**COOK BOOK**

**Installation of Raspbian on Raspberry pi 3**

1. The raspbian is installed on 8 GB memory card. To install raspbian first step is to format the memory card in order to make it suitable for installation. To format memory card SD Memory card format tool 5.0 is used. Steps to format the memory card are as follows:

1. Download SD Memory card format tool from www.sdcard.org\downloads\.

2. Install the memory card formatting tool.

3. Format memory card as FAT.

4. Once formatting is completed, memory card is ready for use.

Once the memory card is formatted, it is ready for use. The next step is to download the operating system either raspbian or noobs from official website <https://www.raspberrypi.org/downloads/>. After downloading the system image of raspbian extract the zip file to some folder. Next step is to write this downloaded system image to memory card. For this purpose Etcher tool is used. Steps to write system image to memory card are as follows:

1. Download Etcher and install it from https://etcher.io/.

2. Connect an SD card reader with the SD card inside.

3. Open Etcher and select from your hard drive the Raspberry Pi .img or .zip file you wish to write to the SD card.

4. Select the SD card you wish to write your image to.

5. Review your selections and click 'Flash!' to begin writing data to the SD card.

Once the system image is written to memory card, insert that memory card into raspberry pi and power on the raspberry pi. On the first boot it will take you through all the normal installation steps. Once you are done with that raspberry pi is ready for use.

**Putty:-**

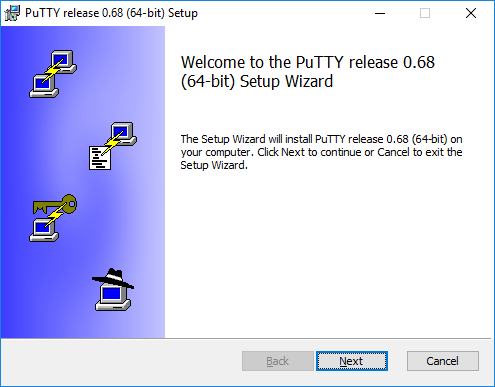
## DOWNLOAD INSTALLATION PACKAGE

One can download latest version of putty from the site <https://www.ssh.com/ssh/putty/download> .The only need is to select the appropriate version of the installer compatible with the computer and operating system.

**STARTING THE INSTALLER**

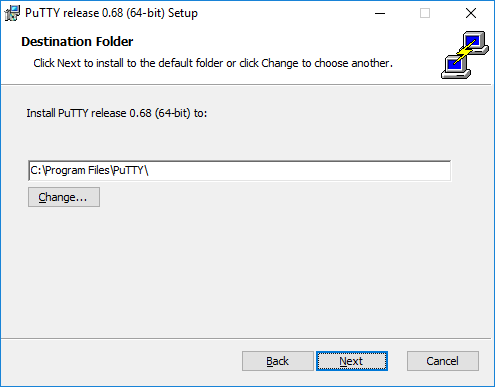
1. Once the download is finished, just click on the installer and start the installation.

## CONFIGURING AND INSTALLING

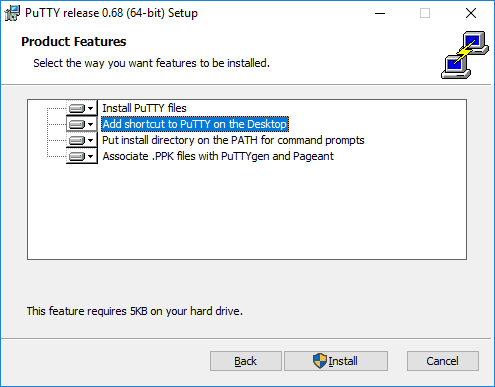


Just hit next to start the installation.

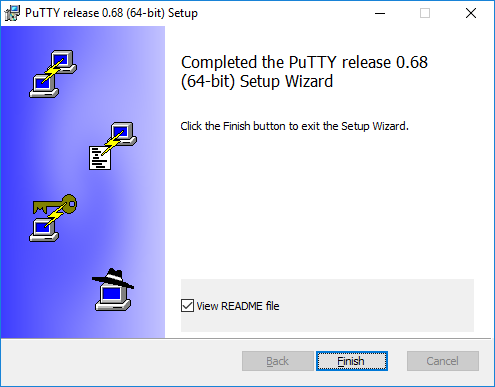
The installer then asks for the Destination folder in which to install the software. Almost always it is best to use the default value. Just click Next.



