**MANAV RACHNA INTERNATIONAL**

**INSTITUTE OF REASEARCH & STUDIES**

**FACULTY OF ENGINEERING AND TECHNOLOGY**



**PRACTICAL FILE**

**FOR**

VIth **SEMESTER (ACADEMIC YEAR** 2020-2021**)**

**SUBJECT NAME:** Internet of Things Lab

**SUBJECT CODE:** BCS-DS-653

|  |  |
| --- | --- |
| **SUBMITTED BY:-** | **SUBMITTED TO: -** |
| **STUDENT NAME:** Sapna Sinha | **FACULTY NAME:** Dr. Savita |
| **ROLL NO.:** 1/18/FET/BCS/079 | **DESIGNATION:** Associate Professor |
| **BRANCH:** B. Tech | **DEPARTMENT:** FET  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **SECTION:** 6CSB |

# Experiment1

**Aim:** (a) Introduction to Arduino,

1. Setup software for Arduino
2. Simulator software.

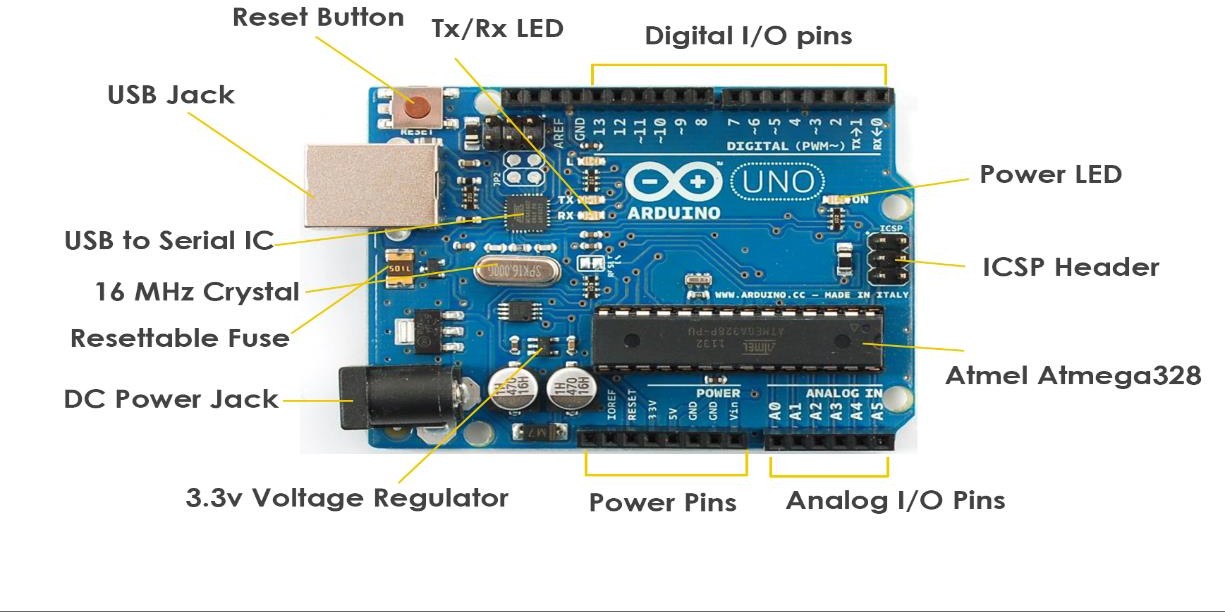
**Arduino** is an [open-source hardware](https://www.ebay.com/itm/1PCS-Slave-HC-06-Wireless-Bluetooth-Transeiver-RF-Master-Module-for-Arduino/201415076859) and [software](https://en.wikipedia.org/wiki/Open-source_software) company, project and user community that designs and manufactures [single-board microcontrollers](https://www.visuino.com/) and [microcontroller](http://en.wikipedia.org/wiki/Microcontroller) kits for building digital devices and interactive objects that can sense and control objects in the physical and digital world.

A pre-assembled Arduino board includes a microcontroller, which is programmed using Arduino programming language and the Arduino development environment. Arduino consists of both a physical programmable circuit board (often referred to as a [microcontroller](http://www.labcenter.com/products/vsm/arduino.cfm)) and a piece of [software,](http://arduino.cc/en/Main/Software) or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. In essence, this platform provides a way to build and program electronic components. Arduino programming language is a simplified from of C/C++ programming language based on what Arduino calls "sketches," which use basic programming structures, variables and functions. These are then converted into a C++ program. We can also use Python and JavaScript as a programming language for Arduino.

The hardware consists of a board designed around an 8-bit microcontroller, or a 32-bit ARM. Current models feature things like a USB interface, analog inputs, and GPIO pins which allows the user to attach additional boards.

Other open-source electronics prototyping projects, such as Wiring and Processing, provide the underpinnings for Arduino technology. Google Android Open Accessory Development Kit is also based on Arduino.

