Cloud Clusters With Raspberry Pi's

Gregor von Laszewski

Editor

laszewski@gmail.com

https://cloudmesh-community.github.io/pub/vonlaszewski-pi.epub

March 27, 2020 - 07:52 AM

Created by Cloudmesh & Cyberaide Bookmanager, https://github.com/cyberaide/bookmanager

Cloud Clusters With Raspberry Pi's

Gregor von Laszewski

Cloud Clusters With Raspberry Pi's

1 INTRODUCTION

1.0.1 About ☁

This document has been created form our document collection to target the creation of Cloud Clusters with Raspberry PI's.

The document can be contributed to by you and compiled on your local machine. More information about this will be presented in this prefix.

We recommend that you read an entire section first before you start copy paste style execution of shell commands or programs. We want you to obtain first a solid overview of what you need to do. Whatever yo do crate a backup first.

1.1 Raspberry PI ☁

Raspberry PI's are a convenient cheap compute platform that allow us to explore create cloud clusters with various software that otherwise would not be accessible to most. The point is not to create a complex compute platform, but to create a *testbed* in which we can explore configuration aspects and prepare benchmarks that are run on larger and expensive cloud environments. In addition Raspberry Pis can be used as a simple Linux terminal to log into other machines.

We will give a small introduction to the platform next.

1.1.1 Raspberry PI 3 B

Till February 2018 the Raspberry PI 3 B was the newest model. Within this class we have access to about 100 of them. The Raspberry PI 3 B is shown in Figure 1

Figure 1: Raspberry PI 3B

The board has the following properties:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Switched Micro USB power source up to 2.5A

1.1.2 Raspberry PI 3 B+

We plan to purchase a number of them so we can conduct performance experiments and leverage the faster hardware. The newest Raspberry PI 3 B+ is shown in Figure 1.

Figure 2: Raspberry PI 3 B+

The board has the following properties:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN
- Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header
- Full-size HDMI
- 4 USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- 4-pole stereo output and composite video port
- Micro SD port for loading your operating system and storing data
- 5V/2.5A DC power input
- Power-over-Ethernet (PoE) support (requires separate PoE HAT)

1.1.3 Raspberry PI Zero

In addition to the PI 3's another interesting platform is the PI Zero, which is a very low cost system that can serve as IoT board. However it is also powerful

enough to run more sophisticated applications on it. The newest Raspberry PI Zero is shown in Figure 3.

Figure 3: Raspbery Pi Zero (source)

The board has the following properties:

- 1GHz single-core CPU
- 512MB RAM
- Mini HDMI port
- Micro USB OTG port
- Micro USB power
- HAT-compatible 40-pin header
- Composite video and reset headers
- CSI camera connector (v1.3 only)

1.1.4 Pin Layout

The PI 3B, 3B+ and Zero come with a number of pins that can be used to attach sensors. It is convenient to have the pinout available for your project. Hence we provide a pinout layout in Figure 4. Other Pis will have a differnt pinout and you will have to locate them on the internet.

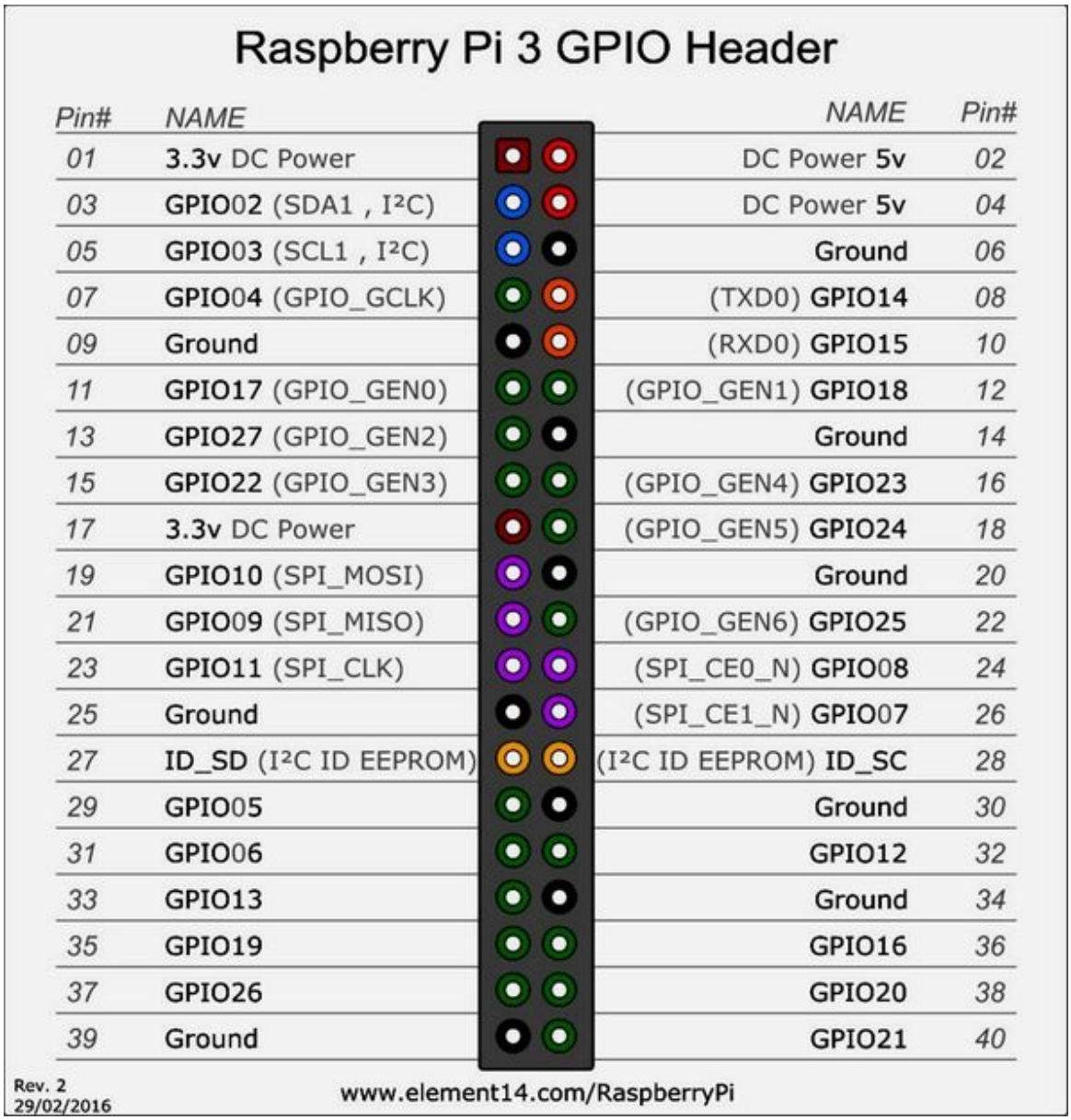


Figure 4: Pinout

1.1.5 Resources

Detailed information about it are available at

- https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md

2 IOT

2.1 Introduction ☁

Internet of Things is one of the driving forces in the modernisation of today's world. It is based on connecting *things* to the internet to create a more aware world that can be interfaced with. This not only includes us humans, but any *thing* that can interact with other things. It is clear that such a vision of interconnected devices will result in billions of devices to communicate with each other. Some of them may only communicate small number of items, while others will communicate a large amount. Analysis of this data is dependent on the capability of the *thing*. If it is to small the analysis can be conducted on a remote server or cloud while information to act are fed back from the device. In other cases the device may be completely autonomous and does not require any interaction. Yet in other cases the collaborative information gathered from such devices is used to derive decisions and actions.

Within this section we are trying to provide you with a small glimpse into how IoT devices function and can be utilized on small projects. Ideally if the class has all such a device we could even attempt to build a cloud based service that collects and redistributes the data.

To keep things simple we are not providing a general introduction in IoT. For that we offer other classes. However, we will introduce you to two different devices. These are

- esp8266
- Raspberry Pi

The reasons we chose them is that

1. They are cheap.
2. We can program both in python allowing us to use a single programming language for all projects and assignments, and
3. They are sufficiently powerful and we can conduct real projects with them beyond toy projects.

4. The devices, especially the Raspberry PI can be used to also learn Linux in case you do not have access to a linux computer. Please note however the raspberry will have memory and space limitations that you need to deal with.

Projects that you can do to test the devices are

esp8266 (easy-moderate, small memory):

- a LED blinker
- a dendrite ⭕
- a robot fish ⭕
- a fish swarm ⭕
- a robot swarm ⭕
- an activity of your desire

Raspberry Pi (easy-moderate, 32GB space limitation):

1. a LED blinker
2. a robot car
3. a robot car with camera
4. a temperature service
5. different clusters

<-!– Crazyflie 2.0 (difficult):

- programming a drone
- programming a drone swarm ->

Indiana University: Please note that for those at IU we do have a Lab in which you can use some of the devices pointed out here. You can arrange for accessing the infrastructure.

In case you want to work on a swarm, we do have positioning sensors that simplify that task.

Due to the small cost involed in these devices you can buy them also simply yourself.

We provide throughout the book lists of hardware that you will need for the various projects.

In general we think that these platforms provide a wonderful introduction into IoT. Such platforms were just a decade ago not powerful enough or too expensive. However today the provide a serious platform for developers. Sensors are available easily as most Android comparable sensors can be used.

Before we jump right into programming the devices, we like to point out that we did not chose to use Arduinos much, as their price advantage is no longer valid. They also are mostly using C and as we focus in our material on python we decided to not spend much time on it.

We also find that esp8266 and Raspberry can interface with most sensors. Having the ability to easily use WiFi however is our primary reason for using them. Furthermore being able to attach a camera to the Raspberry is just superb. Image analysis is one of the drivers for big data.

2.2 Tools ☁️

- **Terminal**: On macOS, when you navigate to the search magnification glass, you can type in *terminal* to start it. A terminal allows you to execute a number of commands to interact with the computer from a commandline interface, e.g. the terminal.

- Bash it the command language used in terminal.

- Pyenv allows to manage multiple versions of python easily. Pyenv link

- XCode is an integrated development environment for macOS containing a suite of software development tools developed by Apple for developing software for macOS, iOS, watchOS and tvOS.

- Homebrew is a *package manager* for OS X which lets the user *install software* from *UNIX* and *open source software* that is not included in macOS.

- pyCharm: is an Integrated Development Environment for Python.

- *Matplotlib*: Matplotlib is a library that allows us to create nice graphs in python. As we typically install python with virtualenv, we need to configure matplotlib properly to use it. The easiest way to do this is to execute the following commands. After you run them you can use matplotlib.

```
$ pip install numpy
$ pip install matplotlib
$ echo "backend : TkAgg" > ~/.matplotlib/matplotlibrc
```

- Macdown a macdown editor for macOS
- Markdown (from Markdown)
- AquaEmacs (from Aquamacs)
- Marvelmind (from Marvelmind if you have marvelmind positioning sensors which are optional)
- Arduino (from Arduino if you like to use their interface to access the esp8266 boards)
- 40 OSX Terminal Tricks

2.2.1 Markdown

MarkDown is a format convention that produces nicely formatted text with simple ASCII text. Markdown has very good support for editors that render the final output in a view window next to the editor pane. Two such editors are

- Macdown: MacDown provides a nice integrated editor that works well.
- pyCharm: We have successfully used Vladimir Schhneiders Markdown Navigator plugin. Once installed you click on a .md file pycharm will automatically ask to install the plugins from Markdown for you.

A detailed set of syntax rules can be found at: **BUG: LINK TO MARKDOWN SYNTAX MISSING**

The following are some basic examples

- To *emphasise* a text you use `*emphasize*`

- To make text **bold** use `**bold**`
- To make text ***bold-and-emphasize*** use `***bold-and-emphasize***`
- To create a hyperlink use `[Google](https://google.com)` which will result in [Google](https://google.com)
- To include an image use `![Bracketed Text](link)`

A list can be created by item starting with *, a - , or a + or a number

```
1. one
2. two
```

1. one
2. two

    ```
    * one
    * two
    ```

- one
- two

If you need to indent items underneath already bulleted items, precede the indent items with four spaces and they will be nested under the item previous to them.

To qoute textc precede it with a “>”.

```
> Quote
```

> *Quote*

Other syntax options can be found in the Format drop-down at the top of the screen between View and Plug-ins of macdown.

2.2.2 AQUAMACS

There are many different versions of emacs available on OSX. Aquamacs is often used as it integrates nicely with the OSX GUI interface.

- AquaEmacs

Aquamacs is a program for Mac devices which allows the user to edit text, HTML, LaTeX, C++, Java, Python, R, Perl, Ruby, PHP, and more. Aquamacs integrates well with OSX and provides many functions through a menu. You will mostly be using the File, Edit, menus or toolbar icons.

Emacs provides convenient keyboard shortcuts, most of which are combinations with the Control or Meta key (The Meta key is the ESC key). If you accidentally end up doing something wrong simply press `CTRL-g` to get out without issue. Other Keyboard Shortcuts include:

- `CTRL-x u` or File>Undo will cancel any command that you did not want done. (CHECK)
- `ESC-g` will cancel any command you are in the middle of.
- You can break paragraph lines with `Ctrl-x w`, where `w` will wrap text around word boundaries.
- To delete text to the end of the current word, press `ESC-d`.
- to delete the whole line from the position of the cursor to the end, press `CTRL-k`.

2.2.3 BASH

Bash is pre-installed in OSX. A *bash* script contains *commands* in plain text. In order to create a bash script please decide for a convenient name. Let us assume we name our script *myscript*. Than you can create and edit such a script with

```
$ touch myscript.sh
$ emacs myscripts.sh
```

Next you need to add the following line to the top ogf the script:

```
!# /bin/bash
```

To demonstrate how to continue writing a script we will be using the bash `echo` command that allows you to print text. Lets make the second line

```
echo "Hello World"
```

You can now save and start executing your script. Click “File” and then “Save”. Open Terminal and type in `cd` followed by the name of the folder you put the

document in. Now we need to execute the script.

Executing a Bash script is rather easy. In order to execute a script, we need to first execute the *permission set*. In order to give Terminal permission to read/execute a Bash script, you have to type

```
chmod u+x myscript.sh
```

After the script has been granted permission to be executed, you can test it by typing

```
./myscript.sh
```

into the terminal. You will see it prints

```
Hello World
```

2.2.4 Arduino

This installation is optional. In the event that there is a TTY error, you will need to install Arduino, since your Mac may be missing some drivers that are included in Arduino. Simply go to Arduino and follow the installation instructions.

2.2.5 OSX Terminal

CoolTerm

download http://freeware.the-meiers.org/CoolTermMac.zip

2.3 Hardware for IoT Projects ☁️

When teaching programming you may find yourself in a situation that things can be done on your computer, but you may not want to install programs that help you to learn programming on your computer. However, we have a solution (or several) for you. We will have some fun with hardware for IoT that at the same time can be used to teach you some very elementary skills in programming. However, if you would rather use your computer you certainly can do this too.

We see the following arguments for using IoT hardware:

- You will have fun with inexpensive hardware

- You will get hands on experience with IOT devices
- You will learn how to program in python
- You can keep your current computer unchanged
- You will get experience with two platforms esp8266 and Raspberry PI 3
- You can customise your choices by conducting some fun projects.
- You have the opportunity to find alternative hardware choices such as the WiPy or the ESP32. You may find cheaper or better alternatives if you buy kits when they are available. And learn in getting an overview about such devices and kits.

Note: Ordering from overseas suppliers may take significant time, so make sure to plan ahead. Prices given here are done to provide an estimate, they may vary.

2.3.1 Raspberry Pi 3

The raspberry PI 3 is a very good development platform. With its base price of $35 it is quite a bargain. You will need some additional components to make sure you can use it. Please be reminded to never connect or power the raspberry with your computers USB port. It draws some significant amperage and we do not want you to destroy your computer. We recommend that you buy a certified power adapter. The price is so cheap that you could even create your own mini cluster as a project. We do not recommend any older versions of Raspberry as they are less powerful and do not contain built-in Bluetooth or WiFi.

Configuration:

- $37.50 Pi 3
- $7.69 Case
- $7.99 Power Adapter This is an aftermarket power adapter. Lots uof us use this one.
- $6.99 HDMI cable

- Monitor/TV with hdmi
- SD Card, 8GB minimum, 32GB maximum

Advantages:

- Full Linux like OS based on debian
- Good environment for learning Linux and Python
- Reasonable interfaces to IoT sensors
- excellent camera support
- excellent choice of expansion packages including Grove Sensors that make it easy to attach sensors and actuators. An example package is the Grove Starter Kit for about $90

Disadvantages:

- We tried the Windows IoT package and were not impressed by it. This is not an issue of the Raspberry, but the Windows IoT platform

---

2.3.2 ESP8266 Robot Car Kit

The ESP8266 has many variants. Some of which are difficult to interface with. However, this does not apply for the ESP8266 NodeMCU. This board is originally flashed with *Lua*, however it can easily be reflashed with MicroPython. In addition it is often offered as part of a platform to develop a robot car. There are arguably better kits available, but the price of $24 for the entire kit is hard to beat. Unfortunately the version of python, as well as the limited memory make the esp8266 not a full fledged platform for python programming and you will quickly see its limitations. Interfacing with it, however, as an IoT device will gain you a lot of insights.

Configuration:

- $14.99 esp8266 & shield or esp8266 & shield
- $12.59 Chasis

- 4 * AA Rechargable Batteries & charger

Optionally you may want to get additional sensors such as wheel Encoders

- Wheel Encoder

Advantages:

- Very low price for what it can do
- We have macOS software available that makes it easy to setup (Other tutorials for other platforms are available on the internet, you can contribute by creating documentation we distribute in class for points)
-
    -

2.3.3 Sensor Kit

It is fun to attach sensors to your IoT board. There are many kits available and we encourage you to do comparisons. One such kit is

- $29.99 Elegoo 37 Sensors

However it does not include a breadboard like other kits. Hence we recommend that you get a breadboard as it makes experimenting easier.

- $5.68 small bread board and wires

2.3.4 Fish Kit

- $29.99 shark
    - cheaper balloons leak
    - before assembly and putting gas in, make sure components work.
    - gas will last typically for one week
    - $39.99 gas can be purchased in party store

- 2 g9 servo
- soldering (for cable, so cheap one will do)
- pins
- esp
- double sided scotch tape
- hot glue gun
- paper clips

2.3.5 Alternative components

2.3.5.1 Esp8266 Alternatives

Two models are good. Adafruit has some added features, but may need soldering

- $8.79 NodeMCU
- $16.95 Adafruit Feather

2.3.5.2 Car Parts Alternatives

- $14.59 Car Chasis
- $22.88 Car Chasis and Arduino

2.3.5.3 Simple sensors

Simple sensors can be attached to the boards with cables (that you need to purchase separately). Examples include

- Elegoo 37 sensor kit
- Breadboard Cable

2.3.5.4 Grove Sensors

Grove sensors have ready-made cables that make them easy to attach to the Raspberry PI. However, they are more expensive. You still need a Raspberry PI. No soldering iron and no breadboards are required.

- Grove Starter Set
- Seed Studio Grove Sensors
- Grove Shield for NodeMCU
- Grove Cable

2.3.6 Alternative Hardware and Sensors

In this section we will list a number of alternative hardware products that we are exploring. If you have used them, please help us improving these sections.

2.3.6.1 Small Footprint Battery Power

The following board provides to the Raspberry Pi a lithium battery power pack expansion board. It is powered by two 18650 Li-ion batteries providing steady. A 4-LED indicator indicates the level of charge. The board costs $16.99.

- https://www.sunfounder.com/plus-power-module.html

2.4 Sensors ☺ FA18-523-84 ☁

This section contains the wiring diagrams and associated classes for sensors that can be used with the Raspberry Pi. In addition to the individual sensors an example project is also included. Before getting started with sensors for the Raspberry Pi you will need to ensure that the Pi is set up with python3 and has the latest version of Raspbian installed. Instructions for setting up the Raspberry Pi can be found here.

2.4.1 DS18B20 Temperature Sensor

The DS18B20 is a thermoresistive temperature sensor and can be found in many of the sensor kits referenced in this book. To set up the DS18B20 connect the jumper wires as shown in Figure 5. If you have an individual sensor instead of a sensor module you will need to use a 4.7k ohm resistor as shown in the diagram. The resistor allows the one wire interface to work properly and should be used to avoid damage to the sensor **???**. If you have a DS18B20 module it may already include a resistor and you will not need to add another. Be sure to check before setting up your sensor.

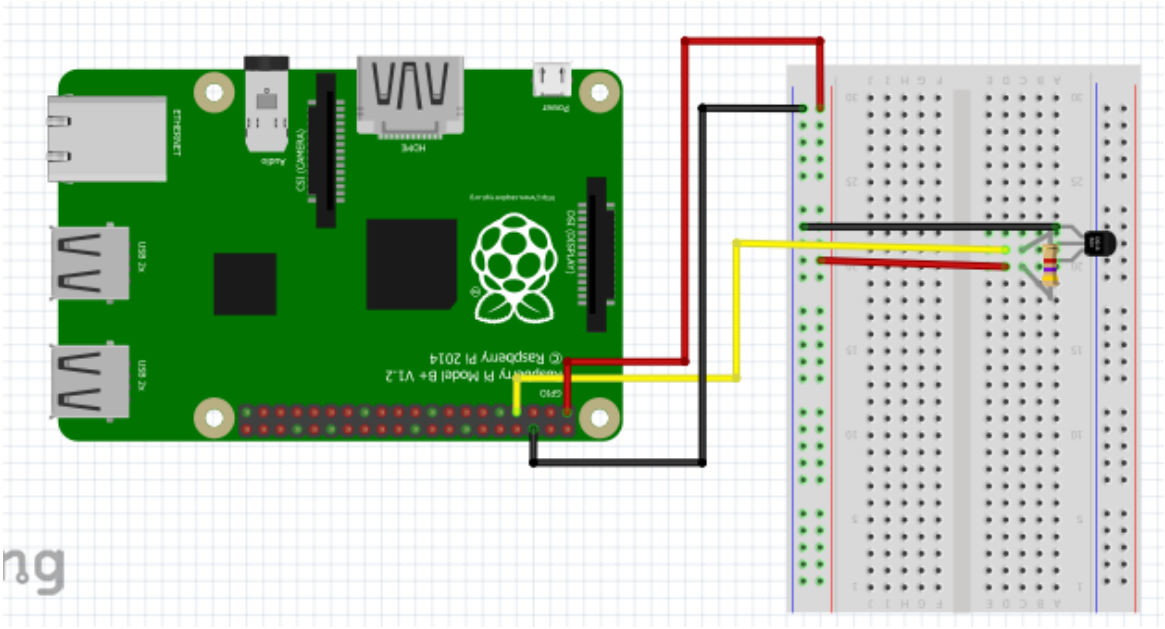


Figure 5: DS18B20 Setup

Once you have set up the wiring of the DS18B20 you will need to set up the one wire interface. This can be done with the following steps **???**.

1. In a terminal enter: `sudo nano /boot/config.txt`
2. Scroll to the bottom of this text file and enter `dtoverlay=w1-gpio`

Once the setup is complete you can use the DS18B20 code provided to output the temperature to the terminal.

DS18B20 Class

2.4.2 Temperature and Humidity Sensor Module

The temperature and humidity sensor used in this example is the DHT11 sensor which can be purchased as a part of the Kookye Smart Home Sensor kit or the Elegoo Uno Kit.. The humidity component of the DHT11 works by measuring the conductivity between two electrodes. Between these electrodes there is a substrate that holds moisture and as the moisture changes the conductivity changes **???**. The temperature sensor of the DHT11 works in the same way as the DS18B20.

To set up the DHT11 sensor connect jumper wires to the Raspberry Pi as shown in Figure 6. Ensure that the ground wire of the DHT11 is connected to the ground rail of the breadboard or a ground pin on the Raspberry Pi. The VCC wire of the DHT11 should be connected to 3.3v from the Raspberry Pi. To receive data the middle pin should be connected to one of the GPIO pins on the Raspberry Pi. In this example and associated code we connect the data wire to GPIO 4 on the Raspberry Pi as shown in Figure 6.

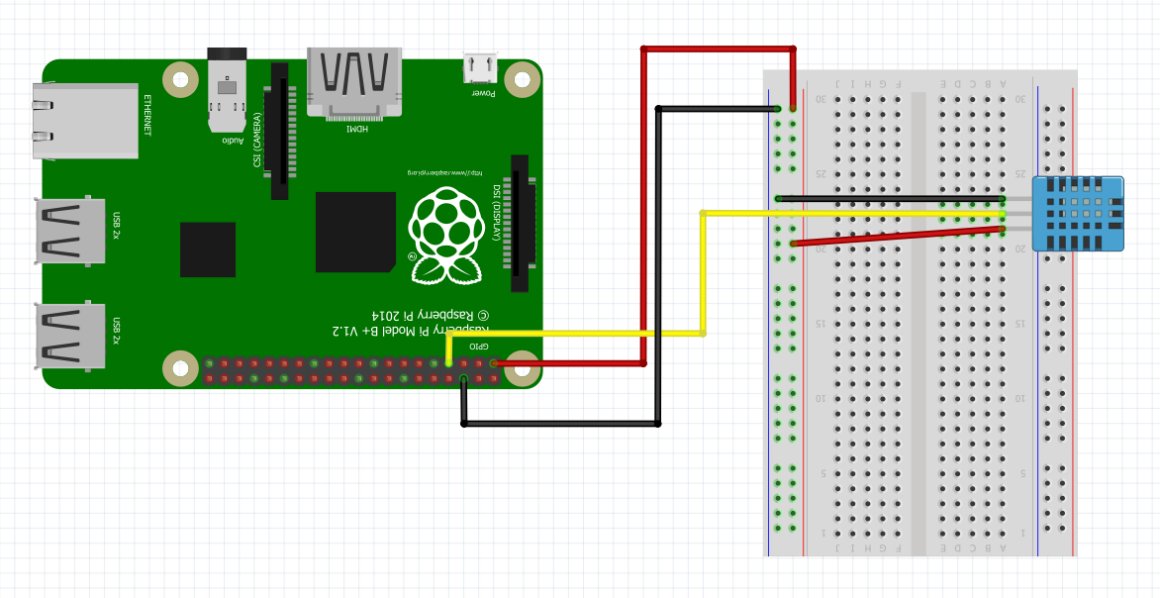


Figure 6: DHT11 Setup

Once you have checked that the DHT11 is set up correctly you will need to set up the Adafruit_DHT module for python. The sample python class utilizes the Adafruit_DHT module which can be set up by executing the following code in a terminal on your Raspberry Pi **???**.

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_DHT
sudo apt-get update
sudo apt-get install build-essential python-dev
sudo python setup.py install
```

Once you have set up the Adafruit_DHT module you can use the python class to display the temperature and humidity reading to the terminal.

Temperature & Humidity Sensor Class

2.4.3 Photosensitive Light Sensor Module

The light sensor used in this example can be purchased individually or as part of a sensor kit. To set up the light sensor module connect the wires to the Raspberry Pi as shown in Figure 7. The sensor shown in this example has three pins. However, some sensor modules may have four pins. In most cases the extra pin is not necessary.

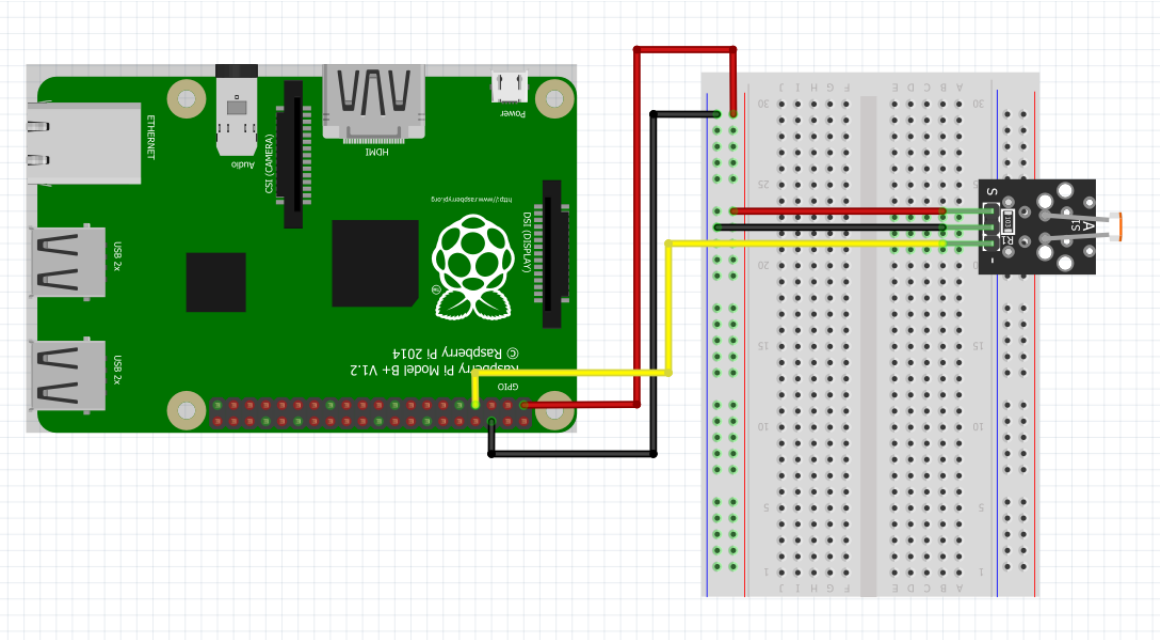


Figure 7: Light Sensor Setup

Once you have set up the light sensor you can use the light_sensor class to retrieve data from the light sensor. The light sensor will return "1" if light is not detected and a "0" if the sensor detects light. Some sensor modules also include a potentiometer which can be adjusted to change the sensitivity of the light sensor.

Light Sensor Class

2.4.4 Capacitive Touch Sensor Module

In this example we are using a momentary capacitive touch sensor. The sensor kits mentioned in this book will also contain this sensor. To set up the touch sensor connect the wires to the Raspberry Pi as shown in Figure 8.

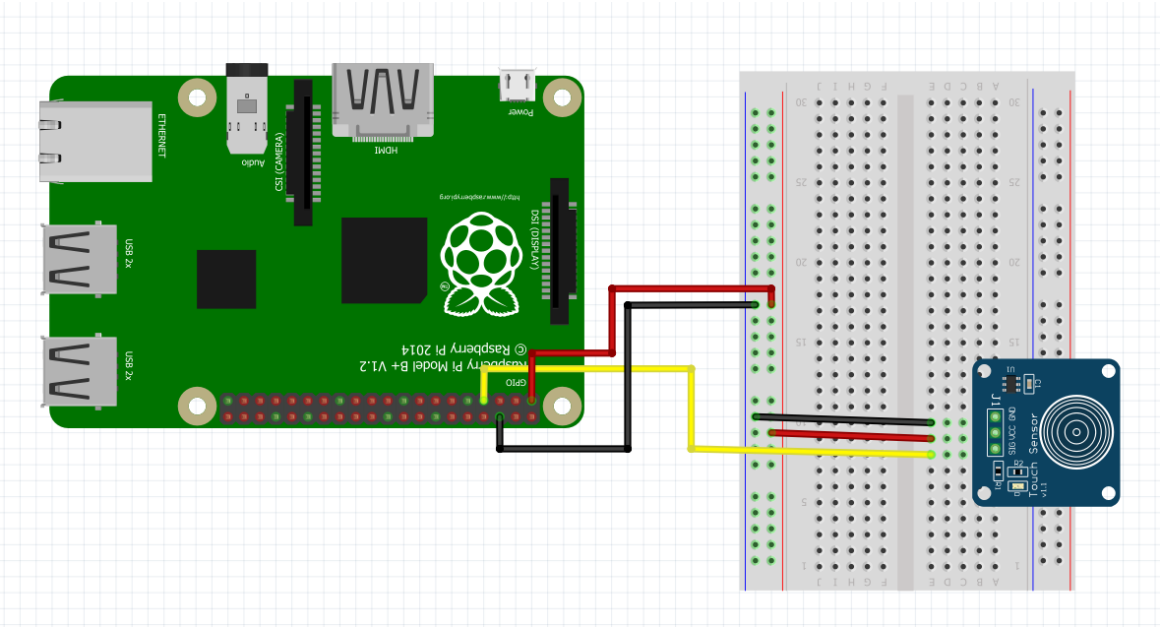


Figure 8: Touch Sensor Setup

Once you have set up the sensor you can use the touch_sensor class to execute another function when the sensor is touched.

Touch Sensor Class

2.4.5 Relay Module

The relay module can be used as a switch to complete a circuit. The module can be purchased as an individual component or may be included on a board with 2, 4 or more relay switches. In this example we will be using a two channel relay module. To set up the relay module connect the wires to the Raspberry Pi as shown in Figure 9.

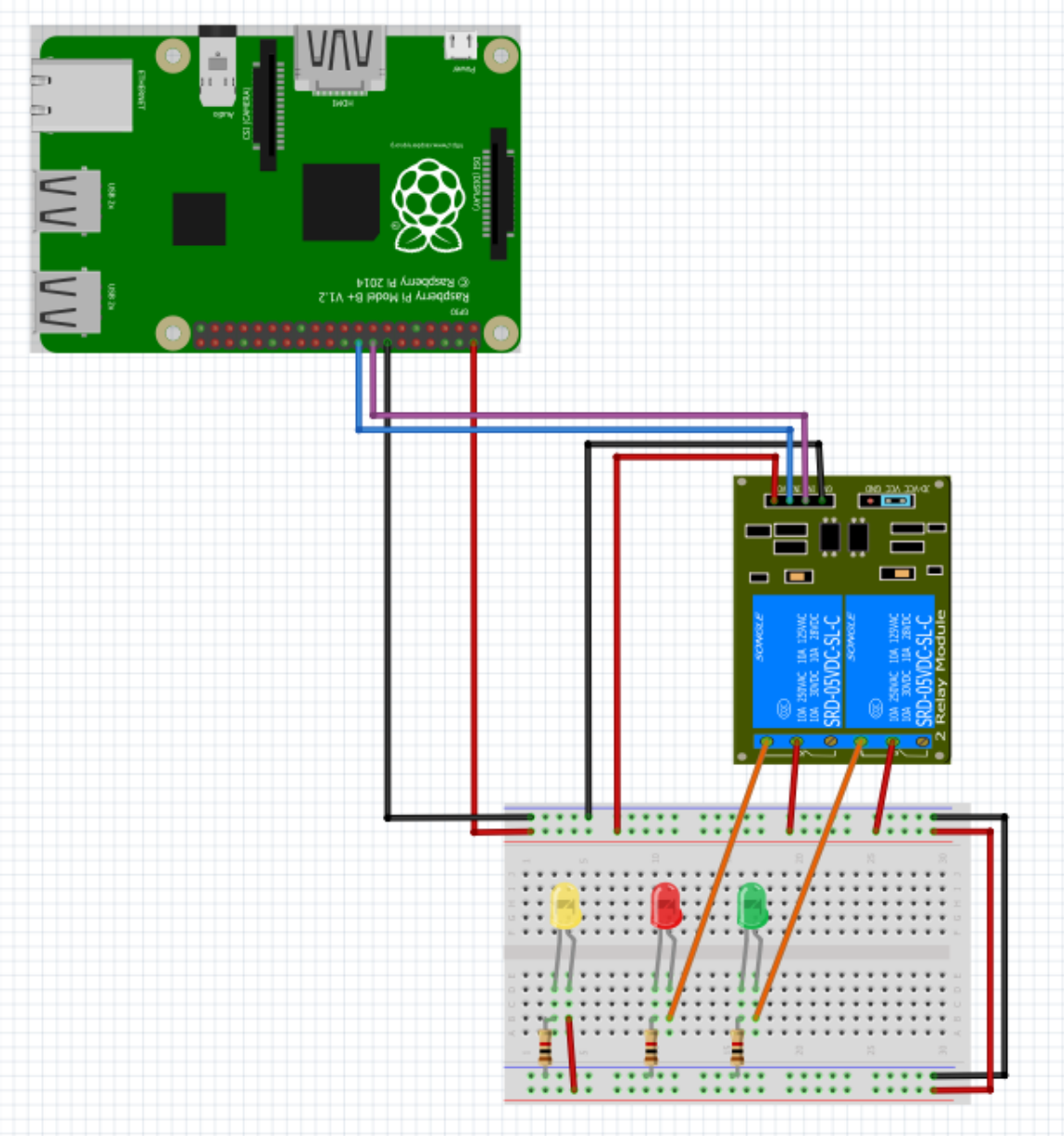


Figure 9: Relay Module Setup

Once the module is set up you can use the relay_switch class to turn the relays on and off.

Relay Class

2.4.6 16 x 2 LCD Screen

The 16x2 LCD screen can be used as a display for the Raspberry Pi. To set up the LCD screen connect the wires as shown in Figure **¿fig:LCD_setup?**. You will

also need two potentiometers in order to adjust the contrast and the brightness.

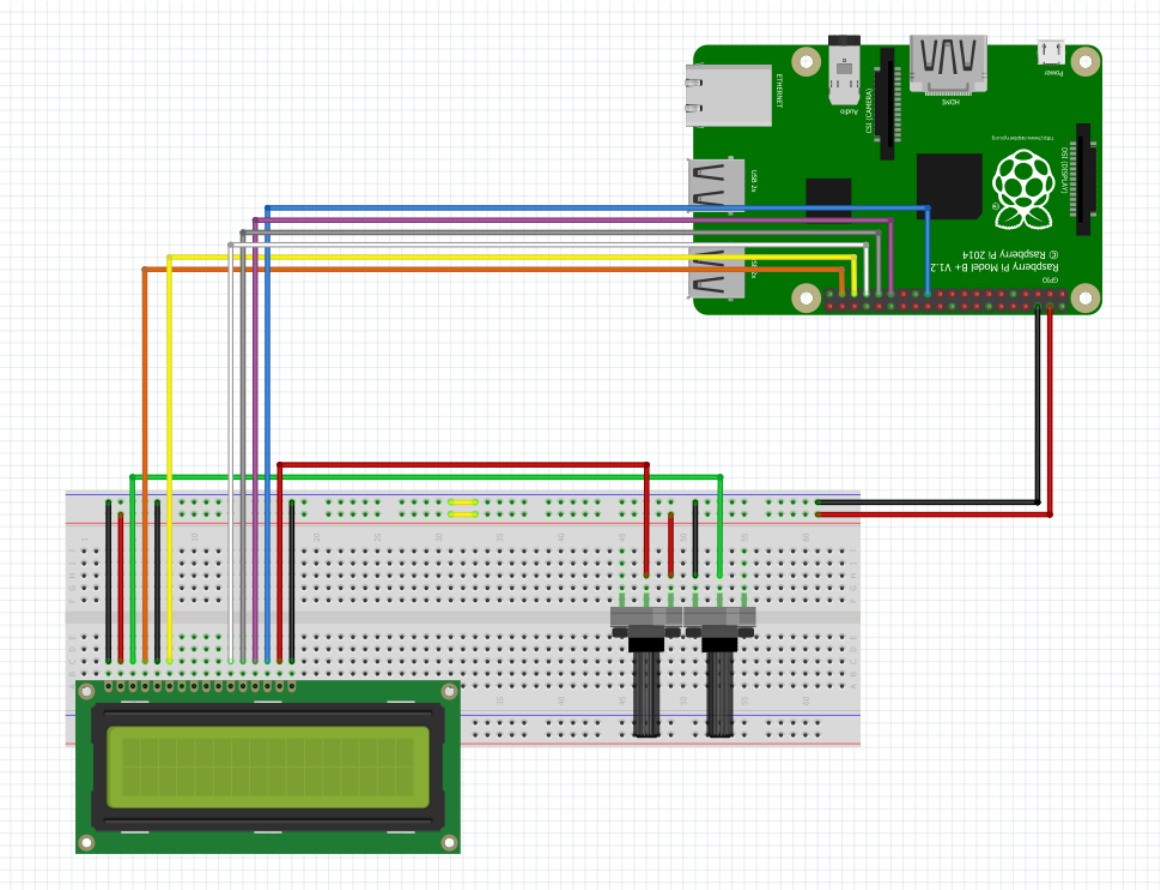


Figure 10: LCD Setup

Once everything is correctly wired up the LCD class can be used to print text to the LCD display.

LCD Class

2.4.7 Compass

TODO: which compass sensor

The default pins are defined in variants/nodemcu/pins_arduino.h as GPIO

