

IoT and Sensors

The Internet, Things;
touching you, touching me.



Welcome!

Your Instructor: Ryan (Ryan cosplays as a hardware engineer). *Slides are not his strong point.*

What is happening: This is a IoT quickstart course (with a focus on hardware/code *with extra sauce*).

Goal: ~understand the hardware building blocks so you can DIY a solution vs paying \$\$\$\$ for a proprietary thing that loses support in 3 years.

Is there lunch? Yes

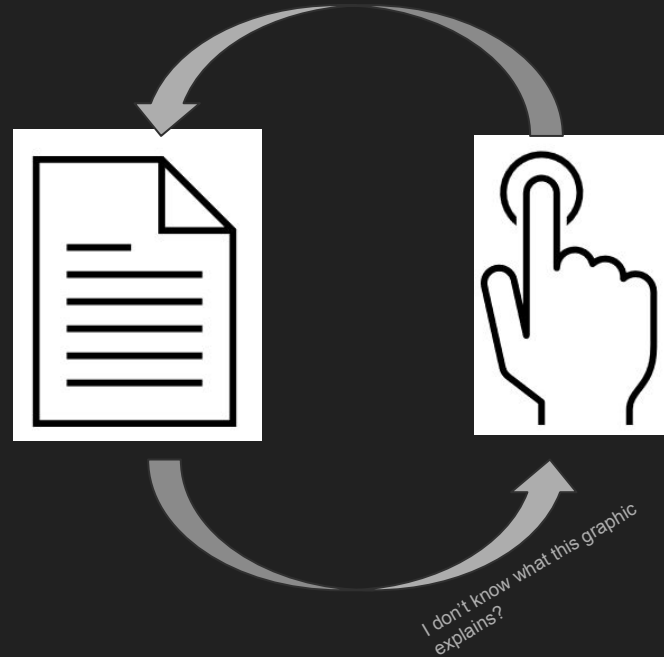
Building FAQs: Coffee/ bathroom locations.

Why are you here? *warning! possible ice breaker.*

Ask questions any time! *Dumb questions are encouraged.*

[Learning] Road Map for Today:

- 1: What is IoT?
- 2: Exploring IoT Hardware
- 3: IoT Hardware and Sensor Integration
- 4: IoT Communication with MQTT
- 5: IoT Network Layer
- 6: Applications of IoT
- 7: Sensors in Manufacturing Automation
- 8: open-ended code funtime ?



[Skill] Road Map

Some topics might
seem confusing.
That's ok!

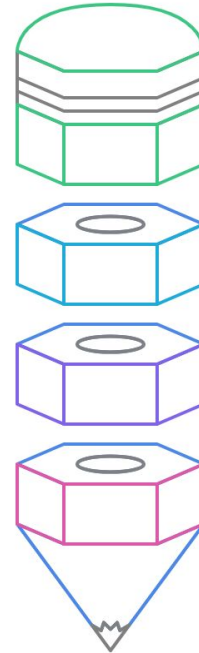
Electrical and Electronic Engineering

Involves the design and maintenance of circuits and devices essential for IoT systems.



Computer Programming

Focuses on writing software that enables devices to communicate and perform tasks.



Computer Science

Provides the theoretical foundation and algorithms necessary for data processing and system integration.



Internet Technologies

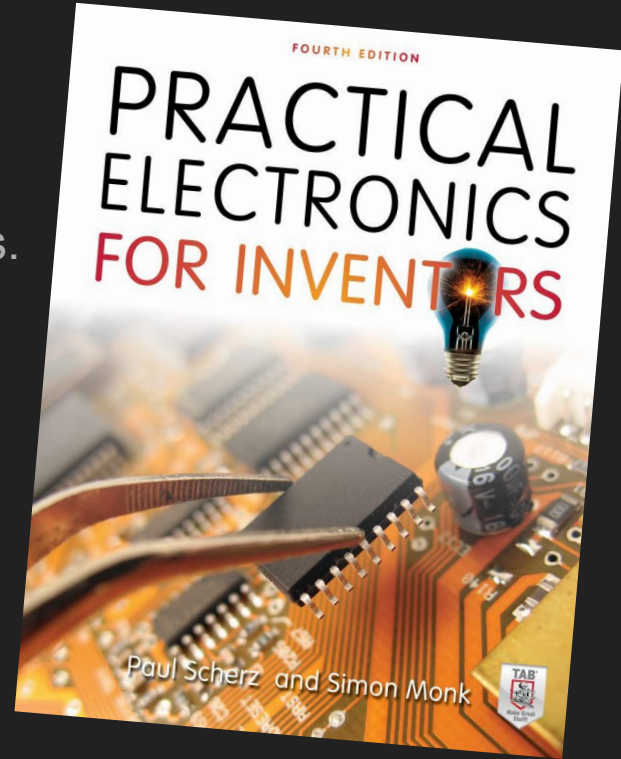
Covers the protocols and infrastructure needed for data transmission and connectivity.

[Skill] Road Map

... just the knowledge you need
to push forward through this course's topics.

Useful knowledge comes from experience.
You have to build stuff.

Relax. Have fun, be curious.



IoT Goals



Understand the benefits.

Grasp some technical fundamentals of:

Computer Science: Memory, Processing,

IT: Networking basics (*relax, nothing obscure*)

Hardware: Microcontrollers, SoC, Sensors

Software: C/C++, Arduino IDE

ChatGPT: it writes and explains code for you.

MQTT: Publisher Subscriber Model

Play with hardware: Monitor Temperature over WiFi, toggle a relay.

Internet of Things

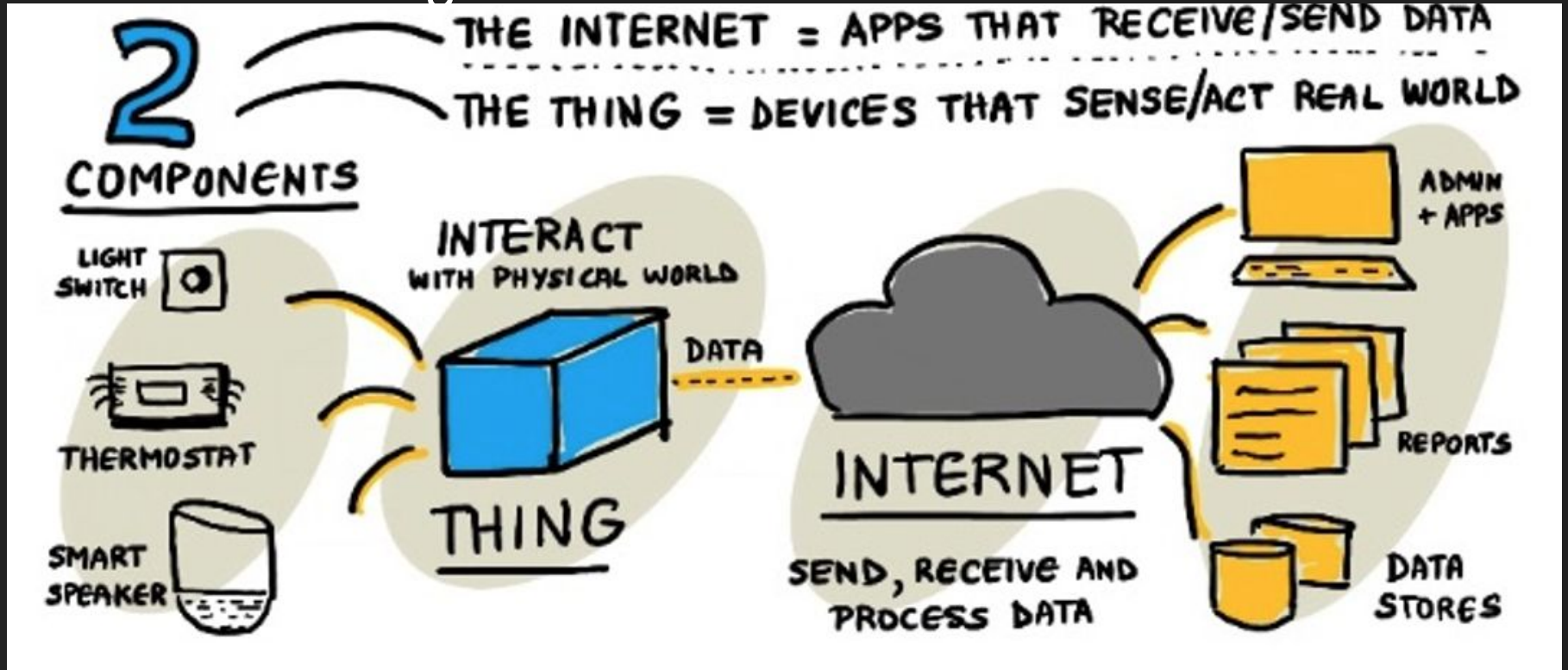
Definition, Scope, Importance.

The term 'Internet of Things' was coined by Kevin Ashton in 1999, to refer to connecting the Internet to the physical world via sensors.

Connect the physical world to the internet using sensors.

Since then, the term has been used to describe any device that interacts with the physical world around it, either by gathering data from sensors, or providing real-world interactions via actuators (devices that turn on a switch or light an LED), generally connected to other devices or the Internet.

Internet of Things



Internet of Things

Definition, Scope, Importance.

Examples!



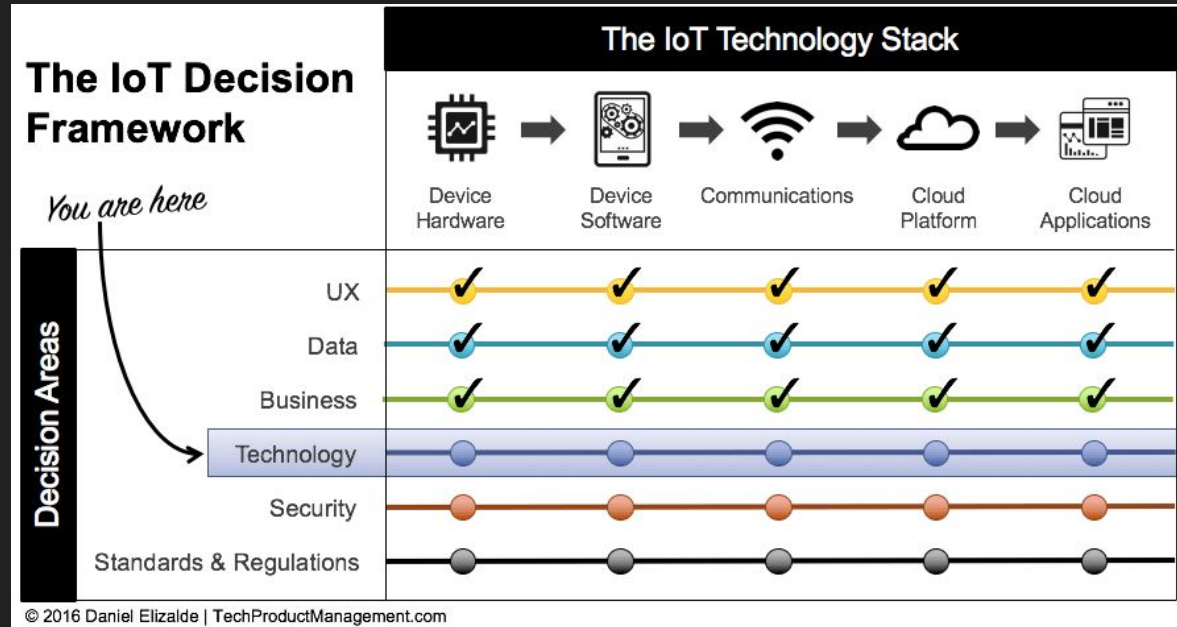
Internet of Things

Scope, Importance.

