# Software Formalization

**Year:** 2019    **Semester:** Spring      **Team:** 2      **Project:** Guard DAWG System
**Creation Date:** February 17, 2019            **Last Modified:** February 17, 2019
**Author:**  Yash Nain                          **Email:** ynain@purdue.edu

**1.0 Utilization of Third Party Software**

| Name | License | Description | Use |
|---|---|---|---|
| [1] MSP432/HC-05 UART Driver | - | Module to connect MSP432 to HC-05 through UART | Reference code |
| [2] Face Recognition | MIT License | Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. | Map faces to verified users |
| [3] DLib | Boost Software License | Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, | Used by Face Recognition to implement algorithm |

| | | embedded devices, mobile phones, and large high performance computing environments. Dlib's open source licensing allows you to use it in any application, free of charge. | |
|---|---|---|---|
| [4] OpenCV | BSD License | A cross-platform library focusing on image processing, video analysis, and face and object detection for real-time computer vision applications | Used by face_recognition and |
| [5] NumPy | NumPy Developers | A python library built in C/C++, offering fast computational capabilities for multi-dimensional arrays and matrices, with a variety of optimized mathematical operations found useful in a wide variety of applications. | Used by DLib and OpenCV for various calculations, for speed and efficiency that Python alone couldn't reach. |
| [6] PyBluez | GNU General Public License | A python extension module for Bluetooth built to allow python applications to use Bluetooth resources in GNU/Linux and Windows XP | Used by the Raspberry Pi to communicate with the Bluetooth chip connected to the MSP432 |
| [7] PiCamera | BSD License | A pure Python interface to the Raspberry Pi camera | The image streaming process |

| [8] Raspbian | Oracle Binary Code License Agreement | An operating system optimized for the Raspberry Pi hardware | Image streaming, Bluetooth with the Micro |
|---|---|---|---|

**2.0 Description of Software Components**

The software for the lock can be divided into two major functional blocks: the low-level embedded software and the Raspberry Pi/Facial Recognition high-level server code. The embedded software is responsible for initializing the lock/unlock sequence, reading in data from UART from the Bluetooth Module, reading in number presses from the keypad, and sending signals to the power electronics to dog/undog the motor though a timer interface. The Raspberry Pi reads in the image data from the Pi Camera, and sends it to a server through WiFi, which detects the faces in the image, and runs the largest one through the Facial Recognition algorithm to determine whether or not it corresponds to a verified user.

**Embedded:**

**2.1** Code Structure

The software on the MSP432 includes data structures for receiving information from peripherals, interrupts to handle events, and functions to initialize system components. The micro will be in a defined state and will only leave that state when an interrupt is triggered such as the door opening or receiving a message from the Raspberry Pi over UART. Having the system be interrupt driven allows the micro to be efficient and . The following are the main software components. Smaller components like functions for the Hall effect sensor and LEDs are not listed as they require much less complexity.

**2.2** UART
- data packet structure: Data packets will be sent a byte at a time allowing for fast transactions and minimal latency. A single byte can represent 256 decoded responses and should accommodate a defined language between the micro and the Raspberry Pi. Each byte the micro receives will be decoded and contain information such as the Raspberry Pi is configured and ready, or a registered face was successfully detected.
- uart_send(char*): This function will attempt to send a request to the Raspberry Pi. There is an internal buffer for data requests that the input character is added to and will be sent out when all other requests have been handled.
- uart_recieve(): This function is an interrupt and will receive a byte packet over UART which will be decoded and used to determine the next state of the micro.

**2.3** Keypad
- keypad_pressed(): This is an interrupt that is triggered when a button on the keypad is pressed. The function will decode the keypad by polling GPIO pins to see exactly which row and column the button was pressed on. Once decoded, the micro will continue

- special_key_pressed(): The button to begin facial recognition was pressed and the micro needs to send information to the Raspberry Pi over UART.
-

## 2.4 Timer
- timer(): A timer interrupt function will be used to assess when to reset states in our system. This occurs when the Raspberry Pi becomes unresponsive, or when no one presses the keypad after initially starting to enter a code.

