Development of a Self-Balancing Robot

Module: PDE2440 Robotics and Mechatronics
Tutor: Professor Mehnaz Mahaboob
Student number: M00717681
Student Name: Yago Adan Gonzalez Carrizo

Table of Contents

Brief

The objective of this project is to design, fabricate and evaluate the performance of a self-balancing robot that can efficiently pick up and transport an object to a predetermined location while maintaining balance on two wheels. The robot will be manually controlled, and it will be required to signal a successful pick-up by illuminating a green LED. Upon reaching the designated location, the robot will place the object and indicate a successful placement by flashing a red LED.
The design of the robot will focus on its mechanical and electronic components. The project timeline spans a period of 10 weeks.

Purpose of my Robot

The focus of my product is to provide a solution to companies that require self-balancing two-wheel robots for the transportation of heavy items. The robots are designed to catch and carry items securely, without dropping them, while maintaining their balance. The technology behind the robots makes them ideal for use in a variety of industries, such as construction, logistics, and manufacturing.
For example, in a construction site, workers can throw bricks, tools, or other materials to the robot, which can catch them while balancing itself, thereby eliminating the risk of accidents and reducing the workload of the workers. Similarly, in logistics or warehouse settings, the robots can be used to transport heavy items such as boxes, packages, or containers to various locations, freeing up workers' time for other tasks.
Moreover, the robot's platform can be used as a mobile work surface, allowing workers to place tools, equipment, or parts on it and move around the site as needed. This feature can be especially useful in large manufacturing plants, where the robots can carry parts and equipment between assembly lines.
In summary, my product aims to meet the needs of companies that require a reliable and versatile solution for the transportation of heavy items. The robots' self-balancing capabilities, coupled with their platform's mobility, make them an excellent choice for various applications across multiple industries.

Project schedule management

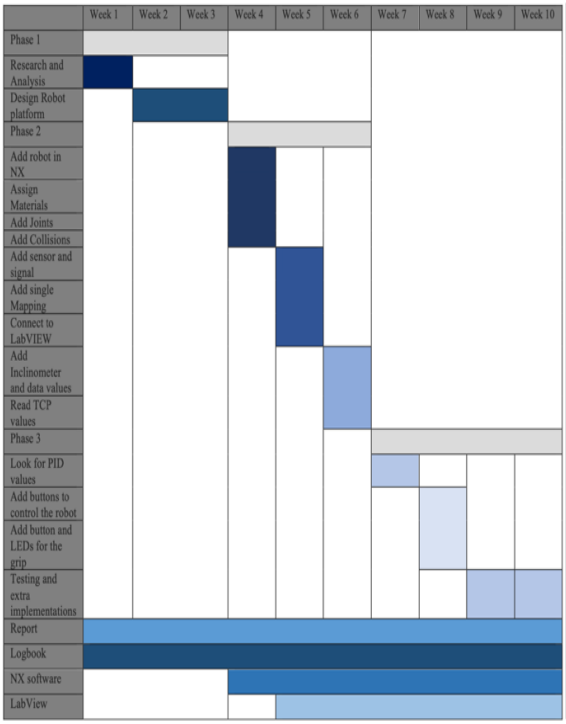
Project Title: Development of a Self-Balancing Robot
Project Timeline: 10 weeks
Project Objectives:

- Sketch the initial design of the robot
- Create a detailed CAD model of the robot and prepare accompanying drawing sheets
- Implement self-balancing functionality of the robot
- Enable manual movement control of the robot
- Design and implement a gripper mechanism for the robot
- Implement autonomous movement of the robot (optional)

Milestones:

- Completion of research and analysis (Week 1)
- Creation of CAD model for the two-wheel robot and importation into NX (Week 3)
- Integration of physics and sensors into NX and development of basic LabView code (Week 6)
- Finalization of PID values (Week 7)
- Completion of development and testing (Week 10)

Gantt Chart/PDS



Task	Description	Priority	Status	Deadline
Research and Development	Research on the different types of self-balancing robots, sensors and actuators used in the design as well as materials.	High	Not started	16/01/2023
Component Selection	Select components for the robot, such as motors and sensors	High	Not started	18/01/2023
Design robot platform	Design the physical structure of the robot, including the chassis, wheels, and sensor and make a prototype using CAD software.	High	Not started	25/01/2023
Import CAD to software (NX)	Import the prototype model to NX and assign materials, fixed joints and collisions.	High	Not started	01/02/2023
Connect both software to move robot	Add sensors and signals and single map everything so NX can connect to LabView.	High	Not started	05/02/2023
Add Inclinometer and data values	Add inclinometer to the robot to measure the angle or slope of an object with respect to gravity. Then read TCP values.	High	Not started	08/02/2023
Software Development	Develop the program using LabView to control the robot, including the PID control algorithm	High	Not Started	15/02/2023
Test robot	Test the robot's balance and movement in various conditions and make necessary adjustments.	High	Not started	20/02/2023
Integrate additional features	Add additional features to the robot, such as buttons to control the movement.	High	Not started	24/02/2023
Debug and refine code	Debug the code and refine it to optimize the robot's balance and movement.	Medium	Not started	03/03/2023
Integrate additional features	Add additional features to the robot, such as control of the grip and making the robot move automatically	Medium	Not started	10/03/2023
Final testing	Conduct final testing to check that the tasks have been accomplished.	High	Not started	11/03/2023
Optimization and Fine-tuning	Optimize the robot's performance and fine-tune the controls based on the testing results.	Medium	Not started	15/03/2023
Make a video	Record a video and upload to YouTube as unlisted, explaining the process and how it works in order to demonstrating the program and the solution found.	High	Not started	20/03/2023
Documentation	Check that the report is well written, and the logbook was up to date with all the progression and data.	High	Not started	22/03/2023
Submission Day	Submit all the work done as a zip file and attach a pdf with the link to the YouTube video before 6pm.	High	Not started	23/03/2023
VIVA	Present the robot and demonstrate the knowledge and skills learnt.	High	Not started	24/03/2023

While having a well-structured project plan is a great start, it is important to remember that there are many factors that can affect the success of a project. It is important to be flexible and adaptable throughout the project and to anticipate any potential challenges or setbacks that may arise. Ultimately, the success of the project will depend on the effort and dedication towards achieving the project goals.

Research

Prior to designing a robot, it is imperative to perform a thorough analysis of the various components, materials, and options available. Careful consideration of these factors can significantly impact the final outcome of the project. Once the necessary components have been identified, an examination of existing two-wheel robots and their skeletal structure can provide valuable insight into design considerations. By reviewing the designs of other robots, we can gain a deeper understanding of the technical criteria required for a successful self-balancing robot design.

Components

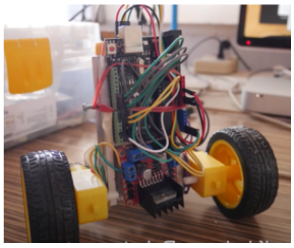
Component	Function	Requirements	Options	Final Component
Frame/Skeleton	Supports all components and provides stability	Strong, lightweight, easy to assemble.	MDF, Acrylic or ABS	Acrylic
Screws	Joins the components	Strong and easy for assembling	Mechanical or Drywall screw	Mechanical
Microcontroller	Controls the robot	Fast processing, multiple inputs and outputs	Arduino, Raspberry Pi or ESP32	LabVIEW
IMU sensor	Measures robot's orientation	Accurate, responsive	MPU-6050 or LSM6DS3	
Wheels	Supports the robot and provides traction	Durable, lightweight, good grip	Rubber, Foam, or ABS	ABS
Motor	Provides motion to robot	High torque, variable speed, brushless	DC or stepper motor	DC motor
Battery	Powers the robot	High capacity, rechargeable	Lithium-ion or NiMH battery	Lipo Battery (5000 mAh)
Control algorithm	Balances the robot	Accurate, efficient	PID or LQR control	PID
Communication module	Sends and receives data from external devices	Reliable, low latency	Bluetooth, Wi-Fi, TCP	TCP

Exemplar Robots

Arduino Nano

The initial prototype of the robot employs a range of components, including an Arduino board, which serves as the main controller, an MPU6050 gyroscope, which provides motion sensing capabilities, and A4988 stepper motors that control the robot's movement. The robot also features two wheels and a 12V battery, which powers the system. Additionally, a custom PCB board has been developed to facilitate the integration of these components, and an HC-05 Bluetooth module enables wireless communication.
The physical structure of the robot is composed of MDF (medium-density fibreboard), which provides a sturdy and durable base for the components. The components are attached to the first two levels of the structure, with the gyroscope and motors located on the top level and the Arduino board and other electronics located on the bottom level. The design ensures that the robot is compact and lightweight, making it easy to transport and manoeuvre. Furthermore, larger wheels have been employed to minimize the number of turns required for the wheels to maintain balance. This design choice not only improves the efficiency of the robot but also enhances its stability, making it less prone to tipping over or losing balance. Overall, the use of high-quality components and thoughtful design choices has resulted in a functional and reliable robot prototype. Its compact size, wireless capabilities, and stable design make it a promising solution for a variety of applications, including transportation and surveillance.
[1].

