Enabling remote computation for soccer robots using 5G mm-waves

Technische Universiteit
Eindhoven
University of Technology

**/ Department of
Mathematics and
Computer Science
/ PDEng Mechatronic
Systems Design**

Enabling Remote Computation for Soccer Robots using 5G mm-Waves

Design, development and validation of a robotic test bench for 5G mm-wave demonstrations

October 2019

Aditya Gurudatt Kamath

Enabling Remote Computation for Soccer Robots using 5G mm-Waves

Design, development and validation of a robotic test-bench for 5G mm-wave demonstrations

Aditya Gurudatt Kamath

October 2019

Eindhoven University of Technology

PDEng Report: **2019/088**

Partners

blueSPACE 5G Technologies

Eindhoven University of Technology

Steering Group

Prof. dr. ir. Idelfonso Tafur Monroy
Dr. Simon Rommel
Ir. Jesse Scholtes
Dr. ir. René van de Molengraft

Date

October 2019

Composition of the Thesis Evaluation Committee:

Chair:	Dr. ir. René van de Molengraft
Members:	Prof. dr. ir. Idelfonso Tafur Monroy
	Dr. Simon Rommel
	Ir. Jesse Scholtes
	Dr. Alessandro Saccon
	Dr. Peter Heuberger

The design that is described in this report has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Date
October, 2019

Contact address
Eindhoven University of Technology
Department of Mathematics and Computer Science
MF 5.072
P.O. Box 513
NL-5600 MB
Eindhoven, The Netherlands
+31 402743908

Published by
Eindhoven University of Technology
Stan Ackermans Institute

PDEng Report
2019/088

Abstract
5G is the next generation of cellular mobile communications and promises lower latencies and higher data rates than ever before. This creates many opportunities for applications in the fields of self-driving vehicles, autonomous robotics and Industry 4.0. This project uses the soccer robots of TU Eindhoven's world champion RoboCup team – Tech United – to demonstrate one such application. The project involves the validation of blueSPACE's 5G performance KPIs and the robot's timing requirements when equipped with 5G mm-wave communication hardware.

Keywords
5G, mm-Waves, Robotics, Cloud Robotics, Industry 4.0, RoboCup MSL

Preferred reference
Enabling Remote Computation for Soccer Robots using 5G mm-Waves: Design, development and validation of a robotic test bench for 5G mm-wave demonstrations. Eindhoven University of Technology, PDEng Report, October 2019 (2019/088)

Partnership
This project was supported by Eindhoven University of Technology and blueSPACE. blueSPACE has received funding from the European Union's Horizon2020 research and innovation programme under grant agreement No 762055.

European Commission
Horizon 2020
European Union funding
for Research & Innovation

Disclaimer Endorsement

Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the Eindhoven University of Technology and Company name. The views and opinions of authors expressed herein do not necessarily state or reflect those of the Eindhoven University of Technology and blueSPACE, and shall not be used for advertising or product endorsement purposes.

Disclaimer Liability

While every effort will be made to ensure that the information contained within this report is accurate and up to date, Eindhoven University of Technology makes no warranty, representation or undertaking whether expressed or implied, nor does it assume any legal liability, whether direct or indirect, or responsibility for the accuracy, completeness, or usefulness of any information.

Trademarks

Product and company names mentioned herein may be trademarks and/or service marks of their respective owners. We use these names without any particular endorsement or with the intent to infringe the copyright of the respective owners.

Copyright

Foreword

5G and the industry 4.0 with smart and interconnected robotics and automation systems are set to become the next revolutions to personal and professional lives. Yet, at a time they are very much under research, how does one connect the latest telecommunications research with an established experimental robotics platform? How can the value of 5G to robotics be proven, despite 5G still being under research and development and no commercial being hardware available? This is the question that underlies this work, the challenge Aditya set out to find an answer to.

Combining the robotics platform of TU/e's RoboCup team with the latest research on mm-wave 5G from the blueSPACE project, Aditya undertook the task to bridge the gap and learn the languages of both robotics and telecommunications. Faced with the difficulties of relying on the outputs of a research project, many hurdles had to be overcome, before a demonstration system could be implemented. None the less, Aditya successfully implemented a demonstration system, showing what value 5G can have for robotics – and along the way contributing to everyone's understanding of the challenges faced and processes to be completed to combine the involved fields!

Dr. Simon Rommel
Project Supervisor - blueSPACE
18th October 2019

5G communication is envisioned to become an important enabler in robotics technology as ultra-hi bandwidth and ultra-low latency specifications of remote data communication will offer a huge opportunity for offloading robotic computations while maintaining real-time performance. The autonomous robot soccer case offers a great benchmark problem as it combines real-time communication of 60 fps camera streams with low-latency real-time control data at a 1 kHz update rate.

Aditya delivered an important first functional proof of concept in showing that one of Tech United's acclaimed soccer robots could actually offload its computations entirely, communicating camera as well as motion data via a 5G-enabled communication stack while preserving its original behavior.

Dr. ir. René van de Molengraft
Technical Director, Tech United
15th October 2019

Preface

This report is the final deliverable of the graduation project for a Professional Doctorate in Engineering in Mechatronic Systems Design (PDEng MSD). This graduation project involves the design and implementation of a functional robotic test bench using 5G hardware and a mobile robot platform. The intent of this report is to enable a technical reader (engineers and researchers) to either understand, replicate or continue this project and develop a fully functional multi-robot demonstrator using 5G.

For the reader specializing in robotics, this report provides a brief background in 5G and the network architecture developed by blueSPACE, the key stakeholders in this project.

For the reader specializing in telecommunications and 5G, this report provides a background into Tech United, a robotics team from TU Eindhoven and their acclaimed TURTLE platform.

For the readers continuing this project (either from blueSPACE or Tech United), this report proposes a multi-robot design, highlights the steps needed to replicate the implemented setup with one robot and the steps towards implementing the complete proposed design with a team of robots.

Finally, for all other readers, this report intends to show the first step towards implementing an industrial multi-robot application using 5G and highlights the relevance and importance of 5G in such a scenario. With 5G technologies on the verge of being commercialized, this report is extremely relevant for the field of robotics, especially in the smart industries of the future.

Ir. Aditya Kamath
10th October 2019

Acknowledgements

This project could not be completed without support, feedback and assistance from various people over the last ten months. First, I would like to thank my supervisors Simon Rommel, Jesse Scholtes and and René van de Molengraft, whose thoughts and feedback were valuable in the formulation of this design assignment. You were always ready to meet and discuss my ideas and provided me with direction and insight, for which I am very grateful.

5G and mobile communication was a new subject for me and my colleagues at blueSPACE were instrumental in helping me get started. For this, I have thank Dimitris Konstantinou and Bruno Cimoli, who provided me with literature and reading material, and were around to answer any of my questions. I also thank Denys Nyzovets, for his wonderful collaboration and for providing me with the tools to get the proposed setup up and running. I also have to thank the members from blueSPACE responsible for implementing the final test setup, which helped me validate my assumptions and successfully conclude this project.

This project could not have been completed without the support of Tech United. Firstly, I must congratulate the entire team for their excellent performances in the tournaments they participated in during this project. Secondly, I must acknowledge them for the awesome robots they build, the meticulous software and documentation, and finally for the instant feedback, tips and suggestions from team members during the weekly meetings. I especially thank Wouter Kuijpers, Wouter Houtman, Jordy Senden and Harrie van de Loo for sharing their experiences and challenges with the Turtle robot and providing direction to this project. I must single out Wouter Kuijpers, who took time out to give me detailed introductions to the hardware and software and was there to help out whenever I was stuck.

In addition, I must thank Peter Heuberger and Ellen van Hoof-Rompen for their continued support over the last two years. Finally, there are my friends and colleagues from the PDEng ASD/MSD program - especially Sabyasachi Neogi and Siddharth Khalate, who were around for support and to deliberate over our problems and findings, and also for the relaxing coffee breaks and walks at lunch.

10-10-2019

Executive Summary

5G is a set of technologies that define the 5th generation of cellular mobile communications and succeeds the 4G cellular network that we currently use. While on the verge of being commercialized, 5G aims at providing higher data rates, reduced latency, energy/cost saving, higher system capacity, and massive device connectivity. These technologies are being developed and realized by multiple companies and research initiatives around the world. One such initiative is the blueSPACE consortium, which is led by TU/e and is working towards developing technologies to improve the currently planned global 5G infrastructure.

blueSPACE intends to validate the hardware they are developing by demonstrating them in different use cases and by comparing the results with pre-defined performance indicators. One such use case is a smart factory setting, with multiple moving robots, communicating and collaborating over 5G. An analogous use case is of Tech United, a soccer team consisting of five robots that collaborate with each other and play soccer autonomously.

This project involves a collaboration with Tech United, in which their robots, known as Turtles are used to design and implement a 5G based application and to validate the blueSPACE KPIs, which in this case is Latency. The proposed design enables the Turtles to stream all their sensor and camera data over 5G mm-waves to a remote server and compute them remotely. The corresponding actuation commands are relayed back to the robot over 5G.

This project explains the requirements of this system, alongside an architecture description using the CAFCR methodology. In addition, this project also proposes a validation methodology to measure and analyze the performance of the network and the robot system. Finally, this design and validation methodology is partially proven using a functional proof-of-concept developed using one Turtle robot and a wireless network developed by blueSPACE.

The measured system performance validates the blueSPACE KPI of Latency. The observed latency of the network, for the implemented application is observed to be well within the maximum allowable limit provided by Tech United. This proves that the robot's computing can be successfully offloaded to a remote computer, while meeting all its original requirements. This would allow Tech United to replace the Turtle on-board computers with a less powerful alternative, hence reducing costs, the weight of the Turtle and also the power consumption. A powerful central processing server would also allow the team to develop computationally expensive algorithms, which would help the soccer robots to make quicker and better decisions during matches.

In conclusion, this project is a first step towards implementing a multi-robot demonstrator and proves that such an application is feasible and advantageous to all involved stakeholders.

Glossary

General Terminologies

TU/e	Eindhoven University of Technology
MSD	Mechatronic Systems Design
PDEng	Professional Doctorate in Engineering
5G	5th Generation of Mobile Communications
5G PPP	5G Infrastructure Public Private Partnership
MSL	Mid-Sized League
TURTLE	Tech United Robocup Team: Limited Edition
ICT	Information and Communications Technology
SME	Small and Medium-Sized Enterprise
HTSC	High Tech Systems Center
CAFCR	Customer Objective, Application, Functional, Conceptual, Realization
PID	Project Initiation Document
PSG	Project Steering Group
COTS	Custom Off-The-Shelf

Telecommunication Terminologies

4G LTE	4G Long Term Evolution
5G NR	5G New Radio
KPI	Key Performance Indicator
eMMB	Enhanced Mobile Broadband
URLLC	Ultra-Reliable Low Latency Communication
mMTC	Massive Machine Type Communication
IMT	International Mobile Telecommunications
RRH	Remote Radio Head
PHY	Physical Layer (Layer 1 of the OSI model)
E2E	End-to-End (latency)
BBU	Base Band Unit
RRU	Remote Radio Unit
RF	Radio Frequencies

SDM	Space Division Multiplexing
CO	Central Office
RAN	Radio Access Network
MCF	Multicore Fibers
OFDM	Orthogonal Frequency Division Multiplexing
SMF	Single Mode Fiber
RN	Remote Node
VNF	Virtualized Network Function
UE	User Equipment
gNB	Next Generation Node-B
eNB	Evolved Node-B
C-RAN	Cloud RAN or Centralized RAN
OSI	Open Systems Interconnect
OAI	OpenAirInterface or OpenAirInterface5G
EPC	Evolved Packet Core
CN	Core Network
SDR	Software Defined Radio
NTP	Network Time Protocol
PTP	Precision Time Protocol
RT	Round Trip (Latency)
NIC	Network Interface Card
FFT	Fast Fourier Transform

Robotics Terminologies

IMU	Inertial Measurement Unit
OS	Operating System
GigE	Gigabit Ethernet (IEEE 802.3 1000baseT)
EtherCAT	Ethernet for Control Automation Technology
Fps	Frames per second
RTDB	Real-Time Database
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
YUV	Color encoding format
RGB	Color encoding format using Red, Green and Blue filters
MSL	(RoboCup) Mid Sized League
TRC	Turtle Remote Control

List of Tables

List of Figures

Contents

1 Introduction

This chapter describes the project context, the problem description and the scope, and highlights the concerns of each stakeholder involved in the project. The contents of this chapter were defined at the beginning of this project and documented in the Project Initiation Document (PID) in Appendix C.1.

1.1 Project Background

5G is a set of technologies that define the 5th generation of cellular mobile communications and succeeds the 4G cellular network that we currently use. Although still under development, 5G aims at providing higher data rates, reduced latency, energy/cost saving, higher system capacity, and massive device connectivity [1].

The development of 5G technologies is not only led by companies like Qualcomm, Huawei or Ericsson, but also by initiatives like the 5G PPP (5G Infrastructure Public Private Partnership). 5G PPP is a joint initiative between the European Commission and the European ICT industry (ICT manufacturers, telecommunications operators, service providers, SMEs and research institutions) [2]. In June 2017, 21 projects were launched under the second phase of the 5G PPP - one of them being blueSPACE, a three-year research project led by the Eindhoven University of Technology [3]. This project is titled as follows:

> **blueSPACE: Building on the Use of Spatial Multiplexing 5G Network Infrastructures and Showcasing Advanced Technologies and Networking Capabilities** [4]

blueSPACE is a consortium that consists of fifteen partners from the European Union and aims at developing optical technologies to support next-generation wireless/5G NR technologies to allow high speeds, reduced latency and reduced power consumption. While the main objective of this project is to develop the infrastructure to support the next generation 5G communications, other objectives include the development of demonstrators to showcase future-oriented use cases of 5G in multiple scenarios. [3]

1.2 Problem Description and Scope

1.2.1 Problem Description

The goal of this PDEng final assignment is to conceptualize, design and develop a demonstrator to showcase an Industry 4.0 [5] application using a 5G network architecture defined in the blueSPACE project. The application is focused towards collaborative teams of robots in accordance with the blueSPACE Experimental Evaluation and Demonstration plans [6]. The project also aims at validating the key performance indicators for a 5G low-latency communication interface in this demonstrator.

One such collaborative team of robots is the Tech United MSL team [7] at TU Eindhoven. These soccer playing robots, called Turtles [8] participate annually in the RoboCup Mid-Sized League [9] tournament and are currently the World Champions [10].

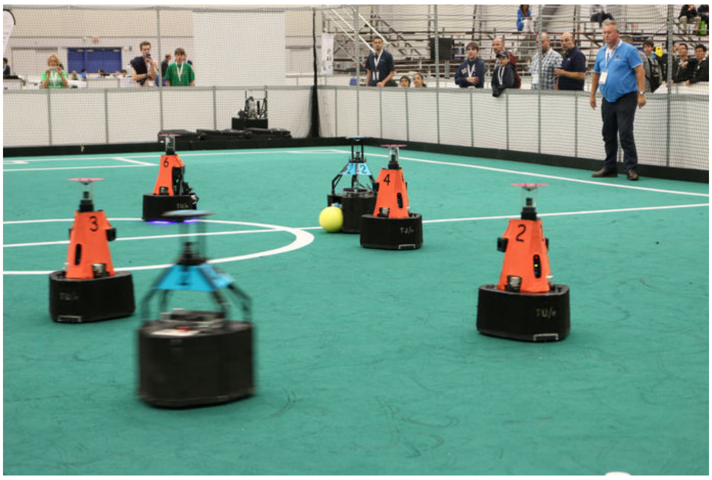


Figure 1.1: The Tech United Turtles (in orange) during a RoboCup MSL match in 2018 [9]

This project involves collaborating with Tech United to demonstrate a 5G application that can add value and also enhance the performance of these soccer robots. After consultation with key stakeholders the use case for the project was proposed and defined as follows:

Completely Decoupled Computation:

- Using the Tech United robot platform (TURTLE), all image and sensor data from the robot are sent to a remote computer over 5G mm-waves.
- The remote computer executes the Turtle's software to process this information and returns the corresponding actuation commands back to the robot.
- The network performance and the Turtle's performance is measured, analyzed and compared with the requirements (blueSPACE KPIs and the Turtle's timing requirements).

The definition of the above use case is a result of a requirements elicitation process involving all stakeholders, and will be described in detail in upcoming Chapter 4.

1.2.2 Project Scope

This section describes the scope of the project. The scope is divided into two categories: Tasks and Delimitations. These describe the tasks involved in the project and the tasks explicitly defined as out-of-scope. The following tasks and delimitations were mutually agreed upon by the trainee and all stakeholders.

Tasks:

- Survey of 5G technologies and future trends of 5G technologies in Industry 4.0 use cases
- A detailed survey of the Tech United robot soccer team and their Turtle platform [8] - including data-flow within the robots, communication with other robots and their corresponding timing/-data size requirements. This is followed by the proposal and definition of a specific use case.
- Derivation of requirements and specifications for the communication of the Turtles over a 5G network and for the interfacing of the 5G hardware with the robot hardware.
- Design and implementation of the proposed application using blueSPACE 5G hardware and the Tech United Turtle software on either a complete robot or a stationary test rig.
- Survey and definition of measurement and evaluation plans to validate blueSPACE and Tech United KPIs and requirements
- Demonstration, verification and validation of the implemented application
- Technical documentation (reports, manuals), presentations and frequent reporting

Delimitations:

- The design and development of the communication hardware/setup will be handled by the blueSPACE team, with recommendations from the trainee.
- Development of additional functionality for the Tech United robots is out of scope. Only the communication aspect will be modified according to the requirements of the project.

1.3 Stakeholders and Concerns

This section introduces the key stakeholders in this PDEng project and highlights their concerns. The stakeholder concerns and a detailed stakeholder analysis is documented in the Stakeholder Analysis Document (Appendix C.2). The concerns in the following sections are a basis for the high-level requirements of this project, which are described in the upcoming chapters.

1.3.1 blueSPACE

blueSPACE [3] are the key stakeholders and the customers in this PDEng assignment. This project is led by prof.dr.ir. Idelfonso Tafur Monroy and supervised by dr. Simon Rommel with assistance from multiple PhD candidates in the team. As explained earlier, blueSPACE is one of the projects under 5G

PPP, a joint initiative between the European Commission, the European ICT industry and universities. blueSPACE is a 3 year project led by a team at Eindhoven University of Technology with multiple industry and university partners around the EU.

The main concerns for these stakeholders are as follows:

- The demonstration of a 5G use case for a real-life robotic scenario
- The validation of pre-determined performance KPIs of the 5G network
- A test-bench for 5G hardware (both off-the-shelf and custom developed hardware)

1.3.2 Tech United

The Tech United MSL team [7] are also stakeholders in this project. Over the years, Tech United has developed a team of autonomous robots that work collaboratively to play soccer in the RoboCup Mid-Sized League. The team are currently world champions (RoboCup 2019) [10] and have consistently ranked first or second over the last few years. The team is constantly improving and upgrading their robots and is working towards being able to play against a human team in the future. Their robots are called Turtles and are used in this assignment. The key stakeholder from Tech United is dr.ir. René van de Molengraft and experimentation with the Turtles is guided by ir. Wouter Kuijpers with feedback and assistance from other members of the team.

Their concerns are as follows:

- To increase computational power on-board the Turtles
- To improve the communication between the Turtles
- To improve the performance of the team of Turtles as a whole
- The demonstration of a 5G use case using the Turtle platform

1.3.3 Other Stakeholders

Besides, blueSPACE and Tech United, there are other stakeholders with interests in this project.

- **PDEng MSD Program:** Supervised by dr.ir. Peter Heuberger (Program Manager, PDEng MSD/ASD), the PDEng program is concerned with successful completion of the PDEng final assignment
- **High Tech Systems Center (HTSC):** Supervised by ir. Jesse Scholtes (HTSC, TU/e Robotics Lab), they are concerned with improving the level of research at TU Eindhoven and attracting industrial funding or collaborations
- **Telecom/Robotics Researchers:** This project presents a 5G mm-waves based robotic application, which can be used by telecommunication engineers to study mobile networks in industrial or robotic scenarios, and by robotics engineers to study topics like cloud robotics

2 Methodology

The methodology involves the definition and planning of the project, definition of the system architecture, design, implementation and finally, validation of the final solution. The upcoming list describes the approach used in each of the following stages.

- **Definition and Planning:** This involves the definition of high-level requirements, use case, scope and resources required during the project. The project definition is signed off by the stakeholders. The timeline of the project is also documented and managed in a Gantt chart.
- **System Architecture:** Once the project and its scope are defined, the CAFCR methodology [11] is used to decompose the architecture description into five views: Customer Objectives, Application, Functional, Conceptual and Realization views. In each category, visual and functional models are used to communicate specifications and design choices.
- **Design and Implementation:** The V-Model [12] is used to design, develop and test the various views from the CAFCR study. Tasks are broken down and managed using the Gantt chart developed in the first stage.
- **Validation:** During the architecture description phase, a measurement and validation plan is defined in different levels of the CAFCR methodology. During the implementation phase, an experiment is designed, setup, measured and analyzed. These validation plans, experimentation and results are described in later chapters.

This methodology, along with details about resources, organization of meetings, deliverables and other project management concerns were defined at the beginning of the project and is documented in the Project Initiation Document (PID, Appendix C.1).

2.1 Work Phase Plan

In order to make the timeline easier to manage, the above four stages were divided into six work phases: Definition, Design, Preparation, Realization, Validation and Follow-Up. The involved tasks and the expected outcomes of each phase is documented in the Project Initiation Document in Appendix C.1. The following list shows the upcoming chapters and which work phase they correspond to:

- **Ch 3 - Literature Study:** Definition phase

- **Ch 4 - Requirements and Use Case Definition:** Definition phase
- **Ch 5 - Architecture and Design Description:** Design phase
- **Ch 6 - System and Subsystem Implementation:** Preparation phase and Realization phase
- **Ch 7 - Validation and Results:** Validation phase
- **Ch 8 - Conclusions and Recommendations:** Follow-Up phase

2.2 CAFCR and the V-Model

CAFCR is an architecting method (Figure 2.1) for the design and development of complicated systems. This method can be used to decompose a problem into multiple views and is especially useful when the problem is open-ended or not clearly defined, like this project. CAFCR stands for Customer Objectives, Application View, Functional View, Conceptual View and Realization View - a set of frameworks/diagrams/models that help explain both the problem and the potential solutions in a well articulated way.

Customer Objectives (**What** does the customer want to achieve?) and the Application View (**How** does the customer realize their goals?) capture the needs of the customers. This provides the justification (**Why**) for the design choices made during the project. The Functional, Conceptual and Realization views describe the the **What** and the **How** of the product. The Conceptual view is a conceptual picture of the proposed design while the Realization view delves into the specifics of how the design and its interfaces are implemented. This split is essential to maintain the stability of the architecture - while the conceptual view is maintained over a longer time, the realization view can change quite quickly. The CAFCR concept is explained in detail in the thesis titled "CAFCR: A Multi-view Method for Embedded Systems Architecting; Balancing Genericity and Specificity" by Gerrit Muller [11] [13].

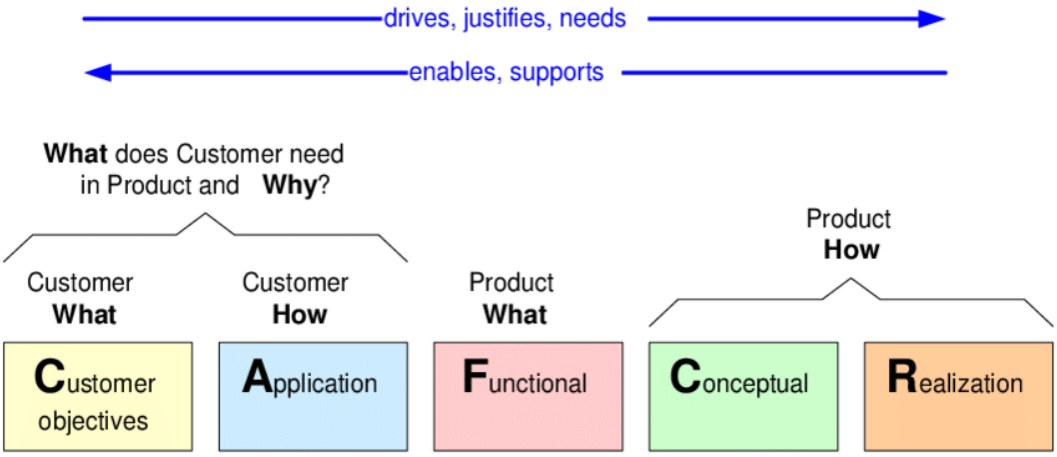


Figure 2.1: The different views of the CAFCR model [13]

The different views of CAFCR are explained below:

- **Customer Objective View:** Requirements, key drivers and other customer objectives

- **Application View:** Stakeholders and concerns, context diagrams, use cases/scenarios
- **Functional View:** Diagrammatic representation/map of technical functionality
- **Conceptual View:** Detailed functional decomposition describing internal interfaces, issues like safety/reliability of functional units, integration plan, risks/uncertainties
- **Realization View:** Bench-marking, performance analysis, safety/reliability analysis

Once the architecture is defined using the CAFCR methodology, the design, implementation and validation is planned according to the V-Model [14]. The V-Model is a systems engineering process that describes the timeline of a project from the literature/background survey to the validation, documentation and maintenance of the developed system. The horizontal axis describes the time and the vertical axis describes the level of abstraction.

It was noticed that the CAFCR views could be abstracted onto the same vertical axis of the V-model and the work phase plan could be plotted on the same horizontal axis. Therefore, a combined diagram involving the work phase plan, the V-model and the CAFCR views could be created (Figure 2.2) to describe the complete methodology used in this project. The implementation of this modified V-Model is described in Chapter 5 and Chapter 6.

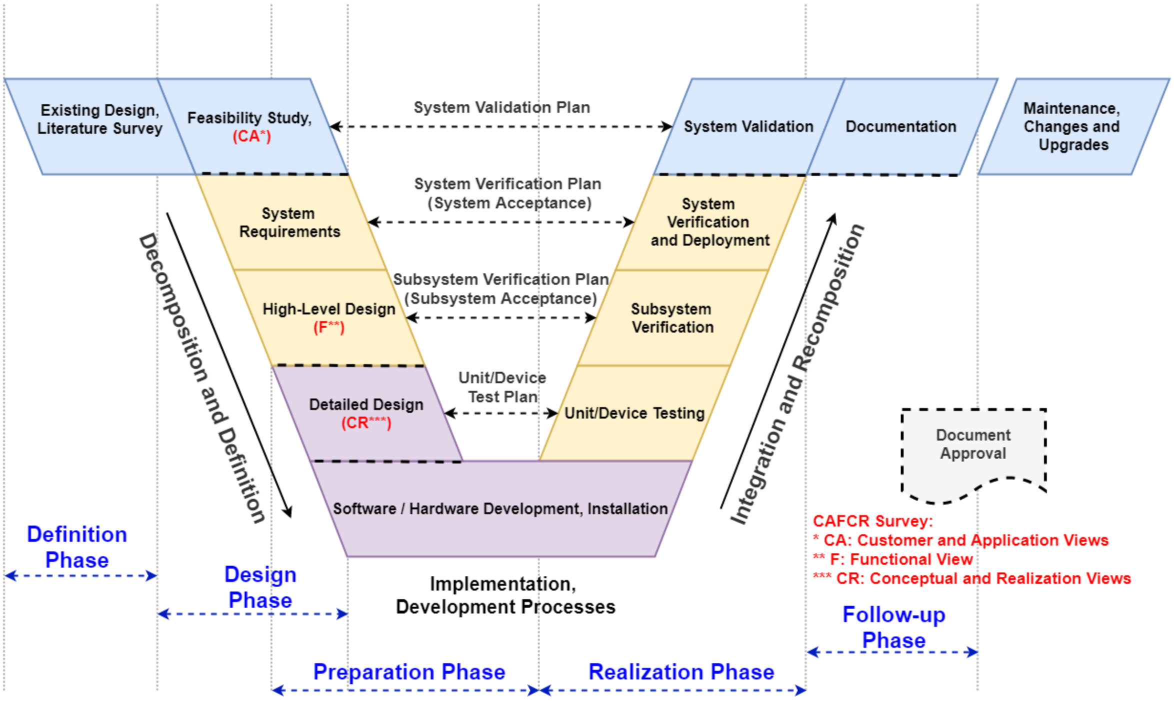

Existing Design, Literature Survey
Feasibility Study, (CA*)
System Validation Plan
System Validation
Documentation
Maintenance, Changes and Upgrades
System Requirements
System Verification Plan (System Acceptance)
System Verification and Deployment
High-Level Design (F**)
Subsystem Verification Plan (Subsystem Acceptance)
Subsystem Verification
Detailed Design (CR***)
Unit/Device Test Plan
Unit/Device Testing
Software / Hardware Development, Installation
Decomposition and Definition
Integration and Recomposition
Document Approval
CAFCR Survey:
* CA: Customer and Application Views
** F: Functional View
*** CR: Conceptual and Realization Views
Definition Phase
Design Phase
Implementation, Development Processes
Follow-up Phase
Preparation Phase
Realization Phase

Figure 2.2: Modified V-Model workflow that also represents the work-phase plan and the CAFCR views. An additional layer of approval is included between abstraction layers and test/validation plans are included within each abstraction layer.

2.3 Project Timeline

The timeline was proposed in the Project Initiation Document (Appendix C.1) during the definition phase of the project. The timeline describes each of the phases in the work phase plan alongside an additional buffer period and time set apart for holidays. Finally, milestones are defined for the project. However, due to technical issues with hardware and software implementation, the resulting timeline of the project varies from the timeline in the PID. More about this is explained in Chapter 6. The final timeline followed during this project is shown below in Figure 2.3.

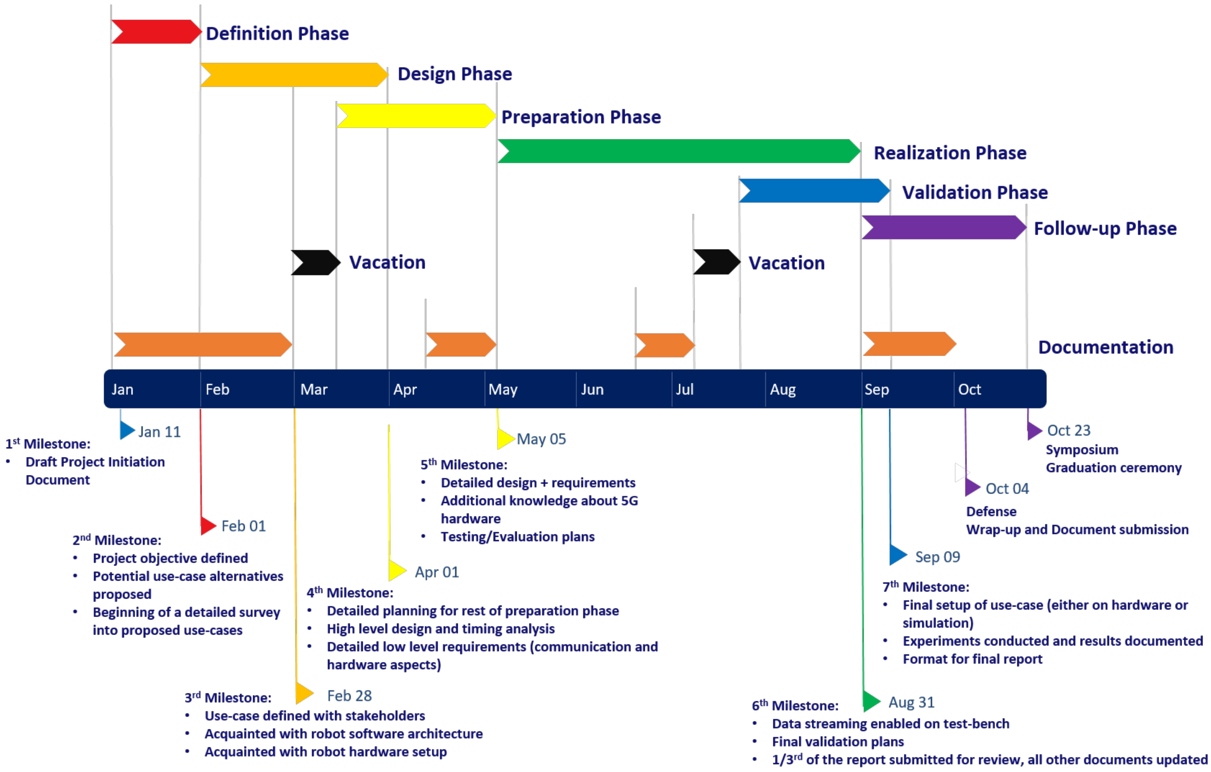

Definition Phase
Design Phase
Preparation Phase
Realization Phase
Validation Phase
Follow-up Phase
Vacation
Vacation
Documentation
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Jan 11
1st Milestone:
• Draft Project Initiation Document
Feb 01
2nd Milestone:
• Project objective defined
• Potential use-case alternatives proposed
• Beginning of a detailed survey into proposed use-cases
Feb 28
3rd Milestone:
• Use-case defined with stakeholders
• Acquainted with robot software architecture
• Acquainted with robot hardware setup
Apr 01
4th Milestone:
• Detailed planning for rest of preparation phase
• High level design and timing analysis
• Detailed low level requirements (communication and hardware aspects)
May 05
5th Milestone:
• Detailed design + requirements
• Additional knowledge about 5G hardware
• Testing/Evaluation plans
Aug 31
6th Milestone:
• Data streaming enabled on test-bench
• Final validation plans
• 1/3rd of the report submitted for review, all other documents updated
Sep 09
7th Milestone:
• Final setup of use-case (either on hardware or simulation)
• Experiments conducted and results documented
• Format for final report
Oct 04
Defense
Wrap-up and Document submission
Oct 23
Symposium
Graduation ceremony

Figure 2.3: The final timeline followed during the project. This timeline varies from the proposed timeline in Appendix C.1 due to technical issues in the Realization and Validation phases.

3 Literature Study

This chapter summarizes the initial survey of blueSPACE and Tech United related literature. The first section provides an introduction to 5G in general and describes the technologies and architectures used by blueSPACE. The second section describes the technologies and methods used by Tech United in their Turtle robot platform. In addition to this, each section also highlights the challenges and constraints of blueSPACE and Tech United respectively.

3.1 5G and blueSPACE

5G is the fifth generation of cellular network technology. It follows 2G, 3G and 4G and promises higher data rates, reduced latency, higher system capacity and massive device connectivity. 5G mm-Wave is a set of technologies just like 4G LTE but uses a different band of frequency to share data.

It is important to understand how 5G differs from 4G LTE networks in use today. Unlike LTE, 5G operates on three different bands of frequencies - the Low-band spectrum (< 6 GHz), the Mid-band spectrum (6 - 24 GHz) and the High-band spectrum (24 - 100 GHz). The high-band spectrum, due to its high frequencies has an extremely short wavelength and is hence known commonly as mm-Wave.

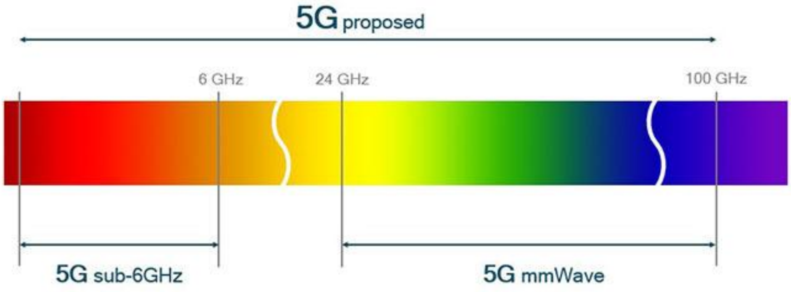


Figure 3.1: 5G spectrum showing the mm-Wave frequency band between 24 and 100 GHz. [15]

There are multiple advantages of using this spectrum for communication. Firstly, this section of the spectrum is currently unused unlike lower bands that are congested with TV, radio and 4G LTE devices (800 - 3000 MHz). So, 5G mm-Wave will have a significantly higher amount of free bandwidth available. This means that 5G networks could accommodate a lot more devices than existing 4G LTE networks.

Secondly, due to its shorter wavelength, faster data rates could be achieved, but at the cost of coverage area. This would required base stations that cover smaller geographic areas in comparison to 4G LTE base stations. A high density of such "small cells" could not only accommodate a lot more users but would also provide higher data rates. Due to these impressive features, 5G can not only be used to connect smartphones and laptops but also a wide range of devices. In general, the possible services offered by 5G can be divided into the three categories below:

- **eMMB: Enhanced Mobile Broadband** - 5G services for mobile handsets
- **URLLC: Ultra-Reliable Low-Latency Communications** - Used in applications such as industrial networked control, robotics and self-driving cars
- **mMTC: Massive Machine Type Communications** - Used in applications such as sensor networks or Internet-of-Things

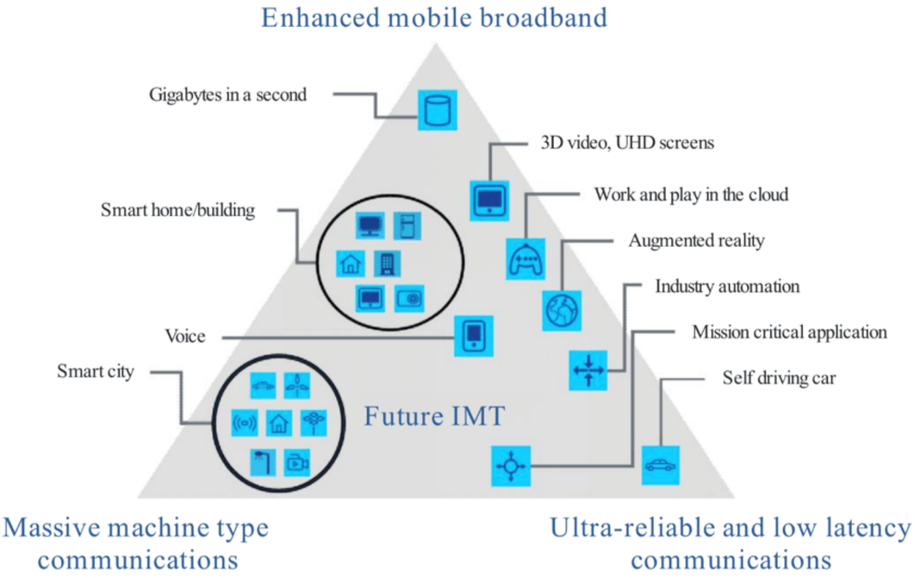

Enhanced mobile broadband
Gigabytes in a second
3D video, UHD screens
Smart home/building
Work and play in the cloud
Augmented reality
Industry automation
Voice
Mission critical application
Smart city
Self driving car
Future IMT
Massive machine type communications
Ultra-reliable and low latency communications

Figure 3.2: A framework for understanding 5G use cases within the three categories - eMMB, mMTC and URLLC, for the future development of International Mobile Telecommunications (IMT) for 2020 and beyond. [16]

3.1.1 blueSPACE Experimental Evaluation and Demonstration Plans

This PDEng project focuses on the URLLC framework and as seen in Figure 3.2, applications in this area include Industry Automation, Mission Critial Applications and Self-Driving Cars. A similar description can also be seen in the use cases defined by blueSPACE in their Experimental Evaluation and Demonstration plans [6]. These demonstration plans involve the definition of blueSPACE use cases as well as definitions of the key performance indicators to be validated for the demonstration of each use case.

Figure 3.3 describes the blueSPACE use cases. Use case D was chosen for this project.

The KPIs defined by blueSPACE for this use case is described as below:

UCD: Industry 4.0 Environment

- **Supported Capacity:** > 1 Gbps per aggregate group of devices

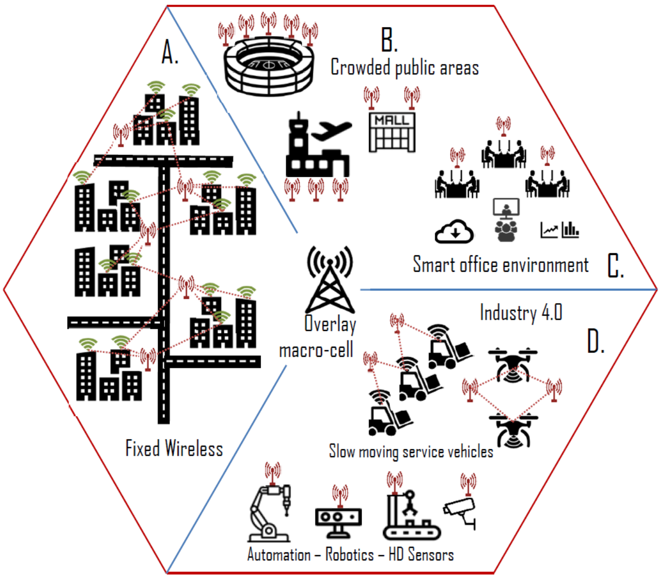


Figure 3.3: Use cases defined by blueSPACE. The use case chosen for this project is D which describes teams of collaborative robots inside smart industries. [6]

- **Information Density:** > 100 Gbps per plant
- **Connected Devices/Users:** > 100 per square meter (group) or > 10,000 per remote radio head (RRH)
- **Latency:** < 5 ms or < 1ms for certain services
- **Mobility:** < 30 kmph

The document also defines the KPI of Low Latency Fronthaul, which needs to be validated in this PDEng project.

- **Definition of KPI:** Physical (PHY) Layer End-to-End (E2E) latency
- **Context/Use case:** UC D (Industry 4.0)
- **Where to measure:** TU/e testbed, E2E link, blueSPACE baseband units (BBU), emulated user terminal
- **How to measure:** Measure E2E latency from ingress PHY layer Base Band Unit (BBU) to egress PHY layer at user terminal. Measure fronthaul latency in loop configuration at the Remote Radio Unit (RRU)

The Experimental Evaluation and Demonstration Plans also highlight the following topics that need to be investigated:

- **Bandwidth hungry device connections** such as video or image processing
- **Critical low latency connections** such as motion control signals over dedicated RF channels

3.1.2 5G Network Architecture

The network defined by blueSPACE involves an optical space division multiplexing (SDM) based architecture that consists of a central office (CO) connected to remote nodes (RNs) using multicore fibers (MCF), which is then connected to base stations known as RRUs. The RRUs receive orthogonal frequency division multipling (OFDM) modulated 5G signals from the CO and up-convert them to the mm-wave (26 GHz) band using photonics. This traffic is received by the user equipment (UE). This architecture is defined in the following image.

