EMBEDDED SYSTEMS
RESEARCH
GROUP

University of Minho
School of Engineering

Integrated Master in Industrial Electronics and Computers Engineering

Project 1

Zephyrus

Hand Motion-Controlled Remote Car

Authors:
Hugo Carvalho A85156
José Mendes A85951

Professor:
Dr. Adriano Tavares

October 2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This document consists of an analysis and presentation of the work done in the development of Zephyrus.

In the first chapters, the product will be put in the right context globally to understand the need for its existence and where it fits in the market. A high-level overview of the project will also be provided in these chapters. This is meant for future reference when designing the product as well as analysing the report.

Further in the document, one will be able to find documentation detailing the design and implementation processes, which will hopefully be simple to understand and reproduce.

The report will culminate in the test and verification of the results achieved in the development stages, in order to understand the impact of the decisions that were made and where to make improvements in future work.

1.1 Problem Statement

Recent developments in research of motion-tracking and gesture-controlled robotic products have shone a new light on entertainment, special needs teaching, and accessibility-focused applications. Products like motion-sensing console controllers, robotic arms, and prosthetics have popularized and extended the knowledge on gesture and impulse-based technologies throughout the past few years, and their market is predicted to continue to grow in the future, especially in Asia and Oceania [10], as presented in figure 1.1.1.

To provide a better understanding of the concepts associated with the forementioned technologies through R&D (Research and Development), the project will consist of a gesture-controlled car with a wristband as a remote controller. The user will be able to accelerate and steer the vehicle using natural wrist and finger movements.

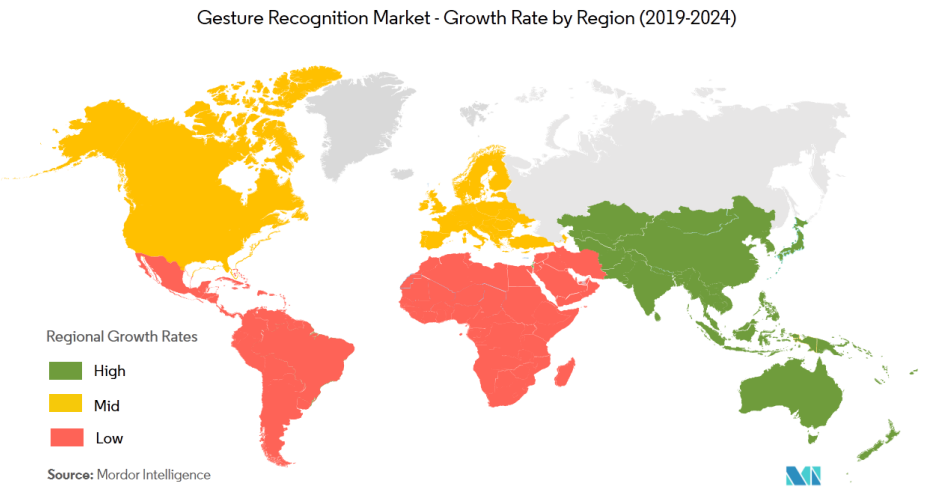

Figure 1.1.1: Gesture Recognition Market Growth

1.2 Problem Statement Analysis

From the Problem Statement Analysis, one can extrapolate an overview of the main components, features they provide and relationships between them. The schematic in figure 1.2.1 describes a system comprised by:

- A **Local Computational Unit**, which will control the wheels of the car in accordance to the user inputs, which it receives from the Remote Computational Unit;

- A **Remote Computational Unit**, which senses user input and converts it into a digital format, sending it to the Local Computational Unit.

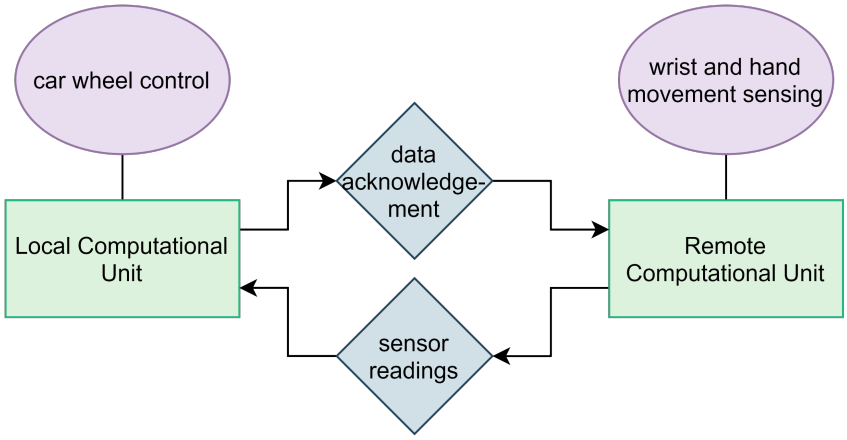

Figure 1.2.1: Problem Statement Diagram

Chapter 2

Analysis

This chapter starts with brief market research about similar products to ultimately distinguish Zephyrus from its competitors.

The requirements and constraints of the project are presented as well as a general system overview to better visualize the objectives. Additionally, the system is analyzed in terms of its software and hardware architectures and divided into local and remote, proceeding with the presentation of various diagrams that allow further understanding of each system.

This stage ends with the proposed initial budget and the project's Gantt diagram.

2.1 Market Study

The market for gesture-controlled or gesture-recognition products is not very crowded, but it is competitive nonetheless [10], as displayed in figure 2.1.1. As stated, some of the best efforts in this area can be found in research and, in that sense, both strands will be analysed in this section, as a mean to understand, not only what this product might try to compete with, but also what technologies are being developed which might be used in future competing products or iterations of this same project.

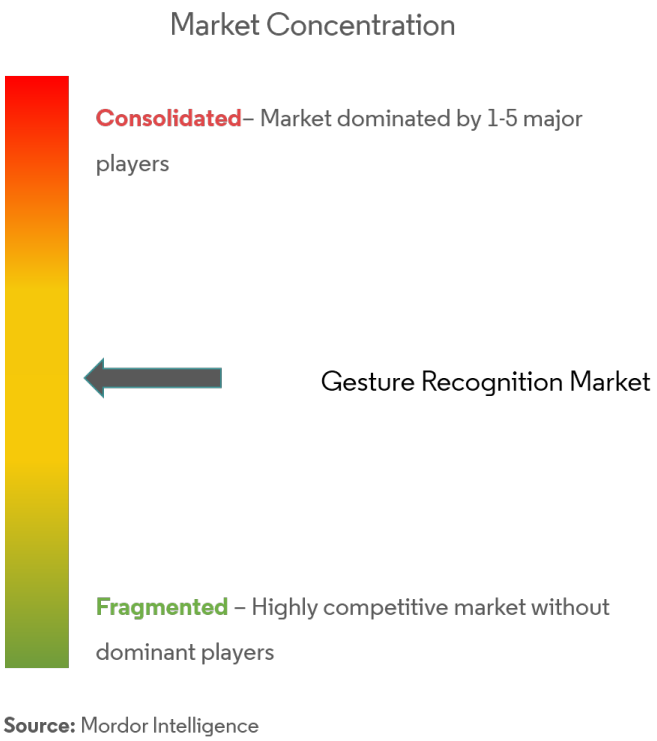

Figure 2.1.1: Gesture Recognition Market Placement

Ultigesture's Gesture controlled RC Car (Kickstarter)

This product was revealed in 2017 along with a Kickstarter campaign, with the promise of being easy and intuitive to control.

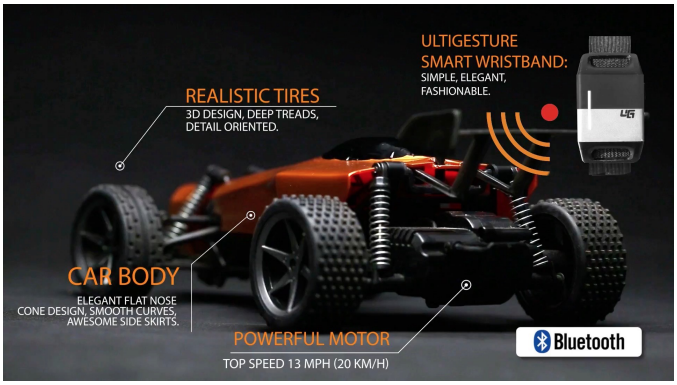

Figure 2.1.2: Ultigesture's arm gesture-controlled RC car

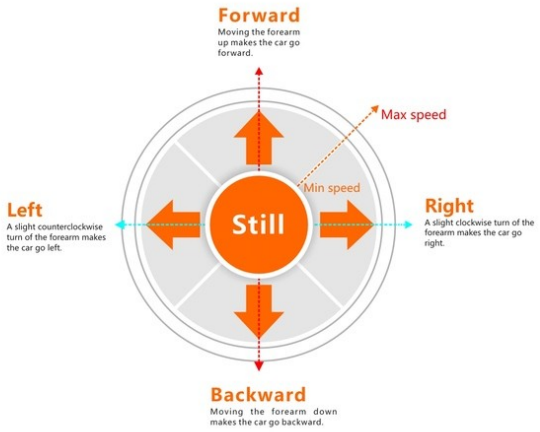

Figure 2.1.3: Ultigesture's gesture-controlled RC car's instructions

Pros:

- Practical setup;
- Appealing design;
- The use of Bluetooth allows the car to be controlled both by arm gestures and smartphone.

Cons:

- Unpractical and uncomfortable to command.

Specified range: $20m$

Predicted cost: 89$

EMG/EEG Controlled RC Car

In 2016 a team composed by Christian Ward, James Kollmer, Robert Irwin and Andrew Powell designed a quick prototype for a toy car which was controlled with arm movement and brain activity. Both sensing peripherals, as well as the car were connected to a computer, which ultimately controlled the movement of the car. It would be unfair to analyse this as a final product but it will serve as a reference for what can be done in practise.

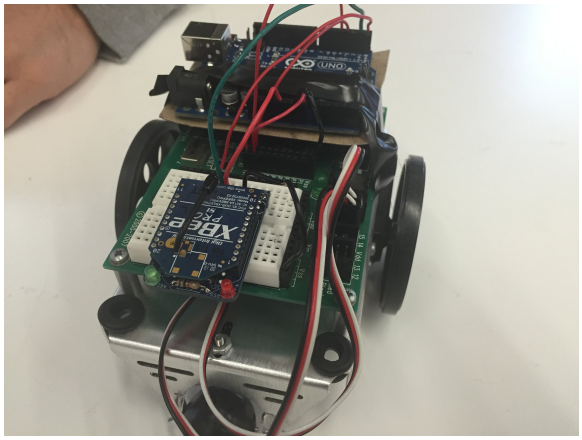

Figure 2.1.4: EMG/EEG Controlled RC Car

Generic Glove-controlled RC Quadcopter Drone

This is a generic toy glove-controlled drone which can found in any Chinese website. It is controlled by hand movements and it also includes a button for keeping the drone hovering in the same place.

