



Extract firmware for fun and non-profit

IOT VILLAGE CFP 2022

SIDDHARTH REDDY T

whoami

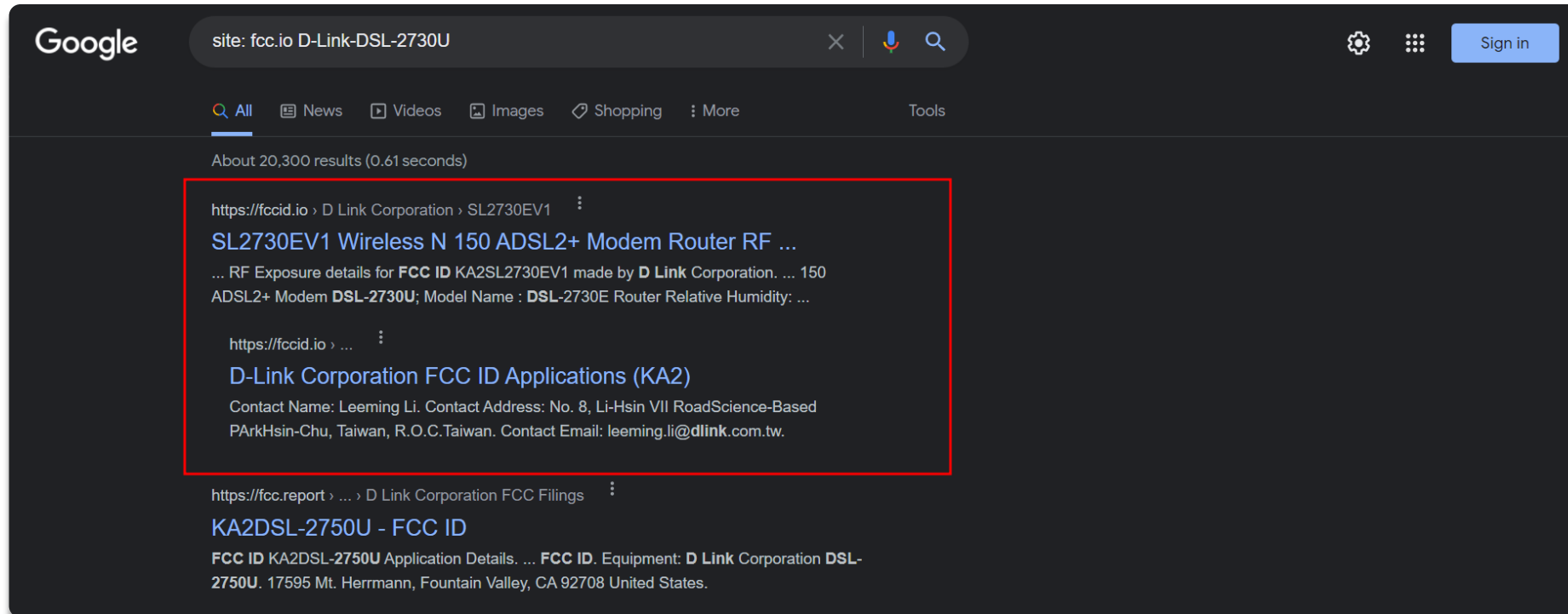
- Hobbyist hacker from India, 20 years old
- Mostly Reverse Engineering, all the bytecode and assembly
- Bachelors student in Mechatronics
- Started playing CTF in 2021
- @Sidd_Tim (DM's are open)