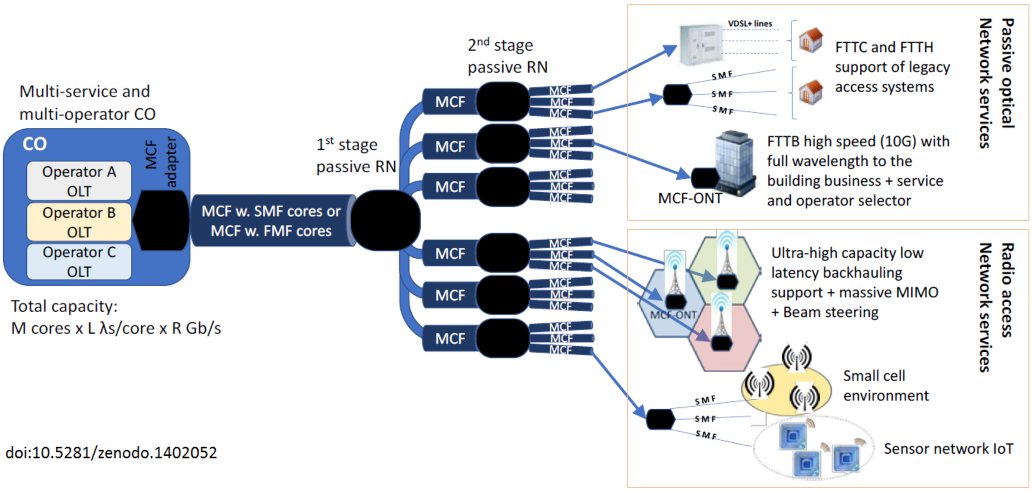

Multi-service and multi-operator CO
CO
Operator A OLT
Operator B OLT
Operator C OLT
MCF adapter
MCF w. SMF cores or MCF w. FMF cores
1st stage passive RN
2nd stage passive RN
MCF
Total capacity:
M cores x L λs/core x R Gb/s
VDSL+ lines
SMF
FTTC and FTTH support of legacy access systems
FTTB high speed (10G) with full wavelength to the building business + service and operator selector
MCF-ONT
Passive optical Network services
Ultra-high capacity low latency backhauling support + massive MIMO + Beam steering
Small cell environment
Sensor network IoT
Radio access Network services
doi:10.5281/zenodo.1402052

Figure 3.4: blueSPACE network architecture vision based on SDM-enabled transport technology and supported services [4]. This PDEng projects limits the scope to Radio Access Network services. SMF: Single mode fiber, RN: Remote node, B: building, C: curb, H: home

This architecture, once implemented will be tested using the Eindhoven 5G Brainport Testbed which is based on the C-RAN concept that accounts for the fronthaul between the CO and the RRU [17] with the wireless interface to the UE. This platform is implemented using fiber optic cables and supports low latency, high data-rate applications using 5G mm-wave signals [4]. The diagram below describes the architecture of this testbed.

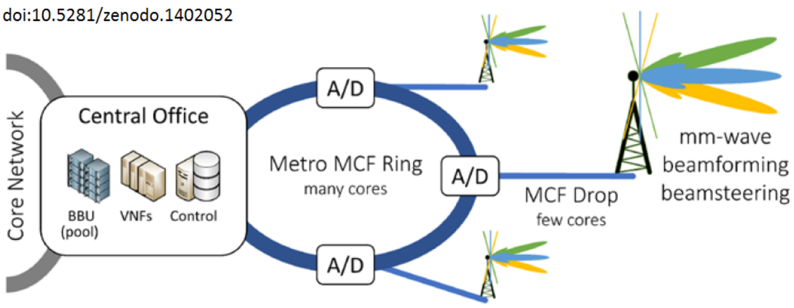


Figure 3.5: The Eindhoven 5G Brainport Testbed, built using fiber optic cables and mm-wave RRUs, based on Figure 3.4 [4]. A/D: add/drop node, VNF: virtualized network function

Additional details about the terminologies and abbreviations used in the above architecture can be found in the blueSPACE Concept Paper [4]. For this PDEng assignment, the relevant aspect of the above architecture is the mm-wave interface between the base stations and the user equipment (UE). In this case, the base station would be a remote computer and the UE would be the on-board computer of a mobile robot.

3.1.3 Open Air Interface

OSI Model and the 5G Protocol Stack

Every telecommunication or computing system can be generalized with the Open Systems Interconnection model (OSI model) [18]. The OSI model breaks down a communication system into seven abstraction layers. These layers are explained in the figure below.

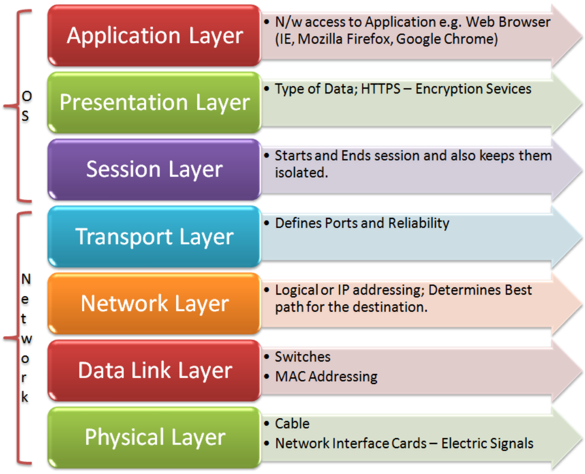

O
S
Network
Application Layer
• N/w access to Application e.g. Web Browser (IE, Mozilla Firefox, Google Chrome)
Presentation Layer
• Type of Data; HTTPS – Encryption Sevices
Session Layer
• Starts and Ends session and also keeps them isolated.
Transport Layer
• Defines Ports and Reliability
Network Layer
• Logical or IP addressing; Determines Best path for the destination.
Data Link Layer
• Switches
• MAC Addressing
Physical Layer
• Cable
• Network Interface Cards – Electric Signals

Figure 3.6: A brief explanation of the OSI model layers [19]

Similarly, a 5G communication interface can also be categorized with these seven layers. The diagram below shows the 5G protocol stack and how these layers correspond to the OSI model.

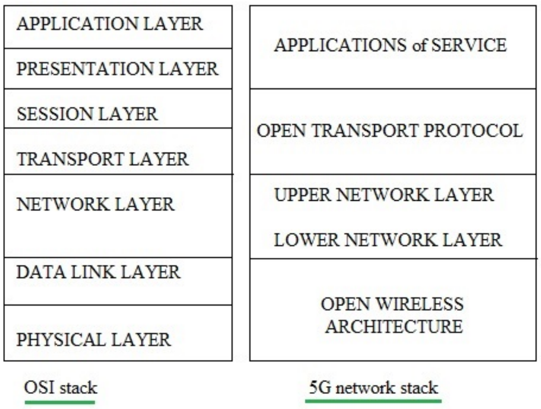


Figure 3.7: OSI model and the corresponding layers of the 5G network stack [20]

Open Air Interface

The OpenAirInterface Software Alliance (OSA) is a network of engineers and researchers that are working towards the development of a 5G cellular stack that will run on COTS hardware. This cellular stack is called the OpenAirInterface. OSA currently has a functioning 4G LTE based implementation using an Intel computer and a commercially available Software Defined Radio. Their 5G implementation, called OpenAirInterface5G is currently under development and is an extension of their existing 4G LTE based OpenAirInterface. Henceforth in this report, OpenAirInterface5G is generalized as OpenAirInterface.

OpenAirInterface (OAI) [21] implements Layer 1, 2 and 3 of the 5G protocol stack on both the base station and the UE, and provides an interface between the higher layers and the physical hardware. Both the base station and UE are Linux computers in this case, and the robot software implements the higher layers of the protocol stack. The base station is commonly known as a Node-B. For LTE and 5G based RANs, base stations are called eNB (evolved Node-B) and gNB (next generation Node-B) respectively. The software running on these base stations are identical. However, since OAI is built using 4G LTE terminology, the base station is henceforth called eNB in the upcoming chapters.

In order to facilitate mm-wave communication between the UE and eNB computers, additional hardware is required to up-convert the generated signals into wireless mm-wave signals. OAI can currently be implemented with 4G LTE frequencies using off-the-shelf hardware like the USRP B210 Software Defined Radio (SDR) [22] and LimeSDR [23]. However, blueSPACE is in the process of developing custom hardware to interface with OAI and communicate with mm-waves.

The diagram below describes the architecture of OSA's 4G LTE demo using OAI. As shown, three computers are used - a User Equipment which is connected to the eNode-B wirelessly (using SDRs), and the eNB connected to an Evolved Packet Core (EPC). The EPC provides the eNB access to the core LTE network. For an OpenAirInterface5G implementation, although the UE and the eNB communicate using 4G LTE band signals, the eNB is connected to a 5G Core Network (CN).

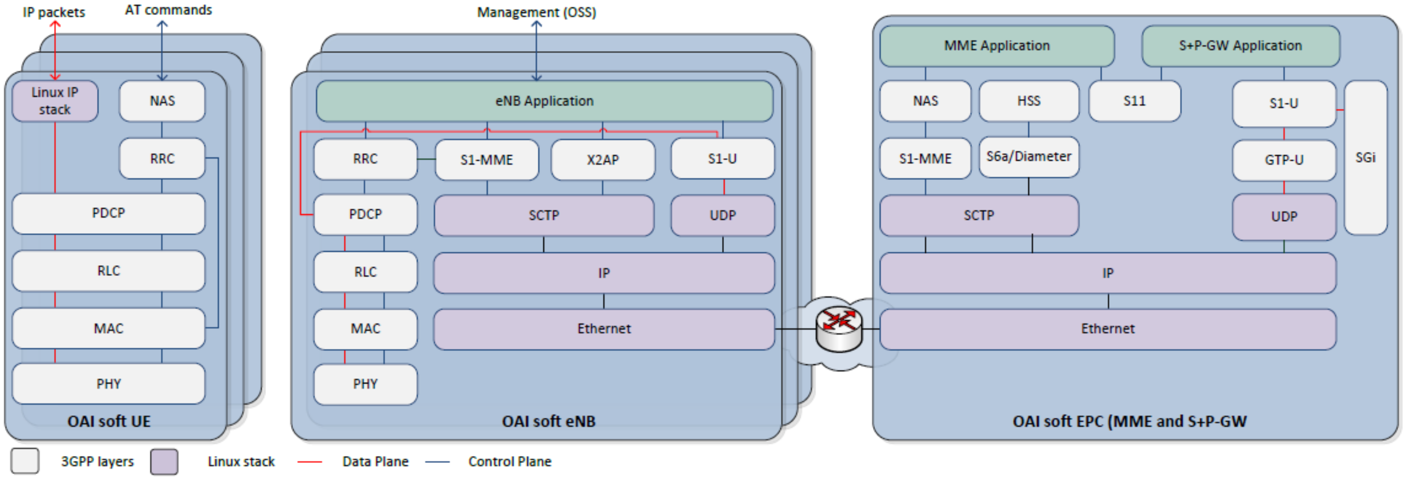


Figure 3.8: This 4G LTE Demo shows the blocks implemented by OpenAirInterface in Layers 1, 2 and 3 on the UE, eNB and the EPC. As shown, the Linux IP stack in the UE sends packets from the higher OSI model layers through the implemented blocks, to the eNB application [24].

Figure 3.9 shows how the 5G protocol stack is implemented on the UE, the next generation Node-

B and the CN. However, this project only focuses on the connection between the UE and the gNB. Without the CN, the UE-gNB connection is identical to OAI's UE-eNB connection.

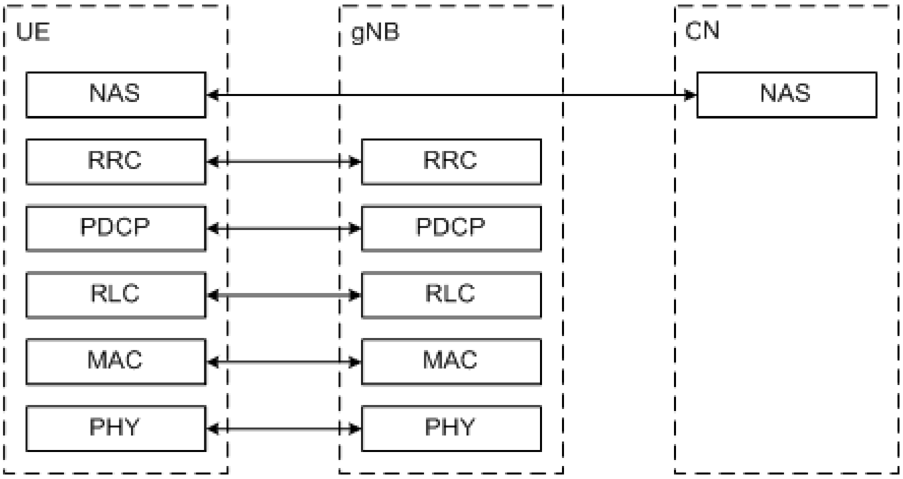


Figure 3.9: The protocol stack implemented on the UE, the gNB and CN. This diagram shows that between the UE and the gNB the implemented blocks are the same as the blocks in Figure 3.8, between UE and the eNB [25].

3.1.4 blueSPACE Challenges and Constraints

The main concern for blueSPACE, as explained above is the demonstration of a robotic application using 5G. This is also a challenge since the blueSPACE team is made up of telecommunications engineers and researchers with no experience in robotics. In addition, the mm-wave communication hardware and the OAI software stack is under development, by blueSPACE and the OpenAirInterface Software Alliance respectively. Due to the lack of experience, configuring this hardware and software on a robot platform is also an added challenge that needs to be solved.

The constraints involve the organization, acquisition and setup of the 5G test setup using blueSPACE's hardware. However, due to delays in the arrival of the blueSPACE hardware, the project is to be implemented using LimeSDR boards [23] and mm-Wave converters as explained in later chapters.

3.2 Tech United Turtle Platform

The Tech United soccer robots are known as TURTLES or Tech United Robot Team: Limited Edition. The robot team is made up of five robots - four players and one goalkeeper robot, who work together to play soccer without any human intervention. The following subsections describe the architecture of the Turtle robot platform as well as challenges currently faced by the team.

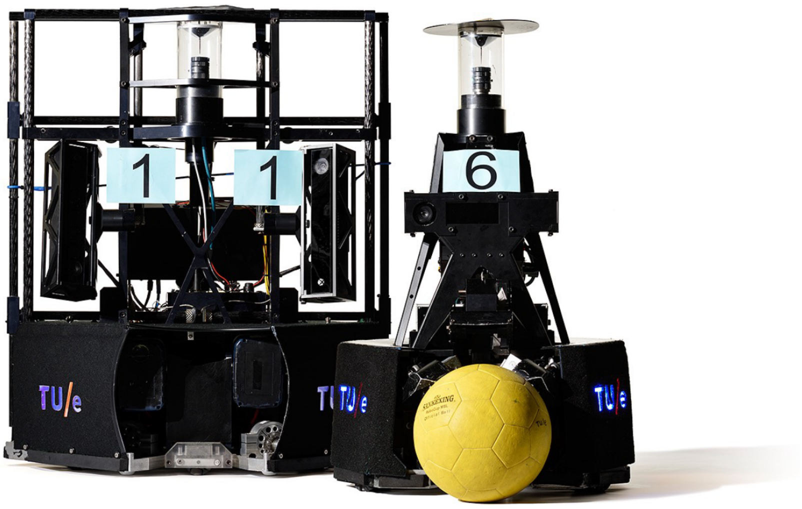


Figure 3.10: The TURTLE robots. The Turtle on the left is the goalkeeper and the one on the right is the player robot [7]

3.2.1 Turtle Hardware

The Turtle hardware is divided into three sections:

- **Base:** The base consists of three drive motors, placed in a triangular formation, each connected to an omni-directional wheel. The base also consists of electronics to drive these motors as well as sensors for odometry and temperature sensing. The base also houses status LEDs and an emergency switch.

- **Ball Handling Mechanism:** The ball handling mechanism consists of actuated wheeled arms to hold and move the ball around, as well as a kicking mechanism to kick the ball in multiple different ways. It also involves encoders to measure the speed of the arms for closed-loop control.

- **Upper Body:** The upper body consists of an Intertial Measurement Unit (IMU) [26], the vision and computing components. The vision system consists of an omnivision camera [27] setup with a parabolic mirror to provide a 360 degree view of the environment. Additionally, there is a Microsoft XBox Kinect v2 [28] camera facing forward as an additional source for detecting the ball and the players during a match. The upper body also houses the computing equipment - an NVidia Jetson [29] board to compute data from the Kinect camera, an industrial Linux PC [30] running a real-time kernel, and a Beckhoff stack [31] which acts as the input-output bridge between the sensors/actuators and the Linux computer. The IMU is connected via USB to the Beckhoff stack and provides accelerometer and gyroscope measurements. The Linux computer is connected to the omnivision camera using a GigE connection, to the Beckhoff stack using an EtherCAT connection [32] and to the NVidia Jetson using an Ethernet connection.

The assembled robot is show in Figure 3.11. This rendering also highlights the important components of the Turtle. Additional details about the Turtle's hardware specifications and schematics can be found on the Robotic Open Platform [33].

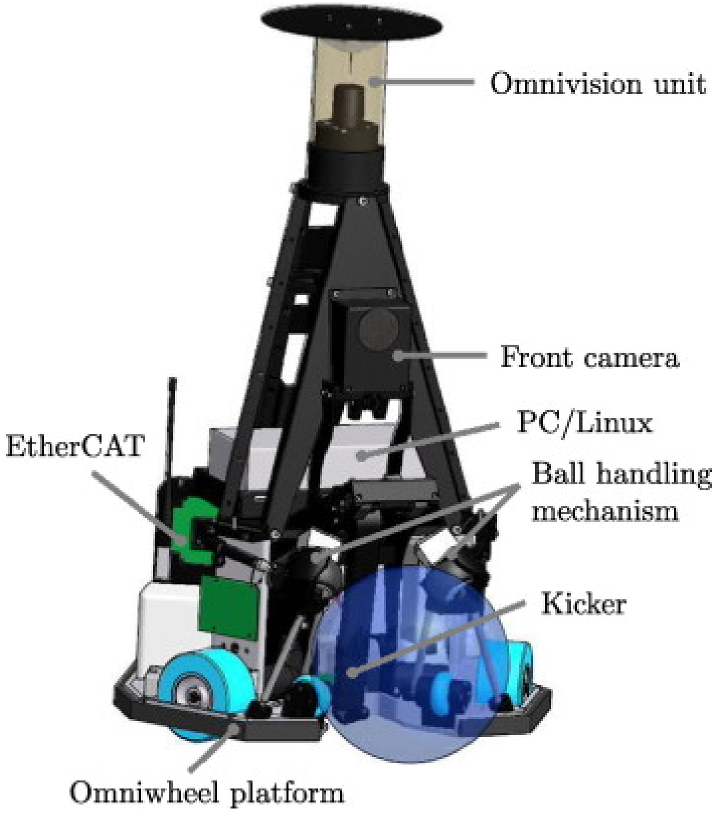


Figure 3.11: A fully assembled robot with the most important subsystems highlighted. [34]

The diagram in Figure 3.12 describes hardware layout of the Turtle - an abstraction of how the actuators, sensors, cameras and computing boards are connected to each other. As shown in this diagram, the Turtles connect to a Wi-Fi access point in order to communicate with a base station and a referee box. The referee box is a computer connected (via the switch) to the robots from both competing teams. This computer allows the referee to assess the game and to intervene by giving pre-determined commands to the players. The base station is used for recording a game during a match for future use, but can also be used for developing and testing software for the Turtle. The Turtles also communicate with each other over the access point using a UDP multicast network to make collaborative decisions. The diagram in Figure 3.13 shows how the Turtles are connected over the multicast network.

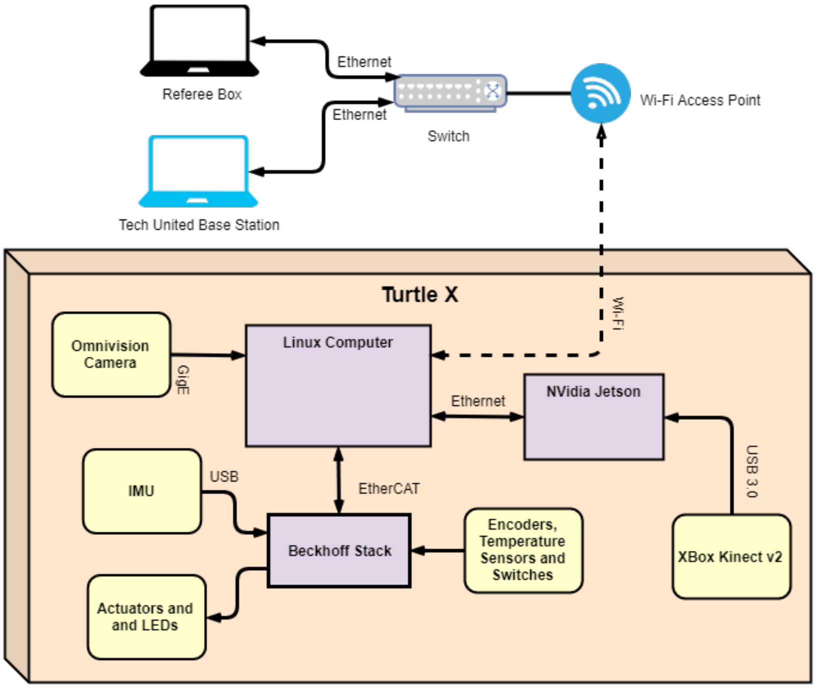


Figure 3.12: Hardware layout diagram that represents how the sensors and actuators are connected with the computers within the robot. The diagram also shows how the robot connects with external computers over Wi-Fi.

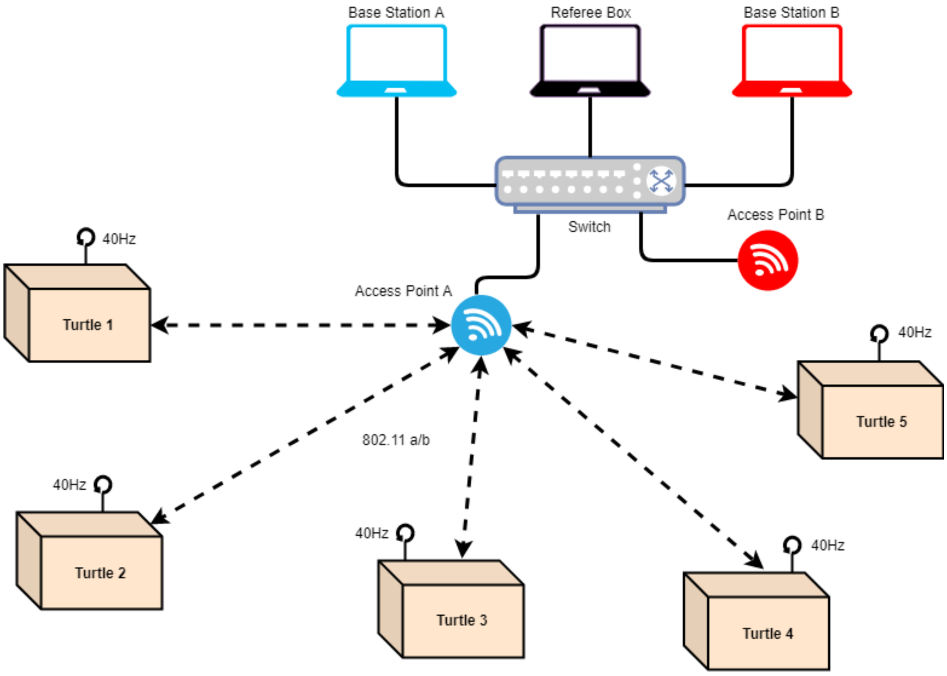


Figure 3.13: UDP multicast based communication framework for the Turtles during game-play. The opponent team uses a similar communication framework using their own access point. The Turtles are programmed to update at a frequency of 40 Hz.

3.2.2 Turtle Software

The Turtle Software was surveyed using the document titled 'A Gentle Introduction to the Tech United Turtle Software'[35] and the Tech United software repository [36]. The software architecture is developed on Simulink using MATLAB/Simulink library blocks and S-functions [**?**] written in C code. The architecture involves software running on the Turtle on-board computers as well as monitoring, visualization and debugging tools running on a Tech United development PC connected over Wi-Fi. The Turtle on-board software can be divided into the following categories:

- **Vision:** The vision block is an executable that acquires images from the omnivision camera at 60 Fps and uses computer-vision algorithms to find the Turtle's pose (position and orientation) on the soccer field, the position of the ball and other robots (teammates and opponents) in the acquired camera image. The resulting data is stored in the Real-Time Database (explained later) and is shared with the other robots.

- **World Model:** The world model executable reads the data from the vision executable (pose, ball position, player positions) and data from all other robots with a frequency of 50 Hz, and uses sensor fusion techniques to determine accurate pose estimates of each of the playing robots (team-mates and opponents) and an accurate position estimate of the ball. This allows the executable to create an accurate representation of what is happening on the field, i.e. a model of the 'world'. This data is used by the strategy executable to determine the team's strategy and the individual Turtle's role and actions.

- **Strategy:** The strategy executable executes at 100 Hz and uses the world model as input. This block uses a look-up table to check the game situation - which Turtle has the ball, which half is the Turtle in, what is the status of the Turtle/Team - attack or defend? Based on this, the strategy block then determines a role and a target position for all robots. This strategy is used to assign actions to the individual Turtle such as 'Go to Target', 'Aim at Target' or 'Shoot at Goal'. Each action corresponds to skills such as Dribbling, Moving, Shooting, Aiming, etc and this data alongside the determined target position is provided to the motion executable which plans the trajectory towards that target.

- **Motion:** The motion executable executes all the control loops for actuating the robot. This executable operates at 1000 Hz and acquires data in real-time from the Beckhoff stack using EtherCAT which is an industrial real-time communication framework. This framework allows the robot computer to be configured as the host and the Beckhoff Stack as the slave and this enables high speed, low-latency data transfer between the robot computer and the Beckhoff stack. Using EtherCAT, the motion executable is able to acquire data from the sensors and provide digital commands to the actuators. The control loops use the sensor data and the output from the strategy executable to plan a trajectory towards the target and to provide actuator commands to navigate that trajectory.

- **Real-Time Database (RTDB)**: RTDB is a database [37] that acts as the communication interface between each of the above executables. Each executable shares some data to the database and reads data that is shared by the other executables. As an example, the compass measurement from the IMU is read by the motion executable and stored as a record in the database. This record is accessed by the vision executable to determine the pose of the Turtle. The records in the RTDB can be divided into two categories: local and shared. Local records are used and updated by the executables locally on the robot computer. Shared records are blocks of memory allocated for data acquired from the other robots and local data that is to be shared with the other robots. The RTDB also helps the robot computer share records with the NVidia Jetson board that processes information from the Xbox Kinect camera. The local RTDB structure on each Turtle computer is shown in Figure 3.14

- **Kinect:** The Kinect block is an executable that runs on the NVidia Jetson, to which an XBox Kinect v2 camera is connected. This executable acquires image and depth data from the Kinect camera and determines the position of the ball and the other robots. These positions are fused with the position estimates from the vision executable and is used to reinforce the ball and player position estimates determined in the world model.
- **Tools:** The multicast network used by the robot can also be accessed by a configured development PC. This PC is used to develop and build the Turtle's on-board executables and also to run tools for monitoring and visualization. The two tools important for this project are the **Turtle Remote Control (TRC)** and the **Turtle Simulator**. These tools are used to provide remote commands to the Turtle and visualize its movement on the development PC.

The complete high-level software architecture is described in the layout diagram in Figure 3.15.

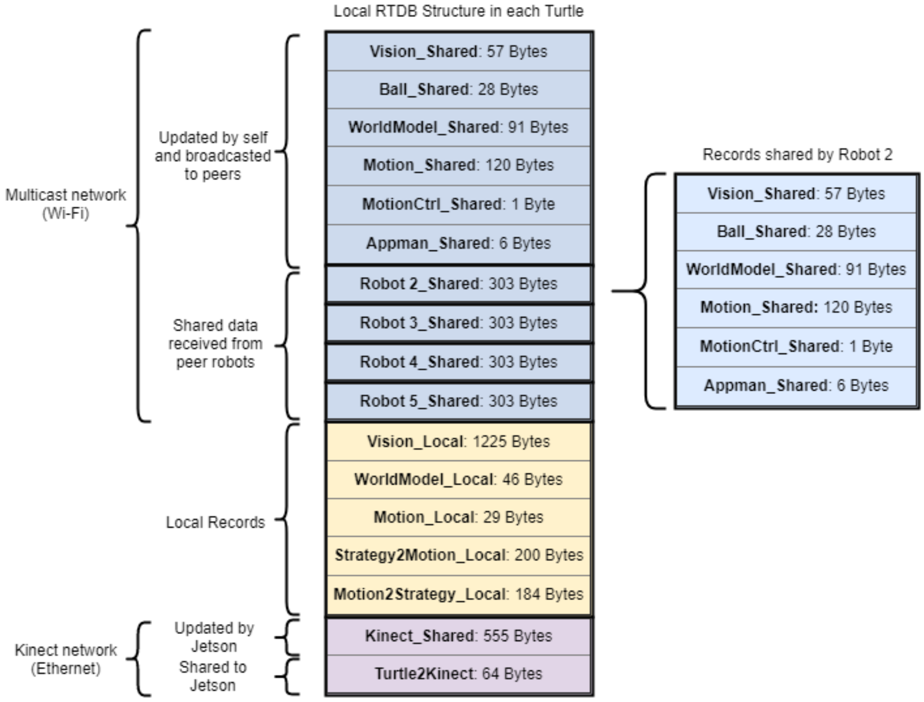

Local RTDB Structure in each Turtle
Vision_Shared: 57 Bytes
Ball_Shared: 28 Bytes
WorldModel_Shared: 91 Bytes
Motion_Shared: 120 Bytes
MotionCtrl_Shared: 1 Byte
Appman_Shared: 6 Bytes
Robot 2_Shared: 303 Bytes
Robot 3_Shared: 303 Bytes
Robot 4_Shared: 303 Bytes
Robot 5_Shared: 303 Bytes
Vision_Local: 1225 Bytes
WorldModel_Local: 46 Bytes
Motion_Local: 29 Bytes
Strategy2Motion_Local: 200 Bytes
Motion2Strategy_Local: 184 Bytes
Kinect_Shared: 555 Bytes
Turtle2Kinect: 64 Bytes
Multicast network (Wi-Fi)
Updated by self and broadcasted to peers
Shared data received from peer robots
Local Records
Kinect network (Ethernet)
Updated by Jetson
Shared to Jetson
Records shared by Robot 2
Vision_Shared: 57 Bytes
Ball_Shared: 28 Bytes
WorldModel_Shared: 91 Bytes
Motion_Shared: 120 Bytes
MotionCtrl_Shared: 1 Byte
Appman_Shared: 6 Bytes

Figure 3.14: Local RTDB structure of each Turtle. The blue blocks describe the shared records of the Turtle. This includes records updated by the other peers on the multicast network as well as the data shared by the Turtle to its peers. The yellow blocks are local records updated locally by the executables running on the robot computer and the purple blocks are shared by the NVidia Jetson and the Kinect process. Each record is made up of variables that are either updated or accessed by each of the Turtles executables.

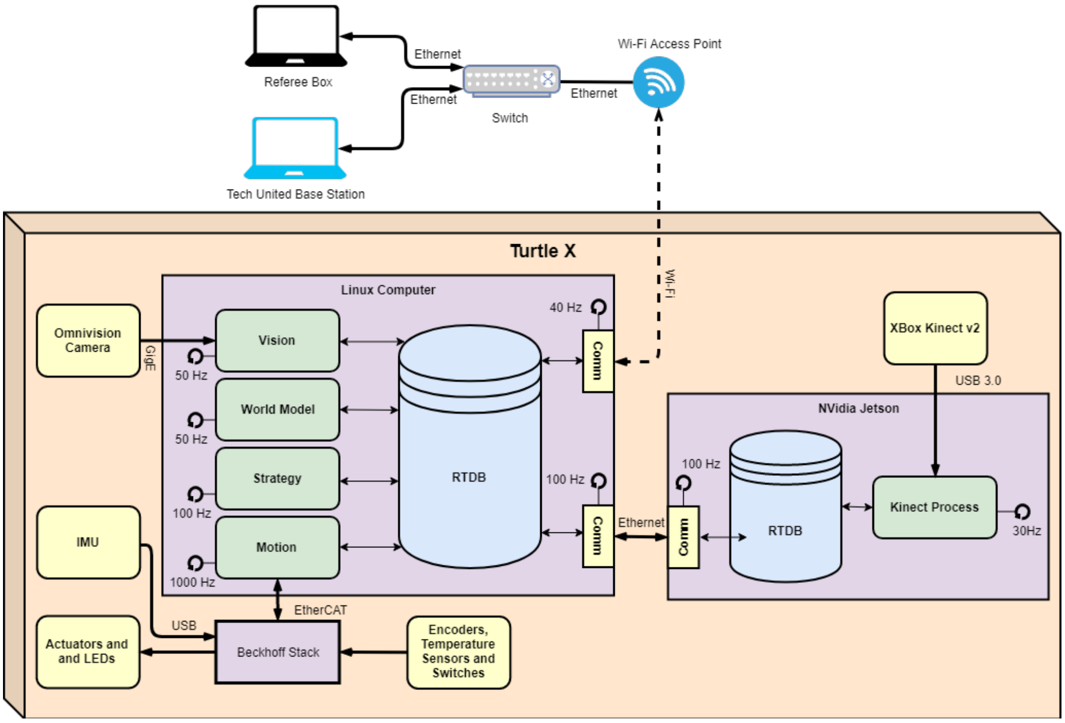


Figure 3.15: Existing software architecture on the Turtle on-board computers. The architecture also shows how the Linux Computer and the NVidia Jetson each execute their own local version of the RTDB and the communication interface between these two computers. The Comm interface describes an IP based interface that operates at 40 Hz (wireless interface) and 100 Hz (wired interface).

3.2.3 Tech United Challenges and Constraints

Challenges

- **Inter-robot communication:** The Turtles are constantly sharing information with their peers about each others' poses and the ball position over a UDP multicast connection. These data updates are asynchronous and cause packet collisions which can result in up to 40% packet losses over the network during an average match. Due to the asynchronous nature of the data updates, the local world model created on each of the robots might not be the same. This affects the strategy created locally on each robot and hence the game-play. Despite these challenges, the team can consistently rank high in tournaments, however an improvement in the inter-robot communication would allow the local world models to be more consistent and hence improve the performance of the team.

- **Computationally expensive algorithms:** As the Turtles are constantly being improved, Tech United is exploring the use of computationally expensive methods such as Machine Learning and Neural Networks to improve ball and player detection. However, the on-board Linux computer is not capable of executing these methods. Currently, team members use an external laptop to train and execute the software. Therefore a solution that involves processing computer vision methods on the cloud would be extremely desirable. Additionally, even the Kinect process could be executed in the cloud and hence the NVidia Jetson would not be needed.

Constraints

- **Full robot team availability:** Due to Tech United team members using some of the available robots and the unavailability of multiple 5G hardware setups, it is not possible to utilize the entire robot team for the experimentation. Therefore, a demonstration involving only a single robot is preferred.
- **Single Turtle availability:** During the period of this project, Tech United participates in multiple tournaments and demonstrations, which involves all of the robots. Therefore, access to a single Turtle during certain periods of the project is not possible and needs to be accounted for in the detailed task breakdown and planning.
- **RoboCup MSL Rulebook:** The current RoboCup MSL rulebook [38] does not allow the usage of networks other than Wi-Fi, however any demonstration with 5G will stand as a proof-of-concept in order for the rulebook to be changed.

4 Requirements and Use Case Definition

This chapter describes the requirements elicitation process. This process involves the definition of high level requirements after consultation with the stakeholders, and the proposal of potential use case alternatives. Finally, after a Project Steering Group (PSG) meeting, a final use case is decided upon, after which low level requirements are derived for that particular use case. The process also involves a risk assessment, where potential risks throughout the project are highlighted and mitigation plans are defined. This chapter describes the Definition Phase of the work phase plan.

4.1 High Level Requirements

After several meetings with all involved stakeholders, and after hearing their concerns, the following list of high level requirements were drawn up:

- **H1:** The proposed application must ***demonstrate the advantages of using 5G*** communication in the field of robotics. This can be either an improvement in inter-robot communication, a reduction in computing power or costs, for example.
- **H2:** The proposed application must involve a ***ground-based robot*** (due to the limited payload capacity of aerial vehicles and hardware safety concerns if the aerial vehicles crash)
- **H3:** The ground-based robot must ***communicate in real time over a single channel, bidirectional 5G connection*** with a remote computer/server
- **H4:** The proposed use-case scenarios must be limited to potential ***Industry 4.0 applications***
- **H5:** The proposed application must focus on and ***validate the Throughput and End-to-End Latency KPIs***

The above requirements were derived from the Stakeholder Analysis (Appendix C.2) and documented in the Requirements Document (Appendix C.3). The literature study was used to propose feasible 5G use cases that can be demonstrated using the Turtle platform. The following sections describe this survey.

4.2 Use Case Definition and Selection

This section describes the concept exploration process, and the process of choosing a concept to implement as a use case in this project. Firstly, two concepts were proposed after a brief feasibility study. This feasibility study involved a study of blueSPACE and 5G related literature to understand how the 5G interface could be implemented, a deeper survey of the Turtle code to see if the proposed concept is feasible and feedback sessions with stakeholders from blueSPACE and Tech United to understand how each proposed use case could add value to them. The feedback sessions resulted in a third use case - **Completely Decoupled Computation** - which was chosen for this project. The three proposed use cases are described in the following subsections:

4.2.1 UC1: Decoupled World Model

The first use case involves replacing the inter-robot communication setup from Wi-Fi (Figure 3.13 to 5G. This involves the usage of a 5G base station (eNB) and five 5G user equipments (UE) on each Turtle. The Wi-Fi access point is replaced by a 5G mm-wave antenna that communicates with the UEs. Since each Turtle shares a very small amount of data (around 303 Bytes) as shown in Figure 3.14, and 5G would allow up to 1 Gbps of data rate at a low latency, the performance of the system would be much more reliable than with the current Wi-Fi framework. This use case also allows each Turtle to communicate with the eNB over unique individual frequencies, thus preventing packet losses due to collisions.

Table 4.1: Shared data in UC1

Data	Source	Destination	Rate	Size	Type
Shared Pose Data	UE	eNB	40 Hz	303 B	RTDB records
Global World Model	eNB	UE	50 Hz	127 B	RTDB records

The corresponding expected data rates and communication methods for uplink and downlink are as follows:

Table 4.2: Required data rates in UC1

Data	Frame Size	Req. Data Rate	Method
Shared Pose Data	303 B	2.424 Mbps	UDP
Global World Model	127 B	1.016 Mbps	UDP

Here, both uplink and downlink satisfy the 5G requirements.

In this use case, as the inter-robot communication is more reliable, the local world model on each Turtle is expected to be almost identical. This means that each robot would have the same understanding of the environment. Additionally, since there are no packet losses due to collisions, the world model is also more accurate. This improves the performance of the team. For example, if only one robot detects the ball, this information is relayed accurately and with minimal latency to all other robots. This would help each robot make better decisions on strategy.

Here, it is convenient to decouple the world model executable and run it on the eNB. This 'global' world model executable would read the data shared by each Turtle, make a model of the world and broadcast it to each Turtle (the UEs could be configured to receive data on the same frequency). As shown in the functional view diagram in Figure 4.1, it would also be possible to increase the update rate of each Turtle from 40 Hz. With a powerful eNB computer, the world model executable could be run remotely at a higher rate, while also reducing the overall CPU usage of each Turtle's on-board computer. However, there are also some drawbacks. The amount of data shared is so low, the same experiment could be repeated with a 4G LTE network to get the same performance. Hence, it does not really show the advantages of 5G over other existing mobile networks.

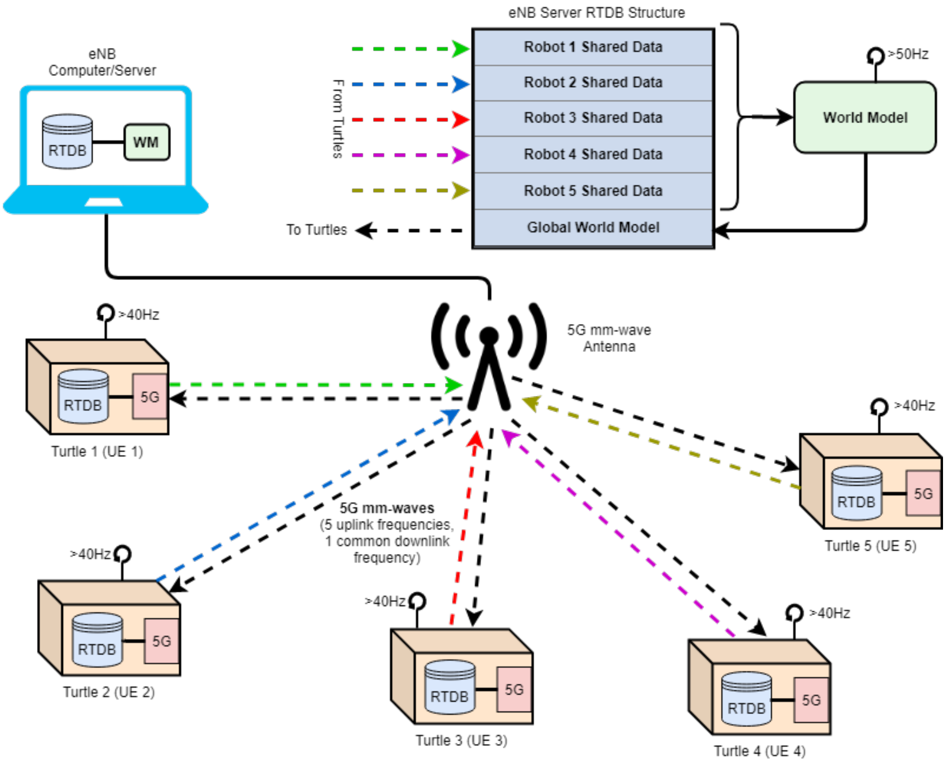


Figure 4.1: **UC1:** This architecture upgrades the Wi-Fi framework in Figure 3.13 to 5G. This allows the World Model executable to be decoupled and executed remotely. A local RTDB is configured which communicates with the world model executable in the eNB and with the local RTDBs of each Turtle over the 5G network. The resulting world model in the eNB is broadcast to each UE.

4.2.2 UC2: Decoupled Vision/Kinect Process

The second use case attempts at tackling the issue of limited computational power on-board the robot. Here, a decoupled architecture is proposed, where the omnivision camera and Kinect images and depth data are acquired on the on-board computer and streamed to a remote computer. The remote computer executes the Vision and Kinect processes and returns the resulting data to the on-board computer.

Like UC1, here it is convenient to also decouple other blocks that do not interact with physical hardware like the World Model and Strategy executables. In the proposed architecture (Figure 4.2), only the Motion block and a local RTDB is executed on the robot computer and everything else is processed remotely. This removes the need for an NVidia Jetson board on the robot, which reduces the cost of the Turtle platform. Additionally, The Kinect process which acquires image and depth data at 30 Fps could also be improved to acquire and process data at a higher frequency.

