**Running the datalogger/magic wand demo using WIFI**

This document shows how to set up a wifi router and TCP server on a host PC for the IOT SAMD21 Xplained pro (with winc1500) to stream data.

**Setting up wifi router**

* Procure a wifi router and setup SSID to “MCHP\_demo\_WINC”
* Setup security to WPA2\_PSK and set the password to “12345678”
* Setup LAN to 192.168.0.255

The TP-Link Archer AC1900 wifi router I purchased was for ~$22 from Walmart.

**Setting up TCP\_Server**

The TCP server will listen to incoming client socket connection and receive data. This data will be streamed to a virtual COM Port.

* Connect an FTDI cable to the host PC and loop TX & RX
* Connect the host PC to “MCHP\_demo\_WINC” SSID and ensure that the ip address is set to 191.168.0.X
* In the python script in appendix A change the COM port by modifying the following lines of code:

ser = serial.Serial('**/dev/ttyUSB0'**, 115200, timeout=1)

On Windows, you would change it to:

ser = serial.Serial('**/dev/COMx**, 115200, timeout=1)

* Run the python script by typing ‘python3 tcp\_server.py’
* Open a serial console with the same COM port

With this you will be able to see the streamed data on the COM port console.

**Appendix A – Python code for TCP server**

|  |
| --- |
| import socket import serial  def start\_tcp\_server(host='0.0.0.0', port=12345):     # Create a TCP/IP socket     server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)      # Bind the socket to the address and port     server\_socket.bind((host, port))      # Listen for incoming connections     server\_socket.listen(5)     print(f"Server listening on {host}:{port}")     ser = serial.Serial('/dev/ttyUSB0', 115200, timeout=1)         while True:         # Wait for a connection         client\_socket, client\_address = server\_socket.accept()         print(f"Connection from {client\_address}")          try:             while True:                 # Receive data from the client                 data = client\_socket.recv(1024)                 if data:                     ser.write(data)                     #print(data)                     # Echo the data back to the client                     #client\_socket.sendall(data)                 else:                     break         finally:             # Clean up the connection             client\_socket.close()             ser.close()             print(f"Connection with {client\_address} closed")  if \_\_name\_\_ == "\_\_main\_\_":     start\_tcp\_server() |