



# PRESIDENCY UNIVERSITY

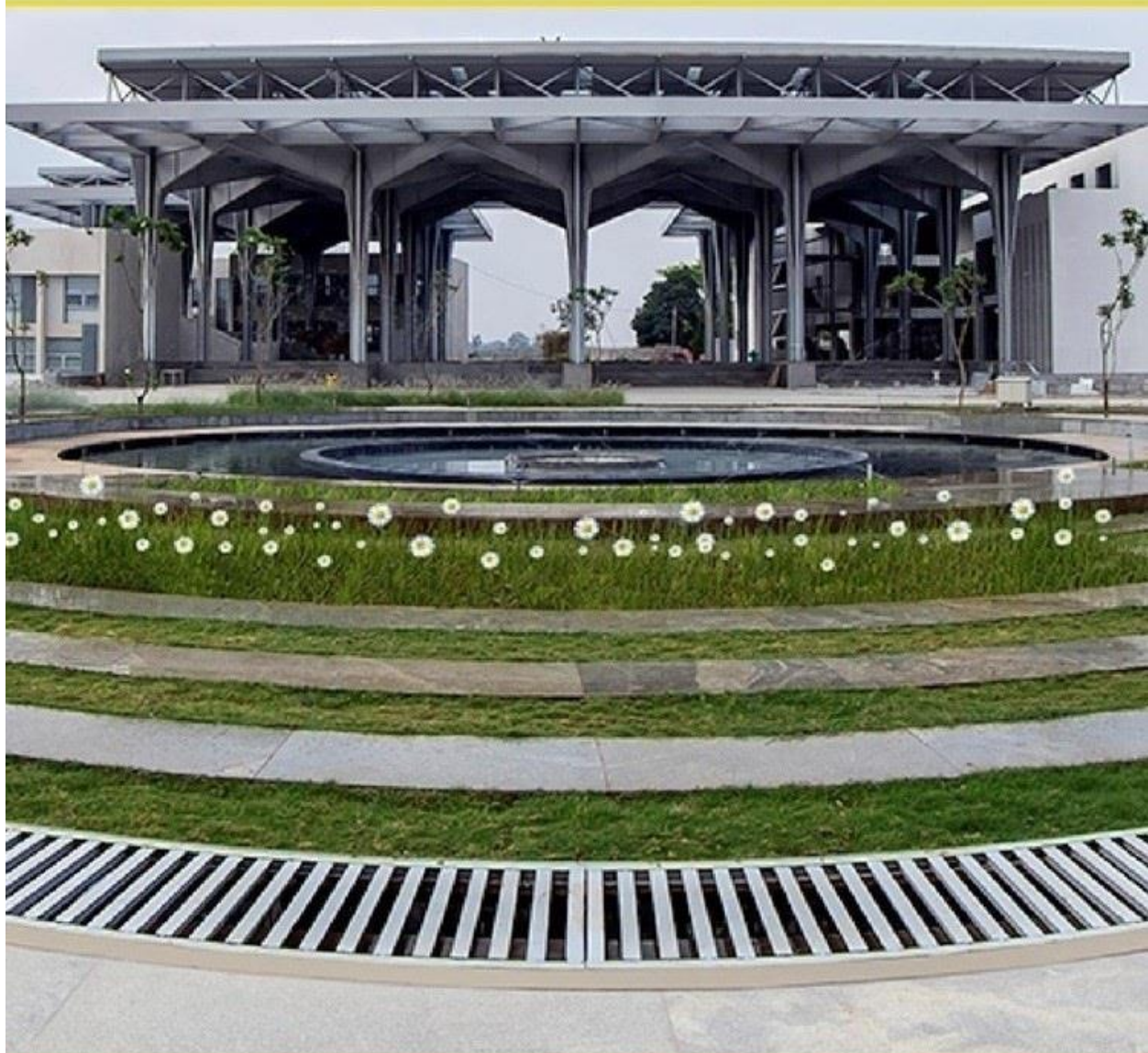
Private University Estd. in Karnataka State by Act No. 41 of 2013



Department of Computer Science and Engineering

Internet of Things -IoT CSE 202

Lab Manual 2018-19



[www.presidencyuniversity.in](http://www.presidencyuniversity.in)

## Laboratory Manual



**Internet of Things (IoT)**

**For**

**Final year Students (CSE)**

Itgalpur Rajanakunte, Yelahanka, Bengaluru, Karnataka 560064

## **FOREWORD**

It is my great pleasure to present this laboratory manual for Final year engineering students for the subject Internet of Things (IoT).

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

As you may be aware that Presidency University has already been awarded with Best Emerging University in south India by ASSOCHAM, INDIA and it's our endure to technically equip our students take the advantage of the procedural aspects of this certification.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relieve them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

Mohammed Mujeer  
Assistant Professor,  
Department of CSE,  
Presidency University

## **LABORATORY MANUAL CONTENTS**

This manual is intended for the final year students of Computer Science and Engineering in the subject of Internet of Things(IoT). This manual typically contains practical/lab sessions related Raspberry pi and Android implemented in Python programming covering various aspects related the subject to enhanced understanding.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions

Dr. Mohan K.G

HOD,CSE

Mr. Mohammed Mujeer

## **PROGRAME OBJECTIVES**

B.Tech. Computer Science Engineering graduates of Presidency University, will be able to:

1. Direct and coordinate activities concerned with software/hardware design, development, and testing.
2. Design and develop products and systems for various applications.
3. Monitor and analyze computing system performance for network traffic, security, and capacity.
4. Order and maintain inventory of computing equipment for customer premises equipment, facilities, access networks and backbone network
5. Operate and trouble shoot computer-assisted engineering, design software and equipment to perform various engineering tasks.

## **PROGRAMME OUTCOMES**

On successful completion of B.Tech. Computer Science Engineering programme from Presidency University a student will be able to

1. Apply mathematics, science and engineering fundamentals to computer science engineering.
2. Solve basic computer science engineering problems using first principles of mathematics and engineering sciences.
3. Design computing systems for domestic, commercial and industrial applications.
4. Investigate computer engineering problems including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.
5. Select appropriate techniques, resources, and modern computer science engineering tools, including prediction and modelling, to complex engineering activities, with an understanding of the limitations.
6. Identify societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to computer science engineering practice.
7. Appraise professional ethics and responsibilities and norms of computer science engineering practice.
8. Relate the impact of computer science engineering solutions in a societal context and demonstrating knowledge of and need for sustainable development.
9. Assess management and business practices, such as risk and change management, and understanding their limitations in the context of computer science engineering.
10. Demonstrate the ability to work as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.
11. Write effective reports and design documentation, making effective presentations, and giving and receiving clear instructions.
12. Justify the need for engaging in independent and life-long learning.

**DOs and DON'Ts in Laboratory:**

1. Make entry in the Log Book as soon as you enter the Laboratory.
2. All the students should sit according to their roll numbers starting from their left to right.
3. All the students are supposed to enter the terminal number in the log book.
4. Do not change the terminal on which you are working.
5. All the students are expected to get at least the algorithm of the program/concept to be implement.
6. Strictly follow the instructions given by the teacher/Lab Instructor.

## LAB INDEX

**Design, Develop and implement following using Arduino, Raspberry Pi compiler and Python language in Linux/Windows environment.**

<b>Arduino Experiments</b>	
<b>1</b>	Installation of arduino IDE & Arduino program to implement scrolling LED, to glow even/odd LED.
<b>2</b>	Arduino program to demonstrate usage of push button to control the LED
<b>3</b>	Arduino program to demonstrates traffic control system
<b>4</b>	Arduino program to demonstrates usage of servo motor with potentiometer
<b>Raspberry pi Experiments</b>	
<b>5</b>	Installation of Raspberry pi software
<b>6</b>	Working basic commands on Raspberry pi & to demonstrate remote logging in raspberry pi
<b>7</b>	Raspberry pi program to implement blinking LED
<b>8</b>	Raspberry pi program to implement camera module for video
<b>9</b>	Raspberry pi program to obtain the temperature using DHT sensors
<b>10</b>	Using a Raspberry Pi with distance sensor (ultrasonic sensor HC-SR04)
<b>11</b>	Raspberry pi program to implement Garage spot light

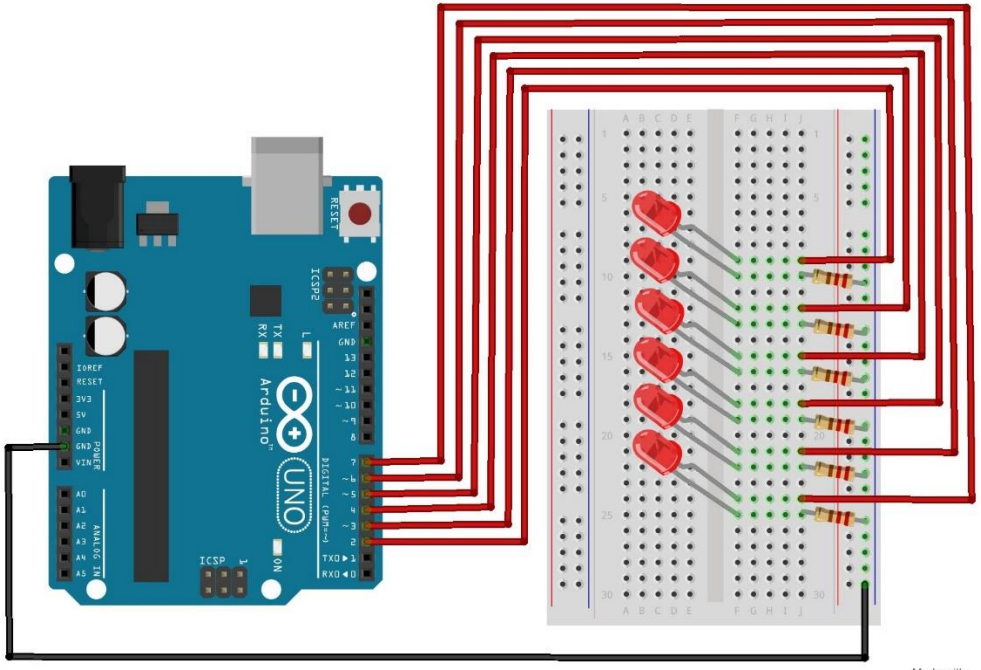


## Experiment No: 1

### Arduino program to implement scrolling LED

This project will blink 6 LEDs, one at a time, in a back and forth formation. This type of circuit was made famous by the show Knight Rider which featured a car with looping LEDs.

Sl no.	Equipment's Needed
1	Arduino board(1)
2	Power cable(1)
3	Breadboard(1)
4	LED(6)
5	220 $\Omega$ Resistor(6)
6	Jumper Wires(7)

Step 1:	Connect the Arduino board to your computer using the USB cable.
Step 2:	Select the board and serial port as outlined in earlier section.
Step 3:	Make the Connection as follows:  Made with fritzing
Step 4:	Open project code to type following program /* For Loop Iteration  Demonstrates the use of a for() loop.

Lights multiple LEDs in sequence, then in reverse.

The circuit:

\* LEDs from pins 2 through 7 to ground

created 2006

by David A. Mellis

modified 30 Aug 2011

by Tom Igoe

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/ForLoop>

\*/

```
int timer = 100;      // The higher the number, the slower the timing.
```

```
void setup() {  
  // use a for loop to initialize each pin as an output:  
  for (int thisPin = 2; thisPin < 8; thisPin++) {  
    pinMode(thisPin, OUTPUT);  
  }  
}
```

```
void loop() {  
  // loop from the lowest pin to the highest:  
  for (int thisPin = 2; thisPin < 8; thisPin++) {  
    // turn the pin on:  
    digitalWrite(thisPin, HIGH);  
    delay(timer);  
    // turn the pin off:  
    digitalWrite(thisPin, LOW);  
  }  
}
```

```
  // loop from the highest pin to the lowest:  
  for (int thisPin = 7; thisPin >= 2; thisPin--) {  
    // turn the pin on:  
    digitalWrite(thisPin, HIGH);  
    delay(timer);  
    // turn the pin off:  
    digitalWrite(thisPin, LOW);  
  }  
}import RPi.GPIO as GPIO  
import time
```

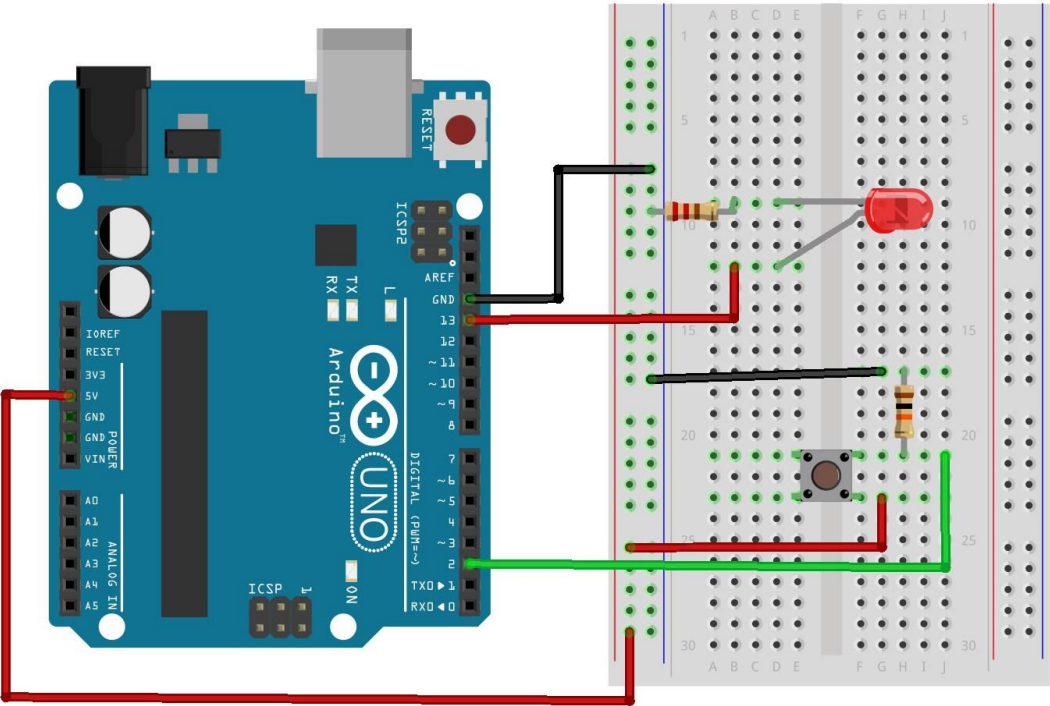
```
i=0  
while(i<4):  
    if(i%2==0):
```