Since the omnivision and Kinect image sizes are significantly higher than the data in UC1, this use case requires a bigger bandwidth and higher data rates, and would not be as efficient with other communication methods such as Wi-Fi or 4G LTE. Hence, UC2 would a perfect example for a 5G use case. The requirements for this use case is shown in the table below.

Table 4.3: Shared data in UC2

Data	**Source**	**Destination**	**Rate**	**Size**	**Type**
Omnivision Data	UE	eNB	60 Hz	1 MB	YUV image
Kinect Data	UE	eNB	30 Hz	1 MB	RGB image + Depth data
Motion Data Uplink	UE	eNB	1000 Hz	185 B	RTDB records
Motion Data Downlink	eNB	UE	1000 Hz	1545 B	RTDB records

The corresponding expected data rates and communication methods for uplink and downlink are as follows:

Table 4.4: Required data rates in UC2

Data	**Frame Size**	**Req. Data Rate**	**Method**
Omnivision Data	1 MB	480 Mbps	gstreamer [39]
Kinect Data	1 MB (compressed)	240 Mbps	kv2streamer [40]
Motion Data Uplink	185 B	1.48 Mbps	UDP
Motion Data Downlink	1545 B	12.36 Mbps	UDP

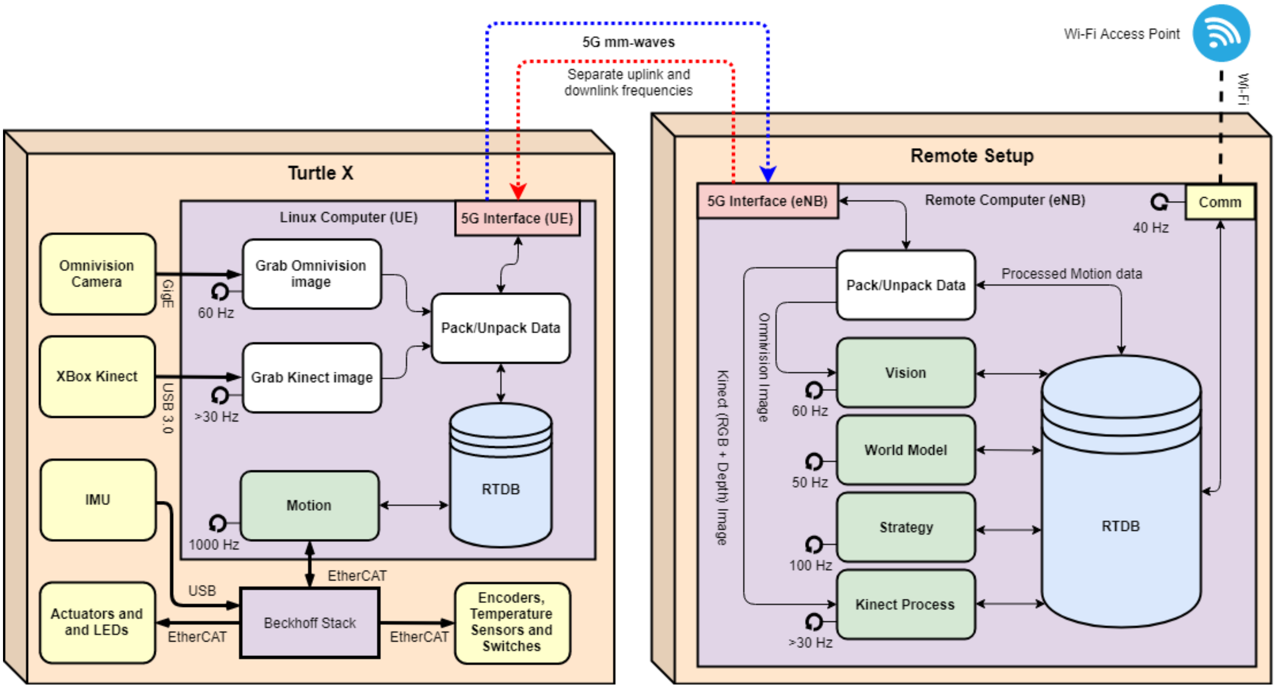


Figure 4.2: **UC2:** This architecture shows how the Vision, Kinect, World Model and Strategy blocks have been offloaded to a remote computer. The white blocks describe the blocks to be implemented for this use case. Once all robots are configured on the same remote server, the Wi-Fi communication can be performed using internal interfaces instead of Wi-Fi.

The overall uplink data rate is $(480 + 240 + 1.48) Mbps + 20\%\ overhead = 865.776 Mbps$, which is less than 1 Gbps and satisfies the 5G requirements. The data overhead is the small amount of data added to each packet - this involves the addressing and routing information which the protocol stack on the receiving end uses. In the above calculation, this has been approximated to 20% of the packet size, and hence 20% of the overall data rate. The downlink data rate is below the maximum data rate even without the overhead. Since the motion block operates at 1000 Hz, the expected latency should be ideally lower than one clock cycle, i.e., <1 ms. For the omnivision and Kinect data, higher latencies are tolerable.

4.2.3 UC3: Completely Decoupled Computation

UC3 is the final use case chosen for this project and was proposed as a result of iterative feedback sessions from the proposals of UC1 and UC2. This use case is an extension of UC2 and involves offloading all the processes from the on-board robot to the remote server. In this case, the on-board computer acts only as a gateway to send acquired sensor, camera and Kinect data to the remote server, and to receive actuation commands from the robot software running remotely. The data that is transmitted or received over the network can be divided into two types:

- **High bandwidth, low frequency:** Omnivision image, Kinect image, Kinect depth data
- **Low bandwidth, high frequency:** EtherCAT sensor data, IMU data, robot actuation commands from the remote computer

The Omnivision image stream, and the Kinect image/depth stream is transmitted from the Turtle to the remote server at frequencies of 60 Hz and 30 Hz respectively. The sensor and IMU data are streamed from the Turtle to the remote server at 1000 Hz and the actuation command returned at 1000 Hz. Using multiple types of data like this not only helps to validate the latency of the system but also the achieved data rates and throughput. The data transmitted and received over the network is seen in the table below.

Table 4.5: Shared data in UC3

Data	**Source**	**Destination**	**Rate**	**Size**	**Type**
Omnivision Data	UE	eNB	60 Hz	1 MB	YUV image
Kinect Data	UE	eNB	30 Hz	1 MB	RGB image + Depth data
Sensor Data	UE	eNB	1000 Hz	248 B	EtherCAT + IMU data
Actuation Data	eNB	UE	1000 Hz	168 B	EtherCAT actuation signals

The corresponding expected data rates and communication methods for uplink and downlink are as follows:

Table 4.6: Required data rates in UC3

Data	**Frame Size**	**Req. Data Rate**	**Method**
Omnivision Data	1 MB	480 Mbps	gstreamer [39]
Kinect Data	1 MB (compressed)	240 Mbps	kv2streamer [40]
Sensor Data	248 B	1.984 Mbps	UDP
Actuation Data	168 B	1.344 Mbps	UDP

Hence, overall uplink data rate is $(480 + 240 + 1.984)Mbps + 20\%\ overhead = 866.38Mbps$, which is less than 1 Gbps and satisfies the 5G requirements. The downlink data rate is 1.344 Mbps without the overhead. Like UC2, the expected latency for the motion data is <1ms. For the omnivision and Kinect data, higher latencies are tolerable.

The proposed architecture for this use case is described in Figure 4.3. The only difference from UC2 is that the motion block and the local RTDB is moved to the remote server. On an implementation level, this change is not a big challenge, however, there are some concerns. As explained in UC2, the the Motion block operates at 1000 Hz, or with a clock cycle of 1 ms. So, with a worst-case latency of 1 ms, the sensor data received by the remote computer is delayed by 1 ms and the actuation commands received by the Turtle is also 1 ms, adding up to 2 ms. This means that over a closed motion control loop, the delay between sensing and actuation is at least 2 clock cycles of the Motion block. However, the motion block has a tolerance of up to 5-7 clock cycles and can handle this delay. These concerns are explained in detail in Chapter 5.

This use case does have a lot of advantages for both Tech United and blueSPACE. Firstly, the application is latency critical and needs a high data rate - which validates the 5G network. On the other hand, it solves the issue of limited computational power on-board the robots and by offloading the processes, would even save on battery life and the cost of each Turtle. The remote server could have no limitations, and could allow the entire team of robots to have their softwares executed in parallel on one server. This means that the inter-robot communication could be performed using internal interfaces on the server instead of Wi-Fi, hence improving the world model updates. Additionally, processes like the Kinect process could be run at a higher rate.

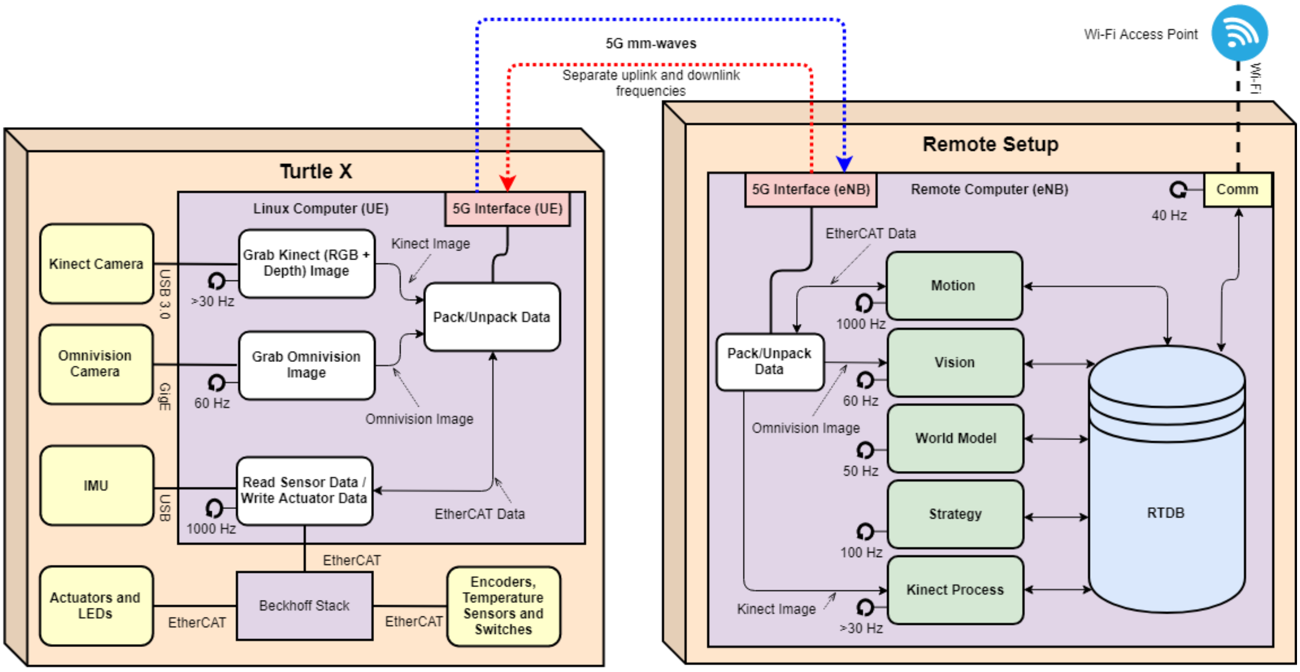


Figure 4.3: **UC3:** This architecture describes an extension of UC2 (Figure 4.2) where all blocks are offloaded to the remote computer. The Turtle computer is responsible for the streaming of omnivision, Kinect and sensor data over 5G mm-waves and for receiving corresponding actuation commands from the remote computer. Here, the RTDB is left unchanged.

4.3 Derived Low Level Requirements

With the stakeholder analysis in Chapter 1.3, high level requirements were defined in Chapter 4.1. Using these high level requirements and the use case defined as UC3, the following low level requirements were derived. The following list is only indicative and the requirements are documented in detail in Appendix C.3. The following list shows the final status of each requirement at the end of the project.

- **L1:** The proposed application must be able to demonstrate and test blueSPACE hardware and architecture. (final status: **achieved**)
- **L2:** The application must improve the performance of the Turtle team and in extension, to fleets of industrial robots in future use cases. (final status: **partially achieved**)
- **L3:** The proposed application must use OpenAirInterface to setup the on-board and remote computers as UE and eNB. (final status: **not achieved**)
- **L4:** The application must enable the transfer of the Omnivision camera stream, Kinect data stream and the sensor data stream from the UE to the eNB. (final status: **partially achieved**)
- **L5:** The application must enable the transfer of the actuation data stream from the eNB to the UE. (final status: **partially achieved**)
- **L6:** The application must be demonstrable using one Turtle (final status: **achieved**)
- **L7:** The application must achieve the network requirements described in Table 4.7 (final status: **partially achieved**)
- **L8:** The application must achieve the Turtle timing requirements described in Table 4.8 (final status: **partially achieved**)
- **L9:** The application must validate/analyze the implemented wireless network and provide recommendations based on it (final status: **partially achieved**)
- **L10:** The application must validate/analyze the robot's performance with 5G and provide recommendations based on it. (final status: **not achieved**)
- **L11:** The application must define the optimal specifications for the UE and the eNB computers for the defined use-case. (final status: **achieved**)
- **L12:** The application must propose recommendations for future demonstrations and integration of a team of Turtles. (final status: **partially achieved**)

Table 4.7: Network requirements

Measurement	**Requirement**
Throughput UE->eNB	866.38 Mbps
Throughput eNB->UE	1.618 Mbps
Bandwidth	1 Gbps
Latency	1 ms
Jitter	< 0.5 ms

Table 4.8: Turtle timing requirements

Measurement	**Requirement**
Camera data Rx frequency (at eNB)	60 Hz
Vision block sample time (at eNB)	16.66 ms
Kinect data Rx frequency (at eNB)	30 Hz
Kinect block sample time (at eNB)	33.33 ms
Sensor data Tx frequency (at UE)	1000 Hz
Sensor data Rx frequency (at eNB)	1000 Hz
Actuation data Rx frequency (at UE)	1000 Hz
Motion block sample time (at eNB)	1 ms
Tolerable maximum round trip latency	5-7 ms
CPU usage of UE	< CPU usage of eNB

4.4 Potential Risks

The following list describes the risks highlighted by the end of the Definition phase. Some of these risks were faced during the project and will be described in upcoming chapters. A detailed risk assessment table, is provided in Appendix C.4. This table includes an detailed list of the highlighted risks, the probability and impact of their occurrence, and relevant mitigation plans.

- **Timeline for 5G hardware/network setup:** Due to development work on the blueSPACE 5G hardware, the exact timeline for acquisition of the hardware and network setup was not known. This would lead to project management as well as system integration issues. One possible contingency is to validate the network using a wired interface to simulate the 5G network.

- **Integration issues:** Since the UE and eNB hardware need to be configure on computers running Tech United software, it is important that both OAI and the Turtle software can function. During the Preparation and Realization phases, issues like hardware configuration and driver issues might arise and solutions need to be found without disrupting the project timeline.

- **Network latency measurement tools:** Since latency needs to be validated, specific tools need to be used to accurately measure the time when data packets are sent and received. While, this can be done individually on both the UE and the eNB, to calculate the latency, the clocks on both computers to be synchronized. Existing tools might not provide the synchronization accuracy that is required for latencies as low as 1ms.

- **Lack of existing literature/technical documentation:** Since 5G is an extremely new concept, not a lot of literature is available, especially for the use of 5G in robotics. This poses a big risk as information gathering and hence, the Definition and Preparation phases would take additional time. Additionally, since off-the-shelf 5G products are not available, the project relies on under-development hardware or commercially available hardware modified to suit the defined use case. The OAI software is also under development and not very well documented.

- **Required data rates might not be achieved:** As delays are expected in the hardware blueSPACE is developing, the project might have to be continued with off-the-shelf hardware configured for 5G communication. One such alternative is the LimeSDR, which would not achieve the expected data rates for the defined use case. The mitigation plan involves reducing the scope of the project so that only a part of the defined architecture can be demonstrated.

5 Architecture and Design Description

This chapter explains the architecture and detailed design (Design Phase of the work phase plan) for the use case defined in Chapter 4.2.3. As explained in the CAFCR methodology in Chapter 2.2, the architecture and design is divided into five views which answers the important questions - **Why, What and How?** These views are described in the following chapters.

5.1 Customer Objectives

This section answers the question: ***What* does the customer want to achieve?**

In this project, the customers are blueSPACE and as explained in above sections, want a 5G based robotic demonstrator that can be used to validate their hardware and network architecture in an Industry 4.0 scenario. They also wish to add value to the robot's performance by using 5G and wish to validate the same. The customer view can be described by the following diagram.

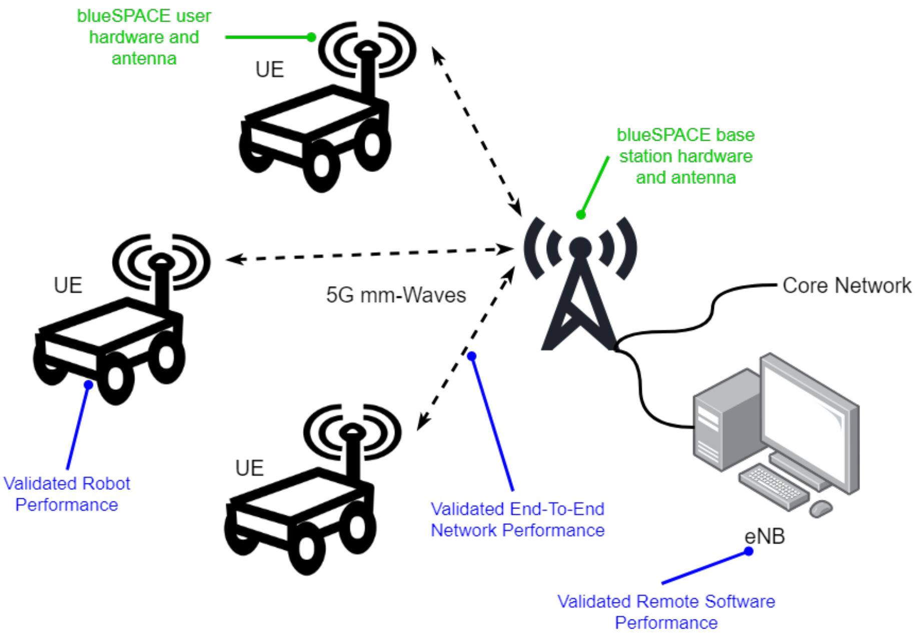

blueSPACE user hardware and antenna
UE
blueSPACE base station hardware and antenna
UE
5G mm-Waves
Core Network
UE
Validated Robot Performance
Validated End-To-End Network Performance
eNB
Validated Remote Software Performance

Figure 5.1: **Customer Objectives:** This diagram describes what blueSPACE wants to achieve from this PDEng project. This view consists of a team of robots (UEs) communicating with a remote server (eNB) over 5G mm-waves. Using this setup, blueSPACE wishes to validate the performance of the network as well as that of an individual robot.

5.2 Application View

This section answers the question: ***How* does the customer realize their goals?**

In order for blueSPACE to achieve its goals, a collaboration with Tech United was initiated and their TURTLE platform was chosen for the demonstrator. However, due to the constraints of robot and 5G hardware availability, the scope was reduced to only one Turtle. In the following application diagram, the Turtle is connected to blueSPACE's radio hardware which converts outbound signals to 5G mm-waves. The hardware on the remote side receives these signals, processes them using the Turtle's software and returns information over mm-waves to the Turtle computer. This communication is facilitated by blueSPACE's hardware and OpenAirInterface (OAI), which configures the Turtle computer and the remote computer as UE and eNB respectively. However, due to delays in the acquisition of blueSPACE hardware, an off-the-shelf software-defined radio called LimeSDR was considered as replacement. Once the blueSPACE hardware is ready, it would replace the LimeSDRs.

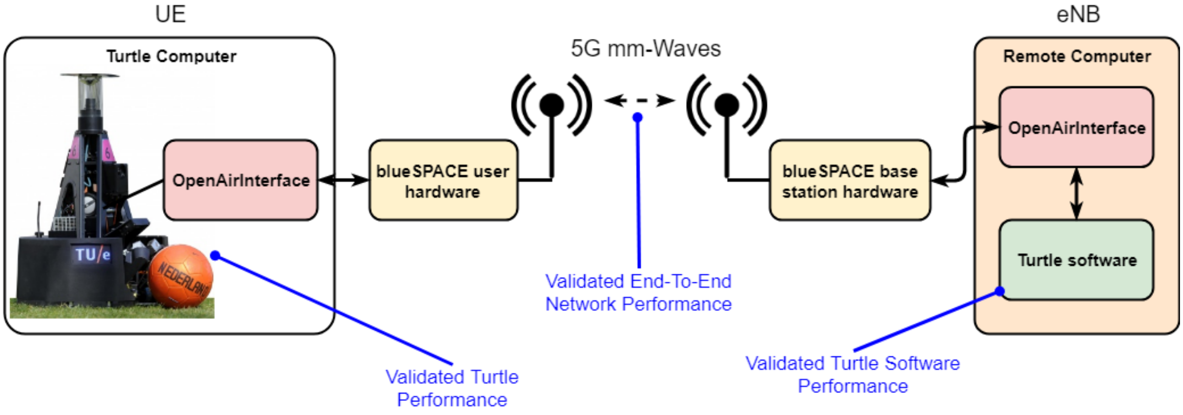

UE
Turtle Computer
OpenAirInterface
blueSPACE user hardware
5G mm-Waves
blueSPACE base station hardware
eNB
Remote Computer
OpenAirInterface
Turtle software
Validated End-To-End Network Performance
Validated Turtle Performance
Validated Turtle Software Performance

Figure 5.2: **Application View:** This diagram describes a high level view of how blueSPACE wishes to realize the proposed demonstrator. Although the scope of the project was reduced to only one Turtle, explained in Chapter 4.2.3 this can be scaled up for an entire team of robots. As shown in the diagram, the OpenAirInterface software runs on both computers and is used to configure them as UE and eNB. Once the setup is implemented, the performance of the UE software, the eNB software and the network is analyzed and validated.

5.3 Functional View

This section answers the question: ***What* is the final product?**

Here, two designs are described. The first design describes the final functional design for the proposed use case. However, due to delays in hardware acquisition, the scope of the project was reduced to only one part of the final design. This is described in the second design.

5.3.1 Final Design

The final product involves a Turtle running OAI and data acquisition software (UE), and a remote computer running OAI and the Turtle's executables (eNB). The UE and the eNB are connected wirelessly using blueSPACE's mm-wave hardware. This can be seen in Figure 5.3. Data is communicated using off-the-shelf tools like gstreamer [39] for streaming video from the omnivision camera, kv2streamer [40] for streaming image and depth data from the Kinect sensor, and UDP to send and receive sensor/actuator data. During experimentation, the 5G enabled Turtle plays a collaborative soccer game with another Turtle with on-board computing and the performance of both are measured and analyzed. The functional view diagram also points out other parameters that will be measured and analyzed in order to validate the design.

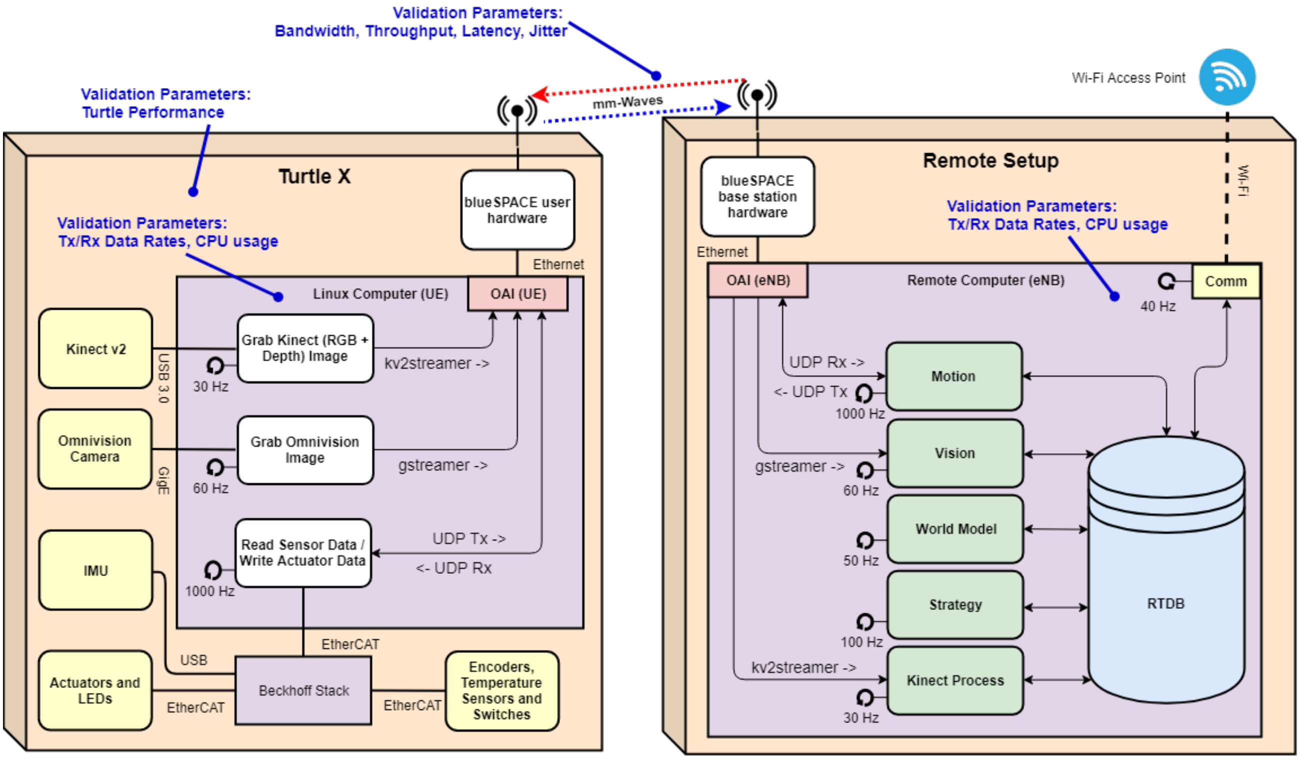


Figure 5.3: **Functional View - Final Design:** This diagram is an extension of Figure 4.3 and describes the hardware interface and the methodologies used to send and receive data. The diagram also shows the validation parameters and where they are measured.

5.3.2 Implemented Design

Unfortunately, due to implementation delays, the blueSPACE hardware could not arrive in time and was replaced by a LimeSDR. LimeSDR is a software-defined radio that interfaces with OAI and communicates using 4G LTE protocols. However, these protocols are very similar to the 5G protocol stack as explained in Chapter 3.1.3. Moreover, the LimeSDR can then be connected to a mm-wave converter that up-converts 4G LTE signals to the mm-wave band and vice versa. Since LimeSDR provides lower data rates than blueSPACE's hardware, the scope of the project needed to be reduced in order to achieve the customer goals. As both blueSPACE and Tech United preferred the demonstration of a latency critical application, it was decided that only the motion block will be offloaded to the remote computer. For the sake of understanding the motion block alone, non-essential processes like Vision, Kinect and World Model are not executed. However, the strategy block is executed on the eNB computer, since it shares data with the motion block via the RTDB.

In this case, the robot sends only the sensor data to the remote setup running the motion and strategy blocks. Since some of the other executables are not running, the motion block is unable to receive any motion commands. Hence, it is provided with manual commands using the Turtle Remote Control (TRC) and resulting actuation commands are sent over mm-waves to the Turtle. The corresponding movement of the robot is visualized using the Turtle simulator, using the received sensor data. While both the TRC and the simulator could be run on another remote computer that is connected to the eNB via the Wi-Fi access point, in order to minimize additional latency and packet losses caused by Wi-Fi, the tools were executed on the eNB computer itself. This reduced functional view is described in Figure 5.4.

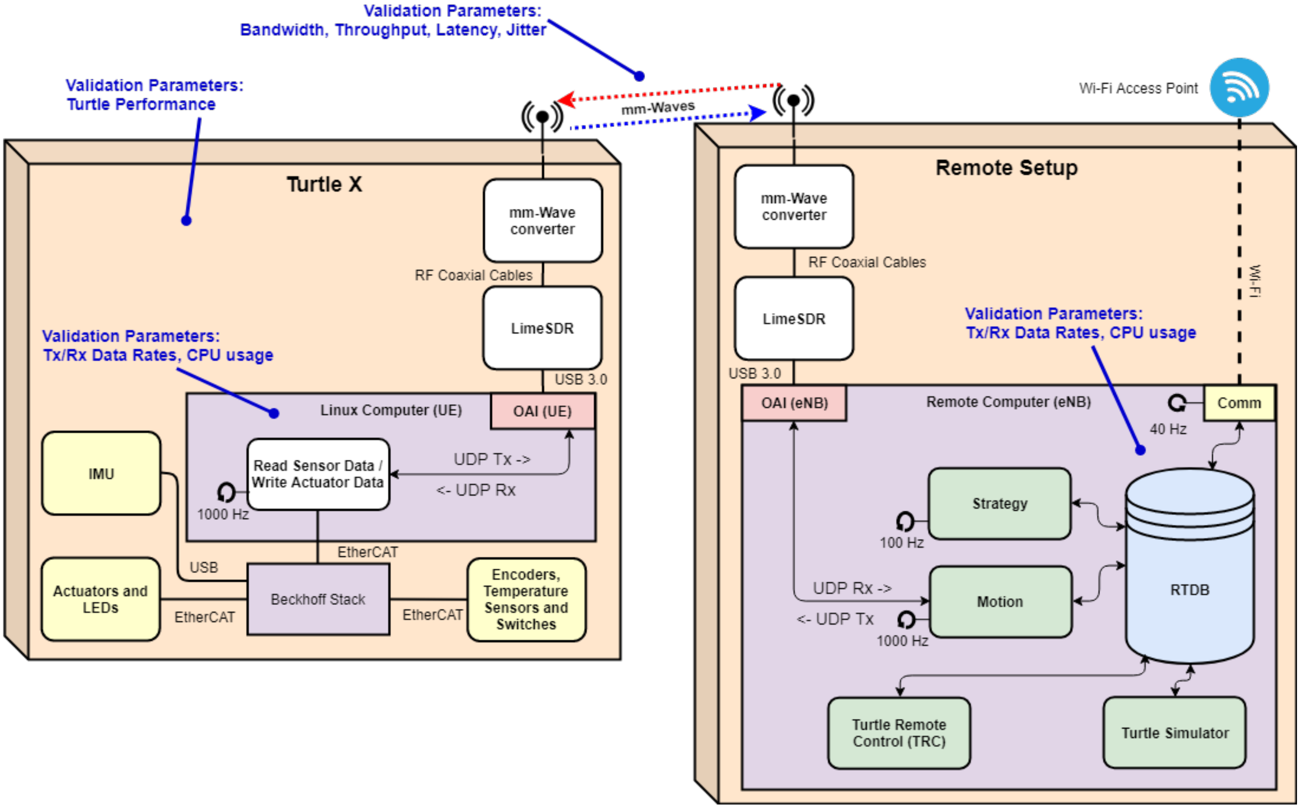


Figure 5.4: **Functional View - Implemented design**: This diagram is a reduced version of Figure 5.3 and offloads only the motion executable to the remote computer. The LimeSDRs on both the UE and the eNB side are attached to mm-wave converters, which up-convert 4G LTE band frequencies to the mm-wave band. Two new blocks (TRC and Turtle Simulator) are introduced on the eNB computer, which are used to provide manual commands to the Turtle and visualize received sensor data.

5.3.3 Validation Parameters

This subsection defines the validation parameters described in the Final Design and the Implemented Design. The validation parameters can be divided into three categories: Network parameters, Turtle software parameters and Turtle performance. The parameters within these categories are defined below:

Network Parameters: These parameters measure the performance of the network between the UE and the eNB computers.

- **Bandwidth:** The bandwidth or the data (bit) rate of the network is the maximum amount of data that can be transmitted from a sender to a receiver. Here, bandwidth is the maximum amount of data transmitted from the UE to the eNB and in the opposite direction.
- **Throughput:** This is the actual amount of data that is being transmitted, unlike the bandwidth which describes the capacity of the network. In this case, the expected throughput is the sum of the expected data rates from each sensor/image data stream.
- **Latency:** End-to-End (E2E) Latency is the total time it takes for a data packet to travel from one node to another. Round-trip latency is the total time it takes for a data packet to be transmitted and returned back to its source.
- **Jitter:** Jitter is the variance (in milliseconds) in the latency between data packets over the network. In other words, jitter is the deviation from true periodicity of a presumably periodic signal. This can cause a disruption in the normal sequence of sending and receiving data packets.

Turtle Software Parameters: These parameters are measured on the UE and the eNB computers and relate to the performance of the Turtle's software package.

- **Tx/Rx Data Rates:** Data rates in the Turtle software context refer to the rate at which data is being published or received by the Turtle software packages. For example, the World Model executable shares 127 Bytes of data with the RTDB at a frequency of 50 Hz. This amounts to a data rate of 1.016 Mbps. It is neccessary to measure the Tx/Rx data rates in order to ensure that software additions on either side (UDP Send, UDP Receive) are executed within one clock cycle of the update frequency.
- **CPU Usage:** CPU usage is measured both on the UE and the eNB in order to compare and analyze whether offloading of computational processes would result in a lower CPU usage on the UE computer.

Turtle Performance: In addition to verifying requirements like Data Rates and Latency, it is necessary to see if the overall performance of the robot remains the same or improves with 5G and offloaded computational processes. This is measured by comparing the performance of a Turtle without 5G mm-waves (all processes running on-board the Turtle) with the performance of a 5G mm-waves enabled Turtle. However, due to the use of only one robot and the implementation of a reduced design (as shown in Figure 5.4), this measurement is only planned but not implemented in the upcoming sections.

The following sections describe in detail how the above parameters are measured.

5.4 Conceptual View

This section answers the question: ***How* is the final product designed?**

The conceptual view describes the functional view in detail. More specifically, this section describes the communication interface, the UE and eNB software, a timing analysis of the sent/received data, and the validation methodology.

5.4.1 Communication Interfaces

To understand the communication interfaces in the design, it is important to look back to the OSI model in Figure 3.6. Every telecommunication system, including the one designed in this project can be abstracted into the seven layers of the OSI model. As shown in Figure 5.5, the OpenAirInterface software implements the Physical, Network and Data Link layers of the OSI model on both the UE and the eNB computers. The Turtle software on both computers implement the Application, Presentation and Session layers of the OSI model. The IP packets sent by the Turtle software on either side of the setup communicates with the transport layer implemented by the Linux operating system, which then sends the data to the OAI layer 3.

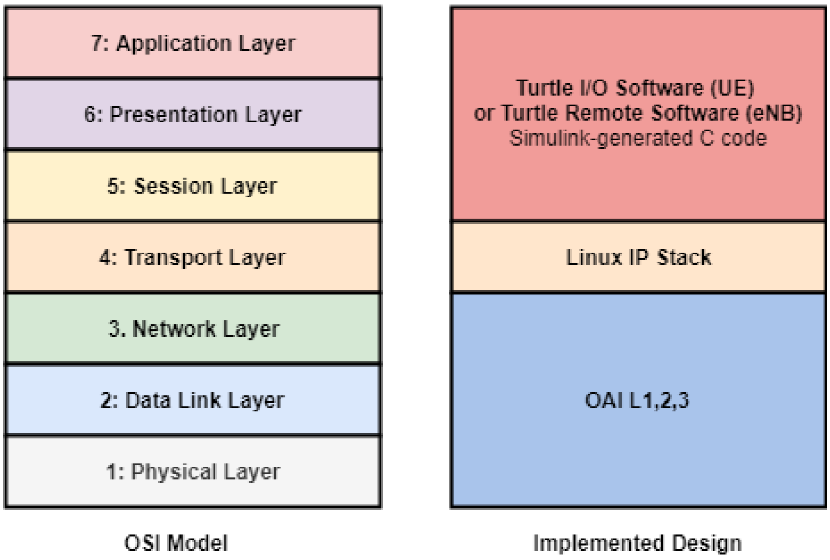


Figure 5.5: This diagram shows how the implemented design is mapped on to the OSI model. The OpenAirInterface software implements the first three layers of the OSI model, while the Turtle software works on layers 6, 7 and 8. The transport layer is implemented by the Linux IP stack.

OpenAirInterface, when built and executed on the Turtle and remote computers, generates IP addresses for the connected hardware and configures the computers as UE and eNB. The communication hardware connected on one computer establishes a connection with the communication hardware on the other side. This allows OAI to establish a local IP based network between the UE and the eNB. The existing Turtle software is then suitably modified to send data to the eNB's IP address over the mm-wave network and to receive data on the UE's IP address. The following Figures 5.6 and 5.7 describe the Conceptual View of the final design and the reduced scope design implemented in this project. Both diagrams also show the details of the data being transferred over the network.

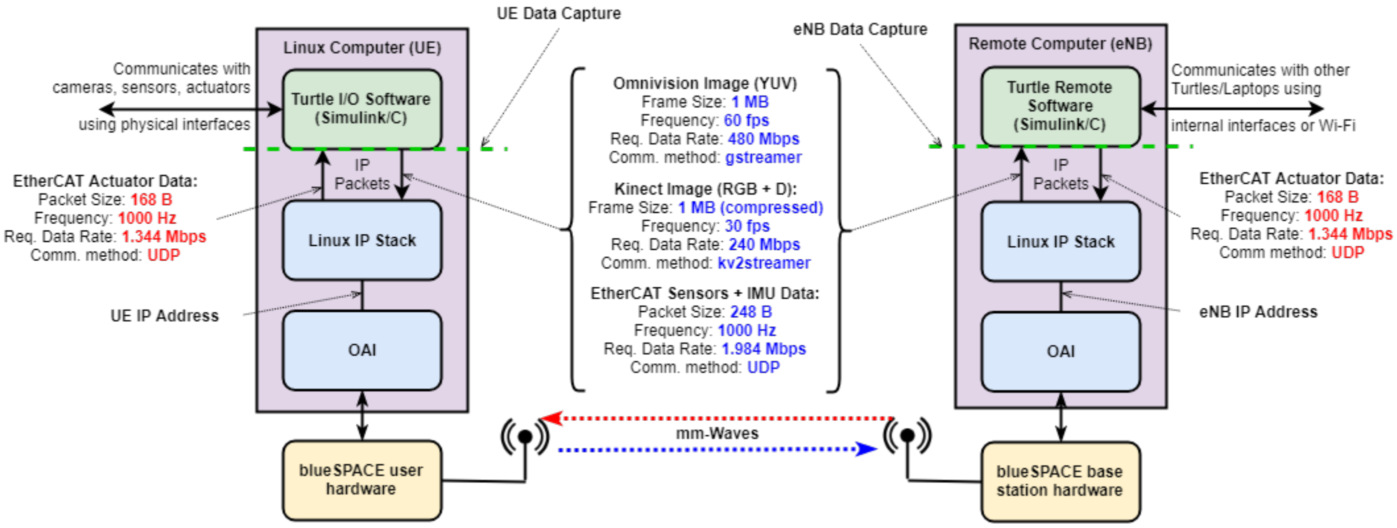


Figure 5.6: **Conceptual View - Final Design:** This diagram describes the 5G mm-Wave communication interface of the final design. The blue values describe data sent from the UE to the eNB and the red values indicate data in the opposite direction. The diagram also points out locations where the UE and the eNB IP addresses are generated as well as points where the validation parameters are captured and recorded.

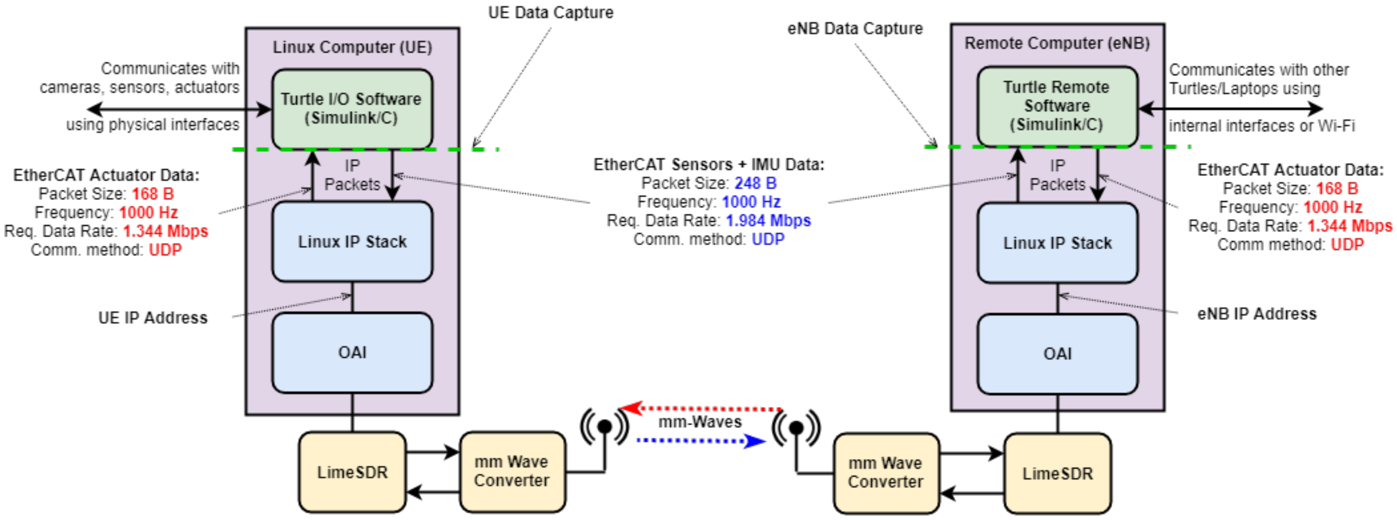

UE Data Capture
eNB Data Capture
Linux Computer (UE)
Remote Computer (eNB)
Communicates with cameras, sensors, actuators using physical interfaces
Turtle I/O Software (Simulink/C)
Turtle Remote Software (Simulink/C)
Communicates with other Turtles/Laptops using internal interfaces or Wi-Fi
IP Packets
EtherCAT Actuator Data:
Packet Size: 168 B
Frequency: 1000 Hz
Req. Data Rate: 1.344 Mbps
Comm. method: UDP
EtherCAT Sensors + IMU Data:
Packet Size: 248 B
Frequency: 1000 Hz
Req. Data Rate: 1.984 Mbps
Comm. method: UDP
Linux IP Stack
UE IP Address
eNB IP Address
OAI
mm-Waves
LimeSDR
mm Wave Converter

Figure 5.7: **Conceptual View - Implemented Design:** This diagram describes the 5G mm-Wave communication interface of the implemented design. This diagram is a modification of Figure 5.7 where the data sent from the UE to the eNB is reduced to only the EtherCAT sensor data. This view also introduces the LimeSDR instead of the blueSPACE communication hardware. Since the LimeSDR provides 4G LTE signals, a mm-wave converter is used to up-convert signals to the mm-wave band.

