Welcome to the EcoFlow Design Document!

System Overview

The EcoFlow smart HVAC system is designed to provide seamless control from our software, through our microcontrollers and finally outputted through HVAC hardware to our users. The system is broken down into two main components: Raspberry Pi and Arduino. Information regarding calibrating temperature and thermal image data are provided to the Raspberry Pi from sensors and a thermal imaging camera. Image processing takes place at a server, and is brought back to the Raspberry pi. With this data, the Raspberry Pi will send commands to the Arduino, and the Arduino will adjust servos that are attached to HVAC hardware accordingly.

Objective

To provide a smart, ecological solution for thermal comfort with reduced energy expenditure, providing an alternative to conventional HVAC units.

System Architecture

System Hardware:

Conventional HVAC parts:

- Honeywell AC Unit
- Wyes (6” to 4” and 4” to 3”)
- HVAC Tubing (6” from AC source and 3” for output)
- Damper (3”)

Computers, Microcontrollers and Associated pieces:

- Arduino UNO R3
- Raspberry Pi 3
- Servo Motors (Futaba S3003)
- Thermal Sensors (DS18B20)
- Flir Lepton
- Server

Build

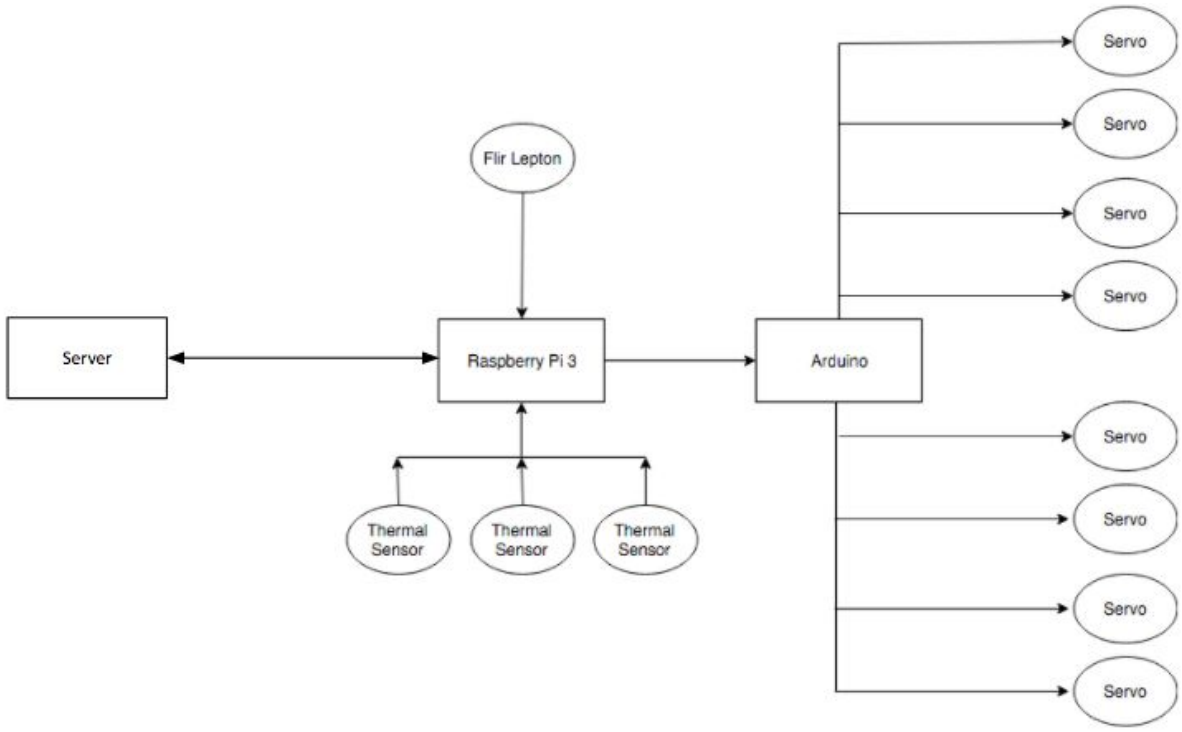
- Desk
- Chair

Eco-Flow

System Software

- Arduino
  - Arduino IDE
- Raspberry Pi
  - BCM2835
  - Opencv 3.3.1
  - LePi
- Server
  - OpenCV 3.3.1
  - Python3
    - numpy >= 1.13.3

Component Design



- Flir Lepton:
  - It receives a thermal image of 120x160 frame.
- Server
  - This server analyzes the thermal image with OpenCV and returns the analysis.
- Thermal Sensors
  - These will serve to calibrate the Flir Lepton, attached to the Raspberry Pi. This will ensure that the Flir Lepton will associate thermal images produced with the room temperature around the user.

