

Robot Arm Applicability Assessment for Gravity Offloading in Mechanisms Testing

Pedro Jorge Rodrigues Lopes

Master Dissertation

Supervisor: Prof. António Mendes Lopes

Co-Supervisor at INEGI: Eng. Luís Carlos Moreira

Co-Supervisor at INEGI: Eng. Ricardo Lopes



Master in Mechanical Engineering

July 2022

Aos meus pais e irmã

Abstract

The demand to test aerospace equipment on Earth before being launched into space leads to the simulation of orbit conditions, namely zero-gravity conditions. There is a wide range of passive equipment to perform gravity compensation but with limitations, normally associated with the complex pathway configuration. Active zero gravity systems have been developed to mitigate the difficulties of passive ones.

The appearance of robotic arms with integrated force control, high-precision movements, and trajectories that could be complex and easily reprogrammable opens the door to applying and creating active gravity compensation solutions. That being said, the goal of this dissertation was to test and validate the usage of a Kuka IIWA 14 robot in terrestrial testing for active gravity compensation. It began with defining a case study in which the performance of the robot could be evaluated using external and robot-inbuilt instrumentation. A Beckhoff setup was arranged up to control the components of the case study and perform communication between the robot controller and a computer running an IoT platform based on Node-RED in order to connect everything and make it possible to exchange and acquire information.

Multiple control techniques were designed and tested based on several approaches for developing a test plan, validating the use of the robot for the application employing impedance control functionalities, and sensing data to create a closed-loop system.

The results obtained are within the validation criteria, creating conditions to affirm, for the application in question, that the gravity compensation was achieved with success using the robot.

Avaliação da aplicabilidade de braço robótico para anular ação gravítica em testes de maquinismos

Resumo

A exigência de testar equipamentos aeroespaciais em Terra antes de serem lançados para o espaço leva à necessidade de simular condições de órbita, nomeadamente condições de gravidade zero. Existe uma vasta gama de equipamentos passivos para realizar a compensação da gravidade, mas com limitações normalmente associadas à complexidade de trajetórias. Os sistemas ativos foram desenvolvidos para mitigar as dificuldades dos passivos.

O aparecimento de braços robóticos com controlo de força integrado, movimentos de alta precisão e trajetórias que podem ser complexas e facilmente reprogramáveis abre as portas para a utilização e desenvolvimento de soluções de compensação de gravidade ativas. Dito isto, o objetivo desta dissertação foi testar e validar a utilização de um robô Kuka IIWA 14, na realização de compensação de gravidade ativa em testes terrestres. Começou com a definição de um estudo de caso onde seria possível avaliar o desempenho do robô usando instrumentação externa e integrada do robô. Para interligar tudo e possibilitar a troca e aquisição de dados, foi montado um conjunto de terminais Beckhoff para controlar os componentes do estudo de caso e fazer a ponte de comunicação entre o controlador do robô e um computador que corre uma plataforma IoT, baseada em Node-RED.

Diferentes estratégias de controlo foram desenvolvidas e validadas definindo um plano de teste, autenticando o uso do robô para a aplicação usando funcionalidades de controlo por impedância e informações de sensorização para criar um sistema em malha fechada.

Os resultados obtidos estão dentro dos critérios de validação, criando assim condições para afirmar que, para a aplicação em questão, a compensação da gravidade foi alcançada com sucesso utilizando o robô.

Agradecimentos

A conclusão desta dissertação de mestrado marca o fim de um desafiante percurso académico de cinco anos. Este caminho nunca poderia ser realizado sem o apoio fundamental de amigos e familiares. Por isso agradeço:

Em primeiro lugar, aos meus pais. É impossível transmitir por palavras tudo o que fizeram, fazem e certamente farão por mim. Graças a eles e ao esforço que durante estes anos aplicaram, houve condições para que tudo isto fosse possível.

A meu Professor António Mendes Lopes, pela partilha de conhecimento e pela honra de ter realizado a presente dissertação com a sua supervisão.

Ao meu orientador Eng. Luís Carlos Moreira, por me ter orientado de forma notável, sempre atento ao andamento do projeto, criando condições para o trabalho desenvolvido sempre num ambiente de motivação e incentivando-me a dar, todos os dias, o meu melhor. A ele, o meu sincero obrigado e apreço.

Ao meu orientador Eng. Ricardo Lopes, pela forma espetacular como me orientou e encarou o projeto. A definição e rigor do trabalho conduzido não poderiam ser os mesmos sem o toque que o Eng. Ricardo impregnou. São muitos os ensinamentos que levo para a minha vida pessoal e profissional provenientes do seu contacto, por isso, o meu profundo agradecimento.

Aos colaboradores do INEGI que sempre demonstraram disponibilidade para ajudar nas diferentes fases da dissertação e pelo ambiente de entre ajuda criado.

À minha família, à Beatriz, aos meus avós, e aos meus amigos por acreditarem em mim.

Muito Obrigado!

Pedro Jorge Lopes

Julho 2022

Table of Contents

1. Introduction.....	1
1.1. Project framework and motivation.....	1
1.2. Integration of the project in the panorama of INEGI.....	1
1.3. Objectives of the project	1
1.4. Method followed in the project.....	2
1.5. Document structure	2
2. State of The Art.....	3
2.1. Aerospace Functional Testing.....	3
2.2. Off-loading systems	4
2.2.1. Passive Mechanisms	5
2.2.2. Active Mechanisms	7
2.3. Robotics and Aerospace.....	9
2.3.1. Robots applications in Aerospace	9
2.3.2. Robot Ground Testing	10
2.4. Robotics	11
3. System requirements and breadboard development	12
3.1. System requirements	12
3.2. Breadboard overview	13
3.3. Hinge Prototype	13
3.3.1. Structural simulation	15
3.4. Motor.....	17
3.4.1. Validate Beckhoff AS1050-0120	17
3.5. Torque Sensor	19
3.5.1. Validate FUTEK TFF400.....	19
4. Hardware Configuration	22
4.1. Collaborative Robot Kuka LBR IIWA	23
4.1.1. Kuka Sunrise Cabinet.....	24
4.2 Beckhoff Coupler.....	25
4.2.1. Beckhoff Ek1100.....	25
4.2.2. Beckhoff EL7041	25
4.2.3. Beckhoff EL3008	26
4.2.4. KRC4 Kuka EL6695-1001	27
4.2.5. Final Setup Considerations.....	28
5. Control Development.....	29
5.1. Sunrise Workbench/Workvisual I/O Mapping	29
5.1.1. Stepper Motor Configuration;	30
5.1.2. Torque Sensor Configuration;	31
5.1.3. Kuka / TwinCAT Interface;.....	31
5.2. TwinCAT Interface	33
5.2.1. Main Program.....	34
5.3. Robot Background Tasks	35
5.3.1. Motor Control Background Task.....	35
5.3.2. Variables Background Task.....	36
5.4. IoT platform – Node-RED	36
5.4.1. Dashboard Development;	37
6. System Characterization	39
6.1. Beckhoff Stepper Motor Velocity.....	39
6.2. Torque Sensor Calibration	39

6.3. Torque Sensor Gradients	40
6.3.1. Torque gradient calculation	40
6.3.2. Mass gradient calculation.....	41
6.4. Beam Angle Gradients	42
7. Robot Control Development	44
7.1. Programming a Compliant Robot.....	44
7.2. Programming Modes	45
7.2.1. Robot Impedance Control (RIC).....	45
7.2.2. Robot Impedance Control plus Torque Sensor (RIC_TS).....	46
7.2.3. Robot Impedance Control plus Torque Sensor plus Encoder (RIC_TS_E).....	47
7.2.4. Robot Impedance Control plus Torque Sensor plus Robot Forces (RIC_TS_RF)	48
7.2.5. Programming global considerations.....	48
8. Experiments.....	50
8.1. Test Plan	50
8.1.1. Setup 0:	50
8.1.2. Setup A:.....	50
8.1.3. Setup B:.....	51
8.1.4. Setup C:.....	51
8.2. Setup 0 – Results	52
8.3. Setup A - Results	53
8.4. Setup B – Results.....	55
8.5. Setup C – Results.....	56
8.6. Setup 0 results discussion.....	57
8.7. Setup A results discussion.....	57
8.7.1. Robot Impedance Control	57
8.7.2. Robot Impedance Control plus Torque Sensor	57
8.7.3. Robot Impedance Control plus Torque Sensor plus Encoder	59
8.7.4. Robot Impedance Control plus Torque Sensor plus Robot Forces.....	60
8.8. Setup B results discussion	61
8.9. Setup C results discussion	63
8.10. Offloading global Discussion.....	64
8.11. Results Discussion – Accuracy and Uncertainties	65
8.11.1. Hinge rigidity issue	65
8.11.2. Instrumentation Accuracy	65
8.11.3. Theoretical/Experimental Accuracy evaluation.....	65
8.12. Additional Test	66
9. Conclusions and Future works	68
References.....	71
Appendix A – Beckhoff AS1050-0120 datasheet.....	75
Appendix B – Futek TFF400 datasheet	76
Appendix C – TwinCAT Main Program.....	77
Appendix D – Sunrise Workbench - Motor Control Background Task	78
Appendix E – Sunrise Workbench - Variables Background Task.....	80
Appendix F – Node-RED Programming tabs	81
Appendix G – Node-RED Dashboard tabs	82
Appendix H – Sunrise Workbench - RIC mode	84
Appendix I – Sunrise Workbench – RIC_TS – XZ plane.....	86
Appendix J – Sunrise Workbench – RIC_TS – XYZ plane	88
Appendix K – Sunrise Workbench – RIC_TS_E.....	90

Appendix L – Sunrise Workbench – RIC_TS_RF – XZ plane	92
Appendix M – Sunrise Workbench – RIC_TS_RF – XYZ plane	94
Appendix N – Offloading results – No offloading 2.7N, 3.2N	97

Table of Figures

Figure 2.1 - Thermo-Elastic Distortion test setup [2].	3
Figure 2.2 - Thermo-vacuum chamber setup [3].	4
Figure 2.3 - LEA-L12 project [4].	4
Figure 2.4 - Offloading concept explanation.	5
Figure 2.5 - Schema of the overlapping antenna ground offloading test [5].	5
Figure 2.6 - Schema of the three overlapping antennas [5].	6
Figure 2.7 - Pulley, cable, and counterweight offloading setup [2].	6
Figure 2.8 - Example of passive offloading setup [2].	6
Figure 2.9 - Deployment setup step 1 [2].	7
Figure 2.10 - Deployment setup step 2 [2].	7
Figure 2.11 - Active two-dimension offloading device [1].	8
Figure 2.12 - Operating principle of the AGM [7].	8
Figure 2.13 - WAGM during tests [7].	8
Figure 2.14 - ExoMars planetary robot [9].	9
Figure 2.15 - New EPOS2 facility[10].	10
Figure 2.16 - ESA GRALS Testbed [11].	10
Figure 2.17 - System architecture of the test bench [12].	10
Figure 2.18 - Pattern used to detect the x-y displacements [12].	11
Figure 3.1 - Robot behavior intent.	13
Figure 3.2 – 0 g passive offloading device working principle.	13
Figure 3.3 - Hinge and setup prototype.	14
Figure 3.4 - Breadboard hinge - principal components.	14
Figure 3.5 - Breadboard/Robot interface.	15
Figure 3.6 - Additive manufacturing fill analysis.	15
Figure 3.7 – Breadboard SOLIDWORKS simulation setup.	16
Figure 3.8 - Breadboard SOLIDWORKS simulation results.	16
Figure 3.9 - Stepper motor AS1050-0120 [16].	17
Figure 3.10 - Torque/Velocity stepper motor curve (Appendix1).	18
Figure 3.11 - Forces applied to the beam.	18
Figure 3.12 – Torque sensor Futek TFF400 [17].	19
Figure 3.13 - Breadboard static diagram.	20
Figure 3.14 - Breadboard static diagram, detailed.	21
Figure 3.15 - Breadboard gears angle measurement.	21
Figure 4.1 - Schema of the hardware connections.	23
Figure 4.2 - Kuka LBR IIWA 14 R820.	23
Figure 4.3 - Kuka LBR IIWA 14 workspace.	24
Figure 4.4 - Beckhoff Ek1100 [18].	25
Figure 4.5 - Beckhoff EL7041 [19].	25
Figure 4.6 - Phoenix Contact 48 V 20 A [20].	26
Figure 4.7 - Futek IAA100 [21].	26
Figure 4.8 - Beckhoff EL3008 [22].	27
Figure 4.9 - Beckhoff EL6695-1001 [23].	27
Figure 4.10 - Beckhoff external complete setup.	28
Figure 5.1 – Kuka Sunrise Workbench work environment.	29
Figure 5.2 - Kuka Workvisual work environment.	30
Figure 5.3 - Workvisual project structure.	30
Figure 5.4 - Stepper motor I/O configuration.	31
Figure 5.5 - Torque Sensor I/O configuration.	31
Figure 5.6 - Beckhoff EL6695 variables I/O configuration.	32
Figure 5.7 - Beckhoff EL6695 input configuration.	32

Figure 5.8 - Beckhoff EL6695 output configuration.	32
Figure 5.9 - TwinCAT process data configuration.	33
Figure 5.10 - TwinCAT PLC project.	33
Figure 5.11 - TwinCAT Visualization.	34
Figure 5.12 - Kuka background task I/O mapping.....	36
Figure 5.13 - Node-RED work environment.	37
Figure 5.14 - Node-RED tablet dashboard.....	37
Figure 6.1 - Torque sensor calibration setup.....	40
Figure 6.2 - Torque sensor calibration results.....	40
Figure 6.3 - Mass gradient calculation setup.	41
Figure 6.4 - Mass gradient results.	41
Figure 6.5 - Mass gradient calculation.	42
Figure 6.6 - Beam angle/stepper motor encoder.	43
Figure 6.7 - Initial beam position.....	43
Figure 7.1 - Compliant robot in contact with the external environment [32].	44
Figure 7.2 - Impedance modes test setup.....	45
Figure 7.3 - SAUTER FK250.	46
Figure 7.4 - Schema of the forces applied to the dynamometer.	46
Figure 7.5 - Position of the robot flange referential.....	47
Figure 7.6 - Difference between the displacement calculation of the two programming modes.	48
Figure 8.1 - Setup A test configuration.....	51
Figure 8.2 - Setup B test configuration.	51
Figure 8.3 - Setup C test configuration.	52
Figure 8.4 - Torque setup 0, no offloading.	52
Figure 8.5 - Torque setup A, robot impedance control.	53
Figure 8.6 - Torque setup A, robot impedance control plus torque sensor.....	53
Figure 8.7 - Torque setup A, robot impedance control plus torque sensor plus encoder.	54
Figure 8.8 - Torque setup A, robot impedance control plus torque sensor plus robot forces..	54
Figure 8.9 - Torque setup B, robot impedance control plus torque sensor.	55
Figure 8.10 - Torque setup B, robot impedance control plus torque sensor plus robot forces.	55
Figure 8.11 - Torque sensor C, robot impedance control plus torque sensor.	56
Figure 8.12 - Torque setup C, robot impedance control plus torque sensor plus robot forces.	56
Figure 8.13 - Force Z setup A, RIC / RIC_TS.....	58
Figure 8.14 - Application point/center of mass displacement on the model.....	58
Figure 8.15 – RIC_TS mode cable inclination, a) deployment beginning, b) deployment end.	59
Figure 8.16 - Robot flange external forces schema.	59
Figure 8.17 - – RIC_TS_E mode cable inclination, a) deployment beginning, b) deployment end.....	60
Figure 8.18 – Setup A, forces X analysis.....	61
Figure 8.19 - Setup B, forces X analysis.....	62
Figure 8.20 - Setup B, forces Y analysis.....	62
Figure 8.21 - Setup C, forces X analysis.....	63
Figure 8.22 - Setup C, forces Y analysis.....	64
Figure 8.23 - Hinge rigidity analysis.	65
Figure 8.24 - Offloading validation, stop and go test.	67

Table of Tables

Table 3.1 - Requirement of the offloading device.	12
Table 3.2 – Functional and operational requirements.	12
Table 3.3 - Stepper motor AS1050-0120 characteristics.	17
Table 3.4 - Torque sensor Futek TFF400 characteristics.	19
Table 4.1 - Kuka LBR IIWA 10 R820 characteristics.	24
Table 4.2 - Kuka Sunrise Cabinet characteristics.	24
Table 5.1 - Output variables going to the robot.	34
Table 5.2 - Input variables from the robot.	35
Table 5.3 - Output variables going to the Node-RED.	35
Table 5.4 - Input variables from the TwinCAT.	36
Table 6.1 - Mass gradient results.	41
Table 7.1 - Workobject weight, robot programming.	48
Table 8.1 - Test plan matrix.	50
Table 8.2 - Setup 0 torque results.	57
Table 8.3 - Setup A torque results, RIC.	57
Table 8.4 - Setup A torque results, RIC_TS.	57
Table 8.5 - Setup A torque results, RIC_TS_E.	60
Table 8.6 - Setup A torque results, RIC_TS_RF.	60
Table 8.7 - Setup A forces X results – RIC_TS_RF / RIC_TS.	61
Table 8.8 - Setup B torque results, RIC_TS_RF / RIC_TS.	61
Table 8.9 – Setup B force's standard deviation, RIC_TS_RF.	62
Table 8.10 - Setup C torque results, RIC_TS_RF / RIC_TS.	63
Table 8.11 - Test plan, global results.	64
Table 8.12 - Theoretical/Experimental relative error.	66
Table 8.13 - Stop and go torque results.	67
Table 9.1 - Offloading, functional, and operational requirements analysis.	69

Acronyms

ABS	Acrylonitrile Butadiene Styrene
CAD	Computer-aided design
ESA	European Space Agency
HMI	Human Machine Interface
IIWA	Intelligent Industrial Work Assistance
I/O	Input/Output
IoT	Internet of Thing
LBR	“Leichtbauroboter”, German for lightweight robot
PLC	Programmable Logic Controller
RIC	Robot Impedance Control mode
RIC_TS	Robot Impedance Control plus Torque Sensor mode
RIC_TS_E	Robot Impedance Control plus Torque Sensor plus Encoder mode
RIC_TS_RF	Robot Impedance Control plus Torque Sensor plus Robot Forces mode
RO	Rated Output
TwinCAT	The Windows Control and Automation Technology
XAR	eXtended Automation Runtime

1. Introduction

1.1. Project framework and motivation

In the development of aerospace application technologies, the need to test components, equipment, and mechanisms on Earth, presents important challenges regarding the simulation of operational conditions in orbit.

One important challenge to overcome in aerospace ground testing is the compensation for the gravity effects, also known as zero-G offloading. One of the most critical conditions to simulate during testing is the absence of gravity.

There are numerous passive systems based on counterweight solutions, pneumatic actuation, balloon suspension, and some active solutions that use force control counter actuation. Passive solutions, on the other hand, are distinguished by the difficulty of performing gravitational counteracting along complex trajectories and are typically limited to elementary trajectories and systems. To address the shortcomings of passive zero gravity systems, active zero gravity systems have been developed.

The appearance of robotic arms, with integrated force control, high-precision movements, and whose trajectories could be complex and easily reprogrammable, opens the possibility of applying and creating solutions, of gravity compensation.

The present master thesis, developed in INEGI, fits in the development of a new approach to offloading using a Kuka collaborative robot, Kuka LBR IIWA 14 performing force control.

1.2. Integration of the project in the panorama of INEGI

INEGI, Institute of Science and Innovation in Mechanical Engineering and Industrial Engineering, is a Research and Technology Organisation (RTO) dedicated to carrying out activities of research and innovation, the development of new technologies, and industry-oriented technologies. This institute, in operation since 1986, was created within the current Department of Mechanical Engineering (DEMec) and Industrial Engineering and Management (DEGI) of the Faculty of Engineering of the University of Porto (FEUP) to contribute to the development of the Portuguese industry, ensuring the enrichment of the academic community.

This master thesis encompasses many areas of research in which INEGI is highly interested such as aerospace, robotics, instrumentation and control systems, additive manufacturing, and IoT development. Pursuing the opportunity for INEGI to widen its scope on offloading device development for the aerospace industry, the project built a baseline of knowledge for potential proposals to ESA's R&D calls on new active offloading device.

1.3. Objectives of the project

The aim of this dissertation is to develop an application that allows characterizing and validating the use of a Kuka IIWA collaborative robot, in gravity compensation for the testing and validation of aerospace systems. Objectively, the project is expected to output an

assessment on the possibility to use the robot as an offloading device, alternatively to the systems that INEGI has been developing and using in multiple aerospace projects.

1.4. Method followed in the project

In a preliminary phase of the master thesis, the work performed was based on researching the state of the art of technologies behind gravity offloading systems to better understand the challenges.

After a trial period of developing the robot control strategy, a use case was created, (hereby designated “breadboard”), where the robot's capabilities were tested and characterized.

The breadboard combined control, data visualization, and acquisition demands. As a result, a method of connecting the robot control and a user's computer was investigated. Because the robot controller incorporates a Beckhoff solution that is easily compatible with a computer's Runtime, that was the path taken to build the communication bridge. The data visualization and acquisition were created using an IoT platform, Node-RED, which allowed the use of an external device (tablet) to analyze the data provided by the robot controller in real-time.

To finalize the setup, the robot programs were developed and preliminarily tested to achieve optimized solutions and create different approaches.

Following the definition of the primary programming modes, a test plan was created, taking into account the requirements that the robot had to meet. The testing results were collected, analyzed, and discussed.

1.5. Document structure

Based on the methodology applied in this project, the present dissertation is divided into ten chapters. In the **State of the Art**, analysis of the aerospace functional tests is made, introducing the concept of offloading and showing the different existing approaches. Robotics in aerospace is explored given the rising of the current technology applications. The chapter **System requirements and breadboard development**, present the offloading and global functional requirements exposition. The development of the test setup of the master thesis, breadboard, is presented here. The applicability of the components, as well as the overall structural design and requirements, are also performed in this chapter. **Hardware Configuration** is responsible for the explanation of the overall hardware configuration, robot controller, breadboard components, and user computer. In the chapter **Control Development**, the software control and communication between the hardware are detailed. It is presented the background software responsible for the breadboard control and data acquisition, englobing the robot, computer, and IoT platform description. The **System Characterization** section is responsible for the demonstration of some constants and gradients used in the control development chapter that defines all systems. The challenge of programming a compliant robot is shown in **Robot Control Development** as well as the explanation of the robot modes used. In **Experiments**, an analysis is made on a set of data from tests performed on the various programming modes of the robot. The results' accuracy is also evaluated, building some basis of reliability on them. The overall conclusions of the work performed are presented in **Conclusions and Future works**, validating the solution. Future works are also stated to further investigation on the subject.

2. State of The Art

2.1. Aerospace Functional Testing

One critical step in the development of aerospace systems and mechanisms is the need to test them on Earth with the same in-orbit environment conditions.

Before the launch of a spacecraft into orbit, it is necessary to test components and mechanisms in an artificial non-gravity environment [1]. Testing this equipment is critical to the mission's success because a single error or missed calculation results in massive economic losses.

Ground testing for aerospace systems is one of the investigation areas on INEGI being an innovation partner for the national and international organizations of aeronautics, space, and defense.

INEGI has a long heritage in testing ground support equipment, such as:

- **Thermo-Elastic Distortion (TED) test;**

This test aims to simulate solar exposure while in-orbit operation. The fact that the hardware is in the space environment, makes solar exposure an extremely aggressive thermal load introducer during the day orbit part.

The main goal is to perceive the displacement effects in the hardware caused by the induced temperature gradients and determine how much unilateral solar radiation will cause the assembly to bend due to the resulting different thermal expansion on opposite sides [2]. One way to characterize the thermo-elastic distortion of an assembly is to place several light emitters along with the testing component.

The test performed on INEGI's facilities was made in the Large Deployable Antenna (LDA) project and the testing procedure consists in deploying the arm and, as said before, adding infrared light emitters along with it. Figure 2.1 shows the setup used.

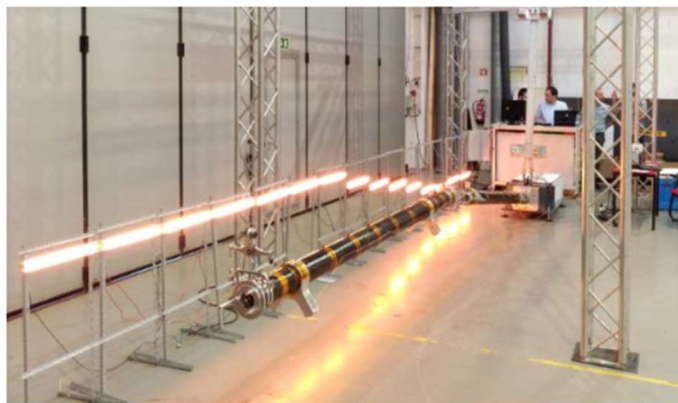


Figure 2.1 - Thermo-Elastic Distortion test setup [2].

- **Thermo-Vacuum Chamber (TVAC) test;**

To simulate the working conditions of the mechanism in the space environment, a special thermo-vacuum chamber enables testing with offloading systems [3]. In this case, INEGI participates in the development of the offloading device.

Figure 2.2 represents the thermo-vacuum chamber setup used on the test.

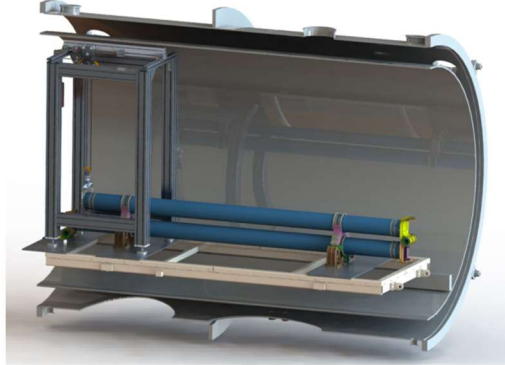


Figure 2.2 - Thermo-vacuum chamber setup [3].

- **Accuracy, repeatability, and Full Deployment test;**

An especially important aspect is the assessment of the deployment capability of aerospace equipment to ensure that everything works (motors, hinge mechanisms, etc) before the launch.

Precision and repeatability, on the other hand, must be evaluated to understand how the structure deployment is accomplished and whether possible systematic errors can be adjusted on Earth before the launch or even if small errors can be remotely solved in the space area.

An example of this point of view is the deployment of the main beam in the LEA family [4]. These types of equipment need to have excellent deployment as well as good repeatability to ensure that the reflective area on top (Figure 2.3) is perfectly aligned to obtain the right reflection angle on the antenna base.



Figure 2.3 - LEA-L12 project [4].

2.2. Off-loading systems

The condition of zero gravity is one of the most important features of the space environment. Due to the weightlessness of the equipment in space, internal components have a different state of stress between the ground testing and the space environment [5].

There are multiple antennas, satellites, and spaceships equipped with moving parts that are adjusted following the mission or need to be unfoldable after reaching the targeted space area. Due to the effect of gravity on the ground, the result of the tests are influenced and can even damage the mechanisms that are being tested.

To show the importance of offloading a structure, a small and schematic example is illustrated in Figure 2.4. Suppose there is a large beam, 6 m in length, with negligible mass, a concentrated mass of 1 kg on one end, and recessed on the other (Figure 2.4). Two sceneries are analyzed: one with offloading and the other with no offloading.

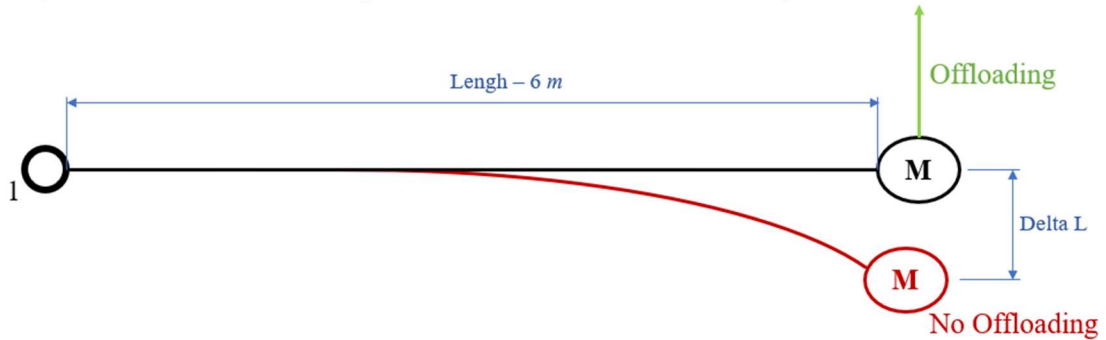


Figure 2.4 - Offloading concept explanation.

Notice that if the beam has no offloading, the static torque on 1 (Figure 2.4) is:

$$M = 6 * 9.81 * 1 = 58.8 Nm \quad (2.1)$$

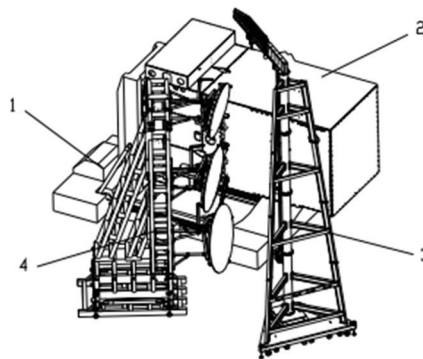
On the other hand, if offloading is applied, the result of the torque is equal to zero because the weight force of the mass is balanced by the offloading device. Comparing the value with the one reached in Equation 2.1, is clear the difference in stress at the recessed area.

The displacement is also a factor to take into consideration [6]. When an offloading device is used, the displacement at the free end of the beam is zero and independent of the material used. Without offloading, tests will be highly dependent on the stiffness of the material used, resulting in unexpected displacements, as the example of “Delta L” in Figure 2.4. Taking all into account, several offloading mechanisms can be used to balance the gravitational force on space hardware mechanisms.

2.2.1. Passive Mechanisms

A passive gravity compensation mechanism is a setup capable of absorbing the gravity force but is normally associated with simple trajectories [5].

Figure 2.5 shows how these passive mechanisms work in an unfolding process of three-overlapping antennas [5]. The schematic diagram of the three-overlapping antennas, Figure 2.6 gives a concept of how the deployment is made.



1-Two-axis turntable; 2-satellite; 3-antenna unloading system; 4-antenna fixing device

Figure 2.5 - Schema of the overlapping antenna ground offloading test [5].

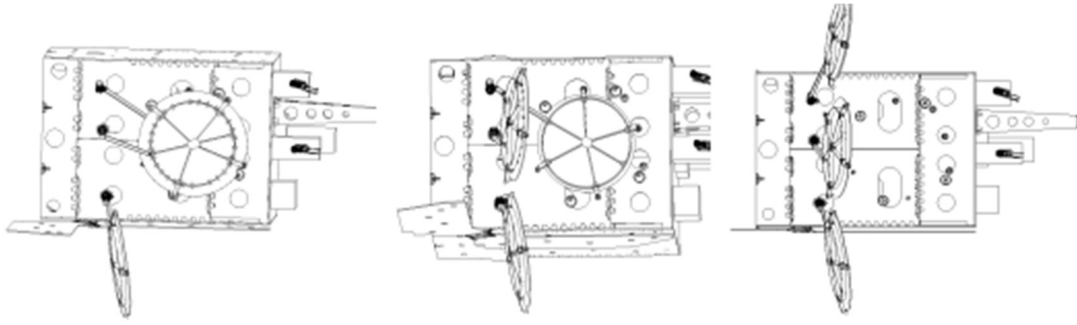


Figure 2.6 - Schema of the three overlapping antennas [5].

This application uses pulleys, cables, and counterweights, associated with a rotational beam. The trajectory is unique for each antenna and describes simple paths.

An example of a pulley, cable, and counterweight offloading setup is shown in Figure 2.7.

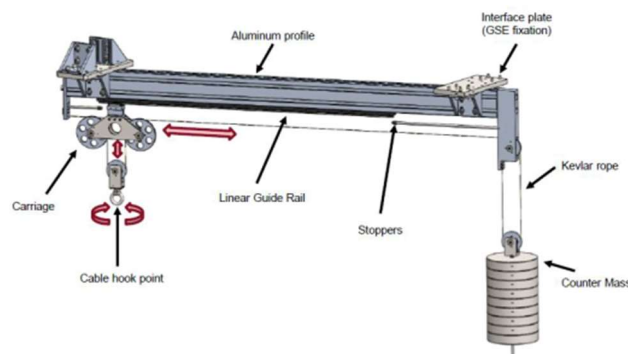


Figure 2.7 - Pulley, cable, and counterweight offloading setup [2].

One example of a passive offloading device that manages to make a more difficult trajectory was developed by INEGI and used to allow a stable deployment for a large arm. Figure 2.8 is a representation of the setup.

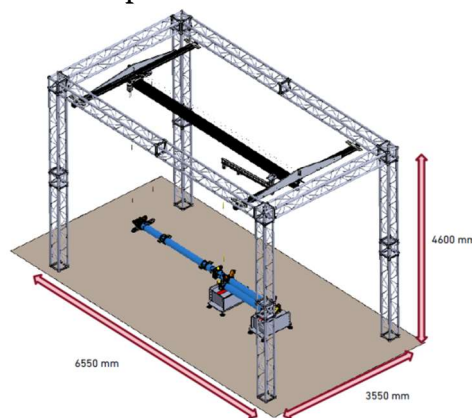


Figure 2.8 - Example of passive offloading setup [2].

The system uses a set of pulleys, cables, and counterweights that follows the movement and keep the system offloaded. In this case, the deployment is made in two steps:

- Step 1 – Deployment using the main beam with a linear rail using a simple configuration and setup (Figure 2.9);
- Step 2 – Deployment using the rotational beam with a complex and harder trajectory (Figure 2.10);

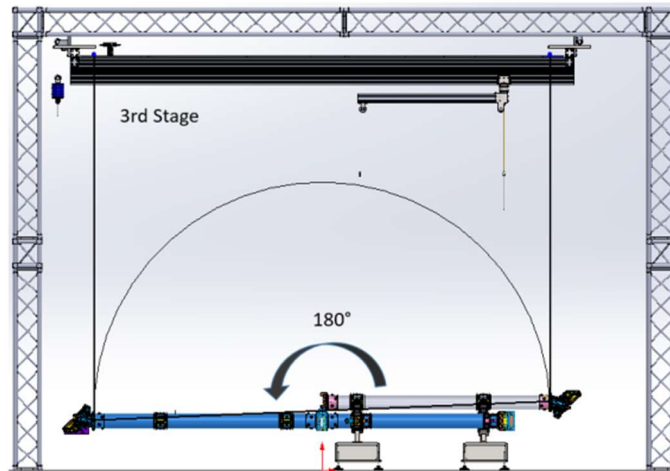


Figure 2.9 - Deployment setup step 1 [2].