CONNECT ☒

GATHER DATA ☒

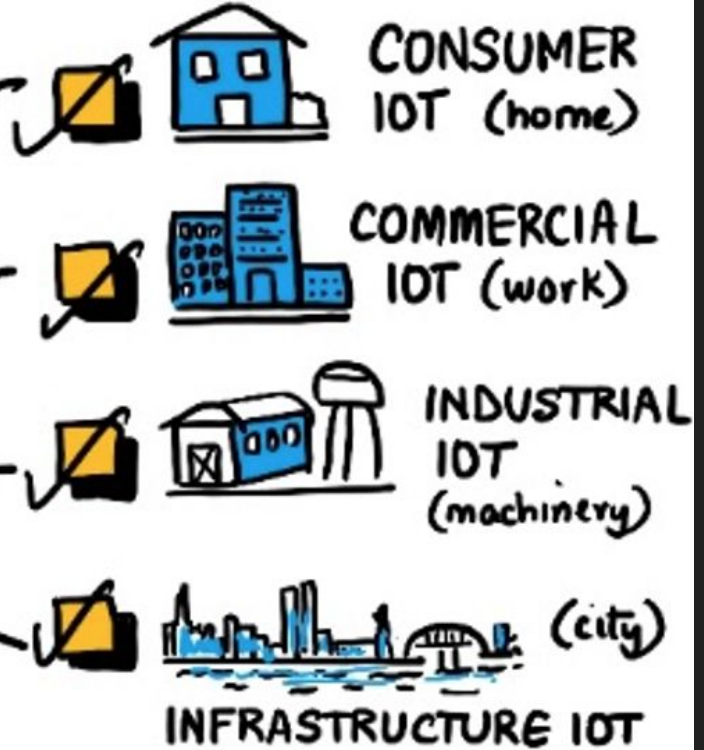
INTERACT ☒



Internet of Things

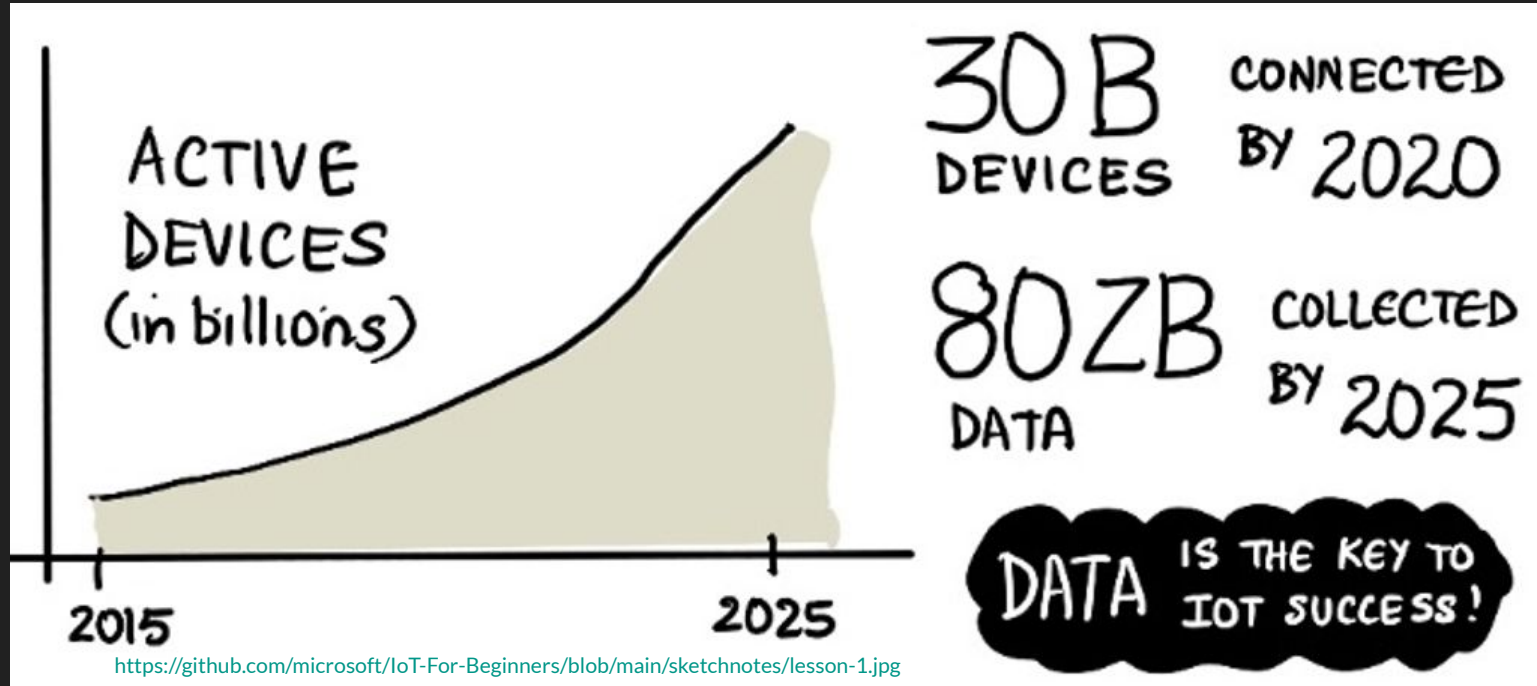
HUGE RANGE OF USE
CASES FOR IOT APPS

4 BROAD
GROUPS



Internet of Things

Definition, **Scope**, Importance.



Internet of Things

Definition, Scope, **Importance**.

1. Enhanced Efficiency and Automation
2. Improved Data Collection and Insights (Real-Time Decision Making)
3. Cost Savings
4. Better Health and Safety
5. Enhanced Connectivity and Convenience

PRO TIPS:

WHATEVER DATA YOU COLLECT, IS IT USEFUL?
BE COST CONSCIOUS ON SCALABILITY


Sh*tternet of Things

Practicality... Usefulness...
Should we? vs. Could we?



David Barnard  
@drbarnard · Follow



 @GE_Appliances won't let me use convection roast on my new oven without connecting the oven to wifi. After a string of profanities I reluctantly downloaded the app and sat on the kitchen floor 10+ minutes trying to get it connected... unsuccessfully. The app is complete garbage

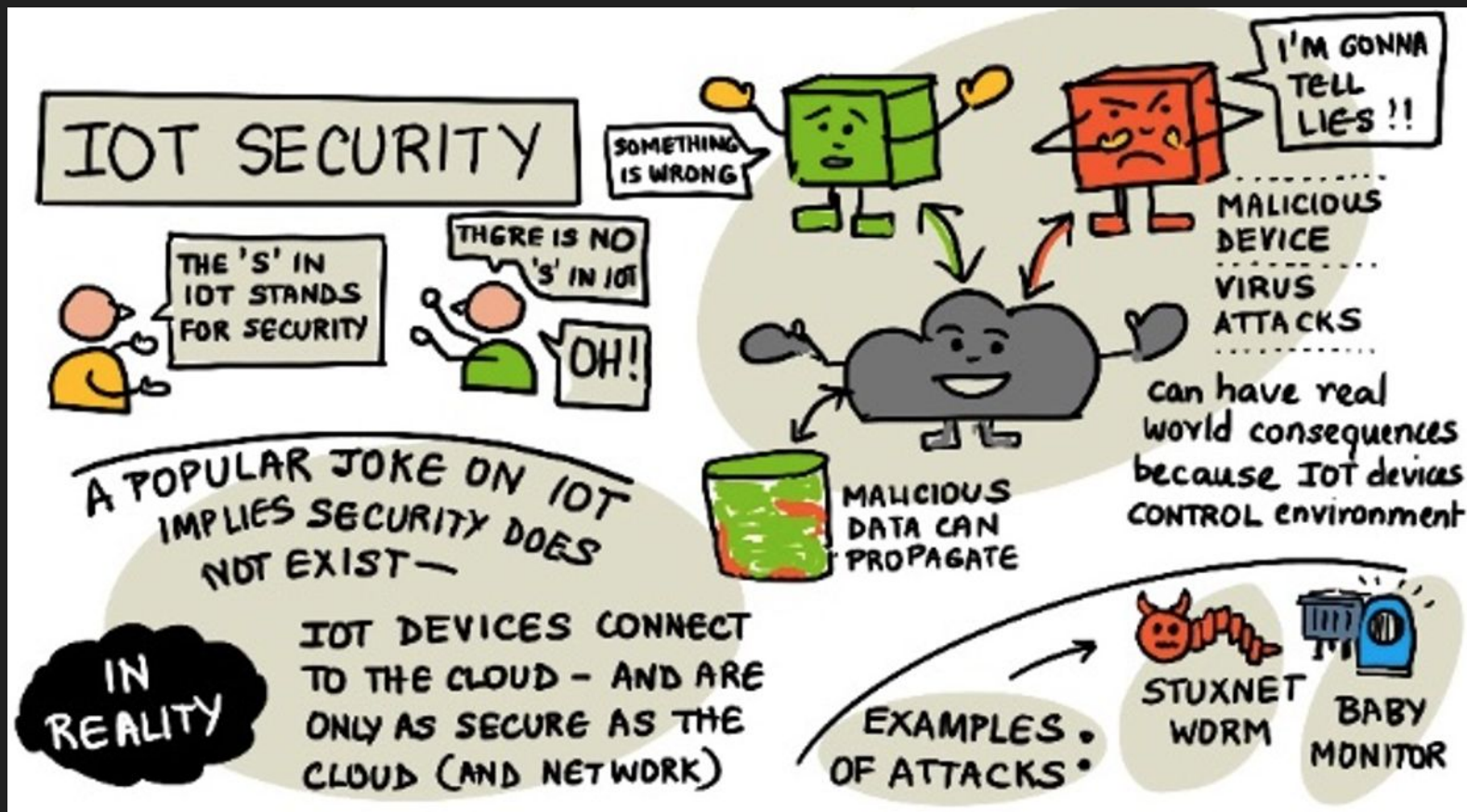


11:24 AM · Mar 5, 2022



 2.7K  Reply  Copy link

[Read 210 replies](#)



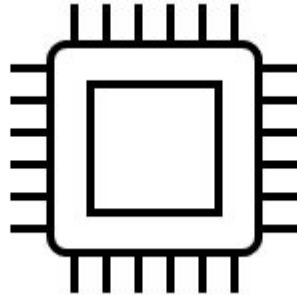
IoT Hardware

Sensors.



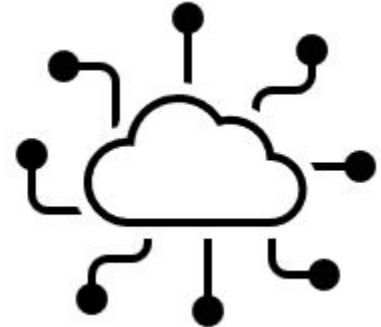
68F

Things.



