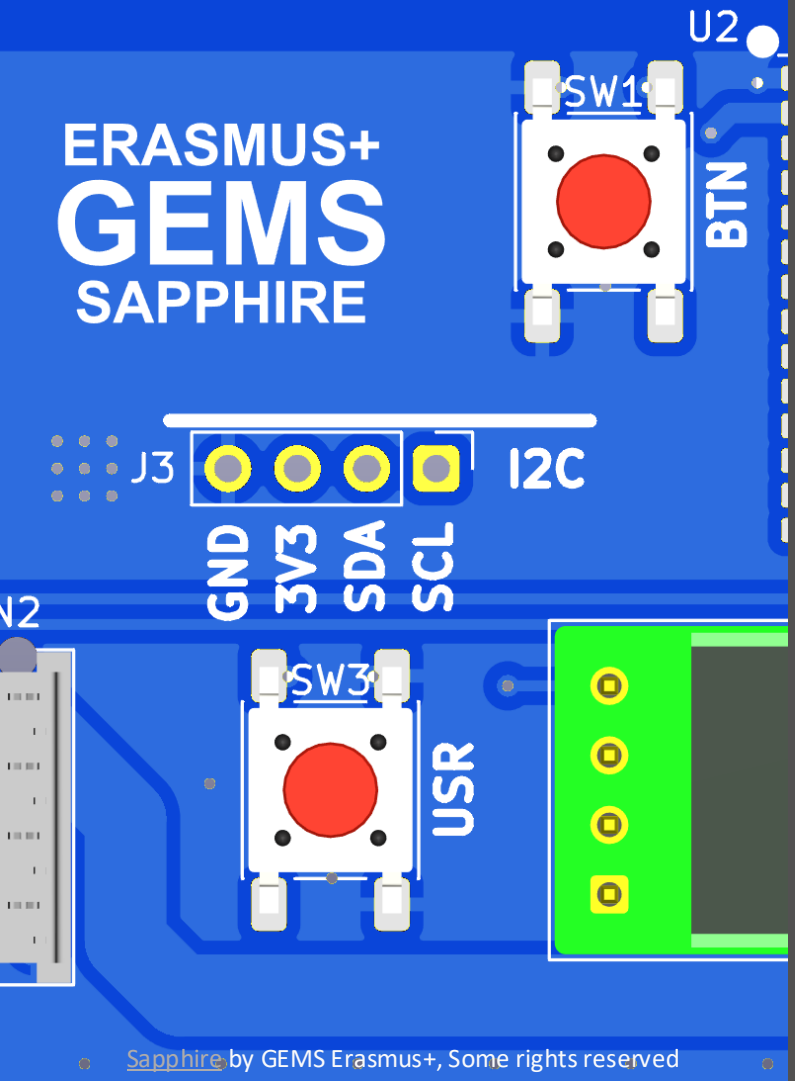


Wireless Communication Technologies

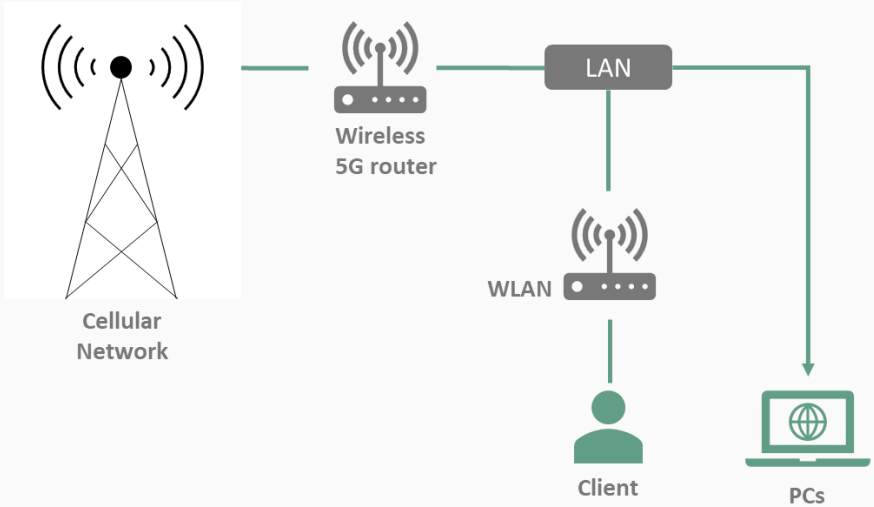
Dimitrios Georgopoulos

Learning Objectives

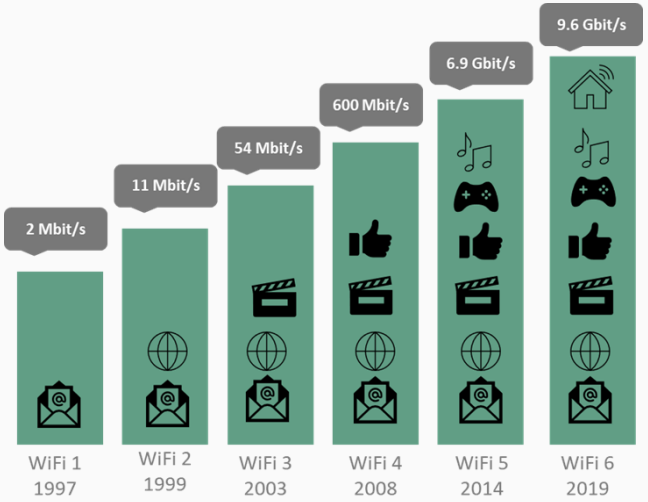
1. Wireless communication fundamentals
2. Wi-Fi, Bluetooth, and other wireless protocols
3. Configuration and settings for serial communication

Wireless communication fundamentals

- A **wireless network** is a grouping, or network, of **multiple devices** where data is sent and received over radio frequencies.
- The **transmission** of data across a wireless network is typically done with **antennas**, which are often small, embedded pieces of hardware within a given device.
- **Wireless Devices:** Cellular phones, Cordless telephones, GPS, Computer peripherals, Wi-Fi Devices, IR Devices
- **Wireless Networks:** WLAN, WWAN, WMA, WPAN, Fixed wireless, CBRS, WMN
- **Cellular Networks:** 2G, 3G, 4G, 5G



Wi-Fi (IEEE 802.11)



Wi-Fi *refers to a group of wireless network protocols based on the IEEE 802.11 standards, widely used for local area networking and Internet access. It enables nearby digital devices to exchange data using radio waves.*

