Planning, Development and Implementation of Monitoring System for Remotely Controlled Car

Tiago da Silva Barra

Thesis to obtain the Master of Science Degree in Electronics Engineering

Supervisor: Prof. Pedro Miguel Pinto Ramos

Examination Committee

Chairperson: Prof. Pedro Manuel Brito da Silva Girão
Supervisor: Prof. Pedro Miguel Pinto Ramos
Members of the Committee: Prof. Nuno Filipe Valentim Roma

April 2019
Things don’t have to change the world to be important.

Steve Jobs
Acknowledgments

It is my radiant sentiment to place on record my best regards, deepest sense of thankfulness to my friends, for always making things look easy and for creating discussions that help ideas emerge, my kind thank you.

To my colleagues, my words for all the invaluable constructive criticism and friendly advice throughout this work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

Special thanks to my family. Words cannot express how grateful I am to my mother and father: Thank you for all of the sacrifices that you have made on my behalf.

To my supervisor, Prof. Pedro M. Ramos, I would like to express my sincere gratitude for all the guidance and teaching, for the continuous support, for their patience motivation and massive knowledge. I also want to thank the Instituto de Telecomunicações for the conditions and materials provided for the completion of dissertation.
Abstract

This document describes the development of a telemetric system for a remote and autonomous controlled miniature car. Telemetry is used to obtain information about the operation of the car acquiring periodically data from a set of sensors installed in the chassis. The system is built based on a Raspberry Pi 3, which manages the acquisition of data and controls the movements of the car based on the values acquired or user settings.

The system has two operating modes: (i) remotely controlled mode where the user decides the direction and speed of the car, and (ii) autonomous mode where the car follows a line drawn on a lane without user intervention. Telemetric data is sent by Bluetooth to an Android application where the user can view the acquired data through a visual interface. In the application it is possible to define the modes of operation and in the controlled mode it is also possible to control the speed and direction of the car.

The telemetry of the vehicle is obtained by sensors that measure the speed, acceleration, driven distance, luminosity and temperature on the outside of the car. In autonomous mode, the reflected light on the lane is used to define the path to follow. The values acquired by the system are sent from the Raspberry Pi to a Wi-Fi connected virtual machine part of a private ad hoc network. In this Ubuntu virtual machine, the data is received by TCP sockets and later presented on a web page and stored in a text file for future analysis.

Keywords

Car telemetry, monitoring system, sensors, data acquisition, remote control, virtual machine, Android application.
Resumo

Este documento descreve o desenvolvimento de um sistema telemétrico para um carro miniatura remoto e autonomamente controlado. A telemetria é usada para obter informação sobre o funcionamento do carro, através da aquisição periódica de dados provenientes de um conjunto de sensores instalados no chassis. O sistema é baseado num Raspberry Pi 3, que gere a aquisição dos dados e controla os movimentos do carro baseando-se nos valores adquiridos ou nas definições do utilizador.

O sistema tem dois modos de funcionamento: (i) o modo remotamente controlado, onde o utilizador decide a direção e velocidade do carro, e (ii) o modo autónomo, onde o carro segue uma linha desenhada numa pista sem intervenção do utilizador. Os dados da telemetria são enviados por Bluetooth para uma aplicação Android onde o utilizador tem uma interface concebida para a visualização dos dados. Na aplicação é possível definir os modos de funcionamento, sendo que no modo controlado é também possível controlar a velocidade e direção do carro.

A telemetria do veículo é obtida por sensores que medem a velocidade, aceleração, distância percorrida, luminosidade e temperatura no exterior do carro. No modo autónomo, a luz refletida na pista é usada para saber qual a trajetória a seguir. Os valores adquiridos pelo sistema são enviados para uma máquina virtual ligada por Wi-Fi pertencente de uma rede ad hoc privada. Nesta máquina virtual Ubuntu, os dados são recebidos através de sockets TCP sendo posteriormente apresentados numa página web e guardados num ficheiro de texto para futura análise.

Palavras Chave

Sistema telemétrico, sistema de monotorização, sensores, aquisição de dados, controlo remoto, máquina virtual, aplicação Android.
Contents

1 Introduction ................................................. 1
   1.1 Purpose and motivation .................................... 2
   1.2 Goals and challenges ....................................... 4
   1.3 Document organization ..................................... 4

2 Related Work ................................................. 7
   2.1 SUBA Project .............................................. 8
      2.1.1 Sensors .................................................. 9
      2.1.2 Motors .................................................. 10
      2.1.3 Consumption ........................................... 11
   2.2 A Sensor Network for the Internet of Things .......... 11
      2.2.1 Architecture ........................................... 11
   2.3 Software Architecture for Autonomous Vehicles .... 13
      2.3.1 Architecture ........................................... 13
      2.3.2 Console ................................................ 14

3 Telemetry System ............................................. 17
   3.1 Conversion and communication .............................. 18
      3.1.1 Analog-to-digital conversion ......................... 18
      3.1.2 Serial Peripheral Interface .......................... 19
      3.1.3 Inter-Integrated Circuit .............................. 20
   3.2 Luminosity .................................................. 22
   3.3 Temperature ............................................... 23
   3.4 Hall Effect ............................................... 24
      3.4.1 Speed Calculation ..................................... 26
      3.4.2 Acceleration Calculation ............................. 26
   3.5 Light Reflected ............................................ 26
   3.6 Printed Circuit Board ..................................... 28

4 Car System ..................................................... 31
   4.1 Architecture ............................................... 32
   4.2 Drive motor ................................................ 33
# List of Figures

1.1 Original car used in this project and the remote control. ........................................ 2
1.2 Inside view of the car used in this project. .............................................................. 3
1.3 Raspberry Pi 3, Model B [1]. .................................................................................... 3

2.1 Vehicle used in the SUBA project [2]. .................................................................. 8
2.2 Differential sensor system of SUBA project [2]. ....................................................... 9
2.3 Steering system of SUBA project [2]. ................................................................... 10
2.4 Sprockets system of SUBA project’s motor [2]. ..................................................... 10
2.5 Hardware components [3]. ..................................................................................... 12
2.6 System overview [3]. ............................................................................................... 13
2.7 The basics of an autonomous vehicle software [4]. .................................................. 14
2.8 Typical architecture: multiple threads and shared memory [4]. ............................... 15

3.1 Circuit for the acquisition and conversion of sensor data [5]. ................................. 19
3.2 Circuit for stabilization of reference voltage. ........................................................... 19
3.3 SPI data communication [5]. .................................................................................. 20
3.4 I2C data communication [7]. .................................................................................. 21
3.5 I2C start and stop conditions [7]. .......................................................................... 21
3.6 I2C command when master writes to slave [7]. ...................................................... 22
3.7 Light dependent resistor circuit. ............................................................................. 22
3.8 Temperature measurement circuit. ........................................................................ 24
3.9 Hall effect measurement circuit. ............................................................................ 25
3.10 Speed measure system. ......................................................................................... 26
3.11 Light reflected measurement circuit. .................................................................... 27
3.12 Sensors array to follow a line: a) Design b) Implementation. ............................... 28
3.13 PCB shield assembled in Raspberry Pi. ................................................................. 29

4.1 Operational modes diagram. .................................................................................... 32
4.2 System communication architecture. ........................................................................ 33
4.3 Circuit used with motor and H-bridge. .................................................................... 34
4.4 Principle of operation of the PWM signal. ............................................................... 34
4.5 Control motor direction using H-bridge. ................................................................. 35
List of Tables

2.1 Consumptions of SUBA project’s car [2]. ........................................... 11

4.1 System consumption values. ............................................................. 40
# List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D</td>
<td>Three Dimensions</td>
</tr>
<tr>
<td>ABS</td>
<td>Acrylonitrile Butadiene Styrene</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog-to-Digital Converter</td>
</tr>
<tr>
<td>ARM</td>
<td>Advanced RISC Machine</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CS</td>
<td>Chip Select</td>
</tr>
<tr>
<td>DC</td>
<td>Direct Current</td>
</tr>
<tr>
<td>GPIO</td>
<td>General Purpose Input/Output</td>
</tr>
<tr>
<td>HDMI</td>
<td>High-Definition Multimedia Interface</td>
</tr>
<tr>
<td>I2C</td>
<td>Inter-Integrated Circuit</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>LDR</td>
<td>Light Dependent Resistor</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>MISO</td>
<td>Master In Slave Out</td>
</tr>
<tr>
<td>MIT</td>
<td>Massachusetts Institute of Technology</td>
</tr>
<tr>
<td>MOSI</td>
<td>Master Out Slave In</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PETG</td>
<td>Poly-Ethylene Terephthalate Glycol</td>
</tr>
<tr>
<td>PIGPIO</td>
<td>Raspberry Pi General Purpose Input/Output</td>
</tr>
<tr>
<td>PLA</td>
<td>Polylactic Acid</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse-Width Modulation</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RC</td>
<td>Remotely Controlled</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>SCLK</td>
<td>Serial Clock</td>
</tr>
<tr>
<td>SD</td>
<td>Secure Digital</td>
</tr>
<tr>
<td>SI</td>
<td>International System of Units</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>SS</td>
<td>Select Slave</td>
</tr>
<tr>
<td>SUBA</td>
<td>Seja Um Bom Aluno (in portuguese)</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
</tbody>
</table>
# 1 Introduction

