BLE - U (understanding)

•••

It's all pairing things

gatttool char-read-hnd “mr-iot”

A man with big Smile

Agenda..

- IoT intro
- Understanding Microcontrollers in IoT platform
  - ESP32
  - Arduino Software
  - C++ and Arduino programming
  - Flash the devices
- Understanding the IoT Device communication
  - Classic and Smart Bluetooth
  - Develop the code and flashing
  - Communicating with smart phone
- Exploitation
  - Installation of software and hardware
  - Tools for exploiting BLE
  - Sniffing BLE packets using ubertooth/BLE sniffer
  - Analyzing BLE packets on the Wireshark
  - Cracking Encryption
  - Latest BLE Attack

Understanding Microcontrollers in IoT Platforms

- Microcontroller is a devices like personal computers which don't have a complex front end
- Names like MCU(micro controller unit) , MOS(metal oxide semiconductors)

- Mostly Development purpose useful
- Prototyping for projects and many other use cases in real time D

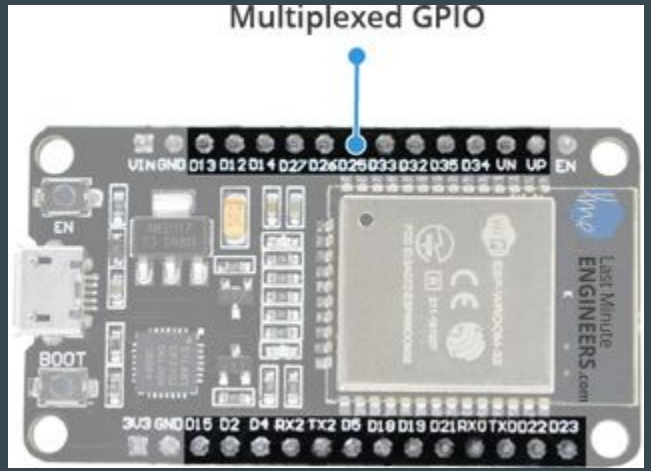
ESP32 -WROOM32

- Developed by Espressif and we can use ESP-IDF (IoT development framework) for menu configuration , then building and flashing the firmware onto an ESP32 board.
- Wi-Fi + BT -BLE - MCU
- Useful in multiple prototype projects
- For more follow this link

Understanding ESP32

Multiplexed I/Os

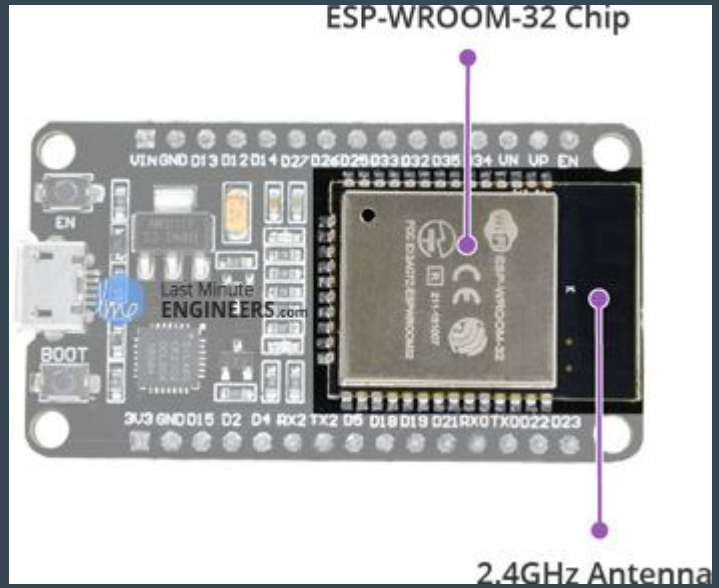
- 15 ADC channels
- 2 UART interfaces
- 25 PWM outputs
- 2 DAC channels
- SPI, I2C & I2S interface
- 9 Touch Pads



Features of ESP-WROOM-32

ESP-WROOM-32 Chip

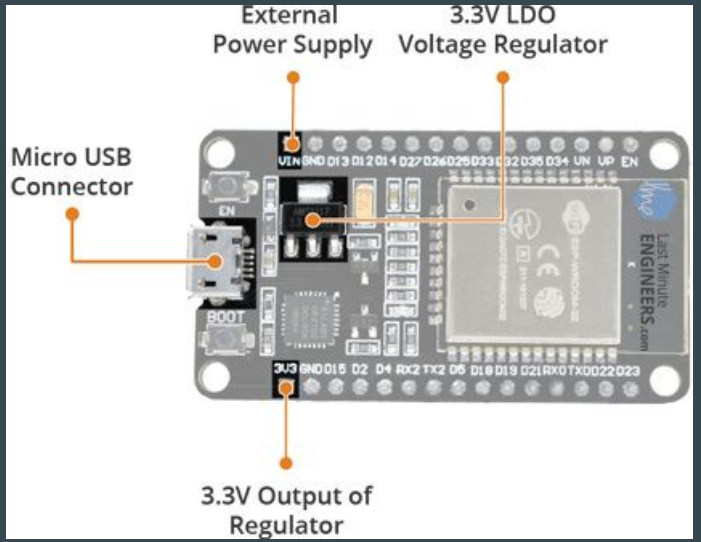
- Xtensa® Dual-Core 32-bit LX6
- Upto 240MHz Clock Freq.
- 520kB internal SRAM
- 4MB external flash
- 802.11b/g/n Wi-Fi transceiver
- Bluetooth 4.2/BLE