Arduino Uno



The second robot prototype was designed with the objective of achieving greater balance and stability, while also being more compact and lightweight. To this end, the robot was built using an Arduino board, an MPU6050 gyroscope, an L298N drive controller, two DC motors, a breadboard, a battery, two wheels, and a set of wires. By utilizing these components, the robot can sense its own motion and make the necessary adjustments to maintain balance. As shown in the accompanying image, the components of the robot are tightly packed together, resulting in a compact structure that is both efficient and easy to balance. The weight distribution of the robot is also centralized, which further enhances its stability and balance.
In order to ensure that the robot can maintain its balance even when subjected to external disturbances, a set of PID values were determined for optimal performance. These values include a proportional gain of Kp = 50, a derivative gain of Kd = 0.5, and an integral gain of Ki = 0. These values were carefully selected to enable the robot to make the necessary adjustments to maintain balance and stability, even in the presence of external forces. Overall, the use of high-quality components and careful design choices has resulted in a functional and reliable robot prototype. Its compact size, centralized weight distribution, and stable design make it an ideal solution for a variety of applications, including surveillance and monitoring [2].

ESP32

The third robot prototype was designed with a focus on performance, utilizing an ESP32 microcontroller in place of the traditional Arduino board due to its superior processing power. Comprising a range of carefully selected components, including an MPU605 gyroscope, DVR8825 stepper motors, a custom PCB board, a 12V battery, two wheels, and a set of wires, the robot was constructed with a sturdy wooden frame. All the robot's components are strategically positioned across the first level of its structure, resulting in a well-balanced and stable design. Furthermore, two bricks have been added to the top of the robot to provide additional protection against damage from falls or impacts. By incorporating these design elements, the robot is better equipped to withstand the rigors of real-world use and maintain its functionality over time.
Overall, the use of high-quality components and thoughtful design choices has resulted in a powerful and reliable robot prototype. Its advanced microcontroller, precise sensors, and sturdy construction make it an ideal solution for a wide range of applications, from industrial automation to environmental monitoring [3].

Matrix table

Despite their similarities in appearance, each of the robots analysed presents unique design choices and technical specifications that make them distinct from one another. As such, it is important to undertake a careful comparative analysis of each robot in order to determine which one will be most suitable for my needs.
This analysis will involve a thorough examination of each robot's technical specifications, such as the type of microcontroller used, the choice of sensors and motors, the physical layout of components, and the overall stability and balance of the structure. By assessing each robot in detail, I can gain a deeper understanding of the technical trade-offs involved in various design choices, and identify which robot is best suited to meet the specific requirements of my project.

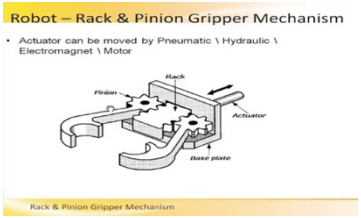
Once this analysis is complete, I will be in a better position to determine which design elements are most essential for my own robot's structure. By incorporating the most effective design features of each robot and applying them in a unique way to my own project, I can create a robot that is optimized for my specific needs and requirements.

Variants / Technical criteria	Arduino Nano Robot	Arduino Uno Robot	ESP32 Robot
Stronger structure	2	2	1
Lightest structure	2	1	4
Efficient	1	1	1
Simple constructuion	2	1	2
Total	7	5	8

As the results show, the most optimised robot is Arduino Uno robot as it has the lowest number in my matrix table.

Grip

Rack and pinion gripper mechanism

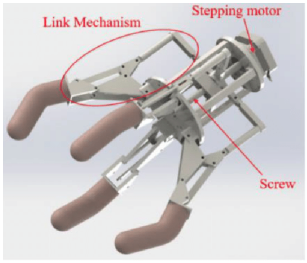


A rack and pinion grip mechanism are a robotic gripping system that employs a rack and pinion system to securely grasp and hold an object. Typically mounted on a moving platform or robotic arm, the mechanism moves towards the object to be grasped when activated. The mechanism opens and closes by bringing together or separating the rack and pinion, respectively. The teeth of the rack and pinion system mesh together to securely grasp the object, and the grip's force is dependent on the pressure applied to the rack and pinion system. This mechanism is well-suited for a variety of robotic applications requiring precise and secure object manipulation [4].

Advantage	Reason	Disadvantage	Reason
Strong grip	The system provides a robust and stable grasp on the object.	Limited range of motion	The mechanism's range of motion may be limited as it usually provides a linear motion.
Precise control	The mechanism enables the operator to have a precise control over the grip.	Limited adaptability	The system is optimized for gripping objects of specific dimensions and contours
Reliability	The mechanism has a long-standing history of successful implementation across various applications, making it a reliable and time-tested choice for gripping tasks.	Complexity of control	Controlling the mechanism to grasp and manipulate objects can be challenging.
Durability	The mechanism is commonly manufactured using high-quality materials, ensuring durability and longevity.	Cost	The mechanism may entail higher manufacturing and maintenance costs.

To conclude this, based on the aforementioned information, the rack and pinion grip mechanism can be considered as a reliable and efficient means of securely grasping objects with a strong grip, making it a viable option for a range of applications.

4-finger soft gripper



The flexible finger gripper is a robotic mechanism that utilizes a set of pliable appendages to grasp an object. The fingers are engineered to be adaptable and shape to the contours of the target object, providing a secure hold. The release mechanism functions by gradually reducing the grip pressure, allowing the fingers to gently and accurately open, ultimately releasing the object with precision. The gripper utilizes a rigid screw and connecting rod to facilitate the grasping motion [5].

Advantage	Reason	Disadvantage	Reason
Gentle grip	The mecchannism offers a delicate grip that minimizes the chances of damaging or distorting the object being grasped.	Limited gripping force	It may not provide as high a level of gripping force as other types of gripper mechanisms,
Versatility	The ability to conform to various shapes and sizes of objects enhances its versatility and adaptability.	Limited speed	The mechanism's speed of operation may be limited by the use of a pneumatic or hydraulic actuation system
Precise control	The actuation system in the gripper mechanism permits accurate control over the gripping action, facilitating the operator to position the object with precision.	Limited temperature range	Thermal limitations that could restrict its applicability in high-temperature environments.
Low maintenance	Constructed using sturdy and long-lasting materials, which results in lower maintenance and repair requirements.	Cost	It is quite expensive compared to other mechanisms

To conclude this, the 4-finger soft gripper mechanism offers versatility and adaptability in grasping objects, with the advantage of providing a secure and gentle grip that minimizes the risk of damage or deformation.

4 claw gripper mechanism (hydraulic)

This robotic gripping mechanism is designed with four claws arranged symmetrically around a central point, allowing for a secure and balanced grip on the object being held. The claws can be controlled to spread apart and conform to the shape of the object, ensuring a secure grip without damaging or deforming it. One unique aspect of this particular mechanism is the use of water as an actuation method, which allows for precise control over the movement and grip of the claws. The use of water also has the advantage of being environmentally friendly and cost-effective [6].

Advantage	Reason	Disadvantage	Reason
Stable grip	Provides a stable and evenly distributed grip on the object, minimizing the risk of slippage or damage.	Limited flexibility	The mechanism can grasp a wide range of objects, its versatility may not be sufficient for specific objects or scenarios that require a specialized gripping mechanism.
Versatility	The mechanism is capable of grasping a variety of objects, including irregularly shaped ones.	Limited grip force	The mechanism may be relatively weaker than some other types of gripping mechanisms.
Simple design	mechanism's simplicity makes it straightforward to design, construct, and sustain.	Difficulty with delicate objects	The mechanism may apply too much force on delicate objects.
Efficiency	The symmetric arrangement of the claws in the mechanism allows for a rapid and efficient grasping motion	Complexity of control	The mechanism’s control to grasp and manipulate objects can be more challenging, especially when dealing with delicate or complex objects.

To conclude this, the 4-claw grip mechanism provides a simple yet effective solution for grasping and manipulating a wide range of objects, with its symmetric design allowing for efficient motion.

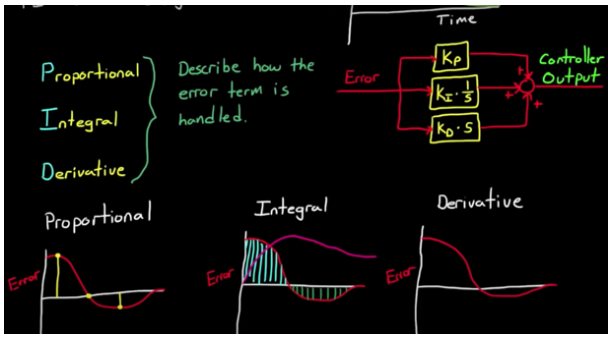
Matrix table

Variants / Technical criteria	Rack and pinion gripper	4 finger soft grippers	4 claw grippers
Strong grip	1	3	3
Precise control	1	3	1
Range of motion	3	2	2
Durable	1	2	2
Cost of production	3	4	5
Total	9	14	13

The matrix table provides a comprehensive comparison of the grip designs and their performance based on the technical criteria set for the project. After evaluating the results, it is concluded that the rack and pinion grip design outperforms other designs and is the most suitable choice for the project.

