Audi Autonomous Driving Cup

Manual 2016

Current version: 1.4

Created: June 1, 2015

Last changed: October 28, 2015

Change Documentation

Title: AADC Manual 2016

Version: 1.4

State: public

Theme	Author	Release	Date	Version
Released	Sebastian Zech	Dr. rer.nat. Patrick Heinemann	07.08.2015	V1.0
Internal Release	Raphael Spies	Sebastian Zech	30.09.2015	V.1.1
Released	Raphael Spies	Sebastian Zech	05.10.2015	V.1.2
Chapter "Use Remote Radio Controller" added Details added in "Speed Controller Setup" WLAN Credentials added	Raphael Spies	Sebastian Zech	07.10.2015	V.1.3
Chapter 3.1.1 added Chapter 8.1. added	Raphael Spies	Erhard Forchhammer	28.10.2015	V.1.4

Contents

List of Figures

Abbreviations

AADC	Audi Autonomous Driving Cup
LiPo	Lithium-Polymer-Accumulator
ADTF	Automotive Data and Time-Triggered Framework
IMU	Inertial Measurement Unit

1 Introduction

It is not at all uncommon to see words like "revolutionary" or "groundbreaking" being used in the automotive industry. But these words do rarely fit so aptly as when they are used to describe piloted driving. Audi has set out to fundamentally change the way we operate our cars. And to improve it. The efforts focus on the intelligence of the technology and the decisions of the driver.

The goal of the Audi Autonomous Driving Cup is to develop algorithms for piloted driving – at a scale of 1:8. For this purpose Audi has exclusively developed model cars and equipped them with basic software packages:

- Braking and Acceleration
- Steering
- Traffic Sign Recognition
- Lane Following

The task of the participants is to create their own software based on these packages to manage a given course – as fast and as elegant as possible.

1.1 Drive

The vehicle is driven by a motor that delivers high power and torque at a scale of 1:10 and 21.5 turns. This is controlled by a brushless crawler speed controller. The combination of both allows the car to be driven very exactly even at a pretty low velocity, forwards as well as backwards.

1.2 Steering

The steering of the cars is provided by a high-quality servo motor that is used in professional model car competitions.
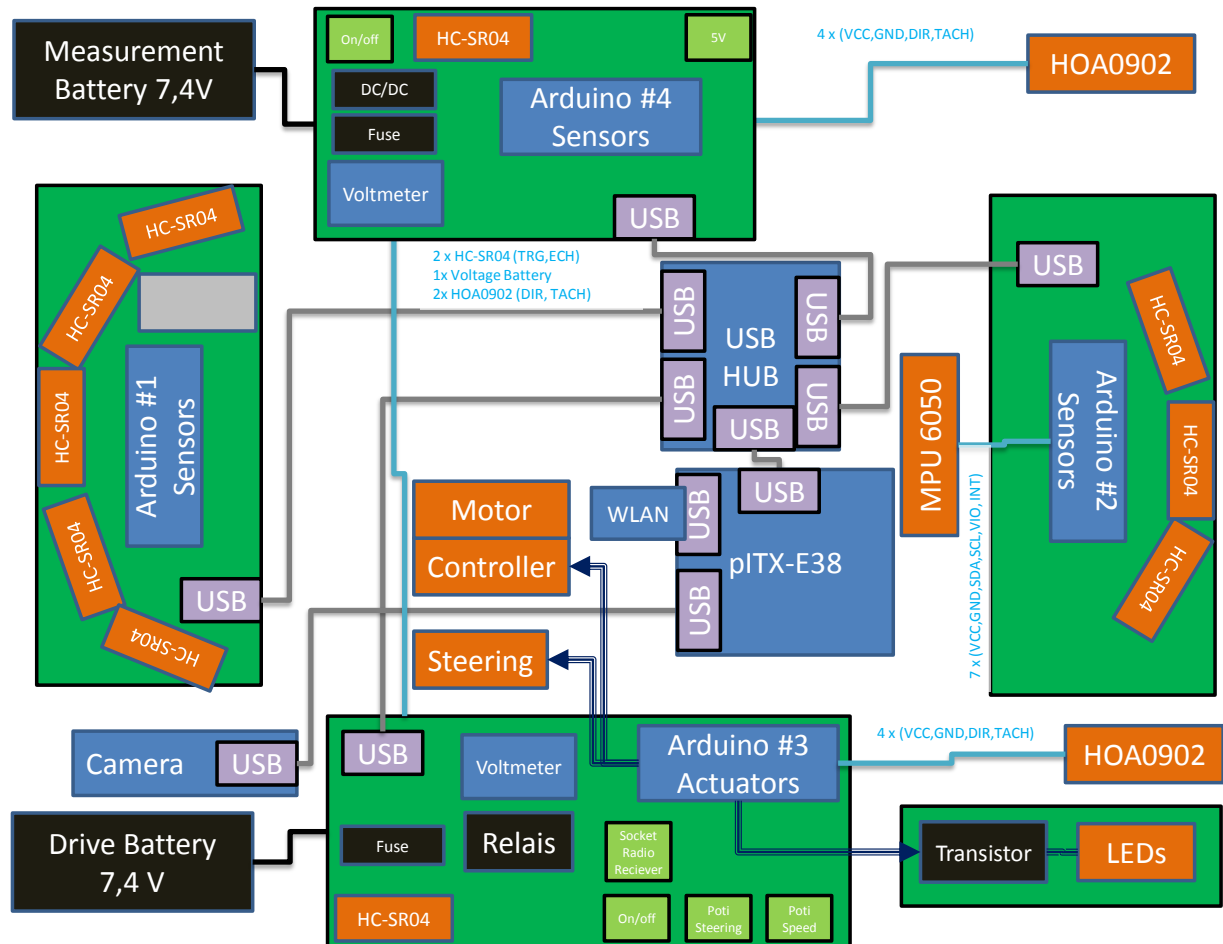
2 Hardware

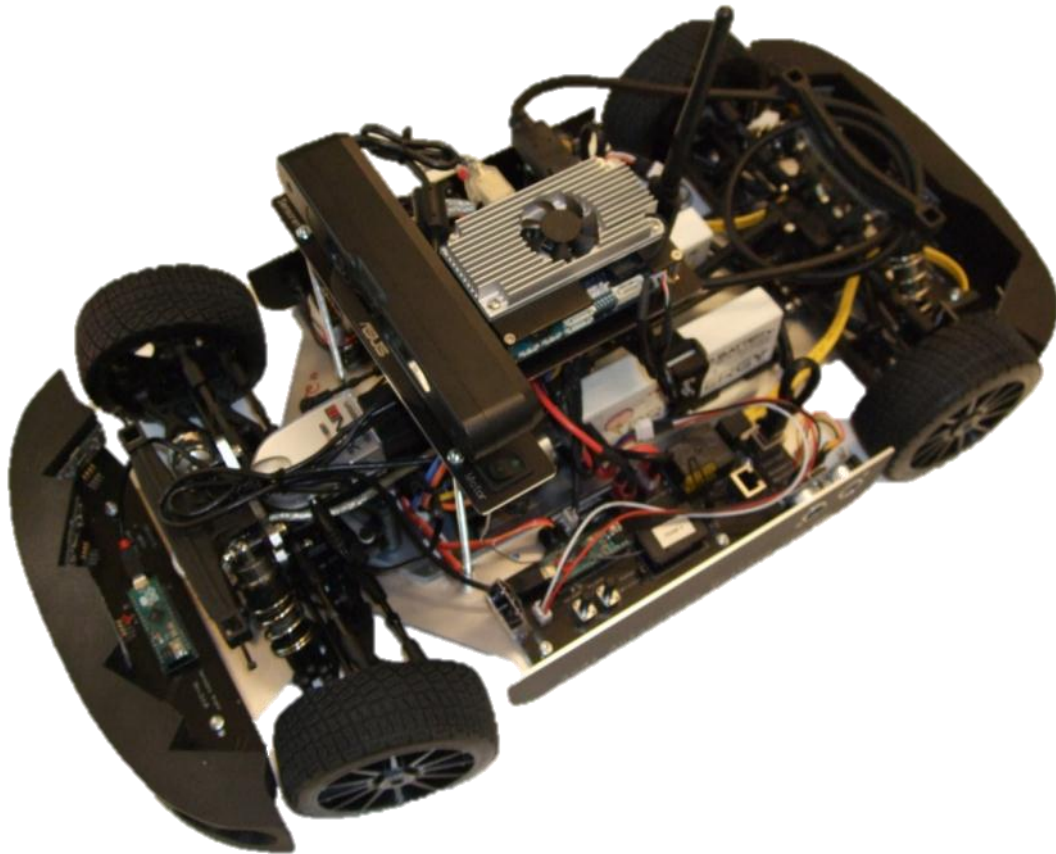## 2.1 Hardware Layout

Figure 1 Hardware Layout

Figure 2 The car

2.2 Kontron pITX-E38 QC 1.91Ghz (E3845)
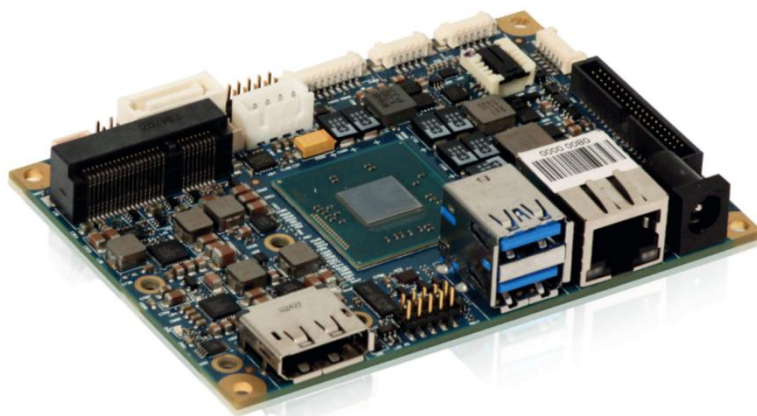
Figure 3 Board Detail Kontron pITX-E38[1]

- Intel® Atom™ E3845 SoC (4 cores), TDP: 5-10W

---

[1] http://www.kontron.de/products/boards-and-standard-form-factors/motherboards/pico-itx/pitx-e38.html

- SO-DIMM Socket DDR3L-1333 Memory (up to 8GB)
- Intel® Gen7 Graphics, OpenGL 3.0, OpenCL 1.2, DX11, H.264, MPEG2, MCV, VC-1, VP8
- LVDS 24Bit dual channel and Display Port 1.1a
- 10/100/1000MBit RJ45 Ethernet LAN, SATA, mSATA or mPCIe in mPCIe connector
- USB 2.0 & USB 3.0 ports, Serial ports 16550 UART, GPIOs
- Bootable high capacity micro SD Card slot
- Lockable DC power connector (5V)
- Extended temperature operating from -25°C to +75°C (E1)

"The pITX-E38 form factor is especially suitable for compact PC applications. Due to its small size (100 x 72mm) it can be easily integrated into any system with severe space constraints and/or thermal restrictions. The compact size of pITX-E38 gives the board a unique advantage over traditional PC based motherboards like the mITX or Flex. Other form factor would simply not fit those requirements. That is why pITX-E38 becomes the preferred choice in any compact applications.

The industry target for smaller size and lower power plays an ever growing role these days. The pITX form factor offers a substantial range of connectivity options and increasing while maintaining its compact size and low power consumption. In addition to that pITX provides leading edge CPU and GPU performance with the next generation Intel® Atom™ System on Chip (SoC) and a large variety of high-speed interfaces.

The pITX-E38 facilitates two display outputs delivering high resolution video content to LVDS driven panels or Display Port monitor as well as 3D graphics performance in a low power envelope. The low power Intel Atom™ Core does provides not only several special characteristics like high resolution video playback option, 3D graphics and CPU core performance, but also offers a wide range of standard interfaces. That is why the pITX-E38 board is well matched with those applications that are of compact size and for which standard interfaces like Gigabit Ethernet, SATA, mSATA & mPCIe are desired in a low power environment. By supporting OS bootable high capacity, high speed microSD media cards shrink the footprint even more since applications do not necessarily require external use of SSD or HDD devices connected to the SATA port. Furthermore, the pITX-E38 board supports high-speed DDR3L memory modules which can be operated over a wide temperature range, with a passively Intel® Atom™ core cooled environment and powered by a single supply of 5Volts. The pITX-E38 maintains a wide variety of embedded and non-embedded operating systems and 7 years supply longevity."[2]

Inside the vehicle the Kontron board is equipped with SO-DIMM 8 GB DDR3L-1333 and a Kingston SSDNow mS200 SSD 60GB disk drive.

---

[2] http://www.kontron.de/products/boards-and-standard-form-factors/motherboards/pico-itx/pitx-e38.html
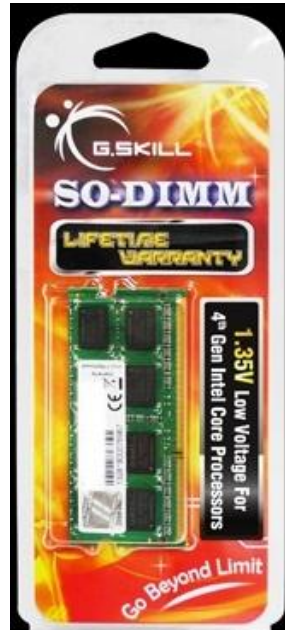
Figure 4 DDR Memory[3]

Figure 5 Kingston SSD[4]

2.3 Arduino Micro

"The Arduino Micro is a microcontroller board based on the ATmega32u4 (datasheet), developed in conjunction with Adafruit. It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and a reset button. It contains everything that is needed to support the microcontroller - getting started simply by connecting it to a computer via a micro USB cable. It has a form factor that enables it to be easily placed on a breadboard.

The Micro is similar to the Arduino Leonardo in that the ATmega32u4 has built-in USB communication, eliminating the need for a secondary processor. This allows the Micro to appear

---

[3] http://www.gskill.com/en/product/f3-1333c9s-8gsl
[4] http://www.kingston.com/datasheets/sms200s3_de.pdf

to a connected computer as a mouse and keyboard, in addition to a virtual (CDC) serial / COM port." [5]
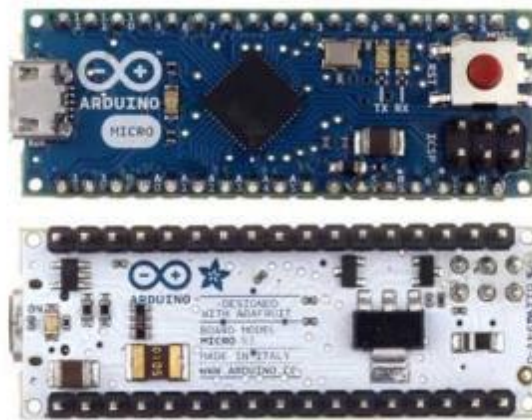
Figure 6 Arduino Micro[6]

- Microcontroller ATmega32u4
- Operating Voltage 5V
- Input Voltage (recommended) 7-12V (limits 6 - 20 V)
- Digital I/O Pins 20
- PWM Channels 7
- Analog Input Channels 12
- DC Current per I/O Pin 40 mA
- DC Current for 3.3V Pin 50 mA
- Flash Memory 32 KB (ATmega32u4)
  (4 KB bootloader)
- SRAM 2.5 KB (ATmega32u4)
- EEPROM 1 KB (ATmega32u4)
- Clock Speed 16 MHz
- Dimensions (LxW) 48 mm x 18 mm
- Weight 13 g

---

[5] http://arduino.cc/en/uploads/Main/ArduinoMicroFront_450px.jpg
[6] http://arduino.cc/en/uploads/Main/ArduinoMicroFront_450px.jpg
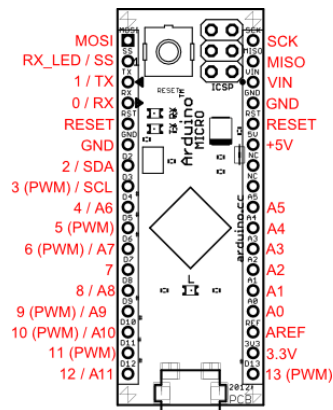
Figure 7 Arduino Micro Pinout[7]

2.4 Sensors

2.4.1   Sensor Ranges
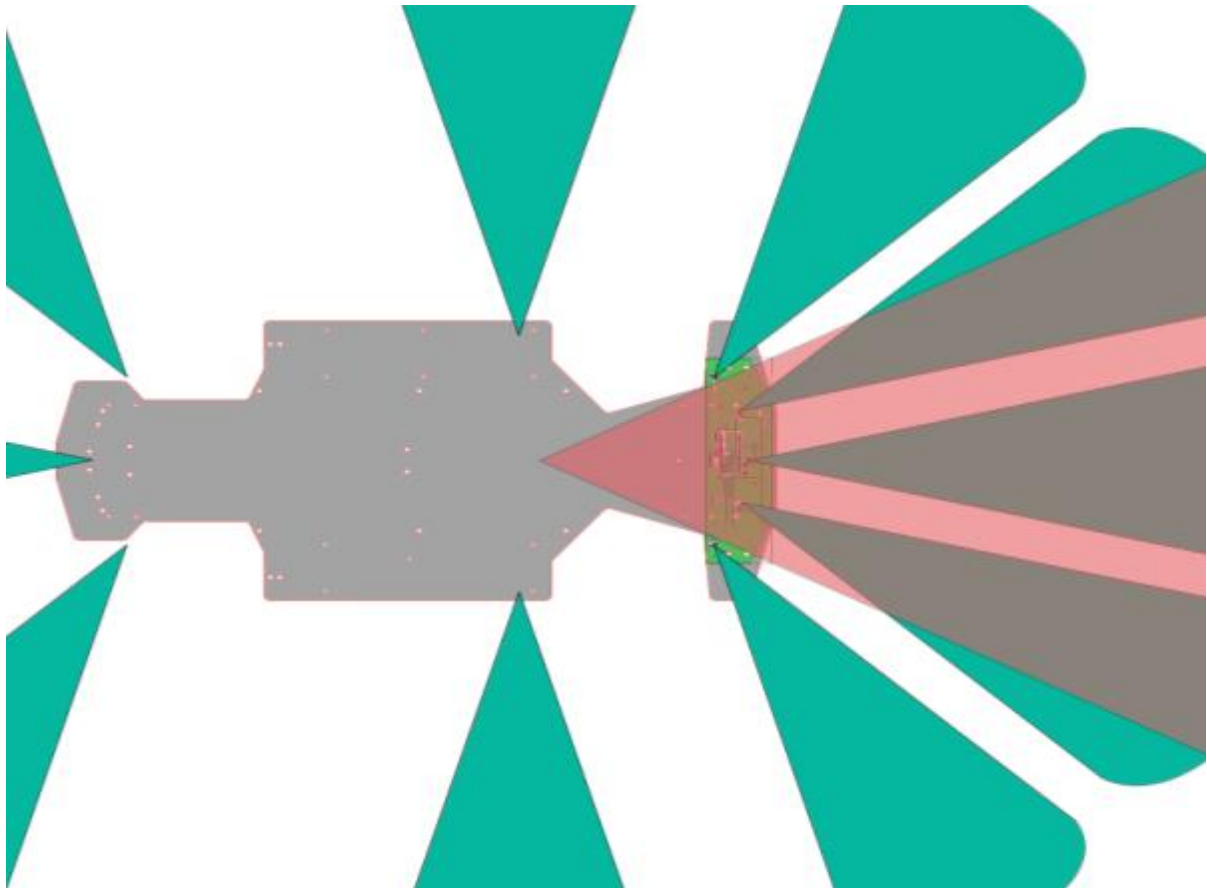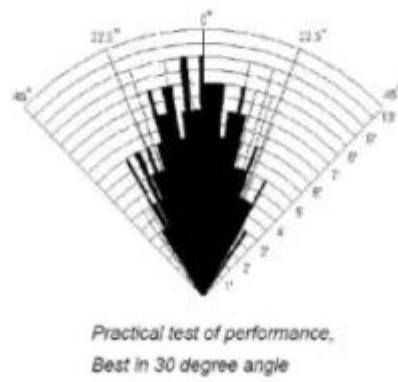
Figure 8 Sensor Layout and Ranges

---

[7] http://arduino.cc/en/uploads/Main/ArduinoMicro_Pinout3.png

2.4.2   HC-SR04 Ultrasonic Sensor

Figure 1 HC-SR04 Ultrasonic
Sensor[8]

HC-SR04 Ultrasonic Sensor

- Operating Voltage: 4.5V to 5.5V
- Quiescent Current: 1.5mA to 2.5mA
- Working Current: 10mA to 20mA
- Ultrasonic Frequency: 40Hz
- Range: 2cm to 400cm
- Resolution: 0.3cm
- Measuring Angle: 30°
- Effectual Angle: <15°
- Dimension (mm):   45 x 20 x 15

Practical test of performance.
Best in 30 degree angle

Figure 9 Measuring Angle HC-SR04 (1)

2.4.3   MPU-6050 Triple Axis Accelerometer and Gyro

<table>
<tr>
<td>

Figure 2 MPU-6050 Triple Axis Accelerometer and Gyro[9]

</td>
<td>

MPU-6050 Triple Axis Accelerometer and Gyro

- Operating Voltage: 2.3V to 3.4V
- Working Current: 3.9mA (GYR + ACC + DMP)
- Working Current: 10mA to 20mA
- Max Data Rate ACC: 1000Hz
- Max Data Rate GYR: 8000Hz
- Dimension (mm):   25.5 x 15.2 x 2.48mm
- I2C Digital-output of 6 or 9-axis MotionFusion data in rotation matrix, quaternion, Euler Angle, or raw data format
- Tri-Axis angular rate sensor (gyro) with a sensitivity up to 131 LSBs/dps and a full-scale range of ±250, ±500, ±1000, and ±2000dps
- Tri-Axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g

</td>
</tr>
<tr>
<td colspan="2">

Figure 10 Block Diagram MPU-60x0 (2)

