

# **Remote Data Transmission Experiments based on Arduino MKR WiFi 1010, Espressif ESP32-S2 Wi-Fi MCU, and SparkFun ESP8266 Thing Dev Board**

## **Test Log**

Yi Zhu

Department of Electrical and Computer Engineering

McGill University

yi.zhu6@mail.mcgill.ca

## **Acknowledgement**

This test log is generated for remote data transmission tests performed during Fall 2020 ECSE496 Honours Lab Rotation 3, in which equipment is provided by Mahshid Lab in the Department of Bioengineering at McGill University. This test log is **neither** a formal technical report **nor** a deliverable required by any course. The purpose of this test log is to provide device information and development insight for researchers from Mahshid Lab working on Lab-on-Smartphone or other projects. Regarding specific questions and further information, please contact the author.

December 15, 2020

# Contents

<b>1. Introduction.....</b>	<b>2</b>
<b>2. Background .....</b>	<b>2</b>
<b>2.1 Message Queuing Telemetry Transport.....</b>	<b>2</b>
<b>2.2 Local Access Point.....</b>	<b>2</b>
<b>2.3 Bluetooth Low Energy .....</b>	<b>3</b>
<b>2.4 Arduino Integrated Development Environment .....</b>	<b>3</b>
<b>3. Experimental and Results .....</b>	<b>3</b>
<b>3.1 Arduino MKR WiFi 1010.....</b>	<b>3</b>
3.1.1 IDE Configuration.....	3
3.1.2 Test Board.....	4
3.1.3 LED Control and Data Transmission via Blynk Mobile Phone Application .....	4
3.1.4 Secure Data Transmission to AWS IoT Core Cloud .....	7
3.1.5 Create Local Access Point .....	10
3.1.6 LED Control and Data Transmission using Bluetooth Low Energy.....	12
<b>3.2 Espressif ESP32-S2 Wi-Fi MCU.....</b>	<b>15</b>
3.2.1 IDE Configuration.....	15
3.2.2 Test Board.....	16
3.2.3 Create Local Access Point .....	17
<b>3.3 SparkFun ESP8266 Thing Dev Board .....</b>	<b>19</b>
3.3.1 IDE Configuration.....	20
3.3.2 Test Board.....	20
3.3.3 Create Local Access Point .....	21
<b>4. Other Transmission Protocols .....</b>	<b>23</b>
<b>5. Conclusion .....</b>	<b>23</b>
<b>References.....</b>	<b>24</b>

# **1. Introduction**

Various methods could be utilized for wireless data transmission, locally or globally. Depended on the requirement of the Internet, these techniques could be generally cataloged into two groups: Cloud Server-based approach and local Access Point-based approach. In this test log, experiments performed on both are introduced. Data transmission experiments using Wi-Fi technology were conducted on three development boards, namely Arduino MKR WiFi 1010, Espressif ESP32-S2 Wi-Fi MCU, and SparkFun ESP8266 Thing Dev Board. Background information on these three boards is given in section 3.1, section 3.2, and section 3.3, respectively. Some explorations on Bluetooth technology and other short-distance wireless data transmission methods are also illustrated. Useful documentation of additional methodologies could be found in section 4.

## **2. Background**

### **2.1 Message Queuing Telemetry Transport**

In terms of data transmission using the Internet, cellular data and Wi-Fi networks are the most common techniques adopted by mobile devices. In this test log, the Wi-Fi technique is concentrated. Message Queuing Telemetry Transport (MQTT) is a type of machine-to-machine connectivity protocol that provides message transportation [1]. In this technique, sensor terminals, such as the COVID-19 testing machine, are connected to a Cloud Server, namely an MQTT broker, using MQTT protocol based on the Wi-Fi network. User devices, such as a cellphone, are connected to the MQTT broker using any protocols, such as Wi-Fi networks or cellular data. MQTT protocol provides messaging subscription and publication transport, in which sensor terminals can publish data to the broker with a topic, while user devices can download data by subscribing to the specific topic accordingly, vice versa. Detailed steps of data transmission experiments using the MQTT protocol on the Arduino MKR WiFi 1010 are given in section 3.1.3 and section 3.1.4.

### **2.2 Local Access Point**

A local Access Point (AP) is generally the “hotspot” of a local wireless network, in which the connection to the Internet is not required. For example, a Wi-Fi network is created on the sensor terminals, such as the COVID-19 testing machine. By connecting user devices, such as cellphones, to the sensor terminals through Wi-Fi, users could directly communicate with the sensor terminals.

The establishment of local AP was conducted on all three development boards. Detailed steps of data transmission experiments using the local AP are given in section 3.1.5, section 3.2.3, and section 3.3.3, respectively.

## **2.3 Bluetooth Low Energy**

Bluetooth Low Energy (BLE), different from Wi-Fi, is a Bluetooth communication technology designed for low-power devices and has lower interference effects in data transmission [2]. With a wide adaption in healthcare devices, Bluetooth technology provides a low-cost data transmission solution. Benefit from the integrated BLE connectivity on the Arduino MKR WiFi 1010 development board, data transmission tests based on BLE was performed. Detailed experiment steps are given in section 3.1.6.

## **2.4 Arduino Integrated Development Environment**

Arduino Software (IDE) is an open-source program development platform that supports C and C++ languages. Benefit from its high compatibility with various selections of hardware devices, Arduino IDE is adopted to write codes for programming three data transmission development boards. By connecting boards to a workstation using the Universal Serial Bus (USB) cable, users are able to upload programs from Arduino IDE to boards easily. An integrated serial monitor in this IDE also offers a convenient real-time data transmission monitor functionality.

# **3. Experimental and Results**

## **3.1 Arduino MKR WiFi 1010**

The Arduino MKR WiFi 1010 consists of three main blocks: a low power Arm Cortex-M0 32-bit SAMD21 processor; a U-BLOX NINA-W10 module that provides 2.4GHz Wi-Fi and Bluetooth connectivity; an ATECC508A CryptoAuthentication chip for secure encrypted communication. Besides, this board also provides a broad selection of Input/Output interfaces and a USB port for power supply and serial data transmission. Further information about this board could be found in [3].

### **3.1.1 IDE Configuration**

First of all, the configuration of the Arduino MKR WiFi 1010 board needs to be done for further programming in the Arduino IDE. Steps are as following:

- a. In Arduino IDE, on the menu bar, click on *Tools* menu and then select *Board: "xxx" > Boards Manager...*

- b. Search for *Arduino SAMD Boards (32-bits ARM Cortex-M0+)*.
- c. Select the latest version and then click *Install*.
- d. In Arduino IDE, on the menu bar, click on *Tools > Board > Board: "xxxx"> Arduino SAMD Boards (32-bits ARM Cortex-M0+) Boards > Arduino MKR WiFi 1010*.
- e. Connect the MKR WiFi 1010 board to the workstation via a USB Micro B cable.
- f. In Arduino IDE, on the menu bar, click on *Tools > Port: "COMx (xxx)"*.
- g. Select the serial port connected to the MKR WiFi 1010 board (If not sure with the COM port using, open *Device Manager* on the workstation, navigate to *Ports (COM & LPT)* in the list, and check for the usage of COM ports).

### 3.1.2 Test Board

To test the functionality and correct connection of the MKR WiFi 1010 board, a simple program will be implemented to blink the onboard LED. Steps are as following:

- a. In Arduino IDE, on the menu bar, click on *Tools > Board > Board: "xxxx"> Arduino SAMD Boards (32-bits ARM Cortex-M0+) Boards > Arduino MKR WiFi 1010*.
- b. Connect the MKR WiFi 1010 board to the workstation via a USB Micro B cable.
- c. In Arduino IDE, on the menu bar, click on *File > Examples > 01.Basics > Blink*.
- d. In Arduino IDE, on the menu bar, click on *Tools > Port: "COMx (xxx)"*.
- e. Select the serial port connected to the MKR WiFi 1010 board.
- f. In Arduino IDE, click the *Upload* button to send the code to the board.
- g. After seeing the message "Done uploading", check the orange LED that near the 5V pin and the "2R2" chip label on the MKR WiFi 1010 board, it should be blinking.
- h. For further testing, change the code content from "delay(1000)" to "delay(100)" to see the LED blink in a higher frequency.

