

## 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](mailto:v@ler.io)

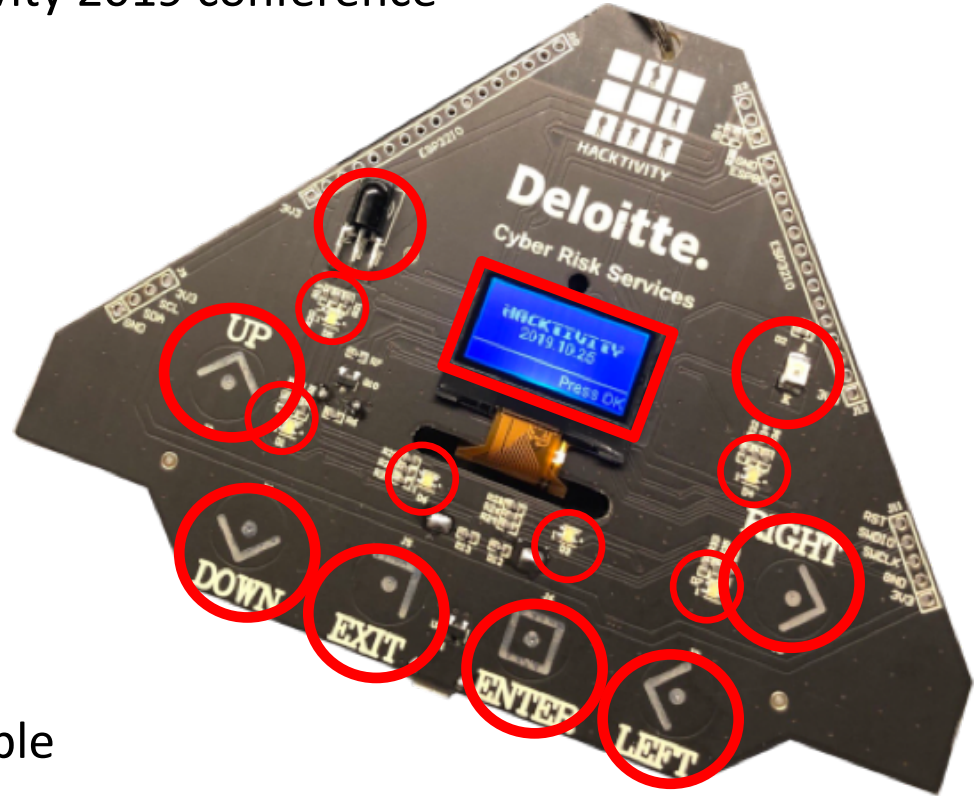
[@valerio](#)

[youtube.com/makemehack](https://youtube.com/makemehack)

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

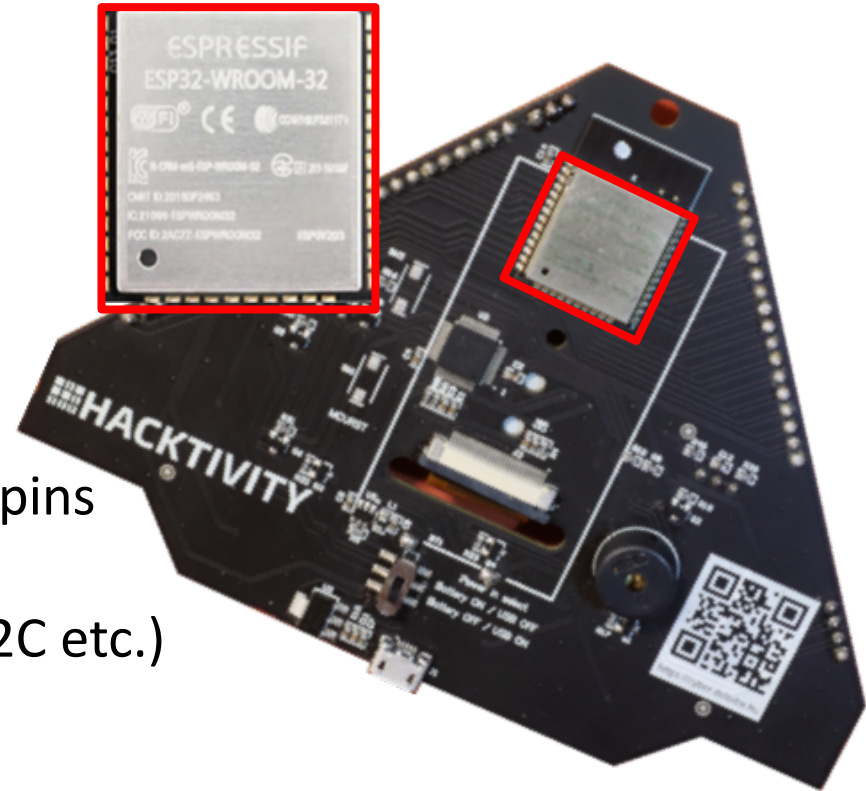
# The Hacktivity 2019 badge

- Distributed last year at Hacktivity 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



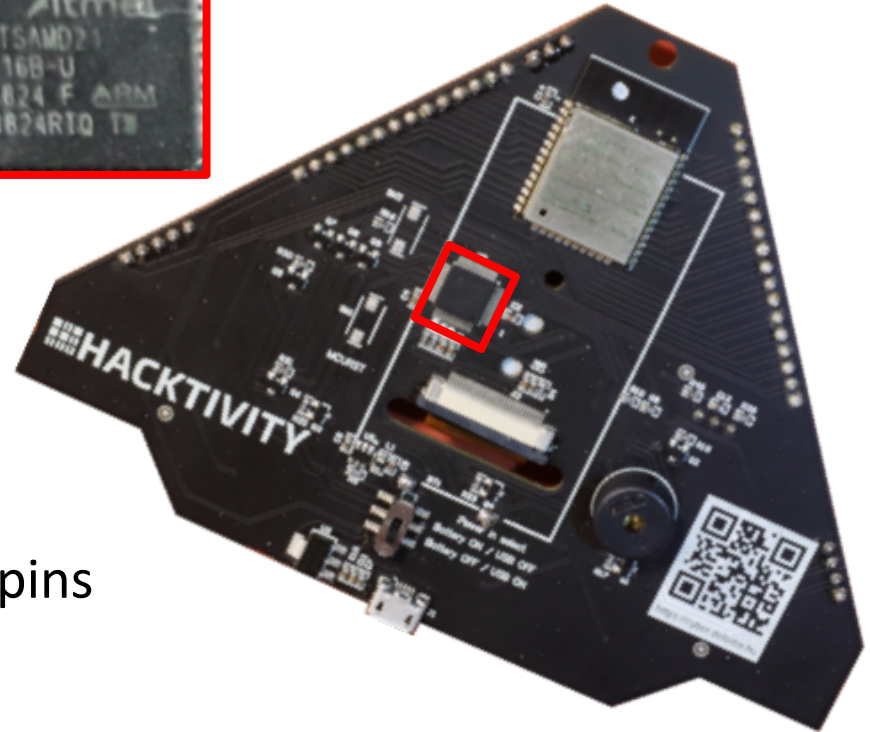
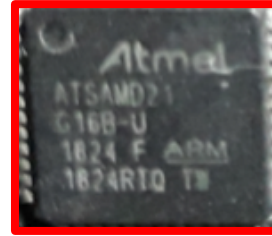
# The Hacktivity 2019 badge

- 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

- Powerful co-processor
- ATSAM21G16B
  - ARM Cortex-M0+ CPU
  - 64Kb flash memory
  - 8Kb SRAM
  - SWD interface
  - USB interface
  - Serial interfaces
  - Digital, analog and PWM I/O pins
  - Touch sensors

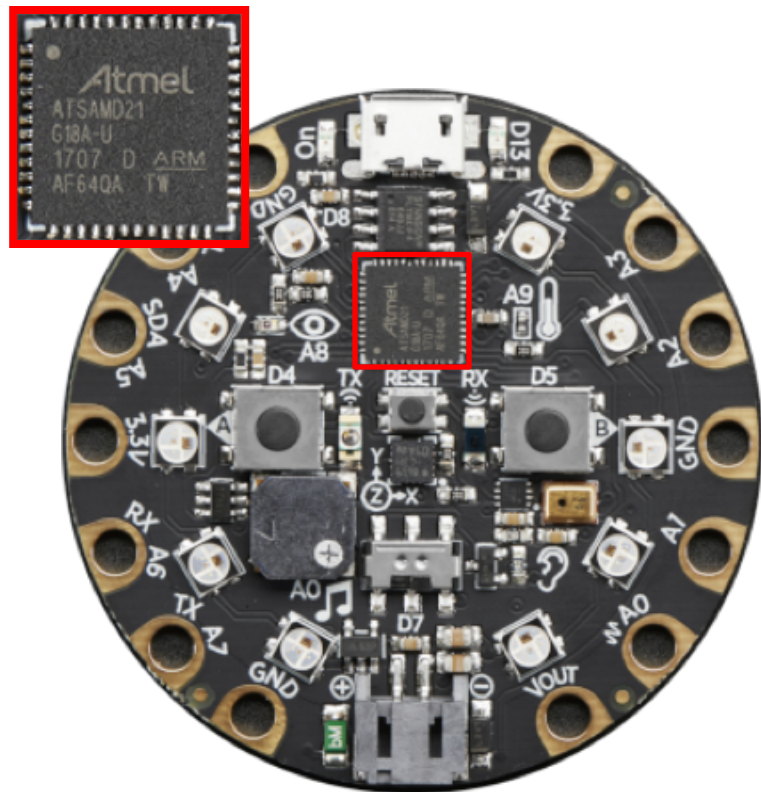


# Bricking the badge

- 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

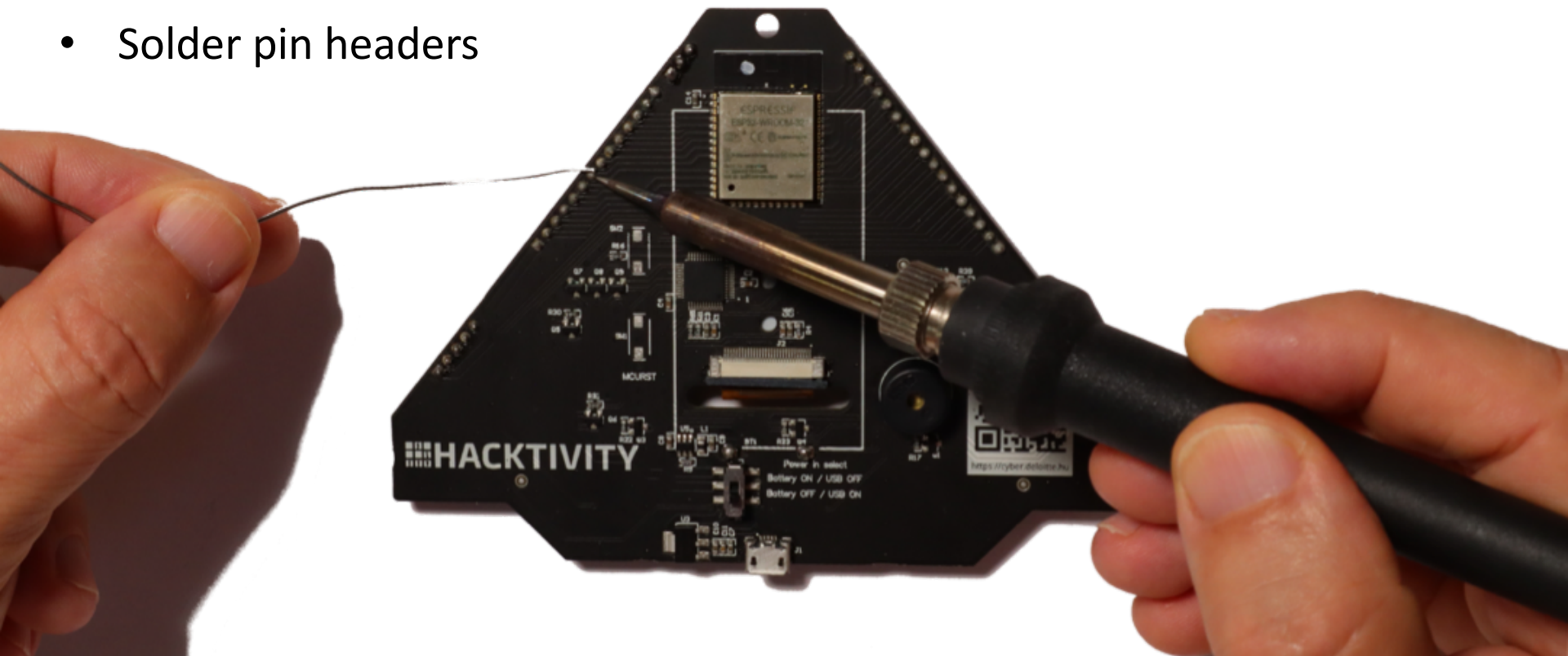


- «Adafruit Playground Express»
  - Based on ATSAM~~D~~21G18A
  - 256Kb flash memory (4x our badge)
  - 32Kb SRAM (4x our badge)