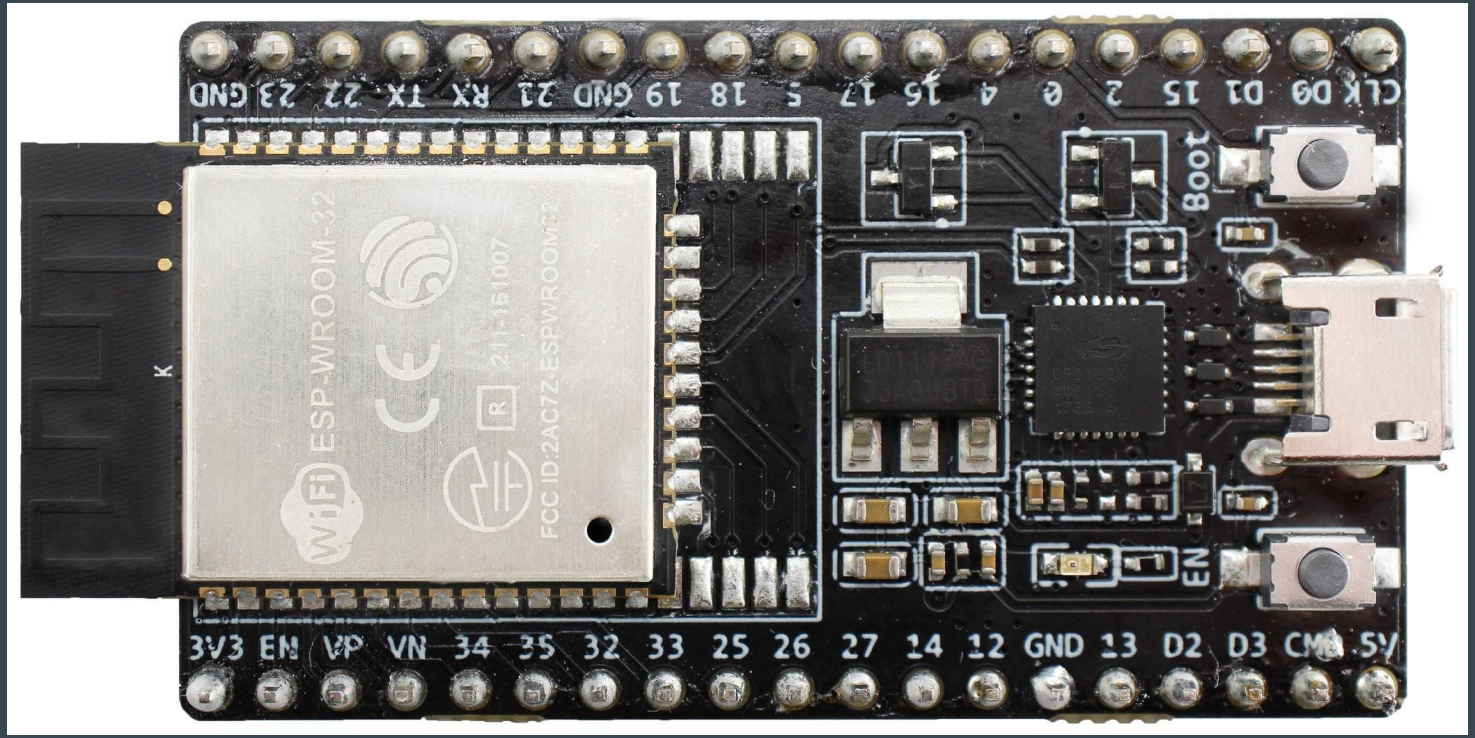
Understanding the ESP32

Power Requirement

- Operating Voltage: 2.2V to 3.6V
- On-board 3.3V 600mA regulator
- 5 µA during Sleep Mode
- 250mA during RF transmissions



