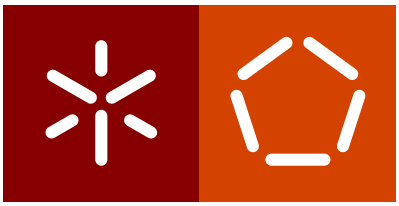


Universidade do Minho
Escola de Engenharia
Departamento de Engenharia Eletrónica
Laboratórios e Práticas Integradas 2

Integrator Project: Final Report

Radio Frequency Camera Assisted Rover (RFCAR)

Group 7

Nuno Rodrigues	A85207
Hugo Carvalho	A85156
Hugo Ferreira	A80665
João Faria	A85632
João de Carvalho	A83564
José Mendes	A85951
José Pires	A50178

Supervised by:
Professor Doutor Vitor Silva

July 2, 2020

Contents

List of Figures

List of Tables

List of Listings

Symbols

Symbol	Description	Unit
ω	angular velocity	rad/s
π	ratio of circumference of circle to its diameter	

1. Introduction

The present work, within the scope of the curricular unit of Laboratórios e Práticas Integradas II, consists in the project of the development of a product with strong digital basis, namely, a remote controlled vehicle used to assist exploration and maintenance in critical or unaccessible areas to human operators.

In this chapter are present the project's motivation and goals, the product concept, the project planning and the document organisation.

1.1. Motivation and Goals

The main goal of this project is to develop a remote controlled vehicle with the following characteristics:

1. Remotely operated: the vehicle must be remotely operated to enable its usage in critical or unaccessible areas to human operators;
2. Provide visual feedback to the user: to be a valuable asset in the exploration and maintenance domains, the vehicle must provide visual feedback to the user of its surroundings.
3. Safe: the vehicle must be safe to use and prevent its damage and of its surroundings
4. Robust: the vehicle must be able to sustain harsh environmental conditions and provide redundant mechanisms to avoid control loss.
5. Affordable: so it can be an economically viable product.

1.2. Product concept

The envisioned product consists of a remote controlled vehicle used to assist exploration and maintenance domains, hereby, denominated as Radio Frequency Camera Assisted Rover (RFCAR). To satisfy such requirements, the vehicle must contain a remotely operated camera that provides a live video feed to the user.

Additionally, the vehicle must include an odometric system that assists the driving and avoids unintentional collisions when remote control is compromised, e.g., when connection is lost. The vehicle provides means for exploration and conditions assessment in critical or unaccessible areas to human operators, such as fluid pipelines and other hazardous locations.

1.3. Planning

In Appendix A is illustrated the Gantt chart for the project (Fig. A.1), containing the tasks' descriptions. It should be noted that the project tasks of Analysis, Design, Implementation and Tests are performed in two distinct iterations as corresponding to the Waterfall project methodology.

Due to unpredictable circumstances, limiting the mobility of team staff and goods, the implementation stage will not be done at full extent, but rather at a simulation stage. Thus, to overcome these constraints, the project focus is shifted to the simulation stage, where an extensive framework as to built to model the system operation, test it, and providing valuable feedback for the dependents modules. As an example, the modules previously connected just by an RS232 link, must now include upstream a web module (TCP/IP) — the data is now effectively sent through the internet, and must be unpacked and delivery serially as expected if only the RS232 link was used.

The tasks are described as follows:

- Project Kick-off: in the project kick-off, the group is formed and the tutor is chosen. A brainstorming about conceivable devices takes place, whose viability is then assessed, resulting in the product concept definition (Milestone 0).

- State of the Art: in this stage, the working principle of the device is studied based on similar products and the system components and its characteristics are identified.

- Analysis: In the first stage — Analysis 1 — contains the analysis results of the state of the art. It should yield the specifications document, containing the requisites and restrictions to the project/product, on a quantifiable basis as required to initiate the design; for example, the vehicle's desired speed should be, at maximum, $2\ \mathrm{m/s}$. The second stage — Analysis 2 — contains the analysis of the first iteration of the development cycle.

- Design: it is done in two segments: modules design — where the modules are designed; integration design — where the interconnections between modules is designed. It can be subdivided into conceptual design and solution design.

  - In the conceptual design, several problem solutions are identified, quantifying its relevance for the project through a measuring scale, inserted into an evaluation matrix, for example, QFD.
  - In the solution design, the selected solution is developed. It must include the solution modelling, e.g.:
    * Control system: analytically and using simulation;
    * Transducer design: circuit design and simulation;
    * Power system: power supply, motors actuation and respective circuitry design and simulation;
    * Communications middleware: communication protocols evaluation and selection;
    * Software layers: for all required modules, and considering its interconnections, at distinct levels:
      · front end layer: user interface software, providing a easy and convenient way for the user to control and manage the system.
      · framework layer: software required to emulate/simulate and test the required system behaviour, providing seamless interfaces for the dependents modules
      · back end layer: software running behind the scenes, handling user commands received, system monitoring and control.

- Implementation: product implementation which is done by modular integration. In the first stage, the implementation is done in a prototyping environment — the assisting framework developed, yielding version alpha; in the second stage it must include the coding on the final target modules, yielding prototype beta.

- Tests: modular tests and integrated tests are performed. Tests are generally considered as those performed over any physical component or prototype. Here, it is used as a broader term, to reflect the tests conducted into the system and the several prototypes.

- Functional Verification/Validation: System verification may be performed to validate overall function, but not for quantifiable measurement, due to the latencies involved. Regarding validation, specially for an external agent, thus, it should be limited to user interface validation.

- Delivery: — project closure encompassing:
  1. Final prototype

2. Support documentation: how to replicate, instruction manual.
3. Final report
4. Public presentation

1.4. Report organisation

This report is organised as follows:

- In Chapter 2 lays out the theoretical foundations for project development, namely the project development methodologies and associated tools, and the communications technologies.
- In Chapter 3 are identified the key requirements and constraints the system being developed must meet from the end-user perspective (requirements) and, by defining well-established boundaries within the project resources (time, budget, technologies and know-how), the list of spefications is obtained.
- After defining the product specifications, the solutions space is explored in Chapter 4, providing the rationale for viable solutions and guiding the designer towards a best-compromise solution, yielding the preliminary design and the foreseen tests to the specifications.
- The preliminary design is further refined in Chapter 5 and decomposed into tractable blocks (subsystems) which can be designed independently and assigned to different design teams, allowing the transition to the implementation phase.
- Next, in Chapter 6, the design solution is implemented into the target platforms.
- Then, in Chapter 7, the implementation is tested at the subsystem level (unit testing) and system level (integrated testing), analysing and comparing the attained performance with the expected one.
- After product testing, in Chapter 8, the specifications must be verified and validated by an external agent. subsystem level (unit testing) and system level (integrated testing), analysing and comparing the attained performance with the expected one.
- Chapter 9 gives a summary of this report as well as prospect for future work.
- Lastly, the appendices (see Section 9) contain detailed information about project planning and development.

2. Theoretical foundations

In this chapter some background is provided for the main subjects. The fundamental technical concepts are presented as they proved its usefulness along the project, namely the project development methodologies and associated tools, and communications in detail.

2.1. Project methodologies

The methodologies used for the project development are briefly described next.

2.1.1. Waterfall

For the domain-specific design of software the waterfall methodology is used. The waterfall model (fig. 2.1) represents the first effort to conveniently tackle the increasing complexity in the software development process, being credited to Royce, in 1970, the first formal description of the model, even though he did not coin the term [1]. It envisions the optimal method as a linear sequence of phases, starting from requirement elicitation to system testing and product shipment [2] with the process flowing from the top to the bottom, like a cascading waterfall.

In general, the phase sequence is as follows: analysis, design, implementation, verification and maintenance.

1. Firstly, the project requirements are elicited, identifying the key requirements and constraints the system being developed must meet from the end-user perspective, captured in natural language in a product requirements document.

2. In the analysis phase, the developer should convert the application level knowledge, enlisted as requirements, to the solution domain knowledge resulting in analysis models, schema and business rules.

3. In the design phase, a thorough specification is written allowing the transition to the implementation phase, yielding the decomposition in subsystems and the software architecture of the system.

