GUJARAT TECHNOLOGICAL UNIVERSITY

Chandkheda, Ahmedabad

Affiliated

I.I.E.T. DHARMAJ

A Report On-

Weather Reporting Project

Under subject of

Internet of Things

B. E. II, Semester – VI

(Computer Branch)

Submitted by:
Group:

Sr.	Name of student	Enrolment No.
1.	Patel Pooja	181010107008

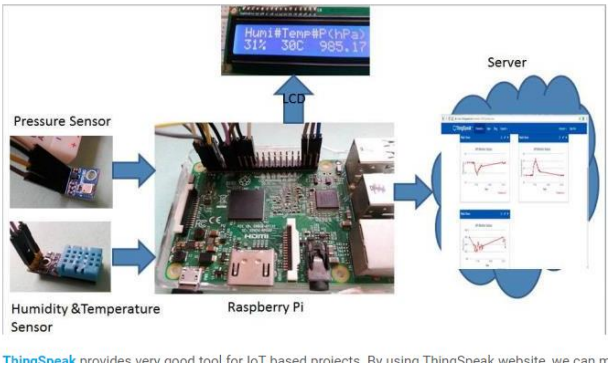
(Asst. Professor)
(Faculty Guide)

Ankita Padhiyar
(Professor)
Head of the Department

Academic Year
(2020-21)

❖Working and Thing Speak Setup:

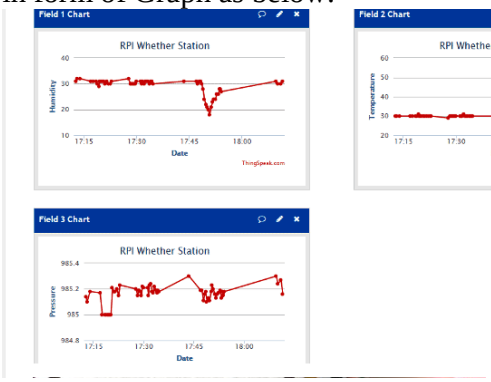
This IoT based project has four sections. Firstly DHT11 sensor senses the Humidity & Temperature Data and BM180 sensor measures the atmospheric pressure. Secondly **Raspberry Pi** reads the DHT11 sensor module's output by using single wire protocol and BM180 pressure sensor's output by using I2C protocol and extracts both sensors values into a suitable number in percentage (humidity), Celsius scale (temperature), hectoPascal or millibar (pressure). Thirdly, these values are sent to ThingSpeak server by using inbuilt Wi-Fi of **Raspberry Pi 3**. And finally **ThingSpeak** analyses the data and shows it in a Graph form. A LCD is also used to display these values locally.



ThingSpeak provides very good tool for IoT based projects. By using ThingSpeak website, we can pr

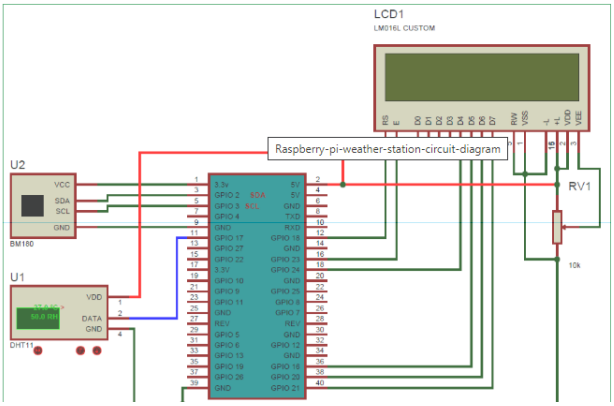
Now we need this 'Feed Get Request URL' in our Python code to open "api.thingspeak.com" and then send data using this Feed Request as query string. And Before sending data user needs to enter the temperature, humidity and pressure data in this query String using variables in program, check in the Code at the end this article.

Working of DHT11 is based on single wire serial communication for fetching data from DHT11. Here we have used **AdaFruit DHT11 library for interfacing DHT11 with Raspberry Pi**. Raspberry Pi here collects the Humidity and temperature data from DHT11 and atmospheric pressure from BMP180 sensor and then sends it to 16x2 LCD and ThingSpeak server. ThingSpeak displays the Data in form of Graph as below:

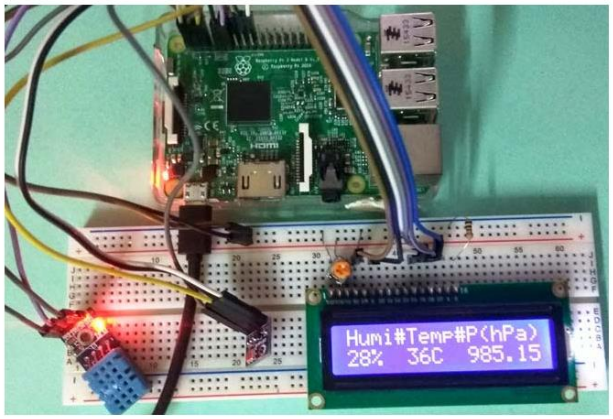


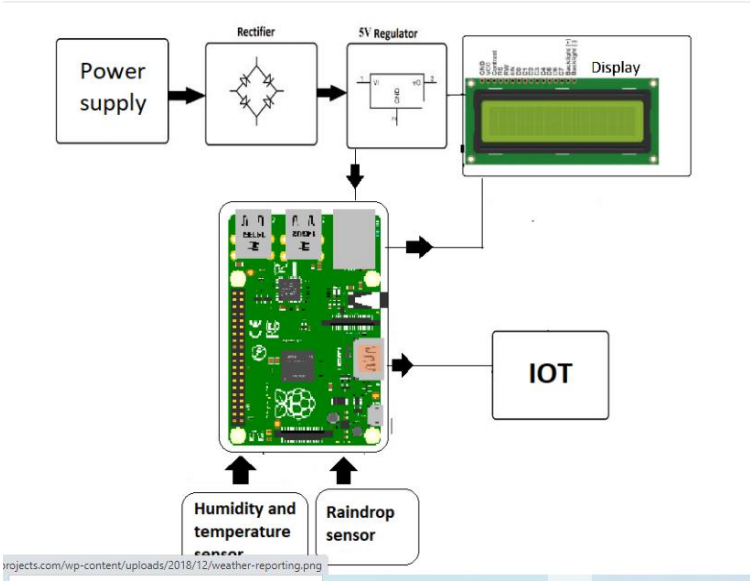
- **Circuit diagram:**

Circuit Diagram:



- **Circuit diagram (2):**





❖ Raspberry Pi Configuration and Python Program:

We are using **Python language** here for the Program. Before coding, user needs to configure Raspberry Pi. You can check our previous tutorials for Getting Started with Raspberry Pi and Installing & Configuring Raspbian Jessie OS in Pi.

First off all we need to install Adafruit Python DHT Sensor Library files to run this project on Raspberry Pi. To do this we need to follow given commands:

```
sudo apt-get install git-core

sudo apt-get update

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

cd Adafruit_Python_DHT

sudo apt-get install build-essential python-dev

sudo python setup.py install
```

- **Programming part** of this project plays a very important role to perform all the operations. First of all we include all required libraries, initiaze variables and define pins for LCD and DHT11.

```
import sys

import RPi.GPIO as GPIO

import os

import Adafruit_DHT

import urllib2
```

```
import smbus

import time

from ctypes import c_short


#Register Address

regCall   = 0xAA

... ......

..... ...
```

In *def main():* function, below code is used for sending the data to the server and display it over the LCD, continuously in *while* loop.

```
def main():


    print 'System Ready...'

    URL = 'https://api.thingspeak.com/update?api_key=%s' % key

    print "Wait...."

    while True:

            (humi, temp)= readDHT()

            (pressure) =readBmp180()


            lcdcmd(0x01)

            lcdstring("Humi#Temp#P(hPa)")
```

```
        lcdstring(humi+'%'+"  %sC  %s" %(temp, pressure))

        finalURL = URL +"&field1=%s&field2=%s"%(humi, temp)+"&f
ield3=%s" %(pressure)

        print finalURL

        s=urllib2.urlopen(finalURL);

        print  humi+ " " + temp + " " + pressure

        s.close()

        time.sleep(10)
```

For LCD, *def lcd_init()* function is used to initialize LCD in four bit mode, *def lcdcmd(ch)* function is used for sending command to LCD, *def lcddata(ch)* function is used for sending data to LCD and *def lcdstring(Str*) function is used to send data string to LCD. You can check all these functions in Code given afterwards.