PID control

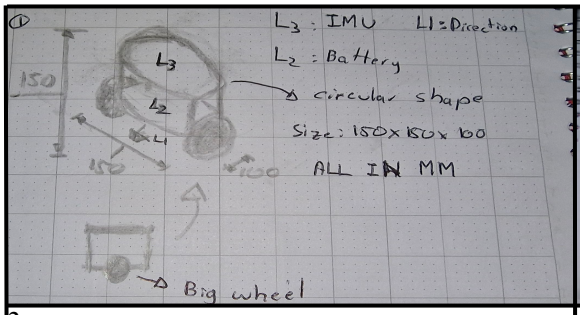
PID stands for "Proportional Integral Derivative and is a type of feedback that regulates and stabilises a system or a process This is done by measuring the error value between the setpoint and the current state of the process [7]. The most optimised way of finding the correct PID values is by tuning first the Proportional value until the system starts oscillating. Once this is achieved, the next step is to tune the Derivative value in order to reduce the oscillation value. Finally, the Integral value allows the device to stabilised faster. Most of the times it is not necessary to tune the Integral value.



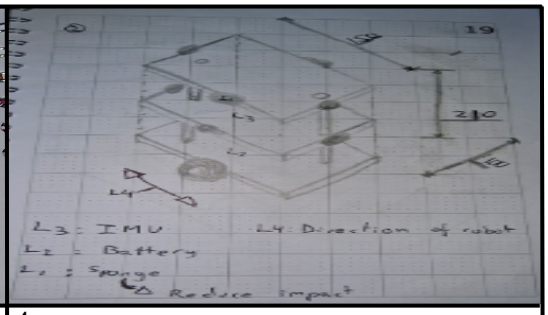
Designs

Once the research was done, the next step was to design different structures for the 2 self-balancing robot and compare them in order to see which structure is the most reliable.

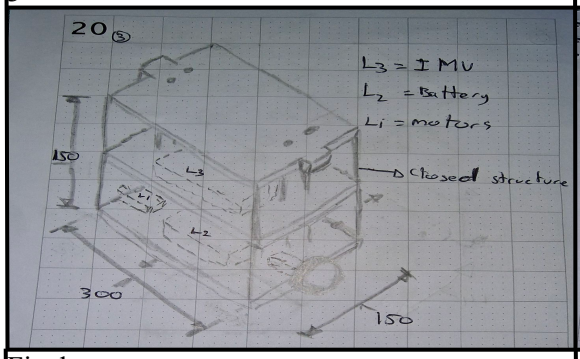
1



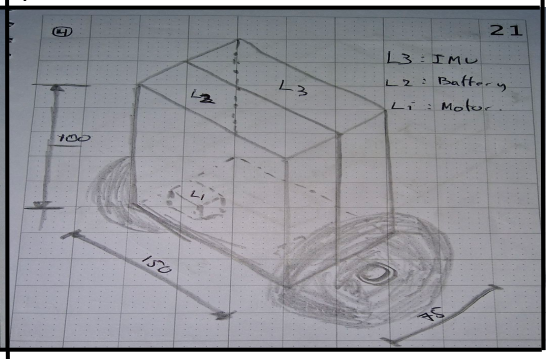
2



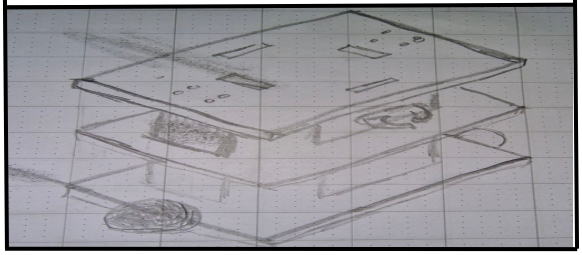
3



4



Final



Matrix Table

Variants / Technical criteria	1	2	3	4	Final
Simple construction	2	4	3	2	3
Strongest structure	4	2	1	4	1
Most centred components	1	2	1	1	1
Most organised structure	3	2	1	2	1
Most suitable mechanism	1	3	1	1	1
Lightest structure	1	4	4	1	3
Total	12	17	11	11	10

This matrix table lists the different components required for a self-balancing robot project, along with their respective functions, requirements, and options. The requirements column describes the specifications or characteristics that the component must have to perform its function effectively. The options column provides different choices or alternatives that can be used to fulfil the function or requirement.
This matrix table can be used as a reference guide for selecting the components needed for the self-balancing robot project. It allows the project team to compare different options and choose the components that best fit the project's specific needs and requirements.

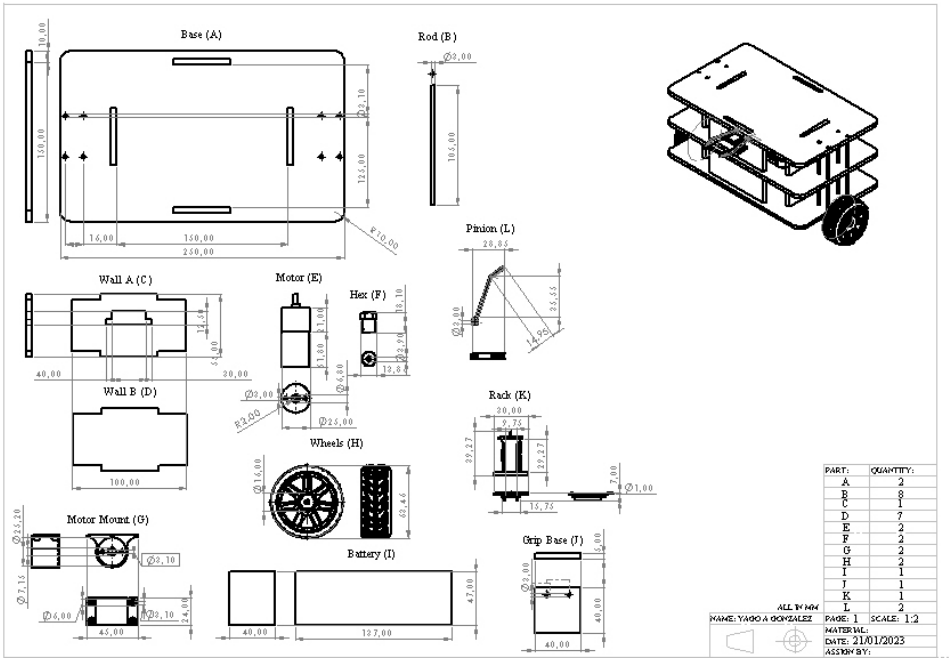
Final Design

Materials and components for the robot

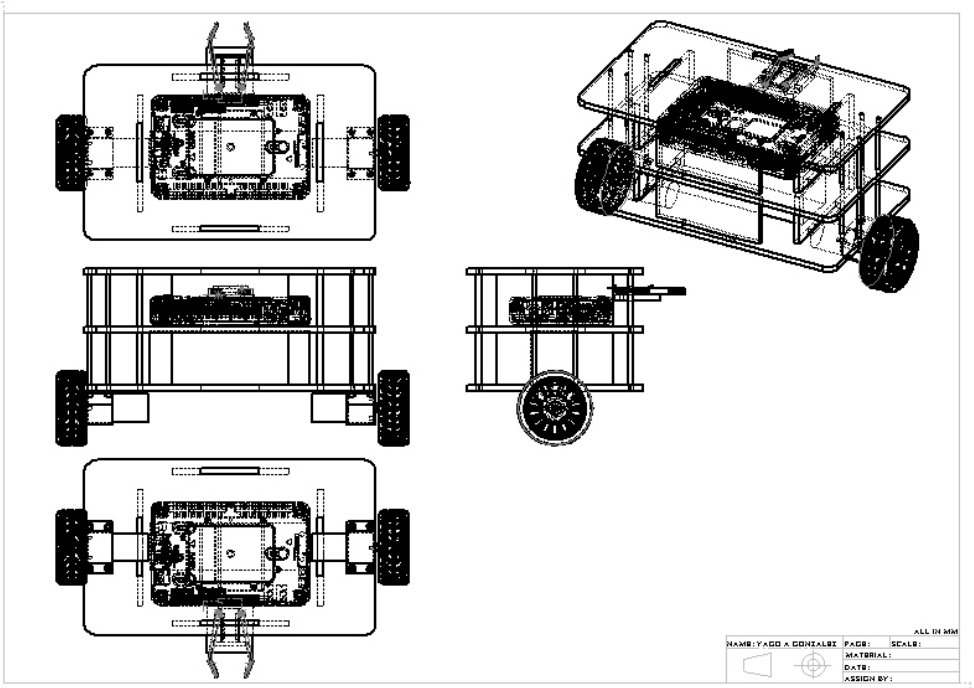
Component	**Final Component**	**Function**	**Reason**
Frame/Skeleton	Acrylic	Supports all components and provides stability	Has a tensile strength of 10000psi, higher than MDF (4500psi) and ABS (4100psi)
Screws	Mechanical	Joins the components	Strong and easy for assembling
Microcontroller	LabVIEW	Controls the robot	Fast processing, multiple inputs and outputs
IMU sensor		Measures robot's orientation	Accurate, responsive
Wheels	ABS	Supports the robot and provides traction	Durable, lightweight and easier to make
Motor	DC motor	Provides motion to robot	High torque and variable speed.
Battery	Lipo Battery (5000 mAh)	Powers the robot	High capacity
Control algorithm	PID	Balances the robot	PID is accurate and efficient.
Communication module	TCP	Sends and receives data from external devices	It is reliable and has a low latency.

