

uDEV Soluções Tecnológicas

Multi-Sensor Sigfox Embedded Development Kit (EDK)

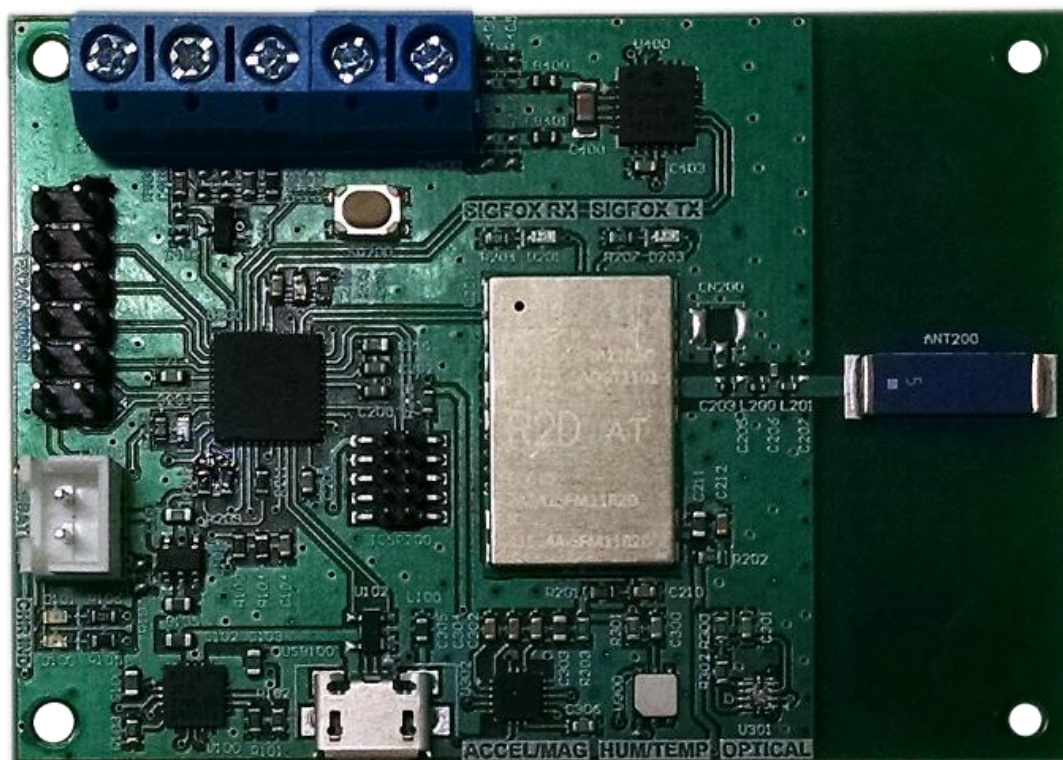


Figure 1. uDEV SIGFOX EDK model

Table of Contents

Revision History.....	3
1. Introduction.....	4
2. Features.....	5
3. Hardware.....	6
3.1. General.....	7
3.2. Sensors.....	7
3.3. Power.....	10
4. Memory Addressing.....	11
5. Installing SAM-BA and the EDK board files.....	16
6. Connecting your EDK.....	17
7. Using SAM-BA.....	19
7. Testing.....	22
7.1. LED Project Template - Installing.....	22
7.2 Generic Project Template.....	31

Revision History

Author	Description	Date
Lucas Brugnaro Badur	First version	2017-11-24

1. Introduction

The SIGFOX Embedded Development Kit (EDK) is a multi-sensor board, designed to be employed by developers aiming to use SIGFOX as a means of easy data transmission, especially for Internet of Things (IoT) applications.

Using the on-board sensors in the EDK allows for the user or developer to easily obtain a plethora of data without the need for additional devices or external sensors, and send this data to the cloud seamlessly via the SIGFOX modem. A great deal of applications are possible, such as monitoring different environment variables and signaling any desired changes.

2. Features

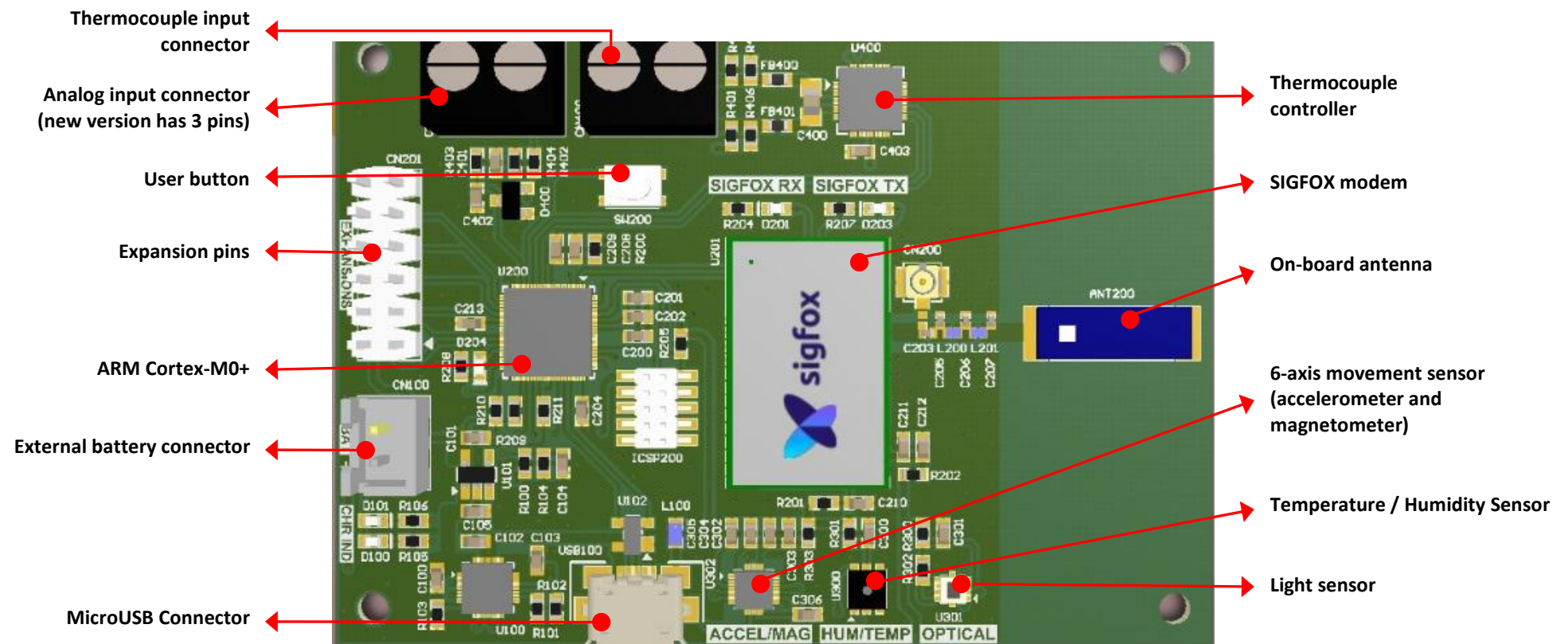
The EDK contains the following features:

- High-performance ARM Cortex-M0+ ATSAMD21G18A-M
- 6-axis motion sensor
- Temperature and humidity sensor
- Ambient light sensor
- Input-ready for external thermocouples
- Connectors for analog input
- Integrated Real-Time Clock
- On-board SIGFOX modem
- On-board Antenna
- Micro USB connection
- External battery connector, with Integrated Battery Charge Management

3. Hardware

The EDK is fully designed at and by uDEV, and allows developers to create quick solutions for many IoT-related systems.

The kit is centered around the ARM Cortex-M0+ ATSAMD21G18A-M processor, and has a ready-to-use SIGFOX modem, complete with on-board antenna for readily available communication with the SIGFOX network.



