



**VIENA**  
Veículo Inteligente Elétrico de Navegação Autónoma  
Documentation Guide

**Current authors:**

**Bruno Tibério:** bruno.tiberio@tecnico.ulisboa.pt

**Revision: 2**

**September 2018**

## Acknowledgments

The group VIENA would like to acknowledge all past and present contributors that are making this project alive and growing. Following is a list of them in no particular order.

---

- IST** For allowing the creation of this project and providing a variety of tools and of course the facilities.
- FST Lisboa** Contributor with main parts that made possible the powertrain. Among them are two essential parts, the battery pack and inverter.
- Dr. João Fernandes** Co-supervisor of project.
- Dr. João Sequeira** Co-supervisor of project.
- Dr. Paulo Branco** Co-supervisor of project.
- Pedro Costa** Former FST leader. Valuable knowledge contributor to get us started with parts supplied by FST team as well as many suggestions to problems that arise or can arise in future. His experience as former FST leader, saved us may time in problems.
- André Agostinho** FST member. Provided us help and information specially with battery pack.
- André Antunes** Former FST member. Enlightened us in the very beginning of CAN connection with inverter.
- AEIST/ BPI** In partnership, provided funding for the project as one of the winners in an open competition fundings.

# Contents

List of Tables . . . . .	vi
List of Figures . . . . .	vii
Nomenclature . . . . .	ix
Acronyms . . . . .	x
<b>1 Introduction</b>	<b>1</b>
1.1 Guide Outline . . . . .	1
1.2 Contributions . . . . .	2
<b>2 Code Guidelines</b>	<b>3</b>
<b>3 Main Server configuration</b>	<b>5</b>
3.1 Requirements . . . . .	6
3.2 Initial Preparation . . . . .	6
3.2.1 Remote access . . . . .	7
3.2.2 Initial update and configuration . . . . .	8
3.3 Prepare CAN interface . . . . .	9
3.4 Additional packages installation . . . . .	9
3.5 Post-Installation Procedures . . . . .	11
3.5.1 Adding user to groups . . . . .	12
3.5.2 Configuring MQTT Broker . . . . .	12
3.5.3 Configuring Rpi as AP . . . . .	13
3.5.4 Configure Nginx . . . . .	16
3.5.5 Configure firewall . . . . .	16
3.6 Troubleshooting . . . . .	17
<b>4 Batteries</b>	<b>19</b>
4.1 Main battery packs . . . . .	19
4.1.1 Connections . . . . .	20
4.1.2 Power requirements . . . . .	20
4.1.3 Enabling power . . . . .	20
4.1.4 List of CAN messages . . . . .	21

4.2 Auxiliary battery pack . . . . .	21
<b>5 Sensors</b>	<b>22</b>
5.1 Definition of frames . . . . .	22
5.1.1 ECEF and ENU Frame . . . . .	22
5.1.2 Body frame, $b$ . . . . .	23
5.2 Euler angles and Rotation Matrix . . . . .	24
5.3 Quaternion . . . . .	25
5.4 Novatel OEM4-G2L FlexPak . . . . .	27
5.4.1 GNSS main errors . . . . .	27
5.5 Sparkfun Razor IMU 9DOF . . . . .	28
5.5.1 ADXL345 Digital Accelerometer . . . . .	28
5.5.1.1 Accelerometer sensor model . . . . .	29
5.5.1.2 Accelerometer calibration . . . . .	29
5.5.2 ITG3200 Digital Gyroscope . . . . .	30
5.5.2.1 Gyroscope sensor model . . . . .	30
5.5.3 HMC5843 Digital compass . . . . .	30
5.5.3.1 Magnetometer model . . . . .	30
5.5.3.2 Geomagnetic field . . . . .	31
5.5.3.3 Hard and soft-iron compensation . . . . .	31
<b>6 Steering</b>	<b>35</b>
6.1 Maxon EPOS 70/10 controller . . . . .	35
6.2 Steering sensor support . . . . .	35
6.3 Calibration process . . . . .	38
6.4 interface library . . . . .	40
6.5 Support and controller advices . . . . .	40
<b>7 Power Train controller</b>	<b>42</b>
7.1 Motor data . . . . .	42
7.2 Siemens Sinamics . . . . .	43
<b>Bibliography</b>	<b>44</b>
<b>A NovatelOEM4 GPS library Documentation</b>	<b>A.1</b>
<b>B Maxon EPOS CANopen Library Documentation</b>	<b>B.1</b>
<b>C Sensor support Drawings</b>	<b>C.1</b>
C.1 Sensor gear 32 teeth . . . . .	C.2
C.2 Half gear 54 teeth . . . . .	C.3
C.3 Sensor back spacer . . . . .	C.4

C.4	Sensor bearing holder . . . . .	C.5
C.5	Sensor case . . . . .	C.6
C.6	Bearing . . . . .	C.7
<b>D</b>	<b>Development CAN board schematics</b>	<b>D.1</b>
<b>E</b>	<b>Calibration script</b>	<b>E.1</b>
<b>F</b>	<b>Steering controller</b>	<b>F.2</b>
F.1	Steering Controller - documentation . . . . .	F.3
F.2	Steering Controller - script . . . . .	F.1

# List of Tables

2.1 Activation/deactivation of venv OS specific . . . . .	4
3.1 Default login details for Raspberry PI (Rpi) . . . . .	7
3.2 Suggested details for Rpi . . . . .	8
3.3 CAN used settings . . . . .	11
3.4 Benchmark results for MQTT . . . . .	14
3.5 Suggested AP login settings . . . . .	16
4.1 Battery pack specifications . . . . .	19
4.2 Main battery low voltage connections description . . . . .	21
5.1 WGS84 parameters necessary to transform ECEF coordinates into ENU . . . . .	23
5.2 Main source of errors in calculations using GNSS . . . . .	28
5.3 Summary of values extracted and calculated for each axis of accelerometer in both sensors	29
5.4 Magnetometers calibration results. . . . .	34
6.1 EPOS electric specifications . . . . .	36
6.2 Maxon EPOS 70/10 Led status . . . . .	36
6.3 Main HEDR-55L2_BY09 sensor characteristics . . . . .	36
6.4 Quadrature sensor settings configured in EPOS . . . . .	36
6.5 Sensor support parts . . . . .	37
6.6 Reference measurements for calibration set up scheme . . . . .	38
7.1 Original car parameters . . . . .	42
7.2 Motor parameters . . . . .	43
7.3 Motor Nameplate parameters . . . . .	43

# List of Figures

3.1	Pinout labels for RaspberryPi 3 . . . . .	5
3.2	Etcher flashing image to microSD card . . . . .	7
3.3	arp -a command example output . . . . .	7
3.4	SSH sucessfull login . . . . .	7
3.5	Raspi-config example output . . . . .	8
3.6	CAN protoboard schematic . . . . .	10
3.7	Protoboard designed for CAN interface . . . . .	11
3.8	Proposed CAN interface board for future use . . . . .	11
3.9	Current interface webpage . . . . .	17
4.1	Main battery pack and connectors . . . . .	20
5.1	ECEF frame and Local ENU frame. . . . .	23
5.2	Two frames axes example . . . . .	24
a	3D view of axes frames example . . . . .	24
b	XY plane of axes frame example . . . . .	24
5.3	Two frames axes example - Euler angles . . . . .	25
5.4	Novatel GNSS devices used . . . . .	27
a	Novatel OEM4-G2L FlexPak receiver. . . . .	27
b	Novatel GPS-701-GG antenna . . . . .	27
5.5	Razor 9DOF IMU. . . . .	28
5.6	ADXL345 calibration poses and expected output. . . . .	29
5.7	Calibration steps of ellipse data . . . . .	32
5.8	Magnetometer calibration steps applied to real data . . . . .	34
a	Trajectory used for calibration. . . . .	34
b	Magnetometer calibration . . . . .	34
c	$\psi$ angle estimation before and after calibration of magnetometers . . . . .	34
6.1	Maxon EPOS 70/10 controller . . . . .	35
6.2	Support sensor 3D in design software . . . . .	37
6.3	Support sensor assembled . . . . .	37
6.4	Ackermann steering model simplification . . . . .	39

a	Four wheel representation . . . . .	39
b	Bicycle Representation (source [38]) . . . . .	39
6.5	Calibration set up scheme . . . . .	39
6.6	Results for Calibration of steering wheel . . . . .	40

# Nomenclature

$\sigma_P$	Standart deviation associated to a position reading provided by a GNSS receiver.
$\sigma_V$	Standart deviation associated to a velocity reading provided by a GNSS receiver.
$\Omega_x$	Skew symmetric matrix of angular velocities used to represent cross products as matrix multiplications.
$\hat{q}$	Normalized quaternion.
$\otimes$	Quaternion product operation generally described as Hamilton product.
$q^*$	Quaternion conjugate.
$q$	Quaternion.
${}^b A$	Physical quantity defined in superscript frame (in this case $b$ , body frame).
${}^w R_{2D}$	A rotation matrix reduced to only two dimensions.
${}^b {}_w R$	Rotation matrix that maps physical quantities defined in subscript frame (in this case $w$ , world frame) to superscript frame (in this case $b$ , body frame).
<b><i>a posteriori</i></b>	Latin expression denoting, in this case, an estimate after performing correction using recent information.
<b><i>a priori</i></b>	Latin expression denoting, in this case, an estimate before atually having information about it and before possible correction.
<b>intrinsic</b>	Intrinsic rotation means that rotation is applied about an axis of the moving frame .
<b>lever arm</b>	A distance correction for the point of application of a physical quantity or to describe the same quantity in a specific location.
<b>polyfit</b>	Polynomial fit function available in Matlab®.
<b>pure quaternion</b>	Pure quaternion is a quaternion with real part equal to zero.

# Acronyms

<b>9DOF</b>	Nine Degrees of Freedom.
<b>A-GNSS</b>	Assisted GNSS.
<b>ADC</b>	Analog to Digital Converter.
<b>AHRS</b>	Attitude and Heading Reference System.
<b>AP</b>	Access Point.
<b>CAN</b>	Controller Area Network.
<b>CEP</b>	Circular error probable.
<b>DC</b>	Direct Current.
<b>DHCP</b>	Dynamic Host Configuration Protocol.
<b>DIY</b>	Do It Yourself.
<b>DNS</b>	Domain Name System.
<b>DOF</b>	Degrees of Freedom.
<b>ECEF</b>	Earth Centered, Earth Fixed.
<b>EEPROM</b>	Electrically-Erasable Programmable Read-Only Memory.
<b>EKF</b>	Extended Kalman Filter.
<b>ENU</b>	East North Up.
<b>EPOS</b>	Easy Positioning System.
<b>FS</b>	Full Scale.
<b>FST Lisboa</b>	Formula Student Team Lisboa.
<b>GNSS</b>	Global Navigation Satellite System.
<b>GPS</b>	Global Position System.
<b>I<sup>2</sup>C</b>	Inter-Integrated Circuit bus.
<b>IC</b>	Integrated Circuit.
<b>IMU</b>	Inertial Measurement Unity.
<b>INS</b>	Inertial Navigation System.
<b>IST</b>	Instituto Superior Técnico.
<b>LSB</b>	Least significant bit.
<b>MAC</b>	Media Access Control.
<b>MARG</b>	Magnetic Angular Rate and Gravity.

<b>MCU</b>	Microcontroller Unit.
<b>MEMS</b>	Microelectromechanical systems.
<b>MQTT</b>	Message Queue Telemetry Transport.
<b>OS</b>	Operating System.
<b>PCB</b>	Printed Circuit Board.
<b>PID</b>	Proportional Integral Derivative.
<b>qc</b>	Quadrature counters.
<b>QoS</b>	Quality of Service.
<b>rms</b>	root mean square.
<b>Rpi</b>	Raspberry PI.
<b>RTC</b>	Real Time Clock.
<b>SBC</b>	Single Board Computer.
<b>Soc</b>	System on Chip.
<b>SPI</b>	Serial Peripheral Interface bus.
<b>UART</b>	Universal Asynchronous Receiver/Transmitter.
<b>VIENA</b>	Veículo Inteligente Elétrico de Navegação Autónoma.
<b>WGS84</b>	World Geodetic System.
<b>ZVU</b>	Zero Velocity Update.

# **Chapter 1**

## **Introduction**

Instituto Superior Técnico (IST) is currently developing research in autonomous electrical vehicles, namely converting from commercial vehicles with upgraded power management. This provides a motivating/attracting setup for new students and a testbed for industry solutions and it is the main motivation for this work.

The main goal of the project is a conversion of an electrical car (Fiat Elettra) property of IST into an autonomous vehicle as framework for future projects or research in this field and also energy efficiency. Within the conversion, researchers, student and collaborators knowledge acquired during academic cycle is put into test and evaluated in a real situation.

With environmental issues and technological waste in mind, this project has also been focused on the reuse of parts and material from other IST projects by the simple fact that they have been replaced by improved versions or will no longer serve the current goals of those projects. Not only it is given a new propose for those parts but it will also allow the cost reduction.

This guide aims to be auxiliary documentation and future memory source as part of the IST project named Veiculo Inteligente Elétrico de Navegação Autónoma (VIENA) and as well as final report for fellowship BL43/2018.

Although all the instructions are given based on Linux operating system, they should be similar to any other Operating System (OS) and the majority of code developed is made in Python, intending to be as much cross-platform as possible.

### **1.1 Guide Outline**

This documentation guide is organized in five main chapters, Code guidelines, server, main sensors, controllers, miscellaneous. In chapter code guidelines is described suggested and used assumption especially focused in code structuring and tools used in development. In the server is provided information about the used hardware, designed pieces, software configuration needed to get started and connections. In main sensors will be discussed the work done with the current available sensors. New sensor additions should be documented under this chapter. In controllers will be reported mainly hardware

controllers and software necessary to connect, configure or communicate with it. Miscellaneous contain other parts that do not fit particularly inside any of those chapters but are necessary or may help in the project.

## 1.2 Contributions

During the fellowship BL43/2018, it was made the main contributions:

1. Development of 3D printed parts for allowing the control of steering wheel without disassembling or violating the integrity of steering shaft column
2. Development of a library in Python based on CAN bus protocol to enable interconnection between software and hardware to control the steering wheel.
3. Development of prototype circuit to add CAN communication for main computer
4. Development and design of PCB for future CAN communication to replace the protoboard for main computer
5. Identification of the function that relates the steering wheel position with angle of the front wheel in a bicycle model of the car.
6. Mounting of available sensors (two GNSS unities and one IMU 9DOF)
7. Calibration of the inertial sensor.
8. Updated code related with MARG sensor.
9. Updated code related with GNSS unities.

Although not initially planned, but because the change of hardware and/or adversities found during the fellowship, following additional work was also contributed:

1. Study of interconnection of the main battery pack
2. Study of software used for management of main battery pack
3. Study of hardware necessary for management of main battery pack
4. Study and initial software development for inverter received from FST Lisboa
5. Creation of an headless structure and an AP station based on Rpi to communicate with vehicle
6. Initial development of an MQTT based design to control the vehicle or check its status.
7. Development of library to interact with Schneider Altivar 71 (abandoned due to change of inverter)

Main code contributions are available under Github repository in <https://github.com/brtiberio/VIENA> and will be kept up to date as soon as possible. The developed 3D pieces are also present in that repository and the companion drawings are shown in this guide also. The developed PCB schematics and board layouts are also presented in this guide.

# Chapter 2

## Code Guidelines

In this chapter is described the main guidelines used for coding and documentation as well as relevant suggestions. Since the majority of code is developed in Python, the guidelines are provided for that language. However similar suggestion may apply for the other languages cases. Here follows a list of suggestions:

### Git

The first suggestion is the code version control system, Git. It has a lot integrations with majority of IDE's and have lot of free tools to manage it. It allows easy contributions from multiple users. Documentation and guides to become familiar with it can be found at [try.github.io](https://try.github.io)

### Python3

Since Python core team is dropping support for python 2.7.x in 2020 [1] and some of important package developers are also dropping support for it [2], the chosen **version is the 3.X**.

### Use virtual environment

Typically, every Linux distribution uses python to run critical routines. Perturbing the main ecosystem of python packages should be avoided, at least during development phase. Raspbian is no exception, so it is suggested to use virtual environments. It is used to create a isolated local installation in the directory you are working. Since version 3.5 the official recommended tool for creating virtual environments is the venv [3]. To create a virtual environment use the following console command `python3 -m venv /path/to/new/virtual/environment` or assuming the user is currently in desired local path, simplify to `python3 -m venv`. To activate/deactivate the local environment follow the specification accordingly to used OS as present in table 2.1, using the same assumption as before. After activation, the shell prefix will change from the default to `(<path of venv>)` which is a easy away to check if environment is active or not.

### Requirements files (`requirements.txt`)

Requirement files are an easy away to install all the requisites necessary to run the code and ensure repeatability of installations. Unless any particular reason, a restriction of any package version should be avoided. Requirements file can be as simple as a list of necessary packages or

more complex structure, if needed [4]. Installation of requirements is done by running the following command `pip install -r requirements.txt`

### Google style docstrings

A good documentation is a key point for keeping good readability of a project. Docstrings format from Google Style Guide [5] is adopted. Many projects can fail from bad documentation.

### Sphinx with automation

Similar to previous point, `sphinx` will allow the auto documentation generation from code. Is a popular tool for creating documentation and is used also in [readthedocs.org](http://readthedocs.org), that allows to integrate documentation update with commits in the three main git repository management services, GitHub, Bitbucket, and GitLab. Using autodoc and napoleon extension allow an easy maintenance of code documentation.

### Argparse

Use `argparse` to create clean and readable argument handling if necessary. It auto generates help, description and handles unknown options.

### Logging

Since the main intention is to run programs in a headless computer, the `logging module` should be used instead of typical `print()` function. Not only it helps keep tracking of events defined in code, but also keeps track of other modules events. It also allows the creation of multiple destinations using the handlers and each handler can have its own format. Typical handlers used in project are files, console and websockets (currently only in development branches)

Shell	Activate	Deactivate
POSIX		
bash/zsh	source ./bin/activate	deactivate
fish	source ./bin/activate.fish	deactivate
csh/tcsh	source ./bin/activate.csh	deactivate
Windows		
cmd	.\Scripts\activate.bat	deactivate
PowerShell	.\Scripts\Activate.ps1	deactivate

Table 2.1: Activation/deactivation of venv OS specific

# Chapter 3

## Main Server configuration

In order to control the current features already developed and used in the car it is used a [Raspberry Pi](#) (currently version 3 model B). The Raspberry Pi, from now on denoted as Rpi, is a Single Board Computer (SBC) affordable and low power, widely used among community and developed by Raspberry Pi Foundation. Figure 3.1 show the relevant parts of Rpi and also the pinout labels.

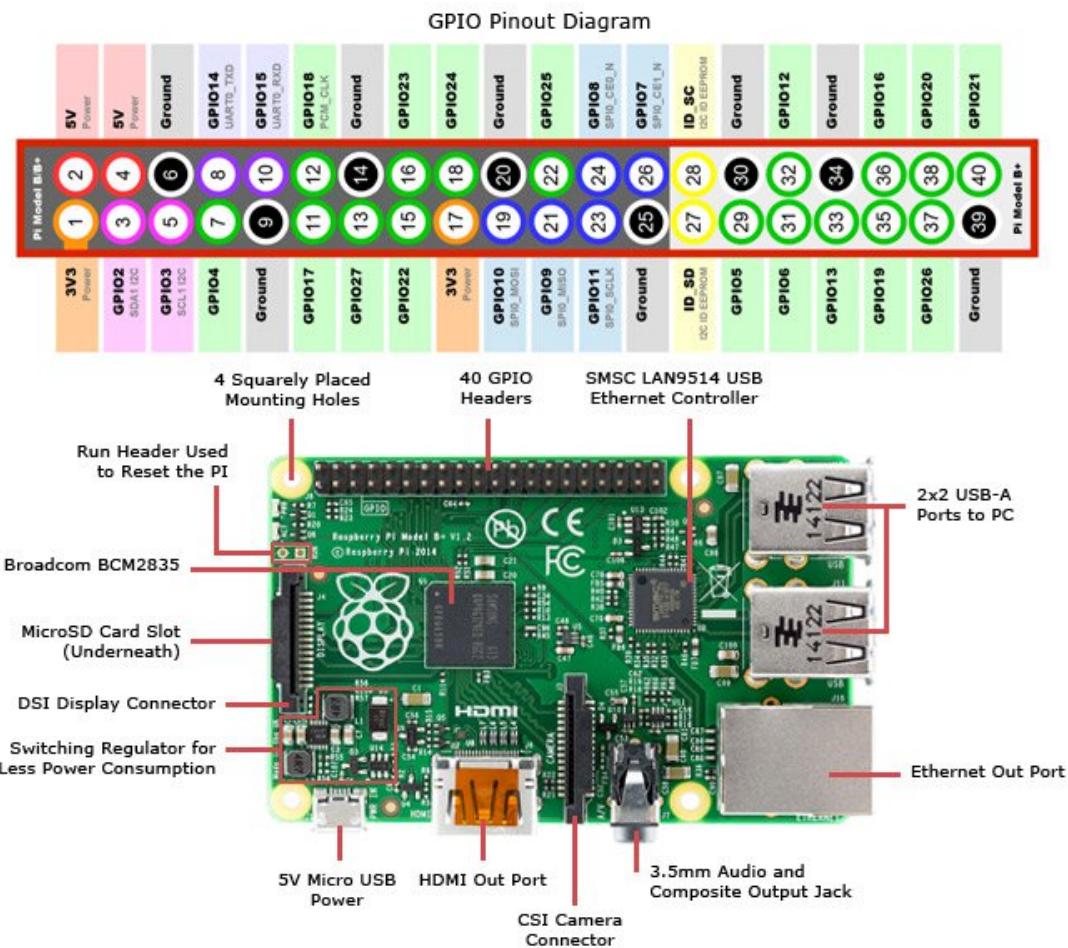


Figure 3.1: Pinout labels for RaspberryPi 3 ([source](#))

## 3.1 Requirements

For using the SBC it is required the following hardware:

- **Raspberry Pi SBC** Recommended version at least 3 since it has a built in wifi.
- **MicroSD card** Recommended with 8Gb capacity at least.
- **Power Supply** Recommended 2.5 Amps . While developing, it is used as the main power for the SBC.
- **Host computer (any OS)** with internet and ability to read MicroSD cards.
- **Ethernet cable** for connecting to router/switch for internet access. It can also be used an crossover cable connecting directly to PC if it is able to share internet connection.

## 3.2 Initial Preparation

This preparation will focus on providing instructions to a minimal headless setup. For this reason, the recommended OS image version is the Raspbian Lite.

For this step it is recommended to be used the [Etcher](#) software for burning the OS image into microSD card. It is assumed the user is familiar with ssh capable software, for example [Putty](#). The next steps are mainly based in the documentation guide provided by [6].

1. Download the OS image, Raspbian Lite version in [here](#)
2. Connect the MicroSD to host computer.
3. Open Etcher and:
  - (a) Select OS image or zip file that have been downloaded.
  - (b) Select the SD card you wish to write your image to.
  - (c) Review your selections and click 'Flash!' to begin writing data to the SD card.
  - (d) after successful write, continue to next step.
4. From microSD card, open the boot partition.
5. Create a new blank file with name "ssh" without extension. This will allow to enable the ssh daemon at first boot time.
6. Remove card from host computer and insert on raspberry PI.
7. Plug in the Ethernet cable and connect the Rpi to the same local network as host computer.
8. Plug in the power supply to Rpi.

If everything went as expected, the Rpi will start to boot and prepare the first setup. It will be seen led light blinking indicating activity. After a few minutes, it should be possible to access it remotely via ssh

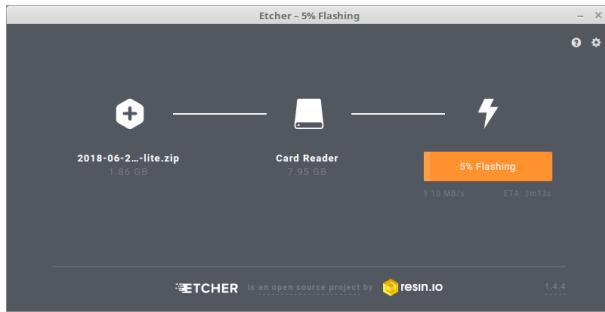


Figure 3.2: Etcher flashing image to microSD card

### 3.2.1 Remote access

The next step is to find the ip address of the Rpi. For example, in linux terminal type `arp -a`. In figure 3.3 is seen an output example. In red stroke is shown the Media Access Control (MAC) address of a Rpi. Every MAC is unique and the first three bytes are fixed in every Rpi which correspond to organizationally unique identifier [7] associated to Raspberry Pi Foundation, B8:27:EB [8].

```
File Edit View Search Terminal Help
bruno@laptop ~ $ arp -a
? (192.168.1.254) at a4:b1:e9:aa:8f:06 [ether] on eth0 Left(1)
? (192.168.1.14) at b8:27:eb:9c:fc:48 [ether] on eth0
? (192.168.1.65) at 00:25:2e:aa:2a:0f [ether] on eth0 report_025Servertex
? (192.168.1.253) at a6:b1:e9:aa:8f:06 [ether] on eth0
? (192.168.1.1) at 04:b1:67:09:aa:1c [ether] on eth0 Review your selected
bruno@laptop ~ $ report_b_DB
report_Preamble B item after successful
report_End(enumerate)
```

Figure 3.3: arp -a command example output

Perform the first login with using the default login details as shown in table 3.1. Using the discovered ip for the Rpi, in Linux, the first login may be performed using `ssh pi@<ip-of-Rpi>` command and entering the default password.

