

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)



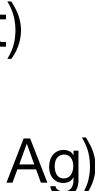
# How Computer Vision helps manage my small permaculture farm

- 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)



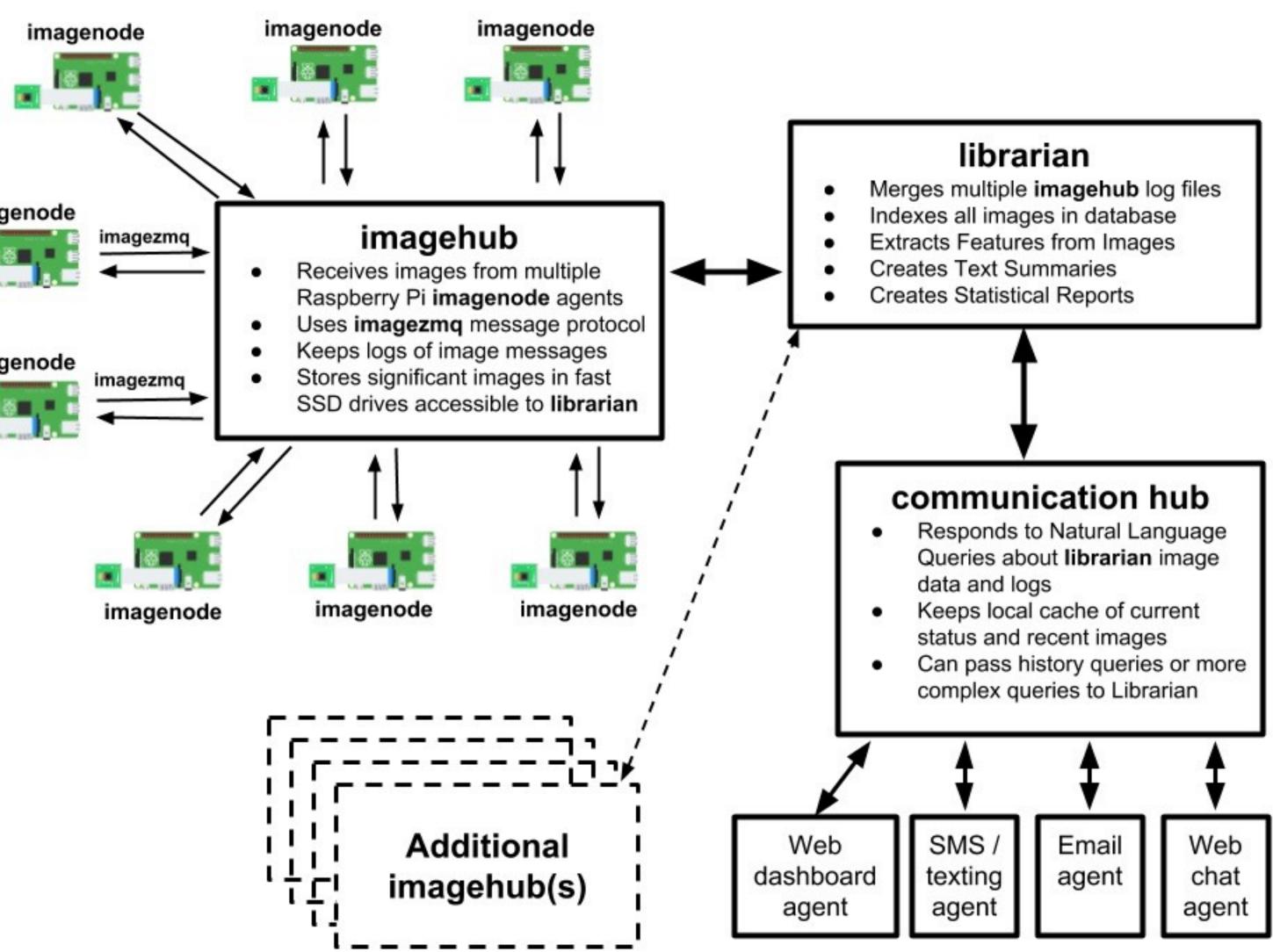
# Interlude: What the heck is permaculture?

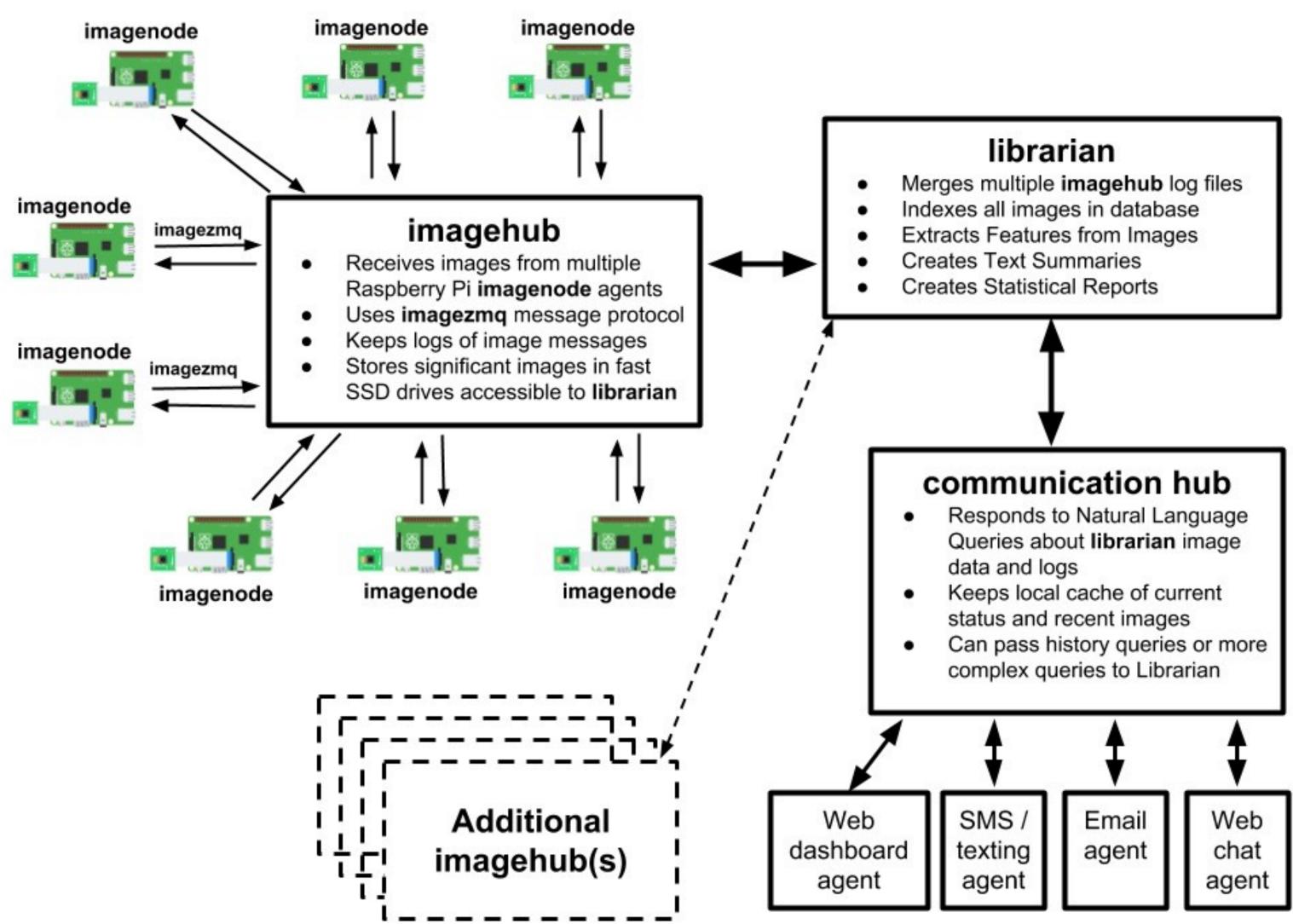
- Designing for long-term sustainability (centuries rather than decades)
- Creating deep living soil with diverse microbiology (emulating an old growth forest)
- Design choices to prioritize sustainability over efficiency (plant polyculture vs. Big Ag)
- Top down designs for land use, water capture and 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

```
# run this program on each RPi to send a labelled image stream
 2
     import socket
 3
     import time
     from imutils.video import VideoStream
 4
 5
     import imagezmq
6
7
     sender = imagezmq.ImageSender(connect_to='tcp://jeff-macbook:5555')
8
9
     rpi_name = socket.gethostname() # send RPi hostname with each image
     picam = VideoStream(usePiCamera=True).start()
10
11
     time_sleep(2.0) # allow camera sensor to warm up
12
     while True: # send images as stream until Ctrl-C
         image = picam.read()
13
         sender.send_image(rpi_name, image)
14
```

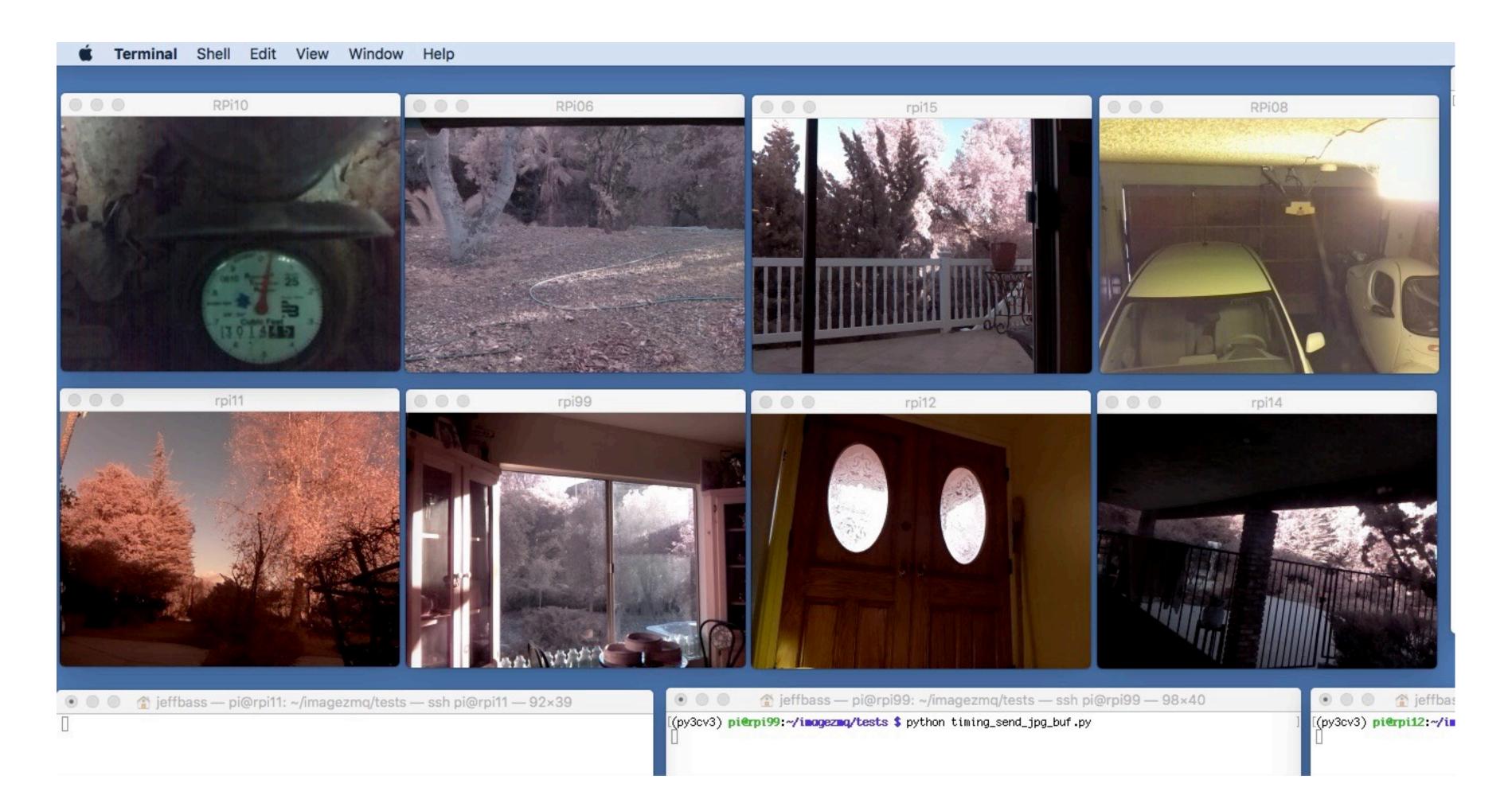
# run this program on the Mac to display image streams from multiple RPis import cv2 import imagezmq image\_hub = imagezmq.ImageHub() while True: # show streamed images until Ctrl-C rpi\_name, image = image\_hub.recv\_image() cv2.imshow(rpi\_name, image) # 1 window for each RPi cv2.waitKey(1) image\_hub.send\_reply(b'OK')

Node Code (RPi)

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
 1
 2
3
           (all the imports, logging setup hidden to fit slide)
     #
     settings = Settings() # get settings for node cameras, ROIs, GPIO
 4
 5
     node = ImageNode(settings) # start ZMQ, cameras and other sensors
6
 7
    # forever event loop
8
    while True:
9
         node.get_sensor_data() # grab camera images and other sensor data
         node.process_sensor_data() # detect motion, etc.
10
         while len(node_msg_q) > 0:
11
12
             try:
                 with Patience(settings.patience): # wait for hub response
13
                     text, image = node.msg_q.popleft()
14
                     hub_reply = node.send_frame(text, image)
15
             except Patience.Timeout: # except no timely response from hub
16
                  hub_reply = node.fix_comm_link()
17
             node.process_hub_reply(hub_reply)
18
19
     #
```

### 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

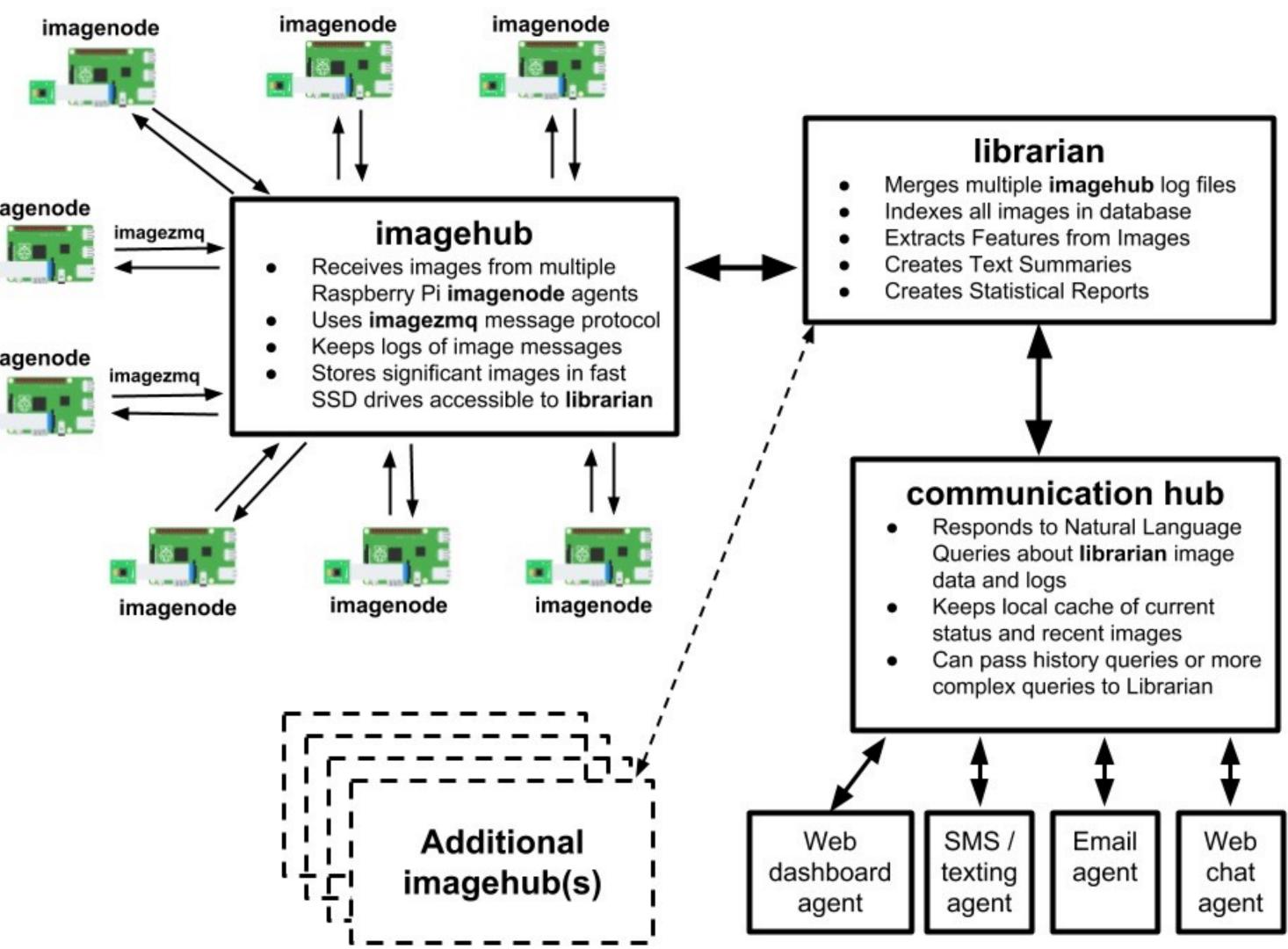
### What the Mac does:

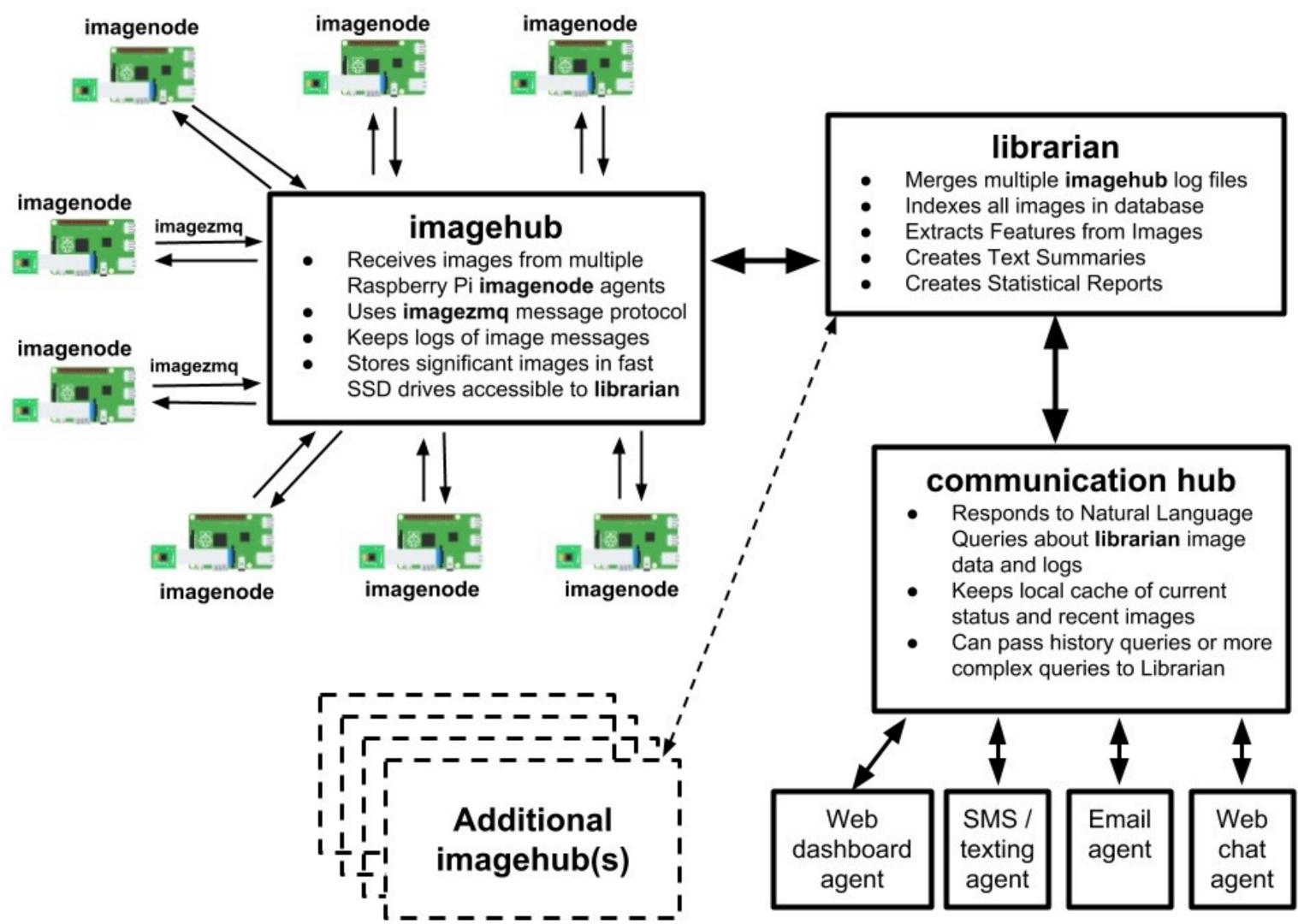
- Acts as ZMQ hub for multiple RPi's
- Receives and acknowledges images and messages from multiple RPi's
- Adds events messages to events log
- Stores and indexes images
- Extracts digits to read the meter
- Keeps history of events; does stats
- Responds to queries about status

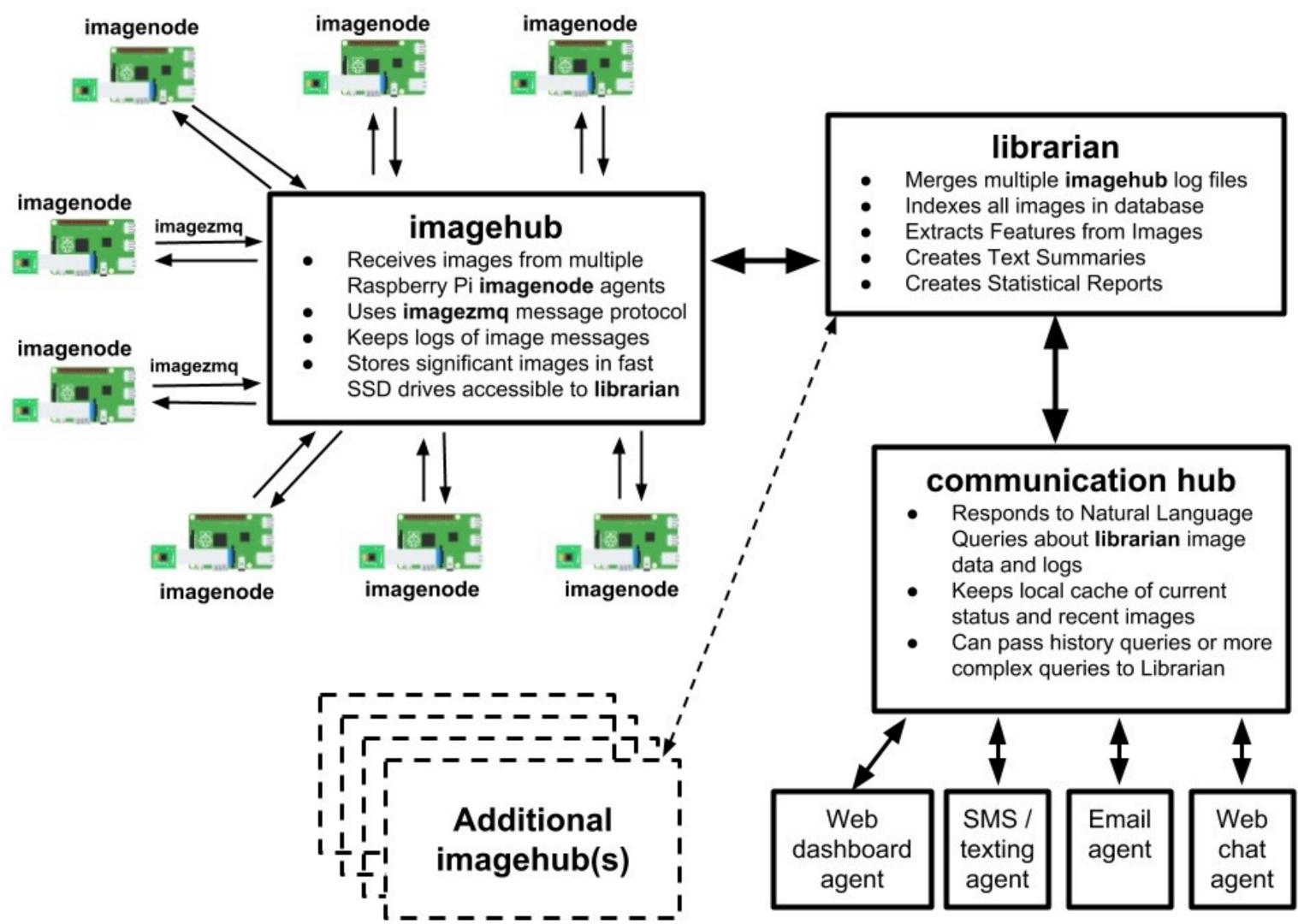


## **Distributed Computer Vision Pipeline (revisited)**

- 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

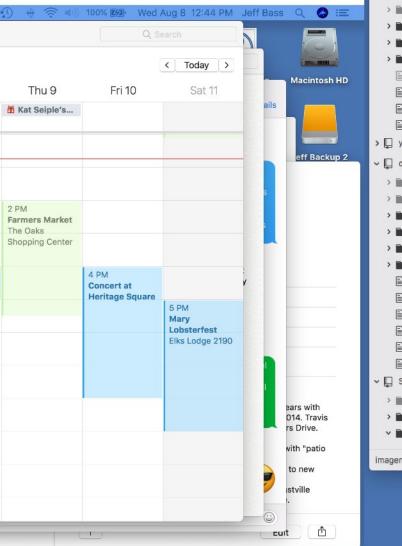




# Macs are Great (at some things)



- 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



### Terminal Shell Edit View Window Help

<ul> <li>South 1 1990 + 1991 + 19</li></ul>	Project	imagenode.py	imaging.py	imagenode.yaml	settings-yaml.rst	nodehealth	n.rst	testing.rst	README.rst	imagenode-uses.rst	utils.py	nodehealth.py	Code Bits.py
<ul> <li>Low definition</li> <li>Low definition<td>imagenode</td><td>16 import ppri</td><td>int</td><td></td><td></td><td></td><td></td><td>tests — jeffbas</td><td>ss@jeff-macbook: ~/S</td><td>DBdev2/imagezmq/tests</td><td>- python timing_rece</td><td></td><td>🛅 tests — pi@RPi10:</td></li></ul>	imagenode	16 import ppri	int					tests — jeffbas	ss@jeff-macbook: ~/S	DBdev2/imagezmq/tests	- python timing_rece		🛅 tests — pi@RPi10:
<pre>     Junction     Junction</pre>	—	17 import logg	jing										ile for imagenode.py PiCo
L. MAr J. M. MAr J.		18 import logg	jing.handlers										
<pre>     Autor with from initial input pressure with set input se</pre>			19 import traceback Size of last jpg bu										er
