Module Code: 6NTCM009W Internet of Things (IoT)

Assessment Title: Coursework – An IoT prototype

[redacted]

[redacted]

A. Development

1. Data Collection from BME280 Sensor

The objective of this task was to write a Python script for the Raspberry Pi Pico W, to demonstrate collecting data from the BOSCH BME280 temperature, humidity, and pressure sensor, on the command Shell.

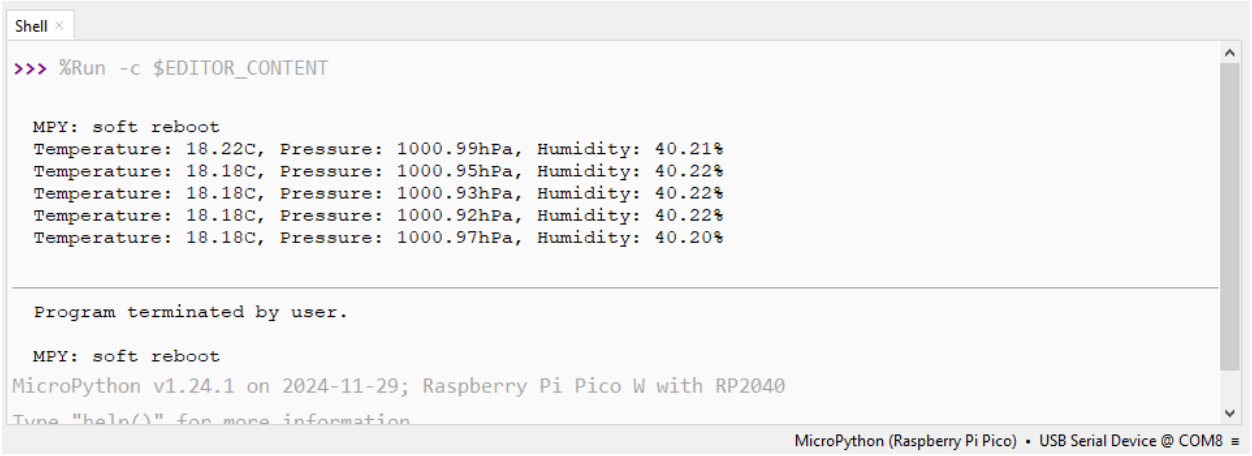

Figure 1: IOTCW_PARTA_1.py command shell

This Python script reads temperature, pressure, and humidity data from a **BME280 sensor** using **I2C communication** and prints the values every **5 seconds**. It first imports necessary modules (machine, time, bme280), then initializes the I2C bus with **SDA on Pin 0** and **SCL on Pin 1** at **400 kHz**. The **BME280 sensor** is set up with bme280.BME280(i2c=i2c), allowing data retrieval. Inside a loop, the script reads sensor values from bme.values and displays them. A **KeyboardInterrupt** exception ensures safe termination. The I2C interface enables the microcontroller to send commands and receive sensor data efficiently.

2. Web Server for Sensor Readings

The objective of this task was to modify the Python script to act as web server holding the sensor readings, using either the home Wi-Fi router or a mobile as a hotspot. Create a web page to serve for the sensor readings.