---

<b>username:</b>	pi
<b>password:</b>	raspberry

---

Table 3.1: Default login details for Rpi

```
pi@raspberrypi: ~
login as: pi
pi@192.168.27.130's password:
Linux raspberrypi 4.14.69-v7+ #1141 SMP Mon Sep 10 15:26:29 BST 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Sep 12 09:58:07 2018

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
pi@raspberrypi: ~ $
```

Figure 3.4: SSH sucessfull login

### 3.2.2 Initial update and configuration

If the user has been granted with permission to login, the next steps are used to perform a few tweaks.  
To do that user must use the command `sudo raspi-config`. Example output is seen in figure 3.5.

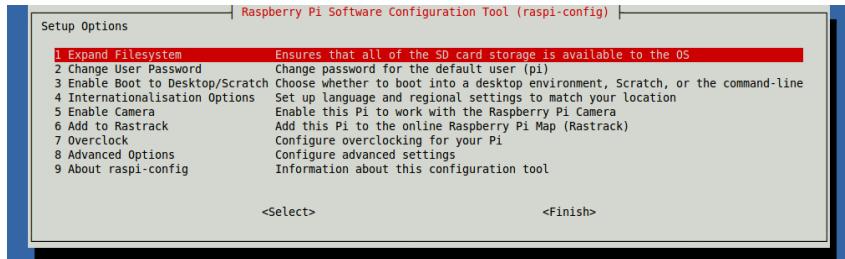


Figure 3.5: Raspi-config example output

It is recommended to configure the following:

- Set Keyboard Layout
- Update to latest software
- Set Timezone
- Set language [optional]
- Change default login password
- Configure Wifi-zone [if present]

If you are connected directly to a switch via Ethernet cable, skip wifi configuration.

- Enable Serial Peripheral Interface bus (SPI) for Controller Area Network (CAN) controller
- Enable Inter-Integrated Circuit bus (I<sup>2</sup>C) for RTC
- Set hostname
- change default password
- change memory for GPU

The current settings are presented in table 3.2.

<b>Username:</b>	pi
<b>Password:</b>	fiatelettra
<b>Hostname:</b>	raspberrypi
<b>wait for network at boot:</b>	no
<b>Language:</b>	en_GB.UTF-8 UTF-8
<b>Timezone:</b>	Europe → Lisbon
<b>Keyboard layout:</b>	pt_PT
<b>WIFI country:</b>	PT_Portugal
<b>SPI:</b>	on
<b>I<sup>2</sup>C:</b>	on
<b>Memory split:</b>	16 (minimum since is running headless)

Table 3.2: Suggested details for Rpi

After successfully changed the settings, perform a full update and upgrade by running:

```
sudo apt update && sudo apt upgrade -y
```

### 3.3 Prepare CAN interface

The current protoboard uses the Integrated Circuit (IC) [MCP2515](#) as CAN network controller and [MCP2511](#) as CAN transceiver. Figure 3.7 shows the used protoboard with highlighted connections names and parts.

The schematic circuit implemented in the protoboard is present in figure 3.6. Current raspbian kernel, automatically support the can interaction via SPI to IC MCP2515, making it available via SocketCAN. To enable that, it is necessary to configure MCP2515 driver on device tree overlay and install the can-utils package.

- Install can-utils using `sudo apt install can-utils`
- Enable MCP2515 overlay by using `sudo nano /boot/config.txt` and append at end:

```
#CAN bus controllers
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=25
dtoverlay=spi-bcm2835
```

- To enable the interface at boot time, use `sudo nano /etc/network/interfaces` and append at end:

```
auto can0
iface can0 inet manual
    pre-up /sbin/ip link set can0 type can bitrate 1000000 \
        triple-sampling on restart-ms 10
    up /sbin/ifconfig can0 up
    down /sbin/ifconfig can0 down
```

This will enable the CAN interface at boot time with the setting present in table 3.3.

The protoboard is considered temporary and new board as been designed and projected to include not only a CAN controller but also a Real Time Clock (RTC) chip, a battery holder for RTC chip and a DC-DC converter to power Rpi from 12V battery pack. In figure 3.8 is show the proposed and developed circuit for replacing the protoboard. The respective schematic is present in appendix D. Suggested board was designed in Eagle Software and files are also included in the Github repository.

For connecting the protoboard please refer to figure 3.1 to see the respective pins in the Rpi side.

### 3.4 Additional packages installation

The following list describes additional package installation that are currently used or are planned to be used.

## python3-pip

Required to enable pip manager.

## python3-venv

To enable the creation of virtual environments.

**libatlas-base-dev**

Required to install numpy package.

## mosquitto

Local Message Queue Telemetry Transport (MQTT) broker used currently in development branch.

It might be useful to install also the `mosquitto-clients`.

## **build-essential**

Required if needed to compile anything.

**git** Add git support to raspbian.

nginx

Local lightweight web server application used to control or see current status of vehicle (in development)

**ufw** Firewall interface to improve security.

## **dnsmasq**

Provides network infrastructure for small networks as Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP). Will be used to configure Rpi as an Access Point (AP).

# hostapd

Daemon for configuring and enabling the AP

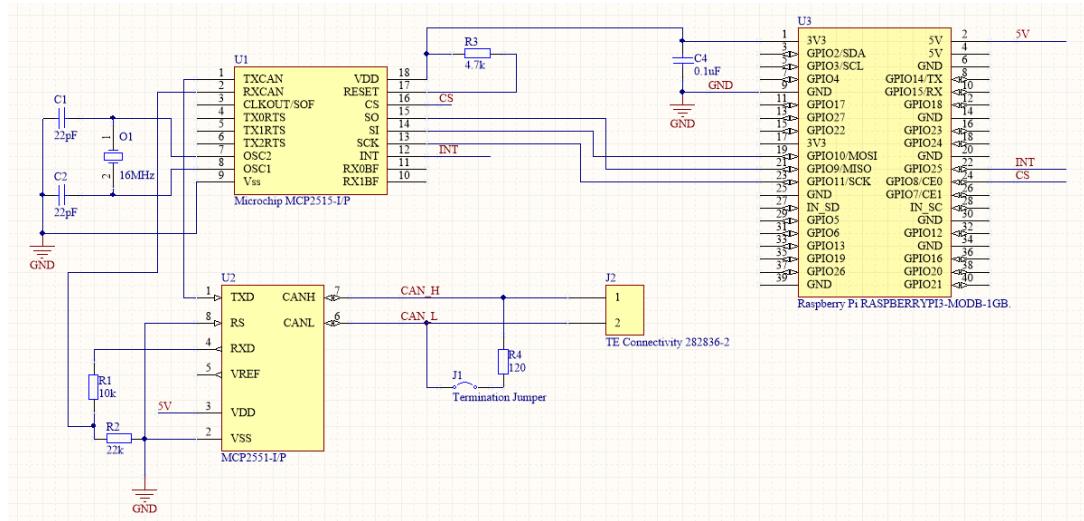


Figure 3.6: CAN protoboard schematic

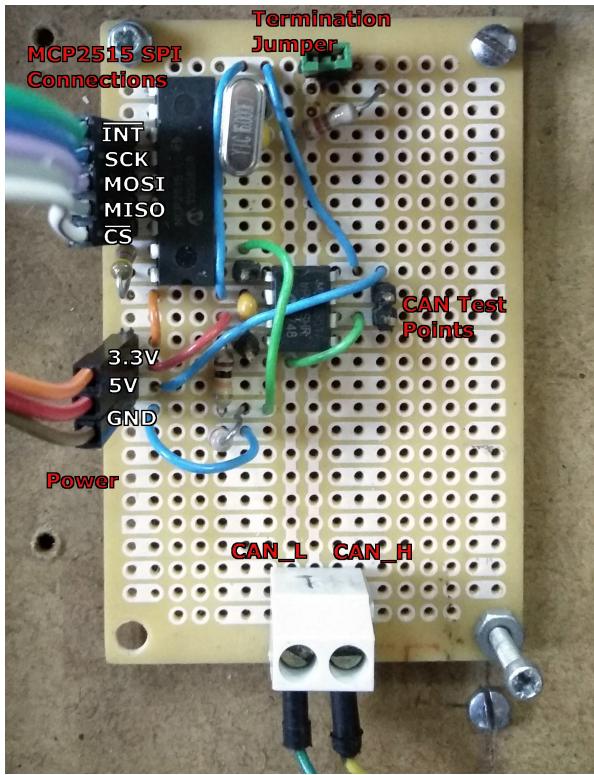


Figure 3.7: Protoboard designed for CAN interface

Parameter	Value
Bitrate	1Mbps
Triple Sampling	on
Sampling point	0.75
Restart	10ms

Table 3.3: CAN used settings

### 3.5 Post-Installation Procedures

After installing the recommended packages, user should perform several configurations. In this section is described each of them.

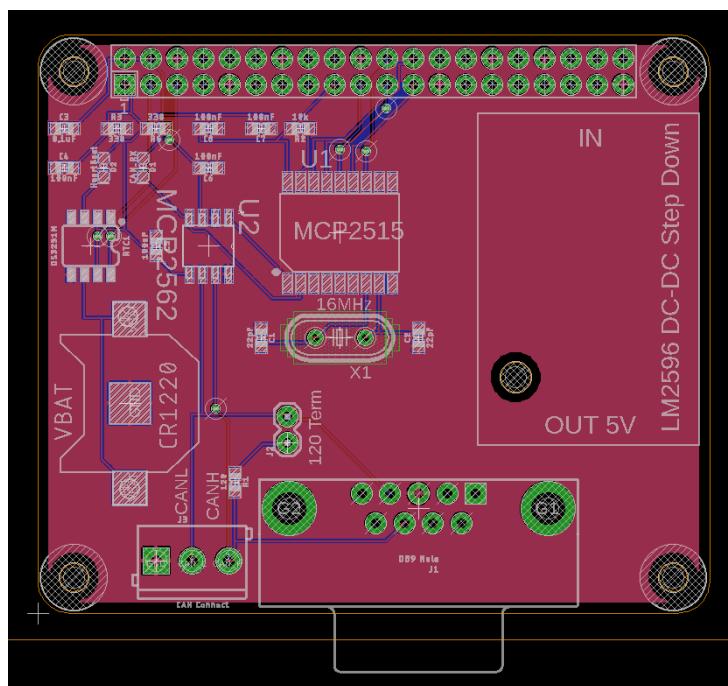


Figure 3.8: Proposed CAN interface board for future use

### 3.5.1 Adding user to groups

After adding user to any group, **a logout must be performed to changes take effect**. Changes can be confirmed by using the command `groups` that prints the groups current user is in.

#### www-data

Grant user ability to perform changes on /var/www. Run `sudo adduser pi www-data`

#### dialout

Grant user ability to use ports. Run `sudo adduser pi dialout`

### 3.5.2 Configuring MQTT Broker

Current and planned development roadmap use MQTT broker to change messages between user or high level software and hardware interface. It is also used websockets and in this section will be shown how to enable it. The current used broker is [mosquitto](#). If not already installed, use:

```
sudo apt install mosquitto
```

After installation edit mosquitto.conf by using `sudo nano /etc/mosquitto/mosquitto.conf` and add an additional port for websockets protocol. If advanced configuration is needed refer to [9]. In order to use mosquitto with Quality of Service (QoS) of 1 or 2 and accept large queues, it was increase maximum queued messages to 10000. Also, since it is not required, persistence file was disabled.

```
pid_file /var/run/mosquitto.pid
persistence false
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

port 1883
listener 8080
protocol websockets

# max_inflight_messages 0
max_queued_messages 10000
```

Since official repository did not have the latest release of mosquitto, it was necessary to manually update it using:

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
```

```

sudo apt-key add mosquitto-repo.gpg.key
cd /etc/apt/sources.list.d/
sudo wget http://repo.mosquitto.org/debian/mosquitto-stretch.list
sudo apt update
sudo apt upgrade

```

In order to discuss future implementations it was tested the ability of using the MQTT interface for fast rate messages. For that, in table 3.4 are shown different approaches using local vs online brokers and implementation on Rpi vs local laptop (Lenovo T480).

As summary, test conditions are as follow:

- 1000 messages sent, using an Int32 with the number of message as payload.
- Same QoS in both ends.
- Using same PC to run the sender and receiver clients (Lenovo T480)
- Comparison between local MQTT broker (mosquitto), running in RaspberryPI 3 and Laptop, as well as online servers.
- Mean of 5 consecutive tests.
- Transport used is websockets in both clients.

Scripts used for testing can be seen in [VIENA-IST/HighLevelCommBenchMarks repository](#).

### 3.5.3 Configuring Rpi as AP

Use only if necessary and not working with an external router.

While in development user is advised to use ethernet connection between development PC and Rpi, it is useful to configure Rpi as an AP so users can connect to it using wifi in field environment. Here will be assumed that internet connection is provided via ethernet cable during development phase, for example via host PC, and during normal operation Rpi will not be connected to internet. This section follows the main instructions given in [10].

- Turn off hostapd and dnsmasq. Perform this using systemctl with:

```

sudo systemctl stop dnsmasq
sudo systemctl stop hostapd

```

- Configuring a static IP. Edit `/etc/dhcpcd.conf` with nano and change the file as shown below, assuming the user wants to assign the server IP address as 192.168.4.1, using wlan0 as interface.

```

# static ip address for wlan0 to be used as AP
interface wlan0
    static ip_address=192.168.4.1/24
    nohook wpa_supplicant

```

Receiving Results						
Server	Is local?	QoS	Mean	std	min.	max.
mosquitto	Yes	0	9754.14	1735.97	7316.35	11653.73
mosquitto	Yes	1	4917.01	172.89	4680.69	5125.19
mosquitto	Yes	2	3663.86	234.97	3339.15	3999.98
mosquitto (Rpi)	Yes	0	820.48	36.45	781.01	860.08
mosquitto (Rpi)	Yes	1	780.07	73.96	650.22	835.46
mosquitto (Rpi)	Yes	2	390.70	30.80	356.01	426.96
mosquitto.org	No	0	103.71	4.47	96.85	108.64
mosquitto.org	No	1	107.87	9.85	90.55	113.12
mosquitto.org	No	2	57.69	0.65	56.89	58.44
broker.hivemq.com	No	0	3209.51	2631.14	555.91	6589.80
broker.hivemq.com	No	1	4917.01	172.89	4680.69	5125.19
broker.hivemq.com	No	2	3663.86	234.97	3339.15	3999.98
Transmitting Results						
Server	Is local?	QoS	Mean	std	min.	max.
mosquitto	Yes	0	20855.58	6606.65	13611.86	29385.87
mosquitto	Yes	1	6370.67	343.24	5797.45	6627.05
mosquitto	Yes	2	4653.94	189.67	4374.28	4883.84
mosquitto (Rpi)	Yes	0	5969.37	570.65	5100.85	6436.20
mosquitto (Rpi)	Yes	1	818.21	25.51	784.16	850.88
mosquitto (Rpi)	Yes	2	409.12	22.39	372.51	430.22
mosquitto.org	No	0	6863.24	1031.36	5768.09	8451.10
mosquitto.org	No	1	106.53	8.66	91.23	111.90
mosquitto.org	No	2	57.64	0.59	56.92	58.43
broker.hivemq.com	No	0	6382.99	905.09	4832.09	7004.41
broker.hivemq.com	No	1	6370.67	343.24	5797.45	6627.05
broker.hivemq.com	No	2	4653.94	189.67	4374.28	4883.84

Table 3.4: Benchmark results for MQTT

- Restart dhcpcd using `sudo service dhcpcd restart`
- Configuring the DHCP server (dnsmasq). Perform a copy of original file and create a new one

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
sudo nano /etc/dnsmasq.conf
```

On new opened configuration file place that will allow dhcp on interface wlan0 giving ip addresses to clients from 192.168.4.2 to 192.168.4.20 with a lease time of 24h:

```
interface=wlan0
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
```

- Configuring the access point host software (hostapd). Table 3.5 show the suggested SSID and password. Edit file /etc/hostapd/hostapd.conf and add:

```
interface=wlan0
driver=n180211
ssid=VIENA
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=fiatelettra
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Now edit file `sudo nano /etc/default/hostapd` and add to it:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

- Restart services:

```
sudo systemctl start hostapd
sudo systemctl start dnsmasq
```

- Edit /etc/sysctl.conf and uncomment line `net.ipv4.ip_forward=1`
- Add a masquerade for outbound traffic on ethernet port:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

- Edit /etc/rc.local and add this just above `exit 0` to install these rules on boot:

```
iptables-restore < /etc/iptables.ipv4.nat
```

- Reboot. After it user should be able to connect to configured network and current user authentication settings.

<b>SSID</b>	VIENA
<b>password</b>	fiateletra

Table 3.5: Suggested AP login settings

### 3.5.4 Configure Nginx

The configuration of nginx will be minimally changed from default values and will follow mainly [11]. For more advanced uses, please refer for example to [12] and [13]. The use of web server is intended for current in development branch of MQTT web interface. A user can connect to local network of Rpi and see current information displayed in dashboard webpage.

If not already, do `sudo apt install nginx`. Test installation by starting the webserver with

```
systemctl status nginx.service
sudo systemctl start nginx.service
```

Change user permissions for `/var/www/html` and clone the current repository of webpage interface using the following commands:

```
sudo rm -R /var/www/html/*
sudo chown -R pi:pi /var/www/html
git clone https://github.com/brtiberio/VIENA_interface.git /var/www/html
```

Use a browser using the ip of Rpi or the defined hostname.local to confirm it is working (see figure 3.9).

### 3.5.5 Configure firewall

First enable firewall by using `sudo ufw enable`. After use the following console command to enable the previously configured ports for ssh connection, mqtt connections and http server configured in nginx:

```
sudo ufw allow 22
sudo ufw allow 80
sudo ufw allow 1883
sudo ufw allow 8080
```

If needed, confirm status with `sudo ufw status`. For more advanced use run `man ufw`.

## 3.6 Troubleshooting

### The command `arp -a` do not show my Rpi

In that case, try use the `sudo nmap -sS 192.168.1.0/24`, assuming the 192.168.1.0/24 is your local network. This is a time consuming command!

### Cannot find my Rpi IP. Is it even running?

Maybe there is some problem with boot or bad microSD reading. The easiest solution is to connect a monitor and keyboard. Check messages at boot time. If nothing seems strange, manually login and then check if ethernet connection is ok using `ifconfig`

### CAN seems to not be working

First see if MCP2515 is successfully connected.

1. Use `dmesg |grep mcp2515*`. If it says successfull configured, go to next step. If not, recheck connections and restart. Also make sure the oscilator frequency match the used on boot configuration file settings as seen [here](#).
2. Check if can is detected as a socketcan interface. Use `ifconfig`. If CAN is available, it should show a section for can0. If not, re-check if can-utils is installed.
3. View details about CAN connection. use `ip -details -statistics link show can0` to get more information. If current state is `BUS-OFF`, manually restart the CAN by using the following:

```
sudo ip link set can0 down  
sudo ip link set can0 up
```

### webpage is nginx default

Restart nginx service by using `sudo systemctl restart nginx.service`. If it still shows a different page, confirm the enabled sites are correct. Use:

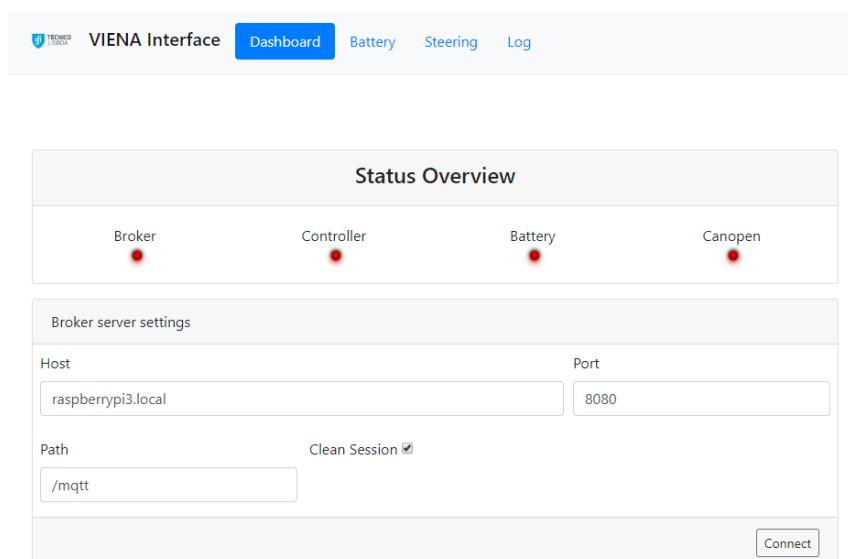


Figure 3.9: Current interface webpage

```
nano /etc/nginx/sites-enabled/default
```

The default values for listen setting and root should be:

```
listen 80 default_server;  
listen [::]:80 default_server;  
...  
root /var/www/html;  
  
# Add index.php to the list if you are using PHP  
index index.html index.htm index.nginx-debian.html;
```

If they are different, change it accordingly to meet desired configuration, by editing:

```
sudo nano /etc/nginx/sites-available/default
```

If still show different then expected, try clear browser cache files.

# **Chapter 4**

## **Batteries**

To operate the car is required two main source of power coming from two distinct sets of batteries. The main battery is an high voltage Direct Current (DC) pack. To activate it is necessary a 24V source. For that is required a secondary power source, where it is extracted the low voltage. They contain a Battery Management System (BMS) managed by a micro controller with whom the user interacts to activate the high voltage section and get information about the status of battery. The communication is performed via CAN interface.

Currently is used two independent set of batteries to provide low voltage. A pack with 24V and another with 12V. However is currently in study and design a single pack with 48V of nominal voltage with integrated BMS to unify the low voltage source and intermediate DC-DC switching converters will be used to provide 24V and 12V rails.

### **4.1 Main battery packs**

The power train battery consists in two interconnected battery packs each other with 72 individual cells. Both pack are connected in series deploying the total voltage for the Siemens power inverter. The battery pack specifications are present in table 4.1. The main battery connections are shown in figure 4.1 In orange is seen the high voltage connectors and the black ones are the low voltage and also control system connectors.

Parameter	Value
Maximum tractive system voltage	600 V
Nominal tractive system voltage	532,8 V
Control system voltage	24 V
Accumulator configuration	144s1p
Total accumulator capacity	10 Ah

Table 4.1: Battery pack specifications (source [14])

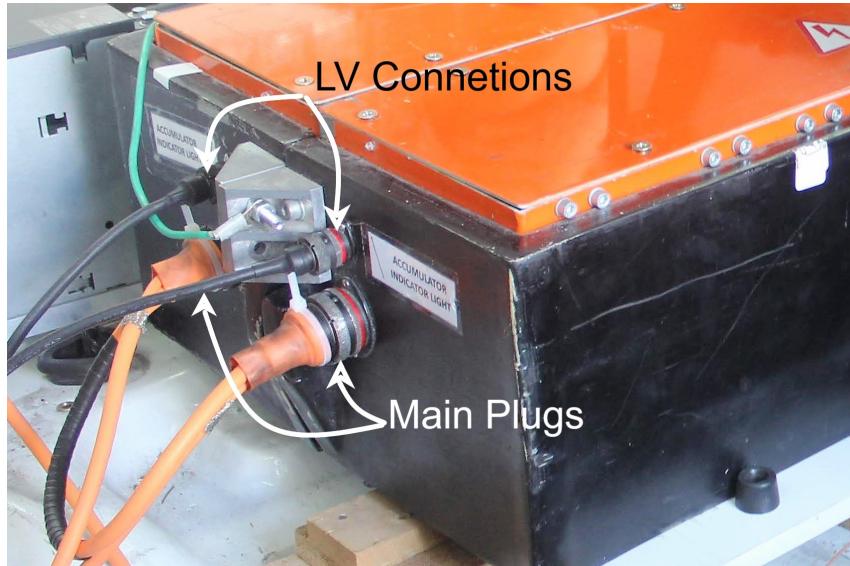


Figure 4.1: Main battery pack and connectors

#### 4.1.1 Connections

The low voltage connectors contain 22 numbered pins listed in table 4.2 with respective description of them plus the expected connection. The expected connection is not the original intended connection signal since they were designed for use within Formula Student Team Lisboa (FST Lisboa) championships and have to oblige several protections like automatic cut power in case of water leakage detection inside specific compartments for instance. Majority of those rules, do not apply for our case. As so, all Accumulator insulation relays (AIR) feeds are connected to 24V. Basically AIR feeds act similar to a series of safety interrupters that must be active to provide high voltage but in this case is bypassed. Description of pins were provided by FST Lisboa members.

**⚠️ The CAN connections are not internal terminated with the  $120 \Omega$  resistor.**

#### 4.1.2 Power requirements

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

#### 4.1.3 Enabling power

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there

Group	Pin Number	Description	Cable color	Value
Supply	1	GND	yellow	GND
	2	VCC_A1	white	24V
	3	Not used		
AIR	4	AIR	white	24V
	5	AIR	white	24V
AIR Supply	6	GND	yellow	GND
	7	VCC_AIR	white	24V
Fans (not used in this version)	8	GND	yellow	GND
	9	VCC_AIR	white	24V

Pins 10 to 19 are not used

CAN	20	CAN_L	Black	CAN Low
	21	CAN_H	Black	CAN High
	22	Not used		

Table 4.2: Main battery low voltage connections description

a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

#### 4.1.4 List of CAN messages

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

## 4.2 Auxiliary battery pack

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

# Chapter 5

## Sensors

In this chapter will be briefly explained several notations used along the report and important definitions that help the reader to walk through the work. Many parts of the text presented here are extracted from [15] where full description is available and user should refer to it, if necessary. The notation system of leading superscripts and subscripts is used to denote relative frames orientation or general physical quantities (vectors, points). For frames orientations, leading subscript refers to the frame being represented with respect to the frame in leading superscript. For example let  $R$  be a rotation matrix. Using the notations stated before,  ${}_b^a R$  describes orientation of frame  $b$  with respect to frame  $a$ . For physical quantities, a vector is represented in the frame defined by is leading superscript,  ${}^a v$  and in similar way points follow the same rule,  ${}^a P = [{}^a x, {}^a y, {}^a z]$ .

### 5.1 Definition of frames

The frames cited are Earth Centered, Earth Fixed (ECEF), World Geodetic System (WGS84), East North Up (ENU) and body frame. The ECEF, WGS84 are just auxiliary frames used to define the ENU frame which will be considered the world frame.

#### 5.1.1 ECEF and ENU Frame

ECEF coordinate system defines a referential axis where the origin is defined as the center of Earth, X axis is defined through the intersection of the plane defined by zero latitude line (Equator) and plane defined by zero longitude line (prime meridian). The X-axis orientation is considered positive from center towards the point defined by zero latitude and zero longitude. Z axis is defined by line intersecting origin and both Poles, being positive towards North Pole. Y axis is perpendicular to the plane defined by X and Z axis and its positive direction is defined by right hand rule.

ENU coordinate system is a local coordinate system where the origin is located at a user defined point in ECEF coordinate system, with Y axis pointing towards North Pole and X axis pointing towards East. The plane defined by X and Y axis is tangent to the WGS84 frame on the origin of ENU. Z axis

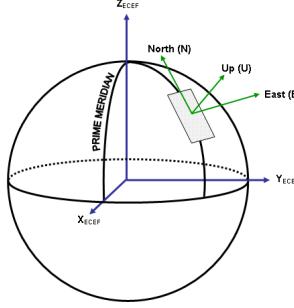


Figure 5.1: ECEF frame and Local ENU frame ([source:wikipedia](#))

express the altitude from defined local plane (see figure 5.1). The ENU frame is considered in this work as the reference frame and will be denoted with superscript or subscript  $w$ .

Given a point of reference in ECEF frame, it is necessary to find the corresponding latitude and longitude of reference point  $(X_r, Y_r, Z_r)$ . To do it is necessary to use the parameters of WGS84 presented in the table 5.1

WGS 84 Defining Parameters[16]		
Parameter	Notation	Value
Semi-major axis	$a$	6 378 137.0 m
Reciprocal of flattening	$1/f$	298.257 223 563
Semi-minor axis	$b$	6 356 752.3142 m
First eccentricity squared	$e^2$	6.694 379 990 14x10 <sup>-3</sup>
Second eccentricity squared	$e'^2$	6.739 496 742 28x10 <sup>-3</sup>

Table 5.1: WGS84 parameters necessary to transform ECEF coordinates into ENU

Using set of equations (5.1) to estimate the latitude  $(\lambda_r)$  and longitude  $(\varphi_r)$  for the reference coordinate point. The final transformation results in applying (5.2) to ECEF physical quantities [17].

$$p = \sqrt{X_r^2 + Y_r^2} \quad (5.1a)$$

$$\theta = \arctan \left( Z_r \frac{a}{pb} \right) \quad (5.1b)$$

$$\lambda_r = \arctan2(Y_r, X_r) \quad (5.1c)$$

$$\varphi_r = \arctan \left( \frac{Z_r + e^2 b \sin^3(\theta)}{p - e'^2 a \cos^3(\theta)} \right) \quad (5.1d)$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{ENU} = \begin{bmatrix} -\sin(\lambda_r) & \cos(\varphi_r) & 0 \\ -\sin(\varphi_r) \cos(\lambda_r) & -\sin(\varphi_r) \sin(\lambda_r) & \cos(\varphi_r) \\ \cos(\varphi_r) \cos(\lambda_r) & \cos(\varphi_r) \sin(\lambda_r) & \sin(\varphi_r) \end{bmatrix} \begin{bmatrix} X - X_r \\ Y - Y_r \\ Z - Z_r \end{bmatrix}_{ECEF} \quad (5.2)$$

### 5.1.2 Body frame, $b$

Each sensor present in the razor board is aligned to match the sensor axes printed in the board as seen in figure 5.5. For simplicity, the razor sensor is placed as possible near the middle point of the

bisector segment between the two wheel axes in such a away that YY axis as marked in the figure 5.5 is pointing towards the front of vehicle, XX axis is pointing to the right side of the car and ZZ axis is pointing to the top. This way, if the Euler angles describing the orientation of body frame related to world frame are all equal to zero, it means the axes in each frame are coincident apart from an offset in origin. Rotation angles are considered positive following the right hand rule in each axis. The origin of body frame is equal to intersection of rear wheel axis with the bisector defined above.

## 5.2 Euler angles and Rotation Matrix

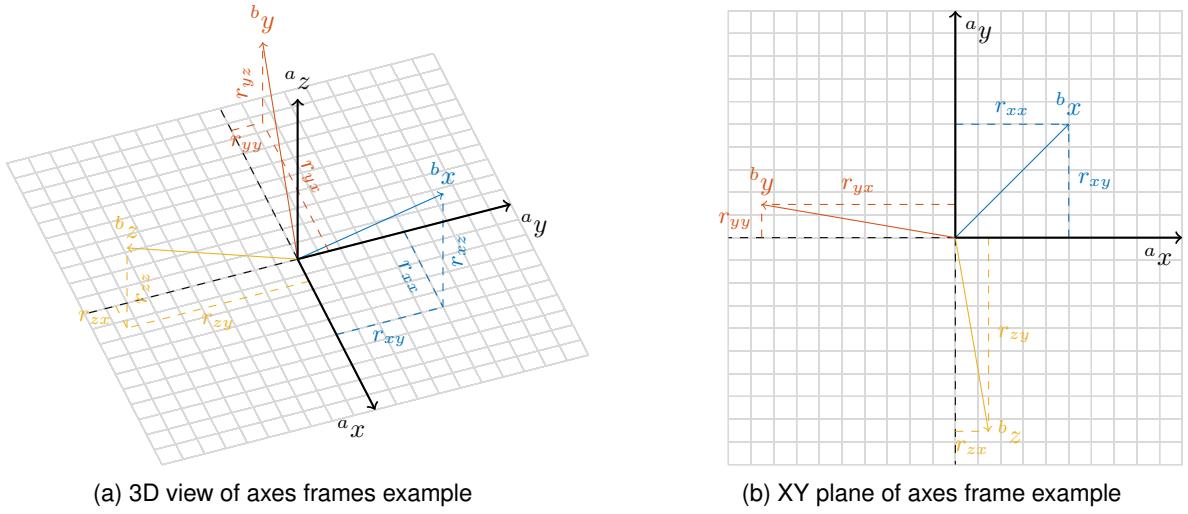


Figure 5.2: Two frames axes example

The rotation matrix is a tool used to describe transformation of coordinates from one frame to another and this also describe orientation of one frame relative to another frame. The convention used in this work is represented by (5.3) and maps quantities described in frame  $b$  to frame  $a$ . Comparing the structure of (5.3) with figure 5.2 it is seen that columns of  ${}^a_b R$  represent each unity vector defining all axes of frame  $b$ .

$${}^a_b R = \begin{bmatrix} {}^a r_{xx} & {}^a r_{yx} & {}^a r_{zx} \\ {}^a r_{xy} & {}^a r_{yy} & {}^a r_{zy} \\ {}^a r_{xz} & {}^a r_{yz} & {}^a r_{zz} \end{bmatrix} \quad (5.3)$$

Rotation matrices belong to the orthonormal group and one important property is that  ${}^a_b R {}^b_a R^T = I$  meaning  ${}^a_b R^T = {}^a_b R^{-1}$ . Also  ${}^a_b R^T$  is equal to  ${}^b_a R$  [18][19].

Euler angles are another form to describe orientation of one frame relative to another by defining three angles. The Euler angles convention adopted in this work is the Tait–Bryan intrinsic rotation sequence Z-Y-X, meaning referential  $a$  axes, represented in figure 5.2, can be mapped into referential  $b$  by performing sequential rotations, first along ZZ axis by an angle  $\psi$ , second along the resulting YY axis by an angle  $\theta$  and final rotation along the resulting XX axis by an angle  $\phi$ . In figure 5.3 is represented the sequence necessary to rotate frame  $a$  into frame  $b$  with the respective Euler angles to achieve the

same orientation as expressed in figure 5.2. It is also represented the intermediary axes resulting from each sequential rotation and its positive direction. Each individual rotation is expressed by its own rotation matrix using the respective Euler angle associated and the final orientation is the result of successive matrices multiplications as shown in equation 5.4 where the intermediary axes sequence is denoted as expressed in the figure 5.3.

$$\begin{aligned} {}_a^b R &= {}_2^b R_x(\phi) {}_1^2 R_y(\theta) {}_a^1 R_z(\psi) \\ &= \begin{bmatrix} \cos(\psi) \cos(\theta) & \sin(\psi) \cos(\theta) & -\sin(\theta) \\ \cos(\psi) \sin(\theta) \sin(\phi) - \sin(\psi) \cos(\phi) & \cos(\psi) \cos(\phi) + \sin(\psi) \sin(\theta) \sin(\phi) & \cos(\theta) \sin(\phi) \\ \sin(\psi) \sin(\phi) + \cos(\psi) \sin(\theta) \cos(\phi) & \sin(\psi) \sin(\theta) \cos(\phi) - \cos(\psi) \sin(\phi) & \cos(\theta) \cos(\phi) \end{bmatrix} \quad (5.4) \end{aligned}$$

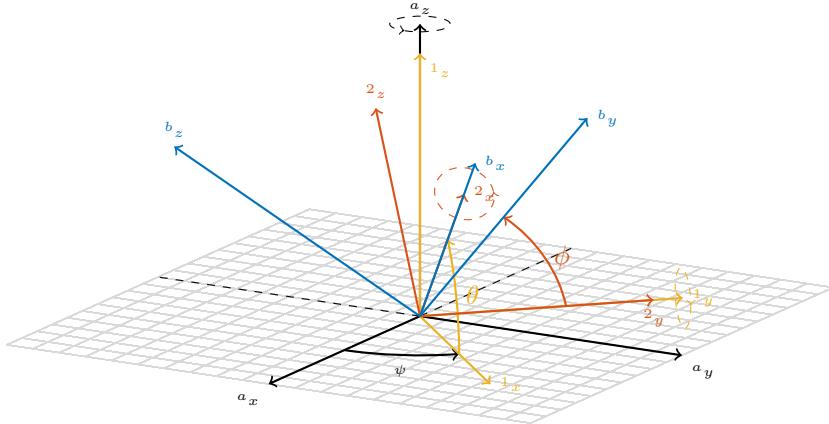


Figure 5.3: Two frames axes example - Euler angles. In this case, the angles are  $\psi = 45^\circ$ ,  $\theta = -45^\circ$ ,  $\phi = 45^\circ$

### 5.3 Quaternion

During the work an Attitude and Heading Reference System (AHRS) algorithm based on [20] and derived in [15] is implemented and it uses quaternions. Basic definitions for understanding the filter terminology are explained in this section following the same notation as present by [15] in case the user has no notion of quaternions at all.

Quaternion is a complex number in four dimension that can be used, similar to rotation matrices and Euler angles, to represent orientation of frames with respect to others. The Euler rotation theorem and the Rodriguez formula are the basis for quaternion representation of rotations but will not be discussed. The theorem states it is possible to describe an orientation of frame relative to other by performing a rotation of  $\alpha$  along an axis  $r$  that is defined by the points that remain static relative to the original frame. [21] quickly resumes the main idea about quaternions, why they can be used to express orientations and relation between them and Euler and Rodriguez formulations.

A quaternion can be represented by (5.5) where  $q_1$  is the norm of it,  $q_2, q_3$  and  $q_4$  are complex coordinates with  $i, j, k$  being the axis versors. If a quaternion is normalized, it is denoted with a circumflex

accent as shown in (5.6).

$$\begin{aligned} q &= q_1 + q_2 i + q_3 j + q_4 k \\ q &= [q_1 \ q_2 \ q_3 \ q_4] \end{aligned} \quad (5.5)$$

$$\hat{q} \implies \|q\| = 1 \quad (5.6)$$

Using the same notation as stated before,  ${}^w_b q$  represent the orientation of body frame with respect to world frame. With (5.5) in mind, the following list of properties/operations are summarized in this list:

**Identities** All quaternions multiplications must obey to the following set of properties described in equation 5.7.

$$i^2 = j^2 = k^2 = ijk = -1 \quad (5.7a)$$

$$ij = -ji = k \quad (5.7b)$$

$$jk = -kj = i \quad (5.7c)$$

$$ki = -ik = j \quad (5.7d)$$

**Quaternion multiplication** The quaternions multiplication, denoted by  $\otimes$ , is defined by the Hamilton product as in equation 5.8

$$\begin{aligned} a \otimes b &= [a_1 \ a_2 \ a_3 \ a_4] \otimes [b_1 \ b_2 \ b_3 \ b_4] \\ &= \begin{bmatrix} a_1b_1 - a_2b_2 - a_3b_3 - a_4b_4 \\ a_1b_2 + a_2b_1 + a_3b_4 - a_4b_3 \\ a_1b_3 - a_2b_4 + a_3b_1 + a_4b_2 \\ a_1b_4 + a_2b_3 - a_3b_2 + a_4b_1 \end{bmatrix}^T \end{aligned} \quad (5.8)$$

**Quaternion conjugate** The quaternion conjugate describes the inverse rotation and is defined as equation 5.9.

$${}^w_b q^* = {}^b_w q = [q_1 \ -q_2 \ -q_3 \ -q_4] \quad (5.9)$$

**Vector rotation** Let's define the following quaternion representation of the same vector but in each referential by using is pure quaternion as  ${}^w v = [0 \ {}^w x \ {}^w y \ {}^w z]$  and  ${}^b v = [0 \ {}^b x \ {}^b y \ {}^b z]$ . The rotation of vector  $v$  from one frame to other, using a quaternion, is performed by equation 5.10.

$${}^b v = {}^w_b \hat{q} \otimes {}^w v \otimes {}_b^w \hat{q}^* \quad (5.10)$$

**Composed rotations** The composition of rotations can be described in quaternions as the product between quaternions. For example the sequence  ${}^a \hat{q} \rightarrow {}^b_c \hat{q}$  is equal to  ${}^a_c \hat{q}$  and is defined as equation 5.11.

$${}^a_c \hat{q} = {}^b_c \hat{q} \otimes {}^a_b \hat{q} \quad (5.11)$$

## 5.4 Novatel OEM4-G2L FlexPak



(a) Novatel OEM4-G2L FlexPak receiver.

(b) Novatel GPS-701-GG antenna

Figure 5.4: Novatel GNSS devices used ([source:NovAtel](#))

Each Novatel GPS sensor is composed with Novatel OEM4-G2L FlexPak receiver and Novatel high performance GPS-701-GG antenna. The Novatel OEM4-G2L FlexPak receiver provides position and velocity estimations using GPS broadcasted satellites radio signals. It has two communications ports that can be configured independently. Additional to common GPS protocols, it has support for custom Novatel protocols (which will be used) providing the best possible estimation the receiver can do making it as simple as possible from the point of view of user. The receiver can handle both L1 and L2 signals provided by satellites [22], however, antenna module only can handle L1 frequency signal (1575.42MHz)[23]. Power supply to FlexPak enclosure should range from 6 to 18 volts DC with typical consumption of 5W. Without differential GPS methods, the Novatel receiver can achieve up to 1.8 meters precision Circular error probable (CEP) with single point operation. Output logs data rate is up to 20Hz in binary mode and 10Hz in ASCII format.

Since the receivers used are industrial grade, no particular effort is made to modeling any error source or trying to improve the solution given by it. However is important to have minimal knowledge the limitations of those types of receivers. It is trusted that the reported position and velocity information is the best the sensor could possibly estimate.

The developed library interface full documentation is present in the attach appendix A, however is advised to use the online version for granting the most recent [updated documentation](#).

### 5.4.1 GNSS main errors

The main cause of Global Navigation Satellite System (GNSS) calculations errors are presented in table 5.2. While some can't be controlled by user (satellite clocks, orbit clocks), others can be mitigated using differential GNSS techniques, multi-frequency (currently civilians have access to more than one), satellite based augmentations system and multi-constellation or using modeling the error source. Multipath is probably the most user dependent but in some situations hard to avoid for example in streets surround with high buildings general denoted as urban canyons.

Source	Value (up to)
Satellite clocks	$\pm 2\text{m}$
Orbit Errors	$\pm 2.5\text{m}$
Inospheric Delays	$\pm 5\text{m}$
Tropospheric Delays	$\pm 0.5\text{m}$
Receiver Noise	$\pm 0.3\text{m}$
Multipath	$\pm 1\text{m}$

Table 5.2: Main source of errors in calculations using GNSS [24]

## 5.5 Sparkfun Razor IMU 9DOF

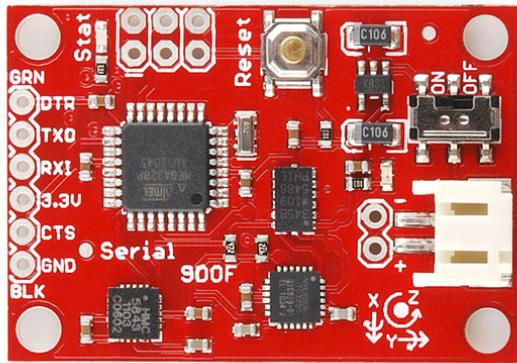


Figure 5.5: Razor 9DOF IMU.

Razor IMU 9DOF is an electronics board, developed by [Sparkfun Electronics](#), which includes an accelerometer, a gyroscope and a magnetometer, each one with three axis of sensibility. Since this board is based on the [Arduino project](#), the chip sensors are interconnected by an Atmel® ATmega328p Microcontroller Unit (MCU) which allows the configuration of sensors, handles the readings and processing of their respective outputs.

### 5.5.1 ADXL345 Digital Accelerometer

The ADXL345 is a Microelectromechanical systems (MEMS) sensor with three axis accelerometer made by [Analog Devices](#). It has an adjustable acceleration scale range value up to  $\pm 16g$ . It is possible to measure both dynamic accelerations resulting from motion and shock or measure static accelerations such as gravity, which allows this device to be used as a tilt sensor also. Acceleration deflects the a proof mass attached to one differential capacitor plate and unbalances it, resulting in a sensor output whose amplitude is proportional to acceleration [25]. Conversion of outputs is made through 10 to 13 bit Analog to Digital Converter (ADC) according to selected range to maintain the same scale of 4mg/LSB.

### 5.5.1.1 Accelerometer sensor model

The output of accelerometer is proportional to the sum of acceleration in each axis. However the accelerometer is not capable of distinguishing if the acceleration is caused by a true acceleration produced by motion or produced by a static force, generally gravity force. As so, equation (5.12) describes the general output of accelerometer [26] where,  ${}^b A_{ext}$  is the sum of real external acceleration due to linear or rotational dynamics in the body frame,  ${}^w R$  is the rotation matrix mapping quantities in world frame into sensor frame,  ${}^w g$  represents the fictitious acceleration due to gravity in the world frame,  ${}^b A_0$  is an offset and  $\delta A_e$  describes additive gaussian noise. Ideally,  $G$  should be equal to identity matrix but in fact it represents the product of two matrices, one describing the cross-axis influence and the other a scale factor for each axis [26].

$${}^b A_s = G[{}^b A_{ext} + {}^w R {}^w g] + {}^b A_0 + \delta A_e \quad (5.12)$$

### 5.5.1.2 Accelerometer calibration

The suggested calibration method by the manufacturer [27] assumes that cross-axis influence is small enough and can be neglected, this way,  $G$  should only be represented by a diagonal matrix with the scale factors for each axis. However, if more precision for the application is needed, calibration using the ellipsoid fitting approach should be made [28].

For this application it will be used the suggested calibration by manufacturer resulting in exposing the sensor to six position combination while resting. This means the  ${}^b A_{ext}$  will be zero and accelerometer will be only influenced by the gravity force, 1g.

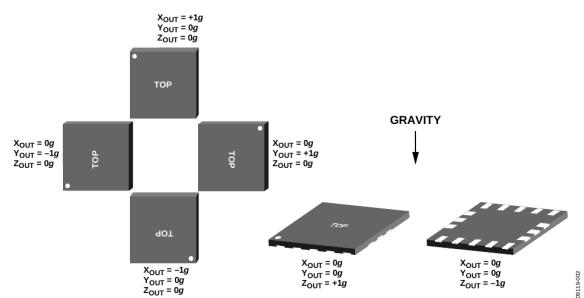


Figure 5.6: ADXL345 calibration poses and expected output [27].

Gains and offset [LSB]	Razor
Z gain	253.66
Z offset	-3.47
Y gain	266.27
Y offset	-3.18
X gain	266.04
X offset	13.83

Table 5.3: Summary of values extracted and calculated for each axis of accelerometer in both sensors

Aligning one axis at a time with gravity vector, as seen in figure 5.6, the mean value of several samples is calculated. Using these means for each axis and the known acceleration values( $\pm 1g$ ), it is calculated the slope and offset of the respective line passing through the two points. Calculated values (see table 5.3) will be stored in the internal EEPROM memory of MCU to be loaded at boot time of each razor. The conversion between LSB and g unities is given by the relation  $254LSB/g$  or  $3.9mg/LSB$  [25]

## 5.5.2 ITG3200 Digital Gyroscope

The ITG-3200 is a microelectronic mechanical integrated circuit chip with three axis independent gyroscopes, built by [Invensense](#) able of measuring angular velocities up to  $\pm 2000^\circ /s$ . When the gyroscopes are rotated about any of the sensor axes, the Coriolis effect causes a deflection that is detected by a capacitive pick-off circuit proportional to angular velocity. The measured signal is filtered and converted using internal ADC. The scale factor that allow to convert the output values to angular velocities in degrees per second is, according to its datasheet [29], factory calibrated and equal to  $14.375 LSB/\text{ }^\circ \text{.s}^{-1}$ .

### 5.5.2.1 Gyroscope sensor model

For a given axis, the gyroscope output is proportional the angular velocity sensed of that axis and is given by equation (5.13) where  ${}^b\omega_s$  represents the vector output of the sensor in body frame at instant  $t$ ,  ${}^b\omega_{real}$  is the real angular velocity vector applied to sensor in body frame,  ${}^b\omega_0$  is the bias vector term and  $\delta\omega_\epsilon$  additive gaussian noise. Since the sensor is described in [29] as factory calibrated,  $G$  which represent a scaled factor correction, is expected to be equal to identity matrix. Note that the equation model (5.13) is in fact a simplified version of a more accurate model described in [26]. As stated in [26] cross-axis misalignment and linear acceleration sensitivity are expected to be small for the application purpose so they are considered irrelevant.

In general, the offset value depends on the temperature value inside chip. Once internal stability is reached, offset values tends to be constant [30]. Electrical temperature stability can be achieved after a few minutes of operation. At the beginning of initialization of Razor, several samples are acquired internally in the microprocessor and mean value is calculated for each axis, defining this way the offset constant for each axis.

$${}^b\omega_s = G {}^b\omega_{real} + {}^b\omega_0 + \delta\omega_\epsilon \quad (5.13)$$

## 5.5.3 HMC5843 Digital compass

The HCM5843 sensor is a digital compass chip designed by [Honeywell](#), designed for low field magnetic sensing. This sensor uses anisotropic magnetic resistor technology, making it immune to vibration noises, and the use of Wheatstone bridge makes it more robust to noise. A material with anisotropic magnetic resistor properties changes its resistance according to absolute value and direction of magnetic field.

### 5.5.3.1 Magnetometer model

The magnetometer sensor model follows a similar structure as two previous sensors and is given by equation (5.14). The  ${}^bH_s$  is the value sensed by the sensor in body frame.  $C$  is a matrix resulting from the multiplication of matrices containing influence of cross-axis, gain scale factor and soft-iron interference [31].  ${}^w h_e$  represent the geomagnetic vector of Earth,  ${}^b h_{offset}$  is an offset vector which describes the

influence of zero-field offset and the ferromagnetic masses fixed relative to body frame usually denoted as hard-iron.  $\delta h_\epsilon$  represents gaussian additive noise.

$${}^b H_s = C_w {}^b R^w h_e + {}^b h_{offset} + \delta h_\epsilon \quad (5.14)$$

### 5.5.3.2 Geomagnetic field

The geomagnetic field can be described as 3D vector in each point as seen in [32]. As previous defined,  ${}^w h_e$  represent the geomagnetic vector of Earth. Let us define now,  ${}^w h_0$  as the horizontal projection of  ${}^w h_e$ . The angle between  ${}^w h_0$  and  ${}^w h_e$  is denominated as inclination,  $I$ . As stated in [32], the horizontal projection of the geomagnetic vector points to the magnetic north which may not be aligned with the north axis of reference frame,  ${}^w Y$ . This defines the magnetic declination angle,  $D$ , as the angle between horizontal projection and the  ${}^w Y$  axis.

For Lisbon, the magnetic declination angle, in January 2017, defined in the reference frame is equal to  $2.48^\circ$ , inclination angle is equal to  $52.77^\circ$  and the  ${}^w h_e$  is equal to  $[-1121.5, 26479.8, -34884.7]^T$  nanoteslas [33]

### 5.5.3.3 Hard and soft-iron compensation

The compensation for hard and soft-iron effects can be done using a geometric approach as described by [31] [34] [35]. If the geomagnetic vector is known and sensor perfect calibrated, free of any type ambient distortion, points collected are expected to belong a surface of an origin centered sphere with radius equal to the magnitude of geomagnetic field  ${}^w h_e$ . Distortions, cross-axis influence, scale-factor inequalities between axis, causes the expected sphere to be deformed into a ellipsoid. Collecting points in 3D space by performing rotations of the sensor frame, allow us to fit the data to an ellipsoid surface and after determinate the matrix  $C$  and offset vector  ${}^b h_{offset}$ .

However since the body frame is a vehicle, it is not physically possible to collect points based in rotations of YY and XX axis. Only ZZ axis rotations are available and the data instead of belonging to an ellipsoid surfaces, will belong to an ellipse in Z-plane as a result from that plan cutting the ellipsoid in some undetermined z coordinate. Because of restriction stated before it will be considered that the influence in ZZ axis is zero and the scale factor is equal to one. Offset in ZZ axis will also be considered zero. In [15] is shown that this assumption is not relevant. (5.14) will be rearranged to equation (5.15) form.

$$\begin{aligned} {}^b H_s &= C_w {}^b R^w h_e + {}^b h_{offset} + \delta h_\epsilon \\ {}^b_w R^w h_e &= C^{-1} [{}^b H_s - {}^b h_{offset} - \delta h_\epsilon] \end{aligned} \quad (5.15)$$

Without loss of generality,  $C^{-1} \delta h_\epsilon$  will result also in a gaussian vector so it will be rewritten as  $\delta h_\epsilon$ . Equation (5.15) can take now be arranged into (5.16) where  ${}^b H_c$  is the corrected value after calibration process.

$$\begin{aligned} {}^b_w R(t)^w h_e &= C^{-1} [{}^b H_s - {}^b h_{offset} - \delta h_\epsilon] \\ {}^b H_c &= C^{-1} [{}^b H_s - {}^b h_{offset}] + \delta h_\epsilon \end{aligned} \quad (5.16)$$

The matrix  $C^{-1}$  and  ${}^b h_{offset}$  is obtained as described by [34] using the *ellipse\_fit*<sup>1</sup>. The function *ellipse\_fit* returns the least square estimate that best fits the data collected from sensor and the parameters that define the estimated ellipse as described in this list:

- **semi-major** - The length of major semi axis of ellipse.
- **semi-minor** - The length of minor semi axis of ellipse.
- $x_0$  - The offset in the XX axis of ellipse.
- $y_0$  - The offset in the YY axis of ellipse.
- $\alpha$  - Rotation angle between semi-major axis and XX axis in reference frame.

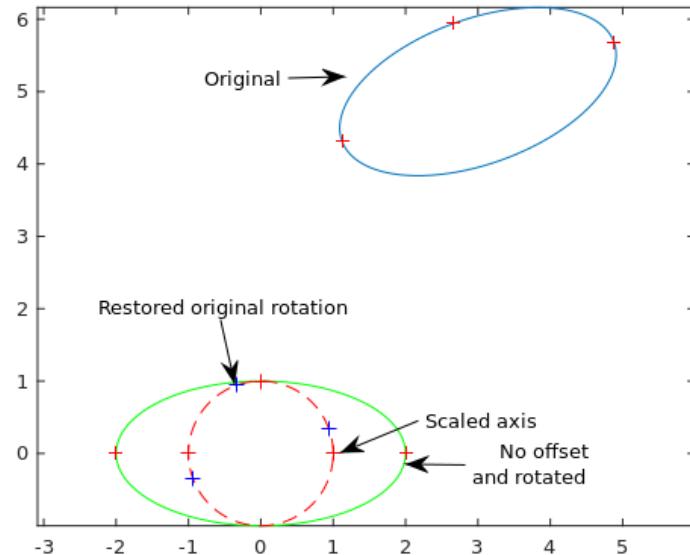


Figure 5.7: Simulation of the calibration steps applied to hypothetical ellipse data.

Take note the fact that since is not physically possible to perform rotations in 3D space, the calibration will only be reflect the XX and YY axis. Using the values returned, the calibration process consists in the following sequential steps:

- ① Remove offset by using equation (5.17) with  ${}^b h_{offset} = [{}^b x_0, {}^b y_0, 0]^T$

$${}^b H_s = {}^b H_s - {}^b h_{offset} \quad (5.17)$$

- ② Align semi-major axis of ellipse with the XX axis of reference frame using a rotation of  $-\alpha$  by using equation (5.18)

---

<sup>1</sup><https://www.mathworks.com/matlabcentral/fileexchange/22423-ellipse-fit>

$$\begin{aligned}
{}^b H_s &= R_{cal} {}^b H_s \Leftrightarrow \\
\Leftrightarrow {}^b H_s &= \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} {}^b H_s
\end{aligned} \tag{5.18}$$

- ③ Apply a scaling matrix  $S$  to make semi-major axis the same length as semi-minor axis using (5.19)

$$\begin{aligned}
{}^b H_s &= S {}^b H_s \Leftrightarrow \\
\Leftrightarrow {}^b H_s &= \begin{bmatrix} \frac{\text{semi-minor}}{\text{semi-major}} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} {}^b H_s
\end{aligned} \tag{5.19}$$

- ④ Restore the initial rotation  $\alpha$  to the scaled data using the transpose of  $R_{cal}$  as seen in (5.20)

$$\begin{aligned}
{}^b H_s &= R_{cal}^T {}^b H_s \Leftrightarrow \\
\Leftrightarrow {}^b H_s &= \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} {}^b H_s
\end{aligned} \tag{5.20}$$

The  ${}^b H_s$  resulting from step four is in fact the expected corrected reading of sensor,  ${}^b H_c$ . In figure 5.7 visually represented the steps applied to a hypothetical ellipse. Resuming, all sequential steps in one equation results in (5.21).

$${}^b H_c = R_{cal} S R_{cal}^T [{}^b H_s - {}^b h_{offset}] \tag{5.21}$$

Comparing the structure of (5.21) with (5.16) this implies that  $C^{-1}$  is equal to  $R_{cal} S R_{cal}^T$ .

The results of calibration are presented for one of the devices in figure 5.8 and the obtained are expressed in the table 5.4. The trajectory used is not important as long as it is able perform one or more rotation of the vehicle in the horizontal plan. In figure 5.8c is shown the results before and after calibration for the estimation of Euler  $\psi$  angle. It is also present the estimation of the same angle using the the one of the gps unities as a reference for the comparison.

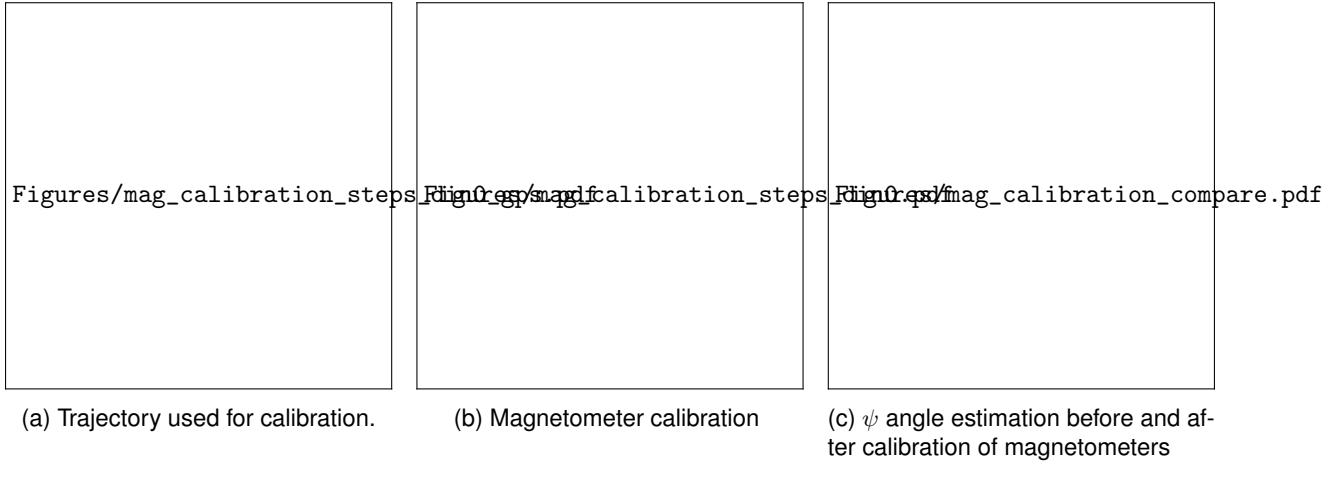


Figure 5.8: Magnetometer calibration steps applied to real data

<b>Razor 1</b>		
${}^b h_{offset}$	[nT]	$[-2.5828 \ -2.1703]^T \times 10^4$
$\alpha$	[deg]	107.5238
$\frac{semi-minor}{semi-major}$		0.9567
$R_{cal}$		$\begin{bmatrix} -0.3011 & -0.9536 & 0 \\ +0.9536 & -0.3011 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
$C^{-1}$		$\begin{bmatrix} +0.9961 & +0.0124 & 0 \\ +0.0124 & +0.9606 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Table 5.4: Magnetometers calibration results.

# Chapter 6

## Steering

The control of steering wheel take advantage of motor for steering assist originally present in the car. The motor is controlled by a positioning controllers designed for brushed DC and brushless DC motors with encoders. denominated Easy Positioning System (EPOS) by many manufactures

### 6.1 Maxon EPOS 70/10 controller



Figure 6.1: Maxon EPOS 70/10 controller

The controller available to use is the [Maxon EPOS 70/10](#), from now on denoted as EPOS. It includes CAN network interface and a RS-232. The relevant electrical specs can be seen in table 6.1. The full specifications as well as connections layout must be seen in [36]. The controller contains two led in a single package that is used as status reference. Table 6.2 present the status of device based on pattern of leds shown by it. The EPOS will be used as a Proportional Integral Derivative (PID) controller for the steering wheel.

### 6.2 Steering sensor support

Maxon EPOS 70/10 accepts quadrature sensors to give feedback of position. The used sensor is a quadrature line driver with index (although index is disabled because its track is damaged). The sensor model is HEDR-55L2-BY09 with main characteristics shown in table 6.3 [37]. Table 6.4 show

the required configuration to be passed to EPOS device to correctly use the steering controller. See appendix B for further description.

The sensor support is mainly 3D printed in ABS filament. It is composed of the following parts

Description	Min.	Max.	Unity
Power supply voltage $V_{CC}$ (Ripple <10%)	11	70	$V_{DC}$
Max. output voltage		$0.9 \cdot V_{CC}$	$V_{DC}$
Max. output current $I_{max} (<1sec)$		25	A
Continuous output current $I_{cont}$		10	A
Sample rate PI - current controller	10K	10K	Hz
Sample rate PI - speed controller	1K	1K	Hz
Sample rate PI - positioning controller	1K	1K	Hz
Max. speed (motors with 2 poles)		25K	rpm

Table 6.1: EPOS electric specifications

Description	Red	Green
The EPOS is in state:		
<ul style="list-style-type: none"> <li>• Switch ON Disabled</li> <li>• Ready to Switch ON</li> <li>• Switched ON</li> <li>• The power stage is disabled</li> </ul>	OFF	Blinking ( $\approx 1\text{Hz}$ )
The EPOS is in state:		
<ul style="list-style-type: none"> <li>• Operation Enable</li> <li>• Quick Stop Active</li> <li>• The power stage is enabled</li> </ul>	OFF	ON
EPOS is in Fault State	ON	OFF
EPOS is in temporary state Fault Reaction Active	ON	ON
There is no valid firmware on the EPOS (due to a failed firmware download)	ON	Flashing

Table 6.2: Maxon EPOS 70/10 Led status

Description	Value
Line Driver	Yes
Counts per revolution	3600
Total number of positions	14400
Shaft diameter	8mm

Table 6.3: Main HEDR-55L2\_BY09 sensor characteristics

Description	Value
Sensor Type	2
Pulse Number	3600
Sensor Polarity	0

Table 6.4: Quadrature sensor settings configured in EPOS

presented in table 6.5. A dimensional sketch of the sensor and bearing used was also designed just for auxiliary purposes and to provide assembly instructions. The drawings with respective dimensions are added in appendix C. The parts were designed in Autodesk Fusion 360 and are available under the Github repository. An visual assembly guide is present [here](#) to enlighten the user. The 3D model and real model assembled in the car are present in figure 6.2 and 6.3

The designed files can be download from [here](#)

Part name	Quantity	Reference design
608 bearing	1	Bearing Diagram
Sensor Case HEDR-55L2_BY09	1	Sensor case
Half Gear 54 teeth	2	Half gear
Sensor gear 32 teeth	1	Sensor gear

Table 6.5: Sensor support parts

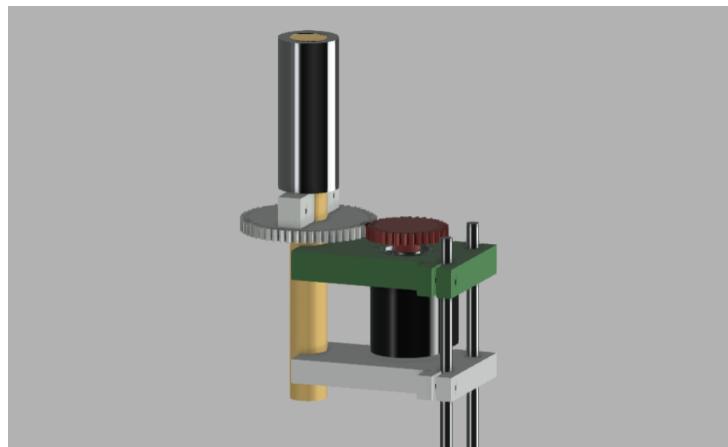


Figure 6.2: Support sensor 3D in design software



Figure 6.3: Support sensor assembled

### 6.3 Calibration process

Here is assumed the vehicle follows a model of Ackermann steering model typical valid for low speed which result in the simplified approximation of the single line model or as in general described in literature, as bicycle model[38] [39]. A simplified representation of the model is present in figure 6.4. This model is normally used in control applications and expected to be used in future high level control. As it is necessary to determine the angle of wheel with respect to body frame YY axis (recall subsection 5.1.2), the first calibration process required is to determine the function relating steering wheel position and angle of fictional wheel. A set up was made fixing the car and placing a white duck tape in front of it, perpendicular as physically possible and apart by 1.47 meters in front of the car. The duck tape will serve as a ruler. In each wheel is attached a laser level, pointing to front duck tape. Several measurements are taken for different steering position values and using trigonometric relations, the angles of each wheel is estimated.

The other calibration process is taken each time a power up is made. It consists in performing a full rotation of the steering wheel to each extreme points. During this movement, the program is always reading sensor position and updating the maximum and minimum value read. After these quadrature counter values are found, user is requested to end calibration and the offset are calculated and stored along with maximum and minimum position. These values will only change in case of shut down performed on steering controller device.

Taking figure 6.5 that represents the calibration set up made, the bold line between points  $max_L$  and  $max_R$  represent the our ruler. The  $d_1$  represent the distance between outer of each wheel where the lasers are attached. The relevant distances using in this step up are present in table 6.6.

Description	Measurement [cm]
$d_1$	144.5
$L_1$	124
$L_2$	49.5
$d_L$	45.5
$h_1$	147
$max_L$	$L_1 + L_2 + d_L = 219$
$max_R$	$L_1 + L_2 + d_L = 219$

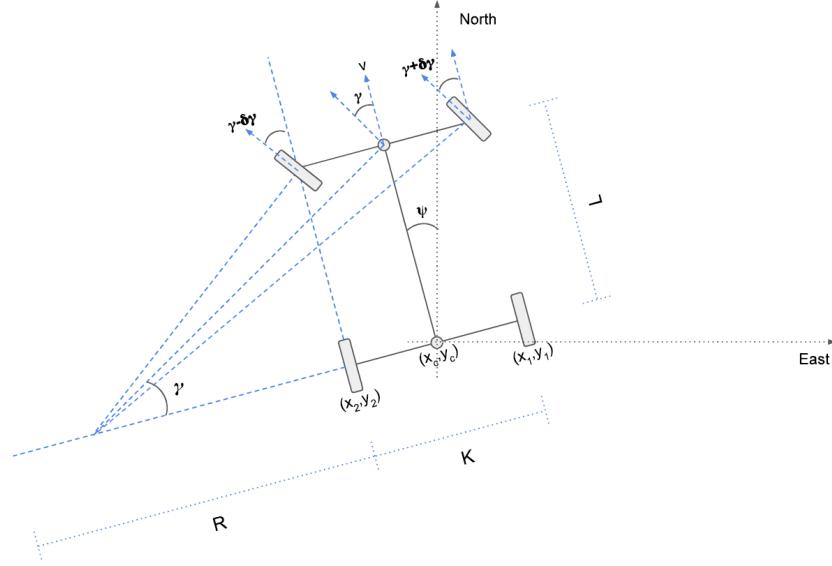
Table 6.6: Reference measurements for calibration set up scheme

Comparing figure 6.4 with figure 6.5 results in (6.1). Then is assumed that:

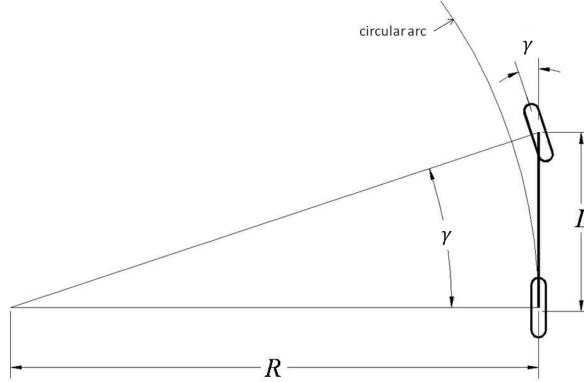
- $\gamma$  must be a first order function.
- $\delta\gamma$  must be always great or equal to zero and therefore, quadratic.
- $\beta$  and  $\alpha$  should be, at least a second order function with opposite concavities orientation.

$$\begin{cases} \beta = \gamma + \delta\gamma \\ \alpha = \gamma - \delta\gamma \end{cases} \quad (6.1)$$

Taking several measurements for different values of quadrature counters, is calculated the respective



(a) Four wheel representation



(b) Bicycle Representation (source [38])

Figure 6.4: Ackermann steering model simplification

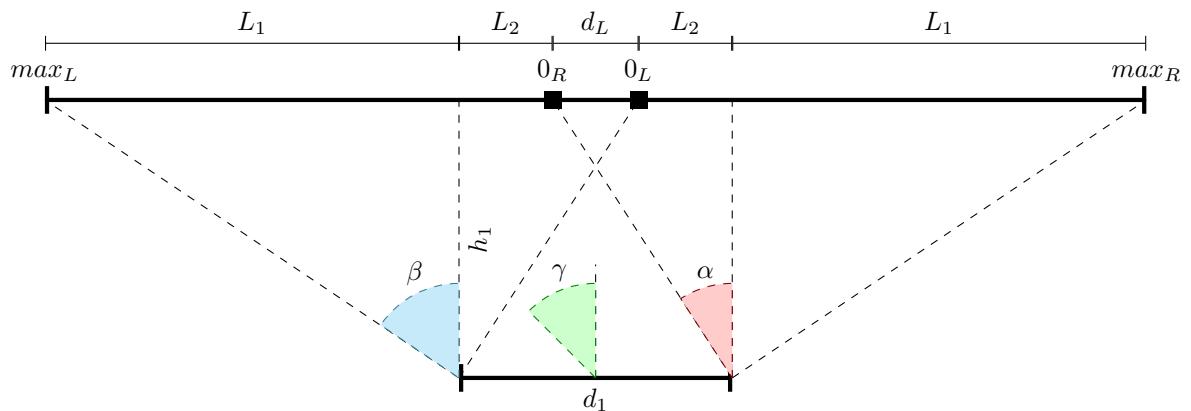


Figure 6.5: Calibration set up scheme

values of  $\alpha$  and  $\beta$ . After, for each of those pairs is calculated the estimated values of  $\delta\gamma$  and  $\gamma$ . Using least squares approach (polyfit from Matlab®) is estimated the best fitting for each variable with respect to Quadrature counters (qc), assuming the discussed points above. The result of fitting is shown in figure 6.6. It shows the evaluated values of fitted functions in the range of -40000 to 40000 qc. Note the

fact that zero value is slightly apart to the right. This is a normal consequence of small misalignment (approximate  $1^\circ$ ) of wheels when the EPOS device is initialized, which is assumed the zero of it. This reflects the requirement to perform the full rotation to each side after device power up, to find the offset at origin.

The estimated ratio between steering wheel qc and the fictional wheel of bicycle model is then  $-7.5014E - 04$ , translating in the equation (6.2), where the offset is determined at each power up calibration step.

The Matlab<sup>TM</sup> used for calibration is present in appendix E

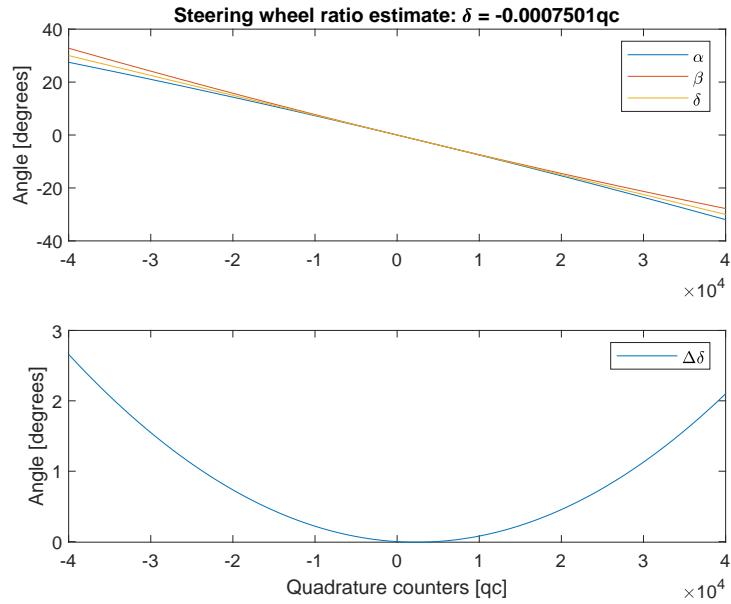


Figure 6.6: Results for Calibration of steering wheel

$$f(qc) = -7.5014E - 04 \times qc + offset \quad (6.2)$$

## 6.4 interface library

The developed interface library for EPOS is present in the same Github repository. The library is made in python language and using the CAN interface to interact with EPOS. User should note that the library still not cover all the possible interactions and modes of operation of EPOS. It is only designed for position control mode. In the same repository are the firmware manual provided by Maxon and they should be consulted for further information. The respective documentation of the library is present on online version [here](#) and also as appendix B. User should always prefer the online documentation to grant is the most updated version.

## 6.5 Support and controller advices

### Caution!

Since the sensor support is 3D printed, care must be taken to avoid abrupt variations of the steering wheel or the sensor gear might brake. User must also be careful to not apply achieve or surpass the extreme positions because it may cause damage to either the structure of car or the motor it self.

The EPOS contains built-in function that disables the operation if difference between the demanded position and current position grows or is higher beyond the predefined maximum following error parameter. These situation can be caused by a badly conditioned reference, if the motor is blocked or if the motor is not able to keep up with the rate of variation requested. A careful reference trajectory generation must be provided by user to avoid damage. During the testing of vehicle in field and development, a series of functions have been developed that can serve as inspiration for future implementations. They are presented in appendix F in the form of a Python class. It may serve as future inspiration for high level design. It contains the following list of base functions:

#### **getQcPosition**

Converts angle of wheels to qc.

#### **getDeltaAngle**

Converts qc of steering wheel to angle of wheel.

#### **saveToFile**

Record qc positions into a csv file.

#### **readFromFile**

Read qc positions from file and follow them

#### **moveToPosition**

Project and plan a trajectory from current position to desired position restricted to low jerk, limited acceleration and speed. The function is an implementation of algorithm presented in [40]

# Chapter 7

## Power Train controller

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

### 7.1 Motor data

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Motor Equivalent Circuit Parameters. (Data from inverter SIEMENS SINAMICS using STARTER. for other Data, check below in Annex ). Motor Nameplate Data

Parameter	Value	Units
Weight	1200	kg
Batteries Weight	400	kg
Autonomy	90	km
Maximum speed	100	km/h
Acceleration (0-50km/h)	8	s
Wheel diameter	60.97	cm
Width	150.8	cm
Height	144.5	cm
Area	2.18	m <sup>2</sup>
Drag Coefficient	0.33	-

Table 7.1: Original car parameters

Parameter	Value	Units
Maximum Power	30	kW
Maximum Torque	130	Nm
Maximum speed	10000	rpm
Weight	41.5	kg
<b>Equivalent circuit parameters</b>		
stator resistance	8.56	mOhms
stator leakage inductance	0.06292	mH
Iron Resistance	$\infty$	mOhms
Mutual Inductance	1.0122	mH
rotor resistance	5.10	mOhms
rotor leakage inductance	0.06709	mH

Table 7.2: Motor parameters

U (V)	f (Hz)	I (A)	Torque (Nm)	Speed (rpm)
76	76	157	65	2200
121	220	90	22	6500
121	305	88	16	9000

Table 7.3: Motor Nameplate parameters

## 7.2 Siemens Sinamics

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

# Bibliography

- [1] Python Software Foundation. Status of python branches, 2018. URL <https://devguide.python.org/#status-of-python-branches>. (2018).
- [2] Numpy Developers. Plan for dropping python 2.7 support, 2017. URL <https://www.numpy.org/neps/nep-0014-dropping-python2.7-proposal.html>. (2018).
- [3] Python Software Foundation. Python 3 documentation, 2018. URL <https://docs.python.org/3/>. (2018).
- [4] Python Software Foundation. Pip 18.0 user guide - requirements file format, 2018. URL [https://pip.pypa.io/en/stable/reference/pip\\_install/#requirements-file-format](https://pip.pypa.io/en/stable/reference/pip_install/#requirements-file-format). (2018).
- [5] Google Python Team. Google python style guide, 2018. URL <https://github.com/google/styleguide/blob/gh-pages/pyguide.md#38-comments-and-docstrings>. (2018).
- [6] Raspberry Pi Foundation. Installing operating systems images, April 2018. URL <https://www.raspberrypi.org/documentation/installation/installing-images/README.md>. (2018).
- [7] Wikipedia. Mac address, 2018. URL [https://en.wikipedia.org/wiki/MAC\\_address](https://en.wikipedia.org/wiki/MAC_address). (2018).
- [8] Wireshark. Oui lookup tool, 2018. URL <https://www.wireshark.org/tools/oui-lookup.html>. (2018).
- [9] R. Light. Mosquitto.conf, N/A. URL <https://mosquitto.org/man/mosquitto-conf-5.html>. (2018).
- [10] Raspberry Pi Foundation. Setting up a raspberry pi as an access point in a standalone network (nat), February 2018. URL <https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md>. (2018).
- [11] Raspberry Pi Foundation. Setting up an nginx web server on a raspberry pi, Nov 2017. URL <https://www.raspberrypi.org/documentation/remote-access/web-server/nginx.md>. (2018).
- [12] J. Ellingwood. How to set up nginx server blocks (virtual hosts) on ubuntu 16.04, May 2016. URL <https://www.digitalocean.com/community/tutorials/how-to-set-up-nginx-server-blocks-virtual-hosts-on-ubuntu-16-04>. (2018).

- [13] N/A. Nginx getting started, April 2018. URL <https://www.nginx.com/resources/wiki/start/>. (2018).
- [14] D. Pinho. *Electrical System Form FSI 2015, Projecto FST NOVABASE*. FST Lisboa.
- [15] B. Tibério. An online filter study for inertial properties estimation based on low-cost sensors. Msc thesis, Instituto Superior Técnico, November 2017. URL <https://fenix.tecnico.ulisboa.pt/downloadFile/1126295043835604/Thesis.pdf>. (2018).
- [16] Wikipedia. Geodetic datum, 2017. URL [https://en.wikipedia.org/wiki/Geodetic\\_datum#Earth\\_reference\\_ellipsoid](https://en.wikipedia.org/wiki/Geodetic_datum#Earth_reference_ellipsoid). (2017).
- [17] Wikipedia. Geographic coordinate conversion, 2017. URL [https://en.wikipedia.org/wiki/Geographic\\_coordinate\\_conversion#From\\_ECEF\\_to\\_ENU](https://en.wikipedia.org/wiki/Geographic_coordinate_conversion#From_ECEF_to_ENU). (2017).
- [18] J. S. Sequeira. Introdução à robótica. Introductory notes about Robotic concepts and definitions, 2016. URL <http://users.isr.ist.utl.pt/~jseq/caps1-6.pdf>.
- [19] Wikipedia. Euler's rotation theorem, 2017. URL [https://en.wikipedia.org/wiki/Euler%27s\\_rotation\\_theorem](https://en.wikipedia.org/wiki/Euler%27s_rotation_theorem). (2017).
- [20] S. O. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Technical report, University of Bristol, 2010. URL [http://www.x-io.co.uk/res/doc/madgwick\\_internal\\_report.pdf](http://www.x-io.co.uk/res/doc/madgwick_internal_report.pdf).
- [21] J. V. Verth. Understanding Quaternions - Presentation at GDC 2013, 2013. URL [https://www.essentialmath.com/GDC2013/GDC13\\_quaternions\\_final.pdf](https://www.essentialmath.com/GDC2013/GDC13_quaternions_final.pdf). (2017).
- [22] NovAtel. *OEM4 Family - USER MANUAL - VOLUME 1*, 2005. URL <https://www.novatel.com/assets/Documents/Manuals/om-20000046.pdf>.
- [23] NovAtel. Antennas GPS-701-GG GPS-702-GG pinwheel ® antennas enhance flexibility and reduce costs dual constellation for enhanced positioning, 2015. URL [https://www.novatel.com/assets/Documents/Papers/GPS701\\_702GG.pdf](https://www.novatel.com/assets/Documents/Papers/GPS701_702GG.pdf). (2017).
- [24] NovAtel. An introduction to gnss, 2015. URL <https://www.novatel.com/an-introduction-to-gnss/>. (2017).
- [25] *ADXL345 Product Specification*. Analog Devices, Inc, 2015. URL <http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf>. Revision E.
- [26] Vectornav. Vectornav support library - calibration, N/A. URL <http://www.vectornav.com/support/library/calibration>. (2017).
- [27] T. Tusuzki. *ADXL345 Quick Start Guide*. Analog Devices, Inc, June 2010. URL <http://www.analog.com/media/en/technical-documentation/application-notes/AN-1077.pdf>. Rev. 0.

- [28] T. Pylvänäinen. Automatic and adaptive calibration of 3D field sensors. *Applied Mathematical Modelling*, 32(4):575–587, 2008. ISSN 0307904X. doi: 10.1016/j.apm.2007.02.004. URL <http://www.sciencedirect.com/science/article/pii/S0307904X07000297>.
- [29] *ITG-3200 Product Specification*. Invensense, 2011. URL <https://www.invensense.com/wp-content/uploads/2015/02/ITG-3200-Datasheet.pdf>. Revision 1.7.
- [30] O. J. Woodman. An introduction to inertial navigation. Technical report, University of Cambridge, 2007. URL <https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf>.
- [31] T. Ozyagcilar. *Calibrating an eCompass in the presence of Hard and Soft-Iron Interference*. Freescale Semiconductor, Inc, November 2015. URL [http://cache.freescale.com/files/sensors/doc/app\\_note/AN4246.pdf](http://cache.freescale.com/files/sensors/doc/app_note/AN4246.pdf). Rev. 4.0.
- [32] IPMA. Geomagnetismo, 2005. URL <http://www.ipma.pt/pt/enciclopedia/geofisica/geomagnetismo/>. (2017).
- [33] NOAA. Magnetic field calculators, 2017. URL <https://www.ngdc.noaa.gov/geomag-web/#declination>. (2017).
- [34] M. Caruso. Applications of magnetic sensors for low cost compass systems. *IEEE 2000. Position Location and Navigation Symposium (Cat. No.00CH37062)*, pages 1–8, 2000. doi: 10.1109/PLANS.2000.838300. URL <https://cdn.shopify.com/s/files/1/0038/9582/files/lowcost.pdf>.
- [35] J. F. Vasconcelos, G. Elkaim, C. Silvestre, P. Oliveira, and B. Cardeira. Geometric approach to strapdown magnetometer calibration in sensor frame. *IEEE Transactions on Aerospace and Electronic Systems*, 47(2):1293–1306, 2011. ISSN 00189251. doi: 10.1109/TAES.2011.5751259. URL <http://www.dem.ist.utl.pt/poliveira/Invest/05751259.pdf>.
- [36] Maxon Motor, AG. *EPOS 70/10 Hardware Reference*. Maxon Motor, AG, 752380-04 edition, December 2008. URL [https://www.maxonmotor.com/medias/sys\\_master/root/8803613802526/300583-Hardware-Reference-En.pdf](https://www.maxonmotor.com/medias/sys_master/root/8803613802526/300583-Hardware-Reference-En.pdf). (2018).
- [37] Avago. *EPOS 70/10 Hardware Reference*. Avago Technologies, av02-3823en edition, October 2014. URL [https://pt.mouser.com/datasheet/2/678/V02-3823EN\\_DS\\_HEDR-5xxx\\_2014-10-300-909317.pdf](https://pt.mouser.com/datasheet/2/678/V02-3823EN_DS_HEDR-5xxx_2014-10-300-909317.pdf). (2018).
- [38] J. M. Snider. Automatic steering methods for autonomous automobile path tracking, February 2009. URL [http://www.ri.cmu.edu/pub/\\_files/2009/2/Automatic{\\_}Steering{\\_}Methods{\\_}for{\\_}Autonomous{\\_}Automobile{\\_}Path{\\_}Tracking.pdf](http://www.ri.cmu.edu/pub/_files/2009/2/Automatic{_}Steering{_}Methods{_}for{_}Autonomous{_}Automobile{_}Path{_}Tracking.pdf). (2018).
- [39] E. Nebot. Navigation system design - lecture notes, April 2006. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.710.3651&rep=rep1&type=pdf>. (2018).

- [40] H. Z. Li, Z. M. Gong, W. Lin, and T. Lippa. Motion profile planning for reduced jerk and vibration residuals. *SIMTech technical reports*, 8(1):32–37, 2007. doi: 10.13140/2.1.4211.2647. URL <https://www.researchgate.net/publication/267794207>.

## **Appendix A**

# **NovateIOEM4 GPS library**

## **Documentation**

**Date** 24 Jul 2018

**Version** 0.4

**Author** Bruno Tibério

**Contact** bruno.tiberio@tecnico.ulisboa.pt



# CHAPTER 1

---

## Changelog

---

**version 0.4** Moved from optoparse to argparse module. changed Queue to make it compatible with python3 queue. Backwards compatibility is maintained. Restructured default location. Moved from Lib folder to base path. Moved examples to proper folder. This cause backwards compatibility problems. On import, replace `import Lib.NovateloEM4` with simply `import NovateloEM4`

**version 0.3** logging configuration as moved outside module to enable user to use already configured logging handler. Check [multimodule logging docs](#)

**version 0.2** data from bestxyz message is now placed into a `Queue.Queue()` FIFO

**version 0.1** initial release

This module contains a few functions to interact with Novatel OEM4 GPS devices. Currently only the most important functions and definitions are configured, but the intention is to make it as much complete as possible.

A simple example can be run by executing the main function which creates a Gps class object and execute the following commands on gps receiver:

- **begin:** on default port or given port by argv[1].
- **sendUnlogall**
- **setCom(baud=115200):** changes baudrate to 115200bps
- **askLog(trigger=2,period=0.1):** ask for log *bestxyz* with trigger *ONTIME* and period *0.1*
- wait for 10 seconds
- **shutdown:** safely disconnects from gps receiver

**Example:**

