SODAQ ONE Tracker v2 with Atmel Studio 7

Version 1 – August 2017 – Vincent van Beveren – vvanbeveren@gmail.com

You will notice how horribly wrong this goes each step of the way, and how we circumvent each barrier thrown in our way. Eventually you'll be able to run the Tracker in a debugging session and fix issues which may occur.

Requirements:

- SODAQ ONE board
- A supported debug probe. I've used a Segger JLink, but Atmel SAM-ICE should work also. Maybe others…
- 5 female mini-PV patch wires
- Atmel Studio 7 (7.0.1417 was used for this document)
- Latest Arduino software
- Sodaq ONE Tracker v2 sources

Notes:

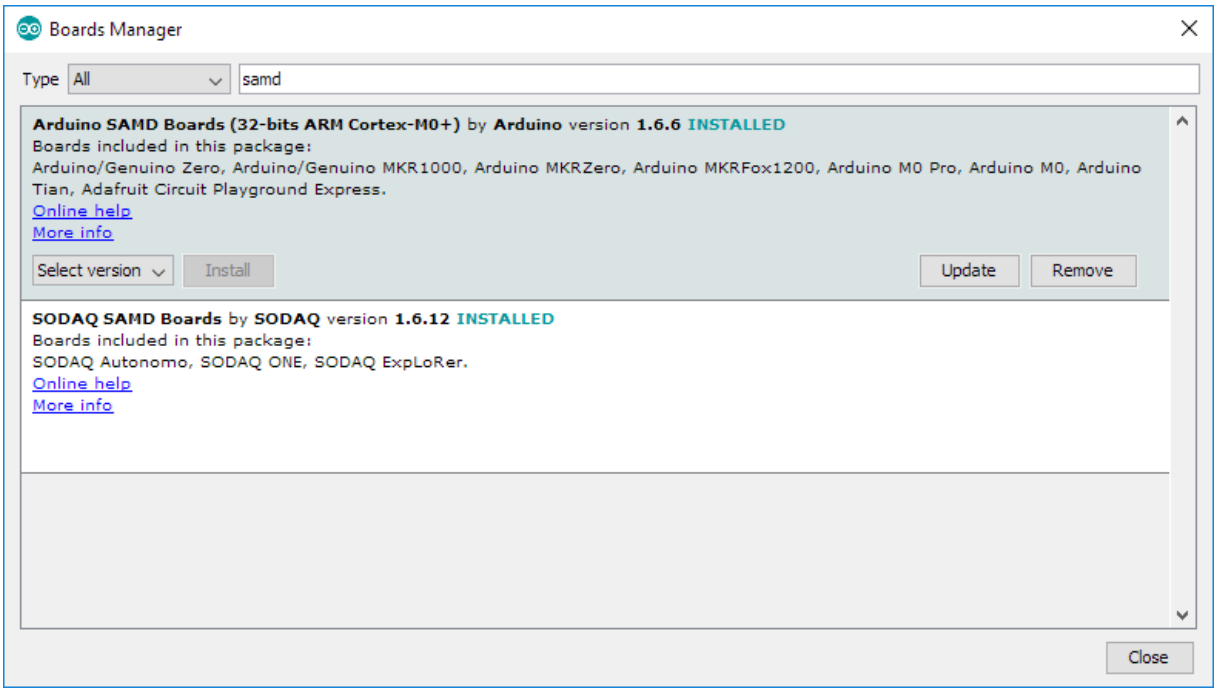
- Original tracker sketch will not be changed. Any changes made in Atmel Studio must be copied back into the Sketch if you wish to continue to use the Arduino environment
- Due to some limitations it is not possible to use the latest Arduino/SODAQ libraries. If you wish to use the latest, do not use this document.
- I have attempted to use VisualMicro. However, it did not seem to do much more than the Arduino environment. For some this may be a solution, but it wasn't one for me.
- ***I take no responsibility for bricked/broken hardware/software. Apply the steps at your own risk!***

Step 1: Correct library versions

First make sure you can build and run the SODAQ ONE tracker v2 application from Arduino. Also have a working Blink Sketch for SODAQ ONE. We're actually going to use Blink as a base, and pervert it to the tracker app.

You may wish to change the name of the Blink sketch to something like Tracker, otherwise the Blink name will persist right into your Atmel Studio project. I did not do this for this document (Didn't realize how difficult it was to change later on). Whenever I refer to Blink this should be the name of whatever you have chosen for your Blink Sketch.