- Raspberry Pi
    - Gathers data provided from Flir Lepton, Thermal Sensors and Server and generates 2 byte commands to send to the Arduino via serial.
- Arduino
    - Receives commands from the Raspberry Pi and makes adjustments to servos connected to respective dampers. The first byte corresponds to the angle in which the servo should adjust, while the second refers to which servo(s) should adjust their angle.
- Servo
    - Adjusts to the correct angle based on the command sent to the Arduino. This is connected to an HVAC Damper. Based on the angle of the damper, the user will receive more heat or cooled air, as well as have adjusted angles to tend to the user's position.

Human Interface Design

- The user will input their preferred temperature into a system-controlled laptop. The system will generate the appropriate response by providing more heat or cooled air to the user. The system will continue to check the room temperature and position of the user and automatically adjust accordingly.

Parts List

Product	Number Requested	Price	Website
Breadboard	1	$5.95	https://www.sparkfun.com/products/12615
Temperature Sensor	3	$11.85	https://www.sparkfun.com/products/245
Arduino R3	1	$24.95	https://www.sparkfun.com/products/11021
Male - Male Jumper Wires	1	$7.42	http://a.co/8O5O8BY
Rasberry Pi 3	1	$39.95	https://www.sparkfun.com/products/13825
Flir Lepton	1	$199.00	https://www.digikey.com/short/q77cjf
Breakout Board	1	$39.99	https://www.digikey.com/short/q77cj9
SD Card	1	$13.99	http://a.co/4sMN6Of
Power Supply	1	$9.99	http://a.co/59bIni4

Female-Female Jumper Wires	1	$4.99	http://a.co/8WegICA
Male - Female Jumper Wires	1	$4.99	http://a.co/iWN4zJB
Pi Cobbler	1	$7.95	https://www.adafruit.com/product/2028
Trade Size 3 Female x Male Aluminum Airflow Damper for Standard Duct	8	$321.76	https://www.mcmaster.com/#1773k41/=19p25vz
6” duct hose	4 x 5ft	$188.20	https://www.mcmaster.com/#5488K66
"3"" duct hose"	8 x 6ft	$226.56	https://www.mcmaster.com/#5488K61
6-4-4 Wye Reducer	2	$57.94	https://www.mcmaster.com/#1766K868
4-3-3 Wye Reducer	4	$49.24	https://www.mcmaster.com/#1766K861
"2 13/16"" to 3 3/4"" Clamps"	6 packs of 5	$51.30	https://www.mcmaster.com/#54155k25/=19rt8fv
"3 5/8"" to 6 1/2 Clamps"	2 packs of 5	$18.62	https://www.mcmaster.com/#5416K37
Servos with feedback	8	$98.96	http://a.co/0aOcfk0
Adafruit 16-Channel 12-bit PWM/Servo Shield - I2C interface	1	$17.50	https://www.adafruit.com/product/1411

Shield stacking headers for Arduino (R3 Compatible)	6	$11.70	https://www.adafruit.com/product/85
3x4 Right Angle Male Header - 4 pack	6	$17.70	https://www.adafruit.com/product/816
10pcs Male & 10pcs Female DC Power Jack Adapter Connector Plug	1	$2.00	https://www.adafruit.com/product/368
5V 10A Switching Power Supply	1	$25.00	https://www.adafruit.com/product/658
Thermal Curtains	1	$39.98	http://a.co/fOdsEx6
Curtain Rod	1	$49.95	http://a.co/fOdsEx6
AmazonBasics Chair	1	$64.99	http://a.co/5rlLSMG
Desk	1	$249.99	http://a.co/3pvg4xp
Pololu Universal Aluminum Mounting Hub for 3mm Shaft, #2-56 Holes (2-Pack)	6	$35.70	https://www.pololu.com/product/1079/pictures
Gorilla 4200101-2 Epoxy (2 Pack) .85 oz, Clear	1	$11.15	http://a.co/ax55PS8

Machine Screw: #2-56, 7/16″ Length Phillips (25-pack)	2	$1.98	https://www.pololu.com/product/1957
Machine Hex Nut: #2-56 (25-pack)	2	$1.98	https://www.pololu.com/product/1067

Setting up an Arduino

This page will describe how to connect your computer to an Arduino UNO R3 Microcontroller.

Hardware Requirements

- An Arduino R3 Microcontroller
- USB Cable - Standard A to B
- Windows/Mac/Linux Machine

Software Requirements

- Arduino IDE

Connecting the Arduino

Connect the Arduino via the USB cable to the computer.

Initialize the Arduino

- Open the Arduino IDE Software
- Ensure that the Arduino is connected to the computer
- Go to the Tools Tab and select Board -> Arduino/Genuino Uno
- Go to the Tools Tab and select Port -> *Associated Port with Arduino connected*
- Your Arduino should be ready to receive code uploaded from the user.