3.1. General

Main Microprocessor:

- **ARM Cortex-M0+ ATSAMD21G18A-M**
 - ◆ The most energy-efficient Cortex processor to date
 - ◆ Integrated power management with many low-power functions
 - ◆ More information and full specification page available at <https://developer.arm.com/products/processors/cortex-m/cortex-m0-plus>

3.2. Sensors

Humidity and Temperature Sensor:

- **Silicon Labs Si7006-A20**
- **Precision Relative Humidity Sensor:**
 - ◆ 0-90% Relative Humidity (RH), $\pm 5\%$ RH (max.)
 - ◆ Operating Range: 0 to 100% RH
- **High Accuracy Temperature Sensor:**
 - ◆ -10 to +85 °C, ± 1 °C (max.)
 - ◆ Operating Temperature: -40 to +125 °C
- **Low power consumption**
 - ◆ 150 μ A active current
 - ◆ 60 nA standby current
- **Factory-calibrated**
- **Full datasheet available at:**
<https://www.silabs.com/documents/public/data-sheets/Si7006-A20.pdf>

Light Sensor:

- **Silicon Labs Si1133**
- **High accuracy UV index sensor**
- **Ambient Light Sensor**
 - ◆ Resolution of under 100 mlx possible
- **Operating Temperature:** -40 to +85 °C
- **Low power consumption**
 - ◆ 500 nA standby current (max.)
- **Full datasheet available at:**
<https://www.silabs.com/documents/public/data-sheets/Si1133.pdf>

Thermocouple input:

- **Microchip MCP9600**
- **Supports Thermocouple types K, J, T, N, S, E, B, R**
- **Integrated Cold-Junction Compensation**
- **Four Programmable Temperature Alert Outputs**
- **I2C Compatible**
- **Low Power:**
 - ◆ Shutdown mode: 2 µA typical current
 - ◆ Burst Mode: 1 to 128 Temperature Samples
- **Full datasheet available at:**
<http://ww1.microchip.com/downloads/en/DeviceDoc/20005426C.pdf>

One-Wire Thermometer:

- **Maxim Integrated DS18B20**
- **Temperature Sensor:**
 - ◆ -55 to +125 °C
 - ◆ Accuracy of ± 0.5 °C from -10 to +85°C
 - ◆ Operating / Storage Temperature Range: -55 to +125 °C
- **Programmable resolution**
- **Full datasheet available at:**
<https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>

6-axis sensor:

- **Digikey FXOS8700CQ**
- **Integrated linear 3-axis accelerometer**
- **Integrated 3-axis magnetometer**
- **Complete e-compass hardware solution**
- **Embedded programmable acceleration event functions**
 - ◆ Freefall and motion detection
 - ◆ Transient detection
 - ◆ Vector-magnitude change detection
 - ◆ Pulse and tap detection (single and double)
 - ◆ Orientation detection (portrait / landscape)
- **Embedded programmable magnetic event functions**
 - ◆ Threshold detection
 - ◆ Vector-magnitude change detection
 - ◆ Autonomous magnetic min/max detection
 - ◆ Autonomous hard-iron calibration
- **Low power consumption**
 - ◆ 240 μ A current consumption at 100 Hz
 - ◆ 80 μ A current consumption at 25 Hz with both sensors active
- **Full datasheet available at:**
<https://media.digikey.com/pdf/Data%20Sheets/NXP%20PDFs/FXOS8700CQ.pdf>

3.3. Power

Stand-Alone System Load Sharing and Li-Ion / Li-Polymer Battery Charge Management Controller:

- **Integrated System Load Sharing and Battery Charge Management**
 - Simultaneously power the system and charge the battery
 - Voltage Proportional Current Control (VPCC) ensures system load has priority over battery charge current
- **Complete Linear Charge Management Controller**
 - Integrated Pass Transistors
 - Integrated Current Sense
 - Integrated Reverse Discharge Protection
 - Selectable Input power sources
- **Automatic End-of-Charge control**
- **Refer to the complete datasheet for more information:**
<http://ww1.microchip.com/downloads/en/DeviceDoc/20002090C.pdf>

This controller allows for this circuit to be powered by either an external battery, or directly via the Micro USB port (for debugging purposes).



Any capacity battery can be used, but it **must conform to the following specifications**:

- Type: Lithium-Ion (Li-ion) or Lithium-Ion Polymer (Li-Po)
- Voltage: 3.7 V
- Recharge Current: 100mA
- Connector: JST-XH-A ([DigiKey Part No. 455-2247-ND](#))



Figure: JST-B2B-XH-A Male Connector

4. Memory Addressing

In order for your application to work correctly, it is necessary to set its starting address at 0x6000. This is due to the custom *Bootloader* that is present on the EDK board, which reserves FLASH memory addresses from 0x0000 to 0x6000.

The address must be indicated **both** on the memory definitions within your application's code, and on SAM-BA, before uploading to the board. (See next chapter)

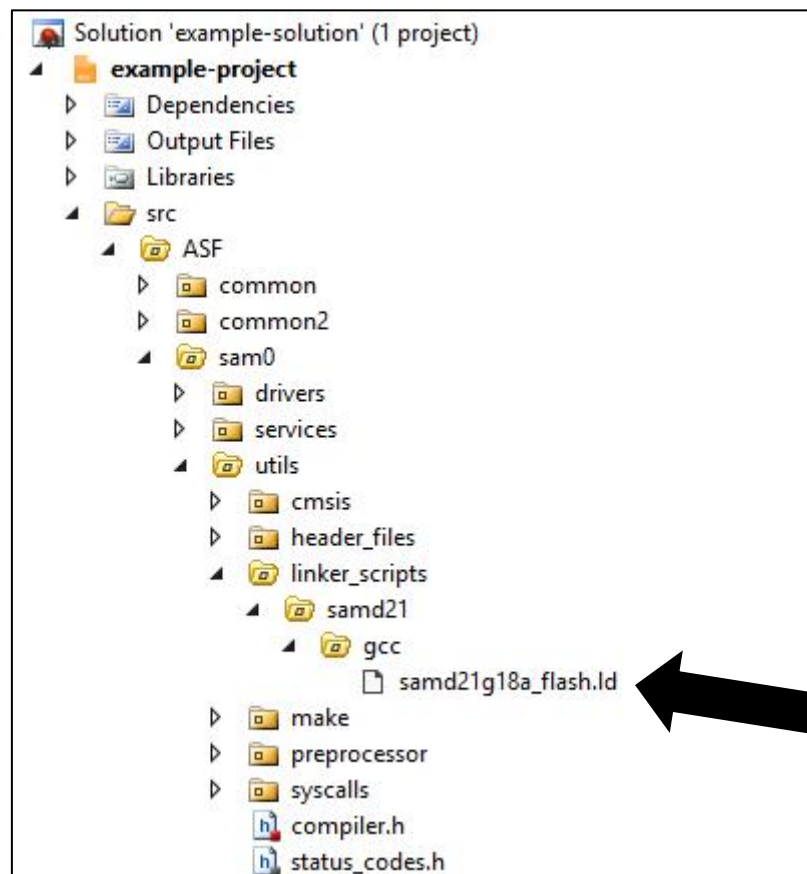
Below is an example on how to do this, building your application using Atmel Studio 7.0 and the SAMD21G18A microprocessor.

There are three methods to set a starting address:

Method A.

You can access the Linker file directly, going to the following path:

Example-solution/example-project/src/ASF/sam0/utils/linker_scripts/samd21/gcc

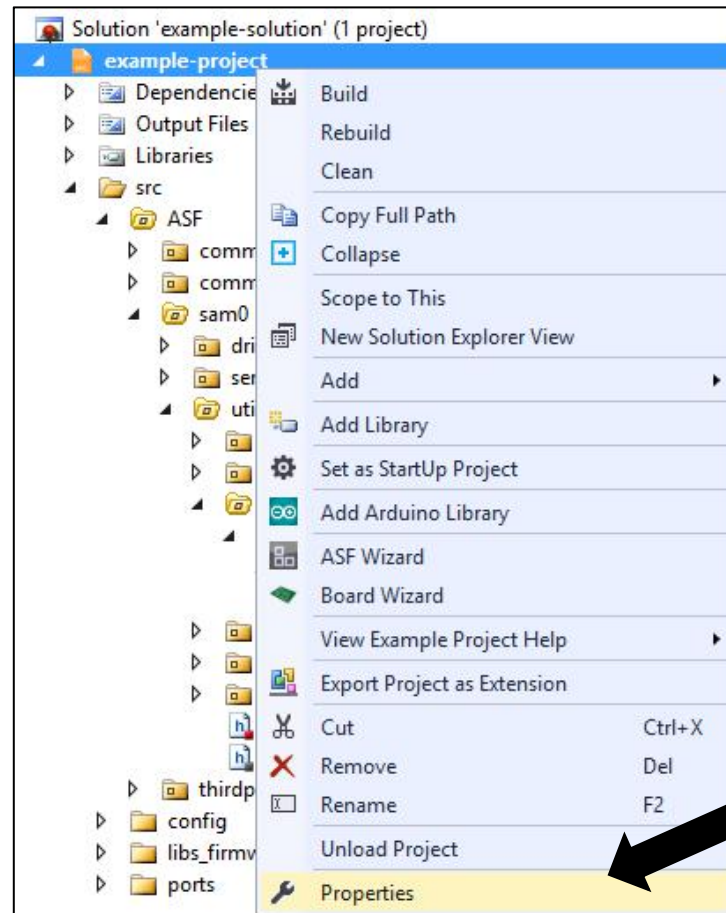


Then, you can edit the ROM ORIGIN attribute, changing the value to **0x00006000**:

```
samd21g18a_flash.ld  X
28  * THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED
29  * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
30  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE
31  * EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR
32  * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
33  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
34  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
35  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
36  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
37  * ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
38  * POSSIBILITY OF SUCH DAMAGE.
39  *
40  * \asf_license_stop
41  *
42  */
43
44
45  OUTPUT_FORMAT("elf32-littlearm", "elf32-littlearm", "elf32-littlearm")
46  OUTPUT_ARCH(arm)
47  SEARCH_DIR(.)
48
49  /* Memory Spaces Definitions */
50  MEMORY
51  {
52      rom      (rx)  : ORIGIN = 0x00006000, LENGTH = 0x0003A000
53      ram      (rwx) : ORIGIN = 0x20000000, LENGTH = 0x00008000
54  }
55
56  /* The stack size used by the application. NOTE: you need to adjust according to your application. */
57  STACK_SIZE = DEFINED(STACK_SIZE) ? STACK_SIZE : DEFINED(__stack_size__) ? __stack_size__ : 0x2000;
58
59  /* Section Definitions */
60  SECTIONS
61  {
62      .text :
63      {
64          . = ALIGN(4);
```

This is the way we recommend setting the Application start address.

