

Novia University of Applied Science
Date: 10.05.2019



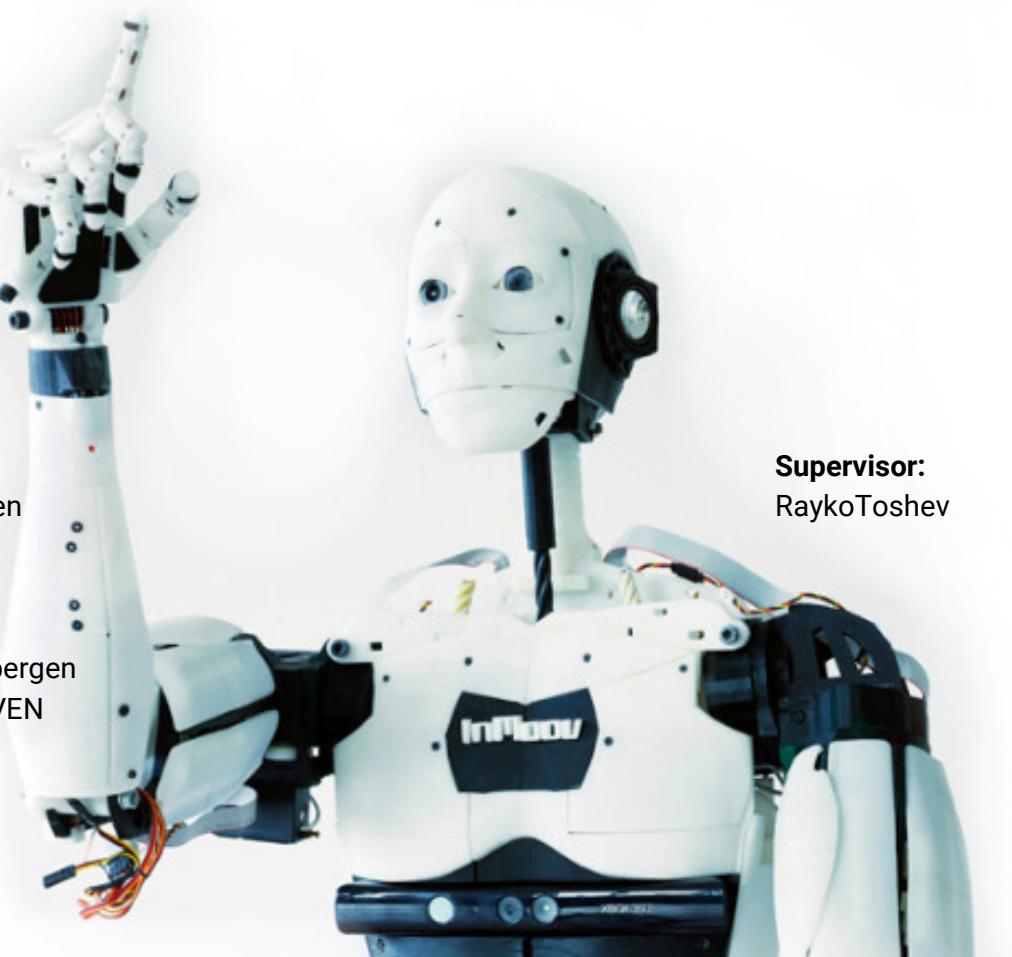
FINAL REPORT

InMoov Humanoid Robot

OPEN-SOURCE PROJECT 3D PRINTED LIFE SIZE ROBOT

Authors:
Britt van Bergen
Oliver Heijster
Ana Sánchez
Niki Kuhar
Brent Van Obbergen
JAN VAN DE VEN

Supervisor:
Rayko Toshev



Acknowledgements

Acknowledgement to Novia University of Applied Sciences for the opportunity to allow us to work on this interesting project, and Technobothnia for supplying all the resources that were required.

We wish to thank various people for their contribution to this project, first Roger Nylund for guiding us this semester through the European Project Semester (EPS) and providing us knowledge of project management. Special thanks should be given to RaykoToshev for supervising our project and giving support during the project.

In addition to all these people we would like to thank the EPS group of this spring semester for the amazing time we have together in Vaasa.

Abstract

During this project 3D printing was combined with robotics. The project mainly includes optimization of the OpenSource Humanoid Robot. The robot was already built by a previous EPS student group at Technobothnia in Vaasa.ⁱ To optimize the humanoid robot, many parts required repairing or replacement, and the software code needed rewriting.

The robot was downloaded from an OpenSource project called InMoov.ⁱⁱ The robot's structure was printed with a 3D printer and other parts to make it function were added; rotary actuator(servos), Microsoft Kinect motion sensing input device, Virtual Reality headset(HtcVive), tablet to control the Arduinos(NuVision tablet), Leap Motion, eye tracking cameras and many others. The main goal of this project is to make the humanoid robot function with all these hardware and software parts and give the robot a purpose. In this report it is described which work has been done to achieve this goal.

Most of the hardware for the humanoid robot had been repaired and replaced. To find a purpose and utility for the humanoid robot the team did a brainstorming session. The outcome of which was that best fields to use this robot are education and information. These fields were further explored by creating multiple scenario's, the librarian was chosen as the final purpose. To implement this librarian scenario to the humanoid an introduction code with gestures was written. To make the robot able to pick up stuff with the hands the finger pressure sensor was optimized. The PIR was tested to this could be of use for implementing gestures. Also, the robot was made fully independent by attaching it to the battery. A switch was added to the battery to turn the robot on and off easily.

Finally, VR was implemented to control the robot with the Virtual Reality controllers. This was achieved by setting up a communication between Unreal Engine and Arduino. The robot was modeled in a VR environment, so the movements made with the controllers would show visually VR.

Abbreviations

3D	Three Dimensional
ABS	Acrylonitrile Butadiene Styrene
PLA	Polylactic acid
Config files	Configuration files
Com ports	Communication ports
EPS	European Project Semester
HR	Human Resources
IT tools	Information Technology tools
PIR	Presence infrared sensor
Mrl	MyRobotLab
Servo	Rotary actuator
VR	Virtual Reality
WBS	Work Breakdown Structure
GPU	Graphic processing unit

Table of contents

1. Preface	10
2. Introduction	12
3. Project Management	13
3.1 Mission and Vision.....	13
3.2 Input and Output	13
3.3 Work Breakdown Structure	14
3.4 Schedule	15
3.5 Human Resource plan.....	17
3.6 Project budget.....	19
3.7 Risks analysis	20
3.8 Quality management.....	22
3.9 Change control.....	24
4. Philosophy presentation.....	27
4.1 Competitor research	27
4.2 Brainstorm session	29
4.3 Humanoid scenarios	31
4.4 Conclusion philosophy presentation	35
5. Improving hardware.....	36
5.1 Repairing 3D printed covers	36
5.2 Hardware revisions	37
Reparation of the neck	37
Reparation of the fingers.....	37
Reparation of the index finger.....	38
Reparation of the bicep	38
Reparation of the zero point in the shoulders	39
Reparation of the left arm	39
Covering of the cables.....	39
Attaching battery	40
6. Improving software.....	42
6.1 Exploring the software	42
6.2 Configuration files	45
6.3 Gestures	46
6.4 Tablet	47

6.4 Testing the Kinect	48
6.5 Presence Infrared sensor – PIR	49
6.5 Fingers' sensor.....	51
6.5 Leap Motion	52
6.6 Testing the eye tracking	53
7. Virtual Reality	56
7.1 Digital twin integration.....	56
7.2 Sending data from Arduino to Unreal Engine	57
7.3 Controlling a LED with Unreal Engine	60
7.4 Controlling multiple LEDs trough Unreal Engine	62
7.5 Using a constantly changing variable to control a LED	64
7.6 Working with servos instead of LEDs	65
7.7 Using VR controllers	66
7.8 Inverse kinematics.....	66
7.9 Inverse kinematics with Unreal Engine	68
8. Humanoid robot model VR integration and level design.....	71
8.1 Level design	71
8.2 Model integration.....	72
8.3 SketchUp	72
8.4 Blender	74
8.5 Unreal Engine 4	78
9. Conclusion	80
10. Recommendations.....	81
References.....	82

List of Figures

Figure 1. InMoov humanoid Robot	10
Figure 2. Belbin roles	11
Figure 3. Input, tools and techniques, output	13
Figure 4. Work Breakdown Structure.....	14
Figure 5. Tasks schedule	15
Figure 6. Gantt Chart Schedule	16
Figure 7. RACI Chart	18
Figure 8. Project Costs	19
Figure 9 Priority list	20
Figure 10. Assessment analysis	21
Figure 11. Quality assessment fish diagram.....	22
Figure 12. 3D print quality.....	22
Figure 13. Project change control log	24
Figure 14. Project change control form 1	25
Figure 15. Project change control form 2.....	25
Figure 16. Project change control form 3	25
Figure 17. Project change control form 4	26
Figure 18. Project change control form 5.....	26
Figure 19. Project change control form 6.....	26
Figure 20. Nao humanoid robot	27
Figure 21. Romeo humanoid robot.....	27
Figure 22. Pepper humanoid robot.....	27
Figure 23. DCR-HUBO	27
Figure 24. Robothespian	27
Figure 25. Brainstorm session	29
Figure 26. Brainstorm Mindmap.....	30
Figure 27. Librarian scenario	35
Figure 28. Blob 3D printer.....	36
Figure 29. Bicep before repairing.....	36
Figure 30. Bicep after repairing.....	36
Figure 31. Neck humanoid robot.....	37
Figure 32. Finger servos	37
Figure 33. Replaced index finger	38
Figure 34. Bicep servo	38
Figure 35. Arm disassembled	39
Figure 36. Cable cover one.....	40
Figure 37. Cable cover two	40
Figure 38. Final cable cover.....	40
Figure 39. Electric scheme battery	41
Figure 40. Battery	41
Figure 41. MyRobotLab interface	42
Figure 42. Tabs interface	43
Figure 43. Monkey engine interface	44
Figure 44. Gesture list	46
Figure 45. Gestures code	47

Figure 46. MRL error.....	48
Figure 47. PIR test codein Arduino.....	49
Figure 48. PIR with LED test code in Arduino	50
Figure 49. PIR with LED test assembly.....	50
Figure 50. Fingers' sensor test code in Arduino	51
Figure 51. Fingers' sensor assembly.....	52
Figure 52. Leap motion	53
Figure 53. FaceDetect	54
Figure 54. LKOpticalTrack.....	55
Figure 55. Unreal engine	56
Figure 56. Arduino sending code	57
Figure 57. Connection to Arduino	58
Figure 58. Changing the light intensity.....	59
Figure 59. Potentiometer testing	59
Figure 60. Arduino LED controlling	60
Figure 61. Unreal code using spacebar to light a LED	60
Figure 62. Controlling a LED with the spacebar	61
Figure 63. Arduino controlling two LEDs.....	62
Figure 64. Controlling multiple LEDs with Unreal Engine	63
Figure 65. Arduino connected to multiple LEDs.....	63
Figure 66. Arduino variable changing	64
Figure 67. Mouse controlling two LEDs	65
Figure 68. Arduino controlling servos	65
Figure 69. Controller setup	66
Figure 70. Connecting the arms to the controllers	66
Figure 71. Arduino inverse kinematics	67
Figure 72. Inverse kinematics	68
Figure 73. Arduino controlling robot arm.....	70
Figure 74. Dell Mixed Reality headset	71
Figure 75. HTC VIVE Headset.....	71
Figure 76. Unreal Engine 4 Enviroment.....	71
Figure 77. SketchUp Enviroment.....	73
Figure 78. Original model.....	73
Figure 79. Model with removed internal parts.....	73
Figure 80. Model with removed limbs	73
Figure 81. Blender Enviroment	75
Figure 82. Detailed model display.....	75
Figure 83. Normal model display	75
Figure 84. Final Blender model.....	76
Figure 85. Blender rig	77
Figure 86. Humanoid next to Unreal Engine model	78
Figure 87. Model skeleton.....	79
Figure 88. Model Pysics.....	79
Figure 89. Final Unreal Engine model	79

1. Preface

During this project we took part in the European Project Semester (EPS)ⁱⁱⁱ at Novia University of Applied Sciences. The EPS project started in 1995 in Denmark. It was organized by Dr. Arvid Anderson and now nearly 25 years later the EPS project is provided by 19 universities all over Europe. EPS is a mixture of “Project Related Courses” with project organized and problem-based learning. During the EPS project students will work together in multinational teams, the focus lays on teamwork. Also, the student will work together with students of a different field of study to learn from each other. As a team you have to grow together to solve several engineering-based problems.

The chosen EPS project for this team is the InMoov project, where an existing humanoid robot will be optimized, see Figure 1. The InMoov project arose by an idea of the French sculptor and designer Gael Langevin. It is the bridge between 3D printing enthusiasts and students in engineering fields. InMoov is the first open source 3D printed life-size humanoid robot consisting of servos, Arduinos, microphone, camera, Microsoft Kinect 3D sensors and computer. InMoov is powered by MyRobotLab and Java framework. During this project these hardware and software parts will be further optimized and revised.ⁱⁱ Also will VR be implemented to the InMoov humanoid robot.



Figure 1. InMoov humanoid Robot

The project team consists of a multi-disciplinary and multi-cultural team. Every member of the team has their own strengths to bring to the project and the team will be supported by Professor Rayko Toshev, of the University of Vaasa. The students participating in this team will be further elaborated in this paragraph, their motivation and background to the project will be explained. Also, Figure 2 shows an overview with the divided Belbin roles, each member of the team did the Belbin test and from that the roles were divided. By coincidence all the Belbin roles were divided over the team and no one had to take any extra roles upon them.^{iv}

Britt van Bergen

Britt is the team leader of this team, she originates from The Netherlands and studies Product Development. She chose the InMoov project based on previous knowledge about 3D printing

and her interest in humanoids and robotics. Britt was chosen as the team leader because of her ability to make fast decisions and organizational skills.

Brent Van Obbergen

Brent is the Secretary of this team, he originates from Belgium and studies Electro mechanic Engineering and Automation. His choice for the InMoov project is based on his interest in specified projects. This project offers him the benefit to learn a new way of programming to what he was learned before. Brent was chosen to be the Secretary because of his ability to cope with many opinions and multitasking within the team.

Oliver Heijster

Oliver is the planner of this team, he originates from Belgium and studies Electromechanical Engineering and Process Automation. His choice in the project is based on the liking of combining his knowledge about mechanics and electronics with his passion about automation and robotics. Oliver is the planner since his involvement in projects of similar magnitude, therefore he knows how to create a good plan and stick to it.

Niki Kuhar

Niki is the Creative Director of the team, he originates from Slovenia and studies Industrial Design. He chose the InMoov project because of the connection of different fields of engineering that the project entrains like mechanics, programming, electrical, electronics and the design part.

Ana Sánchez

Ana is a team player, she originates from Spain and studies Industrial Electronic and Automatic Engineering. In order to practice what she learned during her studies she chose the InMoov project. She knows much about electricity, programming and working with hardware, therefore she is the team player of the group.

Britt van Bergen The Netherlands	Brent Van Obbergen Belgium	Oliver Heijster Belgium	Niki Kuhar Slovenia	Ana Pérez Torres Spain
Teamleader	Secretary	Planner	Creative Director	Teamplayer
Roles: Plant Coordinator Evaluator	Roles: Teamworker Completer-Finisher	Roles: Plant Monitor Coordinator Resource Investigator	Roles: Specialist Shaper Resource Investigator	Roles: Completer-Finisher Evaluator Implementer

Figure 2. Belbin roles

2. Introduction

InMoov is the first Open Source 3D printed life-size robot. It began as a personal project of Gael Langevin, a sculptor and designer. InMoov is thought as a development platform for universities, laboratories and makers. The robot is replicable on any home 3D printer, and the development of the software is based on sharing by forums, the InMoov website and the comments there. That means that with the same base, there are countless projects throughout the world, since the final purpose of the robot is up to the person who is creating it.

This project is useful for the students, because there are many engineering fields involved: mechanics, electronics, electricity and design. It also has a utility for the university, as the final purpose of the robot can serve as a presentation forum of the university. Finally, working on this project is helpful for the InMoov community, as the problems and the solutions found are shared in the forums.

The project that the students work on this semester has already started a few years ago by previous students. Almost all the parts are already printed, except for a pair of broken parts, which have to be repaired or replaced during this semester. The servos were supposed to work but the zero position was not right, and the eye tracking and the Kinect had to be improved as well. The first thing to do, after exploring the whole system, was repairing the servos and the broken plastic parts. Once the robot can move correctly, the second step was to work with the eye tracking and the Kinect. At the same time, the Virtual Reality was further researched and the utility of the robot was explored. Later on in the project when the main problems of the robot were solved, code was written to make the humanoid robot give a presentation.

The final goal of the project is for the robot to serve with a purpose, and to give a demonstrating presentation of this purpose. Within this presentation the hardware and software parts will serve as support to achieve this, like using the finger pressure sensor, eye tracking and VR for example.

3. Project Management

3.1 Mission and Vision

The mission defines the project's objectives and how to achieve them. The mission of this project is to make the humanoid's movements more precise and complete from the initial point to the end point, getting the robot to recognize and follow objects and to interact with humans.

The vision describes the desired goal for the humanoid robot in this case. The vision is to achieve a humanoid robot that could be useful in the society within a certain utility and with a purpose.

3.2 Input and Output

In all projects there are some inputs that revolve in tools & techniques to create an output, which usually is the final goal of the project, as we can observe in Figure 3. In the case of the InMoov project, the project was already started so the main input was the humanoid, but also outputs from planning and analyzing the robot's functionalities. Then the tools and techniques available are the IT tools (Information technology tools) as google drive, dropbox, InMoov website and forum, in addition to myrobotlab software and Technobothnia facilities. So, with these inputs and resources the output expected is to complete the mission and in the best situation realize the vision.



Figure 3. Input, tools and techniques, output

3.3 Work Breakdown Structure

In the Work Breakdown Structure (WBS) all the tasks for this project are summarized. The WBS is divided in four different sections; Hardware, Software, Virtual Reality and Management. These sections are then further divided in subsections and tasks for this project. This WBS gives an overview of the main tasks and make it more manageable, the WBS is shown in Figure 4.

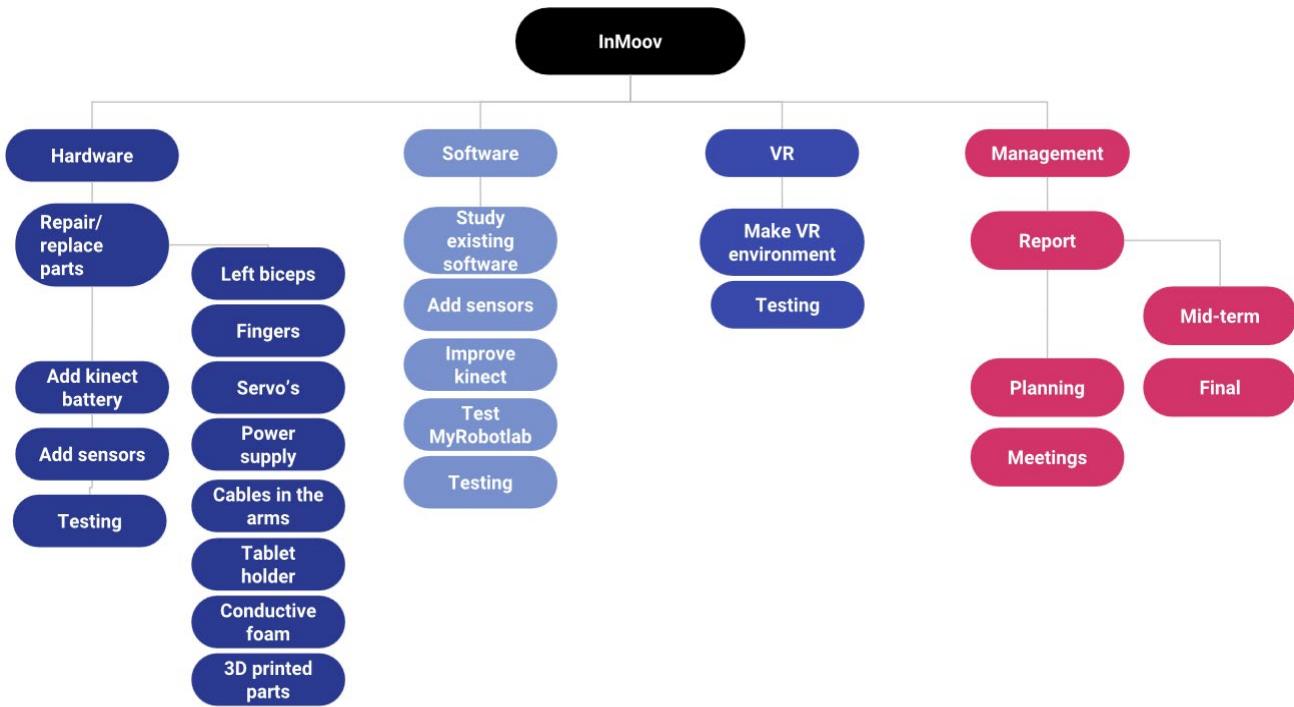


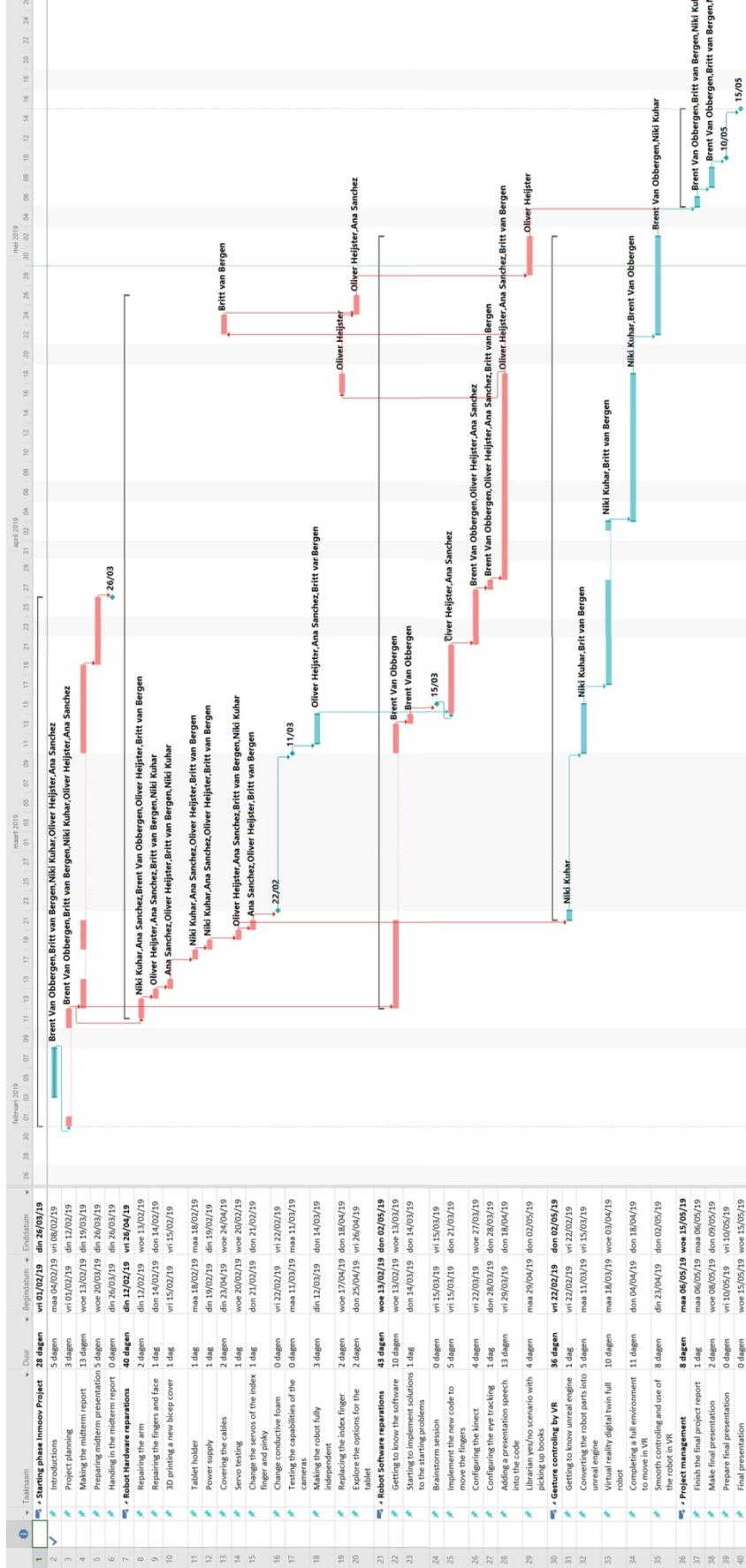
Figure 4. Work Breakdown Structure

3.4 Schedule

At the beginning of this project a schedule was conducted, containing all the tasks from the WBS. This schedule was made with Microsoft Project Software. Within the schedule a timetable was developed to manage the time of execution of every main activity and describe what each activity is composed of. In Figure 5 the tasks in the schedule for the InMoov project are shown and in Figure 6 on the next page the detailed Gantt Chart is presented.