**Bluetooth/Server:**

## 2.5 PiCamera
- The PiCamera is the name of both a hardware component, as well as the Python code that comes with it for easy interfacing. The software allows for the camera to be set to a specific resolution and framerate, and then frames can be captured through a capture_sequence() function call that streams frames to a passed receiver, which for us is the agent responsible for passing the frames to the server over WiFi.

## 2.6 Bluetooth Socket
- The bluetooth socket connection is made using the PyBluez python library, which aims to give a python interface with the Bluetooth functionalities of Linux machines and Windows XP. Since Raspbian runs a Linux terminal, it also allows the connection to the Raspberry Pi's bluetooth capabilities as well.
- On a socket connection, a discover_devices() function call can be made to dynamically discover a bluetooth device by name if the MAC address isn't already known.
- Whether pre-encoded or dynamically found, the bluetooth connection allows for a simple send() function call to send a text string byte by byte to the connected device, which is the slave HC-05 in our current design.
- To receive data, a recv() call is made with an integer passed for the maximum amount of bytes to accept from the party on the other end of the socket. The resulting message is a bytes string, which can be decoded through a UTF-8 format to get the original string message, via Python's String.decode('utf-8') functionality.

**2.7** Facial Recognition/Image Transformation
- The facial recognition algorithm is made through the face_recognition [2] open source library meant to provide a simple interface for facial recognition to python applications.
- Once an image is loaded into a numpy array, a simple call to face_locations() provides the bounding box for the face and an easy way to calculate the area of the picture taken up by the chosen face. The (bottom - top) * (right - left) area calculation is what we use to estimate the largest face in the image and designate them as the current user for the process.
- Once the face is selected, the face_encodings() function call is made to pull out the 128 feature vector for that respective face, and it is then compared to all the known, pre-found features of all registered users to see how many images of each known user matches each frame, and the resulting matches are summed and normalized.
- To compare one frame's feature vector against the collective list of features for one known user, the compare_faces() function call compares a single feature vector against a list of known features in an optimized manner. The result is an array of True/False values, one per known vector, to state whether there was a match by a predetermined threshold or not. The True values are summed and normalized for each respective user, and then the overall normalized result is passed back to the bigger application to send back to the Raspberry Pi as a JSON message.

**3.0 Testing Plan**

Each of the modules will need their own stress-testing, since they involve using several external resources. Below lists how we plan on testing our various software components in our system. A numerical component has been assigned to each indicating the level of importance for successfully verification with 1 being the utmost importance, and 5 being the least.

| Raspberry Pi |
| --- |
| **Facial Size Recognition (3)** |
| 1. Have 1 verified person in frame, stationary<br>2. Unable to recognize verified user (1 person in frame)<br>3. Have no face in frame<br>4. Have 2 people, 1 verified, in frame, stationary, standing side-by-side<br>5. Have 2 verified people in frame, stationary, with one in the foreground and one in background<br>6. Have 3+ people, 1 verified, standing in frame, stationary<br>7. Unable to recognize verified user (2 people in frame)<br>8. Have 1 verified person in frame, in motion<br>9. Have 2 people, 1 verified, in frame, in motion, with one in the foreground and one in the background |

**Facial Feature Encoding (2)**

1. Recognize an authorized user
2. Recognize an unauthorized user
3. Recognize "over-authorized" user? (matches 2+ profiles, may need twins/close siblings)

**Bluetooth Functionality (1)**

1. Send packet to microcontroller
2. Receive packet from microcontroller
3. Stream packets to microcontroller
4. Receive stream packets from microcontroller
5. Test packet overflow

## Microcontroller

**High-Level Driver Functionality (4)**

1. Connect to Raspberry Pi through UART on power-up, Pi ON
2. Connect to Raspberry Pi through UART on power-up, Pi OFF
3. Dog/Undog motor

**UART Read/Parse (1)**

1. Read a character from UART
2. Read a stream of characters from UART
3. Split stream of characters from UART with end of packet character