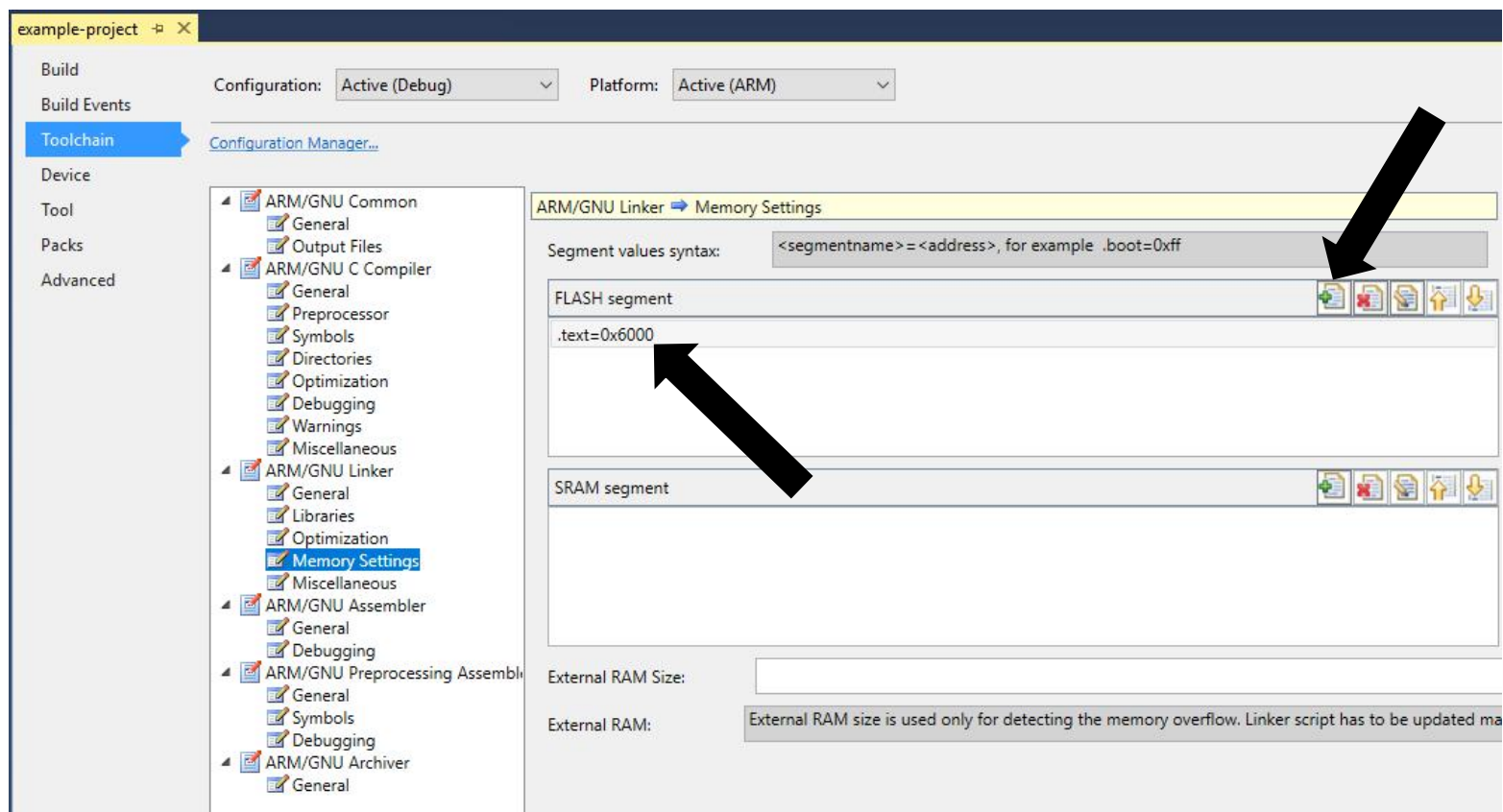
Another way to do this involves adding a command to the Toolchain in the project settings.
Right-click the project, go to **Properties**, and then **Toolchain**.



Then, you have two choices. You can either:

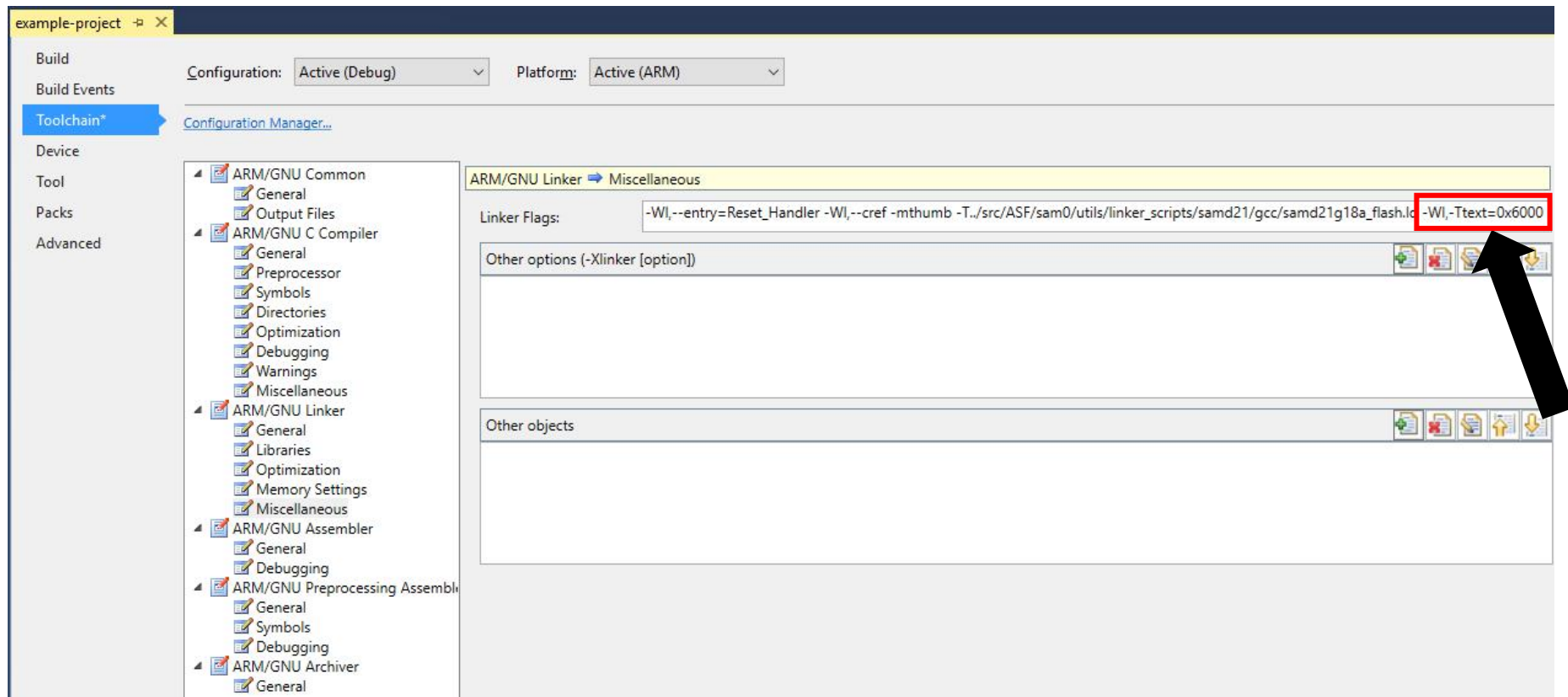
Method B.

- Go to **Memory Settings**, under **ARM/GNU Linker**
 - click “Add Item” under **FLASH segment**
 - write “**.text=0x6000**” and click **OK**



Method C.