	<pre> GPIO.setmode(GPIO.BCM) GPIO.setup(18,GPIO.OUT) GPIO.output(18,GPIO.HIGH) time.sleep(3) GPIO.output(18,GPIO.LOW) GPIO.setmode(GPIO.BCM) GPIO.setup(23,GPIO.OUT) GPIO.output(23,GPIO.HIGH) time.sleep(3) GPIO.output(23,GPIO.LOW) GPIO.cleanup()  else: GPIO.setmode(GPIO.BCM) GPIO.setup(22,GPIO.OUT) GPIO.output(22,GPIO.HIGH) time.sleep(3) GPIO.output(22,GPIO.LOW) GPIO.setmode(GPIO.BCM) GPIO.setup(25,GPIO.OUT) GPIO.output(25,GPIO.HIGH) time.sleep(3) GPIO.output(25,GPIO.LOW) GPIO.cleanup()  i+=1 </pre>
--	---

## Experiment No: 2

### Arduino program to demonstrate usage of push button to control the LED

Sl no.	Equipment's Needed
1	Arduino board
2	Power cable
3	Push button Switch
4	Breadboard
5	LED
6	220 $\Omega$ Resistor
7	Jumper Wires

Step 1:	Connect the Arduino board to your computer using the USB cable.
Step 2:	<p>Make the Connection as follows:</p>  <p>Made with fritzing</p>
Step 3:	<p>Open project code – and write the below program</p> <pre>/*   Button    Turns on and off a light emitting diode(LED) connected to digital   pin 13, when pressing a pushbutton attached to pin 2.   */</pre>

The circuit:

- \* LED attached from pin 13 to ground
- \* pushbutton attached to pin 2 from +5V
- \* 10K resistor attached to pin 2 from ground

\* Note: on most Arduinos there is already an LED on the board attached to pin 13.

created 2005

by DojoDave <<http://www.0j0.org>>

modified 30 Aug 2011

by Tom Igoe

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/Button>

\*/

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin = 13;     // the number of the LED pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

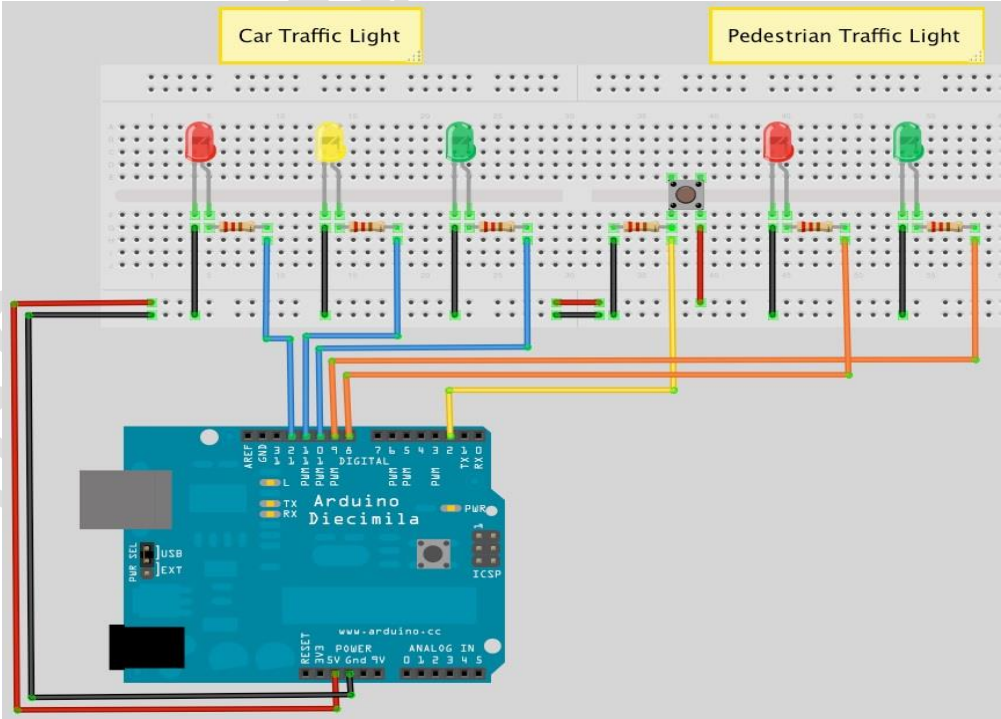
	<pre> } } </pre>
Step 4:	Select the board and serial port as outlined in earlier section.
Step 5:	Click upload button to send sketch to the Arduino

## Experiment No: 3

### Arduino program to implement traffic control system

In this experiment we are going to make a traffic light with pedestrian light along with button, to request to cross the road. The Arduino will execute when the button is pressed by changing the state of light to make the cars stop and allow the pedestrian to cross safely.

Sl no.	Equipment's Needed
1	Arduino board(1)
2	Power cable(1)
3	Breadboard(1)
4	LED(6) 2x Red LED's 1x Yellow LED 2x Green LED's
5	220 $\Omega$ Resistor(6)
6	Jumper Wires(7)
7	Tactile Switch

Step 1:	Connect the Arduino board to your computer using the USB cable.
Step 2:	Select the board and serial port as outlined in earlier section.
Step 3:	Make the Connection as follows: 
Step 4:	Open project code to type following program

```

// Interactive Traffic Lights

int carRed = 12; // To assign the car lights
int carYellow = 11;
int carGreen = 10;
int pedRed = 9; // To assign the pedestrian lights
int pedGreen = 8;
int button = 2; // Tactile button pin
int crossTime = 5000; // time allowed to cross

unsigned long changeTime; // time since button pressed

void setup()
{
  pinMode(carRed, OUTPUT);
  pinMode(carYellow, OUTPUT);
  pinMode(carGreen, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(button, INPUT); // button on pin 2
  // turn on the green light
  digitalWrite(carGreen, HIGH);
  digitalWrite(pedRed, HIGH);
}

void loop()
{
  int state = digitalRead(button);
  /* check if button is pressed and it is
  over 5 seconds since last button press */
  if (state == HIGH && (millis() - changeTime) > 5000)
  {

```



```

// Call the function to change the lights
changeLights();
}
}

void changeLights() {
digitalWrite(carGreen, LOW); // green off
digitalWrite(carYellow, HIGH); // yellow on
delay(2000); // wait 2 seconds
digitalWrite(carYellow, LOW); // yellow off
digitalWrite(carRed, HIGH); // red on
delay(1000); // wait 1 second till its safe
digitalWrite(pedRed, LOW); // ped red off
digitalWrite(pedGreen, HIGH); // ped green on
delay(crossTime); // wait for preset time period
// flash the ped green
for (int x=0; x<10; x++) {
digitalWrite(pedGreen, HIGH);
delay(250);
digitalWrite(pedGreen, LOW);
delay(250);
}
// turn ped red on
digitalWrite(pedRed, HIGH);
delay(500);
digitalWrite(carYellow, HIGH); // Yellow will switch on
digitalWrite(carRed, LOW); // red will switch off
delay(1000);
digitalWrite(carGreen, HIGH);
digitalWrite(carYellow, LOW); // Yellow will switch off

// record the time since last change of lights
changeTime = millis();

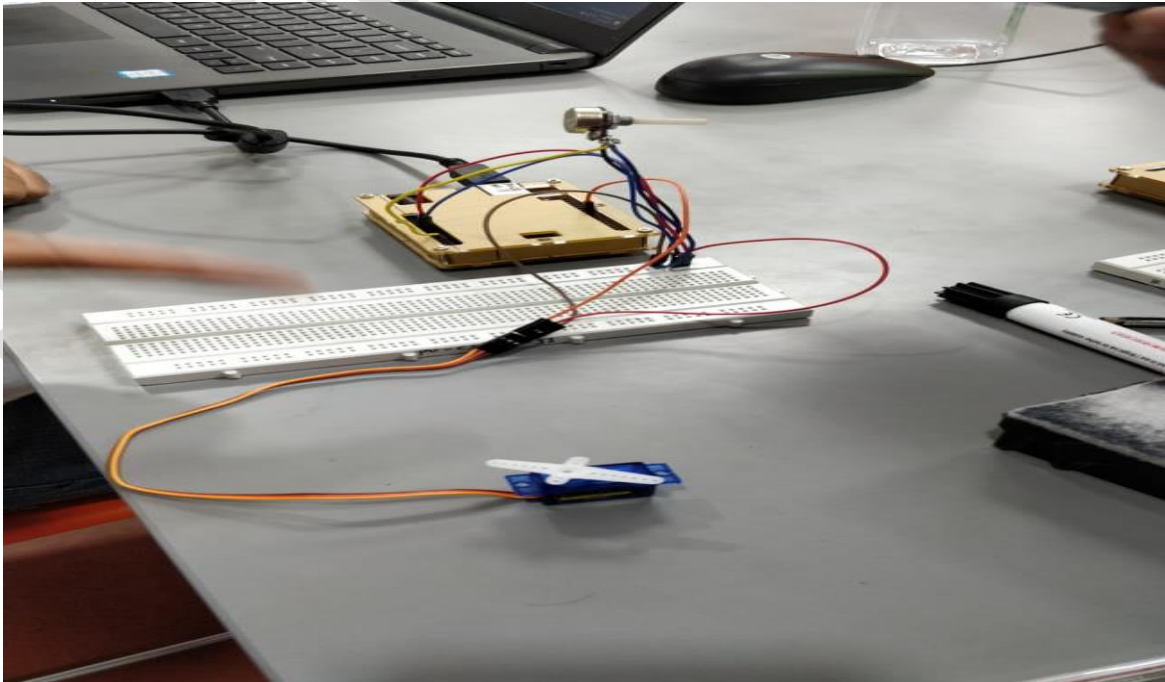
```

	<pre>// Retun / Loop }</pre>
--	------------------------------

## Experiment No: 4

### Arduino program to demonstrate control of servo motor using potentiometer

Sl no.	Equipment's Needed
1	Arduino board
2	Power cable
3	Servo Motor
4	Potentiometer

Step 1:	Connect the Arduino board to your computer using the USB cable.
Step 2:	Select the board and serial port as outlined in earlier section.
Step 3:	<p>Lots of IoT applications with respect to robotics use servo motor. Servo motor comes with 3 different arms with 2 variants full arm and half arm and runs 1800. Black pin-&gt;Gnd pin of arduino Red Pin-&gt; +5V of arduino Orange -&gt;Used to send pulse width modulation, and is connected to pin no 09 from arduino</p> <p>Potentiometer- is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat. 1st Nob-&gt;Gnd pin of arduino 2nd Nob-&gt;A0 pin of arduino 3rd Nob-&gt;+5v pin of arduino</p>
Step 4:	<p>Make the Connection as follows:</p> 

Step 5:	<p>Open project code to type following program</p> <pre> /* Controlling a servo position using a potentiometer (variable resistor) by Michal Rinott &lt;<a href="http://people.interaction-ivrea.it/m.rinott">http://people.interaction-ivrea.it/m.rinott</a>&gt;  modified on 8 Nov 2013 by Scott Fitzgerald <a href="http://www.arduino.cc/en/Tutorial/Knob">http://www.arduino.cc/en/Tutorial/Knob</a> */  #include &lt;Servo.h&gt;  Servo myservo; // create servo object to control a servo  int potpin = 0; // analog pin used to connect the potentiometer int val; // variable to read the value from the analog pin  void setup() {   myservo.attach(9); // attaches the servo on pin 9 to the servo object }  void loop() {   val = analogRead(potpin); // reads the value of the potentiometer (value between 0 and 1023)   val = map(val, 0, 1023, 0, 180); // scale it to use it with the servo (value between 0 and 180)   myservo.write(val); // sets the servo position according to the scaled value   delay(15); } // waits for the servo to get there </pre>
---------	---

## Experiment No: 5

### Installation of Raspberry pi software

Sl no.	Equipment's Needed
1	Raspberry Pi
2	VGA cable
3	Power Cable

**Introduction:** The Raspberry Pi is a fully-fledged mini computer, capable of doing whatever you might do with a computer. It comes with 4x USB, HDMI, LAN, built-in Bluetooth/WiFi support, 1GB RAM, 1.2GHz quad-core ARM CPU, 40 GPIO (General Purpose Input Output) pins, audio and composite video output, and more.



One can use Raspberry Pis as home security cameras, server monitoring devices, cheap headless machines (basically running low-weight scripts 24/7 with a low cost-to-me)... others have used them for media centers and even for voice-enabled IoT devices. The possibilities are endless, but first we need to get acquainted!

Make sure that, if you do get a case, it has openings for the GPIO pins to be connected, you will also need a [1000mA+ mini usb power supply](#) and at least an 8GB micro SD card, but I would suggest a [16 GB micro SD card](#) or greater.

You will also want to have a spare monitor (HDMI), keyboard, and mouse handy to make things easier when first setting up. You wont will eventually be able to control your Pi remotely, so you

won't always need a separate keyboard, mouse, and monitor. If you don't have a monitor with HDMI input, you can buy something like an HDMI to DVI converter.

If you're using an older version board, please see what you might need to change, for example, the older Raspberry Pis take a full-sized SD card, but the latest model requires a micro SD card. Also, the Raspberry Pi 3 Model B has built-in wifi, where the older models will require a wifi dongle.

A typical Raspberry Pi shopping list, assuming you have a mouse, keyboard, and HDMI monitor that you can use temporarily while setting up is:

1. [Raspberry Pi](#) -
2. [1000mA+ mini usb power supply](#) -
3. [16 GB micro SD card](#) -

Additionally, if you plan to join us on the initial GPIO (General Purpose Input Output pins) tutorials, you will also want to pick up:

1. 10 x [Male-to-Female jumper wires](#) (you should consider just buying a bunch of these so you have plenty in the future).
2. 1 x [Breadboard](#) (You may also want multiples of these)
3. 3 x [LED light](#) (...more wouldn't hurt)
4. ~6 x Resistors (between 300 and 1K Ohm). You will need at least 1K and 2K ohms for the distance sensor, then ~300-1K resistance per LED bulb. You probably should just buy a kit, they're super cheap.
5. 1 x [HC-SR04 Ultrasonic Distance Sensor](#) (...you know what I'm going to say...think about maybe a few.)
6. 1 x [Raspberry Pi](#) camera module. You only need one of these!

