Marcus Giarrusso

Human Computer Interaction 530 - Smart Spaces

Introduction and Issue

- Original Thoughts..
  - pets left home alone & need to go outside
  - leave door open / doggie-door?

- Why not make it more functional...
  - friends, family, deliveries, etc.

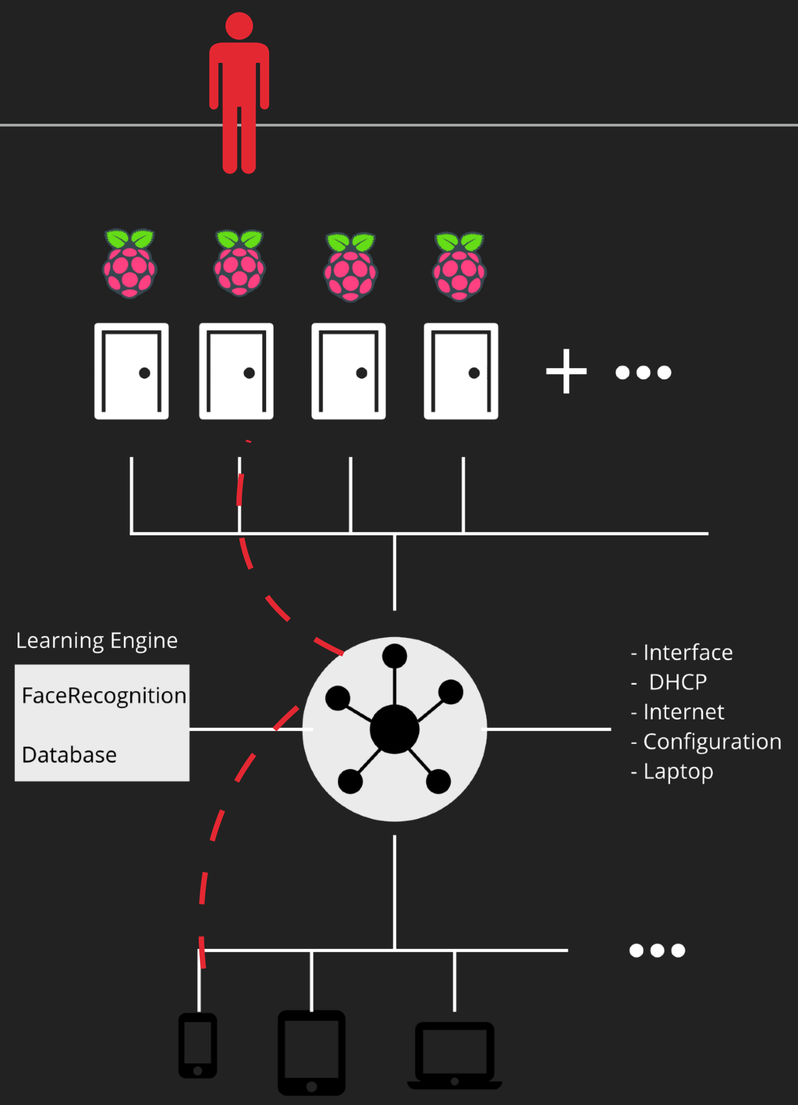
How can I Potentially Solve it?

- Build a system to allow homeowners to open door when not home
  - owner notified, sees person / pet at door & can let them out / in.

- Why not try to automate it?
  - facial recognition
  - but also allow homeowner to see / be notified if wanted