```
$python NovateloEM4.py
```

Contents:

## 1.1 Gps Class

**class** NovateloEM4.Gps (*sensorName*=’GPS’)

Novatel OEM4 GPS library class

This class contents is an approach to create a library for Novatel OEM 4 GPS

**Parameters** **sensorName** (*optional*) – A sensor name if used with multiple devices.

**header\_keys**

all field keys for the headers of messages.

**MessageID**

A dictionary for the types of messages sent. Not all are implemented yet!

### 1.1.1 Methods

#### 1.1.1.1 askLog

Gps.askLog (*logID*=’BESTXYZ’, *port*=192, *trigger*=4, *period*=0, *offset*=0, *hold*=0)

Request a log from receiver.

**Parameters**

- **logID** – log type to request.
- **port** – port to report log.
- **trigger** – trigger identifier.
- **period** – the period of log.
- **offset** – offset in seconds after period.
- **hold** – mark log with hold flag or not.

**Returns** True or false if command was sucessfull or not.

The log request command is defined as:

Field	ID	N Bytes	Description
1	Com header	H = 28	Header of message
2	port	ENUM = 4	identification of port
3	message	Ushort = 2	Message ID of log to output
4	messageType	char = 1	Message type (Binary)
5	RESERVED	char = 1	
6	trigger	ENUM = 4	message trigger
7	period	double = 8	Log period (for ONTIME in secs)
8	offset	double = 8	Offset for period (ONTIME in secs)
9	hold	ENUM = 4	Hold log
10	crc32	Ulong = 4	crc32 value

---

**Note:** Total byte size = header + 32 = 60 bytes

---

Log trigger Identifiers (field 6):

Binary	ASCII	Description
0	ONNEW	when the message is updated (not necessarily changed)
1	ONCHANGED	Current message and then continue to output when the message is changed
2	ONTIME	Output on a time interval
3	ONNEXT	Output only the next message
4	ONCE	Output only the current message
5	ONMARK	Output when a pulse is detected on the mark 1 input

### 1.1.1.2 begin

Gps .begin (dataQueue, comPort='/dev/ttyUSB0', baudRate=9600)

Initializes the gps receiver.

This function resets the current port to factory default and setup the gps receiver to be able to accept new commands. If connection to gps is made, it launches a thread used to parse messages comming from gps.

#### Parameters

- **comPort** – system port where receiver is connected.
- **dataQueue** – a Queue object to store incoming bestxyz messages.
- **baudRate** – baudrate to configure port. (should always be equal to factory default of receiver).

**Returns** True or False if the setup has gone as expected or not.

#### Example