Introductions	5 dagen	maa 04/02/19	vri 08/02/19		Brent Van Obbergen,Britt van Bergen,Niki Kuhar,Oliver Heijster,Ana Sanchez
Project planning	3 dagen	vri 01/02/19	din 12/02/19	2	Brent Van Obbergen,Britt van Bergen,Niki Kuhar,Oliver Heijster,Ana Sanchez
Making the midterm report	13 dagen	woe 13/02/19	din 19/03/19	3	
Preparing midterm presentation	5 dagen	woe 20/03/19	din 26/03/19	4	
Handing in the midterm report	0 dagen	din 26/03/19	din 26/03/19	5	Ana Sanchez,Brent Van Obbergen,Britt van Bergen,Niki Kuhar,Oliver Heijster
Robot Hardware reparations	40 dagen	din 12/02/19	vri 26/04/19		
Repairing the arm	2 dagen	din 12/02/19	woe 13/02/19	3	Niki Kuhar,Ana Sanchez,Brent Van Obbergen,Oliver Heijster,Britt van Bergen
Repairing the fingers and face	1 dag	don 14/02/19	don 14/02/19	8	Oliver Heijster,Ana Sanchez,Britt van Bergen,Niki Kuhar
3D printing a new bicep cover	1 dag	vri 15/02/19	vri 15/02/19	9	Ana Sanchez,Oliver Heijster,Britt van Bergen,Niki Kuhar
Tablet holder	1 dag	maa 18/02/19	maa 18/02/19	10	Niki Kuhar,Ana Sanchez,Oliver Heijster,Britt van Bergen
Power supply	1 dag	din 19/02/19	din 19/02/19	11	Niki Kuhar,Ana Sanchez,Oliver Heijster,Britt van Bergen
Covering the cables	2 dagen	din 23/04/19	woe 24/04/19	28	Britt van Bergen
Servo testing	1 dag	woe 20/02/19	woe 20/02/19	12	Oliver Heijster,Ana Sanchez,Britt van Bergen,Niki Kuhar
Change the servos of the index finger and pinky	1 dag	don 21/02/19	don 21/02/19	14	Ana Sanchez,Oliver Heijster,Britt van Bergen
Change conductive foam	0 dagen	vri 22/02/19	vri 22/02/19	15	Oliver Heijster,Ana Sanchez
Testing the capabilities of the cameras	0 dagen	maa 11/03/19	maa 11/03/19	16	Oliver Heijster,Ana Sanchez,Britt van Bergen
Making the robot fully independent	3 dagen	din 12/03/19	don 14/03/19	17	Oliver Heijster,Ana Sanchez,Britt van Bergen
Replacing the index finger	2 dagen	woe 17/04/19	don 18/04/19	28	Oliver Heijster
Explore the options for the tablet	2 dagen	don 25/04/19	vri 26/04/19	19,13	Oliver Heijster,Ana Sanchez
Robot Software reparations	43 dagen	woe 13/02/19	don 02/05/19		
Getting to know the software	10 dagen	woe 13/02/19	woe 13/03/19	3	Brent Van Obbergen
Starting to implement solutions to the starting problems	1 dag	don 14/03/19	don 14/03/19	22	Brent Van Obbergen
Brainstorm session	0 dagen	vri 15/03/19	vri 15/03/19	23	
Implement the new code to move the fingers	5 dagen	vri 15/03/19	don 21/03/19	24,18	Oliver Heijster,Ana Sanchez
Configuring the kinect	4 dagen	vri 22/03/19	woe 27/03/19	25	Brent Van Obbergen,Oliver Heijster,Ana Sanchez
Configuring the eye tracking	1 dag	don 28/03/19	don 28/03/19	26	Brent Van Obbergen,Oliver Heijster,Ana Sanchez,Britt van Bergen
Adding a presentation speech into the code	13 dagen	vri 29/03/19	don 18/04/19	27	Oliver Heijster,Ana Sanchez,Britt van Bergen
Librarian yes/no scenario with picking up books	4 dagen	maa 29/04/19	don 02/05/19	20	Oliver Heijster
Gesture controlling by VR	36 dagen	vri 22/02/19	don 02/05/19		
Getting to know unreal engine	1 dag	vri 22/02/19	vri 22/02/19	14	Niki Kuhar
Converting the robot parts into unreal engine	5 dagen	maa 11/03/19	vri 15/03/19	31	Niki Kuhar,Britt van Bergen
Virtual reality digital twin full robot	10 dagen	maa 18/03/19	woe 03/04/19	32	Niki Kuhar,Britt van Bergen
Completing a full environment to move in VR	11 dagen	don 04/04/19	don 18/04/19	33	Niki Kuhar,Brent Van Obbergen
Smooth controlling and use of the robot in VR	8 dagen	din 23/04/19	don 02/05/19	34	Brent Van Obbergen,Niki Kuhar
Project management	8 dagen	maa 06/05/19	woe 15/05/19		
Finish the final project report	1 dag	maa 06/05/19	maa 06/05/19	35,29	Brent Van Obbergen,Britt van Bergen,Niki Kuhar,Oliver Heijster,Ana Sanchez
Make final presentation	2 dagen	woe 08/05/19	don 09/05/19	37	Brent Van Obbergen,Britt van Bergen,Niki Kuhar,Oliver Heijster,Ana Sanchez
Prepare final presentation	0 dagen	vri 10/05/19	vri 10/05/19	38	
Final presentation	0 dagen	woe 15/05/19	woe 15/05/19	39	Brent Van Obbergen,Britt van Bergen,Niki Kuhar,Oliver Heijster,Ana Sanchez

Figure 5. Tasks schedule



3.5 Human Resource plan

In addition to the WBS and the schedule for this project, a RACI chart had been made to ensure clear distribution of the different roles and responsibilities within the project. A list with all the tasks from the schedule has been established and everyone has been linked to different tasks on different levels of responsibility: Responsible (R), Accountable (A), Consult (C) and Inform (I). In the Figure 7 is the RACI chart for this project shown.

As the team leader, Britt van Bergen is mostly responsible for the project management tasks. And with her knowledge about Product Development she is mostly overcasting the hardware tasks. Niki Kuhar is mainly focusing on the tasks connected with Virtual reality during this project. Brent Van Obbergen is involved in both the hardware, software and VR tasks. Oliver Heijster and Ana Sánchez are mostly working together on both hardware and software tasks. Overall will all the team members help each other where needed. The supervisor during this Project, RaykoToshev, will be consulting the team when required.

Legend for the RACI chart:

Names:	Symbols:
Ana Sánchez	AS
Britt van Bergen	BB
Brent Van Obbergen	BO
Niki Kuhar	NK
Oliver Heijster	OH
RaykoToshev	RT

RACI Matrix		Project team					Supervisor
Project area	Activity	AS	BB	BO	NK	OH	RT
Hardware	Repairing the arm	A	A	R	I	A	C
	Repairing the fingers and face	A	A	A	I	R	C
	3D printing a new bicep cover	I	R	I	A	I	C
	Tablet holder	I	R	I	A	I	C
	Power supply	R	R	I	I	I	C
	Covering the cables	I	R	I	I	A	C
	Servos testing	A	I	A	I	R	C
	Change the servos of the index finger and pinky	A	I	R	I	A	C
	Change conductive foam	R	I	I	I	A	C
	Testing the capabilities of the cameras	I	I	R	I	A	C
	Making the robot fully independent	R	R	I	I	I	C
	Replacing the index finger	A	A	I	I	R	C
	Explore the options for the tablet	R	I	I	I	A	C
Software	Getting to know the software	R	I	A	I	A	C
	Starting to implement solutions to the starting problems	R	I	A	I	A	C
	Brainstorm session	R	A	R	R	R	C
	Implement the new code to move the fingers	I	I	A	I	R	C
	Configuring the kinect	R	I	A	I	I	C
	Configuring the eye tracking	I	I	R	I	A	C
	Adding a presentation speech into the code	A	I	A	I	R	C
Virtual Reality	Librarian yes/no scenario with picking up books	A	I	I	I	R	C
	Getting to know unreal engine	I	I	A	R	I	C
	Converting the Robot parts into unreal engine	I	I	I	R	I	C
	Virtual reality digital twin full robot	I	I	A	R	I	C
	Completing a full environment to move in VR	I	I	R	R	I	C
Project Management	Smooth controlling and use of the robot in VR	I	I	R	R	I	C
	Finish the final project report	R	R	R	R	R	C
	Make final presentation	A	R	A	A	A	C
	Prepare final presentation	A	R	A	A	A	C
	Final presentation	R	R	R	R	R	C

Figure 7. RACI Chart

3.6 Project budget

To calculate the total cost of this project several things need to be taken into account. First of all, the working hours. As prospective engineers, a wage of 25€/hour will be used. This will need to be multiplied by the amount of time spent on a specific part of the project and the amount of people working on that part. For example: repairing the fingers and the face was a task carried out by Oliver, Britt, Brent and Ana, they spent around 8 hours doing this so this equals a total cost of 800€ for this part of the project. After calculating this for every task, a total cost of 38.200,00 € is the result.

Next, the cost of materials, a big advantage here is that the robot already exists. The cost of materials will mainly come down to repairing and making improvements. The servo in the left bicep needed to be replaced, this came to a cost of 30€. New conductive foam was ordered since the old one did not function properly. This cost 2.45€. Some additional parts needed to be printed as well. This came to a total of around 1 kg of used ABS, 1kg costs around 21€. Usage of the 3D printers' costs money as well. Electricity in Finland costs 0.16€ per kWh, standard use of a 3D printer is 200W, printing the parts took around 80 hours.^v This gives a total of 16kWh and a total cost of 2.56€.

The total cost of the materials comes down to 56.01€

As a side note, building the robot from the ground up would have been a cost of 1920.51€.

The total cost of this project will be 38 256.01€. The specific costs are shown in table 2.

25 per hour per person	costs in hours	Product	Qty	Price (per qty)	Total	Details	Section
Robot Hardware reparations	8.400,00 €	Nervo Board	2	27,50 €	71,58 €		System
Repairing the arm	1.000,00 €	Arduino Mega	2	35,00 €	70,00 €		System
Repairing the fingers and face	800,00 €	HK15298B	10	15,00 €	150,00 €	medium servos for fingers	Arms
3D printing a new bicep cover	800,00 €	MG995	2	7,00 €	14,00 €	medium servos for wrists	Arms
Tablet holder	800,00 €	Braided fishing line	50	0,10 €	5,00 €	In pound; 0,8mm, used for	Arms
Power supply	800,00 €	White ABS	1,5	20,88 €	31,32 €	Filament in grams	Arms
servos testing	800,00 €	Conductive foam	1	2,45 €	2,45 €	5mm	Fingers
change the servos of the index finger and the pinky	1.200,00 €	HS805BB	8	30,00 €	240,00 €	large servos,	Shoulders
conductive foam	- €	Black ABS	1,5	20,88 €	31,32 €	Filament in grams	Shoulders
testing the capabilities of the cameras	600,00 €	HS805BB	2	30,00 €	60,00 €	medium servos for head mc	Head and Torso
adding sensors and testing them	1.600,00 €	HK15298B	1	15,00 €	15,00 €	small servo for jaw	Head and Torso
Robot Software reparations	10.800,00 €	DS929hv	3	14,00 €	42,00 €	micro servo for eyes	Head and Torso
Getting to know the software	2.000,00 €	Hercules HD	1	20,00 €	20,00 €	camera for one eye	Head and Torso
Starting to implement solutions to the upstarting problems	200,00 €	Microsoft LifeCam HD-3000	1	30,00 €	30,00 €	camera for one eye	Head and Torso
Implement the new code to move the fingers	3.000,00 €	Kinect XBOX 360	1	30,00 €	30,00 €	3d camera for torso	Head and Torso
configuring the cinnect	2.400,00 €	White/Clear ABS	5,5	20,88 €	114,84 €	Filament in grams	Head and Torso
configuring the eye tracking	800,00 €	6V12AH	3	18,00 €	54,00 €	Batteries	Head and Torso
Adding a presentation speach into the code	2.400,00 €	eFuel switch Power supply	1	300,00 €	300,00 €	6V-15V Output 20Amps	Head and Torso
Gesture controlling by VR	5.000,00 €	6v Power supply	1	40,00 €	40,00 €	300watts/50amps	Head and Torso
Getting to know unreal engine	200,00 €	Mini Speaker	2	20,00 €	40,00 €	4Ohm 6W	Head and Torso
Converting the Robot parts into unreal engine	1.000,00 €	PIR Sensor	1	2,00 €	2,00 €		Head and Torso
Virtual eality digital twin	3.800,00 €	HS805BB	2	30,00 €	60,00 €	medium servos for stomach	High stomach
Project management	14.000,00 €	HS805BB	2	30,00 €	60,00 €	medium servos for stomach	Mid stomach
Writing our project thesis	7.000,00 €	Ribbon Cable	1	10,00 €	10,00 €	200cm roll at least;	
Making a powerpoint for our presentation	4.000,00 €	Vigor VSD-11AYMB	2	20,00 €	40,00 €	Extra Large Winch Servo	waist
Devide our presentation	1.000,00 €	CYS 8503	2	24,00 €	48,00 €	30 kg Digital Metal Gear Ser	waist
Presenting our project	- €	Bearing Grease	1	13,00 €	13,00 €	16oz Can	
project planning	2.000,00 €	8 inch tablet	1	300,00 €	300,00 €	NuVision TM800W610L Tablet	
		Powerbank 10000mAh 5V	1	26,00 €	26,00 €		
Total cost of time	38.200,00 €				1.920,51 €		40.120,51 €

Figure 8. Project Costs

3.7 Risks analysis

In order to be prepared during this project a risk analysis is implemented where the most critical tasks of the problem have been put into consideration. These critical points were selected with the entire team and will be kept in mind during the entire project. The main risks within this project are;

1. Software
2. Hardware
3. Virtual Reality
4. Omron

The first difficulty encountered was getting to know the software because in the team there wasn't anyone with a good understanding of programming. Besides that, working on a project that has already started gave some difficulties since there were parts like the servos that were configured in a way, which was not correct.

Once the project started, the main risk was to break parts of the covers by testing the servos and the Kinect, as that caused additional work and certain delays in the planning. However, the big risk of the project is implementing new devices like the Virtual Reality and the OMRON. The risk with VR is the lack of knowledge about the system and the how to program it. Then, the difficulty with the OMRON is the same as the VR, and moreover it is being used in another project, so it is not fully available for the InMoov project.

To handle these risks a priority list is made, this list calculates the probability of the risk becoming a reality for the project. And the impact the risk will have on the project. Both factors are estimated on a scale of 1 till 5, in Figure 9 is the priority list shown.

Risk	Probability	Impact	Total:
Software	2	2	4
Hardware	3	2	6
Virtual Reality	4	3	12
Omron	4	3	12

Figure 9 Priority list

This priority list shows where to be most careful during the project. In Figure 10 is an assessment analysis shown where it states which action needs to be undertaken to minimize the impact of the risk

Priority	Total risk	Assessment
High	17 - 25	Immediate requirement to review and investigate the case for removing/reducing the risk or improving the control measures.
Medium	9 - 16	Risks not broadly acceptable. Need investigation to consider reasonably practicable improvements to achieve ALARP (as low as reasonably practicable).
Low	1 - 8	Detailed working to support conclusion.

Figure 10. Assessment analysis

When analyzing the priority list and the assessment analysis it concludes that the software and hardware have a low risk and do not need as much attention during this project. Virtual Reality and Omron on the other hand have a medium risk and should be kept in consideration at all time. At some point within the project a decision will have to be made on how to continue this part of the project. The team leader will keep this in mind and will inform the team when this decision has to be made.

3.8 Quality management

To maintain a certain quality during this project a quality assessment has been made in the shape of a fish diagram, see Figure 11. The desired effect is simple; Quality. To achieve this effect the causes have been established for this project; People, Communication, Process and Tools.



Figure 11. Quality assessment fish diagram

Quality

The desired effect is a high quality, in this case the humanoid robot is the result of the project and will require a high quality. Quality is a vague concept and will therefore be explained using different examples concerning the humanoid robot. With the presentation of the robot it will be important that the gestures and conversations go well organized and make sense for the human it is interacting with. This presentation is divided in two dimensions; software and hardware. The software will mainly be responsible for these gestures and conversations. The hardware will entail the aesthetic appearance of the robot. The expected quality of the 3D printed parts for example is a part that has a nice color and a smooth finish, see Figure 12 for an example of the print quality.

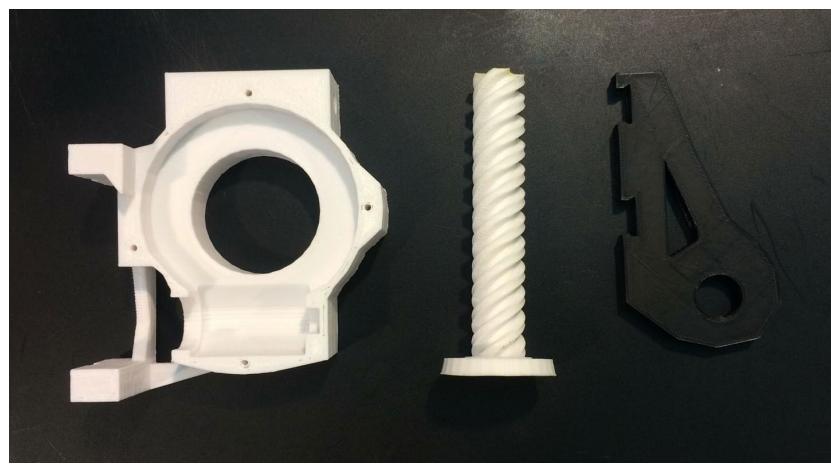


Figure 12. 3D print quality

People

Here is referred to the team members and the stakeholders involved in this project. Each person has a role to play. If one of the members of the team does not function as supposed to this can influence the quality of the result. Dysfunction of a person can be caused by lack of motivation or lack of skills. In either of these cases communication is the best way to solve this.

Communication

As mentioned before, communication within the team can solve many problems, which will result in a better quality. Dysfunction of a team members for example can be solved with communication. If someone does not have the skills to accomplish a task and communicates this to the team, this person will likely find someone who can help. Or when there is a lack of motivation and this is recognized by other team members they can communicate about how to change this attitude.

Process

The process of how to undertake a task is of influence of the required quality. A well-organized process can prevent mistakes.

Tools

In the case of printing parts with the 3D printer is the tool of high influence of the quality. The same applies for the software tasks, with the right software programs working will go more smoothly and will result in better quality.

3.9 Change control

This change control analysis is designed to identify the main proposed changes and track those accepted throughout the project. Every team member and the supervisor can propose a change, and this will be discussed with the team.

As a team leader Britt is responsible for overseeing the different aspects of the project and make sure to raise the question if change is necessary. The team leader also functions as the gatekeeper, this is someone who decided if a change should be executed or not. When a change will be made all stakeholders should be informed as communication is key in change control. In Figure 13 is the change control log shown were all changes are organized in an overview. In Figure 14-19 on the next pages is for each change a change control form showed, here are the objective statement, the description and reason of change further described.

Change Number	Date of Change	Description of Change	Requested By	Status O/C	Schedule Impact
1	18/02	Covering the cables	The team	C	11 weeks
2	26/03	Omron	The team	C	0 weeks
3	26/03	Kinect	The team	C	0 weeks
4	26/03	Virtual Reality task	Supervisor	C	0 weeks
5	02/04	Virtual Reality glasses	Supervisor	C	0 weeks
6	24/04	Tablet	The team	C	11 weeks

Figure 13. Project change control log

Subject: Covering the cables 2019	Date Revised: 18 - 02 -
Objective statement: The cables that are used to control the arms are very visible and open. These cables are planned to lead through the bicep to make it look more clean and organized.	
Description and reason of Change: There is not enough space in the biceps to place the cables there. Covering the cables with cable covers will give a clean look and will therefore change the task a bit. It has also been decided to move this task further to the back of the project because it is not the most important part.	

Figure 14. Project change control form 1

Project title: InMoov Humanoid Robot 2019	Date: 11 - 02 -
Subject: Omron 2019	Date Revised: 26 - 03 -
Objective statement: The Omron was supposed to be connected to the humanoid robot to make it move independently.	
Description and reason of Change: The Omron project has been removed from the tasks. The reason for this is that other groups were working with the Omron and therefore the Omron was occupied. Also, after making a decision on the scenario the Omron would not bring any more added value.	