- **802.11 (1999):** The original standard with speeds around 2 Mbit/s.
- **802.11b (1999):** Operates in the 2.4 GHz band, reaching up to 11 Mbit/s.
- **802.11g (2003):** Enhanced to 54 Mbit/s in the 2.4 GHz band.
- **802.11n (2009):** Introduced MIMO technology, delivering up to 600 Mbit/s on both 2.4 GHz and 5 GHz bands.
- **802.11ac (2013):** Achieves speeds up to 7 Gbit/s, with lower energy consumption due to shorter transmission times.
- **802.11ax (Wi-Fi 6, 2019):** Offers up to 9.6 Gbit/s, with advanced multitasking capabilities like OFDMA.

Bluetooth

Bluetooth *is a standardized protocol designed for transmitting and receiving data over a 2.4GHz wireless connection. It offers secure communication, making it ideal for short-range, low-power, and cost-effective wireless interactions between electronic devices.*

Applications in Robotics

Remote control → Mobile robots controlled via smart devices .

Human-robot collaboration → Wearable devices for human robot interaction.

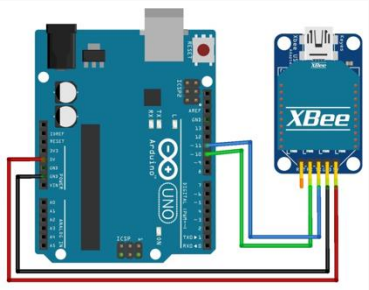
Wireless sensors → Connecting robotic arms to sensors for automation.

