2020 HACKTIVITY
The IT Security Festival in Central and Eastern Europe

Hacking the Hacktivity 2019 badge:

how to brick the device and resurrect it with another soul

Valerio Di Giampietro
Linux enthusiast since 1993
http://va.ler.io
v@ler.io
@valerio
youtube.com/makemehack

What we will talk about …

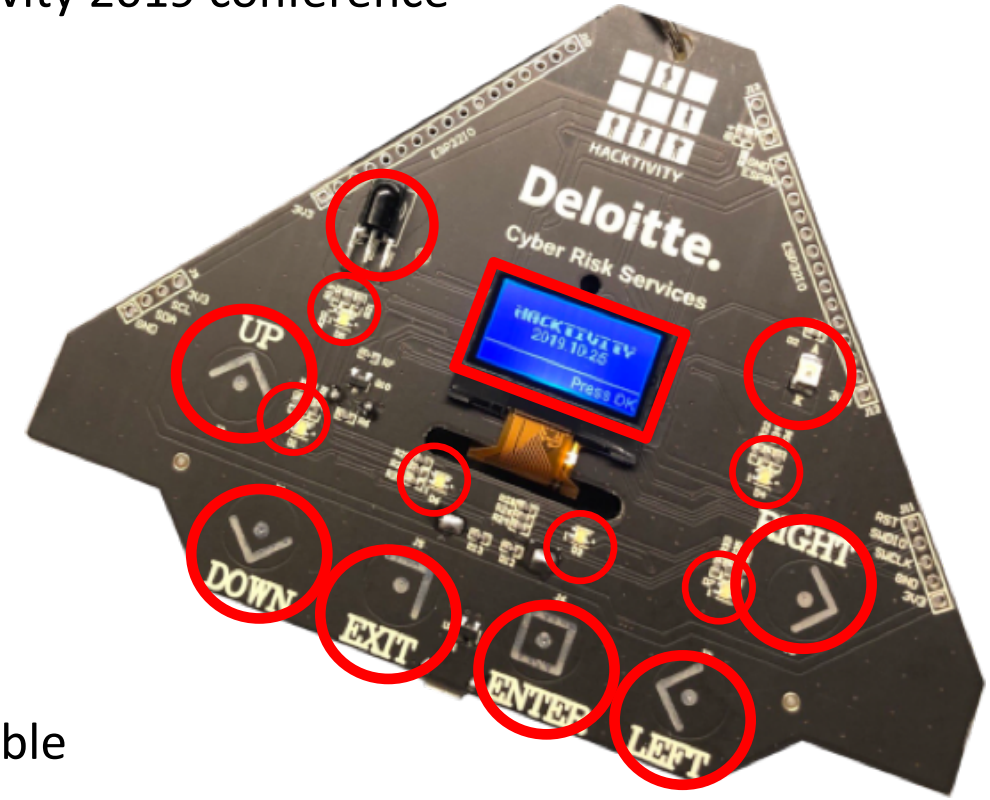
HACKTIVITY
The IT Security Festival in Central and Eastern Europe

- The Hacktivity 2019 Badge
- Bricking the Badge
- Reverse Engineering the PCB
- Arduino IDE on the ESP32
- Re-flashing the SAMD21

The Hacktivity 2019 badge

HACKTIVITY
The IT Security Festival in Central and Eastern Europe

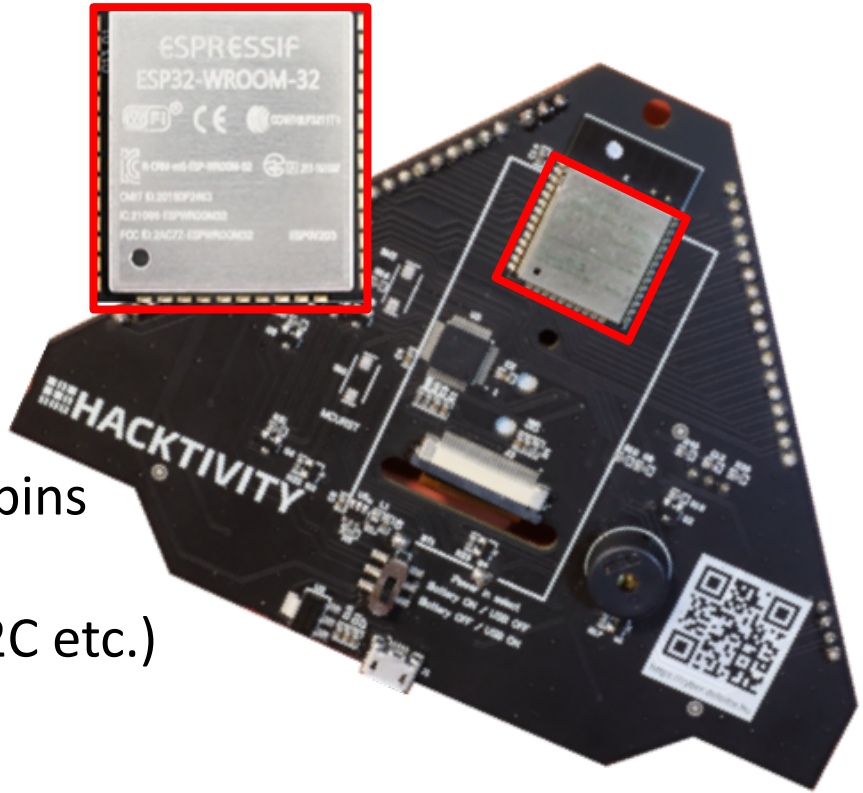
- Distributed last year at Hackitivity 2019 conference
- Features
  - Runs MicroPython
  - Has connectivity over USB, WiFi and IR
  - Has an appstore
  - 6 touch buttons
  - 128×64 LCD screen
  - 6 RGB LEDs
  - IR transmitter and receiver
  - Buzzer
  - Battery or USB powered
  - Designed to be hacked
- No detailed information available on hardware and firmware



The Hacktivity 2019 badge

HACKTIVITY
The IT Security Festival in Central and Eastern Europe

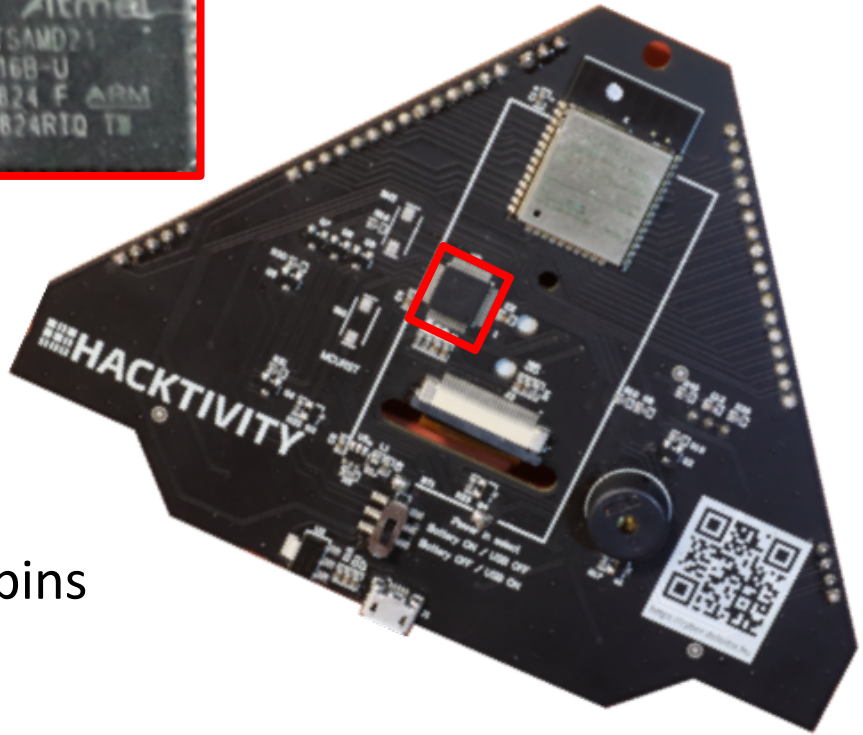
- Pretty “powerful” device
- ESP32-WROOM-32 module
  - 32bit, dual core Xtensa CPU
  - 4 Mb flash memory
  - 520Kb SRAM
  - WiFi/Bluetooth
  - Digital, analog and PWM I/O pins
  - Touch sensors
  - Serial interfaces (UART, SPI, I2C etc.)
  - Ultra low power mode



The Hacktivity 2019 badge

HACKTIVITY
The IT Security Festival in Central and Eastern Europe

- Powerful co-processor
- ATSAMD21G16B
  - ARM Cortex-M0+ CPU
  - 64Kb flash memory
  - 8Kb SRAM
  - SWD interface
  - USB interface
  - Serial interfaces
  - Digital, analog and PWM I/O pins
  - Touch sensors

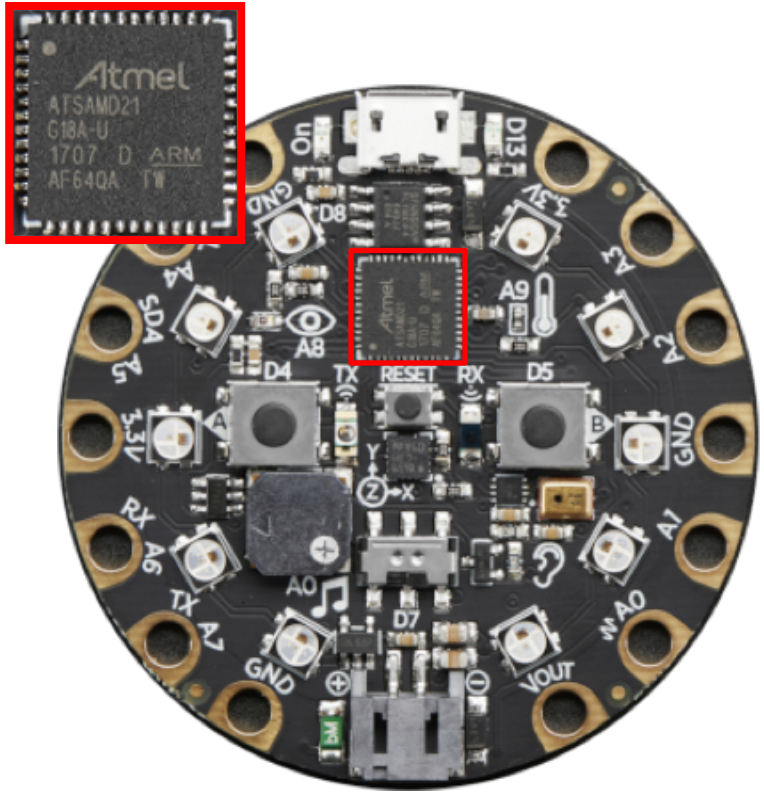


Bricking the badge

- Trying to use the Arduino IDE
- Badge recognized as «Adafruit Playground Express»
- Load the simple «blink» sketch
- Load successful, but the device is bricked!
- The badge is «dead»
- The USB is no more recognized

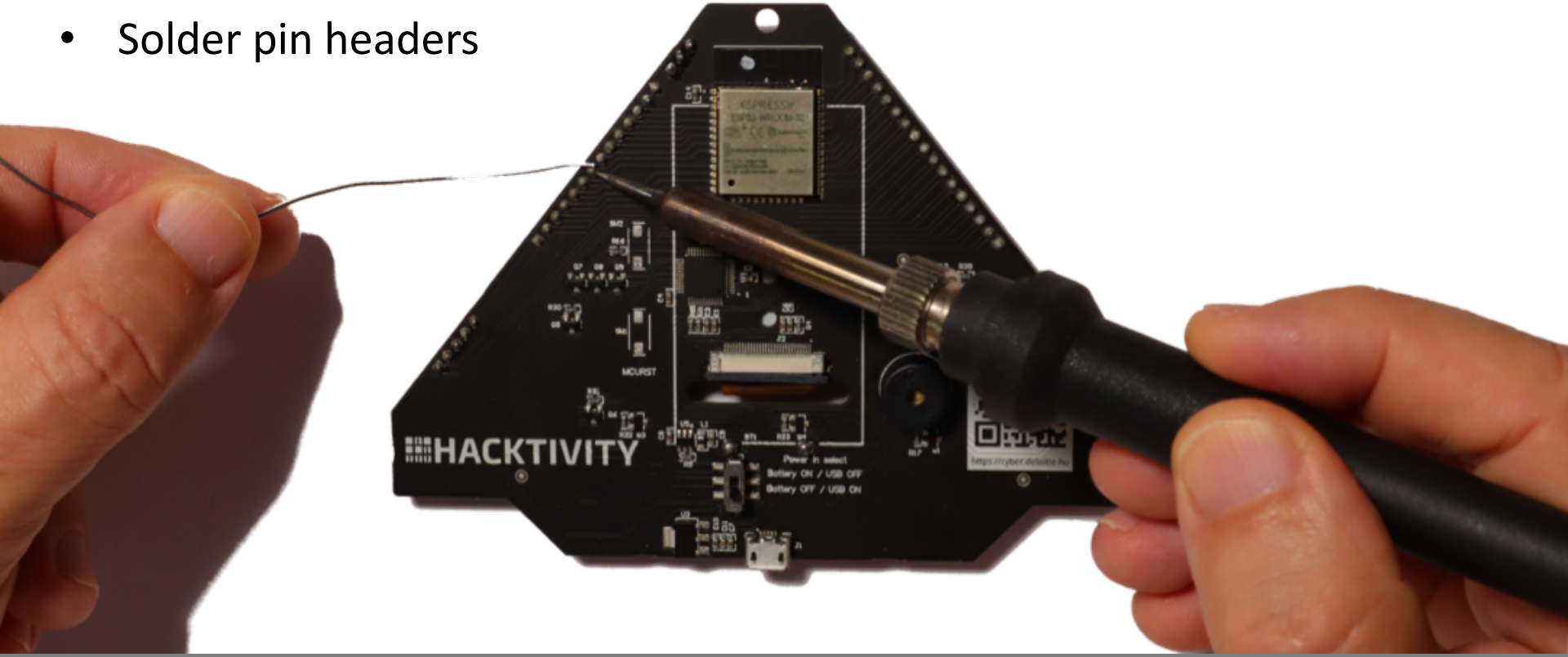
Bricking the badge

- «Adafruit Playground Express»
  - Based on ATSAMD21G18A
  - 256Kb flash memory (4x our badge)
  - 32Kb SRAM (4x our badge)