```
Gps.begin(comPort="<port>",  
          dataQueue=<your Queue obj>,  
          baudRate=9600)
```

#### Default values

**ComPort** "/dev/ttyUSB0"

**BaudRate** 9600

**Warning:** This class uses module logging which must be configured in your main program using the basicConfig method. Check documentation of [module logging](#) for more info.

**HW info:**

**Receptor** Novatel Flexpak G2L-3151W.

**Antenna** Novatel Pinwheel.

### 1.1.1.3 `create_header`

`Gps.create_header (messageID, messageLength, portAddress=192)`

Creates a header object to be passed to receiver.

**Parameters**

- **messageID** – the corresponding value of identifying the message body.
- **messageLength** – size of message in bytes excluding CRC-32bit code.
- **portAddress** – port from where message request is sent.

**Returns** The header of message.

The header is defined as:

Field	Value	N Bytes	Description
1	sync[0]	UChar = 1	Hexadecimal 0xAA.
2	sync[1]	UChar = 1	Hexadecimal 0x44.
3	sync[2]	UChar = 1	Hexadecimal 0x12.
4	header-Length	UChar = 1	Length of the header (should always be 28 unless some firmware update)
5	messageID	UShort = 2	This is the Message ID code
6	mes- sageType	UChar = 1	message type mask (binary and original message)
7	portAd- dress	Uchar = 1	Corresponding value of port
8	message- Length	UShort = 2	Length of message body
9	sequence	UShort = 2	This is used for multiple related logs.
10	idleTime	UChar = 1	The time that the processor is idle in the last second between successive logs with the same Message ID
11	timeStatus	Enum = 1	Indicates the quality of the GPS time
12	week	UShort = 2	GPS week number.
13	ms	int = 4	Milliseconds from the beginning of the GPS week.
14	receiver- Status	Ulong = 4	32 bits representing the status of various hardware and software components of the receiver.
15	reserved	UShort = 2	
16	swVersion	UShort = 2	receiver software build number.

---

**Note:** portAddress=192 (equal to thisport)

---

#### 1.1.1.4 getDebugMessage

Gps.getDebugMessage (*message*)

Create a string which contains all bytes represented as hex values

Auxiliary function for helping with debug. Receives a binary message as input and convert it as a string with the hexadecimal representation.

**Parameters** **message** – message to be represented.

**Returns** A string of corresponding hex representation of message.

#### 1.1.1.5 parseResponses

Gps.parseResponses ()

A thread to parse responses from device

#### 1.1.1.6 reset

Gps.reset (*delay=0*)

Performs a hardware reset

**Parameters** **delay** – seconds to wait before resetting. Default to zero.

**Returns** A boolean if request was sucessful or not

The reset message is defined as:

Field	value	N Bytes	Description
1	header	H = 28	Header of message
2	delay	UL = 4	Seconds to wait before reset
CRC32		UL = 4	

Following a RESET command, the receiver initiates a coldstart boot up. Therefore, the receiver configuration reverts either to the factory default, if no user configuration was saved, or the last SAVECONFIG settings. The optional delay field is used to set the number of seconds the receiver is to wait before resetting.

#### 1.1.1.7 saveconfig

Gps.saveconfig ()

Save user current configuration

**Returns** A boolean if request was sucessful or not

Saveconfig message is defined as:

Field	value	N Bytes	Description
1	header	H = 28	Header of message
CRC32		UL = 4	

This command saves the user's present configuration in non-volatile memory. The configuration includes the current log settings, FIX settings, port configurations, and so on. Its output is in the RXCONFIG log.

### 1.1.1.8 sbascontrol

Gps .**sbascontrol** (*keywordID=1, systemID=1, prn=0, testmode=0*)  
Set SBAS test mode and PRN SBAS

#### Parameters

- **keywordID** – True or false. Control the reception of SBAS corrections Enable = 1, Disable = 0.
- **systemID** – SBAS system to be used.
- **prn** – PRN corrections to be used.
- **testmode** – Interpretation of type 0 messages.

**Returns** A boolean if request was sucessful or not

sbascontrol message is defined as:

Field	value	N Bytes	Description
1	header	H = 28	Header of message
2	keyword	Enum = 4	Enable = 1 or Disable = 0
3	system	Enum = 4	Choose the SBAS the receiver will use
4	prn	UL = 4	0 - Receiver will use any PRN 120~138 - Receiver will use SBAS only from this PRN
5	testmode	Enum = 4	Interpretation of type 0 messages
CRC32		UL = 4	

System (Field 2) is defined as:

Binary	ASCII	Description
0	NONE	Don't use any SBAS satellites.
1	AUTO	Automatically determinate satellite system to use (default).
2	ANY	Use any and all SBAS satellites found
3	WAAS	Use only WAAS satellites
4	EGNOS	Use only EGNOS satellites
5	MSAS	Use only MSAS satellites

Testmode (field 5) is defined as:

Binary	ASCII	Description
0	NONE	Interpret Type 0 messages as they are intended (as do not use).(default)
1	ZEROTOTWO	Interpret Type 0 messages as type 2 messages
2	IG-NOREZERO	Ignore the usual interpretation of Type 0 messages (as do not use) and continue

This command allows you to dictate how the receiver handles Satellite Based Augmentation System (SBAS) corrections and replaces the now obsolete WAASCORRECTION command. The receiver automatically switches to Pseudorange Differential (RTCM or RTCA) or RTK if the appropriate corrections are received, regardless of the current setting.

### 1.1.1.9 sendUnlogall

`Gps.sendUnlogall (port=8, held=1)`

Send command unlogall to gps device.

On sucess clears all logs on all ports even held logs.

**Returns** True or False if the request has gone as expected or not.

unlogall message is defined as:

Field	value	N Bytes	Description
1	header	H = 28	Header of message
2	port	ENUM = 4	identification of port
3	Held	ENUM = 4	can only be 0 or 1. Clear logs with hold flag or not?
CRC32		UL = 4	

---

**Note:** See: OEMStar Firmware Reference Manual Rev 6 page 161

---

### 1.1.1.10 setCom

`Gps.setCom (baud, port=6, parity=0, databits=8, stopbits=1, handshake=0, echo=0, breakCond=1)`

Set com configuration.

#### Parameters

- **baud** – communication baudrate.
- **port** – Novatel serial ports identifier (default 6 = “thisport”).
- **parity** – byte parity check (default 0).
- **databits** – Number of data bits (default 8).
- **stopbits** – Number of stop bits (default 1).
- **handshake** – Handshaking (default No handshaking).
- **echo** – echo input back to user (default false)
- **breakCond** – Enable break detection (default true)

**Returns** True or false if command was sucessfull or not.

The com request command is defined as:

Field	ID	N Bytes	Description
1	Com header	H = 28	Header of message
2	port	ENUM = 4	identification of port
3	baud	Ulong = 4	Communication baud rate (bps)
4	parity	ENUM = 4	Parity
5	databits	Ulong = 4	Number of data bits (default = 8)
6	stopbits	Ulong = 4	Number of stop bits (default = 1)
7	handshake	ENUM = 4	Handshaking
8	echo	ENUM = 4	No echo (default)(must be 0 or 1)
9	break	ENUM = 4	Enable break detection (default 0),(must be 0 or 1)

---

**Note:** Total byte size = header + 32 = 60 bytes

---

COM Serial Port Identifiers (field 2):

Binary	ASCII	Description
1	COM1	COM port 1
2	COM2	COM port 2
6	THISPORT	The current COM port
8	ALL	All COM ports
9	XCOM1	Virtual COM1 port
10	XCOM2	Virtual COM2 port
13	USB1	USB port 1
14	USB2	USB port 2
15	USB3	USB port 3
17	XCOM3	Virtual COM3 port

Parity(field 4):

Binary	ASCII	Description
0	N	No parity (default)
1	E	Even parity
2	O	Odd parity

Handshaking (field 7):

Binary	ASCII	Description
0	N	No handshaking (default)
1	XON	XON/XOFF software handshaking
2	CTS	CTS/RTS hardware handshaking

---

**Note:** See: OEMStar Firmware Reference Manual Rev 6 page 56

---

### 1.1.1.11 setDynamics

Gps.setDynamics (*dynamicID*)

Set Dynamics of receiver.

**Parameters** **dynamicID** – identifier of the type of dynamic.

**Returns** True or False if the request has gone as expected or not.

dynamics message is defined as:

Field	value	N Bytes	Description
1	header	H = 28	Header of message
2	dynamics	ENUM = 4	identification of dynamics
CRC32		UL = 4	

The dynamics identifiers (field 2) are defined as:

Binary	ASCII	Description
0	AIR	Receiver is in an aircraft or a land vehicle, for example a high speed train, with velocity greater than 110 km/h (30 m/s). This is also the most suitable dynamic for a jittery vehicle at any speed.
1	LAND	Receiver is in a stable land vehicle with velocity less than 110 km/h (30 m/s).
2	FOOT	Receiver is being carried by a person with velocity less than 11 km/h (3 m/s).

This command adjusts the receiver dynamics to that of your environment. It is used to optimally tune receiver parameters. The DYNAMICS command adjusts the Tracking State transition time-out value of the receiver. When the receiver loses the position solution, it attempts to steer the tracking loops for fast reacquisition (5 s time-out by default). The DYNAMICS command allows you to adjust this time-out value, effectively increasing the steering time. The three states 0, 1, and 2 set the time-out to 5, 10, or 20 seconds respectively.

---

**Note:**

- The DYNAMICS command should only be used by advanced users of GPS. The default of AIR should not be changed except under very specific conditions.
  - The DYNAMICS command affects satellite reacquisition. The constraint of the DYNAMICS filter with FOOT is very tight and is appropriate for a user on foot. A sudden tilted or up and down movement, for example while a tractor is moving slowly along a track, may trip the RTK filter to reset and cause the position to jump. AIR should be used in this case.
- 

### 1.1.1.12 shutdown

Gps.shutdown()

Prepare for exiting program

**Returns** always returns true after all tasks are done.

Prepare for turn off the program by executing the following tasks:

- unlogall

- reset port settings
- close port

### 1.1.1.13 CRC32Value

Gps.CRC32Value (*i*)

Calculate the 32bits CRC of message.

See OEMStar Firmware Reference Manual Rev 6 page 24 for more information.

**Parameters** **i** – message to calculate the crc-32.

**Returns** The CRC value calculated over the input message.

## **Appendix B**

# **Maxon EPOS CANopen Library Documentation**

**Date** Sep 11, 2018

**Version** 0.1

**Author** Bruno Tibério

**Contact** bruno.tiberio@tecnico.ulisboa.pt

This documentation describes the class Epos developed in Python using CANopen to control the Maxon Motors EPOS 70/10 device.



# CHAPTER 1

---

## Epos Class description

---

```
class epos.Epos(_network=None, debug=False)
```

```
begin(nodeID, _channel='can0', _bustype='socketcan', objectDictionary=None)
```

Initialize Epos device

Configure and setup Epos device.

### Parameters

- **nodeID** – Node ID of the device.
- **channel** (*optional*) – Port used for communication. Default can0
- **bustype** (*optional*) – Port type used. Default socketcan.
- **objectDictionary** (*optional*) – Name of EDS file, if any available.

**Returns** A boolean if all went ok.

**Return type** bool

```
changeEposState(newState)
```

Change EPOS state

Change Epos state using controlWord object

To change Epos state, a write to controlWord object is made. The bit change in controlWord is made as shown in the following table:

State	LowByte of Controlword [binary]
shutdown	0xxx x110
switch on	0xxx x111
disable voltage	0xxx xx0x
quick stop	0xxx x01x
disable operation	0xxx 0111
enable operation	0xxx 1111
fault reset	1xxx xxxx

see section 8.1.3 of firmware for more information

**Parameters** `newState` – string with state which user want to switch.

**Returns** boolean if all went ok and no error was received.

**Return type** bool

**checkEposState ()**

Check current state of Epos

Ask the StatusWord of EPOS and parse it to return the current state of EPOS.

State	ID	Statusword [binary]
Start	0	x0xx xxx0 x000 0000
Not Ready to Switch On	1	x0xx xxx1 x000 0000
Switch on disabled	2	x0xx xxx1 x100 0000
ready to switch on	3	x0xx xxx1 x010 0001
switched on	4	x0xx xxx1 x010 0011
refresh	5	x1xx xxx1 x010 0011
measure init	6	x1xx xxx1 x011 0011
operation enable	7	x0xx xxx1 x011 0111
quick stop active	8	x0xx xxx1 x001 0111
fault reaction active (disabled)	9	x0xx xxx1 x000 1111
fault reaction active (enabled)	10	x0xx xxx1 x001 1111
Fault	11	x0xx xxx1 x000 1000

see section 8.1.1 of firmware manual for more details.

**Returns** numeric identification of the state or -1 in case of fail.

**Return type** int

**loadConfig ()**

Load all configurations

**logDebug (message=None)**

Log a message

A wrap around logging. The log message will have the following structure: [class name : function name ] message

the function name will be the caller function retrieved automatically by using sys.\_getframe(1).f\_code.co\_name

**Parameters** `message` – a string with the message.

**logInfo (message=None)**

Log a message

A wrap around logging. The log message will have the following structure: [class name : function name ] message

**Parameters** `message` – a string with the message.

**printControlWord (controlword=None)**

Print the meaning of controlword

Check the meaning of current controlword of device or check the meaning of your own controlword. Usefull to check your own controlword before actually sending it to device.

**Parameters** `controlword (optional)` – If None, request the controlword of device.

```
printCurrentControlParameters()
    Print the current mode control PI gains
    Request current mode control parameter gains from device and print.

printMotorConfig()
    Print current motor config
    Request current motor config and print it

printOpMode()
    Print current operation mode

printPositionControlParameters()
    Print position control mode parameters
    Request device for the position control mode parameters and prints it.

printSensorConfig()
    Print current sensor configuration

printSoftwarePosLimit()
    Print current software position limits

readControlWord()
    Read ControlWord
    Request current controlword from device.
```

**Returns**

A tuple containing:  
**controlword** the current controlword or None if any error.  
**Ok** A boolean if all went ok.

**Return type** tuple

```
readCurrentControlParameters()
    Read the PI gains used in current control mode
```

**Returns**

A tuple containing:  
**gains** A dictionary with the current pGain and iGain  
**OK** A boolean if all went as expected or not.

**Return type** tuple

```
readCurrentModeSetting()
    Read current value setted
    Asks EPOS for the current value setted in current control mode.
```

**Returns**

A tuple containing:  
**current** value setted.  
**Ok** a boolean if sucessfull or not.

**Return type** tuple

**readCurrentValue ()**

Read current value

**Returns**

a tuple containing:

**current** current in mA.

**Ok** A boolean if all requests went ok or not.

**Return type** tuple

**readCurrentValueAveraged ()**

Read current averaged value

**Returns**

a tuple containing:

**current** current averaged in mA.

**Ok** A boolean if all requests went ok or not.

**Return type** tuple

**readFollowingError ()**

Returns the current following error

Read the current following error value which is the difference between atual value and desired value.

**Returns**

a tuple containing:

**followingError** value of actual following error.

**OK** A boolean if all requests went ok or not.

**Return type** tuple

**readMaxFollowingError ()**

Read the Max following error

Read the max following error value which is the maximum allowed difference between atual value and desired value in modulus.

**Returns**

a tuple containing:

**maxFollowingError** value of max following error.

**OK** A boolean if all requests went ok or not.

**Return type** tuple

**readMotorConfig ()**

Read motor configuration

Read the current motor configuration

Requests from EPOS the current motor type and motor data. The motorConfig is a dictionary containing the following information:

- **motorType** describes the type of motor.
- **currentLimit** - describes the maximum continuous current limit.

- **maxCurrentLimit** - describes the maximum allowed current limit. Usually is set as two times the continuous current limit.
- **polePairNumber** - describes the pole pair number of the rotor of the brushless DC motor.
- **maximumSpeed** - describes the maximum allowed speed in current mode.
- **thermalTimeConstant** - describes the thermal time constant of motor winding is used to calculate the time how long the maximal output current is allowed for the connected motor [100 ms].

If unable to request the configuration or unsucessfull, None and false is returned .

#### Returns

A tuple with:

**motorConfig** A structure with the current configuration of motor

**OK** A boolean if all went as expected or not.

#### Return type

tuple

**readObject** (*index, subindex*)

Reads an object

Request a read from dictionary object referenced by index and subindex.

#### Parameters

- **index** – reference of dictionary object index
- **subindex** – reference of dictionary object subindex

**Returns** message returned by EPOS or empty if unsucessfull

#### Return type

bytes

**readOpMode** ()

Read current operation mode

#### Returns

A tuple containing:

**opMode** current opMode or None if request fails

**Ok** A boolean if sucessfull or not

#### Return type

tuple

**readPositionControlParameters** ()

Read position mode control parameters

Read position mode control PID gains and and feedfoward and acceleration values

#### Returns

A tuple containing:

**posModeParameters** a dictionary containg pGain, iGain, dGain, vFeed and aFeed.

**OK** A boolean if all went as expected or not.

#### Return type

tuple

**readPositionModeSetting** ()

Reads the setted desired Position

Ask Epos device for demand position object. If a correct request is made, the position is placed in answer. If not, an answer will be empty.

**Returns**

A tuple containing:

**position** the demanded position value.

**OK** A boolean if all requests went ok or not.

**Return type** tuple

**readPositionValue()**

Read current position value

**Returns**

a tuple containing:

**position** current position in quadrature counts.

**Ok** A boolean if all requests went ok or not.

**Return type** tuple

**readPositionWindow()**

Read current position Window value.

Position window is the modulus threshold value in which the output is considered to be achieved.

**Returns**

a tuple containing:

**positionWindow** current position window in quadrature counts.

**Ok** A boolean if all requests went ok or not.

**Return type** tuple

**readPositionWindowTime()**

Read current position Window time value.

Position window time is the minimum time in milliseconds in which the output must be inside the position window for the target is considered to have been reached.

**Returns**

a tuple containing:

**positionWindowTime** current position window time in milliseconds.

**Ok** A boolean if all requests went ok or not.

**Return type** tuple

**readQuickStopDeceleration()**

Read the quick stop deceleration.

Read deceleration used in fault reaction state.

**Returns**

A tuple containing:

**quickstopDeceleration** The value of deceleration in rpm/s.

**OK** A boolean if all went as expected or not.

**Return type** tuple

**readSensorConfig()**

Read sensor configuration

Requests from EPOS the current sensor configuration. The sensorConfig is an struture containing the following information:

- sensorType - describes the type of sensor.
- pulseNumber - describes the number of pulses per revolution in one channel.
- sensorPolarity - describes the of each sensor.

If unable to request the configuration or unsucessfull, an empty structure is returned. Any error inside any field requests are marked with 'error'.

**Returns**

A tuple containing:

**sensorConfig** A dictionary with the current configuration of the sensor

**OK** A boolean if all went as expected or not.

**Return type** tuple

**readSoftwarePosLimit()**

Read the software position limit

**Returns**

A tuple containing:

**limits** a dictionary containing minPos and maxPos

**OK** A boolean if all went as expected or not.

**Return type** tuple

**readStatusWord()**

Read StatusWord

Request current statusword from device.

**Returns**

A tuple containing:

**statusword** the current statusword or None if any error.

**Ok** A boolean if all went ok.

**Return type** tuple

**readVelocityModeSetting()**

Reads the setted desired velocity

Asks EPOS for the desired velocity value in velocity control mode

**Returns**

A tuple containing:

**velocity** Value setted or None if any error.

**Ok** A boolean if sucessfull or not.

**Return type** tuple

**readVelocityValue ()**

Read current velocity value

**Returns**

a tuple containing:

**velocity** current velocity in rpm.

**Ok** A boolean if all requests went ok or not.

**Return type** tuple

**readVelocityValueAveraged ()**

Read current velocity averaged value

**Returns**

a tuple containing:

**velocity** current velocity in rpm.

**Ok** A boolean if all requests went ok or not.

**Return type** tuple

**saveConfig ()**

Save all configurrrations

**setCurrentControlParameters (pGain, iGain)**

Set the PI gains used in current control mode

**Parameters**

- **pGain** – Proportional gain.
- **iGain** – Integral gain.

**Returns** A boolean if all went as expected or not.

**Return type** bool

**setCurrentModeSetting (current)**

Set desired current

Set the value for desired current in current control mode

**Parameters** **current** – the value to be set [mA]

**Returns** a boolean if sucessfull or not

**Return type** bool

**setMaxFollowingError (maxFollowingError)**

Set the Max following error

The Max Following Error is the maximum permissible difference between demanded and actual position at any time of evaluation. It serves as a safety and motion-supervising feature. If the following error becomes too high, this is a sign of something going wrong. Either the drive cannot reach the required speed or it is even blocked.

**Parameters** **maxFollowingError** – The value of maximum following error.

**Returns** A boolean if all requests went ok or not.

**Return type** bool

**setMotorConfig** (*motorType*, *currentLimit*, *maximumSpeed*, *polePairNumber*)

Set motor configuration

Sets the configuration of the motor parameters. The valid motor type is:

<b>motorType</b>	<b>value</b>	<b>Description</b>
DC motor	1	brushed DC motor
Sinusoidal PM BL motor	10	EC motor sinus commutated
Trapezoidal PM BL motor	11	EC motor block commutated

The current limit is the current limit is the maximal permissible continuous current of the motor in mA. Minimum value is 0 and max is hardware dependent.

The output current limit is recommended to be 2 times the continuous current limit.

The pole pair number refers to the number of magnetic pole pairs (number of poles / 2) from rotor of a brushless DC motor.

The maximum speed is used to prevent mechanical destroys in current mode. It is possible to limit the velocity [rpm]

Thermal winding not changed, using default 40ms.

**Parameters**

- **motorType** – value of motor type. see table behind.
- **currentLimit** – max continuous current limit [mA].
- **maximumSpeed** – max allowed speed in current mode [rpm].
- **polePairNumber** – number of pole pairs for brushless DC motors.

**Returns** A boolean if all requests went ok or not.

**Return type** bool

**setOpMode** (*opMode*)

Set Operation mode

Sets the operation mode of Epos. OpMode is described as:

<b>OpMode</b>	<b>Description</b>
6	Homing Mode
3	Profile Velocity Mode
1	Profile Position Mode
-1	Position Mode
-2	Velocity Mode
-3	Current Mode
-4	Diagnostic Mode
-5	MasterEncoder Mode
-6	Step/Direction Mode

**Parameters** **opMode** – the desired opMode.

**Returns** A boolean if all requests went ok or not.

**Return type** bool

**setPositionControlParameters** (*pGain, iGain, dGain, vFeed=0, aFeed=0*)

Set position mode control parameters

Set position control PID gains and feedforward velocity and acceleration values.

**Feedback and Feed Forward**

*PID feedback amplification*

PID stands for Proportional, Integral and Derivative control parameters. They describe how the error signal  $e$  is amplified in order to produce an appropriate correction. The goal is to reduce this error, i.e. the deviation between the set (or demand) value and the measured (or actual) value. Low values of control parameters will usually result in a sluggish control behavior. High values will lead to a stiffer control with the risk of overshoot and at too high an amplification, the system may start oscillating.

*Feed-forward*

With the PID algorithms, corrective action only occurs if there is a deviation between the set and actual values. For positioning systems, this means that there always is “in fact, there has to be a position error while in motion. This is called following error. The objective of the feedforward control is to minimize this following error by taking into account the set value changes in advance. Energy is provided in an open-loop controller set-up to compensate friction and for the purpose of mass inertia acceleration. Generally, there are two parameters available in feed-forward. They have to be determined for the specific application and motion task:

- Speed feed-forward gain: This component is multiplied by the demanded speed and compensates for speed-proportional friction.
- Acceleration feed-forward correction: This component is related to the mass inertia of the system and provides sufficient current to accelerate this inertia.

Incorporating the feed forward features reduces the average following error when accelerating and decelerating. By combining a feed-forward control and PID, the PID controller only has to correct the residual error remaining after feed-forward, thereby improving the system response and allowing very stiff control behavior.

According to [Position Regulation with Feed Forward](#) the acceleration and velocity feed forward take effect in Profile Position Mode and Homing Mode. There is no influence to all the other operation modes like Position Mode, Profile Velocity Mode, Velocity Mode and Current Mode

**Parameters**

- **pGain** – Proportional gain value
- **iGain** – Integral gain value
- **dGain** – Derivative gain value
- **vFeed** – velocity feed foward gain value. Default to 0
- **aFeed** – acceleration feed foward gain value. Default to 0

**Returns** A boolean if all requests went ok or not

**Return type** OK

**setPositionModeSetting** (*position*)

Sets the desired Position

Ask Epos device to define position mode setting object.

**Returns** A boolean if all requests went ok or not.

**Return type** bool

**setPositionWindow (positionWindow)**

Set position Window value

Position window is the modulus threshold value in which the output is considered to be achieved.

**Parameters** **positionWindow** – position window in quadrature counts

**Returns** A boolean if all requests went ok or not.

**Return type** bool

**setPositionWindowTime (positionWindowTime)**

Set position Window Time value

Position window time is the minimum time in milliseconds in which the output must be inside the position window for the target is considered to have been reached.

**Parameters** **positionWindowTime** – position window time in milliseconds.

**Returns** A boolean if all requests went ok or not.

**Return type** bool

**setQuickStopDeceleration (quickstopDeceleration)**

Set the quick stop deceleration.

The quick stop deceleration defines the deceleration during a fault reaction.

**Parameters** **quicstopDeceleration** – the value of deceleration in rpm/s

**Returns** A boolean if all went as expected or not.

**Return type** bool

**setSensorConfig (pulseNumber, sensorType, sensorPolarity)**

Change sensor configuration

Change the sensor configuration of motor. **Only possible if in disable state** The encoder pulse number should be set to number of counts per revolution of the connected incremental encoder. range : [16 - 7500] sensor type is described as:

value	description
1	Incremental Encoder with index (3-channel)
2	Incremental Encoder without index (2-channel)
3	Hall Sensors (Remark: consider worse resolution)

sensor polarity is set by setting the corresponding bit from the word:

Bit	description
15-2	Reserved (0)
1	Hall sensors polarity 0: normal / 1: inverted
0	Encoder polarity 0: normal 1: inverted (or encoder mounted on motor shaft side)

**Parameters**

- **pulseNumber** – Number of pulses per revolution.

- **sensorType** – 1,2 or 3 according to the previous table.
- **sensorPolarity** – a value between 0 and 3 describing the polarity of sensors as stated before.

**Returns** A boolean if all went as expected or not.

**Return type** bool

**setSoftwarePosLimit** (*minPos, maxPos*)

Set the software position limits

Use encoder readings as limit position for extremes range = [-2147483648 | 2147483647]

**Parameters**

- **minPos** – minimum possition limit
- **maxPos** – maximum possition limit

**Returns** A boolean if all went as expected or not.

**Return type** bool

**setVelocityModeSetting** (*velocity*)

Set desired velocity

Set the value for desired velocity in velocity control mode.

**Parameters** **velocity** – value to be setted.

**Returns** a boolean if sucessfull or not.

**Return type** bool

**writeControlWord** (*controlword*)

Send controlword to device

**Parameters** **controlword** – word to be sent.

**Returns** a boolean if all went ok.

**Return type** bool

**writeObject** (*index, subindex, data*)

Write an object

Request a write to dictionary object referenced by index and subindex.

**Parameters**

- **index** – reference of dictionary object index
- **subindex** – reference of dictionary object subindex
- **data** – data to be stored

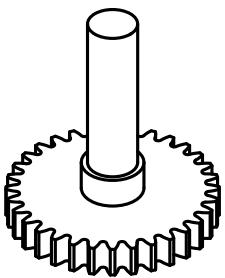
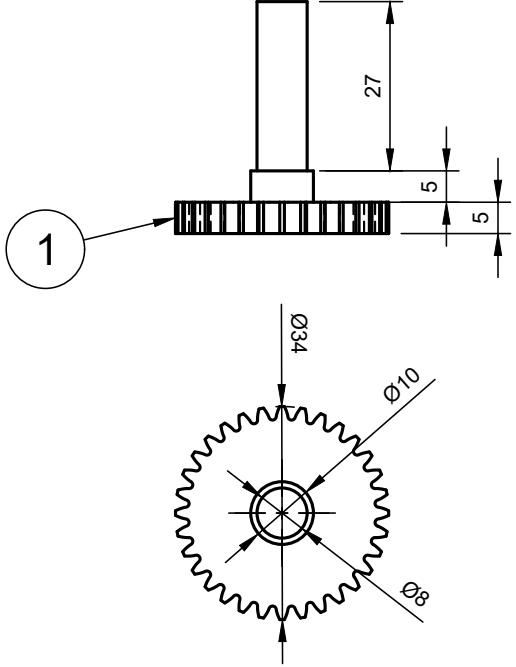
**Returns** boolean if all went ok or not

**Return type** bool

## **Appendix C**

### **Sensor support Drawings**

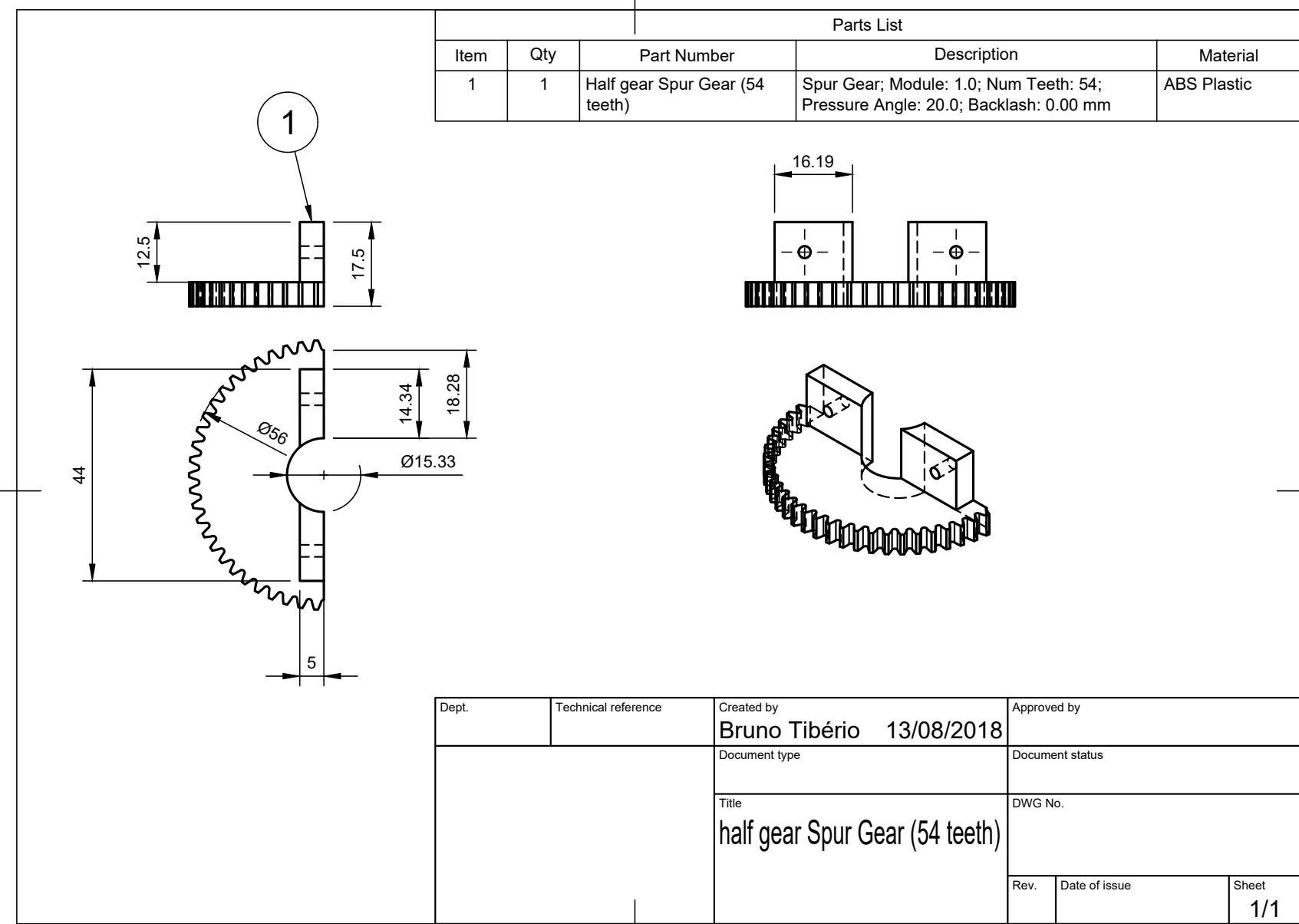
## C.1 Sensor gear 32 teeth

Parts List				
Item	Qty	Part Number	Description	Material
1	1	Sensor gear (32 teeth) 1mm module	Spur Gear; Module: 1.0; Num Teeth: 32; Pressure Angle: 20.0; Backlash: 0.00 mm	ABS Plastic
<hr/>				
				
				
<hr/>				
Dept.	Technical reference	Created by <b>Bruno Tibério</b>	Approved by	
		13/08/2018		
		Document type	Document status	
		Title <b>sensor gear (32 teeth) 1mm module</b>	DWG No.	
			Rev.	Date of issue
				Sheet <b>1/1</b>

C.2

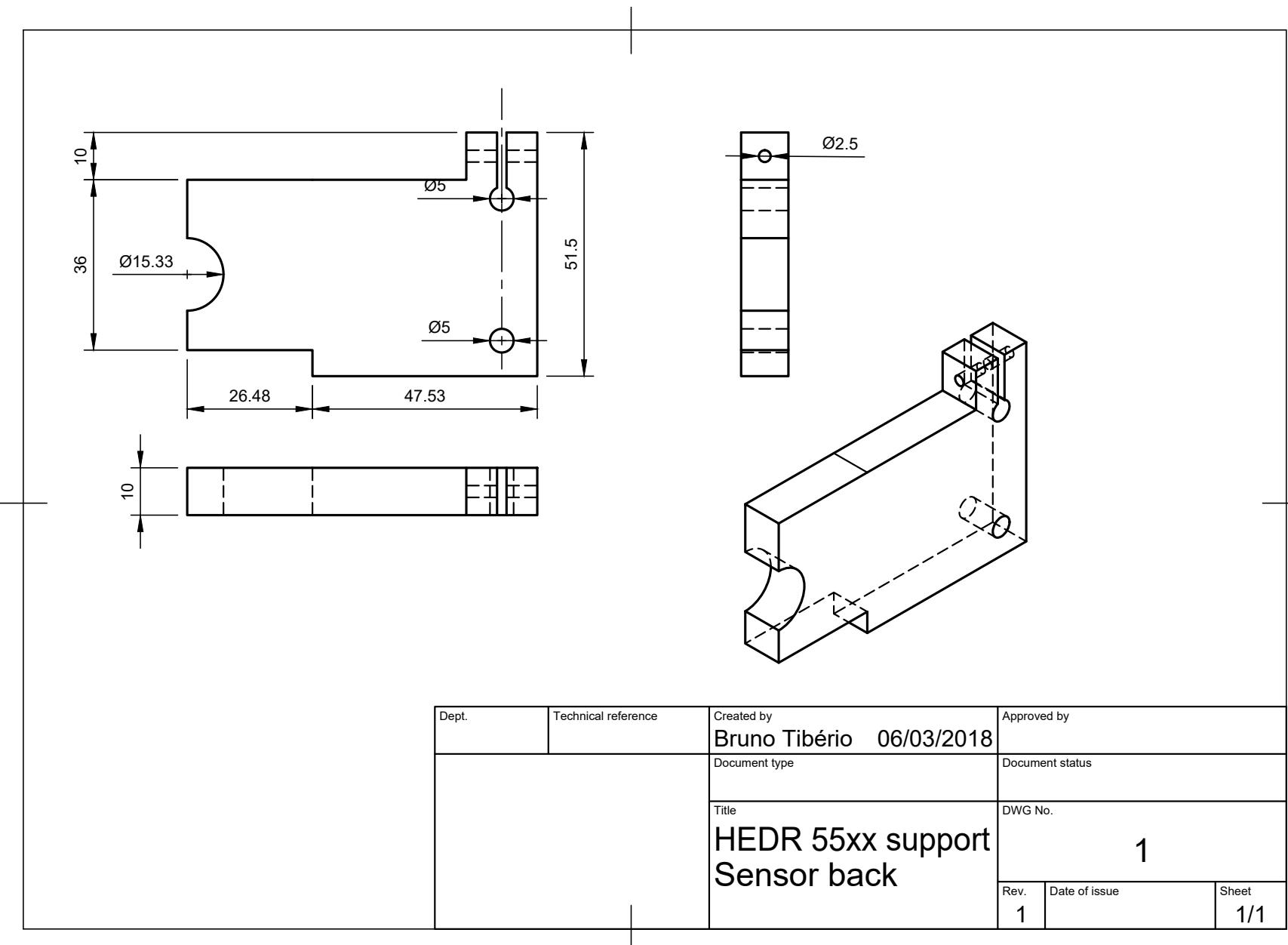
## C.2 Half gear 54 teeth

C.3

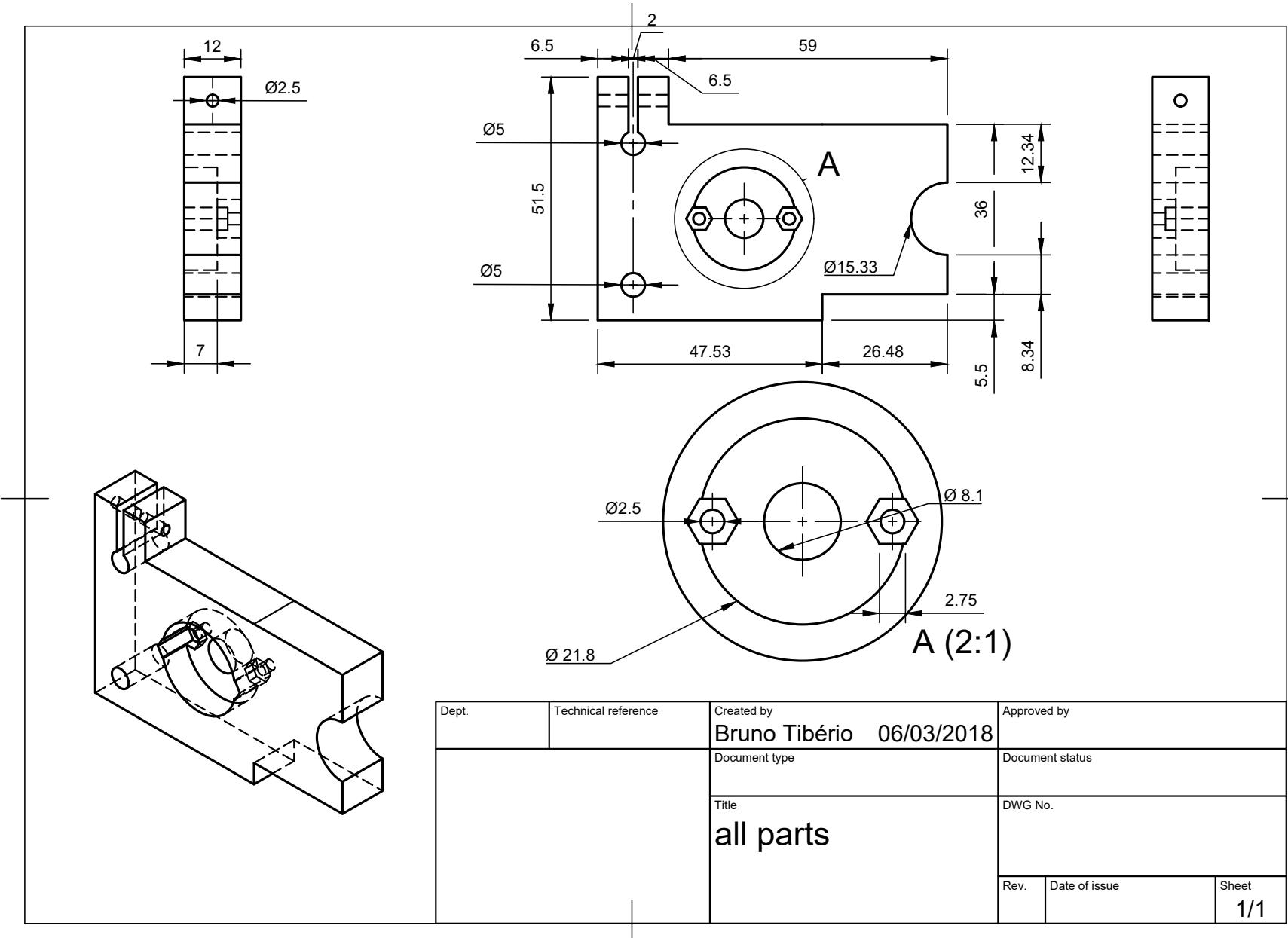


### C.3 Sensor back spacer

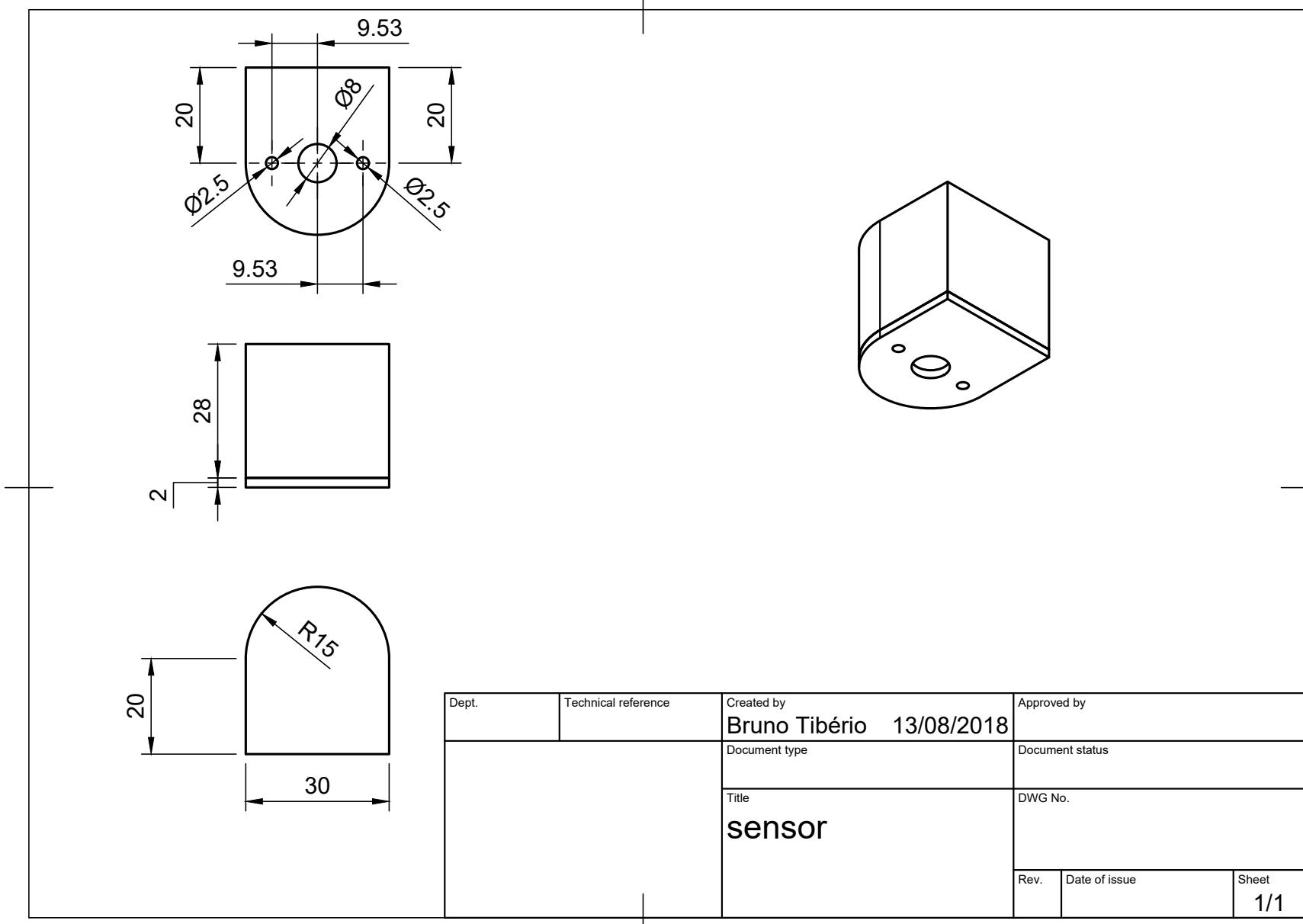
C.4



## C.4 Sensor bearing holder



## C.5 Sensor case

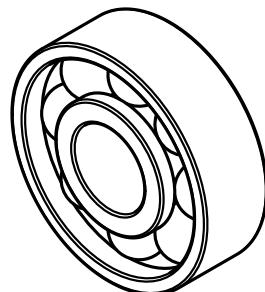
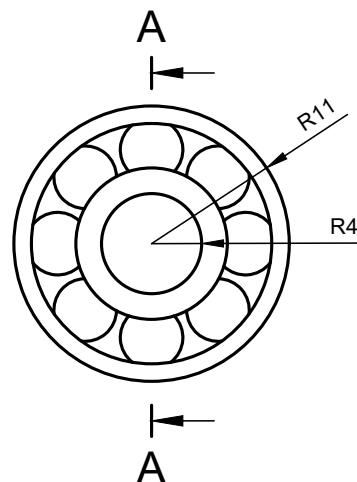
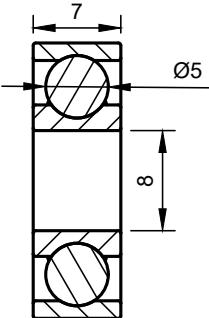


C.6

## C.6 Bearing

C.7

A-A (2:1)



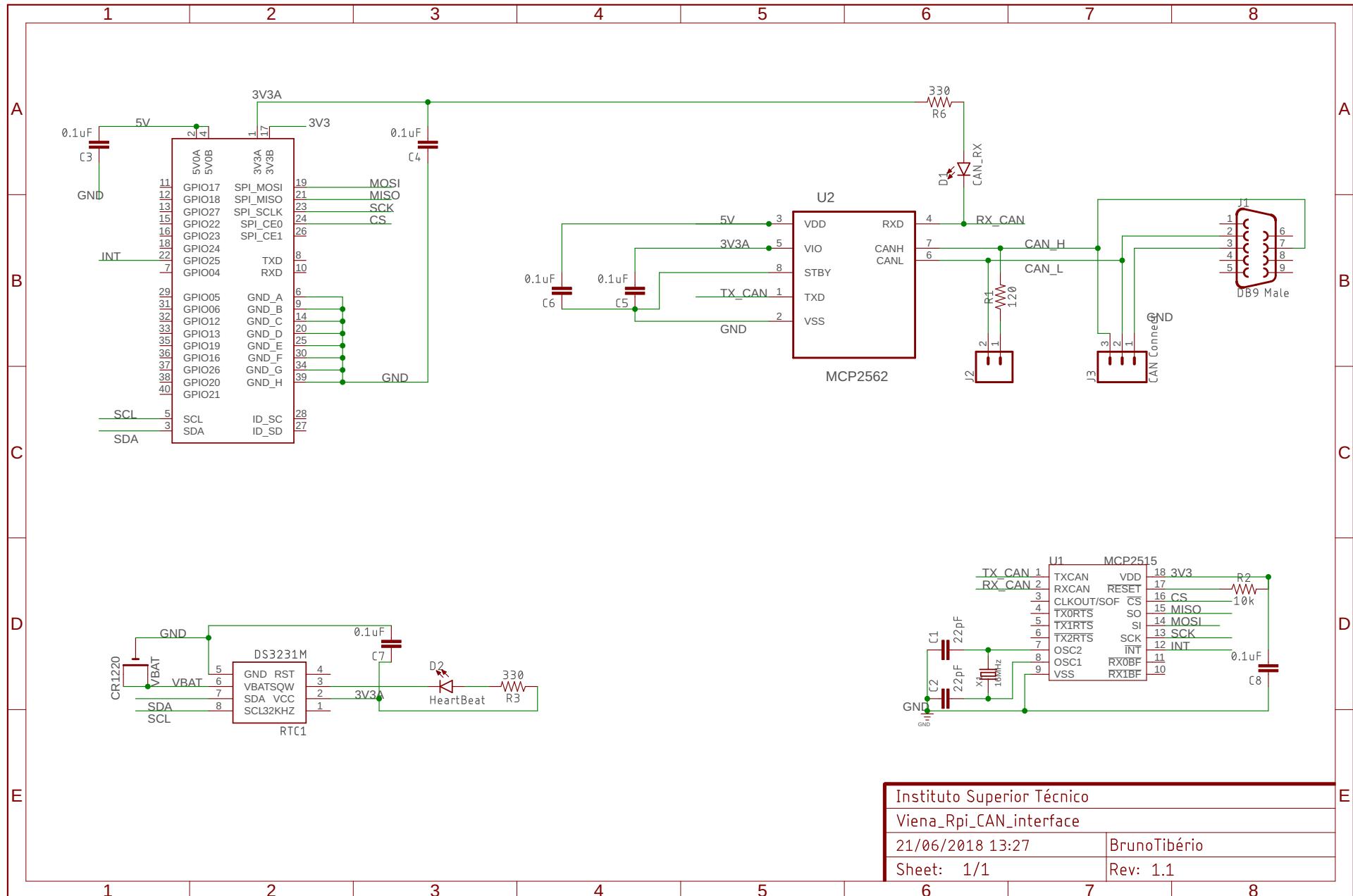
Dept.	Technical reference	Created by Bruno Tibério	Approved by 
		Document type	
	Title bearing 608zz		DWG No.
Rev.	Date of issue	Sheet	1/1

### Parts List

Item	Qty	Part Number	Description	Material
1	1	Bearing 608zz		Steel

## **Appendix D**

### **Development CAN board schematics**



# Appendix E

## Calibration script