Publish: "Room is 68"

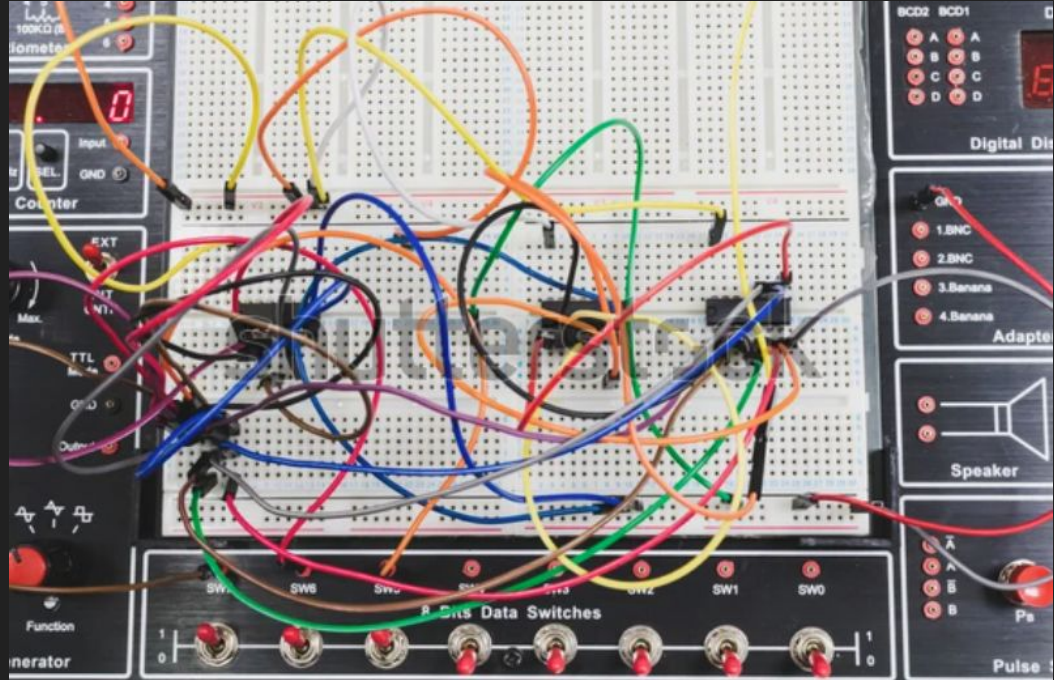
Internet.



Notice Room under temp.
Action: Heater = ON.

IoT Hardware (custom?)

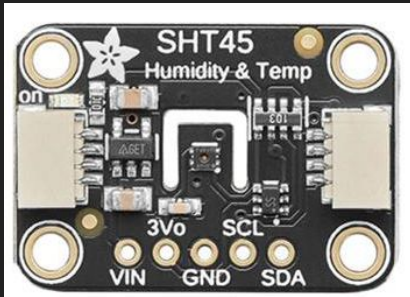
Hardware is easy?
But in the classroom,
simple/reliable wins.



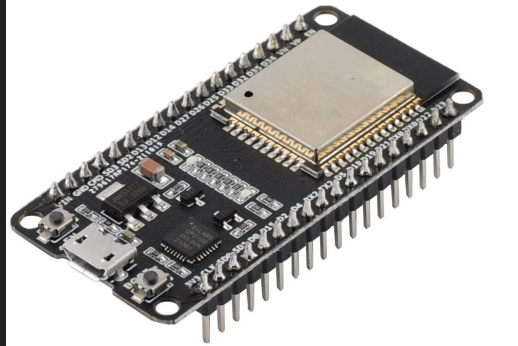
IoT Hardware (what we are using)

Sense the physical world.
Record Data, Action

Handle Data, publish to Internet.



68F

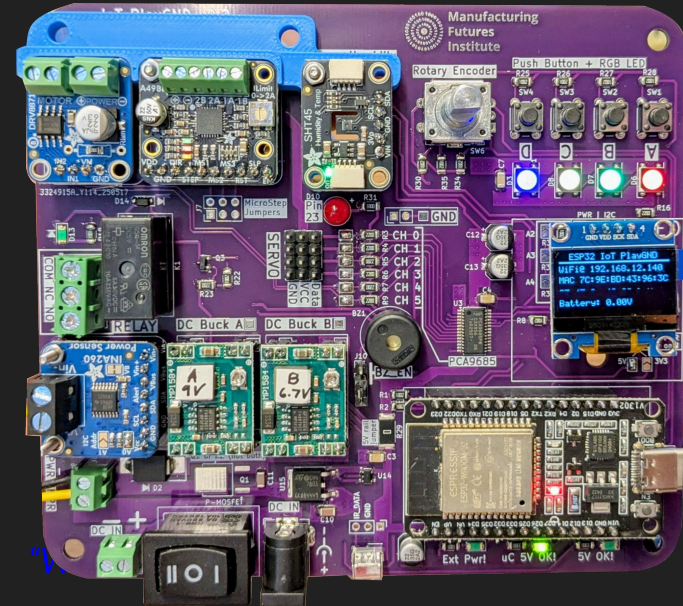
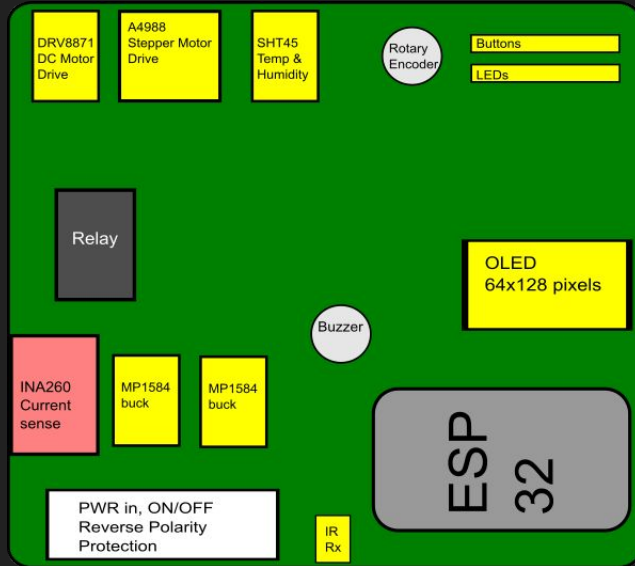


Publish: ["Room is 68"]



Notice Room under Temp.
Action: Heater = ON.

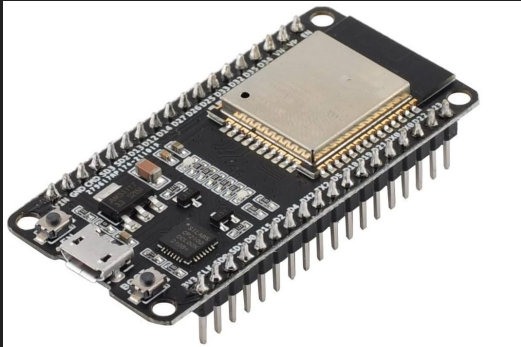
IoT Hardware (custom)



IoT Devices: Development Kit/ Hardware

Microcontrollers
Computers

Single-board



ESP32

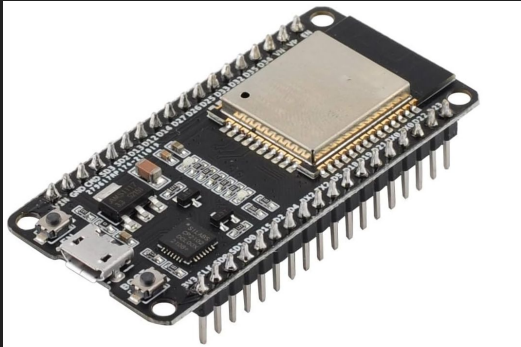
*Microcontroller
Vs
Microprocessor*



Raspberry Pi 5

IoT Devices: Development Kit/ Hardware

Microcontrollers



ESP32

- + cheap! (\$)
- + small
- + low power
- + light framework (no OS required)
- + rugged
- + c/c++, python are most common frameworks
- limited processing ability
- one-at-a-time instruction limitation
- slower than CPUs
- limited memory
- hardware interfacing has its challenges
- requires Integrated Development Environment (IDE)

IoT Devices: Development Kit/ Hardware

Single-board Computers

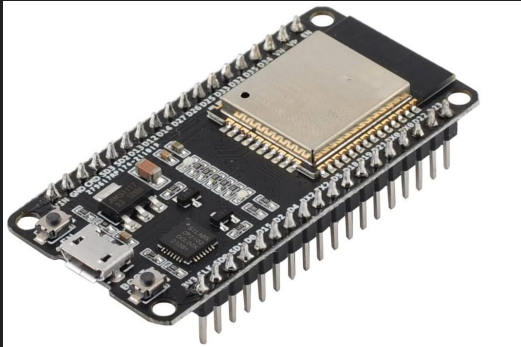
- + needs more power
- + OS (linux environment or run headless- no OS)
- + python typical for linux systems
- + supports remote connections
- + many ports/ connections!
- thermal considerations
- one wrong connection and x_x
- features you might not need
- not as cheap (\$\$)
- supply chain limitations



Raspberry Pi 5

IoT Devices: Development Kit/ Hardware

Microcontrollers



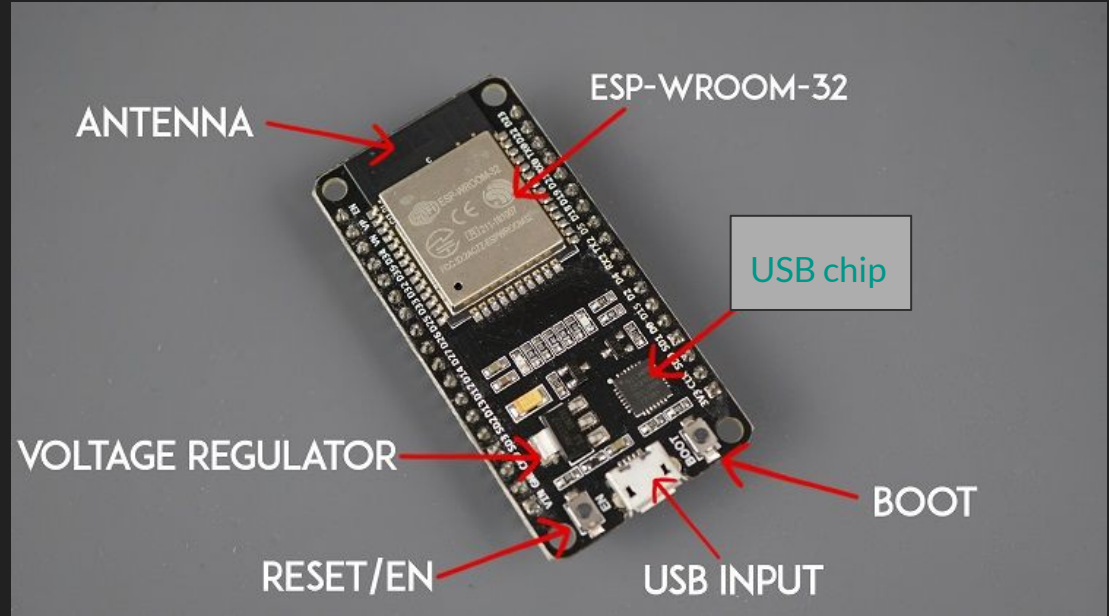
ESP32

Our focus = **microcontroller development kits.**

- + simplicity
- + affordability
- + support
- + practical for our simple example

Cost effective and versatile










ESP32 Hardware Detail (\$5 Wi-Fi Microcontroller)








ESP32 Hardware Detail (\$5 Wi-Fi Microcontroller)

THERE ARE LOTS OF ESP32 VARIANTS.