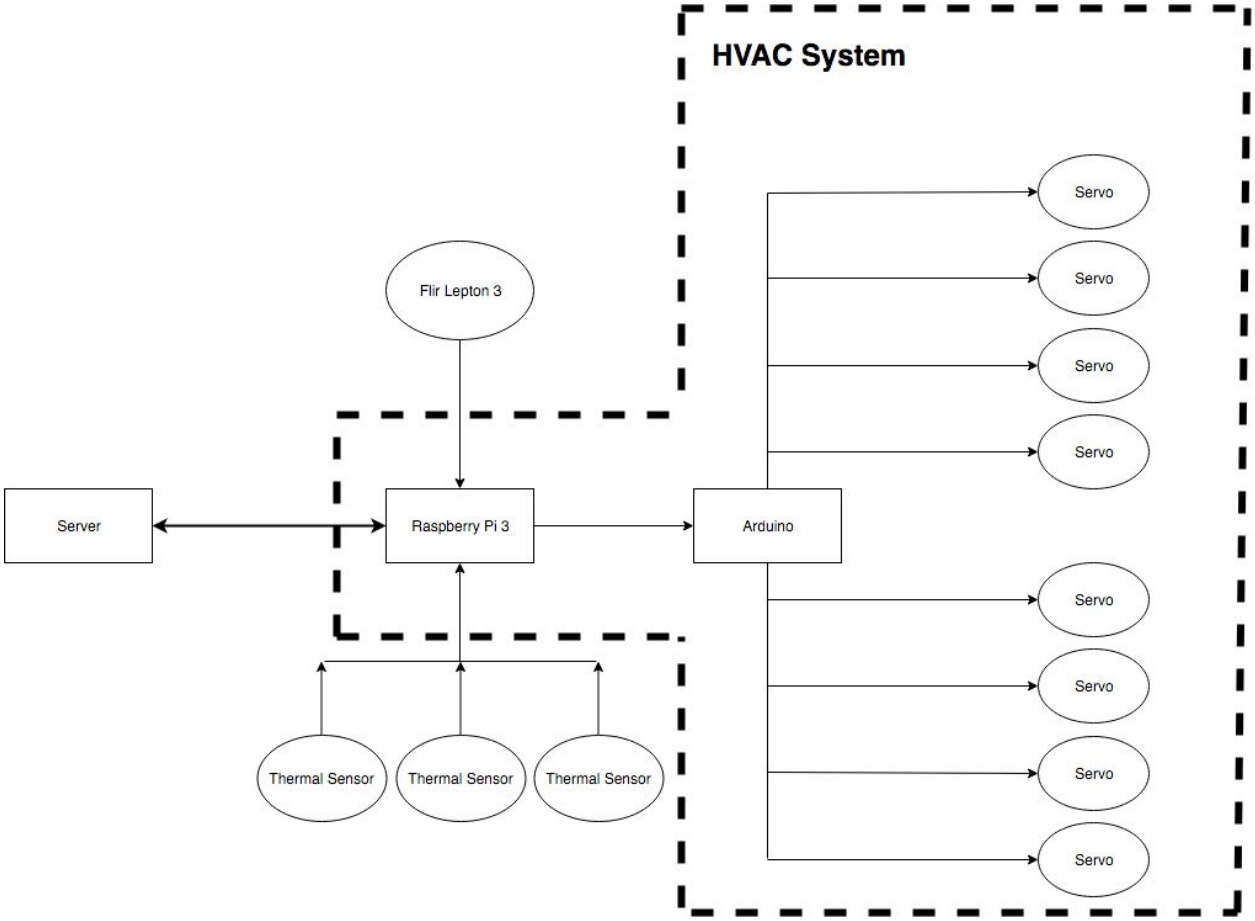
Eco-Flow

HVAC System

Overview

This component is meant to output commands sent over by the Thermal Imaging Pipeline using the PID algorithm, as well as generate responses to the Raspberry Pi over serial. This document will discuss how we execute commands received and sent, as well as regulate desired temperature in the HVAC system.

Figure: HVAC System Diagram



Communication

Arduino Serial Protocol

This message is a 2 byte response to the command sent from the Raspberry Pi in hexadecimal form:

```
+----------------+
|Boolean Response| 1 Byte
+----------------+
|   Error Code   | 1 Byte
+----------------+
```

How Communication Works

The first byte will refer to a boolean value- whether or not the command could be executed:

```
x00: No Error - Successful angle command.
x01: Angle Error - Command Failed.
x02: Large Angle Error - Angle requested is too large.
```

The second byte would refer to an error code, if any:

```
x00: No Error - Successful servo command.
x01: Servo Error - Command Failed.
x02: Servo Error - Servo Address not found.
```

Serial Response Protocol

This message is a 2 byte command sent from the Arduino to servo motors, in hexadecimal form:

```
+----------------+
|Angular Command | 1 Byte
+----------------+
| Servo Address  | 1 Byte
+----------------+
```

How Communication Works

The first byte will refer to the angular position in which the Servos must be in:

```
x00...xB3: 0-179 Degrees
```

The second byte refers to the associated Servos that must turn:

```
x00: Servo 1
x01: Servo 2
x02: Servo 3
x03: Servo 4
x04: Servo 5
x05: Servo 6
x06: Servo 7
x07: Servo 8
```

Temperature Regulation

- We make use of Proportional-Integral-Derivative Controllers (PID controllers) to maintain the temperature requested by the user for the HVAC system.

