Payatu

IoT Security-Part 14 (Introduction To And Identification Of Hardware Debug Ports)

Shakir
26-September-2020

IoT Security-Part 14 (Introduction to and Identification of Hardware Debug Ports)

This blog is part of the IoT Security series where we discuss the basic concepts pertaining to the IoT/IIoT eco-system and its security. If you have not gone through the previous blogs in the series, I would urge you to go through those first. In case you are only interested in the Hardware debug ports, feel free to continue.

IoT Security - Part 1 (101 - IoT Introduction And Architecture)

IoT Security - Part 13 (Introduction To Hardware Recon)

In the previous blog, we discussed how to perform reconnaissance on hardware. In this blog, we will discuss Hardware debug ports, what are the different debug ports, and how to find debug ports using different methods. Finding the hardware debug port process consists of several steps like looking at the various test pads, pinouts, headers. Rarely these pins are labeled on PCB so we have to identify pins manually. Information from the microcontroller datasheet about pins and simple continuity check using a multimeter is good enough for IC packages like DIP/TSOP, TQFP (for more information about IC packages refer this blog IoT Security-Part 13 (Introduction To Hardware Recon) but for packages like QFN, BGA, etc. where pins are not accessible on PCB it is difficult to identify manually. For such cases where continuity check is not possible, tools like Bus Auditor or Jtagulator come very handy. Once JTAG/SWD, UART, I2C pins are identified on PCB, access to the microcontroller, firmware extraction, root shell access is possible.

Why do PCBs have to debug ports?

The hardware debug ports can directly talk to the microcontroller. JTAG/SWD port are used during development to download, debug firmware source code, recover bricked hardware, perform boundary-scan. A boundary scan is a method for testing interconnections on a printed circuit board (PCB). It's the easiest way to troubleshoot the device. On the other hand, UART ports are typically used to access the device via a console for debugging the application running on the hardware. In this blog post, we are going to find UART, JTAG/SWD, I2C on DIVA Board using Bus Auditor.

Bus Auditor:

Bus Auditor (https://expliot.io/products/bus-auditor-pre-order) is a compact multi-protocol tool used to scan and identify debug ports and communication interfaces exposed on any hardware board, it can brute force several hardware protocols including JTAG, Arm SWD, UART, I2C. This device has 16 independent channels, to connect to the header pins on the target PCB. The inbuilt USB port is used for interfacing with EXPLIoT framework (Internet of Things exploitation framework - https://gitlab.com/expliot_framework/expliot) running on the pc.

Fig 1. Bus Auditor

Features :

- USB 2.0 high-speed interface.
- IEEE 1149.1 JTAG, and arm SWD support.
- UART RX, TX pins and Baud rate detection.
- i2c pin and i2c address detection.
- Adjustable target voltage level 1.2V-3.3V.
- 16 I/O channels with level transition and input protection.

So, let's have a look at the types of debug ports and identify it using Bus Auditor.

Note: 16 IO channels are labeled as CH0 to CH15 for easy identification.

DIVA:

DIVA (Damn Insecure and Vulnerable application) IoT Board (https://expliot.io/products/diva-iot-board) is one of the EXPLIoT products. DIVA Board is a connected IoT device and a vulnerable target board designed to teach the basics of IoT security.

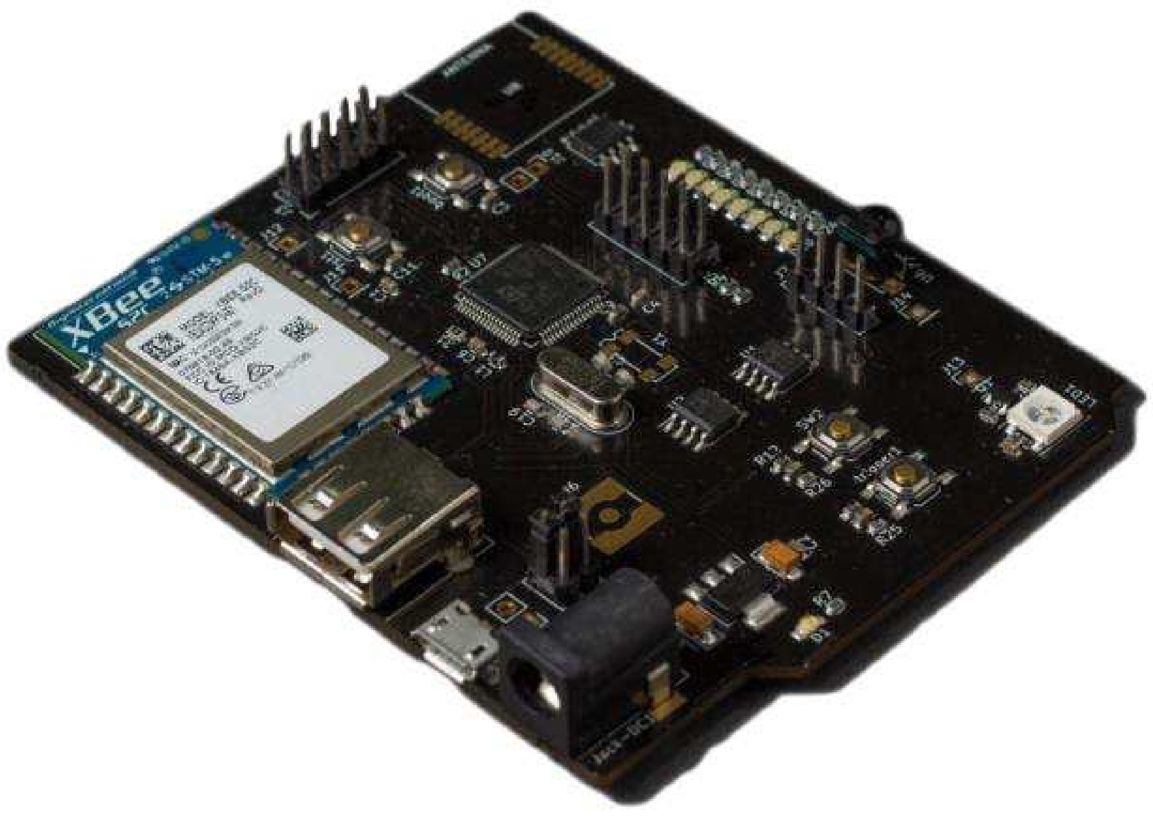


Fig 2. DIVA

The board provides a standard JTAG debug interface as well as an SWD interface that can be used as a debug port for firmware debugging and a UART port for a custom console for debug messages on the terminal. There are 3 breakout headers on the DIVA board and we will scan the breakouts and we will find the debug ports.