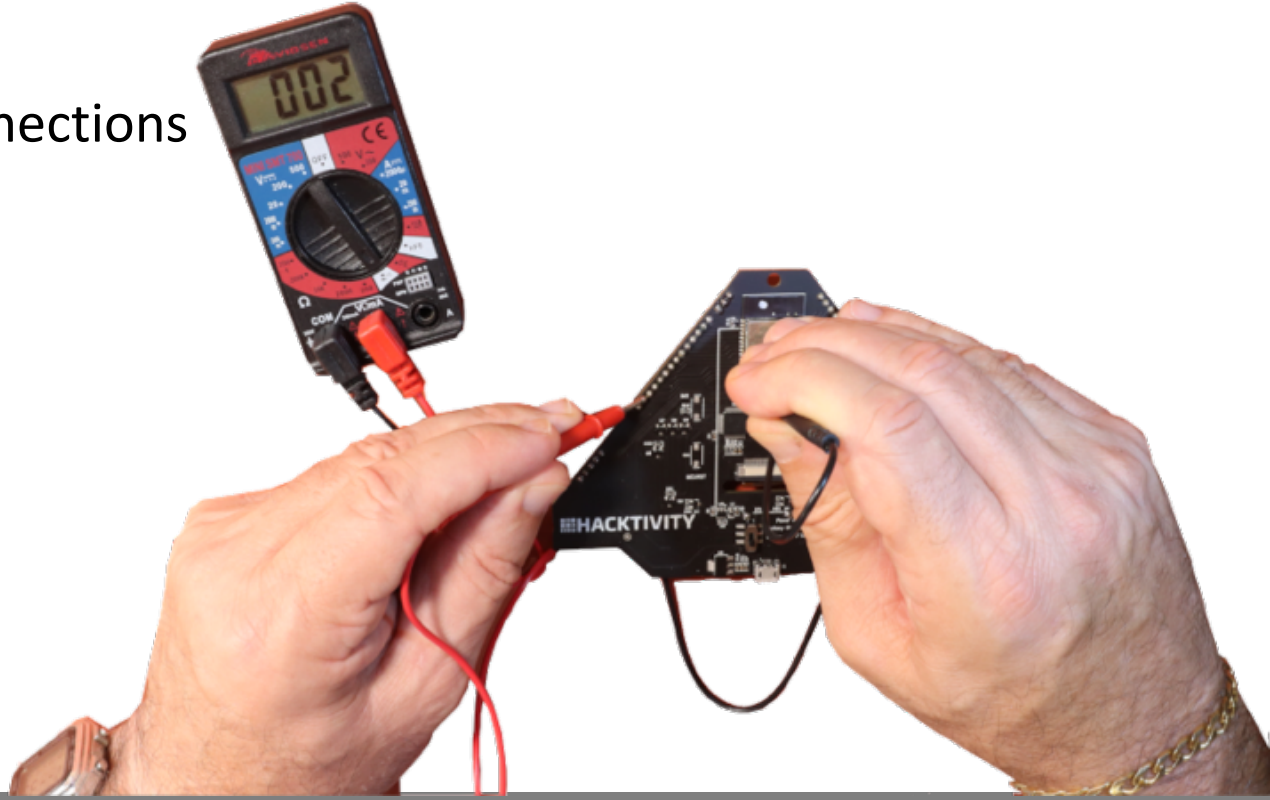
Reversing the PCB

- Solder pin headers



Reversing the PCB

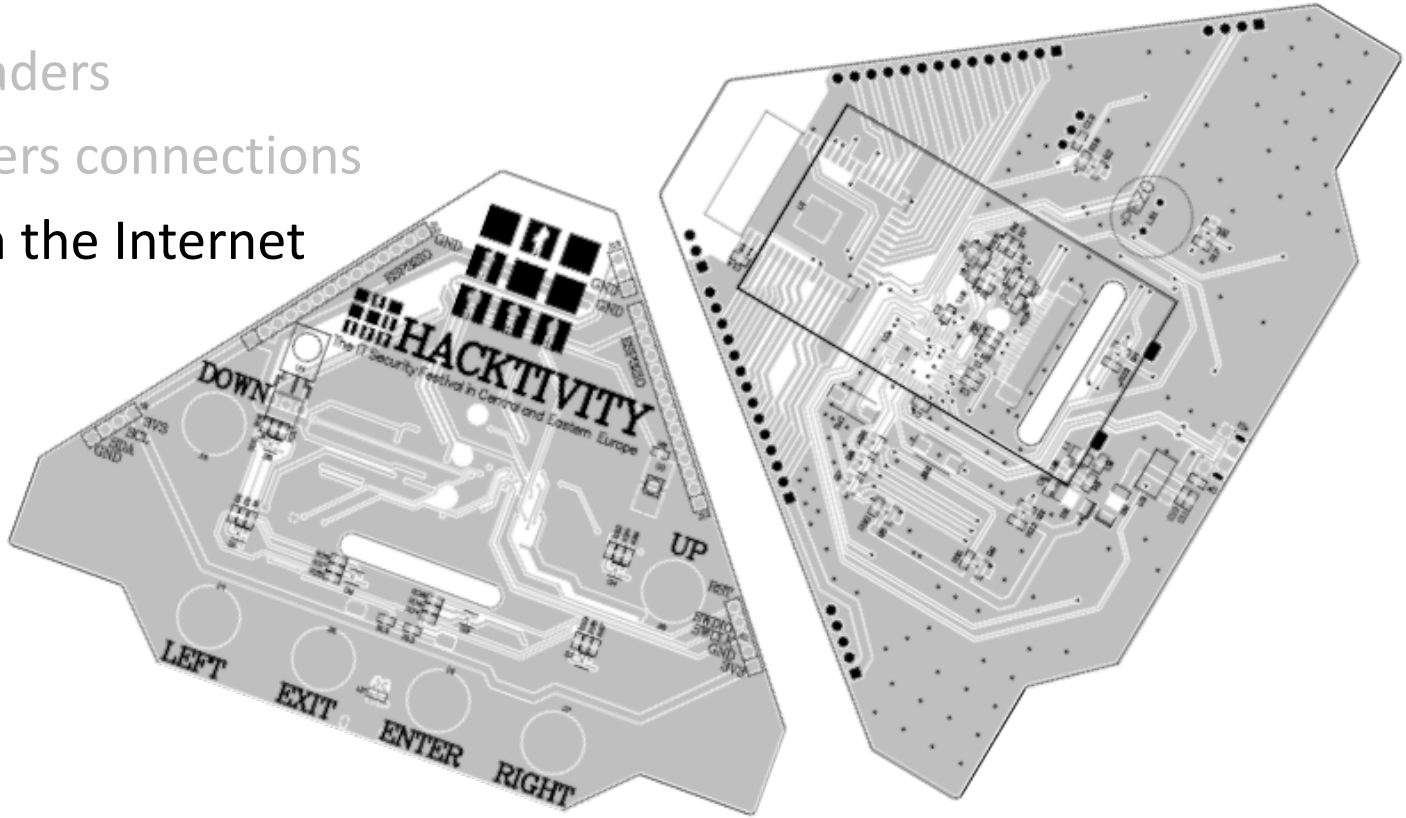
- Solder pin headers
- Find pin headers connections



Reversing the PCB

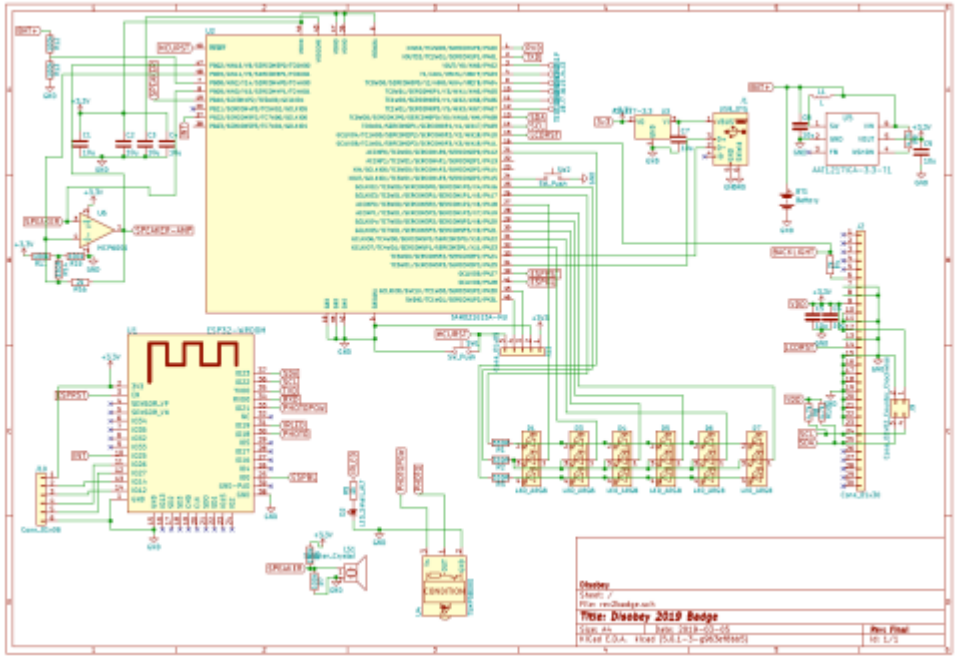
HACKTIVITY
The IT Security Festival in Central and Eastern Europe

- Solder pin headers
- Find pin headers connections
- Search info on the Internet



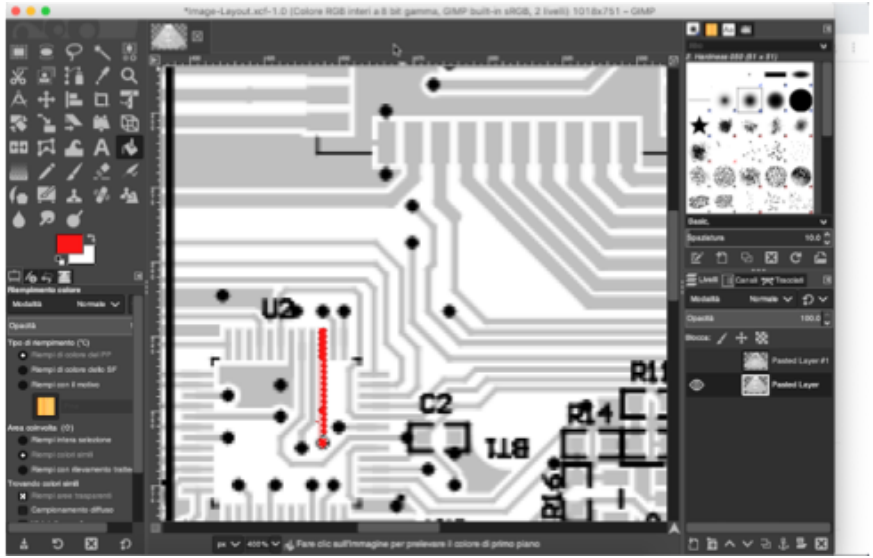
Reversing the PCB

- Solder pin headers
- Find pin headers connections
- Search info on the Internet

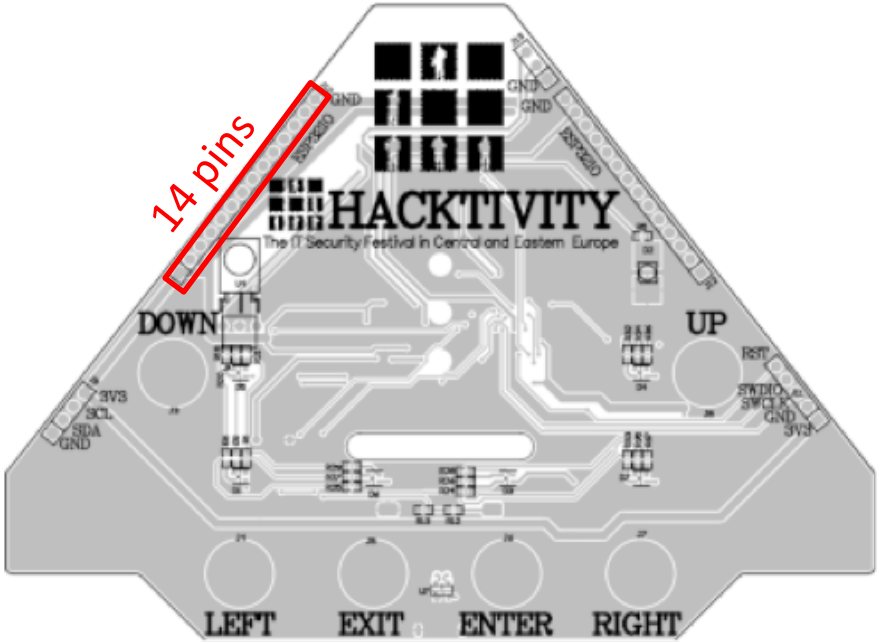


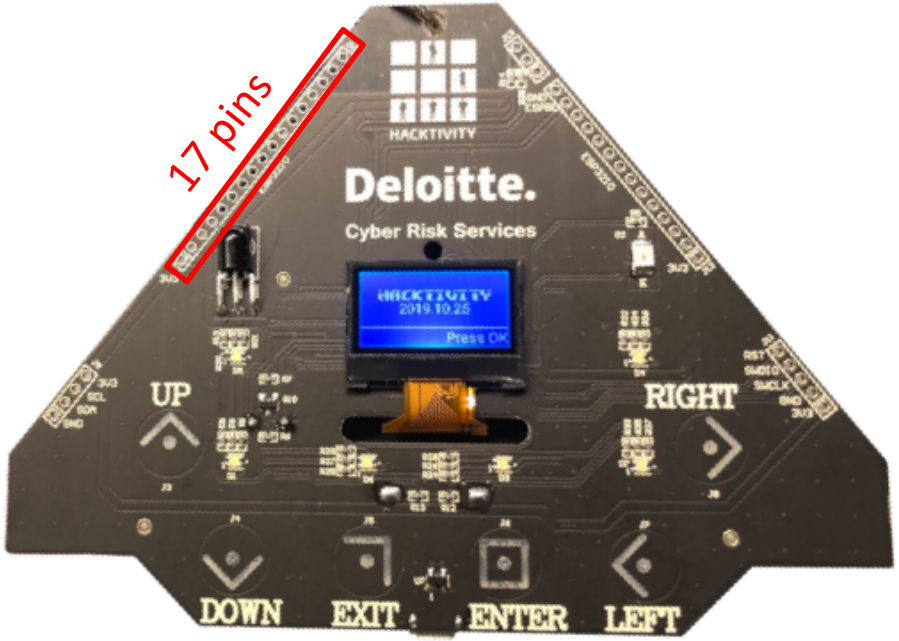
Reversing the PCB

- Solder pin headers
- Find pin headers connections
- Search info on the Internet
- Use Gimp to follow traces on 2 layers PCB

