A4MCAR Introduction and Setup Manual

Release 1.0

Mustafa Özçelikörs

Nov 02, 2017

Contents:

1 Introduction **1**
- 1.1 Scope . . . 1
- 1.2 What is A4MCAR? . . . 1
- 1.3 What is APP4MC? . . . 3
- 1.4 Objectives of A4MCAR Project . . . 5
- 1.5 A4MCAR Methodology and Features . . . 6
- 1.6 Future Developments . . . 7

2 Google Summer of Code Contribution **9**
- 2.1 Development Workflow . . . 10

3 Accessing the Project **11**
- 3.1 Google Summer of Code Page . . . 11
- 3.2 Source Code . . . 11
- 3.3 Documentations . . . 11
- 3.4 External Links . . . 12

4 Hardware Setup **13**

5 Software Setup **15**
- 5.1 Downloading A4MCAR applications from the Git repo . . . 15
- 5.2 Provided Filesystem . . . 15
- 5.3 High-Level Applications Setup . . . 16

6 Indices and tables **21**

Bibliography **23**

CHAPTER 1

Introduction

1.1 Scope

This application manual is dedicated to explaining how to set up applications on A4MCAR. For more detailed documentation towards the research goals and tool support, it is advised to check our research documentation.

Note: If you are looking to **download the project** or **see the comprehensive information** please jump to *Accessing the Project*.

1.2 What is A4MCAR?

A4MCAR is a distributed and multi-core demonstrator RC-Car that is used in the demonstration of parallel applications on real embedded systems. It is originally developed to test and evaluate APP4MC tool. A4MCAR is a project that is developed in FH Dortmund IDiAL and is supported under Project AMALTHEA4public. The project is also granted by The Eclipse Foundation and Google Inc. during the participation to Google Summer of Code 2017 event. During the event, the developed applications have been distributed in open-source manner under Eclipse Public License (EPL). Thus, we hope to help those in the open-source community who are willing to address concurrency in their multi-core embedded applications with the help of the GSoC event.

The demonstrator system features not only low level functionalities such as sensor and motor driving but also high level features such as image processing, camera streaming, server-based wireless driving via Web, bluetooth connectivity via Android application, system core monitoring and analysis features and touchscreen UI. Our experiments along the multi-task heterogeneous demonstrator A4MCAR show that using APP4MC results instead of OS-based or sequential implementations on a distributed heterogeneous system significantly improves its responsiveness in order to potentially reduce energy consumption and replaces error prone manual constraint considerations for mixed-critical applications.

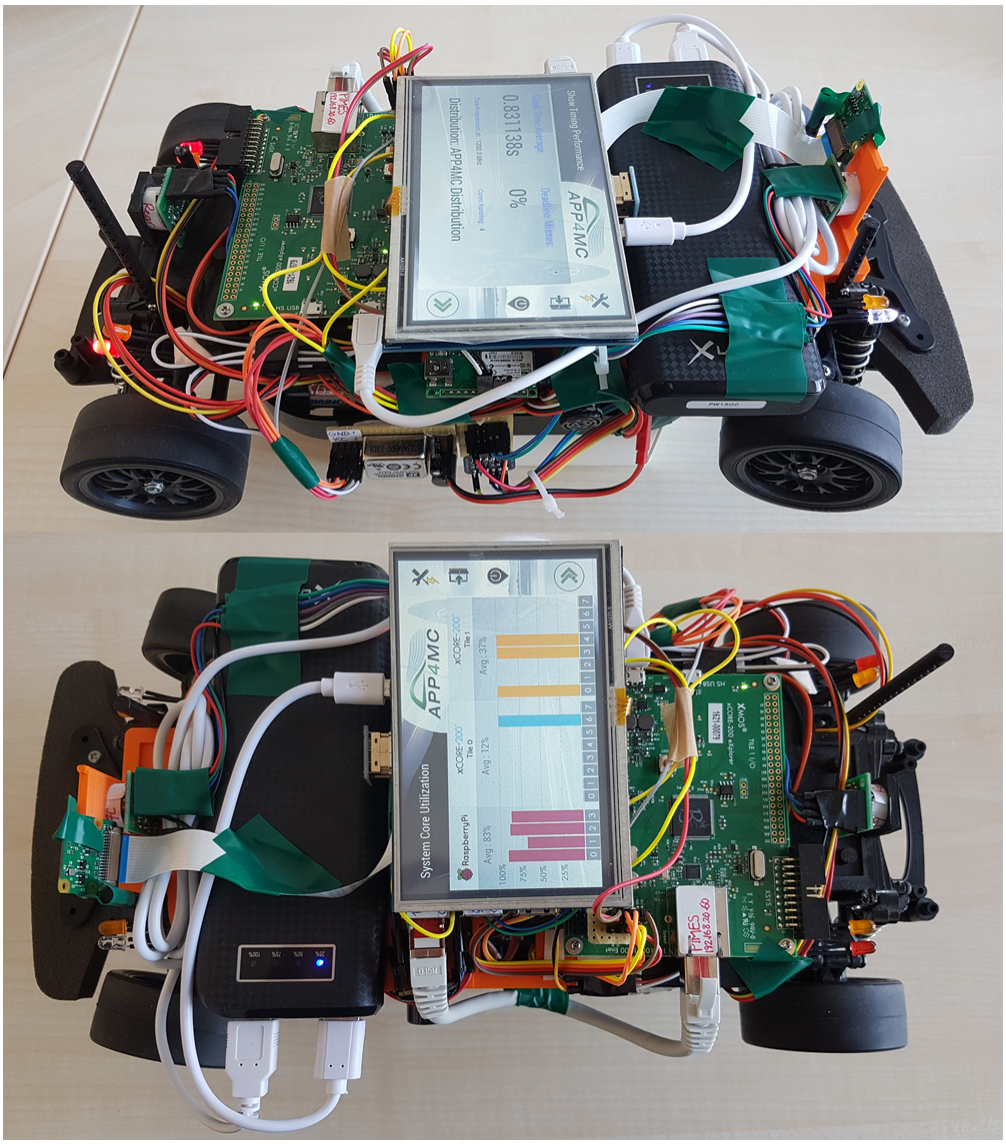


Fig. 1.1: A4MCAR

1.3 What is APP4MC?

While there exist some commercial tools that provide easement in the parallel software development, recent study done in Germany, namely AMALTHEA4public, aims to provide planning and tracing tools especially for multi-core developments in automotive domain with several open source development tools.