<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Purpose and motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Goals and challenges</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Document organization</td>
<td>4</td>
</tr>
</tbody>
</table>
Introduction

1.1 Purpose and motivation

In the development of the telemetric system for a remote and autonomously controlled miniature car, the Remotely Controlled (RC) car used is model Ninco Parkracers Renault Sport. The chassis of this car is made of thermo-plastic and is fully protected in line with European toy regulations. The fact that it is made of plastic allows easy integration of processor and all sensors due to its hardness and low weight. As necessary, Polylactic Acid (PLA) parts are developed, printed on a Three Dimensions (3D) printer to assemble the system, making adjustments to the original model. The model, presented in Figure 1.1, comes equipped with soft plastic tires for improved grip on all surfaces. These characteristics allow it to be a model with good shock resistance, agility and ergonomic performance [8].

![Figure 1.1: Original car used in this project and the remote control.](image)

The original model, was equipped with a 2.4 GHz transmitter with a range of over thirty meters. As far as maximum speed was concerned, it can reach 5.55 m/s. The car is powered by a Li-Ion battery, with 500 mAh and 6.4 V, which provides an operating time of around 13 minutes of continuous use. This model was chose because have a large space available underneath the plastic body which may allow to introduce a large number of electronic systems inside, as shown in Figure 1.2.
One of the key points of this work is the data processing through a Raspberry Pi 3 (Model B), presented in Figure 1.3, a small computer with the size of a credit card but with significant processing capacity. This board has a Broadcom BCM2836 SoC chip in an Advanced RISC Machine (ARM) Cortex-A7 Quad-core architecture that can run at 900 MHz. The Random Access Memory (RAM) is 1 GB, which is perfectly sufficient for what is desired in this project [9]. On the board, four Universal Serial Bus (USB) ports are available. A keyboard and a mouse are connected to control the normal programming features. In addition, a Wi-Fi dongle is connected and is also used the Bluetooth adapter on board for remote connectivity.

There is an High-Definition Multimedia Interface (HDMI) port that allows an high definition visual interface. The connectivity, in a first phase, is made through the Ethernet port and then it will be done through the aforementioned Wi-Fi dongle. Forty General Purpose Input/Output (GPIO) pins are where sensors, actuators, integrated circuits and system power will be connected. And finally, there
is a Micro Secure Digital (SD) reader where the operating system that runs the system is placed \[10\].

The operating system used is Raspbian, a Debian variant optimized for Raspberry Pi hardware. This release contains the LXDE desktop environment, the OpenBox window manager, the Midori browser, software development tools, and sample source code for multimedia functions \[11\] \[12\].

During this project, the programming language used is Python. This is a very productive way to program for a fairly simple and easily readable syntax \[13\]. It has the advantage that many third-party open source libraries or other modules can be added, which gives a standard and well-built library, with some built-ins to enrich this programming language, making Python dynamically typed, and above all, flexible code with less detailed, like variables or classes \[14\] \[15\].

1.2 Goals and challenges

The main objective of this project is to design a monitoring system for a remotely controlled car that collects data from a set of sensors and present it in a mobile interface, display it in a web platform and save the data into a file for future analysis. The project has two modes of operation, an autonomous mode and a controlled by the user. This document consists of the following parts:

• Study of the commercial prototype, in order to implement the adaptations necessary for the purpose of the Dissertation.

• Define the architecture of the hardware to be developed, taking into account the dimensions, consumption and performance that are considered appropriate.

• Design the hardware and prototype it.

• Develop software capable of performing a fast processing of the data received by the sensors and activate actuators according to information received.

• Develop an Android application to operate modes and display sensor values to user.

• Design and implement key parts through design and modeling 3D, in Computer Aided Design (CAD) software.

• Create an ad hoc network to connect Raspberry Pi and a virtual machine where data are displayed to user through a web framework.

1.3 Document organization

This thesis is organized as follows:

• Chapter 1 is an introduction, giving an overview about the different work areas and in what consists the project.

• Chapter 2 is the related work where projects with similar characteristics or specifications are presented, described and analyzed.
• Chapter 3 is where the hardware is analysed. The sensors used and the build in shield are specified and explained.

• Chapter 4 is the chapter where the system is explained. The general architecture of the system is presented and all the integrating functionalities are explained.

• Chapter 5 is where user interface is presented. The Android application and the web page platform are described and explained.

• Chapter 6 is the conclusion and purpose for future work within the thesis.
2 Related Work

Contents

2.1 SUBA Project ......................... 8
2.2 A Sensor Network for the Internet of Things .... 11
2.3 Software Architecture for Autonomous Vehicles .... 13
Related Work

In this chapter, the aim is to present and analyze already existing works with similar characteristics and with relevant processes and approaches. The study of these systems allows to understand the properties of each one of them, being advantageous in the decision making in relation to the best way to implement the new project. The analyzed systems allow to find better solutions, thus guaranteeing that the objectives and specifications are achieved in a more organized and efficient way.

2.1 SUBA Project

Seja Um Bom Aluno (in portuguese) SUBA was a system for teaching and experimenting with mechanical and electronic systems, based on a remote-controlled car model, Subaru Impreza WRX from Fuji Heavy Industries. The model Subaru Impreza WRX was produced by Matoys and in its original form costs about 125 €.

The original model was transformed in order to be controlled by several electronic systems that were developed by students. The SUBA has an aluminium chassis underneath the body. The engine control system was modified in order to be controlled by a signal bus. The upper chassis has the capacity to mount two breadboards, four in-line buses, and a ARM microprocessor board, as can be seen in Figure 2.1.

![Figure 2.1: Vehicle used in the SUBA project](image)

The SUBA was equipped with several sensors and actuators, namely a three-dimensional acceleration sensor, a differential optical sensor and speed sensors. Based on the sensory systems, programs were developed, which allowed mechanisms of speed control, track following and anti-skidding.

The experiments were carried out by first year students of the Bachelor Degree in Electronic Engineering in the disciplines of Physics, Computer Architecture and Algorithms and Data Structures.
2.1.1 Sensors

The SUBA has a three-dimensional accelerometer module that communicates with the "black box" through a proprietary serial port. The three-dimensional accelerometer is based on two orthogonally placed two-dimensional and micromachined accelerometers in an electronic module with a programmable microcontroller in the circuit. The accelerometers are controlled, through the serial port, by the "black box". The commands between the parts are used for the calibration of the three accelerometer axes and control of the communications in the proprietary serial port.

The steering sensor of the SUBA is based on a differential sensor with two phototransistors that receive the reflection of infrared light emitted by two Light Emitting Diode (LED)s. The differential sensor consists of two optical couplers, based on the reflection in an object of the light emitted by the LED, to detect the presence of these objects in the phototransistor. There is another optical coupler placed 28 mm from the center of the steering sensor, and it is used to detect streaks or transverse tread marks which contain a car location code, like a bar code. The steering sensor and the barcode sensor are mounted on a plate and an aluminum track placed in front of the shaft, as can be seen in Figure 2.2. This system was intended to follow a light reflecting strip existing on a floor with low reflection, thereby traversing a complete circuit.

Figure 2.2: Differential sensor system of SUBA project [2].

Later, it was implemented a system of speed sensors for rotating the rear and front wheels. This system consists in fixing a white line to the wheel axle of the car and with an optical sensor to know how many times the white line passed through the sensor in a certain measure of time and thus the speed of the wheels of the SUBA is calculated.

For the lighting system, the SUBA had two sets of LEDs, front and back, which could be associated to give more intensity to the light emitted. These LEDs are used in a common anode configuration. The light system is a serie of LEDs, it should be noted that the headlamps were fed directly from the battery, while the logic circuitry of the interface board was fed from the voltage which was obtained from a voltage stabilizing circuit. There was also an emergency headlamp, with a blue LED, placed on the roof of the vehicle.
2.1.2 Motors

The SUBA has two direct current electric motors: the traction motor and the direction motor. The traction is made by the wheels of the rear axle and the direction is adjusted in the front wheels. The traction engine would make the car move forward or backward, while the steering engine drives the front wheels to the left or right. Both engines have gearboxes made with sprockets. The steering mechanism is based on an articulated parallelogram that the steering motor may deform to the left or right as shown in Figure 2.3.

![Figure 2.3: Steering system of SUBA project](image)

The SUBA has rear-wheel drive with a mechanical differential and a gearbox with two possible reductions, obtained by moving the group of sprockets together with the motor shaft, to the left or to the right, by means of a lever, shown in Figure 2.4. The differential is a device that connects the two traction semi-axles and allows the wheels to have different rotational speeds and, in a turn, the outside wheel rotates faster than the wheel inside the curve.

![Figure 2.4: Sprockets system of SUBA project’s motor](image)
In normal cars, the differential introduces a serious problem: if a traction wheel skids, for example due to the weak friction of the wheel with the ground, all the mechanical energy goes to that wheel and the other one stands still. There is therefore no pulling force in the car. To overcome this effect in sports cars and all-terrain cars, the differential has a system to limit the slip of the differential or even a self-locking system that only lets it slide up to a certain level and from there applies the energy to two semi-axles. In the original project was also implemented an anti-slip differential devices, that if too much power was applied to the traction wheels they would skid and the car would stop.

2.1.3 Consumption

Consumption is a fundamental feature in all vehicles that operate on a battery. It is important that the consumption is the smallest possible so that the time of use is maximized. Throughout the years of operation, new functionalities and improvements in disciplines and projects have been adapted from the original project of the SUBA. However it is important to retain some of the consumption that the original car registered, and these are presented in Table 2.1.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Traction motor running</td>
<td>10</td>
<td>0.46</td>
<td>4.6</td>
</tr>
<tr>
<td>Traction motor blocked</td>
<td>9.77 (left) 9.53 (right)</td>
<td>0.63 (left) 0.63 (right)</td>
<td>6.15 (left) 6.00 (right)</td>
</tr>
<tr>
<td>Both engines blocked</td>
<td>8.02</td>
<td>3.3</td>
<td>26.47</td>
</tr>
<tr>
<td>Steering motor blocked</td>
<td>9.37 (left) 9.42 (right)</td>
<td>0.92 (left) 0.92 (right)</td>
<td>8.62 (left) 8.67 (right)</td>
</tr>
</tbody>
</table>

2.2 A Sensor Network for the Internet of Things

A sensor network for the Internet of Things is a dissertation submitted to the Department of Communication Systems at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering with emphasis on Telecommunication Systems written by Linus Hobring and Philip Soderberg in July 2014. This project was developed through Department of Communication Systems of Blekinge Institute of Technology, Sweden.

2.2.1 Architecture

The system developed by the authors consists of hardware and software. The software implementation is built on several Python and C++ files, which are used as control mechanisms for different parts of the system. The hardware is a single board and a set of sensors, shown in Figure 2.5. Sensors used are described below:

- Temperature/Pressure sensor: Is used a T5403 digital barometric sensor. This sensor uses the Inter-Integrated Circuit bus, which facilitates programming and control using the Python code. Although this system sees the T5403 sensor as two different sensors: pressure and temperature, it is just one physical sensor.
• Humidity sensor: Is used a DHT22 digital humidity sensor that uses a GPIO pin to communicate with the single-board computer. The best readings were obtained using the Raspberry Pi General Purpose Input/Output (PIGPIO) library along with a Python script.

• Light Sensor: Is used an analog light sensor, consisting of a photocell, to measure ambient light. Is also used a digital to analog converter and this converter is consisting of a capacitor and a resistor combined with the light sensor. The photocell is a Light Dependent Resistor (LDR), which makes the time variable directly dependent on ambient light. The Python code is written to measure the time taken for the capacitor load.

• Distance sensor: Is used an HC-SR04 ultrasonic distance sensor to measure the distance between the sensor and an obstacle. It uses a GPIO pin to communicate with the single board. Python code is used to determine the time between sending and receiving reflected ultrasound.

• Radio Frequency (RF) Transmitter/Receiver: are used a 433 MHz transmitter and receiver in different cases to control remote switches. They are connected to a GPIO each and the code is written in C++, instead of Python, because of the logic of hardware and RF signals.

The main Python script is the initiator of the whole system. Here, the logic to start multithreaded programs runs as separate subprocesses on the system. Each sensor is running on its own segment. This allows you to perform several operations at the same time, even if the sensors require different time to measure and process. The software architecture is shown in Figure 2.6. The sensor measurement procedures are using two different files:

• Sensor Base Thread: This file contains all the logic that happens with a specific sensor. It handles data exchange between modules in the system and performs logic for all locally located use cases.
• Sensor Hardware Code: These files contain the code needed to communicate with the hardware. They are Python scripts or, in some cases, a compiled C++ program. It depends on the type of sensor and how the coding had to be done. The hardware code modules communicate directly with their thread code, and no logic, other than sensor detection, is placed in these modules.

2.3 Software Architecture for Autonomous Vehicles

Software Architecture for Autonomous Vehicles is a dissertation to obtain the Master’s Degree in Electrical and Computer Engineering by Instituto Superior Técnico written by André Batista de Oliveira in October 2009 [4]. In this document the author explains a complete software architecture to implement in vehicles from the high-level development of a multi-vehicle graphics console to the integration of the software necessary to create the vehicles’ operating system, as well as the configuration and compilation of the system kernel. This software architecture is implemented in the autonomous catamarans Delfim and DelfimX, developed in the Dynamical Systems and Ocean Robotics laboratory of the Institute of Systems and Robotics at Instituto Superior Técnico.

2.3.1 Architecture

Developing an autonomous vehicle software is not just writing code that implements a particular algorithm that causes the vehicle to move. It has to be built, integrated, written, simulated and tested all the software components needed to make the vehicle a viable platform, taking into account that different standalone vehicles usually have different goals and design. The brain of an autonomous vehicle is its software. Figure 2.7 describes the central role of the software in a standalone vehicle and its supporting hardware components required for interfacing with the environment. This diagram is valid for almost all autonomous vehicles. Hardware sensors are required to perceive the environment.
(these are software inputs) and the hardware actuators are required to run in the environment (these are software outputs). Optionally, but very common and sometimes even fundamental, is the hardware to store acquired data aboard vehicle activities.

![Figure 2.7: The basics of an autonomous vehicle software [4].](image)

Each sub-block would eventually be implemented as a separate thread and they would interrelate through a shared memory region protected by a synchronization primitive, as shown in Figure 2.8. This is the most widespread approach found in practice.

For a system to run faster, it generally needs to be implemented as a single program running as a single process. For a system to be the safest, it usually needs to be implemented as separate programs, running as separate processes. Threads, on the other hand, are neither the fastest nor the safest. There is a common misconception that splitting a program into multiple segments will somehow make it run faster. But the reality is that without multiple processors, running multiple threads will only slow down the system due to scheduling overhead, context switches, and synchronization between threads that the operating system must perform to safely share the processor between multiples topics.

The most frequent cause of software errors is by far the incorrect handling of concurrency. When using threads, is suppose to deal with it properly, or there will be race conditions and deadlocks. If threads are used in the critical software of an autonomous vehicle, it is also necessary to code the different thread priorities, or else the software will behave indefinitely.

### 2.3.2 Console

The user interface for an autonomous vehicle is commonly called a console and its importance in a system is paramount. It provides the user with the ability to easily reset actions, remotely monitor the vehicle during actual operation, interact with sensors and integrated actuators, or change parameters.

The author also mentions that simplicity is a criteria to be always taken into account, nevertheless identifies the main requirements for a good interface. The requirements ordered by their importance are as follows:

- The console should be lightweight, to avoid transport problems, quick to implement and easy to start;
- Monitor and control multiple vehicles simultaneously;
Figure 2.8: Typical architecture: multiple threads and shared memory [4].

- Support for various communication technologies (ethernet, radio, WiFi, etc.);
- A main screen to show the minimum indispensable telemetry data of the vehicles;
- Optimize real estate display (without title bars, menu bars, toolbars, superfluous vehicle telemetry data);
- Operating modes;
- Platform independent, running on as many operating systems and devices as possible;
- Low cost of implementation and use.
3 Telemetry System

Contents

3.1 Conversion and communication ............................................. 18
3.2 Luminosity ............................................................................. 22
3.3 Temperature ............................................................................ 23
3.4 Hall Effect ................................................................................ 24
3.5 Light Reflected ......................................................................... 26
3.6 Printed Circuit Board ................................................................. 28
Telemetry System

In the process of measurement of physical quantities, several sensors are used for this purpose. There are different types of sensors that produce continuous analog signals, these are classified as analog sensors. The continuous output signal that is produced by the sensor is proportional to the measured value \[16\]. On the other hand, a digital sensor is an electronic, or sometimes electrochemical, device where data is digitally converted and transmitted. This type of sensors are often used for measurements \[17\].

In analog signal processing, it is easy to calibrate using mathematical formulas and calculus. Since this project has a very strong mobility component and can act in several and different environments, the characteristic of low environmental dependencies is important \[18\]. There are some additional factors that have been considered in the decision as the importance that the sensors have a long lifetime usage, the easiness to correct the defective components and budget issues, since the final solution must have the lowest cost possible \[19\]. Some disadvantages of use, also identified as analog systems, are less noise-free since noise is added by the signal path \[20\].

In this project, will be used for the referred measurements three analog sensors, regarding to luminosity, Hall effect and light reflected, and two digital sensors, to acquire temperature and slope.

3.1 Conversion and communication

3.1.1 Analog-to-digital conversion

Taking into account that the Raspberry Pi only has digital ports, it will be necessary to use an Analog-to-Digital Converter (ADC), MCP3008 from Microchip \[5\].

An ADC transforms a time-continuous analog signal into a time-discrete sampled signal, quantized within a finite number of integer values. In this case, the ADC has 10 bits, thereby the input signal is transformed into samples with values between 0 and 1023 \[21\].

One of the advantages of using an eight-channel ADC is that the sensors are all read using the same Raspberry Pi port, thus limiting the use of GPIO terminals. Software scans all ADC channels and when one of these eight channels has some data to send, communicate keeping the other channels mute. In Figure 3.1 the circuit used for the acquisition and conversion of the data collected from sensors is presented.

Another specificity of this circuit is that the reference voltage for the conversion must be constant and adequate, for this a Microchip MCP1541 \[22\] was used. The converter power supply voltage is set by the Raspberry at 5 V. The reference voltage could not be derived from this port, since its value can slightly oscillate depending on the use of other components also connected. For it is necessary to realize an assembly, as illustrated in Figure 3.2 to guarantee a precise and constant reference voltage of 4.096 V.

The analog-to-digital converter has a voltage range from 0 V to reference voltage, 4.096 V. However not all sensors used have the same operating range, which means that the sensor output
signal should be amplified. The communication is made through a SPI interface.

3.1.2 Serial Peripheral Interface

The Serial Peripheral Interface (SPI) is primarily used in microcontrollers and their immediate peripheral devices. SPI bus consists of three transmission lines. Each of these three lines contains the information between the different devices connected to the bus. This operation is full-duplex type because data can be transmitted and received simultaneously. Devices connected to the bus are defined as master and slaves. It is the master that initiates the transfer of information to the bus and generates clock and control signals. Slave is a device controlled by the master, each slave is controlled by a bus through a line selector called Chip Select (CS) or Select Slave (SS), so the slave is activated only when this line is selected.

The SPI bus specifies three logic signals:

- Serial Clock (SCLK): it is generated by the master synchronizing the data transfer.
- Master Out Slave In (MOSI): Transports data from master to slave.
• **Master In Slave Out (MISO):** Transports data from slave to master.

Each slave is selected by a low logic level ('0') via the CS line. Data on this bus can be transmitted at a rate of almost zero bits per second up to 1 Mbits/s. In Figure 3.3, we can see that, to initiate a communication, the master sets the clock using a frequency less than or equal to the maximum frequency the slave it supports [23].

![Figure 3.3: SPI data communication](image)

In this case, the advantage of SPI compared to other serial communication protocols, is the fact that the data need to be transferred at high speed. This project has one master, the Raspberry Pi, for that reason this protocol works without problems because it only allows one master and one or more slaves. There are other disadvantages, such as the fact that SPI does not have acknowledgment of receipt, the master may be communicating wrongly the data and not knowing that fact. Another functionality that SPI does not have is addressing, so it needs an CS for each slave [24].

### 3.1.3 Inter-Integrated Circuit

This Inter-Integrated Circuit (I2C) bus was developed by Philips and allows the use of a large number of standardized components, which can perform various functions, as well as enable the efficient exchange of information between them. This new bus was widely accepted in the market, becoming one of the most used and this was decisive for the creation of version 2.0, launched in 1998. The concept of the I2C bus is to facilitate integration of circuits in an application such as sensors and converters, with a control system, so that they can work with their signals in a direct manner. A interesting feature of this bus is the possibility to use, in a same system, components of different technologies without there being incompatibility and nor conflicts in the communication [25].

In I2C, the transmission of data between the devices, visible in Figure 3.4, is done through two wires named Serial Data and Serial Clock. The devices connected have a fixed address (each component has a specific address), and is possible to configure them to receive or transmit data. In this way they are classified as masters or slaves. The I2C bus is multi-master type, as shown in Figure 3.4, this means more than one device can be master. However, during a communication, only one of the masters may be active, or a data collision will occur on the bus. For example: while a microcontroller sends signals to a converter, another microcontroller exchanges information with a
memory using the same bus. All this is possible, thanks to an "arbitration" that determines which of
the signals has priority in sending the data. Serial Data and Serial Clock lines are bi-directional and
must be connected to the positive power supply via a current source or a pull-up resistor to ensure
both lines remain high when the bus is free. One of the advantages of the I2C standard is that it
does not set the transmission speed (frequency) because it will be determined by the Master (Serial
Clock transmission) circuit [26].

![Figure 3.4: I2C data communication [7].](image)

The communication between master devices and the slaves, connected through I2C bus, is
initiated by the start condition and terminated by the stop condition. I2C operation, the start and stop
procedures, shown in Figure 3.5. A transition from high to low level in the Serial Data line, while the
Serial Clock line is at the high level, is indicative of the START situation. Since a transition from the
low level to the high level of the Serial Data line while the Serial Clock line remains at the high level,
it defines a stop condition. The master is always responsible for generating these conditions. After a
start condition the bus is considered busy, and only returns free some time after the stop condition. If
start conditions are generated then the bus will remain busy, but with appropriate circuits the function
of detecting the start and stop conditions in the devices that must be connected to the bus can be
implemented [27].

![Figure 3.5: I2C start and stop conditions [7].](image)

Information placed on the Serial Data line must be 1-byte length. The number of bytes that can be
transferred in each transfer operation is limited, in addition the bytes are transferred by first sending
the most significant bit. If the device that is receiving the signal, for some reason, can not work the
received data, it can change the Serial Clock line, putting it at the low level, and thus forcing the master
into a standby state. That way, encourage when the line is again free the data transfer can continue.
After writing/reading a byte on the bus, the receiving device generates an acknowledge bit. The
acknowledgment signal allows the data transfer to flow. Thus, for example, after the Start condition and the addressing are completed, the selected (Slave) stage must provide the acknowledgment signal [16].

The basic format of an I2C command consists of 7 address bits, used to specify the slave device to be accessed, followed by a read/write indicator bit. Usually the basic address of an I2C command consists of two parts: the first, 4-bit specifies the type of slave device to be accessed. The second, 3-bit, specifies one of up to eight devices of that type, which will be accessed. The Read/Write bit indicates whether the operation is read (level 1) or write (level 0) [28].

![I2C command when master writes to slave](image)

### Figure 3.6: I2C command when master writes to slave [7].

#### 3.2 Luminosity

To measure the light of the environment a, Light Dependent Resistor (LDR) or photocell is used. It is a resistance that varies in function of the light on its surface. The greater the intensity of the light that reaches the surface, the less its resistance will be.

This component has a dark resistance of 1 MΩ which corresponds to the resistance without any light. For a luminosity of 10 lux we obtain a resistance of 9 kΩ and for 1000 lux of 400 Ω [29]. In Figure 3.7 is shown the circuit used. This circuit has a typical response time in the order of the tenths of a second [30].

![Light dependent resistor circuit](image)

**Figure 3.7: Light dependent resistor circuit.**
When the LDR is not exposed to light radiation, the electrons are firmly united in the atoms that make it up but when light radiation occurs on it, this energy releases electrons and the material becomes more conductive, thus decreasing its resistance. LDR resistors only reduce their resistance with light radiation located within a certain wavelength range. Those made with cadmium sulphide are sensitive to all visible light radiation, those made with lead sulphide are only sensitive to infrared radiation [31].

The luminosity measured by the LDR will be used to know if it is necessary or not to have the LEDs on to illuminate the environment where the vehicle is at time. The results will be processed by a mathematical relation [29] where the relationship between the resistance R, a light sensor manufacturer constant (500) and light intensity for a typical LDR is

\[ R_L = \frac{\text{LightConstant}}{Lux}. \]  

(3.1)

As the LDR is connected to 5 V through a 3.3 kΩ resistor, the output voltage is

\[ V = 5 \times \frac{R}{R + 3300}. \]  

(3.2)

Substituting R from (3.1) into (3.2), obtain the light intensity,

\[ Lux = \frac{2500}{3300} - \frac{500}{3300}. \]  

(3.3)

The luminosity value in International System of Units [SI] units, lux, is calculated and will be displayed on user interfaces.

### 3.3 Temperature

To measure the ambient temperature outside the car, a temperature sensor, DS18B20 [32], will be used, as this sensor can measure a very wide temperature range from -55 °C to 125 °C and can be used in very severe conditions. It has a good precision and communicate through 1-Wire interface, this means requires only one port pin for communication.

Its operating principle is different from a thermistor because its resistance varies with temperature due to the chemical composition of the material from which it is made. It is also different from a thermocouple as this produces a voltage between two different conductors, depending on their temperatures. In the case of the thermistor or thermocouple, it would be necessary to use a circuit to obtain a variable voltage with temperature. As this sensor has no moving parts, never wears out, does not need calibration, works in many environmental conditions, is very cheap and is consistent in reading [33].

DS18x20 family members are accurate digital temperature sensors. They use Maxim’s 1-Wire bus protocol, which requires only one wire for receiving and transmitting data, a ground line is also required. Even so, this circuit needs a 4.7 kΩ pull-up resistor, as shown in Figure 3.8. Moreover, DS18B20 support called parasite power mode, when they drain energy from data bus when it’s high, to charge sensor’s integrated capacitors that will be used as a power source. To use it, a master
device initiates a reset pulse, and receives presence pulse from slave devices. Then it searches devices each produced device has a unique 64-bit ROM code, like serial number. Tiny first part of it defines a family of device, in this case, 0x28 for DS18B2 and all remaining bits are a unique item number. Finally, master device selects the necessary slave device with a Match ROM command, otherwise, transmits broadcast commands.

Figure 3.8: Temperature measurement circuit.

To acquire the temperature value, the system need to send a convert command. When the sensor receives it, will initiate data conversion process, to produce two bytes with the measurement. This is relatively slow process, and it can take up to 750 milliseconds. All that measurements are stored in a called scratchpad, a piece of sensor’s RAM. Then the temperature is calculated according to

\[
\text{Temperature} = ((\text{HighByte} \ll 8) + \text{LowByte}) \ast 0.0625. \tag{3.4}
\]

A multiplier of 0.0625 is a conversion coefficient between sensor’s internal values and real temperature, according to 12-bit resolution, on sensor’s scale stands for 0.0625 °C. This calculation is performed by the temperature acquisition algorithm and presents the final value in degrees Celsius [32].

3.4 Hall Effect

Hall effect sensors are widely used to measure the speed of rotation or detect the position of a given element as its output voltage varies when exposed to a magnetic field. Their main advantage is
that they can offer reliable data at any speed of rotation. Its disadvantage is the greater complexity in programming the system and the price in relation to an inductive sensor [34].

The principle of operation of a Hall effect sensor is through linear transducers, as a result, these sensors require a linear circuit for the processing of the sensor output signal. The operation of the Hall effect sensor is simple, as a beam of charged particles cross through a magnetic field, the forces act on the particles and the beam is deflected. When a conductor is placed in a magnetic field perpendicular to the direction of the electrons, which will be deflected from a linear path, the side of the conductor will become negatively charged and the opposite side will become positively charged. The voltage between them is called the Hall voltage [35].

A Hall effect sensor can operate as an electronic switch up to a frequency of 100 kHz. It does not suffer wear from contacts, or any mechanical fatigue because there are no contacts or friction. Another advantage is that they are not affected by environmental contaminants since the sensor is in a sealed package and without contact with the air, so it can be used under severe conditions [36].

Basically, there are two types of Hall effect sensors. One is linear, which means the voltage output linearly depends on the density of the magnetic flux. The other is called threshold which means that there will be a sharp decrease of the output voltage with the magnetic flux density, meaning that once it is triggered it latches and will not unlatch until a magnetic force of reverse polarity and strength is sensed. So if the north pole of a magnet turned it on, the south pole of a magnet is then needed to turn it off. For this project a linear operating sensor, A1324LUA-T, will be used, Figure 3.9 presents the circuit used.

![Figure 3.9: Hall effect measurement circuit.](image)

This system, uses the effect of Hall measurement through four magnets fixed, one within each wheel of car. Whenever a wheel rotates around the axis where the sensor is installed, it is possible to measure a voltage, thereby triggering a control flag in the software system. This flag is used to know the number of complete wheels rotations and determine the distance traveled by the car. This system, a sensor and a magnet as shown in figure 3.10 does not affect the rotation of the wheel, nor does it cause skidding.
3.4.1 Speed Calculation

Through software it is possible to know how many milliseconds have passed between each cross of the magnet through the sensor. Knowing the time variable and using the speed formula,

\[ \text{Speed} = \frac{\text{Distance}}{\text{Time}}. \]  

(3.5)

it is possible to determine the speed relating time and distance. A fixed parameter, is used in (3.5), because the perimeter of the wheel 21.5 cm, a fixed value. This approach is a very effective and accurate method to determine the speed. The disadvantage of this method is the need of the wheel complete a complete rotation in order to calculate the speed.

3.4.2 Acceleration Calculation

Knowing the speed and the time with each rotation of the wheel, it is possible to know the variation of the speed with each cycle, thus using

\[ \text{Acceleration} = \frac{\text{FinalSpeed} - \text{InitialSpeed}}{\text{Time}}. \]  

(3.6)

the acceleration of the car, either positive or negative, is determined.

3.5 Light Reflected

The light reflected sensor, QRE1113 [37], consists of two parts, an infrared emitting LED and an infrared sensitive phototransistor. As more infrared light is detected by the phototransistor, the
The output voltage of the sensor lowers. Two resistors, 200 Ω and 5.6 kΩ are used to operate, the circuit is presented in Figure 3.11. The 5.6 kΩ pull-up resistor helps turn the transistor's light-variable current into a light-variable voltage, and can be used to make it more or less sensitive, because the microcontroller detects only when the phototransistor pulls the Raspberry Pi terminal low.

These sensors are widely used to follow lines because the white surfaces reflect much more light than black, so when directed to a white surface, the voltage output will be less than on a black surface.

![Figure 3.11: Light reflected measurement circuit.](image)

Because the sensor component simply measures the intensity of the infrared and is unaware of the light source, it is susceptible to false positives caused by outside sources that put out infrared such as camera flashes, lamps, and even the sun. To fix these possible errors, the sensors were placed as close to the ground as possible, thus avoiding false measurements by external light interference.

One of the characteristics of this sensor, is its speed in this detection. It is not intended to be used to determine the exact distance, but to check the proximity of objects less than 3 centimeters away. This sensor can also be used to detect white surfaces against blacks because a white surface reflects more light than a black surface resulting in a higher reading. Because of this, an array of these can be used for detection of the line to follow [38].

This array consists of six sensors arc-shaped in the front of the car, as shown in Figure 3.12. This distribution allows the system to detect sharp curves and thus avoid trips off the track. The fact it is arched allows detecting a straight line with both sensors centered, but also curves with intermediate degree of turn with the sensors placed in the middle of the array. In case of high angle turns, the sensors further away allow the car to react faster to the exit of lane avoiding to have to drive backwards. This mechanism has been programmed for the autonomous mode and works with an algorithm that also regulates the speed according to the type of curve allowing to maintain the control but also to have the car always in the middle of the track, making the line be as possible under the two centered sensors.

In this system, the array sensors are acquiring values that are compared to a reference value through the use of comparators. This operation is used because in this context the important thing is to know if the line is under the sensor and not the intensity of the reflected light. The values are compared with a user-set voltage value for the track in test, bearing in mind that the lanes may have various background colors, line colors or material types, more or less brightness. This calibration should be
done at the beginning by using a potentiometer that is placed on the board with the remaining circuit. In the development of this system, the work environment was mostly used a white lane with the black line. However, it was tested in two other environments: brown lane and black line and black lane and white line, where the best results were obtained was in the first scenario, white lane with the black line.

### 3.6 Printed Circuit Board

In this work, was designed a system as centralized as possible, however, it is not always possible to build the entire system on a single board. Thus, this project has two small PCBs and a shield that is assembled over the GPIO ports of the Raspberry Pi making the system easier to connect and implement.

One of the small boards is positioned on the roof of the vehicle and it purpose is only to operate the lighting system, not having any processing, this board only turns the lights on and off and acquires the ambient light through the LDR. The other small board, located in the rear of the car, above the traction section, is attached to the Hall effect sensors on the rear wheels and is intended to prevent very long wires crossing the car to connect Raspberry Pi.

The shield, which is to be assembled in Raspberry Pi, contains the rest of the system circuit. The board is designed to be as compact and small as possible. Its position has also been taken into account in order to have the weight equally distributed by the chassis of the car. All of the circuits explained in this chapter are integrated in this circuit board. In Figure 3.13 can see the application in the system.
Figure 3.13: PCB shield assembled in Raspberry Pi.
4.1 Architecture ................................. 32
4.2 Drive motor ..................................... 33
4.3 Servo ........................................... 36
4.4 Consumption ................................. 39
4.5 Remote control ............................... 40
4.6 3D printed components .................. 41
Car system

In this chapter, the system that was implemented for this project will be explained in detail. The sensor system, explained in the previous chapter, is another part of the system that sends the data to Raspberry Pi for data processing and decision making. The system has two modes of operation, one being user-controlled and the other self-controlled. Furthermore, the traction system and the steering system are implemented in such a way that the car performs as expected with less consumption as possible. The assembling of whole system is done through parts printed in 3D.

4.1 Architecture

This project has two operating modes that can be chosen by the user, and at any time the mode can be changed. This allows the data coming from the sensors to be used for analysis by the user and also for control of the actions to be taken by the car in certain locations. Modes operate using the same vehicle features only differing in decision making because with this approach the mode only affect the movement of the car and not the telemetry system. This make the system more stable and save resources. The overall scheme of the project is shown in Figure 4.1.

![Figure 4.1: Operational modes diagram.](image)
In autonomous mode of operation, the car must run through a circuit following a line in the center of the road. In this mode, the aim is for the car to run centered and at constant speed along the entire course. In parallel, it must send the data of the sensors to the user interface and save the acquired data for future analysis. This mode works as a demonstration of the telemetry system and could be used to test the capabilities of the acquisition system because all sensors are acquiring data periodically. The system has also a button to force acquisition of the sensors, this mechanism allows the user get data from the sensors without waiting the periodical acquisition.

On the other hand, there is also a user controlled mode. In this mode, the trajectory of the car is controlled by the user through a directional control, but the control is done through a mobile Android application, where in addition to the direction of the vehicle, the user can also select the desired speed. The simple control has joystick that control the traction motor by making it move in all directions to control the direction of movement of the steering shaft.

The acquired data, after processing, is sent to the mobile device running an Android application, but also to a virtual machine, running Ubuntu, through WiFi connection, sending the message in a JavaScript Object Notation (JSON) format. In this virtual machine, a responsive web page is created using the Django framework, to show the data and present it in a table. This tables is uptaded each time a new socket is received and the content in added to a new raw in the table. In parallel, write a readable version of the table in a text file saved in virtual machine desktop. This architecture is shown in Figure 4.2.

![System communication architecture](image)

**Figure 4.2:** System communication architecture.

### 4.2 Drive motor

To control the drive motor of the car, a H-bridge, L293D [40] from Texas Instruments is used. This component contains two H-bridges, however only one will be used as the motor driver. This component withstands constant current of 600 mA and peak current around 1.2 A, a voltage of 5 V is used to power the motor. The circuit used to control the drive motor is presented in Figure 4.3.

To control the voltage delivered to the motor, and in turn its power, a Pulse-Width Modulation (PWM) signal is used. Considering a square wave, for the correct operation of the PWM signal must
Figure 4.3: Circuit used with motor and H-bridge.

The capacitor stores a bit of energy and releases the energy. There are three main reasons to use capacitors in parallel with the motor. First of all, to smooth out current spikes caused when a motor starts up or gets stuck, preventing spikes/dips from affecting other components in the circuit. The second reason is to smooth out current spikes caused when controlling a motor speed using digitally-controlled PWM signal. In PWM, the power to the motor is pulsed on and off by a controller rather than varied smoothly using a variable resistor. If a PWM signal is on half of the time, the motor receives half of the power, and the motor behaves accordingly.

The duty-cycle, defined in percentage, controls the motor speed and corresponds to the time the signal remains in the 5V and in this case varies the speed at which the motor is rotating. The minimum value for the car moves corresponds to a duty cycle of 70% and the maximum speed of the car, is around 4.5 m/s, that corresponds to 100% duty cycle.

With the H-bridge driver approach, the motor control is implemented through PWM, otherwise only have the state of rotate the motor to both sides, it would not be possible to have the motor state stopped. In this way, two PWM signals are inserted into the H-bridge that allow the motor to be controlled, it is possible to keep the motor stopped, or to rotate the motor in the desired direction, controlled by the logic level, as shown is Figure 4.5. As previously explained, is also possible, by changing the duty cycle of this signal, to control the speed of rotation of the traction motor.
Two PWM signals are used, with the need to control the direction of the motor rotation, each signal is connected to a terminal to control the polarity of the motor. Taking into account that two PWM signals are used, each connected to one of the motor terminals, this will allow the car to move forward, backward and stop. When the signal sent is that corresponding to the positive terminal, the signal on the negative side is maintained with duty cycle of 0%, and then the car moves forward. The reverse happens when it is necessary to drive backward. In the case of a standstill, both signals have a duty cycle of 0%, keeping the car stationary.

The system, is presented in Figure 4.6 has two modes of operation, and for each mode of operation, the traction system operates in the same way although with different speeds. In the case of the autonomous mode, the car must have a low and constant speed. In this case, a duty cycle of 75% is the standard value used, that is around 1.5 m/s, which allows the correct reading of the lines and gives the vehicle an adequate reaction time. However, the user can choose to vary the speed at which the motor is rotating. On the other hand, in the remote control mode, the user sets the speed at which the car operates and thus controls it at the desired speed.
4.3 Servo

The original model of the car, was equipped with the steering system with only two positions. This system consisted of a DC motor, shown in Figure 4.7, which rotated in both directions and thus turned the wheels through sprockets. One of the problems with this approach is that the user would not be able to turn the wheels slightly, they only rotated to the extreme positions. This led to problems such as damage to the material, since the user was continuously turning to one side, the motor would be forcing the entire system of sprockets and the direction, being plastic, would be rapidly damaged, and the friction between the materials was audible. The other problem of this system is that it would not be possible to implement a position control through PWM since in a Direct Current (DC) motor this application would only be possible to control the direction and the speed of rotation, as explained for the traction motor.
To overcome the described shortcoming, the DC motor was replaced by a servo motor with three poles of ferrite, Futaba S3003 [44], presented in Figure 4.8. This servo can rotate 270° and thanks to the aforementioned three ferrite poles has enough force to turn the entire steering system. Like the DC motor of the original model, it also operates at a voltage of 5 V. This voltage at a speed of approximately 0.21 meter per seconds for every 60° at no load, with a current drain of 7.6 mA. This servo model, weighs about 37 grams, lighter than the original engine, however it is more bulky. Its encapsulation is plastic, with plastic bearing and nylon gears.