Agenda

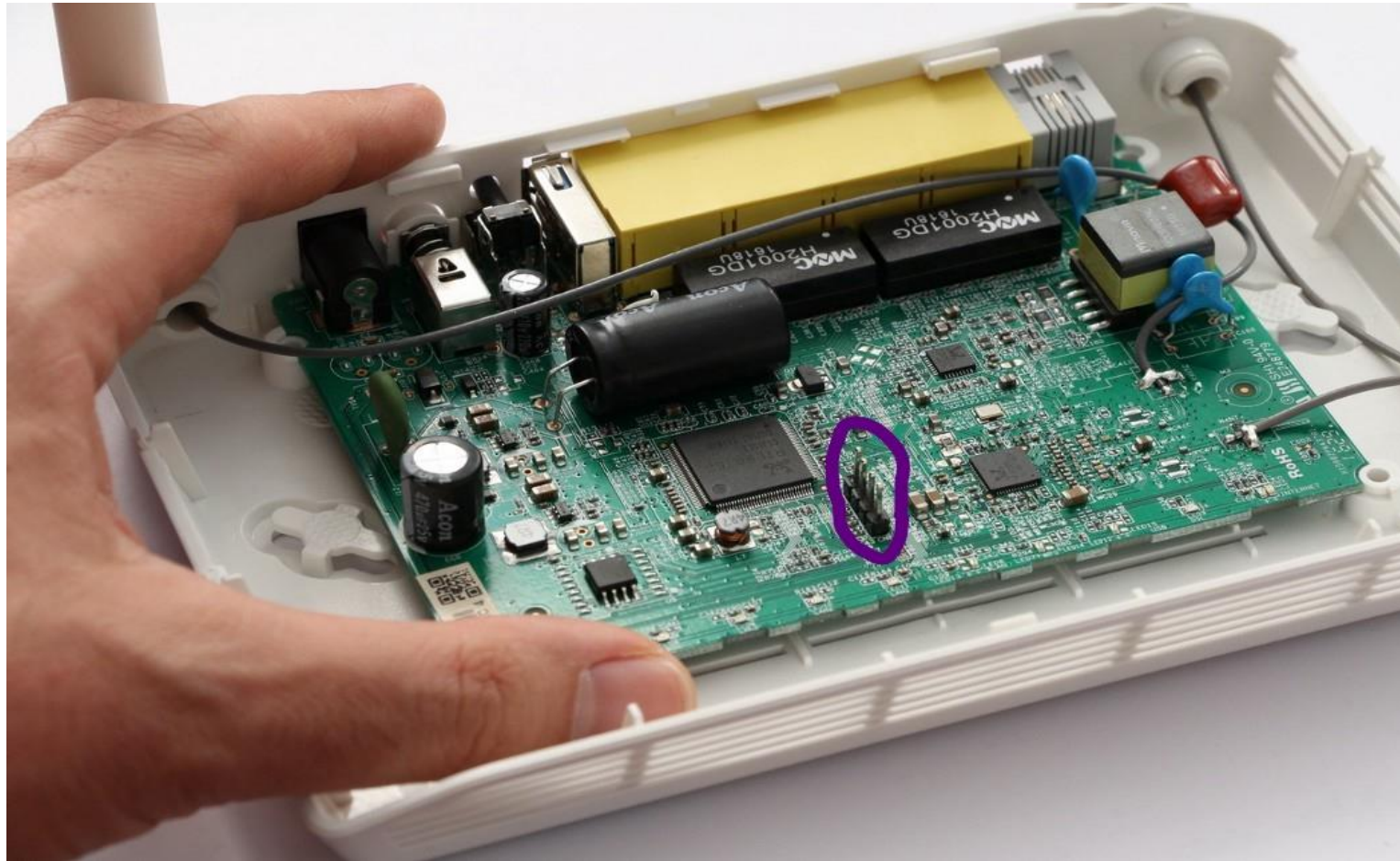
- Information Gathering
- Basics of Serial Communication
- Interacting with this protocol
- Extracting firmware out of it

Information gathering

- Whenever you decide to attack some IOT device, we need to gather as much information as possible, from both **hardware** and **software** perspective. As the saying goes, *Better the enumeration, so are vulnerabilities*.
- When it comes to hardware, we need to take a look at its FCC ID, which describes each and every aspect and specification of that particular thing. Usually found on back of your device. FYI, every wireless communication device in US must have **FCC ID (Federal Communications Commission)**. If you are from other country, try using **google dorks** such as `site:fcc.io "{your device}"`. For example, `site: fcc.io D-Link-DSL-2730U`.



Gathering Info using google dorks



After which, you need to crack open the router, to see if there is some sort of juicy info such as finding UART, JTAG serial port. In my case, I found **UART interface**. The purple mark above is the UART. I'll explain regarding serial communication in the next slide.

Basics of Serial Communication

- In a nutshell, Serial communication is a way by which two chips or devices can communicate with each other
- It has receive and transmit channel called **Rx** and **Tx**
- **Rx** to Receive and **Tx** to Transmit the data
- Few serial communication protocols are **UART** or **Async serial**, **JTAG (Joint Test Action Group)**, **I2C**, **SPI (Serial Peripheral Interface)**
- Everything has its own pros and cons

Interacting with UART

- We can identify UART interface by just having a look at the pins mainly **Rx**, **Tx**, **Gnd**, **Vcc**, not necessarily in that order, we can identify this using multimeter. Sometimes, if lucky enough these pins will be labelled on the PCB. For me, that was the case.
- Once you have identified the interface, you need to use usb to ttl converter to interact with the router.
- After which we need to determine the baud rate, which in my case was **115200**, we can determine the baud rate using trail and error or obtain a sample using **Saleae Logic analyzer** and use **Saleae Logic** to determine the baud rate.