```
SDA=4
SCL=5
D1=5
D2=4.
```

You can also choose the pins yourself using the I2C constructor Wire.begin(int sda, int scl);

2.4.8 Project: Smart Thermostat

In this example we will combine some of the sensors discussed in this section to create a smart thermostat. The first step of this project is to make sure that you have a Raspberry Pi which has Raspbian installed and is configured appropriately. Instructions for how to complete the basic set up of your Raspberry Pi can be found in the Setting up a Single Raspberry PI section of this book.

Prerequisites:

- Raspberry Pi 3 with Raspbian installed
- DHT11 Temperature and Humidity Sensor
- Photosensitive Light Sensor
- Capacitive Touch Sensor
- 2 Channel Relay Module
- 16x2 LCD display
- Three bread boards
- Two potentiometers
- 4 female to female jumper wires
- 24 male to male jumper wires
- 12 male to female jumper wires

If using the LED's you will want the additional components listed next:

- Three 4.7k ohm resistors
- Three LED's
- 3 additional male to male jumper wires
- 4 additional male to female jumper wires

Once you have the necessary components you will need to connect the wires as shown in Figure 11. The LED lights in this example represent the connections to the actual thermostat. It is suggested that you test the set up using the LED's to ensure that everything is wired correctly and that you are getting the expected results. We will cover how to connect the smart thermostat to your HVAC system later in this example. For this project three separate breadboards are used to hold different components. The first one will hold the LCD and potentiometers used to

adjust contrast and brightness. The second will hold all of the sensors and the third will be used to either hold the LED's or distribute power from the HVAC system.

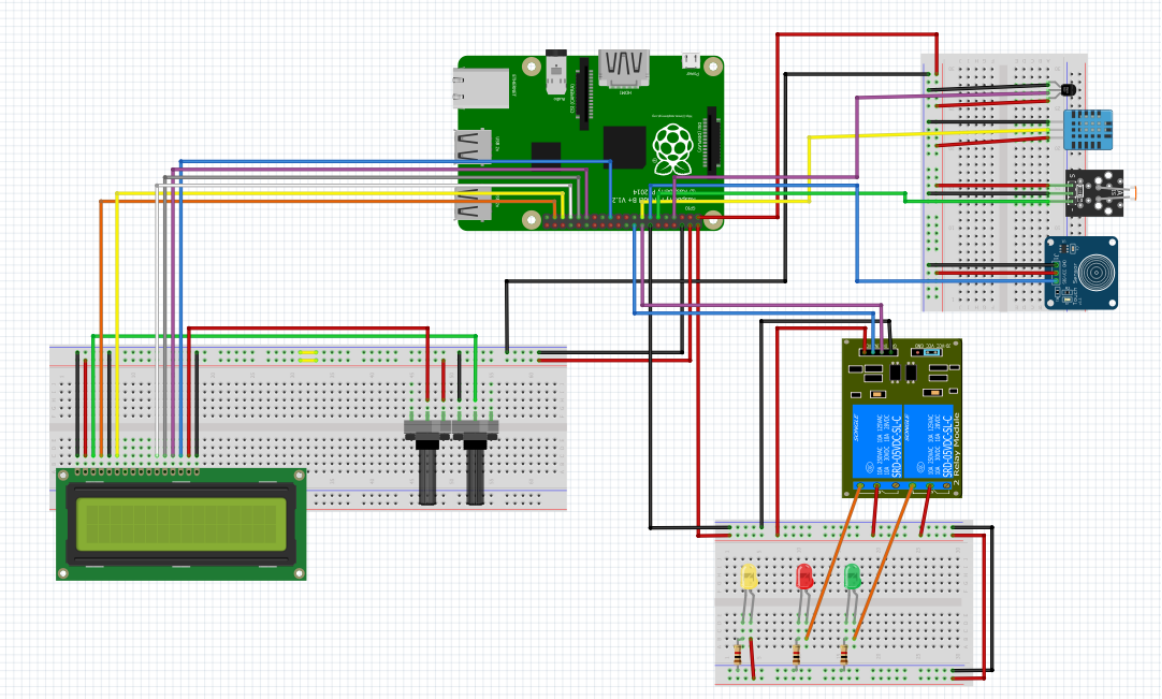


Figure 11: Smart_Thermostat Setup

Once all of the components are connected as shown in Figure 11 we need to test each of the sensors. To do this we will first need to set up the raspberry pi. This can be done by running the thermostat_setup.sh shell script. To run this open a terminal and type `nano thermostat_setup.sh` then copy the code from thermostat_setup.sh. Once you have copied the code, close the editor and run `chmod u+x thermostat_setup.sh`. Then run `./thermostat_setup.sh` to run the code to add the code and dependencies needed for this project.

Once the thermostat_setup.sh script has finished running there will be a new directory containing the code for this project. Navigate to this directory with `cd ~/git-repos/fa18-523-84/paper/code`. We will now test each of the components by running the following commands.

- `python3 LCD.py` This should show the output "Hello World!" on the LCD screen.
- `python3 temp_humid.py` Should show the current temp and humidity.
- `python3 ds18b20.py` Should show temp. (using this sensor as well because temp is more accurate)

- `python3 light_sensor.py` Should show either 1 or 0 depending on if light is detected.
- `python3 relay_switch.py` Connected to the LED's this should turn the LED's on and off.
- `python3 touch_sensor.py` When the sensor is touched "Hey!" should be printed to the terminal.

Once you have tested the components and have ensured that they work you can run `python3 smart_therm_not_connected.py` to start the smart thermostat. The code should print the current system status to the terminal.

Smart Thermostat Code

Now that we have tested each of the components and have tested the smart thermostat code using the LED's we can connect to the HVAC system. Each HVAC system is different so be sure to do some research on how your specific system works. Generally there will be a power wire that you can connect to the relay switch and then connect the other wires to the appropriate terminals **???**. Based on the readings from the other sensors the code will determine which relay to turn on, which will complete the circuit sending a signal to the HVAC system. The system used in this example is shown in Figure 12. For this system the red wire is 24v power, green connects to the fan, white connects to the heat, yellow connects to the AC compressor and blue is ground **???**.

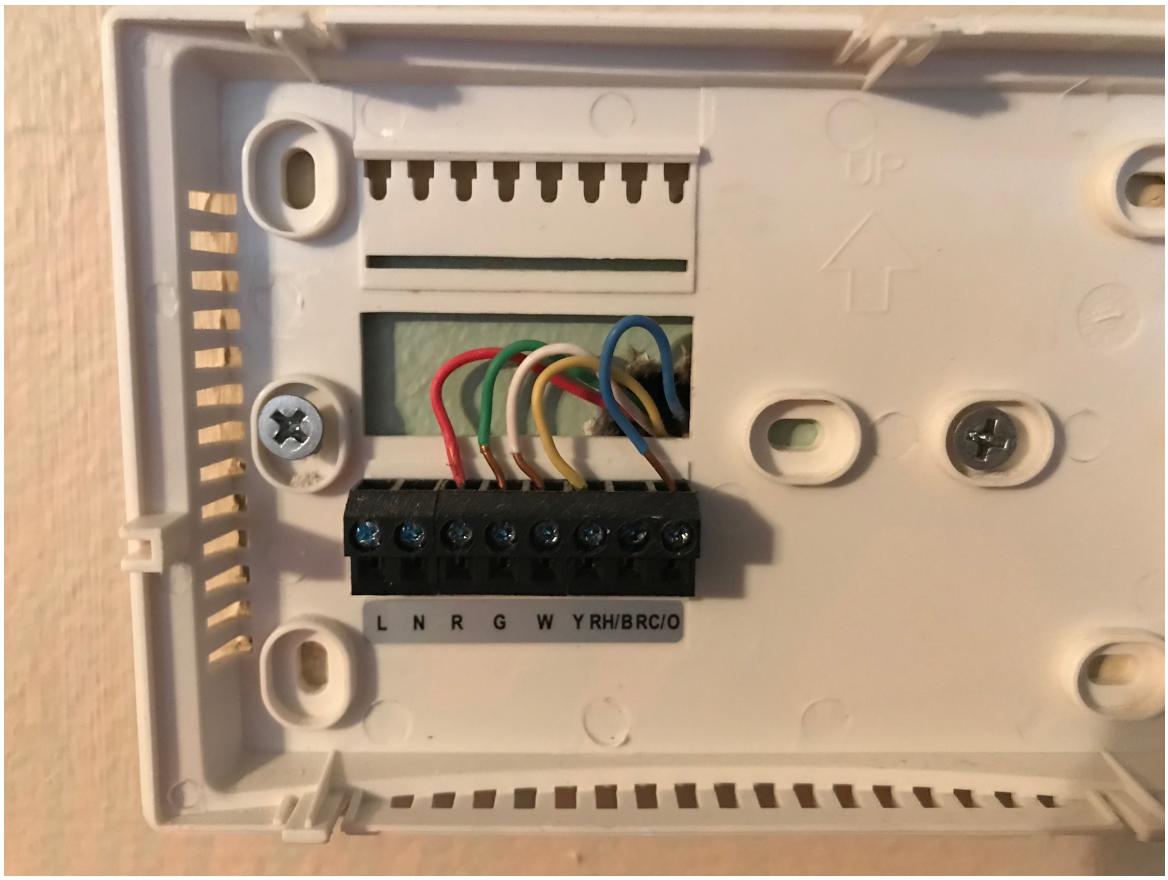


Figure 12: HVAC Wiring

To connect the Raspberry Pi smart thermostat to the system we need a way to supply power to each of the signal wires. To do this we can use a breadboard to supply power to each of the three relay switches as shown in Figure 13. When the relay switch is activated the signal will be supplied to the appropriate wire.

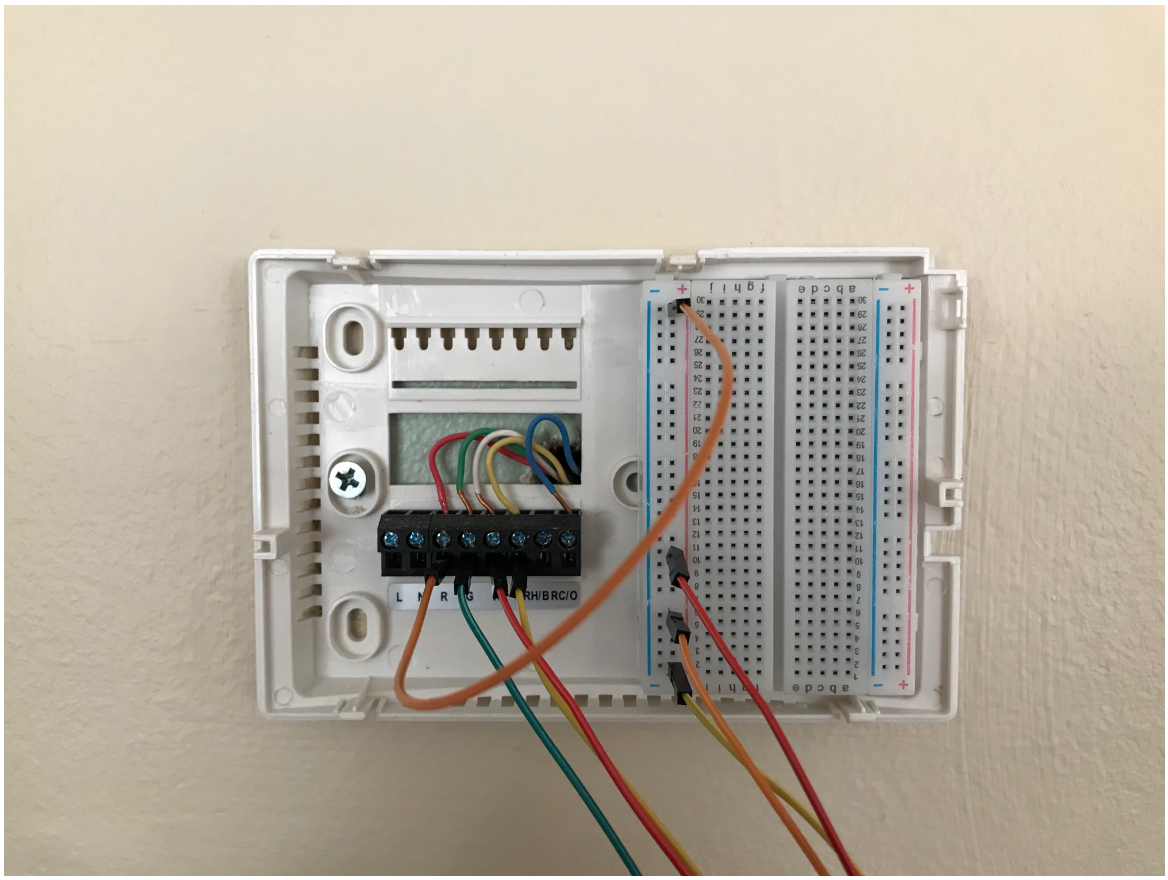


Figure 13: HVAC Wiring Final

Once the system is wired correctly we can test the code again now that it is wired to the HVAC system. Running `python3 smart_therm_not_connected.py` will start the program but to run the program in the background use `nohup python3 smart_therm_not_connected.py &`. When running the program in the background the output will be saved to the nohup.out file. Now you can adjust settings in the program to make your HVAC system more efficient. The final result is shown in Figure 14. Also in this book you can find an example which connects the smart thermostat to a database to store data and also allows anyone on the local network to change settings.

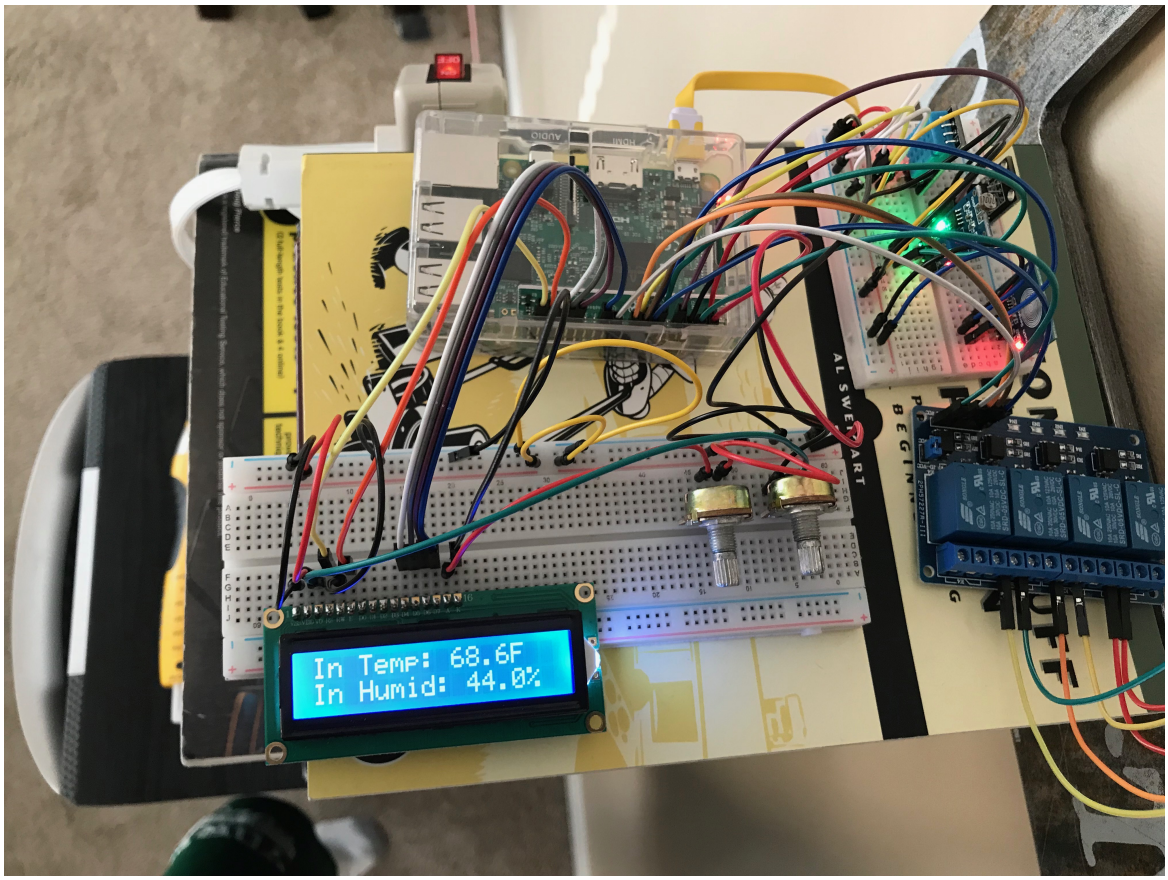


Figure 14: Smart Thermostat Final

2.4.9 Sources for this section:

need to integrate into https://github.com/cloudmesh-community/book/blob/master/chapters/SECTION/SECTION-REFERENCES.md

- DS18B20_resistor: https://arduino.stackexchange.com/questions/30822/the-use-of-4-7kohm-resistor-with-ds18b20-temperature-sensor
- DS18B20_code_setup: http://www.circuitbasics.com/raspberry-pi-ds18b20-temperature-sensor-tutorial/
- Adafruit_setup: https://stackoverflow.com/questions/28913592/python-gpio-code-for-dht-11-temperature-sensor-fails-in-pi-2
- How_DHT11_Works: https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/
- Smart_therm_example: https://dzone.com/articles/how-to-build-your-own-arduino-thermostat

2.5 GrovePi Modules

2.5.1 Introduction

- Electronics: An introduction to the basic principals of electronics.
- Voltage: An introduction to the physics of electricity.
- Unix: An introduction to the Unix os.
- grove_examples: A list of Dexter Industries example code for GrovePi modules.
- GrovePi module classes: A repository for the GrovePi module classes.

2.5.2 LED

An LED is the simplest possible module for a raspberry pi, as it is responsive only to the provided power. For an LED to emit light, it must be exposed to a voltage greater than a certain threshold value. Above this voltage, the conductivity of the diode increases exponentially and its brightness increases likewise. If the current through the LED becomes too high, the LED will burn out. The following link leads to a tutorial from Dexter Industries for the LED module.

- Dexter LED tutorial

Connect the LED to a digital port (see Figure 15). The following code describes an LED class. Since it is connected to a digital output, the voltage has only two states, on and off. The default port for the LED class is D3. The code for the LED class can be found here:

- LED Class

Figure 15: LED

2.5.3 Buzzer

The buzzer is shown in Figure 16. Connect the buzzer to a digital port. The default port for the Buzzer class is D3. You will notice that the Buzzer class and the LED class are interchangeable. This is because they work on the same digital principal. Their two values are on and off. The code for the `Buzzer` class can be found here:

- Buzzer Class

Figure 16: Buzzer

2.5.4 Relay

The relay is shown in Figure 17. The relay acts as a switch in a circuit. When the value on the relay is 1, it allows current to flow through it. When the value is 0, the relay breaks the circuit and the current stops. Connect the relay to a digital port. The default digital port is D4. The `Relay` class can be found here:

- Relay Class

Figure 17: Relay

2.5.5 Light Sensor

The light sensor is shown in Figure 18. The light sensor measures light intensity and returns a value between 0 and 1023. Connect the light sensor to an analog port. The default port is A0. The analog port allows the light sensor to return a range of values. The `LightSensor` class can be found here:

- LightSensor Class

Figure 18: Light Sensor

2.5.6 Rotary Angle Sensor

The rotary angle sensor is shown in Figure 19. The rotary angle sensor measures the angle to which it is turned. Connect the sensor to an analog port. Port A0 is the default. The `RotarySensor` class can be found here:

- RotarySensor Class

Figure 19: Rotary Angle Sensor

2.5.7 Barometer

The barometer is shown in Figure 20. Connect the barometer to an I2C port. In addition to pressure, the GrovePi barometer measures temperature in Fahrenheit and Celsius. The `Barometer` class can be found here.

- Barometer Class

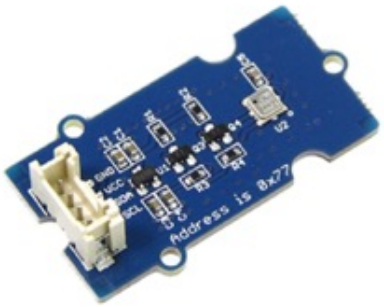


Figure 20: Barometer

2.5.8 Distance Sensor

The distance sensor is shown in Figure 21. Connect the distance sensor to a digital port. The grovepi module has a built-in function to read the distance from the distance sensor, but it is improperly calibrated, so this DistanceSensor class has a calibration based on experimental data. The `DistanceSensor` class can be found here:

- DistanceSensor Class

Figure 21: Distance Sensor

2.5.9 Temperature Sensor

The temperature sensor is shown in Figure 22. The temperature sensor measures both temperature and humidity. Connect the temperature sensor to a digital port.

D7 is the default port. The `TemperatureSensor` class can be found here:

- TemperatureSensor Class

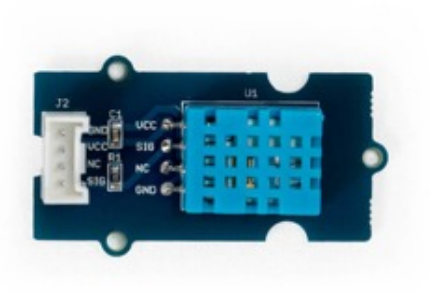


Figure 22: Temperature Sensor

2.5.10 Heartbeat Sensor

the heartbeat sensor is shown in Figure 23. Connect the heartbeat sensor to an I2C port. The heartbeat sensor returns the heart rate of the wearer. The `HeartbeatSensor` class can be found here:

- HeartbeatSensor Class

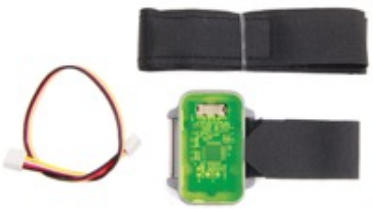


Figure 23: Heartbeat Sensor

2.5.11 Joystick

The joystick is shown in Figure 24. Connect the joystick to an analog port. A0 is the default port. The joystick has an x, y, and click status based on the current

state of the module. The `Joystick` class can be found here:

- Joystick Class

Figure 24: Joystick

2.5.12 LCD Screen

The LCD screen is shown in Figure 25. The LCD screen can be used to display text and colors. In order to use it, plug it into one of the I2C ports. The `LCD` class can be found here:

- LCD Class

Figure 25: LCD Screen

2.5.13 Moisture Sensor

The moisture sensor is shown in Figure 26. Connect the moisture sensor to an analog port. The default port is A0. The `MoistureSensor` class can be found here:

- MoistureSensor Class

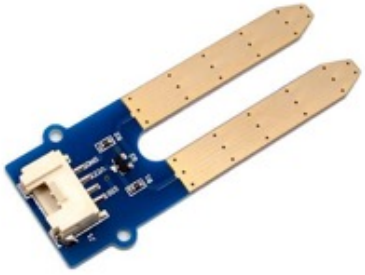


Figure 26: Moisture Sensor

An example of the implementation of the moisture sensor from Dexter Industries can be found here. The program is meant to measure the environmental conditions that affect plant growth.

2.5.14 Water Sensor

The water sensor is shown in Figure 27. The water sensor measures the amount of water in the environment of the sensor. Connect the sensor to a digital point. D2 is the default port. The `WaterSensor` class can be found here:

- WaterSensor Class

Figure 27: Water Sensor

2.6 Dexter

2.6.1 Creating an SD Card

2.6.1.1 macOS

First, install Etcher from etcher.io which allows you to flash images onto the SD card. When flashing make sure you only attach one USB SD card reader/writer or use the build in SD card slot provided in some Mac's.

The version of etcher we used is

- Etcher-1.1.1-darwin-x64.dmg

Make sure to check if there is a newer version

2.6.1.2 DexterOS

DexterOS:

- https://www.dexterindustries.com/dexteros/get-dexteros-operating-system-for-raspberry-pi-robotics
- https://www.dexterindustries.com/download/dexteros

 DexterOS 9:15 Set up SDCard (original Video)

The video is published on the Dexter Web site.

2.6.1.3 Dexter Raspbian

Dexter provides a special image that contains the drivers and sample programs for the GrovePi shield. We had some issues installing it on a plain Raspbian OS, thus we recommend that you use dexters version if you use the GrovePi shield. It is available from

- Google Drive
- Sourceforge

Detailed information on how to generate an SD card while using your OS is provided at

- https://www.dexterindustries.com/howto/install-raspbian-for-robots-image-on-an-sd-card

2.6.1.4 Github

Dexter maintains a github repository that includes their code for the shield and many other projects at

- https://github.com/DexterInd

2.6.1.5 Cloning Grove PI

To clone the GrovePI library on other computers you can use the command

```
git clone https://github.com/DexterInd/GrovePi.git
```

2.6.1.6 Dexter Sample programs

Dexter maintains all GrovePi related programs at

- https://github.com/DexterInd/GrovePi

The python related programs are in a subdirectory at

- https://github.com/DexterInd/GrovePi/tree/master/Software/Python

Here you find many programs and for a complete list visit that link. Dependent on the sensors and actuators you have, inspect some programs. Some of them may inspire you to purchase some sensors.

We have developed a partial library of GrovePi module classes at

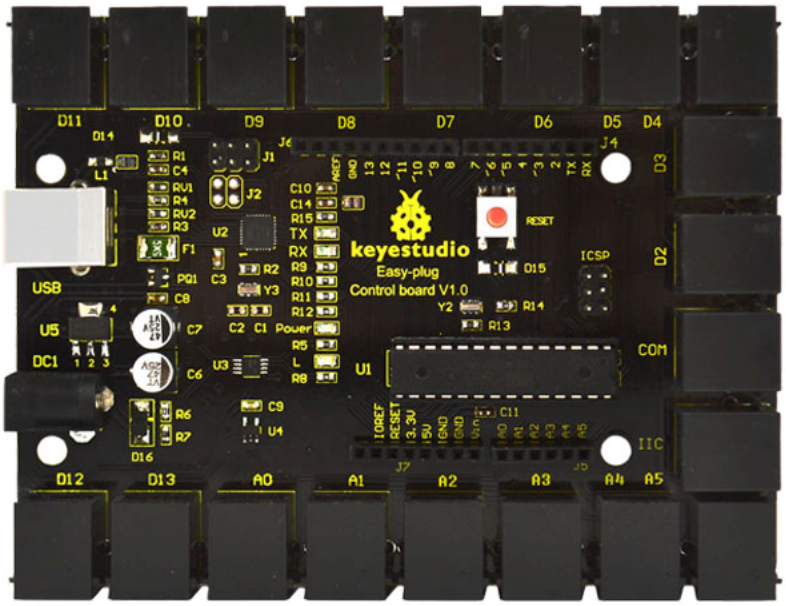
- https://github.com/cloudmesh/cloudmesh-pi/tree/master/cloudmesh/pi

2.7 Easy Plug ☁

Copied from:

- http://wiki.keyestudio.com/index.php/Ks0099_keyestudio_EASY_plug_Control_Board

Keyestudio Easy-plug control board is a microcontroller board based on the ATmega328P-PU. It has 14 digital input/outputs (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again. For convenience of wire connection, we simplify pins GND and VCC into each plug, so you only need one wire to connect a module, no need to separately connect the VCC and GND. The pins on the original UNO are all redesigned into plug interface. On the board, you can find ports D2-D13, A0 to A5, an IIC port and a COM port. All in one simple plug.



2.7.1 SPECIFICATIONS

Microcontroller core	**ATmega328P-PU**
Working voltage	+5V
External input voltage	$+7V - +12V$ (suggested)
External input voltage (externum)	$+6V \leq Vin \leq +20V$

Microcontroller core	ATmega328P-PU
Digital signal I/O interface	14 (of which 6 provide PWM output)
Analog signal input interface	6
DCI/O interface current	20 mA
FlashMemory	32KB (ATmega328) of which 0.5 KB used by bootloader
SRAM static storage capacity	2KB
EEPROM storage capacity	1 KB
EEPROM storage capacity	16 MHz

2.7.2 Connect

Tools -> Arduino/Genuine Arduino

port oon OSX will lock something like this:

- /dev/cu.usbmodem1461

2.7.3 Test code

```
int command;
int port;

int pin_from = 5;
int pin_to = 13;

void Light(int pin){
  digitalWrite(pin,HIGH);
  delay(500);
  digitalWrite(pin,LOW);
}

void setup() {
  Serial.begin(9600);
  int i;
  for (i = pin_from; i <= pin_to; i++){
    pinMode(i,OUTPUT);
  }
}


void loop() {
 command=Serial.read();
  if(command=='a') {
    int i;
    for (i = pin_from; i <= pin_to; i++){
      Light(i);
      Serial.print("Led ");
      Serial.println(i);
      delay(100);
```

```
    }
  }
}
```

2.7.4 Kit List

- http://www.keyestudio.com/keyestudio-easy-plug-learning-kit-for-arduino-super-makers.html

Part	Number
EASY plug controller Board	1
Acrylic Board + Copper bush set	1
EASY plug cable	3
USB cable	1
EASY plug Piranha LED Module	3
EASY plug Line Tracking Sensor	1
EASY plug Infrared obstacle avoidance sensor	1
EASY plug Photo Interrupter Module	1
EASY plug PIR Motion Sensor	1

Part	Number
EASY plug DS18B20 Temperature Sensor	1
EASY plug IR Receiver Module	1
EASY plug IR Transmitter Module	1
EASY plug Single Relay Module	1
EASY plug ADXL345 Three Axis Acceleration Module	1
EASY plug DHT11 Temperature and Humidity Sensor	1
EASY plug DS3231 Clock Module	1
EASY plug Analog Gas Sensor	1
EASY plug Analog Alcohol Sensor	1
EASY plug MQ135 Air Quality Sensor	1
EASY plug BMP180 Barometric Pressure Sensor	1
EASY plug Bluetooth Module	1
EASY plug 1602 I2C Module	1
EASY plug I2C 8x8 LED Matrix	1

2.7.5 Command Language

on PORT

- switches PORT on

off PORT

- switches port off

on all

- switches all ports on

off all

- switches all ports off

dance

- goes serially through ports and switches them on and off

```
String command;


int pin_from = 5;
int pin_to = 13;

String getValue(String data, char separator, int index)
{
  // copied from internet
    int found = 0;
    int strIndex[] = { 0, -1 };
    int maxIndex = data.length() - 1;

    for (int i = 0; i <= maxIndex && found <= index; i++) {
        if (data.charAt(i) == separator || i == maxIndex) {
            found++;
            strIndex[0] = strIndex[1] + 1;
            strIndex[1] = (i == maxIndex) ? i+1 : i;
        }
    }
    return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}

void Light(int pin, int action){
  if (action ==  1) {
    digitalWrite(pin,HIGH);
  } else {
    digitalWrite(pin,LOW);
  }
}

void wait_for_input() {
  while (Serial.available()==0) { }
}

void setup() {
  Serial.begin(9600);
  int i;
  for (i = pin_from; i <= pin_to; i++){
    pinMode(i,OUTPUT);
  }
}


void loop() {

  Serial.print("command:");
  wait_for_input();
  command=Serial.readString();
  Serial.println (command);

   if (command=="dance") {
      for (int i = pin_from; i <= pin_to; i++) {
        Light(i,1);
        delay(100);
        Light(i,0);
        Serial.print("Led ");
        Serial.println(i);
        delay(100);
      }
    } else {

      int action;
      String action_name = getValue(command, ' ', 0);
      String port_name = getValue(command, ' ', 1);

      action = action_name == "on";

      if (port_name == "all") {
        for (int i = pin_from; i <= pin_to; i++){
          Light(i,action);
          Serial.print("Led ");
          Serial.println(i);
        }
      } else {
        int port = port_name.toInt();
```

```
        Serial.println(action);
        Serial.println(port);
        Light(port, action);
      }
    }
}
```

2.8 ESP8266 ☁

When working with a external hardware such as the NodeMCU you will find a lot of information on the internet about it. It is a bit difficult at times to assess what you need to program it. You are exposed to many choices. A NodeMCU typically comes with Lua. However you have many other choices. Such choices include multiple programming languages such as Lua, MicroPython, Arduino/C, Go and others.

As all of them are slightly different you need to identify which works best for you. In addition you need to install images, programs and libraries that support your specific language choice.

For our first experiments we will be using MicroPython. This choice is motivated by the fact that Python is a well established and easy to learn programming language. Recently many educational institutions are offering Python as an introductory programming language making this choice even mor compelling

To simplify the setup and use of the esp8266 for MicroPython we developed an easy to use commandline tool that allows users to set up their computer and interact more easily with the board. We believe that the interface is so simple that it can also be used in STEM activities and not just in the university or by advanced hobbyists.

2.8.0.1 Installation

In this section we discuss the various ways on how to set up the esp8266 `cloudmesh.robot` development environment. You have several options to install it.

1. Option A: OSX with scripts hosted on github (recommended)
2. Option B: OSX from source
3. Option C: Explore your own

While we provide here a detailed option for OSX, you are free to explore other operating systems. We know that it can for example be installed on Ubuntu 16.04. We have not tested any of this on a Windows machine.

We like to get feedback and Installation instructions.

2.8.0.1.1 Option A: OSX install from a Script

For OSX we have created two scripts that you will need

- system.sh, that installs pip, ansible, homebrew, xcode, virtualenv, readline, wget, lua, picocom, mosquito, aquamacs, pycharm, numpy, matplotlib, libusb, USB drivers for selkected esp8266 (ch34x chip)
- user.sh, that installs matplotlib, virtualenv, and the cloudmesh source in ~/github

We recommend that you review these scripts carefully before you use them and check if they fit your needs. If they do not, please just download t hem and adapt them to your needs. The **system** script must be ran on an **Administrator** account as it requires sudo privileges. The **user** script must be ran on a **User** account. We do not recommend to run the IoT software in an administrative account due to security best practices. To execute the **system** script, type in the *Administrator account* terminal

```
$ curl -fsSL http://cloudmesh.github.io/get/robot/osx/system | sh
```

As earlier versions of pip may have some issues, this script will also update pip and setuptools to a newer version

To execute the **user** script, type in the User account terminal

```
$ curl -fsSL http://cloudmesh.github.io/get/robot/osx/user | sh
```

Together these scripts allow you to install in a simple way development tools for our IoT activities.

The following steps are to be executed in the user environment.

Warning: *the scripts do not update pip and setuptools, which may be required due to a bug in setuptools prior to version 34 for setuptools. You may have to*

repeat the update on any pyenv environment that you use. How to do thi sis documented in a later section.

To simplify use, we recommend that you make the following additions to your `~/.bash_profile` file so that python 3 is automatically activated, but does not interfere with the system installed python. Use the command

```
$ emacs ~/.bash_profile
```

or your favourite editor to edit the file and add the following lines at the end.

```
#####################################################################
# PYENV
#####################################################################
open_emacs() {
    # open -na Aquamacs $*
    open -a Aquamacs $*
}
alias e=open_emacs

#####################################################
# PYENV
#####################################################
export PYENV_VIRTUALENV_DISABLE_PROMPT=0
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"
__pyenv_version_ps1() {
    local ret=$?;
    output=$(pyenv version-name)
    if [[ ! -z $output ]]; then
        echo -n "($output)"
    fi
    return $ret;
}
PS1="\$(__pyenv_version_ps1) ${PS1}"
alias ENV3="pyenv activate ENV3"
ENV3
```

Once you start a new terminal you can edit files via aquamacs by typing

```
$ e FILENAME
```

where FILENAME is the name of the file you like to edit. However the file must exists, which you can simply do with

```
touch FILENAME
```

Add the following lines at the end of the file

To learn more about how to you automate the setup of an OSX machine, you may be inspired by

- https://github.com/ricbra/dotfiles/blob/master/bin/setup_osx
- https://blog.vandenbrand.org/2016/01/04/how-to-automate-your-mac-os-x-setup-with-ansible

2.8.0.1.2 Setting Up Git

Sooner or later you will be using git. We recommend that you set your identity on all computers that you will be using. To do this adapt the following example according to your github.com identity that you have. IF you do not, its time to greater one at github.com and follow the directions.

```
$ git config --global user.name "Gregor von Laszewski"
$ git config --global user.email laszewski@gmail.com
$ git config --global core.editor emacs
$ git config --global push.default matching
```

2.8.0.2 Option B: setup from pip

We have removed the pip setup instructions as they do not include installing the drivers.

2.8.0.2.1 Option B: Install Cloudmesh Robot from source

Developers that already have a development environment (e.g. xcode is installed) can install cloudmesh robot also from the terminal while downloading the source. You will need to first obtain the source and compile it with the following commands:

```
$ mkdir github
$ cd github
$ git clone https://github.com/cloudmesh/cloudmesh.common.git
$ git clone https://github.com/cloudmesh/cloudmesh.cmd5.git
$ git clone https://github.com/cloudmesh/cloudmesh.robot.git
$ cd cloudmesh.robot
$ make source
```

To test out if the command has been installed, type

```
$ cms robot welcome
```

If everything works you should see an ASCII image of R2D2 and C3PO. Next, we still have to install some additional programs before you can use other commands.

Once you have installed cloudmesh robots you well be able to install a number of tools automatically with the command

```
$ cms robot osx install
```

This will install services and tools including xcode, homebrew, macdown, pycharm, and aquamacs. If you have some these tools already installed it will

skip the Installation process for a particular tool. Please note that some of the tools require root access and thus you must be able to have access to sudo to run them from our tool. In addition you will need to install the OSX driver for the USB interface to the esp8266. This is achieved with (only to be done if you follow the install from source option)

```
$ cms robot osx driver
```

Now please change your account to be again a standard account.

Now you **MUST REBOOT** the machine. Without rebooting you will not be able to use the USB drivers.

2.8.0.3 Option C: A possible setup for Linux

On a linux computer we recommend that you install emacs, cmake and configure your git. Replace the user name and e-mail with the once that you used to register your account in git:

```
$ mkdir github
$ cd github
$ git clone https://github.com/cloudmesh/cloudmesh.robot.git
$ ssh-keygen
$ sudo apt-get install -y emacs
$ sudo apt-get install -y cmake
$ sudo apt-get install -y libqt4-dev
$ git config --global user.name "Gregor von Laszewski"
$ git config --global user.email laszewski@gmail.com
$ git config --global core.editor emacs
$ git config --global push.default matching
```

This setup is highly incomplete and does not include the setup of the USB drivers. Please help us completing the documentation.

2.8.0.3.1 Option C: A possible setup for Windows

We do not have tried to set this up on Windows or a virtualbox running Linux under windows. If you have tried it, please let us know. If you have difficulties just use a raspberry PI and skip the IoT projects

2.8.0.3.2 Option C: Installation of the cloudmesh.robot Interface via Pip

.. warning:: this option does not include installing the USB drivers. You have to install them first. See examples on how to do that in our install scripts. Generally what we do in our user.sh script is the same way, but also includes the setup of python 3.6.1.

To more easily interface with the robot we have developed a convenient program that is installed as part of a command tool called cloudmesh.

2.8.0.3.3 Install Cloudmesh Robot with Pip (not working)

Note that pip may not include the newest version of cloudmesh.robot and we recommend you use the source install instead.