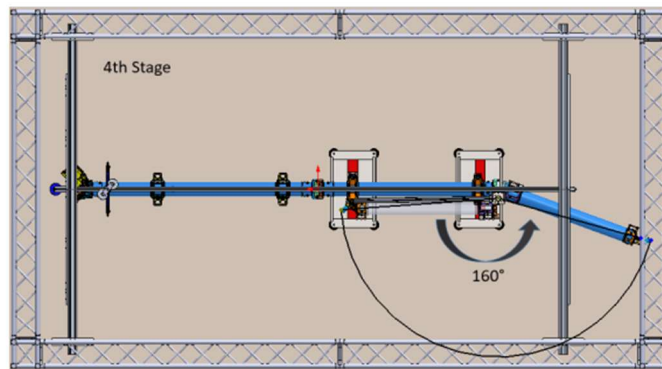


Figure 2.10 - Deployment setup step 2 [2].

Passive approaches, on the other hand, are subject to the unavoidable effects of added inertia or friction forces. The changing form factor of the deployable structure itself and the need to track the trajectory of the center of gravity also pose a challenge to these conventional passive technologies [7].

There are other ways to passively compensate for gravity like using air balloons and water floats, but they represent problems when came to useful mass and viscosity of the fluid, respectively.

2.2.2. Active Mechanisms

Active offloading mechanisms are used to solve the problem of passive mechanisms, specifically complex trajectory problems with three degrees of freedom. Multidirectional framed structures are used with the capability to track and real-time analyze the deployment of the structure [7].

Figure 2.11 represents the structure diagram of a system that uses active offloading compensation for the deployment in two-dimension of a space antenna [1].

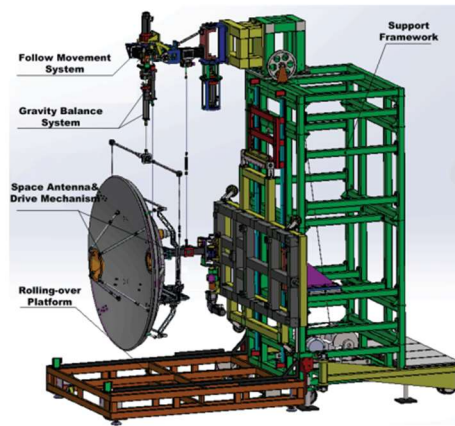


Figure 2.11 - Active two-dimension offloading device [1].

In this case, a set of electric servo cylinders and force sensors provide the servo control for the deployment, creating a constant tension suspension control strategy based on a closed-loop system [1].

The Anti-Gravity Machine (AGM) is also an example of an active offloading device. The AGM is essentially a pressure-regulated air spring. It is shown schematically in Figure 2.12.

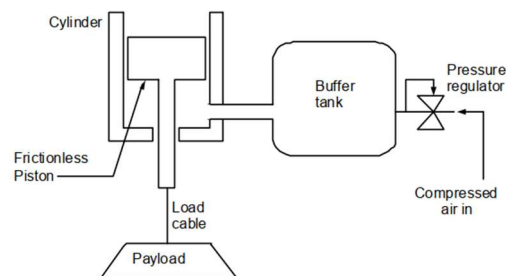


Figure 2.12 - Operating principle of the AGM [7].

The load is supported by the force developed by an air piston that moves vertically in a closely fitted cylinder. The piston is of a special type that combines the functions of a piston and an air bearing.

This mechanism is the original concept for the elaboration of a more complex machine, the Walking Anti-Gravity Machine (Figure 2.13), which is, basically, an AGM machine but with the capability to move on the X-Y plane using bent drives and stepper motors to keep the load cable vertical during active tracking [7].

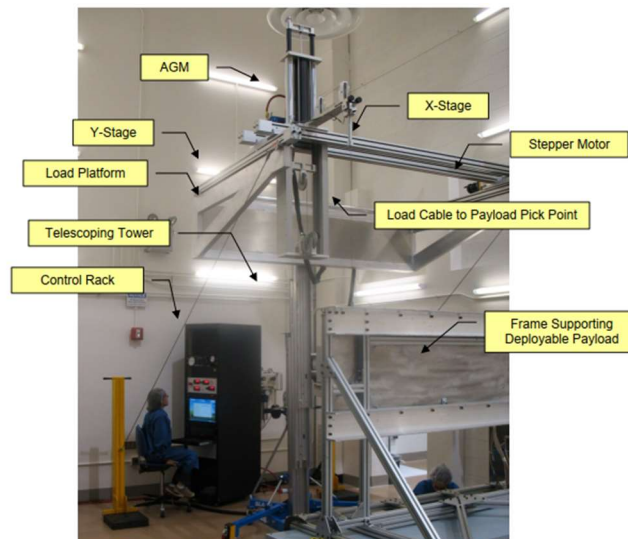


Figure 2.13 - WAGM during tests [7].

The scope of the present master thesis is the development of a new approach to offloading for the elaboration of these types of tests on Earth.

2.3. Robotics and Aerospace

2.3.1. Robots applications in Aerospace

With technological innovation, robotics became a key technology in the industry, showing features never seen before with huge capabilities. This evolution and the mission requirements changes led to the evolution of the utilization of robotics in the aerospace world.

Several space robotic applications already exist such as [8]:

- **Orbital Robotics: In-orbit Servicing Manufacturing and Assembly**

Orbital Robots are made to be used in the absence of gravity and atmosphere. In this particular case, three development grants were concluded and demonstrated in-ground applications of:

- Basic in-orbit services through the European Robotics Support Services (EROSS) grant;
- Assembly in space of very large structures through the Prototype of an Ultra Large Structure Assembly Robot (PULSAR) grant;
- Reconfiguration of modular satellites with the MODular Spacecraft Assembly and Reconfiguration (MOSAR) grant;

- **Planetary Robotics**

Planetary robots differ from orbital robots in that they operate in conditions of gravitational force. The ExoMars (Figure 2.14) will be launched in September of 2022 to Mars and is an example of robotics in aerospace working with the presence of gravity.

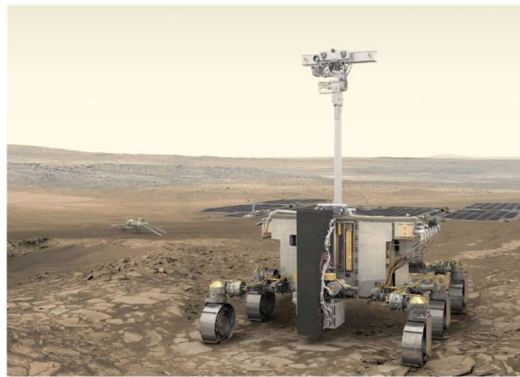


Figure 2.14 - ExoMars planetary robot [9].

Notice that all the robots working on Earth are also planetary. Developers need to take into account the presence of gravity to develop their equipment.

2.3.2. Robot Ground Testing

There is a large number of space robotic facilities around Europe to support the need for testing aerospace equipment on Earth. The next examples show some technologies implemented and used [8]:

- The German Aerospace Center (DLR) department allows simulation of proximity operations using two robots of 100 kg payload mounted on a 25 m track each, Figure 2.15.

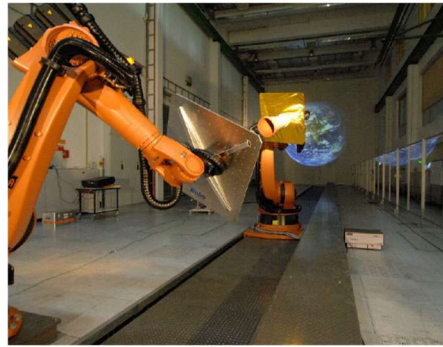


Figure 2.15 - New EPOS2 facility[10].

- At ESA European Space Research and Technology Centre tested rendezvous operations using 10 kg class robots mounted on long tracks as Figure 2.16 suggests.



Figure 2.16 - ESA GRALS Testbed [11].

- Airbus Defense & Space has financed and supervised an experiment using a sensing system based on an industrial robot promoting offloading using a camera setup and a collaborative robot to provide a constant force of displacement, to simulate the absence of gravity [12]. Figure 2.17 shows the setup of the solution.

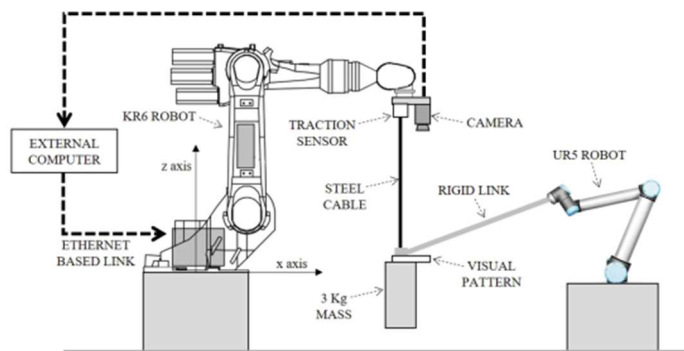


Figure 2.17 - System architecture of the test bench [12].

The basic idea is to detect displacements using a visual pattern and the camera (Figure 2.18) to provide information for the industrial robot to move to a specific position.

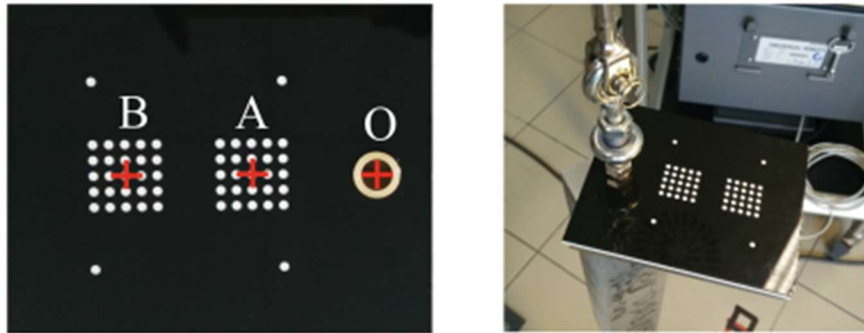


Figure 2.18 - Pattern used to detect the x-y displacements [12].

This system presents a new approach to offloading testing. This will not be the path followed but it is an interesting approach to the problem, using visual technology instead of force control.

Concluding, the examples show that robotics is involved in the aerospace world and is becoming more and more a key technology to invest with high capabilities and new features.

2.4. Robotics

There are several applications for anthropomorphic robots, creating a window of opportunity for their use in new applications. That being said, the usage of this type of robot could be interesting for aerospace testing, taking advantage of [13]:

- Multiple joints create a workspace where it is possible to position the end terminal anywhere;
- Precise movements capable of moving it with minimum error thanks to the kinematics of the robot;
- Modes of positioning using linear, joint, and circular trajectories.

Because the purpose of the work is to characterize the use of a collaborative robot on an application, it is critical to understand how they work and their main characteristics [14] [15]:

- Quick tool/programming change – Can be used on low batch production with high flexibility to change tools and applications;
- Safety Design – Collaborative came equipped with transducers and sensors that detect collisions making them possible to use close to operators;
- Low load capacity – Compared with industrial robots, cobots present low load capacity.
- New programming modes – It is possible to use new operational modes like impedance control thanks to the inbuild instrumentation.

3. System requirements and breadboard development

3.1. System requirements

In the present master thesis, some requirements are targeted to be accomplished, apart from the desired robot programming. Table 3.1 shows the offloading device requirements that need to be accomplished and Table 3.2 sums up the additional functional and operational requirements.

Table 3.1 - Requirement of the offloading device.

Offloading Requirements		Values
OFF-1	Masses	2-5 kg
OFF-2	Deployment Hinge Speed	0.02 rpm
OFF-3	Range	0-1 m
OFF-4	Parasitic Force	0.1 – 0.2 N

Table 3.2 – Functional and operational requirements.

Functional Requirements	
FUNC-1	A breadboard must be designed to test the offloading capabilities of the robot;
FUNC-2	The instrumentation used must be capable of measuring torques/forces applied on the breadboard;
FUNC-3	A data acquisition system for the instrumentation signals must be designed;
FUNC-4	Possible integration of the data acquisition to a live data visualization IoT platform;
FUNC-5	Robot/Breadboard must be controlled using an external computer/tablet;
Operational Requirements	
OPER-1	Ensure that the offloading force is performed vertically;
OPER-2	Assess the applicability of the system to complex trajectories;

Note: The OPER-1 was the first requirement thought when breadboarding the concept. The most straightforward way to assess this operational requirement during the development/testing of the solution is by visual inspection. Using a cable system to connect the breadboard to the robot is possible to see, in firsthand, the displacement of the robot. Figure 3.1 shows the robot behavior intended with the solution.

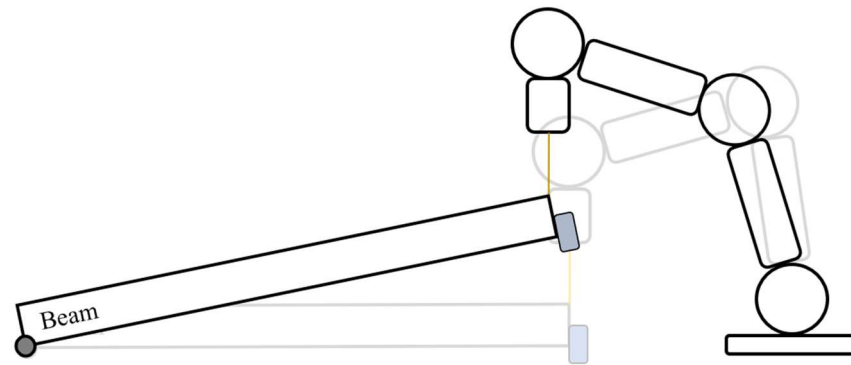


Figure 3.1 - Robot behavior intent.

3.2. Breadboard overview

Given the requirements that the robot needs to accomplish, a testing setup, further denominated by breadboard, had to be designed and manufactured, to characterize the robot's applicability and evaluate its performance in aerospace testing.

Taking this into account the state of the art review on the most common test made, emerged the decision to recreate a space hinge motor powered to simulate the deployment of a structure. This is a common mechanism tested in INEGI and it is easy to recreate for the present master thesis. Figure 3.2 shows the CAD design of a real deployable arm, which inspired the design of the breadboard.

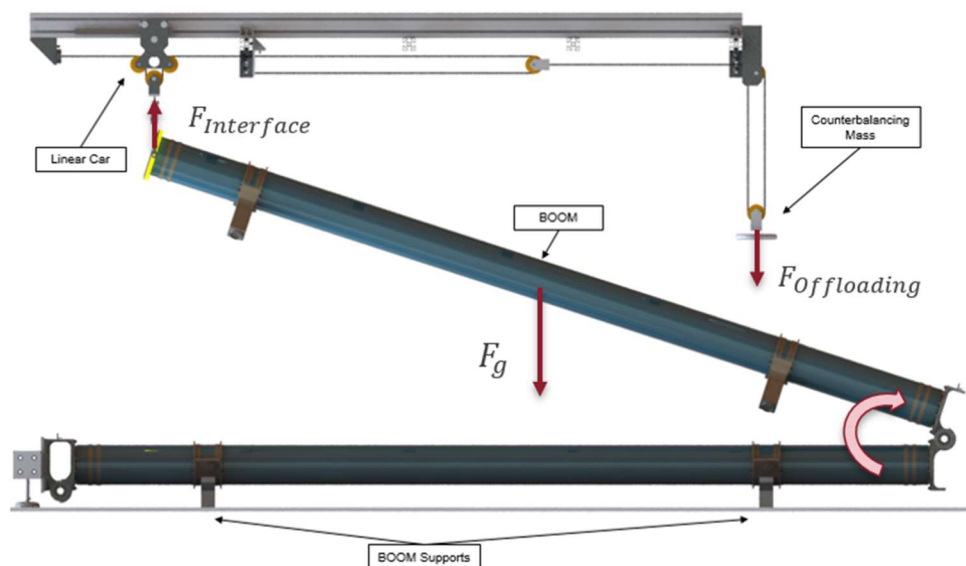


Figure 3.2 – 0 g passive offloading device working principle.

The purpose was to create a scenario capable of characterizing the robot using instrumentation, measuring, and comparing it with the standards used in aerospace testing.

3.3. Hinge Prototype

Given the overview of the hinge to be designed, some requirements were studied:

- Have an acceptable range of deployment, superior to 90° , to provide freedom of choice for the tests;
- Allow the integration of a small servomotor/stepper motor capable of deploying the structure working with offloading and with a torque sensor responsible for measuring the torque applied on the hinge;
- Integration of a set of gears between the servomotor/stepper motor and the hinge axis;

- Interface for the connection with a carbon beam;
- A system capable of adjusting the positioning of the servomotor/stepper motor for a minimal backlash on the mechanism gears;
- Interface for fixation of the hinge to metal profiles (to give a stable base for the breadboard);

The hinge and beam prototype is shown in Figure 3.3. Figure 3.4 displays the assembly of the principal components in the solution.

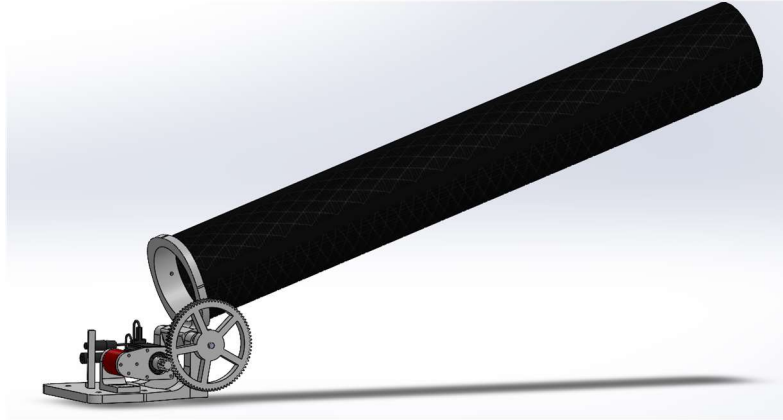


Figure 3.3 - Hinge and setup prototype.

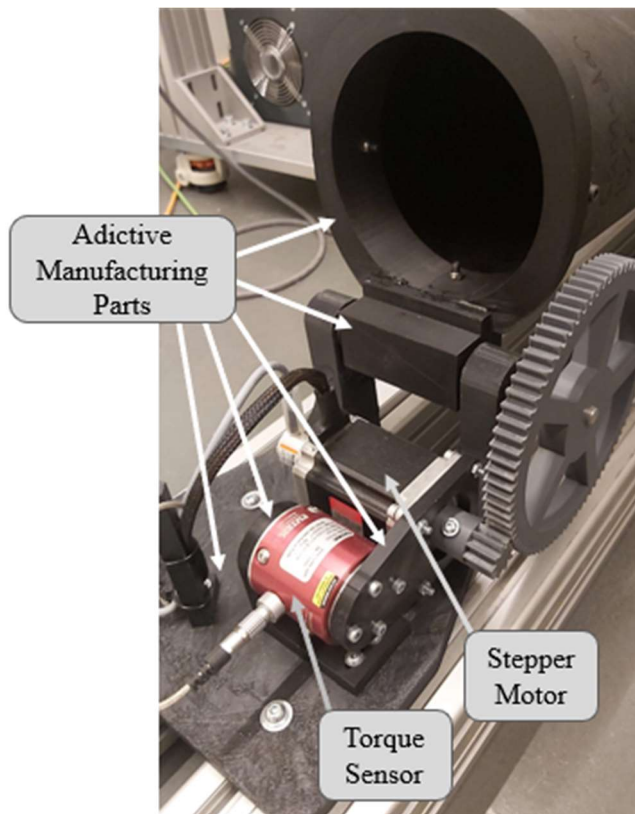


Figure 3.4 - Breadboard hinge - principal components.

Due to the utilization of a setup using a cable and the need to increment the weight applied to the mechanism, two parts were created:

- Robot flange adapter: Part, connected to the robot flange, responsible for accommodating one cable hook.
- Beam end interface (cable and additional weights): Part, connected to the end of the beam, where is possible to add, posteriorly, the additional weights and connect the other hook of the cable.

In Figure 3.5 can be observed the setup connected with two additional weights also applied.

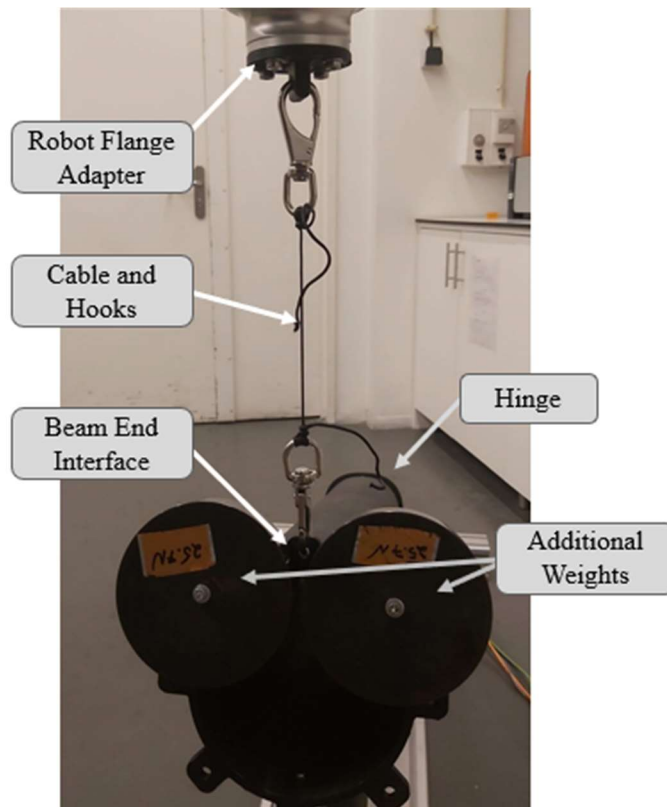


Figure 3.5 - Breadboard/Robot interface.

3.3.1. Structural simulation

With the initial design created, structural tests were made to ensure that the hinge can support the external forces applied.

One important consideration to keep in mind is that parts made by additive manufacturing do not have the same simulation behavior since they do not have a homogeneous distribution of material because the in-fill created by the manufacturing process does not have a 100% ratio, creating hollow spaces in the produced parts and removing stiffness. The simulation considered the part as a solid piece. Figure 3.6 shows a broken part where we can see the fill produced by the additive manufacturing process used. For this reason, the load on the simulation was increased.



Figure 3.6 - Additive manufacturing fill analysis.

Thinking of a hypothetic scenario, the hinge needs to support 10 kg applied on the beam end interface of the carbon tube. Static analysis was made using SOLIDWORKS simulation software with the following parameters:

- The mechanism is fixed in a hinge base (1, Figure 3.7), and supported on a tooth of the gear, (2, Figure 3.7);
- The gear, shaft, and top hinge part are connected and move together;
- 100 N were applied to the end of the carbon tube, 3;
- In the axis uniting all three components, a roller condition was applied;
- The carbon tube was considered a rigid part;
- The components material is ABS;

Notice that the simulation does not contain all the components of the mechanism. The objective was to test the critical components and where the optimization could be made.

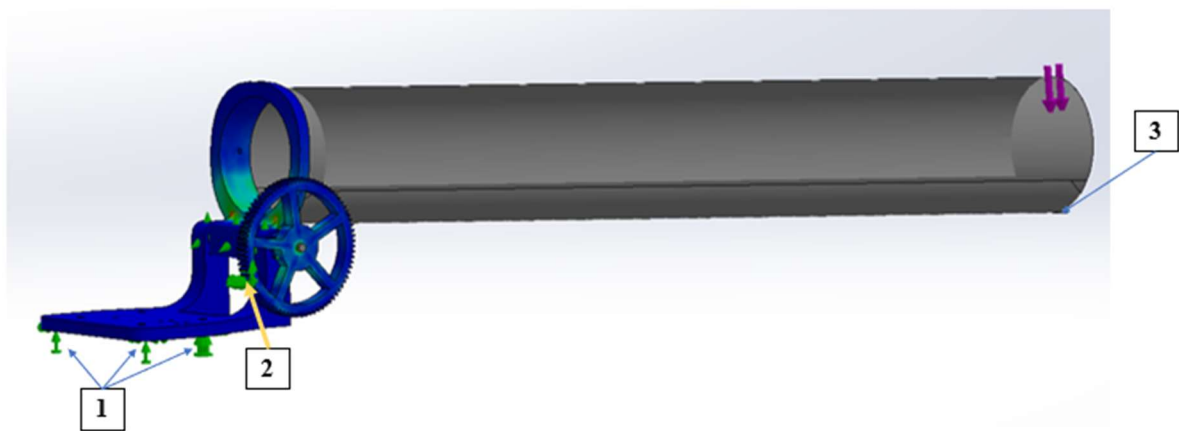


Figure 3.7 – Breadboard SOLIDWORKS simulation setup.

The final result of the simulations (Figure 3.8) shows that the mechanism will support 100 N since the stress in the gear does not exceed the supposed yielding stress of ABS.

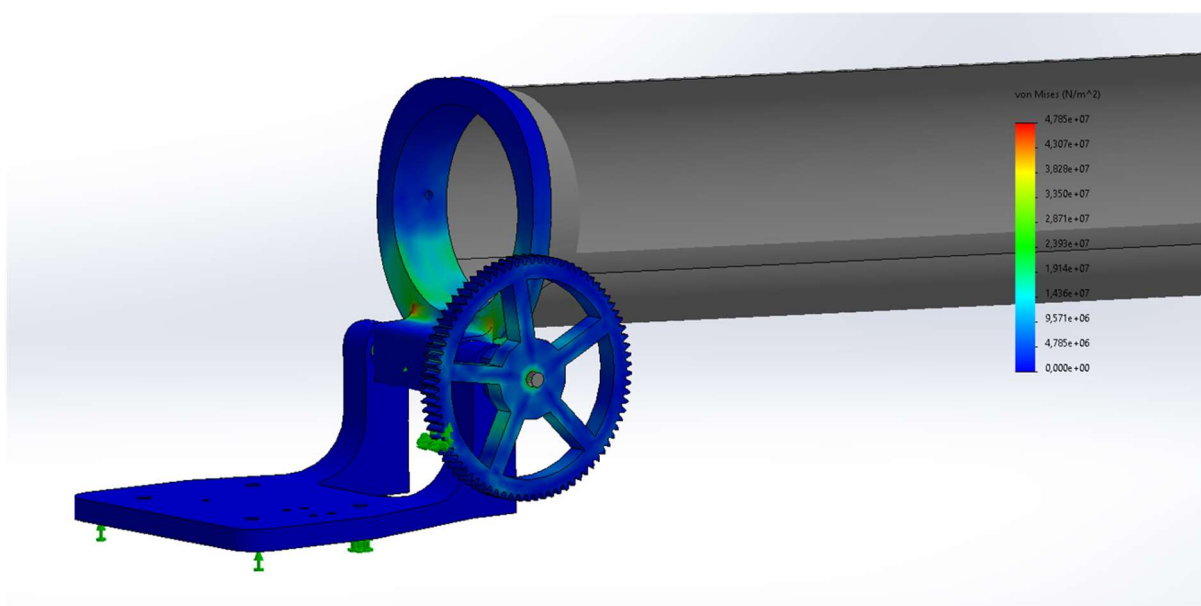


Figure 3.8 - Breadboard SOLIDWORKS simulation results.

3.4. Motor

To power the hinge, a stepper motor AS1050-0120 from Beckhoff (Figure 3.9) was selected. Table 3.3 presents the principal characteristics of the datasheet, Appendix A.



Figure 3.9 - Stepper motor AS1050-0120 [16].

Table 3.3 - Stepper motor AS1050-0120 characteristics.

Characteristics	Beckhoff AS1050-0120
Rated supply voltage	24-50 V
Rated current	5.0 A
Rated power	27.5 W
Standstill torque	1.2 Nm
Mass	1 kg

3.4.1. Validate Beckhoff AS1050-0120

To validate the utilization of the motor on the application, two principal parameters need to be evaluated:

- **Velocity:** On the application, it is required a deployment hinge speed of 0.02 rpm (value driven from the requirement OFF-2 in sub-section 3.1). A set of gears that provide a velocity reduction of 8 was designed and optimized with the best size/benefit ratio. So, the velocity required on the motor is given by Equation 3.1 and Equation 3.2.

$$RPM_{Stepper\ Motor} = RPM_{required\ in\ the\ hinge} * Reduction \quad (3.1)$$

$$RPM_{Stepper\ Motor} = 0.02 * 8 = 0.16\ rpm \quad (3.2)$$

Consulting the characteristic curve of the motor (Figure 3.10) provided on the datasheet of the supplier, it is possible to see that there are no official values for torque on the wanted velocity. Without data from the manufacturer, the motor was tested for low values of velocity and was assumed its maximum torque for 0.9 Nm.

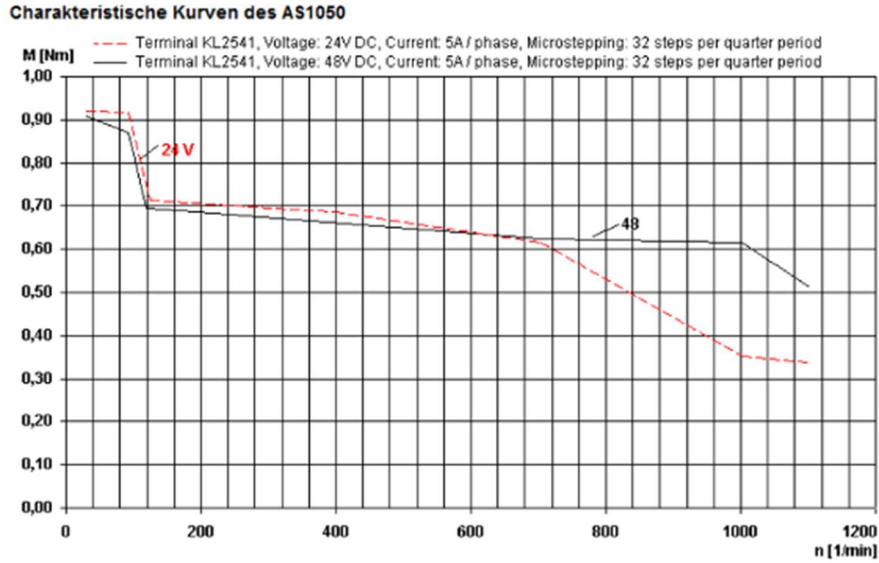


Figure 3.10 - Torque/Velocity stepper motor curve (Appendix1).

- **Torque:** As said before, it is assumed that the motor has a maximum torque of 0.9 Nm. With the gear set used, the maximum torque available is given by Equation 3.3.

$$M_{O_{max}} = 1.2 * 8 = 9.6 \text{ Nm} \quad (3.3)$$

To continue to test the motor without any offloading, the following test setup was used (Figure 3.11) A schema is also presented to show the forces applied to the beam.

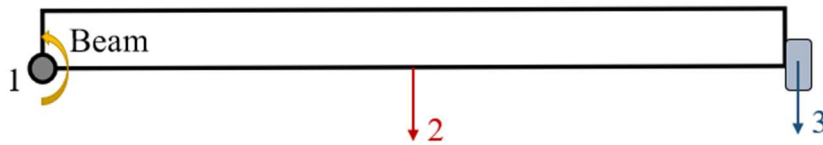


Figure 3.11 - Forces applied to the beam.

With:

- **1** is the joint of the mechanism, where the set motor/reducer will be coupled;
- **2** is the representation of the force of the concentrated mass of the composite beam.
- **3** is the force representation of the external concentrated mass applied to the beam;

With the beam fully horizontal, a static analysis was performed to validate the use of the stepper motor without offloading devices (the worst scenario case - which tests the stepper motor limit). Due to the low velocities required, the inertia of the components will not be considered.

Considering:

- Length of the beam, 1.16 m;
- Mass of the beam, 0.7 kg;
- External Mass, 10 kg;

$$M_{O_{Load \text{ Beam}+10kg}} = \left(\frac{1.16}{2} * 0.7 * 9.81 \right) + (1.16 * 10 * 9.81) = 117.78 \text{ Nm} \quad (3.4)$$

The value of the torque without the use of offloading, resulting from Equation 3.4, is too high and not allowed to be supported by the motor and gear set, which has a maximum torque of 9.6 Nm.

After this conclusion, the next step was to validate the usage of the motor to support the counter-torque produced by the beam mass only.

Equation 3.5 provides the new maximum theoretical torque exercised over the mechanism.

$$M_{o_{Load\ Beam}} = \left(\frac{1.16}{2} * 0.7 * 9.81 \right) = 3.98\ Nm \quad (3.5)$$

This new value is lower than the maximum torque allowed by the stepper motor and the set of gears. The value of 9.6 Nm validates the usage of the motor for the application.

3.5. Torque Sensor

Considering the breadboard's goal, a torque sensor was required to measure the torque produced on the hinge during the structure's deployment process. There was already a torque sensor available, and preliminary calculations revealed that it was suitable for the application.

The torque sensor is a FUTEK TFF400 (Figure 3.12) and its principal characteristics of the datasheet, Appendix B, are shown in Table 3.4.



Figure 3.12 – Torque sensor Futek TFF400 [17].

Table 3.4 - Torque sensor Futek TFF400 characteristics.

Characteristics	FUTEK TFF400
Maximum Torque Capacity	2.8246 Nm
Nonlinearity	±0.2% RO
Hysteresis	±0.2% RO
Nonrepetiability	±0.05% RO

3.5.1. Validate FUTEK TFF400

To validate the usage of the torque sensor, a static diagram was made (Figure 3.13) considering the following parameters:

- 1 – Smaller gear, connected to the stepper motor;
- 2 – Larger gear, connected to the beam axis;
- 3 – Representation of the torque sensor;

- 4 – Stepper motor and torque sensor connection part;
- D1 – Radius of the gear 1;
- D2 – Radius of the gear 2;
- D3 – Distance between the rotational center of the torque sensor and the center of the smaller gear;
- D4 – Total length of the beam;
- F1 – Resultant force from the setup stepper motor plus smaller gear;
- F2 – Resultant force from the weight of the beam;
- FN2 – Contact force of the gear 2 on the gear 1;
- $\alpha 1$ - Angle of contact between the two gears;

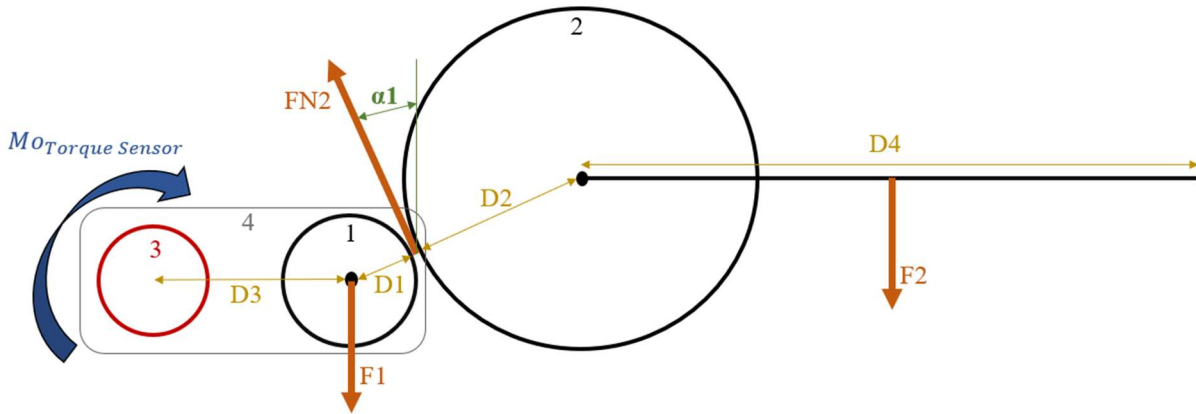


Figure 3.13 - Breadboard static diagram.