# Component of Arduino:



**Benefits of using Arduino:**

* Inexpensive
* Cross-Platform
* Clear programming Environment
* Open source and extensible software
* Open source and extensible hardware

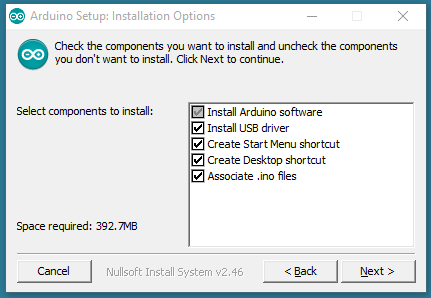
# Different Arduino Boards:



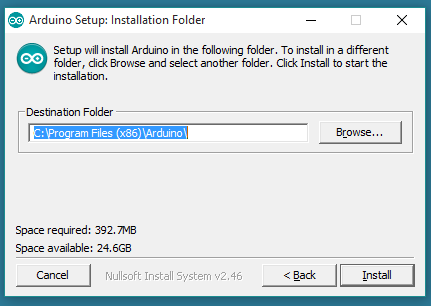
1. **Install the Arduino Software (IDE) on Windows PCs**

**Step-1**: Download the Arduino Software (IDE)

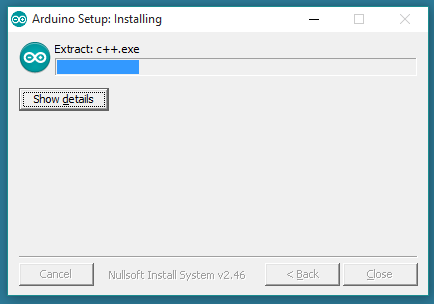
**Step-2:** When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



**Step-3:** Choose the components to install



**Step 4:** Choose the installation directory (keep the default one)



The process will extract and install all the required files to execute properly the Arduino Software (IDE)

There are many simulator or emulator software for Arduino such as, [Simulator for Arduino](https://www.arduino.cc/en/Reference/AnalogRead) (Paid), [simavr](https://github.com/buserror/simavr) (OSS), [emulare](http://www.virtualbreadboard.com/) (OSS), [Atmel Studio](http://arduinodev.com/codeblocks/) (Free), [emulino](https://github.com/ghewgill/emulino) (OSS), [Proteus VSM for Arduino](http://emulare.sourceforge.net/) [AVR](http://www.atmel.com/) (Paid), [simuino](http://web.simuino.com/home-1) (Free), [CodeBlocks Arduino IDE](https://en.wikipedia.org/wiki/Single-board_microcontroller) (Free), [Arduino](https://github.com/Paulware/ArduinoDebugger/) [Debugger/Simulator](https://en.wikipedia.org/wiki/Microcontroller) (OSS), [Virtual Breadboard](https://en.wikipedia.org/wiki/Open-source_hardware), [123d circuits](https://123d.circuits.io/) (Free), [Visuino](http://www.labcenter.com/products/vsm/arduino.cfm) (Paid)

# Experiment 2:

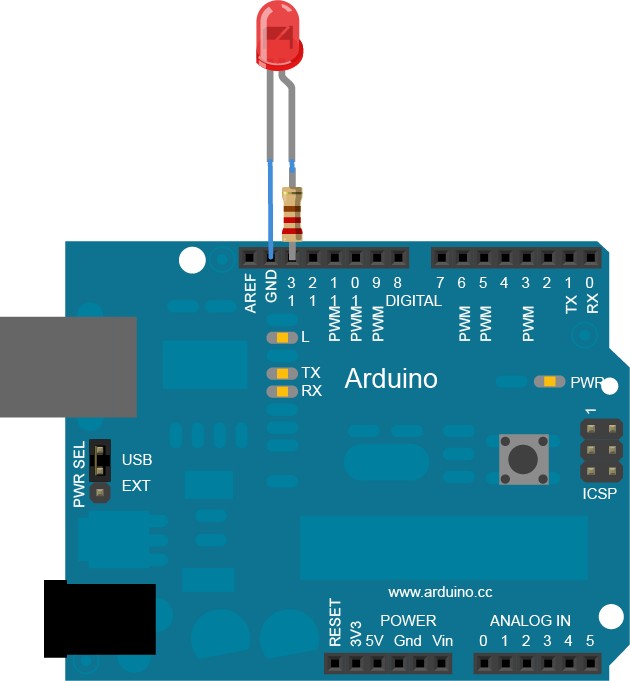
**Aim:** Blink LED light for particular duration (say 10 sec) through Arduino.