After doing a lot of research, I have concluded that these are going to be the components for my 2-wheel self-balancing robot.

Components

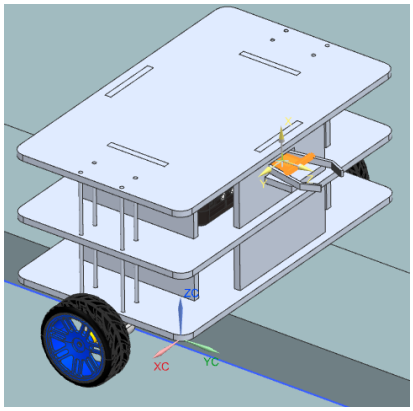


Robot



The 2-wheel robot has been assembled with the controller and battery situated at the centre of the structure, enclosed by walls to prevent their displacement in the event of a fall. This central placement enhances the robot's balance control. Moreover, long rod bars have been added to maintain the structural integrity of the robot and uphold a symmetrical design, facilitating ease of assembly. It is worth noting that the current configuration may pose a challenge in the event of power supply replacement as the disassembly process is time intensive.

Isometric view

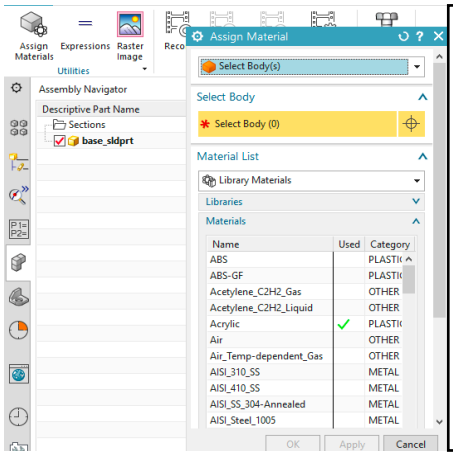


Software Process

Now that there is a CAD model for the 2-wheel self-balancing robot. It will be necessary to import it to Siemens NX so I can start making the robot balance. The 3D model has to be saved as a STEP214 file and import it into a new assembly in NX. Once the robot is imported, I need to use the "Dynamics XYZ controls to rotate the robot to the correct rotation.

{IMAGE}

Assigning materials

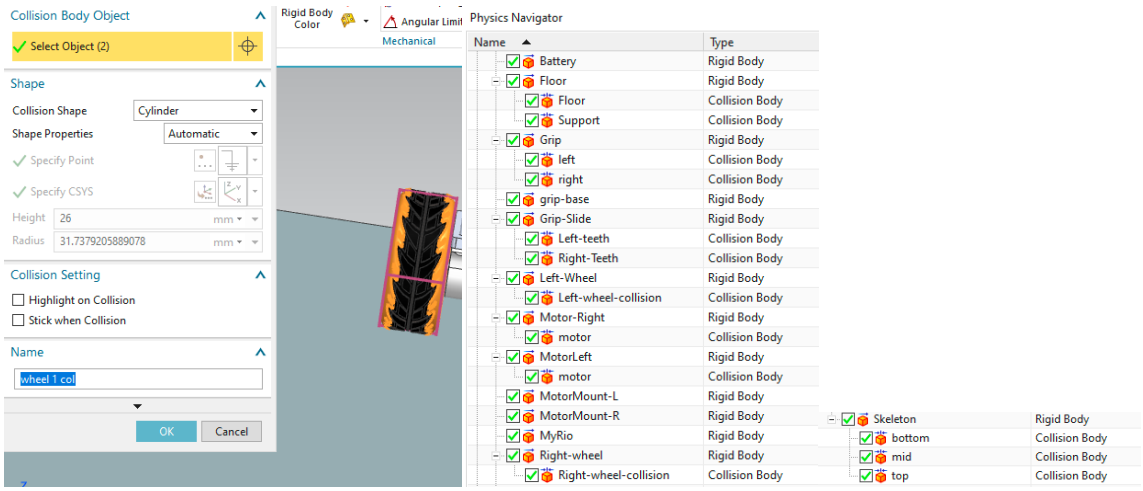


Once the robot is correct orientated, I will need to assign the materials for all the components because by assigning materials in Siemens NX, the software can calculate the correct material properties such as density, stiffness, and thermal conductivity, which are crucial for accurate simulations.
The materials for each component were:

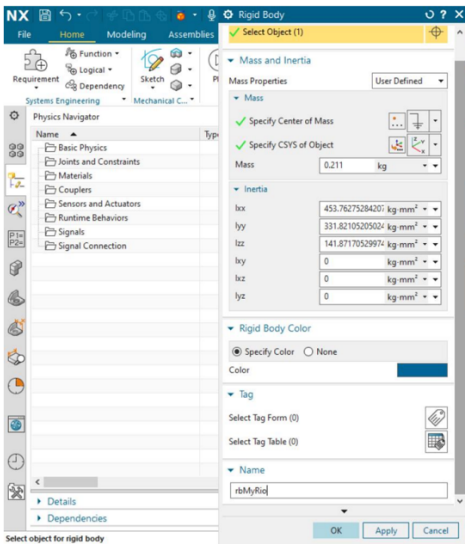
Component	Material
Skeleton	Acrylic
Wheels	ABS
Motor mount	ABS
Hex	Brass
DC motor	Steel
Bar	ABS

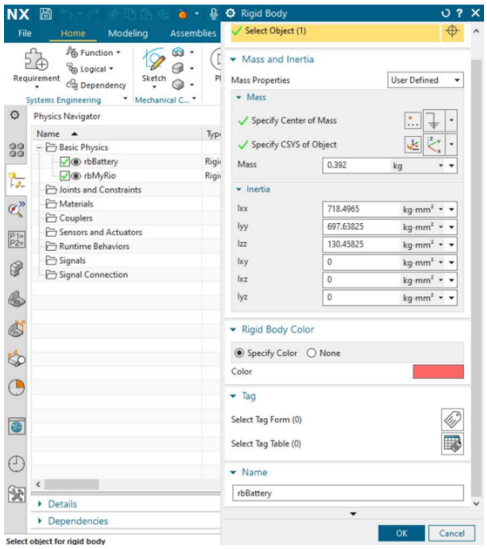
Basic physics

The next step is to add rigid bodies and collisions. By adding rigid bodies to each component, the software will be able to apply the right amount of weight for each component, depending on their material and how big the part is. Additionally, not all the components will have. A collision body as some parts will not collide with obstacles, for example the bar will not require a collision as it will not touch the floor when the robot falls. On the other hand, the skeleton of the robot will collide with the floor. Finally, I needed to create a floor and add its collision body, so the robot has a platform to move around.

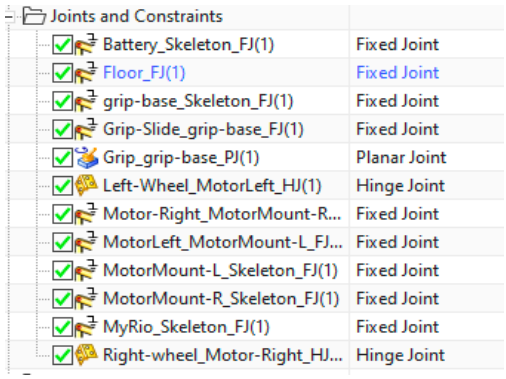


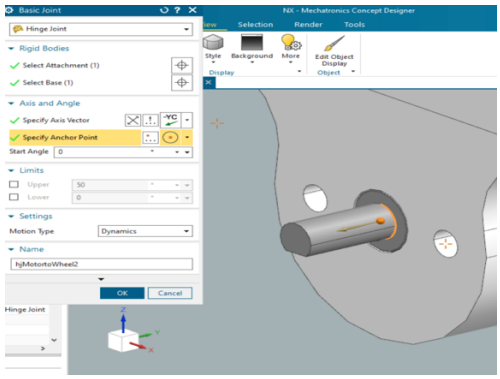
Once all the rigid and collision bodies are selected, I had to change the weight manually for some components such as the skeleton, the controller 'myRio' and the battery because the software gave me incorrect weights, which lead to errors when creating the joints. Therefore, I have applied a weight of 0,211kg for the controller, 0,392kg for the battery and 0,400 for the skeleton.





