Eclipse C/C++ (4.4, Luna) einrichten

Dieser Artikel enthält Hinweise, die das BiDiBOne-Projekt betreffen.

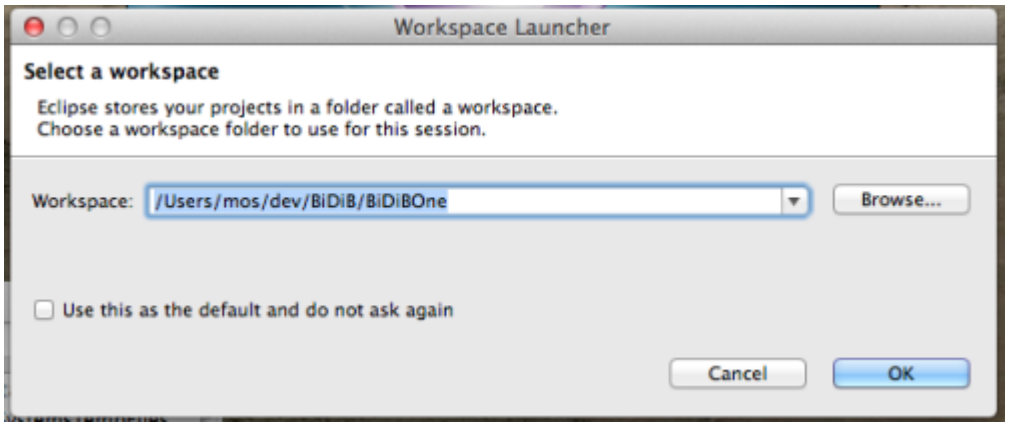
Diese Anleitung wurde mit einem Windows 7-System erstellt. Eine Adaption auf andere Betriebssysteme sollte aber keine Schwierigkeiten bereiten.

Vorausgesetzt wird die Installation einer Java-Umgebung (JRE 8) und des Eclipse-Frameworks C/C++ (Eclipse C/C++). Eine so genannte AVR GNU Toolchain, mit Compiler und Linker, ist i.d.R. auf Linux- und Mac OS-Systemen installiert. Für Windows bietet sich WinAVR an. Die Einbindung für die Hardwareunterstützung (AVR Eclipse Plugin) wird unten beschrieben.

Hier wird u.A. das Clonen des Repositorys mit Eclipse beschrieben. Der Vorteil liegt in der Verwendung eines einzigen Werkzeuges. Verfahren mit anderen Werkzeugen werden z.B. im Kapitel BiDiBOne aus dem Repository laden beschrieben.

Eclipse initial starten

Beim Starten fragt Eclipse nach einem Workspace, in dem die eclipse-spezifischen Informationen (Projektdaten) abgelegt werden sollen. Insbesondere das Verzeichnis .metadata enthält benutzerspezifische Daten.



Verzeichnis wählen und OK

Das ist das Verzeichnis, in dem die Eclipse-Verwaltungsdaten liegen und **nicht** das Verzeichnis, in dem später die **Quelldateien** aus dem Repository gespeichert werden!

BiDiBOne-Software aus dem Repository laden

Unser BiDiBOne-Repository - also die Sammlung der Quellen, die für den Bau einer BIDiBOne-AddOn-Anwendung notwendig ist - haben wir derzeit auf GitLab hinterlegt. Dort muss man einen entsprechenden Zugang (Account) eingerichtet haben.

Weiterhin ist zur Verwendung dieser Quellen eine Zugangsberechtigung notwendig, die von support@fichtelbahn.de zur Verfügung gestellt wird.

Wir schließen die Willkommen-Seite und stellen die Verbindung zum Git-Repository her:

Hauptmenü | Window | Open Perspective | Other… | Git ⇒ OK

In dieser Ansicht: **Clone a Git Repository** anwählen.

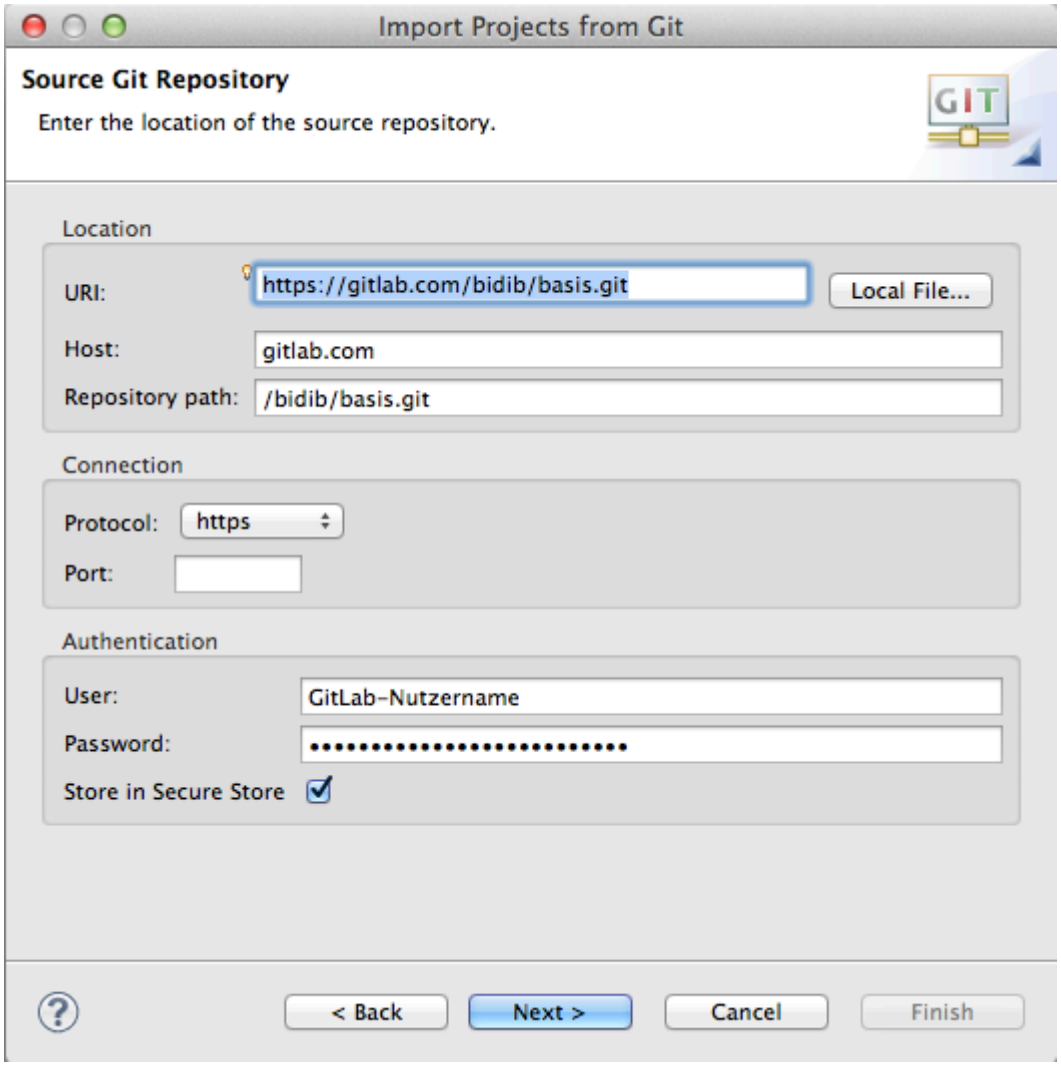
In das Feld URI kann man jetzt die von GitLab bereitgestellte Adresse: „https://gitlab.com/bidib/core.git“ eingeben. Eclipse ergänzt die notwendigen Angaben für „Location“ und „Connection“.

Wenn man die obige Adresse vorher in die Zwischenablage kopiert hat, schlägt Eclipse sie als URI vor.

Fehlen noch:

- User: <Bei GitLab registrierter Nutzername>
- Password: <mit entsprechendem Kennwort>

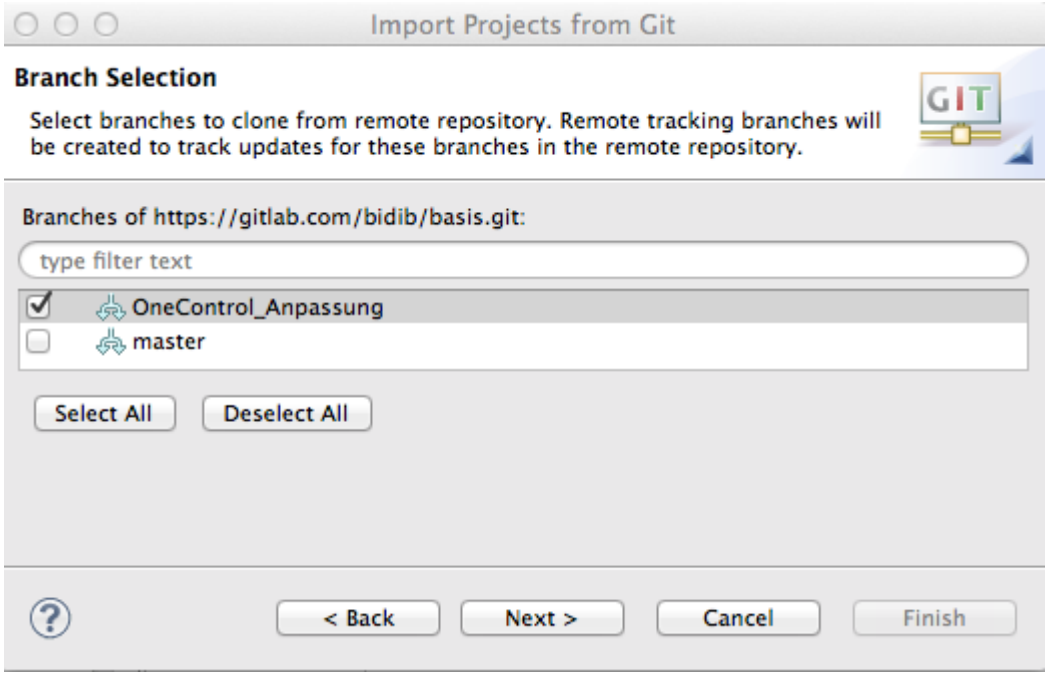
- Store in Secure Store: Anhaken enthebt von ständiger Neueingabe beim Auffrischen



`Next` (Im Bild ist noch

das alte *basis*-Projekt gezeigt; bitte hier das core-Projekt laut Link nehmen.)

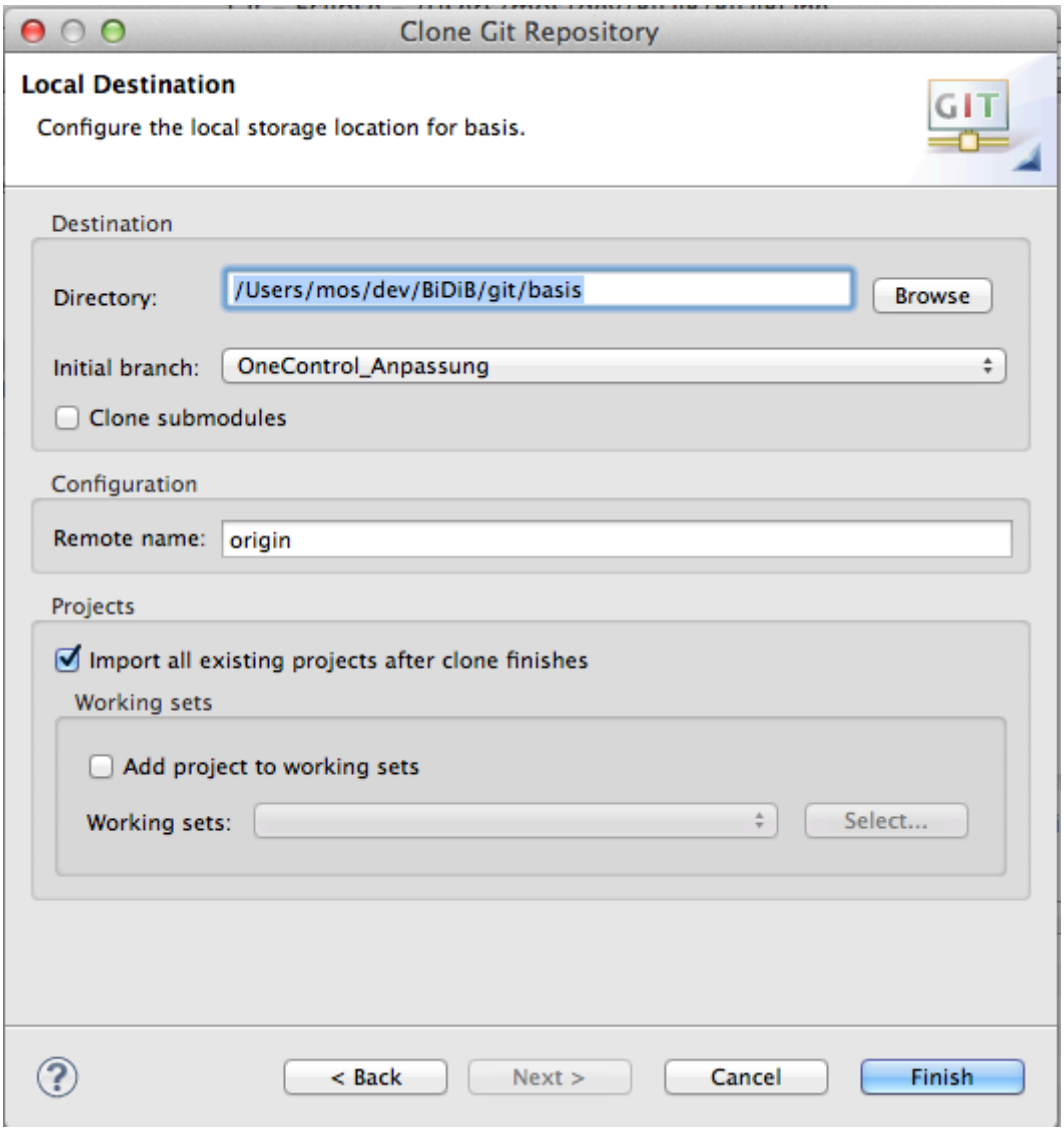
Jetzt müssen wir den aktuellen Branch für die BiDiBOne-Basis auswählen:



`Next`

Mit Stand 23. April 2014 enthält: **master** unsere aktuelle Firmware. Hier werden im Laufe der Entwicklung der Basis-Firmware andere Branchs auftauchen, die verschiedenen Erweiterungen oder Korrekturen enthalten. Lauffähig ist aber in jedem Fall der **master**-Branch.

Es folgt das Zielverzeichnis für die Quelldateien. Bei Bedarf das vorgeschlagene Verzeichnis anpassen. In jedem Falle sollten aber das Eclipse-Projektverzeichnis und Git-Quellenverzeichnis



verschieden sein.

`Finish`

Seit Dezember 2014 werden alle Projekte unabhängig, also nicht mehr in Submodulen organisiert!

Das zweite grundlegende Projekt ist das **support**-Projekt (https://gitlab.com/bidib/support.git), dass alle unterstützen Quellen, wie z.B. Servos enthält.

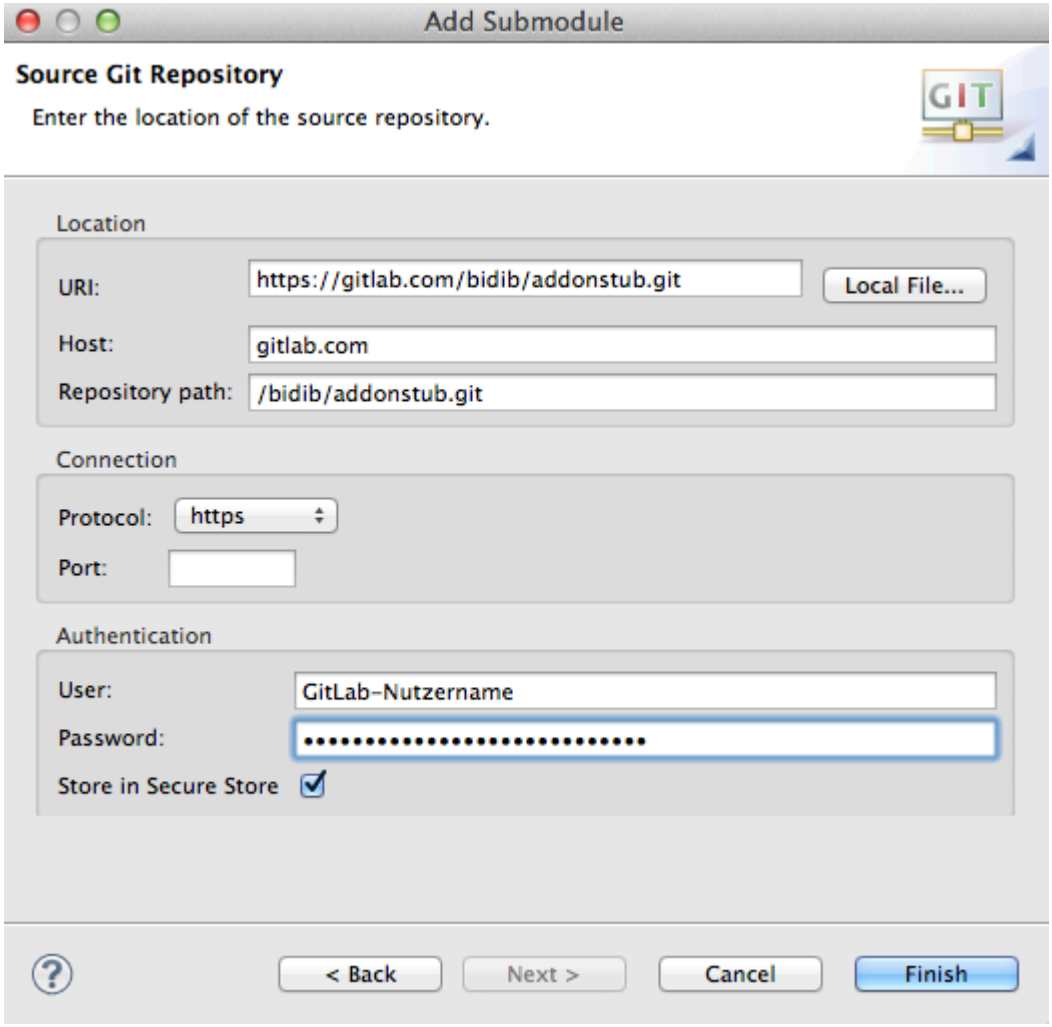
Jetzt haben wir alle Quellen für die Basis-Firmware. Da diese jedoch auf einem AddOn aufbauen, benötigen wir ein minimales AddOn-Projekt.

AddOn-Quellen besorgen

Für die Neuentwicklung eines AddOns steht der **AddOn-Stub** zur Verfügung. Dort sind alle notwendigen Module und Funktionen aufgeführt, um mit dem BiDiB-System zu kommunizieren.

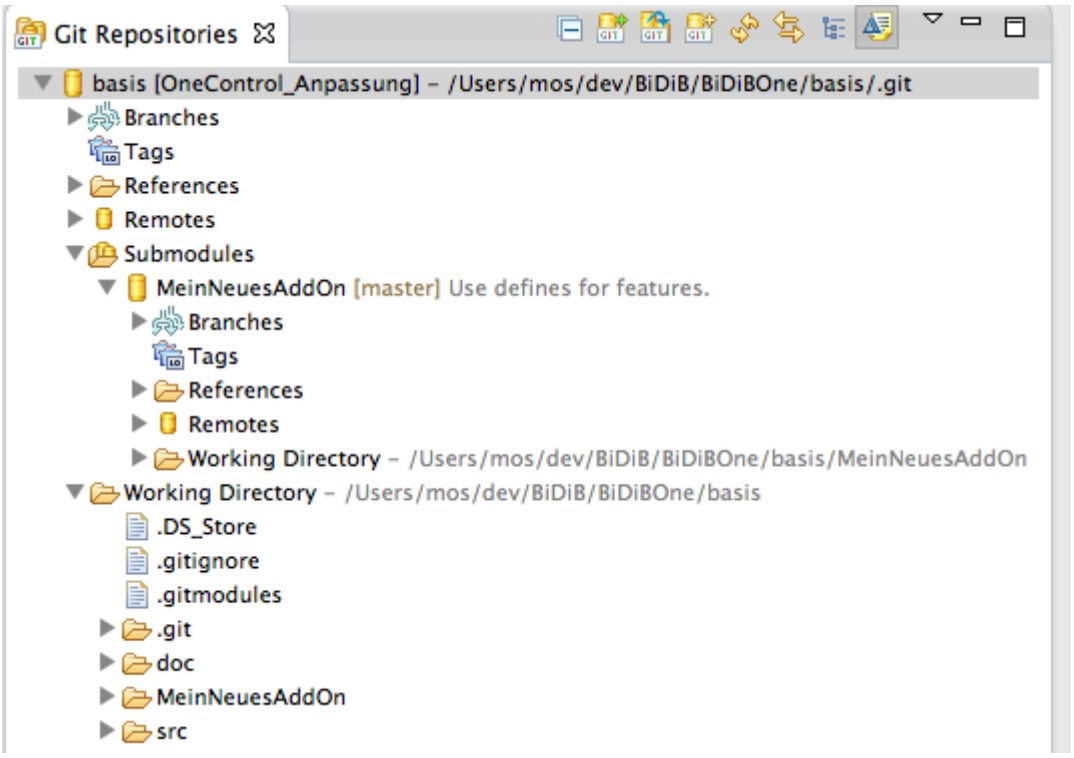
Zusammen mit diesem AddOnStub sind mit der BiDiBOne-Baugruppe Minimalfunktionen möglich.

Falls noch kein AddOn-Projekt zur Verfügung steht, kann der AddOnStub auf GitLab als Muster verwendet werden. Wir klonen das Projekt, um die Quellen später in unser eigenes Git-Projekt kopieren zu können: Den Pfad holt man sich wieder von GitLab: https://gitlab.com/bidib/addonstub.git



`Finish`

Nach erfolgreichem Klonen zeigt die Repository Perspektive alle Projekte an wie sie auf dem Laufwerk abgelegt sind:



Somit stehen alle Quellen bereit und müssen im nächsten Schritt unserem Eclipse-Projekt bekannt gemacht werden.

Im Gegensatz zum Basisprojekt, das nur lesend zur Verfügung steht, führt man aber hier die Änderungen für sein eigenes AddOn durch. Damit das funktioniert, richtet man sich an der gleichen Stelle ein eigenes Repository ein. Man erzeugt einen Ordner mit einem passenden Namen an der gleichen Stelle und kopiert dorthin die Quellen - ohne die .git-Datei.

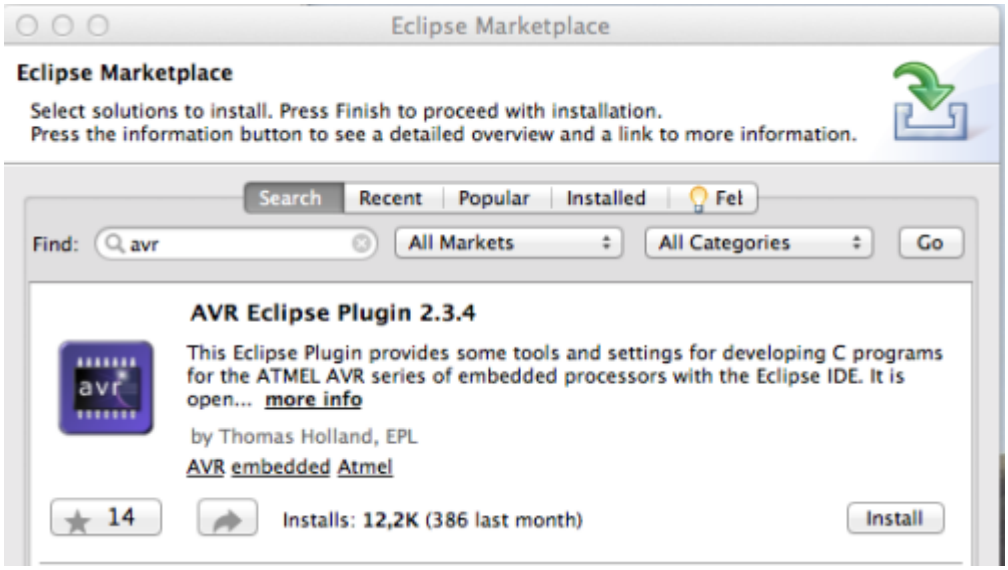
Prinzipiell ist natürlich jeder andere Platz möglich, allerdigs sind die die Projektdateien und zugehörigen makefiles an diese Struktur angepasst. Nach Rechtsklick auf das Modul-Verzeichnissymbol kann man jetzt die Ansichten synchronisieren:

Um „MeinNeuesAddOn“ in Git verwalten zu können, muss es jetzt durch „Add to Index“ erfasst werden.

AVR Eclipse-Plugin installieren

Es wird vorausgesetzt, dass das AVR GNU Toolschain installiert ist. Dadurch werden notwendige Einstellungen automatisch übernommen.

Jetzt gehen wir erst Mal auf dem Markt einkaufen - und zwar das für unsere Hardware nötige Plug-In: Im Hauptmenü Help | Eclipse Marketplace: Im Suchfeld „AVR“ eingeben und mit Go bestätigen.



`Installation` starten.

Im nächsten Dialog alle Features mit `Confirm` bestätigen, dann Lizenzbedingungen zustimmen und mit `Finish` abnicken.

Sicherheitswarnung mit `OK` bestätigen.
Anschließend Eclipse neu starten.

