Design with Microprocessors

Lecture 13

Bluetooth

Year 3 CS

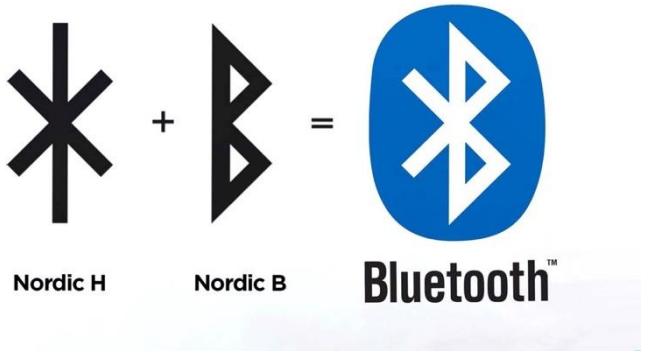
Academic year 2024/2025

1st Semester

Lecturer: Radu Dănescu

Bluetooth

- A short-range wireless communication technology that enables devices to exchange data over short distances.
- Developed by Ericsson in 1994.
- Named after Harald "Bluetooth" Gormsson, a Viking king who unified Denmark and Norway.
- Key Features:
  - Low power consumption.
  - Operates in the unlicensed 2.4 GHz band.
  - Supports both data and voice communication.

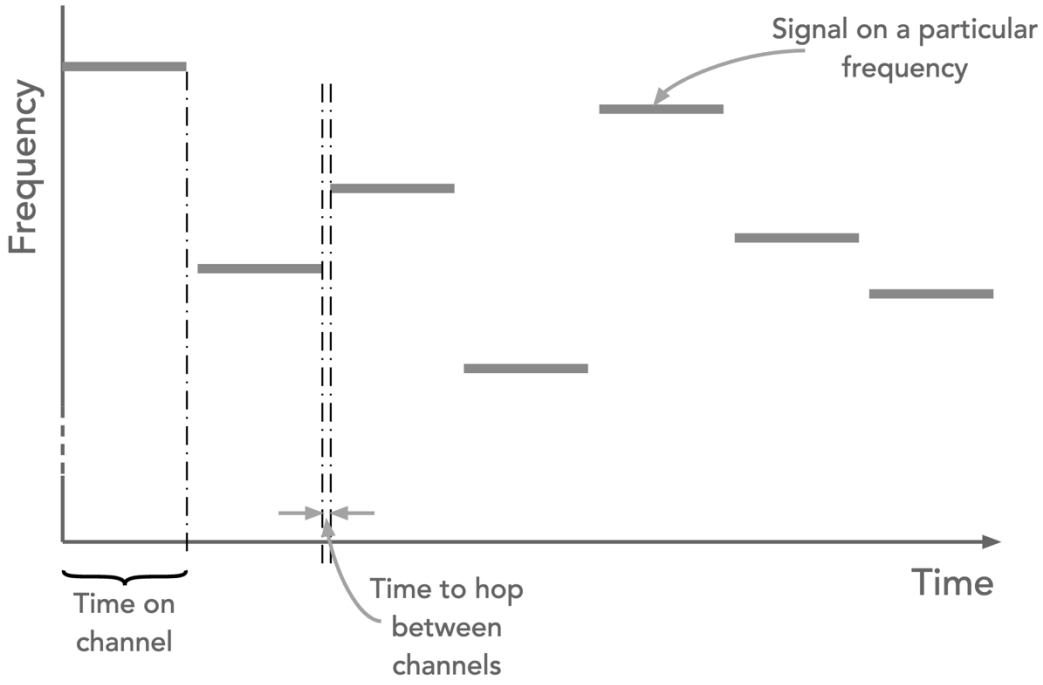


Bluetooth devices

- BT is a very popular wireless connection solution!
- Entertainment: audio devices (speakers, headphones), toys, controllers
- Vehicle to smartphone connection: Android Auto, Apple CarPlay
- Medica devices: health monitors, insulin pumps, …

