Using a Camera with Raspberry Pi

Tutorial /Lab

Mike Sydor, Software Performance Consultant

GOAL

- Do some stop-motion image capture
- Go beyond ‘packaged’ solutions
  - Bend the technology to our will!
- Work with Python
- Practice some software engineering concepts

AGENDA

- Motivation
- Readily Available Software
  - Not everyone wants to code…
- Install and Configure
  - So you can get the the same place as the lab exercise
- ‘Hello World’ for cameras
- Requirements for Time-lapse Operations
- Helper Functions
- Bringing the Pieces Together
- Links
- As Time Allows
  - Current State of the Project

MOTIVATION

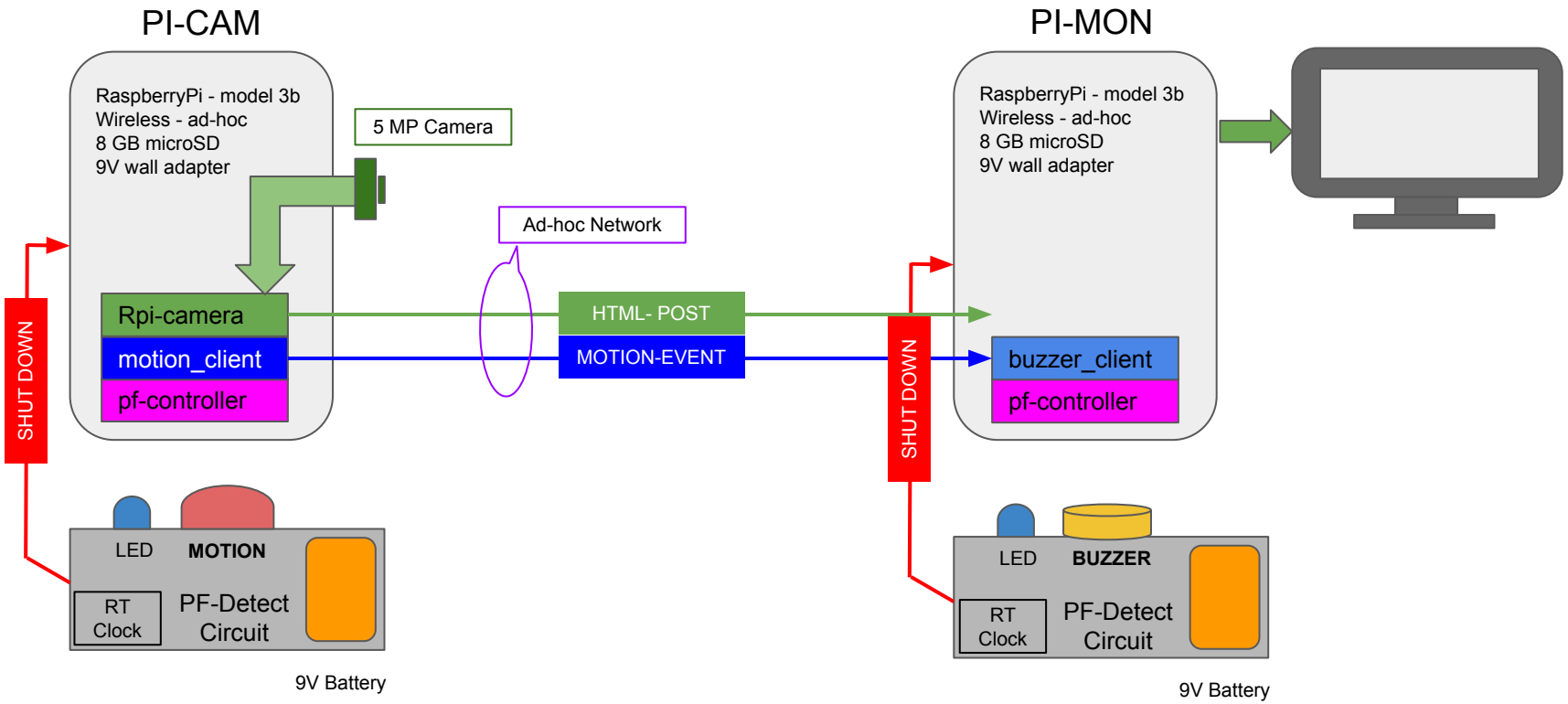
- Generate and Render some content for YouTube