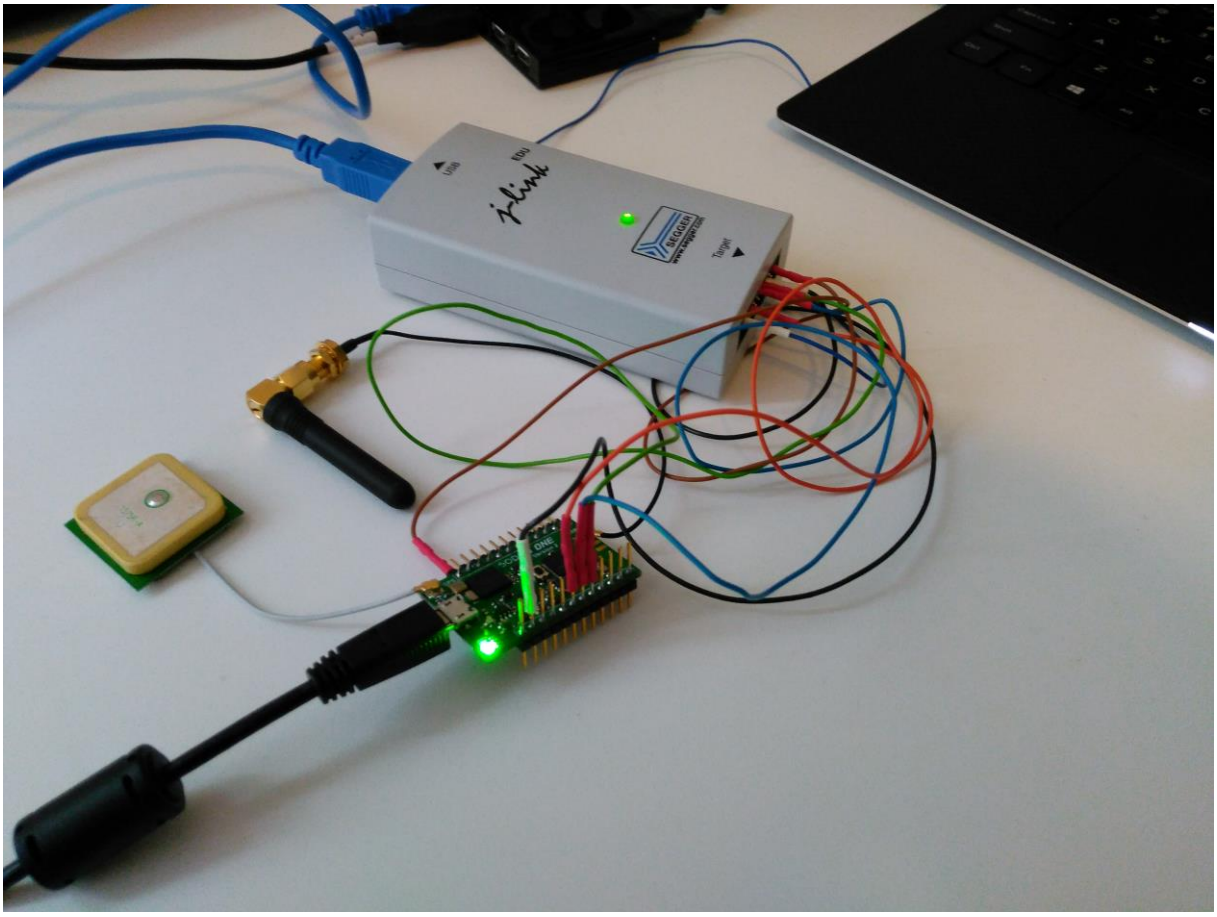
As of this writing Atmel Studio 7 does not support the newer Arduino libraries. Also the SODAQ libraries up to 1.6.12 seem to be based on an older version of the Arduino libraries. As far as I can tell, it seems to be closest to 1.6.6. So we need to downgrade the SAMD Boards to this old one, version 1.6.6, and the SODAQ SAMD Boards library to 1.6.12.

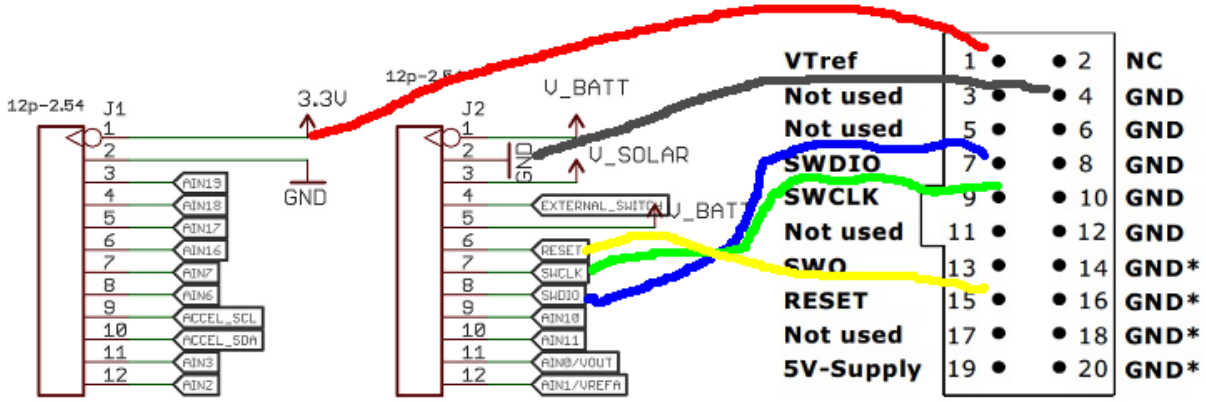


The arduino environment may now not compile anymore for SODAQ ONE, and say "Board SODAQ_one (platform samd, package SODAQ) is unknown". For running it in Atmel Studio this is not a problem. I have not attempted to fix this issue yet, though I suspect reinstalling the Arduino IDE will fix this (maybe the user libraries must be cleared first).