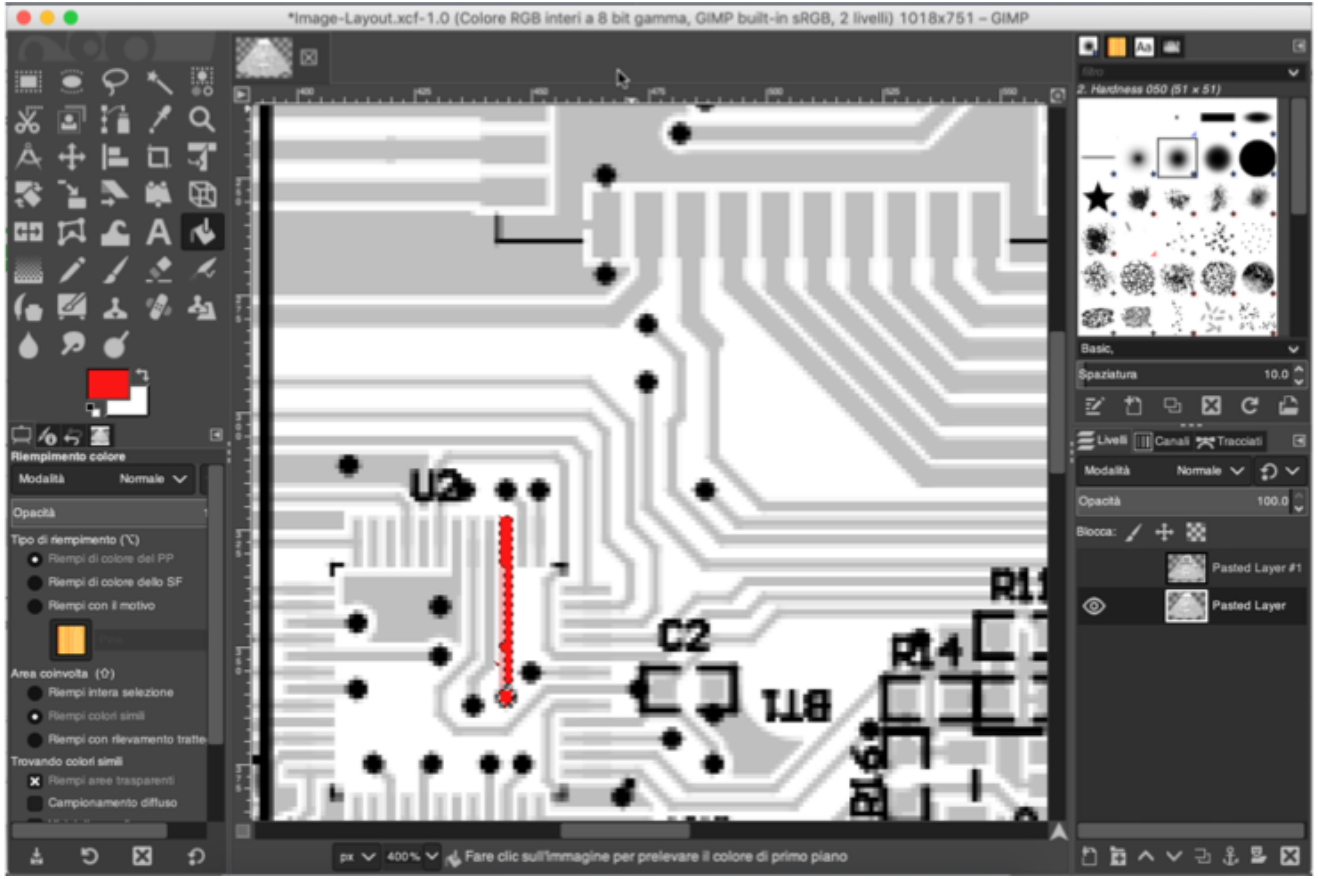


Reversing the PCB

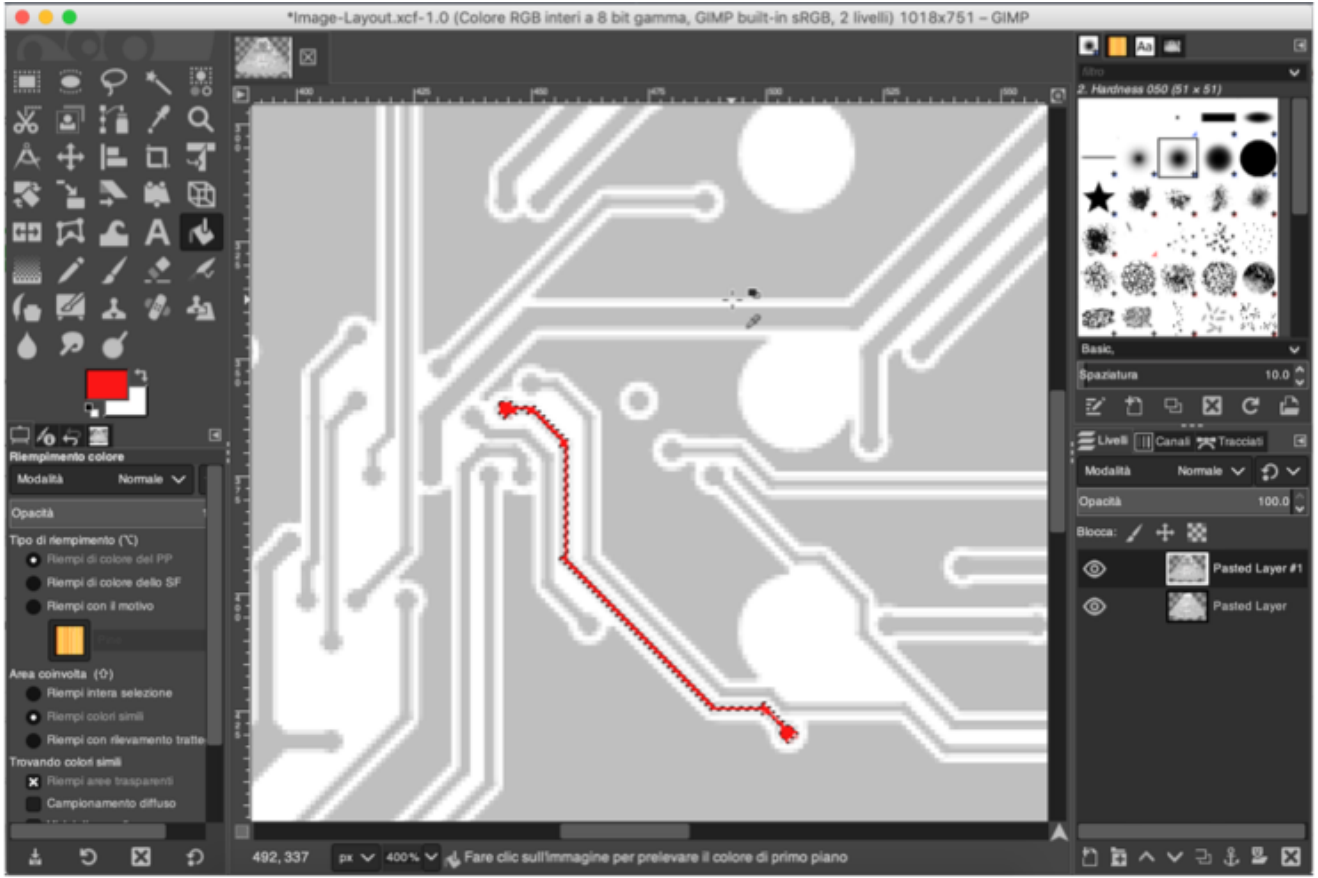




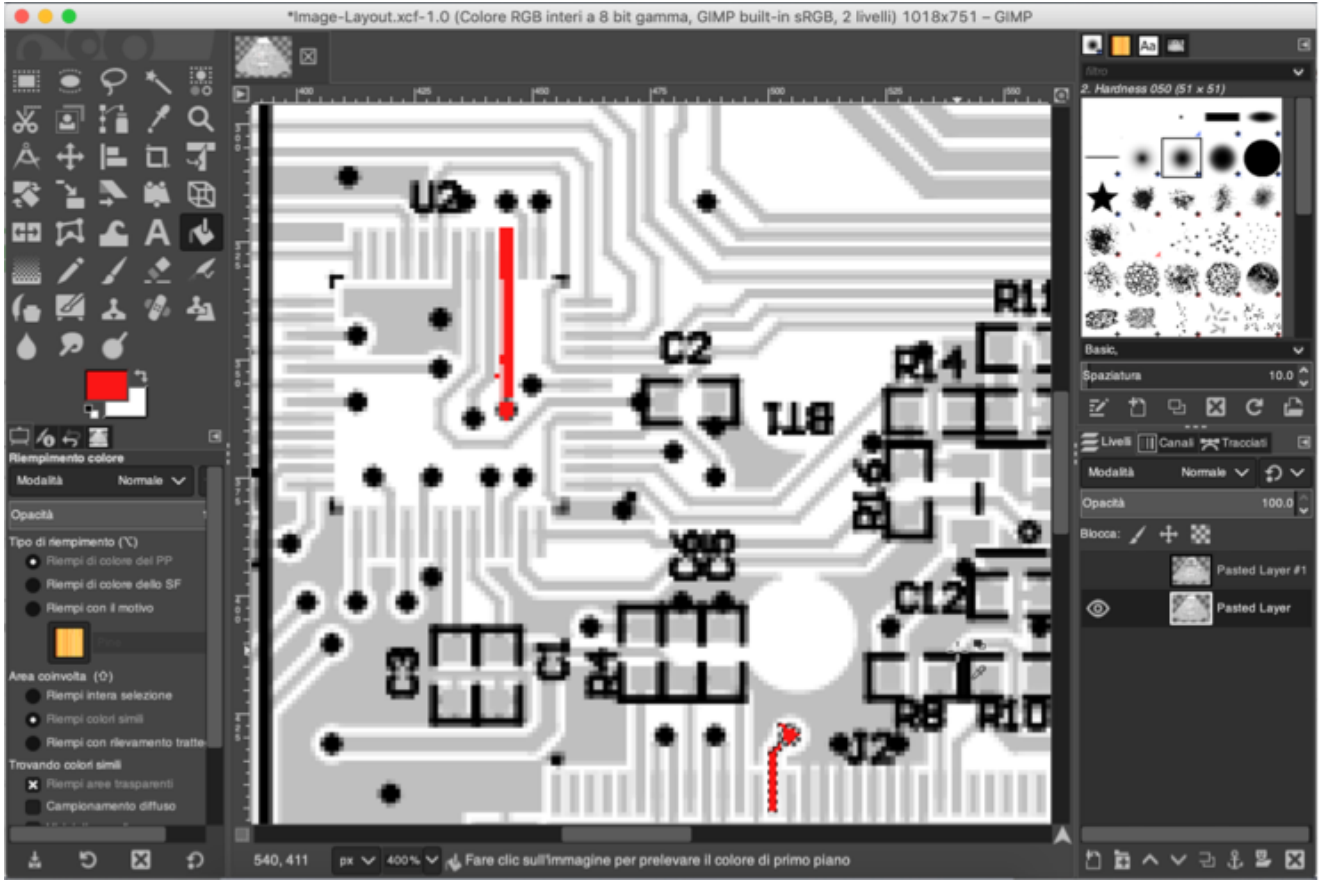
Reversing the PCB



Reversing the PCB

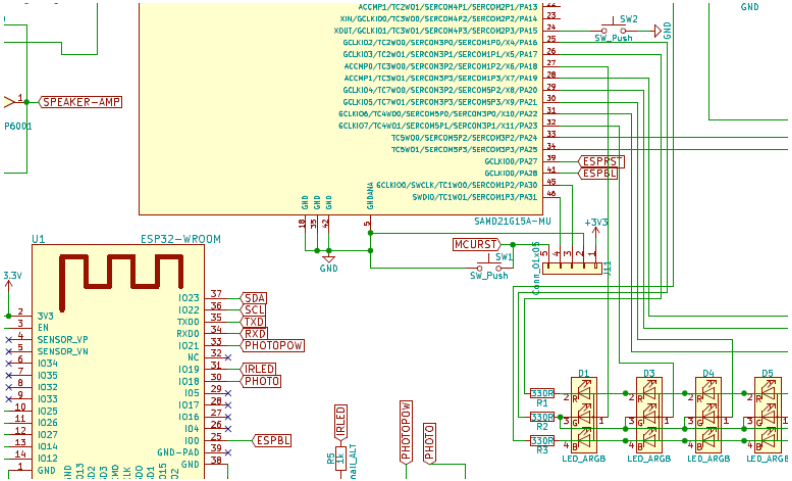


Reversing the PCB



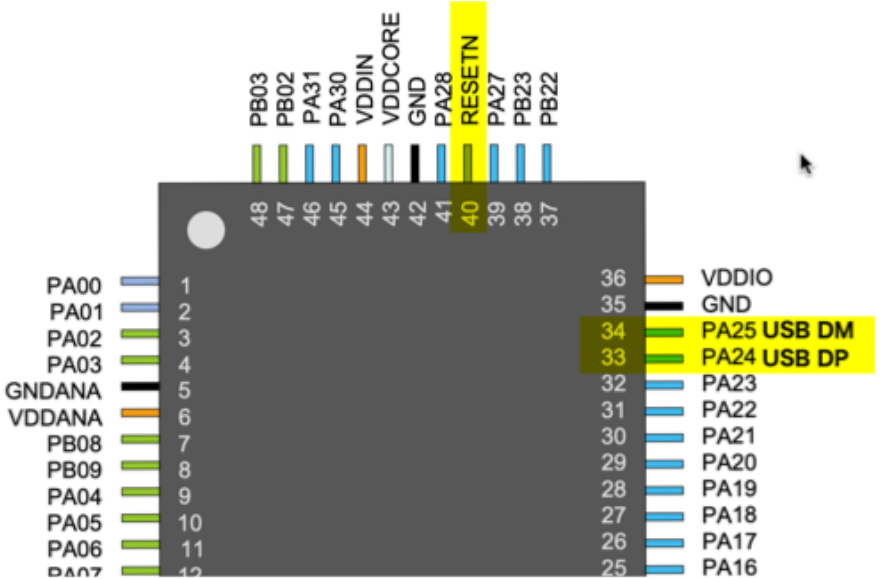
Reversing the PCB

- Solder pin headers
- Find pin headers connections
- Search info on the Internet
- Use Gimp to follow traces on 2 layers PCB
- Use a similar projects as hint



Reversing the PCB

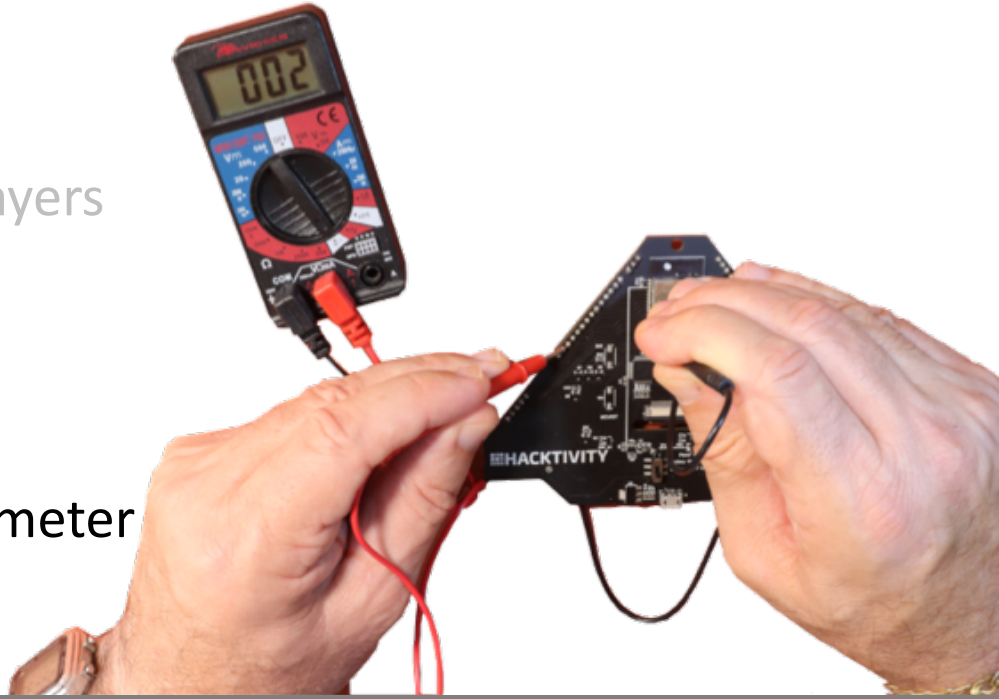
- Solder pin headers
- Find pin headers connections
- Search info on the Internet
- Use Gimp to follow traces on 2 layer PCB
- Use a similar projects as hint
- Use SOCs data sheets as hint



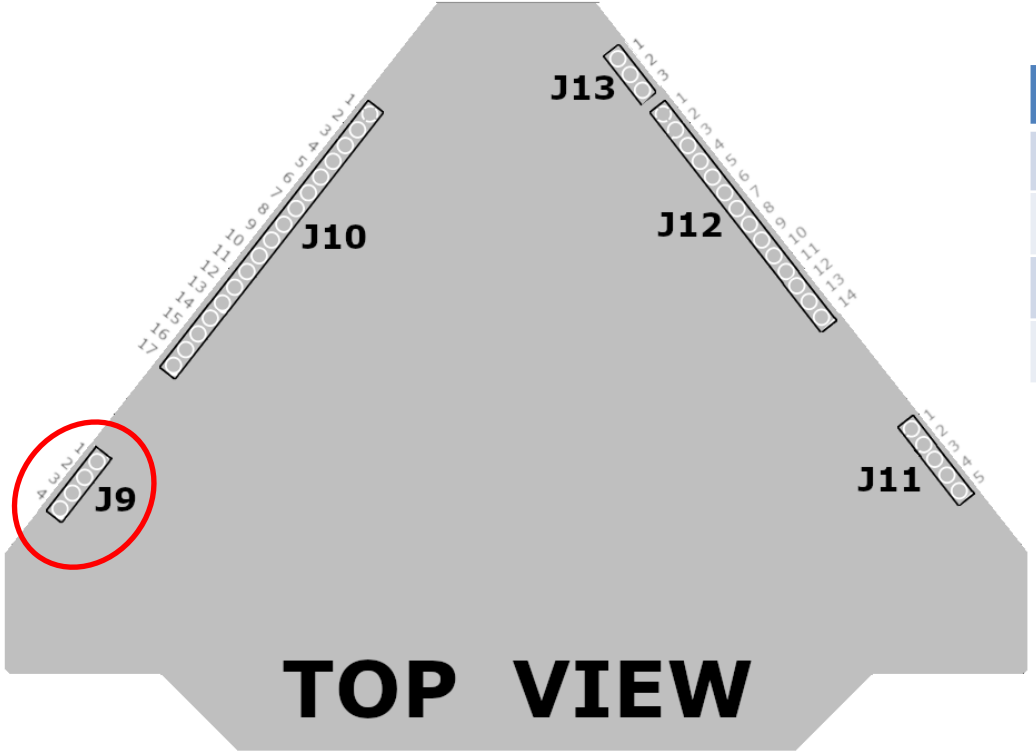
Pin[1]		C	D	E	F	G
SAMD21G	I/O Pin	SERCOM[2][3]	SERCOM-ALT	TC[4]/TCC	TCC	COM
33	PA24	SERCOM3/ PAD[2]	SERCOM5/ PAD[2]	TC5/WO[0]	TCC1/ WO[2]	USB/DM
34	PA25	SERCOM3/ PAD[3]	SERCOM5/ PAD[3]	TC5/WO[1]	TCC1/ WO[3]	USB/DP