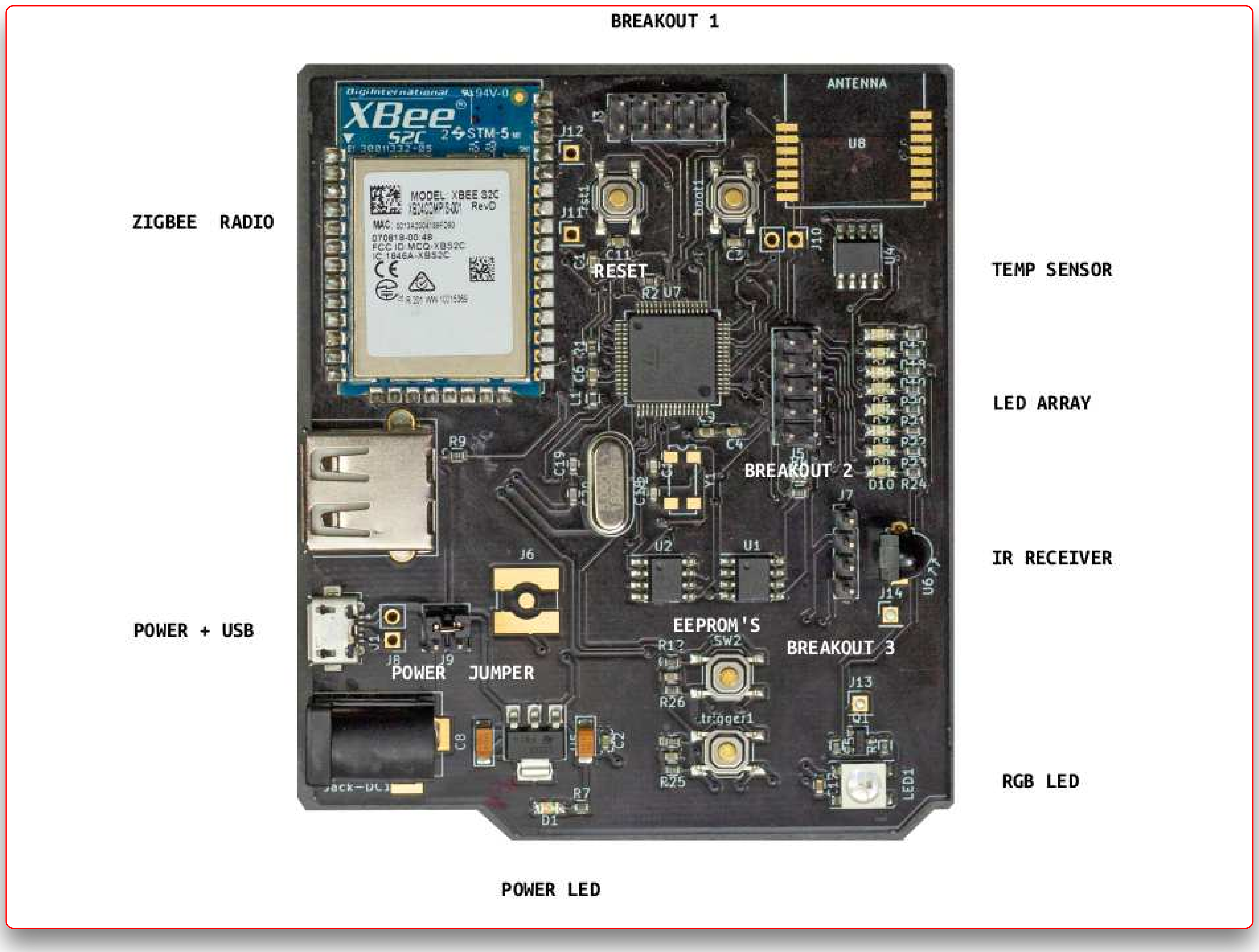


Fig 3. DIVA Board Layout

We can see the DIVA board layout in the photo with all peripherals and breakouts marked. we will identify JTAG/SWD, UART, I2C ports on breakout1, breakout2, and breakout3.

JTAG (Joint Test Action Group):

Nowadays PCBs and SOCs have become smaller and advanced. Over time PCBs have complex and difficult to debug and program after the production. To make it easy for testing, debugging, and programming the boards post-production, the industry developed an interface called Joint Test Action Group(JTAG). It is used for testing, debugging, and programming interfaces using a pinout (Header) or test pads on the PCB board. Later this interface was adopted as IEEE 1149.1 standard. This standard defines the test access port (TAP) controller logic used in the processor with the JTAG interface. JTAG interface requires five pins. In many systems, the optional TRST pin is not implemented so TRST is optional. resulting in its four-wire interface the jtag pins are,

- TMS – Test Mode Select
- TCK – Test Clock
- TDI – Test Data In
- TDO – Test Data Out
- TRST – Test Reset (optional pin)

Our upcoming blogs will cover the JTAG interface in more detail.

Identify JTAG Port on DIVA:

We will scan breakout 1 or J3 header for possible JTAG interface, as of now we don't know which pin is what for the JTAG port.

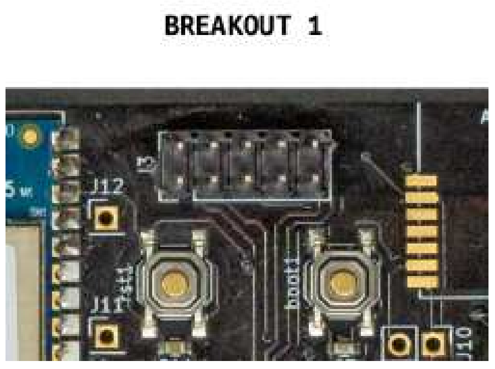


Fig 4. Breakout 1

1) Identify GND (Ground pin). The DIVA board should be powered off when checking continuity. Identify any metallic sheet area (usually metallic sheet areas are connected to ground on the board) or identify the GND pin of the input DC supply of the DIVA board.