Not all pinouts (GPIO) order are similar.

								
Espressif Systems...	Seeed Technology...	Espressif Systems...	Espressif Systems...	Espressif Systems...	Espressif Systems...	ESP32 Development...	Arduino ABX00083...	Espressif Systems...
\$8.09	\$4.99	\$8.09	\$5.06	\$5.38	\$8.74	\$7.99	\$21.00	\$8.09
DigiKey	DigiKey	DigiKey	DigiKey	DigiKey	DigiKey	Amazon.com	DigiKey	DigiKey

				
Amazon.com	Amazon.com · In stock	The Engineering Projects	Arduino · In stock	EZmation
Hit stop ESP32 WROOM-32	DIYmall ESP32 DEV KIT ESP32-WB	ESP32 Pinout, Datasheet, Features	Arduino Nano ESP32 - Arduino	ESP32 ESP-WROOM-32

IoT Devices: Development Kit/ Hardware

ESP32-WROOM-32 Specs

CPU Manufacturer: Espressif

CPU Speed: 240 MHz

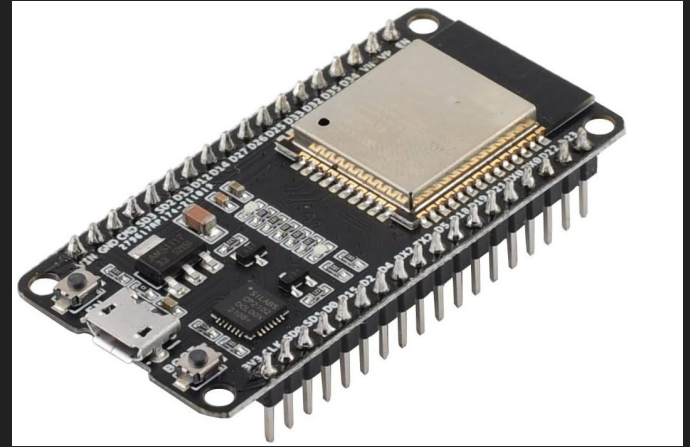
Memory: 4 MB

RAM: 160 KB

WiFi: 802.11b

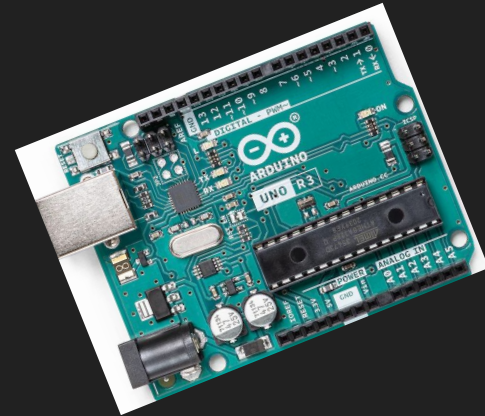
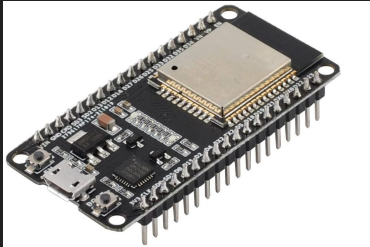
Bluetooth: yes

~\$8



IoT Devices: Development Kit/ Hardware

Feature	ESP32-WROOM-32	Arduino Uno (ATmega328P)
CPU	Dual-core Tensilica Xtensa LX6 @ 240 MHz	8-bit AVR @ 16 MHz
RAM	520 KB SRAM	2 KB SRAM
Flash Memory	4 MB (<i>~125x more flash</i>)	32 KB
Wi-Fi	Yes (802.11 b/g/n)	No
Bluetooth	Yes (Classic + BLE)	No
GPIOs	~30 (depends on module)	14
ADC Resolution	12-bit	10-bit
DAC	2 × 8-bit DACs	No
PWM Channels	16	6
Price	~\$8–12	~\$3–10



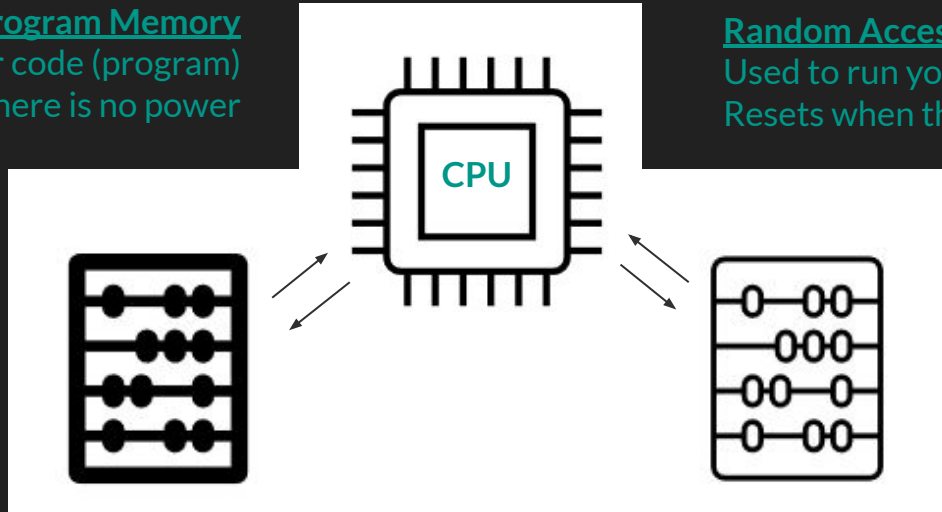
Computer Fundamentals: Memory Types

Program Memory

Stores your code (program)
Persists when there is no power

Random Access Memory (RAM)

Used to run your code when powered
Resets when there is no power



Computer Fundamentals: Memory Size



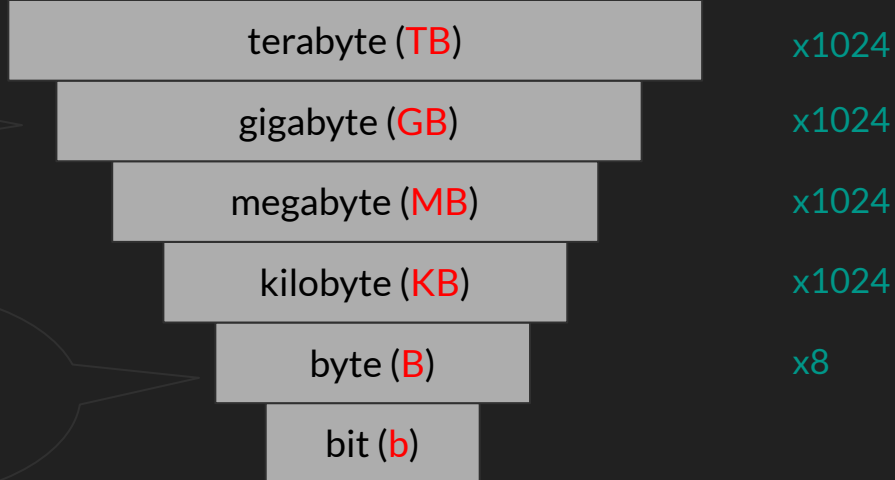
Computer Fundamentals: Memory Size



1080P movie
blu-ray disc ~
50 GB



The letter "A"
in binary =
01000001



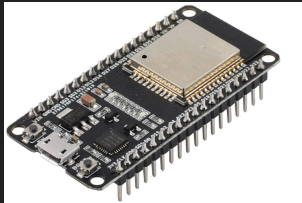
Computer Fundamentals: Memory Size

ESP32 Specs

CPU Speed:

Storage:

RAM:



240 MHz

4 MB

160 KB

iPhone 15 (2023) Specs

CPU Speed:

3.46 GHz

Storage:

128 GB

RAM:

6 GB



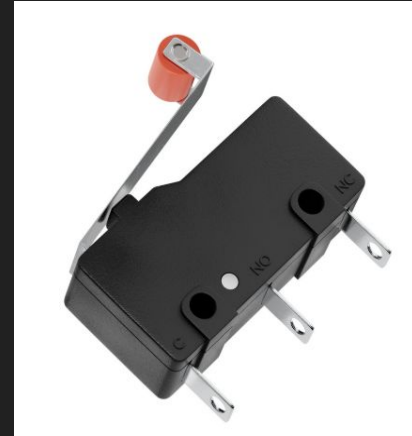
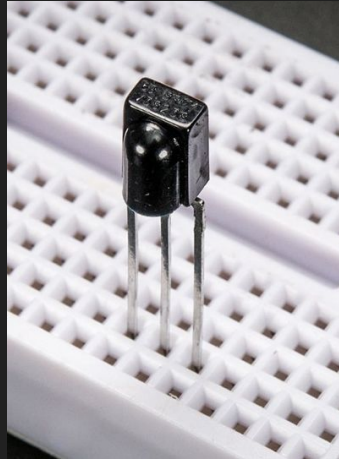
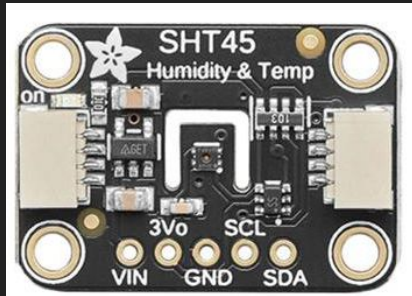
39,300 times larger

Sensors Example (Collects data from something)

SHT45 (Temperature/Humidity)

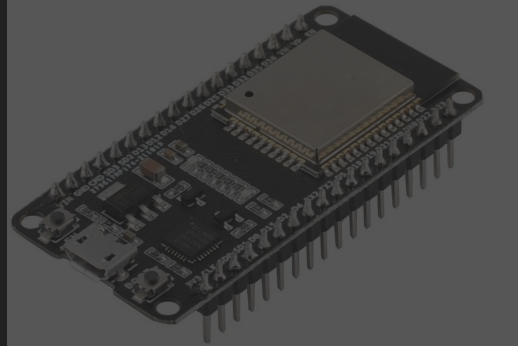
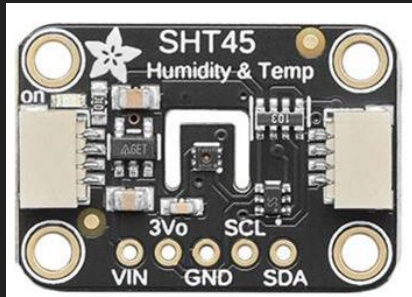
Microswitch

Infrared Receiver



IoT Hardware: Sensors

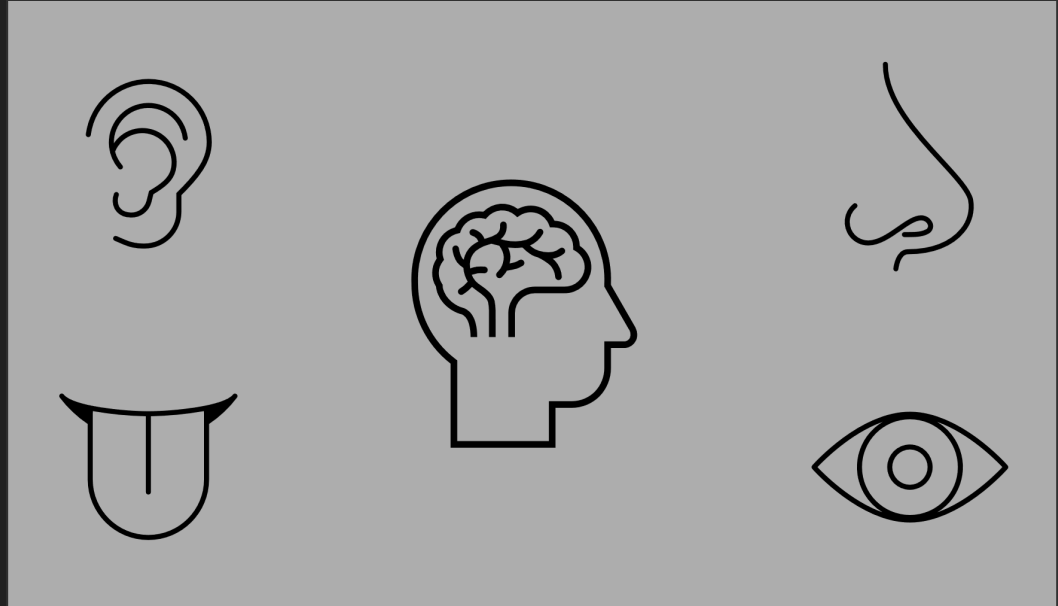
Hardware that can touch the physical world.