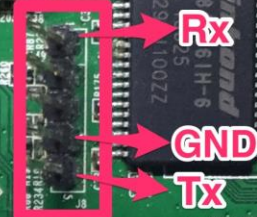
Main IC

Ralink
RT3352F
TPSA611709
1333PTA3

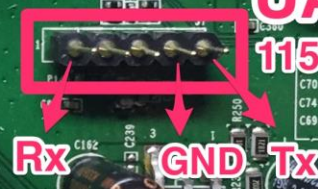
2nd IC

Trend Chip
TC3162LE-401289
D7GNP0009
UA1337F3-M6

UART1
57600 bps



UART2
115200 bps



ETON ET866
9A 94V-0

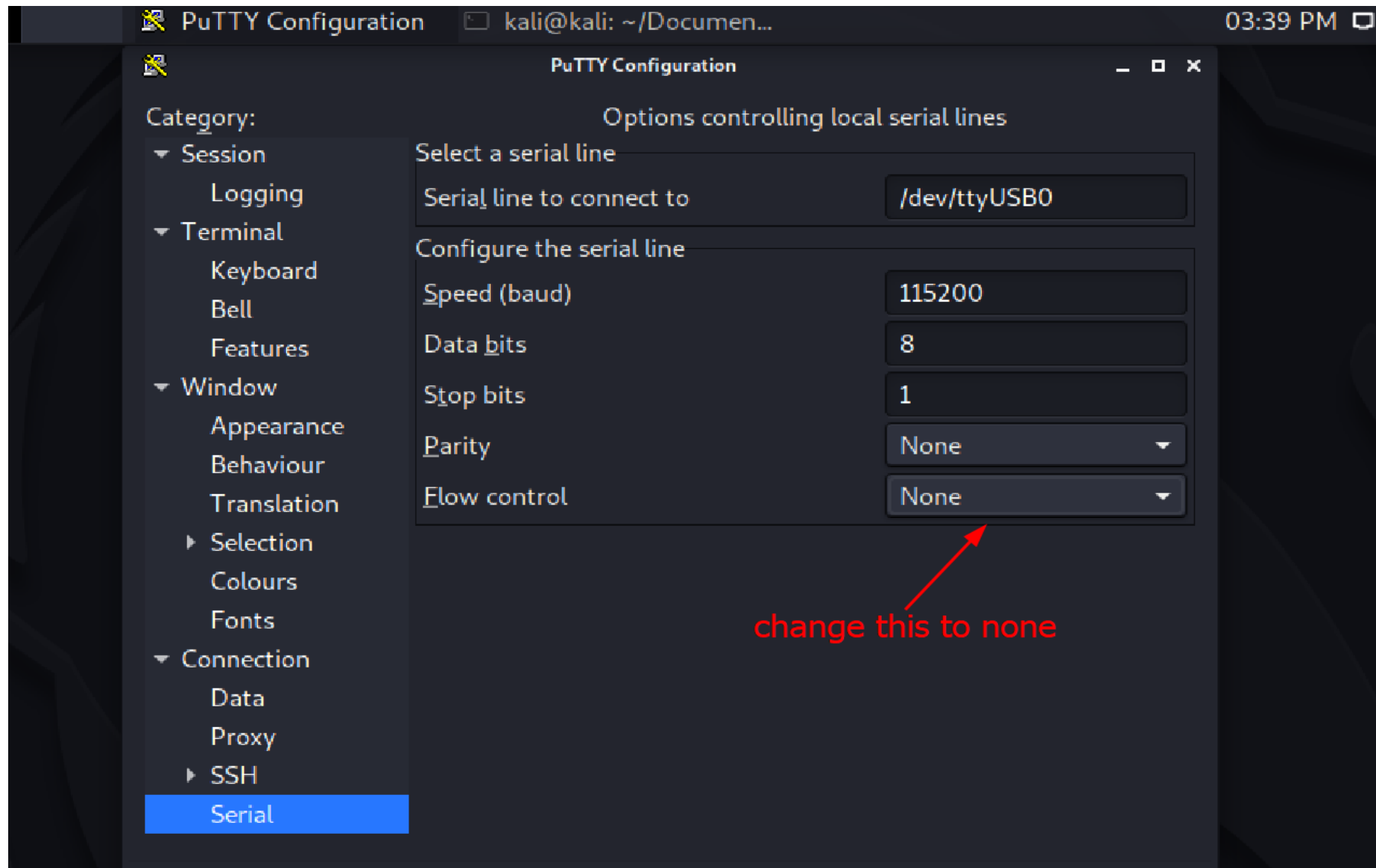
P416062A

MOC
H2002DG
1341D

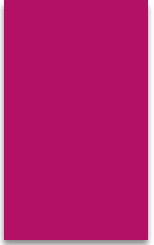
MOC
H2002DG
1341D

MOC
EP-132DG
1334

2074



You can interact it using **puTTY**, **minicom** or **screen**. But personally I like using **puTTY**. After which we are going get a console.



```
ADSL#show version
```

```
Application Version: IN_R_01.00.01  Build Date:  Nov  3 2016 17:27:21
SDK Version:          SDK V1.1.1    Build Date:
Hardware Version:      I1-2/8        MAC Address:  0C:B6:D2:BB:1E:41
SysUpTime:            0 0:7:47      Serial Number: 0CB6D2BB1E41
ManufacturerOUI:      0CB6D2        Manufacturer:  DLINK
ModelName:            DSL-2730U      Description:   DSL-2730U
ProductClass:         IGD           CPU Speed:     448.92 Mhz
Flash Memory:         2M
RAM Memory:           8M
ADSL#
```

But sometimes if lucky, you might directly be able to interact with OS. Let's see above that in the next part of this slide.

Extracting firmware out of it

There are 3 ways by which we can extract the firmware out of it.

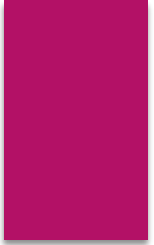
- Using **TFTP** (Trivial File Transfer Protocol)
- Dumping from **bootloader**
- Extracting firmware using **SPI flash**, which is a traditional way, so I'll not be going much deep into that

Using TFTP

- TFTP is a **UDP** protocol, **port 69**, which is used to upload firmware to the router but you can also extract firmware out of it, if the developer enables it from the bootloader by setting up TFTP server and usually the firmware will be named as **MODEL-firmware.bin**.
- You can use **TFTP32** on windows to do this.
- In my case, it was an old router and it was not possible to do it.

Dumping firmware from bootloader

