Autonomous Driving with V2I Communication

José Luís Henriques dos Santos Pereira

Thesis to obtain the Master of Science Degree in
Telecommunications and Informatics Engineering

Supervisor: Prof. Alberto Manuel Ramos da Cunha

Examination Committee

Chairperson: Prof. Fernando Henrique Córte-Real Mira da Silva
Supervisor: Prof. Alberto Manuel Ramos da Cunha
Member of the Committee: Prof. Ricardo Jorge Fernandes Chaves

October 2018
Acknowledgments

I would like to thank my parents and my brother for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible. I would also like to thank my grandparents, aunts, uncles and cousins for their understanding and support throughout all these years.

A special thanks to my dissertation supervisor Prof. Alberto Cunho whom allowed me to develop this Thesis with his insight, support, guidance and knowledge. Many thanks to the Prof. Ricardo Chaves who gave me feedback on the project, corrected me, bringing me back on track, and shared some tips for this Thesis.

I would also like to thank the university Instituto Superior Técnico for this opportunity and to teach me many subjects which were decisive in the creation of this Thesis. I want to thank the RNL (Rede das "Novas" Licenciaturas / Network of "New" Bachelor's) from the Informatic Engineering Department for supplying the LEGO® Mindstorms hardware that allowed me to complete this Thesis as an alternative to the Arduino Hardware.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

To each and every one of you – Thank you.
Abstract

Nowadays, it is possible to see automatic vehicles by large technology companies like Google and also by automobile companies namely Mercedes, Renault, Audi among others. Although there are a few prototypes by these companies, there is something the prototypes lack, the ability to communicate with the infrastructure itself. Companies, like those mentioned before, use a camera in the vehicle to identify what is going on around them. The vehicles depend on that device to identify traffic lights, traffic signs, pedestrians, lanes, obstacles and other elements in the street that are noteworthy of it. However, in this project, the main goal is to create a model-size autonomous vehicle that allows for the emulation of real autonomous vehicle’s behavior, while using wireless technologies to communicate with controllers and detect traffic infrastructures. Considering the vehicle’s autonomy, the technologies used to perform these connections must be low energy consuming and fast transmission.

Keywords

Autonomous; Model; Vehicle; Car; Wireless; Bluetooth; Low Energy; Infrastructure; Traffic; Ultrasonic Sensor;
Resumo

Hoje em dia, é possível ver veículos autônomos por parte de grandes empresas de tecnologia como a Google e também por empresas automóveis nomeadamente Mercedes, Renault, Audi entre outras. E apesar de já haver alguns protótipos por parte destas empresas, esses mesmos protótipos têm algo em falta, a habilidade de comunicar com a infraestrutura. Empresas, como as mencionadas anteriormente, utilizam uma câmara no veículo para identificar o que se passa no seu redor. Os veículos dependem desse dispositivo para identificar sinais luminosos, sinais de trânsito, faixas, obstáculos, peões e outros elementos na estrada que são notáveis. Contudo, neste projecto, o objectivo principal é criar um veículo modelo autónomo que permite a emulação do comportamento de um veículo autônomo real, enquanto usa a tecnologias sem-fios para comunicar com controladores e detectar as infraestruturas de trânsito actuando respectivamente. Tendo em conta a autonomia do veículo, as tecnologias usadas para realizar essas ligações têm de ser de baixo consumo de energia e transferência rápida.

Palavras Chave

Autónomo; Modelo; Veículo; Carro; Sem-fios; Bluetooth; Baixa Energia; Infraestrutura; Trânsito; Sensor Ultra-sónico;
Contents

1 Introduction 1
  1.1 Introduction ......................................................... 3
  1.2 Briefing ............................................................... 4
  1.3 Objectives ............................................................. 4

2 Background 5
  2.1 Related Work .......................................................... 7
    2.1.1 Open source hardware and software platform for robotics and artificial intelligence applications .......................................................... 7
    2.1.2 Line Follower and Obstacle Avoidance Bot using Arduino ............ 8
    2.1.3 World’s First Android autonomous vehicle ............................ 9
    2.1.4 Autonomous Vehicle-to-Vehicle (V2V) Decision Making in a Roundabout 9
  2.2 Summary ................................................................. 10

3 Solution Architecture 11
  3.1 Road Design ........................................................... 13
    3.1.1 Curves .............................................................. 14
    3.1.2 Crosswalk ........................................................ 15
    3.1.3 Intersection ...................................................... 16
    3.1.4 Traffic Signs ...................................................... 16
    3.1.5 Straight Road .................................................... 17
  3.2 Communication ........................................................ 17
    3.2.1 Types of Communication Protocols .................................. 18
      3.2.1.A ANT: .......................................................... 18
      3.2.1.B Bluetooth Low Energy 4.0: .................................. 19
      3.2.1.C RFID: ......................................................... 20
      3.2.1.D Wi-Fi: ......................................................... 21
      3.2.1.E ZigBee: ........................................................ 22
    3.2.2 Communication between Vehicles ........................................ 23
4.1.4 Implementation Process .............................................. 43
  4.1.4.A Assembly .................................................. 44
4.1.5 Road Design ..................................................... 44
4.1.6 Car Software ..................................................... 46
4.1.7 Controller Software .............................................. 46
4.2 User Interface ....................................................... 47
  4.2.1 Controller Software - Computer .................................. 47
  4.2.2 Controller Software - Android .................................. 49
  4.2.3 Car Software .................................................. 49

5 Work Evaluation .......................................................... 53
  5.1 Hardware .......................................................... 55
    5.1.1 Phase 1: Assembly ........................................... 55
    5.1.2 Phase 2: Moving vehicle - Onward ................................ 55
    5.1.3 Phase 3: Moving vehicle - Turning ................................ 55
    5.1.4 Phase 4: Implementation of the Ultrasonic Sensor ............... 56
    5.1.5 Phase 5: Implementation of the Colour Sensor ................ 56
    5.1.6 Phase 6: Building the road .................................... 56
  5.2 Communication ..................................................... 57
    5.2.1 Phase 1: Implementation of Bluetooth .......................... 57
    5.2.2 Phase 2: Implementation of computer App ....................... 57
    5.2.3 Phase 3: Android communication ............................... 57

6 Conclusion ............................................................... 59
  6.1 Conclusions ....................................................... 61
  6.2 System Limitations and Future Work ............................... 61
    6.2.1 System Limitations ........................................... 61
    6.2.2 Future Work .................................................. 63
    6.2.3 Learning purposes ............................................ 63

A Laboratory Project ..................................................... 69
List of Figures

2.1 Mobile App Controller of gripping robot - image from: [8] ................................. 8
2.2 Following line robot with an obstacle - image from: [9] ................................. 9
2.3 The World’s First Android Car - image from: [10] ................................. 9
2.4 Two Autonomous vehicles in a roundabout - image from: [11] ................................. 10

3.1 The design of the road with measurements. ............................................. 14
3.2 Top curve of the road with the proper measures in meters. ....................... 15
3.3 Lower curve of the road with the proper measures in meters. .................... 15
3.4 The middle of the road: An intersection plus crosswalks and the straight roads .... 17
3.6 Assembled vehicle .............................................................................. 32
3.7 Model road and what was based on. ..................................................... 35

4.1 Remote Control - Connecting. .............................................................. 48
4.2 Remote Control - Connected. .............................................................. 48
4.3 Android control- bothScreens. ............................................................. 50
List of Tables

3.1 Comparison between possible vehicle chassis ........................................ 14
3.2 Comparison between technologies. ..................................................... 23
3.3 Comparison between distance sensors. ............................................. 26
3.4 Comparison between line detection sensors. ..................................... 27
1.1 Introduction

This is a project in laboratory structure on a model scale project to help understanding the theme of this theses: “Autonomous driving with V2I Communication”. Before explaining the purpose of this thesis, it is required to, first, define what is an autonomous vehicle. An “autonomous” vehicle is a vehicle that relies solely on its on-board equipment to collect information, take decisions and inform tasks, follow its path, follow the road rules and inform other vehicles of its position and regular status without human intervention. This type of vehicles are going to be the future and, at the moment, many companies are already trying to develop these types of vehicles.

The number of vehicles is still increasing world-wide. Ever since the implementation of automated factories the amount of vehicles in the world is still on a growing rate. More people, nowadays, use a personal vehicle instead of walking, riding or using public transportation. This growth gave industry a chance to improve by enhancing the comfort inside those vehicles [1].

If the world moves one step further, it is possible to change the driving process to be fully autonomous. Of course this brings lots of diverse issues, but on the brighter side this system needs no human interference in driving, providing safety for everyone on the road and people can occupy their time with other things while on the road. In addition, there are many anticipated benefits of automated vehicles, which may include but are not limited to:

- Fewer traffic collisions [2] because of better robustness and shorter reaction times of computers;
- Better traffic flow [3] (less congestion and higher capacity) due to less safety gaps and automatic system scheduling;
- Higher fuel efficiency [4] because the computer can work out the best route in current traffic conditions;

In this theses, the model vehicle not only will receive information from the infrastructures but also it will detect other vehicles or obstacles and allow to be controlled by a Java or Android application. For example, if there’s a speed limit, the vehicle will be able to detect it and regulate its speed accordingly.

The communication Vehicle to Infrastructure (V2I) intends to improve the vehicle automation by letting the vehicle know what is going on without the human interference.

This project was supposed to be done using an Arduino board on a small vehicle chassis with a DC motor, in a small simple road model. However due to some unfortunate inconveniences this project could not be done in Arduino as the materials required never arrived, as such it was possible to do some

---

changes and to finish it with Lego Mindstorm® EV3. It is noteworthy that this project in either case is not a real scale model and the technology used here, most likely would not be usable in a real scale vehicle but provides an awareness of possibilities.

1.2 Briefing

This thesis is organized as follows:

- Chapter 1 is where a brief understanding of this thesis is explained and its motivation.
- Chapter 2 related work which may be similar in some parts with this thesis.
- In Chapter 3 the choice of the devices to use and the solution to the problem they address are explained.
- Chapter 4 an explanation of the implementation of the previous chosen devices.
- Chapter 5 the testing phase.
- Chapter 6 final conclusions, achievements and possible future work.

1.3 Objectives

The objective of this thesis consists in developing a model-size road where autonomous vehicles can safely drive while following traffic rules. The vehicles should be able to communicate with the infrastructure while avoiding obstacles ahead of them. Finally, a laboratory project for future students from MEIC (Masters in Engineering Informatics and Computers) and METI (Masters in Engineering Telecommunication and Informatics) that will allow them to learn more about other technologies and how they can be used to implement V2I communication.
2 Background

Contents

2.1 Related Work ................................................................. 7
2.2 Summary ................................................................. 10
This project is model-size autonomous vehicle in a model-size road. Even though they seem nothing alike, there are some projects from some companies that work with real vehicles and by understanding their thought process, it is possible to help develop this relatively small project. Those companies accomplished what many thought would be impossible: for a vehicle to drive on its own without assistance from a human, even though only prototypes are available. This is something that is not really easy to do. It involves various subjects and technologies such as automation control (in order to drive on its own), computer vision (so it does not collide with any object), and artificial intelligence (to connect and know what to do in every situation). With all this challenges, autonomous driving is just a few steps from happening. There are some companies that developed some prototypes, universities that build Arduino prototypes, and, in the possible future, there will be a revolution in transportation.

In this chapter a small introduction of four different but related projects is explained.

2.1 Related Work

For the related work, it was used four model-size Arduino projects using similar technologies even if it is only in some parts. The goal of this project is develop a fully autonomous vehicle with sensors allowing it to safely drive while gathering information from the road. Even though that this project no longer uses an Arduino vehicle, the logic is the same. Knowingly that the Lego Mindstorm® EV3 is somewhat similar to the Arduino in terms of size and technology the following related work gives a great perspective on what already exists and how can that be applied to this project. The hardware may be different, but the logic should be the same.

2.1.1 Open source hardware and software platform for robotics and artificial intelligence applications

As the technology continues to evolve, more achievements can be reached using new techniques. Developments in open source hardware and software platforms allow the possibility for fast development of low budget and better learning curve system while doing the same or better as older systems. In this particular case, the authors used a combination of open source hardware and software that allowed them to build multiple systems for research, teaching and learning purposes in which there is one noteworthy: An interface between Android app and remote control gripping robot (Mobile App Controller). The hardware used for it to work was an ultrasonic sensor was used in order to detect distances between the robot and any obstacle in front, a Bluetooth module allowing the communication to flow between the Arduino and the Android device and a camera which is used in the control of the robot movement. The Mobile App Controller is divided in two parts, the Android and the Arduino. The Android is the master in terms of communication while the Arduino receives the inputs. The application receives the
commands from the user by pressing buttons or tilting the device then through the Bluetooth module it sends the commands to the Arduino. The commands are similar to the control of the vehicle, it has forward, reverse, left and right while also having the controls for the gripper such as grip, release, lift and drop. The ultrasonic sensor allows the vehicle to avoid any obstacle if it detects anything then the vehicle will abort the input to move forward and turn either way [8].