The first step was to calculate the FN2 to be able to break the mechanism and only analyze the connection in subassembly 4 (Figure 3.13). In this evaluation are used the following parameters:

- Mass of the beam = 0.7 kg;
- Mass of the stepper motor = 1 kg;
- D1 = 0.01 m;
- D2 = 0.08 m;
- D3 = 0.06 m;

Equation 3.6 calculates the FN2 value.

$$FN2 = \left(\frac{1.16}{2} * 0.7 * 9.81 \right) / 0.08 = 49.79 \text{ N} \quad (3.6)$$

Figure 3.14 is a more simplified diagram, representing the principal forces applied in subassembly 4, FN2, and F1.

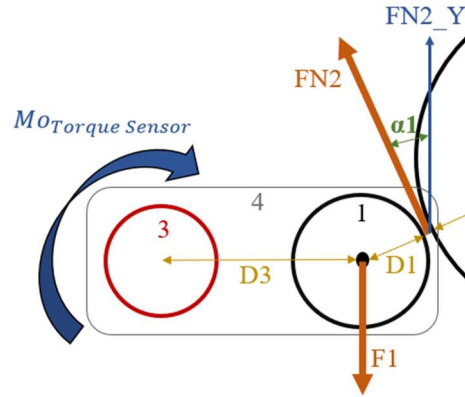


Figure 3.14 - Breadboard static diagram, detailed.

The SolidWorks model was used to calculate the α_1 angle of the teeth contact, reaching the value of 40° , Figure 3.15.

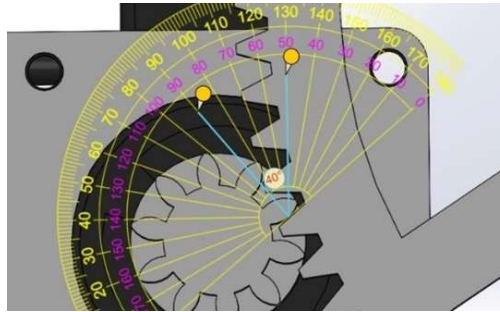


Figure 3.15 - Breadboard gears angle measurement.

Using equation 3.7:

$$M_{oTorque\ Sensor} + (F1 * D3) + (FN2_Y * (D3 + D1)) = 0 \quad (3.7)$$

And Equation 3.8:

$$FN2_Y = 49.79 * \cos(40^\circ) = 38.14\text{ N} \quad (3.8)$$

Reach Equation 3.9, giving the value of the torque requested to the sensor:

$$M_{oTorque\ Sensor} = (1 * 9.81 * 0.06) - (38.14 * (0.06 + 0.01)) = -2.08\text{ Nm} \quad (3.9)$$

The maximum capacity allowed of the torque sensor is 2.8 Nm, higher than the value module reached in Equation 3.8, which validates its usage for the application.

To extend the torque sensor's range, the distance $D3$ (Figure 3.14) needs to be as short as possible. With the CAD information of both suppliers (torque sensor and stepper motor), there was enough information to build part 4 (represented as subassembly 4, Figure 3.14) with the assurance that they do not contact each other.

All the parts were built by additive manufacturing 3D printers which can cause material resistance problems and non-accurate geometries (deviations from CAD model). These deviations may cause the gears' teeth not to contact exactly at the 40° mark. This possible deviation from CAD to manufacturing may cause torque values to be slightly different than expected.

4. Hardware Configuration

The hardware connection was a critical part of the breadboard development. This step physically connects all components, attempting to respect the requirements and the compatibilities between the different manufacturers.

Following an analysis of the robot controller, Kuka Sunrise Cabinet, and validation of integration with Beckhoff hardware and software, it was decided to make it the primary controller for the setup in order to centralize as much control as possible.

The control setup consists of three major components:

- **Robot controller, Kuka Sunrise Cabinet:** Responsible for the control of the robotic arm and interconnection with the Beckhoff Coupler, thanks to the integration with I/O mapping activities in the user programming environment.
- **Beckhoff Coupler:** This Beckhoff Coupler unit, made up of various terminals, is in charge of collecting data from the torque sensor (Beckhoff EL3008), controlling the stepper motor (Beckhoff EL7041), and establishing communication between the robot controller and a computer (Beckhoff EL6695-1001).
- **User computer:** A device running TwinCAT Run Time is collecting and sending data to the robot through the Beckhoff coupler and is also in charge of controlling the breadboard deployment activities. A Node-RED Server, reading the variables directly on the TwinCAT Run Time via ADS protocol, is responsible for data visualization and acquisition.

Additionally, there is a tablet wirelessly connected to the computer is used to display the Node-RED dashboard and control the data display parameters, as well as to enable or disable the storage function, using SQLite.

The hardware connection is made, mainly, between these components. Figure 4.1 represents the schema of connections, in more detail.

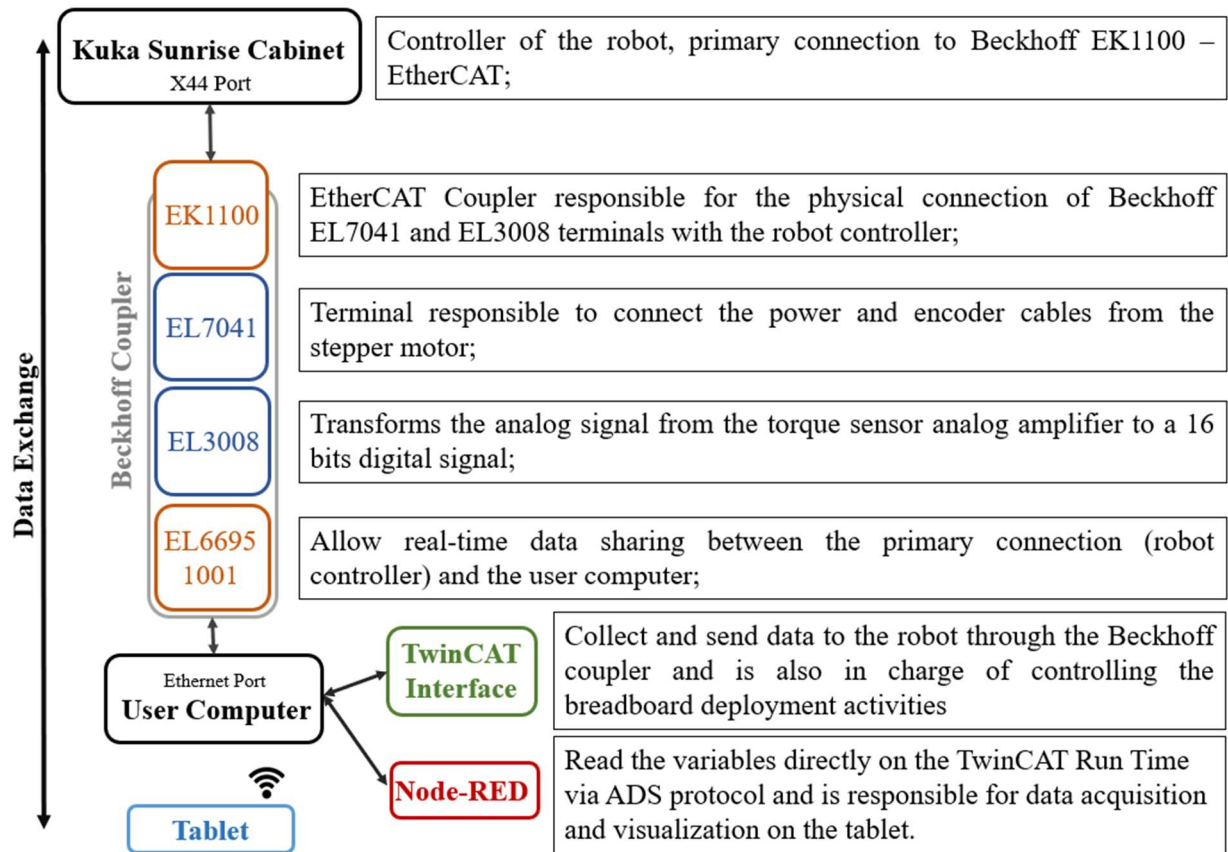


Figure 4.1 - Schema of the hardware connections.

4.1. Collaborative Robot Kuka LBR IIWA

The Kuka LBR IIWA 14 R820 (Figure 4.2) was the robot available for the elaboration of the present master thesis.



Figure 4.2 - Kuka LBR IIWA 14 R820.

This robot has seven axis with torque sensors that provide safety and, at the same time, allow to know external forces and their monitoring. Table 4.1 provides the main characteristics of the robotic arm.

Table 4.1 - Kuka LBR IIWA 10 R820 characteristics.

Characteristics	LBR IIWA 14 R820
Rated Payload	14 kg
Number of axis	7
Maximum Range	1306 mm
Wrist Variant	In-line wrist
Position Accuracy (ISO 9283)	± 0.1 mm
Axis torque accuracy	± 2 %
Mass	29.9 kg

In terms of workspace, thanks to the seven axis configuration, Kuka LBR IIWA 14 has a large workspace. Figure 4.3 shows the range and operation volume of the robot.

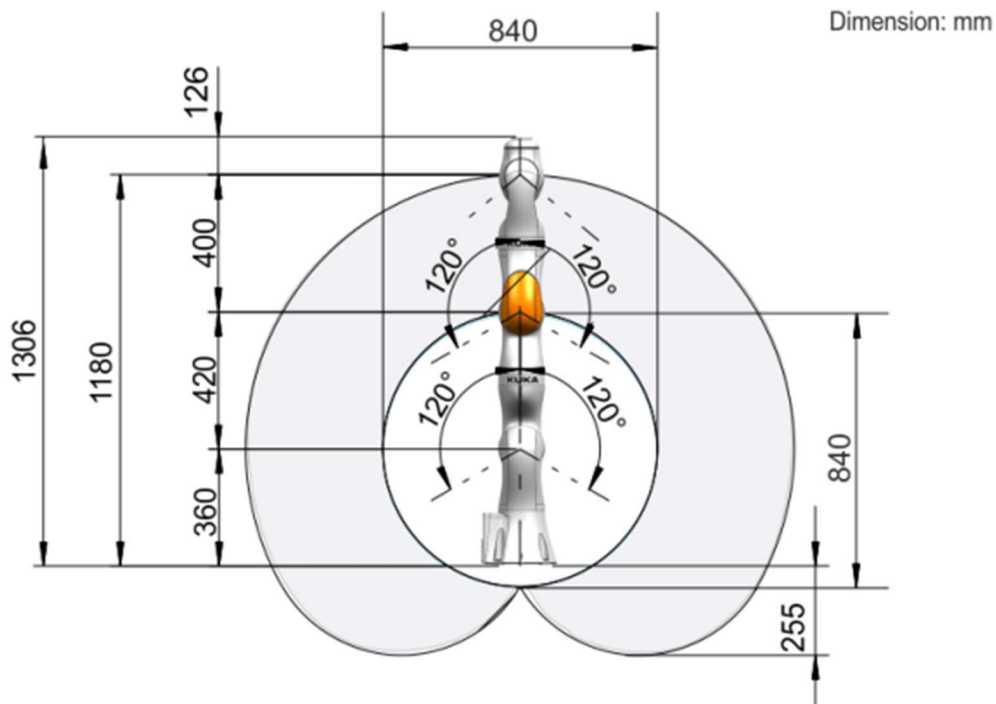


Figure 4.3 - Kuka LBR IIWA 14 workspace.

4.1.1. Kuka Sunrise Cabinet

The Kuka Sunrise Cabinet is the controller used for the Kuka IIWA. This device combines robot and logic control with the ability to interconnect devices from various suppliers, as well as system safety and process control. Table 4.2 provides the main characteristics of the controller.

Table 4.2 - Kuka Sunrise Cabinet characteristics.

Characteristics	Kuka Sunrise Cabinet
Dimensions (D*W*H)	500*483*190 [mm]
Mass	23 kg
Interfaces	USB, Ethernet, DVI-I, EtherCAT

4.2 Beckhoff Coupler

As mentioned before, the Beckhoff Coupler is composed of different terminals.

4.2.1. Beckhoff Ek1100

The Kuka Sunrise Cabinet supports interconnection with Beckhoff equipment. To physically be possible to connect the Beckhoff EL7041 and EL3008 terminals with the robot controller, a Beckhoff EK1100 EtherCAT Coupler [18] (Figure 4.4) was implemented connecting directly to the X44 Ethernet port of the controller.

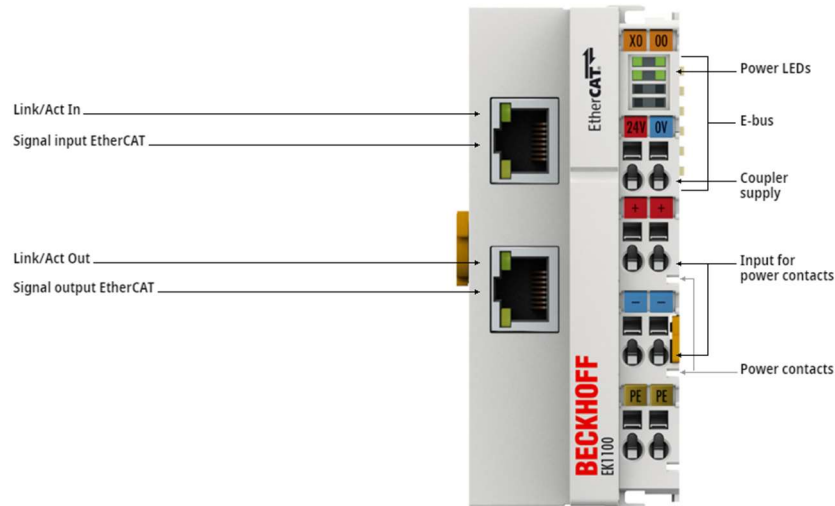


Figure 4.4 - Beckhoff Ek1100 [18].

4.2.2. Beckhoff EL7041

The stepper motor used in the application, Beckhoff AS1050-0120, needs a specific terminal, Beckhoff EL7041 [19], used to connect the power and encoder cables from the stepper motor. The terminal is shown in Figure 4.5.

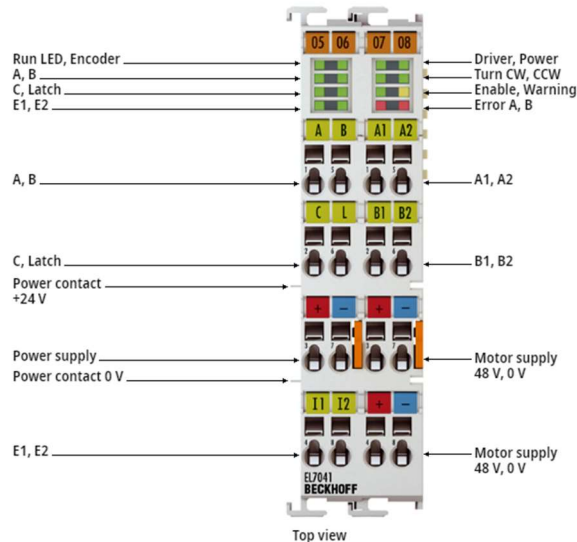


Figure 4.5 - Beckhoff EL7041 [19].

The motor needs a separated DC power source, in this case, a 48 V with enough current for the motor. The power source used is from Phoenix Contact [20] (Figure 4.6) and can provide the 48 V with the maximum current of 20 A. The highest current allowed to pass through the terminal is 5 A, therefore the power supply is more than adequate for the application.



Figure 4.6 - Phoenix Contact 48 V 20 A [20].

4.2.3. Beckhoff EL3008

This is an analog input terminal that handles signals in the -10 to +10 V range and is utilized in the application to transform the analog signal from the Futek IAA100 [21] (Figure 4.7), the analog amplifier of Futek FF400 with voltage output (range of ± 10 V), to a digital signal of 16 bits.



Figure 4.7 - Futek IAA100 [21].

The analog amplifier requires a DC power source of 24 V with more than 100 mA of maximum current. Since there is a 24 V power source in the system, it will be used to power the analog amplifier.

Because the amplifier's output is voltage, ± 10 V, a Beckhoff card was needed to make possible the acquisition of the signal and its manipulation of the programming. For that, an EL3008 Analog Input ± 10 V terminal [22] (Figure 4.8) is used.

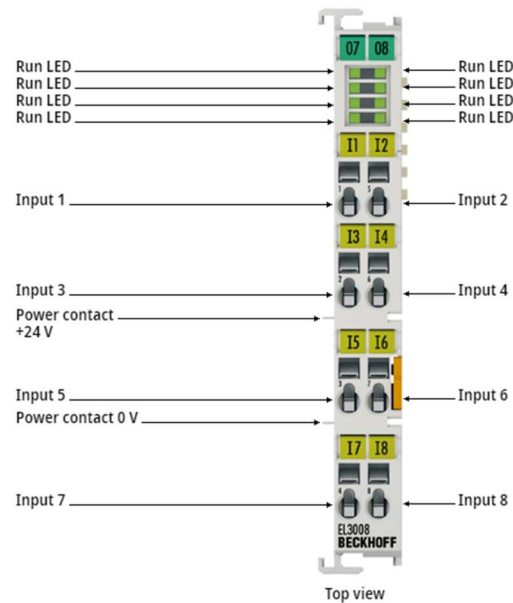


Figure 4.8 - Beckhoff EL3008 [22].

4.2.4. KRC4 Kuka EL6695-1001

To create an interface of connection between the user computer and the robot, a KRC4 Kuka EL6695-1001 [23] (Figure 4.9) was added to allow for real-time data sharing between the two connected devices. The robot, using this terminal, performed a master-master communication with a RunTine TwinCAT 3. The main program on the user's computer was correctly configured with the Beckhoff terminal and supporting software.

In this particular application, the robot is the EL6695 primary terminal, and the computer is the EL6695 secondary. Notice that is possible for the exchange of data in the two directions, as will be seen after in this report.



Figure 4.9 - Beckhoff EL6695-1001 [23].

4.2.5. Final Setup Considerations

In the final configuration, one more terminal is presented, the Beckhoff EL1008. With it, is possible to acquire digital inputs for the system [24]. In the present master thesis, it was not used but remains in the real hardware configuration.

Figure 4.10 shows the complete Beckhoff external setup described.

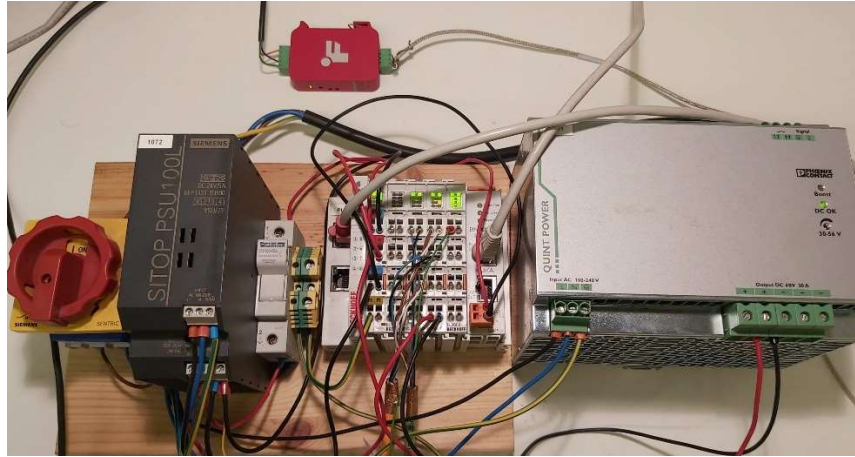


Figure 4.10 - Beckhoff external complete setup.

5. Control Development

Following the completion of the hardware connections, the application program was created. This step in the procedure was crucial because it allowed for the creation of virtual paths through which variables must travel in order to establish communications.

5.1. Sunrise Workbench/Workvisual I/O Mapping

The programming of the robot was made by using Kuka Sunrise Workbench running JAVA technology. In this programming environment (Figure 5.1) was possible to use multiple programming modes, including the compliant modes of the robot, as well as use multiple JAVA functions and functionalities. The Sunrise Workbench is the principal engineering environment for Kuka Cabinet allowing it to map security, tools, work objects, paths, velocities, etc [25].

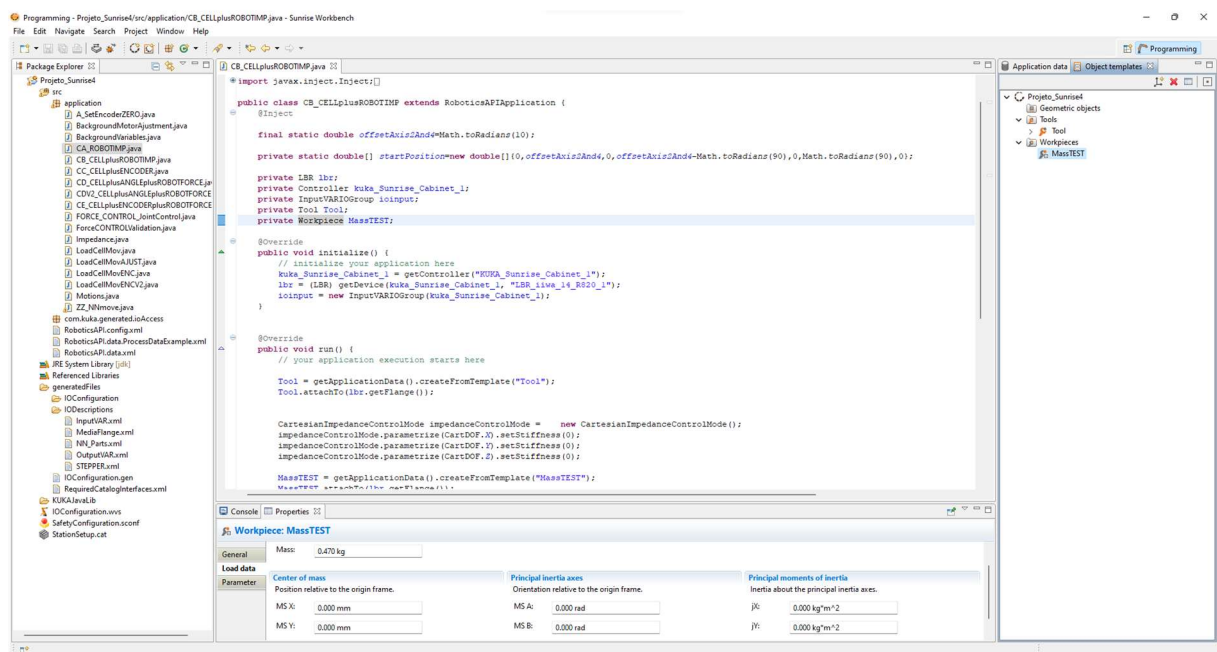


Figure 5.1 – Kuka Sunrise Workbench work environment.

On the other hand, Kuka.WorkVisual (Figure 5.2) was necessary to map and diagnose I/O configurations to the robot controller. It supports different field buses like PROFINET, EtherCAT, EtherNet/IP, etc to respond through different possibilities [26]. Notice that this software has an interface of connection with Kuka Sunrise Workbench.

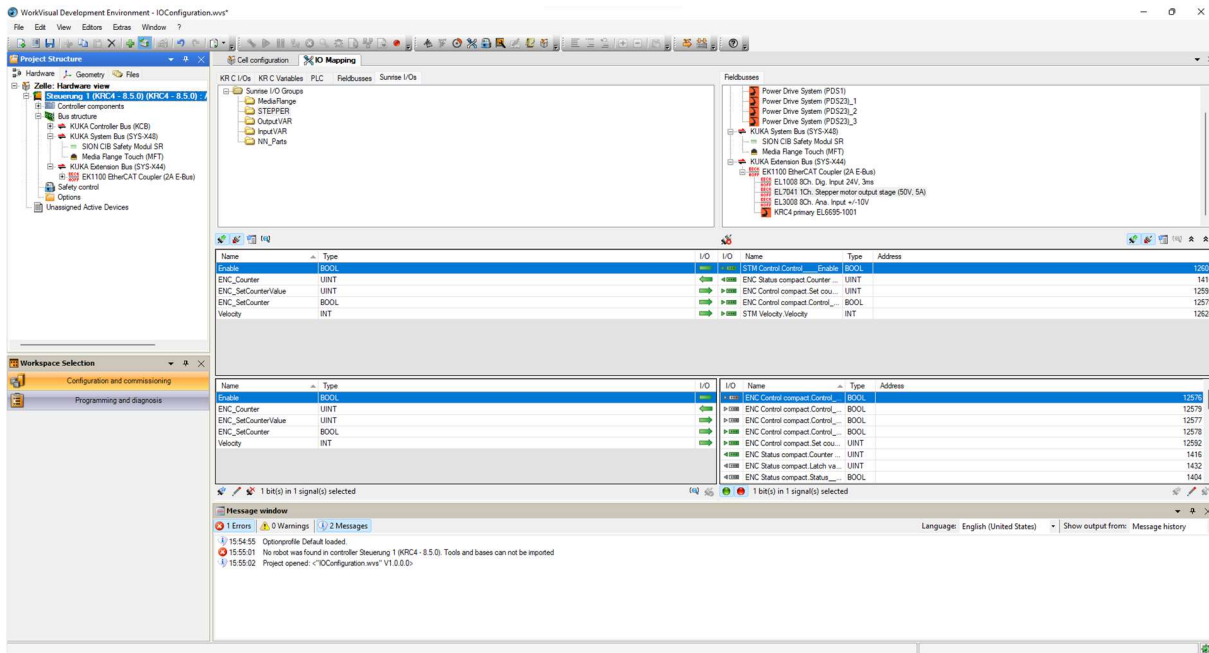


Figure 5.2 - Kuka Workvisual work environment.

The Workvisual needed to properly identify the Beckhoff hardware and the versions used. The configuration files from Beckhoff were added to the software providing, from the manufacturer, all the essential add-ons to become possible to control the systems using third-party software, in this case using Kuka Software to manipulate data and control Beckhoff terminals.

Figure 5.3 shows how the configuration bus was created. Notice that the Beckhoff EL1008 terminals were added despite not being used on the application.

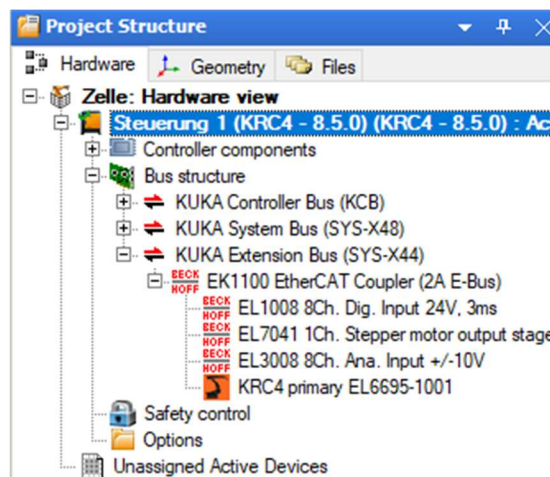


Figure 5.3 - Workvisual project structure.

5.1.1. Stepper Motor Configuration;

To enable the manipulation of the variables in the Sunrise Workbench, an I/O group was created on Workvisual to associate these virtual variables with the real ones in the Beckhoff terminals.

The stepper motor terminal was the first to be configured, Figure 5.4. To easily manipulate the variables and create more organized applications, different I/O groups were created using, in the stepper motor case, the STEPPER group.

Five variables were needed to build the stepper application:

- **STM Velocity.Velocity** – Integer variable that provides the desired running velocity of the stepper motor;
- **STM Control.Control___Enable** – Boolean variable responsible for the enabling of the motor. Enabling the motor, it will run at the current velocity value;
- **ENC Status compact.Counter value** – Positive integer variable that outputs the current value of the stepper motor encoder;
- **ENC Control compact.Control___Set counter** – Boolean variable responsible to set the encoder counter to the value of the variable ENC Control compact. Set counter value;
- **ENC Control compact. Set counter value** - Positive integer variable that defines the value to force the encoder when the ENC Control compact.Control___Set counter is enable.

Name	Type	I/O	Name	Type	Address
Enable	BOOL	→	STM Control.Control___Enable	BOOL	12608
ENC_Counter	UINT	←	ENC Status compact.Counter value	UINT	1416
ENC_SetCounterValue	UINT	→	ENC Control compact.Set counter value	UINT	12592
ENC_SetCounter	BOOL	→	ENC Control compact.Control___Set counter	BOOL	12578
Velocity	INT	→	STM Velocity.Velocity	INT	12624

Figure 5.4 - Stepper motor I/O configuration.

5.1.2. Torque Sensor Configuration;

Similar to the stepper motor, the torque sensor terminal needs to be mapped to a variable on an I/O group. A new group was created named InputVAR where all the input variables are mapped.

The only variable mapped is the AI Standard Channel 1.Value (Figure 5.5) that provides an integer value of the analog input received, in this particular case, from the torque sensor amplifier connected to the Beckhoff EL3008 terminal.

Name	Type	I/O	Name	Type	Address
In1	INT	←	AI Standard Channel 1.Value	INT	1480

Figure 5.5 - Torque Sensor I/O configuration.

5.1.3. Kuka / TwinCAT Interface;

The Kuka / TwinCAT interface allows the variables to be transferred between the Kuka controller and an external computer.

In the configuration tab of the KRC4 primary EL6665-1001, it was possible to select the process data objects by specifying the length of the data transmission between the two terminal-connected devices. Each variable transferred was a double word with a length of 32 bits.

A total length of 128 bytes for inputs and 128 bytes for outputs was defined. With this configuration, it was possible to have 32 inputs and 32 outputs running between the robot and the user computer. The objective of the applied variable definition was to have a numerous range of variable communication.

As for the I/O mapping, and in particular, for the variables to connect, they need to match the variable type. As all the variables on the KRC4 primary EL6665-1001 are default double words, these variables need to be transformed to be mapped. Figure 5.6 shows an example of transformation. On the left side is the original configuration and on the right top is the desired configuration for integer variables. In situations, as described, will only be used the first 16 bits of the original variable, leaving the other 16 bits unmapped.

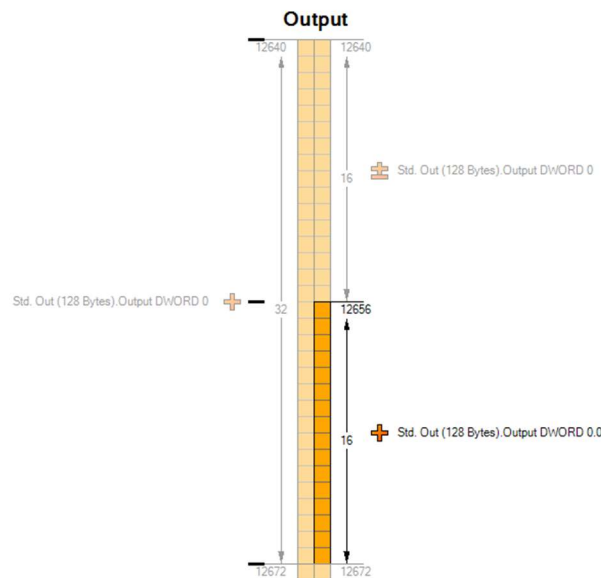


Figure 5.6 - Beckhoff EL6695 variables I/O configuration.

The variables mapping was repeated for these variables, creating a new I/O group named OutputVAR and using a previously used, InputVAR. Figure 5.7 and Figure 5.8 illustrate the input and output mapping. The variable's meaning will be explained in the next sub-section to better understand its application.

Name	Type	I/O	I/O	Name	Type	Address
In_MotorHighL	BOOL	←	←	Std. In (128 Bytes).Input DWO...	BOOL	
In_MotorHighR	BOOL	←	←	Std. In (128 Bytes).Input DWO...	BOOL	
In_MotorLowL	BOOL	←	←	Std. In (128 Bytes).Input DWO...	BOOL	
In_MotorLowR	BOOL	←	←	Std. In (128 Bytes).Input DWO...	BOOL	

Figure 5.7 - Beckhoff EL6695 input configuration.

Name	Type	I/O	I/O	Name	Type	Address
Out0	INT	→	→	Std. Out (128 Bytes).Output D...	INT	
Out1	INT	→	→	Std. Out (128 Bytes).Output D...	INT	
Out2	INT	→	→	Std. Out (128 Bytes).Output D...	INT	
Out3	INT	→	→	Std. Out (128 Bytes).Output D...	INT	
Out4	UINT	→	→	Std. Out (128 Bytes).Output D...	UINT	
Out5	INT	→	→	Std. Out (128 Bytes).Output D...	INT	

Figure 5.8 - Beckhoff EL6695 output configuration.

5.2. TwinCAT Interface

The software TwinCAT was used to create the interface of communication between the robot controller and the computer. The architecture of the software consists of a real-time XAR and uses Microsoft Visual Studio to enable the programming automation objects in several languages, such as programming PLCs, C, C++, and Matlab/Simulink, among other languages [27].

The first step was the installation of the KRC4 Kuka secondary EL6695-1001. Adding the EtherCAT device and the configuration file to the TwinCAT corresponding to the KRC4 Kuka secondary EL6695-1001, the configuration on the process data tab needed to match the one made in the Workvisual, explained in the previous section. In this case, the outputs on the Workvisual are the inputs in the TwinCAT and the other way around. Figure 5.9 shows the example for the process data configuration of the input variables.