Joints and Constraints



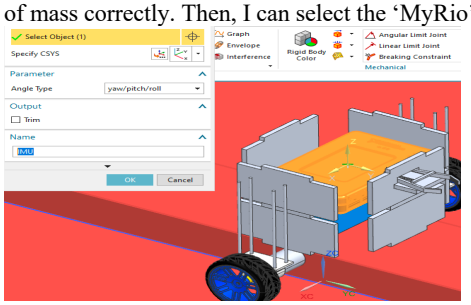


The next step was to create joints between each component, so the robot does not collapse when running the simulation. For the components that do not require to move will have a fixed joint. The way it works is by selecting a base component and attaching another one. For example, the motor mount will be attached to the skeleton structure, so it can later hold the motor. In addition, I will use a hinge joint for the wheel and the motor, the axis vector will be the axis my wheel will rotate around. This joint will allow the wheels to spin around when the simulation is running.

Sensors and Actuators

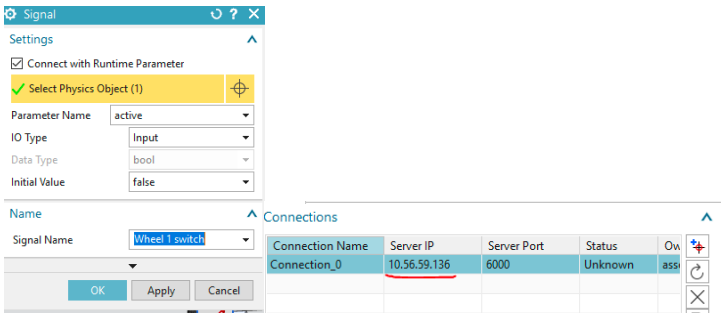
The first step was to add a speed control to both wheels. This allows the user to control the speed of the wheels.

The next step was to add an inclinometer. In order to do this, I had to make sure that rigid body for 'MyRio' had the centred of mass correctly. Then, I can select the 'MyRio' as my inclinometer.

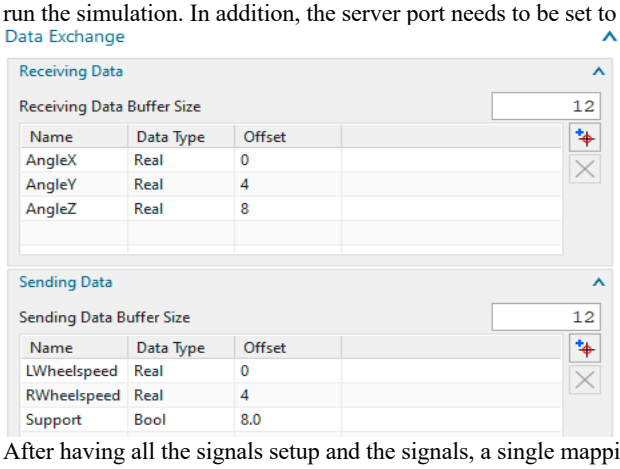


Signal and signal connection

Once the inclinometer is created, the signals for the wheels need to be created and connected to the runtime parameter so it can be controlled.

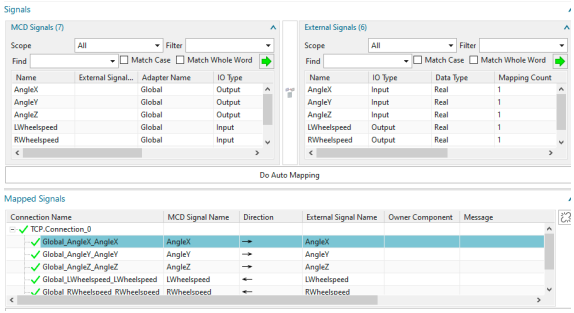


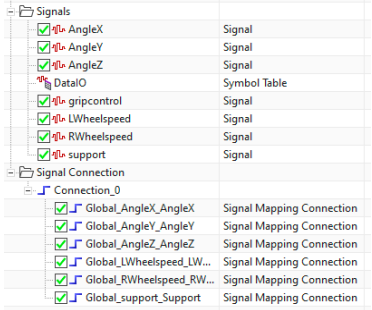
After adjusting the signal. The IP address needs to be changed every time is used, otherwise the software will not be able to run the simulation. In addition, the server port needs to be set to 6000. This can be seen underlined in the image above.



After adjusting everything, the next phase is to add the receiving data name and its data type. For the sending data we write down the speed controllers data type as real and for the support it would be Boolean.
Both, the receiving and sending data have a total value of 12 bytes.

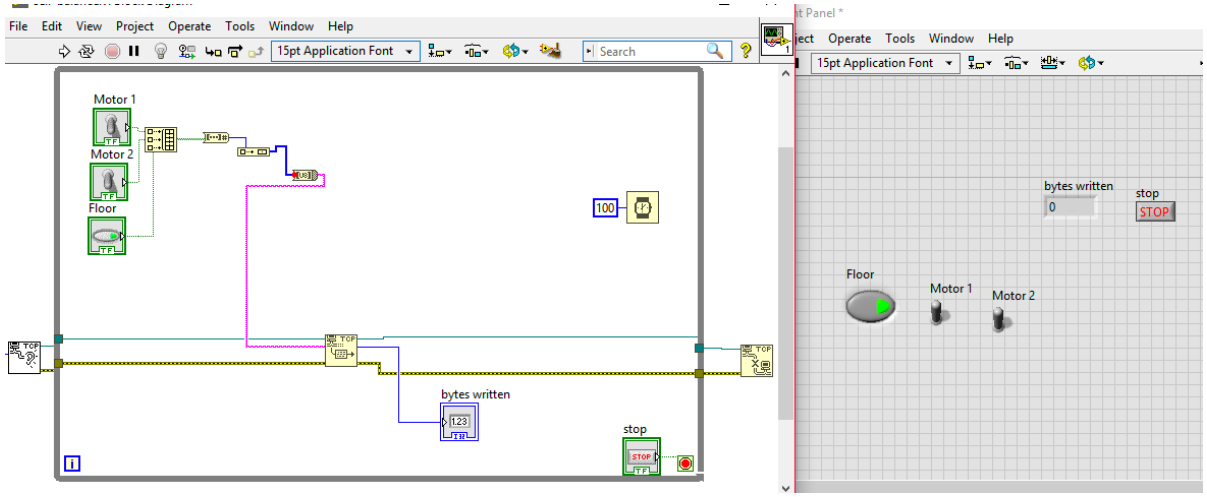
After having all the signals setup and the signals, a single mapping connection will be required in order to connect the software LabVIEW with NX.



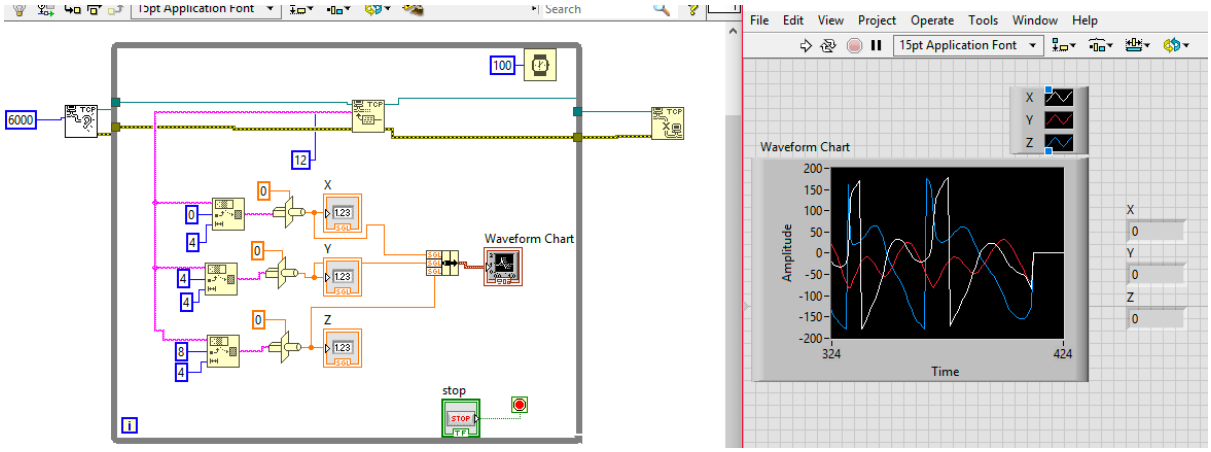


Once all the connections are ready, the process to balance the robot will initialised.

