TABLE OF CONTENT

Sl. No.	TITLE	PAGE No.
1	Project Description	iv
2	Raspberry Pi Setup	v
3	Temperature Sensor Setup	vii
4	Python Code for Temp Sensor	ix
5	Ph Sensor Setup	x
6	Python Code for Ph Sensor	xi
7	Conclusion	xii

LIST OF FIGURES

FIGURE No.	TITLE	PAGE No.
1	Raspberry Pi Model B & Layout	iv
2	Raspberry Model B Pin Layout	v
3	Putty Configuration	vi
4	Temp Sensor Diagram and Connection Layout	vii
5	Ph Sensor Diagram and Connection Layout	viii

Report Contents:

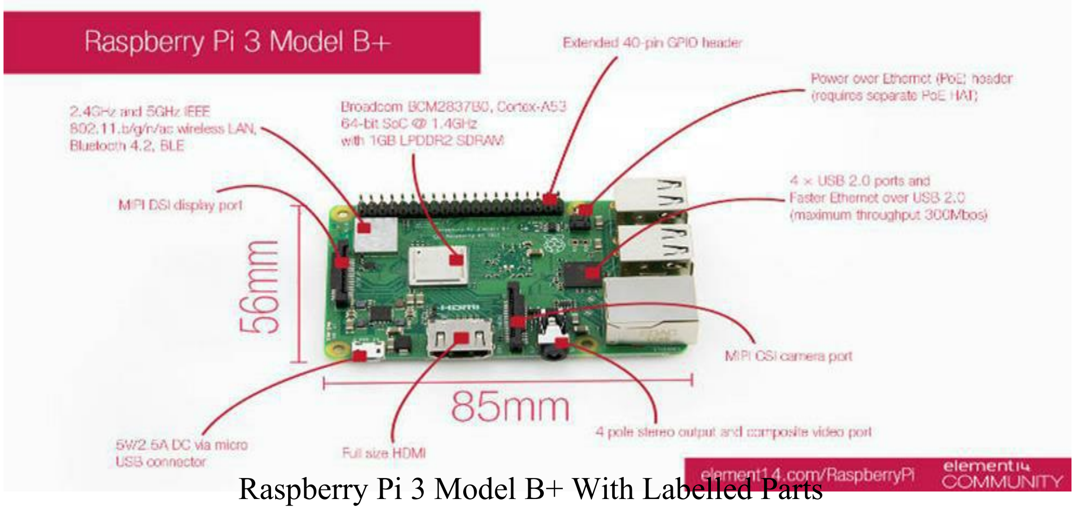
Description of Project (Max 1-2 pages)
System design/block diagram of project:
Algorithm/Flowchart of project:
Hardware Components of projects (Screenshots to be included)
Prototype of project (Screenshot to be included)
Results of prototype developed (Screenshots to be included)
Conclusion

IOT: WATER QUALITY ANALYSIS PROJECT

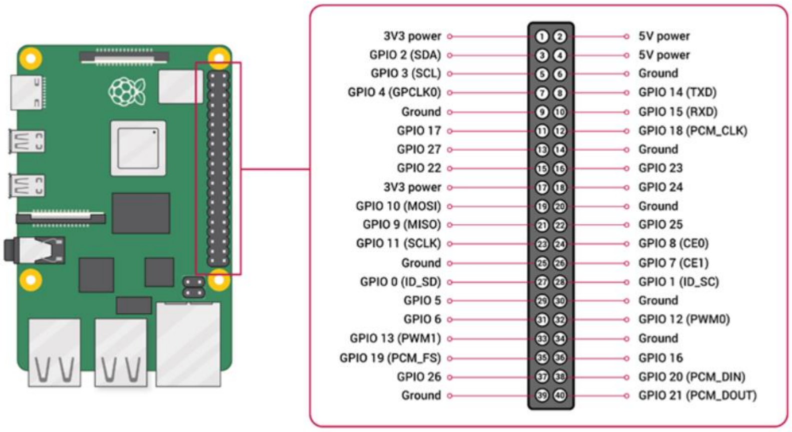
The project is created using Raspberry Pi 3 Model B/ Raspberry Pi 3 Model A+

The following project uses Raspberry Pi to detect and sense the PH and temperature of water.