```
1 function find_calibration_parameters()
2 %-----
3 % Constants
4 %-----
5 % readings at 0 qc.
6 % Since it has different values, it means when the EPOS was turned on,
7 % the visually estimated position for wheels with zero angle has a
8 % small offset. In this case should be near atan2d(0.5*0.05, 1.47) ~ 1
9 % degree
10 steeringWheelZero=0.01*[92.5, 97.5];
11 % dL is distance from OR and OL with steering wheel at center position.
12 % qc should be near zero.
13 dL = 0.01*45.5;
14 % d1 is distance between attached laser position in the outside wheel
15 % face
16 d1 = 1.445;
17 % d0 is distance between center of wheels. Shaft distance.
18 d0 = 1.40;
19 % h1 is distance from wheel front axis to the "rule" of readings.
20 h1 = 1.47;
21 % maxR and maxL is maximum distance achievable from point OR and OL
22 % respectively
23 maxR = 2.19;
24 maxL = 2.19;
25 maxRange = maxR+maxL-dL;
26
27 L1 = 0.5*(maxRange -d1);
28 L2 = 0.5*(d1-dL);
%-----
```

```

30 % kept for future improvements!
31 %-----
32 %     %marked positions of left wheel starting from 0_L to the left side
33 %     distL = 0.01*(0:20:200);
34 %     % readings of encoder
35 %     qcL = [47602, 40267, 30764, 20356, 9374, -1643, -11968, -21040, ...
36 %             -29419, -36495, -41506];
37 %-----
38 % readings right wheel starting from 0_R to the right side
39 distR = 0.01*(0:20:200);
40 qcR = [ -45178, -37680, -28428, -18063, -6735, 4388, 15110, 24331, ...
41             32335, 39414, 44218];
42 % readings of left wheel from 0L to the left, placing the steering
43 % wheel in the same quadrature counters as best as possible
44 distL=[219, 184.5, 158, 133.5, 111, 91, 71.8, 53.8, 37.8, 23, 10];
45
46 %-----
47 % Data correction to fit 0 with steering wheel aligned
48 %-----
49 % adjust measures so \beta = 0 gives a reading of 0 distance.
50 %distL = distL-(dL+L2);
51 distL = 0.01*distL-(dL+L2);
52 % make positive to left side and
53 % adjust measures so \beta = 0 gives a reading of 0 distance.
54 distR = -1*(distR-(dL+L2));
55 %-----
56 % calculate all angles
57 %-----
58 betaL = zeros(1,length(distL));
59 alphaR = betaL;
60 dirac = alphaR;
61 deltaDirac = dirac;
62 for I =1: length(distL)
63     % betaL(I) = atan2d(distL(I), 1.47);
64     betaL(I) = atan2d(distL(I),1.47);
65     alphaR(I) = atan2d(distR(I), 1.47);
66     dirac(I) = 0.5*(betaL(I)+alphaR(I));
67     deltaDirac(I) = 0.5*(betaL(I)-alphaR(I));
68 end
69 %-----
70 % plot figures
71 %-----
```

```

72 figure();
73 plot(qcR,distL,qcR,distR);
74 legend('Dist_L','Dist_R');
75 ylabel('distance [m]');
76 xlabel('Steering wheel position [qc]');
77
78 figure();
79 plot(qcR, alphaR, qcR, betaL, qcR, dirac, qcR, deltaDirac);
80 xlabel('Steering wheel position [qc]');
81 ylabel('angles [degrees]');
82 hold on;
83 line([qcR(1) qcR(end)],[0, 0], 'color', 'red', 'LineStyle', '--');
84 limMax = max(max(alphaR),max(betaL));
85 limMin = min(min(alphaR),min(betaL));
86 line([qcR(1)+0.5*(qcR(end)-qcR(1)) qcR(1)+0.5*(qcR(end)-qcR(1))] ,...
87 [limMin limMax], 'color', 'red', 'LineStyle', '--');
88 legend('\alpha_R', '\beta_L', '\delta', '\Delta\delta', '0');
89
90 % change format to show small values in cmd window
91 format short e;
92 %-----
93 % polyfit of alpha, beta, dirac, deltaDirac
94 %-----
95 % alpha and beta must have oposite concavities
96
97 [pAlpha, sAlpha] = polyfit(qcR, alphaR, 2);
98 fprintf('Polyfit for alpha:');
99 display(pAlpha);
100 fprintf('Norm of the residuals:%.4g\n', sAlpha.normr)
101 [pBeta, sBeta] = polyfit(qcR, betaL, 2);
102 fprintf('Polyfit for beta:');
103 display(pBeta);
104 fprintf('Norm of the residuals:%.4g\n', sBeta.normr)
105 % delta must be concave up always.
106 [pDirac, sDirac] = polyfit(qcR, dirac, 1);
107 fprintf('Polyfit for dirac:');
108 display(pDirac);
109 fprintf('Norm of the residuals:%.4g\n', sDirac.normr)
110 [pDelta, sDelta] = polyfit(qcR, deltaDirac, 2);
111 fprintf('Polyfit for delta:');
112 display(pDelta);
113 fprintf('Norm of the residuals:%.4g\n', sDelta.normr)

```

```

114 % reset format
115 format;
116 %-----
117 % evaluate polyfits for range -40000 to 40000
118 %-----
119 qc = -40000:100:40000;
120 % set zero offset in each polyfit
121 % The values for alpha = beta = dirac = deltaDirac must be all zero for
122 % a perfect zero alignment of the steering wheels. This does not happen
123 % because the wheels at qc = 0 had a small misalignment.
124 pAlpha(end) = 0;
125 pBeta(end) = 0;
126 pDirac(end) = 0;
127 pDelta(end) = 0;
128 alpha = polyval(pAlpha, qc);
129 beta = polyval(pBeta, qc);
130 dirac2 = polyval(pDirac, qc);
131 delta = polyval(pDelta, qc);
132 figure();
133 subplot(2,1,1);
134 plot(qc, alpha, qc, beta, qc, dirac2);
135 % xlabel('Quadrature counters [qc]');
136 ylabel('Angle [degrees]');
137
138 title(['Steering wheel ratio estimate: \delta = ',...
139 sprintf('%.4g', pDirac(1)), 'qc']);
140 legend('\alpha', '\beta', '\delta');
141 subplot(2,1,2);
142 plot(qc, delta);
143 xlabel('Quadrature counters [qc]');
144 ylabel('Angle [degrees]');
145 legend('\Delta\delta');
146 end

```

appendix/find\_calibration\_parameters.m

## **Appendix F**

### **Steering controller**

## F.1 Steering Controller - documentation

### EPOS CONTROLLER TESTER CLASS

Simple class to perform a few tests with EPOS device in the car It is a wrap around Epos class to include additional functions.

**class** steering\_controller.Epos\_controller (\_network=None, debug=False)

Bases: EPOS\_Canopen.epos.Epos

**emcyErrorPrint** (EmcyError)

Print any EMCY Error Received on CAN BUS

**getDeltaAngle** (qc)

Converts qc of steering wheel to angle of wheel

Given the desired qc steering position, convert the requested value to angle of bicycle model in degrees.

**Parameters** **qc** – an int with desired qc position of steering wheel.

**Returns** estimated angle of wheels in degrees or None if not possible

**Return type** double

**getQcPosition** (delta)

Converts angle of wheels to qc

Given the desired angle of wheels, in degrees of the bicycle model of car, convert the requested value to qc position of steering wheel using the calibration performed at beginning.

**Parameters** **delta** – desired angle of wheels in degrees.

**Returns** a rounded integer with qc position estimated or None if not possible

**Return type** int

**moveToPosition** (pFinal, isAngle=False)

Move to desired position.

Plan and apply a motion profile to reduce with low jerk, max speed, max acceleration to avoid abrupt variations. The function implement the algorithm developed in<sup>1</sup>

**Parameters**

- **pFinal** – desired position.
- **isAngle** – a boolean, true if pFinal is an angle or false if is qc value

**Returns**

<sup>1</sup> Li, Huaizhong & M Gong, Z & Lin, Wei & Lippa, T. (2007). Motion profile planning for reduced jerk and vibration residuals. 10.13140/2.1.4211.2647.

### **readFromFile** (*filename=None, useAngle=False*)

Read qc positions from file and follow them

The file must contain time and position in quadrature positions of steering wheel and angle (degrees) of “center” wheel of bicycle model in a csv style

time	position	angle
t1	p1	a1
...	...	...
tN	pN	aN

If the calibration value is not the same, user should set useAngle flag. Because the calibration when the file was created can differ from current calibration, user should set the useAngle flag to calculate the position reference to send for device via the given angle.

#### Parameters

- **filename** – csv file to be read.
- **useAngle** – use the angle value instead of position.

### **saveToFile** (*filename=None, exitFlag=None*)

Record qc positions into a csv file

The following fields will be recorded

time	position	angle
t1	p1	a1
...	...	...
tN	pN	aN

An additional file with same name but with ext TXT will have the current calibration parameters

- minValue
- maxValue
- zeroRef

If filename is not supplied or already used, the current asctime() will be used as filename.

#### Parameters

- **filename** – name of the file to save the data
- **exitFlag** – threading event flag to signal exit.

### **startCalibration** (*exitFlag=None*)

Perform steering wheel calibration

This function is expected to be run on a thread in order to find the limits of the steering wheel position and find the expected value of the zero angle of wheels.

Parameters **exitFlag** – threading.Event() to signal the finish of acquisition

## F.2 Steering Controller - script

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 # The MIT License (MIT)
4 # Copyright (c) 2018 Bruno Tiberio
5 #
6 # Permission is hereby granted, free of charge, to any person obtaining a copy
7 # of this software and associated documentation files (the "Software"), to deal
8 # in the Software without restriction, including without limitation the rights
9 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10 # copies of the Software, and to permit persons to whom the Software is
11 # furnished to do so, subject to the following conditions:
12 #
13 # The above copyright notice and this permission notice shall be included in all
14 # copies or substantial portions of the Software.
15 #
16 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
21 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
22 # SOFTWARE.
23
24 import logging
25 import sys
26 import threading
27 import time
28 import gc
29
30 import numpy as np
31 import canopen
32 import pathlib
33 import csv
34 from EPOS_Canopen.epos import Epos
35 # -----
36 # Pycharm remote debug settings
37 # -----
38 # import pydevd
39
40 # pydevd.settrace("10.42.0.1", port=8000, stdoutToServer=True, stderrToServer=True)
41
42
43 # shortcut for clear console
44
45
46 # def clear():
47 #     os.system('cls' if os.name == 'nt' else 'clear')
48
49 # This code section is based on
50 # https://codereview.stackexchange.com/questions/32207/console-user-main-menu
51 class Menu(object):
52     """Base class for the menu"""
53
54     def __init__(self, name, buttons):
55         # Initialize values
56         self.name = name
```

```

57         self.buttons = buttons
58         self.exitFlag = False
59
60     def display(self):
61         """Displaying the menu alongside the navigation elements"""
62
63         # clear()
64         gc.collect() # garbage collect
65         # Display menu name
66         print(self.name)
67
68         # Display menu buttons
69         for button in self.buttons:
70             print("    ", button.nav, button.name)
71
72         # Wait for user input
73         return self.userInput()
74
75     def userInput(self):
76         """Method to check and act upon user's input"""
77
78         # This holds the amount of errors for the
79         # navigation element to input comparison.
80         errSel = 0
81         inputSel = input("Enter selection> ")
82
83         for button in self.buttons:
84             # If input equals to button's navigation element
85             if inputSel == str(button.nav):
86                 # Do the button's function
87                 return button.nav
88             # If input != navigation element
89             else:
90                 # Increase "error on selection" by one, for
91                 # counting the errors and checking their
92                 # amount against the total number of
93                 # buttons. If greater to or equal that means
94                 # no elements were selected.
95                 # In that case show error and try again
96                 errSel += 1
97
98             # No usable input, try again
99             if errSel >= len(self.buttons):
100                 print("Error on selection; try again.")
101                 return None
102
103
104 class Button(object):
105     """Base class for menu buttons"""
106
107     def __init__(self, name, nav):
108         # Initialize values
109         self.name = name
110         # Navigation element; number which user has to enter to do button action
111         self.nav = nav
112
113
114 class Epos_controller(Epos):

```

```

115     maxFollowingError = 7500
116     minValue = 0 # type: int
117     maxValue = 0 # type: int
118     zeroRef = 0 # type: int
119     calibrated = 0 # type: bool
120     QC_TO_DELTA = -7.501E-4 # type: float
121     DELTA_TO_QC = 1.0 / QC_TO_DELTA # type: float
122     maxAngle = 29 # type: int
123     minAngle = -maxAngle
124     dataDir = "./data/" # type: str
125     errorDetected = False # type: bool
126
127     def emcyErrorPrint(self, EmcyError):
128         """Print any EMCY Error Received on CAN BUS
129         """
130         logging.info('{0} Got an EMCY message: {1}'.format(
131             sys._getframe().f_code.co_name, EmcyError))
132         if EmcyError.code is 0:
133             self.errorDetected = False
134         else:
135             self.errorDetected = True
136         return
137
138     def getQcPosition(self, delta):
139         """ Converts angle of wheels to qc
140
141             Given the desired angle of wheels, in degrees of the bicycle model of car,
142             convert the requested value to qc position of steering wheel using the
143             calibration performed at beginning.
144
145             Args:
146                 delta: desired angle of wheels in degrees.
147             Returns:
148                 int: a rounded integer with qc position estimated or None if not possible
149                 """
150             if not self.calibrated:
151                 self.logInfo('Device is not yet calibrated')
152                 return None
153             if delta > self.maxAngle:
154                 self.logInfo('Angle exceeds limits: maxAngle: {0}\t requested: {1}'.format(
155                     self.maxAngle,
156                     delta))
157                 return None
158             if delta < self.minAngle:
159                 self.logInfo('Angle exceeds limits: minAngle: {0}\t requested: {1}'.format(
160                     self.minAngle,
161                     delta))
162                 return None
163             # perform calculations y = mx + b
164             val = delta * self.DELTA_TO_QC + self.zeroRef
165             val = round(val)
166             return int(val)
167
168     def getDeltaAngle(self, qc):
169         """ Converts qc of steering wheel to angle of wheel
170
171             Given the desired qc steering position, convert the requested value to angle of bicycle model
172             in degrees.