Figure 2.1.5: Generic Glove-controlled RC Drone

Pros:

- Practical and comfortable to command;
- Impactful first impression;
- Cheap.

Cons:

- Cheap-feeling;
- Can only be worn in the right hand.

2.2 Added Value

Although there aren't many products in the market to compete with Zephyrus, strong efforts have been made in certain key aspects, such as good usability and competitive pricing, in the existing products. Zephyrus, specifically, aims to match all those products in those characteristics, but also be easy and versatile to set up, as well as comfortable to use and control.

2.3 Requirements and Constraints

The development requirements are divided into functional and non-functional if they are main functionalities or secondary ones, respectively. Additionally, the constraints of the project are classified as technical or non-technical.

2.3.1 Requirements

Functional Requirements

- Sense the user inputs and generate the band outputs;
- Take the wristband inputs and generate the desired motor outputs;
- Use wireless communication between the local and remote systems;
- Control the establishment and closure of the connection from the remote system.

Non-Functional Requirements

- Have low power consumption;
- Have low latency in the connection between local and remote;
- Have a comfortable and practical wristband/glove;

2.3.2 Constraints

Technical Constraints

- Use the Nucleo-STM32F767ZI as the development board;
- Use at least three different types of sensors;
- Use FreeRTOS as the RTOS;
- Use a controller based on Reinforcement Learning following the Actor-Critic method;
- Use Keil MDK-ARM as the development environment;
- Use Object-Oriented Programming Languages;

Non-Technical Constraints

- Project deadline at the end of the semester;
- Pair workflow;
- Limited budget;

2.4 System Overview

The diagram in figure 2.4.1 represents an initial big picture of the project to facilitate objective identification. Concerning the diagram, one can identify three main blocks:

- The **Local Board** block;
- The **Remote Board** block;
- The **External Environment** block.

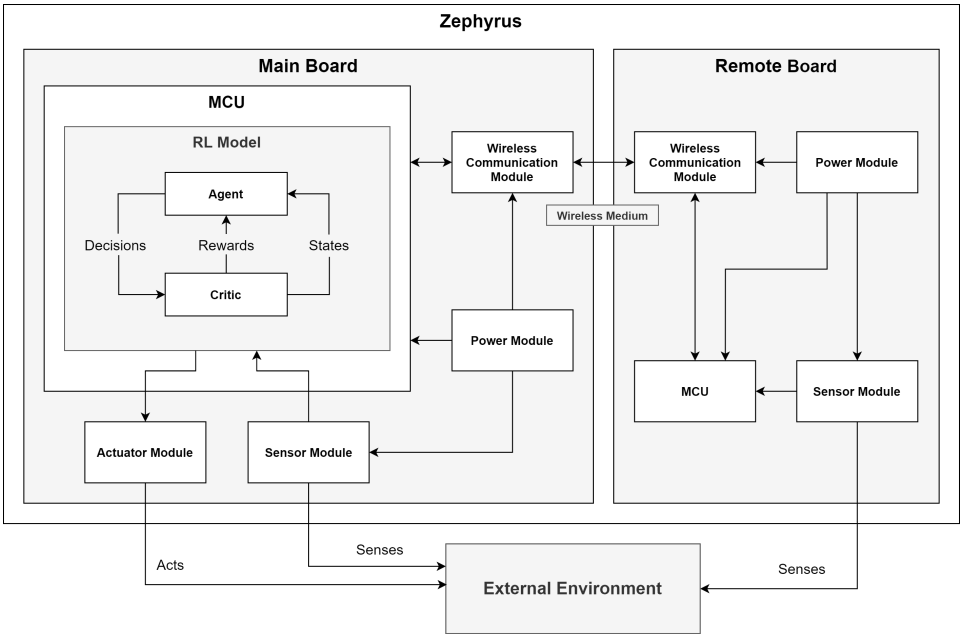

Figure 2.4.1: System Overview Diagram (Aug. in Appendix A.1)

2.5 System Architecture

2.5.1 Hardware Architecture

The diagram in figure 2.5.1 represents an initial hardware big picture to facilitate objective identification. Concerning the diagram, aside from the External Environment, one can identify two main blocks:

- The **Local Board**. This unit is comprised by the main MCU (in the STM32 board) for housing the RL Model and intermediate its interactions with the world, as well as the Actuator Module, for driving the wheel motors, the Power Module, for power delivery and management, and the Wireless Communication Module, for facilitating the communication with the Remote Board through a wireless medium;
- The **Remote Board**. This unit contains the Sensor Modules network of the project for sensing the environment, its own MCU, whose job is to concentrate and multiplex the sensor network, sending data through its own Wireless Communication Module, and the Power Module which will provide all of the other modules with power.

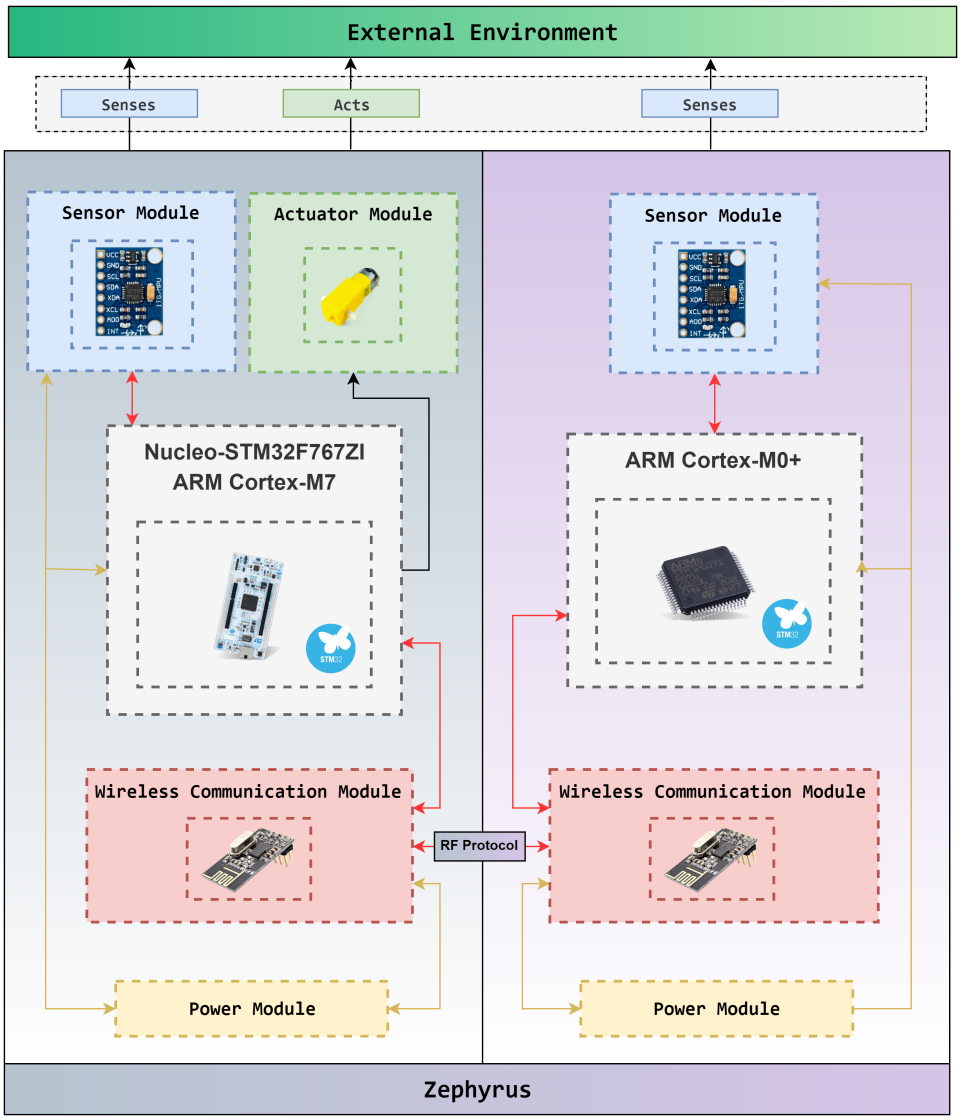

Figure 2.5.1: Hardware Architecture Diagram

2.5.2 Software Architecture

The software architecture for both the local and remote systems (figures 2.5.2 and 2.5.3, respectively) is divided into three layers, those being:

- The **Operating System** layer, which is comprised by the Operating System drivers and Board Support Packages;
- The **Middleware** layer, which includes software for abstracting the lower level packages and streamlining the development of complex algorithms;

- The **Application** layer, where the core functionality of the program is built, with resource to the API's in the lower level layers.

Both are soft real-time systems with, focused on low power consumption and effective communication, both with each other and with the sensor modules. As such, the analysis for both systems foresee the necessity for a Real-Time Operating System for managing real-time events and multi-tasking, drivers for serial communication and specific power management utilities.

The Remote System is mainly focused on sensing and processing values relative to the environment while sending information pertaining those values to the Local System.

The Local System, on the other hand, serves the specific purpose of hosting a controller that is composed by a Reinforcement Learning model. In that sense, it also needs specific, microcontroller-optimized middleware for RL and motor control.

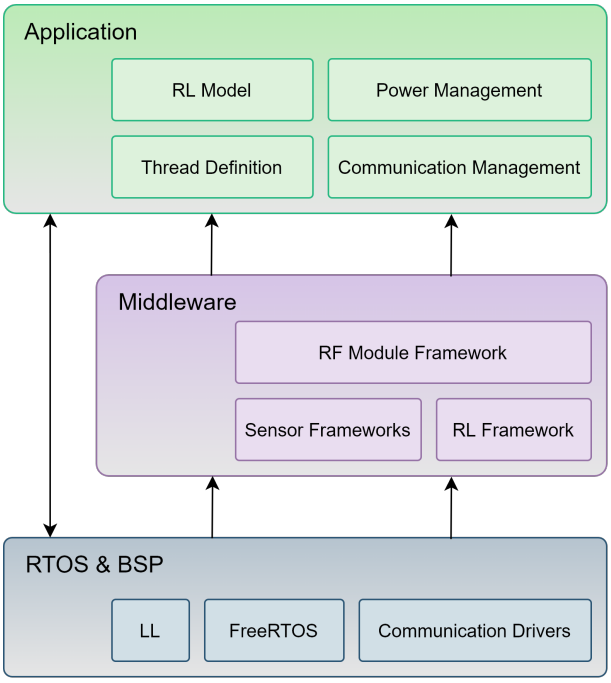

Figure 2.5.2: Software Architecture Diagram - Local System

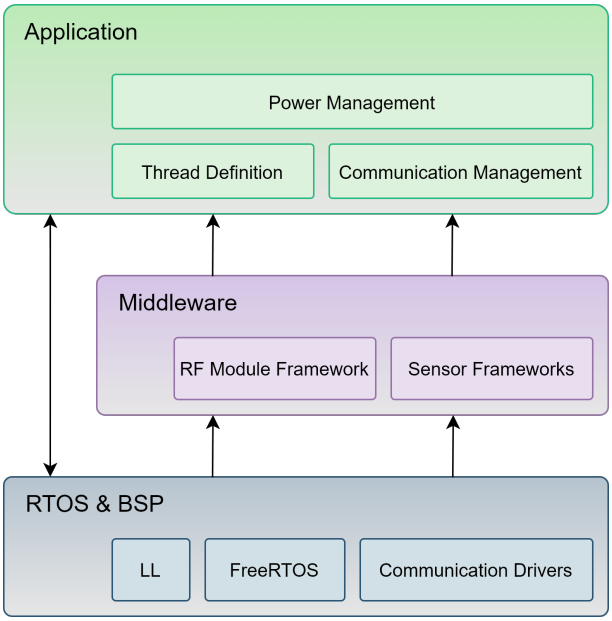

Figure 2.5.3: Software Architecture Diagram - Remote System

2.6 Local System

As referred to in section 2.4, the overall system includes a bulkier **Local System**. The analysis of the latter will be presented in this section.