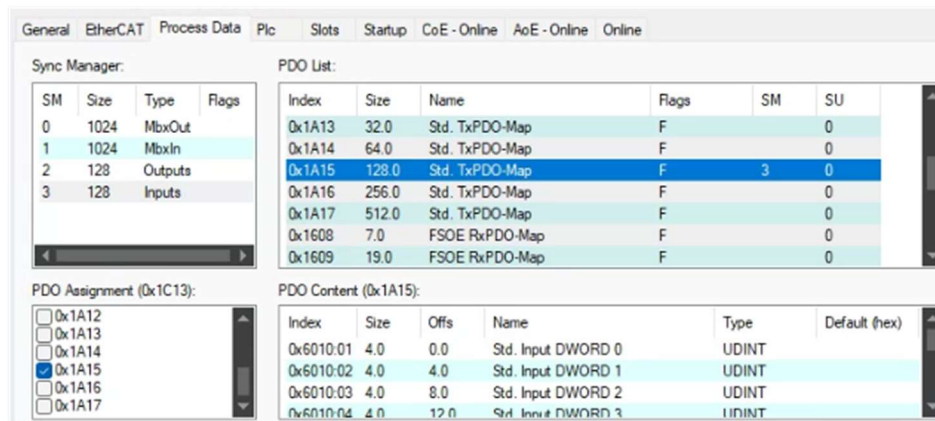


Figure 5.9 - TwinCAT process data configuration.

With the communication process completed, a standard PLC project was added containing a Global Variable List, the Main program, and a Visualization object. The structure of the project is illustrated in Figure 5.10.

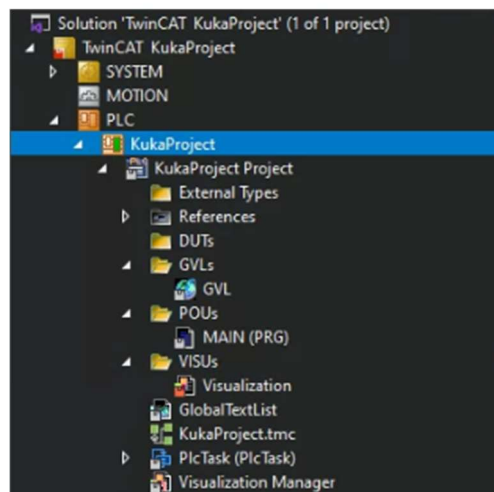


Figure 5.10 - TwinCAT PLC project.

A Global Variable List was created to separate the variables that will communicate between the TwinCAT Run Time and the IoT platform, Node-RED. The mapping and work principles of the platform will be explained in the respective section.

5.2.1. Main Program

The Main program (Appendix C) was the principal task ongoing in the Run Time with the Visualization giving it the necessary inputs. Figure 5.11 shows the Visualization (HMI) created for the application in TwinCAT, responsible for the stepper motor control and the torque sensor calibration.

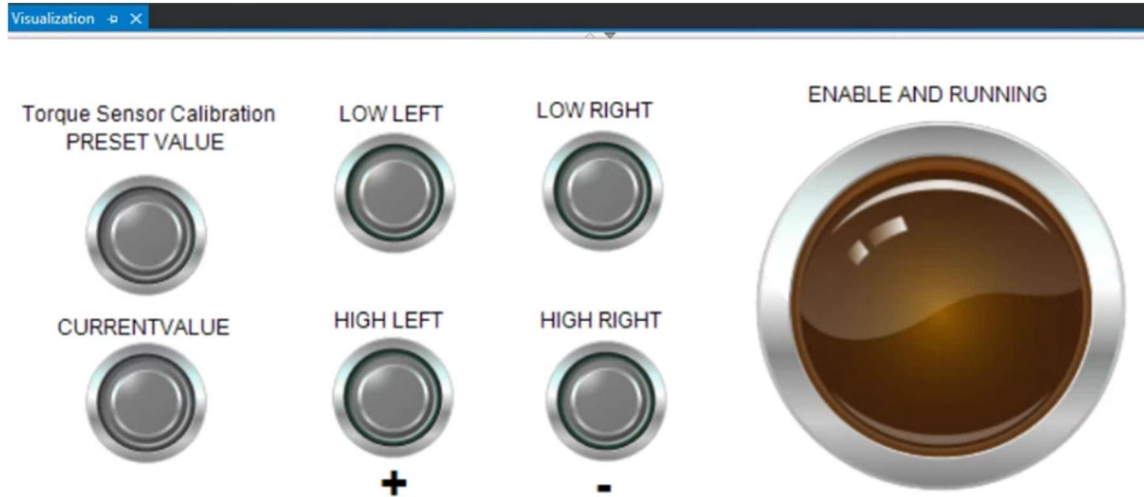


Figure 5.11 - TwinCAT Visualization.

This program has three principal sections:

- **Torque sensor calibration:** In this section, it provides the mapping of the visualization interface with the capability to calibrate the torque sensor.

One of the buttons, Torque Sensor Calibration preset value (Figure 5.11), had the function to set the preset value of the torque sensor to ensure that everything that the sensor measure after that calibration is an external torque applied to the mechanism.

The other button, Torque Sensor Calibration current value (Figure 5.11), sets the current value to the sensor. Like this, the sensor is capable of measuring torque variations with the desired offset.

In sub-section 6.2 there is more detailed information on how the preset value was obtained.

- **Visualization motor adjustment:** Code responsible for the mapping of four Booleans associated at the four buttons on the Visualization, Figure 5.11. Table 5.1 lists the variables that will be four inputs to the robot as well as their mean.

Table 5.1 - Output variables going to the robot.

Variable	Type	Meaning
In_MotorHighL	Bool	Beam rotation clockwise, high velocity
In_MotorHighR	Bool	Beam rotation counterclockwise, high velocity
In_MotorLowL	Bool	Beam rotation clockwise, low velocity
In_MotorLowR	Bool	Beam rotation counterclockwise, low velocity

One led indicator was added to easily see when the stepper motor is enabled and running.

- **Node-RED mapping variables:** Node-RED was the IoT platform used for the data visualization and data acquisition for the application. Starting with the variables, the main goal of this section is to map and transform the five variables provided by the robot controller, described in Table 5.2.

Table 5.2 - Input variables from the robot.

Variable	Type	Meaning
Out0	Int	Stepper motor velocity
Out1	Int	Torque sensor value
Out2	Int	Robot flange external force X-axis
Out3	Int	Robot flange external force Y-axis
Out4	UInt	Stepper motor encoder value
Out5	Int	Robot flange external force Z-axis

The variables must be adjusted in order to reflect meaning to the user. They were changed in the Beckhoff program, while data display and acquisition were left to the Node-RED platform. Table 5.3 sums up the variables involved in the exchange between TwinCAT and Node-RED.

Table 5.3 - Output variables going to the Node-RED.

Variable	Type	Meaning
MotorVEL	Int	Stepper motor velocity
MotorENC	UInt	Stepper motor encoder value
BeamAngle	Real	Beam angle with the horizontal (degrees)
Torque	Real	Torque Value (Nm)
Grams	Real	Offloading error (experiment) (g)
ForceX	Real	Robot flange external force X-axis (N)
ForceY	Real	Robot flange external force X-axis (N)

Stepper motor velocity and encoder value are two values that do not need manipulation, being directly mapped to the corresponding global variable.

In the case of the robot flange external forces, the background task is in charge of obtaining them and transmitting them via the Beckhoff EL6695-1001 terminal, which converts a real value into an integer one to facilitate data interchange. Following that, the TwinCAT merely needs to transform it back to a real value.

The Torque, Grams, and BeamAngle variables involve different approaches with a more complex configuration. Section 6 explains how the conversion of these variables was made.

5.3. Robot Background Tasks

All the control are performed by the robot controller, which enables the interconnection of everything and ensures the correct mapping of the variables.

To be able to perform tasks in the background parallel to a running application, the Kuka Sunrise Cabinet allows the creation of background tasks to run continuous processes. Multiple backgrounds can run at the same time independent of the running robot application.

In this master thesis, the Kuka Sunrise Cabinet is configured to run two background tasks with two hundred milliseconds run cycle.

5.3.1. Motor Control Background Task

This background task is responsible for the control of the stepper motor using the boolean inputs provided by the TwinCAT (Appendix D).

Four boolean variables are used to enable four different velocities of the stepper motor. Table 5.4 lists the input variables that came from TwinCAT to the background task.

Table 5.4 - Input variables from the TwinCAT.

Variable	Action on the control
In_MotorHighL	Stepper motor counterclockwise, velocity=9
In_MotorHighR	Stepper motor clockwise, velocity=9
In_MotorLowL	Stepper motor counterclockwise, velocity=1
In_MotorLowR	Stepper motor clockwise, velocity=1

The justification of the velocities used in the control is made in sub-section 6.1.

For security reasons, programmable logic was implemented with the emergency button of the Kuka smartPAD with constant monitoring of the security state of the robot and disabling the stepper motor in case of emergency.

5.3.2. Variables Background Task

This background task is responsible for the mapping of variables between the Sunrise Workbench and the Beckhoff EL6695-1001 terminal (Appendix E).

Without using the Sunrise Workbench as an intermediary, it is not possible to connect variables from multiple field busses, such as the torque sensor terminal, Beckhoff EL3008, and the Beckhoff EL6695-1001 terminal. So, in this background task of the robot, the variables are being read from the source equipment and then resend to the throw Beckhoff EL6695-1001 outputs. Figure 5.12 shows the example for the stepper motor variables.

```
// Velocity and Encoder to PC
iooutput.setOut0(iostepper.getVelocity());
iooutput.setOut4(iostepper.getENC_Counter());
```

Figure 5.12 - Kuka background task I/O mapping.

In the case of the robot flange external forces readings, every cycle the robot controller is going to get the force vector on the flange, separate the components, multiply by 200 and transform it into an integer number to be possible to transfer them over the Beckhoff EL6695-1001 terminal as an integer number.

A 16-bit integer value accumulates values ranging from -32768 to +32767. When the force value is multiplied by 200, the resolution in the information receiver is 0.005 N. In these conditions, the maximum allowable force to send is 163 N.

5.4. IoT platform - Node-RED

Node-RED is an IoT platform that enables the interconnection of several equipments, dashboard creations to easily visualize data and configure services through JavaScript programming [28].

The purpose of using Node-RED in this master thesis relates to three main aspects:

- Allow the creation of live data display to easily see the results of the tests and, in the development phase, allowed the quick in-site analysis with previous results and the understanding of the phenomenon in live time;
- Data acquisition platform where its possible to configure databases to store the data on many types of formats;
- Using an IoT platform opens the horizons to the use of other devices to display the data, in this case, the usage of a tablet that can move freely around the testing ground.

The data acquisition was performed using SQLite, a relational database management system that allows storing data in the local computer. Using DB Browser (SQLite) it was possible to create two distinct tables, one for the torque sensor information and the other for the robot ones, and ensure that all the data is collected with an open-source library, possible to read and filter with multiple platforms [30].

In the application, the data acquisition and visualization frequency is 1 Hz.

5.4.1. Dashboard Development;

The dashboard feature of Node-RED provides the user see, in live-time, the data acquired by the TwinCAT application. The programming nodes can be observed in Appendix F.

In this scenario, in the Node-RED programming environment (Figure 5.13) it was added a node package responsible for the ADS connection with TwinCAT Run time [29] running on the user computer. These nodes are specifically for applications in which the TwinCAT Run time and Node-RED host server run on the same machine.

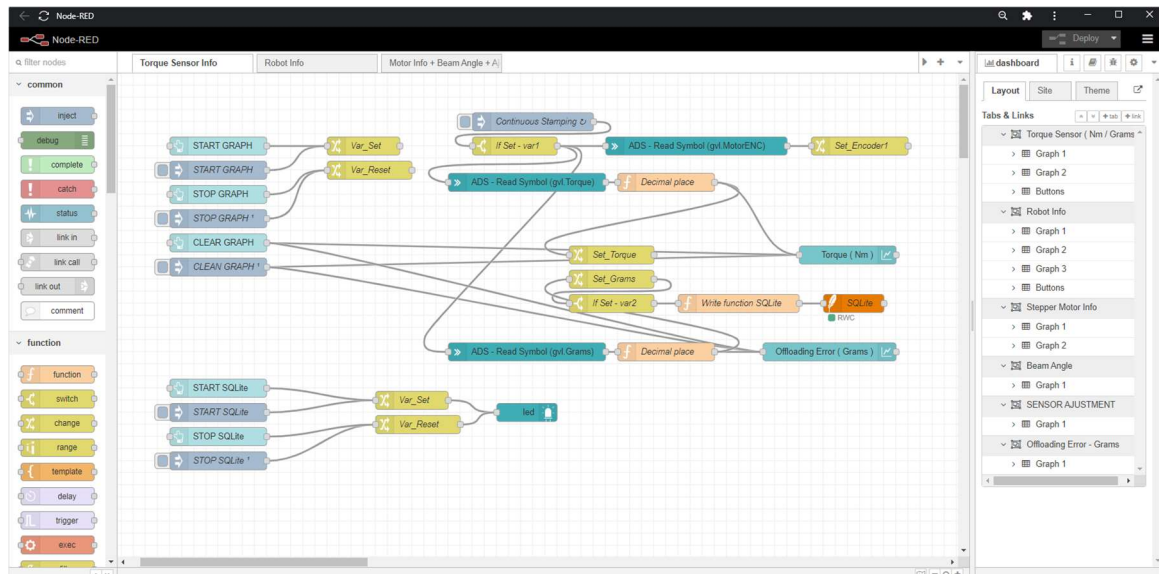


Figure 5.13 - Node-RED work environment.

Knowing the IP of the user computer and connecting it to a common network of a tablet, is possible to use it for the dashboard display, Figure 5.14.

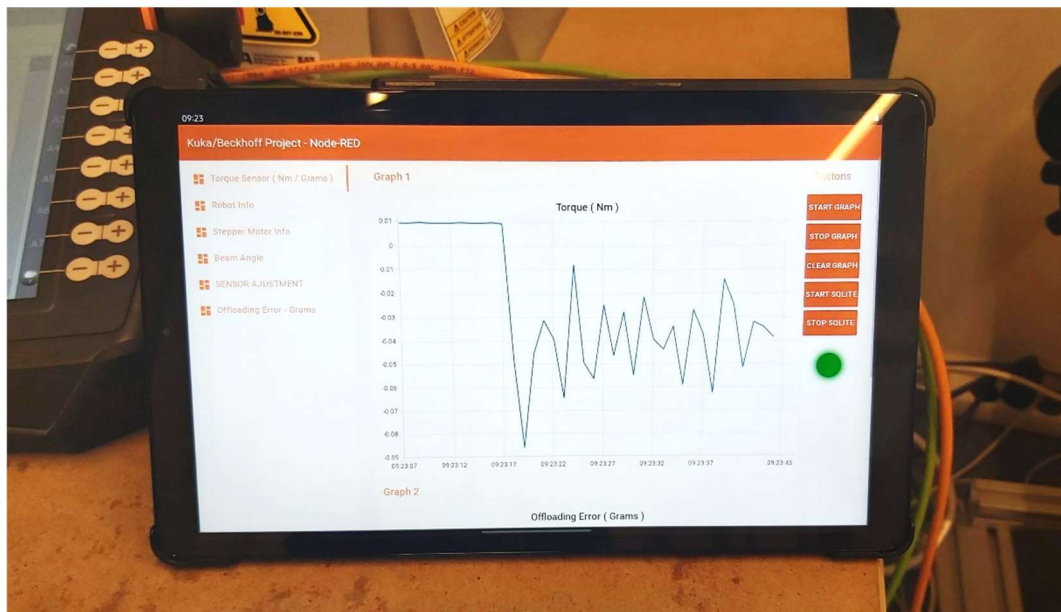


Figure 5.14 - Node-RED tablet dashboard.

The dashboard is composed of six tabs (Appendix G), each one with different functionalities:

- **Torque Sensor Information** – Tab responsible for the visualization of the torque sensor information (Torque conversion and Grams experiment) and the control of the data acquisition;
- **Robot Info** – Made to visualize and control the data acquisition of the Robot Forces on the X, Y, and Z axis;
- **Stepper Motor Info** – Informations of velocity and encoder position are displayed here.
- **Beam Angle** – Outputs the current value of the beam angle;
- **Sensor Adjustment** – Display the real-time value of the torque sensor for the initial calibration of the system before deployment;
- **Offloading Error – Grams** – Graphic that outputs the instant value of grams offloading error;

6. System Characterization

With the overall characterization of the application in mind, this chapter computes all of the constants mentioned in the control development section.

6.1. Beckhoff Stepper Motor Velocity

Velocity is one of the inputs of the stepper motor control, defining the rotational speed that the motor will control while it is running. The velocity is measured by counting the increments of the encoder per second.

In sub-section 3.4.1 was calculated the desired velocity of the stepper motor, which is 0.16 rpm. Equation 6.1 shows how the velocity is obtained [31]:

$$Velocity = 196602 * \frac{n_{set}}{\varphi * f_{max}} \quad (6.1)$$

With:

- Velocity: Setpoint in increments of the encoder per second;
- n_{set} : Setpoint desired in rpm;
- φ : Step angle of the stepper motor, in the case of the Beckhoff AS1050, is equal to 1.8° ;
- f_{max} given by Equation 6.2;

$$n_{max} = f_{max} * \frac{\varphi}{6} \quad (6.2)$$

With:

- n_{max} : Maximum reaching speed, in the case of the Beckhoff AS1050, is equal to 585 rpm.

Having Equations 6.1 and 6.2 into consideration, the value of velocity reached is 8.96 increments/s. As the input value had to be an integer value, velocity was equal to 9 increments/s.

6.2. Torque Sensor Calibration

The torque sensor was calibrated for the application by removing the bigger gear and testing the torque sensor value a number of times in order to exclude any interference from other components that could cause hidden parasitic torque. The torque value is an integer number derived from the analog to digital conversion of the Beckhoff terminal EL3008, and it is used in the calibration because it is the terminal's original value without any conversion (Nm, for example). Figure 6.1 represents the setup and Figure 6.2 is the resulting graphic of the experiment.

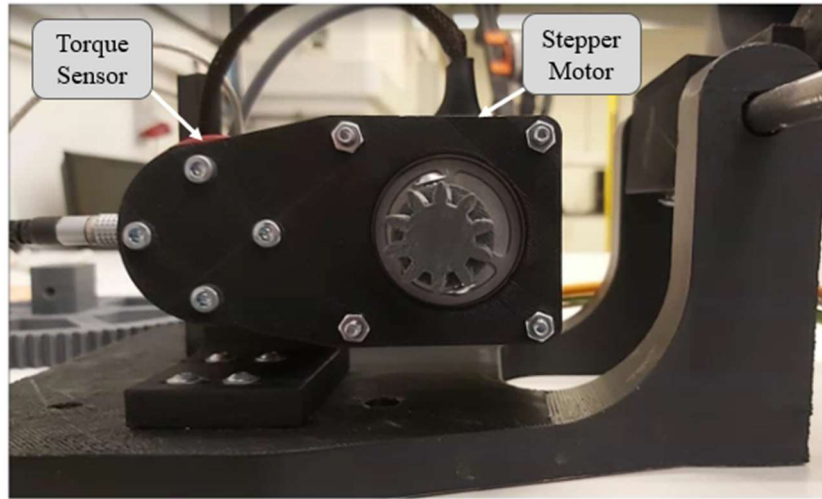


Figure 6.1 - Torque sensor calibration setup.

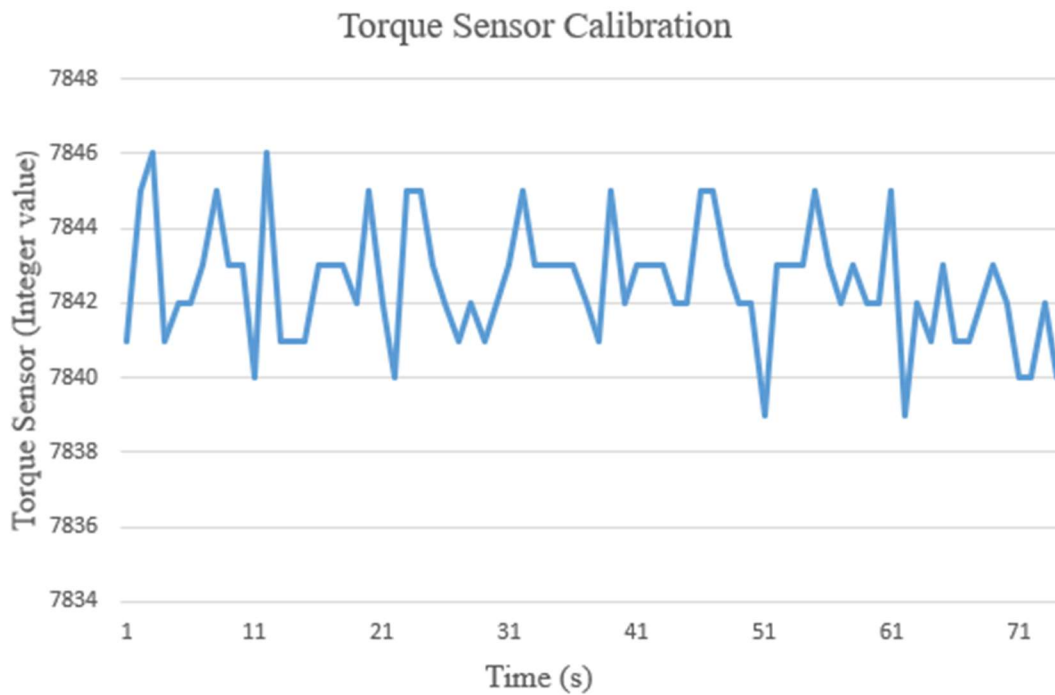


Figure 6.2 - Torque sensor calibration results.

The average value for the torque sensor was 7843, being this the default preset value.

6.3. Torque Sensor Gradients

The tests performed in the application build the need for two gradients calculated using the torque sensor information.

6.3.1. Torque gradient calculation

The torque value, in Nm, represents the torque measure in the sensor. The gradient for the conversion is reached in Equation 6.3:

$$\text{Torque Gradient} = \frac{\text{MaxBinryValue}}{\text{MaxIntegerValue}} = \frac{2.8246}{32767} = 0.0000862 \text{ Nm/Int}_{\text{Sensor}} \quad (6.3)$$

6.3.2. Mass gradient calculation

An experience was designed to make possible the measurement of the offloading error, in grams, of the mechanism, creating a kind of “scale” for developing purposes. Figure 6.3 represents the setup, using the following procedure:

1. Start by positioning the beam in a zero-degree position with no offloading;
2. Reset the torque sensor in a way that it manages to read a value close to zero;
3. One by one, add on the end of the beam, orange arrow on Figure 6.3, the incremental masses of 7 g allowing the torque sensor to stabilize between sets;

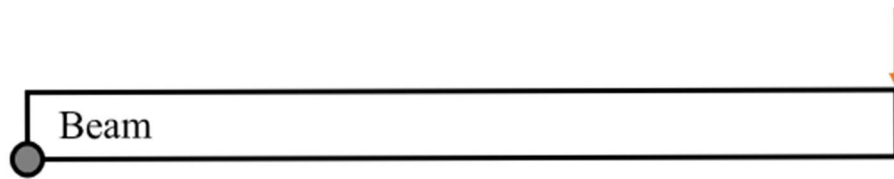


Figure 6.3 - Mass gradient calculation setup.

Figure 6.4 shows the graph obtained with the experiments obtaining the average results for the different masses in Table 6.1.



Figure 6.4 - Mass gradient results.

Table 6.1 - Mass gradient results.

Test	Weight (g)	Average torque sensor value
Level 1	7	-941.56
Level 2	14	-1781.66
Level 3	21	-2639.81
Level 4	28	-3502.86

With the values reached for the different weights, it was possible to establish a linear trendline (Figure 6.5) with the gradient calculated in Equation 6.4.

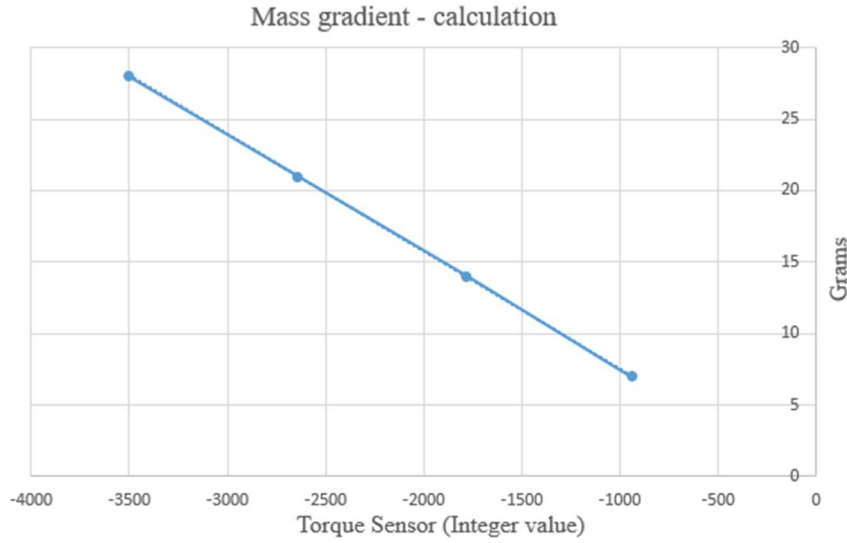


Figure 6.5 - Mass gradient calculation.

$$\text{Grams Gradient} = -0.0081944 \text{ g/Int}_{\text{Sensor}} \quad (6.4)$$

This experimental result is undoubtedly subject to error because the measurements are performed with the mechanism preloaded with the beam weight, a condition that does not match the actual circumstance when offloading tests are performed. The plan was to have a second validation system that was independent of the inverse cinematics of the beam and use this experimental testing to evaluate the theoretical value during application development.

6.4. Beam Angle Gradients

To calculate the angle beam for the data display and robot programming, the procedure was:

- Create a task on the robot to reset the encoder, set to zero, in a known angle position, in this case, when a mechanical hard stop is reached and the torque sensor is measuring a value close to 0 Nm.
- Using a surface level, the zero degrees position of the beam was found and the encoder value was recorded. Repeating this three times obtained a resulting mean value of 367;
- Create a new variable that constantly measures the difference between the zero position encoder value and the actual value of the encoder;
- Multiply by a gradient, Equation 6.5, obtained by positioning the beam at two different angles and registering the encoder values.

$$\text{Degrees Gradient} = 0.011414 \text{ degrees/Int}_{\text{Encoder}} \quad (6.5)$$

Notice that the robot controller manipulates angles in radians. So, another gradient was calculated (Equation 6.6) that changes Encoder integers to radian angles

$$\text{Radian Gradient} = 0.000199 \text{ radians/Int}_{\text{Encoder}} \quad (6.6)$$

In Figure 6.6 it is possible to observe the graph with the evolution of the angle with the increments of the encoder value.

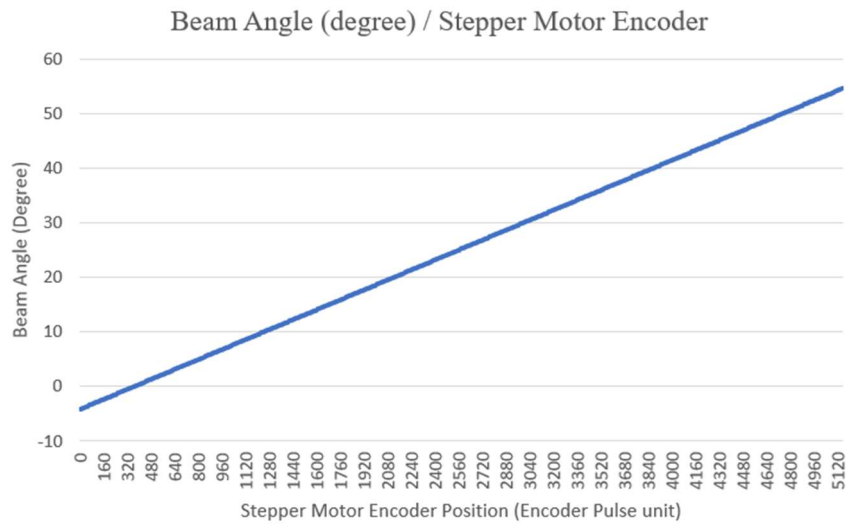


Figure 6.6 - Beam angle/stepper motor encoder.

Verify that the initial position, when the beam touches the mechanical hard stop, is -4.18°, Figure 6.7.

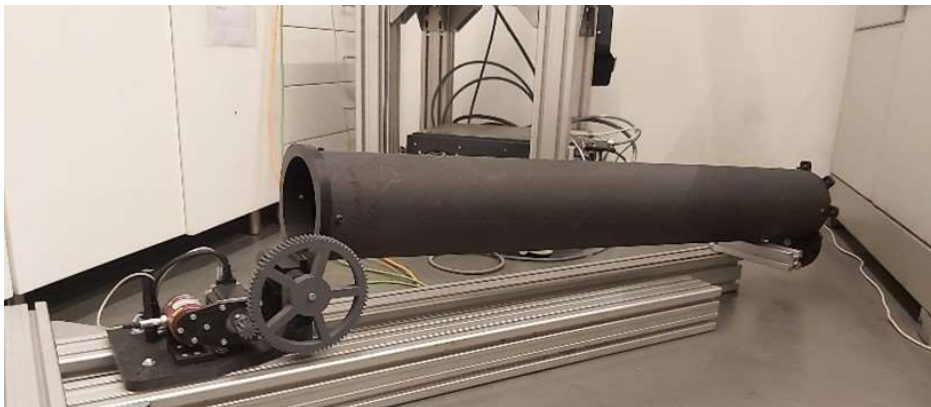


Figure 6.7 - Initial beam position.

7. Robot Control Development

7.1. Programming a Compliant Robot

In addition to position sensors, the Kuka IIWA 14 features torque sensors on each joint to enable current torque monitoring. This feature enables, in addition to position control, the implementation of compliant behavior on the manipulator.

The solution's inherent model is a virtual mass-spring-damper system (Figure 7.1), with configurable stiffness and damping values [32]. When the robotic manipulator comes into contact with the environment, it must exhibit compliant behavior. Impedance control, one of the most reliable strategies, is used to manage the interactions [33].

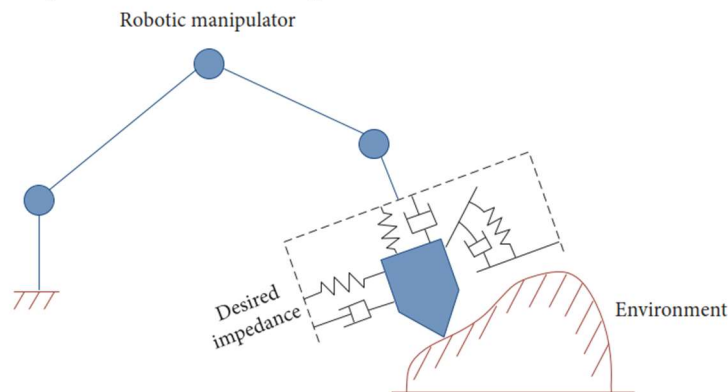


Figure 7.1 - Compliant robot in contact with the external environment [32].

The Impedance controller of the Kuka Sunrise Cabinet has three different modes configurable and possible to be used in programming:

- Cartesian impedance controller – In this mode, it is possible to set stiffness, damping, and forces applied in addition to the spring stiffness for each cartesian axis. The robot builds a cartesian referential with six degrees of freedom and parametrizes each one. Notice that all the axis directions are given in relation to the tool orientation.
- Cartesian impedance controller with overlaid force oscillation – Specific form of the cartesian impedance controller where it is possible to overlay constants or sinusoidal forces to the robot to enable search runs and tool vibration for assembly purposes.
- Joint-specific impedance controller – The robot controller sets a value of stiffness and damping for each of the seven robot joints.

7.2. Programming Modes

7.2.1. Robot Impedance Control (RIC)

Robot Impedance Control (Appendix H) was the first mode tested in the application. Remains in the concept of not providing any input to the robot controller, making it take full control of the manipulation arm without any external input.

The Robotic arm behavior must be as compliant as possible to freely follow the testing setup while it is deploying. Previous collaborative applications [32], [34]–[36] set that stiffness needs to be as low as possible, in this case, equal to zero. The robot can control each of the modes explained above with this condition.

In terms of impedance control modes, cartesian and joint-specific impedance controllers may be suitable for this programming mode because it is unimportant whether the flange is controlled as long as the robot adheres to the testing arrangement. To test what controller could work better for the application, a preliminary test was made using a dynamometer capable of reading forces in one direction. Figure 7.2 shows the setup of the test for the two programming modes.

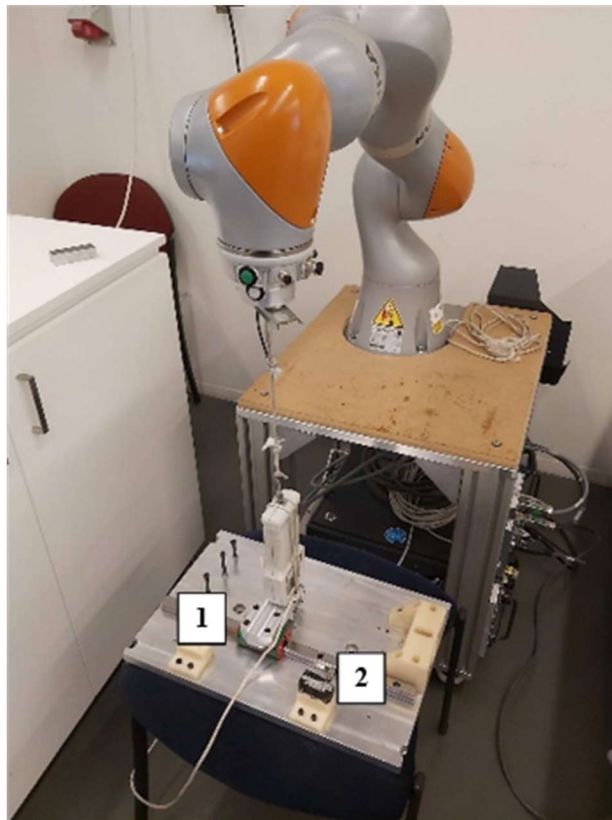


Figure 7.2 - Impedance modes test setup.

This test evaluates the displacement of the cable by measuring the force applied to the dynamometer SAUTER FK250 [37] (Figure 7.3), performing low-velocity movements on the track from 1 to 2 and the other way around, Figure 7.2. The robot was, in the two modes, programmed to make a constant force of 2 kg in the vertical direction, referring to the world referential.



Figure 7.3 - SAUTER FK250.

Considering Figure 7.4 as a schema of the forces applied on the dynamometer with:

- F – Resultant force applied;
- F_z – Constant force applied by the robot;
- F_x – Force applied from the displacement of the cable (schematically over-represented);

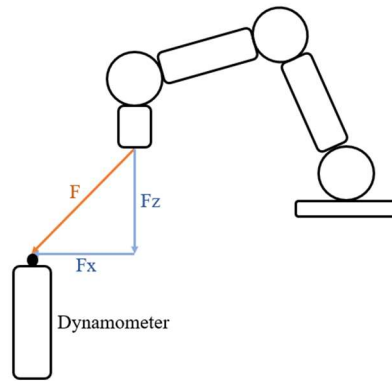


Figure 7.4 - Schema of the forces applied to the dynamometer.