<pre>Makes for model, many for</pre>			= 230,400 by						(210, 620, 6)			patience: 10	
<pre>introductions is a first introduction where if it requires it is a first introduction where if it requires it is a first interpret in a first interpret into first interpret into a first interpret into</pre>			Elasped time: Elasped time:										
<pre>d drawneys b whey b whey</pre>			http://www.indoc.ing						ok ~/SDBdev2/imagezma	/tests \$ python timing_re	ceive_ipg_buf.pv		: False
<pre>b def r bid/i b def r bid</pre>			1	ind genoue			^C					H1: tcp://jeff-	
<pre># statuped = f statupe controlled automa who All f Process or SETTINF results """"""""""""""""""""""""""""""""""""</pre>		25 def main():	5 def main(): Total Number of						ved: 2,481			cameras:	
<pre> i true i isgoi: i</pre>		26 # set u	A the set up controlled shutdown when Kill Process or STATEDM received Number of Images										erDial
<pre>g _ meru, _ market _ loging _ models, _ loging</pre>			signal(signal.S)	IGTERM, clean_sh	utdown_when_kill	ed)			ived: 17.910 bytes				
<pre>De unaquencies be unaquencies be unaquencies be unaquencies be unaquencies be unaquencies be unaquencies be particis be p</pre>			7				Size of last	image received:				vflip: True	
<pre>Bingcodage Bingco</pre>			Compression ratio: 0 P										
<pre>machy massive mady test is a schup (manufacture is a first is a schup (manufactur</pre>	—		Log = Logging.getLogger() Elasped time: 70.03 second										lowing, off)
<pre>instantian in the instantian instantiantian instantian instantinstantinstantian instantian instantian instantian ins</pre>			(mu2ou2) inffranceinf						ok ~/SDBdev2/imagezma	<b>/tests \$</b> python timing_re	ceive_jpg_buf.py	] ROI1:	
<pre>products is provided if the set of the</pre>												top: 45	5
<pre>index.paralline in the interval in the interval inte</pre>	—												
<pre>b webcompany b webcompan</pre>							JeffOff	ice  Window  jpg				right:	82
<pre>a grows b closester b clo</pre>			Level(logging.DE	EBUG)			Size of last	jpg buffer rece				spread: 1	
<pre>D CORRECT D Control Log_info('Starting inager add, pr) Starting: Starti</pre>									(240, 320, 3)				: 35
<pre>producted status = settings = settings for node camers, RDIS</pre>	—		try: Compression ratio: 0.13										IV.
<pre>print("settings node name:, settings.nodename)</pre>			-		nas for node cam	eras. ROIs.	Approximate	FPS: 33.95			and any data bad and	gpio: 18	~
<pre>ndd = Imagelode(settings) # stor 20%, cameras and other sensors is called in the sensors is</pre>					-			rbussejerr-nucoo	ok ~75060ev27 tilogezilig	rests > python timing_re	ceive_jpg_our.py	T1:	
add       print("System Type is', node.health.sys_type)         accs       # forver event         amagening       while True:         ande.get_sensor_data() # grab camera images and other sensor data       mode.get_sensor_data() # detect motion, etc.         addition       mode.get_sensor_data() # detect motion, etc.         addition       intile lon(node.sensor_data() # detect motion, etc.         addition       intile lon(node.sensor_data() # detect motion, etc.         intile lon(node.sensor_data() # detect motion, etc.       intile lon(node.sensor_data() # detect motion, etc.         intile lon(node.sensor_data() # detect motion, etc.       intile lon(node.sensor_data() # detect motion, etc.         intile lon(node.sensor_data() # detect motion, etc.       intile lon(node.sensor_data() # detect motion, etc.         intile lon(node.sensor_data() # detect motion, etc.       intile lon(node.sensor_data() # detect motion, etc.         intile lon(node.sensor_data() # detect motion, etc.       intile lon(node.sensor_data() # detect motion, etc.         intile lon(node.sensor_data() # detect motion, etc.       intile lon(node.sensor_data() # detect motion, etc.         intile lon(node sensor_data() # detect motion, etc.       intile lon(node.sensor_data() # detect motion, etc.         intile lon(node sensor_data() # detect motion, etc.       intile lon(node.sensor_data() # detect motion, etc.         intintile lon(node sensor_data() # detect motion, etc. <td< td=""><td></td><td>42 <b>nod</b></td><td>de = ImageNode(se</td><td>ettings) # star</td><td>t ZMQ, cameras a</td><td>nd other se</td><td>nsors</td><td></td><td></td><td></td><td></td><td></td><td>3</td></td<>		42 <b>nod</b>	de = ImageNode(se	ettings) # star	t ZMQ, cameras a	nd other se	nsors						3