### 3.1.3 LED Control and Data Transmission via Blynk Mobile Phone Application

In this experiment, a small LED connected to the D5 pin of the MKR WiFi 1010 board will be remotely controlled using the MQTT protocol by operating on a mobile phone interface. Besides, a message will be sent from the board to the mobile phone interface and displayed on a virtual LCD screen with the assistant from the *Blynk* mobile application. Before performing this test, section 3.1.1 and section 3.1.2 should be gone through. Detailed steps for this test are as following, related code is also attached separately, namely "mkr1010wifiBlynk.ino". Steps are based on the project given in [4]:

- a. On the breadboard, install a small LED and connect the negative pin of the LED to the GND pin on the MKR WiFi 1010 board.
- b. Connect the positive pin of the LED to a 220ohm resister.
- c. Connect the other pin of the 220ohm resister to the D5 pin of the MKR WiFi 1010 board.
- d. Connect the MKR WiFi 1010 board to a workstation via a USB Micro B cable.
- e. In Arduino IDE, on the menu bar, click on *Tools > Port: "COMx (xxx)"*.
- f. Select the serial port connected to the MKR WiFi 1010 board.
- g. Connect a mobile phone to a Wi-Fi network with SSID (Wi-Fi name) and password required. (School Wi-Fi network that requires username may not work).
- h. Install the *Blynk* application on the mobile phone.
- i. Create an account for *Blynk* with a valid email address and log in to the application.
- j. Create a new project with device selected as "Arduino MKR1000", even Arduino MRK1010 is used, and select connection type as "WiFi".
- k. An authentication token will be sent to the registered email address, keep it for future use (If not received, please check spam).
- l. In Arduino IDE, open "mkr1010wifiBlynk.ino", which attached separately (Detailed code explanations could be found in comments and [4]).
- m. In Arduino IDE, on the menu bar, click on *Tools > Manage Libraries...*
- n. Search for "Blynk" library and install the latest version.
- o. Modify the content in "(Token received)" in "mkr1010wifiBlynk.ino" file to the token received in step k.
- p. Modify the content in "(Any 2.4G Hz WiFi SSID)" and the content in "(Password)" to the local Wi-Fi name and password that used for mobile phone connections in step g.
- q. Click the *Upload* button to send the code to the board.
- r. After seeing the message "Done uploading", in on the *Blynk* application, click the "Run" button (in triangle shape) at the top right corner of project page. If connection is established, no message will appear. If received "Project is offline", double check local Wi-Fi name and password (Please be advised that 5G Hz Wi-Fi is not working with this MKR WiFi 1010 board, switch to 2.4G Hz Wi-Fi if it is available).
- s. Click on the "Terminate" button (in square shape) at the top right corner of project page after running properly.

- t. Click the blank space below and add one button under “Controllers” group and one virtual LCD screen under “Displays” group.
- u. Click on the button created, select the output pin to be “Digital” and “D5 PWN” same as physically connected in step c. Modify two other value under “Output” section to be “0” on the left, and “1” on the right to set up the digital output value. Swipe to “Switch” under “Mode”. Click on *OK* to back to main project page.
- v. Click on the LCD, swipe to “Advanced” and select the input pin to be “Virtual” and “V1”. Click on *OK* to back to main project page.
- w. In Arduino IDE, modify the content in “`lcd.print(n, m, “xxx”)`” to send the desired data to an ideal place on the virtual LCD. Click the *Upload* button to send the code to the board.
- x. After seeing the message "Done uploading.", click the “Run” button at the top right corner of project page on the *Blynk* application. The user should be able to control the LED by pushing the virtual button on and off and see the desired content to be displayed on the virtual LCD. The sample outcome of this test is shown in Figure 1 below:



Figure 1 Result of LED control and data transmission test via MQTT on *Blynk* application

In conclusion, devices physically connected to the Arduino MKR WiFi 1010 could be remotely controlled on the *Blynk* mobile application using the MQTT protocol. By modifying the physically connected pin on the board, the pin selected in the *Blynk* app should be modified accordingly. String data transmitted from the board could also be received. Users could modify the data wish to transmit through the Arduino IDE. Furthermore, by utilizing the analog pin (A#) on the MKR WiFi 1010 board, analog data transmitted to the board, such as the data collected by the COVID-19 testing machine, could be also be sent to the *Blynk* mobile application. However, in this methodology, access to the Internet and usage on the third-party mobile application are required.

### 3.1.4 Secure Data Transmission to AWS IoT Core Cloud

In this experiment, the MKR WiFi 1010 board will be connected to the AWS IoT Core cloud using Wi-Fi and MQTT protocol to realize data message transmission. Thanks to the authentication required by AWS IoT Core and the ATECC508A CryptoAuthentication element installed on the MKR WiFi1010 board, a secured transmission could be established. To do so, a Certificate Signing Request (CSR) needs be generated on the board and then uploaded to the AWS console to create an X.509 certificate. Before performing this experiment, section 3.1.1 and 3.1.2 should be gone through. Detailed steps for this test are as following, related code is also attached separately, namely “mkr1010wifiAWSIoT.ino”. Steps are based on project given in [5]:

- a. Connect the MKR WiFi 1010 board to a workstation via a USB Micro B cable.
- b. In Arduino IDE, on the menu bar, click on *Tools > Port: “COMx (xxx)”*.
- c. Select the serial port connected to the MKR WiFi 1010 board.
- d. In Arduino IDE, on the menu bar, click on *Tools > Manage Libraries...*
- e. Search and install “WiFiNINA”, “ArduinoBearSSL”, “ArduinoECCX08”, “ArduinoMqttClient”, and “Arduino Cloud Provider Examples” libraries.
- f. In Arduino IDE, on the menu bar, click on *File > Examples > ArduinoECCX08 > Tools > ECCX08CSR* to import a sketch for generating the CSR (Please be advised that step f to step l is for creating CSR, which has already been done by the author and it is unnecessary to repeat these steps since the total number of CSR could be created by the board is limited to 5. It will be preferable to re-use the CSR and code “mkr1010wifiAWSIoT.ino” attached separately and jump to step m. Otherwise, follow the next step to create a new CSR).
- g. In Arduino IDE, on the menu bar, click on *Tools > Serial Monitor* to open the serial monitor.
- h. Check the line ending configuration of the serial monitor is set to "Both NL & CR" and the same baud rate as set in the program (where “Serial.begin(115200);” in the code indicates the baud rate).
- i. In Arduino IDE, click the *Upload* button to send the code to the board.
- j. After seeing the message "Done uploading", a message will be displayed on the serial monitor to ask for PERMANENTLY configuring and locking the ECCX08.
- k. Choose “y” and proceed. Leave all answers blank except for the "Common Name", enter a desired name for the board. Choose one slot from slots 1 to 4 to generate and store the private



key as slot 0 has already been used. Finally, chose “y” for confirm, the key will then be displayed.

- l. Copy the generated CSR text including "-----BEGIN CERTIFICATE REQUEST-----" and "---END CERTIFICATE REQUEST-----" and save it to a new text file.
- m. Open a web browser and go to *aws.amazon.com*.
- n. Create an account or login.
- o. Search for "IoT Core" in “Find services”.
- p. On the menu bar, click on *Manage > Things > Register a thing > Create a single thing*.
- q. Name the thing and leave other space blank.
- r. Click on *Next > Create with CSR*.
- s. Select and upload the text file with CSR which was generated in step l or re-use the CSR attached.
- t. On the menu bar, click on *Secure > Policies*.
- u. Input "iot:\*" for the Action and "\*" for the Resource ARN. Check the "Allow" box and then click on *Create*.
- v. On the menu bar, click on *Certificates > Actions > Download* to download the certificate
- w. On the menu bar, click on *Actions > Attach Policy > Select* and select the policy created in step u and then Click on *Attach*.
- x. On the menu bar, click on *Settings* and get the MQTT endpoint and save it to a text file.
- y. In Arduino IDE, open “mkr1010wifiAWSIoT.ino” file attached separately.
- z. In the *arduino\_secrets.h* file, modify the content in “SSID” to the local Wi-Fi name; the content in “PASS” to the local Wi-Fi password; the content in “BROKER” to the MQTT endpoint saved in step x; the content in “SECRET\_CERTIFICATE[] = R"()"” to the certificate content downloaded in step v.
- aa. In Arduino IDE, on the menu bar, click on *Tools > Serial Monitor* to open the serial monitor and check the line ending configuration of the serial monitor is set to "Both NL & CR" and the same baud rate as set in the program (where “Serial.begin(115200);” in the code indicates the baud rate).
- bb. Click *Upload* to upload the sketch to the MRK WiFi 1010 board.

