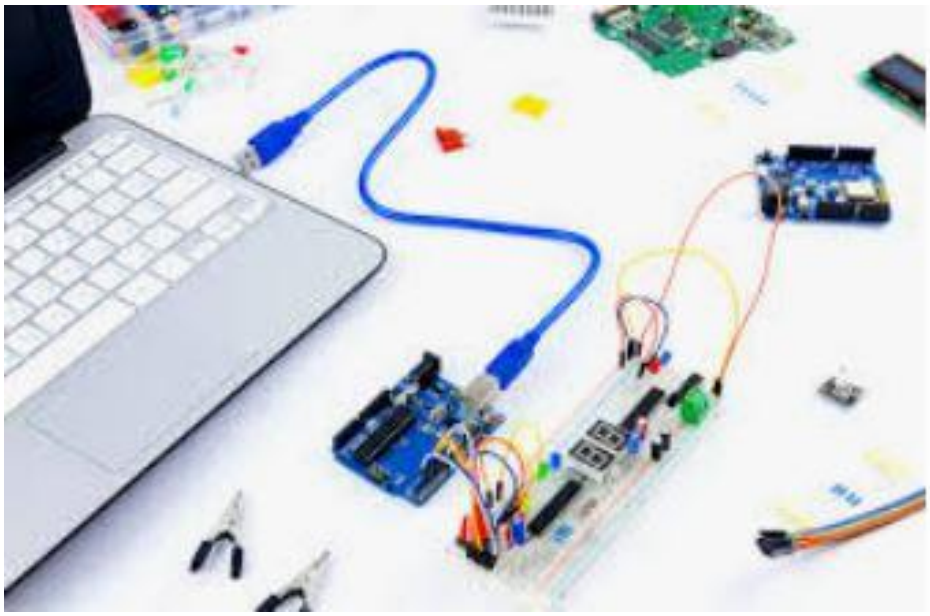
IOT modul Part 2

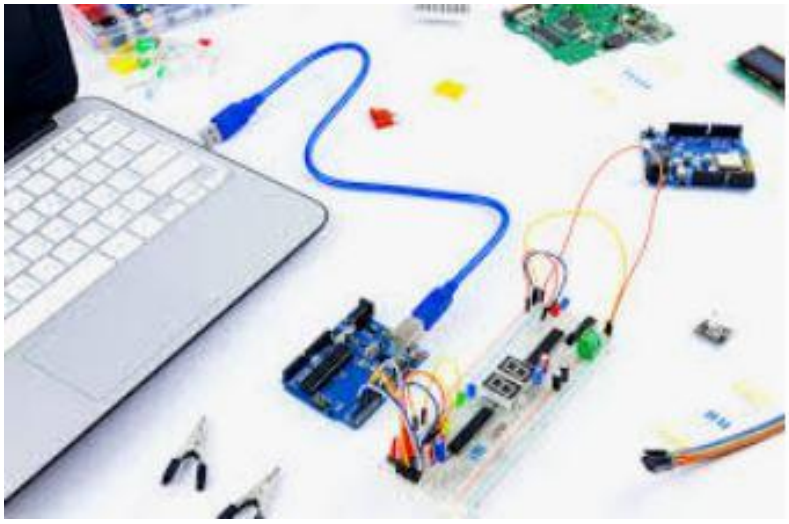


Indhold

HVORFOR ER DEBUGGING SÅ VIGTIG ???

I et embedded system er det meget vigtigt, at man kan debugge så mange af de enkelte delkomponenter, der indgår i et sådant system hver for sig, da man derved kan eliminere en eller flere af de mange fejlmuligheder, der kan være i et embedded system.

I Figur 1 herunder ses et typisk eksempel på et embedded system:



Figur 1 : Typisk Embedded System

Hvad kan vi have af fejlmuligheder her?

1) Fejl i vores embeddede SW (som er her hvor den overvejende del af fejl indenfor den embeddede verden er/laves)

2) Fejl i vores microcontroller (på billedet er det vist en Atmel microcontroller, der er vist)

3) Fejl i tilkoblet HW (Lysdioder, Displays og så videre)

4) Dårlige ledninger

5) Fejl i det/de benyttede Breadboard(s)

6) Timings fejl

Disse er kun et udsnit af de mange fejlmuligheder, vi ser ind i. Så derfor er det ekstremt vigtigt, at vi har de rette værktøjer til at hjælpe os godt på ved, når vi skal fejlsøge på vores embeddede system.

I det efterfølgende vil vi i første se på, hvordan vi sætter debugging op, så vi kan debugge/fejlrette i vores eget udviklede embeddede SW. Dette vil være en rundtur i forskellige debuggere, forskellige IDE'ere og forskellige microcontrollere. Her i skrivende stund (forår 2022) arbejder vi på Tech College med de nedenfor nævnte microcontrollere:

1) AtMega168 (Freeduino boards)
2) AtMeaga328 (Arduino Uno – Arduino Nano boards)
3) AtMega2560 (boards med indbygget JTAG-debugger)
4) NodeMCU (Esp8266 boards)
5) Arduino Zero (Arm chip baseret)

Debug på hver enkelt af de ovenfor nævnte microcontrollere, vil blive behandlet i de efterfølgende afsnit. I hvert/flere af disse afsnit, vil der være underafsnit, som beskriver debug på den aktuelle micro i hvert enkelt udviklingsmiljø, som vi gør brug af. Her i foråret 2022 ser denne liste ud som vist herunder. Denne liste vil utvivlsomt ændre sig som tiden går.

1) Microchip Studio (afløseren til det tidligere Atmel Studio)
2) Arduino PRO IDE
3) Visual Micro (Visual Micro fås som et ekstensionsmodul til både Visual Studio og til Microchip Studio)
4) Visual Studio Code

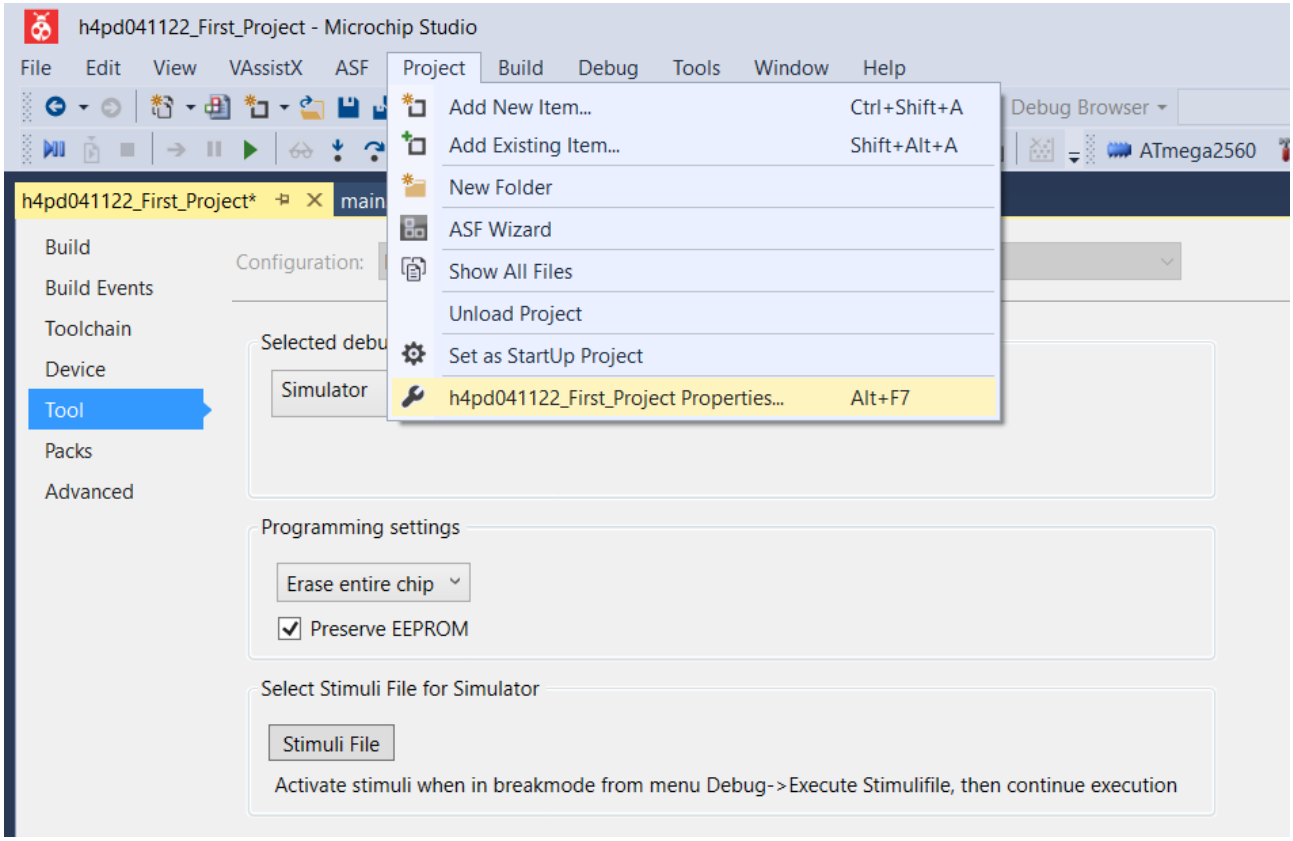
Grunden til at Microchip Studio er nævnt først i listen herover, er den, at når man udvikler i Microchip Studio, så er man (typisk) på et mere Hardware nært niveau, end når man udvikler i de 3 andre nævnte værktøjer. Man kan også sige, at Microchip Studio er et værktøj, der er tiltænkt at blive brugt af de mere professionelle Embeddede SW-udviklere, mens de andre værktøjer typisk vil blive anvendt af udviklere på et mere "amatør baseret" niveau.

Før vi går igen med at se på Embedded Debugging vil vi dog lige se på Embedded Simulering af SW først.
Når vi snakker om Embedded Debugging, kører vores Embeddede SW på den faktiske HW. Det vil sige, at hvis vi for eksempel har lavet et program til en AtMega2560 micro og ønsker at debugge dette, så skal vi ved brug af en IDE have kontakt til vores HW board og så bruge de indbyggede debugging muligheder, der er i vores IDE.
Proceduren er anderledes, når vi "blot" ønsker at simulere vores embeddede SW. Dette kan/vil vi typisk gøre på embedded kode, der ikke er afhængig af input fra den "ydre verden" som f.eks. et eksternt interrupt ved tryk på en knap, timer interrupt og lignende HW nær kode. I Microchip Studio er der en glimrende SW Simulator indbygget, som kan hjælpe os langt hen ad vejen, uanset hvilken af de nævnte microcontrollere vi arbejder på. Når vi ønsker at køre en simulering på vores embeddede SW, behøver vi ikke have noget microcontroller board forbundet til vores PC. Microchip Studio sætter det hele op for os afhængig ad den micro vi vælger (se SOFTWARE Simulering afsnittet).
Når vi "blot" simulerer vores Embeddede SW, taler vi om Embedded Simulering og ikke Embedded Debugging.