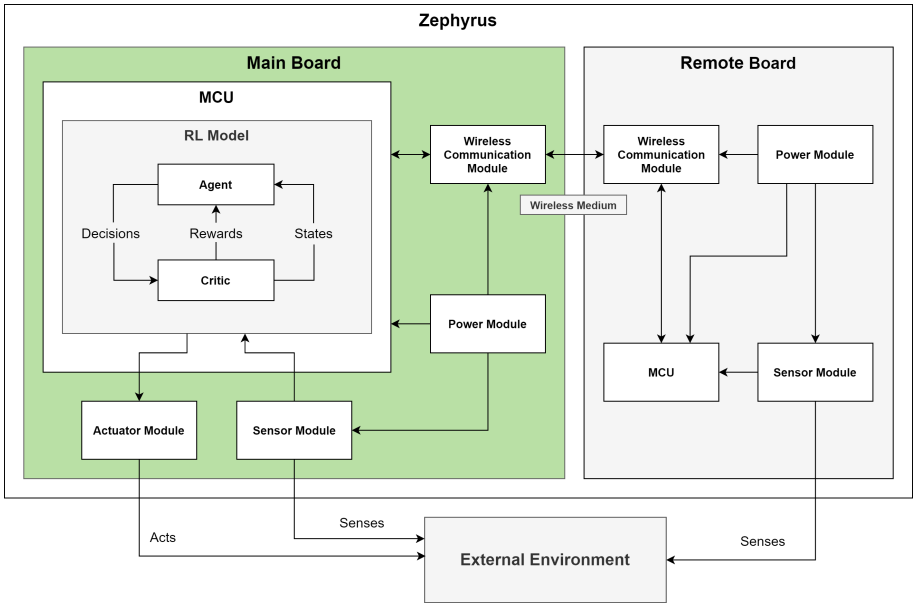

Figure 2.6.1: System Overview Diagram - Local

2.6.1 Events

In order to understand how the system works, it is important to know what are the most important events that can take place in it and how it will respond to each one of them. Table 2.1 highlights these events from the perspective of the Local System, categorizing them by their source and synchronism and linking it to the system's intended response.

Event	**System Response**	**Source**	**Type**
Power on	Initialize sensors and listen for connnection	User	Asynchronous
Connection Request	Verify credentials and respond to the request	Remote System	Asynchronous
Sampling Period Elapsed	Request sample from sensors	Local System	Synchronous
Command Received	Prepare the information to be part of the state observations to be considered by the agent	Remote System	Asynchronous
RL Timestep Elapsed	Calculate target value	Inference Engine	Synchronous
Target Value Calculated	Perform the action infered by the agent	Agent	Asynchronous

Table 2.1: Local Events

2.6.2 Use Cases

It is necessary to study the system's possible use cases to understand and model its intended behaviour. In the Local System's case, all the interactions happen with the Remote System. The diagram in figure 2.6.2 illustrates those interactions.

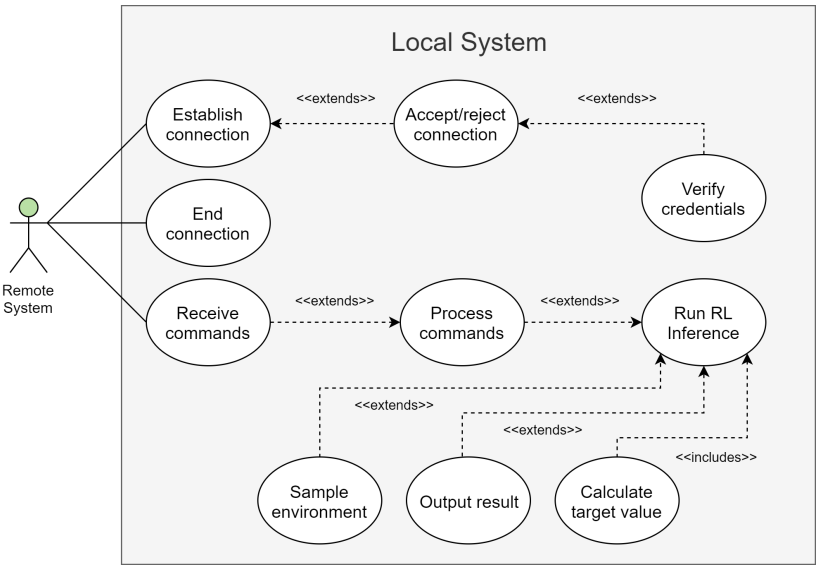

Figure 2.6.2: Local System's Use Case Diagram

2.6.3 State Machine Diagram

The two main functional states of the Local System - *Startup* and *Execution* - and their division into more complex state machines corresponding to each of its subsystems, are depicted in figure 2.6.3.

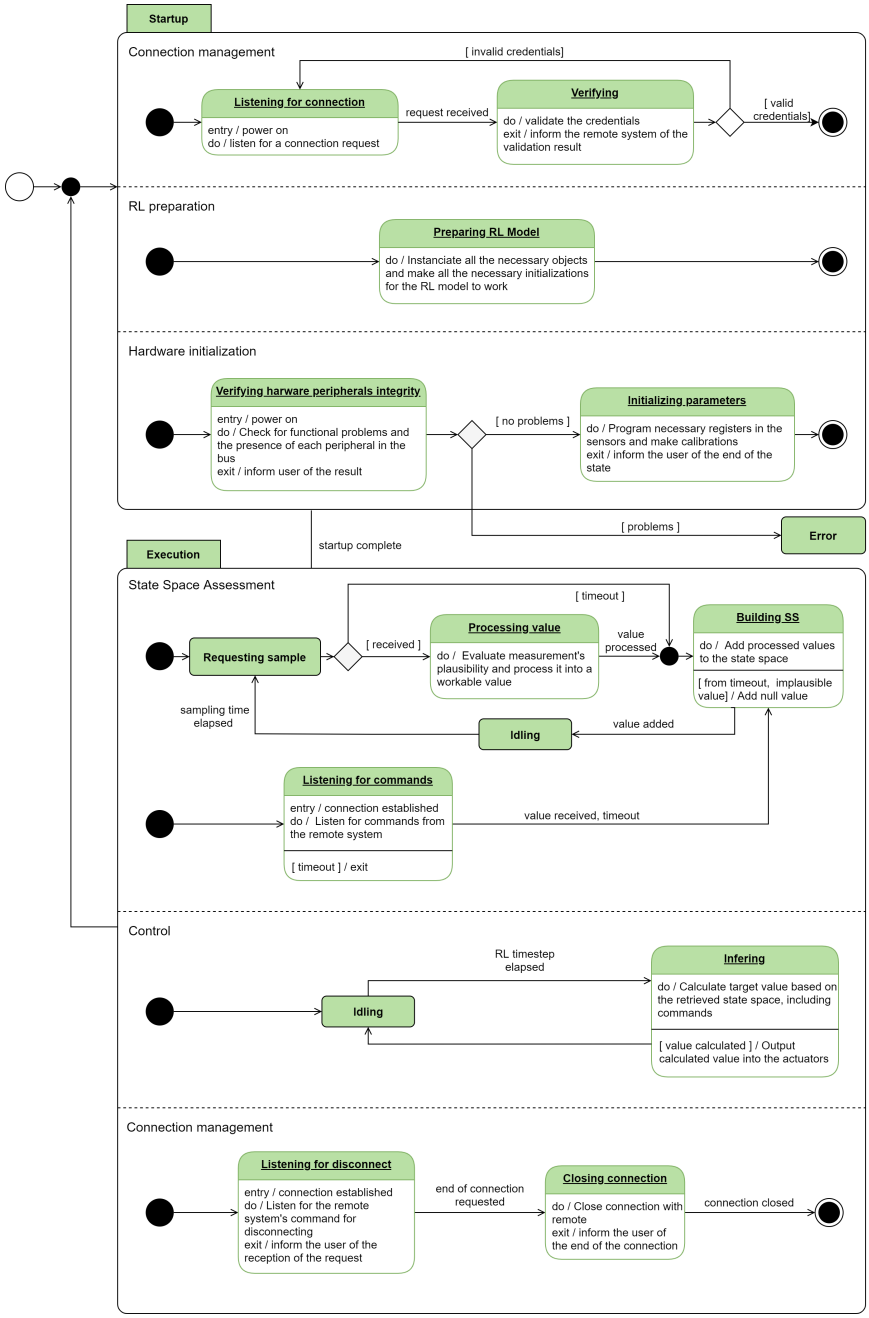

Figure 2.6.3: State Machine Diagram - Local (Aug. in Appendix A.6)

2.6.4 Sequence Diagram

The model for the Local System's intended response and behaviour relative to different stimuli dynamically, is presented as a sequence diagram in figures 2.6.4 and 2.6.5.

When powering on, the Local System will immediately initialize all required aspects of sensor-related hardware components and verify their integrity, to be able to start sampling as soon as it is necessary. It will also start listening to connection requests from the Remote System. Upon reception of one of such requests, it will accept or reject it based on previously established connection credentials. Whenever connected, the Local System will continuously sample the environment around it, changing the state observations for the Reinforcement Learning model.

