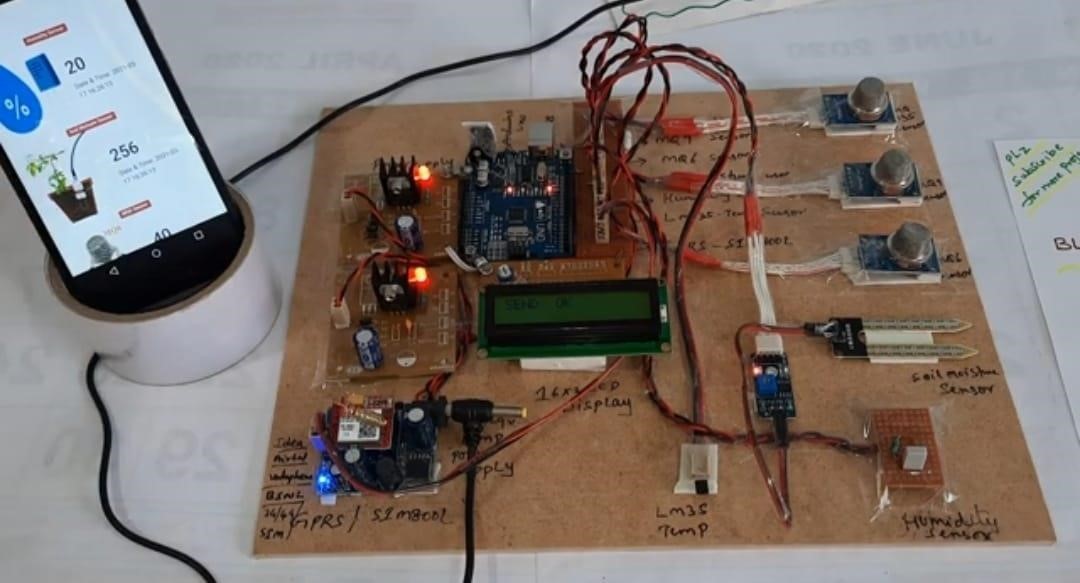
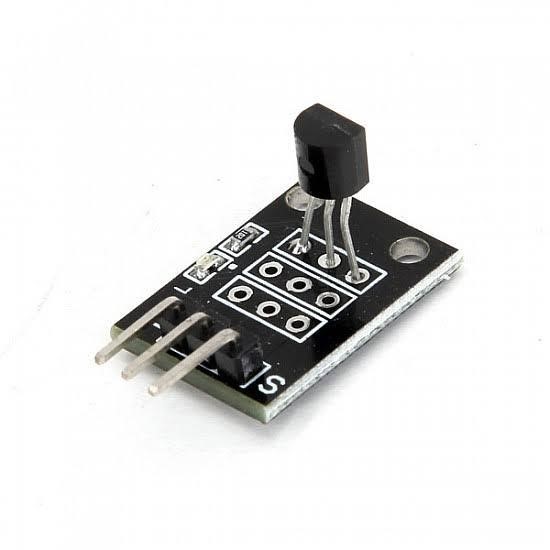
**Environmental monitoring-phase 3**

**PROJECT DESIGN:**



**COMPONENTS :**

**▪︎TEMPARATURE SENSOR :**



**▪︎CAPACITIVE HUMIDITY SENSORS :**



**▪︎MQ 135 SENSOR:**



**▪︎MQ 6 GAS SENSOR :**



**▪︎MQ 9 GAS SENSOR :**



**CODE:**

# Adafruit IO Environmental Monitor for Feather or Raspberry

Pi -

# an internet-enabled environmental monitor

# Import standard python modules import time

# import Adafruit Blinka import board import busio

# import CircuitPython sensor libraries import adafruit\_sgp30 import adafruit\_veml6070

from adafruit\_bme280 import basic as adafruit\_bme280

# import Adafruit IO REST client

from Adafruit\_IO import Client, Feed, RequestError

# loop timeout, in seconds.

LOOP\_DELAY = 10

# Set to your Adafruit IO key.

# Remember, your key is a secret,

# so make sure not to publish it when you publish this code!

ADAFRUIT\_IO\_KEY = 'YOUR\_AIO\_KEY'