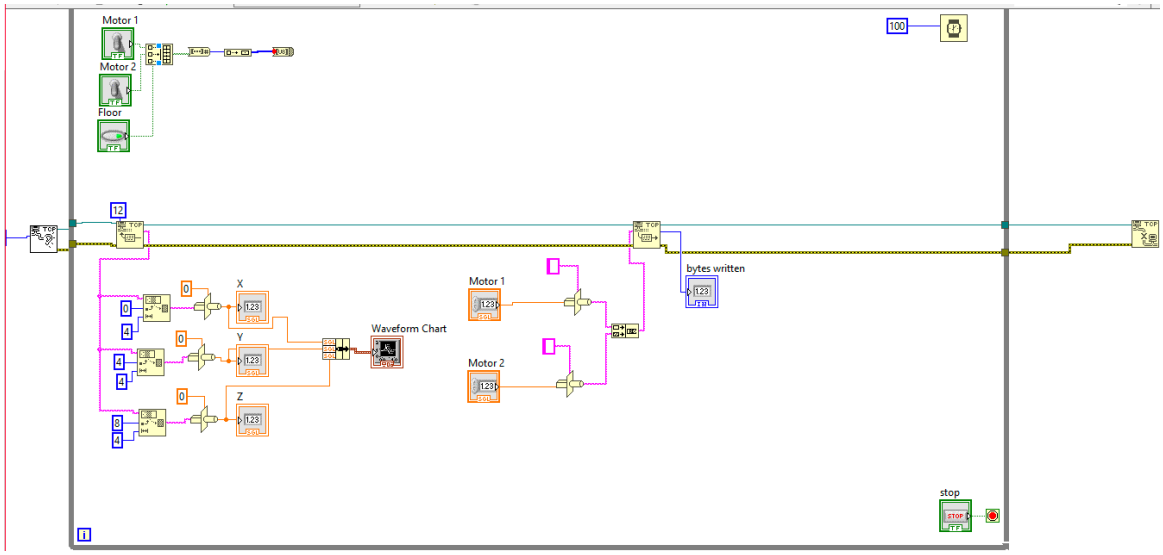
LabVIEW Code



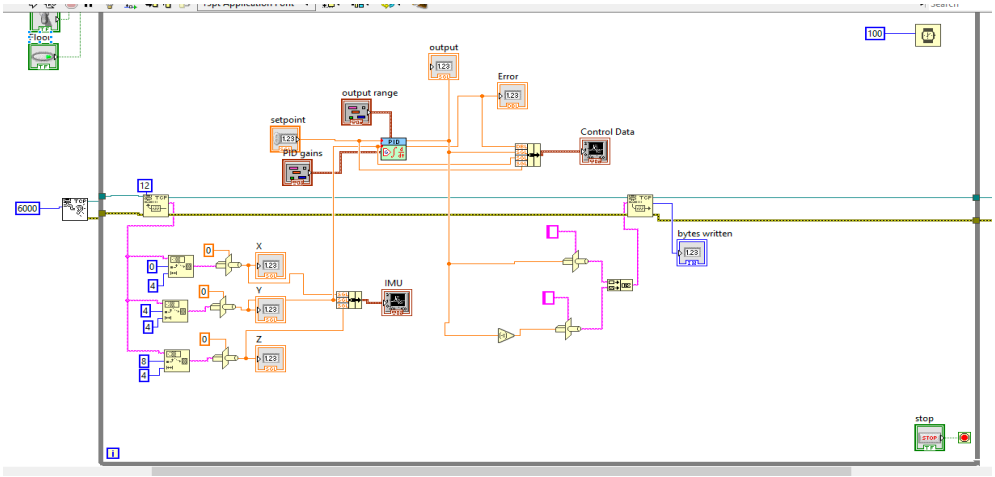
This is what the first code looks like. The function for this code is to turn on and off both motors and control the collision body of the floor. The way it works is by sending data to the TCP using an array that is connected to a converter from Boolean array to number and from an array to a string so the server could understand the code.



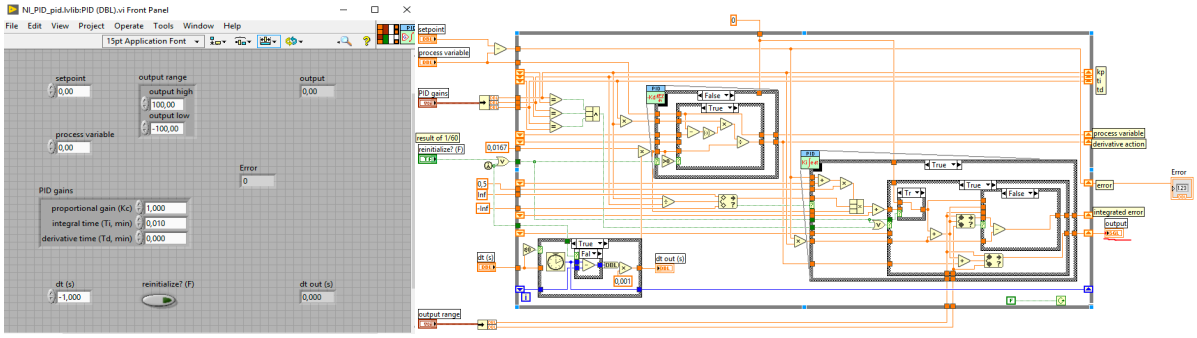
The next step is to create a waveform chart in order to see the location of the robot and how to receive data from NX. As you can see, it demonstrates the robot's position. The idea of this code is that the TCP reads the data, and it converts it from a string to a number that is later displayed in the waveform chart. The data representation in LabVIEW hast to be an SGL because we want to read 1 byte instead of 2 at the same time, like DBL does.



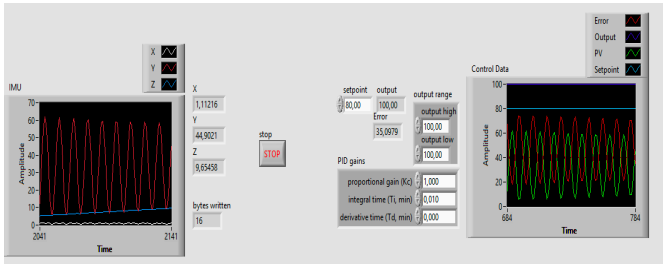
This is what the code should look like if we combine both of the previous codes. In order to manipulate manually the robot speed for each motor, I had to add a TCP that send data so I can manipulate the speed manually.



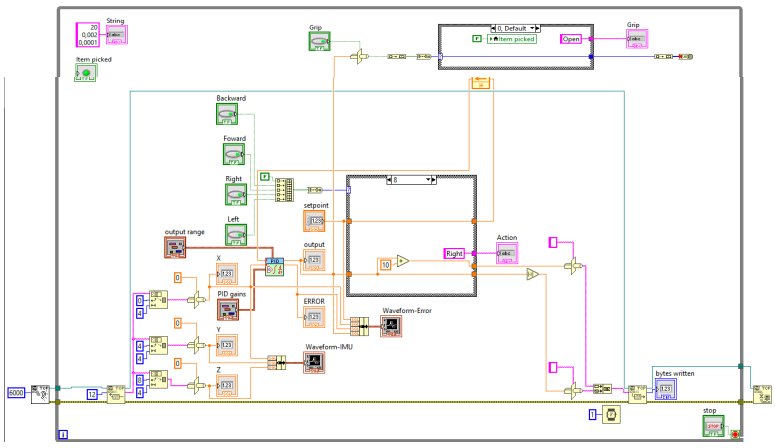
Furthermore, the program will require a PID controller in order to balance the robot. Therefore, the data is added to the requirements for the PID. However, the error indicator was missing so I had to access the PID controller and change some values.



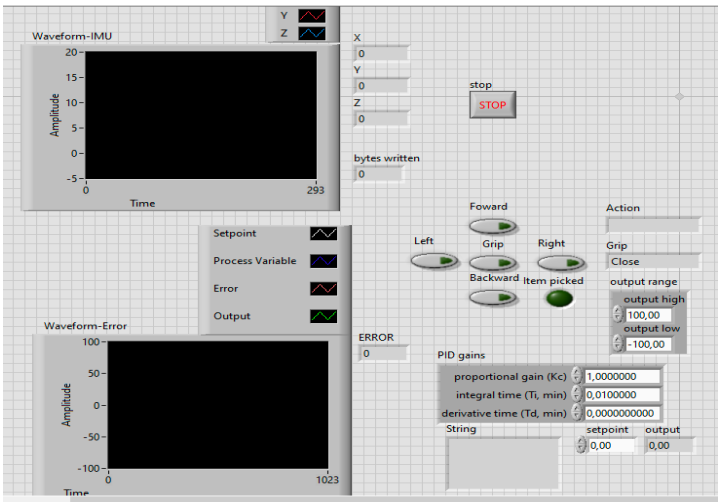
First, I had to change the output DBL to s single precision as shown in the image above underlined in red. This is an important step because if I do not change the signal to single precision, it is going to send it as double precision and it will cause an error in NX, making the robot to fall down. Then we create an indicator for the error section and we connect it to the PID.