# Reversing the PCB

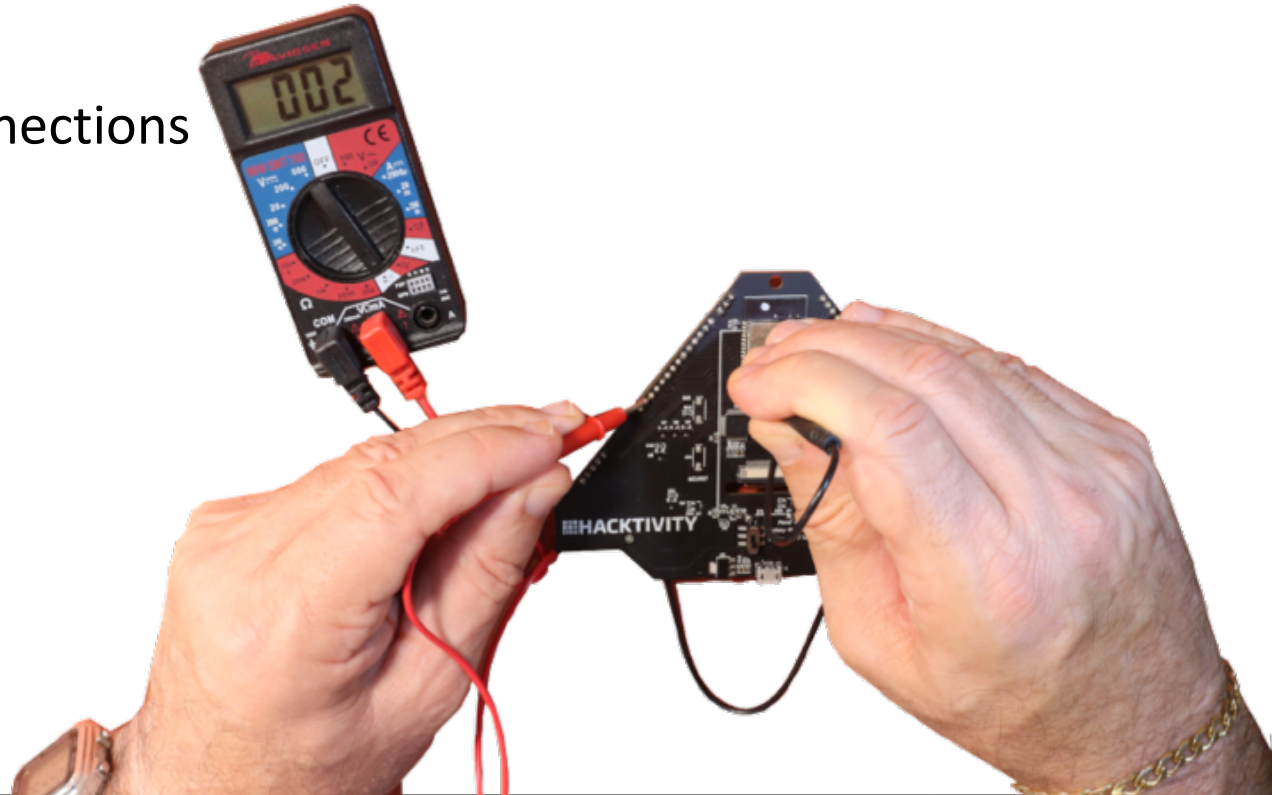
- Solder pin headers





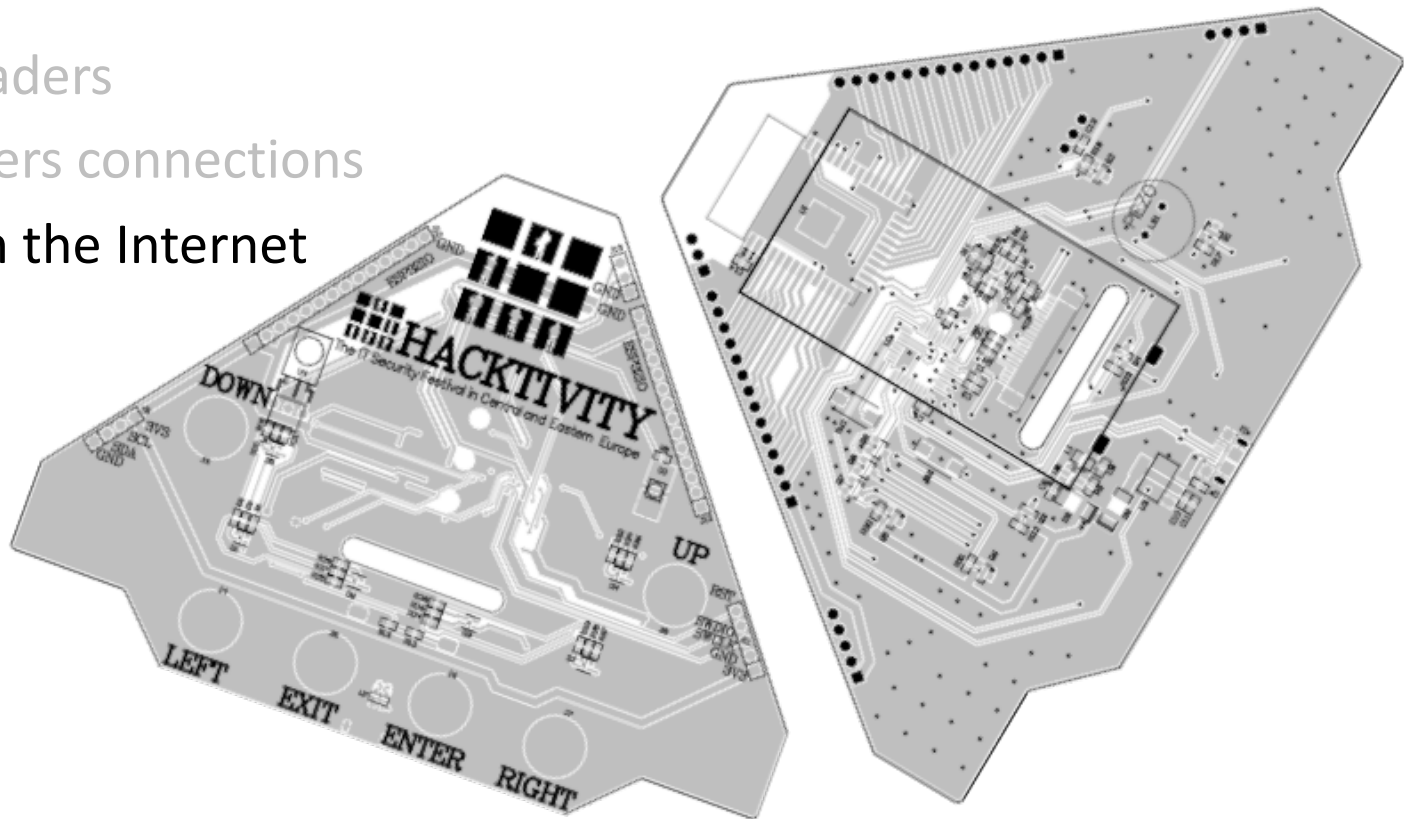
# Reversing the PCB

- Solder pin headers
- Find pin headers connections



# Reversing the PCB

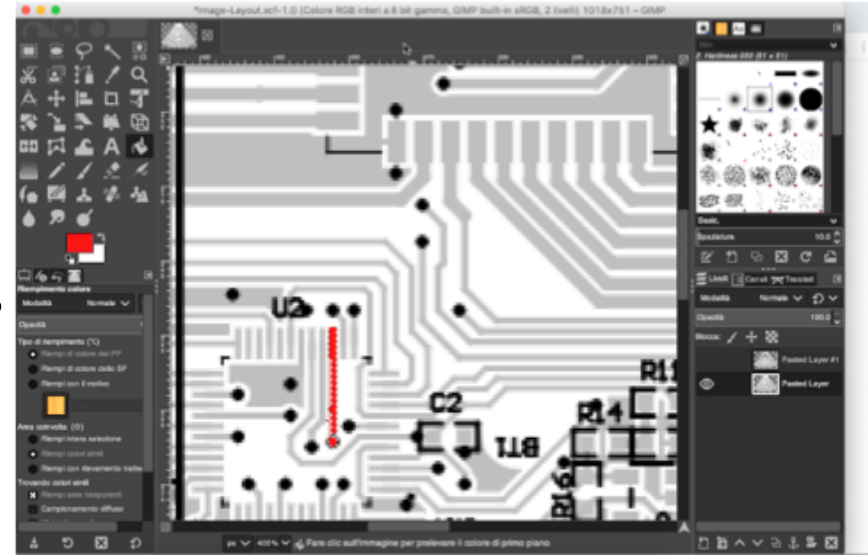
- Solder pin headers
- Find pin headers connections
- Search info on the Internet