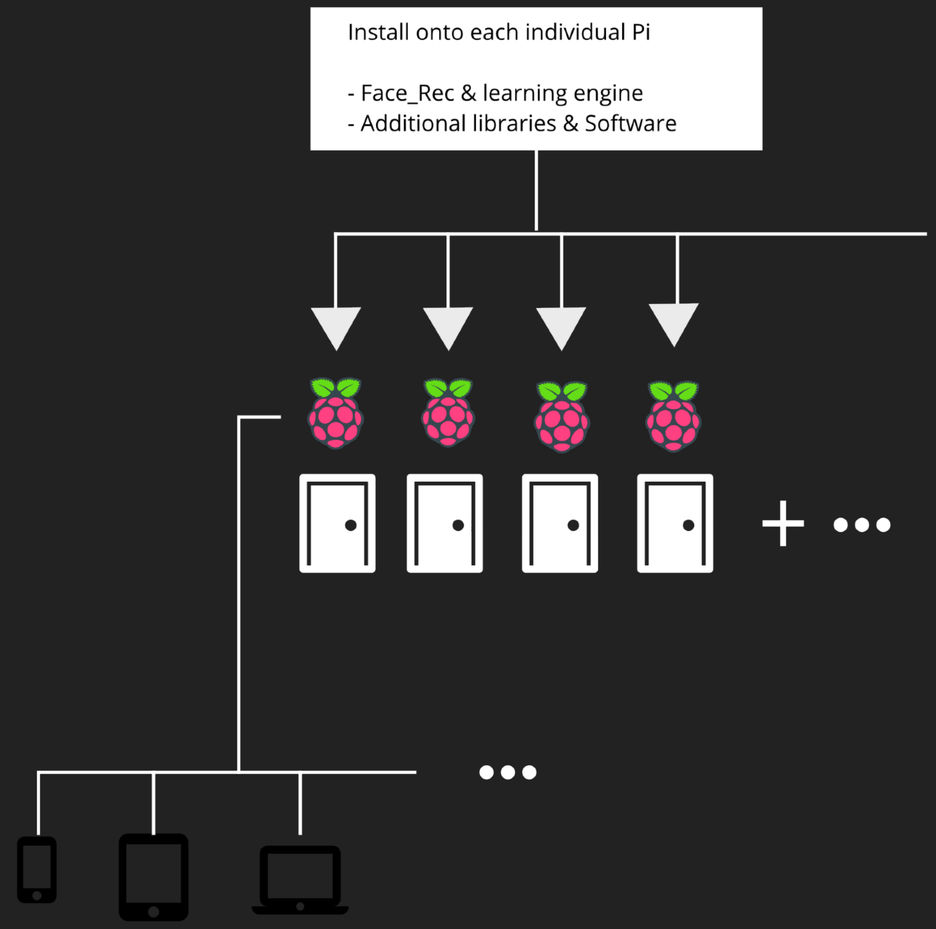
One Possible Solution

- Central ‘Hub’
- Pi / Camera / Actuator at each door
  - Pi detects motion
  - relays camera stream to hub
  - hub processes face & allow / deny access
  - hub notifies homeowner & sends them stream
  - owner can decide to override access



Another Possible Solution

- No Hub
- Pi / Camera / Actuator at each Door
- install software onto each
  - pi detects motion
  - processes face, allow / deny access
  - owner notified & sent stream
  - owner can override system