4. In the implementation stage, the system is developed, following the specification, resulting in the source code.

5. Next, after system assembly and integration, a verification phase occurs and system tests are performed, with the systematic discovery and debugging of defects.

6. Lastly, the system becomes a product and, after deployment, the maintenance phase start, during the product life time.

While this cycle occurs, several transitions between multiple phases might happen, since an incomplete specification or new knowledge about the system, might result in the need to rethink the document.

The advantages of the waterfall model are: it is simple and easy to understand and use and the phases do not overlap; they are completed sequentially. However, it presents some drawbacks namely: difficulty to tackle change and high complexity and the high amounts of risk and uncertainty. However, in the present work, due to its simplicity, the waterfall model proves its usefulness and will be used along the project.

As a reference in the sequence of phases and the expected outcomes from each one, it will be used the chain of development activities and their products depicted in fig. 2.2 (withdrawn from [3]).

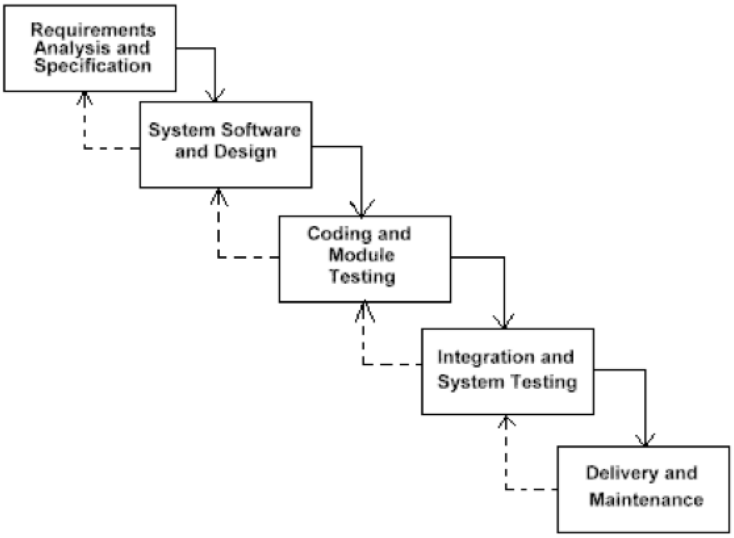


Figure 2.1.: Waterfall model diagram

2.1.2. Unified Modeling Language (UML)

To aid the software development process, a notation is required, to articulate complex ideas succinctly and precisely. The notation chosen was the Unified Modeling Language (UML), as it provides a spectrum of notations for representing different aspects of a system and has been accepted as a standard notation in the software industry [3].

The goal of UML is to provide a standard notation that can be used by all object- oriented methods and to select and integrate the best elements of precursor software notations, namely Object-Modeling Technique (OMT), Booch, and Object Oriented Software Engineering (OOSE) [3]. It provides constructs for a broad range of systems and activities (e.g., distributed systems, analysis, system design, deployment). System development focuses on three different models of the system (fig. 2.2) [3]:

1. **The functional model**: represented in UML with use case diagrams, describes the functionality of the system from the user's point of view.

2. **The object model**: represented in UML with class diagrams, describes the structure of the system in terms of objects, attributes, associations, and operations.

3. **The dynamic model**: represented in UML with interaction diagrams, state-machine diagrams, and activity diagrams, describes the internal behaviour of the system.

2.2. Communications

The communications technologies and the associated tools used for the project development are briefly described next.

2.2.1. Bluetooth

Bluetooth is a ubiquituous radio-frequency technology (2.4 GHz) for wireless communication, generally at short distances. Bluetooth is managed by the Bluetooth Special Interest Group (SIG) and the current version of the standard is 5.0. Starting from version 4.0, also known as Bluetooth Low—Energy (BLE), power consumption was minimised, making it suitable for embedded and Internet of Things (IOT) applications. Bluetooth is a full protocol stack, illustrated in Fig. 2.3, comprising the controller, the host device and the applications interacting with the device. The Host Controller Interface (Internet Protocol) layer enables the host device to interface the controller, required for low-level operations such as asynchronous device

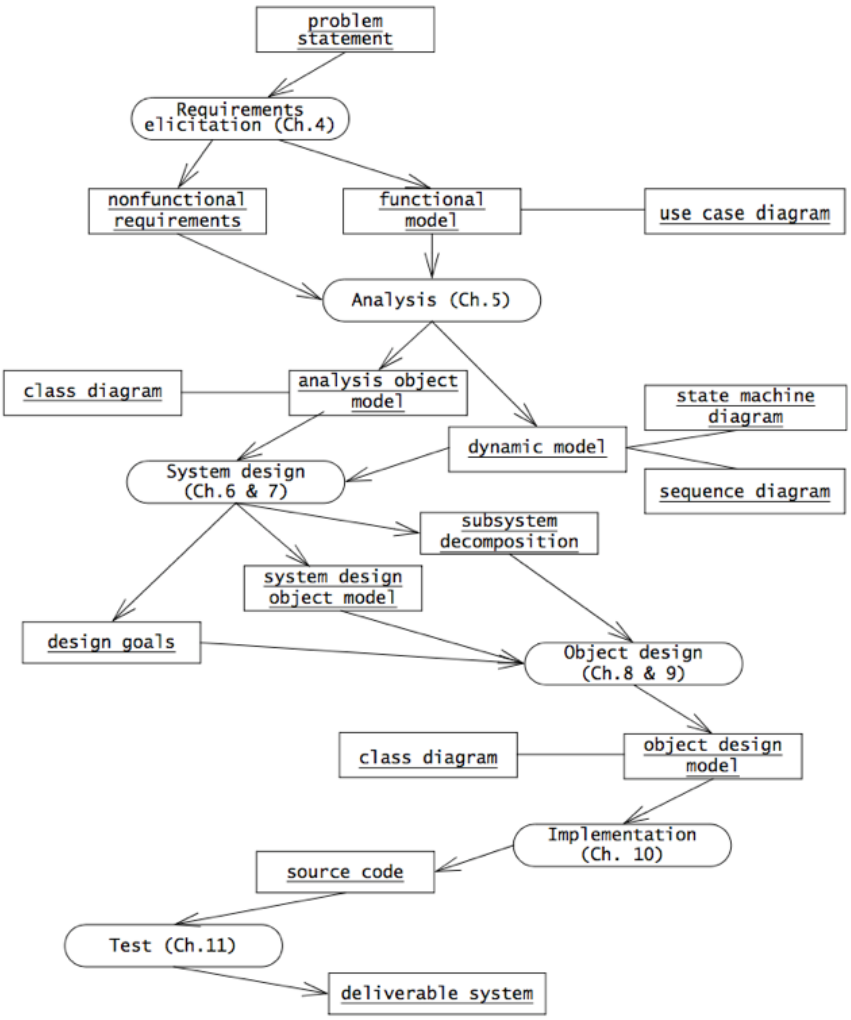


Figure 2.2.: An overview of the object-oriented software engineering development and their products. This diagram depicts only logical dependencies among work products (withdrawn from [3])

discovery or reading radio signal intensity. The host provides profiles to the external applications, easing the interaction betwen device and applications. Conceptually, Bluetooth is very similar to Transmission Control Protocol/Internet Protocol (TCP/IP) stack. Both are part of the network programming class and share the same principles of communication and data exchange between devices. The difference is that Bluetooth was designed for short distance communication, whereas the internet programming does not share this concern. This affects how two devices detect each other initially and how they establish the initial connection. From that moment on, the procedure is similar to the TCP/IP stack (see Fig. 2.4 and Fig. 2.5).