Step 2: Connect debug adapter, and make backup image

Connect an SWD a compatible debug adapter, and connect ARM SWD pins: SWDIO, SWCLK, RESET, GND and V3.3 (to VTref) (Could be different for Atmel debuggers, this is a JLink)



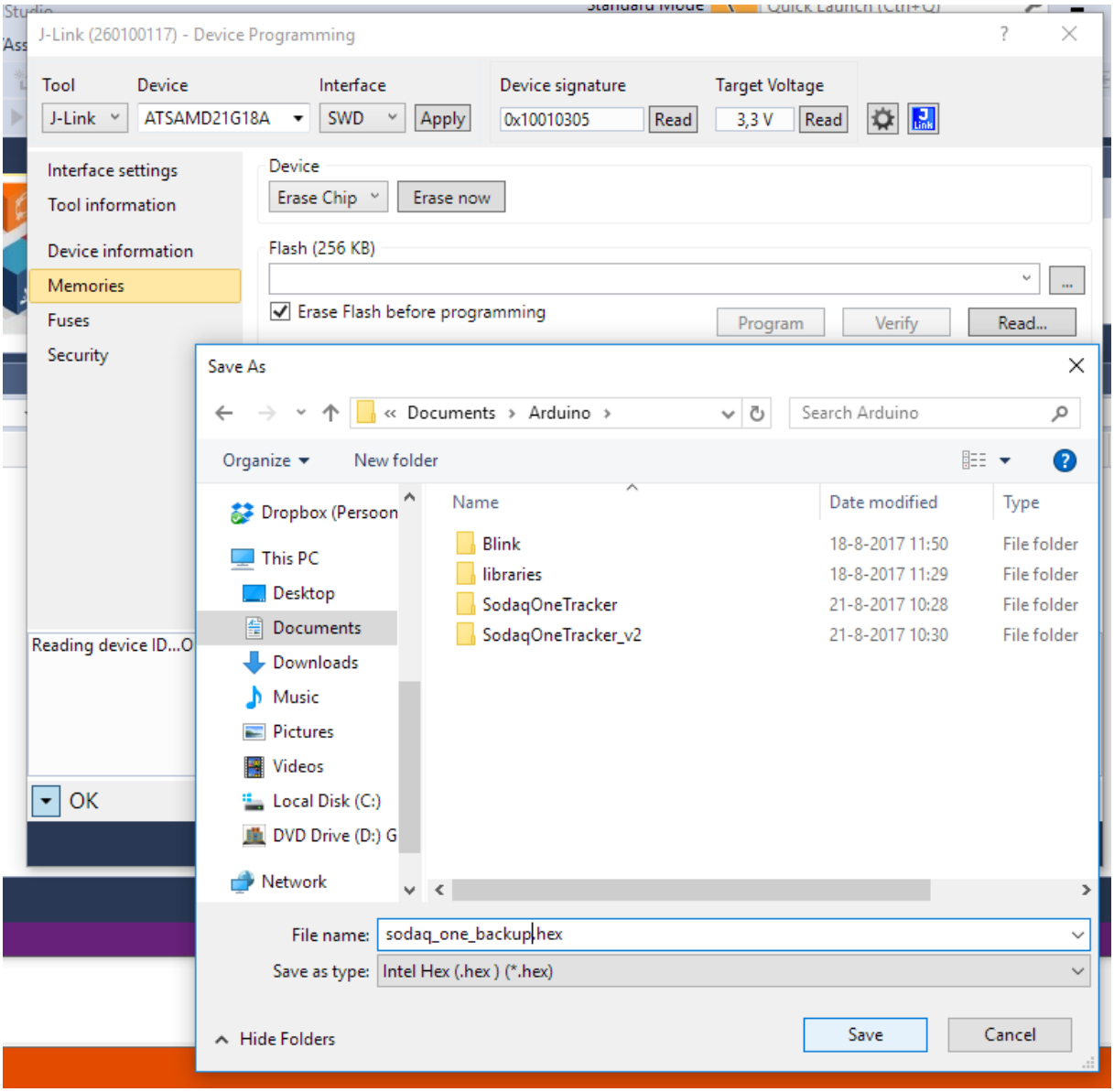


Connection from Link to SODAQ ONE board (colors do not match colors on picture)

Start Atmel Studio 7. First thing you should do is make a backup of the complete flash of the module. This way it will be easy to revert everything. In Atmel studio go to 'Tools' > 'Device programming'.

Select the correct tool, and device (ATSAMD21G18A), interface (SWD) and click on apply. As verification you might want to read the device signature (0x10010305) and check the voltage being 3.3V. If this all checks out you have connected the debugger correctly.