5.4.2 Turtle I/O and Remote Software

The existing Turtle software can be divided into two categories: executables that interact with hardware and executables that don't. The former category involves the Motion, Vision and Kinect executables that interact with the Beckhoff Stack, the Omnivision camera and the Kinect v2 respectively. The second category involves the World Model and Strategy executables that interact with only the RTDB for their input and output values.This section describes how the Motion, Vision and Kinect executables are broken down and the rationale behind the communication method used.

Vision and Kinect Executables

The existing Vision and Kinect executables are both periodic functions that acquire data from the respective camera, process it and update resulting records RTDB on each clock cycle (30Hz for Kinect and 60 Hz for Vision). To decouple data acquisition from the processing, each executable is broken down into two executables, one that acquires the data and streams it, and the other that receives the stream and processes the data. The data stream needs to be transmitted over the mm-Wave setup, which requires an IP based communication methodology, as shown in Figure 5.6.

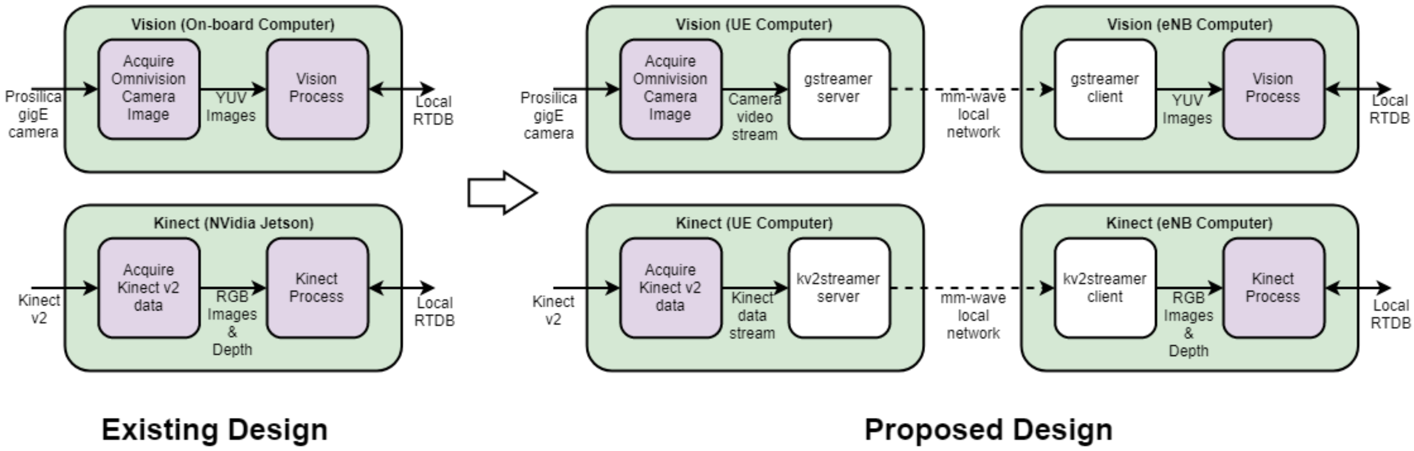


Figure 5.8: This diagram describes how the Vision and Kinect executables are broken down compared to the existing design. In the existing design, the Kinect process is executed on the NVidia Jetson and the Vision process on the Turtle computer. In the proposed design, both processes are executed on the UE computer or the eNB computer.

The two methods used for the Vision and Kinect executables are **gstreamer** and **kv2streamer**. Gstreamer is a commonly used open-source multimedia library that allows for efficient video streaming over IP based networks. On the remote side, the stream can be accessed over the network using Simulink or C code and images can be extracted for processing in the Vision executable. Similarly, kv2streamer is an open-source library developed by Samsung to allow developers to stream Kinect v2 data, including both RGB images and the depth data. It initializes a server and client side application that streams and receives data respectively. These applications can be accessed/modified using C code. Both these methodologies are state-of-the-art and are very commonly used in camera/Kinect streaming applications.

Motion and UDP Communication

The Motion block is broken down similarly to the Vision and Kinect processes explained above. The Motion executable is a periodic function (1000 Hz) that acquires data from the Beckhoff stack using EtherCAT, processes it and writes actuation commands on the EtherCAT frame, which is sent back to the Beckhoff stack. So, unlike the Vision and Kinect processes, the Motion executable involves two-way communication. The executable is broken down into Turtle and Remote side executables as seen below:

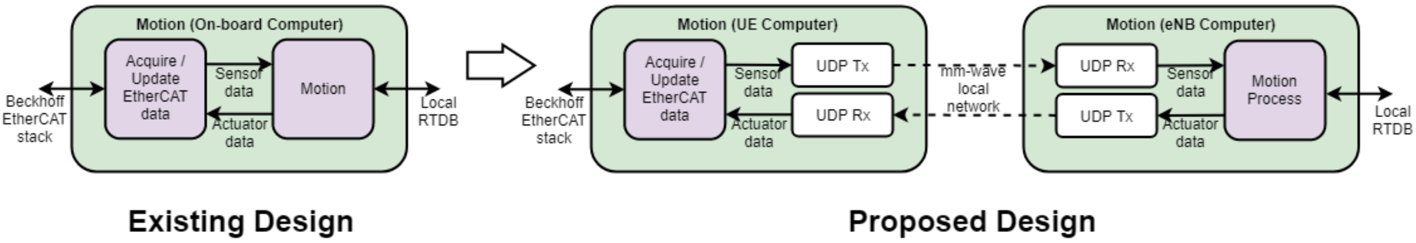


Figure 5.9: This diagram describes how the Motion executable is down compared to the existing design.

The shared data is an array of variables that either contains sensor values (UE to eNB) or actuation values (eNB to UE). The communication here can be achieved using existing IP based protocols instead of additional libraries used for the Vision and Kinect processes. Two of the commonly used protocols are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP is known as a connection oriented protocol, which means that TCP tracks all the transmitted data, requiring an acknowledgement from the receiver if the packet has been received. On the other hand, UDP is a connection-less protocol, which means the data keeps being sent, even if the receiver is unable to receive it. Since, no acknowledgement packets are sent, UDP generally provides a faster speed of transfer than TCP. So, while TCP is suited for applications that require high reliability, UDP is best suited for applications that need fast, efficient transmission. In this case, transmission speed is critical and hence, UDP is used.

Since bi-directional communication is used, reliability is not an issue. If TCP was used and the connection was lost, the UE would not receive acknowledgements from the eNB, and the Turtle would stop. On the other hand, with UDP, if the eNB is unable to receive data, the UE is unaware and still keeps sending data. However, the UE is also unable to receive data from the remote computer and the Turtle would still stop. In this case, the UDP connection acts like TCP with the actuation commands sent by the remote computer functioning as acknowledgements for the received data.

5.4.3 Timing Analysis

By breaking down the Vision, Kinect and Motion executables and introducing a mm-wave network between them, a latency of up to 1 ms is expected between sending and receiving data. For the Vision and Kinect blocks, which acquire and process data every 16.66 ms (60 Hz) and 33.33 ms (30 Hz), a 1 ms delay between acquiring and processing the data is not consequential. However, for the Motion executable, which runs every 1 ms, the latency is a substantial delay. So, it is important to understand how this latency affects the Turtle's performance. Figure 5.10 describes a timing analysis of the existing system.

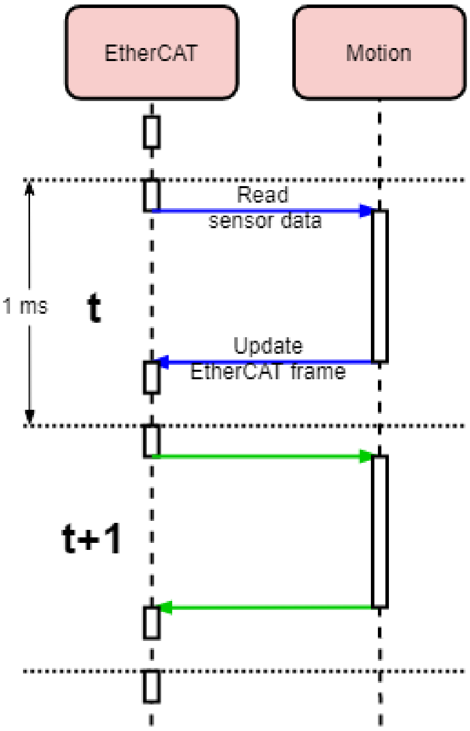


Figure 5.10: **Timing Analysis - Existing Design:** This diagram shows two full consecutive clock cycles of 1 ms each. In each clock cycle, sensor data is read, processed and actuation data is returned to the EtherCAT stack.

In the existing system, the EtherCAT frame is read, processed and updated in the same clock cycle. However, the timing changes when the mm-wave interface is implemented and a 1 ms latency is introduced. As seen in Figure 5.11, for the data sent at clock cycle t from the UE, the corresponding actuation data is received at clock cycle $t + 3$. This describes a round-trip time of around 3 ms, or a delay of 3 clock cycles. Subtracting a processing time for the motion process, the round trip latency is 2 ms. However, it was noticed that the motion controllers currently used by the Turtles have a tolerance of around 5-7 ms, which allows the Turtle to function without affecting its performance.

It is important to note that this is the worst-case scenario. Since an extremely low data rate is required, the E2E latency is expected to be much lower than 1 ms, as the blueSPACE KPIs suggest. This means the the round trip latency is expected to be much lower than 3 ms.

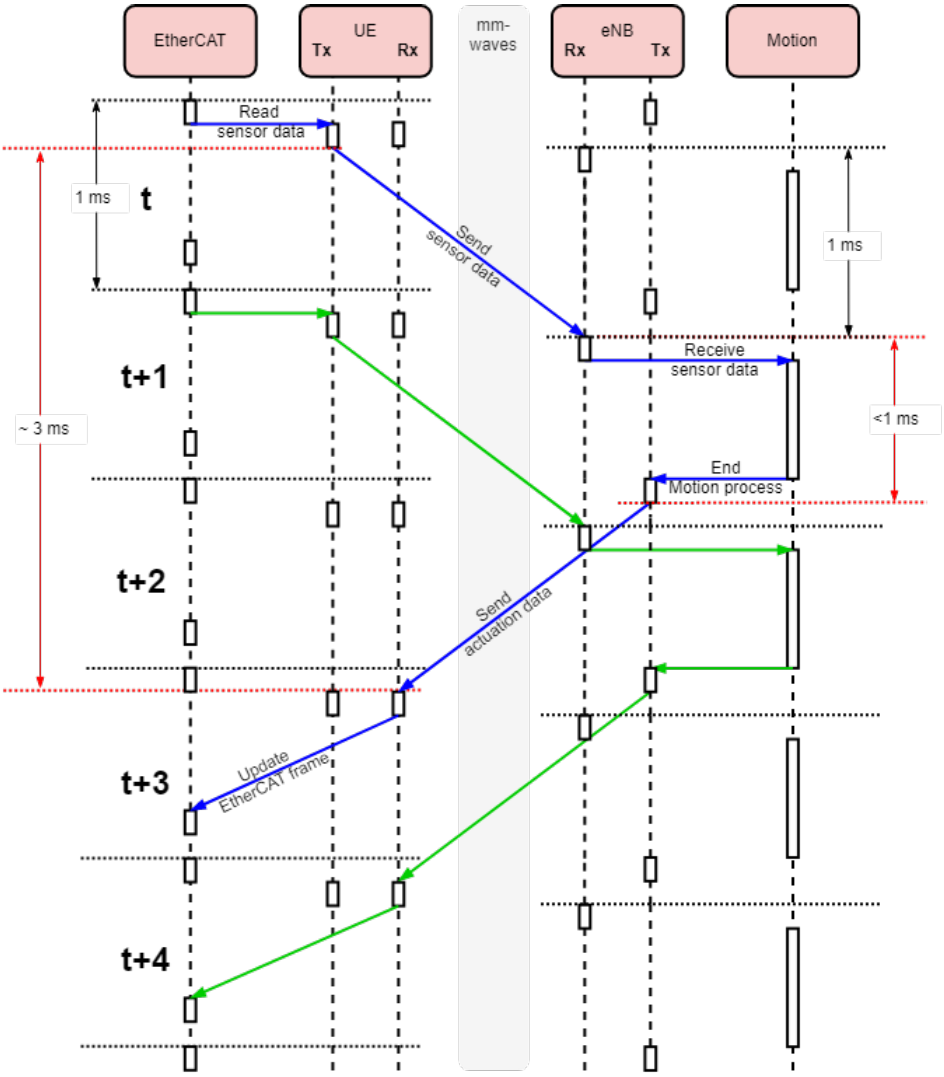


Figure 5.11: **Timing Analysis - Proposed Design:** This diagram shows five full clock cycles from t to $t + 4$. However, due to the latency introduced by the mm-wave communication, it can be seen that for the data sent on clock cycle t, the corresponding data is only received at clock cycle $t + 3$ and then sent to the EtherCAT stack in the same clock cycle. However, since the robot can tolerate a round-trip delay of around 5-7 ms, this delay of approximately 3 ms is tolerable.

5.4.4 Validation Methodology

Once the system is implemented, the validation parameters need to be measured. This is done using timestamps. During each experiment, data being sent or received from each computer is recorded with a timestamp accurate to 1 ns using commercial tools like Wireshark [41]. Once an experiment is completed, the recorded data is analyzed and compared to the timing diagram as shown in Figure 5.11.

Network and Turtle Software Parameters

Using the timestamps, the following parameters can be ascertained:

- **Data update rate:** Absolute difference between the timestamps at which data is sent from UE to eNB on consecutive clock cycles (measured at UE) *OR* Absolute difference between the timestamps at which data is received from UE on consecutive clock cycles (measured at eNB)
- **Round trip latency:** Difference between the time it takes for the UE to send and receive data (measured at UE), and the time it takes the eNB to receive and send data (measured at eNB)
- **End-to-End latency:** Cannot be calculated using the timestamps alone, but is assumed to be half of the Round trip latency.
- **Jitter:** Variance in the round trip latencies
- **Data rate (uplink)** Average number of bytes sent per second from UE (measured at UE)
- **Data rate (downlink)** Average number of bytes sent per second from eNB (measured at eNB)

Additionally, network analyzer tools like iPerf [42] are used analyze the network and determine the other parameters. These tools send controlled dummy data over the network and analyze the performance for multiple user-defined data rates. The following parameters are analyzed using such tools:

- **Packet loss:** Packet loss measured when 1.984 Mbps of dummy data is sent over the network (UE to eNB, measured at UE) *AND* Packet loss measured when 1.344 Mbps of dummy data is sent over the network (eNB to UE, measured at eNB)
- **Bandwidth:** Maximum data rate that can be sent over the network
- **Jitter:** Measured jitter when expected data rates (1.984 Mbps and 1.344 Mbps) are sent from UE to eNB and eNB to UE

End-to-End Latency

None of the above methodologies measure E2E latency - either from UE to eNB or from eNB to UE. This can be measured by calculating the difference between the timestamp at which data is sent from UE (measured at UE) and the timestamp at which the same data is received at eNB (measured at eNB). Since data is being measured on two separate computers, the timestamps are relative to the system clocks on the two computers. So, for an accurate E2E latency measurement, it is important to either synchronize the two computers to the same clock or accurately measure the time difference between the two clocks.

One way of doing this by using the Network Time Protocol (NTP) [43]. NTP is a network protocol used for clock synchronization between computer systems. It uses a client-server model, where the client synchronizes its system time with a server clock. The client can also be a server for another client computer. The source of the clock, which is usually an atomic clock or a GPS clock is known as Stratum 1. The clients that synchronize to these clocks are Stratum 2 clocks. Stratum 2 clocks are usually freely available servers on the internet to which today's computers synchronize to keep the time. So, an eNB synchronized to such a clock becomes a Stratum 3 clock. The UE is synchronized to this eNB server as Stratum 4. NTP can usually maintain time to within tens of milliseconds over the internet, but over a local network, a much better accuracy can be expected. However, for this, the eNB needs to be configured as Stratum 1 and the UE as Stratum 2. In this case, the UE synchronizes to a software generated clock on the eNB. The accuracy in this case can reach the sub-millisecond level, but only if the Stratum 1 clock does not drift. Software generated clocks are known to drift, so a sub-millisecond accuracy is only possible if different highly accurate clock source is used.

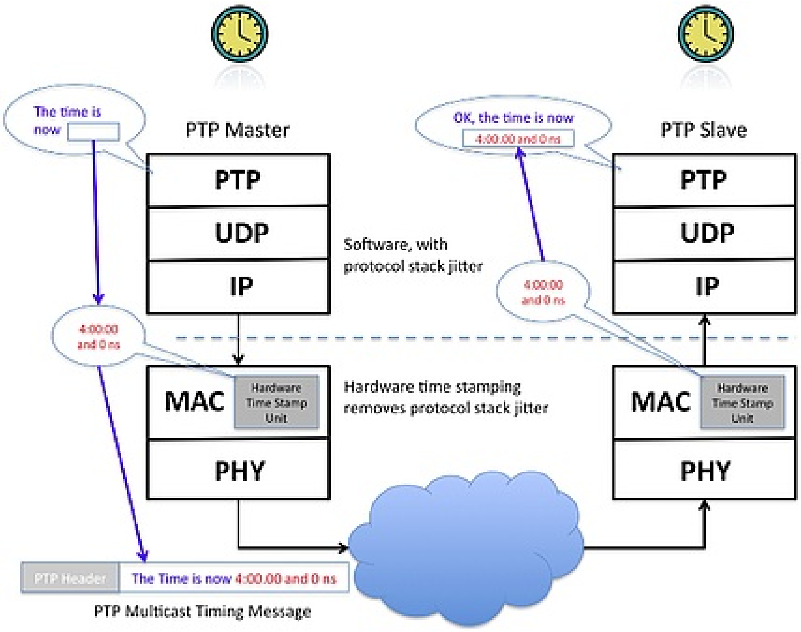

The time is now
PTP Master
PTP
UDP
IP
Software, with protocol stack jitter
OK, the time is now 4:00.00 and 0 ns
PTP Slave
PTP
UDP
IP
4:00:00 and 0 ns
4:00:00 and 0 ns
MAC
Hardware Time Stamp Unit
Hardware time stamping removes protocol stack jitter
MAC
Hardware Time Stamp Unit
PHY
PHY
PTP Header
The Time is now 4:00.00 and 0 ns
PTP Multicast Timing Message

Figure 5.12: This diagram explaines the Precision Time Protocol or PTP. It can be seen how hardware timestamps are used to synchronize the time between the PTP master and the PTP slave. In this project, the PTP master is a computer or networking equipment with a grandmaster clock and PTP slave is the eNB computer [44].

One way of solving this problem is by using the Precision Time Protocol (PTP) [45]. PTP, or IEEE 1588 is a protocol that provides clock synchronization accuracy in the sub-microsecond range. This is done by synchronizing a highly accurate clock source like GPS with a clock that provides hardware timestamps, like the ones generated by a computer's network interface card (NIC), instead of the system clock that uses software timestamps. Hardware timestamps are much more accurate than software timestamps and do not drift. These clocks that provide the accurate time source are called Grandmaster clocks, which the eNB can be synchronized to using PTP. Now, synchronizing the UE to the eNB using NTP would provide much more accurate timestamps, and hence a much more accurate measure of the E2E latency.

Turtle Performance

Once the above measurements are carried out with sufficient results, the Turtle performance can then be measured. The performance is measured by comparing the performance of two Turtles - one with mm-wave communication, one without. On the eNB, the Turtle simulator is executed to visualize the data read from the mm-wave Turtle as well as the data received from the normal Turtle over Wi-Fi. Comparison is made my measuring response times and accuracy of both Turtles when they are made to perform the same actions using manual commands. A validation test setup can be seen in the image below.

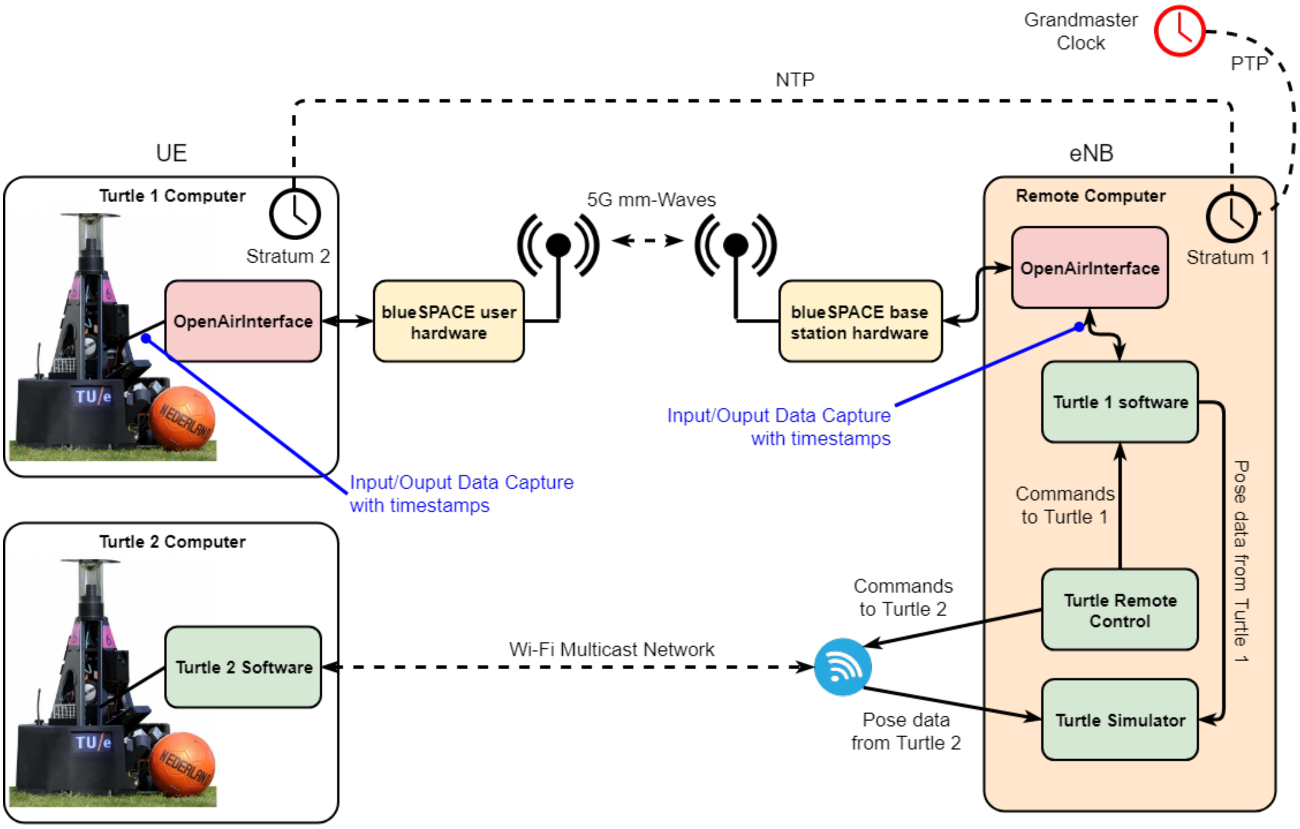


Figure 5.13: Conceptual View - Validation setup: This diagram shows how the validation experiments are set up. However, due to acquisition delays, the blueSPACE hardware is replaced by a LimeSDR attached to a mm-wave converter.

5.5 Realization View

This section answers the question: ***How* is the final product implemented?** The realization of the system can be divided into three subsystems: The Network, the Turtle (UE) and the Remote Computer (eNB) . The specifics of these subsystems are listed below.

5.5.1 mm-Wave Network

The LimeSDRs used in this project are interfaced with OAI installed on the UE and the eNB computers and are configured using separate configuration files. The connection is setup and validated in three stages - wired, wireless using 4G LTE, wireless using mm-wave converters as shown below.

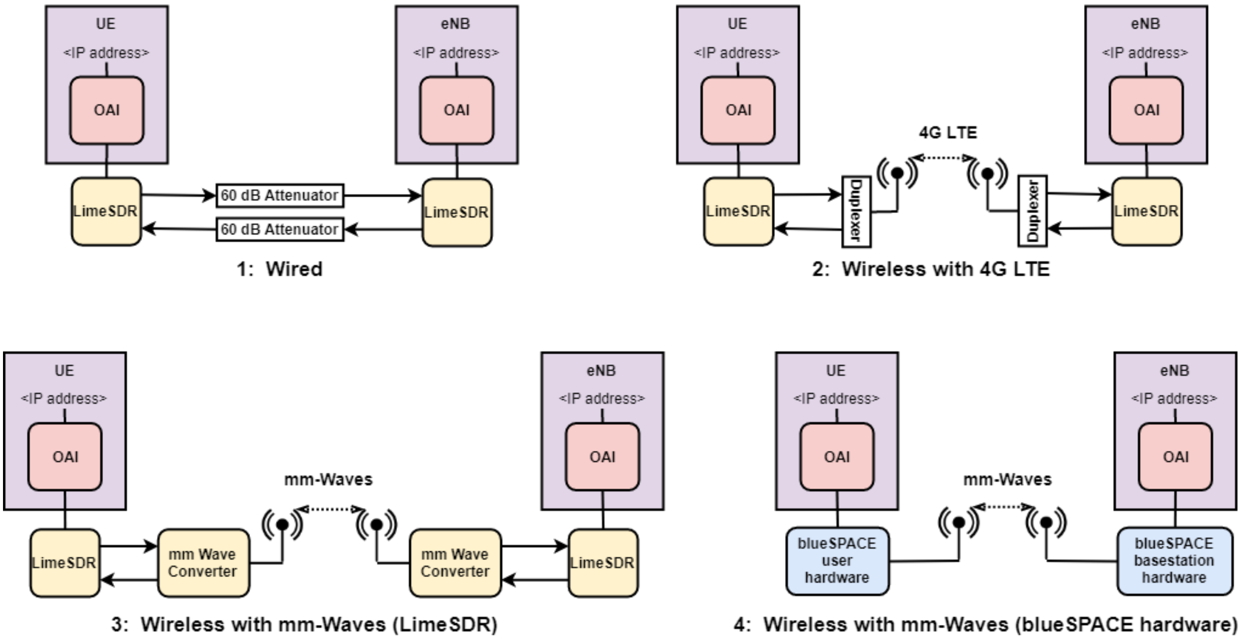


Figure 5.14: **Realization View - Communication Setup:** The 4 stages of implementing the wireless setup - 1: The wired interface uses coaxial cables and attenuators to simulate a wireless interface. 2: The wireless interface replaces the attenuators with a duplexer and an antenna on either side. 3: 4G LTE signals from setup 2 are converted to mm-waves and vice versa. 4: Once the blueSPACE hardware arrives and is configured, they are used instead of the LimeSDR in setup 3.

5.5.2 UE and eNB Computers

The existing on-board computer on the Turtle is an Industrial Beckhoff PC with an Intel Core2 Duo processor. It has the following interfaces:

- **1 ethernet port** for the EtherCAT connection
- **1 USB port** connected to the NVidia Jetson using a USB-ethernet adapter
- **1 gigabit ethernet** port for the Omnivision camera
- **1 internal Wi-Fi NIC** (network interface card)

However, the proposed system also requires additional ports that the Beckhoff PC does not have:

- **1 USB 3.0** connection for the Kinect v2 (since the NVidia Jetson is no longer used)
- **1 gigabit ethernet** port for the blueSPACE hardware
- **1 USB 3.0** port for the LimeSDR

In addition to these interfaces, the UE computer needs to run OAI, which needs an Intel i3 or higher processor. So, for the sake of this test setup, the on-board computer was replaced with a spare laptop (with an i7 processor). At the end of this project, an equivalent on-board computer alternative is proposed, so that the current on-board computer can be replaced.

On the eNB side, the computer also needs to run OAI and connect with the same communication hardware. Additionally, it needs to have a gigabit ethernet port if the blueSPACE hardware is used and 1 USB 3.0 port for the LimeSDR. Other than that, it needs to be powerful enough to execute the existing Turtle software for now. In the future, if more computation is required, the eNB laptop can be upgraded.

The UE and the eNB laptops are both configured with the following specifications:

- **Operating System:** Ubuntu 16.04
- **Kernel:** Low Latency Kernel v4.7.x or 4.8.x with Hyper-threading turned off
- **Tech United software:** MATLAB/Simulink r2016a, other Tech United devPC software [46]
- **Network tools:** OpenAirInterface5G, LimeSuite [47]
- **Validation tools:** Wireshark, iPerf, NTP

5.5.3 Integration

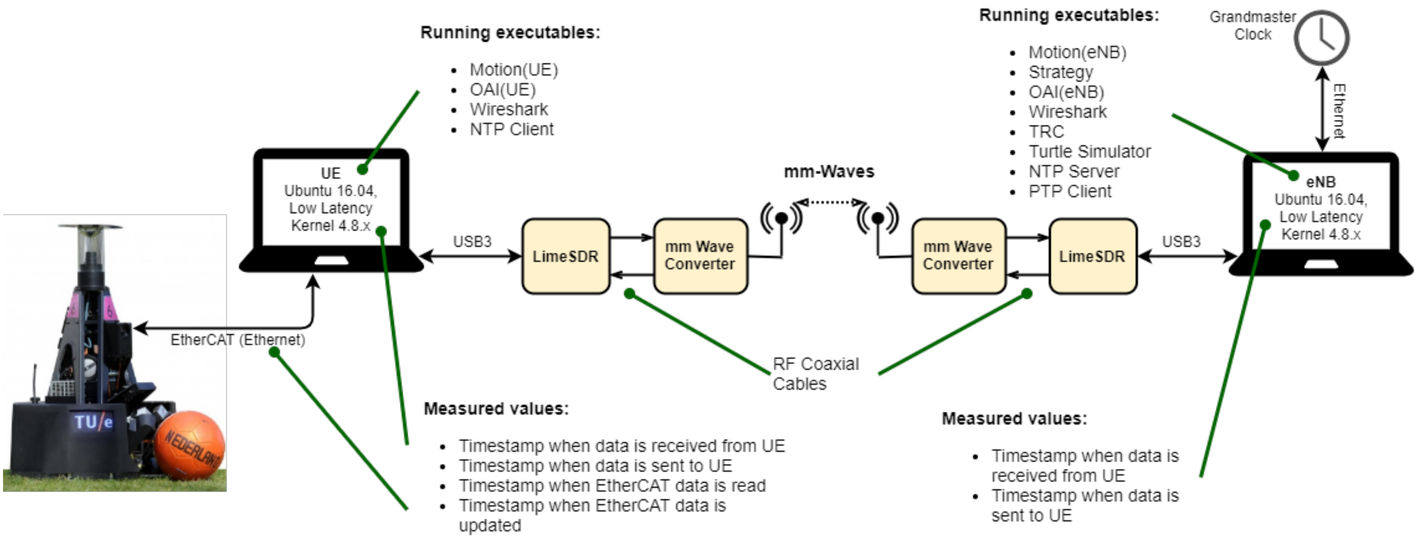


Figure 5.15: **Realization View - Integrated test setup:** This diagram shows how the test setup is integrated. This setup requires a Turtle, two Ubuntu 16.04 laptops configured as UE and eNB, and two LimeSDRs with mm-wave converters. Additionally, this diagram also defines the executables running on each computer and the measurements that are captured on either side

6 System and Subsystem Implementation

This chapter describes the Preparation and Realization Phases of the work phase plan. In order to implement the setup described in Figure 5.15, the implementation plan is broken down into three steps: Turtle I/O and Remote software, OAI implementation with LimeSDR, and integration. These subsystems are described in the following sections.

6.1 Turtle I/O and Remote Software

The first step towards implementing the Turtle software is the configuration of the UE and the eNB computers. Two identical HP Z-book laptops were used and configured with Linux 16.04, MATLAB/Simulink r2016a and other software required for developing, building and executing the Tech United software. In this project, only the motion block is offloaded to the remote computer. Before going ahead, it is important to see how the existing software works.

The Motion Simulink model is divided into two subsystems: High Level Control and Low Level Control. The low level control subsystem that can be seen in Figure 6.1 includes a block for data acquisition and provides input/output interfaces to the high level control (which is shown in Figure A.2 in Appendix A). The full motion block is replicated on two computers - the UE and the eNB computer.

On the UE computer, only the data acquisition block is executed and all other subsystems are removed. Once it is sufficiently proven that the generated code from this reduced model can be built and executed on the UE computer, the data acquisition block is connected to UDP Send and UDP Receive blocks as seen in Figure 6.2 and Figure 6.3. For reliability, if the length of the received signal is 0 (no signal is received), pre-determined actuation commands are sent to the Beckhoff stack so that the Turtle remains stationary.

The UE side of the software is tested by connecting the UE to the Beckhoff Stack directly, and validating if a connection to the EtherCAT slaves is achieved. Additionally, the UDP Send and receive blocks are tested by achieving a dummy UDP connection with a remote computer over wired/wireless LAN. Once the UE side is tested with the actual robot hardware and the communication interface on the application layer is verified, the eNB software is developed.

On the eNB side, the existing Turtle software is executed without the data acquisition block. The data acquisition block is replaced by a UDP Receive block (configured to receive data from the UDP Send block on the UE) and a UDP Send block (configured to send data to the UDP Receive block on the UE). For reliability on the eNB side, if no data is being received, pre-determined commands are sent to the UE in order to stop the Turtle. The eNB software is described in Figure 6.3.

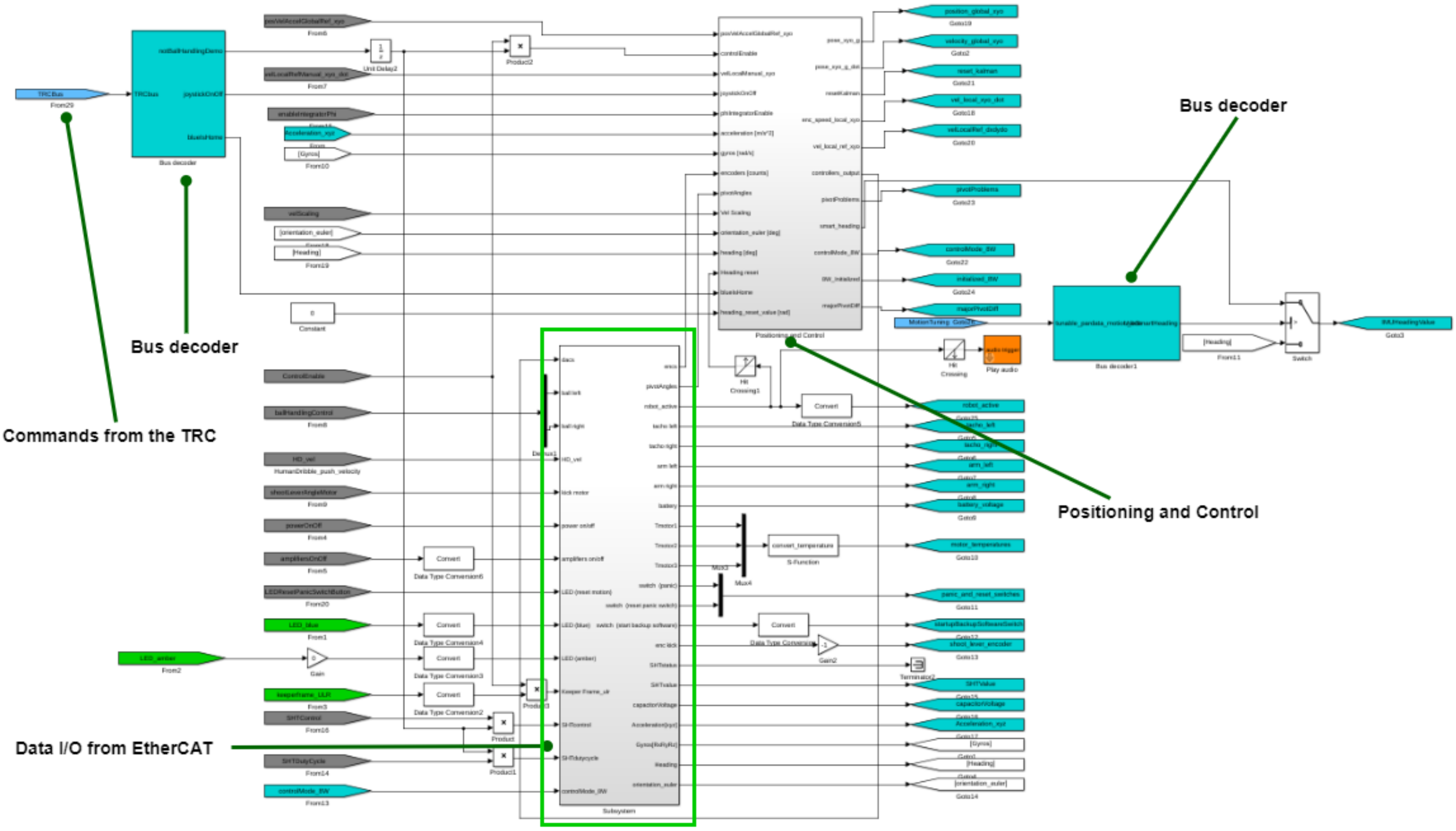


Figure 6.1: The existing low level control subsystem in the Motion block. The block involves sub-blocks that use acquired sensor data and the data from the high level control algorithms to control the position of the Turtle. The Turtle can also use manual commands from the Turtle Remote Control (TRC) as shown. In the next steps, the subsystem highlighted in green is removed and executed on the Turtle and the remaining blocks are executed on the eNB.

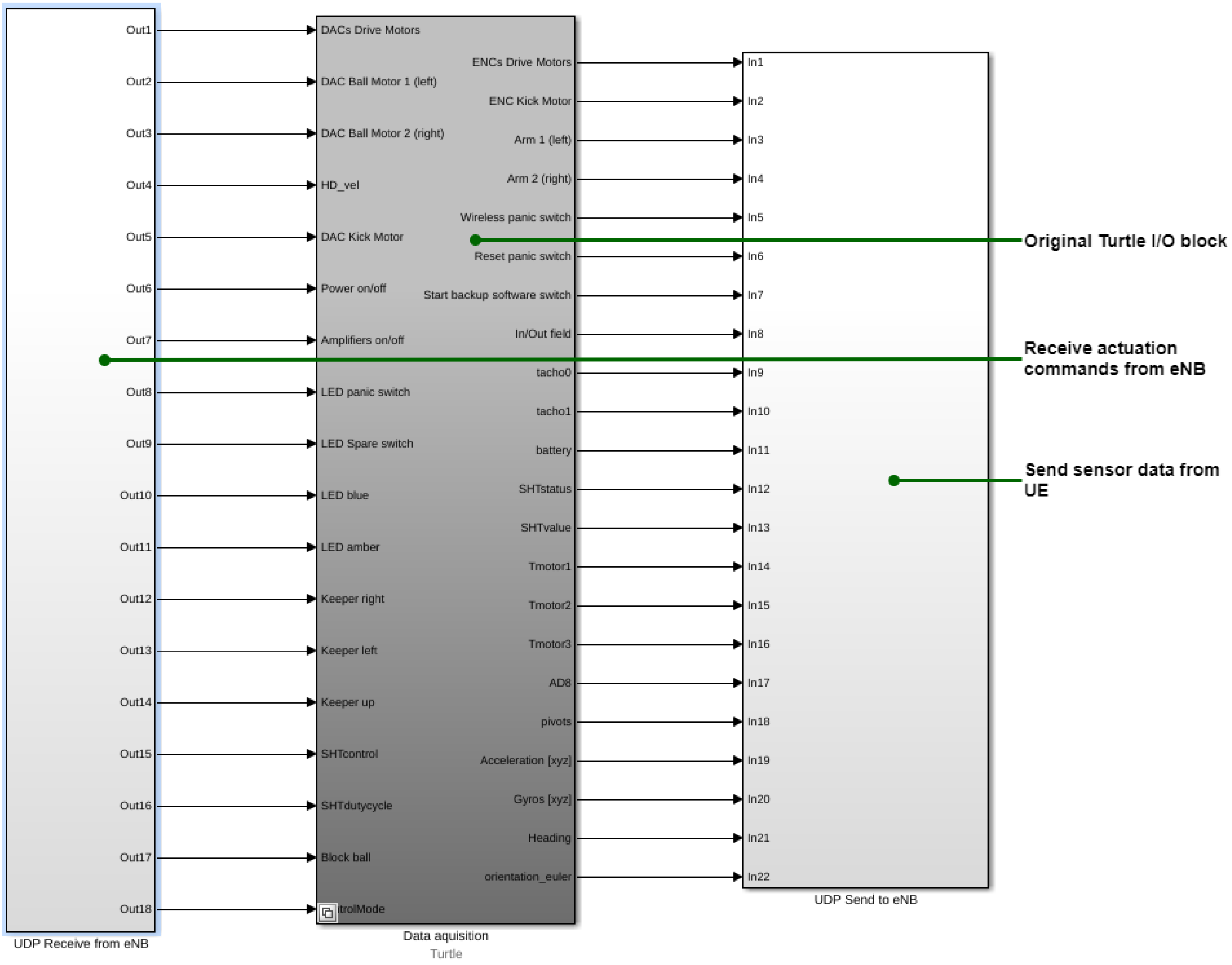


Figure 6.2: The implemented Motion block on the UE computer. Only the data acquisition subsystem is attached to UDP Send and UDP Receive blocks. The UDP Send block packs the acquired sensor data and sends data to the eNB IP address. The UDP Receive block unpacks the received data and sends the signals to the I/O subsystem which updates the EtherCAT frame. The UDP Send and UDP Receive subsystems are explained in detail in Appendix A.