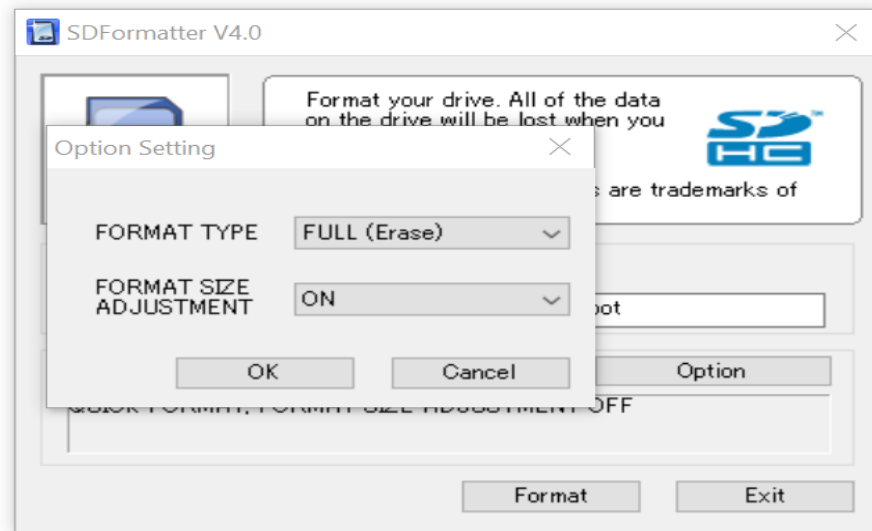
For the jumpers, breadboard, and leds, you could also just buy a kit, something like: [this GPIO starter kit](#).

There are also a few ways to install and use an operating system on the Raspberry Pi. The most user-friendly method is to use the NOOBS (New Out of Box Software) installer. If you're comfortable enough, you can just simply download the operating system ISO, format the SD card, mount the ISO, and boot the Pi. If that sounds like gibberish to you, then follow along with the NOOBS installation option.

While we're working with the SD card, let's go ahead and [Download NOOBS](#), which is just over 1GB.

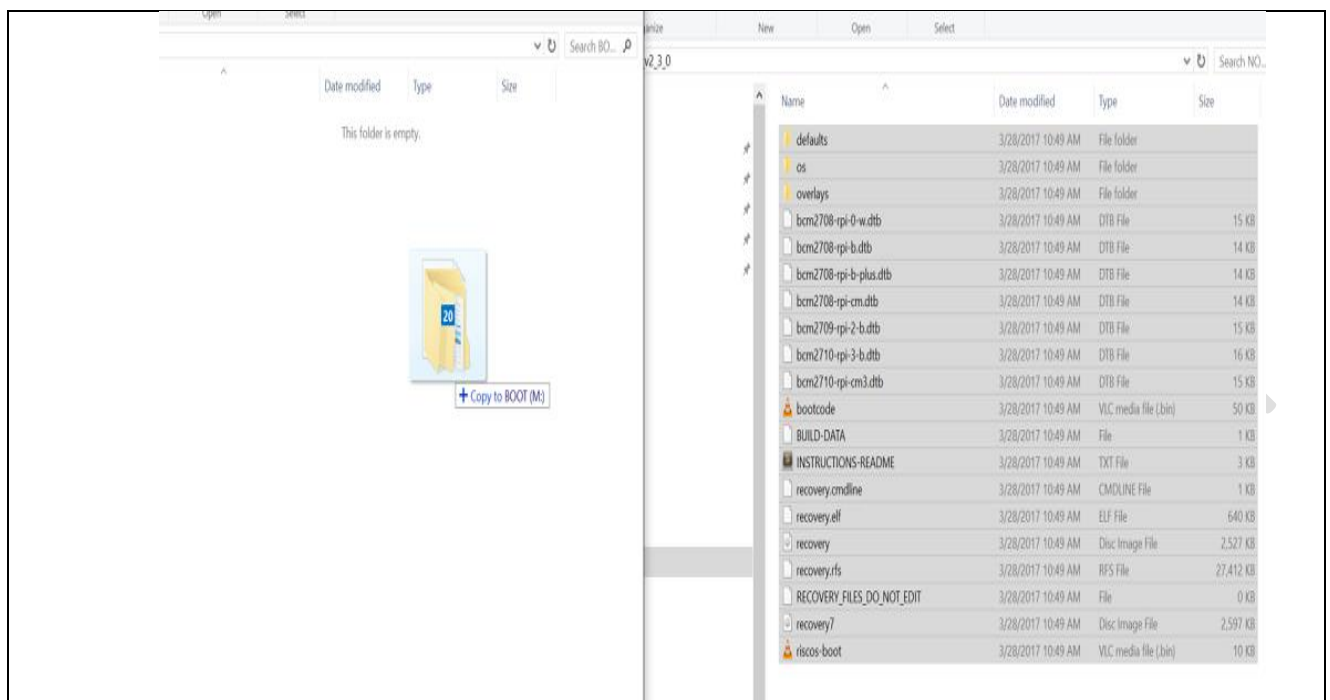
First, we must format the SD card. If you are on Windows, you can use [SD Formatter](#). Mac users can also use SD Formatter, but they have a built in formatter, and Linux users can use GParted. Whatever

the case, you need to format the SD card, do not do a "quick format" and do make sure you have the "resize" option on. Using SDFormatter on Windows, and choosing options:

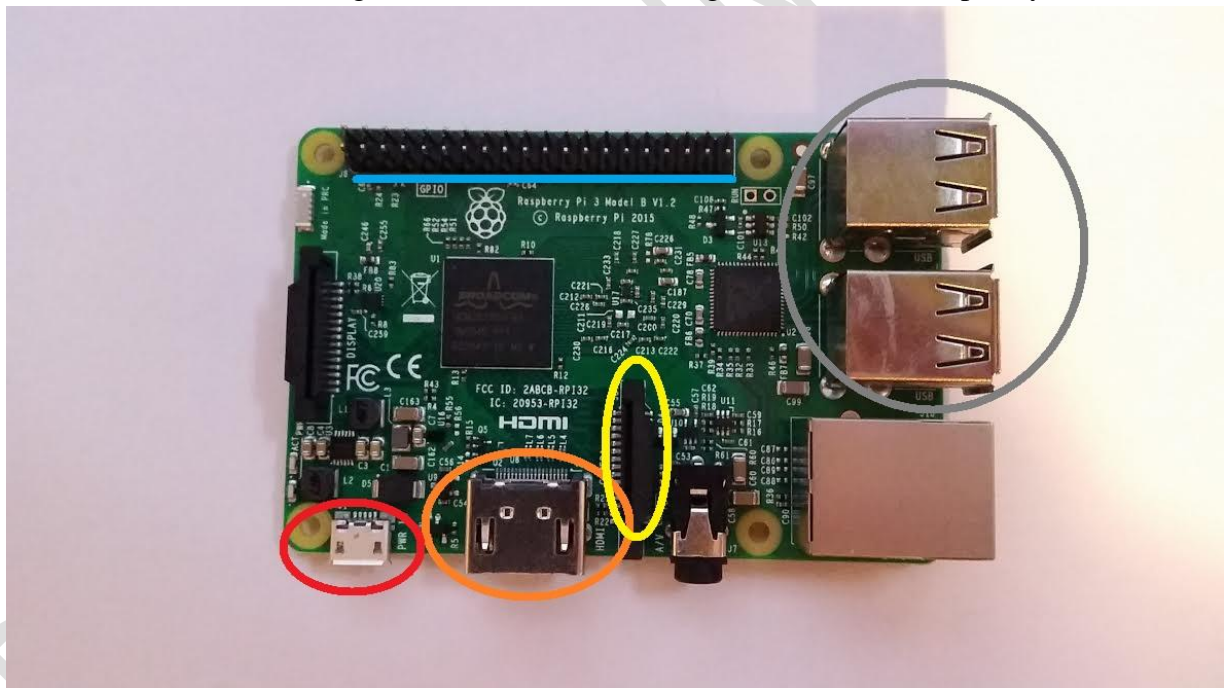


This should go without saying, but do make sure you're formatting the right drive. This will format any flash drive, in alphabetical order. If you had something plugged in already, like your favorite USB drive, and forgot about it, that'd likely be the default choice to format, and then you'd spend all afternoon trying to recover your data rather than enjoying playing with your Raspberry Pi.

Once you're done, great. Now, assuming you've downloaded the NOOBS package, let's go ahead and extract that. Now, we want to copy all these NOOBS contents to our SD Card. Do not drag the directory, but rather the contents:



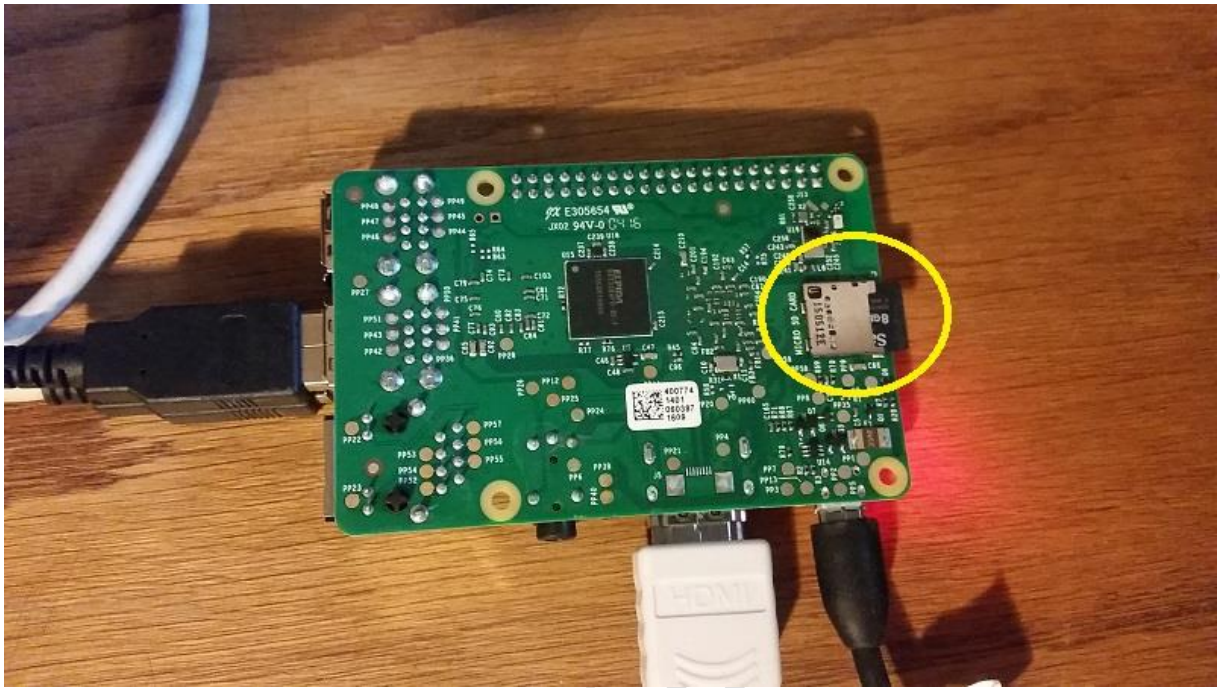
While that's transferring, let's talk about a few things on the actual Raspberry Pi board:



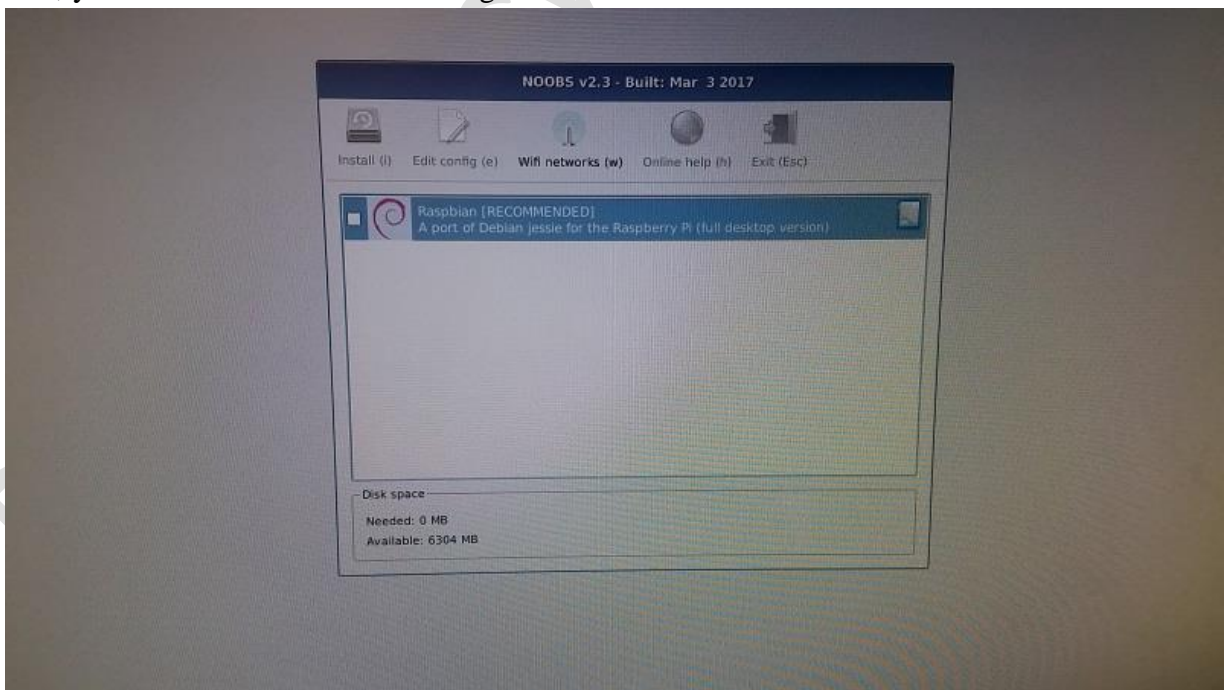
The GPIO (General Purpose Input/Output) pins are underlined in blue. We can use these to control peripheral devices like motors, servos, and more. Circled in red is the micro usb power input for the board. In orange, the HDMI output port. The yellow is where you can plug in the Raspberry Pi camera module. The grey circle has the USB ports. This is obviously not everything, but these are the main things to note.