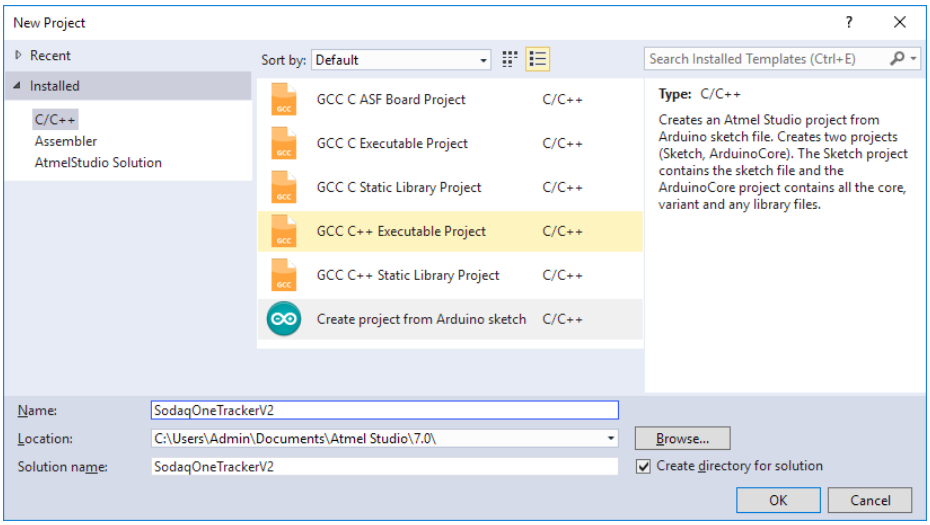
Go to the page ‘Memories’, and click on [Read…] at the Flash (256 KB section). Select a sensible name and save a backup. If you ever wish to revert, you can select this file and Program the micro back to its content as it was.



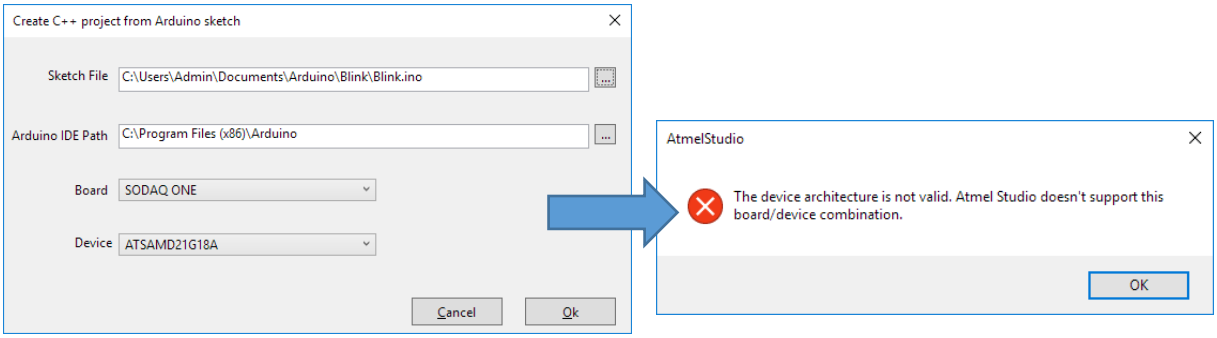
Now close the Device programming screen.

Step 3. Importing the tracker project, getting Blinky to run.

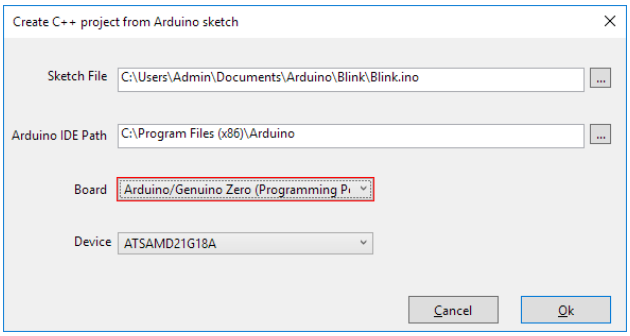
Go to ‘File’ > ‘New…’ > ‘Project…’.



Select ‘Create project from Arduino Sketch’, and give it a good name, e.g. ‘SODAQOneTrackerV2’. Click on [OK]. Select the Blink.ino file, *not the SODAQOneTracker, as Atmel studio completely fails to copy in the correct files to the correct locations*. Now this seems to be going very smoothly. SODAQ ONE is already available! … Unfortunately you will get an error trying this:



You will need to select ‘Arduino/Genduino Zero’, either Native USB / Programming Port, it won’t matter.



Press [OK].

Success! So now we have a Sketch. It will build and you can upload it. It won’t work though. But at least we can debug it, whatever it does.

Step 4. Getting Blinky to run *correctly*

We need to get down and dirty with the file system. Copy the correct files into the Atmel Studio project. Close Atmel Studio.