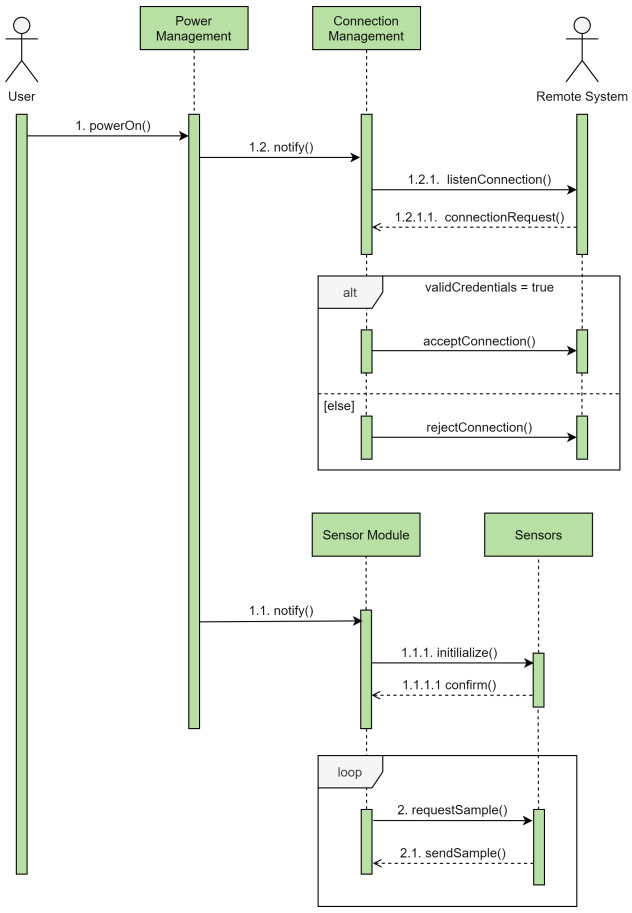

Figure 2.6.4: Local System Sequence Diagram (Aug. in Appendix A.2)

As previously mentioned in table 2.1, whenever a command from the Remote System is received, it is to be made part of the state observations

for the Reinforcement Learning algorithm as well. In case a command is not received in time, the last command received will be repeated to preserve the predictability of the system.

Periodically, an iteration of the control routine is executed. The first logical step in that chain of events is the calculation of the target output values by the Agent, which will be followed by the output of said values and calculation of the reward that will allow the agent to update the policy.

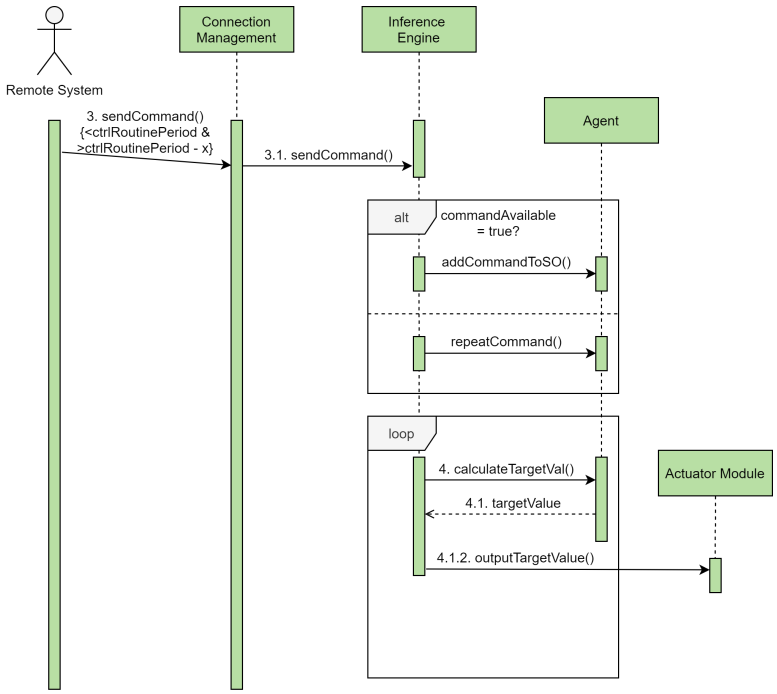

Figure 2.6.5: Local System Sequence Diagram (Aug. in Appendix A.3)

2.7 Reinforcement Learning

All the diverse Reinforcement Learning algorithms can be categorized into one of three main families:

- **Action-value** family;
- **Actor-critic** family;
- **Policy-gradient** family;

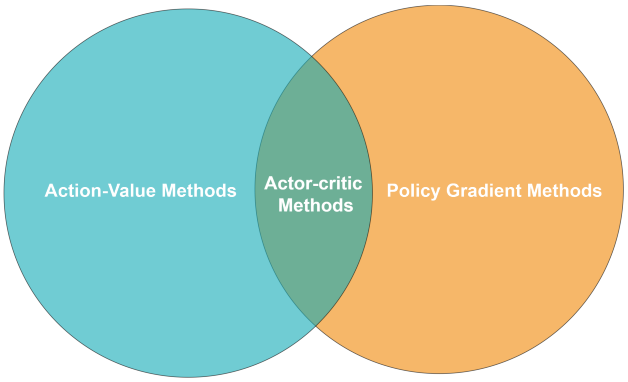

Figure 2.7.1: Reinforcement Learning Algorithm Families

In figure 2.7.2 one can identify three main components of an Actor-Critic Reinforcement Learning Model:

- The **Actor/Agent** is the controlling entity of the system, which dictates actuator outputs based on the current state of the system, internal and external.
- The **Critic** is the entity that works to tune the behaviour of the Actor through a reward-based policy.
- The **External Environment** can be identified as everything that is not the Actor or Critic but is influenced by the Actor's actions and helps to stimulate it.

The model is time-dependent and sequential, since the reward calculation is based, not only on the assessment of the Agent's performance in the present, but also on past rewards. The environment is also stochastic because some inputs change randomly in time and it's also non-deterministic, meaning, the same inputs can generate different outputs as the neural network changes between policy updates.

The model is time-dependent and sequential, since the reward calculation is a sum, not of all past rewards. The environment is also stochastic because it's non-deterministic, meaning, the inputs change in time.

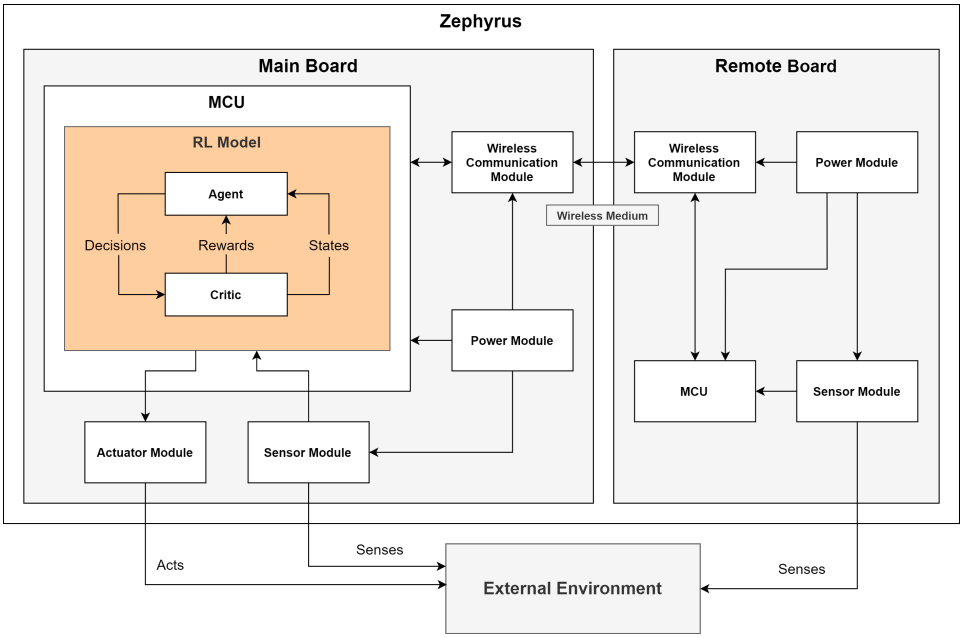

Figure 2.7.2: System Overview Diagram - Reinforcement Learning Model

Reinforcement Learning Workflow

After specifying the type of learning algorithm to utilize, one must follow a basic training workflow [5] as presented in figure 2.7.3.

Figure 2.7.3: Reinforcement Learning Workflow Schematic

Model Conversion & Deployment on Embedded Target

The final step of the mentioned process is the model conversion and deployment on the target device. In this case, the system is a resource-scarce embedded system [1], which demands a more careful deployment considering the compatibility between the model developing tool and the embedded target.

The tool selected for performing the model creation and training was **Tensorflow** since it has been extensively tested with many processors based on the **Arm Cortex-M Series** architecture [4] and has been ported to several other architectures. Moreover, it is important to note that one of the supported developing boards listed by Tensorflow was the **STM32F746 Discovery kit** [3] because Zephyrus' Local Board will be an **STM32 NUCLEO-F767ZI**. This resolves any doubts about compatibility issues as the Tensorflow board support should extend to all the STM32F7 family.

For the deployment stage following this approach, one must convert the reinforcement learning model created and trained in TensorFlow to a TensorFlow Lite model specially made for microcontrollers. At this point, two choices were presented regarding deployment:

- Use TensorFlow Lite APIs
- Use STM32 X-Cube.AI APIs

After some research, one discovered that the deployment using X-Cube.AI was faster and took less flash memory than TensorFlow Lite deployment [6]. Additionally, using the same reinforcement learning-generated model it was also possible to speed up inference [6]. The X-Cube.AI core engine [9] is represented in figure 2.7.4, and the proposed **Edge Machine Learning Framework** [1][7] for Zephyrus is presented in figure 2.7.5.

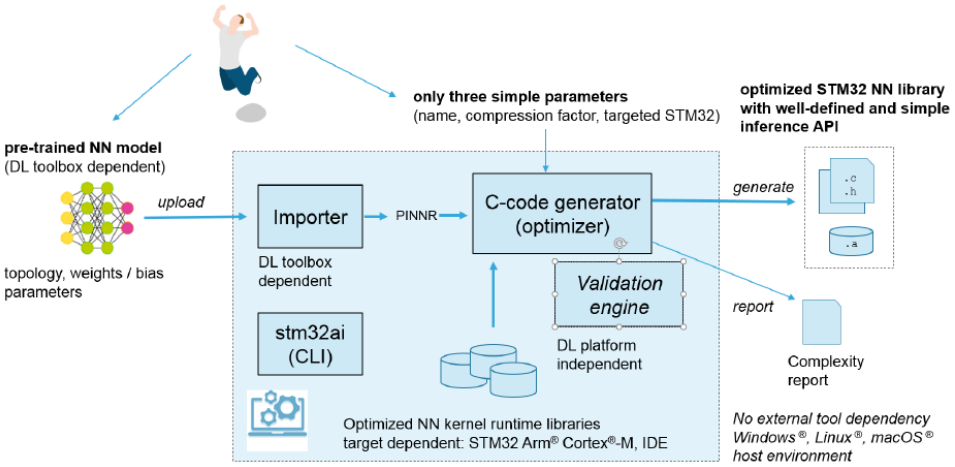

Figure 2.7.4: X-CUBE-AI Core Engine

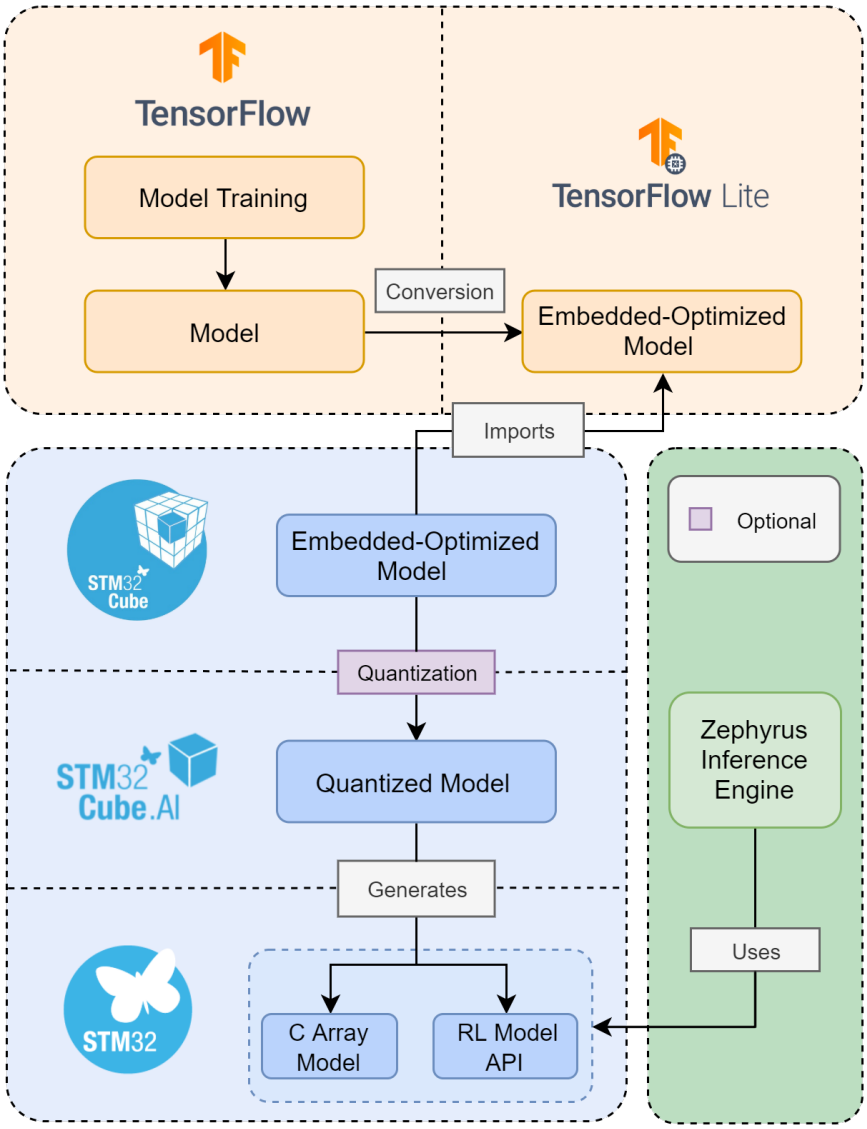

Figure 2.7.5: Zephyrus Edge Machine Learning Framework

2.8 Remote System

The control of the more complex system presented in section 2.6 is intended to be performed by a more compact **Remote System**, to be analysed in this section.

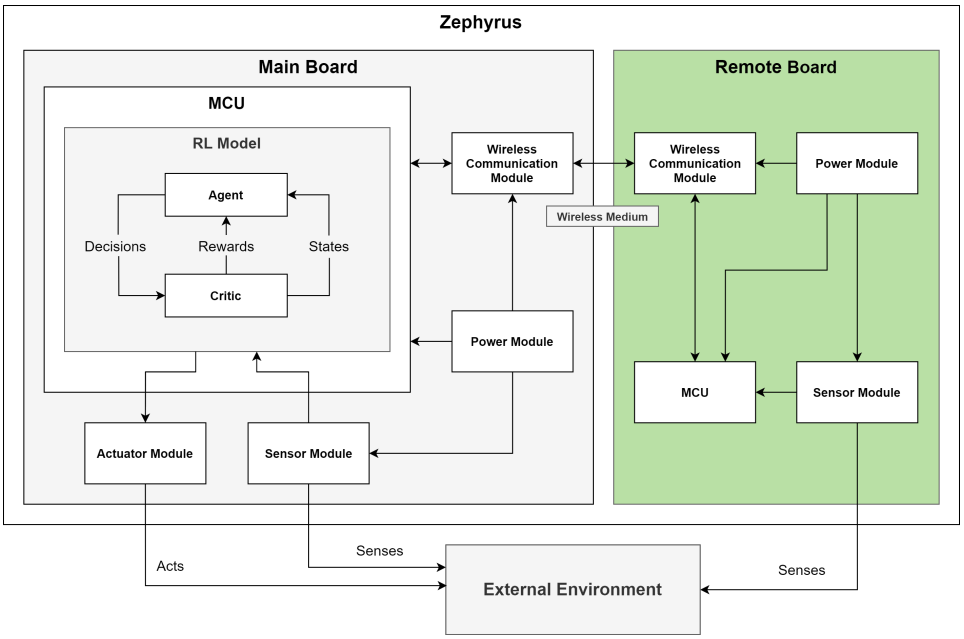

Figure 2.8.1: System Overview Diagram - Remote

2.8.1 Events

Table 2.2 presents the relationship between the most relevant events and their respective responses for the Remote System.

Event	System Response	Source	Type
Init connection button pressed	Try to establish a connection between local and remote	User	Asynchronous
Generate band movement	Send commands to local system	User	Asynchronous
Power on	Initialize sensors	User	Asynchronous
Sampling period elapsed	Request sample from sensors	Remote System	Synchronous

Table 2.2: Remote Events

2.8.2 Use Cases

The Remote System serves as a bridge between the user and the Local System. Thus, its behaviour is defined by its interaction with these two actors, as exemplified by the diagram in figure 2.8.2.

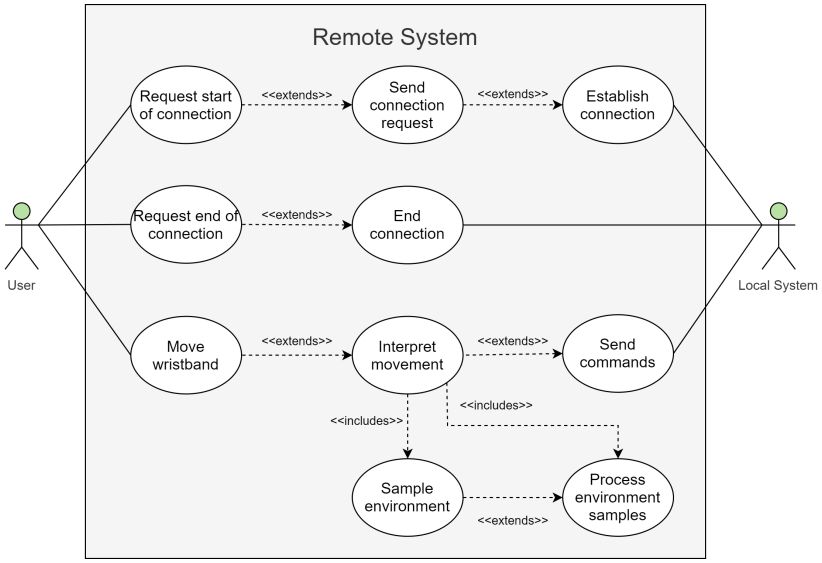

Figure 2.8.2: Remote System's Use Case Diagram

2.8.3 State Machine Diagram

The two main functional states of the Remote System - *Startup* and *Execution* - and their division into more complex state machines corresponding each of its subsystems, are depicted in figure 2.6.3.

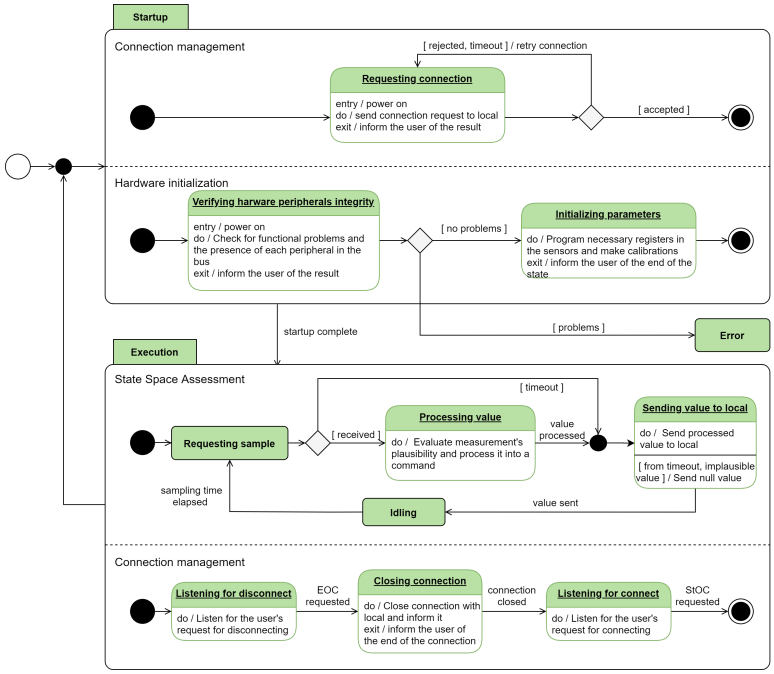

Figure 2.8.3: State Machine Diagram - Remote (Aug. in Appendix A.5)

2.8.4 Sequence Diagram

The Remote System's dynamic behaviour is presented as a sequence diagram in figure 2.8.4

When powering on, the Remote System initializes all required aspects of sensor-related hardware components and verifies their integrity. As soon as a connection with the Local System is established it will start sampling. The next step of the power-on sequence is sending a connection request to the Local System, to be accepted or rejected within an acceptable interval or repeated otherwise.

Periodically, the Remote System's controller will request a sample from the on-board sensors, to be pre-processed as necessary and sent to the Local System, where it will be interpreted as a command.

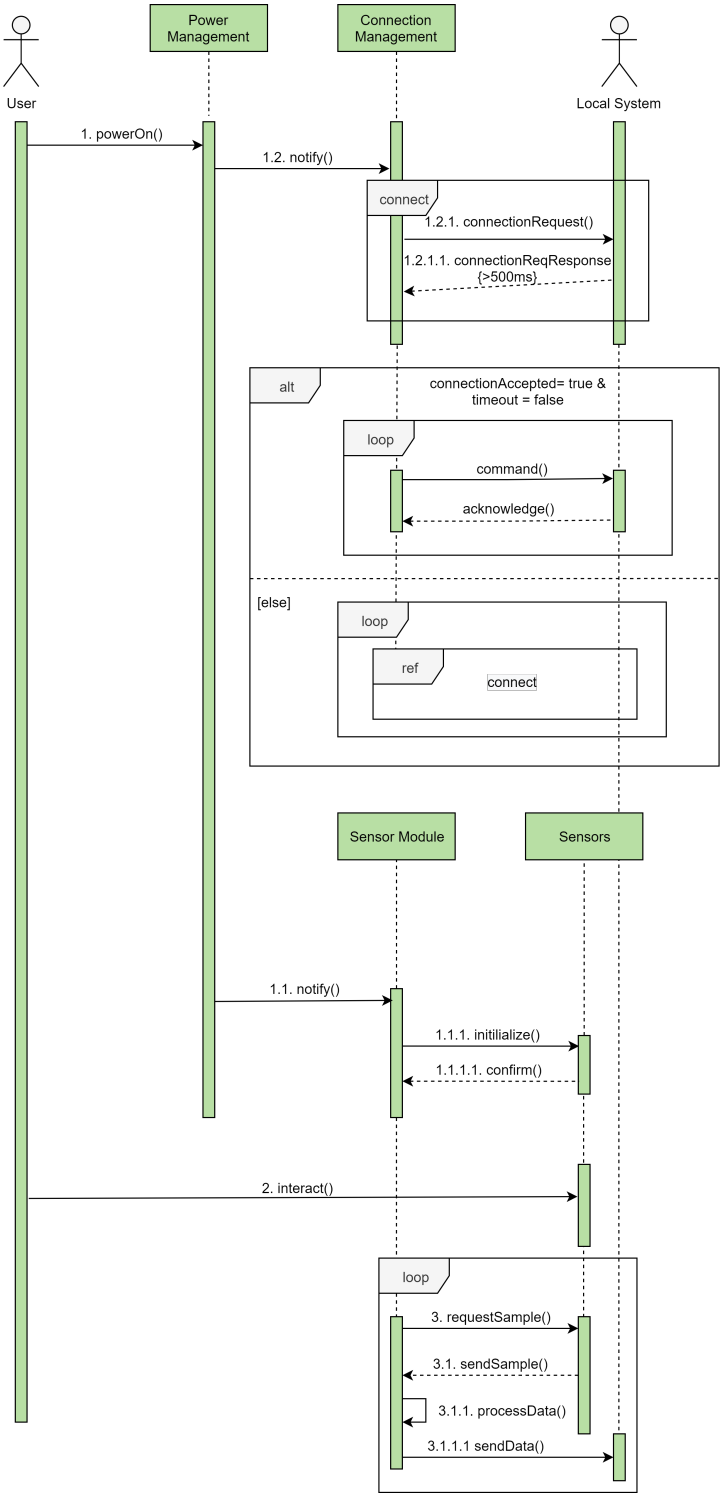

Figure 2.8.4: Remote System Sequence Diagram (Aug. in Appendix A.4)

2.9 Initial Budget

Table 2.3 represents an initial project budget estimate.

Item	**Cost**
STM32 NUCLEO-F767ZI	27€
STM32 F051	3€
Sensor Modules	20€
Power Modules	11€
Communication Modules	7€
Car Structure	14€
Total	82€

Table 2.3: Initial Budget

2.10 Gantt Diagram

	Task Name	Duration	Start	ETA
1	**M0: Planning**			
1.1	Project Choice	2 days	15.10.20	16.10.20
1.2	Problem Statement	3 days	17.10.20	19.10.20
1.3	Project Plan	7 days	20.10.20	26.10.20
2	**M1: Analysis**			
2.1	System Overview	2 days	20.10.20	21.10.20
2.2	Requirements and constraints	1 day	22.10.20	22.10.20
2.3	System Architecture	2 days	23.10.20	24.10.20
2.4	System Analysis	2 days	25.10.20	26.10.20
2.5	Market Research	2 days	27.10.20	28.04.12
2.6	Estimated Budget	2 days	29.10.20	30.10.20
2.7	Diagrams	10 days	31.10.20	9.11.20
3	**M2: Design**			
3.1	Analysis Review	3 days	10.11.20	12.11.20
3.2	Hardware Specification	10 day	13.11.20	22.11.20
3.3	Component Shipping	14 days	23.11.20	7.12.20
3.4	Software Specification	10 days	18.11.20	27.11.20
3.5	RL Model Design and Training	8 days	28.11.20	5.12.20
4	**M3: Implementation**			
4.1	Hardware Testing	7 days	8.12.20	11.12.20
4.2	Software Implementation	32 days	1.12.20	31.12.20
4.3	Hardware Implementation	32 days	1.12.20	31.12.20
4.4	RL Model Conversion	5 days	1.12.20	5.12.20
5	**M4: Testing & Verification**			
5.1	Software Unit Testing	4 days	1.1.21	4.1.21
5.2	Software Integrated Testing	5 days	4.1.21	8.1.21
5.3	Hardwareware Unit Testing	4 days	1.1.21	4.1.21
5.4	Hardware Integrated Testing	5 days	4.1.21	8.1.21
5.5	Verification	3 days	9.1.21	11.1.21

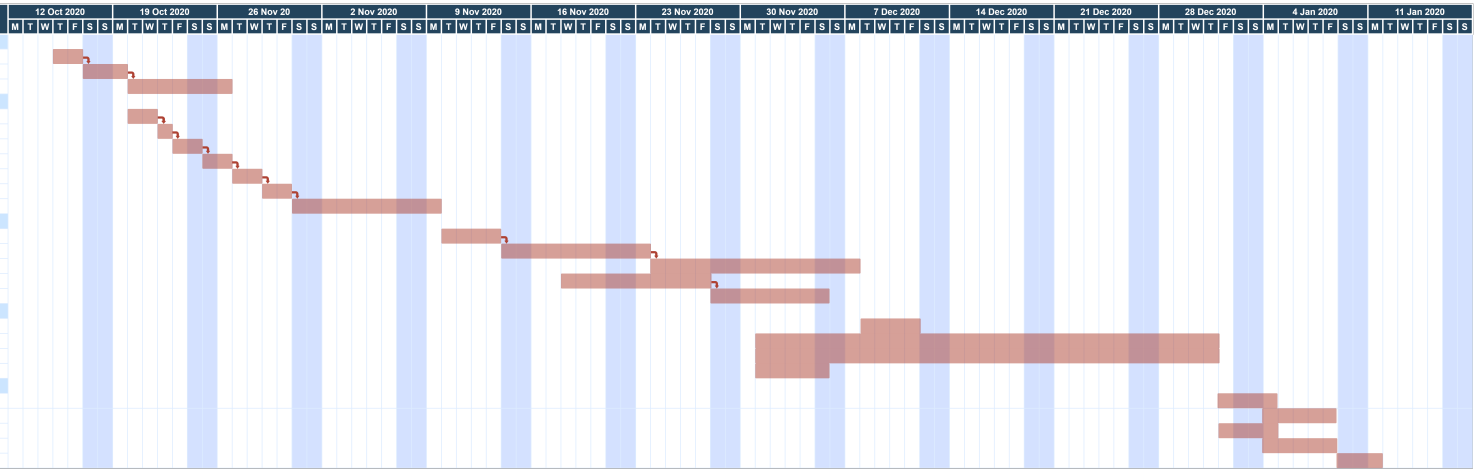

Figure 2.10.1: Gantt Diagram

Bibliography

[1] Branco, S., Ferreira, A. and Cabral, J., 2019. Machine Learning in Resource-Scarce Embedded Systems, FPGAs, and End-Devices: A Survey. Electronics, 8(11), p.1289.

[2] STMicroelectronics. 2020. X-CUBE-AI - Stmicroelectronics. [online] Available at: <https://www.st.com/en/embedded-software/x-cube-ai.html#overview> [Accessed 30 October 2020].

[3] TensorFlow. 2020. Tensorflow Lite For Microcontrollers. [online] Available at: <https://www.tensorflow.org/lite/microcontrollers> [Accessed 31 October 2020].

[4] Ltd., A., 2020. Cortex-M – Arm Developer. [online] Arm Developer. Available at: <https://developer.arm.com/ip-products/processors/cortex-m> [Accessed 31 October 2020].

[5] Mathworks.com. 2020. What Is Reinforcement Learning?- MATLAB & Simulink. [online] Available at: <https://www.mathworks.com/help/reinforcement-learning/ug/what-is-reinforcement-learning.html> [Accessed 31 October 2020].

[6] DigiKey. 2020. Tinyml: Getting Started With STM32 X-CUBE-AI. [online] Available at: <https://www.digikey.com/en/maker/projects/tinyml-getting-started-with-stm32-x-cube-ai/f94e1c8bfc1e4b6291d0f672d780d2c0> [Accessed 31 October 2020].

[7] Sakr, F., Bellotti, F., Berta, R. and De Gloria, A., 2020. Machine Learning on Mainstream Microcontrollers. Sensors, 20(9), p.2638.

[8] Lonza, A., 2019. Reinforcement Learning Algorithms With Python. Birmingham: Packt Publishing, Limited.

[9] St.com. 2020. X-Cube.AI User Manual: Getting started with X-CUBE-AI Expansion Package for Artificial Intelligence (AI). [online] Available at: <https://www.st.com/en/embedded-software/x-cube-ai.html#documentation> [Accessed 31 October 2020].

[10] Mordorintelligence.com. 2020. Gesture Recognition Market - Growth, Trends and Forecast (2020 - 2025) [online] Available at: <https://www.mordorintelligence.com/industry-reports/gesture-recognition-changing-the-way-humans-interact-with-devices-industry> [Accessed 2 November 2020].