- cc. After seeing the message "Done uploading", a message will be displayed on the serial monitor, showing attempt to connect to the Wi-Fi network and if successful, it will try to connect to AWS IoT using MQTT.
- dd. On *aws.amazon.com*, on the menu bar, click on *Test* and go to the MQTT client page.
- ee. Enter *arduino/outgoing* into "Subscribe topic" click on *Subscribe to topic*. A welcome message will be sent from the board to AWS IoT Core and displayed on the screen.
- ff. Enter *arduino/incoming* in "Publish" and click on *Publish to topic* button. A welcome message will be sent from the AWS IoT Core to the MRK WiFi 1010 board and displayed on the serial monitor.
- gg. The contents of both messages can be modified on the AWS IoT Core webpage and Arduino IDE, respectively.
- hh. The secure MQTT protocol connection has been established between the AWS IoT Core cloud server and the MRK WiFi 1010 board. The sample outcome of this test is shown in the Figure 2 below:

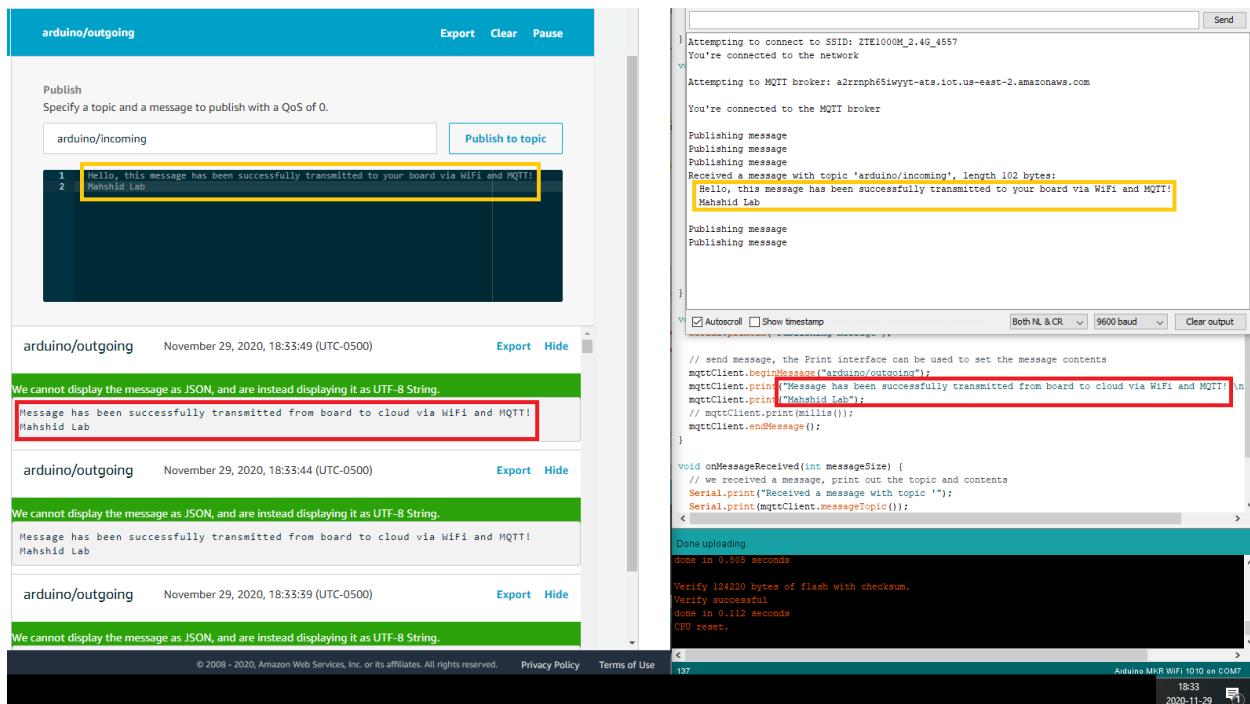


Figure 2 Result of secure data transmission via MQTT protocol and using AWS IoT Core. Yellow frame labels message sent from the AWS IoT Core to the Arduino IDE while red frame labels message sent from the Arduino IDE to the AWS IoT Core

In conclusion, a secure data transmission has been established between the Arduino MKR WiFi 1010 board and the AWS IoT Core using the MQTT protocol. String data transmission between the board and cloud could be achieved. User could modify the data for transmission through the Arduino IDE. Furthermore, by utilizing the analog pin (A#) on the MKR WiFi 1010 board, analog data transmitted to the board, such as the data collected by the COVID-19 testing machine, could also be sent to the AWS IoT Core webpage. However, in this methodology, access to the Internet and usage on the third-party cloud service are required.

### 3.1.5 Create Local Access Point

In this experiment, a local AP will be created on the MKR WiFi 1010 board using the U-BLOX NINA-W10 module. A small LED connected to the D3 pin of the MKR WiFi 1010 board will be remotely controlled by operating on a webpage interface. The webpage is created on the local IP address given by the board and done by writing code in the Arduino IDE. Data value read from an analog pin (A1) on the MKR WiFi 1010 board will be simulated and transmitted to display on the webpage. Before performing this experiment, section 3.1.1 and 3.1.2 should be gone through. Detailed steps for this test are as following, related code is also attached separately, namely “mkr1010wifiAP.ino”. Steps are based on project given in [6]:

- a. On the breadboard, install a small LED and connect the negative pin of the LED to the GND pin on the MKR WiFi 1010 board.
- b. Connect the positive pin of the LED to a 220ohm resistor.
- c. Connect the other pin of the 220ohm resistor to the D3 pin of the MKR WiFi 1010 board.
- d. Connect the MKR WiFi 1010 board to a workstation via a USB Micro B cable.
- e. In Arduino IDE, on the menu bar, click on *Tools > Port: “COMx (xxx)”*.
- f. Select the serial port connected to the MKR WiFi 1010 board.
- g. In Arduino IDE, open “mkr1010wifiAP.ino”, which attached separately (Detailed code explanations could be found in comments and [6]).
- h. In Arduino IDE, on the menu bar, click on *Tools > Manage Libraries...*
- i. Search for “WiFiNINA” library and install the latest version if necessary.
- j. Modify the content in “char ssid[] = “MahshidLabMKR1010WiFiTest”” and the content in “char pass[] = “MKR1010test1234”” to another desired Wi-Fi name and password if necessary, otherwise maintain the same name and password.
- k. In Arduino IDE, on the menu bar, click on *Tools > Serial Monitor* to open the serial monitor.