# Hardware Required,

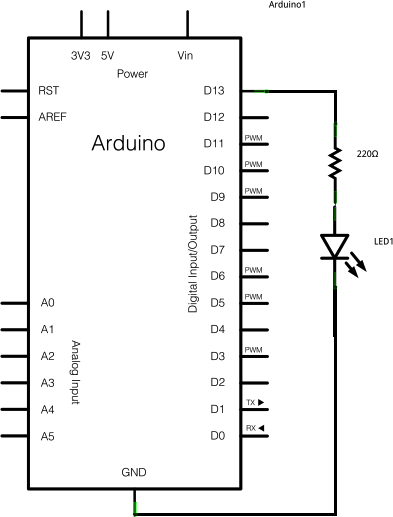
* Arduino
* LED
* 220 ohm Resistor

# Circuit

The LED is connected to a digital pin and its number may vary from board type to board type. Here we connected the LED to 13th pin of Arduino. Connect the long leg of the LED (the positive leg, called the anode) to the other end of the resistor. Connect the short leg of the LED (the negative leg, called the cathode) to the GND. The value of the resistor in series with the LED may be of a different value than 220 ohm; the LED will lit up also with values up to 1K ohm.

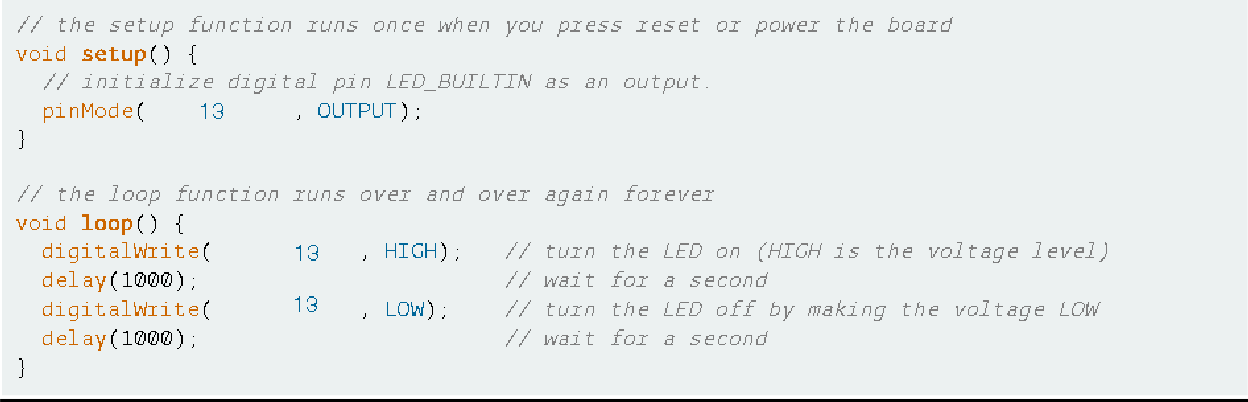


# Schematic



**Code**

After you build the circuit plug your Arduino board into your computer, start the Arduino Software (IDE) and enter the code below. You may also load it from the menu File/Examples/01.Basics/Blink.



# Experiment 3:

**Aim:** To note down the voltage through potentiometer and Arduino.

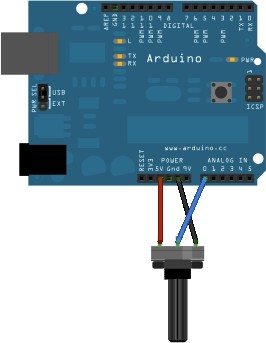
A potentiometer is a simple knob that provides a variable resistance, which we can read into the Arduino board as an analog value.

# Hardware Required,

* Arduino
* 10k ohm potentiometer

# Circuit

* Connect the three wires from the potentiometer to your board. The first goes to ground from one of the outer pins of the potentiometer. The second goes to 5 volts from the other outer pin of the potentiometer. The third goes from the middle pin of the potentiometer to analog input 0.

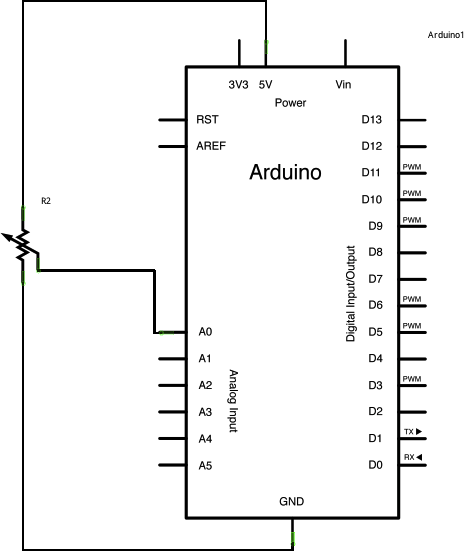


* By turning the shaft of the potentiometer, we change the amount of resistance on either side of the wiper which is connected to the center pin of the potentiometer. This changes the voltage at the center pin.