![Image of Mobile App Controller of gripping robot](image)

**Figure 2.1:** Mobile App Controller of gripping robot - image from: [8]

### 2.1.2 Line Follower and Obstacle Avoidance Bot using Arduino

In April 2017, the International Journal of Advanced Computational Engineering and Networking published a paper that had the purpose to help solve the movements and interactions problems in the industry, some machines need to move from one place to another and back to the same one. This group of 5 people designed and build a line follower and obstacle avoidance robot and this project can be divided in two parts: The Line Follower and the Obstacle Avoidance. As mention in the published paper "Line follower robot is a system which traces white line on a black surface or vice-versa. Obstacle avoidance robot is design to allow robot to navigate in unknown environment by avoiding collisions." [9] The first part, the Line Follower, the robot uses infra-red sensor that detects the black line in a white surface or vice-versa. The concept of line follower requires at least two infra red sensors for the vehicle/robot to follow one single line underneath it. This means a left infra-red sensor, a right infra-red sensor and a table scenario which consists on four different movements for the vehicle (it is noteworthy to mention that this project was build on a white surface with black lines). One, if the sensors do not detect anything (both detect the black line) the vehicle should stop moving. Two, if only the right sensor detects the black line, the vehicle should turn right. Three, if the left sensor detects the black line, the vehicle should turn left. Finally, the fourth scenario if both sensors do not detect a black line, the vehicle should move forward. Regarding the second part, the Obstacle Avoidance, the robot when it detects an obstacle in front of it stops the vehicle and by using a bluetooth module attached to the Arduino Uno it enables the user to manually avoid the said obstacle with an Android Device. By doing so, not only it allows the vehicle to be controlled but also to control itself.
2.1.3 World’s First Android autonomous vehicle

The objective of this project was to create a self-driving autonomous model-size vehicle that stays on road, overtakes obstacles and also park. This small vehicle with an Arduino board, a web camera, sensors and micro-controllers is able to do all that was mentioned. In order to do this project, it is necessary a camera to detect the lanes and obstacles, an Android application in communication with the Bluetooth on-board micro-controller that decides what to do with the information from the camera and send those decisions to the Arduino installed in the vehicle, an infrared sensor and ultrasonic distance sensors. The Android application controls the vehicle’s behavior such as manual and automatic driving, also parking and others. The infrared sensor helps determine if the middle lane is dashed or straight. The ultrasonic distance sensors help determine the surroundings and park the car. [10]

2.1.4 Autonomous Vehicle-to-Vehicle (V2V) Decision Making in a Roundabout

Theoretically, roundabouts promote a continuous flow of traffic and these are being more and more implemented in the roads and the future of vehicles consist in autonomous vehicles. We the introduction of autonomous vehicles some possible specific cases may slow down the roundabout efficiency, removing its advantages. From the University of Tuzla, reaches us a project that combines both topics and
create an Autonomous Vehicle with V2V for roundabout’s safety and improve efficiency. With the help of a QTI (Charge Transfer Infrared) sensor, each vehicle is able to detect the distance between them and the help of XBees as well modules for ZigBee protocol in communications between the cars. As proven, if there are at least two vehicles driving normally and one car decides to enter a roundabout, it, with the assistance of the QTI sensor, detects other vehicle in the roundabout. In this first detection, the entering vehicle communicates with the second one, through Zigbee sending its information. The car in the roundabout takes information and changes speed accordingly with the first vehicle ensuring the safety in both parties. [11]

Figure 2.4: Two Autonomous vehicles in a roundabout - image from: [11]

2.2 Summary

By looking at these projects succeeded, there are a lot of ideas that it is possible to retain. In the first project, we learn that using a ultrasonic sensor is useful to detect obstacles and bluetooth for the vehicle to communicate with the Android phone. The second project teaches the usage of infrared sensors to detect lane allowing the vehicle to maintain its course by using two opposite colours (black tape on white background). The third project confirms the previous premises of using ultrasonic sensors, infrared sensors and bluetooth to create an autonomous vehicles, however this one uses a video camera to interpret the traffic sigs and lanes. Finally, the last project implements the usage of XBees allowing the communication between two vehicles proving not only bluetooth is not the only alternative, but also the importance of having V2V when vehicles need to know about their position.
In this section, it will be presented the initial architecture of the system proposed. Even though the final product is not based on this architecture, this is a great piece of information not only for the future work but also to give an understatement of available technologies and hardware. First the design of the model road where the small vehicle will be driving on in Road Design. Next, on Communication a brief summary of multiple communication technologies and the chosen one for the two types of communication, Vehicle-To-Vehicle and Vehicle-To-Infrastructure. After that, in Sensors section the sensors used are explained. And finally in the Car section, the vehicle, its components and roughly its design.

3.1 Road Design

As previously mentioned in the introduction, the objective of this theses is to build a functioning and autonomous vehicle that is able to communicate with the infrastructures however in order to do that, it is required the infrastructure for the automobile to drive on. But a simple straight road is not enough to successfully and accurately test this project. For that reason, the vehicle not only needs to safely drive the designated road with two more vehicles but also needs to be able to do curves, turn right and left, respect the road signage and avoid crashing either with other vehicles or any other obstacle. In Figura 3.1 we can see the arbitrary design of the road with only its measurements and no traffic signs. This road will have two intersections, both leading to curves. Regarding the measurements, the space available for the construction of this model road is about 1.93m wide and roughly 7m long which represent in real life a total of 26.309m and 120.4m respectively.

The road’s width, in order to fulfill the possible car’s sizes in Table 3.1, will be around 0.45m, giving 0.225m for each lane, giving a safe distance of 7.5cm using the average width of the cars in the tables and that value represents a safe distance to the edge of 0.54m in a real scenario, while the length of the straight roads will have at least approximately 3m, that represents a proportion of 51.6m taking into account the average of the table’s car’s length and that value should give enough space for the vehicle to overtake safely. No speed bumps, or other similarities will be used as we are assuming that the vehicles are autonomous and never exceed the speed limit. Also, because it is not in the specifications of this projects, no roundabouts will be used. To help maintain the vehicle on the road, instead of the usual guard rails there will be black duct tape delimiting the roads. After all these limitations, there is still a way to attempt to reproduce a model-size road capable of testing all the vehicle’s abilities while respecting the road signage, and for there will be curves, crosswalks, intersections, straight roads and traffic signs.

3https://www.amazon.com/Makerfire-4-wheel-Chassis-Encoder-Arduino/dp/B00NAT3VF4, 28 - December - 2017
4https://www.amazon.com/Makerfire-4-wheel-Chassis-Encoder-Arduino/dp/B00NAT3VF4, 4 - January - 2018
Table 3.1: Comparison between possible vehicle chassis

<table>
<thead>
<tr>
<th>Vehicles (4WD)</th>
<th>Length(cm)</th>
<th>Width (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geekcreit Smart Robot Car Chassis Kits 1</td>
<td>25.5</td>
<td>15.5</td>
</tr>
<tr>
<td>DIY DC Smart Robot Car Chassis Kits 2</td>
<td>25</td>
<td>14.8</td>
</tr>
<tr>
<td>Makerfire Robot Smart Car Chassis Kits Car Model 3</td>
<td>24.4</td>
<td>15.2</td>
</tr>
<tr>
<td>Licensed Shelby Mustang GT500 Super Snake Electric RC Car 4</td>
<td>34.29</td>
<td>16.2</td>
</tr>
</tbody>
</table>

Figure 3.1: The design of the road with measurements.

3.1.1 Curves

There are two types of curves when the subject is about roads, it can be either vertical, horizontal or both. A horizontal curve provides a transition between two tangent strips of roadway (in the x-axis), allowing a vehicle to negotiate a turn at a gradual rate rather than a sharp cut. While the vertical curve provides a transition between two sloped roadways (in the y-axis), allowing a vehicle to negotiate the elevation rate change at a gradual rate rather than a sharp cut as well. In this project, vertical and mixed curves will be disregarded. The only requirements for the curve are: Width long enough for the both vehicles to pass safely (0.45 meters), angle of the curve not to sharp to allow the model vehicle to
actually turn safely and multiple types of curve (some sharper than others) to have a broad evaluation possibilities. For that, there are two major curves (top and bottom curve), and small curves on the left side of the road with different angles of curvature. In the drawing, the curves have approximately 0.45m radius, where the top curve (Figura 3.2) has 0.45, while the lower curve (Figura 3.3) has 0.47m. The difference is irrelevant as there is no reason for them to be exactly the same.

Figure 3.2: Top curve of the road with the proper measures in meters.

Figure 3.3: Lower curve of the road with the proper measures in meters.
In both curves the vehicle is prohibited from overtaking and its speed limit is reduced.

3.1.2 Crosswalk

This particular element needs to be wide enough to accommodate the pedestrian flow in both directions [12], in this particular case there will not be a pedestrian traffic light, so the length of the crosswalk is as long as the width of the road, in fact there will only be a tag notifying if the there is a pedestrian waiting to cross or not. In the current project there is the assumption that the the pedestrians will not have a bizarre behavior and they will wait for a vehicle to stop if one is coming. Because of its position, there are only signs of limiting speed when the vehicle comes from a straight road. If the vehicle just passed
the intersection, no sign should appear prior to the crosswalk, except for the crosswalk sign.

### 3.1.3 Intersection

When it comes to the intersection, there is not much to say about this other than it is an intersection where four roads meet. There is a traffic light for each road, and the angle between two adjacent roads is 90 degrees. For a vehicle to change course to the right it needs to follow the right continuous line. To change course to the left it needs to cross halfway the intersection then a sharp left to prevent going into incoming traffic. It is noteworthy to mention that a vehicle cannot stop inside the intersection.

### 3.1.4 Traffic Signs

The objective of the project is to add communication between the vehicle and infrastructure and so, the infrastructure, in this case, the traffic signs are mandatory to be implemented in the road (as it can be seen in Figura 3.2, Figura 3.3 and Figura 3.4. There are two speed limits one relatively higher than the other. The upper limit is currently on the straight road, while the lower limit is on the curvy part of the road. These traffic signs represent two different velocities of the vehicle just like in the real world and whatever their real numbers are, they must have a visually significant difference in speed. In the road we can find the following signs:

- Crosswalk Sign: Right before the crosswalk;
- Traffic Lights Sign: Also right before the crosswalk as a heads up for a traffic light ahead;
- End prohibition of Overtaking: After the crosswalk in the direction away from the intersection and after both curves;
- Prohibition of Overtaking: After the crosswalk in the direction towards the intersection and before both curves;
- Start of Speed Limit: After the crosswalk in the direction away from the intersection and after both curves;
- End of Speed Limit: After the crosswalk in the direction towards the intersection and before both curves;
3.1.5 Straight Road

There will be one straight road at least 3 meters long and 0.45m wide. The wideness is because of the space requirement for two car lanes and the safety space between them. The length of the road is because of the undetermined space required for a vehicle to safely overtake another. It is possible that the length given is too large and excessive, however if needed it can always be shortened. It is easier to shorten the road instead of making it larger. Nevertheless, the straight road must be big enough to have at least 3 vehicles in a straight line.

![Figure 3.4: The middle of the road: An intersection plus crosswalks and the straight roads](image)

3.2 Communication

In the current project, communication between vehicles and infrastructure will be used so that the vehicle is aware of its surroundings. This will make them aware while not using a camera to actually see other surroundings while not consuming too much energy, this means that only low energy technologies were considered.

There are multiple types of technology possible for the two different cases: communication between vehicles and communication between infrastructure. For both cases we need to ask ourselves “what do we need, what is the problem to achieve it and which technology fixes it?” So, what do we want? In
communication between infrastructures, any simple low energy technology should suffice for the vehicle to learn its current position and any traffic laws in the area. However in communication between vehicles, we need at least 2-way network or similar between targets. Next, we require enough bandwidth for the vehicles to send its position, lane, speed, deceleration and the number of cars in front of him. That is, the minimum amount is 1 byte for position (if we use a tinyint), 1 byte for lane (it was possible to use less, but let's maintain the minimum size 1 byte to simplify), another byte for the speed and 1 more for the size of the front traffic, giving a total amount of 4 bytes. The reason for this is two things: One, the position will be determined by traffic signs and road marks as each of them will have a byte telling the position from the start line which is position 0. Second, the main idea to transmit this type of information will be by broadcasting on the device's name, reducing any latency of binding the connection between two or more vehicles. However, there is a slight problem with this type of communication and that is the fact that because of the moving vehicles not always there is a direct line between both elements and they may lose range. That is why it is required to ensure that the signal doesn't break from time to time, or at least in critical moments.