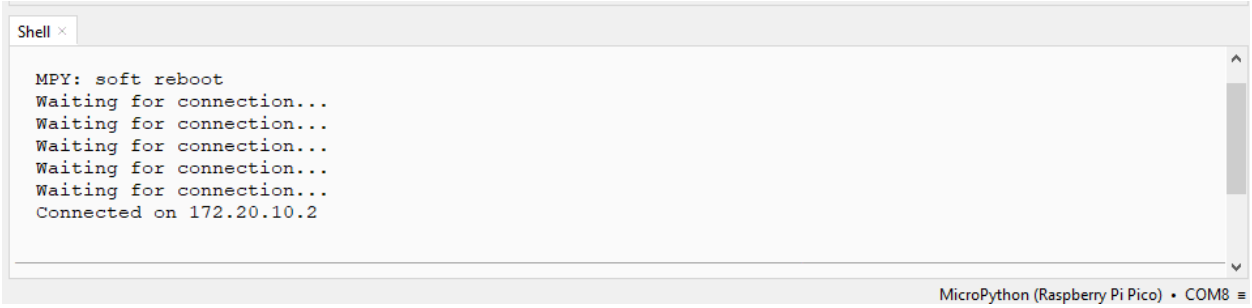

Figure 2: Connection Established Shell

Figure 3: A Web Page with Sensor Readings

This updated script improves the original **BME280 sensor** setup by adding **Wi-Fi connectivity and a web server**, so you can access sensor data remotely. It connects to Wi-Fi using the connect() function, which turns on Wi-Fi, logs in with the given credentials, and gets an IP address. The open_socket(ip) function sets up a **server on port 80**, allowing web clients to connect. The serve(connection) function waits for incoming requests, reads sensor data, creates an **HTML page** using webpage(reading), and sends it to the client. The page **refreshes every 10 seconds**, so the data updates in real-time. Instead of just printing sensor values in the console, you can now view them **from any device on the network**. The script also includes **error handling**.

3. **Cloud-Based Data Logging with Google Sheets**

The objective of this task was to modify the Python script to act as a serverless cloud-based application by logging sensor data to Google Sheets using App Script.

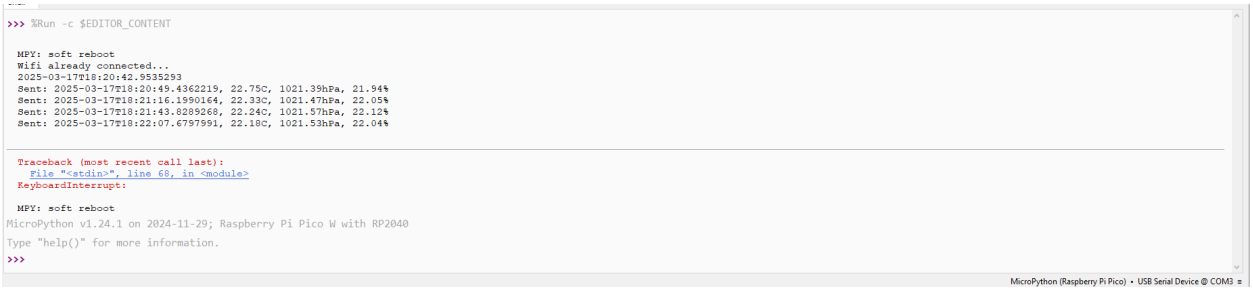

Figure 4: Shell display of connection and sensor data

	A	B	C	D	E
1	Time	Temperature (°C)	Pressure (hPa)	Humidity (%)	Counter
2	2025-03-17 17:47:25	24.87	1020.94	20.04	23
3	2025-03-17 17:47:43	24.5	1021.05	20.09	2
4	2025-03-17 17:48:24	24.8	1020.94	20.04	3
5	2025-03-17 17:48:44	24.47	1021.03	20.16	4
6	2025-03-17 17:49:10	24.42	1021	20.29	5
7	2025-03-17 17:49:32	24.4	1021.08	20.15	6
8	2025-03-17 17:49:50	24.44	1021.09	20.14	7
9	2025-03-17 17:50:08	24.48	1021.05	20.14	8
10	2025-03-17 17:50:25	24.48	1021.03	20.24	9
11	2025-03-17 17:50:47	24.47	1021	20.19	10
12	2025-03-17 18:12:53	24.55	1021.06	19.87	11
13	2025-03-17 18:20:49	22.75	1021.39	21.94	12
14	2025-03-17 18:21:16	22.33	1021.47	22.05	13
15	2025-03-17 18:21:44	22.24	1021.57	22.12	14
16	2025-03-17 18:22:08	22.18	1021.53	22.04	15
17					16
18	2025-03-18 17:15:44	20.11	1019.15	31.48	17
19	2025-03-18 17:16:45	19.41	1019.23	31.86	18
20	2025-03-18 17:17:06	18.86	1019.45	32.06	19
21	2025-03-18 17:17:31	18.66	1019.39	32.77	20
22	2025-03-18 17:17:53	18.6	1019.4	33.07	21
23	2025-03-18 17:18:50	19.85	1019.22	34.45	22
24	2025-03-18 17:19:12	19.84	1019.21	34.31	23