Zigbee

- ***Zigbee*** *is a low-power, short-range wireless communication protocol based on IEEE 802.15.4. It is designed for low data rate, energy-efficient applications*

Why Zigbee in Robotics?

- Low power consumption → Ideal for battery-powered robotic sensors
- Mesh networking → Enables multi-hop communication for better coverage
- High device capacity → Supports up to 65,000 nodes in a network
- Reliable & secure → AES-128 encryption for data security



https://how2electronics.com/how-to-set-up-interface-xbee-module-with-arduino-tx-rx/

5G

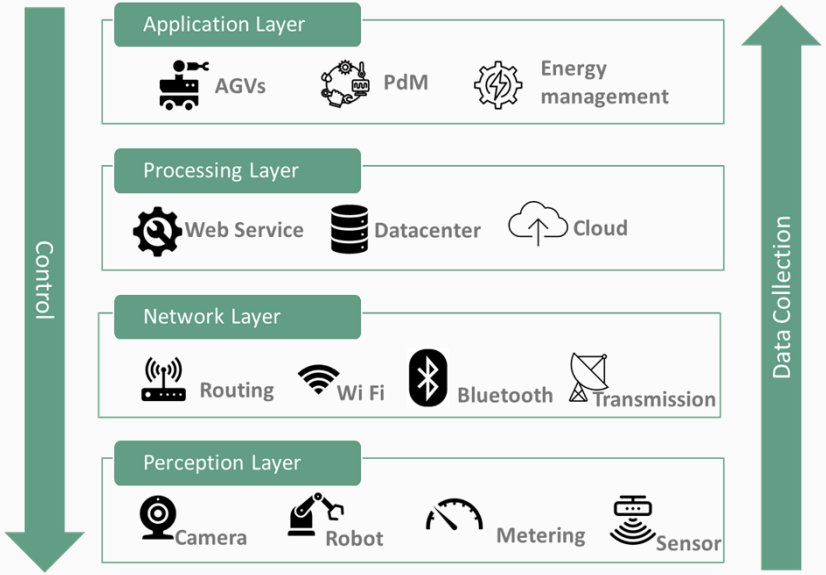
5G *is the Fifth-generation wireless technology, succeeding 4G LTE it is Designed for ultra-fast speeds, ultra-low latency, and high device density . Operates across three frequency bands: low-band, mid-band, and high-band (mmWave)*

Why 5G in Robotics?

- **Ultra-low latency (<1ms)** → Crucial for real-time robotic control
- **High-speed data transfer** → Supports HD video streaming & AI-powered analytics
- **Massive device connectivity** → Handles thousands of robots/sensors in a factory
- **Network slicing** → Creates custom private networks for robotic applications

Wireless communication in Robotics - IIOT

*The **Industrial Internet of Things (IIoT)** refers to the networking of **sensors**, **instruments**, and other devices with computers to support **industrial applications**, such as **manufacturing** and **energy management**.*



Wireless communication in Robotics - IIOT

Challenges of IIoT

- Security Risks
- Data Management
- Interoperability Issues
- High Implementation Costs

Benefits of IIoT

- Increased Efficiency
- Cost Savings
- Enhanced Safety
- Better Decision-Making

IIoT Applications

- Predictive Maintenance
- Supply Chain Management
- Energy Management
- Remote Monitoring

Hands On example: Setting Up a Wi-Fi Network on an ESP32 Module

Required Components:

- **ESP32 Development Board**
- **USB Cable** (for programming)
- **Arduino IDE** (or PlatformIO)
- **Wi-Fi Network** (for STA mode)

Hands On example: Setting Up a Wi-Fi Network on an ESP32 Module

Steps

Step 1: Install ESP32 Board in Arduino IDE

Step 2: Set Up ESP32 as a Wi-Fi Access Point (AP Mode)