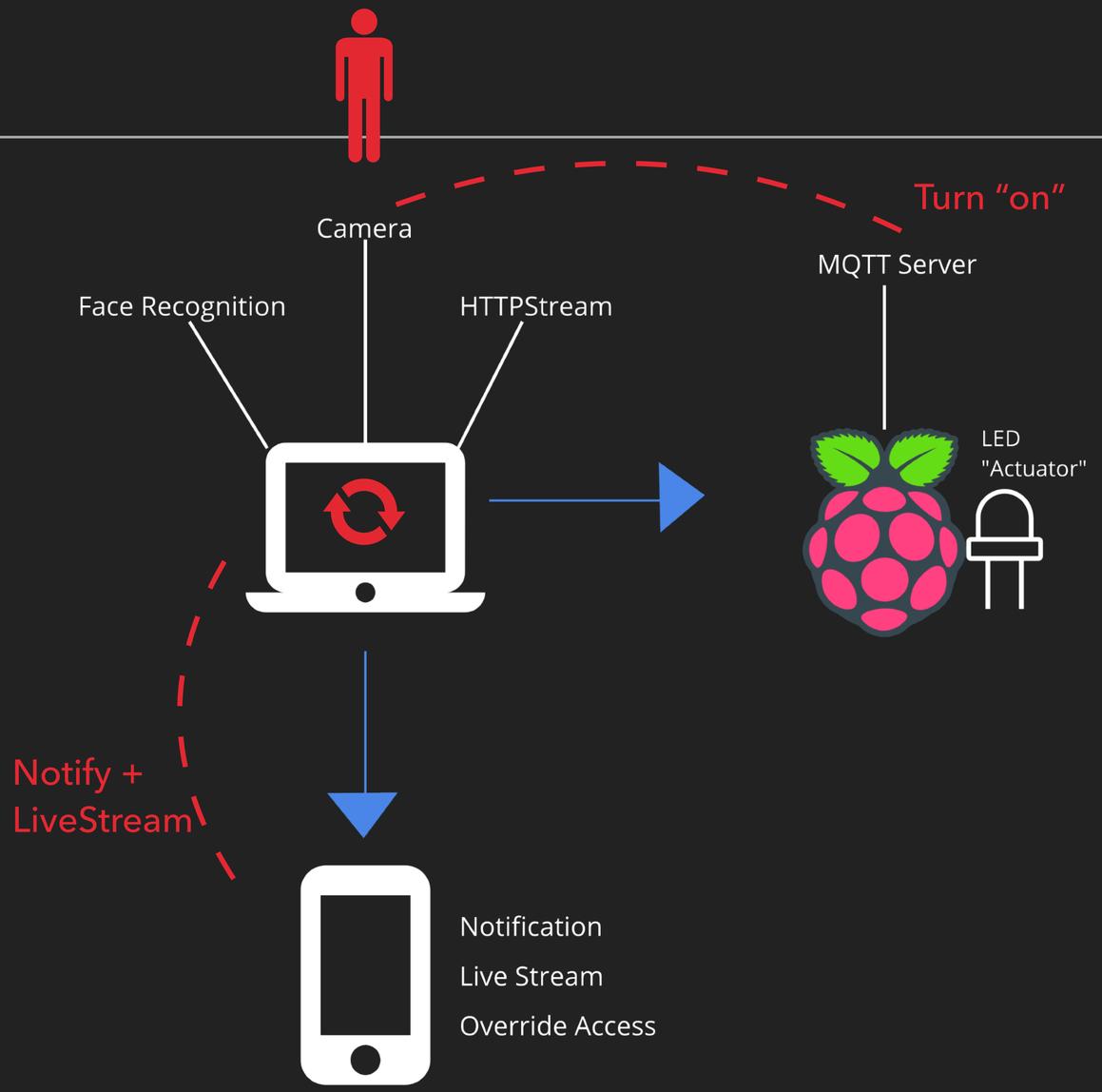


Problems

- However, ran into some obstacles..
  - Time / Resources
    - limited time
    - money
  - Limited Knowledge
    - first time using most pieces involved in project
  - FacialRec library not working correctly with pi

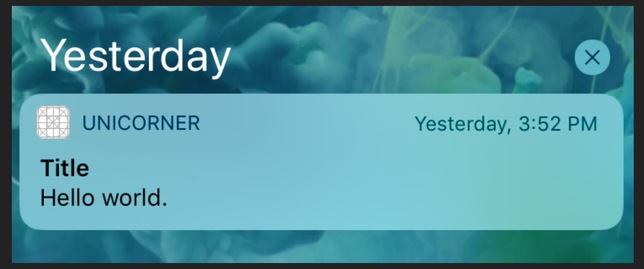
Final Approach / Outcome

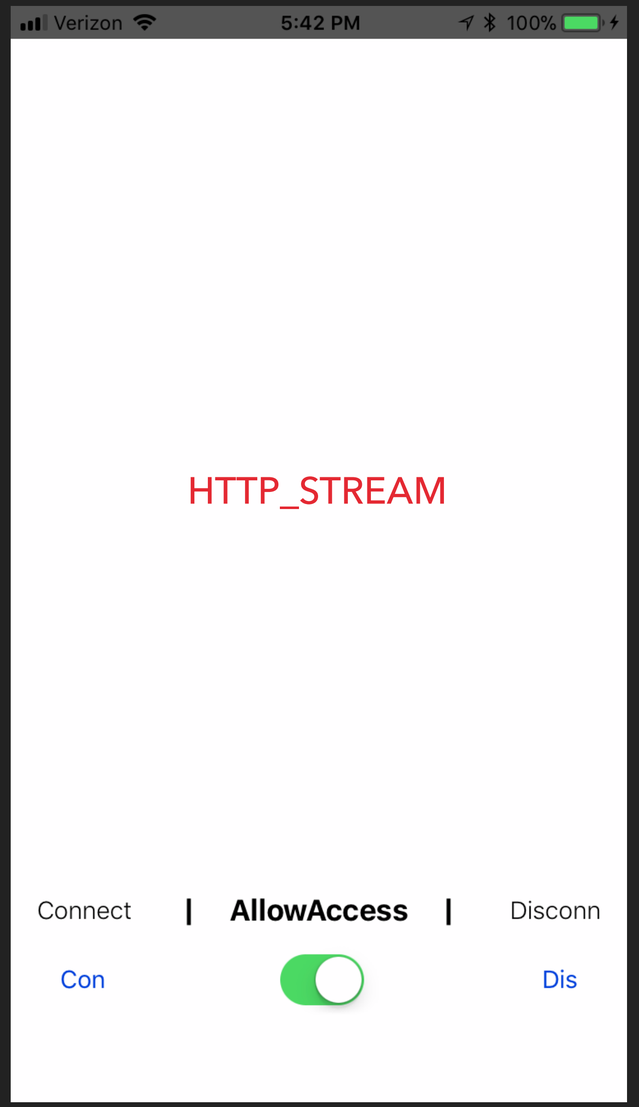
- **What I ended up with...**
  - **FaceRec process on laptop**
  - **sends msg to the pi gpio pin**
  - **sends notification to iPhone**
  - **live http stream on iPhone**
  - **user can override access (gpio/light)**