Figure 5: Google Sheet of BME280 Sensor Readings of temperature, pressure and humidity

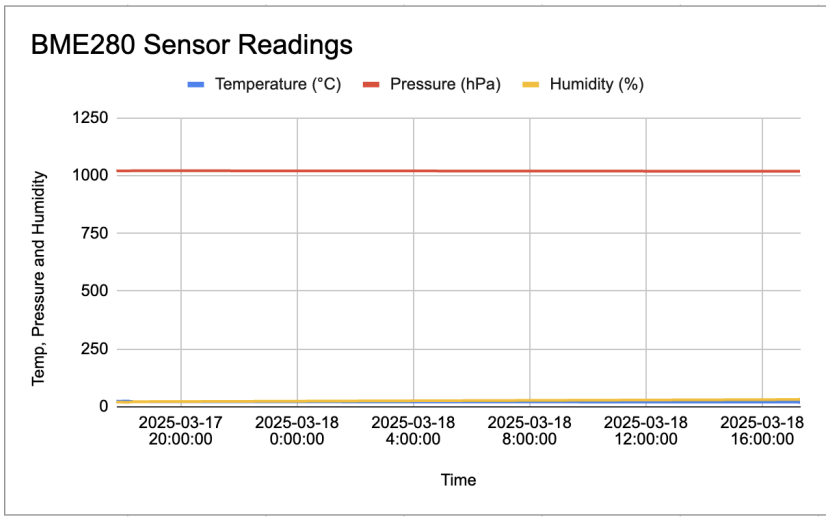

Figure 6: BME280 Sensor Readings of temperature, pressure and humidity

This script sends BME280 sensor data to Google Sheets using HTTP GET requests and a Google Apps Script web app. The Raspberry Pi Pico W connects to Wi-Fi via connectWifi(), ensuring network access. The function getTime() fetches the current timestamp from an online API. Sensor data, including temperature, pressure, and humidity, is read from the BME280 sensor, converted into numerical values, and sent to Google Sheets using send_to_sheet(), which constructs a URL with query parameters and makes an HTTP GET request to the Apps Script web app (SCRIPT_URL). The Apps Script function doGet(e) extracts data from the request, finds the next available row using a counter in E2, and logs the values into columns A-D. The counter is updated to track entries. The system sends 20

sensor readings at 15-second intervals before turning off the onboard LED. This setup enables automated data logging, allowing real-time sensor data storage and analysis in Google Sheets.

The graph in Figure 6 shows that temperature decreases over time, starting around 24-25°C and dropping to 19-20°C, likely indicating a transition from day to night over two days at around the same time. Humidity increases as temperature drops, suggesting an inverse relationship where cooler air holds more moisture. Pressure remains relatively stable, fluctuating slightly around 1020 hPa before settling near 1019 hPa. These trends suggest typical atmospheric changes, where cooler temperatures lead to higher humidity, while pressure shows minor variations across both days.

B. Testing

1. Prototyping Development

Figure 7: Prototype Testing

In order to connect the hardware, I connected the BME280 sensor to the Raspberry Pi Pico W via I2C. Then I ensured that the Wi-Fi credentials, including ssid and password, were correctly set in the code. I powered the Pico W via USB instead of power bank.

For the software development, I opened Thonny IDE and then connected to ‘Micropython (Raspberry Pi Pico) USB Serial Device’ as visible in Figures 1,2 and 4.

My testing methodology includes several use cases. Firstly, I verified that the Pico W read the sensor data and printed the values into the shell. Secondly, I checked if the device was connected to Wi-Fi and displayed the IP address on the shell. Thirdly, I opened an HTML page on Google to confirm real-time sensor updates were displayed. Fourthly, I checked if the sensor readings were successfully logged in Google Sheets through the HTTP GET requests.