- Go to **Miscellaneous**, under **ARM/GNU Linker**
 - Directly on **Linker Flags**, add a space before the last command
 - Write “-Wl,-Ttext=0x6000”















Any of the previous methods will build your project at the correct address, and your program will be ready to upload.

5. Installing SAM-BA and the EDK board files

To write your program on the board, you must use the Microchip SAM Boot Assistance (SAM-BA). The complete installation packages are available at the bottom of the following page, under “Documentation & Software”:

<http://www.microchip.com/DevelopmentTools/ProductDetails.aspx?PartNO=Atmel%20SAM-BA%20I-n-system%20Programmer>

We recommend you download version 2.17 preferably, but any one of the highlighted options below will work, according to your operating system.

Documentation & Software			Back To Top
AppNotes	Last Updated	Size	
★AN_42438 - AT09423: SAM-BA Overview and Customization Process	12/11/2016 10:15:54 AM	6MB	
★AN_42051 - AT03454: SAM-BA for SAM4L	12/11/2016 10:09:14 AM	728KB	
Documents	Last Updated	Size	
★SAM-BA 3.2.1 - Release Notes	10/11/2017 5:03:32 PM	5KB	
★SAM-BA 3.2.1 for Linux	10/11/2017 5:00:00 PM	60MB	
★SAM-BA 3.2.1 for Windows	10/11/2017 4:53:11 PM	14MB	
★SAM-BA v2.17 - Release Notes	6/15/2017 9:16:34 AM	21KB	
★SAM-BA v2.17 for Linux	6/15/2017 9:13:30 AM	20MB	
★SAM-BA v2.17 for Windows	6/15/2017 9:13:27 AM	11MB	
★SAM-BA 2.16 - Release Notes	6/12/2017 6:13:46 PM	21KB	
★atm6124 USB CDC signed driver for Windows (XP, Vista, Win7, Win8)	6/12/2017 6:09:35 PM	6KB	
★SAM-BA 2.16 for Linux	6/12/2017 5:53:09 PM	23MB	
★SAM-BA 2.16 for Windows (XP, Vista, Seven editions)	6/12/2017 5:49:38 PM	11MB	

Except for the 2.16 Windows version, the other packages come in a .zip file, requiring you to unzip the files before installing.

Complete the installation, and before opening the installed program, download the “**board-files.zip**” file from **our GitHub page**.

This zip will contain one single folder called “**tcl_lib**”, that you must paste at the **root** of the install location of your SAM-BA application, and overwrite any files. This will add a couple of customization files for your EDK board, and update a file of a list of devices.

The default installation folder is C:\Program Files (x86)\Atmel\sam-ba_2.1X\
(X being 6 for SAM-BA 2.16, and 7 for SAM-BA 2.17)

6. Connecting your EDK

To connect your EDK to your computer via Micro USB, you can use any Micro USB - USB cable. The board should start automatically in *Bootloader Mode*.

If there is no valid application present, the board will *always* start in *Bootloader Mode*. To reboot the board in *Bootloader Mode* at any time (such as after uploading a valid application):

- Remove any power supply from the board
- While **holding the user button**, power on the board.

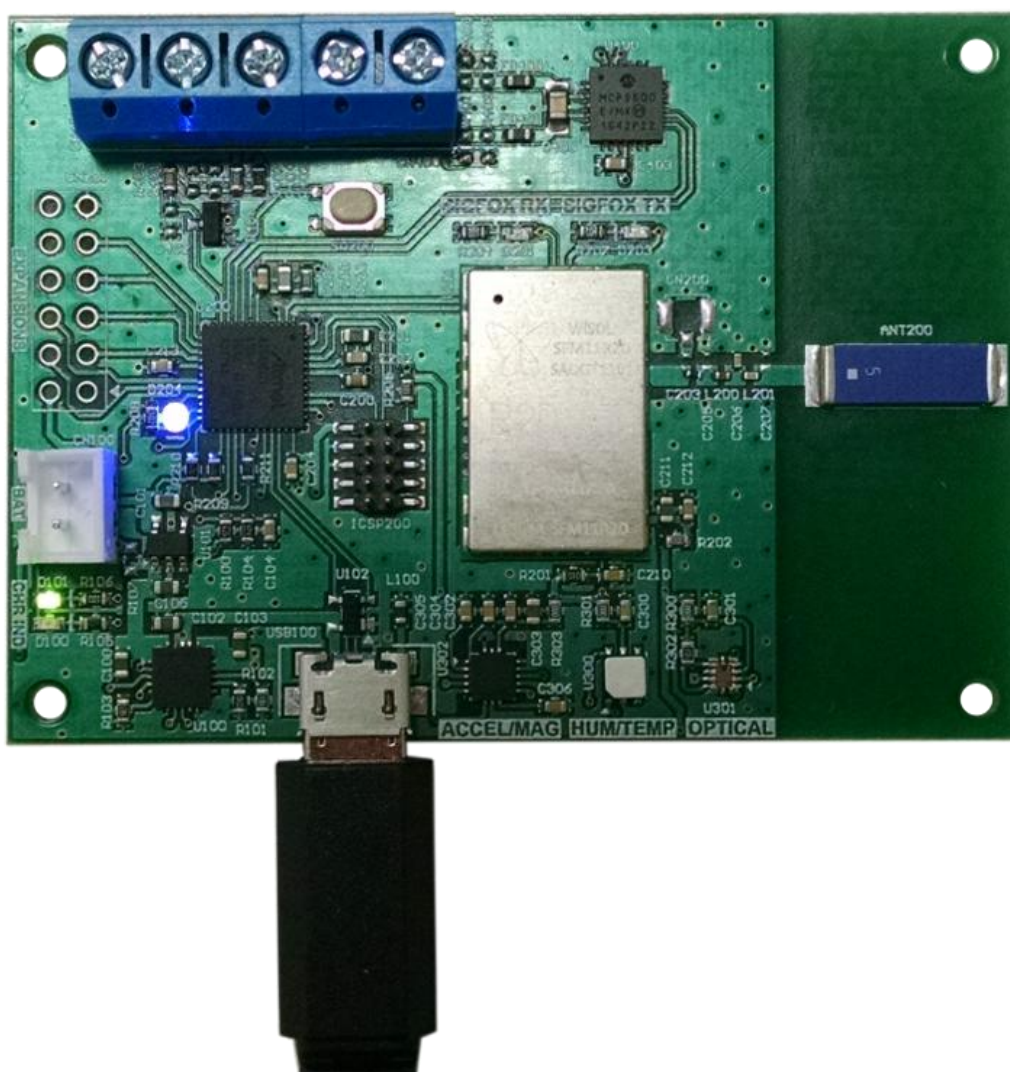


Figure: EDK connected via USB, in Bootloader Mode (solid blue light)

While the board is in *Bootloader Mode* via USB, you can upload the binary file of your application to it using the SAM-BA application (see [next section](#) for details).

Another way to connect your board is via a Programmer, such as the ATMEL-ICE (pictured below). This allows you to use Atmel Studio 7.0 and upload applications more freely to your board.



EDK connected via an ATMEL-ICE programmer, using the SAM interface.

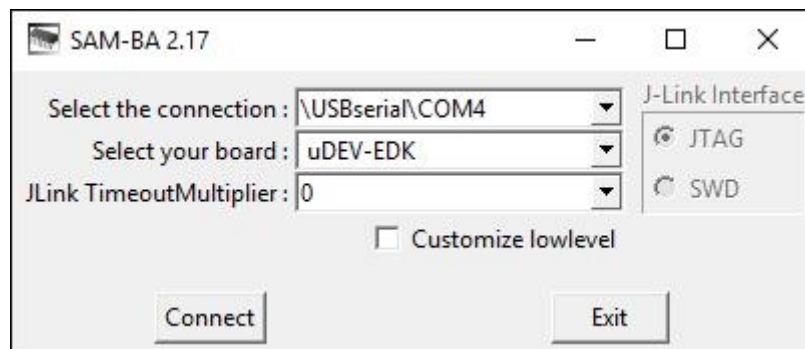
Please double-check the ribbon cable orientation before connecting your board. If using the standard ribbon cable that comes bundled with the ATMEL-ICE, the cable's red edge should be facing the Micro USB - meaning that PIN #1 should be to the side of the user button.

7. Using SAM-BA

Make sure your uDEV EDK board is connected, and has a **solid blue light**, indicating it is in *Bootloader Mode*.