1. Open two explorer windows, one to the following Arduino library location:

C:\Users\<*username*>\AppData\Local\Arduino15\packages\SODAQ\hardware\samd\1.6.12\variants\sodaq_one

And one to the following location inside your solution:

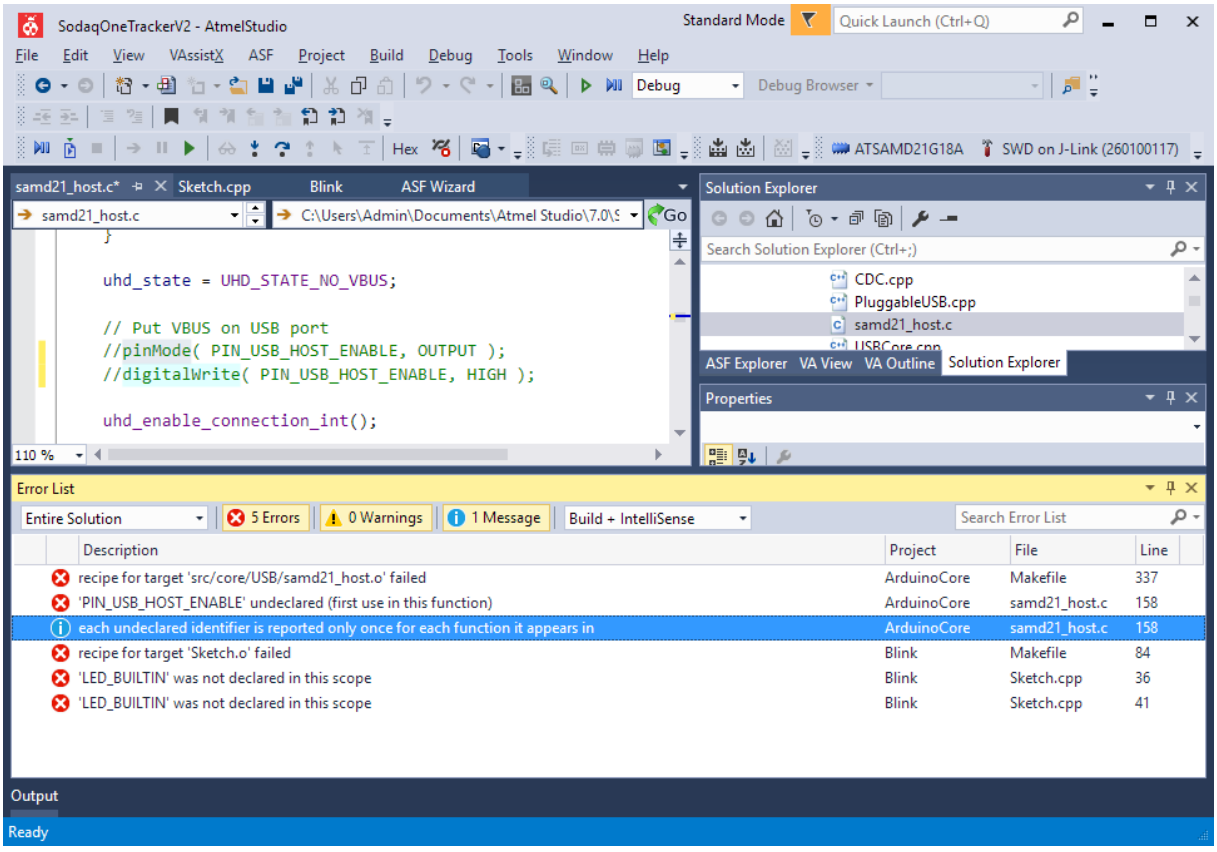
C:\<*solution location*> \ArduinoCore\include\variants

2. Replace the header (.h) files in the include\variants directory with those located in the variants\sodaq_one directory. Don't copy the .cpp file!

3. Now move inside the solution to the ArduinoCore\src\variants directory.

4. Copy variant.cpp from the library location into the solution's ArduinoCore source directory.

Open Atmel studio again and open the Tracker application. Clean and build the solution. It will fail, but with two distinct errors only (Ignore the .o failed messages)!



Click on the 2nd line and comment out the two lines regarding PIN_USB_HOST_ENABLE. Now click on any LED_BUILDIN error. Replace all instances of LED_BUILDIN with LED_GREEN inside your Sketch.cpp file. Now compile again, and there should be no errors!

Run it, and blinky should work! But now you can pause your app and inspect it. So much better…