The Pieces

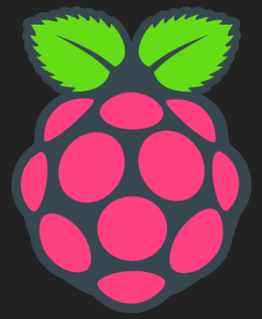
- **Software**
  - **iPhone**
    - **XCode & Swift 4**
    - **CocoaMQTT**
  - **Pycharm & python 3.6**
    - **Face_Recognition Library (lib credit: (github) ageitgey)**
    - **httpStreaming with flask (lib credit: miguelGrinberg)**
    - **paho MQTT, openCV, dlib**

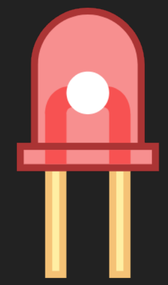


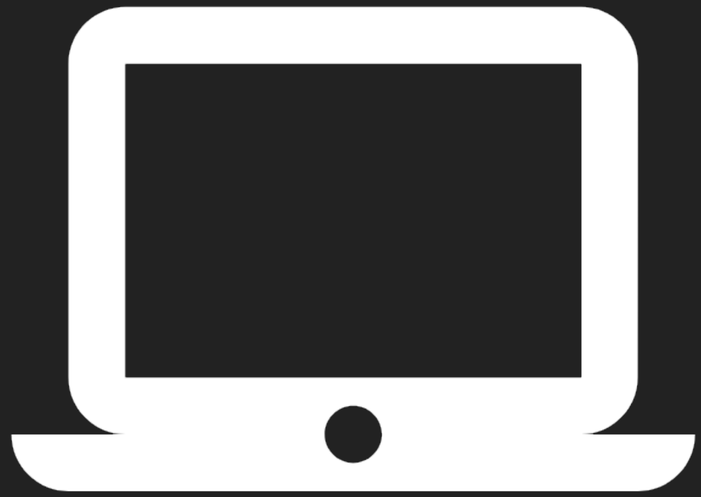


The Pieces

- Hardware
  - iPhone
  - Raspberry pi 3 B
  - Camera Module v2
  - LED, breadboard, gpioPins







The Pieces

Notifications

```python
from pushjack import APNSSandboxClient


def send_notification():
    client = APNSSandboxClient(certificate='Path to Apple iOS Push Service Notification Certificate',
                               default_error_timeout=10,
                               default_expiration_offset=2592000,
                               default_batch_size=100,
                               default_retries=5)
    token = 'DeviceToken#'
    alert = 'Hello world.'

    # Send to a single device (keyword args optional)
    res = client.send(token,
                      alert,
                      badge=1,
                      sound='default',
                      content_available=True,
                      title='Title')

    # List of all tokens sent.
    print(res.tokens)
    # List of errors as APNSServerError objects
    print(res.errors)
    # Dict mapping errors as token => APNSServerError object.
    print(res.token_errors)
```