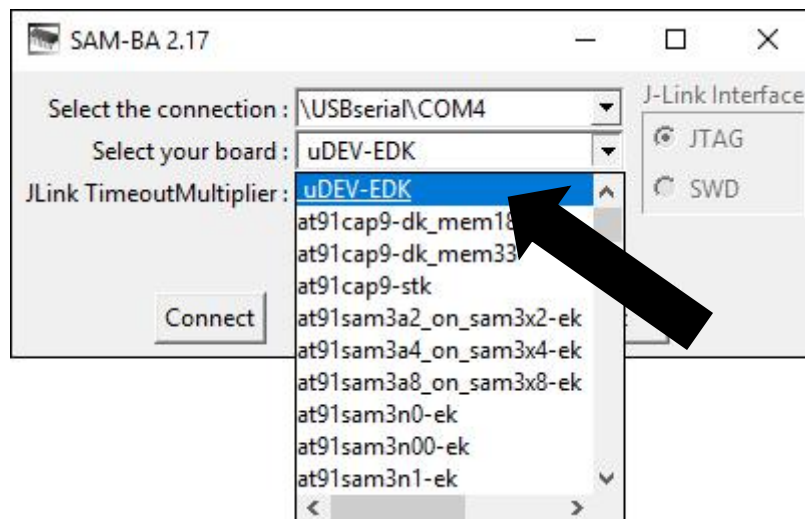
If the board is on but the blue light isn't, reset the board whilst keeping the user button pressed, and it will boot in *Bootloader Mode*.

After opening the **sam-ba.exe** application, the proper connection should be automatically selected.

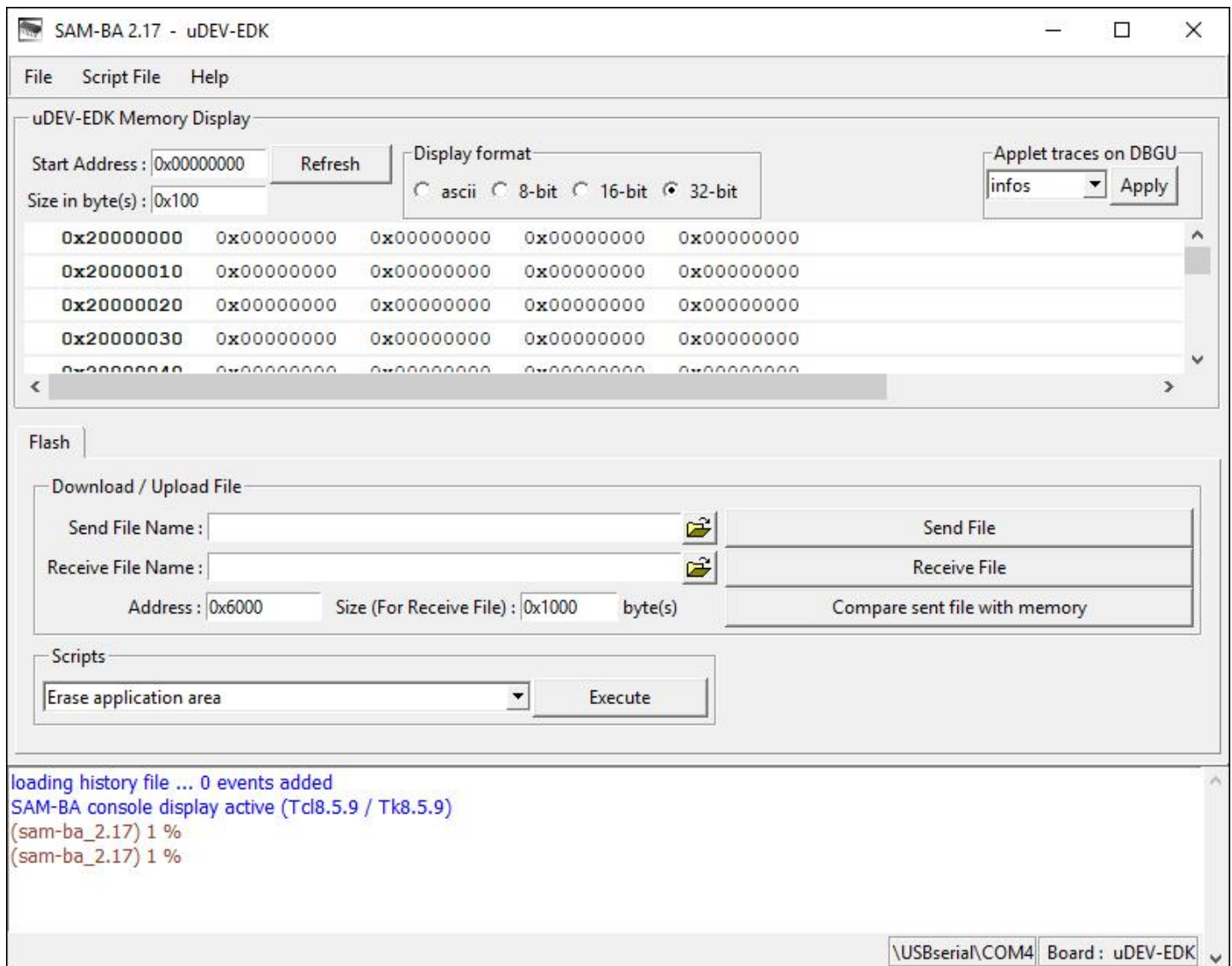


If you copied the board files correctly, the SAM-BA application should start with the uDEV-EDK board automatically selected, under the "Select your board" option.

If that doesn't happen, select **uDEV-EDK** as your board and click "Connect".



After connecting, the application should display the following window:



In this window, you can verify sections of the memory, and send the binary code of your program via the Send File command.

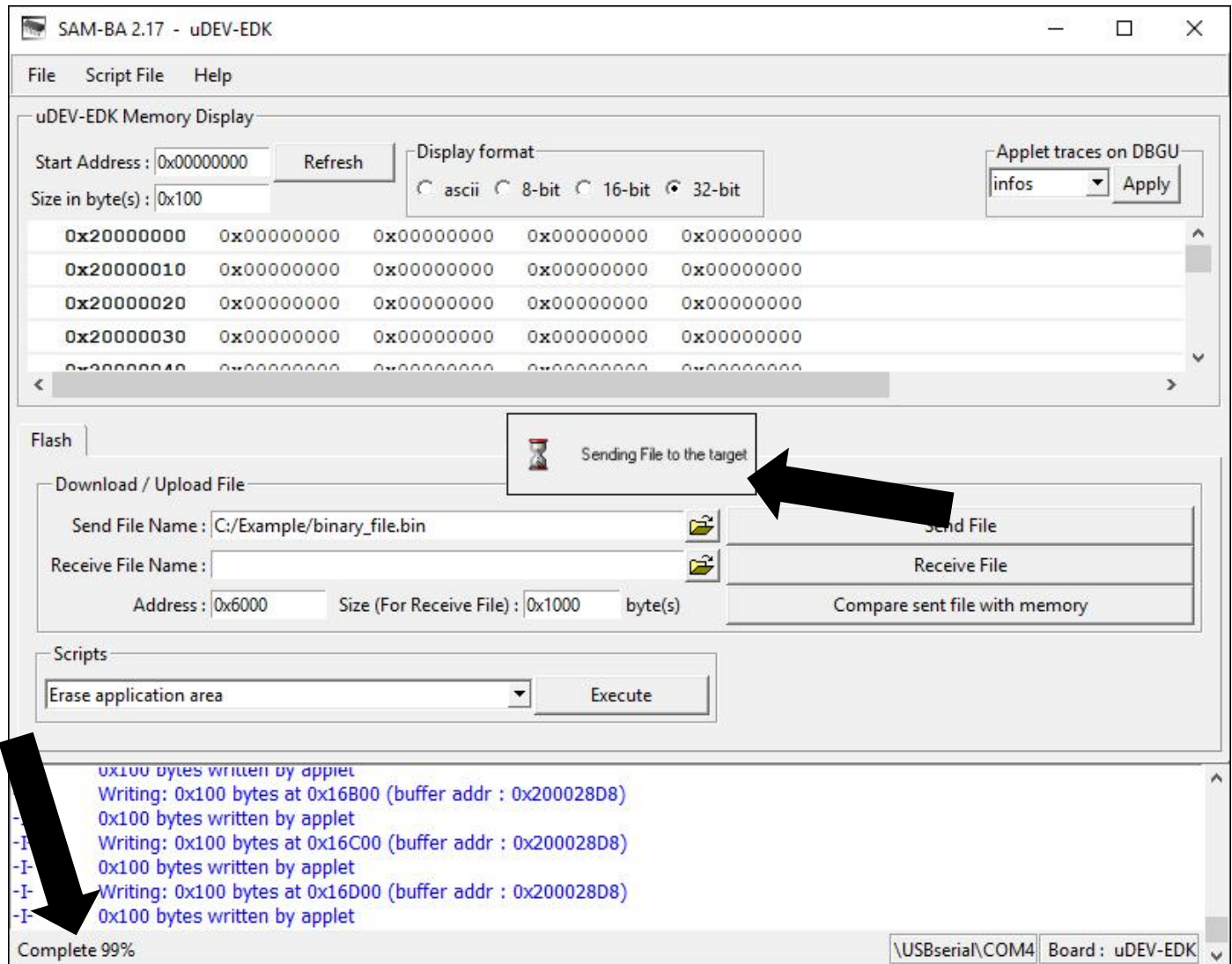
ATTENTION: Your program or application **MUST** start at flash memory address 0x6000 or higher! Any addresses lower than that will fail. Memory addresses 0x0000 - 0x6000 are reserved for the *Bootloader*.

