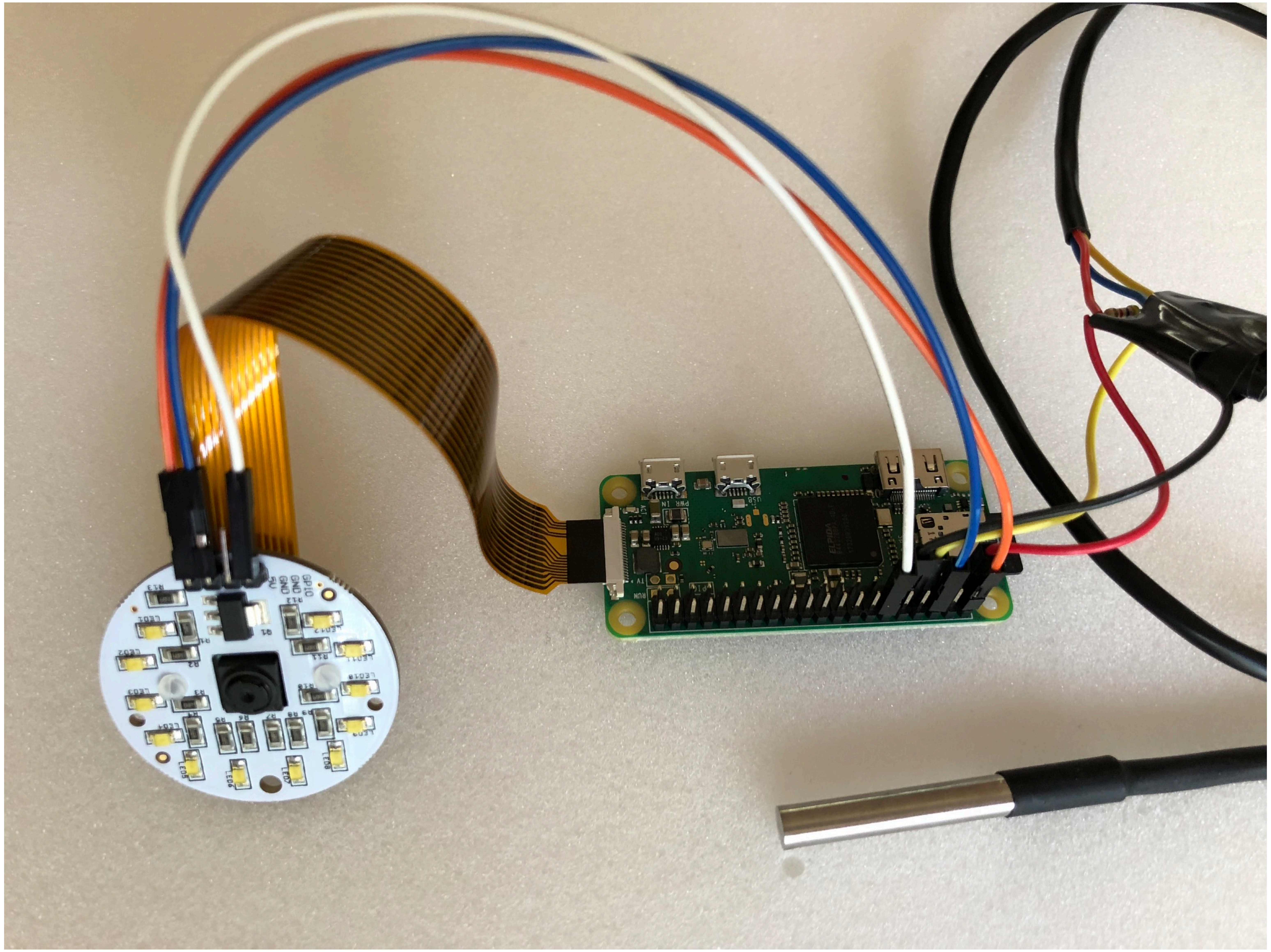
Hardware: More Light Types

White floodlight (18W), 12V power cube, Infrared floodlight (8W)