LePi Helper

recvall(s=*Socket*, n=*Number of bytes*)

Receives a response from the socket of length n.

- Returns:
  - The response in a byte array
  - None if the message is not of length n

Usage:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect("IP ADDRESS", 5995))
Message(s).send_frame_request()

frame = recvall(s, 38410)
```

create_socket(ip_address=*IP Address String*)

Creates a socket to the IP address on port 5995 with a timeout of 10 seconds.

- Exceptions:
  - timeout: This occurs when the client connection to the server times out.
- Returns:
  - Socket

Usage:

```
s = create_socket("IP ADDRESS")
```

LePi Image

normalize_frame(frame=*Byte array of length 38400*)

Does a local normalization and converts the frames to a 0-255 scale.

- Returns:
  - 2D numpy array of length 120 and width 160.

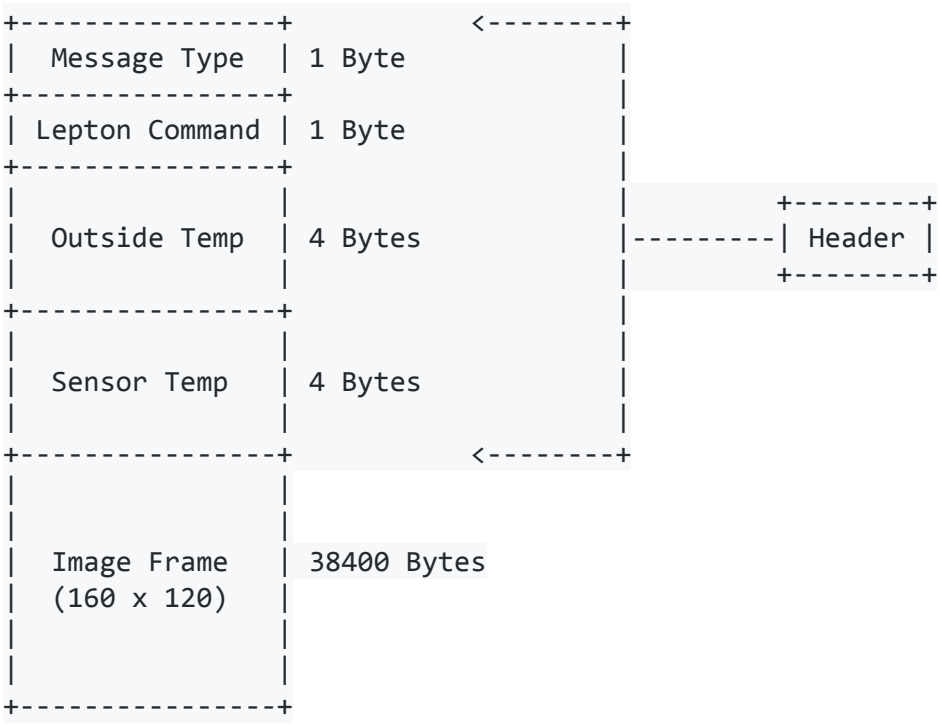
Usage

```
# Connect to the Raspberry Pi Application
s = create_socket("Raspberry Pi IP Address")

# Request a frame
Message(s).send_frame_request()
frame = Message(s).receive_response()

# Receive the response
msg = Message()
msg.open_response(frame)

# Normalize the image data
norm_frame = normalize_frame(msg.data)
```

LePi Server

Objects

These objects are meant to make reading data from our stream socket easier to read.



class ServerResponse

ServerResponse is the response given by the server when issuing a command.

Name	Value
FRAME_READY	0
NO_FRAME	1
I2C_SUCCEED	2
I2C_FAILED	3
RESEND	4

Example

```
# Client Request Above

resp = helper.recvall(self.socket, 38410)

if(resp[0] == ClientRequestType.I2C_CMD):
    print("I2C CMD")
if(resp[1] == ServerResponse.I2C_SUCCEED):
    print("I2C Succeeded")
```

Assuming an I2C command was sent we receive the data from the socket.

```
resp = helper.recvall(self.socket, 38410)
```

Then we output the Message Type and Lepton Command When receiving a response from the server, the Message Type is the same one sent by the client.

```
if(resp[0] == ClientRequestType.I2C_CMD):
    print("I2C CMD")
```

Then the Lepton Command is replaced by server response. In this case I2C commands, can have 3 responses. I2C_SUCCEED, I2C_FAILED, and RESEND

```
if(resp[1] == ServerResponse.I2C_SUCCEED):
    print("I2C Succeeded")
```

class ClientRequestType

This is the type of command you will send. If you want a frame request, you only need to set the Message Type. If you want to send an I2C command, you need to set both the message type and I2C command you want to send.

Name	Value
FRAME_REQUEST	0
I2C_CMD	1
UNKNOWN_MSG	2

Example

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect("IP ADDRESS", 5995))

frame_request = bytearray()
frame_request.append(ClientRequestType.FRAME_REQUEST)

s.send(frame_request)
```

When asking for a frame request, you only need to send the first byte. The server only looks at the first byte.

```
frame_request = bytearray()
frame_request.append(ClientRequestType.FRAME_REQUEST)
```

class I2CCommand

This is the I2C Command you will send to the server.