- Also..
  - Nature Photography - time-lapse of flower opening and closing
  - Security - On Motion, take a picture and send it via email
    - What kind of critters are in my yard
      - Can I identify them automatically?
        - Image Recognition
        - Image Analysis and Enhancement
          - openCV (computer vision)
          - numpy, tensorflow (machine learning)

Readily Available Software

- Single Camera
  - RPi-Cam Web Interface !! git clone https://github.com/silvanmelchior/RPi_Cam_Web_Interface.git
- Multiple Cameras
  - motionpi !! https://github.com/ccrisan/motioneyeos/releases
  - https://pimylifeup.com/raspberry-pi-security-camera

Solution Architecture



picamera

This package provides a pure Python interface to the Raspberry Pi camera module for Python 2.7 (or above) or Python 3.2 (or above).

Links

- The code is licensed under the BSD license
- The source code can be obtained from GitHub, which also hosts the bug tracker
- The documentation (which includes installation, quick-start examples, and lots of code recipes) can be read on ReadTheDocs
- Packages can be downloaded from PyPI, but reading the installation instructions is more likely to be useful

Table of Contents

INSTALL AND CONFIGURE

- raspi-config
  - (5) Interfacing Options
    - Enable Camera
    - (opt) Enable SSH
    - (opt) Enable VNC
      - https://www.realvnc.com/en/connect/docs/raspberry-pi.html#raspberry-pi-setup
  - (7) Advanced
    - Hostname = picam
- (opt) sudo systemctl start vncserver-x11-serviced.service
- (opt) sudo systemctl enable vncserver-x11-serviced.service
- Reboot - and ready for test
  - sudo apt-get install python3-picamera
  - (opt) sudo apt-get install realvnc-vnc-server
  - (opt) sudo apt-get install realvnc-vnc-viewer
  - (opt) sudo apt-get install git-core
  - git clone git://github.com/spyderjacks/HackCCM.git !! Code and slides for the tutorial sessions
    - https://projects.raspberrypi.org/en/projects/getting-started-with-git

‘Hello World’ FOR CAMERAS

- Selfie
  - raspistill -o test.jpg
- Stream
  - raspivid -o vid.h264
  - omxplayer vid.h264

- Open a command window
- Run the commands
- Open the File Explorer
- View the results (jpg only)

REQUIREMENTS FOR TIME-LAPSE OPERATIONS

- Command-line Solution
  - Fussy typing
    - `avconv -r 10 -i image%04d.jpg -r 10 -vcodec libx264 -vf scale=1280:720 timelapse.mp4`
  - Need to be there (local or remote)
- Automation Solution
  - Loop
  - time.sleep()
  - Trigger event?

- Trade-offs? Shortcomings?

PseudoCode

Inits

    Get a camera object

Loop for # images

    Capture image

    Sleep

Until done

Cleanup

Complications

- time.sleep() - this is a *blocking* call
  - short.sleep()
- How to name files and directories
- Runs once, otherwise we need an event to restart it
  - Trigger Event - motion sensor
    - Need to learn about GPIO
    - Some packaged solutions use image analysis for motion detection
      - Shortcomings?