Figure 15. Project change control form 2

Project title: InMoov Humanoid Robot 2019	Date: 11 - 02 -
Subject: Kinect 2019	Date Revised: 26 - 03 -
Objective statement: The Kinect can be used to make the robot copy movements.	
Description and reason of Change: Optimizing the Kinect has been removed from the tasks because it was not useful for the chosen scenario. And with the VR the same copying of the movements can be done and is less fragile.	

Figure 16. Project change control form 3

Project title: InMoov Humanoid Robot 2019	Date: 11 - 02 -
Subject: Virtual Reality task 2019	Date Revised: 26 - 03 -

Objective statement:

When dividing the tasks it was Niki who would work mostly with Virtual Reality.

Description and reason of Change:

After the midterm period it has been decided to also address Brent to this task. They complement each other's skills and therefore will make more progress for the VR.

Figure 17. Project change control form 4

Project title: InMoov Humanoid Robot

2019

Subject: Virtual Reality glasses

2019

Date: 11 - 02 -

Date Revised: 02 - 04 -

Objective statement:

At first it was planned to use the HTC 5 VR glasses.

Description and reason of Change:

These HTC 5 glasses are also used by other students and therefore the Microsoft Augmented Reality glasses will be used from now on. These give a better opportunity to move around freely in the building.

Figure 18. Project change control form 5

Project title: InMoov Humanoid Robot

2019

Subject: Tablet

2019

Date: 11 - 02 -

Date Revised: 24 - 04 -

Objective statement:

The tablet has several problems that needed to be fixed. MyRobotLab would not properly open on the tablet and there was no communication with the Arduino.

Description and reason of Change:

After multiple attempts of fixing the tablet it has been decided that the tablet will not be fixed. The problem most likely is the tablet itself, it does not have the right properties to work with.

Figure 19. Project change control form 6

4. Philosophy presentation

4.1 Competitor research

A few weeks in the project it was time to think about the philosophy of the humanoid robot. Therefore, a brainstorm session was organized, all the team members and the supervisor were invited for this. To prepare the brainstorm session everyone did research about other humanoid robots and their philosophy to get an overview of what is out there. Here are some of the main competitors showed in Figure 20-24.

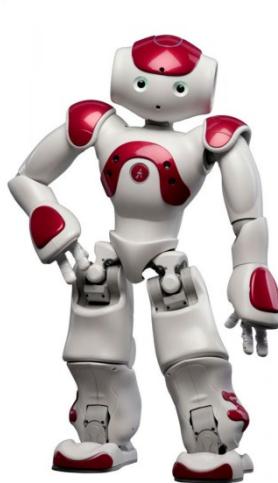


Figure 20. Nao humanoid robot



Figure 21. Romeo humanoid robot



Figure 22. Pepper humanoid robot



Figure 23. DCR-HUBO



Figure 24. Robothespian

Most humanoids on the market have similar functions. These functions are to move autonomously, robots can move in a space avoiding obstacles and without having to be driven or controlled by anyone. Another very important function is to interact with people, they listen and answer.

One good example is Nao created by Softbank group in 2008.^{vi} It can walk, recognize people and objects, talk and interact with people. The speech recognition program has 20 different languages and the code is entirely open and programmable. The first idea was to use Nao to work with autistic children, but then other functions were explored like being an assistant in health care centers, work in information desks or function as a guide.

Another example is Romeo, launched in 2009.^{vii} It was created as a research platform suitable for testing possible service uses that could be included in future robot companions, specialized in supporting elderly or disabled persons. Romeo is equipped with four computers to manage its sight, hearing, movements and artificial intelligence. Romeo incorporates a mass of innovations required for its future role as a personal assistant.

The humanoid Pepper is also programmed to learn the habits of the person it is taking care of, it was created by Softbank in 2014.^{viii} Its main function is to give information and to guide. To accomplish its functions, it has speech recognition and dialogue in 15 languages, perceptions modules to recognize and interact with people, infrared sensors, bumpers, an inertial unit, 2D and 3D cameras, and sonars for omnidirectional and autonomous navigation. Pepper is 120cm tall and has a curvy design which ensures the security of the users, it also has a touch screen on its chest to add information and support its speech.

Engineered Arts created in 2010 a humanoid with similar characteristics to the previous ones but with a different purpose. This robot is RoboThespian, it was created with the idea of being a teacher or actor, which is the most famous function it has.^{ix} Its added value is the expressivity of the face, instead of being a static face as Pepper, for example, the face is a screen which can show different faces easily. This face also gives the opportunity to customize the robot.

DRC-HUBO is a humanoid with a different objective, it was entered by team KAIST to the DARPA Robotics Challenge, and it was the winning robot.^x This competition aimed to develop semi-autonomous ground robots that could do "complex tasks in dangerous, degraded, human-engineered environments". DRC-Hubo, is an adaptable multifunctional device with the ability to transform from a walking robot to rolling on four wheels by bending and using wheels incorporated into the knees. These wheels made it faster and stable, but thanks to the legs it was able to climb stairs and drive a car. The humanoid was created in 2015.

4.2 Brainstorm session

At the beginning of the brainstorm session everybody first shared their research findings of the other humanoid robots. Then the features of the InMoov humanoid robot were listed to have an overview of the strengths to use. The brainstorm session continued in a mind map, on a big whiteboard different kind of fields were listed and explored concerning the philosophy purpose of the InMoov robot. In Figure 25 is a photo shown of the brainstorm session.



Figure 25. Brainstorm session

The mind map is divided in six different fields: Education, Informative, Basic tasks, Fun, Research and Help. Within these fields many ideas for the robot were explored. In Figure 26 is an info graphic shown with the mind map that originated from the brainstorm session.

After setting up this mind map everyone was asked to have a critical look at all the ideas and select the most feasible and interesting ideas. This resulted in two fields that were most interesting to explore further; education and informative. This will be explored further in the next paragraph using scenarios.

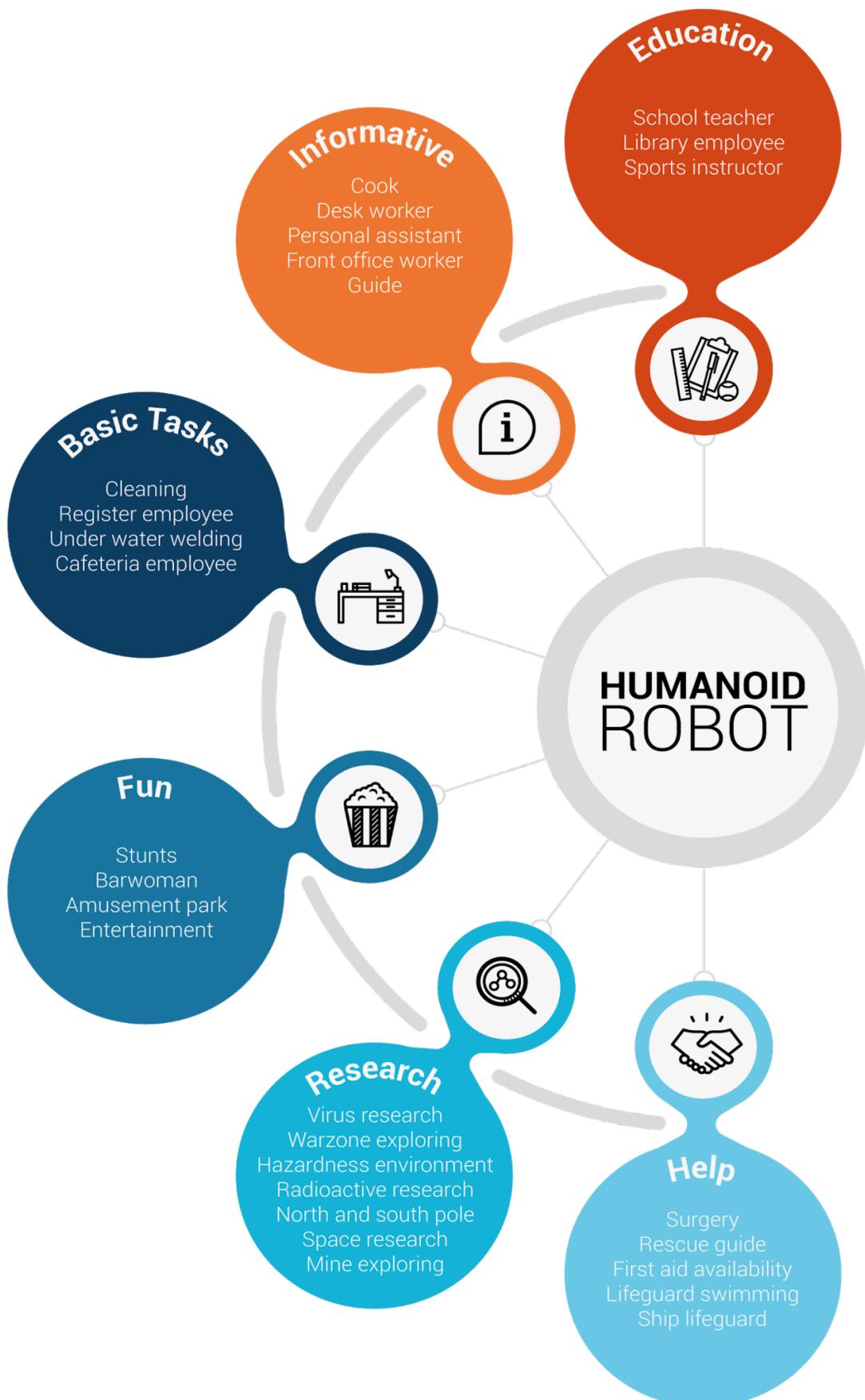


Figure 26. Brainstorm Mindmap

4.3 Humanoid scenarios

As a result, from the brainstorm session with the team, came forward that education and information would be the best fields for the InMoov humanoid robot to work. Within these fields the most feasible scenarios would be an airport worker, a guide, a librarian or an assistant teacher. These four examples will be explained more in the following scenarios. The cost reduction for these jobs should already be interesting enough but a robot will also work 24/7, will do exactly as programmed and would not show any unexpected emotions.

Airport worker

If the humanoid worked at the airport it will help customers revise or book a flight and give information when asked. These tasks are very static and do not require any emotions. As an airport worker, the humanoid could work always without any breaks and stay in one spot without moving. This aligns very good with the capabilities of the InMoov humanoid. Only hardware to scan the flight-ID and a payment method need to be added for this purpose. If this scenario will be applied it would have influence on the following hardware components; eye tracking, voice command, PIR sensor, gestures and finger pressure sensor. Here are two examples showed of how to use this humanoid in a certain situation.

Situation: Info desk

Robot: Welcome at Vaasa Airport, how can I help you?
Customer: I would like to do the check-in with SAS, where can I find the check-in desk?
Robot: You can find the SAS check-in desk if you go left here, and then the first to the right. You will see the desk on your right. (Points to right and then left)
Customer: Thank you!
Robot: You're welcome. Have a nice flight.

Situation: Check-in desk

Robot: Welcome at the SAS check-in desk, where are you flying to?
Customer: I will be going to Spain.
Robot: That is great, can I have your passport please?
Customer: (Customer gives passport)
Robot: Thank you, (takes passport from desk) would you like to check in any luggage?
Customer: Yes I have one suitcase here. (Shows suitcase)
Robot: Please put it on the treadmill here. (Points at treadmill)
Customer: (Puts suitcase on the treadmill)
Robot: Thank you, here is your passport back and the boarding pass. Please be at gate G6 at 4 pm to board. Have a nice flight. (Gives passport and boarding pass)
Customer: Thank you! (Takes passport and boarding pass)

Guide

For the humanoid to function as a guide this needs to move, this can be achieved by adding the Omron robot. For this purpose, the human features will be used to show certain places in a building for example. As a start this could be implied in the Technobothnia building where the humanoid could point to the toilets, nearest exit, etc. when asked. The humanoid can also give a tour by moving through the building. If using this scenario, it would have influence on the following hardware components; eye tracking, voice command, PIR sensor, Omron and gestures. Here are two examples showed of how to use this humanoid in a situation.

Situation: Standing on the entrance of the building

Robot: Welcome to Technobothnia, how can i help you?
Customer: Where can I find the toilet?
Robot: The toilet is on your right; the first door is the men's toilet and the second is for women. (Points to the left)
Customer: Thank you!
Robot: You're welcome, have a nice day.

Situation: Guide of the building

Robot: Welcome to Technobothnia, would you like to do a tour through the building?
Customer: Yes that would be great.
Robot: (Start moving) Okay come with me. If you look on your left here you see the working spaces for the students (point to left). What would you like to see more?
Customer: I would like to see the 3D printers.
Robot: Then we go to the left here (turns left). We have several different 3D printers at Technobothnia, let's go in here (turn left). This is the 3D printing lab.
Customer: So nice, I will stay here for a while. Thank you for the tour.
Robot: You're welcome.

Librarian

Another scenario could be the library where the same appliances are required as from the airport worker. The humanoid would be able to give information and direction on where to find certain books. It could also take information of previous written books and use this to give recommendations for other books. If using this scenario, it would have influence on the following hardware components; eye tracking, voice command, PIR sensor, gestures and finger pressure sensor. Here are two examples showed of how to use this humanoid in a situation.

Situation: At the info desk

Robot: Welcome to Tritonia, how can I help you?
Customer: I'm looking for some books about humanoids.
Robot: Those books are on the technologic department, which is on the first floor next to the stairs. (Points to stairs)
Customer: Thank you!
Robot: You're welcome, have a nice day.

Situation: At the info desk

Robot: Welcome to Technobothnia, how can i help you?
Customer: I would like to return these books.
Robot: Thank you, you can give them to me. Can I give you recommendations for your next read? (Takes books)
Customer: That would be great, I liked this book a lot, do you have something like this? (Points at book)
Robot: In that case I think this would be a good book for you, Robopocalypse. It is written by Daniel H. Wilson. Would you like to try it?
Customer: Yes that would be good, where can I find it?
Robot: You can find it at the second floor at the robotics section. Look for the letter W for Wilson. (Point to stairs)
Customer: Thank you, I'll find it!
Robot: You're welcome, have a good read.

Teacher's assistant

Another purpose for the humanoid to have would be in the scenario as a teacher's assistant. The humanoid could be of help by having a look at the students during class or in the breaks. Or it could answer any questions and spark the student's curiosity for robots. To adapt the humanoid to this scenario no hardware components are necessary, but it could be interesting to add a projector. If using this scenario, it would have influence on the following hardware components; eye tracking, voice command, PIR sensor, projector and gestures. Here are two examples showed of how to use this humanoid in a situation.

Situation: In the classroom

Robot: Hello, how can I help you?
Customer: Would you like to play with me?
Robot: Of course, what game would you like to play?
Customer: I like rock, paper, scissors!
Robot: Okay you start; choose rock, paper or scissors.
Customer: Rock! (Shows rock)
Robot: I have scissors, so you win, good job! Let's try again. (Shows scissors)
Customer: Paper! (Shows paper)
Robot: I have scissors again, so I win! The score is 1-1. (Shows scissors)
Customer: Let's go one more time. Paper! (Shows paper)
Robot: I have rock, so you win, well done! (Shows rock)
Customer: Yes.
Robot: That was a great game, thank you!
Customer: Thank you for playing with me!

Situation: In the classroom

Robot: Hey, how can I help you?
Customer: I would like to learn about geography, could you teach me?
Robot: Of course, if you look here you see the map of the world. Could you point to America for me? (Project world map)
Customer: I think it is here. (Points at America)
Robot: That is correct, and now a bit more difficult, where is Mexico?
Customer: I think here. (Points at map wrong)
Robot: That is Brazil, here is Mexico. (Points to Mexico)
Customer: I understand, thank you!
Robot: You're welcome, if you want to learn more just come to me.
Customer: I will!

4.4 Conclusion philosophy presentation

After a competitor research, a team brainstorm session and looking deeper into the idea's using scenarios, one of the scenarios was chosen to develop. This choice was made in consultation with the team based on many factors. The team considered the added value for the university and the people it will be serving. Also, the amount of hardware and software that need to be adjusted to achieve the scenario played a role in the choice. With these factors in mind the team made a decision together. The team unanimously chose for the librarian scenario, see Figure 27 for a visualization of the chosen scenario.

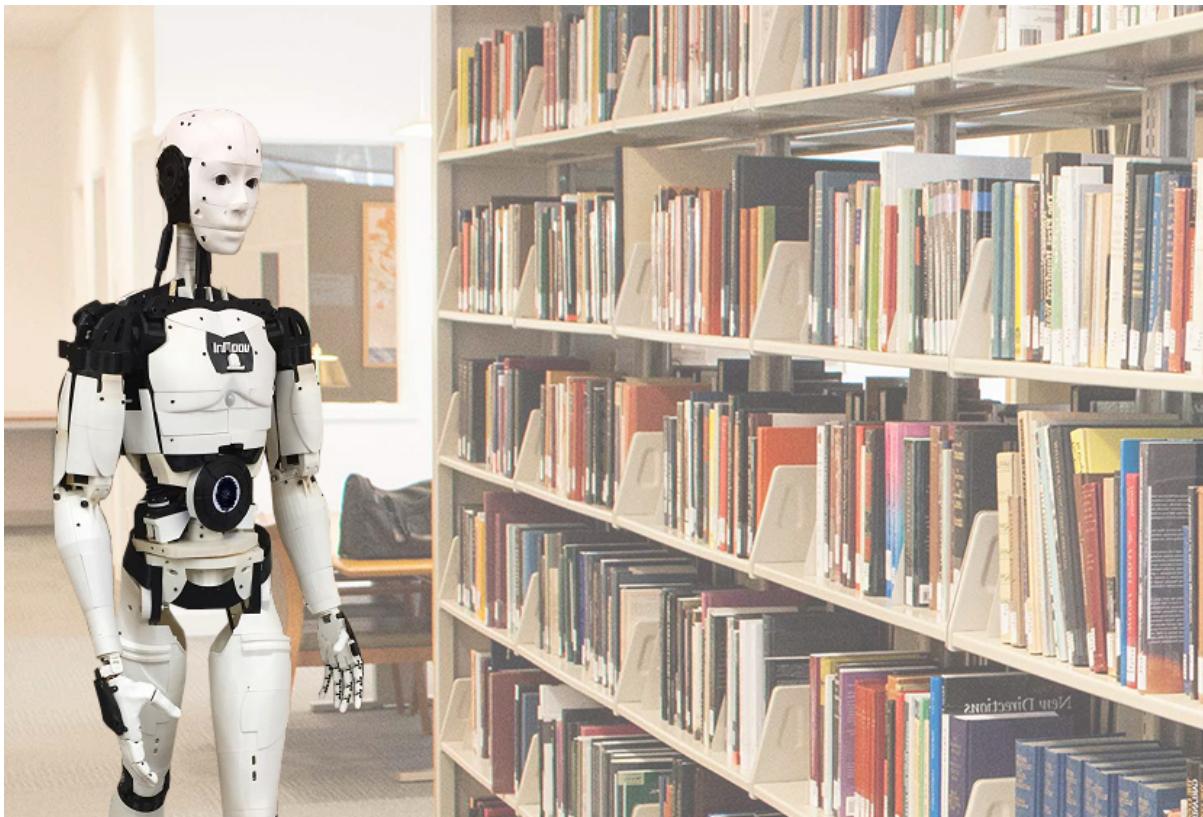


Figure 27. Librarian scenario

The result of this project will show in a demonstration of the librarian scenario. To achieve this scenario and optimize it, the hardware, software and VR parts need many adjustments. These adjustments will be explained in the next chapters.

5. Improving hardware

5.1 Repairing 3D printed covers

One of the basic tasks defined in the beginning of the project was the reparation and upgrading of the robot parts, mostly through the process of modeling and 3D printing parts. For the robot, most of the parts were already modeled by the original author, thus the remaining task was the 3D printing part.

The left bicep covers of the humanoid robot were missing and needed to be replaced. These covers give shape to the robot and cover up the cables that go through the arm. The missing parts were identified and printed with the Prusa 3D printer. During the printing of parts some problems appeared with the ABS material in the 3D printer. Whenever the print went wrong a blob originated as shown in Figure 28. The cause of these problems was mainly from the print settings. The temperature, speed and brim were expected to be most critical for the printing process. By changing these settings and learning from them the quality of the prints increased.

When all the parts were printed, these needed to be sanded and glued together. Acetone was used to glue the parts together since ABS melts slightly when exposed to the acetone, which resolves in one solid piece. After melting the right parts together, these could be clicked to the base of the robot's bicep. See Figure 29 and 30 below for the repaired bicep.



Figure 28. Blob 3D printer



Figure 29. Bicep before repairing



Figure 30. Bicep after repairing

5.2 Hardware revisions

At the beginning of the project it was clearly visible that the robot had many hardware problems. Many parts were broken, badly calibrated or the limits were not correctly configurated. The first task was repairing the hardware, because without the mechanic part of the robot there is no way to test the software.

Reparation of the neck

The first thing was the neck of the robot, every time it booted up the robot bended its neck upward. Repairing this did not take a long time, it was done by taking the neck bolt out and turning it 90°. Because then the programmed zero position stayed the same but the physical zero point changed to the middle of the neck, this is shown in Figure 31.



Figure 31. Neck humanoid robot

Reparation of the fingers

After finishing the neck, the next problem was the fingers. The fishing lines inside the arms had multiple problems, sometimes they were stuck in between the servo that pulled them and other times the wires were just badly knotted or knotted at the wrong point. This job took a few hours to finish. The servos are shown in Figure 32.

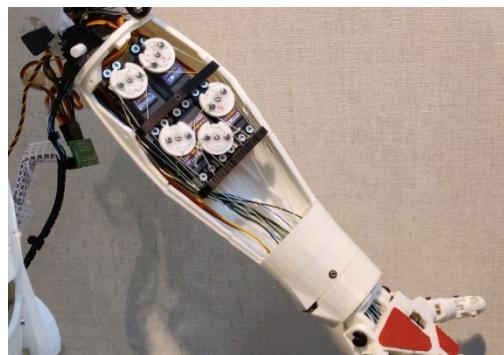


Figure 32. Finger servos

Reparation of the index finger

The entire index finger was replaced by a new 3D printed one because it was entirely broken. This was a difficult task since the wires inside the arm were almost inaccessible. The solution to this problem was to reconstruct the fishing line wires out of the arm so we could arrange the finger parts over these wires. If the wires are pulled out too far the whole hand and forearm would need to be reassembled. The second issue was the finger sensors. These wires cannot be intertwined with the fishing wire otherwise the fingers would not work anymore. Once these sensor wires are at the end of the fingers they needed to be soldered to a plate for better conductivity. In Figure 33 is the new index finger shown.