During the tests, the dynamometer reads higher variations of the values using the cartesian impedance controller, with values of stiffness on axis X, Y, and Z equal to zero. Using the robot in a joint-specific impedance controller, the values of F (Figure 7.4) are much more stable with fewer variations which means that F_x is smaller.

Once the application starts in any programming mode, a while cycle begins making continual relative movements to the same location as the robotic arm, essentially performing it in impedance mode with the stiffness set to zero. In this manner, the robotic arm moves freely in the workspace, engaging with the environment, in this example the breadboard configuration.

This preliminary analysis concludes the joint-specific impedance controller was chosen to perform the first test.

7.2.2. Robot Impedance Control plus Torque Sensor (RIC_TS)

This programming mode (Appendix I (for the XZ plane), J (for the XYZ workspace)) consists of using the torque sensor information to give the robot feedback from the mechanism and close the control loop, this while keeping the torque sensor measuring the torque on the mechanism. The response time of the system does not represent significant relevance in the application due to low-velocity movements, creating conditions for fast adjustments.

With the current test setup (vertical deployment of the hinge), the algorithm controls the torque sensor values such that if they are outside the threshold area, the robot moves up or down a certain distance on the Z-axis, defined by a proportional gain multiplied by the difference between the torque sensor's zero value and the instantly one.

In other words, if the robot senses too much force in the hinge it moves up, if it senses lightweight it moves down.

Like the programming mode before, the robot is operating in a while cycle but in this case, it is performing relative movements as the threshold is reached. To ensure that the flange is always in a known position, always with the same orientation of the different axis (Figure 7.5), the cartesian impedance controller is used and the only axis to parametrize are the X, Y, and Z. The stiffness of the rotational axis will not be modified, their standard value is enough to not allow the rotation of the robot flange.

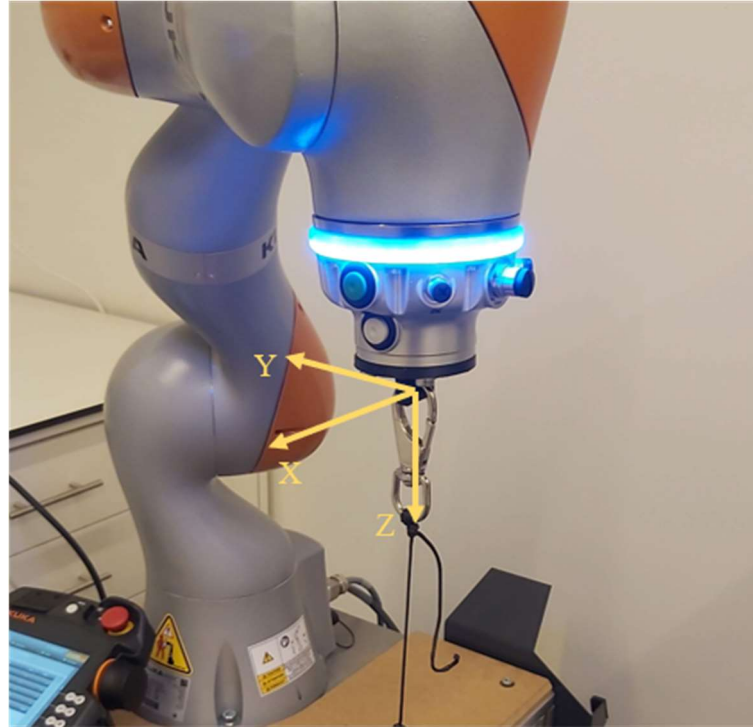


Figure 7.5 - Position of the robot flange referential.

The robot, in this case, needs to be in an impedance mode to become possible the compliant comportment in the no-offloading directions.

7.2.3. Robot Impedance Control plus Torque Sensor plus Encoder (RIC_TS_E)

Using the encoder of the motor as an input for the programming mode (Appendix K) is possible to measure the angle of the beam during the deployment. Using the radian gradient calculated in sub-section 6.4, the robot adjusts its instructions to move.

With the angle value, instead of allowing the robot to perform movements through the cartesian impedance controller mode, the controller calculates the theoretical displacement in the two directions of the XZ deployment plan. Figure 7.6 shows the difference between the displacement calculation with the encoder mode and the previously analyzed, Robot Impedance Control plus Torque.

In this case, the robot moves with a synchronous motion with the hinge's direct input.

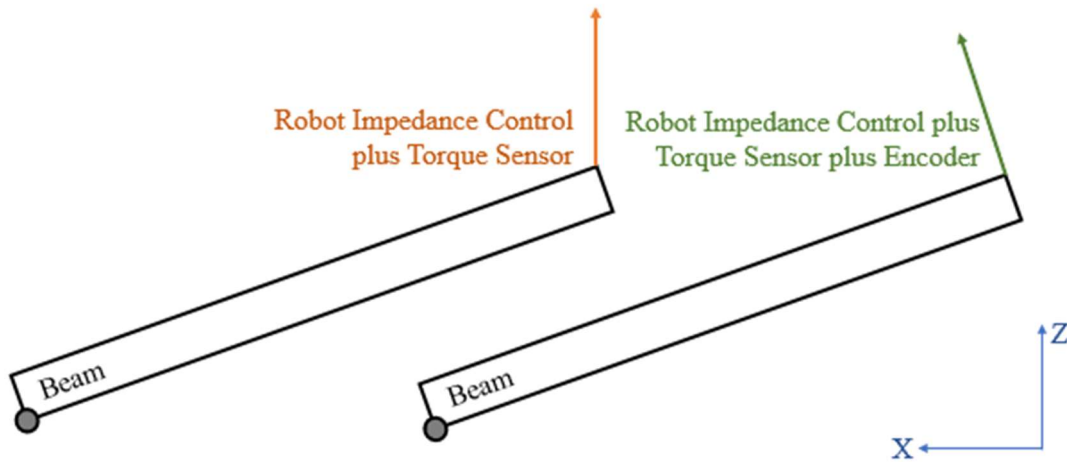


Figure 7.6 - Difference between the displacement calculation of the two programming modes.

7.2.4. Robot Impedance Control plus Torque Sensor plus Robot Forces (RIC_TS_RF)

This last test (Appendix L (for the XZ plane) and Appendix M (for the XYZ workspace)) was performed to create a virtual scenario of having an external bi-dimensional X-Y axis force sensor being the input to the robot. For that, the robot controller is calculating the external forces in the flange and uses them as input for the program. As the program developed in the Robot Impedance Control plus Torque Sensor, when the forces reached a threshold, the robot moves in the direction that it is being “pulled” by the cable.

This mode employs the same orientation of the robot flange as the previous two, and the robot forces are measured using the robot's inverse kinematics, which causes some issues with the stability of the values. To address this, some filtering was implemented, with the logic that the robot only moves if the robot controller verifies the crossing of a threshold n times.

7.2.5. Programming global considerations

Having the programming modes described and how they work, there are some aspects to clarify:

- In all programming modes, the robot is attached to a tool, that is the adapter for the robot flange reviewed in sub-section 3.3 that was taken into consideration while calibrating the robot;
- It was created a work object in the programming that represents the beam with a certain weight, depending on the test. Table 7.1 shows the values of the weight, measured with the dynamometer SAUTER FK250 when 0 N, 25.7 N, and 51.4 N are additionally applied to the beam. This measurement was the mean value of three tests with the beam in the zero degrees position;

Table 7.1 - Workobject weight, robot programming.

Workobject weight - programming	Value
Additional load – 0 N	4.8 N
Additional load – 25.7 N	31.5 N
Additional load – 51.4 N	58 N

- The Robot Impedance Control mode does not allow to block the displacement of the robot in cartesian directions. That being said, in the XZ plane displacement tests this robot mode will operate with no restrictions on the cartesian axis.

- In the case of the other three programming modes, the robot can block the Y direction when only the XZ plane displacement is solicited. This can be made using a high value of stiffness in the direction that the block is wanted.
- The values of stiffness reached for the Z-axis were an iterative process where the overshoot, visual stability of the mechanism, and smoothness of the mechanism deployment were analyzed.
- The constant multiplier for the torque sensor differential is also determined experimentally. The first iteration was created by calculating the displacement of the beam at its end at the required velocity and then changing it to account for the threshold. The goal was to never let the robot enter the dead zone after it started moving. As a result, the controller is constantly correcting the movement based on the torque differential, resulting in fewer oscillations and unwanted torques that could interfere with the proper operation of the mechanism.

8. Experiments

The validation of the system requires testing the different robot programming modes. A test plan was created to build different operating scenarios where the valences of the system are tested.

8.1. Test Plan

Table 8.1 - Test plan matrix.

	Control Load	RIC	RIC_TS	RIC_TS_E	RIC_TS_RF
Setup 0	<i>No offloading</i>	n/a	n/a	n/a	n/a
Setup A	<i>XZ – 0 N</i>	1	3	10	12
	<i>XZ – 25.7 N</i>	2	4	11	13
	<i>XZ – 51.4 N</i>	n/a	5	n/a	14
Setup B	<i>XYZ – 0 N</i>	n/a	n/a	n/a	n/a
	<i>XYZ – 25.7 N</i>	n/a	6	n/a	15
	<i>XYZ – 51.4 N</i>	n/a	7	n/a	16
Setup C	<i>A XYZ – 0 N</i>	n/a	n/a	n/a	n/a
	<i>A XYZ – 25.7 N</i>	n/a	8	n/a	17
	<i>A XYZ – 51.4 N</i>	n/a	9	n/a	18

The testing plan is divided into four setups, and each setup was performed with three loads:

- Load: 0 N –only Beam was deployed, no extra weight was added;
- Load: 25.7 N – Beam plus 1 weight of 25.7 N;
- Load: 51.4 N – Beam plus 2 weights of 25.7 N each;

To allow a more easy interpretation of the test results, a number was given to each test. For example, test 1 was performed under the setup A configuration, with 0 N extra force under RIC control program.

Basically, each test can be read as, for example, test 5 (Setup A, 51.4 N, RIC_TS)

8.1.1. Setup 0:

Initial setup with no connection to the robot. Evaluates the free deployment of the beam.

8.1.2. Setup A:

Setup A performed tests in the XZ plane. The configuration of the robot/beam is shown in Figure 8.1. The principal objective of the configuration was to see how the robot manages two degrees of freedom displacement with different loads. The X and Z axis of the two referential, flange and breadboard, build a plane where the deployment was made.

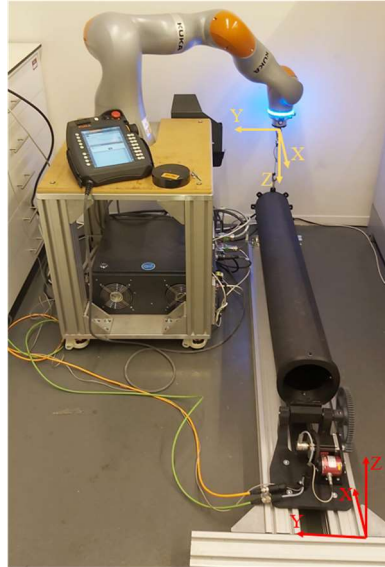


Figure 8.1 - Setup A test configuration.

8.1.3. Setup B:

The current setup was used to test the solution when an X, Y, and Z displacement is requested to the robot during the offloading deployment. In Figure 8.2 can be observed the disposal of the two referential.

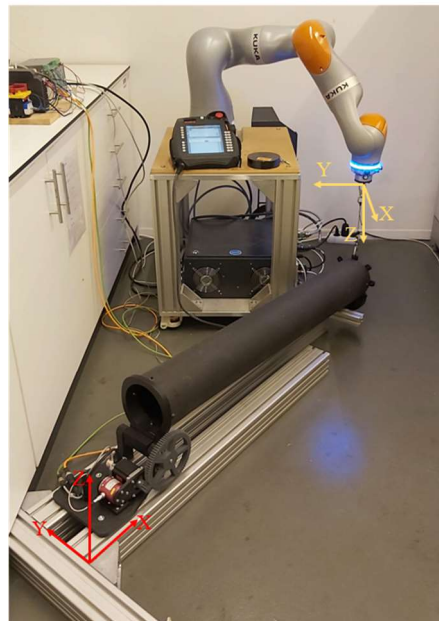


Figure 8.2 - Setup B test configuration.

8.1.4. Setup C:

Setup C was developed to build a complex trajectory where there are no axis aligned between the flange and the setup referential. This setup (Figure 8.3) creates a scenario much more difficult to predict where the torque sensor was not aligned with the offloading direction.

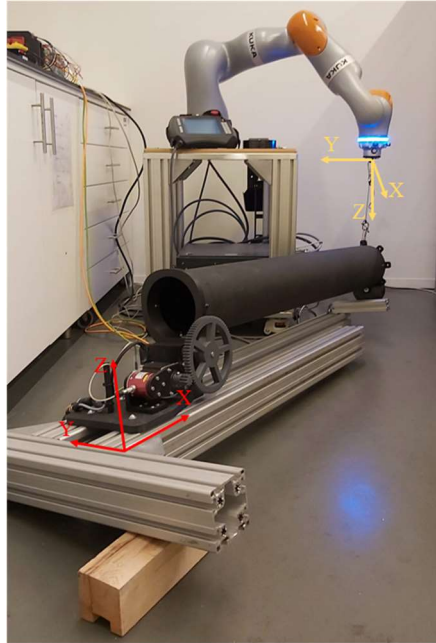


Figure 8.3 - Setup C test configuration.

8.2. Setup 0 - Results

The torque results for setup 0 are presented in Figure 8.4.

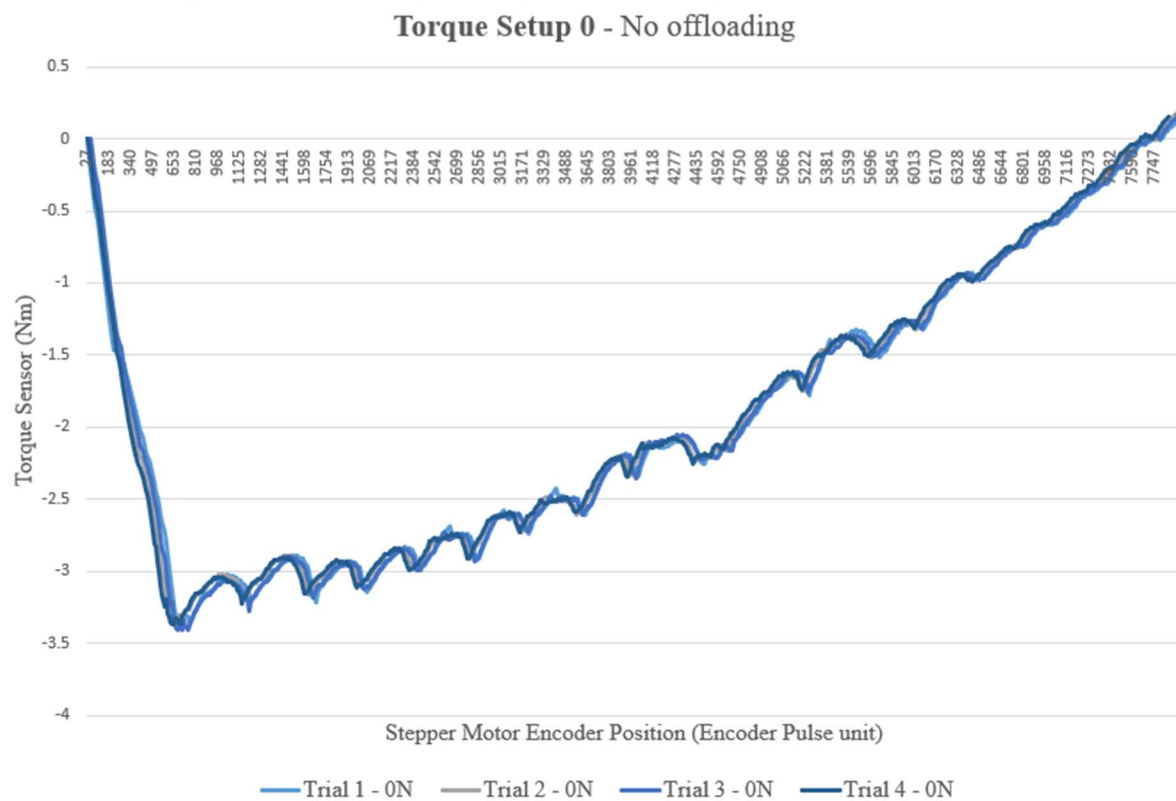


Figure 8.4 - Torque setup 0, no offloading.

8.3. Setup A - Results

The torque results for setup A for the different control strategies and additional load configurations are here presented.

- Robot Impedance Control**

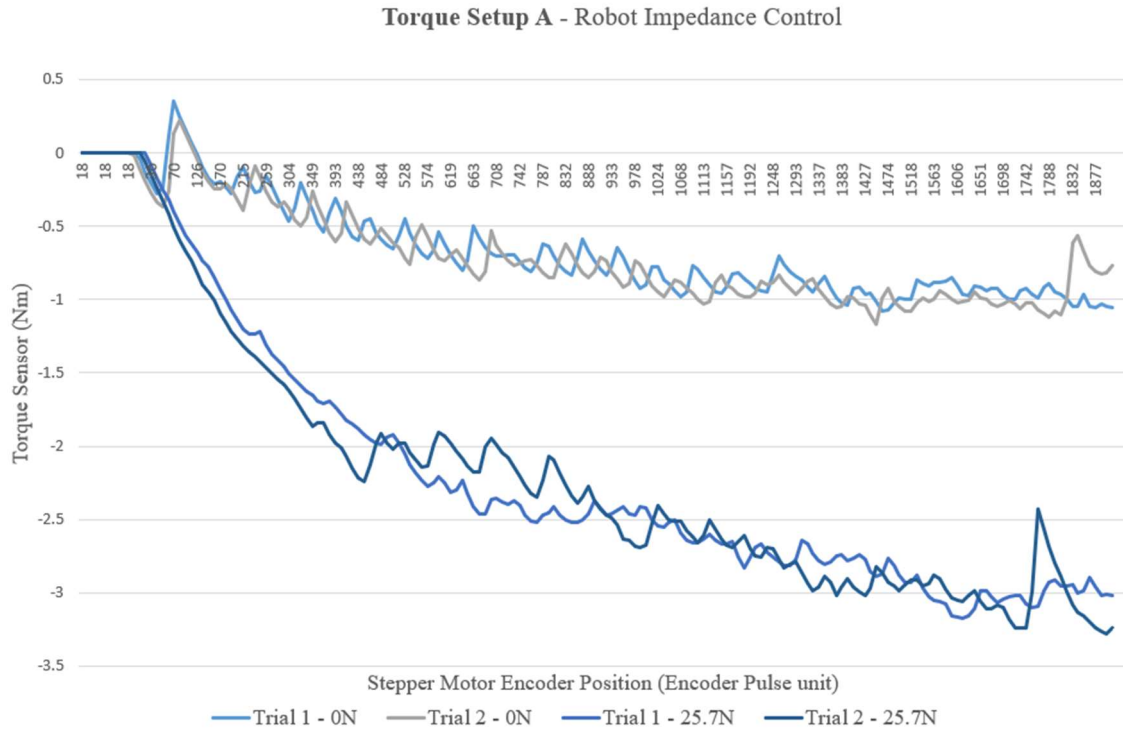


Figure 8.5 - Torque setup A, robot impedance control.

- Robot Impedance Control plus Torque Sensor**

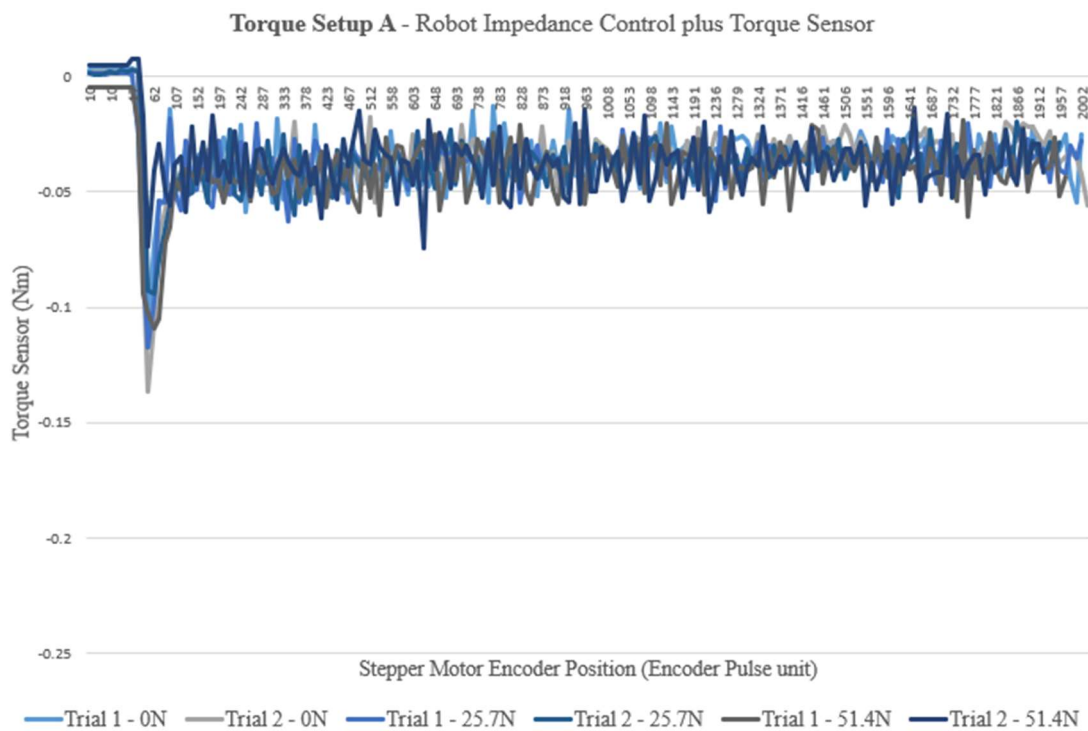


Figure 8.6 - Torque setup A, robot impedance control plus torque sensor.

- **Robot Impedance Control plus Torque Sensor plus Encoder**

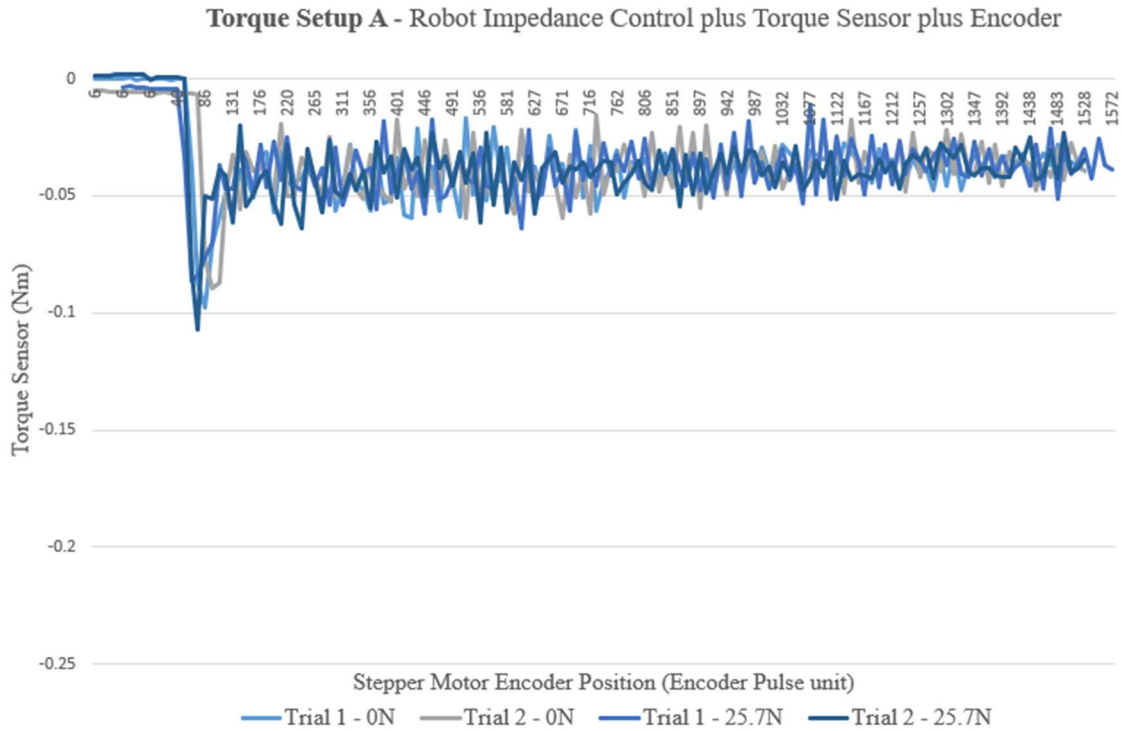


Figure 8.7 - Torque setup A, robot impedance control plus torque sensor plus encoder.

- **Robot Impedance Control plus Torque Sensor plus Robot Forces**

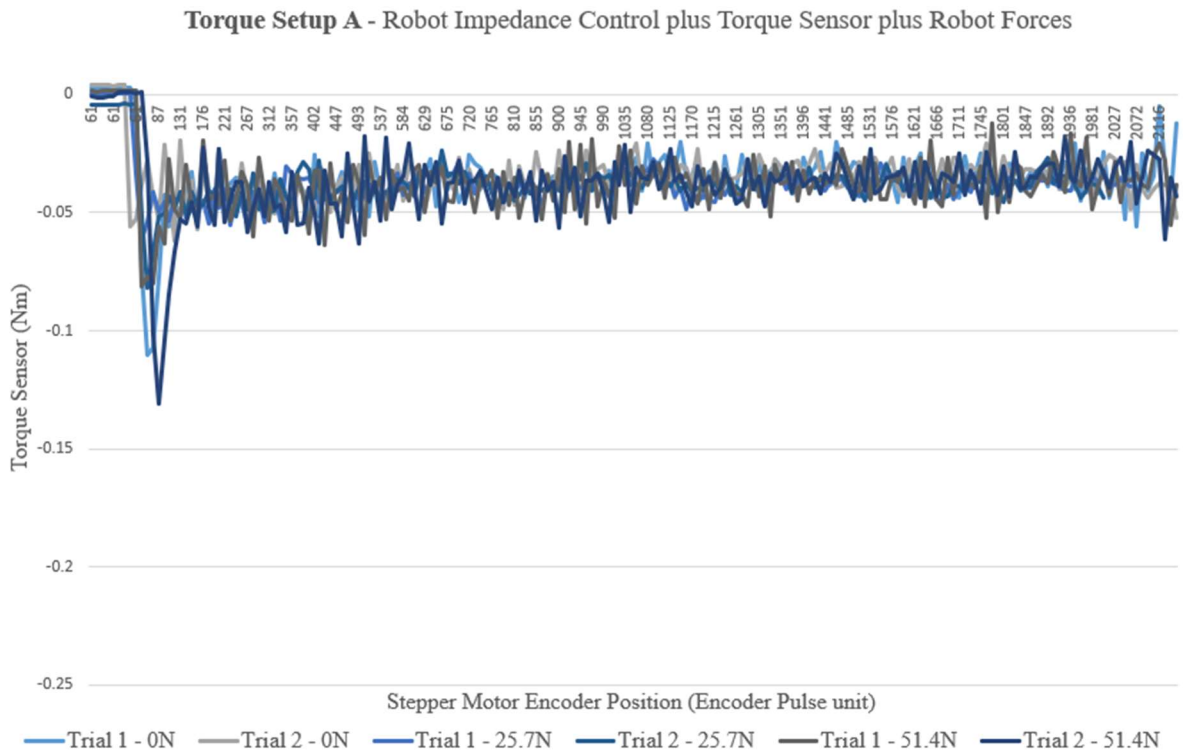


Figure 8.8 - Torque setup A, robot impedance control plus torque sensor plus robot forces.

8.4. Setup B - Results

The torque results for setup B for the different control strategies and additional load configurations are here presented.

- **Robot Impedance Control plus Torque Sensor**

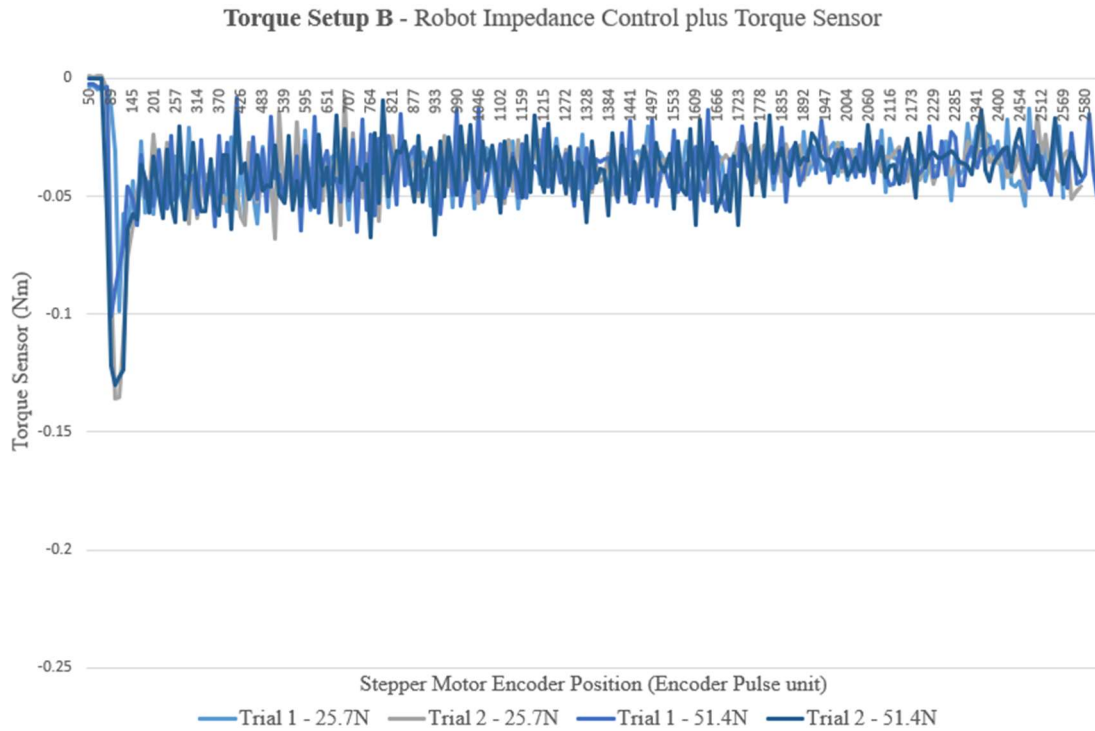


Figure 8.9 - Torque setup B, robot impedance control plus torque sensor.

- **Robot Impedance Control plus Torque Sensor plus Robot Forces**

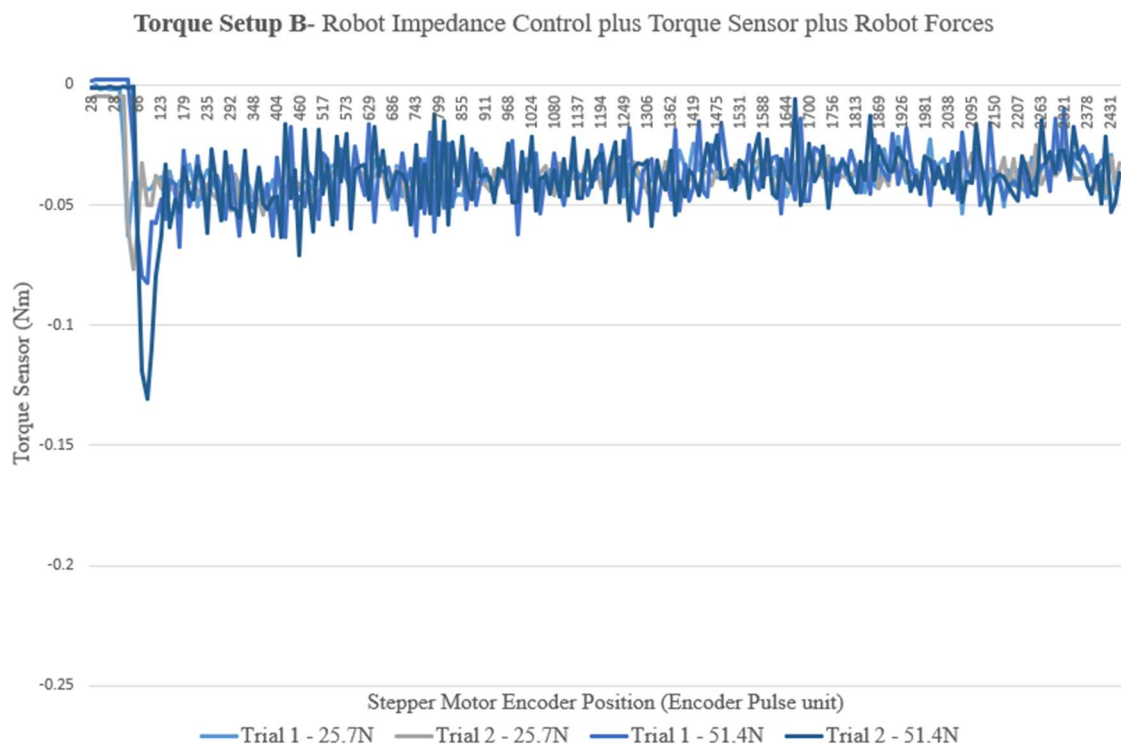


Figure 8.10 - Torque setup B, robot impedance control plus torque sensor plus robot forces.

8.5. Setup C - Results

The torque results for setup C for the different control strategies and additional load configurations are here presented.

