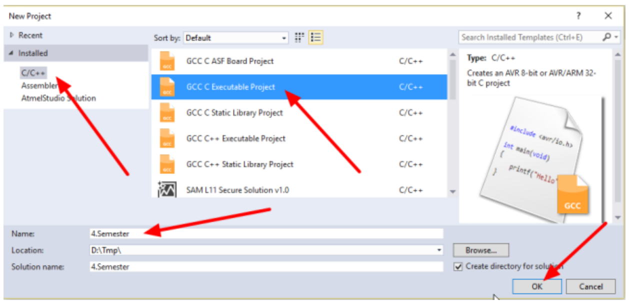
Setup of Project

Create Project

Open Microchip Studio.

Create a new project (here I give it the name *4.Semester*):



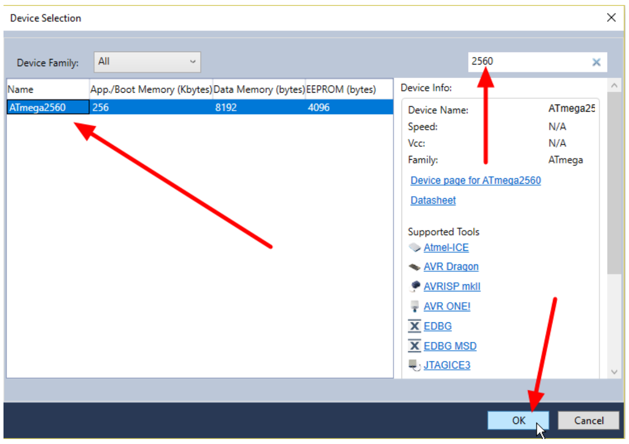
GCC C ASF Board Project	C/C++
GCC C Executable Project	C/C++
GCC C Static Library Project	C/C++
GCC C++ Executable Project	C/C++
GCC C++ Static Library Project	C/C++
SAM L11 Secure Solution v1.0	C/C++

Type: C/C++

Creates an AVR 8-bit or AVR/ARM 32-bit C project

```
#include <avr/io.h>
int main(void)
{
printf("Hello
}
```

Name: 4.Semester
Location: D:\Tmp\
Solution name: 4.Semester
☑ Create directory for solution



Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)
ATmega2560	256	8192	4096

Device Info:

Device Name:	ATmega25
Speed:	N/A
Vcc:	N/A
Family:	ATmega

Device page for ATmega2560
Datasheet

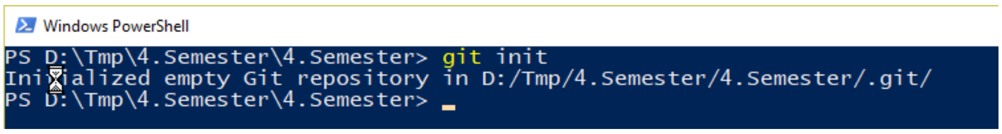
Supported Tools
Atmel-ICE
AVR Dragon
AVRISP mkII
AVR ONE!
EDBG
EDBG MSD
JTAGICE3

Put Project under Git Control

Open a command prompt in the project folder (here */Tmp/4.Semester/4.Semester*).

Init the git repository with this command:

`git init`



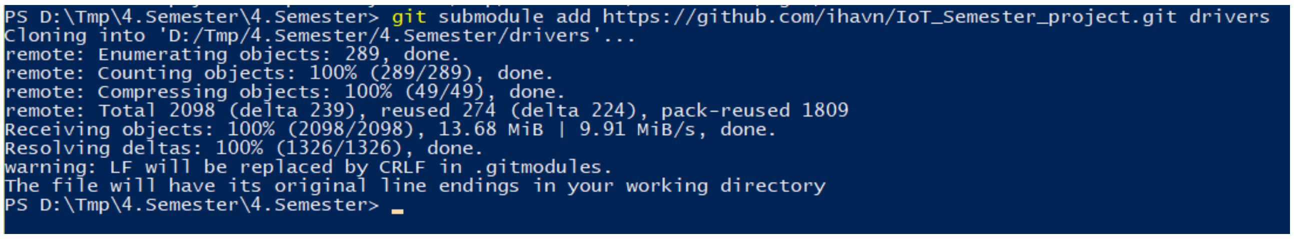
```
PS D:\Tmp\4.Semester\4.Semester> git init
Initialized empty Git repository in D:/Tmp/4.Semester/4.Semester/.git/
PS D:\Tmp\4.Semester\4.Semester> _
```

Add Drivers as Git Submodule

Open a command prompt in the project folder (here */Tmp/4.Semester/4.Semester*).