2. **Data Visualization for Temperature**

Data visualization allows us to have an understanding of temperature trends across different conditions and times of the day. A line graph helps to identify patterns, fluctuations, and anomalies over time. By taking readings from a range of scenarios (indoor/outdoor, day/night) and displaying them on a graph, we can observe how temperature changes in response to environmental factors.

	A	B	C	D
1	Time	Temperature	Counter	Scenario
2	2025-03-17 19:18:45	23.52	18	Night/Indoor
3	2025-03-17 19:19:10	23.25	2	Night/Indoor
4	2025-03-17 19:19:32	23.3	3	Night/Indoor
5	2025-03-17 19:20:04	23.32	4	Night/Indoor
6				
7				
8	2025-03-18 16:52:15	18.26	7	Day/Outdoor
9	2025-03-18 16:53:35	17.49	8	Day/Outdoor
10	2025-03-18 16:54:09	17.28	9	Day/Outdoor
11	2025-03-18 16:54:57	15.38	10	Day/Outdoor
12	2025-03-18 16:55:21	14.31	11	Day/Outdoor
13	2025-03-18 16:55:53	13.29	12	Day/Outdoor
14	2025-03-18 16:58:19	17.61	13	Day/Indoor
15	2025-03-18 16:58:50	17.71	14	Day/Indoor
16	2025-03-18 16:59:24	18.09	15	Day/Indoor
17	2025-03-18 16:59:48	18.31	16	Day/Indoor
18	2025-03-18 17:00:30	18.67	17	Day/Indoor
19	2025-03-18 17:00:52	18.85	18	Day/Indoor

Figure 8: Google Sheet displaying temperature during different scenarios and times.

