

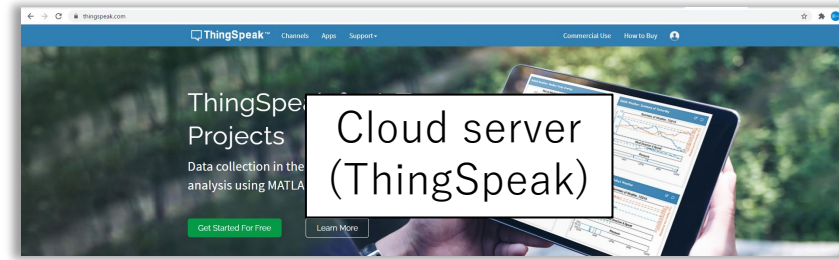
IoTを用いた実験状況の監視システム

Remote monitoring by IoT

Munekazu Horikoshi

Osaka City University

Today's goal

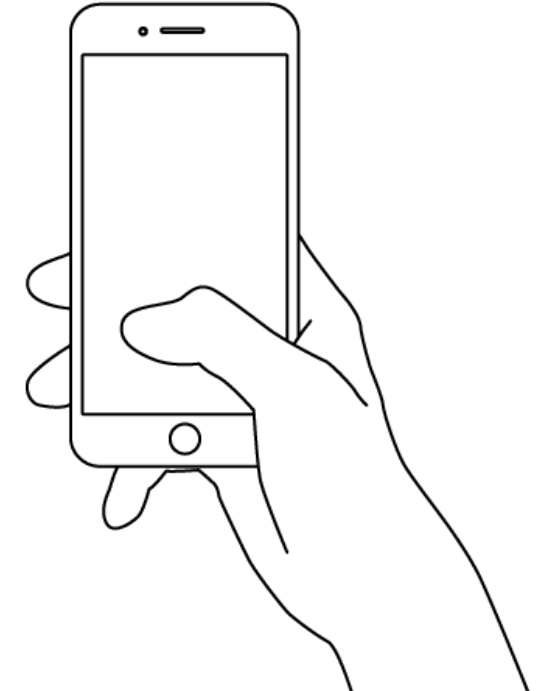
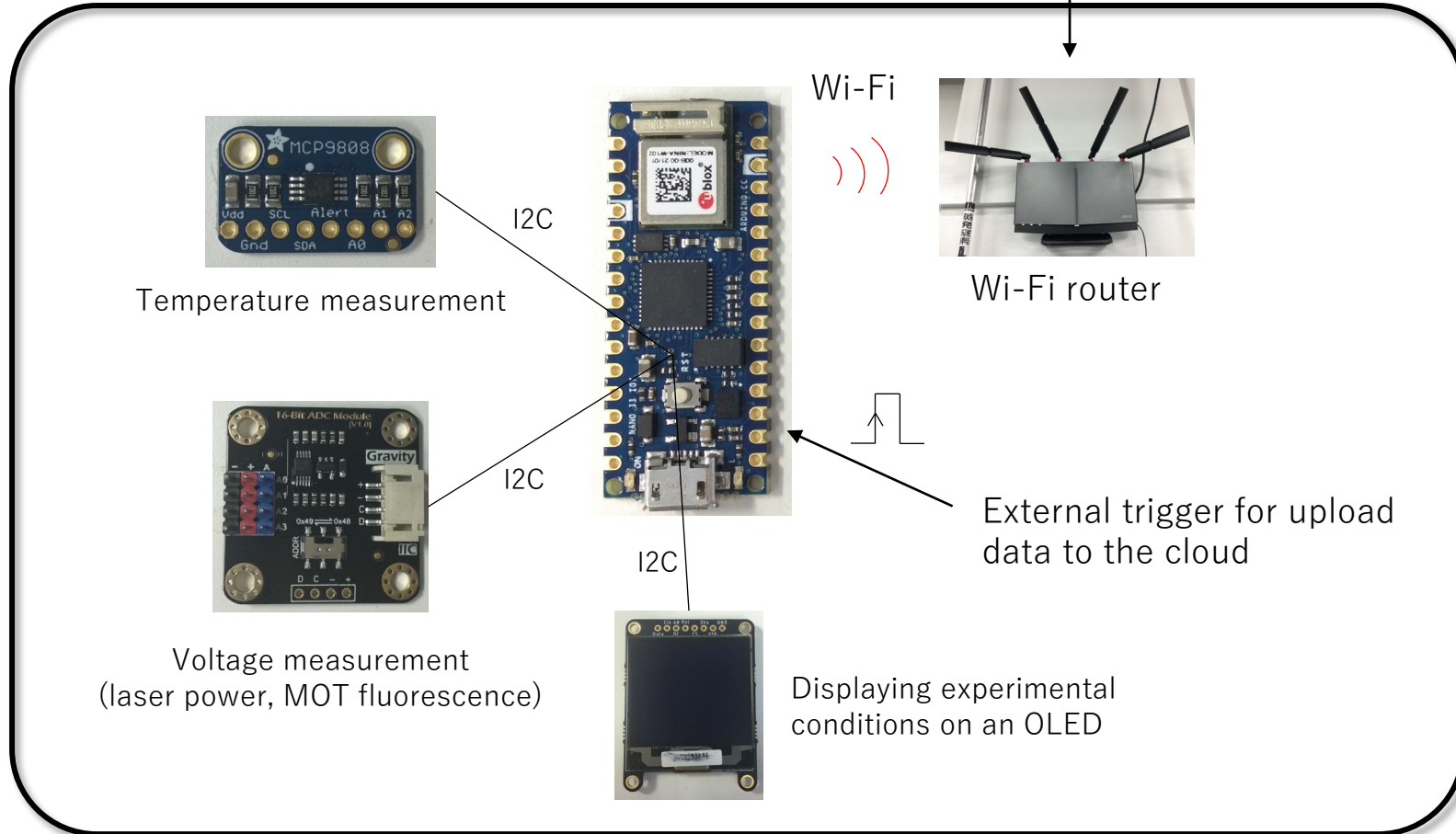


Internet

Inside your laboratory

Internet

Outside your laboratory



Contents

1. Arduino NANO 33 IOT
2. LED blink, Hello world
3. I2C communication
 - OLED display
 - Temperature sensor
 - ADC
4. Wi-Fi connection
5. IoT cloud server

Level : undergraduate students

This slide will be found in our laboratory HP

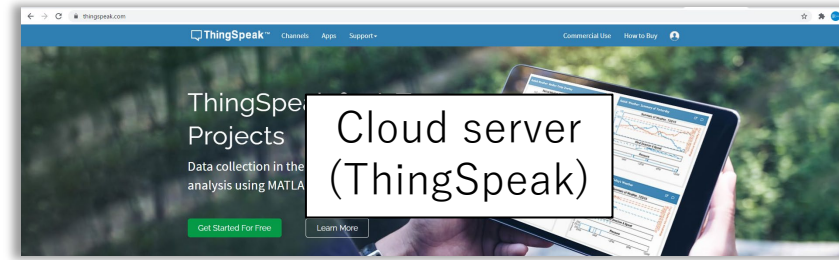


The screenshot shows a web browser window with the URL `sci.osaka-cu.ac.jp/phys/laser/etc.html`. The page title is "Ultracold Quantum Gas Lab". Below the title is a logo and Japanese text: "大阪市立大学 大学院理学研究科 レーザー量子物理学研究室 南部陽一郎物理学研究所". There are language selection buttons for "English" and "日本語". A navigation menu includes "Top", "Research", "Publications", "Members", "Contact", "Class", "Seminar", "Workshop", "Etc", and "Photos". The "Etc" button is highlighted with a red border.

- MATLABを使ってUSB経由でデータを取得するサンプルコード集
 - テクトロニクスのおシロスコープのデータを外部トリガーと同期して取得
 - Rohde & Schwarz FSC3 Spectrum Analyserのデータを取得
 - Pico TechnologyのTC-08 Thermocouple data loggerのデータを取得
 - ArduinoのI2C通信で3軸磁場センサー-LIS3MDLの値を取得
 - ArduinoのI2C通信で温度センサー-MCP9808の値を取得
- MATLABによる2次元フィッティング
[2次元フィッティングデモファイル](#)
デモファイルではランダムな幅とピーク位置を持つノイズ付き2次元ガウス分布を生成し、そのデータを2次元フィッティングします。



Today's goal

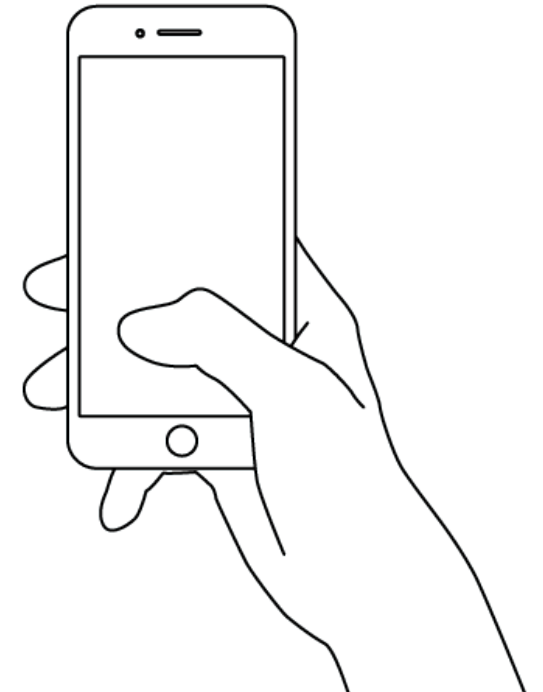
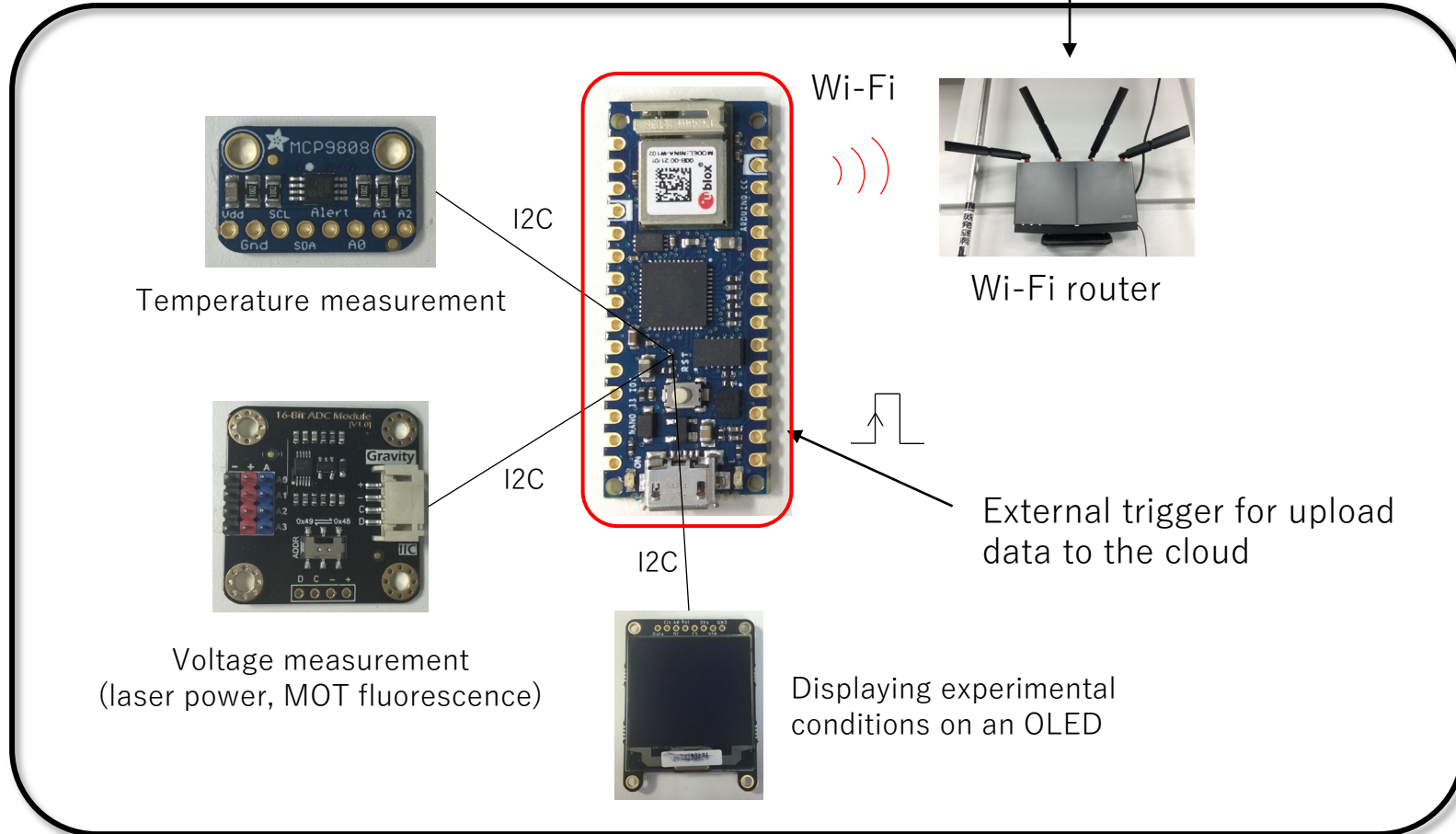


Internet

Inside your laboratory

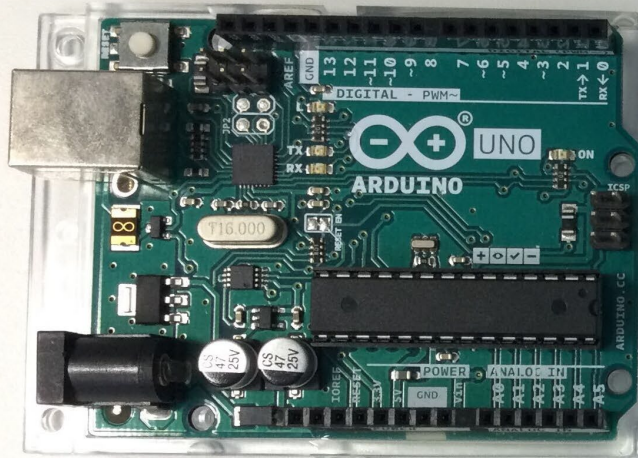
Internet

Outside your laboratory

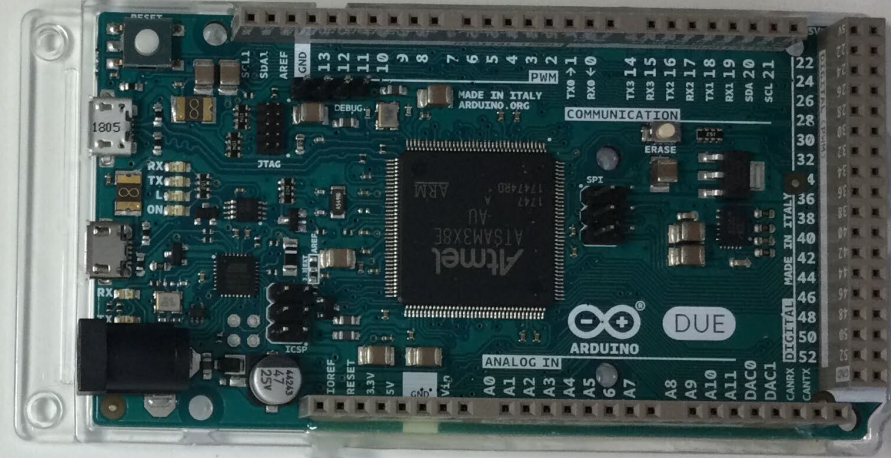


Arduino

- Programmable small computer with C-language by a free development tool
- Digital in/out, analog in, various communication
- Standalone operating system
- Low cost

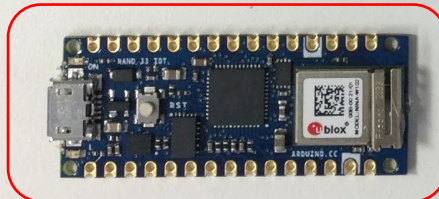


~2,600 yen



~5,000 yen

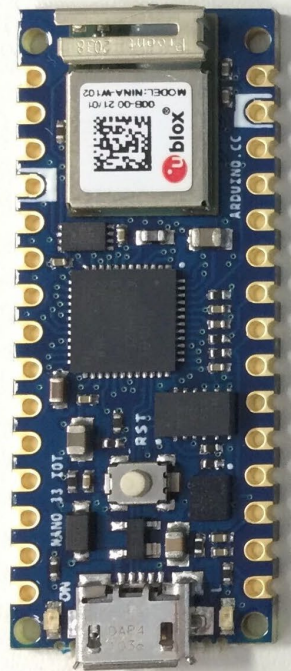
NANO 33 IOT



~2,900 yen



Arduino NANO 33 IOT



STORE.ARDUINO.CC/NANO-33-IOT

SMALL, POWERFUL, EFFICIENT & IoT CONNECTED. EVERY PROJECT MATTERS.

Connectivity	Wi-Fi	BLE 4.2			
Chip	ATSAMD21				
Clock	48 MHz				
Memory	256 KB FLASH	32 KB SRAM			
Interfaces	USB	SPI	I2C	I2S	UART
Voltages	5V INPUT-USB	4.5-21V INPUT-VIN	3.3V OPERATING		
Pinout	14 DIGITAL	6 PWM	8 ANALOG		
Dimensions	18 x 45 mm				











Be careful

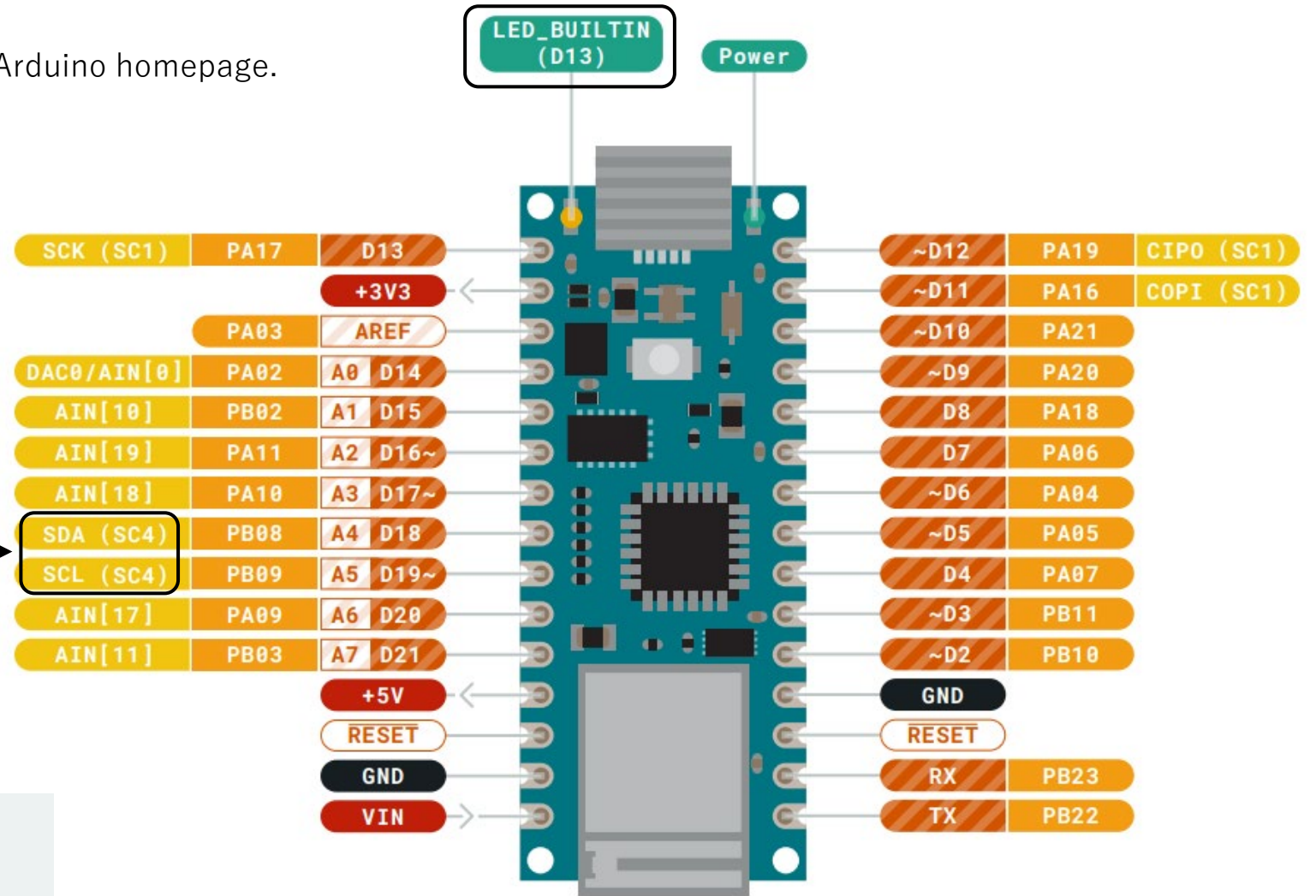
Arduino NANO 33 IOT

The full pinout diagram is available from the Arduino homepage.
<https://store.arduino.cc/usa/nano-33-iot>

For LED blink (Lチカ)

For I2C communication →

 Ground	 Digital Pin
 Power	 Analog Pin
 LED	 Other Pin
 Internal Pin	 Microcontroller's Port
 SWD Pin	 Default

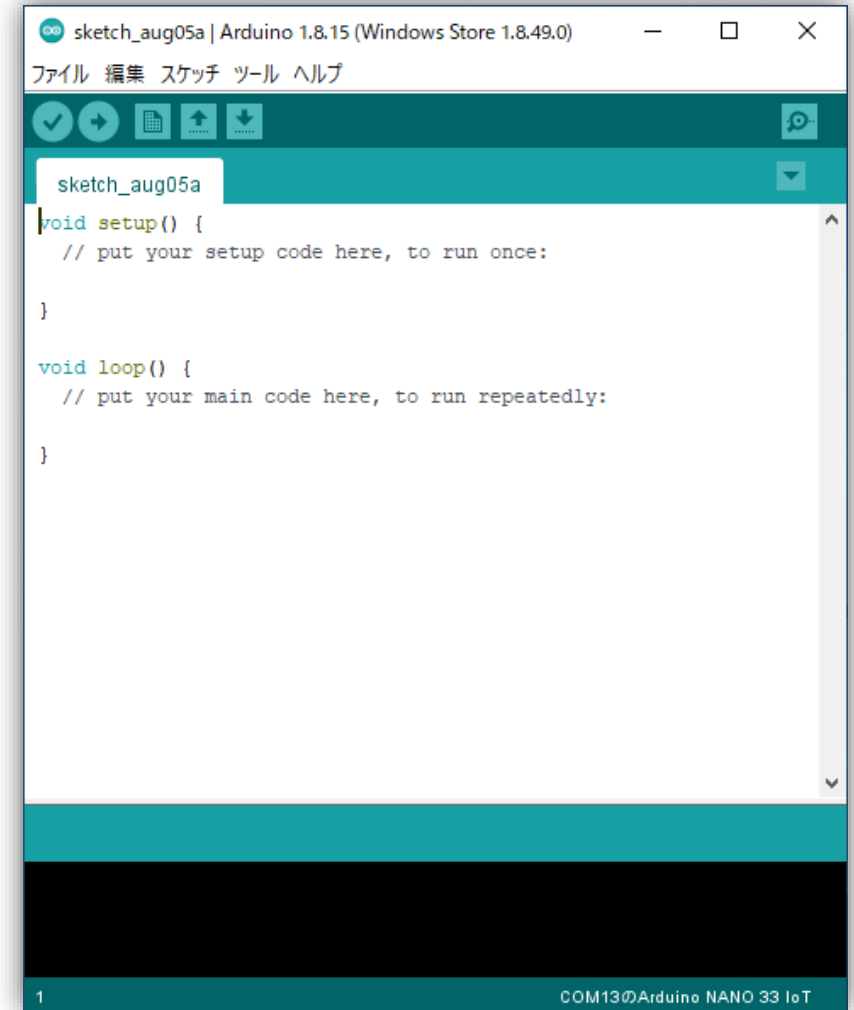


How to program it

- Install Arduino IDE (free) to your PC and launch it (<https://www.arduino.cc/en/software>)



The screenshot shows the Arduino IDE 1.8.15 download page. On the left, there is the Arduino logo and the text "Arduino IDE 1.8.15". Below this, it states: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board." and "Refer to the [Getting Started](#) page for Installation instructions." Under "SOURCE CODE", it says: "Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this](#) gpg key." On the right, under "DOWNLOAD OPTIONS", there are links for "Windows Win 7 and newer", "Windows ZIP file", "Windows app Win 8.1 or 10" (with a "Get" button), "Linux 32 bits", "Linux 64 bits", "Linux ARM 32 bits", "Linux ARM 64 bits", and "Mac OS X 10.10 or newer". At the bottom of the options section, there are links for "Release Notes" and "Checksums (sha512)".



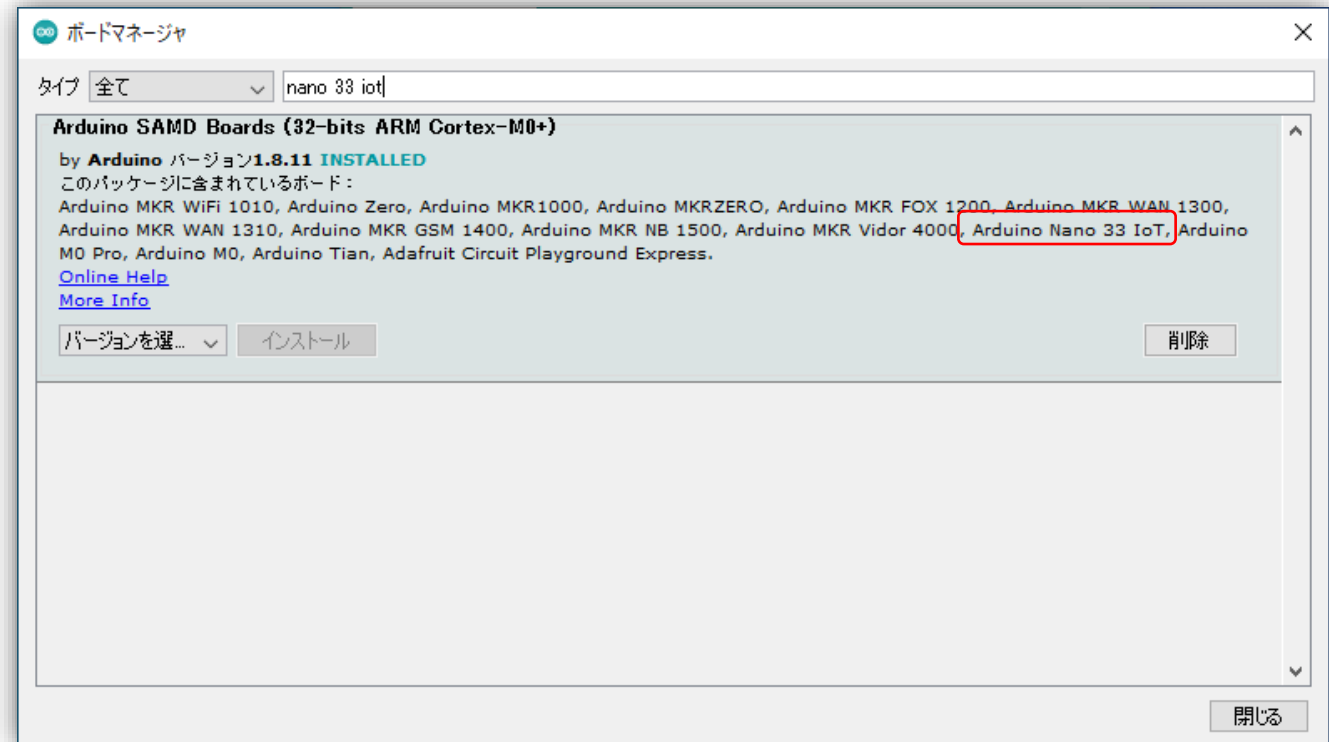
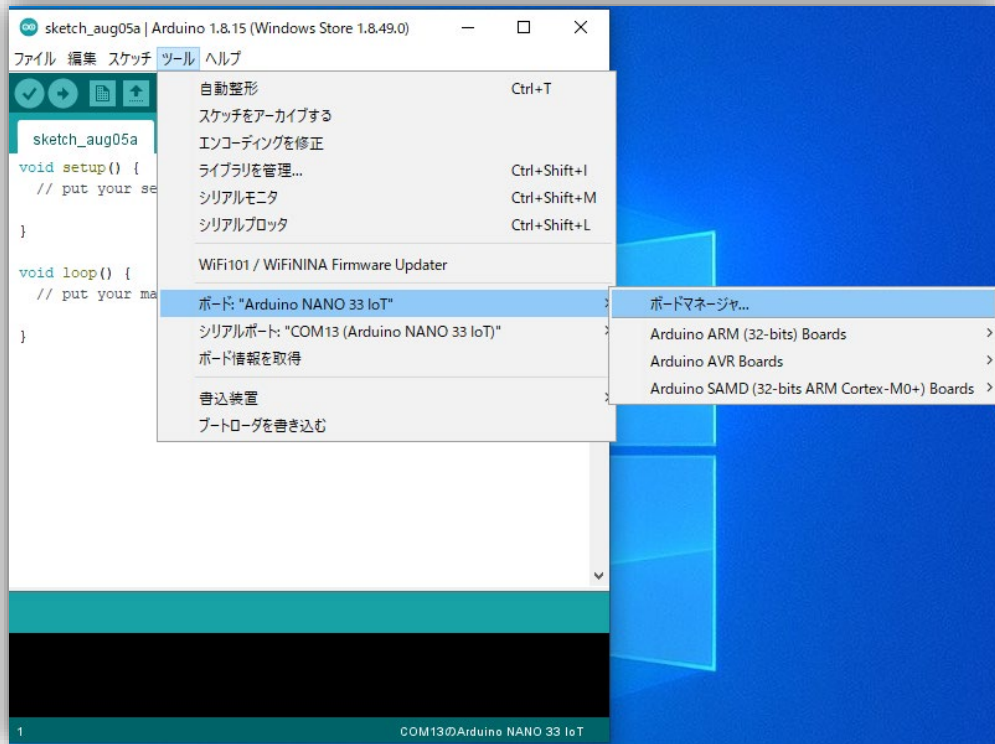
The screenshot shows the Arduino IDE interface. The title bar reads "sketch_aug05a | Arduino 1.8.15 (Windows Store 1.8.49.0)". The menu bar includes "ファイル", "編集", "スケッチ", "ツール", and "ヘルプ". The toolbar contains icons for checkmark, back, forward, save, upload, and download. The main editor area shows a sketch named "sketch_aug05a" with the following code:

```
void setup() {  
  // put your setup code here, to run once:  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
}
```

The status bar at the bottom shows "1" on the left and "COM13のArduino NANO 33 IoT" on the right.

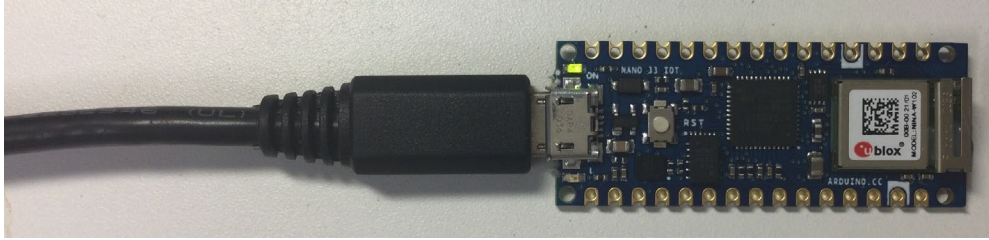
How to program it

- Open board manager and type 'nano 33 iot', and install a package for the CPU

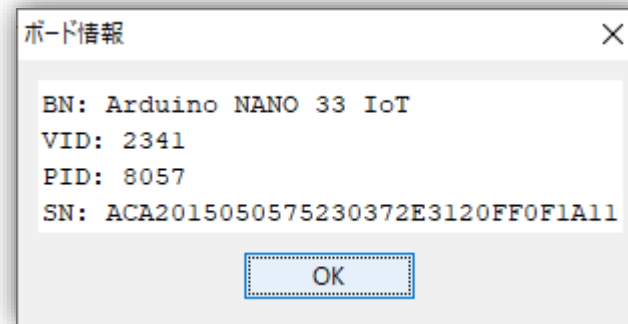
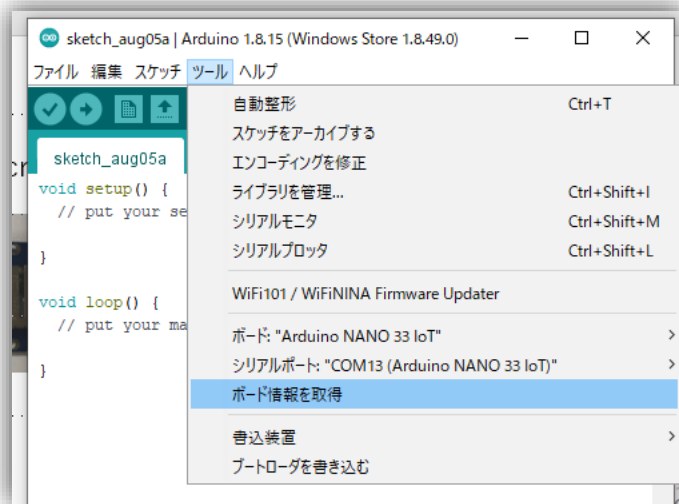
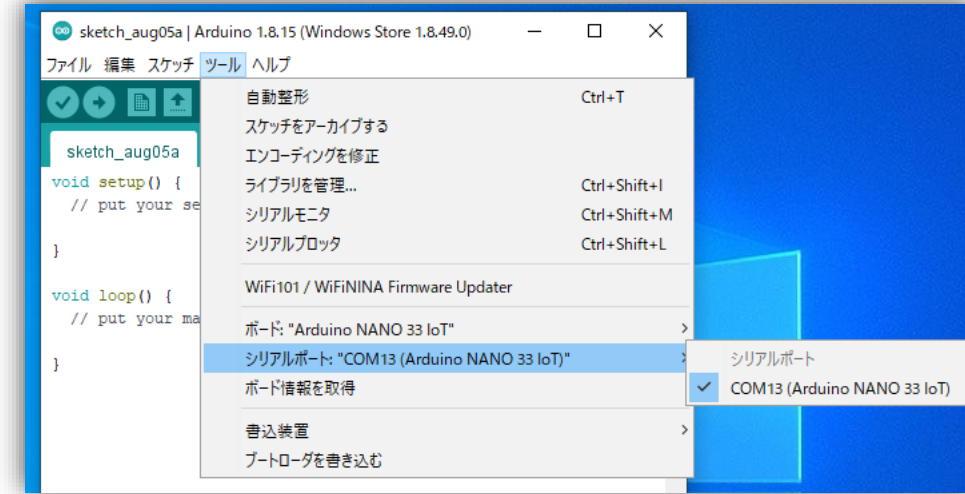


How to program it

- Connect your Arduino to the PC by a USB cable(micro-B type)

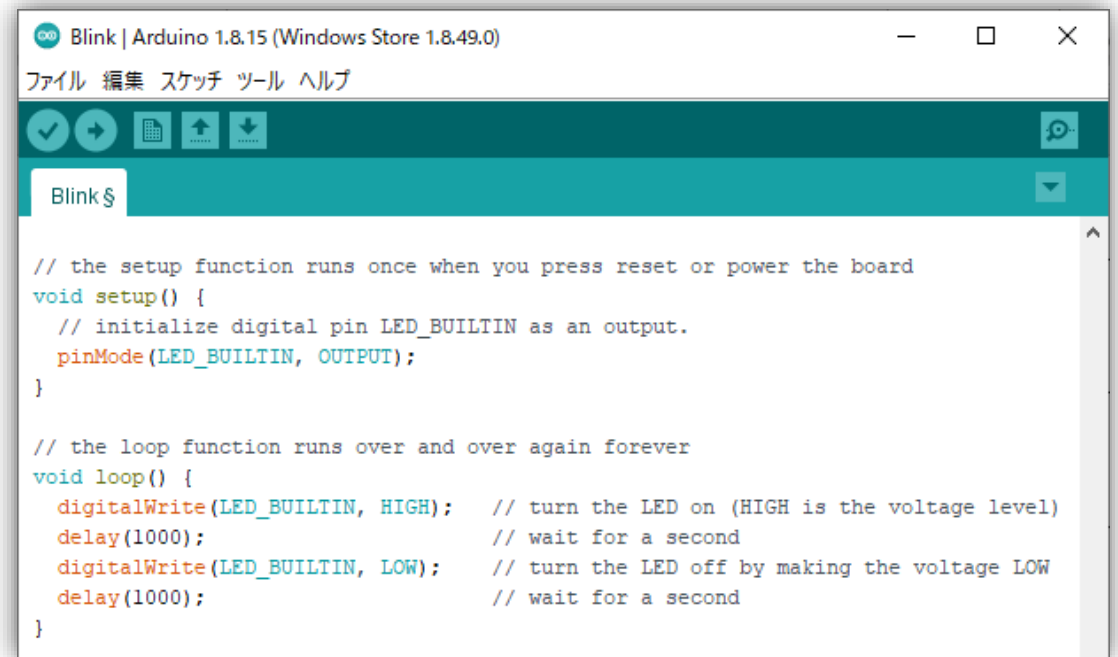
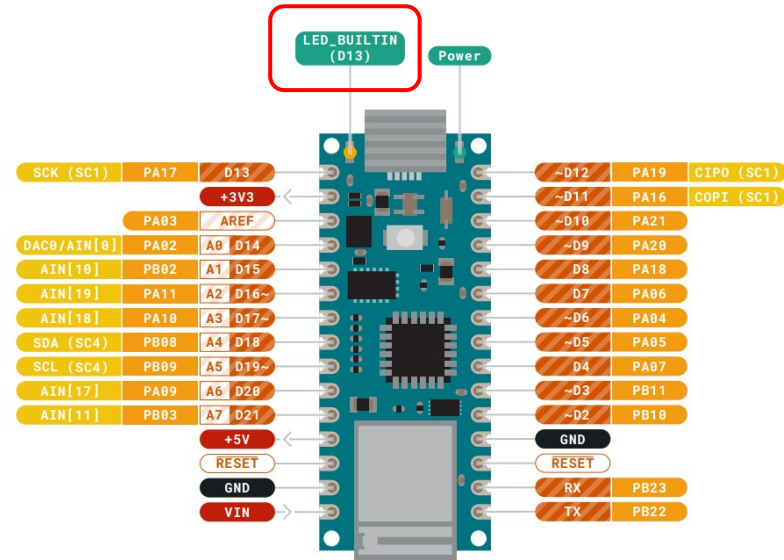
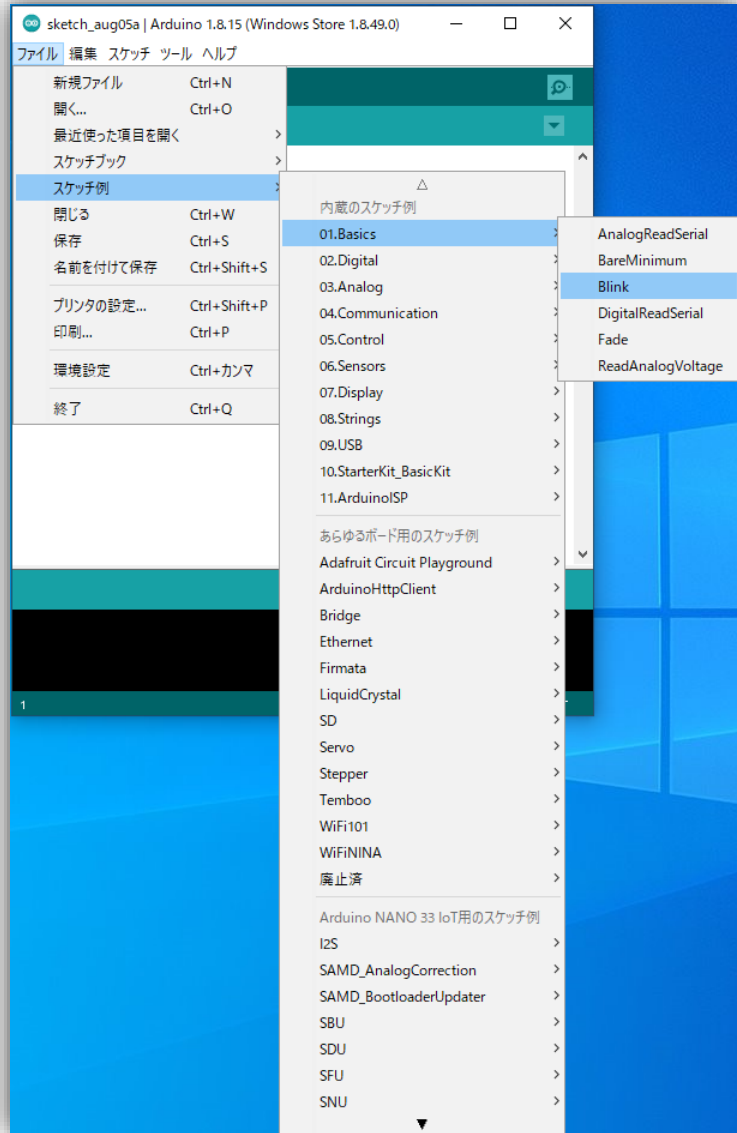


- Select a serial port connecting to the Arduino
- Confirm your Arduino



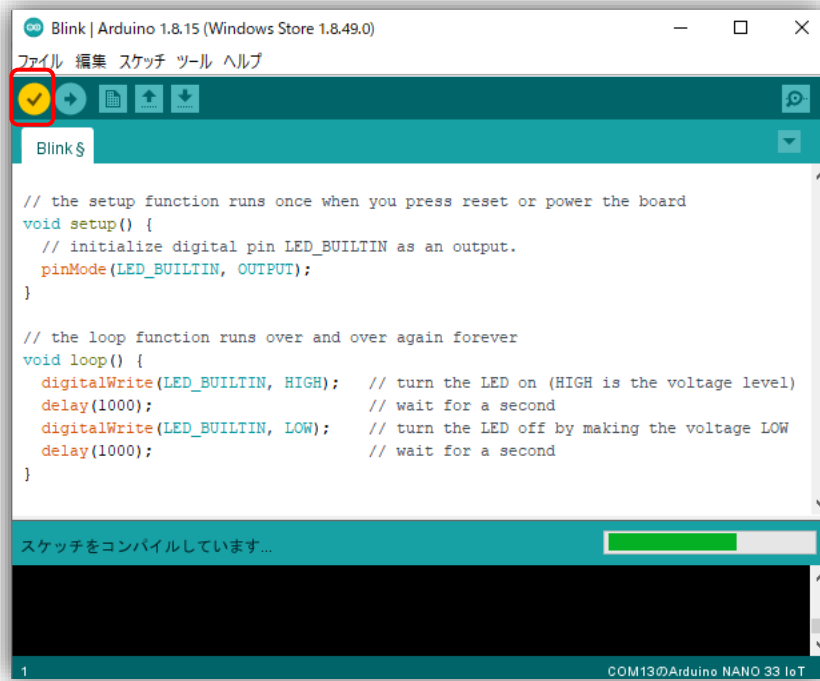
LED blink

- Open a sample code

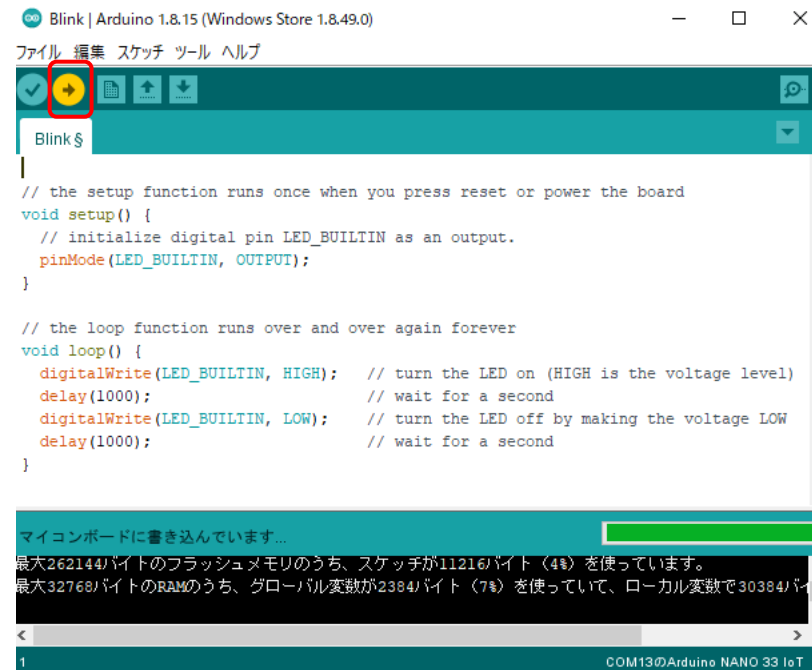


LED blink

- Compile



- Write it to your Arduino



LED blink

Change the output channel and the period

```
Blink | Arduino 1.8.15 (Windows Store 1.8.49.0)
ファイル 編集 スケッチ ツール ヘルプ

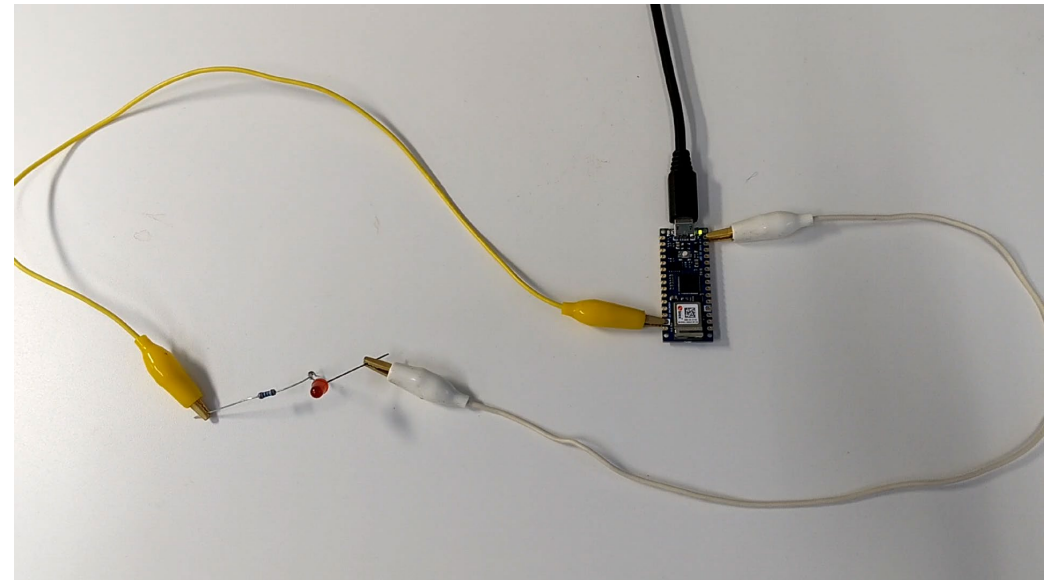
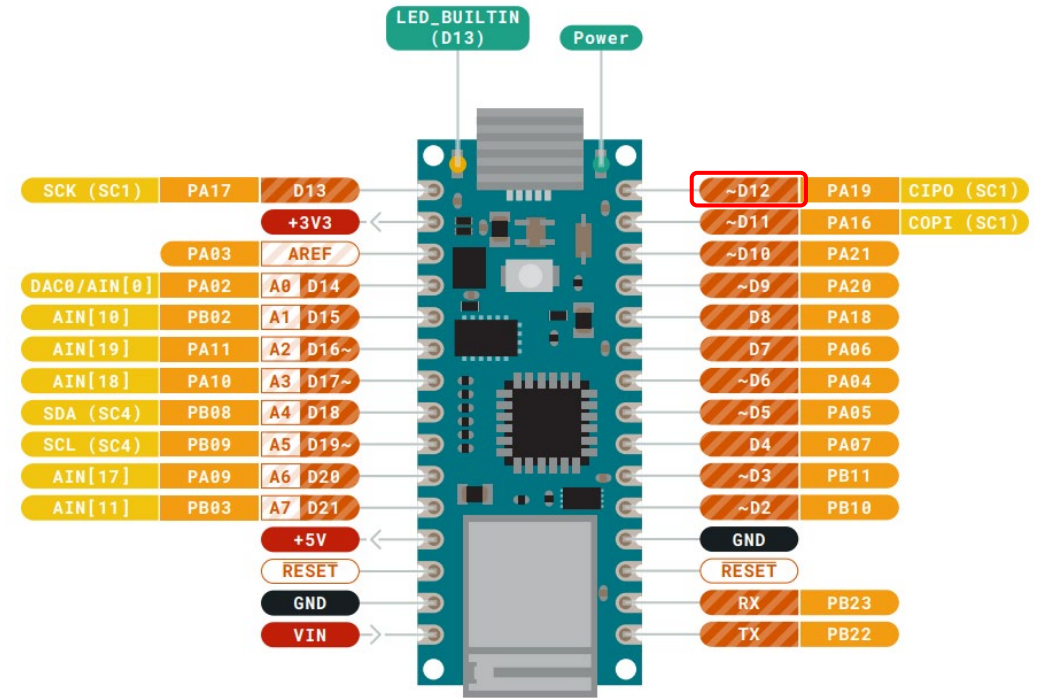
Blink$

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(12, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(12, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(500); // wait for a 0.5 second
  digitalWrite(12, LOW); // turn the LED off by making the voltage LOW
  delay(500); // wait for a 0.5 second
}

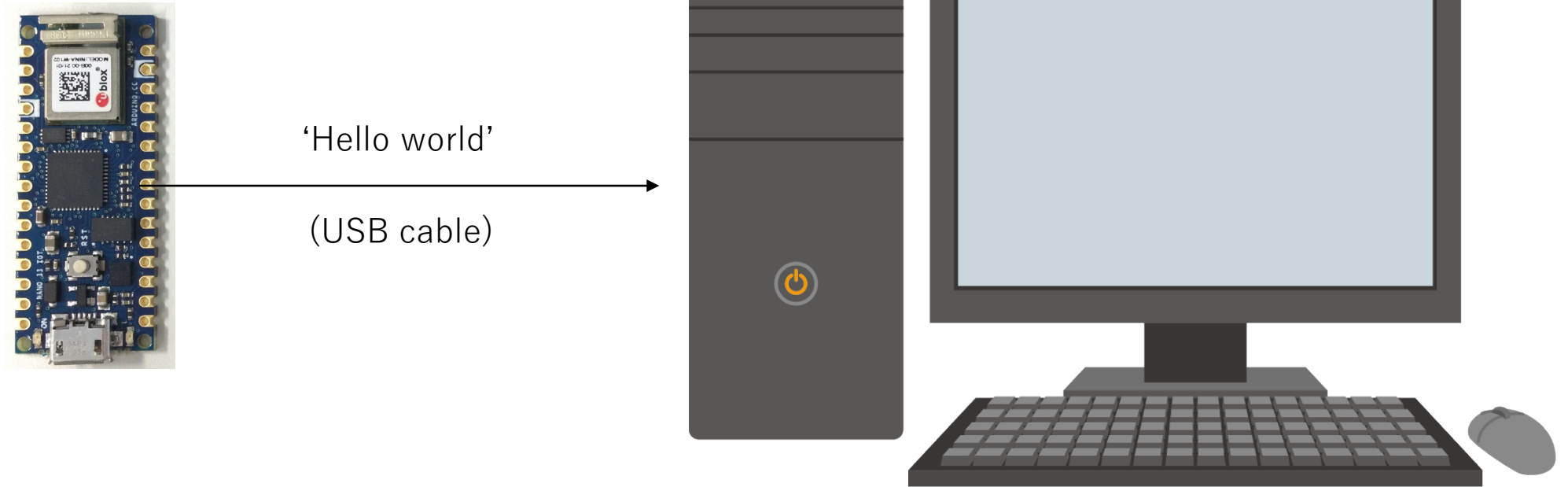
ボードへの書き込みが完了しました。
done in 0.011 seconds
CPU reset.

11 COM13のArduino NANO 33 IoT
```



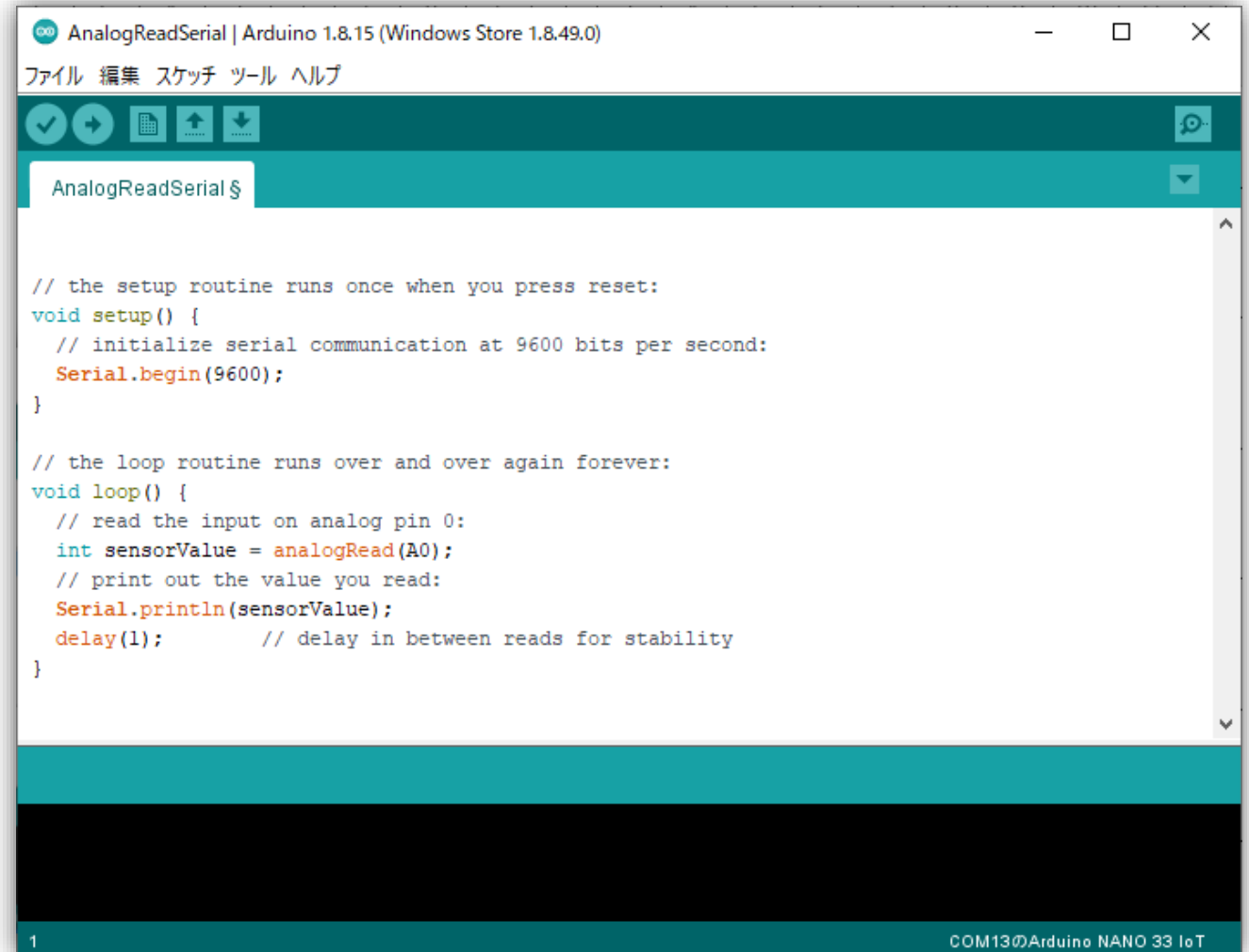
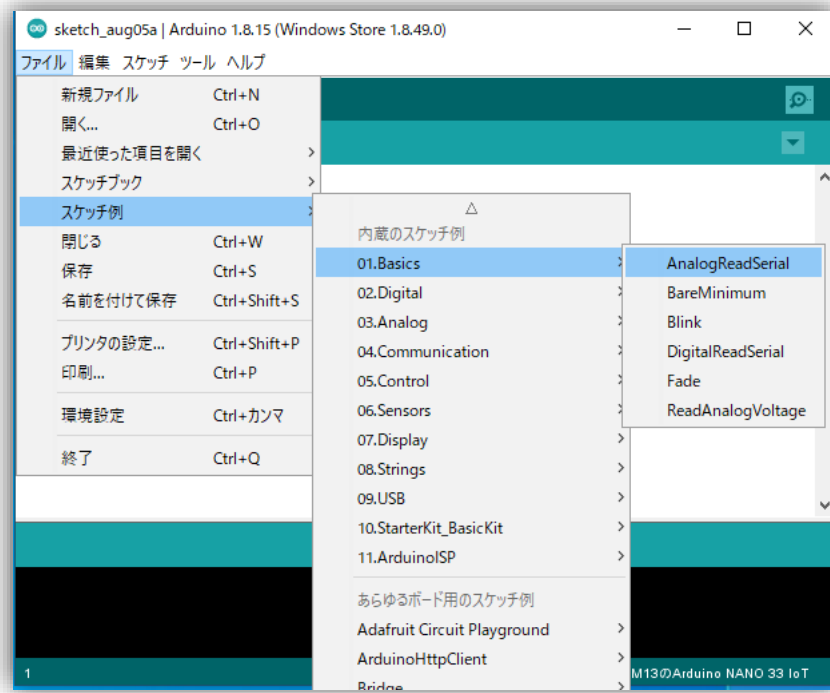
Hello world

Serial communication



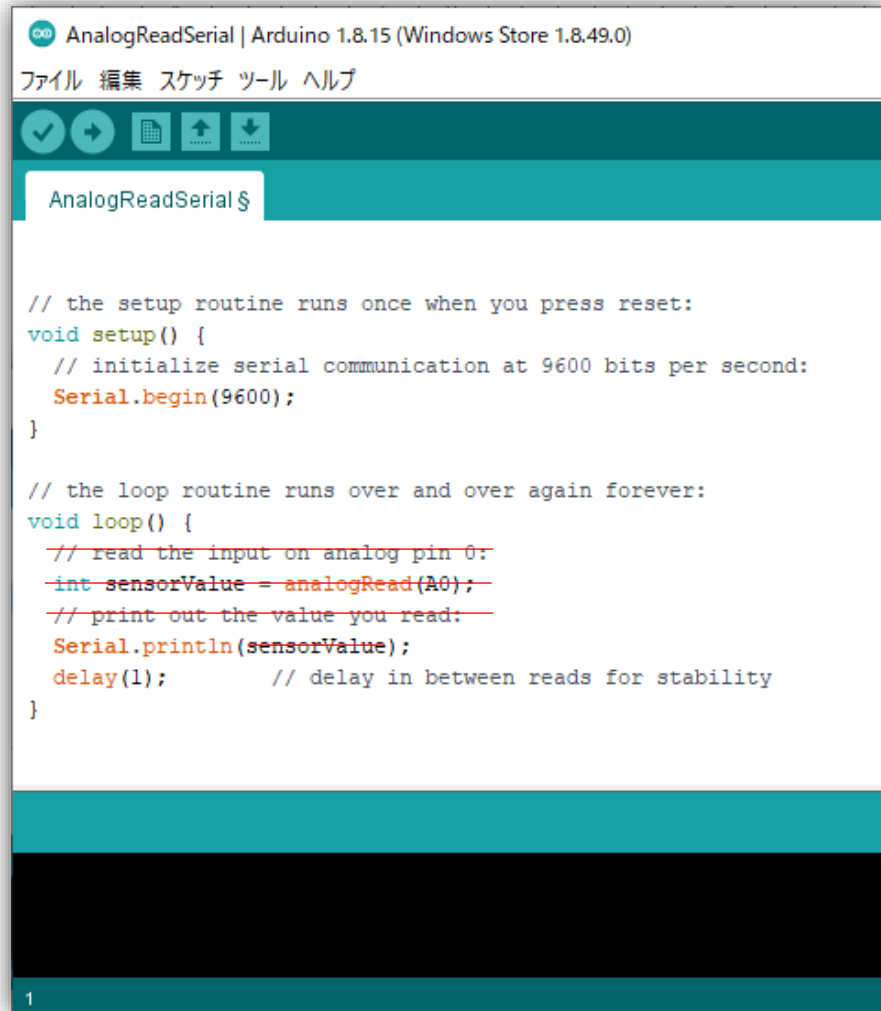
Hello world

- Open a sample code



Hello world

- Edit the code




```
AnalogReadSerial | Arduino 1.8.15 (Windows Store 1.8.49.0)
ファイル 編集 スケッチ ツール ヘルプ

AnalogReadSerial $

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);      // delay in between reads for stability
}
```



```
AnalogReadSerial | Arduino 1.8.15 (Windows Store 1.8.49.0)
ファイル 編集 スケッチ ツール ヘルプ

AnalogReadSerial

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

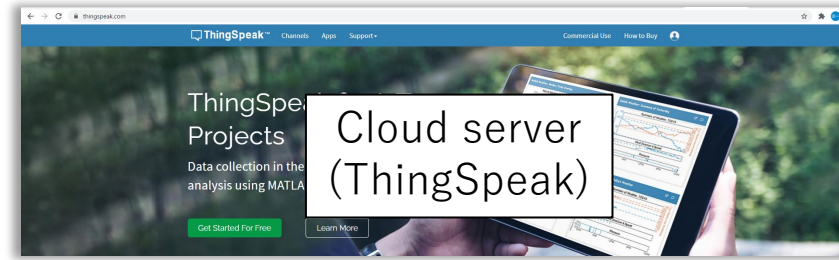
// the loop routine runs over and over again forever:
void loop() {
  String s = "Hello world";
  Serial.println(s);
  delay(1000);
}

ボードへの書き込みが完了しました。
Arduino : FAST_CHIP_ERASE
Arduino : FAST_MULTI_PAGE_WRITE
Arduino : CAN_CHECKSUM_MEMORY_BUFFER
<
14
```

Hello world



Today's goal

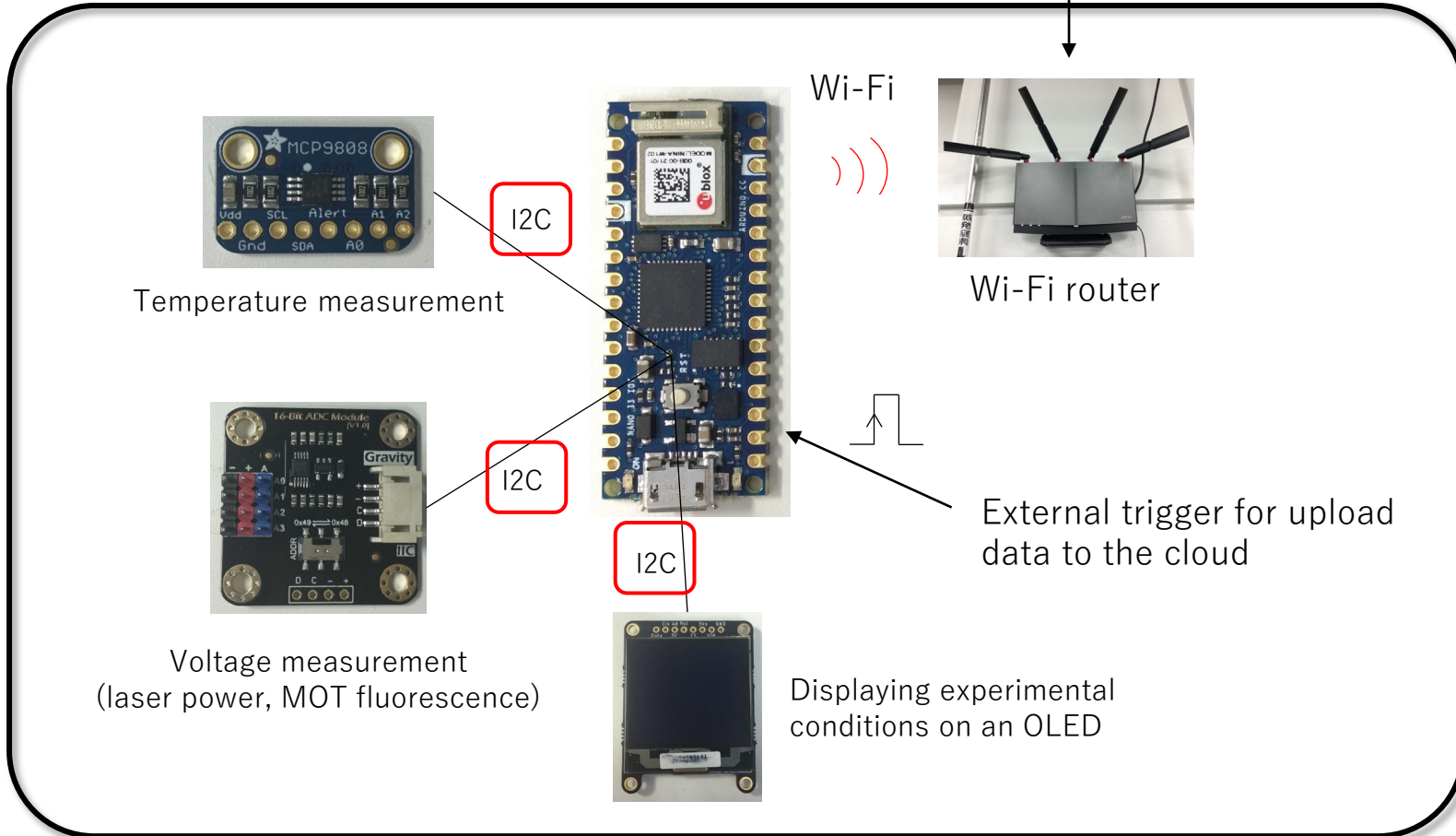


Internet

Inside your laboratory

Internet

Outside your laboratory



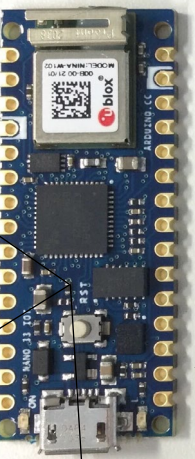
Temperature measurement

I2C



Voltage measurement
(laser power, MOT fluorescence)

I2C



I2C

Wi-Fi



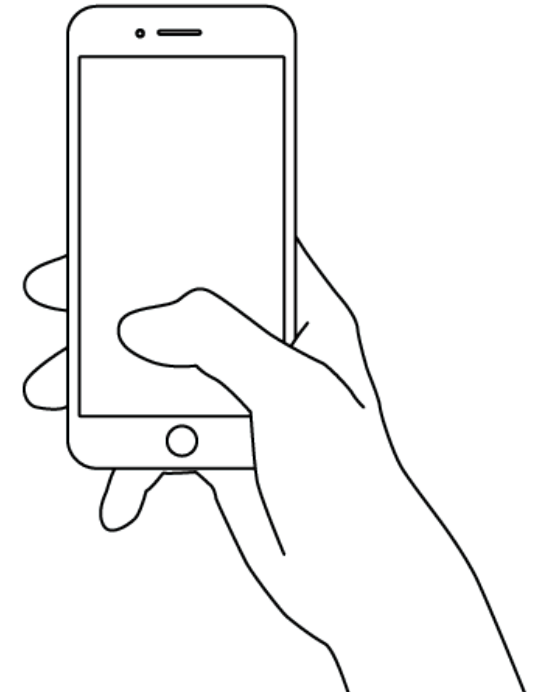
Wi-Fi router



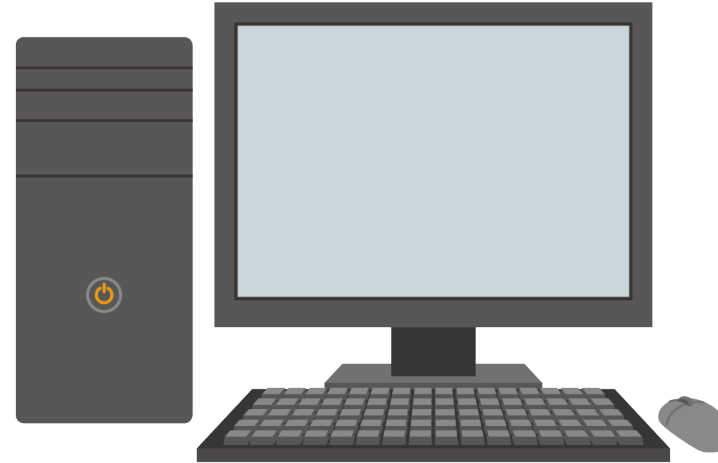
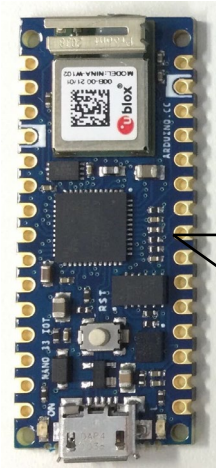
External trigger for upload
data to the cloud



Displaying experimental
conditions on an OLED



Change the display for standalone operating

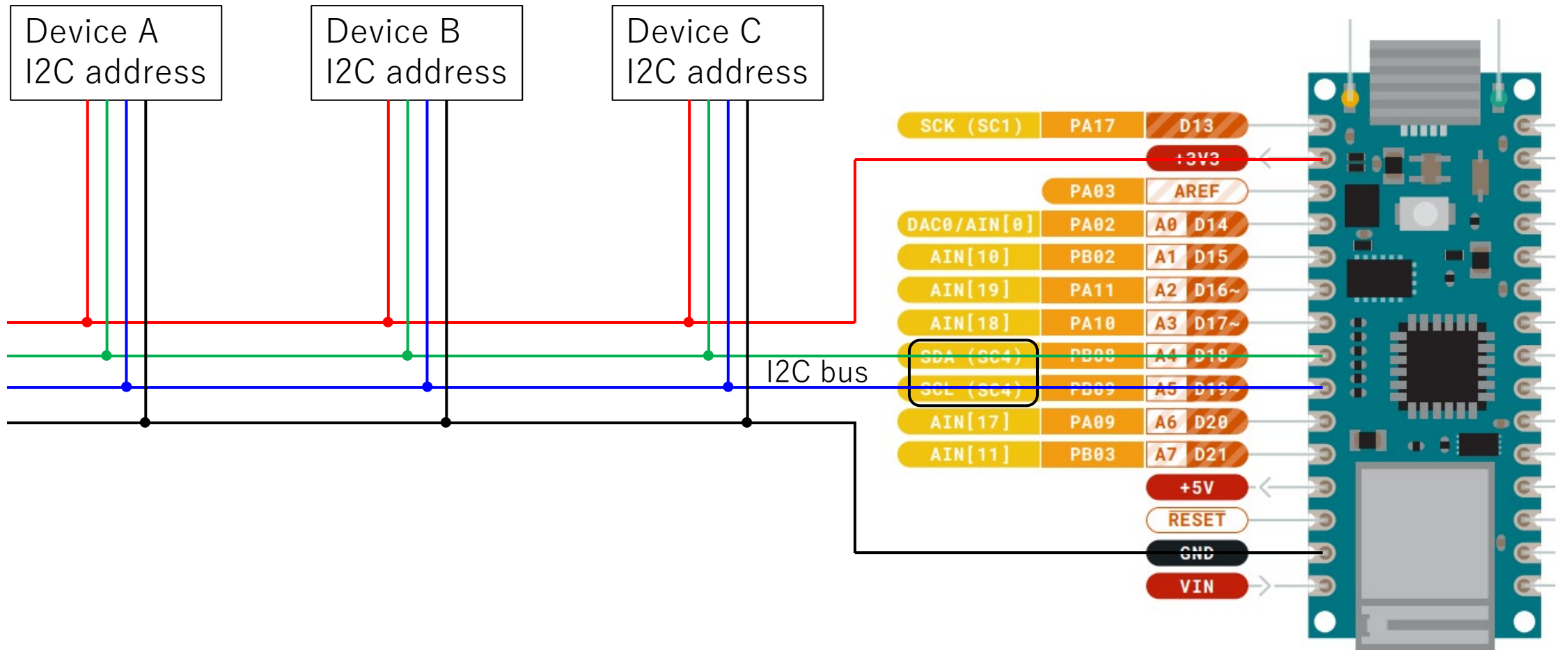


'Hello world'
(I2C communication)



OLED
(Organic Light Emitting Diode)

I2C communication



Various digital devices with I2C communication

Grayscale 1.5" 128x128 OLED Display

'ADAFRUIT SSD1327'

~3,000 yen

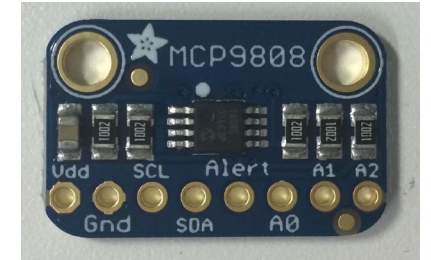


Accuracy Temperature Sensor

'Adafruit MCP9808'

- Up to 8 on a single I2C bus with adjustable address pins
- 0.25° C typical precision over -40°C to 125°C range
- 0.0625° C resolution

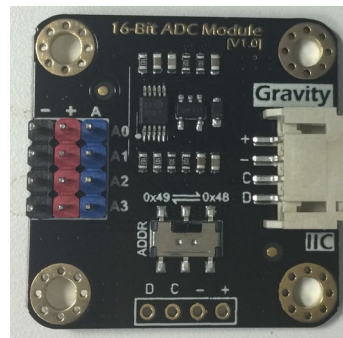
~700 yen



ADS1115 16-Bit ADC - 4 Channel with Programmable Gain Amplifier

'DFR0553'

~1,400 yen

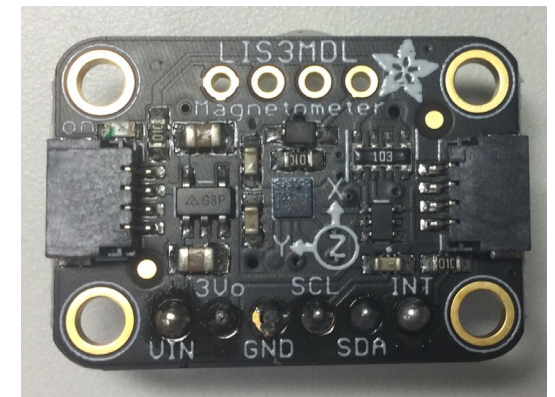


Triple-axis Magnetometer

'Adafruit LIS3MDL'

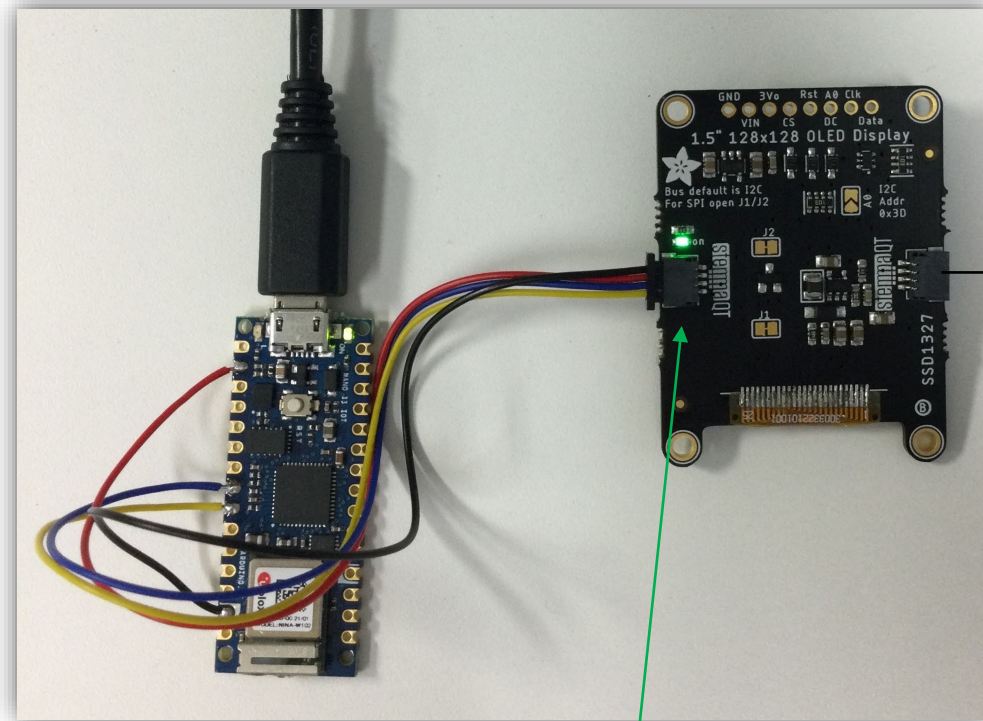
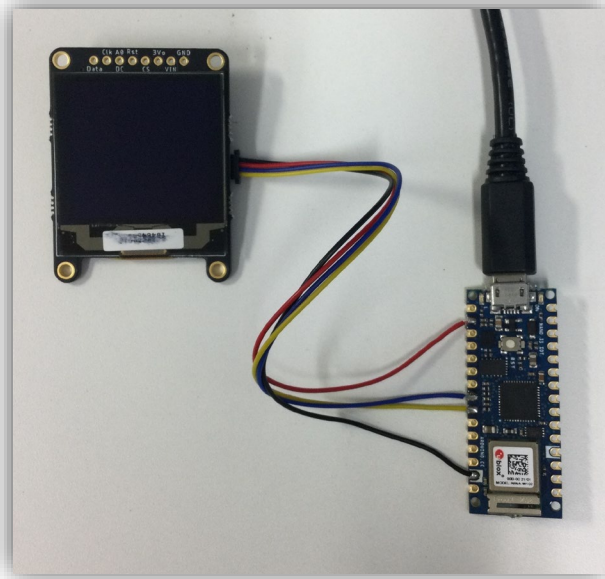
- $\pm 4 / \pm 8 / \pm 12 / \pm 16$ gauss selectable magnetic full scales
- 16-bit data output

~1,000 yen



Change the display for standalone operating

- I2C connection



Additional I2C devices

STEMMA QT / Qwiic connector

12 **[4209] JST SH 4-PIN TO PREMIUM MALE HEA**
電線・配線部材 > ケーブルアセンブリ (Digi-Key) > 長方形ケーブルアセンブリ **定格**

 1個以上 **¥160**
税込 ¥176

メーカー名: Adafruit
型番: 4209
納期: 確認する
品質ランク: S1(Digi-Key) 

[詳細を見る](#) [買カートに入れる](#)

13 **[4210] JST SH 4-PIN CABLE - QWIIC COMPA**
電線・配線部材 > ケーブルアセンブリ (Digi-Key) > 長方形ケーブルアセンブリ **定格**

 1個以上 **¥160**
税込 ¥176

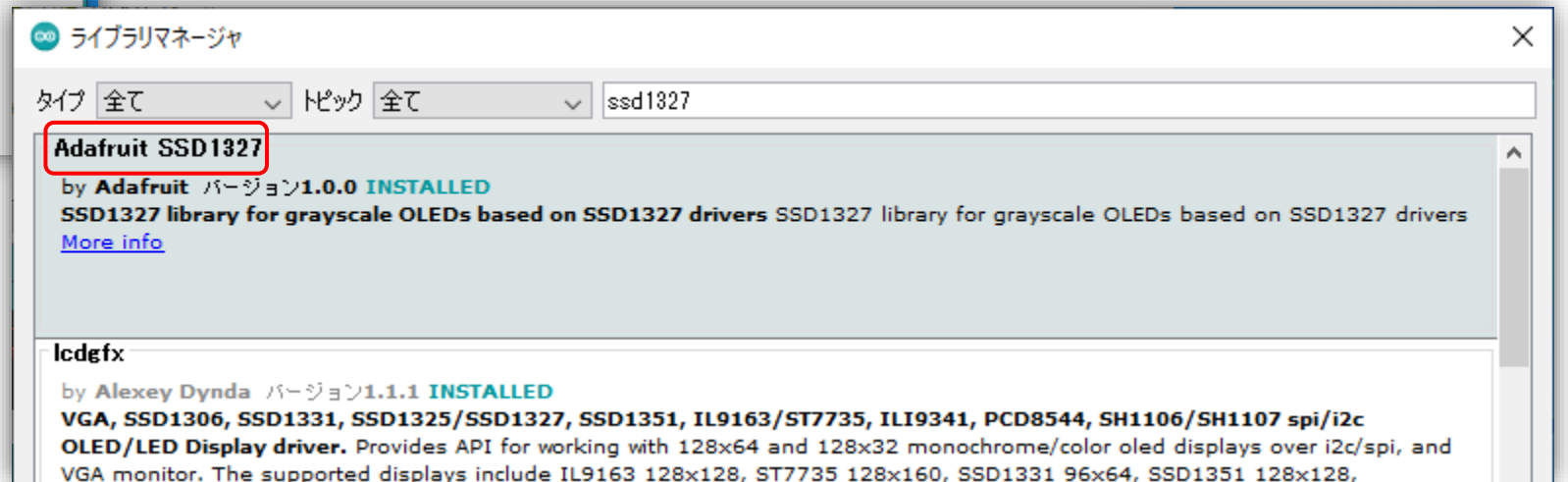
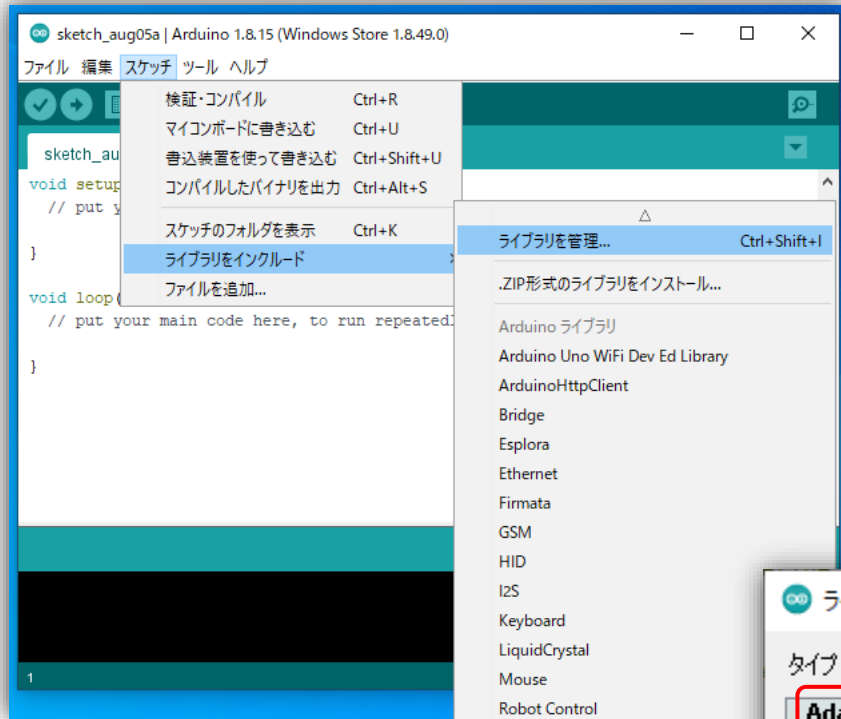
メーカー名: Adafruit
型番: 4210
納期: 確認する
品質ランク: S1(Digi-Key) 

提携先在庫数: 確認する

[詳細を見る](#) [買カートに入れる](#)

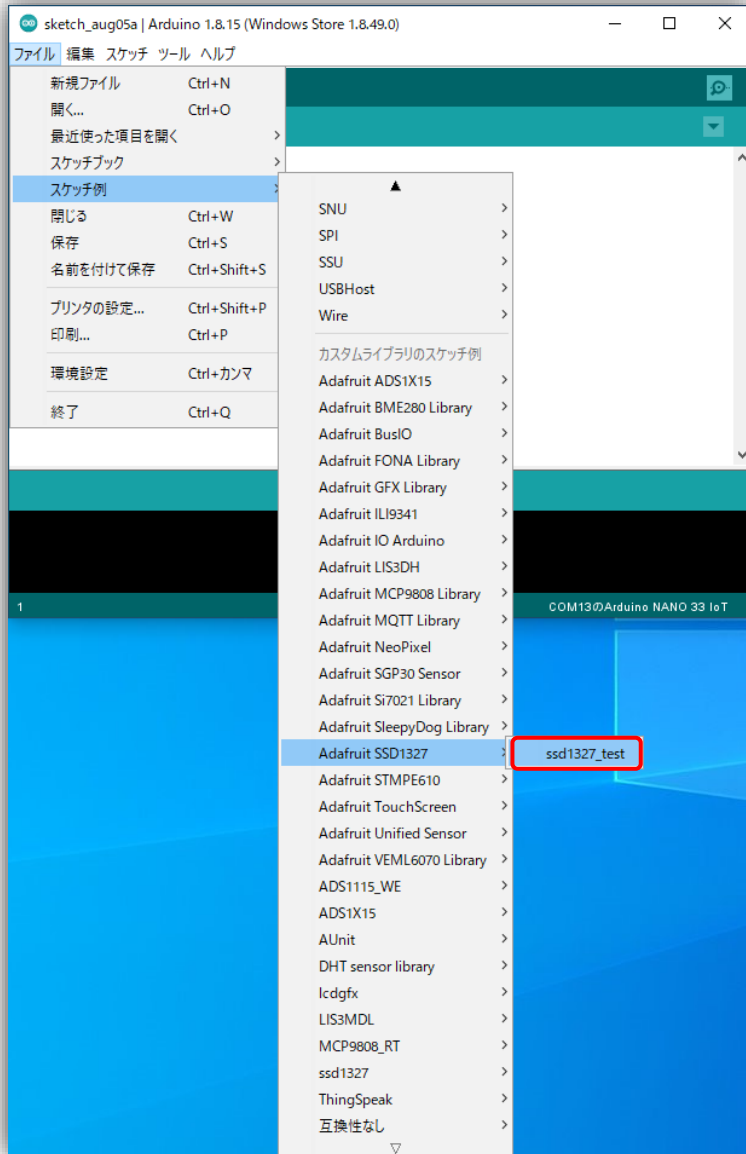
Change the display for standalone operating

- Install the library of SSD1327

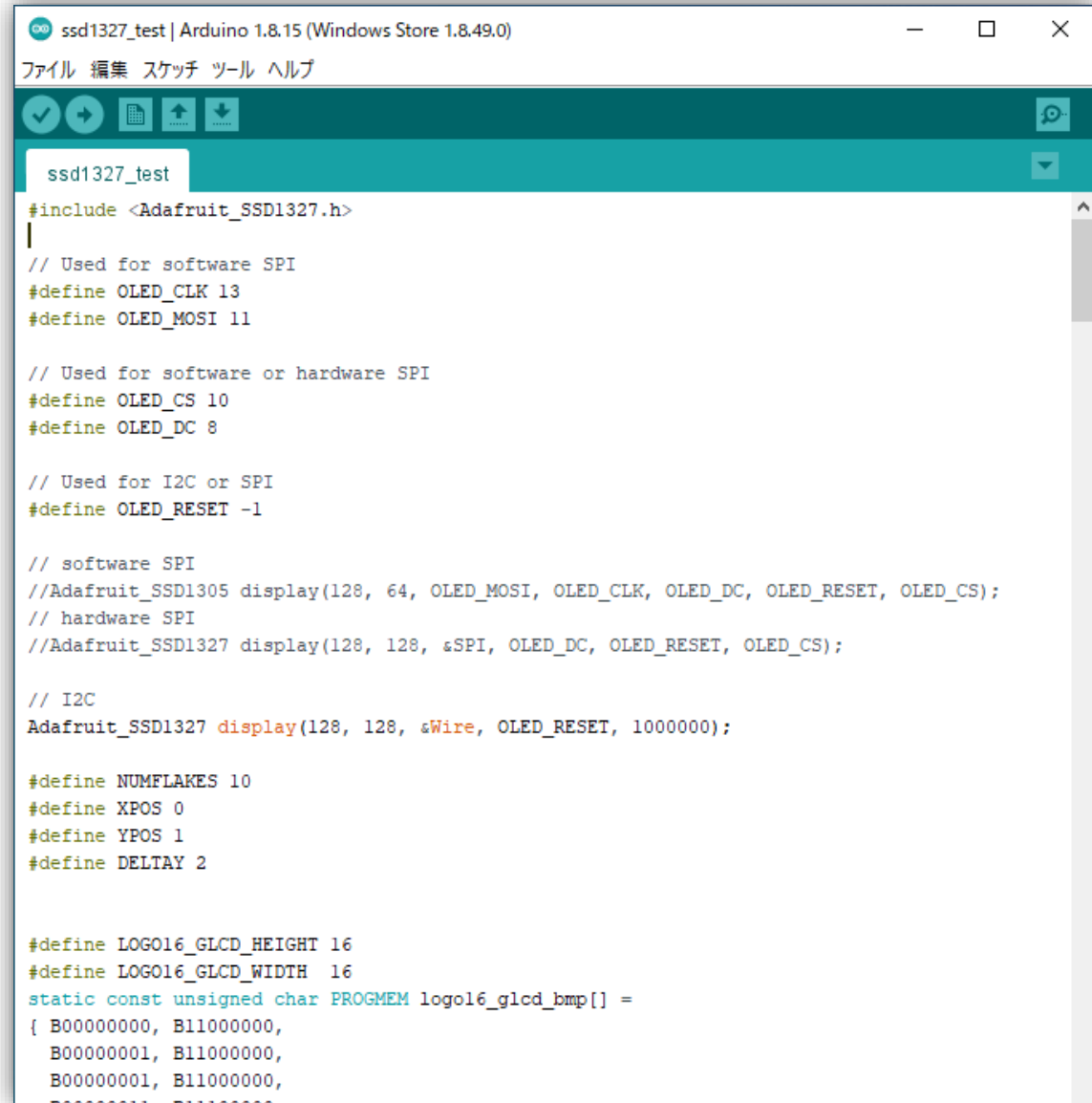


Change the display for standalone operating

- Open a sample code

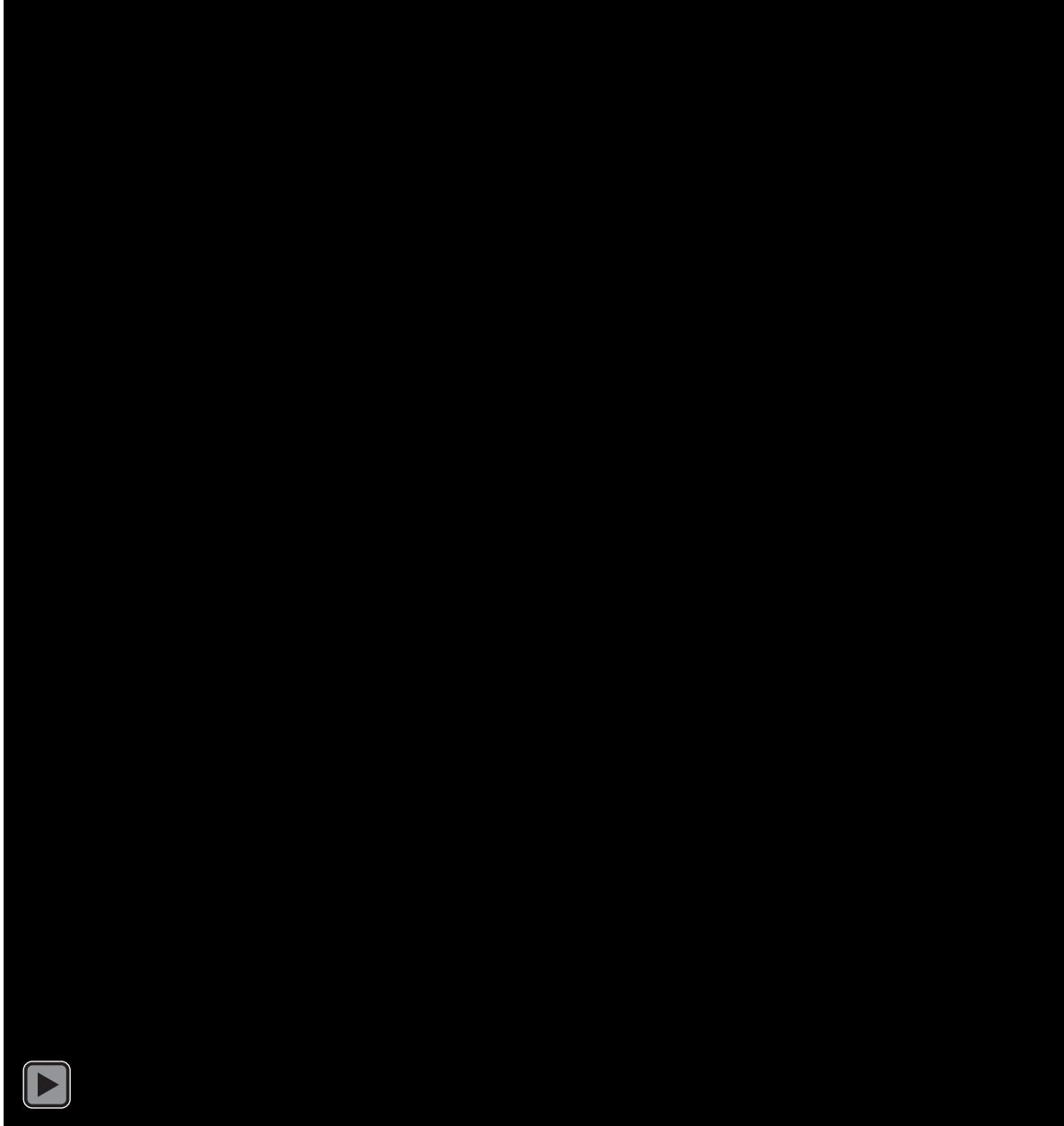


- Write it to your Arduino



Change the display for standalone operating

- Test run



Change the display for standalone operating

- Cut and paste

```
Hello
```

```
#include <Adafruit_SSD1327.h>

// I2C
#define OLED_RESET -1
Adafruit_SSD1327 display(128, 128, &Wire, OLED_RESET, 1000000);

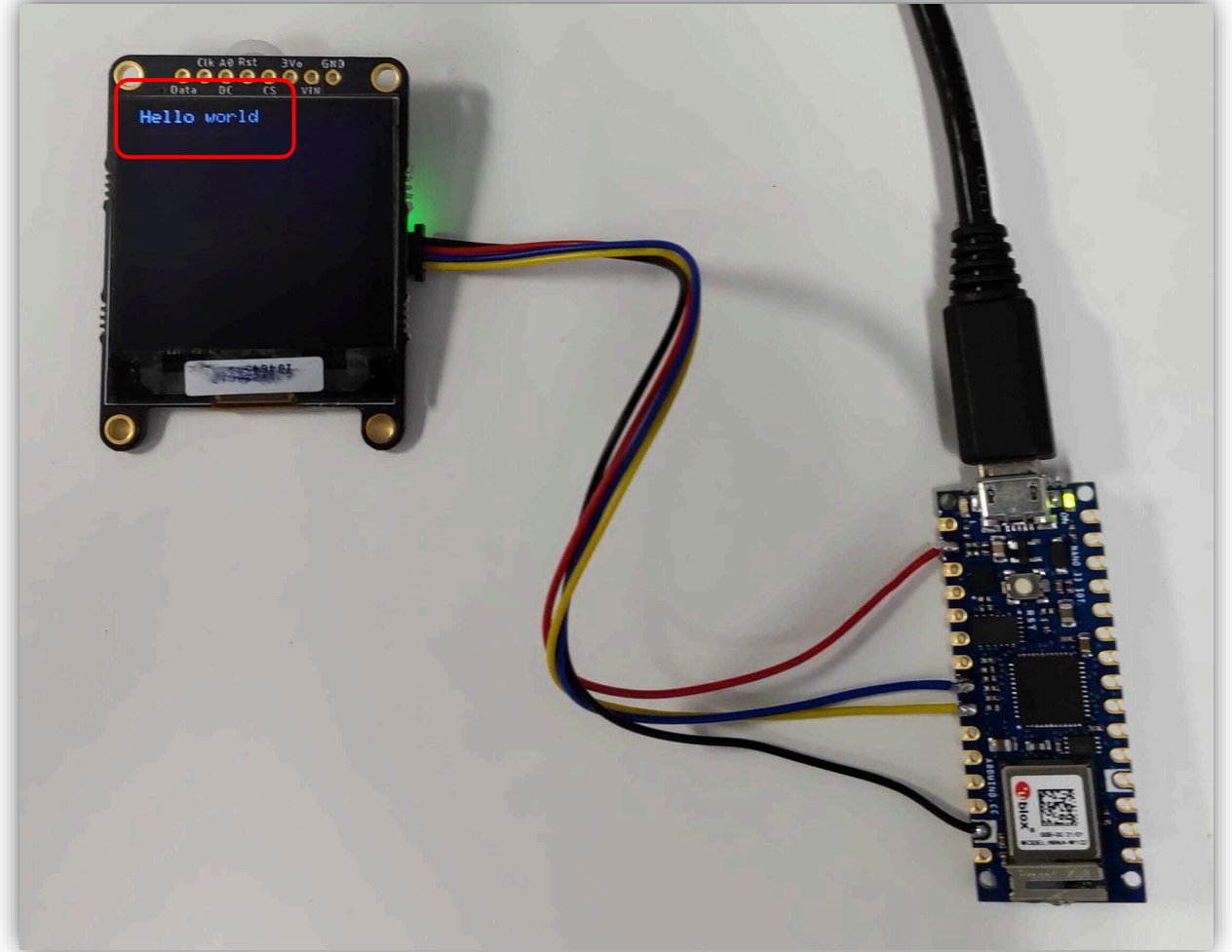
// the setup routine runs once when you press reset:
void setup() {
  display.begin(0x3D);
  display.clearDisplay();
  display.display();
}

// the loop routine runs over and over again forever:
void loop() {
  String s = "Hello world";

  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 0);

  display.println(s);
  display.display();

  delay(1000);      // delay in between reads for stability
}
```



Temperature measurement

'Adafruit MCP9808'

- Up to 8 on a single I2C bus with adjustable address pins
- 0.25° C typical precision over -40°C to 125°C range
- 0.0625° C resolution

~700 yen



address select pins



STEMMA QT / Qwiic type

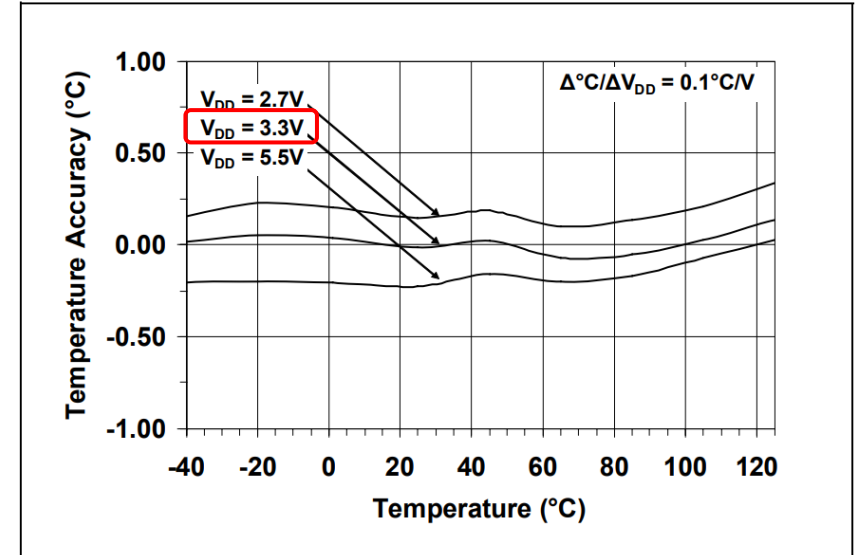
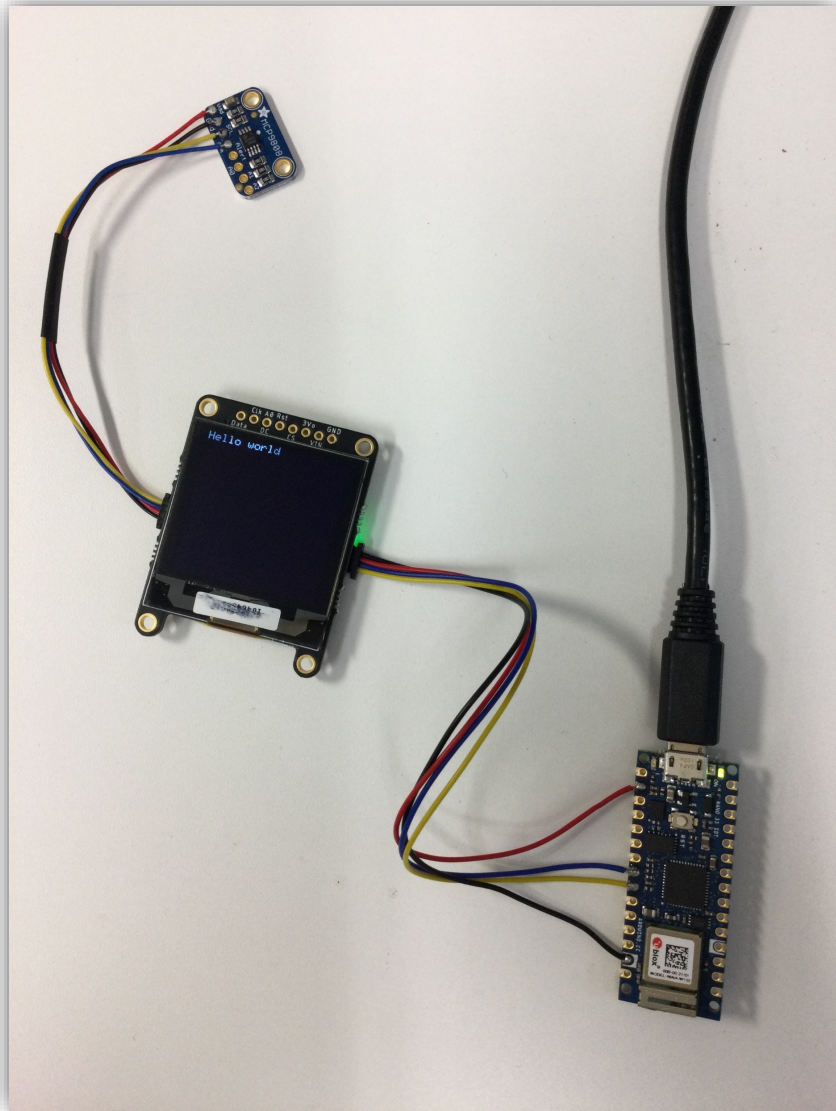


FIGURE 2-11: Temperature Accuracy vs Supply Voltage.

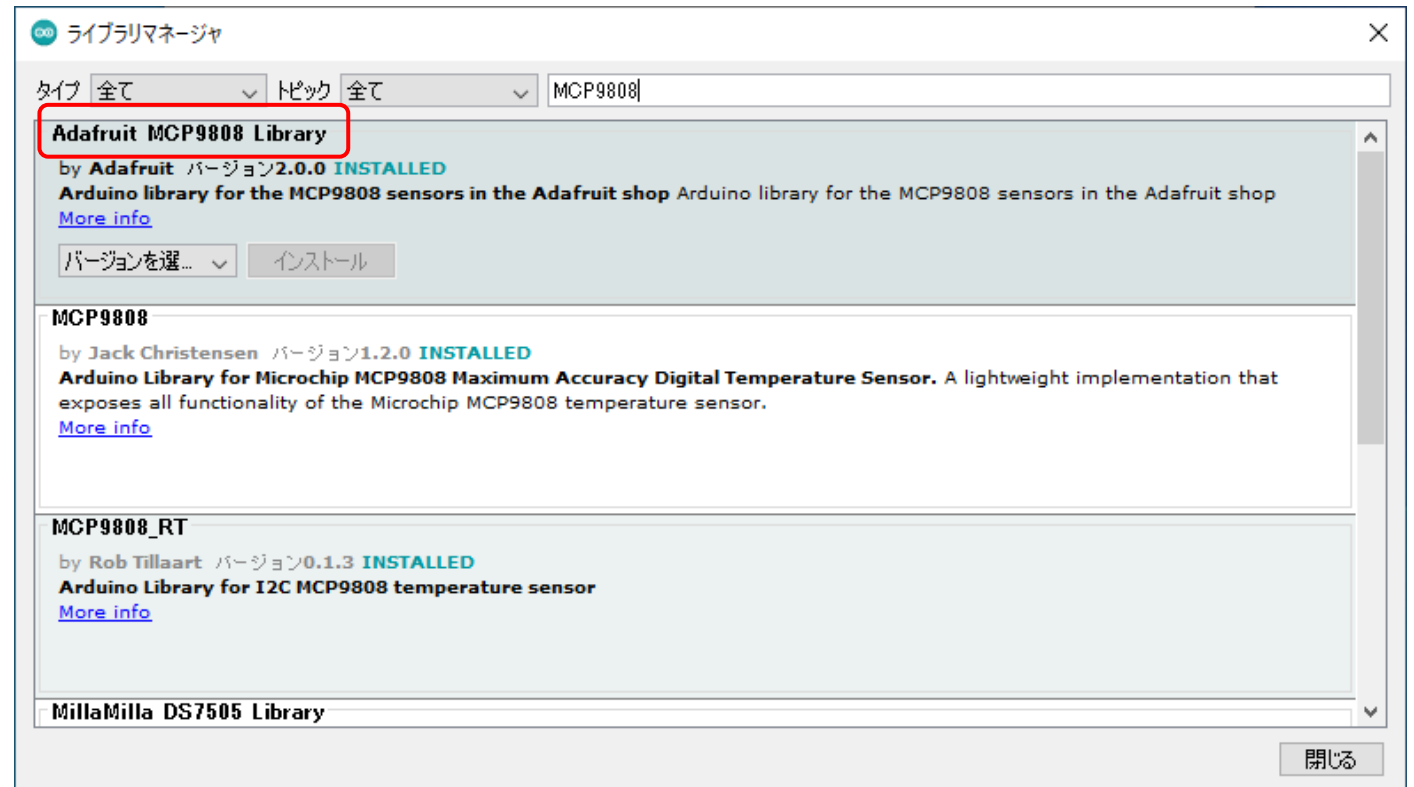
Parameters	Sym	Min	Typ	Max	Unit	Conditions
Temperature Sensor Accuracy						
$-20^{\circ}\text{C} < T_A \leq +100^{\circ}\text{C}$	T_{ACY}	-0.5	± 0.25	+0.5	$^{\circ}\text{C}$	$V_{\text{DD}} = 3.3\text{V}$
$-40^{\circ}\text{C} < T_A \leq +125^{\circ}\text{C}$	T_{ACY}	-1.0	± 0.25	+1.0	$^{\circ}\text{C}$	$V_{\text{DD}} = 3.3\text{V}$
Temperature Conversion Time						
0.5°C/bit	t_{CONV}	—	30	—	ms	33s/sec (typical)
0.25°C/bit		—	65	—	ms	15s/sec (typical)
0.125°C/bit		—	130	—	ms	7s/sec (typical)
0.0625°C/bit		—	250	—	ms	4s/sec (typical)

Temperature measurement

- I2C connection

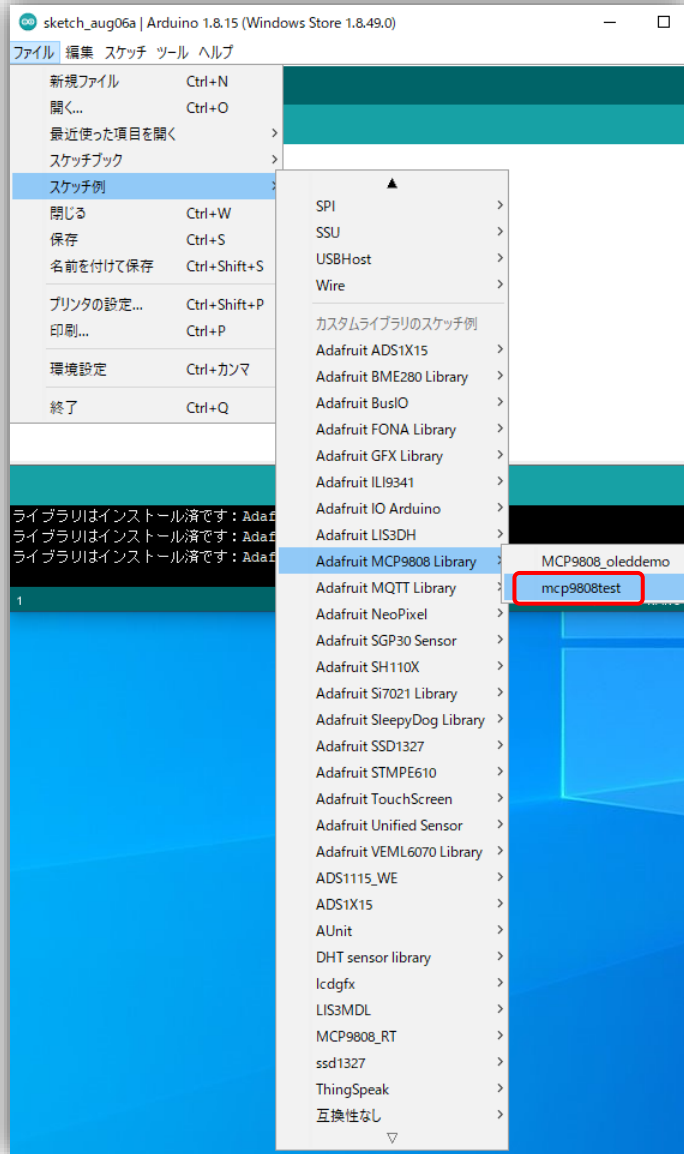


- Install the library of MCP9808



Temperature measurement

- Open a sample code

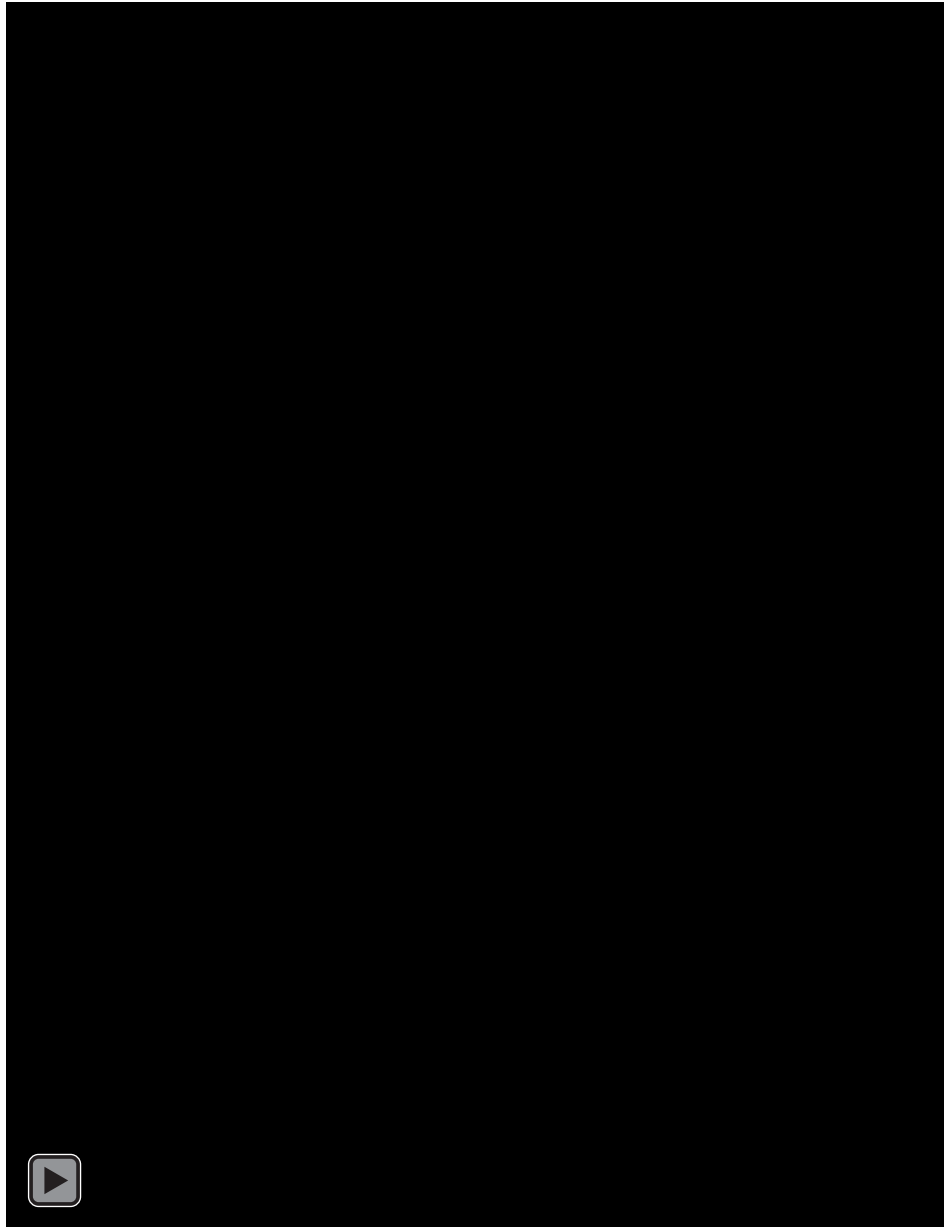


- Write it to your Arduino



Temperature measurement

- Test run



Temperature measurement

- Cut and paste

```
Hello_Temperature $
#include <Adafruit_SSD1327.h>
#include <Wire.h>
#include "Adafruit_MCP9808.h"

// I2C
#define OLED_RESET -1
Adafruit_SSD1327 display(128, 128, &Wire, OLED_RESET, 1000000);

// Create the MCP9808 temperature sensor object
Adafruit_MCP9808 tempsensor = Adafruit_MCP9808();

// the setup routine runs once when you press reset:
void setup() {
  display.begin(0x3D);
  display.clearDisplay();
  display.display();

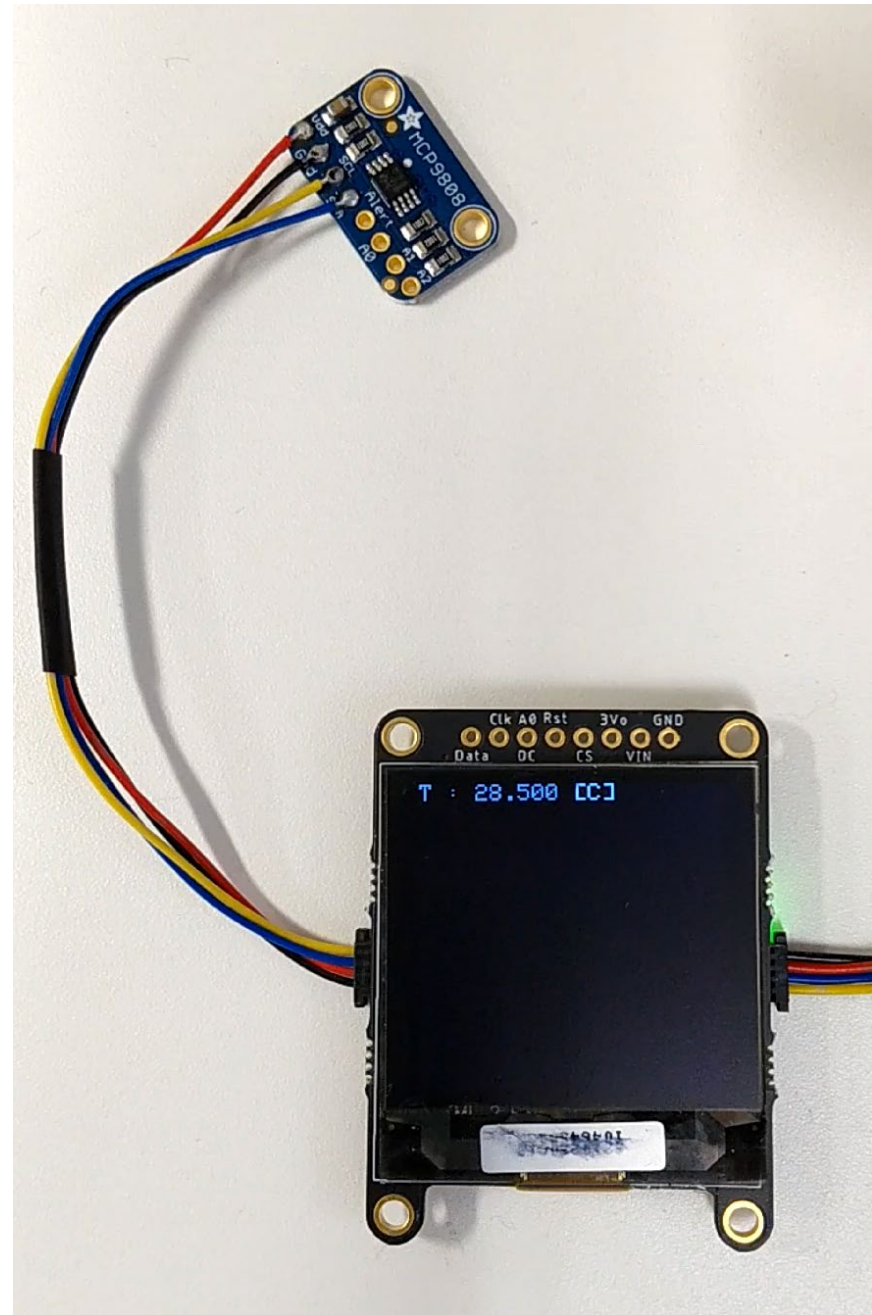
  tempsensor.begin(0x18);
  tempsensor.setResolution(3); // sets the resolution mode of reading, the modes are
  // Mode Resolution SampleTime
  // 0 0.5°C 30 ms
  // 1 0.25°C 65 ms
  // 2 0.125°C 130 ms
  // 3 0.0625°C 250 ms
  tempsensor.wake();
}

// the loop routine runs over and over again forever:
void loop() {
  float c = tempsensor.readTempC();

  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 0);

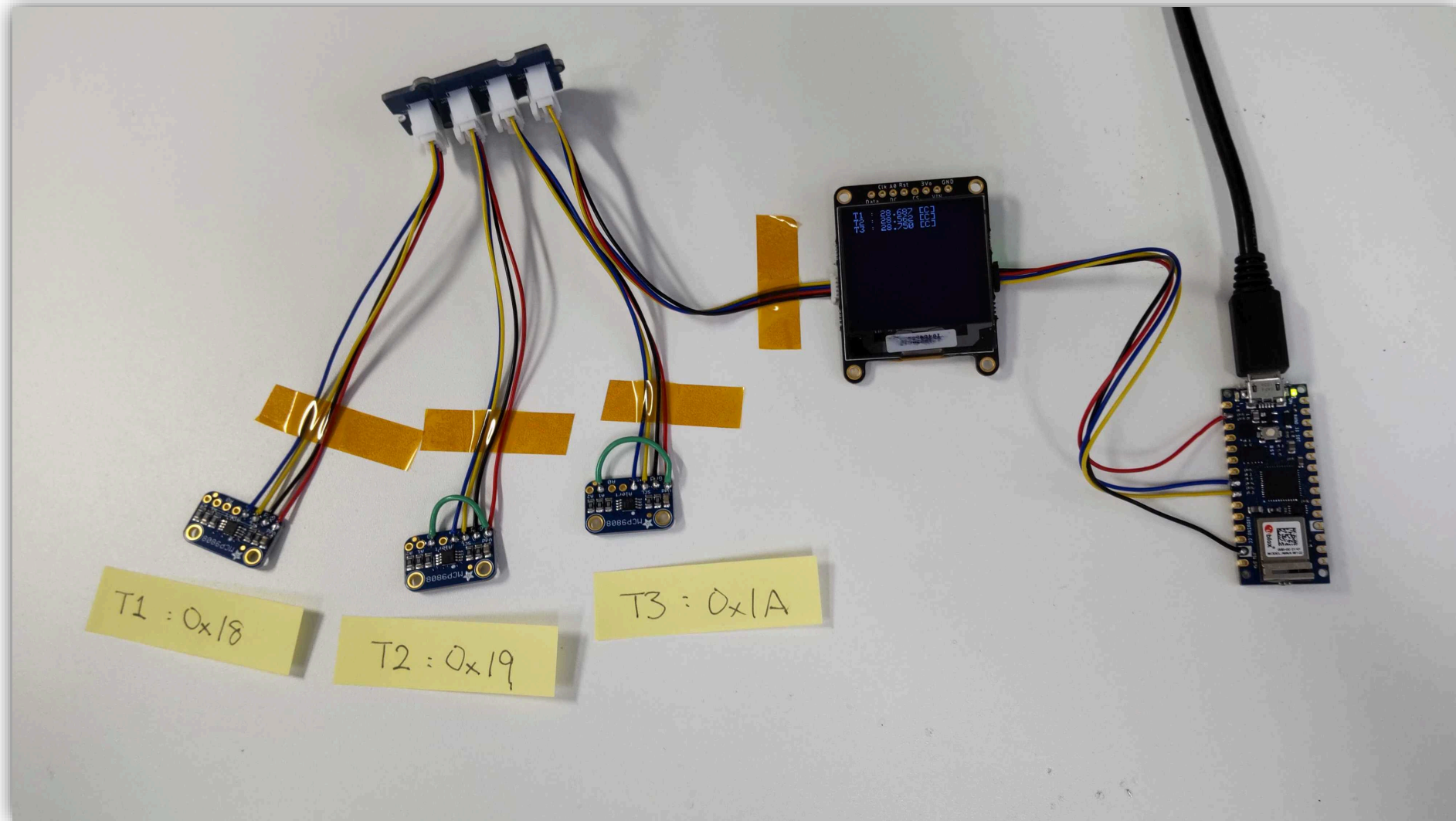
  display.print("T : "); display.print(c, 3); display.println(" [C]");
  display.display();

  delay(500); // delay in between reads
}
```



Temperature measurement

- Multiple sensors



Temperature measurement

- Multiple sensors

```
#include <Adafruit_SSD1327.h>
#include <Wire.h>
#include "Adafruit_MCP9808.h"

// I2C
#define OLED_RESET -1
Adafruit_SSD1327 display(128, 128, &Wire, OLED_RESET, 1000000);

// Create the MCP9808 temperature sensor object
Adafruit_MCP9808 tempsensor1 = Adafruit_MCP9808();
Adafruit_MCP9808 tempsensor2 = Adafruit_MCP9808();
Adafruit_MCP9808 tempsensor3 = Adafruit_MCP9808();

// the setup routine runs once when you press reset:
void setup() {
  display.begin(0x3D);
  display.clearDisplay();
  display.display();

  tempsensor1.begin(0x18);
  tempsensor2.begin(0x19);
  tempsensor3.begin(0x1A);

  tempsensor1.setResolution(3); // sets the resolution mode of reading, the modes are defined in t
  tempsensor2.setResolution(3); // sets the resolution mode of reading, the modes are defined in t
  tempsensor3.setResolution(3); // sets the resolution mode of reading, the modes are defined in t
  // Mode Resolution SampleTime
  // 0 0.5°C 30 ms
  // 1 0.25°C 65 ms
  // 2 0.125°C 130 ms
  // 3 0.0625°C 250 ms
  tempsensor1.wake();
  tempsensor2.wake();
  tempsensor3.wake();
}

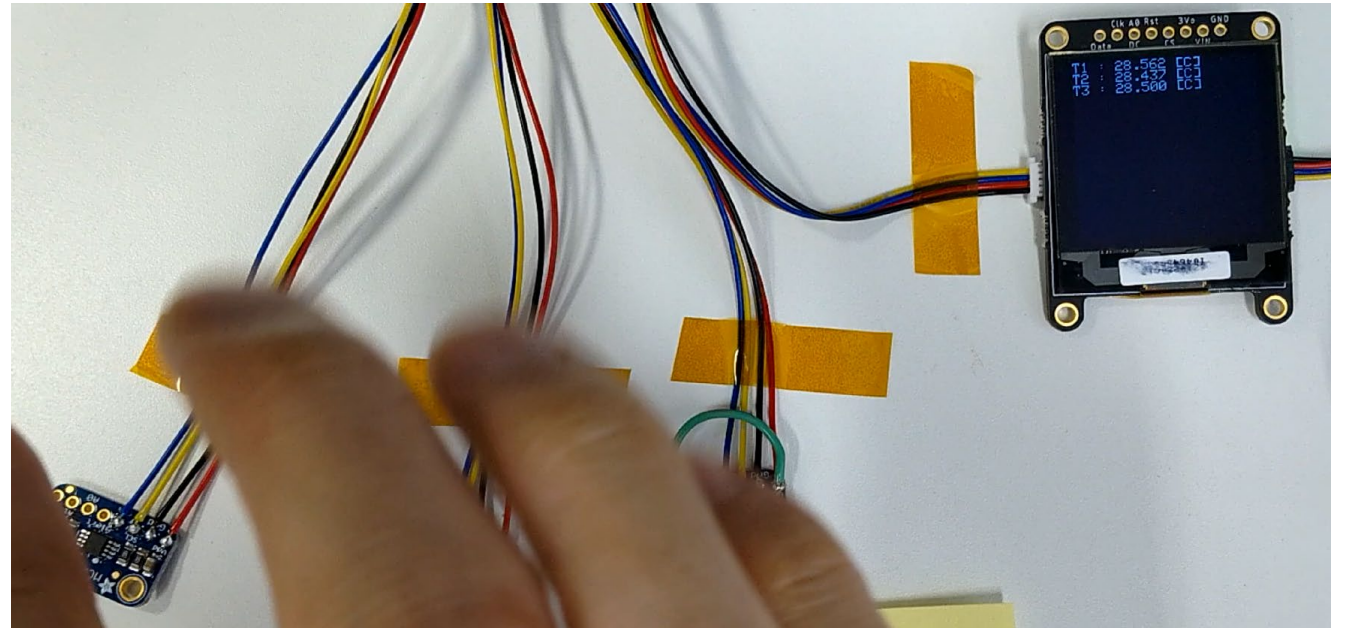
// the loop routine runs over and over again forever:
void loop() {

  float c1 = tempsensor1.readTempC();
  float c2 = tempsensor2.readTempC();
  float c3 = tempsensor3.readTempC();

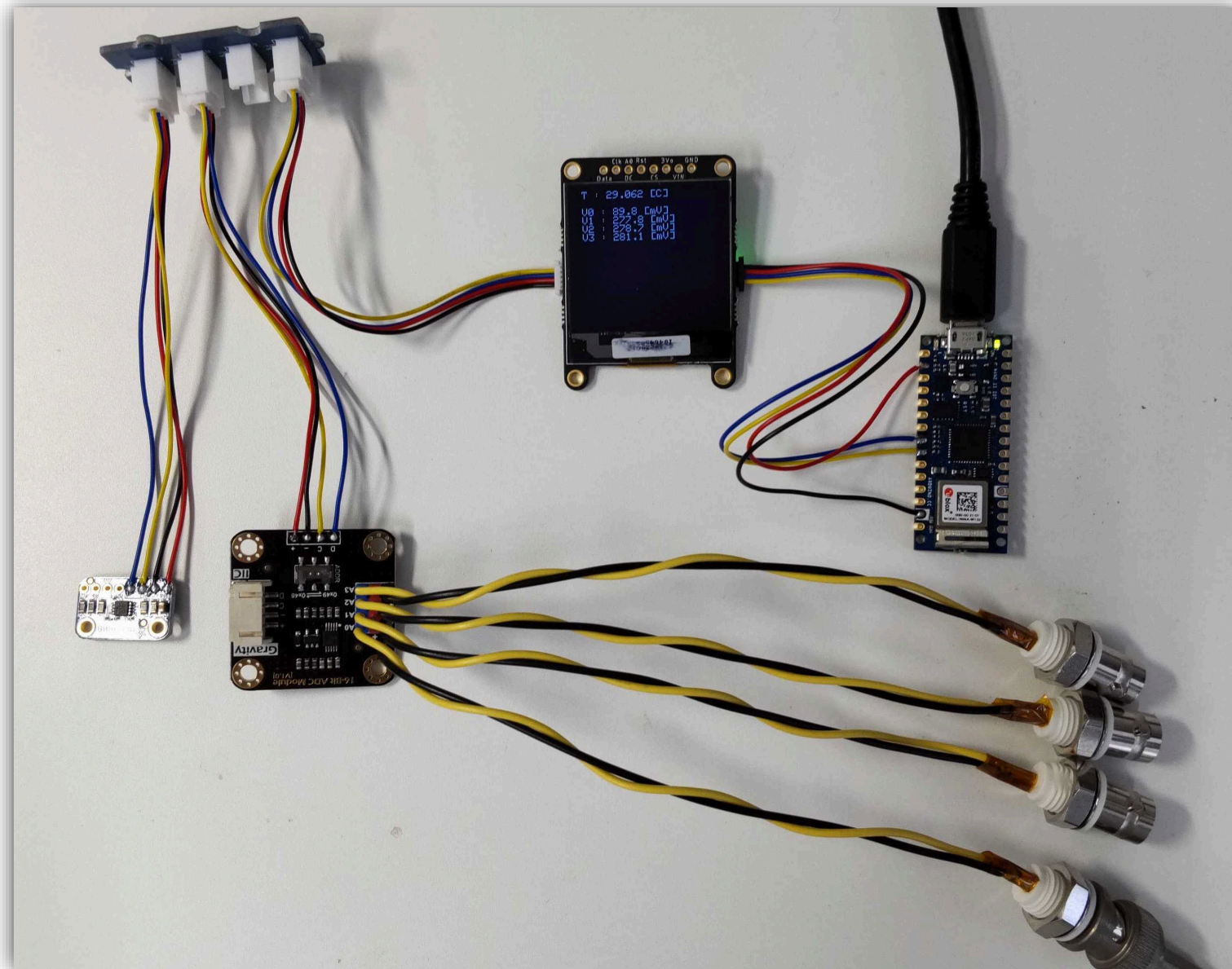
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 0);

  display.print("T1 : "); display.print(c1, 3); display.println(" [C]");
  display.print("T2 : "); display.print(c2, 3); display.println(" [C]");
  display.print("T3 : "); display.print(c3, 3); display.println(" [C]");
  display.display();

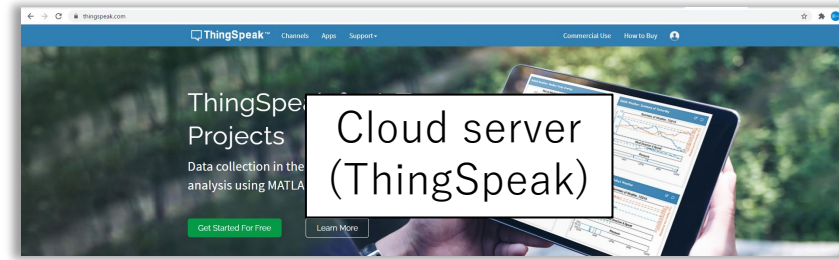
  delay(500); // delay in between reads
}
```



We can add an ADC device as well



Today's goal

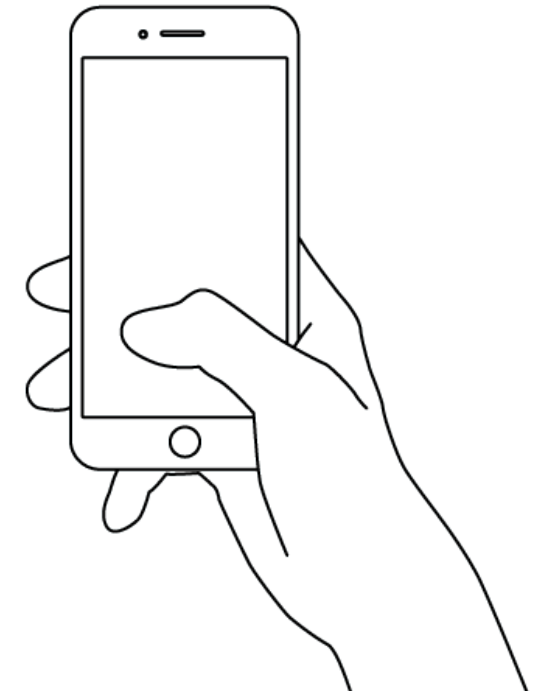
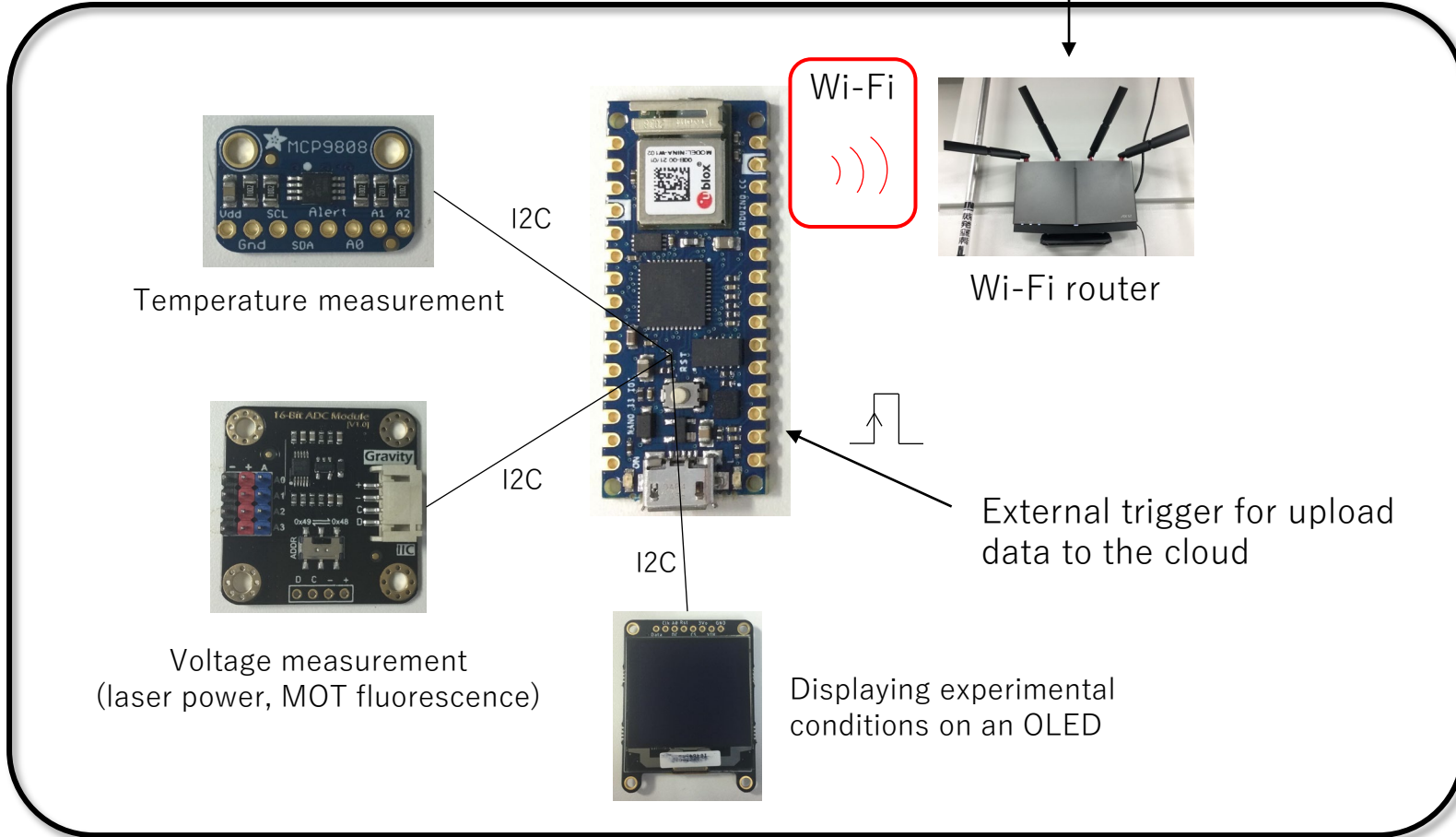


Internet

Inside your laboratory

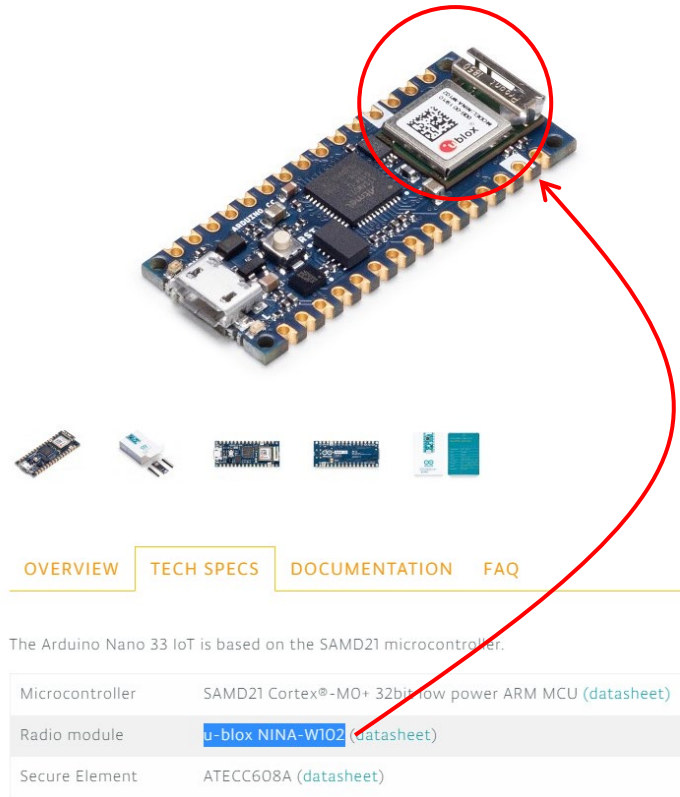
Internet

Outside your laboratory



Wi-Fi connection

- Install the library of WiFiNINA



ライブラリマネージャ

タイプ 全て トピック 全て wifinina

WiFiNINA

by Arduino バージョン1.8.13 **INSTALLED**

Enables network connection (local and Internet) with the Arduino MKR WiFi 1010, Arduino MKR VIDOR 4000, Arduino UNO WiFi Rev.2 and Nano 33 IoT. With this library you can instantiate Servers, Clients and send/receive UDP packets through WiFi. The board can connect either to open or encrypted networks (WEP, WPA). The IP address can be assigned statically or through a DHCP. The library can also manage DNS.

[More info](#)

バージョンを選... インストール

ArduinoOTA

by Juraj Andrassy

Upload sketch over network to Arduino board with WiFi or Ethernet libraries Based on WiFi101OTA library. Uploads over Ethernet, UIPEthernet, WiFi101, WiFiNina, WiFiLink, WiFi, WiFiEspAT to SAMD, nRF5, esp8266, esp32 and to ATmega with more than 64 kB flash memory.

[More info](#)

ArtnetWifi

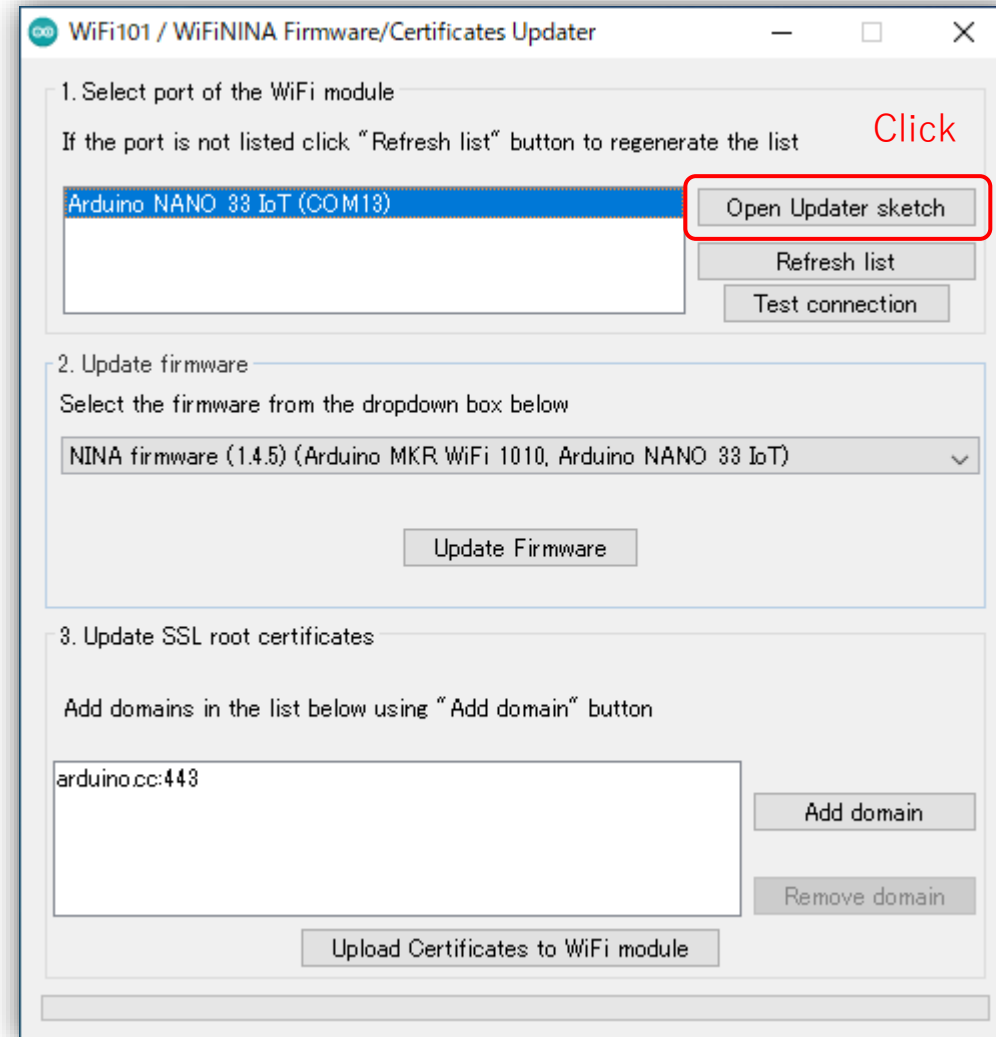
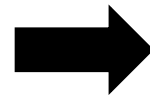
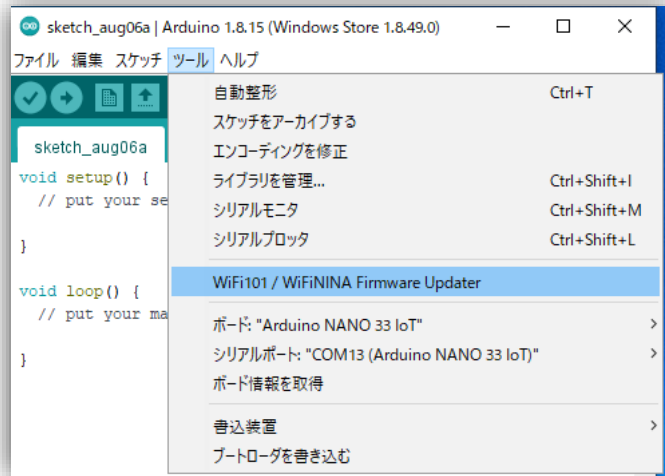
by Stephan Ruloff

ArtNet with the ESP8266, ESP32 and more. Send and receive Art-Net frames using WiFi. Tested on ESP8266, ESP32, WiFi101 and WiFiNINA devices.

閉じる

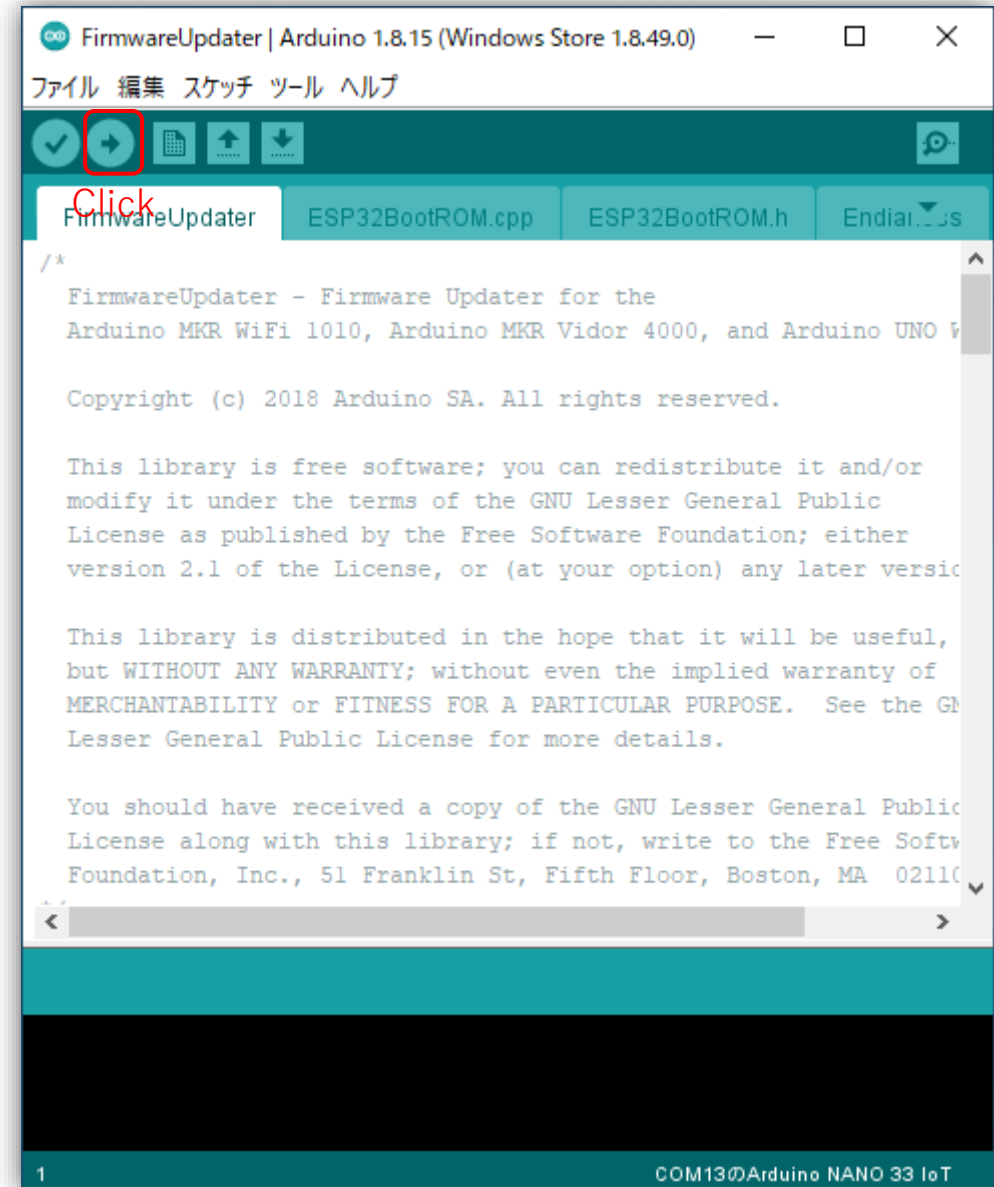
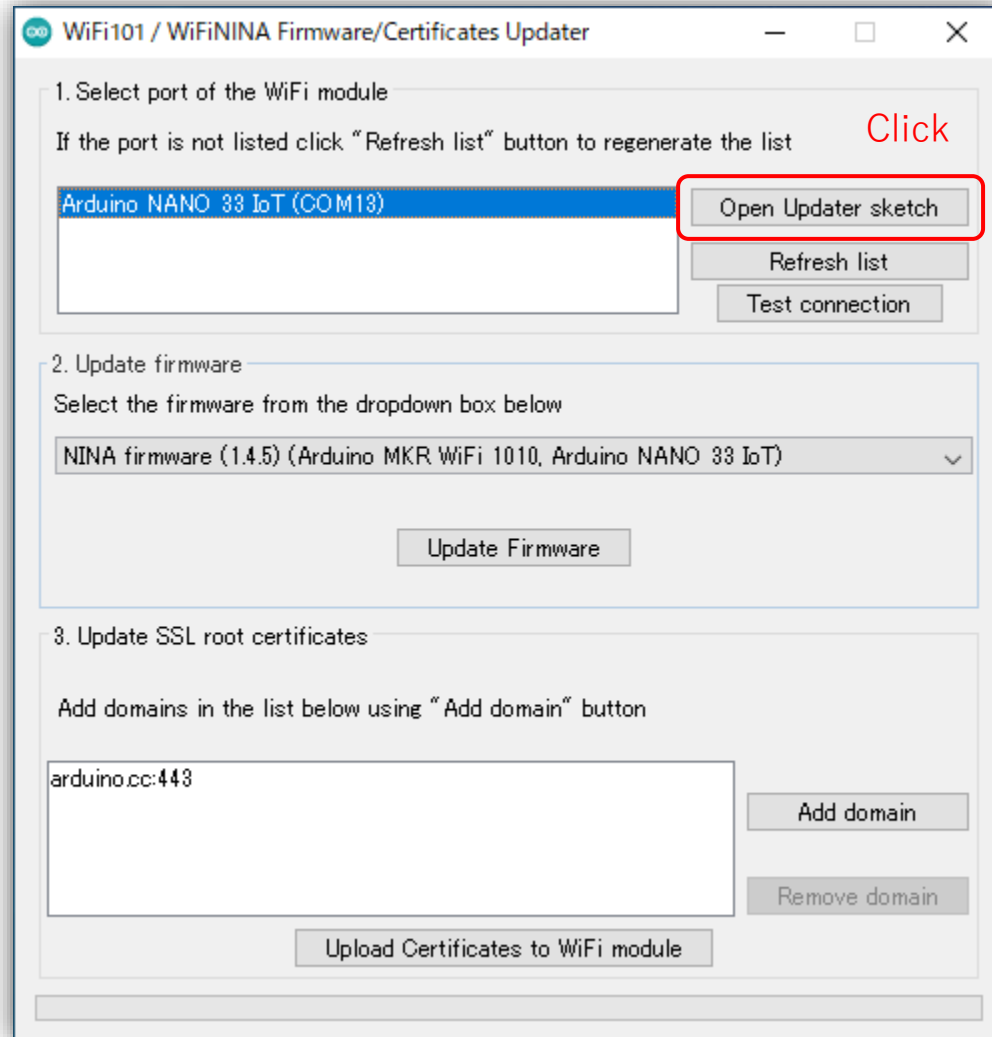
Wi-Fi connection

- Update the firmware



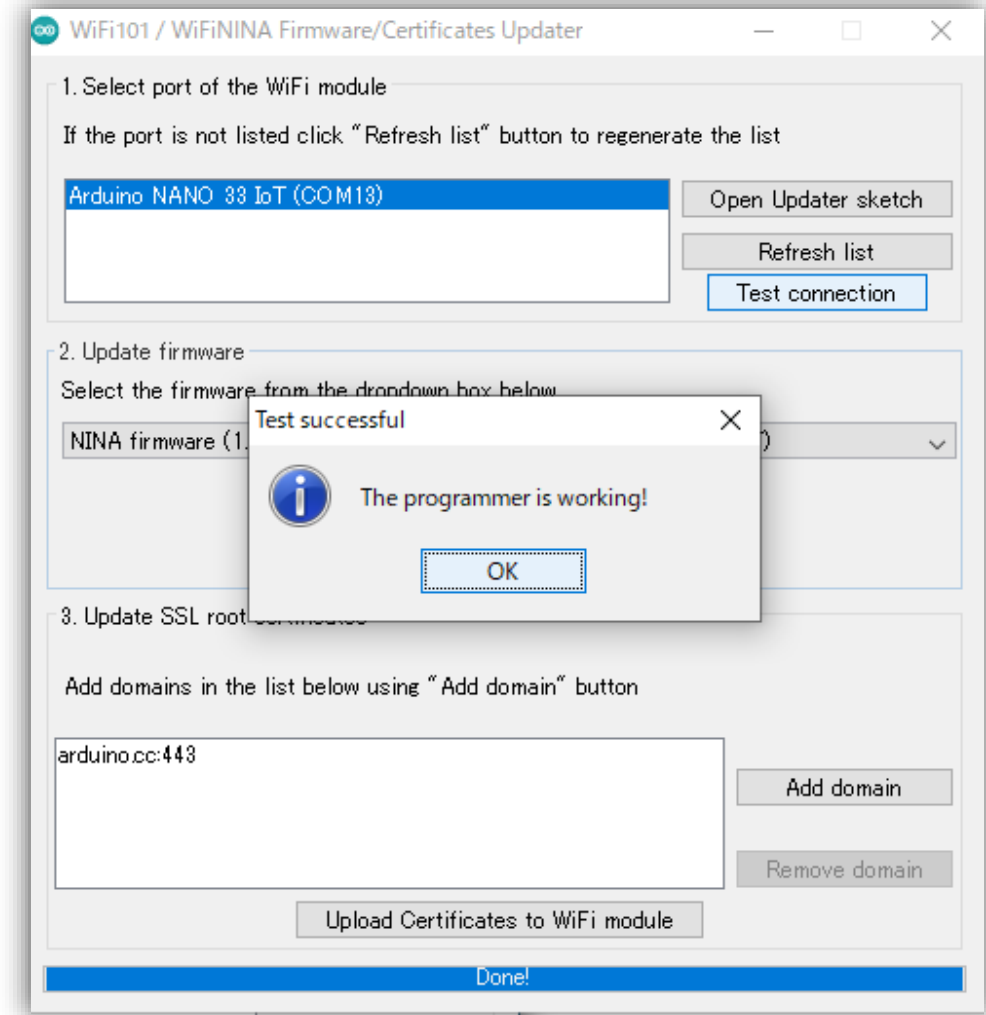
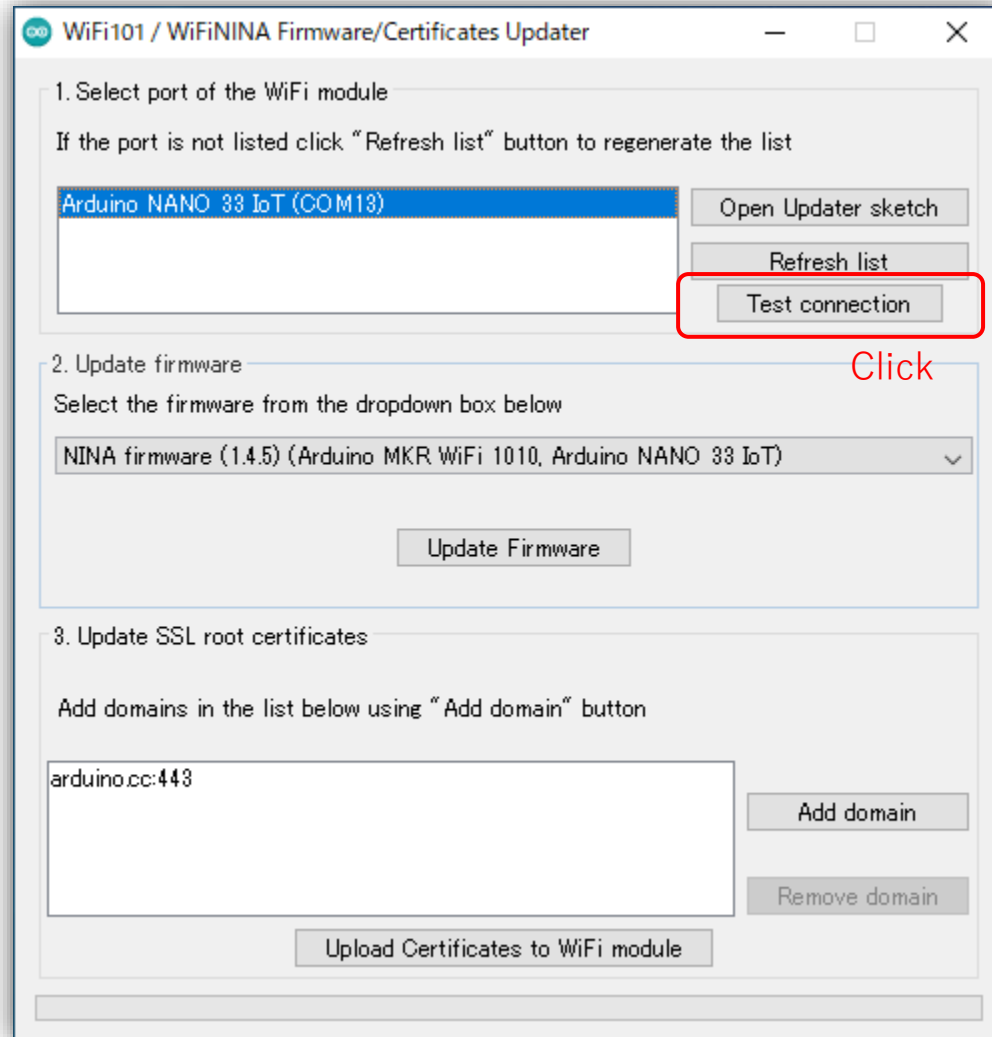
Wi-Fi connection

- Update the firmware



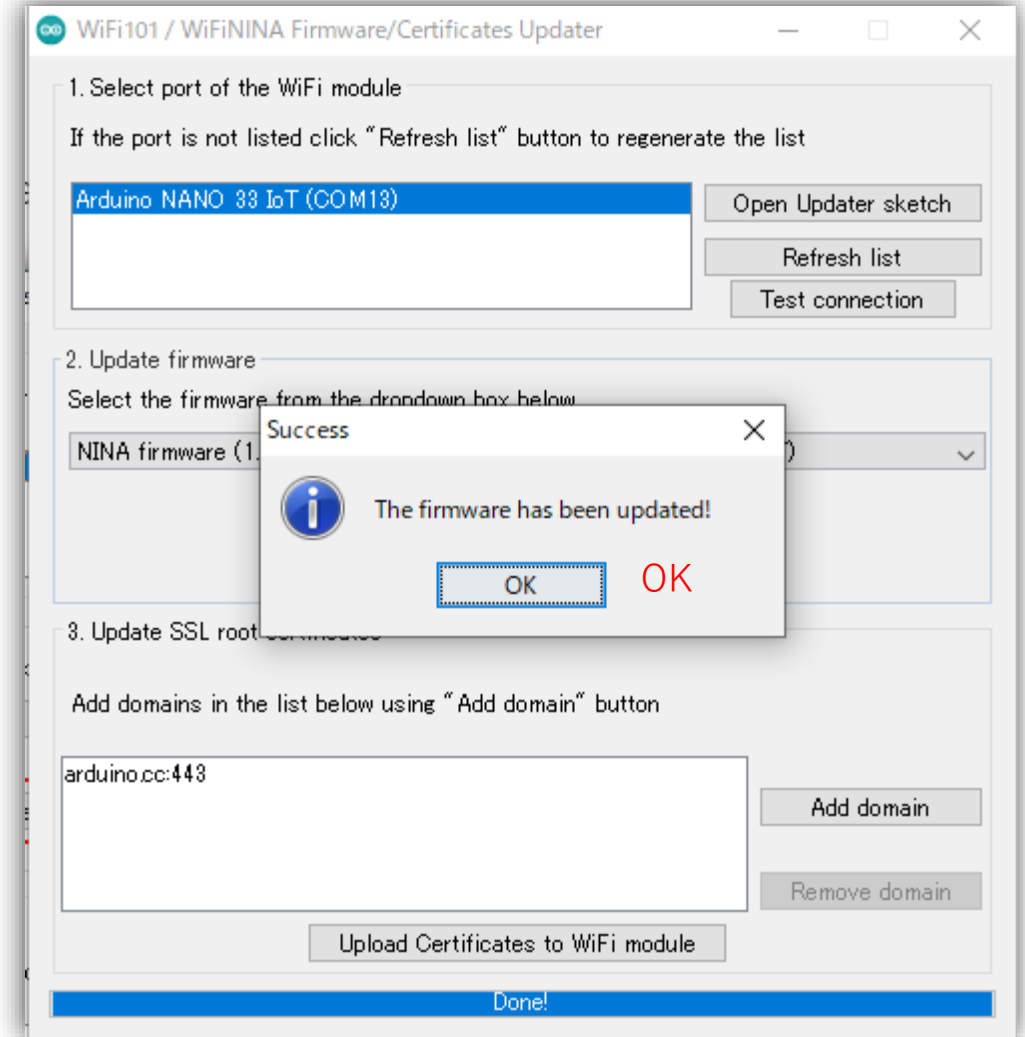
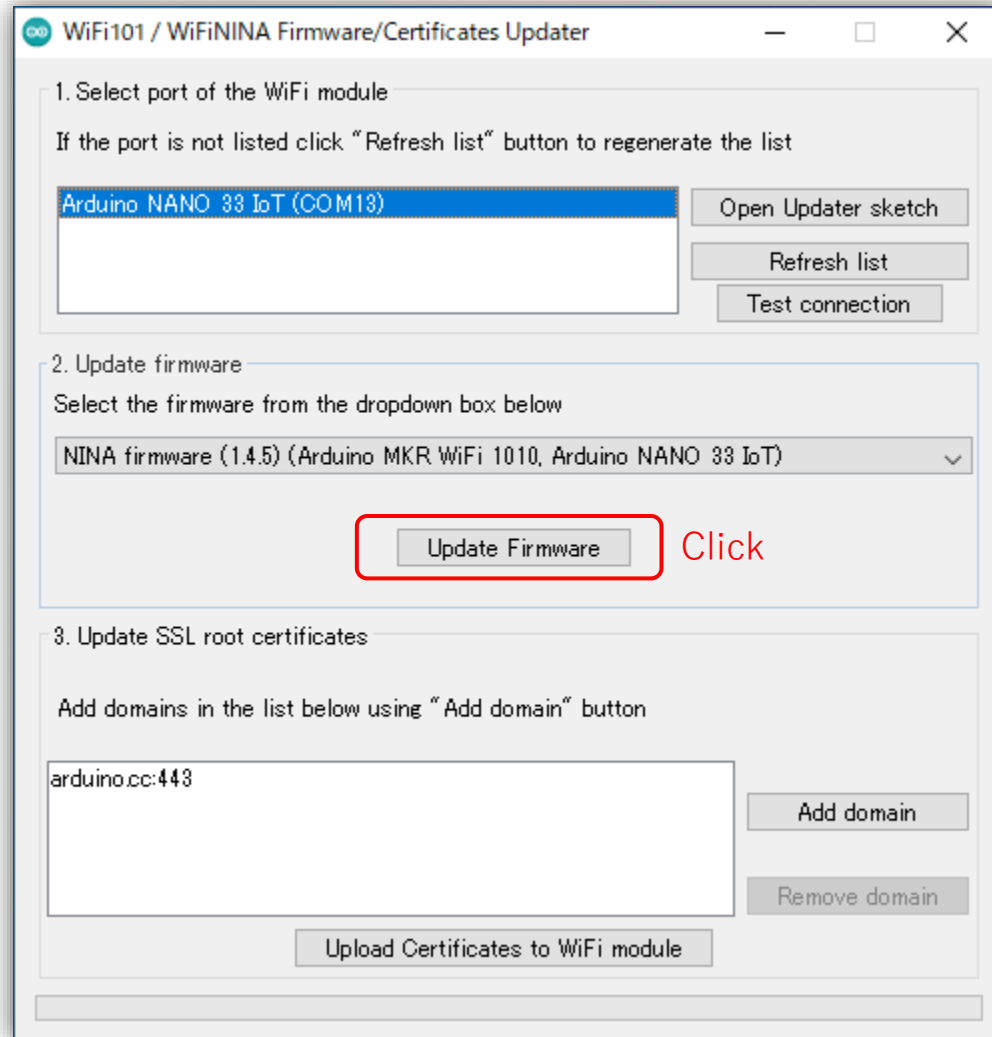
Wi-Fi connection

- Update the firmware



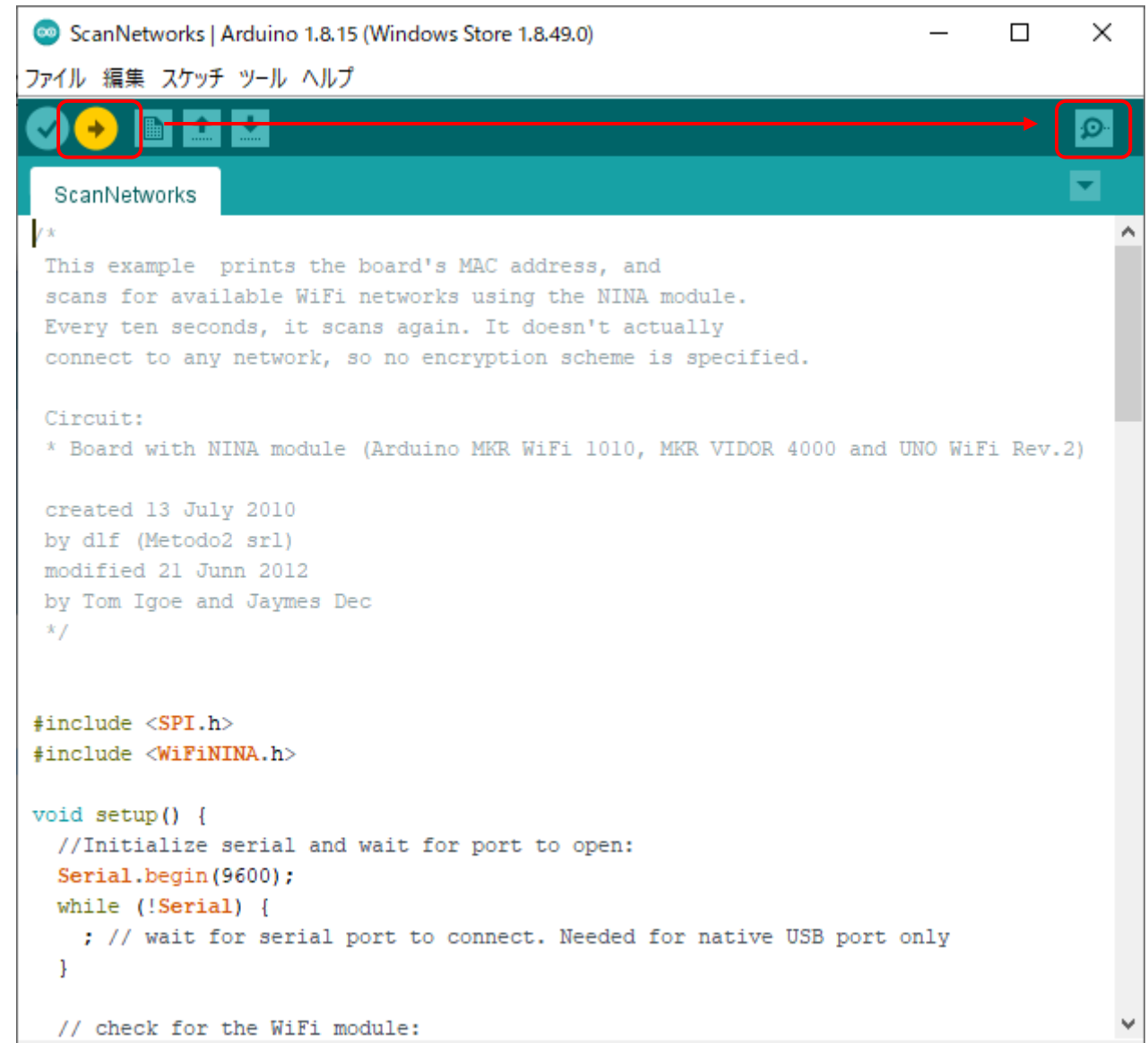
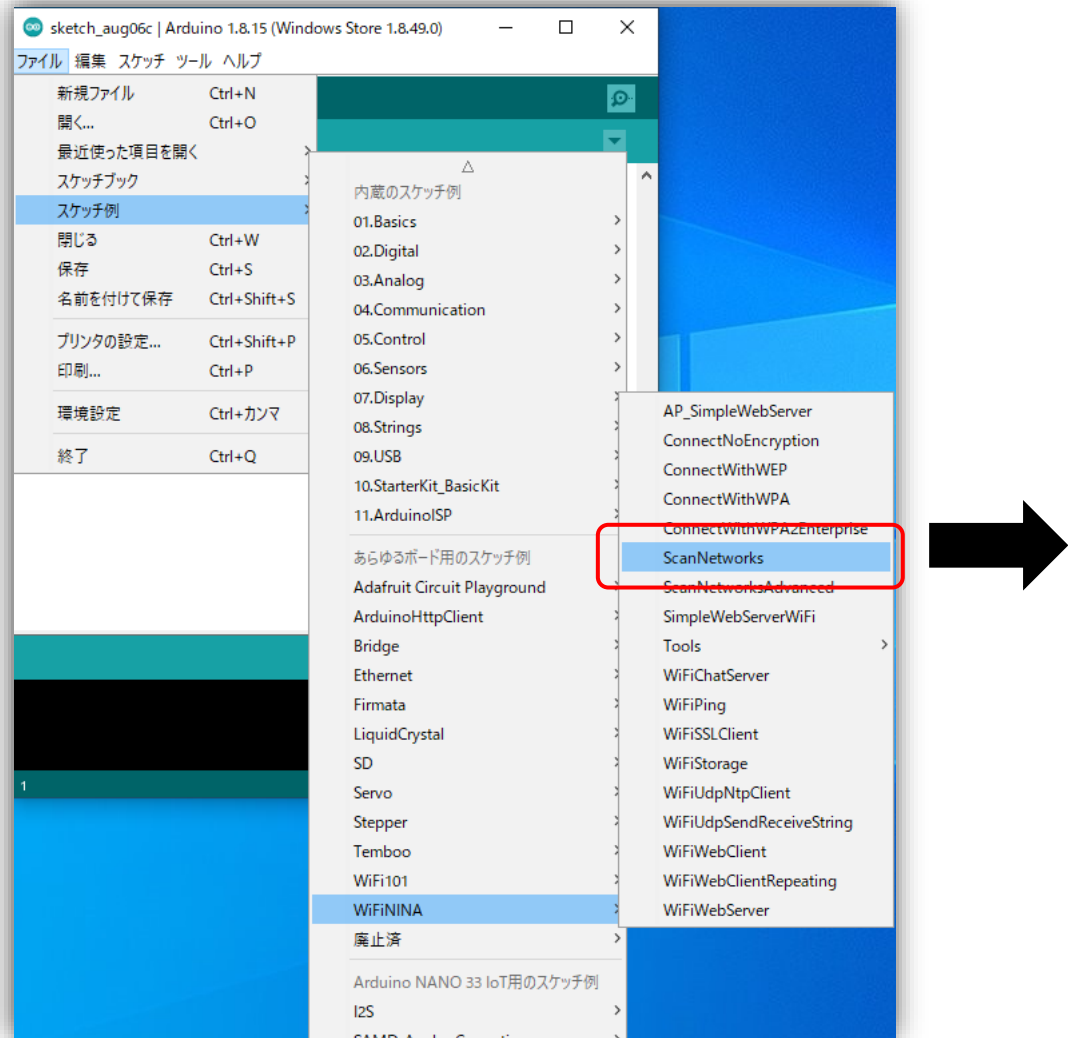
Wi-Fi connection

- Update the firmware

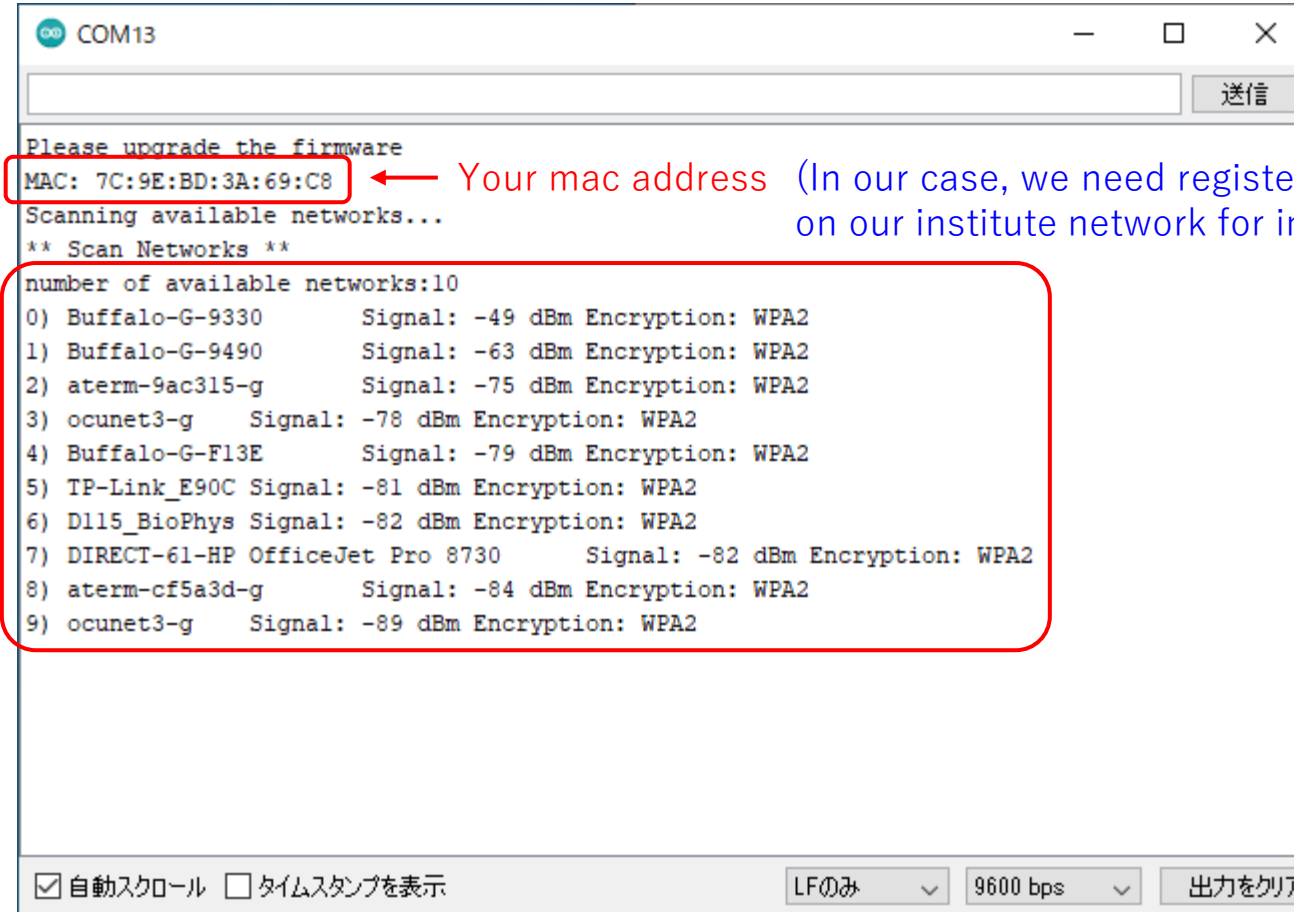


Wi-Fi connection : How to check the mac address

- Open a sample code



Wi-Fi connection : How to check the mac address

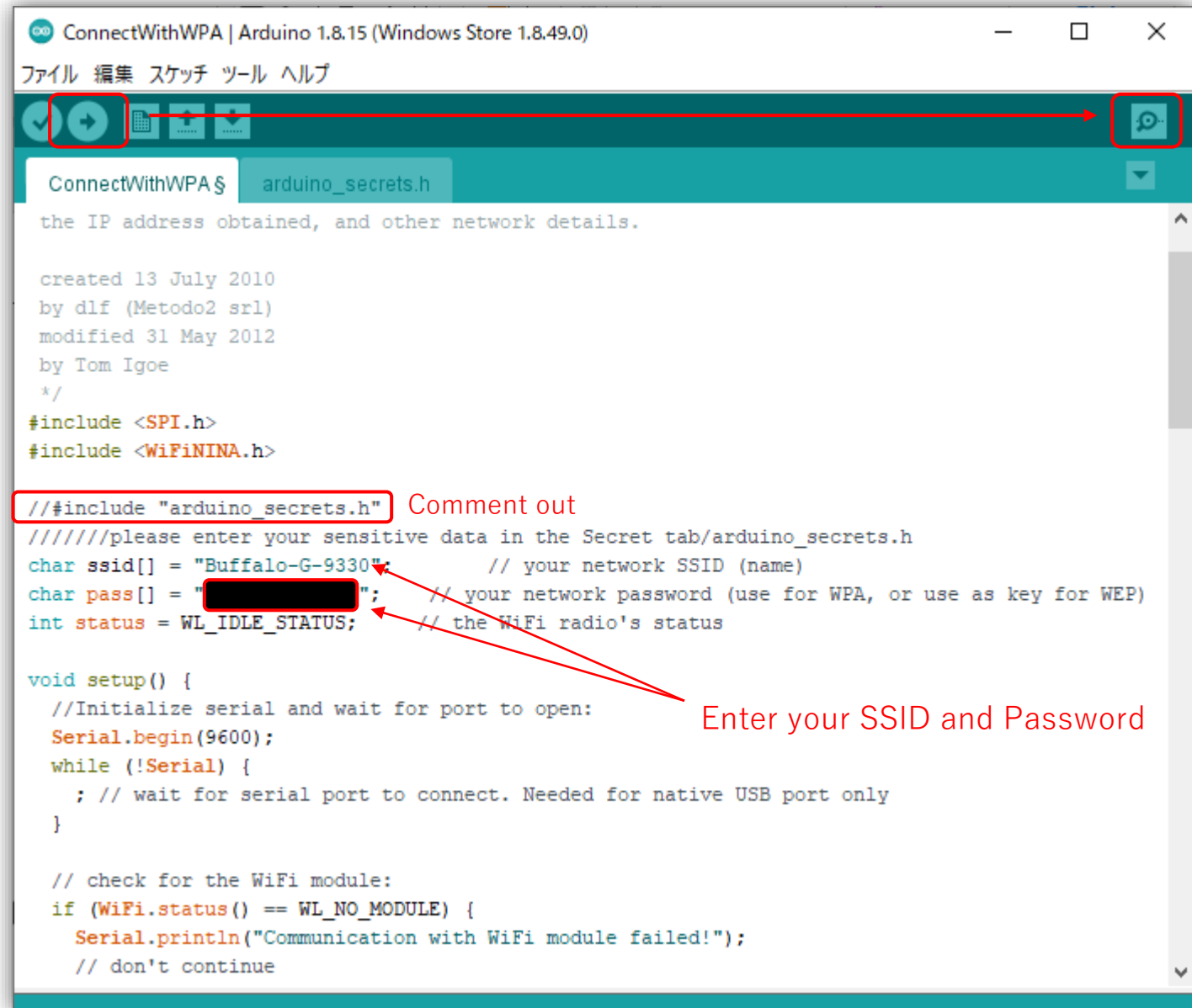
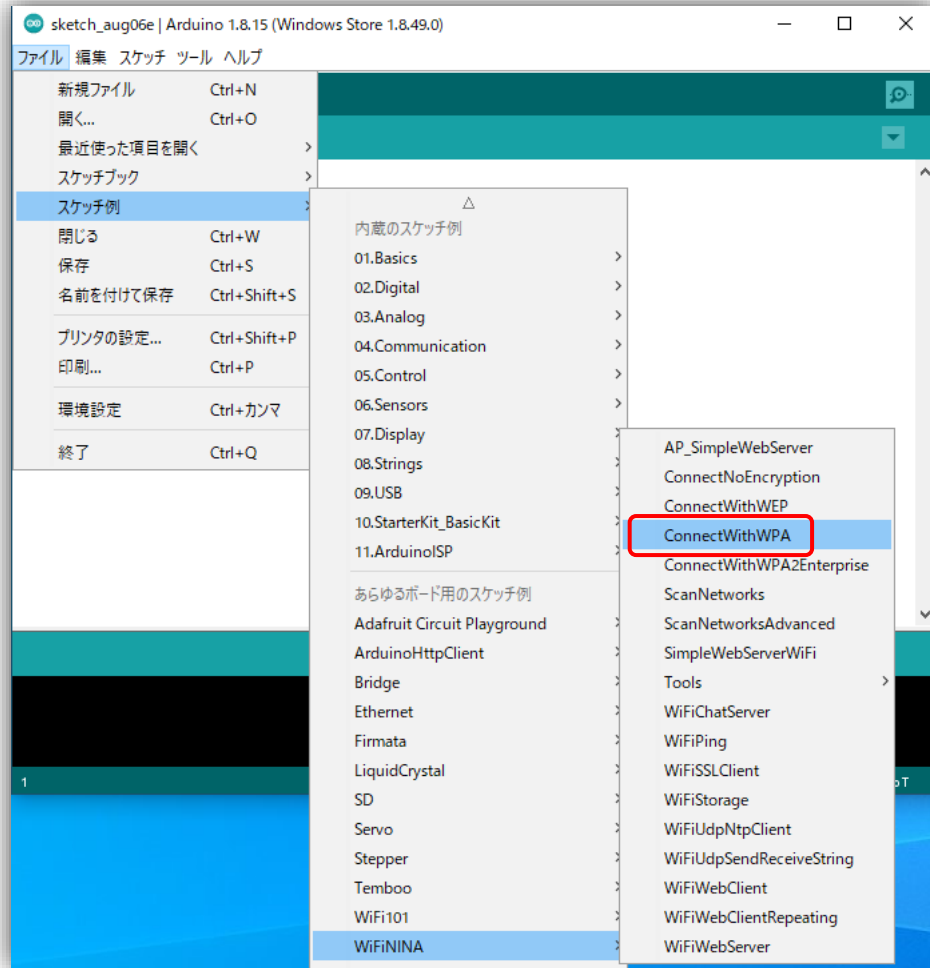


```
COM13
Please upgrade the firmware
MAC: 7C:9E:BD:3A:69:C8
Scanning available networks...
** Scan Networks **
number of available networks:10
0) Buffalo-G-9330      Signal: -49 dBm Encryption: WPA2
1) Buffalo-G-9490      Signal: -63 dBm Encryption: WPA2
2) aterm-9ac315-g      Signal: -75 dBm Encryption: WPA2
3) ocunet3-g          Signal: -78 dBm Encryption: WPA2
4) Buffalo-G-F13E      Signal: -79 dBm Encryption: WPA2
5) TP-Link_E90C        Signal: -81 dBm Encryption: WPA2
6) D115_BioPhys        Signal: -82 dBm Encryption: WPA2
7) DIRECT-6l-HP OfficeJet Pro 8730      Signal: -82 dBm Encryption: WPA2
8) aterm-cf5a3d-g      Signal: -84 dBm Encryption: WPA2
9) ocunet3-g          Signal: -89 dBm Encryption: WPA2
```

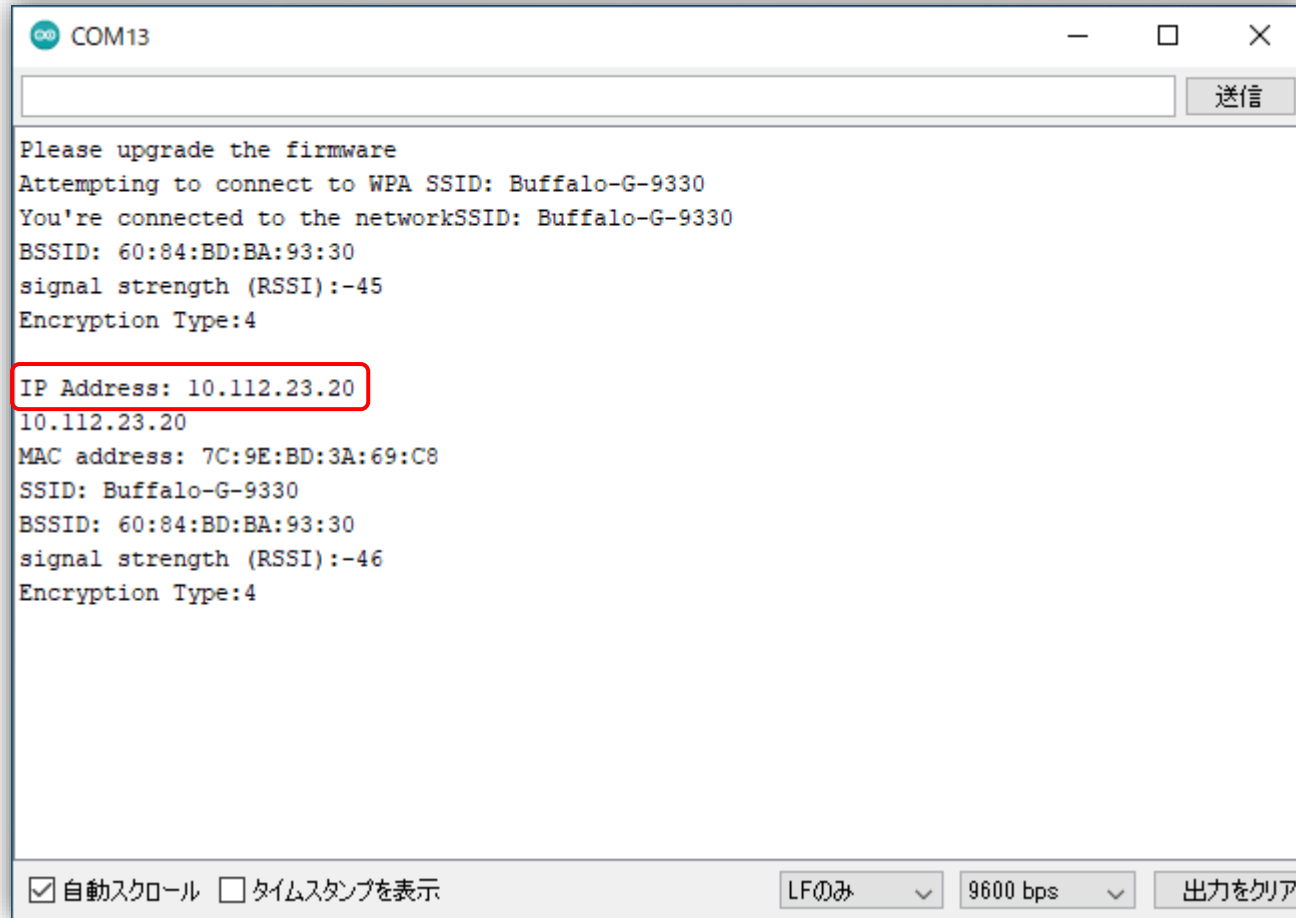
← Your mac address (In our case, we need register the mac address on our institute network for internet connection)

Wi-Fi connection : How to check the mac address

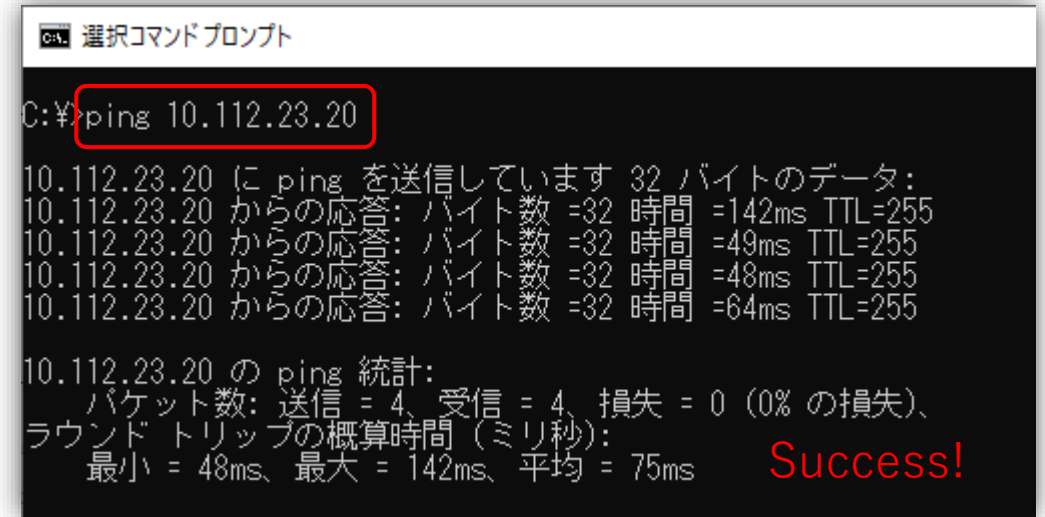
- Connect to the internet



Wi-Fi connection : How to confirm the connection



```
COM13
Please upgrade the firmware
Attempting to connect to WPA SSID: Buffalo-G-9330
You're connected to the networkSSID: Buffalo-G-9330
BSSID: 60:84:BD:BA:93:30
signal strength (RSSI):-45
Encryption Type:4
IP Address: 10.112.23.20
10.112.23.20
MAC address: 7C:9E:BD:3A:69:C8
SSID: Buffalo-G-9330
BSSID: 60:84:BD:BA:93:30
signal strength (RSSI):-46
Encryption Type:4
 自動スクロール  タイムスタンプを表示
LFのみ 9600 bps 出力をクリア
```

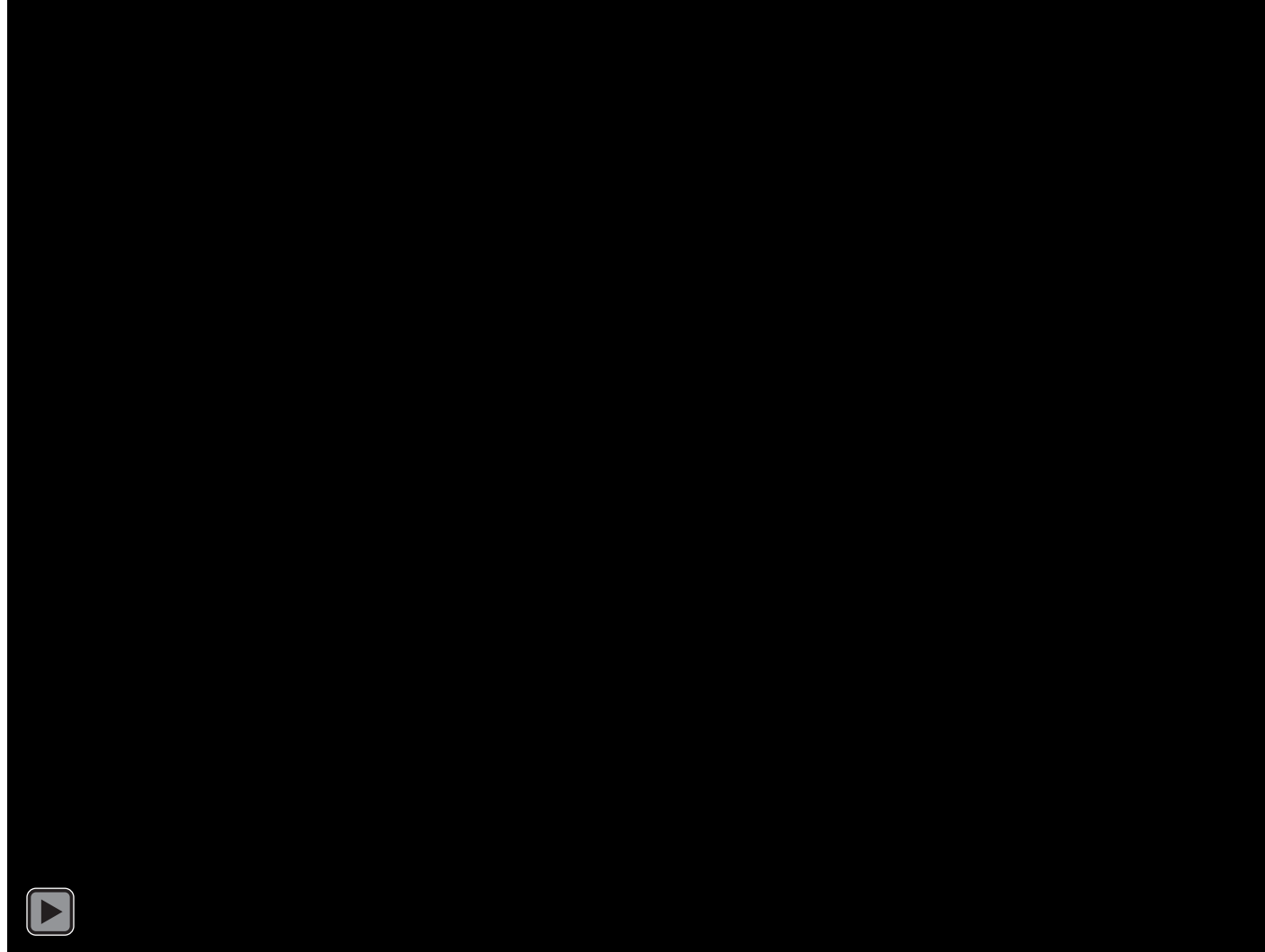


```
選択コマンドプロンプト
C:\>ping 10.112.23.20
10.112.23.20 に ping を送信しています 32 バイトのデータ:
10.112.23.20 からの応答: バイト数 =32 時間 =142ms TTL=255
10.112.23.20 からの応答: バイト数 =32 時間 =49ms TTL=255
10.112.23.20 からの応答: バイト数 =32 時間 =48ms TTL=255
10.112.23.20 からの応答: バイト数 =32 時間 =64ms TTL=255

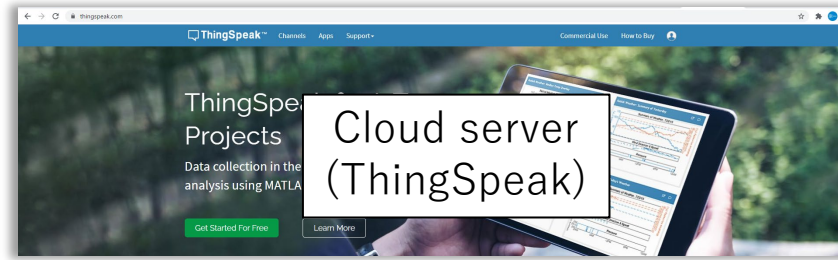
10.112.23.20 の ping 統計:
    パケット数: 送信 = 4、受信 = 4、損失 = 0 (0% の損失)、
    ラウンドトリップの概算時間 (ミリ秒):
        最小 = 48ms、最大 = 142ms、平均 = 75ms      Success!
```

Wi-Fi connection : Automatic internet connection in standalone operating

- Cut and paste, then edit



Today's goal



Cloud server
(ThingSpeak)

Internet

Inside your laboratory

Internet

Outside your laboratory



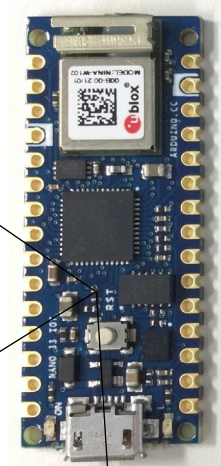
Temperature measurement

I2C



Voltage measurement
(laser power, MOT fluorescence)

I2C



I2C

Displaying experimental
conditions on an OLED

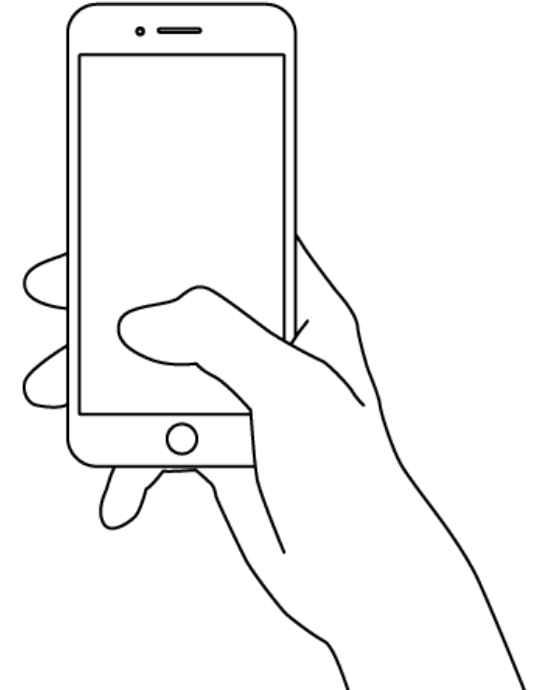
Wi-Fi



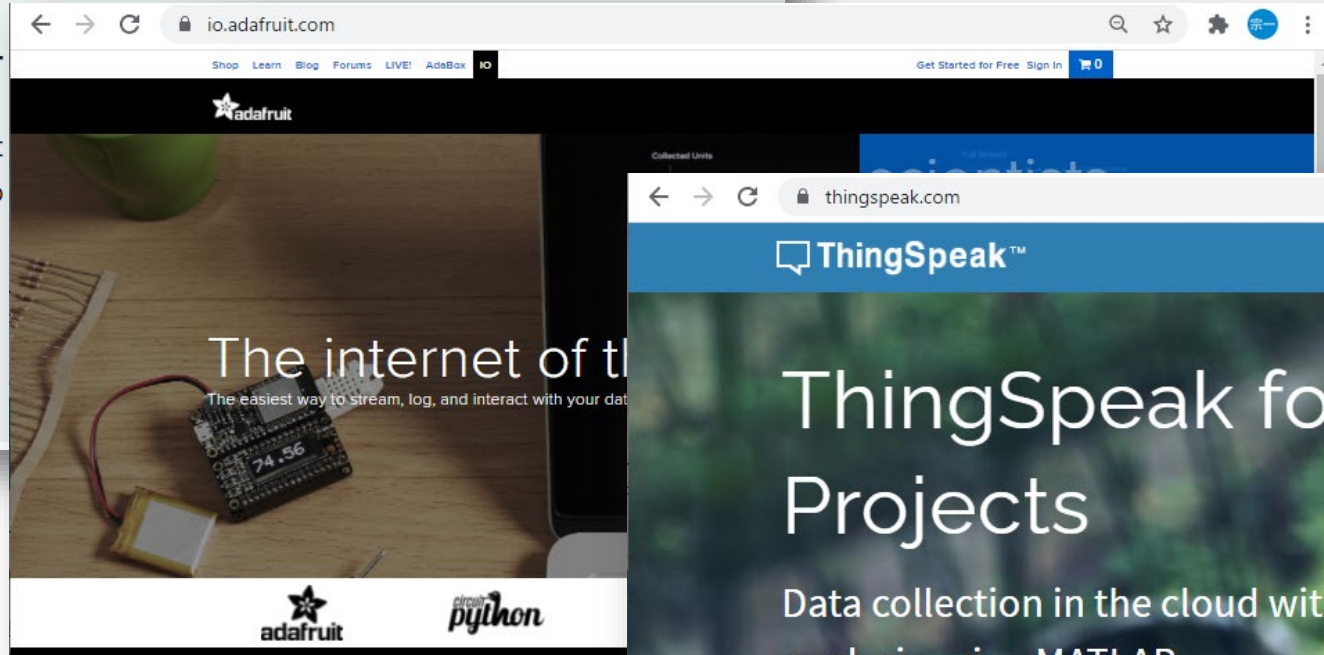
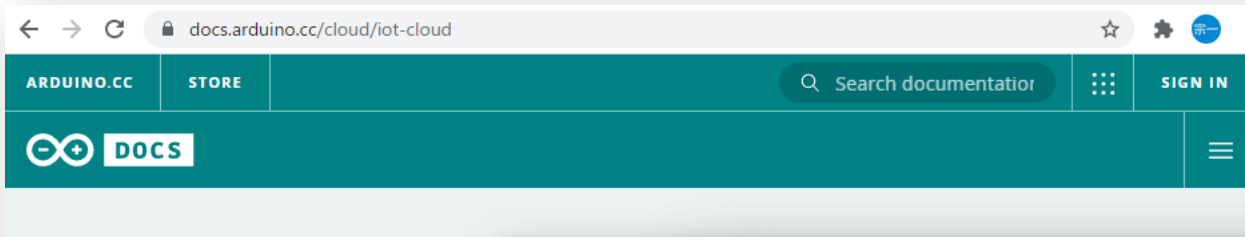
Wi-Fi router



External trigger for upload
data to the cloud



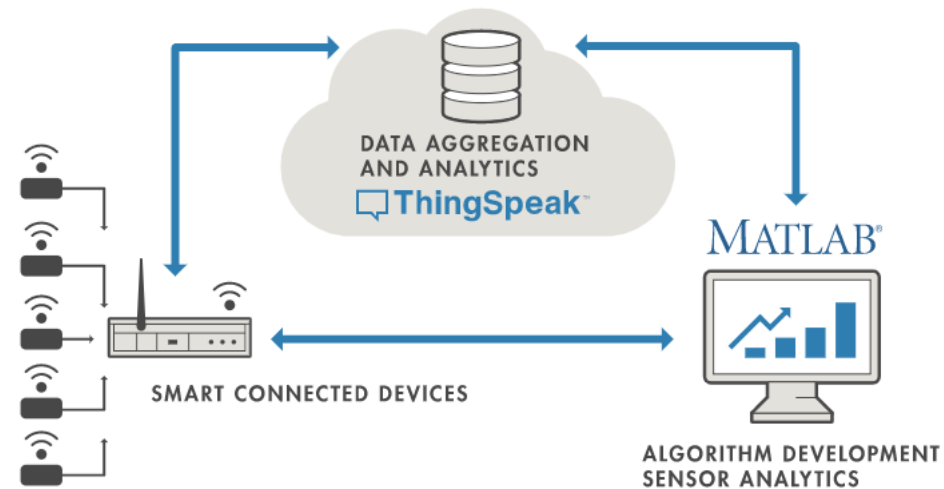
Upload data to an IOT cloud server



Upload data to an IOT cloud server




<https://thingspeak.com/>




https://thingspeak.com/pages/learn_more


Upload data to an IOT cloud server




Connect more sensors




Share channel views with colleagues and students



Faster update rates



Extended compute time for analytics



Live tech support

	FREE For small non-commercial projects	ACADEMIC For academic use by faculty, staff, or researchers at degree-granting institutions ⁽¹⁾
Scalable for larger projects	✘ No. Annual usage is capped.	✔
Number of messages	3 million/year (~8,200/day) ⁽²⁾	33 million/year per unit (~90,000/day per unit) ⁽²⁾
Message update interval limit	Every 15 seconds	Every second
Number of channels	4	250 per unit
MATLAB Compute Timeout	20 seconds	60 seconds
Number of simultaneous MQTT subscriptions	Limited to 3	50 per unit
Private channel sharing	Limited to 3 shares	Unlimited
Technical Support	Community Support	Standard MathWorks support

License type: **Academic**

ThingSpeak units:

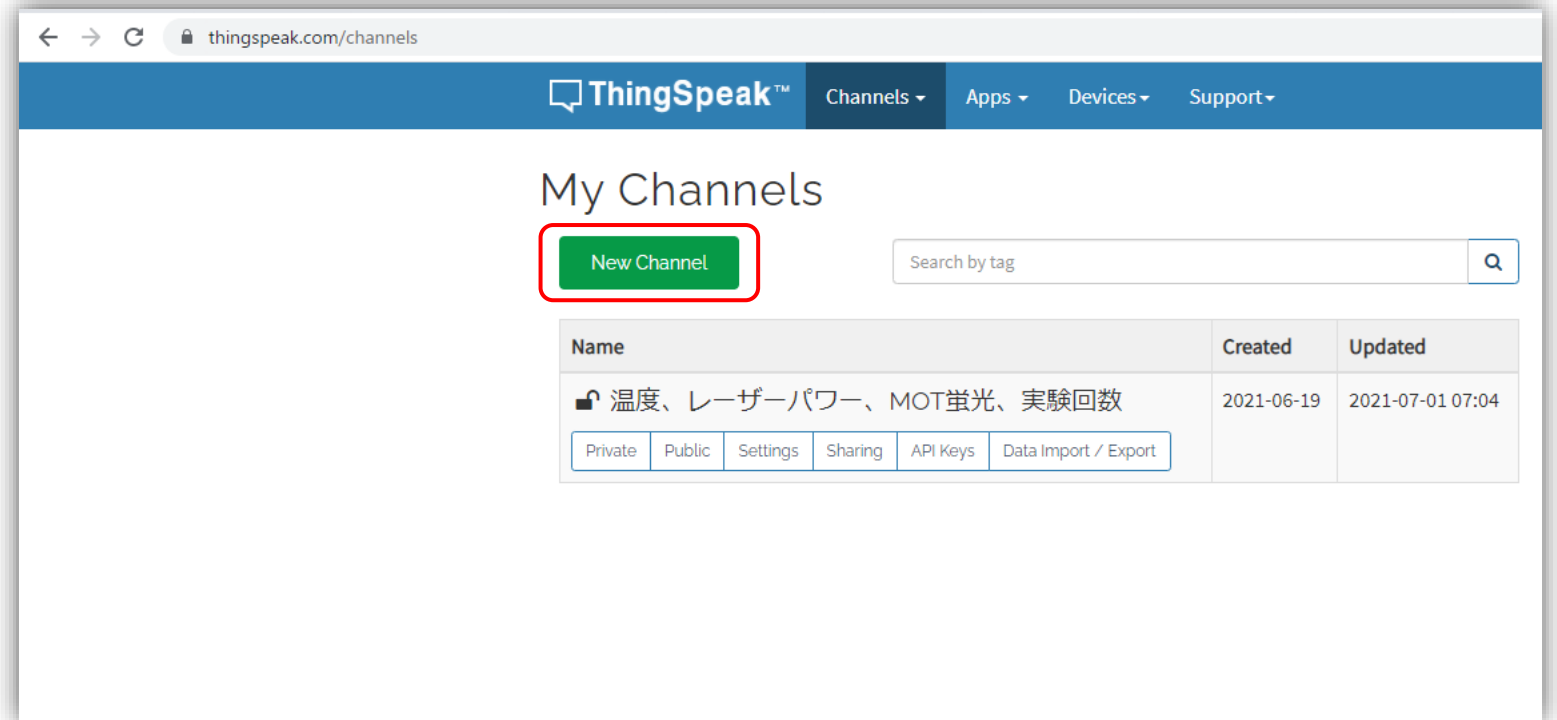
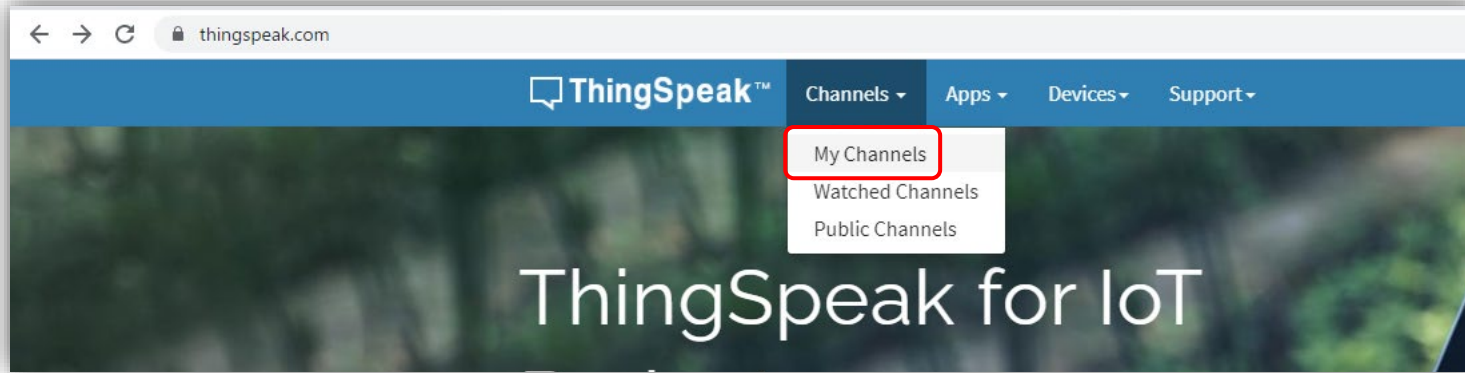
x ¥ 36,500
price/unit/year

Total: **¥ 36,500/year**

[Purchase](#)

You will be taken to the MathWorks store to complete your purchase.

Upload data to the IOT cloud server



Upload data to the IOT cloud server

ThingSpeak Channels Apps Devices Support Commercial Use How to Buy MH

New Channel

Name

Description

Field 1

Field 2

Field 3

Field 4

Field 5

Field 6

Field 7

Field 8

Metadata

Tags
(Tags are comma separated)

Link to External Site

Link to GitHub

Elevation

Show Channel Location

Latitude

Longitude

Show Video

YouTube

Vimeo

Video URL

Show Status

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

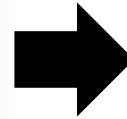
- Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- Channel Name:** Enter a unique name for the ThingSpeak channel.
- Description:** Enter a description of the ThingSpeak channel.
- Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- Tags:** Enter keywords that identify the channel. Separate tags with commas.
- Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- Show Channel Location:**
 - Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
 - Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
 - Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.
- Video URL:** If you have a YouTube[™] or Vimeo[®] video that displays your channel information, specify the full path of the video URL.
- Link to GitHub:** If you store your ThingSpeak code on GitHub[®], specify the GitHub repository URL.

Using the Channel

You can get data into a channel from a device, website, or another ThingsSpeak channel. You can then visualize data and transform it using ThingSpeak Apps.

See [Get Started with ThingSpeak[™]](#) for an example of measuring dew point from a weather station that acquires data from an Arduino[®] device.

[Learn More](#)



ThingSpeak Channels Apps Devices Support Commercial Use How to Buy MH

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Channel 2 of 2 < >

Channel Stats

Created: [about a minute ago](#)
Entries: 0

Edit the title

Field 1 Chart

Temperature

Field 2 Chart

V0

Field 3 Chart

V1

Field 4 Chart

V2

Field 5 Chart

V3

Upload data to the IOT cloud server

These three information are needed to read and write data

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy MH

Ultracold atom workshop "Atom no Kai"

Channel ID: [Redacted] IoT demonstration
Author: [Redacted]
Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Write API Key

Key: [Redacted]

Generate New Write API Key

Read API Keys

Key: [Redacted]

Note: [Empty text area]

Save Note Delete API Key

Add New Read API Key

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=[Redacted]&field=10
```

Read a Channel Feed

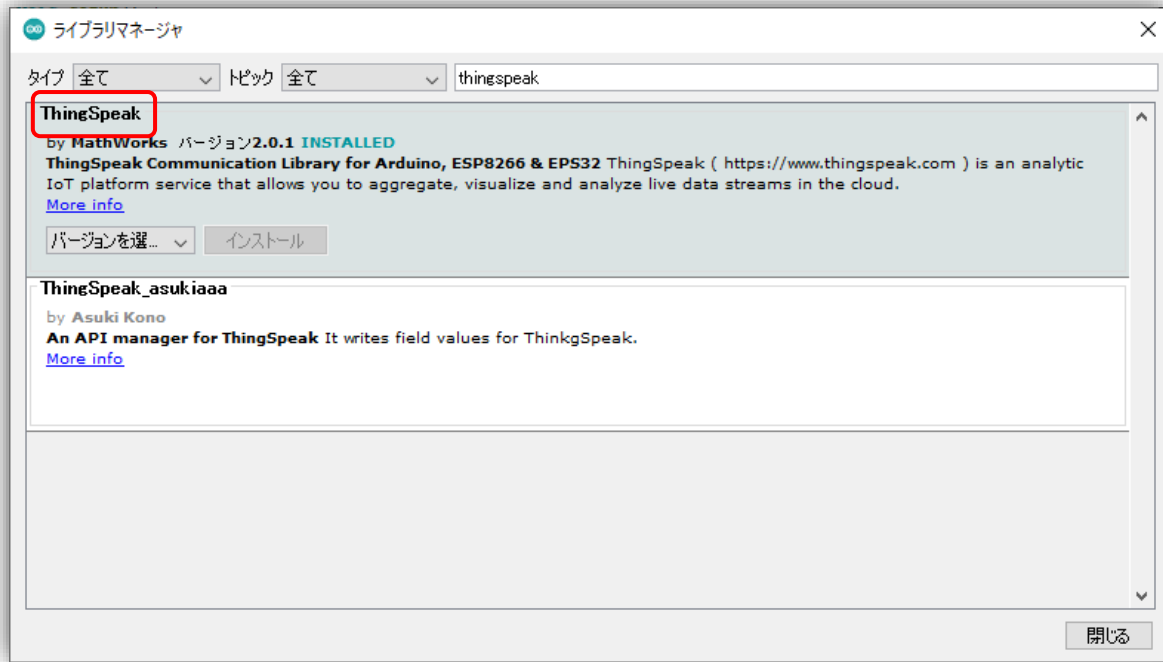
```
GET https://api.thingspeak.com/channel/[Redacted]?api_key=[Redacted]
```

Read a Channel Field

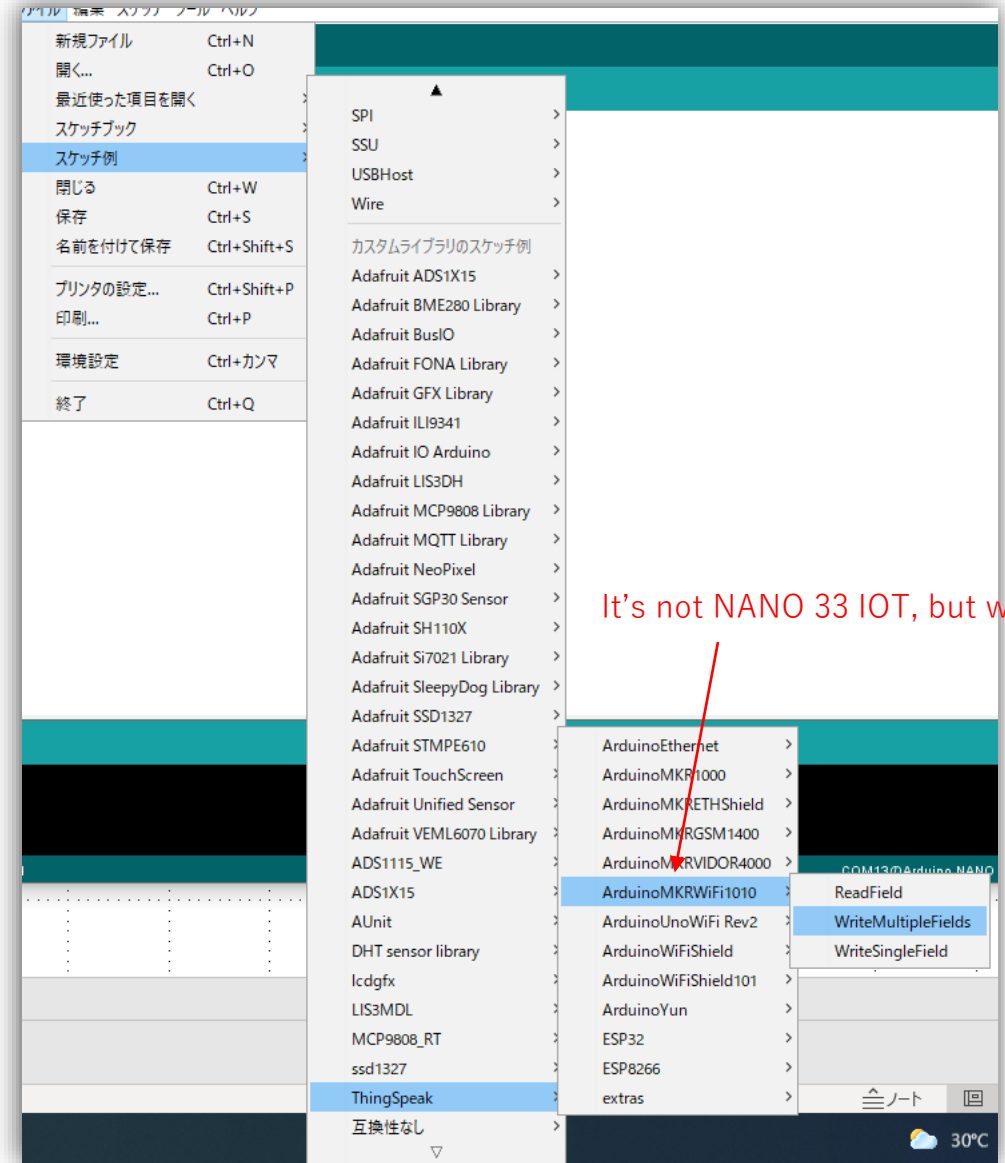
```
GET https://api.thingspeak.com/channel/[Redacted]/son?api_key=[Redacted]
```

Upload data to the IOT cloud server

- Install the library for 'ThingSpeak'



- Open the sample code, cut, paste, and edit



Upload data to the IOT cloud server

- Open the sample code, cut, paste, and edit

```
Hello_Temperature_ADC_Wifi
#include <Adafruit_SSD1327.h>
#include <Wire.h>
#include "Adafruit_MCP9808.h"
#include <ADS1115_WE.h>
#include <SPI.h>
#include <WiFiNINA.h>
#include "ThingSpeak.h" // always include thingspeak header file after other header files and custom macros

char ssid[] = "Buffalo-G-9330"; // your network SSID (name)
char pass[] = " "; // your network password (use for WPA, or use as key for WEP)
int status = WL_IDLE_STATUS; // the WiFi radio's status

WiFiClient client;

unsigned long myChannelNumber = ; // Your channel ID
const char * myWriteAPIKey = " "; // Your Write API Key

// I2C
#define OLED_RESET -1
Adafruit_SSD1327 display(128, 128, &Wire, OLED_RESET, 1000000);

// Create the MCP9808 temperature sensor object
Adafruit_MCP9808 tempsensor = Adafruit_MCP9808();

// Create the ADS1115 ADC object
ADS1115_WE adc = ADS1115_WE(0x48);

// the setup routine runs once when you press reset:
void setup() {
  ThingSpeak.begin(client); //Initialize ThingSpeak

  display.begin(0x3D);
  display.clearDisplay();
  display.setTextSize(1);
  display.display();
  display.setCursor(0, 0);
  display.setTextSize(1);
```

Your channel ID
Your Write API Key

If you forget it, your Arduino freezes.

```
// the loop routine runs over and over again forever:
void loop() {

  float c = tempsensor.readTempC();
  float V0 = readChannel(ADS1115_COMP_0_GND);
  float V1 = readChannel(ADS1115_COMP_1_GND);
  float V2 = readChannel(ADS1115_COMP_2_GND);
  float V3 = readChannel(ADS1115_COMP_3_GND);

  display.clearDisplay();

  display.setCursor(0, 0);

  printWifiData();
  display.println("");

  display.print("T : "); display.print(c, 3); display.println(" [C]");
  display.println("");
  display.print("V0 : "); display.print(V0, 1); display.println(" [mV]");
  display.print("V1 : "); display.print(V1, 1); display.println(" [mV]");
  display.print("V2 : "); display.print(V2, 1); display.println(" [mV]");
  display.print("V3 : "); display.print(V3, 1); display.println(" [mV]");
  display.println("");

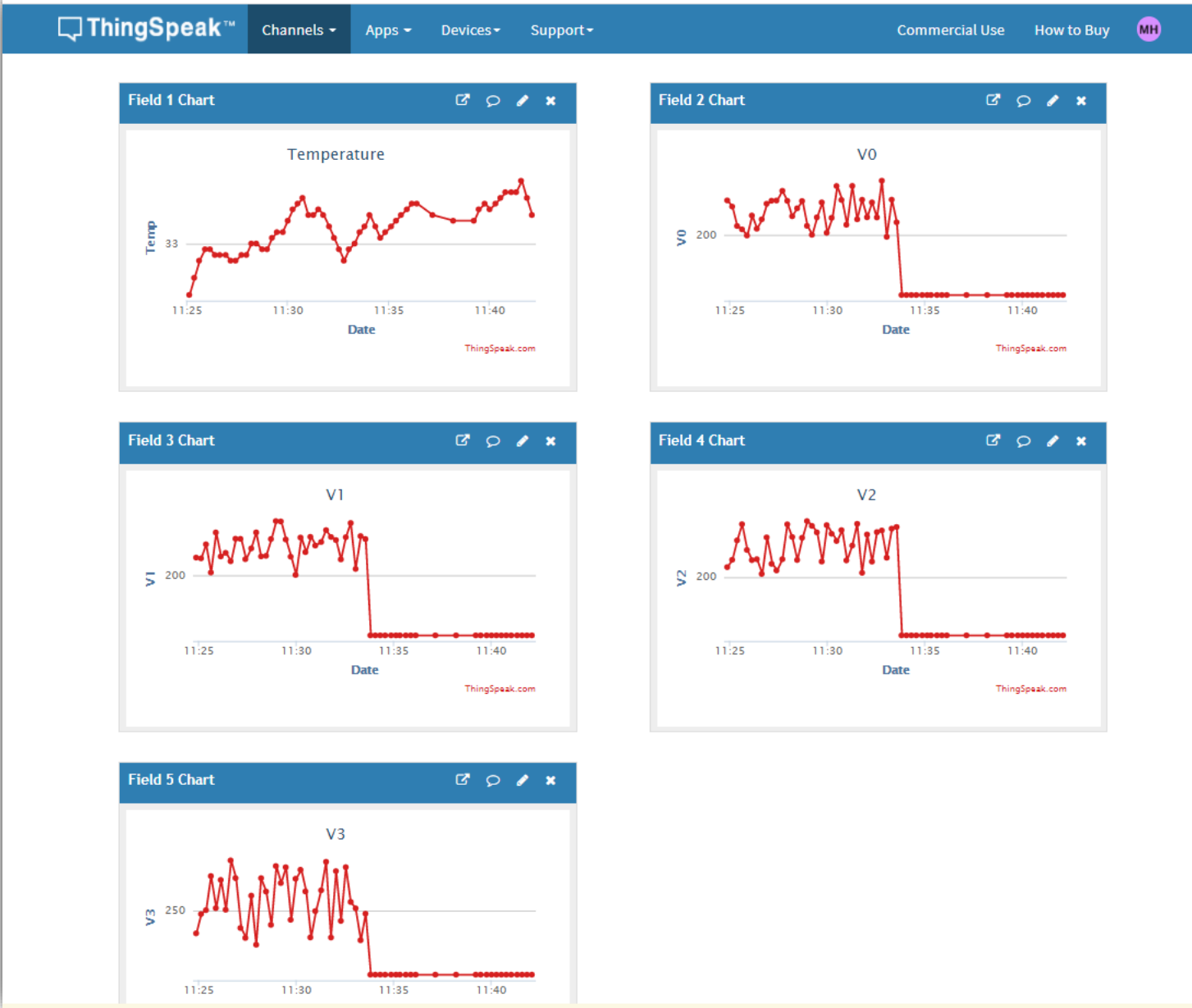
  ThingSpeak.setField(1, c);
  ThingSpeak.setField(2, V0);
  ThingSpeak.setField(3, V1);
  ThingSpeak.setField(4, V2);
  ThingSpeak.setField(5, V3);

  int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
  if(x == 200){
    display.println("Channel update successful.");
  }

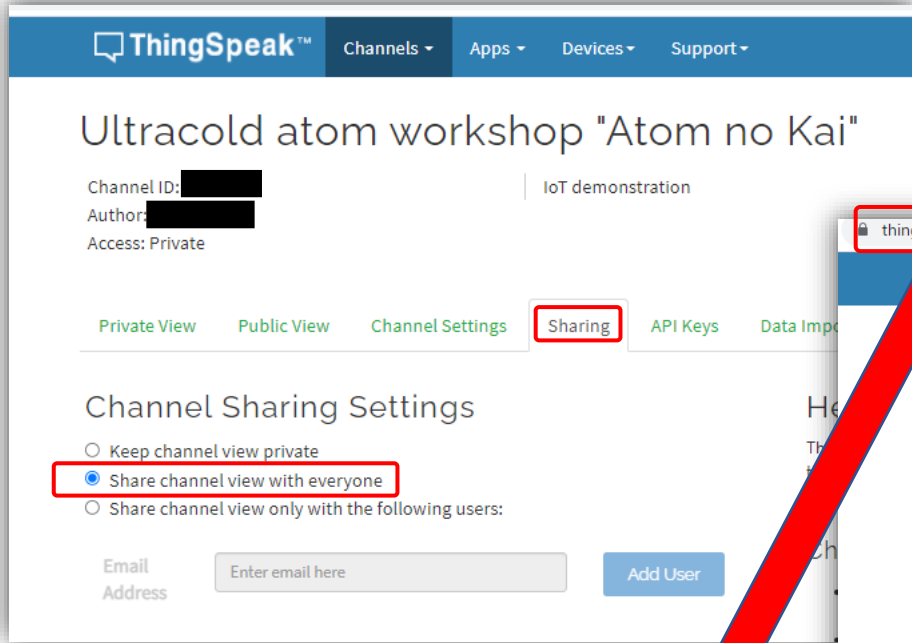
  display.display();

  delay(500); // delay in between reads for stability
}
```

Upload data to the IOT cloud server



Upload data to the IOT cloud server



ThingSpeak™ Channels Apps Devices Support

Ultracold atom workshop "Atom no Kai"

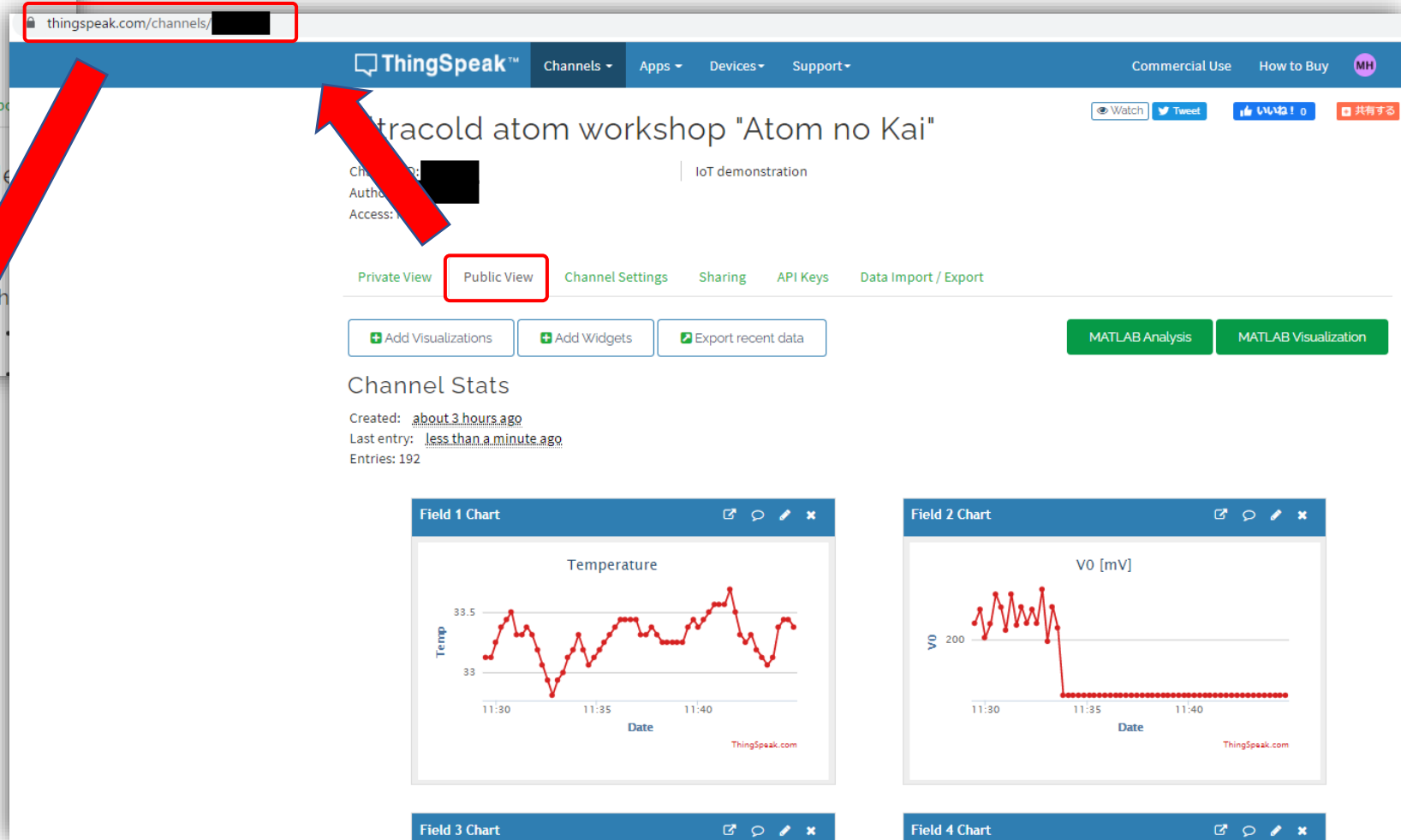
Channel ID: [redacted] IoT demonstration
Author: [redacted]
Access: Private

Private View Public View Channel Settings **Sharing** API Keys Data Import

Channel Sharing Settings

- Keep channel view private
- Share channel view with everyone**
- Share channel view only with the following users:

Email Address



thingspeak.com/channels/[redacted]

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy MH

Ultracold atom workshop "Atom no Kai"

Channel ID: [redacted] IoT demonstration
Author: [redacted]
Access: Private

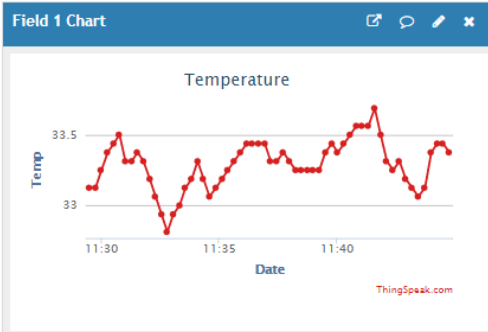
Private View **Public View** Channel Settings Sharing API Keys Data Import / Export

Channel Stats

Created: [about 3 hours ago](#)
Last entry: [less than a minute ago](#)
Entries: 192

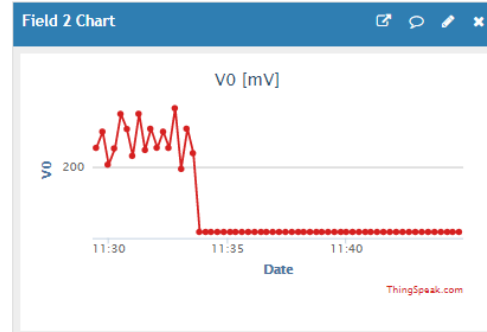
Field 1 Chart

Temperature



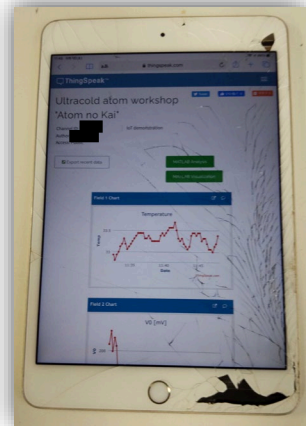
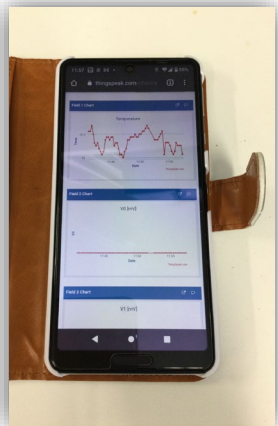
Field 2 Chart

V0 [mV]



Field 3 Chart

Field 4 Chart



Download data from the IOT cloud server

ThingSpeak™ Channels Apps Devices Support

Ultracold atom workshop "Atom no Kai"

Channel ID: [REDACTED] IoT demonstration
Author: [REDACTED]
Access: Public

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Import

Upload a CSV file to import data into this channel.

File:

Time Zone: (GMT+00:00) UTC

Export

Download all of this Channel's feeds in CSV format.

Time Zone: (GMT+09:00) Osaka

Help

Import

The correct format for data is as follows:
field names *field1*, *field2*, and *field3*
datetime, *field1*, *field2*, *field3*
2019-01-01T10:11:12-0500,1,2,3

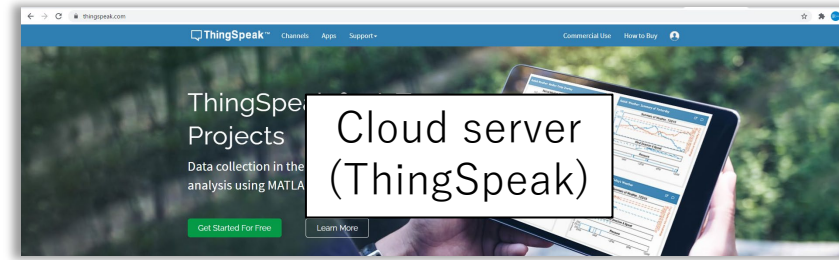
CSV Import Format

Other Import and Export Options

You can also use MATLAB, the ThingSpeak API, or the ThingSpeak CLI to import data.

created_at	entry_id	field1	field2	field3	field4	field5	latitude	longitude	elevation
2021-08-0	1	29	274.875	272.375	282.5	284.3125			
2021-08-0	2	29	277.0625	279.4375	276.5	276.0625			
2021-08-0	3	29.0625	283.1875	278.5625	280.3125	283.6875			
2021-08-0	4	29.0625	284.4375	283.25	272.75	273.8125			
2021-08-0	5	29.0625	284.875	281.5625	279.5	280.375			
2021-08-0	6	29	273.0625	272.75	280.9375	280.375			
2021-08-0	7	29	285.75	277.0625	279	282.9375			
2021-08-0	8	29	276.125	273.8125	285.5	285.125			
2021-08-0	9	29	285.0625	284.9375	270.4375	274.4375			
2021-08-0	10	28.9375	286.125	272.0625	277.5625	282.25			
2021-08-0	11	nan	0	0	0	0			
2021-08-0	12	28.9375	286.3125	283.75	280.625	277.8125			
2021-08-0	13	28.9375	281.125	276.75	274.9375	273.9375			
2021-08-0	14	29	271.3125	271.6875	282.375	283.6875			
2021-08-0	15	29	289.375	285.6875	273	275.375			
2021-08-0	16	29	284.5625	283.6875	272.5625	272.9375			
2021-08-0	17	29.0625	274.8125	279.75	278.125	277.8125			
2021-08-0	18	29	286.6875	281.9375	278.6875	284.9375			
2021-08-0	19	29.0625	272.1875	272.875	275.625	279.375			
2021-08-0	20	29	286.625	281.5	278.3125	282.375			
2021-08-0	21	29.0625	275.0625	276.375	279.5	274.875			
2021-08-0	22	29.0625	282.5625	278.9375	276.9375	276			
2021-08-0	23	29.0625	278	276.5625	276.5	275.875			
2021-08-0	24	29	271.0625	271.9375	281.5625	282.9375			
2021-08-0	25	29	282.25	281.8125	282.5	273.75			

Today's goal

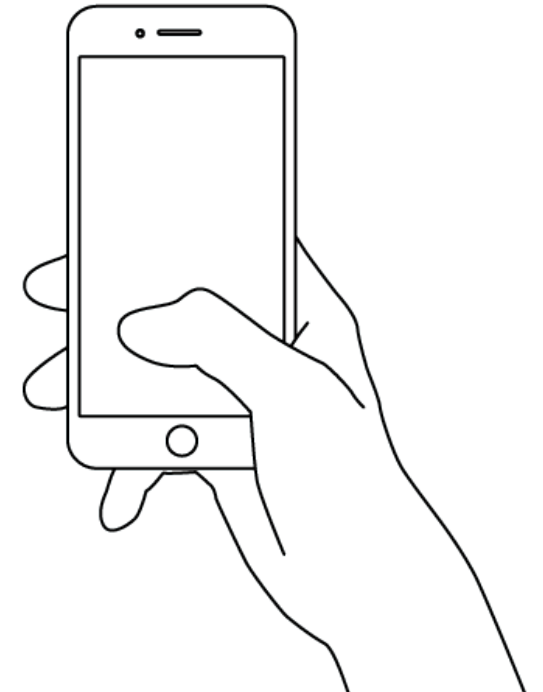
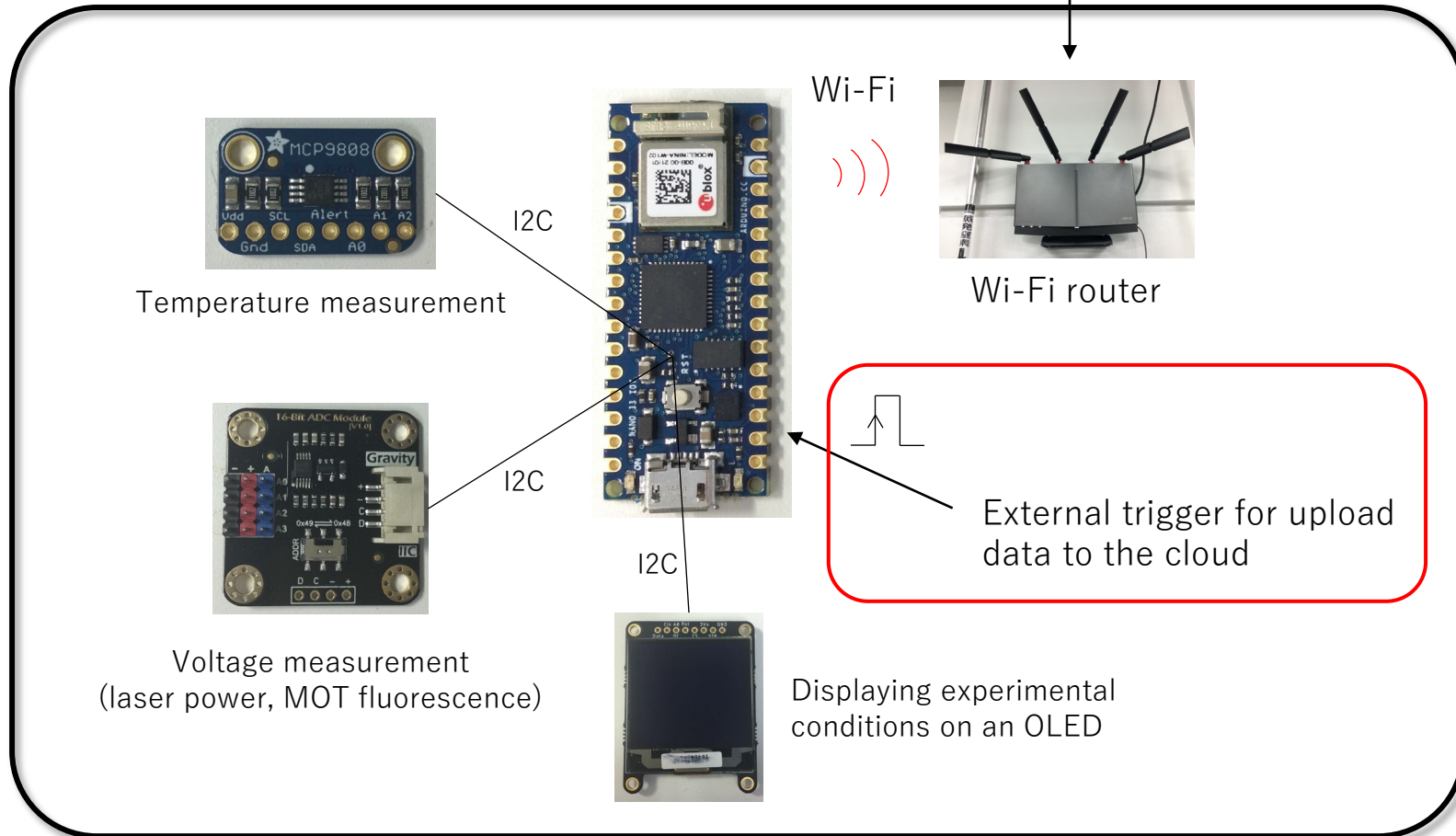


Internet

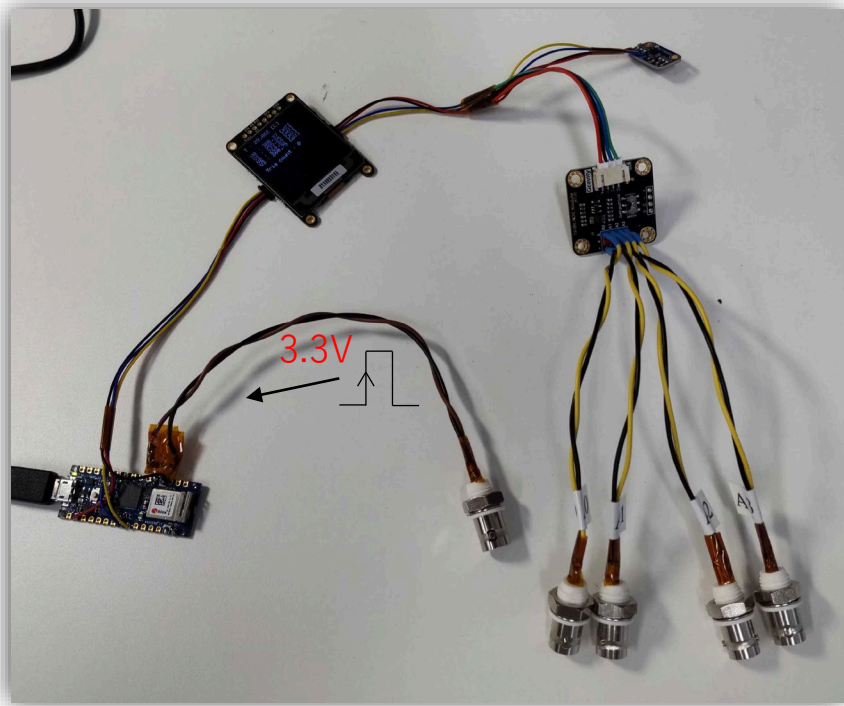
Inside your laboratory

Internet

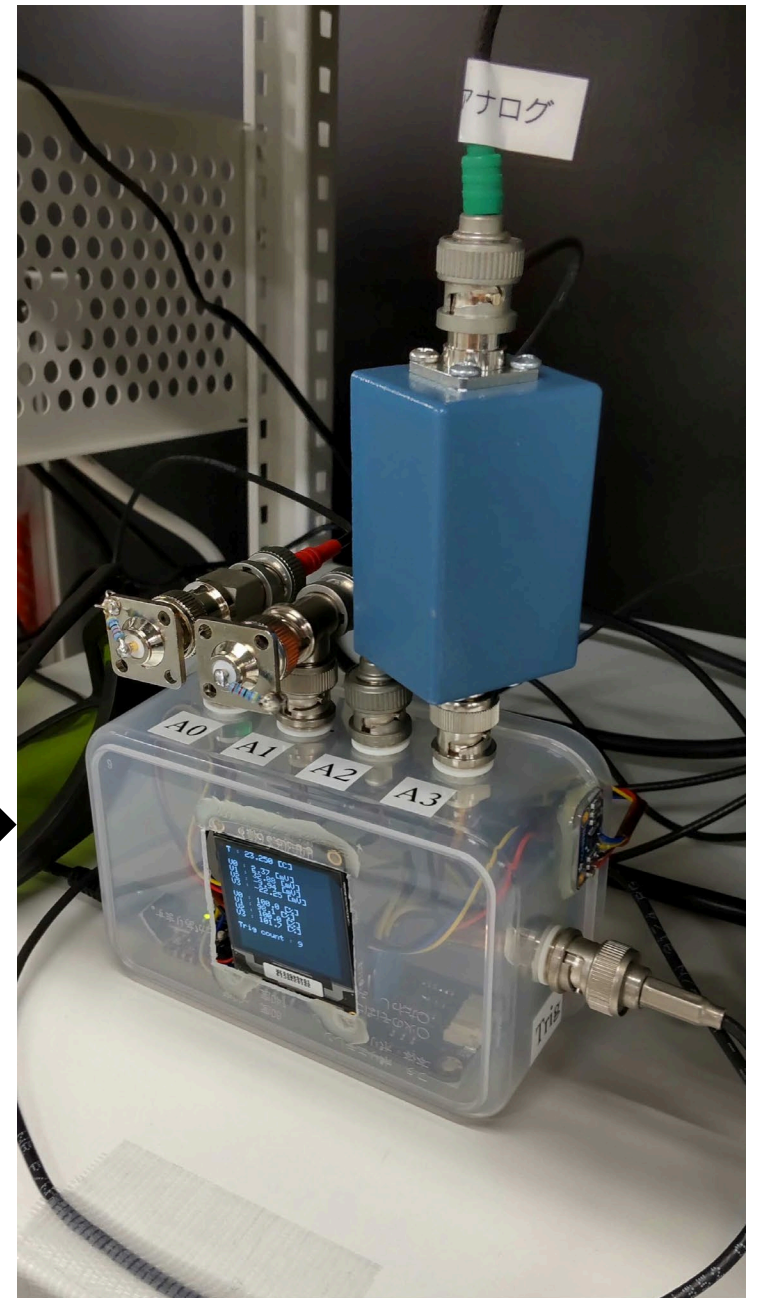
Outside your laboratory



Upload by external triggers

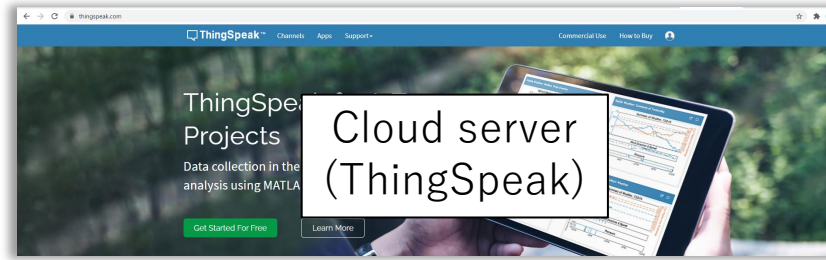


Packing



Conclusion

- Arduino is fun and practical
- LED blink, Hello world
- I2C communication to digital devices
- Wi-Fi connection
- IoT cloud server



Internet

Inside your laboratory

Internet

Outside your laboratory