Similar to the approach used in the traction motor, a PWM signal is also used in the steering servo. However, in this case the signal controls the position of the servo and not its speed. The PWM signal can be used to control servo direction. The angle is determined by the duration of a pulse that is applied. The position of the servo is not defined by the frequency of the signal, but only by the duration of the pulse [45]. The servo waits for a pulse every 20 ms, because unlike the traction motor of the car, a frequency of 50 Hz is used [46].
On the other hand, duty cycle can have values between 3.5% and 12.5%, with these corresponding to the two extremes of 0° and 180° respectively, thus causing the car to turn left or right. The central position, makes the vehicle moves in a straight line and has the value of duty cycle 8%, these situations are presented in Figure 4.9.

![Figure 4.9: Relation between servo position and duty cycle of PWM signal.](image)

The two modes of operation, requires to use two different software approaches for turning, because each mode has specific characteristics.

In manual driving mode, the user decides when to turn by pressing the button on a controller, and the car will turn only while the button is pressed, returning to the center position when the button is released. In this mode, the important thing is to turn as long as the user press the button, thus there will be only three positions, one center and two turning, left and right. In the software, the change is made to the value of duty cycle, using the values above mentioned, 3.5%, 7.5% and 12.5%, as reference positions. However, in the mobile application it will be possible for the user to regulate the steering servo as if using a steering wheel, in which case it is possible to vary the direction 180° of amplitude. While the mobile application wasn’t finished, this operation was tested with potentiometer where the rotation of the component made to vary its resistance that in turn made the direction of the car turn proportionally, making a relation between the voltage in the potentiometer and the duty cycle.

In the autonomous mode of operation, the car should follow a line that will be placed along a circuit and it will have straight and curved parts. In the case of straight parts, the operation is simple because the array of sensors will know which sensors are reading the line at that moment and will turn the car so that the line is always read by the sensors positioned in the center of the vehicle. In curves, and similar to the method used on the straight parts, the microcontroller will know in which sensor the line is in. However, and preventing the tight curves, the closer to the center the sensor is, smoother will be turning that curve. More abruptly turning will occur when the sensors that detect the line are those that are placed near the wheels. This mechanism prevents the car from being misled in case of tight corners, as well as to keep the vehicle always centered on the track. The software used for this implementation increases or decreases the duty cycle depending on the sensor reading the line. The extreme values, 3.5% and 12.5%, will only be used when it is necessary to make the turn in a tight curve, intermediate values will be used for the remaining sensors in the array.
Therefore, the servo is the main component of the steering system, shown in Figure 4.10, in conjunction with the axis originating from the original model and with the printed parts it is possible that this system became better than the one that equipped the original model. In this way the plastic materials of the steering system are not forced as they rotate only in a defined set of positions without overload.

![Steering system used in this project.](image)

**Figure 4.10**: Steering system used in this project.

### 4.4 Consumption

The Raspberry Pi connected but without running any type of software and without connecting any peripheral has a consumption of 220 mA with a voltage of 5 V. The GPIO ports can support 50 mA distributed by the various terminals, which in turn support currents of approximately 16 mA each one.

Taking into account only the Raspberry Pi and sensor system already mentioned, the project presented consumptions of 350 mA at 4.82 V. However, the traction and direction motors were deactivated, and this is just the base line consumption. Nevertheless, with the processor, sensors and motors all running, there was an increase in consumption for 770 mA at 4.64 V. It should be noted that for these measurements a monitor was still connected via HDMI and Raspberry Pi was powered through the mini USB port by a transformer connected to the outlet.

This project has an estimated consumption between 500 mA and 1200 mA depending on the elements in use. For both modes all the previously mentioned sensors and as well the traction motor and the steering servo are used. In the communication systems, also Bluetooth and Wi-Fi modules are used, increasing consumption. The measurements are presented in Table 4.1.

The battery that the original model had equipped would work in this system, as it was a lithium-ion battery with a voltage of 6.4 V and 2 A and a capacity of 500 mAh, but would work for just less than half an hour in full power usage.
Table 4.1: System consumption values.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Servo and motor running</td>
<td>4.92</td>
<td>0.92</td>
<td>4.53</td>
</tr>
<tr>
<td>Sensors acquiring</td>
<td>4.89</td>
<td>0.35</td>
<td>1.71</td>
</tr>
<tr>
<td>Sensors, Bluetooth and WiFi</td>
<td>4.64</td>
<td>0.77</td>
<td>3.57</td>
</tr>
<tr>
<td>Autonomous mode running</td>
<td>4.52</td>
<td>0.43</td>
<td>1.94</td>
</tr>
</tbody>
</table>

4.5 Remote control

One of the objectives of this project is to ensure that the car has a remote control where the data coming from the sensors can be analyzed independently the mode of operation used.

In the autonomous driving mode, the vehicle is detecting a marked line on the ground and sends acquired sensor data to the an Android application, where user can, at any time, change the mode of operation, as shown in Figure 4.11. Is also possible press a button to make a sensor acquisition allowing the user require a measurement anytime without wait for a new cycle of the periodic update in the user interface. The acquired values are updated in the app every second, but this value can be change in the code.

Additionally, the user has a remote control with similar functions to the one that came from the factory, being able to drive the car freely through a joystick that reflects all possible directions and also controls speed value, as shown in Figure 4.12. In this mode the sensor data are also processed and shown, making the car a mobile acquisition system. In this mode, is also possible control the light system and require a sensor acquisition.
The communication between the device and the vehicle is made via Bluetooth, as this protocol presents a fast communication and low power consumption, both characteristics are really important to this kind of systems, to ensure the real-time and battery life, respectively. Another factor to consider is the ease of getting communication with mobile devices regardless of the operating system in use.

4.6 3D printed components

3D printing is a group of add-on manufacturing technologies where a three-dimensional object is created by superimposing successive layers of material. During this work it was necessary to create auxiliary parts to attach several external components in the original base of the car. These pieces were designed to fit and considering the needs and the function they would have in the final system. Thus, Onshape, a online and free CAD software was used for modeling the parts as well as, BQ Prusa Hephestos i3, the 3D printer used for their printing.

The material used for the applications described above is PLA, a biodegradable thermoplastic derived from renewable sources such as corn starch, so it would be the most environmentally friendly option. It has a deterioration time around 24 months buried or 48 months in water, which is a much shorter time when compared to the hundreds of years of other plastics. It presents an opaque color, for aesthetic reasons, in this work the black version will be used. It is a very rigid and resistant thermoplastic, the most rigid compared to commercially available options (Acrylonitrile Butadiene

---

Figure 4.12: Remote control running Controlled Mode.
Styrene (ABS) and Poly-Ethylene Terephthalate Glycol (PETG), difficult to deform or flex. Its high extrusion flow and low shrinkage produces more accurate and more detail-based parts, sharper corners and better surface finish than other thermoplastics [51].

All parts have been designed and printed respecting the needs of the component itself. In the case of the new servo of the steering system, the model in Figure 4.13 was developed. The necessary rotation of the axis as well as the necessity of a secure and robust fixation was taken into account.

Figure 4.13: a) 3D model of the support for steering servo. b) Servo support assembled in steering system.

The Raspberry Pi support box, shown in Figure 4.14, had in its conception a notion of a probable board heating and the need to keep GPIO ports usable. When the entire system is designed, a cover must be projected to secure the wires coming out of the GPIO terminals.

Figure 4.14: a) 3D model of the Raspberry Pi support box. b) Raspberry Pi box assembled.
5

User Interface System

Contents

5.1 Android Application ................................................... 44
5.2 Web Page .................................................................. 50
User Interface System

In this chapter, the User Interface (UI) will be explained in detail. After the acquisition, telemetric data is sent by Bluetooth to an Android application where the user can view the acquired data through a visual interface. In parallel, the acquired values by the system are sent by TCP sockets to a Wi-Fi-connected virtual machine belonging to an ad hoc network. In this Ubuntu virtual machine, the data is received and later presented on a web page through the Django framework and stored in a readable text file for future analysis.

5.1 Android Application

In this project the car telemetry monitoring and tracking application has been developed for the Android operating system. The Android Operating System (OS) was chosen because it is multiplatform, it means, can be used on any mobile device such as a mobile phone, tablet or computer. The platform used to develop the application was MIT App Inventor, an open source online application originally created by Google, and currently maintained by the Massachusetts Institute of Technology (MIT). This tool uses a graphical interface that allows users to drag and drop visual objects to create an application that can run on all Android devices. The tool was released on 12th of July of 2010, and publicly released on 15th of December of 2010 by Hal Abelson and Mark Friedman [52].