In the end, they use different technologies because of their different purpose, as so different technologies were chosen for either of them. The following technologies were taken into account:

- ANT
- Bluetooth Low Energy v4.0
- RFID
- Wi-Fi
- ZigBee

### 3.2.1 Types of Communication Protocols

#### 3.2.1.A ANT:

ANT is a practical wireless sensor network protocol designed for low power, ease of use, efficiency and scalability. Its protocol runs on 2.4GHz ISM (Industrial, scientific and medical) band. ANT easily handles multiple topologies such as broadcast, peer-to-peer, star, mesh, among others. Its design is suited for any kind of low data sensor network in wireless sensor networks. According to the proprietary company itself, ANT requires minimal micro-controller resources reducing system costs and its nodes can operate for years, making it one of the technologies with the most energy efficiency with a signal range of up to 30m. [13]
Each node in an ANT network consists of an ANT protocol engine and a host controller (MCU). The ANT engine encapsulates the complexity of establishing and maintaining ANT connections and channel operations while the host controller handles the particularities of the application. Furthermore, communications between ANT nodes is made between channels. Each node can connect to other nodes via dedicated channels and each channel usually supports only two nodes but a single channel can support up to 65535 nodes. ANT communication is based in a master-slave communication, where the master is the primary transmitter and the slave the receiver. The type of communication between nodes in the channel is determined by ANT data type and it can be of three types: Broadcast, Acknowledgment and Burst transfers. Broadcast Data is system default and the data is always sent from Master node to Slave node or vice-versa when there is a request from the save. Broadcast data consumes an average of 0.012mA, lower amount of RF (Radio Frequency) Bandwidth because of the one way transmission and it has a signal rate of 12.8kB/s. Regarding the Acknowledged Data, in a bi-directional connection an acknowledge data packet is sent back. Whenever either node sends an acknowledge data packet, the receiver responds with the acknowledge message back to the sender. Afterwards, the initial sender is in charge to decide if the message received is correct or incorrect. In terms of power, this type of data consumes an average of 0.059mA and has a signal rate of 20kb/s [14]. It provides more control on the application itself at the cost of more power and bandwidth usage. Finally the Burst Data type is best when there are large amounts of data to be transmitted between devices but at the cost of more power, more specifically 0.167mA and a signal rate up to 60kB/s.

3.2.1.B Bluetooth Low Energy 4.0:

Bluetooth Low Energy 4.0, as the name indicates, is a low energy wireless technology developed by Bluetooth Special Interest Group (SIG) for a short-range communication. This technology is the fourth version of the Bluetooth technology previous developed by the same group and its most known by its low-power solutions on contrast of the previous versions. In opposition to other technologies such as ZigBee, BLE works as a one-hop solution, that is, a point-to-point connection which in a piconet it can have simultaneously connected 7 nodes. BLE operates in 2.4GHz with 40 channels of 2MHz reaching a range up to 100m, with an average energy consumption of 0.0239mA and data rates up to 1Mbps, where they can be either: Advertising channels, which are used for device discovery, broadcast transmission and connection establishment, and data channels, which are used for bidirectional communication between connected devices [15]. When a connection is created, there are always two types of devices, a master and a slave. The master works as a initiator while the slave as an advertiser during the creation. Also, to note that a master can be connected to multiple slaves while a slave can be connected to one and only one master. One of the problems of these types of connections is the power consumption, and the
solution for this was to put the slaves in sleep mode by default and in time, they wake up and check for possible messages in from the master, if there are none, they go back to sleep. But the connection cannot be decided by the slaves, in a priori the master notifies the slave when to wake. In the BLE there are multiple profiles but the most important one for this project it would be the GAP profile. The BLE GAP profile defines four possible roles for the device: Broadcaster, Observer, Peripheral and Central. A device in Broadcaster role, as the name mentions it, only broadcasts data, while an Observer completes the Broadcaster by only receiving the data transmitted by the Broadcaster. The Central role is for a device that begins and manages multiple connections while a Peripheral role is for a simple device in point-to-point connections with Central [16]. The explanation of these profiles have a purpose and therotically it can help overcome the latency of connection problems by skipping the connection part and transmit information directly to the device’s name.

3.2.1.C  RFID:

RFID - Radio Frequency Identification - is an AIDC (Automatic identification and Data Capture) wireless technology that allows the read and write of data through radio signals. This system, according to [20] is composed by three elements, the reader, the tags (also known as transponders) and the data collection application. The reader is responsible to many things including powering the tags, identifying them, reading its data, writing data on it and communicating with the third component. While the tag is attached to an object or target usually connected to another system and ready to transmit that system information to an RFID reader to be processed. Last, but not least, the data collection application receives data from the reader and interprets the data with the current system it is in. There is also an important part of the RFID system according with [20] and that is the “fourth critical component in RFID systems: the air interface between the transponder and the reader”. This fourth component surely influentiates how the system reacts, too much interference and the system will not work. There are many applications for a RFID System, it can be used as an identification system opposing bar codes [21], it can be used as an indoor guide and tracker [22] and it can be used as a communication device to help develop an IoT system [23]. And for different cases we may need one of the two different types of transponders (tags), passive and active [24]. As the name mentions it, passive transponders lack a power supply instead, they receive operating power from the electrical field generated by the reader. This means that a passive transponder doesn’t work unless there is a nearby receiver sending it energy. On the other hand, an active transponder has its own power supply and has, not only a larger read range but also a larger size (as it has its own internal battery). The difference in usage between these two types is that, a passive tag is usually used when it is required a large number of tags, as it is less cheaper than an active tag and in places where reading at close range is not a problem. While the opposite tag,
the active tags, are better when deployed in smaller numbers and there is a need to read something in long range. In a RFID System, the transponders are not the only thing that can be different, there are multiple operating frequencies, more concretely three types: Low Frequency, High Frequency, Ultra-high (or Very High) Frequency [25]. A Low frequency RFID system is one that operates in the range between 125 - 134 kHz, can be read up to 10cm long, works well around liquids and metals and require minimal power, <100mA, however the amount of memory is short and it has low data transmission rate, from 4 to 8kbps. A high frequency device can go from 3 - 30 MHz (typical devices work in 13.56 MHz) it is somewhat more expensive than the lower frequency but it is range is an approximately of 1m. It has more memory and requires higher power, around the 130mA to work giving more transmission rate from 6.7kbps up to 848kbps. Finally, the Ultra-High Frequency (or Very-High Frequency) usually can go from 300MHz up to 3 GHz, its reading range can go up to 25m if passive, and from 30 to 100+ m if active. It has long range, large memory and high data transmission rate that depends on the number of the modulation type and the number of the sub-carrier cycles per symbol, however its price skyrocketed in comparison with the previous ones, being the average price of a high frequency tag is $10 while the price of UHF frequency tag can go over $100.

3.2.1.D Wi-Fi:

In the last 20 years, wireless communications have provided and continue to provide easy access to the Internet in all devices especially the portable ones and this raises a rather huge problem that is the power consumption. WiFi, also known as IEEE 802.11, was designed for high-speed and short-range communications not only for portable devices such as Laptops and PDAs but also for general PC that is away from the access point. The most known types of WiFi are the 802.11b, 802.11g, 802.11n and 802.11ac. Either of the types has its own speed and bandwidth. In 1999, the standard was developed considering a home environment and an office environment with a data rate of 2Mbps initially that grew up to 11Mbps with the deployment of 802.11b. Afterwards, with newer extensions of 802.11g, the maximum data rate increase almost five times, 54Mbps and it is compatible with the previous gear. But, the number of devices kept growing and the need for faster data transmission rates also grew and a newer technology was 802.11n that can go up to 300Mbps of network bandwidth, a indoor range of 70m and is also compatible with both 802.11b/g gear. According with [27], the average device expends less than 180mA sending or receiving and less than 40mA processing information. Up until now, the WiFi technology operating in a specific frequency, 2.4GHz, but now with the present technology a new Wi-Fi band appears, the 5GHz band. This latest version, of the most common standards, uses dual-band wireless technology, this means that it supports both 2.4GHz and 5GHz WiFi band. It is capable of speeds up to 1300Mbps in 5GHz and up to 450Mbps in 2.4GHz with a maximum indoor range of
35m, however it doesn’t offer a very good compatibility with the b/g technology because of them being outdated and a 2.4GHz WiFi Driver cannot detect a 5GHz wireless access point [26]. In both cases the number of nodes is limited to 65536.

There is also a new technology called Low Power WiFi and, according with [27], it uses IEEE 802.15.4, “a standard specifying the physical layer and media access control for low-power and low-rate wireless personal area networks”. It supports a star, tree, cluster tree and mesh networks. It can also be used as a peer-to-peer network however it requires a PAN (Personnel Area Network) coordinator in order to initialize the network start-up procedure. With a 250Mbps and a maximum power output of 1mW its range is in fact lower from 10m to 30m, but offers a better power consumption in overall, much less than the power of a regular WiFi. When idle, the WiFi Lowpower consumes an average of under 10mW, while expends under 6mA transmitting and under12mA listening and transmitting at full power [27].

3.2.1.E ZigBee:

ZigBee is a low-cost, low-power consumption, two-way, wireless communication standard. Is used mostly in mesh networks for connecting sensors. As well as IEEE 802.15.4, Zigbee is also a low data rate wireless network standard with reduced costs. ZigBee can support three types of networks, such as Star, tree and mesh network as mentioned earlier. A ZigBee network consists in a coordinator that is responsible for initializing, maintaining and controlling the network and multiple ZigBee routers. While a Star network has a coordinator and devices are directly connected to this coordinator, in a tree or a mesh network the devices communicate with each other with multi-hops. A device is allowed to join the network as an end device and is successfully joined if the coordinator accept it [28]. According with the official manufacturer specifications manual, [29], ZigBee mesh networks allow full peer-to-peer communication and it can support 65536 nodes. Also, each ZigBee device can start a broadcast message and it only requires 4 bytes plus the message to do so. And according with [30] ZigBee’s max rate is 250kB/s, its frequency can be either 868/915MHz or 2.4GHz with a range up to 100m (outdoor) and in transmission mode its energy consumption goes from 25 to 35mA.

2 It varies accordingly with the device and the specification.
4 The links from where the prices were retrieved are displayed, by order, in the appendix.
<table>
<thead>
<tr>
<th>Frequency band (MHz)</th>
<th>ANT</th>
<th>BLE v4.0</th>
<th>RFID (LF)</th>
<th>WiFi(n)</th>
<th>ZigBee</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2400</td>
<td>2400</td>
<td>0.125 - 0.134</td>
<td>2400</td>
<td>868/915 or 2400</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13.56</td>
<td>5000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>300 - 3000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2400</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max signal rate (kbps)</td>
<td>12.8 - 60</td>
<td>100</td>
<td>4 - 8</td>
<td>300 000</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td>2400</td>
<td>1300 000</td>
<td>848</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td></td>
<td>1, 2, 4, 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Freq/1, Freq/2, LF/4, LF/8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max node</td>
<td>65533 [13]</td>
<td>8</td>
<td>N.A</td>
<td>65536</td>
<td>65536</td>
</tr>
<tr>
<td>Awake current [31]</td>
<td>2.9</td>
<td>4.5 [32]</td>
<td>&lt;100</td>
<td>&lt;180</td>
<td>9.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>130</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N.A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Price (Average in $)</td>
<td>5.69</td>
<td>14.90</td>
<td>2.50</td>
<td>6.95</td>
<td>14.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.83</td>
<td>&lt;23</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[365.00]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Range (meters)</td>
<td>1 - 30</td>
<td>1 - 100</td>
<td>&lt;0.010</td>
<td>70</td>
<td>&lt;30 (indoor)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;100</td>
<td></td>
<td>&lt;100 (outdoor)</td>
</tr>
</tbody>
</table>

Table 3.2: Comparison between technologies.