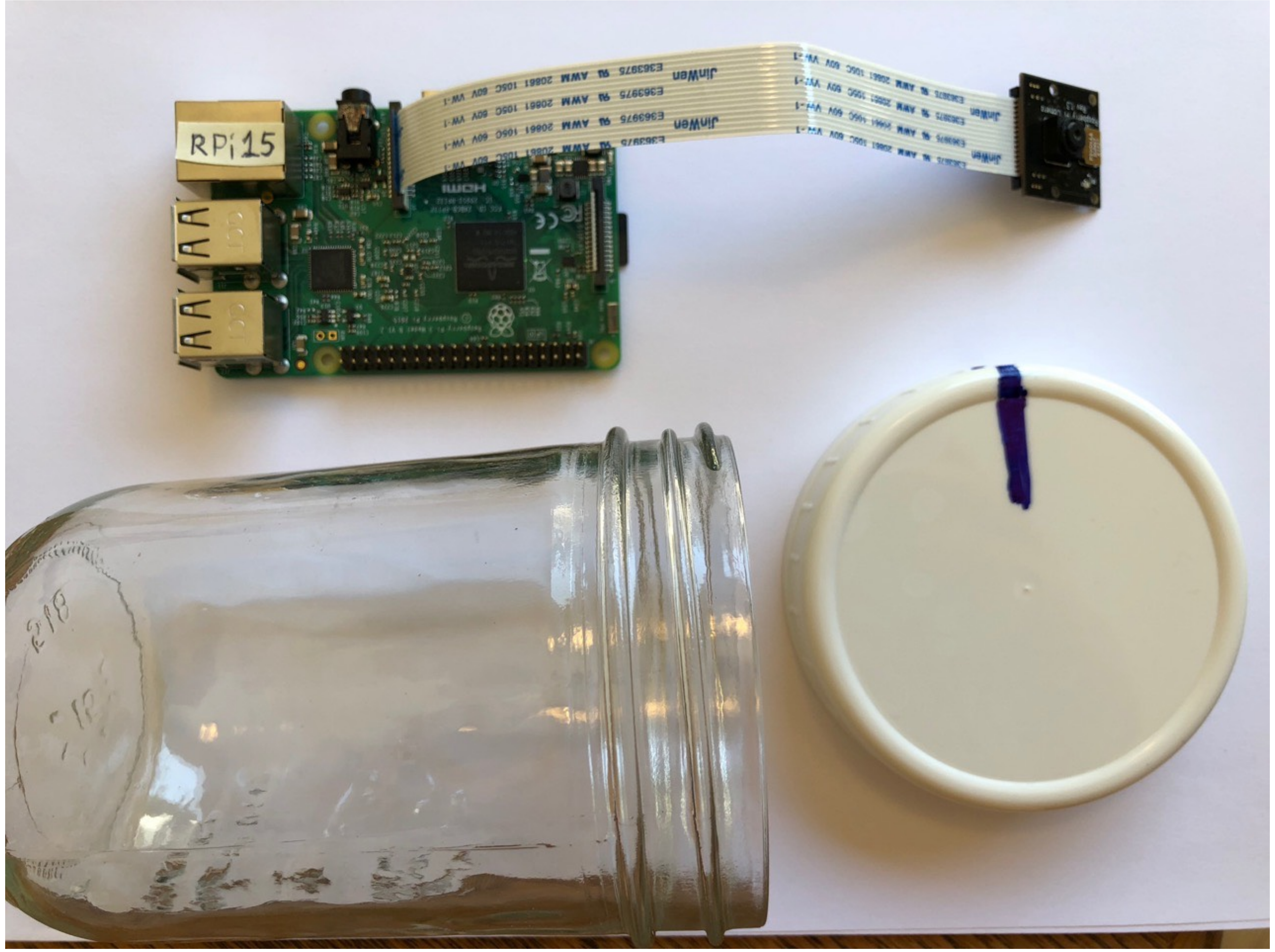
Simple MOSFET circuit provides RPi GPIO control for floodlights

Hardware Enclosures: Light Fixtures!