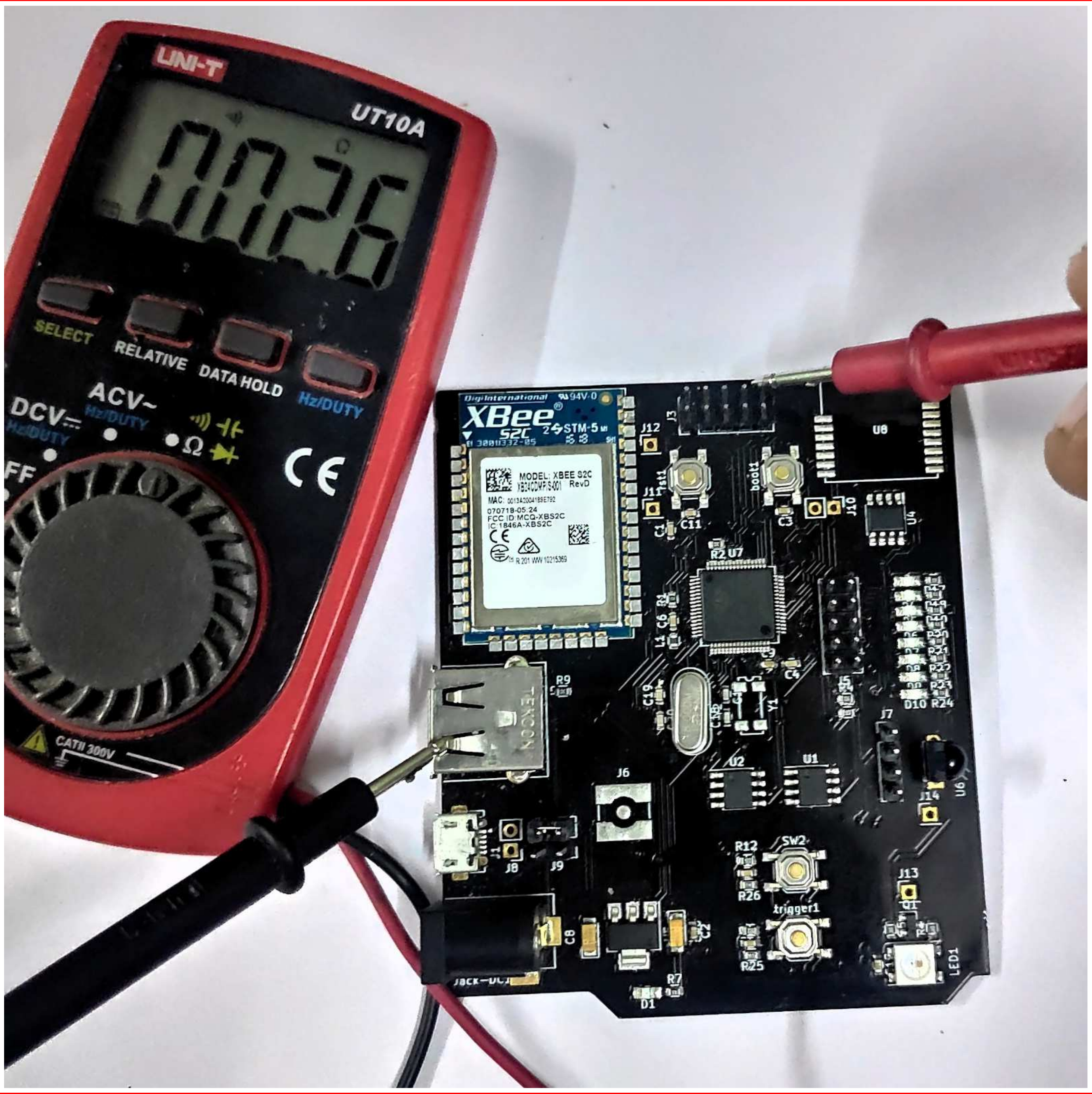


Fig 5. Identify ground pin

2) If the Multimeter beeps, it means that the pinout is connected to the ground i.e. it is a GND pin. Now let's connect Bus Auditor GND pin to diva GND pin and CH0-CH8 pins to header Breakout 1 (Bus Auditor pins are labeled from 0 to 15)

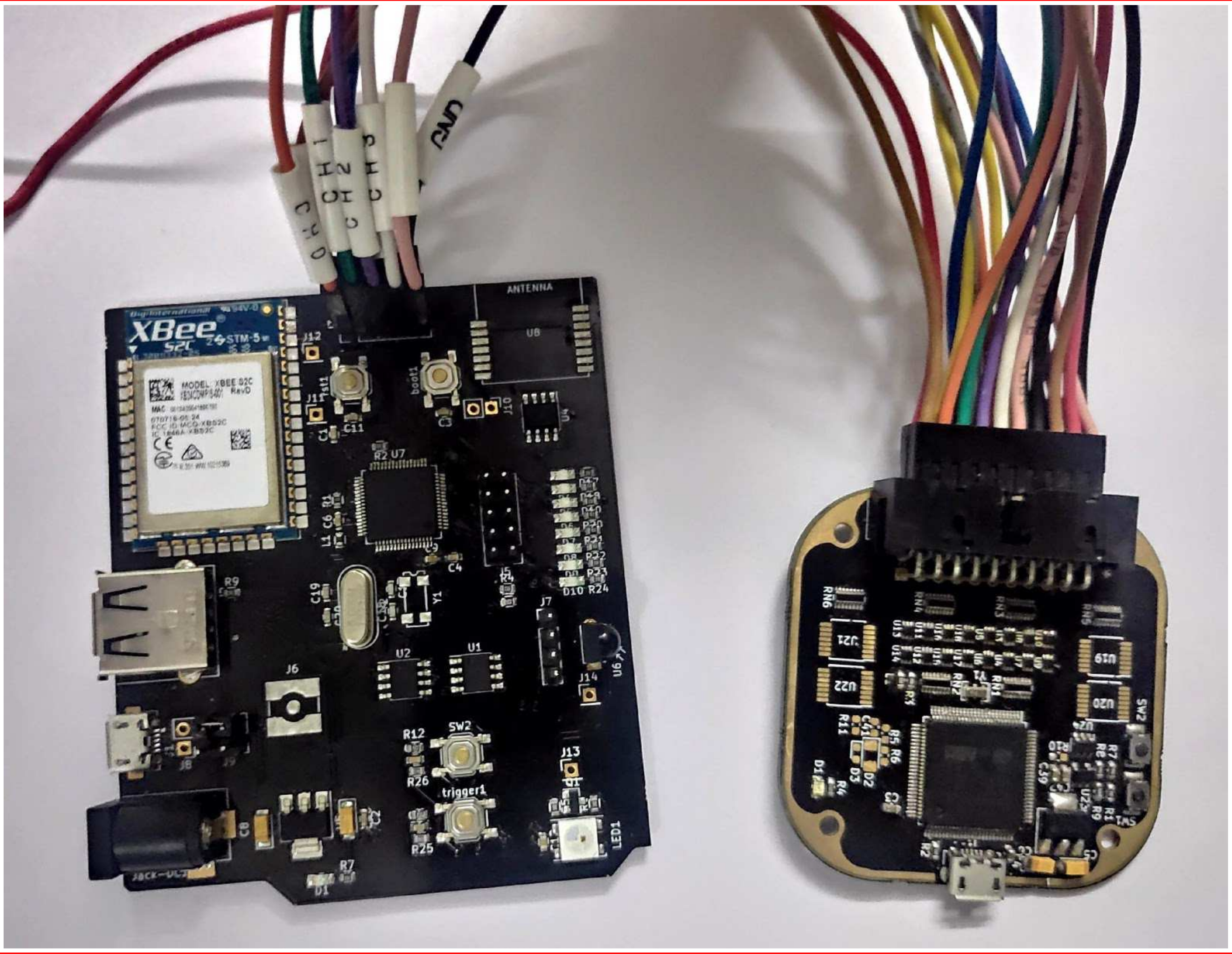


Fig 7. Pin Connections

3) Connect both devices to pc using USB to Micro USB cable.