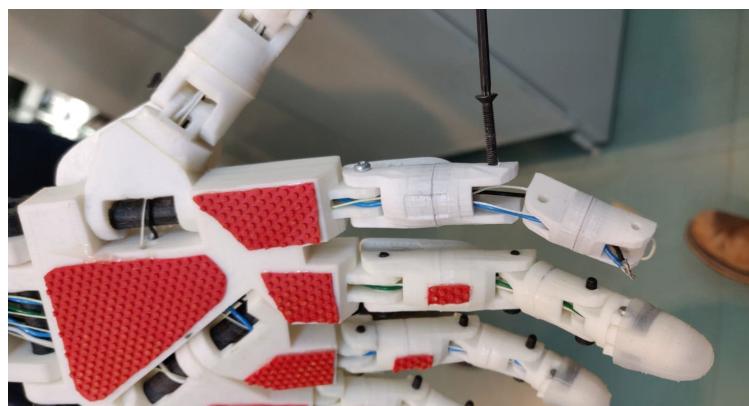


Figure 33. Replaced index finger

Reparation of the bicep

Then the only thing that still didn't work was the bicep, since it always overextended. The first attempt at solving it was to take out the bicep and servo looking for the problem. Concluding that the problem was the potentiometer in the elbow, both servo and potentiometer were taken out and replaced, but it did not work. As it turned out, the replacement servo had already been used but was taken out because it was broken. Therefore, a new out of the box servo was used as a replacement. This worked, and the bicep was repaired as shown in Figure 34.



Figure 34. Bicep servo

Reparation of the zero point in the shoulders

The bicep servo was not the only thing to be repaired; both shoulders in both movements (up-down, inside-out) had the wrong zero position. This position can be described as the point of the servo's potentiometer where if it turns one way, the movement goes one way, and if you turn the other way, the movement goes the opposite.

In order to find the correct zero position the sliders of myrobotlab were used. After some testing and adjusting the position of the potentiometers, this was repaired.

Reparation of the left arm

The whole left arm stopped working after a breakdown with the tablet. Once arm and shoulder were taken off (Figure 35), it was clear that the bolt broke inside the shoulder. A previously printed part was used to replace the bolt. This was not the only thing that needed to be fixed inside the arm. Therefore, it was the perfect opportunity to get the potentiometer at the correct position so that overextending wouldn't happen anymore. The bicep servo was also not reacting anymore. After a thorough analysis we confirmed that the cause of this was a lose wire. After this the reparations of the left arm were complete, and it could once again be reassembled.



Figure 35. Arm disassembled

Reparation of the jaw

Since the beginning of this project the jaw servo did noticeably moving, but the jaw itself only moved a tenth of the length it should move. To be able to see what the problem was, the head needed to be opened. Once this was done, the cause seemed clear. the servo was disconnected from the jaw. But after reattaching it the problem improved but remained the same. A further analysis proved that the head screws weren't holding on to the jaw.

After putting in new screws the reparation of the jaw was complete.

Covering of the cables

For the sake of aesthetics and looking for the safety of the cables before the movements of the robot, the cables were covered. For this multiple tube were analyzed. The first one showed on the right in Figure 36 had the correct size but since it was rigid, placing it in the robot was difficult and it did not allow the proper movement of the robot. The second one shown on the left in Figure 37 was more flexible but it also was too big. Finally, the third and the final one is not a tube but a curly plastic band which wraps the wires, see Figure 38. The great advantage of this method is that it is easy to remove and put it back again, which means that there is fast access to the cables.



Figure 36. Cable cover one



Figure 37. Cable cover two



Figure 38. Final cable cover

Attaching battery

Another goal of the robot was to make it independent from the plug, so a battery and a switch were applied. The first step was to analyze what was going to be powered by the battery, how many power was necessary. From this study and doing some research in InMoov forums and on the web, it was concluded that the battery should be 6V and 6A. And that the two Arduino's Mega and the speakers are feed following the electric scheme of Figure 39. The switch was chosen taking into account the current that would flow through the circuit and choosing a switch of bigger current.

Welding the receivers to the battery was not difficult, since in both cases they were receivers with a positive and a negative cable. The only difficulty was the speaker, since the plug had to be cut, and it was necessary to discover which one was the negative. This was done by plugging the cable and measuring the positive or negative voltage with a multimeter.

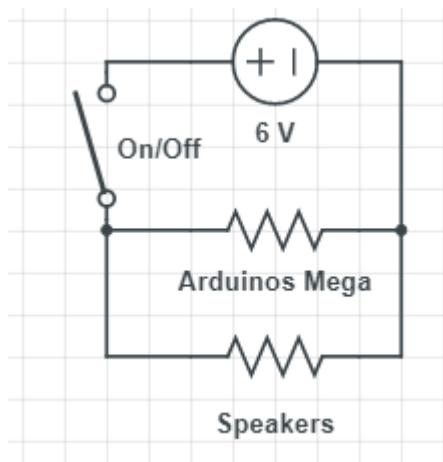


Figure 39. Electric scheme battery

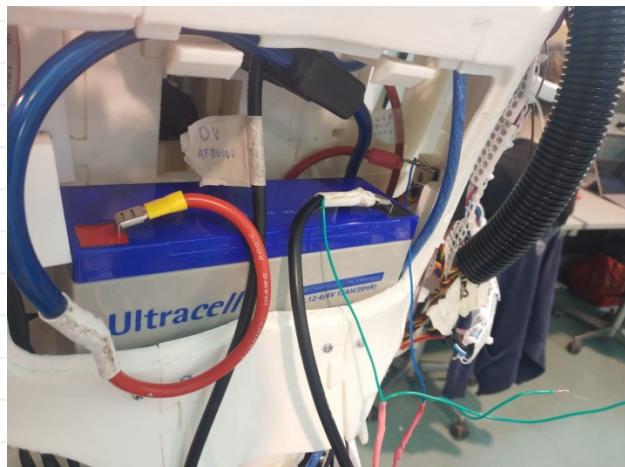


Figure 40. Battery

6. Improving software

6.1 Exploring the software

When the hardware was mostly repaired, the work on the software could start. At first this part was complicated and confusing. After deeper study and experimenting with it things became clearer. To understand the software the InMoov website and the InMoov forum were used. In addition, the GitHub website and the myrobotlab.org were very useful for the programming.

The main software used in the project is MyRobotLab, MRL in short. This environment is used to test the robot in a virtual form with the Monkey engine, further explanation later in the text, and in the real world. The Figure 41 shows the interface of the software.

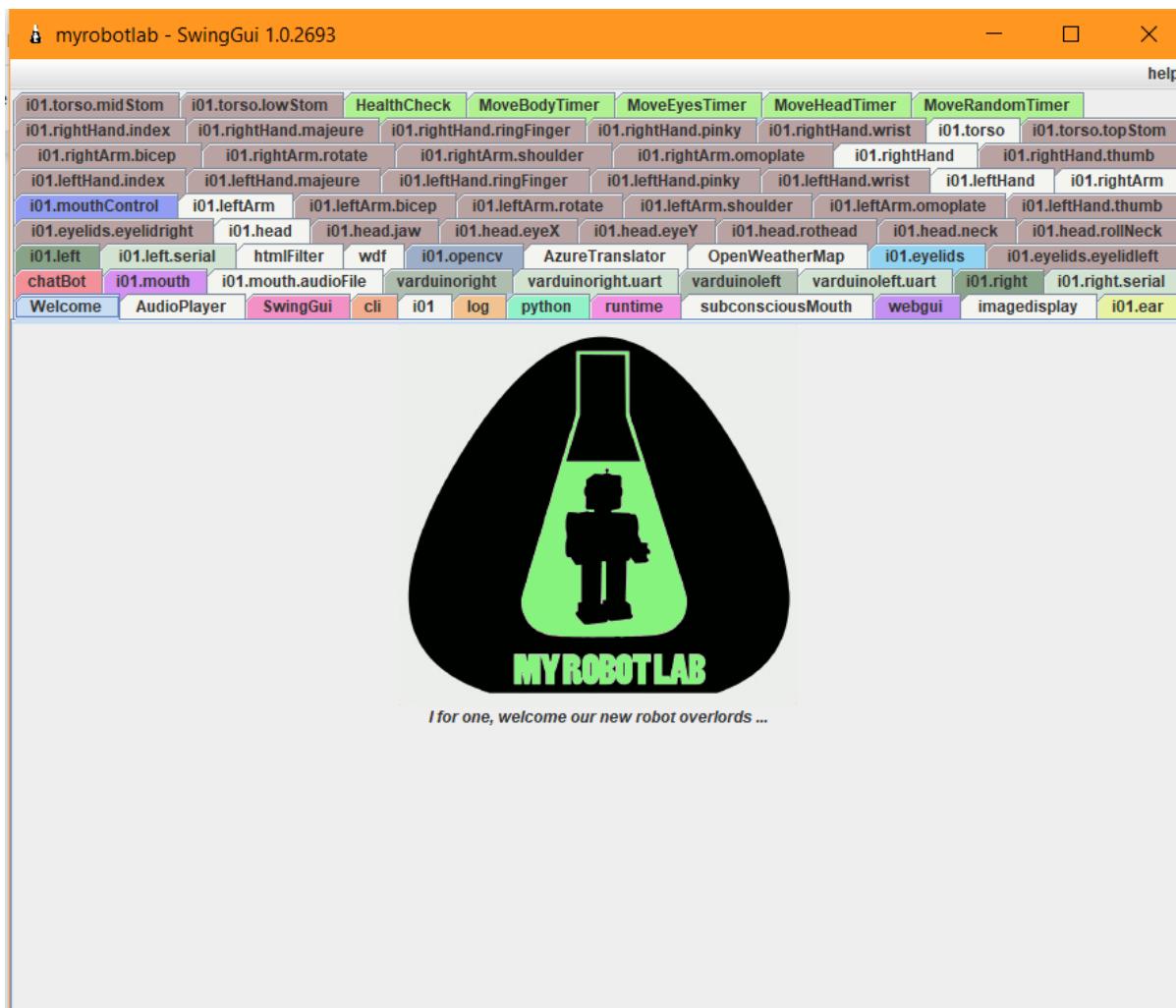


Figure 41. MyRobotLab interface

As the previous Figure shows, there are many tabs in this window, most of them contain sliders that are used to move the servos into specific positions. The description of the tabs is being shown in the Figure 42.

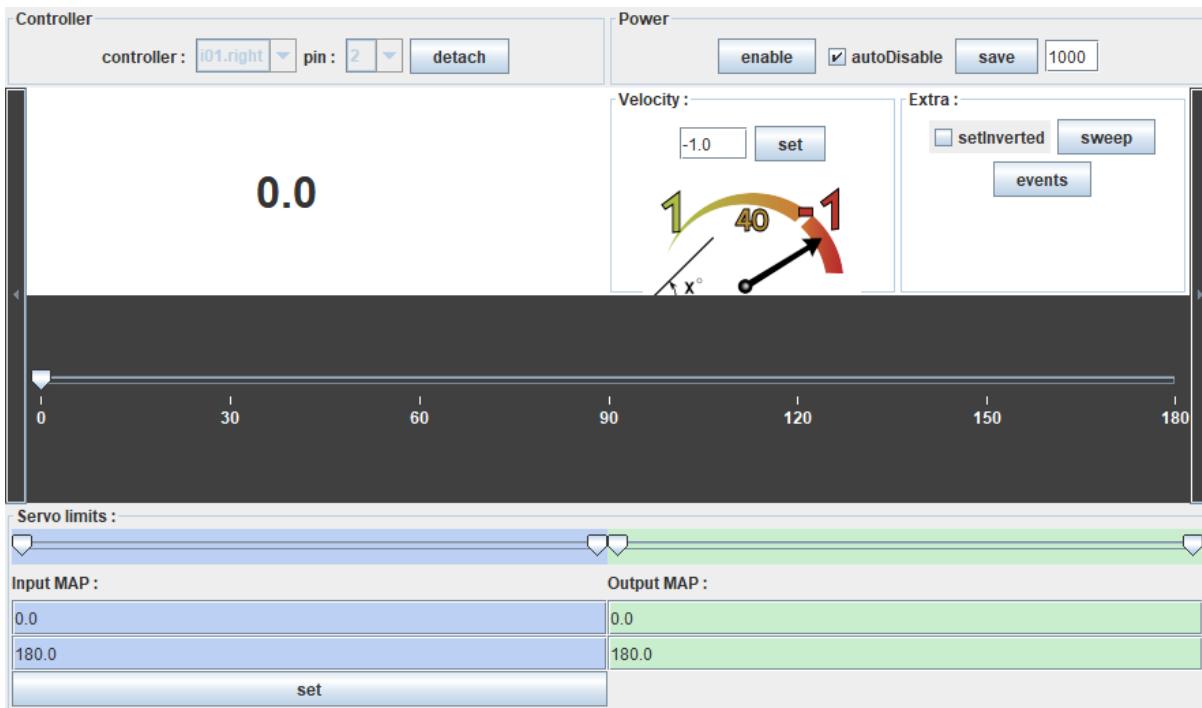


Figure 42. Tabs interface

The position of an individual servo can be adjusted here, as well as the speed at which it moves (standard on -1, full speed) and finally the limits of the servo. This is a testing environment, however, the values changed here will reset once the program shuts down. In order to get permanent changes, the values have to be modified in the configuration files (see next chapter).

Another important tab is the chatbot. This allows the communication with the robot by typing in commands. If the command is unrecognized, it will give a random response. Chatbot has the same function as voice command but it avoids the problems that can be given by the google voice command, since it does not always understand what is said or detects background voices.

In the configuration files there three options when running MRL. It can have a complete connection to the robot, only the right side (the right Arduino card) or a virtual start up. This last option shows a window like the one as in Figures 43.

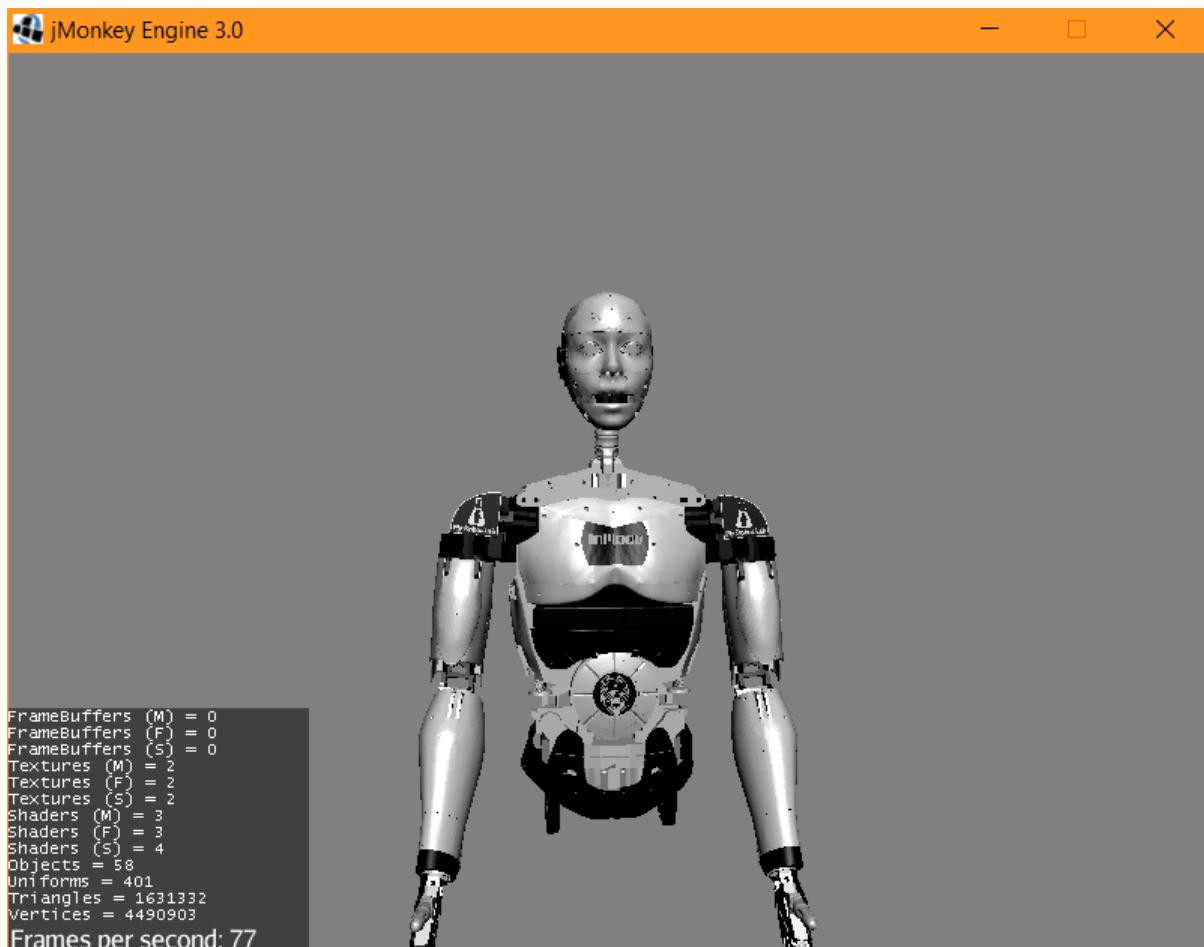


Figure 43. Monkey engine interface

The Monkey engine is useful to see how the robot will move when the servo positions are adjusted. It also allows experimenting in a way that will not harm the robot if it goes wrong.

6.2 Configuration files

The configuration files, config files as an abbreviation, allow to permanently change settings of the robot, for example the minimum and maximum position of a servo as well as the resting position. For the left arm it looks like this:

```
;----- LEFT ARM CONFIGURATION -----  
[MAIN]  
isLeftArmActivated=True  
  
[SERVO_MINIMUM_MAP_OUTPUT]  
,your servo minimal limits  
bicep=0  
rotate=40  
shoulder=0  
omoplate=10  
  
[SERVO_MAXIMUM_MAP_OUTPUT]  
,your servo maximal limits  
bicep=90  
rotate=180  
shoulder=180  
omoplate=80  
  
[SERVO_REST_POSITION]  
bicep=0  
rotate=90  
shoulder=30  
omoplate=10
```

If these values are changed and MRL is restarted, then those changes will be the new standard. This can be done in all the servos.

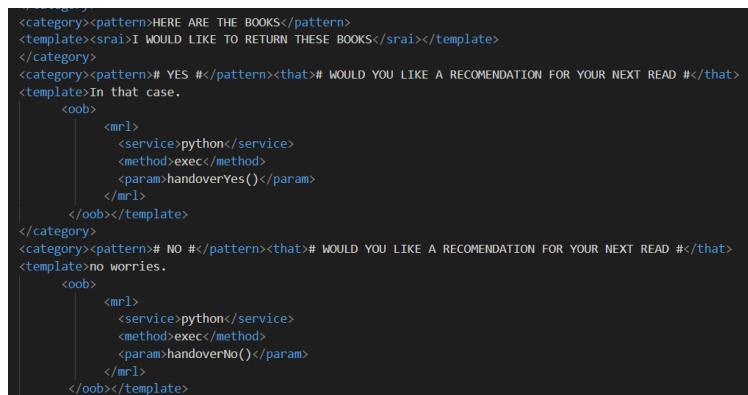
Other settings that can be changed are the spoken language of the robot, if it starts up in virtual mode or not, the communication ports (com ports) that are connected to the arduino, enabling or disabling voice commands, for instance. Any changes made and saved here are permanent until they are changed again.

6.3 Gestures

There are many gestures the robot can perform that are already installed. A very simple one looks like this:

```
def Yes():
    if isHeadActivated==1:
        i01.startedGesture()
        i01.setHeadVelocity(45,45,45)
        i01.moveHeadBlocking(130,90)
        i01.moveHeadBlocking(50,93)
        i01.moveHeadBlocking(120,88)
        i01.moveHeadBlocking(70,90)
        i01.moveHeadBlocking(90,90)
        i01.finishedGesture()
```

This small program will make the robot nod in agreement. New gestures can be added by using the capture motion function, which gives the exact position of every limb used to make that gesture; these positions are translated in code like above. Afterwards the code can be copied and pasted into a python file which then needs to be saved under a python file. When this step is complete the programming can start around the gesture itself, as in the picture below, to make it move, look and respond. The code can be more complex as well and can be several pages long. Figure 44 shows a part of the file that stores all the commands and outputs the voice response (if there is one) and calls upon the code to move the robot as seen above.^{xi}



```
<!-->
<category><pattern>HERE ARE THE BOOKS</pattern>
<template><srai>I WOULD LIKE TO RETURN THESE BOOKS</srai></template>
</category>
<category><pattern># YES #</pattern><that># WOULD YOU LIKE A RECOMENDATION FOR YOUR NEXT READ #</that>
<template>In that case.
<oob>
    <mrl>
        <service>python</service>
        <method>exec</method>
        <param>handoverYes()</param>
    </mrl>
</oob></template>
</category>
<category><pattern># NO #</pattern><that># WOULD YOU LIKE A RECOMENDATION FOR YOUR NEXT READ #</that>
<template>no worries.
<oob>
    <mrl>
        <service>python</service>
        <method>exec</method>
        <param>handoverNo()</param>
    </mrl>
</oob></template>
```

Figure 44. Gesture list

In these files the programmer makes its scenarios, conversations and monologues. These files can not differentiate from the conversation in order for them to work. For example, programming a conversation will need a specific structure to follow, this pathway. This can be made broader by the “” function which will provide a random answer. Redirecting can also be done by the “<srai>” function so that different answers will be redirected to the same conversation and make it seem more fluent. The “yes” or “no” construct is also a pathway that can be used to further the conversation. This can cause multiple problems. These functions need to be written inside the gestures (Figure 45) themselves

instead of the overall programming, therefore the most important thing to keep in perspective is that this may not interfere with the other programmed functions.

```

def handover():
    i01.startedGesture()
    i01.setHandSpeed("left", 0.60, 0.60, 0.60, 0.60, 0.60, 0.60)
    i01.setArmSpeed("left", 0.60, 0.60, 0.60, 0.60)
    i01.setArmSpeed("right", 0.60, 0.60, 0.60, 0.60)
    i01.setHeadSpeed(0.65, 0.65)
    i01.moveHead(90, 90)
    i01.moveArm("left", 26, 105, 30, 25)
    i01.moveHand("left", 2, 2, 2, 2, 2, 90)
    i01.mouth.speak("perfect, You can put them here in front of me")
    sleep(2)
    chatBot.getResponse("SAY " + "Would you like a recommendation for your next read")
    i01.finishedGesture()

def handoveryes():
    print "hand over yes"
    i01.mouth.speak("In that case I would recommend Robopocalypse. It is written by Daniel H. Wilson. If you would like to read it, it is located in isle 4 row R.")
    i01.startedGesture()
    i01.moveArm("left", 60, 60, 45, 30)
    i01.moveArm("right", 60, 60, 45, 30)
    i01.moveHand("left", 20, 20, 20, 20, 20, 90)
    i01.moveHand("right", 20, 20, 20, 20, 20, 90)
    sleep(2)
    i01.moveArm("left", 71, 94, 41, 31)
    i01.moveArm("right", 71, 94, 41, 31)
    i01.moveHand("left", 20, 20, 20, 20, 20, 90)
    i01.moveHand("right", 20, 20, 20, 20, 20, 90)

def handoverno():
    print "hand over no"
    i01.mouth.speak(" Alright then, Have a great day! ")
    relax()

```

Figure 45. Gestures code

These same functions can be used for different goals. The objective during this project was to get the librarian scenario and the guide scenario functional. After choosing which scenario would be best functional for a humanoid, the exact sentences needed to be written each with its form of responses. With each phrase the movements needed to be exact and play within the next answers of the person the robot is interacting with.