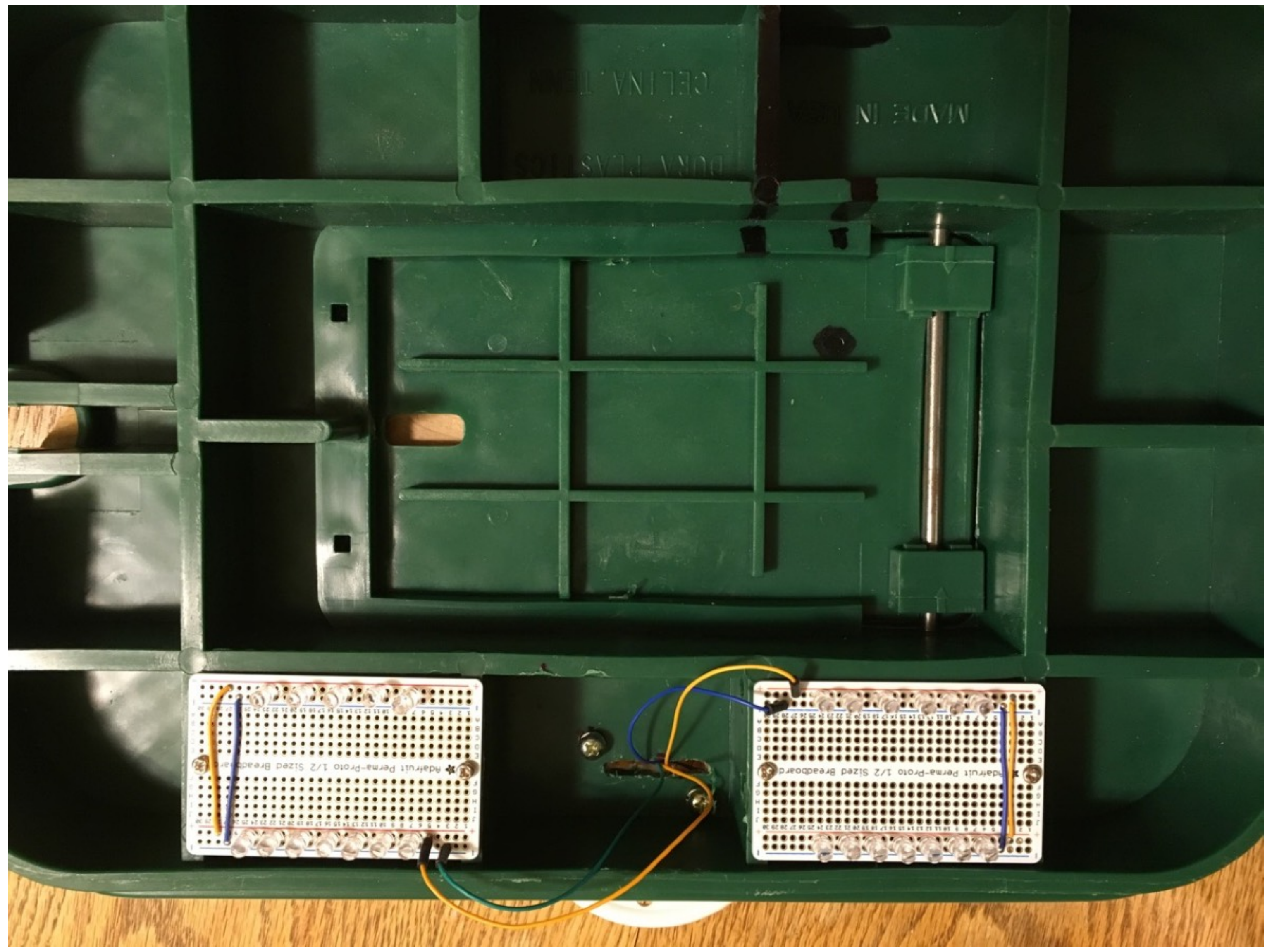