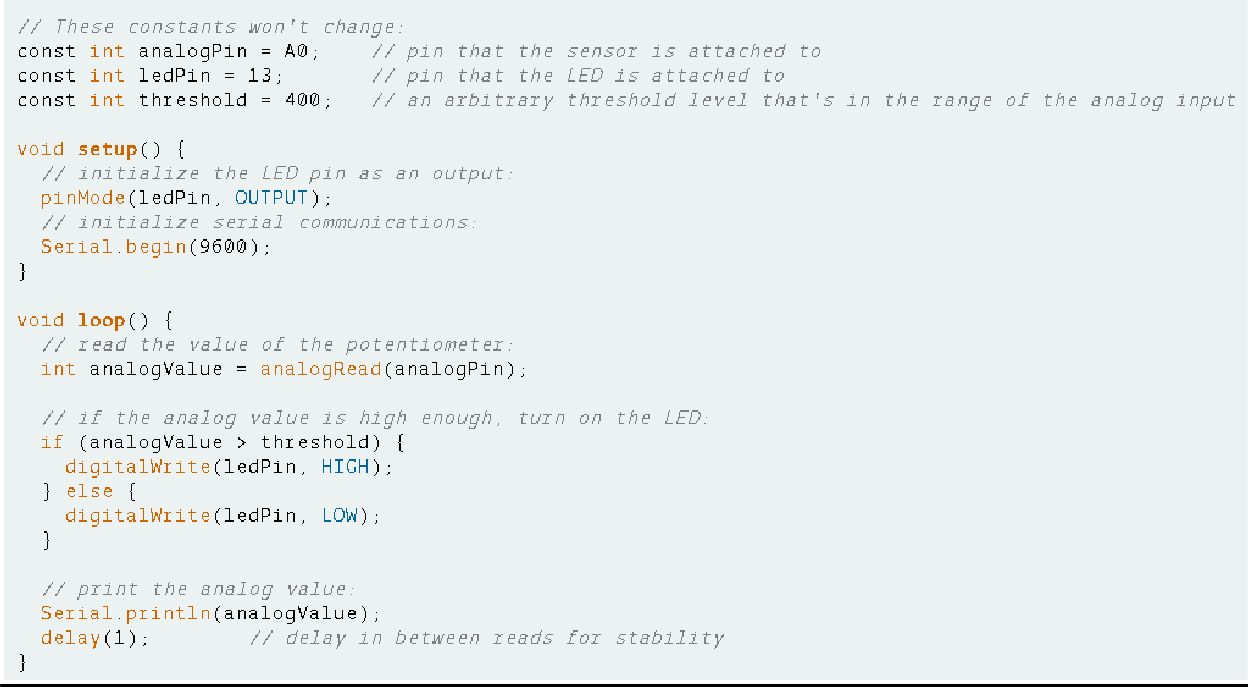
When the resistance between the center and the side connected to 5 volts is close to zero (and the resistance on the other side is close to 10 kilohms), the voltage at the center pin nears 5 volts. When the resistances are reversed, the voltage at the center pin nears 0 volts, or ground. This voltage is the analog voltage that you're reading as an input.

* The microcontroller of the board has a circuit inside called an *analog-to-digital converter* or *ADC* that reads this changing voltage and converts it to a number between 0 and 1023. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and the input value is 0. When the shaft is turned all the way in the opposite direction, there are 5 volts going to the pin and the input value is 1023. In between, [analogRead](https://github.com/Paulware/ArduinoDebugger/)() returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin. A threshold value is set if value cross above threshold LED will off and when it lies between the range, LED will switch on.

# Schematic



**Code**



**Experiment 4:**

**Aim:** To take the voltage reading into the phone through Bluetooth and Arduino.

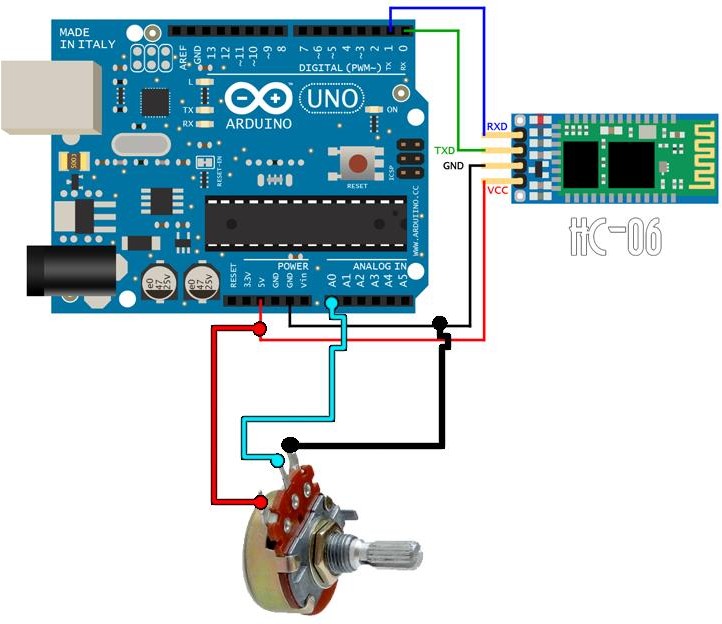
# Hardware Required,

* Arduino
* HC-06

In the experiment, we will read the analog value of a potentiometer between 0 and 5 volts and save the data in a float format so it could have decimals as well. Next we establish a Bluetooth connection with the smartphone and send the read data in a text format. We are using Bluetooth SPP Manager app to take the reading.