```

```

172
173     Args:
174         qc: an int with desired qc position of steering wheel.
175     Returns:
176         double: estimated angle of wheels in degrees or None if not possible
177     """
178     if not self.calibrated:
179         self.logInfo('Device is not yet calibrated')
180         return None
181
182     # perform calculations y = mx + b and solve to x
183     delta = (qc - self.zeroRef) * self.QC_TO_DELTA
184     return float(delta)
185
186 def saveToFile(self, filename=None, exitFlag=None):
187     """Record qc positions into a csv file
188
189     The following fields will be recorded
190
191     +-----+-----+-----+
192     | time | position | angle |
193     +-----+-----+-----+
194     | t1   | p1       | a1      |
195     +-----+-----+-----+
196     | ...  | ...       | ...      |
197     +-----+-----+-----+
198     | tN   | pN       | aN      |
199     +-----+-----+-----+
200
201     An additional file with same name but with ext TXT will have the current
202     calibration parameters
203
204     * minValue
205     * maxValue
206     * zeroRef
207
208     If filename is not supplied or already used, the current asctime() will
209     be used as filename.
210
211     Args:
212         filename: name of the file to save the data
213         exitFlag: threading event flag to signal exit.
214     """
215     # check if inputs were supplied
216     if not exitFlag:
217         self.logInfo('Error: exitFlag must be supplied')
218         return
219     # make sure is clear.
220     if exitFlag.isSet():
221         exitFlag.clear()
222     # -----
223     # Confirm epos is in a suitable state for free movement
224     # -----
225     stateID = self.checkEposState()
226     # failed to get state?
227     if stateID is -1:
228         self.logInfo('Error: Unknown state')
229         return

```

```

230     # If epos is not in disable operation at least,
231     # motor is expected to be blocked
232     if stateID > 4:
233         self.logInfo('Not a proper operation mode: {0}'.format(
234             self.state[stateID]))
235         if not self.changeEposState('shutdown'):
236             self.logInfo('Failed to change Epos state to shutdown')
237             return
238         self.logInfo('Successfully changed Epos state to shutdown')
239     # all ok, proceed
240     if not filename:
241         filename = time.asctime()
242         # windows has a problem with ':' in the name of files
243         # replace it by underscores
244         filename = filename.replace(':', '_')
245         filename = filename.replace(' ', '_')
246
247     # make dir if not already made
248     pathlib.Path(self.dataDir).mkdir(parents=True, exist_ok=True)
249     my_file = pathlib.Path(self.dataDir + filename + '.csv')
250
251     # open the parameters file first
252     my_file = open(self.dataDir + filename + '.txt', 'w')
253     print("minValue = {0}\nmaxValue = {1}\nzeroRef = {2}".format(self.minValue,
254                                                               self.maxValue, self.zeroRef),
255           file=my_file,
256           flush=True)
257     my_file.close()
258
259     # open the csv file
260     my_file = open(self.dataDir + filename + '.csv', 'w')
261     writer = csv.DictWriter(my_file, fieldnames=[
262         'time', 'position', 'angle'])
263     writer.writeheader()
264
265     # -----
266     # start requesting for positions of sensor
267     # -----
268
269     # var to store number of fails
270     numFails = 0
271
272     # get current time
273     t0 = time.monotonic()
274
275     while not exitFlag.isSet():
276         currentValue, OK = self.readPositionValue()
277         tOut = time.monotonic() - t0
278
279         if not OK:
280             self.logInfo('Failed to request current position')
281             numFails = numFails + 1
282
283         else:
284             writer.writerow({'time': tOut, 'position': currentValue,
285                             'angle': self.getDeltaAngle(currentValue)})
286
287             # sleep?
288             time.sleep(0.01)
289
290     self.logInfo('Finishing collecting data with {0} fail readings'.format(
291         numFails))
292     my_file.close()
293
294
295     def readFromFile(self, filename=None, useAngle=False):

```

```

287     """Read qc positions from file and follow them
288
289     The file must contain time and position in quadrature positions of steering
290     wheel and angle (degrees) of "center" wheel of bicycle model in a csv style
291
292     +-----+-----+-----+
293     | time | position | angle |
294     +-----+-----+-----+
295     | t1   | p1       | a1     |
296     +-----+-----+-----+
297     | ...  | ...       | ...    |
298     +-----+-----+-----+
299     | tN   | pN       | aN     |
300     +-----+-----+-----+
301
302     If the calibration value is not the same, user should set useAngle flag.
303     Because the calibration when the file was created can differ from current
304     calibration, user should set the useAngle flag to calculate the position
305     reference to send for device via the given angle.
306
307     Args:
308         filename: csv file to be read.
309         useAngle: use the angle value instead of position.
310     """
311
312     self.logInfo('Filename is {}'.format(filename))
313     # get current state of epos
314     state = self.checkEposState()
315     if state is -1:
316         self.logInfo('Error: Unknown state')
317         return
318
319     if state is 11:
320         # perform fault reset
321         if not self.changeEposState('fault reset'):
322             self.logInfo('Error: Failed to change state to fault reset')
323             return
324     # get current op mode
325     opMode, Ok = self.readOpMode()
326     if not Ok:
327         logging.info('Failed to request current OP Mode')
328         return
329     # show current op mode
330     self.logInfo('Current OP Mode is {}'.format(
331         self.opModes[opMode]
332     ))
333     # check if mode is position
334     if opMode is not -1:
335         if not self.setOpMode(-1):
336             self.logInfo('Failed to change opMode to {}'.format(
337                 self.opModes[-1]
338             ))
339             return
340         else:
341             self.logInfo('OP Mode is now {}'.format(
342                 self.opModes[-1]
343             ))
344     # shutdown
345     if not self.changeEposState('shutdown'):

```

```

345         self.logInfo('Failed to change Epos state to shutdown')
346         return
347     # switch on
348     if not self.changeEposState('switch on'):
349         self.logInfo('Failed to change Epos state to switch on')
350         return
351     if not self.changeEposState('enable operation'):
352         self.logInfo('Failed to change Epos state to enable operation')
353         return
354
355     # check if file exist
356     my_file = pathlib.Path(filename)
357     if not my_file.exists():
358         self.logInfo('File does not exist: {}'.format(
359             my_file))
360         return
361
362     # open csv file and read all values.
363     with open(filename) as csvfile:
364         reader = csv.DictReader(csvfile, delimiter=',')
365         I = 0 # line number
366         for row in reader:
367             tTarget = float(row['time']) # type: float
368             if useAngle:
369                 angle = float(row['angle'])
370                 if angle is not None: # if angle exceed limits, do not update
371                     position = self.getQcPosition(angle)
372                 else:
373                     position = int(row['position'])
374
375             # align to the first position before starting
376             if I is 0:
377                 self.moveToPosition(position)
378                 try:
379                     input("Press any key when ready...")
380                 except KeyboardInterrupt as e:
381                     self.logInfo('Got exception {}... exiting now'.format(e))
382                     # shutdown
383                     if not self.changeEposState('shutdown'):
384                         self.logInfo('Failed to change Epos state to shutdown')
385                     return
386             numFails = 0
387             t0 = time.monotonic()
388             lastRead = 0
389         else:
390             # if is not the first position but tOut is not yet tTarget
391             # sleep
392             # skip to next step?
393             while True:
394                 tOut = time.monotonic() - t0
395                 # is time to send new values?
396                 if tOut > tTarget:
397                     # time to update
398                     if not self.setPositionModeSetting(position):
399                         numFails = numFails + 1
400                         break
401                     # if we are not sending new targets, request current value to see the error
402                     if tOut - lastRead > 0.051:

```

```

403             aux, OK = self.readPositionValue()
404             if not OK:
405                 self.logInfo('Failed to request current position')
406                 numFails = numFails + 1
407             else:
408                 # does error have grown to much?
409                 ref_error = position - aux
410                 if abs(ref_error) > self.maxFollowingError:
411                     self.logInfo(
412                         'Error is growing to much. Something seems wrong')
413                     print('time={0:+08.3f}\tIn={1:+05}\tOut={2:+05}\tError={3:+05}'.format(
414                         tOut, position, aux, ref_error))
415                     if not self.changeEposState('shutdown'):
416                         self.logInfo(
417                             'Failed to change Epos state to shutdown')
418                     return
419                 # for debug print every time on each cycle
420                 print('time={0:+08.3f}\tIn={1:+05}\tOut={2:+05}\tError={3:+05}'.format(
421                         tOut, position, aux, ref_error))
422                 lastRead = tOut
423                 time.sleep(0.001)
424                 I = I + 1 # increase line number
425                 if self.errorDetected:
426                     break
427
428                 if self.errorDetected:
429                     self.logInfo('Exited with emergency error')
430                 else:
431                     self.logInfo('All done: Time to process all vars was {0} seconds with {1} fail
readings'.format(
432                         time.monotonic() - t0, numFails))
433                     self.logInfo('Expected time to process {0}'.format(tTarget))
434                     if not self.changeEposState('shutdown'):
435                         self.logInfo('Failed to change Epos state to shutdown')
436                     return
437
438     def startCalibration(self, exitFlag=None):
439         """Perform steering wheel calibration
440
441             This function is expected to be run on a thread in order to find the limits
442             of the steering wheel position and find the expected value of the zero angle
443             of wheels.
444
445             Args:
446                 exitFlag: threading.Event() to signal the finish of acquisition
447
448             """
449             # check if inputs were supplied
450             if not exitFlag:
451                 self.logInfo('Error: check arguments supplied')
452                 return
453             stateID = self.checkEposState()
454             # -----
455             # Confirm epos is in a suitable state for free movement
456             # -----
457             # failed to get state?

```

```

458     if stateID is -1:
459         self.logInfo('Error: Unknown state')
460         return
461     # If epos is not in disable operation at least, motor is expected to be blocked
462     if stateID > 4:
463         self.logInfo('Not a proper operation mode: {0}'.format(
464             self.state[stateID]))
465         # shutdown
466         if not self.changeEposState('shutdown'):
467             self.logInfo('Failed to change state to shutdown')
468             return
469         self.logInfo('Successfully changed state to shutdown')
470
471     maxValue = 0
472     minValue = 0
473     numFails = 0
474     # -----
475     # start requesting for positions of sensor
476     # -----
477     while not exitFlag.isSet():
478         currentValue, OK = self.readPositionValue()
479         if not OK:
480             self.logDebug('Failed to request current position')
481             numFails = numFails + 1
482         else:
483             if currentValue > maxValue:
484                 maxValue = currentValue
485             if currentValue < minValue:
486                 minValue = currentValue
487             # sleep?
488             time.sleep(0.01)
489
490         self.logInfo(
491             'Finished calibration routine with {0} fail readings'.format(numFails))
492         self.minValue = minValue
493         self.maxValue = maxValue
494         self.zeroRef = round((maxValue - minValue) / 2.0 + minValue)
495         self.calibrated = 1
496         self.logInfo('MinValue: {0}, MaxValue: {1}, ZeroRef: {2}'.format(
497             self.minValue, self.maxValue, self.zeroRef
498         ))
499     return
500
501 def moveToPosition(self, pFinal, isAngle=False):
502     """Move to desired position.
503
504     Plan and apply a motion profile with low jerk, limited max speed, max acceleration in order
505     to avoid abrupt variations.
506     The function implement the algorithm developed in [1]_
507
508     Args:
509         pFinal: desired position.
510         isAngle: a boolean, true if pFinal is an angle or false if is qc value
511     :return:
512
513     .. [1] Li, Huaizhong & M Gong, Z & Lin, Wei & Lippa, T. (2007). Motion profile planning for
514     reduced jerk and vibration residuals. 10.13140/2.1.4211.2647.
515     """

```

```

515     # constants
516     # Tmax = 1.7 seems to be the limit before oscillations.
517     Tmax = 0.2 # max period for 1 rotation;
518     # 1 rev = 3600*4 [qc]
519     countsPerRev = 3600 * 4
520     #
521     # 1Hz = 60rpm = 360degrees/s
522     #
523     # 360 degrees = sensor resolution * 4
524     #
525     # this yields: 1Hz = (sensor resolution * 4)/s
526     #
527     # Fmax = 1 / Tmax;
528     #
529     # maxSpeed = 60 rpm/Tmax [rpm]=
530     #           = 360degrees/Tmax [degrees/s]=
531     #           = (sensor resolution *4)/Tmax [qc/s]
532
533     maxSpeed = countsPerRev / Tmax # degrees per sec
534
535     # max acceleration must be experimental obtained.
536     # reduced and fixed.
537     maxAcceleration = 6000.0 # [qc]/s^2
538
539     # maximum interval for both the acceleration and deceleration phase are:
540     T1max = 2.0 * maxSpeed / maxAcceleration # type: float
541
542     # the max distance covered by these two phase (assuming acceleration equal
543     # deceleration) is 2* 1/4 * Amax * T1max^2 = 1/2 * Amax * T1max^2 = 2Vmax^2/Amax
544     maxL13 = 2.0 * maxSpeed ** 2 / maxAcceleration # type: float
545
546     # max error in quadrature counters
547     MAXERROR = 7500
548     numFails = 0
549     # is device calibrated?
550     if not self.calibrated:
551         self.logInfo('Device is not yet calibrated')
552         return False
553     # is position requested an angle?
554     if isAngle:
555         pFinal = self.getQcPosition(pFinal)
556         # if position can not be calculated, alert user.
557         if pFinal is None:
558             self.logInfo('Failed to calculate position value')
559             if not self.changeEposState('shutdown'):
560                 self.logInfo('Failed to change Epos state to shutdown')
561             return False
562
563         if pFinal > self.MaxValue or pFinal < self.MinValue:
564             self.logInfo('Final position exceeds physical limits')
565             return False
566
567         pStart, OK = self.readPositionValue()
568         numFails = 0
569         if not OK:
570             self.logInfo('Failed to request current position')
571             while numFails < 5 and not OK:
572                 pStart, OK = self.readPositionValue()

```

```

573         if not OK:
574             numFails = numFails + 1
575         if numFails == 5:
576             self.logInfo(
577                 'Failed to request current position for 5 times... exiting')
578             return False
579
580     # -----
581     # get current state of epos and change it if necessary
582     # -----
583     state = self.checkEposState()
584     if state is -1:
585         self.logInfo('Error: Unknown state')
586         return False
587
588     if state is 11:
589         # perform fault reset
590         ok = self.changeEposState('fault reset')
591         if not ok:
592             self.logInfo('Error: Failed to change state to fault reset')
593             return False
594
595     # shutdown
596     if not self.changeEposState('shutdown'):
597         self.logInfo('Failed to change Epos state to shutdown')
598         return False
599     # switch on
600     if not self.changeEposState('switch on'):
601         self.logInfo('Failed to change Epos state to switch on')
602         return False
603     if not self.changeEposState('enable operation'):
604         self.logInfo('Failed to change Epos state to enable operation')
605         return False
606     # -----
607     # Find remaining constants
608     # -----
609     # absolute of displacement
610     l = abs(pFinal - pStart)
611     if l is 0:
612         # already in final point
613         return True
614     # do we need a constant velocity phase?
615     if l > maxL13:
616         T2 = 2.0 * (l - maxL13) / (maxAcceleration * T1max)
617         T1 = T1max
618         T3 = T1max
619     else:
620         T1 = np.sqrt(2 * l / maxAcceleration)
621         T2 = 0.0
622         T3 = T1
623
624     # time constants
625     t1 = T1
626     t2 = T2 + t1
627     t3 = T3 + t2 # final time
628
629     # allocate vars
630     inVar = np.array([], dtype='int32')

```

```

631     outVar = np.array([], dtype='int32')
632     tin = np.array([], dtype='int32')
633     tout = np.array([], dtype='int32')
634     ref_error = np.array([], dtype='int32')
635
636     # determine the sign of movement
637     moveUp_or_down = np.sign(pFinal - pStart)
638     flag = True
639     pi = np.pi
640     cos = np.cos
641     time.sleep(0.01)
642     # choose monotonic for precision
643     t0 = time.monotonic()
644     numFails = 0
645     while flag and not self.errorDetected:
646         # request current time
647         tin = np.append(tin, [time.monotonic() - t0])
648         # time to exit?
649         if tin[-1] > t3:
650             flag = False
651             inVar = np.append(inVar, [pFinal])
652             self.setPositionModeSetting(pFinal)
653             # reading a position takes time, as so, it should be enough
654             # for it reaches end value since steps are expected to be
655             # small
656             aux, OK = self.readPositionValue()
657             if not OK:
658                 self.logInfo('Failed to request current position')
659                 numFails = numFails + 1
660             else:
661                 outVar = np.append(outVar, [aux])
662                 tout = np.append(tout, [time.monotonic() - t0])
663                 ref_error = np.append(ref_error, [inVar[-1] - outVar[-1]])
664             # not finished
665             else:
666                 # get reference position for that time
667                 if tin[-1] <= t1:
668                     aux = pStart + \
669                         moveUp_or_down * maxAcceleration / 2.0 * (T1 / (2.0 * pi)) ** 2 * \
670                         (1 / 2.0 * (2.0 * pi / T1 *
671                             tin[-1]) ** 2 - (1.0 - cos(2.0 / T1 * pi * tin[-1])))
672                 else:
673                     if (T2 > 0 and tin[-1] > t1 and tin[-1] <= t2):
674                         aux = pStart + \
675                             moveUp_or_down * \
676                             (1 / 4.0 * maxAcceleration * T1 ** 2 + 1 /
677                             2.0 * maxAcceleration * T1 * (tin[-1] - t1))
678                 else:
679                     aux = pStart + \
680                         moveUp_or_down * (1 / 4.0 * maxAcceleration * T1 ** 2
681                             + 1 / 2.0 * maxAcceleration * T1 * T2 +
682                             maxAcceleration / 2.0 *
683                             (T1 / (2.0 * pi)) ** 2
684                             * ((2.0 * pi) ** 2 * (tin[-1] - t2) / T1 - 1 / 2.0 *
685                             (2.0 * pi / T1
686
687             * (tin[
688

```

```

-1] - t2)) ** 2 + (
687 )                                     1.0 - cos(2.0 * pi / T1 * (tin[-1] - t2)))
688
689     aux = round(aux)
690     # append to array and send to device
691     inVar = np.append(inVar, [aux])
692     OK = self.setPositionModeSetting(np.int32(inVar[-1]).item())
693     if not OK:
694         self.logInfo('Failed to set target position')
695     numFails = numFails + 1
696     aux, OK = self.readPositionValue()
697     if not OK:
698         self.logInfo('Failed to request current position')
699     numFails = numFails + 1
700     else:
701         outVar = np.append(outVar, [aux])
702         tout = np.append(tout, [time.monotonic() - t0])
703         ref_error = np.append(ref_error, [inVar[-1] - outVar[-1]])
704         if abs(ref_error[-1]) > MAXERROR:
705             self.changeEposState('shutdown')
706             self.logInfo(
707                 'Something seems wrong, error is growing to much!!!!')
708             return False
709             # require sleep?
710             time.sleep(0.005)
711             self.logInfo('Finished with {} fails'.format(numFails))
712             return True
713
714 def main():
715     """EPOS controller tester.
716
717     Simple program to perform a few tests with EPOS device in the car
718     """
719
720     import argparse
721     from time import sleep
722     if sys.version_info < (3, 0):
723         print("Please use python version 3")
724         return
725
726     parser = argparse.ArgumentParser(add_help=True,
727                                     description='Epos controller')
728     parser.add_argument('--channel', '-c', action='store', default='can0',
729                         type=str, help='Can channel to be used', dest='channel')
730     parser.add_argument('--bus', '-b', action='store',
731                         default='socketcan', type=str, help='Bus type', dest='bus')
732     parser.add_argument('--rate', '-r', action='store', default=None,
733                         type=int, help='bitrate, if applicable', dest='bitrate')
734     parser.add_argument('--nodeID', action='store', default=1, type=int,
735                         help='Node ID [ must be between 1- 127]', dest='nodeID')
736     parser.add_argument('--objDict', action='store', default=None,
737                         type=str, help='Object dictionary file', dest='objDict')
738     args = parser.parse_args()
739
740     # set up logging to file - see previous section for more details
741     logging.basicConfig(level=logging.INFO,
742                         format='[%s.%03d] [%s]: %s' % (asctime)s, %msecs)03d] [%s(name)-20s]: %s(%s) %s',

```

```

743         datefmt='%d-%m-%Y %H:%M:%S',
744         filename='mqtt_controller.log',
745         filemode='w')
746
747 # define a Handler which writes INFO messages or higher in console
748 #
749 console = logging.StreamHandler()
750 console.setLevel(logging.INFO)
751
752 # set a format which is simpler for console use
753 formatter = logging.Formatter('%(name)-20s: %(levelname)-8s %(message)s')
754 # tell the handler to use this format
755 console.setFormatter(formatter)
756
757 # add the handler to the root logger
758 logging.getLogger('').addHandler(console)
759
760
761 # event flag to exit
762 exitFlag = threading.Event()
763
764
765 # TODO: in order to be able to use EPOS with the SINAMCIS, network must be shared
766 # For now, create the network from scratch
767 network = canopen.Network()
768
769 try:
770     network.connect(channel=args.channel, bustype=args.bus)
771 except Exception as e:
772     logging.info('Exception caught:{0}'.format(str(e)))
773
774 # instantiate object
775 epos = Epos_controller(_network=network)
776
777 # declare threads
778 eposThread = threading.Thread(name="EPOS", target=epos.startCalibration,
779                               kwargs={'exitFlag': exitFlag})
780
781
782 if not (epos.begin(args.nodeID, objectDictionary=args.objDict)):
783     logging.info('Failed to begin connection with EPOS device')
784     logging.info('Exiting now')
785     return
786
787 # emcy messages handles
788 epos.node.emcy.add_callback(epos.emcyErrorPrint)
789
790 # change default values for canopen sdo settings
791
792 epos.node.sdo.MAX_RETRIES = 2
793 epos.node.sdo.PAUSE_BEFORE_SEND = 0.005
794 epos.node.sdo.RESPONSE_TIMEOUT = 0.01
795
796
797 # test connection
798
799 numFails = 0
800 _, success = epos.readStatusWord()
801 while not success and numFails < 5:
802     numFails = numFails + 1
803     sleep(0.1)
804     _, success = epos.readStatusWord()
805
806 # any success?
807 if numFails is 5:
808     logging.info('Failed to contact EPOS... is it connected? Exiting')
809     return
810
811 # change PID settings

```

```

801 # -----
802 # default values were 52, 1, 15
803 # last used values 54, 1, 3
804 epos.setPositionControlParameters(pGain=250, iGain=1, dGain=50)
805 # show current Position control parameters
806 epos.printPositionControlParameters()
807
808 try:
809     eposThread.start()
810     print("Please move steering wheel to extreme positions to calibrate...")
811     input("Press Enter when done...\n")
812 except KeyboardInterrupt as e:
813     exitFlag.set()
814     eposThread.join()
815     logging.warning('[Main] Got exception {0}... exiting now'.format(e))
816     return
817
818 exitFlag.set()
819 eposThread.join()
820 if epos.calibrated == -1:
821     logging.info("[Main] Failed to perform calibration")
822     return
823 if epos.calibrated == 0:
824     logging.info("[Main] Calibration not yet done")
825     return
826 # reset event()
827 exitFlag.clear()
828 # create software position limits?
829 # TODO: define max and min
830
831 print("-----")
832 print("Max Value: {0}\nMin Value: {1}\nZero Ref: {2}".format(
833     epos.MaxValue, epos.MinValue, epos.zeroRef))
834 print("-----")
835 print("Moving into Zero Ref position....")
836 epos.moveToPosition(epos.zeroRef)
837 print('Done!')
838 print("-----")
839
840 # -----
841 # Menu definitions
842 #
843 mainMenuSaveQC = Button("Save qc to file", 1)
844 mainMenuReadQC = Button("Follow qc from file", 2)
845 mainMenuMove = Button("Move to position", 3)
846 mainMenuShowConfig = Button("Show config", 4)
847 mainMenuQuit = Button("Quit", 0)
848 mainMenuButtons = [mainMenuSaveQC,
849                     mainMenuReadQC, mainMenuMove, mainMenuShowConfig, mainMenuQuit]
850 mainMenu = Menu("Main menu", mainMenuButtons)
851 stopCycle = False
852 try:
853     while not stopCycle:
854         val = mainMenu.display()
855         if val is not None:
856             if val is 0:
857                 # exit program
858                 stopCycle = True

```

```

859     elif val is 1:
860         # save qc to a file to be used later
861         eposThread = threading.Thread(name="Save QC",
862                                         target=epos.saveToFile,
863                                         kwargs={'exitFlag': exitFlag})
864         eposThread.start()
865         print("Recording to file.")
866         input("Press Enter when done...\n")
867         exitFlag.set()
868         eposThread.join()
869     elif val is 2:
870         # get latest file in data dir
871         directory = pathlib.Path('./data/')
872         _, file_path = max((f.stat().st_mtime, f)
873                             for f in directory.iterdir())
874         epos.readFile(str(file_path), useAngle=True)
875     elif val is 3:
876         try:
877             x = int(input("Enter desired position [qc]: "))
878             print(
879                 '-----')
880             print('Moving to position {0:+16,}'.format(x))
881             epos.moveToPosition(x)
882             print('done')
883             print(
884                 '-----')
885             # shutdown
886             if not epos.changeEposState('shutdown'):
887                 logging.info(
888                     '[Main] Failed to change Epos state to shutdown')
889             except KeyboardInterrupt as e:
890                 logging.info(
891                     '[Main] Got exception {0}... exiting now'.format(e))
892         elif val is 4:
893             print("Show configurations:")
894             epos.printPositionControlParameters()
895             epos.printMotorConfig()
896             epos.printSensorConfig()
897             input("Press any key to continue...\n")
898         else:
899             pass
900
901     except KeyboardInterrupt as e:
902         logging.info('[Main] Got exception {0}... exiting now'.format(e))
903     finally:
904         exitFlag.set() # in case any thread is still working
905         epos.disconnect()
906     return
907
908
909 if __name__ == '__main__':
910     main()

```

appendix/steering\_controller.py