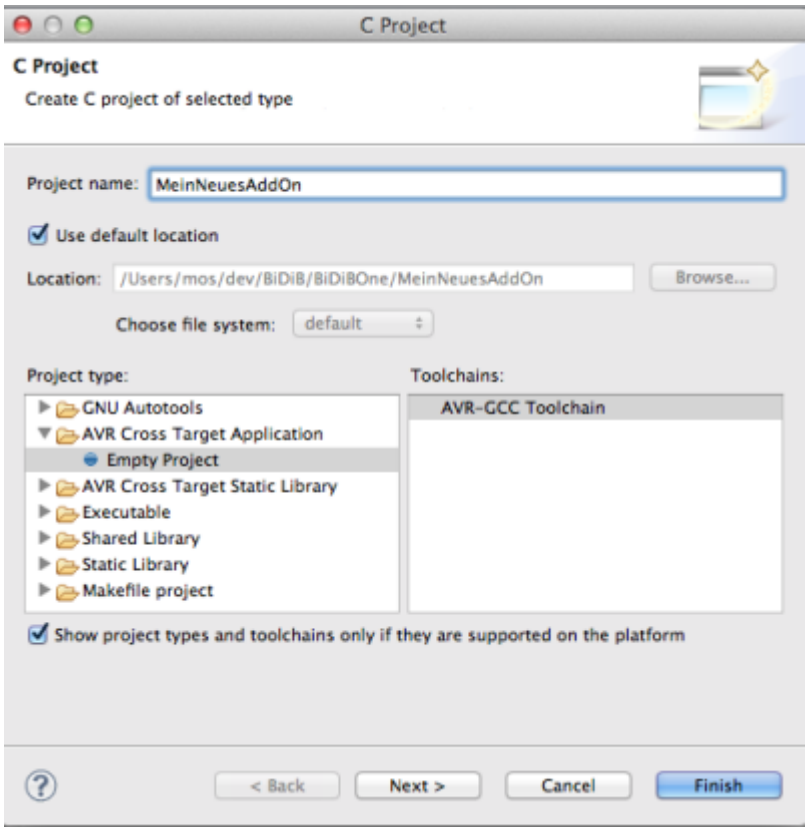
Nach dem Neustart stehen in der C/C++-Perspektive ein neuer Menüpunkt „AVR“ im Hauptmenü sowie ein neues Konfigurationskapitel in den Einstellungen (Eclipse | Einstellungen bzw. Window | Einstellungen) zur Verfügung.

Projekt anlegen

.... in Arbeit

Hauptmenü | File | New | **C Project** ⇒ **AVR Cross Target Application** → **Empty Project**

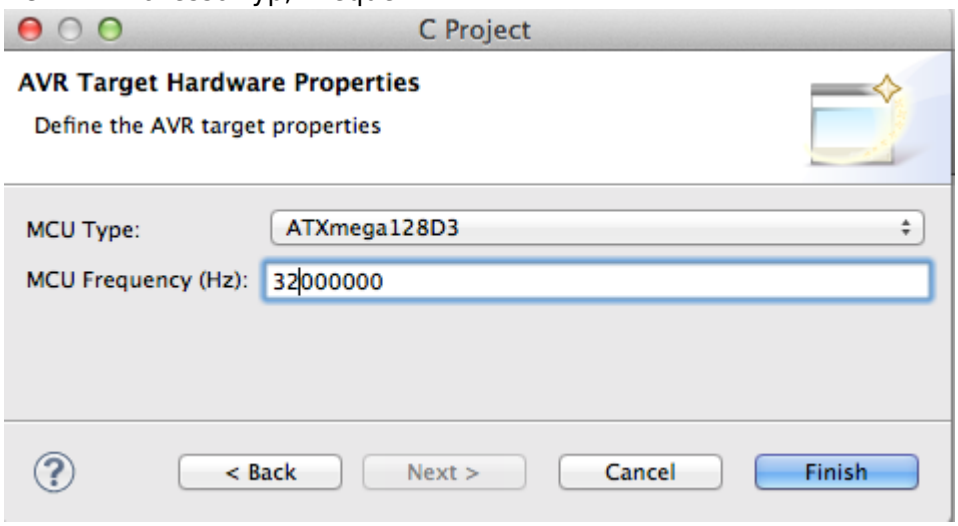
Rechtes Fenster Toolchain: AVR-GCC Toolchain (müsste die einzige Auswahl sein)



(Eclipse-)Projektname:
MeinNeuesAddOn

`Next` ⇒ (Debug-, Release-Configurations)