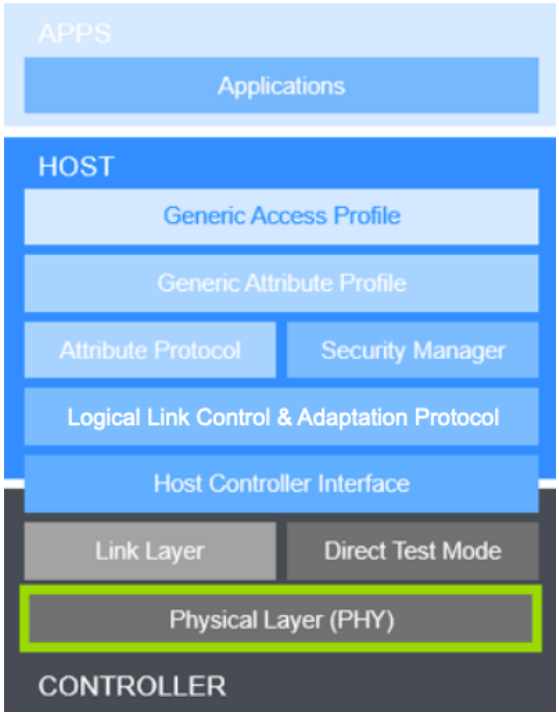


Figure 2.3.: Bluetooth 5.0 protocol stack

2.2.1.1. Establishing a connection

Establishing a connection depends if the device in analysis is trying to establish an outgoing or incoming connection. Basically, for the former case the device sends the first data packet to start the communication, and for the latter the device receives the first data packet. The devices that initiate outgoing connections must choose a target device and a transport protocol, before establishing the connection and exchange data. The devices that initiate incoming connections must select a transport protocol and then listen for incoming connections, before establishing the connection and exchange data [4].

Figures 2.4 and 2.5 illustrate this concept, comparing network, internet and bluetooth programming for outgoing and incoming connections, respectively. For outgoing connections, only the two first steps (choosing a target device, transport protocol and port number) are different; after the connection is established, the process is similar. For incoing connections the procedures are even more similar, with the major difference that port numbers are dynamically assigned for Bluetooth programming. The procedures for programming outgoing and incoming connections are presented next.

Outgoing connection programming procedure:

1. Choose a target device
2. Choose a transport protocol and port number

3. Establish a connection
4. Exchange data
5. Disconnect

Incoming connection programming procedure:

1. Choose a transport protocol and port number
2. Reserve local resources and go into listening mode
3. Wait and accept incoming connections
4. Exchange data
5. Disconnect

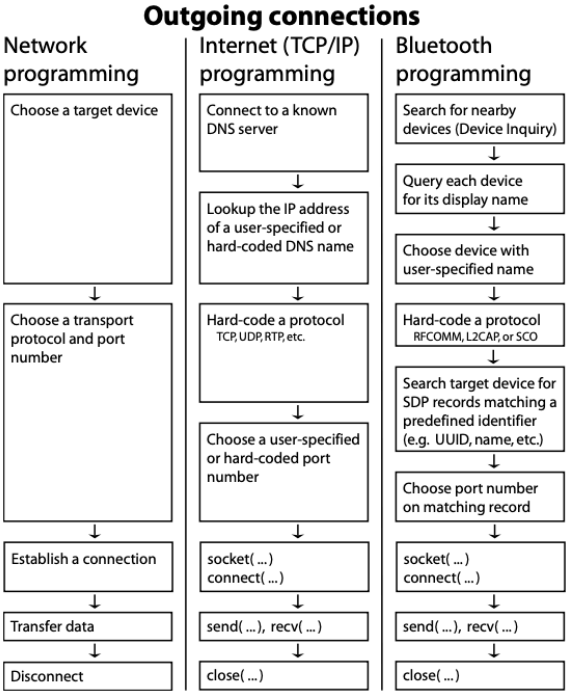


Figure 2.4.: Comparison of Network, Internet and Bluetooth programming for outgoing connections (withdrawn from [4])

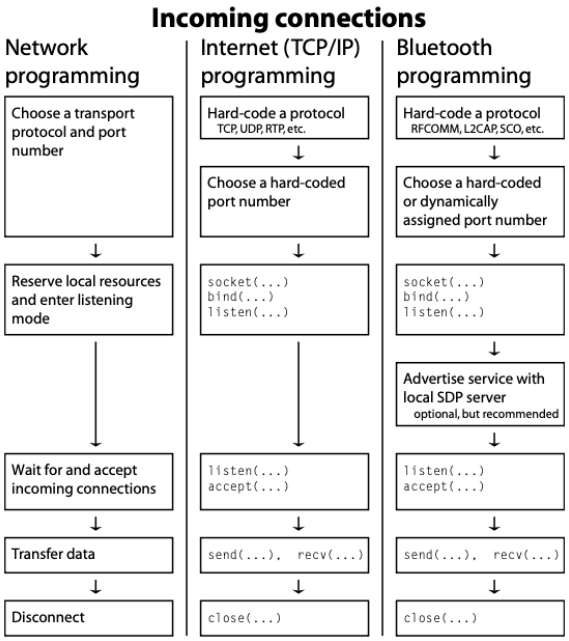


Figure 2.5.: Comparison of Network, Internet and Bluetooth programming for incoming connections (withdrawn from [4])

2.2.1.2. Selecting a target device

Every Bluetooth chip manufactured has a 48-bit unique address — BT address — identical to the Machine Address Code (MAC) for Ethernet protocol, acting as the basic addressing unit for Bluetooth programming.

The Bluetooth device must know the target device address to communicate. However, the client application does not need to know a priori the target address: the end-user provides a user-friendly name — device name — and the client translate this to the physical address when searching for nearby devices. The device name may not be unique, but the address must be.

In the device name lookup process, the device searches for the nearby devices by inquiring each device individually and compiling a list with their addresses, which is generally slow. A Bluetooth device does not announce its presence to other devices; it must start a device discovery process — Device Inquiry — to detect them. In the Device Inquiry process the device broadcast a discovery message and waits for replies. Each reply consists of the physical address of the device and of an integer identifier the class of the device (e.g., smartphone, headset, etc.). More detailed information, such as the device name, may be obtained by contacting each discovered device individually. Additionally, for privacy and power consumption reasons, the device may choose not to respond to device inquiries or connections attempts.

2.2.1.3. Transport protocols

Different applications have different needs, thus the need for different transport protocols. The two factors that distinguish these protocols are guarantees and semantics. The guarantees of a protocol state how hard it tries to deliver a packet sent by the application, yielding: robust protocols, like Transmission Control Protocol (TCP), which ensures that all sent packets are delivered or terminates the connection; best-effort protocols, like Transmission Control Protocol (UDP), which makes a reasonable attempt at delivering transmitted packets, but ignores its failure. The semantics of a protocol concerns if it distinguishes between datagrams beginning and end, and can be either packet-based (UDP) or streams-based (TCP).

Bluetooth contains four essential transport protocols, namely, by relevance order:

1. Radio Frequency Communications (RFCOMM): reliable and stream-based protocol, which emulates well serail ports. It allows only 30 open ports per device and it is the most frequently used protocol.

2. Logical Link Control and Adaptation Protocol (L2CAP): packet-based protocol which can be configured for several levels of reliability, imposing delivery order, unlike UDP. It encapsulates the RFCOMM connection.

3. Asynchronous Connection-Oriented Logical (ACL): It is not explicitly used, but it encapsulates L2CAP connections. Two Bluetooth devices may have, at most, one ACL connection between them, which is used to transport all RFCOMM and L2CAP traffic.

4. Synchronous Connection-Oriented (SCO): best-effort and packet-based protocol, which is used exclusively to transmit voice-quality audio at precisely 64 Kb/s.

The summary of Bluetooth transport protocols is illustrated in Fig. 2.6, where is visibly the connection encapsulation. Two Bluetooth devices may have, at most, one ACL and SCO connection between them, whereas the number of RFCOMM and L2CAP active connection is limited only by the number of available ports.

2.2.1.4. Port Numbers

A port is used to enable multiple applications on the same device to use the same transport protocol. Bluetooth uses a slightly different terminology: for L2CAP, ports are called Protocol Service Multiplexers (PSM) (range of 1–32767); for RFCOMM, ports are called channels (range of 1–30).

Some protocols have a set of reserved/well-known ports, intended for specific usage. L2CAP reserves ports 1–1023 for standardized usage (e.g., Service Device Protocol (SDP) uses port 1), whereas RFCOMM does not have reserved ports, given its small number.