<pre>bageanq bit the True:: torp bageanq bit the True:: torp bageanq bit torp bageang bit torp bageanq bit t</pre>	.git				h.sys_type)							~	
<pre>set: </pre>	docs			op								~	
<pre>b 0.5.Store detect motion, etc. detect motion, etc. detect motion, etc. detect motion, etc. while lendmode.msg.q.&gt; 0 : try: with Patience(settings.patience): # wait for hub response # hub_reply = node.send_remotion() text, image = node.send_remoticxt, image) except Patience.limeout: # accept no lime(by response from hub log.info('No sensor hub reply) except Patience.limeout: # accept no lime(by response from hub log.uning('Virtue') + seconds') hub_reply = node.send_remoticxt, image) except for hub_reply(hub_reply) except for hub_reply = node.send_remotic. final(): ant_sensor_hub_reply = node.losed(): ant_sensor_hub_reply = node.losed(): ant_sensor_hub_re</pre>				n data() # anab		nd other se	neen dete					"watermeter.yaml"	' 39L, 683C written
<pre>diproc diproc dipr</pre>				-			nsor data		(a.a.a	orr 10.0 1 10		00/40	
<pre>g dupore g dupor</pre>					detect motiony c				Jeff	Office  Window  jpg		RPI10	
ARADME.rst       39       winn Mailande(settings.patience): # wait for hub response         BRADME.rst       31       # hub_reply = node.sng.g.applet(1)         hub_reply = node.sng.g.applet(1)       hub_reply = node.sng.g.applet(1)         hub_reply = node.sng.g.applet(1)       hub_reply = node.sng.g.applet(1)         hub_reply = node.sng.g.applet(1)       hub_reply = node.sng.g.applet(1)         igh       str(int(settings.patience)) + ' seconds')         igh       hub_reply = node.sng.g.applet(1)         igh       str(int(settings.patience)) + ' seconds')         igh       node.process.hub/necly(1th)         ubs       set         gdipnee       except KeyboardInterrupt, SystemExit):         igh       log.exception('Unanticipated error with no Exception handler.')         print('Traceback error', ex)       print('Traceback error', ex)         gdupnee       if 'node' in locals():         node.rub.rub.ax       if 'node' in locals():         node.rub.rub.ax       if 'node' in locals():         gdip.       sys.exit()         gdi       if 'name = 'main_' :         mode/magenode.rub       if 'name = 'main_' :         intik mark       if 'name = 'main_' :         intik mark       if 'name = 'main_' :         intik mark <td< td=""><td></td><td></td><td></td><td>3_1</td><td></td><td></td><td></td><td></td><td>12 5</td><td></td><td></td><td></td><td></td></td<>				3_1					12 5				
<pre>improve for the settings in the set of the settings in the set of the settings in the set of the settings in the setting in the setting</pre>		50	with Pa	atience(settings	.patience): # w	ait for hub	response		01				12 and
addsensor_hub       33       hub_reply = node.send_rime(text, image)        pycach       35       except Patience.Timeout; # except no timely response from hub        pycach       35       log.info('No sensor hub reply for '       + str(int(settings,com_link()             node.process_hub_reply(hub_reply)        gss       55       hub_reply = node.fix.com_link()             node.process_hub_reply(hub_reply)        usts       36       except (KeyboardInterrupt, SystemExit):             log.exception as ex: # traceback will appear in log             log.exception is ex: # traceback will appear in log             log.exception is ex: # traceback will appear in log             log.exception is ex: # traceback will appear in log             log.exception is ex: # traceback will appear in log             log.exception is ex: # traceback will appear in log             log.exception is ex: # traceback will appear in log             log.exception is ex: # traceback will appear in log             log.info('bxiting imagenode.py')             sobs.monor,hub_py             sobs	E README.rst	51								ACCOUNTS AND AND	b		and a second
<pre>backhapping b</pre>	yin-yang-ranch										AND THE		and the
<pre>Logochbc_ ipt content of the settings patience) + ' seconds') hbb_reply = node.tix_coom_link() node.process_hub_reply(hub_reply) utis content of the settings patience) + ' seconds') hbb_reply = node.tix_coom_link() node.process_hub_reply(hub_reply) utis content of the settings patience of t</pre>	oldsensor_hub	53			-		from bub						A CONTRACTOR
<pre>indt</pre>	pycache	55				ly response	Trom nub					100	The second