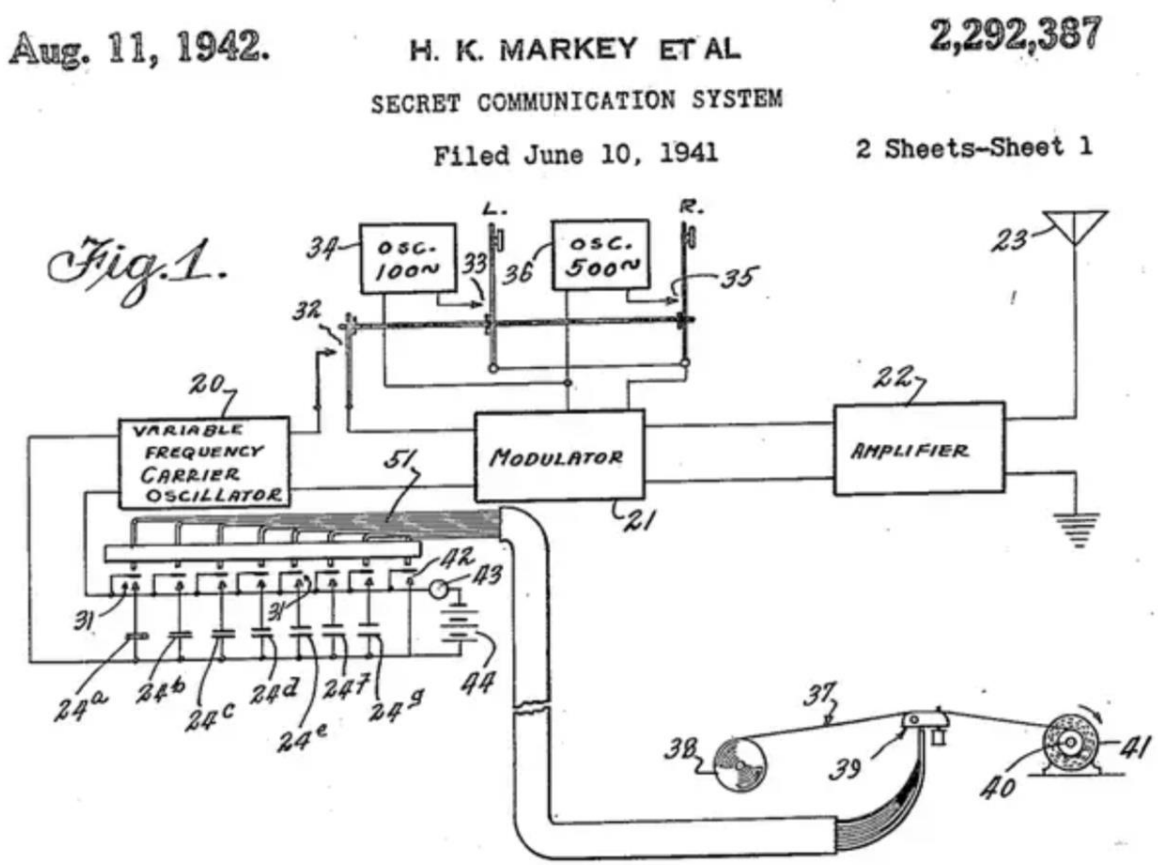
Bluetooth technology

- Bluetooth uses Frequency-Hopping Spread Spectrum (FHSS).
- The available 2.4 GHz band is divided into multiple channels.
- Classic Bluetooth (BR/EDR): 79 channels, each 1 MHz wide (in most countries).
- Bluetooth Low Energy (BLE): 40 channels, each 2 MHz wide.
- Devices "hop" between these channels during communication, switching frequencies up to 1,600 times per second.



Bluetooth technology

- Frequency hopping was designed for military use, to prevent interference (jamming)
- Relies on both sender and receiver using the same pattern of shifting frequencies


Aug. 11, 1942. H. K. MARKEY ET AL 2,292,387

SECRET COMMUNICATION SYSTEM

Filed June 10, 1941 2 Sheets-Sheet 1

Fig. 1.