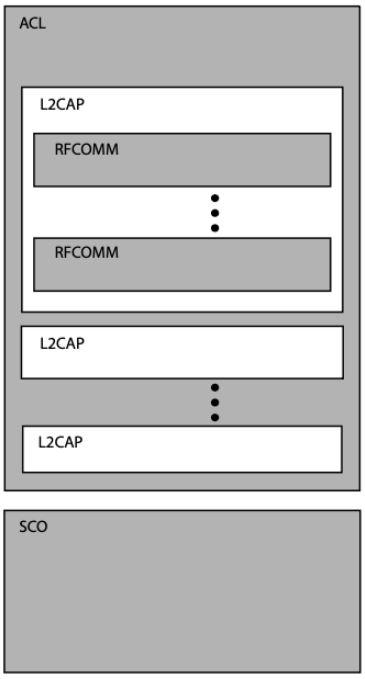


Figure 2.6.: Most relevant Bluetooth transport protocols (withdrawn from [4])

2.2.1.5. Service discovery

For a client application to initiate a outgoing connection it must know in which port the server application is listening. If the application is standard, then a reserved port is used. The traditional approach for Internet programming is hardcoding the same port number at both client and server: when the connection is established, the server listens on that part and the client connects to it. However, the static port definition is cumbersome and prevents two server applications from using the same port.

Bluetooth tries to solve this problem by introducing SDP, as illustrated in Fig. 2.7:

1. Each device keeps an SDP server listening on a well-known port.
2. When the server application is executed, it stores a description of itself and a port number in the SDP server on the local device.
3. Then, when a remote cliente application connects for the first time to the device, it provides a description of the service it is searching for, and the SDP server returns a list of all corresponding services with the respective associated port numbers.

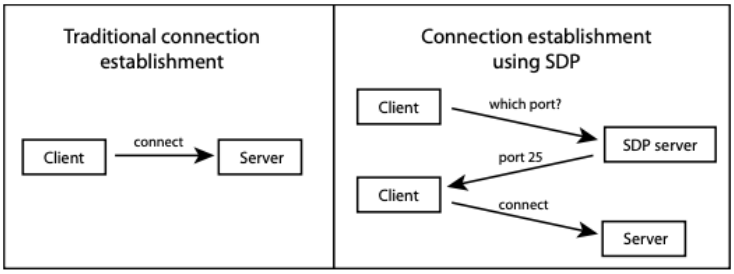


Figure 2.7.: Comparison between traditional connection establishment (internet) and using SDP (Bluetooth) (withdrawn from [4])

2.2.1.6. Host Controller Interface — HCI

The Internet Protocol defines how a host (e.g., a computer) interacts and communicates with the local Bluetooth adapter (the controller). All communications between these two agents are encapsulated in Internet Protocol packets, of four different kinds:

1. Command: sent from the host to the local adapter to control it, which can be used for starting a device discovery, connecting to a remote device, adjusting the connection parameters, amongst others.

2. Event: generated by the local adapter and sent to the host when an event of interest occurs, for example, device detected, connection established, etc.

3. ACL data: encapsulates data to send ou received from a remote Bluetooth device. In this sense, the Internet Protocol is a transport protocol for all transport protocols (ACL,L2CAP, and RFCOMM). When packets arrive to the local adapter, the Internet Protocol headers are removed and the pure ACL packet is transmitted through air.

4. SCO

2.2.1.7. Development Stacks for Bluetooth

A development stack refers to a collection of device drivers, development libraries and tools provided to enable software developers to create Bluetooth applications. In most operating systems there is a Bluetooth dominant stack, easing development, as there is a high level of incompatibility between Bluetooth development stacks. Since Bluetooth is a communications technology, the transport protocols supported by each stack are of particular interest. Occasionaly, wrappers around the developed libraries for a stack are created to provide a higher abstraction programming interface for Bluetooth, in languages like Python or Java. The most relevant development stacks and wrappers are depicted in Fig. 2.8.

PyBlueZ (GNU/Linux)	RFCOMM	L2CAP	SCO	HCI
PyBlueZ (Windows XP)	RFCOMM	L2CAP	SCO	HCI
BlueZ (GNU/Linux)	RFCOMM	L2CAP	SCO	HCI
Microsoft (Windows XP)	RFCOMM	L2CAP	SCO	HCI
Widcomm (Windows XP)	RFCOMM	L2CAP	SCO	HCI
Java – JSR82 (cross-platform)	RFCOMM	L2CAP	SCO	HCI
Series 60 Python (Symbian OS)	RFCOMM	L2CAP	SCO	HCI
Series 60 C++ (Symbian OS)	RFCOMM	L2CAP	SCO	HCI
OS X	RFCOMM	L2CAP	SCO	HCI

Figure 2.8.: Most relevant development Bluetooth stacks and wrappers and its supported protocols (withdrawn from [4])

The Bluetooth development stack selected was **BlueZ**, since its a powerful open source stack, included in all major GNU/Linux distributions and with extensive Application Programming Interface (API) s, supporting a extensive set of protocols which enables to fully explore the Bluetooth local resources. The RFCOMM, L2CAP, and SCO protocols are acessed using the standard sockets interface and the Internet Protocol is more conveniently used through the provided wrappers around the several Internet Protocol commands and events.

2.2.2. Wi-Fi

2.2.3. GPRS

2.2.4. Network programming — sockets

2.2.5. Client—server architecture

3. Requirements Elicitation and Specifications Definition

In this stage the project requirements are elicited, identifying the key requirements and constraints the system being developed must meet from the end-user perspective, captured in natural language in a product requirements document. The end-user perspective is generally abstract, thus requiring a methodic approach to obtain well-defined product requirements, i.e., product specifications. The product specifications are the result of a compromise between end-user requirements and its feasibility within the available project resources (time, budget, and technologies available). As the specifications are well-defined, they serve as design guidelines for the development team and can be tested later on to assess its feasibility and, ultimately, the quality of the product.

3.1. Foreseen specifications

In this section the foreseen product specifications of the system to be developed are provided. Such specifications were obtained through the intersection of customer, functional requirements and project restrictions.

3.1.1. Quality Function Deployment — QFD

The customer requirements are usually abstract and can collide with the functional requirements, compromising the fulfilment of the project. Thus, it raises the need of a methodology which converts abstract requirements into a series of concrete engineering specifications.

An efficient quality assessment methodology is the use of a Quality House (Quality Function Deployment (QFD)). In this method, the desired requirements are laid out as rows and the engineering specifications/restrictions as columns. In the intersections lies a symbol representing the strength (weak, moderate or strong — Figure 3.2) of the relationship requirement-specification. This symbol is one of the many tools that allow the quantification of relations existing between the customer requirements and engineering specifications.

For instance, the 'engine power' specification and the 'fast' requirement have a very strong correlation (9) since the power of the engine is directly responsible for the speed of the car.

Along with the requirements, the importance given to each is also specified, ranging from 1 (lowest importance) to 5 (highest importance) these, along with the number at each intersection, will be used to calculate the importance of each specification and thus assign priorities for the Design Team.

Lastly, the triangle shape (the 'roof' within the house metaphor) serves as another way of measuring relationships, this time between each specification: such is achieved by placing a symbol (ranging from very negative to very positive, see Figure 3.1) in the diagonal intersection of two specifications. I.e., the battery life will have a very negative correlation with the battery temperature, due to the fact that the increase of the temperature will cause a decrease in life time. As such a 'very negative' correlation was placed in the diagonal intersection betwixt 'Battery Life' and 'Battery Temperature'.