<pre>bigs 57 hub_reply = node.fix_comm_link() test 58 node.process_hub_reply(hub_reply) uits 59 yaml 60 except (KeyboardInterrupt, SystemExil):     Jog.sception('Unanticipated error with no Exception handler.') 50B_gloprain.py 61 print('Traceback will appear in log     Jog.sception('Unanticipated error with no Exception handler.') 50B_gloprain.py 63 log.sception('Unanticipated error with no Exception handler.') 50B_gloprain.py 64 print('Traceback will appear in log     Jog.sception('Unanticipated error with no Exception handler.') 50B_gloprain.py 64 print('Traceback error', ex) 50B_gloprain.py 65 finally: 50B_gloprain.py 66 if 'node' in locals(): 50B_gloprain.py 67 node.closeal() # close cameras, GPIO, files 50B_gRPi 68 log.info('Exiting imagenode.py') 50B_gloprain.py 7     test 11 ifname == 'main' : 50B_greated for an into acting: 12 main_' : 50B_greated for acting: 12 ma</pre>	.git					seconds!)			100		Contraction of the second s		
<pre>test 58 node.process_hub_reply(hub_reply) ulis 59 ulis 50B_comm_hub.py 51finally: 50B_comm_hub.py 51finally: 52B_comm_hub.phumac.sh 52 52B_comm_hub.py 53B_comm_hub.py 54finally: 55B_comm_hub.py 55B_com</pre>						seconds ,				The second of the	- Wart	7 Cash	18
usis       60       except (KeyboardInterrupt, SystemExit):       log.warning('ttrl-C was pressed or SIGTERN was received.')         g. dignore       61       log.warning('ttrl-C was pressed or SIGTERN was received.')         g. SDB_comm_hub.py       63       log.exception('Unanticipated error with no Exception handler.')         g. SDB_librarian.py       64       print('Traceback error:', ex)       '/(ip: True, '/'''''''''''''''''''''''''''''''''''	test	58								A CALLER A		19166	
yaminum       61       log.warning('Ctrl-C was pressed or SIGTERM was received.')         g. gitgnore       62       except Exception as ex: # traceback will appear in log         j. SDB_com_hub.py       63       log.exception('Unanticipated error with no Exception handler.')         j. SDB_sensor.hub.py       65       finally:         j. Start_sensor.hub.py       65       finally:         j. start_sensor.hub.chromebook.sh       66       if 'node' in locals():         j. start_sensor.hub.chromebook.sh       67       node.closeall() # close cameras, GPIO, files         j. spl.       log.info('Exiting imagenode.py')       sensor settings         j. git       70        git       70        git       71        git       72        git       73	a utils	59											
<pre>g-ditionce 61 log.warning('Ctrl-C was pressed or SIGTEM was received.') except Exception as ex: # traceback will appear in log 'Vilepide': "Window']) exception('Unanticipated error with no Exception handler.') print('Traceback error:', ex) 5DB_bensor_hub.py 61 print('Traceback error:', ex) 5DB_sensor_hub.py 62 finally: 53DB_sensor_hub.phromebook.sh 63 if 'node' in locals(): 64 node.closseall() # close cameras, GPID, files 65 star_sensor_hub_thronoe bettings 66 sys.exit() 67 node.closseall() # close cameras, GPID, files 68 log.info('Exiting indenode.py') 59B_RPi 69 sys.exit() 70 test 71 ifname == 'main' : 73 node/imagenode.py 46:51 74 ifname == 'main' : 75 node/imagenode.py 46:51 75 finally: 75 node/imagenode.py 46:51 75 node/imagenode.py 75 node/imageno</pre>	yaml												
SDB_comm_hub.py       62       exception ('Unanticipate arror with no Exception handler.')       ivigence:'' ('fip:': True, 'vigence:'', 'findo'')         SDB_comm_hub.py       64       print('Traceback error:', ex)       ''fip:': True, 'vigence:'', 'findo'')         SDB_sensor_hub.py       65       if 'node' in locals():       ''fip:': True, 'vigence:'', 'findo'')         a start.sensor_hub.chromebook.sh       66       if 'node' in locals():       ''fip:': True, 'vigence:'', 'fip:'' 'DiSB282')}         g start.sensor_hub_chromebook.sh       66       if 'node' close cameras, GPIO, files       ''fip:': True, 'vigence:'', 'fip:'' 'DiSB282'}         g start.sensor_hub_mac.sh       67       node.closeall() # close cameras, GPIO, files       Sensor is trut totion.         g spt.       69       sys.exit()       sensor instantiation.       sensor instantiation.         .ght       70       rode: closeall() # close cameras, GPIO, files       sensor instantiation.       sensor instantiation.         .ght       71       if _name_ == '_main_':       main()       sensor instantiation.       sensor instantiation.         .git curve       72       main()       main()       sensor instantiation:       y         .git curve       73       reactions curve       sensor instantiation:       y         .gob int curve       73       reactinses										☆ jeffbass — pi@rpi99:	~/imagenode/imagenode	e — ssh pi@rpi99	