Then, the installer asks to select product features to install. You probably want to add a shortcut on the desktop if you expect to use the software frequently. All the other options generally should be enabled. When ready, click Install.

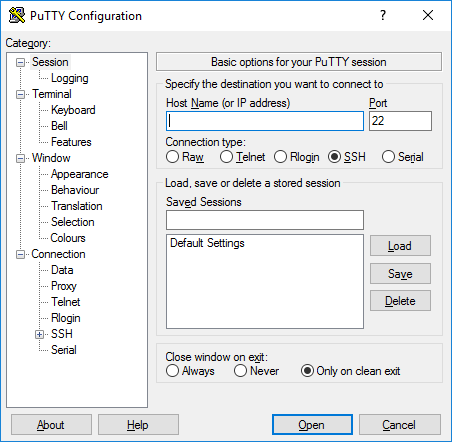


When the installation has completed successfully, it should show a Completed screen. Click Finish to exit the installer.

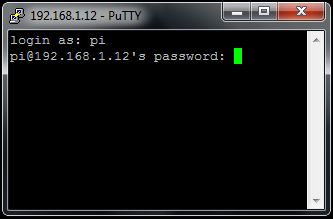


**Running Raspberry Pi GUI:-**

1. Run putty and enter the ip address of raspberry pi and connection type as SSH and hit open.



1. Once the putty raspberry pi is detected the putty will ask for the username and ` password for the raspberry pi.



1. Once the raspberry login is successful the raspberry pi terminal will get opened.

**VNC server**

## DOWNLOAD INSTALLATION PACKAGE

One can download latest version of putty from the site <https://www.realvnc.com/en/connect/download/vnc/windows/> .The only need is to select the appropriate version of the installer compatible with the computer and operating system.

## CONFIGURING AND INSTALLING

Double-click the executable to start the graphical Install Wizard, and follow the instructions.

1. Then start the vcn server from the putty terminal by typing the command vncserver. Then vcn server will generate the number for the the raspberry pi.
2. Open the VCN server and select the corresponding raspberry number. Then the user need to fill the username and password and then raspberry pi GUI will get open.
3. **Python code to integrate all the sensors**

**final\_script\_fyp.py**

import subprocess

import os

import socket

import csv

import time,sched

import RPi.GPIO as GPIO

import glob

import logging

import requests

import json

from errno import ENETUNREACH

headers={'Content-type': 'application/json'} #headers set to specify the content type of the data being sent to the URI

s = sched.scheduler(time.time, time.sleep) #used to start a new thread process

from threading import Timer

from suds.client import Client

URL\_get\_data = "http://192.168.43.104:5010/get\_data" #URI to send JSON data to be inserted into get\_data table

URL\_update\_mois\_data="http://192.168.43.104:5010/update\_mois\_data" #URI to send JSON data to be updated into update\_mois\_data table

URL\_delete\_ip="http://192.168.43.104:5010/delete\_ip" #URI to send JSON data to be deleated from the table

TRIG=23 # trigger pin of ultrasonic

ECHO=24 # echo pin of ultrasonic

mois = 0

RELAY\_1 = 25

RELAY\_2=16

GPIO.setmode(GPIO.BCM)

GPIO.setup(RELAY\_1,GPIO.OUT)

GPIO.setup(RELAY\_2,GPIO.OUT)

GPIO.setup(TRIG,GPIO.OUT)

GPIO.setup(ECHO,GPIO.IN)

moisture\_threshold\_low=550

moisture\_threshold\_high=80

water\_tank\_height=25

water\_tank\_status = ""

distance = 0

raspberry\_id=1

def read\_distance(): # measures using the ultrasonic sensor and returns the level of water in the tank

   try:

       water\_level=0

       print "Water level sensor testing"

       GPIO.output(TRIG, False)

       print "Waiting For Sensor To Settle"

       time.sleep(2)

