Enviromental Monitoring

# **FUNCTIONS AND DESIGNS:**

The IOT-Based Environmental Monitoring and Alerting System incorporates a sophisticated architecture to enable accurate and efficient monitoring. Sensor nodes are strategically deployed in target areas to collect data on air quality parameters and flood-related variables. These sensor nodes communicate with a central microcontroller unit, which processes the data, applies advanced algorithms for pollution and flood detection, and transmits the results to a remote server through IoT technology. The system employs data fusion techniques to integrate information from multiple sensor nodes, facilitating comprehensive and reliable environmental monitoring.

Program(.py)

#save this in home/pi for importing into your Python code

# make sure to install python-smbus using below command

# sudo apt-get install python-smbus

import smbus

import time

from ctypes import c\_short

DEVICE = 0x77 # Default device I2C address

#bus = smbus.SMBus(0) # Rev 1 Pi uses 0

bus = smbus.SMBus(1) # Rev 2 Pi uses 1

def convertToString(data):

# Simple function to convert binary data into

# a string

return str((data[1] + (256 \* data[0])) / 1.2)

def getShort(data, index):

# return two bytes from data as a signed 16-bit value

return c\_short((data[index] << 8) + data[index + 1]).value

def getUshort(data, index):

# return two bytes from data as an unsigned 16-bit value

return (data[index] << 8) + data[index + 1]

def readBmp180Id(addr=DEVICE):

# Chip ID Register Address

REG\_ID = 0xD0

(chip\_id, chip\_version) = bus.read\_i2c\_block\_data(addr, REG\_ID, 2)

return (chip\_id, chip\_version)

def readBmp180(addr=0x77):

# Register Addresses

REG\_CALIB = 0xAA

REG\_MEAS = 0xF4

REG\_MSB = 0xF6

REG\_LSB = 0xF7

# Control Register Address

CRV\_TEMP = 0x2E

CRV\_PRES = 0x34

# Oversample setting

OVERSAMPLE = 3 # 0 - 3

# Read calibration data

# Read calibration data from EEPROM

cal = bus.read\_i2c\_block\_data(addr, REG\_CALIB, 22)

# Convert byte data to word values

AC1 = getShort(cal, 0)

AC2 = getShort(cal, 2)

AC3 = getShort(cal, 4)

AC4 = getUshort(cal, 6)

AC5 = getUshort(cal, 8)

AC6 = getUshort(cal, 10)

B1 = getShort(cal, 12)

B2 = getShort(cal, 14)

MB = getShort(cal, 16)

MC = getShort(cal, 18)

MD = getShort(cal, 20)

# Read temperature

bus.write\_byte\_data(addr, REG\_MEAS, CRV\_TEMP)

time.sleep(0.005)

(msb, lsb) = bus.read\_i2c\_block\_data(addr, REG\_MSB, 2)

UT = (msb << 8) + lsb

# Read pressure

bus.write\_byte\_data(addr, REG\_MEAS, CRV\_PRES + (OVERSAMPLE << 6))

time.sleep(0.04)

(msb, lsb, xsb) = bus.read\_i2c\_block\_data(addr, REG\_MSB, 3)

UP = ((msb << 16) + (lsb << 8) + xsb) >> (8 - OVERSAMPLE)

# Refine temperature

X1 = ((UT - AC6) \* AC5) >> 15

X2 = (MC << 11) / (X1 + MD)

B5 = X1 + X2

temperature = int(B5 + 8) >> 4

temperature = temperature / 10.0

# Refine pressure

B6 = B5 - 4000

B62 = int(B6 \* B6) >> 12

X1 = (B2 \* B62) >> 11

X2 = int(AC2 \* B6) >> 11

X3 = X1 + X2

B3 = (((AC1 \* 4 + X3) << OVERSAMPLE) + 2) >> 2

X1 = int(AC3 \* B6) >> 13

X2 = (B1 \* B62) >> 16

X3 = ((X1 + X2) + 2) >> 2

B4 = (AC4 \* (X3 + 32768)) >> 15

B7 = (UP - B3) \* (50000 >> OVERSAMPLE)

P = (B7 \* 2) / B4

X1 = (int(P) >> 8) \* (int(P) >> 8)