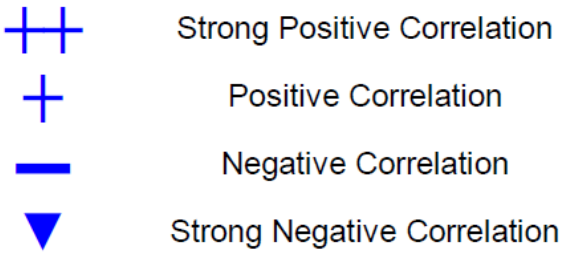


Figure 3.1.: Quality House — Specification Correlation Strength Symbols

Figure 3.3 shows the 'Quality House' for the RF CAR containing:

- **Customer Requirements**: Vehicle Integrity; Obstacle Avoidance; Reliable Feedback; Fast Response; Fast; Budget Friendly; Low Consumption; Small.
- **Functional Requirements or Restrictions**: Autonomy; Battery Temperature; Minimum Distance to Obstacle; Maximum Velocity; Motor Expectancy; Cost of Production; Motor Power; Ramp-Up Speed Time; Frame Rate; Camera Range; Resolution; Communication Range; Communication Speed; Dimensions; Mass.
- **Intersection Values** (referencing the strength of the requirement-specification correlation) — see Figure 3.2.
- **Analytical Data**, depicting, in a quantifiable manner, the aims of the project and the relevance of each entity:
  - Target or Limit Value: The metrics the design team will be based on, white spaces are left for either further discussion and refinement.

- Difficulty: Allows a subjective input to be added so that 'importance' can be changed to balance unforeseen circumstances.
- Importance and Relative Weight: The main conclusion for which the QFD was used, it assigns the priorities for the design team in an objective manner.

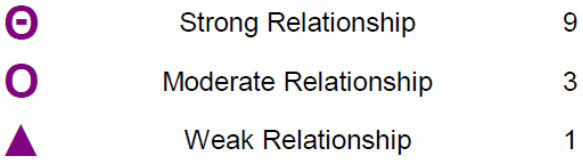
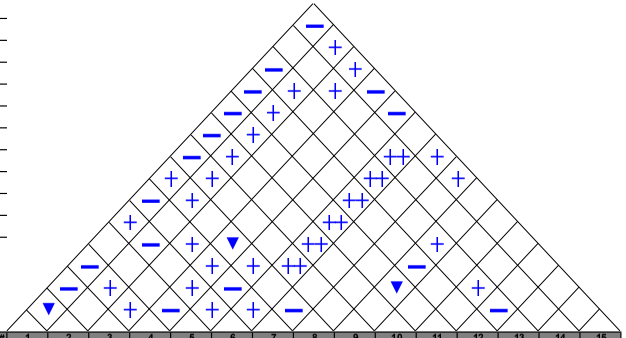


Figure 3.2.: Quality House – Relationship Strength Symbols

With the QFD, the prioritized ranks and specification targets were obtained and diffused within the Design Team with a straightforward guideline. For instance, the low cost requirement should be prioritized over all other specifications, followed by the maximum speed, Ramp-Up Speed Time and so on. On the other hand, the engine expectancy is of little to no consequence (note that the importance added up to a mere 3%), followed by the camera-related specifications. This could be regarded as a point of discussion, which should be prioritized? The functionality of the car or the the feedback provided by the camera?

With the last point in mind, the QFD has the advantage of promoting further discussion, simply by changing the importance of a requirement the priority ranking will change, ergo the priorities can be altered, easily and efficiently, if deemed appropriate.

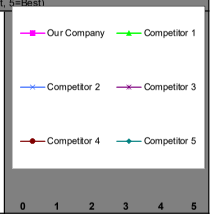
Title: RFCAR
Author: PI7
Date: 4/16/2020
Notes: João Faria A85632
João de Carvalho A83564
Nuno Rodrigues A85207
Hugo Carvalho A85156
José Mendes A85951
Hugo Ferreira A80665
José Pires A50178



Legend

⊙	Strong Relationship	9
O	Moderate Relationship	3
▲	Weak Relationship	1
+++	Strong Positive Correlation	
+	Positive Correlation	
−	Negative Correlation	
▼	Strong Negative Correlation	
▼	Objective Is To Minimize	
▲	Objective Is To Maximize	
X	Objective Is To Hit Target	

Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Demanded Quality (a.k.a. "Customer Requirements" or "Whats") \ Quality Characteristics (a.k.a. "Functional Requirements" or "Hows")	Battey Life (Autonomy)	Battery Temperature	Min Distance to Obstacle	Maximum Velocity	Motor Expectancy	Cost of Production	Motor Power	Ramp-Up Speed Time	Frame Rate	Camera Range	Resolution	Communication Range	Communication Speed	Dimensions	Mass	Our Company	Competitor 1	Competitor 2	Competitor 3	Competitor 4	Competitor 5
				Column #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Competitive Analysis** (0=Worst, 5=Best)					
				Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)	▲	▼	x	x	▲	▼	x	x	▲	▲	▲	▲	▲	x	x							
1	9	16.7	5.0	Vehicle Integrity	▲	⊙	⊙	⊙	▲	⊙	▲	⊙	▲	▲	▲	▲	▲	O	O	5	0	0	0	0	0
2	9	16.7	5.0	Obstacle Avoidant	▲	▲	⊙	⊙	▲	⊙	▲	▲	▲	▲	▲	▲	▲	⊙	O	5	0	0	0	0	0
3	9	13.3	4.0	Reliable Feedback	▲	▲	▲	▲	▲	▲	▲	▲	⊙	⊙	⊙	⊙	⊙	▲	▲	5	0	0	0	0	0
4	9	13.3	4.0	Fast Response	▲	▲	O	▲	▲	⊙	▲	⊙	▲	▲	▲	⊙	⊙	▲	▲	5	0	0	0	0	0
5	9	6.7	2.0	Fast	⊙	O	O	⊙	O	O	⊙	⊙	▲	▲	▲	▲	▲	▲	⊙	5	0	0	0	0	0
6	9	13.3	4.0	Budget Friendly	O	O	O	O	O	⊙	⊙	O	O	O	⊙	O	O	O	O	5	0	0	0	0	0
7	9	10.0	3.0	Low Consumption	⊙	O	▲	⊙	O	O	⊙	⊙	▲	▲	▲	O	O	▲	O	5	0	0	0	0	0
8	9	10.0	3.0	Small	▲	▲	O	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	⊙	▲	5	0	0	0	0	0
9																									
10																									
				Target or Limit Value	14400s/4h	35º-50ºC	5cm	1 m/s				1s	60 fps	20m	480p	50m		20 x 12 x 5 cm	0.5 Kg						
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)																					
				Max Relationship Value in Column	9	9	9	9	3	9	9	9	9	9	9	9	9	9	9						
				Weight / Importance	260.0	293.3	453.3	526.7	160.0	613.3	340.0	500.0	233.3	233.3	313.3	360.0	360.0	373.3	266.7						
				Relative Weight	4.9	5.5	8.6	10.0	3.0	11.6	6.4	9.5	4.4	4.4	5.9	6.8	6.8	7.1	5.0						



Powered by QFD Online (http://www.QFDOnline.com)

Figure 3.3.: Project Study – RFCar Quality House

3.1.2. Vehicle Autonomy

The vehicle is operated in wireless mode, thus, a portable power source must be included. The autonomy refers to the vehicle operating hours since the battery is fully charged and safely discharged and should be observed for the following scenarios:

- No load and vehicle operating at maximum speed;
- No load and vehicle operating at mean speed;
- Maximum load and vehicle operating at maximum speed;
- Maximum load and vehicle operating at mean speed.

3.1.3. Speed

The vehicle must be operated within a safe range of speed, while also not increasing excessively the power consumption. Thus, these speed boundaries should be tested in the absence of an external load and in the presence of the maximum load.

3.1.4. Safety

Vehicle self integrity protection is a requirement in product design, especially considering the vehicle is to be remotely operated. The safety in the operation can be analysed in two ways, and considers the preservation of people and goods. For the former, it is important to assure safe interaction as well as user operation — the vehicle may encounter several obstacles along its path, but it must not inflict any damage. For the latter, the vehicle under operating conditions must not inflict any damage to goods. Thus, in the presence of conflicting user commands violating the safety of people and goods, the local system should override them, taking corrective measures to prevent it. The same holds true if the communication between user and system is lost.

3.1.5. Image acquisition

The vehicle is equipped with a camera to assist in its navigation, thus, requiring it to be fed to the user's platform appropriately. To do so, several functionalities details need to be addressed efficiently. It was selected the most relevant three and these include the frame rate, the resolution and the image range.