The program should resemble this, having two waveform charts, one for the IMU XYZ values, the other one to adjusts the PID values. Now I need to find PID values for my robot. This can be done by trial and error, following the method done in the research. Starting by looking for the Kc value, then the Td value and finally if needed the Ti value. The values ended being 20 for the proportional gain, 0,002 for the integral time an 0,0001 for the derivative time.

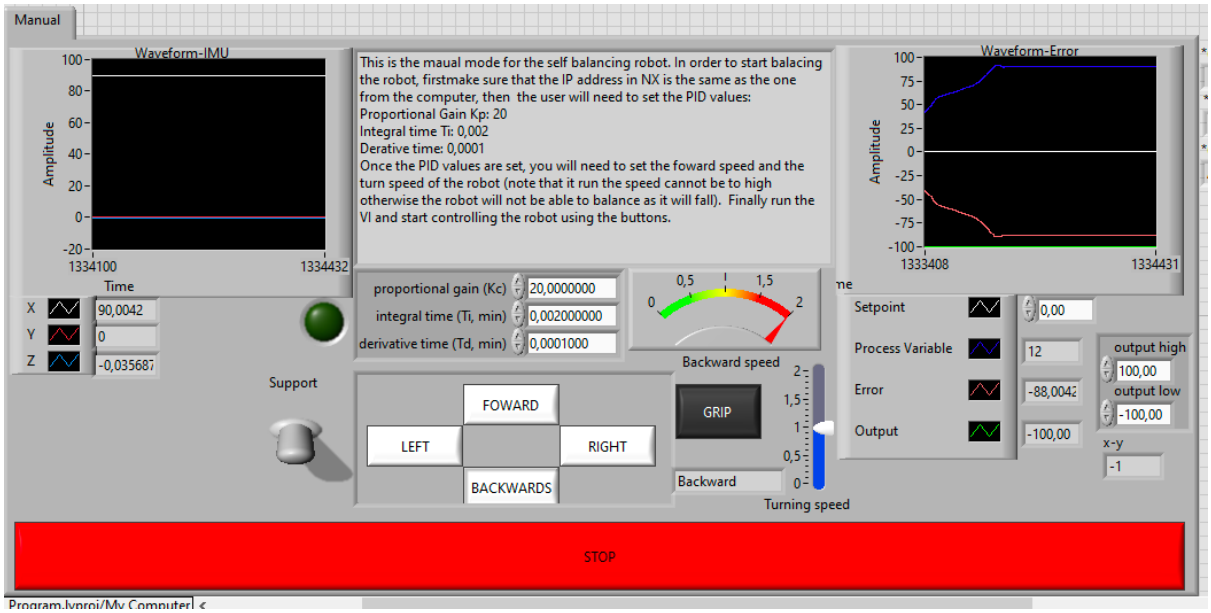


Once the robot balances, we can add a case structure to add controls for the movement such as moving forward or backward or even turning left or right and a button for the grip for further development. The case structure uses binary in order to access each stage of its stages. For example, in the image above, the right option is set to the number 8 as it is the 4th option, first one would be stationary at 0 bit, then backwards at 2 bits, then forward at 4 bits, then right at 8 bits and finally left at 16 bits.

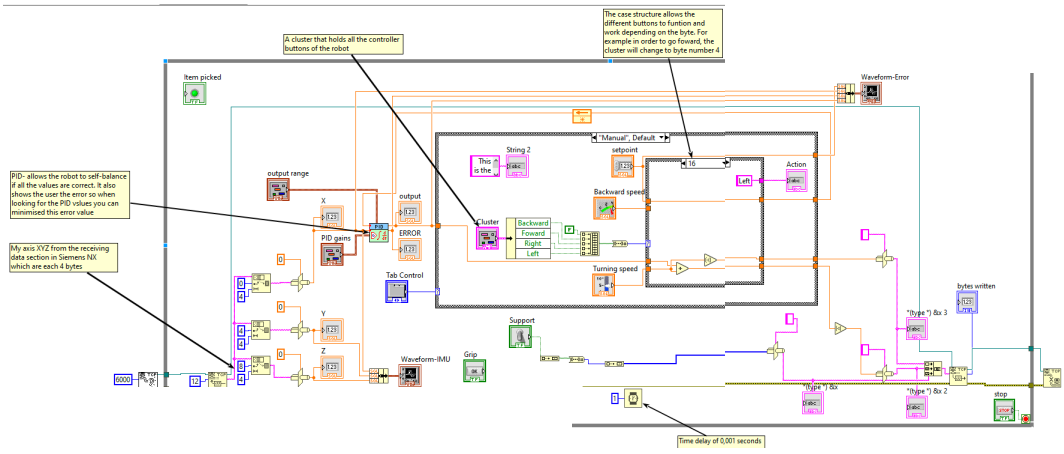


This is what the code should look like to the user when using the program in order to balance the robot.

Final LabVIEW code

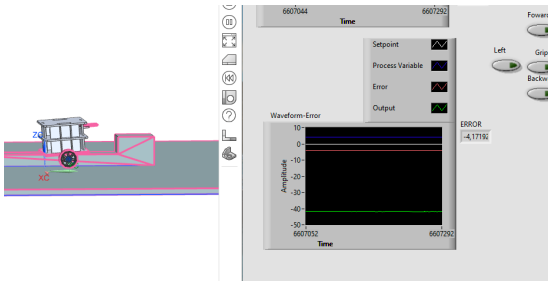


This is my final code in LabVIEW. I rearranged everything to make it more organised. In addition, I added a description to explain how to use the software.

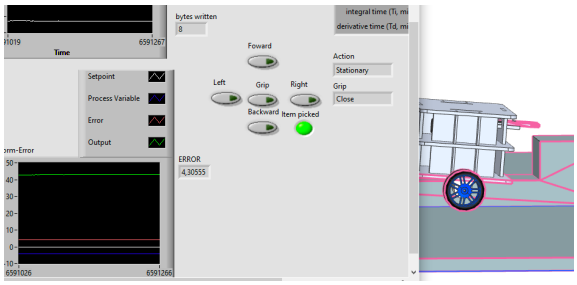


PID Testing

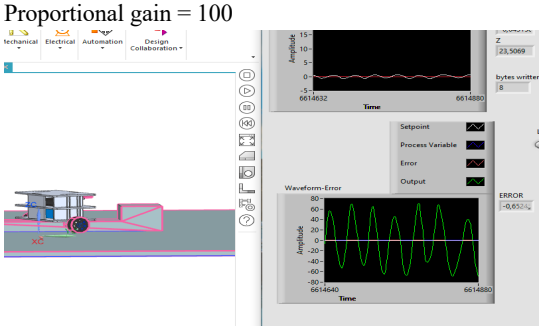
Robot tilted to the left, proportional gain = 10



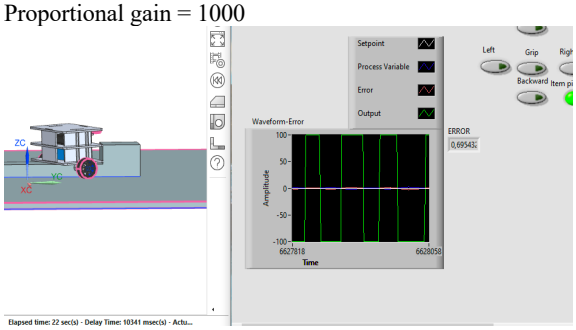
Robot tilted to the right, proportional gain = 10