Per default, when choosing the uDEV-EDK board, the address will be automatically set to 0x6000. Make sure this value is not changed before uploading your program to the board.

See the next section for an example of how to build your project at the correct address using Atmel Studio 7.0.

After selecting the .bin file and sending it with “Address” **set to 0x6000**, the SAM-BA application will load the program into the board’s flash memory.

A window will pop-up, indicating that the upload is in progress. Pay attention to the **lower status bar**, as it will display the progress of the upload.



If it reaches 100% and doesn't display any errors, your program should have been loaded into the flash memory and will be ready to use. Simply reboot the uDEV EDK board, and it will automatically begin in *Application Mode*.



You can verify if everything is correctly uploaded by using the **"Compare sent file with memory"** function, which compares the file in the **"Send File Name"** path with the *Flash Memory*, starting at the address indicated in the **"Address"** field.

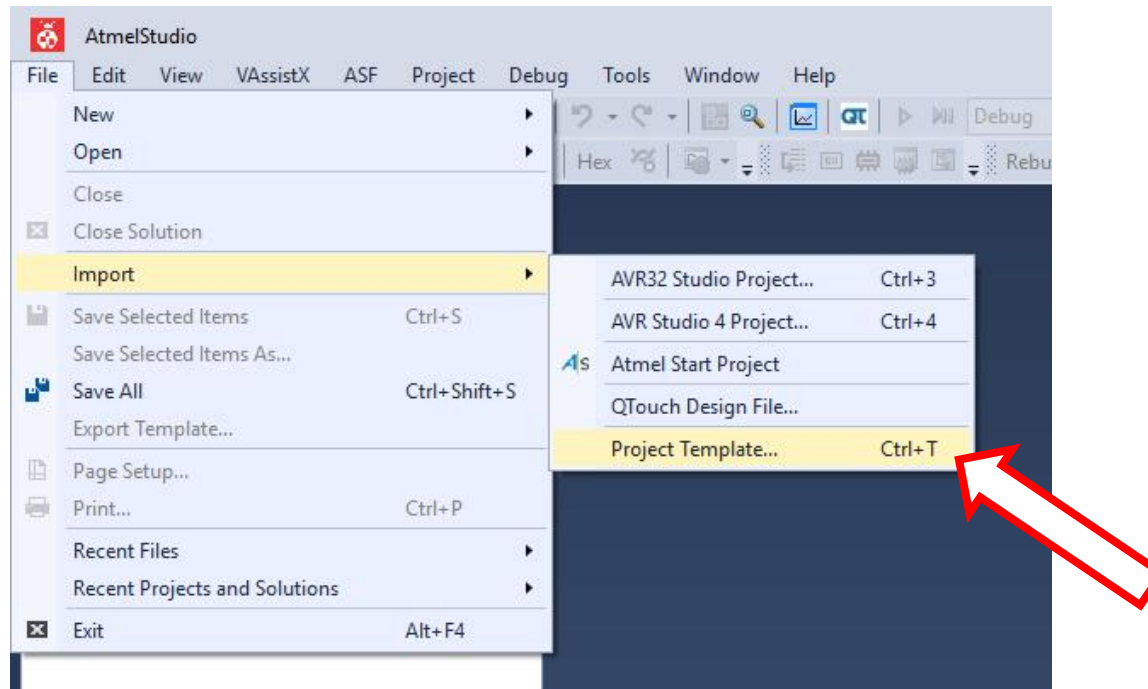
7. Testing

7.1. LED Project Template - Installing

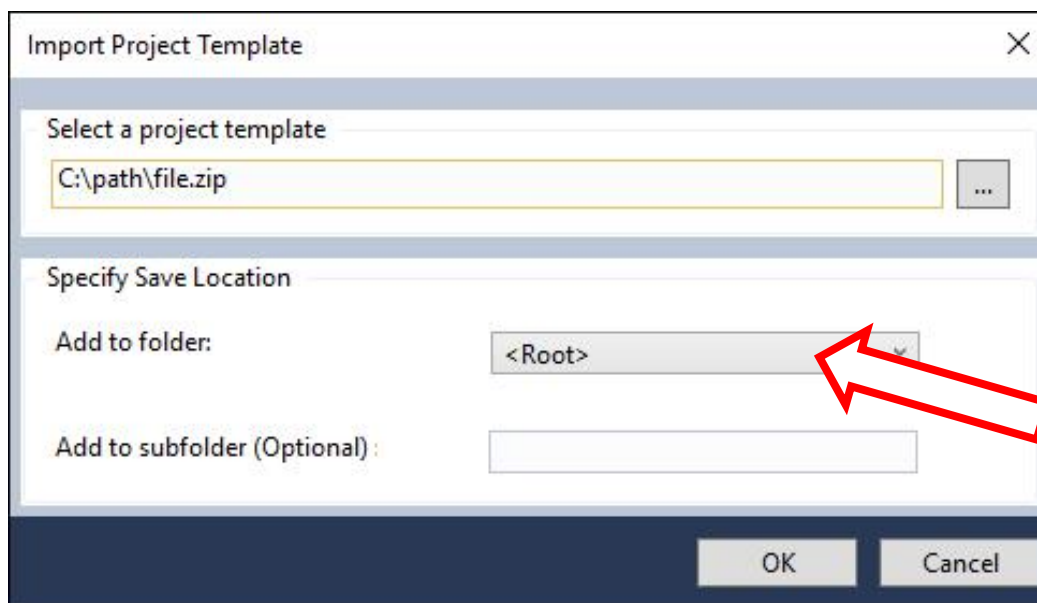
We have prepared a project template for Atmel Studio 7.0 so that you can install them and easily start your own Projects, specifically made for your uDEV EDK. The LED component project template is available at our page at [GitHub](#).

To import the template into your Atmel Studio application:

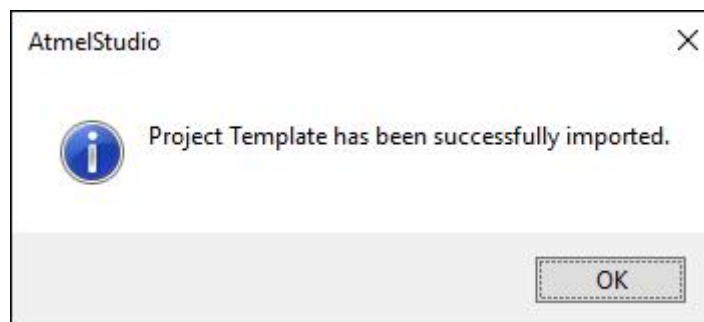
1. Go to **Import -> Project Template...**