**Timer (3)**

1. Enable timer on startup
2. When enabling lock, set timer to check for initialization button press

**Keypad (2)**

1. Press and hold button, test for button debouncing
2. Press several buttons in series
3. Press button in rapid succession, test for button debouncing

**Override Buttons (3)**

1. Test lock/unlock switch override functionality when idle
2. Test lock/unlock switch override functionality when reading from keypad
3. Test lock/unlock switch override functionality when reading packet from UART

**Hall-Effect Sensor/Door Lock (3)**

1. Determine threshold of signal for opened/closed door
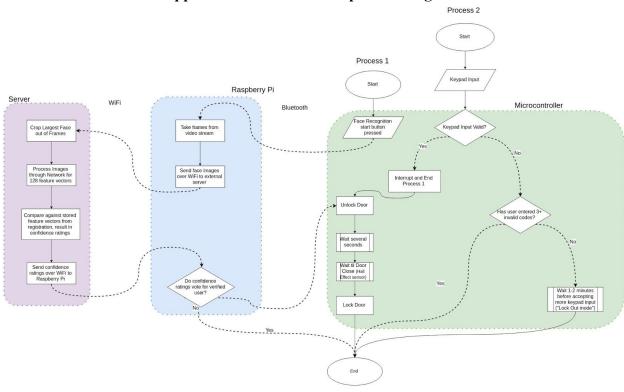2. Test closed/open
3. Test almost closed

## HC-05/06 Bluetooth Module

**UART Interface**

1. Send characters through TX
2. Read characters through RX
3. Unable to connect to microcontroller
4. Unable to connect to Raspberry Pi
5. Invalid UART command from microcontroller

**4.0 Sources Cited:**

[1] Martinez, A. (2019). *MSP432-HC-05*. [online] GitHub. Available at:
https://github.com/amartinezacosta/MSP432-HC-05 [Accessed 17 Feb. 2019].

[2] Geitgey, A. (2019). *face_recognition*. [online] PyPI. Available at:
https://pypi.org/project/face_recognition/ [Accessed 17 Feb. 2019].

[3] King, D. (2019). *Dlib-ml: A Machine Learning Toolkit*. [online] Dlib.net. Available at:
http://dlib.net/ [Accessed 17 Feb. 2019].

[4] Bradski, G. (2019). *The OpenCV library*. [online] Opencv.org. Available at:
https://opencv.org/ [Accessed 17 Feb. 2019].

[5] Oliphant, T. (2019). *A guide to NumPy*. [online] Numpy.org. Available at:
http://www.numpy.org/ [Accessed 17 Feb. 2019].

[6] Peters, T. and Govostes, R. (2019). *PyBluez*. [online] GitHub. Available at:
https://github.com/karulis/pybluez [Accessed 23 Feb. 2019].

[7] Jones, D. (2019). *Picamera 1.13 Documentation*. [online] Picamera.readthedocs.io. Available
at: https://picamera.readthedocs.io/en/release-1.13/index.html [Accessed 17 Feb. 2019].

[8] Thompson, M. and Green, P. (2019). *Raspbian*. [online] Raspbian.org. Available at:
https://www.raspbian.org/ [Accessed 17 Feb. 2019].

**Appendix 1: Software Component Diagram**

Process 2

Process 1

Server                    WiFi                    Raspberry Pi                    Bluetooth                    Microcontroller

Start

Start

Keypad Input

Crop Largest Face out of Frames

Take frames from video stream

Face Recognition start button pressed

Keypad Input Valid?

Process Images through Network for 128 feature vectors

Send face images over WiFi to external server

Interrupt and End Process 1

Has user entered 3+ invalid codes?

Compare against stored feature vectors from registration, result in confidence ratings

Unlock Door

Send confidence ratings over WiFi to Raspberry Pi

Wait several seconds

Wait til Door Close (Hall Effect sensor)

Wait 1-2 minutes before accepting more keypad input ("Lock Out mode")

Do confidence ratings vote for verified user?

Lock Door

Yes

No

Yes

No

Yes

No

End