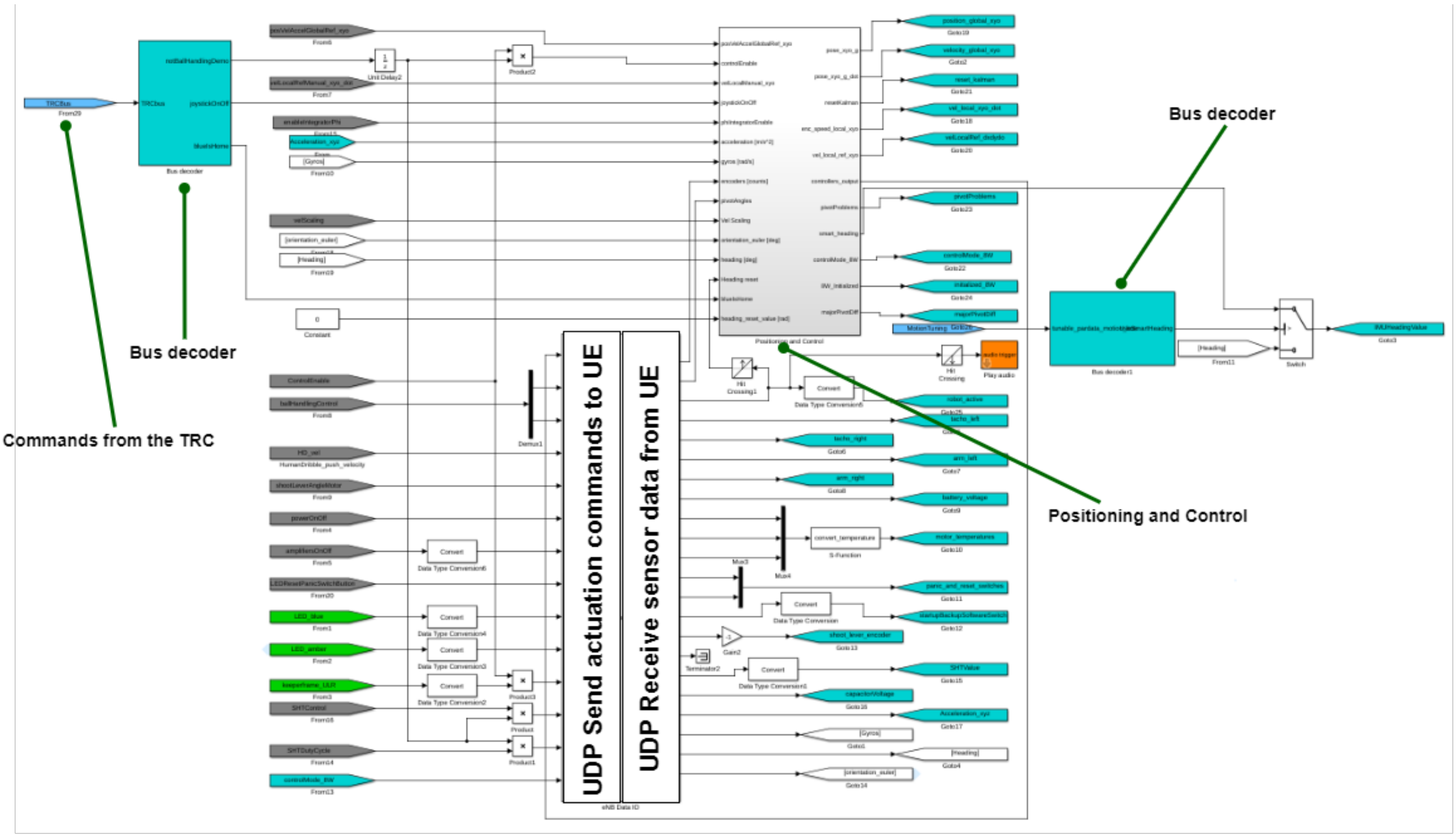


Figure 6.3: This simulink model describes an extension to the existing Motion block in Figure 6.1. Here, the data acquisition subsystem has been replaced with a block that contains both UDP Send and UDP Receive blocks. UDP Send packs the actuation commands generated by the control algorithms and sends them to the UE IP address. On the other hand, the UDP Receive block reads sensor data sent from the UE. The subsystem is explained in detail in Appendix A.

Once both sides of the software are setup and tested, the integrated communication is tested by connecting the UE to the Turtle and also to the eNB computer using an ethernet cable. The UE and eNB side software are executed and validated using the Turtle Remote Control (TRC) and the Turtle Simulator tools. Figures 6.4 and 6.5 describe these tools.

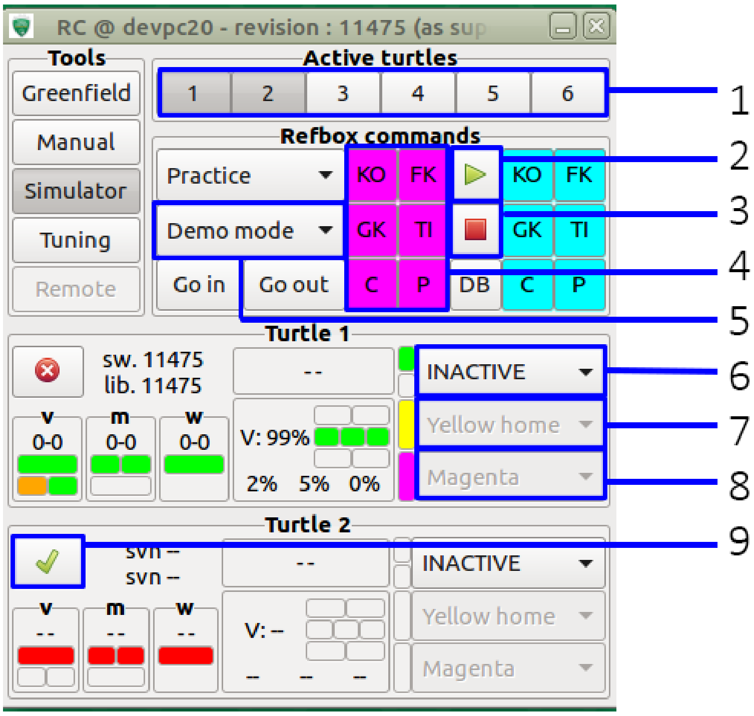


Figure 6.4: This figure shows the Turtle Remote Control (TRC) GUI. The highlighted numbers can be described as follows: **1:** Buttons to activate the robots in the TRC, **2:** Play, **3:** Stop, also the key 's' on the keyboard works, make sure you have always one finger on this button in case of emergency, **4:** Refbox tasks, KO: Kick Off, FK: Free Kick, GK: Goal Kick, TI: Trow In, C: Corner, P: Penalty, **5:** Here the different actions can be chosen, **6:** Role selection, Attacker, Defender, Goalkeeper etc., **7:** Home goal, Yellow and Blue, **8:** Team color, Magenta and Cyan, **9:** Button to activate the robot's software [7]

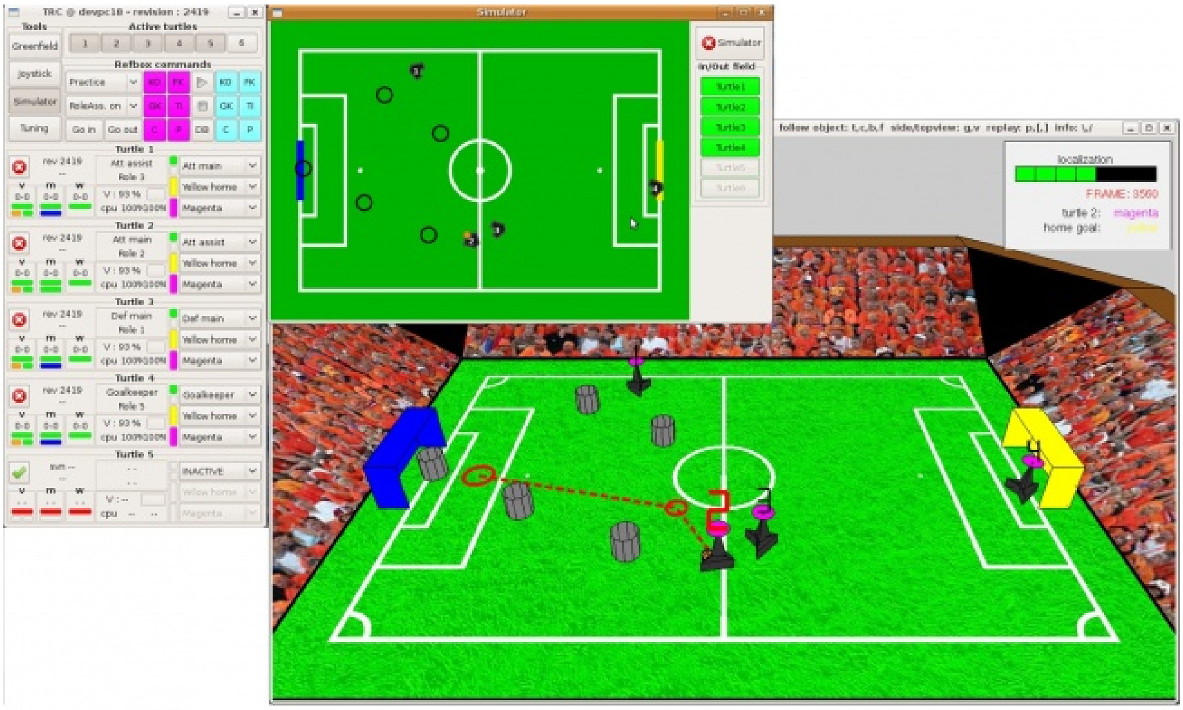
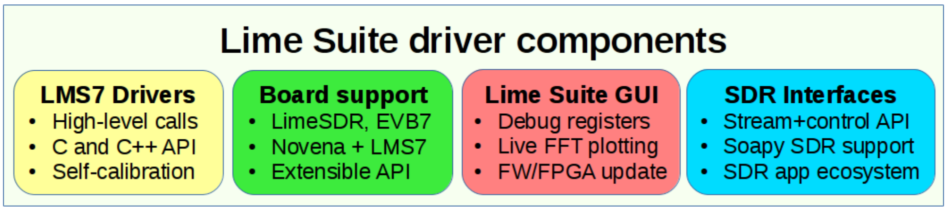


Figure 6.5: This figure is a screenshot of the Turtle Simulator in action. On the left is the TRC GUI, the image on the top describes the simulator used in this project and the image on the bottom describes the Greenfield simulator that provides a 3D simulation of the gameplay with additional information such as the status of the robots and the ball position. The Turtle simulator is able to show both the Tech United robots as well as opponent robots and the ball. In this project, only one robot was visualized by using the shared sensor data. [7]

6.2 OAI and LimeSDR

The second step of the integration is the implementation of the OAI-LimeSDR setup. First, the OpenAirInterface is set up on the UE and eNB computers. This requires a low latency kernel to be installed on the operating system of each computer, with hyper-threading turned off. Once configured, the LimeSDR software and dependencies are installed.



Lime Suite driver components

LMS7 Drivers
- High-level calls
- C and C++ API
- Self-calibration

Board support
- LimeSDR, EVB7
- Novena + LMS7
- Extensible API

Lime Suite GUI
- Debug registers
- Live FFT plotting
- FW/FPGA update

SDR Interfaces
- Stream+control API
- Soapy SDR support
- SDR app ecosystem

Figure 6.6: This image describes the software dependencies of the LimeSuite, a configuration GUI for the LimeSDR. As shown, the required drivers include not only drivers for the hardware but also dependencies for debugging and visualization. [47]

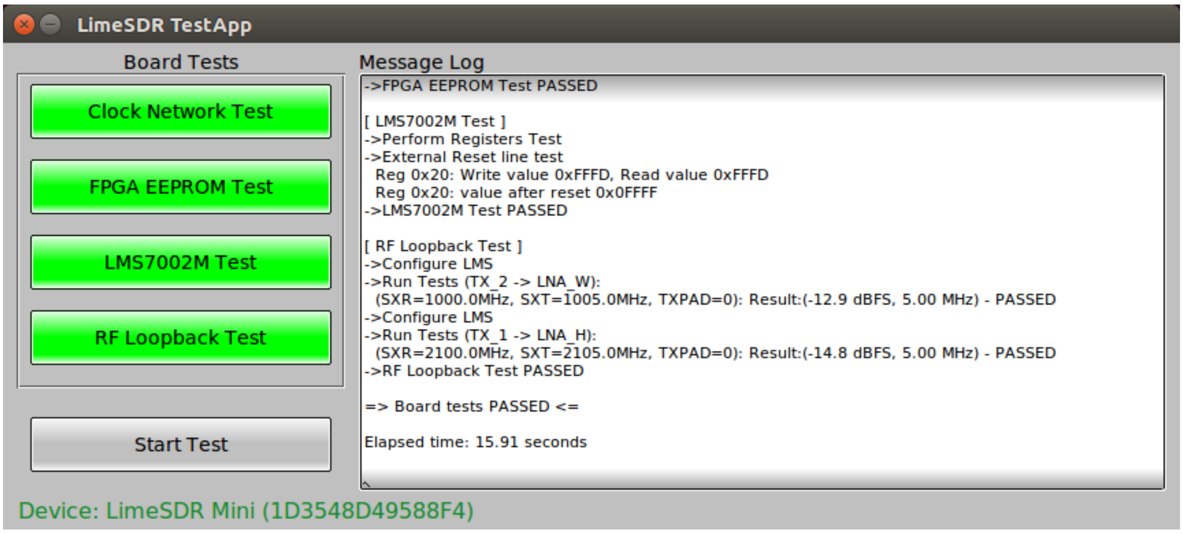
In order to make sure that the OAI-LimeSDR installation can work alongside the Turtle software, the UE and eNB tests from the previous section are conducted again. A few dependency issues arise and need to be resolved before going ahead.

- USB Library: The Turtle software installs *libusb-1.0.8* while the LimeSDR requires *libusb-1.0.0-dev* for communicating with the USB ports. To resolve this, *libusb-1.0.8* is removed and the Turtle software is built again successfully. The built code is run on the Turtle to confirm that it still functions the same.
- C Compiler Version: The Turtle software requires GCC v4.7 while OAI requires GCC v4.9. However, different versions can be installed together and chosen when required. So, the GCC v4.7 could be used to build the robot software and can then be easily switched to GCC v4.9 to build the OAI software.

Once all dependency issues are resolved, the LimeSDR can be tested individually on each computer and the communication network can be set up using OAI.

6.2.1 LimeSDR

Before using the hardware in the project, loopback, TX and RX tests are conducted with the LimeSDR. The LimeSDR comes with a dedicated testing utility called LimeQuickTest which carries out a number of checks on the hardware like the clock, the FPGA, the programmable radio IC and a RF loopback test. The loopback test consists of an arbitrary signal being sent by the LimeSDR from an output port and being read by an input port which is internally connected to the output port. This test usually takes around 15-20 seconds to complete and results in a PASS or a FAIL result. The following diagram shows the GUI output of the LimeQuickTest.

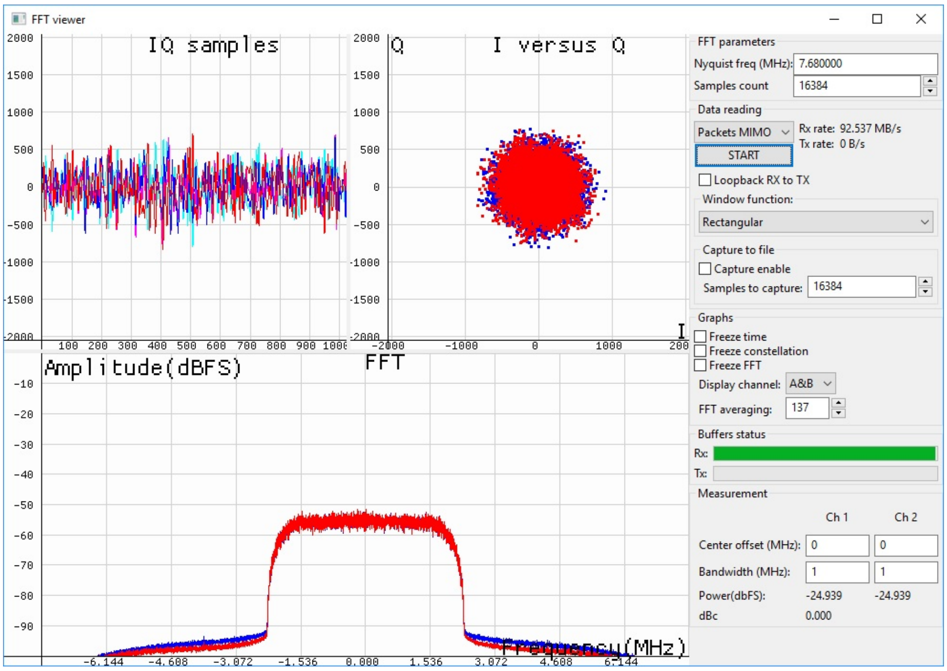


```
->FPGA EEPROM Test PASSED

[ LMS7002M Test ]
->Perform Registers Test
->External Reset line test
  Reg 0x20: Write value 0xFFFD, Read value 0xFFFD
  Reg 0x20: value after reset 0x0FFFF
->LMS7002M Test PASSED

[ RF Loopback Test ]
->Configure LMS
->Run Tests (TX_2 -> LNA_W):
  (SXR=1000.0MHz, SXT=1005.0MHz, TXPAD=0): Result:(-12.9 dBFS, 5.00 MHz) - PASSED
->Configure LMS
->Run Tests (TX_1 -> LNA_H):
  (SXR=2100.0MHz, SXT=2105.0MHz, TXPAD=0): Result:(-14.8 dBFS, 5.00 MHz) - PASSED
->RF Loopback Test PASSED

=> Board tests PASSED <=

Elapsed time: 15.91 seconds
```

Device: LimeSDR Mini (1D3548D49588F4)

Figure 6.7: This screenshot describes the LimeSDR quick test GUI. As shown, the test verifies the hardware's clock, the on-board FPGA, the RF hardware (LMS7002M) and performs a loopback test. While this specific image is for the LimeSDR Mini, the LimeSDR-USB used in this project uses the same quick test GUI for performing initial tests. [48]

In order to test the LimeSDR further, another loopback test is conducted using the LimeSuite GUI. Using this application, the LimeSDR is connected to the computer and an example configuration file is loaded. The configuration file consists of initialization parameters for the LimeSuite, which are then programmed onto the LimeSDR board. This configuration file configures the Tx/Rx frequencies, the clock cycle and also configures an internal loopback between a Tx and a Rx port. In this test, a test waveform is sent via the Tx port and read by the Rx port. This test signal can be visualied using the FFT viewer, and can be seen in the following figure.



FFT parameters

Nyquist freq (MHz): 7.680000

Samples count 16384

Data reading

Packets MIMO

Rx rate: 92.537 MB/s

Tx rate: 0 B/s

□ Loopback RX to TX

Window function: Rectangular

Capture to file

□ Capture enable

Samples to capture: 16384

Graphs

□ Freeze time

□ Freeze constellation

□ Freeze FFT

Display channel: A&B

FFT averaging: 137

Buffers status

Rx: 🟩

Tx:

Measurement

	Ch 1	Ch 2
Center offset (MHz):	0	0
Bandwidth (MHz):	1	1
Power(dbFS):	-24.939	-24.939
dBc	0.000	

Figure 6.8: This image describes the results of a more comprehensive initial test for the LimeSDR. This image is a screenshot of the FFT Viewer from the LimeSuite software. The red and blue signals show transmitted and received signals respectively. The IQ samples describe the change in the amplitude of the sent/received waves. The I vs Q graph plots the magnitude and phase of the sent/received signals in a polar coordinate frame. [49]

6.2.2 OAI

OAI is implemented on both the UE and the eNB. Once the low latency kernel is installed and hyperthreading is turned off, the OpenAirInterface5G software can be installed. Next, this software is used to build and configure the two computers. A general OAI based 5G setup consists of three computer - the UE, the eNB and the Core Network (CN). However, the OAI documentation also provides alternatives such that only the UE-eNB connection can be established [50].

For the setup, OAI uses two configuration files. The first configuration file configures OAI to interface with the attached SDR and sets up IP addresses on both the computers. The second file configures the SDR to send/receive the specified frequencies and to synchronize with the UE or the eNB.

In this project, both used configuration files were made available in the OAI documentation. However, the first configuration file is configured with the USRP B210 software defined radio and needs to be modified to interface with the LimeSDR. The second configuration file is specific to the LimeSDR and easily available online.

6.2.3 mm-Wave network

The next step in the process is to set up the mm-wave network. This is done using the setups shown in Figure 5.14. According to the OAI documentation, the USRP B210 SDR configuration file is used and modified to work with the LimeSDR. This configuration file defines the carrier frequency (2.68 GHz for the USRP B210 and 2.66 GHz for the LimeSDR), timing parameters and also the IP addresses for the UE and the eNB.

Once setup 1 from Figure 5.14 is connected, the eNB is configured using the first configuration file. When executed, the eNB computer opens a scope that can be used to read the sent and received data over the network. The second step is to configure the UE with the second configuration file, and then run the UE executable. The UE executable scans around the eNB carrier frequency for upling and downlink channels and synchronizes the UE to the eNB. This establishes a local network between the UE and the eNB IP addresses.

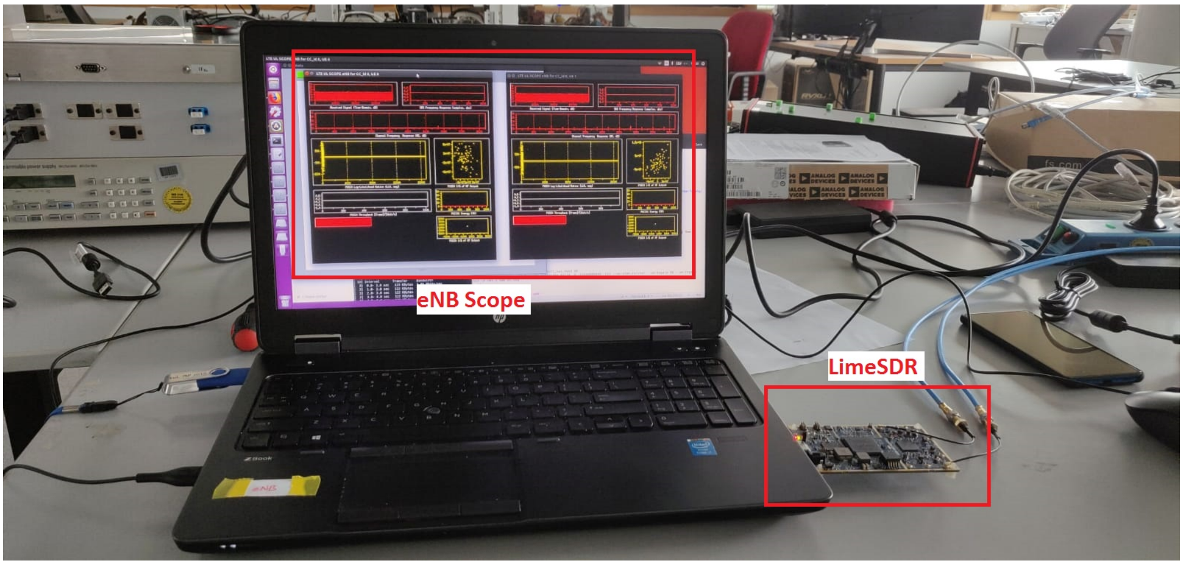


Figure 6.9: This setup shows the LimSDR-USB connected to the eNB computer via USB 3.0. The LimeSDR is connected to the SDR on the UE side using RF coaxial cables. The screen of the eNB shows a scope provided by OAI to visualize received and transmitted data.

However, during implementation, it was found that while the UE is able to receive commands from the eNB and the eNB is able to receive commands from the UE, both the computers are not actually synchronized. This means that while there was some data sent and received on either side, the data could not be comprehended without synchronization. Despite multiple efforts to try and find a solution (extensive study of the LimeSDR and OAI forums/mailing lists, and modification of the configuration parameters to try and understand what is going wrong), the connection could not be established within the project's time-frame. Therefore, a backup network was implemented and is explained in the next section.

6.3 Integrated Setup

Since the LimeSDRs could not be synchronized and the blueSPACE hardware was delayed, the mm-wave network could not be set up. However, the UE and the eNB side Turtle software needs to be validated using a mobile network and the network validation methodology needs to be proven. In order to perform this validation, a backup setup was developed to replace the LimeSDR network. This backup network was developed and implemented by members/graduate students in blueSPACE.

In this setup, both UE and eNB computers are connected to Media Converters using ethernet cables. The media converters convert IP data packets into optical signals and vice versa. These optical signals from the UE are sent via optical fibers to an antenna which transmits the message with a transmission frequency of: 27GHz center frequency, with a transmission in the band from 26GHz to 28GHz.

On the remote side, an antenna receives the message and using electronics, converts it into optical signals which are then sent to the media converter connected to the eNB computer.

On the other hand, the data packets from the eNB are converted to optical signals using the media converter and sent using an optical fiber cable to the media converter attached the UE computer. While this wired method was chosen to simplify the hardware setup due to time constraints, it can also be used to compare network performance between the wired and the wireless routes.

The media converters use modules known as SFPs (small form-factor pluggables). These interface modules are used in telecommunication applications and here, they are used as an interface between the media converters and fiber optic cables. However, in order to function, these SFPs require both an input and an output. While both the media converters have an input and an output, there is a third SFP used in the electronics on the remote side. This SFP only outputs the UE message to the eNB media converter. Since it cannot function without an input, the eNB media converter output is provided as input. These inputs are terminated and are solely for the SFP to function.

This setup was implemented in a Faraday cage, which is an enclosed room that blocks any external electromagnetic signals. Due to to this, a GPS could not be used as the PTP Grandmaster clock. The alternatives are expensive network equipment that need to be acquired. Despite the lack of PTP, the system clocks of both computers are synchronized using the Network Time Protocol. This is shown in the setup diagram in Figure 6.10.

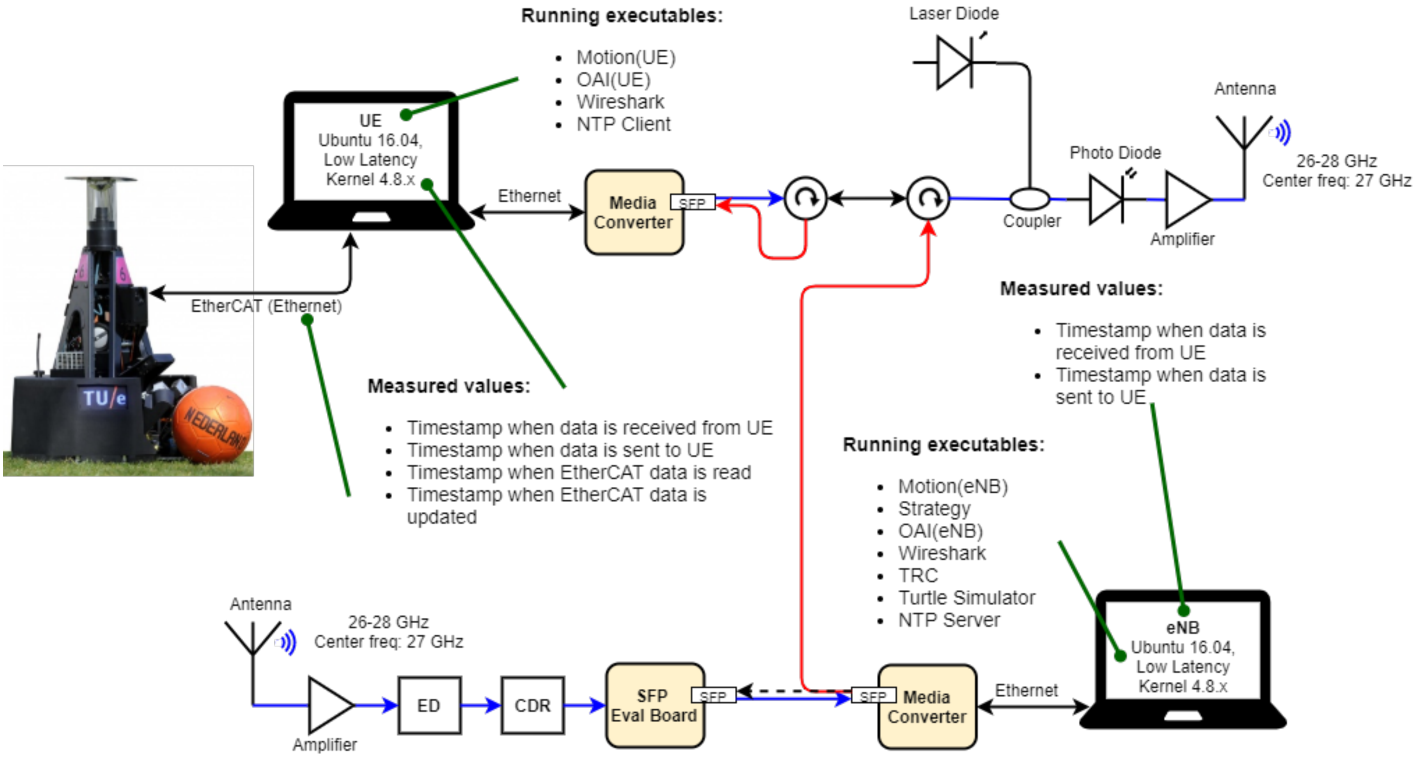


Figure 6.10: Hardware layout of the implemented setup. The setup includes 3 SFPs - one on each media converter, and one on the evaluation board on the electronics side. The blue lines show the wireless path from the UE to the eNB and the red lines show the wired path from the eNB to the UE. The solid lines describe optical fiber communication (unless specified) and the dotted lines describe the electronic signals. The black dashed line shows the replicated eNB output used as inputs to the evaluation board SFP. The black solid line shows a bidirectional optical fiber link between two optical circulators, which act as multiplexers/demultiplexers in this case. ED: Envelope Detector, CDR: Clock and Data Recovery board

The measured values as shown in the above image are analyzed and summarized in the next chapter.

7 Validation and Results

This chapter describes the specifics of the validation test setup and explains the obtained results. The following sections describe the Validation Phase of the work phase plan.

7.1 Validation Test Setup

As explained earlier, the setup in Figure 6.10 was implemented and needs to be validated. The following sections describe the process of setting up the hardware and software for validation.

7.1.1 Hardware Setup

The communication setup was implemented inside a Faraday cage on the 8th floor of the Flux building in the TU Eindhoven campus. The configured UE and eNB computers were attached to the media converters on either end of this network using ethernet cables. The Beckhoff stack on the Turtle was disconnected from the on-board computer and attached to the UE laptop. Since the connection is wired and the Turtle does not have much space to move, it was placed on a thick pad, so that the wheels do not touch the ground. This experimental test setup is shown in the following pictures.

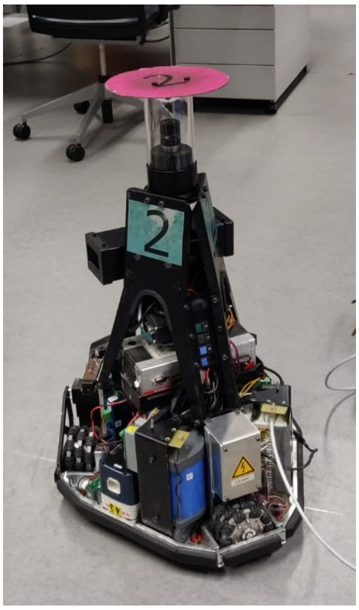


Figure 7.1: Turtle 2 is connected to an Ethernet cable, which is then connected to the UE computer

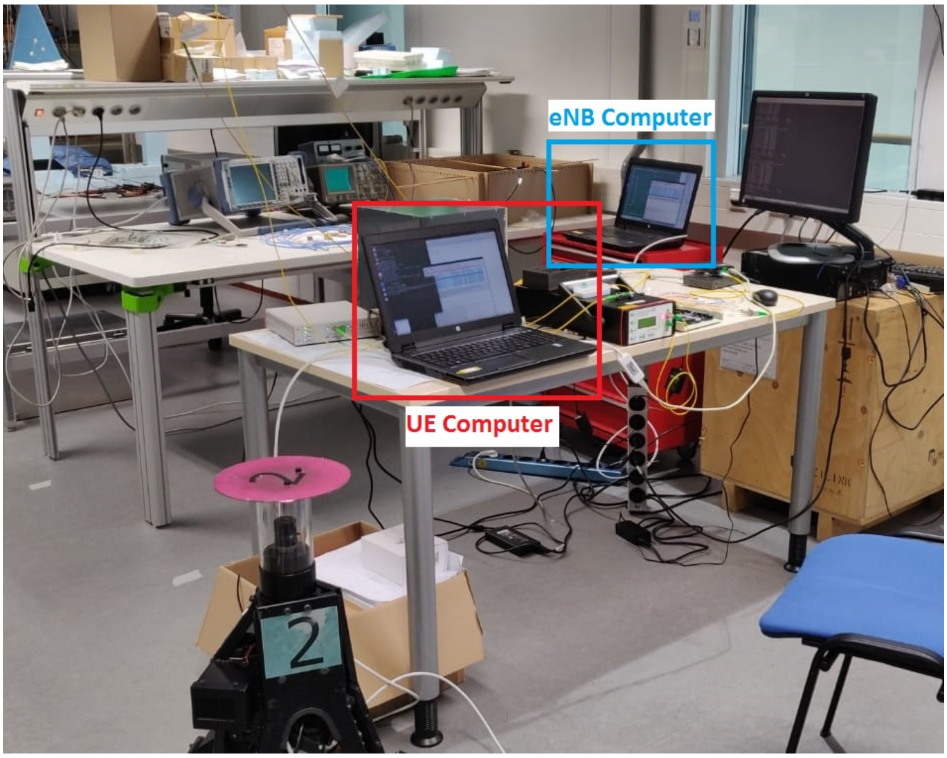


Figure 7.2: Setup of UE and eNB computers at the TU/e Flux Faraday cage

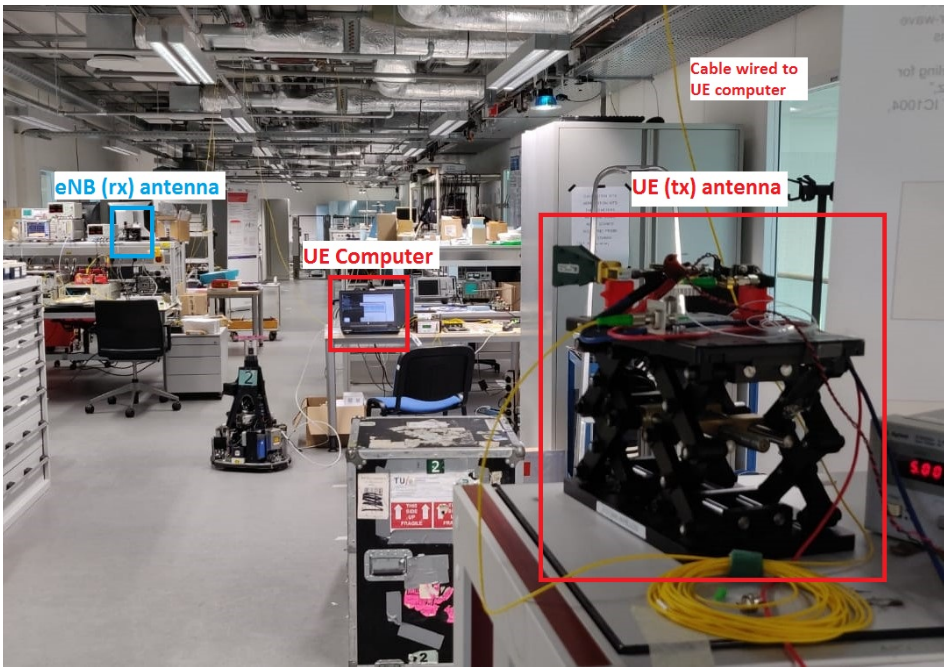


Figure 7.3: Setup of UE and eNB cantennas at the TU/e Flux Faraday cage. The UE antenna is connected to the UE computer via the media converter using an overhead cable.

7.1.2 Software Setup

Once the hardware is set up, the Turtle software on the UE and the eNB need to be built. The Simulink models on either side were configured with the following information:

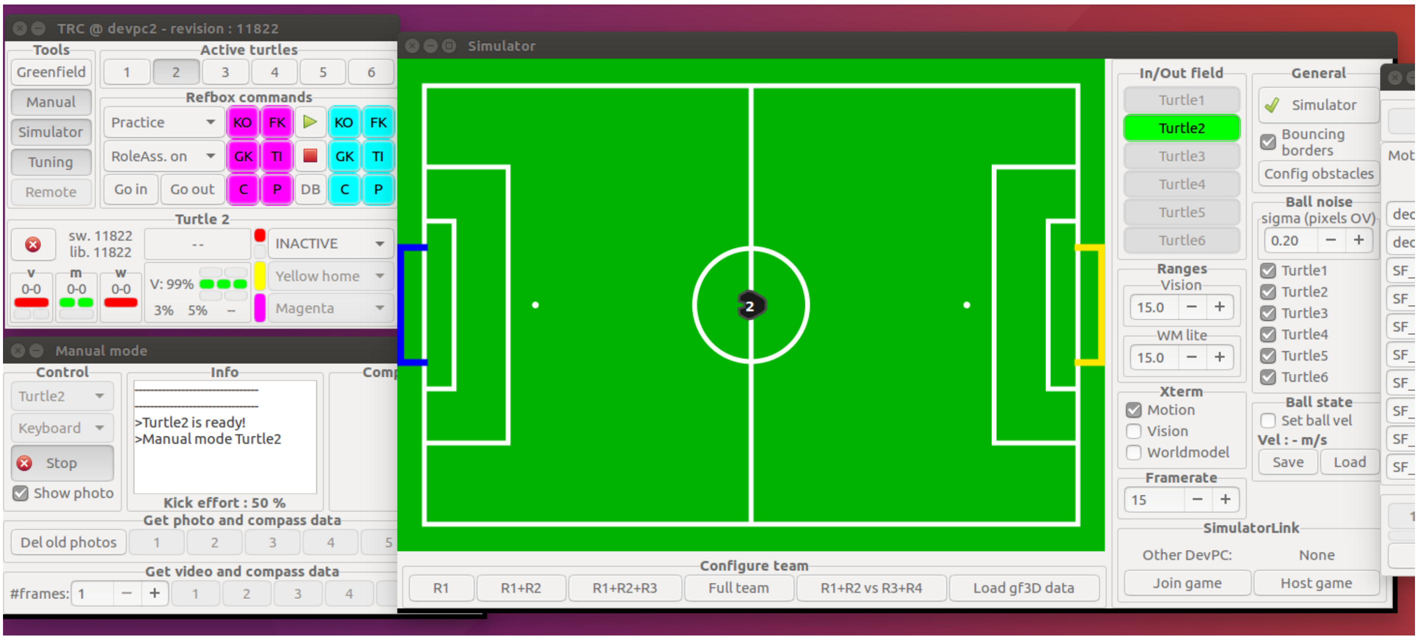
UE:

- Ethernet cable from the Beckhoff stack interfaced with port **eth0** of the UE computer
- Ethernet cable from the media converter interfaced with port **eth1** with IP address **192.168.1.5**
- UDP Send on local software configured with:
  - **Remote IP address: 192.168.1.6**
  - **Local port: 25002**
  - **Remote port: 25003**
- UDP Receive on local software configured with:
  - **Remote IP address: 192.168.1.6**
  - **Local port: 25001**
  - **Remote port: 25000**

eNB:

- Ethernet cable from the eNB media converter interfaced with port **eth0** with IP address **192.168.1.6**
- UDP Send on local software configured with:
  - **Remote IP address: 192.168.1.5**
  - **Local port: 25000**
  - **Remote port: 25001**
- UDP Receive on local software configured with:
  - **Remote IP address: 192.168.1.6**
  - **Local port: 25003**
  - **Remote port: 25002**

Once configured, the software on either side is built. The UE software is executed first, which establishes a connection with the Beckhoff stack and then start transmitting sensor data to the eNB IP address. The eNB software is run next, which returns actuator data once the sensor data is received. The actuator data does not change till the TRC GUI is run and the manual control mode is chosen. Once manual control is switched on, the Turtle can be controlled using the eNB laptop's keyboard. The sensor data is used to compute a position estimate in the eNB software, which is then visualized using the Turtle simulator.



>Turtle2 is ready!
>Manual mode Turtle2

Figure 7.4: This screenshot from the eNB computer shows the Turtle Remote Control (Top left), the Manual Control window (Bottom left) and the Simulator view on the right. On the TRC, Turtle 2 is chosen and the values *v, m, w* show which executables are running. The green LEDs under *m* shows that Motion and Strategy executables are communicating. The info tab on the Manual mode window shows that Turtle is ready, which means that a connection is established between the Turtle software on the UE and the eNB.

Once the UE and the eNB software is running, Wireshark is used to record sent/received data on each computer. Wireshark needs to be started individually on each computer, which means that each computer will start recording data at different times. This needs to be taken into account while calculating the round trip latencies.

For calculating the E2E latency, both computers need to be synchronized together. The timing on both computer clocks are synchronized before entering the Faraday cage. Synchronization is done using NTP with the eNB computer as Stratum 3 and the UE computer as Stratum 4. The eNB computer is synchronized to an online NTP server based in the Netherlands, and the UE is synchronized with the eNB [51].

Jitter is calculated by sending dummy data over the network and measuring it using iPerf. iPerf is also used to send dummy data with higher data rates in order to measure the capacity of the network.The CPU and memory usage of each computer is also logged to measure any possible improvements to the system. These measured values are analyzed and summarized in the following section.

7.2 Results

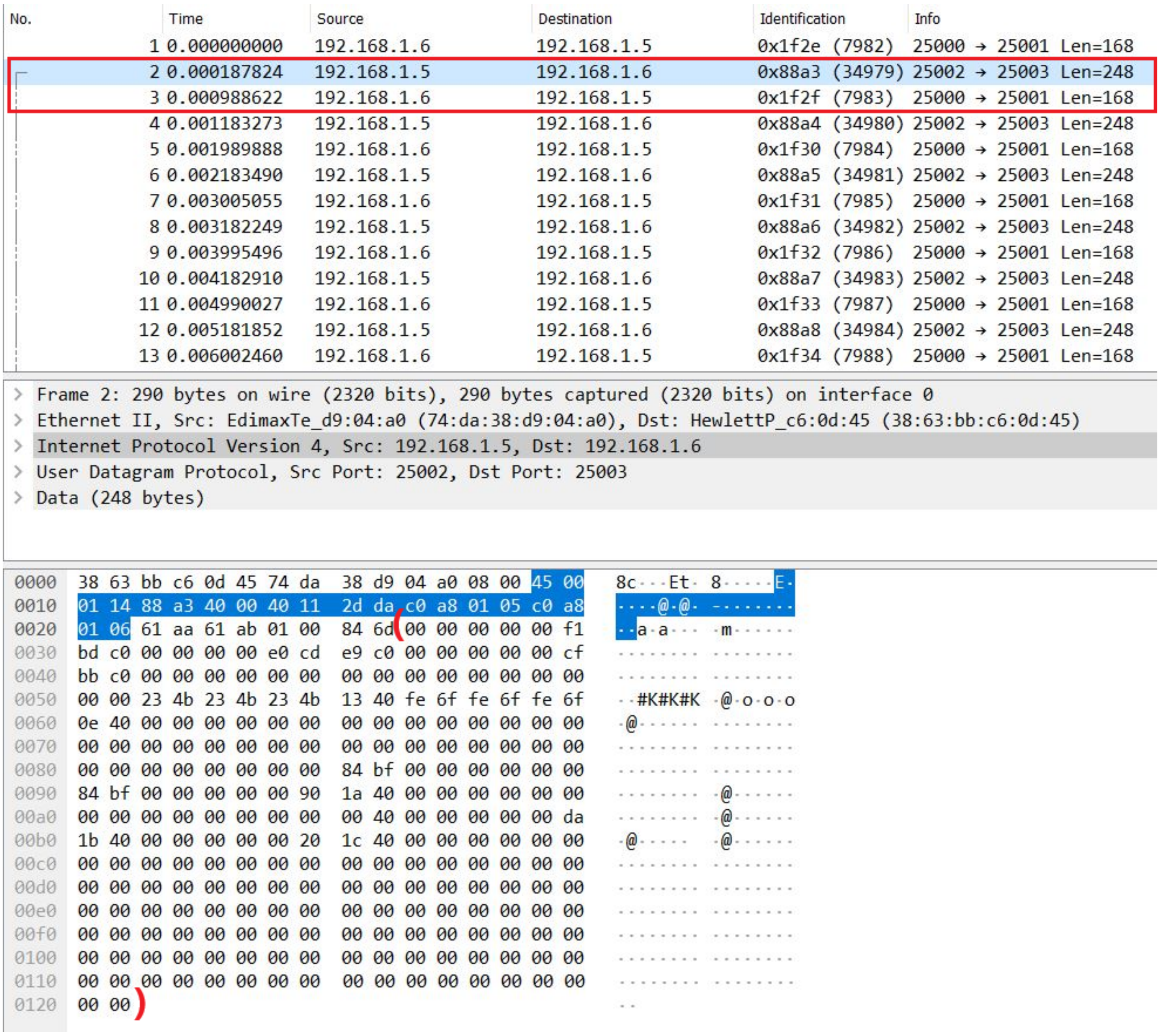
The following sections analyze the three measurements made in the validation phase - Wireshark measurements, iPerf measurements and CPU/Memory logs.