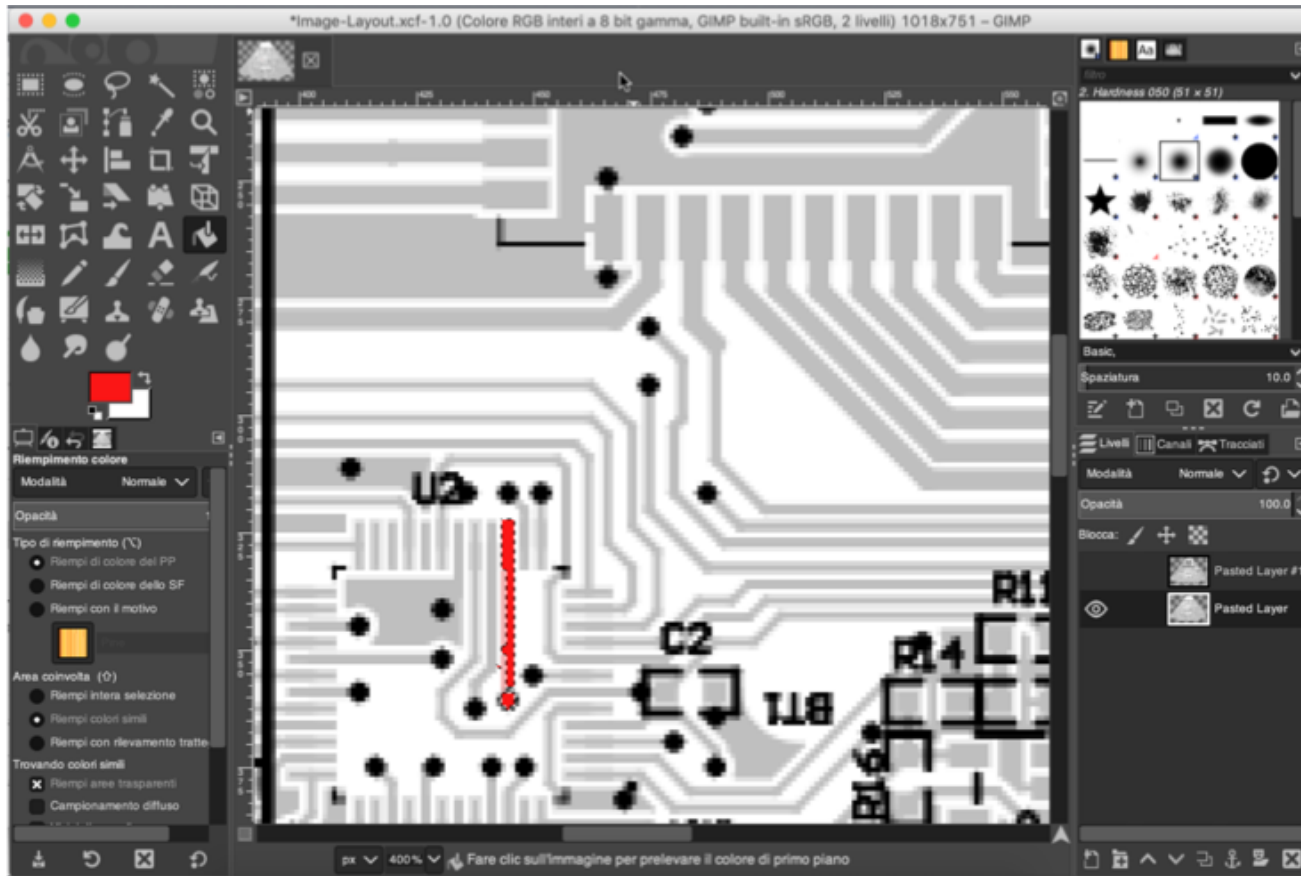
# Reversing the PCB

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

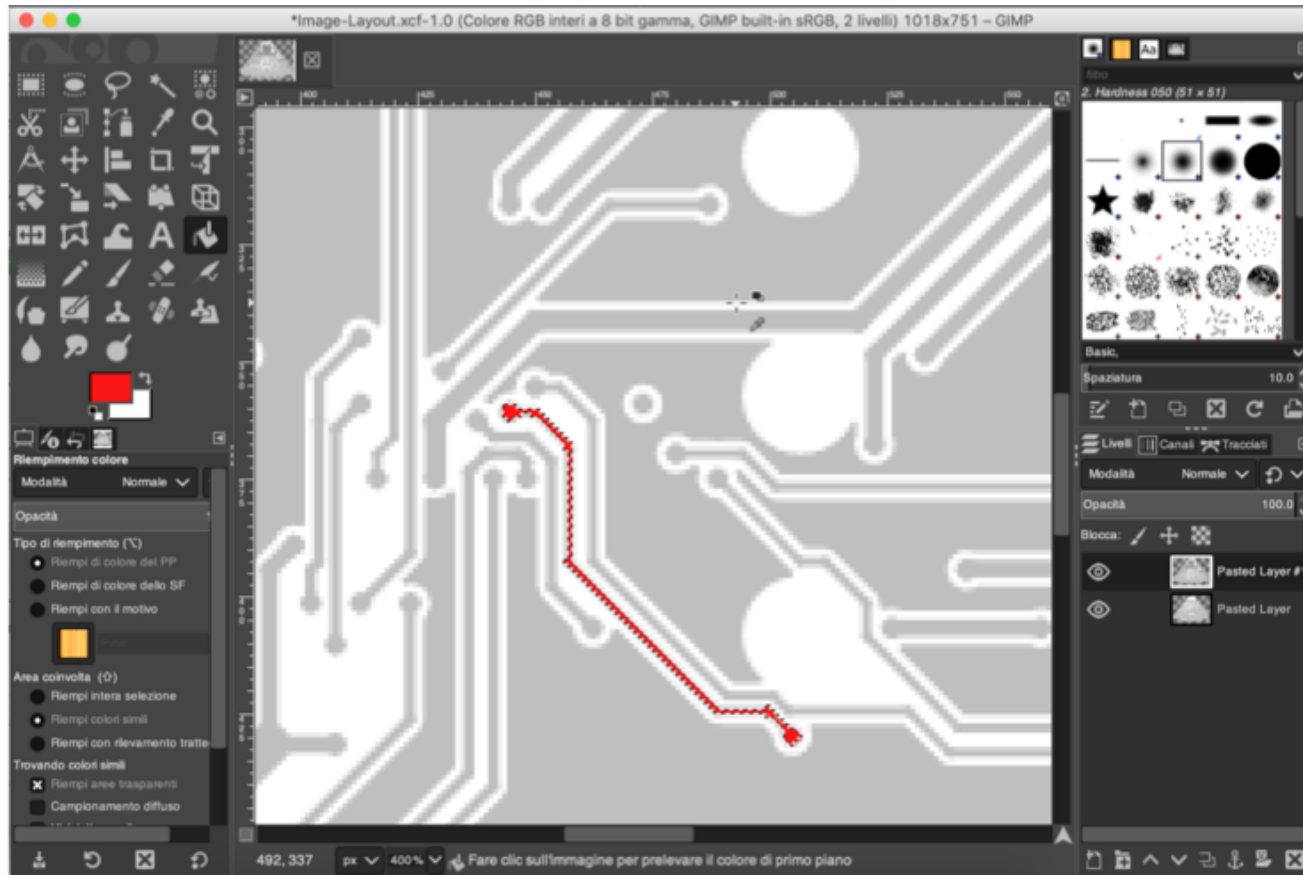




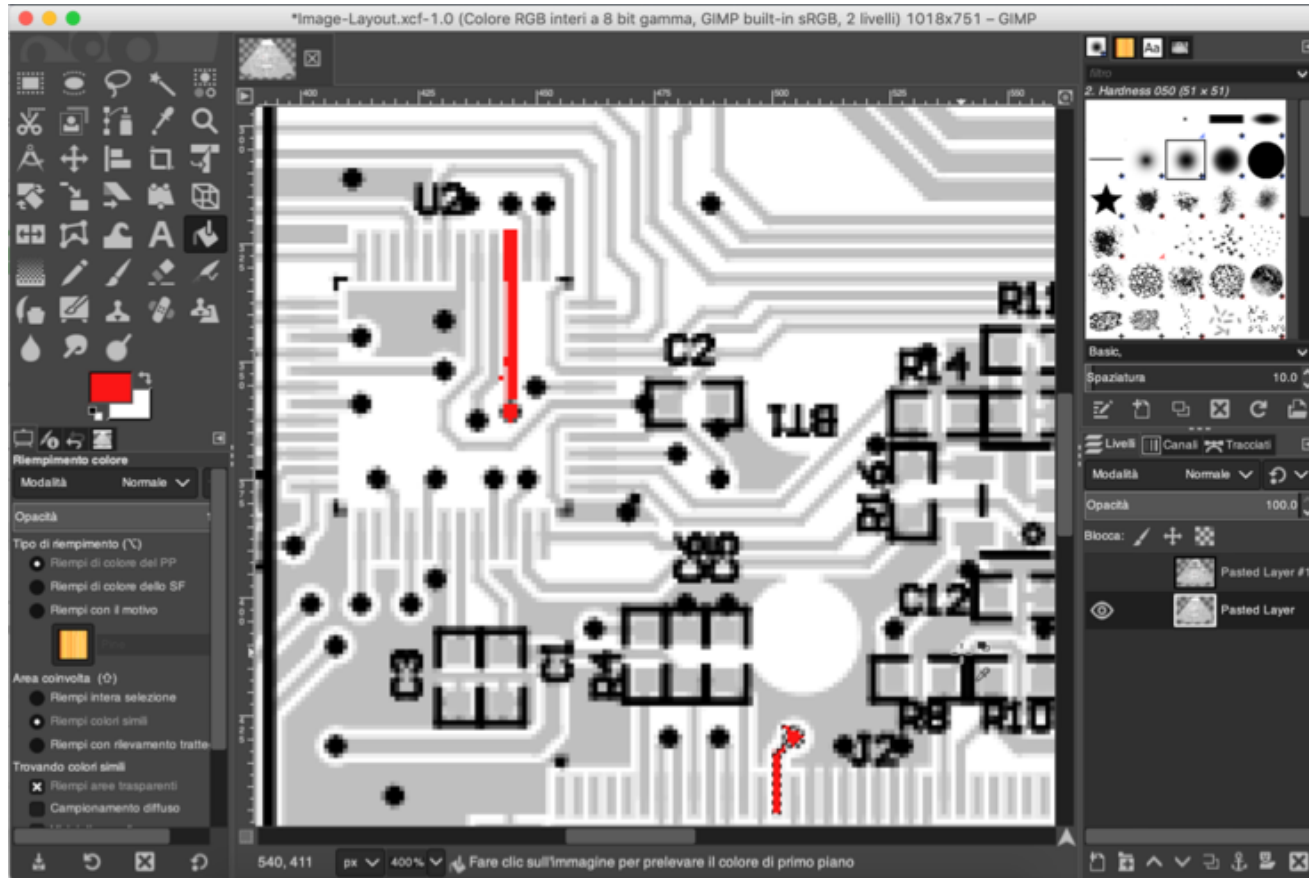
# Reversing the PCB



# Reversing the PCB



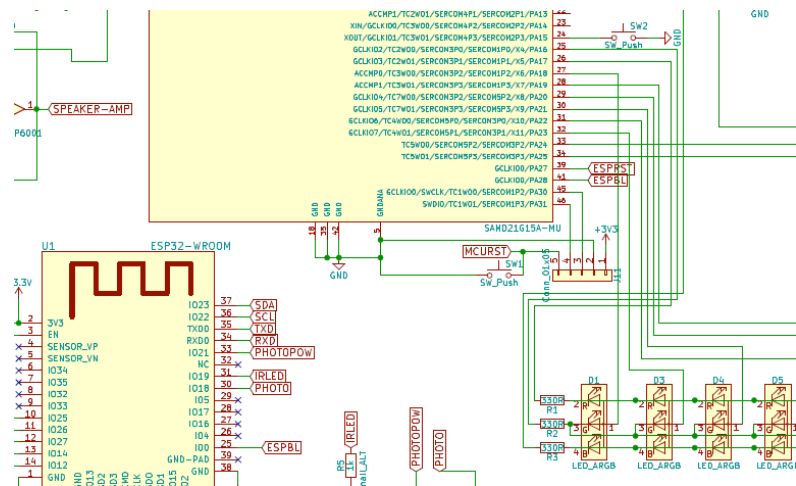
# Reversing the PCB





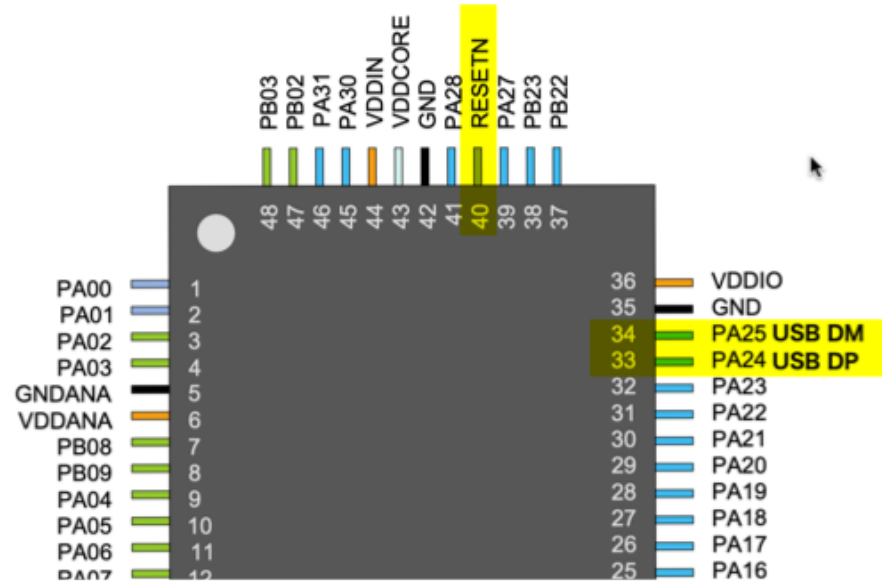
# Reversing the PCB

- 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

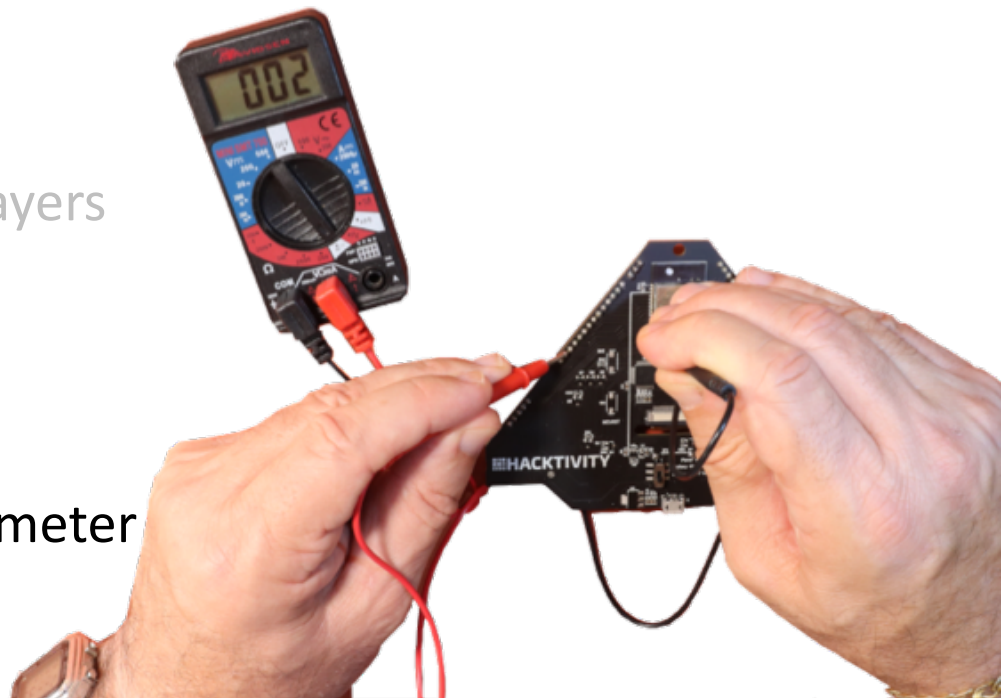
- Solder pin headers
- Find pin headers connections
- Search info on the Internet
- Use Gimp to follow traces on 2 layer PCB
- Use a similar projects as hint
- Use SOCs data sheets as hint

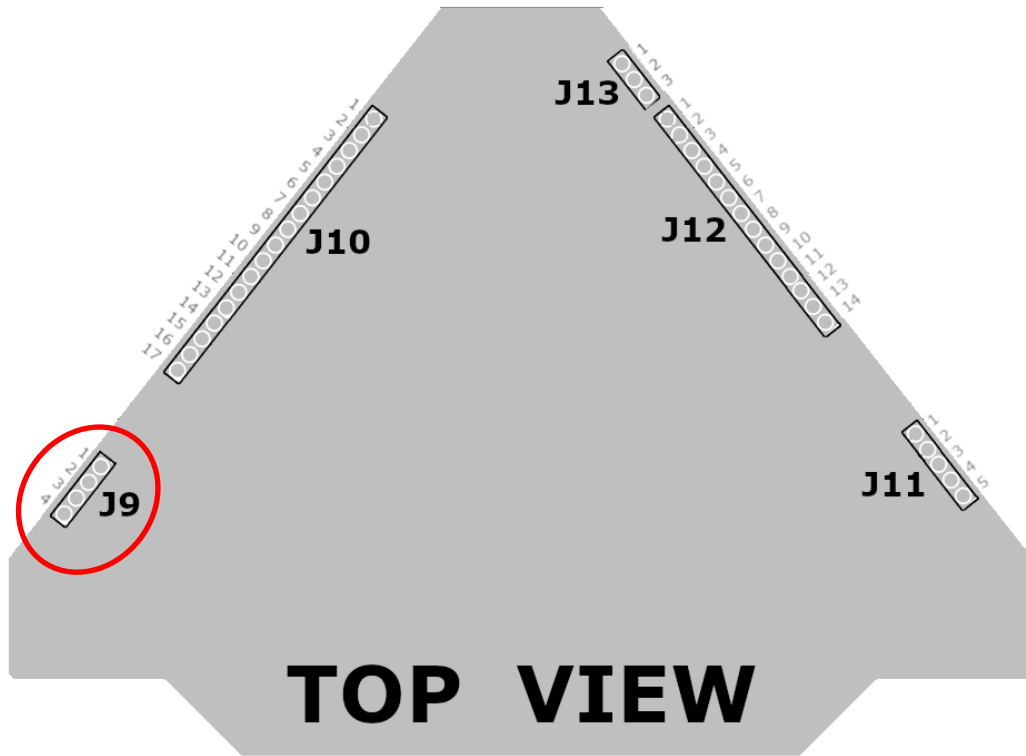


Pin <sup>(1)</sup>		C	D	E	F	G
SAMD21G I/O Pin	SERCOM <sup>(2)(3)</sup>	SERCOM-ALT	TC <sup>(4)</sup> /TCC	TCC	COM	
33	PA24	SERCOM3/ PAD[2]	SERCOM5/ PAD[2]	TC5/WO[0]	TCC1/ WO[2]	USB/DM
34	PA25	SERCOM3/ PAD[3]	SERCOM5/ PAD[3]	TC5/WO[1]	TCC1/ WO[3]	USB/DP