6.4 Tablet

Since one of the objectives of the project was that the robot could be moved freely, the device chosen for the software was a tablet. The one from the project is a NuVision tablet, 8" of screen, so it can fit in the back of the robot, the operative system that uses is Windows 10 and it has 2GB RAM.

In despite of having this tablet, it was not really useful as it had several problems. The first one is that it can connect to any of the Wi-Fi in Technobothnia, so a hotspot from a computer was used; this problem prevents the system from being truly independent.

In addition to this first problem, the battery doesn't last long enough, and the tablet cannot be charged if it is connected to the robot. To solve this, an USB-hub and a power bank could be used.

However, the bigger struggle was with the software, myrobotlab. At the beginning, the program did not send the commands to the robot, so the tablet was rebooted, and everything was deleted. After this a new problem appeared (Figure 46), the program did not start completely, some tabs did not appear. This second problem only happened in the full start-up, not in virtual.

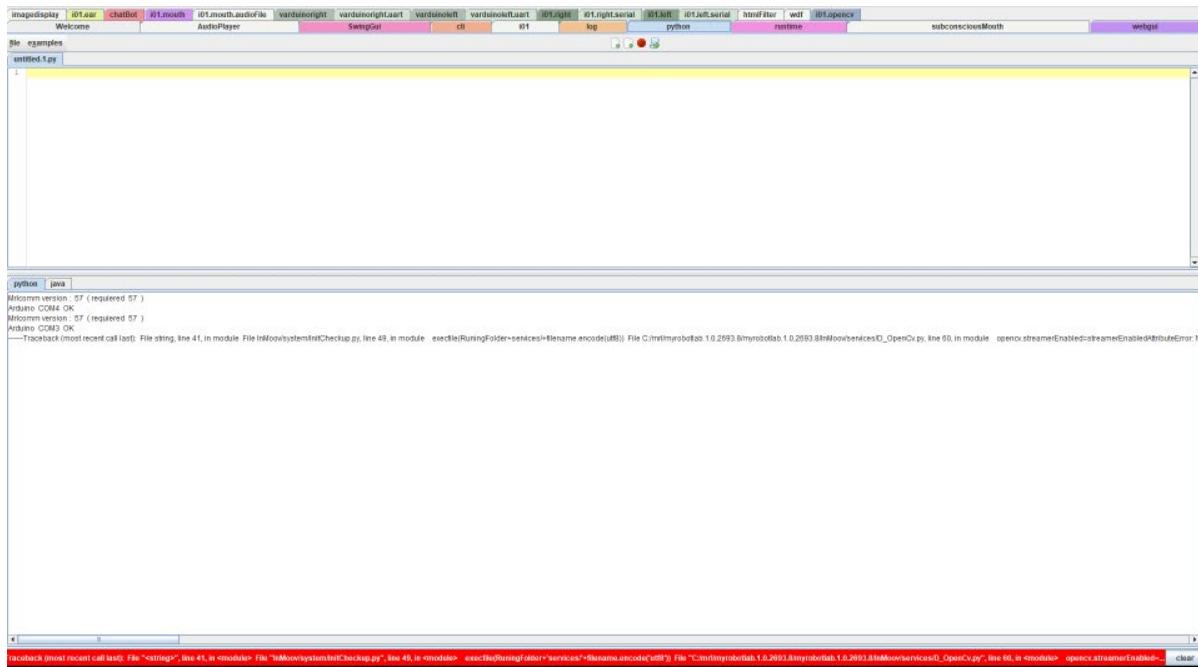


Figure 46. MRL error

To solve the problem, the InMoov google group was used and they recommended checking the version of Java, in this case it has to be 64 bit Java. Gael, the creator of the InMoov project, also sent a file with the software.

This solved the problem, but the tablet is still useless since it is needed at least 4GB RAM (8 GB RAM recommended). Summing up, a new tablet is necessary for this project.

6.4 Testing the Kinect

The Kinect maps the area in front of the sensor, hence it can construct 3D objects. In order to do that, the Kinect consists of three sensors, four microphones and a motor.

With the IR (infrared) transmitter a cloud of dots is layered over the scenery in front of the Kinect. Due to the calibration of the Kinect, it knows the distance of these object and with that information, it generates the depth image. In addition, there is an RGB-camera integrated in the Kinect, it takes a coloured picture and merges it with the depth image. This leads to a 3D scan. The device also has four microphones located around it. These can capture sounds and locate the source of the noise as well at the same time. It is a unique technique with which multiple players and sounds can be separated from each other. The last part is a small motor on the foot of the device which is the reason for its ability to tilt the camera up and down for about 30°.

The Kinect is a useful device for the InMoov project, since it can detect objects in front of it. Thus, it could turn on the eye tracking or even copy movements of people. Before testing the Kinect, it was necessary to understand its configuration, for that the InMoov website was really helpful. Once the configuration was done it was time to start testing it. In order to test the Kinect, the person to be detected must be at least two meters away from the robot, and the person has to be in a clear space, so the Kinect can scan the place more easily.

Finally, the Kinect was discarded from the project, because it did not have much use in the scenario that the robot wanted to be given (guide and librarian). In addition, making the device independent of the electrical network required extra material.

6.5 Presence Infrared sensor – PIR

The system has a presence infrared sensor, which detects movement and its output is digital (0/1). This sensor is mainly used to wake up the robot, as it has a sleep mode that disables all servos, or to launch the eye tracking.

Before implementing the sensor on the code, it was necessary to test it. This could be done by using the webgui oscope or a code in Arduino. If the oscope test is chosen it is recommended to use the newest version of myrobotlab (myrobotlab.1.1.155), it can be downloaded from the InMoov google group (build.myrobotlab.org:8080/job/myrobotlab-maven/155/artifact/target/myrobotlab.jar). Then it is important to upload the arduino communication files to the robot, after that the system is ready for the test.

However, for the sake of easiness, thet test can be done using the following code in Arduino (Figure 47)

```
PIR §
const int PIRPin = 23; //The Pir is connected to pin 23
void setup()
{
    Serial.begin(57600); //Using the serial monitor
    pinMode(PIRPin, INPUT);
}

void loop()
{
    int value= digitalRead(PIRPin);

    if (value == HIGH) //If the PIR detects someone
    {
        Serial.print("On "); //On will be printed continuosly
        delay(100);
    }
    else
    {
        Serial.print("Off "); //If the PIR doesn't detect anything, Off will be printed
    }
}
```

Figure 47. PIR test code in Arduino

If a more visual test is required, it can be used a LED and additional arduino, with the following assembly (Figure 48) and code (Figure 49)

```

PIR §

const int PIRPin = 5; // the Pir is connected to 5
const int LEDPin = 13; // the Led is connected to 13
void setup()
{
    Serial.begin(57600);

    pinMode(LEDPin, OUTPUT);
    pinMode(PIRPin, INPUT);
}

void loop()
{
    int value= digitalRead(PIRPin);

    if (value == HIGH)//If the Pir detects someone
    {
        Serial.print("On ");// On will be printed on the serial monitor
        digitalWrite(LEDPin, HIGH);//The Led will blink
        delay(100);
        digitalWrite(LEDPin, LOW);
        delay(100);
    }
    else //If the PIR doesn't detect anything
    {
        digitalWrite(LEDPin, LOW); //The LED will be off
        Serial.print("Off "); //Off will be printed
    }
}

```

Figure 48. PIR with LED test code in Arduino

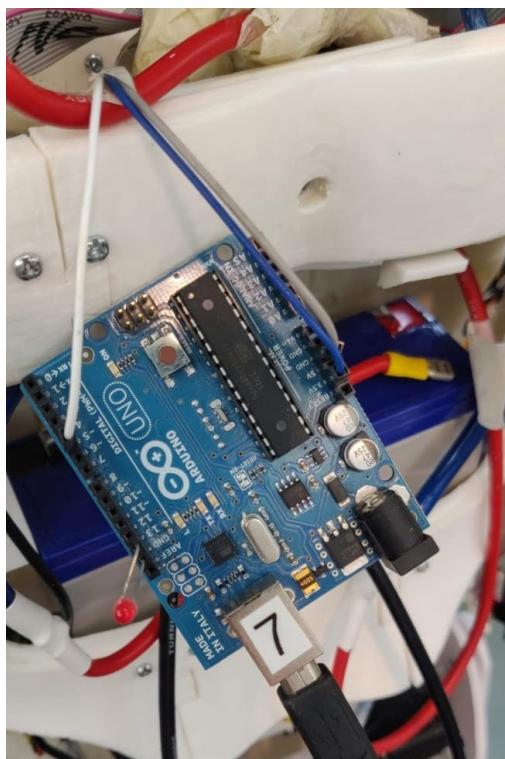


Figure 49. PIR with LED test assembly

Once the PIR is tested, if it works, it can be configurated to start the eye tracking or to wake up the system in the configuration files (C:\mrl\myrobotlab.1.0.2693.8\InMoov\config\InMoovLife).

6.6 Fingers' sensor

The hands of the humanoid are complemented by pressure sensors in each finger. This sensor is formed by two copper plates, which are powered by the Arduino, creating an open circuit. This circuit closes thanks to a conductive foam, so when the finger do pressure on the conductive foam the circuit is closed. However, the answer of the sensor is analogic, which means that if it is tested, the value obtained represents how much pressure the finger is doing. In order to test if these sensors work, the webgui oscope can be used as explained in the PIR text. But again, there is an easier test which consist on using an additional Arduino with the code of Figure 50 and the assembly of Figure 51.

Fingers §

```
int fsrPin = 0; // the FSR and 10K pulldown are connected to a0
    int fsrReading; // the analog reading from the FSR resistor divider
void setup(void) {
    // We'll send debugging information via the Serial monitor
    Serial.begin(57600);
}
void loop(void) {
    fsrReading = analogRead(fsrPin);
    Serial.print("Analog reading = ");
    Serial.print(fsrReading); // the raw analog reading
    // We'll have a few thresholds, qualitatively determined
    if (fsrReading < 50) {
        Serial.println(" - No pressure");
    } else if (fsrReading < 80) {
        Serial.println(" - Light touch");
    } else if (fsrReading < 100) {
        Serial.println(" - Light squeeze");
    } else if (fsrReading < 150) {
        Serial.println(" - Medium squeeze");
    } else {
        Serial.println(" - Big squeeze");
    }
    delay(1000);
}
```

Figure 50. Fingers' sensor test code in Arduino

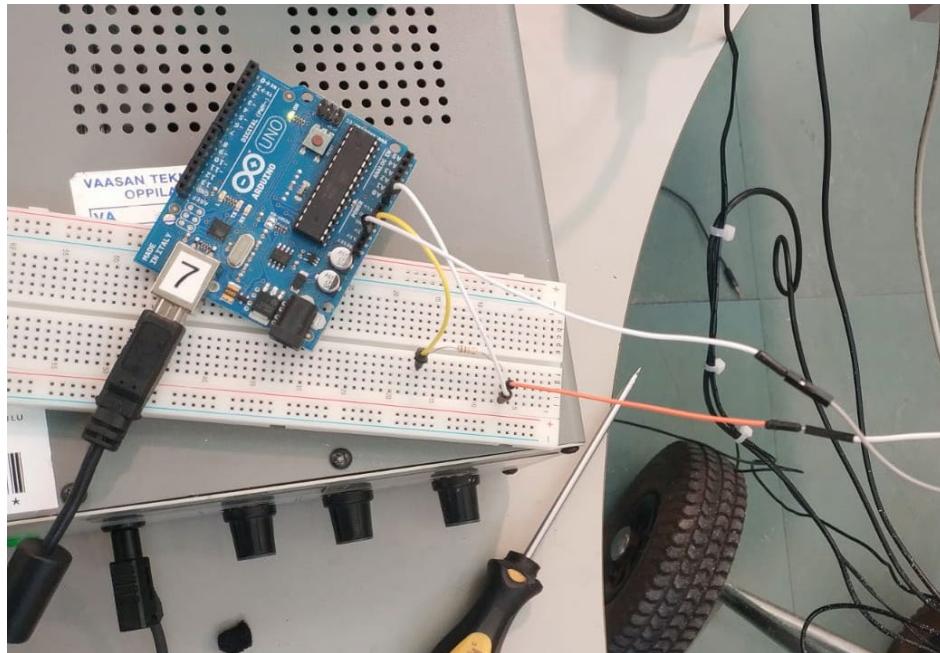


Figure 51. Fingers' sensor assembly

The cables orange and white that come from the right part of the photo are the ones from the plates. The white one goes to the power of the Arduino card, the orange cable goes to the resistance and to the analog pin 0; finally the circuit closes with the other extreme of the resistance on the Arduino card ground pin.

During the test in the serial monitor it will appear how much pressure is detected depending on the pressure is done on the finger.

These sensors are useful for grabbing objects of different shapes or thickness. They are also useful to control the strength of the hands.

6.7 Leap Motion

One of the objectives is controlling the robot from a distance by using human gestures. This is possible by combining hand tracking software and augmented reality glasses.

Leap motion is the best hand tracking hardware on the market right now, it is incredibly fast and accurate. The resulting image looks like Figure31. Combining the leap motion with VR glasses and integrating this with the robot's software will complete this objective.

Right now, there is a long way to go before this is possible. Writing new programs on the robot has so far been unsuccessful. The same goes for working on the VR station. In other words, to create this work abundant research needs to be done. Moreover, since this is complemented with the VR, it was decided to focus the work on the virtual reality, rather than on the leap motion.

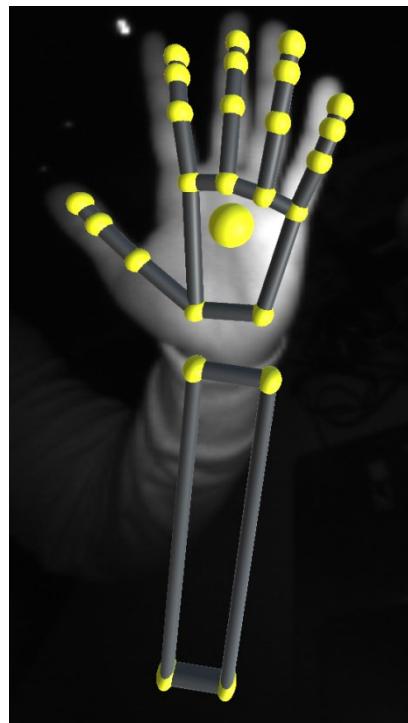


Figure 52. Leap motion

6.8 Testing the eye tracking

To initialize the eye tracking the command 'Face detection' needs to be given to the robot. Once this is done the robot will move its head to follow the person in front of it. The tracking can be done in different ways, frontalface_alt and eye are the ones we are interested in the most. Frontface_alt simply detects a person's face and once it has one the robot will be able to follow it, this is shown in Figure32. This method works the most consistent for different people. Eye will look for a person's eye and follow it. This method certainly works but tends to give problems for people with glasses.



Figure 53. FaceDetect

Another option is using the command 'track'. This command does not automatically start tracking the first face it sees, some points need to be selected on for the camera to follow. This is done with LKOpticalTrack. If these points move, the robot will follow the movement, this is shown in Figure33.

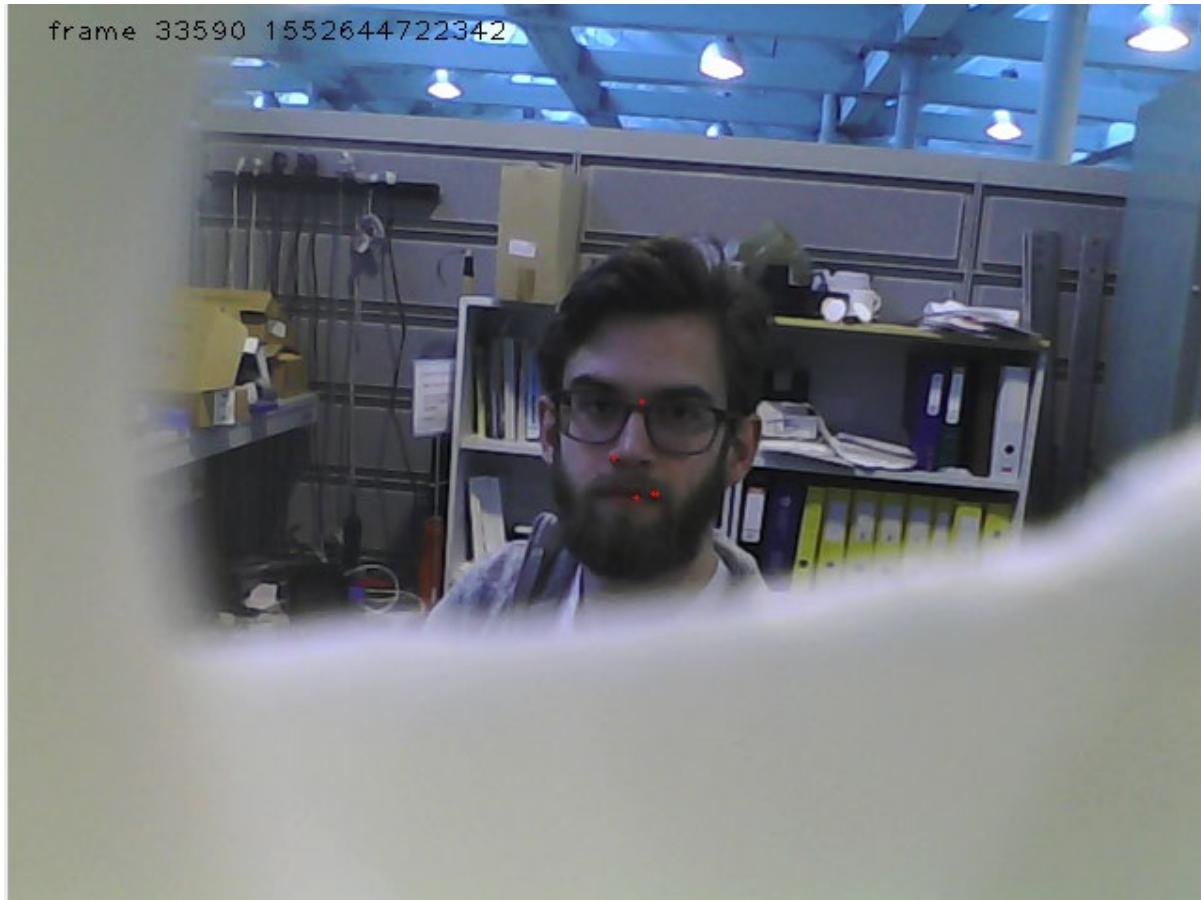


Figure 54. LKOpticalTrack

The point selection can also be done automatically but this can lead to some problems if the points are put on the background, since the background does not move the robot will not move either. Therefore, the automatic point selection will not be used, the ideal option is using Face Detect. This command was programmed with a PID-controller (proportional, Integrating, differential) where the P will send a direct signal if it detects a movement, the Integer will slow that signal down so the movement looks more fluent and the D value we turned off. We turned it off because it is used to predict the next P value and gives an early signal to compensate it. But in our system, this would only make the movements less fluent. here we programmed in a feedback principle so that the p value keeps changing with the movement around him.

7. Virtual Reality

The main goal of the Virtual Reality part of this project is to control the robot arms by using a VR controller. This will be done by using a combination of Arduino and Unreal Engine. Since this is a very ambitious goal it will be split in different sub goals, starting small and getting more complex over time.

7.1 Digital twin integration

In conjunction with repairing and improving the software for the robot we must design a role for our robot that we want it to do. For that, the first step is to develop two foundational functions on which the role can be based on. The first one is developing virtual reality digital twin connectivity and the second one is robotic remote control – tele-presence.

Work on the virtual reality digital twin integration is based on Unreal engine with HTC VIVE virtual reality glasses and Arduino connection, the software used is Unreal engine and Arduino program for Arduino board. The end goal is to fully integrate all the robot's function into the virtual reality digital twin. This includes movement, cameras, Kinect, PIR sensor, sound and speech recognition.

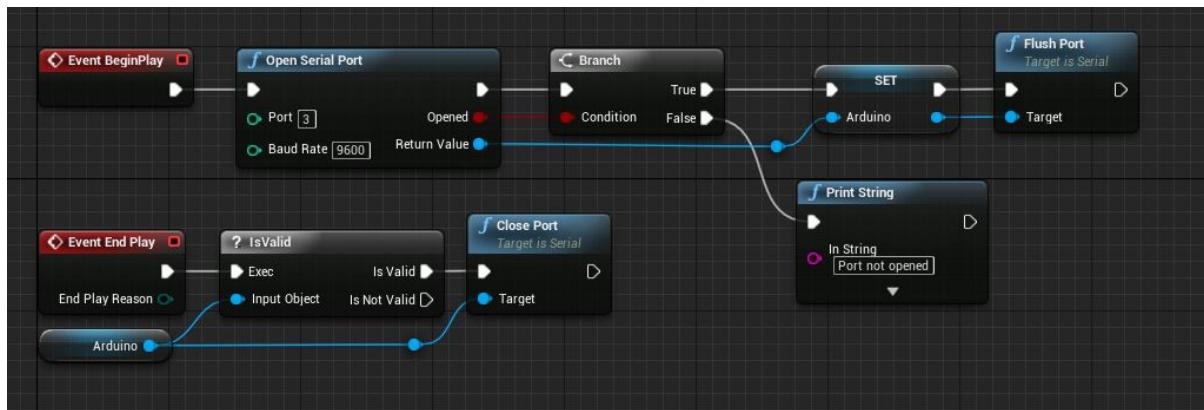


Figure 55. Unreal engine

7.2 Sending data from Arduino to Unreal Engine

The initial goal of the VR implementation was to make a virtual light change its intensity by turning a potentiometer. The potentiometer would give a signal to an Arduino Uno which would in turn send the value of the potentiometer to Unreal Engine. To properly communicate between these two the plug-in UE4duino is used.

The first difficulty was getting a connection between the Arduino and the Unreal Engine. After searching for a long time online only one tutorial was found. This tutorial however was a code that was made, and the instructions were to copy paste them into the software. Once this was done, the code was outdated and did not properly work. After some more research and programming, a solution was found that made the code work. In Arduino it looks like Figure 56.

```
const int analogInPin = A0; // Analog input pin that the potentiometer is attached to
const int analogOutPinX = 13; // Analog output pin that the LED is attached to
const int digitalOutPinY = 7; // Analog output pin that the LED is attached to

int sensorValue = 0;      // value read from the pot
int outputValue = 0;      // value output to the PWM (analog out)

void setup() {
    Serial.begin(9600);
    while (!Serial); // Wait until Serial is ready
    Serial.println("Enter LED Number 0 to 7 or 'x' to clear");
    // initialize serial communications at 9600 bps:
    Serial.begin(9600);
}

void loop() {

    // read the analog in value:
    sensorValue = analogRead(analogInPin);
    // map it to the range of the analog out:
    outputValue = map(sensorValue, 0, 1023, 0, 255);

    // print the results to the serial monitor:
    Serial.println(outputValue);
    delay(200);
}
```

Figure 56. Arduino sending code

This code reads the input from the potentiometer and sends it to the Unreal Engine. The code in Unreal Engine looks like in Figure 57.

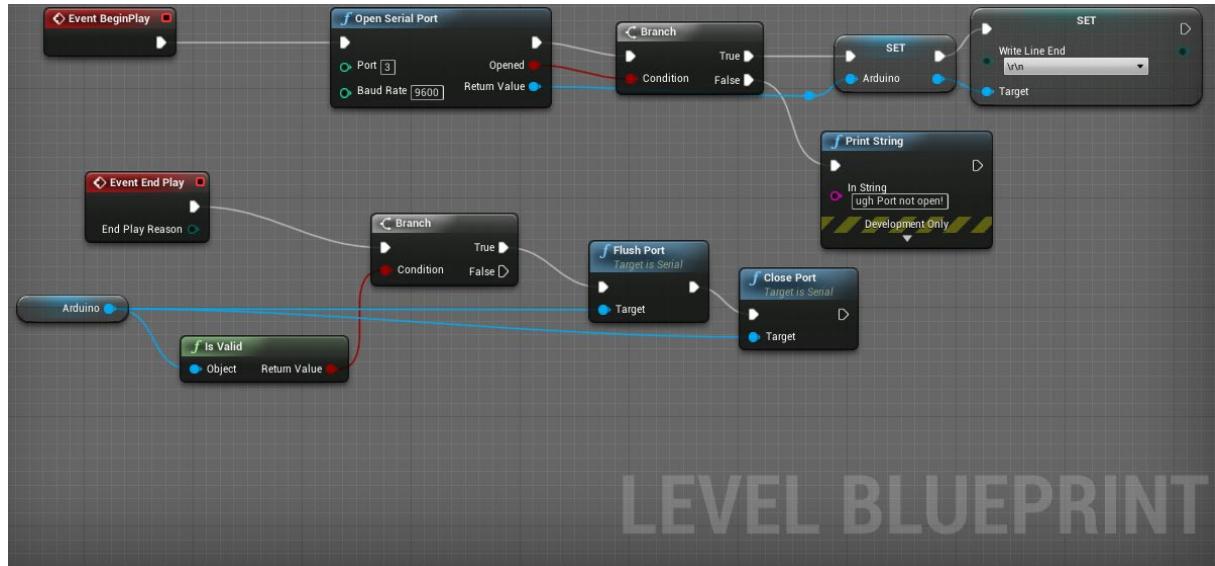


Figure 57. Connection to Arduino

Programming in Unreal Engine is more different compared to Arduino. It is a very visual way of programming. It works by connecting different building blocks that come together and make a working 'code'.

In this case, the starting point is the red block called 'Event Begin Play', this sends a single signal once the program starts. It will open the serial port which is connected to the Arduino, note that the Port and Baud Rate in this block have to be the same as the ones used in the Arduino code. The program will then check if the communication works, if it does not, a message will show up on the screen. If it does then it will set a serial variable that will be used to connect to the Arduino from here on out.

The last part of the code that is executed begins with 'Event End Play', this sends a single signal once the program is closed. This will check if there still is a connection to the Arduino and if there is it will close the connection.

The most difficult part was making the value of the potentiometer change the intensity of the light on the screen. Since the original tutorial for it did not work this took a long time to figure out. The Unreal code is shown in Figure 58.

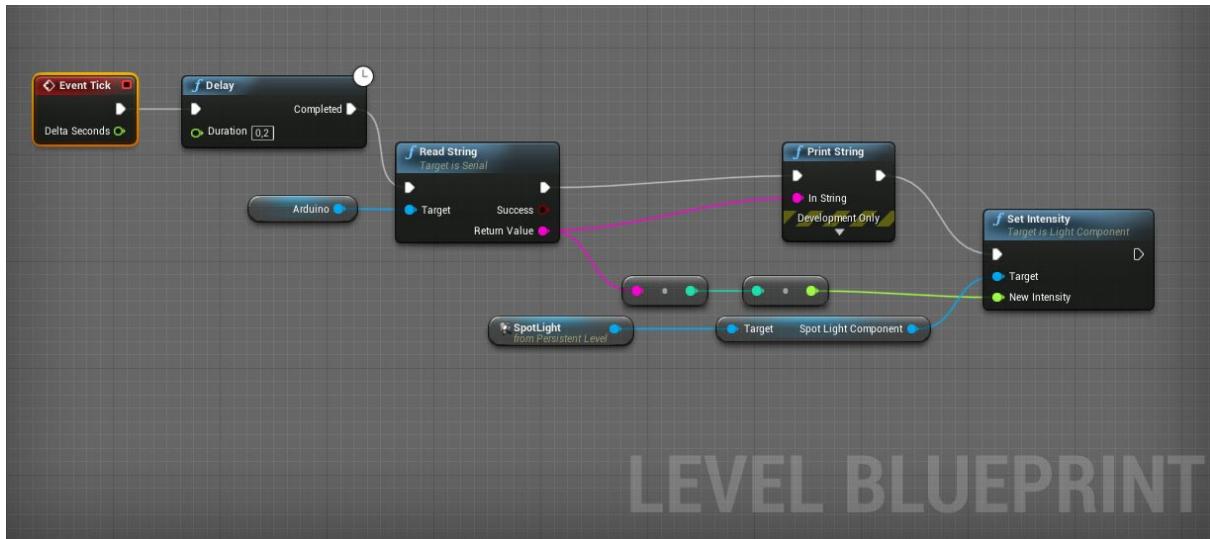


Figure 58. Changing the light intensity

In this code the 'event tick' is used. This will send a signal on every frame of the testing. This usually happens at 60 frames per second so signals will be sent at a frequency of 60 Hz. A delay is added of 0.2 seconds, this is the same delay that is used in the Arduino code and is needed to synchronize both. Arduino works a lot faster than 60 Hz so this delay is necessary. The value that is sent from the Arduino is read as a string which is then converted first to an integer and then to a float value. The string is also printed on the screen as a way to check if the value is correct. The resulting float value is then used to adjust the intensity of the light. The testing setup of the Arduino with the potentiometer is shown in Figure 59.

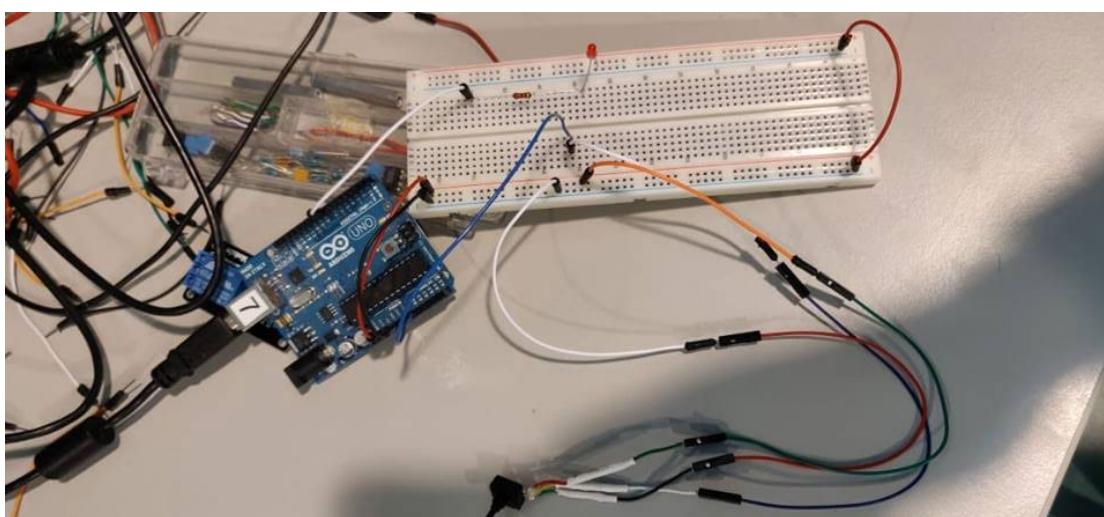


Figure 59. Potentiometer testing

7.3 Controlling a LED with Unreal Engine

Once it is possible to sent data from Arduino to Unreal Engine, the process could be reversed. The basic principle took some time as it required a Arduino code and an Unreal code. The Arduino code is showed in Figure 60 and the Unreal code in Figure 61.

```
const int digitaloutpin = 7; // digital output pin that the led is at
int LightIntensity = 0; // value read from the unreal engine

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    while(!Serial); //wait until Serial is ready
    Serial.println("Not ready");
    //initialize serial communication at 9600 bps
    Serial.begin(9600);
    pinMode(digitaloutpin, OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    if (Serial.available()){
        LightIntensity = Serial.read();
        analogWrite(digitaloutpin, LightIntensity);
    }
    delay(100);
}
```

Figure 60. Arduino LED controlling

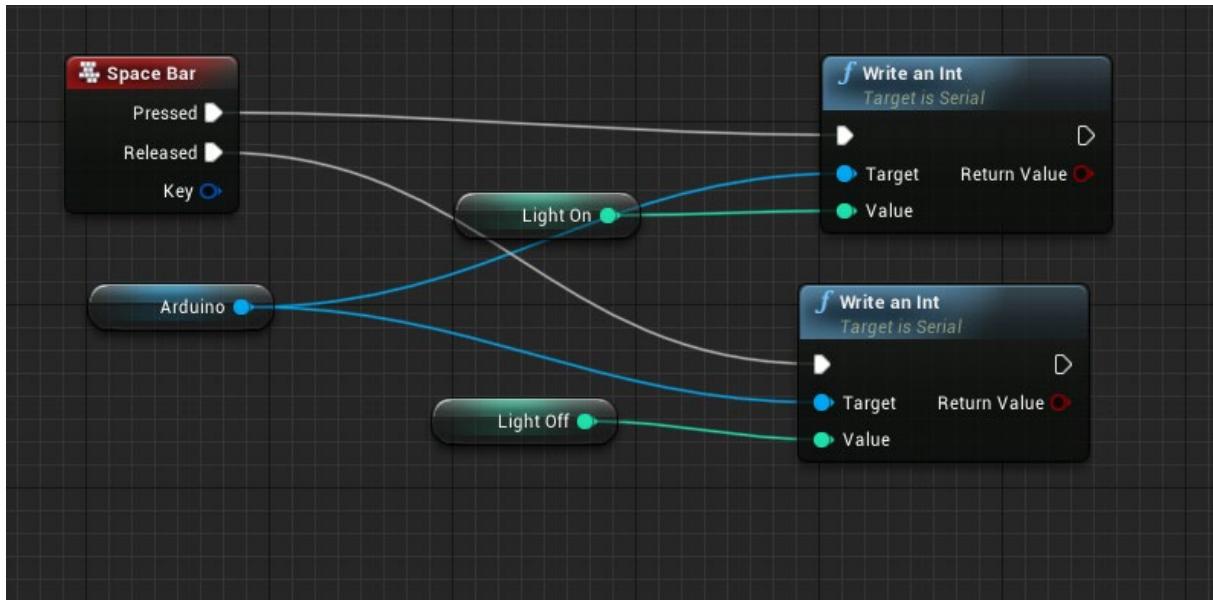


Figure 61. Unreal code using spacebar to light a LED

This previously showed code will send a signal if the space bar is pressed or when it is released. When it is pressed, a value of 255 is sent to the Arduino, this will set the LED to its maximum intensity. When it is released, a value of 0 is sent to turn the LED off. The physical testing setup is showed in Figure 62.

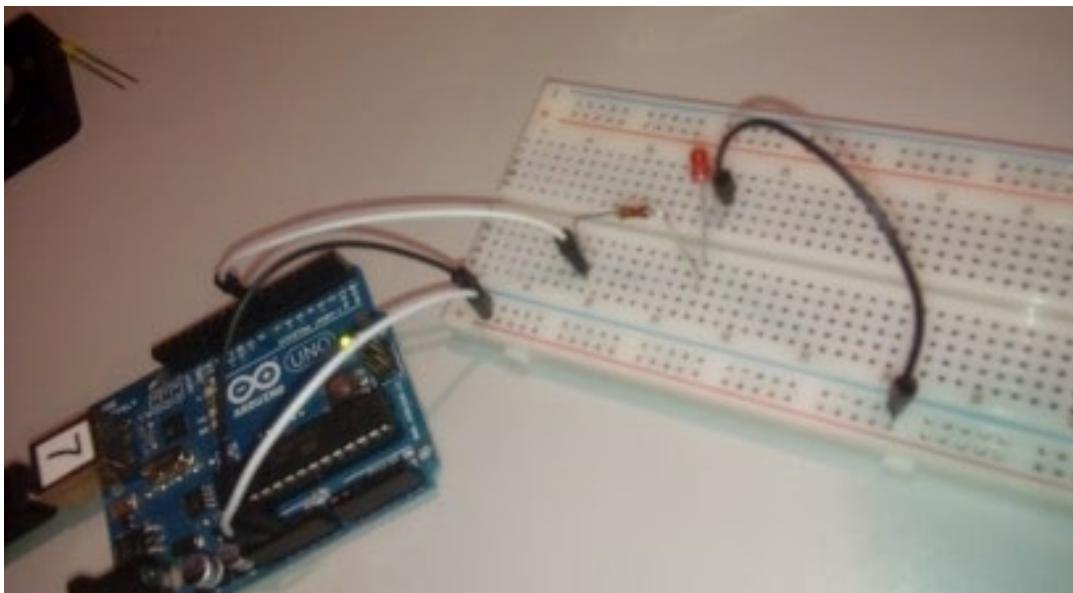


Figure 62. Controlling a LED with the spacebar

7.4 Controlling multiple LEDs trough Unreal Engine

The previous code worked the way it should when using only one LED but started giving problems when trying to add another one. The main problem being that only one variable can be sent per cycle of the program. This is not useful when trying to control multiple outputs at once. After some testing it was discovered that the signal from Unreal Engine is sent in the form of four bytes. Using the command `Serial.read()` will only read one line of input on the Arduino, in this case only the first byte. This limits the options since a byte only gives a value from 0 to 255 whereas an integer uses a value from -32 768 to 32 768. To solve this problem an array was used. Arrays are a collection of variables of the same type, in this case integers. The code will wait until four bytes are received from Unreal Engine. These four bytes are then put into an array and are multiplied by a factor of 256. By adding the results, the original integer is now stored in Arduino. This integer value can be used to determine both the light strength and which LED will be used. Each LED will correspond with an integer value. This LED value will be multiplied by 1000 in Unreal Engine and will then be added to the light intensity. This can safely be done since the maximum light intensity is 255. As an example, the integer 3150 represents LED number four (arrays start at zero so the 3000 will be used for the 4th LED) with a light intensity of 150, see Figure 63.

```

const int GreenPin = 9;
const int RedPin = 10;

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    while (! Serial);      // Wait untilSerial is ready
    Serial.println("Enter LED Number 0 to 7 or 'x' to clear");
    // initialize serial communications at 9600 bps:
    Serial.begin(9600);
}

void loop() {
    // put your main code here, to run repeatedly:
    int lightactive[4];
    while (Serial.available()<4) {}   // Wait until there are 4 Bytes waiting
    for(int n=0; n<4; n++){        // Add the 4 Bytes to the array
        lightactive[n] = Serial.read();
    }
    int UnrealInt = lightactive[0] + lightactive[1]*256 + lightactive[2]*256*256 + lightactive[3]*256*256*256;      //
    converts the value of the Bytes to an integer
    int SelectLight = floor(UnrealInt / 1000.0);      //Removes the last 3 digits
    int Lights[2] = {GreenPin,RedPin};      //Makes an array of all available outputs
    int LightIntensity = UnrealInt - (SelectLight * 1000); //Returns the last 3 digits
    analogWrite(Lights[SelectLight], LightIntensity);    //Sets the selected light at the desired intensity
    delay(200);
}

```

Figure 63. Arduino controlling two LEDs

The Unreal Engine code used to control two different LED lights is showed in Figure 64.

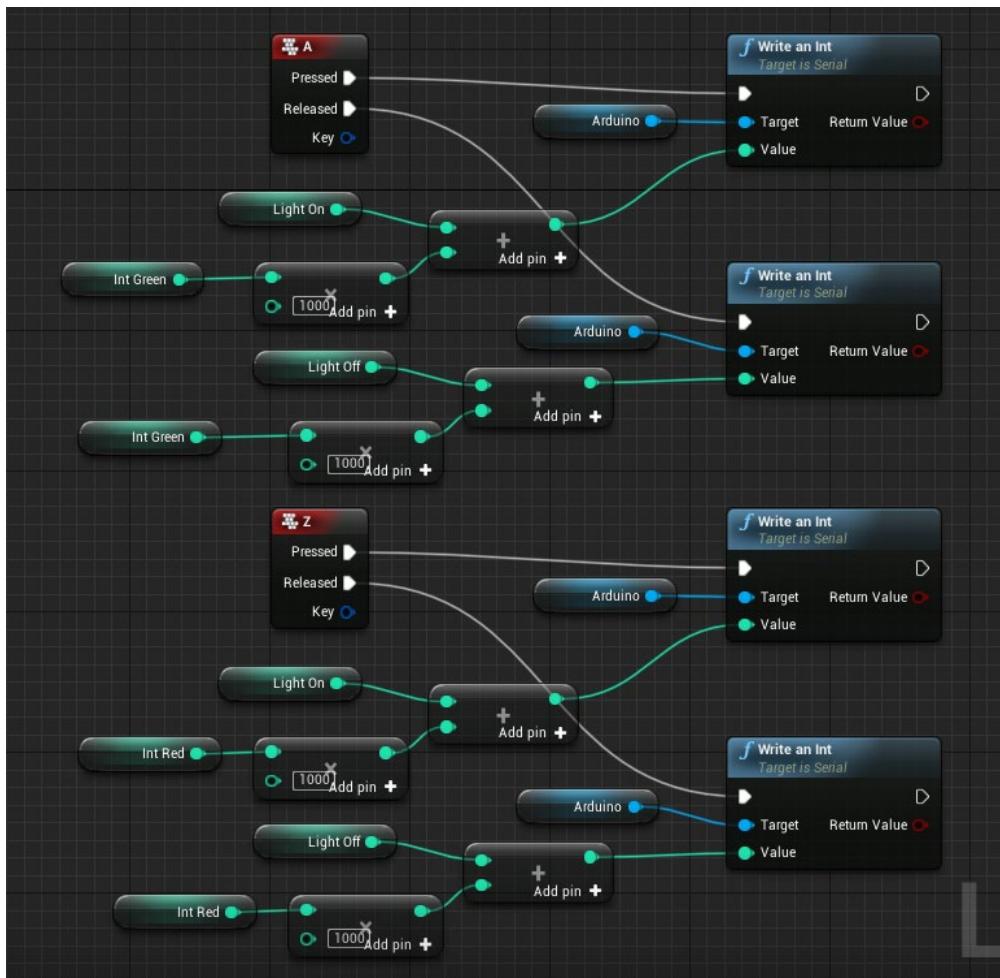


Figure 64. Controlling multiple LEDs with Unreal Engine

The code will send an integer every time the 'a' or 'z' buttons are pressed or released. This will then update the different LED lights. In Figure 65 is the physical setup showed.