SOFTWARE SIMULERING

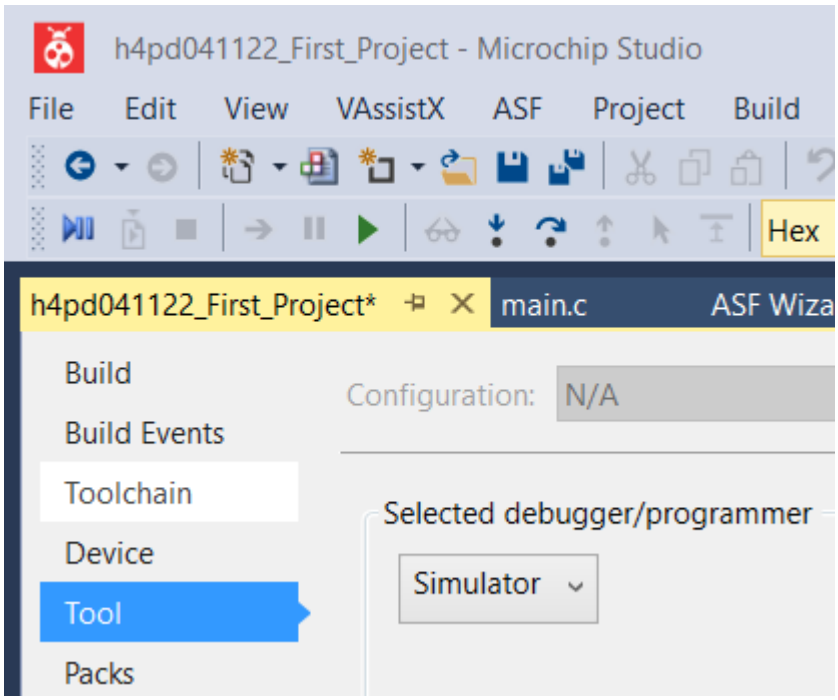
Som nævnt i sidste afsnit er der en glimrende SW Simulator indbygget i Microchip Studio.

Denne SW Simulator er let at få op at køre. Det man i praksis gør er, at man går ind i menupunktet Project\"Projekt Navn" Properties som vist i Figur 2.



Figur 2 : Opsætning af SW Simulator i Microchip Studio

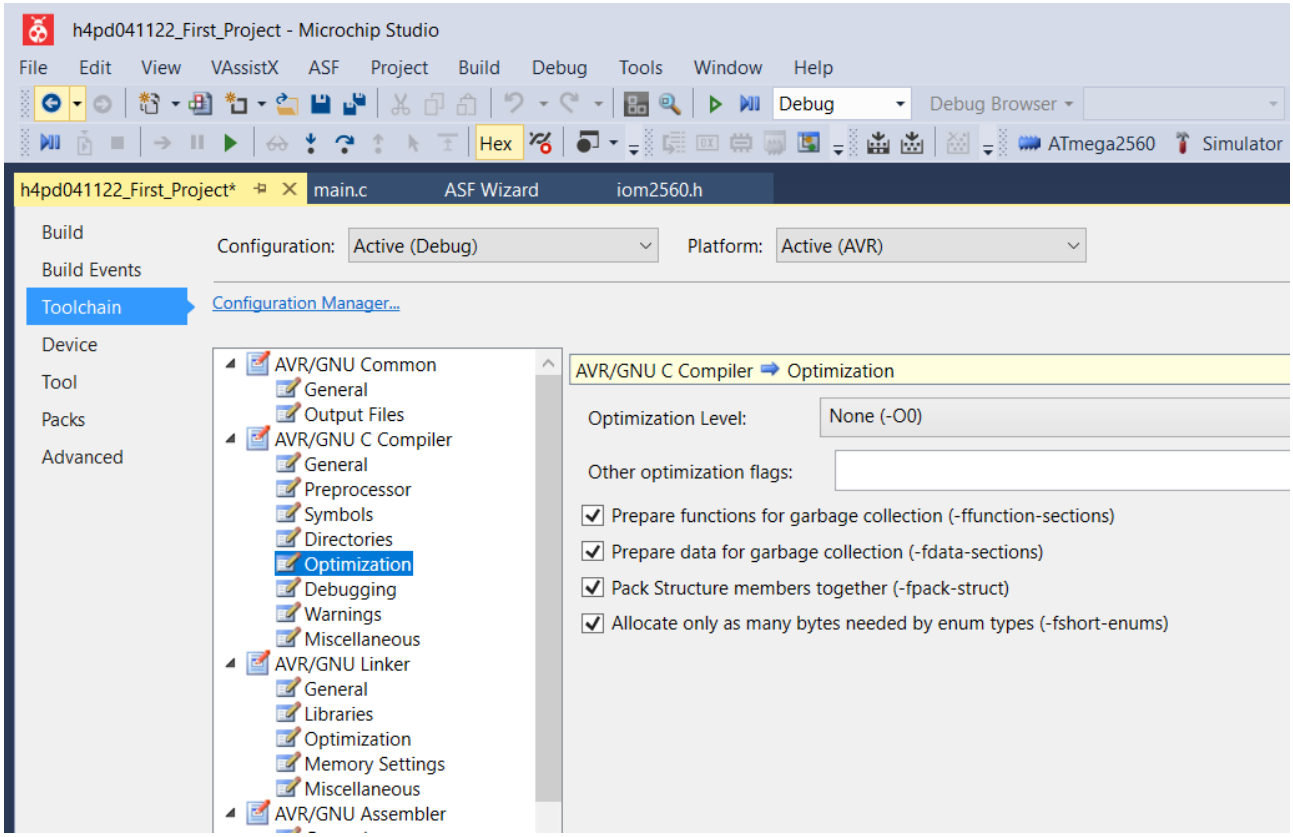
Når man er inde under dette menupunkt, vælger man punktet **Tool** og når dette punkt er valgt, vælger man under Selected debugger/programmer Simulator som mulighed.



Figur 3 : Valg af Simulator som SW Debug tool

Før vi får den fulde gode debug oplevelse, når vi udfører Embedded Simulering, er der lige en detalje vi skal være opmærksom på. Vi skal vælge at compilere vores kode uden SW-optimering. Når man laver SW-optimering på sin embeddede kode, vil den indbyggede Simulator (og faktisk også den indbyggede emulator -> se efterfølgende afsnit om Embedded Debugging i Microchip Studio), let kunne blive forvirret og så kommer man typisk i en situation, hvor linjenumre og andet kan komme ud af synkronisering.

Måden vi undgår at vores embeddede SW bliver udsat for optimering under compileringsfasen, er meget let. Vi går igen ind under menupunktet Project\"Projekt Navn" Properties som vist i Figur 2. Nu vælger vi så punktet **Toolchain** og kommer herefter frem til situationen, som vist i Figur 4. Under AVR/GNU C Compiler vælger vi Optimization Level og sætter dette til: **None (-O0)**.

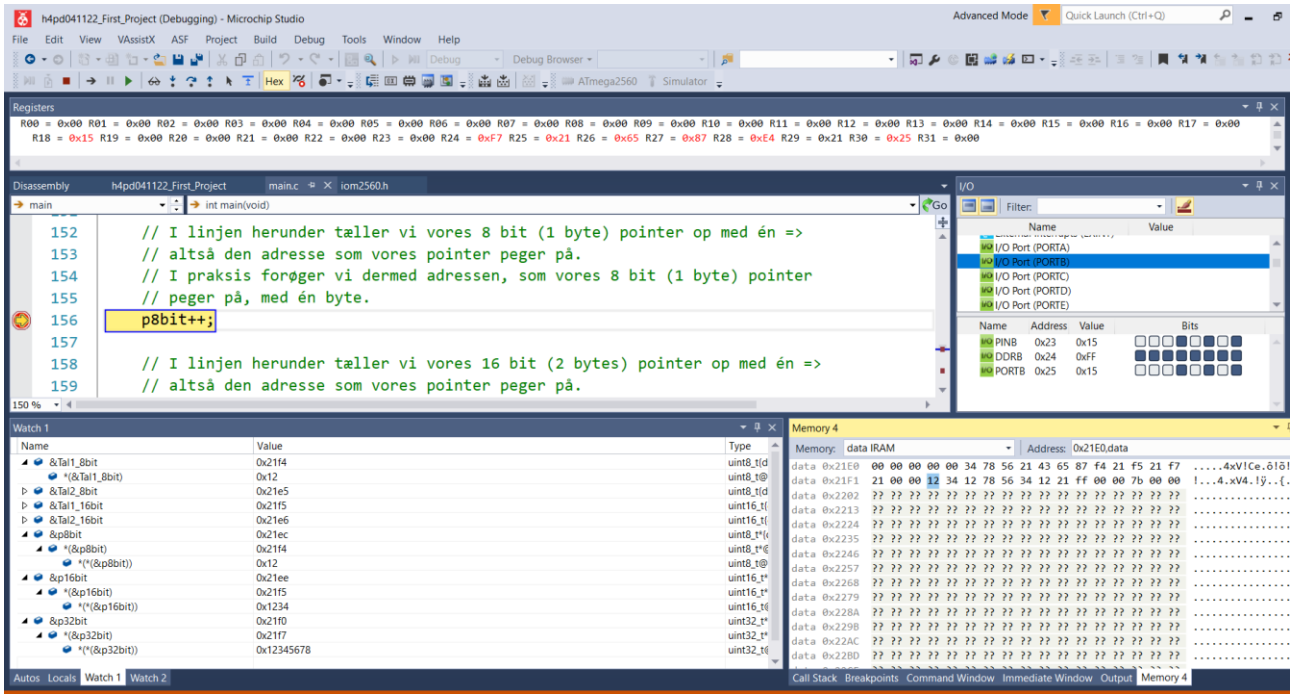


Figur 4 : Opsætning af Optimization level for vores projekt

Proceduren beskrevet her for Software Simulering er helt den samme, vi skal udføre, når vi taler om Embedded Debugging.

Når vi har udført de her beskrevne steps har vi i Microchip Studio en meget fin fungerende debugger, der er helt/næsten på højde med den debugger, vi kender fra for eksempel Visual Studio. Debuggeren i Microchip Studio er dog mere HW nær end debuggeren i Visual Studio, så vi kan let se på ting som memory områder, porte, registre og så videre.