This online development environment, shown in Figure 5.1, has many features that can be accessed. One of the main features is the access to most of the phone's functionality: phone calls, SMS texting, sensors for location, orientation, and acceleration, text-to-speech and speech recognition, sound, video, etc., and is also possible invoke other apps, with the ActivityStarter component. In terms of programming, the control is just as with a textual language. There are blocks for conditionals (if, if else, etc.), foreach, and while, and a fairly comprehensive list of math and logic blocks. Database access is a plus, on the device and on the web, because is possible to save data persistently, and with a proper web database share data amongst phones. The access to web information sources (APIs) is manageable, and is possible to bring data from Facebook, Amazon, etc. [53] [54].

However, there are still some limitations in creating the applications. The UI builder can’t build any user interface. For instance, it is not possible to create apps with multiple screens and handling orientation changes has some glitches. More, polymorphic components are not a available, this means, function blocks are tied to specific components, so there is no way to call functions on a generic component. For instance, to create a procedure MoveXY, it has to be tied to a specific image sprite, not a general image sprite. To share an app, is difficult, because the access to the Android Market is limited. The apps (.apk files) generated by App Inventor lack the required configuration for direct inclusion in the market [55].
This platform allows for programming through blocks, as shown in Figure 5.2, allowing intuitively build an Android application. However, there are limitations in this technology and some of the features of this application have become more difficult than if it was programmed in Java code, like all Bluetooth communication with Raspberry Pi, from pairing devices to sending and receiving messages. The connection to Raspberry Pi is done by pressing only on the Bluetooth symbol, this is because the addresses of both devices are known and have been previously paired. Now, the algorithm looks specifically for the address B8:27:EB:A0:09:DB, which corresponds to Raspberry Pi and connects in a few seconds. Bluetooth communication between the devices is done through a serial connection where Raspberry Pi functions as master and the app is slave.

This application aims to be the control center of the car. The user after downloading the installation file and installing the app on their Android smartphone, comes across a startup menu, Figure 5.3, after a few seconds to the main menu of the application.

In the main menu, Figure 5.4, it is possible to choose which mode of operation you want or leave the application. By pressing the symbol corresponding to the menu, the user is automatically forwarded to the menu of the chosen operating mode. You can return to the main menu at any time by pressing the smartphone return key.

In both menus, the user only have to press on the Bluetooth symbol to connect turn on the serial communication because devices have been previously paired.

In Controlled Mode menu, in Figure 5.5, the user has full control over the car and can use it through the joystick by varying the direction, speed and steering of driving according to the position. The joystick is based on a point read on an XY axis and with these coordinates changes the trajectory of the vehicle. It is also possible to control the lighting system by choosing between on, off or automatic, which means the light sensor is used to control the system. While using this mode, all sensors are acquiring periodically and their values are updated in the application interface every second by...
In Autonomous Mode the user interface looks the same except for the joystick as it deals with the way the car is driven based on sensor readings. The Bluetooth connection works the same as the controlled mode, and you can turn the communication on or off at any time. Regarding to the sensors, the operation is identical. Acquisitions are made periodically with the values printed on the screen every second.
In this mode the user has two different functionalities, as shown in Figure 5.6. One is the acquisition button, allowing the user to acquire sensor values at any time without having to wait for the stipulated time between acquisitions. The other feature is to allow the user to decide when to start or end the autonomous locomotion of the car. This last feature is intended to prevent the vehicle from starting to walk without being positioned in the correct place or allowing the user to stop the program if the car loses the contact with the line.
Figure 5.5: Controlled interface screen.
Figure 5.6: Autonomous interface screen.
5.2 Web Page

In this project, the access technology was chosen WiFi, defined by the IEEE 802.11 standard. WiFi is a wireless networking technology that allows the communication of devices through an airborne signal. Within the possibilities for common WiFi networks, there are the most varied applications, such as access to the Internet in public places, simply by connecting to an available access point. In order for nodes to communicate with each other without the need for an access point, they must be configured in ad-hoc mode, which is a type of local network built by the devices. The nodes themselves coordinate sending and receiving messages to each other. In this way, it is possible to keep the network functional even in areas without common access points, because the WiFi module integrated in the Raspberry Pi 3 and a WiFi dongle are used to connect a computer with a Linux operating system, where the data will be processed from the sensors [56] [57] [58]

The ad-hoc network created has the name "thesiscar", and nodes use static IPs to communicate only with each other, not with the outside, increasing network security. In the network "thesiscar", the address 192.168.25.1 was chosen for the computer, that is, the server where the TCP socket that is responsible for the communication is generated. The TCP client, in this case Raspberry Pi, has the address 192.168.25.2 and is where a JSON format file is generated with the values acquired by the telemetry sensors of the car, Figure 5.7.

However, if you need to add more nodes to the network, it is a simple and quick process, just add a new address to the server. Two nodes were used for this system. The TCP client, where data is acquired using a Raspberry Pi 3 microcontroller with the Jessie Raspbian operating system, and with temperature, brightness, velocity, acceleration and reflected light sensors connected through the device’s GPIO interface. For the server a laptop computer with the Ubuntu 17 operating system is used [59] [60] [61].

Django is a web development framework, written in Python, that uses the standard model-template-view, this allows you to create servers and receive TCP sockets with data. In this project, the TCP sockets arrive at the server with messages in the JSON format, after which the values corresponding to an acquisition of the sensors are removed. The values are stored in a temporary file where the HTML page fetches values to update the table with each new entry, Figure 5.8. The values are also written in a text file in a human-readable format and with a date and time for future user analysis.
Figure 5.8: Web page created with acquired data from sensors.
6

Conclusions
Conclusions

In this work the subject of a car telemetry system with the functionality of capturing data of several sensors was considered. The system collects the data and also processes the info and acts accordingly.

This system was planned, developed and implemented starting from a commercial remote controlled car. The various options of sensors and actuators were analyzed to allow everything to work as planned. From the analysis of several sensors, several were chosen to be tested, and those used were the ones that worked best for the desired application. Their role in the project was taken into account, but also their consumption, cost and implementation.

Some changes and adaptations were necessary in the structure of the car as well as the alteration of the steering system and respective servo. These changes increase the reliability of the car and have a better performance in the turning, also allowing a greater number of positions of the wheels. Two PWM signals are applied to the drive motor to control the motor, so the speed of rotation of the rear wheels could be defined. For direction, control an H-bridge is used where the two signals can switch and thus define the polarity of the drive motor.

The two modes of operation, are working accordingly to planned. The algorithms of this car are in Python, and through the use of interrupts and specific functions of Rasperry Pi it was possible to make the program faster to process and not waste time and battery on resources when they are not being used. The communication systems, using Bluetooth and WiFi are implemented and were tested in many different scenarios. An Android application was made to give to user a interface to analyse data and control the car, both goals were completed. The web site is responsive, this means that when some new socket arrives to the server, the web page updates adding a new line to the table where the acquired data are displayed.

The 3D drawings were designed according to the current features and specifications of the project and can be modified if this is beneficial.

6.1 Future work

The whole system could be based on real-time and also in this field it is possible to accelerate the system response by improving the code developed in Python. The creation of specific tasks and the implementation of priorities will cause the processor to slightly reduce its consumption. The question of consumption is another factor to take into consideration since the system is mobile and its power will be made from a battery in the car. More sensors could be added to the system, such as gyroscope, to measure the slope, humidity sensor or battery level sensor, thus increasing the capabilities in the execution of modes, but also increasing the information available to the user. All this could be integrated in a better printed circuit board, smaller and with better connections.

In terms of user interface, the Android application could be improved, to a better user experience, such as search for Bluetooth devices to allow adding more cars to the same app. Also regarding to
mobile app, is also possible improve menus creating more useful options to the use, like save the data to a text file saved in the smartphone. Although, this kind of these new features could be implemented always in Java programming environment. The web page, could have a more attractive layout and be added a ID column allowing multiple acquisition from more than one car.

The integration of all the hardware into a single board would allow the vehicle more flexibility. Since this project has a major mobility component, it will be necessary to develop an interface to communicate with the user in real time via a mobile application. The captured and processed data could be stored in the cloud organized by measurement and the user can access at any time to analyze the data. It is also possible redesign some parts of the car through a 3D designer and print new parts to assemble in the car, creating less weight and more aerodynamics to the system.
References


[40] L293D datasheet, Texas Instruments, 2016.