Now take a deep breath as we're going to change blinky into tracker…

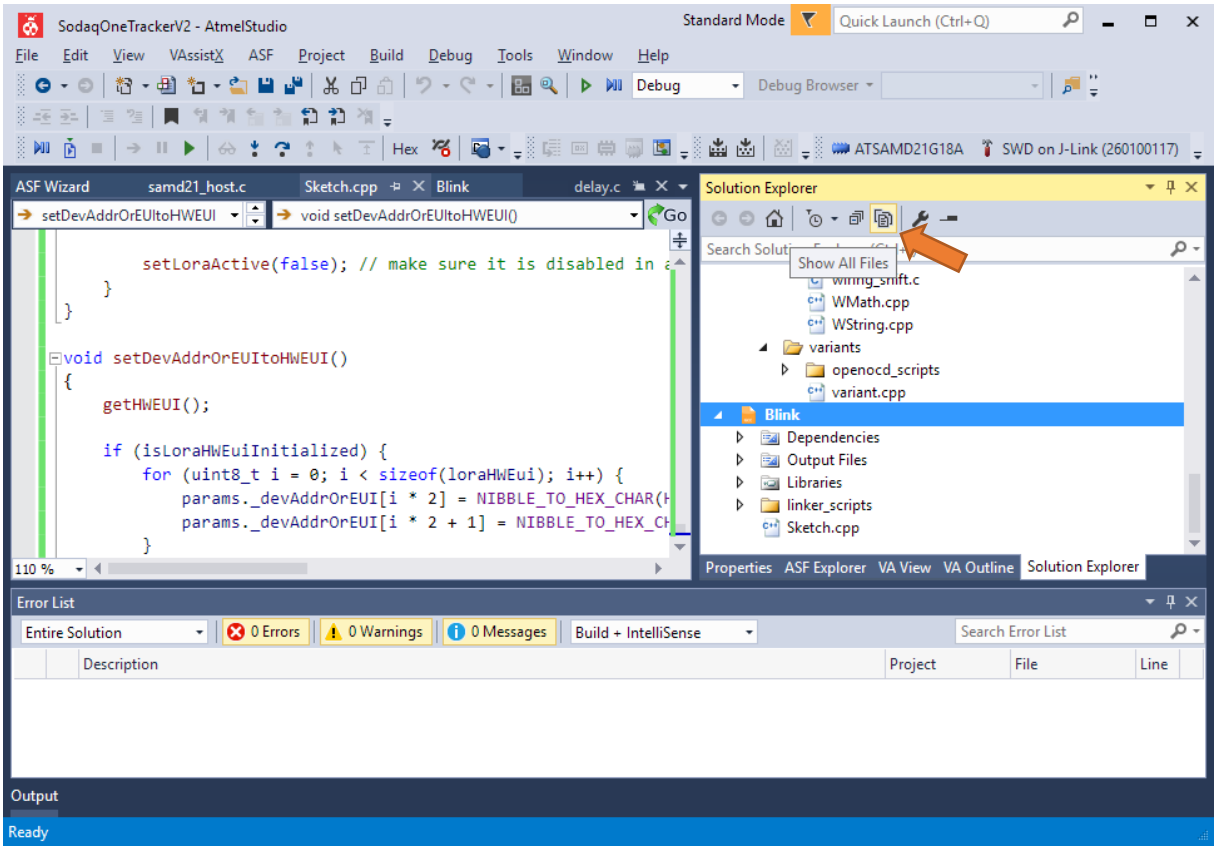
Step 5. Migrating the SODAQ ONE Tracker V2 application

First we'll need to copy all tracker files into the Blink project, in our solution. Open Windows explorer and navigate to the original tracker sketch. Select all C, CPP and H files in the Arduino SODAQ One Tracker V2 and copy it in the Blinky folder, next to Sketch.cpp.

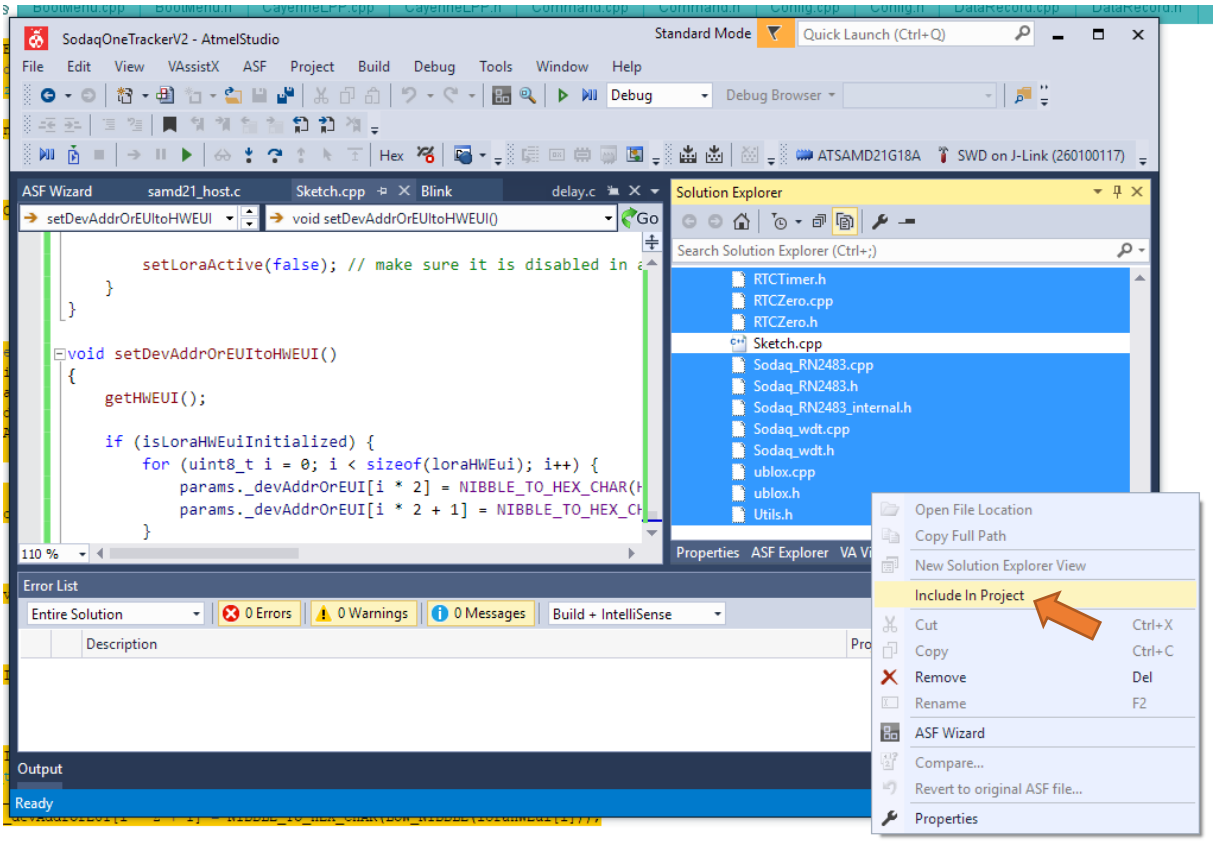
Open the SODAQOneTracker_v2.ino file from the Arduino project, and select its content, and copy everything (CTRL+A, CTRL+C) into the copy/paste buffer.