I Figur 5 er der vist et typisk eksempel på de vinduer, man kan sætte op i Microchip Studio, når man udfører en Embedded Simulering session. Det vil typisk også være de samme vinduer, man sætter op, når man udfører en Embedded Debugging session.



Figur 5 : Typisk opsætning af vinduer i Microchip Studio under Embedded Simulering session

ATMEGA168 (FREEDUINO BOARDS)

AtMega168 Embedded Debugging i Microchip Studio

Da der ikke er nogen debugger (JTAG-debugger eller andet) indbygget på et AtMega168 board, kan vi **ikke** lave Embedded Debugging på et sådant board i Microchip Studio.

ATMEAGA328 (ARDUINO UNO – ARDUINO NANO BOARDS)

AtMega328 Embedded Debugging I Microchip Studio

Da der ikke er nogen debugger (JTAG-debugger eller andet) indbygget på et AtMega328 board, kan vi **ikke** lave Embedded Debugging på et sådant board i Microchip Studio.

ATMEGA2560 (BOARDS MED INDBYGGET JTAG-DEBUGGER)

Som nævnt i overskriften har en AtMega2560 mikro en indbygget JTAG-debugger. Dette giver den umiddelbart nogle helt klare fordele mht. debugging i forhold til AtMega168 – og AtMega328 microer. I hvert fald når vi anvender Microchip Studio som vores udviklingsværktøj. Når de andre udviklings IDE'er anvendes, er der ikke den store forskel på en AtMega2560 micro i forhold til AtMage168, AtMega328 og NodeMCU microer.

AtMega2560 Embedded Debugging i Microchip Studio

Når man skal udføre Embedded Debugging i Microchip Studio på et specifikt AtMega2560 board, er der en række opsætninger, der skal være opfyldt, før man kommer dertil, at man kan lave Embedded Debugging på samme niveau, som man kan lave Embedded Simulering (se Figur 5). Fordelen er så, at når vi har udført de forskellige opsætninger nævnt i det efterfølgende, så laver vi Embedded Debugging og ikke Embedded Simulering. Vores Embeddede SW kører altså på den faktiske microcontroller og ikke på en simuleret version i Microchip Studio af denne.

Forudsætning 1: Vi skal have en Atmel SAM AVR-boks tilgængelig. Det er den boks, der indeholder HW (og SW) og som muliggør, at vi ved brug af Microchip Studio kan udføre Embedded Debugging. Denne boks er vist herunder i Figur 6.

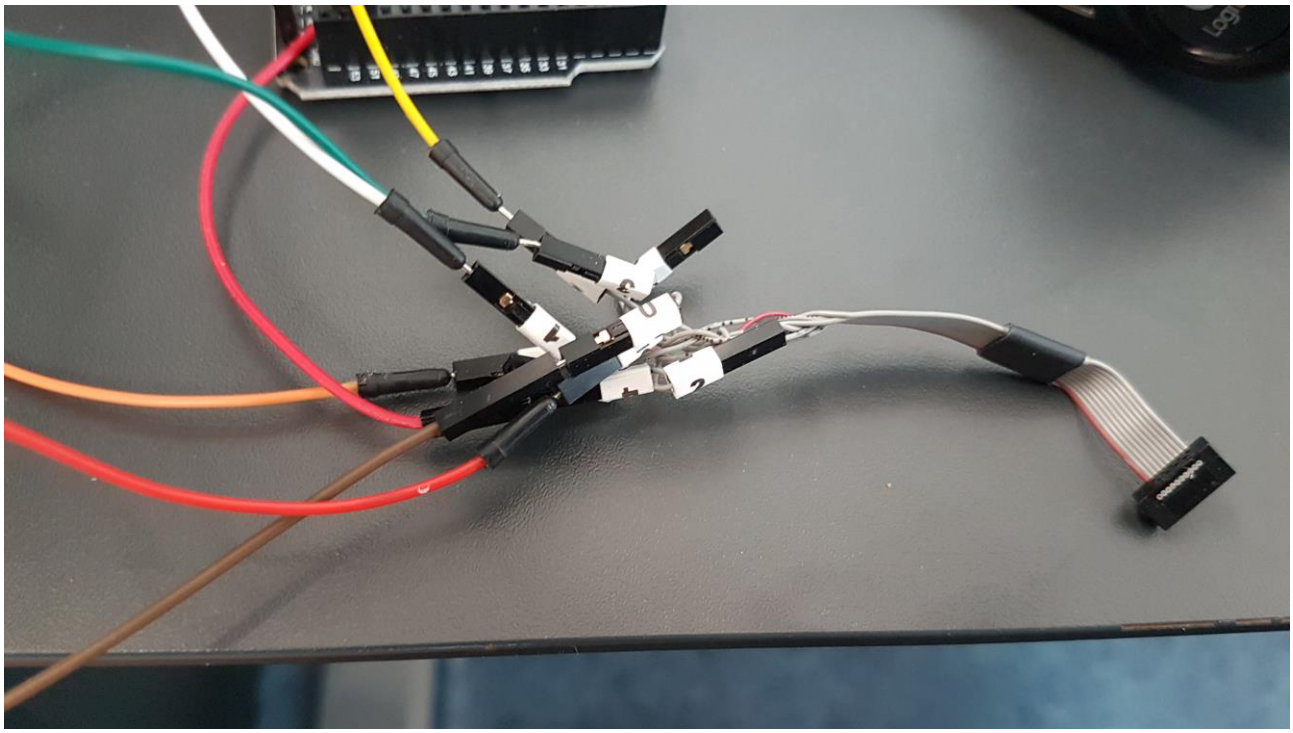
Figur 6 : Atmel SAM AVR JTAG-Debugger boks

Forudsætning 2: Det første vi skal have udført er, at vi skal have ændre Fuse settings for det specifikke AtMega2560 board, vi arbejder på lige nu. For at få adgang til Fuse settings på vores

specifikke AtMega2560 board, skal vi anvende **ISP** interfacet fra vores Atmel SAM AVR-boks. Vores Atmel SAM AVR-boks har 2 interfaces tilgængelig. Udover det allerede nævnte **ISP-interface** har den også **JTAG-interface**. Men for at kunne benytte JTAG-interfacet kræver det, at vi har ændret i nogle Fuse settings på vores specifikke AtMega2560 board. Og det er vi nødt til at gøre med ISP interfacet!!!

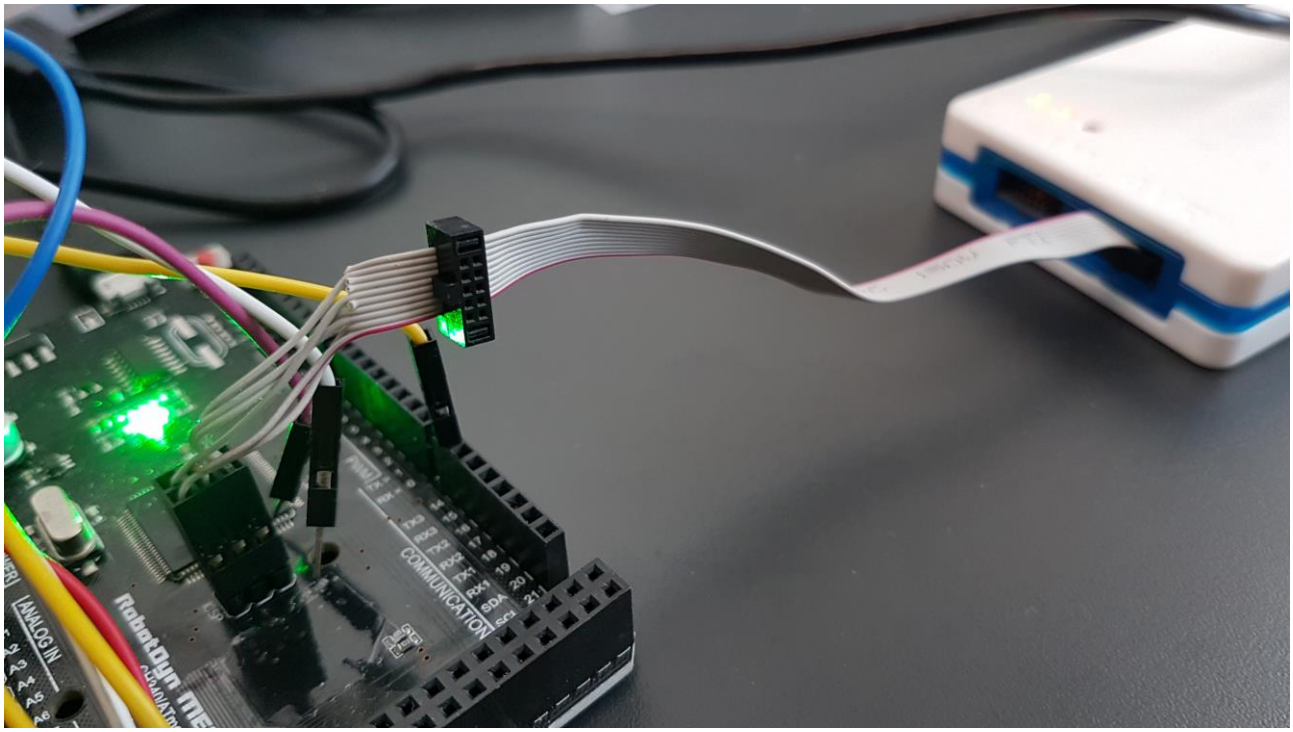
Hvis man køber en Atmel SAM AVR-boks med alt ting inkluderet, kommer denne følgelig også med kabler til både at kunne forbinde med target via ISP-interface og via JTAG-interface. Figurerne herunder viser ISP – henhold JTAG-interface kablerne.