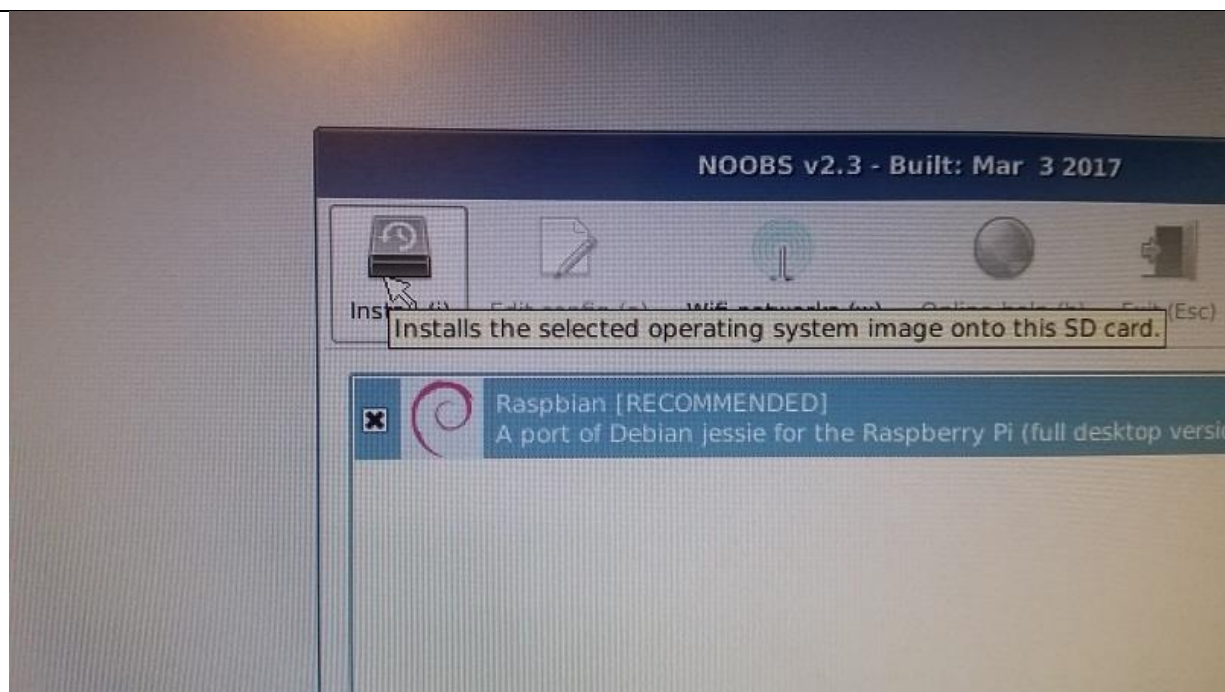
Once everything is transferred to the micro SD card, you can put it in the Raspberry Pi. The slot is on the bottom side of the board, circled in yellow here:



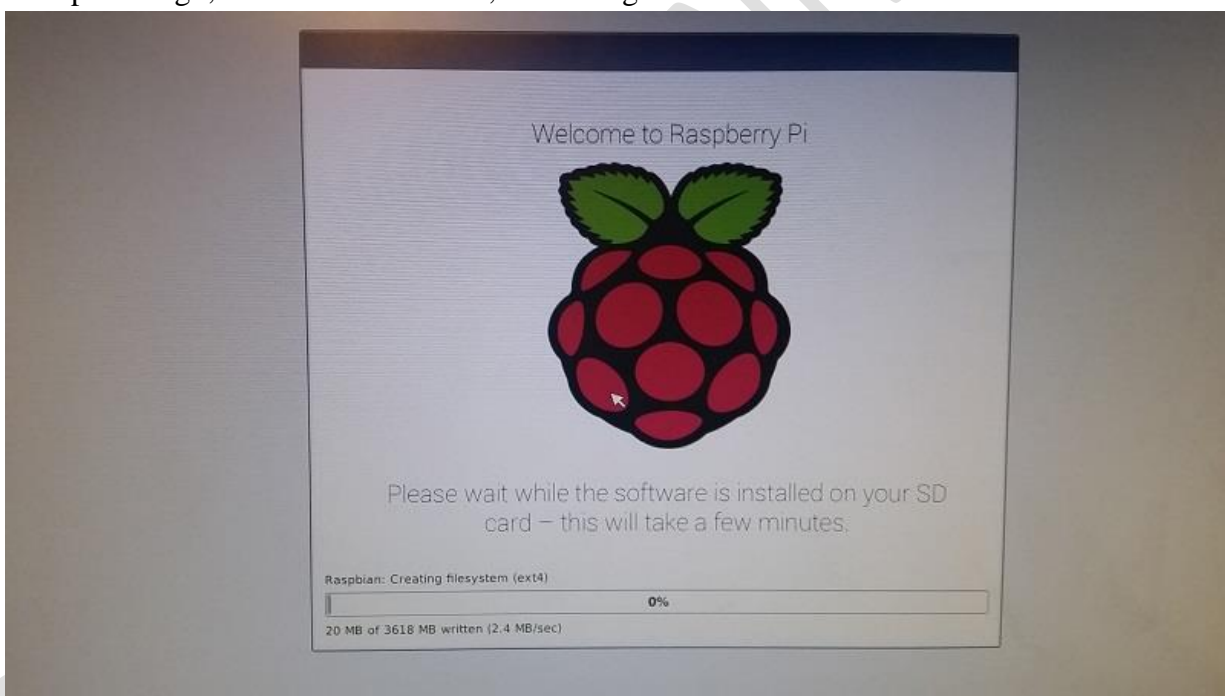
Once you've got the SD card plugged in, go ahead and plug in your keyboard, mouse, and HDMI cable to your monitor. Finally, plug in the power, and this will start up the Raspberry Pi. Once fully loaded, you should land on the following screen:

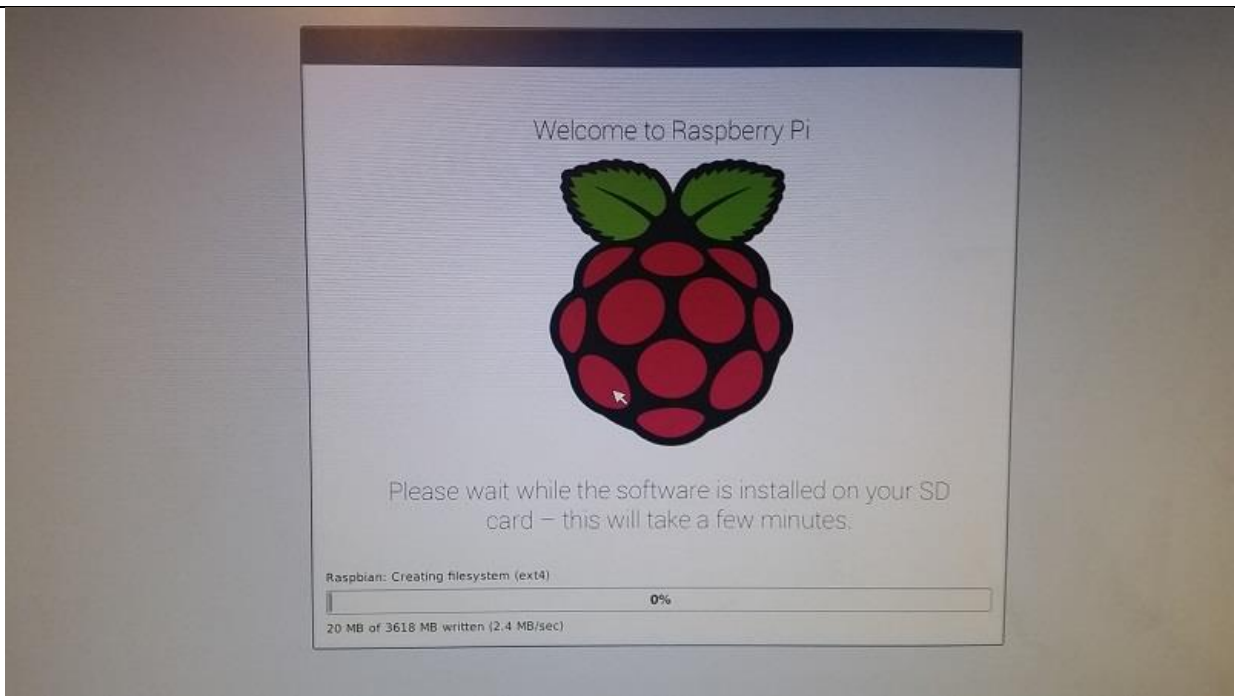


Now you can choose the operating system. In my case, the only option is Raspbian, so I will check that box, then click "install."

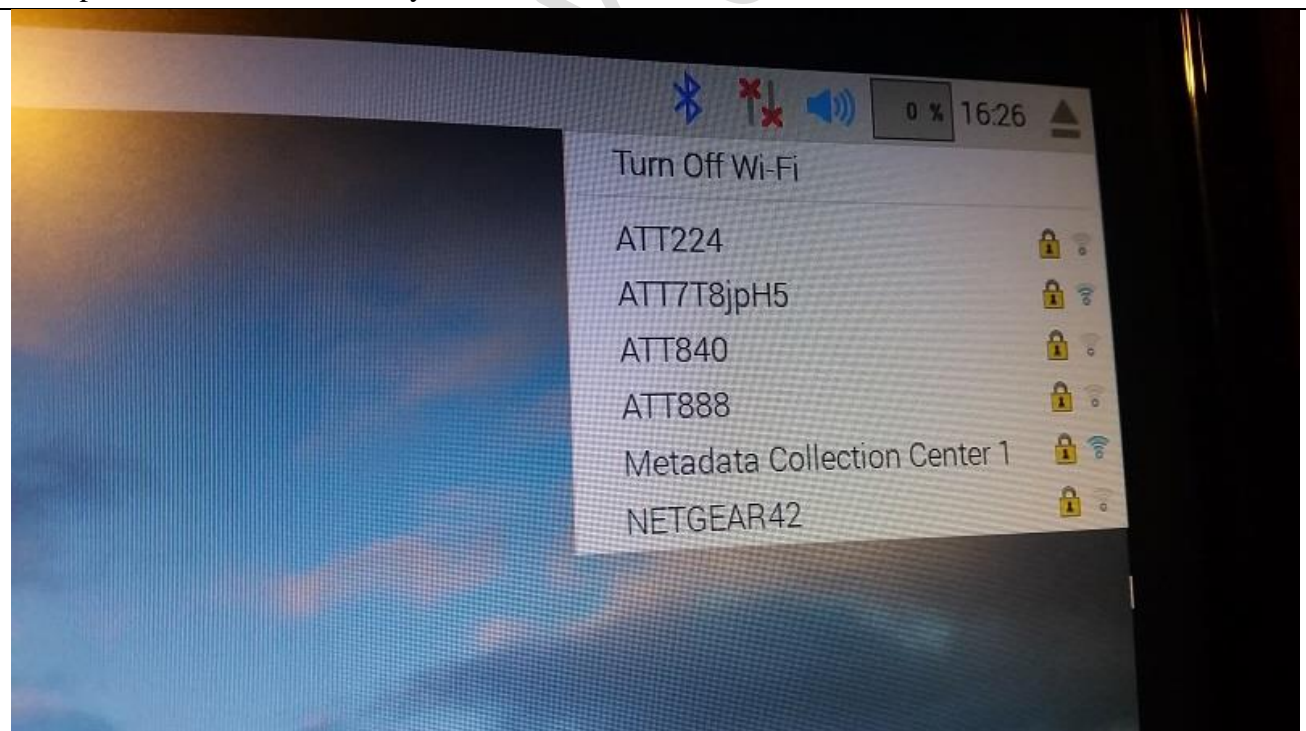


Let the process go, this will take a while, something like 20-30 minutes or so.





Once that's done, hit okay and the device should reboot to desktop. While on the desktop, wait for a moment for wifi to start up and find available connections. Connect to your wifi network if possible. You can also plug directly in with an ethernet cable if you don't have wifi. You can also just continue interacting directly with the Raspberry Pi with the mouse and keyboard connected to it if you like, but I prefer to access it remotely.



Once we've connected to our network, we'd like to actually interface with the Pi. First, we want to update. Open a terminal by either right clicking on the desktop and opening terminal that way, or by doing control+alt+t. Now, in the terminal, do:



```
$sudo apt-get update  
and then
```

```
$ sudo apt-get upgrade
```

You do not type the \$ sign, it's there to denote when you're typing something in the command line. The upgrade might take a minute. While we wait, your Raspberry Pi's default credentials are: username: pi password: raspberry. For some reason, the apt-get upgrade for me was taking absurdly long. You need to be connected to your network, and have internet access, so make sure you have those things first before doing this, but I still was having trouble. I was able to solve this by doing:

```
$ sudo nano /etc/apt/sources.list
```

Then replace everything here with:

```
deb http://archive.raspbian.org/raspbian jessie main contrib non-free  
deb-src http://archive.raspbian.org/raspbian jessie main contrib non-free
```

control+x, y, enter

```
$ sudo apt-get dist-upgrade
```

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

```
$ sudo apt-get upgrade
```

To save some space you can also do: \$ sudo apt-get purge wolfram-engine and then \$ sudo apt-get autoremove. This alone freed up almost 700mb of space for me.

This is going to conclude the first part of this tutorial series. In the next tutorial, we're going to cover how we can remotely access our Raspberry Pi.

## Experiment No: 6

### Working basic commands on Raspberry pi

Sl no.	Equipment's Needed
1	Raspberry Pi
2	VGA cable

#### Introduction:

In this Experiment, we're going to have a quick crash course for using the terminal. Most of our interactions with the Raspberry Pi will be via shell, since this is the simplest and most lightweight to keep your Pi accessible.

To begin, let's assume you've just logged in. The location that you will usually log in to will be the root directory for your user. In this case, our user is "pi," so we log in to /home/pi. We can confirm this by using the command: pwd (print working directory)

```
pi@raspberrypi:~ $ pwd
/home/pi
```

Often times, to denote the terminal, you will see people use the \$ sign. Since, most of the time, people won't be using the same usernames and hostnames, this is the standard to denote that we're operating in the terminal, so, instead, if you googled how you get your current directory in linux, you'd probably see: \$ pwd. This doesn't mean that you should actually type the \$ sign, it's just meant to denote that you're in the terminal.

Next, many times you might see a "permission denied" error when trying to do something. For many tasks, you need to act as the super user, like the administrator account. The command to act as the super user is sudo, which is short for "super user do." An example of this was when we wanted to update and upgrade, we used sudo. Be careful when using sudo to create files...etc. The creator is the owner, so if you use sudo to create the file, it can only be further edited by the super user. Many files will require sudo to edit, however.

To move around the system, you can use `cd`, which stands for "change directory." At any time, we can use `~` to reference the current user's home. In our case `~` is short for `/home/pi`. We can change directories into there with `$ cd ~`.

We were already there, but, just in case you weren't, you are now!

Next, to discover the contents of the directory you are in, you can use `ls` to list directory contents:

```
$ ls
```

```
pi@raspberrypi:~ $ ls
```

```
Desktop Documents Downloads Music Pictures Public python_games Templa
```

It won't always be the case, but often you will see different colors for different types of files. In our case, we just have directories, so they are all the same color.

Let's assume we want to make our own directory, to do this, we can use the `mkdir` command, short for make dir.

```
$ mkdir example
```

We can list the contents again:

```
$ ls
```

Seeing our new directory:

```
pi@raspberrypi:~ $ ls
```

```
Desktop Documents Downloads example Music Pictures Public python_games
```

Then we can change directories into our new directory:

```
$ cd example
```

Maybe we don't want to be here. We can move backwards with: `$ cd ..`

```
pi@raspberrypi:~/example $ cd ..
```

```
pi@raspberrypi:~ $
```

We can also move back a few directories at a time: `$cd ../../`

```
pi@raspberrypi:~ $ cd ../../
```

```
pi@raspberrypi:/ $ ls
```

```
bin boot debian-binary dev etc home lib lost+found man media mnt opt proc root run
sbin srv sys tmp usr var
```

Let's go into our example dir now:

```
$ cd example
```

Next, we can create a simple file with any of the built in editors. There are many to choose from, I think nano is the easiest, so I will use that:

```
$ nano test.py
```

Nano can be used to both create or edit files. If the file doesn't yet exist, it will be created. If it does exist, it will allow you to save a new one.

