FH Technikum Wien

AVR-Programmierung unter Mac OSX

im Studiengang BEL3

Lehrveranstaltung Embedded Systems

Tutorial

ausgeführt von:
Jürgen Hausladen
A-2460 Bruck/Leitha, Obere Neugasse 6

Wien 01.02.2011

FACHHOCHSCHULE
TECHNIKUM WIEN

Inhaltsverzeichnis

1 Einleitung

Für den Abschluss des 3. Semesters an der FH-Technikum Wien, war im Fach Embedded Systems ein Mikrocontroller-Projekt zu realisieren. Hierfür wurde der Mikrocontroller AT90USB162 verwendet. Da jedoch die Entwicklungswerkzeuge von ATMEL nur bedingt für andere Plattformen als Microsoft Windows verfügbar sind, bzw. nahezu keine Anleitungen für die Installation bzw. Konfiguration vorhanden sind, habe ich mir zum Ziel gesetzt, diese unter Mac OSX nativ zu betreiben. Ein weiteres Ziel war, die Verwendung der Entwicklungsumgebung Eclipse für die Programmierung, da diese plattformunabhängig erhältlich ist, und zahlreiche Features bzw. Erweiterungen besitzt. Das komplette Tutorial wurde unter Mac OSX 10.6 Snow Leopard getestet.

2 Grundlegendes

Zu Beginn muss, um unter Mac OSX überhaupt Code erzeugen zu können XCode installiert sein. Das Installationspaket hierfür befindet sich auf der „Mac OSX Install DVD“ unter „Optionale Installationspakete“.

3 CrossPack for AVR Development

„CrossPack for AVR Development“ ist vergleichbar mit dem Programm WinAVR unter Windows. Es beinhaltet die komplette GNU Compiler Suite, C-Bibliotheken für die AVR-Programmierung, AVRDUDE zum Programmieren des Controllers und noch viele weitere Features. Es ist mit diesem Package auch das Debuggen und Simulieren möglich. Das Installationspaket hierfür ist unter „http://www.obdev.at/products/crosspack/download.html“ erhältlich.

4 Eclipse

4.1 Installation

Die Entwicklungsumgebung „Eclipse“ ist eine sehr verbreitete Entwicklungsumgebung und ist für die verschiedensten Programmiersprachen erhältlich. Der große Vorteil liegt hier darin, dass die Entwicklungsumgebung plattformunabhängig ist und man sich daher nicht jedes Mal bei einem Systemwechsel eine andere Entwicklungsumgebung anlernen muss. Eclipse ist unter „http://www.eclipse.org/downloads/“ erhältlich. Die benötigte Version trägt den Namen „Eclipse IDE for C/C++ Developers“. Hierbei ist zu beachten, dass bei Verwendung von Mac OSX 10.6 (Snow Leopard) die 64-Bit Version verwendet werden sollte, da Mac OSX 10.6 ein 64-Bit Betriebssystem ist und daher nicht jedes Mal beim Start die 32-Bit Bibliotheken geladen werden müssen. Das eben geladene Archiv kann dann anschließend in

einen Ordner entpackt werden und über Drag & Drop in den Programmordner gezogen werden. Ausgeführt wird das Programm über die Eclipse App im Hauptverzeichnis.

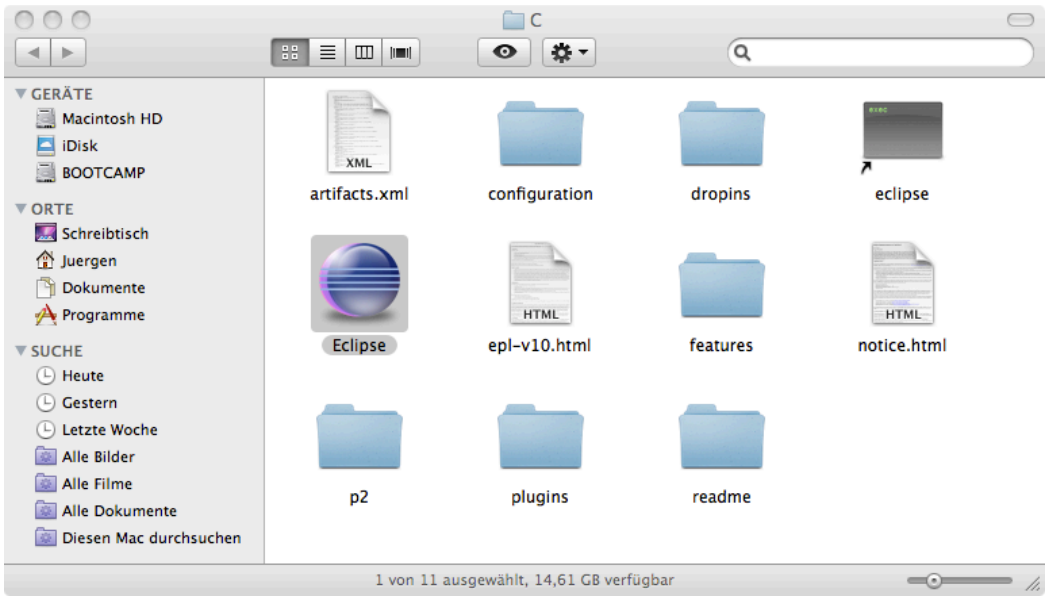

Abb.1: Eclipse Programmordner

Für Entwickler, welche Eclipse auch für die Java-Programmierung einsetzen, empfiehlt es sich beide Java Apps getrennt zu installieren, um auch beide Apps gleichzeitig ausführen zu können.

4.2 Konfiguration

Da nun Eclipse fertig installiert ist müssen nun noch die entsprechenden Plugins für die AVR-Programmierung installiert werden. Dafür wird Eclipse gestartet und der Plugin-Manager über das Menü „Help“ und „Install New Software“ aufgerufen.