Figur 7 : ISP-interface kabel



Figur 8 : JTAG-interface kabel

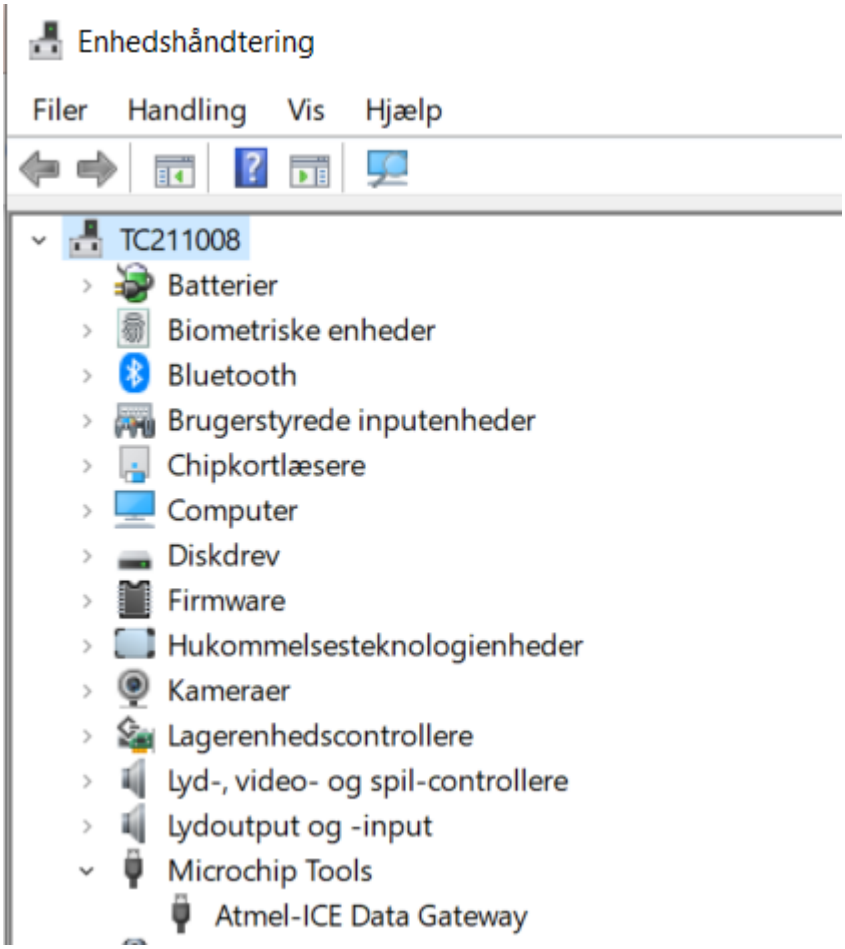
Når man skal forbinde sin Atmel SAM AVR-boks til sit AtMega2560 boards ved brug af ISP-interface kablet, skal det forbindes som vist herunder i Figur 9.



Figur 9 : Forbindelse fra Atmel SAM AVR-boks til AtMegas2560 board ved brug af ISP-interface

Som det af Figur 9, skal ISP-interface kablet forbindes til det højre side på Atmel SAM AVR-boksen, når man ser ind på denne fra fronten. Og den anden ende af kablet skal forbindes til I(C)SP pins på AtMega2560 boardet, sådan at hakket i dette stik vender ind mod midten af AtMega2560 boardet. **Når man bruger ISP-interfacet fra vores Atmel SAM AVR-boks, kan man forbinde til alle Atmel<xx> boards**.

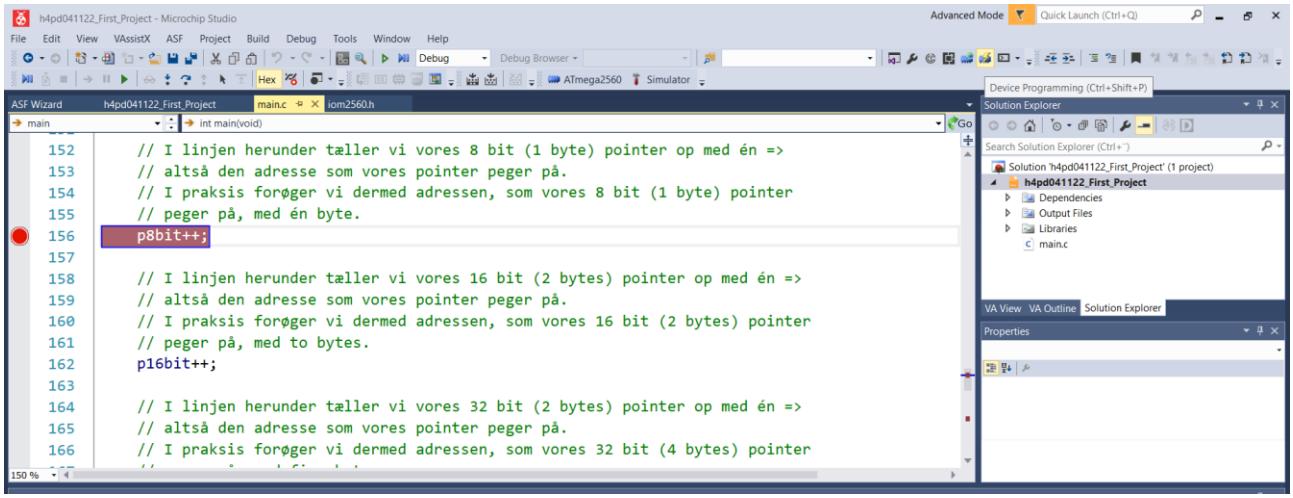
Uanset om man anvender ISP-interface kablet, eller man anvender JTAG-interface kablet til at forbinde Atmel SAM AVR-boksen til sit AtMega2560 board, skal man selvfølgelig også have forbundet denne til sin PC via det medfølgende USB-kabel. Når man gør dette, skal man gerne kunne se sin Atmel SAM AVR-boks i enhedshåndtering på sin PC som vist i Figur 10.



Figur 10 : Atmel SAM AVR-boks detekteret på PC

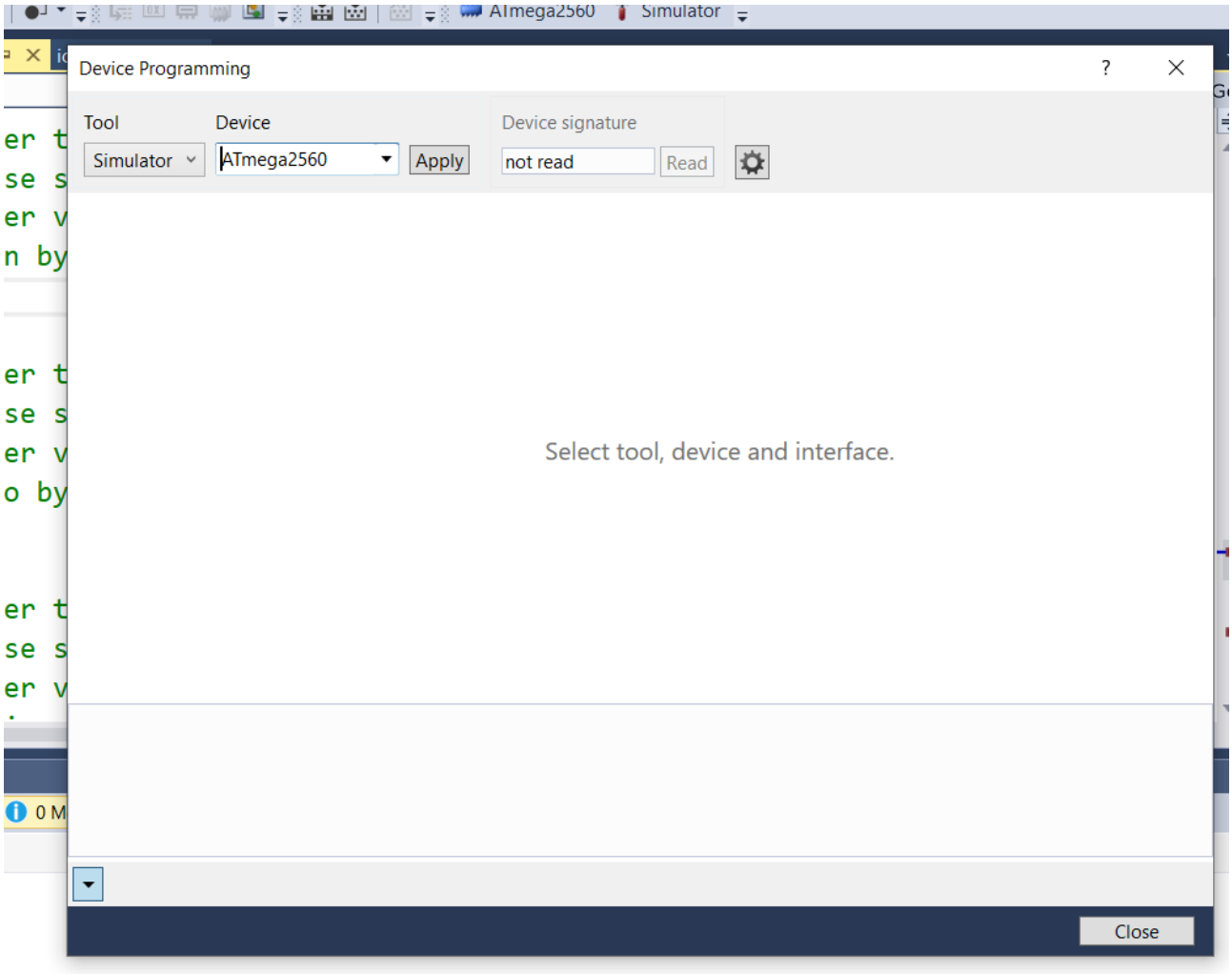
Driveren til Atmel SAM AVR-boksen burde være med som standard på enhver Windows 10 PC. Oplever man problemer, når man forbinder boksen til sin PC, kan det dog godt være, at man selv skal ud og finde en driver til boksen.

Med ISP-interface kablet forbundet er vi nu klar til at ændre Fuse settings for vores specifikke AtMega2560 board. Vi starter Microchip Studio op og trykker på Device Programming ikonet som vist i Figur 11.