Antheil and Lamarr

Bluetooth classic and Bluetooth Low Energy

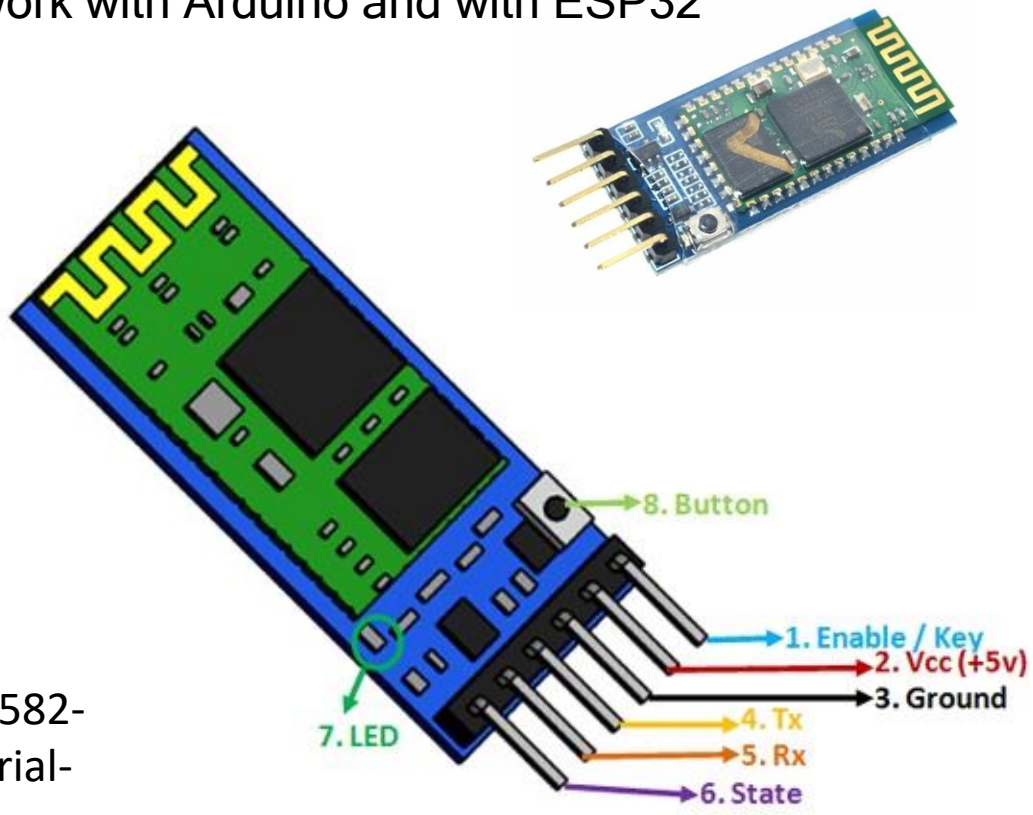
Feature	Bluetooth Classic	Bluetooth Low Energy (BLE)
Data Rate	1 Mbps for BR 2-3 Mbps for EDR	500kbps-1Mbps
RF Bandwidth	2.4 GHz ISM band (2400-2483.5 MHz)	2.4 GHz ISM band (2400-2483.5 MHz)
Number of Channels	79 Channels each of width 1 MHz	40 Channels each of width 2 MHz
Communication Range	The Same (8m up to 100m)	The Same (8m up to 100m)
Power Consumption	High (up to 1W)	Low (0.01W up to 0.5W)
Device Pairing is Mandatory?	YES	NOT Mandatory
Supported Topologies	Peer-to-peer (1:1)	Peer-to-peer (1:1), Star topology (many:1), Broadcast (1:many), and Mesh (many:many)
Modulation Technique	GFSK for BR 8-DPSK or π/4-DQPSK for EDR	GFSK
Latency	35ms	2-16ms (Avg. 9ms)

HC-05 Bluetooth Module

Technical Specifications

Power voltage: 3.6 - 6V
Power consumption: maxim 30mA
Voltage level for logic 1: 3.3V - can work with Arduino and with ESP32
UART Serial Communication
Baudrate: 9600 - 460800 bps
Transmission range: up to 10m