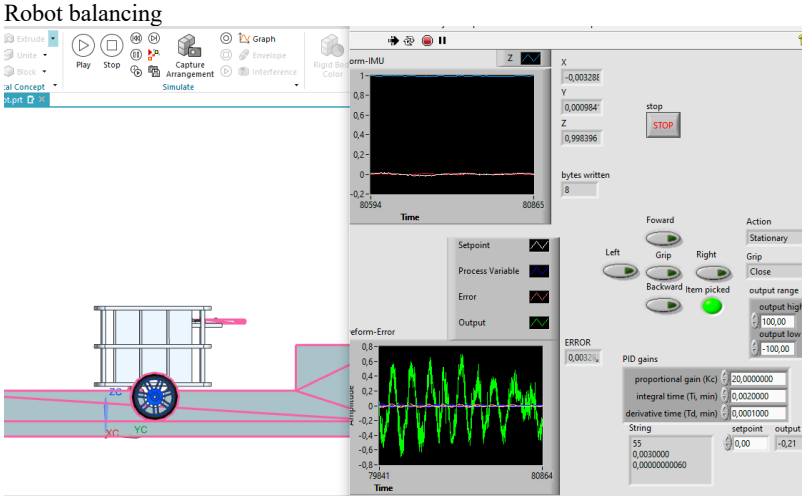
Proportional gain = 100

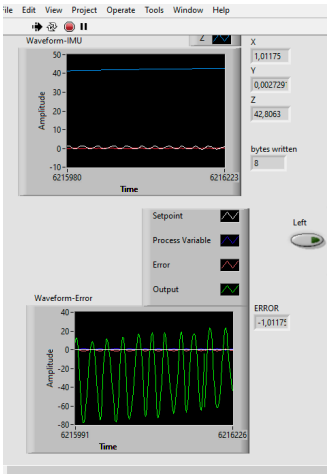


Proportional gain = 1000



Robot balancing

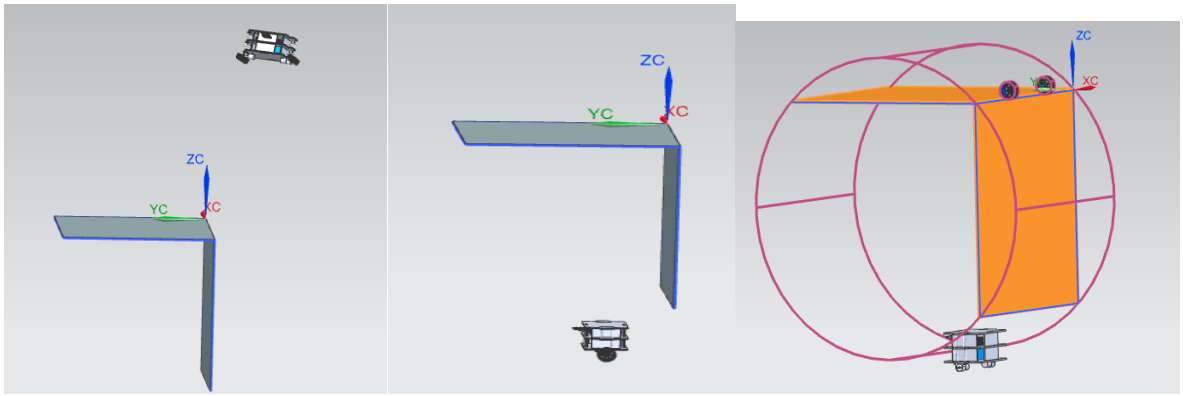




The aim for the PID controller was to find the optimal proportional, derivative, and integral values for a robot to achieve a stable balance. After testing various proportional gains values, I observed that values closer to 10 resulted in improved balancing. However, values above 10 caused the robot to move quickly to the left and right, exhibiting an obvious instability. Therefore, I narrowed down the range of values to above and below 10. When testing values below 10 I concluded that it led to no oscillation, suggesting that the value I was looking was higher than 10. After increasing the values 1 by 1, I determined the optimal proportional gain value to be 20.
After selecting the proportional gain, I incorporated derivative time to minimize oscillations when balancing, which resulted in a value of 0.0001. Finally, I added integral time to prevent the robot from falling forward and found a value of 0.02 to be optimal. By combining all three values, the robot achieved a perfectly balanced motion.
This demonstrated the importance of finding the optimal values of proportional, derivative, and integral gains for robots to perform their intended functions accurately and effectively.

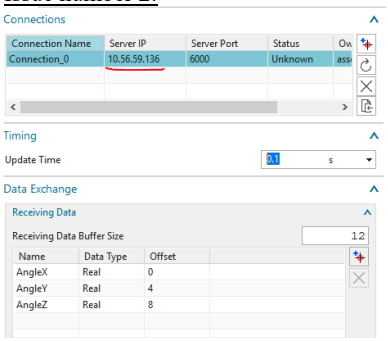
Issues encountered

Issue number 1:
When running the simulation, I found out that the robot does not lay on the floor. Instead, it jumps around and flips the wheels. On the other hand, if I deactivate the collision for the wheels, the robot goes through the floor.



Solution to the issue:
Therefore, I found out that I accidentally put a sphere shape on the floor as a collision body instead of a cube one. This is why the robot was jumping around.

Issue number 2:



I did not know how to connect back both software, NX and LabView, after closing them and opening again. I found out that every time I connect to the university server through the VPN, I needed to clarify to NX that the IP changed and write the new one.

Issue number 3:
When running the LabVIEW code with NX, I had an issue where everything in NX would disappear. This was due to having 2 different READ TCP in LabVIEW instead of one and the software tried to read both at the same time, interfering with the signals. The solution for this was to use only one READ TCP and connect the motors and the grip there.

Conclusion

The design process employed for this project was critical in achieving the desired specifications for the self-balancing robot. After successfully completing the project, I found the experience to be both interesting and enjoyable. The self-balancing robot that I designed has the ability to maintain its balance while being manually controlled, demonstrating its success in meeting its design objective. Users can adjust its speed, PID values, and control the support's collision body, making it a versatile and useful tool.

Throughout the project, I referred to a well-designed project plan that helped me stay organized and meet my deadlines. The plan included tasks that were color-coded by importance, allowing me to prioritize my workload and complete tasks in a timely manner. I was able to meet the set deadlines due to good time management and organisation. This approach was particularly helpful when designing a well-structured two-wheel self-balancing robot. Within my plan, I conducted extensive research on other self-balancing robots, sensors, and actuators, with a focus on how other designs achieved the correct PID values and balance their robot. This research helped me design five different ideas for the robot's structure, ultimately settling on the most suitable and reliable final drawing as well as getting a better understanding of how PID control functions. Learning on how PID works was crucial as it helped me later in the project when balancing the robot. When creating the CAD model for the robot, I referred to my original design, which helped me visualize what I wanted the robot to look like.

The process of balancing the robot was not without its difficulties, particularly in relation to the inability of the robot's TCP to connect to NX while attempting to grasp objects highlighted a significant limitation as it interfered with the speed controllers. To address this issue, future revisions would need to be focused on. Generally, I believe that the final device met the desired specifications. It is a versatile and reliable self-balancing robot that can be manually controlled, with adjustable speed and PID values. However, if I were to take the project further, there are several changes that I would make. Firstly, I

would enhance the robot's structure with more detailed components, such as screws and bolts, to increase its overall strength and durability and improve its overall stability and reliability. Secondly, I would further develop on the proximity sensors as it would need to be further developed to detect walls and obstacles more accurately, improve its ability to navigate its environment more effectively. Thirdly, I would explore the possibility of enabling the robot to move autonomously, as this would increase its utility and make it more versatile. Finally, if feasible, I would consider scaling up the robot's size to increase its capacity to handle larger and heavier items.

In conclusion, this project was a great learning experience as it provided a valuable opportunity to gain hands-on experience in digital robot design, and to identify areas for future improvement. I was able to successfully design and build a self-balancing robot that met the desired specifications, and I feel confident that with further development, it could become an even more versatile and useful tool. I am grateful for the opportunity to work on this project and look forward to applying the skills and knowledge that I have gained in future projects.

References

[1] - MR Innovative, "Diy self-balancing robot arduino based," YouTube video, 16:44, September 13th, 2019, https://www.youtube.com/watch?v=aUbBUd-hBLI. [Accessed: January 14th, 2023]

[2] - Maker Tutor, "Ultimate Guide to Make Self Balancing Robot for Beginners – ICStation.com," YouTube video, 7:53, December 21st, 2018, https://www.youtube.com/watch?v=CON0sWNDUco&t=3s. [Accessed: January 14th, 2023]

[3] - Electrical Projects [CreativeLab], "How to Make DIY Balancing Robot [My Experience]," YouTube video, 11:14, July 11th, 2020, https://www.youtube.com/watch?v=vPx2oyZGv50&t=531s. [Accessed: January 14th, 2023]

[4] - YedaCenter, "Rack & Pinion Gripper Mechanism," YouTube video, 2:01, May 11th, 2019, https://www.youtube.com/watch?v=JR80erVhxE4. [Accessed: January 16th, 2023]

[5] - Tang, Zhijie & Lu, Jiaqi & Wang, Zhen & Ma, Gaoqian. (2019). The development of a new variable stiffness soft gripper. International Journal of Advanced Robotic Systems. 16. 172988141987982. 10.1177/1729881419879824. [Accessed: January 16th, 2023]

[6] - Yagoadan, "Collection of projects," Padlet, August, 2021. [Online]. Available: https://padlet.com/yagoadan/collection-of-projects-ex8dvnozyurg/wish/2341326176. [Accessed: January 16th, 2023]

[7] - Brian Douglas, "PID Control – A brief introduction," YouTube video, 7:43, December 14th, 2012, https://www.youtube.com/watch?v=UR0hOmjaHp0&t=324s.