HELPER FUNCTIONS

- How to implement light.sleep
- How to name a file, sequentially
- How to set up a directory, with timestamp

lightSleep()

```
1234 ################################
1235 ## PROCESS SYNCHRONIZATION ###
1236 ################################
1237 """
1238 # sleep() is a blocking call so let's give plenty of chance
1239 # to process any interrupt
1240 #
1241 def lightSleep( delay ):
1242     for i in range(0, int(delay*10 )):
1243         time.sleep(0.1)
1244
```

createAndOpenDir()
createTimeStampName()

```
1173 def createAndOpenDir( camera, path):
1174
1175         now = datetime.now().strftime('%Y%m%d_%H%M%S')
1176         dirname = path + '/' + camera + '_' + now
1177
1178         # create the directory
1179         os.mkdir(dirname)
1180
1181         return dirname
1182
1183 def createTimeStampName(fileType):
1184
1185         now = datetime.now().strftime('%Y%m%d_%H%M%S')
1186         filename = now + fileType
1187
1188         return filename
1189
```

BRINGING THE PIECES TOGETHER

APIs + Libraries + Glue Code = Programs

PROGRAM STRUCTURE

- Declarations
- Helper Functions
- Initialization
- Loop
- Cleanup

Challenge #1

- Create a Directory with timestamp (helper function)
- Create a function to *sequentially number* the images (borrow from createTimeStampName()) after passing imageCount
- Collect 10 images, whatever delay you like, rendered as *.jpg

Challenge #2

- Render the current collection of images
- `avconv -r 10 -i image%04d.jpg -r 10 -vcodec libx264 -vf scale=1280:720 timelapse. mp4`
  - https://www.raspberrypi.org/documentation/usage/camera/raspicam/timelapse.md

- (review **t_render2.py**) for an External Command Strategy

LINKS

- https://thepihut.com/blogs/raspberry-pi-tutorials/16021420-how-to-install-use-the-raspberry-pi-camera
- http://picamera.readthedocs.io/en/release-1.10/
- https://libav.org/avconv.html

‘Red Pill’ Topics

- https://docs.opencv.org/3.4.1/d0/de3/tutorial_py_intro.html
- https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-data-science-part-1-f18242424c38
- https://www.quora.com/What-are-the-best-Python-libraries-for-data-science/answer/Atul-Kumar-Singh-129?share=abfcceb4&srid=o0YR

WRAP-UP

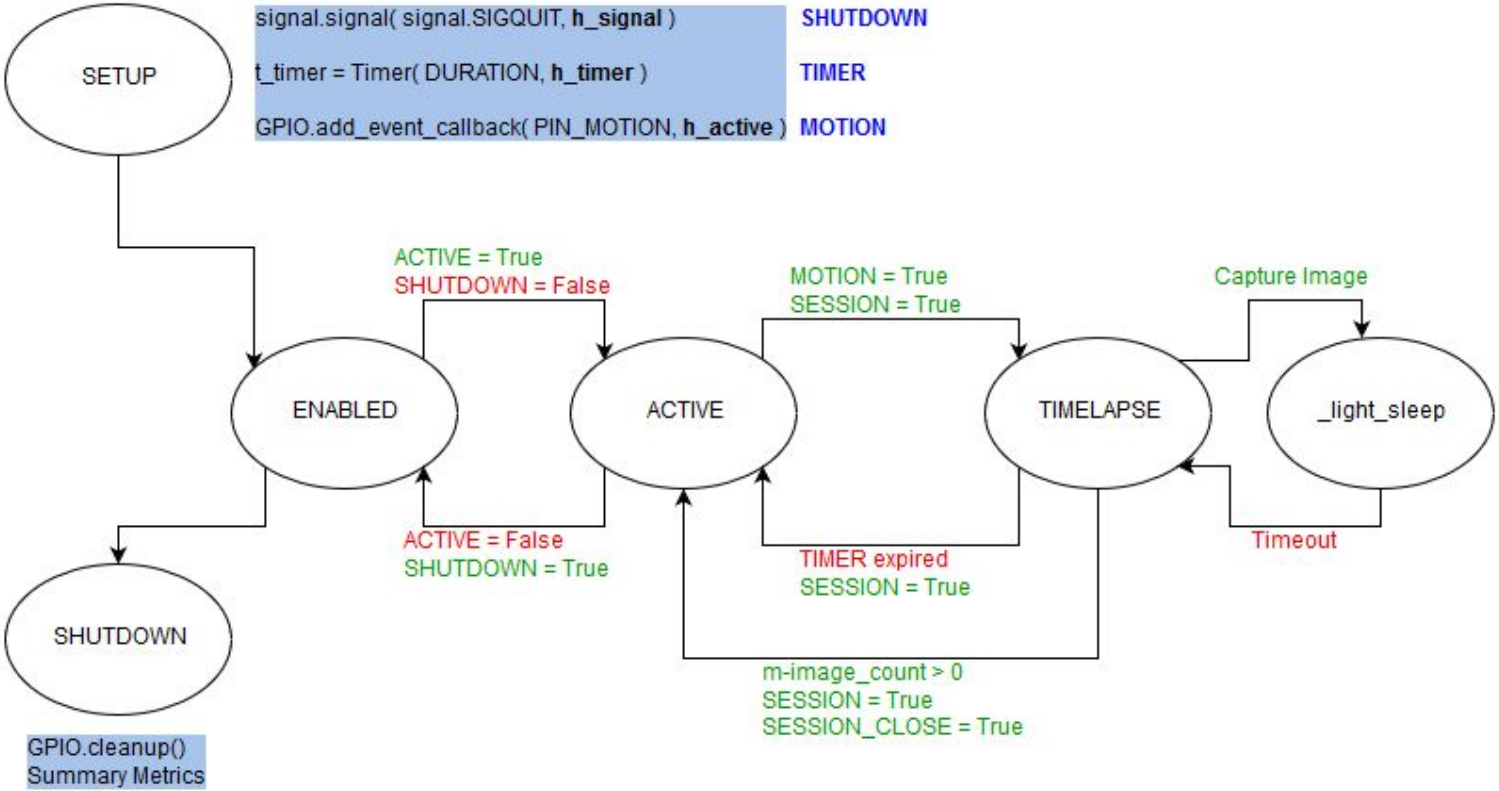
- Cameras - can be cool
- There usually is a library that can simply your effort
- Build the smallest possible program, first.
  - Test
  - Then add features.
  - Iterate
- What else would we need for a robust solution???