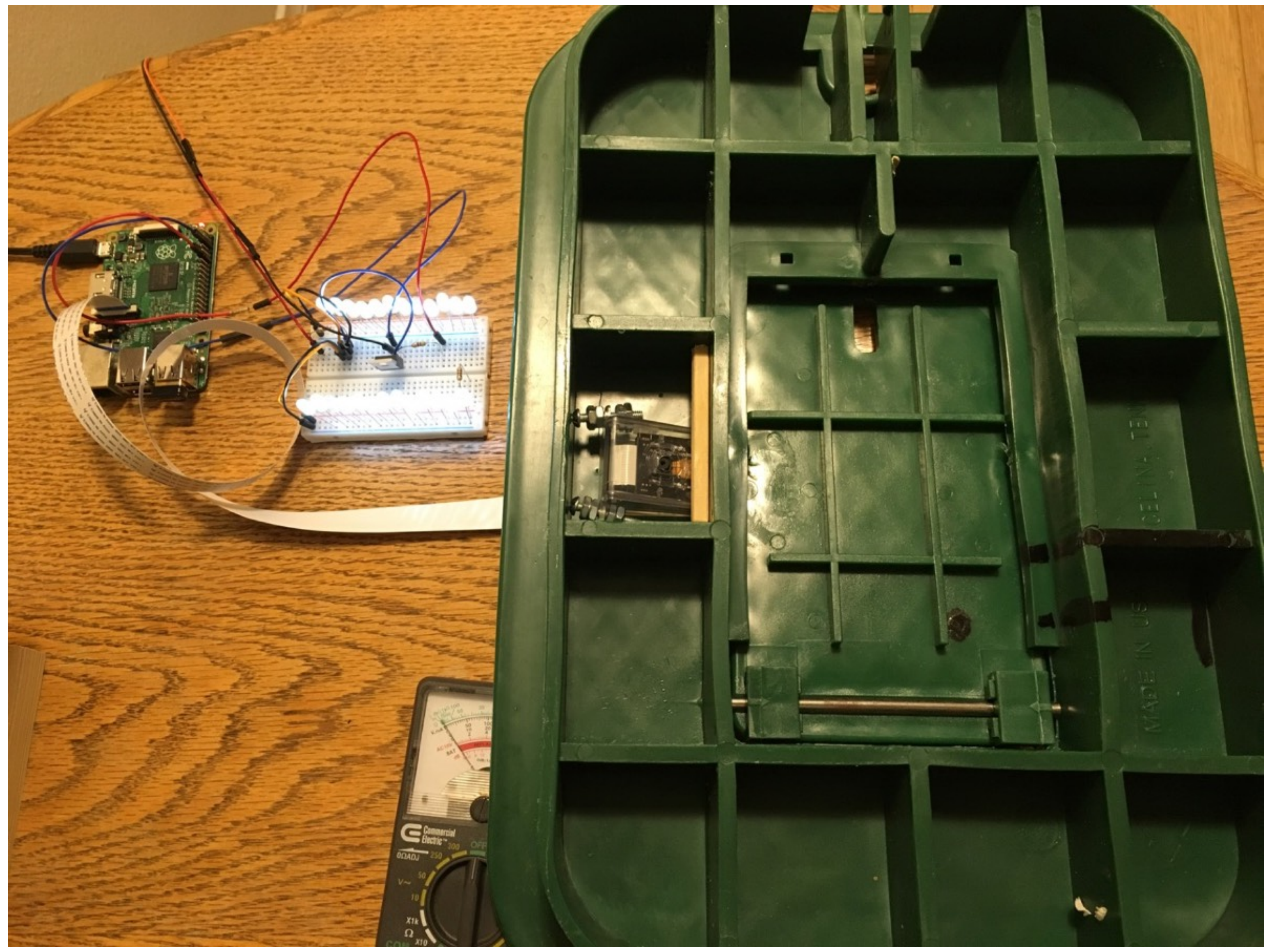
… and temperature sensors!

