VIENA

Veículo Inteligente Elétrico de Navegação Autónoma

Documentation Guide

Current authors:

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

Revision: 1

September 10, 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   Contributtor 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

List of Figures

Nomenclature

$^b_w R$ 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).

Acronyms

9DOF   Nine Degrees of Freedom.

FST Lisboa Formula Student Team Lisboa.

GNSS   Global Navigation Satellite System.

GPS   Global Position System.

IMU   Inertial Measurement Unity.

IST   Instituto Superior Técnico.

MAC   Media Access Control.

MARG   Magnetic Angular Rate and Gravity.

PCB   Printed Circuit Board.

Rpi   Raspberry PI.

SBC   Single Board Computer.

SPI   Serial Peripheral Interface bus.

VIENA   Veiculo Inteligente Elétrico de Navegação Autónoma.

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 four main chapters, server, main sensors, controllers, miscellaneous. 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 CANBus 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 inertial 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 developement for invertor received from FST Lisboa (in progress)
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 and/of check status.

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

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.

2.1   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.

2.2   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 [1].
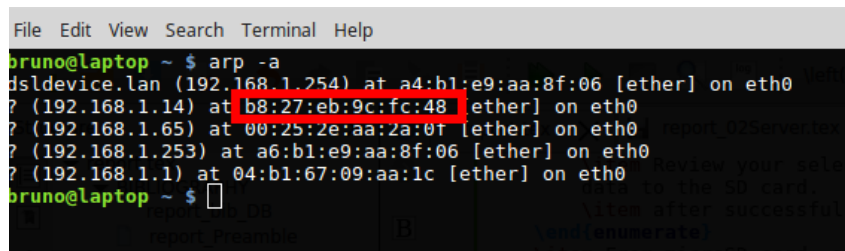
1. Download the OS image, Raspbian Lite version in here
2. Connect the MicroSD to host computer.
3. Open Etcher and:

(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

2.2.1 Remote access

The next step is to find the ip address of the Rpi. For example, in linux terminal type `arp -a` . In figure 2.1 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 wich correspond to organizationally unique identifier [2] associated to Raspberry Pi Foundation, B8:27:EB [3].

Figure 2.1: arp -a command example output

Perform the first login with using the default login details as shown in table 2.1. 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.

username:	pi
password:	raspberry

Table 2.1: Default login details for Rpi

2.2.2 Initial update and configuration

If the user as 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 2.2.
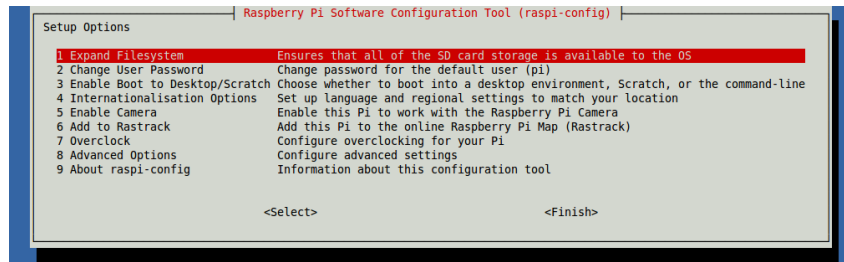
Figure 2.2: Raspi-config example output

- Set Keyboard Layout

- Update to lastest software

- Set Timezone

- Set language [optional]

- Change default login password

- Configure Wifi-zone [if present]

- Enable Serial Peripheral Interface bus (SPI) for CAN controller

- Enable Inter-Integrated Circuit bus (I^2C) for RTC

- Set hostname

- change default password

- change memory for GPU

The current settings are presented in table 2.2.

Username:	pi
Password:	fiatelettra
Hostname:	raspberrypi
Language:	en_GB.UTF-8 UTF-8
Locale and Timezone:	Lisbon
Keyboard layout:	pt_PT
SPI:	on
I2C:	on
Memory split:	16 (minimum since is running headless)

Table 2.2: Suggested details for Rpi

2.3 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 ehternet connection is ok using `ifconfig`

Chapter 3

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 sugestions:

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. Documentation and guides to become familiar with it can be found at try.github.io

Python3

Since Python core team is dropping support for python 2.7.x in 2020 [4] and some of important package developers are also dropping support for it [5], 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 [6]. 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 3.1, 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 struture, if needed [7]. Installation of requirements is done by running the following command `pip install -r requirements.txt`

Use Google style docstrings

A good documentation is a key point for keeping good readability of a project. Docstrings format from Google Style Guide [8] is adopted.

Use sphinx with automation

Similar to previous point, sphinx will allow the auto documentation generation for code. Sphinx is a popular tool for creating documentation and is used also in 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 branchs)

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 3.1: Activation/deactivation of venv OS specific

Chapter 4

Batteries

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   Main battery packs

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

NovatelOEM4 GPS Library

Date 24 Jul 2018

Version 0.4

Author Bruno Tibério

Contact bruno.tiberio@tecnico.ulisboa.pt

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 wich 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:

gps

Bibliography

[1] Raspberry Pi Foundation. Installing operating systems images, April 2018. URL https://www.raspberrypi.org/documentation/installation/installing-images/README.md. (2018).

[2] Wikipedia. Mac address, 2018. URL https://en.wikipedia.org/wiki/MAC_address. (2018).

[3] Wireshark. Oui lookup tool, 2018. URL https://www.wireshark.org/tools/oui-lookup.html. (2018).

[4] Python Software Foundation. Status of python branches, 2018. URL https://devguide.python.org/#status-of-python-branches. (2018).

[5] Numpy Developers. Plan for dropping python 2.7 support, 2017. URL https://www.numpy.org/neps/nep-0014-dropping-python2.7-proposal.html. (2018).

[6] Python Software Foundation. Python 3 documentation, 2018. URL https://docs.python.org/3/. (2018).

[7] 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. (2018).

[8] Google Python Team. Google python style guide, 2018. URL https://github.com/google/styleguide/blob/gh-pages/pyguide.md#38-comments-and-docstrings. (2018).