Reversing the PCB

- Solder pin headers
- Find pin headers connections
- Search info on the Internet
- Use Gimp to follow traces on 2 layers PCB
- Use a similar projects as hint
- Use SOCs data sheets as hint
- Confirm connection with a multimeter



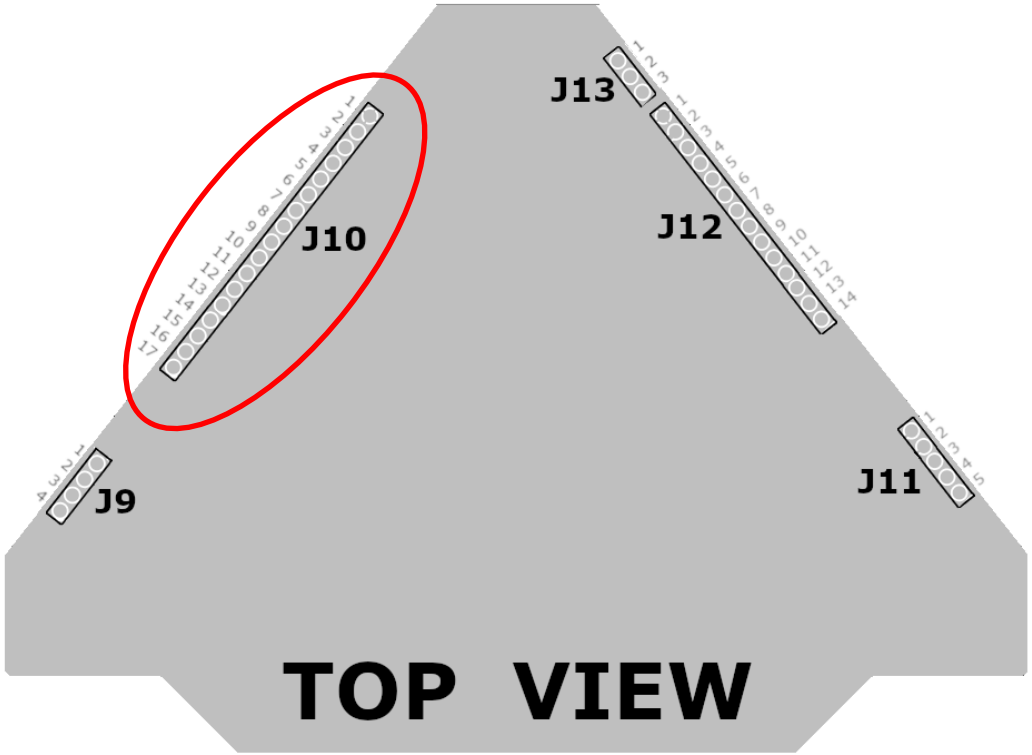
Pin headers – J9



J9 (I2C Bus)

PIN	Name	ESP32	SAMD21
1	VCC		
2	SCL	36 – IO22	14 – PA09
3	SDA	37 – IO23	13 – PA08
4	GND		

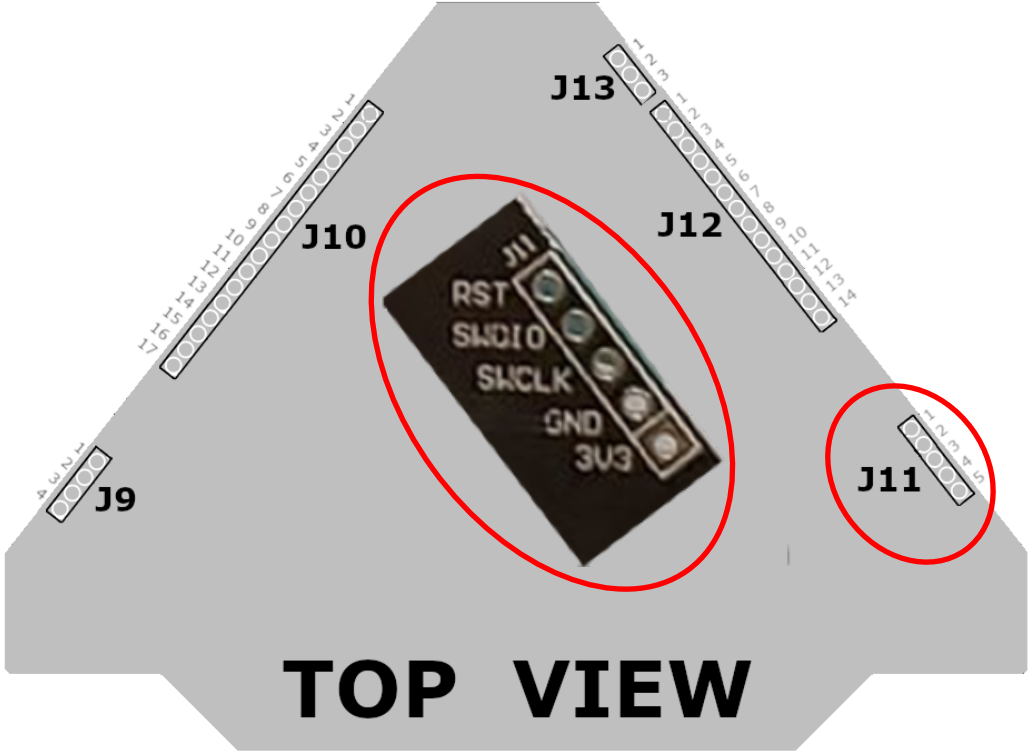
Pin headers – J10



J10 (to ESP32)

PIN	ESP32	PIN	ESP32
1	GND	10	23-IO15
2	33-IO21	11	22-IO8
3	31-IO21	12	21-IO7
4	30-IO18	13	20-IO6
5	29-IO15	14	19-IO11
6	28-IO17	15	18-IO10
7	27-IO16	16	17-IO9
8	26-IO4	17	VCC
9	24-IO2		

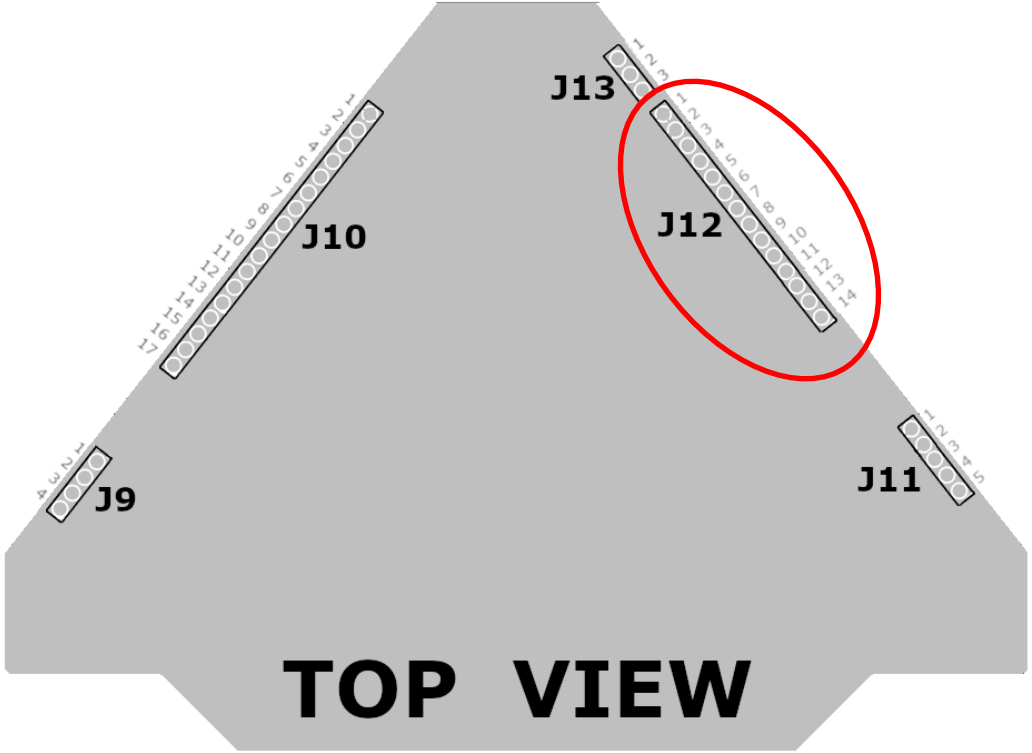
Pin headers – J11



J11 (SAMD21 programming interface)

PIN	SAMD21
1	40 - nRST
2	46 - SWDIO
3	45 - SWCLK
4	GND
5	VCC

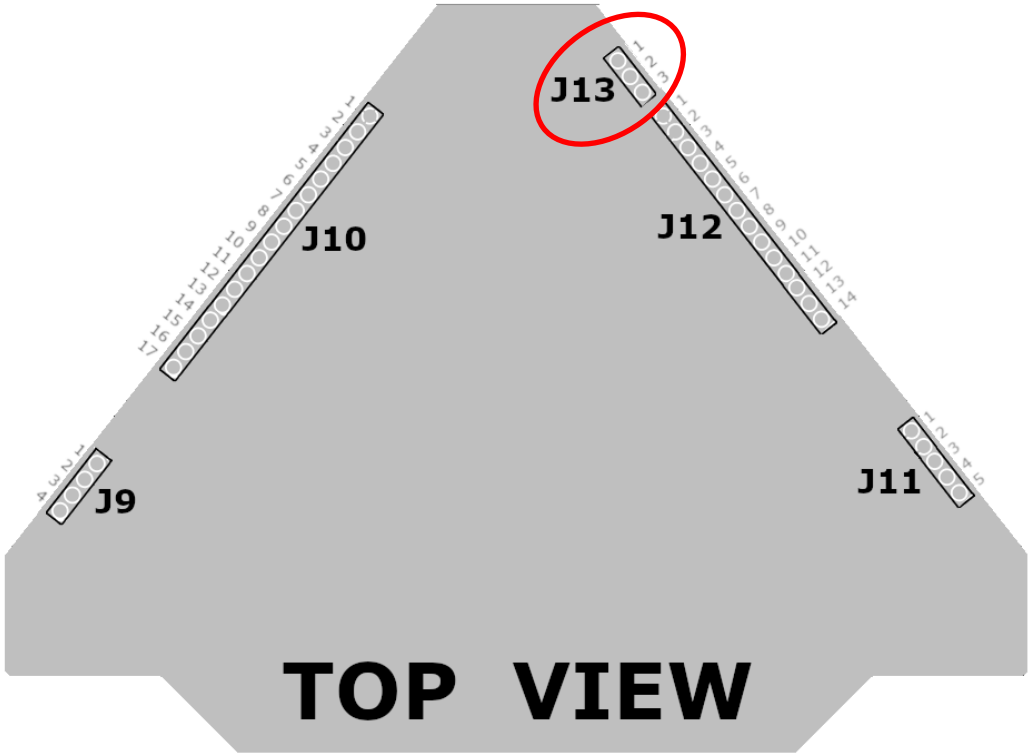
Pin headers – J12



J12 (to ESP32)

PIN	ESP32	PIN	ESP32
1	GND	8	9 – IO33
2	25 – IO0 (B.LDR)	9	11 – IO26
3	4 – IO36	10	12 – IO27
4	5 – IO39	11	13 – IO14
5	6 – IO34	12	14 – IO12
6	7 – IO35	13	16 – IO13
7	8 – IO32	14	VCC