```
$ pip install cloudmesh.robot
```

This will install a program `cms` on your computer that allows you to easily communicate with the robot.

2.8.0.4 Using cloudmesh robot

Once you have successfully installed the drivers and the commands you can look at the manual page of the robot command with

```
$ cms robot help
```

You will see a manual page like this:

```
Usage:
   robot welcome
   robot osx install
   robot osx driver
   robot image fetch
   robot probe [--format=FORMAT]
   robot flash erase [--dryrun]
   robot flash python [--dryrun]
   robot test
   robot run PROGRAM
   robot credentials set SSID USERNAME PASSWORD
   robot credentials put
   robot credentials list
   robot login
   robot set PORT NOT IMPLEMENTED
   robot ls [PATH]
   robot put [-o] SOURCE [DESTINATION]
   robot get PATH
   robot rm PATH
   robot rmdir PATH
   robot dance FILE IPS
   robot inventory list [--cat] [--path=PATH] [ID]
```

2.8.0.4.1 Testing the board

Before you can use you ESP8266, you must have the appropriate drivers installed on your computer. Click on this link and follow the instructions on how to install these drivers.

Next is to connect a esp8266 with a USB cable to the computer. The ESP8266 should look similar to this.

After you connected it, press the reset button. Before doing anything on the board, we must test it. Once you have plugged it in, execute the following command:

```
$ cms robot probe
```

This command takes about ten seconds to execute. The ESP8266's led should flash irregularly as it is probed. When the probe is finished, an image similar to the following should appear in your terminal:

```
+-----------+--------------------------+
| Attribute | Value                    |
+-----------+--------------------------+
| chipid    | b' 0x00d0f9ec'           |
| mac       | b' 00:10:FA:6E:38:4A'    |
| tty       | /dev/tty.wchusbserial1410 |
+-----------+--------------------------+
```

Please note that you should only have one board attached to your computer.

2.8.0.4.2 Flashing the image onto the robot board

Next we need to flash the image on the robot board. Naturally we need to fetch the image first from the internet. We do this with the command

```
$ cms robot image fetch
```

This will fetch an image that contains MicroPython into your local directory.

Next we need to *flash* the image on the board.

Before you begin, make sure that the ESP8266 is connected to your computer. The board may come with a pre-installed image such as Lua or some custom image from the vendor. In order to write programs in python, we need to the chips to run micropython. To get micropython on our ESP8266's, a number of steps are required.

2.8.0.4.3 Erase the chip

First we need to erase the chip.

Run the following command in your terminal terminal, and then **stop.**

```
$ cms robot flash erase
```

Your terminal should respond with the following query:

```
/dev/tty.SLAB_USBtoUART
Please press the right buttons
continue? (Y/n)
```

Before taking any further steps, press both buttons on the ESP8266 at the same time. Once you have done this, type `Y` and press `enter`. The process should take under ten seconds to complete.

2.8.0.4.4 Putting Python on the chip

Before proceeding, you must once again press both of the buttons on the ESP8266. Once this is done, you are ready to flash the chip with python with the following command:

```
$ cms robot flash python
```

2.8.0.4.5 Testing if it works

To test running a python program execute

```
$ cms robot test
```

Be careful as it overwrites the file `test.py`. If the ESP8266 is set up properly, it should return this in your terminal:

```
Count to 3
1
2
3
```

2.8.0.4.6 Execute an arbitrary program

Lets assume you have placed a program in the file `prg.py` with the command

```
$ cms robot put prg.py
```

You must reboot the ESP8266 before using a new program. This can be done manually by pressing the reset button on the chip, or in terminal with the command

```
$ cms robot reset
```

Once the chip is reset, you can run `prg.py` with the following command:

```
$ cms robot run prg.py
```

2.8.0.4.7 Interactive Python shell on the board

To get into the interactive python shell on the board you need to reset the ESP8266 and run the following command:

```
$ cms robot login
```

2.8.0.4.8 Cleaning an reinstalling a development version

IN case you are a developer and you need to modify the source code, we found that it is sometimes necessary to clean your development directory and libraries. The easiest way to do this is to go to the repository that you like to reinstall. Let us assume it is *cloudmesh.robot*. Than the following commands will clean the repository

```
$ cd cloudmesh.robot
$ pip uninstall cloudmesh.robot
```

Do the pip uninstall as many times till you see an error that no more cloudmesh.robot versions can be found. Than execute

```
$ make clean
```

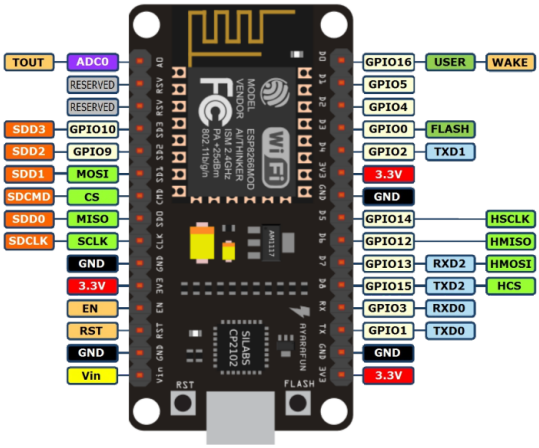
After this you can reinstall it with

```
$ python setup.py install; pip install -e .
```

the -e flag is optional, but allows you to change the code without the need of recompiling. A very useful feature in python.

2.8.0.4.9 NodeMCU ESP12 Dev Kit Pin Definition

For V1.0



nodemcu

The GPIO numbers of teh NodeMCU, do not correspond with the actual numbers used in micropython's pin library. The numbers are as follows:

Pin/GPIO	NodeMCU
15	D8

2.8.0.4.10 LED

2.8.0.4.10.1 Program

`python import machine led = machine.Pin(15,machine.Pin.OUT) led.high() led.low()

import machine led = machine.Pin(15,machine.Pin.OUT) while True: led.high() time.sleep(0.5) led.low() time.sleep(0.5)

schema
schema

schema
schema

2.8.0.4.10.2 Real Time Clock

Get the library urtc.py:

```
$ wget https://raw.githubusercontent.com/adafruit/Adafruit-uRTC/master/urtc.py
```

Place it on the esp8266

```
$ cms robot put urtc.py
```

Connect the board the following pins

```
SDA to pin 5 = D1
SCL to pin 4 = D2
```

Login to the board

```
$ cms robot login
```

Execute the following code

```
import machine
i2c = machine.I2C(sda=machine.Pin(5), scl=machine.Pin(4))
i2c.scan()

>       [87, 104]

from urtc import DS3231
t = DS3231(i2c)
t.datetime()

>               DateTimeTuple(year=2000, month=1, day=1,
>                                               weekday=1, hour=0, minute=15,
>                                               second=53, millisecond=None)
```

Assignment: Create an object oriented class and fill out the details while using code from `urtc.py`

```
class Clock (object):
    def __init__(self, sda=5, scl=4):
        pass
    def get(self):
        pass
    def __str__(self):
        pass

c = Clock()
print (c.get())
print (c)
```

2.8.0.5 Resources

```
$ https://github.com/adafruit/Adafruit-uRTC/blob/master/urtc.py
$ git clone https://github.com/adafruit/Adafruit-uRTC.git
```

2.8.0.6 Alternative boards

2.8.0.6.1 HUZZAH Feather esp8266

Many different 8266 based alternative boards exist. One of these boards is the HUZZAH Feather. IT behaves the same as the other boards, but ay be using

different drivers and USB ports. The *cms robot* command line tool is clever enough to identify automatically if it is attached and uses the appropriate settings. More documentation about this board can be found at

- doc

This site has also many other examples and you can search for them with keywords such as feather, esp8266, micropython.

An example on how to use the LED on the *feather* is documented at

- Feather HUZZAH ESP8266

To place micropython on the feather you can plug in the to the usb port. The good thing about this board is that you do not need to press any buttons as it detects the upload nicely. If not make sure to reset it or for flashing press both buttons. You can do the following:

Probe the board with:

```
$ cms robot probe
```

Erasing the feather is simple as it has a build in mechanism to detect if it is going to be erased. Hence no reset button needs to be pressed:

```
$ cms robot flash erase
```

Get the python image:

```
$ cms robot fetch python
```

Flashing is conducted with 460800 baud, it will take about 15 seconds. After flashing you should try to login:

```
$ cms robot login
```

Set the boudrate to 115200:

```
    CTRL-A CTRL-B>

    *** baud:
    ```

type in:

115200 <ENTER>
```
```

```
Make sure that echo is switched to OFF:

CTRL-A CTRL-C
```

toggles it. Now you should see:

```
    >>>
```

Try typing in:

```
    print("Hello")
```

2.8.0.7 Appendix

2.8.0.8 Installing ESP8266 USB drivers

We provide here a section to explain which drivers we have tested on various esp8266. Please note that if you have different versions you may need different drivers. On OSX we found that we get good results with the following commands

```
$ brew tap mengbo/ch340g-ch34g-ch34x-mac-os-x-driver https://github.com/mengbo/ch340g-ch34g-ch34x-mac-os-x-driver
$ brew cask install wch-ch34x-usb-serial-driver
```

Start a new terminal after the driver has finished installing.

```
$ wget http://www.silabs.com/Support%20Documents/Software/Mac_OSX_VCP_Driver.zip
$ unzip Mac_OSX_VCP_Driver.zip
```

Click on the driver install file contained inside the zip file and the driver should start installing. Once the driver has finished installing, make sure to start a new terminal.

Please remember that you need to close all terminals, as well as reboot the computer to use the drivers. They will typically not work if you have not rebooted.

For other boards that also use the CH340G chip the following page may help:

- http://kig.re/2014/12/31/how-to-use-arduino-nano-mini-pro-with-CH340G-on-mac-osx-yosemite.html

2.9 PROJECTS ☁

Please see the introduction to the IoT section to get started.

Term project suggestion combining IoT and Big Data:

1. Recognizing street sign in a car robot with a camera
2. Recognizing street lines in a car robot with camera
3. Driving a Robot car swarm without collisions
4. Simulating a City with robot cars
5. Control a robot fish with cameras
6. Build a distributed sensor system (with your classmates)

Drones:

1. Control a drone swarm with positioning system

Suggest your own

2.10 Turtle Graphics ☁️

Python comes with a nice demonstration program that allows you to learn some simple programming concepts while moving a turtle on the screen. It can be started with

```
python -m turtledemo
```

2.10.1 Program example

You can also create programs with your favorite editor and run it. Let us put the following code into the program `turtle_demo.py`. Never save a file with the name `turtle.py` because python will import it instead of the built-in turtle import that you need.

```
import turtle

window = turtle.Screen()
robot = turtle.Turtle()

robot.forward(50)   # Moves forward 50 pixels
robot.right(90)     # Rotate clockwise by 90 degrees

robot.forward(50)
robot.right(90)
```

```
robot.forward(50)
robot.right(90)

robot.forward(50)
robot.right(90)

turtle.done()

window.mainloop()
```

After saving it you can run it from a terminal with

```
$ python turtle_demo.py
```

2.10.2 SHAPE

```
shapes: "arrow", "turtle", "circle", "square", "triangle", "classic"
```

You can change the shape of your turtle to any of these shapes with the Turtle method `shape(name)`. For example, if you have an instance of the Turtle class called `robot`, you can make it appear as a turtle by calling `robot.shape("turtle")`.

You can add your own shapes with the following functions:

```
turtle.register_shape(name, shape=None)

turtle.addshape(name, shape=None)
```

There are three different ways to call this function:

name is the name of a gif-file and shape is None: Install the corresponding image shape.

```
window.register_shape("turtle.gif")
```

Note: Image shapes do not rotate when turning the turtle, so they do not display the heading of the turtle!

name is an arbitrary string and shape is a tuple of pairs of coordinates: Install the corresponding polygon shape.

```
window.register_shape("triangle", ((5,-3), (0,5), (-5,-3)))
```

name is an arbitrary string and shape is a (compound) Shape object: Install the corresponding compound shape.

Add a turtle shape to TurtleScreen's shapelist. Only the registered shapes can be used by issuing the command shape(shapename).

2.10.3 Links

- http://openbookproject.net/thinkcs/python/english3e/hello_little_turtles.html
- https://docs.python.org/3/library/turtle.html

2.10.4 Robot Dance Simulator

```
cms robot dance dance.txt
```

2.10.5 Scratch

- Scratch

2.10.6 MBlock

- MBlock

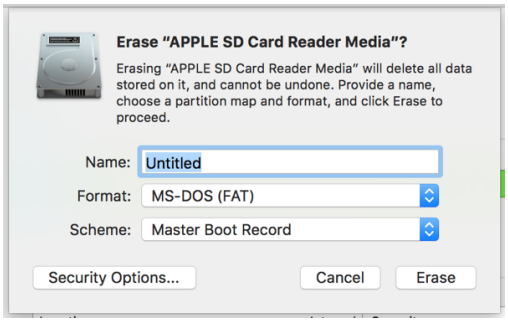
2.11 Raspberry PI 3 ☁️

2.11.1 Installation

2.11.1.1 Erasing the SD Card

Before you can install an OS on your sd card, you must erase it and put it in the proper format.

1. Insert your sd card into your micro-sd adapter and open Disk Utility with a spotlight search.
2. In the Disk Utility, right click the name of the sd card and select erase.
3. Name the sd card and format it as MS-DOS (FAT). Then click erase.



image

4. If it does not erase the first time, try again. It sometimes takes multiple tries to work.

2.11.1.2 Installation of NOOBS

NOOBS is an OS that includes Raspbian. The official description of Raspbian can be found here. It comes pre-packaged with many useful programming tools, and is easy to use.

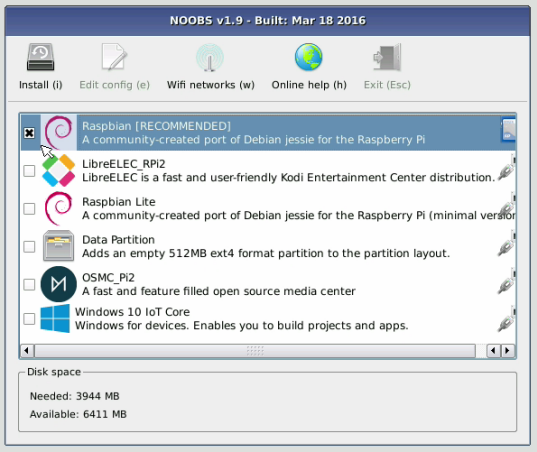
1. Download Noobs here. This will take around 30 minutes.

2. Go to your Finder and in Downloads, search for NOOBS.

3. Open the NOOBS folder and drag its contents into the sd card in the devices section. There should be 20 files and folders in the NOOBS folder. The download should take about 3 minutes.

4. Once installed, eject the sd card and put it in your raspberry pi.

5. Power up your raspberry and you will see a menu like this



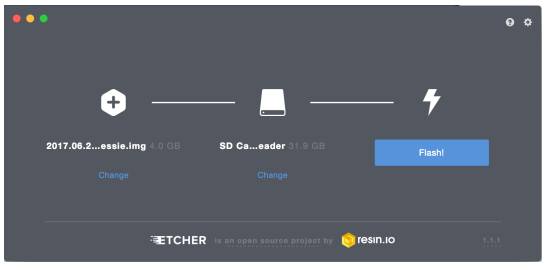
Noobs

1. Select Raspbian and click `Install (i)`

2.11.1.3 Installation of Dexter

The version of Dexter that you want to flash onto your sd card is called Raspbian for Robots. This is a Raspbian based os that is compatible with the GrovePi board. It also comes with pre-installed Dexter Industries software.

1. First, download the most recent Dexter_Industries_jessie.zip file from here.
2. Once the file has downloaded, uncompress it and insert your sd card into the micro-sd adapter.
3. Open etcher and flash the uncompressed jessie image onto the sd card.



Etcher

1. Eject your sd card and insert it into your raspberry pi.

2.11.2 CONFIGURE

2.11.2.1 Prepare OS

2.11.3 UPDATE

The following are essential updates:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install emacs
dpkg -l > ~/Desktop/packages.list
pip freeze > ~/Desktop/pip-freeze-initial.list
```

The following are necessary for the scientific libraries, but they require lots of space. Our sd cards do not have enough space for them.

```
sudo apt-get install build-essential python-dev python-distlib python-setuptools python-pip python-wheel libzmq-dev l
sudo apt-get install xsel xclip libxml2-dev libxslt-dev python-lxml python-h5py python-numexpr python-dateutil python-
sudo pip install bottleneck rtree
```

add to .bashrc

```
cd
git clone git://github.com/yyuu/pyenv.git .pyenv
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(pyenv init -)"' >> ~/.bashrc
source ~/.bashrc

export PATH="/home/pi/.pyenv/bin:$PATH"
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"

curl -L https://raw.githubusercontent.com/pyenv/pyenv-installer/master/bin/pyenv-installer | bash
```

source

2.11.3.1 Update to Python 3.6.1

2.11.4 CHANGE PYTHON VERSION

- [https://linuxconfig.org/how-to-change-from-default-to-alternative-python-version-on-debian-linux] (https://linuxconfig.org/how-to-change-from-default-to-alternative-python-version-on-debian-linux)

Upgrade setuptools for pip install with

```
    $ pip3 install --upgrade setuptools
```

Test your python version with

```
    $ python --version
```

Check your python version alternatives

```
$ update-alternatives --list python
```

If python2.7 is not in your alternatives, add it with

```
$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python2.7 1
```

If python3.4 is not in your alternatives, add it with

```
$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python3.4 2
```

Now make python3.4 to your default with

```
update-alternatives --config python
```

Select python3.4

2.11.5 INSTALL 3.6.1

To install python 3.6.1, follow steps 1 and 2. This is unnecessary for our purposes.

- better get 3.6.1

2.11.6 INSTALL CLOUDMESH-PI

pip install cloudmesh-pi

pip install cloudmesh-pi with

```
$ git clone https://github.com/cloudmesh/cloudmesh-pi.git
$ cd cloudmesh-pi
$ sudo pip3 install .
```

see how we do this in macOS/linux can this be done on raspberry? if not document update from source with altinstall

2.11.6.1 Install scientific Libraries

check if they are already installed we do not have enough space to install all of these.

```
sudo apt-get install python-numpy python-matplotlib python-scipy python-sklearn python-pandas
```

numpy
matplotlib
scipy
scikitlearn

2.11.6.2 cloudmesh-pi

cloudmesh-pi is a repository for our GrovePi module classes. These classes require Dexter software, so you need to either have Raspbian for Robots or download the software separately.

If you have Raspbian for Robots, run the following in your terminal:

```
cd
mkdir github
cd github
git clone https://github.com/cloudmesh/cloudmesh-pi.git
cd cloudmesh-pi
sudo pip install .
```

2.11.6.3 Install VNC

describe how to install and configure VNC

2.11.7 Sensors

2.11.7.1 Grove Sensors ⭕

⊗
No

we already have draft

2.11.7.2 Non Grove Sensors ⭕

⊗
No

Elegoo as example

2.11.8 Notes To integrates

2.11.8.1 Connecting

Hostnames:

- raspberrypi.local
- raspberrypi.

change

recovery.cmdline

forgot what these were:

```
runinstaller quiet ramdisk_size=32768 root=/dev/ram0 init=/init vt.cur_default=1 elevator=deadline
silentinstall runinstaller quiet ramdisk_size=32768 root=/dev/ram0 init=/init vt.cur_default=1 elevator=deadline
```

Connect the cable

You will see the activity LEDs flash while the OS installs. Depending on your SD-Card this can take up to 40-60 minutes.

2.11.9 VLC on macOS

- http://www.videolan.org/vlc/index.en_GB.html
- http://get.videolan.org/vlc/2.2.6/macosx/vlc-2.2.6.dmg
- http://www.mybigideas.co.uk/RPi/RPiCamera/
- Camera on Pi — — — —

  sudo apt-get install vlc

- https://www.raspberrypi.org/learning/getting-started-with-picamera/worksheet/
- https://www.hackster.io/bestd25/pi-car-016e66

2.11.10 Streaming video

- https://blog.miguelgrinberg.com/post/stream-video-from-the-raspberry-pi-camera-to-web-browsers-even-on-ios-and-android

2.11.11 Linux Commandline

- http://www.computerworld.com/article/2598082/linux/linux-linux-command-line-cheat-sheet.html

2.11.12 Enable SPI

go to the configuration interfaces and enable

2.11.13 RTIMUlib2

git clone https://github.com/RTIMULib/RTIMULib2.git cd RTIMULib

Add the following two lines to /etc/modules

```
i2c-bcm2708
i2c-dev
```

reboot

```
ls /dev/i2c-*
sudo apt-get install i2c-tools

sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

create a file /etc/udev/rules.d/90-i2c.rules and add the line:

```
KERNEL=="i2c-[0-7]",MODE="0666"
```

note: does not work

instead we do

```
sudo chmod 666 /dev/i2c-1
```

Set the I2C bus speed to 400KHz by adding to /boot/config.txt:

```
dtparam=i2c1_baudrate=400000
```

reboot. In terminal change directories to

```
cd /home/pi/github/RTIMULib2/RTIMULib/IMUDrivers
```

and open

```
emacs RTIMUDefs.h
```

In RTIMUDefs.h change

```
#define MPU9250_ID 0x71
```

To

```
#define MPU9250_ID 0x73
```

```
cd /home/pi/github/RTIMULib2/RTIMULib
```

In terminal

```
mkdir build
cd build
cmake ..
make -j4
sudo make install
sudo ldconfig
```

2.11.14 Compile RTIMULib Apps

```
cd /home/pi/github/RTIMULib2/Linux/RTIMULibCal
make clean; make -j4
sudo make install
cd /home/pi/github/RTIMULib2/Linux/RTIMULibDrive
make clean; make -j4
sudo make install
cd /home/pi/github/RTIMULib2/Linux/RTIMULibDrive10
make clean; make -j4
sudo make install
cd /home/pi/github/RTIMULib2/Linux/RTIMULibDrive11
make clean; make -j4
sudo make install


cd /home/pi/github/RTIMULib2/Linux/RTIMULibDemo
qmake clean
make clean
qmake
make -j4
sudo make install
cd /home/pi/github/RTIMULib2/Linux/RTIMULibDemoGL
qmake clean
make clean
qmake
make -j4
sudo make install
```

2.11.15 Camera

- Camera Tutorial

.

```
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

sudo apt-get install libxvidcore-dev libx264-dev

sudo pip install virtualenv virtualenvwrapper
sudo rm -rf ~/.cache/pip
```

copy into ~/.profile:

```
 echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
 echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
 echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
```

source ~/.profile

```
mkvirtualenv cv -p python3
```

workon cv

command line has (cv) in front

```
pip install numpy

wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip
wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip
unzip opencv.zip
unzip opencv_contrib.zip
```

2.11.16 Lessons and Projects

- Gui
- Solder
- PI Camera Line Follower
- Pi car flask

2.11.17 OTHER TO BE INTEGRATED

2.11.17.1 PI emulator on Windows

We have not yet tried it, but we like to hear back from you on experiences with

- https://sourceforge.net/projects/rpiqemuwindows/

2.11.17.2 Scratch

- scratch

2.11.18 Web Server

- Web Server Flask

3 CASE

3.1 Pi Cluster Form Factor ☁

In this chapter we will discuss a number of opportunities to build small scale compute and cloud cluster resources using Raspberry Pi's.

This includes the following:

- NAS server with one Raspberry Pi 3
- Cluster using 1 Raspberry Pi as master and 4 Raspberry Zeros as workers
- Case With Cooling for 5 Pi)
- Build an Octapi
- Cluster with 40 Raspberry Pi's
- Cluster with 144 Raspberry Pi's
- Build Your Own 5 Node Pi Cluster

3.1.1 NAS (1 Pi)

Although a NAS is not really a compute cluster the Pi has used many times to build a Network Attached Storage (NAS) server. In this configuration a HDD is attached to the Raspberry and the network features of the Raspberry is used to access the disk drive via software installed on the PI that make this easily possible. Many tutorials exists on the Web that help setting op such a device.

We like to hear from you if you have successfully developed such a NAS and provide us with such links. Links that may help include:

- https://hackmypi.com/NASpi.php

3.1.2 ClusterHat (4 Zero + 1 Pi)

The smallest cluster we came across is actually a hybrid cluster in which 4 Pi zeros attached to a Raspberry Pi 3. This sis achieved via an add on board to the Pi 3 allowing to plug in PI=i Zeros:

- https://clusterhat.com/

The Cluster HAT (Hardware Attached on Top) allows to attach 4 Raspberry Pi Zeros via to be attached to a regular Raspberry PI 3 to simulate a small cluster.

According to the Web Site it supports the following features:

- USB Gadget Mode: Ethernet and Serial Console.
- Onboard 4 port USB 2.0 hub.
- Raspberry Pi Zeros powered via Controller Pi GPIO (USB optional).
- Individual Raspberry Pi Zero power controlled via the Controller Pi GPIO (I2C).
- Connector for Controller Serial Console (FTDI Basic).
- Controller Pi can be rebooted without interrupting power to Pi Zeros (network recovers on boot).

Figure: Clusterhat for PI Zero's

Although this setup seems rather appealing, the issue is with obtaining Pi Zeros for the regional price of $5. Typically users can only by one for that price and must pay shipping. To by more one has to buy a kit for about $20. However, for that amount of money it may just be worth while to get Pi 3's instead of zero's. Nevertheless the form factor is rather appealing.

Additional information can be found at:

- https://www.raspberrypi.org/magpi/clusterhat-review-cluster-hat-kit/

3.1.3 Cluster Case With Cooling (5 Pi's)

Many instructions on the Web exist describing how to build clusters with 3 or more Pi's. One of the considerations that we have to think about is that we may run rather demanding applications on such clusters causing heat issues. To eliminate them we must provide proper cooling. In some cluster projects cooling is not adequately addressed. Hence we like to provide an example that discusses in detail how to add a fan and what the fan has for an impact on the temperature.

Closed case for 5 Pi's with case

- http://climbers.net/sbc/add-fan-raspberry-pi/
- http://climbers.net/sbc/diy-raspberry-pi-3-cluster/

From the previous Web page we find the following information as shown in the following table. From the data in the table it is clear that we need to keep the Pi from throttling while being in a case by adding a fan as obvious from experiment No. 2.

Table: Temperature comparison of fan impact

No.	Case	Fan	Direction	RPM	Idle	100% Load	Performance
1	no	no	-	-	41.0C	75.5C	OK (barely)
2	yes	no	-	-	45.0C	82.5C	throttles
3	yes	5V	in	unknown	37.9C	74.5C	OK (barely)
4	yes	7V	in	800	35.6C	69.5C	OK
5	yes	12V	in	1400	32.5C	61.1C	OK
6	yes	7V	out	800	34.5C	66.4C	OK

Interesting is also the design of the case that uses snaps instead of screws to affix the walls to each other. The case layout can be found at:

- http://climbers.net/sbc/diy-raspberry-pi-3-cluster-2017/

3.1.4 Octapi (8 Pi's)

A set of instructions on building an is available at

- https://projects.raspberrypi.org/en/projects/build-an-octapi

Octapi

3.1.5 BITSCOPE CASE (40 PI'S)

A company from Australia called BitScope Designs offers a number of cases that leverage their Pi Blade boards allowing up to four Pis to be put together and sharing the same power supply. The blades are shown in the next Figure. The rack to place 10 of them is shown in the Figure after that.

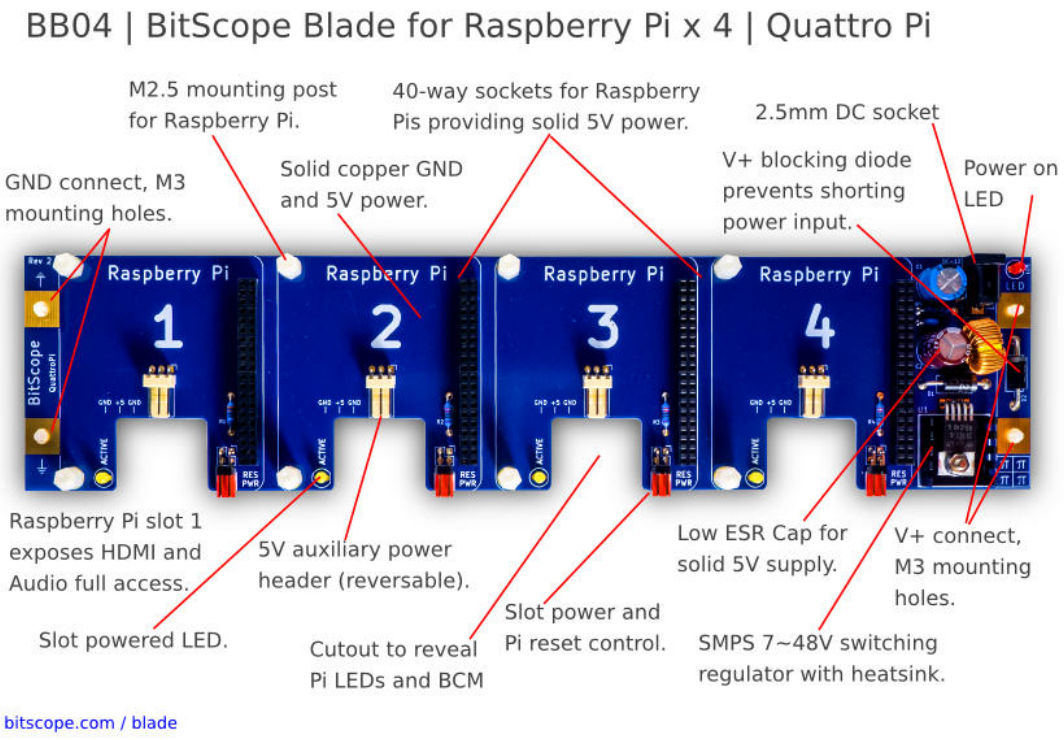


Figure: BitScope blade for 4 Pi's.

Figure: 40 Pi Blade rack.

The cost of the blade rack is $ 795.45 + $60.00 shipping + import tax. This may originally sound expensive when compared to a single case, however as we can store 40 Pis in them and they can share the power-supply and reduce cabling we think this case is quite interesting overall due to its price-point of $20 per Pi.

3.1.6 BitScope Cluster (144 Pi's)

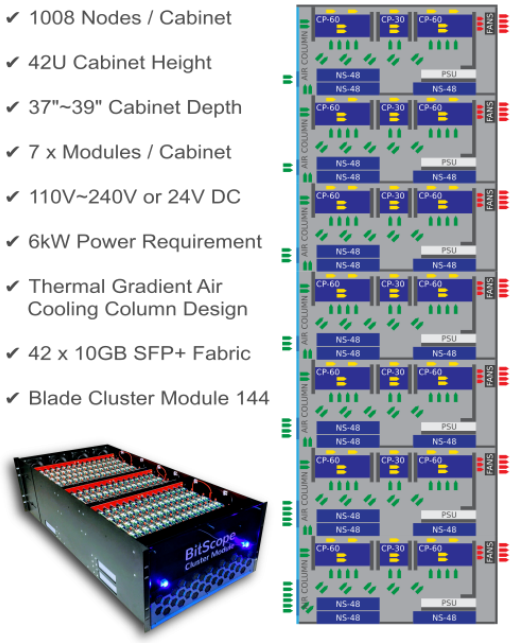
- https://www.youtube.com/watch?v=78H-4KqVvrg

Together with LANL a new cluster module that holds 144 Pis is developed. This system is targeted to be placed into a rack to create a large Pi cluster. The cost for such a module is about $15K.

The next Figure shows the module.

BitScope 144 cluster module.

The next figure shows how multiple modules can be placed into a single rack.



Rack placement of multiple BitScope 144 cluster modules.

Additional information about this form factor can be found at the following links:

- https://cluster.bitscope.com/solutions
- https://www.pcper.com/news/General-Tech/BitScope-Unveils-Raspberry-Pi-Cluster-2880-CPU-Cores-LANL-HPC-RD
- http://my.bitscope.com/store/
- http://my.bitscope.com/store/?p=view\&i=item+7

- http://www.newark.com/bitscope/bb04b/quattro-pi-board-raspberry-pi/dp/95Y0643
- http://linuxgizmos.com/rpi-expansion-boards-support-up-to-40-pi-clusters/

3.1.7 Oracle Cluster (1060 Pi's)

Oracle has displayed at Oracle World 2019 a 1060 node Raspberry Pi Cluster.

Oracle 1060 Pi Cluster source

More images are available at this (link)[https://imgur.com/gallery/wx1hZ5D]. the supercomputer features scores of racks with 21 Raspberry Pi 3 B+ boards each. The system is used to demonstrate Oracle Autonomous Linux source.

3.1.8 Build Your Own 5 Node Pi Cluster

To experiment with building an elementary cluster one does not need to have a big budget. Such clusters are often dedicated to research tasks and are bound into security protocols that do not allow direct access. Instead it is possible to build such a cluster based on Raspberry Pi's yourself if you are willing to spend the money or if you have access to Pi's that you may loan from your department.

Table Parts lists one such possible parts list that will allow you to build a cluster for up to 5 nodes. However make sure to buy at least 3 Raspberry Pi's with the appropriate memory. At minimum we recommend you get the 32GB SD card. We do not recommend any smaller as otherwise you will run out of memory. Additionally, you can add memory and disks on the USB ports. If you attach a HDD, make sure it has an external power supply and do not drive it from the USB power as otherwise the PI becomes unstable. A fan is at this time not yet included.

Naturally it is possible to modify the parts list and adapt. If you find better parts let us know. We have not included any case and you are welcome to share your suggestions with the class. For a case we are looking also for a good solution for a fan.

We suggest that when you build the cluster to do it on a table with a large white paper or board, or a tablecloth and take pictures of the various stages of the build so we can include it in this document.

Initially we just put Raspbian as Operating system on the SD cards and test out each PI. To do so you will naturally need an SD card writer that you can hook up to your computer if it does not have one. As you will have to potentially do this more than once it is not recommended to buy an SD card with the OS on it. Buy the SD card writer instead so you can redo the flashing of the card when needed. In addition to the SD card you need a USB mouse and keyboard and a monitor or TV with HDMI port.

Locate setup instructions and write a section in markdown that we will include here once it is finished. The section is to be managed on github.

.

Price	Description	URL
$29.99	Anker 60W 6-Port USB Wall Charger, PowerPort 6 for iPhone 7 / 6s / Plus, iPad Pro / Air 2 / mini, Galaxy S7 / S6 / Edge / Plus, Note 5 / 4, LG, Nexus, HTC and More	link

Price	Description	URL
$8.90	Cat 6 Ethernet Cable 1 ft White (6 Pack) - Flat Internet Network Cable - Jadaol Cat 6 Computer Cable short - Cat6 Ethernet Patch Lan Cable With	link
$19.99 [1]	D-link 8-Port Unmanaged Gigabit Switch (GO-SW-8G)	link
$10.49	SanDisk Ultra 32GB microSDHC UHS-I Card with Adapter, Grey/Red, Standard Packaging (SDSQUNC-032G-GN6MA)	link
$8.59	Short USB Cable, OKRAY 10 Pack Colorful Micro USB 2.0 Charging Data Sync Cable Cord for Samsung, Android Phone and Tablet, Nexus, HTC, Nokia, LG, Sony, Many Digital Cameras-0.66ft (7.87 Inch)	link
$7.69	50 Pcs M2 x 20mm + 5mm Hex Hexagonal Threaded Spacer Support	link
$7.99	Easycargo 15 pcs Raspberry Pi Heatsink Aluminum + Copper + 3M 8810 thermal conductive adhesive tape for cooling cooler Raspberry Pi 3, Pi 2, Pi Model B+	link
$34.49	Raspberry Pi 3 Model B Motherboard (you need at least 3 of them)	link
$59.99 [2]	1TB drive	link
$15.19	64GB flash	link
$6.99	HDMI Cable, Rankie 2-Pack 6FT Latest Standard HDMI 2.0 HDTV Cable - Supports Ethernet, 3D, 4K and Audio Return (Black) - R1108	link
$12.99	AUKEY USB C Adapter, USB C to USB 3.0 Adapter Aluminum 2 Pack for Samsung Note 8 S8 S8+, Google Pixel 2 XL, MacBook Pro, Nexus 6P 5X, LG G5 V20 (Gray)	link
$19.19	For Raspberry Pi 3 2 TFT LCD Display, kuman 3.5 Inch 480x320 TFT Touch Screen Monitor for Raspberry Pi Model B B+ A+ A Module SPI Interface with Touch Pen SC06	link

[1] items were replaced with similar [2] item was not available

3.1.8.1 Assembling the Pi Cluster ⭕

⊗

No

TODO: replace the images with one that has white background

Figure	Description

Figure	Description
	First, aluminum and copper heat syncs need to be attached to each Pi. The two aluminum heat syncs are attached to the Broadcom chip and the SMSC Ethernet controller located on the top of the Pi. The blades of the heat syncs are parallel to the longer side of the Pi as shown in black aluminum fanned heat syncs are attached to the top of the pi as shown.
	Flat copper heat sync is attached to the bottom of the pi as shown.
	After attaching the heat syncs, threaded hexagonal spacer supports are used to connect the Pis together. A fully-assembled 5-node Pi cluster is shown.

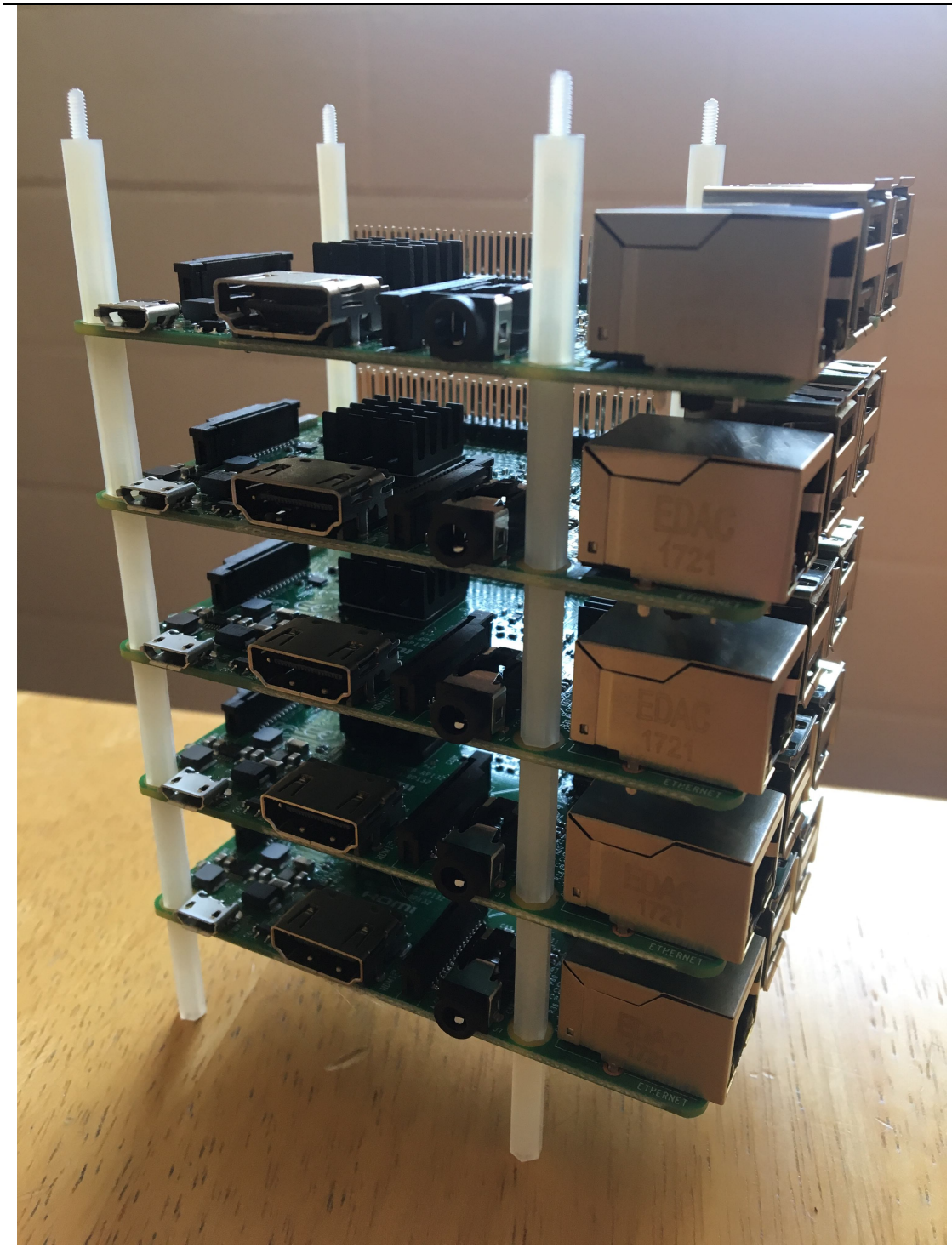
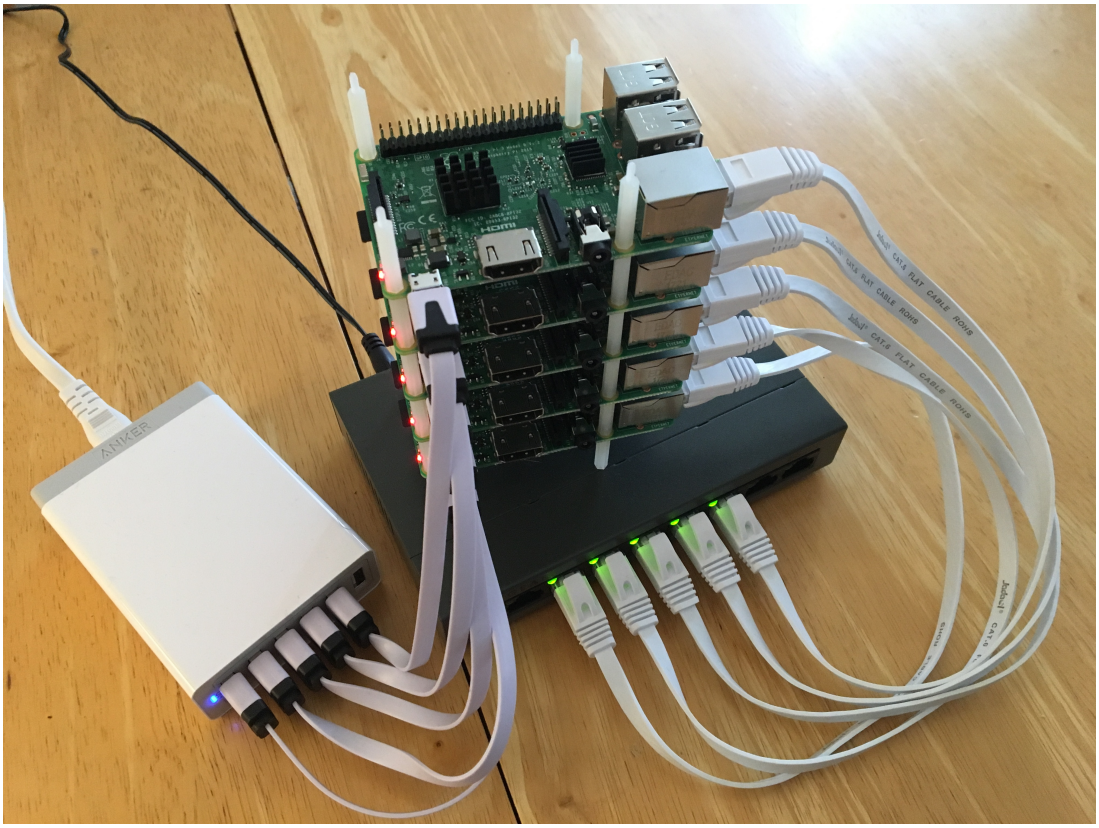
Figure	Description
	

Figure	Description
	Each node of the cluster is then attached to the switch using an Ethernet cables and to the power supply using a USB cables. The fully wired cluster is shown.