ESP32 Dev Kit

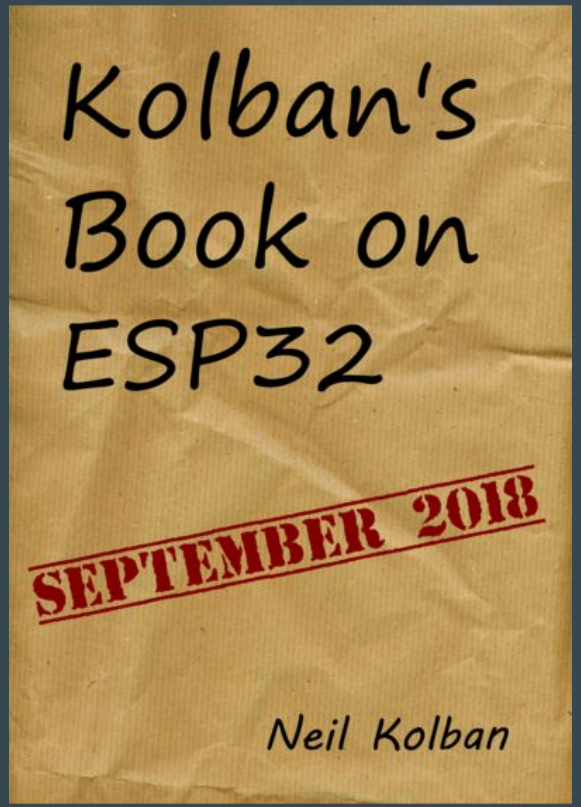


A best book to follow

https://leanpub.com/kolban-ESP32

https://github.com/nkolban

https://www.youtube.com/channel/UChKn_BlaVrMrhEquPNI6HuQ



Arduino IDE(.ino)

- It is open-source arduino software(IDE)
- Helps us to create code and flashing into the devices
- Required to languages like embedded c and c++
- Arduino coding is unique will help us to understanding devices features and communications and working of it
- Mostly saves in “.ino” format

C ++ programming (.cpp)

Not much important today

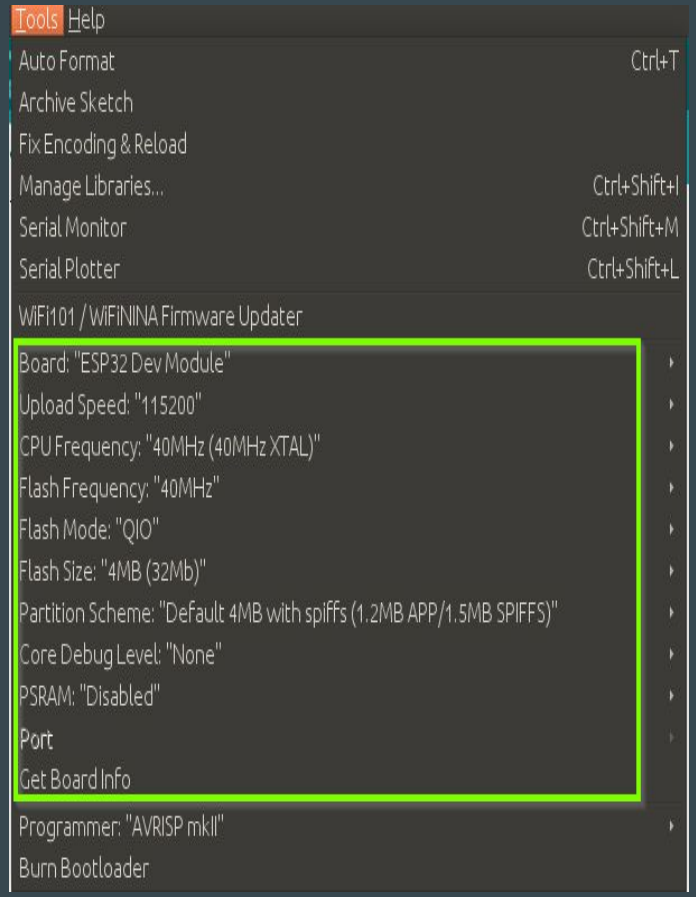
But still we use some of the concepts will discuss in coding time