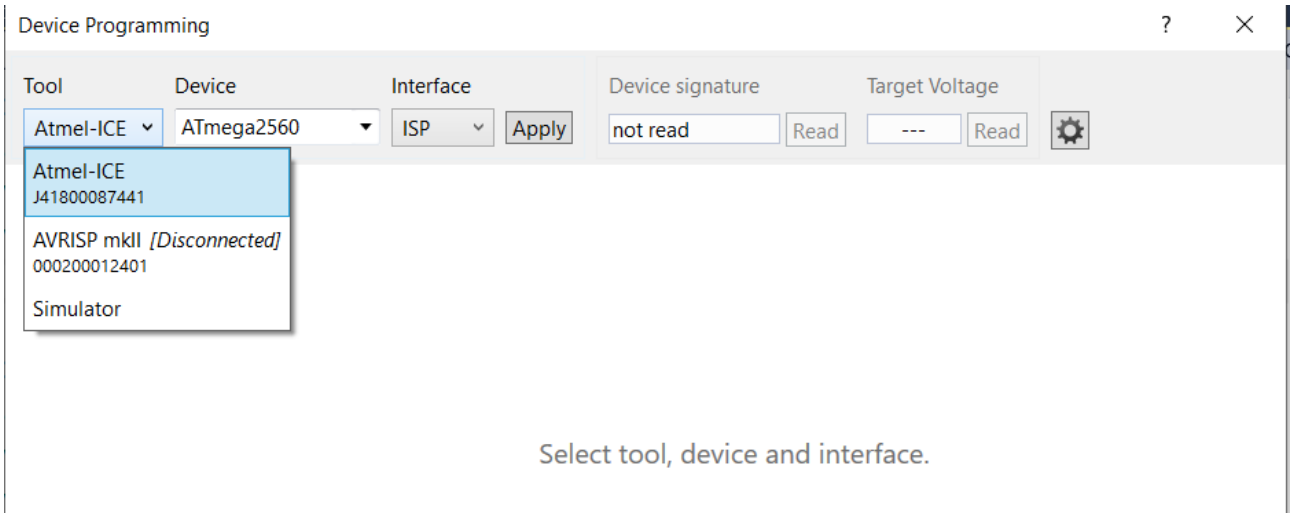
Figur 11 : Device Programming ikon fremhævet

Trykker vi nu på det fremhævede Device Programming ikon, kommer billedet i Figur 12 eller et lignende billede typisk frem. I hvert fald hvis det er første gang, man har dette menupunkt åben på sin PC.

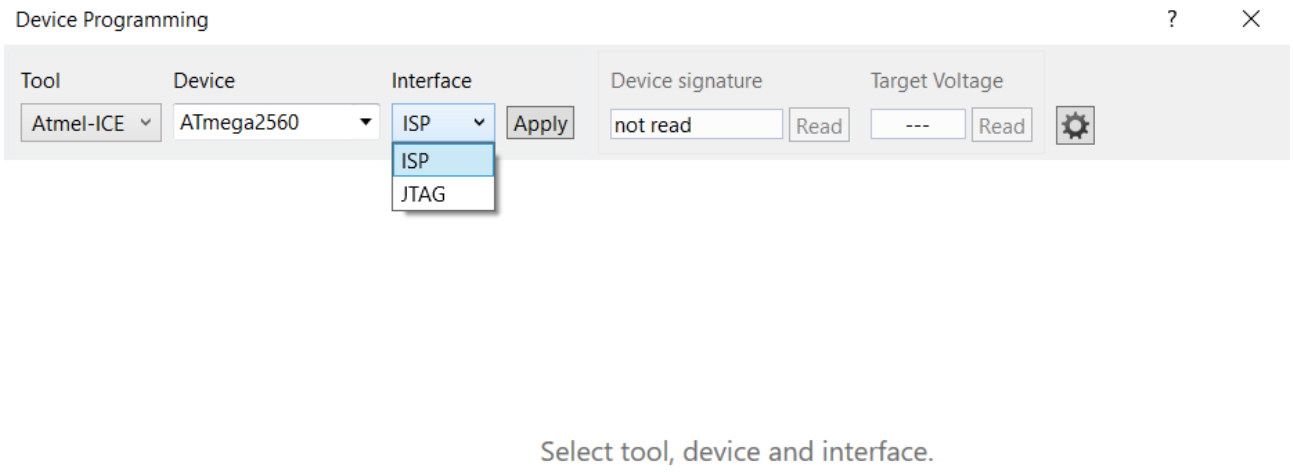


Figur 12 : Programming Device vindue i Microchip Studio

Fra situationen i Figur 12 trykkes nu på pilen under Tool punktet (til højre for Simulator i Figur 12). Herved får vi mulighed for at vælge Atmel SAM AVR-boksen, som vist i Figur 13. I Microchip Studio er betegnelsen for vores Atmel SAM AVR-boks -> Atmel-ICE!!!
Når vi har valgt Atmel-ICE, kommer der en drop down boks op til højre for Device punktet. Når vi har tilkoblet vores Atmel SAM AVR-boks, har vi her mulighed for at vælge 2 interfaces, nemlig de føromtalte **ISP** og **JTAG**, som det ses i Figur 14. Her er det vigtigt, at vi første omgang vælger ISP-interfacet, da vi som tidligere nævnt, først skal have ændret Fuse-settings i vores specifikke AtMega2560 board for at vi kan anvende JTAG-interfacet.

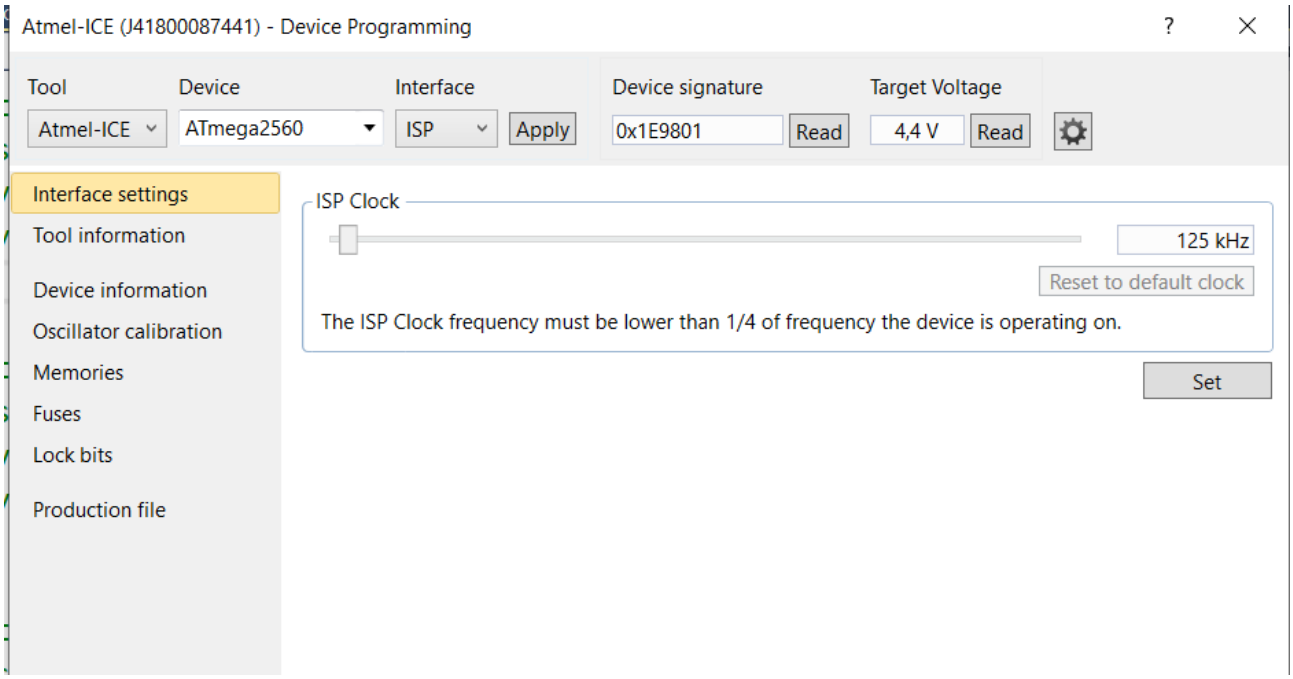


Figur 13 : Valg af Atmel SAM AVR-boks



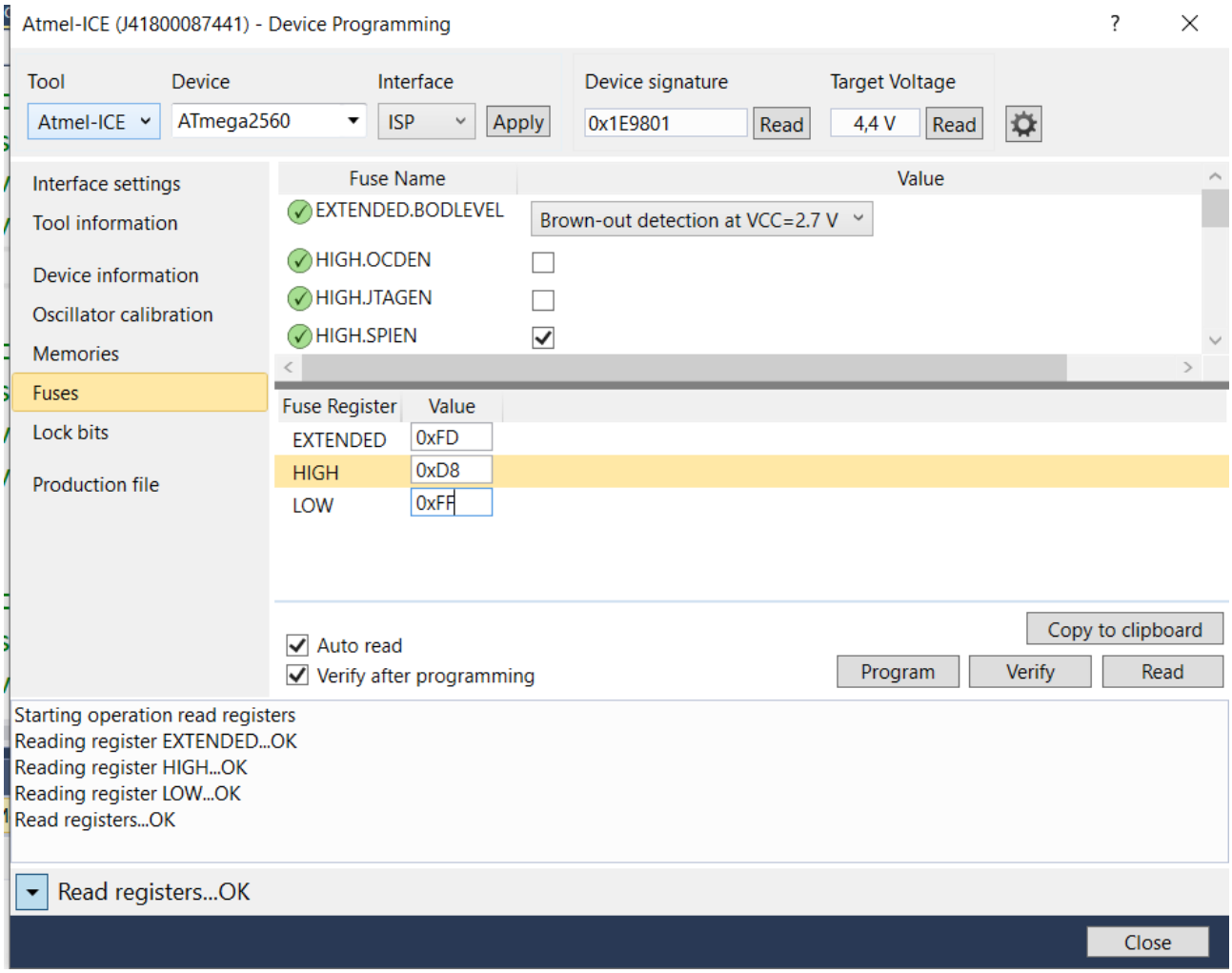
Figur 14 : Valg af ISP-interface

Når vi har valgt ISP-interfacet, kan vi trykke på Apply knappen og Read knappen, der begge bliver aktive efter vores valg af ISP-interface. Når vi har gjort dette, skal vi gerne kunne se et skærmbillede, lignende det der er vist i Figur 15.