Raspberry Pi3 Model B


Raspberry Pi 3 Model B+
Extended 40-pin GPIO header
Power over Ethernet (PoE) header (requires separate PoE HAT)
2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz with 1GB LPDDR2 SDRAM
4 × USB 2.0 ports and Faster Ethernet over USB 2.0 (maximum throughput 300Mbps)
MIPI DSI display port
56mm
MIPI CSI camera port
85mm
4 pole stereo output and composite video port
5V/2.5A DC via micro USB connector
Full size HDMI
element14.com/RaspberryPi
element14 COMMUNITY

Raspberry Pi 3 Model B+ With Labelled Parts



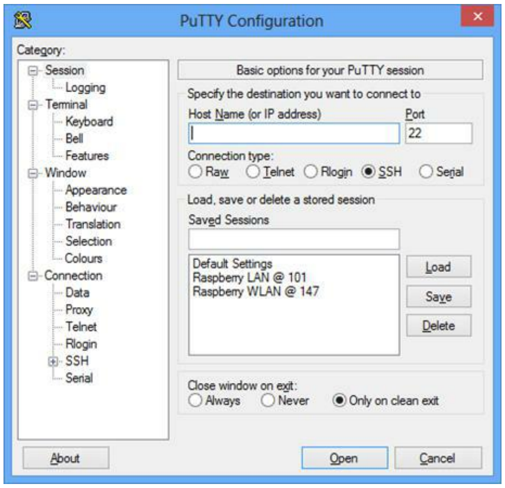
Raspberry Pi 3 Model B Pin Layout

Setting up Raspberry Pi 3:

Requirements:

1. Ethernet Cable
2. Noobs/Raspbian – Jessie Installed
3. Micro SD Card
4. Win32Disk Formatter
5. Python
6. Power Supply
7. Putty and Xming Server Software Installed

Step 1: Configure the Putty with Xming and connect with Raspberry Pi:



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions

Default Settings
Raspberry LAN @ 101
Raspberry WLAN @ 147

Close window on exit: ○ Always ○ Never ● Only on clean exit

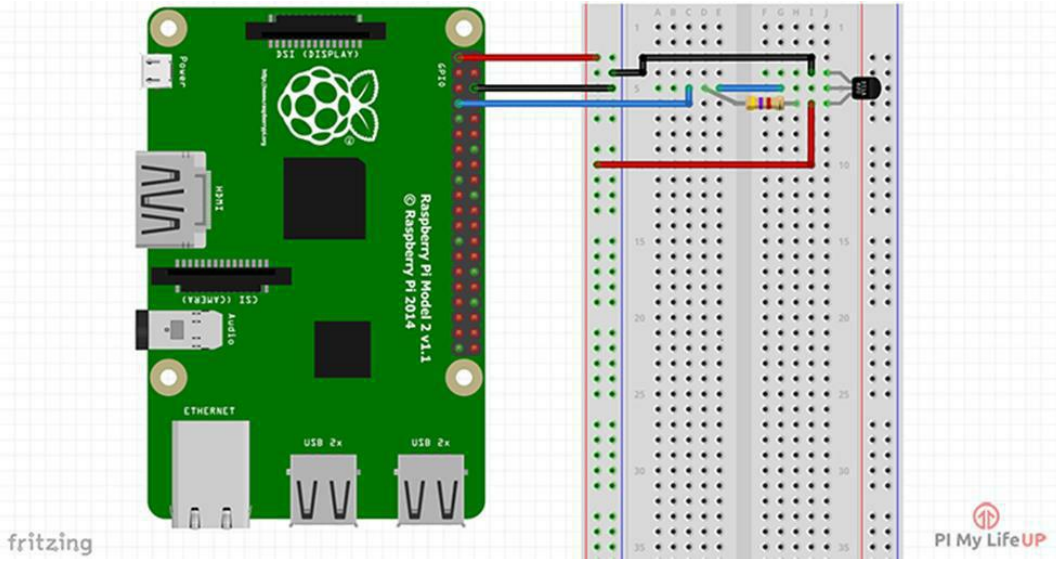
Step 2: Write host name as: raspberrypi.myhome.net – Port 22

1. Enable port forwarding in Xming.
2. X Display Location: -localhost:0
3. Give a session name in sessions, save it and open.