Flashing the devices

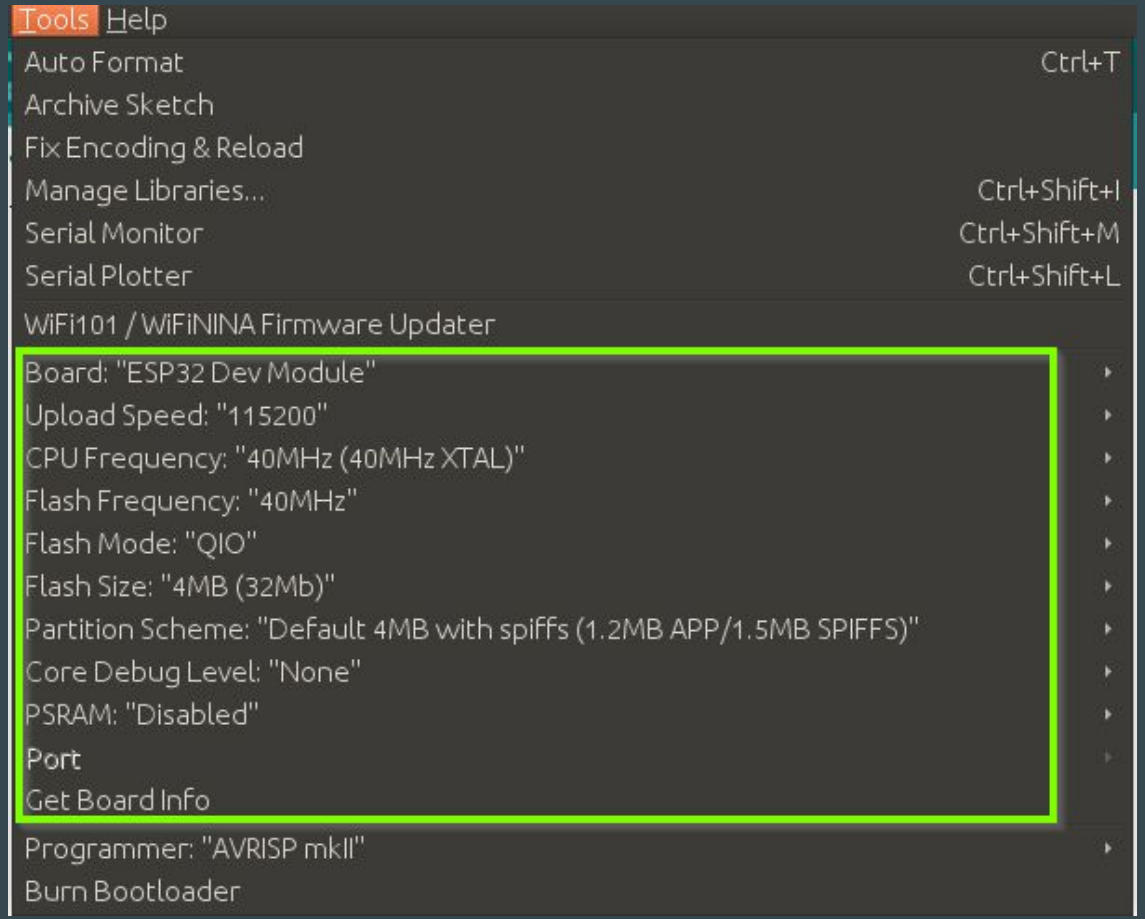
- Flashing the devices just click this symboles was enough ?

“No”

- Check the tools location in arduino software whether device got connected properly or not



Configurations before flashing the device



Understanding IoT devices communication

There are some wired and wireless communication helps to IoT devices communicates
Most of them still insecure

Reasons can be anything for insecurity of device

1. CODE
2. COMMUNICATION
3. CONFIGURATION
4. ********** ONCE EXPLOIT IS CAME OUT

Bluetooth History

. What is bluetooth ?

Bluetooth is a wireless technology standard for exchanging data between fixed and mobile devices over short distances using short-wavelength UHF radio waves in the industrial, scientific and medical radio bands, from 2.400 to 2.485 GHz, and building personal area networks (PANs). It was originally conceived as a wireless alternative to RS-232 data cables.

Nokia originally developed BLE for an in-house project called ‘WIBREE,’ which was later on, taken over by the Bluetooth SIG. BLE was conceived with an emphasis on better pairing speed and energy efficiency.

Classic

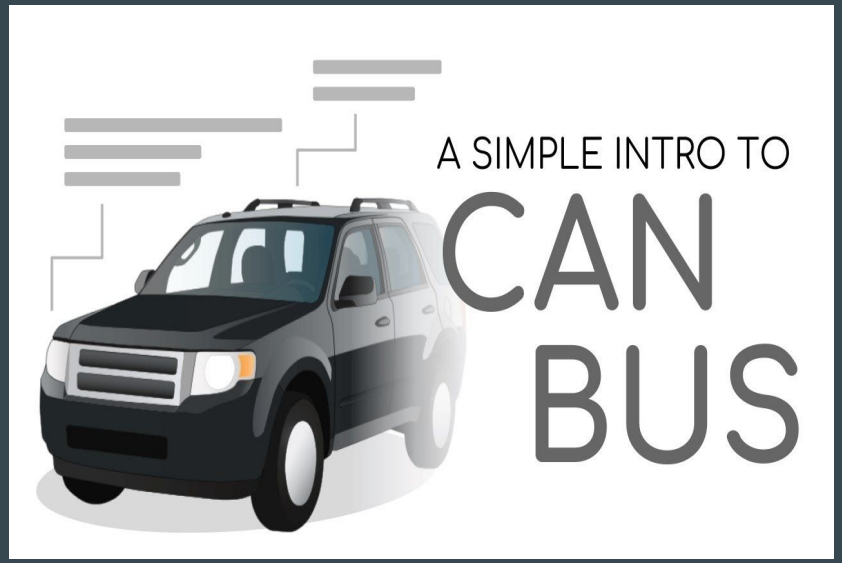
```
#include "BluetoothSerial.h"

#if !defined(CONFIG_BT_ENABLED) ||
!defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please
run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  delay(5000);
  Serial.println("Starting..");
  SerialBT.begin("MyESP32"); //Bluetooth
device name
  Serial.println("The device started, now
you can pair it with bluetooth!");

}

void loop() {
  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }
  if (SerialBT.available()) {
    Serial.write(SerialBT.read());
  }
  delay(20);
}
```