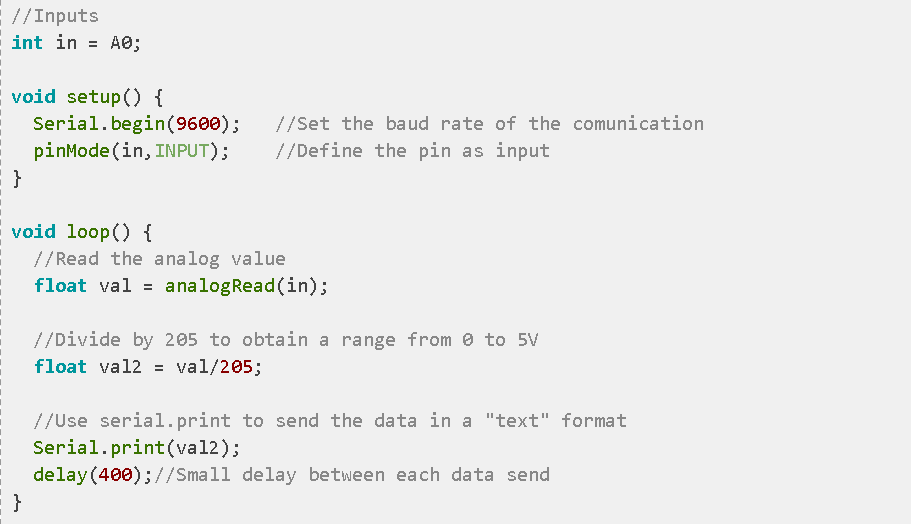
# Circuit

The potentiometer is connected to 5V and GND and the middle pin to the analog input A0 of the Arduino. The Bluetooth module is [the HC06](http://virtronics.com.au/Simulator-for-Arduino.html?epid=817572491&hash=item2ee5462bfb%3Ag%3A6GgAAOxyCQ5TlsHx) and it has a wart communication so it will use the Tx and Rx pins.



Make sure Bluetooth connection between the Arduino and the phone. After establishing connection open the app to see the reading.

# Code:



**Experiment 5**

**Aim: Send the data stored in local database in the cloud and get a response from the cloud to ensure that data is sent.**

**What this exercise is about**

Establish a secure and consistent communication from Pi to the cloud

# What you should be able to do

At the end of this exercise, you should be able to:

-- Know how to setup a Cloud service

-- Know how to establish communication from Pi to the Cloud

-- Transmit and Receive information from Pi to the Cloud

-- Store sensor data to the Cloud

# Introduction

This exercise will enable you to establish basic connectivity from Pi hardware to the Cloud service.

# Requirements

**--** Raspberry Pi headless system in setup

-- Cloud service has been setup

-- Reliable connection between Cloud and Pi is available

# Steps:

Step 1: Connect with the Cloud server and establish a reliable connection with it.

Step 2: Read the sensor data from SQLite (local store) and transmit to the Cloud service Step 3: Get the acknowledgement from the cloud

Step 4: On the cloud platform, store to the database and show simple graphs on the website Step 5: As you keep receiving data from the Pi hardware, display a time-series data on website.

# Experiment -6

**AIM: -**Setup the Raspberry Pi on your local Wireless network and create a headless connection to Pi

# What this exercise is about

To setup the Raspberry Pi system to enable us to write IoT applications on it.

# What you should be able to do

* 1. Download all the required software on your laptop
  2. Download Raspbian OS image on SB card
  3. Boot up Raspberry Pi hardware
  4. Setup a headless Pi connection using your computer
  5. Setup wireless networking on Pi
  6. Connect via ssh to Pi

# Introduction

This exercise will enable you to start working with Raspberry Pi as the controller for ultimately creating an IoT solution to monitor the Air Conditioner, and be a part of the larger Smart Home System.

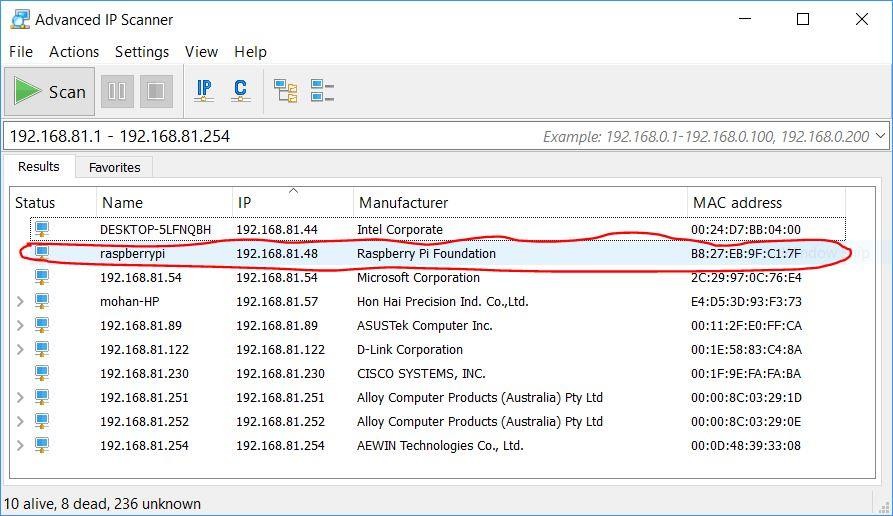
# Requirements

1. All requirements from Exercise 1 are the same and valid for this exercise
2. Bring Raspberry Pi 3 module to the lab
3. A computer with Wi-Fi capabilities
4. A Wireless network to connect with
5. Temperature sensors foe the solution
6. Micro SD card for the Raspbian image