IoT Devices (Sensors)

What “sensors” do you have?

What can you sense?



IoT Devices (Sensors)

What senses can be detected by hardware?

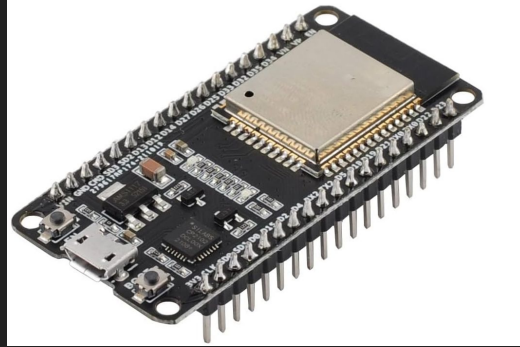
Human Sense	Application
Taste	Chemical Detection, PH level, Salinity
Smell	Vapor (VoC), Smoke, Particulate Detection [gas]
Touch	Physical [impact, force, pressure], Heat, Distance, Roughness, Capacitance
Feel	Gravity, Acceleration, Orientation/ Direction, Voltage, Current, Magnetic Orientation
Vision	Light, Color, Infrared, Motion, Point Cloud Map,
Hear	Sound, Vibration, Noise, Ultrasonic Echo,

Sensors Critical Thinking

What sensors are in a dishwasher?

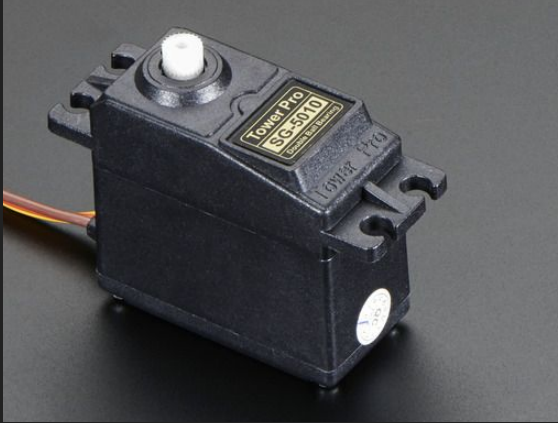


IoT Hardware: Review/ Questions?

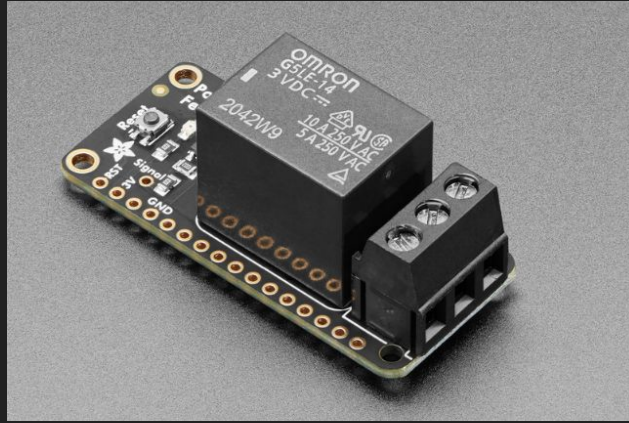


Actuators Example (Interacts with something)

Servo Motor



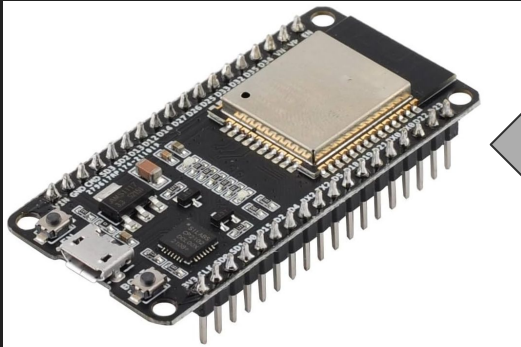
Relay



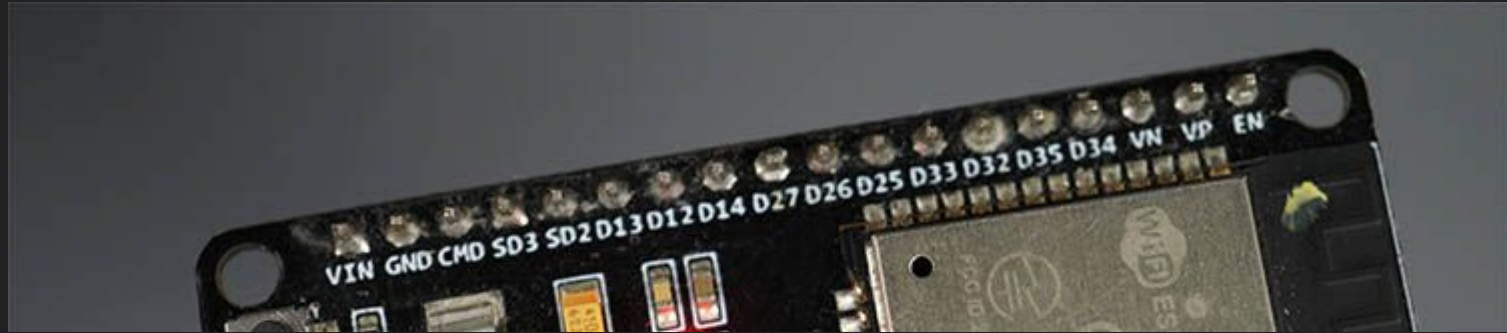
HARDWARE detail START

Arduino Integrated Development Environment (IDE)

Let's upload code to our Microcontroller!



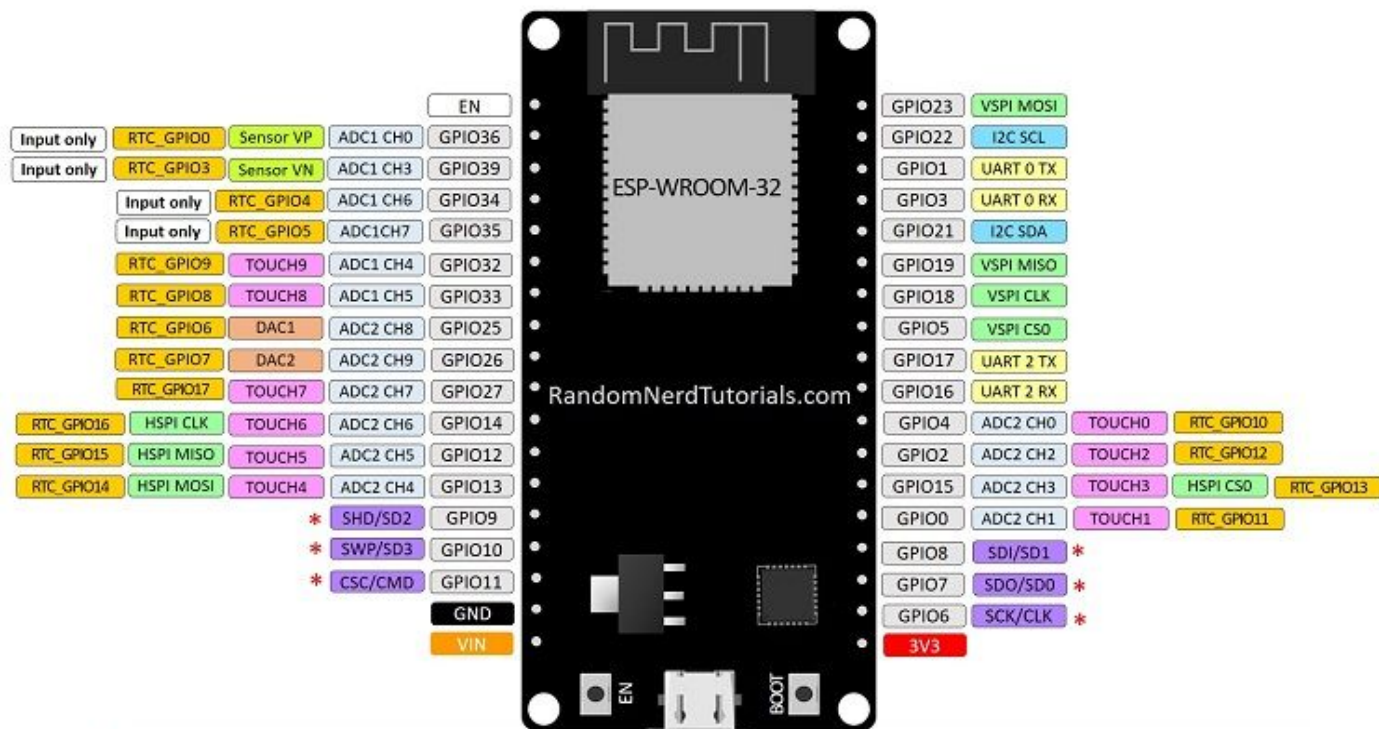
General Purpose Input/Output (GPIO) Pins



