Introduction

In this lab, we will be exploring different functionalities of the Pi, and use them to gather data of the environment the SmartPlug is in. The Camera Module is an officially supported accessory for the Raspberry Pi. You will find that using a VNC will come in handy for working with the Camera Module. Additionally, the Pi has Bluetooth on board, so we are able to do some exciting things with this. We will also deal with posting semantic data to the API in order to provide access to the new data we are obtaining.

Pi Camera and basic Computer Vision

For the first part of this lab we will be using the Pi’s camera module to do some analysis on what the Pi is seeing. Specifically, we will attempt to detect (not recognize) faces as well as detect movement. This would be handy in a security device application!

In order to accomplish those tasks, we will be using the OpenCV library (www.opencv.org). This library provides most of the functions that will help us in achieving our goals.

First off, we need to install OpenCV. Run:

```
sudo apt-get install python-opencv
```

This will take a long time, so feel free to work on other portions of the lab while this task finishes.

In order to detect faces, we will be performing object detection using Haar feature-based cascade classifiers, with a cascade specific to faces. You can read OpenCV’s documentation on this topic at: http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html. The cascades they are referring to can be found at:
https://github.com/opencv/opencv/tree/master/data/haarcascades

Additionally, movement detection can be accomplished with background subtraction. The tutorial can be found here: http://docs.opencv.org/trunk/db/d5c/tutorial_py_bg_subtraction.html

Now, for the ***challenge.*** You must implement both algorithms to perform real time analysis on the video input. You can use whatever code you like, but feel free to use the template attached on the next page. Do not feel intimidated, this is not a Computer Vision lab, you are simply required to have a big-picture understanding of the process and implement the simple functions.

```python
import cv2
import sys

# Camera Imports
from picamera.array import PiRGBArray
from picamera import PiCamera

# Sleep function to wait on Camera
from time import sleep

# Get user supplied values
cascPath = sys.argv[1]

# Create the haar cascade
face_cascade = cv2.CascadeClassifier(cascPath)

# Capture Object
camera = PiCamera()
camera.resolution = (640,480)
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(640,480))

# Wait for the camera to start up
sleep(1.0)

for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    # Grab an image from the camera
    image = frame.array

    # Shrink it down
    height, width = image.shape[:2]
    image = cv2.resize(image,(width/2,height/2),interpolation=cv2.INTER_CUBIC)

    # Your code here...
    # Follow the OpenCV tutorial to use their detection algorithms!

    # Press 'q' to quit
    key = cv2.waitKey(1) & 0xFF
    rawCapture.truncate(0)
    if key == ord("q"):
        break
```

Semantic API calls

From time to time, you might want to post information that the API does not support. The API has the ability to record temperature, humidity, and light, but it can’t record something like “something moved” or “face detected”. In order to provide users with a bit more flexibility, we added a “semantic label” functionality. Semantic labels permit users to input any label they choose to push to the API, so a semantic label of “movement” would create a field for “movement” events. For example, to add a semantic event:

Add Semantic Data

POST	/api/v1/devices/{id}/semantic

ARGUMENTS

META Arguments	Type	Required	Default
data	String	yes	
semantic_label	String	yes	

HEADER

Header Arguments	Type	Required	Default
Token-Authorization	String	yes	

So every time you wish to add a “movement” event, you would set semantic_labe to “movement” and fill out the other information accordingly.

Create a main_camera.py file which is constantly looking for movement. In your routine, add a semantic_label POST request for movement. Every time the camera detects movement, post it to the API.

Here is the GET request as well:

Get Semantic Data

GET	/api/v1/devices/{id}/semantic/{count?}

This method retrieves all semantic events recorded by a named device.

PARAMETER

Parameter Arguments	Type	Required	Default
count	int	no	

HEADER

Header Arguments	Type	Required	Default
Token-Authorization	string	yes	

NOTE: This will return all semantic events, not just those under a specific label.

Bluetooth

We will now add Bluetooth functionality to our SmartPlug. We will be using the pybluez bluetooth module, which can be found at:
https://github.com/karulis/pybluez

The documentation for this module can be found inside the repository, under the docs/ folder.

Our implementation of the bluetooth tool will accomplish three things:

1. Scan the area for devices
2. Keep a list of all devices in the area
3. Update the list when devices leave or are turned off.

Setup

To get this part working, we need to install the following libraries through the terminal: bluez, bluetooth, and python-bluez, which lets you work with bluetooth from a python script. To do this we use the commands:

```
sudo apt-get update
sudo apt-get install bluetooth
sudo apt-get install bluez
sudo apt-get install python-bluez
```

Requirements

You must write a class named **BluetoothSearchHub.py** which contains the following functions:

- start_bluetooth_scan()
    - Scan the area for all devices, new and preexisting.
- ping_addresses()
    - Pings all registered devices, in order to determine whether they are active or not.
- check_devices()
    - Updates the device list.

The use of additional functions is highly recommended in order to accomplish complex tasks (also, don't forget your __init__!)

Turn in instructions

- Submit all your code to github.
- Submit a picture of your working code (face detection and movement)
- Email a PDF file with all pertinent analysis, your device's MAC Address, and your GitHub repository URL.
- Include Team name in Filename (TeamName_Lab5.pdf)
- State which API credentials were used