Appendices

Appendix A

Augmented Figures

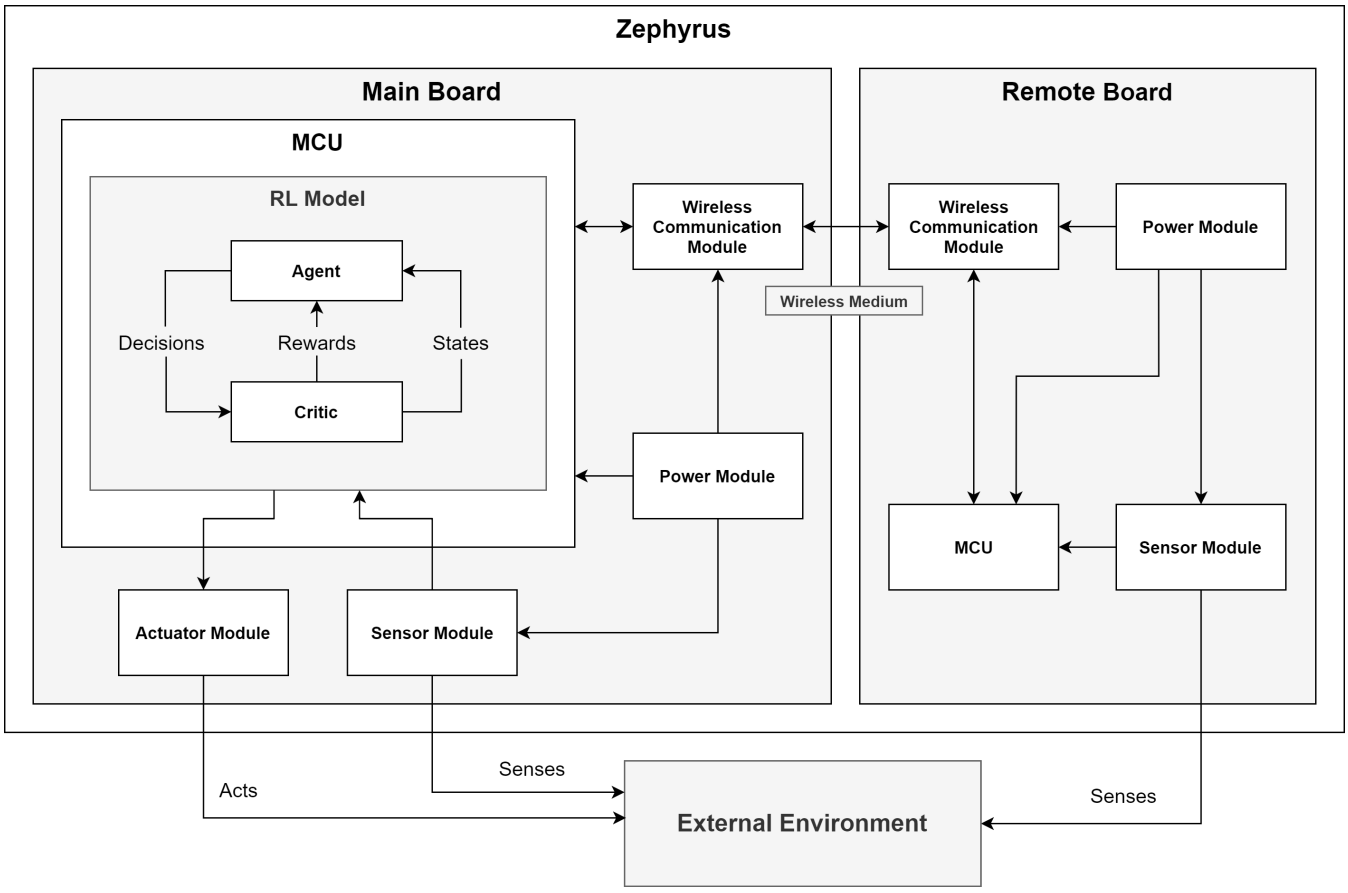

Figure A.1: System Overview Diagram Augmented

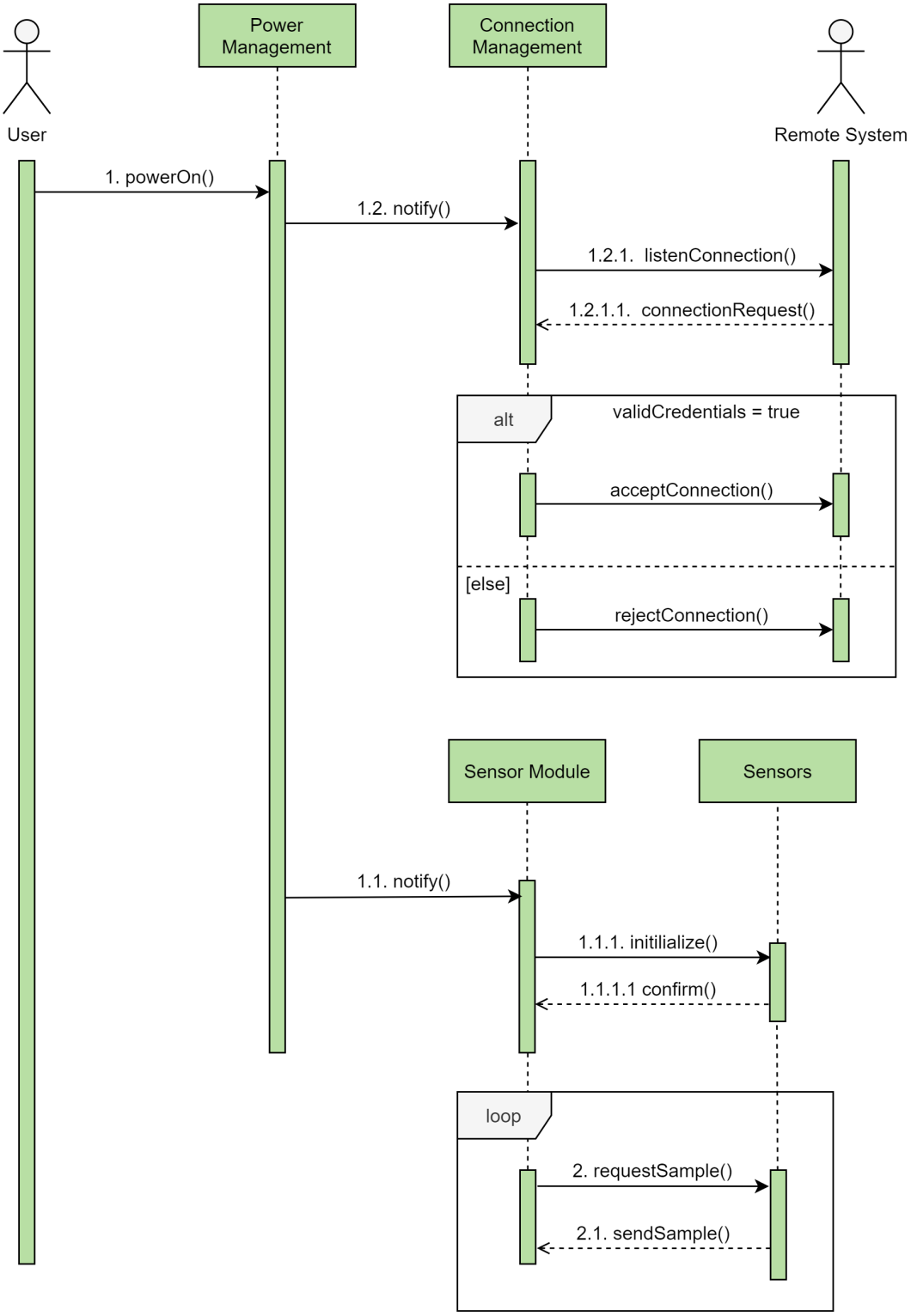

Figure A.2: Local System Sequence Diagram Augmented

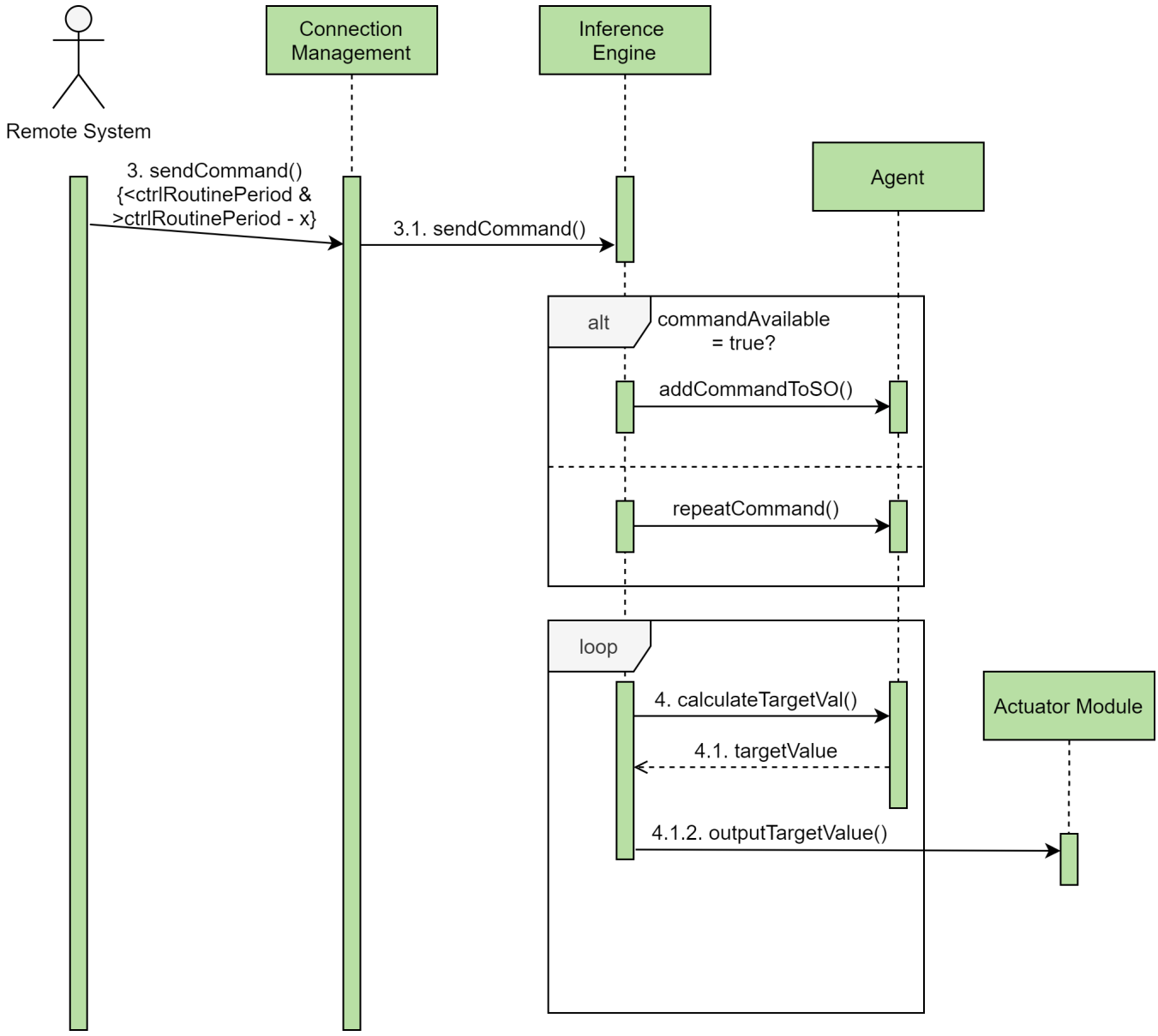

Figure A.3: Local System Sequence Diagram Augmented