7.2.1 Wireshark Measurements

Wireshark captures on each computer provide the following information for each sent or received packet:

- **Wireshark sequence number** (starting at 0 on each computer)
- **Timestamp** (either absolute time from the system clock or relative time beginning at 0 when the Wireshark capture starts)
- **Data type** (in this case, only UDP)
- **Source and Destination IP addresses**
- **Source and Destination Ports**
- **Packet size** (both with and without headers)
- **Packet data** (including packet sequence number)

Wireshark measurements were initialized on each computer at different times. So, the data needs to be synchronized. Since Wireshark was initialized on the eNB first, the data sent and received before the initialization of Wireshark on UE needs to be ignored. This was done by checking the packet sequence number of the first captured data packet that was sent/received from the UE, and finding the same sequence number in the eNB records. All captured data before this packet are ignored. These measurements are explained in the following figures.

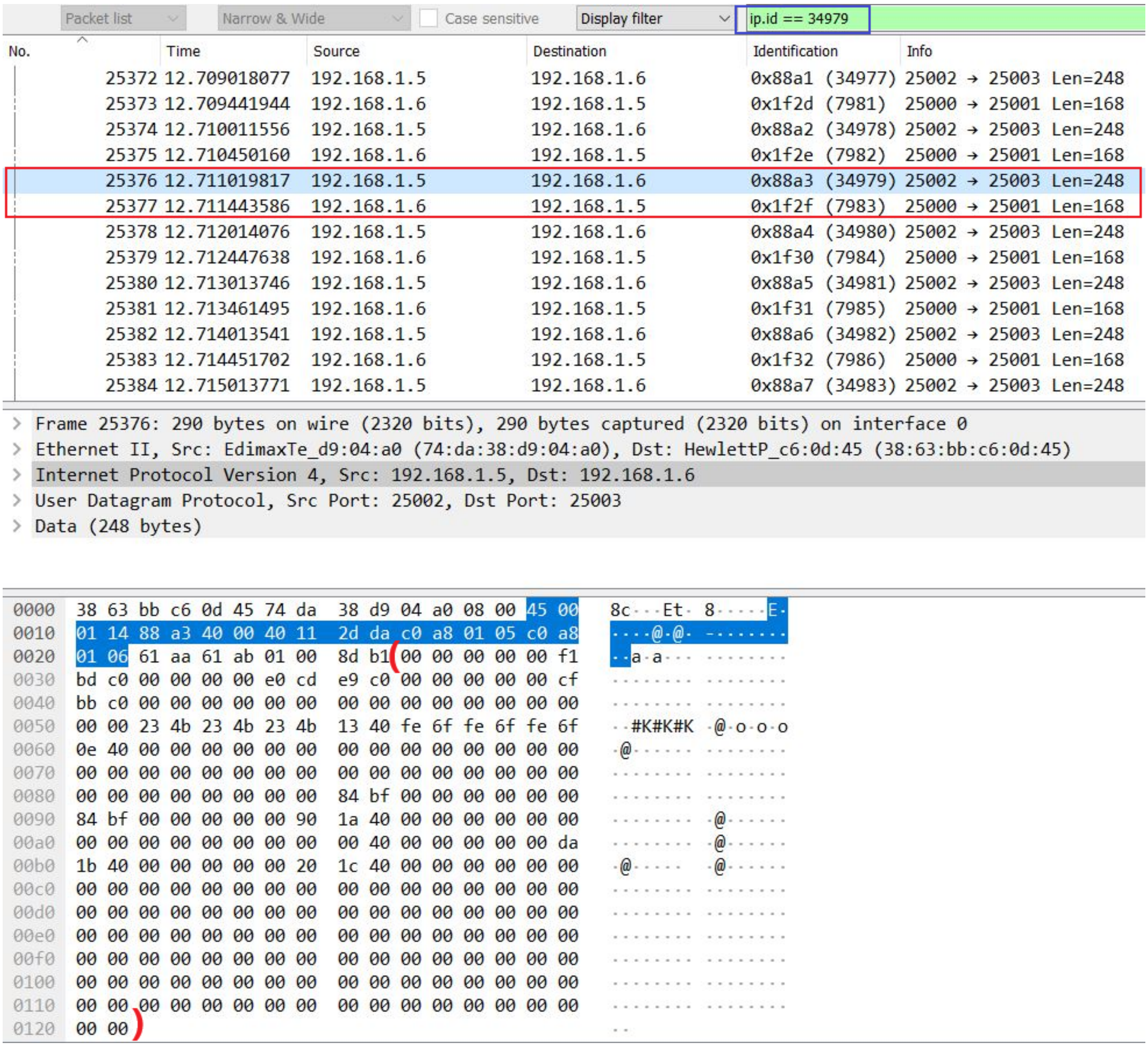


No.	Time	Source	Destination	Identification	Info
1	0.000000000	192.168.1.6	192.168.1.5	0x1f2e (7982)	25000 → 25001 Len=168
2	0.000187824	192.168.1.5	192.168.1.6	0x88a3 (34979)	25002 → 25003 Len=248
3	0.000988622	192.168.1.6	192.168.1.5	0x1f2f (7983)	25000 → 25001 Len=168
4	0.001183273	192.168.1.5	192.168.1.6	0x88a4 (34980)	25002 → 25003 Len=248
5	0.001989888	192.168.1.6	192.168.1.5	0x1f30 (7984)	25000 → 25001 Len=168
6	0.002183490	192.168.1.5	192.168.1.6	0x88a5 (34981)	25002 → 25003 Len=248
7	0.003005055	192.168.1.6	192.168.1.5	0x1f31 (7985)	25000 → 25001 Len=168
8	0.003182249	192.168.1.5	192.168.1.6	0x88a6 (34982)	25002 → 25003 Len=248
9	0.003995496	192.168.1.6	192.168.1.5	0x1f32 (7986)	25000 → 25001 Len=168
10	0.004182910	192.168.1.5	192.168.1.6	0x88a7 (34983)	25002 → 25003 Len=248
11	0.004990027	192.168.1.6	192.168.1.5	0x1f33 (7987)	25000 → 25001 Len=168
12	0.005181852	192.168.1.5	192.168.1.6	0x88a8 (34984)	25002 → 25003 Len=248
13	0.006002460	192.168.1.6	192.168.1.5	0x1f34 (7988)	25000 → 25001 Len=168

```
> Frame 2: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits) on interface 0
> Ethernet II, Src: EdimaxTe_d9:04:a0 (74:da:38:d9:04:a0), Dst: HewlettP_c6:0d:45 (38:63:bb:c6:0d:45)
> Internet Protocol Version 4, Src: 192.168.1.5, Dst: 192.168.1.6
> User Datagram Protocol, Src Port: 25002, Dst Port: 25003
> Data (248 bytes)
```

```
0000  38 63 bb c6 0d 45 74 da  38 d9 04 a0 08 00 45 00   8c···Et· 8·····E·
0010  01 14 88 a3 40 00 40 11  2d da c0 a8 01 05 c0 a8   ····@·@· -·······
0020  01 06 61 aa 61 ab 01 00  84 6d(00 00 00 00 00 f1   ··a·a··· ·m······
0030  bd c0 00 00 00 00 e0 cd  e9 c0 00 00 00 00 00 cf   ········ ········
0040  bb c0 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 23 4b 23 4b 23 4b  13 40 fe 6f fe 6f fe 6f   ··#K#K#K ·@·o·o·o
0060  0e 40 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ·@······ ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0080  00 00 00 00 00 00 00 00  84 bf 00 00 00 00 00 00   ········ ········
0090  84 bf 00 00 00 00 00 90  1a 40 00 00 00 00 00 00   ········ ·@······
00a0  00 00 00 00 00 00 00 00  00 40 00 00 00 00 00 da   ········ ·@······
00b0  1b 40 00 00 00 00 00 20  1c 40 00 00 00 00 00 00   ·@······ ·@······
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0120  00 00)                                             ··
```

Figure 7.5: **UE Wireshark Capture:** The red box indicates packet 2 and 3. Packet 2 is the first recorded packet sent from the UE (192.168.1.5) to the eNB (192.168.1.6), and Packet 3 is the corresponding data received from the eNB. The list also shows the time each packet was received (relative to the first recorded packet), the packet sequence number, the source and destination ports and finally, the length of the packet without headers. The numbers at the bottom show the entire data frame of Packet 2. The data highlighted in blue indicates the sequence ID of Packet 2 and the data within the red brackets describes the payload from the UE.



```
ip.id == 34979
```

```
No.        Time          Source         Destination    Identification  Info
     25372 12.709018077  192.168.1.5    192.168.1.6    0x88a1 (34977) 25002 → 25003 Len=248
     25373 12.709441944  192.168.1.6    192.168.1.5    0x1f2d (7981)  25000 → 25001 Len=168
     25374 12.710011556  192.168.1.5    192.168.1.6    0x88a2 (34978) 25002 → 25003 Len=248
     25375 12.710450160  192.168.1.6    192.168.1.5    0x1f2e (7982)  25000 → 25001 Len=168
     25376 12.711019817  192.168.1.5    192.168.1.6    0x88a3 (34979) 25002 → 25003 Len=248
     25377 12.711443586  192.168.1.6    192.168.1.5    0x1f2f (7983)  25000 → 25001 Len=168
     25378 12.712014076  192.168.1.5    192.168.1.6    0x88a4 (34980) 25002 → 25003 Len=248
     25379 12.712447638  192.168.1.6    192.168.1.5    0x1f30 (7984)  25000 → 25001 Len=168
     25380 12.713013746  192.168.1.5    192.168.1.6    0x88a5 (34981) 25002 → 25003 Len=248
     25381 12.713461495  192.168.1.6    192.168.1.5    0x1f31 (7985)  25000 → 25001 Len=168
     25382 12.714013541  192.168.1.5    192.168.1.6    0x88a6 (34982) 25002 → 25003 Len=248
     25383 12.714451702  192.168.1.6    192.168.1.5    0x1f32 (7986)  25000 → 25001 Len=168
     25384 12.715013771  192.168.1.5    192.168.1.6    0x88a7 (34983) 25002 → 25003 Len=248
```

```
> Frame 25376: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits) on interface 0
> Ethernet II, Src: EdimaxTe_d9:04:a0 (74:da:38:d9:04:a0), Dst: HewlettP_c6:0d:45 (38:63:bb:c6:0d:45)
> Internet Protocol Version 4, Src: 192.168.1.5, Dst: 192.168.1.6
> User Datagram Protocol, Src Port: 25002, Dst Port: 25003
> Data (248 bytes)
```

```
0000  38 63 bb c6 0d 45 74 da  38 d9 04 a0 08 00 45 00   8c···Et· 8·····E·
0010  01 14 88 a3 40 00 40 11  2d da c0 a8 01 05 c0 a8   ····@·@· -·······
0020  01 06 61 aa 61 ab 01 00  8d b1(00 00 00 00 00 f1   ··a·a··· ········
0030  bd c0 00 00 00 00 e0 cd  e9 c0 00 00 00 00 00 cf   ········ ········
0040  bb c0 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 23 4b 23 4b 23 4b  13 40 fe 6f fe 6f fe 6f   ··#K#K#K ·@·o·o·o
0060  0e 40 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ·@······ ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0080  00 00 00 00 00 00 00 00  84 bf 00 00 00 00 00 00   ········ ········
0090  84 bf 00 00 00 00 00 90  1a 40 00 00 00 00 00 00   ········ ·@······
00a0  00 00 00 00 00 00 00 00  00 40 00 00 00 00 00 da   ········ ·@······
00b0  1b 40 00 00 00 00 00 20  1c 40 00 00 00 00 00 00   ·@······ ·@······
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0120  00 00)                                             ··
```

Figure 7.6: **eNB Wireshark Capture:** Since, the eNB was initialized first, the UE and the eNB captures need to be synchronized. The first recorded packet sent by the UE is Packet 2 with sequence number 34979. Using this sequence number, the corresponding packet was found in the eNB capture (highlighted in the blue box). The red box here indicates the packets corresponding to the ones highlighted in Figure 7.5. As seen, received packet number 25376 corresponds to the packet sent by the UE. It also shows the time this packet was received, relative to the first recorded packet on the eNB. This also determines the time difference between initializing Wireshark on the UE and the eNB (12.7110 - 0.00018). All data above packet 25376 is ignored since it was not recorded by the UE. Once again, the data highlighted in blue indicates the packet sequence number and the data within the red brackets indicate the received sensor data.

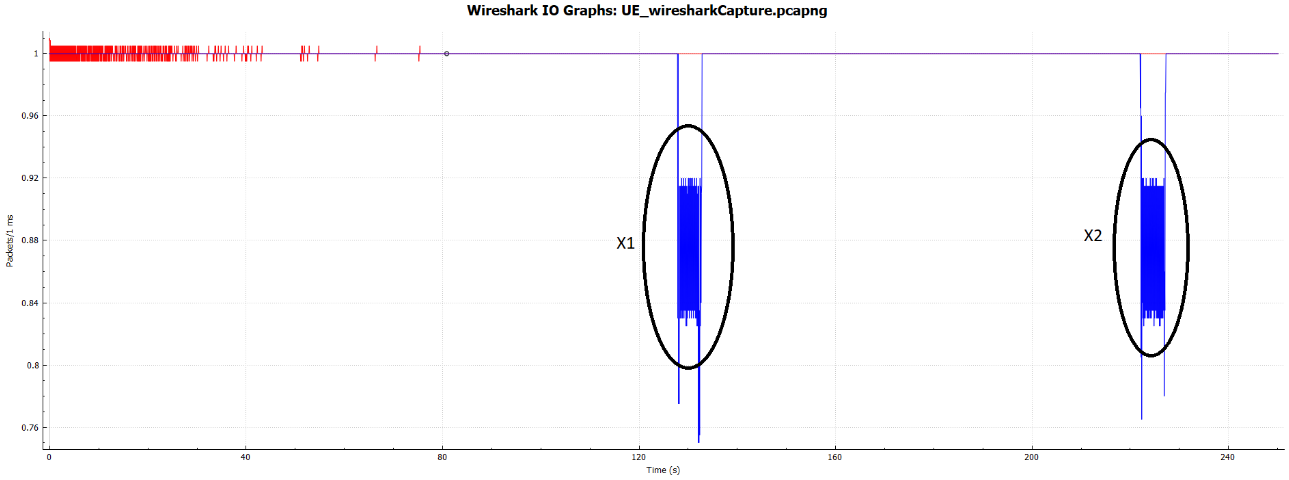


Figure 7.7: **UE Capture Statistics:** The above graph plots the data rates of the received and sent data as time progresses. The vertical axis represents the number of packets per millisecond and the horizontal axis represents time in seconds. **The blue plot represents data sent from the UE and the red plot represents data received at the UE**. The red plot remains almost constant at 1 packet/ms because of the wired connection between eNB to the UE. The minor fluctuations on the red plot are caused because of jitter. On the blue plot, there are two substantial drops (x_1 and x_2), which are due to disturbances in the wireless network. The Turtle was placed in the line-of-sight between the to antennas as shown in Figure 7.3 and disturbances were cause when the Turtle had to be physically moved to measure IMU readings. Other than these two anomalies, the blue plot remains at a value of 1 packet/ms with very minor fluctuations.

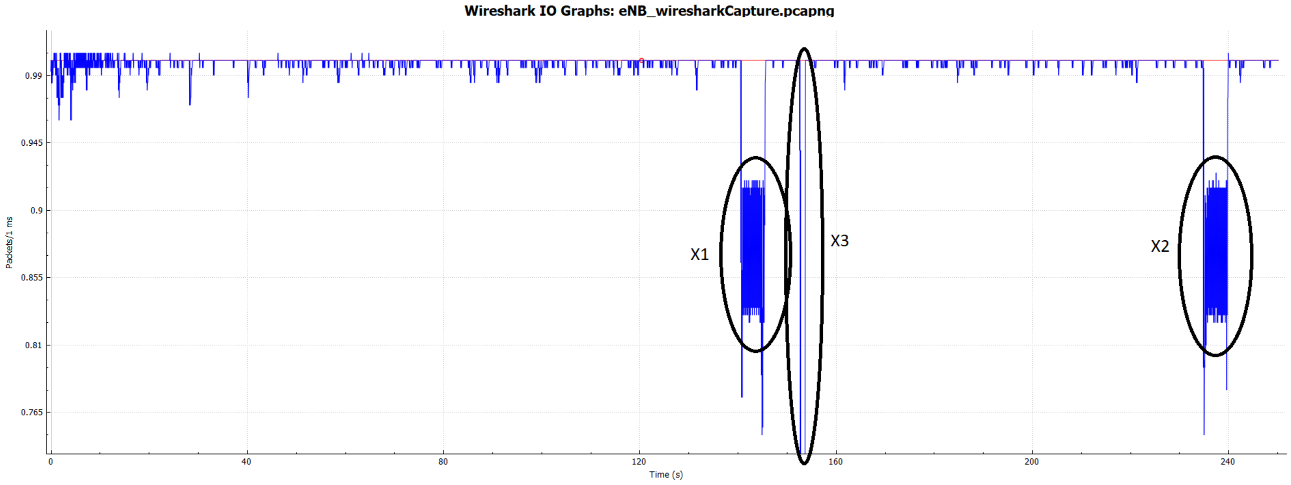


Figure 7.8: **eNB Capture Statistics:** The above graph shows the same packets as in Figure 7.7 but is now recorded on the eNB. **The blue plot represents data received at the eNB the red plot represents data sent to the UE**. As expected due to the wired connection from the eNB to the UE, the red plot remains at 1 packet/ms. The blue plot shows the same fluctuations as in Figure 7.7, however this has been offset by around 12 seconds as computed in Figure 7.6. The sharp drop in the blue plot between the two fluctuations (x_3)can be attributed to latency.

The recorded data can be analyzed by comparing the timing between sending and receiving data at the UE, and the timing between receiving and sending data at the eNB. These measurements can be used to draw a timing diagram like in Figure 5.11. The following table describes timing values of three full clock cycles of data.

Table 7.1: Timing data of packets recorded at UE and eNB

Recorded at UE					**Recorded at eNB**				
No.	**Time (s)**	**Src.**	**Dst.**	**ID.**	**No.**	**Time (s)**	**Src.**	**Dst.**	**ID.**
2	0.000187824	UE	eNB	34979	25376	12.71101982	UE	eNB	34979
3	0.000988622	eNB	UE	7983	25377	12.71144359	UE	eNB	7983
4	0.001183273	UE	eNB	34980	25378	12.71201408	UE	eNB	34980
5	0.001989888	eNB	UE	7984	25379	12.71244764	UE	eNB	7984
6	0.00218349	UE	eNB	34981	25380	12.71301375	UE	eNB	34981
7	0.003005055	eNB	UE	7985	25381	12.7134615	UE	eNB	7985

This table can be used to draw the following timing diagram.

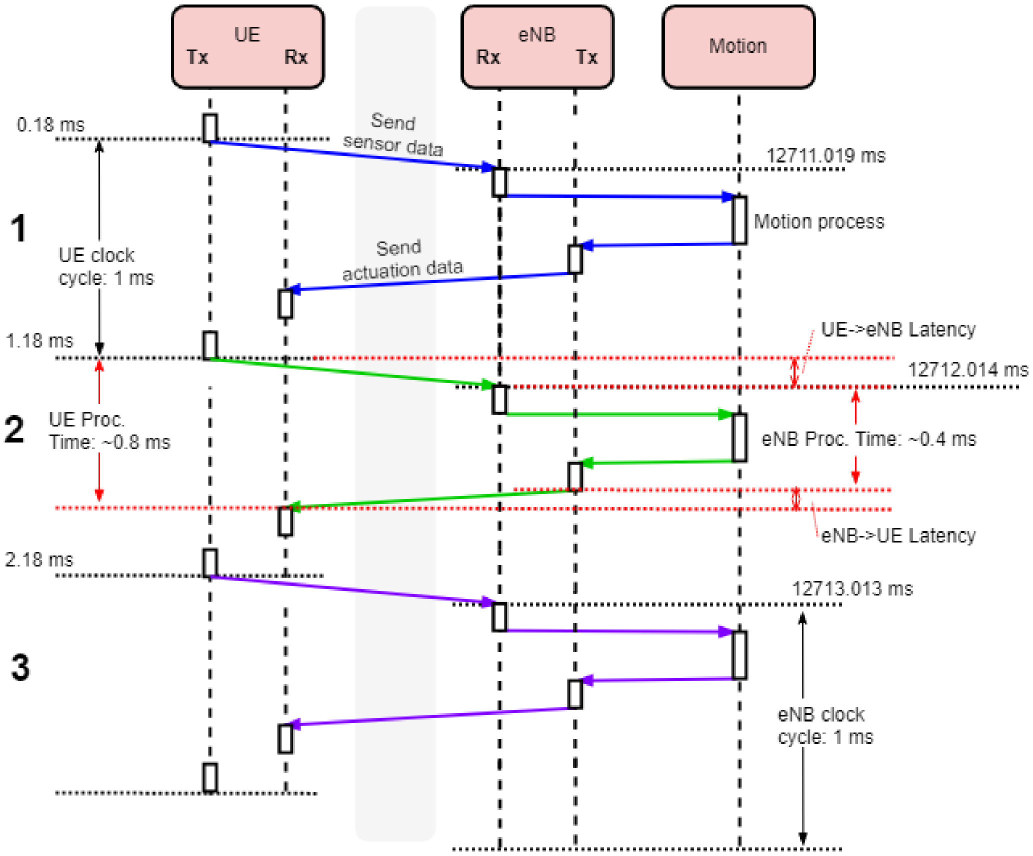


Figure 7.9: The resulting timing diagram for three clock cycles shows that sensor data is sent and corresponding actuation commands are received in the same clock cycle at the UE. The diagram also shows how the UE processing time is the sum of the eNB processing time and the two E2E latencies (the sum of the 2 E2E latencies is the round trip latency).

With the Wireshark records and the above the timing diagram, the following parameters can be measured:

Latency

The time spent on the network between sending a packet from the UE and receiving the corresponding packet from the eNB can be defined as the round trip latency. The round trip (RT) latency is the sum of end-to-end latencies from the UE to the eNB and from the eNB to the UE. This can also be inferred from the proposed and measured timing diagrams in Figures 5.11 and 7.9. As an example, the round trip latencies for the first three transmitted packets are calculated using Table 7.1 and shown in the table below.

Table 7.2: **Latency:** Calculated as an absolute difference between the processing times at the UE and the eNB

Recorded at UE			Recorded at eNB			
No.	**Time (s)**	**Proc. Time (ms)**	**No.**	**Time (s)**	**Proc. Time (ms)**	**RT Latency (ms)**
2	0.000187824		25376	12.71101982		
3	0.000988622	0.800798	25377	12.71144359	0.42377	**0.377028**
4	0.001183273		25378	12.71201408		
5	0.001989888	0.806615	25379	12.71244764	0.43356	**0.373055**
6	0.00218349		25380	12.71301375		
7	0.003005055	0.821565	25381	12.7134615	0.44775	**0.373815**

Similarly, the round trip latency is measured for 10,000 recorded packets and an average is calculated. **The average round trip latency is 0.000278439 seconds (0.278 ms)**.

For the above calculation, the system clocks on the two computers do not need to be synchronized. However, E2E latency is calculated by computing the absolute difference between the time a packet is transmitted from the UE and received at the eNB, or when a packet is transmitted from the eNB and received at the UE. This difference includes the E2E latency as well as the clock difference between the two absolute times. So, in order for the latency measurement to be accurate, the magnitude of the clock difference should be lower than the magnitude of the expected latency, which is not possible in this case, as PTP could not be implemented.

However, computers were synchronized using NTP. The eNB computer was synchronized to an online server as **Stratum 3 with a clock offset of 109 ms**. The UE was synchronized with the eNB as **Stratum 4 with a clock offset of 113 ms**. Both clocks also had a variance of 3-4 ms, which made the clock offset unusable to calculate the E2E latency.

It would not be right to state that the UE to eNB latency is the same as the eNB to UE latency, since one is a wired network and the other is wireless. However, since the round trip latency is so low and within the requirements set by blueSPACE and Tech United, it is safe to assume that the average E2E latency is half of the average round trip latency.

Packet Loss

Packet loss can be analyzed by looking at the graphs in Figures 7.7 and 7.8. The red plot in both Figures describe the transmission from the eNB to the UE. Since this part of the network is wired, no packet losses are observed here. The red plot in Figure 7.8 is a straight line, while the plot in Figure 7.7 shows fluctuations, which is attributed to Jitter. However, for every fluctuation that is less than 1 packet/ms, there is a following fluctuation that is an equal amount more than 1 packet/ms. In this section, no packet loss is observed.

In the blue plots in both Figures, major packet losses were observed in x_1, x_2 and x_3. This is due to disturbances in the network. Ignoring these, the blue plot in Figure 7.7 shows a solid blue line. However, the plot in Figure 7.8 show fluctuations that are not as uniform as the red plot in Figure 7.7. These are due to packet losses in the network. On average, while there is 1 packet sent per millisecond, only around 0.98 packets are received. This amounts to an **average packet loss of 2%**

Table 7.3: **Packet Loss:** Measured from Figures 7.7 and 7.8

Data Stream	**Packet Loss**	**Measured Time Period**
x_1	17%	∼6 seconds
x_2	17%	∼6 seconds
x_3	100%	∼1 second
Average UE to eNB	**2%**	**∼300 seconds**

Jitter

Jitter is measured by calculating the absolute difference between the round trip latencies calculated from the sample of records. The average of these differences is the Jitter. A sample jitter calculation is shown in the table below.

Table 7.4: **Jitter:** Calculated as an average of the difference between the consecutive round trip latencies.

Recorded at UE		**Recorded at eNB**			
No.	**Time (s)**	**No.**	**Time (s)**	**RT Latency (ms)**	**Jitter (ms)**
2	0.000187824	25376	12.71101982		
3	0.000988622	25377	12.71144359	0.377028	-NA-
4	0.001183273	25378	12.71201408		
5	0.001989888	25379	12.71244764	0.373055	**0.003973**
6	0.00218349	25380	12.71301375		
7	0.003005055	25381	12.7134615	0.373815	**0.000761**

Similarly, the latencies calculated for the 10,000 data points are used to calculate the average jitter. **The average round trip Jitter is 0.0000132579 seconds (13.258 us)**.

This means that the round latency could vary by an average of 13.258 us, which is small enough and satisfies the requirements.

Turtle Timing Requirements

The Turtle software is divided into the UE and the eNB executables. Both these executables need to operate at 1000 Hz, i.e. with a clock cycle of 1 ms. Since only specific measurements are possible, The clock cycle of the UE is defined as the time between consecutive sensor data updates and the clock cycle of the eNB is defined as the time between consecutive arrival of sensor data. These values are calculated and averaged for the entire sample. The measurements are not entirely accurate due to the jitter in the network, hence the meeasured clock cycles are not exactly 1 ms, but very close.

- **The average clock cycle of the UE executable is 0.0009998 seconds ($\sim$ 1 ms)**
- **The average clock cycle of the eNB executable is 0.001000801 seconds ($\sim$ 1 ms)**

The clock cycle of 1 ms is a deadline for the UE and the eNB processes. However, processing time is expected to be much lower than the deadline. The average processing time for the UE and the eNB executables are calculated from the sample data.

- **The average processing time of the UE executable is 0.000799813 (0.8 ms)**
- **The average processing time of the eNB executable is 0.000507483 (0.5 ms)**

It needs to be noted that the Turtle is being controlled manually using the TRC and many of the control algorithms in the eNB executable are not being computed. The resulting low processing time allows the Turtle (UE) to send sensor data and receive corresponding actuation commands within the same clock cycle. However, if the eNB executable is to run at full capacity, the processing time could be higher and would cause the Turtle to receive actuation commands in the next clock cycle. While the latency would remain the same, the actuation would be delayed by one clock cycle. However, this is within the tolerance of 5-7 ms provided by Tech United.

7.2.2 iPerf Measurements

iPerf was used to measure the network by initializing one computer as the server and the networked computer as the client. Once the server is running, the client is executed in the 'bidirectional' mode with a specified data rate. The client transmits dummy data packets with sizes corresponding with to the tested data rate value and receives packets of the same size from the server. In this process, the actual data rates and the jitter are measured and shown in the table below.

Table 7.5: The iPerf results list the measured data rates and jitter for specified test values

Test No.	Tested Data Rate	Measured Max. Data Rate	Measured Avg. Jitter
1	2 Mbps	2.01 Mbps	0.006 ms
2	3 Mbps	3.02 Mbps	0.007 ms
3	300 Mbps	302 Mbps	0.038 ms
4	500 Mbps	511 Mbps	0.030 ms
5	1000 Mbps	**816 Mbps**	0.054 ms

In the above table, the first two tests were performed to simulate the eNB to UE data and the UE to eNB data respectively. The measured end-to-end jitter is 6-7 microseconds on average, which means that the round trip jitter is expected to be around 12-14 microseconds. This verifies the round-trip jitter calculation of 13.258 us as shown in the earlier sections.

Next, higher data rates are tested in order to determine the bandwidth, or the maximum data rate allowed by the network. Three tests with 300 Mbps, 500 Mbps and 1000 Mbps are conducted. As seen, the measured data rates for the first four tests are in the same range as the tested data rates. However, when the tested data rate is increased to 1000 Mbps or higher, the measured data rate is limited to around 816 Mbps. When 1000 Mbps is sent, the packet loss is hence, around 18.4%. **The bandwidth of the implemented network is determined to be 816 Mbps**. It must also be noted that when the network was tested by members of blueSPACE before the Turtle experiment, bandwidths of up to 1 Gbps were obtained.

7.2.3 CPU/Memory Logs

The CPU usage of both the UE and the eNB computer were recorded for a short period of time to see the difference between running the software on-board the robot and offloading it to the remote server.

Table 7.6: CPU/Memory usage for executables running on UE and eNB

Computer	**Executable**	**% CPU usage**	**% Mem usage**
UE	Motion_UE	38.2	0.2
eNB	Strategy	30.7	0.2
eNB	Motion_eNB	32.8	0.3

The motion block on the eNB shows a lower CPU usage value since the robot is controlled manually and most high level control processes are not executed. However, since the Strategy block is offloaded, the offloaded system is already reducing the CPU usage on the UE.

The results described in the sections above are summarized in the following chapter.

8 Conclusions and Recommendations

This chapter describes the Follow-Up Phase of the work phase plan. Here, the results are compared with the requirements defined at the beginning of the projects and recommendations are made accordingly. This chapter also provides a list of steps to be taken for continuing this project.

8.1 Summary

This project involved the design of an experiment to demonstrate and validate a 5G mm-network in an Industry 4.0 based application. In order to achieve this, a V-model approach was used which was divided into six phases: Definition, Design, Preparation, Realization, Validation and Follow-Up. In the Definition phase, a stakeholder analysis was carried out to understand the key stakeholders and their concerns. Next, a literature survey of blueSPACE and the Tech United Turtles was carried out, which was used to propose possible use cases. Once a final use case was chosen, the Design phase was initiated, which resulted in an architecture description using the CAFCR methodology. In the Preparation and Realization phases, the architecture was designed in detail, and then implemented. This implemented setup was then validated.

During this process, multiple hardware acquisition delays were encountered. This led to the scope of the project being reduced to use off-the-shelf hardware as a proof of concept. However, the chosen software (OAI) is not very well documented or well tested with the chosen hardware (LimeSDR), which caused delays and issues in the Preparation and Realization phases. This led to a backup network being set up using media converters. This setup was validated using the defined validation methodology.

While planned, the implementation of a mm-wave network or a 4G LTE network was not possible. However, the implemented project is a first step towards testing the blueSPACE hardware on the Turtle platform, once it arrives. The setup is designed such that the implemented network can be replaced by the blueSPACE hardware and 5G mm-wave communication, and the defined KPIs can be validated using the methodology provided in this project.

8.2 Conclusions

Despite the changes in the implemented network, the rest of the setup was implemented according to the designed architecture. While not 5G, the setup helps validate the Turtle's requirements and prove the network validation methodology. The measured results can be compared with the requirements defined in the table below.

Table 8.1: Comparison between required and measured values

Measurement	**Requirement**	**Result**
Sensor data Tx frequency (at UE)	1000 Hz	1000 Hz
Sensor data Rx frequency (at eNB)	1000 Hz	1000 Hz
Actuation data Rx frequency (at UE)	1000 Hz	1000 Hz
Motion block sample time (at eNB)	1 ms	1 ms
Round trip latency	<2 ms	0.287 ms
End to End latency	<1 ms	0.143 ms
Jitter	<0.5 ms	6.6 us
Bandwidth	1 Gbps	816 Mbps

As a result of this project, the following conclusions can be made.

- **In order to implement OAI, the Turtle computer needs to be upgraded, however it does not need to be as powerful as the the existing on-board computer.** OAI needs an Intel i3 processor at least, which the existing on-board computer does not have. With the proposed design, the communication hardware is added, and the NVidia Jetson is removed and the Kinect is connected directly to the Turtle computer. This requires additional interface ports which the on-board computer does not have.

- **Under the implemented network conditions and with the upgraded computer, the Turtle's Motion executable can be successfully successfully decoupled with the data acquisition.** The software was successfully decoupled and it was proven that the data acquisition could be run independently on the UE and the rest of the Motion executable could be run independently on the eNB.

- **This decoupling meets the timing requirements of the Turtle platform.** The updated Turtle Software allows the UE and the eNB sides to function at 1000 Hz and allows the data to be sent and received with the required data rates and within the expected tolerance as shown in Table 8.1.

- **This decoupling allows a reduction in the computing requirements of the Turtle's computer.** The Turtle on-board computer is replaced with an HP Z-book. An identical laptop is used as the eNB computer. A comparison is made between the data acqusition running on the UE and the Motion and Strategy executables running on the eNB.

- **This decoupling partially proves the network validation methodology.** While the validation methodology is used to calculate the round trip latency, end-to-end latency could not be calculated due to a varying clock offset between the two computers. However, the use of Precision Time Protocol is recommended.

- **The implemented application can be extended to the full team of robots.** Currently, the remote computer retains the Wi-Fi interface in order to connect and communicate with other Turtles that are not equipped with the implemented communication setup. However, once all Turtles are equipped with the communication hardware and the computational processes of all robots are offloaded to a single remote server, internal interfaces can be used for inter-robot communication. This improves the existing inter-robot communication framework.

- **Once the blueSPACE hardware arrives and the mm-waves network is configured, it can be plugged in directly with the designed architecture and the proven network validation methodology can be used.** All the designs - the proposed design using blueSPACE hardware, the implemented design using LimeSDR and the validated design using the media converters - use IP based communication. blueSPACE's 5G mm-wave network which would provide IP addresses on both sides could be connected directly to the provided Turtle software on the UE and the eNB computers.

8.3 Recommendations

This section recommends next steps for anybody continuing this project and provides recommendations for blueSPACE and Tech United.

8.3.1 Next Steps

The following next steps need to be taken in order to completely realize the proposed design in this project.

- Acquisition and configuration of the blueSPACE hardware with OAI.
- Setup of the 5G mm-wave network between the UE and the eNB computers attached to the blueSPACE hardware
- Acquisition of a PTP Grandmaster clock and the synchronization of the UE and the eNB computer clocks using the proposed method
- Replication of the implemented experiment by running the Turtle software on the UE and the eNB and validation of the mm-wave using the provided methodology
- Decoupling of the Vision and Kinect executables using the provided methodology
- Testing in the Faraday cage using the provided methodology
- Setup of the 5G mm-wave network at the Tech United soccer field
- Replication of the setup on all five Turtles and execution of all their decoupled software on a powerful remote server

8.3.2 Recommendations

blueSPACE

- For robotics, it is preferable to use small form-factor computers, which are housed in shock proof cases. Most off-the-shelf alternatives do not have a full sized PCIe slots but can have one or more Ethernet ports. So, an Ethernet interface with the 5G hardware is recommended.
- For the Turtle to function autonomously, it needs to be on a soccer field and needs to be able to detect the field lines. It is highly preferable for a 5G test bench be set up at the Tech United field in the Gemini Building at TU/e. The other alternative would be a big indoor space where a demo field could be set up.

Tech United

- The OpenAirInterface software is necessary for the 5G hardware to function. This needs at least an Intel i3 processor to function, which the Turtle does not have. Additionally, the additional hardware requires additional ports which are not available on the current on-board computer. Hence, the Turtle computer needs to be replaced. The recommended configuration for an on-board computer is as follows:
  - Intel i3 or higher processor with Ubuntu 16.04+ and a low-latency kernel v.4.7x+
  - 1 Gigabit Ethernet port for the omnivision camera
  - 1 Gigabit Ethernet port for the Beckhoff stack
  - 1 or more Ethernet ports for the blueSPACE hardware. However, USB to Gigabit Ethernet adapters could be used.
  - 1 USB 3.0 port for the Kinect v2
  - 1 USB 3.0 port if the LimeSDR is used

  Here, an off-the-shelf computer such as an Intel Nuc can be used with USB and Ethernet hubs. The remote computer needs an Intel i3 or higher processor with Ubuntu 16.04+ with a low-latency kernel v.4.7.x+ and whatever hardware interface is required by the blueSPACE base station hardware.
- In this project, it would have been possible to run the robot autonomously, if there was a way to simulate only the Vision and World Model executables and run it alongside the existing Motion and Strategy executables. Being able to simulate individual executables is desirable as it could also be used in the development and testing of other Turtle functionalities.

8.3.3 Future Scope

This project is the first step towards the testing of 5G in a robotic application at TU Eindhoven. As explained in this report, the proposed design adds value to blueSPACE by providing a means to test, validate and demonstrate their hardware and architecture in a robotic use case. Since both blueSPACE and Tech United are based in the TU Eindhoven campus, this project promotes future collaboration between telecommunication and robotics researchers from the two groups in innovative applications. With the help of the High Tech Systems Center, this collaboration would attract additional partnership and funding from the industry.

For Tech United, this project adds value by improving the performance of the Turtle team. With 5G mm-waves, Tech United will be able to research and implement processor-intensive algorithms that would not have been possible on the existing on-board computer. In addition, the proposed design will improve inter-robot communication by using internal interfaces on a single server instead of a Wi-Fi network.

When implemented with the entire team of Turtles, this project also adds value to the RoboCup Mid-Sized League. Using the Tech United implementation as proof-of-concept, RoboCup could be convinced to change the rulebook and allow remote computation using 5G mm-waves. This would not only bring all participating teams on a level playing field but as a result would improve the quality of the robot soccer matches. This could potentially bring RoboCup closer to their goal of competing against human players in the future.

The proposed design could also be used in multiple other scenarios, specifically in Industry 4.0. Factories could install multiple 5G base stations which fixed machines or robots such as AGVs could connect to. With such high data rates and low latencies, the control software could be run either on dedicated servers on the factory premises or on the cloud. This would allow a lot of data to be collected and used to improve efficiency. By sharing data collected from different sources, the robots and machines will be able to collaborate better, just like in the Tech United use case. This data could also be used for predictive maintenance of the connected devics. Additionally, the reduced computing requirements on-board the moving robots would result in lower costs and also an improved battery life.

In conclusion, this project presents a design and validation methodology that helps realize the goals of blueSPACE and Tech United. This is just the first step towards a wide range of possible applications in academia and industry in the future.

Bibliography

[1] 5G - Wikipedia. *https://en.wikipedia.org/wiki/5G*.

[2] 5G Infrastructure Public Private Partnership. *https://5g-ppp.eu/*.

[3] blueSPACE - 5GPPP. *https://www.bluespace-5gppp.eu/*.

[4] Space Division Multiplexing 5G Fronthaul with Analog and Digital Radio-over-Fiber and Optical Beamforming - the blueSPACE Concept. *Rommel, Simon; Raddo, Thiago R.; Tafur Monroy, Idelfonso, DOI: 10.1109/CAMAD.2018.8514944*.

[5] What is Industry 4.0? *https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/*.

[6] blueSPACE Deliverable D6.1: Report on targeted use cases, demonstration scenarios and description of testbed implementation plan. *blueSPACE Consortium, October 2019, https://bluespace-5gppp.eu/publications*.

[7] Tech United Eindhoven Wiki. *http://www.techunited.nl/wiki/index.php?title=Main_Page*.

[8] Tech United TURTLEs. *http://www.techunited.nl/nl/voetbalrobots*.

[9] Robocup Mid Sized League. *https://www.robocup.org/leagues/6*.

[10] Robocup 2019 - Sydney. *https://2019.robocup.org/*.

[11] Gaudi Systems Architecting - Homepage. *https://www.gaudisite.nl*.

[12] V-Model - Wikipedia. *https://en.wikipedia.org/wiki/V-Model_(software_development)*.

[13] CAFCR: A Multi-view Method for Embedded Systems Architecting; Balancing Genericity and Specificity. *https://www.gaudisite.nl/ThesisBook.pdf*.

[14] The Evolution of Systems Engineering - The MITRE Corporation. *https://www.mitre.org/publications/systems-engineering-guide/systems-engineering-guide/the-evolution-of-systems*.

[15] Image Source: 5G Spectrum. *https://techblog.comsoc.org/2019/05/22/tech-mahindra-india-needs-to-begin-5g-spectrum-auction-now/*.

[16] IMT Vision - Framework and overall objectives of the future development of IMT for 2020 and beyond. *ITU-R: Radiocommunication Sector of International Telecommunication Union*.