Pinout

ESP32 DEVKIT V1 – DOIT

version with 36 GPIOs



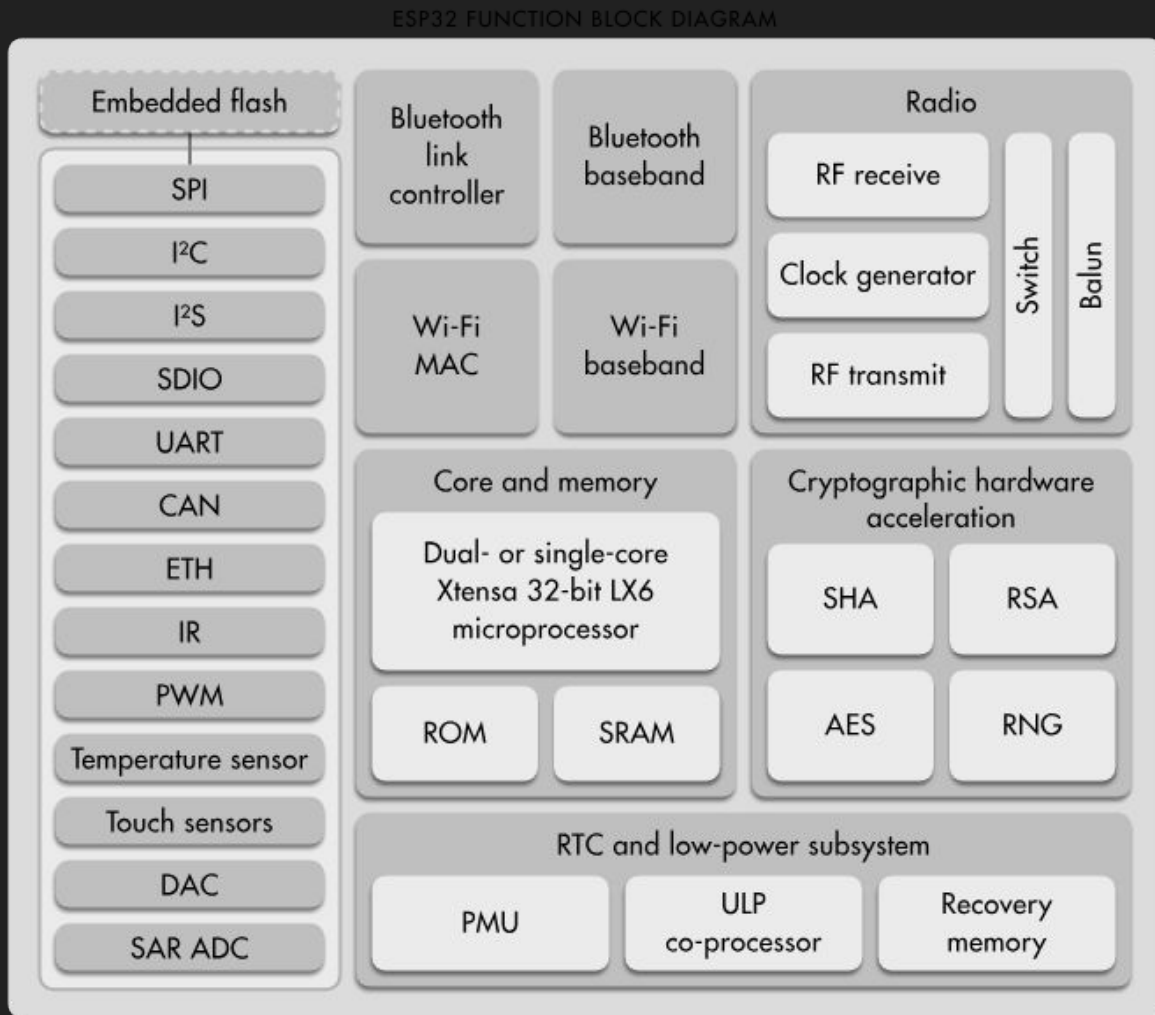
* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and CSC/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on ESP-WROOM-32 and are not recommended for other uses.

Hardware (Functional Diagram)

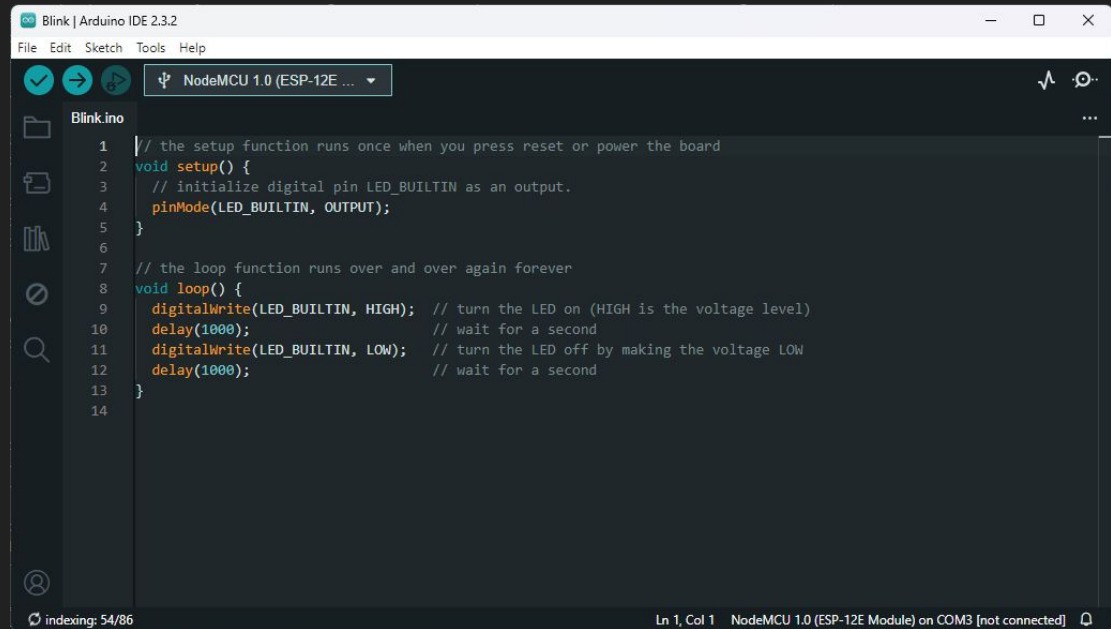
Communication protocols
for
wired GPIO and wireless
interfacing.

We are using i2c and WiFi

<http://esp32.net/>



Arduino Integrated Development Environment (IDE)

A screenshot of the Arduino IDE interface. The window title is "Blink | Arduino IDE 2.3.2". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar shows icons for checking, running, and uploading code, along with a dropdown menu currently set to "NodeMCU 1.0 (ESP-12E ...)". The left sidebar contains icons for file explorer, search, and other IDE functions. The main editor area displays the "Blink.ino" file with the following code:

```
1 // the setup function runs once when you press reset or power the board
2 void setup() {
3   // initialize digital pin LED_BUILTIN as an output.
4   pinMode(LED_BUILTIN, OUTPUT);
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
10  delay(1000); // wait for a second
11  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
12  delay(1000); // wait for a second
13 }
14
```

The status bar at the bottom indicates "indexing: 54/86" on the left and "Ln 1, Col 1 NodeMCU 1.0 (ESP-12E Module) on COM3 [not connected]" on the right.

Digital Pins: Output vs Input

```
const int ledPin = 13;  
pinMode(LEDpin, OUTPUT);  
digitalWrite(ledPin, HIGH); //  
turn LED on:
```

Tip! Max current drawn per GPIO = 40mA



```
const int buttonPin = 2;  
pinMode(buttonPIN, INPUT);  
If (digitalRead(buttonPin) == HIGH) //  
    {digitalWrite(ledPin, HIGH);}  
check if pressed, do something.
```

Tip! Do not exceed 3.3V (working voltage of board).



Arduino IDE: Whaaaaaaaaaat?

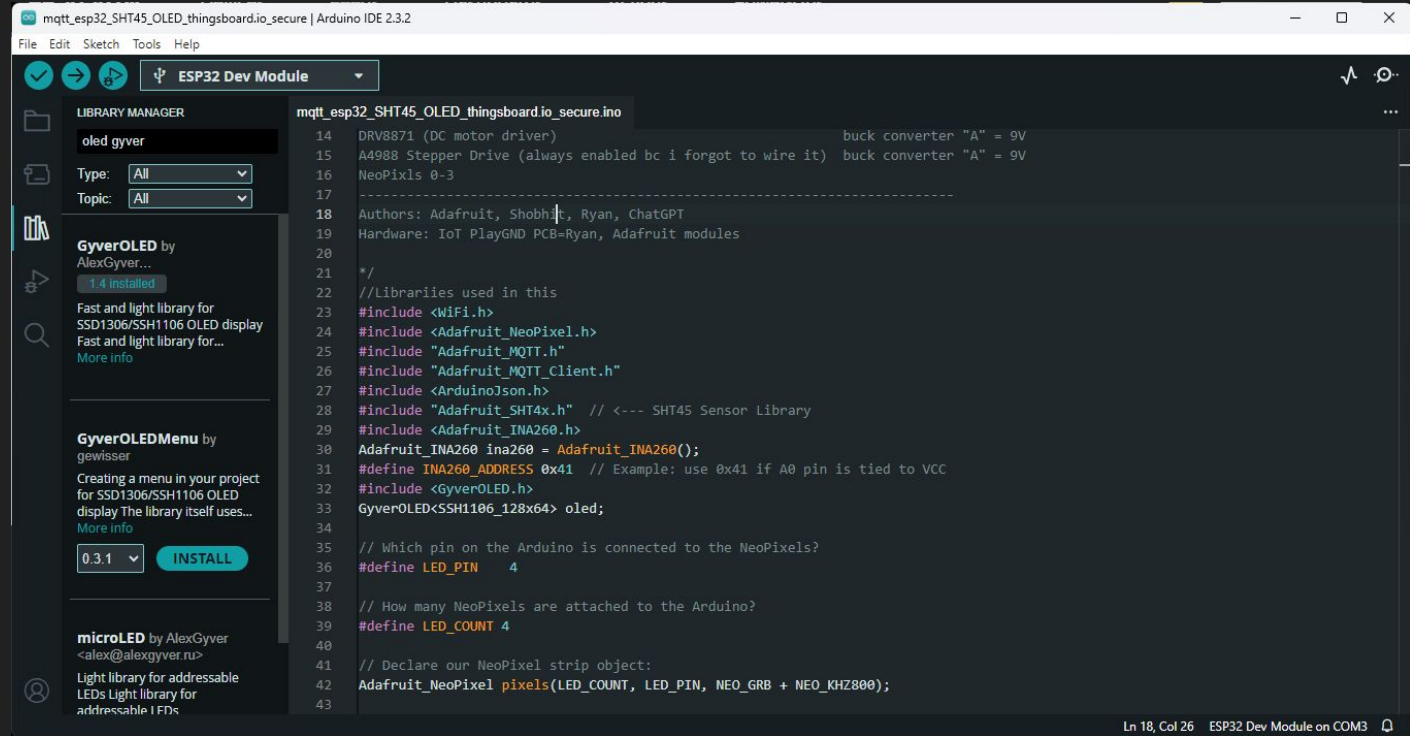
Libraries

Board manager

Dark theme

COM ports

Serial Monitor



Analog Pins: Output (PWM) vs Input

```
const int ledPin = 13;  
pinMode(LEDpin, OUTPUT);  
analogWrite(ledPin, 127); //  
turn LED to 50% duty cycle:
```

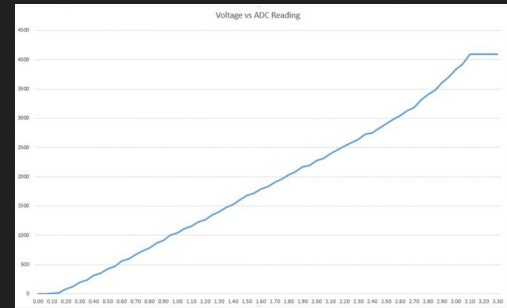
Max current drawn per GPIO = **40mA**



```
const int potPin = 13;  
(analogRead(potPin) // read the voltage  
at the pin
```

ADC Sensitivity = 12-bit resolution (2^{12})
 $3.3 \text{ V} / 4096 = 0.008\text{v}$

Always read your device's datasheet for details.