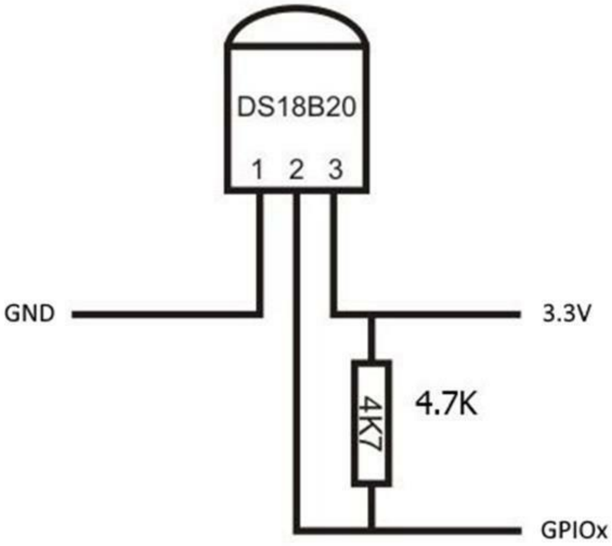
Step 3: Now in the terminal:

1. Username: Pi
2. Password: “Raspberry”
3. Then write lxterminal to open Raspbian terminal shell.
4. Give command lxde to open Raspbian software and we are ready to code!

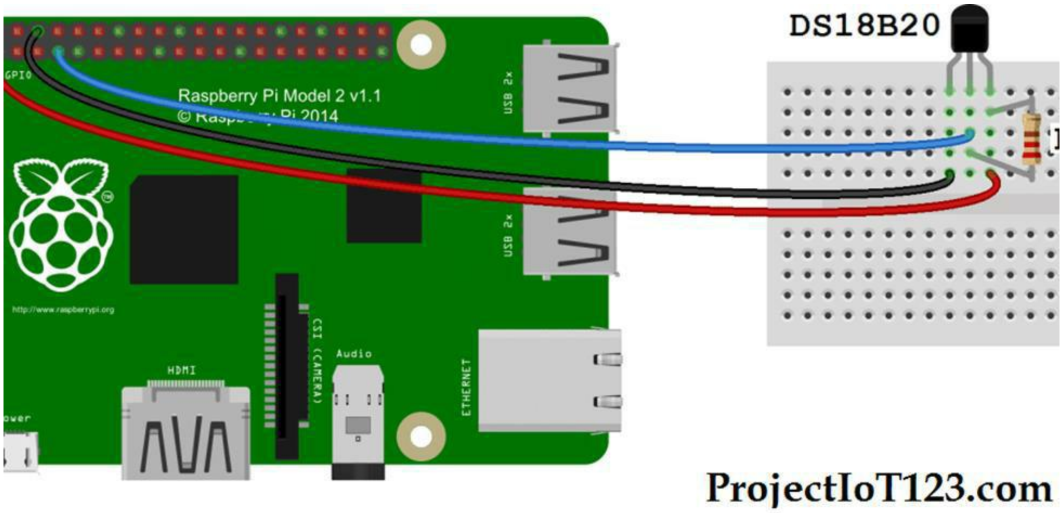
Setting Up Connection Of DB18B20 Temperature Sensor with Breadboard and Raspberry Pi:



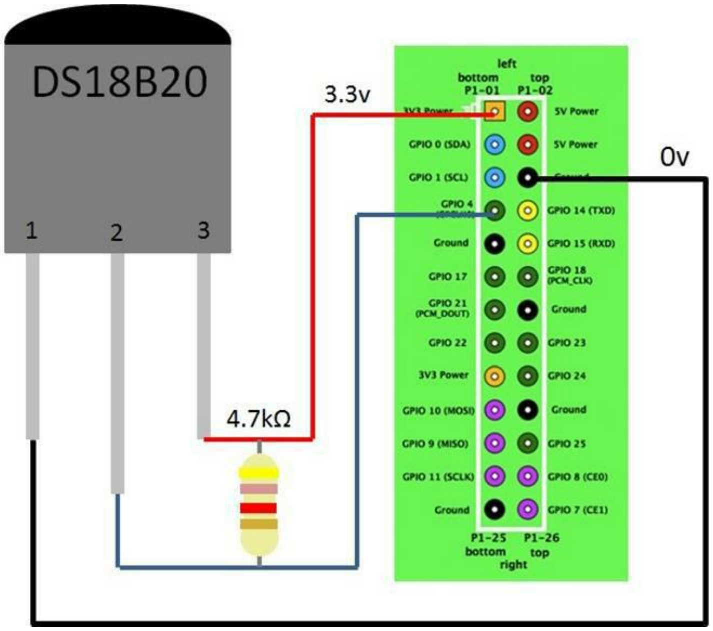
We will make connections with 1, 4 and 7 GPIO pins.