Let's just add: `print("hi")` to this file. When done, we can exit with `control+x`, then press `y` to save, and then enter to keep the name.

To run this file, we can do: `$ python test.py`

```
pi@raspberrypi:~/example $ python test.py
```

```
hi
```

Next, let's make another directory:

```
$ mkdir downloads
```

```
$ cd downloads
```

To demonstrate remote logging in raspberry pi

Sl no.	Equipment's Needed
1	Raspberry Pi
2	VGA cable

### Introduction :

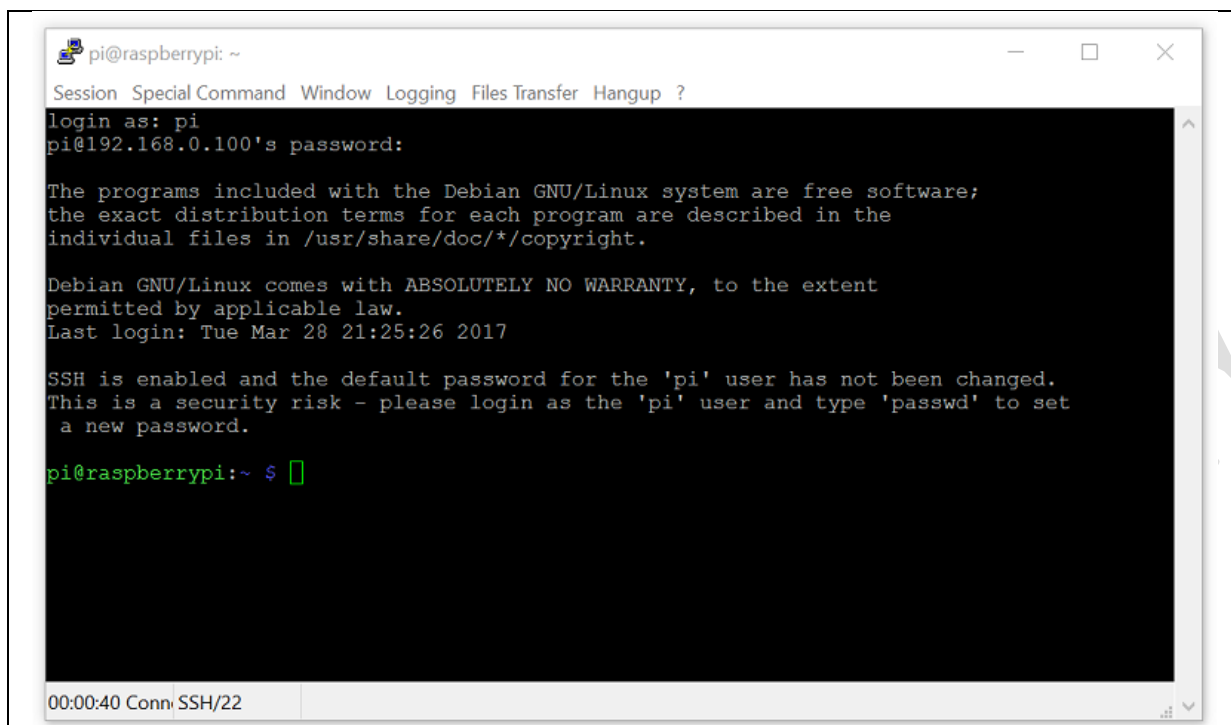
In this Experiment, we're going to cover how we can remotely access our Raspberry Pi, both with SSH and with a remote desktop client. We want to eventually be able to remotely access our Raspberry Pi because much of the "value" of the Raspberry Pi is its size, and that it can be put in a variety of places that we might not want to have a keyboard, mouse, and monitor attached to it at all times and we probably don't want to have to carry over all of this stuff when we do want to access it.

First, let's connect via shell (SSH). Open the terminal on the Raspberry Pi (control+alt+t), and type ifconfig. If you're connected via wifi, then go under the wlan section, and look for your inet address. This will be your local ip, something like 192.168.XX.XXX. We can use this to connect via SSH (user: pi, pass: raspberry), BUT we first have to enable the SSH server. To do this, type sudo raspi-config. Here, we can do quite a few things, but let's head into option #5 interfacing options, next choose the 2nd option for SSH and enable the server. Once this is done, you can shell into the Raspberry Pi.

On Windows, you will need to use an SSH client. But the recommended is [PuTTY](#). Once downloaded, you can open PuTTY, fill in "host name" field with your Pi's local ip, hit enter, and then you will be asked for a username and password.

When successful, you should have something like:





```
pi@raspberrypi: ~
Session Special Command Window Logging Files Transfer Hangup ?
login as: pi
pi@192.168.0.100's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Mar 28 21:25:26 2017

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $
```

This is identical to the terminal you accessed earlier from the Pi's desktop.

Now, sometimes, you might really want to get access to the desktop instead of just the terminal. To do this, you need the "host" to have remote desktop capabilities, as well as whatever remote PC you attempt to use to access it. First, let's deal with the Raspberry Pi. We're going to install xrdp.

```
sudo apt-get remove xrdp vnc4server tightvncserver
```

```
sudo apt-get update
```

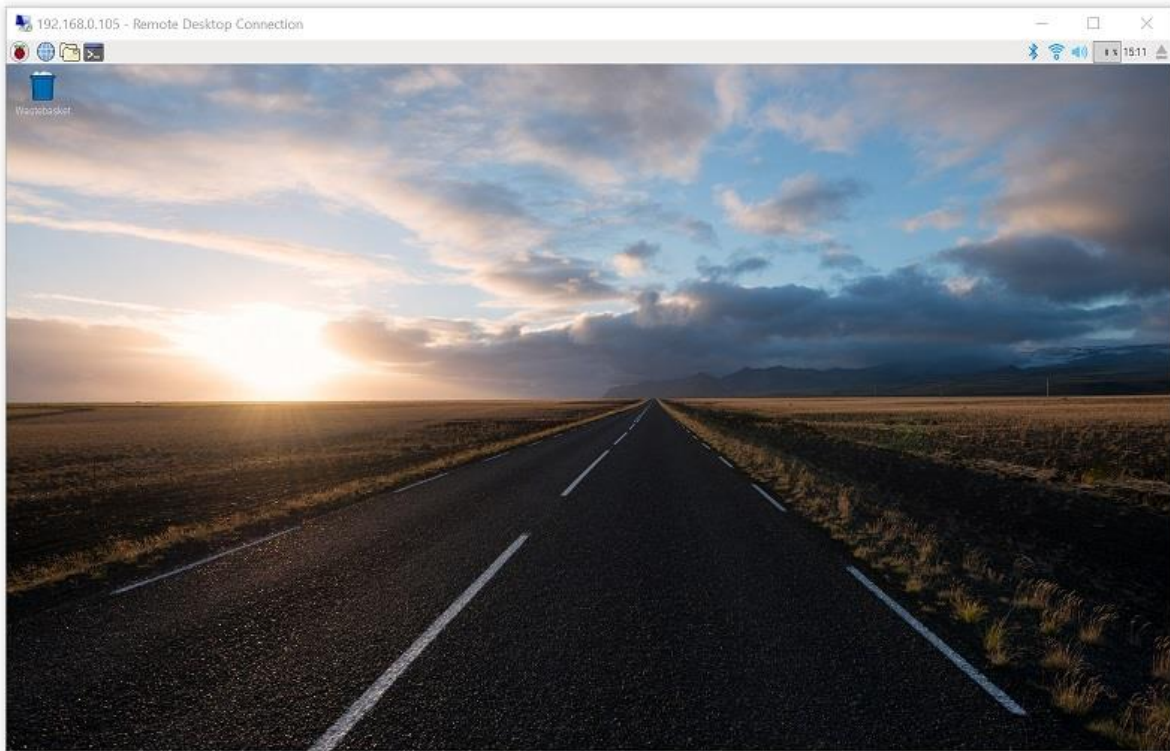
```
sudo apt-get install xrdp
```

```
sudo apt-get install tightvncserver
```

Now let's deal with the computer from which you plan to connect your Raspberry Pi.

For Mac and Windows, you can use the Microsoft application called Remote Desktop. On linux, you can use grdesktop (sudo apt-get install grdesktop). All three of these examples are going to act the same way. Run them, fill in the IP address of the Raspberry pi, the username, the password, and connect!

When done, you should have something like:



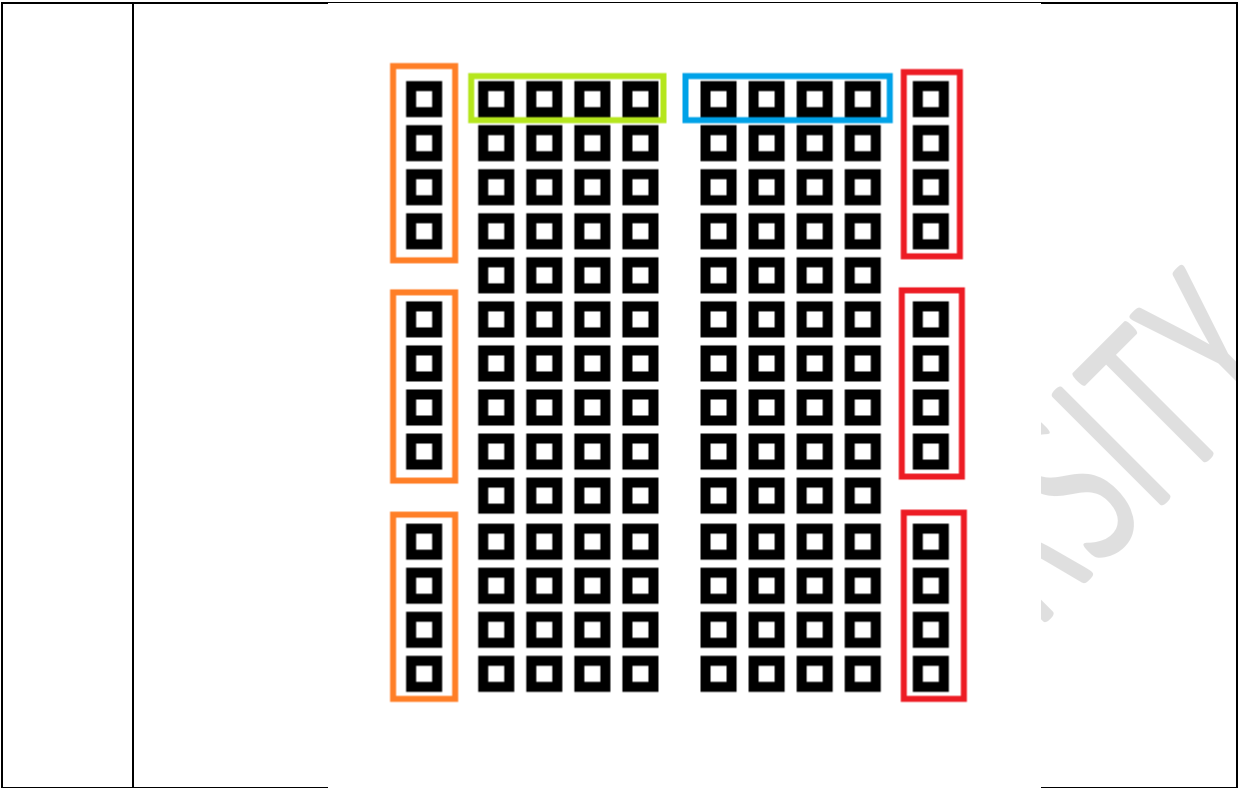
With the SSH server turned on, you will likely be seeing warnings every time you log in that your password is still the default password. The SSH server is turned off by default because, as the Pi has gained popularity, people who might not realize the security risk are using it in places like their homes and businesses. All it takes is someone to connect to your wifi, scan for other local ips, and try them all with default usernames and passwords for various devices, like the Raspberry Pi, and boom they're in. If you want to change your password, you can do `sudo raspi-config`, and it's the first option.

## Experiment No: 7

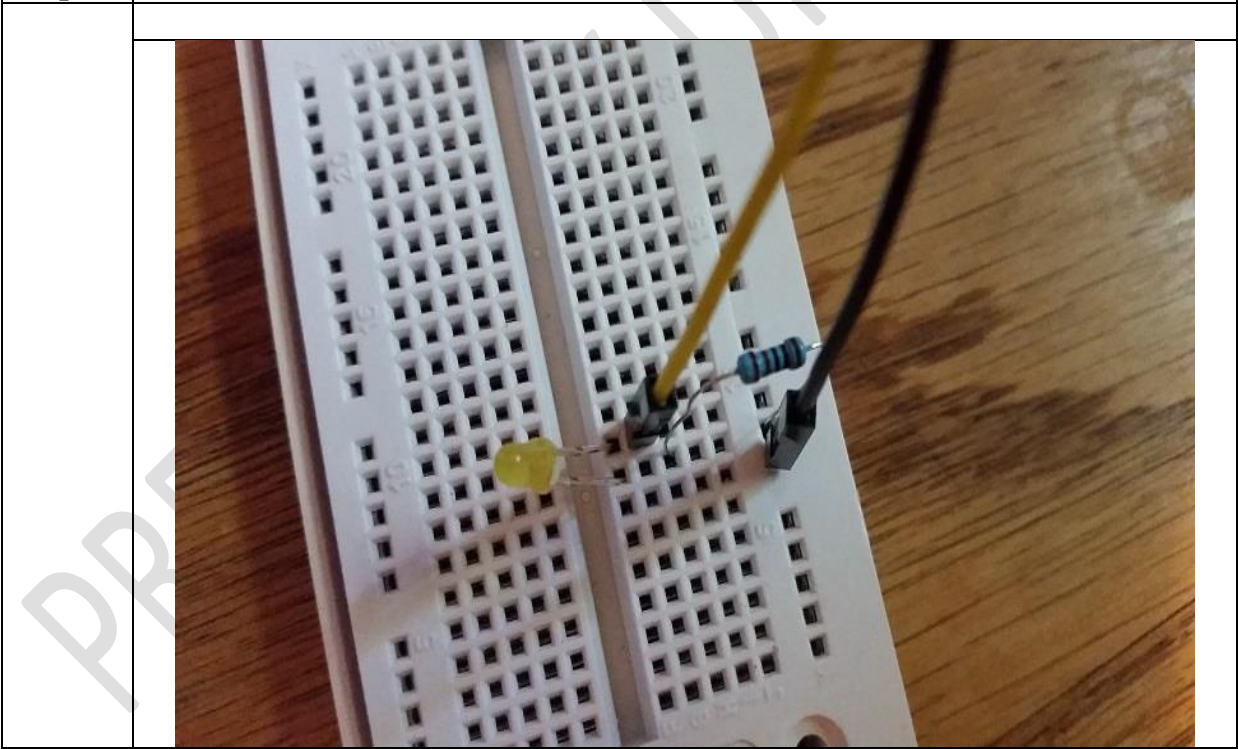
Raspberry pi program to implement blinking LED