SDB_librarian.py       64       print('Traceback error:', ex)       'viewnie': 'window'}}         SDB_sensor_hub.py       65       finally:       sensors settings         SDB_sensor_hub_chromebook.sh       66       if 'node' in locals():       sensors settings         Start_sensor_hub_mac.sh       67       node.closeall() # close cameras, GPI0, files       Sensors (hubmits)         Start_sensor_hub_mac.sh       67       node.closeall() # close cameras, GPI0, files       Sensors (hubmits)         Start_sensor_hub_mac.sh       68       log.info('Exiting imagenode.py')       sensors instantiation.       s.ny         splt       69       sys.exit()       sensors list length 1       sensors (hubmits): jag         .glt       70       comera (homes: 'N instantiation.       sensors istantings: jag         .utils       72       main()       remain()       remain()       y         P init_ent_       73       rom settings: 32       y       y         rom code/imagenode.py       46:51       Y       send_nows from settings: 0       s.ny						-	• •						pi@rpi10 — 80×24
SDB_sensor_hub.py       65       finally:         SDB_sensor_hub.chromebook.sh       66       if 'node' in locals():         start_sensor_hub_chromebook.sh       66       if 'node' in locals():         start_sensor_hub_mac.sh       67       node.closeall() # close cameras, GPIO, files         SDB_RPI       68       log.info('Exiting imagenode.py')       sensor in tomerainit 11         .git       70       sensor in comerainit 12       sensor in stantiation.         .git       70       sensor in comerainit 14       y         .git       70       Sensor sensor is telores tings: jpg       Comera key is: P1 just before comera instantiation.       y         .git       72       main()       72       main()       y         .git       73       .git       Sensor is: Senso	—				with no excepti	on nandter.	)						
start_sensor_hub_chromebook.sh       66       if 'node' in locals(): node.closeall() # close cameras, GPI0, files       ImageNode got the settings HealtMinitor got the settings Sensor Ki Just before Sensor instantiation. Sensor Ki Just before Camera instantiation. comera (N) if _name_ == '_main_' : main()       s.py         .git       70       Sensor Lint_ P1 Sensor Ki Just before Camera instantiation. comera (N) is 32 Vilip from settings: 32 Vilip from settings: 32 Vilip from settings: 32 Vilip from settings: ('sent': 50) Frame_count: 59 Viewname from settings: ('sent': 50) Frame_count: 59 Viewname from settings: Window       s.py									{'T1': {'g	pio': 4, 'type': 'DS18B20	'}}		у
67       node.closeal() # close cameras, GPIO, files       Healthnontor got the settings         68       log.info('Exiting imagenode.py')       sensor in Camerainit T1         50B_RPi       69       sys.exit()         .git       70         1       ifname == 'main' :         72       main()         73       rome-closeal() # close cameras, GPIO, files         9       sys.exit()         9       sensor in camerainit T1         9       reatings: (ideo// ideo//				s():					ImageNode	got the settings			
68       log.info('Exiting imagenode.py')       sensor in ComerainitT1       s.py         SDB_RPi       69       sys.exit()       Sensors list length 1       y         .git       70       Send_type from settings: jpg       Camera Key is: P1 just before Camera instantiation.       y         test       71       ifname == 'main' :       Camera Key is: P1 just before Camera instantiation.       y         utils       72       main()       Resolution from settings: (320, 240)       y         utils       73       Send_amount from settings: 32       y         utils       74       Send_amount from settings: 32       y         viewname from settings: ('event': 50)       Frame_count: 50       Send_amount from settings: Vindow         viewname from source (src) from settings: 0       s.py	start_sensor_hub_mac.sh	67	<pre>node.closeall()</pre>	) # close camera	s, GPIO, files						r instantiation.		