3.1.8.2 Virtual Raspberry Cluster ⭕

⊗

No

It should also be possible to create a virtual raspberry PI cluster while for example using virtual box. This requires two steps. First the deployment of a virtualized Raspberry PI. The following information may be useful for this

- http://dbakevlar.com/2015/08/emulating-a-raspberry-pi-on-virtualbox/

The next step includes the deployment of multiple VMs emulating Raspberry's. Naturally each should have its own name so you can distinguish them. Instead of just using the GUI, it would be important to find out how to start them from a command line as a shell script as well as tear them down.

Next you will need to make sure you can communicate from the Pi's to each other. This is naturally the same as on a real cluster

⚠ *TODO: provide a section*

This can be chosen as part of your project, but you need to develop a cloudmesh command for managing the cluster. This includes starting and stopping as well as check-pointing the cluster from a cloudmesh command. Furthermore you need to benchmark it and identify how to do this and contrast this to other clusters that you may start or have access to. Please get in contact with Gregor. This project is reserved for online students, as residential students will have access to real Raspberry PI hardware.

Please note that this project may have to use QEMU.

3.1.9 IU 100 Node Cluster Case ⭕ ☁

⭕

At Indiana University we have currently about 200 Raspberry Pi's available for students that work with us as part of classes.

Our goal for the use of these PI's is multitude.

First, we naturally like that all PI's could be used together in a large cluster. Second we like that student teams up to 3 students can use a small cluster of between 6 - 5 Pi's and use them to provision their own OS. Third, we like to crate a case and shelf structure that allows this modular utilization of the cluster while breaking it down easily, but also assembling or adding to the cluster easily.

This section will document our efforts to support this. The section is heavily under construction and currently two STEM students work on the case. They are joined by an independent study student offer the summer that looks at the software for this cluster.

To simplify management we will be using github for managing tasks, but we will start collecting these tasks here. At this time the STEM students are using still a google docs document till they migrate to this or their own github.

The outline of the section is as follows 🚧

First we provide an introduction to which tools to use for designing a case. This is followed by some examples that may lead to a case design. Next we provide an introduction on how to use the Laser cutter at IU. The last part of this section includes a list of links and some images of clusters designed by others.

3.1.9.1 Designing a Case with CAD

We believe the best program for us to design a case for the cluster is OpenSCAD. Links and tutorials to the software can be found here:

- http://www.openscad.org/
- http://www.openscad.org/cheatsheet/
- https://github.com/RigacciOrg/openscad-rpi-library
- https://www.youtube.com/watch?v=WQd5db9lsQk

Although OpenSCAD requires programming, it seems to be easier than creating the design with a GUI based CAD program. For this reason we will be using SCAD.

Other CAD software may be accessible to you also. However will not use for our case design. This includes the IU Maker Lab standard Software is Fusion 360 which is available from

- https://www.autodesk.com/products/fusion-360/overview

As it does require a license it may not be accessible to many students. However you can use alternative that can be used for free. This includes FreeCAD

- FreeCAD Software https://www.freecadweb.org/
- FreeCAD Documentation https://www.freecadweb.org/wiki/Main_Page

This program allows you to design the layout of a 3D case via a sophisticated GUI just like Fusion 360.

3.1.9.2 IU cluster Case design

We have designed the following cases

TBD

Right now we just put some templates here to showcase we can actually generate the cases. More details will be added here soon.

Obviously you need to be able to have some walls and shelfs for the case. The following SCAD file shows how to use a simple cube to do that. This is just a start as the design does not include any connectors.

scad file

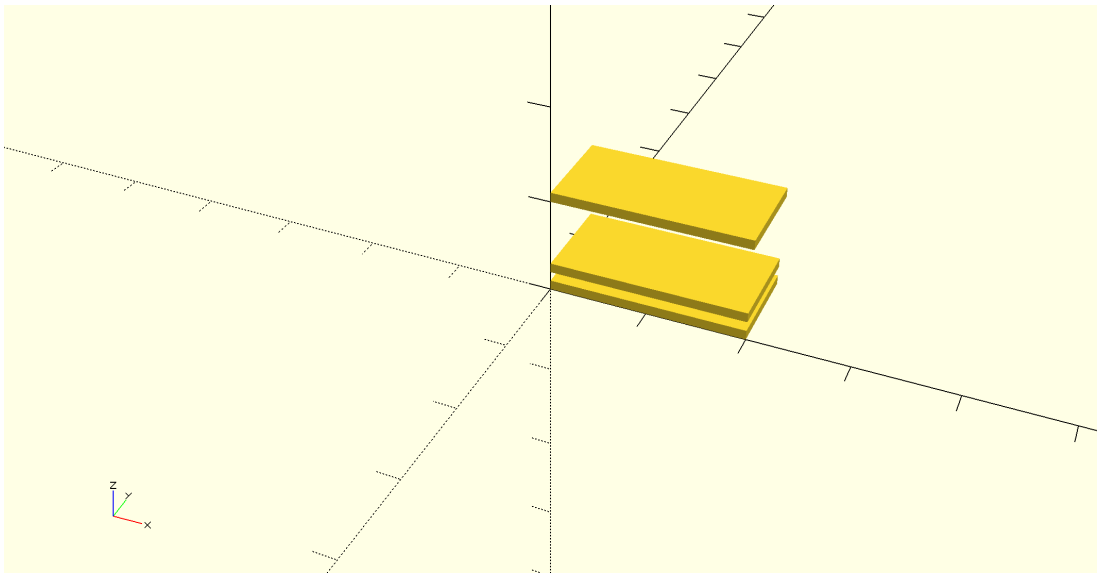


Figure: TBD

Next we show you how to use some predefined Raspberry PI boards in SCAD and use for loops to create rows and columns of 5 PIs each.

100-pis

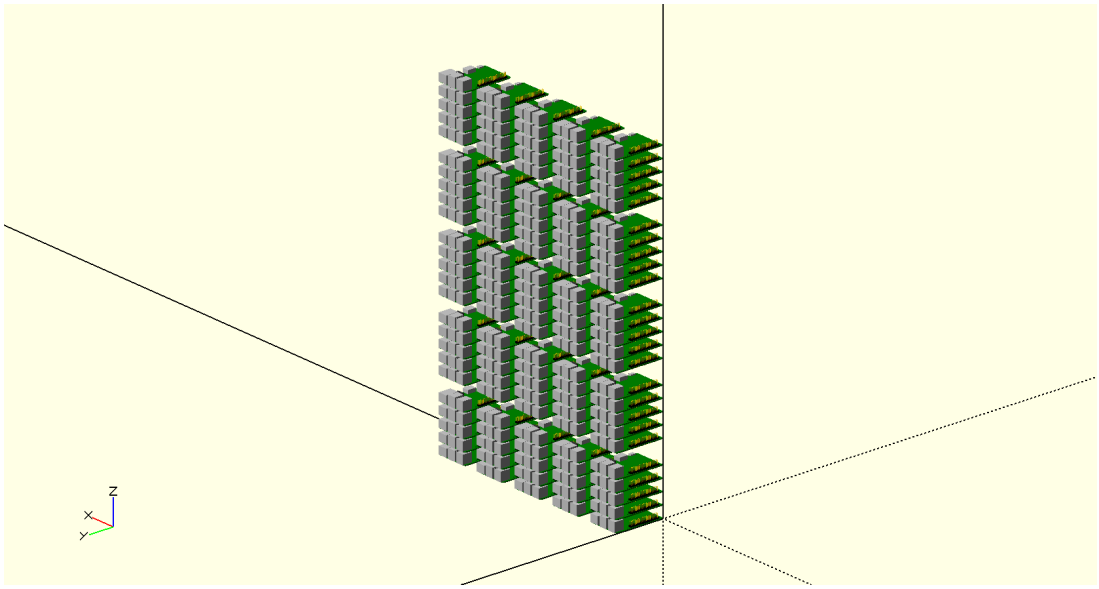


Figure: TBD

In the last design we showcase how the basis of a 5 node cluster could start. Naturally many other thing s need to be improved. Such as the connectors, and the proper width. However this SCAD will provide you with a good start to improve your SCAD programming abilities and come up with a final design.

100-pis

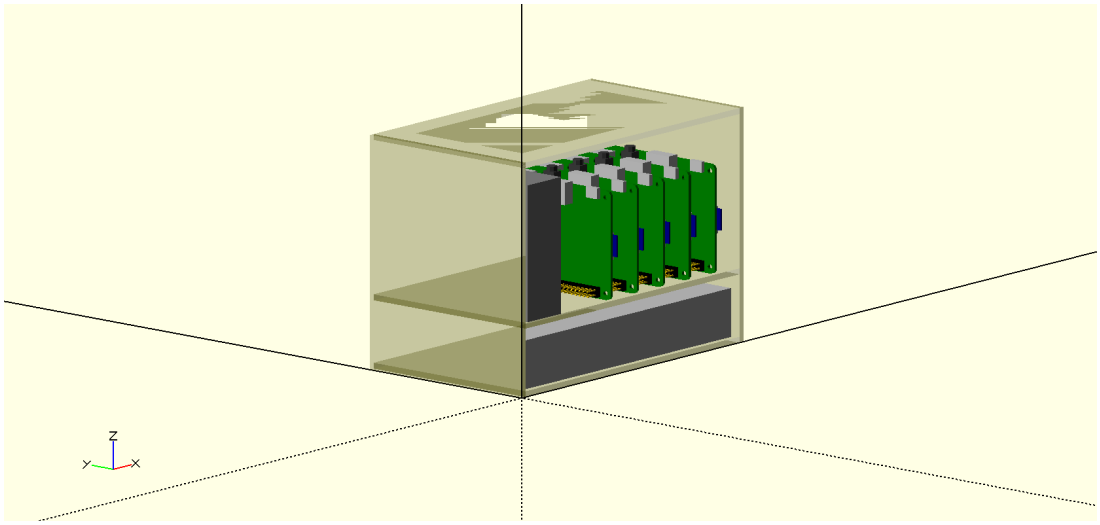


FIGURE: TBD

Once you have designed the 5 node cluster, you will also need to design a shelf in which you can place them. Power plugs need to be added. so you can power the PI within the cluster.

3.1.9.3 Laser Cutter

With a laser cutter we can create very precise cases. First, you will need an overview of what a laser cutter is. There are many good resources about this on youtube. One such video is provided by a company that sells laser cutters:

- https://www.youtube.com/watch?v=SIjUVCho_xU

Disclaimer: While pointing to this video we do not endorse the product. We do not own this particular brand.

After you have designed your case, you can create it on the laser cutter available at Indiana University. The laser cutter is located at

- TBD

and operated by ISE. You can get access to it by contacting the help desk with an email. You will need to get certified for operating the laser cutter to use it.

⭕ TODO: add contact email

The certification will teach you how to use the laser cutter properly. We will walk you through the steps that you can expect will be tought to you as part of the certification.

The first step includes turning on laser cutter.

An instructor will demonstrate how to turn the laser cutter on and carry out other preparation tasks on how to operate the laser cutter. Please get familiar but do not yet touch any buttons from the laser cutter.

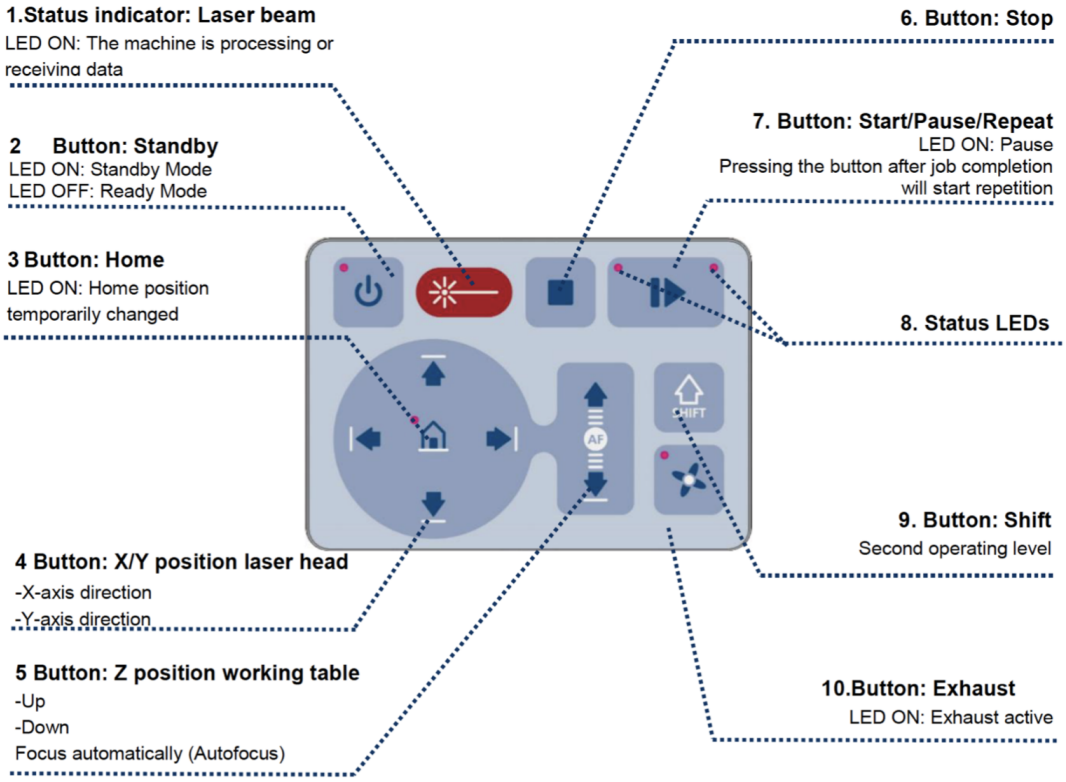


Figure: Control Panel

Next the instructor will give you a walk through of the machine so you can replicate these steps next time.

1. Turn the laser cutter on: Use key to turn on the laser cutter.
2. Turn the exhaust fan on: The fan is a separately switched unit. Turn on the switch on the wall next to the laser cutter to turn on the exhaust in order to have proper ventilation during the cutting process.
3. Put the cutting material onto the laser cutterbed. The instructor puts the cutting material onto the laser cutter bed. In case the material is smaller than the honeycomb bed you will need to place is as follows TODO: TBD

4. The buttons for the laser cutter do the following

    1. The middle top one is on-hold button, one can press it to save the power of the laser cutter.
    2. The start/pause button can be used to pause a job in the middle. Once you start from a paused state the job resumes.
    3. The stop button will cancel the job. You will need to restart the job from the beginning if this button is pressed to stop a job.
    4. The four buttons on the bottom left controls the x/y horizontal shift of the laser head. The other two controls the z-axis, or vertical movement of the cutting bed.

    Note that while the stop button stops the entire process, the pause button will halt the process so you can continue.
5. To properly cut the material, you will need to focusing the laser head. Move the laser head above the cutting material, then will place the focusing tool on the laser head. Then carefully raise the cutting bed. When the focusing tool touches the material on the bed it will fall off indicating the laser is focused. You must be careful to not raise the bed too fast in order not to damage the laser head.

6. Now we need to send the file to the job control software and initiate cutting

    1. Send the file to the Job Control by opening the saved file in Adobe Illustrator. TODO: how do we get the SCAD file into it, which format do we need. You click print and set the printer to “Trotec Laser Cutter”.
    2. Set laser cutting parameter will be set to width and height of the cutting area in the size column. By clicking the “Preferences” tab, one can adjust the settings of different colors (traditionally red lines are for cutting and black lines and areas are for engraving)

    3. Move the file to Job Control work surface by turning on the Job Control software and discover the job is in the queue on the right of screen.

    4. Next drag the file to be cut on to the Job Control work surface and set it up in the upper right corner.
    5. Select the Play arrow in JobControl

    6. An exhaust warning will appear. Select OK. The job will start.

Questions to be integrated:

- What is the importance of setting cutting parameters?
- What will you check before pressing the start button to ensure safety?

3.1.9.4 Cases and Parts Designed by the Community

A 3D model of the Raspberry PI is available at

- STL https://www.thingiverse.com/thing:1701186
- https://grabcad.com/library/raspberry-pi-3-reference-design-model-b-rpi-raspberrypi-raspberry-pi-2

These models are important as they allow us to use them in the CAD drawings to identify proper placement.

Certainly we need to identify how to connect the walls of the case. The following links provide some ideas for such connections.

- T-nut design: A convenient connector using screws ant nuts to connect walls in 90 degree angles [link]
- How To: Make Cheap LaserCut Custom Boxes for Your DIY Electronics, includes t-bolt design [link]
- A tutorial on how to make snug joints. It has some good tips of placing circles as stress relieve points [link]
- Screw only: A design to just use screws to connect the walls. The disadvantage is that the receiving ends need to be cut by a hand tool. [link]
- A sample design that just uses groves, so this would have to be glued. [link]
- Another design with groves that needs to be glued [link]
- A guide to handle acrylic [link]
- Some special brackets which could be 3 D printed. These brackets have the advantage that only slots and holes need to be placed [link].
- Although CNC, here are some connector ideas that could be replicated or modified for acrylic [link]
- Some tips for working with acrylic in general [link]
- [link]
- A design based on metal beams to create the outside. This may be more expensive, but stable [link]
- Some also used lego technics [link]

We also previously indicated that the width of the mounting holes on a Pi is about 7 spaces wide, making it possible to create a lego technic case.

3.1.10 Pi Clusters on the Internet ☁️

There is a large number of projects related to creating Pi clusters on the internet. They vary in size and software installed on them. Naturally a Pi cluster is a useful training and development environment for research organizations and thus many bigger projects are located at universities as well as government labs. However, we also have many projects done by enthusiasts.

We distinguish two different efforts. First, we often find projects that target the creation of cases for such clusters and second, we find projects that develop software fort these clusters. This section will provide an overview of these activities and provide links to these activities.

3.1.10.1 Cluster Cases

3.1.10.1.1 Lego

When looking at the placement of the wholes on the Raspberry Pi, the width between the wholes on the small side seems to be exactly 7 Lego Technic beam wholes apart. This has the advantage that one could build a quick frame form Lego pieces such as a

- 2 * 11 ($0.22) piece [link]
- 2 * two piece * 4 * 4 ($0.192) [link]
- A number of connector pins [link] [link]

The cost is about $0.25 per piece = $2.74 per pi.

So if we are having 100 pis we end up with $274. However we also need still to get screw and Lego connectors which we at this time have not counted and included in this calculation.

Naturally Lego's have been explored by others

- Images on google
- University of Southampton: [link] Instructions are no longer at the original link
- Lego Technic: [link], [link]
- Lego Technic [link]
- Old style Lego [link]

Other ideas using Lego's include:

- Compact case: Single board No screws [link]
- Zero 3D print Thingverse [link]
- B+ 3D Pi case [link]

Other interesting but not cluster related links include

- Lego hat [link]
- Brick Pi [link]

3.1.10.1.2 Beast by resion.io

This company has provided some larger designs for Raspberry Pi clusters and tries to create a modular system to put a number of Pis on plates that than can be connected.

- Beast 3 [link]
- Beast 2 [link]
- Beast 2 Tile [link]

3.1.10.2 Clusters

Additional links which could be useful include:

- SDSC Raspberry Pi2 MPI and Tiled Wall Viz Cluster [link]
- Six Common Errors When Building a Raspberry Pi Cluster [link]
- 5 Most Popular Raspberry Pi Cluster Supercomputer Projects [link]
- 10 amazing Raspberry Pi clusters [link]
- Build an OctaPi [link]

An older document on how to create an MPI cluster is located at

- Pi 2 MPI cluster, Boise State [link]

3.1.10.2.1 Swarm

How to set up docker swarm is documented here

- Pi Docker Cluster [link]

4 EXERCIZE

4.0.1 EXERCISE

4.0.1.1 Single Raspberry Pi Temperature

You have been presented in Section Cluster Case With Cooling (5 Pi) with a table that compares temperatures. Your task is to identify issues with the experiment and the table. Furthermore we like you to rerun a temperature experiment in the entire class.

1. Get a PI3 model B, an HDMI cable, a power supply, a case. Such a configuration is listed in the IoT section.

2. Buy or manufacture a case of your choice. You can use a 3d-printer if you have one available.

3. Conduct a temperature experiment.

Discussion of these assignment is to be executed openly in class. Points will be issued only once the class agrees upon an experiment.

This exercise is not only to learn about the behavior of the Pi, but also about how to coordinate experiments with a large number of students.

Pi.Single.1

: What temperature measurement is missing from the table.

Pi.Single.2

: How would you create an experiment under *load*.

Pi.Single.3

How would you create an experiment to which all students in different classes could contribute their values? Can the cloud be used?

Pi.Single.4

Collect the information from all class members using cloud services.

Pi.Single.5

Identify how to use the VPN server so you can use your Laptop instead of a TV or computer monitor. Write a section

4.0.1.2 Small Pi Cluster

In this set of exercises we will be building a small Raspberry Pi cluster. All of you will have to do exercise Pi.Cluster.Build as well as one of the tasks related to Swarm, kubernetes or Spark.

It is important that you write down all steps very carefully as you are expected to use the steps to develop an automated deployment. For your cluster. Your section will be tested by other groups and easy of installation completeness, and correctness will be evaluated. Teams that find issues and improve deployment sections will receive points. TA's will also replicate these steps to identify a fair evaluation without bias.

Pi.Cluster.Build

Build groups of up to 5 people. Make a plan on what needs to be done to build the cluster and develop a schedule. Include in this plan (a) obtaining the material the hardware build, (b) the installation of the operating system (c) the testing of the system (d) familiarizing with the OS.

Pi.Cluster.DockerSwarm

Install a docker Swarm cluster on your PI. Develop a section in markdown and mind plagiarism. Contribute your section to this

document to get acknowledged and credit. Work with others in class to coordinate a single section.

Pi.Cluster.Kubernetes

Install a kubernetes cluster on your PI. Develop a section in markdown and mind plagiarism. Contribute your section to this document to get acknowledged and credit. Work with others in class to coordinate a single section.

Pi.Cluster.Spark

Install a spark cluster on your PI. Develop a section in markdown and mind plagiarism. Contribute your section to this document to get acknowledged and credit. Work with others in class to coordinate a single section.

4.0.1.3 Cluster Case

In this exercise you will be designing a cluster case so you can put the Pi's in a professional looking case that also protects the equipment.

It will be up to you to decide upon the form factor of the case. Please consult with the appropriate sections as you will need to fit in not only the Pi's but also the parts.

Please use OpenSCAD as basis. You may also try FreeCAD.

PI.case.ClusterHat

Design a case for the Cluster Hat

PI.case.5

Design a case

PI.case.n

Please reuse the Pi.Case.5 as basis, e.g. it is a shelf in which we place the cases. Make sure to organize power and allow for easy access to the cases and the pis within the case.

4.0.1.4 Cluster Case Lego

Pi.lego 1

count the connectors and make a more accurate calculation

Pi.lego 2

design a frame in which the Pi's can be placed in groups of 5.

Pi.lego.3

design a cluster case for the raspberry PI with LDD https://www.lego.com/en-us/ldd. You do not have to buy the pieces, but just provide the design.

5 SETUP

5.1 RASPBERRY PI SETUP (SMALL NUMBER OF PIS) FA18-516-03

This section will be the start for the replacement for all previous setup instructions. I think we want ultimately the section "PI Network of Workstations" to also use this or be the final section.

Once the content has either been integrated here or it is determined that the previous file is no longer needed, we will move the other file into a deprecated directory, and remove the file from chapter.yaml.

We will also need to manage a second documentation just for CM-burn in the cm-burn repo that just focuses on cm-burn as this is also a stand-alone program.

The duplicated sections we are aware of include:

If its integrated mark the check mark. We need to be careful not to lose info

- [ ] https://github.com/cloudmesh/cm-burn/blob/master/README.md
- [ ] https://github.com/cloudmesh-community/book/blob/master/chapters/pi/setup.md
- [x] https://github.com/cloudmesh-community/book/blob/master/chapters/pi/setup-dev.md Now only contains information for a development environment. It needs to be renamed. Stays in this file for now.
- [ ] https://github.com/cloudmesh-community/book/blob/master/chapters/pi/pi-passwordreset.md
- [ ] https://github.com/cloudmesh-community/book/blob/master/chapters/pi/run-at-boot.md
- [x] deleted https://github.com/cloudmesh-community/book/blob/master/chapters/pi/sd-card.md integrated in setup-ultimate
- [ ] https://github.com/cloudmesh-community/book/blob/master/chapters/pi/clusters/pi-configure-cluster.md

- [ ] https://github.com/cloudmesh-community/book/blob/master/chapters/pi/clusters/pi-setup.md

There may even be more such documentation as part of student projects. No student that does a PI project MUST DESCRIBE HOW THEY SET UP THE CLUSTER IN THEIR REPORT. THEY ALL MUST IMPROVE OR USE THIS SECTION.

I also see that portions of other files include or can leverage what we do in cm-burn and thus we can replace that info or merge portions of it such as in and than these sections need to be fixed while using our ultimate guide, e.g. make a pointer to it

- [ ] https://github.com/cloudmesh-community/book/blob/master/chapters/pi/kubernetes/pi-kubernetes.md
- [ ] https://github.com/cloudmesh-community/book/blob/master/chapters/pi/kubernetes/526/readme-kube.md
- [ ] https://github.com/cloudmesh-community/book/blob/master/chapters/pi/kubernetes/417/pi-kubernetes.md

As you see everyone duplicated in part the steps. So what we need is a single section that describes the cm-burn procedure, but also the steps needed by hand for those that can not afford the $50 investment of the mount prg.

Next we propose an outline. Help improving the outline than contribute here to this single document while not replicating sections but refer to sections if needed. IF difference between windows osx and linux, aslo include the differences.

5.1.1 Image Choice

When it comes to the operating system install, we have multiple options. One of the options you will find is the installation of what is called NOOBS. NOOBS is actually not an operating system, but it installs an operating system. NOOBS has the feature to install an operating system of choice on the Raspberry. It also supports recovering from a faulty OS. A good introduction that showcases some of the features of NOOBS is available at:

- https://www.raspberrypi.org/blog/introducing-noobs/

It also provides the necessary tools to modify the `config.txt` file that is used at boot time.

However, as we at this time only intend to use Raspbian as the OS, there is no need to install NOOBS. If the OS breaks, we simply burn a new SD card. Hence the features we gain from NOOBS are not as beneficial to us.

Instead we will directly install Raspbian on our SD card and configure it appropriately.

- https://www.raspberrypi.org/downloads/

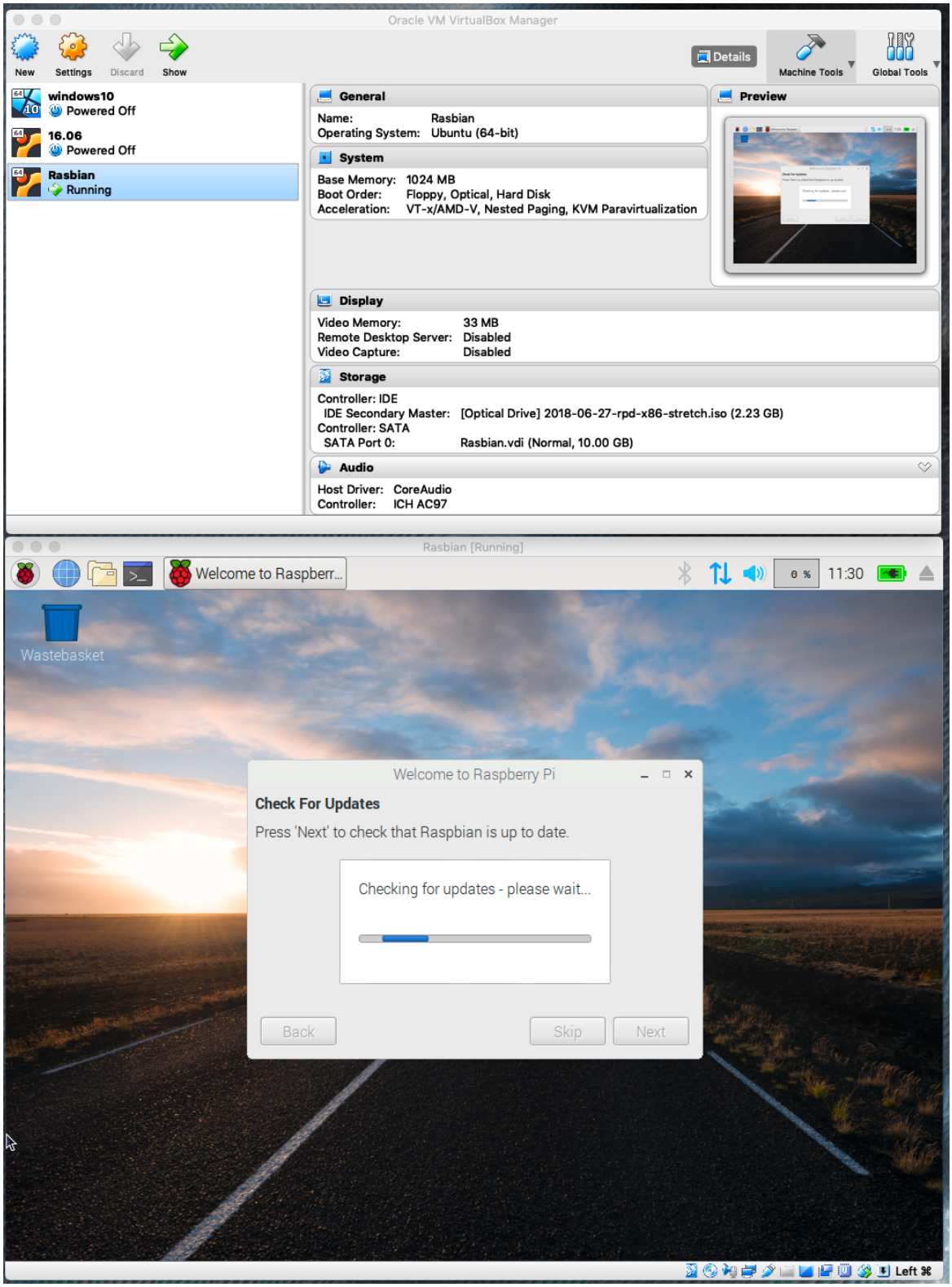
5.1.2 Simulating a Raspberry PI on a Computer

In case you do not have a computer available, you can also install a Raspberry Pi in a virtual machine.

- Raspberry Pi Desktop Downloads
- Raspberry Pi Desktop image direct link

You can download the image and start it via VirtualBox. As we work with newer PIs we recommend that you set up under Linux virtual machine with 64 bit with Other.

Once completed, you will see that it looks something like



Virtual Raspberry Pi

5.1.3 Setting up a Single Raspberry PI

We discuss here the steps to set up a single Raspberry Pi while installing Raspbian on an SD Card. For this we will use the Etcher SD card burning tool for Windows and macOS. Other solutions such as using command line scripts are also available and are demonstrated in the section about burning SD Cards in Linux.

5.1.3.1 Burn an SD Card with cm-burn fa18-516-03

A very convenient program to create an SD card for a Raspberry Pi is using the program `cm-burn`. The program is available from

- https://github.com/cloudmesh/cm-burn

It can be installed with

```
mkdir -p cloudmesh-community
cd cloudmesh-community
git clone https://github.com/cloudmesh/cm-burn.git
cd cm-burn
pip install .
```

You will now have the program `cm-burn` available. Please note that `cm-burn` is provided without any warranties to work and that if you damage your system we do not have any liability.

The command sets up the Raspberry Pi directly on the SD Card without the need for rebooting it. The downside is that you need to have an OS that can mount the ext file system. The ext file system is native to Linux but it is not currently natively supported on Windows 10 or macOS. A third party file system driver is available from Paragon Software that supports ext. The Windows version called Linux File Systems for Windows costs about $20 USD and the macOS version called extFS for Mac costs about $40 USD. Linux supports ext natively and is fully supported by cm-burn. Detailed information on how to use cm-burn is provided at

- https://github.com/cloudmesh/cm-burn/blob/master/README.md

To use `cm-burn` you must first download an image of the Raspberry Pi OS Raspbian Stretch from https://www.raspberrypi.org/downloads/raspbian/. We have tested the software and these steps with the latest version released on 2018-11-13 and

several previous versions. You can download the latest Raspbian Stretch Lite version or directly download Raspbian Stretch Lite 2018-11-13.

After downloading the zip file please unzip it and create a directory in your user folder called `.cloudmesh/images` and place the `.img` file in this directory. This is the default location that `cm-burn` uses to search for image files to burn to the SD card.

You can now burn your first SD card using the following command on Linux or macOS (TODO: add Windows command):

```
$ cm-burn create --name red01 \
  --ips 192.168.1.101 --domain 192.168.1.1 \
  --image 2018-11-13-raspbian-stretch-lite.img
```

In order to provide an option to setup an SD card without purchasing any software we have included the manual setup steps in this document. Please read and follow the steps in the following sections:

- Install Raspbian on a SD card
- Password
- Set the hostname
- Wireless Network at Home
- Wireless Network at IU
- Update the system

5.1.3.2 Install Raspbian on a SD card

For many Raspberry Pi related projects we need to install an Operating system on an SD card. We use **Raspbian** as the OS as it is widely supported. Other OS have recently been added to the available list of operating systems for the PI, but we will at this time not consider them here.

To install the OS on an SD Card you will need another computer. We describe next the process if you have either a MAC or an Linux Ubuntu machine. If you have other OSes and like to contribute, please add your suggestions.

The processes described in this section only work for a few SD cards and is not suitable for burning hundreds of SD cards as we would need for a cluster consisting out of many PI's.

5.1.3.2.1 Download Raspbian

No matter which OS you create the SD Cards on, you will need to download the Raspbian OS.

Next, you need to download the Raspbian image and place it in a directory. As you may reuse the image multiple times, we recommend to place it in a location you remember. Let us assume you place it in `~/Download` directory. Visit the link

- https://downloads.raspberrypi.org/raspbian_latest

and download the image into the folder of your choice (we assume `~/Download`).

5.1.3.2.2 Etcher from Windows and macOS, Linux

An easy way to burn a SD Cards on Windows, macOS, and Linux is with a program called Etcher. Etcher can be downloaded from

- https://etcher.io/

Chose the download suitable for your OS. On Windows you have a couple of options. We recommend that you use the 64 bit Installer version if your OS supports it. If you have a Windows 32bit OS, it may be time to upgrade your computer and/or OS. Also on Linux you need to make sure you distinguish between 32bit and 64bit.

5.1.3.2.3 Windows 10

Once you download it, start Etcher and select the unzipped Raspbian image file. Now select the drive of the SD card. Click Burn and your image will be written to the SD card. You can monitor the progress and once it is completed the SD card will automatically unmount. Use it now in your Raspberry Pi.

The process is the same as the one on macOS

5.1.3.2.4 macOS

Once the image is downloaded you copy it with Etcher onto the SD-card.

1. Place an SD Card into a SD card reader. We recommend a 32GB card.
2. Attach the card reader to the computer

3. Open Etcher and select the downloaded `.img` or `.zip` file which you will likely find in the `~/Download` folder if you followed our previous steps
4. Select the SD card to write the image to. Be careful, to chose the right location as otherwise you could create unexpected data loss
5. Hence, review selections carefully and click *Flash!* to begin writing data to the SD card.

5.1.3.2.5 Ubuntu from Etcher

The process is the same as on macOS.

5.1.3.2.6 Ubuntu from Commandline

In Ubuntu we can use the advanced Linux commands to burn the SD Cards. In the file explorer, right click on the SD card and format the SD card. This is done as follows:

1. Run

   ```
   $ df -h
   ```

   to list all the drives in the computer
2. Insert the SD card and run the command again
3. Now a new entry will be listed which is the SD card
4. The left column of the results from `df -h` command gives the device name of your SD card. It will be listed as something like `/dev/mmcblk0p1` or `/dev/sdX1`, where X is a lower case letter indicating the device. The last part (p1 or 1 respectively) is the partition number.
5. Write down the name of the SD card (without the partition)

6. Unmount the card so that the card can not be read from or written to with the following command:

   ```
   $ unmount dev/mmcblk0p1
   ```

   Make sure to use correct name for the card. If your card has multiple partitions unmount all partitions

7. Next write the image to the SD card by running the command:

   ```
   $ dd bs=4M if=<path to .img> of=/dev/mmcblk0 status=progress conv=fsunc
   ```

Make sure `if=` contains the path to image and `of=` contains the name of the SD card otherwise you may ruin your hard disk

To check, if the image was properly written you can do the following:

1. Create an image again from the SD card by running the following command:

```
$ dd bs=4M if=/dev/sdX of=from-sd-card.img
```

2. Truncate the image to be the same size as that of the Raspbian image

```
$ truncate --reference <original raspbian image> from-sd-card.img
```

3. Run `diff` to see if the two files are same by running the following command:

```
$ diff -s from-sd-card.img <original raspbian image>
```

   If everything is OK, `diff` should say that the two files are the same.

In most cases the verification step will not be needed.

5.1.3.3 Password

Before you bring your Raspberry Pi on the network, you need to reset the password. This can be done by starting the terminal and typing in it the command

```
pi$ passwd
```

The original password is `raspberrypi` and everyone knows it. So if you put your pi on the network it is easily compromised. Hence, change your password first.

5.1.3.4 Locale

You want to also set your system to use your language settings for the keyboard. You can do this from the terminal with

```
pi$ sudo raspi-config
```

or

```
pi$ sudo dpkg-reconfigure locales
```

or using the GUI.

5.1.3.5 Set the Hostname

The hostname is stored in `/etc/hostname`. Edit the file and change it to a name such as green00, green01, green02, green03, green04, green05. Be consistent with the names. The 00 host should be the topmost host in the cluster.

edit

```
pi$ sudo nano /etc/hostname
```

after you edited the hostname

```
pi$ sudo /etc/init.d/hostname.sh start
```

The Pi can also give an error if the hostname set in `/etc/hostname` does not also have an entry in `/etc/hosts` as the local loopback. To fix this, edit `/etc/hosts` and on the last line you should see:

```
127.0.1.1       raspberrypi
```

This should be changed to the new host name set in `/etc/hostname`.

```
    pi$ sudo nano /etc/hosts
```

5.1.3.6 Wireless Network at Home

The easiest way to get internet access and to continue the setup is using a wireless network. You can configure it either via the GUI or command line. The `raspi-config` utility can also setup a WiFi connection. The steps shown next were taken from the Offical Raspberry Pi Wireless setup.

The Raspberry Pi is already configured to connect to a WiFi network, all you need to do is set your network name (ssid) and passphrase. Additionally, you should set the current country since some locations have different restrictions on the available WiFi radio bands. The file `/etc/wpa_supplicant/wpa_supplicant.conf` contains details about network names and passwords and you can add your details directly to the bottom of this file for any wireless networks you would like to connect to. You can directly add the plain text of your wireless passphrase but it is much better to add the hash of the passphrase since this will not expose your passphrase. (Please note, however, that the hash can be used by any computer to connect to the network and a brute force search could recover your password, but it still better than plain text).