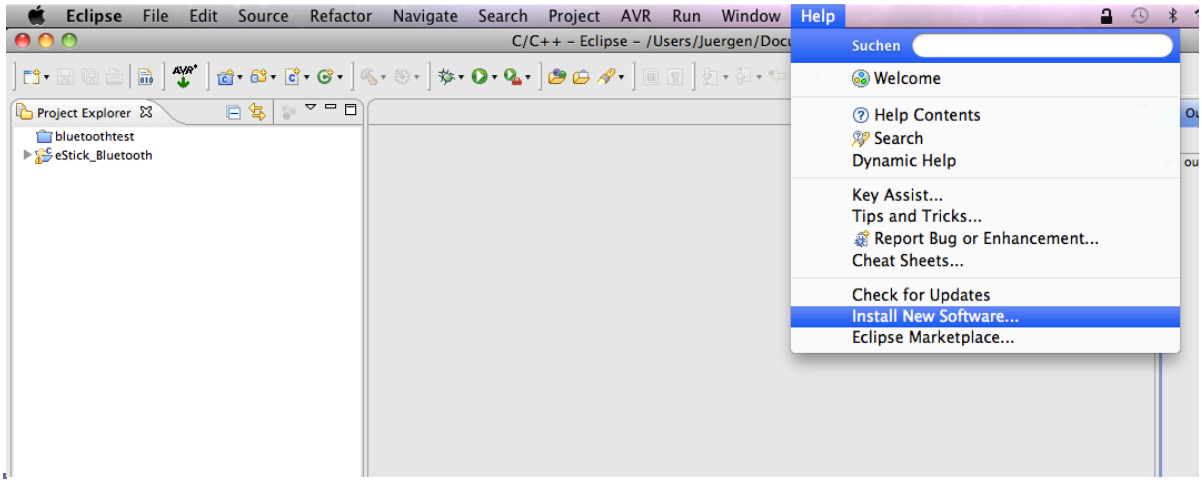

Abb.2: Eclipse Plugin-Manager

Anschließend daran wird in das Feld „Work with:“ die URL zum AVR-Plugin eingegeben („http://avr-eclipse.sourceforge.net/updatesite/“). Nun muss nur noch der Plugin in dem darunterliegenden Feld ausgewählt werden und über die Schaltfläche „Next“ installiert werden.

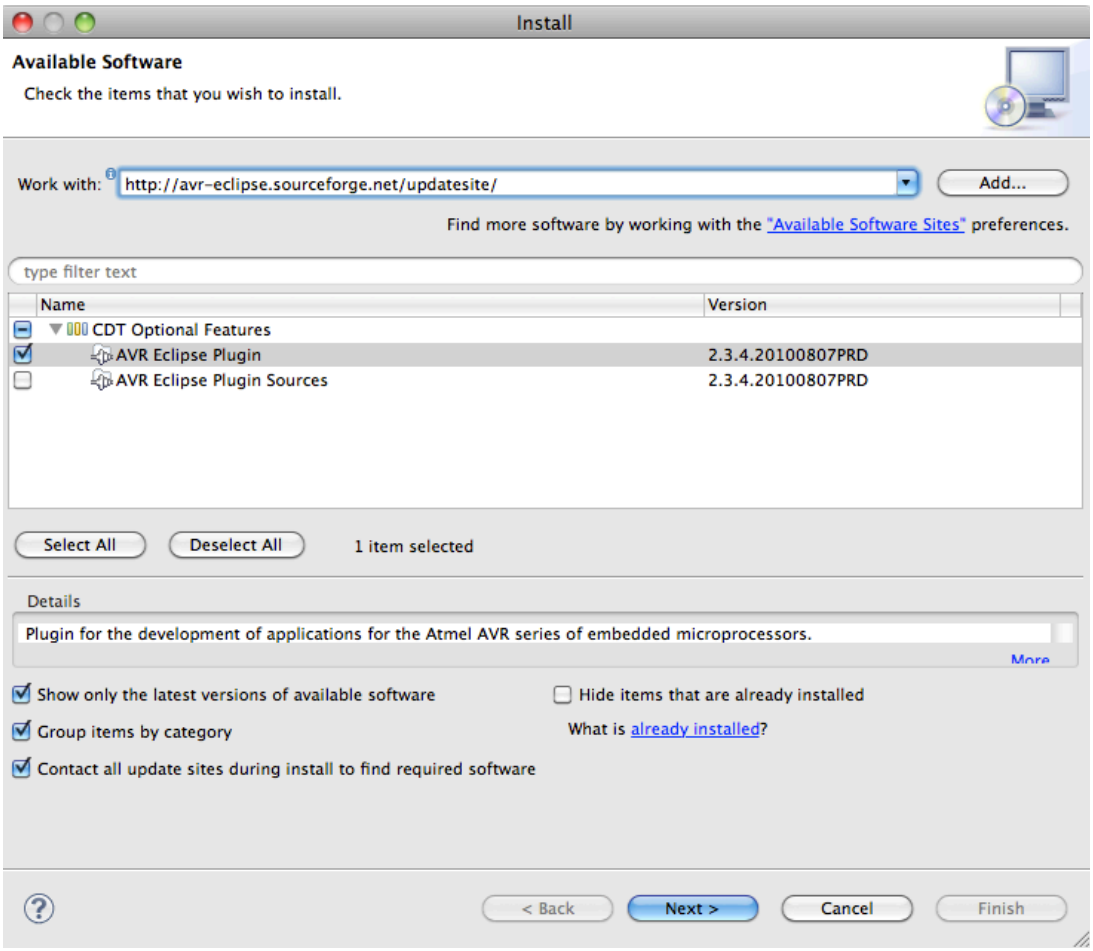

Abb.3: Eclipse AVR-Plugin Installation

Falls Eclipse nach der Installation neu starten möchte, so sollte dies auch zugelassen werden, um die korrekte Funktion des Plugins zu gewährleisten.

4.3 Konfiguration eines AVR-Programmers (optional)

Da der e-Stick nicht über einen entsprechenden Programmer programmiert wird, sondern über einen Bootloader geflasht wird, ist dieser Schritt nicht zwingend erforderlich.

Um einen AVR-Programmer für die Programmierung des Mikrocontrollers einzurichten, muss dieser im Plugin „AVRDude“ definiert sein. Dieser Plugin ist unter „Eclipse“, „Einstellungen“, „AVR“, „AVRDude“ zu erreichen.

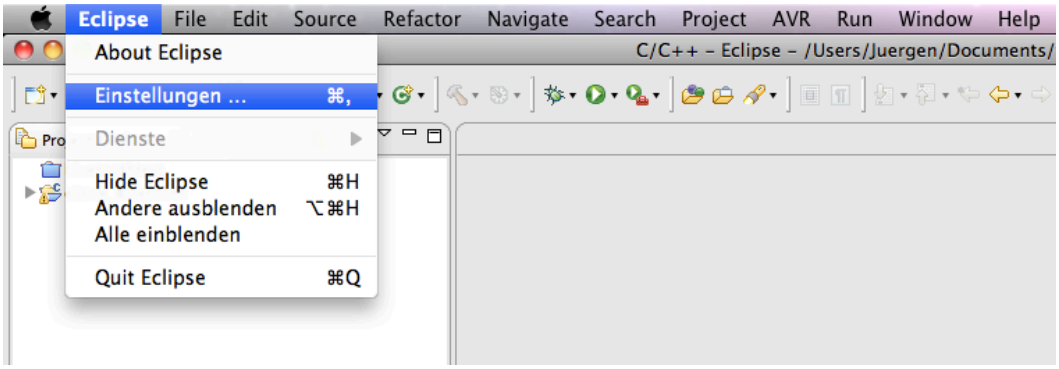

Abb.4: Einstellungen

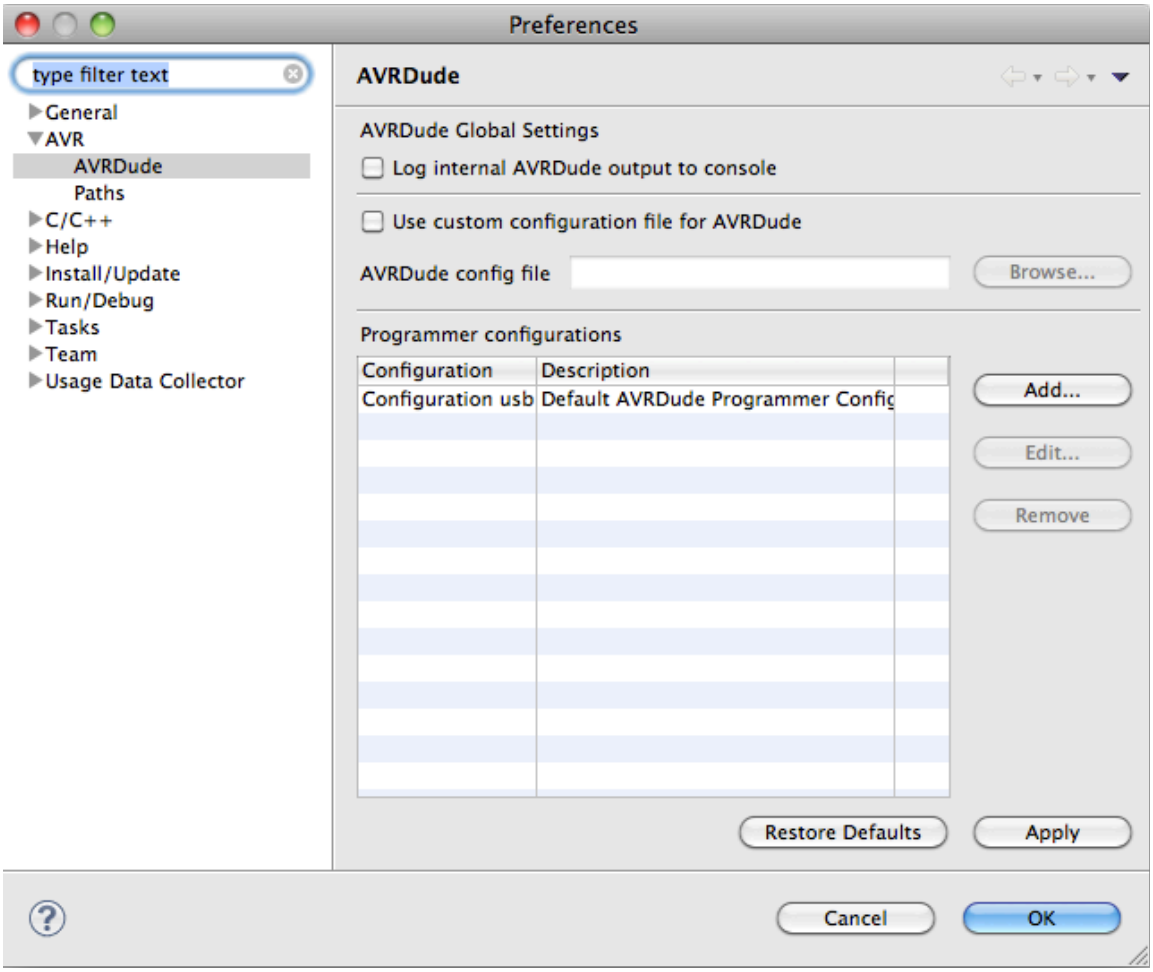

Abb.5: AVRDude

Um die entsprechende Hardware hinzuzufügen, muss zunächst der Button „Add“ gedrückt werden, wodurch eine große Auswahl an Programmern aufgelistet wird.

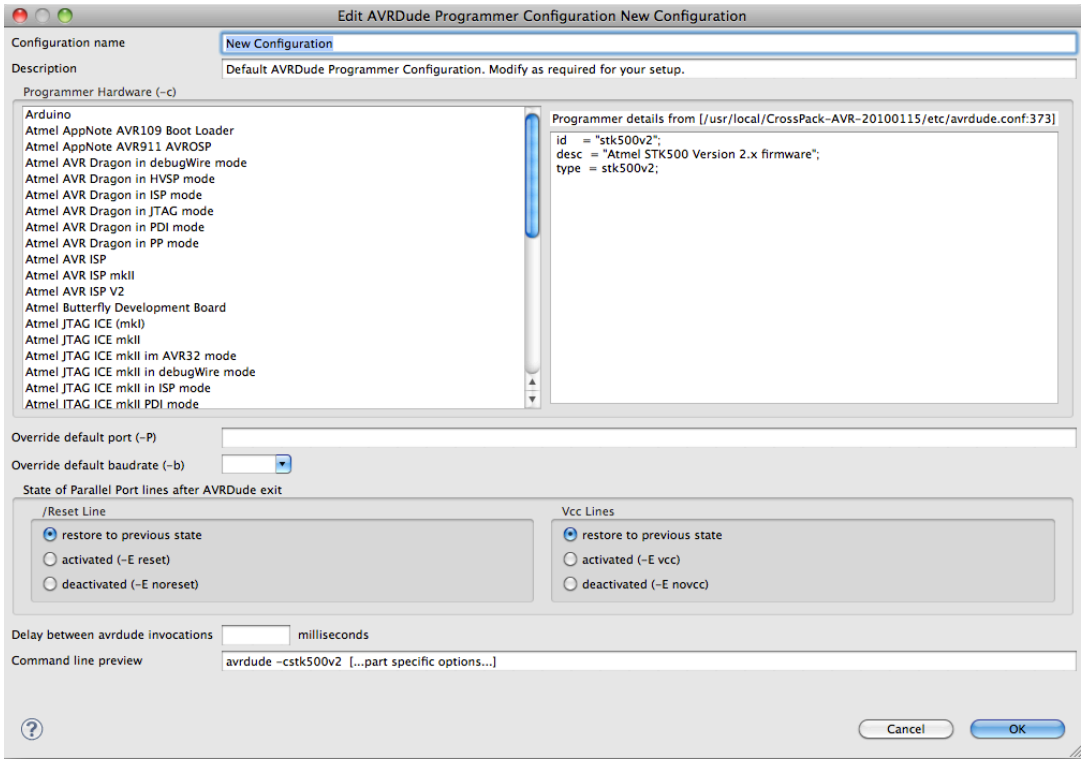

Abb.6: AVR-Programmer

Hier muss nun nur noch der richtige Programmer ausgewählt, eventuell die Baud-Rate, der Port des Programmers bzw. der Name der Konfiguration festgelegt werden. Zum Abschluss sollte noch unter „Paths“ die Pfade zum Compiler überprüft werden.

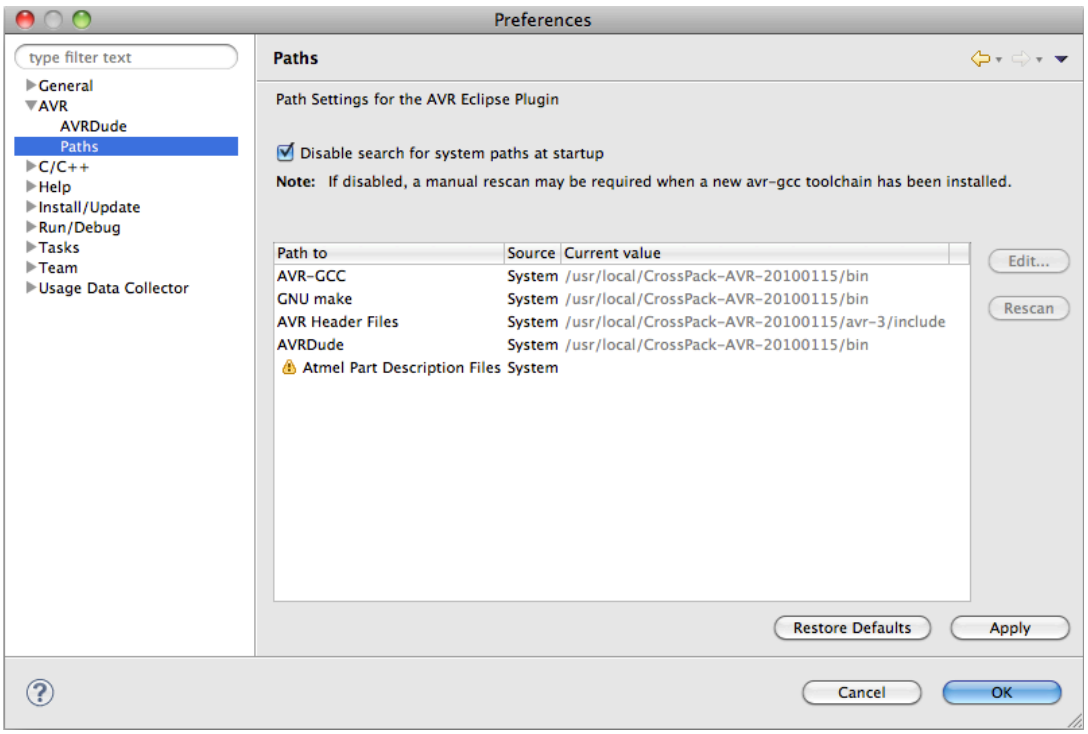

Abb.7: Paths

5 dfu-programmer

5.1 USB-Bibliothek

Damit der dfu-programmer den angeschlossenen Mikrocontroller richtig erkennt, sind zwingend die entsprechenden USB-Bibliotheken für den entsprechenden Mikrocontroller notwendig. Diese können direkt von der Seite „http://www.ellert.se/twain-sane/“ geladen werden. Für eine schnelle Installation ohne jeglichen Aufwand sollte hier je nach System das binary package von „libusb“ geladen werden. Erfahrene Nutzer können sich natürlich hier auch die Sourcefiles downloaden und die Bibliothek selbst kompilieren.

5.2 dfu-programmer installieren

Der dfu-programmer ist ein USB-Flash Tool für AVR Mikrocontroller, welche einen Bootloader besitzen. Dieses Programm ist vergleichbar mit dem FH-Tool oder ATMEL Flip. Dieses Tool muss als Einziges leider von Hand kompiliert werden. Dazu werden die Sourcefiles von „http://sourceforge.net/projects/dfu-programmer/“ geladen, in einen beliebigen Ordner entpackt und anschließend über „Programme“, „Dienstprogramme“, „Terminal“ konfiguriert und kompiliert. Ich empfehle hier den dfu-programmer in der Version 0.5.2 zu verwenden, da die Version 0.5.4 Probleme bei der Konfiguration unter Mac OSX 10.6 macht. Zur Konfiguration und Kompilierung wechselt man im Terminal in das entpackte Verzeichnis über den Befehl „cd Beispielverzeichnis/.../dfu-programmer-0.5.2“(„...“ steht für die Verzeichnisse bis zu dem entpacken dfu-programmer Verzeichnis). Anschließend wird das Skript bootstrap.sh ausgeführt. Dies geschieht über den Befehl „./bootstrap.sh“.

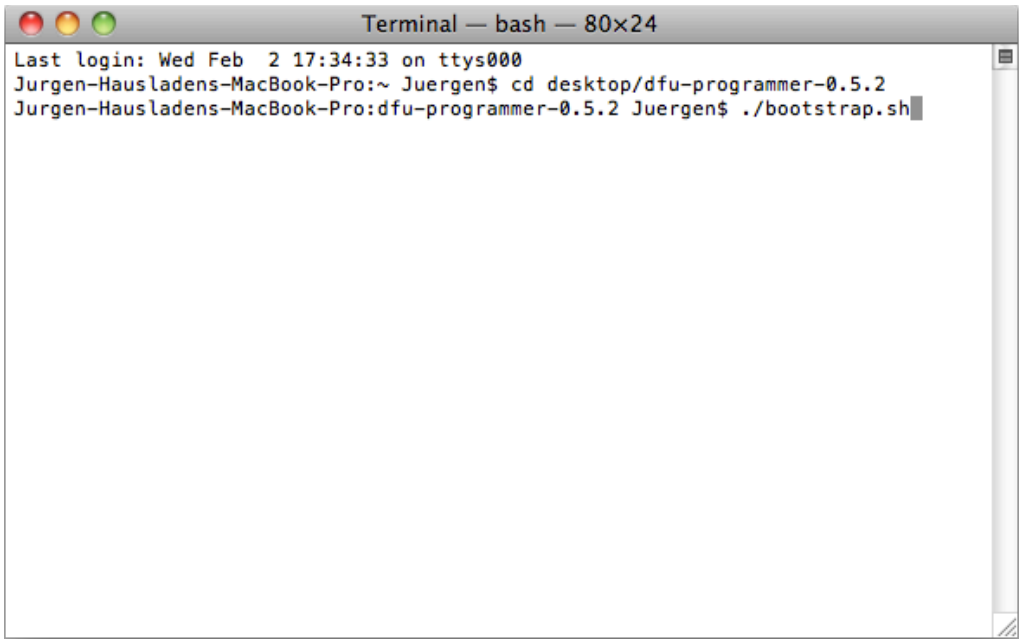

Abb.8: dfu-programmer konfigurieren & installieren

Im nächsten Schritt konfigurieren wir das Projekt für die Kompilierung über den Befehl „./configure“. Ist dies geschehen, wird das Projekt über den Befehl „make“ kompiliert. Zum Installieren muss nun nur noch der Befehl „make install“ ausgeführt werden. Sollte hier ein Fehler auftreten, so wurde höchstwahrscheinlich aufgrund der fehlenden administrativen Rechte, der Zugriff für die Installation verweigert. Für diesen Fall lautet der Befehl wie folgt „sudo make install“. Dabei wird für die Authentifizierung als root vor der Installation nach dem Benutzer-Passwort gefragt.
Wurden all diese Schritte erfolgreich durchgeführt, so kann nun über die folgenden Befehle der Mikrocontroller über den Terminal geflasht werden:

Syntax: dfu-programmer <Mikrocontrollertyp> <Befehl>

dfu-programmer at90usb162 erase .....Löscht den kompletten Mikrocontroller
dfu-programmer at90usb162 flash main.hex .....Flasht das Programm „main.hex“ auf den Mikrocontroller
dfu-programmer at90usb162 start .....Startet das eben geflashte Programm

6 Erstellen eines AVR-Projektes

6.1 Projektkonfiguration

Zum Erstellen eines AVR-Projektes wird im Menü „File“ der Punkt „New“ und im weiteren der Punkt „Project“ angeklickt.

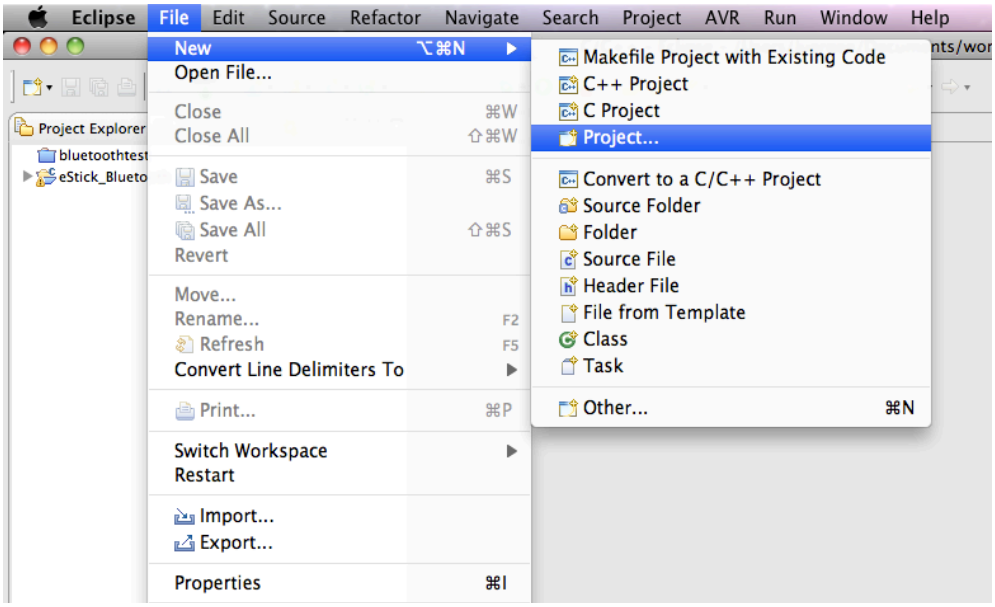

Abb.9: Projekt erstellen

Anschließend öffnet sich ein Fenster mit der Auswahl der Programmiersprache. Hier wird der Punkt „C Project“ im Menü „C/C++“ ausgewählt.

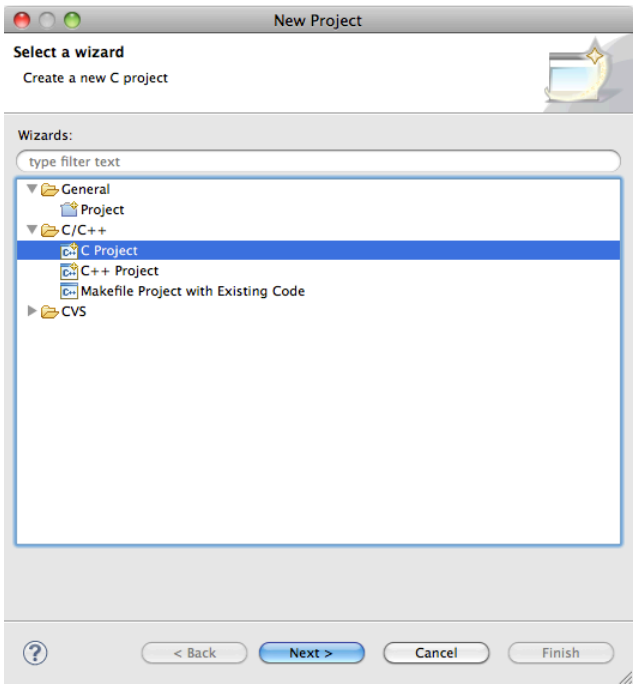

Abb.10: Programmiersprachenauswahl

Ist der Eintrag ausgewählt, wird über den Button „Next“ der nächste Konfigurationsschritt eingeleitet. Hier spezifiziert man unter „Project Name“ den Projektnamen und wählt darunter „AVR Cross Target Application“ und „Empty Project“ aus.

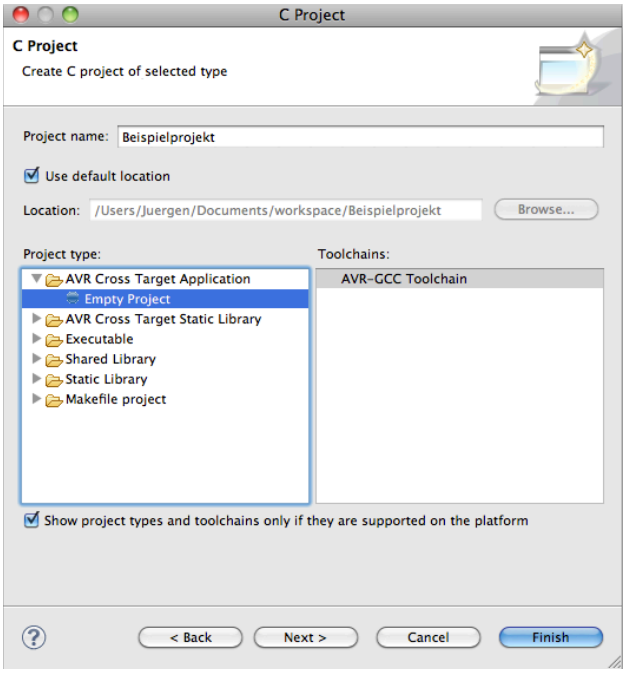

Abb.11: AVR Projekt Auswahl

Durch Bestätigen mit „Next“ wird wieder das nächste Menü angezeigt. Hier kann zwischen den Build-Konfigurationen gewählt werden. Da mit dem e-Stick nicht gedebuggt werden kann, wird hier nur der Punkt „Release“ angewählt.

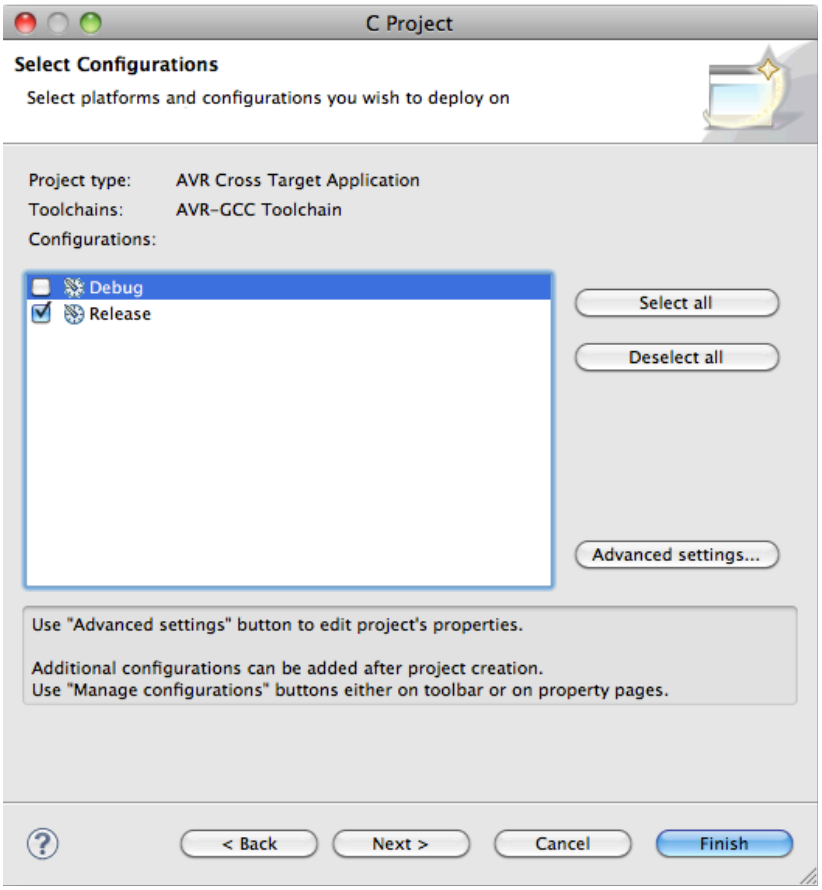

Abb.12: Build- Konfiguration

Nun muss nur noch der Button „Finish“ gedrückt werden um das Projekt anzulegen. Das Projekt ist jederzeit im Project Explorer einsehbar.

6.2 Anlegen der C-Source Datei

Um nun das C-Sourcefile für den späteren Programmcode anzulegen, muss über einen Rechtsklick auf das Projekt über „New“ und „Source File“ eine neue C Datei angelegt werden. Dieser Prozess ist bei H-Files analog.

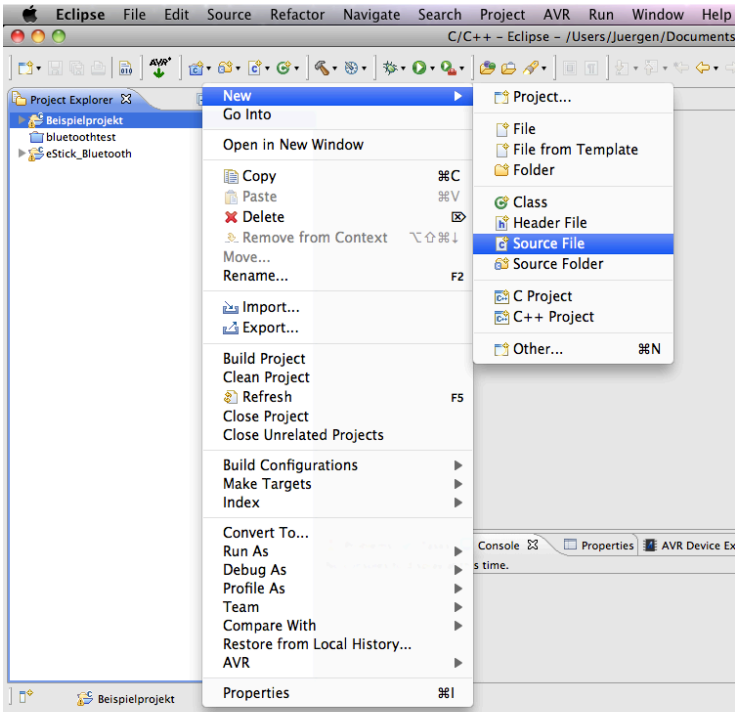

Abb.13: C-Sourcefile anlegen

In dem nun aufgegangenen Fenster muss nun nur noch der Name der C-Datei eingegeben und mit „Finish“ abgeschlossen werden.

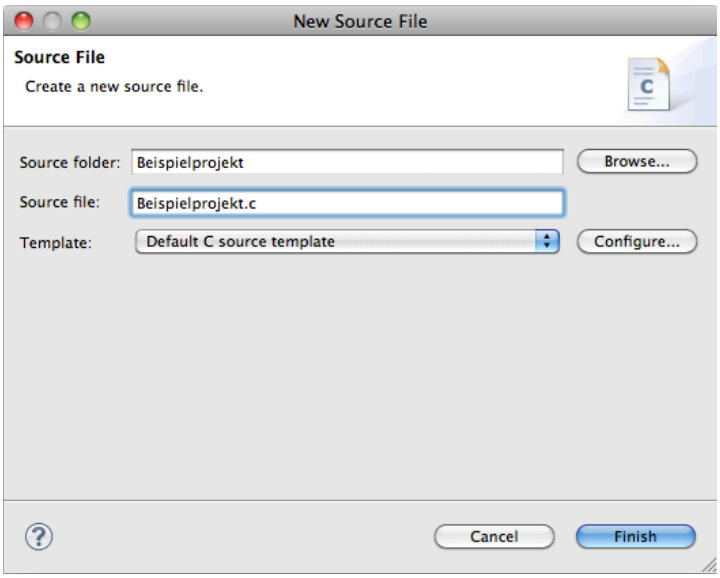

Abb.14: Source-File Konfiguration

6.3 Auswahl des Programmers (optional)

Diese Option würde benötigt werden, wenn der Mikrocontroller über einen externen Programmer programmiert werden würde. Um diesen für das Projekt einzurichten, muss dieser über AVRDude ausgewählt werden. Dazu wählt man durch einen Rechtsklick auf das Projekt den Punkt „Properties“.