### 3.2.2 Communication between Vehicles

This type of communication will be used between two vehicles in a way that they easily understand and locate the other one driving positioning, acceleration, deceleration and speed. After some investigation, it was possible to discover multiple communications that were low energy but not all of them were optimal to choose. As mentioned earlier, the total amount of the bandwidth is, at least, 4 bytes, and all the considered technologies options can take 4 or more bytes of bandwidth so there was nothing to worry about in it. One major point is the direction and if there is a direct line towards it and some of the wireless technologies mentioned earlier are not able to fulfill this requirement. To summarize, almost all technologies mentioned do all that with the exception of RFID that (almost) requires a direct line between the emitter and receiver. In the end, the technology chosen was **Bluetooth Low-Energy** because of not only its library support ready to work for Arduino but also because of its low power consumption mentioned in the previous Table. The reason behind the minimum data transmitted is so that, if it is possible to have the smallest data possible so it can compress the information into the device’s name so that when a vehicle with a broadcaster profile will let everyone around it every x seconds about its current location, speed, last traffic sign detected and possibly if it has a vehicle ahead.
3.2.3 Communication between Vehicle and Infrastructure

In opposition to the prior communication, “Communication between Vehicles”, in this type of communication there are different requirements in choosing the technology as it was already seen. In this case we have two elements, one a moving vehicle, while the other a static road element. The vehicle only needs to read the element and doesn’t require to send its information to the static element. So there is no need to create a connection and exchange information. Taking into account all the choices available, the chosen solution was High Frequency RFID. This type communication allows the reader to faster read the tags that represent a traffic symbol. Because of the lack of need of a reply, the RFID (HF) is a good solution that provides an acceptable range of 12cm, which matches in a rough approximate of 144cm if the calculations were made in real life, this is more than enough to be detected by the vehicle. RFID ensures all that while the other wireless protocols do not. As we can see in Table 3.2.

3.3 Sensors

The vehicle can not only rely on the communications to drive autonomously, sensors are required not only to detect your surroundings that are not “smart” elements, for instance a ball, branches, rocks, or even a pedestrian, but also stay on the road. For those two actions, it will be used two types of sensors: one to detect distances and another to detect the lanes.

3.3.1 Distance Sensor:

In order to avoid any collision between the vehicle and anything else, the car needs a sensor to estimate the distance between it and the target so that it can estimate its distance and regulate the vehicle’s speed so prevent the collision while not stopping if not necessary. For instance if a vehicle detects an object too far away, it doesn’t make sense to emergency brake while having 1 meter distance. To deliver these needs it was taken into account two sensors: Ultrasonic Sensor and Infrared Proximity Sensor. Both these sensors are currently being used in numerous projects as a mid-range distance measurements in robots and vehicle applications, and both use different materials and formulas to calculate the distance to the object.
3.3.1.A Ultrasonic Sensor

An ultrasonic sensor is a device that emits ultrasonic waves and waits for its reflective waves. If there is an object, the same waves reflect on that object and are detected by the same device, on another hand if no waves are reflected then there is no obstacle in front of the sensor [33]. By knowing a priori the speed of sound, this type of sensor can measure the distance between the sensor and the object by the following formula:

\[
\text{Distance} = \text{Speed of Sound (in the air)} \times \left( \frac{\Delta \text{Wave time}}{2} \right)
\]

This will help prevent any collisions by the vehicle. As with everything, this type of technology comes with advantages and disadvantages and one of the reasons to use this technology is because of its lower price compared to optical sensors and image processing, while knowingly that, usually, the emission of the ultrasound signal have an opening angle and it may be influenced by the object, leading to false positives.

3.3.1.B Infrared Proximity Sensor

In opposition to the ultrasonic sensor, the infrared proximity sensor works, as the name indicates, based on the detection of a light wavelength in the the infrared spectrum. This sensor is composed by mainly two elements, a IR LED (Light Emitting Diode) that transmits the infrared light to an object in front and a light receiver that receives the reflection of the emitted light. This type of devices measure the distance based on the difference between emitted light and received light. However there are some downsides. If the object has a uneven surface and a colour that difficult the reflection of the emitted light, false negatives may occur, this is the sensor doesn’t detect the object but it is there. Also, it can easily be affected by other light sources that may induce the device in error. [34]

3.3.1.C Comparison

So, to reach a decision to decide which type of device to use there were mainly two choices with specific devices, HC-SR04 for the ultrasonic sensor, and the GP2Y0A21YK for the infrared proximity sensor. As we can see in 3.3, the values side-by-side of the two elements retrieved from the data sheet of the seller’s website “SparkFun” can be easily compared to make a decision on which device to use. With an increased range, lower energy consumption and at lower price the Ultrasonic Sensor HC-SR04 was the chosen device.
### Ultrasonic Sensor

| Range (cm) | 400 | 80 |
| Voltage (V) | 5 | 5 |
| Price ($) | 3.95 | 13.95 |
| Energy Consumption (mA) | 15 | 40 |

Table 3.3: Comparison between distance sensors.

#### 3.3.2 Lane detection sensor:

A vehicle to safely drive in a road needs to follow lanes and this vehicle needs a device so that it can do so without the assistance of a camera. As the road lines will be on the floor, one way to detect them is to place a sensor aiming at the floor and there are only two outcomes, either a lane is detected or it is not. In a way, what is required for this vehicle to properly function is a device that can safely and correctly detect if it is on a line or not. For that purpose, there are multiple choices of sensors that can do it however two were taken into account: Infrared Sensor and RGB (Red/Green/Blue) Sensor.

#### 3.3.2.A Infrared Sensor:

What is called an infrared sensor is mainly an infrared transmitter coupled to an infrared receiver. This sensor is comprised of a LED, the transmitter, that emits its radiation towards the ground and that same radiation is reflected and captured by the infrared receiver. The output is different by the type and colour value (grey values) of the incident object or surface [35]. This means that for instance, the white colour has 100% reflection, while the black colour has 0% and a grey-ish one has somewhat between those values. After the radiation is received, the data is worked and depending on that information the vehicle might or might not correct it is course. For instance lets say that the right outer road line is black, if the infrared receiver detected anything that’s not white then the vehicle would know that, if nothing is done then it would go out of bounds (out of the road). This infrared sensor will allow the vehicles to follow the lines maintaining in road while preventing going off-road. For instance, let’s say if the right side sensor detects the black line then the vehicle knows that its on the edge of going off-road. To correct its course, it would need to go slightly left.

---

1 All data retrieved from SparkFun website - [https://www.sparkfun.com/products/13959](https://www.sparkfun.com/products/13959)

2 All data retrieved from SparkFun website - [https://www.sparkfun.com/products/242](https://www.sparkfun.com/products/242)
3.3.2.B RGB Sensor

The road is going to be a white surface with black tape making the lines and a green tape only on the center of the road when the vehicle can overtake. As mentioned there will be three colours available (white, green and black) and a good way to detect them is a RGB Sensor. This sensor is similar to the infrared sensor however instead of the infrared emitting light, the emitter uses white light and the detector captures that reflection detecting the colour. In fact, it does not retrieve the colour itself, the receiver follows a RGB model that indicates that any colour can be reproduced joining same or different amounts of red, green and blue. As such, the output of this sensor is the amount of red, the amount of green and the amount of blue used in the reflected colour which can later be deciphered to the actual colour name as we know it. For instance the a colour that reflects 0% red, 0% green and 0% blue in a rgb model, is the colour is black. This sensor can easily differentiate between the white, black and green colour and by doing so it is a good device for this problem. [36]

3.3.2.C Comparison

The two technologies presented provide the solution for the problem but to correctly compare between which technology to use, it is require to compare sensors of each type. In this case two elements were use, one for each and both sold by the same website “Adafruit”. For the Infrared Sensor we have a Reflective IR sensor and for the RGB Sensor there is RGB Color Sensor with IR filter and White LED. As we can see in the 3.4, the infrared sensor is cheaper than the RGB colour sensor and voltage is the same even though that RGB colour sensor can work with both from 3v up to 5v. In terms of range the infrared sensor reaches up to 1cm while the RGB Colour sensor varies due to the luminosity in the room. Taking these elements into account, the technology that will be used is the infrared sensor.

<table>
<thead>
<tr>
<th></th>
<th>Infrared Sensor(^1)</th>
<th>RGB Colour Sensor(^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range (mm)</td>
<td>2-10</td>
<td>Varies with luminosity</td>
</tr>
<tr>
<td>Voltage (V)</td>
<td>5</td>
<td>3-5</td>
</tr>
<tr>
<td>Price ($)</td>
<td>1.95</td>
<td>7.95</td>
</tr>
</tbody>
</table>

Table 3.4: Comparison between line detection sensors.
3.4 Car

To ideally test this project the perfect amount of cars should be three. But two is more than enough there are multiple cases where it is able to simulate real case scenarios, vehicles in an intersection and emergency braking while all cars are on a line. Each vehicle itself will be built with all devices needed to do the following objective: Driving safely while communicating with infrastructures and other vehicles. For that it will require, at least, these elements per vehicle:

- 1 Model Vehicle Chassis;
- 2 UltraSonic Sensors (back and front);
- 3 Infrared Sensors (left, center and right);
- 1 HF RFID Reader;
- 4 Wheels;
- 2 Surfaces (included in the Model Vehicle Chassis);
- 4 DC motor;
- 1 Arduino Genuino 101 (with Bluetooth Low-Energy Modules v4.0 incorporated);
- 1 PowerBank or 1 Battery holder;
- 1 Adafruit Motor shield;
- 1 BreadBoard.

First of all, there will be an Arduino board in the center of the vehicle with the breadboard, so that it can reach any required element. The 4 DC motor will be under the top surface connected directly to the wheels, while also connected to the Adafruit Motor Shield. An RFID reader would be, ideally, connected under the vehicle or on its side in a way so that it easily detects the transponders on the road. The energy supply (the PowerBank or the batteries holder) will be laid on the lower surface, connected to the Arduino in order to supply the required energy for it to work and allowing the Motor Shield and other elements to function. As it was out of scope, the aerodynamics of the vehicle were not taken in consideration and also the wheels surface in order to improve the movement of the same. The Ultrasonic Sensors will be placed in different parts of the car, one in the front to prevent crashes and allow overtakes, while the other one would be placed in the back of the vehicle to allow the possibility of a reverse implementation. The Infrared sensors are all in front of the vehicle but one on the left side, on the center and the last

1 All data retrieved from - https://www.adafruit.com/product/2349
2 All data retrieved from - https://www.adafruit.com/product/1334
one on the right side. This allows the vehicle to detect if it is going out of bounds where should it go to correct the course. For the vehicle to steer, the wheels from one side will accelerate while the other side decelerates or vice-versa, allowing the vehicle to be able to turn, slightly turn or hard turn using only the motors. For the assembly of all elements, some may require the use of a resistor to perform correctly.

3.5 Summary

The solution provided is one of many possibilities to achieve the same goals. By having a vehicle with RFID for the vehicle to infrastructure communication, Bluetooth for communication between vehicles, infrared sensors to maintain its course in the lane, an ultrasonic sensor to avoid obstacles and an Arduino to control all these elements, making them work together autonomously. However this cannot be achieved without those materials. Unfortunately, the materials described in 3.4 never arrived due to bureaucratic details, and approval of the respective university department and after a long wait, an alternative was needed to complete this theses. As mentioned, there are many alternatives however not all of them can be as optimal as this, that's why the next solution is something that can fulfill the minimum requirements of this theses.

Due to unfortunate events the previous architecture cannot be used at this point because the required elements were unavailable. Instead a LEGO Mindstorm® EV3 was available and it was used used to finish this project. The previous architecture was planned for the Arduino device however in order for the same project to work, it is required to do small multiple changes for this project to be adapted into this piece of hardware. In this chapter the final architecture is going to be explained as with each element used. The system proposed will be the following: First, the Car - Adapted section, the vehicle build, structure and assembly. Next, the Sensors section explains what type of sensors were used. Afterwards the Communications, where a summary of the communication technology will be presented as well. Finally, the design of the road Road Design where the LEGO Mindstorm® EV3 will drive on.

3.6 Car - Adapted

For this project and knowingly that there were issues with receiving the supplies previously mentioned in the Car section from chapter 3, an alternative appeared: LEGO Mindstorm® EV3. This device comes with multiple pieces and hardware that allows the user to create and command their own robotic LEGO construction. The good thing about this type of device is that with the pieces available anyone can create a robot, a creature, a machine or even a vehicle. But this full pack can be divided in three parts: The programmable brick, the hardware and the LEGO pieces.


3.6.1 Programmable EV3 Brick

The programmable brick is the brain and heart of the project. It has the same purpose as an Arduino board. It serves as a control center and power station, it is responsible for running code, storing information, collecting information from sensors and supply energy to all the hardware connected. It is composed by an SD Card port for custom or additional data if it’s memory is not enough, it also has a USB port for Wi-Fi dongle to allow the brick to connect with routers for user purposes. It has the ability to connect with the internet, although in this case it was not needed. An PC port (Mini-USB) to connect the device to a computer. It has 8 Ethernet ports, where four of them are strictly for motors (Output Ports) while the other four for sensors (Input Ports), both will be explained in the Hardware section next. For the user interface, the brick has an LCD display and five input buttons and one escape button. The LCD display shows what’s happening inside the brick, as well allows the code to print information to the user, either numerical or graphic responses. While the five button allow the user to provide input to the code and move through the its environment, the Escape button provides an Abort option to cancel, abort a running program and even shut down the device. The Brick also has a Speaker that allows the play of some sound effects and a Status Light on the five buttons that helps the user to know the current brick status.