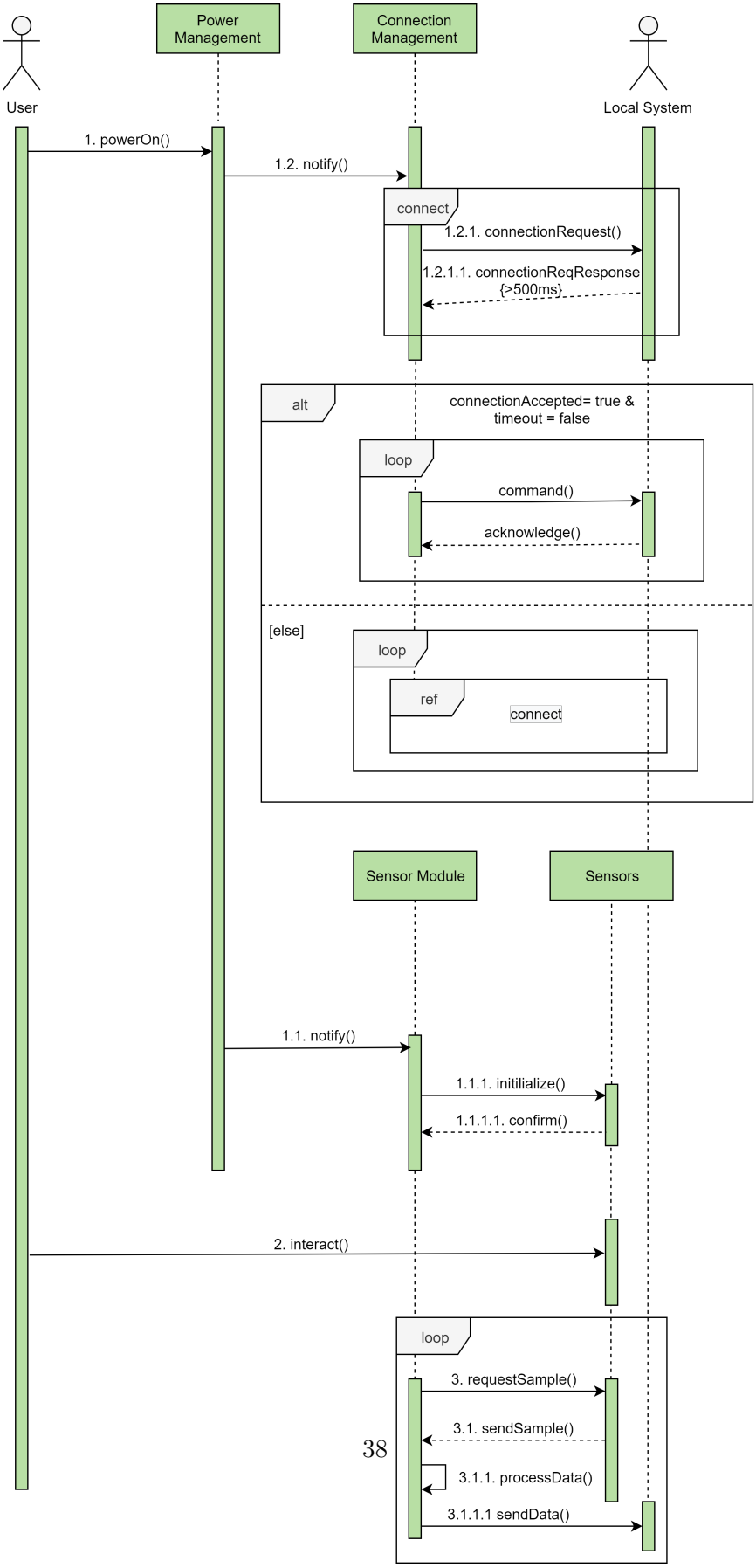

Figure A.4: Remote System Sequence Diagram Augmented

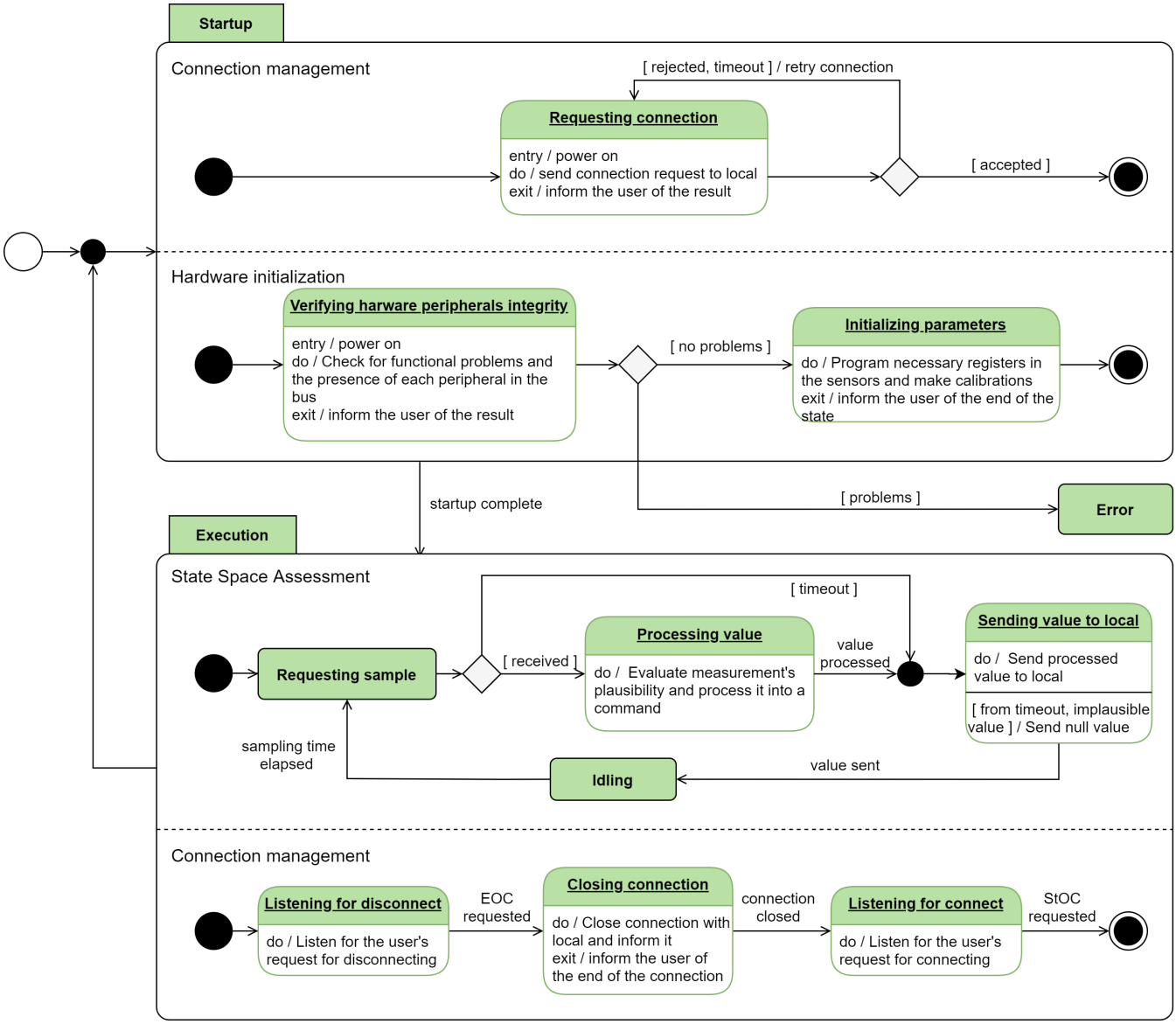

Figure A.5: State Machine Diagram - Remote Augmented

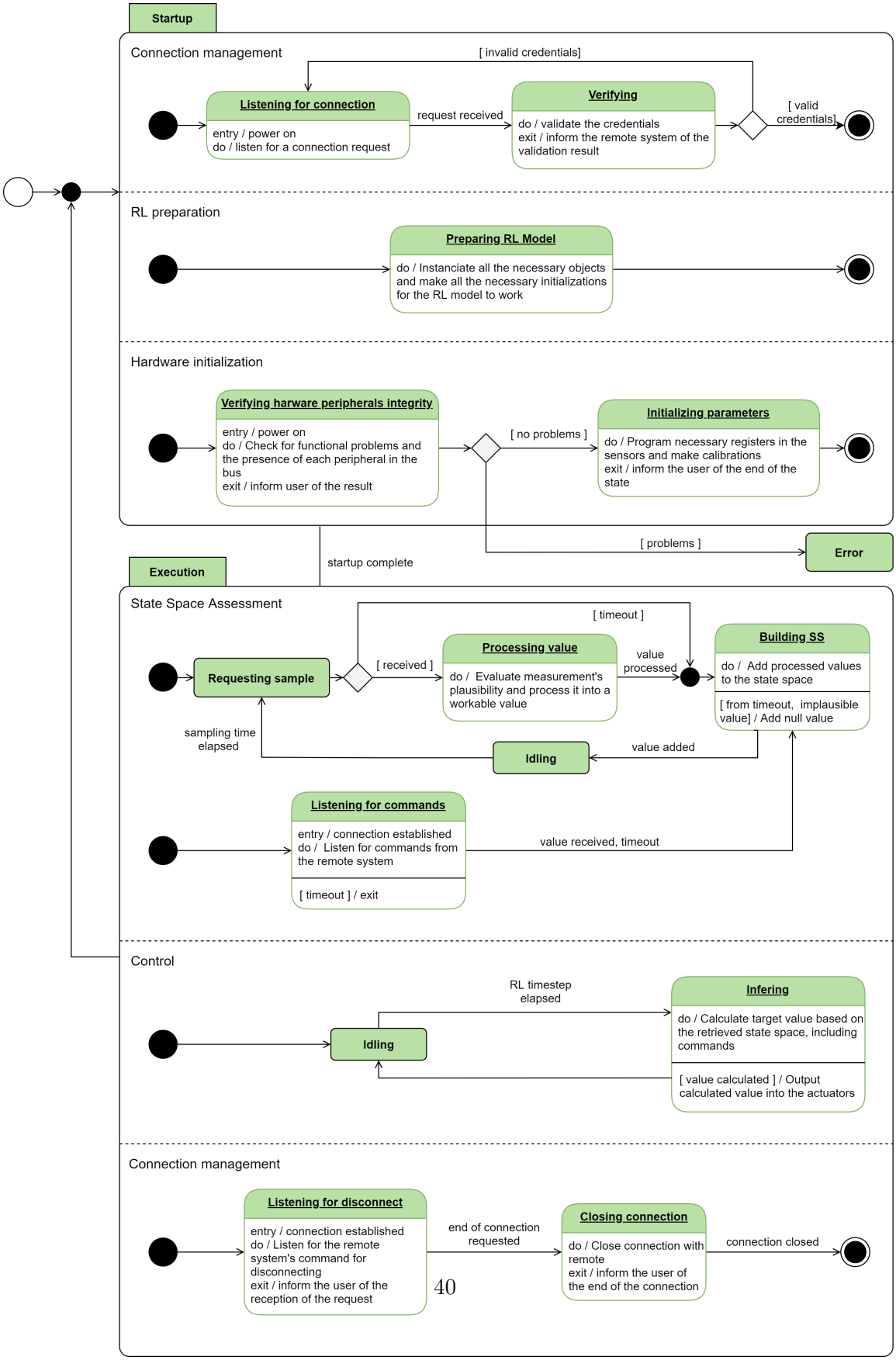

Figure A.6: State Machine Diagram - Local Augmented