- Robot Impedance Control plus Torque Sensor**

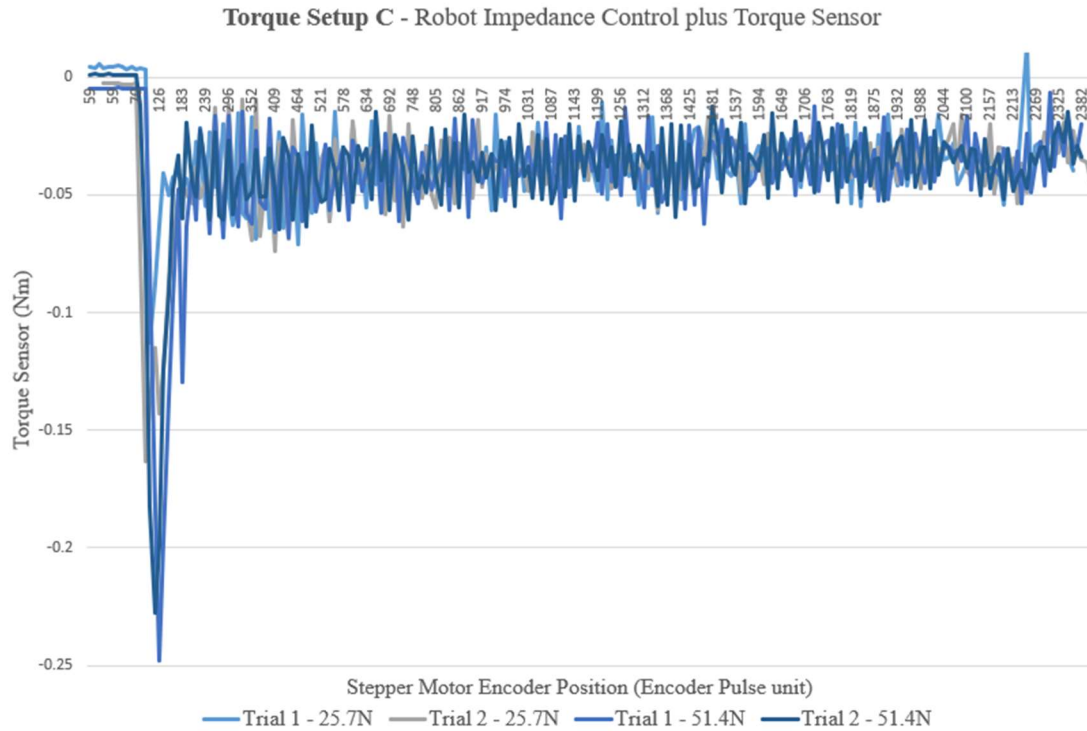


Figure 8.11 - Torque sensor C, robot impedance control plus torque sensor.

- Robot Impedance Control plus Torque Sensor plus Robot Forces**

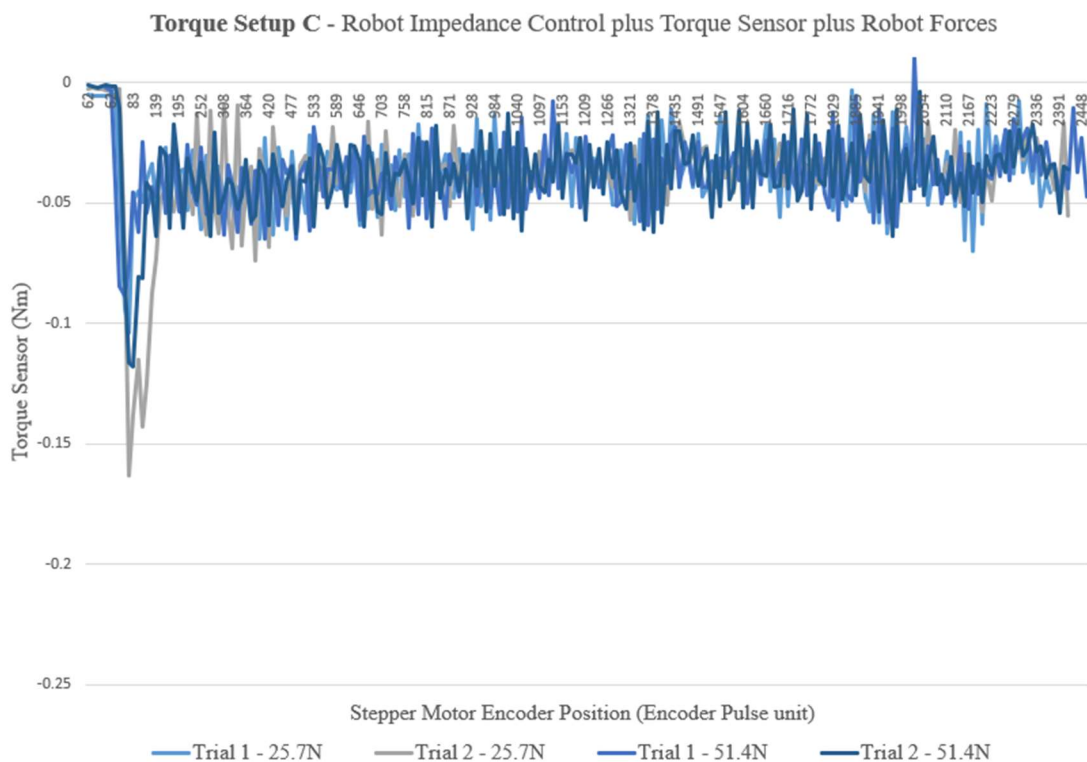


Figure 8.12 - Torque setup C, robot impedance control plus torque sensor plus robot forces.

8.6. Setup 0 results discussion

The set of tests carried out with setup 0 enables evaluation of how the torque sensor behaves during deployment. One important characteristic of the mechanism was its repeatability, as can be seen, in Figure 8.4, sub-section 8.2, The differences in torque values were imperceptible after four series of tests.

Table 8.2 presents the maximum absolute value of the torque obtained in the set of the tests.

Table 8.2 - Setup 0 torque results.

Setup 0 - Characteristic	Value
Maximum absolute values of torque	3.372 Nm

8.7. Setup A results discussion

Setup A represents the first configuration tested with the different programming approaches.

8.7.1. Robot Impedance Control

The first robot mode tested was the Robot Impedance Control, corresponding to tests 1 and 2. The torque curves obtained, in Figure 8.5, sub-section 8.3 show that the absolute value of torque has a growing trend in either load configuration. Table 8.3 shows the maximum absolute values for each testing load.

Table 8.3 - Setup A torque results, RIC.

Setup A - Characteristic Torque – RIC	Value
Maximum absolute – Test 1 0N	1.172 Nm
Maximum absolute – Test 2 25.7N	3.283 Nm

Even though the robot is aware of the mass increment, the additional load applied to the beam causes an offset in the value of the torque. There were no further tests with this programming mode because the torque values obtained were higher than expected, and the torque sensor could pass its overflow value on the setup, around 3.5 Nm, with the load of 51.4 N, if the growing trend between loads continues.

8.7.2. Robot Impedance Control plus Torque Sensor

Robot Impedance Control plus Torque Sensor was the second mode tested, now using a close loop system. The values of the torque sensor (Figure 8.6, sub-section 8.3) show more stable measurements with a much lower value of torque. There is an initial overshoot due to the movement start of the robot. Table 8.4 resumes the principal characteristics of the torque values obtained after the initial overshoots. The external loads do not seem to add an influence on the torque results, so the results of tests 3, 4, and 5 were jointly treated.

Table 8.4 - Setup A torque results, RIC_TS

Setup A - Characteristic Torque – RIC TS	Value
Maximum absolute	0.074 Nm
Average	-0.037 Nm
Standard deviation	0.090

The absolute torque values increased during the deployment, recalling the initial mode tested (Robot Impedance Control). The force delivered by the robot on the Z-axis was constant and predetermined by the user. If the torque values in the Robot Impedance Control plus Torque Sensor mode remain essentially constant during the deployment, the Z forces on the robot must be increased to counterbalance the effect. Figure 8.13 shows the forces measured in the robot flange in both cases, acknowledging the situation.

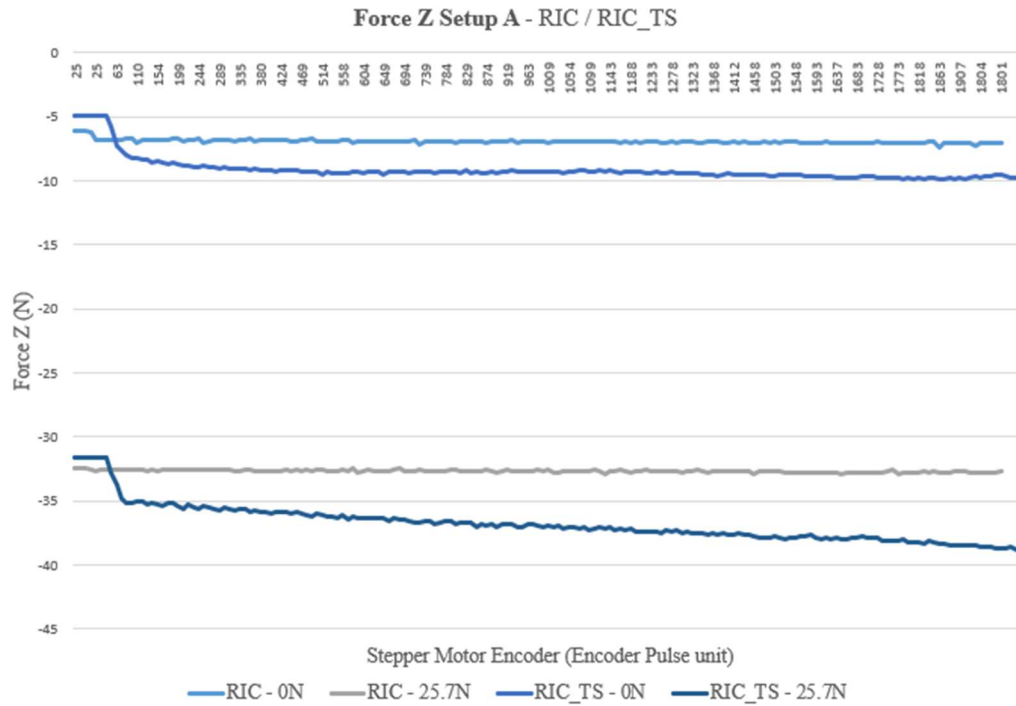


Figure 8.13 - Force Z setup A, RIC / RIC_TS.

Two hypotheses could be an explanation for this behavior:

The first implies that the distance between the beam center of mass and the point of application of the robot force does not remain constant during deployment. In Figure 8.14 it can be observed using a simplified model where the mass of the beam is neglectable and the center of the mass is the center of the additional weight.

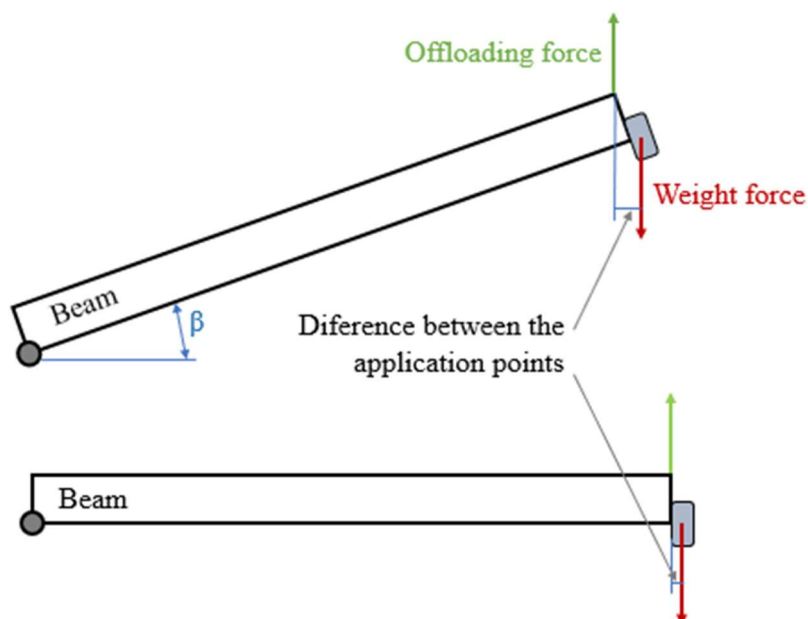


Figure 8.14 - Application point/center of mass displacement on the model.

As the deployment angle β (Figure 8.14) increases, the separation of the two centers of mass is more accentuated, which causes the appearance of additional torque in the hinge.

The second theory for the Z-axis behavior is that the robot is unable to follow the deployment movement and must raise the Z-axis force to compensate for displacements in the cable connecting the robot to the beam during no-additional load tests (0 N). Figure 8.15 shows the cable disposal at the beginning of the test, a), and at the end, b).

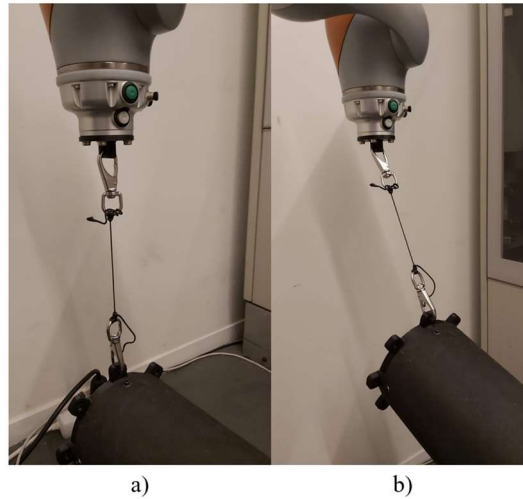


Figure 8.15 – RIC_TS mode cable inclination, a) deployment beginning, b) deployment end.

Is possible to conclude that there is a lot of displacement, which could cause measuring errors and undesirable forces in the no-offloading directions. The robot flange moves when F_x (Figure 8.16) reaches a certain value and the robot controller assumes there is an external force applied to the flange. F_x only depends on two factors: the value of F and the displacement of the cable, given by the length of the F_x vector. In theory, if the value of F rises, the displacement of the cable decreases because the value of the force to make the robot move in the F_x direction is the same.

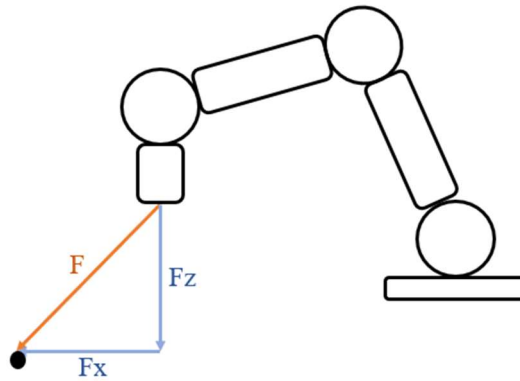


Figure 8.16 - Robot flange external forces schema.

Attaching loads to the beam end interface, the robot responds as it was planned, and the displacement of the cable was almost imperceptible, validating the usage of this programming mode for additional loads greater than 25.7 N

8.7.3. Robot Impedance Control plus Torque Sensor plus Encoder

Since the mode analyzed in the sub-section before is not perfectly adjusted for mechanisms with low load, the addition of the encoder could provide a better response for the robot when it comes to following the trajectory. In terms of the values of the torque, in Figure 8.7, sub-section 8.3, there is practically no difference compared with the previous mode, Table 8.5. Notice that the values of the table are calculated after the initial overshoot of the torque.

Table 8.5 - Setup A torque results, RIC_TS_E.

Setup A - Characteristic Torque – RIC_TS_E	Value
Maximum absolute	0.0594 Nm
Average	-0.039 Nm
Standard deviation	0.0095

The difference presents itself in a smaller displacement of the cable concluding the deployment with no load, performing test 10. Figure 8.17 shows the cable inclination at the beginning of the test, a), and at the end, b).

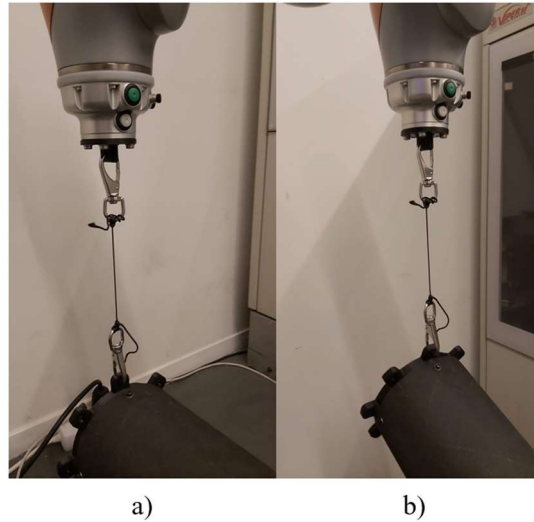


Figure 8.17 -- RIC_TS_E mode cable inclination, a) deployment beginning, b) deployment end.

Despite these improved results, it was not able to proceed with the other testing configurations since they needed a perfect measurement of the beam angle in every direction associated with the flange referential, which could only be accomplished with external measuring equipment (laser radar or laser tracker). The mode was created to mitigate the errors of displacement for a low load mechanism but has no major advantage compared to the previous mode analyzed, Robot Impedance Control plus Torque Sensor. Tests 4 and 11 present, visually, identical displacements.

8.7.4. Robot Impedance Control plus Torque Sensor plus Robot Forces

Torque sensor results of this programming mode (Figure 8.8, sub-section 8.3) on tests 12, 13, and 14 show values slightly stable compared with the programming mode Robot Impedance Control plus Torque Sensor. The standard deviation was calculated, in Table 8.6, and comparing it with the one obtained before, only in torque sensor mode, the standard deviation of the Robot Impedance Control plus Torque Sensor plus Robot Forces mode is lower.

Table 8.6 - Setup A torque results, RIC_TS_RF.

Setup A - Characteristic Torque – RIC_TS_RF	Value
Maximum absolute	0.063 Nm
Average	-0.037 Nm
Standard deviation	0.0081

This mode suffers, too, from large cable displacements operating with no load attached, being this issue corrected when additional weight is applied. On the other hand, the measured forces on the flange on the X-axis are more unstable.

Figure 8.18 shows the difference between the robot measured forces with the Robot Impedance Control plus Torque Sensor plus Robot Forces mode and the Robot Impedance Control plus Torque Sensor.

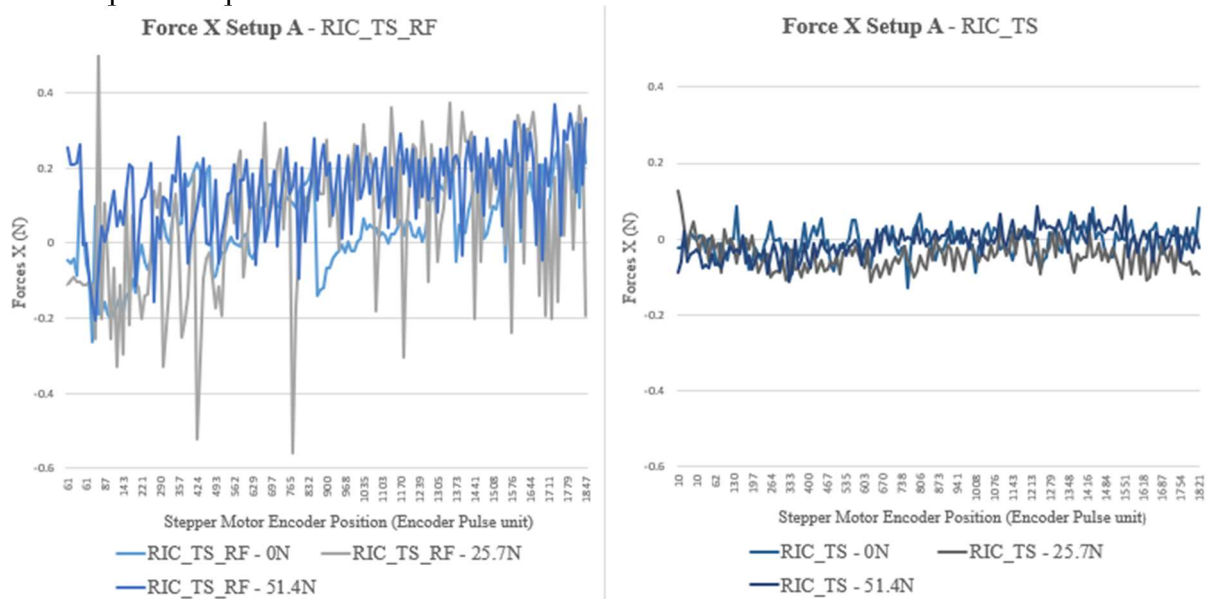


Figure 8.18 – Setup A, forces X analysis.

Table 8.7 sums up some characteristics of the X-axis force measured in the robot flange.

Table 8.7 - Setup A forces X results – RIC_TS_RF / RIC_TS.

Setup A - Characteristic X-axis Force	RIC_TS_RF	RIC_TS
Maximum absolute	0.56 N	0.13 N
Average	0.085 N	-0.021 N
Standard deviation	0.148	0.040

Comparing the values, the X forces on the Robot Impedance Control plus Torque Sensor are much more stable compared with the mode using the robot forces as input.

8.8. Setup B results discussion

Setup B represents the second configuration tested with the different programming approaches. Due to the similarity of the results in both programming modes, they were analyzed together. Table 8.8 shows the principal characteristics of the results of the torque sensor after the initial overshoots for the Robot Impedance Control plus Torque Sensor plus Robot Forces mode, tests 15 and 16 (Figure 8.10, sub-section 8.4) and for Robot Impedance Control plus Torque Sensor, tests 6 and 7 (Figure 8.9, sub-section 8.4).

Table 8.8 - Setup B torque results, RIC_TS_RF / RIC_TS.

Setup B - Characteristic Torque	RIC_TS_RF	RIC_TS
Maximum absolute	0.070 Nm	0.068 Nm
Average	-0.038 Nm	-0.038 Nm
Standard deviation	0.0087	0.0104

This set of tests requires a two-dimensional adjustment of the robot flange in no-offloading directions, in this case, in the X and Y-axis. The Z remains assigned to the offloading

direction. It can be observed in Figure 8.19 and Figure 8.20 the graphics obtained for the two programming modes for the direction of the X and Y axis, respectively.

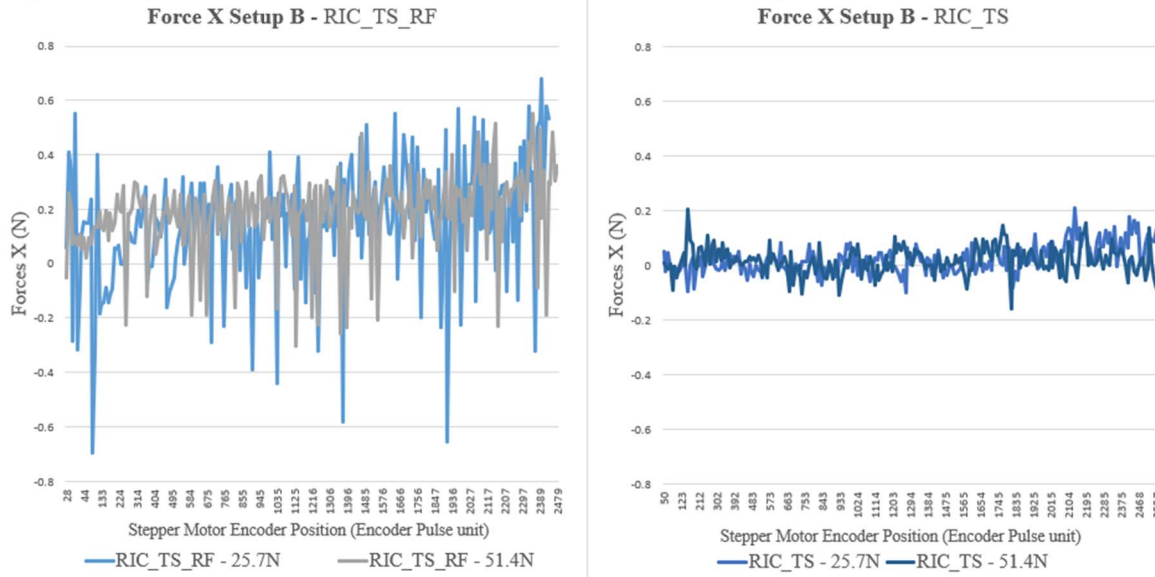


Figure 8.19 - Setup B, forces X analysis.

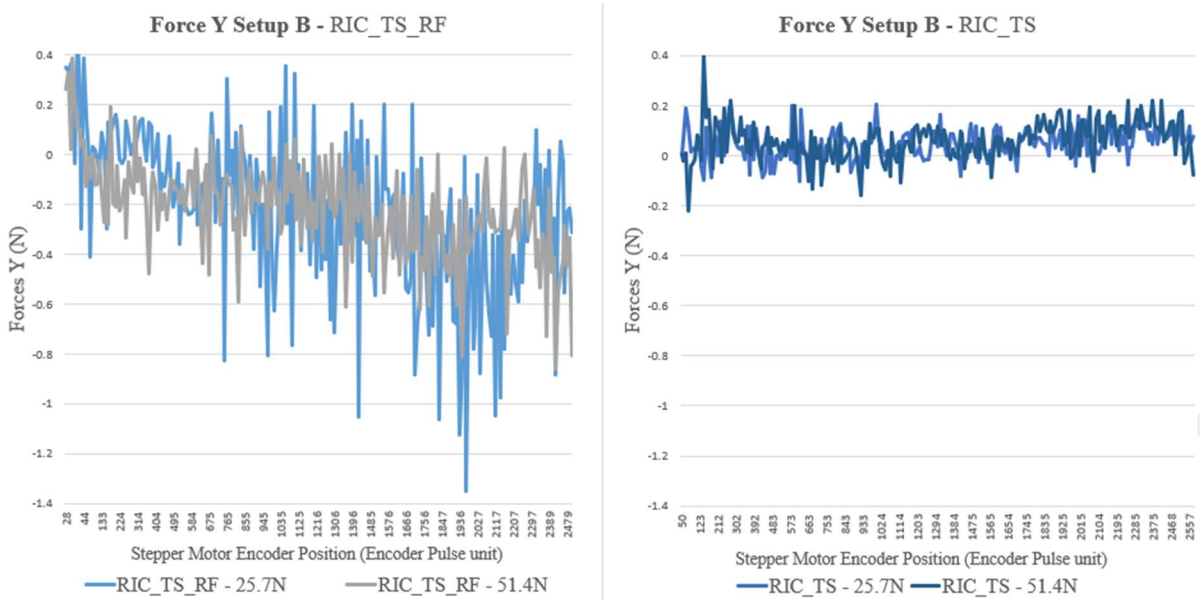


Figure 8.20 - Setup B, forces Y analysis

The mode Robot Impedance Control plus Torque Sensor plus Robot Forces mode presents a more unstable behavior, worst that the same mode but only working in one no-offloading direction. As seen in the programming sub-section 7.2, the inaccuracy of the robot force calculation causes some problems, which could be the case here since the robot is performing under user input movements in the axis that is being force monitored. However, is a significant improvement in the stability of the robot force control with the higher load attached. Table 8.9 shows that the standard deviation is lower when the robot is attached to a higher load.

Table 8.9 – Setup B force's standard deviation, RIC_TS_RF.

Standard Deviation	25.7 N	51.4 N
Force X	0.223	0.147
Force Y	0.312	0.192

8.9. Setup C results discussion

As happened in the previous setup, the results of the two programming modes analyzed are very similar. Table 8.10 shows the principal characteristics of the results of the torque sensor after the initial overshoots for the Robot Impedance Control plus Torque Sensor plus Robot Forces mode, tests 17 and 18 (Figure 8.12, sub-section 8.5) and for Robot Impedance Control plus Torque Sensor, tests 8 and 9 (Figure 8.11, sub-section 8.5).

Table 8.10 - Setup C torque results, RIC_TS_RF / RIC_TS.

Setup C - Characteristic Torque	RIC_TS_RF	RIC_TS
Maximum absolute	0.074 Nm	0.071 Nm
Average	-0.038 Nm	-0.037 Nm
Standard deviation	0.0126	0.0121

By data analysis, the torque values for setup C are a little more unstable than the results for setup B. The maximum absolute has increased due to the position of the setup. The principal movement direction is no longer aligned with the Z-axis of the flange. So the adjustments of the XY plane are much more requested, causing more oscillation.

The torque values of the overshoot of the Robot Impedance Control plus Torque Sensor have increased, creating a more visually bumpy start but stabilizing during the first seconds of deployment.

Figure 8.21 and Figure 8.22 show the results for the forces in the X and Y axis for the two programming modes, respectively.

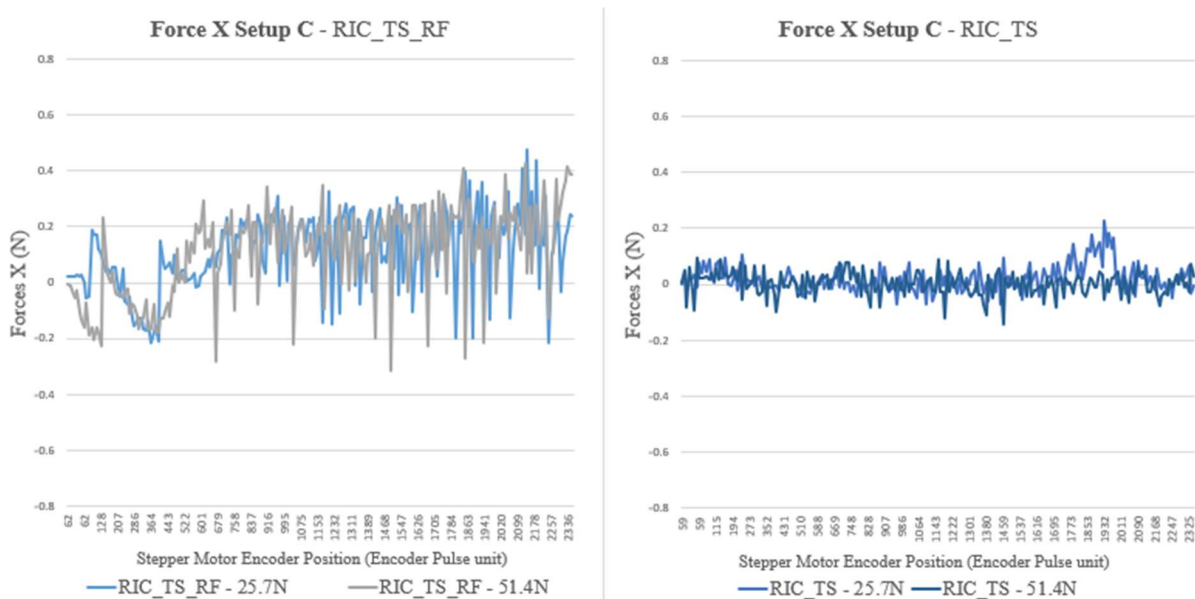


Figure 8.21 - Setup C, forces X analysis.

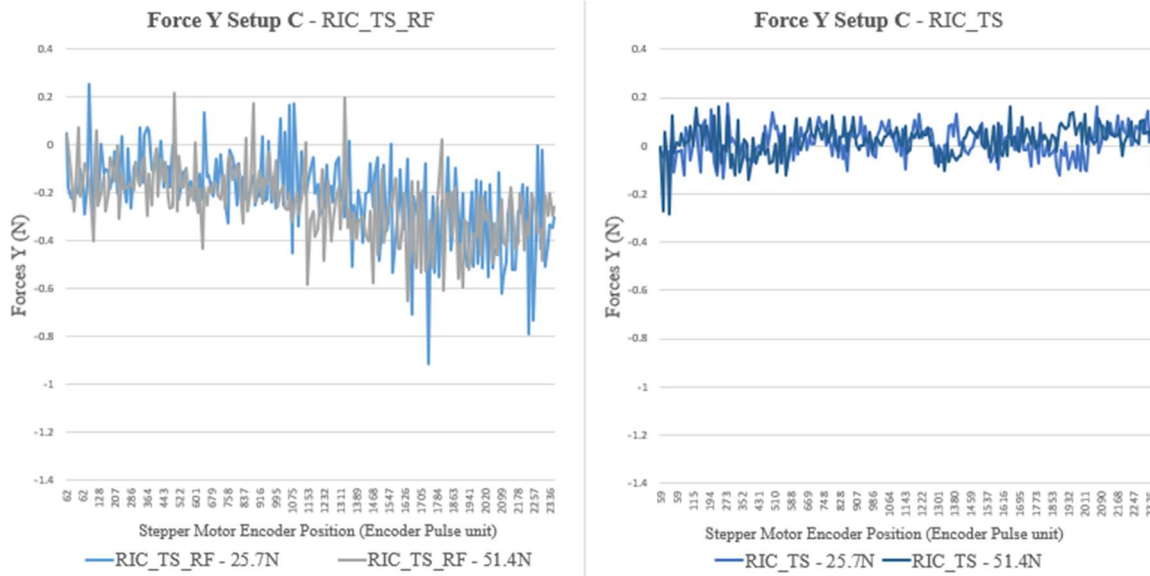


Figure 8.22 - Setup C, forces Y analysis.

The Robot Impedance Control plus Torque Sensor plus Robot Forces mode presents much better results for the forces in X and Y on this setup than compared with setup B which was unexpected having into consideration the complexity of the trajectory.

8.10. Offloading global Discussion

Equations 8.1, 8.2, and 8.3 use the inverse kinematics of the beam and some of the considerations used for the breadboard development, (sub-section 3.4) to calculate the maximum torque allowed in the torque sensor to meet OFF-4, parasite force lower than 0.2 N. This value of force is applied at the end of the breadboard beam.

$$FN2 = (1.21 * 0.2) / 0.08 = 2.967 \text{ N} \quad (8.1)$$

$$FN2_Y = 2.967 * \cos(40^\circ) = 2.272 \text{ N} \quad (8.2)$$

$$Mo_{Torque\ Sensor} = (2.272 * (0.06 + 0.01)) = 0.159 \text{ Nm} \quad (8.3)$$

The maximum value of torque admissible on tests to match the requirement was 0.159 Nm. Table 8.11 shows the torque values obtained before for each programming mode. Also indicating what are the tests that perform according to the requirement, green cells, and the ones that do not, red cells.

Table 8.11 - Test plan, global results.

		RIC	RIC TS	RIC TS E	RIC TS RF
Setup 0	3.372 Nm	n/a	n/a	n/a	n/a
Setup A	<i>XZ – 0N</i>	-1.172 Nm	-0.037 Nm	-0.039 Nm	-0.037 Nm
	<i>XZ – 25.7N</i>	-3.283 Nm	-0.037 Nm	-0.039 Nm	-0.037 Nm
	<i>XZ – 51.4N</i>	n/a	-0.037 Nm	n/a	-0.037 Nm
Setup B	<i>XYZ – 0N</i>	n/a	n/a	n/a	n/a
	<i>XYZ – 25.7N</i>	n/a	-0.038 Nm	n/a	-0.038 Nm
	<i>XYZ – 51.4N</i>	n/a	-0.038 Nm	n/a	-0.038 Nm
Setup C	<i>A XYZ – 0N</i>	n/a	n/a	n/a	n/a
	<i>A XYZ – 25.7N</i>	n/a	-0.037 Nm	n/a	-0.038 Nm
	<i>A XYZ – 51.4N</i>	n/a	-0.037 Nm	n/a	-0.038 Nm

According to Table 8.11, the best modes to operate the robot for gravity compensation are the Robot Impedance Control plus Torque Sensor and the Robot Impedance Control plus Torque Sensor plus Robot Forces once they can operate in the three testing setups with good results and match the OPER-2 complex trajectory requirement.