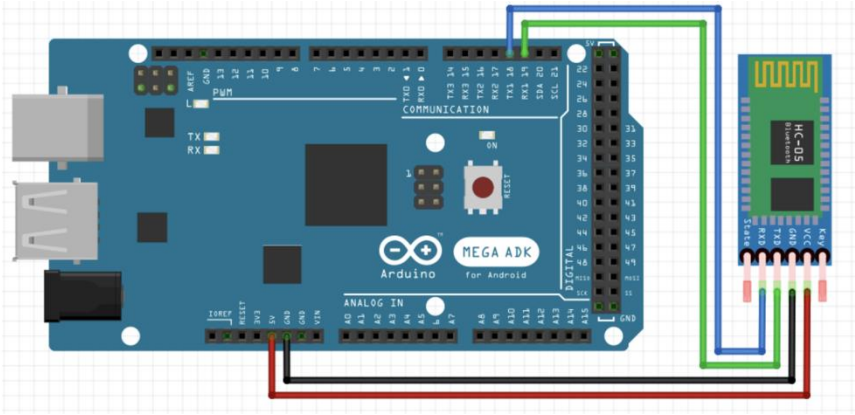
Default pairing PIN : 1234



https://ardushop.ro/ro/comunicatie/1582-modul-bluetooth-hc-05-transciever-serial-6427854023520.html

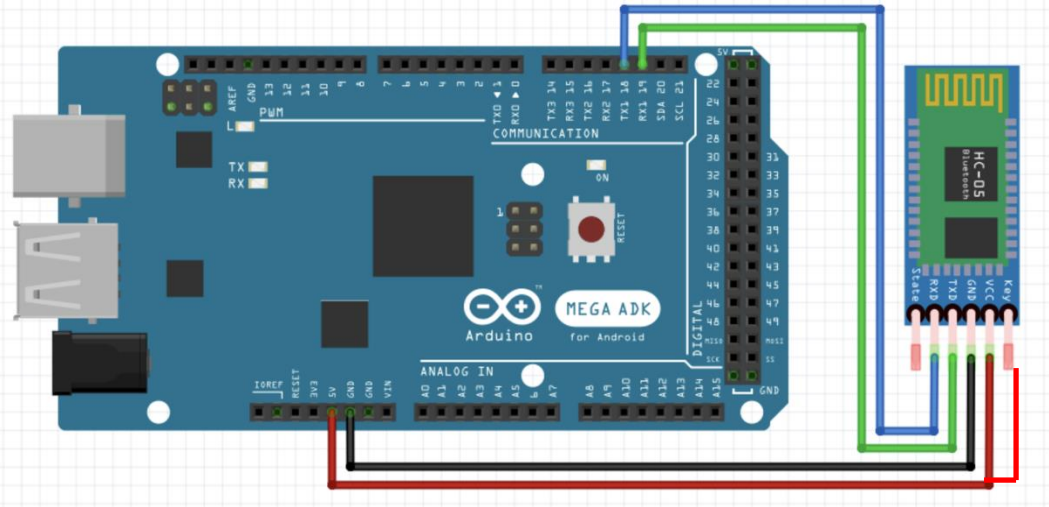
HC-05 Bluetooth Module with Arduino

```
void setup()
{
Serial.begin (9600); // Serial 0 interface for PC
Serial1.begin (9600); // Serial 1 interface for Bluetooth module
}

void loop()
{
if (Serial1.available()) // Read from Bluetooth and send to PC
        Serial.write(Serial1.read());

if (Serial.available()) // Read from PC and send to Bluetooth
        Serial1.write(Serial.read());
}
```



HC-05 Bluetooth Module with Arduino

```
void setup()
{
Serial.begin (9600); // Serial 0 interface for PC
Serial1.begin (38400); // Serial 1 interface for Bluetooth module - default BAUD in config mode
}

void loop()
{
if (Serial1.available()) // Read from Bluetooth and send to PC
        Serial.write(Serial1.read());
if (Serial.available()) // Read from PC and send to Bluetooth
        Serial1.write(Serial.read());
}
```



Connect ‘Key’ or ‘En’ to VCC

HC-05 Bluetooth Module with Arduino

Use ‘Serial Monitor’ or any other terminal to send commands to HC-05 for configuration
Set line endings to Both NL & CR (Newline and Carriage Return)