69       sys.exit()         .git       70         test       71         ifname == 'main':         utils       72         main()         rode/imagenode.py 46:51    Sensors list length 1 Send_type from settings: (328, 248) Framerate from settings: (328, 248) Framerate from settings: (328, 248) Framerate from settings: (170, 248) Framerate fro				imagenode.py')									s.py y
test 71 ifname == 'main': utils 72 main() Disit or 73 node/imagenode.py 46:51 Camera Key is: P1 just before Camera instantiation. camera in Camerainit P1 Resolution from settings: 320, 240) Fromerate from settings: 320 Vf lip from settings: True Send_amount from settings: {'event': 50} Frame_count: 50 Vebcam source (src) from settings: 0 Sendy Settings: 8 Sendy Settings: 8 Settings: 8 Sendy Settings: 8 Settings:			exit()						Sensors li	st length 1			1
utils       72 main()         node/imagenode.py       46:51    Resolution from settings: (320, 240) Framerate from settings: 32 Vflip from settings: True Send_amount from settings: ['event': 50] Frame_count: 50 Frame_count: 50 Vebram source (src) from settings: 0 s.py			- I mate I						Camera Key	is: P1 just before Camer	a instantiation.		
The initial of the settings: 32       Framerate from settings: 32         Framerate from settings: 17ue       Y         Inode/imagenode.py       46:51         Frame-count: 50       Frame-count: 50         Frame-count: 50       Frame-count: 50         Velue       Velue         Very work       Very work         Very work       Send_amount from settings: Vindow         Very work       Very work         Very work       Send_source (src) from settings: 0			_ == 'main' :								)		У