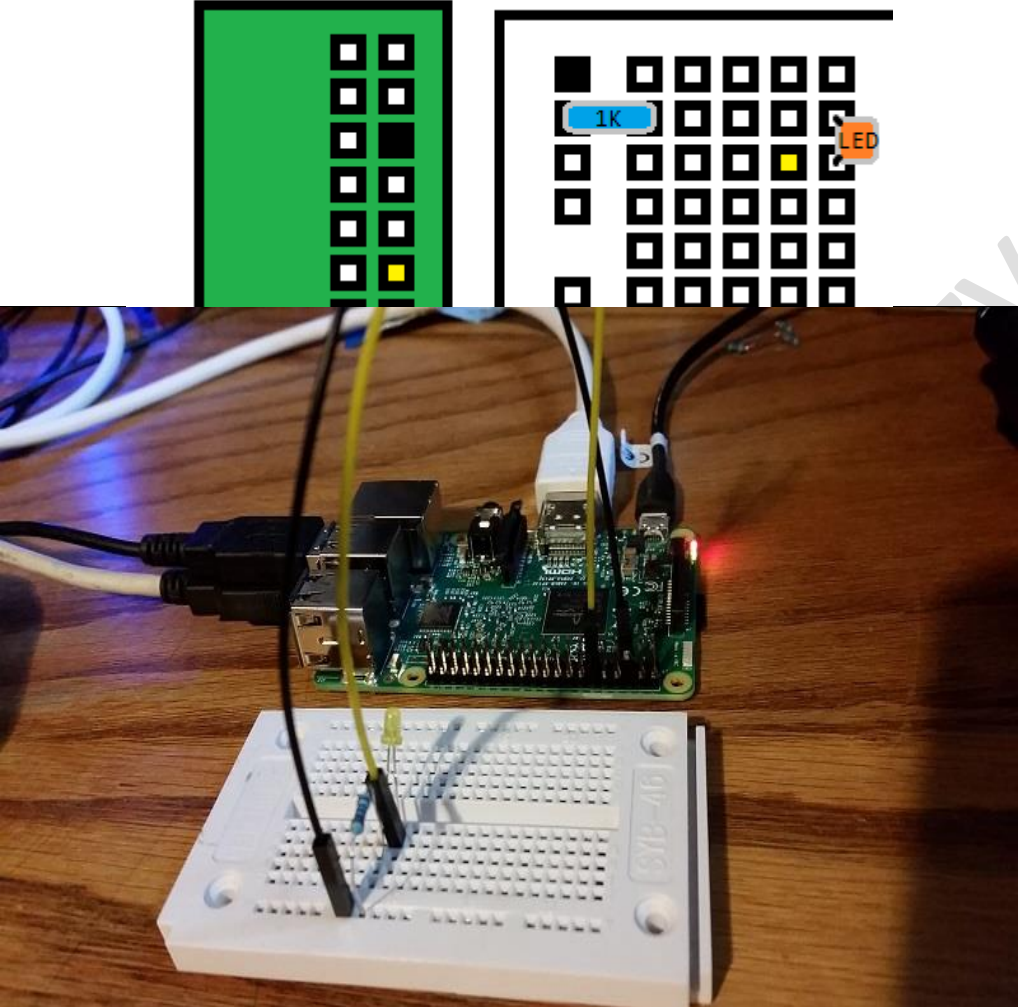
Sl no.	Equipment's Needed
1	Raspberry Pi
2	VGA cable
3	Power Cable
4	Jumper Cables

Step 1:	Understanding of Raspberry pi Pin out diagram and Bread board connections									
	<div><div><div>Available for GPIO if I2C is disabled using raspi-config</div><div><div>3.3V</div><div>GPIO2</div><div>GPIO3</div><div>GPIO4</div><div>GND</div><div>GPIO17</div><div>GPIO27</div><div>GPIO22</div><div>3.3V</div><div>GPIO10</div><div>GPIO9</div><div>GPIO11</div><div>GND</div><div>DNC</div><div>GPIO5</div><div>GPIO6</div><div>GPIO13</div><div>GPIO19</div><div>GPIO26</div><div>GND</div></div><div><div>5V</div><div>5V</div><div>GND</div><div>GPIO14</div><div>GPIO15</div><div>GPIO18</div><div>GND</div><div>GPIO23</div><div>GPIO24</div><div>GND</div><div>GPIO25</div><div>GPIO8</div><div>GPIO7</div><div>DNC</div><div>GND</div><div>GND</div><div>GPIO12</div><div>GND</div><div>GPIO16</div><div>GPIO20</div><div>GPIO21</div></div><div>Available for GPIO if serial is disabled using raspi-config</div></div><div><div>GPIO Availability:</div><div>No</div><div>Maybe</div><div>Yes</div></div><div><div>Pins above this line are present on all Raspberry Pi boards</div><div>Pins below line on Models A+, B+ and Pi 2</div></div></div>									



**Step 2:** Make the connections as follows




	
<b>Step 3:</b>	Write the python program to implement blinking LED
	<pre> import RPi.GPIO as GPIO import time  GPIO.setmode(GPIO.BCM)  GPIO.setup(18, GPIO.OUT) GPIO.output(18, GPIO.HIGH)  time.sleep(3)  GPIO.output(18, GPIO.LOW) GPIO.cleanup() </pre>

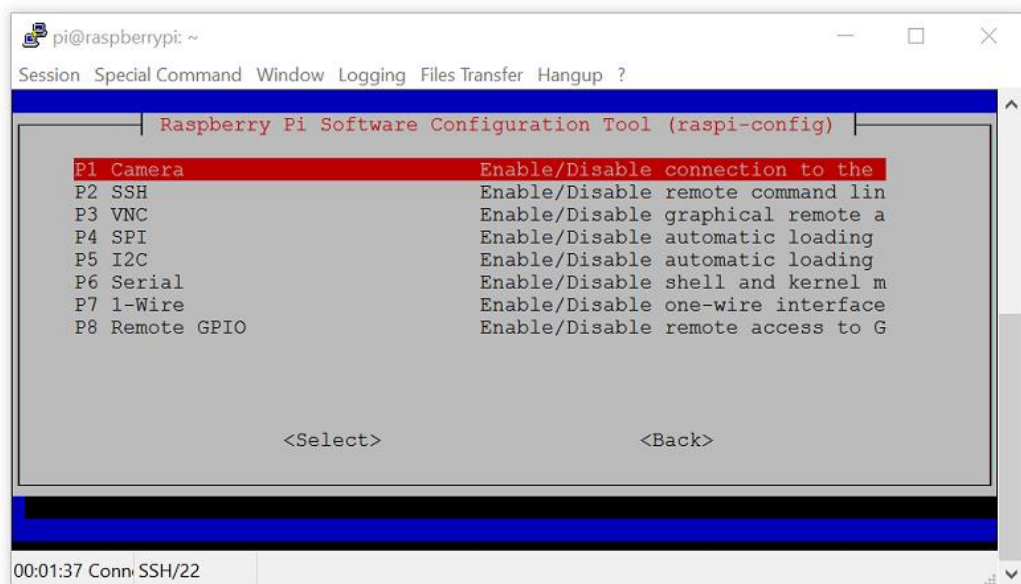
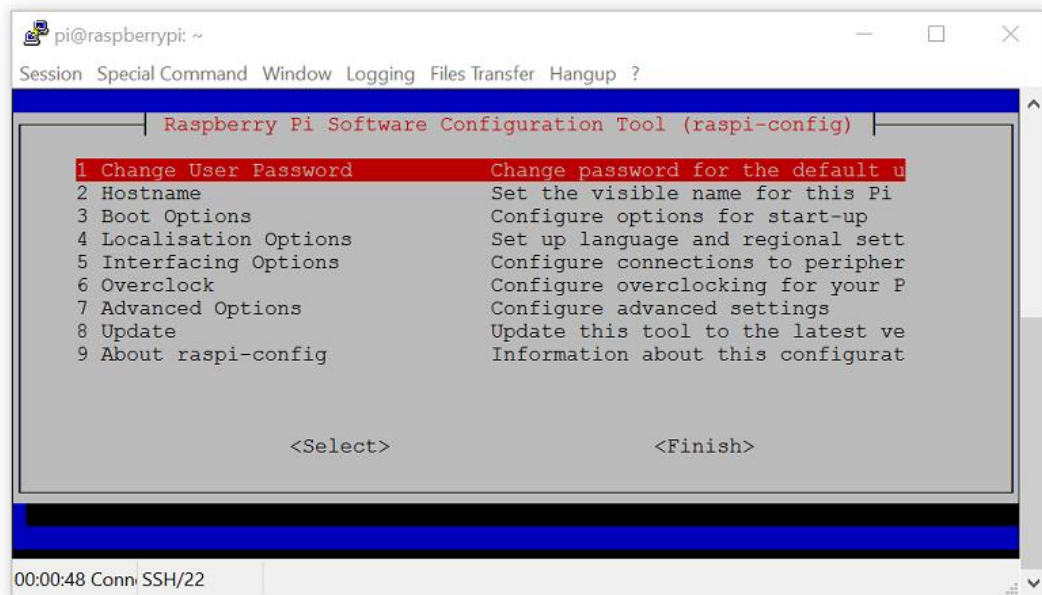


## Experiment No: 8


### Raspberry pi-camera module

Sl no.	Equipment's Needed
1	Raspberry Pi
2	Raspberry Pi Camera Module
3	VGA cable
4	Power Cable

<b>Step 1:</b>	<p>Open the notch “<b>Connect camera</b>” module to raspberry pi Blue should be facing towards Ethernet port.</p> 
<b>Step 2:</b>	<code>pi@raspberrypi:~ \$cd Desktop/</code>
<b>Step 3:</b>	<p>Enable Camera <code>pi@raspberrypi:~ /Desktop\$ sudo raspi-config</code> Interfacing option → Enable camera</p>



Yes to enable, and then go ahead and reboot:

	
<b>Step4:</b>	<p>Let's open the terminal next (control+alt+t) and do a \$ sudo apt-get install python3-picamera. You will likely find that you already have it, but we want to be sure.</p> <p>Now, in our cameraexample.py file:</p>
<b>Step 5:</b>	<p>Write the Python program</p> <pre>pi@raspberrypi:~ /Desktop\$ nano cameraexample.py</pre> <pre>import picamera import time  camera = picamera.PiCamera() camera.capture('example.jpg')  camera.vflip = True  camera.capture('example2.jpg')  camera.start_recording('examplevid.h264') time.sleep(5) camera.stop_recording()</pre>
<b>Step 6:</b>	<p>Command to view recorded video</p> <pre>\$ omxplayer examplevid.h264.</pre>



## Experiment No: 09

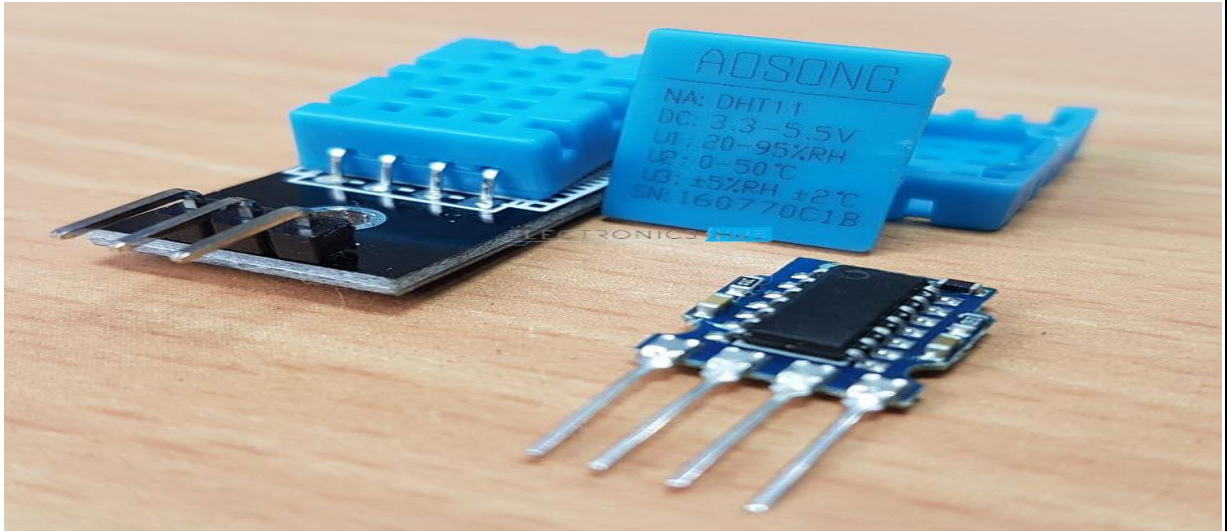
Raspberry pi program to implement temperature sensor

Sl no.	Equipment's Needed
1	Raspberry Pi Model 3
2	Raspberry Pi Camera Module
3	Digital humidity and temperature sensor-(DHT 22)
4	VGA cable
5	Power Cable

Step No.	Raspberry Pi DHT11 Humidity and Temperature Sensor Interface
1	<p><b>Raspberry Pi DHT11 Humidity and Temperature Sensor Interface</b></p> <p>In this project, we will learn about DHT11 Humidity and Temperature Sensor and how the Raspberry Pi DHT11 Humidity Sensor interface works. By Interfacing DHT11 Temperature and Humidity Sensor with Raspberry Pi, you can implement a basic IoT Project like a simple Weather Station.</p>  <p>Overview</p> <p>DHT11 is a Digital Sensor consisting of two different sensors in a single package. The sensor contains an NTC (Negative Temperature Coefficient) Temperature Sensor, a Resistive-type Humidity Sensor and an 8-bit Microcontroller to convert the analog signals from these sensors and produce a Digital Output.</p>

I have already worked with the DHT11 Sensor in my [DHT11 Humidity Sensor on Arduino](#) Project. In that project, I have mentioned the Pin Configuration of the DHT11 Sensor, how to interface it with a Microcontroller and how the digital Output from the DHT11 Sensor can be decoded.