Name	Value
RESET	0
REBOOT	1
FCC	2
SENSOR_TEMP_K	3
SHUTTER_OPEN	4
SHUTTER_CLOSE	5
VOID	6

Usage:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect("IP ADDRESS", 5995))

i2c_request = bytearray()
i2c.append(ClientRequestType.I2C_CMD)
i2c.append(I2CCommand.FCC)

s.send(frame_request)
```

When requesting an I2C command, you must set the header and the Lepton Command.

```
i2c_request = bytearray()
i2c.append(ClientRequestType.I2C_CMD)
i2c.append(I2CCommand.FCC)
```

class Message

This class allows you to easily send and receive messages from the Raspberry Pi to the server.

init(self, s=None)

Creates a message with a socket. Default is no socket.

Usage:

With a socket

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect("IP ADDRESS", 5995))
msg = Message(s)
```

Without a socket

```
msg = Message()
```

open_response(self, frame_resp)

This opens a frame and sets the object attributes. Modifies the enclosed object.

- Exceptions:
  - ValueError: When the frame_resp is invalid.
- Returns:
  - Nothing

Usage:

Take in the input the 38410 bytes response.

```
msg = Message()
msg.open_response(response)
```

property msg(self)

Creates a message from all the values in the class and returns an array of bytes.

- Exceptions:
  - AttributeError: This exception is called when the Object attributes are invalid.
- Returns:

- An array of bytes of length 38140 in the format of the Lepton Protocol.

***property* frame_request(self)**

Prepares a frame request message for the Lepton

Usage:

Prepare a frame request message. If you sending a frame request, use the function send_frame_request()

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect("IP ADDRESS", 5995))

msg = Message()
s.send(msg.frame_request)
```

is_frame_ready(self)

Checks the header of a message and returns if the frame is ready.

- Returns:
  - True if it is ready
  - False if it is not

Usage:

```
if msg.is_frame_ready():
  print("It is ready")
```

send_fcc(self)

Sends a FCC request to the server application.

- Returns:
  - True if the FCC request was successful
  - False if the FCC request was not successful

Usage:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect("IP ADDRESS", 5995))
```

```
Message(s).send_fcc()
```

send_frame_request(self)

Sends a FCC request to the server application.

- Returns:
  - True if the frame request is successful
  - False if the frame request is not successful**Usage:**

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect("IP ADDRESS", 5995))
Message(s).send_frame_request()
```

receive_response(self)

Receives a response from the server. Utilizes the recvall function and receives all 38410 bytes.

- Returns:
  - The response in a byte array
  - None if the message is not of length 38410

Usage:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect("IP ADDRESS", 5995))
Message(s).send_frame_request()

frame = Message(s).receive_response()
msg = Message()
msg.open_response(frame)
```

RaspberryPi 3 Lepton Wiring

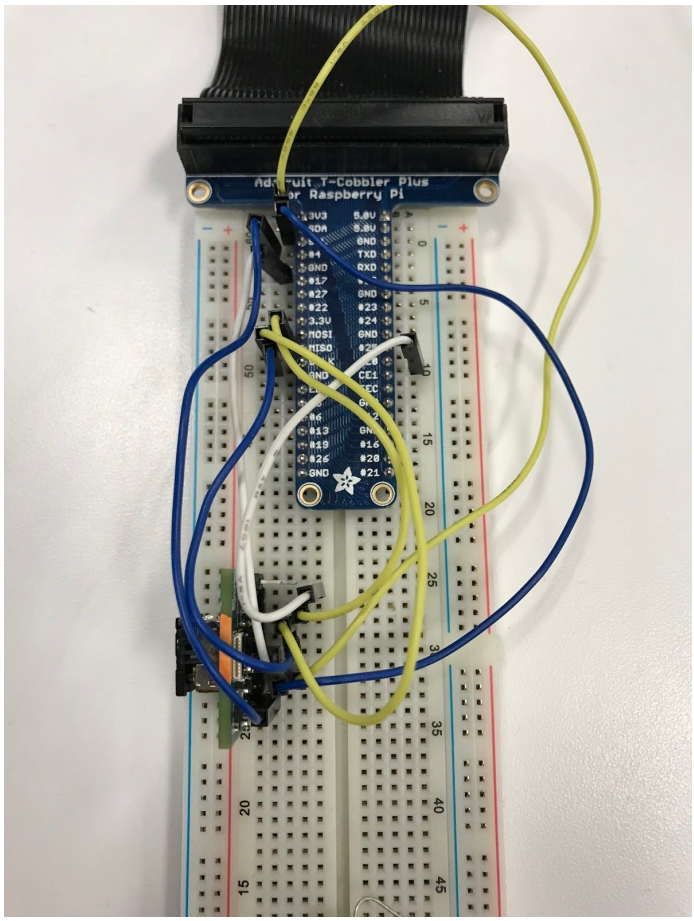
Requirements

1. Flir Lepton 3
2. Breakout Board

Pin Configuration

- SCL: Pin#5
- SDA: Pin#3
- VIN: Pin#1
- GND: Pin#25
- CLK: Pin#23
- MISO: Pin#21
- MOSI: Pin#19
- CS: Pin#24