To find the WiFi networks that your Pi can currently detect run this command:

```
$ sudo iwlist wlan0 scan
```

The proper format for a wireless network definition is:

```
...
country=US
network={
        ssid="network_name"
        psk="WiFi passphrase or hash
}
...
```

The Pi comes with a utility that can automatically generate your WiFi passphrase hash called `wpa_passphrase`. You can execute this command to hash your passphrase and append it to the correct file:

```
$ wpa_passphrase "network_name" "passphrase" | sudo tee -a /etc/wpa_supplicant/wpa_supplicant.conf
```

Or you can run this command and type your pass phrase at the prompt followed by Enter:

```
$ wpa_passphrase "network_name" | sudo tee -a /etc/wpa_supplicant/wpa_supplicant.conf
```

Unfortunately, `wpa_passphrase` includes the original passphrase in plain text so you will need to edit the file by hand to remove it. Use your favorite editor and remove the commented out line with the plain text passphrase. At this time you should also add the country designation as this may be necessary in some cases.

```
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

The original file should be changed from this:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
        ssid="network_name"
        #psk="my plaintext passphrase"
        psk=0617cac0927403beefda5705f5ff97bbc562f5d1907b40f02c39912a7d595b0f
}
```

to this:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US
network={
        ssid="network_name"
        psk=0617cac0927403beefda5705f5ff97bbc562f5d1907b40f02c39912a7d595b0f
}
```

You can then reconfigure the wireless adapter and it should get an IP address. You can complete this and see the current setup with the following commands.

```
$ wpa_cli -i wlan0 reconfigure
$ ifconfig wlan0
$ iwgetid
```

Note that if you have configured your `wlan0` interface with a static IP address then it will only use this static address. Also, if you configure the same static IP address for both `eth0` and `wlan0` (this is the default if you use `cm-burn`) then only one of the interfaces will be assigned an IP address. This is not a problem and if you disconnect either of the interfaces then the other one will immediately be assigned the IP address.

If you need to renew your DHCP lease you can use the following command. If you want to renew the lease for the Ethernet adapter then replace `wlan0` with `eth0`. Note that you will lose your connection to the Pi during this process if you are connected using the same interface, but the Pi should come back online after 20 seconds:

```
$ sudo dhclient -r wlan0; sleep 10; sudo dhclient wlan0
```

5.1.3.7 Wireless Network at IU ⭕

⭕ {#s-wireless-at-iu}

⭕ TODO: Update with new IU public wireless information

IU runs several different networks. This includes IUSecure, Eduroam, and ATT Wifi. The first two would require you to use your IU username and password to be entered in the configuration. Although technically possible we find the method

- ⚠ **HIGHLY** insecure and
- ⚠ **STRONGLY** advice against doing so.

Let us assume you put your information on a PI and than someone takes the SD Card from it. They can than look into the card and steal your password. Obviously this is not advisable. In other cases you may have configured your software wrong and someone could login remotely and lift your password remotely. Obviously this is not advisable.

Regardless, we have seen from instructors the advice to use IUSecure.

> *This is* ⚠ ***WRONG!***

Do not listen to them about this particular issue and advise them to use an alternative setup.

One such alternative (which is also not ideal) is to use the free wifi offered by ATT Wifi. It is a bit complex to setup as you need to go to the Web browser to the address http:\\iu.edu and click on the connect button. Sometimes that button is not visible so you need to scroll down to see it.

We also have an internal network that we will not discuss here, but can be used upon consultation with Dr. von Laszewski.

5.1.3.8 Update

We want to update the software and make sure everything is up to date. This is done with

```
pi$ sudo apt-get update
```

To develop easily we need a number of programs on our Pi. Additional programs can be installed with the command

```
pi$ apt-get install PACKAGE
```

where `PACKAGE` is the name of the software we like to install. A good example is emacs which can be installed with

```
pi$ apt-get install emacs
```

5.1.3.9 Remote access via ssh

In the latest Raspbian OS ssh is enabled by default. However, if you discover that it is not enabled, the following commands should enable it.

```
pi$ sudo systemctl enable ssh
pi$ sudo systemctl start ssh
```

Naturally you need to do a bit more such as placing your public key in the `authorized_keys` file explained later, but for now we will just activate ssh.

To access the machine using public keys we recommend that you add your public key to the `~pi/.ssh/authorized_keys` file

5.1.4 Setting up a Small Cluster by Hand

If you would like to setup a Raspberry Pi cluster please refer to the section Network of Pis for details on configuring a cluster by hand or with our convenient tools and scripts such as `cm-burn`.

5.1.5 System Preparation without Monitor

⭕ there is lots of duplication here to the ultimate setup

5.1.5.1 hostname

Find a way on how to name the host as each of the cluster nodes will need its own unique name. We simple use color01, where color is the color of the USB cables you have and the number is the ith PI starting from the top

5.1.5.2 SSH

Describe how you enable SSH without a monitor

5.1.5.3 key

Describe how you can generate a private key at the right location on the SD Card. Place your own public key on the SD Card

Write python programs for this.

5.1.5.4 password

Describe how you can change the password on the SD Card

5.1.6 Post configuration

5.1.6.1 Network Addresses

Some online guides recommend sending a ping to the broadcast address of your network but Raspbian ignores broadcast pings by default so the Raspberry Pis will not respond to this and then will not show up in arp tables. `nmap` will work, however.

This works on a Pi substitute your network submask for `192.168.1.0/24`:

```
$ sudo apt-get install -y nmap
$ nmap -sn 192.168.1.0/24
# will list all devices on the network
$ arp -a
# will list devices in arp cache and lookup hostname
$ arp -a -n
# Same as previous but skips hostname lookup
```

`nmap` is also available on Windows and macOS. It can be downloaded directly from Nmap installation instructions or using Homebrew on macOS as `brew install nmap`. Usage is as listed previously.

5.1.6.2 key

Write a python program that does the following:

1. login with ssh on each PI and call ssh-keygen to generate a unique key on each PI.
2. copy this .pub keys to your computer and store them into a file called `authorized_keys`
3. copy that file to all Pis
4. you may also have to copy your authorized key file to

5.1.6.3 VNC

find out how to set up vnc so you can login into the PI and see its GUI

- https://www.raspberrypi.org/forums/viewtopic.php?t=74176
- https://www.raspberrypi.org/documentation/remote-access/vnc/

```
host$ alias IP=<IPADDRESSOFPI>
host$ sh pi@$IP

pi@IP's password:
Linux raspberrypi 3.10.25+ #622 PREEMPT Fri Feb 3 20:00:00 GMT 2018 armv6l
```

```
pi@raspberrypi ~ $ sudo apt-get tightvncserver   # download the VNC server
pi@raspberrypi ~ $ tightvncserver                # start the VNC server
pi@raspberrypi ~ $ vncserver :0 -geometry 1920x1080 -depth 24  #start a VNC session
```

Now, back on your Mac: In the Finder, select Go => Connect To Server… Type `vnc://xxx.xxx.xxx.xxx` (where `xxx.xxx.xxx.xxx` is the IP address that you discovered in step 2. Click [Connect]. This will launch the Screen Sharing application, and with a little luck, you should see this image.

5.1.7 Setting up a Small Cluster with cm-burn

This discusses how to set things up for many PIs with `cm-burn`

5.1.8 Setting up a large cluster with cm-burn

5.1.9 DHCP setup

5.1.9.1 Configure Head Node (port forwarding and DNS)

Install Dependencies:

```
$ apt-get update
$ apt-get install -qy dnsmasq clusterssh iptables-persistent
```

5.1.9.1.1 Create Static IP

TODO: Verify: This should already be done by `cm-burn`

Copy old config (-n flag prevents overwrite):

```
$ \cp -n /etc/dhcpcd.conf /etc/dhcpcd.conf.old
```

To update DHCP configuration, add the following to **/etc/dhcpd.conf**:

```
interface wlan0
metric 200

interface eth0
metric 300
static ip_address=192.168.50.1/24
static routers=192.168.50.1
static domain_name_servers=192.168.50.1
```

5.1.9.1.2 Configure DHCP Server:

Copy old config (-n flag prevents overwrite):

```
$ \cp -n /etc/dnsmasq.conf /etc/dnsmasq.conf.old
```

To update DNS configuration, add the following to **/etc/dhcpd.conf**

```
interface=eth0
interface=wlan0

dhcp-range=eth0, 192.168.50.1, 192.168.50.250, 24h
```

5.1.9.1.3 NAT Forwarding

To Setup NAT Forwarding, uncomment the following line in **/etc/sysctl.conf**:

```
net.ipv4.ip_forward=1
```

5.1.9.1.4 IP Tables

Create IP Tables:

```
$ sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
$ sudo iptables -A FORWARD -i $INTERNAL -o wlan0 -j ACCEPT
$ sudo iptables -A FORWARD -i $EXTERNAL -o eth0 -j ACCEPT
```

Make rules permanent:

```
$ iptables-save > /etc/iptables/rules.v4
```

5.1.9.2 SSH Configuration

Generate SSH keys:

```
$ ssh-keygen -t rsa
```

Copy key to each compute node:

```
$ ssh-copy-id <hostname>
```

For hostnames rp1-4 (final node names will be: rp0, rp1, rp2, rp3, rp4).

5.1.9.3 Configure Cluster SSH

To update Cluster SSH configuration, add the following to **/etc/clusters**:

```
$ rpcluster rp1 rp2 rp3 rp4
```

Now you can run commands to all clusters by:

```
$ cssh rpcluster
```

NOTE: This seems to be related to using `cssh` Cluster SSH to update all the nodes together. I would suggest this is better down by using Docker or Ansible.

5.1.9.4 PXE Boot

5.1.10 Using Advanced setups with Ansible

5.1.11 Change Password on the SD-Card

It is possible to reset the password for a PI SD Card. This comesin handy when you did forget it or the team that worked on a Pi has left the project but valuable information may still be on the PI. To do so, You need tou unplug the raspberry pi and remove the SD card from the slot. Next you need to have the ability to mount the file systems. On macOS and Windows you can use extFS. Naturally if you have a linux machine or another PI, you can use an SD Card reader/writer and mount it directly. You will need root access on the machine where you execute the password reset.

After you inserted the card, please Locate and edit the `etc/shadow` file on the SD card. To create a new password use the command

```
$ openssl passwd -1 -salt <unique string>
```

Next, we e must find the line that starts with pi and replace the text between the first and second with the output from the previous command we had executed in the `etc/shadow` file

Now you can Eject the SD card from the computer, and insert it into the Pi. Boot the raspberry pi and log in to it while using the new password. This naturally only works if you allow password login. ON many systems we however disable it and use public key authentication only. In this case you need to just replace the public key in the `authorized_keys` file. Using just keys is obviously more convenient.

Naturally mounting the SD Card and looking in the filesystem would also allow you to look at the network setup. That is certainly not good and before a PI is returned sensitive information should be cleaned from the SD Card.

5.1.12 Creating Backup

In this section we will explain how to create a backup of the image from the SD Card in a PI.

5.1.13 Duplication

Let us assume you have installed a lot of great programs on the SD Card. In a cluster, we need to duplicate this card for each PI in the cluster. Is there a way for us to duplicate the software

5.1.14 Exercises

SD-Card.1

> Improve the Ubuntu SD-card documentation

SD-Card.2

> Could a script be written that does the entire process via a python or shell command in Ubuntu.

SD-Card.2

> Could a script be written that does the entire process via a python or shell command in macOS?

SD-Card.3

> could a script be written that does the entire process via a python or shell command in gitbash for Windows?

SD-Card.4

> In general the Ubuntu documentation is complex, how can it be simplified? Maybe through automation?

5.2 Setup of a Development Environment

5.2.1 Editors

Naturally we need a useful editor. We have made good experience with emacs as it supports a variety of different formats and is also available for macOS and Windows.

You can install it with

```
$ apt-get install <program>
```

Other editors include `emacs`, `vim`, `gedit` and so on. If you are concerned about space, use `vi` which is pre-installed. If you like to use other editors use the command we can install them respectively with

```
$ apt-get install emacs
$ apt-get install vim
$ apt-get install gedit
...
```

5.2.2 Python on the Pi  fa18-516-03

Raspbian Stretch Lite version 2018-11-13 comes by default with Python 2.7.16 and Python 3.5.3. However, it does not come with `pip` which is the default Python configuration management tool.

Add instructions to install pip and pip3 which are not installed by default in the Lite images:

```
$ sudo apt-get install -y python-pip python3-pip
$ pip -V
pip 9.0.1 from /usr/lib/python2.7/dist-packages (python 2.7)
$ pip3 -V
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.5)
```

Note that pip for Python 3 can also be run by using the following command

```
$ python3 -m pip install ...
```

The installation packages for pip on Raspbian will also properly setup the https://www.piwheels.org package repository which contains pre-compiled binary wheels for many popular python packages. A *wheel* is a python package that is already compiled into a binary form for a particular OS and hardware chipset. Installing a wheel is much faster and less error prone than building a package from source. These wheels have also been optimized for the ARM chipset that the

Raspberry Pi uses so they will run at the highest speeds possible. For example, you can install the wheel for numpy and scipy with:

```
$ pip3 install numpy scipy
```

If you want to upgrade to the latest python version you can build it from source as follows

```
$ sudo apt-get install build-essential checkinstall
$ sudo apt-get install libreadline-gplv2-dev libncursesw5-dev
$ sudo apt-get install libssl-dev libsqlite3-dev tk-dev libgdbm-dev
$ sudo apt-get install libc6-dev libbz2-dev
$ cd /usr/src
$ wget https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tgz
$ sudo tar xzf Python-3.6.5.tgz
$ cd Python-3.6.5
$ sudo -s
$ bash configure
$ make altinstall
```

5.2.3 Python IDLE

Click Menu -> Programming -> Python 3 (IDLE), and you'll get a new window called 'Python 3.6.5 Shell:'. This Shell works just like Python on the command line. Enter print("Hello World") to see the message.

5.2.4 Go

To install go use

```
$ wget https://storage.googleapis.com/golang/go1.9.linux-armv6l.tar.gz
$ sudo tar -C /usr/local -xzf go1.9.linux-armv6l.tar.gz
$ export PATH=$PATH:/usr/local/go/bin
```

If you like to have it included every time you start a terminal please please the line

```
export PATH=$PATH:/usr/local/go/bin
```

in your ~/.profile file and reboot.

Now you are also able to program go on your Pi.

5.3 Visual Feedback Management

Now that we have a Pi configured we need to make sure that we can send sometimes visual clues about its operations. This can be done via displays and

LEDs. In case no display is available we can even connect to the PI with a virtual display redirection.

5.3.1 Default Display Setup

The raspberry pi comes with a defualt setup and any modern HDMI enabled TV or monitor will most likely work out of the box without any modifications. Just plug the HDMI cable in and start the Pi and you will probably see some boot messages and after a while see the splash screen for your installed operating system.

5.3.2 Automatic Display Detection

In case you like to use the Raspberry Pi in your office, at home, or in the field, you may find yourself in a situation where different monitors with different resolutions are attached. Especially in the field it is convenient if the Pi could do this adjustment for you and use the resolution assigned for the attached device.

In this section we describe such a solution. We will automatically detect the resolution based on the monitor attached. Once detected the config file will be rewritten if necessary and the Pi will be rebooted with the correct resolution in the configuration file.

> ⚠ *In order for your monitor to work, you will need to add it to the script we provide. Use the following program and add appropriate cases for your monitor.*

```
/usr/bin/tvservice -d /boot/edid.dat
/usr/bin/edidparser /boot/edid.dat
```

5.3.2.1 How it works

To achieve this we have developed the file displaydetect.py that automatically detects and set the display for the pi. To integrate it in the Raspbian OS please follow these steps

1. Install the display detect script. Run this as root:

```
$ wget https://raw.githubusercontent.com/cloudmesh/cloudmesh-pi/master/displaydetect.py -O /bin/displaydetect.
$ chmod a+x /bin/displaydetect.py
```

2. Create a copy of `/boot/config.txt` and rename it to `/boot/config.txt.in`

```
$ cp /boot/config.txt /boot/config.txt.in
```

3. Add the following lines to the end of the config.txt.in file:

```
# customized display setting
hdmi_group=2
hdmi_mode=87
hdmi_cvt {x} {y} 60 6 0 0 0
display_rotate={display_rotate}
```

4. Add the display detect script to rc.local so it runs every time the pi is booting. Add this to end of /etc/rc.local before the “exit 0” line:

```
$ /usr/bin/python /bin/displaydetect.py
```

5. Make sure rc.local would be running during boot.

```
$ systemctl status rc-local.service
```

6. Check if the service is ‘active’.

   Once it is installed, the pi will be check the display during the first boot, and it will set the correct display parameters if found necessary and do a second boot to finish the configuration change.

Now you can go in the filed and use the monitor of your choice

5.3.2.2 Adding new monitors

1. Look at the monitor name
2. look at the preferred resolution
3. create an new condition in the switch

5.3.3 Using the Laptop as a monitor

In case you do not have a monitor at hand there are various ways on using your computer as a monitor fo the Pi. The initial setup is recommended to do with an HDMI monitor. There are instructions on the Web that will let you set up the PI even without such a monitor, but here we go the simpler route to set it up once, so you can access it without monitor once you completed the setup.

First, install the remote desktop program on the Pi with

```
pi$ sudo apt-get install xrdp
```

Second, connect now your Laptop with an an ethernet cable to the Laptop. As services start up automatically, you will have to wait for a while. till the LAptop has assigned an IP address. Furthermore you may need to allow the Laptop to assign DHCP addresses when you plug in the ethernet cable.

⭕ we could need your help here telling us how you do this for your machine.

After some time you will see that the Laptop has assigned an ip address

You can run on your Pi, but also on your Laptop (if it supports `ifconfig`)

the command

```
pi$ ifconfig
```

and on your laptop

```
laptop$ ifconfig
```

You will see the typical output from `ifconfig`

look on the pi for the line that starts with

```
inet addr
```

it will be followed by an ip address such as

```
192.168.10.1
```

Now you can connect to this address, with the help of a program such as *Remote Desktop Connection* on Windows or Bonjour on macOS.

Type in the ip address and you will see the desktop.

In order to avoid always having to do the first step and getting different ip addresses, we recommend that you set up a static address for the raspberry.

⭕ please help identifying how to do this

> ⚠ *We recommend not to use the WiFi and the ethernet adapter at the same time as it can come to issues when you boot them up while they are both connected.*

5.3.4 Exercises

E.Display.1:

Configure your Raspberry Pi so you can access it via VNC.

E.Display.2:

Make the displaydetect.py truly discoverable, find a default resolution that you put in to the else statement. Identify the preferred solution from the script and use that. Parse the appropriate parameters such as x,y, ans aspect ratio, rotation and other parameters.

5.4 VNC ☁

Note: *If you like to connect to your Raspberry from your laptop, we recommend to use VNC. If you rather like to connect a monitor and keyboard as well as a mouse to the Raspberry, you can skip the steps with the VNC update.*

5.4.1 Setting up VNC

We had some issues with the installed version of VNC that is customized for connecting a Laptop via the ethernet cable to the PI. However as we connect wirelessly, our setup is slightly different. The easiest way that we found is to update the Raspbian OS as follows. In a terminal type

```
sudo apt-get update
sudo apt-get install realvnc-vnc-server
sudo apt-get install realvnc-vnc-viewer
```

Next you enable the VNC server in the configuration panel via the Raspbian GUI by selecting

```
Menu >
   Preferences >
      Raspberry Pi Configuration >
         Interfaces.
```

Here you toggle the VNC service to enabled. As we are already at it in our setup we enabled all other services, especially those that deal with Grove sensor related bins and wires.

Next reboot and double check if the settings are preserved after the reboot

5.4.1.1 Install VNC on macOS

To install a vnc server of your liking on your Mac. You find one at

- http://www.realvnc.com/download/vnc/latest/

Be sure to download the version of the VNC Viewer for the computer you are going to use to virtually control the Pi (there is a version listed for Raspberry Pi– do not download this one. For us this is the Mac version.)

5.4.1.2 Run VNC Viewer on macOS

Once you have downloaded the VNC viewer installed it you can open the program. Next you can start vnc viewer and enter the ip address of your raspberry. Make sure you are on the same network. You can find the address by using ifconfig.

5.5 Motherboard LED ☁

The Raspberry pi contains an LED that can also be used to provide us with some information as to the status of the PI. It is usually used for reporting the power status.

The green LED can be made blinking as follows in root

```
echo 1 > /sys/class/leds/led0/brightness
echo 0 > /sys/class/leds/led0/brightness
```

Naturally this ac be done via a remote command if your ssh keys are uploaded and your originating computer is added to the authorized_keys. Now you can can control them via ssh

```
ssh pi@red03 "echo 1 > led; sudo cp led /sys/class/leds/led0/brightness"
ssh pi@red03 "echo 0 > led; sudo cp led /sys/class/leds/led0/brightness"
```

This is extremely useful as it allows us to check if we the OS is available and we can access the PI.

One strategy is to for example switch the light of, once it is booted, so we can see which board may be in trouble.

5.6 Run Commands at Boot time ☁️

In many cases we need to provide configurations and programs that run at boot time. A number of different methods exist to run commands and programs at boot time.

We will be focusing here only a few of them

5.6.1 rc.local

On your Pi you will find under `/etc/rc.local` a file in which you can list programs that are started up at boot time. The programs should success fully run and exit with the status 0, and they must not continuously run in which case they need to be started in background.

To make sure you do not forget it, simply add the following line at the end of your program

```
exit 0
```

indicating that the start was successful. Programs in rc.local must use the absolute file path.

5.6.2 Crontab

Crontab is a service the schedules jobs that can run at various times repeatedly. For example we can use `crontab` to run commands every hour, every day, every half hour or other time intervals or at reboot.

To use crontab follow these steps

1. Open a terminal and enter the command

```
$ crontab -e
```

If you are doing this for the first time, you will be asked to chose an editor. Please, choose your favorite editor

2. To run the program at boot time, add the following line to the at the end of the file

```
@reboot <command>
```

Let us look at an example and assume we have test.py program in your home directory at `/home/pi/test.py`. Once you add it to crontab with

```
@reboot python /home/pi/test.py
```

it will be run at boot time

It is important to provide the absolute path to the file. In case your file produces output you need to redirect it into a file

```
@reboot python /home/pi/test.py > /home/pi/test.log
```

- When the raspberry pi reboots, the program will run automatically.

5.6.3 References

A good introduction to the various methods is provided at

- https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/

For more information on crontab see for example:

- http://www.adminschoice.com/crontab-quick-reference

5.7 Pi Software Collection ☁

Please improve the sections.

5.7.1 Web Programming

5.7.1.1 Coder

Targeted at the very beginner to Web programming, we will not use this here.

- https://googlecreativelab.github.io/coder

provide a section

5.7.2 COMPUTING

5.7.2.1 Python on the Raspberry Pi  fa18-516-03

Python packages are typically installed using the `pip` tool. `pip` will automatically detect if you are running a compatible OS and platform and will download a Python wheel for a given package which is a pre-compiled binary package that is compatible with your system. Since the Pi is running an ARM processor (not Intel or AMD compatible) most of the wheels hosted on PyPi (the standard Python package directory server) are not compatible. However, there are Pi-compatible wheels hosted on https://pythonwheels.com/. The pip package in the latest version of Raspbian is updated to look in piwheels as an additional package index. If you have an older version of Raspbian installed you can get the update by running `sudo apt upgrade` to update your system. There is a piwheels FAQ that you may consult if you have any questions or issues.

5.7.2.2 Numpy

Refer to other section in book and describe what is different

provide a section

5.7.2.3 Scipy

Refer to other section in book and describe what is different

provide a section

5.7.2.4 Image Processing

Refer to other section in book and describe what is different

provide a section

5.7.3 System

5.7.3.1 DHCP Server

- http://www.noveldevices.co.uk/rp-dhcp-server

5.7.3.2 hostname

Please see the section Set the hostname to set the hostname on the Pi.

5.7.3.3 Gather the MAC addresses

The MAC address is the hardware address of an Ethernet network device. The MAC address is set by the manufacturer and does not change when you join a different network like an IP address can. You can get the MAC address for the Ethernet interface `eth0` or the wireless interface `wlan0` on the Pi by using the `ifconfig` command and looking for the line that begins with `ether`. The following command will directly output the MAC address for the interface that you specify. You can run `ifconfig` with no parameters to see a list of all the interfaces.

```
pi$ ifconfig eth0 | awk '/ether/ {print $2}'
b8:27:eb:9c:b8:6e
pi$ ifconfig wlan0 | awk '/ether/ {print $2}'
b8:27:eb:ce:ef:3b
```

5.7.3.4 Enable SSH

Not tested

```
sudo mv /boot/boot_enable_ssh.rc /boot/boot.rc
sudo reboot
```

- http://www.noveldevices.co.uk/rp-ssh

> ⚠ *Not sure if this is needed:*

" You may find that you can connect to your Pi with SSH but the session hangs after a successful logon. This is usually caused because of a network QoS

mismatch that affects certain switches and routers but you can correct this by editing the two files

```
/etc/ssh/ssh_config
/etc/ssh/sshd_config
```

and adding

```
IPQoS 0x00
```

to each file as the last record. " THis is quoted and needs the citation

Develop a a python script to do that

5.7.3.5 USB stick

```
cat /var/log/messages
```

find sda*

Make sure to find the right name.

```
sudo fdisk /dev/<device-name>
sudo mkfs -t vfat /dev/<device-name>
mkdir ~/<mount-point>
sudo mount /dev/<device-name> ~/<mount-point>
```

5.7.3.5.1 DHCP server on 00

```
sudo apt-get update
sudo apt-get install isc-dhcp-server
sudo nano /etc/network/interfaces
```

Change it to

```
iface eth0 inet static
address <the-IP-address-of-your-Pi-that-will-be-the-DHCP-server>
netmask <the-subnet-mask-of-your-LAN>
gateway <the-IP-address-of-your-LAN-gateway>
sudo nano /etc/dhcp/dhcpd.conf
```

uncomment the info so the server can start

```
subnet <starting-IP-address-of-your-network> netmask <starting-IP-address-of-your-network> {

     range <first-IP-address-of-your-DHCP-address-range> <last-IP-address-of-your-DHCP-address-range>;

     option routers <the-IP-address-of-your-gateway-or-router>;

     option broadcast-address <the-broadcast-IP-address-for-your-network>;

}
```

edit /etc/default/isc-dhcp-server

```
DHCPD_CONF=/etc/dhcp/dhcpd.conf
DHCPD_PID=/var/run/dhcpd.pid
INTERFACES="eth0"

sudo service isc-dhcp-server restart
```

5.7.3.6 Temperature

```
cat /sys/class/thermal/thermal_zone0/temp
```

5.7.3.7 grafana

Could be helpful to monitor cluster/clusters

- https://github.com/grafana/grafana
- https://github.com/weaveworks/grafanalib

There are many more, just search. We have not tested them example with yaml

- https://github.com/jakubplichta/grafana-dashboard-builder

Light scheme

in /etc/grafana/grafana.ini uncomment line and set

```
default_theme = light
```

6 MULTIPLE

6.1 SETTING UP LARGE PI CLUSTERS

6.1.1 CM-BURN

`cm-burn` is a program to burn many SD cards for the preparation of building clusters with Raspberry Pi's. The program is developed in Python and is portable on Linux, Windows, and OSX. It allows users to create readily bootable SD cards that have the network configured, contain a public ssh key from your machine that you used to configure the cards. The unique feature is that you can burn multiple cards in a row.

A sample command invocation looks like:

```
cm-burn --name  red[5-7] \
        --key ~/.ssh/id_rsa.pub \
        --ips 192.168.1.[5-7] \
        --image 2018-06-27-raspbian-stretch
```

This command creates 3 SD cards where the hostnames `red5`, `red6`, `red 7` with the network addresses `192.168.1.5`, `192.168.1.6`, and `192.168.1.7`. The public key is added to the authorized_keys file of the pi user. The password login is automatically disabled and only the ssh key authentication is enabled.

6.1.2 PROCESS

The process of the burn is as follows.

1. start the programm with the appropriate parameters the program will ask you to place an SD Card in the SD Card writer. Place it in
2. the specified image will be burned on the SD Card
3. next the SD Card will be mounted by the program and the appropriate modifications will bbe conducted.

4. after the modifications the SD Card will be unmounted
5. you will be asked to remove the card
6. if additional cards need to be burned, you will go to step 2.

In case a SD Card of a PI in the cluster goes bad, you can simply burn it again by providing the appropriate parameters, and just print the subset that are broken.

6.1.3 Setting up a Single Large Cluster with cm-burn

`cm-burn` will setup a simple network on all cluster nodes configured. There are different models for networking configuration we could use. However we have decided for one that allows you to interface with your local Laptop to the cluster via Wifi. The setup is illustrated in Figure Networking.

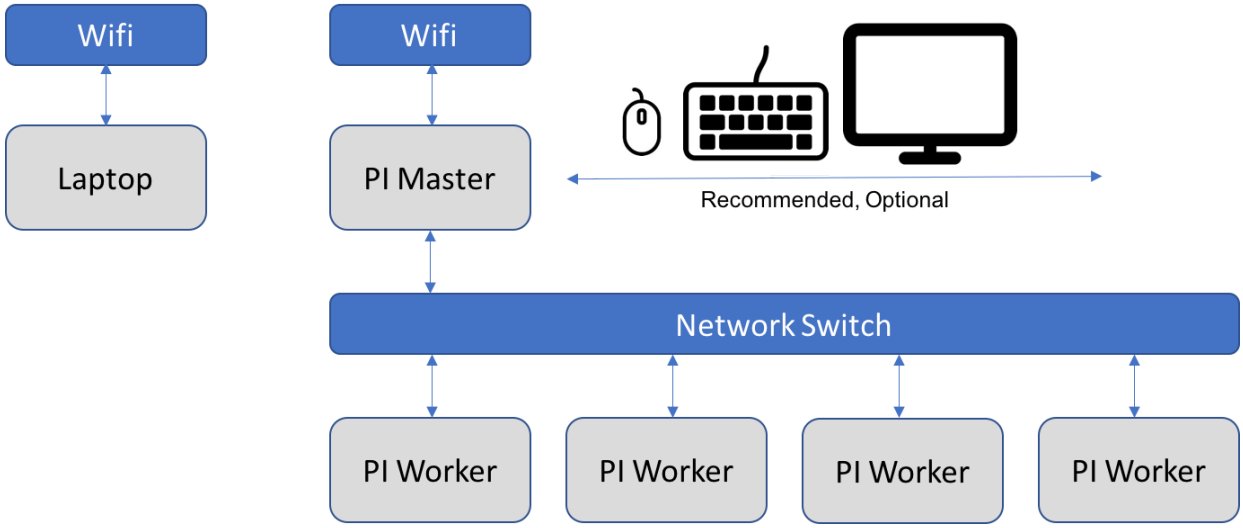


Figure: Networking

We assume that you have used cm-burn to create all SD cards for the Pi's. One of the Pi's is specially configured with the command

```
cm-burn --master red01
```

The SD Card in the SD Card writer will be configured as a `master`. If the name does not match it will be configured as a worker. Only the `master` is connected with the Wifi network. All other nodes rout the internet connection through

the master node. As the `master` node is on the same Wifi network as the laptop you can login to the 'master' node and from there log into the workers. To simplify access you could even setup ssh tunneled connections from the Laptop via the master. But this is left up to you if you wish.

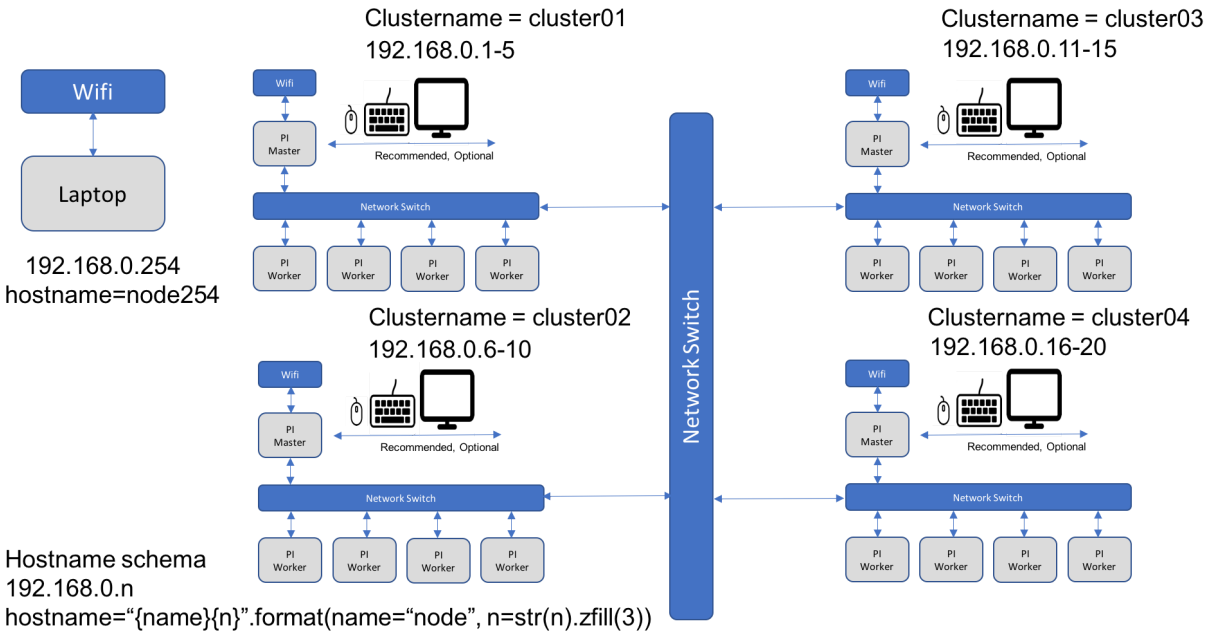
As a result you will be able to login on each of the machines and execute commands such as

```
sudo apt-get update
```

Certainly you can even have a much simpler setup by just attaching a keyboard, mouse and monitor/TV to your `master`. This will allow you to directly work on the master node, not needing any additional hardware.

6.1.4 Setting up a Cluster of Clusters with cm-burn

To integrate the clusters into a single network, we need a switch or combination of switches to which we connect the clusters. This is depicted in the Figure Cluster of Clusters



Each cluster is naemed cluster01-clusterNN. The hostnames are node followed by 3 zeros padded with the node number There is a correlation

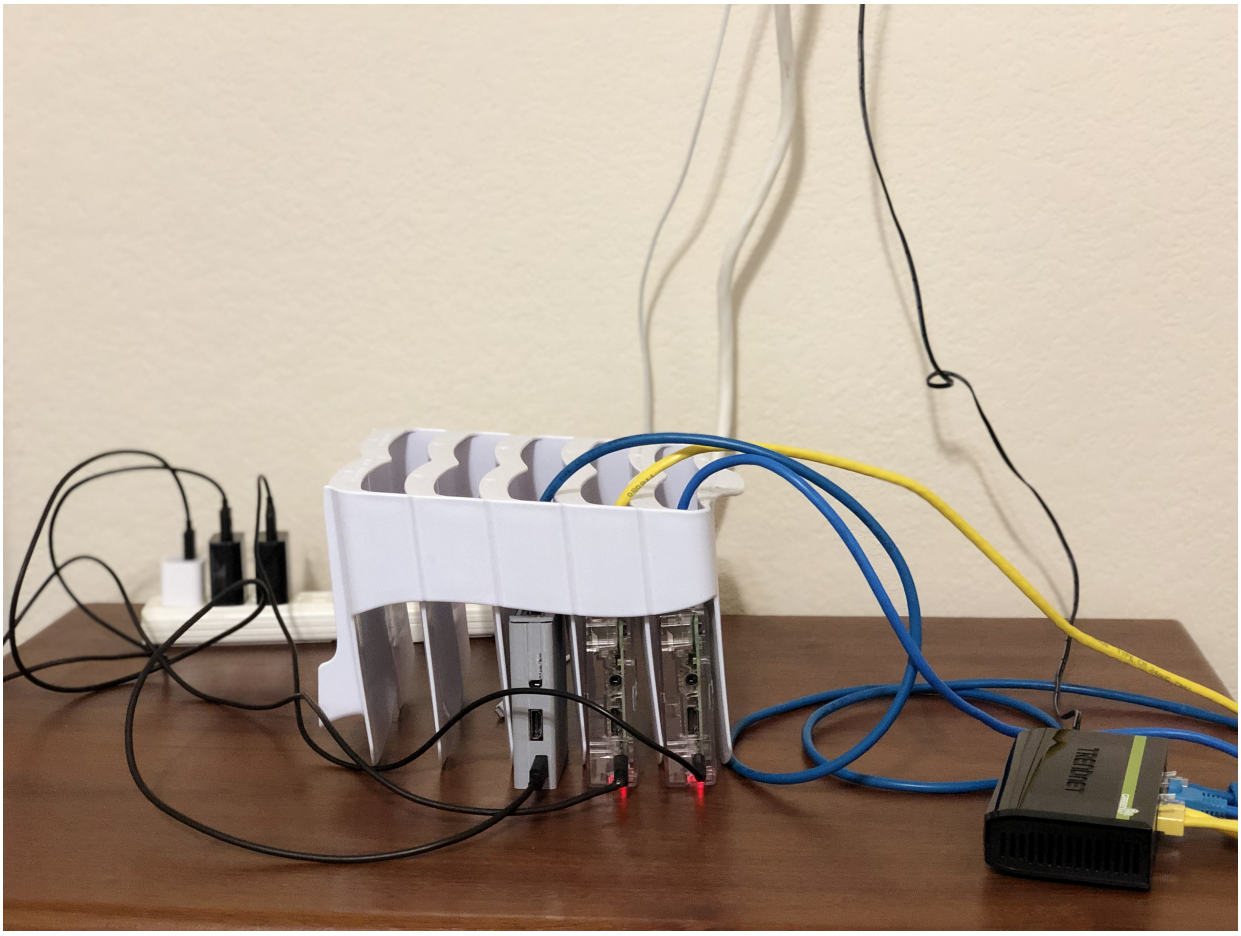
between the cluster number and the node numbers in the following interval

a cluster has the nodes

```
[(clustername - 1) * 5 + 1, (clustername - 1) * 5 + 5]
```

For convenience we will be also enabeling a cluster burn logic, that burns all images for a given cluster

cm-burn –workers=5 –name=cluster –nodes=nodes –id=3



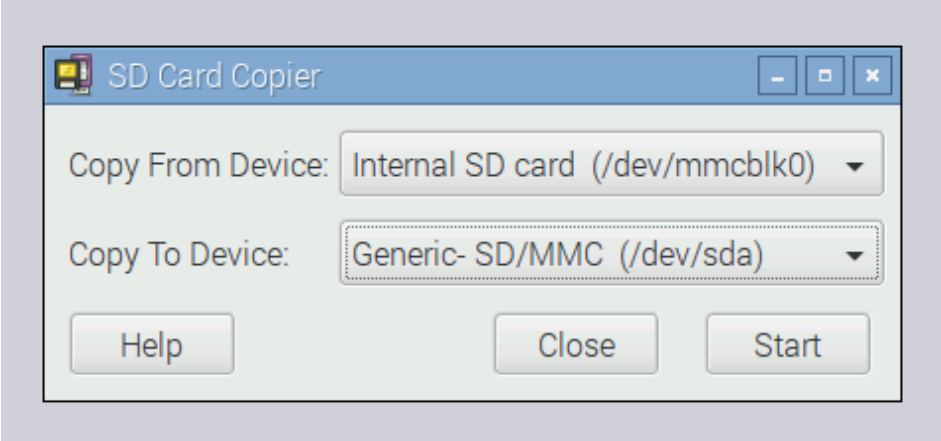
6.1.5 Prerequisits

6.1.5.1 Raspberry Pi

We assume that you have set up a raspberry pi with the newest raspbian OS. We assume that you have changed the default password and can log into the pi.

We assume you have not done anything else to the OS.

The easiest way to duplicate the SD card is simply to clone it with the build in SD Card copier. This program can be found in the menu under Accessories.



SD Card Copier

Figure: SD Card Copier

This program will copy the contents of the card plugged into the PI onto another one. The only thing you need is an USB SD Card writer. You cn accept the defaults when the cards are plugged in which allow you to copy the Internal SD Card onto the other one. Just be carefull that you do not overwrite your internal one. This feature can also be used to create backups of images that you have worked on and want to preserve.

Thus as you can see there is not much you need to do to prepare a PI to be used for burning the SD Card.

TODO: Python3

6.1.5.1.1 Card Burning from commandline

- Insert card and find mmcblk0, e.g. no letter p in it for partition

```
sudo ls -ltr /dev/*
```

```
sudo dd bs=1M if=~/.cloudmesh/images/imagename.img of=mmcblk0 status=progress conv=fsync
```

6.1.5.2 OSX

6.1.5.2.1 Card Burning

On OSX a good program is to use etcher for burning the images on disk:

- https://etcher.io

To access it form the commandline you can also use

- https://etcher.io/cli

6.1.5.2.2 File System Management

Unfortunately, the free versions of writing the ext file system are no longer supported on OSX. This means that as of writing of this document the best solution we found is to purchase and install extFS on the MacOS computer you use for burning the SD Cards. If you find an alternative, please let us know. (We tested ext4fuse, which unfortunately only supports read access, see Appendix)

To easily read and write ext file systems, please install extFS which can be downloaded from

- https://www.paragon-software.com/home/extfs-mac

The purchase price of the software is $39.95.

If you like to not spend any money we recommend that you conduct the burning on a raspberry pi.

TODO: PYTHON3 use pyenv

Tip: An alternative would be using virtualbox and using a virtual machine to avoid purchasing extFS.

6.1.6 WINDOWS

6.1.6.0.1 Elevate permissions for Python.exe in Windows

- Create a shortcut for python.exe
- Change the shortcut target into something like C:....exe
- Click "advance…" in the property panel of the shortcut, and click the option "run as administrator"

6.1.6.0.2 Executable needed to burn the image on SD Card:

Download CommandLineDiskImager from the following url

- https://github.com/davidferguson/CommandLineDiskImager

The previous executable will be used by cm-burn script.

It's necessary to burn the raspbian image to the SD card with this executable manually or thru Etcher in order to continue with next step.