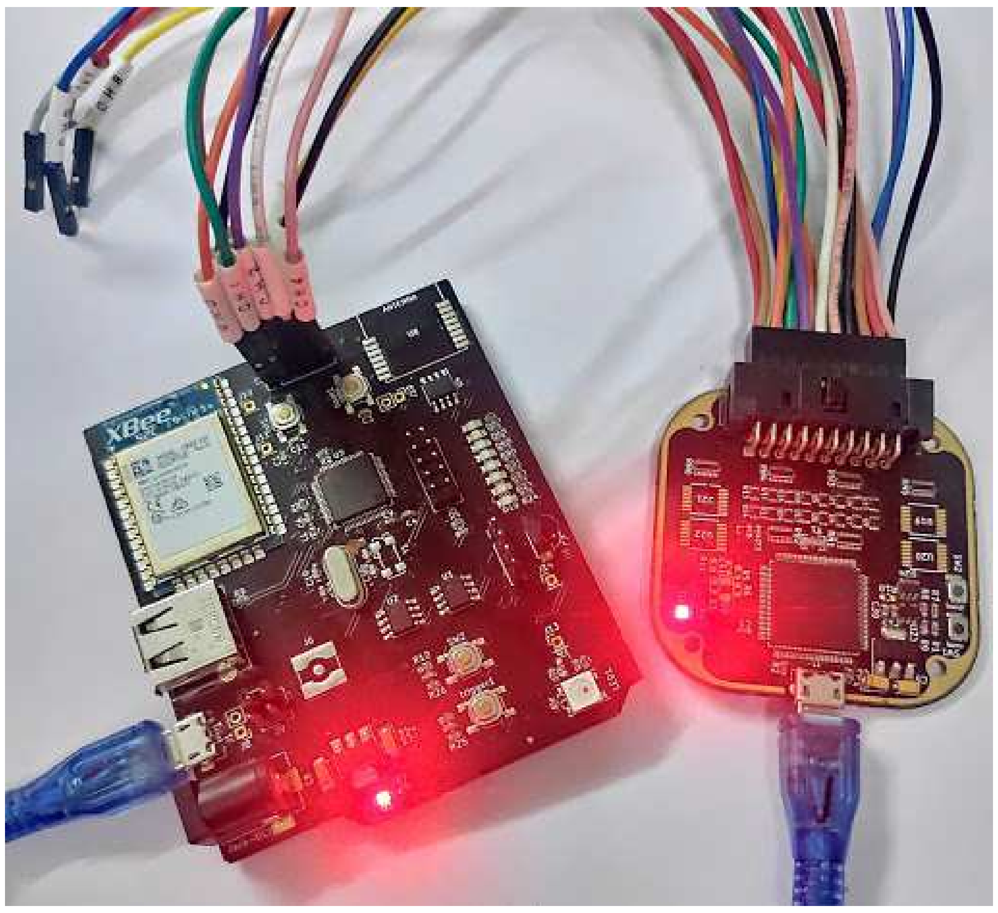


Fig 9. Boards Power up

4) Run the EXPLIoT framework (use root/sudo if your system does not allow user processes to connect to tty* devices).

- $ expliot

5) Run the Busauditor plugin

- ef> run busauditor.generic.jtagscan -v 3.3 -p /dev/ttyACM0 -s 0 -e 10

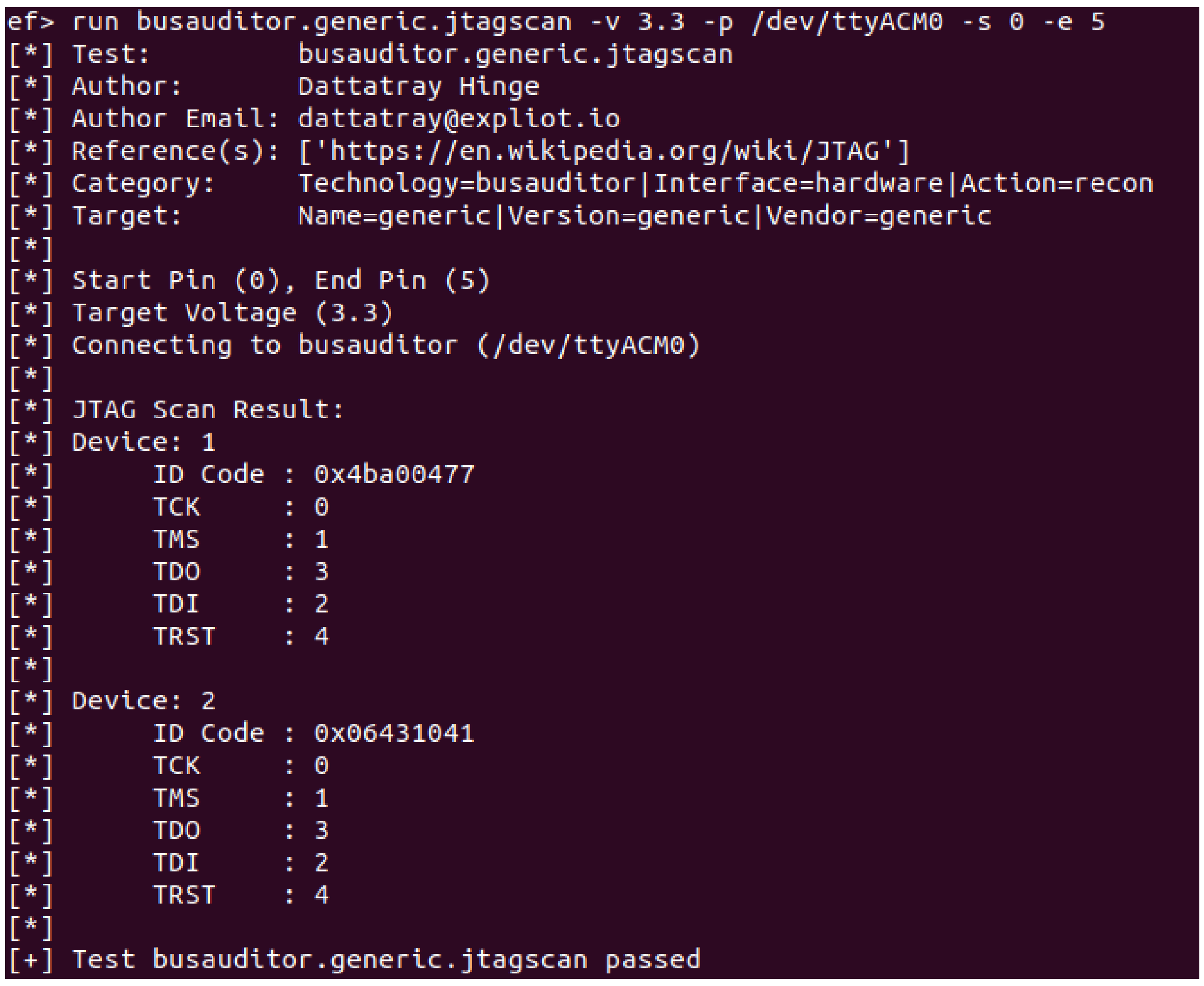
-v is for setting Voltage.

-p dev/tty* port.

-s starting channel.

-e ending channel.

wait for the plugin to complete JTAG port scanning.



```
ef> run busauditor.generic.jtagscan -v 3.3 -p /dev/ttyACM0 -s 0 -e 5
[*] Test:         busauditor.generic.jtagscan
[*] Author:       Dattatray Hinge
[*] Author Email: dattatray@expliot.io
[*] Reference(s): ['https://en.wikipedia.org/wiki/JTAG']
[*] Category:     Technology=busauditor|Interface=hardware|Action=recon
[*] Target:       Name=generic|Version=generic|Vendor=generic
[*]
[*] Start Pin (0), End Pin (5)
[*] Target Voltage (3.3)
[*] Connecting to busauditor (/dev/ttyACM0)
[*]
[*] JTAG Scan Result:
[*] Device: 1
[*]      ID Code : 0x4ba00477
[*]      TCK     : 0
[*]      TMS     : 1
[*]      TDO     : 3
[*]      TDI     : 2
[*]      TRST    : 4
[*]
[*] Device: 2
[*]      ID Code : 0x06431041
[*]      TCK     : 0
[*]      TMS     : 1
[*]      TDO     : 3
[*]      TDI     : 2
[*]      TRST    : 4
[*]
[+] Test busauditor.generic.jtagscan passed
```

Fig 9. JTAG Scan

Bus Auditor Result :

and here we found JTAG pins and ID code

TCK: CH0

TMS: CH1

TDO: CH3

TDI: CH2

TRST: CH4

and ID CODEs are

ID Code : 0x4ba00477

ID Code : 0x06431041

ID CODE is a 32bit number unique manufacturer ID that identifies the part type in the JTAG chain. The ID code differs from vendor to vendor. here we found 2 ID codes one is from the main controller from STM electronics and one is the jtag controller in the SOC.