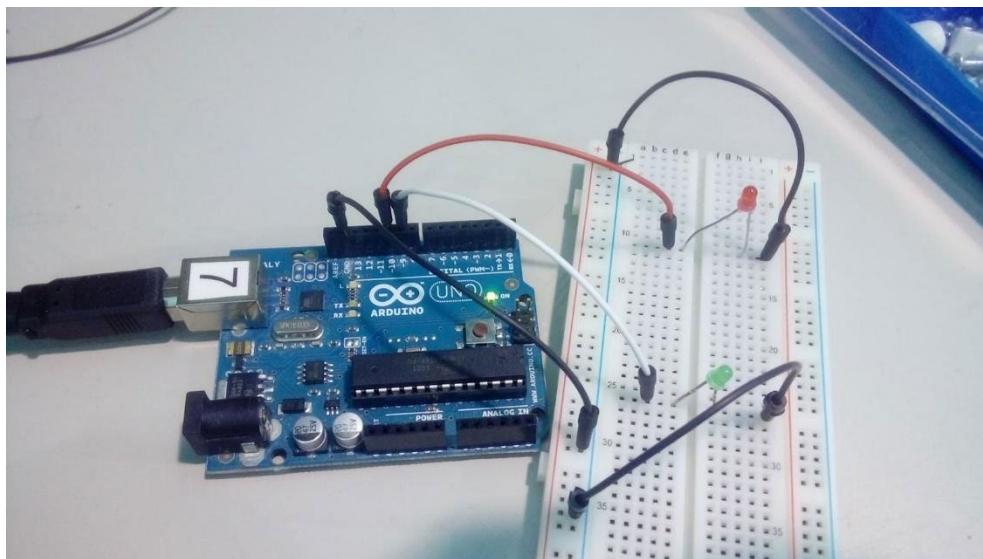


Figure 65. Arduino connected to multiple LEDs

7.5 Using a constantly changing variable to control a LED

The previous codes worked fine with a fixed variable however a constantly changing variable is much more interesting and will give a significant amount of more options later. The changing variable will be taken from a computer mouse based on its location. One variable will be sent for the x-direction and another one for the y-direction. These variables will be modified the same way as in the previous code so they can change the intensity of multiple LEDs. The Arduino and Unreal code to achieve this is showed in Figure 66 and 67.

```

const int GreenPin = 9; //digital output pin that the led is attached to
const int RedPin = 10; //digital output pin that the led is attached to

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    while (! Serial); // Wait untilSerial is ready - Leonardo
    Serial.println("Enter LED Number 0 to 7 or 'x' to clear");
    // initialize serial communications at 9600 bps:
    Serial.begin(9600);
    pinMode(GreenPin, OUTPUT);
    pinMode(RedPin, OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    if (Serial.available()){
        int lightactive[4];
        while (Serial.available()<4) {} // Wait untill there are 4 Bytes waiting
        for(int n=0; n<4; n++){ // Add the 4 Bytes to the array
            lightactive[n] = Serial.read();
        }
        int UnrealInt = lightactive[0] + lightactive[1]*256 + lightactive[2]*256*256 + lightactive[3]*256*256*256; // converts the value of the Bytes to an integer
        int LightIntensity = floor(UnrealInt / 100.0); //Removes the last 2 digits
        int SelectLight = UnrealInt - (LightIntensity * 100); //Returns the last 2 digits
        int Lights[2] = {GreenPin,RedPin}; //Makes an array of all available outputs
        analogWrite(Lights[SelectLight], LightIntensity); //Sets the selected light at the desired intensity
    }
    delay(4);
}

```

Figure 66. Arduino variable changing

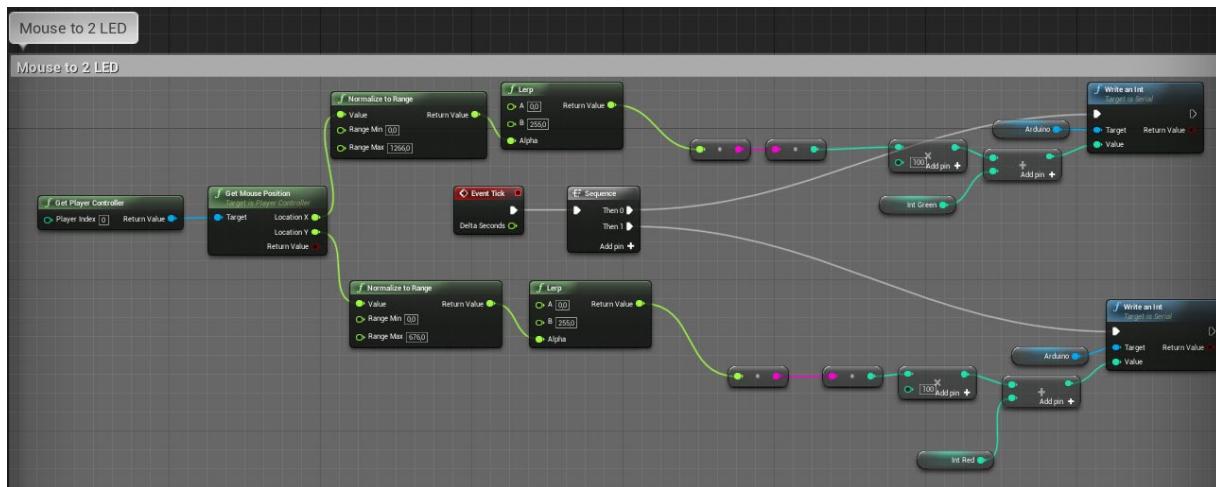


Figure 67. Mouse controlling two LEDs

7.6 Working with servos instead of LEDs

Using LEDs is a great way to test these programs but controlling servos is the eventual goal. The reason LEDs were used is that they are cheaper to replace in case of a big failure. To change from LED to servo only the Arduino code needed to be changed. An example of using servos in the code looks like this is showed in Figure 68. This code is very similar to the one used for controlling one LED. Changing the previous programs to use servos instead is a simple matter.

```
#include <Servo.h>
Servo servoindex; // Define index servo

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    while (! Serial); // Wait untilSerial is ready
    Serial.println("Enter LED Number 0 to 7 or 'x' to clear");
    // initialize serial communications at 9600 bps:
    Serial.begin(9600);
    servoindex.attach(3); // Set index servo to digital pin 3
}

void loop() {
    // put your main code here, to run repeatedly:
    if (Serial.available()){
        int UnrealByte[4];
        while (Serial.available()<4) {} // Wait until there are 4 Bytes waiting
        for(int n=0; n<4; n++){ // Add the 4 Bytes to the array
            UnrealByte[n] = Serial.read();
        }
        int UnrealInt = UnrealByte[0] + UnrealByte[1]*256 + UnrealByte[2]*256*256 +
        UnrealByte[3]*256*256*256; // converts the value of the Bytes to an integer
        int FingerPosition = floor(UnrealInt / 100.0); //removes the last 2 digits
        servoindex.write(FingerPosition);
    }
}
```

Figure 68. Arduino controlling servos

7.7 Using VR controllers

Getting data from a VR controller works in the same way as getting the data from a computer mouse. However, when the user is holding a controller it is possible that the computer thinks the arm is in a different position. For this reason, a virtual body needs to be designed and connected to the VR controllers and glasses. The reason the glasses need to be added is because the position of the controllers is relative to the glasses.

First, a character needs to be made with a skeleton that can be connected to the controllers. Fortunately, Unreal Engine has a standard character that can be used for this. Next, a setup like shown in figure 69 needs to be made in the character blueprint. This will connect the camera and the controllers to Unreal Engine.

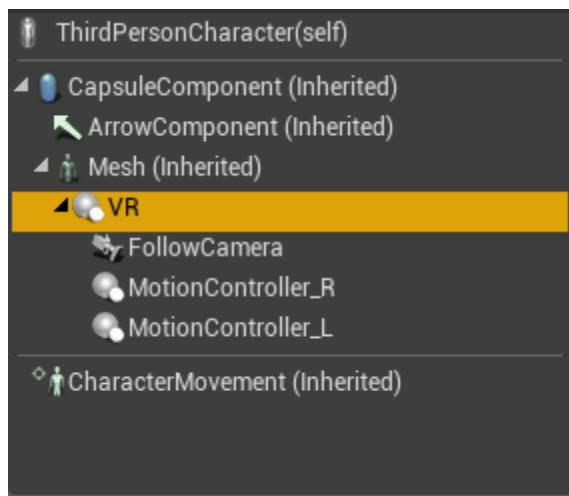


Figure 69. Controller setup

When this was done, the controllers need to be able to control the arm movement. This is done in the animation blueprint and will set up the inverse kinematics of the character arms and hands. Now the character arms will move the same way as the person holding the controllers.

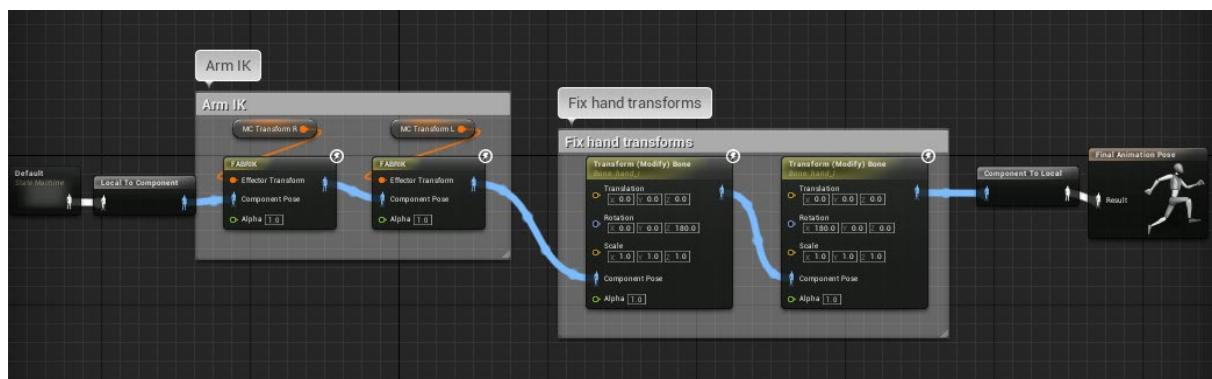


Figure 70. Connecting the arms to the controllers

7.8 Inverse kinematics

After connecting the VR controller to a virtual robot, the next step is to figure out how to translate a set of coordinates to a position for a physical robot arm to move to. This is done by using inverse kinematics. There are different ways to implement this but the one that

seemed the easiest uses goniometric calculations. The decision was made to work only in one plane (using only the x and z coordinates) since three dimensions are significantly more complex to program. The Arduino code is shown in Figure 71.

However, after extensive testing and trying to change this code it never worked. This idea was scrapped and replaced with another one, using Unreal Engine to directly acquire the rotation of the arm joints and using this to control the servos. This has the advantage that the math is done by the program that is directly connected to the VR controllers and it will easily work in three dimensions.

```
#include <math.h>
const int Yellow = 9;
const int Red = 10;
int x = 1;
int z = 1;
int LengthBiceps = 30;
float LengthForeArm = 37.5;

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
}

void loop() {
    //put your main code here, to run repeatedly:
    if (Serial.available()){
        int UnrealBytes[4];
        while (Serial.available()<4) {} // Wait untill there are 4 Bytes waiting
        for(int n=0; n<4; n++){ // Add the 4 Bytes to the array
            UnrealBytes[n] = Serial.read();
        }
        int UnrealInt = UnrealBytes[0] + UnrealBytes[1]*256 + UnrealBytes[2]*256*256 + UnrealBytes[3]*256*256*256; //
        converts the value of the Bytes to an integer
        int Position = floor(UnrealInt / 10.0);
        int Direction = UnrealInt - (Position * 10);
        if (Direction == 1) {
            x = Position;
        }
        else if (Direction == 2){
            z = Position;
            z = map(z, 0, 99, -65, 65);
        };
        double Elbow = (x*x+z*z-LengthBiceps*LengthBiceps-
LengthForeArm*LengthForeArm)/(2*LengthBiceps*LengthForeArm); // Inverse kinematics formula
        int ElbowCorner = acos(Elbow) * 57.2957795; //Convert radians to degrees
        analogWrite(Yellow, int(ElbowCorner));
        double ShoulderCorner = atan(z/x)-
atan((LengthForeArm*sin(ElbowCorner))/(LengthBiceps+LengthForeArm*cos(ElbowCorner)));
        // Inverse kinematics formula
        ShoulderCorner *= 57.2957795; //Convert radians to degrees
        analogWrite(Red, int(ShoulderCorner));
    }
    delay(4);
}
```

Figure 71. Arduino inverse kinematics

7.9 Inverse kinematics with Unreal Engine

Unreal Engine has a function block that returns the angles at which a bone from a skeleton is connected to its parent bone. The bones that are needed to get the angles from the elbow and shoulder joints are the lower arm and the upper arm. The Elbow is fairly easy since the physical robot has only one possible movement in the elbow, up and down. This means only one resulting angle will be used and the value of it sent to the Arduino. The shoulder takes a bit more work since all three angles need to be given a specific index number and they cannot all be sent at once. What this looks like in Unreal Engine is showed in Figure 72.

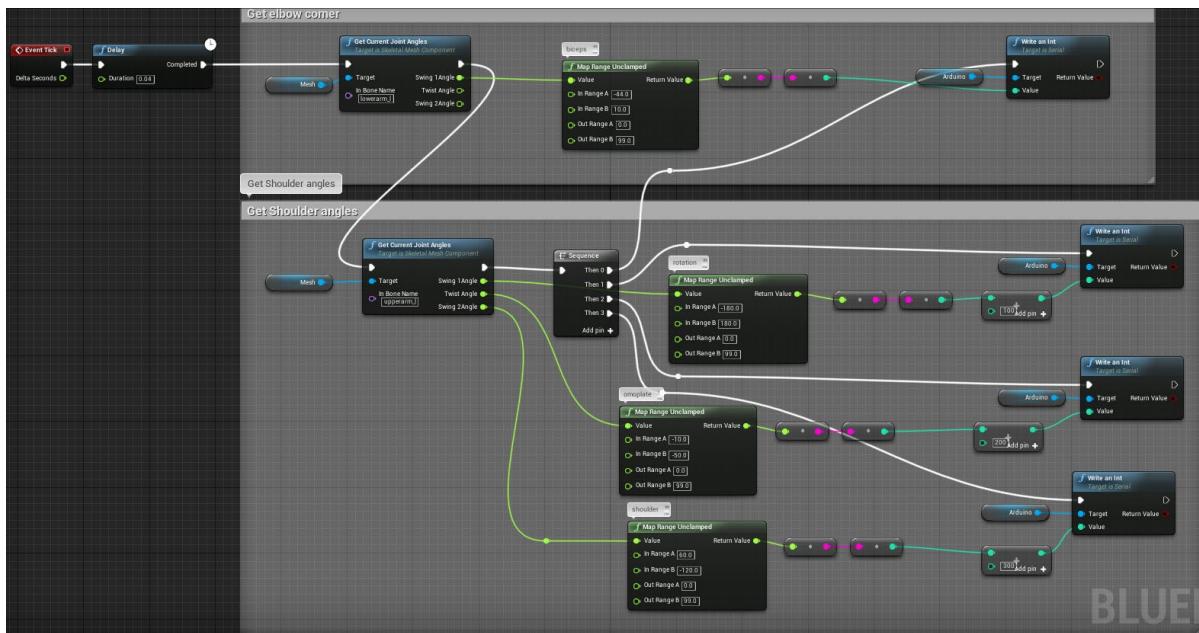


Figure 72. Inverse kinematics

Since converting a negative integer to a byte and back can occasionally give problems, all values are mapped to a positive range before being sent to the Arduino, see Figure 73.

```

int Elbow, Shoulder, Rotation, Omoplate, Elbow2, Shoulder2, Rotation2, Omoplate2;
#include <Servo.h>
Servo ElbowServo;
Servo ShoulderServo;
Servo RotationServo;
Servo OmoplateServo;

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    ElbowServo.attach(6);
    RotationServo.attach(9);
    ShoulderServo.attach(10);
    OmoplateServo.attach(11);
}

void loop() {
    // put your main code here, to run repeatedly:
    if (Serial.available()){
        int UnrealBytes[4];
        while (Serial.available()<4) {} // Wait untill there are 4 Bytes waiting
        for(int n=0; n<4; n++){    // Add the 4 Bytes to the array
            UnrealBytes[n] = Serial.read();
        }
        int UnrealInt = UnrealBytes[0] + UnrealBytes[1]*256 + UnrealBytes[2]*256*256 + UnrealBytes[3]*256*256*256; // converts the value of the Bytes to an integer
        int Joint = floor(UnrealInt / 100);
        if (Joint == 0) {
            Elbow = UnrealInt - Joint * 100;
        }
        else if (Joint == 1){
            Rotation = UnrealInt - Joint * 100;
        }
        else if (Joint ==2) {
            Omoplate = UnrealInt - Joint * 100;
        }
        else if (Joint == 3) {
            Shoulder = UnrealInt - Joint * 100;
        };
        Elbow2 = map(Elbow, 0, 99, 0, 85);
        if (Elbow2 > 85) {
            Elbow2 = 85;
        };
        if (Elbow2 < 0) {
            Elbow2 = 0;
        };
        Shoulder2 = map(Shoulder, 0, 99, 60, 170);
        if (Shoulder2 > 170) {
            Shoulder2 = 170;
        };
        if (Shoulder2 < 60) {
            Shoulder2 = 60;
        };
    }
}

```

```
};

Rotation2 = map(Rotation, 0, 99, 0, 180);
if (Rotation2 > 180) {
    Rotation2 = 180;
};
if (Rotation2 < 0) {
    Rotation2 = 0;
};
Omoplate2 = map(Omoplate, 10, 70, 10, 70);
if (Omoplate2 > 70) {
    Omoplate2 = 70;
};
if (Omoplate2 < 10) {
    Omoplate2 = 10;
};
ElbowServo.write(Elbow2);
ShoulderServo.write(Shoulder2);
RotationServo.write(Rotation2);
OmoplateServo.write(Omoplate2);
}
delay(40);
}
```

Figure 73. Arduino controlling robot arm

In the Arduino code all values are once again mapped to a new range, this time to the correct range of the servos. The reason this is not done in Unreal Engine is because these ranges differ from servo to servo. The omoplate and the elbow have a range of two digits and the shoulder and rotation have a range of three digits. To make sure all variables stay within this range a safety net is added. Each value is then sent to the corresponding servo.

This code worked fine for one servo, simply controlling the elbow was no problem, however controlling multiple servos proved to be problematic. Firstly, there is a power issue. The Arduino Uno that was used to test this could not power four servos at once. The second problem was speed. Sending four variables and having to wait each time to collect all the bytes and convert this back to an integer to work with gave big delays. It could take over 30 seconds to make the arm move into the desired position. Adding a delay in both Unreal Engine and Arduino to synchronize them helped, but not enough to make it faster.

8. Humanoid robot model VR integration and level design

One of the main tasks was to Design environment for the robot and replace the Unreal Engine 4 placeholder character with the model of Caroline humanoid.

8.1 Level design

The first task – level design, was mostly done on the low level due the fact that main purpose for the VR was not decided yet, based on the insufficient data and capabilities of the final integration.

For the given task basic level was created to test the First-person perspective then Third-person perspective and finally both with the VR glasses – Dell Mixed Reality and HTC VIVE were used. Unreal Engine 4 has built in tools and assets to create basic programming and level model. The ones that were used are First-person, Third-person and VR pre-set, each having different uses which furthermore have their advances and shortcomings.



Figure 74. Dell Mixed Reality headset



Figure 75. HTC VIVE Headset



Figure 76. Unreal Engine 4 Environment

8.2 Model integration

8.3 SketchUp

The second task required substantial more work to be completed – the model had to be made first, then optimized and polished to work inside engine environment and finally rigged to behave as a character.

This process started with getting the model. Fortunately, model was already made by Inmoov community inside SketchUp environment.

SketchUp is a 3D Design software mostly used by architects and interior designers due the ease of use and fast basic modelling, albeit not precise as some advance modelling programs SketchUp is easy to use, fast to learn, and has a very good export options to use models with other software.

SketchUp is an intuitive 3D modelling application that lets you create and edit 2D and 3D models with a patented “Push and Pull” method. The Push and Pull tool allows designers to extrude any flat surface into 3D shapes. All you have to do is click an object and then start pulling it until you like what you see. SketchUp is a program used for a wide range of 3D modelling projects like architectural, interior design, landscape architecture, and video game design, to name a few of its uses. The program includes drawing layout functionality, surface rendering, and supports third-party plugins from the Extension Warehouse. The app has a wide range of applications, including in the worlds of architecture, interior design, landscaping, and video game design. Sketchup has also found success with people who want to create, share, or download 3D models for use with 3D printers.^{xi}

The first problem encountered was that the model had all the internal parts modelled and the number of polygons was close to the max that 3D printer can print with normal settings. That being problematic to work with, meaning that a lot of optimization needed to be done on that front.

First thing to do in sketch up was to remove all the internal parts and the parts that are invisible from Third-person perspective. That greatly decreased the file size and the polygon count. File size at the first reduction was 93mb which is 25% reduction from original 124mb. But that was later proven to not suffice the Unreal Engine requirements for the model to be able to be rendered in real time on the GPU. Besides that, the arms and legs had to be removed and replaced with the Unreal Engine placeholders due the fact that original cannot be rigged properly. That is because there were to many independent parts and surfaces which as mentioned before are precisely detailed, which means the polygon count is to numerus. First

solution was, to reduce the polygons, but that was proven to be too destructive to the geometry details, that's why it was abandoned, and the replacement option was used instead. This further decreased the file size to 54mb, a reduction of extra 42%, in total file size was shrunk by 68mb, an 64% decrease. This greatly improved the further workflow.

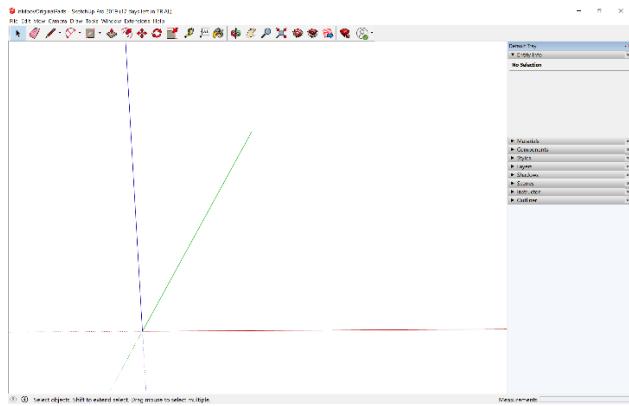


Figure 77. SketchUp Environment

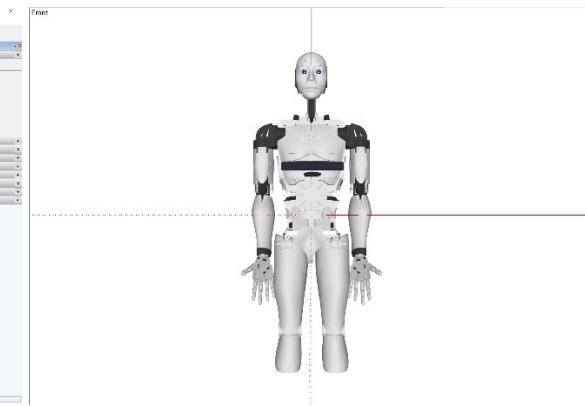


Figure 78. Original model

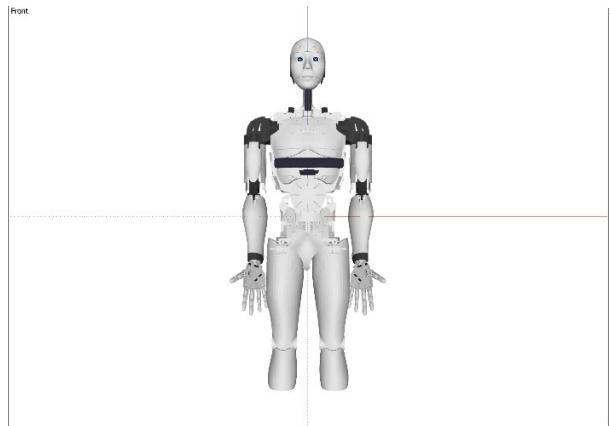


Figure 79. Model with removed internal parts

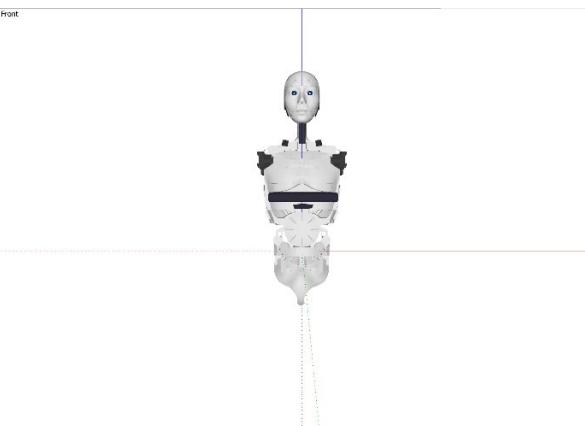


Figure 80. Model with removed limbs

8.4 Blender

At this stage work in SketchUp was mostly completed. Next up in the workflow was the Blender.