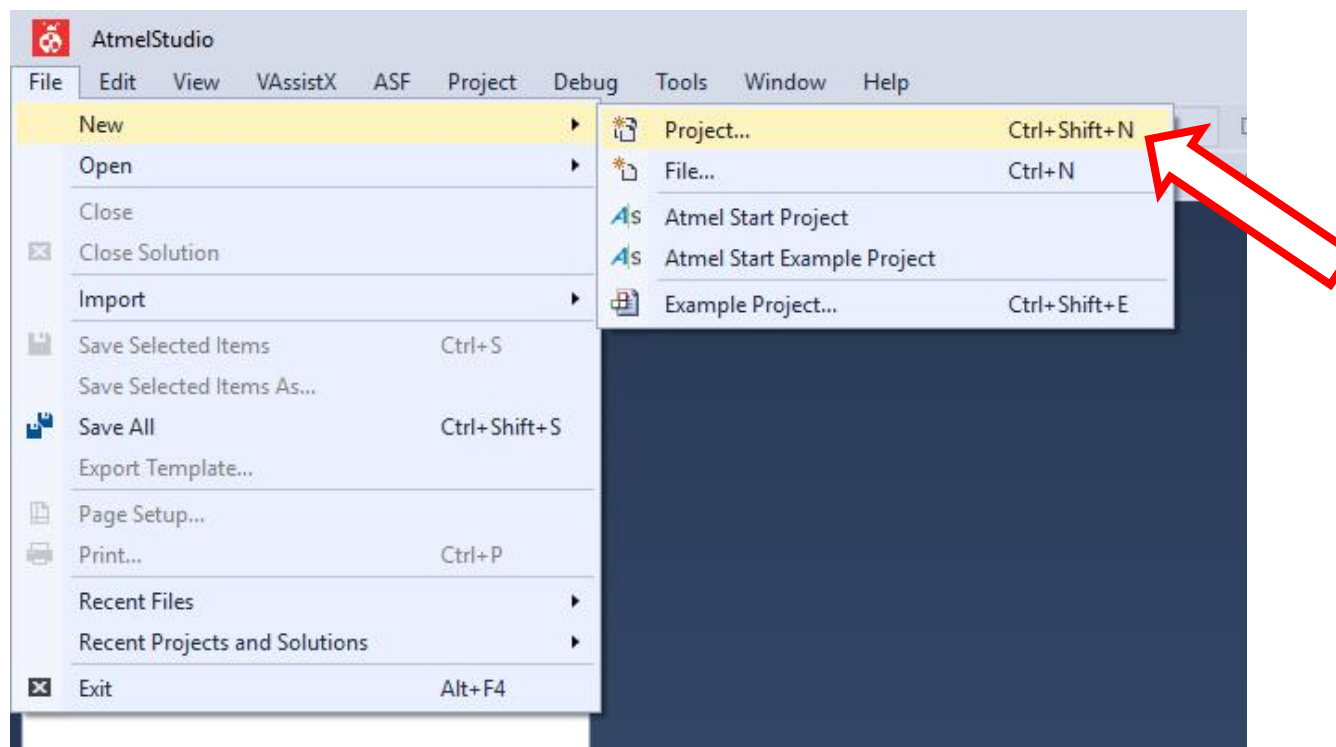
2. Then select the correct .ZIP file that has the template you want to install, and set the template to whatever folder you want. You may also add a Subfolder if desired.

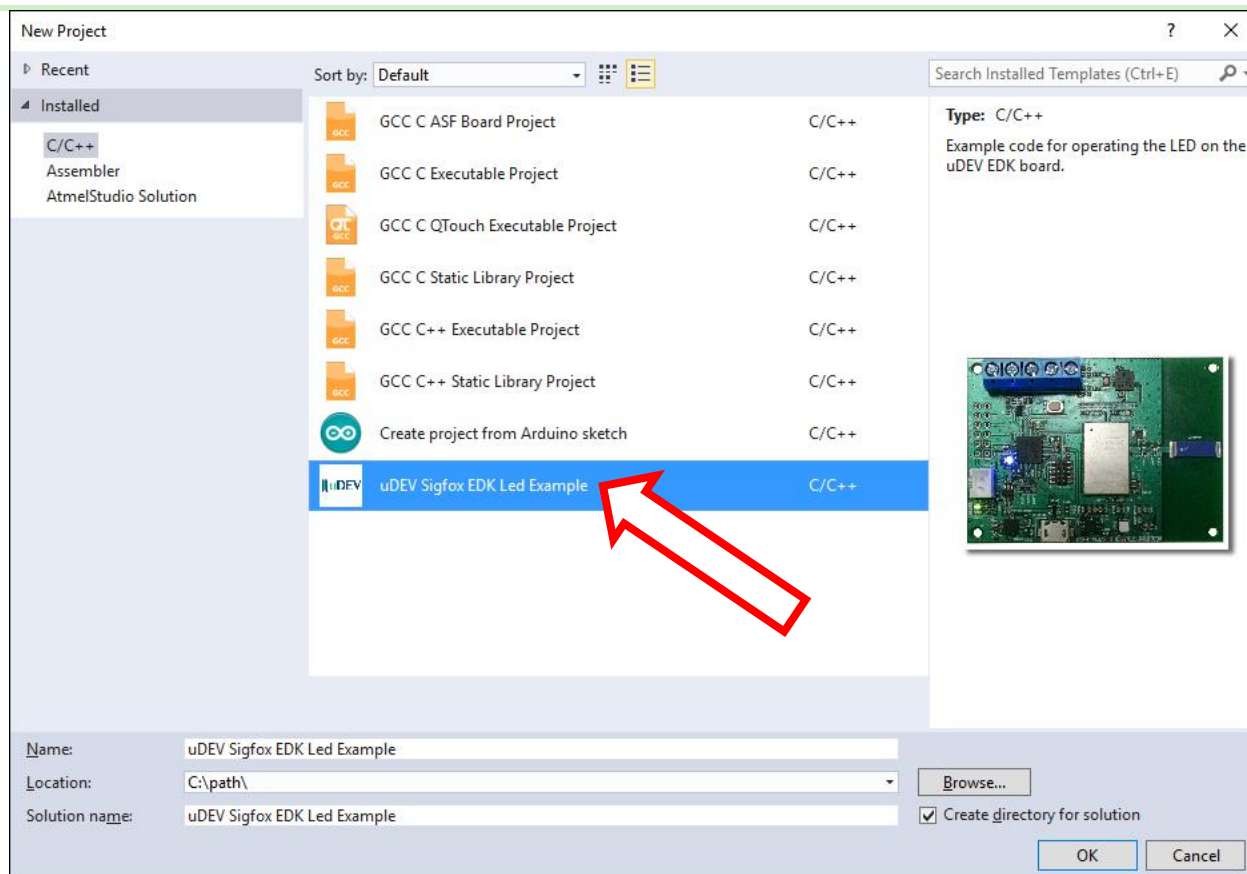


Atmel should display the following window:



3. After that, the Template should be available in **New -> Projects...**



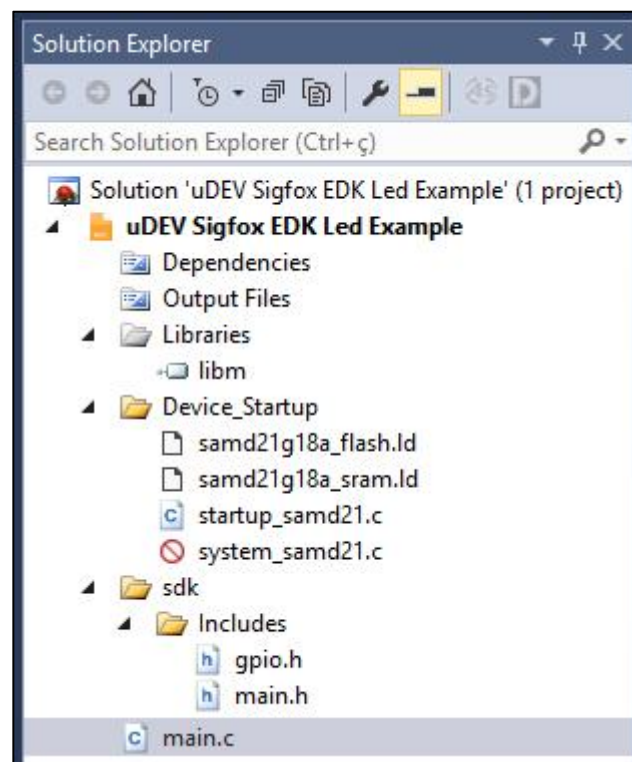


If the template does not appear, try restarting the Atmel Studio application completely, as it may be trying to refresh the template list.

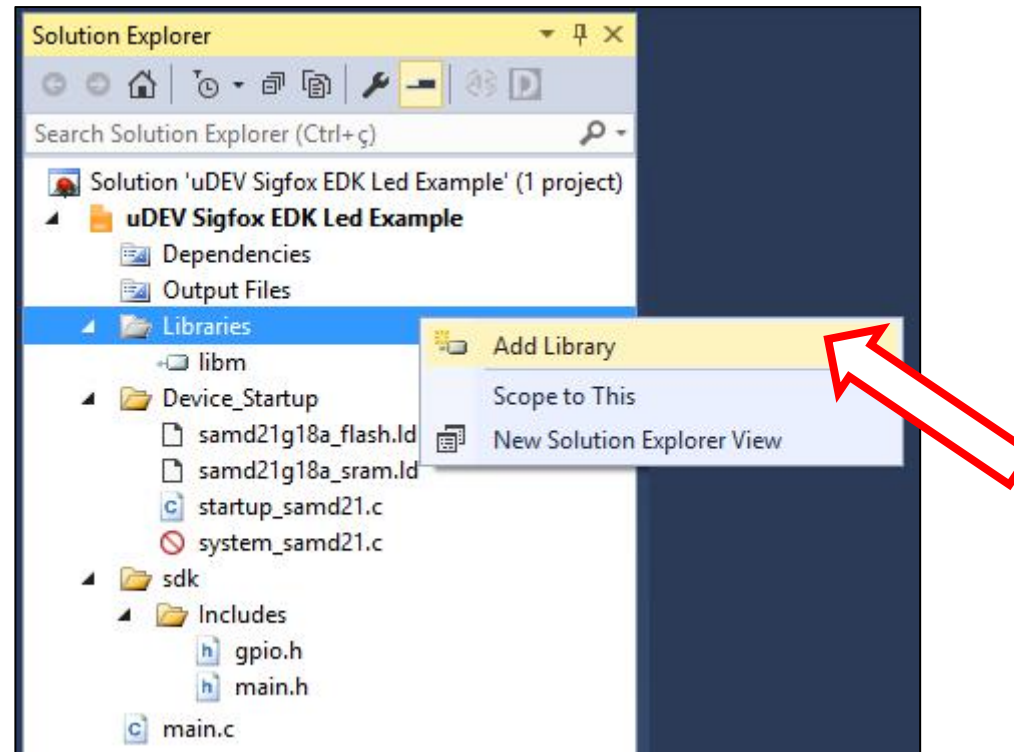
After creating your project using the uDEV template, you have to manually add the Library with the board configuration to the project parameters.