AT+UART=<baud_rate>,<stop_bits>,<parity>

AT+UART=9600,1,0
9600 Baud, 1 stop bit, no parity

AT+ROLE=<role>

AT+ROLE=0
0 - slave, 1 - master

AT+NAME=<new_name>

AT+NAME=DMP_BT_07

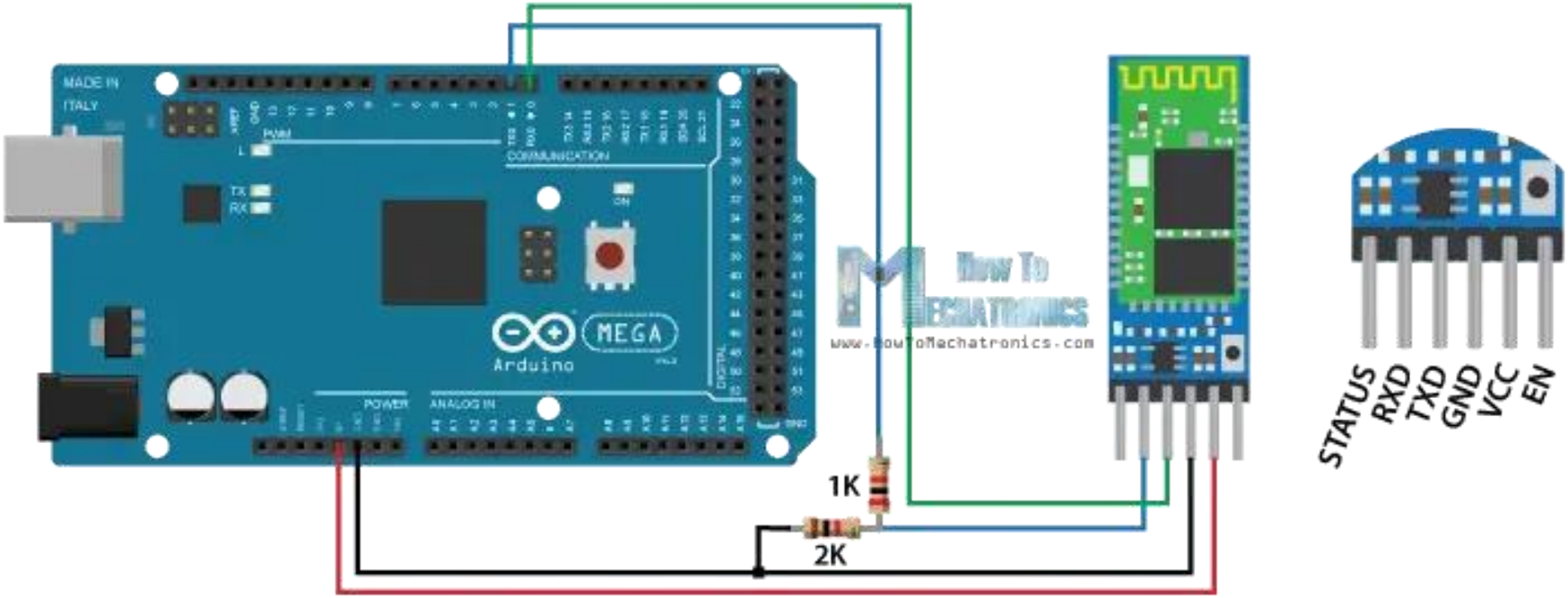
AT+ORGL

Restores factory settings

After sending the AT command, the device should respond with ‘OK’ if no error occurred.

