Einführung in die Projektarbeiten IU/IMR/PAUSS SS2014

Themen:

Ubuntu Grundlagen, Mira Grundlagen, SCITOS remote control, Erzeugen/Kompilieren/Einbinden/“Debuggen“ von Projekten für Mira

Dokumentation zur Projekteinführung

von

David Bongermino

05.05.14

Matrikelnummer, Kurs	732526, MECM2
Professor	Prof. M. Rätsch

Kurzfassung

Die vorliegende Dokumentation dient als Einführung in die Bearbeitung der Projekte am SCITOS G5 mit Mira und Ubuntu 12.04 der Pflichtvorlesung Image Understanding (IU, auch Bildverarbeitung), sowie den Wahlpflichtvorlesungen Interaktive Mobile Roboter (IMR) und Personalisierte Assistenz und Service Systeme (PAUSS).

Begonnen wird mit einer Kurzeinführung zu den wichtigsten Hilfsmitteln. Danach folgen die allgemeinen Grundlagen zu Mira, woraufhin kurz die remote Steuerung des SCITOS dargestellt wird.

Den Hauptteil ein Tutorial zu Projekten ein. Es wird dargestellt wie Projekte erzeugt werden, wie diese kompiliert werden und wie sie in den Start von Mira eingebunden werden. In diesem Zusammenhang wird eine Debug-Empfehlung dargestellt.

Im Literaturverzeichnis sind Links für weitere Recherchemöglichkeiten gegeben.

Inhaltsverzeichnis

1 Einleitung

1.1 Problemstellung und Zielsetzung

Zur Bearbeitung verschiedener Projekte aus den Vorlesungen IU, IMR und PAUSS sind gewissen Grundlagen in Ubuntu und Mira nötig.

In vorliegender Dokumentation wird mit einer Kurzeinführung zu den wichtigsten Hilfsmitteln begonnen. Danach folgen die allgemeinen Grundlagen zu Mira, woraufhin kurz die remote Steuerung des SCITOS dargestellt wird.

Den Hauptteil nimmt ein Tutorial zu Projekten ein. Es wird dargestellt wie Projekte erzeugt werden, wie diese kompiliert werden und wie sie in den Start von Mira eingebunden werden. In diesem Zusammenhang wird eine Debug-Empfehlung dargestellt. Im Literaturverzeichnis sind Links für weitere Recherchemöglichkeiten gegeben.

Die vorliegende Dokumentation wird nach bestem Wissen und Gewissen erstellt. Es besteht kein Anspruch auf Richtigkeit und Vollständigkeit.

1.2 Projektumfeld

Die Dokumentation und Umsetzung der Beispiele orientiert sich an folgenden Randbedingungen:

- Serverseitig:
  - SCITOS G5
  - Installierte Software:
    - Ubuntu 12.04 64 Bit,
    - Mira
- Clientseitig:
  - VMware Player 6.0.2 build-1744117
  - VMware Image „Ubuntu 64-bit.vmx“
    - Vom 24.04.2014
    - Ordnerbezeichnung: Ubuntu 64-bit_BU20140419_Student_NeuInstMiraEclipseQTCreator
  - Installierte Software:
    - Ubuntu 12.04 64 Bit,
    - Mira,
    - Eclipse,
    - QTCreator

2 Die wichtigsten Hilfsmittel

2.1 SCITOS

Um den SCITOS zu starten, muss der Schlüsselschalter gedreht werden (vgl. Nr.1 in Abbildung 1) und anschließend durch Drücken des Drehschalters (vgl. Nr. 2 in Abbildung 1) „Start Robot & PC“ ausgewählt (vgl. Abbildung 2) werden.