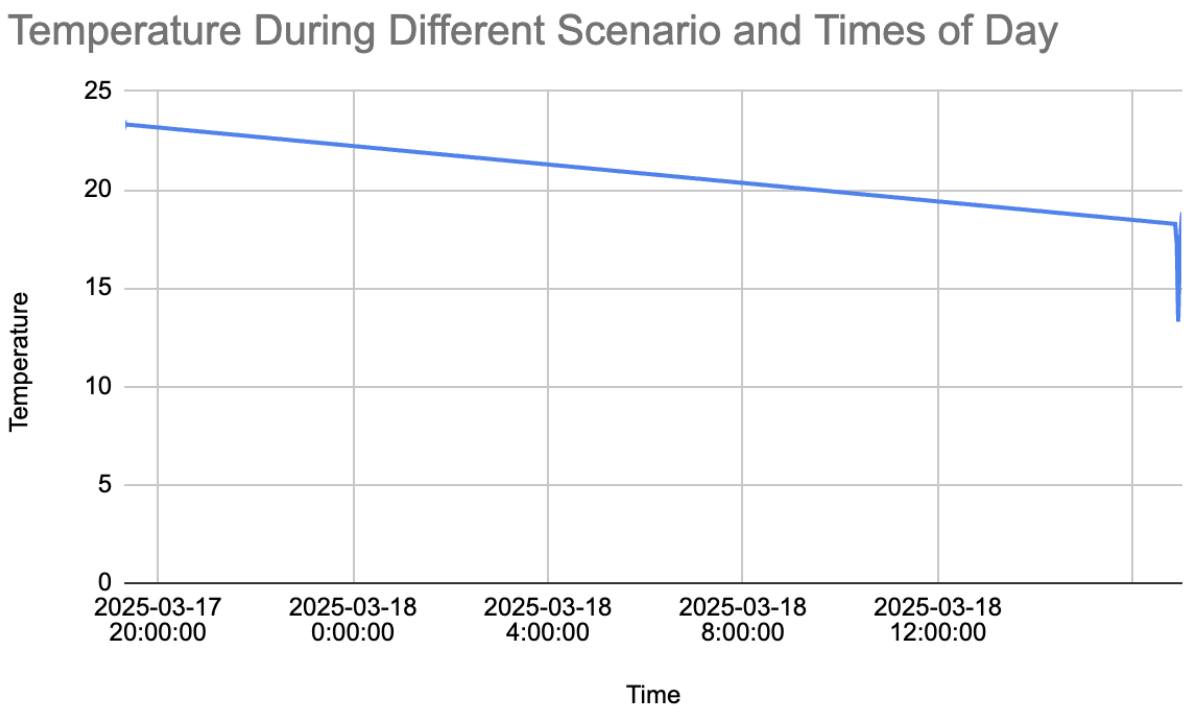

Figure 9: Line chart displaying temperature during different scenarios and times.

The sensor was placed in different environments, including indoor at night, outdoor during the day, and indoor during the day.

In Figure 9, the graph displays an overall trend that shows a gradual decrease in temperature over time. Initially, the the readings display a higher temperature at around 24°C which was taken at nightime indoors. The readings are stable. However, over time, in the daytime outdoor scenario, the temperature decreases more significantly towards 18°C-19°C. There is a sharp decline at the end of the graph which suggests a notable change in temperature. This might be due to the fact that switching from indoor to outdoor conditions causes a higher shift in temperature.

It can be concluded that indoor results are more stable and outdoor temperature decrease as the day progresses.

3. Connection Loss Handling

Figure 10: Connection loss handling method

In my code, I handle connection loss through a few mechanisms.

Firstly, I checked for a WiFi connection before running the script. For example, the function connect_wifi() ensured the device connects before starting data collection. If the device is not connected, it repeatedly attempts to establish a connection.

```
def connect_wifi():
    wlan.active(True)
    if not wlan.isconnected():
        wlan.connect(ssid, password)
        while not wlan.isconnected():
            print("Connecting to WiFi...")
            time.sleep(2)
            board_led.off()
        board_led.on()
        print("Connected. IP:", wlan.ifconfig()[0])
```

Figure 11: connect_wifi() function

Secondly, the send_to_sheet() function attempts to send data via an HTTP request. If can error occurs, an exception is caught an error is printed. This causes the script to continue running instead of crashing.

```
def send_to_sheet(time, temp):
    try:
        url = f"{SCRIPT_URL}?time={time}&temp={temp}"
        response = urequests.get(url, timeout=10)
        print("Response:", response.text)
        response.close()
        gc.collect()
        return True
    except Exception as e:
        print("Send Error:", e)
        gc.collect()
        return False
```

Figure 12: send_to_sheet() function

Lastly, if the send_to_sheet() fails, the script waits 10 seconds before retrying

```
    success = send_to_sheet(current_time, temp)
    if success:
        print(f"Sent: {current_time}, {temp}°C")
    else:
        print("Retrying...")
        time.sleep(10)
```

Figure 13: Re-try code snippet

4. **Enhancements for Air Pollution Monitoring**
   To turn this prototype into an IoT air pollution monitoring system, a few key upgrades are needed. First, more sensors should be added, like PM2.5/PM10 for dust, gas sensors (MQ-135) for harmful gases, and an ozone sensor, to measure air quality properly. Second, instead of just using Google Sheets, the data should be sent to a cloud platform like AWS IoT or ThingsBoard, so it can be viewed in real-time, stored long-term, and trigger alerts if pollution levels get too high. Lastly, it should be more power-efficient, using solar panels or batteries, and process some data on the device itself to reduce internet use. These changes would make the system smarter, more reliable, and ready for real-world air quality monitoring.