Blender is the free and open source 3D creation suite. It supports the entirety of the 3D pipeline—modelling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. Blenders mission is to build a free and open source complete 3D creation pipeline for artists and small teams. Advanced users employ Blender's API for Python scripting to customize the application and write specialized tools; often these are included in Blender's future releases. Blender is well suited to individuals and small studios who benefit from its unified pipeline and responsive development process. Examples from many Blender-based projects are available in the showcase. Blender is cross-platform and runs equally well on Linux, Windows, and Macintosh computers. Its interface uses OpenGL to provide a consistent experience. To confirm specific compatibility, the list of supported platforms indicates those regularly tested by the development team. As a community-driven project under the GNU General Public License (GPL), the public is empowered to make small and large changes to the code base, which leads to new features, responsive bug fixes, and better usability. Blender has no price tag, but you can invest, participate, and help to advance a powerful collaborative tool: Blender is your own 3D software.^{xiii}

Optimization was the main activity in Blender and continued to be the focus on the model before further work can commence. When the model was first imported into Blender, the given triangles number was 1,200,000. Note that the max number for current highly optimized, leading AAA titles is 150,000 to 100,000; under 50,000 being the recommended number. Blender works more closely to Game Engine than 3D modelling software that is why it automatically transforms polygons into triangles, generally one polygon equals to 2 triangles, that is because GPUs work only with triangles. Figure 82 showing the first import with the dissolve – collapse modifier already applied at 0,5, that reduces the number of triangles substantially, yet still far from desired. Black colored regions have too much triangles, blue is the recommended to work with. Figure 83 is showing further optimized humanoid robot to 400,000 triangles. A further 50% reduction from previous and 66.6 % from original import. That is the number that satisfied the quality and obtained details of the humanoid robot without sacrificing too much work performance of the computer, that is why it was used for continued work.

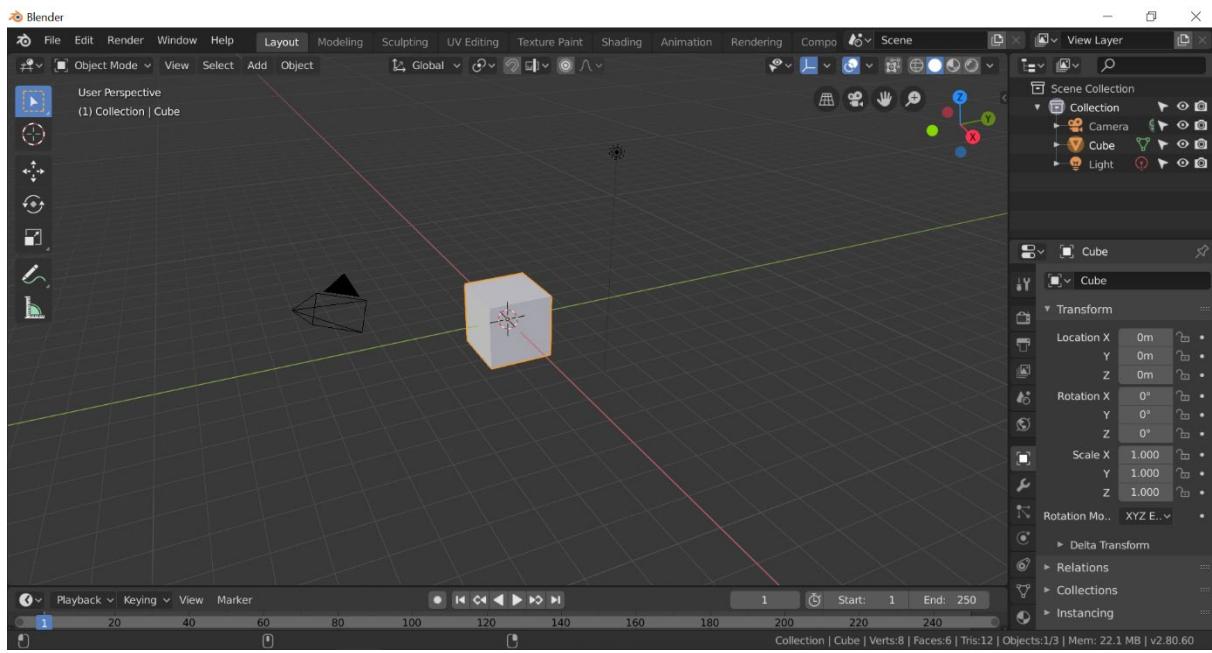


Figure 81. Blender Environment

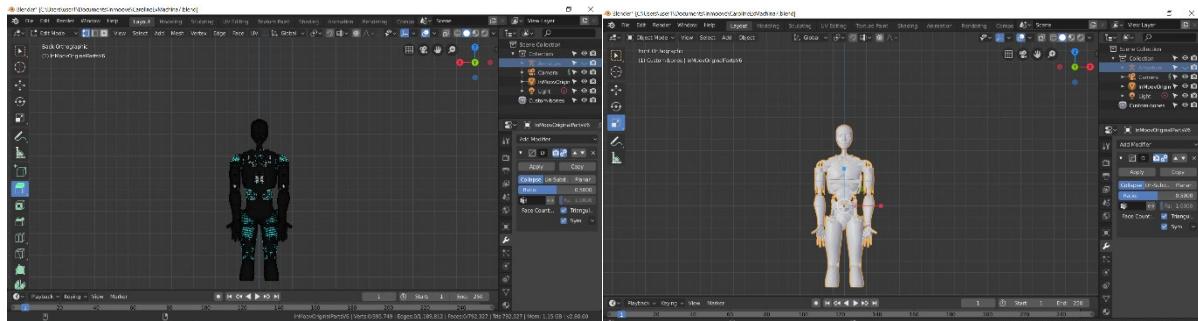


Figure 82. Detailed model display

Figure 83. Normal model display

Following up came the rigging process inside the blender environment. Besides the fact that the Unreal Engine is industry standard, encompassing multitude of aspects, rigging is not included in the current iteration. That is why it was done in blender.

Rigging is a technique used in skeletal animation for representing a 3D character model using a series of interconnected digital bones. Specifically, rigging refers to the process of creating the bone structure of a 3D model. This bone structure is used to manipulate the 3D model like a puppet for animation. Pretty much anything can be rigged. A space ship, a soldier, a galaxy, a door, it doesn't make any difference what the object is. Adding bones will allow any object to be animated freely. Rigging is most common in animated characters for games and movies. This technique simplifies the animation process and improves production efficiency. Once rigged with skeletal bones, any 3D object can be controlled and distorted as needed. In the entertainment industry rigging is a major step in the standard way of animating characters. Achieving smooth and complex animations is entirely dependent on the quality of the rigging phase in the animation pipeline.^{xiv}

The process started with creation of the armature and continued up with the weight painting, weight painting is used to attached armature (bones) to the specific parts of body. As briefly mentioned before, that is where major problems were found. The humanoid robot simply put was not good for the rigging, despite the effort put into optimization, that is due the fact that it has a hollow body and is composed of many different parts, some being interconnected, some not. The solution to that was to replace the parts that move the most, and that the final animation is going to reside in. That parts were the limbs. Limbs were replaced with the placeholder Unreal Model ones. Which was shown to be good solution, functionally and visually. Figure 84 shows the final model with the parts replaced. Armature was then merged with the body. The Unreal Engine model was again proven to be good solution as it required almost next to no weight painting due the existent integration of the bones. Figure 85 shows the armature of the humanoid robot model.

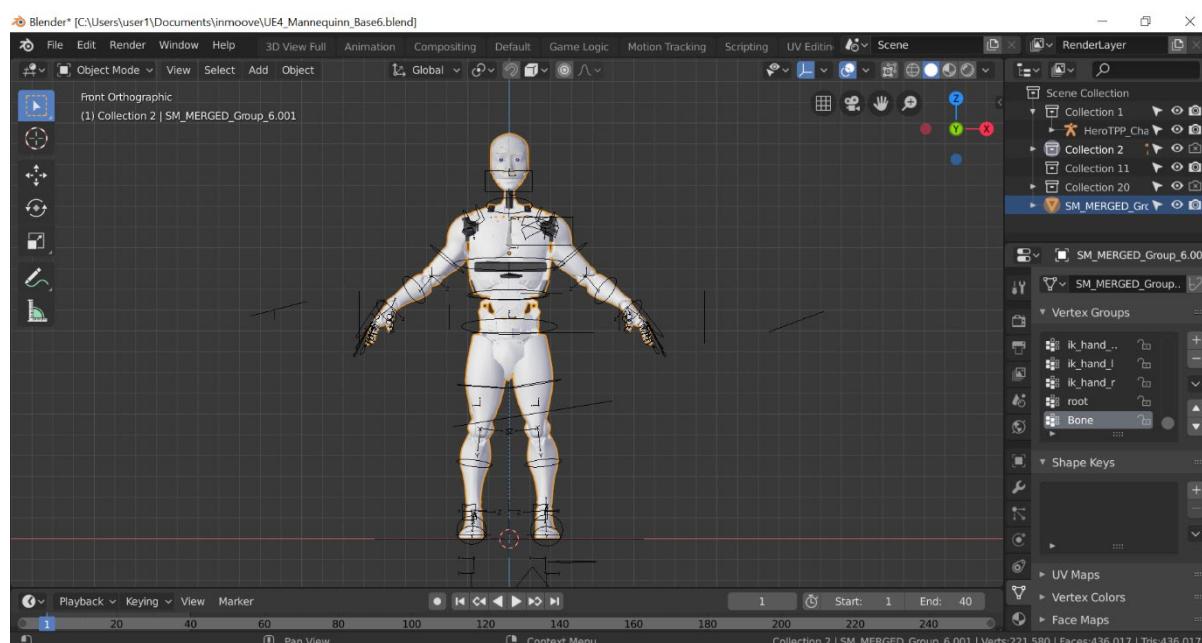


Figure 84. Final Blender model

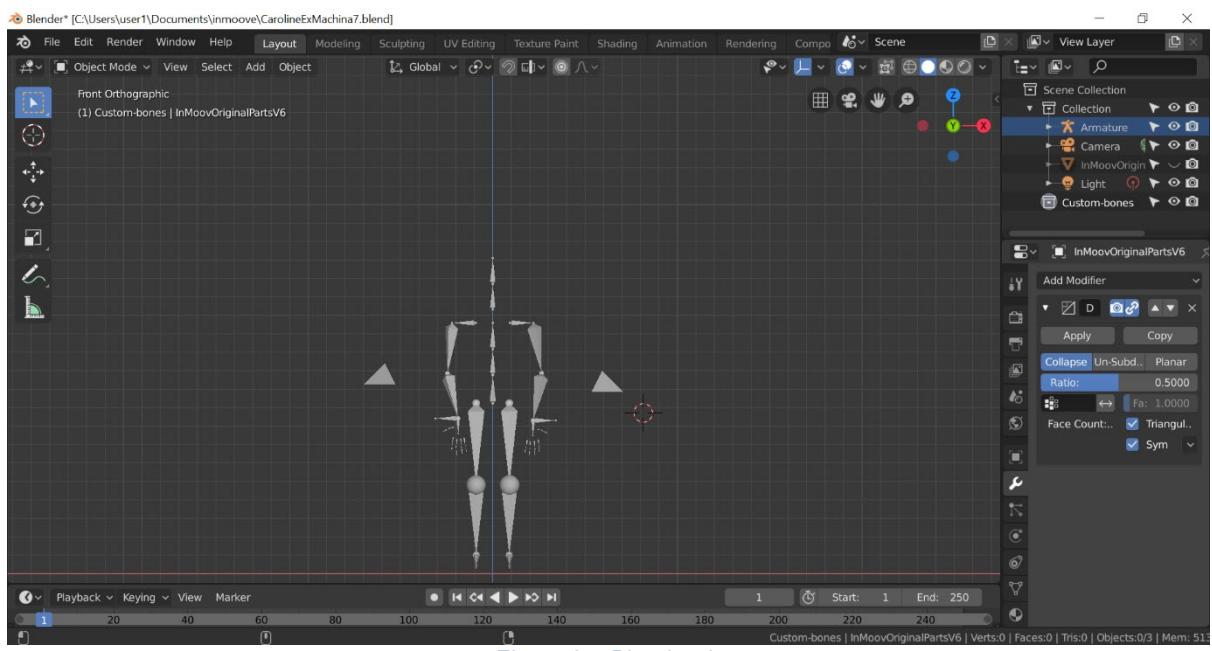


Figure 85. Blender rig

8.5 Unreal Engine 4

The final stage was the integration of the robot in the Unreal Engine and the replacement of the existing one with the use of retargeting.

Unreal Engine 4 is a complete suite of development tools made for anyone working with real-time technology. From enterprise applications and cinematic experiences to high-quality games across PC, console, mobile, VR and AR, Unreal Engine 4 gives you everything you need to start, ship, grow and stand out from the crowd. A world-class toolset and accessible workflows empower developers to quickly iterate on ideas and see immediate results without touching a line of code, while full source code access gives everyone in the Unreal Engine 4 community the freedom to modify and extend engine features.^{xv}

Work began with import of the final Blender model, import from Blender to Unreal Engine is not as smooth as from SketchUp. That's why after the import changes to the mesh had to be done, and a small problem based around Unreal Engine giving actor to every independent object persisted for some time. This was later fix by simple yet not always perfect merging of actors. Note that Unreal engine would create 1500 independent actors, which slowed down workflow considerably before problem was fixed. Figure 86 shows basic Unreal engine layout and Humanoid robot model next to the Unreal Engine one, note the actor problem in the top right corner.

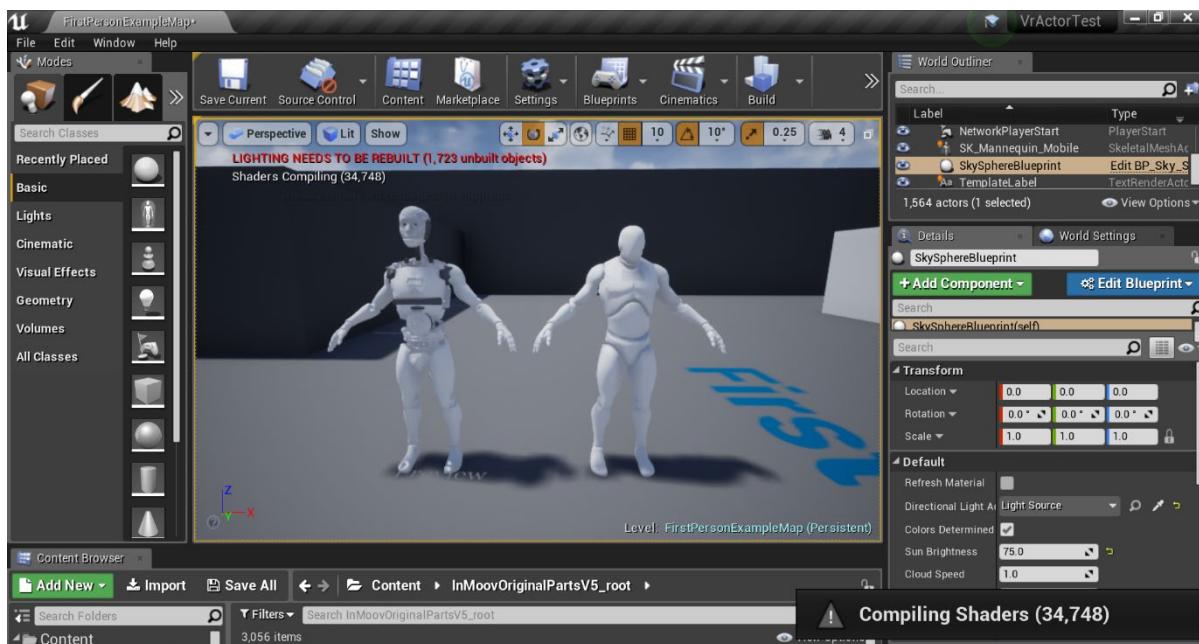


Figure 86. Humanoid next to Unreal Engine model

Then the final part of the Unreal Engine integration could commence. This was the rig retargeting. That task was done by first adding the Built-in animations of Unreal Engine to the humanoid robot such as standing still, walking and running. This was not an easy task, since the main body was prone to not animate correctly, which in turn gave an awkward animated output. After the completion of the task, humanoid robot was used to retarget the original so it could be used as a main actor for VR. At the end, animations were not perfect, but the concept works in a semi-realistic way, which is adequate. Figure 87 describes armature bones and physics of the humanoid robot. Figure 89 shows final model as an actor inside prebuilt environment.

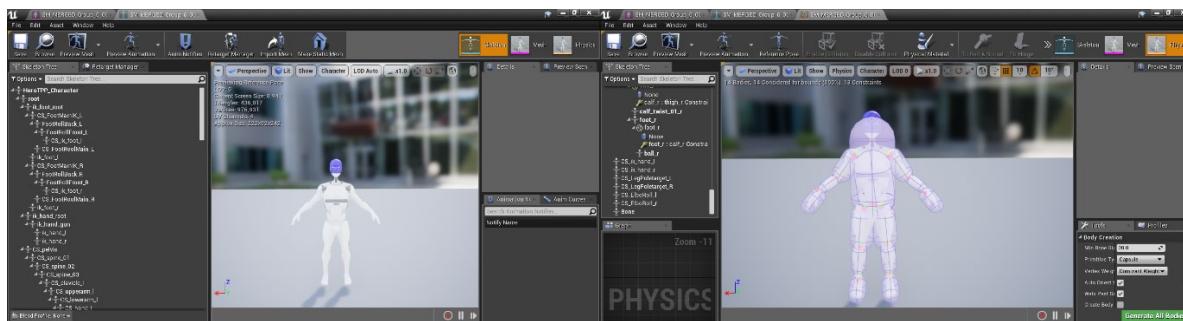


Figure 87. Model skeleton

Figure 88. Model Physics



Figure 89. Final Unreal Engine model

9. Conclusion

In this report the achieved work during this project is described. It contains several project management tasks, a philosophy presentation, optimized hardware, software improvements and the connection with Virtual Reality.

Most of the hardware for the humanoid robot had been repaired and replaced. To find a purpose and utility for the humanoid robot the team did a brainstorming session. The outcome of which was that best fields to use this robot are education and information. These fields were further explored by creating multiple scenario's, the librarian was chosen as the final purpose.

To implement this librarian scenario to the humanoid an introduction code with gestures was written. To make the robot able to pick up stuff with the hands the finger pressure sensor was optimized. The PIR was tested to this could be of use for implementing gestures. Also, the robot was made fully independent by attaching it to the battery. A switch was added to the battery to turn the robot on and off easily.

Finally, VR was implemented to control the robot with the Virtual Reality controllers. This was achieved by setting up a communication between Unreal Engine and Arduino. The robot was modeled in a VR environment, so the movements made with the controllers would show visually VR.

To conclude did this project make a lot of progress and with the report is all this progress well documented. The documentation of this report is not only to show what has been achieved but also, for the next group that will continue working with the InMoov robot to give the right information on how to proceed. In addition, there is a website about the project, where there is contact space in case of any doubt: <https://technorobot.webnode.com/>
In the next chapter are a number of recommendations given for a good continuation of the project.

“A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.”

- *Alan Turing*

10. Recommendations

The conclusion of this project has been discussed but that is not the end of this project just yet. As this project is still ongoing the team gives a few recommendations for further work with the humanoid robot. These recommendations are summed up in the list below:

- Doing the robot completely independent by attaching chemical batteries to the Kinect, as it is explained in the InMoov website.
- Adding a QR reader to the eye cameras as a complement to the librarian scenario. There is some information about it on the InMoov Google group.
- Adding a bar code to the Kinect as a complement to the librarian scenario.
- Implement all the robot in Virtual Reality.
- Attaching the robot to the OMRO so it can move independently.
- Replacing the fishing lines of the fingers by plastic lines.
- Improving the Kinect, so the robot can copy movements with it.
- Getting a new tablet which fulfills the specifications (proper size and at least 4Gb RAM).
- Adding a screen or a picoprojector to add value to the information that the robot gives.

References

- ⁱLORENZONI, Federico, Patricia HAHLBOHM, Carlos RODRÍQUEZ, and Matthijs SOUILLJEE. R3-Dfin, OPEN-SOURCE PROJECT 3D PRINTED LIFE SIZE ROBOT. Rep. Print.
- ⁱⁱLangevin, G. (2012, January). InMoov. Retrieved March 25, 2019, from <http://inmoov.fr/>
- ⁱⁱⁱWelcome | European Project Semester. (2019). Retrieved March 25, 2019, from <http://www.europeanprojectsemester.eu/>
- ^{iv}Belbin, M. (2019). The Nine Belbin Team Roles. Retrieved March 25, 2019, from <https://www.belbin.com/about/belbin-team-roles/>
- ^vGlobal Petrol Prices. (2019). Finland electricity prices, June 2018. Retrieved March 25, 2019, from https://www.globalpetrolprices.com/Finland/electricity_prices/
- ^{vi}SoftBank Robotics. (2006). NAO the humanoid robot | SoftBank Robotics EMEA. Retrieved March 25, 2019, from <https://www.softbankrobotics.com/emea/en/nao>
- ^{vii}Génération Robots. (2018, July 31). All about Romeo – Nao's big brother robot. Retrieved March 25, 2019, from <https://www.generationrobots.com/blog/en/romeo-naos-big-brother-robot/>
- ^{viii}SoftBank Robotics. (2006). Pepper the humanoid robot | SoftBank Robotics EMEA. Retrieved March 25, 2019, from <https://www.softbankrobotics.com/emea/en/pepper>
- ^{ix}Engineered Arts Limited. (2017). RoboThespian, The Acting Humanoid Robot. Retrieved March 25, 2019, from <https://www.engineeredarts.co.uk/robothespian/>
- ^xRainbow Robotics. (2017). Rainbow Robotics DRC-HUBO. Retrieved March 25, 2019, from <http://www.rainbow-robotics.com/new/>
- ^{xi}Langevin, G. (2012, January). How to create gestures for InMoov. Retrieved March 25, 2019, <http://inmoov.fr/how-to-create-gestures-for-inmoov/>
- ^{xii}T. (2019). 3D Design Software | 3D Modeling on the Web. Retrieved May 10, 2019, from <https://www.sketchup.com/>
- ^{xiii}Foundation, B. (2019). Home of the Blender project - Free and Open 3D Creation Software. Retrieved May 10, 2019, from <https://www.blender.org/>
- ^{xiv}Petty, J. (2018, October 05). What is 3D Rigging For Animation & Character Design? Retrieved May 10, 2019, from <https://conceptartempire.com/what-is-rigging/>
- ^{xv}E. (2004). Unreal Engine. Retrieved May 10, 2019, from <https://www.unrealengine.com/en-US/>