SWD (Serial Wire Debugger):

The JTAG connector requires four pins for many applications A variant of the JTAG was introduced called serial wire debug (SWD) it uses just 2 pins with clock and bi-directional data pin SWDIO and SWCLK are overlaid on the TMS and TCK pins.

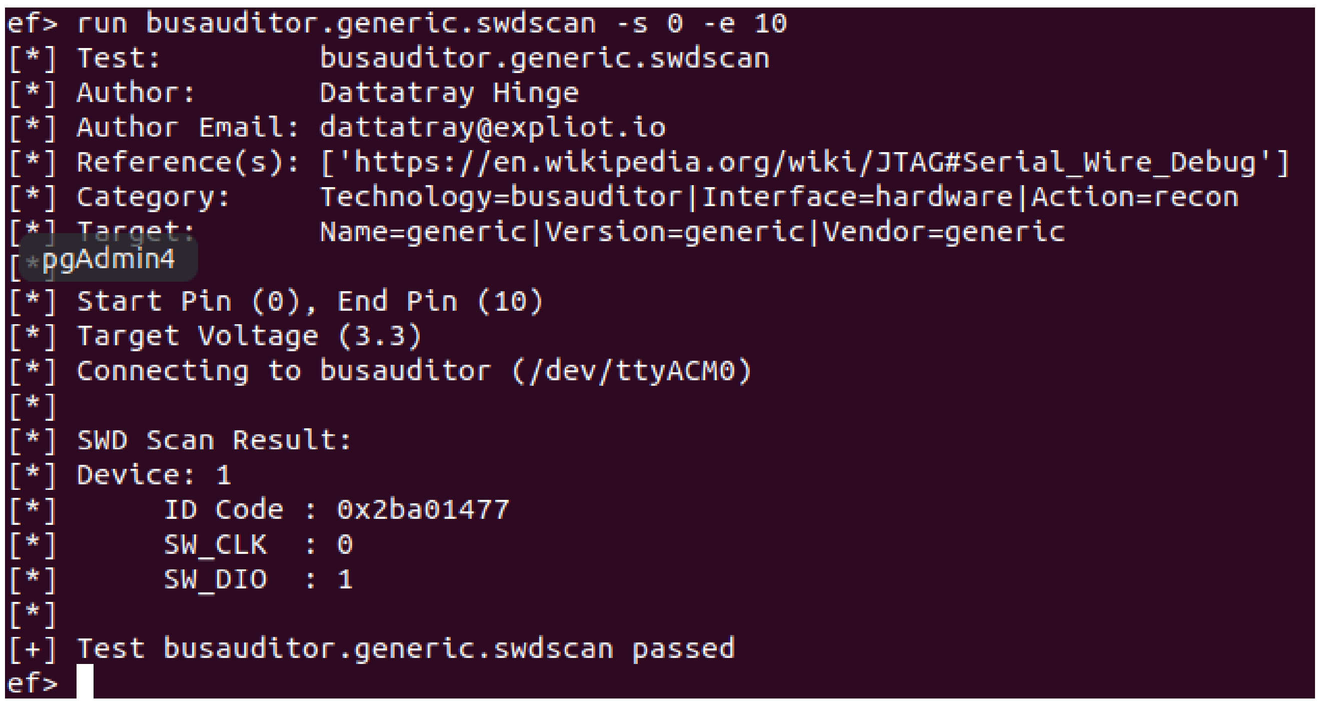
Identify SWD Port on DIVA:

We will keep the connection same for identifying SWD pins,

Run command:

- ef> run busauditor.generic.swdscan -s 0 -e 10

(Note: voltage and device port it will take default 3.3v and ttyACM0)



```
ef> run busauditor.generic.swdscan -s 0 -e 10
[*] Test:         busauditor.generic.swdscan
[*] Author:       Dattatray Hinge
[*] Author Email: dattatray@expliot.io
[*] Reference(s): ['https://en.wikipedia.org/wiki/JTAG#Serial_Wire_Debug']
[*] Category:     Technology=busauditor|Interface=hardware|Action=recon
[*] Target:       Name=generic|Version=generic|Vendor=generic
[*pgAdmin4
[*] Start Pin (0), End Pin (10)
[*] Target Voltage (3.3)
[*] Connecting to busauditor (/dev/ttyACM0)
[*]
[*] SWD Scan Result:
[*] Device: 1
[*]      ID Code : 0x2ba01477
[*]      SW_CLK  : 0
[*]      SW_DIO  : 1
[*]
[+] Test busauditor.generic.swdscan passed
ef>
```

Fig 10. SWD Scan

Bus Auditor Result :

And here we found Pins

SW_CLK: CHO

SW_DIO: CH1

and ID Code : 0x2ba01477

We found SWD pins and ID code. Using this information, attackers can extract firmware, reverse engineer the logic, and flash malicious firmware on the device. JTAG/SWD adds significant capability when debugging microcontrollers However, they require access to the JTAG/SWD pins on the processor. Debugging tools like GDB and OpenOCD (Open On-Chip Debugger) can be used over JTAG/SWD to debug/manipulate the code/execution.

I2C(Inter-Integrated Circuit):

I2C is an asynchronous, multi-master, multi slave, serial communication bus which means that multiple chips can be connected to the same bus and each one can act as a master by initiating data transfer. I2C is used to communicate between low-speed peripherals ICs and processor/microcontroller within a short distance on the same board. Like UART communication, I2C only uses two wires to transmit and receive data between devices.

- SDA (Serial Data) – The line for the master and slave to send and receive data.
- SCL (Serial Clock) – The line that carries the clock signal.

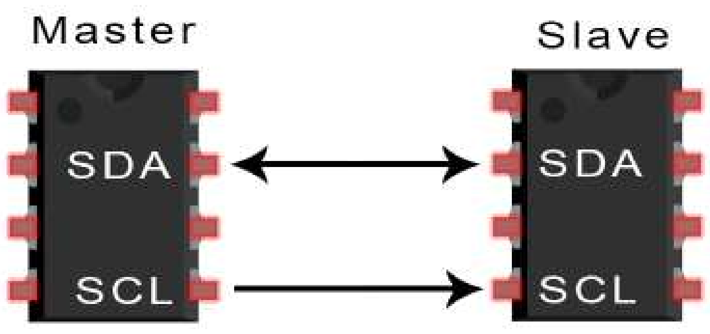


Fig 11. I2C-Communication

(Image source: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/)

Identify i2c on the diva board

We will connect the bus auditor to diva pin header breakout 2

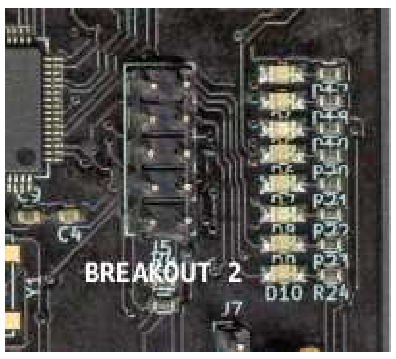


Fig 12. Breakout 2

1) Identify the ground pin as the same we found earlier using a multimeter continuity test. Connect GND and other CHO-CH8 channels to the pin header breakout board see in the below figure connection made