The APP4MC project provides an Eclipse-based tool chain environment and a de-facto model standard to integrate tools for all major design steps in the multi- and many-core development phase. A basic set of tools are available to demonstrate all the steps needed in the development process. The APP4MC project aims at providing according to *[App4mcProposalEclipse]*:

- A basis for the integration of various tools into a consistent and comprehensive tool chain.
- Extensive models for timing behaviour, software, hardware, and constraints descriptions (used for simulation / analysis and for data exchange).
- Editors and domain specific languages for the models.
- Tools for scheduling, partitioning, and optimizing of multi- and many-core architectures.

APP4MC targets multi-core and many-core platforms, while the main focus is the optimization of embedded multi-core systems . Due to its focus, APP4MC is partnered with many automotive OEMs and part suppliers that deal with embedded software engineering. Furthermore, it supports interoperability and extensibility and unifies data exchange in cross-organizational projects. Additionaly, since APP4MC uses Eclipse platform to its purposes, the development environment has a complete open-source nature under Eclipse Public License (EPL).

Eclipse APP4MC platform editor window can be seen in the following figure:

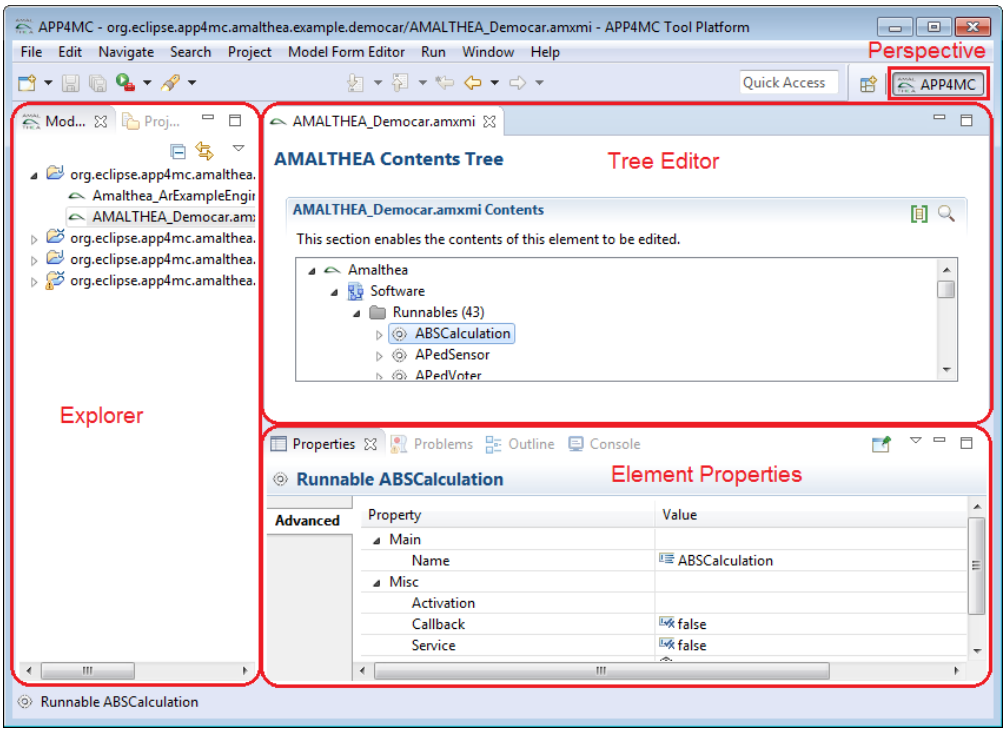


Fig. 1.2: Eclipse APP4MC platform Editor Window

In the figure, Explorer window is used for finding models, performing operations such as partitioning, task generation, mapping, and model migration. The tree editor shows the hierarchical structure of the selected AMALTHEA model,

whereas the Element Properties window is used for editing the properties of AMALTHEA model elements selected in the Tree Editor.

APP4MC is a project that has a lot of synergies with its predecessor AMALTHEA4public project. APP4MC uses AMALTHEA models, which are XML models that describe software components and hardware platforms. Main operation of APP4MC involves modeling the system by creating AMALTHEA models and performing partitioning, mapping, optimization on parallel programs. APP4MC also has the ability to trace simulate parallel programs. Basic ingredients for an AMALTHEA model is illustrated in the following figure:

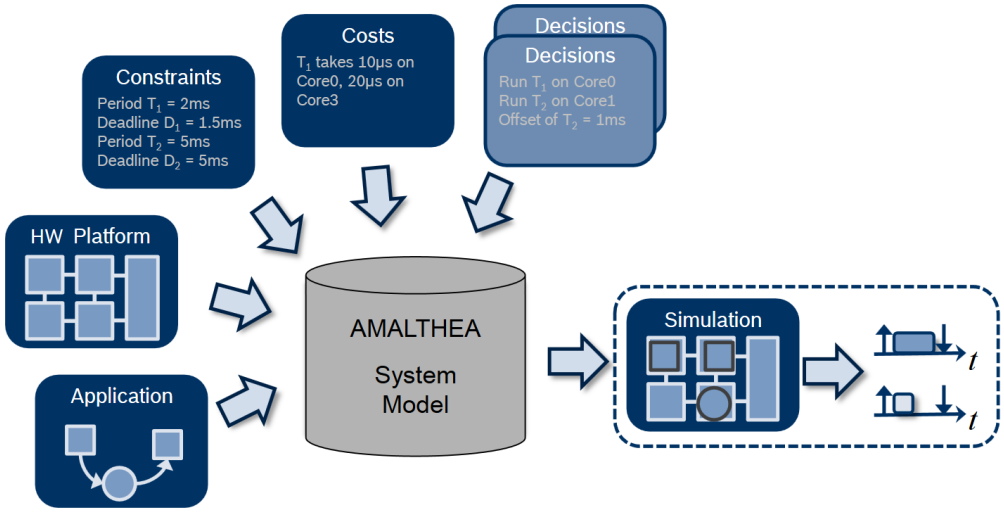


Fig. 1.3: AMALTHEA Model for APP4MC

It is seen that AMALTHEA model can contain software decisions, costs, constraints, as well it can contain hardware platform information. Constraint models are used to define process groups to make sure some processes belong together. Furthermore, a target platform dependency of a process group is also modeled using constraints model.