```
CommandLineDiskImager.exe C:\Users\John\Downloads\raspbian.img G
```

6.1.6.0.3 File System Management

Download the Open source ext3/4 file system driver for Windows installer from

- http://www.ext2fsd.com
- Open Ext2fsd exe
- The burned image in the previous step in SD card will have 2 partition
- FAT32 partition will be assigned with the Drive letter - Boot Drive
- Assign Drive Letter for EXT4 (Right click on the EXT4, Assign letter. The drive letter will be used while running cm-burn) - Root Drive
- Setting Automount of this EXT4

- F3 or Tools->Ext2 Volume Management
- Check-> Automatically mount via Ext2Mgr
- The previous instructions needed for the Ext2fsd to reserve the Drive Letters and any raspbian image burned to SD will be auto mounted to the specific reserved drive letters. These drive letters need to be specified while using cm-burn

---

6.1.7 INSTALLATION

6.1.7.1 Install on your OS

Once you have decided which Computer system (MacOS, Linux, or Windows) you like to use for using the cm-burn program you need to install it. The program is written in python3 which we assume you have installed and is your default python in your terminal.

To install cm-burn, please execute

```
git clone https://github.com/cloudmesh/cm-burn.git
cd cm-burn
pip install .
```

In future it will also be hosted on pypi and you will be able to install it with

```
pip install git+https://github.com/cloudmesh/cm-burn
```

To check if the program works please issue the command

```
cm-burn check install
```

It will check if you have installed all prerequisites and are able to run the command as on some OSes you must be in the sudo list to runi it and access the SD card burner as well as mounting some file systems.

6.1.7.2 Usage

6.1.7.2.1 cmburn.yaml

```
cloudmesh:
    burn:
       image: None
```

6.1.7.2.2 Manual page

1. git clone https://github.com/cloudmesh/cm-burn
2. cd cm-burn
3. python setup.py install
4. Copy the Raspberyy PI images to be burned under ~/.cloudmesh/images

The manual page is as follows:

```
cm-burn -h
Cloudmesh Raspberry Pi Mass Image Burner.

Usage:
  cm-burn create --group GROUP --names HOSTS --image IMAGE [--key=KEY]  [--ips=IPS]
  cm-burn gregor --group GROUP --names HOSTS --image IMAGE [--key=KEY]  [--ips=IPS]
  cm-burn ls
  cm-burn rm IMAGE
  cm-burn get [URL]
  cm-burn update
  cm-burn check install
  cm-burn (-h | --help)
  cm-burn --version

Options:
  -h --help     Show this screen.
  --version     Show version.
  --key=KEY     the path of the public key [default: ~/.ssh/id_rsa.pub].
  --ips=IPS     th ips in hostlist format

Location of the images to be stored for reuse:

  ~/.cloudmesh/images
  ~/.cloudmesh/inventory


Description:
  cm-burn create [--image=IMAGE] [--group=GROUP] [--names=HOSTS]
                 [--ips=IPS] [--key=PUBLICKEY] [--ssid=SSID] [--psk=PSK]
                 [--domain=DOMAIN]
                 [--bootdrive=BOOTDRIVE] [--rootdrive=ROOTDRIVE]
                 [-n --dry-run] [-i --interactive]
  cm-burn ls [-ni]
  cm-burn rm IMAGE [-ni]
  cm-burn get [URL]
  cm-burn update
  cm-burn check install
  cm-burn hostname [HOSTNAME] [-ni]
  cm-burn ssh [PUBLICKEY] [-ni]
  cm-burn ip IPADDRESS [--domain=DOMAIN] [-ni]
  cm-burn wifi SSID [PASSWD] [-ni]
  cm-burn info [-ni]
  cm-burn image [--image=IMAGE] [--device=DEVICE]
                [-ni]
  cm-burn (-h | --help)
  cm-burn --version

Options:
  -h --help         Show this screen.
  -n --dry-run      Show output of commands but don't execute them
  -i --interactive  Confirm each change before doing it
  --version         Show version.
  --key=KEY         the path of the public key [default: ~/.ssh/id_rsa.pub].
  --ips=IPS         the IPs in hostlist format
  --image=IMAGE     the image to be burned [default: 2018-06-27-raspbian-stretch.img].
```

```
Example:
  cm-burn create --names red[000-010] ips --image rasbian_latest
  cmb-urn create --group g1 --names red[001-003] --key c:/users/<user>/.ssh/id_rsa.pub --image 2018-06-27-raspl
```

6.1.8 Appendix

6.1.8.1 OSX ext4fuse

Unfortunately ext4fuse only supports read access. To install it please use the following steps. However it will not allow you to use the cm-burn program. It may be useful for inspection of SD Cards

On OSX you will need brew and install osxfuse and ext4fuse

```
brew cask install osxfuse
brew install ext4fuse
```

To run it, your account must be in the sudoers list. Than you can do the following

```
mkdir linux
mkdir boot
cp  ../*.img 00.img
brew cask install osxfuse
brew install ext4fuse
hdiutil mount 00.img
```

This will return

```
/dev/disk3              FDisk_partition_scheme
/dev/disk3s1            Windows_FAT_32                  /Volumes/boot
/dev/disk3s2            Linux
```

We can now access the boot partition with

```
ls /Volumes/boot/
```

This partition is writable as it is not in ext format.

However to access the Linux partition in read only form we need to mount it with fuse

```
sudo mkdir /Volumes/Linux
sudo ext4fuse /dev/disk2s2 /Volumes/Linux -o allow_other
ext4fuse /dev/disk2s2 linux
less linux/etc/hosts
sudo umount /Volumes/Linux
```

6.1.8.2 Activate SSH

see method 3 in https://www.raspberrypi.org/documentation/remote-access/ssh

Draft:

Set up ssh key on windows (use and document the ubuntu on windows thing)

you will have ~/.ssh/id_rsa.pub and ~/.ssh/id_rsa

copy the content of the file ~/.ssh/id_rsa.pub into ???/.ssh/authorized_keys ??? is the location of the admin user i think the username is pi

enable ssh on the other partition while creating the fike to activate ssh

6.1.8.3 Hostname

change /etc/hostname

6.1.8.4 Activate Network

see https://www.raspberrypi.org/learning/networking-lessons/rpi-static-ip-address

6.1.8.5 Change default password

From the net (wrong method):

Mount the SD card, go into the file system, and edit /etc/passwd. Find the line starting with “pi” that begins like this:

```
pi:x:1000:1000...
```

Get rid of the x; leave the colons on either side. This will eliminate the need for a password.

You probably then want to create a new password by using the passwd command after you log in.

The right thing to do is to create a new hash and store it in place of x. not yet sure how that can be done a previous student from the class may have been aboe to do that Bertholt is firstname.

could this work? https://unix.stackexchange.com/questions/81240/manually-generate-password-for-etc-shadow

```
python3 -c "from getpass import getpass; from crypt import *; p=getpass(); print('\n'+crypt(p, METHOD_SHA512))
if p==getpass('Please repeat: ') else print('\nFailed repeating.')"
```

6.1.9 Unmount Drives on Windows

RemoveDrive.exe needs to be downloaded to c:from the following path and to have the Administrator rights (Right Click on the exe -> Properties -> Compatibility Tab -> Run this program as an Administrator

- https://www.uwe-sieber.de/drivetools_e.html

See also

- https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/remove-psdrive?view=powershell-6

Gregor thinks that unmounting is much easier in an aelevated command prompt using

```
mountvol <Drive Letter>: /d
```

6.2 Links

- https://github.com/cloudmesh-community/hid-sp18-419/blob/master/cluster/headless_setup.md
- https://medium.com/@viveks3th/how-to-bootstrap-a-headless-raspberry-pi-with-a-mac-6eba3be20b26
    - network setup is not good as it requires additional step, we want to preconfigure on sd card and plug in multiple pis at once not a single one.
- https://github.com/cloudmesh/cloudmesh-pi/blob/dev/bin/cm-burn
- http://www.microhowto.info/howto/mount_a_partition_located_inside_a_file_or_logical_volume.html
- http://www.janosgyerik.com/mounting-a-raspberry-pi-image-on-osx/
- https://github.com/Hitabis/pibakery
- http://osxdaily.com/2014/03/20/mount-ext-linux-file-system-mac/
- https://linuxconfig.org/how-to-mount-rasberry-pi-filesystem-image
- https://www.jeffgeerling.com/blogs/jeff-geerling/mounting-raspberry-pis-ext4-sd
- https://blog.hypriot.com/post/cloud-init-cloud-on-hypriot-x64/
- https://www.paragon-software.com/home/extfs-mac/

6.3 OSX DURING BURNING

```
/dev/disk0 (internal):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        2.0 TB     disk0
   1:                        EFI EFI                     314.6 MB   disk0s1
   2:                 Apple_APFS Container disk1         2.0 TB     disk0s2

/dev/disk1 (synthesized):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      APFS Container Scheme -                      +2.0 TB     disk1
                                 Physical Store disk0s2
   1:                APFS Volume Macintosh HD            811.4 GB   disk1s1
   2:                APFS Volume Preboot                 26.8 MB    disk1s2
   3:                APFS Volume Recovery                519.0 MB   disk1s3
   4:                APFS Volume VM                      9.7 GB     disk1s4

/dev/disk2 (external, physical):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:     FDisk_partition_scheme                        *31.9 GB    disk2

/dev/disk3 (external, physical):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:     FDisk_partition_scheme                        *31.9 GB    disk3
```

Experiment DIY multiSDCard writer

We intend to experiment to build a multiSD card writer via USB. We will attempt to do this for OSX initially, therefore we like to order the following product

- USB Hub 3.0 Splitter, LYFNLOVE 7 Port USB Data

We will use multiple USB card readers (possibly just USB2 till we replacethem with USB3)

Than we will rewrite our program to attempt using the SDcard writers

```

## 6.4 Cluster Setup ⭕ ☁️

No

In this section we discuss how we setup the cluster. To explore that we can conduct the setup without monitor, we are collecting in this section a variety of tasks in regards to it and documenting solutions for them. Students in the residential or in the online classes that participate in cluster building can contribute to this section.

A number of students have been assigned for particular tasks. and will be added here by the TA's. It is expected that you conduct your task ASAP.

### 6.4.1 Links

A number of useful links may help for this task. However be aware that we want to develop a *script* for most of the tasks eliminating input by hand. The information here deals to identifying the information needed to do so.

- https://hackernoon.com/raspberry-pi-headless-install-462ccabd75d0
- https://installvirtual.com/enable-ssh-in-raspberry-pi-without-monitor
```

According to

- https://www.raspberrypi.org/documentation/configuration/raspi-config.md

rasp-config writes a file /boot/config.txt. Maybe it is a good idea to inspect this file manipulate it and see if there are field we simply could write and overwrite it on the SD card. See also

- https://elinux.org/R-Pi_configuration_file

6.4.2 Add-on Hardware

recently ordered add on hardware

power switches

- https://www.amazon.com/dp/B01DE57SD4/ref=psdc_6396124011_t2_B075WZJL6N

SD Card Writers

- https://www.amazon.com/Collection-MicroSD-MicroSDHC-MicroSDXC-Kingston/dp/B01IF7TPMS/ref=sr_1_15?s=electronics&ie=UTF8&qid=1518271583&sr=1-15&keywords=Micro+SD+Card+writer

7 TRADITIONAL CLUSTER

7.1 NETWORK OF PIS (NOW) FA18-516-03

Network of Pis

The purpose of setting up a cluster of Raspberry Pi computers is to be able to experiment with different server and cluster technologies on a small scale. To this end we want to be able to use a network configuration that mirrors a large scale cloud configuration. This section will explain how to setup several Raspberry Pis in a cluster configuration to run experiments on them.

7.1.1 NETWORK OF PIS CONFIGURATIONS

There are several possible configurations for a network of Pis. One possibility is to connect each of the Pis to your local network so that you can directly connect to each of the Pis in the cluster from your laptop. This option can be easy to setup but it may have problems scaling up to several hundred Raspberry Pis. The second option is to use one of the Pi computers as a master or router and for the other Pis in the cluster to access the Internet through this master Pi's connection. Note that in most situations each Pi will need to be able to access the Internet to download packages and interact with public services. In the second option, however, the worker Pis will be separated from the main network by the master Pi and it may not be possible to directly access them, for example, over `ssh` from your laptop.

In the first option we can directly connect each Raspberry Pi to your local network using either the Ethernet adapter on the Pi or using the Wireless adapter on the Pi. If using the Ethernet adapter and intermediary router or switch can also be used to connect the Pis to the local network. The network can be visualized in Figure 28. To setup this kind of cluster please follow the instructions in Direct Network Cluster.

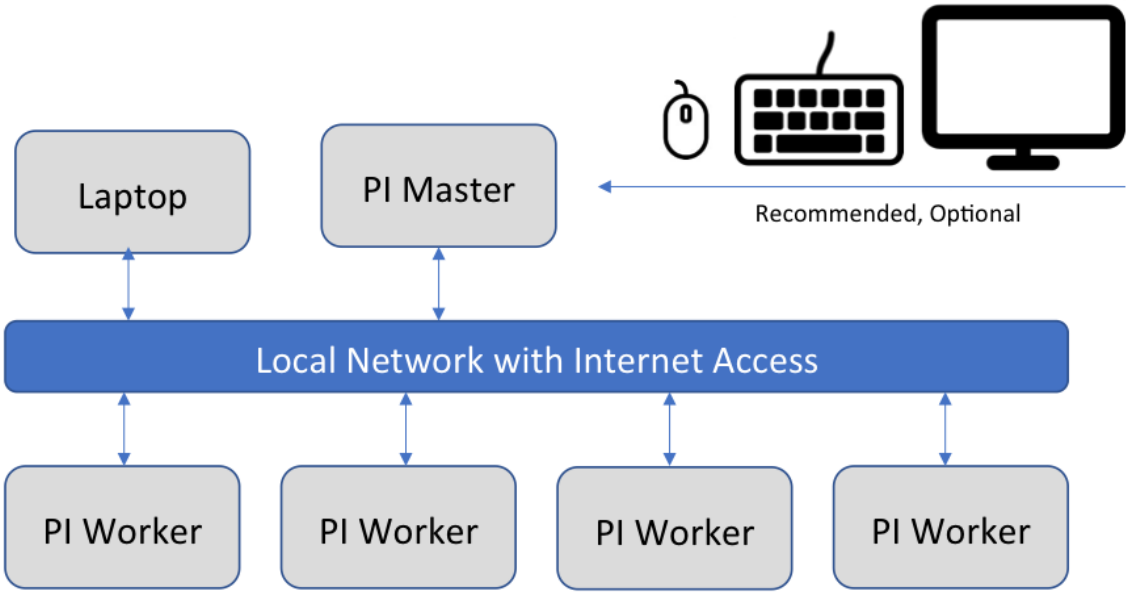


Figure 28: Direct Network Cluster

In the second option we will need to configure the master Pi to have two network interfaces enabled. One of the interfaces will connect to the local network and have direct internet access and will accessible to the other computers on the network. The second interface will be attached to the private network that the worker Pis are connected to and will serve as the DHCP server and router for that private network. Since the Raspberry Pi comes with an Ethernet adapter and a wireless network adapter, you can use the built-in Ethernet adapter on all of the Pis to connect to a switch and form a private network this way. The master Pi can then connect to your local network using its wireless adapter. Another possibility is to use a USB Ethernet adapter (purchased from this list of Raspberry Pi compatible USB Ethernet adapters) on the master Pi so that it can have a stable, wired connection to both networks. In either case the network setup is illustrated in Figure 29. The steps to setup this kind of private network cluster can be found in Private Network Cluster.

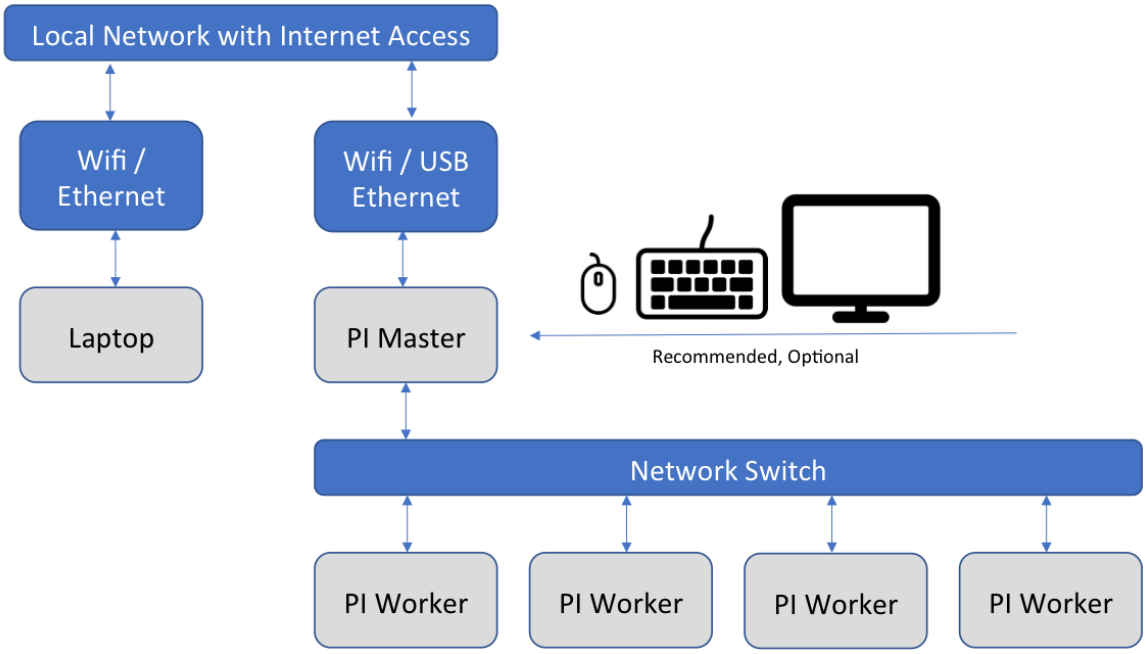


Figure 29: Private Network Cluster

7.1.2 Network of Pis Hostnames

When setting up a cluster of Pis it is necessary to assign a hostname to each Pi in the cluster. These names are important because they must not be repeated across Pi clusters and they must not conflict with other devices on the same network. If you are setting up a small network, almost any hostname will be fine, but for larger networks you should come up with a naming scheme to avoid conflicts. For a small cluster you may pick a designation such as a color or name and then assign a number to each Pi. For example, if you have three Pis you could call them `red01`, `red02`, and `red03`.

For a large cluster we recommend giving each cluster a unique id, for example, a number from 1 to 100, and then giving each node in the cluster an id that is based on the cluster that it is assigned to. For example, if each cluster is named `clusterNN` where `NN` is a zero-padded number from `01` to `99` then we would have clusters named `cluster01`, `cluster02`, ..., `cluster11`, etc. The cluster name will not be assigned to any particular Pi. Each Pi will be given a name that is based on the cluster it is assigned to. If each of your clusters are made of 5 Pis then you can number your individual Pis with the following formula:

```
node number = (cluster number - 1) * 5 + pi number in cluster
```

Our `cluster01` cluster would then have the Pis `node001` to `node005` and `cluster02` would have the Pis `node006` to `node010`. Our `cluster11` would have nodes `node051` to `node055`. We assume the lowest numbered node in each cluster is the master node. If you have more than 999 Pis or clusters with more than 5 Pis per cluster then you will have to adjust the naming scheme accordingly.

7.1.3 Pi Cluster Preparation

To prepare to setup a Pi cluster you will need to choose whether you will be setting up each Raspberry Pi by hand or by using the tools and scripts that we have developed to make this task easier and less error prone. The primary tool that will save time in setting up a Pi cluster is cm-burn which was introduced in the section Burn an SD Card with cm-burn. The installation and setup instructions for cm-burn can be found in the cm-burn README.md. Once you have `cm-burn` successfully installed you can use the instructions here to setup your cluster. We have also developed some scripts to help setup a Pi cluster in the cloudmesh-community/pi project. These scripts can be copied to the Pi after it is running to help complete various setup tasks easily. Use of these scripts will be covered in the following sections.

If you choose not to use `cm-burn` or to use our scripts we will provide the manual setup steps for you to complete. In most cases the manual steps are the exact same as the operations the scripts perform, so you can also check the manual steps if you are curious about what the scripts are doing. Before beginning the manual steps shown next, we assume you are able to burn an image to an SD card, to login to the Pi, and to complete the locale and hostname setup at a minimum. If you have not completed these steps, please see the following sections for details:

- Install Raspbian on a SD card
- Password
- Set the hostname

We also recommend using ssh keys to connect to your Raspberry Pis rather than using password authentication. Please see the SSH keygen section for details on generating a key on your laptop.

Any other required steps will be explained in the following sections.

7.1.4 Direct Network Cluster Setup

An overview of this cluster setup is included in the Network of Pis Configurations section. To complete this setup you will need to select a set of hostnames for the PIs in your cluster. Please see the Network of Pis Hostnames section for our recommendation on setting hostnames. Since each Pi in the cluster will directly connect to the local network each Pi will have the same network setup. This makes using this option easier for initial setup and experimentation with a cluster of Pis. You will need to choose whether the Pis will connect to your network through a wired Ethernet connection or through a WiFi connection. In either case you can choose to statically assign an IP address or to let each Pi get a dynamic IP address using DHCP. Using DHCP may be easier at first but it can also be a problem if you do not have a monitor connected to the Pi because you then will not know in advance the IP address that is assigned to each Pi. Please see the section Discover Pi DHCP Network Addresses for details on how to find the IP address of a device assigned by DHCP.

7.1.4.1 Direct Network Cluster Setup with cm-burn

The `cm-burn` tool directly supports setting up a cluster of Pis with direct access to your local network. You can choose to use Ethernet or wireless to connect and you can statically assign IP addresses or use DHCP. We will give examples of each use case. First, ensure that `cm-burn` is installed following the directions at Burn an SD Card with cm-burn. We will assume that you have five Pis to setup with the names red01 to red05 and the IP addresses 192.168.1.101 to 192.168.1.105 for static IPs and that your domains submask is 255.255.255.0. Please substitute the actual values of your local network here. We further assume that you have setup an ssh key so that you can login to the Pi without specifying a password. Please see the SSH keygen section for details. If you do not have an ssh key then you can leave the `--key` setting out of `cm-burn` and skip the manual sections relating to ssh keys.

7.1.4.1.1 Static IP Ethernet Setup

Static Ethernet setup is one of the easiest options with cm-burn. This command will burn 5 SD cards with the hostnames `red01` to `red05` in the 192.168.1.1 domain with IPs 192.168.1.101 to 192.168.1.105. It will copy the public ssh key from your computer onto each of the Pis and disable password logins. After each card is burned it can be removed and put into a Pi and booted. The Pi will appear on your network in about one minute after booting.

```
$ cm-burn create --name red[01-05] \
  --ips 192.168.1.[101-105] --domain 192.168.1.1 \
  --key ~/.ssh/id_rsa.pub \
  --image 2018-11-13-raspbian-stretch-lite.img
```

You should now be able to connect to the Pi over ssh:

```
$ ssh pi@192.168.1.101
```

If you would like to connect to the Pi using the hostname then you will need to setup the hosts on your host OS. On macOS and Linux this can be done by editing the `/etc/hosts` file and adding a line at the end for each of the Pis. The format is to start with the IP address, then have whitespace (blank spaces or tabs) and then the hostname. The file should look like this:

```
...
192.168.1.101     red00
192.168.1.102     red01
192.168.1.103     red02
192.168.1.104     red03
192.168.1.105     red04
```

7.1.4.1.2 Static IP WiFi Setup

Setting up static IP addresses over Wifi is very similar to doing it over Ethernet. The only difference is when you burn the SD card with cm-burn you will need to specify the wireless access point's ssid and passphrase on the command. You can use the output of `wpa_passphrase` as the `--psk-hash` parameter or you can specify the actual passphrase for the wireless network in plain text using the `--psk` parameter. We strongly recommend using the hashed passphrase for some added security. For more details on the wireless setup please see the Wireless Network at Home section.

Using the psk hash:

```
$ cm-burn create --name red[01-05] \
  --ips 192.168.1.[101-105] --domain 192.168.1.1 \
  --ssid home_network --psk-hash 0617cac0a00f87d23cda5705f5ff97bbc562f5d1907b40f02c39912a7d595b0f \
  --key ~/.ssh/id_rsa.pub \
  --image 2018-11-13-raspbian-stretch-lite.img
```

Using the actual wireless passphrase:

```
$ cm-burn create --name red[01-05] \
  --ips 192.168.1.[101-105] --domain 192.168.1.1 \
  --ssid home_network --psk "my passphrase has spaces" \
  --key ~/.ssh/id_rsa.pub \
  --image 2018-11-13-raspbian-stretch-lite.img
```

For other details on connecting to the Pis please see the Static IP Ethernet Setup section.

7.1.4.1.3 DHCP Ethernet Setup

To use DHCP over Ethernet the only change from static setup is to remove the setting of the IP addresses and the domain. Since you do not specify an IP address, cm-burn will not change the standard setup of the Pi which is to find an address using DHCP.

```
$ cm-burn create --name red[01-05] \
  --key ~/.ssh/id_rsa.pub \
  --image 2018-11-13-raspbian-stretch-lite.img
```

When the SD cards are finished you can put them into the Pi and they should boot and join the local network over DHCP in a minute or two. To find the address assigned to the Pi see the section Discover Pi DHCP Network Addresses.

7.1.4.1.4 DHCP Wifi Setup

The DHCP Wifi Setup is only different from the static IP setup in that the static IP addresses are removed from the cm-burn command. See the Static IP WiFi Setup for more details on setting the wireless ssid and psk.

Using the psk hash:

```
$ cm-burn create --name red[01-05] \
  --ssid home_network --psk-hash 0617cac0a00f87d23cda5705f5ff97bbc562f5d1907b40f02c39912a7d595b0f \
  --key ~/.ssh/id_rsa.pub \
  --image 2018-11-13-raspbian-stretch-lite.img
```

Using the actual wireless passphrase:

```
$ cm-burn create --name red[01-05] \
  --ssid home_network --psk "my passphrase has spaces" \
  --key ~/.ssh/id_rsa.pub \
  --image 2018-11-13-raspbian-stretch-lite.img
```

7.1.4.2 Direct Network Cluster Setup by hand

To setup networking on a Pi cluster by hand you can follow these steps depending on your needs. If you want to setup DHCP over Ethernet you do not need to do anything. The Pi will automatically connect to DHCP over Ethernet when it is connected to a network.

To setup static IP addresses for Ethernet or wireless connections you need to edit the `/etc/dhcpcd.conf` file and add the following lines, substituting the desired IP address and address of your local router:

```
...
interface eth0

static ip_address=192.168.1.101/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1

interface wlan0

static ip_address=192.168.1.101/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

The instructions for connecting a Pi to the WiFi network can be found in the Wireless Network at Home section.

7.1.5 Private Network Cluster Setup

An overview the design of a private Pi cluster is included in the Network of Pis Configurations section. To complete this setup you will need to select a set of hostnames for the PIs in your cluster. Please see the Network of Pis Hostnames section for our recommendation on setting hostnames.

7.1.5.1 Private Network Cluster Setup with cm-burn ⭕

⭕

Cm-burn does not currently support setting up the master node in a private network cluster. When it is enhanced to support this we will add the documentation here.

To setup the worker nodes in a cluster you can simply decide whether you are using static IP addresses or DHCP IP addresses and then use the appropriate section about setting up a direct network. For static IPs instead of using the domain of your local network you should use the domain of the private Pi network. Also, you should generate an ssh key on the master Pi and use it when setting up the worker Pis so that you can connect to them securely from the master.

7.1.5.2 Private Network Cluster Setup by hand

The master node of the cluster must use one network device to talk to the local network and another network device to talk to the other Pis on the private network. For these steps we will assume that `wlan0` is on the local network and that

`eth0` is on the private Pi network. These could be switched or replaced with a USB Ethernet connection with no change to the steps. We assume for these steps that you have already connected `wlan0` to your local network and these steps will then complete setting up `eth0` as the bridge device to the private Pi network.

We need the `dnsmasq` service as a simple DNS server and the convenience package `iptables-persistent` for making changes to iptables:

```
$ apt-get update
$ apt-get install -qy dnsmasq iptables-persistent
```

To setup our `wlan0` as the *favored* interface for the Pi to communicate over the Internet we need to set its metric lower than the `eth0` interface. Normally the Pi will prefer to use the `eth0` interface since it is usually faster. This change can be made in `/etc/dhcpcd.conf`. This file also where we setup static IP addresses. If you are not using a static IP address for `wlan0` then you will not have the lines beneath `interface wlan0` to set the static IP address. We will setup our private Pi network to have the IP address range 192.168.50.1 to 192.168.50.255 which means it is 192.168.50.1/24 or, equivalently, uses the 255.255.255.0 subnet mask. You can freely change this and you must choose a network that does not match the local network. Change `/etc/dhcpcd.conf` to match this:

```
interface eth0
metric 300

static ip_address=192.168.50.1/24
static routers=192.168.50.1
static domain_name_servers=192.168.50.1

interface wlan0
metric 200

static ip_address=192.168.1.107/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Next you need to update `/etc/dnsmasq.conf` to include the following lines to enable giving out DHCP addresses on the `eth0` network and to give out addresses in the proper range:

```
interface=eth0
dhcp-range=eth0, 192.168.50.2,192.168.50.250,24h
```

We then need to enable NAT Forwarding by uncommenting (or adding) the following line in `/etc/sysctl.conf`:

```
net.ipv4.ip_forward=1
```

The final step is to setup and save our iptables configuration to do the actual forwarding of packets. Run these commands to set this up properly:

```
$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
$ sudo iptables -A FORWARD -i eth0 -o wlan0 -j ACCEPT
$ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
$ sudo iptables-save | sudo tee /etc/iptables/rules.v4
```

At this point you can restart the services and everything should be working:

```
$ sudo service dhcpcd restart
$ sudo service dnsmasq restart
```

To find out what IP address has been assigned to each Pi you can use the Discover Pi DHCP Network Addresses section or you can manually look in the `/var/lib/misc/dnsmasq.leases` which will list each lease and the MAC Address of the device it is leased to.

7.1.6 Discover Pi DHCP Network Addresses

If you setup your Pis using DHCP on your local network then you may not know the IP address that has been dynamically assigned to each Pi. If you have statically assigned IP addresses to each Pi then you will need to make node of these assignments and add the hostname mapping to each device that needs to be aware of the hostnames.

If you have physical access to each Pi and a compatible monitor and keyboard then you can login to each of them in sequence and then run `ifconfig` to determine which IP address has been assigned to each of them. If you have access to the DHCP server that assigns IP address (for example, in your home network) you can also usually access that device through a web browser to find out which IP address has been assigned to each device on the network. If you have properly configured the hostname on each Pi then it should be registered with that name on your DHCP server.

It is not trivial to detect all of the devices on a local network. In addition, if you use static networking then the devices will typically not register or report their hostnames. However, if you are using DHCP and you have properly configured the hostname on each Pi, then the following method should work.

To begin you need the `nmap` tool installed on your system. It can be installed on Linux (on a Pi, for example) using the standard package installation tools such as

`sudo apt-get install nmap`. If you are using Windows or macOS, please see the Nmap installation instructions or use Homebrew on macOS as `brew install nmap`.

To find the Pis you must be on the same network as they are. If you are using the Direct Network Cluster setup then the Pis will all be on the same network as your laptop. If you are using the Private Network Cluster setup then only the master Pi will be on your local network. If you want to discover the IP address of the Pis on the private network then you should first login to the master Pi node and then execute the following commands.

This works on a Pi substitute your network address range for `192.168.1.0/24`. The first command `nmap -sn` will search your local network IP address range for any devices attached to the network. This process is to find out which devices are reachable from the host. As a result of the `nmap` process, the host's ARP table will be updated with a record of every device (up to the arp cache size limit but this is probably larger than you will need) on the local network. You can then use the `arp -a` command to list the devices that were found. `arp` will show all devices on any network reachable from this computer, so if you are running this on the master Pi then it will show devices on both the local network and the private Pi network. You can filter the `arp` results by hostname or IP address range if you would like using `grep`. Note: if you see a lot of results from `arp` listed as `(incomplete)` that is OK it means there is probably not a device at that IP address but the OS is still waiting for a response. Every OS has a different timeout for responses and any incomplete entries should eventually disappear.

```
# optional: if you want to you can clear the arp cache first
$ sudo arp -a -d
# Search for devices on the local network
$ nmap -sn 192.168.1.0/24
# will list devices in arp cache and lookup hostname
$ arp -a
# only show results with hostnames starting with "red"
$ arp -a | grep '^red'
# only show results with IP addresses on the specified network
$ arp -a | grep '192.168.1.'
```

In the following example output from `arp -a`, the entry for `blue02` is a Raspberry Pi set to DHCP. The entry for `cred` is my laptop. The entry listed first with the IP `10.0.0.103` is a Pi set to a static IP address and the `10.0.0.17` is another device on my network. Even though `arp` lists the fully qualified domain name, you can directly access a host with just the first part of the name as long as you are also on the same local network (which you must be or `nmap` and `arp` would not list the address).

```
? (10.0.0.103) at b8:43:eb:6e:cf:b7 [ether] on wlan0
? (10.0.0.17) at 10:29:92:53:9e:1b [ether] on wlan0
cred.hsd1.in.comcast.net (10.0.0.90) at e0:f8:8e:2d:34:79 [ether] on wlan0
blue02.hsd1.in.comcast.net (10.0.0.21) at b8:27:b3:73:8d:a3 [ether] on wlan0
```

If you are trying to determine whether your DHCP server contains an entry for a particular device you can use the `dig` tool to determine this. `dig` is not installed by default on a Pi but can be installed with `sudo apt-get install dnsutils` and you can lookup a host on any nameserver or you can specify your local router with the `@` symbol:

```
# lookup red01 on all nameservers
$ dig red01
# lookup red01 on the local router DNS
$ dig red01 @192.168.1.1
```

If `dig` is successful you should see something like this:

```
;; ANSWER SECTION:
red01.                  0       IN      A       192.168.1.43
```

7.1.7 SSH KEYGEN

An ssh key is a secure means to verify your identity to another computer. Ssh keys can be used to login to a remote computer without needing a password. This enhances security because an attacker cannot attempt to crack the password. However, the private keys that are stored on the client computer are a potential weakness and must be carefully protected to ensure that they are not compromised. If you would like more information on SSH keys the GitHub SSH guide is highly recommended.

To generate a new ssh key on macOS or Linux use the `ssh-keygen` program. The same procedure can be followed on Windows by using Git Bash. It will save your key by default in your home folder in `~/.ssh/id_rsa` and the public key in `~/.ssh/id_rsa.pub`. It is more secure if you supply a passphrase. If you do supply a passphrase then it must be entered any time you want to use the key. If you do not supply a passphrase then the private key can be used by anyone and if someone has a copy of it they can impersonate you and gain access to any computer that you have access to.

```
$ ssh-keygen
```

There is also a command to copy your ssh public key to other computers if you have password access to them already. This can be useful to do when setting up the private Pi cluster.

```
$ ssh-copy-id <hostname>
```

7.1.8 CONFIGURE CLUSTER SSH

This was moved from the Kubernetes section.

Install Dependencies:

```
$ apt-get install -qy clusterssh
```

For hostnames rp1-4 (final node names will be: rp0, rp1, rp2, rp3, rp4).

To update Cluster SSH configuration, add the following to **/etc/clusters**:

```
$ rpcluster rp1 rp2 rp3 rp4
```

Now you can run commands to all clusters by:

```
$ cssh rpcluster
```

7.1.9 Parallel Shell

TODO

7.1.10 Cloudmesh Parallel

TODO

7.1.11 Other Parallel Execution

TODO

- https://www.rittmanmead.com/blog/2014/12/linux-cluster-sysadmin-parallel-command-execution-with-pdsh/
- https://www.linux.com/news/parallel-ssh-execution-and-single-shell-control-them-all
- https://www.tecmint.com/using-dsh-distributed-shell-to-run-linux-commands-across-multiple-machines/
- https://github.com/vallard/psh
- https://github.com/karrick/psh/blob/master/psh

7.2 Message Massing Interface Cluster ☁️

For more information on this topic see:

- https://thenewstack.io/installing-mpi-python-raspberry-pi-cluster-runs-docker
- https://www.hackster.io/darthbison/raspberry-pi-cluster-with-mpi-4602cb
- https://www.meccanismocomplesso.org/en/cluster-e-programmazione-in-parallelo-con-mpi-e-raspberry-pi

7.3 SLURM ⭕ ☁️

⭕

This may be inspiring

- https://github.com/ajdecon/ansible-pi-cluster/blob/master/README.md

7.3.1 Installation

provide a information

7.3.2 Configuration

provide a information

7.3.3 Example MPI program

provide a information

7.3.4 Using the Batch Queue

provide a information

7.3.5 REST Services

develop OpenAPI REST services to interact with the cluster

7.4 PXE Booting ⭕ ☁️

⭕

TODO: Problem description

TODO: provide an example architecture drawing

TODO: provide the actual steps

7.4.1 Resources

Network booting information can be found here:

- https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/net_tutorial.md
- http://web-docs.gsi.de/~bloeher/howto/rpi3_netboot.html

7.5 Fortan ☁️

Although this section is not about parallel programming, you may find still that many scientific programs are written in fortran. For some smaller fortran programs it is even possible to run them on a Raspberry pi. Naturally you will need to install a fortran compiler, which you can do with

```
pi$ sudo apt-get install gfortran
```

To test it out store the following program into `hello.f90`

```
program hello
    print *, "Hello World!"
end program hello
```

Now you can compile it with

```
pi$ gfortran -o hello hello.f90
```

Execute it with

```
pi$ hello
```

8 CLOUD CLUSTER

8.1 DOCKER

Docker is a tool that allows us to deploy applications inside of software containers. A container allows a developer to package the application along with dependencies associated with it and put all in a box which is an isolated environment so that the underlying host operating system is completely abstracted from the application running inside the box.

It is a method of packaging software, to include not only our code, but also other components such as a full file system, system tools, services, and libraries. This can be useful for the Raspberry Pi because it allows users to run applications without lot of steps, as long as the application is packaged inside of a Docker image. We simply install Docker and run the container.

According to the developers of Docker it includes the following features:

- Portability
- Density
- Scalability
- Security

8.1.1 INSTALLATION

First we need to make sure the Raspberry Pi is up to date so we can install a recent version of docker. The automated script maintained by the Docker project will create a systemd service file and copy the relevant Docker binaries into `/usr/bin/`.