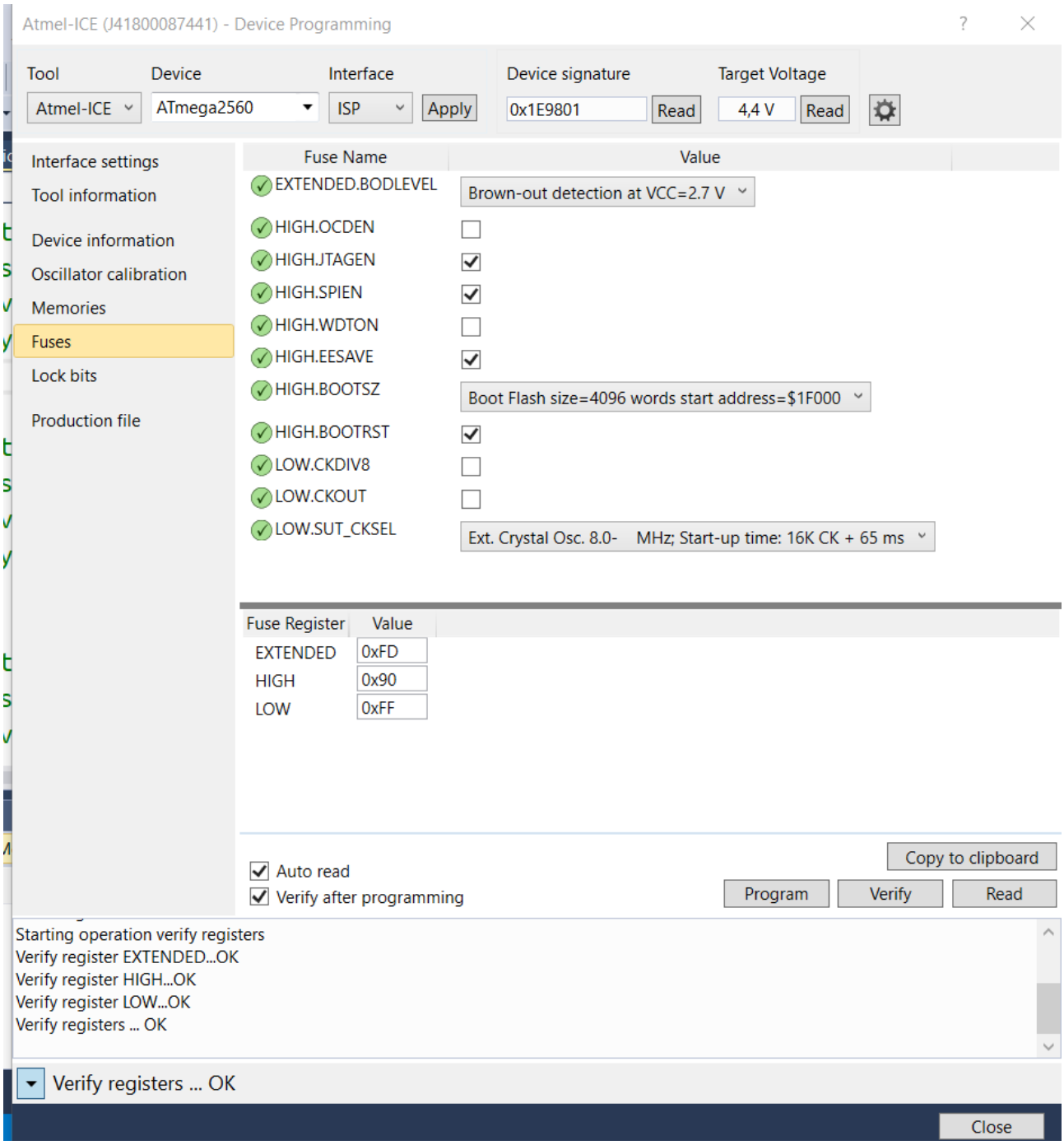
Figur 15 : Device Programming interface efter valg af interface og tryk på Apply og Read knapper

Hvis vi nu går ned under punktet Fuses, får vi følgelig adgang til Fuse bits for vores AtMega2560 board. Disse er nærmere beskrevet på side i 336 i databladet for AtMega2560 microen. Når man ikke har programmeret/ændret Fuse bits på sin AtMega2560 micro, vil værdien af de 3 FUSE bytes være som følger: **EXTENDED (0xFD), HIGH (0xD8) og LOW (0xFF)**. Dette er også vist i Figur 16. Det er en rigtig god ide at huske på disse initiale værdier, hvis man på et senere tidspunkt har behov for at få sat en AtMega2560 micro tilbage til dens initiale tilstand.



Figur 16 : Initielle værdier af Fuse bits (Fuse bytes) for en AtMega2560 micro

For at få adgang til at kunne benytte JTAG-interface fra vores Atmel SAM AVR-boks til vores AtMega2560board, skal vi have programmeret Fuse bytes/bits som vist i Figur 17.



Figur 17 : Fuse bits/bytes indstillinger for enabling af JTAG debugging

Når I skal gøre det på jeres eget AtMega2560 board, kan I vælge at klikke på hver enkelt Fuse bit, eller også kan I skrive de viste værdier fra Figur 17 direkte ind i de 3 felter, EXTENDED, HIGH og LOW, der er de 3 Fuse bytes. Jeg har fundet fuse bits/bytes opsætningen fra dette link: http://ww1.microchip.com/downloads/en/DeviceDoc/DOC2475.PDF

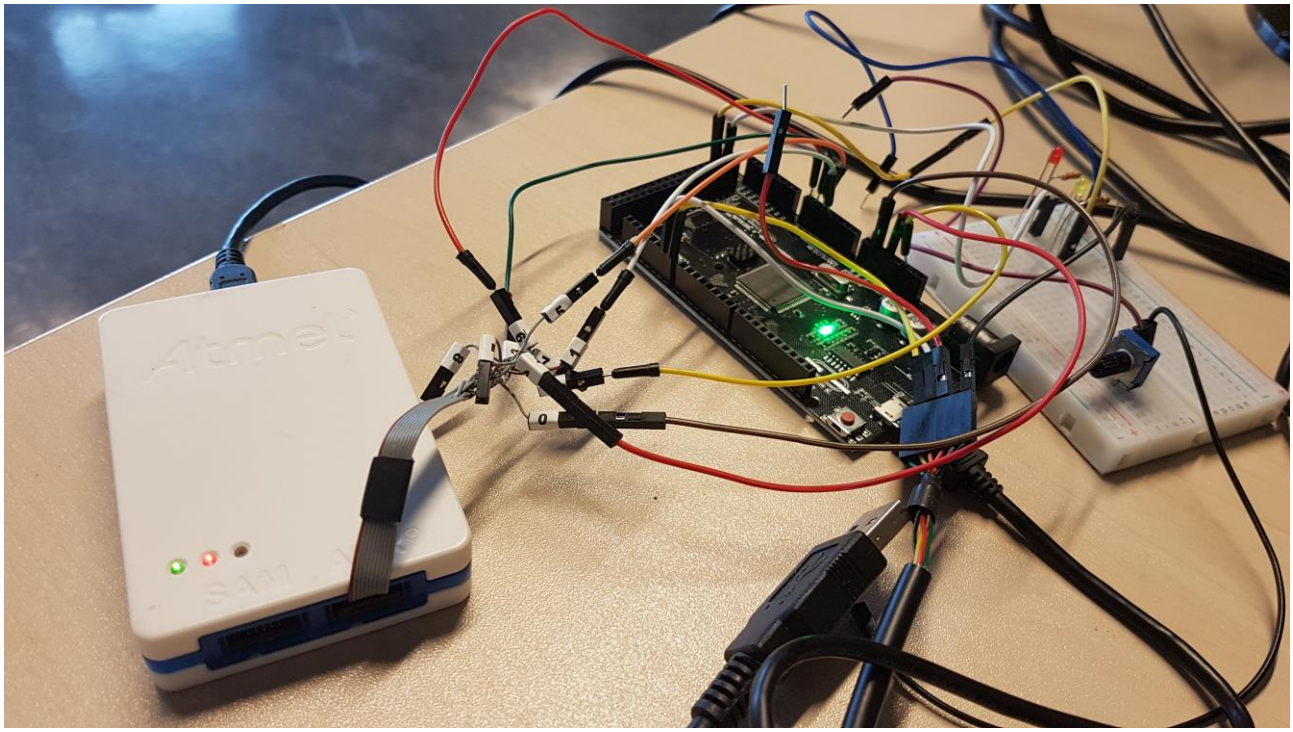
Når I har sat jeres Fuse bits/bytes som vist i Figur 17, trykker I på **Program** for at få disse værdier skrevet ned til jeres AtMega2560 board.

Efter download af de ændrede Fuse bits/bytes værdier, kan I skifte jeres ISP-interface kabel ud med jeres JTAG-interface kabel (se Figur 8). Som det fremgår af Figur 8, ender JTAG-interface kablet ud i 10 nummererede enkelt kabler. Disse kabler skal forbindes til jeres ATMega2560 micro i henhold til Figur 18.