X1 = (X1 \* 3038) >> 16

X2 = int(-7357 \* P) >> 16

pressure = int(P + ((X1 + X2 + 3791) >> 4))

#pressure = float(pressure / 100.0)

altitude = 44330.0 \* (1.0 - pow(pressure / 101325.0, (1.0/5.255)))

altitude = round(altitude,2)

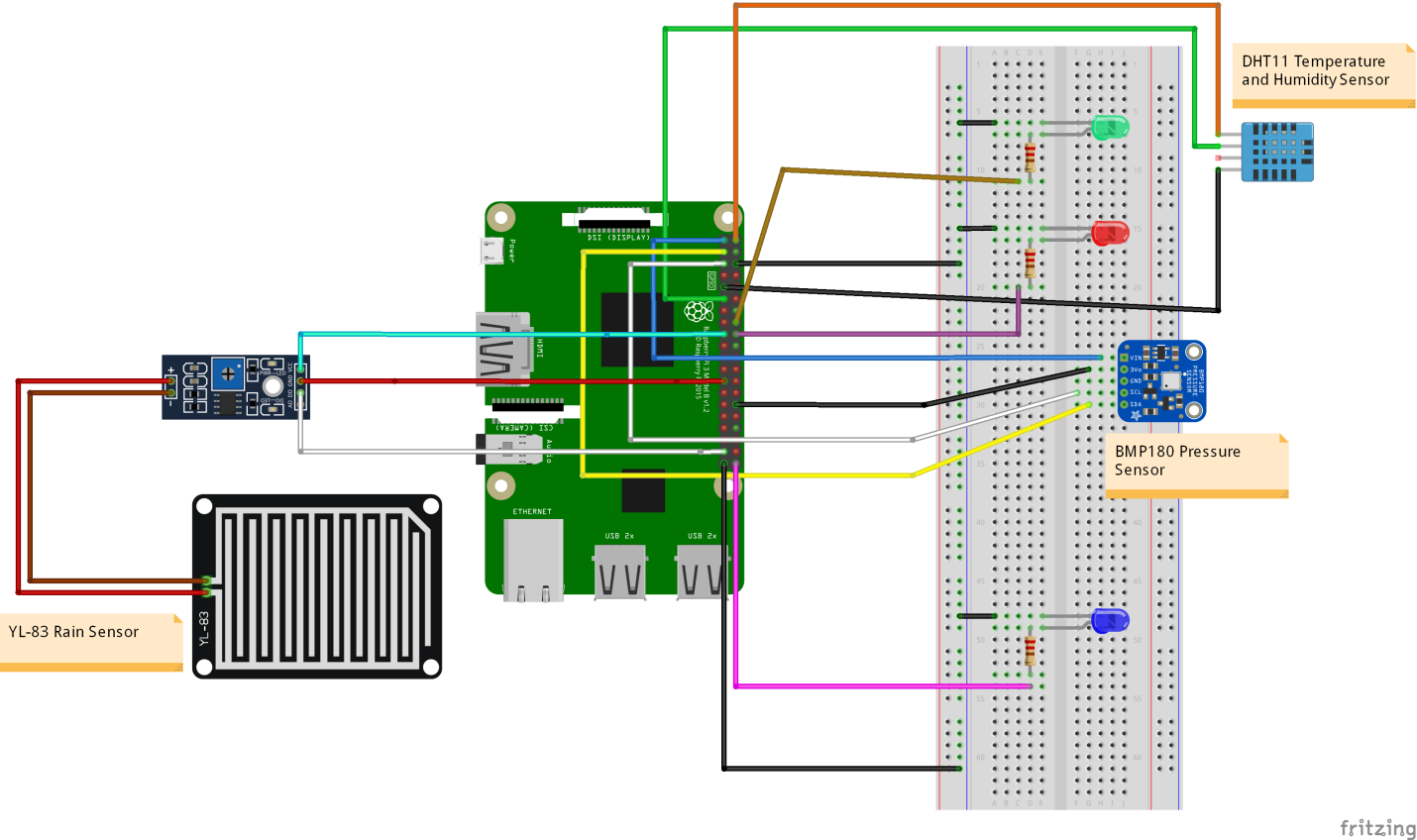
return (temperature,pressure,altitude)

Function Libraries:

Smbus

Ctypes

C\_short

Sensor output: