Contents

[1 Software 2](#_Toc121057977)

[1.1 Quartus (Quartus Prime 18.1) Lite Edition 2](#_Toc121057978)

[1.2 Download 7Zip: 2](#_Toc121057979)

[1.3 Arduino IDE 2.0.0 2](#_Toc121057980)

[1.4 Template 2](#_Toc121057981)

[2 Wie legt man ein neues Projekt an und kompiliert dieses? 3](#_Toc121057982)

[2.1 FHS-Project-Template Ordner "FHS\_DIG\_VidorProject" kopieren 3](#_Toc121057983)

[2.2 Im neuen Pfad Quartus durch Doppelklick auf MKRVIDOR4000.qpf starten: 3](#_Toc121057984)

[2.3 Jetzt können alle IOs des FPGAs im Schematic Design verwendet werden. 3](#_Toc121057985)

[2.4 CMD-File 4](#_Toc121057986)

[2.5 Kompilieren des Projekts durch Doppelklick auf ‘Compile Design’ im linken unteren Bereich der IDE starten. 4](#_Toc121057987)

[2.6 Erzeugen von app.h 5](#_Toc121057988)

[2.7 Arduino Sketch Template vorbereiten 5](#_Toc121057989)

[2.8 Laden des Programms mittels Arduino Sketch 5](#_Toc121057990)

[2.9 Aufbauen 6](#_Toc121057991)

# Software

**Die Folgende Anleitung ist für Windows ausgelegt!**

## Quartus (Quartus Prime 18.1) Lite Edition

**Wird für Mac nicht angeboten…**

Download unter:

<https://www.intel.com/content/www/us/en/collections/products/fpga/software/downloads.html?s=Newest&f:guidetmD240C377263B4C70A4EA0E452D0182CA=%5BIntel%C2%AE%20Quartus%C2%AE%20Prime%20Design%20Software%3BIntel%C2%AE%20Quartus%C2%AE%20Prime%20Lite%20Edition%3B18.1%5D>

Graphical user interface, text, application, email

Description automatically generated

**Dauer ca. 2-5 Stunden je nach Downloadgeschwindigkeit 5Gb**

Die Datei muss im Anschluss entpackt werden. Dafür wurde 7zip verwendet. Es geht aber auch jedes andere ZIP-Programm für .tar

## Download 7Zip:

Download unter:

<https://www.7-zip.org/download.html>

Beim Starten die „FreeLizenz“ wählen.

## Arduino IDE 2.0.0

Download unter:

<https://www.arduino.cc/en/software>

Shape

Description automatically generated

**Dauer ca. 5 Minuten je nach Downloadgeschwindigkeit**

## Template

Download unter:

<https://fhsalzburg.sharepoint.com/:f:/r/sites/TEAM-ITS-BAC1-FPGA/Freigegebene%20Dokumente/General/Arduino%20MKR%20Vidor%20Templates/00_FHS_DIG_VidorProject?csf=1&web=1&e=qbeKUb>

# Wie legt man ein neues Projekt an und kompiliert dieses?

Weg über Schematic Editor / FHS Template.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Wichtig - Zu beachten:**

* **3.3V IOs. Anlegen von 5V kann die IOs zerstören.**
* **Nie denselben Pin als Output auf MCU und FPGA konfigurieren!**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

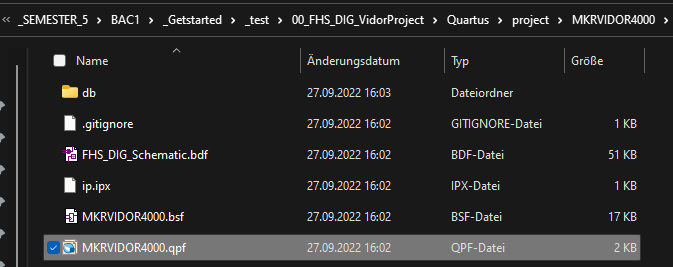
## FHS-Project-Template Ordner "FHS\_DIG\_VidorProject" kopieren

Graphical user interface, application

Description automatically generated

## Im neuen Pfad Quartus durch Doppelklick auf MKRVIDOR4000.qpf starten:

* ..\Quartus\project\MKRVIDOR4000\MKRVIDOR4000.qpf



* Quartus zeigt möglicherweise "IP upgrade recommended" an. Dieses über das IP Upgrade Tool durchführen ("Perform Automatic Upgrade")
* Im Project Navigator links auf "FHS\_DIG\_Schematic" doppelklicken um das Design zu öffnen

Diagram

Description automatically generated with low confidence

## Jetzt können alle IOs des FPGAs im Schematic Design verwendet werden.

Für uns interessant sind vor allem die digitalen IOs bMKR\_D[14..0].



Unter "Symbol Tool" können Logikbausteine (Libraries - c:/intelfpga\_lite/... - primitives) platziert werden.

Graphical user interface, application

Description automatically generated

Mit einem **Rechtsklick** auf eine Verbindung, kann unter Properties der Name der Verbindung geändert werden.