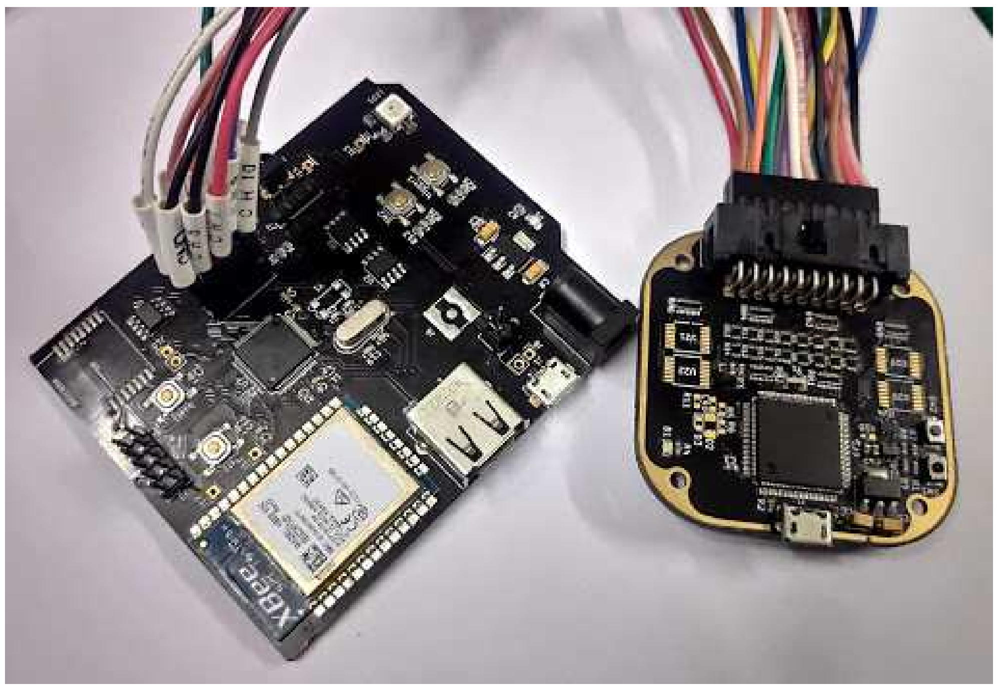


Fig 13. Pin Connections

2) Connect both devices to pc using USB to Micro USB cable.

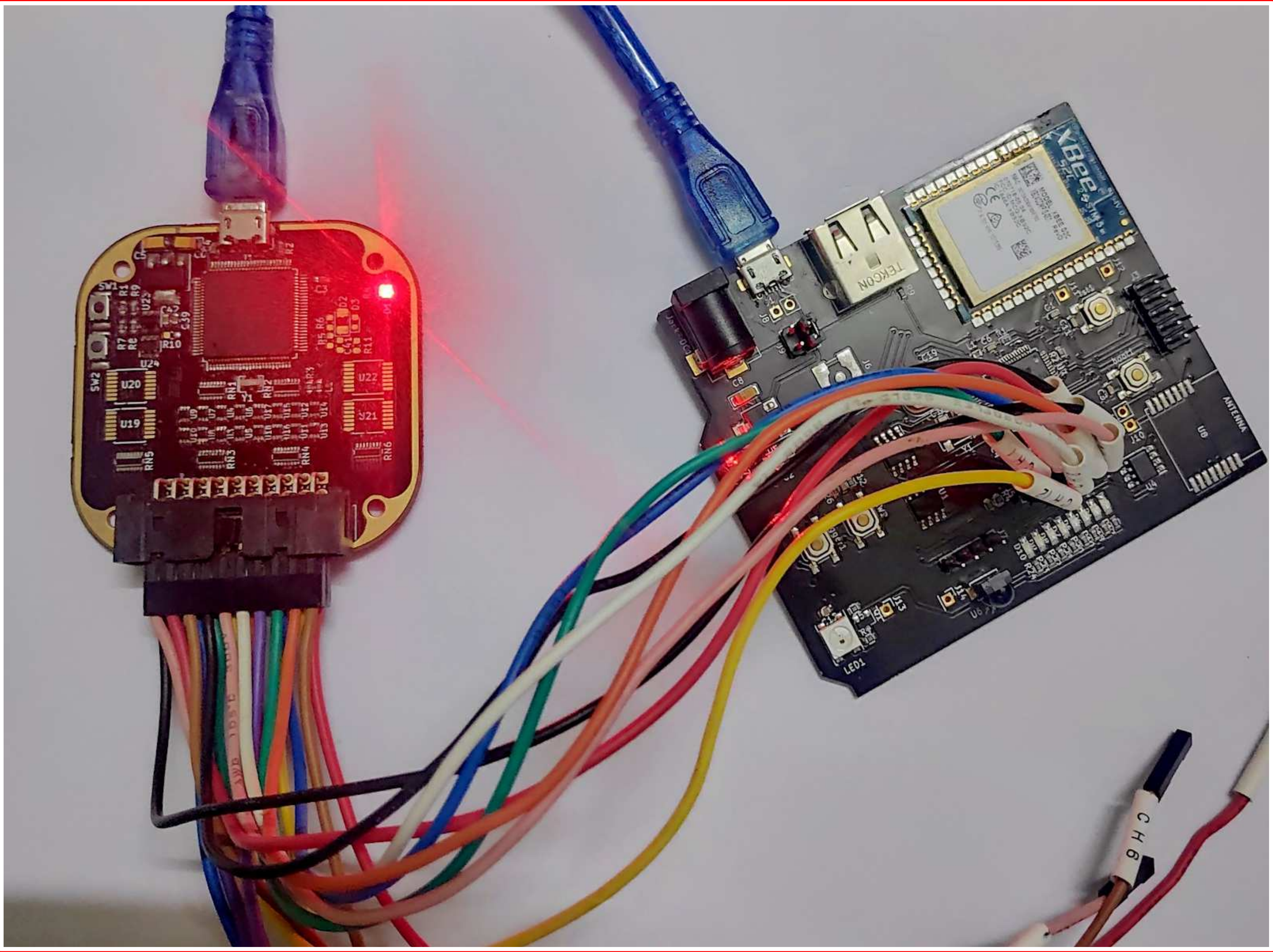
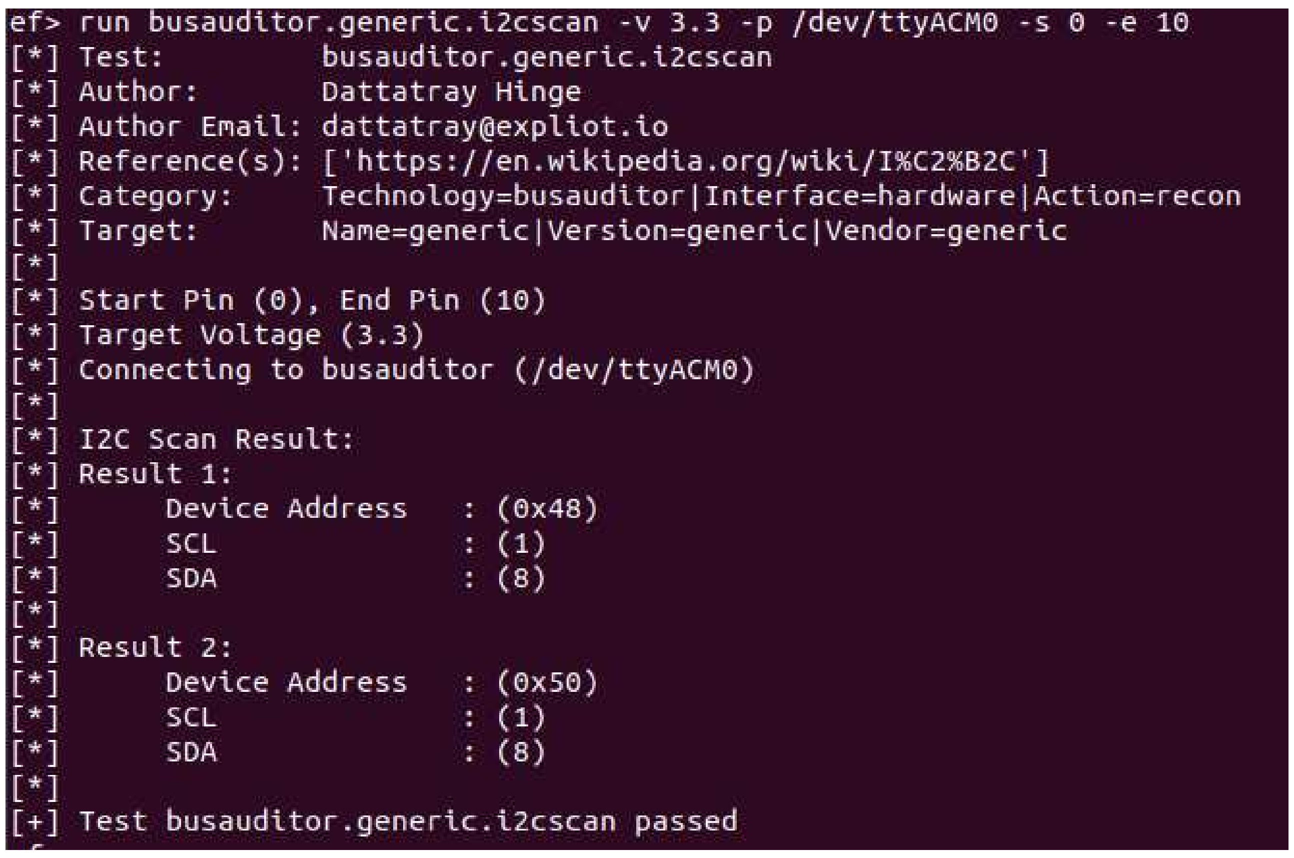


Fig 14. Boards Power up

3) Open EXPLIoT framework Run command:

- ef> run busauditor.generic.i2cscan-v 3.3 -p /dev/ttyACM0 -s 0 -e 10



```
ef> run busauditor.generic.i2cscan -v 3.3 -p /dev/ttyACM0 -s 0 -e 10
[*] Test:         busauditor.generic.i2cscan
[*] Author:       Dattatray Hinge
[*] Author Email: dattatray@expliot.io
[*] Reference(s): ['https://en.wikipedia.org/wiki/I%C2%B2C']
[*] Category:     Technology=busauditor|Interface=hardware|Action=recon
[*] Target:       Name=generic|Version=generic|Vendor=generic
[*]
[*] Start Pin (0), End Pin (10)
[*] Target Voltage (3.3)
[*] Connecting to busauditor (/dev/ttyACM0)
[*]
[*] I2C Scan Result:
[*] Result 1:
[*]      Device Address   : (0x48)
[*]      SCL              : (1)
[*]      SDA              : (8)
[*]
[*] Result 2:
[*]      Device Address   : (0x50)
[*]      SCL              : (1)
[*]      SDA              : (8)
[*]
[+] Test busauditor.generic.i2cscan passed
```

Fig 15. I2C Scan

Bus Auditor Result :

We can see that the bus auditor identified 2 pins out of 8 pins and found the device address.

SCL: CH1

SDA: CH8

Device Address : (0x48)

Device Address : (0x50)

One of the use cases of I2C is in EEPROM chips that are connected to the microcontroller I2C pins and typically store data or code. Typical attacks would include tampering with the data, extracting sensitive information, corrupting the data, etc. We should analyze the data at rest on the EEPROM chip as well as perform run-time analysis by sniffing the I2C communication to understand the behavior and security implications.

UART (Universal asynchronous receiver-transmitter):

UART (Universal Asynchronous Receiver Transmitter) is a hardware component that allows asynchronous serial communication between two hardware components. This can be on the same devices (microcontrollers connected to any sensor on the device) or two different devices (device microcontroller talking to a PC via an external UART hardware)

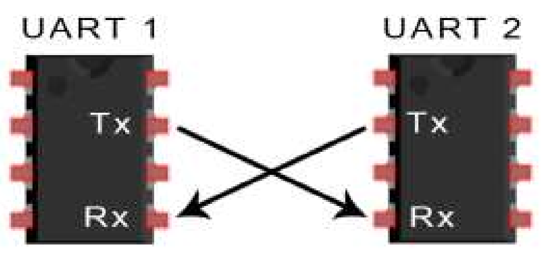


Fig 16. UART Communication

Image source : (https://www.circuitbasics.com/basics-uart-communication/)

Identify UART on DIVA:

Here we will identify UART pinouts on the PCB board when we do not know the microcontroller UART pins and the datasheet of the microcontroller.

1) We will connect bus auditor channel pins to pin header breakout 3 which is on the diva.

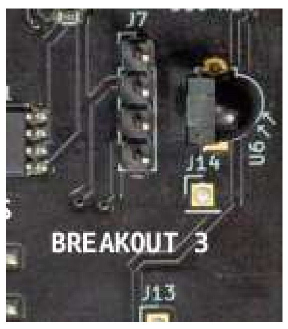


Fig 17. Breakout 3

2) First, we will identify the ground pin using a multimeter as explained earlier. one pin is VCC no need to connect it and for identifying TX, RX we will connect CH0 and CH1 of Bus Auditor.

3) Connect both devices to PC using USB to Micro USB cable.