# Reversing the PCB

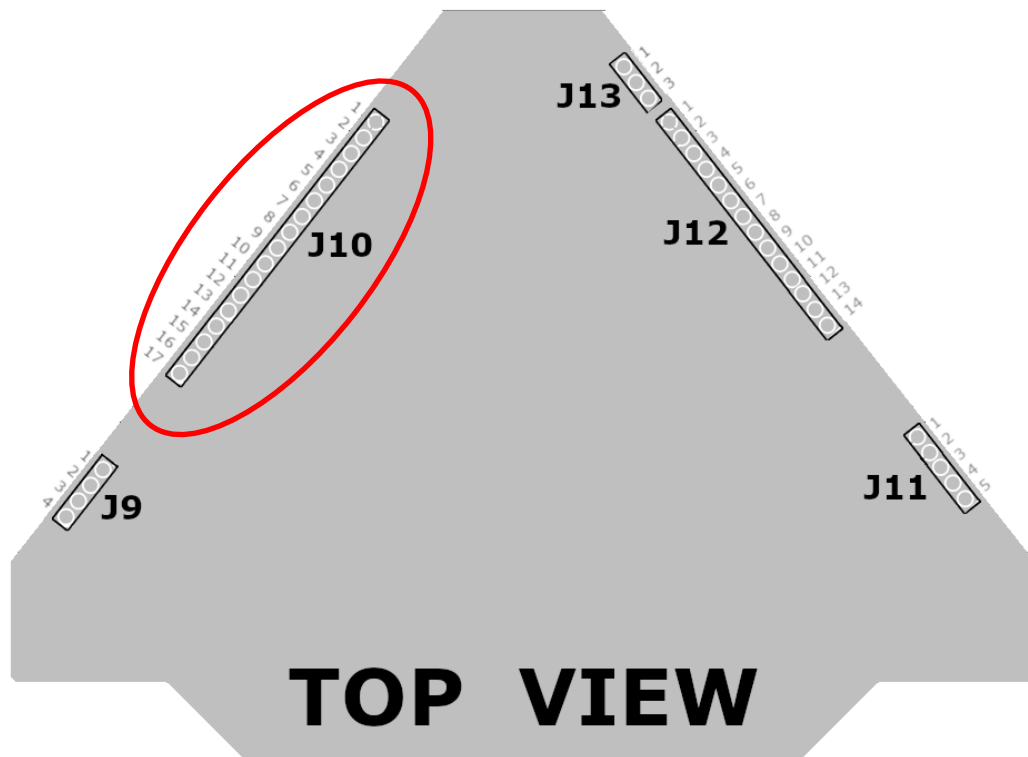
- Solder pin headers
- Find pin headers connections
- Search info on the Internet
- Use Gimp to follow traces on 2 layers PCB
- Use a similar projects as hint
- Use SOCs data sheets as hint
- **Confirm connection with a multimeter**





## J9 (I2C Bus)

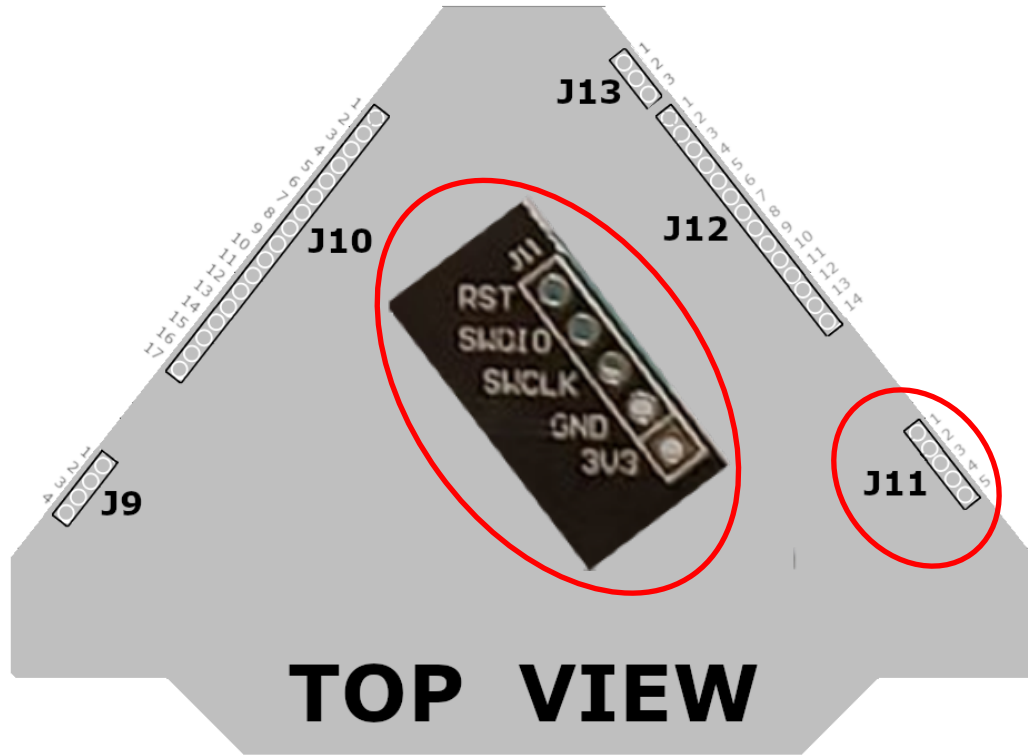
PIN	Name	ESP32	SAMD21
1	VCC		
2	SCL	36 – IO22	14 – PA09
3	SDA	37 – IO23	13 – PA08
4	GND		



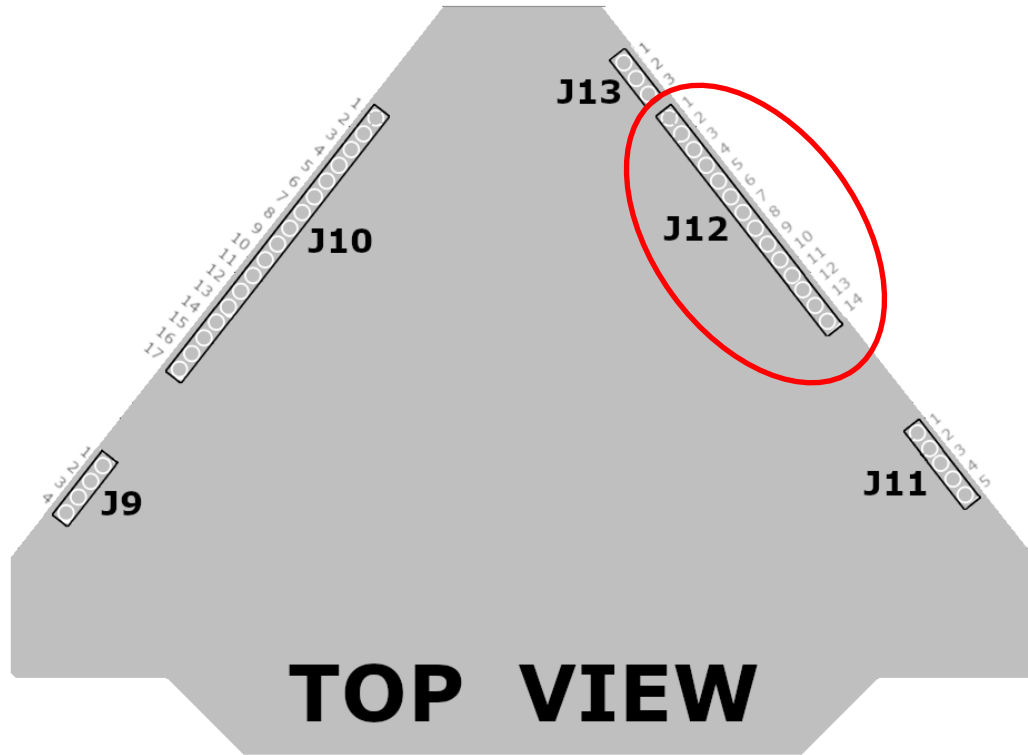
## J10 (to ESP32)

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		

## J11 (SAM21 programming interface)

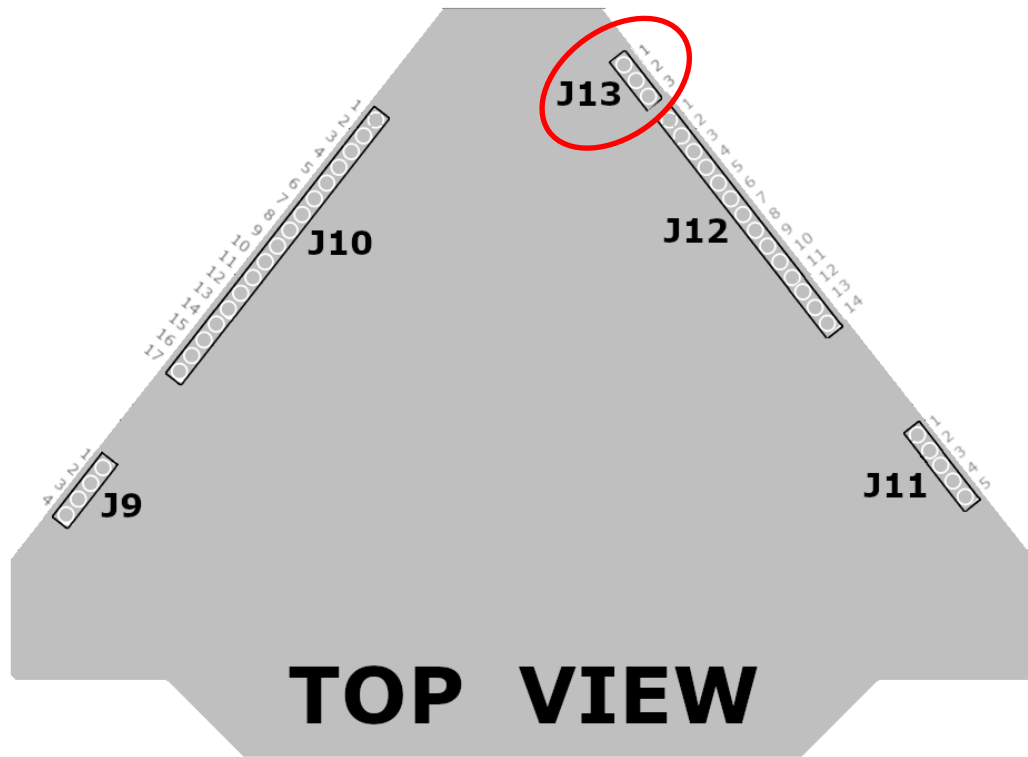


PIN	SAMD21
1	40 - nRST
2	46 - SWDIO
3	45 - SWCLK
4	GND
5	VCC



## J12 (to ESP32)

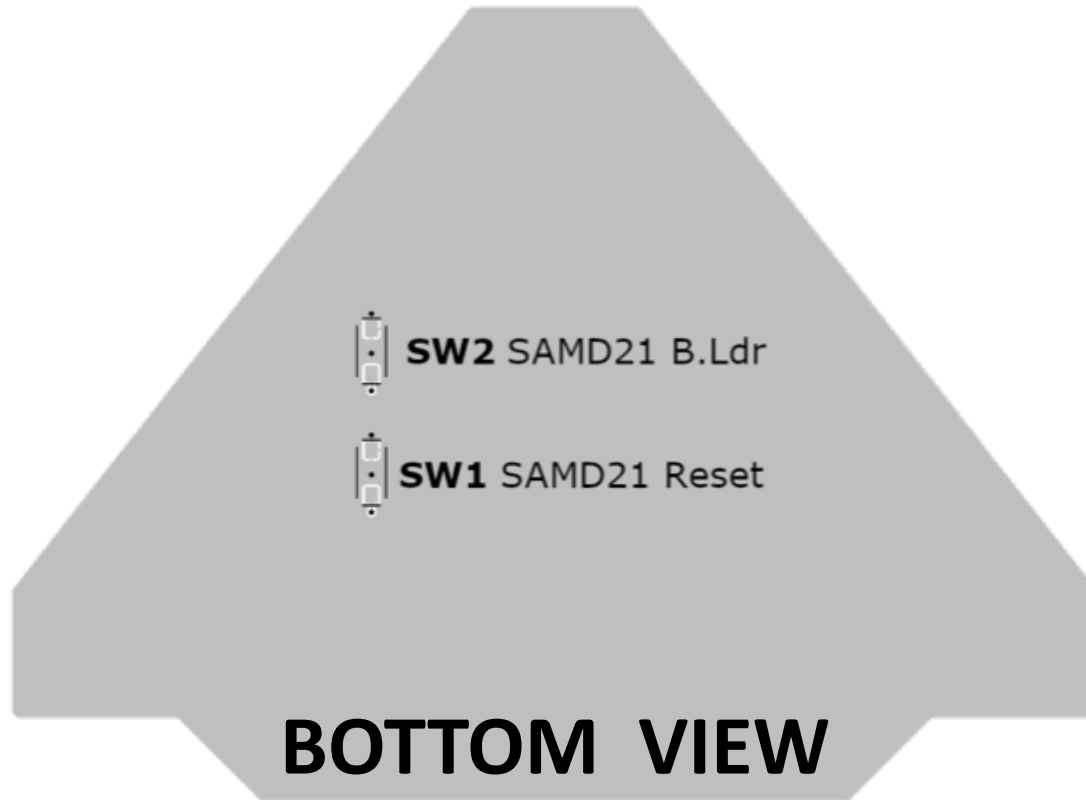
PIN	ESP32	PIN	ESP32
1	GND	8	9 – IO33
2	25 – IO0 (B.LDR)	9	11 – IO26
3	4 – IO36	10	12 – IO27
4	5 – IO39	11	13 – IO14
5	6 – IO34	12	14 – IO12
6	7 – IO35	13	16 – IO13
7	8 – IO32	14	VCC