Thanks for Your Attention

AS TIME ALLOWS

- State Diagram
- Camera Code Walkthrough
  - Shared (package of helper functions)
  - Handlers
  - Global variables for multi-process support
    - (1) LED, multiple status updates (different programs)
    - (1) Logfile, multiple writers
    - (10) devices, one configuration (env)

State Diagram for camera.py



State Machine Implementation

```
37 import time
38 from threading import Timer
39
40 MOTION = False
41
42 INTERVAL = 2
43 DURATION = 10
44 def h_timer():
45     global MOTION
46     MOTION = False
47     print('timer expired')
48
49 t_timer = Timer(DURATION, h_timer)
50 t_timer.start()
51 #t_timer.cancel()
52
53 # Simulate MOTION event...
54 #
55
56 MOTION = True
57 time.sleep(1)
```

```
59
60 while MOTION == True:
61     print('first session...')
62     if( t_timer.is_alive() ):
63         print('taking a picture...')
64         time.sleep(INTERVAL)
65     else:
66         print('timer is not alive...')
67         #MOTION = False
68
69 time.sleep(1)
70 t_timer = Timer(DURATION, h_timer)
71 t_timer.start()
72 MOTION = True
73
74 while MOTION == True:
75     print('second session...')
76     if( t_timer.is_alive() ):
77         print('taking a picture...')
78         time.sleep(INTERVAL)
79     else:
80         print('timer is not alive...')
81         MOTION = False
82
83 time.sleep(2)
84 print('no motion...')
```

```
In [36]: runfile('C:/Users,
msydor/Documents/GitHub/Pi
first session...
taking a picture...
first session...
taking a picture...
first session...
taking a picture...
first session...
taking a picture...
first session...
taking a picture...
timer expired
second session...
taking a picture...
second session...
taking a picture...
second session...
taking a picture...
second session...
taking a picture...
second session...
taking a picture...
timer expired
no motion...
```

Demo - look at actual code