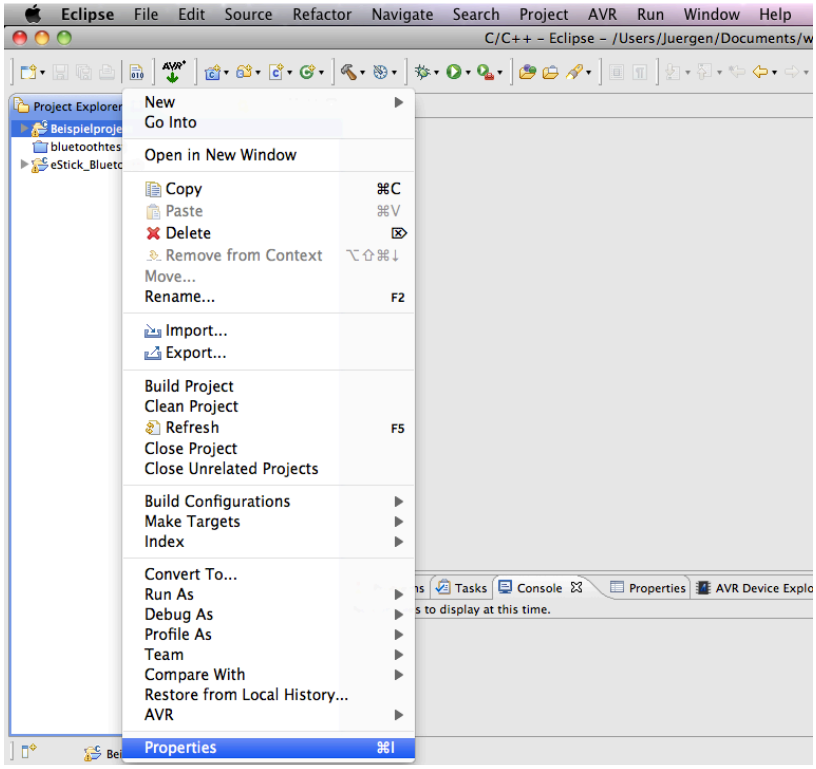

Abb.15: Projekteigenschaften

Anschließend daran geht ein Menü auf, wodurch der Eintrag „AVR“ und weiters der Eintrag „AVRDude“ gewählt werden kann. Dieser erlaubt die Auswahl des Programmers. Jedoch muss dieser vorher wie in Kapitel 4.3 beschrieben konfiguriert worden sein.

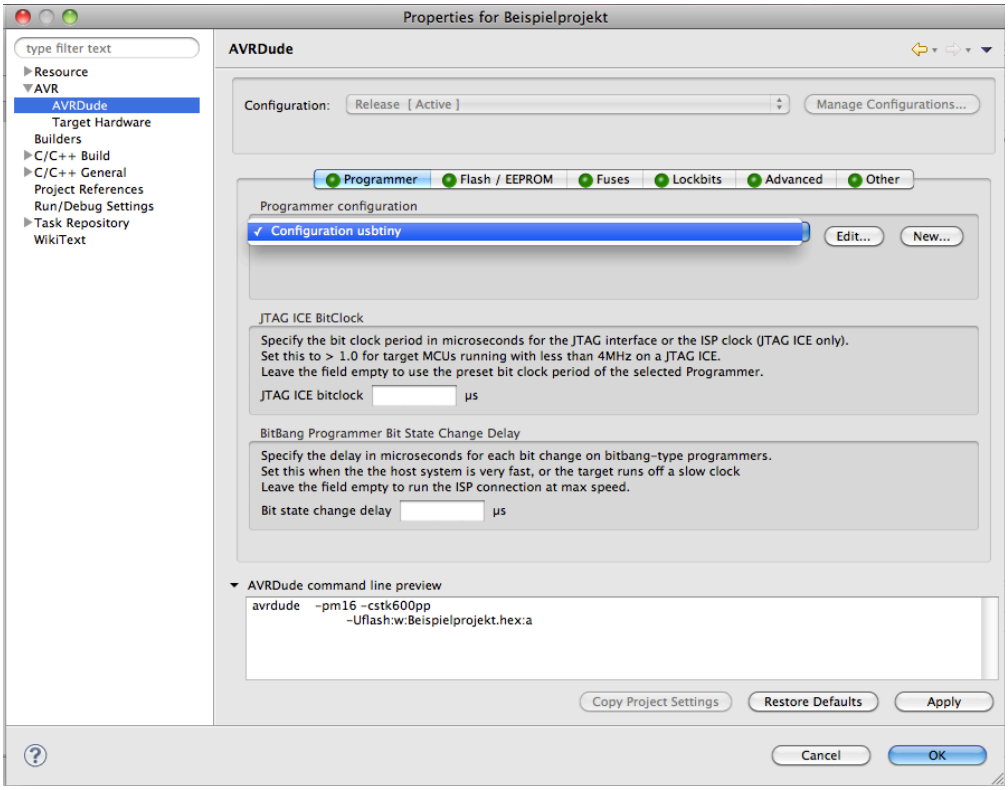

Abb.16: Programmer Auswahl

Weiters ist es anzuraten die FUSE-Bits im Tab „Fuses“ zu setzen. Diese können auch über ein Icon neben dem Eintrag „direct hex values“ vom Mikrocontroller ausgelesen werden. Zum Abschluss muss noch über den Unterpunkt „Target Hardware“ im AVR Menüpunkt, der Mikrocontroller ausgewählt und die Clockfrequenz gesetzt werden. Diese Werte können ebenfalls auch automatisch über „Load from MCU“ ausgelesen werden.

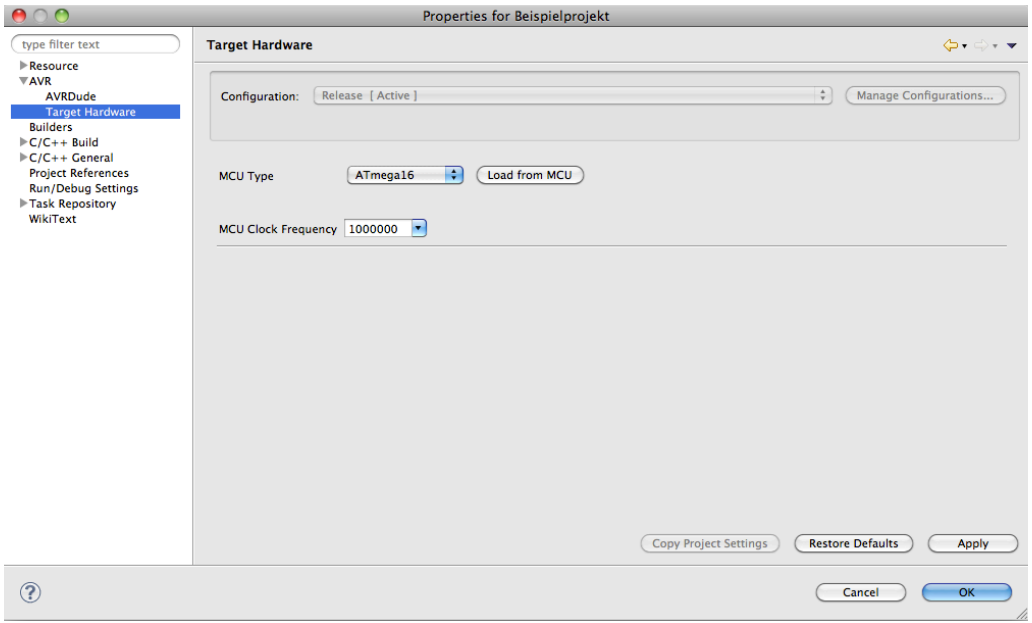

Abb.17: Target Hardware

6.4 Kompilieren

Das erstellte Projekt kann wenn alle Schritte befolgt wurden über das Hammer-Symbol kompiliert werden. Achtung: Auf die Richtige Konfiguration achten (Debug bzw. Release).