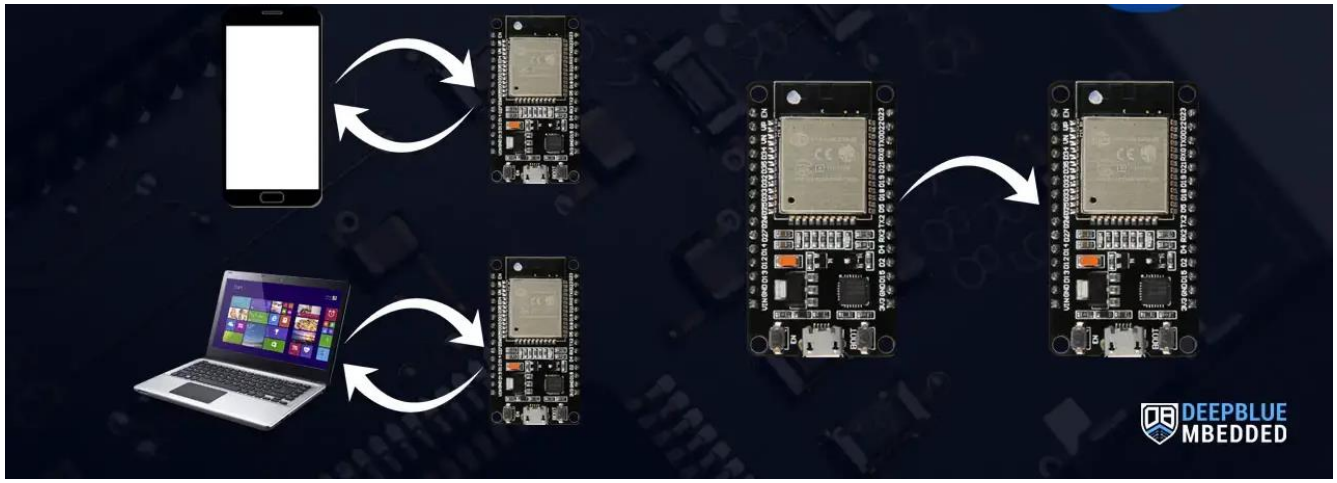
HC-05 Bluetooth Module with Arduino

Use a voltage divider, it's safer!



https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/

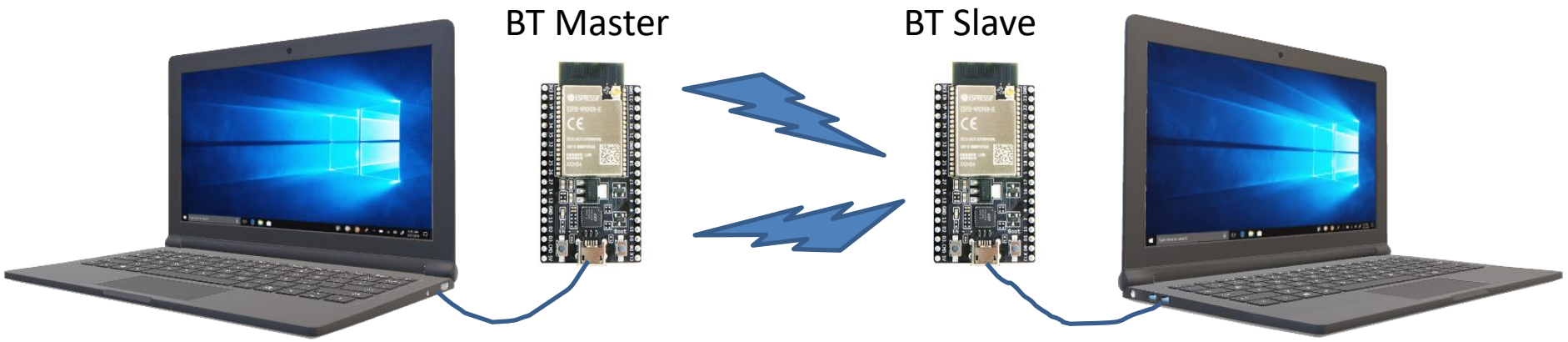
ESP32 Bluetooth



https://deepbluembedded.com/esp32-bluetooth-classic-with-arduino-complete-guide/

- ESP32 includes Classic Bluetooth, and Low Energy Bluetooth (BLE)
- BT can be used to communicate between ESP32 and a PC/SmartPhone (ESP32 as a slave device), or it can be used to communicate with other BT slaves, such as another ESP32 (ESP32 as a master device)
- Wi-Fi and Bluetooth share the same 2.4 GHz radio. When both are active, they use time-division multiplexing to share the RF resource. This may slightly reduce the throughput of Wi-Fi or Bluetooth depending on usage.