</td>
</tr>
</table>
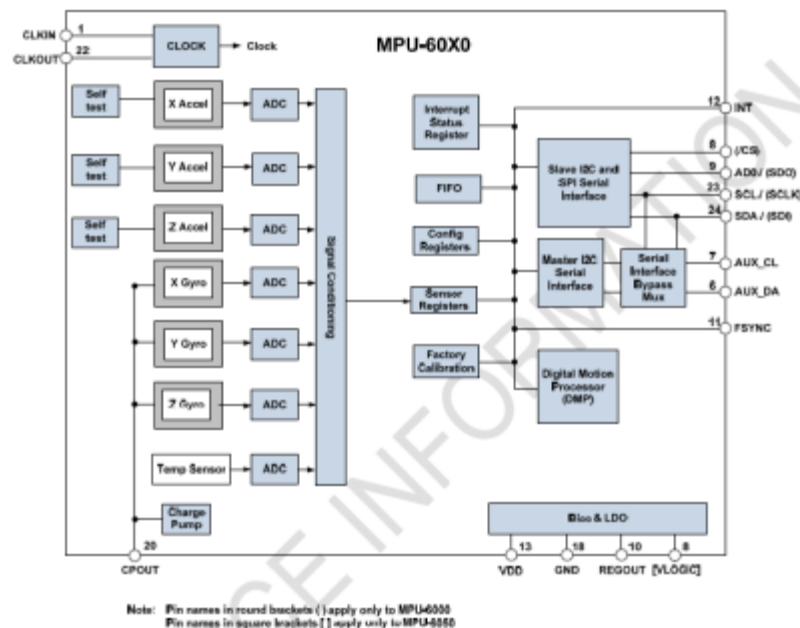
[9] https://cdn.sparkfun.com//assets/parts/6/3/5/5/11028-01.jpg

2.4.4   HOA0902-11 – Encoder

Figure 3 HOA0902-11
Transmissive Encoder Sensor
(3)

Figure 11 Encoder Wheel

Honeywell HOA0902-11 – Transmissive Encoder Sensor

- Operating voltage: 4.5V to 5.5V (Detector)
- Operating voltage: 1.6V (Emitter)
- Revers Leakage Current: 10uA (Emitter)
- Supply Current: 12mA
- Slot width: 3.2 mm
- Resolution: up to 0.457mm
- Tach Pulse Width: 3…20 us
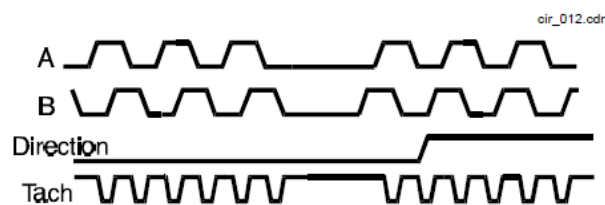
Figure 12 Output Timing Diagram HOA0902-11 (3)
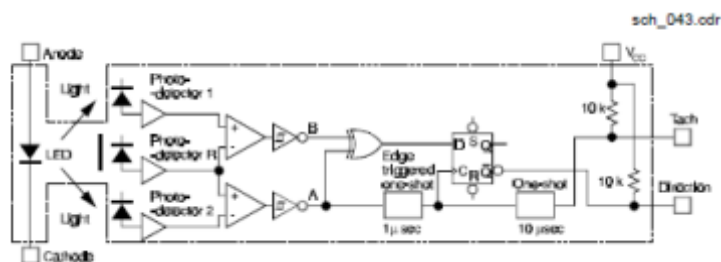
Figure 13 Functional Block Diagram HOA0902 (3)

2.5 Actuators

2.5.1 Absima "ACS1615SG" Steering Servo

	Absima "ACS1615SG" Combat Series
  Figure 4 Absima "ACS1615SG" Combat Series[10]	• Operating voltage: 4 .8V to 6.0V • Pulling Force: 13kg to 15kg • Regulating time: 0,11s to 0,09s • Angle: 60° • Weight: 77 g

2.5.2 Robitronic Speedstar Brushless Speed Controller

	Robitronic Speedstar Brushless Regler 8,5T
  Figure 5 Robitronic Speedstar Brushless Crawler[11]	• LiPo Cells: 2-3s • B.E.C. Voltage: 5V • B.E.C. Current: 3,0A • Ampacity: 260A / phase (brushless) • Motor Turn Limit: 8,5 Turn (brushless) • Internal Resistance : 0,005 Ohm / phase (brushless) • Dimensions (mm): 40 x 41 x 28.7 • Weight: 43g

[10] http://shop.absima.com/absima_en/prodpic/Servo-ACS1615SG-Combat-Series-2030011_b_0.JPG

[11] ttp://www.robitronic.com/files/produktbilder/Robitronic/Fahrtenregler/R01202_Speedstar_BL8-5T/r01202_800.jpg

2.5.3   Robitronic Platinium Motor

	Robitronic Platinium Brushless Motor 1/10
Figure 6 Robitronic Platinium Brushless Motor[12]	• Technology: Sensored, Brushless • Scale: 1/10 • LiPo Cells: max. 2S / 7 NiMH • Voltage: 7,4V • Length: 53mm • Diameter: 36mm • Weight: 163g • R/min per Volt: 2.100 • Nominal current: 17A • Windings: 21,5T

2.5.4   Absima CR2s V.2 Radio System

\n\nFigure 14 Absima CR2S V.2 Radio Transmitter\n\n\n\nFigure 15 Absima CR2S V.2 Radio Receiver	Absima CR2S V.2 Radio Transmitter\n\n- Channels: 2\n- RF power: less than 20 dbm\n- Modulation: GFSK\n- Code type: digital\n- Sensitivity: 1024\n- Low voltage warning: yes (less than 4,5 V)\n- Power: 6 V (1.5V AA *4)\n- Weight: 328 g\n- ANT length: 26 mm\n- Size: 220 x 150 x 100 mm\n\nAbsima CR2S V.2 Radio Receiver\n\n- Channels: 3\n- Frequency band: 2.4 GHz\n- Modulation: GFSK\n- Sensitivity: 1024\n- RF receiver sensitivity: -100 dbm\n- Power: 4.5–7.2 V\n- Weight: 5 g\n- ANT length: 26 mm\n- Size: 37.6 x 22.3 x 13 mm

2.6 Power supply

2.6.1   Yuki Model 7.4V Battery

\n\nFigure 7 Yuki Model 7.4V Accumulators	Yuki Model 7.4V Accumulators\n\n- Configuration: 2s1p\n- Nominal voltage: 7.4 V\n- Rated capacitance: 5.200 mAh\n- Max. Charge rate: 2C (10.4 A,)\n- Cont. Discharge rate: 30C (156 A)\n- Dimensions: 138.5 x 46.5 x 23.5 mm\n- Weight: 297 g\n- Connection cable: silicon\n- Connection: golden contact 4,0 mm\n- Balancer cable: PVC, compatible with JST XH

2.6.2   Absima CP-1P Charger

  Figure 8 Absima CB-1P Charger[13]	• Operating Voltage: DC 11.0 ~18.0 V • Input Voltage: 110 ~ 240 V • Circuit Power: max. 50 W for charging, max. 3 W for discharging • Charge Current Range: 0.1 ~ 5.0 A • Discharge Current Range: 0.1 ~ 1.0 A • Current drain for balancing LiPo: 200 mAh/cell • Lithium battery cell count: 1–4 cells • Weight: 430g • Dimensions: 157×140.6×79.3mm

2.6.2.1   Charging Batteries

Figure 16 Charging Batteries

First connect the battery to the charger as shown in Figure 8. Connect the charger to the power socket and conjunct the balancer cable and the power cable from battery pack to the charge.

For charging it is recommended to use the charging in balance mode. Using this mode, each cell of the pack will be monitored by the internal processor thereby controlling the charging current in order to normalize the voltage.

[13] http://www.absima.com/uploads/media/CB-1P_Header_688x398_wei%C3%9F.jpg
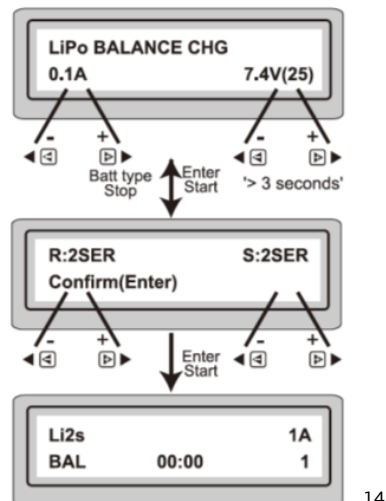
Figure 17 Flow Chart Battery Charging

1. Connect cables
2. Select Battery Select Lipo" with Enter → LiPo Battery choosen
3. Select „LiPo BALANCE CHG" with arrow buttons
4. Confirm with Enter (Short Press)
5. Set charge current with arrows, for this battery pack you can use 5.0.A
6. Confirm charge current with short press Enter
7. Select battery type 7,4V(2S) with arrows
8. Confirm selection with short press on Enter
9. Pack battery in LiPo Guard (gray bag)
10. Confirm settings with long press on Enter
11. Start charging with short press on Enter
12. After finishing charger signalizes completion

---

[14] http://www.absima.com/absima_uploads/Ladeger%C3%A4t/CB-1P%20Manual%20English.pdf

3 Software

3.1 Linux

- Kernel Version: 3.19.0-28-generic
- Distro: Ubuntu 14.04.3
- User name: aadc
- Password: aadc2016

3.1.1   Flashing a new image

1. Backup our ADTF licence file first! Please see `/home/aadc/ADTF License/*.lic`

2. Download and create Clonezilla bootable USB Drive
   http://clonezilla.org/liveusb.php

3. Prepare USB Stick with Linux Image from www.audi-autonomous-driving-cup.de.
   Download and extract archive with tar:
   ```
   $ tar -x AADC_Image_v*.tar
   ```

4. And copy all containing files to a folder on a second USB Drive

5. Making PC booting from USB Stick if not done automatically
   a. Plug in Clonezilla USB Stick
   b. Press DEL (Entf) while turning PC on to enter the BIOS menu.
   c. Select Boot (Press right arrow three times)
   d. Search for Entry "USB HDD: Generic Flash Disk" and select
   e. Press "+" until this entry is at position 1.
   f. Select Exit (Press right arrow once)
   g. Select Exit Saving Changes and Yes
6. Start again and the clonezilla menu will be started

7. Select one of the first three entries ("Clonezilla live (Default settings, VGA 1024x768)"
   for example
8. Clonezilla starts and blue/grey menu should appear
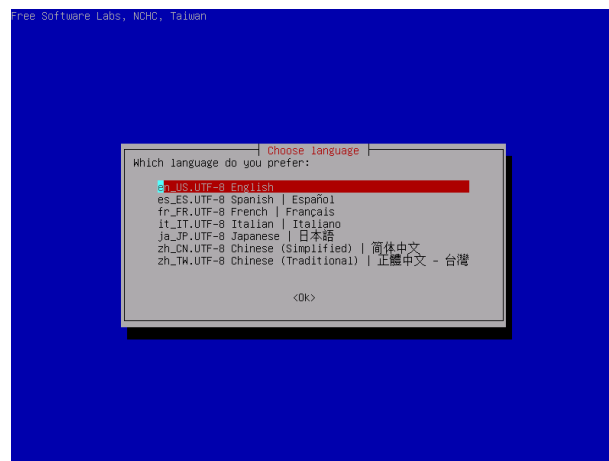
9. Select Language
10. Modify keymap if necessary
11. Select Start_Clonezilla
12. Select device_image
13. Select local_dev
14. Plug second USB Stick with image in the board
15. Select the second USB Stick with the image in the menu (in most times the last entry if it was plugged in at the correct time)
16. Select the folder with the image
17. Some file system usage infos are shown, press Enter to continue
18. Select Beginner
19. Select restoredisk
20. Next screen should show the image file, press Enter to continue
21. Choose the target (only "sda_60.0GB_KINGSTON_SM200S_KINGSTON_SMS200S360G_..." should be shown (the local mSATA drive)
22. Press "Enter" to continue
23. Press "Enter" again
24. Press "y" to accept
25. Press "y" again
26. Restoring should be started now
27. Now drink a coffee and look for some interesting news on the internet
28. If restoring was finished press "Enter"
29. Select poweroff
30. Remove all USB Disks
31. Switch PC on
32. If necessary press "F" in blue XUbuntu screen to fix some errors automativcally
33. Maybe you are asked which panel config to use after starting. Select "Use Default Config"

3.2 WLAN Access

Each team gets one WLAN Access Point with the following login credentials:

- Setup user: admin, Setup password: admin
- WLAN SSID: AADC_WLAN_0xx_0xx, Passwort: aadc2016

3.2.1   SDKs

- Qt
  - o Version: 4.7.1 (required by ADTF)
  - o Path: /opt/qt/4.7.1
  - o Download: http://download.qt.io/archive/qt/4.7/
  - o License: LGPL License
  - o Compile Options:

```
./configure -prefix /opt/qt/4.7.1 -no-webkit -no-declarative
```

  - o
- OpenCV
  - o Version: 3.0.0
  - o Path: /opt/opencv/3.0.0
  - o Download: http://opencv.org/downloads.html
  - o License: BSD License
  - o Compile Options:

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/opt/opencv/3.0.0 -D
WITH_TBB=ON -D WITH_V4L=ON -D WITH_OPENGL=ON -D INSTALL_C_EXAMPLES=ON -D
BUILD_EXAMPLES=ON -D WITH_IPP=ON -D WITH_OPENNI2=ON -D WITH_OPENNI=OFF ..
```

- Aruco
  - o Version: 1.3.0
  - o Path: /opt/aruco/1.3.0/
  - o Download: http://sourceforge.net/projects/aruco/files/1.3.0/aruco-1.3.0.tgz/download
  - o License: BSD License
  - o Compile Options:

```
cmake -D CMAKE_INSTALL_PREFIX=/opt/aruco/1.3.0/ -D BUILD_SHARED_LIBS=ON -D
BUILD_UTILS=ON ..
```

- OpenNi
  - o Version: 2.3.0
  - o Path: /opt/openni/2.3.0
  - o Download: https://github.com/OpenNI/OpenNI2/archive/master.zip
  - o License: Apache License, Ver.2.0
- OpenSceneGraph
  - o Version: 3.2.0
  - o Path: /opt/osg/3.2.0/
  - o Download:
             http://trac.openscenegraph.org/downloads/developer_releases/OpenSceneGraph-3.2.0.zip
  - o License: OSGPL (OpenSceneGraph Publi License)
  - o Compile Options:

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/opt/opencv/3.0.0 -D
WITH_TBB=ON -D WITH_V4L=ON -D WITH_OPENGL=ON -D INSTALL_C_EXAMPLES=ON -D
```

```
BUILD_EXAMPLES=ON -D WITH_IPP=ON -D WITH_OPENNI2=ON -D WITH_OPENNI=OFF ..
```

3.3 ADTF

3.3.1   ADTF License

Each vehicle is delivered with one valid ADTF License file. It is located in

```
/home/aadc/ADTF License/
```

3.3.2   ADTF Installation

- Path: /opt/adtf/2.13.1
- Version: 2.13.1
- Toolboxes:
    - ADTF Display Toolbox 2.1.1

3.3.3   ADTF Documentation

The ADTF Documentation is located at:

```
/opt/adtf/2.13.1/doc/adtf_sdk.chm

/opt/adtf/2.13.1/doc/ADTFUserManual.pdf
```

3.4 AADC ADTF Source Package

The vehicles are delivered with a Source Package which helps the teams getting started with their own ADTF filter development. It provides a lot of standard functionalities to use the vehicle, meaning how to read its sensors, control its actuators and understand the conditions of the competition.

The Source Package contains a lot of different ADTF Filters in three categories:

- AADC Base Filters
- AADC Demo Filters
- AADC User Filters

The filters in AADC Base serve their users mainly as the standard communication with the Arduinos and the implementation of the communication with the Jury during the competition. These filters must not be modified by the teams. If any question or errors occur refer to the website forum and the described procedure in the regulations. The Base Filter are described in 3.4.3.

The filters in AADC Demo offer an extended scope of functionalities for the use of the vehicle. The package contains filters to visualize sensor values, to do calibration procedures, to convert values or several ways to control the speed and steering controllers of the car. All these filters can be modified by the teams or can be used as a start-up for their own algorithms and implementations. The Demo Filter are described in 3.4.4.

The AADC User Filter package contains one template filter without any specific functions. All the individual filters of the teams have to be placed in this package and its corresponding directory. The User Filter are described in **Fehler! Verweisquelle konnte nicht gefunden werden.**.

The Source Package also contains some ADTF Configurations, which explain the usage of the compiled filters. The ADTF Configuration are described in 3.4.5.

The Source Package works both on Linux and Windows. For developing on Windows some ADTF Licenses have to be requested (chapter 3.4.1.1).

3.4.1   Building Filters

All the three categories in the source package have a prepared build environment including a complete CMake-Project.

The user has to execute the corresponding script to build the sources.

The delivered vehicle contains the latest source package in the folder:

```
/home/aadc/AADC
```

The source code is located in:

```
/home/aadc/AADC/src
```

In this folder are three build scripts each for Windows and for Linux.

- Windows:
  - build_base_win.bat
  - build_demo_win.bat
  - build_user_win.bat
- Linux:
  - build_base.sh
  - build_demo.sh
  - build_user.sh

On Windows these scripts create a build in form of a Visual Studio projects that can be opened and compiled using Visual Studio. On Linux these scripts create a CMake Project and the gcc is used to build the project. To use the CMake Project in another IDE the CMake-Files and especially the Generator has to be adjusted.

For development it is recommended to choose your own development IDE (CodeBlocks, Eclipse...) and load the CMake projects there.

A lot of useful Information can be found also in the ADTF SDK. Search for some of following topics:

- ADTF CMake Introduction
- Using CMake to build the ADTF Examples
- Programming ADTF
- ADTF Plugin SDK - Developing my first ADTF plugin

The SDK can be opened from ADTF via Help -> SDK Documentation.

3.4.1.1 Development under Windows

The vehicle is delivered with Linux and a valid ADTF license for this vehicle. For ADTF development, students of universities can request additional ADTF Licenses. Please refer to www.elektrobit.com.

Please send an informal request with the formular found at https://automotive.elektrobit.com/contact-us/?type=34&prod=15 or send an informal request to support.adtf@elektrobit.com . Elektrobit will reply to you as soon as possible. After signing some papers by the professorships they can get as many licenses as needed.
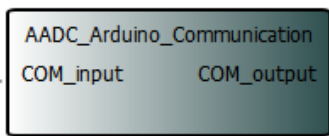
3.4.2 Starting Configurations

In the top folder of the source package are several scripts to start the delivered configurations:

- Windows:
    - start_aadc_user_project.bat -> starts the User Configuration
    - start_aadc_example_project.bat -> starts the Example Configuration
- Linux:
    - start_aadc_user_project.sh -> starts the User Configuration
    - start_aadc_example_project.sh -> starts the Example Configuration

3.4.3 AADC Base Filter

3.4.3.1 Arduino Communication

Category:	Bridge Device
Version:	1.0.0
Filename:	aadc_arduinoCommunication.plb
Source:	src\adtfBase\src\arduino\AADC_ArduinoCommunication
Guid:	adtf.aadc.arduinoCommunication

This is the filter which implements the communication with the Arduino. It sends the package data of the media samples given on the input pin to Arduino and transmits all the received packages from the Arduino to the output pin. With the property COM Port, the serial port to be used can be set. On Linux it should be something like /dev/ttyACM0 to /dev/ttyACM3 and on Windows it corresponds to the number of the COM Port given by the system, preceded by a \\.\\ .

For this filter it does not matter which Arduino of the car is bounded to the port, since it is the only filter available for the Arduino. To Read or Write all Arduinos you must add four instances of this filter to the ADTF Configuration.

Pinname	MediaDescription	Notes

Inputs	COM_input	tArduinoData	Struct
Outputs	COM_output	tArduinoData	Struct
	Name	**Description**	**Notes**
Property	COM Port	The device name for the COM port of the Arduino. On Windows the notation is typically \\\\.\\COMx, on Linux it is /dev/ttyACM0.	Default: \\.\\COM0
	Connection Timeout in millisec	The value for the timeout, meaning how long is waited for the Arduino to send something onto the serial interface	Default: 3000 Range: [0...5000]

3.4.3.2   Arduino Sensors

Category:	Sensor Device
Version:	1.0.0
Filename:	aadc_arduinoSensors.plb
Source:	src\adtfBase\src\arduino\AADC_ArduinoSensors
Guid:	adtf.aadc.arduinoSensors

This filter receives all the media samples from the Arduino Communication Filter and sends them to the different output pins depending on their content (i.e. ID). The outputs of all four Arduino Communication Filters interacting with the four different Arduinos mounted in the car, should be connected with this filter to provide all sensor data on the output pins.

The samples on the pin Ultrasonic contain structs with all sensor data from all ultrasonic sensors up to the update time of the sample.

	Pinname	**MediaDescription**	**Notes**
Inputs	ArduinoCOM_input	tArduinoData	Struct
Outputs	Ultrasonic_Front_Left	tSignalValue	Value in m
	Ultrasonic_Front_Center_Left	tSignalValue	Value in m
	Ultrasonic_Front_Center	tSignalValue	Value in m
	Ultrasonic_Front_Center_Right	tSignalValue	Value in m

	Ultrasonic_Front_Right	tSignalValue	Value in m
	Ultrasonic_Side_Left	tSignalValue	Value in m
	Ultrasonic_Side_Right	tSignalValue	Value in m
	Ultrasonic_Rear_Left	tSignalValue	Value in m
	Ultrasonic_Rear_Center	tSignalValue	Value in m
	Ultrasonic_Rear_Right	tSignalValue	Value in m
	Ultrasonic_Struct	tUltrasonicStruct	Struct with all Ultrasonic Values in cm and their Arduino timestampls
	Voltage_Measurement	tSignalValue	Value in V
	Voltage_SpeedCntrl	tSignalValue	Value in V
	InerMeasUnit_Struct	tInerMeasUnitData	Struct with IMU Data
	WheelLeft_Struct	tWheelData	Struct with Data from left wheel
	WheelRight_Struct	tWheelData	Struct with Data from right wheel
	Name	**Description**	**Notes**
Property	Debug Output to Console	If enabled additional debug information is printed to the console (Warning: decreases performance)	Default: false

3.4.3.3   Arduino Actuators

	Category:	Sensor Device
	Version:	1.0.0
	Filename:	aadc_arduinoActuators.plb
	Source:	src\adtfBase\src\arduino\AADC_ArduinoActuators
	Guid:	adtf.aadc.arduinoActuators

To this filter the user can send the data which should be send to the Arduino. With input pins the speed and steering controller can be controlled, the light can be switched on and off, the watchdog can be triggered or the Emergency stop can be called.