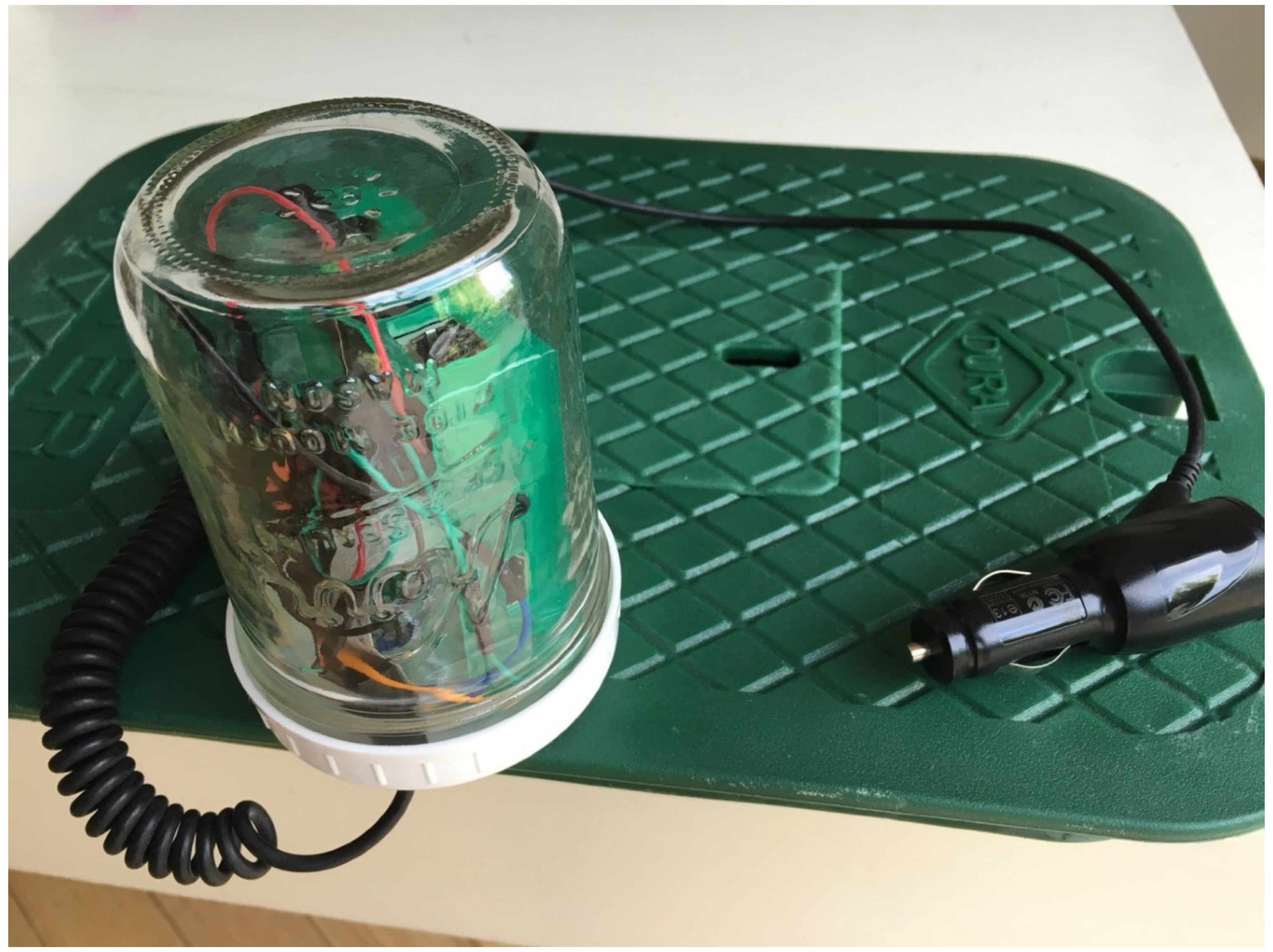
Hardware Enclosures: Mason Jars!