8.11. Results Discussion - Accuracy and Uncertainties

Having done all the testing procedures, it was imperative to evaluate the accuracy of the results to validate all the work performed.

The results obtained were quite positive however it has some inaccuracies associated:

8.11.1. Hinge rigidity issue

During tests, was observed that the structural rigidity of the mechanism was not as good as expected. When the mechanism was performed with no offloading, it was identified undesired displacements of the hinge mechanism. There were three parts (Figure 8.23) selected as being the most problematic since they directly interfere with the measurements of the torque sensor in particular the base of the sensor that must be as stable as possible.

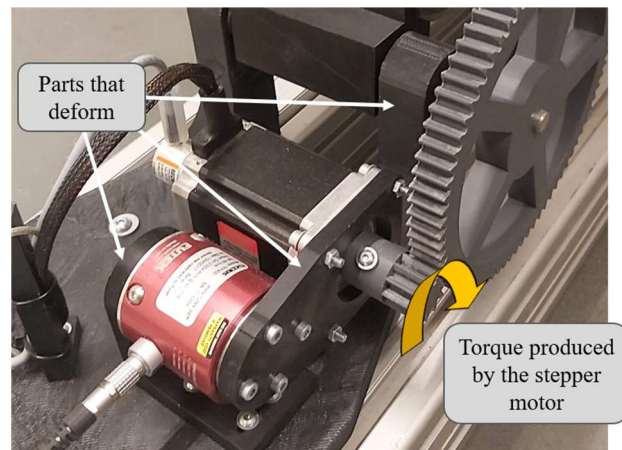


Figure 8.23 - Hinge rigidity analysis.

8.11.2. Instrumentation Accuracy

The measurements using the dynamometer SAUTER FK250 could have some uncertainty in the results because it has a resolution of 0.1 N. Looking for the magnitude of the measurements, small variations in the dynamometer values cause differences in the results.

In the case of the robot, is not possible to ensure that the measurement of the forces is as accurate as it should be. The accuracy of the calculated external force depends on the dynamics of the robot motion and of the actual force, among other things.

8.11.3. Theoretical/Experimental Accuracy evaluation

The accuracy evaluation was into account the error between the theoretical and the experimental values of torque.

Setup 0, sub-section 8.2, was weighted with the dynamometer SAUTER FK250 in the zero degrees position at the same application point that the robot uses, reaching the value of 3.7 N.

Notice that the force measured with the dynamometer is applied to the beam end and the flange adapters create an offset of 3 cm relative to the previous analysis performed in the breadboard development.

To obtain the theoretical value of torque, it is used the same inverse cinematics equations used before. Applying them it achieved the value of 3.00 Nm.

The experimental mean maximum torque value obtained in the four series is 3.37 Nm, representing a relative error of (Equation 8.4):

$$Error_{3.7N} = \frac{3.37-3.00}{3.00} = 12.39\%. \quad (8.4)$$

Two more sets of tests were performed (Appendix N) to assess the error under other configurations:

- Removing the end interface of the beam and adding one hook (half cable), decreasing the force of the dynamometer to 3.2 N and the application point to 1.20 m.
- Removing the end interface of the beam, decreasing the force of the dynamometer to 2.7 N and the application point to 1.20 m.

The results of the relative error are present in Table 8.12.

Table 8.12 - Theoretical/Experimental relative error.

Test	Relative Error
Load 3.7 N	12.39%
Load 3.2 N	9.74%
Load 2.7 N	9.50%

Considering all the previous analyses, it is possible to confirm the existence of an error between 10% to 15% on the torque sensor results. This evidence has to be taken into account when the critical analysis is made of the obtained results in this section.

8.12. Additional Test

To see if the order of magnitude of the values was identical in the different tests, it was performed a new test, a stop-and-go test. The robot was executing the Robot Impedance Control plus Torque Sensor mode and, after a while, it was forced a pause the execution. The robot stops the movement but the stepper motor continues working, loading the hinge and performing the ascendant movement of the beam. When the cable stops from being solicited, in other words, when the robot is no longer holding the beam, the execution is resumed, and the robot performs the movement to the actual offloading position.

To put it in another way, the deployment started with gravity compensation by the robot, and suddenly the compensation purposely shut down to see what happens to torque values. After a while, the gravity compensation was turned back on, and the robot immediately proceeded to offload the hanging weight.

In Figure 8.24 is possible to see four tests, two performing deployment without offloading and the other two performing the stop and go test. It is clear that the values of the four series, on the Offloading OFF area of the graphic, converge on the same order of torque, 1 Figure 8.24.

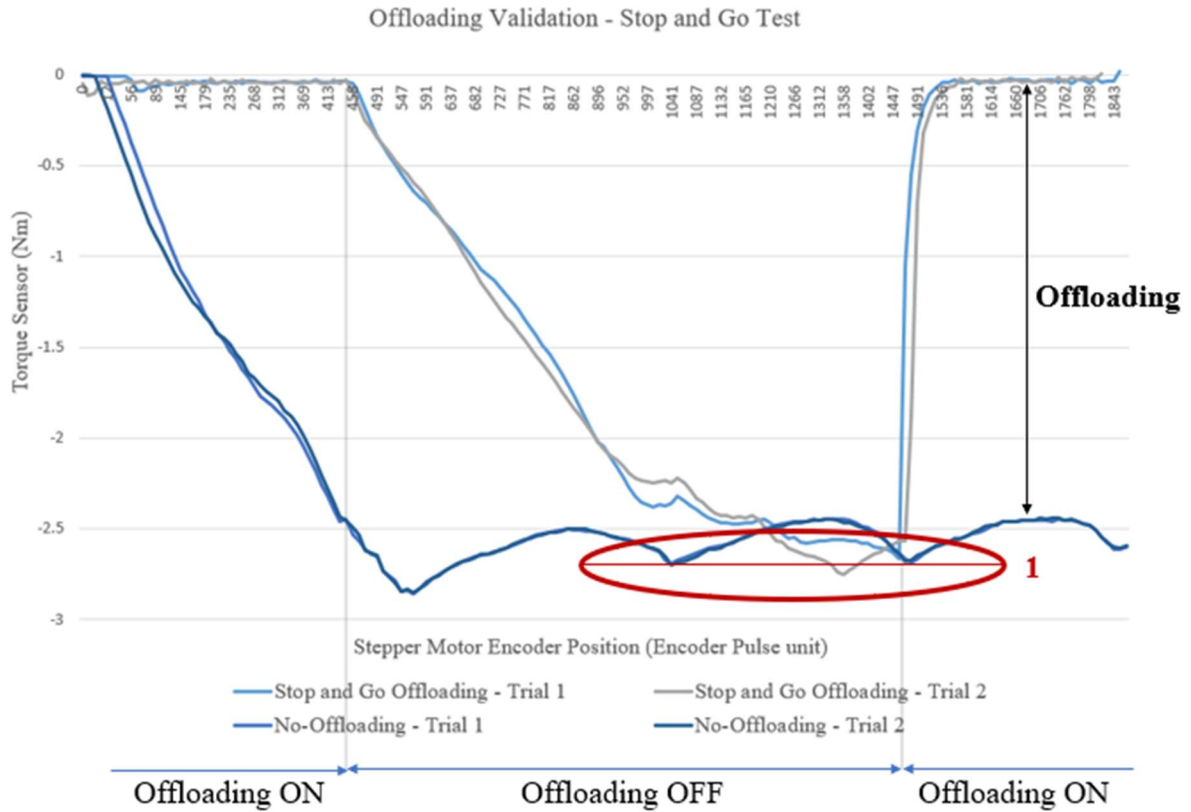


Figure 8.24 - Offloading validation, stop and go test.

The beam configuration used for the no-offloading testing was the same as for the offloaded tests. When the robot comes to a halt, the beam will support one of the cable's hooks. To imitate the circumstance, no offloading tests with one hook were added. The cable's mass was despised here.

The mean offloading values in the second Offloading ON area of the graphic are presented in Table 8.13.

Table 8.13 - Stop and go torque results.

Stop and Go – Torque Characteristic	No-offloading	Offloading
Average	-2.43 Nm	-0.035 Nm

Equation 8.5 calculates the offloading, percentage, of this stop-and-go test.

$$\text{Offloading} = \frac{-2.43 - (-0.035)}{-2.43} = 98.6\%. \quad (8.5)$$

9. Conclusions and Future works

The work performed validated the applicability of a collaborative robot on an active offloading device. The breadboard, which was designed and built using a genuine flying mechanism as an example, performed as expected for the application, generating the necessary interfaces for the stepper motor and torque sensor hardware integration, as well as robot and external masses connection support.

Beckhoff equipment used in the hardware configuration created a powerful interface between the Kuka IIWA, stepper motor, torque sensor, and external computer with the ability to control the stepper motor via robot controller, acquire analog signals from the torque sensor amplifier, and potentially exchange 32 bits variables between the robot controller and external computer.

The robot controller integration with Beckhoff equipment was a useful feature explored, creating a pathway of information exchange between any computer with a Run Time. The data acquisition and visualization acquired with the IoT platform, Node-RED, reveals itself fruitfully in the development stage because it was conceivable to analyze, in live time, the response of the system to the programming adjustments.

The test plan created, intending to validate the usage of the robot control with different disposals of the breadboard, was constituted by three setups, regarding two (Setup A) and three (Setup B) dimensional deployments of the breadboard with an additional Setup C, where the robot must follow a highly complex path (additional angle to the Setup B).

Defined the test plan, tests were performed on the different programming modes of the robot, each with different approaches to the control strategy in order to evaluate which was the best. The RIC mode was operating in the concept of not providing any input to the robot controller, performing movements under impedance mode. This approach does not allow to the robot obtain any feedback which, for this breadboard, was a problem because the robot's connection to the beam was not in the center of gravity of the mechanism, creating parasite forces in the hinge which resulted in the mode failing to meet the parasite force requirement. In the case of the RIC_TS, the torque sensor works as an input for the impedance control strategy, creating a close loop system that brings much better results for the parasite force. For no mass tests, the vertical condition of the cable was compromised, bringing to the table the RIC_TS_E mode that uses encoder position in the control providing better results for no-load deployments, but has no advantages for loaded ones once the RIC_TS mode ensures the cable verticality on them. Because it was difficult to adjust the RIC_TS_E mode for setups B and C, it was not possible to test the mode on these setups. The last mode tested was RIC_TS_RF which uses inbuilt robot instrumentation to calculate the external forces applied to the flange in the programming algorithm for better force calculation. The intent was to implement the future a two/three-dimensional force sensor in the setup. The hardware configuration (Beckhoff coupler) is already prepared to implement additional instrumentation.

The robot programming modes obtained permitted the characterization of the application using the different deployment setups. Using the inverse kinematics of the breadboard, the hinge maximum parasite torque was calculated to match the 0.2 N maximum parasite force, reaching the value of 0.159 Nm. This maximum parasite torque validates the

usage of RIC_TS and RIC_TS_RF modes to offload the breadboard developed in the three setups (A, B, and C) for the weights of 25.7 N and 51.4 N with an average value of 0.038 Nm of hinge parasite torque.

The torque values obtained are suitable but not perfect. Due to the lack of mechanical rigidity in the hinge and instrumentation issues, there are some inaccuracies and uncertainties, resulting in a 10% to 15% error between the theoretical and experimental torque values.

Concluding, this dissertation presents one innovative solution of gravity offloading with centralized control, based on closed-loop systems. The results were pretty good with a high level of interest in the aerospace sector, creating the baseline of knowledge required for potential ESA calls on a new active offloading device. So, the objective of the project was accomplished in reaching solutions that meet the requirements established in section 3, summing up the global status and observations in Table 9.1.

Table 9.1 - Offloading, functional, and operational requirements analysis.

Requirements		
	Status	Observations
OFF-1	Compliant	n/a
OFF-2	Compliant	n/a
OFF-3	Compliant	n/a
OFF-4	Compliant	Programming mode RIC does not meet the requirement;
FUNC-1	Compliant	n/a
FUNC-2	Compliant	n/a
FUNC-3	Compliant	n/a
FUNC-4	Compliant	n/a
FUNC-5	Partially Compliant	Due to wireless connection instability, the robot/breadboard control via tablet was not implemented (Safety Precautions);
OPER-1	Compliant	Compliant with the mass range specified in the requirement OFF-1 (Visual Inspection);
OPER-2	Compliant	Programming mode RIC_TS_E does not meet the requirement; Programming mode RIC was not tested;

Having in mind the improvement of the work performed, for future works is suggested:

- Improvement/rebuild of the hinge breadboard used. Upgrade the overall stiffness of the mechanism, trying to approximate the theoretical and experimental values.
- Using a three-dimensional force sensor (3-axis load cell) applied to the robot flange that could work as input in the Robot Impedance Control plus Torque Sensor plus Robot Forces mode instead of using the indirect X and Y force measurement provided by the robot controller. The Z-axis could also be monitored using this external force sensor.
- Use an external device (laser radar or laser tracker) to evaluate the cable displacement that could work as input for the robot control strategies.
- Deployment of the hinge plus beam in other test setups (horizontal for example) changing the robot control strategy.

References

- [1] M. Yang, Z. Xu, Y. He, Y. Liu, and B. Wang, “Zero gravity tracking system using constant tension suspension for a multidimensional framed structure space antenna,” in *Proceedings of 2016 7th International Conference on Mechanical and Aerospace Engineering, ICMAE 2016*, Aug. 2016, pp. 614–621. doi: 10.1109/ICMAE.2016.7549613.
- [2] R. Lopes, “Large Deployable Antenna - GSE’s Critical Design Review,” Porto, Apr. 2019.
- [3] R. Lopes, “Articulated Booms – Large Ultrastable Deployable Structures (ABDS) - GSE Detailed Design & Analysis,” Porto, Dec. 2017.
- [4] M. Loesch *et al.*, “European Large Deployable Reflector and Arm Developments as Pathfinder for Future Earth Observation Missions,” 2022.
- [5] Y. Fan, G. Liu, L. Liu, and C. Yang, “Design of Zero-gravity Unloading Process Equipment for Satellite Overlapping Antenna on Ground Test,” in *IOP Conference Series: Materials Science and Engineering*, Oct. 2019, vol. 611, no. 1. doi: 10.1088/1757-899X/611/1/012074.
- [6] J. F. Silva Gomes, *Mecânica dos Sólidos e Resistência dos Materiais*, Edições INEGI. 2004.
- [7] O. Han, D. Kienholz, P. Janzen, and S. Kidney, “Gravity-Offloading System for Large-Displacement Ground Testing of Spacecraft Mechanisms,” 2010.
- [8] ESA, IPC, and THAG, “Automation and Robotics,” Dec. 2021.
- [9] “ESA - Rover ExoMars.” https://www.esa.int/Space_in_Member_States/Portugal/Rover_ExoMars (accessed Jun. 21, 2022).
- [10] T. Tzschichholz, T. Boge, H. Benninghoff, and T. Tzschichholz, “Visual Navigation For On-Orbit Servicing Missions,” 2013. [Online]. Available: <https://www.researchgate.net/publication/259899819>
- [11] “ESA - GRALS Testbed.” https://www.esa.int/ESA_Multimedia/Images/2018/05/GRALS_Testbed (accessed Jun. 17, 2022).
- [12] S. D’Attanasio and J. Charron, “Development of a sensing system for zero gravity simulation using a robot manipulator to test antenna reflector unfolding in satellites,” in *Mechanisms and Machine Science*, 2019, vol. 66, pp. 41–48. doi: 10.1007/978-3-030-00365-4_6.
- [13] A. M. Lopes, “Modelação Cinemática e Dinâmica de Manipuladores de Estrutura em Série,” Porto, 2001.
- [14] Cobot Trends Staff, “Cobots vs. industrial robots: what are the differences?” <https://www.cobottrends.com/cobots-vs-industrial-robots-what-are-differences/> (accessed Jun. 17, 2022).
- [15] J. M. Buchert, “What is the difference between industrial robots and cobots?,” 2021. <https://manplasmachines.com/differences-robots-cobots/> (accessed Jun. 17, 2022).

- [16] Beckhoff, “AS1050 | stepper motor 1.2 Nm (M0), N2 (NEMA23/56 mm) | Beckhoff Worldwide.” <https://www.beckhoff.com/en-en/products/motion/compact-drive-technology/asxxxx-stepper-motors/as1050.html> (accessed Jun. 21, 2022).
- [17] Futek, “Reaction Torque Sensor TFF400 : FSH03986.” <https://www.futek.com/store/torque-sensors/reaction-torque-sensor/reaction-TFF400/FSH03986> (accessed Jun. 21, 2022).
- [18] Beckhoff, “EK1100 | EtherCAT Coupler | Beckhoff Worldwide.” <https://www.beckhoff.com/en-en/products/i-o/ethercat-terminals/ek1xxx-bk1xx0-ethercat-coupler/ek1100.html> (accessed Jun. 21, 2022).
- [19] Beckhoff, “EL7041 | EtherCAT Terminal, 1-channel motion interface, stepper motor, 48 V DC, 5 A, with incremental encoder | Beckhoff Worldwide.” <https://www.beckhoff.com/en-en/products/i-o/ethercat-terminals/el7xxx-compact-drive-technology/el7041.html> (accessed Jun. 21, 2022).
- [20] Phoenix Contact, “Power supply unit - QUINT-PS/1AC/48DC/20 - 2866695 | Phoenix Contact.” <https://www.phoenixcontact.com/en-il/products/power-supply-quint-ps1ac48dc20-2866695> (accessed Jun. 21, 2022).
- [21] Futek, “Strain Gage Amplifier | IAA100 Analog Strain Gauge Signal Conditioner.” <https://www.futek.com/store/instruments/analog-amplifier/strain-gauge-analog-amplifier-voltage-output-IAA100/FSH03863> (accessed Jun. 21, 2022).
- [22] Beckhoff, “EL3008 | EtherCAT Terminal, 8-channel analog input, voltage, ± 10 V, 12 bit, single-ended | Beckhoff Worldwide.” <https://www.beckhoff.com/en-en/products/i-o/ethercat-terminals/el3xxx-analog-input/el3008.html> (accessed Jun. 21, 2022).
- [23] Beckhoff, “EL6695 | EtherCAT Terminal, communication interface, EtherCAT bridge, extended functions | Beckhoff USA.” https://www.beckhoff.com/en-us/products/i-o/ethercat-terminals/el6xxx-communication/el6695.html?pk_campaign=AdWords-AdWordsSearch-EtherCat_dynamic_EN&pk_kwd=&gclid=Cj0KCQjw2MWVBhCQARIsAIjbwoOtk3yMrcWjBmCkGds76kKetjNi1huSHWvYS7IQP_y5rxinvY88SLUaAjcnEALw_wcB (accessed Jun. 21, 2022).
- [24] Beckhoff, “EL1008 | EtherCAT Terminal, 8-channel digital input, 24 V DC, 3 ms | Beckhoff Worldwide.” <https://www.beckhoff.com/en-en/products/i-o/ethercat-terminals/el1xxx-digital-input/el1008.html> (accessed Jun. 21, 2022).
- [25] “KUKA Sunrise.OS | KUKA AG.” <https://www.kuka.com/en-de/products/robot-systems/software/system-software/sunriseos> (accessed Jun. 17, 2022).
- [26] “KUKA.WorkVisual | KUKA AG.” https://www.kuka.com/en-de/products/robot-systems/software/system-software/kuka_systemsoftware/kuka_work-visual (accessed Jun. 17, 2022).
- [27] Beckhoff, “TwinCAT 3 | Getting Started.” https://download.beckhoff.com/download/document/catalog/TwinCAT_3_Booklet.pdf (accessed Jun. 21, 2022).
- [28] Node-RED, “Node-RED.” <https://nodered.org/> (accessed Jun. 21, 2022).
- [29] Node-RED, “node-red-contrib-ads-client - Node-RED.” <https://flows.nodered.org/node/node-red-contrib-ads-client> (accessed Jun. 21, 2022).
- [30] SQLite, “About SQLite.” <https://www.sqlite.org/about.html> (accessed Jun. 21, 2022).
- [31] Beckhoff, “Stepper Motor Velocity.” <https://infosys.beckhoff.com/english.php?content=../content/1033/ep7047-1032/8820689931.html&id=> (accessed Jun. 21, 2022).
- [32] H. F. N. Al-Shuka, S. Leonhardt, W. H. Zhu, R. Song, C. Ding, and Y. Li, “Active impedance control of bioinspired motion robotic manipulators: An overview,” *Applied Bionics and Biomechanics*, vol. 2018. Hindawi Limited, 2018. doi: 10.1155/2018/8203054.

- [33] N. Hogan, “Impedance Control: An Approach to Manipulation,” in *1984 American Control Conference*, Jul. 1984, pp. 304–313. doi: 10.23919/ACC.1984.4788393.
- [34] J. K. Salisbury, “Active Stiffness Control of a Manipulator in Cartesian Coordinates,” in *Proceedings of the IEEE Conference on Decision and Control*, 1980, vol. 1, pp. 95–100. doi: 10.1109/cdc.1980.272026.
- [35] L. Roveda, “Adaptive Interaction Controller for Compliant Robot Base Applications,” *IEEE Access*, vol. 7, pp. 6553–6561, 2019, doi: 10.1109/ACCESS.2018.2889849.
- [36] F. Caccavale, C. Natale, B. Siciliano, and L. Villani, “Six-DOF impedance control based on angle/axis representations,” *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 289–300, 1999, doi: 10.1109/70.760350.
- [37] RS Components, “Sauter FK 250 Force Gauge 1000Hz, Range: 250N, Resolution: 0.1 N | RS Components.” <https://uk.rs-online.com/web/p/force-gauges/0559767> (accessed Jun. 21, 2022).

Appendix A - Beckhoff AS1050-0120 datasheet

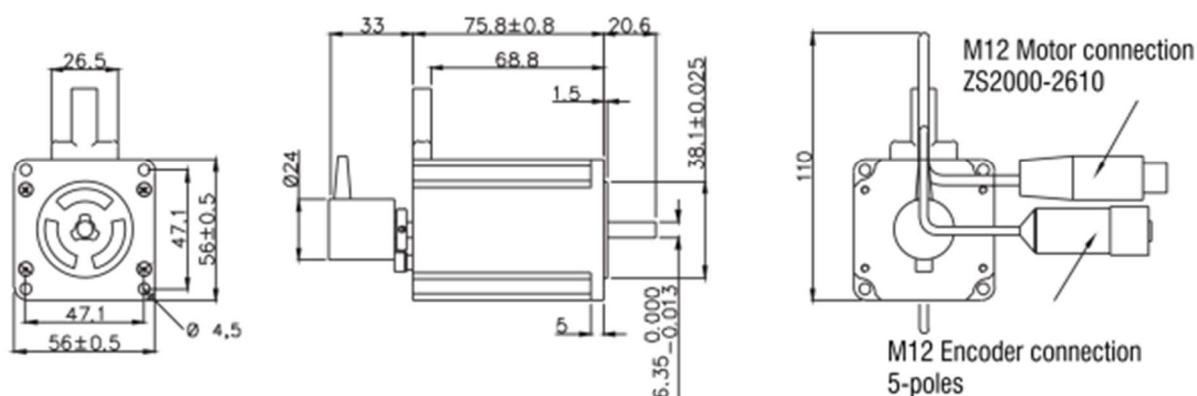
11.5 AS1050-0xxx

Technical data	Symbol [Unit]	AS1050
Rated supply voltage	V_{DC}	24 - 50
Rated current	A	5,0
Rated power	W	27,5
Standstill torque	M_0 [Nm]	1,2
Winding resistance Ph-Ph	R_{20} [Ω]	0,28
Winding inductance Ph-Ph	L [mH]	0,86
Rotor moment of inertia	J [kg cm ²]	0,36
Resolution	[steps]	1.8° / 200 full steps
Weight	[kg]	1,00
Cogging torque	M_r [Nm]	0.046 ~ 0.058

Mechanical data	Symbol [Unit]	AS1050
Axial load	[N]	15
Radial load 0 mm from the shaft end	[N]	52
Radial load 5 mm from the shaft end	[N]	65
Radial load 10 mm from the shaft end	[N]	85
Radial load 15 mm from the shaft end	[N]	123
Backlash at standstill	[°]	± 0,054
Axial backlash max. ¹⁾	[mm]	0,075
Radial backlash max. ²⁾	[mm]	0,025
Bearing life	[h]	30.000

1) measured at a load of 9 N

2) measured at a load of 4.4 N



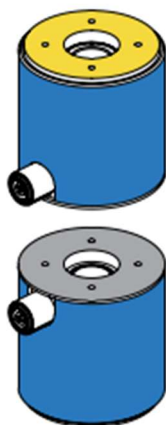
Appendix B - Futek TFF400 datasheet



FEATURES

- Easily integrates into OEM applications
- Designed for Torque auditing
- Aluminum construction
- Built-in overload protection on lower ranges
- Strain gauge based
- Easy cable disconnect

- Active End
- Fixed End
- Non-loading surface, do not contact



SPECIFICATIONS

PERFORMANCE

Nonlinearity	±0.2% of RO
Hysteresis	±0.2% of RO
Nonrepeatability	±0.05% of RO

ELECTRICAL

Rated Output (RO)	1 mV/V nom (5 in-oz) 2 mV/V nom (10 in-oz to 500 in-lb)
Excitation (VDC or VAC)	18 max
Bridge Resistance	350 Ohm nom (5 to 1000 in-oz) 700 Ohm nom (100 to 500 in-lb)
Connection	4 Pin LEMO® Receptacle (EGG.08.304.CLL)
Wiring/Connector Code	CC4

MECHANICAL

Weight (approximate)	9 oz [250 g]
Safe Overload	300% (5 to 1000 in-oz) of RO 150% (100 in-lb to 500 in-lb) of RO
Material	Aluminum
IP Rating	IP40

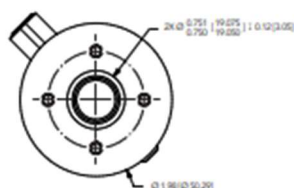
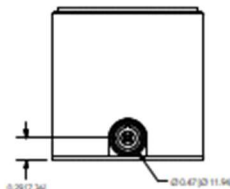
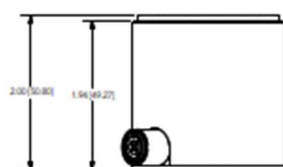
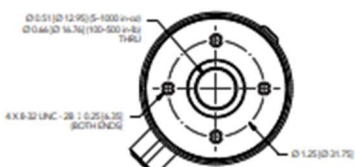
TEMPERATURE

Operating Temperature	-60 to 200°F (-50 to 93°C)
Compensated Temperature	60 to 160°F (15 to 72°C)
Temperature Shift Zero	±0.002% of RO/°F (0.0036% of RO/°C)
Temperature Shift Span	±0.002% of Load/°F (0.0036% of Load/°C)

CALIBRATION

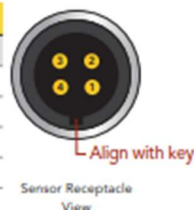
Calibration Test Excitation	10 VDC
Calibration (standard)	5-pt CW
Calibration (available)	5-pt CW & CCW
Shunt Calibration Value	60.4 kOhm (10 to 1000 in-oz) 100 kOhm (5 in-oz, 100 to 500 in-lb)

DIMENSIONS inches [mm]



LEMO 4-PIN

PIN	COLOR	DESCRIPTION
1	Red	+ Excitation
2	Green	+ Signal
3	White	- Signal
4	Black	- Excitation



CAPACITIES

ITEM #	in-oz	Nm	Torsional Stiffness in-oz/rad	Natural Frequency (Hz)
FSH03981	5*	0.04	560	95
FSH03982	10*	0.07	560	95
FSH03984	50*	0.35	4200	260
FSH03985	160*	1.1	21000	504
FSH03986	400*	2.8	62400	860
FSH03987	1000*	7.1	211900	1760
FSH04015	100 in-lb	11	36400 in-lb/rad	2980
FSH04016	200 in-lb	22	61400 in-lb/rad	3850
FSH04017	500 in-lb	56	122000 in-lb/rad	5400

*With overload protection.

For high capacities refer to models TFF600 and TDF600-675.

Appendix C - TwinCAT Main Program

```

1  PROGRAM MAIN
2  VAR
3
4      Vel AT %I* : INT ;
5      Enc AT %I* : UINT ;
6      Torque AT %I* : INT ;
7      ForceX AT %I* : INT ;
8      ForceY AT %I* : INT ;
9      ForceZ AT %I* : INT ;
10
11
12      MotorLowR AT %Q* : BOOL ;
13      MotorLowL AT %Q* : BOOL ;
14      MotorHighR AT %Q* : BOOL ;
15      MotorHighL AT %Q* : BOOL ;
16
17      R_In_CellCalibration : R_TRIG ;
18      R_In_CellCalibrationStatic : R_TRIG ;
19
20      TorqueCalibration : INT := 7843 ;
21      EncoderINIT : UINT := 367 ;
22      EncDif : DINT ;
23
24
25  END_VAR
26

```

```

1  //Torque Sensor Calibration
2  R_In_CellCalibration ( clk := gvl . In_CellCalibration ) ;
3  R_In_CellCalibrationStatic ( clk := gvl . In_CellCalibrationStatic ) ;
4
5  IF R_In_CellCalibration . Q THEN
6      TorqueCalibration := Torque ;
7  END_IF
8
9  IF R_In_CellCalibrationStatic . Q THEN
10     TorqueCalibration := 7843 ;
11  END_IF
12
13  //Visualization Motor Ajustment
14  MotorLowR := gvl . In_MotorLowR ;
15  MotorLowL := gvl . In_MotorLowL ;
16  MotorHighR := gvl . In_MotorHighR ;
17  MotorHighL := gvl . In_MotorHighL ;
18  gvl . Out_LampVisu := ( gvl . In_MotorHighL OR gvl . In_MotorHighR OR gvl .
19  In_MotorLowL OR gvl . In_MotorLowR ) ;
20
21  //Node-RED mapping variables
22  gvl . MotorVEL := Vel ;
23  gvl . MotorENC := Enc ;
24  gvl . Torque := ( Torque - TorqueCalibration ) * 0.0000862006 ;
25  gvl . Grams := ( Torque - TorqueCalibration ) * -0.0081944 ;
26  gvl . ForceX := ForceX * 0.005 ;
27  gvl . ForceY := ForceY * 0.005 ;
28  gvl . ForceZ := ForceZ * 0.005 ;
29
30  EncDif := ( Enc - EncoderINIT ) ;
31  gvl . BeamAngle := EncDif * 0.011414 ;

```

Appendix D - Sunrise Workbench - Motor Control Background Task

BackgroundMotorAdjustment.java

```
package application;

import javax.inject.Inject;
import java.util.concurrent.TimeUnit;

import com.kuka.generated.ioAccess.InputVARIORGroup;
import com.kuka.generated.ioAccess.STEPPERIORGroup;
import com.kuka.roboticsAPI.applicationModel.tasks.CycleBehavior;
import com.kuka.roboticsAPI.applicationModel.tasks.RoboticsAPICyclicBackgroundTask;
import com.kuka.roboticsAPI.controllerModel.Controller;
import com.kuka.roboticsAPI.controllerModel.sunrise.ISafetyState;
import com.kuka.roboticsAPI.controllerModel.sunrise.SunriseSafetyState.SafetyStopType;
import com.kuka.roboticsAPI.deviceModel.LBR;

public class BackgroundMotorAdjustment extends RoboticsAPICyclicBackgroundTask {
    @Inject

    private Controller kuka_Sunrise_Cabinet_1;
    private STEPPERIORGroup iostepper;
    private InputVARIORGroup ioinput;
    private LBR lbr;

    @Override
    public void initialize() {

        kuka_Sunrise_Cabinet_1 = getController("KUKA_Sunrise_Cabinet_1");
        iostepper = new STEPPERIORGroup(kuka_Sunrise_Cabinet_1);
        ioinput = new InputVARIORGroup(kuka_Sunrise_Cabinet_1);
        lbr = (LBR) getDevice(kuka_Sunrise_Cabinet_1, "LBR_iiwa_14_R820_1");

        // initialize your task here
        initializeCyclic(0, 200, TimeUnit.MILLISECONDS,
            CycleBehavior.BestEffort);

    }

    @Override
    public void initialize() {

        kuka_Sunrise_Cabinet_1 = getController("KUKA_Sunrise_Cabinet_1");
        iostepper = new STEPPERIORGroup(kuka_Sunrise_Cabinet_1);
        ioinput = new InputVARIORGroup(kuka_Sunrise_Cabinet_1);
        lbr = (LBR) getDevice(kuka_Sunrise_Cabinet_1, "LBR_iiwa_14_R820_1");

        // initialize your task here
        initializeCyclic(0, 200, TimeUnit.MILLISECONDS,
            CycleBehavior.BestEffort);

    }

    @Override
    public void runCyclic() {

        // Emergency Stop Enabling
        ISafetyState Safety = lbr.getSafetyState();
        SafetyStopType SafetyStop = Safety.getSafetyStopSignal();
    }
}
```

```

// HMI for external control for the Stepper Motor
if (SafetyStop==SafetyStopType.NOSTOP) {

    if (ioinput.getIn_MotorLowL()==true) {
        iostepper.setEnabled(true);
        iostepper.setVelocity(-1);
    }
    if (ioinput.getIn_MotorLowR()==true) {
        iostepper.setEnabled(true);
        iostepper.setVelocity(1);
    }
    if (ioinput.getIn_MotorHighL()==true) {
        iostepper.setEnabled(true);
        iostepper.setVelocity(-9);
    }
    if (ioinput.getIn_MotorHighR()==true) {
        iostepper.setEnabled(true);
        iostepper.setVelocity(9);
    }

    if
    ((ioinput.getIn_MotorHighR()==false)&&(ioinput.getIn_MotorHighL()==false)&&(ioinput.getIn_MotorLowR()==false)&&(ioinput.getIn_MotorLowL()==false)) {
        iostepper.setEnabled(false);
        iostepper.setVelocity(0);
    }
}
else {

    iostepper.setEnabled(false);
    iostepper.setVelocity(0);
}

}
}