node/imagenode.py 46:51 Frome_count: 50 viewname from settings: {'event': 50} Frome_count: 50 viewname from settings: Window Webcam source (src) from settings: 0 \$.py	E) init ov								Framerate	from settings: 32	50		у
Frame_count: 50 viewn_coe from settings: Window Webcam source (src) from settings: 0 s.py	enode/imagenode.py 46:51								Send_amoun	t from settings: {'event'	: 50}		
Webcam source (src) from settings: 8 S-py													
											0		s.py y

te processing of images n; classification h power as 8 or 10 RPis le for outdoors



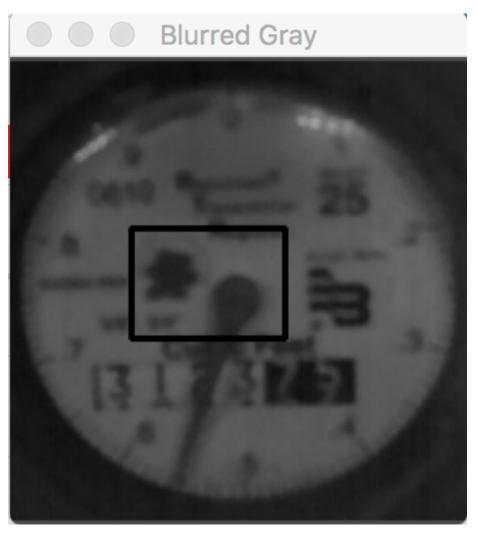
# Image Pipeline for Reading a Water Meter



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 ison or configs
  - Large settings file becomes a nested Python dict

Itests — pi@RPi10: ~ — ssh pi@rpi10 — 66×42 # Settings yaml file for imagenode.py PiCamera node: name: WaterMeter aueuemax: 50 patience: 10 heartbeat: 10 send\_type: jpg print\_settings: False ub\_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: 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**

- 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



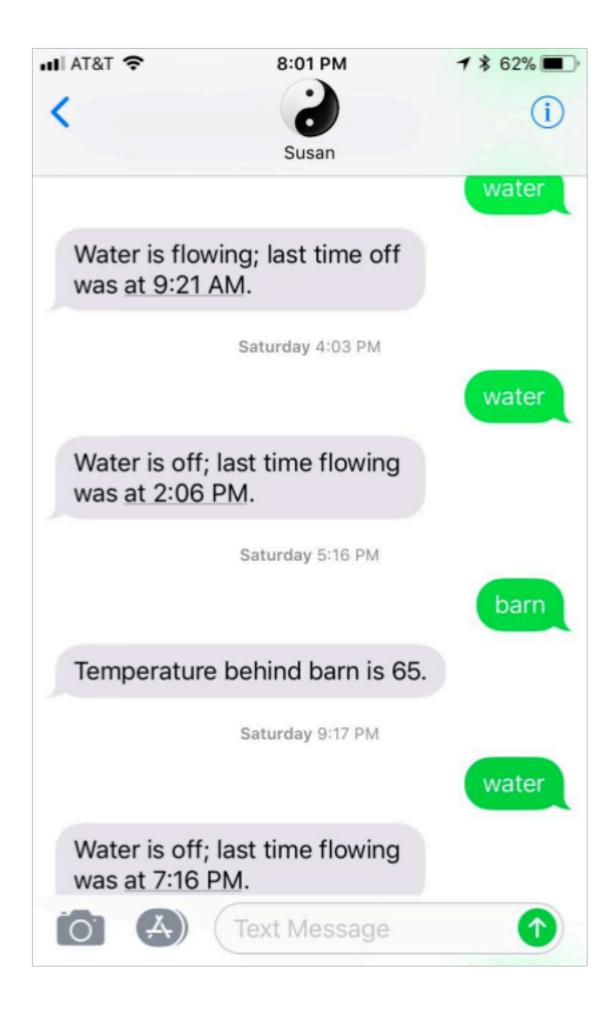
# **Communications: Some Learnings**

- 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 ;-)

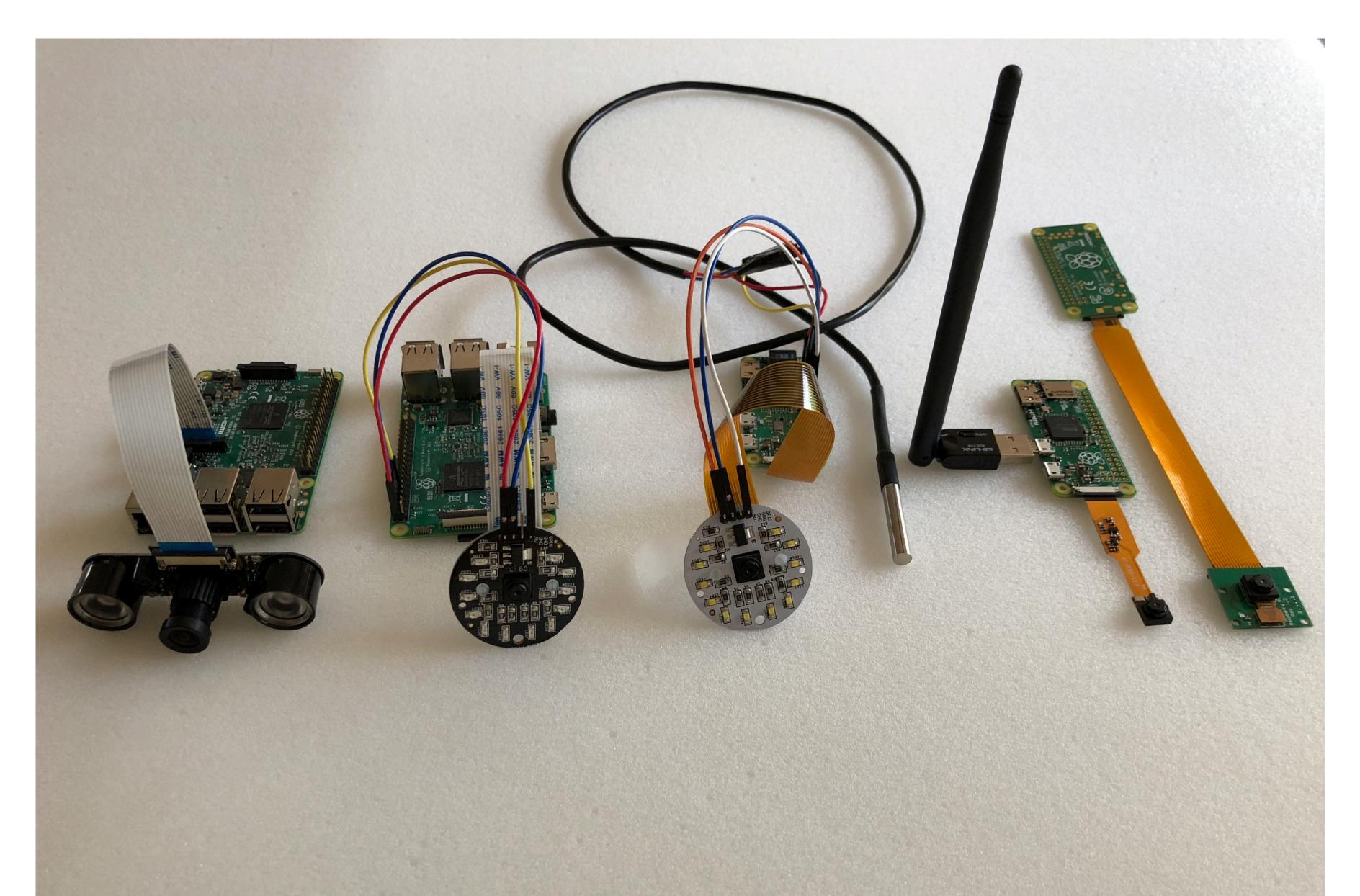
- Communicating "conclusions" from computer vision observations:
  - "Water is off; last time flowing at 7:15am"
  - "Garage is closed; last open at 8:20pm yesterday"
  - "Animal spotted at 3:15am; coyote (77%)"
- Communicating via multiple channels
  - Simple CLI text chat interface for testing
  - 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:

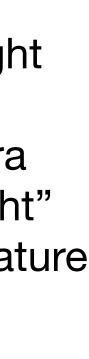
RPi with Waveshare Combo IR lens and dual IR floodlights