The lights can be switched with media sample of the type tBoolSignalValue. The value "true" in bValue means lights on, the value "false" in bValue means lights off.

The pin Watchdog_Alive_Flag also waits for a media sample of type tBoolSignalValue. If the media sample contains a "true" in bValue a trigger is send to the Arduino. If this trigger is not received for more than half a second, the relais on the board toggles and the Speed Controller is

turned off.

When the speed controller is first switched on it will last some seconds for initialization. During this time the speed controller requires the "Zero Position" which corresponds to an input value of 90.0. To prevent troubles with the initialization phase the media samples received on the input pin SpeedController are set to 90.0 during this time period. Nevertheless incoming samples are necessary during that time to trigger the messages sent to the Arduino.

The range for the samples for the input pin Speed Controller is between 0.0 and 180.0, resulting the already mentioned zero position at 90.0. This way the angle of the servo motor is set to values in-between the same range 0.0° and 180.0°, corresponding to full speed reverse and forward. The direction and also the real car speed depends on the calibration of the speed controller itself. For further information please refer to Chapter Calibration.
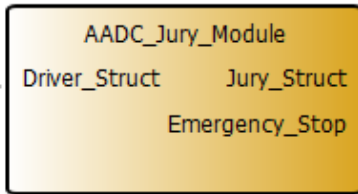
The range for the samples for the input pin Steering Controller is between 60.0 and 120.0 and corresponds to an angle of the steering servo between 60.0° and 120.0°. This results in a steering angle between -30.0° and 30.0° around the straight ahead position.

	Pinname	MediaDescription	Notes
Inputs	SpeedController	tSignalValue	Range [0...180]; Servo Angle
	SteeringController	tSignalValue	Range [60...120]; Servo Angle
	headLightEnabled	tBoolSignalValue	True: On; False: Off
	turnSignalLeftEnabled	tBoolSignalValue	True: On; False: Off
	turnSignalRightEnabled	tBoolSignalValue	True: On; False: Off
	hazzardLightsEnabled	tBoolSignalValue	True: On; False: Off
	brakeLightEnabled	tBoolSignalValue	True: On; False: Off
	reverseLightsEnabled	tBoolSignalValue	True: On; False: Off
	Watchdog_Alive_Flag	tBoolSignalValue	True: Activate; Deactivate: Off
	Emergency_Stop	tBoolSignalValue	True: On; False: Off
Outputs	ArduinoCOM_output	tArduinoData	Struct
	Name	**Description**	**Notes**
Property	Debug Output to Console	If enabled additional debug information is printed to the console (Warning: decreases performance)	Default: false
	Omit Watchdog Signal from Console Output	If disabled all the watchdog sample are also printed to console (Warning: decreases performance)	Default: true

3.4.3.4 Watchdog Trigger

Category:	Tool
Version:	1.0.0
Filename:	aadc_watchdogTrigger.plb
Source:	src\adtfBase\src\watchdog\AADC_Watchdog Trigger
Guid:	adtf.aadc.watchdogTrigger

The filter generates the samples of type tBoolSignalValue which can be passed to the Arduino Actuators filter to keep the Speed Controller alive. The Transmit Rate can be altered in the properties, although there is usually no need for adjustment.

	Pinname	MediaDescription	Notes
Outputs	WatchdogAliveSignal	tBoolSignalValue	
	Name	**Description**	**Notes**
Property	Transmit rate in ms	Sets the interval between two sent watchdog triggers in msec	Default: 250 Range: [0...250]

3.4.3.5 Jury Module (DEPRECATED)

Category:	OBJCAT_Tool
Version:	1.0.0
Filename:	aadc_juryModule.plb
Source:	src\adtfBase\src\jury\AADC_JuryModule
Guid:	adtf.aadc.juryModule

This filter is deprecated, the standalone Jury Application in Chapter 3.4.6 must be used instead.

This filter is used by the jury to control the vehicles during the competition.

The Module sends samples to a driver module which has to be implemented by the teams but can be based on the Demo filters. The sent samples contain the tJuryStruct and are used to check or set the state of the cars. The struct is defined as followed:

```
typedef struct
{
    tInt8 i8ActionID;
    tInt16 i16ManeuverEntry;
} tJuryStruct;
```

The i8ActionID contain one of the following actions:
- action_isReady: The vehicle is asked if it is ready to start the maneuver given in

i16ManeuverEntry.
- action_start: After receiving this sample the car should start the maneuver given in i16ManeuverEntry. Before that the sample the same maneuver is asked if it is ready.
- action_stop: After receiving this sample the car should stop the maneuver given in i16ManeuverEntry. If the car is in a different maneuver it should be stopped as well.

The output pin Emergency_Stop should be connected directly to the pin Emergency_Stop of the Arduino Actuator Filter. It forces the Speed Controller to be switched off immediately.

The struct tJuryStruct is defined in aadc_structs.h in src\aadcBase\include and the used enums are defined in juryEnums.h in src\aadcBase\include.
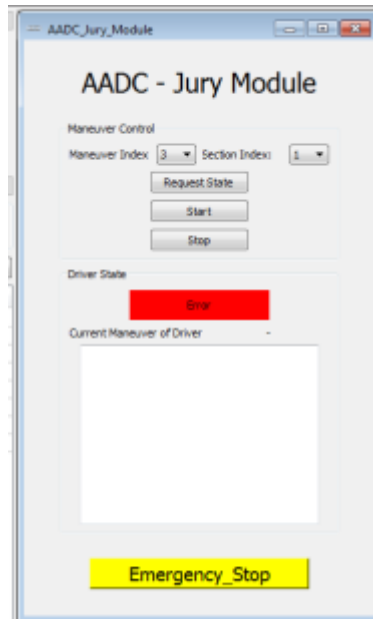
Figure 9 GUI Jury Module

In the upper part of the GUI, the user can send messages to car. If a maneuver file was loaded successfully one can select a section or a maneuver index and press "Request State", "Start" or "Stop" to transmit the corresponding sample to the vehicle.
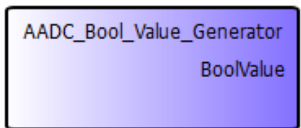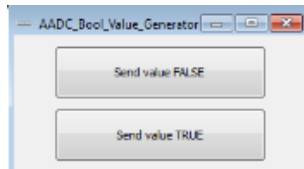
The last received state from the driver is shown in the middle of the GUI as well as on the console view.

	Pinname	MediaDescription	Notes
Inputs	Driver_Struct	tDriverStruct	Struct to be sent by the Driver Module
Outputs	Jury_Struct	tJuryStruct	Struct to be sent to the Driver Module
	Emergency_Stop	tBoolSignalValue	Value to be sent to for Emergency Stop
	Name	**Description**	**Notes**
Property	ManeuverFile	Here you have to set the maneuver file of type xml to be used in this filter.	
	Send time interval in sec	Specifies the time between two send media samples in sec.	Default: 1
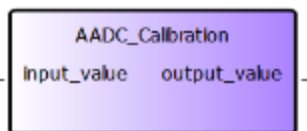
Debug Output to Console	If enabled additional debug information is printed to the console (Warning: decreases performance)	Default: false

3.4.4   AADC Demo Filters

3.4.4.1   Bool Value Generator
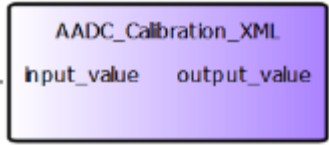
	Category:	OBJCAT_Auxiliary
AADC_Bool_Value_Generator	**Version:**	1.0.0
BoolValue	**Filename:**	aadc_boolValueGenerator.plb
	Source:	src\aadcDemo\src\helper\AADC_BoolValueGenerator
	Guid:	aadc_boolValueGenerator

With this small helper, simple Media Samples of the tBoolSignalValue can be generated. If started, a GUI with two buttons is pops up. When clicking on "Send Value FALSE" a Media Sample with "False" in the media description element bValue is transmitted, when clicking on "Send Value TRUE" a Media Sample with "True" is transmitted.

Figure 18 GUI Bool Value Generator

	Pinname	MediaDescription	Notes
Outputs	BoolValue	tBoolSignalValue	

3.4.4.2   Calibration

	Category:	Data Filter
AADC_Calibration	**Version:**	1.0.0
input_value output_value	**Filename:**	aadc_calibration.plb
	Source:	src\adtfBase\src\datafilter\AADC_Calibration
	Guid:	adtf.aadc.aadc_calibration

This filter does a simply scaling of the value given on the input pin with the value set in the property "Scale Factor" and returns the result on the output pin.

	Pinname	MediaDescription	Notes

Inputs	input_value	tSignalValue	value to be calibrated
Outputs	output_value	tSignalValue	calibrated value
	Name	**Description**	**Notes**
Property	Scale Factor	The input value is multiplied with the here given factor	Default: 1

3.4.4.3 Calibration XML
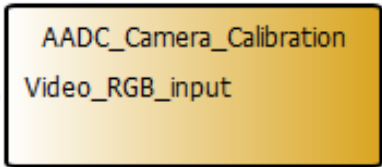
Category:	Data Filter
Version:	1.0.0
Filename:	aadc_calibrationXml.plb
Source:	src\adtfBase\src\datafilter\AADC_CalibrationXml
Guid:	adtf.aadc. aadc_calibrationXML

Contrary to the Calibration Filter, this filter is for an extended calibration. The user has to generate a XML-File set in the corresponding property. This XML-File has to include a calibration table with x- and y-values and a mode which has to be used for interpolation.

When the filter receives an input value it looks on the x-Axis of the calibration table and gets the corresponding value on the y-Axis using the set interpolation. This result is transmitted on the pin output_value. If the value on the input pin is greater than the maximum value in the table the maximum value is used, if it is smaller than the minimum value the minimum value is used.

The x-Values in the table must be in increasing order otherwise the calibration does not work and prints an error.

The structure of the XML has to be like this:

```xml
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<calibration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <settings>
      <mode>linear</mode>
 </settings>
 <supportingPoints>
      <point>
            <xValue>307</xValue>
            <yValue>-45</yValue>
      </point>
      <point>
            <xValue>572</xValue>
            <yValue>45</yValue>
      </point>
 </supportingPoints>
```

```
        </calibration>
```

If no interpolation mode is set in the XML – File, the interpolation mode is chosen from the filter properties (i.e. the mode in the XML-File always overwrite the filter property).

	Pinname	MediaDescription	Notes
Inputs	input_value	tSignalValue	value to be calibrated
Outputs	output_value	tSignalValue	calibrated value
	Name	**Description**	**Notes**
Property	Interpolation	Sets the mode of interpolation between the given points in the XML	Default: Linear Range: Linear, Cubic, None
	Configuration File For Interpolation	The XML to be loaded has to be set here	
	Border Warnings to Console	If enabled a warning is printed to console each time the border points of the given xml are reached	Default: False
	Print initial table to Console	If enabled the loaded points of the interpolation table of the XML are printed to console	Default: False

3.4.4.4   Camera Calibration

Category:	OBJCAT_Tool
Version:	1.0.0
Filename:	aadc_cameraCalibration.plb
Source:	src\aadcDemo\src\xtion\AADC_CameraCalibration
Guid:	aadc_cameraCalibration

This filter is for doing the intrinsic calibration of the camera in use. After starting, a GUI is shown that displays the video stream. There are two buttons available for starting the calibration or saving the file.

The calibration is done with the standard opencv function for intrinsic calibration described in the opencv documentation. The main functions are used from opencv samples in (opencv/samples/cpp/calibration.cpp). For further information have look in this documentation.

To perform the calibration with this ADTF Filter, a calibration pattern has to be printed at first. For this purpose, a standard chessboard pattern is located in *src\aadcDemo\src\xtion\AADC_CameraCalibration\ pattern.png.* It is recommended to print it on an A3 paper and use either a thick paper or stick the paper to solid background.

Afterwards the RGB Output of the Xtion Filter has to be connected to the input pin of this filter and the configuration has to be started. Do not forget to set the setup file for the Xtion Filter.

The side length of one square of the printed chessboard pattern has to be measured and the length be set in the property Square Size in the Filter (unit is meter).
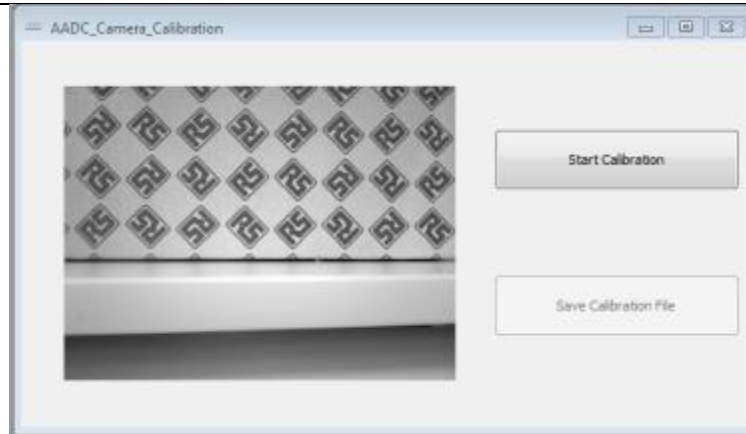
Figure 19 GUI Camera Calibration

When the configuration is started you can click "Start Calibration" in the GUI of this filter. After that the algorithms search for the chessboard on the image. If a chessboard with the given number of squares is detected it is marked with some diagonal lines. During this procedure the chessboard has to be held in different views and angles the get a better calibration. In the console is logged how many datasets have been collected and how many are still needed.
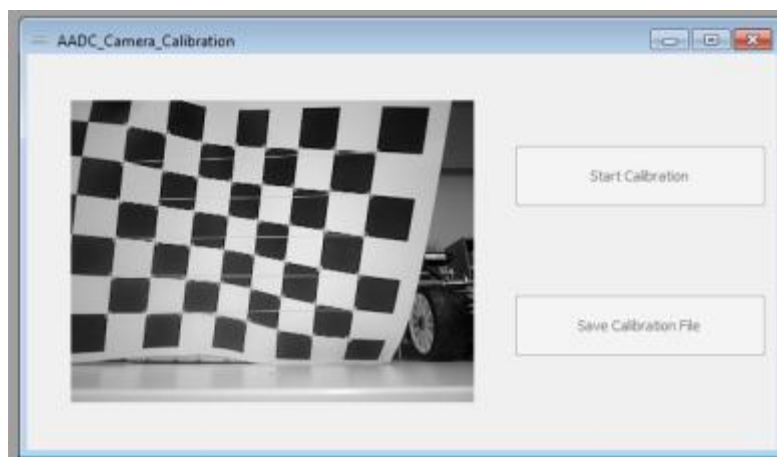
Figure 20 Chessboard Detection in Calibration

All the calibration dataset being collected the parameters are saved as a standard opencv-yml-File by pressing button „Save Calibration File".
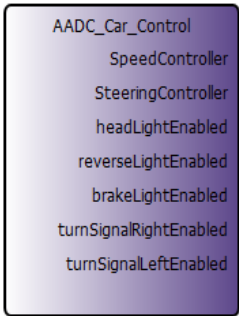
In doing so the file will contain the following parameters:

- Camera Matrix
- Distortion Coefficients
- Reprojection Errors
- Extrinsic Parameters

The number of squares as well as the square size can be set as a filter property. Please note further that especially these properties of the printed out chessboard have to be set in advance.

	Pinname	MediaDescription	Notes
Inputs	Video_RGB_input	Video Pin	
	Name	**Description**	**Notes**
Property	Width [number of squares]	The number of squares in horizontal axis (Range: 5 to 15)	Default: 5 Range: 5...15
	Height [number of squares]	The number of squares in vertical axis (Range: 5 to 15)	Default: 6 Range: 5...15
	Square Size	Square size (length of one side) in meters (0.036 by default)	Default: 0.036 Range: 0...20
	Aspect Ratio	Fix aspect ratio (fx/fy) (1 by default)	Default: 1.0 Range: 0...20
	Calibration Pattern	Defines the pattern which is used for calibration	Default: Chessboard

3.4.4.5   Car Control

AADC_Car_Control SpeedController SteeringController headLightEnabled reverseLightEnabled brakeLightEnabled turnSignalRightEnabled turnSignalLeftEnabled	**Category:**	Application
	Version:	1.0.0
	Filename:	aadc_carControl.plb
	Source:	src\adtfBase\src\helper\AADC_CarControl
	Guid:	adtf.aadc.aadc_carControl

With this filter the basic functions of the car can be controlled. You can set the steering and the speed controller with the two bars or use the keys described in the window. To use the keys the focus has to be set on this window by clicking in the window or the title bar. With the buttons at the bottom the lights of the car can be switched on or off.
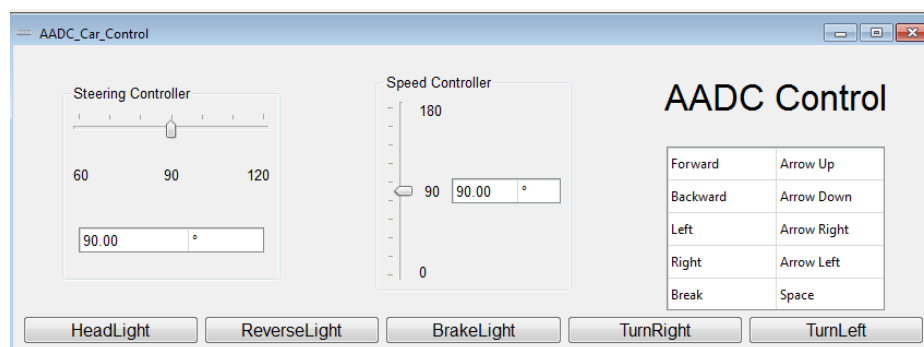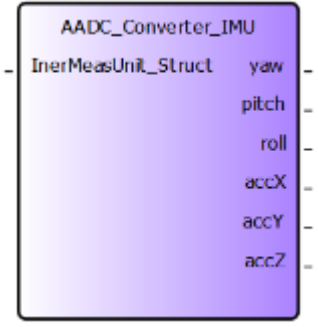
Figure 21 GUI Car Control Filter

	Pinname	MediaDescription	Notes
Outputs	SpeedController	tSignalValue	Range: [0....180]; Servo Angle

	Name	Description	Notes
	SteeringController	tSignalValue	Range: [0….180]; Servo Angle
	headLightEnabled	tBoolSignalValue	true or false to switch on or off
	reverseLightEnabled	tBoolSignalValue	true or false to switch on or off
	brakeLightEnabled	tBoolSignalValue	true or false to switch on or off
	turnSignalRightEnabled	tBoolSignalValue	true or false to switch on or off
	turnSignalLeftEnabled	tBoolSignalValue	true or false to switch on or off
	Name	**Description**	**Notes**
Property	Actuator Update Rate [Hz]	defines how much updates for steering and speed controller are sent in one second (Range: 0 to 100 Hz)	Default: 30 Range: [0…100]
	Actuator Startup Time Delay [sec]	defines at what delay the first actuator value should be send to arduino (Range: 0 to 10 sec)	Default: 5 Range: [0…10]
	Zero Position Steering Controller [°]	defines the offset for the zero position of the steering controller (Range: -10° to 10°)	Default: 0 Range: [-10…10]
	Zero Position Speed Controller [°]	defines the offset for the zero position of the speed controller (Range: -10° to 10°)	Default: 0 Range: [-10…10]
	Enable Speed Controller Fallback	When enabled the speed controller value falls back to zero automatically.	Default: True

3.4.4.6   Converter IMU

Category:	Data Filter
Version:	1.0.0
Filename:	aadc_converterIMU.plb
Source:	src\adtfBase\src\datafilter\AADC_ConverterIMU
Guid:	adtf.aadc.aadc_converterIMU

This filter takes the struct given by the Arduino Sensors Filter with the data from the IMU and converts the data to Euler Angles and the acceleration splitted to the three axes. The IMU data directly from Sensor is in Quaternions which is not easy to understand. The unit of the acceleration data is in $\frac{m}{sec^2}$, the yaw, pitch and roll angles are in radiant. With the property "Euler Angle System" the Euler System to be used has to be set. The different systems varies in their order of rotation of the X-, Y- and Z-Axis and if static or rotating axis are used.

Static Axes:

EulOrdXYZs, EulOrdXYXs, EulOrdXZYs, EulOrdXZXs,

 EulOrdYZXs, EulOrdYZYs, EulOrdYXZs, EulOrdYXYs,

 EulOrdZXYs, EulOrdZXZs, EulOrdZYXs, EulOrdZYZs

Rotating axes:
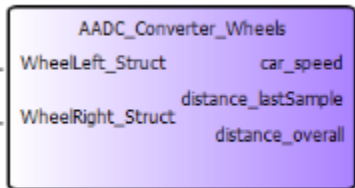
EulOrdZYXr, EulOrdXYXr, EulOrdYZXr, EulOrdXZXr,

EulOrdXZYr, EulOrdYZYr, EulOrdZXYr, EulOrdYXYr,

EulOrdYXZr, EulOrdZXZr, EulOrdXYZr, EulOrdZYZr

For further information refer to "Graphic Germs IV" from Ken Shoemake or other appropriate literature.