```

Appendix E - Sunrise Workbench - Variables Background Task

BackgroundVariables.java

```
package application;

import javax.inject.Inject;
import java.util.concurrent.TimeUnit;

import com.kuka.generated.ioAccess.InputVARIORGroup;
import com.kuka.generated.ioAccess.OutputVARIORGroup;
import com.kuka.generated.ioAccess.STEPPERIORGroup;
import com.kuka.roboticsAPI.applicationModel.tasks.CycleBehavior;
import com.kuka.roboticsAPI.applicationModel.tasks.RoboticsAPICyclicBackgroundTask;
import com.kuka.roboticsAPI.controllerModel.Controller;
import com.kuka.roboticsAPI.deviceModel.LBR;
import com.kuka.roboticsAPI.geometricModel.World;
import com.kuka.roboticsAPI.geometricModel.math.Vector;
import com.kuka.roboticsAPI.sensorModel.ForceSensorData;

public class BackgroundVariables extends RoboticsAPICyclicBackgroundTask {
    @Inject

    private Controller kuka_Sunrise_Cabinet_1;
    private STEPPERIORGroup iostepper;
    private InputVARIORGroup ioinput;
    private OutputVARIORGroup iooutput;
    private LBR lbr;

    @Override
    public void initialize() {

        kuka_Sunrise_Cabinet_1 = getController("KUKA Sunrise Cabinet 1");
        lbr = (LBR) getDevice(kuka_Sunrise_Cabinet_1, "LBR iiwa 14 R820 1");
        iostepper = new STEPPERIORGroup(kuka_Sunrise_Cabinet_1);
        ioinput = new InputVARIORGroup(kuka_Sunrise_Cabinet_1);
        iooutput = new OutputVARIORGroup(kuka_Sunrise_Cabinet_1);

        // initialize your task here
        initializeCyclic(0, 200, TimeUnit.MILLISECONDS,
            CycleBehavior.BestEffort);
    }

    @Override
    public void runCyclic() {

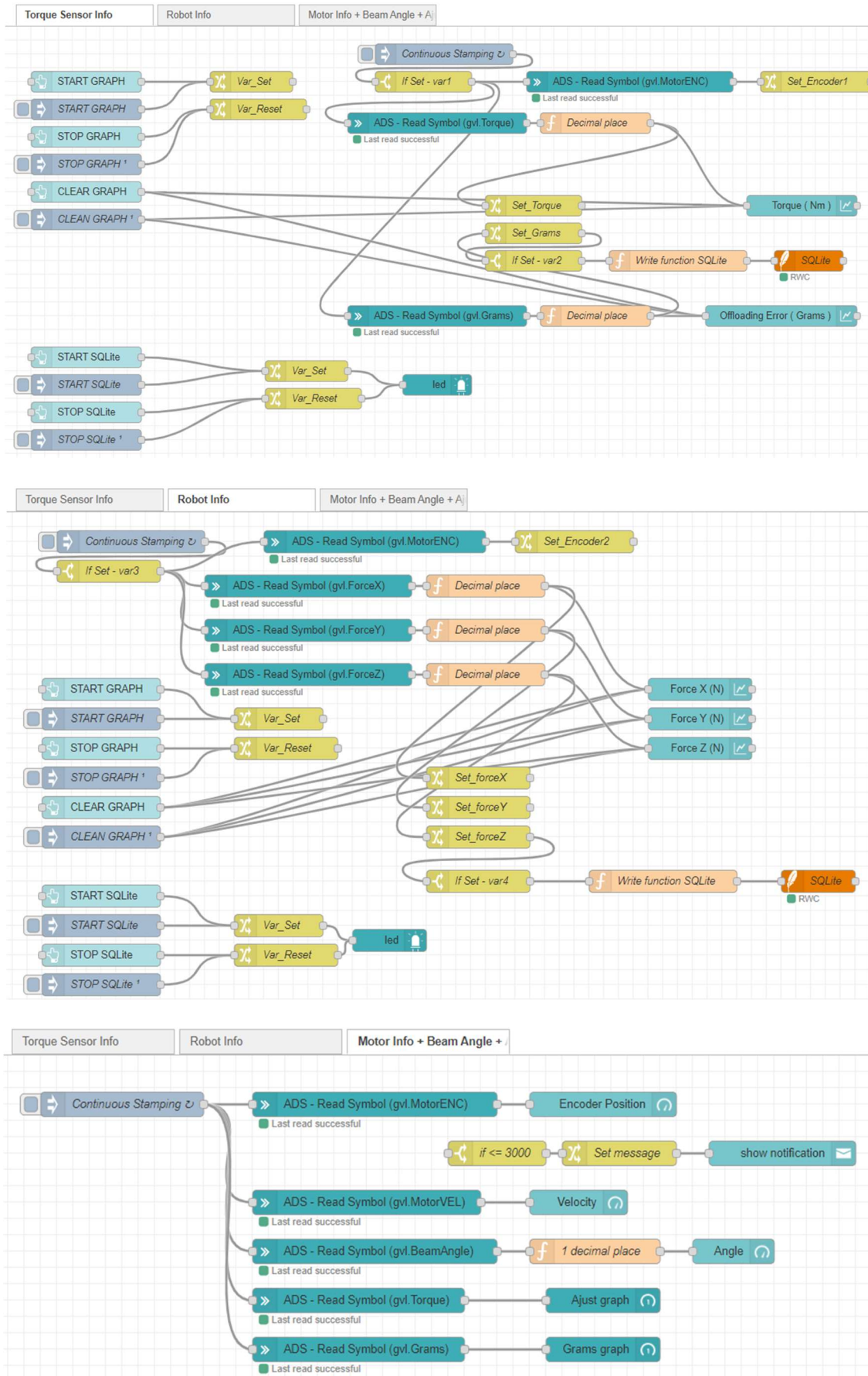
        // Velocity and Encoder to PC
        iooutput.setOut0(iostepper.getVelocity());
        iooutput.setOut4(iostepper.getENC_Counter());

        // Torque sensor to PC
        iooutput.setOut1(ioinput.getIn1());

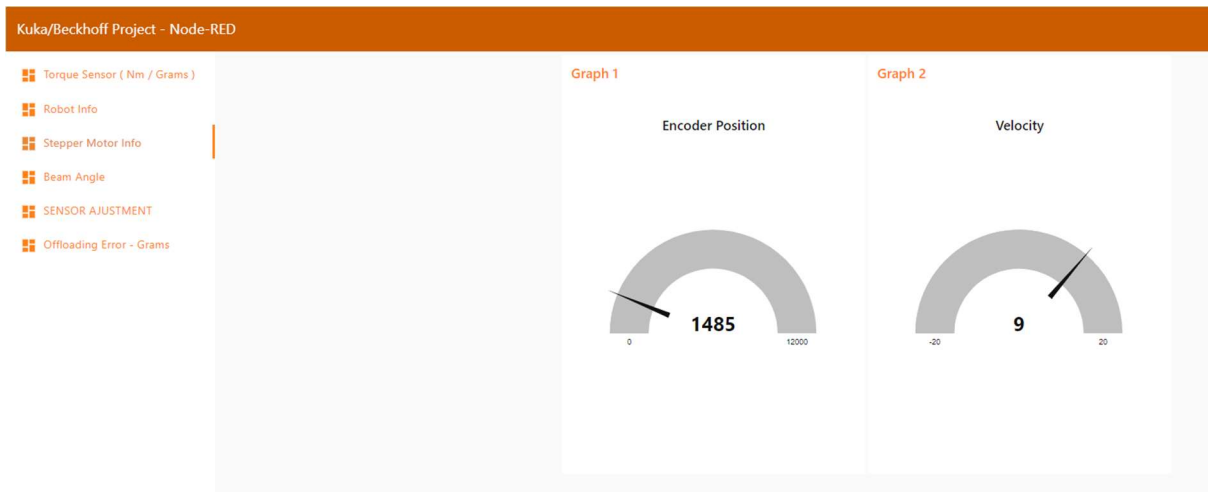
        // XY forces to PC
        ForceSensorData datal =
            lbr.getExternalForceTorque(lbr.getFlange(), World.Current.getRootFrame()
        );

        Vector forceV = datal.getForce();
        double forceInX = forceV.getX()*200;
        double forceInY = forceV.getY()*200;
        double forceInZ = forceV.getZ()*200;
        int forceInX_INT = (int)forceInX;
        int forceInY_INT = (int)forceInY;
        int forceInZ_INT = (int)forceInZ;
        iooutput.setOut2(forceInX_INT);
        iooutput.setOut3(forceInY_INT);
        iooutput.setOut5(forceInZ_INT);
    }
}
```

Appendix F - Node-RED Programming tabs



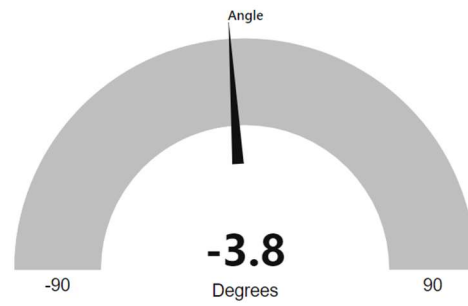
Appendix G - Node-RED Dashboard tabs



Kuka/Beckhoff Project - Node-RED

- Torque Sensor (Nm / Grams)
- Robot Info
- Stepper Motor Info
- Beam Angle
- SENSOR AJUSTMENT
- Offloading Error - Grams

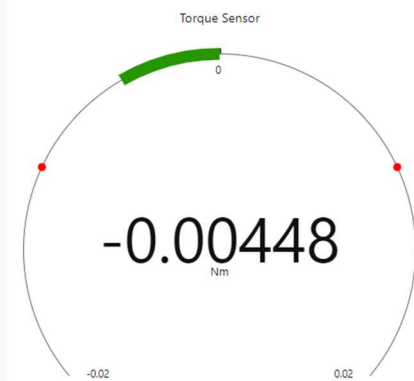
Graph 1



Kuka/Beckhoff Project - Node-RED

- Torque Sensor (Nm / Grams)
- Robot Info
- Stepper Motor Info
- Beam Angle
- SENSOR AJUSTMENT
- Offloading Error - Grams

Graph 1



Kuka/Beckhoff Project - Node-RED

- Torque Sensor (Nm / Grams)
- Robot Info
- Stepper Motor Info
- Beam Angle
- SENSOR AJUSTMENT
- Offloading Error - Grams

Graph 1



Appendix H - Sunrise Workbench - RIC mode

CA_ROBOTIMP.java

```
package application;

import javax.inject.Inject;

import com.kuka.roboticsAPI.applicationModel.RoboticsAPIApplication;
import static com.kuka.roboticsAPI.motionModel.BasicMotions.*;

import com.kuka.generated.ioAccess.MediaFlangeIOGroup;
import com.kuka.roboticsAPI.controllerModel.Controller;
import com.kuka.roboticsAPI.deviceModel.LBR;
import com.kuka.roboticsAPI.geometricModel.LoadData;
import com.kuka.roboticsAPI.geometricModel.Tool;
import com.kuka.roboticsAPI.geometricModel.Workpiece;
import com.kuka.roboticsAPI.motionModel.controlModeModel.JointImpedanceControlMode;
import com.kuka.roboticsAPI.uiModel.ApplicationDialogType;

public class CA_ROBOTIMP extends RoboticsAPIApplication {

    final static double offsetAxis2And4=Math.toRadians(10);

    @Inject
    private LBR lbr;
    private Controller kuka_Sunrise_Cabinet_1;
    private MediaFlangeIOGroup io;

    //Variables declaration Tool and Workpiece;
    private Tool Tool;
    private Workpiece MassTEST;

    public void initialize() {
        kuka_Sunrise_Cabinet_1 = getController("KUKA Sunrise Cabinet 1");
        lbr = (LBR) getDevice(kuka_Sunrise_Cabinet_1, "LBR iiwa 14 R820 1");
        io = new MediaFlangeIOGroup(kuka_Sunrise_Cabinet_1);
    }

    public void run() {

        MassTEST = getApplicationData().createFromTemplate("MassTEST");

        LoadData MassTEST_DATA = MassTEST.getLoadData();
        double Mass = MassTEST_DATA.getMass();

        String str = "Confirm the Mass of the Workpiece. If the value is not OK please  
Cancel the program and change it OFFLINE. MassTEST = "+Mass+" kg?";

        getLogger().info("Show modal dialog and wait for user to confirm");
        int isCancel =
        getApplicationUI().displayModalDialog(ApplicationDialogType.QUESTION,str, "OK",
        "Cancel");
        if (isCancel == 1)
        {
            return;
        }
        ;
        getLogger().info("Robot is now free to move.");

        JointImpedanceControlMode jointImp =
            new JointImpedanceControlMode(0, 0, 0, 0, 0, 0, 2000);

        //Position mode active;
        Tool = getApplicationData().createFromTemplate("Tool");
        Tool.attachTo(lbr.getFlange());
    }
}
```

```

io.setLEDBLue(true);
boolean position = true;
while (position==true) {
    lbr.moveAsync(linRel(0, 0, 0).setCartVelocity(2000).setMode(jointImp));

    if (io.getUserButton()==true) {
        position= false;
        io.setLEDBLue(false);
    }
}

//Load attach and axes parameterization;

MassTEST.attachTo(lbr.getFlange());
jointImp.setStiffness(0,0,2000,0,2000,0,2000);
jointImp.setDampingForAllJoints(0.0);

try {
    Thread.sleep(2000);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

//Start robot load mode

boolean load = true;
while (load==true) {
    lbr.moveAsync(linRel(0, 0, 0).setCartVelocity(1000).setMode(jointImp));

    if (io.getUserButton()==true) {
        load = false;
    }
}

getLogger().info("Show modal dialog while executing gravity compensation.");

//Robot stop application
Tool.detach();
MassTEST.detach();
}
}

```

Appendix I - Sunrise Workbench - RIC_TS - XZ plane

CB_xz_SENSORplusROBOTIMP.java

```
package application;

import javax.inject.Inject;

import com.kuka.generated.ioAccess.InputVARIORGroup;
import com.kuka.roboticsAPI.applicationModel.RoboticsAPIApplication;
import static com.kuka.roboticsAPI.motionModel.BasicMotions.*;

import com.kuka.roboticsAPI.controllerModel.Controller;
import com.kuka.roboticsAPI.deviceModel.LBR;
import com.kuka.roboticsAPI.geometricModel.CartDOF;
import com.kuka.roboticsAPI.geometricModel.LoadData;
import com.kuka.roboticsAPI.geometricModel.Tool;
import com.kuka.roboticsAPI.geometricModel.Workpiece;
import com.kuka.roboticsAPI.motionModel.controlModeModel.CartesianImpedanceControlMode;
import com.kuka.roboticsAPI.uiModel.ApplicationDialogType;

public class CB_xz_SENSORplusROBOTIMP extends RoboticsAPIApplication {
    @Inject

    final static double offsetAxis2And4=Math.toRadians(10);

    private static double[] startPosition=new
double[]{0,offsetAxis2And4,0,offsetAxis2And4-Math.toRadians(90),0,Math.toRadians(90),0}
;

    private LBR lbr;
    private Controller kuka_Sunrise_Cabinet_1;
    private InputVARIORGroup ioinput;
    private Tool Tool;
    private Workpiece MassTEST;

    @Override
    public void initialize() {
        // initialize your application here
        kuka Sunrise Cabinet 1 = getController("KUKA Sunrise Cabinet 1");
        lbr = (LBR) getDevice(kuka Sunrise Cabinet 1, "LBR iiwa 14 R820 1");
        ioinput = new InputVARIORGroup(kuka Sunrise Cabinet 1);
    }

    @Override
    public void run() {
        // your application execution starts here

        MassTEST = getApplicationData().createFromTemplate("MassTEST");

        LoadData MassTEST_DATA = MassTEST.getLoadData();
        double Mass = MassTEST_DATA.getMass();

        String str = "Confirm the Mass of the Workpiece. If the value is not OK please
Cancel the program and change it OFFLINE. MassTEST = "+Mass+" kg?";

        getLogger().info("Show modal dialog and wait for user to confirm");
        int isCancel =
getApplicationUI().displayModalDialog(ApplicationDialogType.QUESTION,str, "OK",
"Cancel");
        if (isCancel == 1)
        {
            return;
        }
    }
}
```

```

Tool = getApplicationData().createFromTemplate("Tool");
Tool.attachTo(lbr.getFlange());

CartesianImpedanceControlMode impedanceControlMode = new
CartesianImpedanceControlMode();
impedanceControlMode.parametrize(CartDOF.X).setStiffness(0);
impedanceControlMode.parametrize(CartDOF.Y).setStiffness(0);
impedanceControlMode.parametrize(CartDOF.Z).setStiffness(0);

MassTEST = getApplicationData().createFromTemplate("MassTEST");
MassTEST.attachTo(lbr.getFlange());

getApplicationUI().displayModalDialog(ApplicationDialogType.INFORMATION,
"ATTENTION!! Load attached to the robot. It will reposition the flange.", "OK");

// Robot flange movement to perpendicular position
lbr.move(ptp((startPosition)).setMode(impedanceControlMode).setJointVelocityRel
0.3));

impedanceControlMode.parametrize(CartDOF.Z).setStiffness(200);
impedanceControlMode.parametrize(CartDOF.Y).setStiffness(200);

getApplicationUI().displayModalDialog(ApplicationDialogType.INFORMATION, "Make
sure that torque sensor is calibrated. Once it's done, press OK to start de
application.", "OK");

// Application logic
boolean enable = true;
while (enable==true) {
    int difference = ioinput.getIn1()-7843;
    if (difference<=-150) {
        lbr.move(linRel(0,0,(difference*0.002)).setCartVelocity(500).setMod
e(impedanceControlMode));
    }
    if (difference>=150) {
        lbr.move(linRel(0,0,(difference*0.002)).setCartVelocity(500).setMod
e(impedanceControlMode));
    }
}
}

```

Appendix J - Sunrise Workbench - RIC_TS - XYZ plane

CB_xyz_SENSORplusROBOTIMP.java

```
package application;

import javax.inject.Inject;

import com.kuka.generated.ioAccess.InputVARIORGroup;
import com.kuka.roboticsAPI.applicationModel.RoboticsAPIApplication;
import static com.kuka.roboticsAPI.motionModel.BasicMotions.*;

import com.kuka.roboticsAPI.controllerModel.Controller;
import com.kuka.roboticsAPI.deviceModel.LBR;
import com.kuka.roboticsAPI.geometricModel.CartDOF;
import com.kuka.roboticsAPI.geometricModel.LoadData;
import com.kuka.roboticsAPI.geometricModel.Tool;
import com.kuka.roboticsAPI.geometricModel.Workpiece;
import com.kuka.roboticsAPI.motionModel.controlModeModel.CartesianImpedanceControlMode;
import com.kuka.roboticsAPI.uiModel.ApplicationDialogType;

public class CB_xyz_SENSORplusROBOTIMP extends RoboticsAPIApplication {
    @Inject

    final static double offsetAxis2And4=Math.toRadians(10);

    private static double[] startPosition=new
double[] {0,offsetAxis2And4,0,offsetAxis2And4-Math.toRadians(90),0,Math.toRadians(90),0}
;

    private LBR lbr;
    private Controller kuka_Sunrise_Cabinet_1;
    private InputVARIORGroup ioinput;
    private Tool Tool;
    private Workpiece MassTEST;

    @Override
    public void initialize() {
        // initialize your application here
        kuka_Sunrise_Cabinet_1 = getController("KUKA Sunrise Cabinet 1");
        lbr = (LBR) getDevice(kuka_Sunrise_Cabinet_1, "LBR iiwa 14 R820 1");
        ioinput = new InputVARIORGroup(kuka_Sunrise_Cabinet_1);
    }
    @Override
    public void run() {
        // your application execution starts here

        MassTEST = getApplicationData().createFromTemplate("MassTEST");

        LoadData MassTEST_DATA = MassTEST.getLoadData();
        double Mass = MassTEST_DATA.getMass();

        String str = "Confirm the Mass of the Workpiece. If the value is not OK please
Cancel the program and change it OFFLINE. MassTEST = "+Mass+" kg?";

        getLogger().info("Show modal dialog and wait for user to confirm");
        int isCancel =
getApplicationUI().displayModalDialog(ApplicationDialogType.QUESTION,str, "OK",
"Cancel");
        if (isCancel == 1)
        {
            return;
        }
        ;

        Tool = getApplicationData().createFromTemplate("Tool");
    }
}
```

```

Tool.attachTo(lbr.getFlange());

CartesianImpedanceControlMode impedanceControlMode = new
CartesianImpedanceControlMode();
impedanceControlMode.parametrize(CartDOF.X).setStiffness(0);
impedanceControlMode.parametrize(CartDOF.Y).setStiffness(0);
impedanceControlMode.parametrize(CartDOF.Z).setStiffness(0);

MassTEST = getApplicationData().createFromTemplate("MassTEST");
MassTEST.attachTo(lbr.getFlange());

getApplicationUI().displayModalDialog(ApplicationDialogType.INFORMATION,
"ATTENTION!! Load attached to the robot. It will reposition the flange.", "OK");

// Robot flange movement to perpendicular position
lbr.move(ptp((startPosition)).setMode(impedanceControlMode).setJointVelocityRel
0.3));

impedanceControlMode.parametrize(CartDOF.Z).setStiffness(200);

getApplicationUI().displayModalDialog(ApplicationDialogType.INFORMATION, "Make
sure that torque sensor is calibrated. Once it's done, press OK to start de
application.", "OK");

// Application logic
boolean enable = true;
while (enable==true) {
    int difference = ioinput.getIn1()-7843;
    if (difference<=-150) {
        lbr.move(linRel(0,0,(difference*0.002)).setCartVelocity(500).setMod
e(impedanceControlMode));
    }
    if (difference>=150) {
        lbr.move(linRel(0,0,(difference*0.002)).setCartVelocity(500).setMod
e(impedanceControlMode));
    }
}
}

```

Appendix K - Sunrise Workbench - RIC_TS_E

CC_xz_SENSORplusENCODER.java

```

package application;

import javax.inject.Inject;

import com.kuka.generated.ioAccess.InputVARIORGroup;
import com.kuka.generated.ioAccess.STEPPERIORGroup;
import com.kuka.roboticsAPI.applicationModel.RoboticsAPIApplication;
import static com.kuka.roboticsAPI.motionModel.BasicMotions.*;

import com.kuka.roboticsAPI.controllerModel.Controller;
import com.kuka.roboticsAPI.deviceModel.LBR;
import com.kuka.roboticsAPI.geometricModel.CartDOF;
import com.kuka.roboticsAPI.geometricModel.LoadData;
import com.kuka.roboticsAPI.geometricModel.Tool;
import com.kuka.roboticsAPI.geometricModel.Workpiece;
import com.kuka.roboticsAPI.motionModel.controlModeModel.CartesianImpedanceControlMode;
import com.kuka.roboticsAPI.uiModel.ApplicationDialogType;

public class CC_xz_SENSORplusENCODER extends RoboticsAPIApplication {
    @Inject

    final static double offsetAxis2And4=Math.toRadians(10);

    private static double[] startPosition=new
double[] {0,offsetAxis2And4,0,offsetAxis2And4-Math.toRadians(90),0,Math.toRadians(90),0}
;

    private LBR lbr;
    private Controller kuka Sunrise Cabinet 1;
    private InputVARIORGroup ioinput;
    private STEPPERIORGroup iostepper;
    private Tool Tool;
    private Workpiece MassTEST;

    @Override
    public void initialize() {
        // initialize your application here
        kuka Sunrise Cabinet 1 = getController("KUKA Sunrise Cabinet 1");
        lbr = (LBR) getDevice(kuka Sunrise Cabinet 1, "LBR iiwa 14 R820 1");
        ioinput = new InputVARIORGroup(kuka_Sunrise_Cabinet_1);
        iostepper = new STEPPERIORGroup(kuka_Sunrise_Cabinet_1);
    }

    @Override
    public void run() {
        // your application execution starts here

        MassTEST = getApplicationData().createFromTemplate("MassTEST");

        LoadData MassTEST DATA = MassTEST.getLoadData();
        double Mass = MassTEST DATA.getMass();

        String str = "Confirm the Mass of the Workpiece. If the value is not OK please
Cancel the program and change it OFFLINE. MassTEST = "+Mass+" kg?";

        getLogger().info("Show modal dialog and wait for user to confirm");
        int isCancel =
getApplicationUI().displayModalDialog(ApplicationDialogType.QUESTION,str, "OK",
"Cancel");
        if (isCancel == 1)
        {
            return;
        }
    }
}

```

```

    }
    ;

    Tool = getApplicationData().createFromTemplate("Tool");
    Tool.attachTo(lbr.getFlange());

    CartesianImpedanceControlMode impedanceControlMode = new
    CartesianImpedanceControlMode();
    impedanceControlMode.parametrize(CartDOF.X).setStiffness(0);
    impedanceControlMode.parametrize(CartDOF.Y).setStiffness(0);
    impedanceControlMode.parametrize(CartDOF.Z).setStiffness(0);

    MassTEST = getApplicationData().createFromTemplate("MassTEST");
    MassTEST.attachTo(lbr.getFlange());

    getApplicationUI().displayModalDialog(ApplicationDialogType.INFORMATION,
    "ATTENTION!! Load attached to the robot. It will reposition the flange.", "OK");

    // Robot flange movement to perpendicular position
    lbr.move(ptp((startPosition)).setMode(impedanceControlMode).setJointVelocityRel
    0.3));

    impedanceControlMode.parametrize(CartDOF.Z).setStiffness(200);
    impedanceControlMode.parametrize(CartDOF.X).setStiffness(200);
    impedanceControlMode.parametrize(CartDOF.Y).setStiffness(200);

    getApplicationUI().displayModalDialog(ApplicationDialogType.INFORMATION, "Make
    sure that torque sensor is calibrated. Once it's done, press OK to start de
    application.", "OK");

    int ENCIInit = 367;

    // Application logic
    boolean enable = true;
    while (enable==true) {

        int difference = ioinput.getIn1()-7843;

        if ((difference<=-150)|| (difference>=150)) {

            Integer ENCCurrent = iostepper.getENC_Counter();
            int ENCDif = ENCCurrent-ENCIInit;
            double Angle = ENCDif*0.000199;

            double DispX = difference*0.004*Math.sin(Angle);
            double DispZ = difference*0.002*Math.cos(Angle);

            lbr.move(linRel(DispX,0,DispZ).setCartVelocity(500).setMode(impedan
            ceControlMode));

        }
    }
}

```

Appendix L - Sunrise Workbench - RIC_TS_RF - XZ plane

CD_xz_SENSORplusROBOTIMPplusROBOTFORCE.java

```

package application;

import javax.inject.Inject;

public class CD_xz_SENSORplusROBOTIMPplusROBOTFORCE extends RoboticsAPIApplication {
    @Inject

    final static double offsetAxis2And4=Math.toRadians(10);

    private static double[] startPosition=new
double[] {0,offsetAxis2And4,0,offsetAxis2And4-Math.toRadians(90),0,Math.toRadians(90),0}
;

    private LBR lbr;
    private Controller kuka Sunrise Cabinet 1;
    private InputVARIOWGroup ioinput;
    private Tool Tool;
    private Workpiece MassTEST;

    @Override
    public void initialize() {
        // initialize your application here
        kuka_Sunrise_Cabinet_1 = getController("KUKA_Sunrise_Cabinet_1");
        lbr = (LBR) getDevice(kuka_Sunrise_Cabinet_1, "LBR_iiwa_14_R820_1");
        ioinput = new InputVARIOWGroup(kuka_Sunrise_Cabinet_1);
    }

    @Override
    public void run() {
        // your application execution starts here

        MassTEST = getApplicationData().createFromTemplate("MassTEST");

        LoadData MassTEST DATA = MassTEST.getLoadData();
        double Mass = MassTEST DATA.getMass();

        String str = "Confirm the Mass of the Workpiece. If the value is not OK please
Cancel the program and change it OFFLINE. MassTEST = "+Mass+" kg?";

        getLogger().info("Show modal dialog and wait for user to confirm");
        int isCancel =
getApplicationUI().displayModalDialog(ApplicationDialogType.QUESTION,str, "OK",
"Cancel");
        if (isCancel == 1)
        {
            return;
        }
        ;

        Tool = getApplicationData().createFromTemplate("Tool");
        Tool.attachTo(lbr.getFlange());

        CartesianImpedanceControlMode impedanceControlMode = new
CartesianImpedanceControlMode();
        impedanceControlMode.parametrize(CartDOF.X).setStiffness(0);
        impedanceControlMode.parametrize(CartDOF.Y).setStiffness(0);
        impedanceControlMode.parametrize(CartDOF.Z).setStiffness(0);

        MassTEST = getApplicationData().createFromTemplate("MassTEST");
        MassTEST.attachTo(lbr.getFlange());
    }

```

```

    getApplicationUI().displayModalDialog(ApplicationDialogType.INFORMATION,
    "ATTENTION!! Load attached to the robot. It will reposition the flange.", "OK");

    // Robot flange movement to perpendicular position
    lbr.move(ptp((startPosition)).setMode(impedanceControlMode).setJointVelocityRel
0.3));

    impedanceControlMode.parametrize(CartDOF.Z).setStiffness(200);
    impedanceControlMode.parametrize(CartDOF.Y).setStiffness(200);
    impedanceControlMode.parametrize(CartDOF.X).setStiffness(200);

    getApplicationUI().displayModalDialog(ApplicationDialogType.INFORMATION, "Make
sure that torque sensor is calibrated and the robot is stable. Once it's done, press OK
to start de application.", "OK");

    //Initialized Steep Motor Encoder;
    double DispX=0;
    int CountXNeg=0;
    int CountXPos=0;

    // Application logic
    boolean enable = true;
    while (enable==true) {

        int difference = ioinput.getIn1()-7843;

        if ((difference<=-150)|| (difference>=150)){

            double DispZ = difference*0.002;

            ForceSensorData data2 =
lbr.getExternalForceTorque(lbr.getFlange());
            Vector forceV = data2.getForce();
            double forceInX = forceV.getX();
            if (forceInX <= -0.20) {
                CountXNeg = CountXNeg+1 ;
            } else {
                CountXNeg = 0;
            }
            if (forceInX >= 0.20) {
                CountXPos = CountXPos+1 ;
            } else {
                CountXPos = 0;
            }
            if (CountXNeg>=4) {
                DispX=-2;
            }
            if (CountXPos>=4) {
                DispX=2;
            }
            if ((CountXPos<4)&&(CountXNeg<4)) {
                DispX=0;
            }

            lbr.move(linRel(DispX,0,DispZ).setCartVelocity(500).setMode(impedan
ceControlMode));

        }

    }

}

```

Appendix M - Sunrise Workbench - RIC_TS_RF - XYZ plane

```

CD_xyz_SENSORplusROBOTIMPplusROBOTFORCE.java

package application;

import javax.inject.Inject;

public class CD_xyz_SENSORplusROBOTIMPplusROBOTFORCE extends RoboticsAPIApplication {
    @Inject

    final static double offsetAxis2And4=Math.toRadians(10);

    private static double[] startPosition=new
double[]{0,offsetAxis2And4,0,offsetAxis2And4-Math.toRadians(90),0,Math.toRadians(90),0}
;

    private LBR lbr;
    private Controller kuka Sunrise Cabinet 1;
    private InputVARIORGroup ioinput;
    private Tool Tool;
    private Workpiece MassTEST;

    @Override
    public void initialize() {
        // initialize your application here
        kuka_Sunrise_Cabinet_1 = getController("KUKA_Sunrise_Cabinet_1");
        lbr = (LBR) getDevice(kuka_Sunrise_Cabinet_1, "LBR_iiwa_14_R820_1");
        ioinput = new InputVARIORGroup(kuka_Sunrise_Cabinet_1);
    }

    @Override
    public void run() {
        // your application execution starts here

        MassTEST = getApplicationData().createFromTemplate("MassTEST");

        LoadData MassTEST DATA = MassTEST.getLoadData();
        double Mass = MassTEST DATA.getMass();

        String str = "Confirm the Mass of the Workpiece. If the value is not OK please
Cancel the program and change it OFFLINE. MassTEST = "+Mass+" kg?";

        getLogger().info("Show modal dialog and wait for user to confirm");
        int isCancel =
getApplicationUI().displayModalDialog(ApplicationDialogType.QUESTION,str, "OK",
"Cancel");
        if (isCancel == 1)
        {
            return;
        }
        ;

        Tool = getApplicationData().createFromTemplate("Tool");
        Tool.attachTo(lbr.getFlange());

        CartesianImpedanceControlMode impedanceControlMode = new
CartesianImpedanceControlMode();
        impedanceControlMode.parametrize(CartDOF.X).setStiffness(0);
        impedanceControlMode.parametrize(CartDOF.Y).setStiffness(0);
        impedanceControlMode.parametrize(CartDOF.Z).setStiffness(0);

        MassTEST = getApplicationData().createFromTemplate("MassTEST");
        MassTEST.attachTo(lbr.getFlange());
    }
}

```

```

getApplicationUI().displayModalDialog(ApplicationDialogType.INFORMATION,
"ATTENTION!! Load attached to the robot. It will reposition the flange.", "OK");

// Robot flange movement to perpendicular position
lbr.move(ptp((startPosition)).setMode(impedanceControlMode).setJointVelocityRel
0.3));

impedanceControlMode.parametrize(CartDOF.Z).setStiffness(200);
impedanceControlMode.parametrize(CartDOF.Y).setStiffness(200);
impedanceControlMode.parametrize(CartDOF.X).setStiffness(200);

getApplicationUI().displayModalDialog(ApplicationDialogType.INFORMATION, "Make
sure that torque sensor is calibrated and the robot is stable. Once it's done, press OK
to start de application.", "OK");

//Initialized Stepper Motor Encoder;
double DispY=0;
double DispX=0;
int CountYNeg=0;
int CountYPos=0;
int CountXNeg=0;
int CountXPos=0;

// Application logic
boolean enable = true;
while (enable==true) {

    int difference = ioinput.getIn1()-7843;

    if ((difference<=-150)|| (difference>=150)){

        double DispZ = difference*0.002;

        ForceSensorData data2 =
lbr.getExternalForceTorque(lbr.getFlange());
        Vector forceV = data2.getForce();
        double forceInY = forceV.getY();
        double forceInX = forceV.getX();

        if (forceInY <= -0.2) {
            CountYNeg = CountYNeg+1 ;
        } else {
            CountYNeg = 0;
        }

        if (forceInY >= 0.2) {
            CountYPos = CountYPos+1 ;
        } else {
            CountYPos = 0;
        }

        if (CountYNeg>=4) {
            DispY=-1;
        }
        if (CountYPos>=4) {
            DispY=1;
        }
        if ((CountYPos<4) && (CountYNeg<4)) {
            DispY=0;
        }

        if (forceInX <= -0.2) {
            CountXNeg = CountXNeg+1 ;
        } else {
            CountXNeg = 0;
        }
    }
}

```

```
    }  
    if (forceInX >= 0.2) {  
        CountXPos = CountXPos+1 ;  
    } else {  
        CountXPos = 0;  
    }  
    if (CountXNeg>=4) {  
        DispX=-2;  
    }  
    if (CountXPos>=4) {  
        DispX=2;  
    }  
    if ((CountXPos<4)&&(CountXNeg<4)) {  
        DispX=0;  
    }  
  
    lbr.move(linRel(DispX,DispY,DispZ).setCartVelocity(500).setMode(imp  
edanceControlMode));  
  
    }  
  
    }  
  
}
```

Appendix N - Offloading results - No offloading 2.7N, 3.2N