An illustration of how parallel software can be designed for embedded multi-core platforms are given in the *Illustration of how parallel software are designed using APP4MC platform*. The following remarks can be made regarding the design procedure with APP4MC platform:

- **Modeling**: Design of a parallel software starts with modeling in APP4MC. An AMALTHEA model is constructed that involves three seperate models: (1)- hardware model, (2)-software model, (3)-constraints model. In the hardware model, each distributed ECU is modeled in a hierarchical manner. Hardware model involves information such as number of processor cores, system clock frequency for processors, and memory details. In the software model, runnables are modeled. Runnables are found with the help of binary analysis tools and by using the decomposition technique mentioned in the Section ref{designtech}. Each runnable are modeled by making use of information such as granularity (number of instructions) and label accesses (memory read-write). In the early development stages, model contains a rough model of the software, but the model is constantly improved by using the Tracing functionality of the APP4MC as well it can be improved by using several other tracing software.
- **Partitioning**: Partitioning stage in APP4MC-aided parallel design corresponds to identification of initial tasks. After an initial AMALTHEA model is constructed, the one can perform partitioning in APP4MC by simply selecting the model and pressing the “Perform Partitioning” button. At this stage, APP4MC will analyze the runnables and runnable label accesses in order to suggest how tasks should look like for a balanced parallel distribution. APP4MC uses two partitioning algorithms that are ESSP (Earliest Start Schedule Partitioning) (performed by default) and CPP (Critical Path Partitioning) in order to find the partitions of the system. ESSP and CPP algorithms are based on the Graph Theory which is commonly used in hardware and software co-design. Partitioning algorithms are used for analysis of the granularity and communication costs of individual runnables and create best possible parallelized partitions.

- **Task Generation**: Initial tasks (partitions) are finalized by pressing "Generate Tasks" button. By making use of the dependencies between partitions and by grouping them, APP4MC generates a model that contains the desired amount of tasks with the help of "Task Generation" phase.
- **Mapping (with Simulation and Optimization)**: As known, mapping is the stage of placing the software distributions (tasks, processes, threads) into the processors. By making use of the hardware capabilities and using optimization strategies (such as Integer Linear Programming), APP4MC is able to find a mapping model of the system. The utilization details of the simulations will be seen at the end of the mapping stage.
- **Code Generation**: Since APP4MC provides model-based development, code generation features for C language are available. If desired, APP4MC generates tasks that are written in C language by using the AMALTHEA system model.
- **Tracing**: By making use of binary tracing, AMALTHEA trace model can be observed and can be re-used to update the system model.

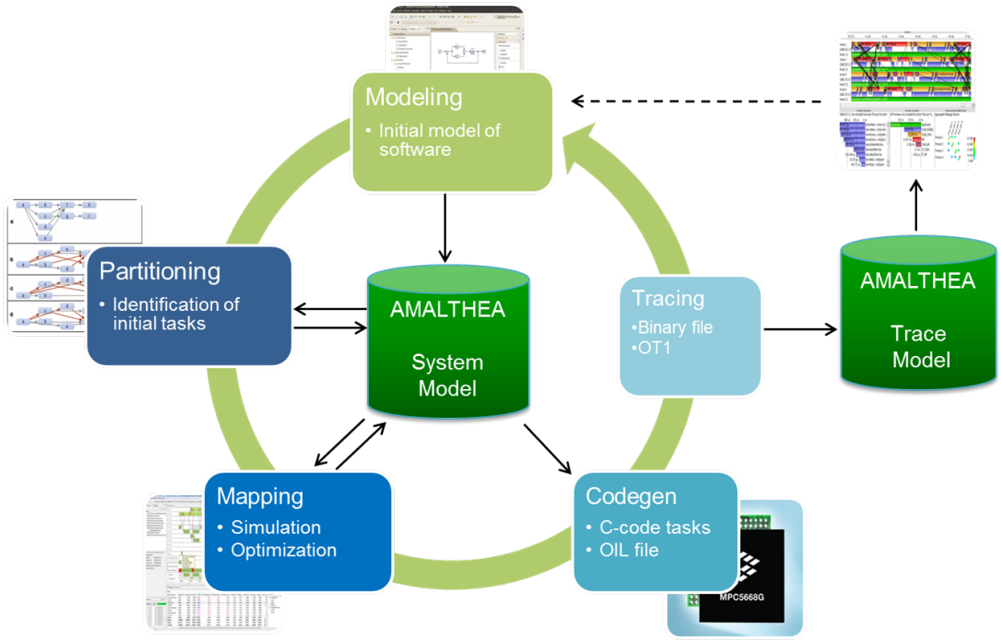


Fig. 1.4: Illustration of how parallel software are designed using APP4MC platform

APP4MC promises beneficial set of tools for embedded parallel software development. However, the demonstration of its features is needed for further improvement. Therefore, in this work APP4MC is used for system parallelization for A4MCAR, a demonstrator RC-Car.

1.4 Objectives of A4MCAR Project

With A4MCAR project, we aim to investigate and evaluate APP4MC's performance with a real-world distributed multi-core system in several aspects. The objectives of this project are as follows:

- Development of a distributed multi-core demonstrator for the APP4MC platform that involves typical automotive application features.
- Investigation of new trends in parallel software development (such as Real-time Linux parallel programming, POSIX threads, RTOS, evaluation methods etc.)
- Researching techniques to retrieve information (number of instructions, communication costs) and system trace from platforms such as xCORE and Linux to achieve precise modelling with APP4MC.

- In order to achieve optimization goals such as reduced energy consumption and reduced resource usage, different affinity constrained software distributions will be evaluated and energy features will be invoked to see if the goals can be achieved.
- Developing a basic online parallelization evaluation software that will retrieve scheduling properties such as slack times, execution times, and deadlines from all the processes and that will tell which deadlines were met and which not. Also, the software distribution assessment is in focus as well as investigating methods to develop schedulable and traceable threads and processes.
- Recording detailed system traces in order to provide offline software evaluation and consequently figuring out means to balance the load on cores.
- Comparing the conventional schedulers of non-constrained affinity distribution (such as a Linux OS scheduler) to the affinity constrained distribution from APP4MC to see if performance can be improved.