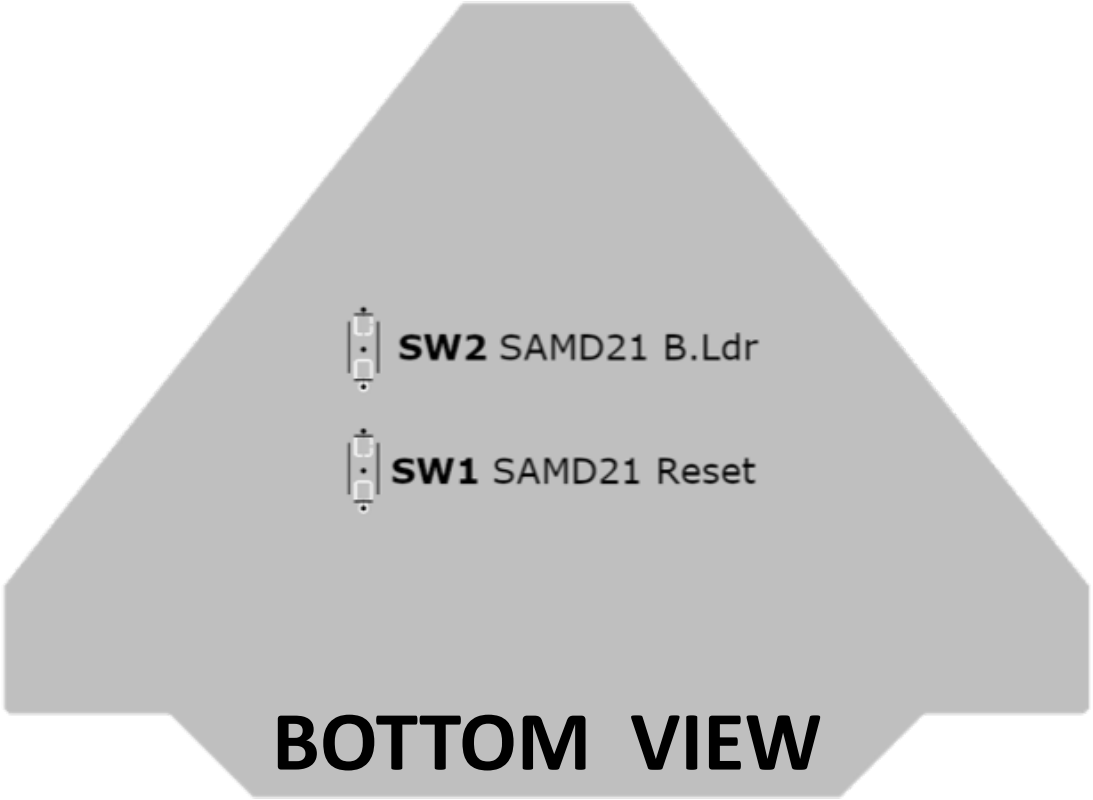
Pin headers – J13



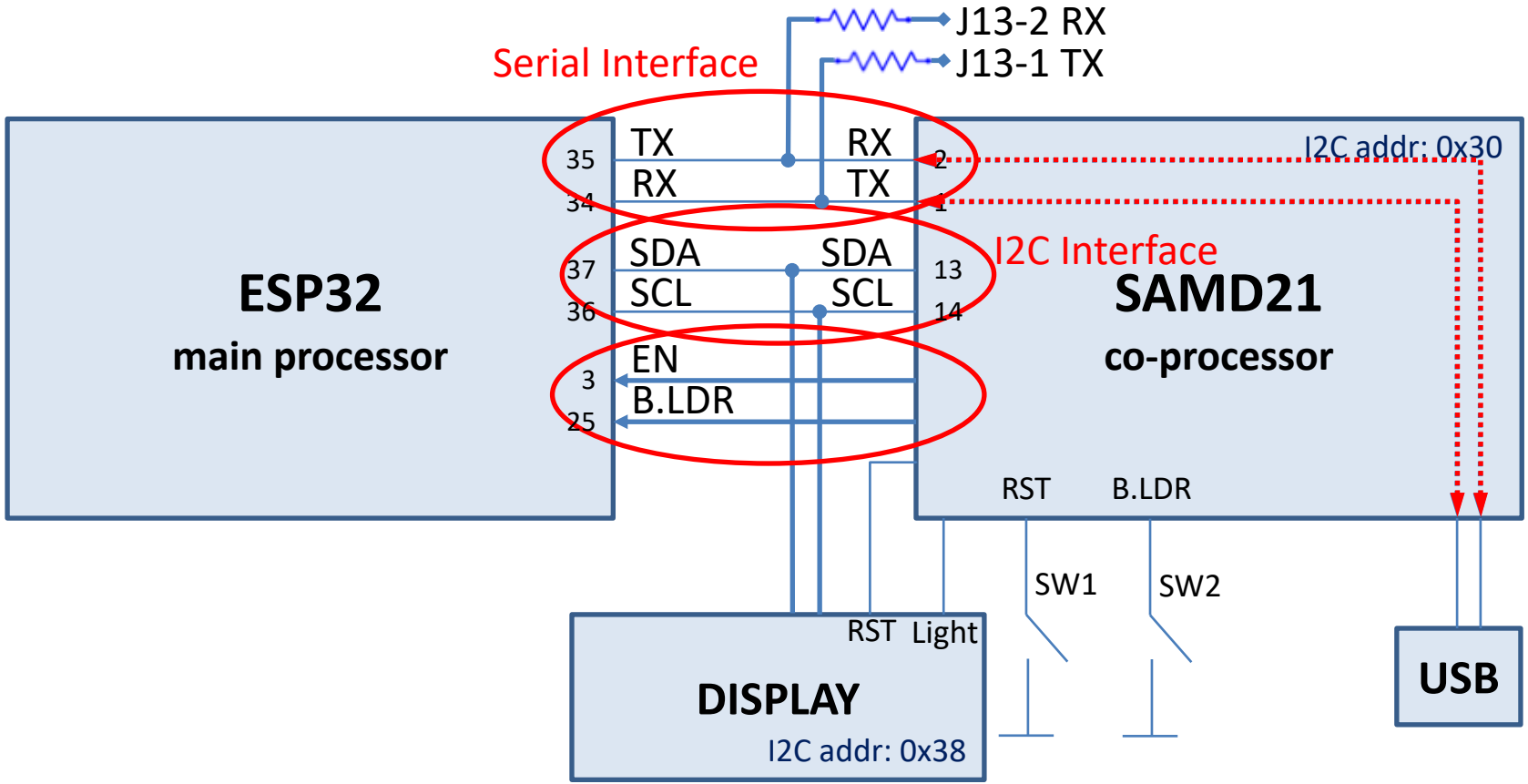
J13 (ESP32 serial interface)

PIN	NAME	ESP32	SAMD21
1	TX	35 (330 Ω)	2 - RX
2	RX	34 (330 Ω)	1 - TX
3	GND		

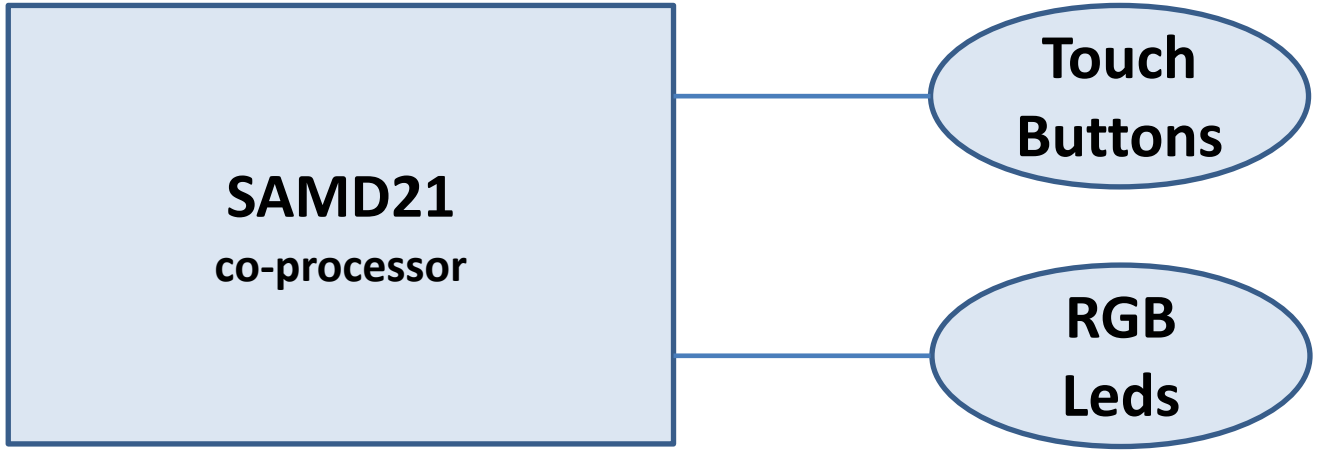
SAMD21 Switches



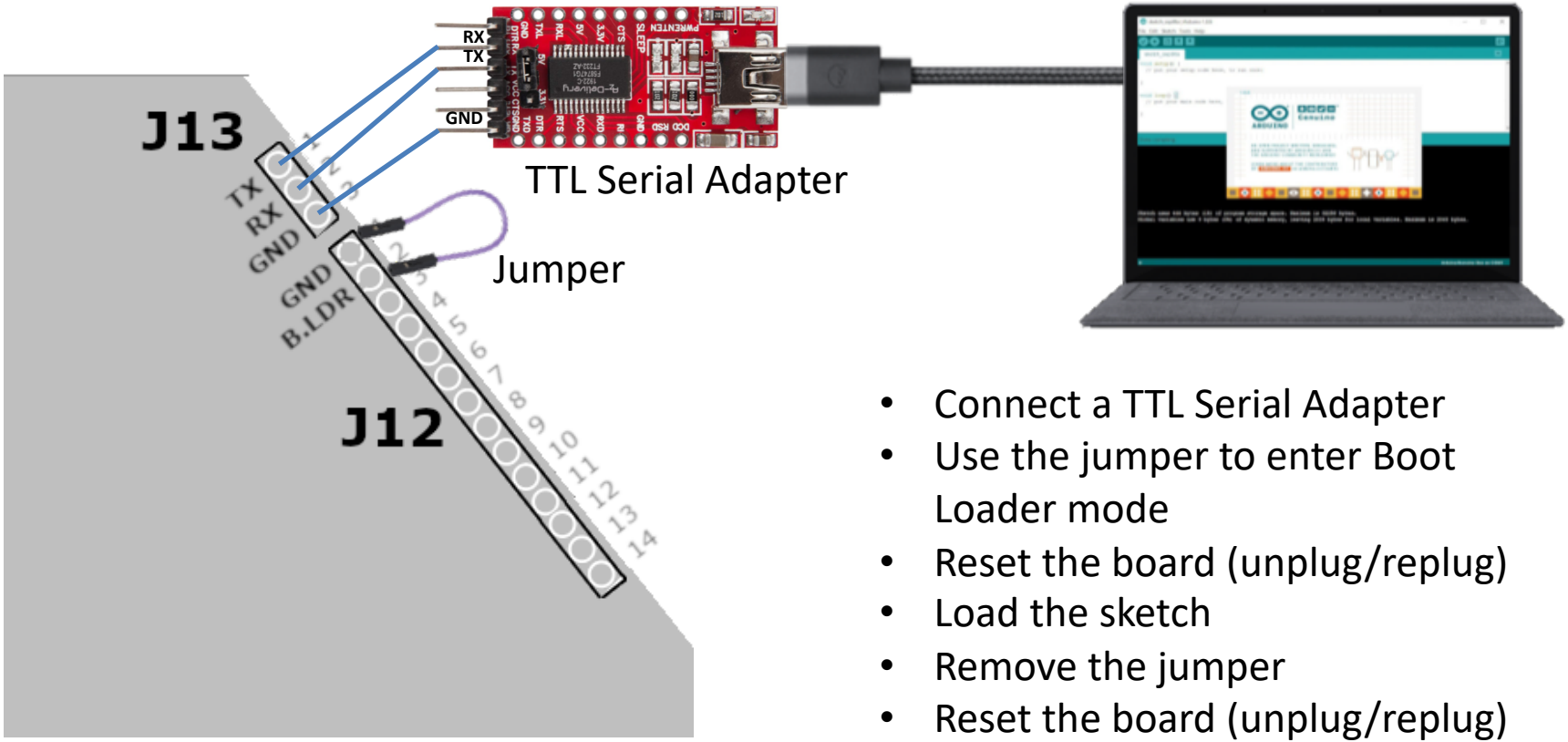
ESP32 – SAMD21



SAMD21 peripherals



ESP32 with Arduino IDE

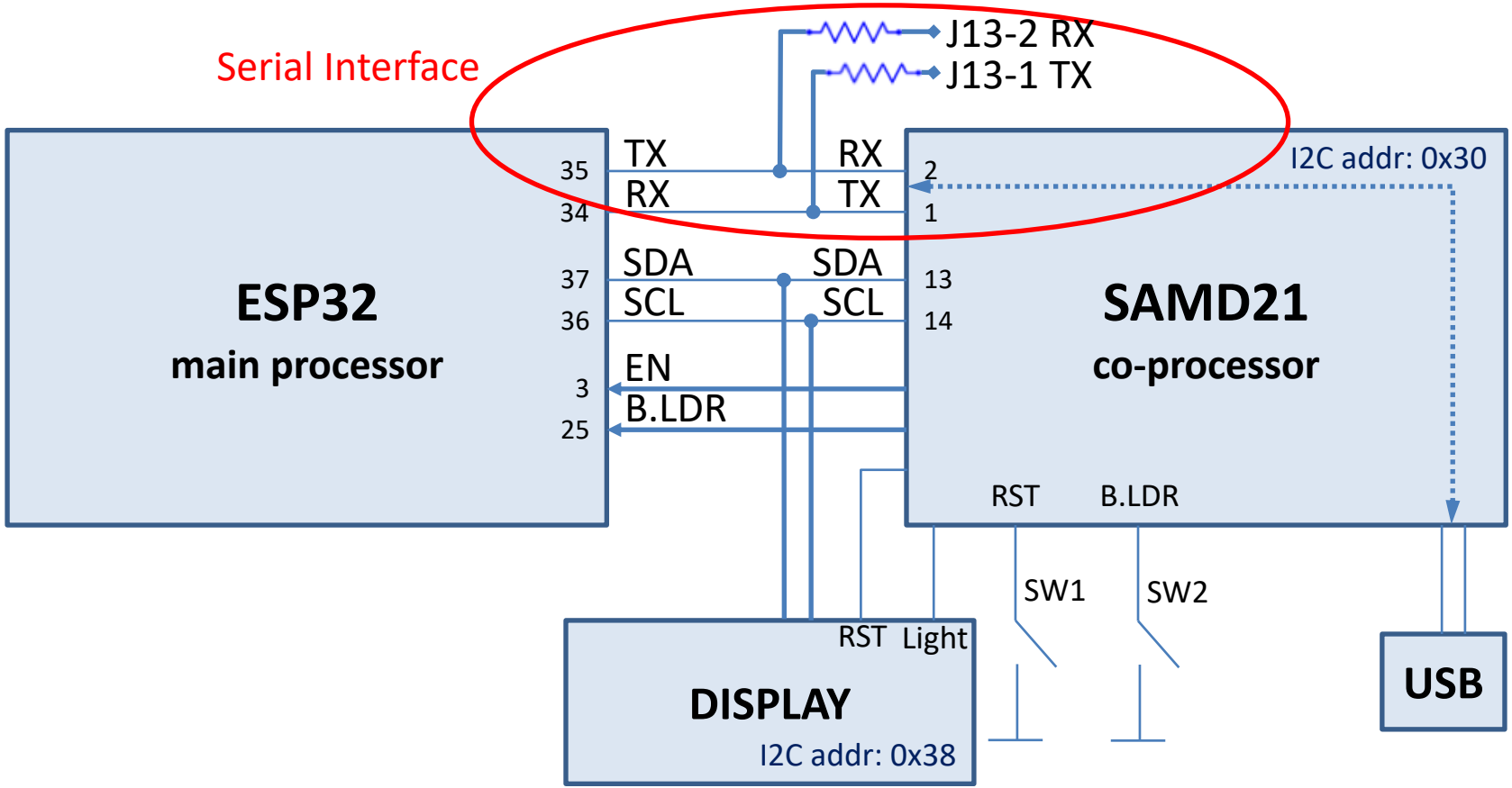


- Connect a TTL Serial Adapter
- Use the jumper to enter Boot Loader mode
- Reset the board (unplug/replug)
- Load the sketch
- Remove the jumper
- Reset the board (unplug/replug)

Hello World with a LED

Video loading the blinking LED on the ESP32

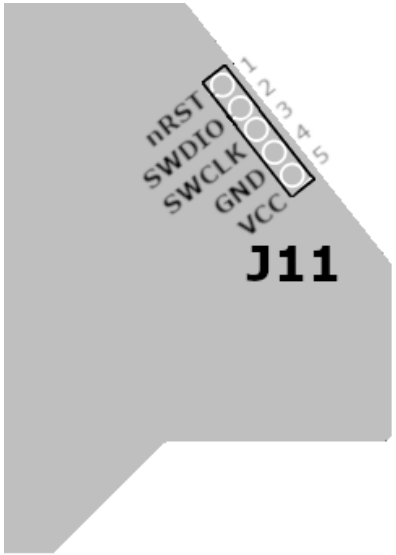
Hello World with a LED



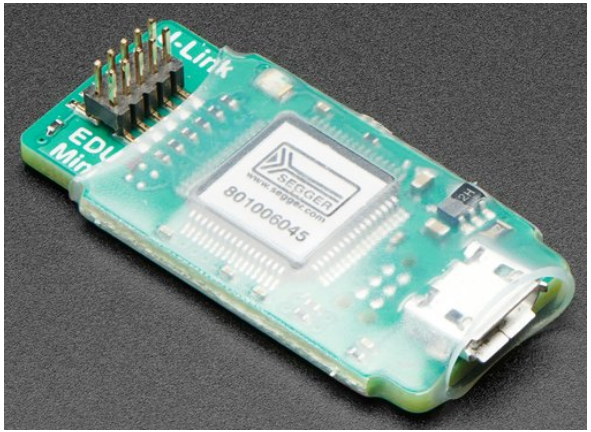
Why re-flash the SAMD21

- I want to use the badge USB interface to program the ESP32 with the Arduino IDE
  - The SAMD21 have to read from USB serial interface and write to the ESP32 serial interface and vice-versa
- I want to light the LEDs, attached to the SAMD21
- I want to use the touch buttons attached to the SAMD21

Re-flashing Hardware



- On the badge we have the SWD (Serial Wire Debug) interface headers
- We can use:
  - Segger J-Link Edu Mini, very good proprietary software