Go to Atmel Studio, and replace the content of Sketch.cpp with the content of the SODAQOneTracker_v2.ino (should still be in the buffer, just CTRL+A, CTRL+V in the Sketch.cpp editor). Save it (CTRL+S).

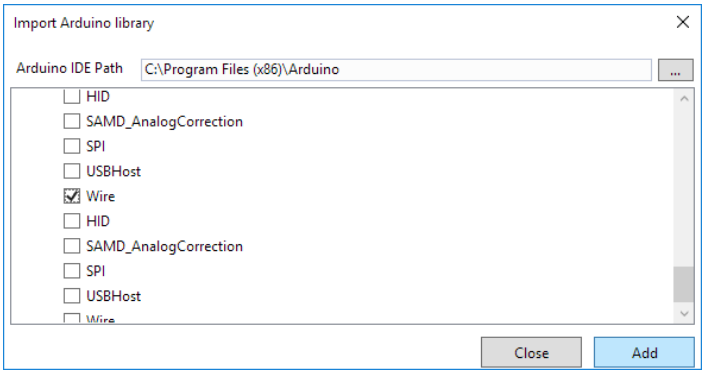
Though we copied all the tracker files into the Blink project, they're not visible and not compiled. Select the Blink project and click 'Show all files'.



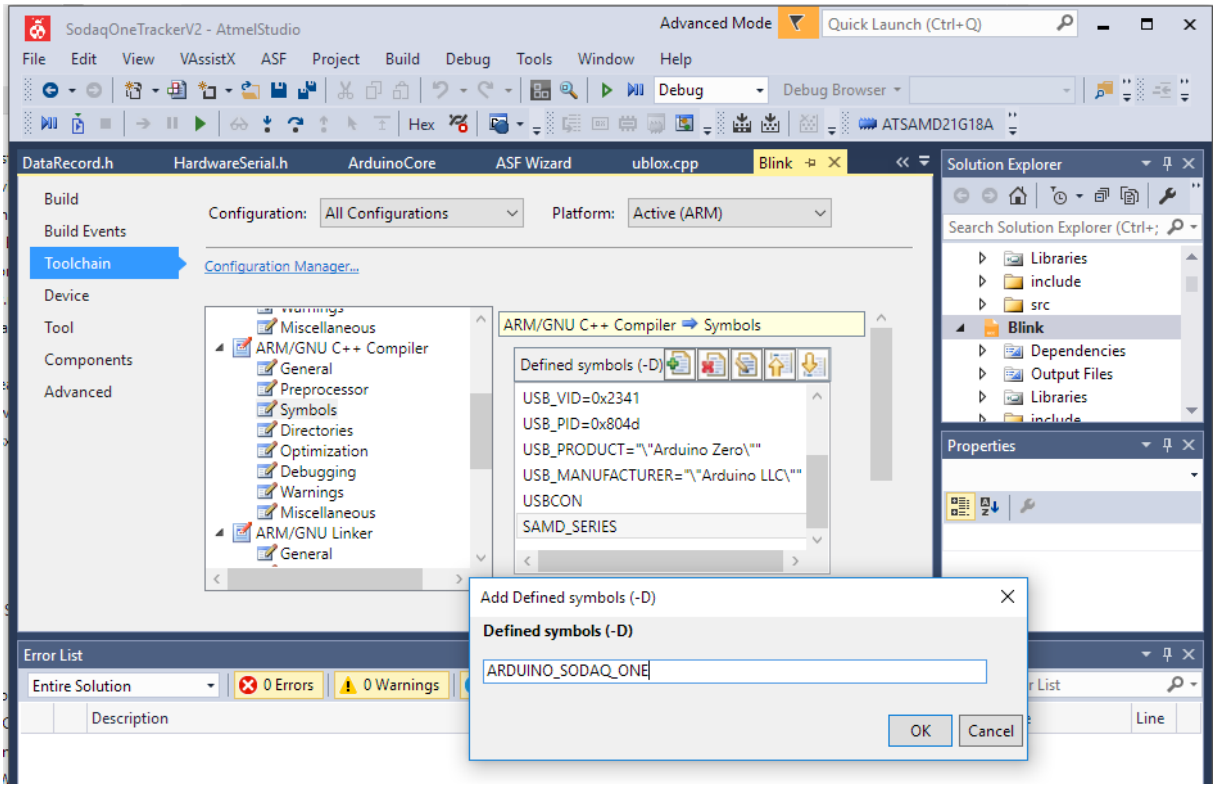
Next, select all the CPP, C and H files not yet in the Blink project. Now select ‘Include in project’.



