**IoT Worksheet 3 – Webserver to send Realtime sensory Data**

Aim of this project is to send sensory data from the microbit , in this example the light level, to a simple web server in real time. This project involves three things, the microbit, and its serial data connection and the webserve.

How it works:

1. The microbit sends data through uart module which makes use of the serial usb connection
2. The data is received through the python pyseries module by making a connection to the usb port. (PLEASE NOTE THAT you need to change the ‘COM3’ to your usb port. To do this go to device manager and go to Ports and check which one has Microsoft one then change accordingly )
3. The data is received from a flask application. Flask is used to create a webserver
4. Flask then updates the html in real time with the sensory data received
5. The light level is displayed on your webpage

Explanation of the code:

1. **Importing Libraries:**

A black screen with white text

Description automatically generated

The code starts by importing necessary libraries. **serial** is used for serial communication with the Micro:bit, **Flask** is the web framework, **render\_template** is used to render HTML templates, and **SocketIO** is used for real-time communication. The **threading** module is used for thread synchronization.

1. **Creating Flask App and SocketIO Instance:**

A black background with white text

Description automatically generated

Here, a Flask app instance is created, and a SocketIO instance is initialized with the app. The **cors\_allowed\_origins** parameter is set to allow cross-origin requests from any origin.

1. **Serial Connection:**

A black background with white text

Description automatically generated

This line establishes a serial connection to the Micro:bit, assuming it is connected to the COM3 port at a baud rate of 115200.

1. **Background Task:**

A screen shot of a computer program

Description automatically generated

This function runs in the background as a separate thread. It constantly reads lines from the serial connection with the Micro:bit, decodes them using UTF-8 encoding, and sends the data to connected clients using SocketIO's **emit** function. The event name used here is **'data\_update'**, and the data is sent as a dictionary with a **'data'** key.

1. **Flask App Route:**

A screen shot of a computer

Description automatically generated

This route handles the root URL ("/"). It renders an HTML template named "index.html". This template contains the client-side code that will display the real-time sensor data.

1. **SocketIO Connection:**

A screen shot of a computer program

Description automatically generated

This is a SocketIO event handler that's triggered when a client connects. It starts the background task (defined earlier) in a separate thread to continuously read data from the Micro:bit and emit it to the connected clients.

1. **Main Block:**

A computer screen shot of a number

Description automatically generated

This block ensures that the application is run only when the script is executed directly (not imported as a module). It starts the Flask-SocketIO server with the specified host and port.

Overall, this code creates a web application that communicates with a Micro:bit through serial communication and sends the sensor data to connected clients in real-time using Flask and SocketIO. The web page has JavaScript code that listens for the **'data\_update'** event and updates the displayed data accordingly.

A screen shot of a computer code

Description automatically generated

In the JS it connects to the socker server and receives updated data

Here is the microbit code

A screen shot of a computer

Description automatically generated

The threading in python helps to get the real time connection needed without making the code blocking

**Video**

I have included a video filed named sensor data video, where it shows the microbit capturing the light level and the data is sent to webserver in real time

**How to Run**

* install the necessary libraries needed
* Create a virtual environment

python -m venv venv # Create a virtual environment

source venv/bin/activate # Activate the virtual environment (Linux/macOS)

# OR

venv\Scripts\activate # Activate the virtual environment (Windows)

* pip install Flask
* flash the microbit code
* once its done to run the app, run the code
* python app.py