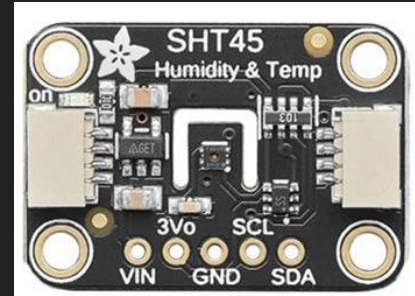


Temp and Humidity Sensor (SHT45)

Talks over i2c.

I²C (Inter-Integrated Circuit) is a **two-wire serial communication protocol** used to connect low-speed **peripheral devices (sensors, displays, etc.)** to microcontrollers (like Arduino, ESP32, etc.) over short distances.

- Short range (1 meter)
- Slow (400khz) *USB 1.0 is about 10x to 100x faster.*



Temp and Humidity Sensor (SHT45)

Upload code to ESP32, watch the serial monitor

JSON

JSON (JavaScript Object Notation) open standard file / data interchange format that uses human-readable text to store and transmit data objects consisting of name–value pairs and arrays (or other serializable values).

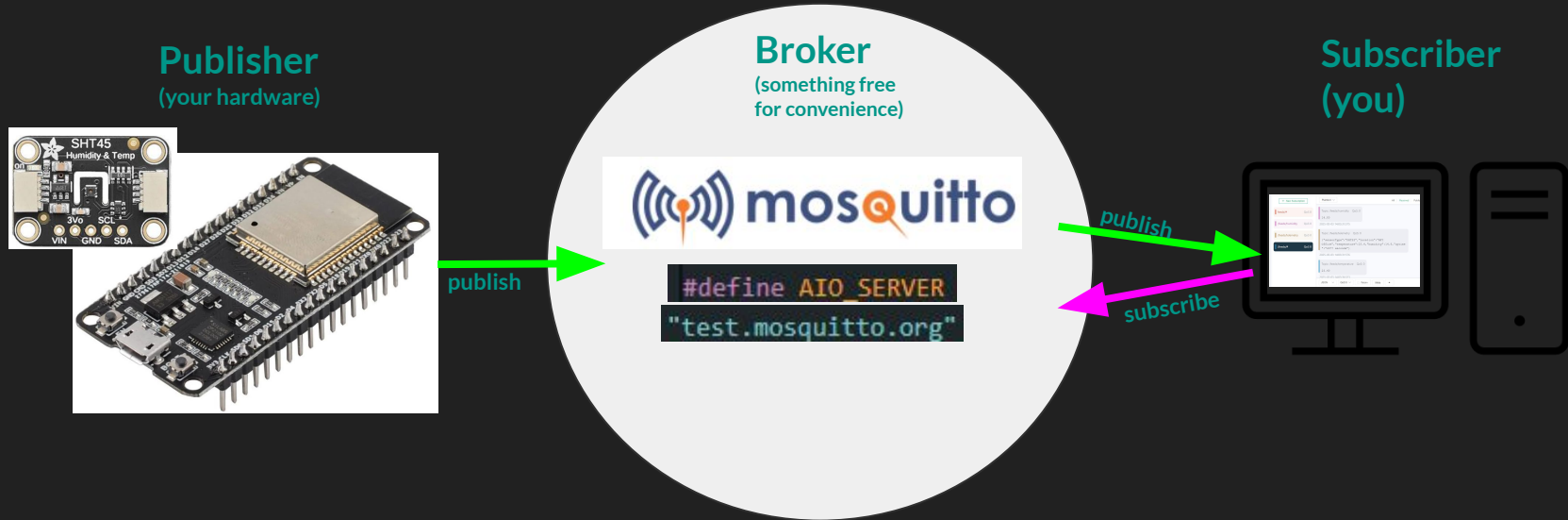
```
{"first_name": "John", "last_name": "Smith"},
```

JSON format example

```
{  
  "first_name": "John",  
  "last_name": "Smith",  
  "is_alive": true,  
  "age": 27,  
  "address": {  
    "street_address": "21 2nd Street",  
    "city": "New York",  
    "state": "NY",  
    "postal_code": "10021-3100"  
  }  
}
```

Explanation: It's text labels & text with formatting.
Objects, arrays, and values.

What is MQTT ?



MQTT is kinda like the post office

Think of MQTT like a postal service for smart devices:

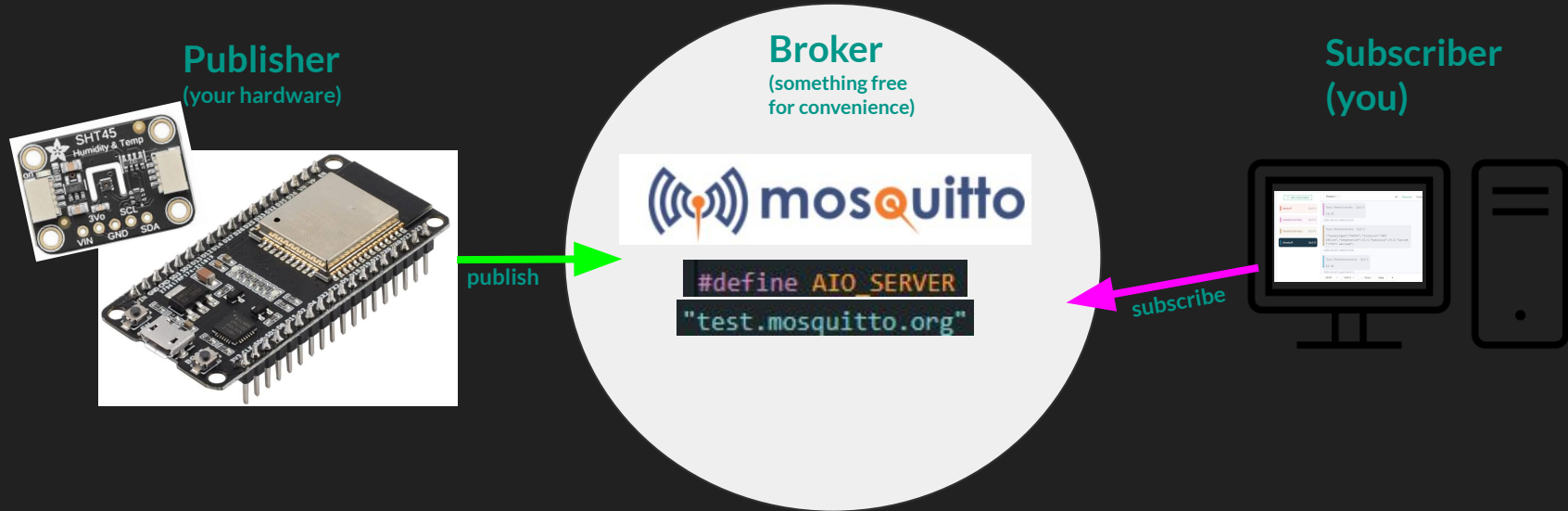
The **broker** is the post office—it sorts and delivers messages to the right recipients.

Devices (**clients**) subscribe to topics like people receiving mail at their home address.

Other devices (**clients**) publish messages like someone sending a letter to a specific topic (mailbox).

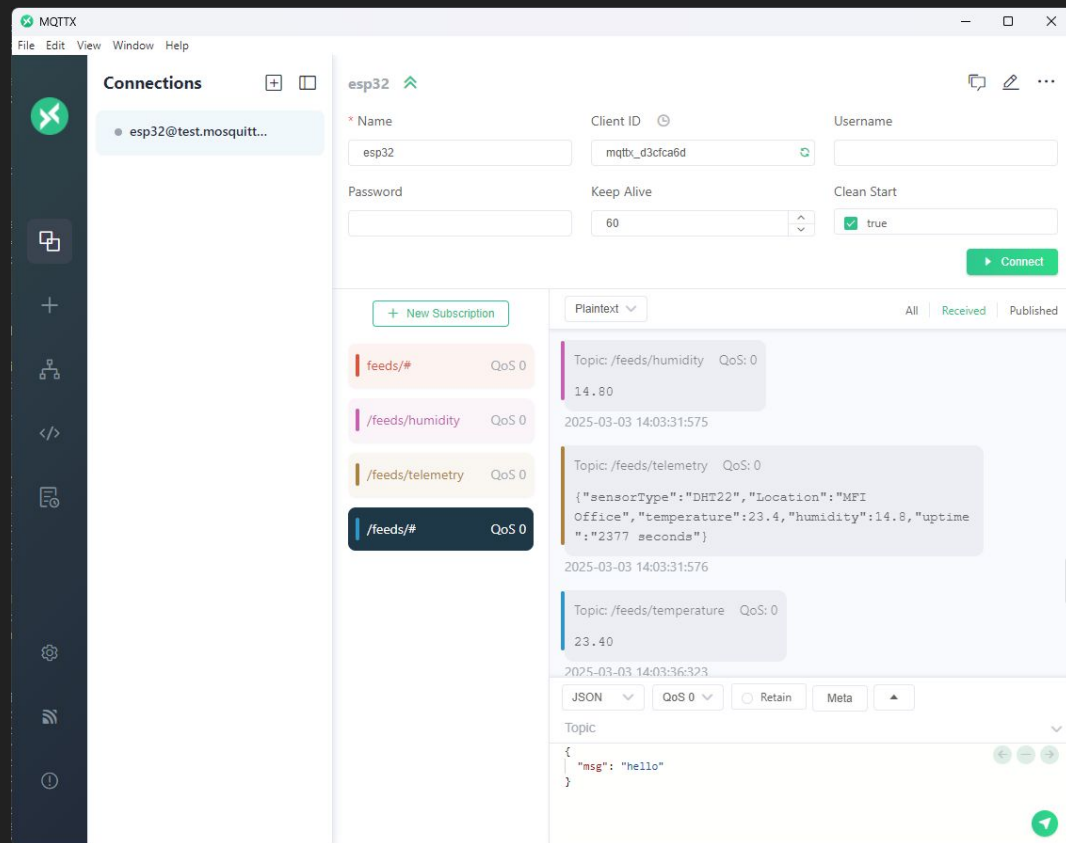
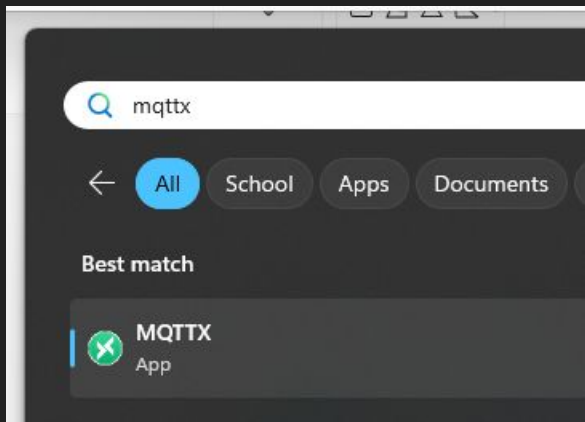
Only those subscribed to that topic get the message—just like only the right recipient gets the mail.