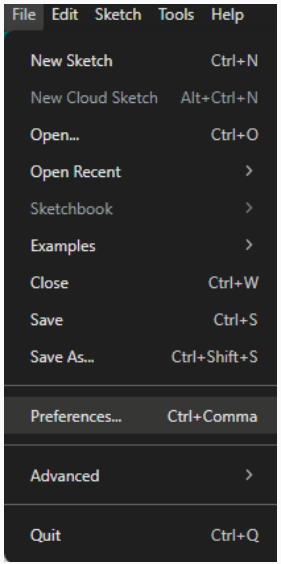
Step 3: Set Up ESP32 as a Wi-Fi Station (STA Mode - Connecting to a Router)

Step 4: Test the Wi-Fi Network

Hands On example: Setting Up a Wi-Fi Network on an ESP32 Module

Step 1: Install ESP32 Board in Arduino IDE

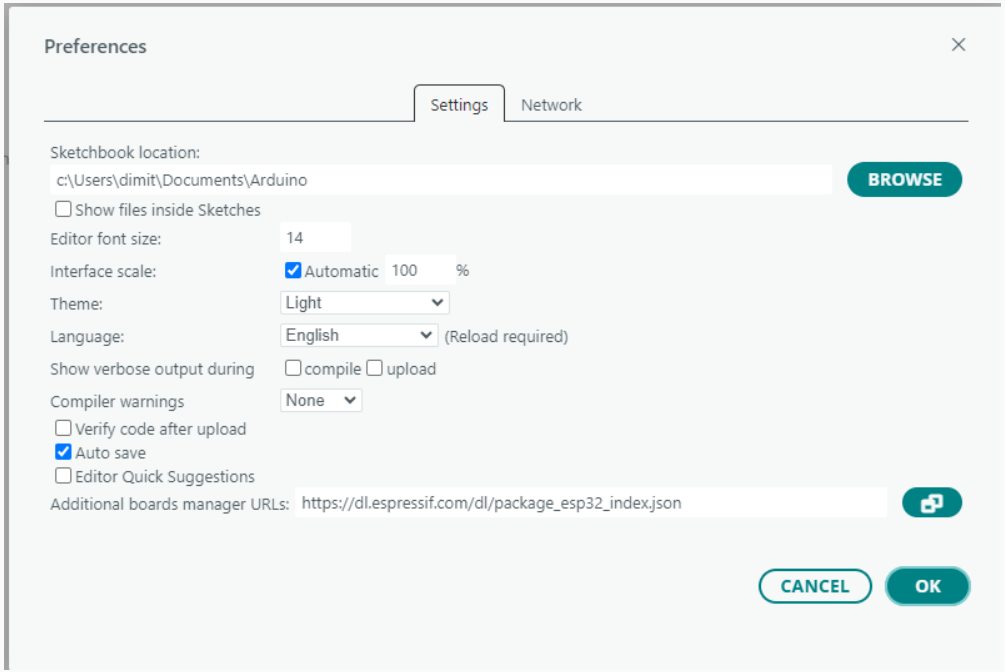
1. Open **Arduino IDE** → Go to **File → Preferences**