Given def *readDHT()* function is used for reading DHT11 Sensor:

```
def readDHT():

   humi, temp = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, DHTp
in)

   return (str(int(humi)), str(int(temp)))
```

def readBmp180 function is used for reading pressure from the BM180 sensor. BM180 sensor can also give temperature but here we have only used it for calculating pressure.

```
def readBmp180(addr=deviceAdd):

  value = bus.read_i2c_block_data(addr, regCall, 22)  # Read calibration
data
```

```
# Convert byte data to word values

AC1 = convert1(value, 0)

AC2 = convert1(value, 2)

AC3 = convert1(value, 4)

AC4 = convert2(value, 6)

..... .......

........ ......
```

So this is the basic **Raspberry Pi Weather Station**, you can further extend it to measure various weather related parameters like wind speed, soil temperature, illuminance (lux), rainfall, air quality etc.

Code

```
import sys
import RPi.GPIO as GPIO
import os
import Adafruit_DHT
import urllib2
import smbus
import time
from ctypes import c_short

#Register Address
regCall   = 0xAA
regMean   = 0xF4
regMSB    = 0xF6
regLSB    = 0xF7
regPres   = 0x34
regTemp   = 0x2e

DEBUG = 1
sample = 2
deviceAdd =0x77
```

```
humi=""
temp=""

#bus = smbus.SMBus(0)  #for Pi1 uses 0
I2cbus = smbus.SMBus(1) # for Pi2 uses 1

DHTpin = 17

key="30BCDSRQ52AOI3UA"       # Enter your Write API key from
ThingSpeak

GPIO.setmode(GPIO.BCM)
# Define GPIO to LCD mapping
LCD_RS = 18
LCD_EN  = 23
LCD_D4 = 24
LCD_D5 = 16
LCD_D6 = 20
LCD_D7 = 21

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(LCD_E, GPIO.OUT)
GPIO.setup(LCD_RS, GPIO.OUT)
GPIO.setup(LCD_D4, GPIO.OUT)
GPIO.setup(LCD_D5, GPIO.OUT)
GPIO.setup(LCD_D6, GPIO.OUT)
GPIO.setup(LCD_D7, GPIO.OUT)

def convert1(data, i):   # signed 16-bit value
  return c_short((data[i]<< 8) + data[i + 1]).value

def convert2(data, i):   # unsigned 16-bit value
  return (data[i]<< 8) + data[i+1]

def readBmp180(addr=deviceAdd):
  value = bus.read_i2c_block_data(addr, regCall, 22)  # Read calibration
data

  # Convert byte data to word values
  AC1 = convert1(value, 0)
  AC2 = convert1(value, 2)
  AC3 = convert1(value, 4)
```

```
AC4 = convert2(value, 6)
AC5 = convert2(value, 8)
AC6 = convert2(value, 10)
B1  = convert1(value, 12)
B2  = convert1(value, 14)
MB  = convert1(value, 16)
MC  = convert1(value, 18)
MD  = convert1(value, 20)

  # Read temperature
bus.write_byte_data(addr, regMean, regTemp)
time.sleep(0.005)
(msb, lsb) = bus.read_i2c_block_data(addr, regMSB, 2)
P2 = (msb << 8) + lsb

# Read pressure
bus.write_byte_data(addr, regMean, regPres + (sample << 6))
time.sleep(0.05)
(msb, lsb, xsb) = bus.read_i2c_block_data(addr, regMSB, 3)
P1 = ((msb << 16) + (lsb << 8) + xsb) >> (8 - sample)

 # Refine temperature
X1 = ((P2 - AC6) * AC5) >> 15
X2 = (MC << 11) / (X1 + MD)
B5 = X1 + X2
temperature = (B5 + 8) >> 4

# Refine pressure
B6  = B5 - 4000
B62 = B6 * B6 >> 12
X1  = (B2 * B62) >> 11
X2  = AC2 * B6 >> 11
X3  = X1 + X2
B3  = (((AC1 * 4 + X3) << sample) + 2) >> 2

X1 = AC3 * B6 >> 13
X2 = (B1 * B62) >> 16
X3 = ((X1 + X2) + 2) >> 2
B4 = (AC4 * (X3 + 32768)) >> 15
B7 = (P1 - B3) * (50000 >> sample)
```

```
  P = (B7 * 2) / B4

  X1 = (P >> 8) * (P >> 8)
  X1 = (X1 * 3038) >> 16
  X2 = (-7357 * P) >> 16
  pressure = P + ((X1 + X2 + 3791) >> 4)

  return (str(pressure/100.0))

def readDHT():
    humi, temp = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11,
DHTpin)
    return (str(int(humi)), str(int(temp)))

def lcd_init():