Vs

LE

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>

// See the following for generating
UUIDs:
// https://www.uuidgenerator.net/

#define SERVICE_UUID
"4fafc201-1fb5-459e-8fcc-c5c9c3319
4b"
#define CHARACTERISTIC_UUID
"beb5483e-36e1-4688-b7f5-ea07361b2
a8"

void setup() {
  Serial.begin(115200);
  Serial.println("Starting BLE
work!");

  BLEDevice::init("Long name works
now");
  BLEServer *pServer =
BLEDevice::createServer();
  BLEService *pService =
pServer->createService(SERVICE_UUID
);
  BLECharacteristic
*pCharacteristic =
pService->createCharacteristic(

CHARACTERISTIC_UUID

BLECharacteristic::PROPERTY_READ |

BLECharacteristic::PROPERTY_WRITE

);

  pCharacteristic->setValue("Hello
World says Neil");
  pService->start();
```

Know about ESP32 WROOM 32 ...



Bluetooth Stack

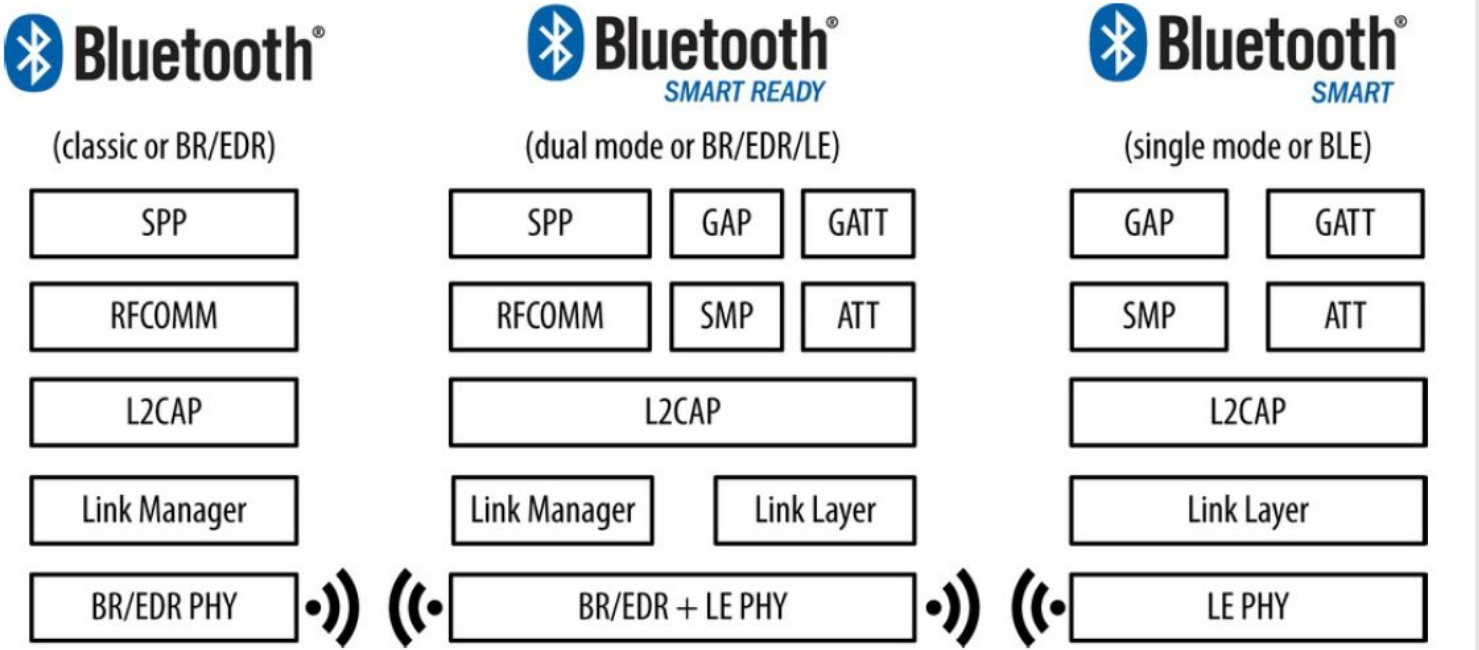
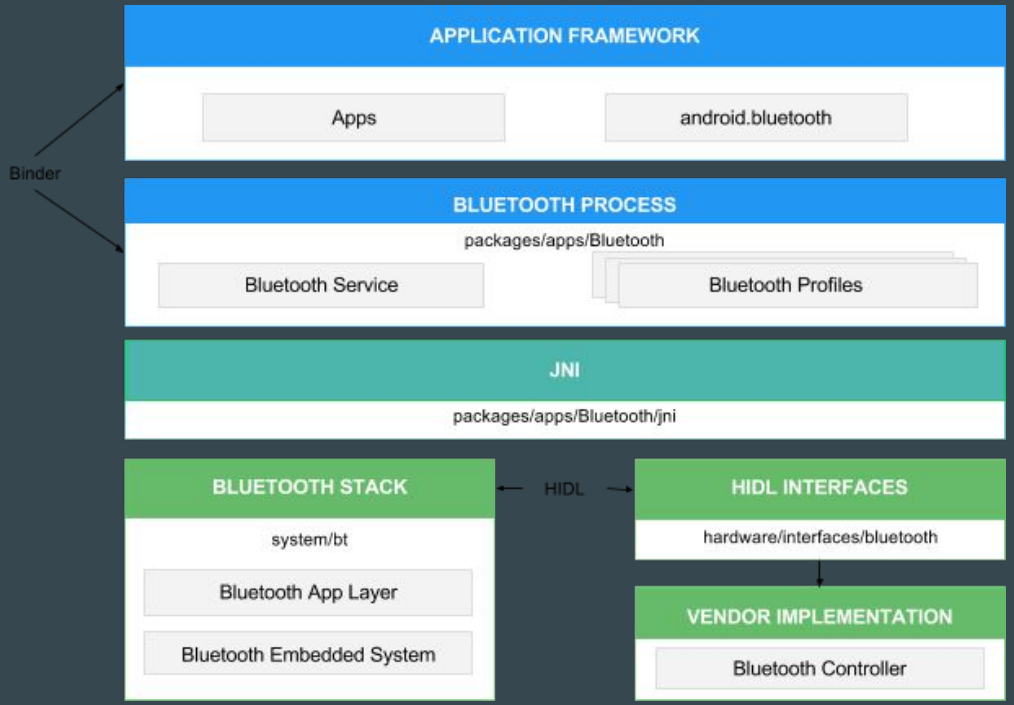


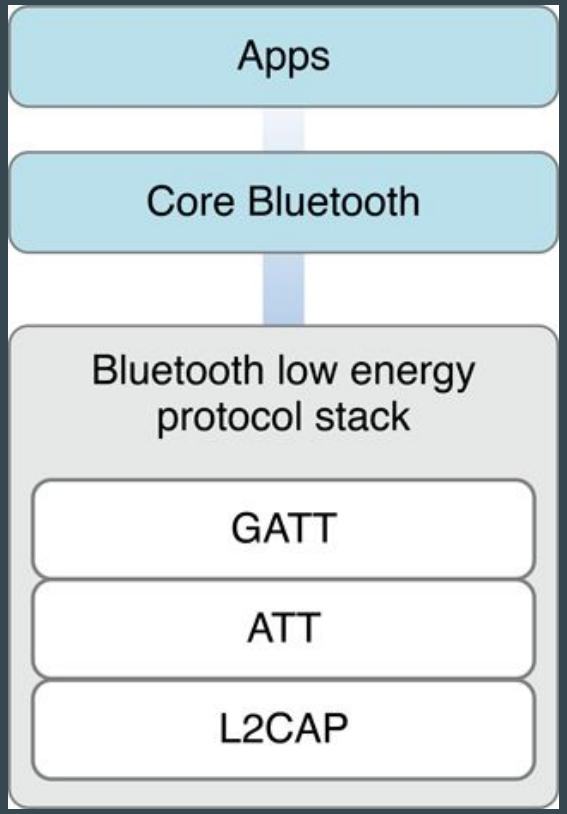
Figure 1-1. Configurations between Bluetooth versions and device types

Android Bluetooth Stack - Android 8.0



https://source.android.com/devices/bluetooth

iOS bluetooth (core)