	Pinname	MediaDescription	Notes
Inputs	InerMeasUnit_Struct	tInerMeasUnitData	Struct
Outputs	yaw	tArduinoData	yaw angle in radiant
	pitch	tSignalValue	pitch angle in radiant
	roll	tSignalValue	roll angle in radiant
	accX	tSignalValue	acceleration in x-axis in $\frac{m}{sec^2}$
	accY	tSignalValue	acceleration in y-axis in $\frac{m}{sec^2}$
	accZ	tSignalValue	acceleration in z-axis in $\frac{m}{sec^2}$

	Name	Description	Notes
Property	Debug Output to Console	If enabled additional debug information is printed to the console (Warning: decreases performance)	Default: false
	Euler Angle System	Here you can set the euler system which should be used for transformation. It defines the order of rotation of the axes and whether static or rotating axes are used.	Default: EulOrdXYZs

3.4.4.7   Converter Wheels

	Category:	OBJCAT_DataFilter
	Version:	1.0.0
	Filename:	aadc_converterWheels.plb
	Source:	src\aadcDemo\src\datafilter\AADC_ConverterWheels
	Guid:	aadc_converterWheels

AADC_Converter_Wheels
WheelLeft_Struct   car_speed
                    distance_lastSample
WheelRight_Struct
                    distance_overall

This filter calculates the speed of the vehicle and some distance measurements. It takes the wheel structs containing the direction and the interrupt counter of both wheels from the Arduino sensor as input values. The wheels have a small disk mounted at the inner side and a Transmissive Encoder Sensor which detects the slots in the rotating disk. The detected slots are transmitted to an interrupt routine at an Arduino that counts the ticks and sends them periodically to ADTF.

The speed is calculated by the difference between the increasing tick counter in the incoming media samples and the corresponding time difference between the media samples. For a correct calculation, the wheel circumference has to be set to the correct value in the properties, the default value is 0.34m.

The output samples are triggered to the wheel right struct, so if no samples are received from that sensor no output samples are generated by this filter.

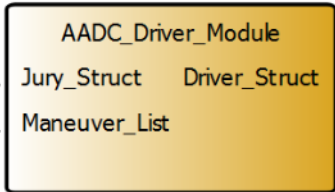If the property *Filtering enabled* is set to true, a first order filtering is applied to smooth the signals. The first order filter constant can also be set in the properties.

	Pinname	MediaDescription	Notes
Inputs	WheelLeft_Struct	tWheelData	The structs with wheel data from the left wheel
	WheelRight_Struct	tWheelData	The structs with wheel data from the right wheel
Outputs	car_speed	tSignalValue	Speed of car in $\frac{m}{sec^2}$
	distance_lastSample	tSignalValue	Distance since last sample in meter
	distance_overall	tSignalValue	Overall distance since initialization
	Name	**Description**	**Notes**
Property	Wheel circumference	Set the wheel circumference in meter here	Default: 0.34
	Filter constant first order	Set the filter constant for first order here	Default: 0.05

	Filtering enabled	Enables or disables the low pass filtering of speed result	True or False

3.4.4.8   Driver Module

Category:	OBJCAT_Tool	
Version:	1.1.0.	
Filename:	aadc_driverModule.plb	
Source:	src\adtfBase\src\jury\AADC_DriverModule	
Guid:	adtf.aadc.driverModule	

AADC_Driver_Module

Jury_Struct   Driver_Struct

Maneuver_List

This filter was developed as a prototype to explain the interaction with the Jury Module.

It receives the structs from the Jury module and shows them in the middle of the GUI. The user can respond the received messages with the four buttons "Ready to Start", "Running", "Error" and "Complete".

After clicking on the button the sample is transmitted on the output pin Driver_Struct an contains the following struct:

```
typedef struct
{
    tInt8 i8StateID;
    tInt16 i16ManeuverEntry;
} tDriverStruct;
```

The possible i8StateID are:
- stateCar_Error: This is sent if some error occurred on the car.
- stateCar_Ready: If the car is ready to start a maneuver ID this state is sent including the maneuver ID in i16ManeuverEntry.
- stateCar_Running: Sent during running the maneuver contained in i16ManeuverEntry
- stateCar_Complete: Sent if the car finished the whole maneuver list.
- stateCar_Startup: Sent at the initial phase to indicate that car is working properly
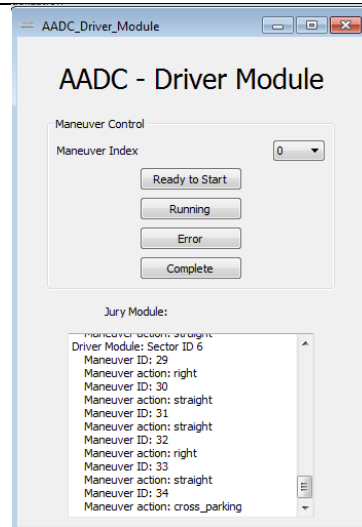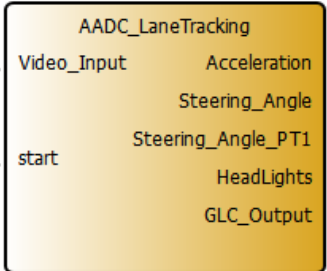
Figure 10 GUI Driver Module

The struct tDriverStruct is defined in aadc_structs.h in src\aadcBase\include and the used enums are defined in juryEnums.h in src\aadcBase\include

The teams must not implement any filter containing a Qt GUI because there is no opportunity to control the car with a GUI in the competition. The only way to interact with the car is through the jury module which is controlled by the jury.

	Pinname	MediaDescription	Notes
Inputs	Jury_Struct	tJuryStruct	Struct from Jury Module
	Maneuver_List	tManeuverList	Maneuver list from Jury Module
Outputs	Driver_Struct	tDriverStruct	Struct to Jury Module
	Name	**Description**	**Notes**
Property	Debug Output to Console	If enabled additional debug information is printed to the console (Warning: decreases performance)	Default: false

3.4.4.9   Lane Tracking

Category:	OBJCAT_Tool
Version:	1.0.0
Filename:	aadc_laneTracking.plb
Source:	src\aadcDemo\src\algorithms\AADC_LaneTracking
Guid:	aadc_laneTracking

This filter does a lane tracking for the car. The source code contains the documentation of this filter.

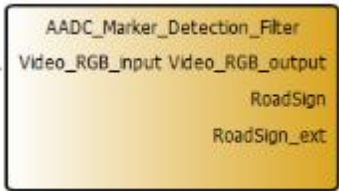|	Pinname	MediaDescription	Notes
Inputs	Video_Input	Video Pin	
	start	tBoolSignalValue	
Outputs	Acceleration	tSignalValue	Speed in $\frac{m}{sec^2}$
	Steering_Angle	tSignalValue	Servo Angle in °
	Steering_Angle_PT1	tSignalValue	Servo Angle in °
	HeadLights	tBoolSignalValue	Turn Headlights on or off
	GLC_Output	GCL Command	Overlay for Video Image
	Name	**Description**	**Notes**
Property	LaneDetection::Near area::Near Line	The y value of the near line used for lateral control.	
	LaneDetection::Near area::Max offset for near line	The maximum offset to adjust near line to the current speed.	
	LaneDetection::Near area::Lane width min near	The Minimum value for the near lane width. If the calculated lane width is smaller than this value the blind count will be increased.	
	LaneDetection::Near area::Lane width max near	The Maximum value for the near lane width. If the calculated lane width is greater than this value the blind count will be increased.	
	LaneDetection::Far area::Far Line	The y value of the far line used for longitudinal control.	
	LaneDetection::Far area::Lane width min far	The Minimum value for the far lane width. If the calculated lane width is smaller than this value the blind count will be increased.	
	LaneDetection::Far area::Lane width max far	The Maximum value for the far lane width. If the calculated lane width is greater than this value the blind count will be increased.	
	LaneDetection::Camera Offset	The offset of the camera in relation to the center of the car.	
	LaneDetection::ThresholdValue	The threshold value for canny to detect lines.	
	LongitudinalControl::Max Acceleration	Acceleration value used on a straight.	
	LongitudinalControl::Min Acceleration	Acceleration value used in a turn.	
	LongitudinalControl::Far Near difference	The difference between far and near point in pixel.	
	Common::Show Debug	If true, the opencv windows will be shown and the gcl output is enabled.	
	Common::Enable Lightbeam Trigger	If true, start_trigger pin will be used to start driving.	
	Common::Drive Time in milliseconds	If enable lightbeam trigger is set to true, this value will be used to stop driving after the given time. If the value is 0, the stop trigger is disabled.	
	Common::Emergency Stop Time in milliseconds	f enable lightbeam trigger is set to true, this value will be used to perform an emergency stop after the given time. If the value is 0, the emergency stop trigger is disabled.	

PID::Controller Proportional Gain	The proportional gain of the PID controller.	
PID::Controller Integral Gain	The integral gain of the PID controller.	
PID::Controller Differential Gain	The differential gain of the PID controller.	
PT1::Tau	The tau value of the PT1 controller.	
PT1::Sample_Time	The sample time of the PT1 controller.	
PT1::InputFactor	The factor to normalize the input value (difference of the lane center and the place to be) to the range of the output values.	

3.4.4.10   Marker Detection Filter

Category:	OBJCAT_Tool
Version:	1.0.0
Filename:	aadc_markerDetection.plb
Source:	src\aadcDemo\src\algorithms\AADC_MarkerDetection
Guid:	aadc_markerDetection

This filter searches for Aruco markers in the given frame on the input video pin. It uses mainly the Aruco lib to find the markers and get the marker Id and their additional parameters and mark them optionally in the video frame. If one or more markers are detected in the frame samples on the pins RoadSign and RoadSign_ext are generated containing the parameters of the sign.

The samples of the pin RoadSign include the marker identifier and the image size of the marker. The samples of the pint RoadSign_ext include the marker identifier, the image size of the marker and the rotation and translation vector. For further description of the samples refer to chapter 0.
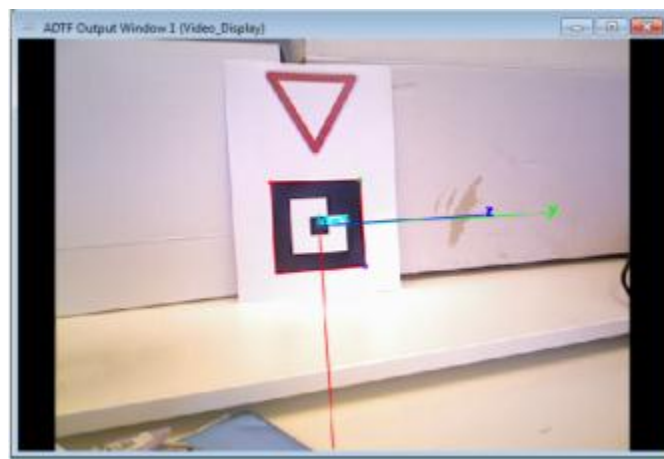
Figure 22 Video frame with detected marker and vectors

For the calculation of the rotation and translation vector the intrinsic parameters of the camera has to be known. These parameters can be get with the camera calibration filter and the calibration file generated by that filter loaded with in the property *Calibration File for used Camera.*

For the correct assignment of the marker IDs the dictionary file has to be set in the properties as well. The dictionary file includes the assignment of the bit codes of the markers to the predefined ID and has the following content:

```
%YAML:1.0
nmarkers: 12
markersize: 3
marker_0: "010010111"
marker_1: "001100010"
marker_2: "001001111"
marker_3: "110011101"
marker_4: "011001010"
marker_5: "110101111"
marker_6: "000111101"
marker_7: "011101000"
marker_8: "001010101"
marker_9: "010011001"
marker_10: "110011010"
marker_11: "010010100"
```

The IDs are mapped to the signs as the following table shows:

MARKER_ID_UNMARKEDINTERSECTION	0
MARKER_ID_STOPANDGIVEWAY	1
MARKER_ID_PARKINGAREA	2
MARKER_ID_HAVEWAY	3
MARKER_ID_AHEADONLY	4
MARKER_ID_GIVEWAY	5
MARKER_ID_PEDESTRIANCROSSING	6
MARKER_ID_ROUNDABOUT	7
MARKER_ID_NOOVERTAKING	8
MARKER_ID_NOENTRYVEHICULARTRAFFIC	9
MARKER_ID_ONEWAYSTREET	0

The named are enums are also contained in the file *aadc_roadSign_enums.h* in

src\aadcDemo\include\aadc_roadSign_enums.h . The dictionary contains the following signs:

ID 466

ID 140

ID 484

ID 371

ID 166

ID 491
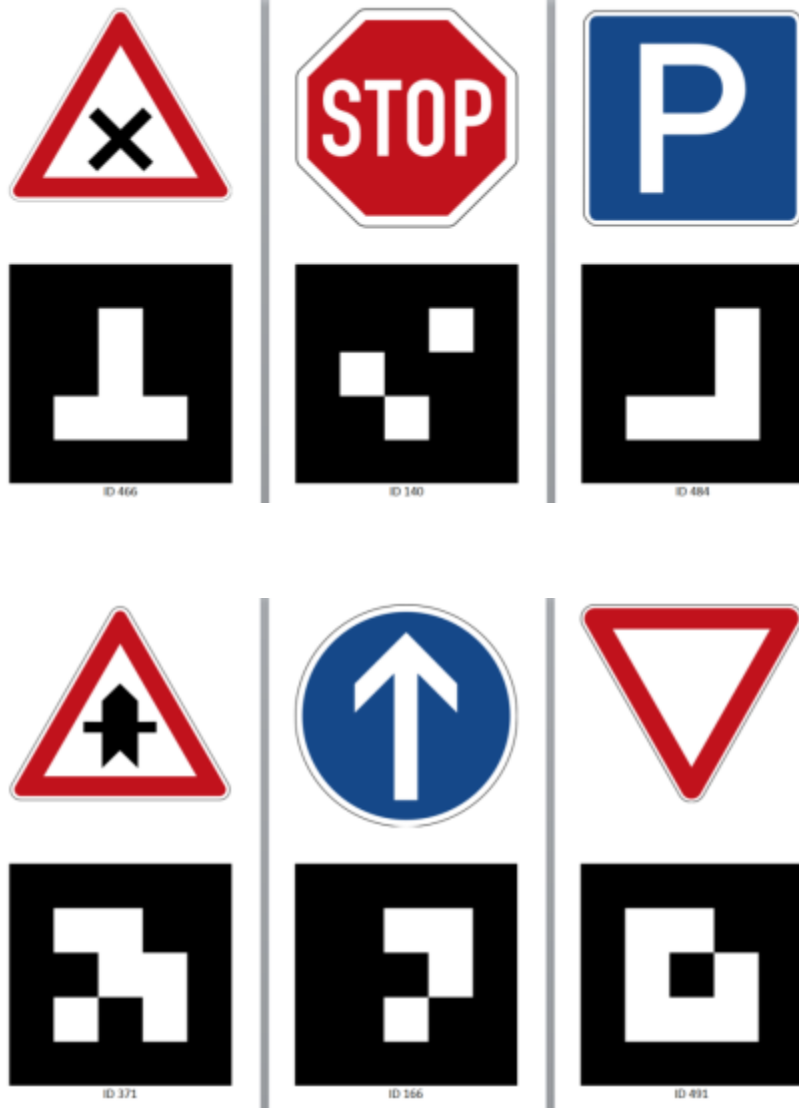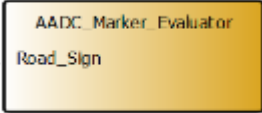
Figure 23 Road signs in dictionary file

	Pinname	MediaDescription	Notes
Inputs	Video_RGB_input	Video Pin	
Outputs	Video_RGB_output	Video Pin	
	RoadSign	tRoadSign	
	RoadSign_ext	tRoadSignExt	
	Name	**Description**	**Notes**
Property	Debug Output to Console	If enabled additional debug information is printed to the console (Warning: decreases performance)	Default: False Range: True or False
	Dictionary File For Markers	Here you have to set the dicitionary file which holds the marker ids and their content	
	Calibration File for used Camera	Here you have to set the file with calibration paraemters of the used camera	

		If enabled the video stream with the highlighted markers is transmitted at the output pin (Warning: decreases performance).	Default: 1: None Range: 1: None 2: Detected Signs
	Size of Markers	Size (length of one side) of markers in m	Default: 0.114

3.4.4.11   Marker Evaluator Filter

Category:	OBJCAT_Tool
Version:	1.0.0
Filename:	aadc_markerEvaluator.plb
Source:	src\aadcDemo\src\helper\AADC_Marker Evaluator
Guid:	aadc_markerEvaluator

The results of the marker detection filter can be visualized with this tool. The input pin Road_Sign has to be connected with the output pin Road_Sign of the Marker Detection Filter.
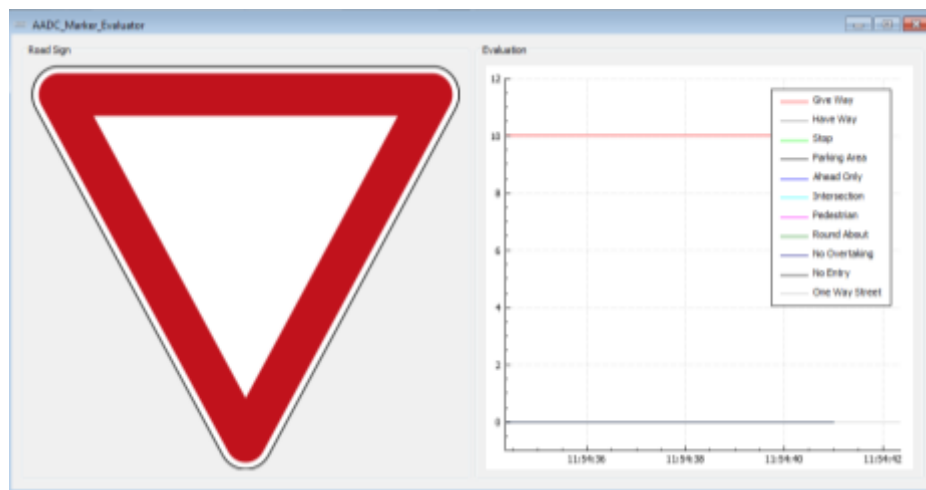
Figure 24 GUI Marker Evaluator Filter
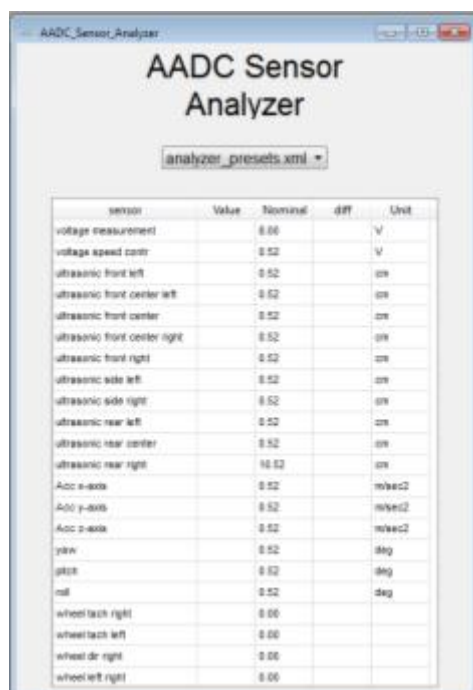
The GUI shows the symbol of the road sign with the highest frequency in the incoming samples. On the right side a graph is plotted which shows the frequency of all signs, i.e. Marker IDs, in the incoming Media Samples from the marker detection filter.

	Pinname	**MediaDescription**	**Notes**
Inputs	Road_Sign	tRoadSign	Has to be connected to Marker Detection Filter
	Name	**Description**	**Notes**
Property	Debug Output to Console	If enabled additional debug information is printed to the console (Warning: decreases performance)	Default: False Range: True and False

3.4.4.12   Sensor Analyzer

	Category:	OBJCAT_Application
	Version:	1.0.0
	Filename:	aadc_sensorAnalyzer.plb
	Source:	src\aadcDemo\src\helper\AADC_Sensor Analyzer\cSensorAnalyzer.h
	Guid:	aadc_sensorAnalyzer

AADC_Sensor_Analyzer
- Ultrasonic_Front_Left
- Ultrasonic_Front_Center_Left
- Ultrasonic_Front_Center
- Ultrasonic_Front_Center_Right
- Ultrasonic_Front_Right
- Ultrasonic_Side_Left
- Ultrasonic_Side_Right
- Ultrasonic_Rear_Left
- Ultrasonic_Rear_Center
- Ultrasonic_Rear_Right
- Voltage_Measurement
- Voltage_SpeedCntrl
- InerMeasUnit_Struct
- WheelLeft_Struct
- WheelRight_Struct

With this filter the sensors can be checked by comparing of their values with a set of predefined parameters. In order to check the ultrasonic sensors the vehicle can be put in a special environment with a measured distance to some obstacles around the car. The measured distance have to be written into an XML-file and the Sensor Analyzer will verify the signal values. The GUI plots the box with the values within the specified range in green color, boxes with values out of the specified range in red color.

Figure 25 GUI Sensor Analyzer

The XML-file contains presets for sensors by using the Sensor-ID, setting a nominal value as well as maximum positive and negative deviation. The Sensor-IDs can be found in the sample Sensorpreset-file *analyzer_presets.xml* in the folder *configuration_files\Sensorpresets*.