Luckily, this is a brief process.

Your solution explorer should be exactly as below:

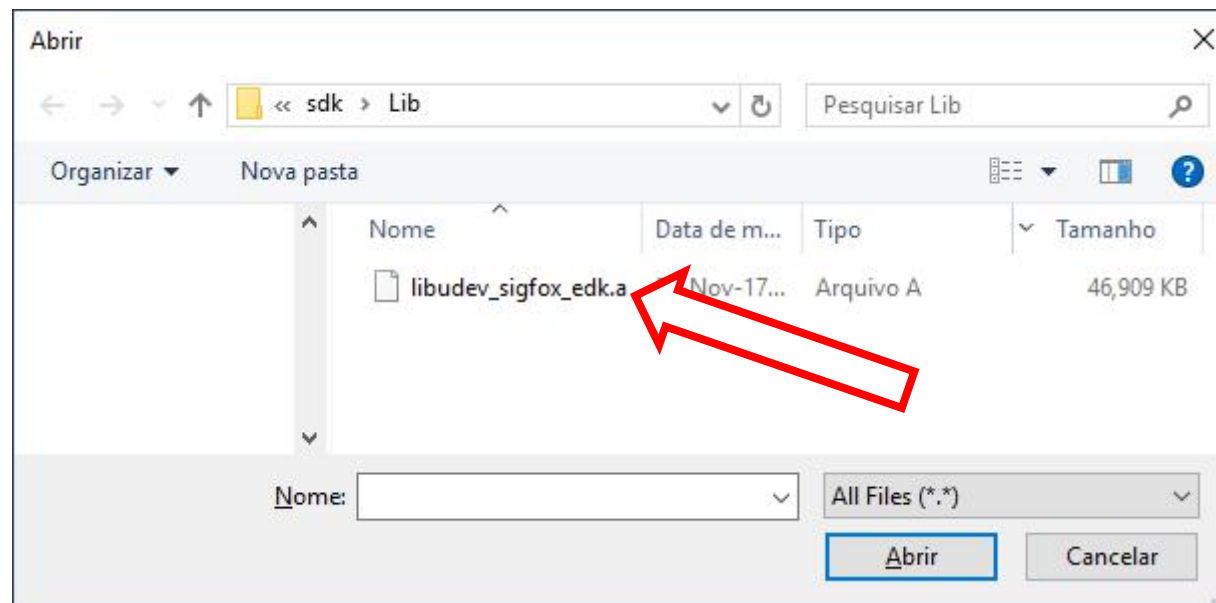
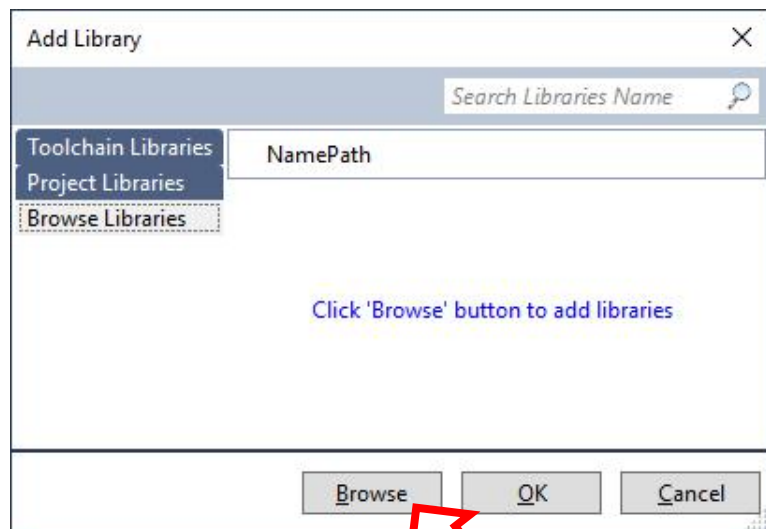


4. Simply **right-click** the “**Libraries**” folder, and select “**Add Library**”

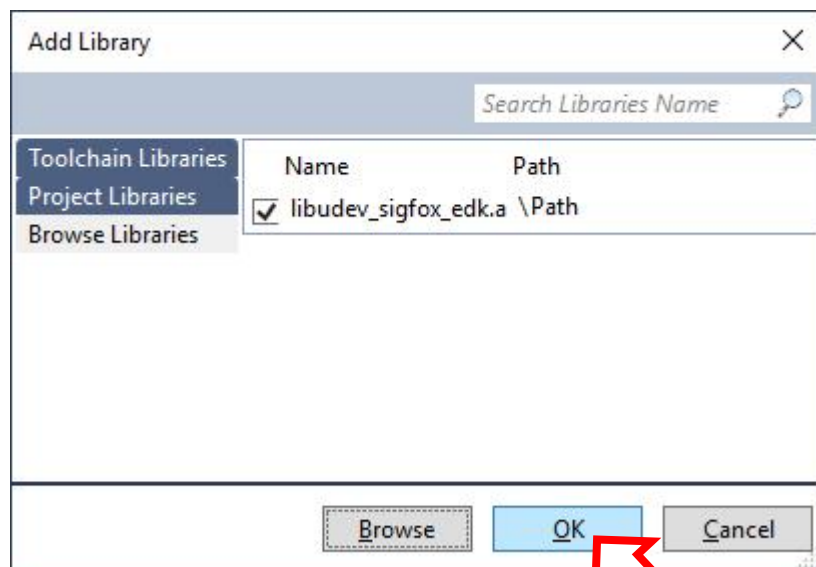


This will open a new window, so you can select which library to add to the project.

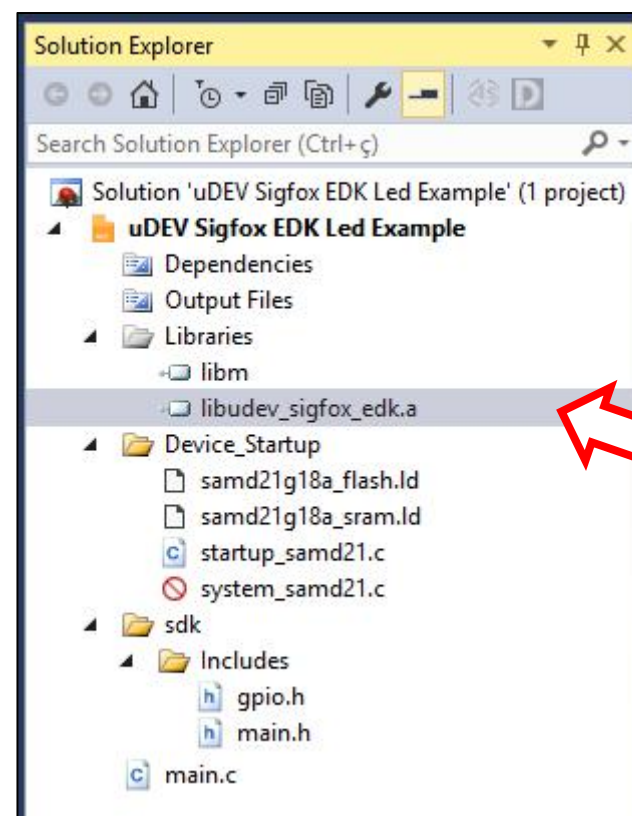
Choose “**browse library**” and navigate to wherever you saved the **udev_sigfox_edk.a** static library, and add that.



Select and add the library indicated, and click **ok**.



5. Check if everything was correctly added by verifying that the library appears in the Solution Explorer:



The LED Example is now ready to be built and executed on your uDEV Sigfox EDK!

7.2 Generic Project Template

Alternatively, a generic project template can be installed, and headers can be added as needed to the project.

Simply repeat the steps described above, but with the **Generic Project Template** .ZIP archive that is available in our GitHub.

This template will not contain **gpio.h** or other headers (apart from **main.h**) in the Includes folder, and you will need to add the header files that you'll use in your project, at your discretion.