With the help of the A4MCAR project, it is intended that the Real-time Linux community will benefit from the published libraries and documentation that involve code snippets and information instructions on how to develop more optimized distributed and parallel software. Furthermore, the Eclipse APP4MC community will benefit from the A4MCAR via advanced tool support for RPI developments, open source example applications, and validations of APP4MC parallelization results in order to create a better tooling available to the public. Those results can be used to assess and compare different parallelization scenarios and consequently identify optimal solutions regarding timing efficiency for the A4MCAR. Thereby, a point of reference can be given as well as an easy starting point for developers approaching parallelism with their developments.

1.5 A4MCAR Methodology and Features

As A4MCAR targets automotive industry and parallelization studies done by APP4MC, it features not only sensing and actuation related features but also applications that would help with task to core distributions and parallelization performance evaluation. One could see the featured applications for the A4MCAR in *Applications developed and/or maintained for A4MCAR*.

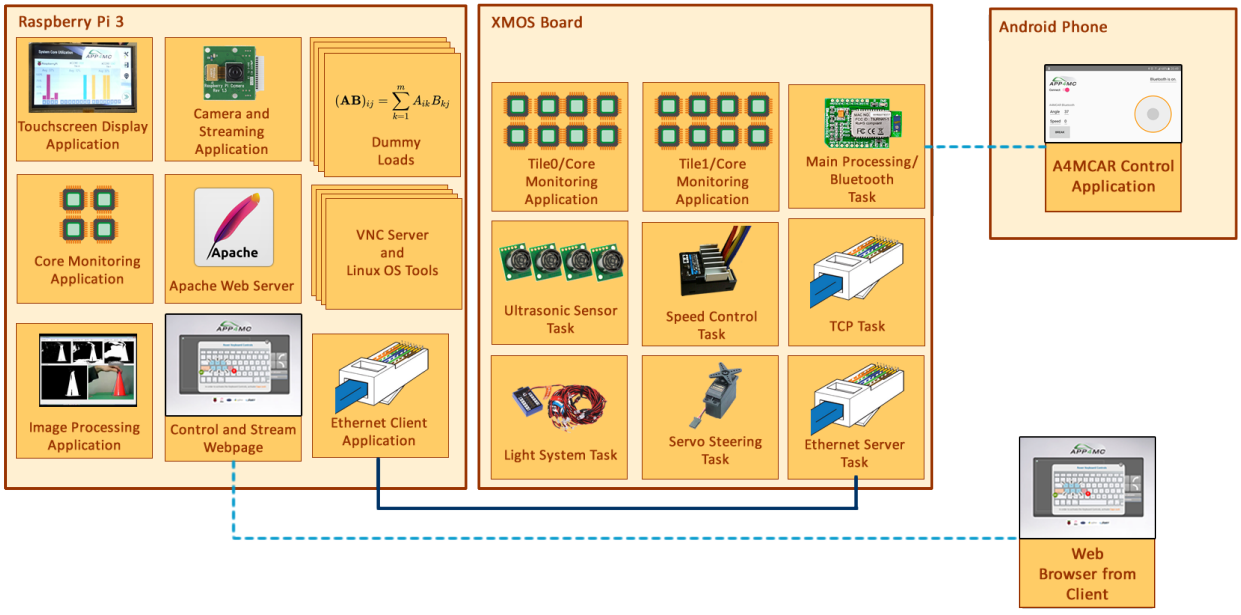


Fig. 1.5: Applications developed and/or maintained for A4MCAR

In the figure, it is seen that the low level module of A4MCAR, built using xCore-200 eXplorerKIT targets mostly actuation and sensing related applications. The full list of tasks developed for the low-level module includes:

- Core monitoring applications for each tile (two exist) that calculates the average core usages.
- Bluetooth task to configure bluetooth module in slave mode and receive data over UART interface.
- Proximity measurement task that obtaines the distance sensor information from four SR-04 sensors over an I2C sensor network.
- Speed control task in order to use PWM to drive speed controlling motor.
- Steering task that controls a servo motor using PWM signaling in order to steer the A4MCAR using external inputs.
- Light system task in order to control a light module for certain conditions.
- Ethernet server task to maintain a TCP server for data reception and transmission from high level module.
- TCP task and several other ethernet configuration related tasks to configure ethernet module (PHY) drivers and establish proper TCP connection.

In order to investigate parallelization outcomes on Real-time Linux and make use of high level features such as web server, image processing and touchscreen interface high-level module is introduced to the system. High-level module is designed so that it can communicate with low-level module over TCP in order to send driving information and retrieve core information from low-level module. It is important to mention that high-level module uses Raspberry Pi 3 in order to achieve high level tasks using a robust Debian-based OS, namely Raspbian. Although the features of the high-level module is illustrated in the following figure, full feature list can be given as follows:

- Core monitoring application that calculates the average core usages on each core.
- Image processing application that helps to find a traffic cone.
- Apache Web Server that is used to host a web page which shows average core usage, show Raspberry Pi 3 camera (Raspicam) stream and helps to drive the A4MCAR via web page controls.
- Ethernet client application that writes handles data transmission and reception to and from server using file operations and data parsing.
- Camera and streaming application that starts the Raspicam and maintains the stream using configuration parameters such as resolution, quality, frame rate, port and etc.
- A webpage which is used for driving the A4MCAR as well as display core usages on each core and Raspicam stream.
- A touchscreen display application which is used for displaying all cores and their utilization, starting and killing all applications on high-level module, allocation of processes on high-level module to cores dynamically, visualization of timing related performance indicators such as average slack time and deadline misses percentage, selecting different distributions, and configuration of the IP addresses on high-level module.
- Dummy load processes that perform random matrix multiplication in order to find performance indicators in full utilization.
- Several Linux processes that run Linux OS kernel and VNC server process that provides PC connection via SSH connection.

1.6 Future Developments

Future developments involve:

- AGL (Automotive Grade Linux) or Apertis Linux distro integration of the applications.