[17] The Fronthaul Infrastructure of 5G Mobile Networks. *Eindhoven University of Technology, DOI: 10.5281/zenodo.1403140.*

[18] OSI Model - Wikipedia. *https://en.wikipedia.org/wiki/OSI*$_{m}odel.$

[19] The OSI Model for Anyone. *https://medium.com/@agreene0308/the-osi-model-vs-the-tcp-ip-model-network-protocols-and-messages-e68df1af86e9.*

[20] 5G Network Architecture - RF Wireless World. *https://www.rfwireless-world.com/Tutorials/5G-network-architecture.html.*

[21] OpenAirInterface. *https://www.openairinterface.org/.*

[22] USRP B210 Software Defined Radio. *https://www.ettus.com/all-products/ub210-kit/.*

[23] LimeSDR USB - Myriad RF. *https://myriadrf.org/projects/component/limesdr/.*

[24] LTE Demo with LimeSDR and OpenAirInterface. *https://www.crowdsupply.com/lime-micro/limesdr/updates/oai-lte-demo.*

[25] 5G Protocol Stack - RF Wireless World. *https://www.rfwireless-world.com/Terminology/5G-Protocol-Stack-Layer-1-Layer-2-and-Layer-3.html.*

[26] IMU - Xsens 3D Motion Tracking. *https://www.xsens.com/tags/imu/.*

[27] IMU - Xsens 3D Motion Tracking. *https://www.xsens.com/tags/imu/.*

[28] Technical Specifications of the Kinect v2 (Fig from: A Post-Rectification Approach of Depth Images of Kinect v2 for 3D Reconstruction of Indoor Scenes. *https://www.researchgate.net/figure/Technical-specifications-of-the-Kinect-v2*$_{t}bl1_{3}21048476.$

[29] Jetson TX2 Module - NVidia Developer. *https://developer.nvidia.com/embedded/jetson-tx2.*

[30] BECKHOFF New Automation Technology - C6920 Control cabinet Industrial PC. *https://www.beckhoff.com/english.asp?industrial*$_{p}c/c6920.htm?id = 11792199638.$

[31] BECKHOFF New Automation Technology. *https://www.beckhoff.com/EtherCAT/.*

[32] EtherCAT Protocol Overview. *https://www.rtautomation.com/technologies/ethercat/.*

[33] Robotic Open Platform - Wiki. *http://roboticopenplatform.org/wiki/TURTLE.*

[34] A novel ball handling mechanism for the RoboCup middle size league, Jeroen de Best, René van de Molengraft and Maarten Steinbuch. *Mechatronics vol.21, no.2, pages:469 - 478, 2011.*

[35] A Gentle Introduction to the Tech United Turtle Software - Peter van Lith. *Technische Universiteit Eindhoven, version 2018-5, vanlith.peter@gmail.com.*

[36] Tech United Software Repository (SVN). *https://robocup.wtb.tue.nl/svn/techunited/trunk/src/Turtle2/.*

[37] Coordinating distributed autonomous agents with a real-time database: The CAMBADA project - L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva and L. Seabra Lopes. *Computer and Information Sciences - ISCIS 2004 Lecture Notes in Computer Science, vol 3280. Springer, Berlin, Heidelberg.*

[38] Rules and Regulations - RoboCup Soccer Mid Sized League. *https://msl.robocup.org/rules*.

[39] GStreamer - Open Source Multimedia Framework. *https://gstreamer.freedesktop.org/*.

[40] Samsung KV2Streamer - Github. *https://github.com/Samsung/kv2streamer*.

[41] Wireshark. *https://www.wireshark.org/*.

[42] iPerf - The ultimate speed test tool for TCP, UDP and SCTP. *https://iperf.fr/*.

[43] Home of the Network Time Protocol. *http://www.ntp.org/*.

[44] Ethernet-based precision timing enables real-time distributed military systems. *http://mil-embedded.com/articles/ethernet-based-distributed-military-systems/*.

[45] Precision Time Protocol - Wikipedia. *https://en.wikipedia.org/wiki/Precision*$_{T}ime_{P}rotocol$.

[46] How to make your computer a DevPC with Ubuntu 16.04. *http://www.techunited.nl/wiki/index.php?title=How_to_make_your_computer_a_DevPC_with_Ubuntu_16.04*.

[47] LimeSuite - Myriad RF. *https://wiki.myriadrf.org/Lime*$_{S}uite$.

[48] Testing the LimeSDR - Myriad RF. *https://wiki.myriadrf.org/Testing*$_{t}he_{L}imeSDR$.

[49] LimeSDR-USB Quick Test - Myriad RF. *https://wiki.myriadrf.org/LimeSDR-USB*$_{Q}uick_{T}est$.

[50] How to Connect OAI eNB with OAI UE without S1 Interface - OAI Tutorial. *https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/HowToConnectOAIENBWithOAIUEWithoutS1Interface*.

[51] NTP Server/Client Tutorial. *https://www.thegeekstuff.com/2014/06/linux-ntp-server-client/*.

A Appendix: Simulink Models

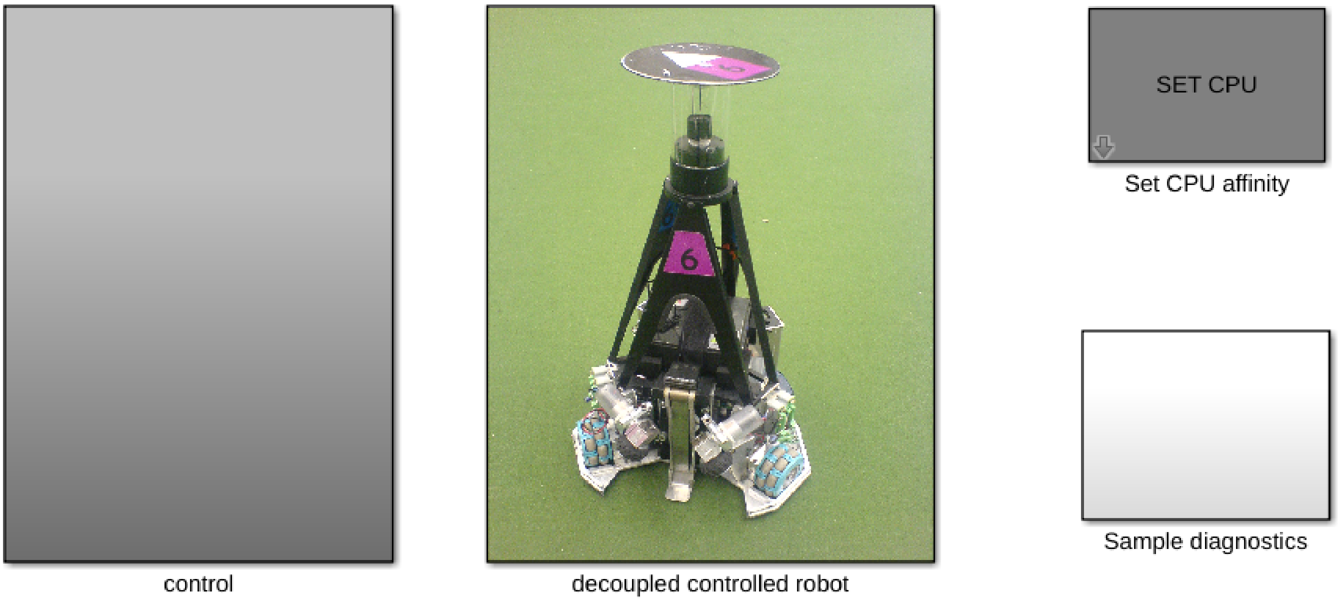


Figure A.1: The Motion simulink model and its subsystems. The control block contains the high level control and the decoupled controlled robot block contains the model shown in 6.1

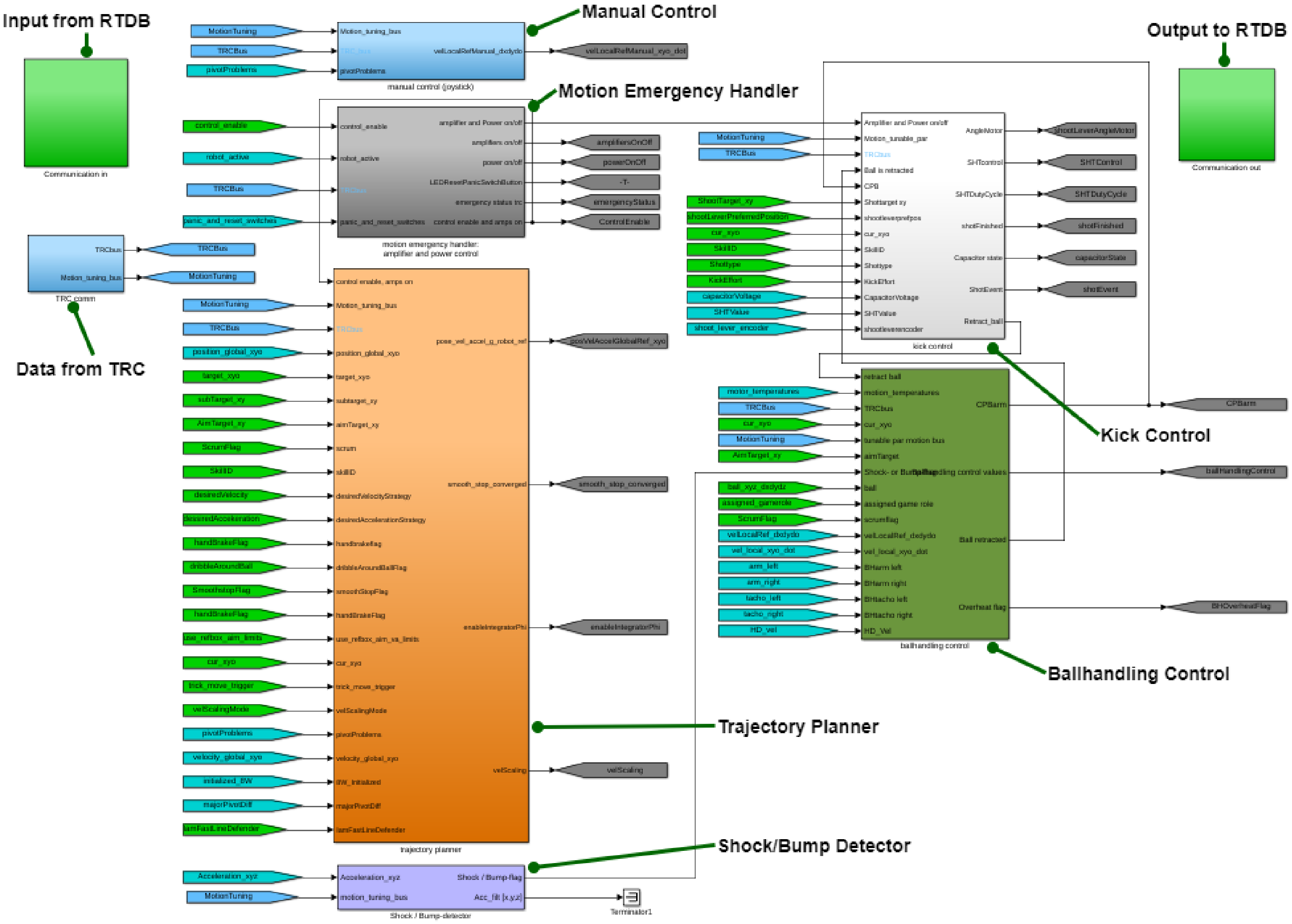


Figure A.2: The high level control subsystem in the Motion block. This block reads data from the low level control subsystem in order to plan the trajectory for the Turtle and control the ball-handling and kicking mechanisms. Additionally, this model allows input and output communication with the Real-Time Database. Also shown are the Manual Control block that reads data from the Turtle Remote Control, a Motion Emergency Handler and a Shock/Bump Detector.

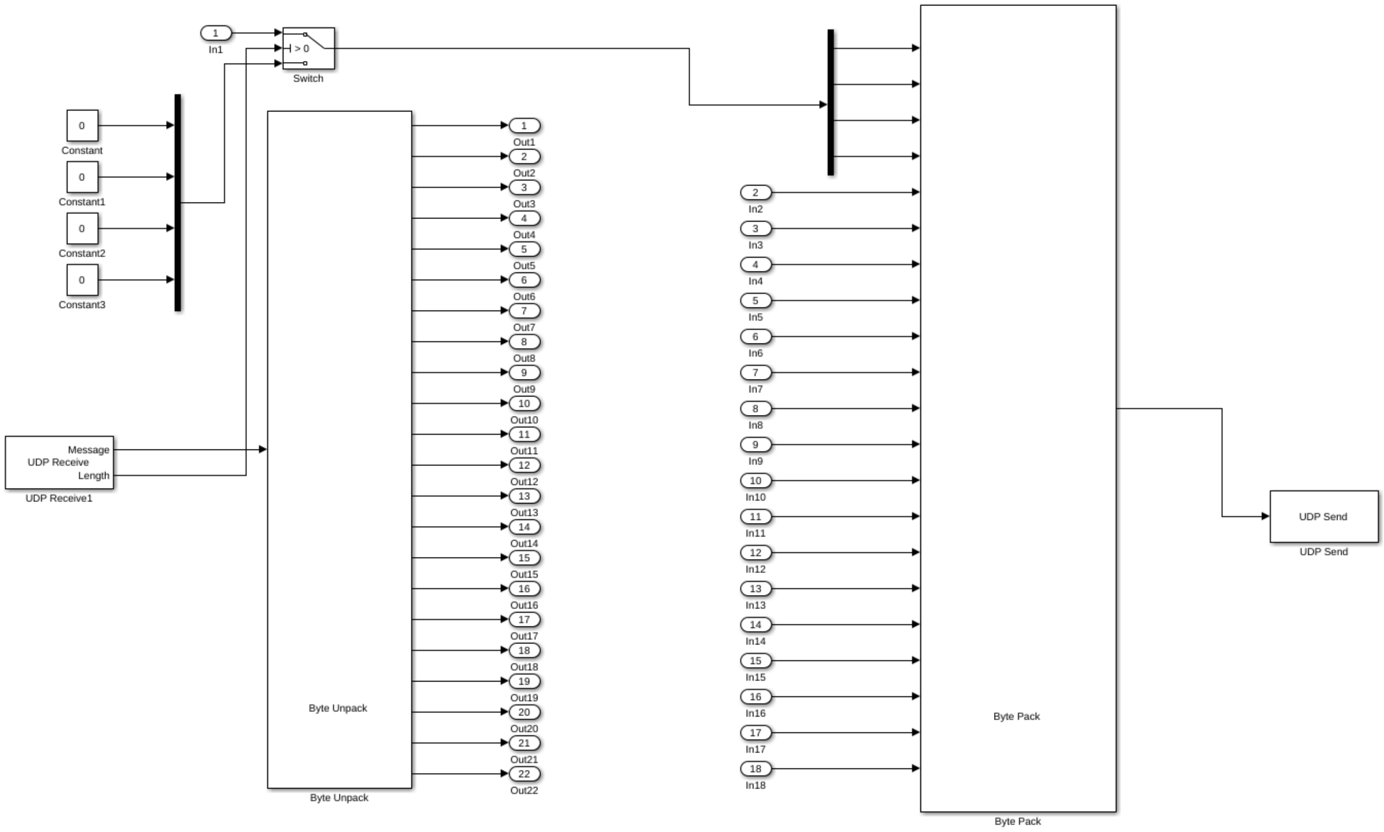


Figure A.3: In the eNB simulink model, the data acquisition block is replaced by this subsystem. UDP Receive is used to acquire sensor data sent from the UE. This data is unpacked and provided to the eNB software. The actuation commands generated in the eNB is packed and transmitted to the UE via UDP Send. As seen, a switch reads the length of the received signal and if there is no data being received, the Turtle is provided with arbitrary values, thus making it stop.

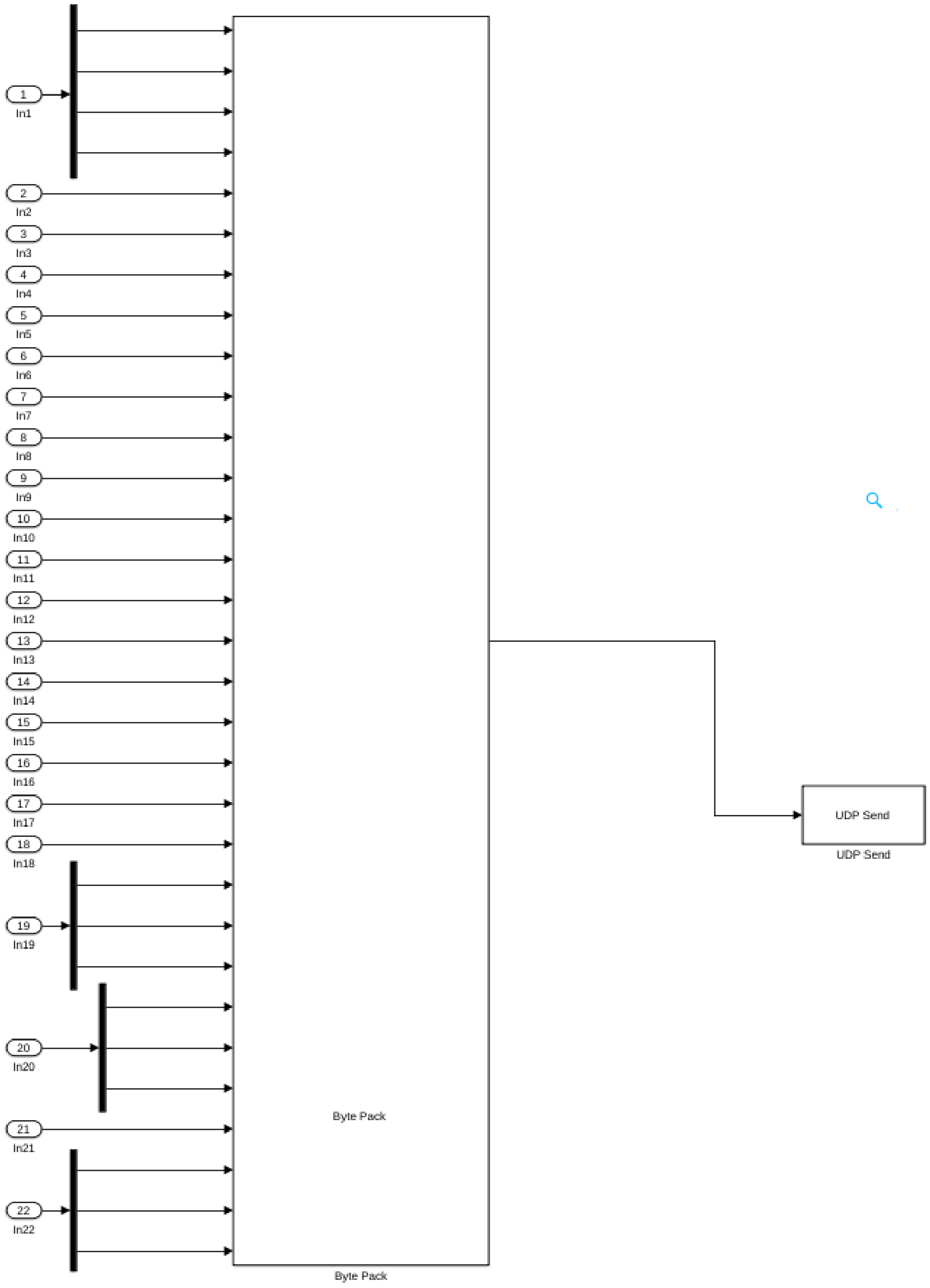


Figure A.4: In the UE software, the data acquisition block is connected to a UDP Send block that transmits sensor data to the eNB. This sensor data is acquired from the Beckhoff stack, packed into one array and transmitted using UDP Send.

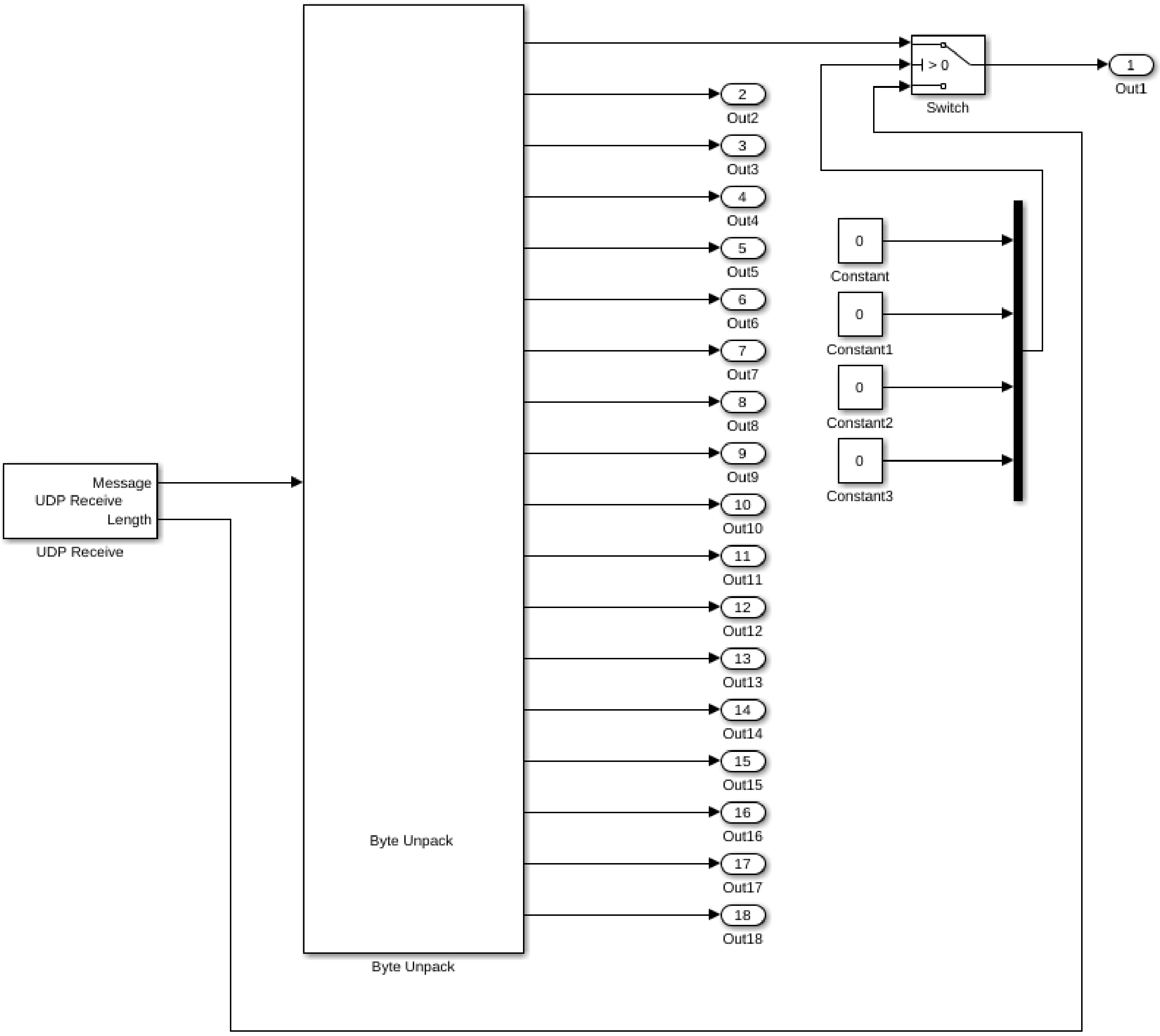


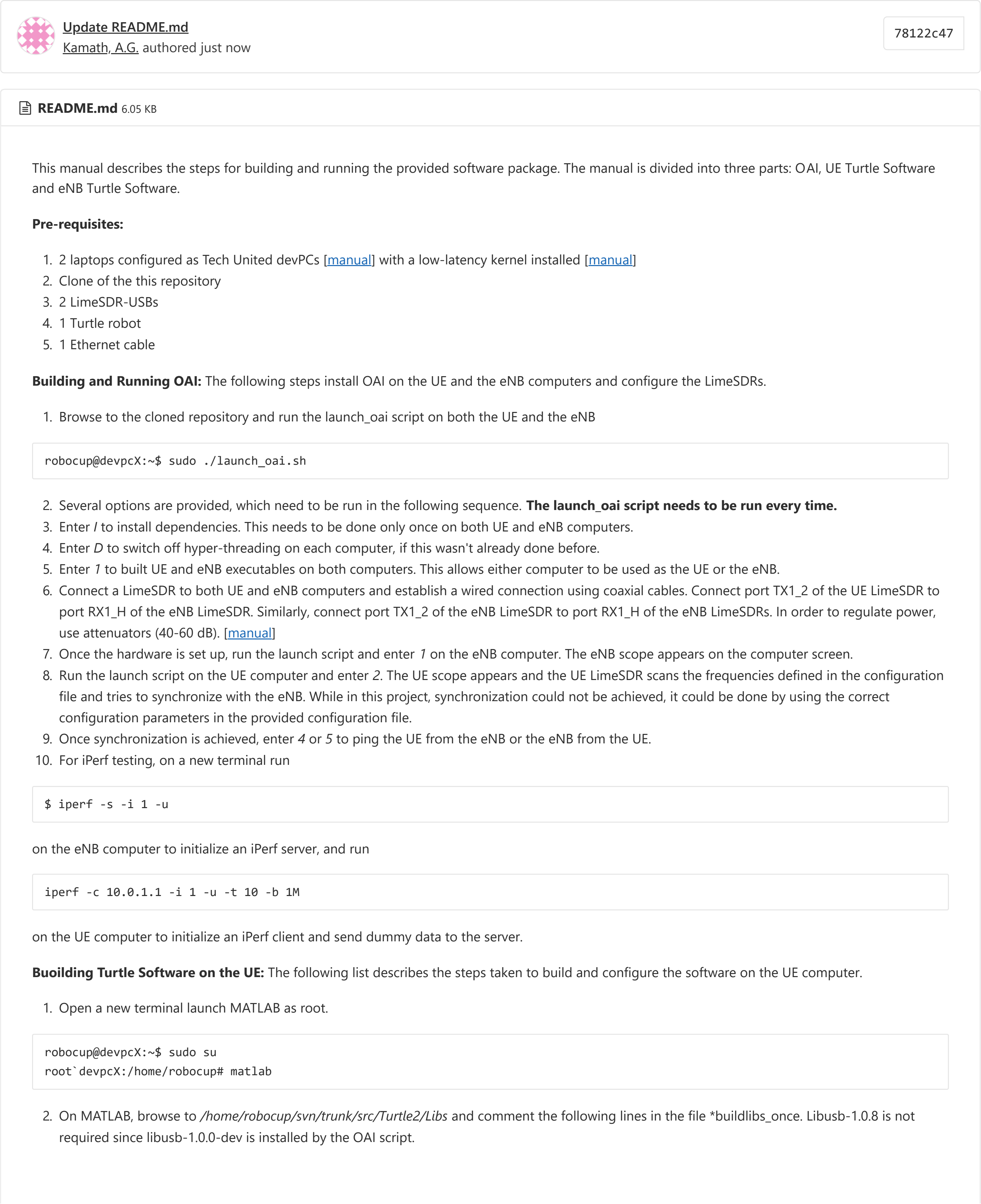
Figure A.5: In the UE software, the data acquisition block is connected to a UDP Receive block that receives actuation commands from the eNB. This actuation commands are received, unpacked and then provided to the Beckhoff stack so that the actuators can be updated. As shown, a switch is used to send arbitrary data to the motors if there is no data received from the eNB.

B Appendix: Scripts and Manuals

This appendix includes the following sections:

- B.1: Manual to build and run OAI, LimeSDR and the Turtle software on the UE and the eNB
- B.2: OAI script
- B.4: Turtle scripts

B.1 Manual



Update README.md

Kamath, A.G. authored just now

78122c47

README.md 6.05 KB

This manual describes the steps for building and running the provided software package. The manual is divided into three parts: OAI, UE Turtle Software and eNB Turtle Software.

Pre-requisites:

1. 2 laptops configured as Tech United devPCs [manual] with a low-latency kernel installed [manual]
2. Clone of the this repository
3. 2 LimeSDR-USBs
4. 1 Turtle robot
5. 1 Ethernet cable

Building and Running OAI: The following steps install OAI on the UE and the eNB computers and configure the LimeSDRs.

1. Browse to the cloned repository and run the launch_oai script on both the UE and the eNB

```
robocup@devpcX:~$ sudo ./launch_oai.sh
```

2. Several options are provided, which need to be run in the following sequence. **The launch_oai script needs to be run every time.**
3. Enter *I* to install dependencies. This needs to be done only once on both UE and eNB computers.
4. Enter *D* to switch off hyper-threading on each computer, if this wasn't already done before.
5. Enter *1* to built UE and eNB executables on both computers. This allows either computer to be used as the UE or the eNB.
6. Connect a LimeSDR to both UE and eNB computers and establish a wired connection using coaxial cables. Connect port TX1_2 of the UE LimeSDR to port RX1_H of the eNB LimeSDR. Similarly, connect port TX1_2 of the eNB LimeSDR to port RX1_H of the eNB LimeSDRs. In order to regulate power, use attenuators (40-60 dB). [manual]
7. Once the hardware is set up, run the launch script and enter *1* on the eNB computer. The eNB scope appears on the computer screen.
8. Run the launch script on the UE computer and enter *2*. The UE scope appears and the UE LimeSDR scans the frequencies defined in the configuration file and tries to synchronize with the eNB. While in this project, synchronization could not be achieved, it could be done by using the correct configuration parameters in the provided configuration file.
9. Once synchronization is achieved, enter *4* or *5* to ping the UE from the eNB or the eNB from the UE.
10. For iPerf testing, on a new terminal run

```
$ iperf -s -i 1 -u
```

on the eNB computer to initialize an iPerf server, and run

```
iperf -c 10.0.1.1 -i 1 -u -t 10 -b 1M
```

on the UE computer to initialize an iPerf client and send dummy data to the server.

Buoilding Turtle Software on the UE: The following list describes the steps taken to build and configure the software on the UE computer.

1. Open a new terminal launch MATLAB as root.

```
robocup@devpcX:~$ sudo su
root`devpcX:/home/robocup# matlab
```

2. On MATLAB, browse to */home/robocup/svn/trunk/src/Turtle2/Libs* and comment the following lines in the file *buildlibs_once. Libusb-1.0.8 is not required since libusb-1.0.0-dev is installed by the OAI script.

```
###### install lisusb versie -prehistorie ######
cd Libs/libusb-1.0.8
chmod 755 configure
./configure -q
sudo make -s install
```

3. While in the Libs directory */home/robocup/svn/trunk/src/Turtle2/Libs*, execute the following in the command window (you may have to chmod 777 the file to make it executable):

```
>> !./buildlibs_once
```

4. Browse to */home/robocup/svn/trunk/src/Turtle2* and run:

```
>> addpathrobocup
>> savepath
```

5. Copy *UE_turtleFiles/build_all_UE.m* to */home/robocup/svn/trunk/src/Turtle2*
6. Copy *UE_turtleFiles/build_motion_UE.m* and *UE_turtleFiles/motion_UE.slx* to */home/robocup/svn/trunk/src/Turtle2/Motion*
7. From */home/robocup/svn/trunk/src/Turtle2*, run the following in the command window:

```
>> make_all_install
>> build_all_UE
```

Building Turtle Software on the eNB: The following list describes the steps taken to build and configure the software on the eNB computer.

1. Follow steps 1-4 from the previous section.
2. Browse to */home/robocup/svn/trunk/src/Turtle2/Targets/mttarget* and open *mt_main.c*. At the beginning of the file, add the following line and save the file. This defines that the software will execute without needing an EtherCAT connection.

```
#define NO_EC 1
```

3. Copy *eNB_turtleFiles/build_all_eNB.m* to */home/robocup/svn/trunk/src/Turtle2*
4. Copy *eNB_turtleFiles/build_motion_eNB.m* and *eNB_turtleFiles/motion_eNB.slx* to */home/robocup/svn/trunk/src/Turtle2/Motion*
5. From */home/robocup/svn/trunk/src/Turtle2*, run the following in the command window:

```
>> make_all_install
>> build_all_eNB
```

Running Turtle Software on the UE and the eNB: Once the software is built on both UE and eNB, the following steps need to be followed:

1. Connect the Beckhoff Stack of the Turtle to the UE computer using an ethernet cable. Make sure that the ethernet interface is named *eth0*.
2. Browse to the cloned repository and run the *launch_turtle5g.sh* script on both the UE and the eNB computers.

```
$ sudo ./launch_turtle5g.sh
```

3. Several options are provided, which need to be run in the following sequence. **The launch_turtle5g script needs to be run every time.**
4. On the eNB, enter *1* to run the communication interfaces.
5. On a new terminal on the eNB, run the script and enter *2* to run Strategy.
6. On a new terminal on the eNB, run the script and enter *3* to run Motion_eNB.
7. On a new terminal on the eNB, run TRC as super user.

```
robocup@devpcX:~$ sudo su
root`devpcX:/home/robocup# matlab
```

8. Switch on the Turtle. On the UE, open a new terminal, run the script and enter *3* to run Motion_UE
9. If the network is configured and working, on the TRC window on the eNB computer, specify which Turtle is being used and open the simulator window. If everything was done correctly, the Turtle should be visible in the middle of the simulator window. If the Turtle is moved physically, the Turtle icon on the simulator should also move.
10. Using the TRC, the manual control can now be initialized and the Turtle can be controlled. For safety, please keep the wireless emergency switch handy.

B.2 OAI Script

```
#!/bin/bash
#Denys Nyzovets
#Linux devpc2 4.13.0-36-lowlatency #40~16.04.1-Ubuntu SMP PREEMPT Sat Feb 17 00:18:34 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
#Ubuntu 16.04
#Pin on LimeSDR: TX1_2, RX1_H
#Based on https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/HowToConnectOAIENBWithOAIUEWithoutS1Interface
#
echo;
echo "Select of OpenAIR interface:"
echo "  I-install dependency(one time per PC)"
echo "  D-switch off Hyper-Threading"
echo "  E-switch on Hyper-Threading"
echo "  1-build eNB and UE"
echo "  2-run eNB"
echo "  3-run UE"
echo "  4-Ping from UE to eNB"
echo "  5-Ping from eNB to UE"
echo "  6-UDP Test from UE to eNB"
echo "  Q-Exit"

read Keypress
case "$Keypress" in
  "I" )
  echo "Installation of dependency packages"
  sudo add-apt-repository -y ppa:myriadrf/drivers
        sudo apt-get update
        sudo apt-get install limesuite liblimesuite liblimesuite-dev limesuite-udev limesuite-images
        sudo apt-get install soapysdr soapysdr-module-lms7

        sudo apt-get install subversion git i7z cpufrequtils linux-image-lowlatency linux-headers-lowlatency

        #Power management
        sudo echo 'blacklist intel_powerclamp'>>/etc/modprobe.d/blacklist.conf
  #https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/OpenAirKernelMainSetup

        #Disable CPU Frequency sPing from UE to eNBcaling
        sudo echo 'GOVERNOR="performance"'>>/etc/default/cpufrequtils
        sudo update-rc.d ondemand disable
        sudo /etc/init.d/cpufrequtils restart
        #cpufreq-info
  ;;
  "D" )
  echo "Switch off Hyper-Threading"
  bash -c "echo 0 > /sys/devices/system/cpu/cpu1/online"
  bash -c "echo 0 > /sys/devices/system/cpu/cpu3/online"
  bash -c "echo 0 > /sys/devices/system/cpu/cpu5/online"
  bash -c "echo 0 > /sys/devices/system/cpu/cpu7/online"

  grep "" /sys/devices/system/cpu/cpu*/topology/core_id
  grep -q '^flags.*[[:space:]]ht[[:space:]]' /proc/cpuinfo && \
    echo "Hyper-threading is supported"
```

```
grep -E 'model|stepping' /proc/cpuinfo | sort -u
;;
"E" )
echo "Switch on Hyper-Threading"
bash -c "echo 1 > /sys/devices/system/cpu/cpu1/online"
bash -c "echo 1 > /sys/devices/system/cpu/cpu3/online"
bash -c "echo 1 > /sys/devices/system/cpu/cpu5/online"
bash -c "echo 1 > /sys/devices/system/cpu/cpu7/online"

grep "" /sys/devices/system/cpu/cpu*/topology/core_id

grep -q '^flags.*[[:space:]]ht[[:space:]]' /proc/cpuinfo && \
  echo "Hyper-threading is supported"

grep -E 'model|stepping' /proc/cpuinfo | sort -u
;;
"1" )
echo "eNB installation is selected"
      mkdir -p ~/openairinterface5g
      cd ~/
      git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git --single-branch --branch master
      cd ~/openairinterface5g/
      source oaienv
      cd cmake_targets
      ./build_oai --eNB --UE -w LMSSDR -x -c -C -I --install-system-files
./build_oai -w LMSSDR --build-lib telnetsrv
./build_oai -w LMSSDR --build-lib enbscope
./build_oai -w LMSSDR --build-lib uescope
./build_oai -w LMSSDR --build-lib msc
sudo cp ~/openairinterface5g/targets/bin/.ue_emm.nvram0 ~/openairinterface5g/targets/bin/.ue_emm.nvram
sudo cp ~/openairinterface5g/targets/bin/.ue.nvram0 ~/openairinterface5g/targets/bin/.ue.nvram
sudo cp ~/openairinterface5g/targets/bin/.usim.nvram0 ~/openairinterface5g/targets/bin/.usim.nvram
;;
"2" )
echo "Run eNB"
sudo rmmod nasmesh
sudo cp -u enb.band7.tm1.50PRB.usrpb210.conf ~/openairinterface5g/targets/PROJECTS/GENERIC-LTE-EPC/CONF/enb.band7.tm1.50PRB.usrpb210.conf
cd ~/openairinterface5g/
source oaienv
source ./cmake_targets/tools/init_nas_nos1 eNB
sudo -E ./targets/bin/lte-softmodem.Rel14 -d -O $OPENAIR_TARGETS/PROJECTS/GENERIC-LTE-EPC/CONF/enb.band7.tm1.50PRB.usrpb210.conf --rf-config-file ./targets/ARCH/LMSSDR/LimeSDR_above_1p8GHz_1v4.ini 2>&1 | tee ENB.log
;;
"3" )
echo "Run UE"
sudo rmmod nasmesh
cd ~/openairinterface5g/
source oaienv
source ./cmake_targets/tools/init_nas_nos1 UE
sudo -E ./targets/bin/lte-uesoftmodem.Rel14 -U 1 --noS1 -C 2660000000 -r 25 --ue-scan-carrier --ue-txgain 90 --ue-rxgain 115 -d --rf-config-file ./targets/
```

```
    ARCH/LMSSDR/LimeSDR_above_1p8GHz_1v4.ini >&1 | tee UE.log
  ;;
  "4" )
  echo "Ping from UE to eNB"
  ping 10.0.1.1 -c 10
  ;;
  "5" )
  echo "Ping from eNB to UE"
  ping 10.0.1.9 -c 10
  ;;
  "6" )
  echo "UDP Test from UE to eNB"
  #eNB iperf -s -i 1 -u
  #UE iperf -c 10.0.1.1 -i 1 -u -t 10 -b 1M
  ;;
  "Q" )
  echo "E-Exit is selected"
  exit 0
  ;;

esac
exit 0
```

Listing B.1: OAI Launch Script: launch_oai.sh

B.3 Turtle Manual

B.4 Turtle Scripts

Main Launch Script

```
#!/bin/bash
# Aditya Kamath
# Executed on both UE and eNB computers - some options run on UE, some on eNB
# Ubuntu 16.04
#
echo;
echo "Select Script:"
echo "  1 - Communication"
echo "  2 - Strategy"
echo "  3 - Motion"
echo "  4 - TRC"
echo "  a - Wireshark Capture"
echo "  b - iPerf UDP Test: Server"
echo "  c - iPerf UDP Test: Client (eNB)"
echo "  d - iPerf UDP Test: Client (UE)"
echo "  e - Log average CPU load"
echo "  f - Log CPU load per task"

read Keypress

cd launch_scripts
case "$Keypress" in
  "1" )
  echo "Running Communication"
  ./launch_comm.sh
  ;;
  "2" )
  echo "Running Strategy Executable"
  ./launch_strategy.sh
  ;;
  "3" )
  echo "Running Motion Executable"
  ./launch_motion.sh
  ;;
  "4" )
  echo "Running TRC"
  ./launch_trc.sh
  ;;
  "a" )
  echo "Running Wireshark"
  sudo wireshark -i eth0 -k -a duration:10 -w wiresharktest.pcapng
  ;;
  "b" )
  echo "Running iPerf Server"
  iperf -s -i 0.5 -u
  ;;
  "c" )
  echo "Running iPerf Client (eNB)"
  iperf -c 192.168.1.5 -u -d -b 20m
  ;;
  "d" )
  echo "Running iPerf Client (UE)"
  iperf -c 192.168.1.6 -u -d -b 20m
  ;;
  "e" )
  echo "Logging Uptime, Average Processor Load"
  while true; do uptime >> uptime.log; sleep 1; done
  ;;
  "f" )
  echo "Logging CPU load of top 10 most intensive tasks"
  while true; do (echo "%CPU, %MEM ARGS $(date)" && ps -e -o -pcpu,pmem,args --sort=pcpu | cut -d" " -f1-5 | tail) >> ps.log; sleep 5; done
  ;;

esac
exit 0
```

Listing B.2: UE and eNB main launch script to run Turtle software: launch_turtle5g.sh

Communication

```
#!/bin/bash
#Aditya Kamath
#Script to run communication interface for Turtle 2 on eNB
#
cd ~/svn/trunk/src/Turtle2/Libs/multicast/bin
sudo -s << COMM
export AGENT=2
./comm eth1
```

```
COMM
```

Listing B.3: Launch script for running the communication interfaces on the UE: launch_scripts/launch_comm.sh

Strategy

```
#!/bin/bash
#Aditya Kamath
#Script to run Strategy for Turtle 2 on eNB
#
cd ~/svn/trunk/src/Turtle2/Strategy
sudo -s << STRATEGY
export AGENT=2
./strategy
STRATEGY
```

Listing B.4: Launch script for running Strategy on the eNB: launch_scripts/launch_strategy.sh

Motion

```
#!/bin/bash
#Aditya Kamath
#Script to run Motion for Turtle 2 on UE and eNB
#
cd ~/svn/trunk/src/Turtle2/Motion
sudo -s << MOTION
#include MATLAB library path to run the UDP send and receive blocks
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/MATLAB/R2016a/bin/glnxa64
export AGENT=2
./motion_turtle -w
MOTION
```

Listing B.5: Launch script for running Motion on UE and the eNB: launch_scripts/launch_motion.sh

TRC

```
#!/bin/bash
#Aditya Kamath
#Script to run TRC on eNB
#
sudo -s << TRC
trc
TRC
```

Listing B.6: Launch script for running TRC on the eNB: launch_scripts/launch_trc.sh

C Appendix: Project Management Documents

The following project management documents have been attached in this appendix.