Add a git submodule with the drivers into your project using this command:

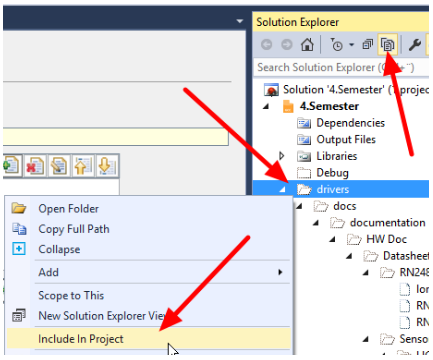
`git submodule add https://github.com/ihavn/IoT_Semester_project.git drivers`



```
PS D:\Tmp\4.Semester\4.Semester> git submodule add https://github.com/ihavn/IoT_Semester_project.git drivers
Cloning into 'D:/Tmp/4.Semester/4.Semester/drivers'...
remote: Enumerating objects: 289, done.
remote: Counting objects: 100% (289/289), done.
remote: Compressing objects: 100% (49/49), done.
remote: Total 2098 (delta 239), reused 274 (delta 224), pack-reused 1809
Receiving objects: 100% (2098/2098), 13.68 MiB | 9.91 MiB/s, done.
Resolving deltas: 100% (1326/1326), done.
warning: LF will be replaced by CRLF in .gitmodules.
The file will have its original line endings in your working directory
PS D:\Tmp\4.Semester\4.Semester> _
```

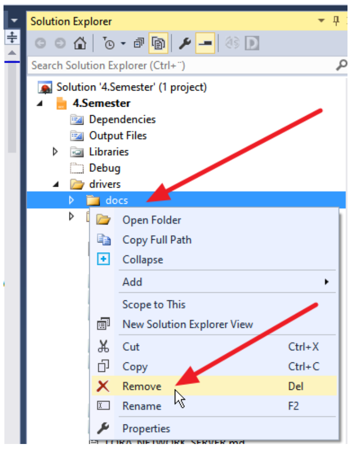
Include the drivers in the Project

Click on the *Show All Files* icon in the *Solution Explorer* and right click on the *drivers* folder and then *Include In Project*

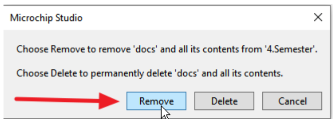


Now exclude the *drivers/docs* folder because it makes Microchip Studio slow.

Right click on the docs' folder and then *Remove*



And the click *Remove* not *Delete*


Microchip Studio

Choose Remove to remove 'docs' and all its contents from '4.Semester'.

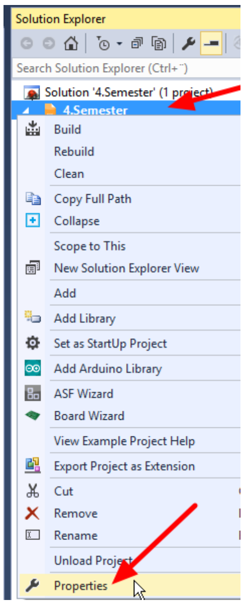
Choose Delete to permanently delete 'docs' and all its contents.

Then click again on *Show All Files* to deselect it.