With this we can see that
JTAGPIN9->TDI->ADC7
JTAGPIN3->TDO->ADC6
JTAGPIN5->TMS->ADC5
JTAGPIN1->TCK->ADC4
JTAGPIN10->ARDUINO GRND
JTAGPIN6->ARDUINO RESET
JTAGPIN4->ARDUINO 5V
Other Pins Not Needed

Figur 18 : Forbindelse af JTAG-interface kabel til AtMega2560 micro/board

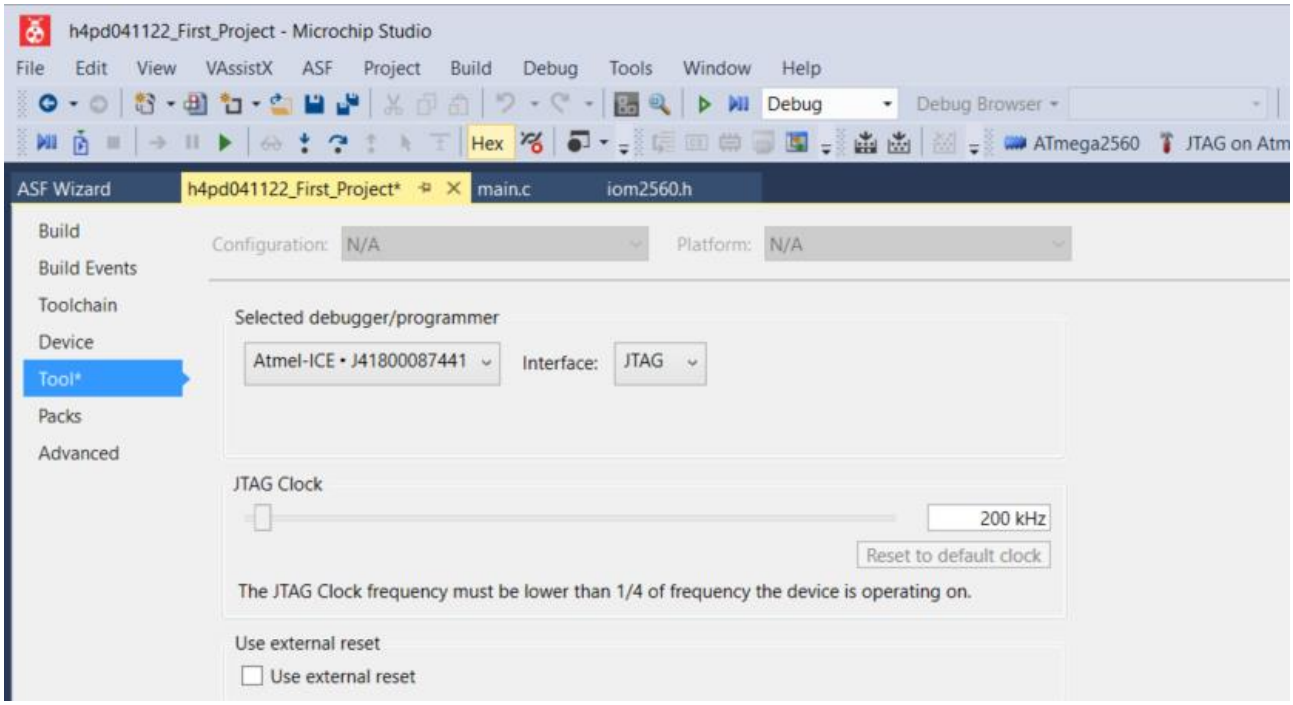
Som det fremgår af Figur 19, skal JTAG-interface kablet, som tilfældet var med ISP-interface kablet, også forbindes til det højre stik på Atmel SAM AVR-boksen.



Figur 19 : Forbindelse af JTAG-interface kabel til AtMega2560 micro/board og til Atmel SAM AVR-boks

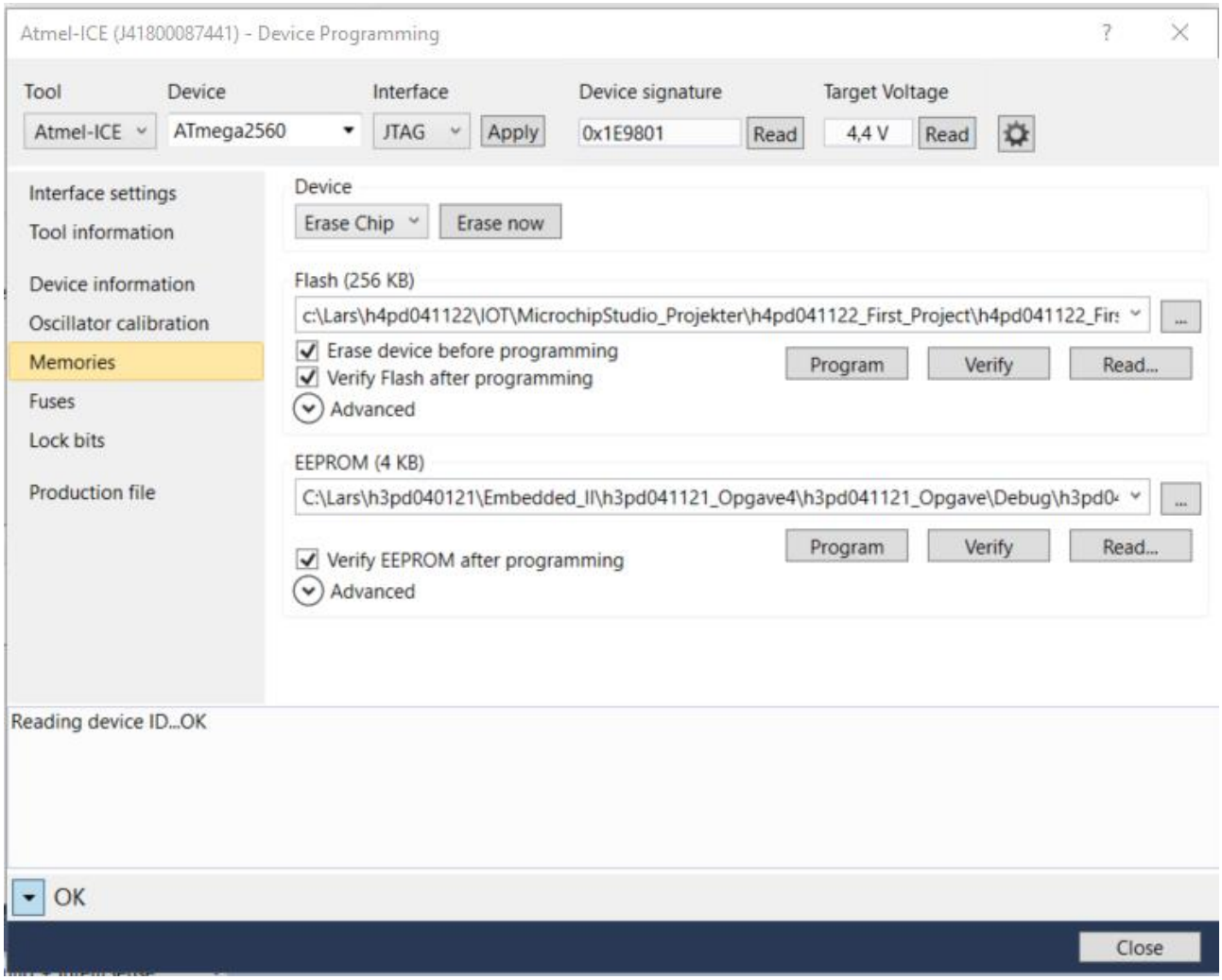
Når vi nu er nået hertil, er vi stort set klar til at udføre Embedded Debugging. Vi skal dog lige have ændret et par ting i vores Microchip Studio projekt først. Vi tager her igen udgangspunkt i det samme projekt, som vi anvendte, da vi snakkede om Embedded Simulering.

I Microchip Studio går vi igen ind under menupunktet: Project\"Projekt Navn" og vælger igen Tools her (se også Figur 3). Men nu vælger vi i stedet for Simulator at anvende Atmel-ICE og når denne er valgt, vælger vi JTAG som interface som vist i herunder i Figur 20.



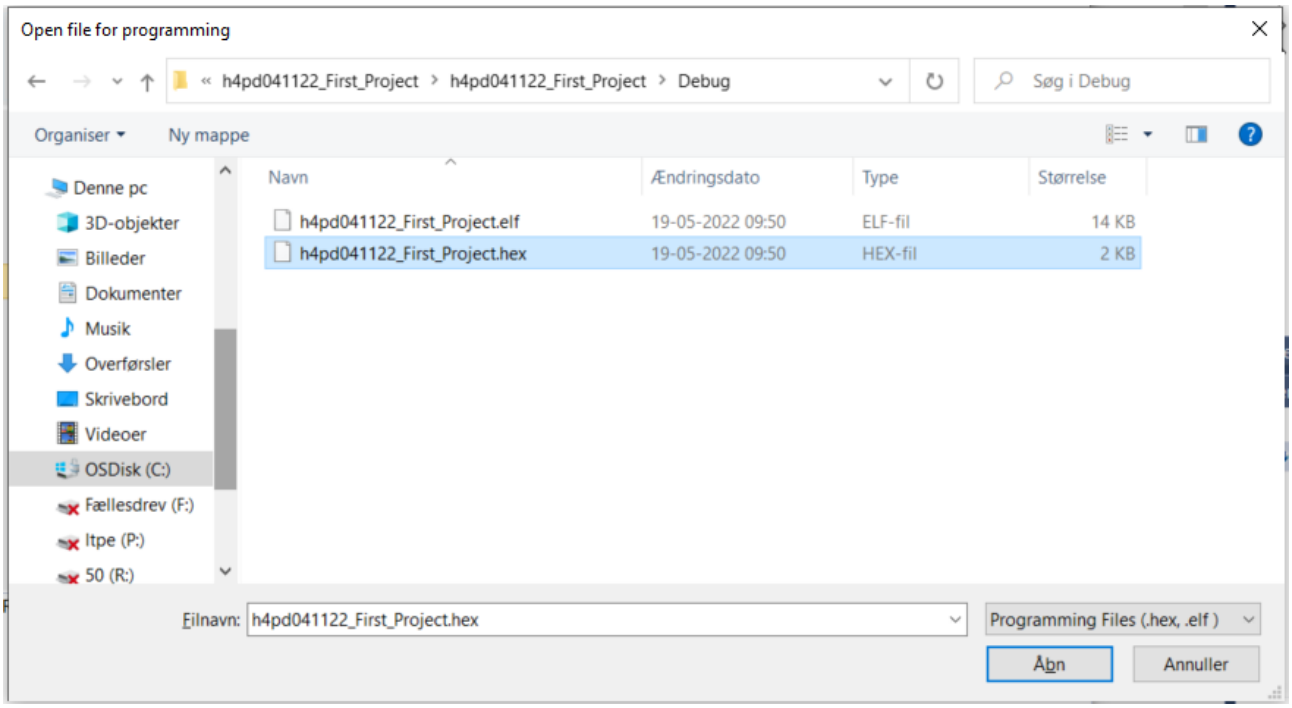
Figur 20 : Valg af Atmel-ICE JTAG som Debugger

Det allersidste vi mangler, er at få downloaded vores kode til vores AtMega2560 board. Vi starter med at bygge vores projekt, som vi kender det fra Visual Studio. Når dette er gjort, trykker vi igen på Device Programming ikonet og går gennem de samme steps som vist i Figur 13 - Figur 15. Den eneste forskel er, at vi nu vælger JTAG som interface i stedet for ISP som vist i Figur 21.