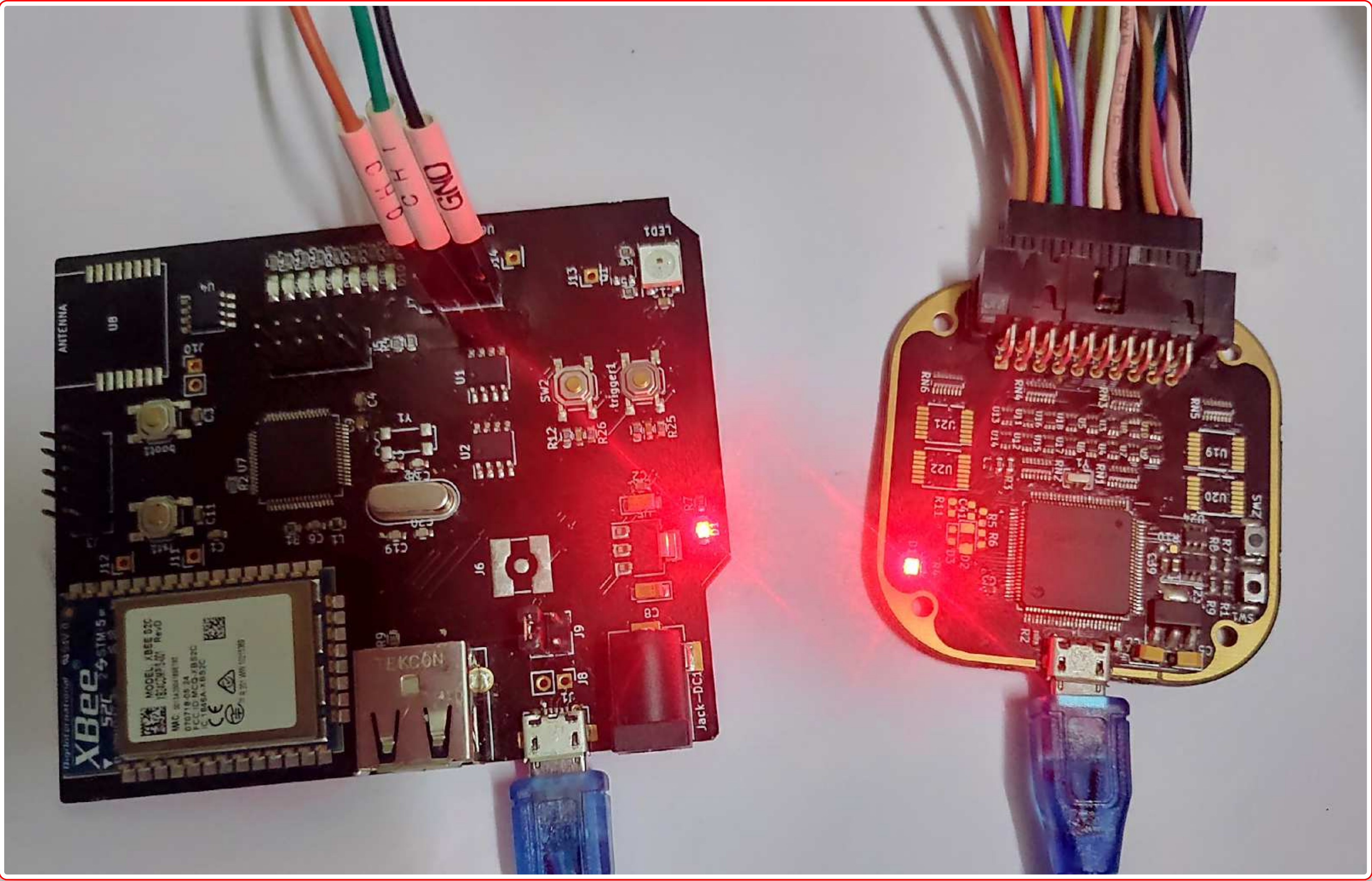
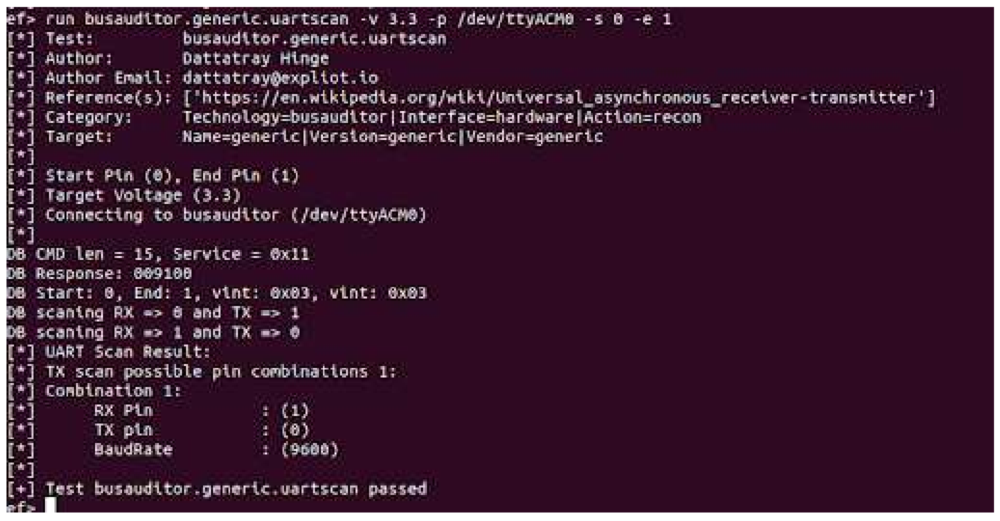


Fig 18. Pin Connections

4) open EXPLIoT framework Run command:

- ef> run busauditor.generic.uartscan -v 3.3 -p /dev/ttyACM0 -s 0 -e 1



```
ef> run busauditor.generic.uartscan -v 3.3 -p /dev/ttyACM0 -s 0 -e 1
[*] Test:         busauditor.generic.uartscan
[*] Author:       Dattatray Hinge
[*] Author Email: dattatray@expliot.io
[*] Reference(s): ['https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter']
[*] Category:     Technology=busauditor|Interface=hardware|Action=recon
[*] Target:       Name=generic|Version=generic|Vendor=generic
[*]
[*] Start Pin (0), End Pin (1)
[*] Target Voltage (3.3)
[*] Connecting to busauditor (/dev/ttyACM0)
[*]
DB CMD len = 15, Service = 0x11
DB Response: 009100
DB Start: 0, End: 1, vint: 0x03, vint: 0x03
DB scaning RX => 0 and TX => 1
DB scaning RX => 1 and TX => 0
[*] UART Scan Result:
[*] TX scan possible pin combinations 1:
[*] Combination 1:
[*]      RX Pin           : (1)
[*]      TX pin           : (0)
[*]      BaudRate         : (9600)
[*]
[+] Test busauditor.generic.uartscan passed
ef>
```

Fig 19. UART Scan

Bus auditor Result:

We found the correct baud rate and correct RX, TX pins,

Baud rate is 9600

RX: CH1

TX: CH0

It is an interesting attack surface as it may allow read/write access to the application, running on the device, over serial. In many devices, UART ports on the board are left open which anyone can connect and access over serial to get a console of some sort i.e. simple shell, custom command line consoles, log output, etc. A device will typically have a group of pin-outs connected to the microcontroller UART RX and TX pins, which are used for sending and receiving serial data. Here we identified the UART port using the Bus auditor and in the upcoming blog posts, we will see how to access the UART ports on a device.

Bus Auditor:

Bus Auditor is our very own home-grown debug port scanner. If you would like to purchase Bus Auditor or for that matter any of our other hardware security tools, feel free to check out our online store at EXPLIoT - https://expliot.io/collections/frontpage

Conclusion :

We hope this blog post gives you an overview of the hardware debug ports, how Bus auditor scans and identifies debugging and communication interfaces exposed on the PCB test pads and pinout headers. Which gives you useful information to perform hardware assessment. In later blog posts, we will cover more details of each hardware debug port.

Continue to the next part - IoT Security - Part 15 (101 - Hardware Attack Surface : SPI)