- Developing more applications in the direction of IoT.
- Developing Eclipse Hono interaction library and involving Cloud interactions in parallel with APPSTACLE project.

CHAPTER 2

Google Summer of Code Contribution

A4MCAR Project is a project that is initiated by Fachhochschule Dortmund IDiAL Institute. The institute is highly involved with the AMALTHEA4public Project ,its successor APP4MC Project and APPSTACLE Project which are developed as ITEA research projects and funded by Federal Ministry of Education and Research (BMBF) in Germany. The projects are strongly partnered with Bosch GmbH and Eclipse Foundation GmbH.

In the scope of three of the projects, and especially parallel with the APP4MC Project, which aims to create an open-source platform that is used for engineering embedded multi-core and many-core software systems, the demonstrator projects **A4MCAR Project** and **Rover Project** are developed which are used effectively to test the tools and standards created by the IDiAL institute and evaluate the results.

Google Summer of Code is an annual program that is organized by Google Inc. and that is done for encouraging students to open-source projects in open-source organizations. In the program, Google Inc. delivers stipends to students who successfully complete the program. With the A4MCAR project, Mustafa Özçelikörs participated in Google Summer of Code 2017 with the supervision of Robert Höttger with The Eclipse Foundation as the mentoring organization.

During his master studies and Google Summer of Code 2017 participation, author of the A4MCAR project, Mustafa Özçelikörs was involved in both of the projects **A4MCAR** (mainly) and **Rover** in order to address several topics:

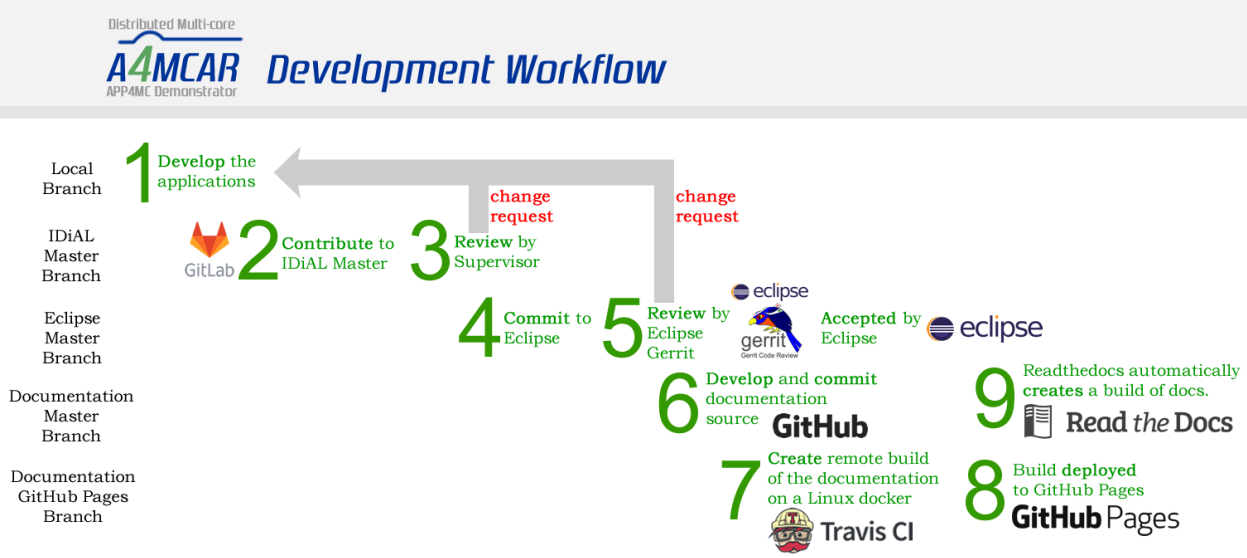
- Developing embedded drivers and libraries for embedded units
- Developing means and tool support to test APP4MC
- Addressing schedulability and traceability of the programs
- Developing necessary Human Machine interfaces for embedded units
- Using APP4MC actively and reporting the bugs to the development team
- Developing interfaces for technologies such as cloud and IoT

In order to learn more about the project or download the source code of **A4MCAR** and **Rover** projects, go to the link *Accessing the Project*.

To see the commits done to the project, please follow the link http://git.eclipse.org/c/app4mc/org.eclipse.app4mc.examples.git/log/.

2.1 Development Workflow

Check out the open-source development workflow that we have adopted from the figure below:



CHAPTER 3

Accessing the Project

3.1 Google Summer of Code Page

To see the Google Summer of Code 2017 page of the A4MCAR project, visit Summer of Code With Google - Projects.

3.2 Source Code

Source code in several environments, AMALTHEA models, and documentation of the **A4MCAR** and **Rover** projects are distributed under Eclipse Public License from the Git repository: https://git.eclipse.org/r/app4mc/org.eclipse.app4mc.examples. To see the instructions on how to clone and install applications, please follow: *Downloading A4MCAR applications from the Git repo*.

3.2.1 Commits

In order to see the commits done to the project, please follow the link http://git.eclipse.org/c/app4mc/org.eclipse.app4mc.examples.git/log/.

3.2.2 Source Tree

To view the **A4MCAR** and **Rover** project source tree online, one should visit http://git.eclipse.org/c/app4mc/org.eclipse.app4mc.examples.git/tree/.

3.3 Documentations

Two documentations are delivered with A4MCAR project:

- This A4MCAR Introduction and Setup manual that is hosted under https://mozcelikors.github.io/a4mcar/ or http://a4mcar.readthedocs.io.

- The comprehensive documentation of A4MCAR which address research focuses can be accessed using This PDF

3.4 External Links

- APP4MC Project
- APP4MC Project Wiki
- Rover Extensive Documentation
- Rover Project Wiki
- Google Summer of Code 2017 Ideas Wiki - Eclipse.org
- APP4MC-based project accepted for Google Summer of Code 2017
- PolarSys Rover User Story: Application Platform Project for MultiCore (APP4MC)
- IoT Playground EclipseCon Europe 2016 APP4MC Entry
- IoT Playground EclipseCon Europe 2017 Wiki Entry

CHAPTER 4

Hardware Setup

A4MCAR is featured with two controllers: (1)- High-level controller controller that is equipped with a Raspberry Pi, with applications ported to Raspbian OS on ARM Cortex A53 processor, and (2)- Low-level controller based on an XMOS xCORE-200 ExplorerKIT development kit that is equipped with a 16-core microcontroller.