Hardware Enclosures: Water Meter Cam







Hardware Enclosures: Barn Coyote Cam

Glass blocks Infrared, so Mason jars can't be used for Infrared Cameras. But a PiNoir camera does fine outdoors under a couple of old cedar shingles.

Interlude: Coyote Cameos with Rabbit

Taken with RPi “PiNoir” Infrared camera, using an 8W Infrared floodlight

Hardware Enclosures: Fake Security Cam for $5

This $5 fake security cam can be modified to hold a RPi Zero with PiCamera.

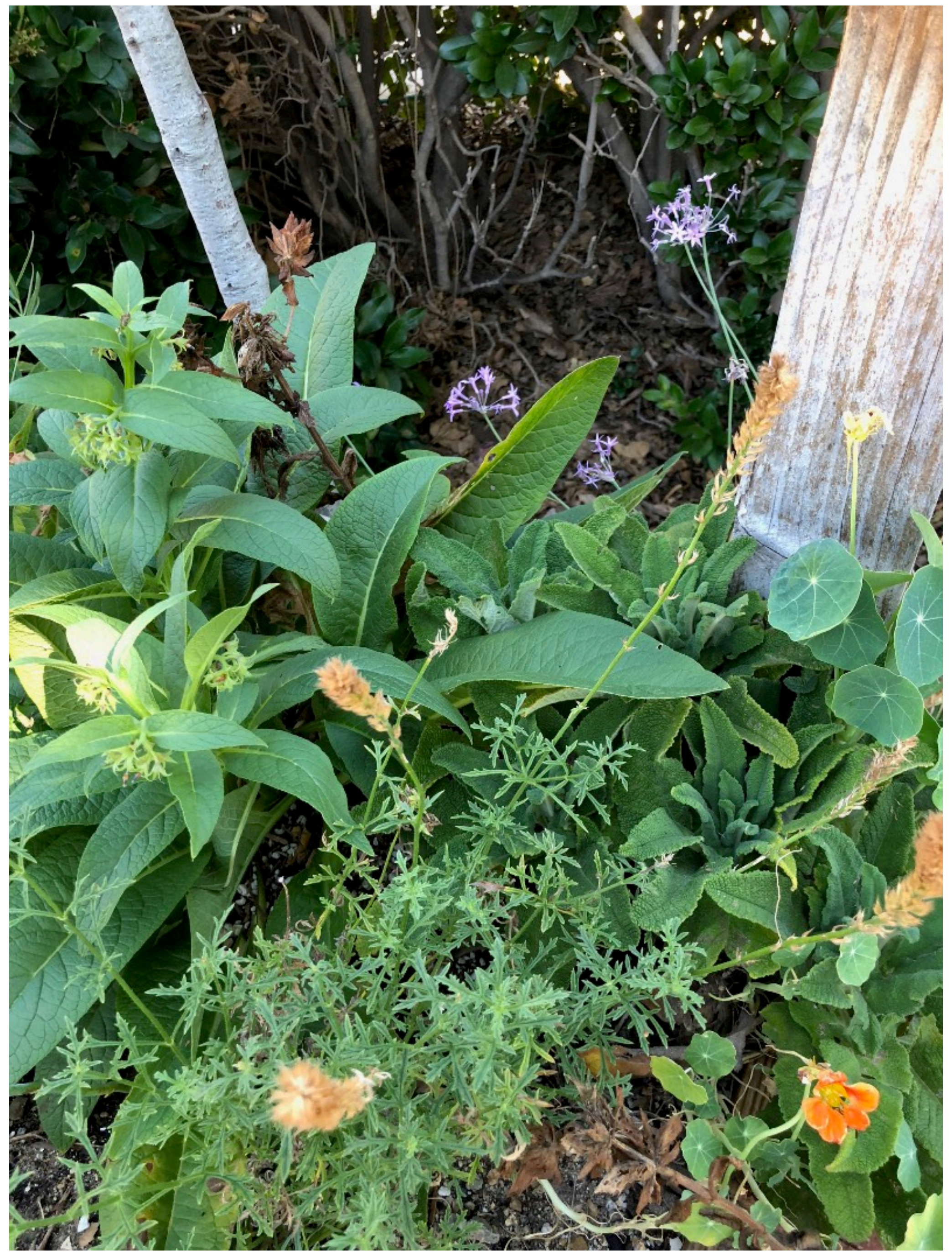
It is waterproof and has tripod quality adjustable view angle.