Appendix

- Source Code Listing:
  - Filename: [redacted].txt

Student name: [redacted]
Student ID: [redacted]

# Filename: IOTCW_PARTA_1.py
import machine
from machine import Pin, I2C
import time

```
import bme280

i2c=I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)

bme = bme280.BME280(i2c=i2c)

try:
    while True:
        temp = bme.values[0]
        pressure = bme.values[1]
        humidity = bme.values[2]

        print(f"Temperature: {temp}, Pressure: {pressure}, Humidity: {humidity}")
        time.sleep(5)

except KeyboardInterrupt:
    print("Program terminated by user.")

# Filename: IOTCW_PARTA_2.py
import machine
from machine import Pin, I2C
import network
import socket
import time
import bme280

ssid = [redacted]
password = [redacted]

i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)
bme = bme280.BME280(i2c=i2c)

def connect():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    while not wlan.isconnected():
        print('Waiting for connection...')
        time.sleep(1)
    ip = wlan.ifconfig()[0]
    print(f'Connected on {ip}')
    return ip

def open_socket(ip):
```

```
    address = (ip, 80)
    connection = socket.socket()
    connection.bind(address)
    connection.listen(1)
    return connection

def webpage(reading):
    html = f"""
    <!DOCTYPE html>
    <html>
    <head>
    <title>Pico W BME280 Sensor</title>
    <meta http-equiv="refresh" content="10">
    </head>
    <body>
    <h1>BME280 Sensor Readings</h1>
    <p>{reading}</p>
    </body>
    </html>
    """
    return str(html)

def serve(connection):
    while True:
        client = connection.accept()[0]
        request = client.recv(1024)
        request = str(request)

        temp = bme.values[0]
        pressure = bme.values[1]
        humidity = bme.values[2]

        reading = f"Temperature: {temp}, Pressure: {pressure}, Humidity: {humidity}"
        html = webpage(reading)
        client.send(html)
        client.close()

try:
    ip = connect()
    connection = open_socket(ip)
    serve(connection)
except KeyboardInterrupt:
    machine.reset()
```

```
# Filename: IOTCW_PARTA_3_2.py
import machine
import time
from machine import Pin, I2C
import network
import urequests
import json
import gc
from bme280 import BME280

wlan = network.WLAN(network.STA_IF)
board_led=machine.Pin("LED", machine.Pin.OUT)
ssid = [redacted]
password = [redacted]

SCRIPT_URL =
"https://script.google.com/macros/s/AKfycbzRaU8Y5SBSsduLzI-zzcLS3ONQUiwcPtZWjnDYvg5
ZqOrXFk6xfx_P7lz07Y5Gqiop/exec"
TIME_URL = "https://timeapi.io/api/Time/current/zone?timeZone=Europe/London"

i2c = I2C(0, scl=Pin(1), sda=Pin(0))
bme = BME280(i2c=i2c)

def getTime():
    res=urequests.get(url=TIME_URL)
    time=json.loads(res.text)["dateTime"]
    res.close()
    return time

def connectWifi():
    wlan.active(True)
    if not wlan.isconnected():
        wlan.connect(ssid, password)
        while not wlan.isconnected() and wlan.status() >= 0:
            print("Waiting to connect: ")
            time.sleep(1)
            board_led.off()
        board_led.on()
        print(wlan.ifconfig())
    else:
        print("Wifi already connected...")
        print(getTime())
```

```
def send_to_sheet(time, temp, pressure, humidity):
    try:
        url =
f"{SCRIPT_URL}?time={time}&temp={temp}&pressure={pressure}&humidity={humidity}"
        response = urequests.get(url)
        response.close()
        gc.collect()
    except Exception as e:
        print("Failed to send data:", e)


connectWifi()
wlan.active(True)

for i in range(20):
    timestamp = f"{getTime()}"
    temp_str, pressure_str, humidity_str = bme.values
    temp = float(temp_str[:-1])
    pressure = float(pressure_str[:-3])
    humidity = float(humidity_str[:-1])

    send_to_sheet(timestamp, temp, pressure, humidity)