(Funktioniert auch mittels Doppelklick auf das Wire und direktem Eingeben des Namens)

**Mit dem Namen z.B. bMKR\_D[6] wird das Wire auf den digitalen Pin 6 verbunden.**

A picture containing diagram

Description automatically generated

## CMD-File

**Mit dem CMD File können die nächsten zwei Punkte übersprungen werden:**

**Das CreateSketchFile in den Projektordner Kopieren**

Ein Bild, das Text, Screenshot, Monitor, Bildschirm enthält.

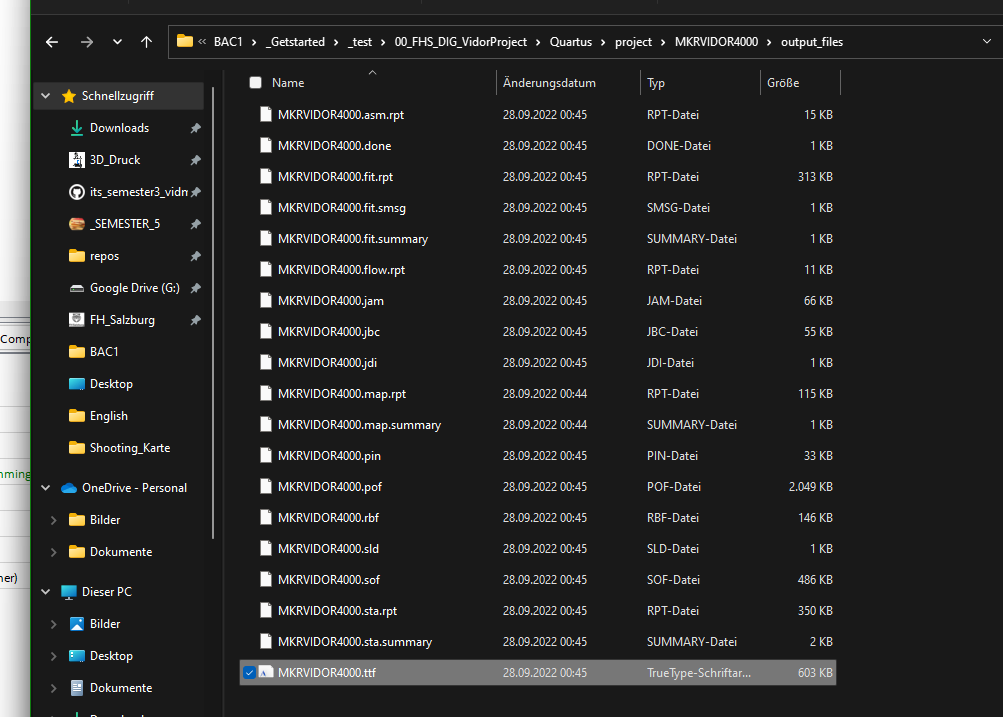
Automatisch generierte Beschreibung

## Kompilieren des Projekts durch Doppelklick auf ‘Compile Design’ im linken unteren Bereich der IDE starten.

Graphical user interface, text, application, email

Description automatically generated

--> Es wird ein ttf-File unter ..\Quartus\project\MKRVIDOR4000\output\_files angelegt.



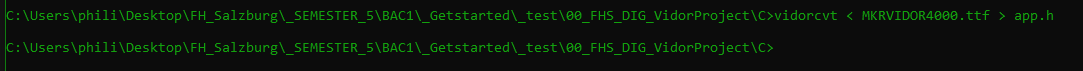
## Erzeugen von app.h

* ..\Quartus\project\MKRVIDOR4000\output\_files\MKRVIDOR4000.ttf in das C Verzeichnis des Projekts kopieren.

A screenshot of a computer

Description automatically generated with medium confidence

* Mit **cmd** in das C Verzeichnis des Projekts wechseln und app.h erzeugen: vidorcvt < MKRVIDOR4000.ttf > app.h (eventuell Antivirus Programm auschalten)



* app.h enthält jetzt die selbe Information wie das ttf File (Hex FPGA Inhalt), allerdings mit benötigter inversierter Bitfolge.

Es handelt sich also quasi um einen Hex-Dump des FPGA-Inhalts.