A plain RPi zero case is usually $5.

Some Lessons Learned So Far

- Experiment & observe before optimizing; change, document carefully & iterate
- Tune computer vision parameters carefully (thresholds, pixel area parameters, etc.)
- Get the lighting right! Control the lighting brightness with RPi (PWM & MOSFETs)
- Optimize network load; use isolated subnetworks (subnetworks add security, also)
- Reliability planning: recovery from power failures, internet outages, etc.
  - What to prioritize for battery backup; use laptops (have batteries) as hubs
  - When to connect with WiFi vs. wired ethernet
  - Self-restarting subsystems (init.d scripts versus systemd services)

Next: Indicator Plants to Measure Deep Soil Moisture



The most effective soil moisture test is an "indicator" plant that has deep roots and wilts before the plants around it do. Observation of such an "indicator" plant is a great use case for computer vision.

Interlude: Using project like this as a Resume Builder

- Build a portfolio that shows **broad practical experience**; even “hobby” projects are helpful
- Document your portfolio; emphasize specific examples and learnings (what’s special about it?)
- Demonstrate ability to **complete** large projects fully (80% solutions are good; 100% solutions show ability to finish)
- Show attention to project details (and to project **management** details)
- Examples of iteration and continuous improvement, including iteration of design
- **Effective documentation** for yourself, for others, for long term maintenance; for publications
- Effective use of Git/GitHub (or other version control) as a collaboration tool
- Communication skills (including to non-tech audience) and team collaboration skills
- Be able to tell a **short, compelling project story**; the “elevator speech” about your big project

Next Steps for This Project

- Deep learning for more specific conclusions from these computer vision pipelines
  - Reliably distinguish coyote vs raccoon; butterfly vs hummingbird; census counts
  - Recognize and label specific critter individuals (I can tell 2 different coyotes apart from a video; will develop code to have computer vision / machine learning do it)
  - Use CV / DL to classify “indicator plants” as “water stressed” (or not) in real time
- Measure partly cloudy vs hazy sun; compute effective value of sunlight for cumulative plant photosynthetic growth; visually measure water stress of plants
- Natural Language Processing for better communication of computer vision results
- Fine tuning parameters via interactive text commands (rather than changing yaml files)…or… using deep learning to optimize parameters?
- Implementing IOT actions: close the garage; irrigate specific garden areas, etc.

Helpful Resources

- **Electronics Resources:**
  - Adafruit has extremely helpful tutorials as well as a deep selection of electronic parts and kits: https://www.adafruit.com/
  - Sparkfun also has great tutorials and electronic goodies; lots of cameras; lots of fun gadgets: https://www.sparkfun.com/
- **ZMQ:** http://zeromq.org/ `and` the Python bindings: https://pyzmq.readthedocs.io/en/latest/
- **Permaculture:**
  - Toby Hemenway, *Gaia's Garden A Guide to Home-Scale Permaculture*. Also by Toby: *The Permaculture City: Regenerative Design for Urban, Suburban and Town Resilience*
    Link: http://a.co/izt7Xun
- **To learn more about (or contribute to!) this project:**
  - `https://github.com/jeffbass`