3.6.2 The Hardware

As mentioned earlier, the vehicle comes with multiple hardware devices either sensors or motors however only four of those were used, but here’s an explanation of them all.

3.6.2.A The Motors

There are two types of motors for the Mindstorm® EV3, the Large Motors and the medium motors. The large motor acts similar to a DC Motor, while the medium motor works as a Stepper motor. The Large motor runs at 160-170 rpm with a running torque of 20N cm\(^1\) and a stall torque of 40N cm, while the Medium Motor runs at 240-250 rpm with a running torque of 8N cm and a stall torque of 12N cm. This means that the large motor is slower but stronger in comparison to the medium motor. For this project only two large motors were used.

3.6.2.B The Sensors

There are multiple sensors for the programmable brick, Colour Sensor, Touch Sensor, Ultrasonic Sensor, Infrared Sensor and Remote Infrared beacon (not available in this pack), being the most important one the Colour Sensor. The Colour Sensor is a digital sensor that can detect the colour or intensity

\(^1\)The torque units are usually Newton metre, but in this case it is Newton centimetre.
of light that enters the sensor. This particular sensor has three different modes: Colour Mode, Reflected Light Intensity Mode and Ambient Light Intensity Mode. In Colour mode, the colour sensor uses an RGB colour model and accordingly detects the colour and outputs the detected colour to the brick by detecting the amounts of red, green and blue on the reflected colour. There are seven colours recognizable and predetermined by the colour sensor however due to some issues this seven colours aren’t reliable so fewer colours were used. In Reflected Light Intensity Mode, the colour sensor measures the intensity of light reflected back from a red light - emitting lamp. The last mode, Ambient Light Intensity Mode, measures the light in the environment surrounding the sensor. This colour sensor sample rate is 1kHz/sec. The Touch Sensor is an analog sensor that can detect when the sensor’s button is pressed and when it is released. It can be programmed to action using three conditions - pressed, released or bumped. The Ultrasonic Sensor generates sound waves and reads the echoes to detect and measure distance from objects up to 2.5 metres with an accuracy of 1 cm average. The Infrared Sensor is a digital sensor that can detect infrared light reflected from solid objects. In this case it’s used in proximity mode where emits light waves with a max range of 70cm and reads the reflected light calculating the distance between the vehicle and the reflected light. These reads may not be too accurate due to irregular obstacles or surfaces. Finally the Remote Infrared Beacon, allows to remotely control the vehicle through the infrared beacons with a max distance of 2m in the direction it’s facing. For this project, only the Colour Sensor and the Ultrasonic Sensor were used.

### 3.6.3 LEGO® Pieces

A large amount of LEGO® pieces were needed to assemble the vehicle to give it a vehicle form while holding the EV3 Brick and sensors.


**Figure 3.5: LEGO® Mindstorm elements**

---

1Image retrieved from: https://www.lego.com/en-us/mindstorms/products/mindstorms-ev3-31313 at 28/09/2018
3.7 Sensors

Although there are five sensors available for the vehicle, only two were used: Colour Sensor and Ultrasonic Sensor.

3.7.1 Colour Sensor

The Colour Sensor is a RGB Sensor and not only it is used to detect the road but also to detect any traffic signs. It is placed around 2-4 millimetres from the floor and connected below the vehicle facing directly downwards without an angle so it has the best accuracy while examining the surface. Comparing the effectiveness of the previous architecture we can compare the usage of the Colour Sensor with both the previous Infrared Sensor and the RFID communication. In fact, the Colour Sensor will allow for the vehicle to stay on the road, doing the same as Infrared Sensor, while also detecting other colours representing Traffic Signs, the same as the RFID. Due to the lack of colours which the Colour Sensor can correctly detect, this sensor can not compete equally to the usage of RFID where the latter could be used for numerous RFID Tags which represent Traffic Signs and the position of those. The Colour Sensor does not have the capacity to safely and correctly detect Traffic Signs with position of those. For this main reason the vehicle will no longer have a Vehicle-To-Vehicle communication. Let's have an hypothetical case, say that there are two vehicles in a circuit and one detects the colour "Red" which represents (for example) a STOP sign. Now, if he sends that information to the other vehicle, this second vehicle has no idea where the first vehicle is Stopped because there's not a way do detect its position unless by motor tracking and if that vehicle came across a roundabout or a crossroad then the traveled distance values would be already wrong. In the positive side with one device (Colour Sensor) we can do the job of almost two, the Infrared Sensor and RFID.
3.7.2 Ultrasonic Sensor

Luckily there was an Ultrasonic Sensor compatible with the EV3 and for that reason the same technology could be used for the same purpose. It is used to detect any obstacle in front of it. It is placed in front of the vehicle like a pair of headlights. The information used in this device is the same as in Ultrasonic Sensor in chapter 3.

3.8 Communications

Due to the limitations of the programmable brick, there are only a few technologies available for the communication, Wi-Fi and Bluetooth. Because of the exclusivity of these two technologies the advantage of having a Vehicle-To-Infrastructure and a Vehicle-To-Vehicle communication is lost. Therefore only the V2I communication was implemented as it is now made with a colour sensor, while the V2V was removed due to the fact that the vehicle cannot pinpoint its position. To replace the V2V communication it was implemented the communication between the vehicle and a controller (Computer or Android).

3.8.1 Vehicle-To-Vehicle

This type of communication was removed due to the fact that, without the RFID there isn’t a way to track the vehicle’s positioning and also even if it was to give out the current speed or the traffic sign there is no way that the receiving vehicle knows what information is for it or even the sender’s position. This lack of knowledge may give incorrect inference which causes more problems than solves them.

3.8.2 Vehicle-To-Infrastructure

For the Vehicle-To-Infrastructure, the vehicle will use the previously mentioned Colour Sensor allowing the car to determine whether it is on the road or what traffic sign is currently active. No information is given to the road, being this type of communication one-way only (the car gets information from the road).

3.8.3 Vehicle to Controller

The Vehicle-To-Controller communication refers to the communication between a controller (Computer using Java, or an Android application) and a vehicle. This communication could be either Wi-Fi and/or Bluetooth, however for the Wi-Fi to work it is require to obtain a Wi-Fi Dongle (which is not included in the pack). Therefore, the Bluetooth technology was the solution. The Mindstorm uses Bluetooth® PAN network in order to communicate with other devices instead of regular Bluetooth®.
For this communication, there are three modes of controlling, the Manual mode, the Semi-Manual mode and Automatic mode. The Manual mode consists in using the controller to control every movement, its speed and direction. The Semi-Manual mode allows the user to control only the route to take, for instance in the next crossroad which turn should the vehicle take. Finally, the Automatic mode, the user can only see the vehicle status and change to another mode.

3.8.3.A Bluetooth PAN

As it was not fully explained earlier, a Bluetooth PAN is a technology that enables an Ethernet network with wireless links between multiple devices allowing one device to connect with another via IP address. This means that the Bluetooth network simulates a regular Ethernet network from an application's perspective. A PAN is an ad-hoc network of devices communicating on a standard network configuration. In this type of network there are only from 2 to 8 participating member devices at the same time, with each one of them with a single role such as: A PANU (Personal Area Network User) which works as a client of the network. A NAP (Network Access Point) is a device with the role of the server or router of the network. And lastly, a GN (Group ad-hoc Network) device acts similar as a NAP with a little difference. The PANU device is a device that seeks entry into a network or participating in a Peer-To-Peer connection. A PAN-GN is a device that assumes the role of a forwarding node or host and the attached PANU devices act as clients of that node. The wireless network is formed without the need of additional hardware like a hub or router as in the case of conventional cabled network. A Group Ad-hoc Network (GN) is a temporary network. It is called ad-hoc because of its created for a particular purpose which is to connect up to eight (including the host) available Bluetooth-enabled devices in the surroundings, that are willing to participate. Finally, a PAN-NAP works as either a proxy, bridge or a router between an existing network ant the Bluetooth-enabled devices. The PAN-NAP takes the same amount of devices as the PAN-GP and it can also communicate with any device in the network.

3.9 Road Design

After much thought, the road design was something that either vehicle whether it is an Arduino or a Mindstorm®, however due to its capacities somethings needed to change. First of all the the road will be a single road, so that the vehicle could follow the road using the colour sensor. This means that the road will be similar to a circuit, in fact one some of the testing experiments were based of real F1 Circuits, Singapore to be exact.

The difference between the components before and after the changes are that now there is only one lane, the traffic signs are adjacent to lanes so that the vehicle who's always driving in the right side of the lane's limit can detect it. There are five colours used:
• Blue - Lane;
• Green - Safe zone, can go faster;
• Red - Tight curve, go slower;
• Yellow - Crossroad (can go left, front or right);
• White - Offroad.

The road consists in white paper sheets A3 while the lanes and the traffic signs are made from coloured tape. The objective at this point was to build a road that fulfill the needs and encompasses all traffic signs and scenarios. Further explanation will be provided in Chapter 5.

\(^2\)Image retrieved from: http://www.singaporegp.sg at 12/09/2018
Solution Implementation

Contents

4.1 Development Process ................................................. 39
4.2 User Interface ......................................................... 47
4.1 Development Process

In order to do this project, a few steps were required to reach the best solution. That involved research, planning, testing, failing and perseverance. This were the fundamentals of the success of this project. Such elements can be divided in five different parts:

- Technology Research and Related Works
- Requirements Gathering
- Design of the Architecture
- Implementation Process
- Testing and Functional Validation

4.1.1 Technology Research and Related Works

This project consists in multiple elements with different technologies and/or techniques, there is the main part which is the vehicle, there is also the controllers (one computer controller and a mobile controller) and finally the road design. Before diving into the problems it is required to a priori learn and do research. That research was previously done in the Solution Architecture for the original project and later in the Road Design. What it is possible to say is that, in the beginning there were many alternatives and the best type of technology for each objective was chosen, however due to some inconveniences, the project ended up being adapted and restricted to the given technology. But it doesn’t mean that it is unique, in fact, there are many projects are similar to this one. There are line followers out there, but what there isn’t is a multi-function line follower with sensors. What makes this project unique is the communication between the vehicle and the controller, the ability to detect obstacles and interpret readings from the road (coloured traffic signs).

In terms of technology research, there is not too much to grab about. Given the fact that this project was modified in such way that it was restricted to specific hardware, that there is not a wide freedom of choice. Initially, the EV3 programmable brick came with the LEGO Mindstorm® Software and with it the vehicle can do many things including detecting a colour, an obstacle, receive a Bluetooth® connection and more as long the user programs it in its unique programming language. This language is considered a command box programming® where already defined functions are in blocks. The main difference of having a block language is its simplicity, even though that simplicity makes the language easier to understand and do simple tasks, it does not help a more complicated one. For that reason a different
software was needed.

For the technology used for the programmable brick there was many options in which almost all were viable but some would give tremendous amount of unnecessary work in comparison. So to choose the best option, it was needed to research more about them. There were three main technologies that are worth mentioning, RobotC, ev3dev and leJOS as all of these were the best option and a choice between them would not matter in terms of efficiency or difficulty. In short, this is what each one of them are:

4.1.1.A RobotC

ROBOTC is a cross-robotics-platform programming language for popular educational robotics systems. ROBOTC is the premiere robotics programming language for educational robotics and competitions. It is a C-Based Programming Language with an Easy-to-Use Development Environment. It supports many platforms such as Arduino, LEGO NXT®, TETRIX, PIC, VEX IQ and more. The main drawback of this software is that it is not free. It has a free 10-day trial version but in order to proceed using the software a license needs to be bought. For that reason, any following information regarding RobotC was disregarded.

4.1.1.B ev3dev

ev3dev is a Debian Linux-based operating system that runs on several LEGO® MINDSTORMS compatible platforms including the LEGO® MINDSTORMS EV3 and Raspberry Pi-powered BrickPi. ev3dev is not a firmware but an OS. The programmable brick works similarly to a dual-boot hardware, where it has LEGO® OS and the ev3dev. ev3dev runs from a microSD card and doesn’t ever touch the firmware installed on the EV3. Since ev3dev is built on Debian Linux there is no language restriction to what anyone can use as it can use C, C++, JAVA, Python3, Node.js and more. The main difference is that in this distribution, there is a low-level driver framework for sensor, motors, screen controlling and more.