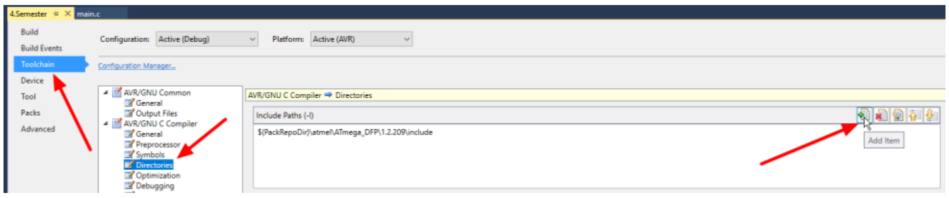
Setup Project references to the drivers

Include Path

First right click on the project and select *Properties*



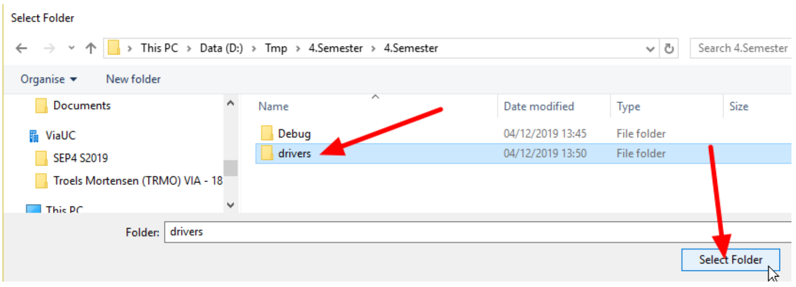
Then select *Toolchain, Directories* and *Add Item*



Configuration: Active (Debug) Platform: Active (AVR)

$(PackRepoDir)\atmel\ATmega_DFP\1.2.209\include

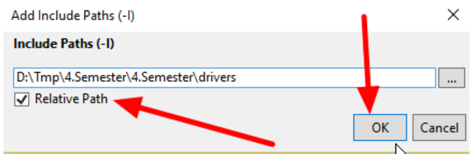
Browse to the *drivers* directory and *Select Folder*



Name	Date modified	Type	Size
Debug	04/12/2019 13:45	File folder	
drivers	04/12/2019 13:50	File folder	

Folder: drivers

Make it a *Relative Path:*

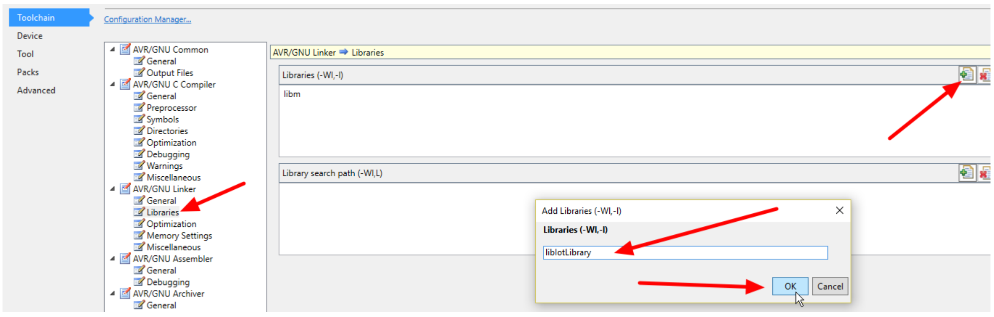


D:\Tmp\4.Semester\4.Semester\drivers

☑ Relative Path

Library Path and File

Click on *Libraries* and the add icon and put in the library name ***libIotLibrary*** be sure to spell it correct!!

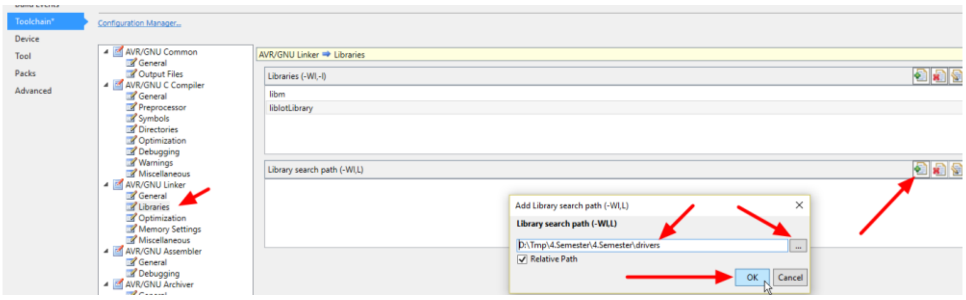


Libraries (-Wl,-l)

libm

Libraries (-Wl,-l): libIotLibrary

Now add the path to the folder where the library is located by clicking on the add icon and browse to the driver's folder



AVR/GNU Linker ➡ Libraries

Libraries (-Wl,-l)
libm
liblotLibrary

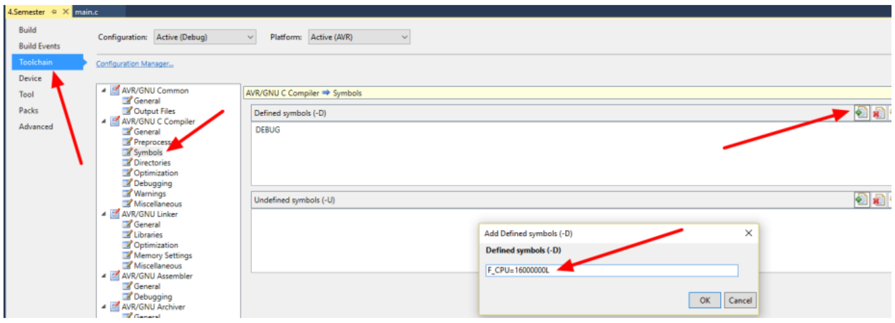
Library search path (-Wl,L)