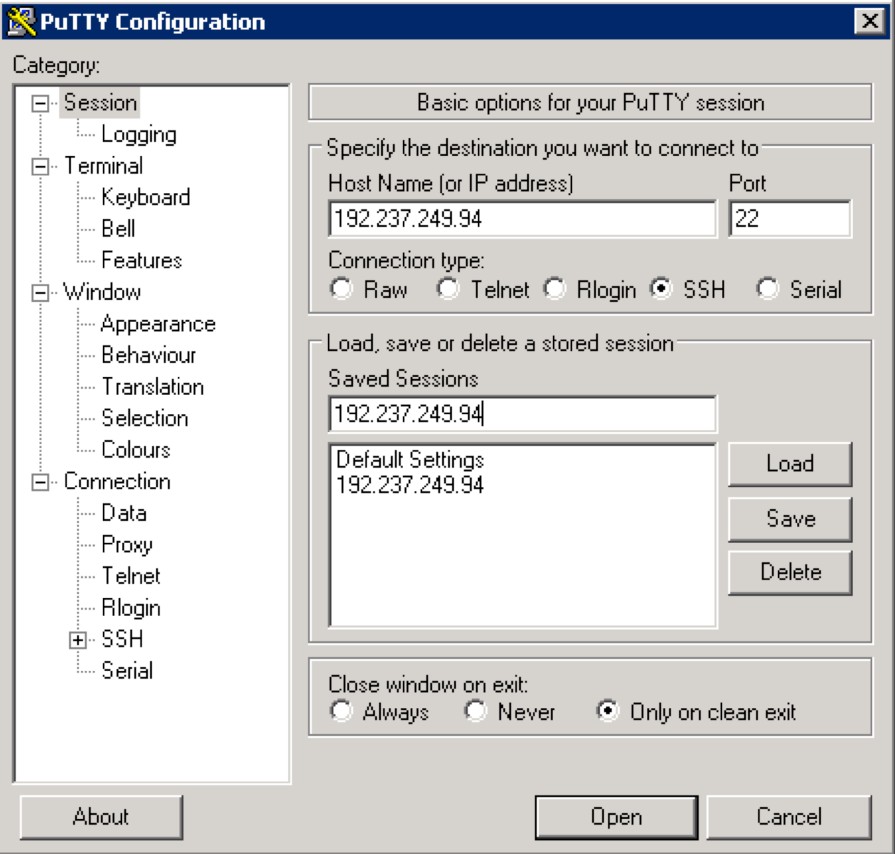
# Steps:

Step 1: Install the following programs on your computer

* + Advanced IP Scanner

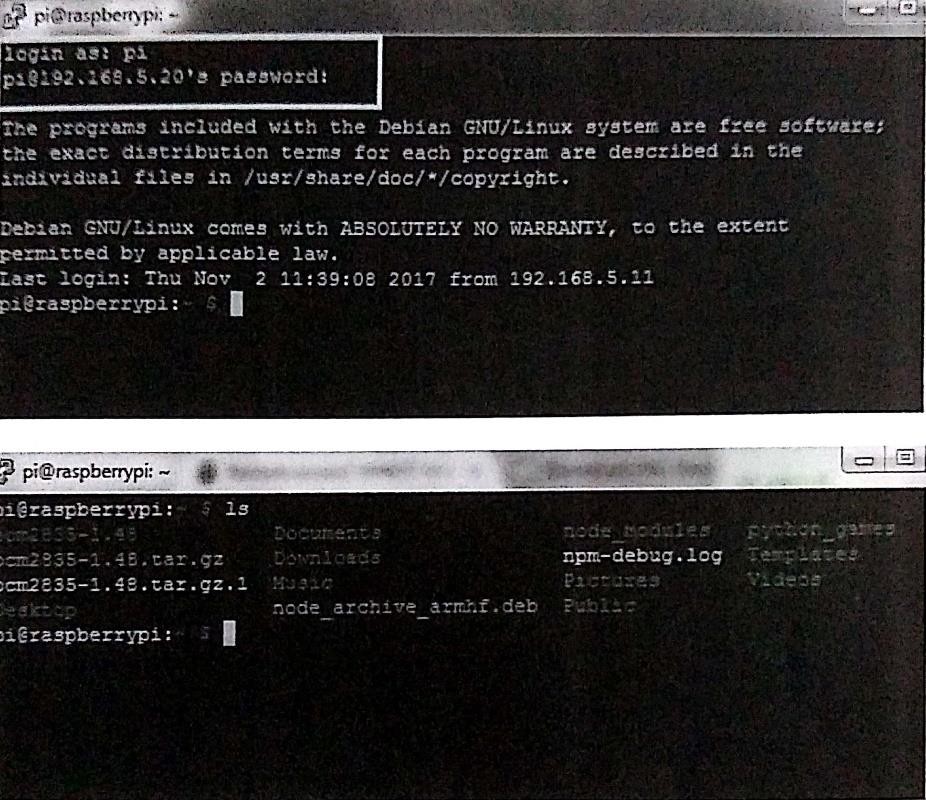


* + Putty SSH client

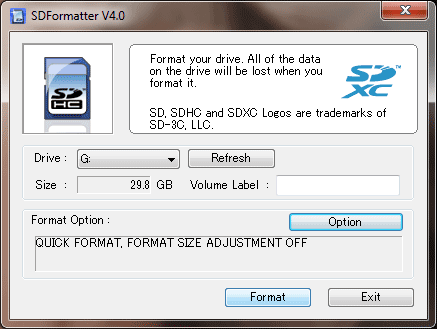


* + Default RaspberryPi **login: pi**

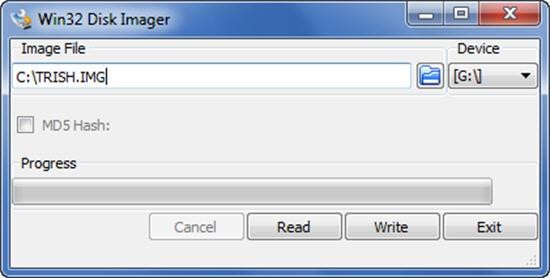
# Password: raspberry



* + SD Card formatting utility



* + Win32 Disk Imaging utility



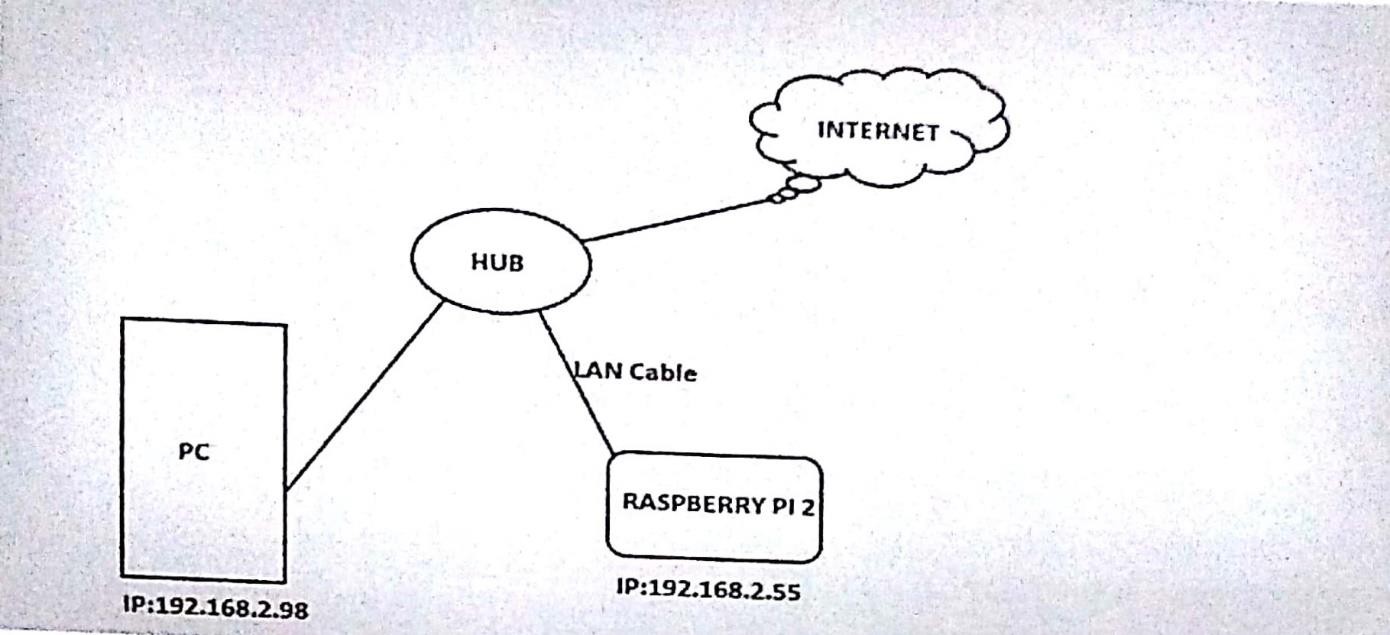
* + Raspbian OS image from the Pi website

Step 2: Download the Raspbian image to the SD card

Step 3: Power up the Pi hardware using USB charger from your phone Step 4: Connect to your laptop via Wired Ethernet

Step 5: Enable Wi-Fi on Raspberry Pi. Remove the Wired Ethernet and confirm that it works via Wi-Fi

Step 6: Establish putty SSH connection from your laptop



Step 7: Download all the required updatesfor Pi and get it ready for the next step

# Experiment-7

**AIM: Connect all the required sensors and write the application on PI What this exercise is about**

To setup the Raspberry Pi System and write our application programs on it connecting to sensors and getting the data from it.