Example



macOS

Overview

This page will describe how to connect your MacBook Pro 2016+ to the Raspberry Pi 3.

Hardware Requirements

MacBook Dongles

1. Belkin USB-C to Gigabit Ethernet Adapter

or

1. Thunderbolt to Gigabit Ethernet Adapter
2. Thunderbolt 3 (USB-C) to Thunderbolt 2 Adapter

Raspberry Pi 3

1. Micro usb cable for power.
2. SD Card

Creating the operating system

1. Copy of Raspbian
2. Flash the SD Card using Etcher.io.

Connecting to the Pi

Default Username and Password

```
ssh pi@raspberrypi.local
password: raspberry
```

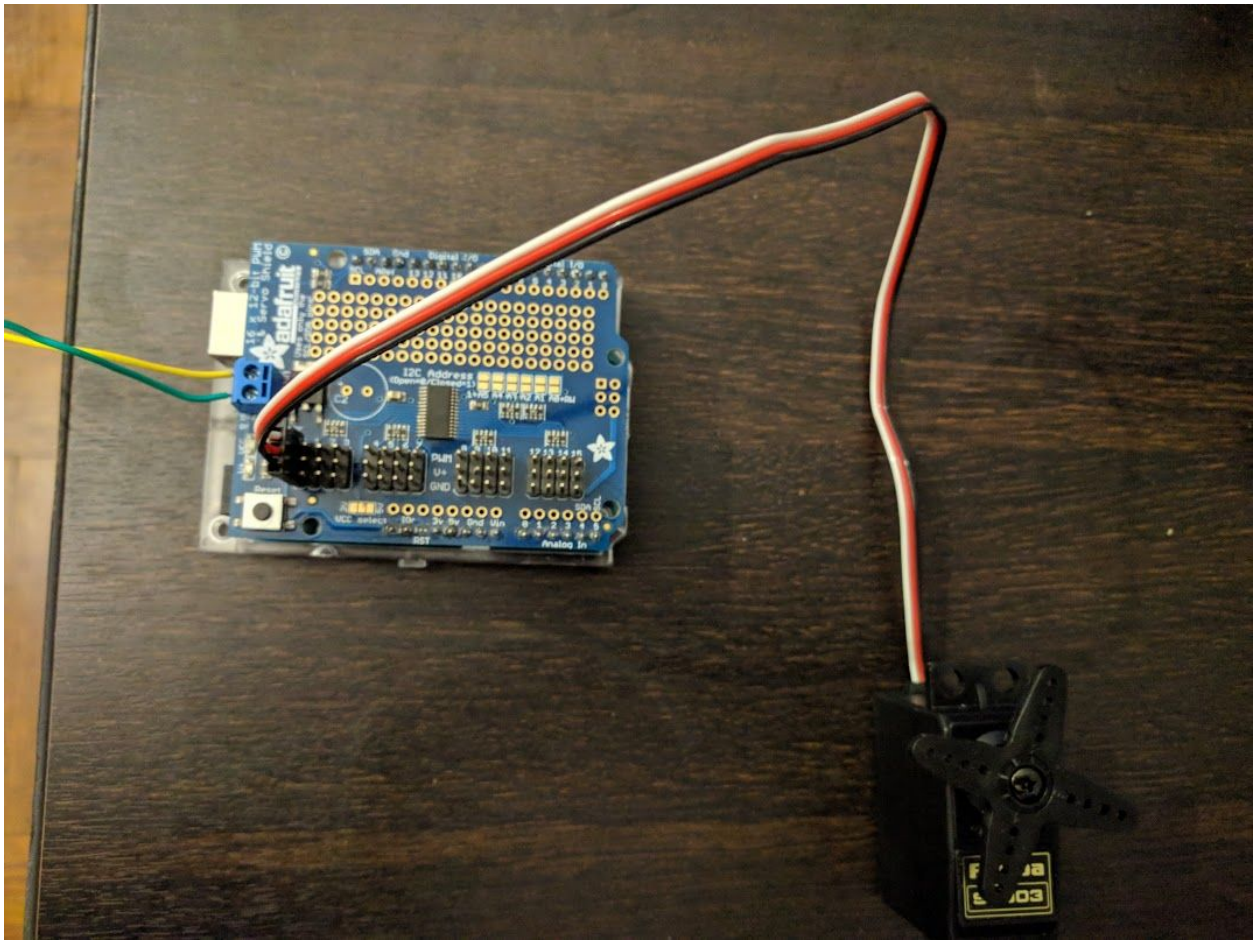
Servo Wiring Guide

- Arduino UNO R3
- Servo Shield - I2C Interface
- Servomotors (8)

Pin Configuration

Each channel corresponds to a servo motor. Connect the servo motor, ensuring the PWM, V+, and GND pins are connected correctly.