## Arduino Sketch Template vorbereiten

Erzeugte 'app.h' nach '…\Sketch\' kopieren

A screenshot of a computer

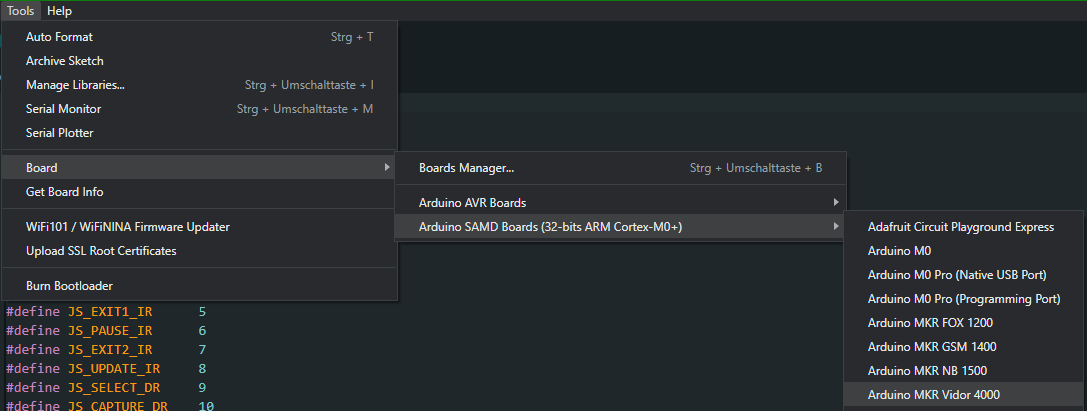
Description automatically generated with medium confidence

## Laden des Programms mittels Arduino Sketch

* '\Sketch\Sketch.ino' mit Arduino IDE öffnen



* Unter 'Tools - Board: - Board Manager' nach 'Arduino SAMD Boards' suchen und installieren, wenn nicht installiert.
* Verbindung herstellen mit 'Tools - Board: - Arduino SAMD Boards - Arduino MKR Vidor 4000'



* 'Tools - Port' - Port auswählen, an dem das Board verbunden ist

Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, text, application

Description automatically generated Graphical user interface, text, application

Description automatically generated

* 'defines.h' definiert, dass 'app.h' als Payload auf das FPGA geladen wird
* In der setup() Routine können optional Pins der MCU konfiguriert werden
* (Sollte für unser Digitaltechnik Projekt nicht der Fall sein)
* In der main loop() kann optional Code für die MCU implementiert werden
* (Sollte für unser Digitaltechnik Projekt nicht der Fall sein)
* FPGA über 'Sketch - Upload' programmieren

Background pattern

Description automatically generated

Background pattern

Description automatically generated

**Achtung:**

**Nie denselben Pin als Output auf MCU und FPGA konfigurieren!**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

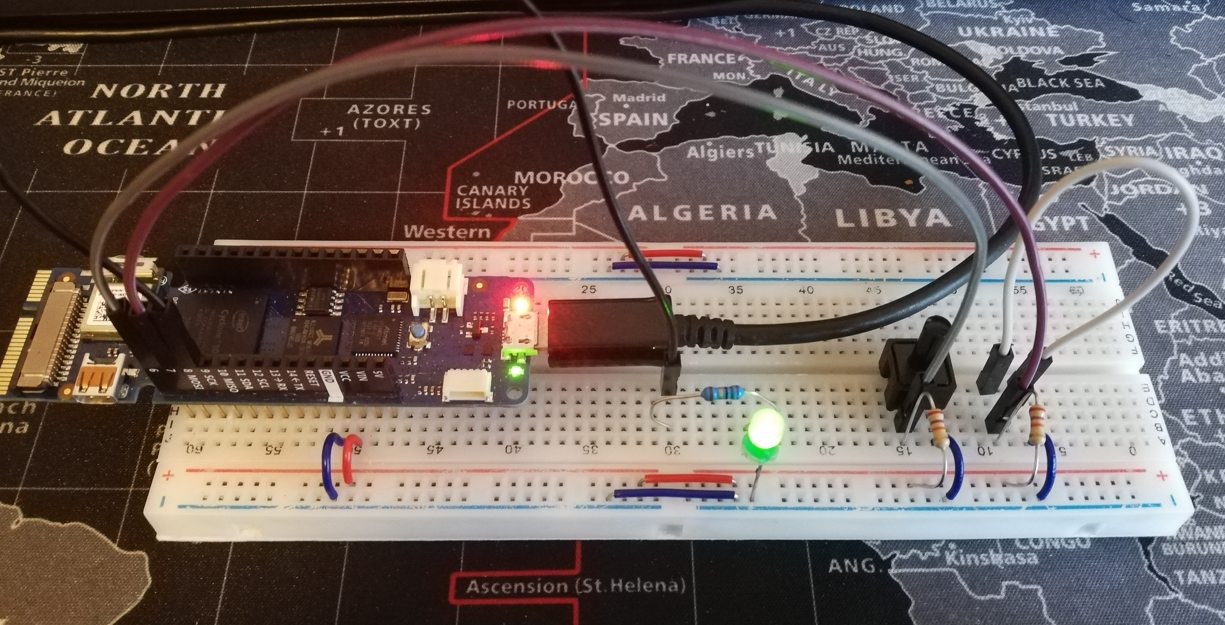
Pin mapping:

C:\Users\Hannes Waclawek\Documents\Git\Arduino MKR Vidor\VidorFPGA\constraints\MKRVIDOR4000

Pin IP folder:

C:\Users\Hannes Waclawek\Documents\Git\Arduino MKR Vidor\VidorFPGA\ip

## Aufbauen



Pin -> 6 & 7 Input mit PullUp (25kOhm laut Degenblatt FPGA)

Pin 8 -> Output

**INFO: Logisch ist der Schalter mit Pullup ein Öffner à der verbaute Schalter ist somit invertiert!**