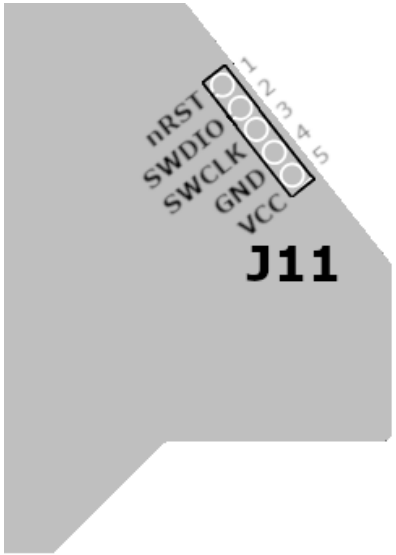


SEGGER J-Link EDU Mini - JTAG/SWD Debugger

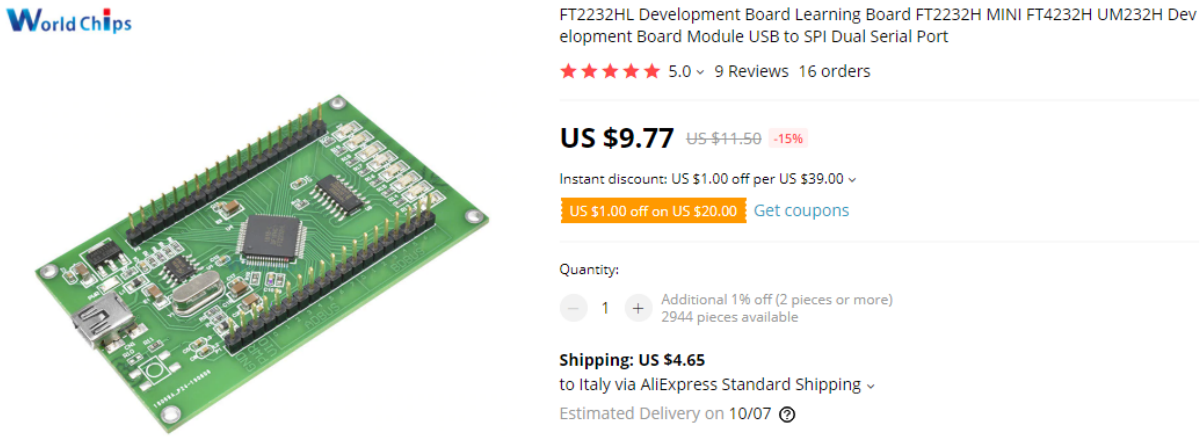
PRODUCT ID: 3571

$19.95

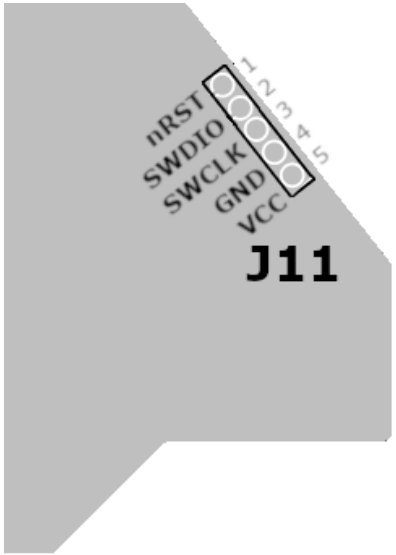
Re-flashing Hardware



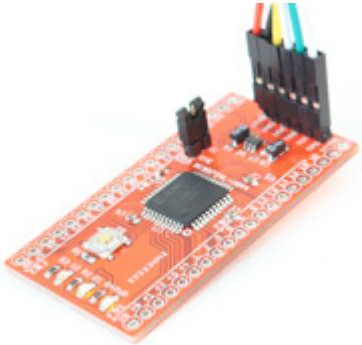
- On the badge we have the SWD (Serial Wire Debug) interface headers
- We can use:
  - Segger J-Link Edu Mini, very good proprietary software
  - Breakout boards with FT2232H and OpenOCD software



Re-flashing Hardware



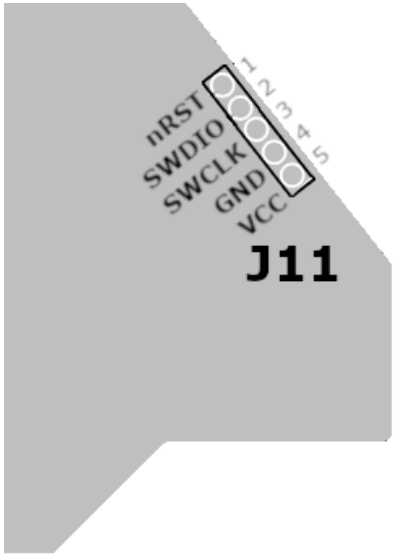
- On the badge we have the SWD (Serial Wire Debug) interface headers
- We can use:
  - Segger J-Link Edu Mini, very good proprietary software
  - Breakout boards with FT2232H and OpenOCD software
  - Bus Bluster (based on FT2232H) and OpenOCD software



Project Summary	
Name:	Bus Blaster
Buy it:	Get one for $34.95 at Seeed Studio
Price:	$34.95
Status:	Mature
Manufacturing:	Shipping
Forum:	Bus Blaster Forum

$ 34.95

Re-flashing Hardware



- On the badge we have the SWD (Serial Wire Debug) interface headers
- We can use:
  - Segger J-Link Edu Mini, very good proprietary software
  - Breakout boards with FT2232H and OpenOCD software
  - Bus Bluster (based on FT2232H) and OpenOCD software
  - Bus Pirate

BUS PIRATE - V3.6A

Item no.: TOL-12942

€29.95

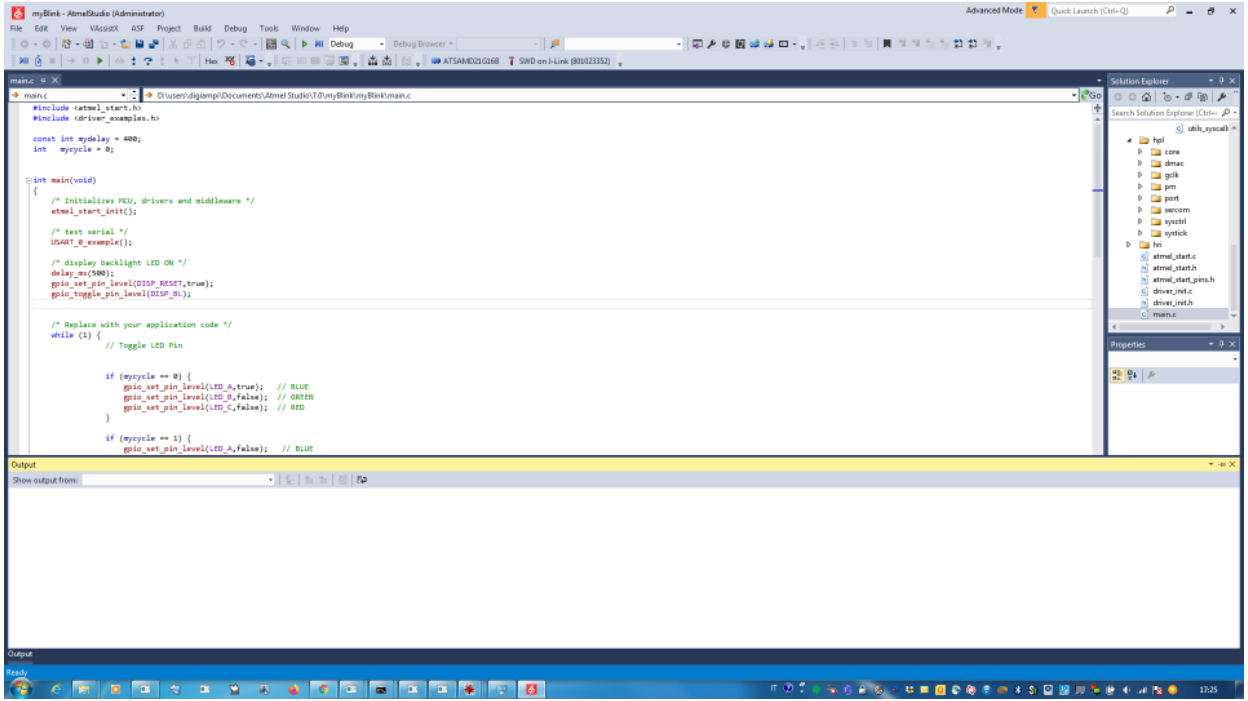
Incl. VAT: €36.24

Qty: 1

In stock

Atmel Studio

- Windows Only



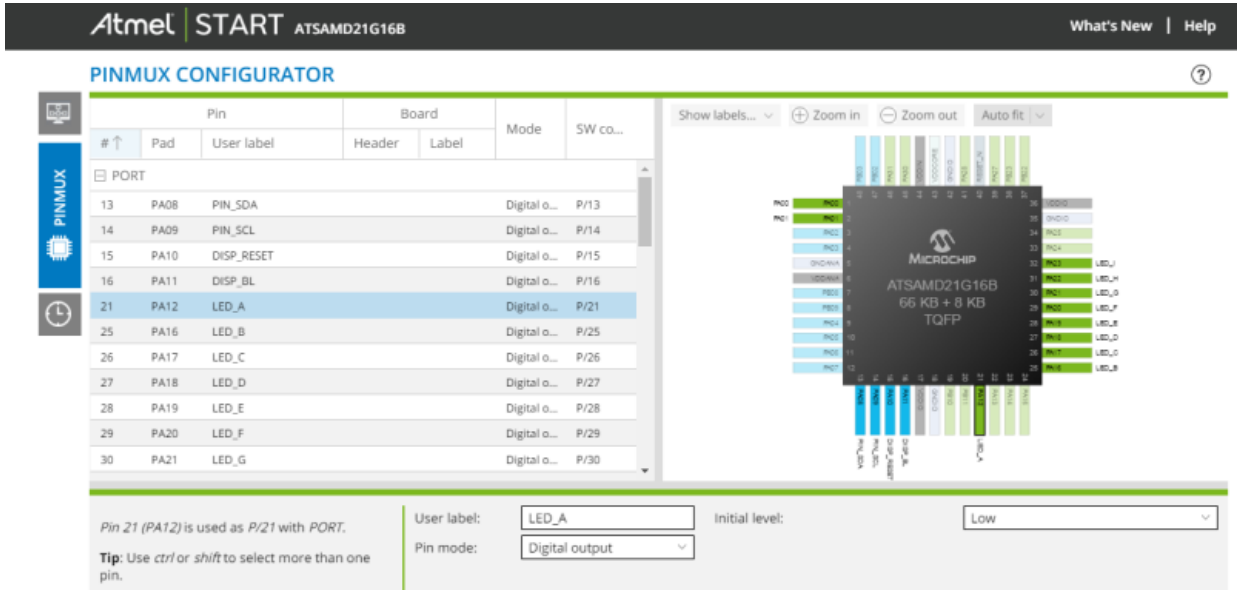
Atmel Studio

- Windows Only
- Over-bloated

Microsoft .NET Framework 4.5.1 Multi-Targeting Pack (ENU)	Microsoft Corporation	26/08/2020	74,5 MB	4.5.50932
Atmel LibUSB0 Driver (x64)	Atmel	26/08/2020	725 KB	7.0.125
Microsoft .NET Framework 4.5.2 Multi-Targeting Pack	Microsoft Corporation	26/08/2020	49,4 MB	4.5.51209
Microsoft System CLR Types for SQL Server 2014	Microsoft Corporation	26/08/2020	5,69 MB	12.0.2402.11
Microsoft .NET Framework 4.5.1 Multi-Targeting Pack	Microsoft Corporation	26/08/2020	49,3 MB	4.5.50932
Microsoft .NET Framework 4.5 Multi-Targeting Pack	Microsoft Corporation	26/08/2020	41,8 MB	4.5.50710
Atmel Driver Files	Atmel Corporation	26/08/2020	5,05 MB	8.1.39
Microsoft .NET Framework 4.5.2 Multi-Targeting Pack (ENU)	Microsoft Corporation	26/08/2020	74,4 MB	4.5.51209
Microsoft SQL Server 2014 Management Objects	Microsoft Corporation	26/08/2020	24,7 MB	12.0.2000.8
Atmel WinUSB	Atmel	26/08/2020	2,58 MB	6.2.32
Microsoft .NET Framework 4.5.1 SDK	Microsoft Corporation	26/08/2020	19,4 MB	4.5.51641
Atmel Segger USB Drivers (501e)	Atmel	26/08/2020	1,90 MB	7.0.417
Microsoft Visual Studio 2015 Shell (Isolated)	Microsoft Corporation	26/08/2020	1,50 GB	14.0.23107.10
Microsoft Visual C++ 2013 Redistributable (x86) - 12.0.21005	Microsoft Corporation	26/08/2020		12.0.21005.1
Microsoft Visual C++ 2013 Redistributable (x64) - 12.0.21005	Microsoft Corporation	26/08/2020		12.0.21005.1
Atmel Studio 7.0	Atmel	26/08/2020	4,12 GB	7.0.2397