Add Library search path (-Wl,L)
Library search path (-Wl,L)
D:\Tmp\4.Semester\4.Semester\drivers
☑ Relative Path

Then save the settings by saving the property file.

Setup MCU Clock Frequency

Click on *Symbols* and the add icon and set *F_CPU=16000000L*



Configuration: Active (Debug)   Platform: Active (AVR)

AVR/GNU C Compiler ➡ Symbols

Defined symbols (-D)
DEBUG

Undefined symbols (-U)

Add Defined symbols (-D)
Defined symbols (-D)
F_CPU=16000000L

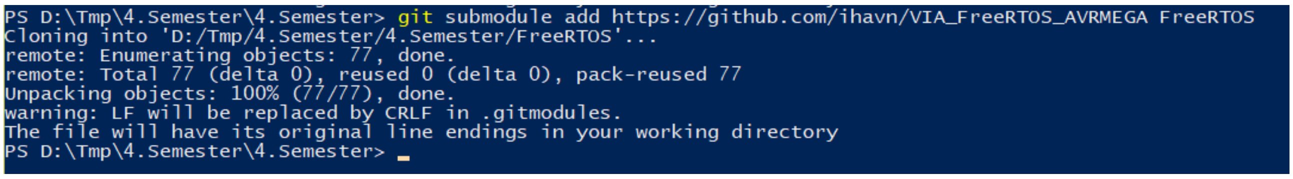
Add FreeRTOS as Git Submodule

The last thing is to add the ATMEGA2560 port for FreeRTOS as a submodule. This is found and described here:

https://github.com/ihavn/VIA_FreeRTOS_AVRMEGA

Here is how I have added the submodule

git submodule add https://github.com/ihavn/VIA_FreeRTOS_AVRMEGA FreeRTOS

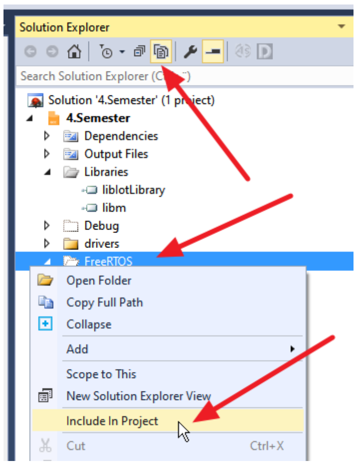


```
PS D:\Tmp\4.Semester\4.Semester> git submodule add https://github.com/ihavn/VIA_FreeRTOS_AVRMEGA FreeRTOS
Cloning into 'D:/Tmp/4.Semester/4.Semester/FreeRTOS'...
remote: Enumerating objects: 77, done.
remote: Total 77 (delta 0), reused 0 (delta 0), pack-reused 77
Unpacking objects: 100% (77/77), done.
warning: LF will be replaced by CRLF in .gitmodules.
The file will have its original line endings in your working directory
PS D:\Tmp\4.Semester\4.Semester> _
```

Finally include the FreeRTOS source in the project the same way as you did for the drivers (see Include the drivers in the Project).

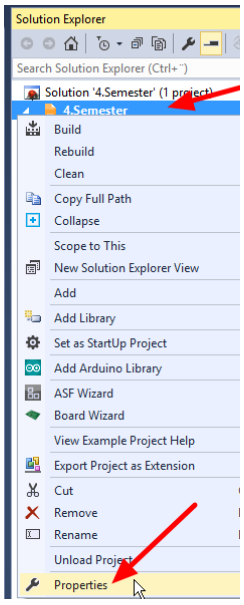
Include FreeRTOS in the project

Click on the *Show All Files* icon in the *Solution Explorer* and right click on the *FreeRTOS* folder and then *Include In Project*