So, I suggest you to refer to that project once for more information on DHT11 Humidity and Temperature Sensor. I'll explain a few thing which I have missed in the Arduino Project.



We know that the output from the DHT11 Sensor is Digital. But how exactly we can read this digital data?

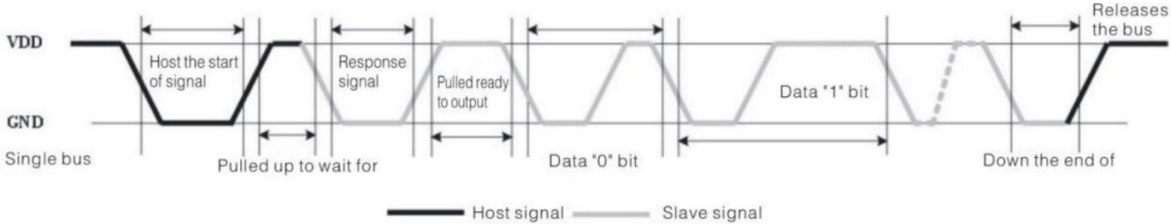
#### Reading Digital Output from DHT11

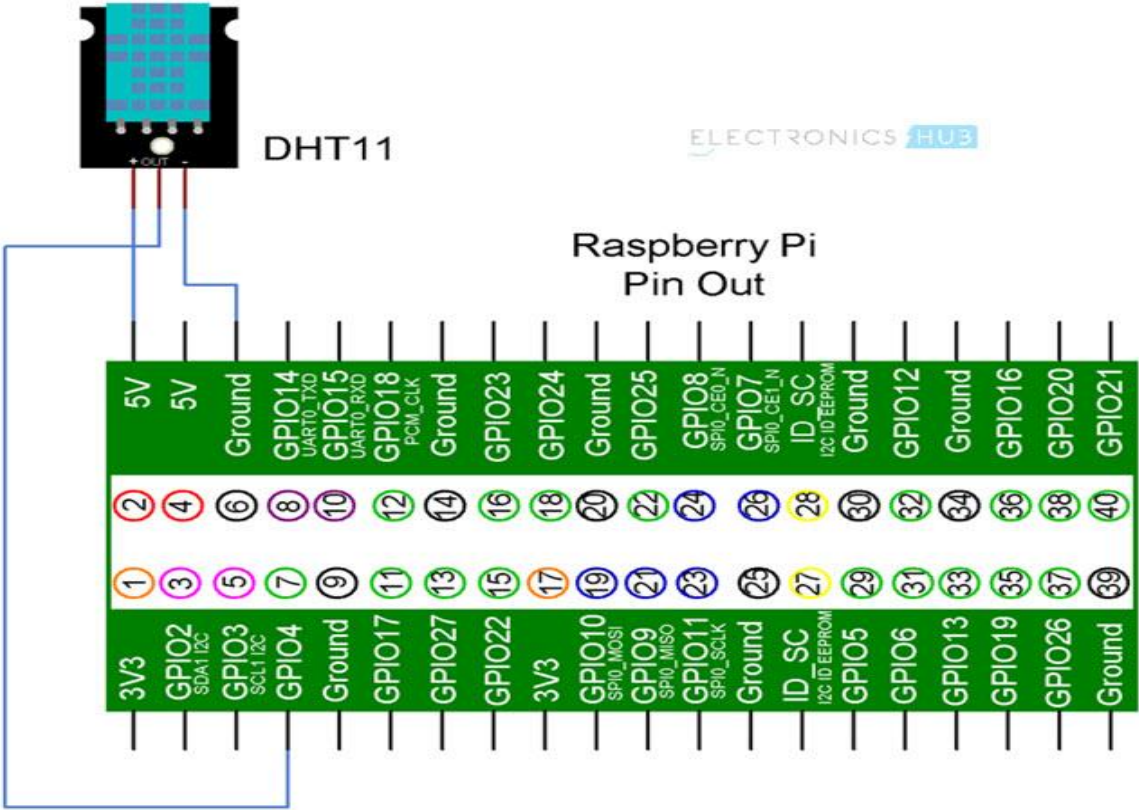
DHT11 uses a Single bus data format for communication. Only a single data line between an MCU like Arduino or Raspberry Pi and the DHT11 Sensor is sufficient for exchange of information.

In this setup, the Microcontroller acts as a Master and the DHT11 Sensor acts as a Slave. The Data OUT of the DHT11 Sensor is in open-drain configuration and hence it must always be pulled HIGH with the help of a 5.1K $\Omega$  Resistor.

This pull-up will ensure that the status of the Data is HIGH when the Master doesn't request the data (DHT11 will not send the data unless requested by the Master).

Now, we will see how the data is transmitted and the data format of the DHT11 Sensor. Whenever the Microcontroller wants to acquire information from DHT11 Sensor, the pin of the

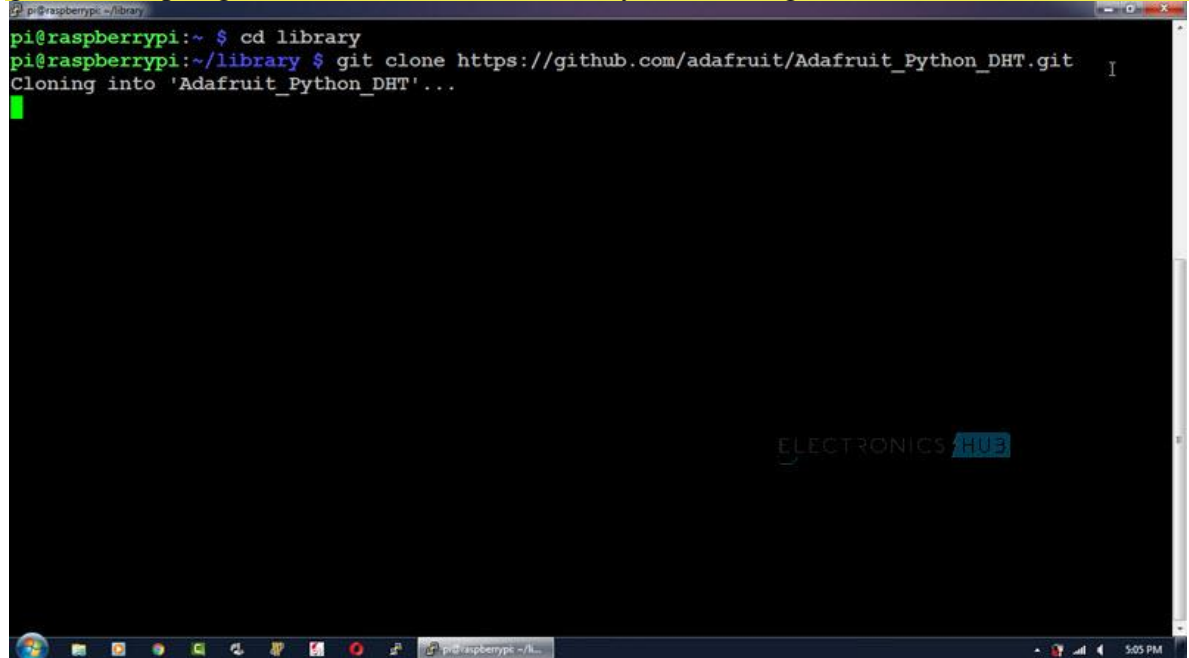
	 <p>Microcontroller is configured as OUTPUT and it will make the Data Line low for a minimum time of 18ms and releases the line. After this, the Microcontroller pin is made as INPUT.</p> <p>The data pin of the DHT11 Sensor, which is an INPUT pin, reads the LOW made by the Microcontroller and acts as an OUTPUT pin and sends a response of LOW signal on the data line for about 80μs and then pulls-up the line for another 80μs.</p> <p>After this, the DHT11 Sensor sends a 40 bit data with Logic '0' being a combination of 50μs of LOW and 26 to 28μs of HIGH and Logic '1' being 50μs of LOW and 70 to 80μs of HIGH.</p> <p>After transmitting 40 bits of data, the DHT11 Data Pin stays LOW for another 50μs and finally changes its state to input to accept the request from the Microcontroller.</p> <p><b>NOTE:</b> We have implemented this logic while programming the Arduino. But for Raspberry Pi, we used a library that takes care of all these things.</p> <p>Raspberry Pi DHT11 Humidity and Temperature Sensor Interface</p> <p>By interfacing the DHT11 Sensor with Raspberry Pi, you can build your own IoT Weather Station. All you need to implement such IoT Weather is a Raspberry Pi, a DHT11 Humidity and Temperature Sensor and a Computer with Internet Connectivity.</p>
2	<p><b>Circuit Diagram</b></p> <p>The following is the circuit diagram of the DHT11 and Raspberry Pi Interface.</p>

	 <p style="text-align: center;">Raspberry Pi Pin Out</p>
3	<p>Components Required</p> <ul style="list-style-type: none"> <li>• Raspberry Pi 3 Model B</li> <li>• DHT11 Temperature and Humidity Sensor</li> <li>• Connecting Wires</li> <li>• Power Supply</li> <li>• Computer</li> </ul>
4	<p>Circuit Design</p> <p>If you observe the circuit diagram, there is not a lot of stuff going on with respect to the connections. All you need to do is to connect the VCC and GND pins of the DHT11 Sensor to +5V and GND of Raspberry Pi and then connect the Data OUT of the Sensor to the GPIO4 i.e. Physical Pin 7 of the Raspberry Pi.</p>
5	<p>Installing DTH11 Library</p> <p>Since we are using a library called Adafruit_DHT provided by Adafruit for this project, we need to first install this library into Raspberry Pi.</p>

First step is to download the library from GitHub. But before this, I have created a folder called 'library' on the desktop of the Raspberry Pi to place the downloaded files. You don't have to do that.

Now, enter the following command to download the files related to the Adafruit\_DHT library.

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```



All the contents will be downloaded to a folder called 'Adafruit\_Python\_DHT'. Open this directory using `cd Adafruit_Python_DHT`. To see the contents of this folder, use 'ls' command.

In that folder, there is file called 'setup.py'. We need to install this file using the following command.

```
sudo python setup.py install
```

Code

As we are using the library Adafruit\_DHT for this project, there is nothing much to do in the Python Programming part. All you need to do is to invoke the library with the Sensor and GPIO Pin and print the values of Temperature and Humidity.

```
import sys
import Adafruit_DHT
import time
```

```
while True:
```

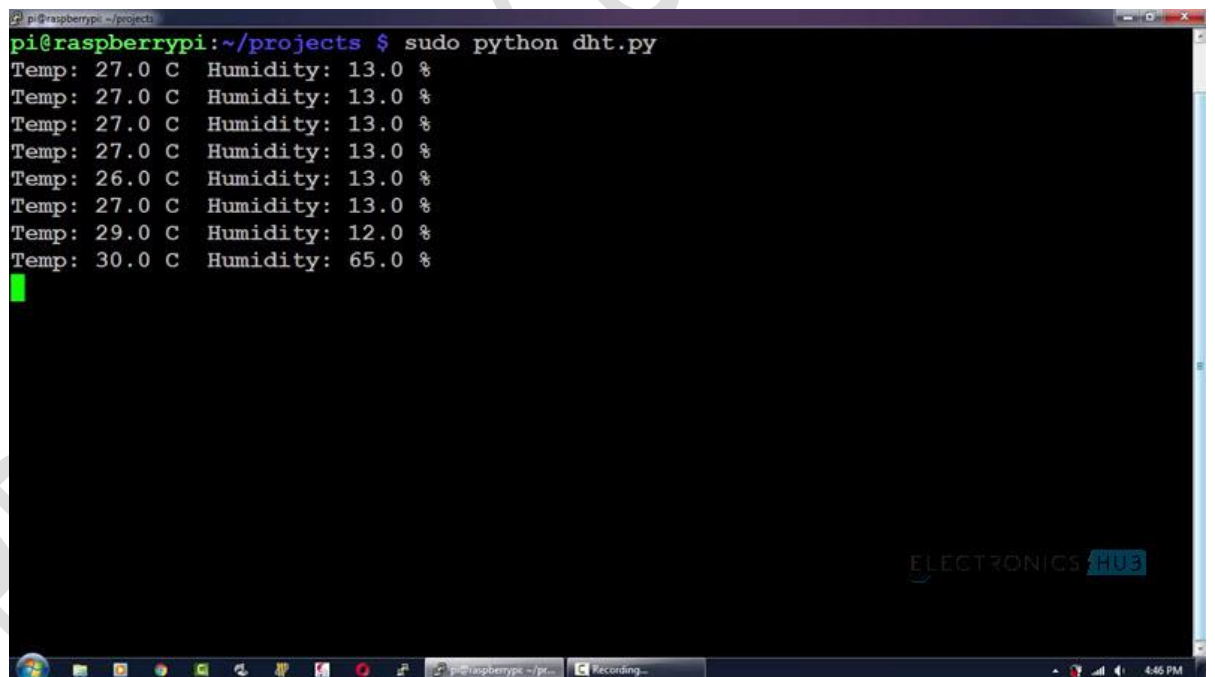
```
    humidity, temperature = Adafruit_DHT.read_retry(22, 4)
```

```
    print 'Temp: {0:0.1f} C Humidity: {1:0.1f} %'.format(temperature, humidity)
```

```
    time.sleep(1)
```

view [rawRaspberry Pi DHT 11.py](#) hosted with [by GitHub](#)  
Working

Make the connections as per the circuit diagram and install the library. Use the above python program to see the results.