3.1.5.1. Frame rate

Frame rate refers to the frequency at which independent still images appear on the screen. A better image motion is the result of a higher frame rate but the processing overhead increases as well, so a compromise must be achieved between the quality of the image and the increased processing overhead required. The minimum frame rate defined must be such that allows a clear view of the navigation.

3.1.5.2. Range

How far can the camera capture images without being distorted or unseen by the user. The range must be such that allows the user to see the obstacles when the car is heading to them and provide enough time to change the direction.

3.1.5.3. Resolution

The amount of detail that the camera can capture. It is measured in pixels. The quality of the acquired image is proportional to the number of pixels but a increased resolution requires a greater data transfer and processing overhead, thus, a compromise must be achieved. The minimum resolution must be such that provides the least amount of information required for the user.

3.1.6. Communication

For the implementation of the communication, several stages must be considered: Reliability, redundancy and communication range.

3.1.6.1. Reliability

A communication is reliable if it guarantees measures to deliver the data conveyed in the communication link. As reliability imposes these measures, it also increases memory footprint, which must be considered depending on the case. For the devised product, an user command must be acknowledged to be processed, otherwise, the user must be informed; on the other hand, loosing frames from the video feed is not so critical — user can still observe conveniently the eld of vision if the frame rate is within acceptable boundaries.

3.1.6.2. Redundancy

The communication protocols are not flawless and the car relies on them to be controlled. If the communication is lost, the car cannot be controlled. A possible solution for this issue is using redundancy in the communication protocols (e.g Wi-Fi and GPRS), so if one protocol fails, the car will still be controlled using the other.

3.1.6.3. Range

The communication protocols have a limited range of operation, and, as such, regarding the environment on which the car is used the range can be changed. The range established the maximum distance allowed between user and system for communication purposes.

3.1.7. Responsiveness

The movement of the car will be determined by the tilt movement of the smartphone. Sensibility refers to the responsiveness of the car on the minimum smartphone tilt movement. The sensibility must be in an range of values in which small unintentional movements will not be enough to change the state of the car and it does not take big smartphone tilts for the car to move.

3.1.8. Closed loop error

The speed, direction and safe distance to avoid collisions must be continuously monitored to ensure proper vehicle operation. The closed loop error must then be checked mainly in three situations as a response to an user command:

- speed: the user issued an command with a given mean speed, which should be compared with the steady-state mean speed of the vehicle.
- direction: the user issued an command with a given direction, which should be compared to the vehicle direction.
- safe distance to avoid collisions: the user issued an command with a given direction and speed which can cause it to crash. The local control must influence, to prevent collision, and the final distance to the obstacles must be assessed and compared to the defined one.

3.1.9. Summary

Table 3.1 lists the foreseen product specifications.

Table 3.1.: Specifications

	Values	Explanation
Autonomy	4 h	Time interval between battery fully charged and safely discharged
Speed Range	0.1 to 1 m/s	Speed at which the car can operate
Frame Rate	60 fps	Frequency at which independent still images appear on the screen
Camera Range	20 m	How far can the camera capture images without loosing resolution
Camera resolution	480p	Amount of detail that the camera can capture
Communication Range	50 m	Maximum distance between the car and the smarphone without losing connection
Speed Error	5 %	Maximum difference between desired and real speed
Direction Error	5%	Maximum difference between desired and real direction
Distance Error	5 %	Maximum difference between desired and real distance to the obstacle
Dimensions	20x12x5 cm	Dimensions of the car
Weight	0.5 kg	Weight of the car

4. Analysis

After defining the product specifications, it is possible to start exploring the solutions space within the project's scope, providing the rationale for viable solutions and guiding the designer towards a best-compromise solution. In this chapter are presented the preliminary design and the foreseen tests to the specifications.

4.1. Initial design

Following an analysis of the product's family tree (remote controlled cars), the state of the art and the QFD matrix in fig. 3.3, an initial design of the product itself can be produced (fig. 4.1). The selected approach was top-down, in the sense that the requirements and specifications were addressed and that resulted in a general diagram of the product concept. Some macro-level decisions were made in this stage to narrow the problem's solutions pool, as follows:

- The car itself should be battery-powered, as it is a free-moving object that is intended to work in environments where trailing cables could interfere with its regular movement.
- The device used to control the car should ideally be one already owned by the user, with an integrated screen (e.g. smartphone), as it would make it more affordable and have a more straightforward interface.
- The protocols for communication between the controlling device and the Rover should be chosen from within the pool of those readily available to smartphones (e.g. Wi-Fi, GPRS, Bluetooth) to keep the price of the overall product down and make it as practical as possible.
- The control and communication unit for the Rover should be divided into two modules: one which can measure and process sensor inputs and control the actuators in real-time, as well as communicate with the user-operated device through a low-latency connection. And another one which can interface directly with the camera module and manage data transmission to the user at the applicational level of the TCP/IP protocol stack, with enough throughput for the specified video resolution and framerate.

The latter should also exchange sensor reading values and commands with the user-controlled device, introducing redundancy to the controls and thus allowing for more reliable operation.

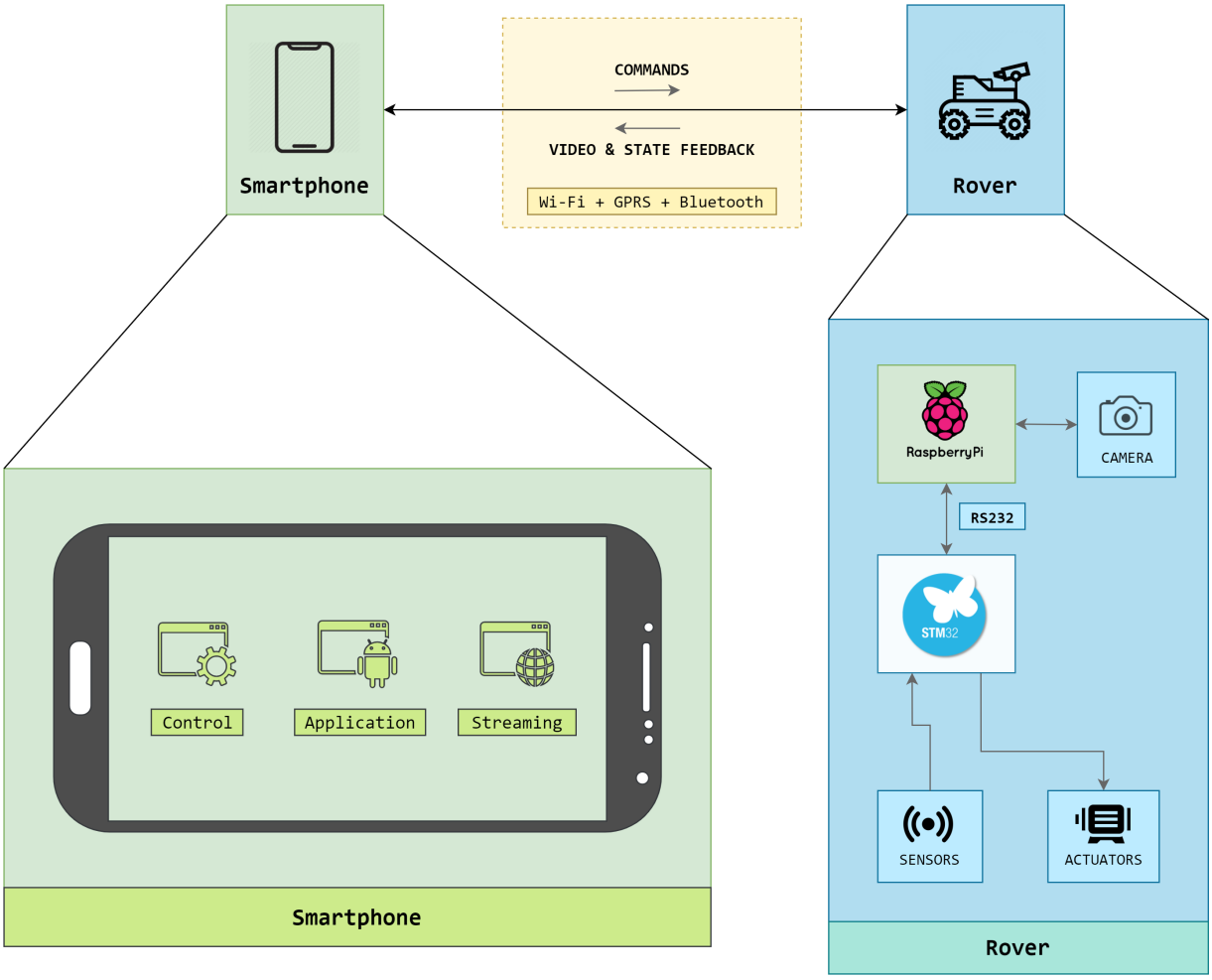


Figure 4.1.: Initial design: Block diagram view

Thus, summarising, the initial design yields the system illustrated in fig. 4.1, comprised of:

- **Raspberry Pi**: Interfaces with the camera directly, streaming that information to the smartphone. It also receives user commands and sends sensorial information it receives from the STM32 module back to the Smartphone, all through redundant Wi-Fi and GPRS connections;
- **STM32**: Receives user commands and sends back sensorial information through a Bluetooth connection, information it also uses to control the actuators;
- **Actuators**: DC Motors that control the movement of the Rover and headlights for nocturnal or low light conditions;

- **Sensors**: Odometric sensors that support the detection of obstacles and luminosity sensors;
- **Camera**: Device connected to the Raspberry Pi that allows the live stream of the car´s surrounding environment;
- **Smartphone**: Grant visual feedback from the live feed of the camera also allowing the user to control the movement of the vehicle intuitively;

For simulation purposes and in conformance with the extraordinary conditions imposed by the recently enacted confinement measures, the need rose to create a virtual environment to simulate the various subsystems of the Rover. This solution allows for integrated testing without the need to deploy it to the hardware, as illustrated in fig. 4.2.

The first of said subsystems is the Physical Environment Virtual Subsystem, which simulates the Rover and the physical environment, also receiving actuator inputs from the Navigation Virtual Subsystem, for which it generates the sensor readings. The latter also exchanges that information, as well as the status of the Rover and the commands from the user with the Smartphone module through a Bluetooth connection, which separates it from the non-simulated environment.

The Remote Vision Virtual Subsystem interfaces with the computer's Web Camera, which is meant to simulate the onboard camera of the Rover. Moreover, it can also communicate through Wi-Fi and GPRS, thus ensuring that communication is kept despite any failure from the Navigation Virtual Subsystem, as well as having shared access to the sensor reading values for monitoring of certain key parameters, depending on their refresh rate.

The RS232 connection between both controlling subsystems ensures proper synchronism and cooperation between them.

4.2. Foreseen specifications tests

The functional testing is generally regarded as those performed over any physical component or prototype. Here, however, a broader sense is used, to reflect the tests conducted into the system and the several prototypes, under the abnormal present circumstances. Moreover, as indicated in the design, the current development strategy encompasses the virtualization of all hardware components, enclosed in a single virtual environment.

Thus, it does not make sense to perform hardware related tests such as velocity measurements, autonomy, safety, etc. As such, the focus is shifted towards software and control verification, encompassing the following tests: functionality, image acquisition, communication, and control algorithms correctness.

The tests are divided into verification and validation tests.

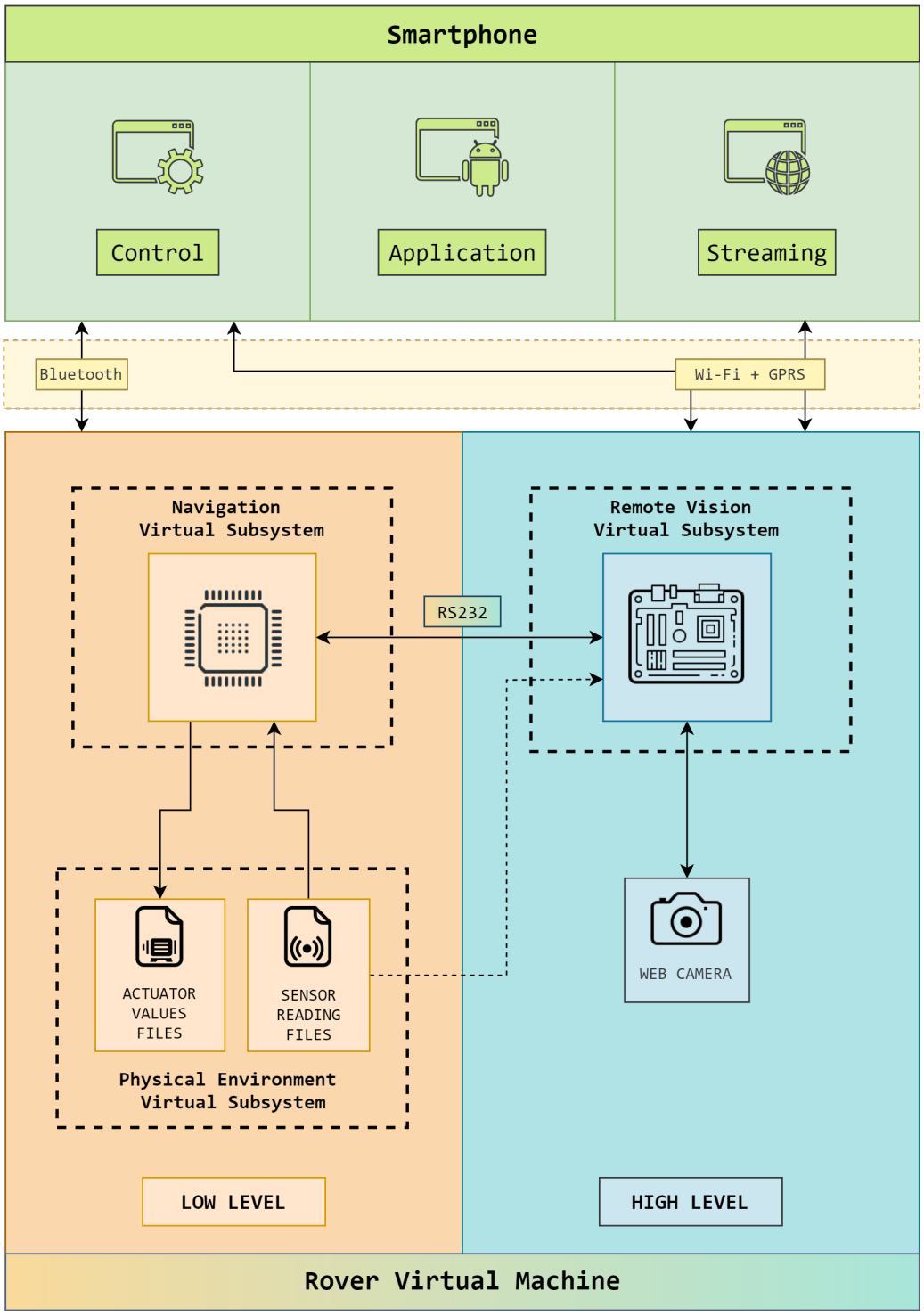


Figure 4.2.: Initial design: Virtual environment block diagram view

4.2.1. Verification tests

The verification tests are tests performed internally by the design team to check the compliance of the foreseen specifications. These tests are done after the prototype alpha is concluded.

4.2.1.1. Functionality

The remotely operated vehicle is composed of several modules distributed along several different platforms, some of which distanced from each other. In doing so, the proposed sets of functionalities should be tested in the integrated system, by tracking and analysing the user commands issued along the way until it finally reaches the vehicle (in the virtual environment), assessing if it is correctly processed. For example, if the user issues the vehicle to move to a given place (via smartphone interaction), the message sent to the vehicle must be signalled in each endpoint hit, and the vehicle should move to that place, symbolically detected by the modification of its virtual coordinates.

4.2.1.2. Image acquisition

The vehicle is equipped with remote vision to assist the user in its navigation, thus, requiring the following variables to be tested: frame rate, range, and resolution. In the current scenario, the virtual environment should provide access to a integrated camera, being fairly common nowadays in every modular component, thus, enabling easy testing.

4.2.1.2.1. Frame rate

To test frame rate, the user screen must be updated with the number of frames received from the camera per second and checked against the defined boundaries.