Figur 21 : Valg af JTAG som interface under Device Programming

Som det fremgår af Figur 21, er det nu punktet **Memories**, der er valgt. Og dette punkt er valgt, da det er her, vi skal være, når vi skal downloade den compilerede kode til Flash Prom i vores AtMega2560 board.
Vi vælger her den fil, der skal downloades under Flash teksten. Denne fil vil typisk være en *.hex fil, som befinder sig i Debug direktoriet for vores nuværende projekt. Dette er vist i Figur 22.



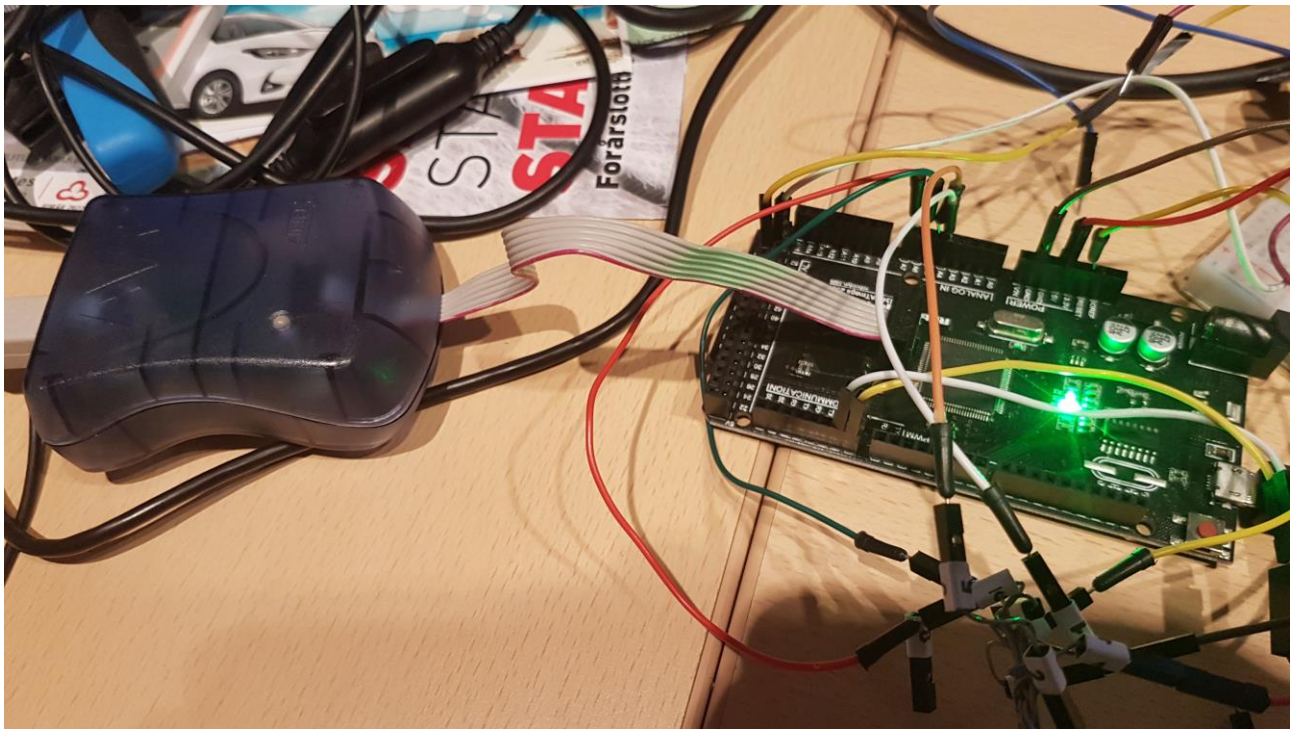
Figur 22 : Valg af *.hex fil til Download

Når dette er gjort, er vi klar til at foretage Embedded Debugging helt på samme måde som beskrevet under Software Simulering. Den eneste og selvfølgelig ikke uvæsentlige forskel er, at vi nu debugger på kode, der kører i vores AtMega2560 target og ikke kode, som bliver simuleret i vores Microchip Studio værktøj.

Vi skal lige være opmærksom på en meget detalje her i forbindelse med download af SW til vores AtMega2560 bord ved brug af Microchip Studio som IDE og vores Atmel SAM AVR-boks som download boks/JTAG-debugger. Og det er, at når vi downloader SW på denne måde, vil vi få overskrevet den bootloader SW, der er programmet i Flash Prom på et AtMega2560 board, når dette forlader produktionsfabrikken. ***Dette betyder, at vi så ikke (umiddelbart) kan downloade SW til vores specifikke AtMega2560 board, når vi arbejder i ethvert af de andre beskrevne IDE'er beskrevet i dette dokument.***
Ønsker vi at arbejde i en af de andre beskrevne IDE'er med et AtMega2560 board, hvor der har været downloaded SW til ved brug af Atmel SAM AVR-boksen i Microchip Studio, er vi nødt til at downloade bootloader SW til Flash Prom igen i vores AtMega2560 board. Dette gør vi på samme måde, som vi downloader eget udviklet SW i Microchip Studio. Den eneste forskel her er, at vi så skal finde en passende bootloader fil, der passer til det specifikke board vi arbejder på.
I direktoriet Bootloader findes der bootloader filer for de forskellige AtMega<xx> boards beskrevet i dette dokument. For "problemet" med overskrivning af den initiale bootloader i Flash Prom på vores target, ***gør sig gældende for ALLE de AtMega boards, hvor der er downloaded SW til ved brug af Microchip Studio og en downloadboks i stil med den her beskrevne Atmel SAM AVR-boks.***

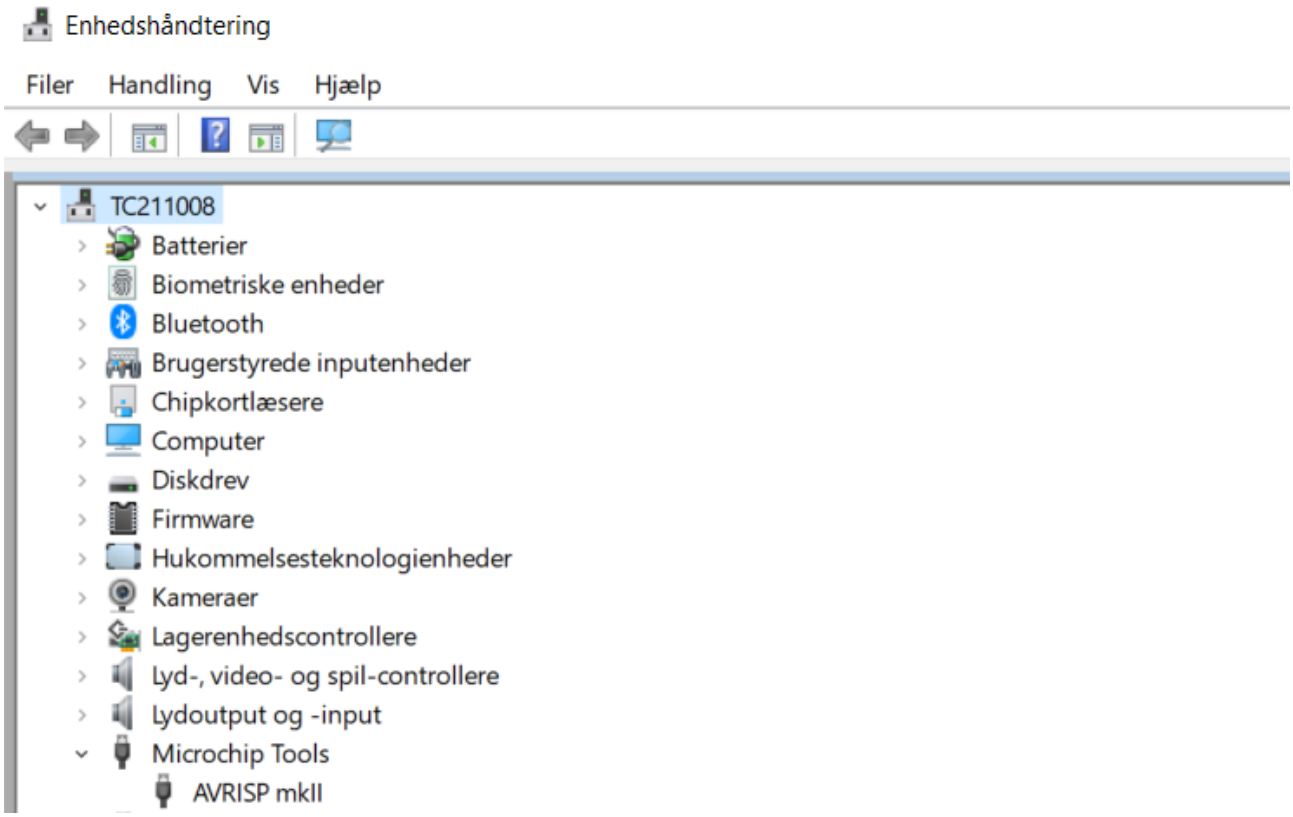
Skal måske også her nævne, at de blå Atmel mkII bokse I har fået udleveret, kan gøre det samme som en Atmel SAM AVR-boks kan, når denne kører i ISP-interface mode. I Figur 23 er det vist,

hvordan man kobler den blå Atmel mkII boks til et AtMega<xx> board. Det er med fuldt overlæg, at der er skrevet et AtMega<xx> board, da den blå Atmel mkII boks kan kobles til alle slags AtMega<xx> boards. Dette samme gør sig jo også gældende, som nævnt tidligere, for Atmel SAM AVR-boksene.



Figur 23 : Tilkobling af blå Atmel mkII boks til AtMega<xx> board

Når kobler en blå Atmel mkII boks til sin PC ved brug af et USB-kabel, skal man gerne kunne se sin boks i enhedshåndtering på sin PC som vist i .



Figur 24 : Blå Atmel mkII boks detekteret på PC

Driveren til den blå Atmel mkII boks burde være med som standard på enhver Windows 10 PC. Oplever man problemer, når man forbinder boksen til sin PC, kan det dog godt være, at man selv skal ud og finde en driver til boksen.

NODEMCU (ESP8266 BOARDS)

ARDUINO ZERO (ARM CHIP BASERET)