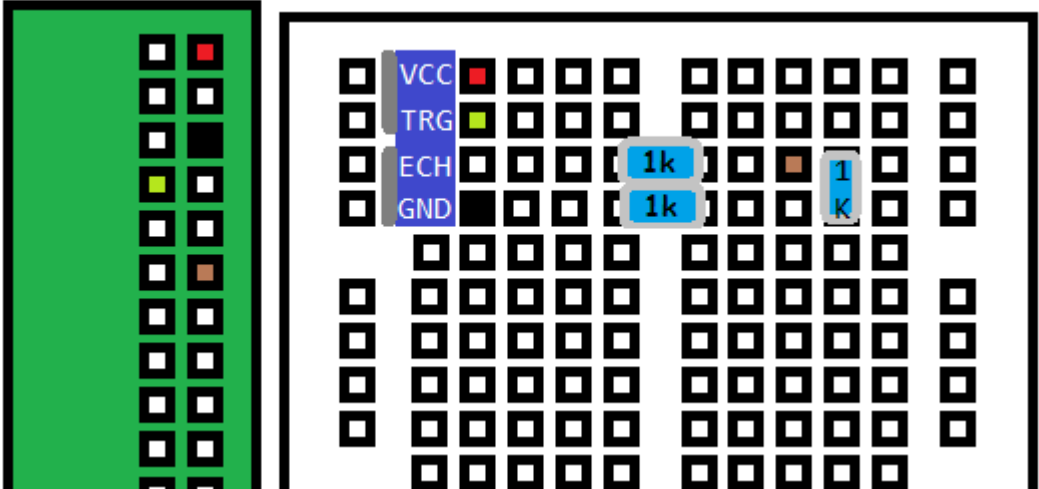
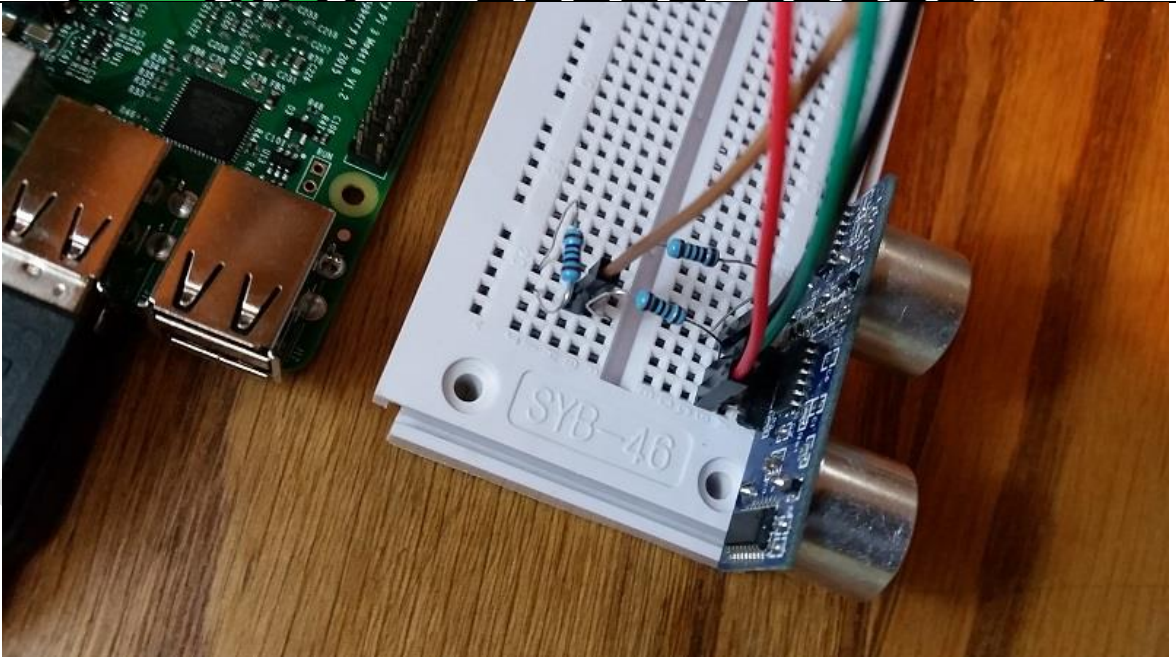
```
pi@raspberrypi:~/projects $ sudo python dht.py
Temp: 27.0 C Humidity: 13.0 %
Temp: 27.0 C Humidity: 13.0 %
Temp: 27.0 C Humidity: 13.0 %
Temp: 27.0 C Humidity: 13.0 %
Temp: 26.0 C Humidity: 13.0 %
Temp: 27.0 C Humidity: 13.0 %
Temp: 29.0 C Humidity: 12.0 %
Temp: 30.0 C Humidity: 65.0 %
```



## Experiment No: 10

Demonstration of Raspberry Pi with distance sensor (ultrasonic sensor HC-SR04)

Sl no.	Equipment's Needed
1	Raspberry Pi Model 3
2	Raspberry Pi ultrasonic sensor HC-SR04
3	VGA cable
4	Power Cable

Step 1:	Make the connections as follows
	
	
Step 2:	Open the terminal CTRL+ALT+T and type \$ Sudo nano Filename.py to open empty notepad
Step 3:	Write the python program to implement use of raspberry pi with temperature sensor
	<pre>import RPi.GPIO as GPIO</pre>

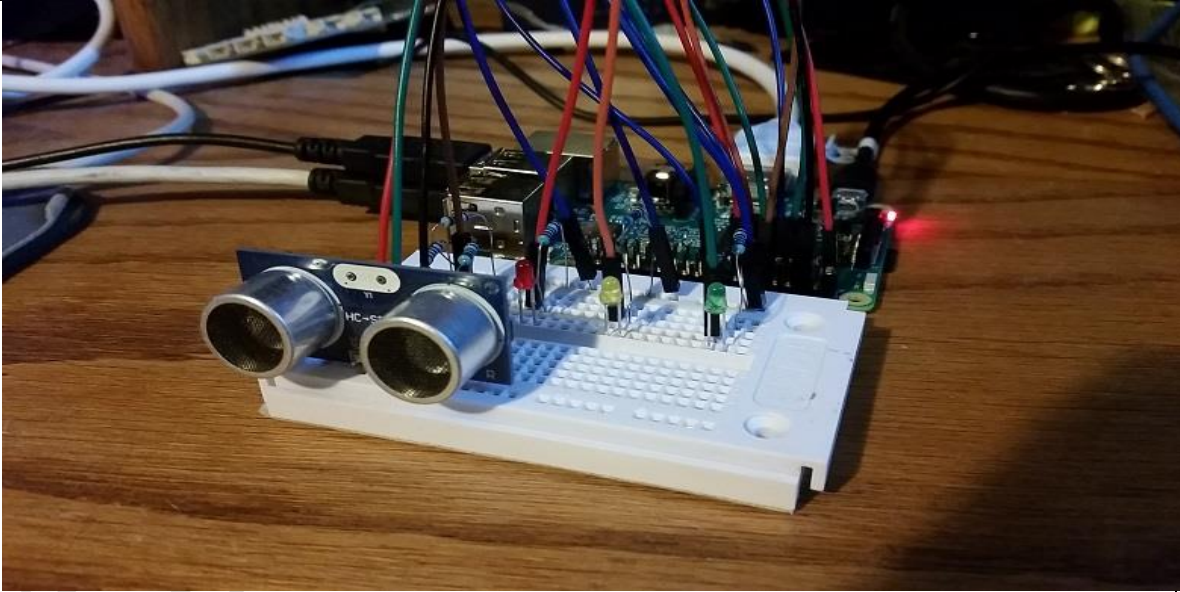
	<pre> import time  GPIO.setmode(GPIO.BCM)  TRIG = 4 ECHO = 18  GPIO.setup(TRIG,GPIO.OUT) GPIO.setup(ECHO,GPIO.IN)  GPIO.output(TRIG, True) time.sleep(0.00001) GPIO.output(TRIG, False)  while GPIO.input(ECHO) == False:     start = time.time()  while GPIO.input(ECHO) == True:     end = time.time()  sig_time = end-start  #CM: distance = sig_time / 0.000058  #inches: #distance = sig_time / 0.000148  print('Distance: { } centimeters'.format(distance))  GPIO.cleanup() </pre>
Step 3:	<p>Compile the program using command \$ Python Filename.py</p> <p>And verify the output</p>

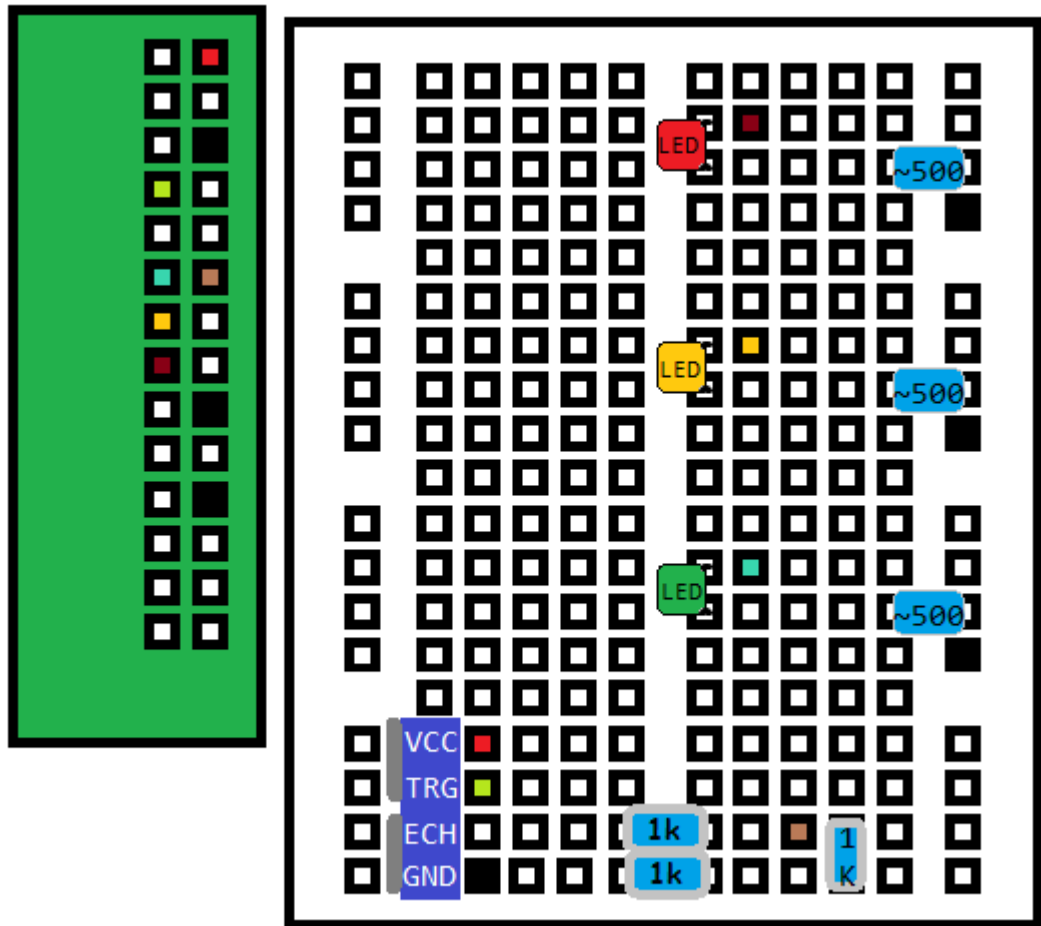


## Experiment No: 11

Raspberry pi program to implement Garage spot light.

Sl no.	Equipment's Needed
1	Raspberry Pi Model 3
2	Raspberry Pi ultrasonic sensor HC-SR04
3	VGA cable
4	Power Cable
5	Jumper cables- Male to Female
6	3 different colored LEDs-RGB

<b>Step 1:</b>	The idea of a garage stop-light is to show green when you have plenty of room to pull your car forward in your garage, and then turn yellow as you approach the fully forward position, and then red when you should stop. We're going to build this system with our Raspberry Pi, and use some distances that we can easily test.
	
<b>Step 2:</b>	To start, we'll just leave the distance sensor hooked up as is, but now we're going to add the light circuits, only this time we have three. Here's how I set mine up:



**Step 3:** Same as in the previous tutorials, the filled in boxes are the jumper wires by color, and the boxes themselves are the pins on the pi or the openings on the breadboard. The filled in blue blobs are the resistors, the leds are labeled. Black boxes are ground.

Once you have everything hooked up and you're confident you've not messed anything up, you can power on your Raspberry Pi. You may want to re-reference the Adafruit image shared earlier:

	<div> <div>Available for GPIO if I2C is disabled using raspi-config</div> <div>Used by DotStars GPIO unavailable</div> <div>           GPIO Availability:  <div>No</div> <div>Maybe</div> <div>Yes</div> </div> </div> <div> <div>3.3V</div> <div>GPIO2</div> <div>GPIO3</div> <div>GPIO4</div> <div>GND</div> <div>GPIO17</div> <div>GPIO27</div> <div>GPIO22</div> <div>3.3V</div> <div>GPIO10</div> <div>GPIO9</div> <div>GPIO11</div> <div>GND</div> <div>DNC</div> <div>GPIO5</div> <div>GPIO6</div> <div>GPIO13</div> <div>GPIO19</div> <div>GPIO26</div> <div>GND</div> </div> <div> <div>5V</div> <div>5V</div> <div>GND</div> <div>GPIO14</div> <div>GPIO15</div> <div>GPIO18</div> <div>GND</div> <div>GPIO23</div> <div>GPIO24</div> <div>GND</div> <div>GPIO25</div> <div>GPIO8</div> <div>GPIO7</div> <div>DNC</div> <div>GND</div> <div>GPIO12</div> <div>GND</div> <div>GPIO16</div> <div>GPIO20</div> <div>GPIO21</div> </div> <div>Available for GPIO if serial is disabled using raspi-config</div>
--	---

Pins above this line are present on all Raspberry Pi boards

Pins below line on Models A+, B+ and Pi 2

```
GREEN = 17
```

```
YELLOW = 27
```

```
RED = 22
```

```
GPIO.setup(TRIG,GPIO.OUT)
```

```
GPIO.setup(ECHO,GPIO.IN)
```

```
GPIO.setup(GREEN,GPIO.OUT)
```

```
GPIO.setup(YELLOW,GPIO.OUT)
```

```
GPIO.setup(RED,GPIO.OUT)
```

```
def green_light():
```

```
    GPIO.output(GREEN, GPIO.HIGH)
```

```
    GPIO.output(YELLOW, GPIO.LOW)
```

```
    GPIO.output(RED, GPIO.LOW)
```

```
def yellow_light():
```

```
    GPIO.output(GREEN, GPIO.LOW)
```

```
    GPIO.output(YELLOW, GPIO.HIGH)
```

```
    GPIO.output(RED, GPIO.LOW)
```

```
def red_light():
```

```
    GPIO.output(GREEN, GPIO.LOW)
```

```
    GPIO.output(YELLOW, GPIO.LOW)
```

```
    GPIO.output(RED, GPIO.HIGH)
```

```
def get_distance():
```

```

GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)

while GPIO.input(ECHO) == False:
    start = time.time()

while GPIO.input(ECHO) == True:
    end = time.time()

sig_time = end-start

#CM:
distance = sig_time / 0.000058

#inches:
#distance = sig_time / 0.000148
#print('Distance: { } centimeters'.format(distance))

return distance

while True:
    distance = get_distance()
    time.sleep(0.05)
    print(distance)

    if distance >= 30:
        green_light()
    elif 30 > distance > 10:
        yellow_light()
    elif distance <= 10:

```

	<div data-bbox="401 140 553 186" data-label="Text"> <pre>red_light()</pre> </div>
	<div data-bbox="336 260 456 306" data-label="Section-Header"> <h2>Output</h2> </div>
	<div data-bbox="355 317 1495 955" data-label="Image"> </div>
	<div data-bbox="336 970 600 1016" data-label="Text"> <p>Hand getting close:</p> </div>
	<div data-bbox="355 1029 1495 1667" data-label="Image"> </div>