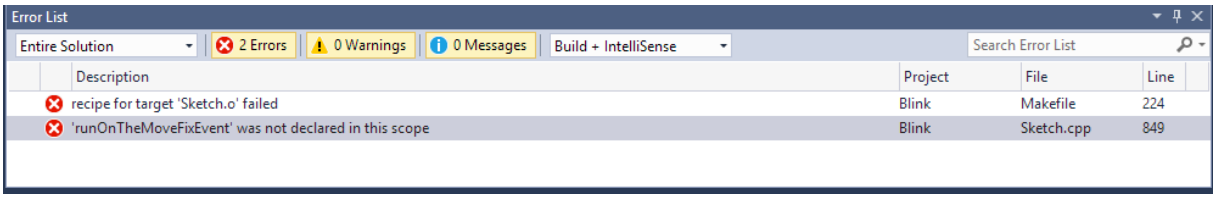
Right mouse click again on the Blink project, and select *Add Arduino Library*. Scroll down to SAMD and select Wire. Press [Add].



Now we need to add the symbol ARDUINO_SODAQ_ONE to the Blink Toolchain. Right mouse click on the Blink project, and select Properties. Go to 'ARM/GNU C Compiler' > 'Symbols' and add ARDUINO_SODAQ_ONE to the list of defined symbols, also go to to 'ARM/GNU C++ Compiler' > 'Symbols' and add the define there also.



Try to build it again. Now there should be just one error left:

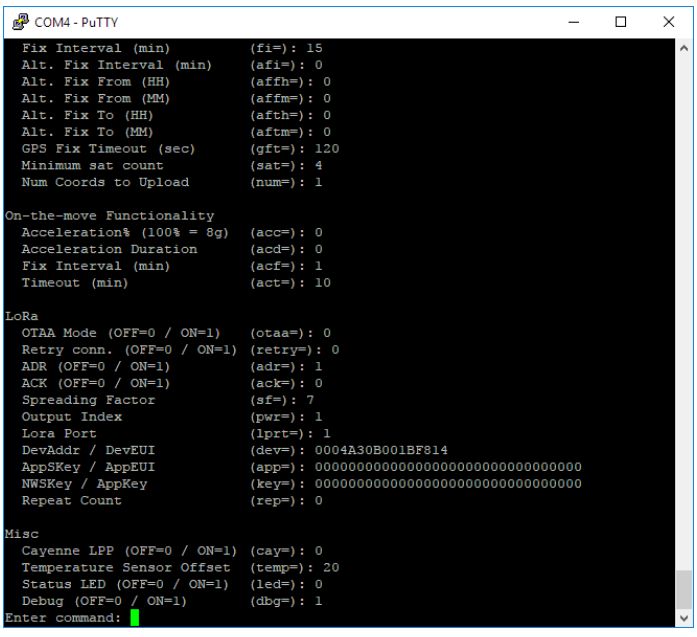


This can easily fix this by pre-declaring this function. Add the line:

```
void runOnTheMoveFixEvent(uint32_t now);
```

just before the

```
void resetRtcTimerEvents() function.
```



Now it should build and run correctly!

Some more remarks:

- Debug build still builds with optimization on, so there is no one to one relationship between the code and the program operation. Change the debug optimization to None(-O0).
- When running the DEBUG build you will see debugging output of the u-blox GPS. If this annoys you, you should remove the 'DEBUG' symbol from the 'Defined symbols' at the Toolchain properties, or modify ublox.cpp.
- If you get: underfined reference to 'vtable for <class name>' during the debug build a virtual function has not been set to 0. I encountered this when I tried to build with optimization level 0. I'm not sure why the error is not generated otherwise.

  HardwareSerial should look like this:

```
class HardwareSerial : public Stream
{
  public:
    virtual void begin(unsigned long) = 0;
    virtual void begin(unsigned long baudrate, uint16_t config) = 0;
    virtual void end() = 0;
    virtual int available(void) = 0;
    virtual int peek(void) = 0;
    virtual int read(void) = 0;
    virtual void flush(void) = 0;
    virtual size_t write(uint8_t) = 0;
    using Print::write; // pull in write(str) and write(buf, size) from Print
    virtual operator bool() = 0;
};
```

  Start of DataRecord should look like this:

```
class DataRecord
{
public:
    virtual void init() = 0;

    virtual bool isValid() const = 0;
    virtual uint16_t getSize() const = 0;
    virtual uint8_t getFieldCount() const = 0;
    // … rest is OK
```