       GPIO.output(TRIG, True) # trigger starts sending pulse

       time.sleep(0.00001)

       GPIO.output(TRIG, False) # trigger stops sending pulse

       while GPIO.input(ECHO)==0: # loops till echo receives back the reflected pulse

               pulse\_start = time.time()

       while GPIO.input(ECHO)==1:

               pulse\_end = time.time()

       pulse\_duration = pulse\_end - pulse\_start # calculates the duration of pulse traversed

       distance = pulse\_duration \* 17150

       distance = round(distance, 2)

       print distance

       final\_distance = distance

       print int(100-((100\*distance)/water\_tank\_height))

       return final\_distance

   except:

       return 0

def run\_status():  # checks for the functioning status of nodemcus

   try:

       data="rashmipawar921@gmail.com"

       json\_data = json.dumps(data) # converts dictionary data into json

       r = requests.get(url = URL\_delete\_ip, json = json\_data, headers = headers) # the data in json format is sent to the specified URI using GET method

       print "running alive\_status\_nmcu.py"

       execfile("/home/pi/Desktop/alive\_status\_nmcu.py") # executes the specified file

   except:

       pass

def run\_sensor():  # requests the nodemcus for sensor readings and sends it to server

   temp=0

   raspberry\_id="1"

   water\_tank\_level=30

   water\_pump\_status="off"

   send\_noti="1"

   water\_level=20

   ip\_list=[]

   f=open("/home/pi/Desktop/nmcu\_ip.csv","r") # opens file in read mode

   file\_contents=csv.reader(f) # reads each record from csv into a dictionary

   ip\_list=list(file\_contents) # converts dictionary into list

   moisture\_readings={}

   for i in range(len(ip\_list)): # iterates for each ip of the alive nodemcus

       try:

           deadline = time.time() + 5.0 # used to set a time limit of 5 seconds

           UDP\_IP=ip\_list[i][1]

           UDP\_PORT=1885

           MESSAGE="aiscmm\_smart\_irrigation\_169.254.152.165\_sensor\_data" # message sent to the nodemcu requesting for sensor readings

           print "UDP target ip:", UDP\_IP

           print "udp target port:", UDP\_PORT

           print "msg:",MESSAGE

           sock=socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM) # creates a socket

           try:

               sock.sendto(MESSAGE,(UDP\_IP, UDP\_PORT)) # udp message is sent to the specific nodemcu using ip and port

               print "Packet send"

           except IOError as e:

               if e.errno==ENETUNREACH:

                   pass

           sock.settimeout(deadline-time.time()) # closes connection after timeout

           chunks=[]

           bytes\_recd=0

           chunks=sock.recv(4096) # receives data sent by nodemcu

           print chunks

           chunks\_list=[]

           chunks\_list=chunks.split(",") # list of sensor readings is created

           temp=chunks\_list[0]

           mois = chunks\_list[2]

           moisture\_readings[i]=float(mois)

           raspberry\_id="1"

           water\_pump\_status="off"

           send\_noti="1"

           flag = 2

       except socket.timeout:

           print "time out"

           moisture\_readings[i]=2000

           continue

   no\_of\_pipes=2 #static

   pipe\_list1=[]

   pipe\_list2=[]

   for i in moisture\_readings:

       pipe\_no=(i+1)%no\_of\_pipes;

       if pipe\_no==1:

           pipe\_list1.append(moisture\_readings[i]) # appends into list readings of moisture near pipe 1

       else:

           pipe\_list2.append(moisture\_readings[i]) # appends into list readings of moisture near pipe 2

   mois\_avg1=0;

   mois\_avg2=0;

   faulty\_count1=0

   faulty\_count2=0

   mois\_avg\_list=[]

   for i in range(len(pipe\_list1)):

       if pipe\_list1[i]==2000:

           faulty\_count1=faulty\_count1+1

           continue

       else:

           mois\_avg1=mois\_avg1+pipe\_list1[i]

       mois\_avg1=mois\_avg1/(len(pipe\_list1)-faulty\_count1) # calculates average of moisture readings near pipe1

   mois\_avg\_list.append(mois\_avg1)

   for i in range(len(pipe\_list2)):