The full sample file is printed here:

```xml
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<presets>
 <sensorPreset>
        <sensor>VOLTAGE_MEASUREMENT</sensor>
        <nominalValue>8.0</nominalValue>
        <maxNegDeviation>1.0</maxNegDeviation>
        <maxPosDeviation>1.0</maxPosDeviation>
 </sensorPreset>
 <sensorPreset>
        <sensor>VOLTAGE_SPEEDCTR</sensor>
        <nominalValue>0.52</nominalValue>
        <maxNegDeviation>2.5</maxNegDeviation>
        <maxPosDeviation>2.5</maxPosDeviation>
 </sensorPreset>
 <sensorPreset>
        <sensor>ULTRASONIC_FRONT_LEFT</sensor>
        <nominalValue>0.52</nominalValue>
        <maxNegDeviation>2.5</maxNegDeviation>
        <maxPosDeviation>2.5</maxPosDeviation>
 </sensorPreset>
 <sensorPreset>
        <sensor>ULTRASONIC_FRONT_CENTER_LEFT</sensor>
        <nominalValue>0.52</nominalValue>
        <maxNegDeviation>2.5</maxNegDeviation>
        <maxPosDeviation>2.5</maxPosDeviation>
 </sensorPreset>
 <sensorPreset>
        <sensor>ULTRASONIC_FRONT_CENTER</sensor>
        <nominalValue>0.52</nominalValue>
        <maxNegDeviation>2.5</maxNegDeviation>
        <maxPosDeviation>2.5</maxPosDeviation>
 </sensorPreset>
 <sensorPreset>
        <sensor>ULTRASONIC_FRONT_CENTER_RIGHT</sensor>
        <nominalValue>0.52</nominalValue>
```

```
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>ULTRASONIC_FRONT_RIGHT</sensor>
                <nominalValue>0.52</nominalValue>
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>ULTRASONIC_SIDE_LEFT</sensor>
                <nominalValue>0.52</nominalValue>
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>ULTRASONIC_SIDE_RIGHT</sensor>
                <nominalValue>0.52</nominalValue>
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>ULTRASONIC_REAR_LEFT</sensor>
                <nominalValue>0.52</nominalValue>
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>ULTRASONIC_REAR_CENTER</sensor>
                <nominalValue>0.52</nominalValue>
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>ULTRASONIC_REAR_RIGHT</sensor>
                <nominalValue>10.52</nominalValue>
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>GYROSCOPE_X_ACC</sensor>
                <nominalValue>0.52</nominalValue>
```

```
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>GYROSCOPE_Y_ACC</sensor>
                <nominalValue>0.52</nominalValue>
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>GYROSCOPE_Z_ACC</sensor>
                <nominalValue>0.52</nominalValue>
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>GYROSCOPE_YAW</sensor>
                <nominalValue>0.52</nominalValue>
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>GYROSCOPE_PITCH</sensor>
                <nominalValue>0.52</nominalValue>
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
        <sensorPreset>
                <sensor>GYROSCOPE_ROLL</sensor>
                <nominalValue>0.52</nominalValue>
                <maxNegDeviation>2.5</maxNegDeviation>
                <maxPosDeviation>2.5</maxPosDeviation>
        </sensorPreset>
    </presets>
```
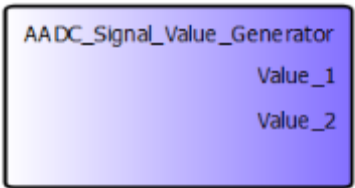
In the Property *Directory for Sensorpresets* a folder can be specified where the filter should look for multiple preset files. The recognized files in the folder can be selected afterwards in the GUI with the dropdown menu at the top of the window.

	Pinname	**MediaDescription**	**Notes**
Inputs	Ultrasonic_Front_Left	tSignalValue	Value in m

	Ultrasonic_Front_Center_Left	tSignalValue	Value in m
	Ultrasonic_Front_Center	tSignalValue	Value in m
	Ultrasonic_Front_Center_Right	tSignalValue	Value in m
	Ultrasonic_Front_Right	tSignalValue	Value in m
	Ultrasonic_Side_Left	tSignalValue	Value in m
	Ultrasonic_Side_Right	tSignalValue	Value in m
	Ultrasonic_Rear_Left	tSignalValue	Value in m
	Ultrasonic_Rear_Center	tSignalValue	Value in m
	Ultrasonic_Rear_Right	tSignalValue	Value in m
	Voltage_Measurement	tSignalValue	Value in V
	Voltage_SpeedCntrl	tSignalValue	Value in V
	InerMeasUnit_Struct	tInerMeasUnitData	Struct with IMU Data
	WheelLeft_Struct	tWheelData	Struct with Data from left wheel
	WheelRight_Struct	tWheelData	Struct with Data from right wheel
	Name	**Description**	**Notes**
Property	Directory for Sensorpresets	Here you have to select the folder which contains the sensor preset files	

3.4.4.13   Signal Value Generator

Category:	OBJCAT_Auxiliary
Version:	1.0.0
Filename:	aadc_signalValueGenerator.plb
Source:	src\aadcDemo\src\helper\AADC_SignalValueGenerator
Guid:	aadc_signalValueGenerator

This filter can be used to generate two waveforms in Media Samples of the type tSignalValue.

The waveforms can be chosen in table of the GUI of the filter by specifying a timestamp and two corresponding values which are transmitted on the output pins. The filter does a linear interpolation between the given points. The sample rate of the output samples must be set in the property *Actuator Update Rate [Hz]*.

The filter is further able to work with a default file containing predefined values. This file can be selected in the properties and the values can be loaded with the button *Load Defaults.* After editing values the button *Save* can be used to store the values into a new file or back into to the file with the default values used in the properties.

The values in the properties *Default Value 1* and *Default Value 2* are always transmitted except when the defined waveform is transmitted, i.e. before the waveform starts and after the waveform ends.

The file must be a simple file containing in one line first the timestamp, followed by the value 1 and value 2. A file looks like:

```
1000 0 80
8000 3 110
15000 3 90
```
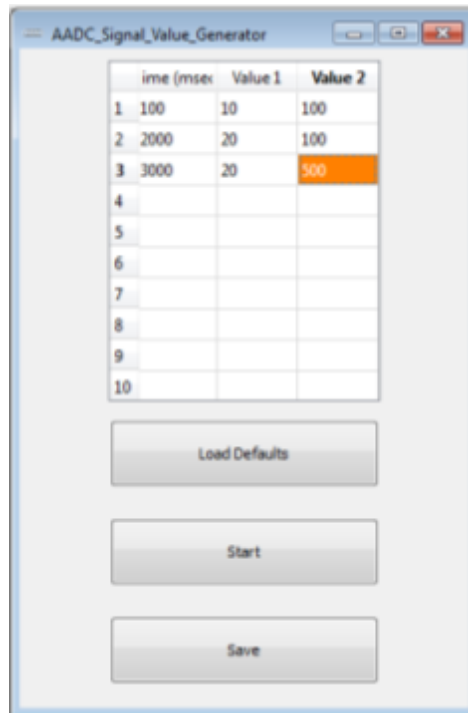

Figure 26 GUI Signal Value Generator

	Pinname	MediaDescription	Notes
Outputs	Value_1	tSignalValue	
	Value_2	tSignalValue	

	Name	Description	Notes
Property	Actuator Update Rate [Hz]	Defines how much updates for steering and speed controller are sent in one second (Range: 0 to 100 Hz)")	Default: 30 Range: 0…100
	Default Value 1	Defines the default value which is transmitted when nothing is read from the list	Default: 0
	Default Value 2	Defines the default value which is transmitted when nothing is read from the list	Default: 0
	File for Default Values	Set the file with the default files here	

3.4.4.14   State Controller

Category:	OBJCAT_Tool
Version:	1.1.0
Filename:	aadc_stateController.plb
Source:	src\aadcDemo\src\helper\AADC_StateController\
Guid:	aadc_stateController

```
AADC_State_Controller
_ Jury_Struct          Driver_Struct  _
_ Set_State_Ready
_ Set_State_Running
_ Set_State_Complete
_ Set_State_Error
_ Increment_Maneuver_ID
_ Restart_Section
_ Maneuver_List
```

This filter implements a state machine that defines the several possible states of the vehicle and manages the communication with the Jury module. It can be used by the teams as reference to build their own individual state machine to communicate and respond correctly with the Jury Module.

This filter implements the following states which are defined in *juryEnums.h*

```
enum stateCar{
    stateCar_ERROR=-1,
    stateCar_READY=0,
    stateCar_RUNNING=1,
    stateCar_COMPLETE=2,
    stateCar_STARTUP=-2
};
```

The Jury uses the following actions to control the vehicle:

```
enum juryActions{
    action_STOP=-1,
    action_GETREADY=0,
    action_START=1
};
```

The vehicle transmits the following states to the jury:

```
enum stateCar{
    stateCar_ERROR=-1,
    stateCar_READY=0,
    stateCar_RUNNING=1,
    stateCar_COMPLETE=2,
    stateCar_STARTUP=-2
};
```

The received actions from the jury or the receiving of a sample on the pins *Set_State_xxx* forces the state machine to change its state. The following state changes are implemented:
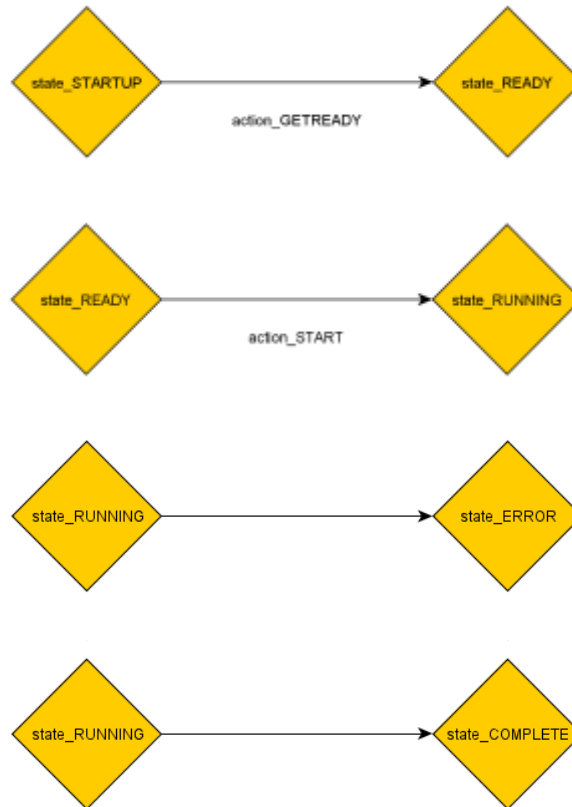
After receiving an action_STOP the vehicle has to change to a state from which it can be activated again. Possible would be change to state_STARTUP
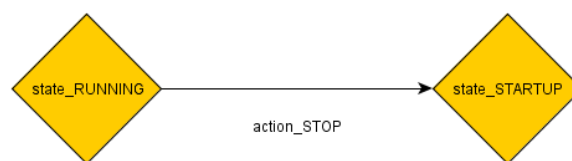
As already mentioned this filter can be used in two different modes:

- The filter automatically reacts on incoming Jury Structs on the inputpin *Jury_Struct*
- The filter changes its state depending on inputs on the inputpins Set_State_Ready, Set_State_Running, Set_State_Complete, Set_State_Error

Mode 1: Jury Structs:

Started, the vehicle is in state state_STARTUP. In this state no DriverStructs are generated as output and the vehicle is waiting for the action action_GETREADY from the Jury. When such an action_GETREADY is received the state will change to state_READY and start with cyclic sending of DriverStruct with the state state_READY. This Media Sample contains the current ManeuverID.

Note that after first changing to state_READY it starts with ID 0.

When the filter receives now an action_START command the filter changes its state to state_RUNNING and sends this state periodically to the Jury Module. At this state change the filter verifies if the ManeuverID of the state_READY command and the action_START command are the same. If it is unequal a warning is emitted but the state is also changed.

If a sample of type tBoolSignalValue with a TRUE in the bValue element is received on the input pin Increment_Maneuver_ID, while being in state state_RUNNING, the current ManeuverID is incremented.

If the filter receives now an action_STOP command it changes its state to state_STARTUP and stops transmitting states to the jury.

Mode 2: Using the Set_State_xxx pins

Media Samples of type tBoolSignalValue have to be transmitted to the pins Set_State_Ready, Set_State_Running, Set_State_Complete, Set_State_Error, Increment_Maneuver_ID and Restart_Section. If the sample contains a TRUE in the bValue element the filter reacts as the following list shows:
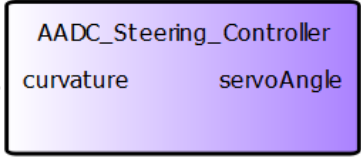
- Set_State_Ready: change to state_READY
- Set_State_Running: change to state_RUNNING
- Set_State_Complete: change to state_COMPLETE
- Set_State_Error: change to state_ERROR
- Increment_Maneuver_ID: increments the counter for the ManeuverID
- Restart_Section: resets the ManeuverID to the first ID in the current section.

Note: To perform the last two actions a ManeuverFile has to be set in the Properties.

The Media Samples with the tBoolSignalValues can be generated with the Filter Bool Value Generator.

	Pinname	MediaDescription	Notes
Inputs	Jury_Struct	tJuryStruct	
	Set_State_Ready	tBoolSignalValue	
	Set_State_Running	tBoolSignalValue	
	Set_State_Complete	tBoolSignalValue	
	Set_State_Error	tBoolSignalValue	
	Increment_Maneuver_ID	tBoolSignalValue	
	Restart_Section	tBoolSignalValue	
	Maneuver_List	tManeuverList	Maneuver list from Jury Module
Outputs	Driver_Struct	tDriverStruct	
	Name	**Description**	**Notes**
Property	Debug Output to Console	If enabled additional debug information is printed to the console (Warning: decreases performance)	Default: False Range:

3.4.4.15 Steering Controller

Category:	OBJCAT_DataFilter
Version:	1.0.0
Filename:	aadc_steeringController.plb
Source:	src\aadcDemo\src\algorithms\AADC_SteeringController
Guid:	aadc_steeringController

This filter can be used to set the steering servo with a curvature in meter. It loads a XML-file given in the properties and does a mapping of the given curvature to a servo angle based on the XML-file. The entries in the XML has to be found out by experiment and can be supported by the Steering Calibration Filter.

A sample file was generated before and can be used for beginning. The sample file *SteeringController.xml* is located in the folder *configuration_files.*
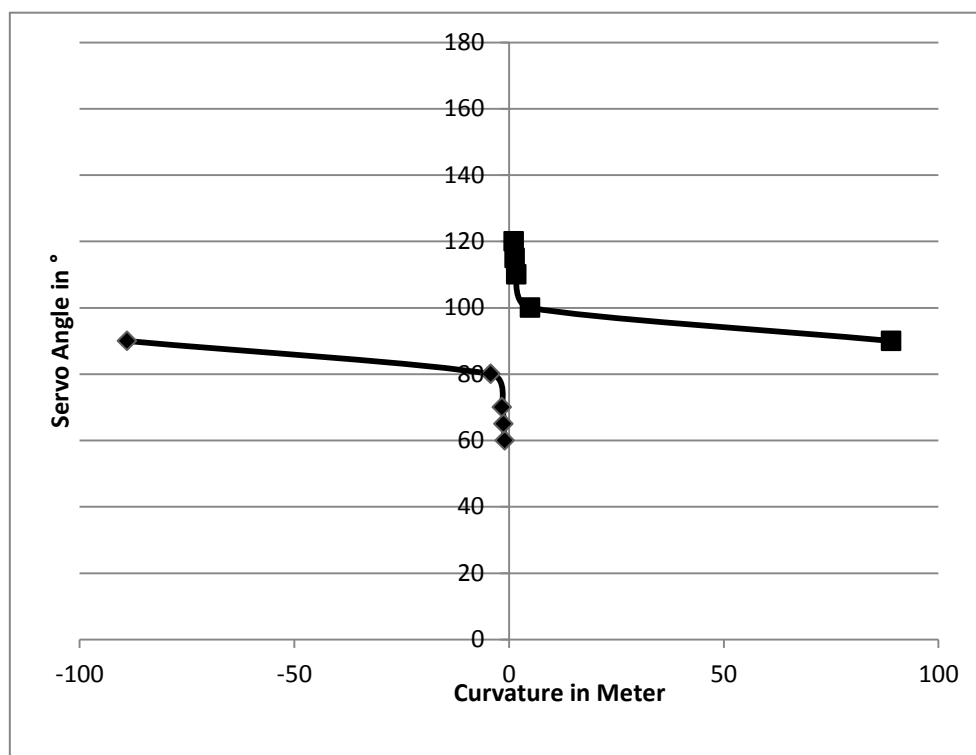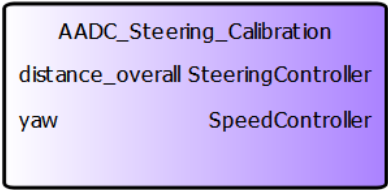

Figure 27 Mapping from Curvature to Servo Angle

The normal Calibration XML Filter cannot be used for this mapping because there is inconstancy about the point of 90° in the graph.

	Pinname	MediaDescription	Notes

Inputs	curvature	tSignalValue	curvature in meter
Outputs	servoAngle	tSignalValue	servo angle in °
	Name	**Description**	**Notes**
Property	Debug Mode	If true debug infos are plotted to console	Default: False
	Configuration File For Mapping	The XML to be loaded has to be set here	

3.4.4.16 Steering Calibration Filter

AADC_Steering_Calibration distance_overall SteeringController yaw SpeedController	**Category:**	OBJCAT_DataFilter
	Version:	1.0.0
	Filename:	aadc_steeringCalibration
	Source:	src\aadcDemo\src\datafilter\AADC_SteeringCalibration\cSteeringCalibrationFilter.h
	Guid:	aadc_steeringCalibration

This filter can be used to obtain a calibration table for the Steering Controller which can be used to steer the car with given curvature and not only by setting the servo angle of the steering controller. The servo angle is an abstract value and does not describe the car behavior explicitly. The curvature is the more useable to control the car on the street.
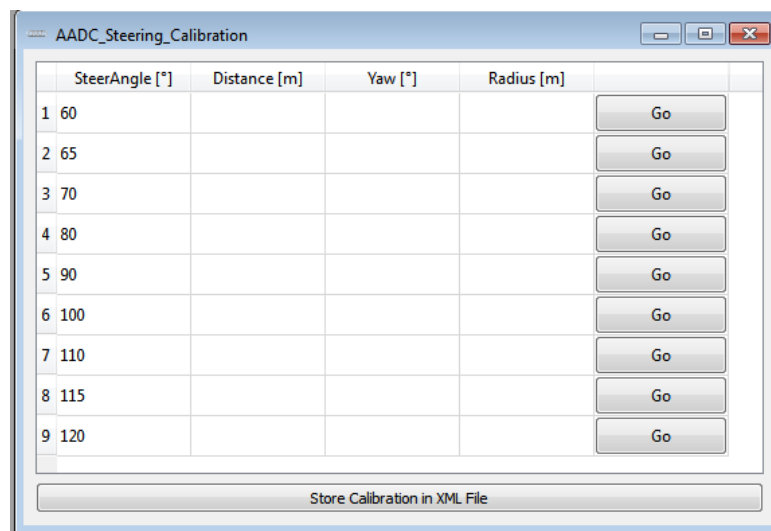
Figure 28 GUI Steering Controller

The calibration is done by series of driven curvature defined by different servo steering angles as the GUI shows. If the button *Go* is clicked the car starts to drive a curvature with the servo steering angle in the line of the table. If the maximum arc distance or the maximum angle from the properties is reached the filter calculates the radius of the driven curvature by using the driven distance and the yaw angle obtained by the IMU. The calculated radius is shown in the

column Radius.

With the button *Store Calibration in XML File* the results can be saved to an XML File.

The output pin SpeedController must not be connected to the Arduino Actuators only by using the Wheel Speed Controller.

	Pinname	MediaDescription	Notes
Inputs	distance_overall	tSignalValue	from Converter Wheels
	yaw	tSignalValue	from Converter IMU
Outputs	SteeringController	tSignalValue	to Arduino Actuators (Servo Angle in °)
	SpeedController	tSignalValue	to Wheel Speed Controller (in $\frac{m}{sec^2}$)
	Name	**Description**	**Notes**
Property	max arc distance [m]	The maximum arc distance driven in calibration	Default: 5.0 Range: 1...20
	max angle [°]	The maximum angle driven in calibration	Default: 170.0 Range: 10..170
	average speed [m/s]	The average speed during calibration procedure.	Default: 1

3.4.4.17   Visualization

Category:	Application
Version:	1.0.0
Filename:	aadc_visualization.plb
Source:	src\adtfBase\src\helper\AADC_Visualization
Guid:	adtf.aadc.aadc_visualization

AADC_Visualization
Ultrasonic_Front_Left
Ultrasonic_Front_Center_Left
Ultrasonic_Front_Center
Ultrasonic_Front_Center_Right
Ultrasonic_Front_Right
Ultrasonic_Side_Left
Ultrasonic_Side_Right
Ultrasonic_Rear_Left
Ultrasonic_Rear_Center
Ultrasonic_Rear_Right
Voltage_Measurement
Voltage_SpeedCntrl
InerMeasUnit_Struct
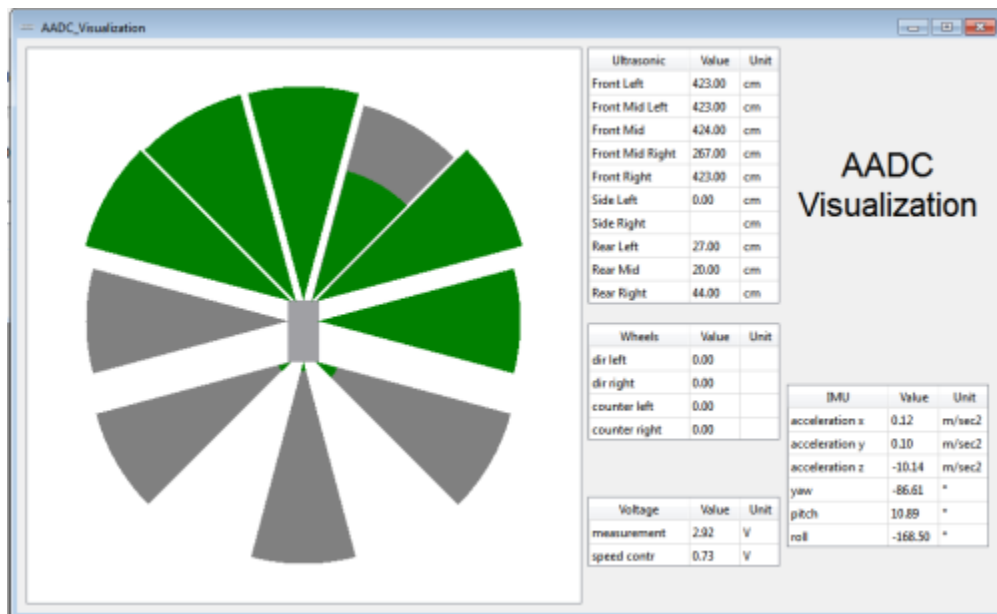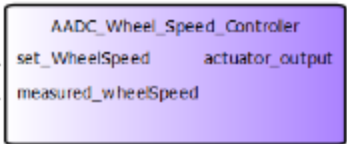WheelLeft_Struct
WheelRight_Struct

Figure 11 GUI Visualization Filter

This filter opens a QT window, which visualizes most of the sensor data. Thereon several tables are shown including the data from the ultrasonic sensors, from the wheel speed sensors, from the voltage sensors and from the inertial measurement sensor (IMU). On the left side of the windows is also a graphical scene illustrating the values from the ultrasonic sensors.