- l. Check the line ending configuration of the serial monitor is set to "Both NL & CR" and the same baud rate as set in the program (where "Serial.begin(9600);" in the code indicates the baud rate).
- m. Click the *Upload* button to send the code to the board.
- n. After seeing the message "Done uploading", a series of messages will be displayed on the serial monitor.
- o. If received a message "Please upgrade the firmware" on the serial monitor, check [7] for further information on updating the firmware of "WiFiNINA" library.
- p. Use a mobile device, either a cellphone, a pad, or a laptop, to search for a Wi-Fi network with SSID (Wi-Fi name) set in step j.
- q. Connect the mobile device to the Wi-Fi network with password set in step j.
- r. Follow the information given on the serial monitor to open the web browser on the mobile device and go to the http address given.
- s. The user should be able to control the LED by clicking on the blue words *here* on the webpage to turn it on and off. A random value read from the analog pin A1 will also be printed below. The information of the local Wi-Fi network created by the Arduino MKR WiFi 1010 board and output on the serial monitor is shown in Figure 3, while the sample outcome of this test on the webpage is shown in Figure 4 below:

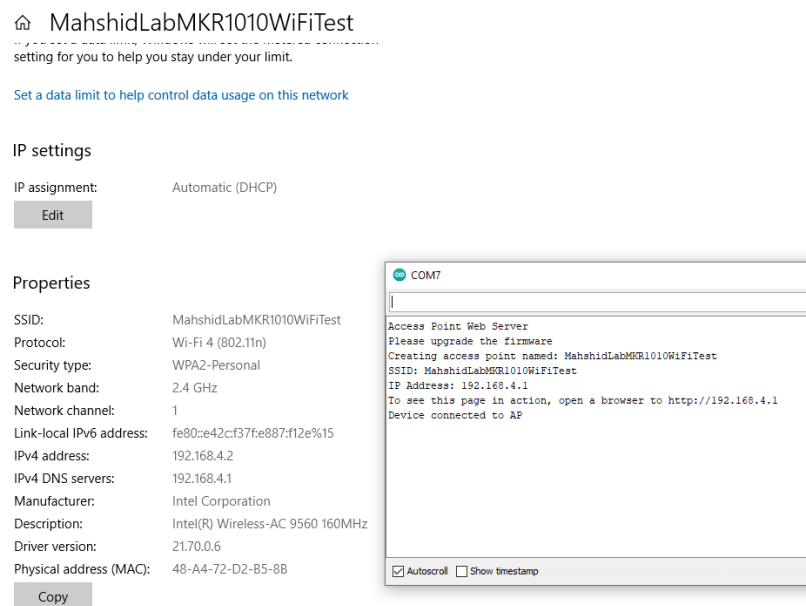


Figure 3 Information of the local Wi-Fi network created by the Arduino MKR WiFi 1010 board and the output displayed on the serial monitor

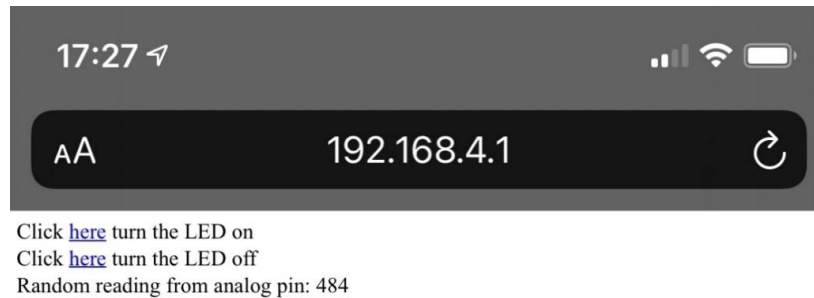


Figure 4 Webpage result on the IP address given by the Arduino MKR WiFi 1010 board

In conclusion, a local AP has been created on the Arduino MKR WiFi 1010. Small devices physically connected to the Arduino MKR WiFi 1010 board could be remotely controlled by mobile devices connected to the local AP. Analog data read from the board could also be transmitted to and received on mobile devices. The webpage created on the local host IP address could be modified and customized by directly programming in the Arduino IDE. Furthermore, as the program written to the board is stored in the Flash memory, it is not necessary to upload the code each time using. Once the program has been uploaded to the board, the local AP will be functional as long as essential power supply to the board is provided. Functionalities of other pins on the Arduino MKR WiFi 1010 board could be further explored.

### 3.1.6 LED Control and Data Transmission using Bluetooth Low Energy

In this experiment, a BLE function on the MKR WiFi 1010 board provided by the U-BLOX NINA-W10 module will be tested. A small LED connected to the D2 pin of the MKR WiFi 1010 board will be remotely controlled using BLE by operating on a mobile device application, namely *LightBlue*. Data value read from an analog pin (A1) on the MKR WiFi 1010 board will be simulated and transmitted to display on *LightBlue*. Before performing this experiment, section 3.1.1 and 3.1.2 should be gone through. Detailed steps for this test are as following, related code is also attached separately, namely “mkr1010wifible.ino”. Steps are based on project given in [8]:

- a. Download *LightBlue* application on a mobile device.
- b. On the breadboard, install a small LED and connect the negative pin of the LED to the GND pin on the MKR WiFi 1010 board.
- c. Connect the positive pin of the LED to a 220ohm resister.
- d. Connect the other pin of the 220ohm resister to the D2 pin of the MKR WiFi 1010 board.
- e. Connect the MKR WiFi 1010 board to a workstation via a USB Micro B cable.
- f. In Arduino IDE, on the menu bar, click on *Tools > Port: “COMx (xxx)”*.

- g. Select the serial port connected to the MKR WiFi 1010 board.
- h. In Arduino IDE, open “mkr1010wifible.ino”, which attached separately (Detailed code explanations could be found in comments and [8]).
- i. In Arduino IDE, on the menu bar, click on *Tools > Manage Libraries...*
- j. Search for “ArduinoBLE” library and install the latest version.
- k. Modify the content in “BLE.setLocalName(“MahshidLabBLETest”)” to another desired Bluetooth service if necessary, otherwise maintain the same name.
- l. In Arduino IDE, on the menu bar, click on *Tools > Serial Monitor* to open the serial monitor.
- m. Check the line ending configuration of the serial monitor is set to “Both NL & CR” and the same baud rate as set in the program (where “Serial.begin(9600);” in the code indicates the baud rate).
- n. Click the *Upload* button to send the code to the board.
- o. After seeing the message “Done uploading”, a series of messages will be displayed on the serial monitor.
- p. When received a message “Bluetooth device active, waiting for connections” on the serial monitor, turn on the Bluetooth function on the mobile device.
- q. Use the *LightBlue* application to search for a Bluetooth device named “Arduino” and Local Name set in step k.
- r. Connect the mobile device to the Arduino MKR WiFi 1010 board using Bluetooth and click on the *Arduino* device.
- s. Under “ADVERTISEMENT DATA”, the Local Name of the device could be confirmed.
- t. Under “Device Information”, *Digital Output* and *Analog* options will be displayed (This has been tested to be in Hexadecimal format on IOS devices).
- u. Click on the *Digital Output* option, user could control the LED connected to the D2 pin by clicking on *Write new value* to write value “1” to turn it on and value “0” to turn it off.
- v. Click on the *Read again* button to refresh the data reading from D2 pin.
- w. The status of the LED will be displayed on the serial monitor.
- x. Back to the “Device Information” page and click on the *Analog* option, the analog value read by A1 pin on the Arduino MKR WiFi 1010 board will be displayed. User could Click on the *Read again* button to refresh the data reading from the A1 pin.

- y. The output displayed on the serial monitor is shown in Figure 5 while the sample outcome of this test on the *LightBlue* application is shown in Figure 6 below:

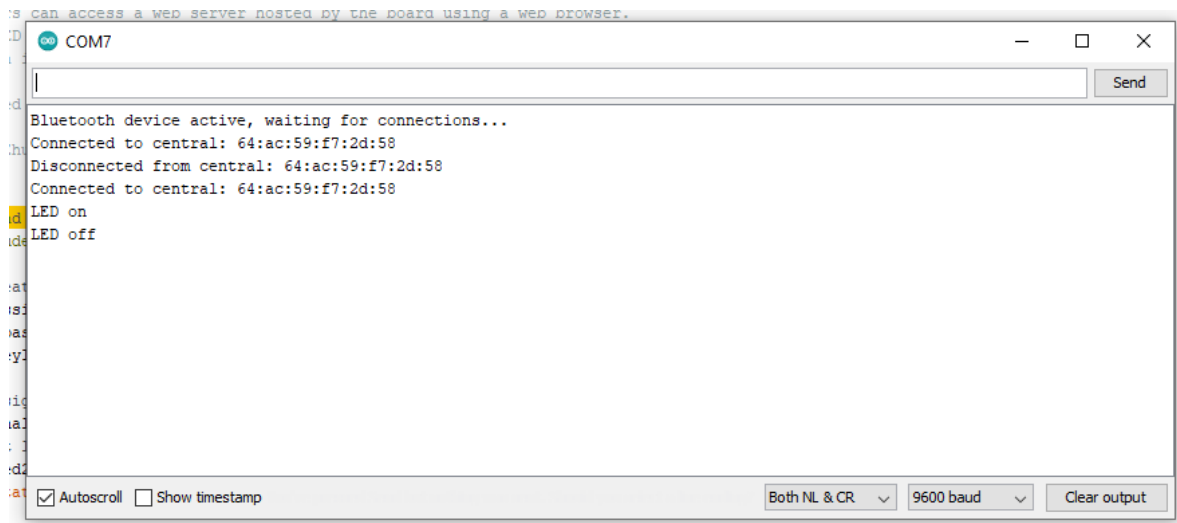


Figure 5 Output displayed on the serial monitor

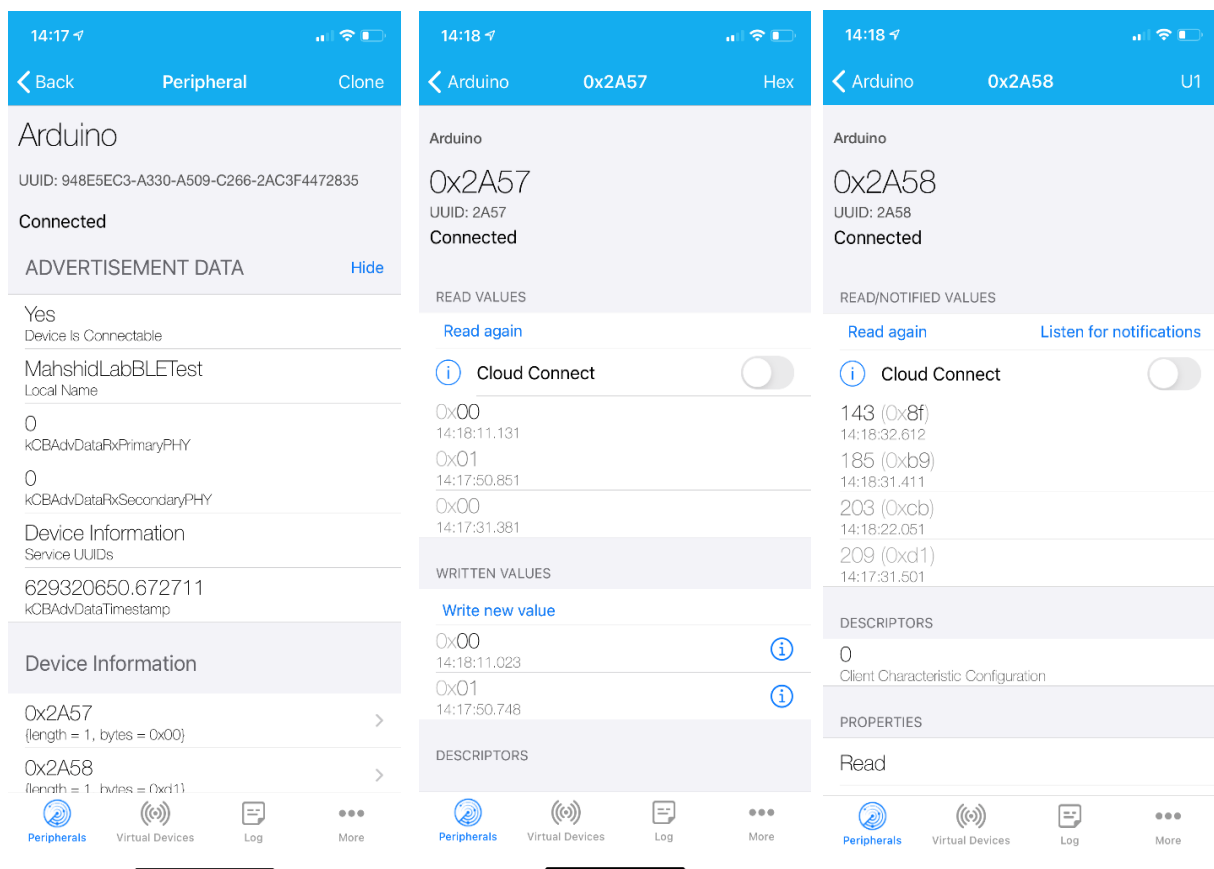


Figure 6 Result of LED control and analog data transmission test via Bluetooth on *LightBlue* application

In conclusion, the Arduino MKR WiFi 1010 provides an extra BLE communication solution. Small devices physically connected to the Arduino MKR WiFi 1010 board could be remotely controlled by mobile devices using Bluetooth technology. Analog data read from the board could also be transmitted to and received on mobile devices. However, a third-party mobile application is required. This experiment only provides a basic test on the application of BLE technology in data transmission, further research could be done. Useful information and research on BLE technology may be found in [9, 10].

### 3.2 Espressif ESP32-S2 Wi-Fi MCU

ESP32-S2 is a single-core Wi-Fi System on a Chip that supports 20MHz, 40MHz bandwidth Wi-Fi data transmission in the 2.4GHz band. It consists of an Xtensa single-core 32-bit LX7 microprocessor and 43 programmable general-purpose input/output (GPIO) pins. A USB Micro-B port is provided onboard for serial communication with a workstation. More detailed information on the Espressif ESP32-S2 chip could be found in [11]. Some development insights regarding general ESP32 chips could also be found in [12]. It is worth mention that the tested ESP32-S2 board does not support Bluetooth connectivity. If BLE data transmission experiments on the ESP32 board are required in the future, ESP32-C3 Series DevKits would be recommended.

#### 3.2.1 IDE Configuration

First of all, the configuration of the Espressif ESP32-S2 board needs to be done for further programming in the Arduino IDE. Steps are as following:

- a. In Arduino IDE, on the menu bar, click on *File > Preferences*.
- b. In the area after “Additional Board Manager URLs”, enter:  
“[https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)” and click on *OK*.
- c. In Arduino IDE, on the menu bar, click on *Tools* menu and then select *Board: "xxx" > Boards Manager...*
- d. Search for *esp32*.
- e. Select the latest version and then click *Install*.
- f. In Arduino IDE, on the menu bar, click on *File > Preferences*.
- g. Click on the path under “More preference can be edited directly in the file”.
- h. Go to *packages > esp32 > hardware > esp32 > 1.0.4*.
- i. Close the Arduino IDE.



- j. Delete all files under folder *1.0.4*.
- k. Open a web browser and go to <https://github.com/espressif/arduino-esp32/tree/esp32s2>.
- l. Switch branches/tags to *esp32s2*.
- m. Click on *Code > Download ZIP*.
- n. Copy all files in the Zip file downloaded to the folder *1.0.4* stated in step j.
- o. In folder *1.0.4* Go to *tools* and click on *get.exe* (This step has been tested on Windows OS only, if not working on Mac OS or Linux, try clicking on *get.py*).
- p. In Arduino IDE, on the menu bar, click on *Tools > Board > Board: "xxxx"> ESP32 Arduino > ESP32S2 Dev Module*.
- q. Connect the ESP32-S2 board to the workstation via USB Micro B cable.
- r. In Arduino IDE, on the menu bar, click on *Tools > Port: "COMx (xxx)"*.
- s. Select the serial port connected to the ESP32-S2 board (If not sure with the COM port using, open *Device Manager* on the workstation, navigate to *Ports (COM & LPT)* in the list, and check for the usage of COM ports).

### 3.2.2 Test Board

To test the functionality and correct connection of the ESP32-S2 board, a Wi-Fi scan program will be implemented to search for all local Wi-Fi networks. Steps are as following, based on project given in [13]:

- a. In Arduino IDE, on the menu bar, click on *Tools > Board > Board: "xxxx"> ESP32 Arduino > ESP32S2 Dev Module*.
- b. Connect the ESP32-S2 board to the workstation via a USB Micro B cable.
- c. In Arduino IDE, on the menu bar, click on *Tools > Port: "COMx (xxx)"*.
- d. Select the serial port connected to the ESP32-S2 board.
- e. In Arduino IDE, on the menu bar, click on *File > Examples > WiFi > WiFiScan*.
- f. In Arduino IDE, on the menu bar, click on *Tools > Serial Monitor* to open the serial monitor.
- g. Check the line ending configuration of the serial monitor is set to "Both NL & CR" and the same baud rate as set in the program (where "Serial.begin(115200);" in the code indicates the baud rate).
- h. In Arduino IDE, click the *Upload* button to send the code to the board.