       if pipe\_list2[i]==2000:

           faulty\_count2=faulty\_count2+1

           continue

       else:

           mois\_avg2=mois\_avg2+pipe\_list2[i]

       mois\_avg2=mois\_avg2/(len(pipe\_list2)-faulty\_count2) # calculates average of moisture readings near pipe1

   mois\_avg\_list.append(mois\_avg2)

   print mois\_avg\_list

   water\_tank\_status = "off"

       try:

       for i in range(len(mois\_avg\_list)):

           try:

               if(mois\_avg\_list[i]>moisture\_threshold\_low): # checks if moisture level measured is lesser than required threshold

                   distance = read\_distance()

                   water\_tank\_level=distance

                   print(distance, water\_tank\_height)

                   if(distance<water\_tank\_height): # checks if there is sufficient water in the tank

                       print("inside the pump")

                       if i==0:

                           GPIO.output(RELAY\_1,True) # turns on the water pump 1

                           water\_tank\_status = "motor on"

                           print water\_tank\_status

                           data={"raspberry\_id":raspberry\_id,"pump\_id":i+1,"mois":mois\_avg\_list[i]} # creates dictionary of current values to be sent to server

                           print data

                           json\_data = json.dumps(data) # converts dictionary to json

                           r = requests.get(url = URL\_update\_mois\_data, json = json\_data, headers = headers) # sends json data to the URI

                           print "data is updated to db"

                           time.sleep(10)

                           GPIO.output(RELAY\_1, False) # turns off the water pump 1

                           water\_tank\_status = "off"

                           print water\_tank\_status

                       else:

                           GPIO.output(RELAY\_2,True) # turns on the water pump 2

                           water\_tank\_status = "motor on"

                           print water\_tank\_status

                           data={"raspberry\_id":raspberry\_id,"pump\_id":i+1,"mois":mois\_avg\_list[i]} # creates dictionary of current values to be sent to server

                           json\_data = json.dumps(data) # converts dictionary to json

                           r = requests.get(url = URL\_update\_mois\_data, json = json\_data, headers = headers) # sends json data to the URI

                           print "data is updated to db"

                           time.sleep(10)

                           GPIO.output(RELAY\_2, False) # turns off the water pump 2

                           water\_tank\_status = "off"

                           print water\_tank\_status

                   else:

                       print water\_tank\_status

               else:

                   print water\_tank\_status

           except:

               continue

           data={"temp":temp,"mois":mois\_avg\_list[i],"raspberry\_id":raspberry\_id,"water\_tank\_level":water\_tank\_level,"water\_tank\_status":water\_tank\_status,"send\_noti":send\_noti} # creates dictionary of current values to be sent to server

           json\_data = json.dumps(data) # converts dictionary to json

           r = requests.get(url = URL\_get\_data, json = json\_data, headers = headers) # sends json data to the URI

           print "data is updated to db"

   except:

       pass

while(1):

   s.enter(30, 1, run\_sensor, ()) # schedules the process thread to run every 30 seconds

   s.enter(300, 2, run\_status, ()) # schedules the process thread to run every 300 seconds

   s.run()

1. Check Alive or Dead status of nodeMCU

**alive\_status\_nmcu.py**

import subprocess

import os

import socket

import csv

import time,sched

import RPi.GPIO as GPIO

import glob

import logging

import json

import requests

s = sched.scheduler(time.time, time.sleep)

from threading import Timer

from suds.client import Client

headers={'Content-type': 'application/json'} #headers set to specify the content type of the data being sent to the URI

URL = "http://192.168.43.104:5010/insert\_ip" #URI to send JSON data to be inserted into insert\_ip table

def run\_sensor\_status(): # checks is the nodemcu is functioning else sends the ip of faulty to server

   faulty\_nmcu=[]

   raspberry\_id="1"

   send\_noti="6"

   ip\_list=[]

   f=open("/home/pi/Desktop/nmcu\_ip.csv","r") # opens file in read mode

   file\_contents=csv.reader(f) # reads each record from csv into a dictionary

   ip\_list=list(file\_contents) # converts dictionary into list

   for i in range(len(ip\_list)):  # iterates for each ip of the nodemcus

       try:

           deadline = time.time() + 5.0 # used to set a time limit of 5 seconds

           UDP\_IP=ip\_list[i][1]