- Inorder to dump firmware from bootloader, we need to somehow have access to the bootloader. Well, you can get access to the bootloader by using **esc**. Sometimes it will be password protected, you can use **Fault Injection** to bypass this. But its kind of time consuming.
- You can find the extracted firmware in my github, I'll provide all the links to this at the end of the slide.
- You can see that we have entered into bootloader, which will be showcased in the next slide.



```
load [address]
xmodem [address]
tftp [ip] [server ip] [file name]
tftpbk [size] [clientip] [serverip] [filename]
multicast
flashsize [256(k)/128(k)/1(M)/2(M)/4(M)/8(M)/16(M)]
memsize ROW[2k/4k/8k/16k] COL[256/512/1k/2k/4k] BANK[2/4]
uart [0(enable)/1(disable)]
<RTL867X>info
(c)Copyright Realtek, Inc. 2012
Project RTL8676S LOADER (LZMA)
Version 00.01.02a-rc (Apr 24 2014 13:41:19)
```

```
<RTL867X>BootLine:
MAC Address [0]: 00:23:79:11:22:33
Entry Point: 0x80000000
Load Address: 0x80000000
Application Address: 0x9FC10000
Flash Size: 16M
Memory Configuration: ROW:4K COL:512 Bank:4Banks
MII Selection: 0 (0: Int. PHY 1: Ext. PHY)
UART is enabled
```

```
<RTL867X>■
```



Here, we are interested in **entry address: 0x80000000** and above we can see different commands and we have to use **d** to dump the firmware, you might have thought, how much should we dump? Well, you have to dump about **16M**. So we can use **d 0x80000000 2097152** to dump the firmware.

Incase you are confused about the address, its where the **SPI flash** is.

Well, the firmware uses a compression algorithm called **LZMA**, so you need to search for its **magic bytes**, which is **5D 00 00 80**. After which, the firmware will be unpacked.

You might even ask, until which address. Well, it depends. Usually you need to stop when you see a lot of null bytes but still might not be sure because there is a lot of room of null bytes. You need to trial and error to find this out.