	Pinname	MediaDescription	Notes
Inputs	Ultrasonic_Front_Left	tSignalValue	Value in m

Ultrasonic_Front_Center_Left	tSignalValue	Value in m
Ultrasonic_Front_Center	tSignalValue	Value in m
Ultrasonic_Front_Center_Right	tSignalValue	Value in m
Ultrasonic_Front_Right	tSignalValue	Value in m
Ultrasonic_Side_Left	tSignalValue	Value in m
Ultrasonic_Side_Right	tSignalValue	Value in m
Ultrasonic_Rear_Left	tSignalValue	Value in m
Ultrasonic_Rear_Center	tSignalValue	Value in m
Ultrasonic_Rear_Right	tSignalValue	Value in m
Voltage_Measurement	tSignalValue	Value in V
Voltage_SpeedCntrl	tSignalValue	Value in V
InerMeasUnit_Struct	tInerMeasUnitData	Struct with IMU Data
WheelLeft_Struct	tWheelData	Struct with Data from left wheel
WheelRight_Struct	tWheelData	Struct with Data from right wheel

3.4.4.18   Wheel Speed Controller

Category:	OBJCAT_DataFilter
Version:	1.0.0
Filename:	aadc_wheelSpeedController.plb
Source:	src\aadcDemo\src\algorithms\AADC_WheelSpeedController\
Guid:	aadc_wheelSpeedController

This filter implements a controller to set the wheelspeed of the vehicle with a P/PI/PID or PT1 algorithm. The input pin *measured_wheelSpeed* has to be connected to the output pin of the Converter Wheels Filter and the desired wheel speed has to be set to the input pin set_WheelSpeed.

The controller parameters have to be adapted to each individual car. The default values for the PT1 controller are a good start for the controller, but maybe they have to be adapted to the individual team car. There are different methods to get the correct parameter, please refer to other literature.

The output pin actuator_output have to be connected to a Calibration XML which maps the speed in $\frac{m}{sec^2}$ to servo angle of steering controller. In experiments the following mapping was found:
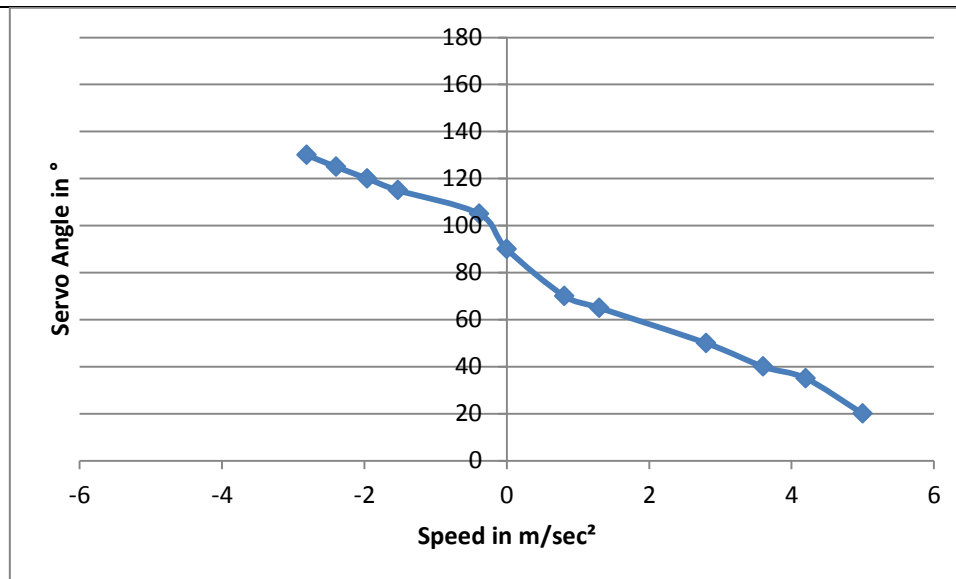
Figure 29 Mapping Speed to Servo Angle

This values are saved in the Sample XML *SpeedController.xml* in the folder *configuration_files* and can be used at the beginning.

A typical configuration with the Wheel Speed Controller should contain at least the following Filters:
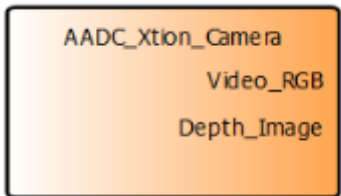

Figure 30 Configuration Wheel Speed Controller

	Pinname	MediaDescription	Notes
Inputs	set_WheelSpeed	tSignalValue	in $\frac{m}{sec^2}$
	measured_wheelSpeed	tSignalValue	in $\frac{m}{sec^2}$

Outputs	actuator_output	tSignalValue	in $\frac{m}{sec^2}$
	Name	**Description**	**Notes**
Property	PT1::TimeConstant	Time Constant for PT1 Controller	Default: 1.5
	PT1::Gain	Gain for PT1 Controller	Default: 6
	PT1::OutputFactor	The factor to normalize the output value	Default: 1
	PT1::Correction Factor	Correction factor for input set point	Default: 1.15
	PID::Kp_value	The proportional factor Kp for the PID Controller	Default: 1
	PID::Ki_value	The integral factor Ki for the PID Controller	Default: 1
	PID::Kd_value	The differential factor Kd for the PID Controller	Default: 1
	PID:Sample_Interval[msec]	The sample interval in msec used by the PID controller	Default: 1
	Debug Mode	If true debug infos are plotted to registry	Default: False

3.4.4.19   Xtion Camera

Category:	OBJCAT_CameraDevice
Version:	1.0.0.
Filename:	aadc_xtionCamera.plb
Source:	src\aadcDemo\src\xtion\AADC_XtionCamera
Guid:	aadc_xtionCamera

This filter grabs the video streams from the xtion camera by using the openni library. The RGB video stream is transmitted on the output pin Video_RGB and the depth image stream is transmitted on the pin Depth_Image.

All the settings are managed with the configuration file set in the property *Configuration File.* The file is as the following:

```xml
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<xtionSettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <depthImage>
        <x_resolution>320</x_resolution>
        <y_resolution>240</y_resolution>
        <fps>30</fps>
 </depthImage>
 <colorImage>
        <x_resolution>640</x_resolution>
        <y_resolution>480</y_resolution>
        <fps>30</fps>
 </colorImage>
 <options>
        <setDepthColorSync>0</setDepthColorSync>
        <setRegistration>0</setRegistration>
        <setDepthInMillimeter>0</setDepthInMillimeter>
        <setAutoExposure>0</setAutoExposure>
        <setAutoWhiteBalance>1</setAutoWhiteBalance>
 </options>
</xtionSettings>
```

A Sample file *xtionSettings.xml* is located in the folder *configuration_files.*

The parameter in „setDepthInMillimeter" sets the scale of the values in the depth image. If set to "0" the format of pixel is directly passed from OpenNi, if set to "1" the flag in OpenNi is set to "openni::PIXEL_FORMAT_DEPTH_1_MM" so the unit of the pixels is millimeter.

	Pinname	MediaDescription	Notes
Outputs	Video_RGB	Video Pin	
	Depth_Image	Video Pin	
	Name	**Description**	**Notes**
Property	Configuration File	The configuration file for the xtion device	

3.4.5 ADTF Projects

3.4.5.1 Base Configuration

The ADTF Base Configuration manages all the communication with the Arduinos. It includes four Arduino Communication Filters, one Arduino Actuators Filter, one Arduino Sensors Filter and the Jury Config as a Subconfiguration. All the sensors values from the Arduinos are available on the output ports at the right side and all the actuator values to the Arduinos are received on the input ports on the left side.

The incoming driver structs from the user are transmitted to the Jury Configuration and the received Jury Structs from the Jury Configuration are transmitted back to the Configuration, which includes this Base Configuration.
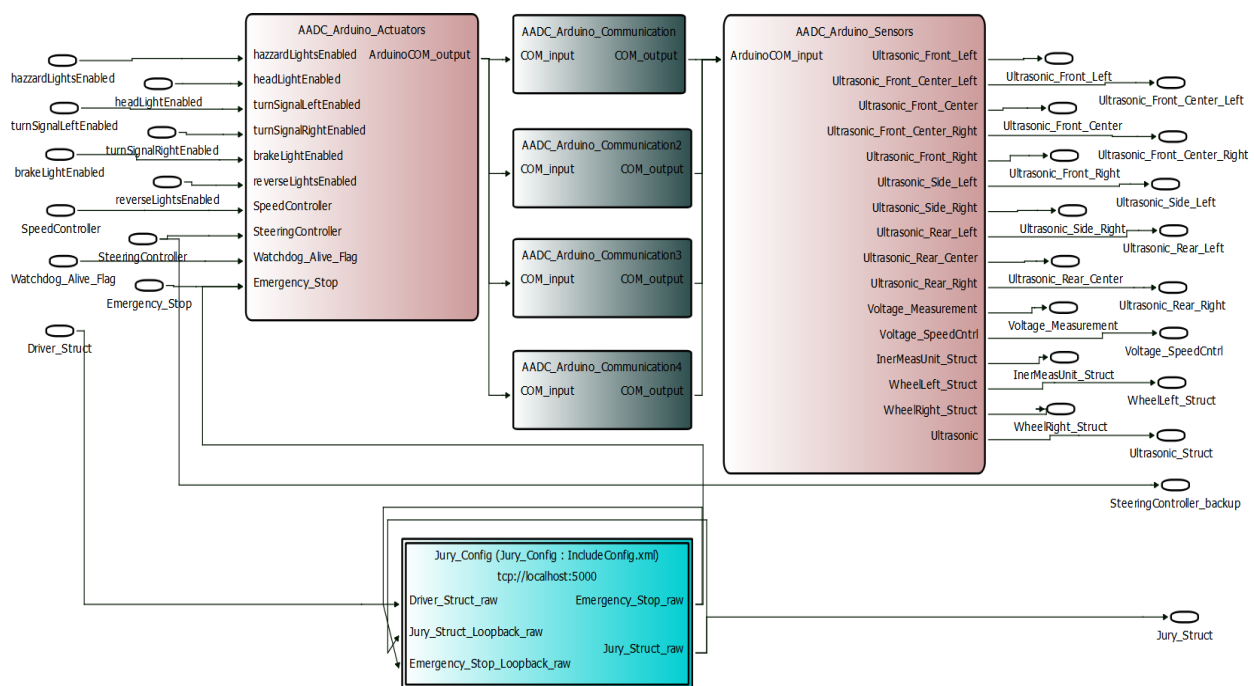

Figure 31 Layout Base Configuration

This Base Configuration must be included by the teams in their own configuration. It is located at \home\aadc\AADC\config\BaseConfig\BaseConfig\config\system.xml .

3.4.5.2 Jury Configuration

The Jury Configuration is running on the Jury ADTF PC and is controlled by the Jury. To test the individual team configuration they can use the delivered Jury Module which transmits the Jury Struct to control the car and also visualizes the received DriverStructs.
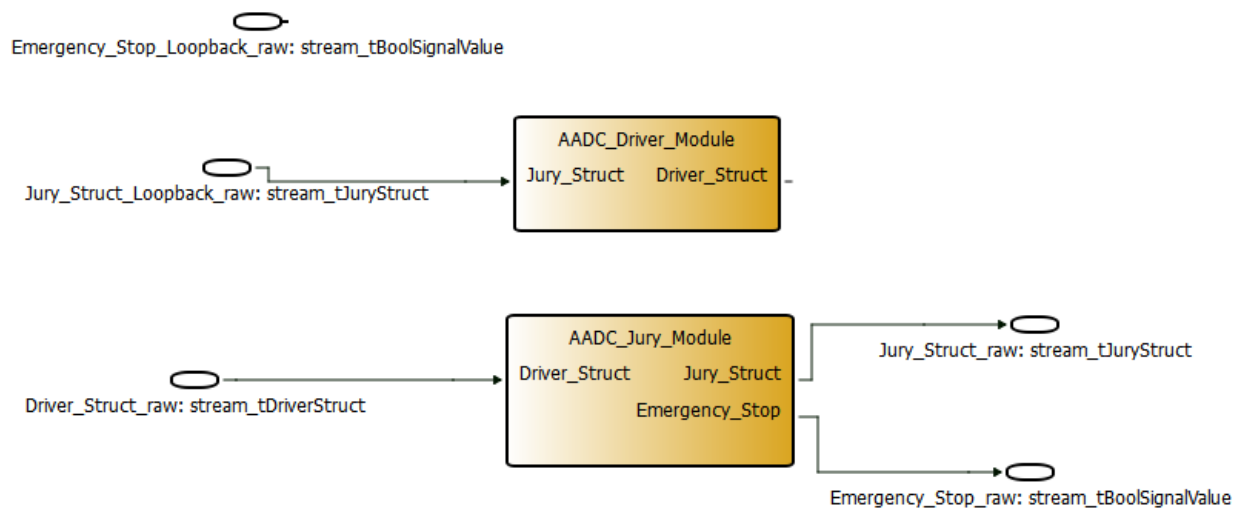
Figure 32 Jury Configuration for developing

The Jury Configuration is located at \home\aadc\AADC\config\JuryConfig\JuryConfig.prj .

3.4.5.3   User Configuration

The User Configuration should be used by the teams as a template to setup their own projection. The User Configuration includes the base configuration and provides all the sensor values and an interface for the actuator commands. The User Configuration is also set as a template in ADTF to be used when creating a new project.

The Example Configuration can also be referred to understand how to build a new configuration.

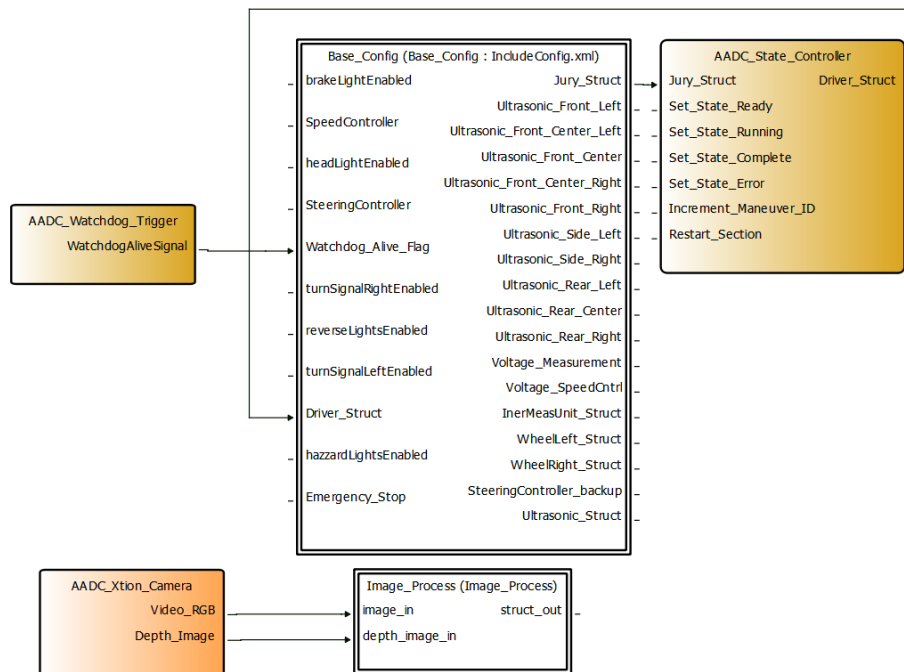The User Configuration is located at \home\aadc\AADC\config\UserConfig\UserConfig.prj .

Figure 33 User Configuration

3.4.5.4　Live Visualization Configuration

The Configuration LiveVisualizationConfig includes a more extended Configuration based on the User Configuration.

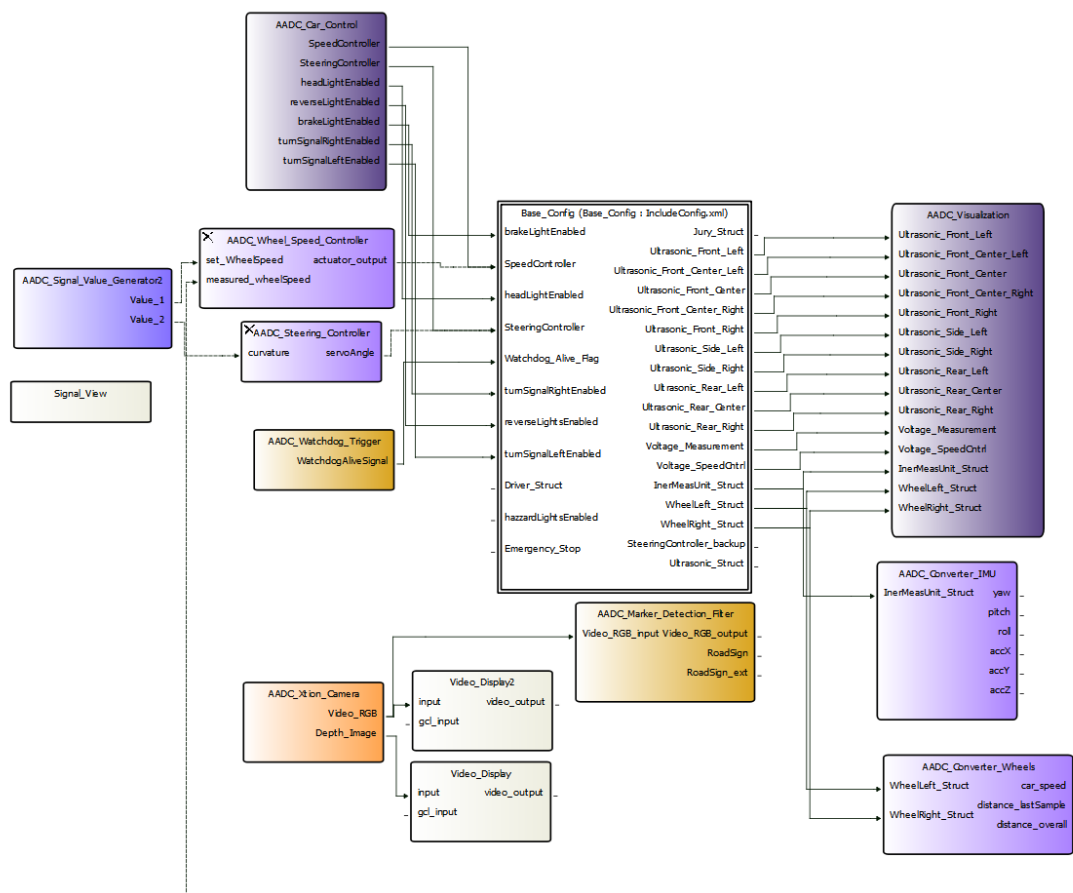The LiveVisualizationConfig is located at _\home\aadc\AADC\config\LiveVisualizationConfig_ _LiveVisualizationConfig.prj_ .

Figure 34 LiveVisualizationConfig

3.4.5.5 Camera Calibration Configuration

This configuration can be used for calibrating the Xtion camera.

3.4.5.6 Record and Playback Configuration

This configuration can be used for recording the data from the Xtion and from the Arduinos and an appropriate configuration to play the recorded data.

3.4.5.7 Speed Controller Calibration Configuration

This configuration can be used for calibrating the Speed Controller as described in Chapter 6.1.

3.4.6 Jury Module Application

NOTE: This standalone application replaces the deprecated Jury Module Filter. It is located for Linux 64 Bit, Windows 32 Bit and 64 Bit in the bin folder:

- bin/Jury_Module_Linux64
- bin/Jury_Module_win64
- bin/Jury_Module_win32

The application sends samples to a driver module which has to be implemented by the teams but can be based on the Demo filters. The sent samples contain the tJuryStruct and are used to check or set the state of the cars. The struct is defined as followed:

```
typedef struct
{
    tInt8 i8ActionID;
    tInt16 i16ManeuverEntry;
} tJuryStruct;
```

The i8ActionID contain one of the following actions:
- action_isReady: The vehicle is asked if it is ready to start the maneuver given in i16ManeuverEntry.
- action_start: After receiving this sample the car should start the maneuver given in i16ManeuverEntry. Before that the sample the same maneuver is asked if it is ready.
- action_stop: After receiving this sample the car should stop the maneuver given in i16ManeuverEntry. If the car is in a different maneuver it should be stopped as well.

The last received state from the driver is shown in the middle of the GUI as well as on the console.

The use this application you have to load the most recent Media Description File first if it was modified in the meantime. Otherwise the internal predefined settings are used.

Next the Maneuver File has to be loaded with the button next to the text box for the maneuver file. After loading the file the IP Address of the car has to be set in the field for IP – Address and by clicking on *Connect* the connection can be established.

The actions of the Jury are performed with the buttons in the frame Maneuver Control.

