**********Draft**********

Arduino Nano RP2040 Connect Test

10/05/22

**********Draft**********

adding line 47 #define INT_1 INT_IMU
commenting 62 //#define INT_1 A5
https://docs.arduino.cc/tutorials/nano-rp2040-connect/rp2040-web-server-rgb#programming-the-board

```
/**
 ******************************************************************************
 * @file    X_NUCLEO_IKS01A3_LSM6DSOX_MLC.ino
 * @author  SRA
 * @version V1.1.0
 * @date    March 2020
 * @brief   Arduino test application for the STMicrolectronics X-NUCLEO-IKS01A3
 *          MEMS Inertial and Environmental sensor expansion board.
 *          This application makes use of C++ classes obtained from the C
 *          components' drivers.
 ******************************************************************************
 * @attention
 *
 * <h2><center>&copy; COPYRIGHT(c) 2020 STMicroelectronics</center></h2>
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *   1. Redistributions of source code must retain the above copyright notice,
 *      this list of conditions and the following disclaimer.
 *   2. Redistributions in binary form must reproduce the above copyright notice,
 *      this list of conditions and the following disclaimer in the documentation
 *      and/or other materials provided with the distribution.
 *   3. Neither the name of STMicroelectronics nor the names of its contributors
 *      may be used to endorse or promote products derived from this software
 *      without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
```

```
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ******************************************************************************
 */
//NOTE: This example isn't compatible with Arduino Uno.
//NOTE: For this example you need the STEVAL-MKI197V1 board connected to the DIL24 connector of the X-NUCLEO-IKS01A3

// Includes
#include "LSM6DSOXSensor.h"
#include "lsm6dsox_activity_recognition_for_mobile.h"

#define INT_1 INT_IMU



#ifdef ARDUINO_SAM_DUE
#define DEV_I2C Wire1
#elif defined(ARDUINO_ARCH_STM32)
#define DEV_I2C Wire
#elif defined(ARDUINO_ARCH_AVR)
#define DEV_I2C Wire
#else
#define DEV_I2C Wire
#endif
#define SerialPort Serial

//#define INT_1 A5

//Interrupts.
volatile int mems_event = 0;

// Components
LSM6DSOXSensor AccGyr(&DEV_I2C, LSM6DSOX_I2C_ADD_L);

// MLC
ucf_line_t *ProgramPointer;
int32_t LineCounter;
int32_t TotalNumberOfLine;

void INT1Event_cb();
void printMLCStatus(uint8_t status);

void setup() {
  uint8_t mlc_out[8];
  // Led.
  pinMode(LED_BUILTIN, OUTPUT);

  // Force INT1 of LSM6DSOX low in order to enable I2C
  pinMode(INT_1, OUTPUT);
```

```
  digitalWrite(INT_1, LOW);

  delay(200);

  // Initialize serial for output.
  SerialPort.begin(115200);

  // Initialize I2C bus.
  DEV_I2C.begin();

  AccGyr.begin();
  AccGyr.Enable_X();
  AccGyr.Enable_G();

  /* Feed the program to Machine Learning Core */
  /* Activity Recognition Default program */
  ProgramPointer = (ucf_line_t *)lsm6dsox_activity_recognition_for_mobile;
  TotalNumberOfLine = sizeof(lsm6dsox_activity_recognition_for_mobile) / sizeof(ucf_line_t);
  SerialPort.println("Activity Recognition for LSM6DSOX MLC");
  SerialPort.print("UCF Number Line=");
  SerialPort.println(TotalNumberOfLine);

  for (LineCounter=0; LineCounter<TotalNumberOfLine; LineCounter++) {
    if(AccGyr.Write_Reg(ProgramPointer[LineCounter].address,
ProgramPointer[LineCounter].data)) {
      SerialPort.print("Error loading the Program to LSM6DSOX at line: ");
      SerialPort.println(LineCounter);
      while(1) {
        // Led blinking.
        digitalWrite(LED_BUILTIN, HIGH);
        delay(250);
        digitalWrite(LED_BUILTIN, LOW);
        delay(250);
      }
    }
  }

  SerialPort.println("Program loaded inside the LSM6DSOX MLC");

  //Interrupts.
  pinMode(INT_1, INPUT);
  attachInterrupt(INT_1, INT1Event_cb, RISING);

  /* We need to wait for a time window before having the first MLC status */
  delay(3000);

  AccGyr.Get_MLC_Output(mlc_out);
  printMLCStatus(mlc_out[0]);
}

void loop() {
```

```
  if (mems_event) {
    mems_event=0;
    LSM6DSOX_MLC_Status_t status;
    AccGyr.Get_MLC_Status(&status);
    if (status.is_mlc1) {
      uint8_t mlc_out[8];
      AccGyr.Get_MLC_Output(mlc_out);
      printMLCStatus(mlc_out[0]);
    }
  }
}

void INT1Event_cb() {
  mems_event = 1;
}

void printMLCStatus(uint8_t status) {
  switch(status) {
    case 0:
      SerialPort.println("Activity: Stationary");
      break;
    case 1:
      SerialPort.println("Activity: Walking");
      break;
    case 4:
      SerialPort.println("Activity: Jogging");
      break;
    case 8:
      SerialPort.println("Activity: Biking");
      break;
    case 12:
      SerialPort.println("Activity: Driving");
      break;
    default:
      SerialPort.println("Activity: Unknown");
      break;
  }
}
```

```
X_NUCLEO_IKS01A3_LSM6DSOX_MLC-1 Nano-RP2040-Connect port is using /dev/ttyACM0
Activity: Stationary
Activity: Jogging
Activity: Walking
Activity: Biking
Activity: Stationary
```

wifi-rp2040-ap

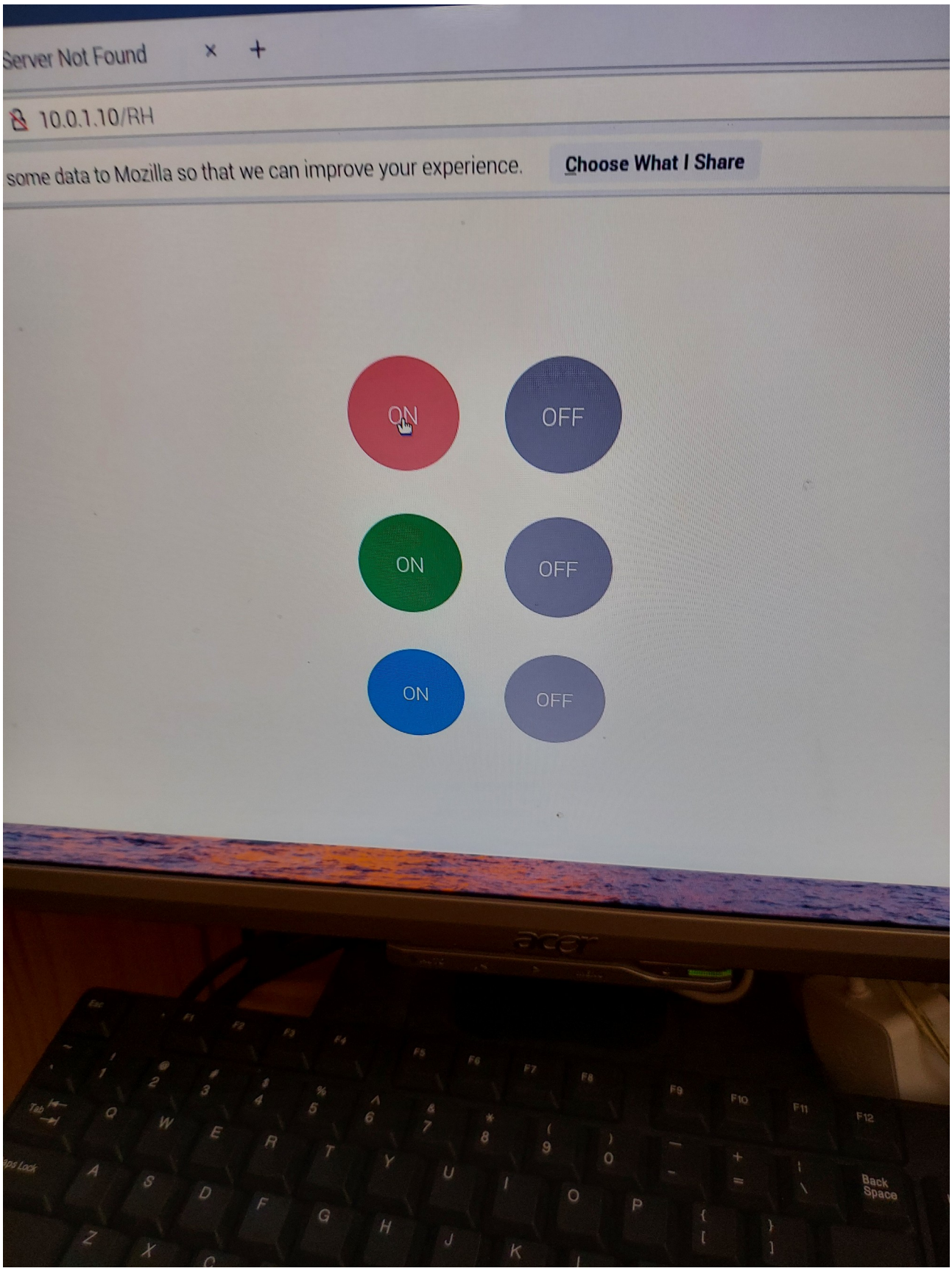
```
#include <SPI.h>
#include <WiFiNINA.h>

char ssid[] = "nanotest";        // your network SSID (name)
```

```
char pass[] = "12345678";    // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0;                // your network key index number (needed only for WEP)

int status = WL_IDLE_STATUS;
WiFiServer server(80);

void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  Serial.println("Access Point Web Server");

  pinMode(LEDR, OUTPUT);
  pinMode(LEDG, OUTPUT);
  pinMode(LEDB, OUTPUT);

  // check for the WiFi module:
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // don't continue
    while (true);
  }

  String fv = WiFi.firmwareVersion();
  if (fv < WIFI_FIRMWARE_LATEST_VERSION) {
    Serial.println("Please upgrade the firmware");
  }

  // by default the local IP address will be 192.168.4.1
  // you can override it with the following:
  WiFi.config(IPAddress(10, 0, 1, 10));

  // print the network name (SSID);
  Serial.print("Creating access point named: ");
  Serial.println(ssid);

  // Create open network. Change this line if you want to create an WEP network:
  status = WiFi.beginAP(ssid, pass);
  if (status != WL_AP_LISTENING) {
    Serial.println("Creating access point failed");
    // don't continue
    while (true);
  }

  // wait 10 seconds for connection:
  delay(10000);

  // start the web server on port 80
  server.begin();
```

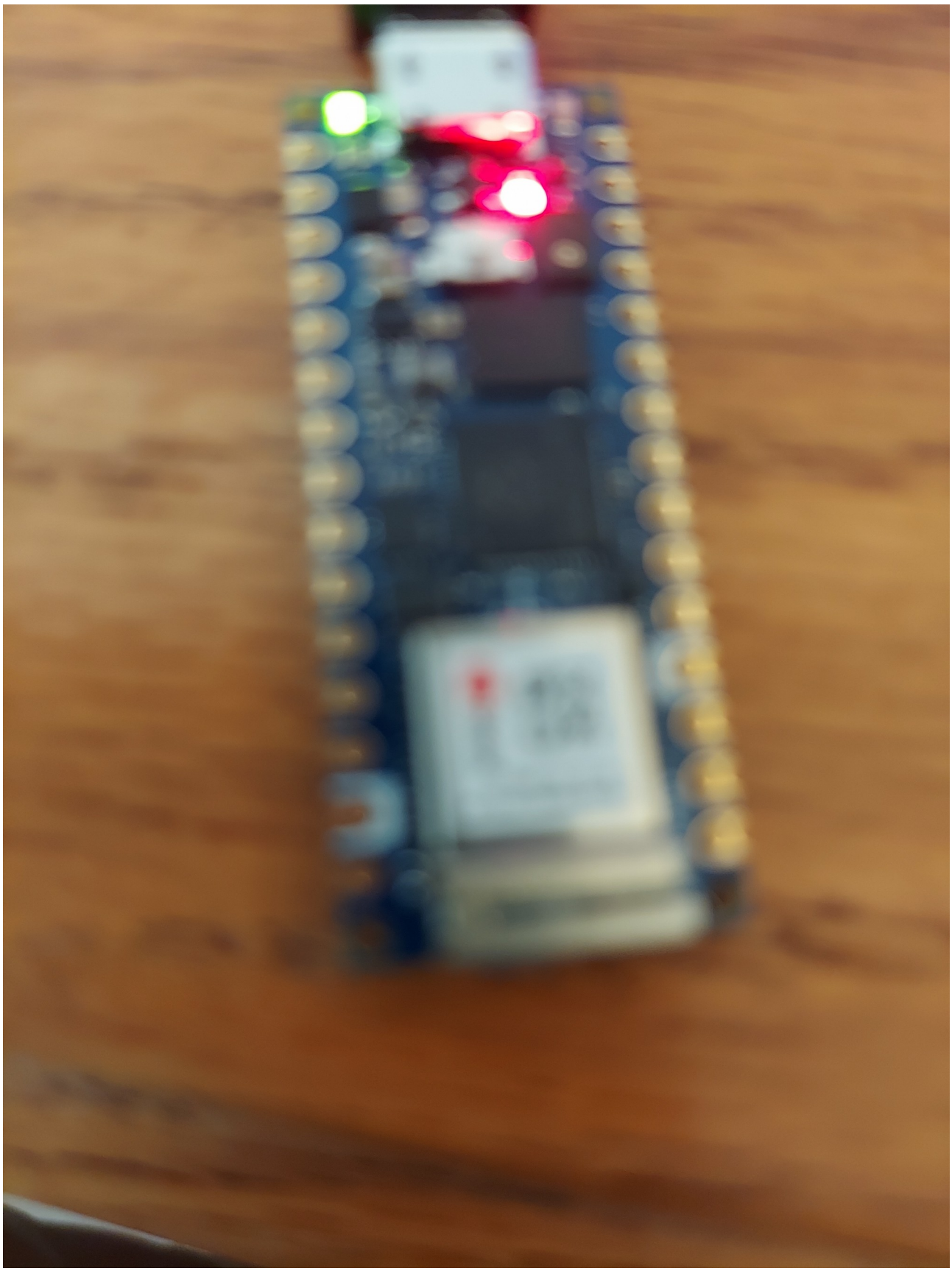
```
  // you're connected now, so print out the status
  printWiFiStatus();
}


void loop() {
  // compare the previous status to the current status
  if (status != WiFi.status()) {
    // it has changed update the variable
    status = WiFi.status();

    if (status == WL_AP_CONNECTED) {
      // a device has connected to the AP
      Serial.println("Device connected to AP");
    } else {
      // a device has disconnected from the AP, and we are back in listening mode
      Serial.println("Device disconnected from AP");
    }
  }

  WiFiClient client = server.available();   // listen for incoming clients

   if (client) {                             // if you get a client,
    Serial.println("new client");           // print a message out the serial port
    String currentLine = "";                // make a String to hold incoming data from the client
    while (client.connected()) {            // loop while the client's connected
      if (client.available()) {             // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        if (c == '\n') {                    // if the byte is a newline character

          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();

            // the content of the HTTP response follows the header:
            client.print("<style>");
            client.print(".container {margin: 0 auto; text-align: center; margin-top: 100px;}");
            client.print("button {color: white; width: 100px; height: 100px;");
            client.print("border-radius: 50%; margin: 20px; border: none; font-size: 20px; outline: none;
transition: all 0.2s;}");
            client.print(".red{background-color: rgb(196, 39, 39);}");
            client.print(".green{background-color: rgb(39, 121, 39);}");
            client.print(".blue {background-color: rgb(5, 87, 180);}");
            client.print(".off{background-color: grey;}");
            client.print("button:hover{cursor: pointer; opacity: 0.7;}");
```

```
            client.print("</style>");
            client.print("<div class='container'>");
            client.print("<button class='red' type='submit'
onmousedown='location.href=\"/RH\"'>ON</button>");
            client.print("<button class='off' type='submit'
onmousedown='location.href=\"/RL\"'>OFF</button><br>");
            client.print("<button class='green' type='submit'
onmousedown='location.href=\"/GH\"'>ON</button>");
            client.print("<button class='off' type='submit'
onmousedown='location.href=\"/GL\"'>OFF</button><br>");
            client.print("<button class='blue' type='submit'
onmousedown='location.href=\"/BH\"'>ON</button>");
            client.print("<button class='off' type='submit'
onmousedown='location.href=\"/BL\"'>OFF</button>");
            client.print("</div>");

            // The HTTP response ends with another blank line:
            client.println();
            // break out of the while loop:
            break;
          } else {    // if you got a newline, then clear currentLine:
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a carriage return character,
          currentLine += c;      // add it to the end of the currentLine
        }

        // Check to see if the client request was /X
        if (currentLine.endsWith("GET /RH")) {
          digitalWrite(LEDR, HIGH);
        }
        if (currentLine.endsWith("GET /RL")) {
          digitalWrite(LEDR, LOW);
        }
        if (currentLine.endsWith("GET /GH")) {
          digitalWrite(LEDG, HIGH);
        }
        if (currentLine.endsWith("GET /GL")) {
          digitalWrite(LEDG, LOW);
        }
        if (currentLine.endsWith("GET /BH")) {
          digitalWrite(LEDB, HIGH);
        }
        if (currentLine.endsWith("GET /BL")) {
          digitalWrite(LEDB, LOW);
        }
      }
    }
    // close the connection:
    client.stop();
    Serial.println("client disconnected");
  }
```

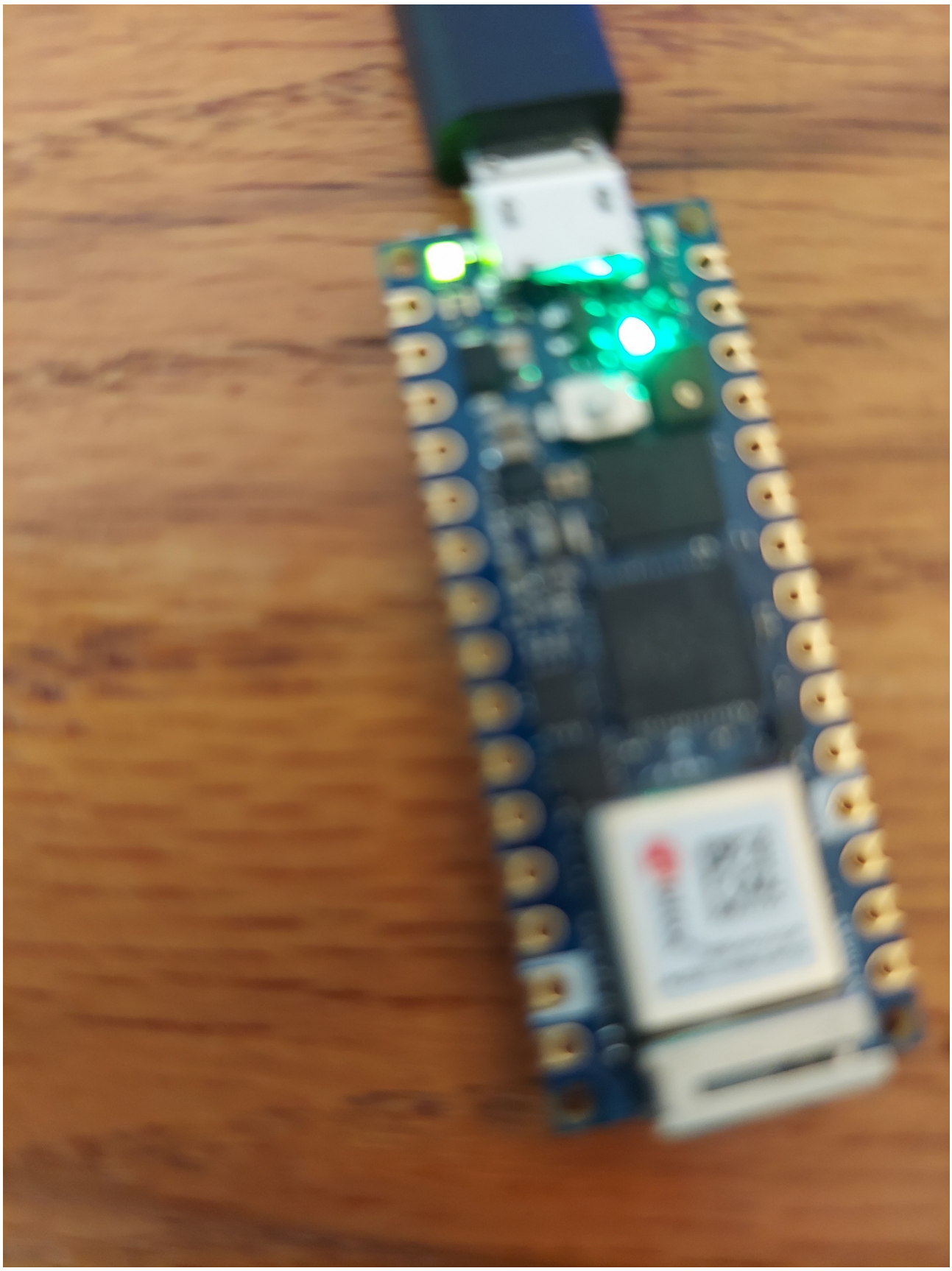
```
}

void printWiFiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print where to go in a browser:
  Serial.print("To see this page in action, open a browser to http://");
  Serial.println(ip);

}
```


Server Not Found
10.0.1.10/RH
some data to Mozilla so that we can improve your experience.
Choose What I Share
ON
OFF
ON
OFF
ON
OFF
acer

xx



xx



xx