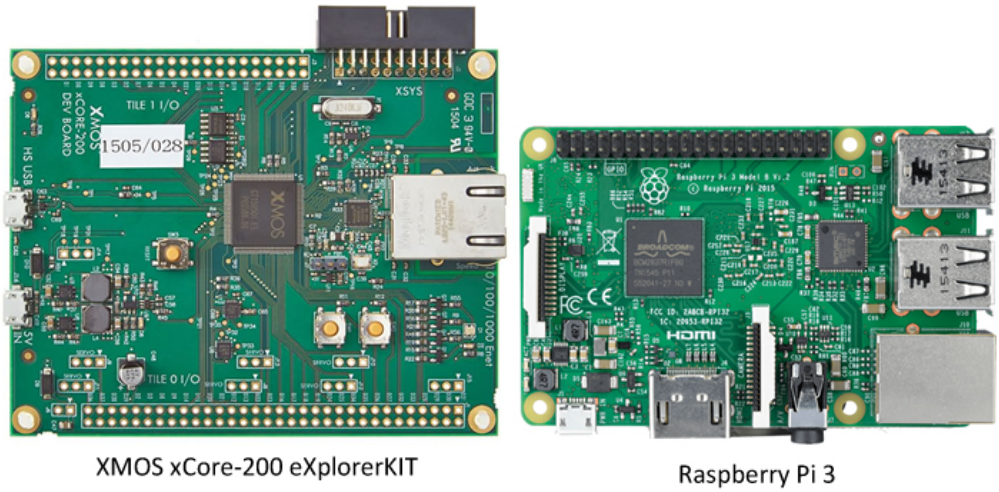


Fig. 4.1: Development boards used in A4MCAR

Development boards in A4MCAR are interfaced with many peripherals for different purposes. A brief look of how devices are interfaced and low-level schematics are given below for reference.