Http/WebView

```swift
    @IBOutlet weak var webView: UIWebView!

    override func viewDidLoad() {
        super.viewDidLoad()
        view.backgroundColor = UIColor.darkGray
        // **** Set IP Address and port of your RasPi or local http stream ("http://xxx.xxx.x.xxx:xxxx")
        let stream_uri = "### Enter Stream_Url ###"

        // Delegate and Load
        webView.delegate = self
        webView.loadRequest(NSURLRequest(url: NSURL(string: stream_uri )! as URL) as URLRequest)

        // Change background color of webView or reduce size of webView in main.storyboard
        webView.backgroundColor = UIColor.clear
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
```

MQTT

```swift
import UIKit
import CocoaMQTT

class ViewController: UIViewController, UIWebViewDelegate {
    // **** Set IP Address
    let mqttClient = CocoaMQTT(clientID: "iOS Device", host: "### IPAdress ###", port: 1883)

    // Button to connect to mqtt
    @IBAction func connectButton(_ sender: UIButton) {
        mqttClient.connect()
        print("Connected..")
    }

    // Button to disconnect from mqtt
    @IBAction func disconnectButton(_ sender: UIButton) {
        mqttClient.disconnect()
        print("Disconnected..")
    }

    // Ability to toggle gpio pin/led light on/off
    @IBAction func gpio40_Switch(_ sender: UISwitch) {
        if sender.isOn{
            mqttClient.publish("rpi/gpio", withString: "on")
            print("Switch_ON")
        }
        else{
            mqttClient.publish("rpi/gpio", withString: "off")
            print("Switch_OFF")
        }
    }
```

The Pieces

```python
# Get a reference to webcam
video_capture = cv2.VideoCapture(0)

# load sample picture & learn how to recognize it
sample_image = face_recognition.load_image_file("jobs.jpg")
sample_image_encoding = face_recognition.face_encodings(sample_image)[0]

# load 2nd sample image & learn how to recognize it
sample_2_image = face_recognition.load_image_file("zucks.jpg")
sample_2_image_encoding = face_recognition.face_encodings(sample_2_image)[0]

# Create arrays of known face encodings & their names
known_face_encodings = [
    sample_image_encoding,
    sample_2_image_encoding
]

known_face_names = [
    "jobs",
    "zucks"
]

# Optional if time
unknown_face_names = []
blocked_face_names = []

# Initialize some variables
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True
```

FaceRec 1 of 1

```python
    if process_this_frame:
        # Find all faces & encodings in current video frame
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
        face_names = []
        for face_encoding in face_encodings:
            # See if face matches known face
            matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
            name = "Unknown..."
            # If match made in known_face_encodings, just use first one
            if True in matches:
                first_match_index = matches.index(True)
                name = known_face_names[first_match_index]
                # if faceRecognized equals specified person, do action
                if counter > 60:
                    if not has_been_called:
                        if name == "known_face_name":
                            print('known_face_name')
                            has_been_called = True
                            # Send Apple Push Notification
                            send_notification()
                            # Connect to MQTT
                            client = mqtt.Client()
                            # Specify IP Address, port, etc.
                            client.connect("IpAddress", 1883, 60)
                            # Publish actuator action
                            client.publish("rpi/gpio", "on")
                            client.disconnect()
                            # global open_stream
                            # Open up http stream (correct/best way to do this??)
                            open_stream = subprocess.Popen(
                                ['python3', 'Path to Stream_2/main.py'],
                                shell=False)
```

FaceRec 1 of 2

```python
    # Display results
    for (top, right, bottom, left), name in zip(face_locations, face_names):
        # Scale back up face locations since the frame we detected in was scaled 1/4 size
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4
        # Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
        # Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)
    # Display resulting image
    cv2.imshow('Video', frame)
    # Counter Increment, name has_been_called?, count to slow recognition time/close application?
    counter += 1
    # Hit 'q' on keyboard to quit application
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Release handle to the webcam
video_capture.release()
cv2.destroyAllWindows()
# Terminate SubProcess (nice way?)
pid = open_stream.pid
open_stream.terminate()
try:
    os.kill(pid, 0)
    open_stream.kill()
    print('Force Killed Process')
except OSError as e:
    print('Terminated Gently...')
```

FaceRec 3 of 3

Future Work

- With more time & resources
  - add user interface & Database
    - allow homeowner to add new person immediately
    - or add them to a ‘blocked’ list
    - build pleasing UI
    - maybe implement one of the more ideal scenarios
    - and any other small features that add to its usability

Conclusion

- Demonstrates class topics
  - smart spaces
  - internet of things
  - automation
- Demo
- Questions…?

THANK YOU !