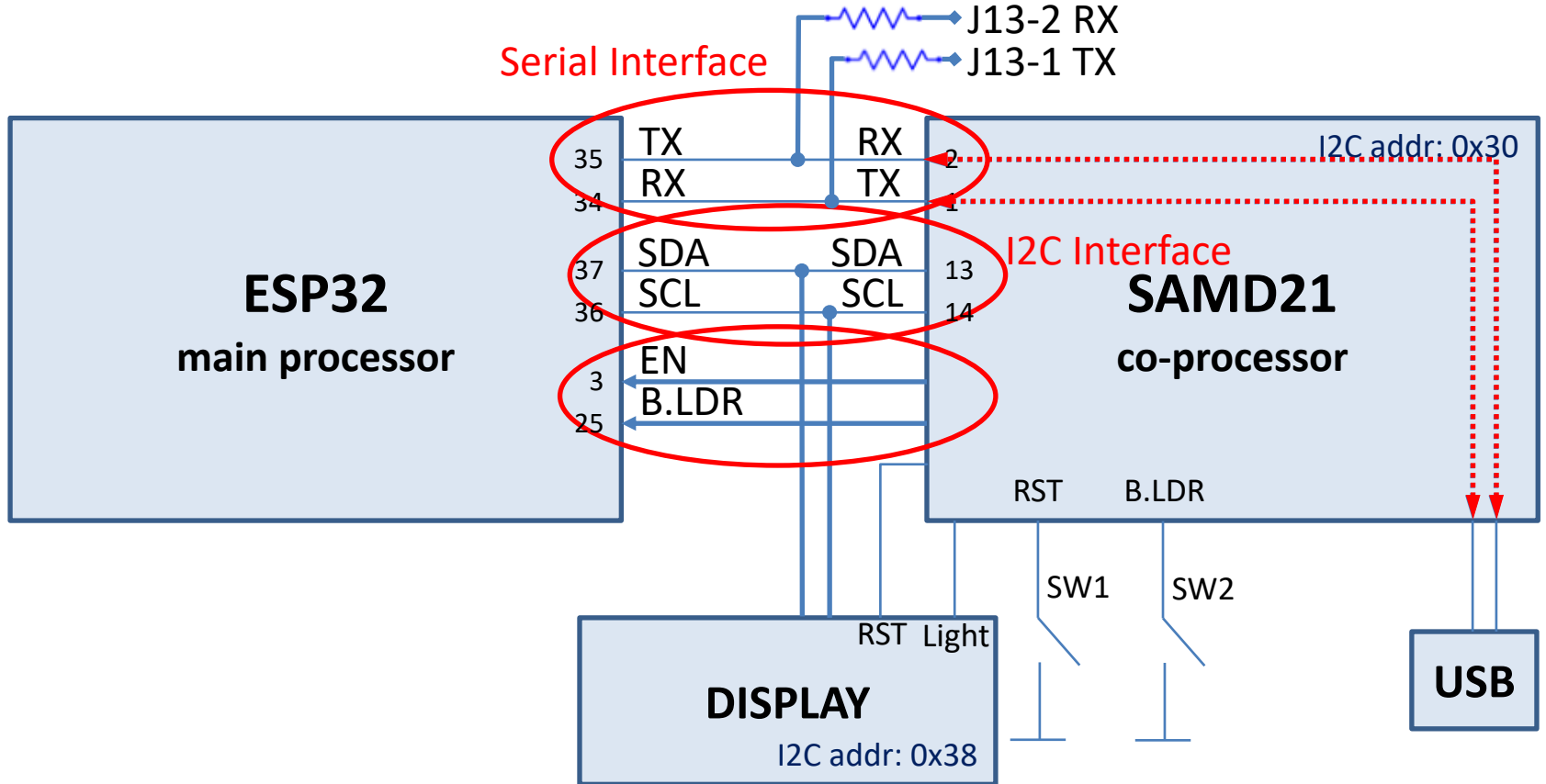
## J13 (ESP32 serial interface)

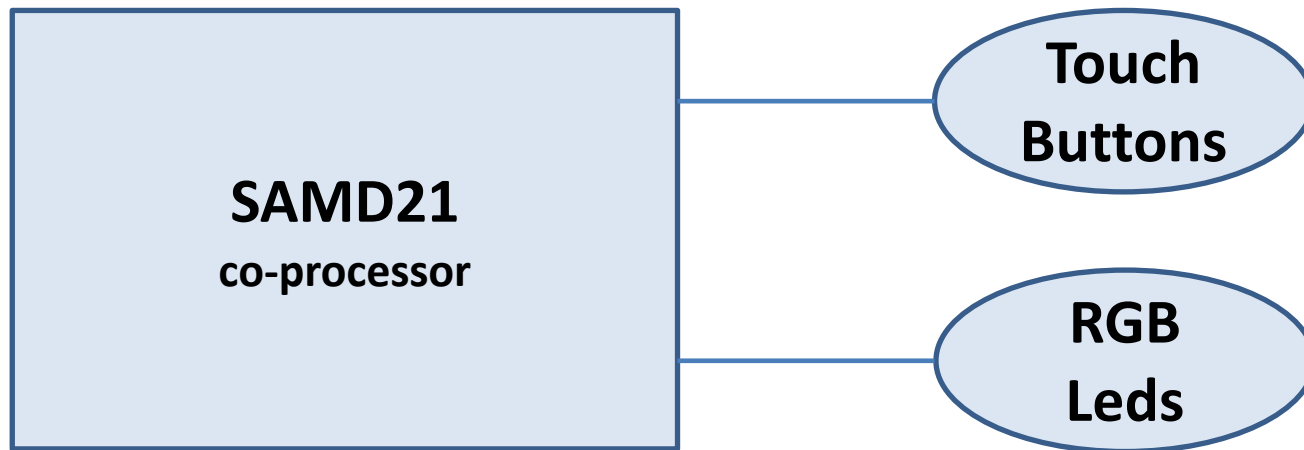
PIN	NAME	ESP32	SAMD21
1	TX	35 (330 $\Omega$ )	2 - RX
2	RX	34 (330 $\Omega$ )	1 - TX
3	GND		



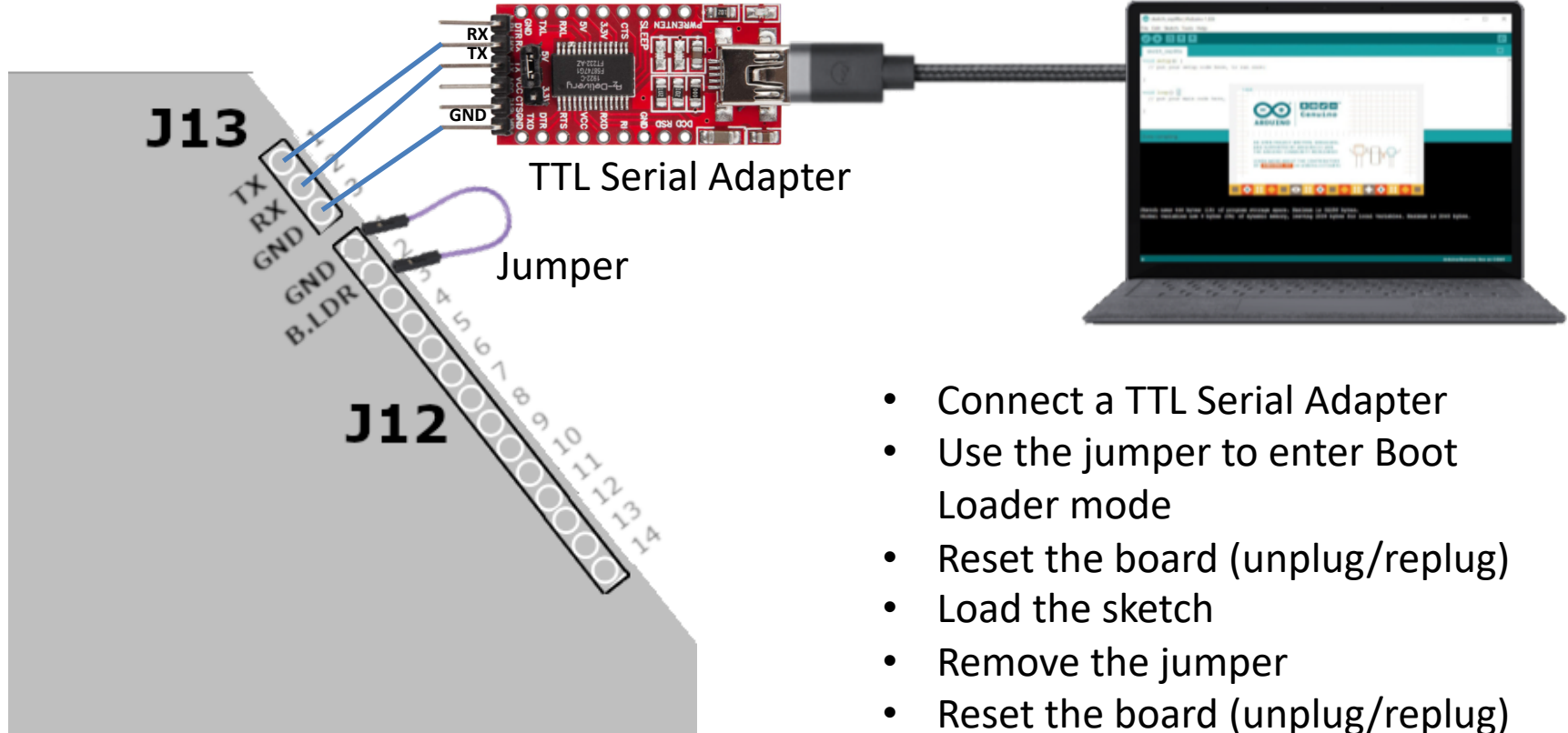


# ESP32 – SAMD21





# ESP32 with Arduino IDE

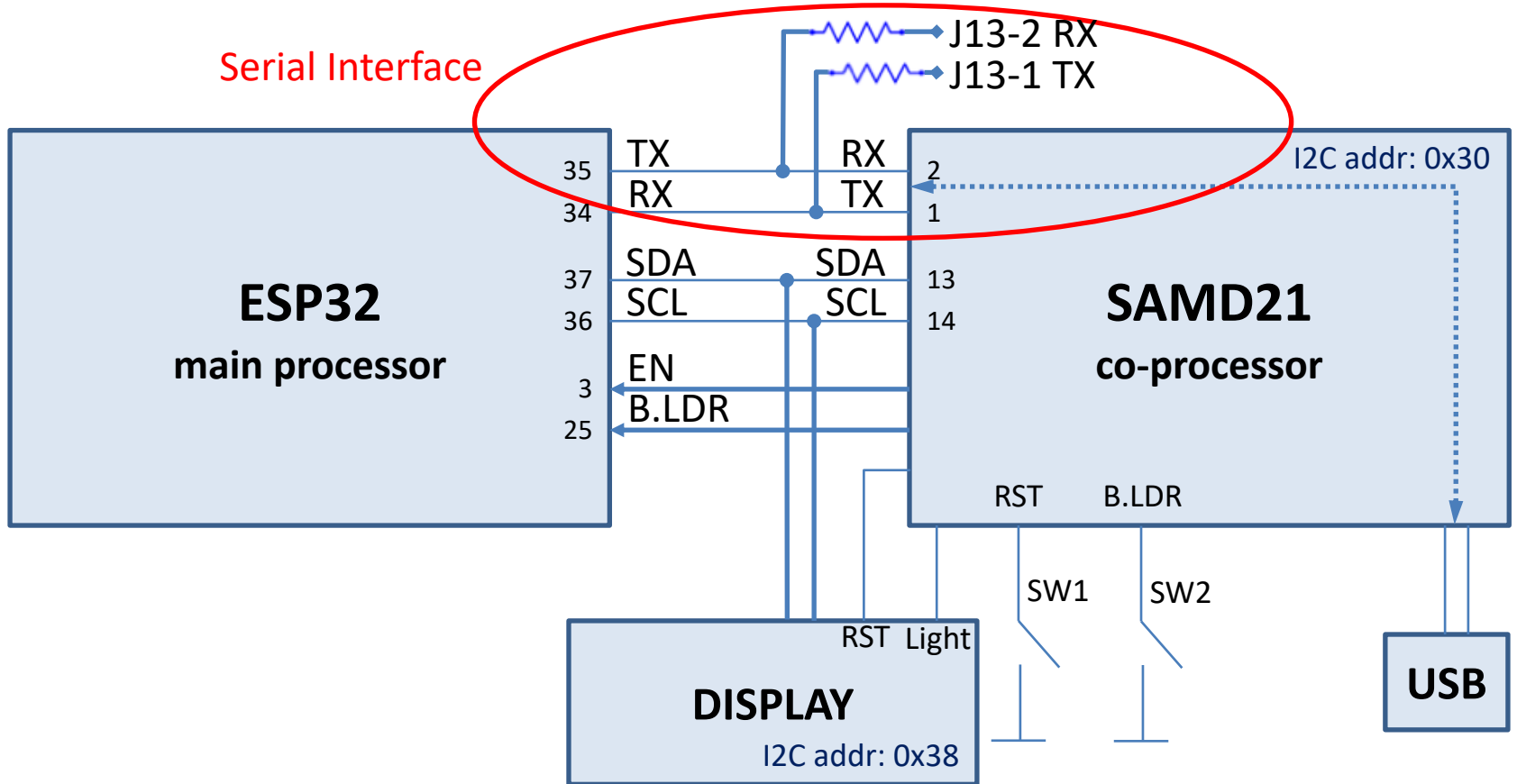


- Connect a TTL Serial Adapter
- Use the jumper to enter Boot Loader mode
- Reset the board (unplug/replug)
- Load the sketch
- Remove the jumper
- Reset the board (unplug/replug)