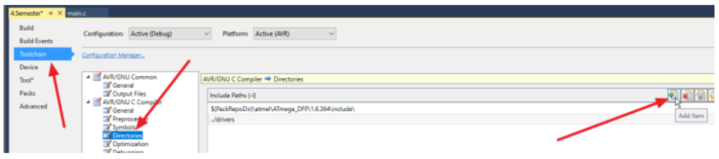


Include Path

First right click on the project and select *Properties*



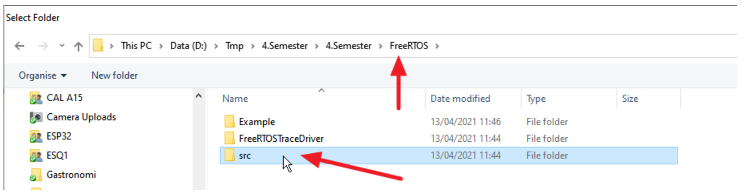
Then select *Toolchain, Directories* and *Add Item*



Configuration: Active (Debug)
Platform: Active (AVR)

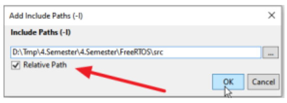
Include Paths (-I)
$(PackRepoDir)\atmel\ATmega_DFP\1.6.364\include\
../drivers

Browse to the *FreeRTOS\src* directory and *Select Folder*



Name	Date modified	Type	Size
Example	13/04/2021 11:46	File folder	
FreeRTOSTraceDriver	13/04/2021 11:44	File folder	
src	13/04/2021 11:44	File folder	

Make it a *Relative Path*



D:\Tmp\4.Semester\4.Semester\FreeRTOS\src
[✓] Relative Path

Example files

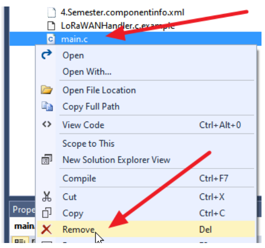
In both the FreeRTOS and drivers' submodules there are some example files that can be used as inspiration.

For this semester project you are only interested in the files found under the drivers *Example* folder.

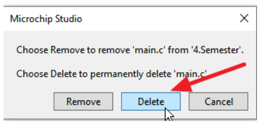
In the this C project you are interested in:

- LoRaWANHandler.c.example
- main.c.example

First delete the *main.c* file that is in the root folder of the project



This time you must delete *main.c* completely!

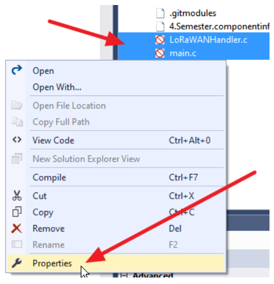

Microchip Studio

Choose Remove to remove 'main.c' from '4.Semester'.

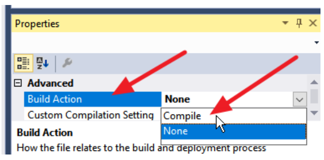
Choose Delete to permanently delete 'main.c'.

You can then copy the two example files to your projects root folder and then remove the *.example* extension form them.

Include them in the compilation by selecting both files and right click and the *Properties*:



Select *Compile* under *Build Action*



Build Action

How the file relates to the build and deployment process

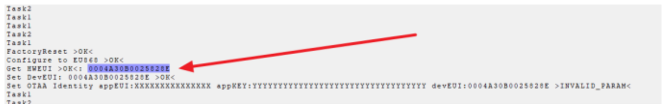
First Run

Now it should be possible to build your project without any errors.

Connect the hardware you have been given.

Setup a serial terminal on your laptop and configure it to 57600,N,8,1. As serial terminal H-Term is recommended (https://www.der-hammer.info/pages/terminal.html).

When you run your program for the first time the out put to the terminal will contain a *HWEUI* hex string that you must email to your ESW1/SEP4 IoT Teacher.



```
Task2
Task1
Task1
Task2
Task1
FactoryReset >OK<
Configure to EU868 >OK<
Get HWEUI >OK<: 0004A30B0025828E
Set DevEUI: 0004A30B0025828E >OK<
Set OTAA Identity appEUI:XXXXXXXXXXXXXXXX appKEY:YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY devEUI:0004A30B0025828E >INVALID_PARAM<
Task1
Task2
```

I will then setup the LoRaWAN Network Server to make it possible for your device to connect to the LoRaWAN. You will receive and email with the *appEUI* and *appKEY* hex strings that you will need for your application.