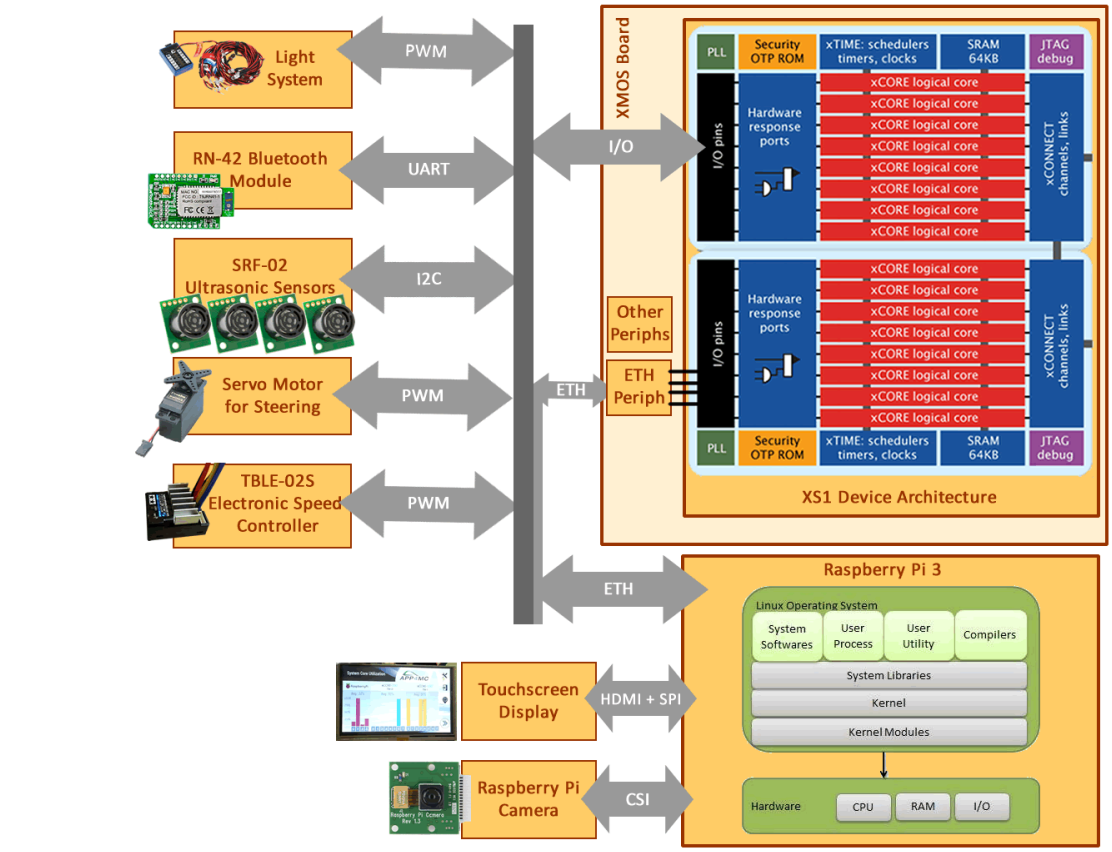


Fig. 4.2: Hardware overview of A4MCAR

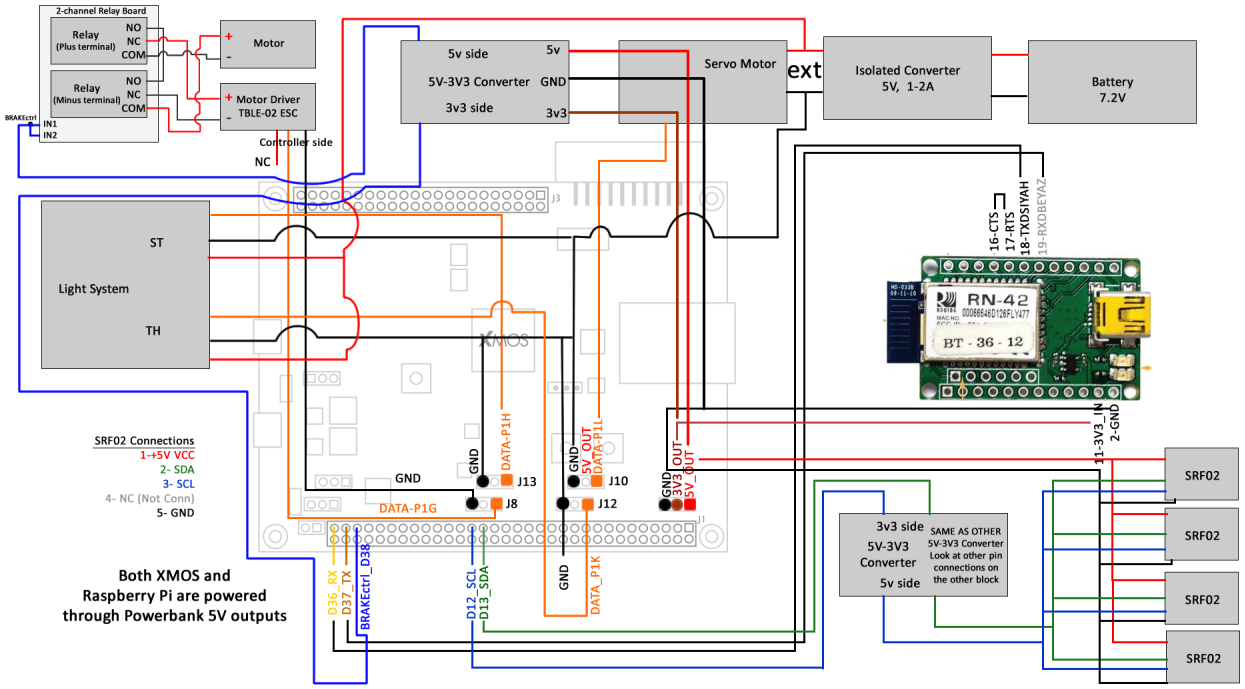


Fig. 4.3: A4MCAR Low-level Schematics (Click to Enlarge)

CHAPTER 5

Software Setup

5.1 Downloading A4MCAR applications from the Git repo

The A4MCAR's software is distributed in an open-source repository. To clone it to your system please use the following command.

Note: Note that in this documentation it is assumed that you download your a4mcar folder to the `/home/pi` directory.

Note: You should have `git` installed. Run `sudo apt-get install git` if you don't have it.

```
git clone https://git.eclipse.org/r/app4mc/org.eclipse.app4mc.examples
```

Then, move a4mcar folder to the `/home/pi` directory by running the following:

```
mv org.eclipse.app4mc.examples/a4mcar/ /home/pi
```

5.2 Provided Filesystem

The repository should have the following main folders:

- **web_interface :** The web interface that is developed for A4MCAR project which is used to control A4MCAR over remote Wi-Fi connection. In order to set up web_interface, run the setup script: `web_interface/setup_web_interface.sh`. In order to run the web_interface correctly, the high-level modules core_reader and ethernet_client should be ready and working. To run the `web_interface` one should connect to the access point of Raspberry Pi from a client computer web browser and visit `http://<IP_Address>/jqueryControl2.php` or `http://<IP_Address>/jqueryControl.php`.

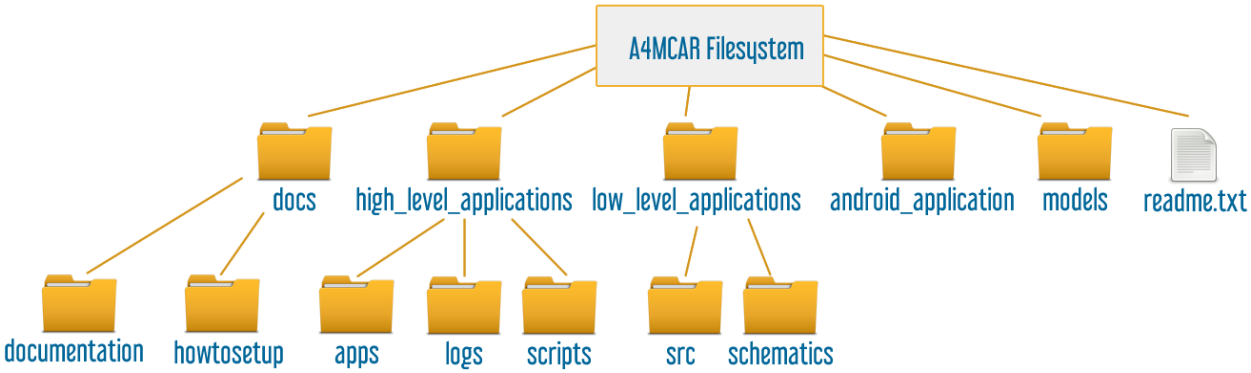


Fig. 5.1: A4MCAR Filesystem

- **high_level_applications :** This module consists of several high-level applications that are developed for A4MCAR's high-level module (Raspberry Pi). These applications involve: `touchscreen_display`, `core_recorder`, `dummy_loads`, `ethernet_client`, and `image_processing`. n order to run the applications, respective Python files could be run or C/C++ binaries could be executed. Also the scripts that are located under scripts folder could be used to initialize some of the applications. In order to set up high_level_applications module dependencies, one should run the setup script and follow the instructions: `high_level_applications/setup_high_level_applications.sh`.
- **models:** A4MCAR's hardware and software model with Eclipse APP4MC is located in this directory.
- **android_application:** This directory consists of the source files that belong to the A4MCAR's bluetooth based driving application. The source and design files could be used in an Android IDE in order to make tweaks to the application and to generate new .apk files.
- **low_level_applications:** `low_level_applications` module involves the source code for the low-level module that are run using a multi-core microcontroller XMOS xCORE-200 eXplorerKIT. The low level applications are responsible for tasks such as sensor driving, actuation, communication, and core monitoring of the A4MCAR. The low_level_applications module could be imported into xTIMEcomposer to make tweaks to the tasks.
- **docs:** Involves documentation and Sphinx-generated how-to-setup manual.

5.3 High-Level Applications Setup

5.3.1 Raspberry Pi Basic Setup

In order to install the high level applications to the Raspberry Pi 3, several preparations should be made. In order to boot a Raspberry Pi 3 with a Raspbian Jessie distribution, one should use the Raspbian Jessie Download Link.