- Maneuverlist: Sends the Maneuverlist to the car.
- Request State: Asks the vehicle if it is ready to start.
- Start: Starts the maneuver of selected index in drop down field Maneuver Index
- Stop: Stops the maneuver

At the bottom of the GUI is the button to do an Emergency Stop of the car.
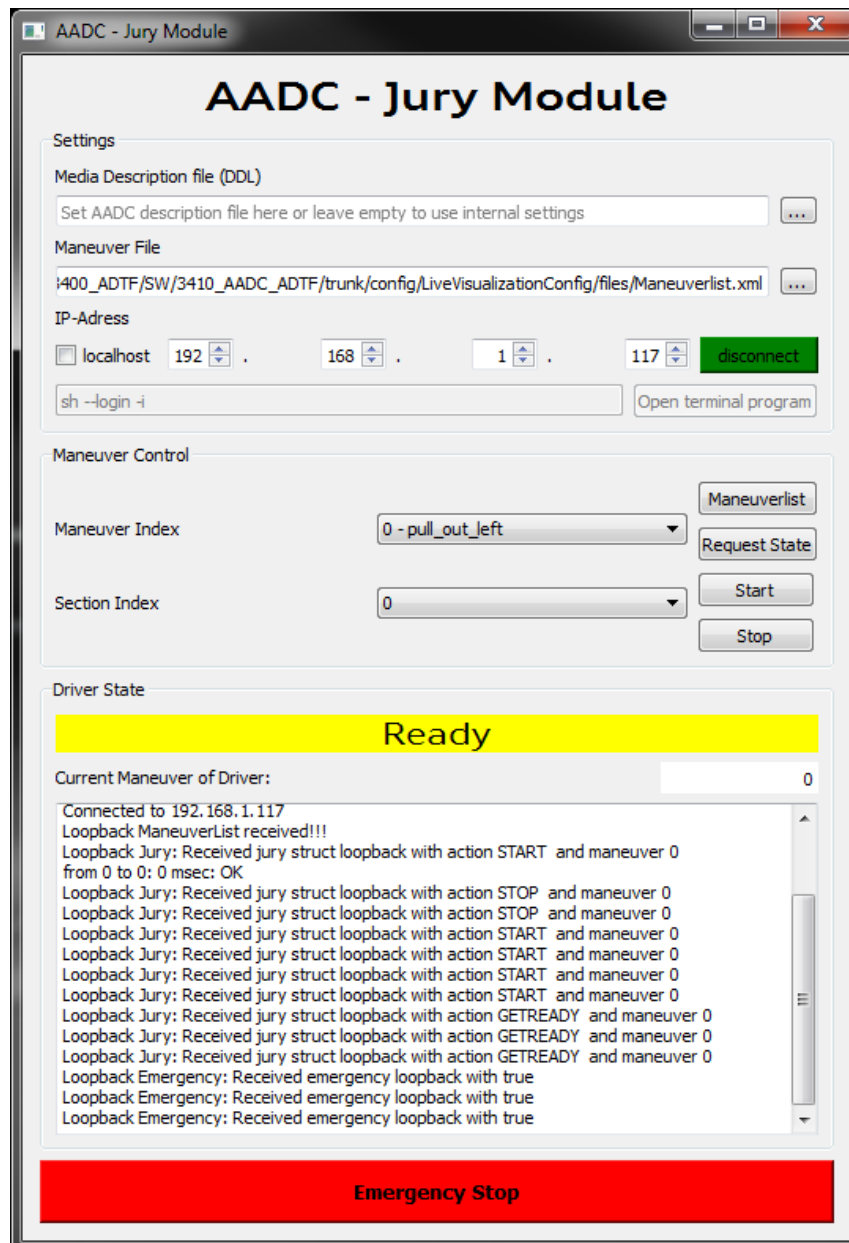
Figure 35 GUI Jury Application

3.4.7   ADTF DDL Descriptions

name	type	element	bytepos	arraysize
tArduinoData	tUInt8	ui8SOF	0	1
	tUInt8	ui8ID	1	1
	tUInt32	ui32ArduinoTimestamp	2	1
	tUInt8	ui8DataLength	6	1
	tUInt8	ui8Data	7	25
tJuryEmergencyStop	tBool	bEmergencyStop	0	1

tJuryStruct	tInt8	i8ActionID	0	1
	tInt16	i16ManeuverEntry	1	1
tDriverStruct	tInt8	i8StateID	0	1
	tInt16	i16ManeuverEntry	1	1
tSignalValue	tUInt32	ui32ArduinoTimestamp	0	1
	tFloat32	f32Value	4	1
tBoolSignalValue	tUInt32	ui32ArduinoTimestamp	0	1
	tBool	bValue	4	1
tWheelData	tUInt32	ui32ArduinoTimestamp	0	1
	tUInt32	ui32WheelTach	4	1
	tInt8	i8WheelDir	8	1
tInerMeasUnitData	tUInt32	ui32ArduinoTimestamp	0	1
	tFloat32	f32Q_w	4	1
	tFloat32	f32Q_x	8	1
	tFloat32	f32Q_y	12	1
	tFloat32	f32Q_z	16	1
	tFloat32	f32A_x	20	1
	tFloat32	f32A_y	24	1
	tFloat32	f32A_z	28	1
tUltrasonicStruct	tSignalValue	tFrontLeft	0	1
	tSignalValue	tFrontCenterLeft	8	1
	tSignalValue	tFrontCenter	16	1
	tSignalValue	tFrontCenterRight	24	1
	tSignalValue	tFrontRight	32	1
	tSignalValue	tSideLeft	40	1
	tSignalValue	tSideRight	48	1
	tSignalValue	tRearLeft	56	1
	tSignalValue	tRearCenter	64	1
	tSignalValue	tRearRight	72	1
tRoadSign	tFloat32	f32Imagesize	0	1
	tInt16	i16Identifier	4	1
tRoadSignExt	tInt16	i16Identifier	0	1
	tFloat32	f32Imagesize	2	1
	tFloat32	af32TVec	6	3
	tFloat32	af32RVec	18	3
tManeuverList	tInt32	i32Size	0	1
	tChar	aManeuverList	4	i32Size

4 Arduino

All the four Arduino Micros are connected to pITX with USB Cables and use the USB connection as a serial interface. The Serial Interface is implemented in the Arduino Communication Filter and uses the following protocol to communicate with the Arduinos.

4.1.1 Arduino Protocol

SENS_STEERING

0x55	0x11	uint32_t timeStamp	2	uint16_t ui16Angle	crc

SENS_WHEEL_RIGHT

0x55	0x21	uint32_t timeStamp	5	uint32_t ui32WheelTach	int8_t i8WheelDir	crc

SENS_WHEEL_LEFT

0x55	0x22	uint32_t timeStamp	5	uint32_t ui32WheelTach	int8_t i8WheelDir	crc

SENS_GYRO

0x55	0x31	uint32_t timeStamp	16	int16_t i16A_x	int16_t i16A_y	int16_t i16A_z	int16_t i16Q_w	int16_t i16Q_x	int16_t i16Q_y	int16_t i16Q_z	crc

SENS_US_FRONT_LEFT

0x55	0x41	uint32_t timeStamp	2	uint16_t ui16UsData	crc

SENS_US_FRONT_RIGHT

0x55	0x42	uint32_t timeStamp	2	uint16_t ui16UsData	crc

SENS_US_FRONT_CENTER

0x55	0x43	uint32_t timeStamp	2	uint16_t ui16UsData	crc

SENS_US_LEFT

0x55	0x44	uint32_t timeStamp	2	uint16_t ui16UsData	crc

SENS_US_RIGHT

0x55	0x45	uint32_t timeStamp	2	uint16_t ui16UsData	crc

SENS_US_BACK_LEFT

0x55	0x46	uint32_t timeStamp	2	uint16_t ui16UsData	crc

SENS_US_BACK_RIGHT

0x55	0x47	uint32_t timeStamp	2	uint16_t ui16UsData	crc

SENS_US_BACK_CENTER

0x55	0x48	uint32_t timeStamp	2	uint16_t ui16UsData	crc

SENS_VOLT_SPEEDCONT

0x55	0x61	uint32_t timeStamp	2	uint16_t volt_speed_cont	crc

SENS_VOLT_MEAS

0x55	0x62	uint32_t timeStamp	2	uint16_t volt_meas	crc

SENS_ERROR

0x55	0x00	uint32_t timeStamp	2	uint16_t error_nr	crc

ACT_WATCHDOG

0x55	0xA1	tUInt8 ui8IsTriggerd	crc

ACT_EMERGENCY_STOP

0x55	0xA2	tUInt8 ui8IsTriggerd	crc

ACT_STEER_SERVO

0x55	0xB1	uint8_t ui8Angle	crc

ACT_SPEED_CONTR

0x55	0xC1	uint8_t ui8Angle	crc

ACT_LIGHTS

0x55	0xD1	bitmask lightData	crc

5 Starting the Car

5.1 Connecting the batteries or the power supply

The car has two batteries one for power supply of the actuators, i.e. the speed controller and the steering servo, and one for power supply of the pITX Board, the Arduinos and the measurement technology.

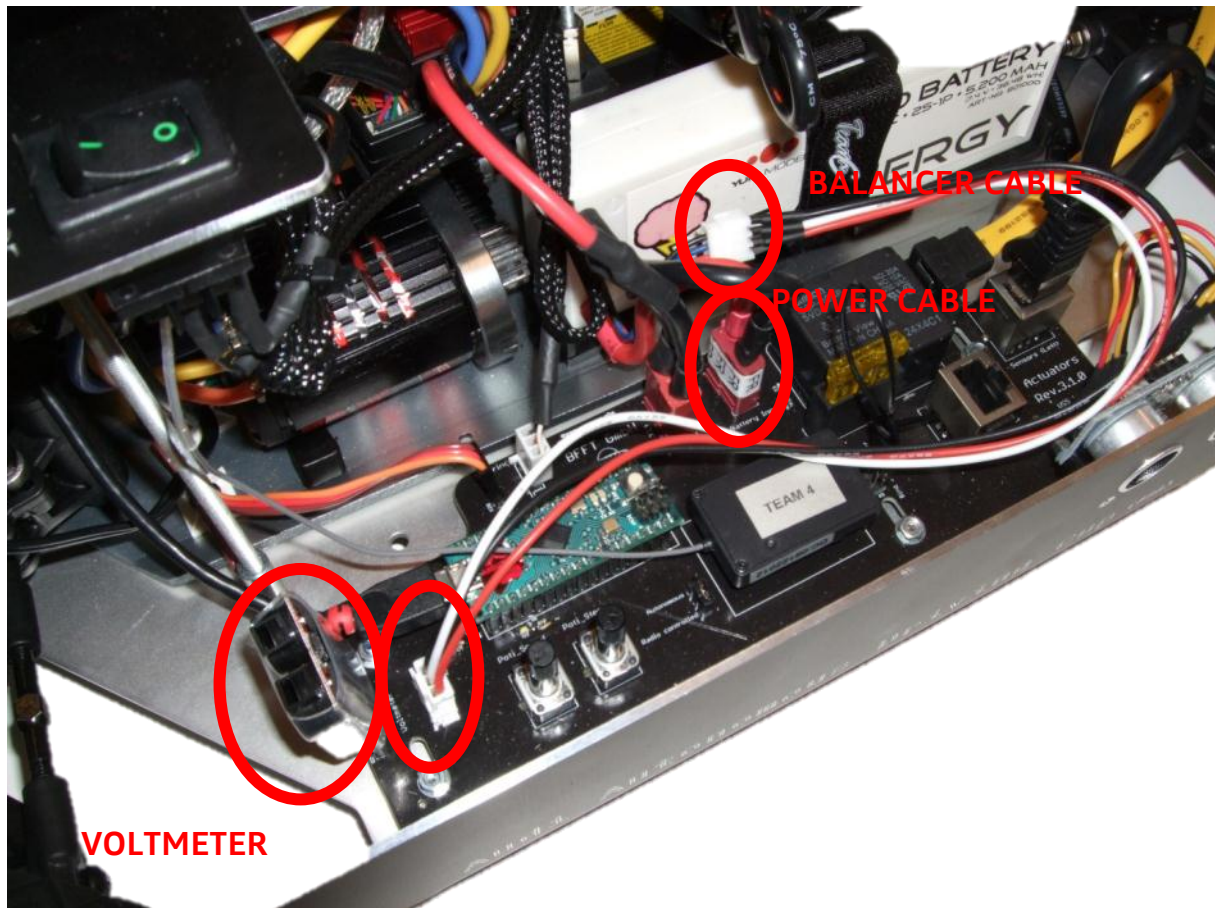These two batteries have to be charged and placed into the vehicle and connected to the board.

Figure 36 Connecting the motor battery

For connecting the battery make sure to connect the power cable (thick one with red and black cable) and the balancer cable (with three thin cables, black, white and red). Also make sure that the battery voltmeter is placed in the connector and after connecting the batteries it must show the voltage of the battery in big red letters.

THE BATTERY VOLTMETERS MUST BE USED DURING USING THE CAR. OTHERWISE THE BATTERIES COULD BE IRREVERSIBLE DAMAGED, OR COULD CATCH FIRE AND DAMAGE THE CAR. EVERYTIME THE CAR IS USED THE VOLTMETER MUST SHOW A VOLTAGE AND IF THEY START TO MAKE LOUD NOISE THE BATTERIES MUST BE CHARGED.
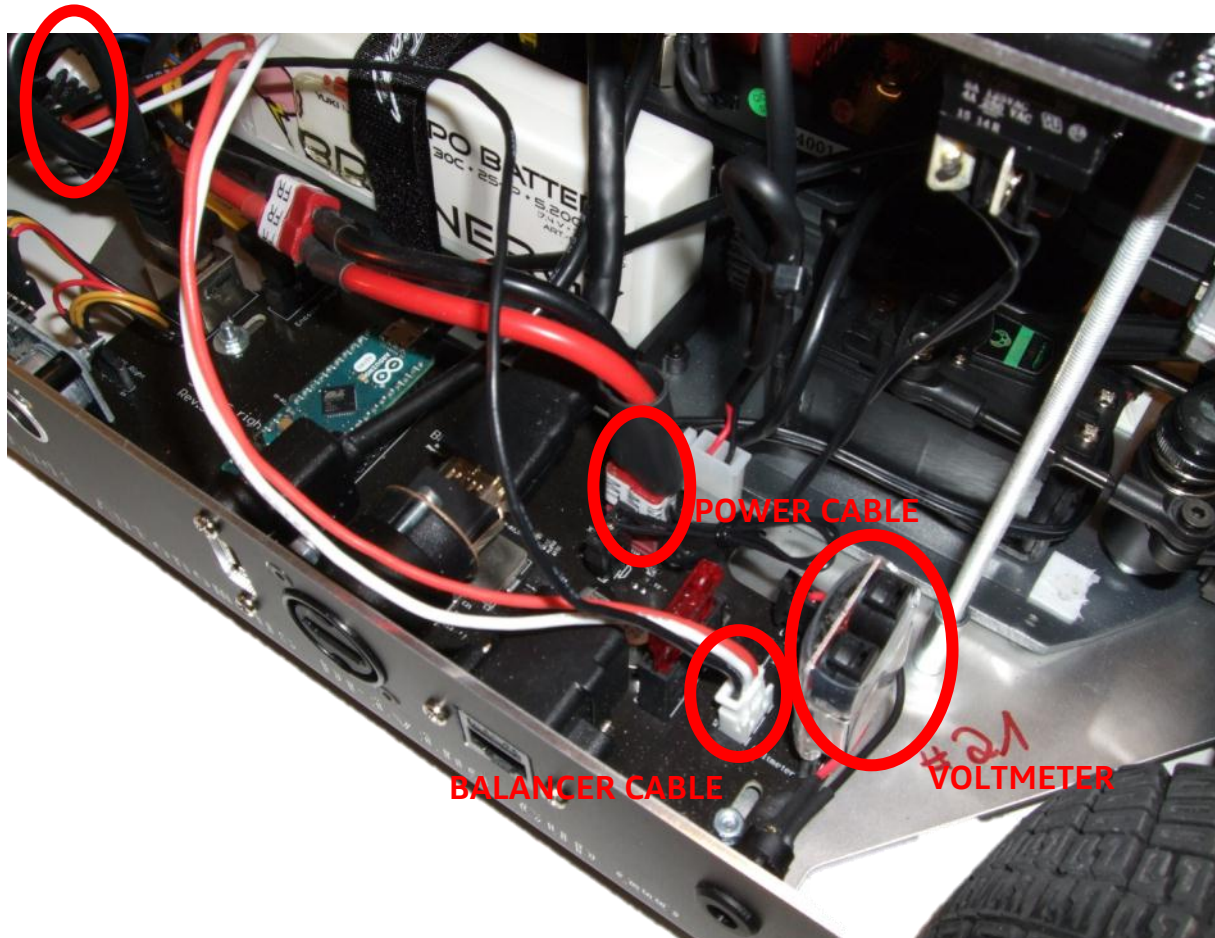
Figure 37 Connecting the sensor battery

The battery for power supply of the sensors has to be connected in the same way as the other one. Also make sure the balancer cable is connected and the voltmeter shows the voltage of the battery.

During developing procedure the external 12V power supply could be used instead of the sensor battery.

5.2 Switching on

SWITCH SENSOREN
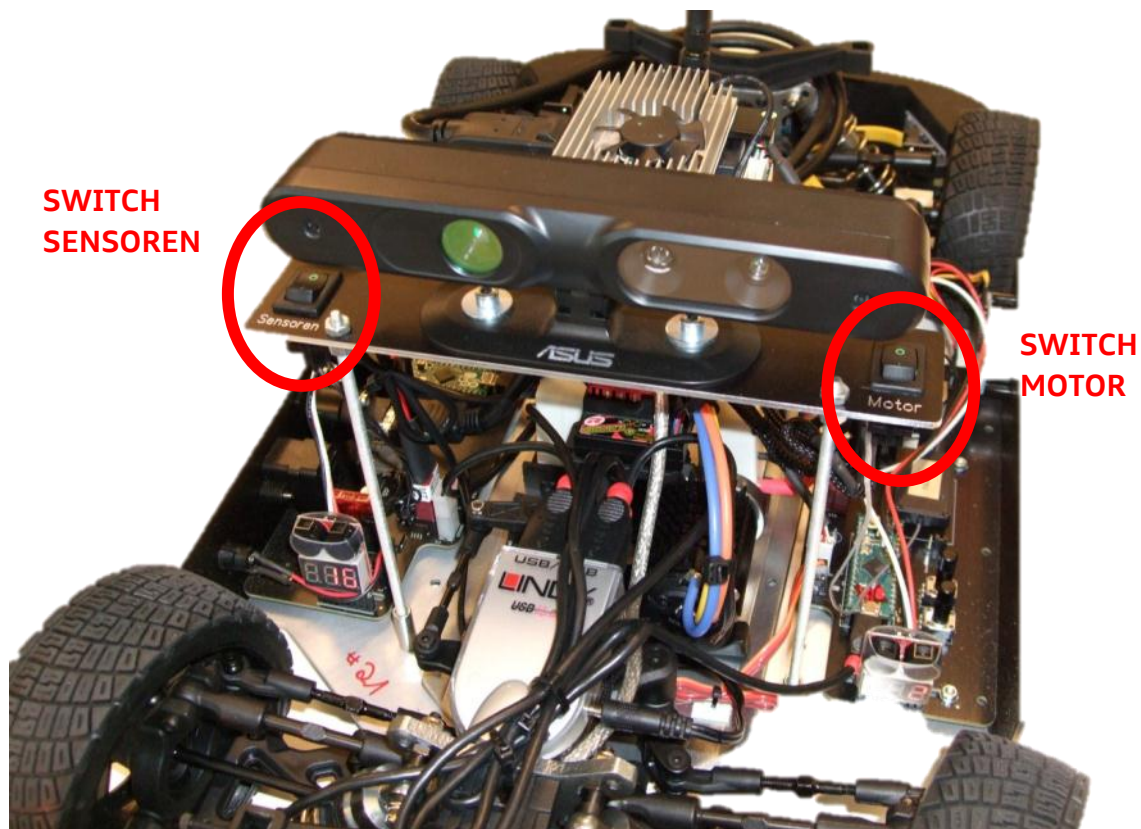
SWITCH MOTOR

Formel 1 Switches on the car

The car has two switches to start the car. The switch labeled *Sensoren* turns the pITX, the Arduinos and the measurement technology on. The other switch *Motor* turns the actuators on and need to be enabled if the car should move. In developing process it could be turned off to prevent unexpected driving of the car.

5.3 Connecting other devices

On the right side of the car are several plugs for other devices:

- USB Slot of pITX
- HDMI Video Socket of pITX
- Network Socket of pITCX
- Power Supply for car (12V with 4A must be put here, enclosed with delivery). This Power Supply could be used instead of the Sensor battery.

Figure 38 Connectors car

5.4 Use Remote Radio Controller

To use the controller described in Chapter 2.5.4 the remote receiver on the actuator board has to be enabled or disabled by some hardware changes.

5.4.1   Activate Remote Radio Controller

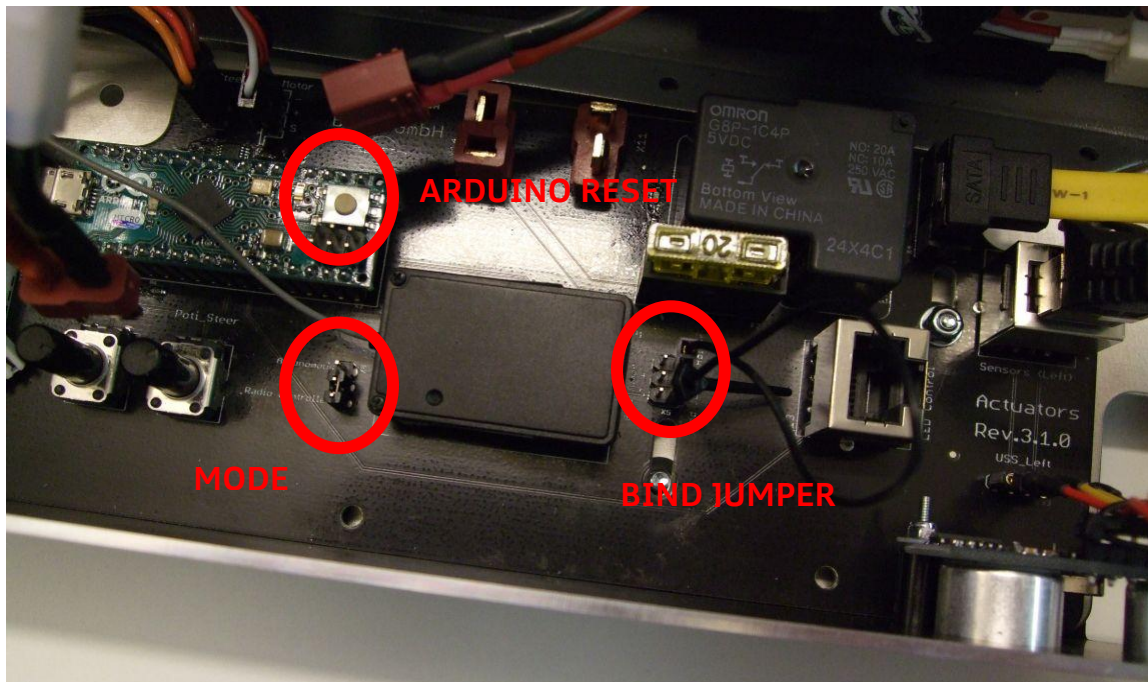Please follow the instructions below to use the radio controller.

