

# **Tutorial ESP32-S3 I**

## using Arduino IDE software

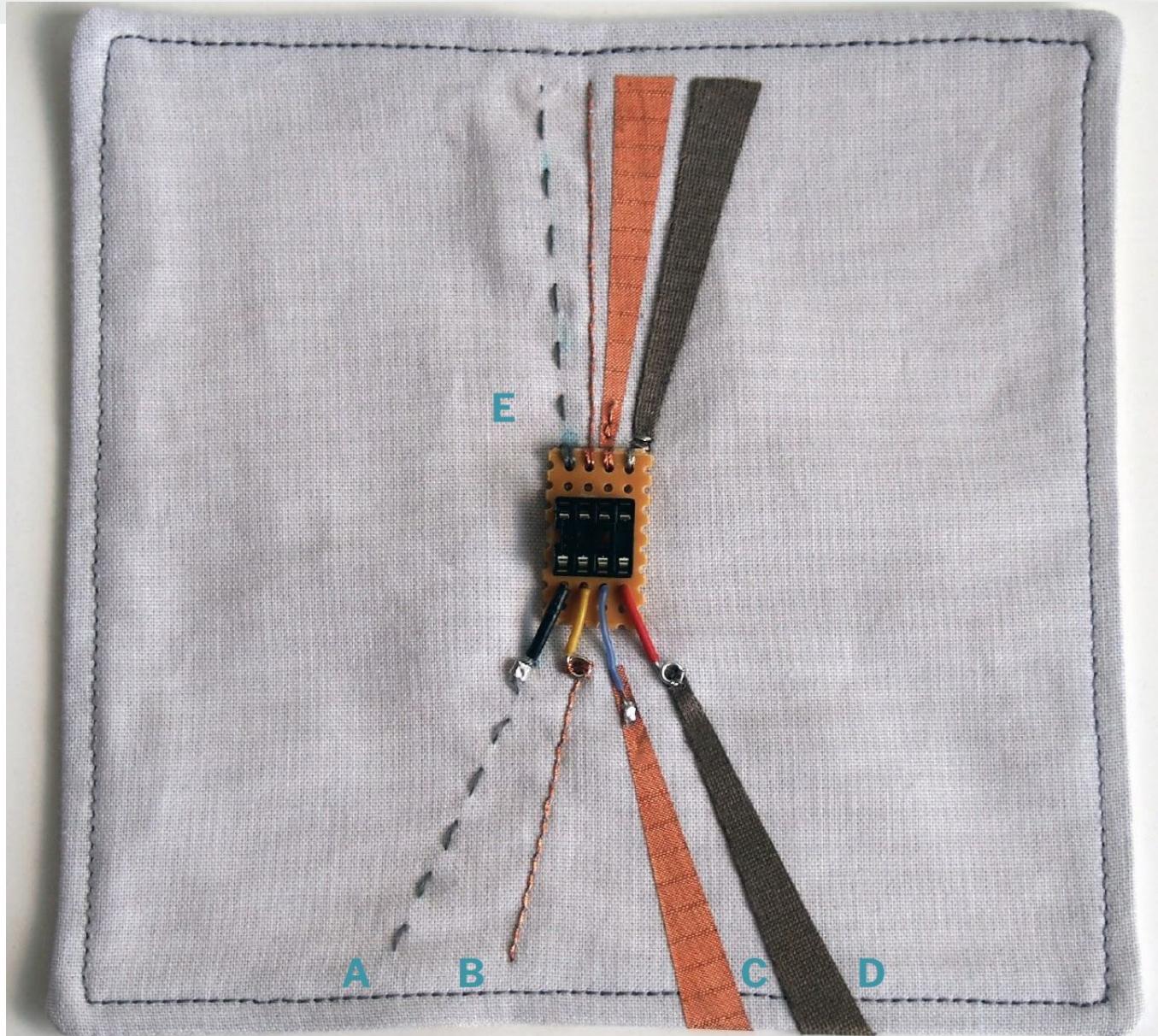
---

# Learning Outcomes

- > Connecting hard circuit to soft circuits
- > Introduction to Physical Computing + XIAO ESP32 S3
- > Set up circuit to connect your sensor to XIAO ESP32 S3
- > Write basic code, read sensor data
- > Extra: connect two sensors, two data points in code

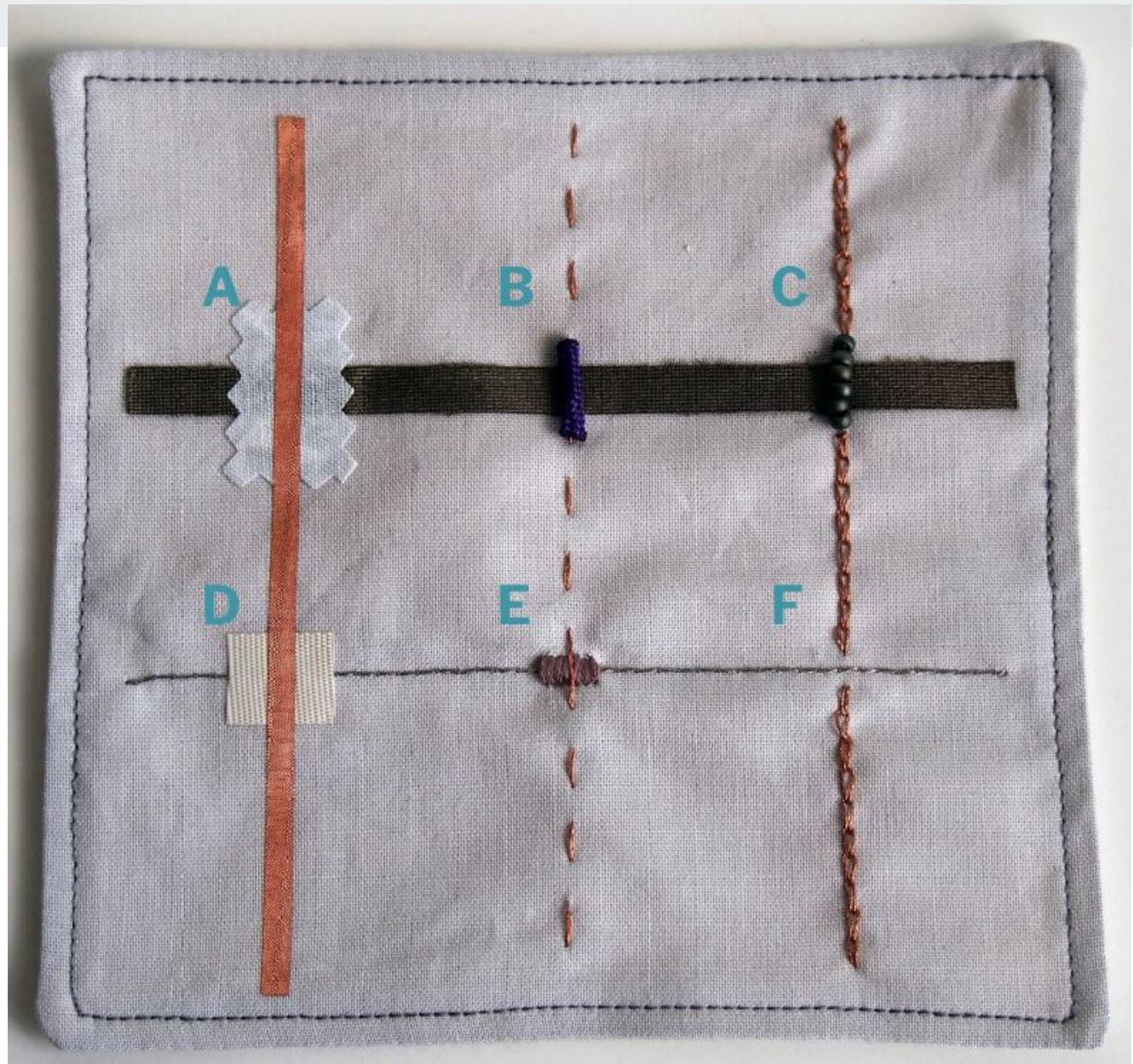
## Hard to soft circuit.

- A} Use a crimp bead to connect to the thread;
- B} Use pliers to create a sewable hole with a wire (same principle as we did with the LED);
- C} solder on fabric > Needs to be tested before hand and depending on the fabric if it works or not
- D} sew wire directly to fabric;
- E} Desoldering the pins: leaves you with metal loops which you can sew conductive thread in;
- F} Using items like snap-buttons or safety pins.



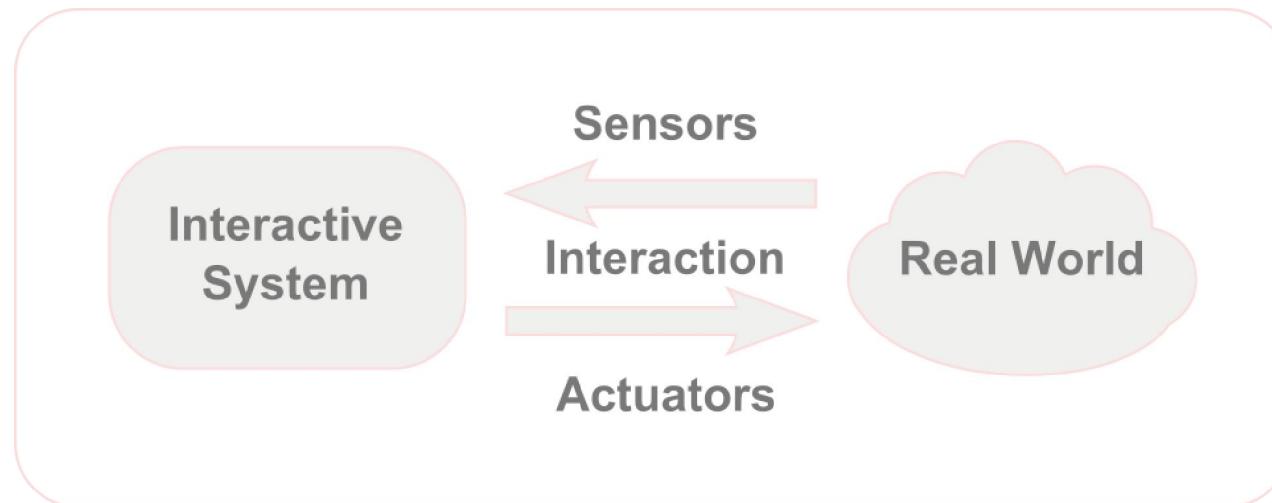
## Hard to soft circuit: overlapping traces.

- A} Fabric (with iron-on adhesive);
- B} Paracord;
- C} Beads;
- D} Tape;
- E} Embroider;
- D} Sew under.

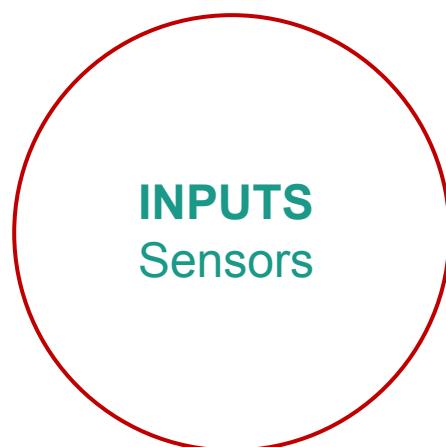
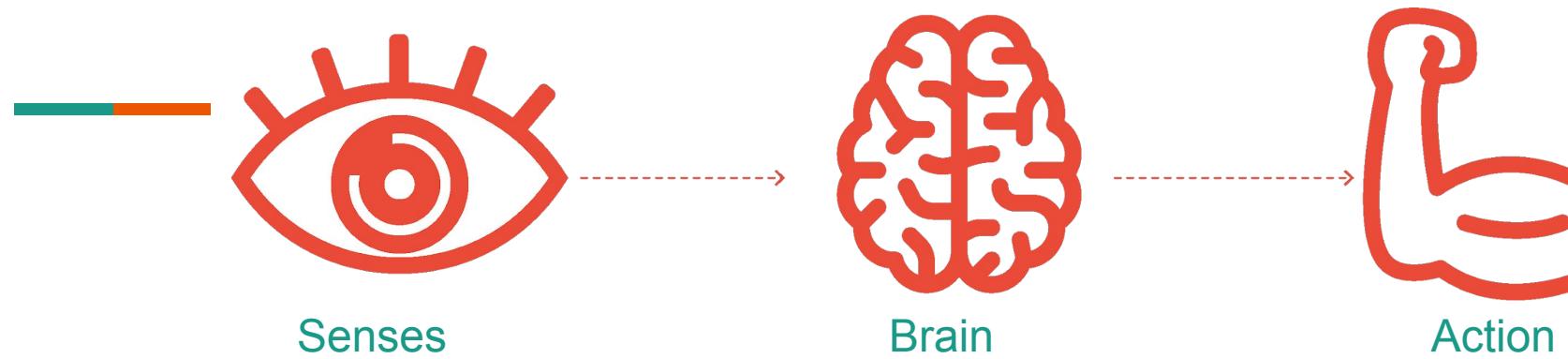


<https://www.computationalcraft.io/about/>

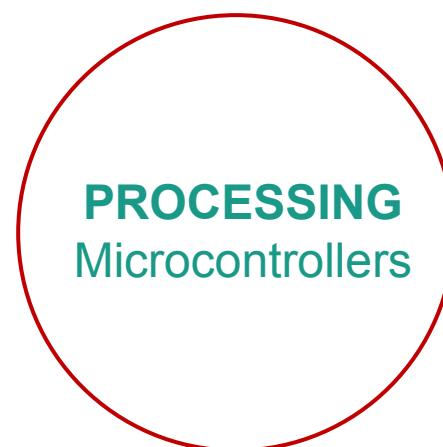
# PHYSICAL COMPUTING



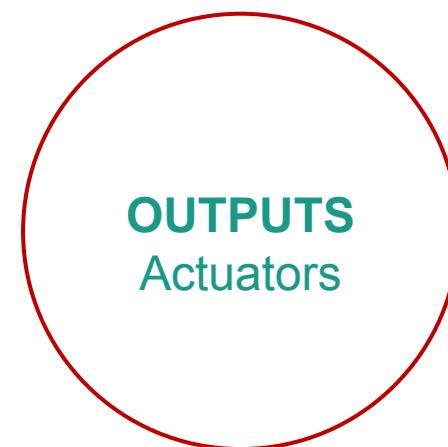
## ELECTRIC SYSTEM



Take information



Read and process the information



React to the information

# ELECTRIC SYSTEM: Processors



Processors (CPU)

Microcontrollers

Printed circuit board (PCB)

## HARDWARE

## SOFTWARE

Code:

Java, C, C++, Python, C#

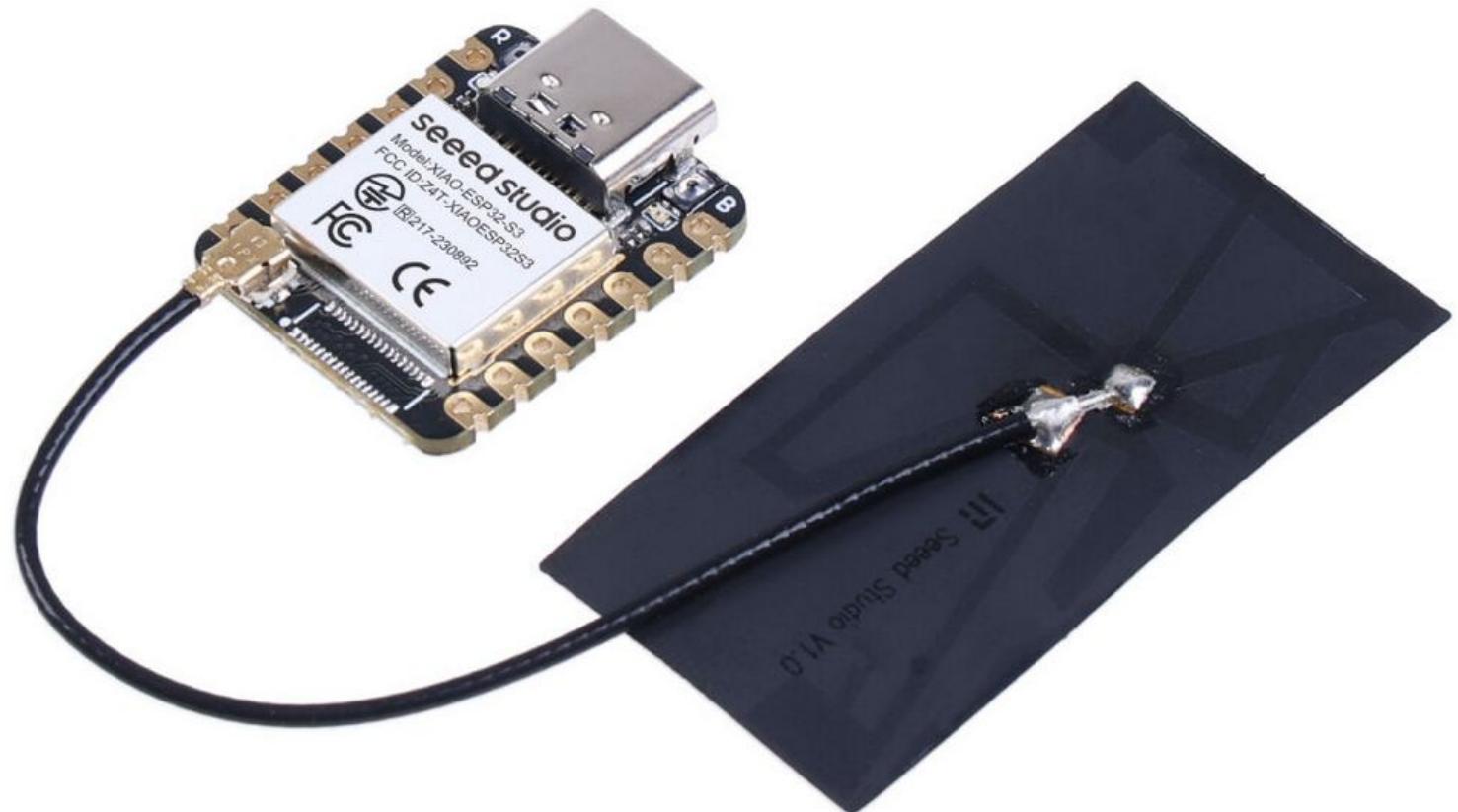
```
1  function execute(sessId, raw) {
2      var tokens = raw.match(/\w+|[\+=();]/g);
3      var peek = _ => tokens[0];
4      var eat = _ => tokens.shift();
5      var ate = x => peek() === x && eat();
6      var want = x => {
7          if (!ate(x)) throw 'Expected \'' + x + '\'';
8      };
9
10     function statement() {
11         scope[sessId][eat()] = [want('='), expr()];
12     }
13
14     function expr() {
15         for (var v = term(); ate('+'); v = v + term())
16             eat();
17     }
18
19     function term() {
20         for (var v = atom(); ate('('); v = v([expr()])
21             eat());
22             eat();
23         eat();
24         return v;
25     }
26
27     function atom() {
28         if (ate('\"')) {
29             var s = '';
30             while (ate('"')) eat();
31             return s;
32         }
33         if (ate('\'')) {
34             var s = '';
35             while (ate('\'')) eat();
36             return s;
37         }
38         if (ate('0') || ate('1') || ate('2') || ate('3') || ate('4') || ate('5') || ate('6') || ate('7') || ate('8') || ate('9')) {
39             var s = '';
40             while (ate('0') || ate('1') || ate('2') || ate('3') || ate('4') || ate('5') || ate('6') || ate('7') || ate('8') || ate('9')) s += eat();
41             return s;
42         }
43         if (ate('.')) {
44             var s = '';
45             while (ate('.')) s += eat();
46             return s;
47         }
48         if (ate('/')) {
49             var s = '';
50             while (ate('/')) s += eat();
51             return s;
52         }
53         if (ate('*')) {
54             var s = '';
55             while (ate('*')) s += eat();
56             return s;
57         }
58         if (ate('<')) {
59             var s = '';
60             while (ate('<')) s += eat();
61             return s;
62         }
63         if (ate('>')) {
64             var s = '';
65             while (ate('>')) s += eat();
66             return s;
67         }
68         if (ate('&')) {
69             var s = '';
70             while (ate('&')) s += eat();
71             return s;
72         }
73         if (ate('&gt;')) {
74             var s = '';
75             while (ate('&gt;')) s += eat();
76             return s;
77         }
78         if (ate('&lt;')) {
79             var s = '';
80             while (ate('&lt;')) s += eat();
81             return s;
82         }
83         if (ate('&gt;=')) {
84             var s = '';
85             while (ate('&gt;=')) s += eat();
86             return s;
87         }
88         if (ate('&lt;=')) {
89             var s = '';
90             while (ate('&lt;=')) s += eat();
91             return s;
92         }
93         if (ate('&gt;gt;')) {
94             var s = '';
95             while (ate('&gt;gt;')) s += eat();
96             return s;
97         }
98         if (ate('&lt;lt;')) {
99             var s = '';
100            while (ate('&lt;lt;')) s += eat();
101            return s;
102        }
103    }
104}
```

## HARDWARE:

Wiki “Getting Started with Seeed Studio XIAO ESP32S3 Series”:  
[https://wiki.seeedstudio.com/xiao\\_esp32s3\\_getting\\_started/](https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/)

### Seeed Studio XIAO ESP32-S3

- Arduino / MicroPython supported
- Supports 2.4GHz WiFi and BLE 5.0 dual wireless communication
- Supports 100m+ remote communication when connected with U.FL antenna



Read all technical specifications and data-sheet here:

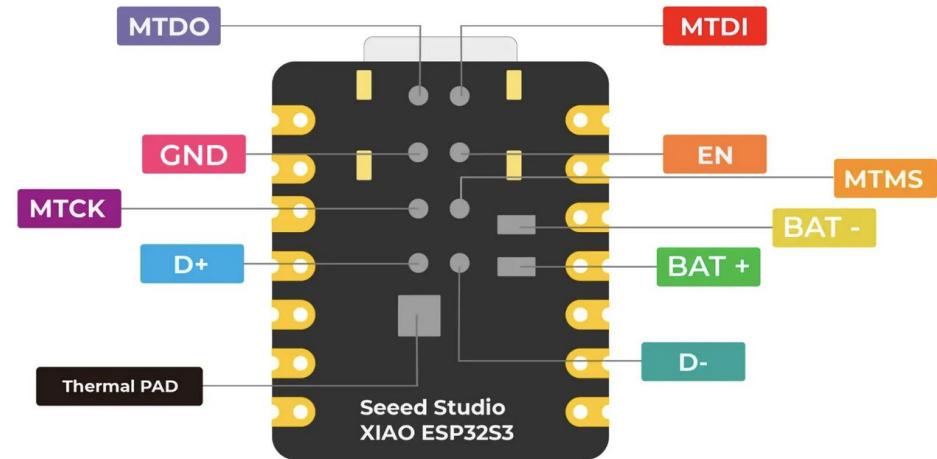
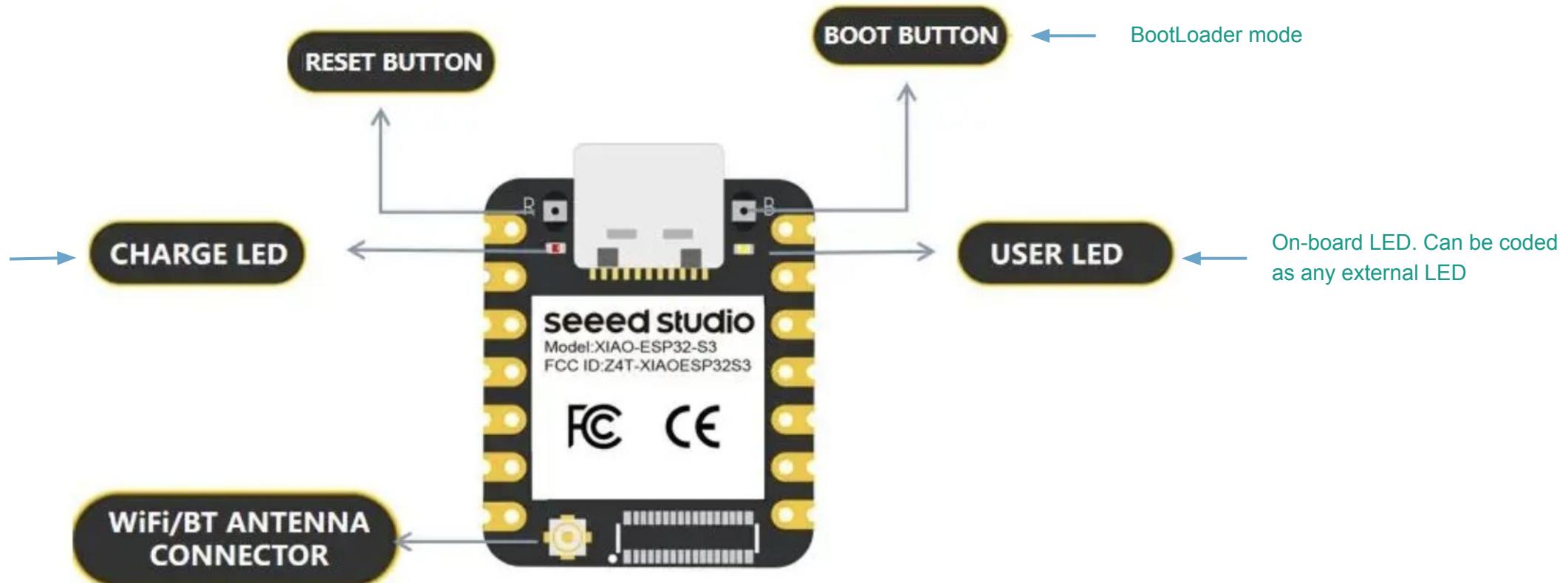
[https://files.seeedstudio.com/wiki/SeeedStudio-XIAO-ESP32S3/res/esp32-s3\\_datasheet.pdf](https://files.seeedstudio.com/wiki/SeeedStudio-XIAO-ESP32S3/res/esp32-s3_datasheet.pdf)

# OVERVIEW

---

## XIAO ESP32 S3

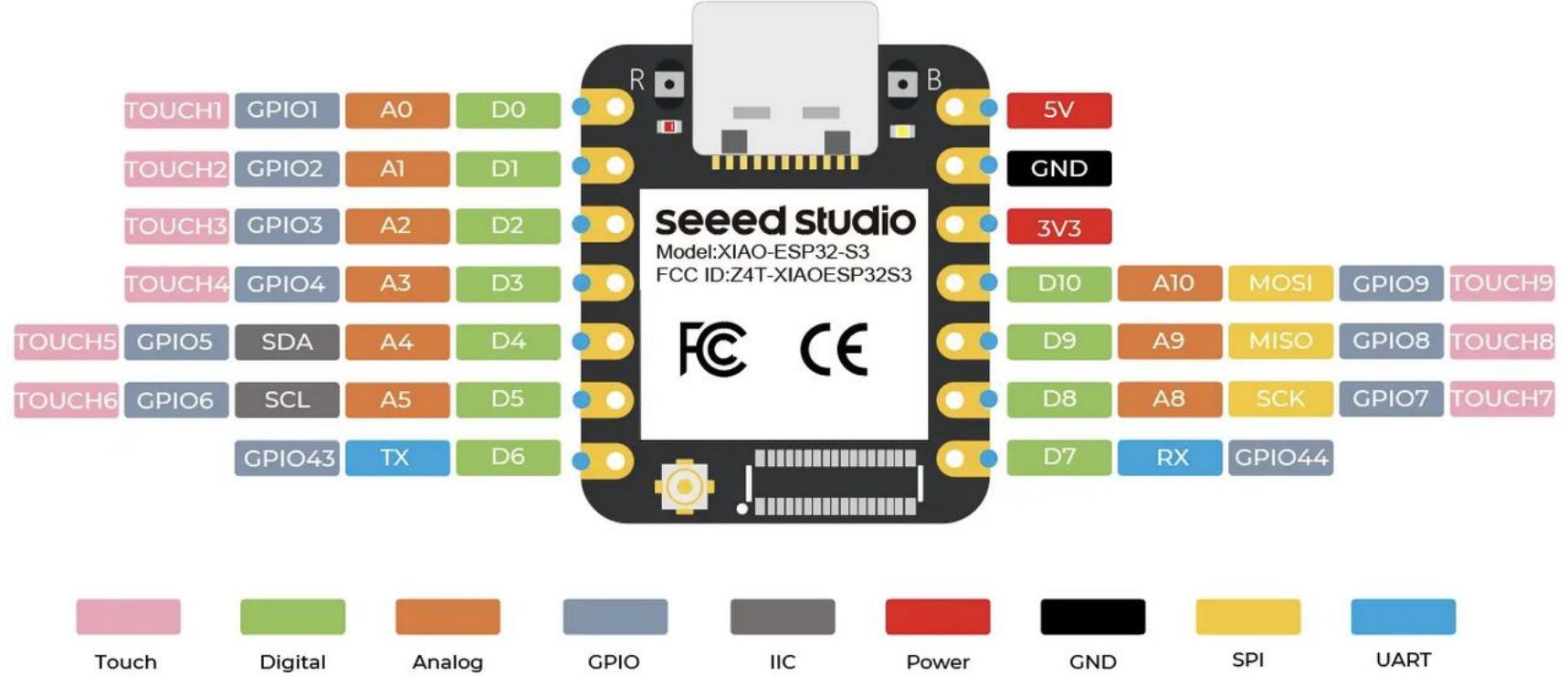
Will flicker when new code uploaded and when resetting board



# PIN OUT

---

## XIAO ESP32 S3

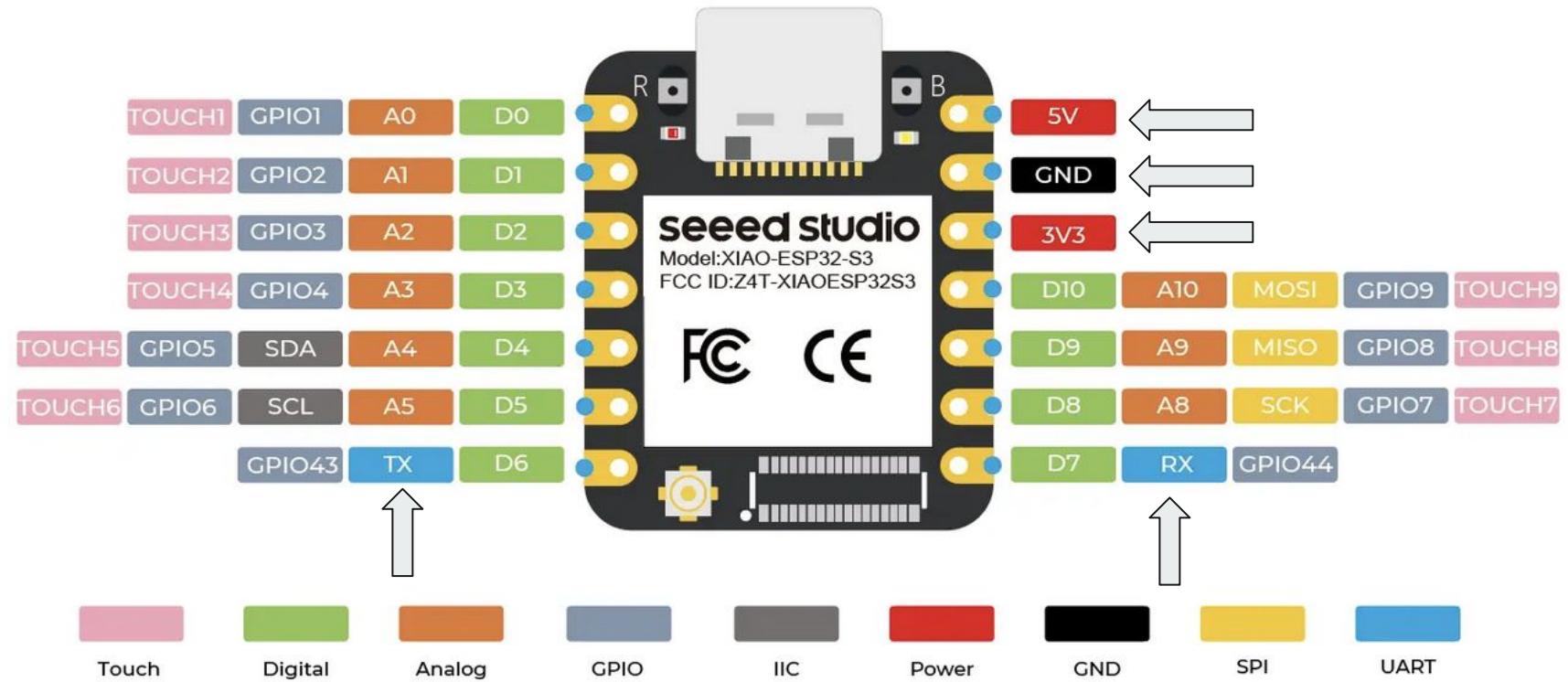


A pin is how **inputs** (buttons, etc) and **outputs** (LEDs, speakers, etc) communicate with the Microcontroller.

**TX/RX**  
//serial -  
transmit/receive

**1 ground pin**

**2 power pins**  
// 5V, 3.3V



## **POWER PINS**

XIAO ESP32 S3

# 10 Digital I/O Pins

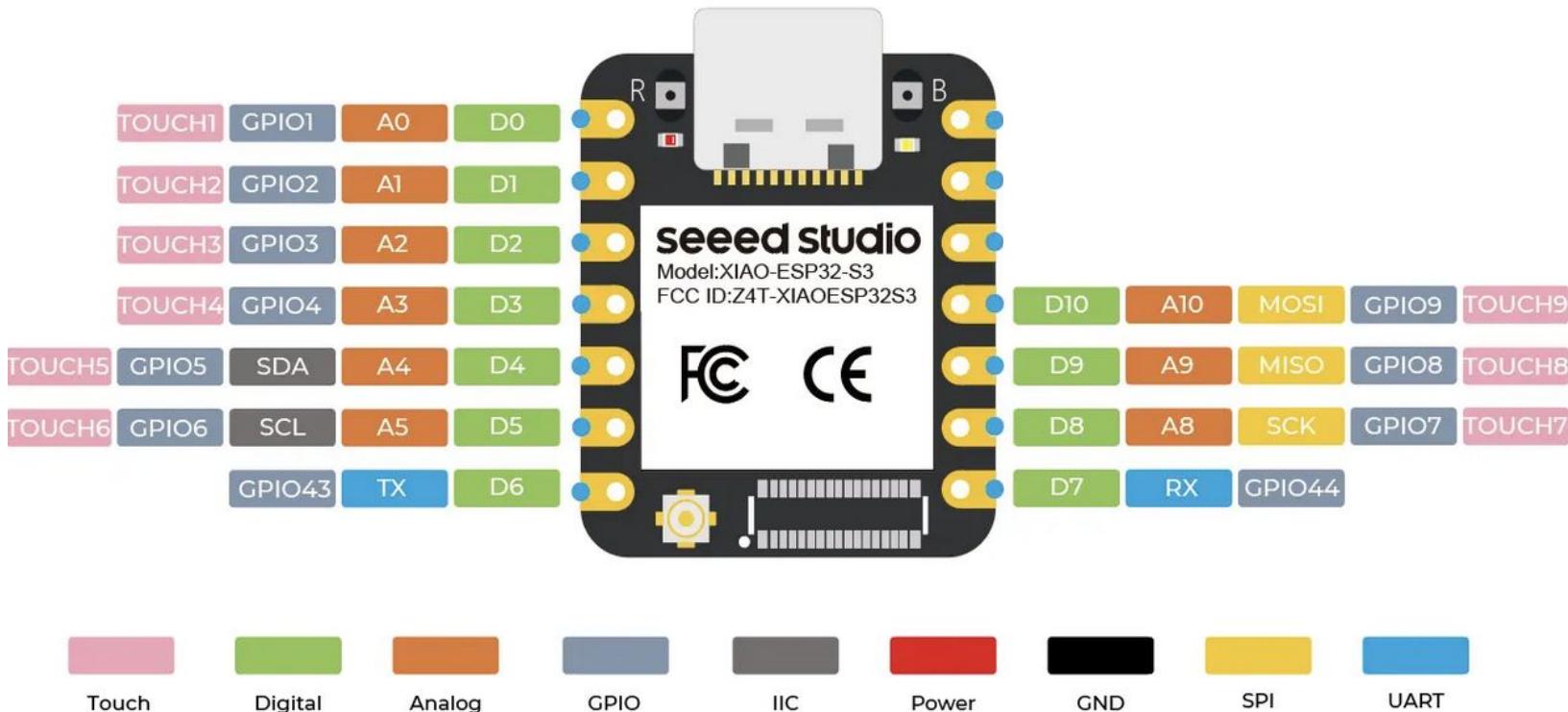
On Off

High Low

# 10 Analog Input pins

## i2C communication SDA/ SCL

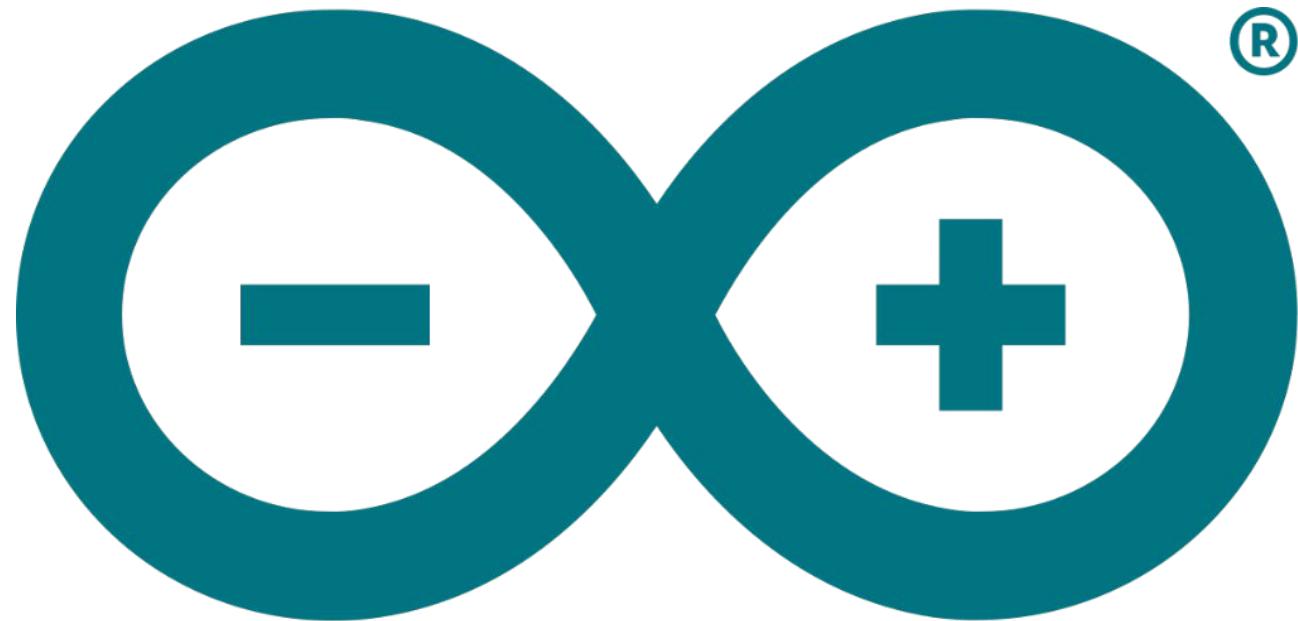
manage all data exchange  
with I2C-compatible  
components



# DATA PINS

XIAO ESP32 S3

## SOFTWARE: Arduino IDE



Open-source electronic prototyping platform enabling users to create interactive electronic objects.

Alternative software: Visual Studio Code with PlatformIO extension



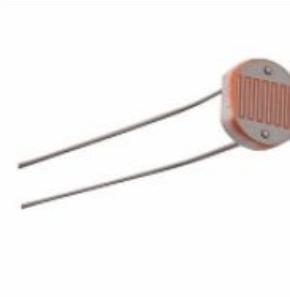
**PlatformIO Core 4**

# SENSORS

Detect events or changes in its environment and send the information to other electronics.

*Can be digital and analog, but most **commonly** are analog (range of values)*

Touch  
Light  
Pressure  
 $T^\circ$   
etc..



# ACTUATORS

Component of a machine that is responsible for moving and controlling a mechanism or system:

*Can be digital and analog, but **most commonly** are digital (0 or 1)*

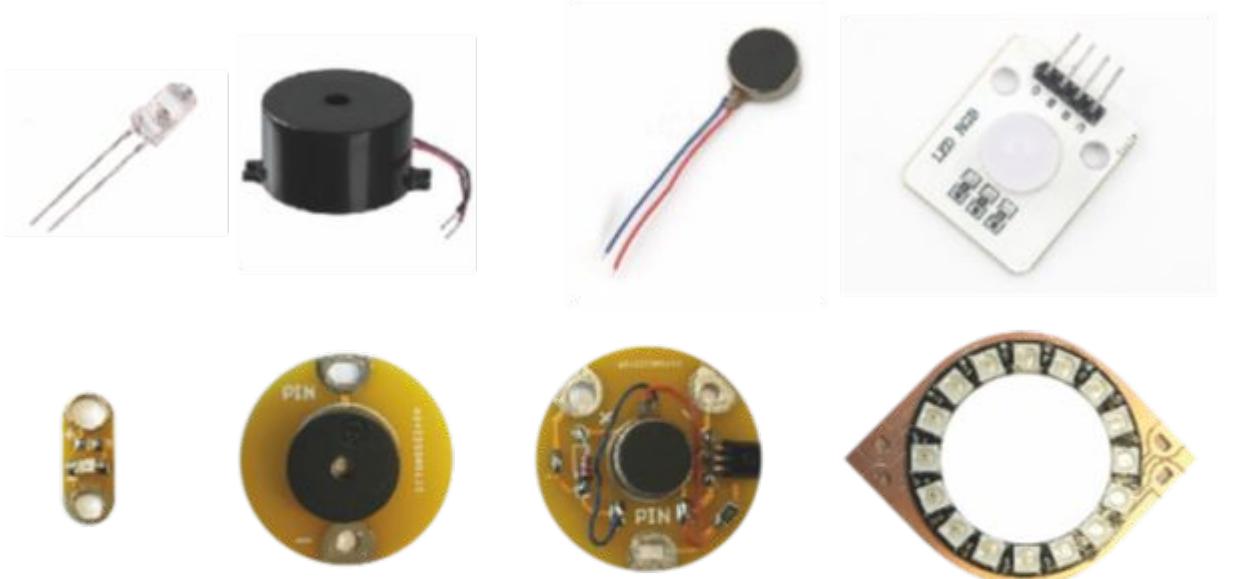
Movement

Sound

Light

Signals

etc..



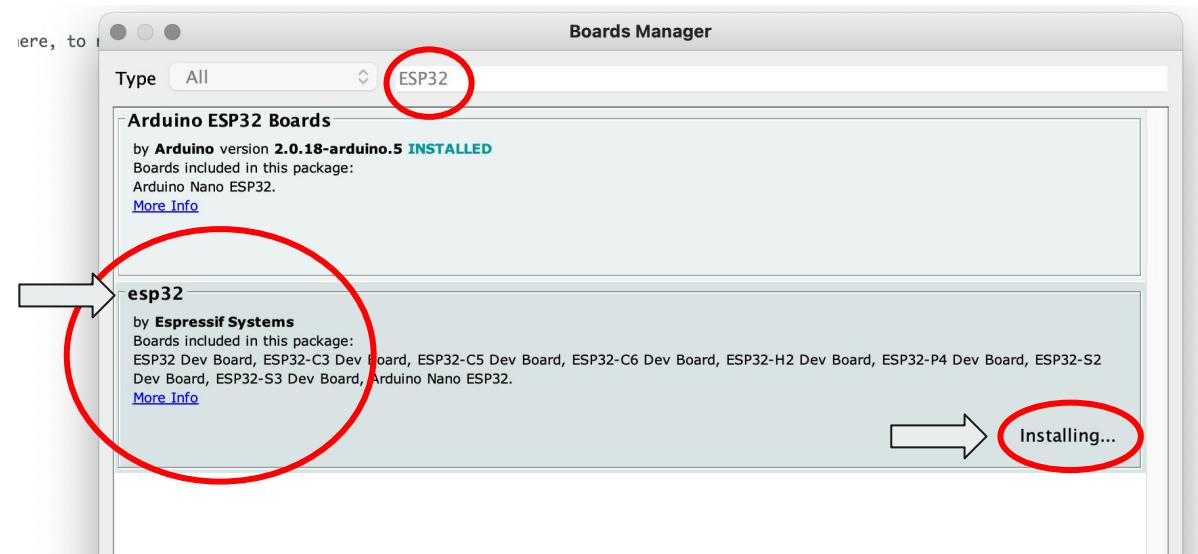
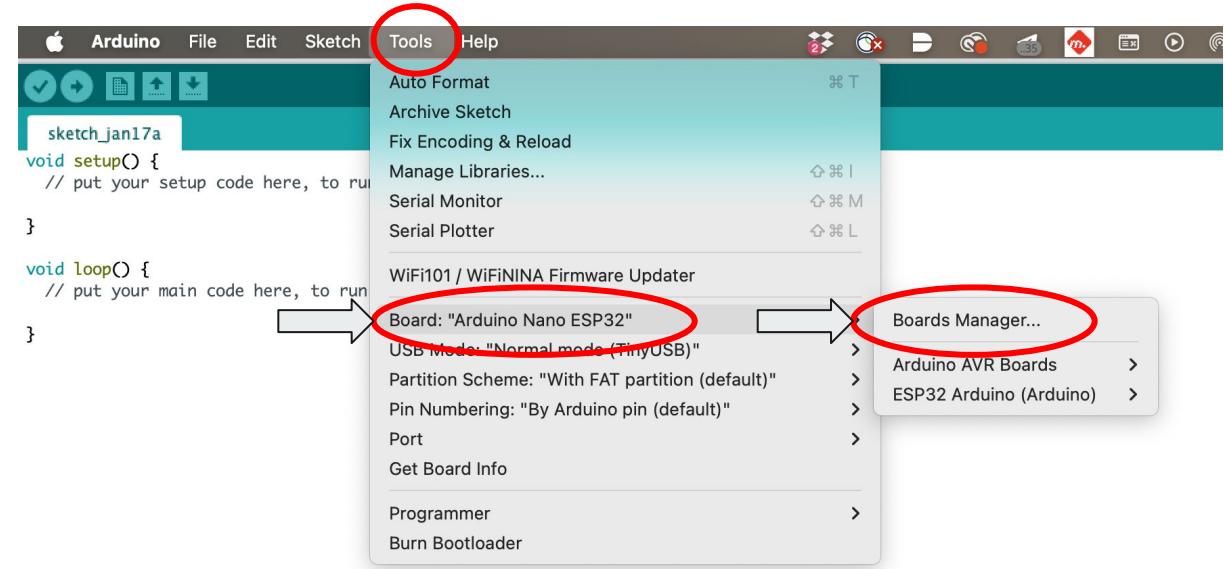
# **GETTING STARTED WITH THE XIAO ESP32 S3**

**Step 1.** Download and Install the stable version of Arduino IDE according to your operating system.

<https://www.arduino.cc/en/software/>

**Step 2.** Launch the Arduino application.

**Step 3.** Add ESP32 board package to Arduino IDE.

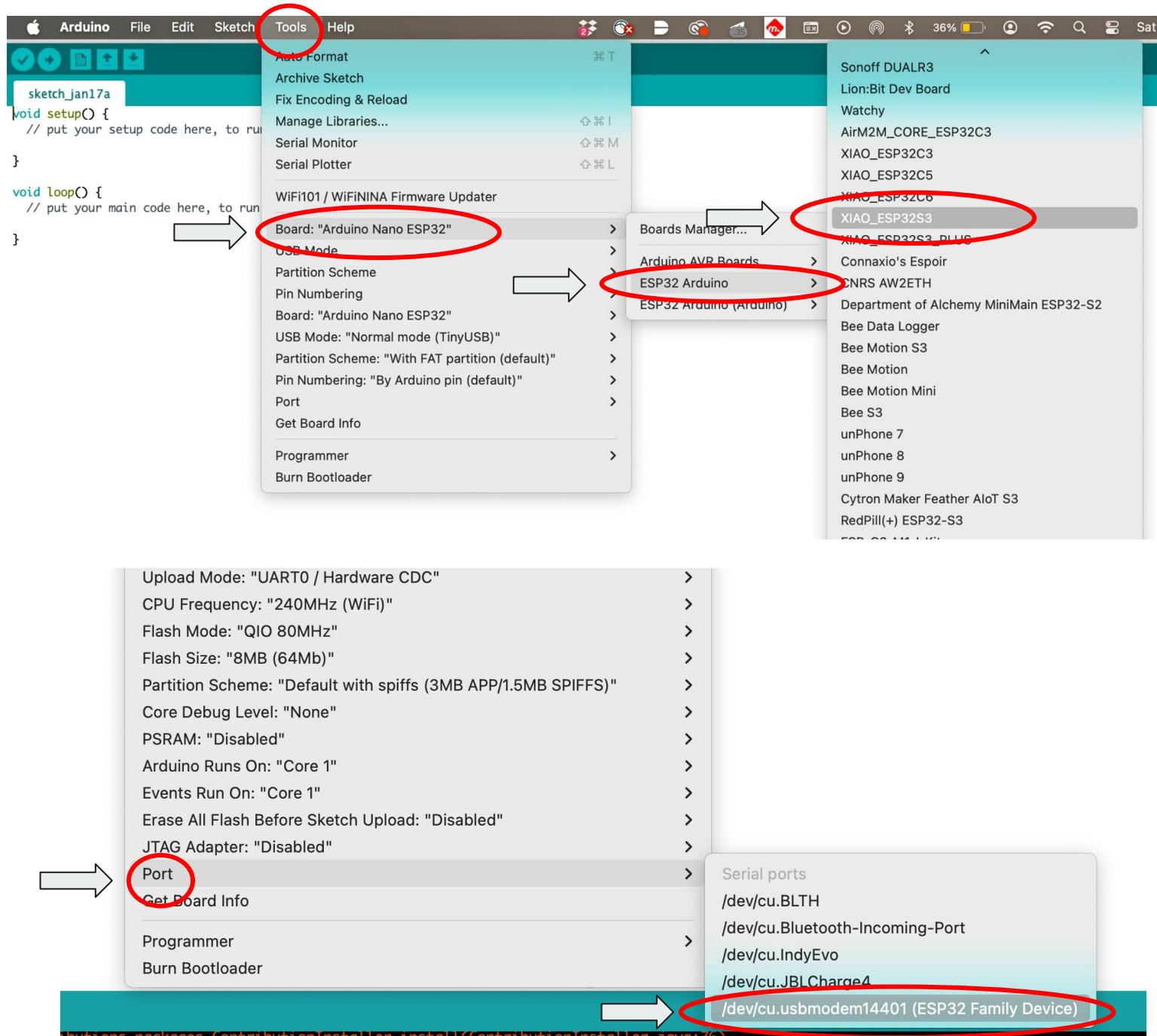


## Step 4. Select your board and port.

At the top of the Arduino IDE you can directly select the port. This will probably be the one with "usbmodem" or "usbserial" in the name.

If you're unsure, unplug and plug again to see which port is missing.

Select **XIAO\_ESP32S3**.



General problems on first use:

- The XIAO is connected to the computer, but *no port number* is found.
- The XIAO is connected, and a port number appears, but the *program upload fails*.

**Use BootLoader Mode:**

[https://wiki.seeedstudio.com/xiao\\_esp32s3\\_getting\\_started/#bootloader-mode](https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/#bootloader-mode)

- **Step 1.** Press and hold the BOOT button on the XIAO ESP32S3 without releasing it.
- **Step 2.** Keep the BOOT button pressed and then connect to the computer via the data cable. Release the BOOT button after connecting to the computer.
- **Step 3.** Upload the **File > Examples > 01.Basics > Blink** program to check the operation of the XIAO ESP32S3.

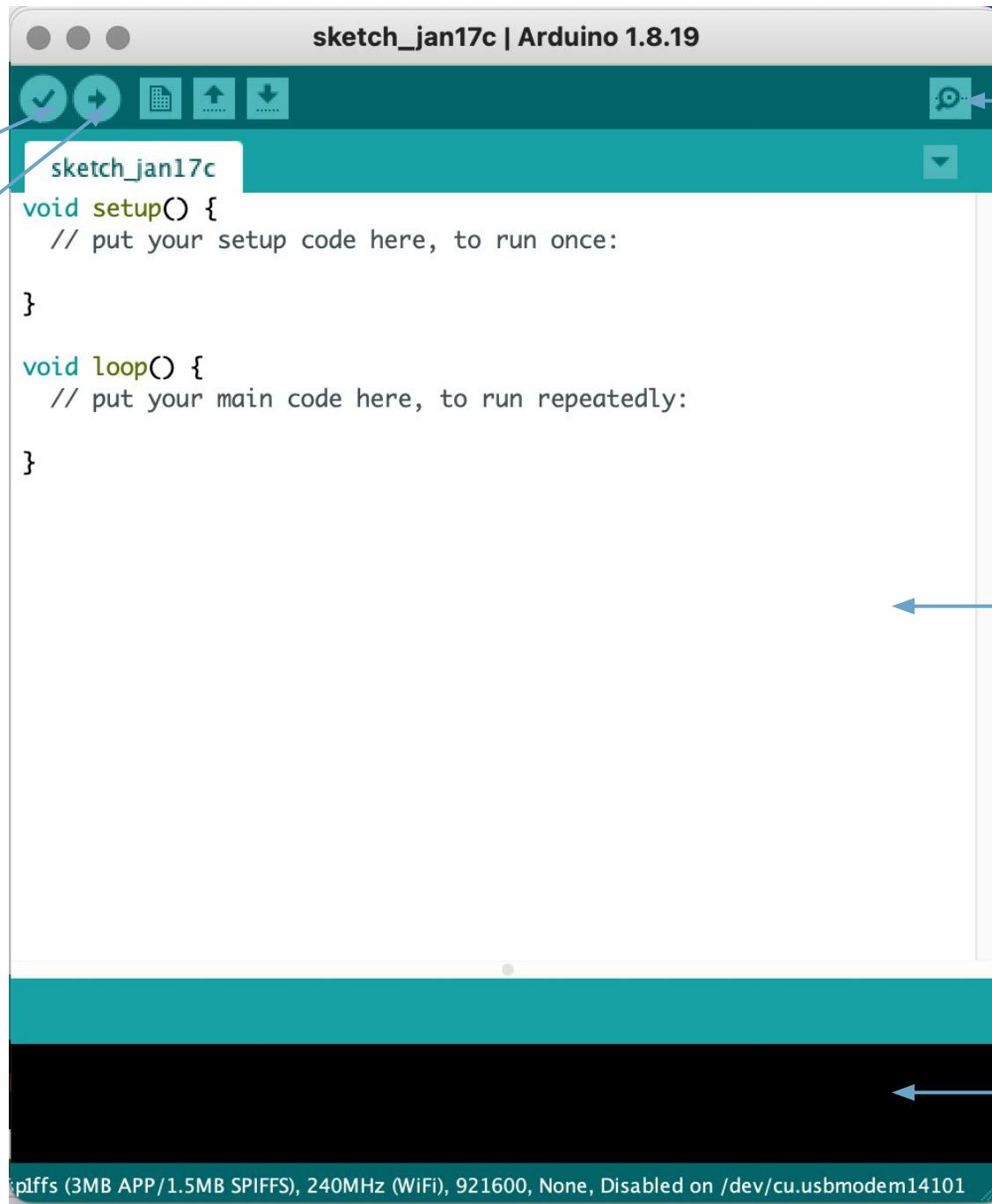
¿How to write code in Arduino?

C++ \*

# ARDUINO IDE

Compile code

Upload code



Serial Monitor

Code lines:  
C++

Errors

The screenshot shows the Arduino IDE interface with the title bar "sketch\_jan17c | Arduino 1.8.19". The code editor contains the following code:

```
sketch_jan17c
void setup() {
  // put your setup code here, to run once:
}

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

Two blue arrows point from the text "Basic functions of the code which control the structure:" to the "setup()" and "loop()" function names in the code.

At the bottom of the IDE, the status bar displays: "SPIFFS (3MB APP/1.5MB SPIFFS), 240MHz (WiFi), 921600, None, Disabled on /dev/cu.usbmodem14101".

Basic functions of the code which control the structure:

**void setup() {}**

Executes just **1** time, after powering up the microcontroller or after resetting it.

**void loop() {}**

Executes **infinite** times

The screenshot shows the Arduino IDE interface with the title bar "example\_serial\_communication | Arduino 1.8.19". The code editor contains the following sketch:

```
int randomValue; ←

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  randomValue =random(100,500);

  Serial.println(randomValue);
  delay(500);
}
```

A status bar at the bottom indicates "Hard resetting via RTS pin...".

**Above setup** is where you can declare **global** (scope) variables.

A variable is a way of naming and storing a value for later use by the program, such as data from a sensor or an intermediate value used in a calculation.

Changes to light blue.  
int is the most common

String destIp;	//multiple characters e.g. IP address
int destPort;	//integer number, can be replaced by new values
const int numReadings = 10;	//constant integer number, can't be replaced
float ValTemp;	//float number, number with decimals
boolean state = true;	//0 or 1 variable, true or false

The screenshot shows the Arduino IDE interface with the title bar "example\_serial\_communication | Arduino 1.8.19". The code editor contains the following sketch:

```
int randomValue;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  randomValue =random(100,500);

  Serial.println(randomValue); ←
  delay(500);
}
```

The line `Serial.println(randomValue);` has a blue arrow pointing to it from the explanatory text below.

At the bottom of the IDE, a status bar displays: "Hard resetting via RTS pin..." and "espffs (3MB APP/1.5MB SPIFFS), 240MHz (WiFi), 921600, None, Disabled on /dev/cu.usbmodem14101".

Segmenting code into functions allows a programmer to create modular pieces of code that perform a defined task and then return to the area of code from which the function was "called".

Changes to colour orange.  
Always follow by ()

## 1) Start serial communication

```
Serial.begin (speed); // e.g. 9600bps;
```

## 2) Assign component to PIN number and decide if it is input or output

```
pinMode (pin number, OUTPUT or INPUT);
```

## 3) Give it some time before it executes the next line of code

```
delay (time in milliseconds); //1000 = 1 second
```

## 4) Apply 5V to a particular Pin

```
digitalWrite (pin number, HIGH or LOW); // if actuator is digital
```

```
analogWrite (pin number, 0 to 255); // if actuator is analog
```

## 5) Read incoming data from digital or analog sensor

```
digitalRead (pin number);
```

```
analogRead (pin number);
```

## 6) Write to Serial Monitor

```
Serial.println ("message");
```

```
Serial.println (variable);
```

## 7) Map incoming values from one range to another

```
map (value, min original, max original, min new, max new);
```

# SOME IMPORTANT FUNCTIONS

Blink | Arduino 1.8.19

```
24
25 // the setup function runs once when you press reset or power the
26 void setup() {
27
28   Serial.begin(9600);
29
30   // initialize digital pin LED_BUILTIN as an output.
31   pinMode(LED_BUILTIN, OUTPUT);
32 }
33
34 // the loop function runs over and over again forever
35 void loop() {
36   digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is
37   // on)
38   Serial.println("LED on!");
39
40   delay(1000);                      // wait for a second
41   digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making
42   // it a low signal
43   Serial.println("LED off!");
44
45   delay(1000);                      // wait for a second
46 }
```

/dev/cu.usbmodem35775501 (Teensy) Serial

```
LED on!
LED off!
```

Autoscroll      Newline      Clear output

# SERIAL MONITOR

YOUR FIRST SKETCH

Use the serial monitor to send and receive messages from Arduino. For example, we can use it to read sensor values and debug.

```
Blink | Arduino 1.8.19

Blink §

24
25 // the setup function runs once when you press reset or power the
26 void setup() {
27
28   Serial.begin(9600); ←
29
30   // initialize digital pin LED_BUILTIN as an output.
31   pinMode(LED_BUILTIN, OUTPUT);
32 }
33
34 // the loop function runs over and over again forever
35 void loop() {
36   digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is
37
38   Serial.println("LED on!"); ←
39
40   delay(1000);           // wait for a second
41   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making it
42
43   Serial.println("LED off!"); ←
44
45   delay(1000);           // wait for a second
46 }
```

**Serial.begin(baud rate);** in setup

**Serial.println("String");** in loop

**Serial.println(variable);** in loop

# SERIAL MONITOR

YOUR FIRST SKETCH

**PROGRAMMING WITH  
ARDUINO IDE / ESP32 S3  
(C/C++)**

## INSTRUCTIONS

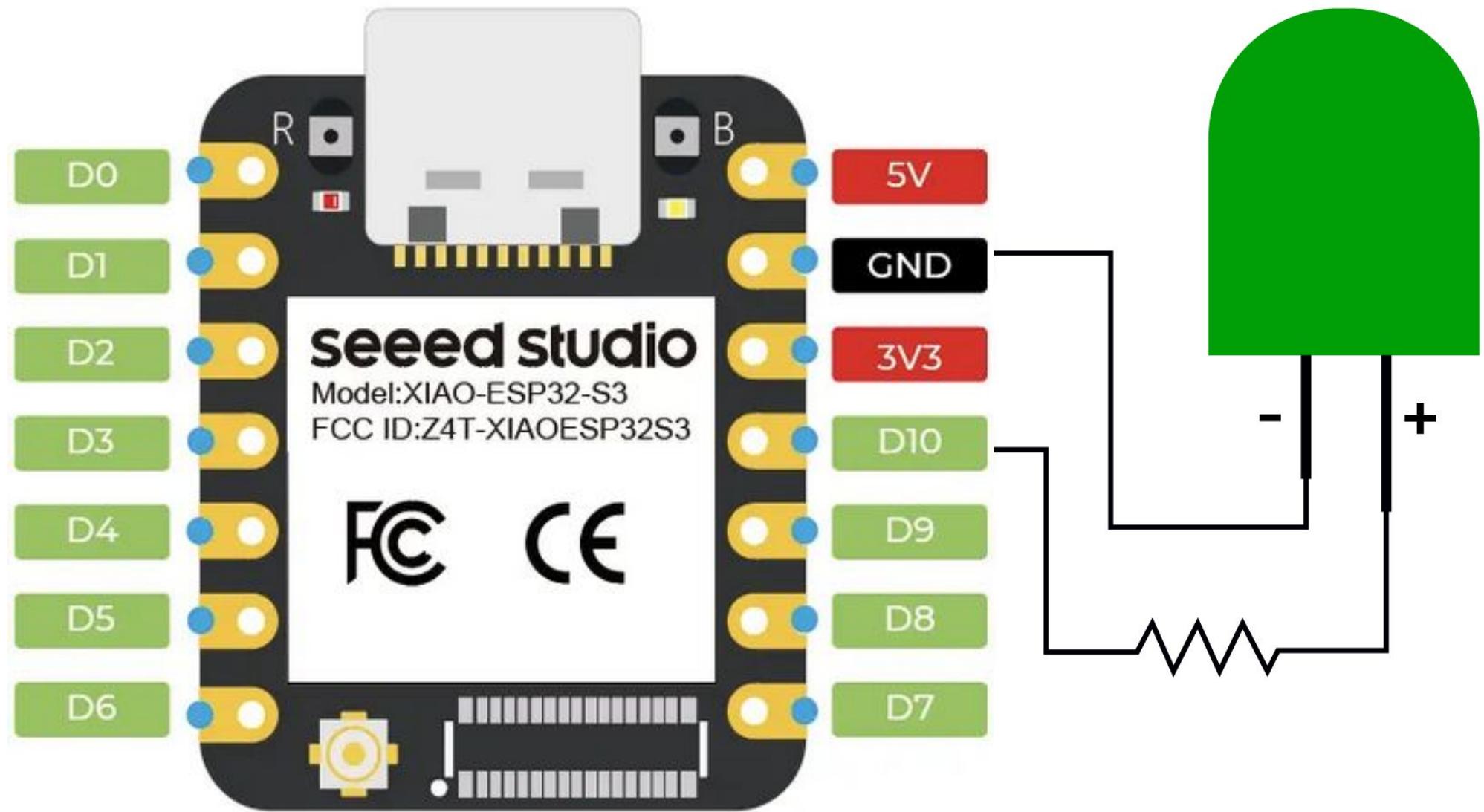
**IF** sensor is pressed/ stretched  
**TURN ON LED**  
**ELSE**  
**TURN OFF LED**

## ELSE..IF

IF (x) is true, then (y) occurs, otherwise (z) occurs.  
meaning...

```
if (condition x = true) {  
    //action A happens  
}  
else {  
    //action B happens  
}
```

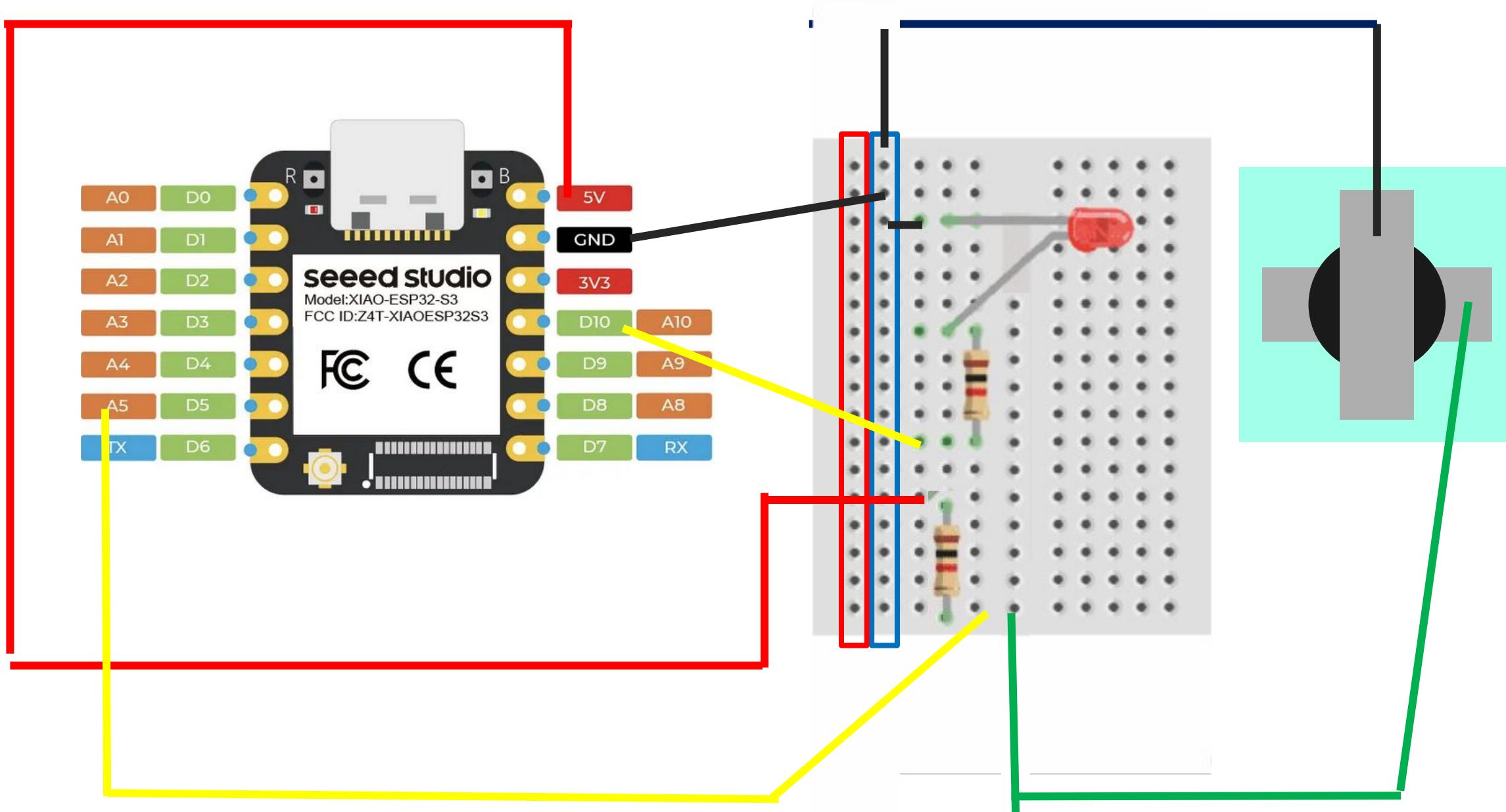
## Connecting an LED



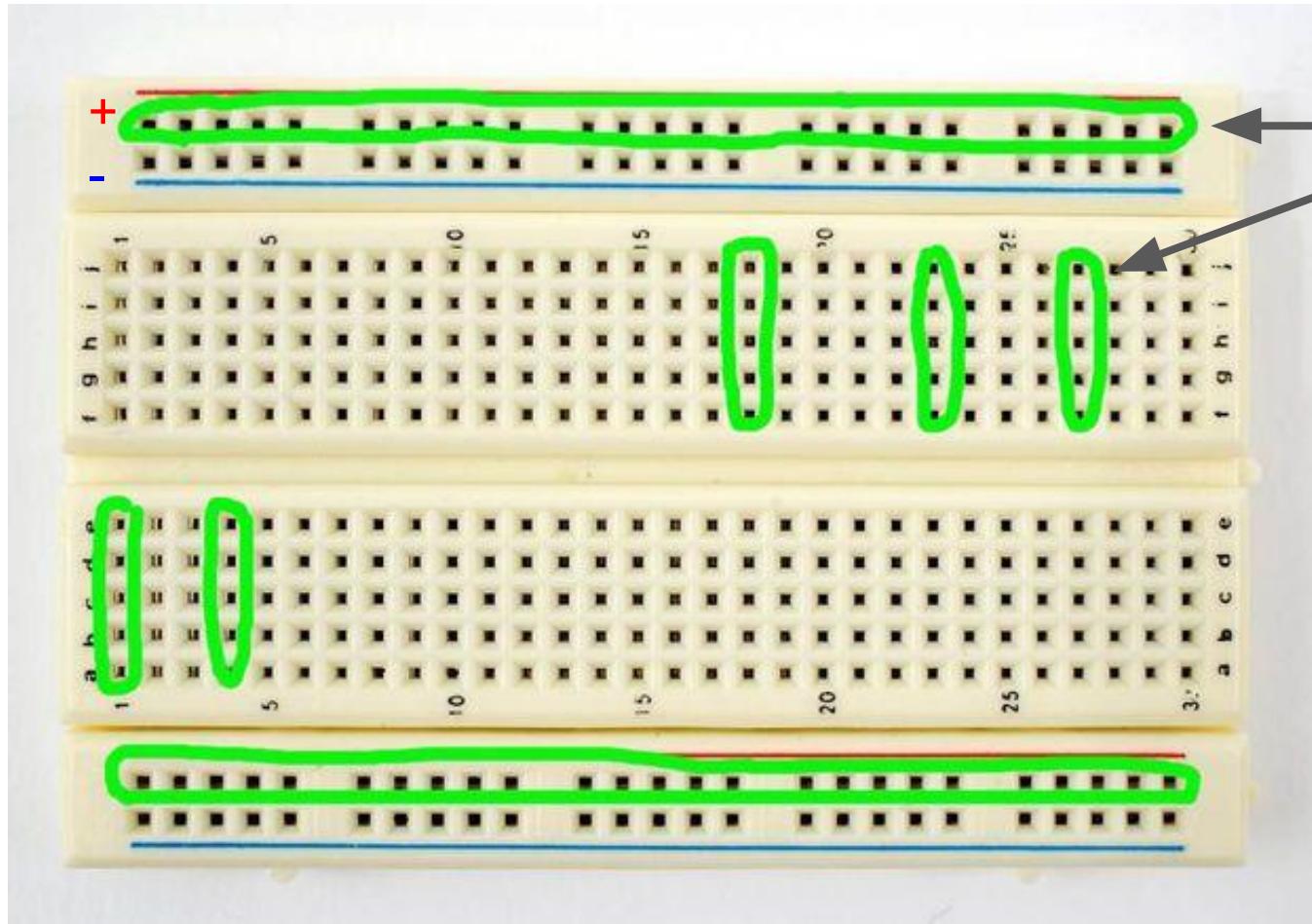
# RESISTIVE SENSOR



## RESISTIVE SENSOR + LED



# Bread board



# Let's code

1. Install Arduino IDE + ESP32 board package
2. Choose correct board
3. Choose correct port
4. Find the example code for 'blink'
5. Upload code and test

6. Connect one sensor and LED
7. Copy and write code (if-statement)
8. Upload code and test

**IF** button is pressed  
**TURN ON LED**  
**ELSE**  
**TURN OFF LED**

Extra: pair up with someone who is ready and set up two sensors. (**same wiring principle as one sensor**)

>What do you need to add to the Arduino code to get two data points?

>Can you add other conditions for your if-statement?

# For next week

During next tutorial we will learn how to collect data by WiFi (require Arduino IDE and python environment)

Link: <https://realpython.com/installing-python/#reader-comments>

Video: <https://www.youtube.com/watch?v=QhukcScB9W0>

IDEs: PyCharm, Visual Studio Code .....

## **Tutorial ESP32-S3 II**

using Arduino IDE software

## About me

Yao Zhang

2nd year phd student in Smart Wearable Lab, focusing on AI about human motion understanding

Supervised by Prof. Xiao Yu

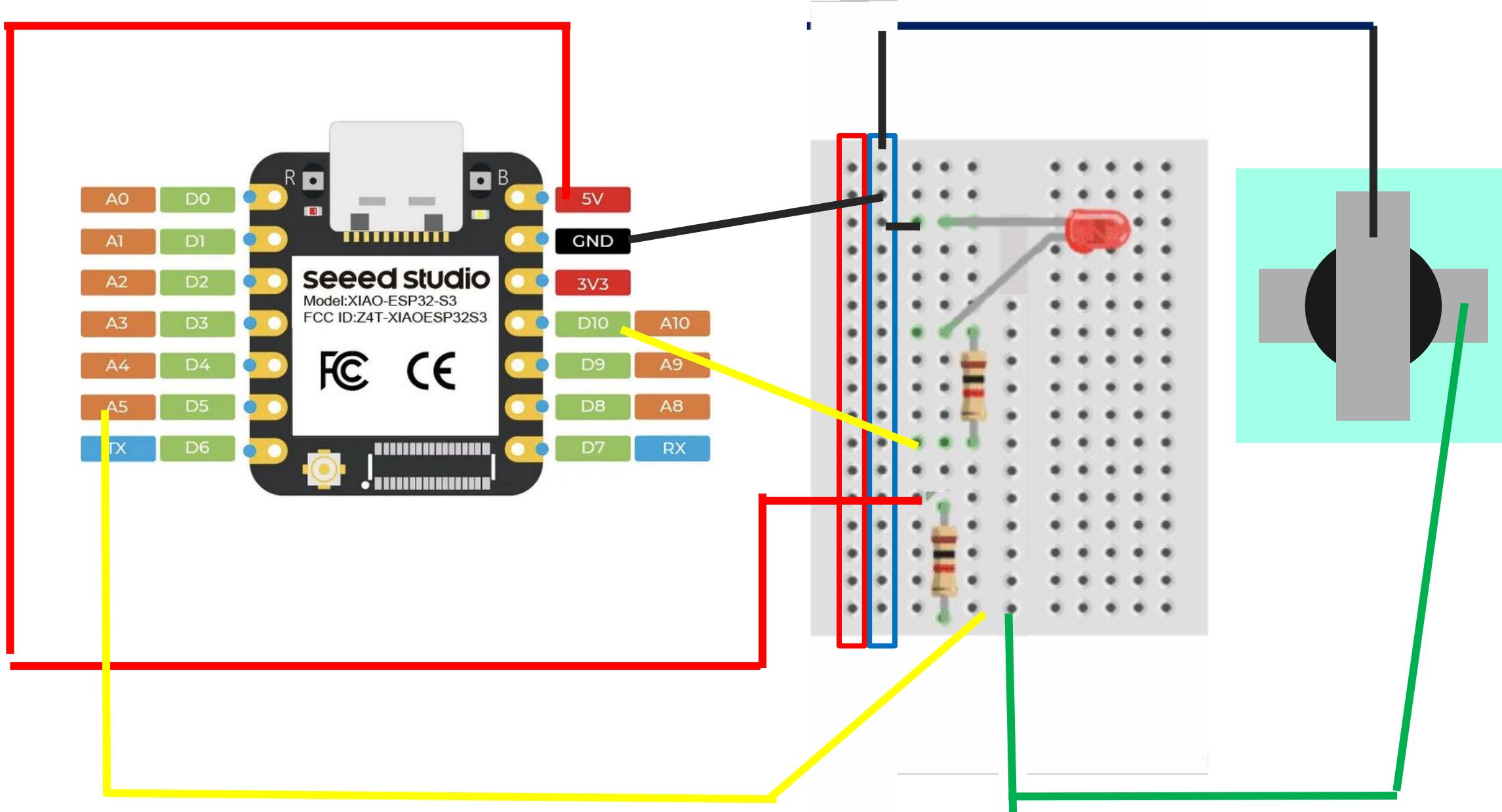
Course assistant for II part of smart wearable course

[yao.1.zhang@aalto.fi](mailto:yao.1.zhang@aalto.fi)

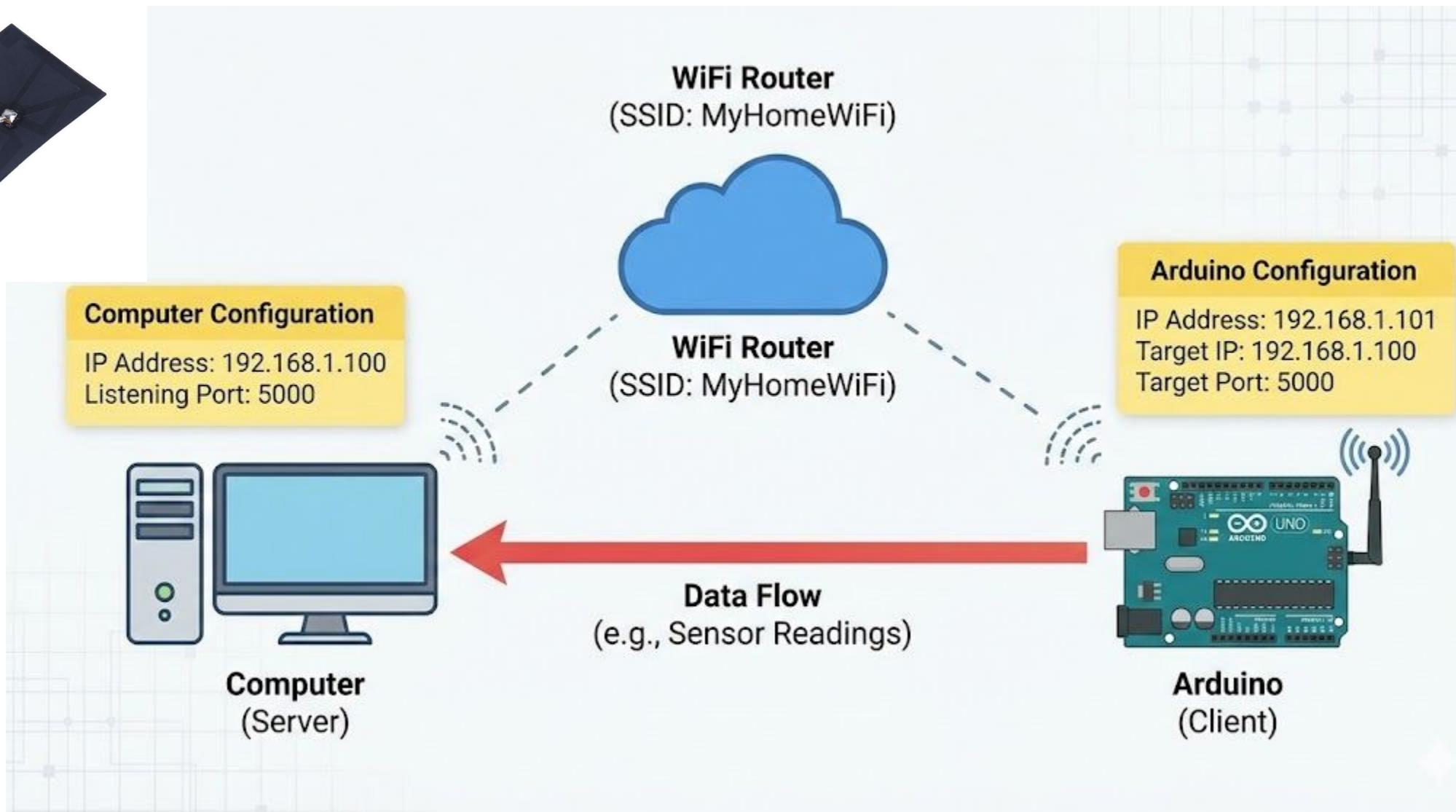
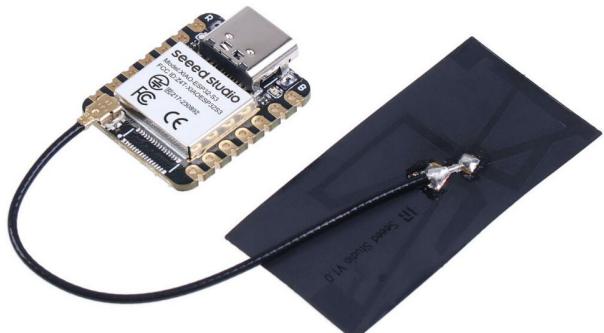
# Learning Outcomes

- > Introduction to sensor data collection
- > Basics about python environment
- > How to collect data through WiFi

## Last week: resistive sensor + LED



## WiFi module of ESP32



## How to use WiFi module

Important things:

1. connect Arduino(sender) and your computer(server) to same WiFi(like: your phone hotspot).
2. changing your Arduino code with your own WiFi id and password.
3. get server IP after you connect your computer to the same WiFi.
4. keep Port number as same for both sender and server part.

```
// WiFi credentials
const char* ssid = "zyyyyy";      // Replace with your WiFi SSID
const char* password = "zy561526"; // Replace with your WiFi password

// Server details (your computer's IP and port)
const char* serverIP = "172.20.10.14"; // Replace with your computer's local IP
const int serverPort = 5000; // Must match the Python script
```

# Server (our computer)

IDEs:

PyCharm

VS code

...

The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar displays a project structure for 'code\_arduino2' containing 'code\_arduino2', 'read\_serial', 'saving\_data', and 'server\_saving\_data.py'. The 'server\_saving\_data.py' file is open in the main editor, showing Python code for a TCP server that reads serial data and saves it to a CSV file. The code uses `socket`, `csv`, `datetime`, and `joblib` modules. The terminal below shows the server's response to incoming connections, with messages like 'Received: 2741,' and 'Server stopped.' The status bar at the bottom indicates the file is saved, the encoding is CRLF, and the file size is 4 spaces.

```
1 import socket
2 import csv
3 import datetime
4 import joblib
5
6
7 # Server settings
8 HOST = "0.0.0.0" # Listens on all available interfaces
9 PORT = 5000 # Must match Arduino's serverPort
10 # print(datetime.datetime.now())
11 # Create a socket (IPv4, TCP)
12 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13 server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, value: 1) # Allow reusing the same port
14 server_socket.bind((HOST, PORT))
15 server_socket.listen(1)
16 print(f"Listening for connections on port {PORT}...")
17
18 # Accept connection
19 client_socket, client_address = server_socket.accept()
20 print(f"Connected to {client_address}")
21
22 # Open CSV file for writing data
```

Run server\_saving\_data

Received: 2741,  
Received: 2704,  
Received: 2717,  
Received: 2733,  
Received: 2736,  
Server stopped.

Process finished with exit code 0

code\_arduino2 > server\_saving\_data.py

5:1 CRLF UTF-8 4 spaces smart\_wearable

## Python

## Interpreter(environment)

Settings

Project: code\_arduino2 > Python Interpreter

Python Interpreter: smart\_wearable C:/Users/yao zhang/miniconda3/envs/smart\_wearable/python.exe Add Interpreter

Package	Version	Latest version
anyio	4.8.0	
argon2-cffi	23.1.0	
argon2-cffi-bindings	21.2.0	
arrow	1.3.0	
asttokens	3.0.0	
async-lru	2.0.4	
attrs	25.1.0	
babel	2.17.0	
beautifulsoup4	4.13.3	
blas	1.0	
bleach	6.2.0	
brotlicffi	1.0.9.2	
bzip2	1.0.8	
ca-certificates	2025.9.9	
certifi	2025.1.31	
cffi	1.17.1	
charset-normalizer	3.4.1	
colorama	0.4.6	
comm	0.2.2	
contourpy	1.3.1	
cycler	0.12.1	
debugpy	1.8.12	

OK Cancel Apply

## How to get IP

<https://www.avast.com/c-how-to-find-ip-address>

For Windows: ipconfig

```
Command Prompt
Microsoft Windows [Version 10.0.16299.248]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\dan>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

  Connection-specific DNS Suffix  . : localdomain
  Site-local IPv6 Address . . . . . : fec0::fea9:b10d:f2f1:15bb:74c3%1
  Link-local IPv6 Address       . . . . . : fe80::b10d:f2f1:15bb:74c3%3
  IPv4 Address. . . . . : 10.211.55.4
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : fe80::21c:42ff:fe00:18%3
                                10.211.55.1

Tunnel adapter Teredo Tunneling Pseudo-Interface:

  Connection-specific DNS Suffix  . :
  IPv6 Address. . . . . . . . . . : 2001:0:34f1:8072:1083:a5a:f52c:c8fb
  Link-local IPv6 Address . . . . . : fe80::1083:a5a:f52c:c8fb%5
  Default Gateway . . . . . . . . . : ::

C:\Users\dan>
```

For Mac: ifconfig | grep "inet"

## Collecting data

```
#include <WiFi.h>

// WiFi credentials
const char* ssid = "zyyyyy";      // Replace with your WiFi SSID
const char* password = "zy561526"; // Replace with your WiFi password

// Server details (your computer's IP and port)
const char* serverIP = "172.20.10.6"; // Replace with your computer's local IP, the
const int serverPort = 5000; // Must match the server Python script

int sensorPin1 = A5;
int sensorValue1 = 0;
//const int buttonPin = 2;

WiFiClient client;

void setup() {
    Serial.begin(115200);
    // comment when no serial
    // while (!Serial);

    // 1. start connection
    Serial.print("Connecting to WiFi...");
    WiFi.begin(ssid, password);

    // 2. check status
    while (WiFi.status() != WL_CONNECTED) {
```

```
Listening for connections on port 5000
Connected to ('172.20.10.2', 5000)
Received: 15
Received: 16
Received: 15
Received: 15
Received: 16
Received: 16
Received: 15
Received: 15
Received: 14
Received: 16
Received: 16
Received: 14
Received: 16
```

```
# Server settings
HOST = "0.0.0.0" # Listens on all available interfaces
PORT = 5000 # Must match Arduino's serverPort
# print(datetime.datetime.now())
# Create a socket (IPv4, TCP)
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, value: 1) # Allow
server_socket.bind((HOST, PORT))
server_socket.listen(1)
print(f"Listening for connections on port {PORT}...")

# Accept connection
client_socket, client_address = server_socket.accept()
print(f"Connected to {client_address}")

# Open CSV file for writing data
csv_filename = "./saving_data/training_sensor_data_1.csv"
with open(csv_filename, mode="w", newline="") as file:
    writer = csv.writer(file)
    writer.writerow(["Timestamp", "Sensor Value", "label"]) # Write header

    try:
```