4.2.1.2.2. Range

To test camera's range, an object must be captured at increasing distances, until the object resolution fades or is unclear.

4.2.1.2.3. Resolution

The minimum resolution should be tested as providing the least amount of information required for the user, while minimizing data transfer and processing overhead.

4.2.1.3. Communication

The communication tests are performed in compliance to the specifications provided in Section 3.1.6, namely reliability and redundancy.

4.2.1.3.1. Reliability

To test communication reliability, the most critical communication link is chosen, namely the wireless communication between user's platform (smartphone) and vehicle's platform (virtual environment). Then, the communication link and protocol selected are tested by monitoring the ratio of dropped packets versus total packets, using an appropriate tool on both directions for transmission and reception.

4.2.1.3.2. Redundancy

To test communication redundancy, one communication channel should be turned off, verifying if the communication between nodes is still possible through another communication channel. For example, in the communication between host (user's platform) and remote (virtual environment) guest systems, the priority communication is performed between host and high-level subsystem. However, if this is turned off, the host must also be able to communicate with the remote system via low-level subsystem.

4.2.1.4. Correctness of the control algorithms

As previously mentioned, the speed and position must be continuously monitored to ensure proper vehicle operation. Under the current circumstances, where the sensors and actuators are virtualized, the control loops can be externally stimulated through input files containing the relevant data. Then, the behaviour of the system can be analysed and verified for some cornercase situations, assessing the control algorithms correctness.

4.2.2. Validation tests

The validation tests should be performed by the client using the product's manual, so it is expected that a user without prior experience with the product should be able to use it correctly and safely. On the current circumstances, validation tests should be limited to user interface validation.

For this purpose, an external agent will be provided with the software application and the respective installation and usage manuals, and the feedback will be collected and processed to further improve the product.

5. Design

The multi-material part fabrication is a complex topic and most current commercially available systems have been designed for mono-material part fabrication [5] and are unprepared for multi-material processing due to the lack of flexibility and processing capability.

5.1. Navigation Virtual Subsystem

5.1.1. Control

5.2. Physical Environment Virtual Subsystem

5.3. Remote Vision Virtual Subsystem

5.4. Smartphone

6. Implementation

The multi-material part fabrication is a complex topic and most current commercially available systems have been designed for mono-material part fabrication [5] and are unprepared for multi-material processing due to the lack of flexibility and processing capability.

6.1. Navigation Virtual Subsystem

6.1.1. Control

6.2. Physical Environment Virtual Subsystem

6.3. Remote Vision Virtual Subsystem

6.4. Smartphone

7. Testing

The multi-material part fabrication is a complex topic and most current commercially available systems have been designed for mono-material part fabrication [5] and are unprepared for multi-material processing due to the lack of flexibility and processing capability.

7.1. Unit testing

7.1.1. Navigation Virtual Subsystem

7.1.1.1. Control

7.1.2. Physical Environment Virtual Subsystem

7.1.3. Remote Vision Virtual Subsystem

7.1.4. Smartphone

7.2. Integrated-testing

8. Verification and Validation

The multi-material part fabrication is a complex topic and most current commercially available systems have been designed for mono-material part fabrication [5] and are unprepared for multi-material processing due to the lack of flexibility and processing capability.

8.1. Verification

8.2. Validation

9. Conclusion

In this chapter the conclusions and prospect for future work are presented.

Bibliography

[1] Ian Sommerville. Software process models. ACM computing surveys (CSUR), 28(1):269–271, 1996.

[2] Michael A Cusumano and Stanley A Smith. Beyond the waterfall: Software development at microsoft. 1995.

[3] Bernd Bruegge and Allen H Dutoit. Object-Oriented Software Engineering Using UML, Patterns and Java-(Required), volume 2004. Prentice Hall, 2004.

[4] Albert S Huang and Larry Rudolph. Bluetooth essentials for programmers. Cambridge University Press, 2007.

[5] Terry Wohlers. Wohlers report 2011: Additive manufacturing and 3d printing, state of the Industry, wohlers associates, ft. Collins, CO, 2011.

Appendices

A. Project Planning — Gantt diagram

In Fig. A.1 is illustrated the Gantt chart for the project, containing the tasks' descriptions.

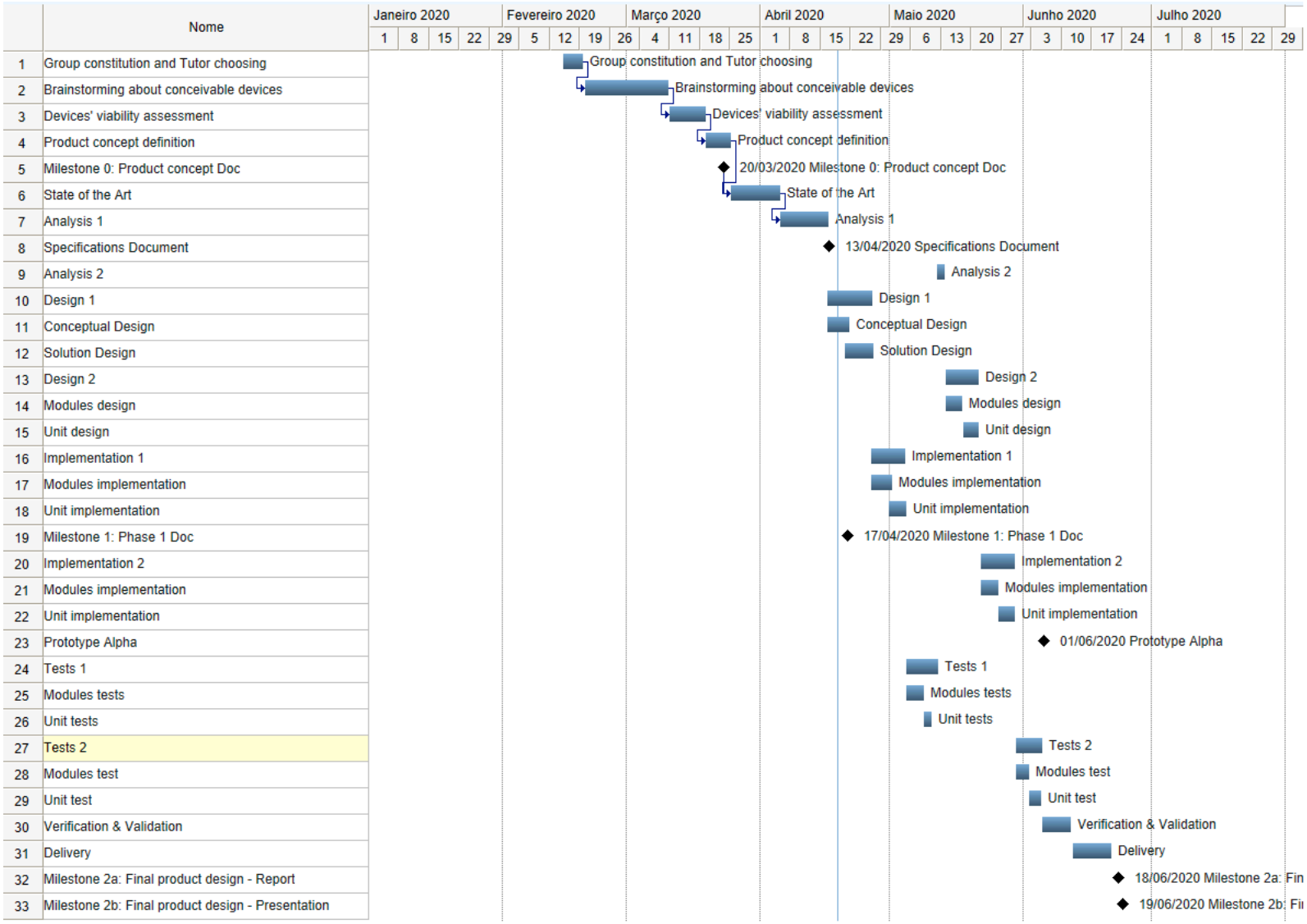


Figure A.1.: Project planning — Gantt diagram