Hands On example: Setting Up a Wi-Fi Network on an ESP32 Module

Step 1: Install ESP32 Board in Arduino IDE

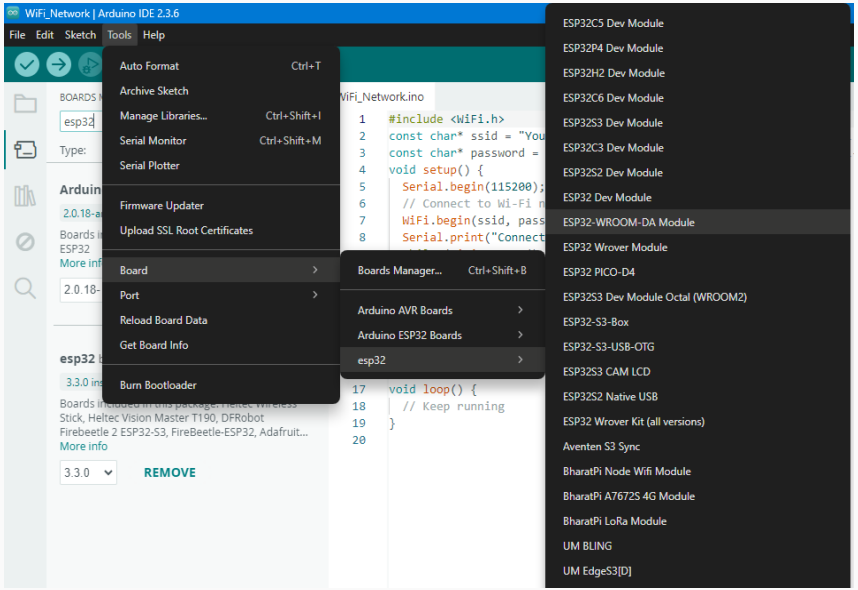
Add the ESP32 board manager URL: → https://dl.espressif.com/dl/package_esp32_index.json



Hands On example: Setting Up a Wi-Fi Network on an ESP32 Module

Step 1: Install ESP32 Board in Arduino IDE

Select your **ESP32 board** from Tools → Board



Hands On example: Setting Up a Wi-Fi Network on an ESP32 Module

Step 2: Set Up ESP32 as a Wi-Fi Access Point (AP Mode)

```
#include <WiFi.h>
const char *ssid = "ESP32_Robot_Network";  // Wi-Fi SSID
const char *password = "12345678";         // Wi-Fi Password (Min 8 chars)
void setup() {
 Serial.begin(115200);
 // Set ESP32 as Access Point
  WiFi.softAP(ssid, password);
  Serial.println("Wi-Fi Access Point Started");
  Serial.print("ESP32 IP Address: ");
  Serial.println(WiFi.softAPIP());  // Print AP IP Address
}
void loop() {
  // Keep running
}
```

Hands On example: Setting Up a Wi-Fi Network on an ESP32 Module

Step 3: Set Up ESP32 as a Wi-Fi Station (STA Mode - Connecting to a Router)