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

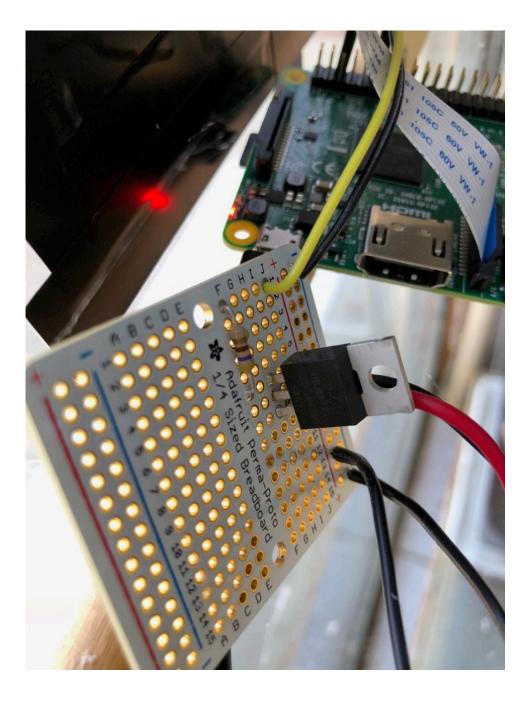




### Hardware: More Light Types

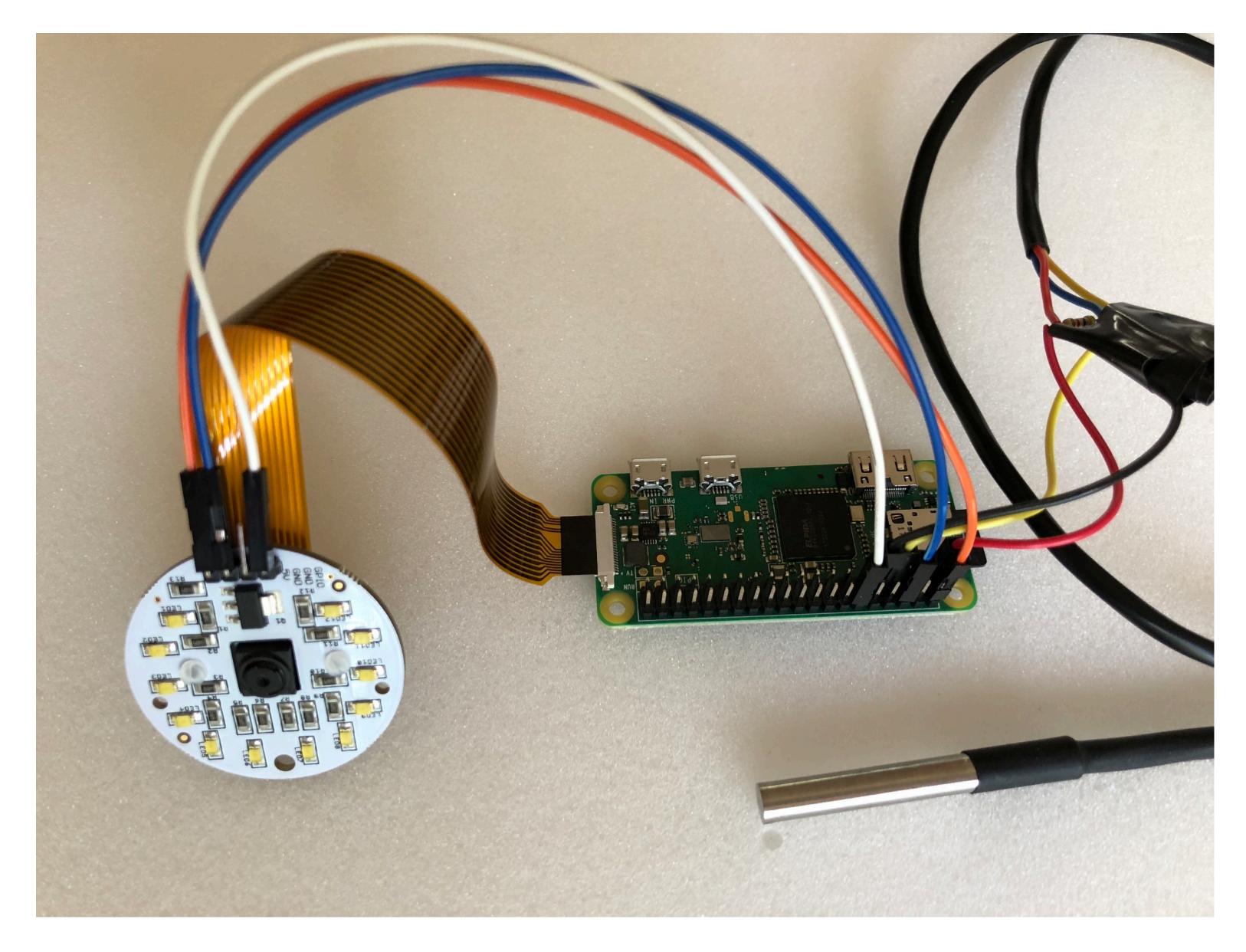


White floodlight (18W), 12V power cube, Infrared floodlight (8W)



Simple MOSFET circuit provides RPi GPIO control for floodlights

### Hardware Enclosures: Light Fixtures!

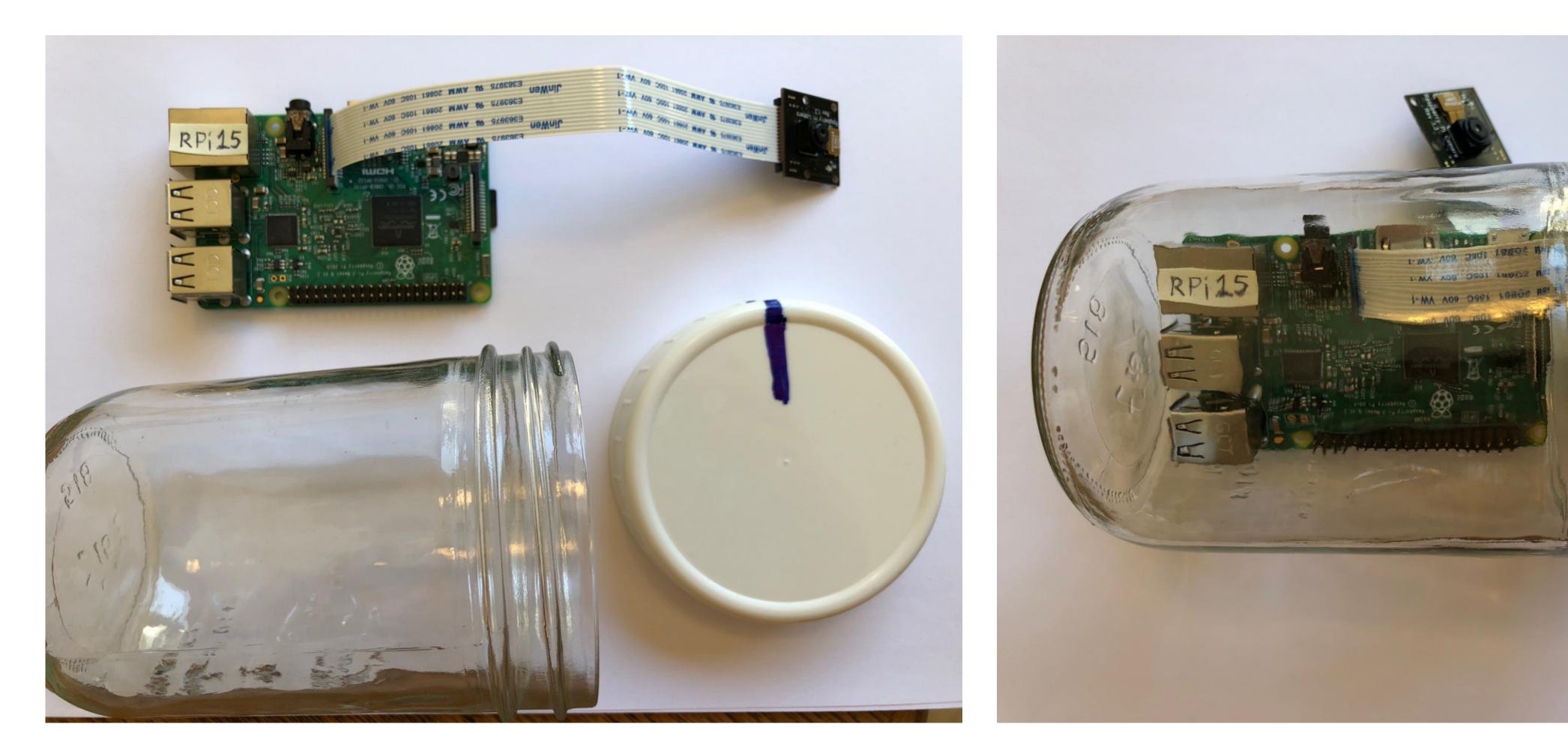


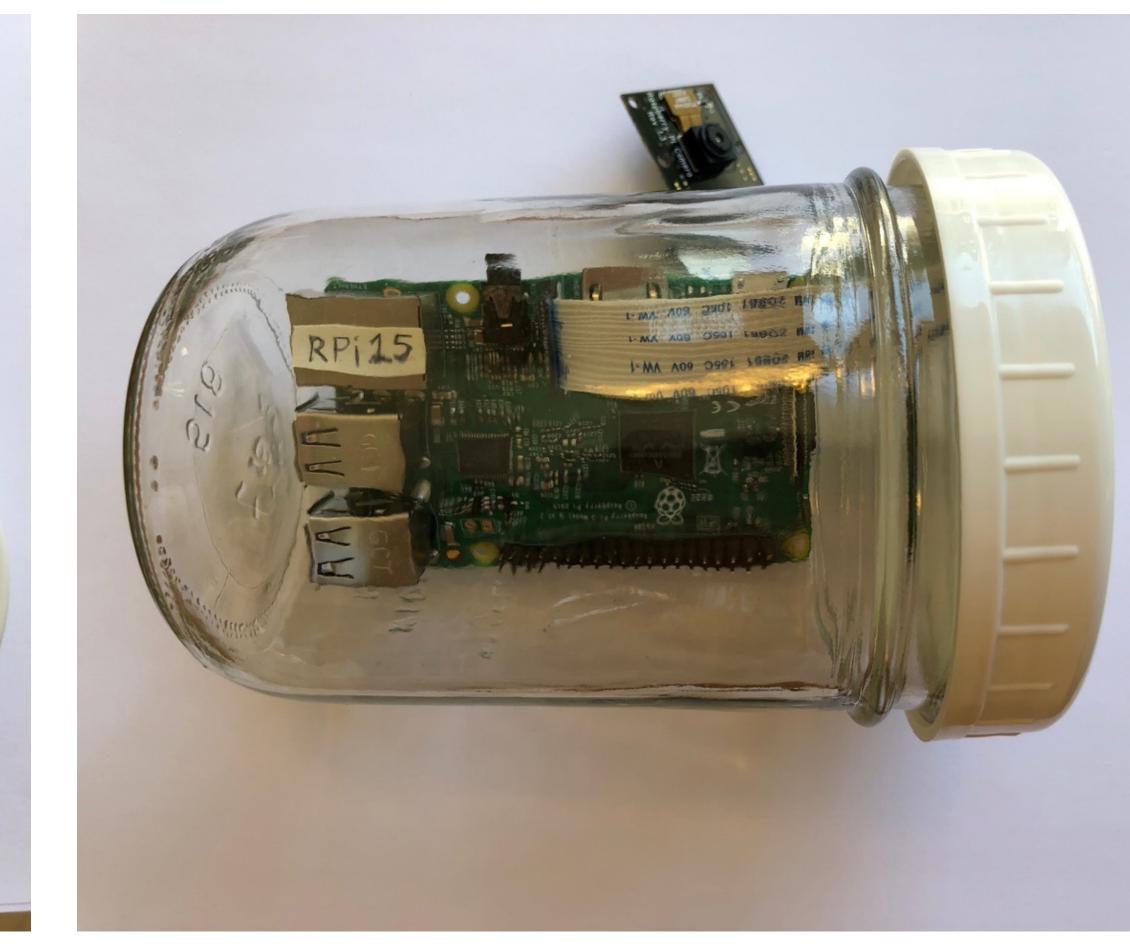


... and temperature sensors!



### Hardware Enclosures: Mason Jars!

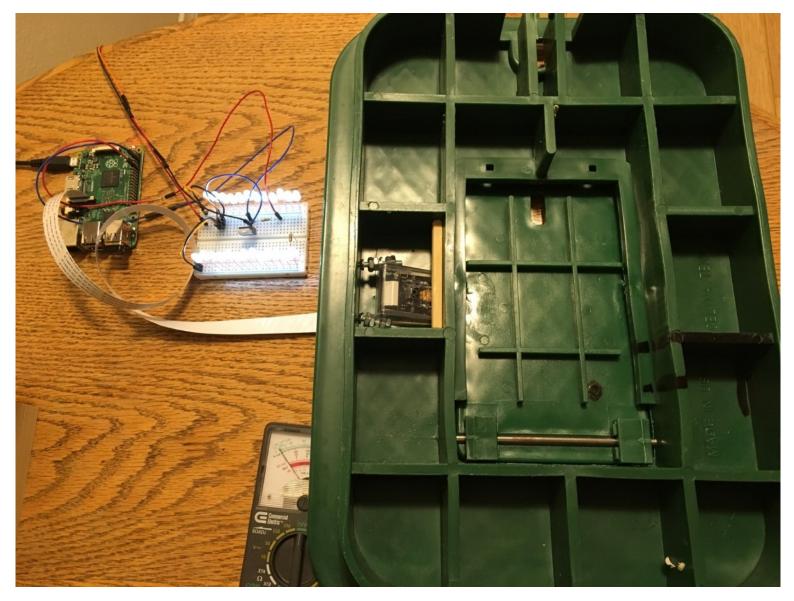


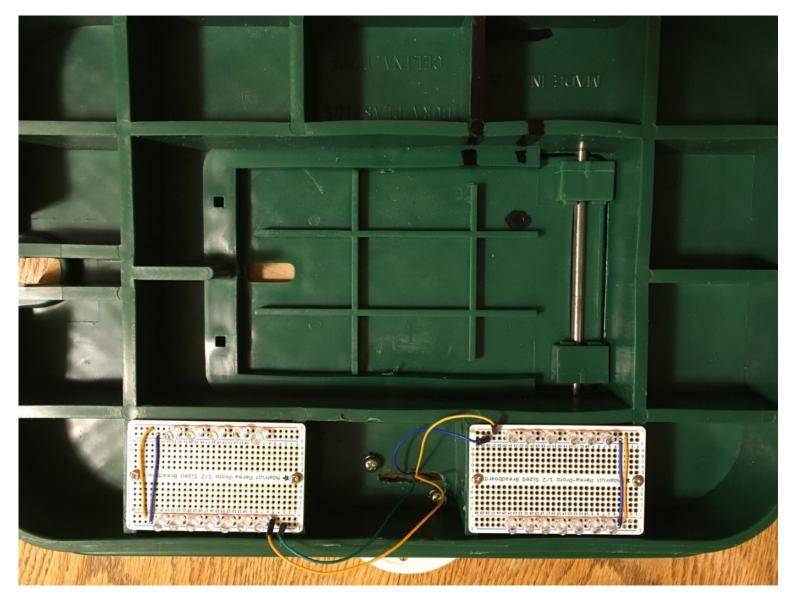




### Hardware Enclosures: Water Meter Cam









### Hardware Enclosures: Barn Coyote Cam



under a couple of old cedar shingles.

### Interlude: Coyote Cameos with Rabbit

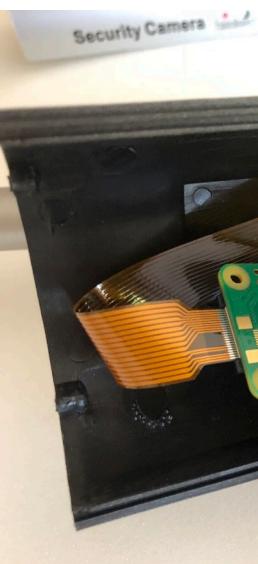


Taken with RPi "PiNoir" Infrared camera, using an 8W Infrared floodlight



### Hardware Enclosures: Fake Security Cam for \$5





It is waterproof and has tripod quality adjustable view angle.

A plain RPi zero case is usually \$5.

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



### 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:
  - www.adafruit.com/
  - www.sparkfun.com/
- **ZMQ:** <u>http://zeromq.org/</u> and the Python bindings: <u>https://pyzmq.readthedocs.io/en/latest/</u>
- Permaculture:
  - **Link:** http://a.co/izt7Xun
- To learn more about (or contribute to!) this project:
  - https://github.com/jeffbass

Adafruit has extremely helpful tutorials as well as a deep selection of electronic parts and kits: <u>https://</u>

• Sparkfun also has great tutorials and electronic goodies; lots of cameras; lots of fun gadgets: <u>https://</u>

• 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