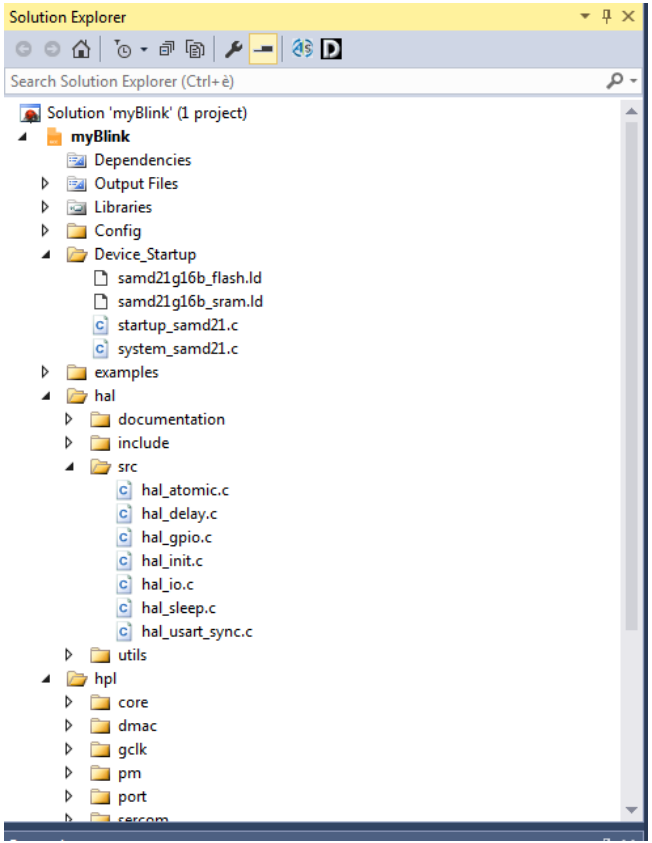
Atmel Studio

- Windows Only
- Over-bloated
- GUI for everything



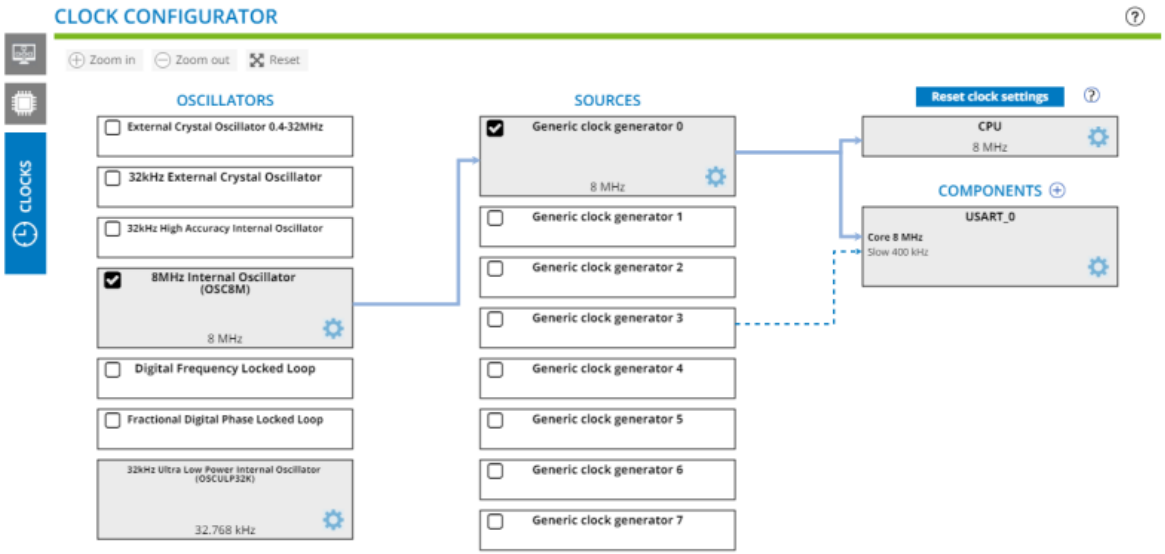
Atmel Studio

- Windows Only
- Over-bloated
- GUI for everything
- Over-bloated ASF framework



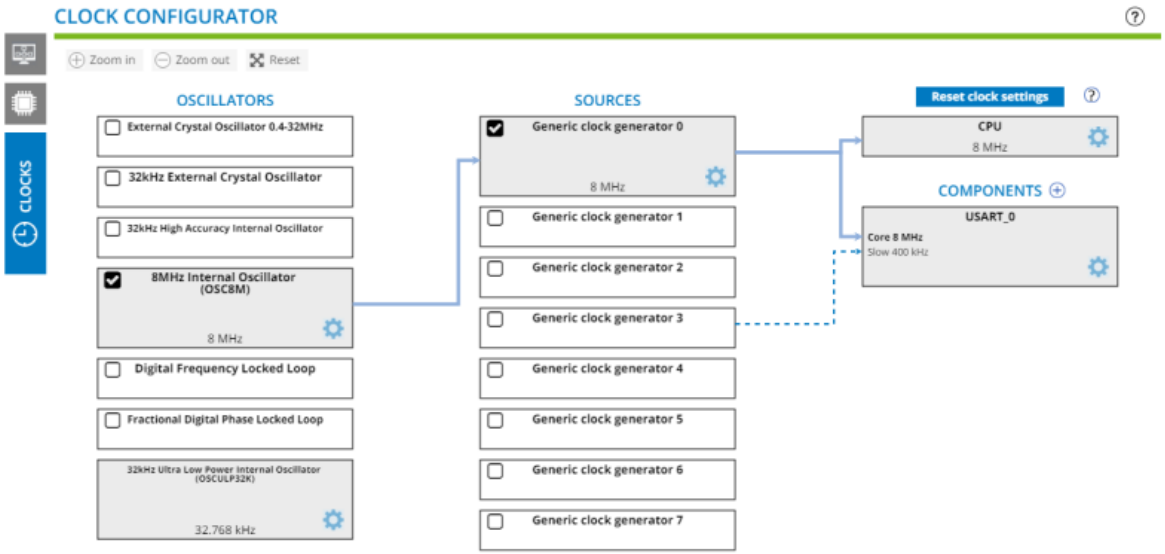
Atmel Studio

- Windows Only
- Over-bloated
- GUI for everything
- Over-bloated ASF framework
- GUI allows illegal configurations



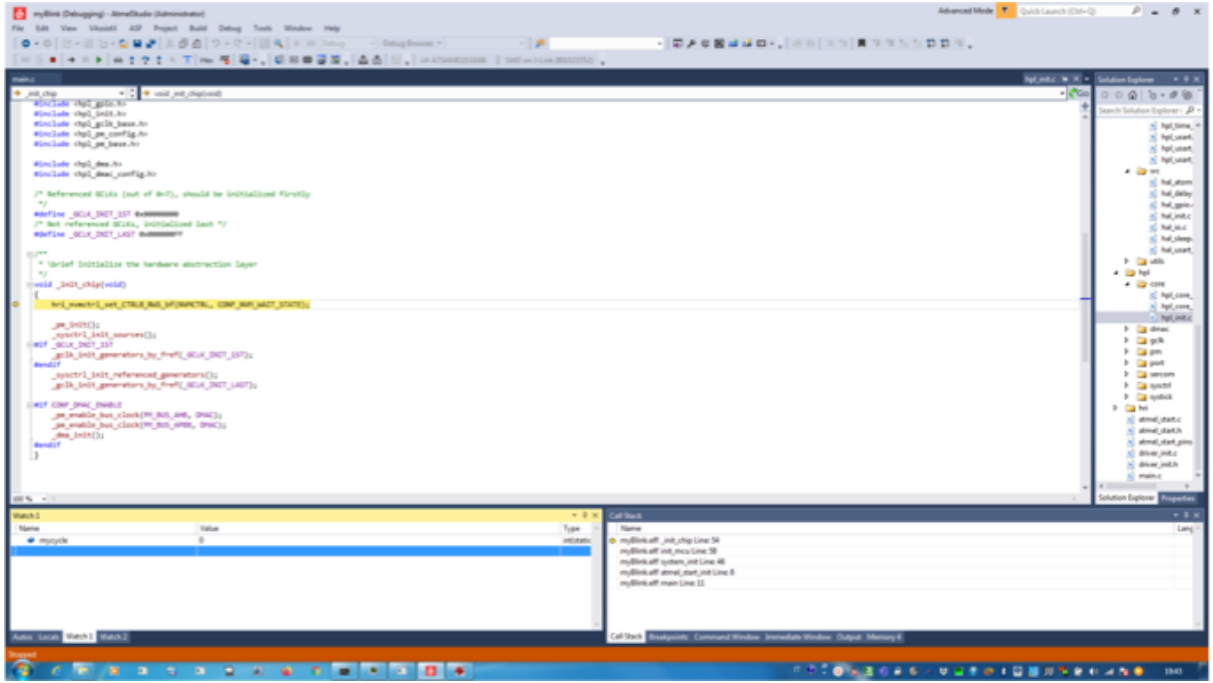
Atmel Studio

- Windows Only
- Over-bloated
- GUI for everything
- Over-bloated ASF framework
- GUI allows illegal configurations
- Complex clocks configuration



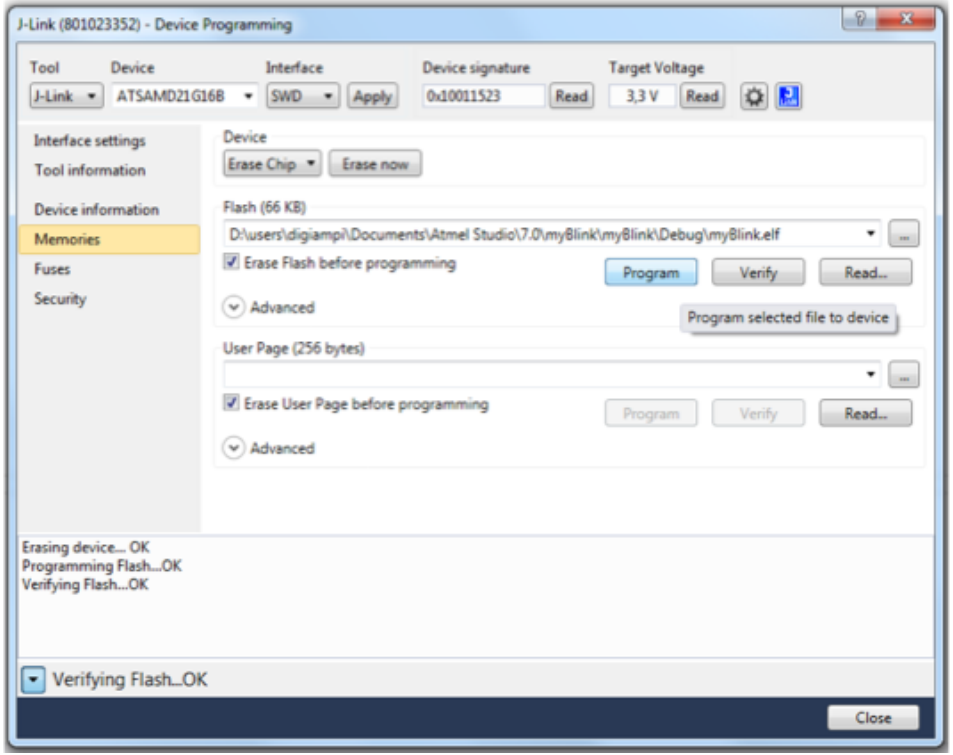
Atmel Studio

- Visual Studio is a good and pleasant GUI



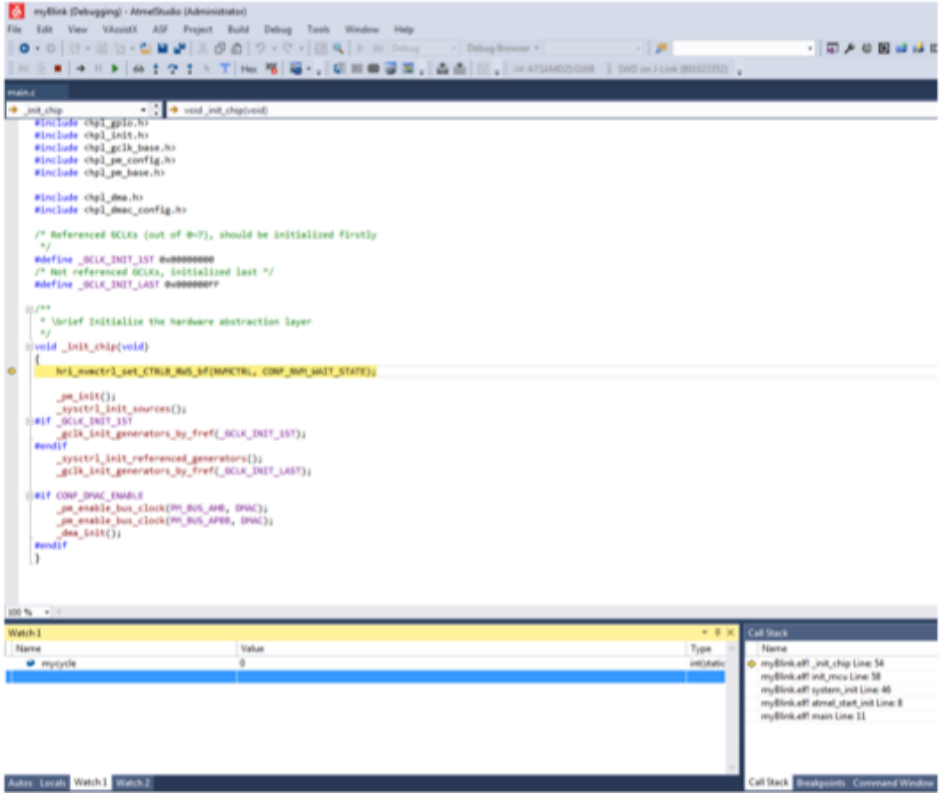
Atmel Studio

- Visual Studio is a good and pleasant GUI
- Good integration with J-Link probe to flash the device



Atmel Studio

- Visual Studio is a good and pleasant GUI
- Good integration with J-Link probe to flash the device
- Easy debugging interface using the J-Link probe and GDB



Hello World with Atmel Studio

- Confirm LED position with a «Hello World» program

Atmel | START main.c

main int main(void)

```c
        while (1) {
                // Toggle LED Pin


                if (mycycle == 0) {
                    gpio_set_pin_level(LED_A,true);   // BLUE
                    gpio_set_pin_level(LED_B,false);  // GREEN
                    gpio_set_pin_level(LED_C,false);  // RED
                }

                if (mycycle == 1) {
                    gpio_set_pin_level(LED_A,false);   // BLUE
                    gpio_set_pin_level(LED_B,true);    // GREEN
                    gpio_set_pin_level(LED_C,false);   // RED
                }

                if (mycycle == 2) {
                    gpio_set_pin_level(LED_A,false);   // BLUE
                    gpio_set_pin_level(LED_B,false);   // GREEN
                    gpio_set_pin_level(LED_C,true);    // RED
                }

                mycycle++;
                if (mycycle > 2) {mycycle = 0;}

                delay_ms(mydelay);

                gpio_toggle_pin_level(LED_F);  // from 0 to 1 LED D5 off
                delay_ms(mydelay);

                gpio_toggle_pin_level(LED_D);  // from 0 to 1 LED D1 off
                delay_ms(mydelay);

                gpio_toggle_pin_level(LED_H);  // from 0 to 1 LED D6 off
                delay_ms(mydelay);

                gpio_toggle_pin_level(LED_I);  // from 0 to 1 LED D3 off
                delay_ms(mydelay);

                gpio_toggle_pin_level(LED_E);  // from 0 to 1 LED D7 off
                delay_ms(mydelay);

                gpio_toggle_pin_level(LED_G);  // from 0 to 1 LED D4 off
                delay_ms(mydelay);

                USART_0_example();

        }
}
```

100 %

Hello World with Atmel Studio

- Confirm LED position with a ~~«Hello World»~~ program

  Merry Christmas

Hello World with Atmel Studio

Video loading the blinking LED on the ESP32

Working with Atmel Studio

- Spent few weeks trying to program a USB serial to ESP32 serial bridge
- Difficult to find mix and merge available examples
- Writing something from scratch seems impossible without studying the 1000+ pages SAMD21 data sheet and the 500+ pages Atmel Studio manual!
- The ”Atmel Studio” way is the opposite of the ”Arduino way”
- Restart searching on Internet how to bring our SAMD21 in the Arduino IDE platform

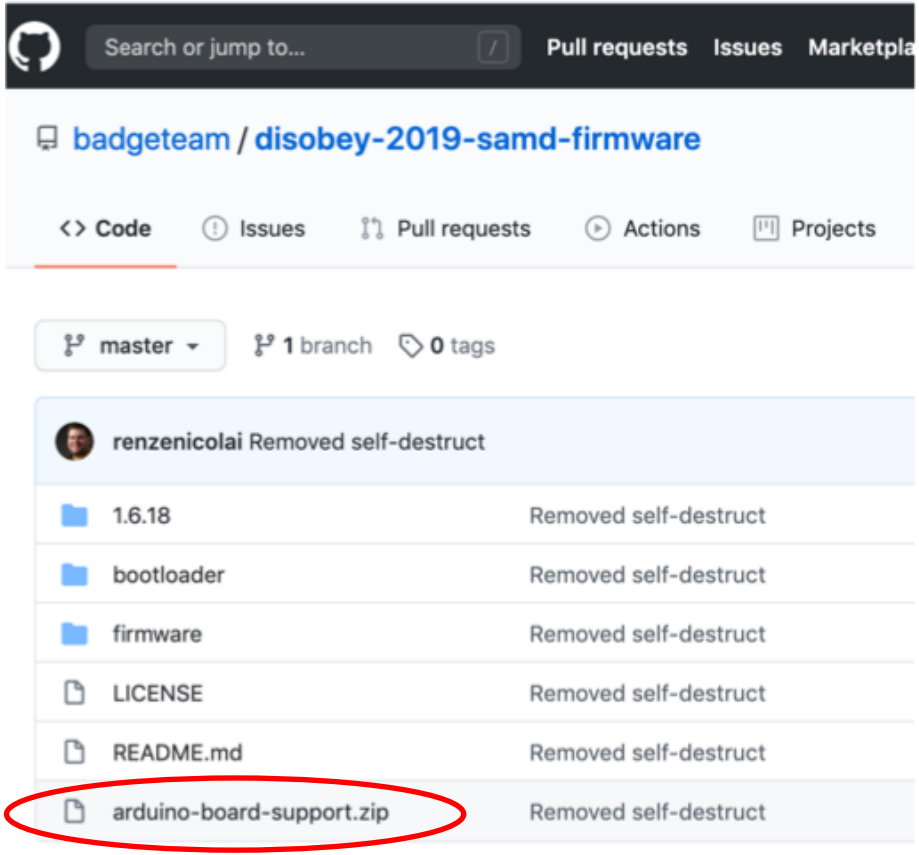
The Arduino Way

A lot of beginners approaching electronics for the first time think that they have to learn how to build everything from scratch. This is a waste of energy: what you want is to be able to confirm that something's working very quickly so that you can motivate yourself to take the next step.