    print(f"Sent: {timestamp}, {temp}C, {pressure}hPa, {humidity}%")
    time.sleep(15)

board_led.off()
```

```
// Student Name: [redacted]
// Student ID: [redacted]


// Filename: IOTCW_PARTA_3_2_AppScript


var SHEET_NAME = "pico2";
var COUNTER_CELL = "E2"; // Store counter in E2 (outside the data range)


function doGet(e) {
 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(SHEET_NAME);
 var lastLog = sheet.getRange(COUNTER_CELL).getValue() || 0;
 lastLog++;
```

```
// Start logging from row 2 (skip header in row 1)
var targetRow = lastLog + 1;

// Update counter in E1
sheet.getRange(COUNTER_CELL).setValue(lastLog);

// Write data to columns A-D
sheet.getRange(`A${targetRow}`).setValue(e.parameter.time);
sheet.getRange(`B${targetRow}`).setValue(e.parameter.temp);
sheet.getRange(`C${targetRow}`).setValue(e.parameter.pressure);
sheet.getRange(`D${targetRow}`).setValue(e.parameter.humidity);
sheet.getRange(`E${targetRow}`).setValue(lastLog); // Entry counter

return ContentService.createTextOutput("Data logged successfully");
}
```

```
# Filename: IOTCW_PARTB_2.py

import machine
import time
from machine import Pin, I2C
import network
import urequests
import json
import gc
from bme280 import BME280

wlan = network.WLAN(network.STA_IF)
board_led = Pin("LED", Pin.OUT)
ssid = [redacted]
password = [redacted]
SCRIPT_URL =
"https://script.google.com/macros/s/AKfycbwXG8LdWe7FF8dfrYfcQUPfOHB41k1K5_o5VPqhZ
Uel8HuEWr8k_X_eFNHEOE3rDqw_/exec"
TIME_URL = "https://timeapi.io/api/Time/current/zone?timeZone=Europe/London"

i2c = I2C(0, scl=Pin(1), sda=Pin(0))
bme = BME280(i2c=i2c)
```

```
def get_time():
    try:
        response = urequests.get(TIME_URL, timeout=5)
        time_str = json.loads(response.text)["dateTime"]
        response.close()
        return time_str
    except Exception as e:
        print("TimeAPI Error:", e)
        return None

def connect_wifi():
    wlan.active(True)
    if not wlan.isconnected():
        wlan.connect(ssid, password)
        while not wlan.isconnected():
            print("Connecting to WiFi...")
            time.sleep(2)
            board_led.off()
        board_led.on()
        print("Connected. IP:", wlan.ifconfig()[0])

def send_to_sheet(time, temp):
    try:
        url = f"{SCRIPT_URL}?time={time}&temp={temp}"
        response = urequests.get(url, timeout=10)
        print("Response:", response.text)
        response.close()
        gc.collect()
        return True
    except Exception as e:
        print("Send Error:", e)
        gc.collect()
        return False

connect_wifi()

for _ in range(20):
    current_time = get_time()
    if not current_time:
        print("Failed to fetch time. Retrying...")
        time.sleep(5)
        continue
```

```
    temp_str, _, _ = bme.values
    temp = float(temp_str[:-1])

    success = send_to_sheet(current_time, temp)
    if success:
        print(f"Sent: {current_time}, {temp}°C")
    else:
        print("Retrying...")
        time.sleep(10)

    time.sleep(15)

board_led.off()
```

```
// Student Name: [redacted]
// Student ID: [redacted]

// Filename: IOTCW_PARTB_2_AppScript

var SHEET_NAME = "pico3";
var COUNTER_CELL = "C2"; // Counter stored in C1

function doGet(e) {
 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(SHEET_NAME);
 var lastLog = sheet.getRange(COUNTER_CELL).getValue() || 0;
 lastLog++;

 var targetRow = lastLog + 1; // Data starts at row 2 (header in row 1)
  sheet.getRange(COUNTER_CELL).setValue(lastLog); // Update counter in C1
  // Write data to columns A, B, and C
 sheet.getRange(`A${targetRow}`).setValue(e.parameter.time);
 sheet.getRange(`B${targetRow}`).setValue(e.parameter.temp);
 sheet.getRange(`C${targetRow}`).setValue(lastLog); // Entry counter
  return ContentService.createTextOutput("OK");
}
```