ESP32 Bluetooth - master/slave example



- One ESP32 is used as BT master, and one as BT slave
- The master will seek to connect to the slave - it must know the name or the address of the slave and the pairing PIN
- Each ESP32 acts as a transceiver, sending data received from the PC to Bluetooth, and sending the data received from Bluetooth to the PC

ESP32 Bluetooth - code for slave

```
#include "BluetoothSerial.h"

BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  SerialBT.begin(" ESP32-BT-Slave");
  SerialBT.setPin("1234");
}

void loop()
{
if (SerialBT.available()) // Read from Bluetooth and send to PC
        Serial.write(SerialBT.read());
if (Serial.available()) // Read from PC and send to Bluetooth
        SerialBT.write(Serial.read());
}
```

ESP32 Bluetooth - code for master

```
#include "BluetoothSerial.h"

String myName = "ESP32-BT-Master";
String slaveName = "ESP32-BT-Slave";
const char *pin = "1234";

BluetoothSerial SerialBT;

void setup() {
  bool connected;
  Serial.begin(115200);

  SerialBT.begin(myName, true);
  SerialBT.setPin(pin);

  connected = SerialBT.connect(slaveName);
  if(connected) {
    Serial.println("Connected Successfully!");
  } else {
    while(!SerialBT.connected(10000)) {
      Serial.println("Failed to connect.");
    }
  }
}
```

```
void loop() {
  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }
  if (SerialBT.available()) {
    Serial.write(SerialBT.read());
  }
}
```

ESP32 Bluetooth - search for devices

```
#include "BluetoothSerial.h"

BluetoothSerial SerialBT;

#define BT_DISCOVER_TIME  10000
static bool btScanSync = true;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32test"); //Bluetooth device name

  if (btScanSync) {
    Serial.println("Starting discover...");
    BTScanResults *pResults = SerialBT.discover(BT_DISCOVER_TIME);
    if (pResults)
      pResults->dump(&Serial);
    else
      Serial.println("Error on BT Scan, no result!");
  }
}

void loop() {
  delay(100);
}
```

ESP32 Bluetooth - search for devices

https://deepbluembedded.com/esp32-bluetooth-classic-with-arduino-complete-guide/

ESP32 Bluetooth - search for devices

Bluetooth Master:
.connect() and .disconnect()

You can connect to a slave using either a name, or a MAC address

```
connected = SerialBT.connect(SlaveName);     // Use Name
connected = SerialBT.connect(SlaveAddress);  // Or MAC Address
```

disconnect() will close the connection to a slave.

```
// Disconnect() may take up to 10 seconds
if (SerialBT.disconnect()) {
  // Disconnected Successfully!
}
```

ESP32 Bluetooth - event based operation

```
void BT_EventHandler(esp_spp_cb_event_t event, esp_spp_cb_param_t *param) {
  if (event == ESP_SPP_START_EVT) {
    Serial.println("Initialized SPP");
  }
  else if (event == ESP_SPP_SRV_OPEN_EVT ) {
    Serial.println("Client connected");
  }
  else if (event == ESP_SPP_CLOSE_EVT  ) {
    Serial.println("Client disconnected");
  }
  else if (event == ESP_SPP_DATA_IND_EVT ) {
    Serial.println("Data received");
    while (SerialBT.available()) {
      int incoming = SerialBT.read();
      Serial.println(incoming);
    }
  }
}

void setup() {
  SerialBT.begin(device_name);
  SerialBT.setPin(pin);
  SerialBT.register_callback(BT_EventHandler); // Attach The CallBack Function
  ...
}
```

Using BT connections from PCs

- Once you pair your PC (Windows, Linux, Mac, etc) with your BT device (ESP32, Arduino with HC-05, others), you will see the device as a **regular serial port**
- You can use Arduino's Serial Monitor, or any terminal program, to communicate with your device
- You can write programs in any language (C++, Python, Java, etc) and open the port as a regular serial port to send and receive data.