After the Raspbian image is downloaded, a disk imager software could be used to burn the image to an SD card. To do that you can follow the instructions given in this link.

5.3.2 Access Point Setup

For the convenient development, Raspberry Pi 3 should be set up as an access point. The access point helps regarding connecting to the Raspberry Pi's shell via SSH for entering Linux commands.

To set up the access point, the guidelines given in PolarSys Rover Raspberry Pi Configuration Doc Step 1.4 can be followed.

5.3.3 Installation of A4MCAR's high-level application and its dependencies

Automatic Setup and Installation

Provided applications involve a setup script that can be used to install required dependencies manually. To execute the script, run:

```
sudo bash /home/pi/a4mcar/high_level_applications/setup_high_level_applications.sh
```

Manual Setup and Installation

Installation of Some Useful Applications

Note: This part is created for instructive purposes. The setup shown here is carried out automatically with *Installation of A4MCAR's high-level application and its dependencies*.

A4MCAR requires some third party software during both development and application execution. These applications could be listed as follows:

- `git` is a tool that is used for cloning repositories to your computer.
- `python-pip` is a package that is used for installing python modules.
- `cmake` is a very easy-to-use tool that generates Makefiles very conveniently.
- `build-essential` package helps regarding compiling applications using GNU/Linux systems.
- `python-prctl` package is used in A4MCAR for registering thread names with Python.
- Also, python development modules and resources should be installed: `python-dev`, `python3-dev`, `python-pkg-resources`, `python3-pkg-resources`.

You should install them using:

```
sudo apt-get update
sudo apt-get install python-dev python3-dev python-pkg-resources python3-pkg-resources
sudo apt-get install git build-essential python-prctl python-pip cmake
```

Cloning Necessary Modules

Note: This part is created for instructive purposes. The setup shown here is carried out automatically with *Installation of A4MCAR's high-level application and its dependencies*.

A repository is present for providing necessary dependencies for the A4MCAR project's high-level applications. This repository should be cloned and installations should be carried out using this documentation.

To clone the repository,

```
cd ~/Downloads
sudo git clone https://gitlab.pimes.fh-dortmund.de/RPublic/a4mcar_required_modules.git
```

Using virtkey

Note: This part is created for instructive purposes. The setup shown here is carried out automatically with *Installation of A4MCAR's high-level application and its dependencies*.

After cloning necessary modules, you should execute the following command to move the virtkey module to where it is used.

```
sudo cp -r ~/Downloads/a4mcar_required_modules/high_level_applications/virtkeyboard/* /home/$USER/a4mcar/high_level_applications/apps/touchscreen_display/
```

Camera and Streaming Setup

Note: This part is created for instructive purposes. The setup shown here is carried out automatically with *Installation of A4MCAR's high-level application and its dependencies*.

In order to set-up the Raspberry Pi camera and the streaming application `mjpg-streamer`, following steps should be followed:

First, you need to enable the Raspberry Pi camera from the OS. To do that, type

```
sudo raspi-config
```

A configuration screen should appear. Here, select `Advanced` >> `Enable Camera` and then save and exit from the screen.

To stream using Raspberry Pi camera, an experimental third-party streamer module `mjpg-streamer-experimental` is used.

First, start by installing the dependencies and carrying out other necessary steps:

```
sudo apt-get install libjpeg8-dev imagemagick libv4l-dev
sudo ln -s /usr/include/linux/videodev2.h /usr/include/videodev.helps
```

Now, we should install the `mjpg-streamer-experimental`:

Installing psutil from Python for Core Monitoring

Note: This part is created for instructive purposes. The setup shown here is carried out automatically with *Installation of A4MCAR's high-level application and its dependencies*.

Warning: Here we use pip version 1.5.6 to install.

You can use the following commands to install the `psutil` package from our repository:

```
sudo easy_install pip==1.5.6
sudo pip install ~/Downloads/a4mcar_required_modules/high_level_applications/psutil/
```

Apache Web Server Installation

Note: This part is created for instructive purposes. The setup shown here is carried out automatically with *Installation of A4MCAR's high-level application and its dependencies*.

In order to install Apache Web Server on your system to be able to use front-end web interface functionality of A4MCAR, you can follow the guidelines provided in this section.

First, you should upgrade your system's repository list and your upgrade your system via:

```
sudo apt-get update
sudo apt-get upgrade
```

Then type the following commands to install apache2 server, php mods for the server, and easy Makefile tool cmake:

```
sudo apt-get install apache2 -y
sudo apt-get install php5 libapache2-mod-php5 -y
sudo apt-get install cmake
```

After these steps, the website should appear at the `/var/www/html` directory. Now, we are off to grant some permissions to our Raspberry Pi's user groups for this directory. Security is not a main concern in our application, so please use them at your own risk:

```
sudo chgrp -R www-data /var/www/html/
sudo find /var/www/html/ -type d -exec chmod g+rx {} +
sudo find /var/www/html/ -type f -exec chmod g+r {} +
```

The following gives permissions to your specific user. If you use username other than `pi`, make sure to adjust it in the following commands:

```
sudo chown -R pi /var/www/html/
sudo find /var/www/html/ -type d -exec chmod u+rwx {} +
sudo find /var/www/html/ -type f -exec chmod u+rw {} +
```

Subsequently, you should enable access to Linux file-system via webpages by using:

```
sudo visudo
```

and then, use your favourite editor to add the following to the end, save and exit.

```
www-data ALL=(ALL) NOPASSWD: ALL
```

Now you can place our web interface to the `/var/www/html/` directory and start visualizing the interface from `http://<Your_IP_Address>/jQueryControl.php` or `http://<Your_IP_Address>/jQueryControl2.php`

Installing OpenCV 3.0

OpenCV should be installed for Image Processing application to work. In A4MCAR, C++ development library for OpenCV is used. The application is built and linked using GNU Make / CMake.

For OpenCV 3.0 C++ download and installation, please refer to OpenCV 3.0 Install.

Installing Raspicam

To install raspicam-0.1.3, please refer to raspicam-0.1.3 Install.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Bibliography

[App4mcProposalEclipse] https://projects.eclipse.org/proposals/app4mc