```
pi$ sudo apt-get update
pi$ curl -sSL https://get.docker.com | sh
```

In order for us to start the docker daemon at the next boot, we add it as follows:

```
pi$ sudo systemctl enable docker
```

Now if we reboot, the Docker daemon will start. In case you like to avoid the first reboot, you can use the command:

```
pi$ sudo systemctl start docker
```

Naturally you do not have to do this after you reboot the next time.

The Docker client can only be used by `root` or members of the `docker` group. Thus, let us add the user pi (or your equivalent user) to the docker group using:

```
pi$ sudo usermod -aG docker pi
```

After executing the previous command, we log out of the terminal restart it so we are sure the user permissions are available in the shell we use.

To test docker is installed successfully, we run the `hello-world` docker image with the command:

```
pi$ docker run hello-world
```

If Docker is installed properly, we will see a `Hello from Docker!` message.

8.1.2 Docker Swarm

Swarm is a native clustering and scheduling tool for Docker. Instead of just managing containers on a single server, we can manage containers on a set of servers. The containers will be automatically scheduled on the pool of servers making them appear as a single resource. We will set up and use Docker on a number of Raspberry Pi's install Docker on them and register them into a Docker Swarm.

8.1.3 Creating a Network of Pi's with docker

In Network of Pis section we explained how to set up a network of Pis. Here we assume that we start from such a network. The Pi's have all different names, and are registered on the network. Each Pi has the public

key installed from the machine where you will login from for setting up the swarm.

Let us assume the names of the hosts are stored in a shell variable called

```
hostnames = (red00 red01 red02 red03 red04)
```

Naturally, we want to install on these machines docker and register them to the swarm. A variety of tools exist to simplify this process, such as

- parallel shell https://github.com/vallard/psh
- cloudmesh parallel (TODO: find the link)

For now we use this simple shell program to install docker on each of the hosts in the hostnames

```
hostnames = (red00 red01 red02 red03 red04)
for host in "${hostnames[@]}"
do
      ssh pi@$host curl -sSL https://get.docker.com | sh
done
```

Save this script in a file called `docker-install.sh` and set the executable rights with

```
chmod u+x docker-install.sh
```

When we execute it with

```
$ docker-install.sh
```

It will sequentially install docker on each host. This is not very efficient and only works for a small number of hosts.

8.1.4 Registering the Pi to the Swarm

Next we need to run on one of the nodes the management node for the swarm to which all others servers register as workers. Although we could run on this node als a worker, we will just run the manager on it as we want to avoid overloading it and make sure it operates smoothly.

We select the first host in our hostlist for it called `red00` Let us assume the host has the ipaddress `<manager-ip-address>`. We can log into this computer and execute the command

```
$ sudo docker swarm init --advertise-addr <manager-ip-address>:2377
```

This command will print out a token that we can use on the workers to register with our swarm. The token will look something like:

```
SWMTKN-abc...xyz
```

Let us use the term `<token>` to indicate the token. To register a worker a two step process is used.

If you ever forget the token, you simply can use the following command on the manager

```
$ docker swarm join-token worker
```

It will print out the command that you will have to execute on a worker.

```
$ sudo docker swarm join --token SWMTKN-abc...manager...xyz <manager-ip-address>:2377
```

To see the list of nodes, you can use the command

```
I $ sudo docker node ls
```

8.1.5 Docker Cheat Sheet

The following table is copied from the [docker manual]

.

Purpose	Command
Image	
Build an image	docker image build –rm=true .
Install an image	docker image pull ${IMAGE}
List of installed images	docker image ls
List of installed images (detailed listing)	docker image ls –no-trunc
Remove an image	docker image rm ${IMAGE_ID}
Remove unused images	docker image prune
Remove all images	docker image rm $(docker image ls -aq)
Containers	

Purpose	Command
Run a container	docker container run
List of running containers	docker container ls
List of all containers	docker container ls -a
Stop a container	docker container stop ${CID}
Stop all running containers	docker container stop $(docker container ls -q)
List all exited containers with status 1	docker container ls -a –filter “exited=1”
Remove a container	docker container rm ${CID}
Remove container by a regular expression	docker container ls -a | grep gregor | awk ‘{print $1}’ | xargs docker container rm -f
Remove all exited containers	docker container rm -f $(docker container ls -a | grep Exit | awk ‘{ print $1 }’)
Remove all containers	docker container rm $(docker container ls -aq)
Find IP address of the container	docker container inspect –format ‘{{ .NetworkSettings.IPAddress }}’ ${CID}
Attach to a container	docker container attach ${CID}
Open a shell in to a container	docker container exec -it ${CID} bash
Get container id for an image by a regular expression	docker container ls

8.1.6 Exercise

Swarm.1

> Your task is is to identify technologies to execute the Installation in parallel. Suitable technologies include

- psh
- ansible
- puppet
- python threads
- cloudmesh

We like that the class is split up in groups and each group develops this solution. Naturally you can test this first with not installing docker, but with a simple command such as `uname -a`

Swarm.2

> Develop a python cloudmesh command called

```
cms swarm config hostnames.yaml
```

where the yaml file looks something like

```
manager: <ip00>
worker:
- <ip01>
- <ip02>
- <ip03>
```

Similarly create other convenient functions such as

- `cms swarm kill`, which kills the swarm
- `cms swarm ls`, which gives details about the swarm

8.2 Kubernetes 🔧 fa18-516-03 ☁

In this section we will discuss how to set up a Kubernetes cluster on a number of Raspberry Pis.

8.2.1 Todo

- [ ] all the simple setup with sd cards, ssh, keys, and so on should be moved to the NOW cluster section. This way we can require simply a NOW and start without duplication on the real Kubernetes install.
- [x] we have two sections of Kubernetes contributed by two students. What we need is to merge them and save the usable things. We need to identify if the setup is significantly different before we can do this.
- [ ] so before you can work on the Kubernetes section you need to make sure the NOW section is up to date.

8.2.2 Resources Needed

In Network of Pis section we explained how to set up a network of Pis. Here we assume that we start from such a network. We recommend that the cluster will have at least one master and three worker nodes. The test should not use too many resources otherwise the system may be unnecessarily slow. In particular we should have one dedicated master. We use three nodes to support testing the distribution of containers. (It may work with two, but we have not tested it). Please give us feedback on this and let us know what works for you. We will integrate your feedback.

8.2.3 Overview of Kubernetes Cluster Setup

A Kubernetes cluster is made of one master and several worker nodes. Each node must have the standard Kubernetes setup completed and the master must also have additional setup. Once the master and worker nodes are setup then the worker nodes can join the network created by the master node. For the Raspberry Pi we support two modes of setting up the master and workers. The first method is to use the scripts that we provide to do the required installations. The second method is to perform each step by hand. We will begin by explaining how to use the scripts to setup your cluster quickly.

8.2.4 Kubernetes Cluster Setup with Scripts

These steps have been verified with the latest build of Raspbian Stretch which is 2018-11-13-raspbian-stretch-lite. If you have installed Raspbian Stretch with Desktop or Raspbian Stretch with Desktop and Recommended Software then some of these steps will not be required, but repeating them will not be a problem.

The required scripts are stored in the Cloudmesh Community Pi repository and must be copied to each Raspberry Pi in order to run. This guide assumes that each Pi has internet access which is required to download the necessary tools. The first steps to setup the Pi tools is listed on the README.md for the Pi tools repository. We will repeat those steps here for convenience.

8.2.4.1 Pi Tools Prerequisites

To use the Cloudmesh Pi tools you need `git` to download the tools from the github repository. You must also update the Pi's list of software, install `git`, and then download or clone the `git` repository. Run these steps at the Pi command prompt:

```
$ sudo apt-get update; sudo apt-get install -y git
$ git clone https://github.com/cloudmesh-community/pi.git
```

When that successfully completes you will have a copy of the Pi tools on your Pi and you can now run them.

8.2.4.2 Kubernetes Shared Setup

Every Kubernetes node, whether master or worker, needs to complete the following setup steps. The Pi scripts are stored in the `bin` directory. Every Kubernetes Pi master and worker must run the `kubernetes-setup.sh` script which will download and install Docker and Kubernetes and make the necessary system changes to support both. When this script completes the Pi must be rebooted to properly configure its memory system for Kubernetes. Execute the following commands:

```
$ sudo pi/bin/kubernetes-setup.sh
$ sudo reboot
```

If you are connected to the Pi over `ssh` your session may hang at this point. You can either wait for `ssh` to timeout or kill the session by typing a tilde then a period. The tilde on a new line is a special command to `ssh` and the period means to disconnect the session.

```
~.
```

At this point the worker is ready to connect to the Kubernetes master node. The command to connect to the master node is `kubeadm join` but we need to finish setting up the master node in order to get the token necessary to authenticate with it.

8.2.4.3 Kubernetes Master Setup

To setup the Kubernetes master node you should first complete the Kubernetes Shared Setup. After the Pi reboots you can run the master setup script:

```
$ sudo pi/bin/kubernetes-master-setup.sh
```

The master setup script will run `kubeadm init` which can take a long time and will occasionally timeout on the Raspberry Pi without completing. This

does not indicate a failure of the Pi setup. If the command finishes with the error

```
Unfortunately, an error has occurred:
        timed out waiting for the condition
```

then it is possible to restart the setup and it will usually complete successfully the second time. To do this (only if the master setup failed) run `kubeadm reset` and be sure to answer `y` to the prompts. Then run the master setup script again:

```
$ sudo kubeadm reset
$ sudo pi/bin/kubernetes-master-setup.sh
```

When the master setup successfully completes you should see:

```
Your Kubernetes master has initialized successfully!
```

and there will be further instructions on how to setup the master. These steps have already been performed by the setup scripts so you do not need to do them. The output will also list the required `kubeadm join` command that can be issued on each worker node that wishes to join this Kubernetes master node. In addition, the scripts have stored the join command, the master IP address, the join token, and the CA Hash in a YAML file `kubeadm-settings.yml` in the current directory. If you need to add nodes in the future, you may refer to this file for the required parameters.

8.2.5 Join Workers to Master

Now login to each of the workers and issue the `kubeadm join` command from the master node. If you have not successfully completed the master node setup, please see Kubernetes Master Setup for the required steps.

As of this writing there is a version incompatibility between the latest Kubernetes and the latest Docker. Kubernetes has not yet verified Docker version 18.09 which is installed by the default Docker install script. If you use our provided setup script then the version of Docker will be automatically downgraded to 18.06.1 which is verified by Kubernetes. You can follow the steps to downgrade your Docker version given in Install

Docker or you can skip the version check by specifying `--ignore-preflight-errors=SystemVerification` on the command line. An example `kubeadm join` command would be:

```
$ sudo kubeadm join 10.0.0.101:6443 \
    --token vstt3y.faa67q2dp383xhgv \
    --discovery-token-ca-cert-hash \
    sha256:7fa06185f14b89234235aa9f03ef60835ade825e2553cd97a52b5894566edeb5
```

Once the worker nodes have joined the cluster, you can login to the master node and see their status with the following command:

```
$ sudo kubectl get nodes
NAME     STATUS   ROLES    AGE     VERSION
blue00   Ready    master   4h56m   v1.12.2
blue01   Ready    <none>   4h44m   v1.12.2
blue02   Ready    <none>   4h46m   v1.12.2
blue03   Ready    <none>   4h42m   v1.12.2
blue04   Ready    <none>   4h1m    v1.12.2
```

When the workers are joining the cluster they will initially be in a `NotReady` state for a while as they complete their setup. This is the normal expected behavior and each node should reach the `Ready` state within a few minutes. To continue experimenting with your Kubernetes cluster, please see the Kubernetes First Steps section.

8.2.6 Manual Kubernetes Cluster Setup

If you do not want to use our setup scripts or would like to change some steps in the installation you can use the following steps to manually setup a Kubernetes cluster on several Pis. First, each node in the cluster must have Docker and Kubernetes installed along with some system configurations. Then the master should be launched and each worker node connected to the master. Please follow these instructions carefully and you should have a working Kubernetes cluster.

8.2.6.1 Install docker

Kubernetes depends on a containerization platform to run applications. The standard platform used with Kubernetes is Docker, although other container platforms are also support. We will use Docker on the Pi. Install Docker with the convenience script at `get.docker.com`. You may also download the script

manually and see what operations it performs. The basic steps in the script are to detect your operation system and computer architecture, setup the proper Docker package repositories and keys for your system, and finally install the Docker packages and dependencies. The current version of Docker 18.09 has not been verified by Kubernetes, so after installation we will downgrade this package to 18.06 in the following steps. Once the installation finishes we recommend running the following `usermod` command to run Docker as a non-root user. This is an optional but recommended step.

```
$ curl -sSL get.docker.com | sudo sh
$ sudo usermod -aG docker pi
```

The current version of Docker 18.09 installed by the convenience script has not been verified by Kubernetes v1.12.2 yet so it will give an error and Kubernetes will fail to start. You can fix this by giving the `--ignore-preflight-errors=SystemVerification` flag to `kubeadm init` and `kubeadm join`. However, a better solution is to downgrade the Docker version installed. The Docker install script will also setup the proper `apt-get` repositories that host previous versions of Docker. To downgrade, first stop the `docker.service` that is running, then downgrade the package and restart the service. The Docker service should restart automatically after the downgrade, so restarting the service with `systemctl` is just done in case of a problem.

```
$ sudo systemctl stop docker.service
$ sudo apt-get install -qy --allow-downgrades \
  docker-ce=18.06.1~ce~3-0~raspbian
$ sudo systemctl start docker.service
```

8.2.6.2 Install Kubernetes

After installing Docker we must also install Kubernetes. The Kubernetes package sources and GPG keys need to be added to the `apt-get` package manager first, then we must update the `apt-get` package list with the new sources before we can finally install the Kubernetes services and administration package `kubeadm`.

```
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |\
  sudo apt-key add -
$ echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" |\
  sudo tee /etc/apt/sources.list.d/kubernetes.list
$ sudo apt-get update -q; sudo apt-get install -qy kubeadm
```

Once Kubernetes and Docker are correctly installed there are some system configuration changes necessary for Kubernetes.

8.2.6.3 System configuration

Kubernetes is not compatible with SWAP memory and as of version 1.8 it will fail if swap is enabled on a node, therefore we need to disable swap memory on Raspbian. It is possible that disabling swap on the Pi could cause other issues, especially due to the low memory on the Raspberry Pi, but it is accepted practice for running any Kubernetes cluster to disable swap. There is a flag `--fail-swap-on=false` that can be passed to `kubeadm` to skip the check for swap but we have not tested with this setting. To disable swap execute the following commands on the Pi:

```
$ sudo dphys-swapfile swapoff && \
  sudo dphys-swapfile uninstall && \
  sudo update-rc.d dphys-swapfile remove
```

You should now not see any entries in this command:

```
$ sudo swapon --summary
```

Next some kernel cgroup settings need to be changed for Kubernetes. This is a boot-time option that can only be changed by altering the options passed to the Linux kernel during boot. These options are stored in the file `/boot/cmdline.txt` on the Raspberry Pi. This file only contains a single line that specifies the kernel options. The following three options must be added to the end of the line `cgroup_enable=cpuset cgroup_memory=1 cgroup_enable=memory`. You may edit the file in a text editor if you are confident that you can make the change correctly or simply run the following lines at the command prompt. They will first backup the current file, then then append the new options and finally write the entire string back to the original file:

```
$ sudo cp /boot/cmdline.txt /boot/cmdline.bak.txt
$ new_options="$(head -n1 /boot/cmdline.txt) \
cgroup_enable=cpuset cgroup_memory=1 cgroup_enable=memory"
$ echo "$new_options" | sudo tee /boot/cmdline.txt
```

Kubernetes also expects certain kernel modules to be loaded. It will enable these kernel modules during setup but we can also specify them to always be loaded on boot which removes the warning messages. To enable the

required kernel modules, execute the following lines which will append these modules to the list of enabled kernel modules stored in `/etc/modules`.

```
$ cat << EOF | sudo tee -a /etc/modules
ip_vs
ip_vs_sh
ip_vs_rr
ip_vs_wrr
nf_conntrack_ipv4
EOF
```

Since these changes only take place at boot time, you *must reboot* before continuing with the rest of the section. If you do not reboot then Kubernetes will refuse to run and issue an error.

8.2.6.4 Setup Kubernetes Cluster

After the Pi reboots and you reconnect to it there are a few steps to perform on the master Kubernetes node to prepare it for the worker nodes to connect to. First we recommend pulling (downloading) the Kubernetes images so that this step is separate from initializing the cluster. You can issue the following command and note that it will take several minutes depending on your network connection:

```
$ sudo kubeadm config images pull
```

Once that is complete we can now initialize the Kubernetes master. You need to know the IP address of the master node and you should choose a CIDR for the pod network. Note that in this case we are setting the join token to have a time-to-live of 0 which means it will never expire. This is reasonable for initial setup and testing but in any permanent system the token should be allowed to expire in a few hours or days to prevent unauthorized nodes from joining the cluster should the token accidentally be leaked. The following command will setup the master:

```
$ POD_CIDR=10.244.0.0/16
$ APISERVER_IP=10.0.0.101
$ sudo kubeadm init --token-ttl=0 \
  --pod-network-cidr="$POD_CIDR" \
  --apiserver-advertise-address="$APISERVER_IP"
```

When the master setup completes successfully then you should see output similar to the following:

```
Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join 10.0.0.101:6443 --token vstt3y.faa67q2dp383xhgv --discovery-token-ca-cert-hash sha256:7fa06185
```

You should follow these instructions, running these commands:

```
$ mkdir -p "$HOME/.kube"
$ sudo cp -i /etc/kubernetes/admin.conf "$HOME/.kube/config"
$ sudo chown "$(id -u)":"$(id -g)" "$HOME/.kube/config"
```

The `kubeadm join` command should be copied and stored for later use. If you lose the details you can view the existing tokens with `kubeadm token list`

```
$ sudo kubeadm token list
```

And you can find the sha256 hash of the CA Cert with:

```
$ openssl x509 -in /etc/kubernetes/pki/ca.crt -noout -pubkey |\
  openssl rsa -pubin -outform DER 2>/dev/null | sha256sum |\
  cut -d' ' -f1
```

The original token is for development use and so we set it to have an unlimited time-to-live. This is not recommended for a production system, however, a shorter TTL of a few hours or days should be specified and a new token should be generated when the previous one has expired. A new token can be created with the following command.

```
$ sudo kubeadm token create --print-join-command
```

This will not retrieve the original token but will generate a new one. These tokens should be carefully managed as they allow a node to join the Kubernetes cluster which is a potentially unsafe operation for untrusted nodes.

The final step is setting up the networking. These instructions use Weave Net to enable the Kubernetes network architecture. Another recommended

solution to use with the Raspberry Pi is Flannel. Here is the command to setup Weave Net on the Kubernetes cluster.

```
 kubectl apply -f \
"https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
```

After waiting a minute or so, you should see the following output from this command:

```
 $ sudo kubectl get pods --namespace=kube-system
NAME                              READY   STATUS    RESTARTS   AGE
coredns-576cbf47c7-hn55k          1/1     Running   0          4m51s
coredns-576cbf47c7-nvmm4          1/1     Running   0          4m51s
etcd-blue00                       1/1     Running   0          3m55s
kube-apiserver-blue00             1/1     Running   0          4m7s
kube-controller-manager-blue00    1/1     Running   0          4m5s
kube-proxy-9xwdn                  1/1     Running   0          4m51s
kube-scheduler-blue00             1/1     Running   0          4m
weave-net-xj4tc                   2/2     Running   0          73s
```

Now that the master is running and networking is enabled, you can run the `kubeadm join` command on each Pi in the cluster and issue the identical command for them to join the master. Please see the section Join Workers to Master for more details.

8.2.7 Kubernetes First Steps

Now that you have the Kubernetes cluster running you can deploy pods on the cluster. For production use of Kubernetes it is recommended to use a Controller which will manage the details of deploying pods to nodes and ensuring replication and self-healing. Please see the Kubernetes Pod Overview section of the Kubernetes documentation for information on creating Deployments, StatefulSets or DaemonSets on your cluster. For our simple use case we will create a few pods by hand.

8.2.7.1 Kubernetes Pods

The Cloudmesh Community Pi repository has several pod definition files that you can use directly or customize to your particular use case for experimenting with the Kubernetes cluster. If you used our install scripts you should already have these on your Kubernetes master. If you did the

setup by hand you can get the repository by installing `git` and then cloning it onto the Pi using the instructions in the Pi Tools Prerequisites section.

As an initial test you can create the markdown renderer deployment on your cluster with the command:

```
$ cd pi/kubernetes
$ sudo kubectl create -f markdownrender.yml
service/markdownrender created
deployment.apps/markdownrender created
```

If you look at the markdownrender.yml file you will see that it defines a Service and a Deployment and it maps port 8080 from the Service to the external port 31118. The Deployment specifies that it should have one replica of the application which runs a Docker container from Docker Hub titled functions/markdownrender:latest-armhf. This container supplies a simple service to translate a markdown document into an HTML document. You can test it with the following commands to see it working:

```
$ curl -4 http://localhost:31118 -d "# test"
$ curl -4 http://localhost:31118 --data-binary @../README.md
```

The 31118 port will also be accessible to any computer that can reach the Kubernetes master (unless firewall rules dictate otherwise) so you can also test this from your own computer if it is on the same network as the Kubernetes master. For example you could run:

```
$ curl -4 http://blue00:31118 -d "# test"
$ curl -s \
  https://raw.githubusercontent.com/cloudmesh-community/pi/master/README.md |\
  curl -4 http://blue00:31118 --data-binary @-
```

You can see the status of your pods by using `kubectl get pods` on the master. Adding the `-o wide` parameter will also output the node and node ip of the node that the pod is deployed to.

```
$ sudo kubectl get pods -o wide
NAME                              READY   STATUS    RESTARTS   AGE   IP
NODE     NOMINATED NODE
markdownrender-7d8d6f74d6-67bsg   1/1     Running   0          20m   10.44.0.1
blue01   <none>
```

```
sudo kubectl get pods -o jsonpath='{.items[*].spec.nodeName}'
```

8.2.7.2 Removing a node from a cluster

To remove a Kubernetes node from a cluster, you must first drain the node which will evict every pod in the node and cordon it off so that no new pods will be scheduled in it. For cluster setup each node will be running the Weave daemon so it is necessary to specify `--ignore-daemonsets` to drain the node. The drain command should complete without errors.

```
$ sudo kubectl drain --ignore-daemonsets <node>
```

If there are pods scheduled to the node then you should wait until those pods complete and are shutdown and removed from the node. You can observe the node state using `kubectl get nodes` and `kubectl get pods`. You could see a sequence of events such as this when draining a node that has a deployed pod.

```
$ sudo kubectl drain blue01 --ignore-daemonsets
node/blue01 cordoned
WARNING: Ignoring DaemonSet-managed pods: kube-proxy-2zplz, weave-net-r7b85
pod/markdownrender-7d8d6f74d6-67bsg evicted
$ sudo kubectl get pods -o wide
NAME                              READY   STATUS              RESTARTS   AGE   IP       NODE     NOMINATED NO
markdownrender-7d8d6f74d6-lqk58   0/1     ContainerCreating   0          14s   <none>   blue03   <none>
$ sudo kubectl get pods -o wide
NAME                              READY   STATUS    RESTARTS   AGE   IP          NODE     NOMINATED NODE
markdownrender-7d8d6f74d6-lqk58   1/1     Running   0          24s   10.39.0.1   blue03   <none>
```

When the node has no pods scheduled to it you can remove it from the cluster permanently by issuing the `delete` command on the master. The node should be in the `Ready,SchedulingDisabled` status. For example, in this output the node `blue04` has been drained and is ready to be deleted from the cluster.

```
$ sudo kubectl get nodes
NAME     STATUS                     ROLES    AGE     VERSION
blue00   Ready                      master   40h     v1.12.2
blue01   Ready                      <none>   40h     v1.12.2
blue02   Ready                      <none>   40h     v1.12.2
blue03   NotReady                   <none>   40h     v1.12.2
blue04   Ready,SchedulingDisabled   <none>   4m10s   v1.12.2
```

```
$ sudo kubectl delete node <node>
```

Once this is complete you can login to the node itself and reset it so that it can join another cluster or be used for other purposes. The `kubeadm reset` command will accomplish this. At this point Kubernetes should be shut down on the Pi and you should not see any entries in the `systemctl` table for `kubernetes` or `kubelet` and you should not see any running Docker images that are related to Kubernetes. This can be confirmed with the following commands run *on the node* not on the master:

```
$ sudo kubeadm reset
$ sudo systemctl list-units | grep -E 'kubernetes|kubelet'
$ docker ps
```

Remember that Kubernetes required swap to be disabled and it may need to be re-enabled if you are planning to use the Raspberry Pi for other uses. There is some debate about whether swap on a Pi is actually a good idea in general, however, since the SD Card is rather slow and doesn't handle repeated reads and writes well. If you have a USB hard drive this could be a good solution to increasing swap. The memory and cpuset cgroups were also enabled for Kubernetes by modifying the `/boot/cmdline.txt` kernel options file. Leaving these enabled will not cause problems for other uses but they can be easily turned off by removing the lines that were added in the System configuration section. Here are the required commands to re-enable swap.

```
$ sudo dphys-swapfile install && \
  sudo dphys-swapfile swapon && \
  sudo update-rc.d dphys-swapfile defaults
```

Swap will be enabled immediately and the changes will persist after reboot.

8.2.8 Files

- kubernetes/526/bin/adm_kub_config.yaml
- kubernetes/526/bin/config_kub.sh
- kubernetes/526/bin/copy_dk_kub_install_script_to_nodes.sh

- kubernetes/526/bin/docker_kubernites_install.sh

- docker_setup.sh
- README.md
- dhcp_setup.sh
- join
- kube_head_setup.sh
- kube_worker_setup.sh
- kubeadm_conf.yaml

- opt_setup.sh

8.2.9 References

- https://gist.github.com/alexellis/fdbc90de7691a1b9edb545c17da2d975
- https://cloud.google.com/solutions/real-time/kubernetes-redis-bigquery
- https://kubecloud.io/setup-a-kubernetes-1-9-0-raspberry-pi-cluster-on-raspbian-using-kubeadm-f8b3b85bc2d1
- https://www.hanselman.com/blog/HowToBuildAKubernetesClusterWithARMRaspberryPiThenRunNETCoreOnOpenFaas.aspx
- https://marcussmallman.io/2018/02/18/diy-rasberry-pi-kubernetes-cluster/
- https://blog.hypriot.com/post/setup-kubernetes-raspberry-pi-cluster/
- https://blog.sicara.com/build-own-cloud-kubernetes-raspberry-pi-9e5a98741b49

8.3 Raspberry PI Hadoop Cluster 🅾️ ☁️

🅾️

8.3.1 Todo

- [ ] all the simple setup with sd cards, ssh, keys, and so on should be moved to the NOW cluster section. This way we can require simply a NOW and start without duplication on the real kubernetes install.
- [ ] so before you can work on the section you need to make sure the NOW section is up to date.

8.3.2 Links

See another effort documented at:

- Benchmarking Hadoop and Spark on Mutiple Platforms http://cyberaide.org/papers/vonLaszewski-cloud-vol-9.pdf#page=27&zoom=100,0,96

8.4 Raspberry PI Spark Cluster 🅾️ ☁️

⭕

We provide step-by-step instructions on installing a Spark cluster on a cluster of Raspberry Pi's.

8.4.1 Todo

- [ ] all the simple setup with sd cards, ssh, keys, and so on should be moved to the NOW cluster section. This way we can require simply a NOW and start without duplication on the real kubernetes install.
- [ ] so before you can work on the section you need to make sure the NOW section is up to date.
- [ ] The section contains some issues
- [ ] A per node setup is used instead of a scripted setup
- [ ] Some text in the later part is unclear

8.4.2 Prerequisites ⭕

⭕

In Network of Pis section we explained how to set up a network of Pis. Here we assume that we start from such a network. We assume that you have on all the Raspberry Pi nodes the following software and configuration files installed:

SSH:

> Configure passwordless SSH key based authentication: All the public keys of the nodes must be added to all the nodes' authorized keys files. See our SSH information in section on how to do this.

Java:

> To install Java on the PI please use the following commands

```
pi$ sudo add-apt-repository ppa:webupd8team/java
pi$ sudo apt-get update
pi$ sudo apt-get install oracle-java8-installer
```

```
pi$ echo "export JAVA_HOME=/usr/lib/jvm/java-8-oracle" >> ~\.bashrc
pi$ source ~/.bashrc
```

Scala:

To install Scala you can use the following commands

```
pi$ sudo apt-get install scala
```

Hostnames:

⚠ *this can b automatically done and > needs to be documented in the Pi NOW section. THe way we do this > is set up key authentication foirts and than use scp or better > cloudmesh to copy it. We need to describe that provess in more > detail in the NOW section.*

IN our example we assume we use 3 hosts. The hosts will be added to the file `/etc/hosts`. Please use IP numbers for your network configuration. For us this is

```
192.168.10.2        pi-master
192.168.10.3        pi-slave0
192.168.10.4        pi-slave1
```

8.4.3 Download

Download the most recent version of Spark from the Apache website (we use here version 2.3.2).

⚠ *if a newer version is available, your task will be to use the newer version and create a new updated set of instructions. At this time the newest version is 2.3.2. Please double check.*

- [Apache Spark Latest Download]

After the download is completed run the command

```
pi$ wget https://www.apache.org/dyn/closer.lua/spark/spark-2.3.2/spark-2.3.2-bin-hadoop2.7.tgz
```

8.4.4 Installation

Create the folder for storing spark install files

```
pi$ sudo mkdir -p /opt/spark-2.3.0
```

Unzip the tar fle into destination folder

```
pi$ bash tar -xzf spark-2.3.2-bin-hadoop2.7.tgz -C /opt/spark-2.3.0 --strip-components=1
```

Update the `PATH` variable

```
pi$ echo "export SPARK_HOME=/opt/spark-2.3.2" >> ~\.bashrc
pi$ echo "export PATH=$PATH:$SPARK_HOME/bin" >> ~\.bashrc
pi# echo "export PATH=$PATH:$SPARK_HOME/sbin" >> ~\.bashrc
pi$ source ~/.bashrc
```

Copy the template from `spark-env.sh.template` to `spark-env.sh`

```
pi$ cp $SPARK_HOME/conf/spark-env.sh.template $SPARK_HOME/conf/spark-env.sh
```

Edit `spark-env.sh` file to change the configurations and add the following configurations in to the `spark-env.sh` file:

```
#!/usr/bin/env bash
SPARK_WORKER_MEMORY = 512m
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export SPARK_WORKER_CORES=1
```

Now edit slaves file on master node

```
cp $SPARK_HOME/conf/slaves.template $SPARK_HOME/conf/slaves
vi $SPARK_HOME/conf/slaves
```

And add the following content. (Change this according to the number of slaves you configure).

```
pi-slave0
pi-slave1
```

Note: The previously mentioned slaves are of the same names of the hostnames specified in the /etc/hosts file in the prerequisites section.

8.4.5 Run Spark

Now that you've followed the installation steps completely you can start the Spark cluster. Since the SSH server configurations have been done, you only need to run the following command on the master and it will automatically start the Spark workers on the slaves we've mentioned in the `/etc/hosts` file and setup the whole cluster.

Run this on the master:

```
$SPARK_HOME/sbin/start-all.sh
```

You can now view the Spark cluster information in the Spark Master UI:

```
http://master_IP:8080
```

You can run the following command on the master to stop the cluster.

```
$SPARK_HOME/sbin/stop-all.sh
```

> ⚠ *see also the file `pi-spark-orig.md` as it still contains useful information such as the output when running pi spark.*

8.4.6 Towards a cm4 command for pi-spar instalation

We suggest that a command be developed as part of cm4 taht installs sparck on a number of machines. THis needs t be done in phases as not to duplicated work

- Phase 1: add keys: here you add keys to the hosts in the network using python hostlists as already demonstrated in cm4

  `cm4 deploy keys --hosts HOSTNAMES [--key FILEANME]` uses the publick key

- Phase 2: deploy spark: This deployes spark on the different hosts

  `cm4 deploy spark --hosts HOSTNAMES`

- Phase 3: test spark: This runs a simple test to see if things work

```
cm4 test spark --hosts HOSTNAMES
```

For the implementation either ansible could be used, or simply a queue in python executing the various commands or shell scripts.

8.4.7 REFERNCES

See another effort documented at:

- Edge Computing and Big Data Processing using Raspberry Pi http://cyberaide.org/papers/vonLaszewski-cloud-vol-9.pdf#page=104&zoom=100,0,96

- Benchmarking Hadoop and Spark on Mutiple Platforms http://cyberaide.org/papers/vonLaszewski-cloud-vol-9.pdf#page=27&zoom=100,0,96

- We have an extensive section on how to use SSH keys. However others also pointed to this article It describes also how to disable the password for root.

9 DRAFT

9.1 Automated Headless Configuration of a Pi Cluster ⭕ ☁️

⭕

Our goal is perform the following configuration automatically:

- Enable ssh permanently (initial ssh access will be enabled when we burn the SD cards)
- Change the password
- Set up one of the Pis as a DHCP server

These actions are all done with two scripts. The first script, `configure-pi.sh`, runs on the computer used to set up the Pis. The second, `setup-pi.sh`, enables ssh, changes the password for the pi user, and configures the master node as a DHCP server. Determination of whether the node is the master or a worker is done with the `-m` flag.

9.1.1 Prerequisites

- Assemble a Pi Cluster
- Burn SD cards with names changed and ssh enabled
- Install `expect` on computer running `configure-pi.sh`. On a Mac, this is done with `brew install expect`. On Unix, use `apt-get install expect` or `yum install expect`. More information on `expect` can be found here.

9.1.2 Setting up DHCP

> ⚠ *The information presented here insecure!*

TODO: Find out if cluster should be set up with or without internet enabled. Tutorial for getting Pi on IU Secure here.

TODO: New version of isc-dhcp-server. Find out if INTERFACES in /etc/default/isc-dhcp-server should be V4, V6 or both.

TODO: Switch is eth8 when internet sharing is off, names show up, and IP addresses are on different subnet. When internet sharing is on, names do not show up. How should DHCP and network be setup? What is the use case? Listed next is output of `arp -a` with internet sharing. I turned it on so that installations via apt-get would work:

```
(2.7.16) BKS-MBP:project-code bsobolik$ arp -a
hello (192.168.1.1) at 58:ef:68:a9:51:4e on en0 ifscope [ethernet]
bertoltksiphone.hsd1.in.comcast.net (192.168.1.126) at e4:9a:79:7f:19:55 on en0 ifscope [ethernet]
? (192.168.1.255) at ff:ff:ff:ff:ff:ff on en0 ifscope [ethernet]
? (192.168.2.7) at b8:27:eb:0:c3:55 on bridge100 ifscope [bridge]
? (192.168.2.8) at b8:27:eb:d1:21:33 on bridge100 ifscope [bridge]
? (192.168.2.255) at ff:ff:ff:ff:ff:ff on bridge100 ifscope [bridge]
? (224.0.0.251) at 1:0:5e:0:0:fb on en0 ifscope permanent [ethernet]
? (239.255.255.250) at 1:0:5e:7f:ff:fa on en0 ifscope permanent [ethernet]
broadcasthost (255.255.255.255) at ff:ff:ff:ff:ff:ff on en0 ifscope [ethernet]
```

When internet sharing is off I get:

```
(2.7.16) BKS-MBP:~ bsobolik$ arp -a
raspberrypi.local (169.254.177.219) at b8:27:eb:0:c3:55 on en8 [ethernet]
? (169.254.255.255) at (incomplete) on en0 [ethernet]
hello (192.168.1.1) at 58:ef:68:a9:51:4e on en0 ifscope [ethernet]
? (192.168.1.255) at ff:ff:ff:ff:ff:ff on en0 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en0 ifscope permanent [ethernet]
broadcasthost (255.255.255.255) at ff:ff:ff:ff:ff:ff on en0 ifscope [ethernet]
```

☁️ ## Preparing the SD card ⭕

⭕

TODO: We should at this time assume we have an OS.

Download the latest Raspbian Jessie Lite image from

```
https://www.raspberrypi.org/downloads/raspbian/
```

Please note that Raspbian Jessie Lite image contains the only the bare minimum amount of packages.

9.1.3 Download Etcher ⭕

⭕

- https://etcher.io/

Now follow the instructions in Etcher to flash Raspbian image on the SD card. Before ejecting the SD card do the following.

9.1.4 Enable SSH on the SD Card ⭕

⭕

To prevent Raspberry Pis from being hacked the RPi foundation have now disabled SSH on the default image. So, create a text file in /boot/ called ssh - it can be empty file or you can type anything you want inside it.

Please note that you have renamed the ssh.txt to ssh i.e. without extension.

Now insert the SD card, networking and power etc.

9.1.5 Starting Pi ⭕

⭕

Once you boot up the Raspberry Pi, Connect using SSH

```
$ ssh pi@raspberrypi.local
```

The password is raspberry.

For security reasons, please change the default password of the user pi using the passwd command.

Note: If you want to change the hostname of the Pi, Use an editor and change the hostname Raspberry Pi in:

```
* /etc/hosts
* /etc/hostname
```

9.2 Raspberry Pi Robot Car with Face Recognition and Identification ☁️

Mani Kumar Kagita
mkagita@iu.edu

Indiana University - Bloomington
hid: SP18-711

Keywords: I523, HID319, SP18-711, Robot Car, Face Recognition

9.2.1 Introduction

A Computer Vision application which has always encouraged people, concern about the capability and capacity of robots and computers to determine, detect, recognize and interact with human beings **???**. We will prevail the advantage of cheaper tools that are available in the market for computing and detecting a human face from the image, recognizing the face using hardware like Raspberry Pi and a video camera that is dedicated to Raspberry Pi. Simple and open source software like OpenCV is used to detect a human face from the video that is being captured and the image will be sent to Kairos face recognition software which allows a high-level approach to this process.

In this fastest information era, every information is travelled in a split of a second. There is much more need for accurate and fastest methods for identifying, recognizing and authentication of humans. In the present world, face recognition had become most important and crucial form of human identification methods. As per Literature survey statistics in face recognition, the two trends to receive significant attention for the past several years are; the first is the law enforcement applications and also a wide range of commercial techniques, and the second is exponential booming of applications and feasible technologies after 30 years of research [1].

The aim is achieved by a possibility to locate human beings or identities like faces from the live video capture and within the context of the picture. Most advanced human detection applications have this functionality already available. When the picture is captured and loaded into the system, it will scan the picture and will look for human faces in it. The current implementation is to detect face and register them with a name. If the face is detected and not recognized, Robot car will ask to register the detected face with a name. If the human is already registered in Kairos, then once the face is detected, Robot car will greet the human with the associated name.This whole process determines the Face detection and Face recognition techniques using Raspberry Pi and Robot car.

Facial biometric data is to be computed first in creating a complete recognition system. This biometric data is then compared with the face database and to associate with the human identity. The difference between a human and machine is, a human can easily and quickly identify characteristics of a human face but then can only save few hundreds of faces. Whereas a machine or computers prevails at storing and mapping human characteristics and meta data. In the current generation, facial recognition software can identify a human face within millions of images from the database in seconds. Humans tend to forget human faces as time pass by. Machines store them forever. Most of the Law firms across the world follow the process and spend huge money on the development of these facial recognition systems that can easily identify criminals in real-time. A well-known example is studying human faces in airports and bus stations.

The design of the Robot car integrated with Face recognition system will navigate through dangerous or natural disaster locations where humans unable to enter. Robot car while avoiding obstacles on its way, will continuously monitor for human faces who got stuck or in danger and will recognize the faces based on the user database. Once the human face is recognized, it will intimate to corresponding authorities about the human and will help in guiding assistance.

9.2.2 Face Detection

Face Detection is a technique referred to computer vision technology which is able to identify human faces within digital images [2]. Face detection applications work using algorithms and machine learning formulas for detecting human faces in the visual images. Identifying only human faces from these images which can contain landscapes, houses, animals is called Face Detection technique.

Face Detection is termed to only identify if there are any humans present in the image or a video. It lacks the inability to recognize which human face is present. Common widely used face detection techniques are in auto-focus of a digital camera. During auto-focus, camera lens will look for human faces in the range and identify them to have focus in that particular area. Face Detection techniques will be widely used in counting how many numbers of visitors attending a particular event.

9.2.2.1 How Face Detection Works

While Face Detection process is somewhat complex, the algorithms will start off by searching for human eyes at first. Eyes usually represent a valley region and its the easiest feature in human face to detect. Once the eyes are detected, then the algorithms will look for rest of the characteristics of a human face such as iris, nose, mouth, eyebrows, and nostrils. Face detection algorithm then summarizes the data and shows that it has successfully detected a human face from the facial region. Additional tests can be conducted by the algorithm to make sure and validate if the human face is detected [3].

9.2.3 Face Recognition

Like most of the biometric solutions, face recognition technology will be used for identification and authentication purposes by measuring and matching the unique facial characteristics of a human face. Using a digital camera connected to the Raspberry Pi, once the face is detected, face recognition software will quantify the characteristics of face and then will match with the stored images in the database. Once the match is positive, then the corresponding name will be displayed as output [4].

Facial biometrics can be integrated with any system having a camera. Border control agencies use face recognition to verify identities of the travellers and can separate them from the trespassers. Government Law agencies replace all the security cameras around the world with biometric applications to scan faces in CCTV footage and to identify persons of interest in the field. Face recognition has become one of the fastest and human un-intervention techniques to find out the identity of a particular human [4].

For the past few years, Face recognition has become one of the most commonly used biometric authentication techniques. It mainly deals with the pattern recognition and analysing the images. Two main tasks of face recognition are: Face Verification and Face Identification. Face Verification is comparing a human face in an image with a template image and recognizing the correct patterns. Face Identification is comparing human face in an image with multiple images in the database. Face recognition techniques have more advantages than any other biometrics. With well-sophisticated algorithms and coding, face recognition has a high recognition rate or high identification rate of more than 90% [1]. Figure 30 shows the various levels of face recognition process [5].

Figure 30: Block Diagram of a Face Recognition System

9.2.3.1 Face Recognition and Big Data Analysis

Face Recognition and Big Data are two distinct technologies which are having hardly anything in common. But when they both are put together, a technology drift takes place in terms of biometric authentication. Storing a massive unique characteristics and libraries of human faces, algorithms are used to run on these characteristics to recognize the human face accurately. Using big data, a real-time analysis can be done which identifies faces and applies the rules as they are happening.

Robot car is designed to collect all the facial features that are encountered on its path and store them in the cloud. When the face is detected by the camera, it sends the picture to the cloud and facial recognition software will compare with huge database of faces that are located in the cloud. Once the face features match, robot car will respond with the unique name that is set for that human face.

9.2.4 Software and Hardware Specifications

OpenCV is to be installed in Raspberry Pi to detect human faces within the captured images. Kairos face recognition software is used to recognize a human face and identify with the corresponding name.

9.2.4.1 Software Used

9.2.4.1.0.1 Raspbian OS

This is the recommended OS for Raspberry Pi 3. Raspbian OS is Debian based operating system. It can be installed from NOOBS installer. Raspbian comes with various pre-installed software such as Python, Sonic Pi, Java, Mathematica for programming and education.

9.2.4.1.0.2 Putty

PuTTY is an SSH and telnet client for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers. Here we are using putty for accessing our Raspberry Pi remotely.

9.2.4.1.0.3 OpenCV

Open Source Computer Vision Library (OpenCV) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects and extract 3D models of objects [6].

9.2.4.1.0.4 Python 2 IDE

Python 2.7.x version Integrated Development Environment is used to compile python program in Raspberry Pi. IDE is a text editor plus terminal combination which is used to work on large projects with complex code bases.

9.2.4.1.0.5 Kairos Face Recognition Software

Kairos is an artificial intelligence company specializing in face recognition. Through computer vision and machine learning, Kairos can recognize faces in videos,photos, and the real-world. A captured image is sent to Kairos using an API call and then Kairos will search with the face database. If it matches then will reply with the human name.

- Identity
- Emotions
- Demographics

Kairos navigates the complexities of face analysis technology.

9.2.4.2 Hardware Used

9.2.4.2.0.1 Raspberry Pi 3

Raspberry Pi 3 is the latest version of Raspberry Pi. Unless previous versions, this have an inbuilt Bluetooth platform and a wifi support module. There are total 40 pins in RPI3. Of the 40 pins, 26 are GPIO pins and the others are power or ground pins (plus two ID EEPROM pins). There are 4 USB ports, 1 Ethernet slot, one HDMI port, 1 audio output port and 1 micro USB port. And also we have one micro SD card slot wherein we have to install the recommended operating system on the micro SD card. There are two ways to interact with your Raspberry Pi. Either you can interact directly through HDMI port by connecting HDMI to VGA cable, and keyboard and mouse or else you can interact from any other system through Secure Shell (SSH) [7].

9.2.4.2.0.2 Raspberry Pi Camera

The Raspberry Pi camera module can be used to take high-definition video, as well as stills photographs. It's easy to use for beginners but has plenty to offer advanced users if you're looking to expand your knowledge. There are lots of examples online of people using it for time-lapse, slow-motion and other video cleverness. You can also use the libraries we bundle with the camera to create effects.

9.2.4.2.0.3 Robot Car Chassis Kit

The Mechanical design of the Robot car includes hardware such as motor and wheel placement and body set-up. Robot car uses two gear-motors attached to wheels and one free-wheel for having various movements like forward, backward, left and right. Free-wheel ball is placed at the rear side of the robot which helps for 360 degrees free movement [8]. L298N DC Stepper Motor Drive controller is used to control the speed and direction of the two gear motor wheels. Ultrasonic sensors are placed on the front side of the robot which is capable to detect the objects on its path [9].

9.2.5 System Architecture

System Architecture consists of following blocks:

- Raspberry Pi
- Raspberry Pi Camera Module

- L298N Dual H-Bridge Stepper Motor Controller
- DC power supply 12v and 5v
- Robot Car chassis kit
- HC-SR04 Ultrasonic Sensor
- SG90 Servo Motor.
- Wires, Breadboard, Small PCB.

9.2.6 SETUP

9.2.6.1 Connect Raspberry Pi

This section includes connectivity of Raspberry Pi to wifi.

- Download Raspbian operating system to an SD card with a minimum capacity of 8GB.
- Plugin USB power cable, keyboard, mouse and monitor cables to Raspberry Pi.
- Insert the SD card with Raspbian OS into Pi and boot the system. Once the Pi is booted up, a window will appear with the Raspbian operating system. Click on Raspbian and Install.
- Once the install process has completed, the Raspberry Pi configuration menu (raspi-config) will load. Here set the time and date for your region.
- Enable wifi located at the upper right corner of the desktop and connect to wifi sid.

9.2.6.2 Connect Raspberry Pi Camera Module

Before setting the camera configurations, first connect the camera to Raspberry Pi. The cable slots into the connector situated between the Ethernet and HDMI ports, with the silver connectors facing the HDMI port. Once the connection is completed, boot up the Raspberry Pi and run the following commands to install various supporting libraries. Skip first two steps if Python is already installed on Raspberry Pi.