# Hello World with a LED

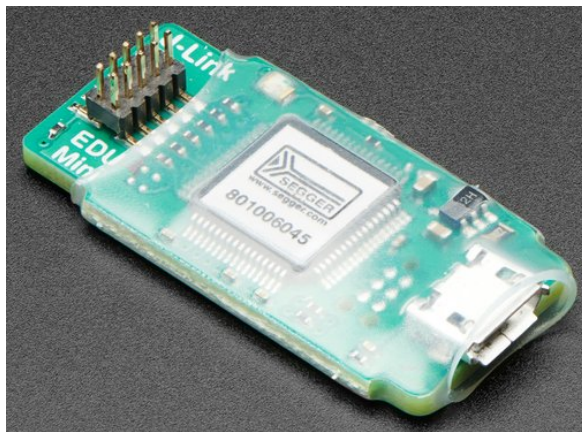
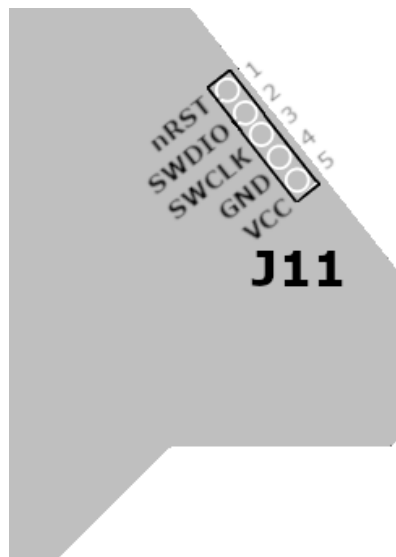
*Video loading the blinking  
LED on the ESP32*

# Hello World with a LED



- 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

- 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

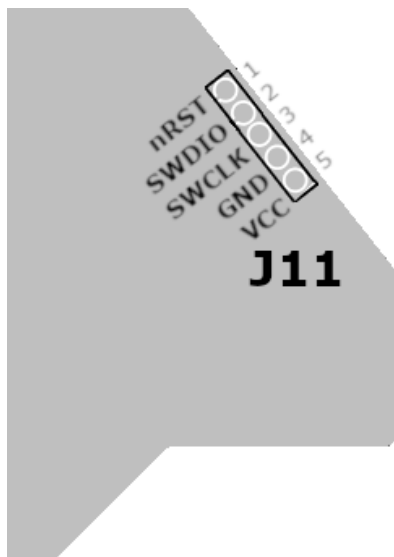
PRODUCT ID: 3571

**\$19.95**

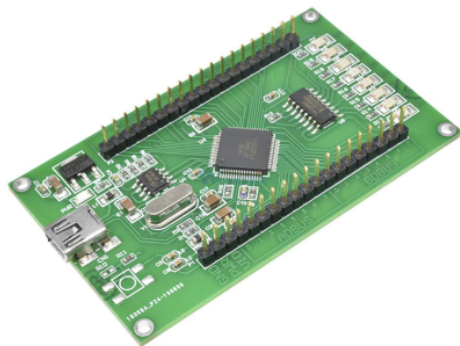


# Re-flashing Hardware

- 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



World Chips



FT2232HL Development Board Learning Board FT2232H MINI FT4232H UM232H Development Board Module USB to SPI Dual Serial Port

★★★★★ 5.0 9 Reviews 16 orders

**US \$9.77** ~~US \$11.50~~ -15%

Instant discount: US \$1.00 off per US \$39.00

US \$1.00 off on US \$20.00 [Get coupons](#)

Quantity:

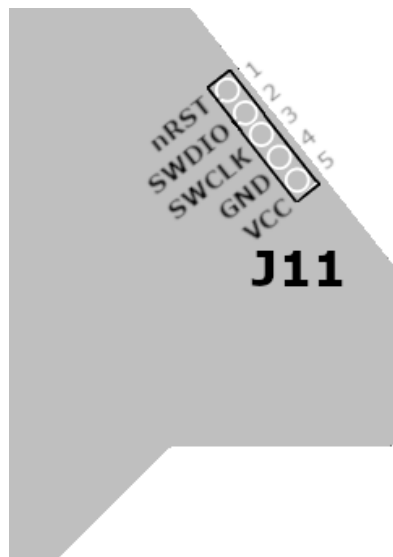
1 Additional 1% off (2 pieces or more)  
2944 pieces available

**Shipping: US \$4.65**

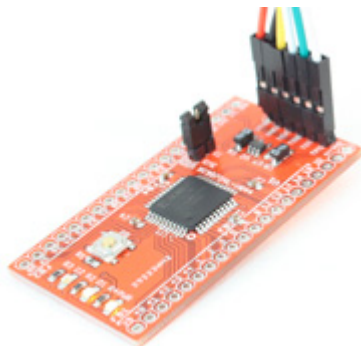
to Italy via AliExpress Standard Shipping

Estimated Delivery on 10/07

# Re-flashing Hardware

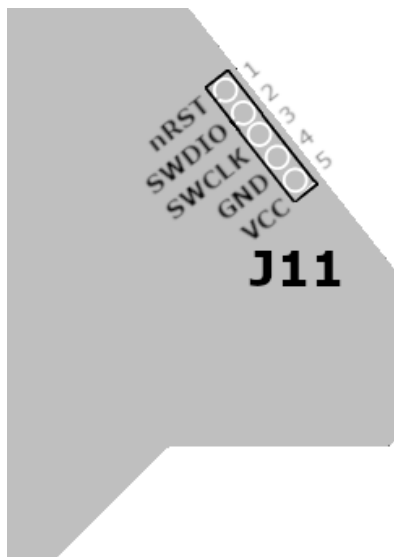


- On the badge we have the SWD (Serial Wire Debug) interface headers
- We can use:
  - Segger J-Link Edu Mini, very good proprietary software
  - Breakout boards with FT2232H and OpenOCD software
  - Bus Bluster (based on FT2232H) and OpenOCD software



Project Summary	
Name:	Bus Blaster
Buy it:	<a href="#">Get one for \$34.95 at Seeed Studio</a>
Price:	<b>\$34.95</b>
Status:	Mature
Manufacturing:	Shipping
Forum:	<a href="#">Bus Blaster Forum</a>

**\$ 34.95**



- 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

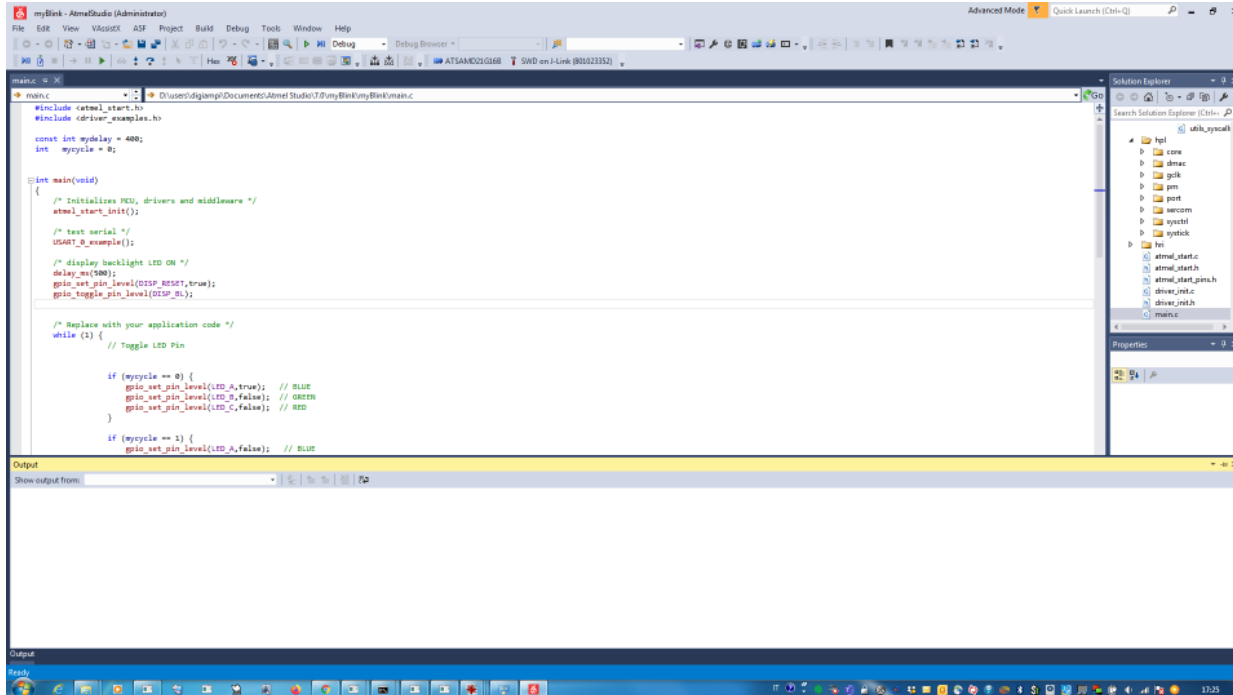
Qty:

1

















In stock ■ ■ ■ ■



- Windows Only



- 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
 <b>Atmel Studio 7.0</b>	<b>Atmel</b>	<b>26/08/2020</b>	<b>4,12 GB</b>	<b>7.0.2397</b>

- Windows Only
- Over-bloated
- GUI for everything

Atmel | START | ATSAMD21G16B | What's New | Help

### PINMUX CONFIGURATOR

# ↑	Pad	User label	Header	Label	Mode	SW co...
PORT						
13	PA08	PIN_SDA			Digital o...	P/13
14	PA09	PIN_SCL			Digital o...	P/14
15	PA10	DISP_RESET			Digital o...	P/15
16	PA11	DISP_BL			Digital o...	P/16
21	PA12	LED_A			Digital o...	P/21
25	PA16	LED_B			Digital o...	P/25
26	PA17	LED_C			Digital o...	P/26
27	PA18	LED_D			Digital o...	P/27
28	PA19	LED_E			Digital o...	P/28
29	PA20	LED_F			Digital o...	P/29
30	PA21	LED_G			Digital o...	P/30

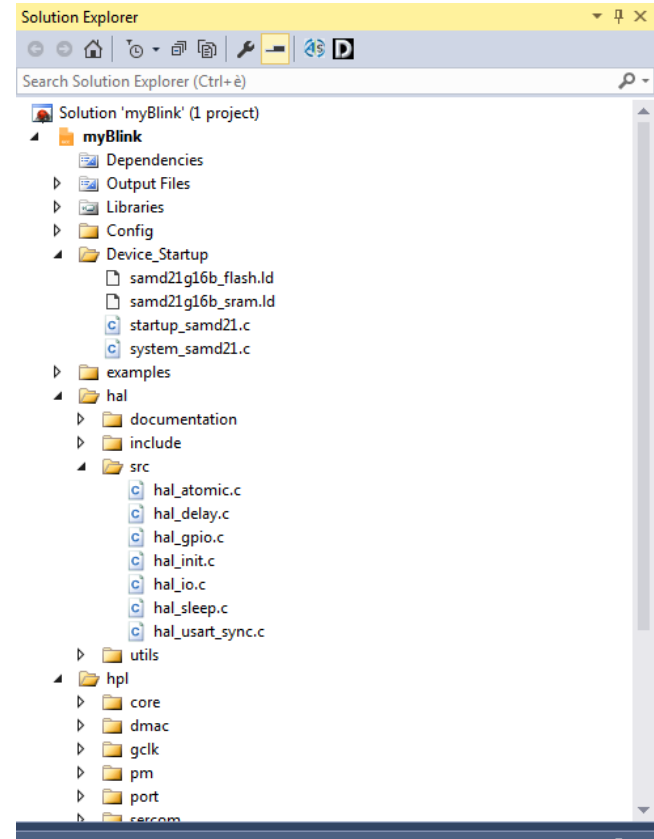
Microchip  
ATSAMD21G16B  
66 KB + 8 KB  
TQFP

Pin 21 (PA12) is used as P/21 with PORT.  
Tip: Use ctrl or shift to select more than one pin.