Massimo Banzi
co-founder of Arduino

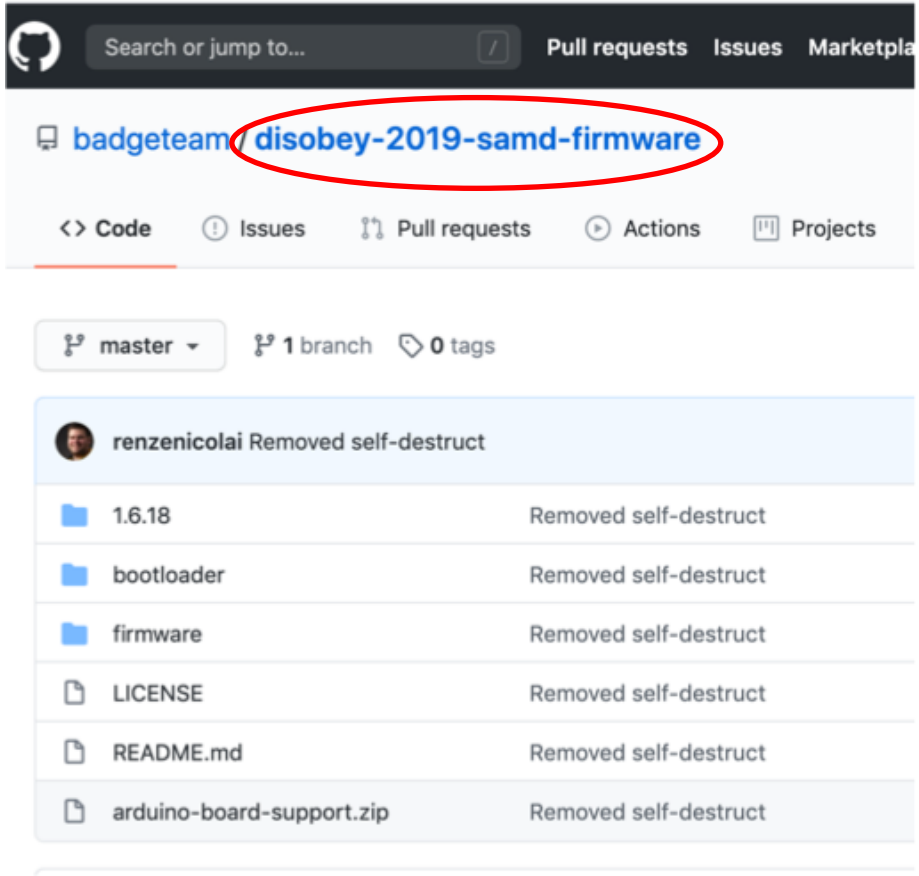
SAMD21 Arduino IDE

- On GitHub the SAMD21G16 was already ported to Arduino IDE



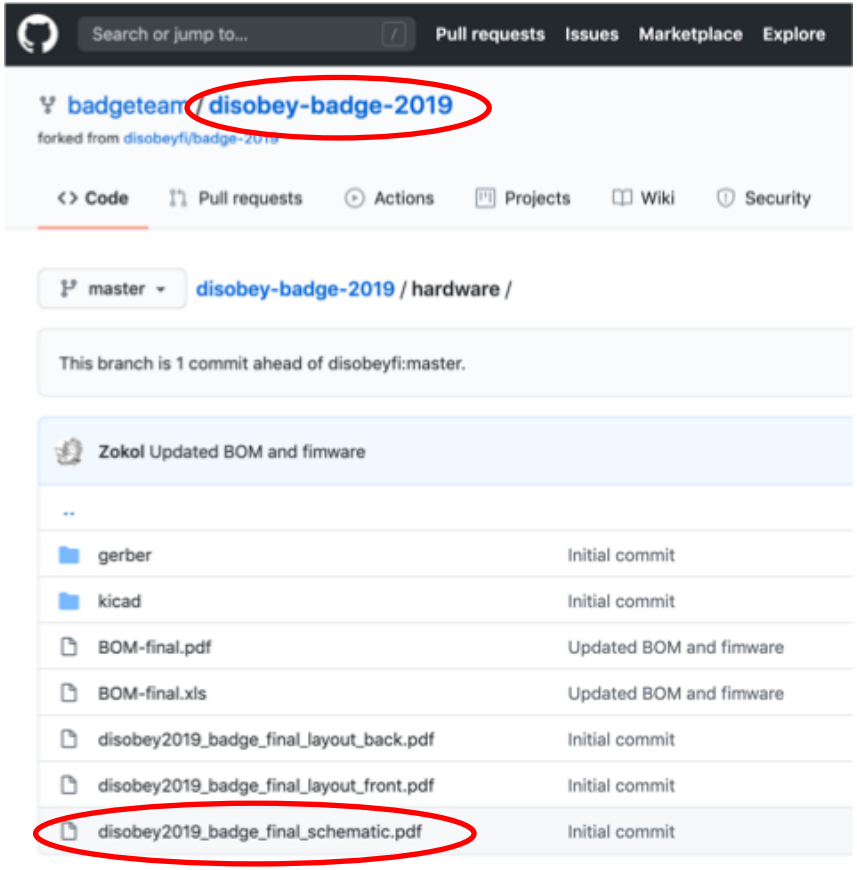
SAMD21 Arduino IDE

- On GitHub the SAMD21G16 was already ported to Arduino IDE
- The project has the firmware for the “Disobey 2019” badge, very similar to our badge



SAMD21 Arduino IDE

- On GitHub the SAMD21G16 was already ported to Arduino IDE
- The project has the firmware for the “Disobey 2019” badge, very similar to our badge
- Related repository with badge details, including schematics



USB Serial to ESP2 serial bridge

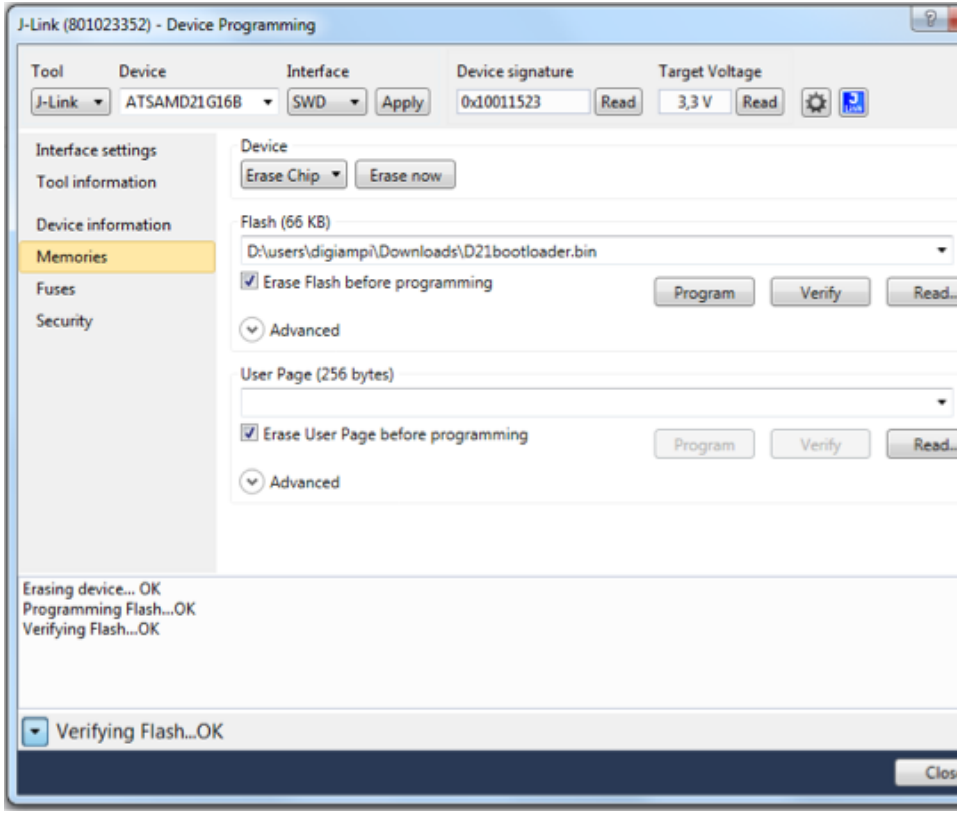
- In the Arduino IDE the USB Serial to ESP32 serial bridge is only 10 lines of code!

```
inline void update_serial()
{
  while (Serial.available()) {
    Serial1.write(Serial.read());
  }

  while (Serial1.available()) {
    Serial.write(Serial1.read());
  }
}
```

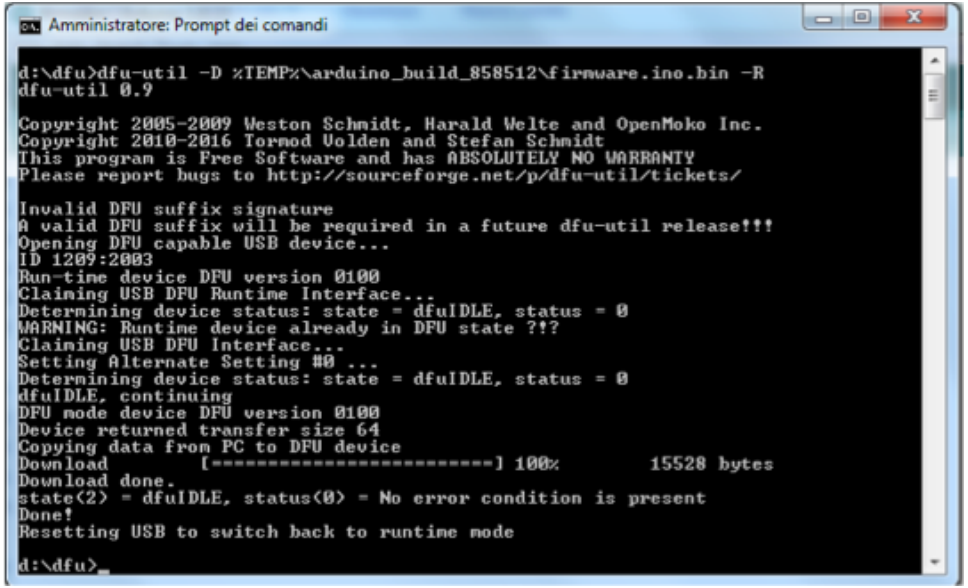
USB Serial to ESP2 serial bridge

- In the Arduino IDE the USB Serial to ESP2 serial bridge is only 10 lines of code!
- Write the bootloader to the SAMD21 using Atmel Studio



USB Serial to ESP2 serial bridge

- In the Arduino IDE the USB Serial to ESP2 serial bridge is only 10 lines of code!
- Write the bootloader to the SAMD21 using Atmel Studio
- Compile the “firmware.ino” and load to the SAMD21 using dfu-util



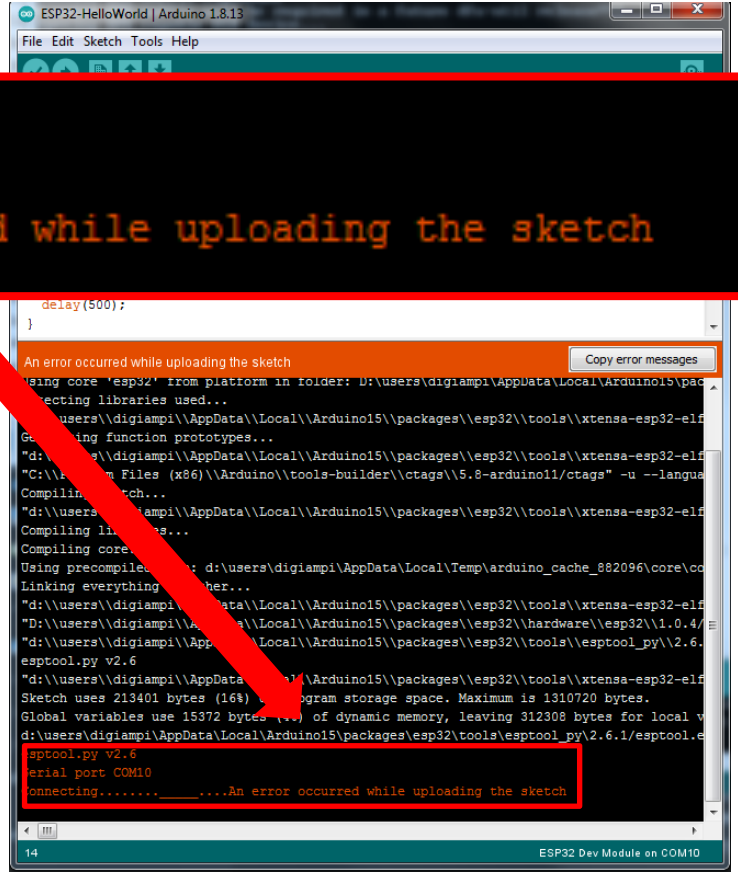
USB Serial to ESP2 serial bridge

HACKTIVITY
The IT Security Festival in Central and Eastern Europe

- In the Arduino IDE the USB

```
esptool.py v2.6
Serial port COM10
Connecting........_____....An error occurred while uploading the sketch
```

- Write the bootloader to the SAMD21 using Atmel Studio
- Compile the “firmware.ino” and load to the SAMD21 using dfu-util
- Use the Arduino IDE with ESP32: timeout!



USB Serial to ESP2 serial bridge

- The problem was that the serial bridge was not fast enough

```
inline void update_serial()
{
  while (Serial.available()) {
    Serial1.write(Serial.read());
  }

  while (Serial1.available()) {
    Serial.write(Serial1.read());
  }
}
```

USB Serial to ESP2 serial bridge

- The problem was that the serial bridge was not fast enough
- Arduino IDE hides complexity, but can generate code that can be very inefficient

```
inline void update_serial()
{
  while (Serial.available()) {
    Serial1.write(Serial.read());
  }

  while (Serial1.available()) {
    Serial.write(Serial1.read());
  }
}
```

USB Serial to ESP2 serial bridge

- The problem was that the serial bridge was not fast enough
- Arduino IDE hides complexity, but can generate code that can be very inefficient
- Increase the serial buffer in the SAMD21 Arduino configuration

```
// Define constants and variables for buffering incoming serial data.  We're
// using a ring buffer (I think), in which head is the index of the location
// to which to write the next incoming character and tail is the index of the
// location from which to read.
#define SERIAL_BUFFER_SIZE 64
```

```
// Define constants and variables for buffering incoming serial data.  We're
// using a ring buffer (I think), in which head is the index of the location
// to which to write the next incoming character and tail is the index of the
// location from which to read.
#define SERIAL_BUFFER_SIZE 512
```

USB Serial to ESP2 serial bridge

- The problem was that the serial bridge was not fast enough
- Arduino IDE hides complexity, but can generated code that can be very inefficient
- Increase the serial buffer in the SAMD21 Arduino configuration
- Rewrite the “serial bridge” function to be non-blocking

```
inline void my_update_serial()
{
  while ((Serial.available()  and Serial1.availableForWrite())
      or (Serial1.available() and Serial.availableForWrite())) {

    if (Serial.peek() >= 0) {
      if (Serial1.availableForWrite() > 0) {
        Serial1.write(Serial.read());
      }
    }

    if (Serial1.peek() >= 0) {
      if (Serial.availableForWrite() > 0) {
        Serial.write(Serial1.read());
      }
    }

  }
}
```

Serial bridge Working!

Demo Video loading an ESP32 sketch

Conclusion

- Success in using Arduino IDE to program the ESP32 on the badge
- Still to do: modifying an existing, SPI based, library do drive our I2C display
- Something learned once more
  - Don’t get stuck, move on with a different approach
  - Partial info and similar product info can be very useful in any reverse engineering project

HACKTIVITY
The IT Security Festival in Central and Eastern Europe

Thank You
Question Time

Valerio Di Giampietro
http://va.ler.io
v@ler.io
@valerio
youtube.com/makemehack