Figure 39 Jumpers Radio Controller

1. First make the sure that the *Bind Jumper* is placed in the *Bind Jumper Holder* position and not in the *Bind Jumper* position (i.e. the position next the remote radio receiver).
2. Then the Mode Jumper has to be set to the *Radio controlled* position as shown in Figure 39.
3. After setting the jumpers correctly the Arduino on this board has to be reset by pressing the *Arduino Reset* button. After this reset the relais should toggle and the Speed Controller will initialize.

NOTE: When setting the Mode to *Radio Controlled* all the actuators commands from the Arduino and also from ADTF are ignored.

NOTE: The zero position of the Radio Controller System maybe has to be adjusted. Take care of the last point in Chapter 5.4.3.
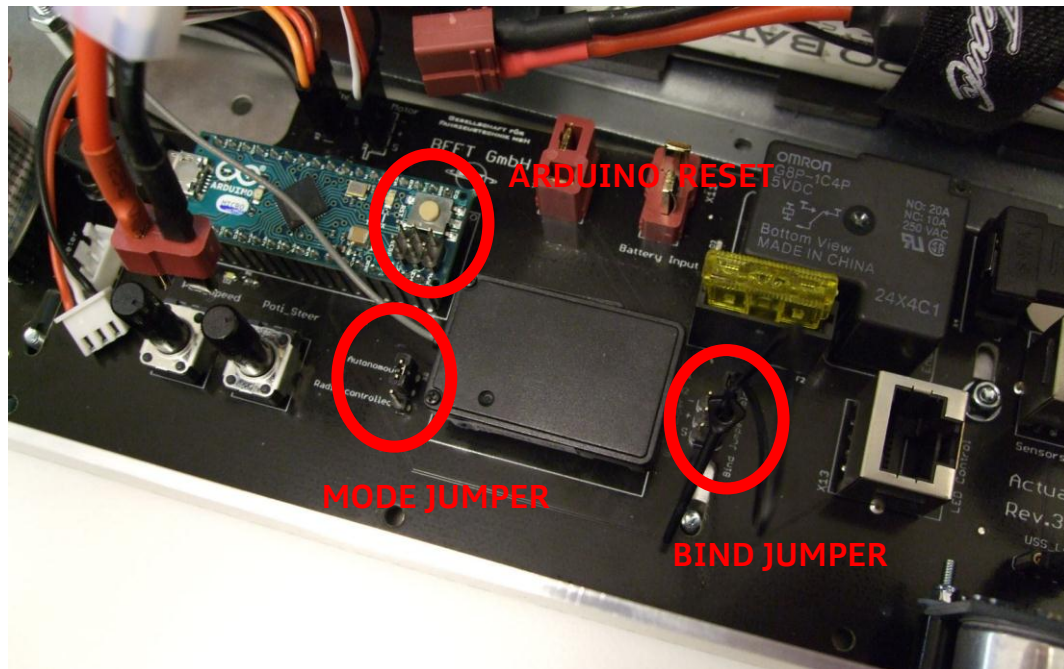
5.4.2 Deactivate Remote Radio Controller

Figure 40 Deactivate Radio Controller

To deactivate the Radio Controller again and use the Arduino and ADTF commands do the following steps:

1. Set the Mode Jumper to *Autonomous* as shown in Figure 40.
2. Press the Arduino Reset Button to restart the Arduino and use the control commands from the Arduino.

5.4.3 Bind Radio Controller to Radio Remote Receiver

The Radio Remote Receiver (mounted on the board) has to be matched to the Radio Controller to join the devices. Therefore every controller can be used with every receiver by binding the two devices. The binding also prevents receiving signals from the other controllers of the same model.
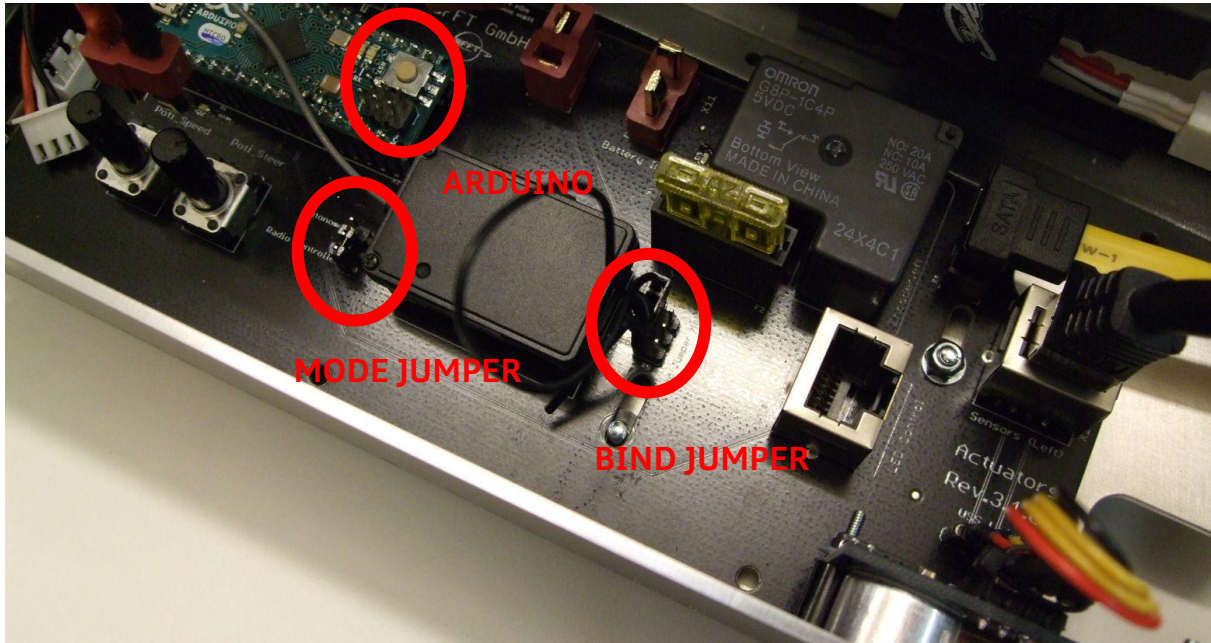
Figure 41 Binding Controller and Receiver

The binding was made for every car before delivery so normally nothing has to be done by the teams. If the binding was lost follow the instructions below to match the devices again:

1. Set the *Bind Jumper* to the *Bind Jumper* position (as shown in Figure 41)and not to the *Bind Jumper Holder* position .
2. Make sure that the *Mode Jumper* is set to *Radio Controlled* mode.
3. Reset the Arduino by pressing the Arduino reset button.
4. Remove the *Mode Jumper* once and put it again to *Radio Controlled* mode. This restarts the remote receiver. The red LED of the receiver should blink now.
5. Take the Remote Controller and make sure charged batteries are in the controller.

Figure 42 Radio Controller Buttons

6. Press and hold the *Bind Button* with a pencil or something other useful. While pressing the *Bind Button* turn the *Power Switch* to ON. The red LED on the remote receiver on the car should stop blinking.

7. No remove the *Mode Jumper* once more. After removing the *Mode Jumper* place the *Bind Jumper* back to the *Bind Jumper Holder* position again. Then place the *Mode Jumper* back to the mode *Radio controlled*.

8. Now the controller can be turned on again and binding is done.

9. Normally the speed controller has to be calibrated to the output of the radio controller. This setup routine maps the maximum motor speed to the maximum PWM signal of the radio controller system and the minimum motor speed to the minimum PWM signal of the radio controller system. The PWM signals from the Arduino which are used in software driven mode are different from the signals from the radio system, so normally the speed controller has to be calibrated each time the mode is switched from Arduino to the Radio System and vice versa. A small trick can be used to avoid this effort.
When the speed controller is powered on (after pressing the Button *Motor* near the camera) it waits for the zero position (PWM signal which corresponds to motor speed zero set during calibration procedure) . In this time the red LED of the speed controller is on. When the speed controller gets the zero position it does some more beeping and the green LED is also turned on and the speed controller works properly.
This zero position can be adjusted by the poti *THROTTLE TRIM* (shown in Figure 42). So after turning the power for the speed controller on (some first beeping from it) you can adjust this poti to set the correct zero position. If you found the correct position the speed controller does some more beeping. Maybe you have to try this multiple times, each time switch the motor on and try another position of the poti.

6 Calibration

6.1 Speed Controller Setup

Usually the Speed Controller does not need to be calibrated because it was done before delivering the vehicles. The Speed Controller Setup does a mapping from the received PWM Signal (from the Arduino or the Radio Remote Control) to the minimum and maximum range of the motor.

If for any reasons it has to be done again, please follow these instructions:

1. Start Car, Switch Speed Controller on and load and start the SpeedControllerCalibrationConfiguration Project (Chapter 3.4.5.7). Make sure the Property *Enable Speed Controller Fallback* of the Filter *Car Control* is turned to False.
2. Press the setup-button of the Speed Controller for more than a second. The green led will start to indicate the pressing will start to light continuously if the button is released now.
3. Now the full throttle has to be transmitted to the steering controller. The red light will confirm the full throttle position, then you have to release. The full throttle can be transmitted from ADTF from the Car Control Filter by moving the bar to topmost position.
4. Now the maximum Reverse has to be transmitted and will be confirmed by red and green light. Release to zero position again. The maximum reverse can be transmitted from ADTF from the Car Control Filter by moving the bar to lowermost position.
5. The Controller will proceed with some blinking and peeping to confirm the calibration.

For further details refer to the manual of the controller:

http://www.robitronic.com/en/manual-robitronic.html?file=files/anleitungen/robitronic/Speedstar_Brushless_Crawler_Manual.pdf

NOTE: The speed controller needs the zero position at startup phase. When the speed controller is powered on it does some first beeping and the red LED is turned on and it waits for the zero position. After receiving it the green LED turns on and it does some beeping. This zero position is set during the described setup procedure.

6.2 Steering Calibration

The steering calibration consists mainly of a mapping from the servo angle to a valid curvature. This can be done by the Steering Calibration Filter and is described in chapter 3.4.4.16. A hardware adjustment can be done by the screws at the steering linkage or with the potientiometer on the Actuator Board.

6.3 Camera Calibration

To calibrate the Xtion Camera Device the ADTF Filter Camera Calibration can be used. Please follow the instructions given in chapter 3.4.4.4. The CameraCalibrationConfiguration can be used for this Calibration (Chapter 3.4.5.5).

6.4 Adjustment with Potentiometers

On the Actuator Board are two potentiometer mounted which can be used for fine adjustment of the steering servo and the speed controller.

Figure 43 Potentiometer in Zero Position

The poti for the steering servo changes its zero position to make the car drive straight ahead. The poti for the speed controller modifies the mapping over the complete scope as shown in Figure 44.
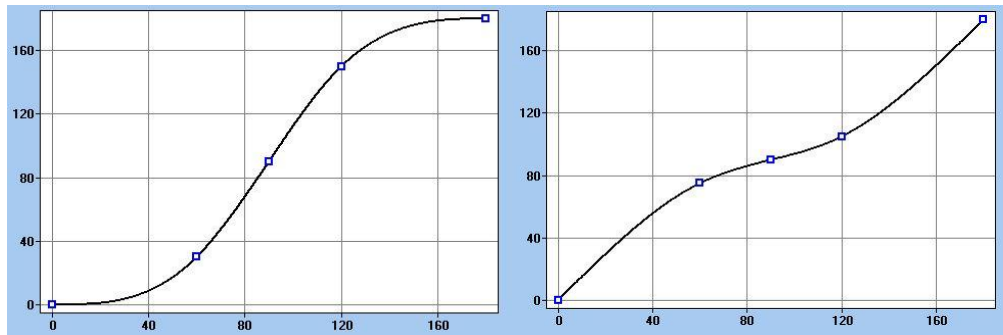

Figure 44 Modifications with speed controller poti for minimal and maximal setting

7 Competition Procedure

The procedure at the competition for each team will contain the following steps:

- The team has to bring their car to the start position and start the pITX.
- The jury will connect to the car. The main connection will be over ADTF Messagebus with the running base configuration and the specified ports. After a successful connect the Jury Module transmits the Maneuverlist to the car. Additionally a Remote Desktop Connection or a SSH Connection could be established for supervision (not for control) of the vehicle during competition.
- The Jury Module sends the predefined commands for starting maneuvers.
- The Jury supervises the drive of the vehicle. If errors occurred the Jury Module will stop the vehicle and maybe restarts a sector.

For further information refer to *Audi Autonomous Driving Cup Regelwerk 2016.*

7.1 Maneuver list

The Maneuver List is a XML File concluding several sections consisting of multiple maneuvers. This file defines what the vehicle has to do on streets, i.e. for instance to which direction it has to turn for the case of an intersection. There are no definitions of time. The file only defines the succession of actions.

An example file could look like:

```xml
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<AADC-Maneuver-List description="sample">
 <AADC-Sector id="0">
        <AADC-Maneuver id="0" action="left" />
        <AADC-Maneuver id="1" action="straight" />
        <AADC-Maneuver id="2" action="right" />
        <AADC-Maneuver id="3" action="straight" />
        <AADC-Maneuver id="4" action="straight" />
        <AADC-Maneuver id="5" action="left" />
 </AADC-Sector>
 <AADC-Sector id="1">
        <AADC-Maneuver id="6" action="straight" />
        <AADC-Maneuver id="7" action="straight" />
        <AADC-Maneuver id="8" action="straight" />
        <AADC-Maneuver id="9" action="straight" />
        <AADC-Maneuver id="10" action="straight" />
 </AADC-Sector>
 <AADC-Sector id="2">
        <AADC-Maneuver id="11" action="straight" />
        <AADC-Maneuver id="12" action="cross_parking" />
 </AADC-Sector>
 <AADC-Sector id="3">
```
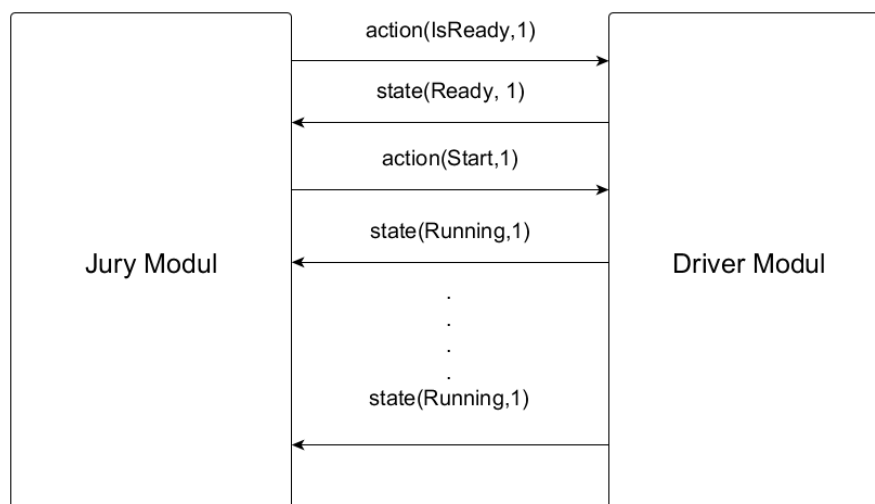
```
        <AADC-Maneuver id="13" action="pull_out_left" />

        <AADC-Maneuver id="14" action="straight" />

        <AADC-Maneuver id="15" action="parallel_parking" />

        <!--<AADC-Maneuver id="..." action="..." />-->

    </AADC-Sector>

</AADC-Maneuver-List>
```
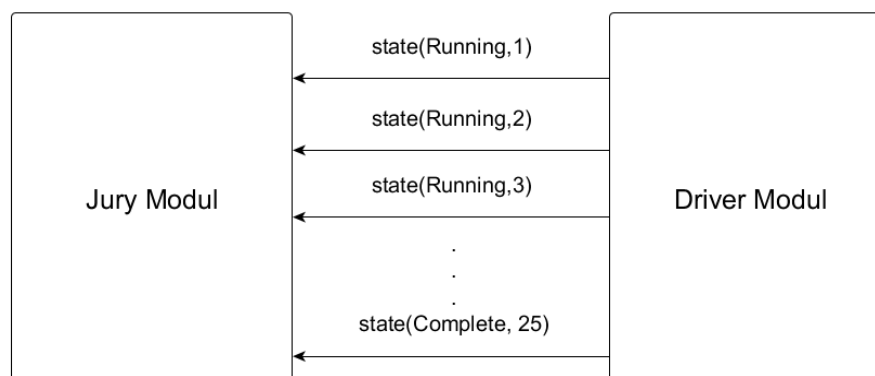
7.2 Sectors and Maneuvers

A sector can include several manuevers. The sector ranges are used for rating by the jury and therefore only a sector can be restarted. For further information refer to *Audi Autonomous Driving Cup Regelwerk 2016.*

7.3 Jury Module Procedure

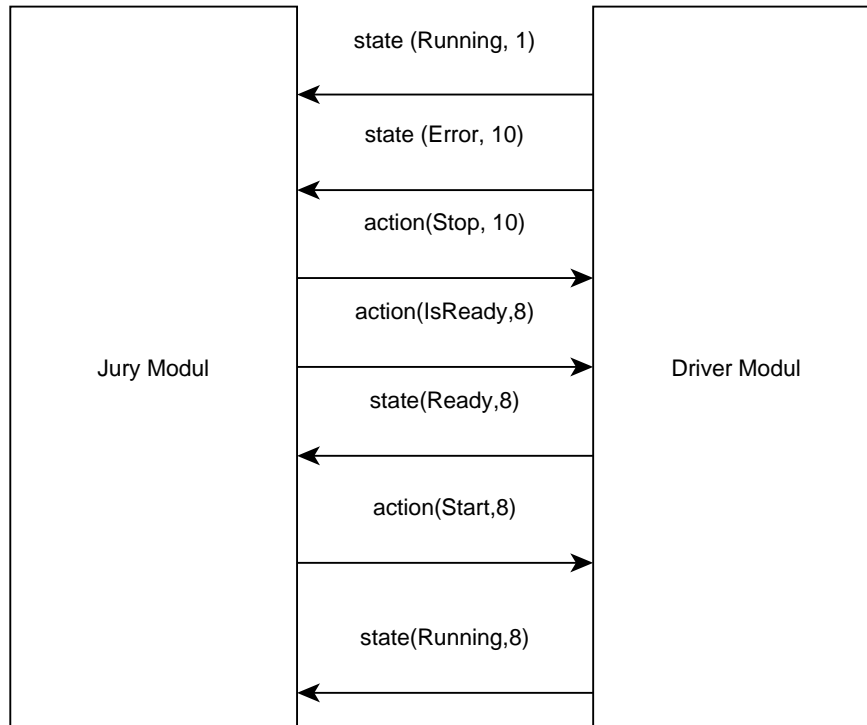### 7.3.1 Starting Vehicle

7.3.2 Finishing Parcours

7.3.3   Error and Restart

(Here: error at maneuver 10, restart at maneuver 8)

8 Troubleshooting

If any errors in software or in using the car occur please refer to the support forum at
www.audi-autonomous-driving-cup.de .

8.1 Boot Problems

Sometimes the CMOS Battery is not working properly and the pITX loses the BIOS settings after disconnection the power supply.

In this case check the following options in the BIOS menu which can be entered by pressing DEL while booting:

- Advanced -> OS Selection must be set to LINUX (not to Windows)
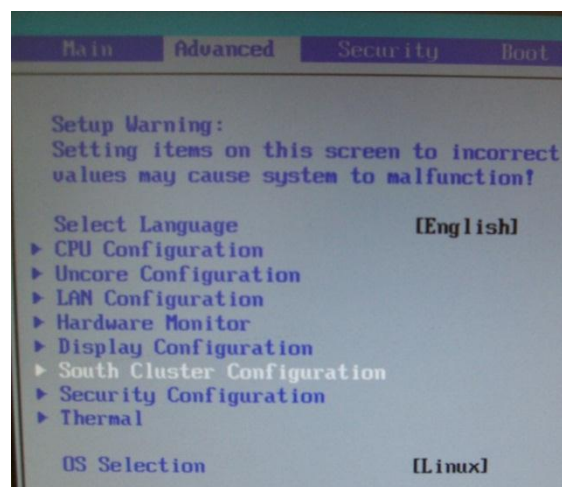

Figure 45 OS Selection in BIOS

- Advanced -> South Cluster Configuration -> Miscellaneous Configuration -> mSATA/mPCIE Selection must be set to mSATA
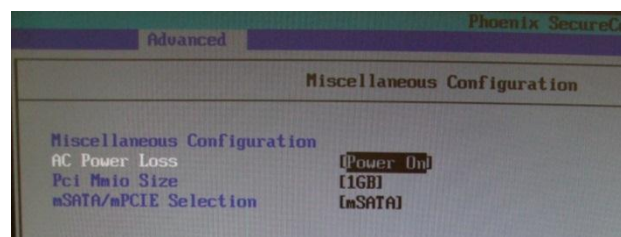

Figure 46 Enable mSATA in BIOS

9 Bibliography

1. **Cytron Technologies.** HC-SR04 User's_Manual. [Online] 05 01, 2013. [Cited: 6 3, 2015.] https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit?pli=1.

2. **Invensense.** http://www.invensense.com/. *MPU-6000 and MPU-6050 Product Specification Revision 3.1.* [Online] 11 24, 2011. [Cited: 6 3, 2016.] http://43zrtwysvxb2gf29r5o0athu.wpengine.netdna-cdn.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

3. **Honeywell.** HOA0902 Transmissive Encoder Sensor. [Online] [Cited: 6 3, 2015.] https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0CCsQFjAB&url=http%3A%2F%2Fmedia.digikey.com%2Fpdf%2FData%2520Sheets%2FHoneywell%2520Sensing%2520%26%2520Control%2520PDFs%2FHOA0902-011.pdf&ei=CrhuVZvUGubZywOHn4Eo&usg=AFQjC.

| **Author:** | BFFT Gesellschaft für Fahrzeugtechnik mbH Dr.-Ludwig-Kraus-Straße 2 85080 Gaimersheim www.bfft.de |