Extracting firmware using SPI flash

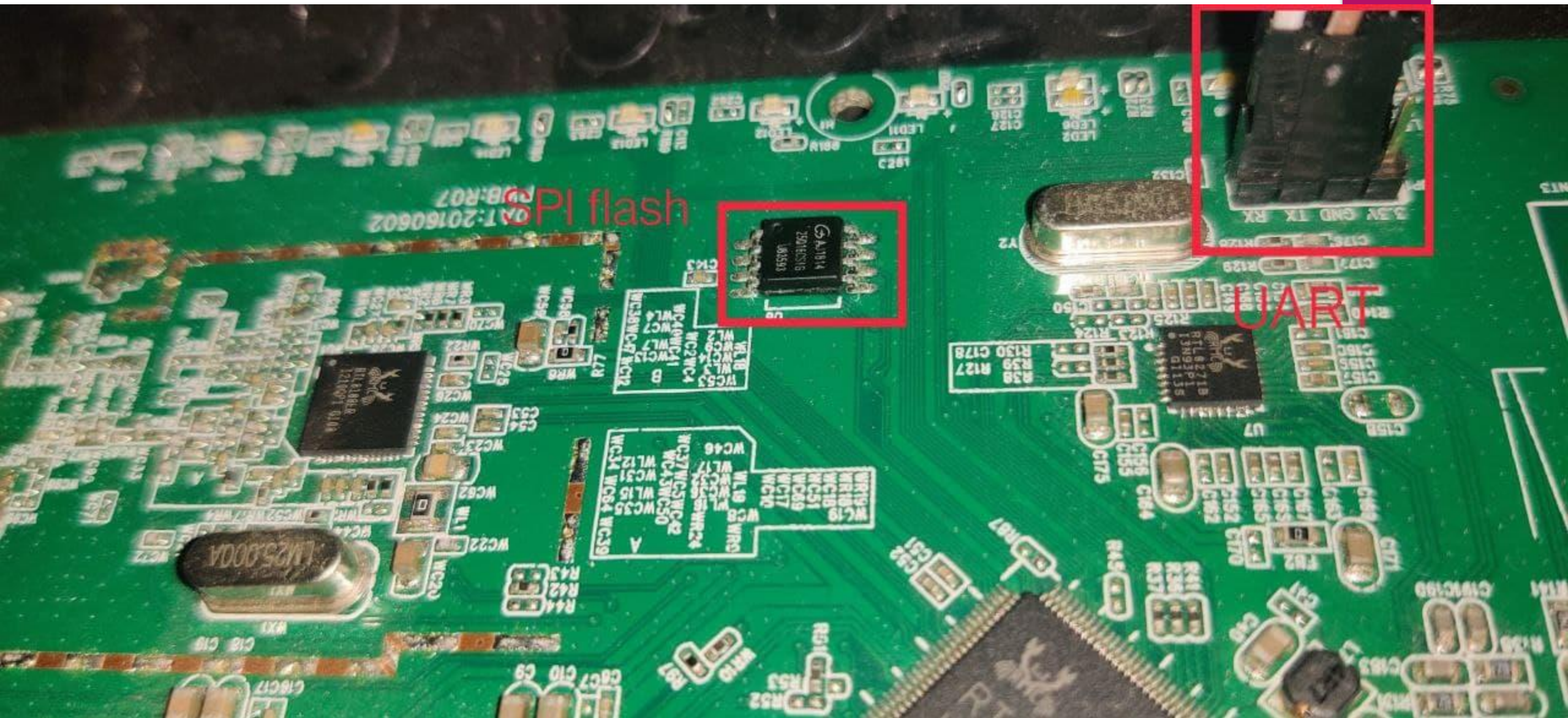
Since this is a legacy method I'll not be discussing much on this.

This method is very popular in dumping firmware from an IOT device. For this, we need to locate where the **SPI flash** is. Usually, it is present nearby to **UART** serial port or processor. I'll show this in the next slide.

All you need is a **Programmer** and **JMT-SM2** wire connector and use **flashrom** or **UEFITool** to extract the firmware out of it.

If you find a **JTAG** interface, you can use **Jtagulator** and select few options related to it and then you are good to go.

You can also dump from **UART** or **SPI** using bus blaster and then connect it to your PC using a USB and open **Jtagulator** and select few options such as which **protocol** to be used and it will do its job.





The best part is all the tools I've mentioned are Open-Source.

Project

The images, firmware and all the scripts I've used are in my [Github](#).



Thank you