- C.1: Project Initiation Document
- C.2: Stakeholder Analysis
- C.3: Requirements Document
- C.4: Risk Assessment Document

C.1 Project Initiation Document

Enabling Remote Computation for Soccer Robots Using 5G mm-Waves

Project Initiation Document

Kamath, A.G.

EINDHOVEN UNIVERSITY OF TECHNOLOGY

CONTENTS

REVISION HISTORY

Version	Date	Description
0	10/01/2019	First draft
1	18/01/2019	Updated according to feedback from Simon
2	06/02/2019	Updated after Progress Update #2 (Updated problem description, scope, stakeholder analysis)
3	28/02/2019	Updated after Review Meeting #1 (Updated scope, risks, management plan)
4	16/03/2019	Updated V-model diagram and project timeline
5	07/06/2019	Updated title, work phase plan / timeline and final updates before doc. review

OVERVIEW

This document provides an overview of the planning for the final assignment for the PDEng Mechatronic Systems Design diploma. This project will be conducted under the supervision of dr.ir. Simon Rommel (representing blueSPACE) and ir. Jesse Scholtes (representing TU/e and High Tech Systems Center) alongside technical supervision from dr.ir. René van de Molengraft and the TU Eindhoven robotics team – Tech United. The PDEng program is managed by dr.ir. Peter Heuberger, who will be responsible for the administrative arrangements for the completion of the project and subsequent graduation.

PROJECT BACKGROUND

5G is a set of technologies that define the 5th generation of cellular mobile communications and succeeds the 4G cellular network that we currently use. Although still under development, 5G aims at providing higher data rates, reduced latency, energy/cost saving, higher system capacity, and massive device connectivity. [1]

The development of 5G technologies is not only led by companies like Qualcomm or Ericsson, but also by initiatives like the 5G PPP (5G Infrastructure Public Private Partnership). 5G PPP is a joint initiative between the European Commission and the European ICT industry (ICT manufacturers, telecommunications operators, service providers, SMEs and research institutions). In June 2017, 21 projects were launched under the second phase of the 5G PPP – one of them being blueSPACE, a three-year research project led by the Eindhoven University of Technology. [2]

blueSPACE: Building on the Use of Spatial Multiplexing 5G Network Infrastructures and Showcasing Advanced Technologies and Networking Capabilities

This project is a consortium that consists of fifteen partners from the European Union and aims at developing optical technologies to support next-generation wireless/5G NR technologies to allow high speeds, reduced latency and reduced power consumption. While the main objective of this project is to develop the infrastructure to support the next generation 5G communications, other objectives include the development of demonstrators to showcase future-oriented use-cases of 5G in multiple scenarios. [3]

PROJECT DESCRIPTION

The goal of this PDEng final assignment is to design and develop a demonstrator to showcase a robotic application communicating over 5G, using the communication architecture and hardware developed as a part of the blueSPACE project. This robot application is to be developed in collaboration with the TU/e robotics team – Tech United [4] in order to enhance the performance of their robots using 5G technologies. This project also involves the validation of the blueSPACE Performance KPIs (Key Performance Indicators) and the robot's timing requirements using the proposed use-case.

To prevent unnecessary damage to hardware during testing, the scope of this robotic application is limited to ground-based robots, and not aerial robots. In addition, the focus of this project is towards Industry 4.0 applications, and more specifically, collaborative teams of robots, in accordance with the blueSPACE Experimental Evaluation and Demonstration plans [5].

During the first review meeting on February 27, 2019, the trainee along with key stakeholders defined the use-case for the project as follows:

Distributed computing:

- ***Using the Tech United TURTLE platform, image and sensor data from the platform's sensors is streamed to a remote computer over 5G***
- ***The remote computer will process this information using the TURTLE's software and return the corresponding actuation commands back to the robot.***
- ***The network performance and the robot's performance will be analyzed and matched with the requirements***

The above use-case is explored in detail in the Architecture/Design document [7].

SCOPE

This section describes the tasks and delimitations (the tasks that are out of scope for the PDEng trainee) for this project. The following lists can be extended throughout the project after agreement with the stakeholders of this project

Tasks:

- Survey of 5G technologies and future trends of 5G technologies in Industry 4.0 use-cases
- A detailed survey of the Tech United robot soccer team and their Turtle platform [6] – including dataflow within the robots, communication with other robots and their corresponding timing/data size requirements. This is followed by the proposal and definition of a specific use-case.
- Derivation of requirements and specifications for the communication of the Turtles over a 5G network and for the interfacing of the 5G hardware with the robot hardware.
- Design and Implementation of the proposed application using blueSPACE 5G hardware and the Tech United Turtle software on either a complete robot or a stationary test rig.
- Survey and definition of measurement and evaluation plans to validate blueSPACE and Tech United KPIs and requirements
- Verification and validation of the implemented application
- Demonstration
- Technical documentation, presentations and frequent reporting

Delimitations:

- The design and development of the communication interface will be handled by the blueSPACE team, with recommendations from the trainee.
- Setup of the 5G network at the location of the robot demonstration - this will be handled by the blueSPACE team.
- Development of additional functionality for the Tech United robots is out of scope. Only the communication aspect will be modified according to the requirements of the project.

HIGH-LEVEL REQUIREMENTS

The following requirements provide a high-level view of the problem description. Additional requirements will be derived from these at later stages of the project.

1. The proposed application must define a use-case that is also applicable to the Industry 4.0 use-case described in the Experimental Evaluation and Demonstration plan [4] – Concentrated robots, Slow moving service vehicles, latency critical applications
2. The proposed application must demonstrate the advantages of using 5G low-latency communication in the field of robotics, for example, improved performance of the robots
3. The proposed application must use the Tech United TURTLE platform as the ground-based robot
4. The ground-based robot must communicate in real time over a single channel, bidirectional 5G connection with a remote computer/server
5. The proposed application must validate the End-to-End Latency KPI for 5G in robotics
6. The proposed application must adhere to timing requirements of the existing robot functionality

Derived requirements are documented in the Requirements document [8]

DELIVERABLES

The following deliverables will be provided to the concerned stakeholders during and at the end of the PDEng final assignment. The following list also describes tentative deadlines for the submission of the deliverables and their updates. These documents will be available to stakeholders throughout the project, in a shared repository.

1. **Project Management**
    - Project Initiation Document *(After each work-phase)*
    - Gantt Chart *(On-going document, in shared repository)*
    - Requirements Document *(After each work-phase)*
2. **Technical Documentation**
    - Architecture/Design Document *(After preparation phase)*
    - Hardware/software description, manual – Gitlab wiki *(On-going after preparation phase)*
    - Software packages *(to be updated on Gitlab repository)*
3. **Presentations**
    - Intermediate presentations *(for stakeholders and Project Steering Group during scheduled meetings)*
    - Return Day presentations *(for ASD/MSD trainees during scheduled dates)*
    - Final Presentation *(before defense)*
4. **Final Report**

STAKEHOLDERS AND CONCERNS

This project involves 4 key stakeholder groups, with blueSPACE being the project owner.

- blueSPACE *[Idelfonso Tafur Monroy, Simon Rommel, Dimitrios Konstantinou]*
- Tech United / Control Systems Technology Group (CST) *[René van de Molengraft, Wouter Kuijpers, Wouter Houtman, Yanick Douven, Jordy Senden]*
- High Tech Systems Center (HTSC) *[Jesse Scholtes]*
- PDEng Mechatronic Systems Design program *[Peter Heuberger]*

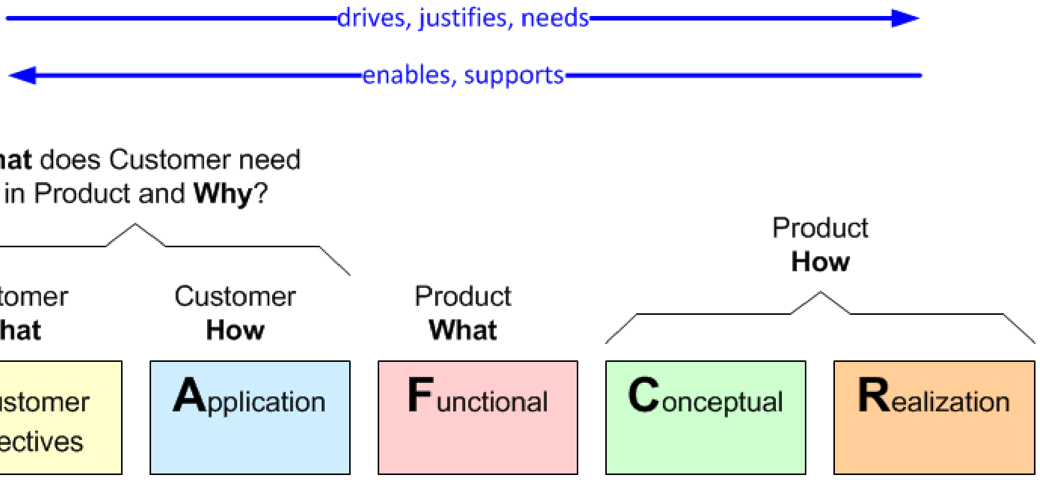
The attached Stakeholder Analysis document describes the stakeholders, their roles and concerns, as well as the approach towards communication and support.

DESIGN AND IMPLEMENTATION PLAN

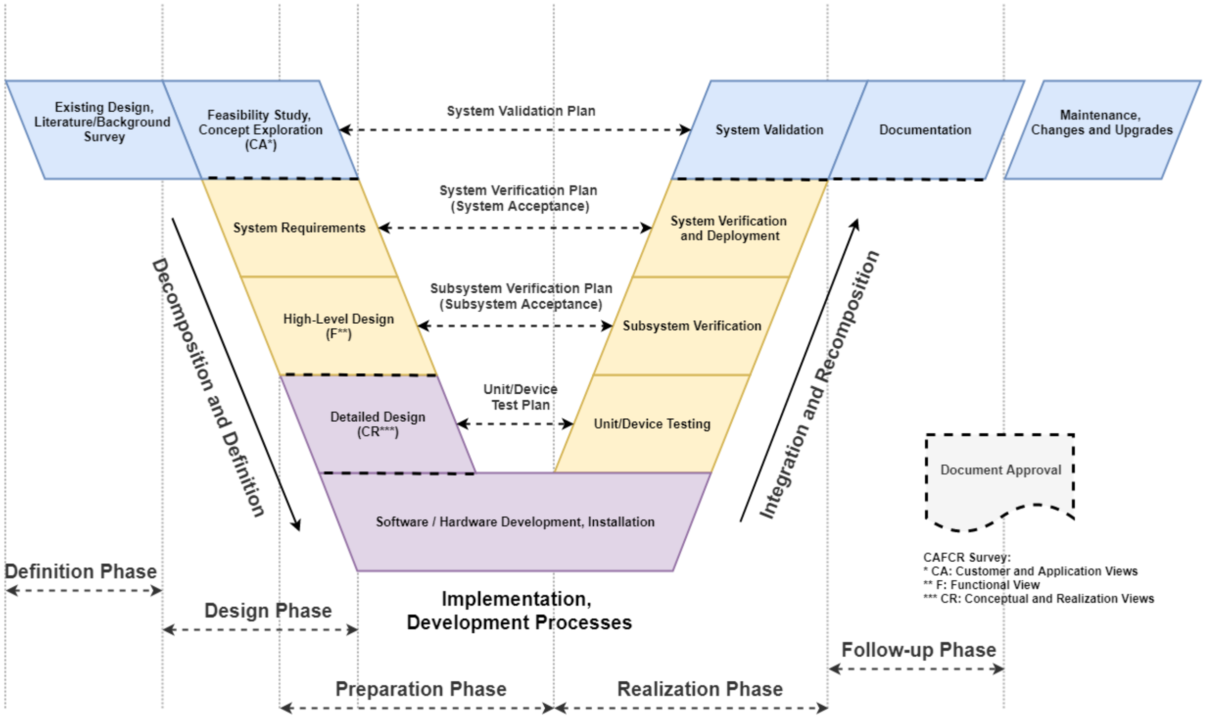
METHODOLOGY

The methodology involves the definition and planning of the project, definition of the system architecture and finally, design and implementation of the final solution. The following list describes the approach used during different parts of the project. The below list is later divided into work phases and will be described in the next section

1. **Definition and Planning**: The high-level requirements, scope and resources are defined and documented in the Project Initiation Document (this document. Additionally, a rough planning of the entire project is done using the Work Phase Plan and then detailed using a Gantt Chart
2. **System Architecture**: Once the project and its scope are defined, the CAFCR methodology is used to describe the project in the following categories. In each category, visual models, functional models and documents are used to communicate specifications and design choices.
   - **C**ustomer Objective View: Requirements, key drivers and other customer objectives
   - **A**pplication View: Stakeholders and concerns, context diagrams, use-cases/scenarios
   - **F**unctional View: Diagrammatic representation/map of technical functionality
   - **C**onceptual View: Detailed functional decomposition describing internal interfaces, issues like safety/reliability of functional units, integration plan, risks/uncertainties
   - **R**ealization View: Benchmarking, performance analysis, safety/reliability analysis



3. **Design and Implementation**: The V-Model is used to design, develop and test the various views from the CAFCR study. Tasks are broken down and managed using the Gantt chart developed in the first stage.


Existing Design, Literature/Background Survey
Feasibility Study, Concept Exploration (CA*)
System Validation Plan
System Validation
Documentation
Maintenance, Changes and Upgrades
System Requirements
System Verification Plan (System Acceptance)
System Verification and Deployment
High-Level Design (F**)
Subsystem Verification Plan (Subsystem Acceptance)
Subsystem Verification
Detailed Design (CR***)
Unit/Device Test Plan
Unit/Device Testing
Software / Hardware Development, Installation
Decomposition and Definition
Integration and Recomposition
Document Approval
CAFCR Survey:
* CA: Customer and Application Views
** F: Functional View
*** CR: Conceptual and Realization Views
Definition Phase
Design Phase
Implementation, Development Processes
Follow-up Phase
Preparation Phase
Realization Phase

WORK PHASE PLAN

The Work Phase Plan is divided into six phases. The following sections describe the tasks and outcomes expected in each phase.

INITIALIZATION PHASE

Tasks:

- Project kickoff: get an overview of the project, meet stakeholders/supervisors
- Literature survey: understand existing concepts/technologies, review potential use-cases/scenarios
- Project management: develop a rough plan for the project, review the technologies/resources needed, define high-level requirements, define project methodology

Outcomes:

- A generalized overview of the project (**Project Initiation Document**)
- Potential use-case/scenario alternatives
- Potential alternatives on the scope of the project

DEFINITION PHASE

Tasks:

- Formalize problem definition and scope of the project
- Finalize resources – robot hardware, software tools, use-cases

- Get feedback on draft Project Initiation Document
- Form Project Steering Group (finalize company and university supervisor)
- Finalize seating arrangements
- Detailed project planning using a Gantt Chart

Outcomes:

- Formalized problem description and high-level requirements (**Project Initiation Document, Requirements Document**)
- Formalized stakeholder analysis (**Project Initiation Document**)
- Risk assessment (**Project Initiation Document**)
- Project planning with high-level task breakdown (**Gantt Chart**)

DESIGN PHASE

Tasks:

- Architecture description for the robot (existing architecture and in potential use-cases)
- Detailed design – 5G interface and data streaming methodology
- Detailed task-breakdown & backlog for Preparation phase
- Derive low-level requirements – interface requirements, timing analysis, robot's timing requirements, robot's software/hardware requirements, etc.
- Prepare implementation and evaluation plans

Outcomes:

- Architecture description – Customer, Application and Functional views (**Architecture Description Document v1**)
- Draft Implementation/measurement/evaluation plans (**Architecture Description Document**)
- Updated **Project Initiation Document**
- Low-level requirements (**Requirements Document**)
- Task breakdown and management (**Gantt Chart**)

PREPARATION PHASE

Tasks:

- Make and finalize detailed plan about test setup (robot to base station) communication
- Setup communication software on both robot and base station
- Interface robot and base station computers with radio hardware
- Setup simple server-client setup for communication between the robot and the base station test setup
- Finalize implementation, evaluation and validation plans
- Planning of tests/trials, and required hardware/support

Outcomes:

- Required OS and communication software is setup on both robot and base station computers

- The robot is able to connect and communicate with the remote server (send and receive data) using the test setup
- 5G hardware is interfaced with the robot and the remote computer and is capable of communicating with the two machines.
- Detailed measurement, testing and KPI evaluation plans (**Architecture Description Document**)

REALIZATION PHASE

Tasks:

- Communicate bi-directional dummy data using the test setup
- Develop software packages to stream, receive and analyze:
    - Omnivision camera data
    - Kinect sensor data
    - IMU + EtherCAT data from robot to the remote server
    - EtherCAT data from the remote server to the robot
- Validate the software packages over the test setup network
- Define and plan for the final experiment – final hardware layout, access to lab and robots

Outcomes:

- Test network is validated and is able to be measured and analyzed
- The robot computer is able to stream sensor and image data to the base station over the test network
- The base station computer is able to stream actuation data to the robot over the test network
- The data streams can be measured and analyzed
- Detailed planning for validation stage (**Architecture Description Document, Gantt Chart**)
- A draft format for the final report is generated and a part of it is submitted for review (**Final Report**)

VALIDATION PHASE

Tasks:

- Integrate streaming and communication packages on the base station with the robot software
- Setup final test-bench and perform experiments according to validation plans
- Evaluate network performance, data stream performance and robot performance
- Document implementation process, execution process and results

Outcomes:

- Final demonstrator is setup according to the plans and the robot is able to communicate with the base station
- The robot is able to stream and receive data from the base station
- The data streams are able to achieve required specifications (data rate, latency) and the corresponding robot software blocks are able to run
- Comparison between current scenario and the proposed use-case is performed and documented (**Architecture Description Document**)

- Network, data stream and robot performance is analyzed and documented (**Architecture Description Document**)
- Software packages are pushed to the Gitlab repository (**Gitlab**)

FOLLOW-UP PHASE

Tasks:

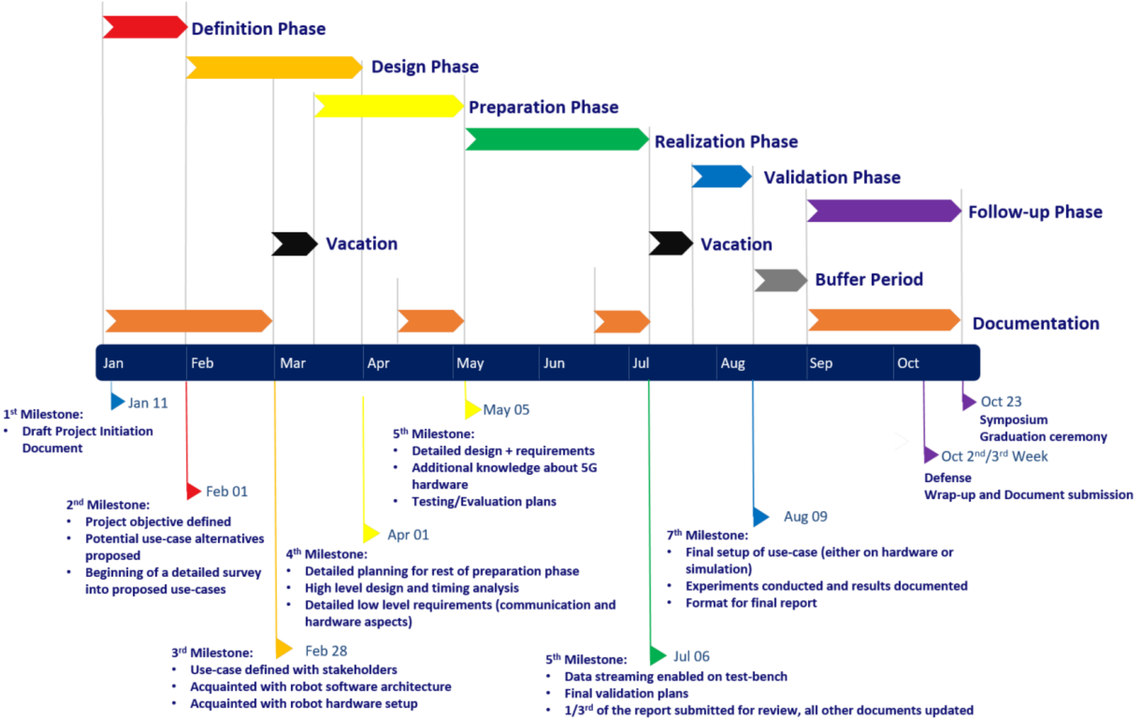
- Get feedback on progress and performance during the entire project
- Document implementation and evaluation details on the Gitlab Wiki
- Write the rest of the final report document
- At least 2 document review sessions (with stakeholders, professional writing coach)
- Final report submission
- Final presentation/defense

Outcomes:

- Final Report, Final Presentations, Gitlab repository and Wiki are submitted/reviewed
- Feedback from stakeholders, program management
- Final defense (**Final Report + Final Presentation**), Graduation – PDEng Diploma

MANAGEMENT PLAN

TIME



1st Milestone:
- Draft Project Initiation Document

2nd Milestone:
- Project objective defined
- Potential use-case alternatives proposed
- Beginning of a detailed survey into proposed use-cases

3rd Milestone:
- Use-case defined with stakeholders
- Acquainted with robot software architecture
- Acquainted with robot hardware setup

4th Milestone:
- Detailed planning for rest of preparation phase
- High level design and timing analysis
- Detailed low level requirements (communication and hardware aspects)

5th Milestone:
- Detailed design + requirements
- Additional knowledge about 5G hardware
- Testing/Evaluation plans

5th Milestone:
- Data streaming enabled on test-bench
- Final validation plans
- 1/3rd of the report submitted for review, all other documents updated

7th Milestone:
- Final setup of use-case (either on hardware or simulation)
- Experiments conducted and results documented
- Format for final report

Defense
Wrap-up and Document submission

Symposium
Graduation ceremony

Start Date: 17th December 2018

Phase	Start Date	End Date	Time Period
Initialization Phase*	17th December 2018	11th January 2019	20 Days
Definition Phase	14th January 2019	1st February 2019	15 Days
Design Phase	4th February 2019	30th March 2019	33 Days
Vacation 1	1st March 2019	12th March 2019	8 Days
Preparation Phase**	13th March 2019	31st May 2019	53 Days
Realization Phase	3rd June 2019	5th July 2019	24 Days
Vacation 2	8th July 2019	18th July 2019	9 Days
Validation Phase	19th July 2019	9th August 2019	16 Days
Buffer Period**	12th August 2019	30th August 2019	15 Days
Follow-up Phase	1st September 2019	18th October 2019	35 Days

End Date: 18th October 2019 (tentative)

* This phase also includes the Christmas holidays

** The buffer period is an accumulation of the buffer time required for each of the other phases of the project. If a particular phase takes longer than expected, time will be taken from the buffer period.

Scheduled holidays are mentioned in the PDEng arrangement document [9]

RESOURCES

This project requires the following resources:

- Seating arrangements – to be provided by TU/e Robotics Lab
- Lab access – to be provided by blueSPACE
- Ground-based robot or hardware to build a robot – to be provided by TU/e Robotics Lab and Tech United
- 5G communication hardware – to be provided by blueSPACE
- Software components – MATLAB/Simulink, MS Office licenses are provided by TU Eindhoven. Additionally, open-source tools will be used

QUALITY

Quality is maintained by regular meetings with stakeholders and supervisors. The following meetings/review sessions are organized:

- **Progress Update Meetings**: Once every two weeks with key stakeholders from blueSPACE and TechUnited. Progress since the previous progress update will be shared along with next steps for the following weeks. Next progress update meeting is planned at this time.
- **Review Meetings**: At the end of every phase with relevant stakeholders to review design, technology, process and documentation. Some review meetings might coincide with the bi-weekly Progress Update Meetings
- **Weekly Tech United Meetings**: Weekly meetings organized every Wednesday by Tech United for students and team members who are working on the soccer robots. Weekly progress pertaining to the Turtle platform will be shared and comments/feedback will be noted
- **Return Days**: Presentations organized by the PDEng ASD/MSD management to gain relevant feedback from other ASD/MSD trainees and Professional Development coaches.
- **Project Steering Group Meetings**: Organized for key stakeholders and the PDEng ASD/MSD management to review progress and performance, and raise concerns if necessary

INFORMATION

• CODING PROCEDURE AND DISTRIBUTION

The code will be written using guidelines provided by Tech United. Distribution and version control will be done using a Gitlab repository shared with the stakeholders during the project. Document sharing will be done using the blueSPACE Sharepoint folder

• AUTHORIZATION AND APPROVAL OF DOCUMENTS

All documents are reviewed by key stakeholders at blueSPACE and Tech United. The final report will also be sent to the PDEng Professional Writing coaches for feedback. All documents are shared by email. For some documents, a local copy will be updated regularly by the trainee. The updates of these documents will be available upon request.

RISKS AND UNCERTAINTIES

Risks and uncertainties are documented in the attached Risk Assessment document [10].

REFERENCES

[1] 5G – Wikipedia (https://en.wikipedia.org/wiki/5G)

[2] 5G Infrastructure Public Private Partnership (https://5g-ppp.eu/)

[3] blueSPACE – 5GPPP (https://blueSPACE-5gppp.eu/)

[4] Tech United (http://www.techunited.nl)

[5] blueSPACE: WP6 – Experimental Evaluation and Demonstration, Simon Rommel

[6] Tech United Eindhoven Wiki (http://www.techunited.nl/en/soccer_robots)

[7] Architecture/Design document, Aditya Kamath

[8] Requirements document, Aditya Kamath

[9] PDENG MSD Final project arrangements, Dr. P.S.C. (Peter) Heuberger

[10] Risk Assessment document, Aditya Kamath

C.2 Stakeholder Analysis

Stakeholder Name	Stakeholder Group	Communications Approach	Concerns	Current Status	Desired Support	Project Roles	Actions Desired	Actions and Communication
Idelfonso Tafur Monroy	blueSPACE	Keep informed via Simon and Dimitrios	> Design and develop a robotics application to demonstrate a 5G use-case, > Validation of End-to-End latency KPI in an Industry 4.0 robotic application	High	Low	Project Owner		Project Steering Group meetings
Simon Rommel	blueSPACE	Keep satisfied		Supporter/Critic	High	Project Owner / Supervisor	Feedback on design, 5G hardware, guidance on evaluation methodology, tools	Progress Update meetings, Review Meetings, Project Steering Group meetings
Dimitrios Konstantinou	blueSPACE	Keep satisfied		Supporter/Critic	High	Supervisor	Feedback on design, 5G hardware, guidance on evaluation methodology, tools	Progress Update meetings, Review Meetings, Project Steering Group meetings
Jesse Scholtes	High Tech Systems Center	Keep satisfied	> Provide administrative support and guidance to the PDEng trainee, > To enable collaboration between blueSPACE/5G related activites and Mechanical Engineerting, > To supervise daily activies of trainee and engage in technical discussions during meetings	Supporter/Critic	Medium	Project Manager / Daily Supervisor		Progress Update meetings, Review Meetings, Project Steering Group meetings, emails and regular conversations
René van de Molengraft	Tech United	Keep satisfied	> To enable research in collaborative/networked robots using 5G, > To provide technical guidance to the PDEng trainee, > To assist with Tech United related queries as Technical Director of the team	Supporter/Critic	Medium	Technical Supervisor	Feedback on design/architecture	Review meetings, Project Steering Group meetings, Meetings for technical guidance
Peter Heuberger	PDEng ASD/MSD	Keep informed	> Timely completion of the project, > Quality of the project (Process and Product)	Advocate	Low	Program Manager		Project Steering Group meetings
Wouter Kuijpers, Wouter Houtman, Jordy Senden, Yanick Douven	Tech United	Keep informed, ask for help if required, get feedback	> To provide guidance and support on the soccer robots to the trainee, > To help the trainee with material (hardware/software) and provide contacts to original developers of the software, > To enhance the performance of the robots by using 5G	Supporter	High	Tech United team members (Technical Guides)	Robot hardware, guidance on software and architecture	Tech United stand-up meetings, Regular conversations at robotics lab, Doubt clearing sessions if required

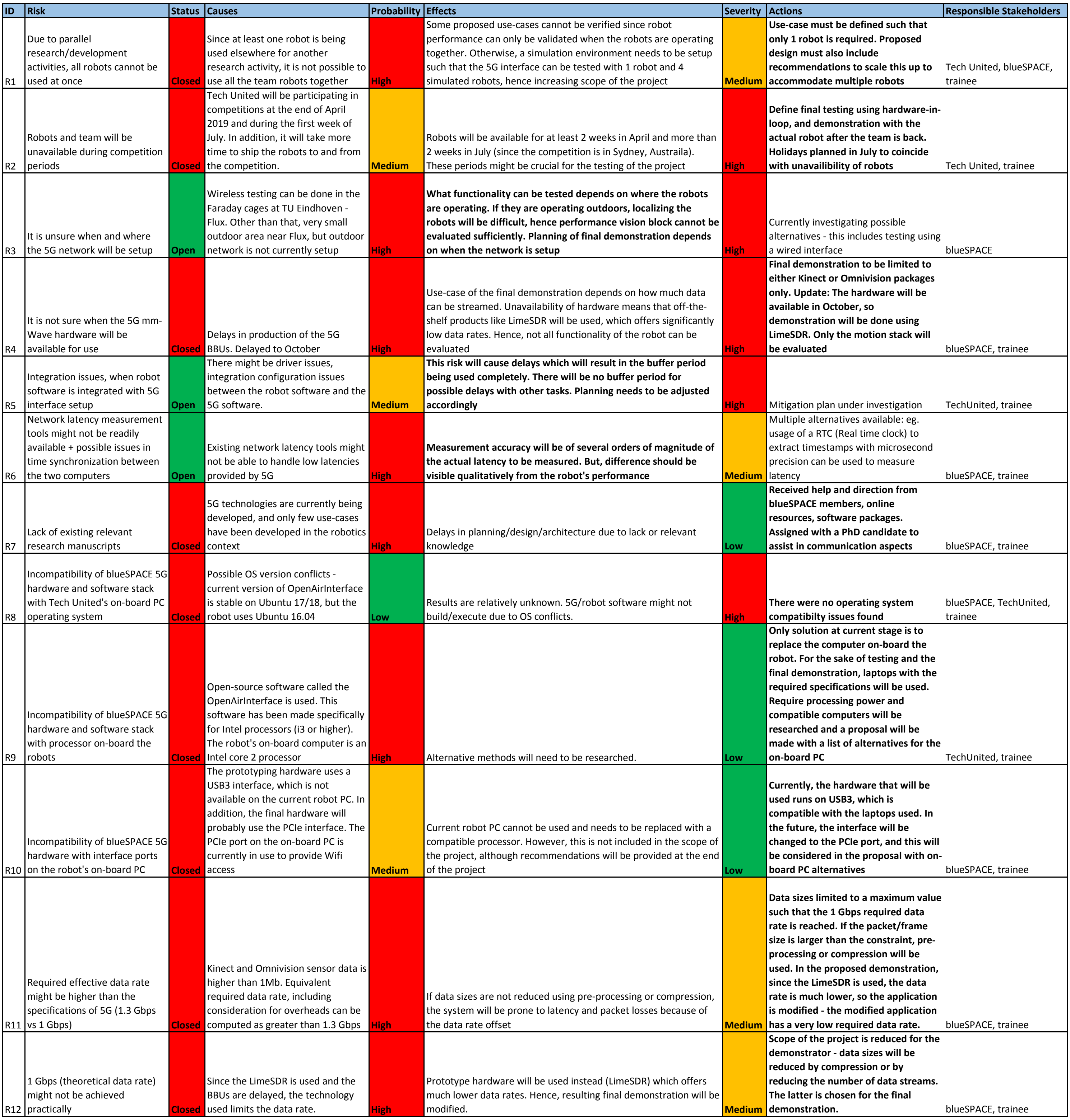
Meeting Type	Frequency
Progress Update meetings	Every week (Jesse to attend every second week)
Project Steering Group meetings	Every 6 weeks
Review Meetings	At the end of each work phase
Tech United stand-up meetings	Every week (Wednesday)
Technical meetings with René	Every month

C.3 Requirements Document

	High Level Requirements	
ID	Requirement	Low Level Reqs.
H1	The proposed application must demonstrate the advantages of using 5G communication in the field of robotics	L1
H2	The proposed application must involve a ground-based robot	L2
H3	The ground-based robot must communicate in real time over a single channel, bidirectional 5G connection with a remote computer/server	L3, L4, L5, L6, L7, L8, L9
H4	The proposed use-case scenarios must be limited to potential Industry 4.0 applications	L10
H5	The proposed application must focus on and validate the Throughput and End-to-End Latency KPIs	L11, L12

	Low Level Requirements	
ID	Requirement	Derived from
L1	The proposed application must clearly define the added-value for the individual robot application as well as for a team of robots	H1
L2	The proposed application must use the Tech United TURTLE platform	H2
L2.1	The proposed application must use 1 robot (due to availability issues of the robots and 5G hardware)	L2
L2.2	The proposed application must be scalable for the entire team	L2
L2.3	The proposed application must be demonstrable using 1 Turtle robot	L2
L3	The proposed application must achieve 5G requirements set by blueSPACE/5G PPP	H3
L3.1	Minimum data rate of the network should be 1 Gbps	L3
L3.2	Maximum latency of the network should be 1 ms	L3
L4	The proposed application must use OpenAirInterface to setup the on-board and remote computers as User Equipment and E-Node B (Base Station)	H3
L4.1	The proposed application must use the LimeSDR-USB as a prototype to communicate data over the physical layer of the OSI model	L4
L4.2	The OpenAirInterface software must be integrated with the LimeSDR-USB software packages and drivers	L4
L4.3	The OpenAirInterface installation must use the Linux Low-Latency Kernel	L4
L4.4	The LimeSDR must be connected to the on-board and remote computers using a USB3 connection	L4
L4.5	The robot computer and the remoter computer must both run on Intel i3 or higher processors	L4
L5	The proposed application must enable streaming of the Omnivision camera stream from the robot to the remote server	H3
L5.1	The size of each image frame (including overhead) should not exceed 1MB. If exceeded, either pre-processing or compression must be used to reduce the size within limits	L5
L5.2	The resolution of the transmitted images must be as specified by the robot software (Vision block)	L5
L5.3	The remote server must receive omnivision images at 60 frames per second (fps) with a latency as per the network setup (Max. data rate of 480 Mbps)	L5
L5.4	The Omnivision camera must be connected via a GigE interface to the on-board computer	L5
L6	The proposed application must enable streaming of the Kinect camera/sensor stream from the robot to the remote server	H3
L6.1	The size of each image/depth frame (including overhead) should not exceed 1MB. If exceeded, either pre-processing or compression must be used to reduce the size within limits	L6
L6.2	The resolution of the transmitted data must be as specified by the robot software (Kinect block)	L6
L6.3	The remote server must receive the Kinect stream at 30 frames per second (fps) with a latency as per the network setup (Max. data rate of 240 Mbps)	L6
L6.4	The Kinect v2 camera must be connected via a USB3 interface to the on-board computer	L6
L7	The proposed application must enable bi-directional streaming of sensor(EtherCAT, IMU) and actuation(EtherCAT) data between the robot and remote server	H3
L7.1	EtherCAT and IMU data must be packaged and transmitted from the robot. The frame size must not exceed 1KB	L7
L7.2	Actuation commands (EtherCAT messages) must be transmitted from the remote server. The frame size must not exceed 1KB	L7
L7.3	The remote server must receive the sensor data stream at 1000Hz with a latency as per network setup (Max. data rate of 8 Mbps)	L7
L7.4	The robot computer must receive the actuation command stream at 1000Hz with a latency as per network setup (Max. data rate of 8 Mbps)	L7
L7.5	Time delay of the remote server process (between receiving sensor data and transmitting actuation data) must be less than 1ms	L7
L7.6	Time delay of the robot process (between receiving actuation data and transmitting next sensor data) must have a deadline of 1ms	L7
L7.7	The Beckhoff Stack (EtherCAT) must be connected to the on-board computer via an Ethernet cable	L7
L7.8	The IMU must be connected over USB to the on-board computer	L7
L8	The proposed application must define the optimal specifications for the on-board (robot) and remote computers for the proposed use-case	H3
L9	The proposed application must propose recommendations for future demonstrations and integration of a team of robots	H3
L10	The proposed application must define the added-value of the demonstration for fleets of industrial robots	H4
L11	The proposed application must validate/analyze the 5G interface and provide recommendations based on it	H5
L12	The proposed application must validate/analyze the robot's data streams and the robot's performance with 5G and provide recommendations based on it	H5

C.4 Risk Assessment Document



ID	Risk	Status	Causes	Probability	Effects	Severity	Actions	Responsible Stakeholders
R1	Due to parallel research/development activities, all robots cannot be used at once	Closed	Since at least one robot is being used elsewhere for another research activity, it is not possible to use all the team robots together	High	Some proposed use-cases cannot be verified since robot performance can only be validated when the robots are operating together. Otherwise, a simulation environment needs to be setup such that the 5G interface can be tested with 1 robot and 4 simulated robots, hence increasing scope of the project	Medium	**Use-case must be defined such that only 1 robot is required. Proposed design must also include recommendations to scale this up to accommodate multiple robots**	Tech United, blueSPACE, trainee
R2	Robots and team will be unavailable during competition periods	Closed	Tech United will be participating in competitions at the end of April 2019 and during the first week of July. In addition, it will take more time to ship the robots to and from the competition.	Medium	Robots will be available for at least 2 weeks in April and more than 2 weeks in July (since the competition is in Sydney, Austraila). These periods might be crucial for the testing of the project	High	**Define final testing using hardware-in-loop, and demonstration with the actual robot after the team is back. Holidays planned in July to coincide with unavailibility of robots**	Tech United, trainee
R3	It is unsure when and where the 5G network will be setup	Open	Wireless testing can be done in the Faraday cages at TU Eindhoven - Flux. Other than that, very small outdoor area near Flux, but outdoor network is not currently setup	High	**What functionality can be tested depends on where the robots are operating. If they are operating outdoors, localizing the robots will be difficult, hence performance vision block cannot be evaluated sufficiently. Planning of final demonstration depends on when the network is setup**	High	Currently investigating possible alternatives - this includes testing using a wired interface	blueSPACE
R4	It is not sure when the 5G mm-Wave hardware will be available for use	Closed	Delays in production of the 5G BBUs. Delayed to October	High	Use-case of the final demonstration depends on how much data can be streamed. Unavailability of hardware means that off-the-shelf products like LimeSDR will be used, which offers significantly low data rates. Hence, not all functionality of the robot can be evaluated	High	**Final demonstration to be limited to either Kinect or Omnivision packages only. Update: The hardware will be available in October, so demonstration will be done using LimeSDR. Only the motion stack will be evaluated**	blueSPACE, trainee
R5	Integration issues, when robot software is integrated with 5G interface setup	Open	There might be driver issues, integration configuration issues between the robot software and the 5G software.	Medium	**This risk will cause delays which will result in the buffer period being used completely. There will be no buffer period for possible delays with other tasks. Planning needs to be adjusted accordingly**	High	Mitigation plan under investigation	TechUnited, trainee
R6	Network latency measurement tools might not be readily available + possible issues in time synchronization between the two computers	Open	Existing network latency tools might not be able to handle low latencies provided by 5G	High	**Measurement accuracy will be of several orders of magnitude of the actual latency to be measured. But, difference should be visible qualitatively from the robot's performance**	Medium	Multiple alternatives available: eg. usage of a RTC (Real time clock) to extract timestamps with microsecond precision can be used to measure latency	blueSPACE, trainee
R7	Lack of existing relevant research manuscripts	Closed	5G technologies are currently being developed, and only few use-cases have been developed in the robotics context	High	Delays in planning/design/architecture due to lack or relevant knowledge	Low	**Received help and direction from blueSPACE members, online resources, software packages. Assigned with a PhD candidate to assist in communication aspects**	blueSPACE, trainee
R8	Incompatibility of blueSPACE 5G hardware and software stack with Tech United's on-board PC operating system	Closed	Possible OS version conflicts - current version of OpenAirInterface is stable on Ubuntu 17/18, but the robot uses Ubuntu 16.04	Low	Results are relatively unknown. 5G/robot software might not build/execute due to OS conflicts.	High	**There were no operating system compatibilty issues found**	blueSPACE, TechUnited, trainee
R9	Incompatibility of blueSPACE 5G hardware and software stack with processor on-board the robots	Closed	Open-source software called the OpenAirInterface is used. This software has been made specifically for Intel processors (i3 or higher). The robot's on-board computer is an Intel core 2 processor	High	Alternative methods will need to be researched.	Low	**Only solution at current stage is to replace the computer on-board the robot. For the sake of testing and the final demonstration, laptops with the required specifications will be used. Require processing power and compatible computers will be researched and a proposal will be made with a list of alternatives for the on-board PC**	TechUnited, trainee
R10	Incompatibility of blueSPACE 5G hardware with interface ports on the robot's on-board PC	Closed	The prototyping hardware uses a USB3 interface, which is not available on the current robot PC. In addition, the final hardware will probably use the PCIe interface. The PCIe port on the on-board PC is currently in use to provide Wifi access	Medium	Current robot PC cannot be used and needs to be replaced with a compatible processor. However, this is not included in the scope of the project, although recommendations will be provided at the end of the project	Low	**Currently, the hardware that will be used runs on USB3, which is compatible with the laptops used. In the future, the interface will be changed to the PCIe port, and this will be considered in the proposal with on-board PC alternatives**	blueSPACE, trainee
R11	Required effective data rate might be higher than the specifications of 5G (1.3 Gbps vs 1 Gbps)	Closed	Kinect and Omnivision sensor data is higher than 1Mb. Equivalent required data rate, including consideration for overheads can be computed as greater than 1.3 Gbps	High	If data sizes are not reduced using pre-processing or compression, the system will be prone to latency and packet losses because of the data rate offset	Medium	**Data sizes limited to a maximum value such that the 1 Gbps required data rate is reached. If the packet/frame size is larger than the constraint, pre-processing or compression will be used. In the proposed demonstration, since the LimeSDR is used, the data rate is much lower, so the application is modified - the modified application has a very low required data rate.**	blueSPACE, trainee
R12	1 Gbps (theoretical data rate) might not be achieved practically	Closed	Since the LimeSDR is used and the BBUs are delayed, the technology used limits the data rate.	High	Prototype hardware will be used instead (LimeSDR) which offers much lower data rates. Hence, resulting final demonstration will be modified.	Medium	**Scope of the project is reduced for the demonstrator - data sizes will be reduced by compression or by reducing the number of data streams. The latter is chosen for the final demonstration.**	blueSPACE, trainee

Where innovation starts