Development part 1

IOT\_Phase3

Air Quality MONITORING USING IOT



### REquirements:

**Hardware Requirements:**

**Raspberry Pi Pico:**

* Pi Pico is a microcontroller board that can run python scripts.
* It runs on Raspberry OS (previously raspian)

**MQ135 Gas sensor**

* It is a sensor capable of sensing gases and providing output in two ways. (analog and digtal)

**Red and Green LED, Buzzer**

**Software Requirements:**

**Raspberry Pi Pico**

* Raspberry Pi OS previously known as Raspian
* RPi.GPIO (Comes pre-installed with raspian mostly

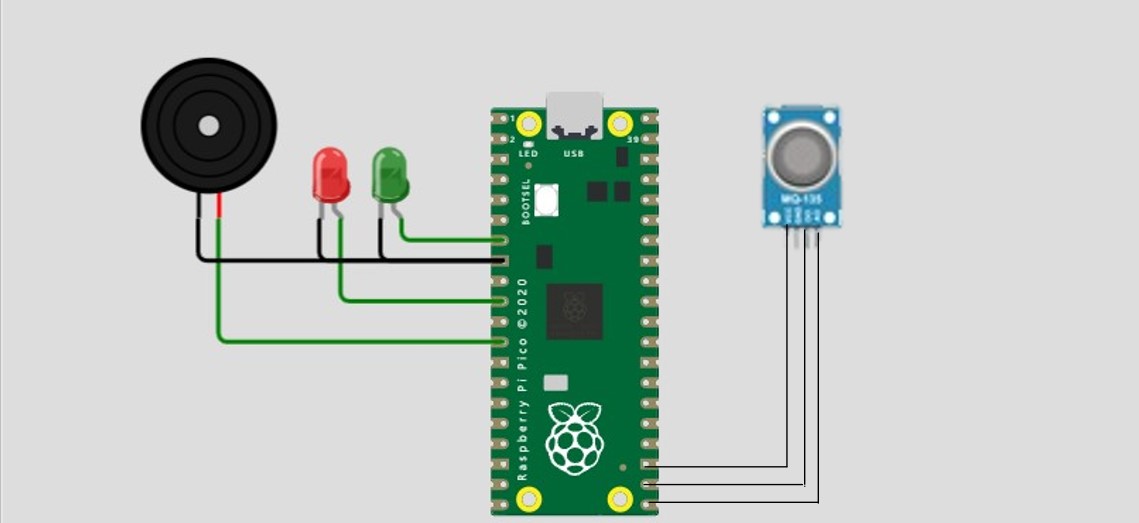
**Circuit COnnections**

LEDs and Buzzer

* Green LED to pin 5
* Red LED to pin 7
* Buzzer (speaker) to pin 9
* All above 3 connected to GND2

MQ135

* MQ135 supply to pin 16
* MQ135 GND to GND5
* MQ135 Analog output (AO) to pin 17

**Circuit schematics**

*Python Script* for **Air Quality detection**

**import requests**

**import time**

**from picozero import Speaker**

**greenled = Pin(5, Pin.OUT)**

**redled = Pin(7, Pin.OUT)**

**speaker = Speaker(9)**

**# Define the URL where you want to send the data**

**POST\_URL = "http://example.com/api/endpoint" # Replace with your actual URL**

**# Define the analog input pin connected to AO**

**ANALOG\_PIN = 17**

**# Define the voltage reference (typically 3.3V for Raspberry Pi)**

**VOLTAGE\_REFERENCE = 3.3**

**# Set up GPIO (for analog input)**

**try:**

**import RPi.GPIO as GPIO**

**GPIO.setmode(GPIO.BCM)**

**except ImportError:**

**pass**

**####### Define air quality thresholds (you can adjust these as needed)**

**RLOAD=10.0**

**RZERO=76.63**

**#Parameters for calculating ppm of CO2 from sensor resistance**

**PARA=116.6020682**

**PARB=2.769034857**

**# Parameters to model temperature and humidity dependence**

**CORA=0.00035**

**CORB=0.02718**

**CORC=1.39538**

**CORD=0.0018**

**# Atmospheric CO2 level for calibration purposes**

**ATMOCO2=397.13**

**############**

**with open(CSV\_FILE, mode='w', newline='') as file:**

**writer = csv.writer(file)**

**writer.writerow(headers)**

**while True:**

**# Read the analog sensor value (0 to 1.0)**

**try:**

**analog\_value = GPIO.input(ANALOG\_PIN) / 1024.0 # 10-bit ADC on Raspberry Pi**

**voltage = analog\_value \* VOLTAGE\_REFERENCE**

**except NameError:**

**voltage = 1.0 # If GPIO is not available, use a placeholder value**

**print(f"Analog voltage: {voltage} V")**

**# Determine air quality based on thresholds**

**# calculations done to caluculate ppm value from analog output.**

**air\_qual=((1023/voltage) \* 5 - 1)\*RLOAD**

**air\_quality = PARA \* ((resistance/RZERO)\*\*(-PARB))**

**if air\_quality < 800:**

**air\_quality\_condition = "Very good quality Air"**

**greenled.toggle()**

**speaker.off()**

**elif(( air\_quality > 800 )&&( air\_quality <2000)):**

**air\_quality\_condition = "Average quality air"**

**redled.toggle()**

**speaker.off()**

**else:**

**air\_quality\_condition = "Bad quality air! Please use masks or get away immediately"**

**speaker.on()**

**print(f"Air Quality: {air\_quality}")**

**# Send the data to the website**

**try:**

**response = requests.post(POST\_URL, json={"air\_quality": air\_quality, "air\_quality\_condition": air\_quality\_condition})**

**response.raise\_for\_status() # Check for HTTP errors**

**print(f"Data sent successfully. Status code: {response.status\_code}")**

**except requests.exceptions.RequestException as e:**

**print(f"Error sending data: {e}")**

**time.sleep(200) # Adjust this delay as needed**

**#saving data in a file for predictive analytics**

**timestamp = time.time()**

**row = [timestamp, air\_quality]**

**writer.writerow(row)**

**read\_sensor()**

*Python Script* for **Predictive Analytics:**

**import pandas as pd**

**from sklearn.linear\_model import LinearRegression**

**df = pd.read\_csv('sensor\_data.csv')**

**X = df[['Timestamp']]**

**y = df['Sensor Value']**

**model = LinearRegression()**

**model.fit(X, y)**

**X\_test = '''your predction value (timestamp) here '''**

**y\_pred = model.predict(X\_test)**

**print(y\_pred)**