```
pi$ sudo apt-get install python-pip
pi$ sudo apt-get install python-dev
pi$ sudo pip install picamera
pi$ sudo pip install rpio
```

Once the libraries are installed, follow the next steps to check if camera is installed in Raspberry Pi.

```
pi$ sudo raspi-config
```

If the camera option is not listed in the options, run the following commands to update Raspberry Pi.

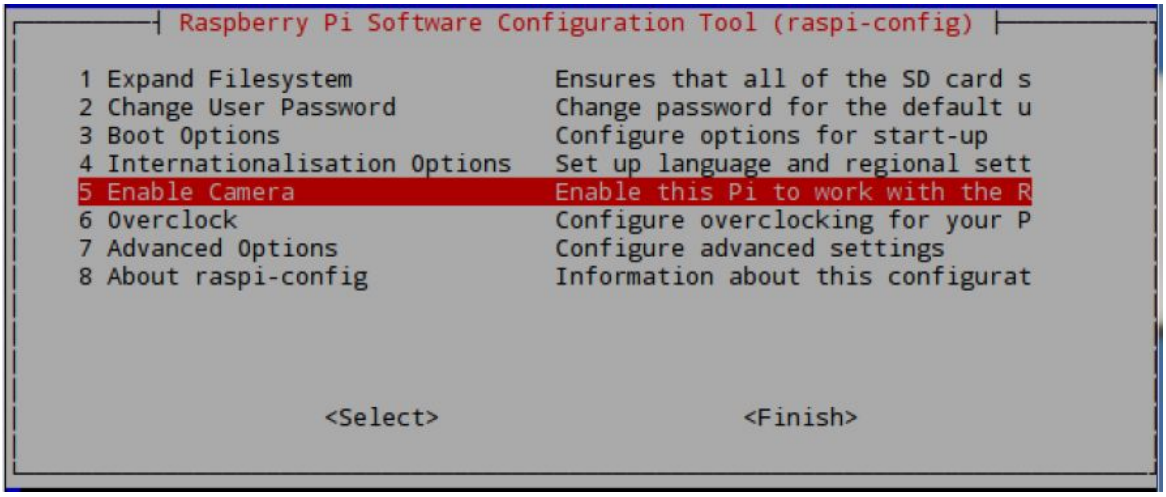
```
pi$ sudo apt-get update
pi$ sudo apt-get upgrade
```

9.2.6.2.0.1 Enable Camera

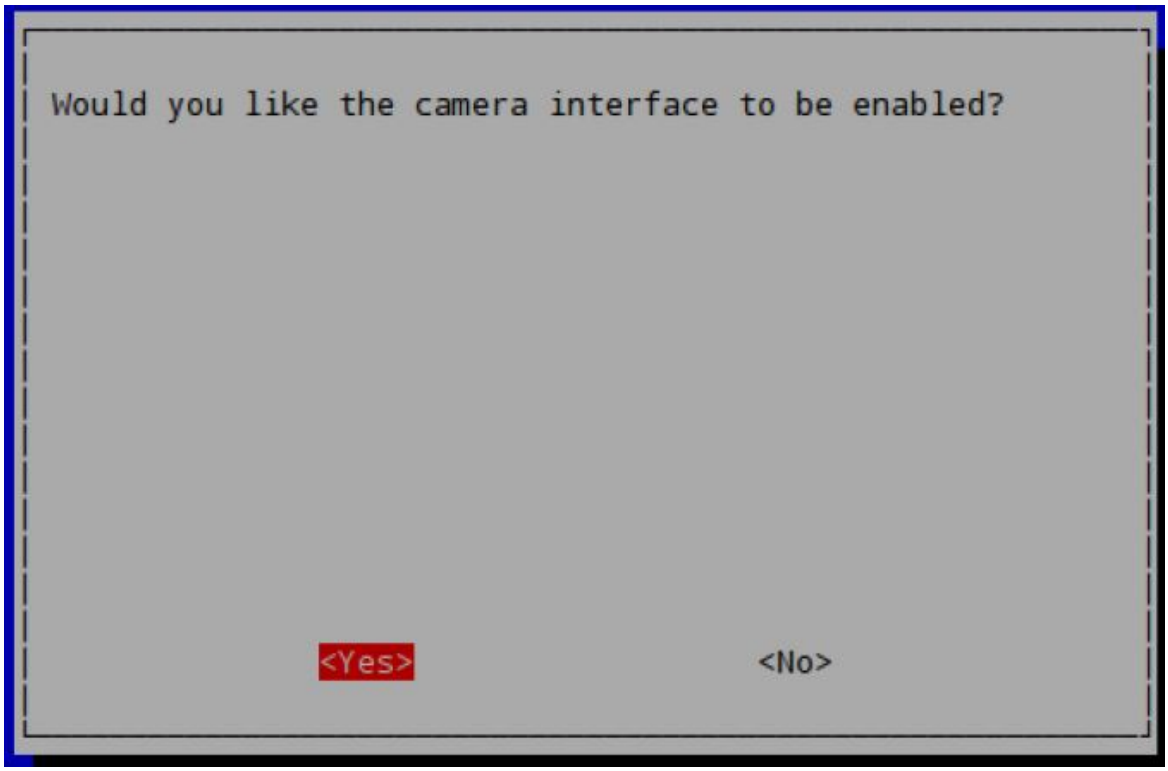
For face detection, PiCamera should be enabled from Raspberry Pi. The following list of figures shows the detailed steps on how to enable PiCamera [10].

As shown in Figure [F:raspi], execute the configuration command from terminal. From the listed options, select "Enable Camera" as shown in Figure [F:selcamera]. Click on "Yes" option to enable the camera interface as shown in Figure [F:enbcamera].

Enable Camera



Enable Camera



Enable Camera

9.2.6.3 Install OpenCV and Required Libraries

OpenCV computer vision library is used to for face detection from the live video streaming. Execute the following commands to install OpenCV dependencies on the Raspberry Pi.

```
pi$ sudo apt-get install build-essential
pi$ sudo cmake pkg-config python-dev libgtk2.0-dev \
        libgtk2.0 zlib1g-dev libpng-dev \
        libjpeg-dev libtiff-dev libjasper-dev \
        libavcodec-dev swig unzip
```

Select yes for all options and wait for the libraries and dependencies to be installed.

Download opencv-2.4.9 zip file to Raspberry Pi. Change to the corresponding directory and execute the following commands.

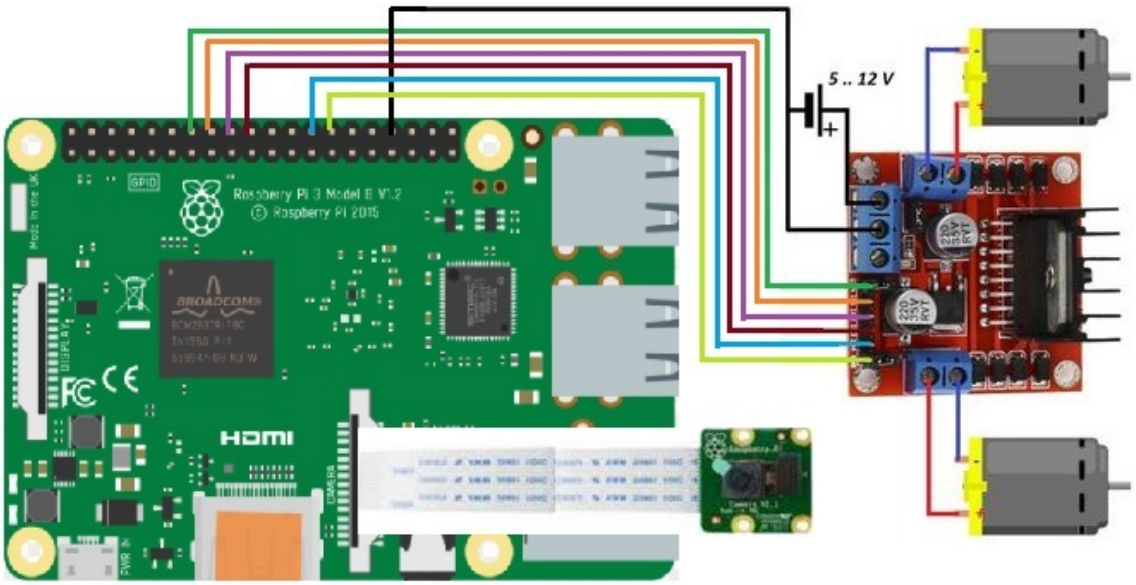
```
pi$ cd opencv-2.4.9
pi$ sudo apt-get install build-essential cmake \
        pkg-config
pi$ sudo apt-get install libjpeg-dev libtiff5-dev \
        libjasper-dev libpng12-dev
pi$ sudo apt-get install python-dev python-numpy \
```

```
        libtbb2 libtbb-dev libjpeg-dev \
        libpng-dev libtiff-dev libjasper-dev \
        libdc1394-22-dev
pi$ sudo apt-get install python-opencv
pi$ sudo apt-get install python-matplotlib
```

After executing the commands the latest version of OpenCV is now installed in Raspberry Pi. Time taken to install OpenCV is about 15 minutes.

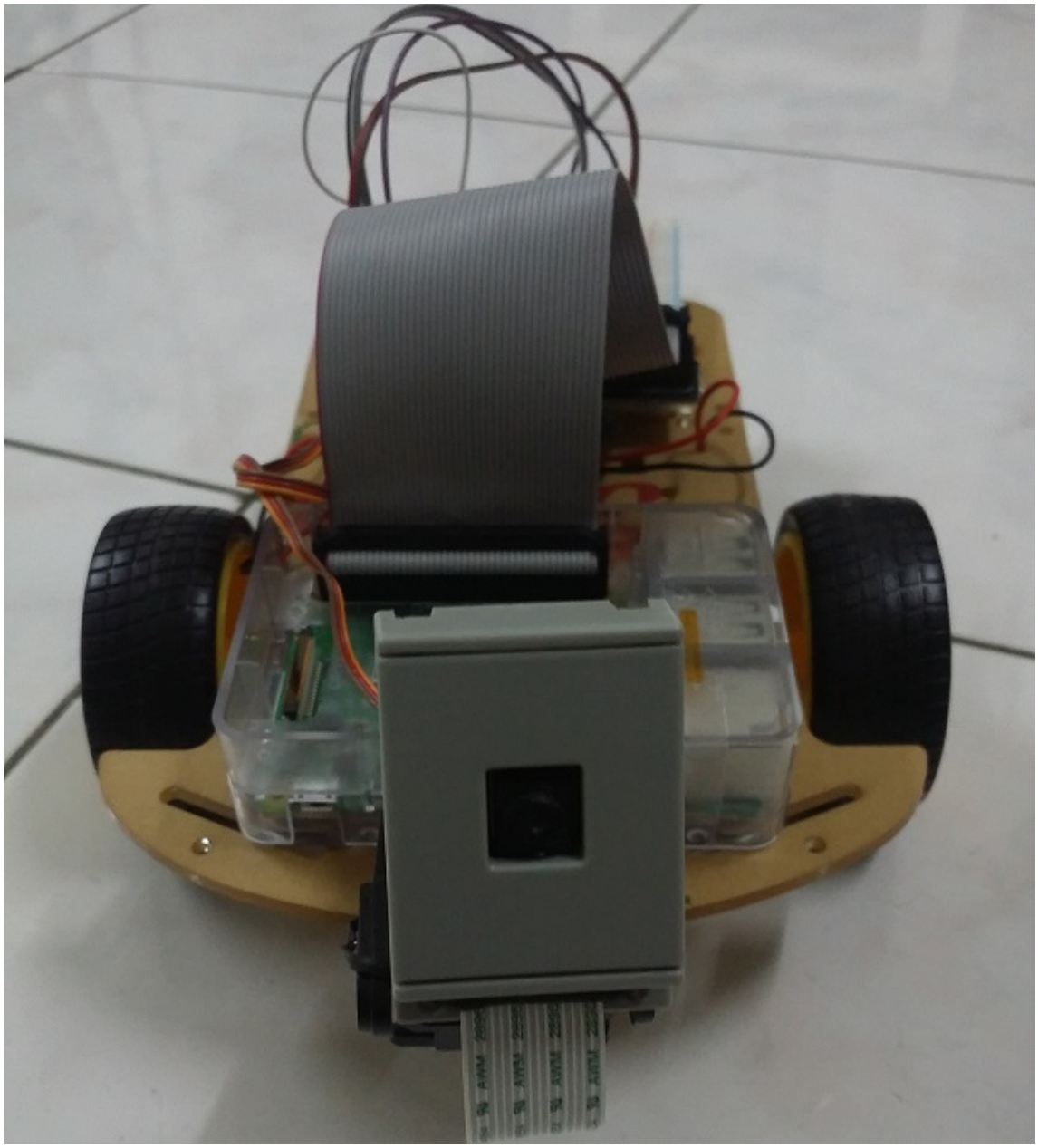
9.2.6.4 Integration of Raspberry Pi with Robot Car

Raspberry Pi connected with PiCamera is integrated with Robot car to navigate using a web server. During the navigation, robot car will look for human faces using PiCamera and then detects the face. Once the face is detected, python program will call Kairos face detection software to identify the person and greet with the name. If the human face is unidentified then robot car will ask human to register their name. The diagram in Figure [F:circuit] shows the circuit connections of Raspberry Pi to stepper motor controller and DC motors of a robot car.

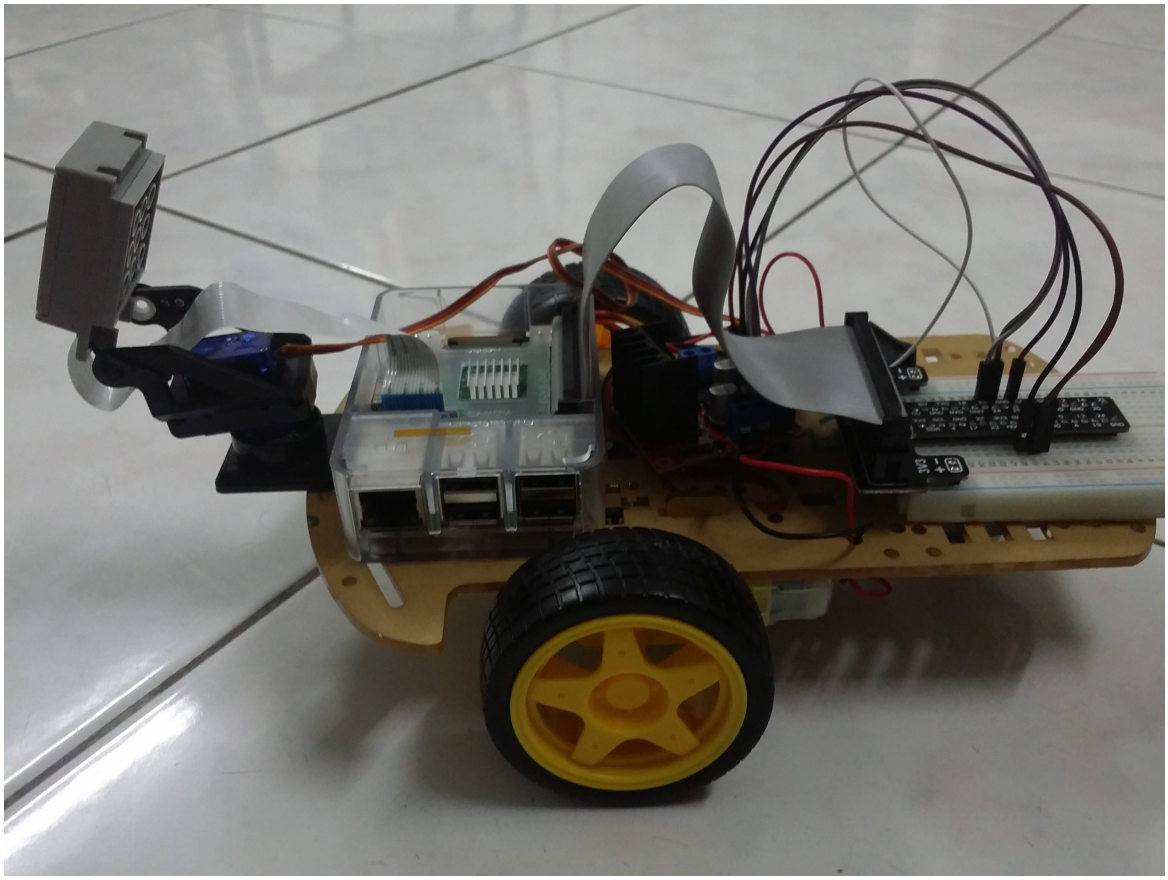


Raspberry Pi Robot Car Integration

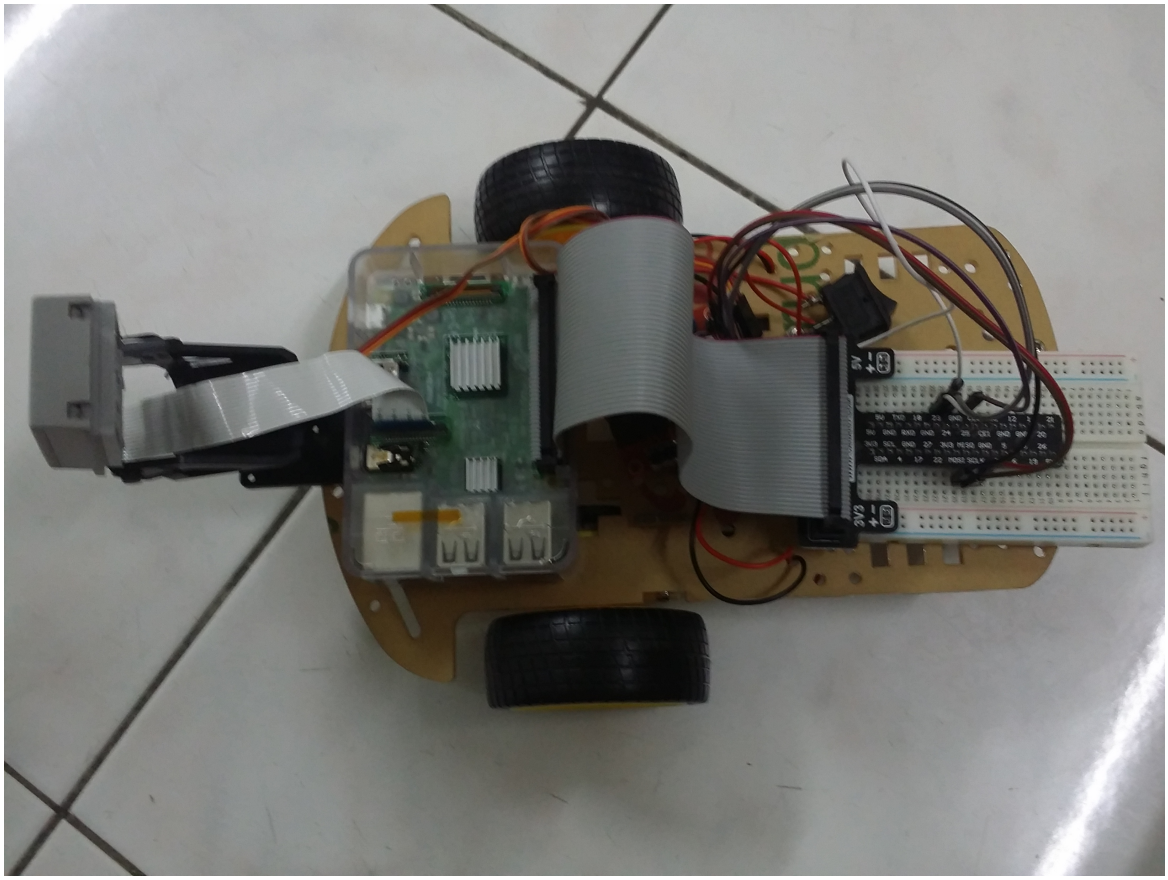
Figures [F:robotfront], [F:robotside], [F:robottop] represents corresponding front view, sideview and topview of the robot car connections.



Top view of Robot Car



Top view of Robot Car



Top view of Robot Car

In the Table [T:pinlayout], shows the connectivity of Raspberry Pi GPIO pins to L298N stepper motor controller.

9.2.6.5 Actuator Raspberry Pi GPIO Pin L298N Pin

Motor1A GPIO23 IN1 Motor1B GPIO24 IN2 Motor1Enable GPIO25 ENA Motor2A GPIO9 IN3 Motor2B GPIO10 IN4 Motor2Enable GPIO11 ENB

: Pin connections of Raspberry Pi to stepper motor controller

9.2.6.6 Kairos Face Recognition Setup

Kairos Face Recognition system has a free developer account which is used to identify the human name from the images. Once registered a human name with an image, the code will call Kairos API with a newly detected human face and will look for the registered name. Kairos will do a quick look-up in the human

database from the registered account and if it matches, will send the name of the human back to the code.

Setup as follows:

- Register with Kairos using url “https:/www.kairos.com” as a free developer account
- Login with registered username and password
- Create an appname
- An app id and a key will be generated. Save this for future use.

- Enroll a sample user and a gallery name with the user image using following POST request.

```
POST /enroll HTTP/1.1
Content-Type: application/json
app_id: your-app-id
app_key: your-app-key
{
"image":" http://media.kairos.com/user.jpg ",
"subject_id":"User",
"gallery_name":"MyGallery"
}
```

With the completed steps, Kairos face recognition application will be created and ready for face recognition from the images.

9.2.7 Code Explanation

9.2.7.1 Face Detection

Before configuring face detection for the robot car, related libraries including PiCamera and PiRGBArray libraries for camera to operate should be imported in the code. These libraries will help to capture video and images from the PiCamera.

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import sys
import imutils
from fractions import Fraction
import base64
import requests
import json
import random
import os
```

Haar Feature-based cascade classifier is an effective face or object detection method to capture the frontal features of the face [5]. This tool will help to continuous monitoring of any human face to detect. Once detected a human face, the output values will provide as Human Face Detected from the capturing video.

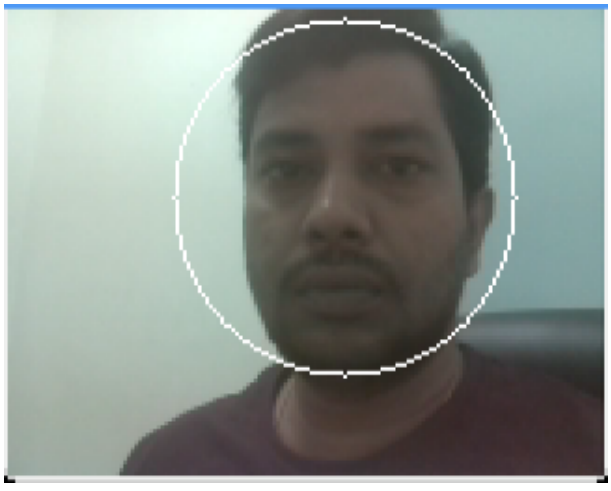
```
## Get user supplied values
cascPath = './haarcascade_frontalface_default.xml'

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

Camera settings need to be updated, our suggested changes are shown in the code next. The captured image is to be sent to Kairos for face recognition and so we will set the resolution to a lower level. This will help to send the image faster over the network without any delay.

```
## initialize the camera
#camera capture
camera = PiCamera()
camera.resolution = (160, 120)
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(160, 120))
```

The code shown next represents PiCamera continuously monitor for human faces detected from the grayscale video capture. Once the human face is detected, espeak function in Raspberry Pi will send the voice to a connected speaker and will output as “Human face detected”. This detected image is then saved as “User-Image.jpg” which is then will be sent to Kairos during Face recognition.

Images in Figures [F:frontview], [F:sideview1], [F:sideview2] represents face detection of front and side views within the circle using OpenCV.



Side view 2 Face detection

Side view 2 Face detection

Side view 2 Face detection

```
## allow the camera to warm-up
time.sleep(0.1)
lastTime = time.time()*1000.0
## capture frames from the camera
for frame in camera.capture_continuous(rawCapture, \
    format="bgr", use_video_port=True):
    ## grab the raw NumPy array representing the image,
    ## then initialize the timestamp and
    ## occupied/unoccupied text
    image = frame.array
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    ## Detect faces in the image
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags = cv2.cv.CV_HAAR_SCALE_IMAGE
    )
    print time.time()*1000.0-lastTime," \
    Found {0} faces!".format(len(faces))
    lastTime = time.time()*1000.0

    ## Draw a circle around the faces
    for (x, y, w, h) in faces:
        cv2.circle(image, (x+w/2, y+h/2), \
        int((w+h)/3), (255, 255, 255), 1)
    ## show the frame
```

```
    cv2.imshow("Frame", image)
    key = cv2.waitKey(1) & 0xFF
    if len(faces) == 1:
        print("Taking image...")
    camera.capture("foo.jpg")
    os.system('espeak "Human face detected"')
    inputImage= "./foo.jpg"
    del camera
    break
    ## clear the stream in preparation for the
    #next frame
    rawCapture.truncate(0)

    ## if the `q` key was pressed, break from
    #the loop
    if key == ord("q"):
        del camera
        exit()
```

9.2.7.2 Face Recognition

For the Face Recognition, we use Kairos to detect the facial characteristics. A JSON config file is to be placed in the same folder as of the code with Kairos API app id and key value. When the human face is detected, the code will generate an API call to Kairos software along with the gallery name, API app id and key values. Image when sending to Kairos, it will be base64 encrypted and will send over the network for security purpose. This encrypted image will then be decrypted at Kairos platform.

```
KAIROS = "api.kairos"
KairosGallery = 'MyFace'
KairosConfig = './kairos_config.json'

def trainKairos(image, name):
    global KairosGallery
    headers = {
        'app_id': 'your-app-idd39fc1b1',
        'app_key': 'your-app-key'
    }
    data = {
        'image': base64.b64encode(image),
        'gallery_name': KairosGallery,
        'subject_id': name
    }
    r = requests.post('http://api.kairos.com/enroll', \
        headers=headers, data=json.dumps(data))
    print(r.text)
    return(None)

class Recognize():
    def __init__(self, API, config_file):
        self.api = API
        self.config = config_file

    #def recognize(self, image_path):
    ##    return self.__recognizeKairos(image_path)

    def recognizeKairos(self, image):
        with open(image, "rb") as image_file:
            encoded_string = base64.b64encode\
            (image_file.read())
        with open(self.config, "rb") as config_file:
            config = json.loads \
            (config_file.read())
        data = {
            "image": encoded_string,
            "gallery_name": config["gallery_name"]
        }
```

```
        headers = {
            "Content-Type": "application/json",
            "app_id": config["app_id"],
            "app_key": config["app_key"]
        }
```

The output from Kairos software is in JSON format. The output is then segregated as per the key-value pairs and then saved into local variables. When the image is recognized, a success transaction message will be obtained from Kairos along with subject id and face id.

```
try:
    r = requests.post("https://api.kairos.com/recognize", \
                  headers=headers, data=json.dumps(data))
    data = r.json()
    print data
    ## print json.dumps(data, indent=4)
    faces = []
    if "images" in data:
        for obj in data["images"]:
            if obj["transaction"]["status"] == \
            "success":
                face_obj = {}
                face_obj["person"] = \
                obj["transaction"]["subject_id"]\
                .decode("utf_8")
                #face_obj["faceid"] = \
                obj["candidates"][0]["face_id"]\
                .decode("utf_8")
                face_obj["confidence"] = \
                obj["transaction"]["confidence"]
                faces.append(face_obj)
            elif obj["transaction"]["status"] == \
            "failure":
                face_obj = {}
                face_obj["person"] = "unidentified"
                face_obj["confidence"] = 0
                faces.append(face_obj)
            else:
                print "its in last loop"
            return faces
 except requests.exceptions.RequestException as \
 exception:
       print exception
        return None
```

The output from Kairos face recognition software is to be read to understand if the person name is identified or not. If it is identified then the person name will be listed according to the corresponding person in the image. If the human is not identified, then the code will suggest if the user wants to registered for face recognition. Once the user key in the name, Kairos API call is generated while sending newly registered name and the gallery name to that corresponding application id. Here the newly recognized user will be registered with the name and his image. When the user is recognized by the camera in next corresponding events, then Robot car will greet the user with his name.

```
if __name__ == "__main__":
    r = Recognize(KAIROS, "kairos_config.json")
    x = r.recognizeKairos(inputImage)

    #print x
    #print x["person"]
    #print x[0]["person"]
    string1 = x[0]["person"]
    #print string1
```

```
    os.system('espeak "Hello...""{}"'.format(string1))
    if x[0]["person"] == "unidentified":
        os.system('espeak "Please enter your \
                  name to Register"')
        nameToRegister = raw_input("Please enter \
                        your name to Register :")
        binaryData = open(inputImage, 'rb').read()
        print('Enrolling to Kairos')
        trainKairos(binaryData, nameToRegister)
        print "You are now Registered as :", \
        nameToRegister os.system('espeak \
        "Hello...""{}"'.format(nameToRegister))
        exit()
```

9.2.7.3 Robot Car Navigation

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)

#Connecting two wheel motors to Raspberry Pi GPIO
#Left Motor (Motor 1) connections
Motor1A = 16 #(GPIO 23 - Pin 16)
Motor1B = 18 #(GPIO 24 - Pin 18)
Motor1Enable = 22 #(GPIO 25 - Pin 22)

#Right Motor (Motor 2) Connecctions
Motor2A = 21 #(GPIO 9 - Pin 21)
Motor2B = 19 #(GPIO 10 - Pin 19)
Motor2Enable = 23 #(GPIO 11 - Pin 23)

#Ouptut of Morors to set as OUT
GPIO.setup(Motor1A,GPIO.OUT)
GPIO.setup(Motor1B,GPIO.OUT)
GPIO.setup(Motor1Enable,GPIO.OUT)
GPIO.setup(Motor2A,GPIO.OUT)
GPIO.setup(Motor2B,GPIO.OUT)
GPIO.setup(Motor2Enable,GPIO.OUT)

## Defining function for Robot car to move forward
def forward():
    GPIO.output(Motor1A,GPIO.HIGH)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1Enable,GPIO.HIGH)
    GPIO.output(Motor2A,GPIO.HIGH)
    GPIO.output(Motor2B,GPIO.LOW)
    GPIO.output(Motor2Enable,GPIO.HIGH)

    sleep(2)

## Defining function for Robot car to move backward
def backward():
    GPIO.output(Motor1A,GPIO.LOW)
    GPIO.output(Motor1B,GPIO.HIGH)
    GPIO.output(Motor1Enable,GPIO.HIGH)
    GPIO.output(Motor2A,GPIO.LOW)
    GPIO.output(Motor2B,GPIO.HIGH)
    GPIO.output(Motor2Enable,GPIO.HIGH)

    sleep(2)

## Defining function for Robot car to turn right
def turnRight():
    print("Going Right")
    GPIO.output(Motor1A,GPIO.HIGH)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1Enable,GPIO.HIGH)
    GPIO.output(Motor2A,GPIO.LOW)
    GPIO.output(Motor2B,GPIO.LOW)
    GPIO.output(Motor2Enable,GPIO.LOW)

    sleep(2)

## Defining function for Robot car to turn left
def turnLeft():
    print("Going Left")
```

```
    GPIO.output(Motor1A,GPIO.LOW)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1Enable,GPIO.LOW)
    GPIO.output(Motor2A,GPIO.HIGH)
    GPIO.output(Motor2B,GPIO.LOW)
    GPIO.output(Motor2Enable,GPIO.HIGH)

    sleep(2)

## Defining function for Robot car to stop
def stop():
    print("Stopping")
    GPIO.output(Motor1A,GPIO.LOW)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor1Enable,GPIO.LOW)
    GPIO.output(Motor2A,GPIO.LOW)
    GPIO.output(Motor2B,GPIO.LOW)
    GPIO.output(Motor2Enable,GPIO.LOW)
```

9.2.7.4 Controling Robot Car using webserver

```
from flask import Flask, render_template, \
request, redirect, url_for, make_response
import RPi.GPIO as GPIO
import motors

#set up GPIO
GPIO.setmode(GPIO.BOARD)

#set up flask server
app = Flask(__name__)

#when the root IP is selected, return index.html page
@app.route('/')
def index():

    return render_template('index.html')

#recieve which pin to change from the button press on \
#index.html
#each button returns a number that triggers a command in \
#this function
#
#Uses methods from motors.py to send commands to the GPIO \
## to operate the motors
@app.route('/<changepin>', methods=['POST'])
def reroute(changepin):

    changePin = int(changepin) #cast changepin to an int

    if changePin == 1:
        motors.turnLeft()
    elif changePin == 2:
        motors.forward()
    elif changePin == 3:
        motors.turnRight()
    elif changePin == 4:
        motors.backward()
    else:
        motors.stop()


    response = make_response(redirect(url_for('index')))
    return(response)

#set up the server in debug mode to the port 8000
app.run(debug=True, host='0.0.0.0', port=8000)
```

9.2.8 APPLICATIONS

There are lots of applications of face recognition. Face recognition is already being used to unlock phones and specific applications. Face recognition is also used for biometric surveillance. Banks, retail stores, stadiums, airports and other facilities use face recognition to reduce crime and prevent violence.

9.2.9 Conclusion

A robot car using Raspberry Pi is designed to detect and recognize the human faces from the images taken from PiCamera attached to the Raspberry Pi. Using Python programming language, the system is being built such that it can face detect and recognize in real time scenarios. For this solution Kairos Face recognition software is being used which have a free developer account. Face recognition is tested with various types of facial views like the front view and side view. The Round Trip Time for robot car to take picture and recognize face is nearly 3 seconds. The efficiency of the system was analysed based on the rate of face detection in real time. As per the analysis, this current system shows tremendous performance efficiency where the face detection and recognition can be performed even with very low-quality images.

The authors would like to thank Dr. Gregor von Laszewski for his support and suggestions in writing this paper.

9.2.10 Links

The code is located at

- https://github.com/cloudmesh-community/hid-sp18-711/tree/master/project-code

10 REFERENCES

[1] S. B. Y. Riddhi Patel, "A literature survey on face recognition techniques," *International Journal of Computer Trends and Technology*, vol. 5, p. 189, Nov. 2013.

[2] B. B. M. Divyarajsinh N. Parmar, "Face recognition methods & applications," *Int.J.Computer Technology & Applications*, vol. 4, pp. 84–86, 2013 [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1403/1403.0485.pdf

[3] J. D. WEST, "FACE detection vs. FACE recognition." Web Page, May-2017 [Online]. Available: https://www.facefirst.com/blog/face-detection-vs-face-recognition/

[4] BIOMETRICS, "Facial recognition." Web Page, Jan-2014 [Online]. Available: https://findbiometrics.com/solutions/facial-recognition/

[5] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 ieee computer society conference on computer vision and pattern recognition. CVPR 2001*, 2001, vol. 1 [Online]. Available: http://http://ieeexplore.ieee.org/abstract/document/990517

[6] OpenCV, "About opencv." Web Page, 2017 [Online]. Available: https://opencv.org/about.html

[7] D. Technologies, "MQTT to connect raspberry pi to internet of things." Web Page, Sep-2017 [Online]. Available: https://hackaday.io/project/27344-mqtt-to-connect-raspberry-pi-to-internet-of-things

[8] Arduino, "Arduino software (ide)." 2015 [Online]. Available: https://www.arduino.cc/en/Guide/Environment

[9] G. von Laszewski, “ESP8266.” Code Repository, Sep-2017 [Online]. Available: https://cloudmesh.github.io/classes/lesson/iot/esp8266/esp8266.html

[10] B. Landoni, “Raspberry pi and the camera pi module: Face recognition tutorial.” Web Page, Oct-2014 [Online]. Available: https://www.open-electronics.org/raspberry-pi-and-the-camera-pi-module-face-recognition-tutorial/