Example



Thermal Imaging and HVAC System Integration

The members will be combining the tasks allocated to them to produce a system that receives data from components attached to the Raspberry Pi and sends them to the Arduino. The Arduino should accept these commands, execute commands to the associated servo motors and successfully respond to the Raspberry Pi with a message from HVAC system in real-time.

Raspberry Pi to Arduino Communication

- The Raspberry Pi will serve as a wrapper to integrate the Thermal Imaging Pipeline and HVAC System.
    - The Raspberry Pi must establish a secure serial connection to the Arduino and have the ability to send and receive messages to and from the Arduino.

HVAC Build

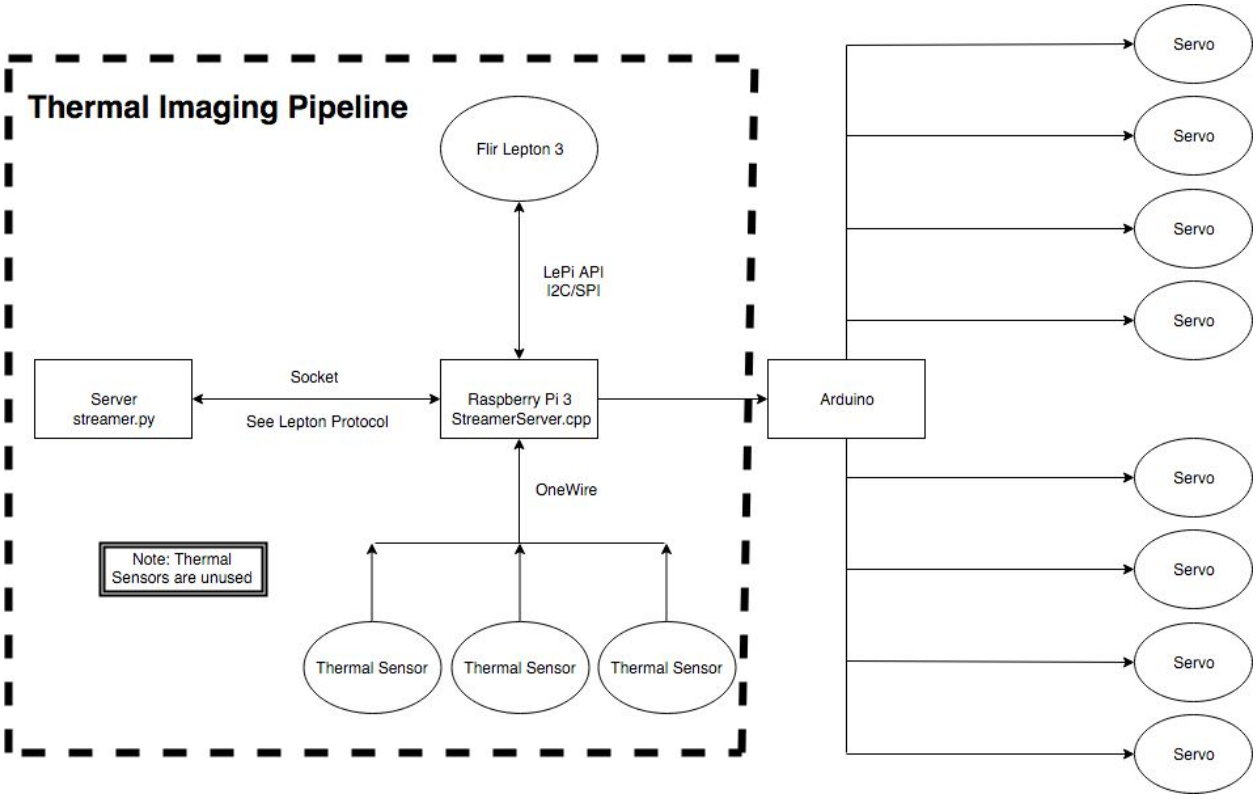
- The members will also ensure that the build for the HVAC unit and environment is operable/functional.
    - A robust build of HVAC parts must be made to adjust according to commands from the Arduino.
    - Airflow must be maintained from the source to the output tubes.
    - A secure connection between servo motors and dampers must be set to ensure smooth rotation.

Before beginning visit the Lepton 3 Wiring Guide to make sure the Flir Lepton is properly connected.