# What you should be able to do

At the end of this exercise, you should be able to:

* + Setup the breadboard and establish connections with sensors
  + Use the GPIOs of PI and connect to the sensors
  + Write a program to read data from the sensors
  + Store the data to a local database on Pi

# Introduction

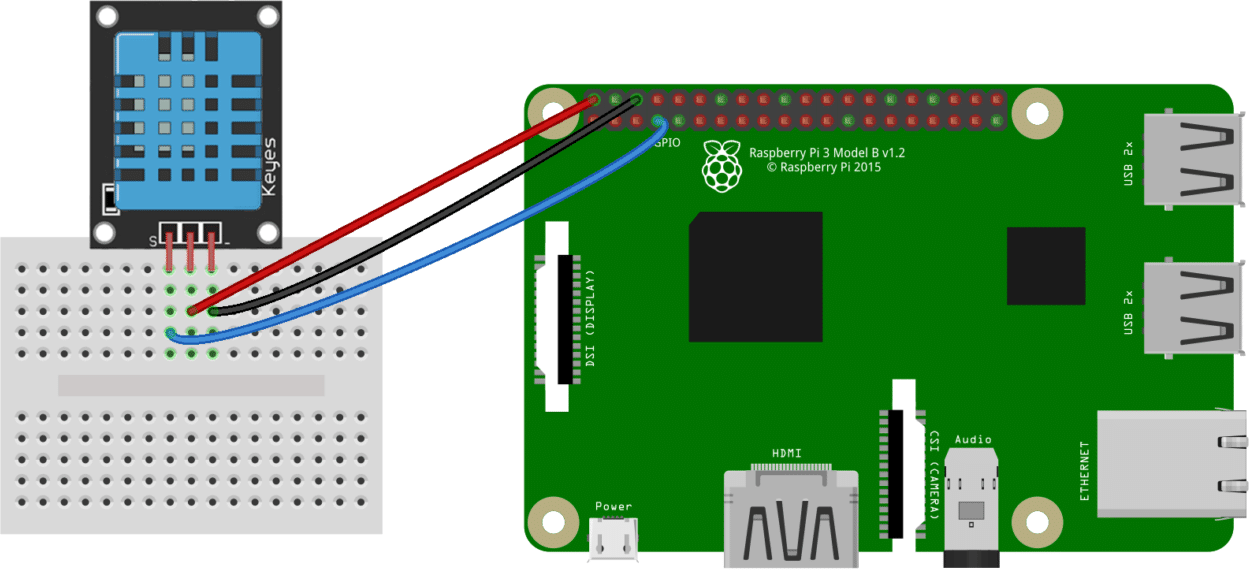
This exercise will enable you to start working with Raspberry Pi as the controller and connect with a few sensors, read data from them and store locally on the Pi hardware.

# Requirements

* + Raspberry Pi headless system is setup
  + SSH connection to Pi is up and running
  + All sensors are working and available for our first set of tests
  + Breadboards, wires, power supplies and any other connectors required for our program
  + Connection with sensor and hardware is established
  + Write the program on the Pi to read data from sensors
  + Local database on the Pi is setup

# Steps

**Step 1:** Read the technical specs of the Temperature sensors, and wire them correctly to the Pi hardware



**Step 2:** Establish simple connectivity with the sensors by writing a basic program to read data from the sensors

**Step 3:** Create a simple schema to store the data on the local database - we suggest that you use SQLite as the data store

**Step 4:** Using the schema, connect to SQLite read the data from the sensors and store them locally on the SD card

**Step 5:** Read back from the database to ensure that the data integrity is maintained

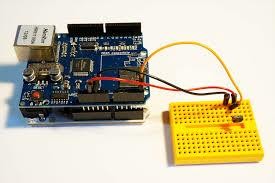
# Experiment 8

**Aim:** Read local temperature and humidity and compare it with your city’s historical averages.

# Requirements

* Raspberry Pi has been setup with Raspbian image and is booted up
* One temperature and one relative humidity sensor
* Breadboard for connections
* Python libraries and IDLE UI has been installed on Pi

It will enable you to connect with multiple sensors and correlate the data with your local city’s historical averages.



# Steps

1. Download the latest Python image to Raspberry Pi.
2. Download all the required libraries to establish connection with Cloud.
3. Write a program to read data from temperature and humidity sensor that are connected locally to Pi.
4. Store the data to your local database (MySQL or SQLite).
5. Connect to the weather services via the weather APIs.
6. Compare the data that you are reading from your sensors with the city’s historical averages.
7. Display the output on the local HDMI display output.