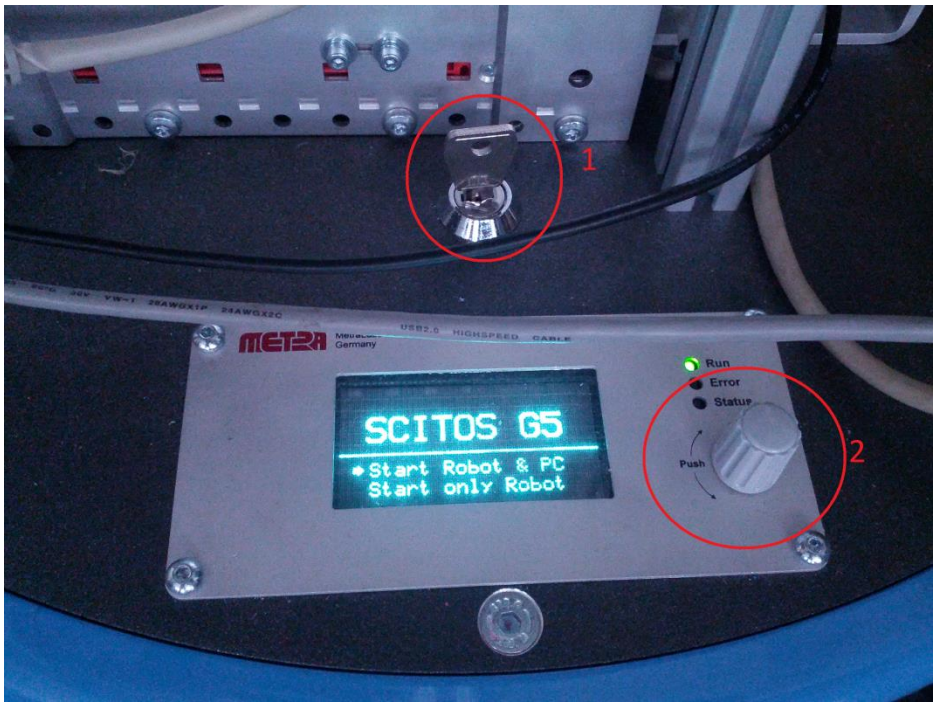


Abbildung 1: SCITOS Schlüssel und Taster

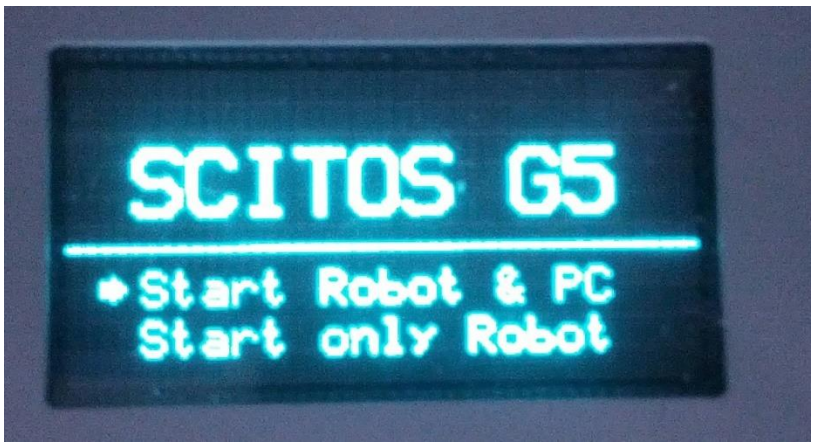


Abbildung 2: SCITOS Display

2.2 VMware Player

Zum Starten des VMware – Images mit Ubuntu 12.04 und Mira ist es nötig einen VMware – Player installiert zu haben. Dieser ist z.B. von der Homepage von vmware (http://www.vmware.com) herunter zu laden und zu installieren.

Um ein VMware –Image zu starten, muss nach der Installation die entsprechende Datei über den „Open a Virtual Machine“ Dialog (vgl. Abbildung 3) geöffnet werden und anschließend über den „Play virtual machine“ Button (Abbildung 4) gestartet werden. Über den „Edit virtual machine settings“ Button (Abbildung 4) können Pfade und Ressourceneigenschaften angepasst werden.

Zum Entwickeln der Projekte soll unbedingt von der VMware aus programmiert werden und nicht auf dem SCITOS direkt!

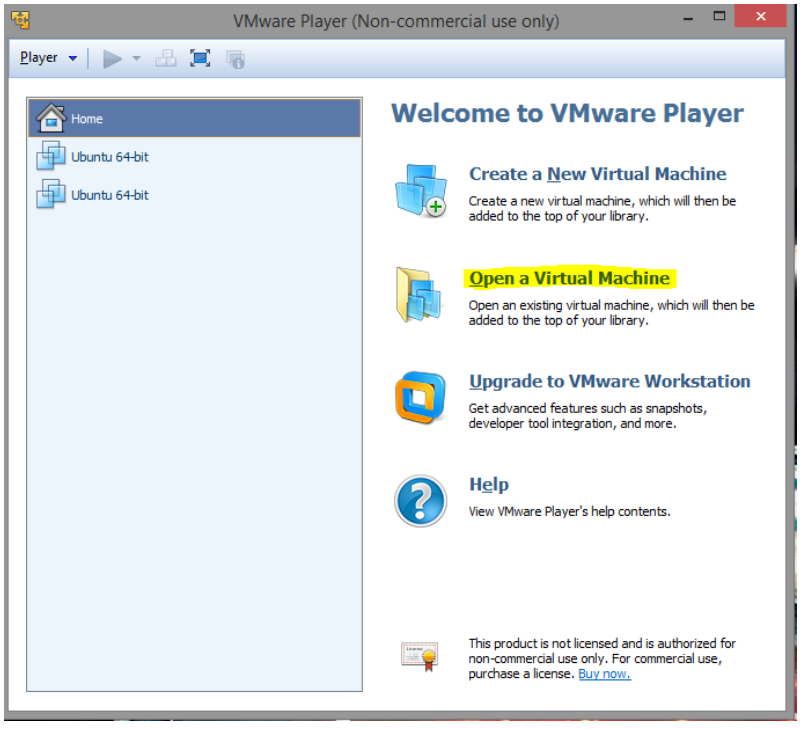


Abbildung 3: VMware - Open a Virtual Machine

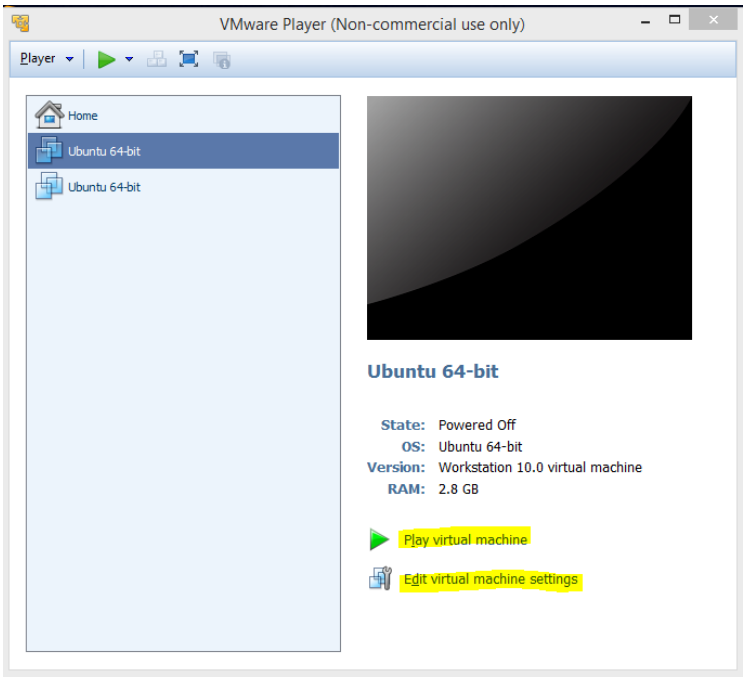


Abbildung 4: VMware - Play / Edit VMware

2.3 Ubuntu

2.3.1 Terminal

Auch wenn Ubuntu immer intuitiver bedienbar wird, wird Ubuntu in vielen Bereichen über Konsoleneingaben gesteuert. Für diese Steuerung kann z.B. das „Terminal“ verwendet werden. Dieses lässt sich starten indem

- Drücken der „Windows“ – Keyboard – Taste und anschließender Eingabe von „Terminal“, oder
- Drücken des „Dash-Home“ – Softbuttons in Ubuntu und anschließender Eingabe von „Terminal“

Das Terminal sieht folgendermaßen aus:

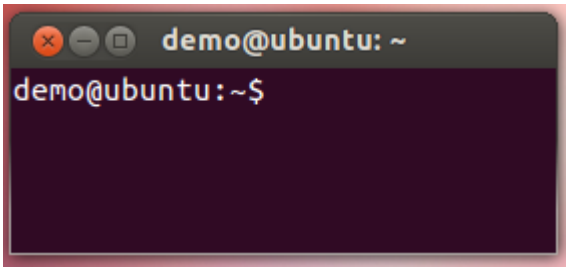


Abbildung 5: Terminal

Ubuntu wird standardmäßig als Benutzer „demo“ ausgeführt. In Ubuntu gibt es einen Systemverwalter „root“ der unter Windows mit dem Administrator vergleichbar ist.

Hinweis: Das Administratorkennwort in der VM lautet „3“.

In nachfolgender nicht vollständiger Tabelle sind für die Projektbearbeitung die wichtigsten Befehle zusammengefasst.

Eingabe	**Beschreibung**
sudo -s	Vom lokalen User „demo“ in den Systemverwalter „root“ wechseln. Hinweis: Passwort = 3
sudo –nautilus bzw. wenn bereits sudo nautilus	Öffnen eines Explorers als root user. Z.B. zum Editieren schreibgeschützter Dateien. Hinweis: der Zugriff ist nicht umsonst passwortgeschützt!
miracenter	Starten des Miracenters (GUI) ohne zusätzliche Konfigurationen
mirawizard	Starten des Mirawizards (GUI) zum Erstellen von Externen Projekten, Toolboxen, Domains oder Komponenten.
mirapackage	Starten des Mirapackages (GUI): Software zum Installieren von Komponenten für Mira (i.d.R. nur für Administratoren nötig)
sudo bash ./dateiname.sh	„Bashen“ einer Datei. Installation einer .sh Datei.
sudo apt-get install SOFTWARENAME	Installation der Software „SOFTWARENAME“
sudo mv SOURCE TARGET	Verschieben einer Quelle „SOURCE“ zum Ziel „TARGET“
ls	Anzeige der im aktuellen Pfad befindlichen Dateien und Ordner
lsusb	Liste der Usb-Geräte
cd PFAD/	In den Pfad PFAD wechseln
history	Anzeige der eingegeben Befehle (auch in der versteckten Datei .bahs_history auslesbar).
make	„Baut“ ein Projekt falls benötigte Dateien vorhanden
ifconfig	Anzeige der IP

Tabelle 1: Übersicht verschiedener Konsolenbefehle

2.3.2 .bashrc

Im Laufe der Erzeugung von Projekten oder bei Installationen kann es nötig sein Anpassungen in der .bashrc vorzunehmen. Diese Datei ist eine im „Home

Folder“ versteckte Konfigurationsdatei, welche beim Start des Terminals mitgeladen mit. Hierdurch werden z.B. Pfade für Mira mitgeladen.

Versteckte Dateien können in einem Ordner durch die Tastenkombination „Strg + H“ angezeigt werden.

2.4 Eclipse

Auf dem VMware-Image ist Eclipse DIE for C/C++ Developers Version Kepler Service Relase 2 installiert. Eclipse kann über das Icon auf dem Hauptdesktop gestartet werden.

Auf die Bedienung von Eclipse oder die C++ Programmierung wird nicht weiter eingegangen.

2.5 QTCreator

Der QTCreator ist ein Editor der in späterer Betrachtung vor allem beim „Debugging“ verwendet wird. Ein großer Vorteil des Editors ist es, dass ein Syntax-Highlighting für C++ existiert. Ebenso sind klassische einfache Programmierfehler oder eine „Intellisense“ (= Assistenz zur schnellen Eingabe vorhandener Befehle) wie bei einer klassischen Programmierumgebung vorhanden.

3 MIRA und remote Control des SCITOS

3.1 Miracenter

Das Miracenter kann durch den Befehl „miracenter“ gestartet werden. Soll hierbei eine Verbindung zum SCITOS aufgebaut werden, muss auf dem SCITOS das Miracenter mit dem Befehl:

„miracenter /opt/MIRA-commercial/domains/robot/SCITOSConfigs/etc/SCITOS-Pilot.xml –var MCFFile=/home/demo/etage1/etage1.mcf –k“

gestartet werden. Daraufhin kann sich ein Client oder mehrere Clients über ein VMware – Image durch den Befehl:

„miracenter –k 192.168.1.24:1234“

auf dem SCITOS anmelden. Hierbei muss ggf. die IP durch die IP des SCITOS ersetzt werden. In der Regel stimmt der Port 1234 jedoch überein.

Das Miracenter ist in folgender Abbildung zu sehen:

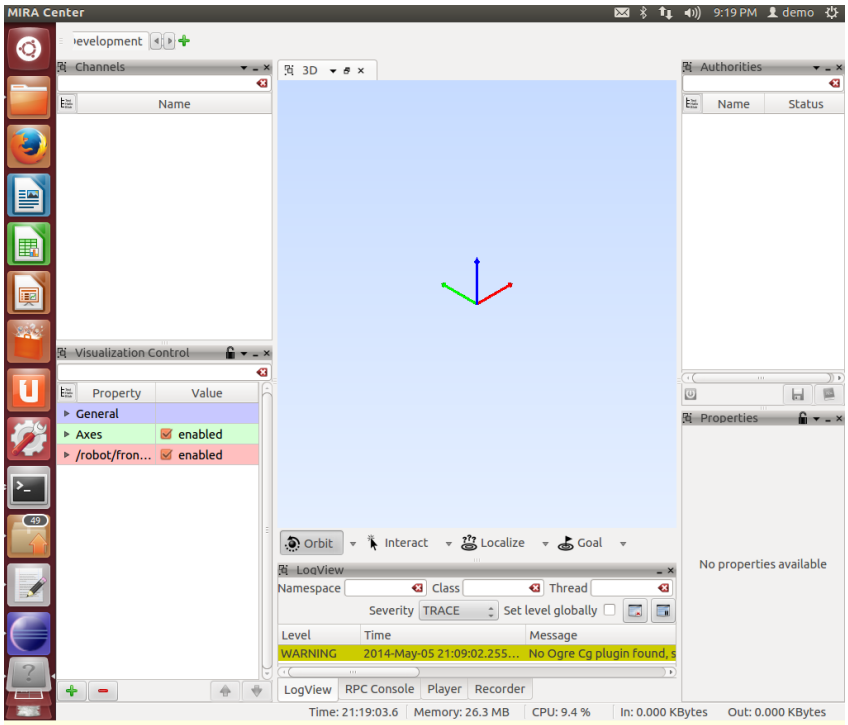


Abbildung 6: Miracenter

Um weitere Ansichten hinzuzufügen müssen diese durch den Dialog „Choose a view“ hinzugefügt werden. Dieser kann durch die Tastenkombination „Strg + D“ bzw. über das Fenster Window>Show View aufgerufen werden:

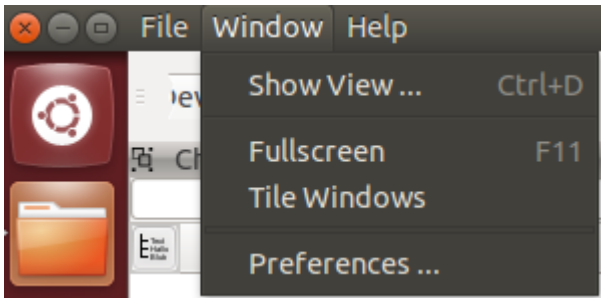


Abbildung 7: Aufruf des Show View Dialogs

Hierdurch kann z.B. die „Drive View“ Ansicht geladen werden, mit der man die Antriebe des Roboters remote steuern kann.

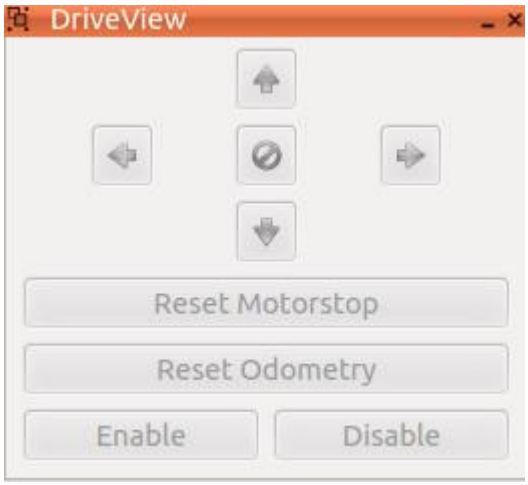


Abbildung 8: Drive View

Hinweis: Der letzte Befehl der am Roboter ankommt wird ausgeführt.

Hinweis: Die Antriebe sind ggf. sehr schnell eingestellt. Eine Kollision sollte auf jeden Fall vermieden werden!

Hinweis: Eine Kartenansicht über den VMware – Client ist nicht möglich.

4 Tutorial Anlegen eines Projektes

4.1 Allgemeines

Indem man im Terminal „mirawizard“ eingibt, wird ein Dialog gestartet mit dem die folgenden Dialoge aufgerufen werden können:

- Create a new External Project,
- Create a new Toolbox,
- Create a new Domain,
- Create a new Component (Unit, Visualization, View, etc.).

Die Beschreibung der Auswahlmöglichkeiten befindet sich auf der offiziellen Website: http://www.mira-project.org/MIRA-doc/ConceptsPage.html

4.2 Grundlagen: Projekterstellung und Einbindung ins Miracenter

- Terminal öffnen und „mirawizard“ eingeben
- „Create a new External Project“ wählen

Abbildung 9: Mira Wizard[1]

[1] Quelle: http://www.mira-project.org/MIRA-doc-devel/TutorialCreateProjectPage.html Einsicht: 05.05.2014

- Projektname eingeben und ggf. Pfad anpassen

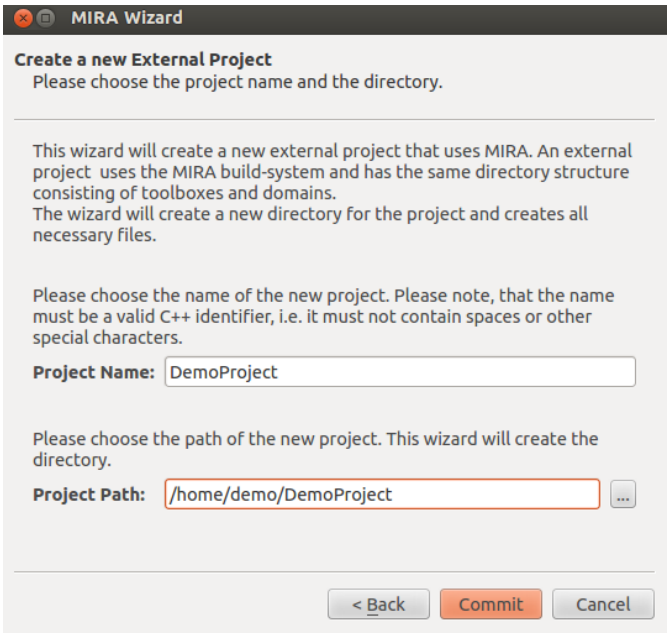


Abbildung 10: Mira Wizard – Create a new External Project

- Checkbox „set environment variables automatically“ setzen. (Hinweis: hierdurch werden in die .bashrc~ die Daten gesetzt).

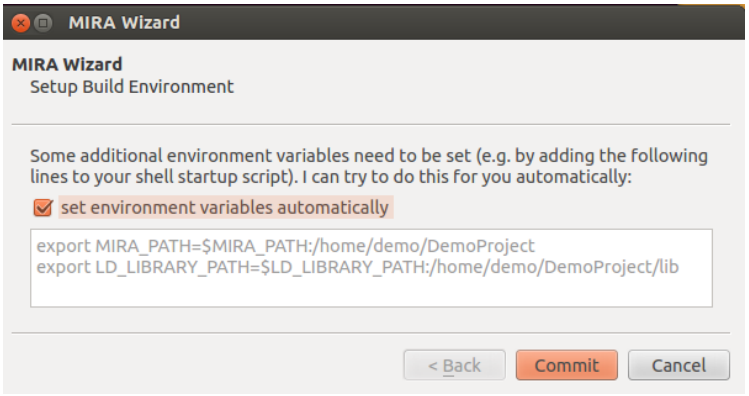


Abbildung 11: Mira Wizard: enviroment variables

- Hinweis quittieren

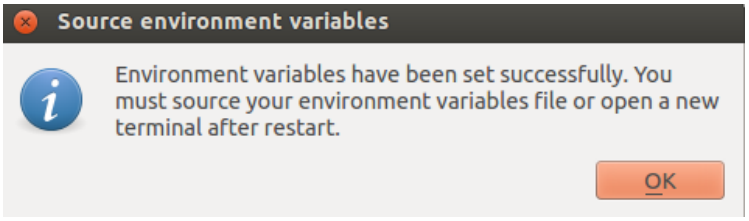


Abbildung 12: Mira Wizard: environment variables: Quittierung

- Auswahl einer Domain und einer Unit

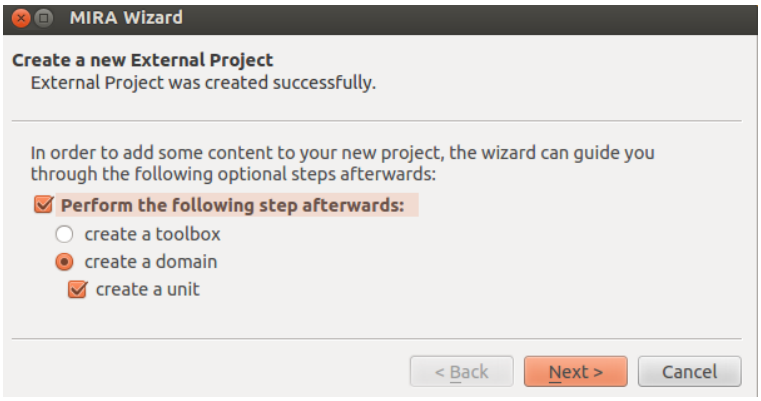


Abbildung 13: Mira Wizard: Domain und Unit auswählen

- Domain erstellen

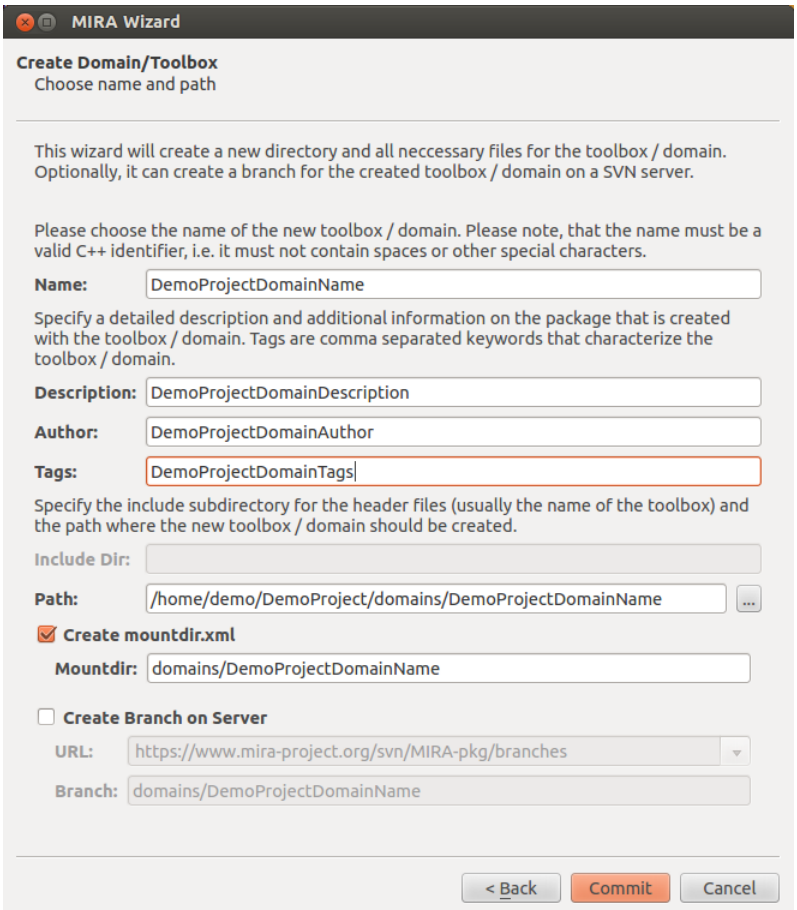


Abbildung 14: Mira Wizard: Domain erstellen

- Unit erstellen

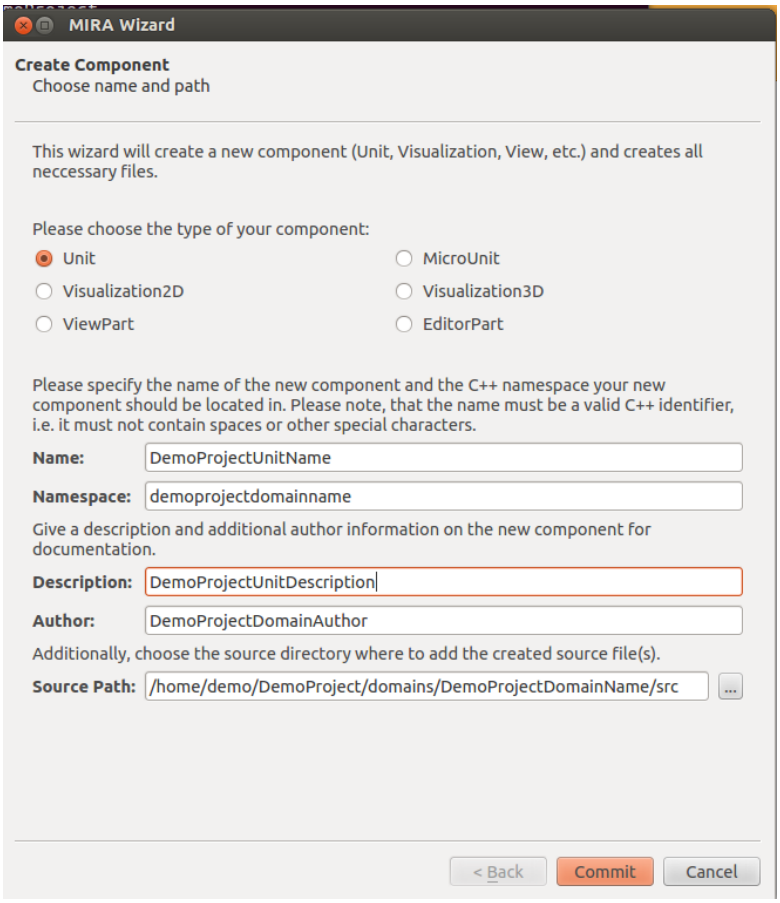
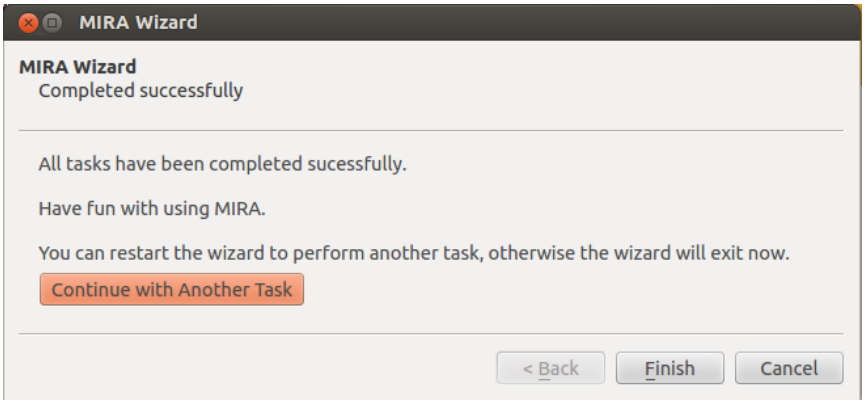


Abbildung 15: Mira Wizard: Unit erstellen

- Mira Wizard beenden



- Projekt „bauen“:
  - cd DemoProject/
  - make
- Nachfolgende Ausgabe sollte erscheinen: (besonders auf „[100%] Build…“ ist zu achten).

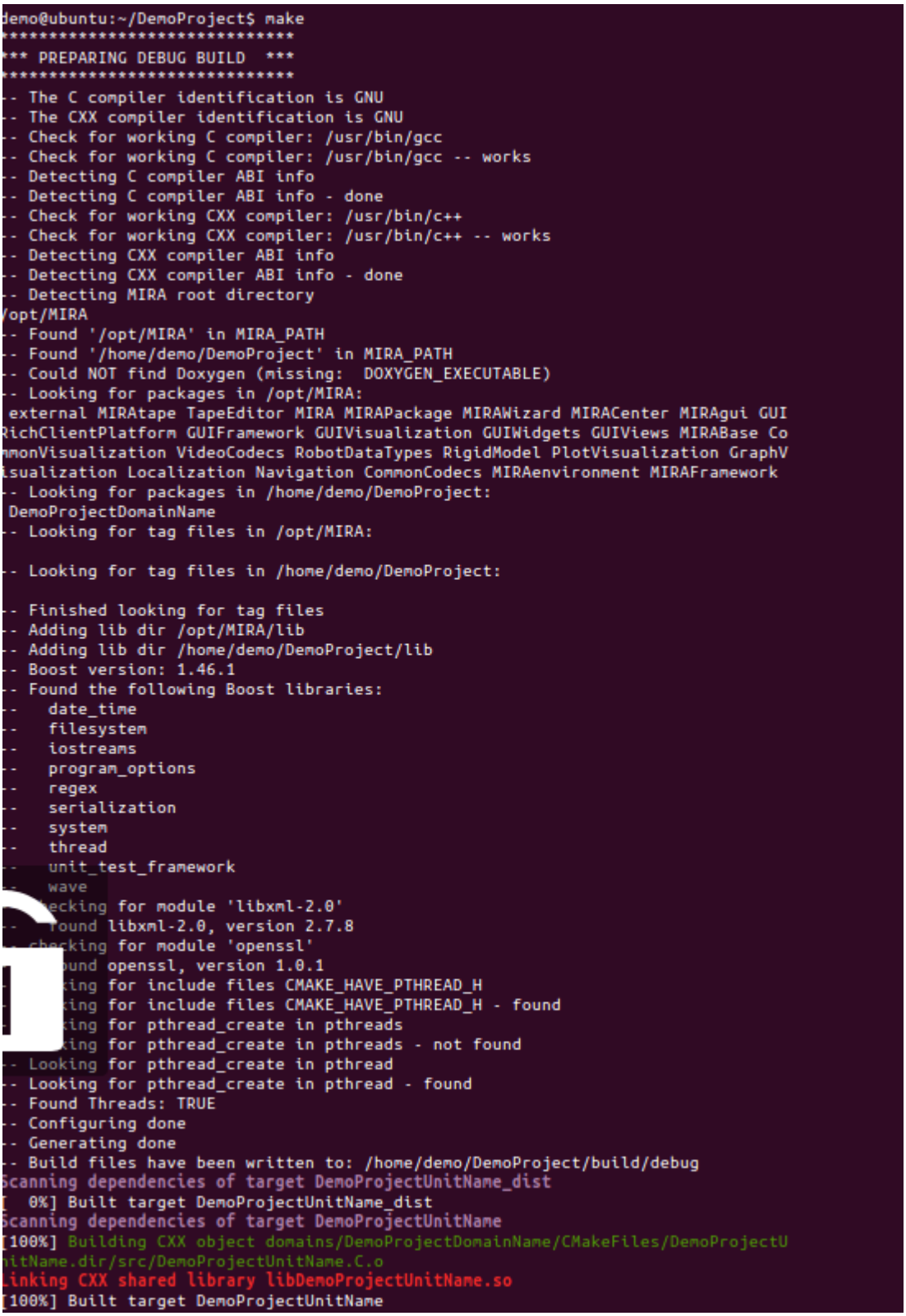


Abbildung 16: Ausgabe Make Project

- Um das neue Projekt im miracenter zu laden, muss noch eine Konfigurationsdatei angelegt werden. Diese kann z.B. im Pfad „/home/demo/DemoProject/domains/DemoProjectDomainName“ erstellt werden. Sie sollte möglichst entsprechend dem Projektnamen bezeichnet werden, hier „DemoProjectUnitName.xml“. Um sie zu editieren ist der Text Editor zu verwenden. Folgendes ist einzutragen:

```
<root>
        <unit class="demoprojectdomainname::DemoProjectUnitName" id="DemoProjectUnitName" />
</root>
```

Abbildung 17: XML Konfigurationsdatei

Hinweis: Die Zusammenhänge zwischen Namespace, Klassenname und ID sind zu berücksichtigen, welche durch die Eingaben bei der Erstellung von Projekt und Unit selbst definiert wurden!

- Starten des Miracenters mit dem neuen Projekt durch den Befehl „miracenter /home/demo/DemoProject/domains/DemoProjectDomainName/src/DemoProjectUnitName.xml“

- Prüfen ob das Projekt erfolgreich geladen wurde durch:
    - Keine Fehlermeldungen im Terminal
    - Korrekte Darstellung im Miracenter (Status: OK)

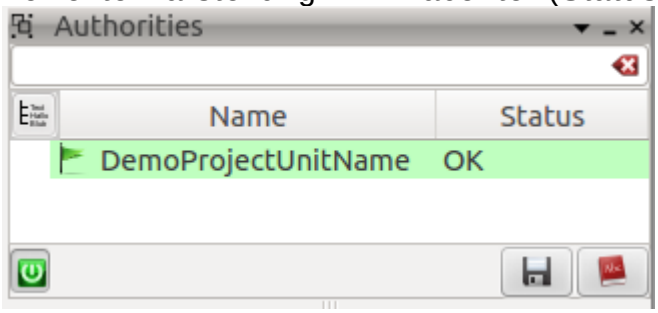


Abbildung 18: Miracenter: Status Angelegtes Projekt

Hinweis: „Authorities“ ist durch den „Choose a view“ Dialog (Strg + D) einblendbar.

Hinweis: Durch den „Resume/pause the selected authority“ Button (links unten) kann die Unit aktiviert bzw. pausiert werden.

4.3 Programmierung von Units

Nachdem das Projekt erstellt wurde, kann mit der tatsächlichen Programmierung begonnen werden. Hierzu muss die erstellte Unit „/home/demo/DemoProject/domains/DemoProjectDomainName/DemoProjectUnitName.c“ geöffnet werden. Defaultmäßig werden die Dateien mit dem QTCreator geöffnet.

Die erstellten Units besitzen einigen Templatecode. Selbsterklärend wird die Funktion „initialize“ bei der Initialisierung ausgeführt und die Funktion „process“ zyklisch ausgeführt. Die Zykluszeit wird durch die Zahl in der Zeile „DemoProjectUnitName::DemoProjectUnitName() : Unit(Duration::milliseconds(250))“ angegeben.

Empfehlung: Verwendung der Zykluszeit 250 ms anstelle der Defaultzeit 100 ms.

Empfehlung: Eclipse wurde beabsichtigt installiert! Eine sinnvolle Programmierung (sauberes Debugging etc.) kann nur in einer richtigen Programmierumgebung erfolgen. Deshalb: Programmierung des Codes in C/C++ mit Eclipse und anschließender Transfer mithilfe von QTCreator in die entsprechende Unit.

Das Beispielprojekt wird um folgenden Code erweitert:

```
DemoProjectUnitName::DemoProjectUnitName() : Unit(Duration::milliseconds(250))
{
   cout << "In Funktion DemoProjectUnitName::DemoProjectUnitName() :
               Unit(Duration::milliseconds(250))";
}

void DemoProjectUnitName::initialize()
{
   cout << "In Funktion void DemoProjectUnitName::initialize()";

}

void DemoProjectUnitName::process(const Timer& timer)
{
   cout <<"In Funktion void DemoProjectUnitName::process(const Timer& timer)";
}
```

Hinweis: Zur Nutzung des „cout“ Befehls ist es nötig die Unit um folgendes zu erweitern:

- using namespace std,
- #include <stdio.h>

Hinweis: Nachdem eine Unit editiert wurde, muss sie wie in vorherigem Kapitel beschrieben mit dem Befehl „make“ neu gebaut werden und das Miracenter neu gestartet werden.

Durch den Start des Miracenters sieht man im Terminal eine veränderte Ausgabe:

```
demo@ubuntu:~$ miracenter /home/demo/DemoProject/domains/DemoProjectDomainName/src/DemoProjectUnitName.xml

In Funktion DemoProjectUnitName::DemoProjectUnitName() : Unit(Duration::milliseconds(250))
In Funktion void DemoProjectUnitName::initialize()
In Funktion void DemoProjectUnitName::process(const Timer& timer)
In Funktion void DemoProjectUnitName::process(const Timer& timer)
In Funktion void DemoProjectUnitName::process(const Timer& timer)
In Funktion void DemoProjectUnitName::process(const Timer& timer)
In Funktion void DemoProjectUnitName::process(const Timer& timer)
In Funktion void DemoProjectUnitName::process(const Timer& timer)
In Funktion void DemoProjectUnitName::process(const Timer& timer)
In Funktion void DemoProjectUnitName::process(const Timer& timer)
In Funktion void DemoProjectUnitName::process(const Timer& timer)
In Funktion void DemoProjectUnitName::process(const Timer& timer)
In Funktion void DemoProjectUnitName::process(const Timer& timer)
demo@ubuntu:~$
```

Abbildung 19: Terminalausgabe nach Programmierung

Hierdurch wird die Abarbeitungsreihenfolge des Codes deutlich.

Hinweis: Eine Ausgabe über „printf()“ ist analog zu „cout“ möglich.

4.4 Erweiterung: Debugging von Projekten

Um den programmierten Code zu Debuggen gibt es prinzipiell zwei Empfehlungen:

- Bei „Layoutfehlern“: Durch den make Befehl werden Fehler ausgegeben, z.B. falls Abhängigkeiten nicht gefunden werden können. Vergleichbar mit sonstigen Compilern.

- Bei „Programmierfehlern“: Arbeiten mit „cout“ um Werte, Inhalte, Zustände und Positionen im Code in der Konsole auszugeben zur nachverfolgung des Codes.

4.5 Erweiterung: Hinzufügen von Referenzen und Sourcen

Um Abhängigkeiten zu externen Bibliotheken (libs) oder Quellen (sourcen) zu schaffen, ist es nötig die Konfigurationsdatei „CMakeLists.txt“ aus dem Ordner „/home/demo/DemoProject/domains/DemoProjectDomainName“ anzupassen. Vorsicht: Im übergeordneten Ordner darf diese Anpassung nicht stattfinden, auch wenn es dort eine gleichnamige Datei gibt!

Eine Anpassung kann folgendermaßen aussehen:

```
##########################################

MIRA_ADD_LIBRARY(DemoProjectUnitName
        SHARED
        #PACKAGE $PACKAGE$
        SOURCE
                src/DemoProjectUnitName.C
                src/Test.c

        LINK_LIBS
                MIRABase
                MIRAFramework
                opencv_core
                zbar
                opencv_imgproc
```

Abbildung 20: Anpassungen für externe Referenzen und Sourcen

5 Literatur

1. Wiki: http://wiki.ubuntuusers.de/Bash/bashrc
2. Organisation: http://www.mira-project.org/MIRA-doc-devel/TutorialCreateProjectPage.html
3. Forum: http://www.mira-project.org/osqa/

Abkürzungen

Zeichen	Bedeutung
IMR	Interaktive Mobile Roboter
IU	Image Understanding
PAUSS	Personalisierte Service und Assistenzsysteme

Tabelle 2: Abkürzungen

Abbildungsverzeichnis

Tabellenverzeichnis