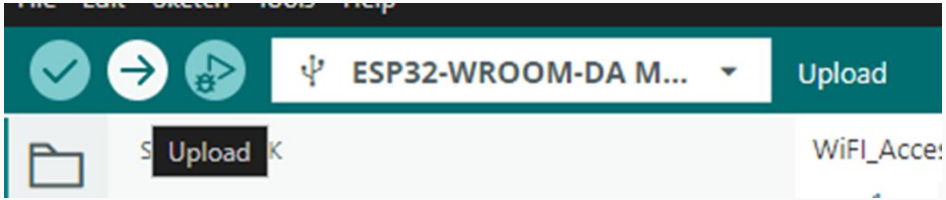
WiFi_Network.ino

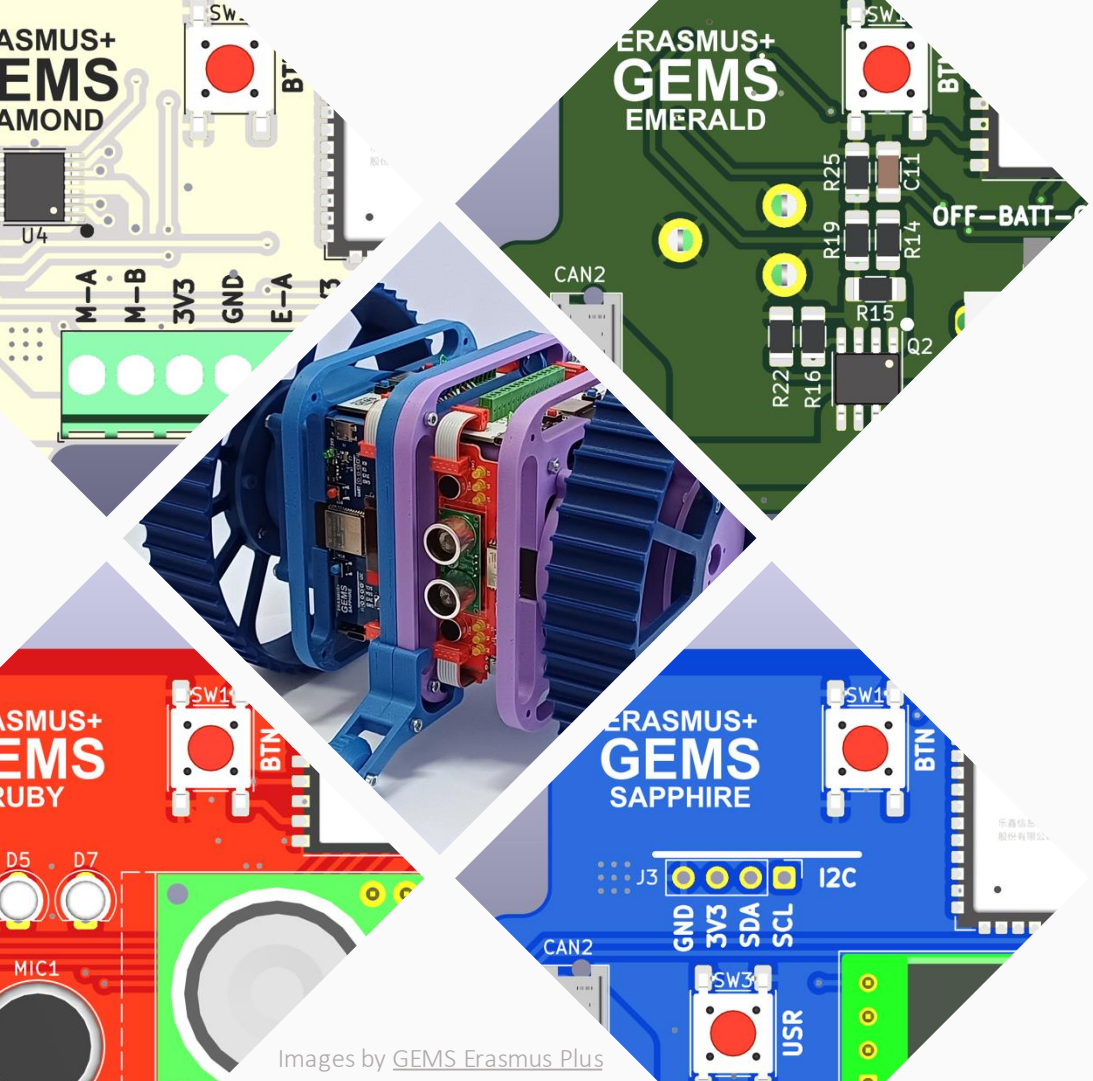
```
#include <WiFi.h>
const char* ssid = "Your_WiFi_SSID";      // Replace with your Wi-Fi SSID
const char* password = "Your_WiFi_Password"; // Replace with your Wi-Fi password
void setup() {
  Serial.begin(115200);
  // Connect to Wi-Fi network
  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected!");
  Serial.print("ESP32 IP Address: ");
  Serial.println(WiFi.localIP());  // Print ESP32's assigned IP
}
void loop() {
  // Keep running
}

```

Hands On example: Setting Up a Wi-Fi Network on an ESP32 Module

Step 4: Test the Wi-Fi Network

1. Upload the code using Arduino IDE2
2. Open Serial Monitor to check Wi-Fi status
3. If using AP mode, connect to "ESP32_Robot_Network" from your phone/laptop
4. If using STA mode, find the ESP32's assigned IP address in the serial monitor





Images by GEMS Erasmus Plus

Thank you for watching!

This video was created by Teaching Factory Competence Center for the GEMS Erasmus+ project (a collaboration between University of Ljubljana, University of Alcala, Teaching Factory Competence Center, and Delft University of Technology). It is released under a **Creative Commons Attribution Non Commercial Share Alike 4.0 International License**

Co-funded by the European Union