4.1.1.C leJOS

leJOS is a firmware replacement to LEGO® Mindstorm EV3 and NXT. This OS includes a Java VM, which allows the EV3 or NXT to be programmed in Java language. This OS offers an object oriented language (Java), threads, arrays, recursion, synchronization, exceptions, java types, the java.lang, java.util and java.io classes as well a well-documented API that allows the access to any device connected with the programmable brick. This last offer is the main reason that leJOS is one of the best alternatives. Not only it allows the user to program the vehicle to do its bidding but also it does so using a well-known
After the research of these alternatives, one needs to be chosen. The most comfortable choice seems to be leJOS. Thanks to a well documented API and remote support, the leJOS is a really good environment for this project.

4.1.2 Requirements Gathering

There are two main things that this project requires which are: Code programming (Java) and Sensor knowledge. Other road elements are a must to further progress in this project as they will be the base of the road design, however any person that knows even the smallest thing about cars, then it knows, at least, the bare minimum of traffic signs and that it is enough.

To succeed in this project these two primary aspects are mandatory, especially the first one. If there is not a previous knowledge of programming (in this case in Java) then many hours are going to be spent in learning and not solving issues from the second characteristic. When it says "Sensor knowledge" it doesn’t mean to be an expert in sensors, whatsoever, but to have a minimum awareness of how sensors work is enough, otherwise much of the time spent will be wondering why the output isn't correct. In theory, everything works as it is supposed to, there are no issues whatsoever, it is assumed that the received values from the sensors are accurate and fast. But the moment you add a sensor and you wait for it to send or receive values, then you already left theory. The main obstacle were the learning of two particular sensors, the Colour Sensor and the Ultrasonic Sensor being the first one much harder than the latter. The Colour Sensor was one of the main problems and it was one of the filters that prevented the usage of more colours. A Colour Sensor is a RGB Sensor that detects the amount of Red, Green and Blue Colour reflected from the light emitted. The amount received will then be interpret by the sensor and let the controller know that a percentage of each colour (Red, Green or Blue) was received. First of all, in order to successfully interpret this values it is required the knowledge of the value of the colours in a RGB Model. Second, the hardware can be unpredictable and if not correctly calibrated then it is highly likely that most cases will lead to incorrect values. To understand the second point, it is a must to know the possible cases of input the sensor will receive. In the regular cases, the "happy" scenario, where everything works as expected and there are no errors in detection, the Colour Sensor either detects one colour, or another. While in the other cases when the Colour Sensor emits light over more than one colour at the same time, either between Blue and White or White and Red for example, the values are not as expected. In the "happy" scenario the vehicle only detects the colours mentioned in Road Design, but what happens when the vehicle is moving from one colour to another and the sample rate coincides in that moment where there's a part of a colour and another part of another colour, that is the real case
scenario and this was one of the most problematic obstacles in this project. The RGB Model is a light colour model and mixing colours from this model is very different than mixing paint. For instance, if you mix all colours in a RGB model the result will be the colour white, however if you mix all colours in paint, the result will be black. This means that normal mixtures that occur in every day life, don't apply to this case. Not only that, but mixing colours in RGB Model result in unexpected colours. This shouldn't be a problem unless the sum of some colours used result in another colour used. Taking into account the colours used, there are some issues when the following colours are together:

- Red and Green
- Red and White
- Black and White

These three combinations are not good due to the fact mentioned earlier, that they may give out a colour that is not there. In terms of RGB Values, the standard colours (Red, Green and Blue) have they're own values more dominantly than the other two. Although it is noteworthy to mention that due to high brightness of either tape used or the colour itself the reflection from the Colours Red and Green also showed that the other values are higher than expected (but never near the highest value).

The first combination, Red and Green, in terms of addition is a third colour with high red value and high green value while maintaining a low blue value. The colour that follows that description is Yellow. This means that sometimes if there's a green colour near a red one and the vehicle detects the middle in between, then the result will probably be yellow. Due to that problem it was made sure that those two colours never touch each other or were near.

The second combination, Red and White, is similar to the first because of the display of the RGB Sensor. The White Colour is a colour with high red, green and blue values and in this case there are times that the vehicle detects tries to interpret both colours at the same time. If everything went as expected even after knowing that detection in between colours may happen, then the result of these two colours would be a higher red value and the same values for green and blue as the white. But these colours aren’t one hundred percent accurate nor is the Colour Sensor. There's a time when the sensor is attempting to detect the colours and the red and blue light emitted is over the Red colour and the green light is over the White colour, giving the red and blue values of Colour Red and the green value of the Colour White. This means that the resulted colour is a colour with high red and green values whilst low blue values. This is the same colour in the first combination, the Yellow colour. This is a major problem because if the vehicle should slow down (Red sign) and instead thinks it is on a crossroad then it is more than likely that it would be offroad.

The third combination, Black and White, well this was one of the main reasons that made the removal of the Black colour. In the beginning the Black colour was meant to be the lane instead of blue however
because the vehicle is almost always driving over the limit between the lane and the off-road (Black and white) the resulting colour of this two elements together was always Blue. So, in order to prevent any future mistakes, the lane colour changed from Black to Blue.

To continue and resolve the project, not only it was required to know these combinations and more but also to understand how to overcome this obstacles, taking into account what the hardware output can give instead of assuming that it will always give what it is expected.

4.1.3 Design of the Architecture

This project consists on two vehicles either on the same Bluetooth® network or in separated networks that autonomously drive in a given road. These vehicles may be controlled by a controller device such as a computer or a mobile phone (Android).

In terms of software, the vehicles have the same programming information and it is recommended that before attempting to drive they calibrate their Colour Sensor. Afterwards, the vehicle may enter in control mode or autonomous mode where they either wait for any sign of the controller or start driving, in both cases it enters in a infinite loop which will only leave if told to do so. In control mode, they keep receiving information from the controller and replying with current updates of the current status (such as speed, next turn on the crossroad and last colour detected). While on autonomous mode, the vehicle drives over the blue lane and everytime it detects a new colour it acts accordingly.

4.1.4 Implementation Process

After the design, the only logical thing to do is to start from the vehicle's hardware. This because the vehicle is the main element of this project and without it nothing else really matters. What is a road without a car, what is a controller without the object to control?

As such, this project can be divided in four different but major parts: Assembly, Road Design, Car software and Controller Software. The first two are hardware oriented whilst the latter two are software oriented. After the vehicle's been built (assembly), the Road Design and the Car software went hand-to-hand as one complemented the other and vice-versa. Also, by doing so, possible testing was done in the meantime as the project advanced. After the completion of the road and the autonomous part of the vehicle's software, the controller software started and in the same way that the Road Design and Car software went hand-to-hand, now that same software and the Controller Software were a pair. As one progressed so did the other to match the advances, so that it can both be in the same stage at the same time.

To understand better the implementation process, the following parts are explained in depth:
4.1.4.A Assembly

The LEGO Mindstorms® Kit did not come with an already assembled vehicle, obviously. As such, it was needed to build the car from scratch, taking into account not only all the elements previously mentioned, like the large motors, the Ultrasonic Sensor, the Colour Sensor and the programmable Brick and also two wheels, a metal ball to help sustain the vehicle's balance and the LEGO® pieces that will make the vehicle's chassis supporting every hardware together.

To help with this process, there was an instructions manual that came along with the kit, which allowed to easily assemble the two large motors together as well the metal ball (that are used with another pre-made robot). Following more the instructions book, it was also fairly easy to assemble some components with the Ultrasonic Sensor, so that it was ready for a "plug and play" approach. However, although it was assemble-ready, the Ultrasonic Sensor could not be placed due to the fact that if it did, the Colour Sensor would not have any space left, unless it was placed on front of the vehicle (which removed the purpose of the Ultrasonic Sensor in front). For that reason not only the Colour Sensor was implemented first, but also an increased length of the vehicle to fit the Colour Sensor in the middle. This Sensor is facing downward from approximately 2mm above ground (this distance is optimal for an accurate approach of the colour read). The Ultrasonic sensor was "plugged" in front of the vehicle after the Colour Sensor, lasting the last piece of hardware: The LEGO Mindstorms® EV3 programmable brick. This piece of hardware is the core, the brain and the nervous system of the vehicle. For it to reach every piece of hardware it would ideally be in the middle. However its size wouldn’t fit right in the middle so a new platform was built for the brick to be on top of the vehicle, like a ceiling. By doing so, the brick could easily reach every piece of hardware and as well be in a user-friendly position. An average real size vehicle has between 4.5m and 4.8m length, as the Lego car has 0.23m which represents approximately 20 times less. As the brick is 11cm, it would represent a 2.2m box on top of a real life vehicle to control that said vehicle.

4.1.5 Road Design

At this time it should be already clear that the Road Design is one of the most important aspects of this project. Not only it is the vehicle’s "home", but it is also the testing zone where every scenario is tested.

Some person may think that: "Well, a road is just a road. Just draw a line and it's done." It is not that simple, there are many variables taking into account. Originally, the road was supposed to be with two lanes, however the vehicle no longer will have two infrared sensors to detect them but it will have a single Colour Sensor in the center, for that reason those two lanes wouldn’t be practical and the solution would be a Line Follower vehicle. One Lane, one Line. The lane needs two requirements: A colour different
than White and width longer than the Colour Sensor emitting light diametre ( <1.5 cm) at two millimetres
distance. Both problems were solved with the use of electrical tape. This is a coloured duct tape with
an average of 2 centimetres width. At this point, the road was nearly complete, it only lacked two things,
“traffic signs” and a circuit.

For the first, the choices were limited due to the number of colours detected by the Colour Sensor.
As previously mentioned in Road Design, only three colours remained for this (White was the off-road
while the Blue is the Lane). The Green and Red colours mean safe and danger respectively, similar
to a Traffic Light however the Red doesn’t mean to stop but to means to slow down due to danger
(usually a tight curve). The last colour, Yellow was used to notify the vehicle of a crossroad ahead. It
was possible to bring other colours such as Black and or Brown (as LEGO® Colour Sensor detailed
information mentions) however, due to testing it was concluded that these two colours were not a good
choice because the Black provided too much reflection the hardware most of the times mistakenly took
as Blue and the Brown due to it’s similarity to Red was also not used. To represent the traffic signs,
small pieces of coloured tapes were place along side the road in certain points, the same way a real
traffic sign would be, for instance before a curve to notify the driver of a tight curve or before a crossroad
to tell the driver that there’s a crossroad ahead.

Even though this seems a good environment, there is a problem reading the road lines and traffic signs
and that is due to the hardware limitation. There were and still are some incorrect reads by the Colour
Sensor, that the vehicle might interpret as incorrect, such as the Red colour as Yellow. This happens
due to sensor mixing the Red and White colour in the same radius. This Colour Sensor works as
RGB and the colour Red has high red, low green and low blue from the RGB model, while the colour
White has high red, green and blue. So when the hardware detects and interprets the received RGB
information of a mixed Red and White he detects the red from the Red tape, the green from the White
and doesn’t detect the blue because Red doesn’t have blue, giving a Yellow colour (high red, high green
and low blue). To solve this problem, a wider range of possible values of each Colour were given (without
overlapping) to the vehicle, so that when he detects a colour that may not be exactly red but it is mostly
red, he detects it as Red, the same with other colours. That’s is the main reason that the Brown colour
was not used.

On the second item, the circuit. In order to design a proper circuit, some requirements need to be met.
First, one design fits all. In this case, it was needed a design that could fit every traffic sign mentioned
above, as well as have enough space for the vehicle to ride it without colliding with another vehicle on
different roads. Second, how tight can a vehicle curve? In this case, there were two things to take into
account: What’s the curve radius limit for the vehicle to be able to turn? And, because there is only one
sign that notifies the vehicle to slow down, one speed must match all types of tight curves, some tighter
than others. The first question can be answered with testing. Technically, the vehicle can turn in its place
by moving the one wheel forward and the other wheel backwards, making it spin without moving forward or backward. But this is not desired as the vehicle would not move if it kept changing direction, it would be stuck in a loop where constantly tries to fix it's positioning. So, to reach the solution of this problem, a top-down approach was made. At this point, it was already proven that the vehicle could turn regular and easy turns, so a matter of testing was necessary in order to know the minimum radius possible. First a curve with 8 centimetres radius, then 7, then 6 and so on... It was proven that, with the proper code the vehicle could do any curve if specified. What it could not was to both do a really small radius curve and a curve with high radius (over 6 centimetres radius). To achieve a perfect balance of coding in which it can do all types of curves given a large interval, a limit was reached and it was at 4 centimetre radius.

4.1.6 Car Software

After the software decision, it's time to work on the perfect solution. First of all, we can split the coding in two sections: setup and loop. In setup, the code is run before the loop and it is only run once on every start. In this section the variable initialization is done as well as the hardware discovery, to ensure that the motors and sensors are connected. After that, in the same section, a menu asking which mode the user wants to work on. After that an attempt on connecting with the controller device is made, depending on the mode selected.