- i. After seeing the message "Done uploading", a series messages including “scan start”, “scan done”, and “networks found” with a list of local Wi-Fi network names should be displayed on the serial monitor.

### 3.2.3 Create Local Access Point

In this experiment, a local AP will be created on the ESP32-S2 board. A small LED connected to the D3 pin of the ESP32-S2 board will be remotely controlled by operating on a webpage interface. The webpage is created on the local IP address given by the board and done by writing code in the Arduino IDE. Data value read from an analog pin (A1) on the ESP32-S2 board will be simulated and transmitted to display on the webpage. Before performing this experiment, section 3.2.1 and 3.2.2 should be gone through. Detailed steps for this test are as following, related code is also attached separately, namely “esp32wifiAP.ino”. Steps are based on project given in [14]:

- a. On the breadboard, install a small LED and connect the negative pin of the LED to the GND pin on the ESP32-S2 board.
- b. Connect the positive pin of the LED to a 220ohm resister.
- c. Connect the other pin of the 220ohm resister to the D3 pin of the ESP32-S2 board.
- d. Connect the ESP32-S2 board to a workstation via a USB Micro B cable.
- e. In Arduino IDE, on the menu bar, click on *Tools > Port: “COMx (xxx)”*.
- f. Select the serial port connected to the ESP32-S2 board.
- g. In Arduino IDE, open “esp32wifiAP.ino”, which attached separately (Detailed code explanations could be found in comments and [14]).
- h. Modify the content in “const char\* ssid[] = "MahshidLabESP32WiFiTest"” and the content in “const char\* pass[] = "ESP32test1234"” to another desired Wi-Fi name and password if necessary, otherwise maintain the same name and password.
- i. In Arduino IDE, on the menu bar, click on *Tools > Serial Monitor* to open the serial monitor.
- j. Check the line ending configuration of the serial monitor is set to "Both NL & CR" and the same baud rate as set in the program (where “Serial.begin(115200);” in the code indicates the baud rate).
- k. Click the *Upload* button to send the code to the board.
- l. After seeing the message "Done uploading", a series of messages will be displayed on the serial monitor.

- m. Use a mobile device, either a cellphone, a pad, or a laptop, to search for a Wi-Fi network with SSID (Wi-Fi name) set in step h.
- n. Connect the mobile device to the Wi-Fi network with password set in step h.
- o. Follow the information given on the serial monitor to open the web browser on the mobile device and go to the http address given.
- p. The user should be able to control the LED by clicking on the button on the ESP32 Web Server webpage to turn it on and off. A random value read from the analog pin A1 will also be printed below. The connection information of the local Wi-Fi network created by the ESP32-S2 is shown in Figure 7, while the sample web page outcome of this test is shown in Figure 8. Output message displayed on the serial monitor is shown in Figure 9 below:

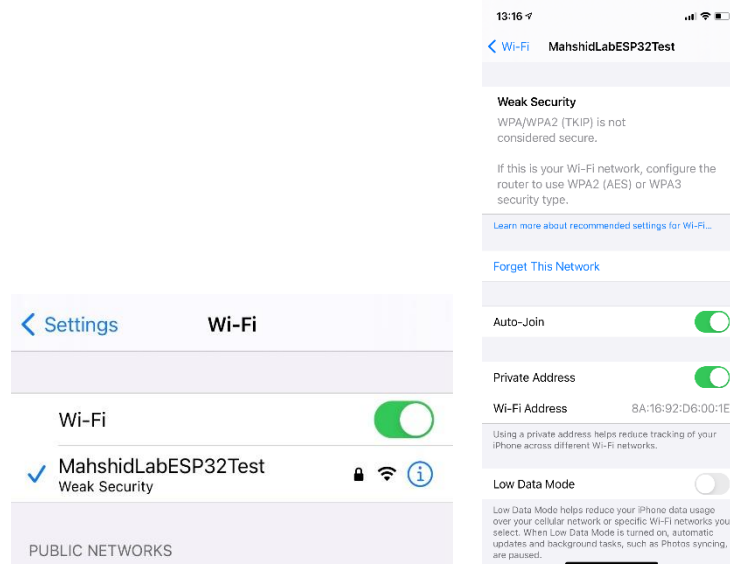


Figure 7 Connection of the Wi-Fi network created by the ESP32-S2 board on an IOS device

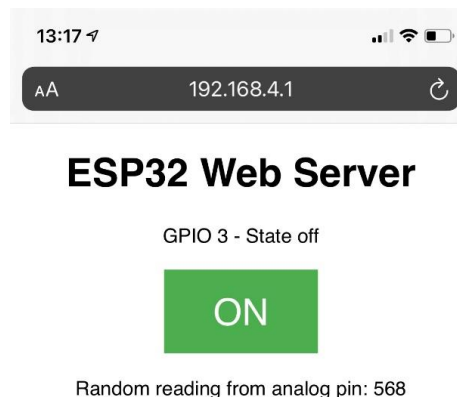
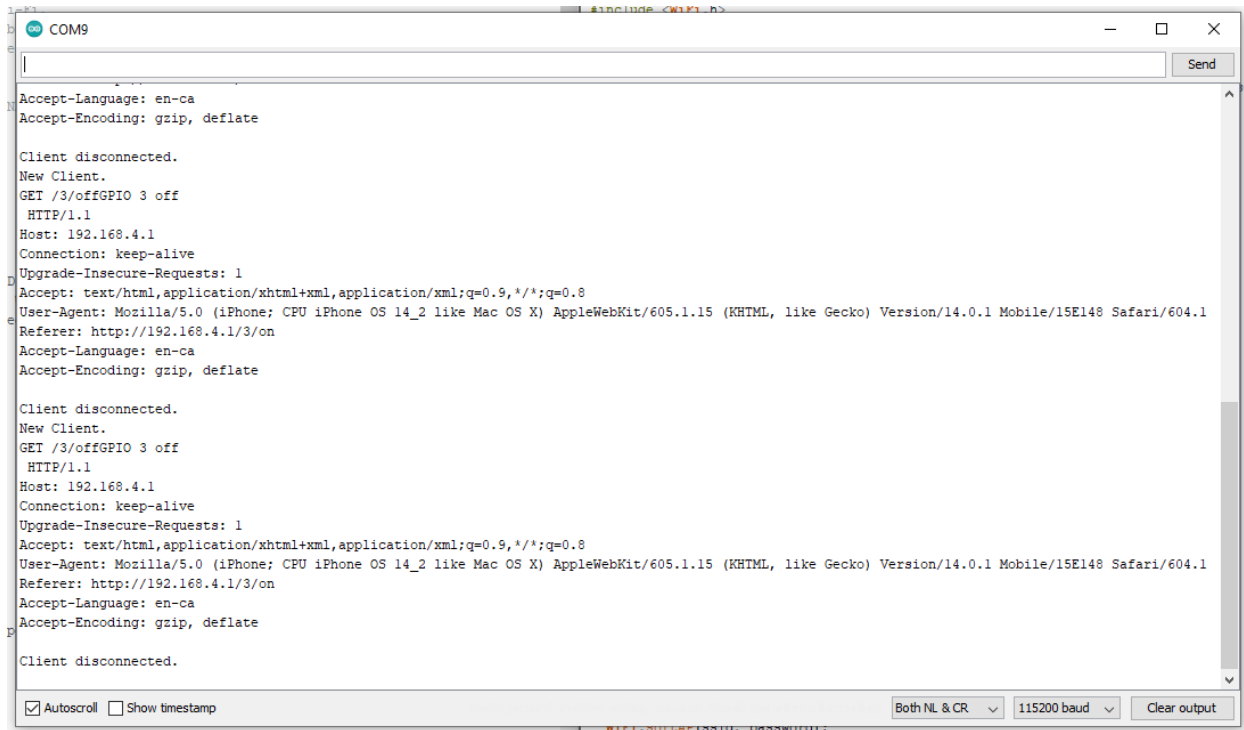


Figure 8 Webpage result on the IP address given by the ESP32-S2 board



```
COM9
Accept-Language: en-ca
Accept-Encoding: gzip, deflate

Client disconnected.
New Client.
GET /3/offGPIO 3 off
HTTP/1.1
Host: 192.168.4.1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 14_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0.1 Mobile/15E148 Safari/604.1
Referer: http://192.168.4.1/3/on
Accept-Language: en-ca
Accept-Encoding: gzip, deflate

Client disconnected.
New Client.
GET /3/offGPIO 3 off
HTTP/1.1
Host: 192.168.4.1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 14_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0.1 Mobile/15E148 Safari/604.1
Referer: http://192.168.4.1/3/on
Accept-Language: en-ca
Accept-Encoding: gzip, deflate

Client disconnected.

☒ Autoscroll ☐ Show timestamp
Both NL & CR 115200 baud Clear output
```

Figure 9 Information of the local Wi-Fi network created by the ESP32-S2 board displayed on the serial monitor

In conclusion, similar to the test done on the Arduino MKR WiFi 1010 board, a local AP has been created on the Espressif ESP32-S2 board. Small devices physically connected to the ESP32-S2 board could be remotely controlled by mobile devices connected to the local AP. Analog data read from the board could also be transmitted to and received on mobile devices. The webpage created on the local host IP address could be modified and customized by directly programming in the Arduino IDE. Furthermore, it is also not necessary to upload the code each time using. Once the program has been uploaded to the board, the local AP will be functional as long as the essential power supply to the board is provided. As the ESP32-S2 board contains 43 programmable GPIOs, functionalities of other pins on the board could be further explored.

### 3.3 SparkFun ESP8266 Thing Dev Board

The SparkFun ESP8266 Thing Dev Board is a simple development board that integrated with a low power 32-bit CPU and provides Wi-Fi connectivity. A USB Micro-B port is installed onboard for serial communication with a workstation. Only 11 digital pins and 1 analog pin on the board are programmable using the Arduino IDE. Further information regarding the SparkFun ESP8266 board could be found at [15].

### 3.3.1 IDE Configuration

The configuration of SparkFun ESP8266 board in the Arduino IDE is similar to Espressif ESP32-S2. Steps are as following:

- a. In Arduino IDE, on the menu bar, click on *File > Preferences*.
- b. In the area after “Additional Board Manager URLs”, enter:  
“[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)” and click on *OK*.
- c. In Arduino IDE, on the menu bar, click on *Tools* menu and then select *Board: "xxx" > Boards Manager...*
- d. Search for *esp8266*.
- e. Select the latest version and then click *Install*.
- f. In Arduino IDE, on the menu bar, click on *Tools > Board > Board: "xxxx" > ESP8266 Boards (2.7.4) > SparkFun ESP8266 Thing Dev*.
- g. Connect the ESP8266 board to the workstation via a USB Micro B cable.
- h. In Arduino IDE, on the menu bar, click on *Tools > Port: "COMx (xxx)"*.
- i. Select the serial port connected to the ESP8266 board (If not sure with the COM port using, open *Device Manager* on the workstation, navigate to *Ports (COM & LPT)* in the list, and check for the usage of COM ports).

### 3.3.2 Test Board

To test the functionality and correct connection of the ESP8266 board, a simple program will be implemented to blink the onboard LED. Steps are as following:

- a. In Arduino IDE, on the menu bar, click on *Tools > Board > Board: "xxxx" > ESP8266 Boards (2.7.4) > SparkFun ESP8266 Thing Dev*.
- b. Connect the ESP8266 board to the workstation via a USB Micro B cable.
- c. On the ESP8266 board, turn the switch at the corner from “OFF” to “ON” to wake up the board.
- d. In Arduino IDE, on the menu bar, click on *Tools > Port: "COMx (xxx)"*.
- e. Select the serial port connected to the ESP8266 board.
- f. In Arduino IDE, on the menu bar, click on *File > Examples > 01.Basics > Blink*.
- g. Write “`#define LED 5`” above the “`void setup()`” to define the onboard LED pin (above the comment line as well).
- h. Change all “`LED_BUILTIN`” in the code to “`LED`”.
- i. In Arduino IDE, click the *Upload* button to send the code to the board.

- j. After seeing the message "Done uploading", check the blue LED on the ESP8266 board, it should be blinking.
- k. For further testing, change the code content from “delay(1000)” to “delay(100)” to see the LED blink in a higher frequency.

### 3.3.3 Create Local Access Point

In this experiment, a local AP will be created on the ESP8266 board. The onboard LED (connected to D5 pin) on the ESP8266 board will be remotely controlled by operating on a webpage interface. The webpage is created on the local IP address given by the board and done by writing code in the Arduino IDE. Data value read from the only analog pin (A0) and a digital pin (D12) on the ESP8266 board will be transmitted to display on the webpage. Before performing this experiment, section 3.3.1 and 3.3.2 should be gone through. Detailed steps for this test are as following, related code is also attached separately, namely “esp8266wifiAP.ino”. Steps are based on project given in [16]:

- a. Connect the ESP8266 board to a workstation via a USB Micro B cable.
- b. On the ESP8266 board, turn the switch at the corner from “OFF” to “ON” to wake up the board.
- c. In Arduino IDE, on the menu bar, click on *Tools > Port: “COMx (xxx)”*.
- d. Select the serial port connected to the ESP8266 board.
- e. In Arduino IDE, open “esp8266wifiAP.ino”, which attached separately (Detailed code explanations could be found in comments and [16]).
- f. Modify the content in “const char ssid[] = "MahshidLabESP8266WiFiTest"” and the content in “const char\* pass[] = "ESP8266test1234"” to another desired Wi-Fi name and password if necessary, otherwise maintain the same name and password.
- g. In Arduino IDE, on the menu bar, click on *Tools > Serial Monitor* to open the serial monitor.
- h. Check the line ending configuration of the serial monitor is set to "Both NL & CR" and the same baud rate as set in the program (where “Serial.begin(115200);” in the code indicates the baud rate).
- i. Click the *Upload* button to send the code to the board.
- j. After seeing the message "Done uploading", a series of messages will be displayed on the serial monitor.
- k. Use a mobile device, either a cellphone, a pad, or a laptop, to search for a Wi-Fi network with SSID (Wi-Fi name) set in step f.

- l. Connect the mobile device to the Wi-Fi network with password set in step f.
- m. Open the web browser on the mobile device and go to the local IP address created by the ESP8266 board (192.168.4.1).
- n. Follow the instruction on the webpage to control the onboard LED (input “192.168.4.1/led/0” in browse address bar to turn it off and input “192.168.4.1/led/1” to turn it on) and check the value read by A0 analog pin and D12 digital pin (input “192.168.4.1/read” to go to related page). The connection information of the local Wi-Fi network created by the ESP8266 board is shown in Figure 9, while the sample web page outcome of this test is shown in Figure 10.

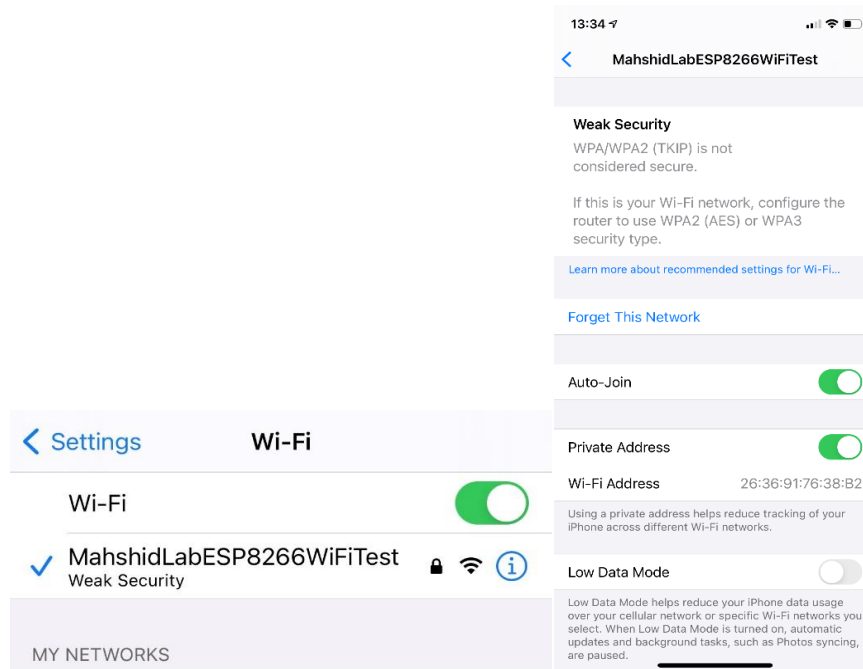


Figure 9 Connection of the Wi-Fi network created by the ESP8266 board on an IOS device

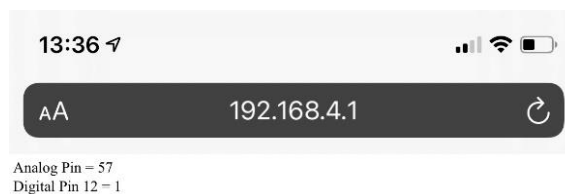


Figure 10 Webpage result on the IP address given by the ESP8266 board

In conclusion, a local AP has been created on the SparkFun ESP8266 Thing Dev board. The onboard LED could be remotely controlled by mobile devices connected to the local AP. Analog

and digital data read from the board could also be transmitted to mobile devices. The webpage created on the local host IP address could also be modified and customized by directly programming in the Arduino IDE. Furthermore, as the program has been uploaded to the board, the local AP will be functional while essential power supply is provided. However, since the ESP8266 board contains only 11 programable digital GPIOs and 1 programable analog pin, fewer functionalities on this board could be utilized.

## **4. Other Transmission Protocols**

Besides Wi-Fi and Bluetooth technologies, there are various selections of short-distance data transmission methodologies. Z-Wave and ZigBee are famous wireless sensor networks widely adopted in the Internet of Things (IoT) projects. Useful publications on these two topics could be found in [17-19]. A comparison study between different wireless networks is also given in [20].

## **5. Conclusion**

This test log provides general insight on wireless data transmission experiments using Arduino MKR WiFi 1010, Espressif ESP32-S2 Wi-Fi MCU, and SparkFun ESP8266 Thing Dev Board. Overall, all three boards support local AP creation and Wi-Fi data transmission. The Arduino MKR WiFi 1010 provides an extra BLE connectivity. Besides, the presence of the Arduino microcontroller is not necessary as all three development boards could perform data processing independently. The connection stability has not been specifically examined, while no remarkable connection issues were detected during experiments. In terms of cost, the Arduino MKR WiFi 1010 is the most expensive board but provides more functionalities. SparkFun ESP8266 board is more costly than the Espressif ESP32-S2 but includes fewer GPIOs. Consequently, the Espressif ESP32-S2 will be ideal for pure Wi-Fi based data transmission.

As for future work, parallel control variable experiments could be performed using the 4-Port Portable USB Hub purchased. Further research on packet transmissions, such as data array or list transmission using these three boards could be done. For data collection, software engineering techniques on webpage design may be required to create a better website on the local host IP address for result visualization and drag data received on the local AP for further interpolation. Integration tests using both the Wi-Fi development boards and electrochemical sensor devices could also be conducted. Data transmission solutions using Bluetooth, Z-Wave, and ZigBee could also be reviewed and examined.



## References

- [1] D. Eridani, K. T. Martono and A. A. Hanifah, "MQTT Performance as a Message Protocol in an IoT based Chili Crops Greenhouse Prototyping," *2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Yogyakarta, Indonesia, 2019, pp. 184-189.
- [2] N. Kajikawa, Y. Minami, E. Kohno and Y. Kakuda, "On Availability and Energy Consumption of the Fast Connection Establishment Method by Using Bluetooth Classic and Bluetooth Low Energy," *2016 Fourth International Symposium on Computing and Networking (CANDAR)*, Hiroshima, 2016, pp. 286-290.
- [3] Arduino. "Getting started with the MKR WiFi 1010."  
<https://www.arduino.cc/en/Guide/MKRWiFi1010> (accessed Nov 18, 2020).
- [4] Arduino\_Genuino. "Control Your IoT Cloud Kit via Blynk."  
<https://create.arduino.cc/projecthub/officine-innesto/control-your-iot-cloud-kit-via-blynk-ec6a16> (accessed Nov 20, 2020).
- [5] Arduino\_Genuino. "Securely Connecting an Arduino MKR WiFi 1010 to AWS IoT Core."  
[https://create.arduino.cc/projecthub/Arduino\\_Genuino/securely-connecting-an-arduino-mkr-wifi-1010-to-aws-iot-core-a9f365](https://create.arduino.cc/projecthub/Arduino_Genuino/securely-connecting-an-arduino-mkr-wifi-1010-to-aws-iot-core-a9f365) (accessed Nov 22, 2020).
- [6] K. Söderby. "Web Server using Access Point (AP) mode with MKR WiFi 1010."  
<https://www.arduino.cc/en/Guide/MKRWiFi1010/web-server-ap-mode> (accessed Dec 7, 2020).
- [7] Arduino. "Wi-Fi NINA Firmware Updater."  
<https://www.arduino.cc/en/Tutorial/WiFiNINA-FirmwareUpdater> (accessed Dec 8, 2020).
- [8] K. Söderby. "MKR WiFi 1010 Bluetooth Low Energy."  
<https://www.arduino.cc/en/Guide/MKRWiFi1010/enabling-ble> (accessed Dec 8, 2020).
- [9] Z. Feng, L. Mo and M. Li, "Analysis of low energy consumption wireless sensor with BLE," *2015 IEEE SENSORS*, Busan, 2015, pp. 1-4.
- [10] Z. Lin, C. Chang, N. Chou and Y. Lin, "Bluetooth Low Energy (BLE) based blood pressure monitoring system," *2014 International Conference on Intelligent Green Building and Smart Grid (IGBSG)*, Taipei, 2014, pp. 1-4.
- [11] L. ESPRESSIF SYSTEMS (SHANGHAI) CO. "ESP32-S2 A Secure and Powerful Wi-Fi MCU with Numerous I/O Capabilities."  
<https://www.espressif.com/en/products/socs/esp32-s2> (accessed Nov 24, 2020).
- [12] LastMinuteEngineers.com. "Insight Into ESP32 Features & Using It With Arduino IDE."  
<https://lastminuteengineers.com/esp32-arduino-ide-tutorial/> (accessed Nov 24, 2020).

- [13] R. Santos. "Installing the ESP32 Board in Arduino IDE (Windows, Mac OS X, Linux)."  
<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/> (accessed Dec 9, 2020).
- [14] R. Santos. "How to Set an ESP32 Access Point (AP) for Web Server."  
<https://randomnerdtutorials.com/esp32-access-point-ap-web-server/> (accessed Dec 9, 2020).
- [15] S. Electronics. "SparkFun ESP8266 Thing - Dev Board."  
<https://www.sparkfun.com/products/13711> (accessed Nov 26, 2020).
- [16] jimblom. "ESP8266 Thing Development Board Hookup Guide."  
<https://learn.sparkfun.com/tutorials/esp8266-thing-development-board-hookup-guide/example-sketch-web-server>  
(accessed Nov 26, 2020).
- [17] C. Yuan, H. Wang and J. He, "Remote Monitoring System Based on MC9S12NE64 and Z-WAVE Technology," *2010 International Conference on E-Product E-Service and E-Entertainment*, Henan, 2010, pp. 1-4.
- [18] P. m. Linh An and T. Kim, "A Study of the Z-Wave Protocol: Implementing Your Own Smart Home Gateway," *2018 3rd International Conference on Computer and Communication Systems (ICCCS)*, Nagoya, 2018, pp. 411-415.
- [19] T. Elarabi, V. Deep and C. K. Rai, "Design and simulation of state-of-art ZigBee transmitter for IoT wireless devices," *2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, Abu Dhabi, 2015, pp. 297-300.
- [20] M. Zareei, A. Zarei, R. Budiarto and M. A. Omar, "A comparative study of short range wireless sensor network on high density networks," *The 17th Asia Pacific Conference on Communications*, Sabah, 2011, pp. 247-252.