Thermal Imaging Pipeline

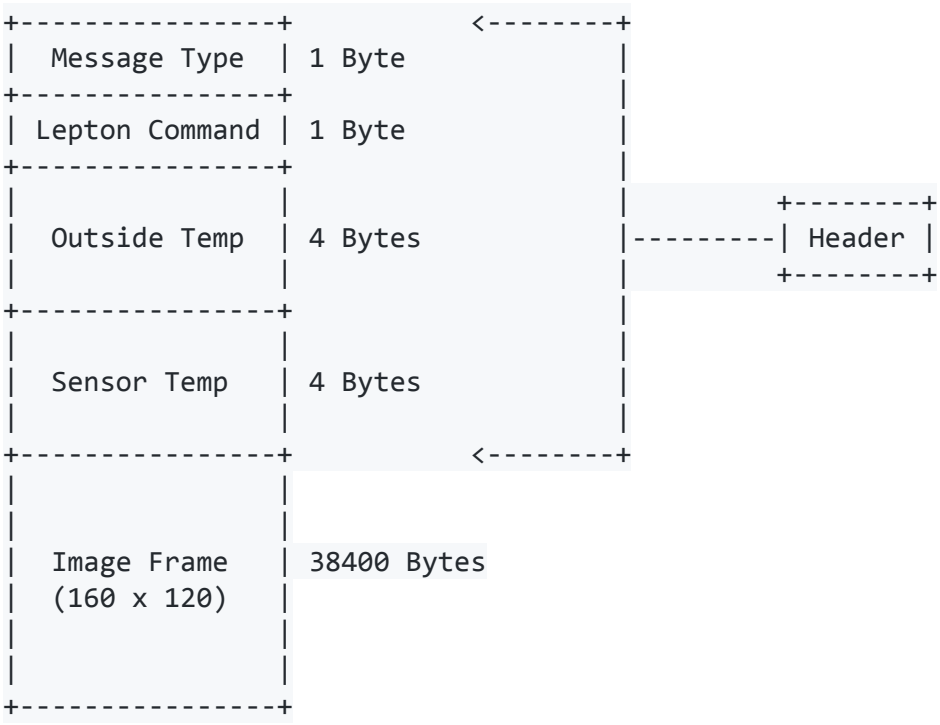
This component is meant to extract the temperature and location of the user and output that as data for the HVAC Systems' PID algorithm. This document will discuss how we extract and process Lepton Images through an external machine.

Figure: Thermal Imaging Camera Diagram



Communication

Lepton Protocol



How communication works

When an I2C command is to the StreamerServer, it reads the first byte determines it's a I2C command then executes the I2C command set in the second byte. The server then sends a response with first byte set to I2C command and the second byte header set to I2C_SUCCEED.

Example of a I2C command being received by the server.

```
[0]SERVER -- RECV -- Number of bytes read: 2
[0]SERVER -- RECV -- Message: I2C_CMD
[0]SERVER -- SEND -- Sending message response...  Message sent!
```

When an FRAME_REQUEST is sent to the StreamerServer, it reads the first byte determines it's a frame request. The server then sends a response with first byte set to FRAME_REQUEST, the second byte header set to FRAME_READY or NO_FRAME. Finally the image frame is also filled with the values from the Flir Lepton.

Example of a frame request being received by the server.


```
[1]SERVER -- RECV -- Number of bytes read: 1
[1]SERVER -- RECV -- Message: FRAME_REQUEST
[1]SERVER -- SEND -- Sending message response...  Message sent!
```

Applications running

There are 2 programs that need to be run.

1. StreamerServer.cpp
2. Streamer.py

StreamerServer.cpp

This server initiates I2C commands and collects the frames from the Lepton and sends it to the client. This application was created by Andrei Cosma. Run the the application on the Raspberry Pi 3.

streamer.py

This application connects to the streamer server through sockets, sends a FCC command, then collects frames and analyzes them utilizing opencv. This application was created by Jeff Schulthies. Run the the application on the server. This application uses the Python 3 LePi module located in ./sd-18-eco-furniture/raspberry-pi/modules/LePi/.

Getting Started

Configuration

Line 43 StreamerServer.cpp. Replace this ip address with the IP address of your Raspberry Pi

```
string ip_address = "169.254.244.43";  // New ip address
```

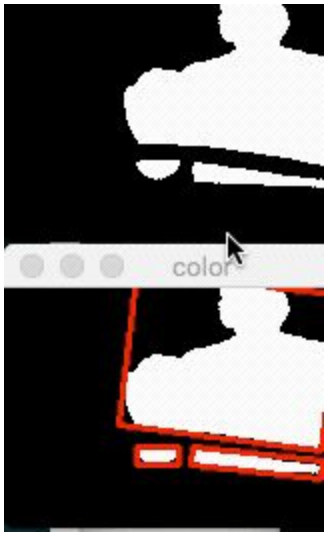
Line 14 Streamer.py Replace this ip address with the IP address of your Raspberry Pi

```
s = create_socket("169.254.44.12")
```

Usage

1. Start Streamer Server
2. Start streamer.py

Example output



References

Flir Lepton

- **Datasheet**

Temperature Sensor

- **Datasheet**