Code Developed

- Open provided OVA file
- Goto Session folders

- Android , iOS , arduino and bluetooth codes already available

Communicating with smart phone

- Nrf Connect android app (desktop app)

Some Test Cases

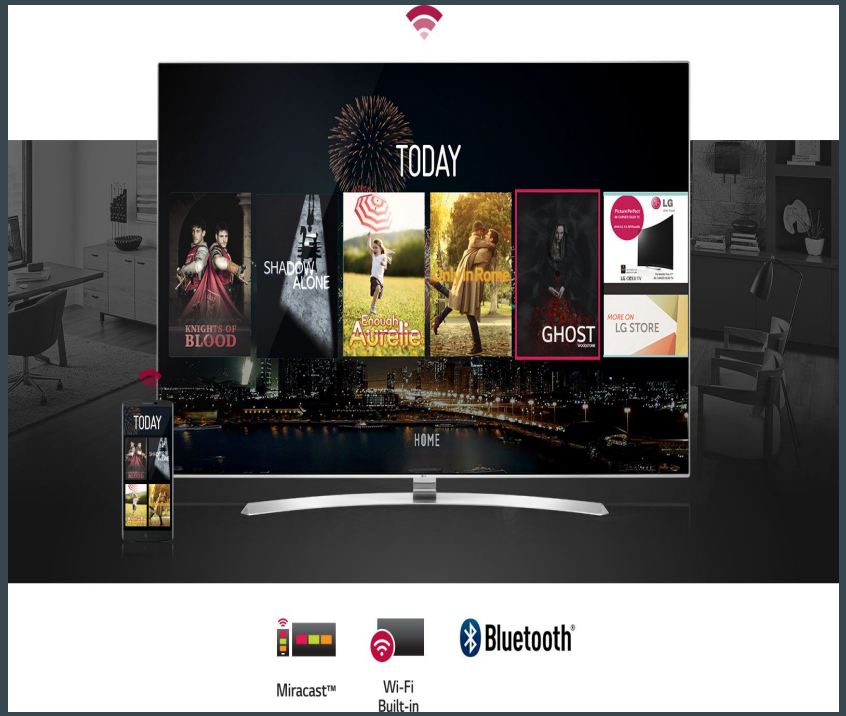
Where all we using - medical

BLE BP Monitors

Mindwave products

Smart wearables

BLE Beacons and TVs



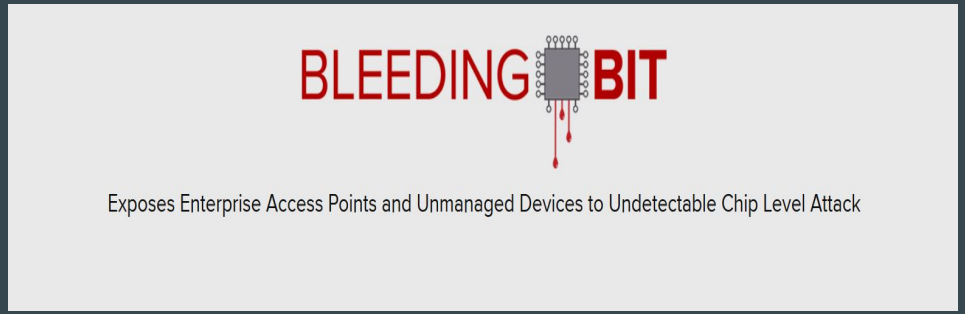
Helmets

Product Specs

- MESH INTERCOM™
- BLUETOOTH 4.1
- INTERCOM 2 KM / 1.2 MILES
- MULTI-WAY INTERCOM
- AUDIO MULTITASKING™
- BUILT-IN SPEAKERS & MIC
- ADVANCED NOISE CONTROL™
- VOICE COMMAND
- FM RADIO
- REMOTE CONTROL COMPATIBLE

Most Interesting - YULU in Bangalore

Test Cases about BLE



Xiaomi M365 Electric Scooter Hacked and Remotely Controlled

Burning down house with IoT

https://www.pentestpartners.com/security-blog/burning-down-the-house-with-iot/

A walkthrough on the Blueborne Attacks

Android Bluetooth - 'Blueborne' Information Leak (1)

EDB-ID:	CVE:	Author:	Type:	Platform:	Published:
44554	2017-0781	KERT OJASOO	REMOTE	ANDROID	2017-08-09

EDB VERIFIED: ✕

EXPLOIT: / {}