On the second part, the loop, the first thing that the vehicle does in the loop is detecting the Bluetooth mode. If it is in the full control mode, then it enters in another loop that does not leave until stated otherwise. Else if it is in another two modes he only get that one piece of information required (which is to let the vehicle know what’s the next turn or to change mode) and continues in the loop. After this, the vehicle continues to check if he is in a crossroad or not, if affirmative then there’s special condition to proceed. Following that, the vehicle functions as it should, detecting a colour with the Colour Sensor, detecting any obstacle with the Ultrasonic Sensor and if everything checks out, it moves accordingly with the colour and the obstacle.

For this code to work it was important to know what should come first, the bluetooth part or the colour detecting part. And following the logic the best solution was the one presented. Now, this doesn't explain every bit of line of coding, however it helps to understand the major parts of the code.

4.1.7 Controller Software

For the controller software there were many alternatives which with their base programming language. The only requirements there were for the controller software were that it could control the vehicle through wireless technology and it has an easy to use interface.

There are currently two controllers for the vehicle, a computer and a Mobile phone, both communicate
through Bluetooth®. For the computer, Java was the language of choice and Android was the other one. The reason behind Java was the language of choice was not only because of the compatibility it had with the leJOS but also because it was able to replicate some code into the Android controller. In the computer, the main obstacle was the keyboard control of the vehicle. By using regular IO library, the program can not detect holding and releasing keys. This is a major problem because if a user presses the button for the vehicle to move forward and then releases it, the vehicle should stop moving. This obstacle was overcome with the help of Java’s Abstract Window Toolkit (awt) library, which allows different events to occur when a user presses, holds and release a keyboard key. This cannot be done using regular IO library. Also, with this library, a window giving information of the vehicle is shown so that the user can see the details of it. Think that this is more like a car game and you see the dashboard in a screen. For the second controller, the Android mobile phone, the user first detects any vehicle on the Bluetooth® network and may attempt to connect to it. The main objective is to give the user a friendly environment that not only allows an easy control over the car but also the ability to simply read the vehicle’s data on screen.

### 4.2 User Interface

Regarding the user interface, the user has three ways to monitorize the events happening in this project, either by watching the controller software (computer or android) and the vehicle itself. Note the fact that either option of controlling the vehicle is optional as the vehicle itself does not need a human to control it. However if for some reason the user wants to override the system he can do it two ways.

#### 4.2.1 Controller Software - Computer

The user can control the vehicle’s through a computer java application. It is a simple menu with no pictures and with enough information for the user to safely interact with the car through the application. Initially the program will ask the user what mode in what the vehicle will be controlled in Subfigure 4.1(a) and after choosing the program will search for any vehicle in the current Bluetooth PAN Network Subfigure 4.1(b). After choosing the vehicle an attempt of connection will be made by the controller and it will wait for either the vehicle’s to be in the menu before driving or if it is already driving as it can be seen in Subfigure 4.2(a). Finally after the connection, it can be seen in the screen the instructions to properly control the vehicle as well the information presented by the vehicle, for instance the vehicle’s speed, next turn and control mode as seen in Subfigure 4.2(b).
Figure 4.1: Remote Control - Connecting.

(a) Remote control - EV3 sucessfully connected.
(b) Remote Control - Control Menu.

Figure 4.2: Remote Control - Connected.
4.2.2 Controller Software - Android

The android software is more simplistic in comparison with the computer version. It only has two screens, choosing the EV3 and controlling the ev3. For the vehicle to be recognized first the mobile must not be connected to any other network except for the PAN. That is due to the fact that by being connected to another network such as 192.x.x.x/24, the default gateway will be through the 192.x.x.x network instead of the 10.x.x.x that is the EV3. In the first screen, Subfigure 4 4.3(a) and after the vehicle’s chose its controlling mode, the user can attempt to control the vehicle by pressing it in the screen. This will lead to the control screen, Subfigure 4 4.3(b). This application can be divided in two parts, the interactive part and the informative part. For the first one, the user can press buttons that allow him to control the vehicle such as "Speed+" to increase speed as well as "Speed-" to decrease. Three radio buttons that let the user know which mode the vehicle’s in and four more buttons intuitively placed in the screen that change the action accordingly to the vehicle’s mode, for example: If the vehicle’s in the Full manual mode, the buttons spell out "Move x" where x stands for the direction (Forward, Left, Right and Backward), while if the vehicle is in the Semi-Manual Mode (or Semi-Auto Mode) the buttons spell out "Next Turn x", where x stands for (Forward, Left, Right and Random), finally in the last mode (Full-Auto Mode) the buttons are greyed out and cannot be pressed. In fact only the Radio Buttons are able to be pressed in this mode. Still in this mobile Controller, the user can see the vehicle’s data such as speed and what’s the next turn.

4.2.3 Car Software

As the user loads up the vehicle’s program for it to run, a message appears in the vehicle’s LCD (liquid crystal display) telling the user to choose which bluetooth mode is the vehicle’s running. After that choice, the program shows the user another menu where one of five choices may be picked:

- Enter: Loop
- Left: Debug
- Right: Manual-Calibration
- Down: Auto-Calibration
- Escape: Exit

The first option can be chosen by pressing the "Enter" button and it leads to running the program as it would normally do while the last option can be chosen by pressing the "Escape" button and as the name indicates leaves the program. In the other three options, things are not as simple and each one these do something different. The user by pressing the Left button will go to the "Left: Debug"
Figure 4.3: Android control - both Screens.
option, this option will detect debug the Colour Sensor and present the screen the RGB colours of the current detected colour. If the colour was not found then a message saying "Colour not detected" with the respective RGB values appear in the screen. The user can press any key to proceed and re-read more values except for the "UP" key which will make the program return to the previous menu. The Right button leads to "Right: Manual-Calibration" which will let the user into the manual calibration mode where he/she can choose which specific colour to calibrate. The last button, Down button will lead to the last option "Down: Auto-Calibration" will lead the user into calibrating every colour by placing the vehicle on top of the said colour and press ENTER. This will store the RGB values of that colour which will be used later on.
To successfully prove that this project works it is necessary to successfully test. The vehicle, in the end, should be able to drive autonomously in a small model road without crashing or leaving its path. It should also be able to turn safely to either side in the crossroad, slow down on tight curves, speed up on straight roads and completely stop before an obstacle.

This project will have multiple small phases so that not only it is easier to monitor as well encourages the progress. The following timeline will be as such:

5.1 Hardware

5.1.1 Phase 1: Assembly

In the beginning, the assembly is required for the project to work. The first things that should be done is a planning of the assembly so that, in the future, the other elements will not raise any issues with cables and positioning. Next, the following elements should be the first to be assembled, the two motors, the vehicle chassis, the Ultrasonic Sensor, the Colour Sensor and lastly the programmable brick. These elements are essential for the project and after the assembly and connection of all motors and sensors to the programmable brick, this one can easily test them to ensure that they are properly connected and working.

5.1.2 Phase 2: Moving vehicle - Onward

After the Assembly phase, the objective of this phase is to make the vehicle moving forward, without turning and without taking into consideration any device other than the 2 DC motors. For that, the research of a proper API needs to be done as well as the coding of the vehicle to move forward. In this part the beginning of the skeleton code may be done in advance but it is not required for the completion of this phase.

5.1.3 Phase 3: Moving vehicle - Turning

As the name mentions, in this phase, the vehicle will begin to turn disregarding any other devices. This phase is similar to the previous one however, to fully complete this phase the following objectives should be met: The vehicle can turn either slightly left or slightly right, hard left or hard right and a completely 90 degrees turn to either sides. What this means is that, in the end the vehicle should be able to correct
its course when a lane is detected, if the current speed is high (slight turns), low (hard curves) or a crossroad (90 degrees turn).

5.1.4 Phase 4: Implementation of the Ultrasonic Sensor

The ultrasonic sensor is implemented at this point and able to detect a wall or any other object that has a surface large enough for detection, for example another vehicle. The vehicle is finally able to safely drive in a straight line while not bumping into anything that may cross its path.

5.1.5 Phase 5: Implementation of the Colour Sensor

Here lies one of the hardest parts of the project, the Colour Sensor. Initially, the vehicle should detect all seven colours but only two modes should be active either it is white or non-white. This means that the vehicle now can be placed above a blue line or any other colour (except for white) and follow it. This is what makes the vehicle a line follower. In the end of this point, the vehicle not only will follow a line of the respective colour (which is blue) but also slow down on the red colour, speed up on the green colour and act as if it was on a crossroad on the yellow colour. This part was by far one of the hardest parts (if not the hardest) to fully and correctly implement in this project. This is because of the constant testing and arbitrary values that the Colour Sensor gave as output that did not make sense and it wasn’t easy to replicate the error in order to understand the issue. The problem occur “randomly” as the factors to replay it were not easy and sometimes it required laps and laps for it to happen again. This was a major issue due to the fact that an autonomous vehicle, that was supposed to be safe, could sometimes be an accident in an instant. This was not good, and it needed to be fixed immediately. After many testing there was a breakthrough which lead to the fact of calibration prior of running the vehicle. In the end, the vehicle could detect any colour (previously programmed and calibrated to) in order to differentiate the road and signs from off-road.

5.1.6 Phase 6: Building the road

After having all the devices connected to the vehicle and all the sensors up and running, the vehicle now needs to do a test drive in the final road. However it is not yet build, this step should fix that. In this phase the time should be taken to ensure the construction of the road. After the road has been built any vehicle should be able to ride it without any problems, even with the other vehicles on the road. In this phase some elements from phase 3 came back. This is, the testing of how tight can a curve be in order
for the vehicle to maintain on the road. And, as mentioned earlier, that value was 4. Knowing this value a safe road could be built. After this point, only the communications.

5.2 Communication

5.2.1 Phase 1: Implementation of Bluetooth

At this time starts the beginning of the Bluetooth. Basically the Bluetooth is used for vehicles to communicate with the controller. So in this phase the Bluetooth is correctly working with one vehicle on the road and the other one connected to the computer reading the messages that the first one is sending while driving. To function, the controller must be previously connected to the Bluetooth PAN access point in order to detect the vehicles. Only then he attempts to establish a connection with one device and sends information via sockets to the vehicle in the same network. The information sent to the vehicle consists in specific codes where as each type of information is coded by the first number sent of four numbers. If the first number is the number 1 then the following three digits mean the current speed of the vehicle (up to 250). If the first number is the number 2, then the vehicle will know that the following digits would mean left, forward or right (000, 001 and 002 respectively). Other potentialities are done such as Next Direction (3), Set Bluetooth Mode (4), Update (7) and Shutdown (9). In the Update case, the vehicle will know that it needs to send its current speed, next direction and previous detected traffic sign to the controller (in that order).

5.2.2 Phase 2: Implementation of computer App

Before the start of the application it was possible since the beginning to connect to the programmable brick from the computer via Bluetooth and ping to ensure that both Bluetooth’s worked. After this, the beginning of small tasks at first, such as send a message to test the connectivity, then others more complex that allow the controller to take control of the motors to move the vehicle. Finally it was possible for the computer to fully control the vehicle and receiving the data from it and print it out to the user.

5.2.3 Phase 3: Android communication

One of the tasks was the addition of a vehicle controller via Bluetooth. This can be using a computer, a tablet or a mobile phone. Because of the technology in hand and the elements available, the usage of a mobile phone with the an Android operating system will be chosen for this task. The application
on the mobile phone should have at least four driving modes: Free mode, Semi-auto pilot and full-auto pilot. In Free mode, the user “overrides” any traffic sign and sensor feedback, allowing him full control of the vehicle. In this mode the user can go off-road, cross continuous lines and go over the speed limit. In Semi-auto pilot the vehicle drives itself and the user controls where to turn, for instance the user sends the order “turn next right” or “turn next left”. Finally, the Full-auto pilot the user does not control the vehicle whatsoever and its course is chosen randomly. All the future tests and implementations are made by using a auto pilot mode.
Conclusion

Contents

6.1 Conclusions .................................................. 61
6.2 System Limitations and Future Work .......................... 61
6.1 Conclusions

Initially this project was to be made with Arduino-based vehicles, however things didn’t work out as supposed to and some changes needed to be made. Even with the changes it was still possible to prove that with fewer and more simplistic technology it was still possible to build an environment where vehicles could drive in a road autonomously without outside assistance, obtain road signage information from the road itself without human interference with the help of a Colour Sensor, prevent crashes thanks to the Ultrasonic Sensor and still, if wanted, managed to be controlled by a human through the control interfaces.