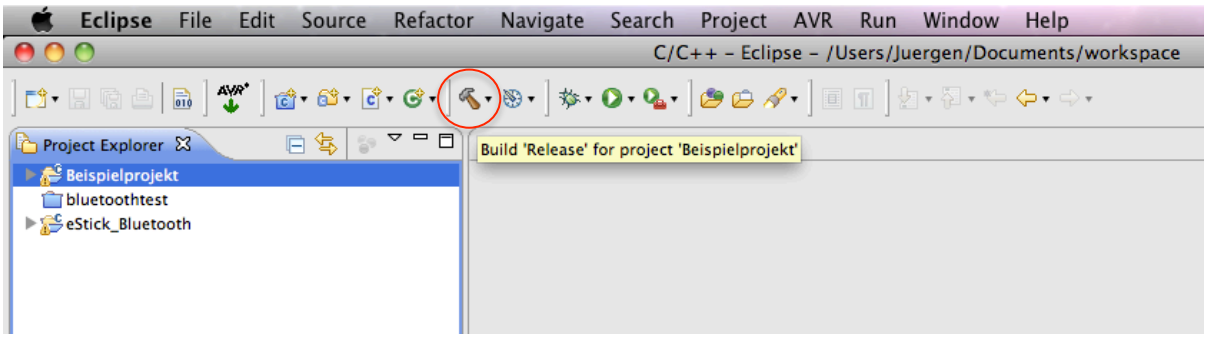

Abb.18: Kompilieren

6.5 Programmieren

Um das Projekt bei angeschlossenen Programmer auf den AVR zu laden, muss lediglich der grüne AVR Upload Button gedrückt werden. Diese Option ist beim e-Stick leider nicht verfügbar.

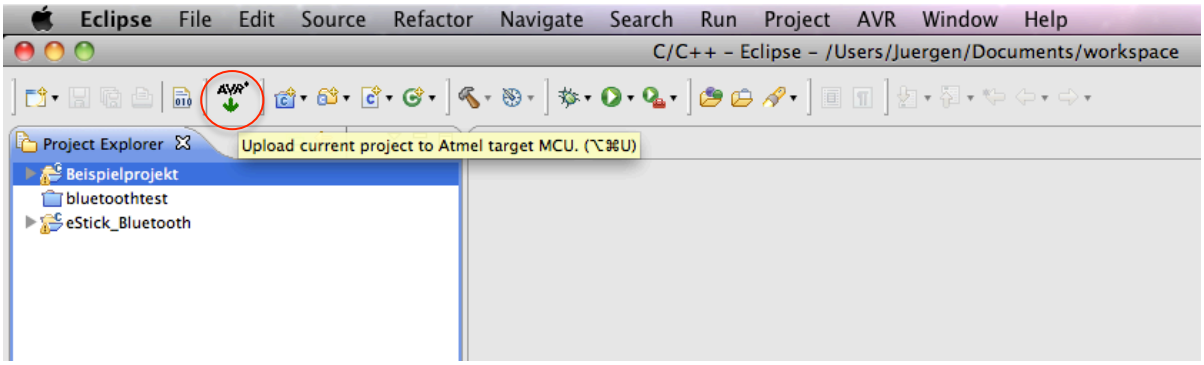

Abb.19: Programmieren des Mikrocontrollers

6.6 Flashen

Zum Flashen des Mikrocontrollers wird wie in Kapitel 5.2 beschrieben über „Programme“, „Dienstprogramme“, „Terminal“ ein Terminalfenster aufgerufen, über welches mit dem Befehl „cd Beispielverzeichnis/.../Hexfileordner“ zu dem Ordner navigiert wird in dem sich das erstellte Hexfile befindet. Unter Eclipse befindet sich dieses Verzeichnis standardmäßig im Hauptprojektverzeichnis und trägt den Namen „Release“ bzw. „Debug“.

Anschließend daran wird der e-Stick über den Befehl „dfu-programmer at90usb162 erase“ gelöscht, mit dem Befehl „dfu-programmer at90usb162 flash filename.hex“ geflasht und das Programm über den Befehl „dfu-programmer at90usb162 start“ gestartet.

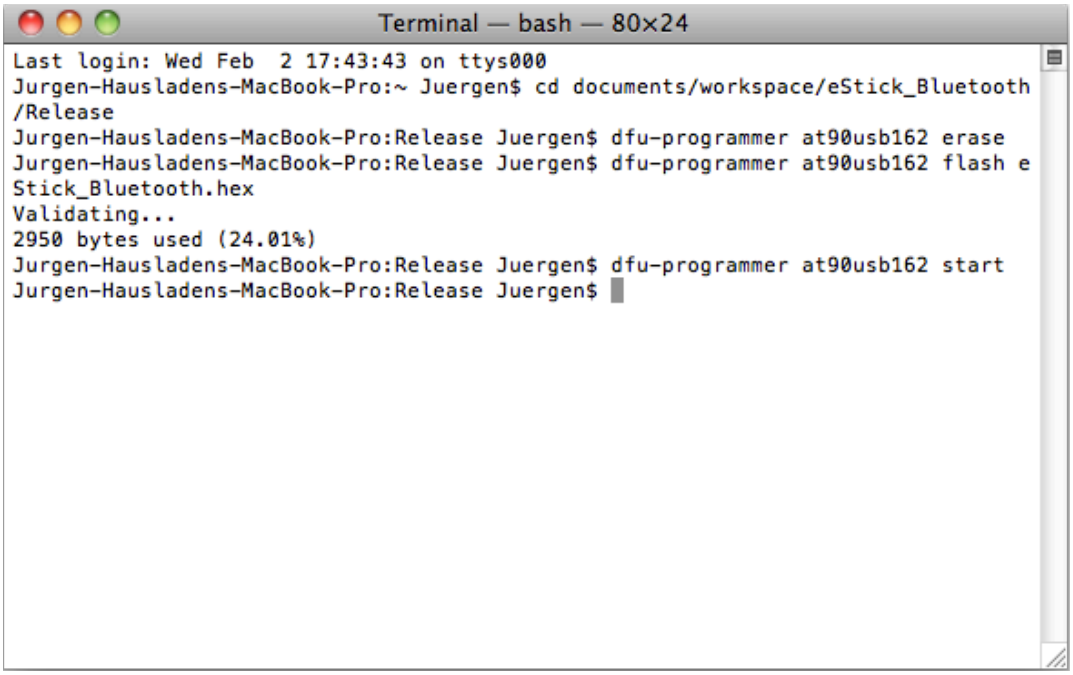

Abb.20: dfu-programmer Flashvorgang

Abbildungsverzeichnis