VULNERABLE APP:

LineageOS 14.1 Blueborne - Remote Code Execution

EDB-ID:	CVE:	Author:	Type:	Platform:	Published:
44415	2017-0781	MARCIN KOZLOWSKI	REMOTE	ANDROID	2018-04-06

EDB VERIFIED: ✕

EXPLOIT: / {}

VULNERABLE APP:

What is BlueBorne ...!

BlueBorne is an attack virus that spreads through air and gets into a device via bluetooth and can then take full control of the device. The targeted device does not need to be paired to the attacker's device or even to be set on discoverable mode. If your bluetooth is on and you are in vicinity of already infected device, then the attack virus will get easily transferred to your device without asking for any permission. Thus, it needs zero human interaction and no internet connection.

marcinguy / android712-blueborne

Watch 3 | Star 50 | Fork 13

Code | Issues 6 | Pull requests 0 | Projects 0 | Security | Insights

Dismiss

Join GitHub today

GitHub is home to over 36 million developers working together to host and review code, manage projects, and build software together.

Sign up

Android Blueborne RCE CVE-2017-0781

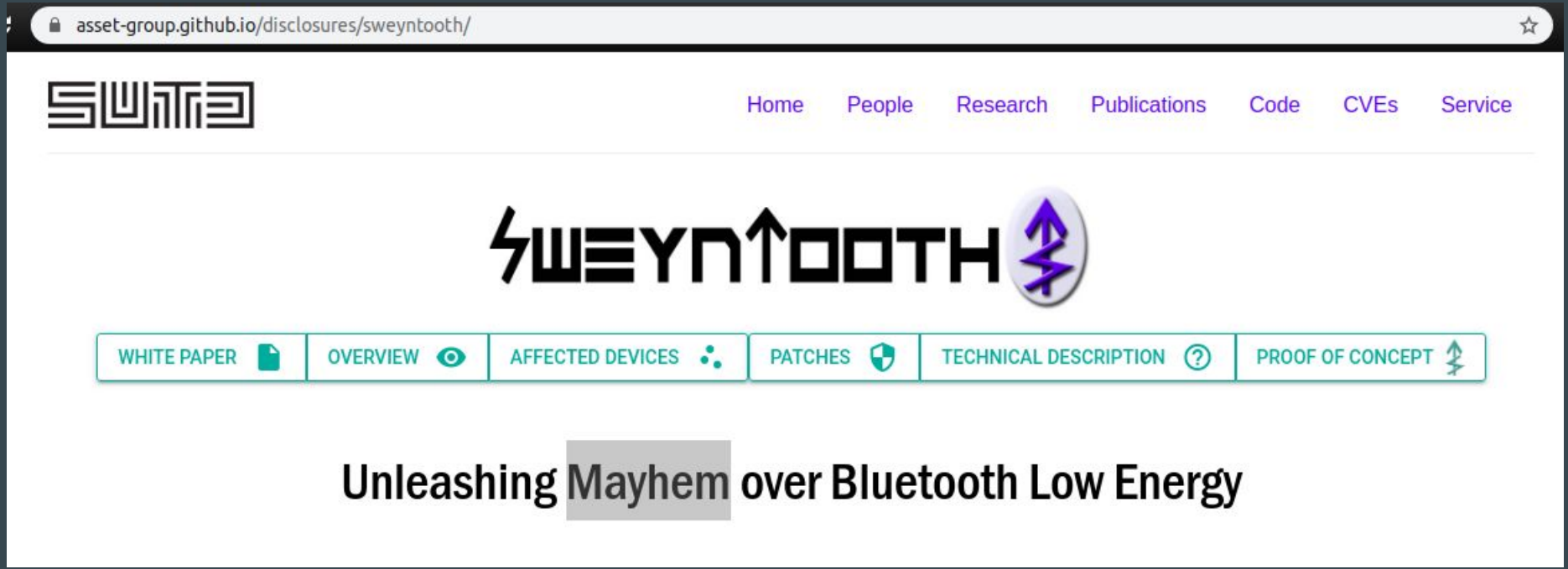
31 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Find File | Clone or download

marcinguy Update README.md — Latest commit 0c764a5 on Apr 4, 2018

README.md	Update README.md	a year ago
bluedroid.py	Add files via upload	a year ago
btsock.py	Add files via upload	a year ago
connectback.py	Add files via upload	a year ago
exp4.py	Update exp4.py	a year ago

A Dozen Vulnerabilities



Bluefrag

BlueFrag (CVE-2020-0022): a critical bluetooth vulnerability in Android

February 13, 2020

Security researchers at ERNW disclosed a vulnerability in **Android** bluetooth stack that lets attackers silently deliver malware to and steal data from nearby phones simply knowing the **Bluetooth MAC** address of the target (easy to guess just by looking at the **WiFi MAC** address).

Resources

https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/bluetooth

https://www.systutorials.com/docs/linux/man/8-hciconfig/

https://www.systutorials.com/docs/linux/man/8-hciconfig/

http://ianharvey.github.io/bluepy-doc/index.html

https://www.elinux.org/RPi_Bluetooth_LE#Using_Bluetooth_LE_with_Python