The Colour Sensor used in both vehicles allow the vehicle to maintain its course along the tracks and detect any traffic sign simulated in different colours. The Ultrasonic Sensor prevents the vehicle from crashing with the other vehicle or any obstacle. The Programmable Brick connects all sensor and motors, coordinating the information and deciding what to do with that same information. Each LEGO® Mindstorms EV3 had the same code and same type of hardware. Although in this case it wasn’t needed, it is proven with the controller communication that these vehicles can communicate with each other through Bluetooth PAN if wanted to, as the communication are made with sockets.

In the end, the results were not expected as in the initial objective with multiple sensor communication but it was still able to prove that with two different Sensors (Colour and Ultrasonic) it is possible to build an autonomous vehicle. The communication V2V with Bluetooth or any other technology is possible but without the correct Sensors, there is no advantage in doing so. Theoretically, with Bluetooth Low Energy 4.0 it is possible to implement an Observer profile and a Broadcaster Profile so that the first will detect any broadcaster name while the latter will transmit its current information. However without BLE this cannot be done and without knowing the vehicle position, the communication between vehicles is not as relevant.

6.2 System Limitations and Future Work

6.2.1 System Limitations

Even though there were some adaptations to the original project, there are still some limitations regarding the functionalities of this project. These limitations are the result of a combinations of multiple flaws in which can be resolved with more adequate hardware. The visible limitations are the following:

- Colour Sensor inaccurately detects colours;
- The vehicle is not aware of its position;
- If the vehicles are facing each other, no movement will occur;
- In random cases, one of the wheels increases speed for no apparent reason;
- Sensor not recognized.

These presented limitations can be fixed with more sensors or sensor alternatives and vehicle-to-vehicle communication. The problem presented in the first two limitations is that, sometimes the Colour Sensor does not correctly detects proper the colours properly even though it has been calibrated. This may happen by either faulty Colour Sensor, too much or too little ambient light that disrupts the reflection or the road’s reflection is too high so too little is absorbed. In terms of the second limitation, there is nothing unless the motor “history” that allows the vehicle to know its position in the road. And even then it would only work as long as there is a specific starting point and there were no crossroad or roundabouts in the circuit. The moment that the same vehicle used a crossroad or a roundabout then the vehicle’s position calculation would be wrong and inaccurate. To solve these two limitations it would be possible to revert to the original combination of both sensors, Infrared Sensor and RFID instead of the Colour Sensor, as it would do a better job at not only maintaining the vehicle’s course in the road but also it will allow the vehicle to understand the current traffic sign via RFID and read its information which has not only the type of sensor but also the position as it is located. By knowing the traffic sign’s position, the vehicle can easily calculated it’s current position and keep correcting the information every time it passes a traffic sign.

In the third limitation it initially was thought for the vehicles to when facing each other divert their course in order to pass one another but without the vehicle’s location and the only way to detect an obstacle is by using the Ultrasonic Sensor, the vehicle’s can’t tell the difference between a regular obstacle or another vehicle. This problem could be solved if each vehicle could know its position and transmit that information to the other vehicles using V2V communication. As of the moment that it was establish that each vehicle would not know its current location, the V2V communication was not needed as there was no information to be passed on between vehicles.

The next limitations happens when a vehicle is driving in the road and one of the wheels gets more frictional force than it should and attempts to compensate by correcting it’s position and accelerating that same wheel. This will most likely lead the vehicle to off-road and probably it would not be able to return to lane. To solve this issue, there are multiple things that could be done, either manage some better motors that have higher torque or change the type of wheel so that it could be disregard part of the frictional force. Both these alternatives are valid however they need to be tested first before reaching a conclusion. Finally, the last limitation consists in sometimes, an error appears prior launching mentioning that the sensor in a specific port (the port of the Colour Sensor) is not a Colour Sensor which makes the program to crash, a simple re-launch will fix the issue. This may be caused due to the library errors or incorrect cable issue but the second cause may seem unlikely due to the fact that in one moment it doesn’t work in the next it does.
6.2.2 Future Work

Autonomous vehicles are appearing more and more often as technology is evolving, and sensor communication is really important to enter this world. This model-size project is able to simulate a possible future road with future vehicles. But, there’s a lack of sensor that was not used due to compatibility issues or availability. Although it is possible to work on top of this project, learn from it and even develop it further, it is better to attempt to finish the Solution Architecture presented earlier that will provide a good head-start to a new project and also give the ability to learn sensor communication, understand how devices work with each other and still communicate in a network. The LEGO® hardware is limited and an Arduino is only limited as its power capacity. No only it is possible to use the Arduino architecture but also to improve it to use other devices, other sensors, other functionalities, numerous possibilities are at bay.

6.2.3 Learning purposes

As this theses was evolving and changing, it was possible to say that there is many things that a student can learn from it. Taking into account the sensors, communications and the solution of building a V2I environment, a normal student can use this theses as a project to learn more about these technologies, improve skills by actually working directly with a vehicle. This kind of experience is almost unique, and better than a simple simulation on a computer. Working with actually elements and seeing them function in a real case scenario is really different than a simple theory exercise. For those reasons a laboratory project was done and it can be seen in the annex.
Bibliography


[16] Mats Andersson. 2014. Use case possibilities with Bluetooth low energy in IoT applications. u-blox AG. - on 02/11/2017


[22] Jorge Peixoto Barbosa Neiva. 2012. Localização e Orientação “Indoor” com recurso à Tecnologia RFID. Faculdade de Engenharia da Universidade do Porto. - on 19/12/2017


[27] Ugaitz Amozarrain Peraz. 2015. Low Power WiFi: A study on power consumption for Internet of Things. Universitat Politècnica de Catalunya (UPC) - BarcelonaTech. - on 21/12/2017


[29] ZigBee Alliance. 2012. ZigBee Specification. - on 21/12/2017


[31] Artem Dementyev, Steve Hodges. Power Consumption Analysis of Bluetooth Low Energy, ZigBee and ANT Sensor Nodes in a Cyclic Sleep Scenario - University of Washington, Seattle, WA, USA


[36] Ahmad Amhani and Zaki Iqbal. AUTOMATED COLOR SENSOR SYSTEM USING LDR AND RGB LEDS CONTROLLED BY ARDUINO. Department of Engineering, RAK Research and Innovation CenterAmerican University of Ras Al Khaimah - on 12/04/2018
Laboratory Project
Xth Lab work: Lego EV3 autonomous vehicle – Optimization

Goal
The goal of this work is to achieve multiple tasks that require implementation and optimization of the LEGO EV3 autonomous vehicle.

Description
This work is based on a thesis work Autonomous Driving with V2I Communication and it requires the assembly of the vehicle with the following elements:

- Two Large Motors + Two wheels;
- EV3 Brick;
- Colour Sensor facing down (average 2mm from the ground);
- Ultrasonic Sensor in front of the vehicle;
- leJos OS on SD Card;

In this lab project, multiple tasks will be asked for the students to develop so that they can understand not only how the sensor work but how the communication between them and the EV3 brick.

The following excerpt was taken from the thesis:

"leJOS is a firmware replacement to LEGO® Mindstorm EV3 and NXT. This OS includes a Java VM, which allows the EV3 or NXT to be programmed in Java language. This OS offers an object oriented language (Java), threads, arrays, recursion, synchronization, exceptions, java types, the java.lang, java.util and java.io classes as well a well-documented API that allows the access to any device connected with the programmable brick. This last offer is the main reason that leJOS is one of the best alternatives. Not only it allows the user to program the vehicle to do its bidding but also it does so using a well-known programming language. Also, it has remote support between multiple devices (EV3-EV3, EV3-NXT, EV3-RCX, EV3-PC & EV3-Android). “

The students will create a line follower vehicle and find the optimal algorithm, so the vehicle will spend the most possible time in the line without leaving its course.

- The vehicle should follow a line;
- A time race between groups incentivizing the students to find the best line following algorithm;
- Avoid static obstacles;
• Colour Sensor calibration;
• Explore other available hardware that may help with this project.

LeJos uses java to program the vehicle’s Brick. You can look up the libraries that leJos has to offer in the following link: [http://www.lejos.org/ev3/docs/](http://www.lejos.org/ev3/docs/).

In the end the vehicle fully assembled should be similar to this:

![Vehicle Image]

Be sure to connect the sensor cables to sensor ports and the motor cables to the motor ports.
Recommendations

In order to fulfill your work with security and not damaging the hardware involved, remember to carry out the recommendations below. As you are working fill the boxes to be certain that you fulfill all security measures.

| Always work with the circuits disconnect from its power sources. |
| Call the teacher, or responsible for the laboratory, before you connect the circuits to its power sources. |
| Ensure that the cables that come from Motors are correctly in the motor ports. |
| Ensure that the cables that come from Sensors are correctly in the sensor ports. |

Make sure that all Lego pieces are correctly placed on one another and the vehicle can freely move (wheels aren't being braked by a Lego piece).
LeJos Programming

To put the system to work it is necessary to install the OS in the EV3 brick, if not already install. For that you can follow this guide: [Getting Started with leJOS EV3](#)

For the programming to work it is highly recommended to install Eclipse and the leJOS plugin in that Eclipse. This will allow the student to compile and run the code directly in the eclipse to the vehicle. For more guides you can check this website: [LeJos EV3 Wiki Home](#)

The vehicle will run a java program just like in the computer but with the libraries available in leJOS. It is highly recommended to create a Setup() and a Loop() method that represent the start of the vehicle and the looping method of the vehicle respectively.

Here's a code example of the vehicle used in the Thesis. You can learn more from it by reading the library used in leJOS.

```java
//Hardware
import lejos.hardware.Keys;
import lejos.hardware.Sound;
import lejos.hardware.Button;
import lejos.hardware.ev3.EV3;
import lejos.hardware.port.Port;
import lejos.hardware.lcd.TextLCD;
import lejos.hardware.BrickFinder;

//Sensors
import lejos.hardware.sensor.EV3ColorSensor;
import lejos.hardware.sensor.EV3UltrasonicSensor;
import lejos.robotics.Color;
import lejos.robotics.SampleProvider;
import lejos.hardware.motor.EV3LargeRegulatedMotor;

The names are self-explanatory.
EV3 ev3 = null; //ev3 brick
Port s3; //port position colourSensor = 3
Port s4; //port position ultrasonic = 4
Port mD; //motor left = D
Port mA; //motor right = A
//Colour Sensor
float[] colourSample;
SampleProvider colourProvider;
SampleProvider usProvider;
EV3ColorSensor colourSensor;
short colour;
//US Sensor
float[] usSample;
EV3UltrasonicSensor ultraSonic;
EV3LargeRegulatedMotor leftMotor;
EV3LargeRegulatedMotor rightMotor;
```
In the vehicle's constructor you'll need to initialize the previous variables:

```java
//bind EV3
ev3 = (EV3) BrickFinder.getDefault();
lcd = ev3.getTextLCD();
keys = ev3.getKeys();

s3 = ev3.getPort("S3");
s4 = ev3.getPort("S4");
mD = ev3.getPort("D");
mA = ev3.getPort("A");

//Get the Color + Ultrasonic sensor
colourSensor = new EV3ColorSensor(s3);
colourProvider = colourSensor.getRGBMode();
colourSample = new float[colourProvider.sampleSize()];
ultraSonic = new EV3UltrasonicSensor(s4);
ultraSonic.enable();

usProvider = ultraSonic.getDistanceMode();
usSample = new float[usProvider.sampleSize()];

//Get the Motors
leftMotor = new EV3LargeRegulatedMotor(mD);
rightMotor = new EV3LargeRegulatedMotor(mA);

Use the Provider.fetchSample() to retrieve the sensor values:

colourProvider.fetchSample(colourSample, 0);
usProvider.fetchSample(usSample, 0);

Samples return an array and depending on the current sensor mode values can be in any position of the array.
**Xth Lab work: Lego EV3 autonomous vehicle – Optimization**

<table>
<thead>
<tr>
<th>Group:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1:</td>
</tr>
<tr>
<td>Student 2:</td>
</tr>
</tbody>
</table>

**Results**

Fill the following fields or provide the corresponding printed listings.

1. Describe the vehicle’s implementation. Describe the changes in the program developed in lab 2 (Sensing the Real World) to port it to the new configuration with two controllers.
   a. What mode used in Colour Sensor and why?

   b. Explain the solution for the obstacle detection.
c. Explain the movement algorithm to maintain the vehicle in the line.

d. Describe any other implementation that seem more relevant (implementation of tasks).

2. Explain your solution of the loop() method:
3. Time attack race: Write out the times taken for the vehicle to complete a course and if any major change is done, explain it.

4. The colour sensor is not coherent, different hardware have different sensitization. For it to work it's required to calibrate it first. How do you achieve this?