           UDP\_PORT=1885

           MESSAGE="aiscmm\_smart\_irrigation\_169.254.152.165\_sensor\_status" # message sent to the nodemcu requesting for sensor status

           print "UDP target ip:", UDP\_IP

           print "udp target port:", UDP\_PORT

           print "msg:",MESSAGE

           sock=socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM) # creates a socket

           sock.sendto(MESSAGE,(UDP\_IP, UDP\_PORT)) # udp message is sent to the specific nodemcu using ip and port

           print "Packet send"

           sock.settimeout(deadline-time.time()) # closes connection after timeout

           chunks=[]

           bytes\_recd=0

           chunks=sock.recv(4096) # receives data sent by nodemcu

           chunks\_list=[]

           chunks\_list=chunks.split(",")

           st=chunks\_list[0]

           print st

           raspberry\_id="1"

           send\_noti="6"

           if st!="alive":

               faulty\_nmcu.append(ip\_list[i]) # unrechable ips are appended into the list

       except socket.timeout:

           print "time out"

           faulty\_nmcu.append(ip\_list[i])

           continue

   print faulty\_nmcu

   return faulty\_nmcu

faulty\_nmcu=run\_sensor\_status()

for row in faulty\_nmcu: # iterates through each faulty nodemcu ip

   data={"nodemcu\_id":row[0],"raspberry\_id":1,"nodemcu\_ip":row[1],"raspberry\_ip":"169.254.152.165","email":"rashmipawar921@gmail.com"} # creates dictionary of current values to be sent to server

   json\_data = json.dumps(data) # converts dictionary to json

   r = requests.get(url = URL, json = json\_data, headers = headers) # sends json data to the URI

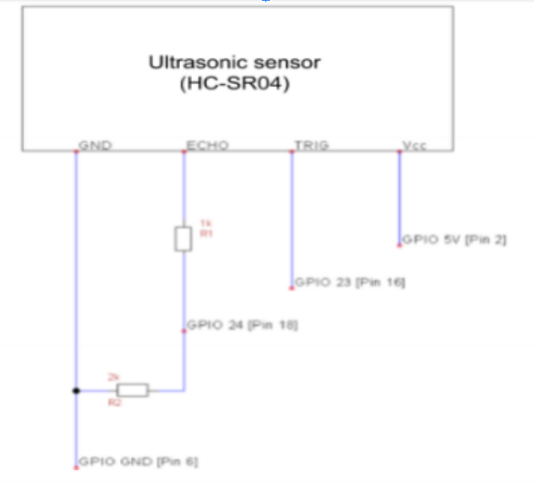
print "data is updated to db"

1. Run both the codes using terminal

**Connections to Raspberry Pi**

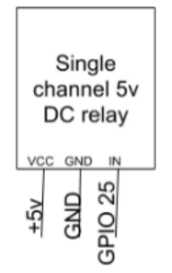
**Ultrasonic**

HC-SR 04 ultrasonic distance sensor is used to measure water level in storage tank. This sensor operates on 5V. The maximum measuring distance is 200cm.The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by water it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver. Once the signal is received by receiver using time distance speed formula we can calculate the water level in the tank. A basic ultrasonic sensor consists of one or more ultrasonic transmitters (basically speakers), a receiver, and a control circuit. The transmitters emit a high frequency ultrasonic sound, which bounce off any nearby objects. Some of that ultrasonic noise is reflected and detected by the receiver on the sensor. That return signal is then processed by the control circuit to calculate the time difference between the signal being transmitted and received. This time can subsequently be used, along with some clever math, to calculate the distance between the sensor and the reflecting object.



**RELAY**

The single channel 5v relay is used to control the water pump. The operating voltage of the relay is 5v. This relay is connected to GPIO pin of raspberry pi. Whenever the moisture level is below threshold value water pump is switched on using the relay and also water pump is switched off once threshold level is reached. Connection consist of 2 relay one connected to GPIO 25 and the other to GPIO 16.



**WATER PUMP**