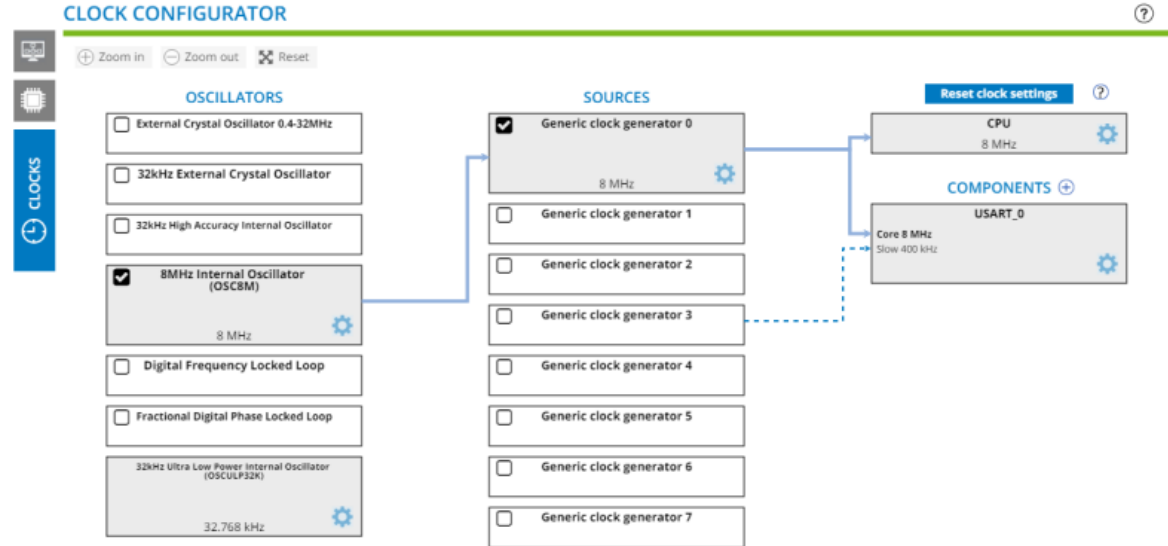
User label:  Initial level:

Pin mode:

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

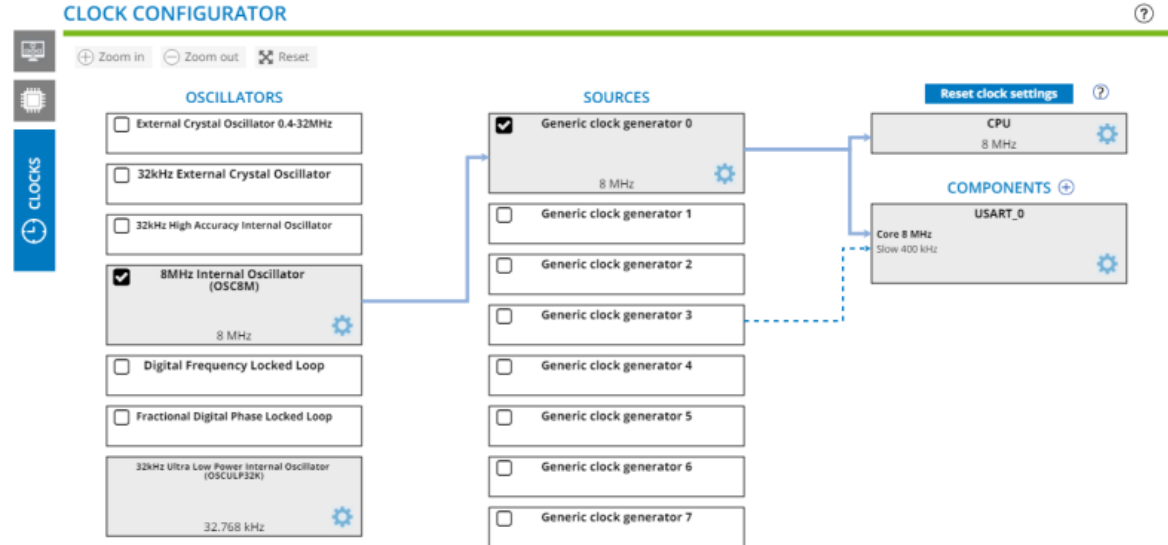


- Windows Only
- Over-bloated
- GUI for everything
- Over-bloated ASF framework
- GUI allows illegal configurations

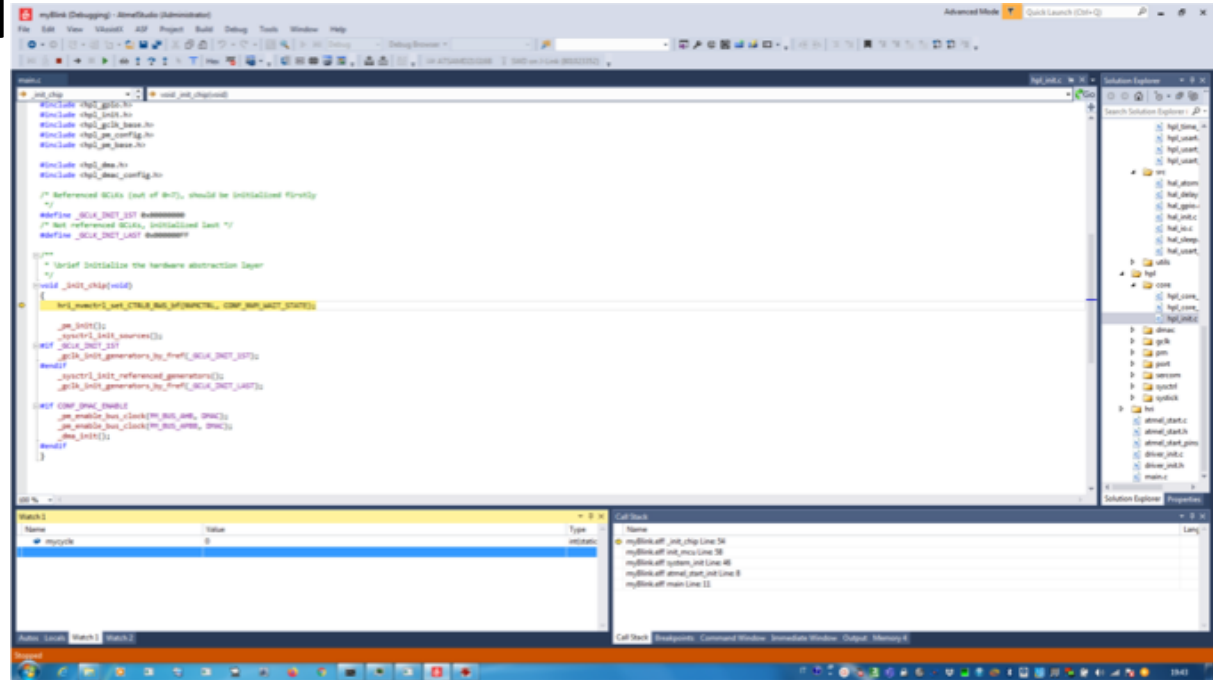




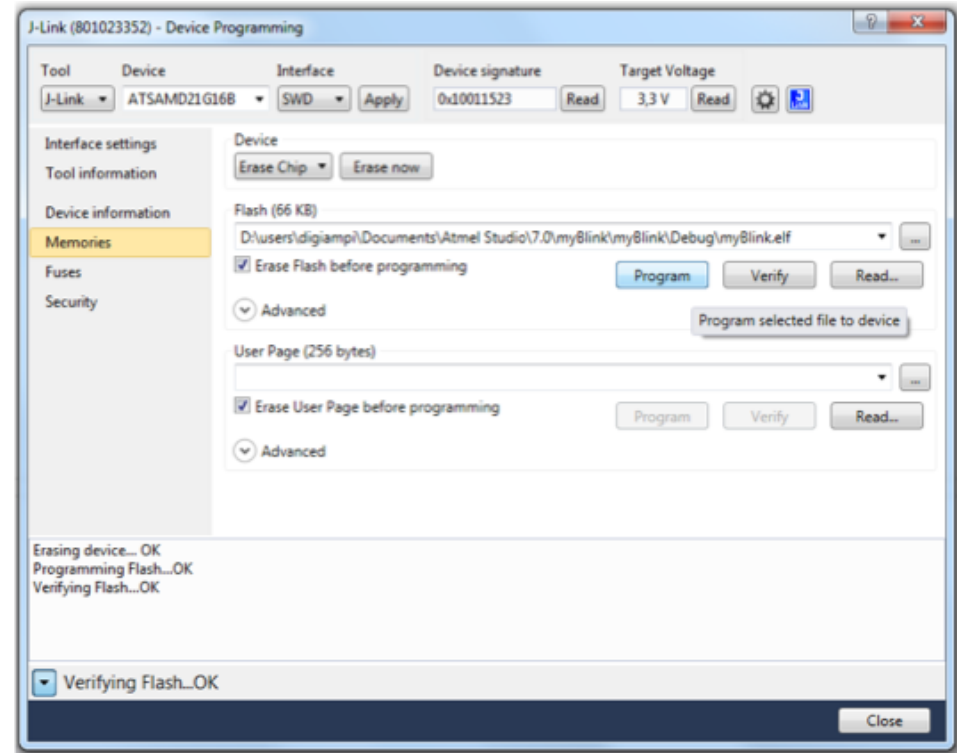
- Windows Only
- Over-bloated
- GUI for everything
- Over-bloated ASF framework
- GUI allows illegal configurations
- Complex clocks configuration



- Visual Studio is a good and pleasant GUI

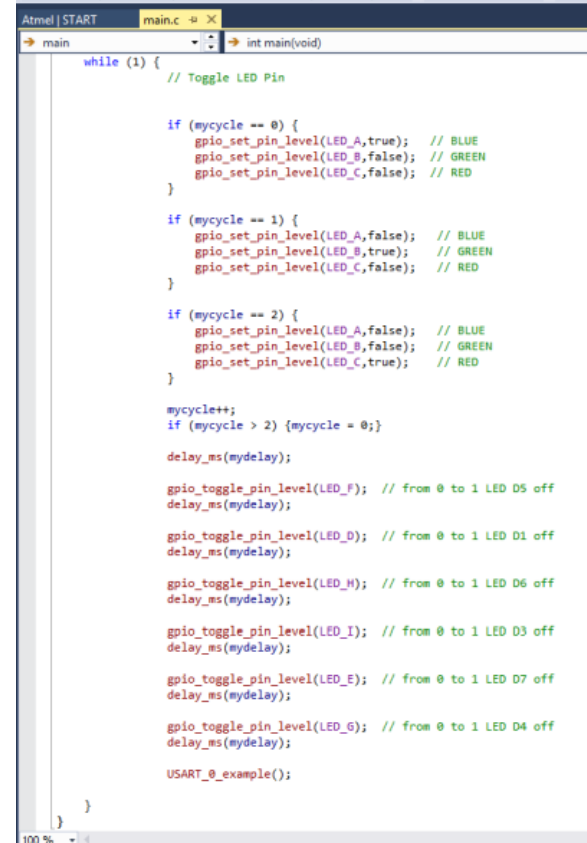


- Visual Studio is a good and pleasant GUI
- Good integration with J-Link probe to flash the device





- Confirm LED position with a «Hello World» program



```
Atmel | START | main.c | x
main | int main(void)
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();
}
}
```

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

Video loading the blinking  
LED on the ESP32

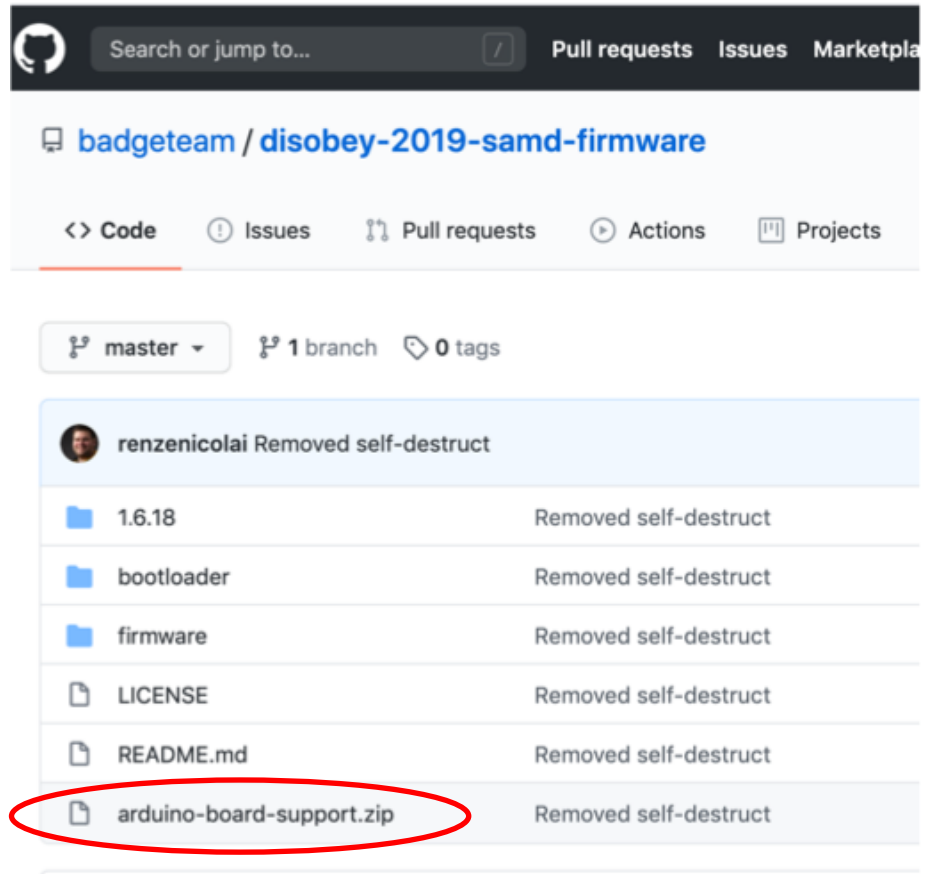
- Spent few weeks trying to program a USB serial to ESP32 serial bridge
- Difficult to find mix and merge available examples
- Writing something from scratch seems impossible without studying the 1000+ pages SAMD21 data sheet and the 500+ pages Atmel Studio manual!
- The "Atmel Studio" way is the opposite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



*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*

- On GitHub the SAMD21G16 was already ported to Arduino IDE



The screenshot shows the GitHub interface for the repository 'badgeteam / disobey-2019-samd-firmware'. The repository is on the 'master' branch and has 1 branch and 0 tags. A commit by 'renzenicolai' is shown, with the message 'Removed self-destruct'. The commit details list several files and folders that have been removed, including '1.6.18', 'bootloader', 'firmware', 'LICENSE', 'README.md', and 'arduino-board-support.zip'. The file 'arduino-board-support.zip' is circled in red.