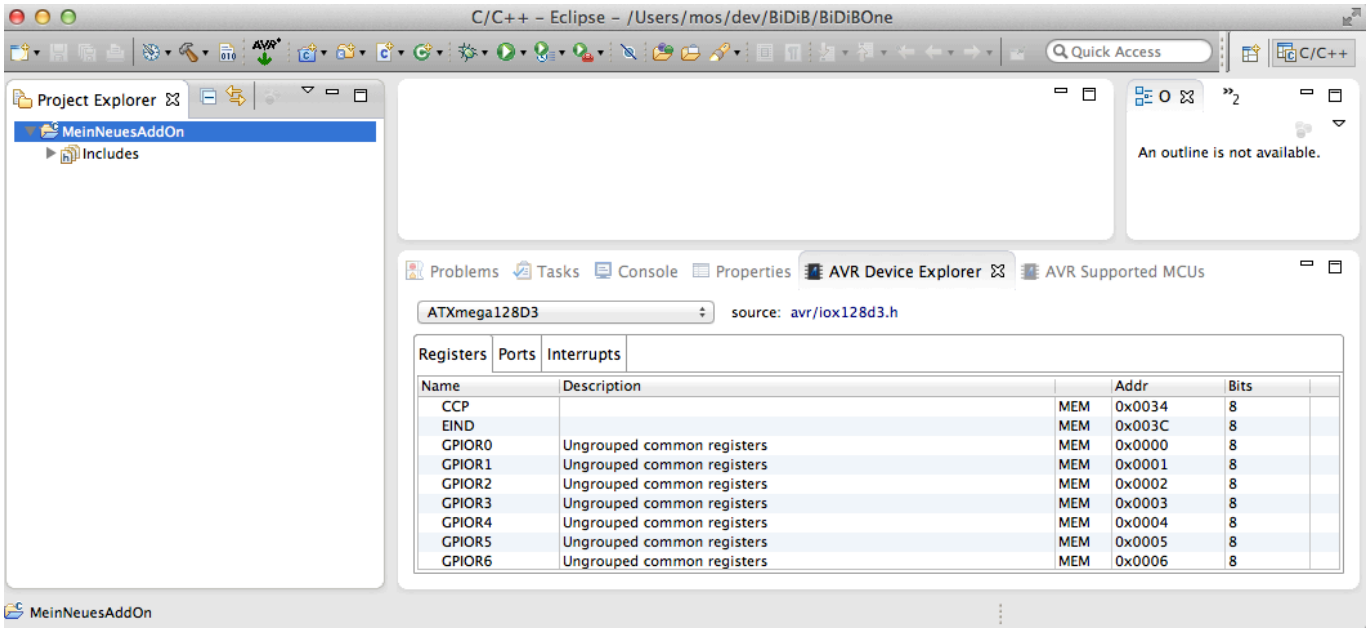
`Next` ⇒ Prozessortyp, -frequenz



- MCU Type: **ATXmega128D3** (für BiDiBOne-Basisplatine)
- MCU Type: **ATXmega128A3** (für BiDiBOne-Plus-Basisplatine)
- MCU Frequency (Hz): **32000000**

`Finish`

Im „Project Explorer“ wird jetzt ein leeres Projekt angelegt. Die „Includes“ können wir unbehelligt lassen, da wir einige davon benötigen werden.



Wie man sieht, stellt das Plug-In auch einige nette Informationen zu der gewählten Hardware bereit. Zur Erstellung eines BiDiBOne-AddOns mit den bereitgestellten Schnittstellen und „Hooks“ werden diese Informationen aber in der Regel nicht benötigt.

Beachte: Nimmt man die Vorgaben aus dieser Anleitung, gibt es jetzt ein Verzeichnis BiDiBOne/MeinNeuesAddOn, das die Eclipse Informationen enthält. Daneben existiert ein Verzeichnis git/basis[/MeinNeuesAddOn] mit den Quellen für BiDiBOne-Basis und dem neuen AddOn.

Jetzt müssen wir die Quellen aus dem Repository in unser Projekt einbinden:

- Hauptmenü | File | Import ⇒ Git → **Projects from Git** `Next`.
- **Existing local repository** `Next`
- Add **…**
- `Browse` **…** und zum Verzeichnis **basis** navigieren

Einstellungen

.... in Arbeit

Tools | Options

- remove white spaces - tabs to white spaces ...

Tastaturkürzel

Refactor-Rename ...

Change Signature ...

Refactor Document Method

Zeilennummern

Tabulatoren und Einzüge

Pfad zu Compiler und Linker

Toolchain

Text ...

Abhängig vom Projekt die Eigenschaften einstellen.

Modulinformationen

→ Outline

FUSES

(siehe übergeordnetes Kapitel)

Item Templates

(siehe übergeordnetes Kapitel)

Plug-Ins

Versionskontrolle

Seit Eclipse 4.3 (Kepler) steht GiT als Plug-In zur Verfügung. Eine Installation, um GiT aus Eclipse heraus zu verwenden ist dadurch nicht notwendig.

Die Anmeldung bei GiTLab ist in

GiT Anmeldung

beschrieben.

Repository einrichten …

Dokumentation

Zur Dokumentation oder zum Lesen dokumentierter Quellen bietet sich Doxygen an.

Tipps

Zugriff auf Systemdaten gestatten

Bestimmte Security-Software blockiert standardmäßig den Zugriff auf Systemdateien etc. Dieser Zugriff muss für manche Applikationen explizit gewährt werden:

- arm-none-eabi-gcc.exe
- cc1.exe
- as.exe
- collect2.exe
- id.exe
- arm-none-eabi-objcopy.exe
- arm-none-eabi-objdump.exe
- arm-none-eabi-size.exe

Einige Funktionen werden erst im Laufe der Zeit bzw. beim Debugging aufgerufen.

Unit Tests

… wird fortgesetzt.