# Set to your Adafruit IO username.

# (go to https://accounts.adafruit.com to find your username)

ADAFRUIT\_IO\_USERNAME = 'YOUR\_AIO\_USERNAME'

# Create an instance of the REST client

aio = Client(ADAFRUIT\_IO\_USERNAME, ADAFRUIT\_IO\_KEY)

try: # if we already have the feeds, assign them.

tvoc\_feed = aio.feeds('tvoc') eCO2\_feed = aio.feeds('eco2') uv\_feed = aio.feeds('uv')

temperature\_feed = aio.feeds('temperature') humidity\_feed = aio.feeds('humidity') pressure\_feed = aio.feeds('pressure') altitude\_feed = aio.feeds('altitude') except RequestError: # if we don't, create and assign them.

tvoc\_feed = aio.create\_feed(Feed(name='tvoc')) eCO2\_feed = aio.create\_feed(Feed(name='eco2')) uv\_feed = aio.create\_feed(Feed(name='uv')) temperature\_feed =

aio.create\_feed(Feed(name='temperature'))

humidity\_feed = aio.create\_feed(Feed(name='humidity')) pressure\_feed = aio.create\_feed(Feed(name='pressure')) altitude\_feed = aio.create\_feed(Feed(name='altitude'))

# Create busio I2C

i2c = busio.I2C(board.SCL, board.SDA) # Create VEML6070 object.

uv = adafruit\_veml6070.VEML6070(i2c) # Create BME280 object. bme280 = adafruit\_bme280.Adafruit\_BME280\_I2C(i2c) bme280.sea\_level\_pressure = 1013.25 # Create SGP30 object using I2C.

sgp30 = adafruit\_sgp30.Adafruit\_SGP30(i2c) sgp30.iaq\_init()

sgp30.set\_iaq\_baseline(0x8973, 0x8aae)

# Sample VEML6070 def sample\_VEML(): for \_ in range(10): uv\_raw = uv.uv\_raw return uv\_raw

while True:

print('Reading sensors...') # Read SGP30.

eCO2\_data = sgp30.eCO2 tvoc\_data = sgp30.TVOC

# Read VEML6070.

uv\_data = sample\_VEML()

# Read BME280.

temp\_data = bme280.temperature # convert temperature (C->F) temp\_data = int(temp\_data) \* 1.8 + 32 humid\_data = bme280.humidity pressure\_data = bme280.pressure alt\_data = bme280.altitude

print('sending data to adafruit io...') # Send SGP30 Data to Adafruit IO. print('eCO2:', eCO2\_data) aio.send(eCO2\_feed.key, eCO2\_data) print('tvoc:', tvoc\_data) aio.send(tvoc\_feed.key, tvoc\_data) time.sleep(2)

**# Send VEML6070 Data to Adafruit IO.**

**print('UV Level: ', uv\_data) aio.send(uv\_feed.key, uv\_data) time.sleep(2)**

**# Send BME280 Data to Adafruit IO.**

**print('Temperature: %0.1f C' % temp\_data) aio.send(temperature\_feed.key, temp\_data) print("Humidity: %0.1f %%" % humid\_data) aio.send(humidity\_feed.key, int(humid\_data)) time.sleep(2)**

**print("Pressure: %0.1f hPa" % pressure\_data) aio.send(pressure\_feed.key, int(pressure\_data)) print("Altitude = %0.2f meters" % alt\_data) aio.send(altitude\_feed.key, int(alt\_data)) # avoid timeout from adafruit io time.sleep(LOOP\_DELAY \* 60)**