MQTT with our Hardware

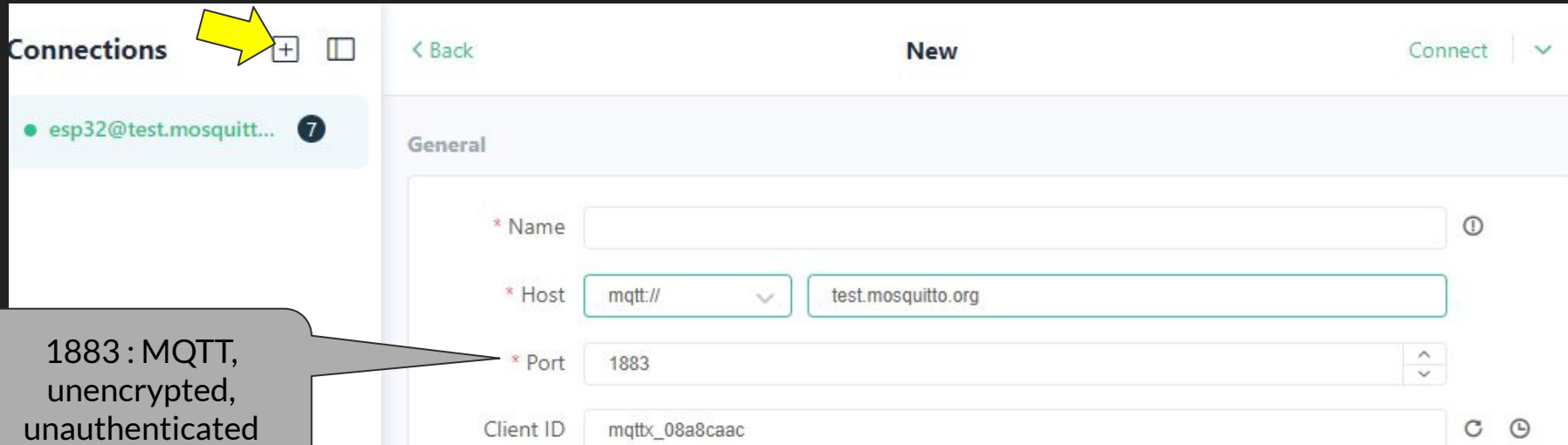


Client Toolbox: MQTTX

WinKey + “MQTTX”



Connect to Broker

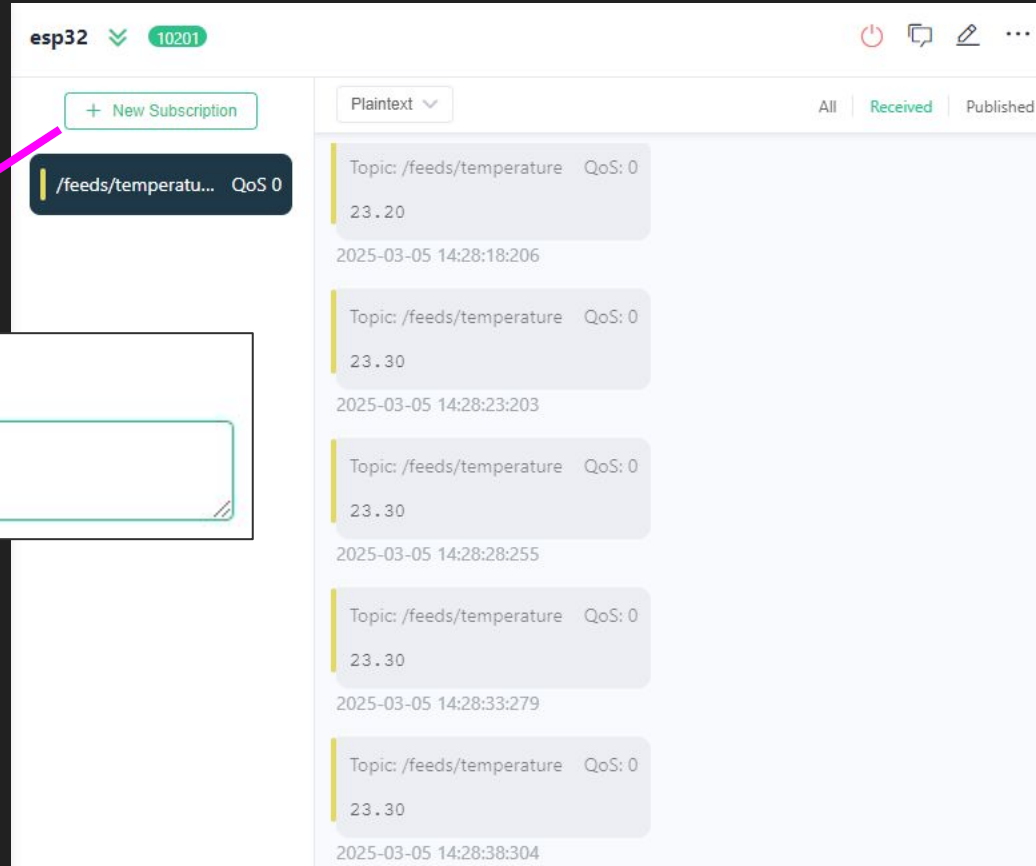


The screenshot shows the MQTT client interface. On the left, the 'Connections' list contains one entry: 'esp32@test.mosquitto...' with a status indicator and a '7' in a circle. A yellow arrow points to the '+' button next to this entry. The main area is titled 'New' and contains a 'General' section with the following fields:

- Name:** An empty text field.
- Host:** A dropdown menu set to 'mqtt://' and a text field containing 'test.mosquitto.org'.
- Port:** A text field containing '1883'.
- Client ID:** A text field containing 'mqtx_08a8caac'.

A speech bubble points to the 'Port' field with the text: '1883: MQTT, unencrypted, unauthenticated'.

Subscribe to a topic.



The screenshot shows the MQTT Explorer interface for an 'esp32' device with ID '10201'. A pink arrow points from the 'Topic' input field in the foreground to the subscription list in the background.

Subscription List:

- + New Subscription**
- /feeds/temperatu... QoS 0**

Message History:

Topic	QoS	Message	Timestamp
/feeds/temperature	0	23.20	2025-03-05 14:28:18:206
/feeds/temperature	0	23.30	2025-03-05 14:28:23:203
/feeds/temperature	0	23.30	2025-03-05 14:28:28:255
/feeds/temperature	0	23.30	2025-03-05 14:28:33:279
/feeds/temperature	0	23.30	2025-03-05 14:28:38:304

* Topic

/feeds/temperature

Subscribe to all topics.

- 1) Subscribe to the topic `"/feeds/#"`
 - 2) Change the incoming formatting to look for JSON.
- Payload is a JSON string

```
// Create the JSON document
DynamicJsonDocument doc(1024); //allocate 1KB of memory
doc["sensorType"] = "DHT22";
doc["Location"] = "MFI Office";
doc["temperature"] = tempEvent.temperature; // value in degrees Celsius
doc["humidity"] = humEvent.relative_humidity; // value in percent
doc["uptime"] = String(millis()/1000)+ " seconds";
```

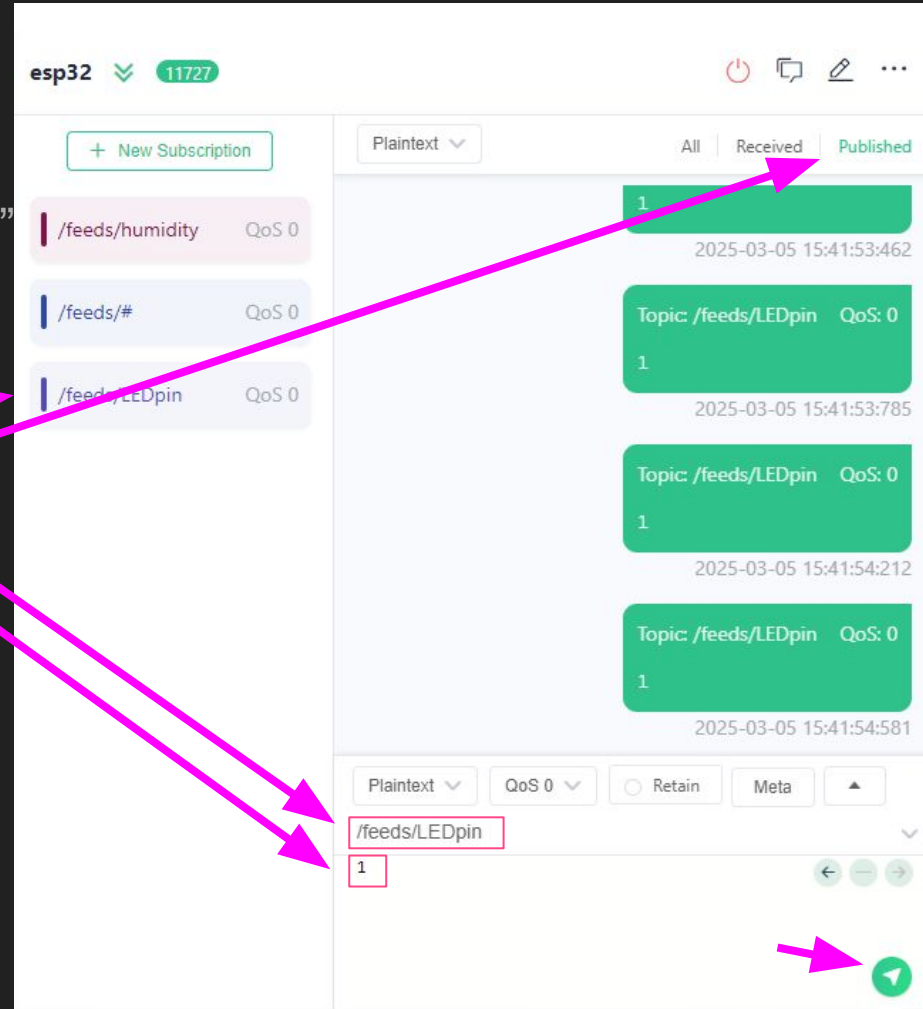
The screenshot shows the MQTT Explorer interface for an 'esp32' device with ID '10876'. A new subscription is created for the topic `/feeds/#` with a QoS of 0. The message format is set to 'JSON'. The interface shows two received messages:

- Message 1: Topic `/feeds/humidity`, QoS: 0, Payload: `37`. Received at 2025-03-05 15:26:12:611.
- Message 2: Topic `/feeds/telemetry`, QoS: 0, Payload: `{ "sensorType": "DHT22", "Location": "MFI Office", "temperature": 22.9, "humidity": 36.9, "uptime": "123 seconds" }`. Received at 2025-03-05 15:26:12:612.

At the bottom, there are controls for message format (Plaintext), QoS (QoS 0), Retain status, and a scroll bar.

Publish to Device

- 1) Subscribe to the topic “/feeds/LEDpin”
- 2) Switch to Published tab.
- 3) Type in the Topic “/feeds/LEDpin”
- 4) Send the message “0” or “1”



Broker Services (Dashboard)



Review Topics!

What are IoT benefits?

Explain some technical fundamentals of:

Computer Science: Memory, Processing,

IT: Networking basics (*relax, nothing obscure*)

Hardware: Microcontrollers, SoC, Sensors

Software: C/C++, Arduino IDE

ChatGPT: it writes and explains code for you.

MQTT: Publisher Subscriber Model

What does our hardware do: Monitor Temperature over WiFi,
toggle a relay.