  lcdcmd(0x33)
  lcdcmd(0x32)
  lcdcmd(0x06)
  lcdcmd(0x0C)
  lcdcmd(0x28)
  lcdcmd(0x01)
  time.sleep(0.0005)

def lcdcmd(ch):
  GPIO.output(RS, 0)
  GPIO.output(D4, 0)
  GPIO.output(D5, 0)
  GPIO.output(D6, 0)
  GPIO.output(D7, 0)
  if ch&0x10==0x10:
    GPIO.output(D4, 1)
  if ch&0x20==0x20:
    GPIO.output(D5, 1)
  if ch&0x40==0x40:
    GPIO.output(D6, 1)
  if ch&0x80==0x80:
    GPIO.output(D7, 1)
  GPIO.output(EN, 1)
  time.sleep(0.0005)
  GPIO.output(EN, 0)
```

```
 # Low bits
 GPIO.output(D4, 0)
 GPIO.output(D5, 0)
 GPIO.output(D6, 0)
 GPIO.output(D7, 0)
 if ch&0x01==0x01:
   GPIO.output(LCD_D4, 1)
 if ch&0x02==0x02:
   GPIO.output(LCD_D5, 1)
 if ch&0x04==0x04:
   GPIO.output(LCD_D6, 1)
 if ch&0x08==0x08:
   GPIO.output(LCD_D7, 1)
 GPIO.output(EN, 1)
 time.sleep(0.0005)
 GPIO.output(EN, 0)

def lcddata(ch):
 GPIO.output(RS, 1)
 GPIO.output(D4, 0)
 GPIO.output(D5, 0)
 GPIO.output(D6, 0)
 GPIO.output(D7, 0)
 if ch&0x10==0x10:
   GPIO.output(D4, 1)
 if ch&0x20==0x20:
   GPIO.output(D5, 1)
 if ch&0x40==0x40:
   GPIO.output(D6, 1)
 if ch&0x80==0x80:
   GPIO.output(D7, 1)
 GPIO.output(EN, 1)
 time.sleep(0.0005)
 GPIO.output(EN, 0)

 # Low bits
 GPIO.output(D4, 0)
 GPIO.output(D5, 0)
 GPIO.output(D6, 0)
 GPIO.output(D7, 0)
```

```
  if ch&0x01==0x01:
    GPIO.output(LCD_D4, 1)
  if ch&0x02==0x02:
    GPIO.output(LCD_D5, 1)
  if ch&0x04==0x04:
    GPIO.output(LCD_D6, 1)
  if ch&0x08==0x08:
    GPIO.output(LCD_D7, 1)
  GPIO.output(EN, 1)
  time.sleep(0.0005)
  GPIO.output(EN, 0)

def lcdstring(Str):
  l=0;
  l=len(Str)
  for i in range(l):
    lcddata(ord(message[i]))

lcd_init()
 lcdcmd(0x01)
lcdstring("Circuit Digest")
lcdcmd(0xc0)
lcdstring("Welcomes you")
time.sleep(3) # 3 second delay

# main() function
def main():

    print 'System Ready...'
    URL = 'https://api.thingspeak.com/update?api_key=%s' % key
    print "Wait...."
    while True:
            (humi, temp)= readDHT()
            (pressure) =readBmp180()

            lcdcmd(0x01)
            lcdstring("Humi#Temp#P(hPa)")
            lcdstring(humi+'%'+"  %sC  %s" %(temp, pressure))
            finalURL = URL +"&field1=%s&field2=%s"%(humi,
temp)+"&field3=%s" %(pressure)
```

```
print finalURL
s=urllib2.urlopen(finalURL);
print  humi+ " " + temp + " " + pressure
s.close()
time.sleep(10)
```

❖ **Conclusion:**

Weather prediction is a very important factor, which forecasts the climate in a region based upon the values of weather parameters. So the calculated results from this system can be made use in forecasting the weather of that locality for a period of time. As we made use of Raspberry pi in this model, immediate alert message or e-mail can be sent to the mobile phone, when the parameters changes are drastic. The technology changes day by day. Using the sensors for air temperature, air humidity, light, soil moisture, and rain detection in combination with Raspberry PI a prototype had been developed. Data from the sensors is transmitted to sever where it can be viewed globally which will be easily accessible to everyone. This IoT based system gives real-time monitoring of environmental parameters.

❖ **Refrences:**

1) **https://www.irjet.net/archives/V6/i1/IRJET-V6I1220.pdf**
2) **https://iotworld.co/2018/01/iot-based-weather-station-by-using-raspberry-pi-3/**
3) **https://circuitdigest.com/microcontroller-projects/raspberry-pi-iot-weather-station-to-monitor-temperature-humidity-pressure**
4) **https://www.raspberrypi.org/products/**

s