Step 1: Place the 4.7l resistor in between the positive lead and output lead in the breadboard.



Step 2: Connect the output pin wire to the GPIO pin 7.



Step 3: After the connection has been established we have to enable one wire interface for GPIO4 for input/output through our sensor to Raspberry Pi.

Step 4: The following commands are to be executed:

1. Open PUTTY
2. Sudo nano boot/config.txt
3. dtoverlay = w1-gpio then press Ctrl+x+y to save and exit.
4. Reboot raspberry pi.
5. To check if the device is working: sudo modprobe w1-gpio sudo modprobe w1-therm
6. Cd /sys/bus/w1/devices to change directory
7. The type ‘ls’ to see the dir where the temperature is stored.
8. Now ‘cd’ to that directory.
9. Now to output data: cat w1_slave

Python Code For Reading Temperature Data And Storing It In DB:

```
import os
import glob
import time
import MySQLdb

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
  device_folder = glob.glob(base_dir + '28*')[0]
       device_file = device_folder + '/w1_slave'
conn = MySQLdb.connect(host="localhost", user="root",
passwd="", db="nolinka")

cursor = conn.cursor()

     def read_temp_raw():
f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
         if equals_pos != -1:
     temp_string = lines[1][equals_pos+2:]

        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c

while True:
     print(read_temp())
     mytemp = read_temp()
     loggit = "UPDATE temperature SET Value=%s WHERE ID=1"
     cursor.execute(loggit, (mytemp)) conn.commit()


          time.sleep(5)
```

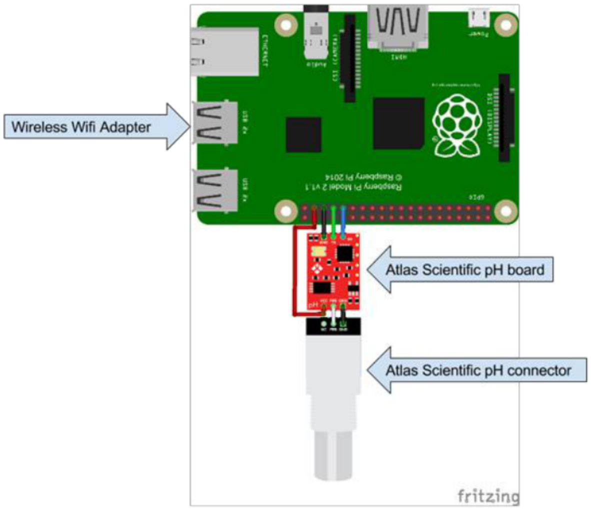
To test the code write the following command… in your shell: →

```
sudo python thermometer_sensor.py
```

Adding Ph Sensor To The Raspberry Pi Setup:

Items needed for this setup:

1. Ph Probe
2. (18 bit) ADC Converter
3. 3 Pin Connector



Connect the Ph probe with the ADC Converter and connect the ADC to the Raspberry Pi setup.

Execute the following commands in the terminal:

```
Sudo apt-get update
Sudo apt-get upgrade
```

Enable ‘spi’ and ‘i2c’ in the Raspi configuration setup.

Install ‘Spidev’: pip install spidev

Python Code For Ph Sensor:

```
import spidev

import RPi.GPIO as GPIO

import time

import sys


GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)


GPIO.setup(21, GPIO.OUT)
```

```
servo = GPIO.PWM(21, 50)
servo.start(2.5)

spi = spidev.SpiDev() # create spi object
spi.open(0, 0) # open spi port 0, device (CS) 0, for the MCP8008/ADC
spi.max_speed_hz=1000000

def readadc(adcnum): # read out the ADC
    if ((adcnum > 7) or (adcnum < 0)):
        return -1
    r = spi.xfer2([1, (8 + adcnum) << 4, 0])
    adcout = ((r[1] & 3) << 8) + r[2]
    return adcout


while True:
    Value = readadc(0) #read adc channel 0
    phvalue = float(value)*3.3/1024/6
    phvalue = 3.5*phvalue

    print (phvalue "PH")
    time.sleep(10)
```

Conclusion:

Thus, the simple setup described above can be used to monitor the quality of water in water bodies. To get more accurate and reliable data more sophisticated setups are required.