File/Folder	Action
1.6.18	Removed self-destruct
bootloader	Removed self-destruct
firmware	Removed self-destruct
LICENSE	Removed self-destruct
README.md	Removed self-destruct
arduino-board-support.zip	Removed self-destruct

# SAMD21 Arduino IDE

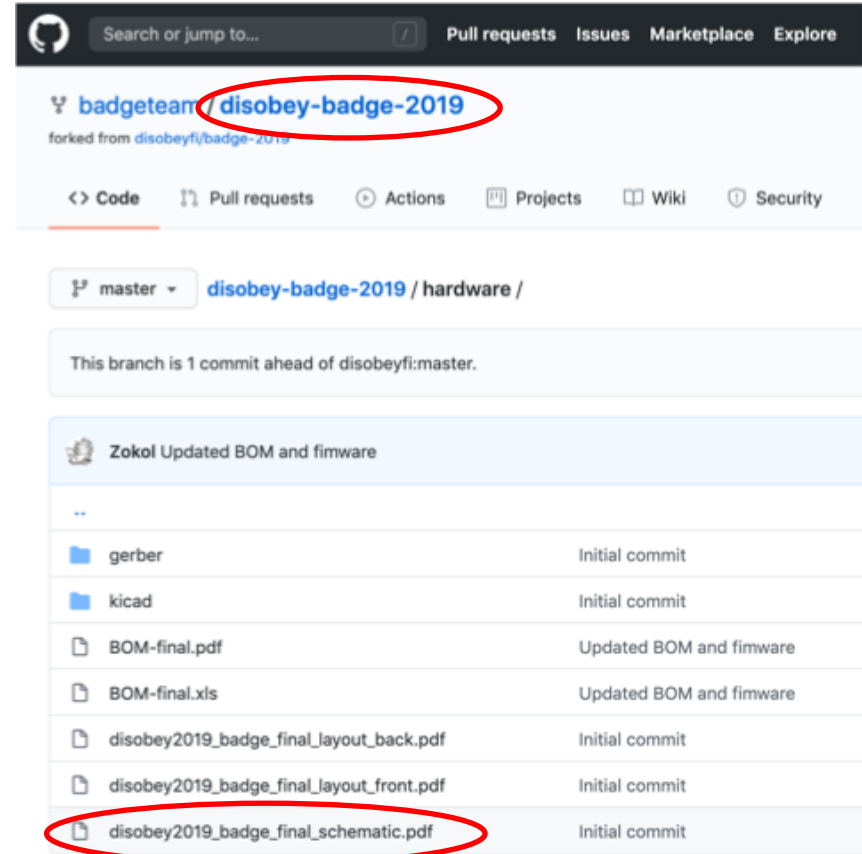
- On GitHub the SAMD21G16 was already ported to Arduino IDE
- The project has the firmware for the “Disobey 2019” badge, very similar to our badge

The screenshot shows the GitHub interface for the repository 'disobey-2019-samd-firmware' by 'budgeteam'. The repository name is circled in red. Below the repository name, there are navigation tabs for 'Code', 'Issues', 'Pull requests', 'Actions', and 'Projects'. The 'Code' tab is selected. Below the tabs, there is a dropdown menu for 'master' with '1 branch' and '0 tags'. The main content area shows a commit by 'renzenicolai' with the message 'Removed self-destruct'. Below the commit, there is a list of files and folders, each with a 'Removed self-destruct' status:

1.6.18	Removed self-destruct
bootloader	Removed self-destruct
firmware	Removed self-destruct
LICENSE	Removed self-destruct
README.md	Removed self-destruct
arduino-board-support.zip	Removed self-destruct

# 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

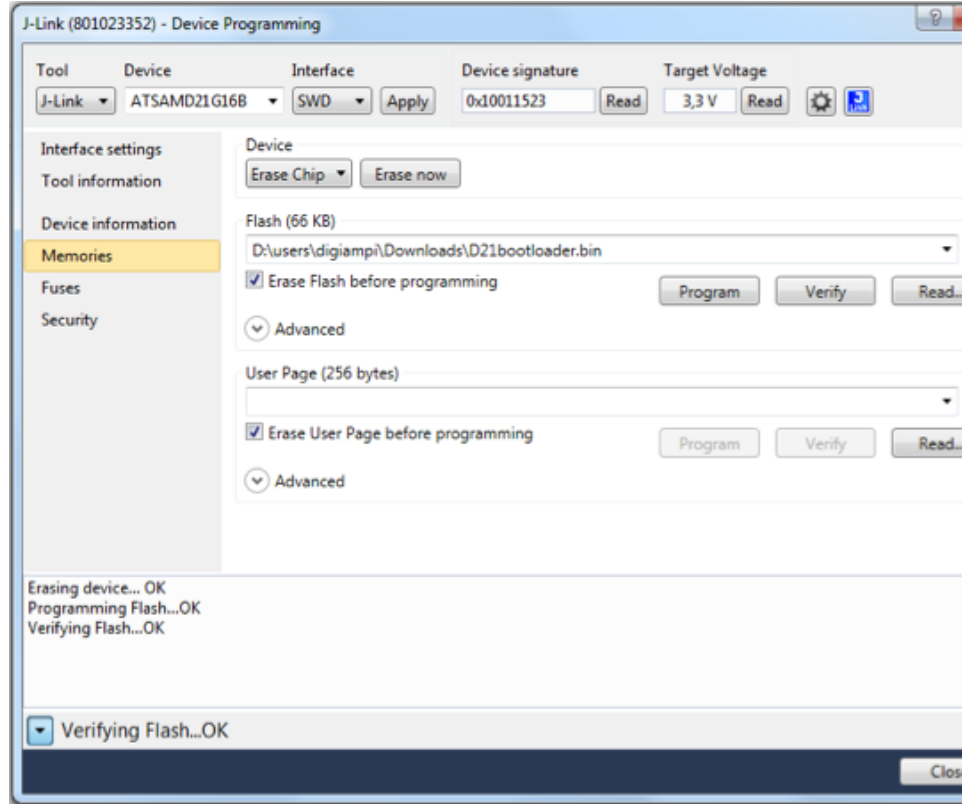


- In the Arduino IDE the USB Serial to ESP32 serial bridge is only 10 lines of code!

```
272 inline void update_serial()
273 {
274     while (Serial.available()) {
275         Serial1.write(Serial.read());
276     }
277
278     while (Serial1.available()) {
279         Serial.write(Serial1.read());
280     }
281 }
```

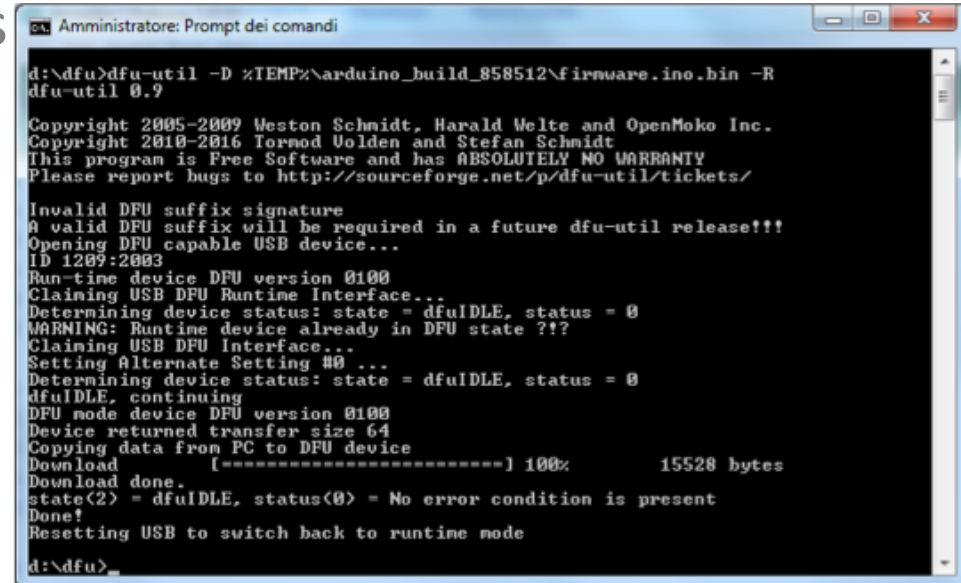
# USB Serial to ESP2 serial bridge

- 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



```
Amministratore: Prompt dei comandi
d:\dfu>dfu-util -D %TEMP%\arduino_build_858512\firmware.ino.bin -R
dfu-util 0.9

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

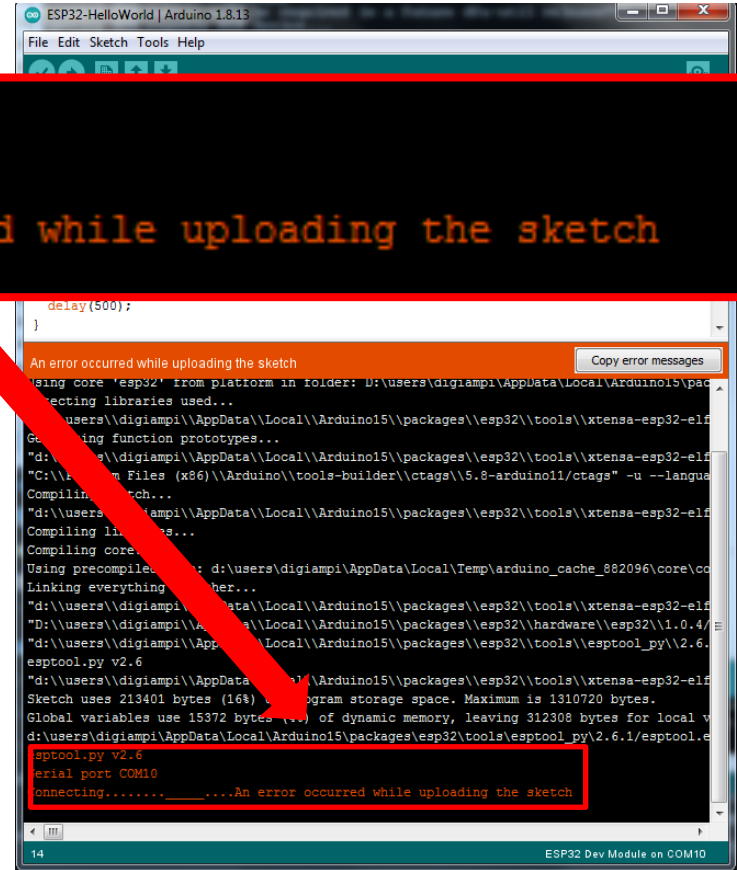
Invalid DFU suffix signature
A valid DFU suffix will be required in a future dfu-util release!!!
Opening DFU capable USB device...
ID 1209:2003
Run-time device DFU version 0100
Claiming USB DFU Runtime Interface...
Determining device status: state = dfuIDLE, status = 0
WARNING: Runtime device already in DFU state ???
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 0100
Device returned transfer size 64
Copying data from PC to DFU device
Download [=====] 100% 15528 bytes
Download done.
state(2) = dfuIDLE, status(0) = No error condition is present
Done!
Resetting USB to switch back to runtime mode
d:\dfu>
```

# USB Serial to ESP2 serial bridge

- 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!



```
ESP32>HelloWorld | Arduino 1.8.13  
File Edit Sketch Tools Help  
delay(500);  
}  
An error occurred while uploading the sketch  
Using core 'esp32' from platform in folder: D:\users\digiampi\AppData\Local\Arduino15\pac  
Selecting libraries used...  
Using library 'ESP8266' at location 'D:\users\digiampi\AppData\Local\Arduino15\packages\esp32\tools\xtensa-esp32-elf  
Generating function prototypes...  
d:\users\digiampi\AppData\Local\Arduino15\packages\esp32\tools\xtensa-esp32-elf  
C:\Program Files (x86)\Arduino\tools-builder\ctags\5.8-arduino11\ctags" -u --language  
Compiling sketch...  
d:\users\digiampi\AppData\Local\Arduino15\packages\esp32\tools\xtensa-esp32-elf  
Compiling libraries...  
Compiling core...  
Using precompiled headers: d:\users\digiampi\AppData\Local\Temp\arduino_cache_882096\core\co  
Linking everything together...  
d:\users\digiampi\AppData\Local\Arduino15\packages\esp32\tools\xtensa-esp32-elf  
D:\users\digiampi\AppData\Local\Arduino15\packages\esp32\hardware\esp32\1.0.4  
d:\users\digiampi\AppData\Local\Arduino15\packages\esp32\tools\esptool_py\2.6  
esptool.py v2.6  
d:\users\digiampi\AppData\Local\Arduino15\packages\esp32\tools\xtensa-esp32-elf  
Sketch uses 213401 bytes (16%) of program storage space. Maximum is 1310720 bytes.  
Global variables use 15372 bytes (4%) of dynamic memory, leaving 312308 bytes for local v  
d:\users\digiampi\AppData\Local\Arduino15\packages\esp32\tools\esptool_py\2.6.1/esptool.e  
esptool.py v2.6  
Serial port COM10  
Connecting.....An error occurred while uploading the sketch
```



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

```
272 inline void update_serial()
273 {
274     while (Serial.available()) {
275         Serial1.write(Serial.read());
276     }
277
278     while (Serial1.available()) {
279         Serial.write(Serial1.read());
280     }
281 }
```

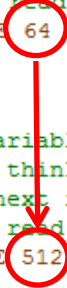
- The problem was that the serial bridge was not fast enough
- Arduino IDE hides complexity, but can generate code that can be very inefficient

```
272  inline void update_serial()
273  {
274      while (Serial.available()) {
275          Serial1.write(Serial.read());
276      }
277
278      while (Serial1.available()) {
279          Serial.write(Serial1.read());
280      }
281  }
```

- 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
```



- 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
- 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

- Success in using Arduino IDE to program the ESP32 on the badge
- Still to do: modifying an existing, SPI based, library to 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

# Thank You

# Question Time

Valerio Di Giampietro

<http://va.ler.io>

v@ler.io

@valerio

[youtube.com/makemehack](https://youtube.com/makemehack)