ENS 492 – Graduation Project (Implementation)

Progress Report II

Project Title: A Self-Following Luggage

Group Members:

Muhammad Mustafa Akhtar

Abbas Ali Hakeem

M Safwan Yasin

Supervisor(s):

Melih Turkseven

Erchan Aptoula

Date: 31.03.2024

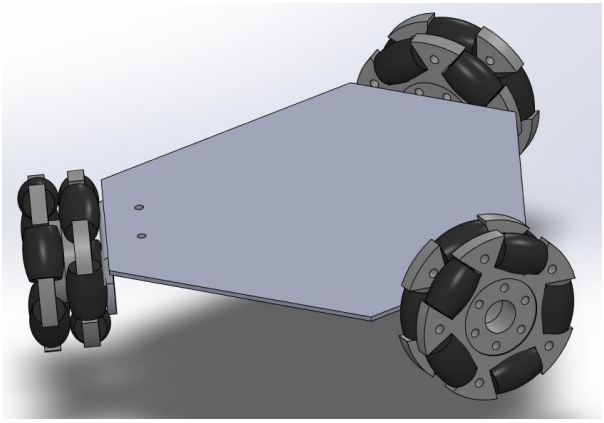
1. PROJECT SUMMARY

The project focuses on developing a prototype for a smart luggage platform that autonomously follows its owner. This is achieved through a combination of computer vision algorithms, stereo cameras, and a mechatronic system. The objective is to create a cost-effective and accessible solution to transform ordinary luggage into smart luggage with AI capabilities.

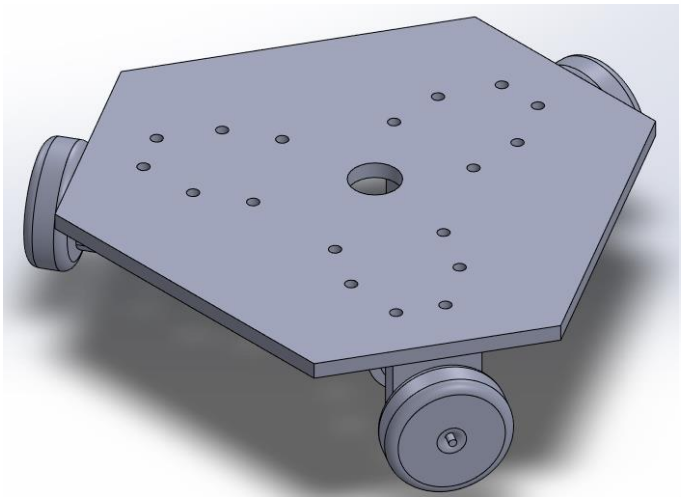
2. SCIENTIFIC/TECHNICAL DEVELOPMENTS

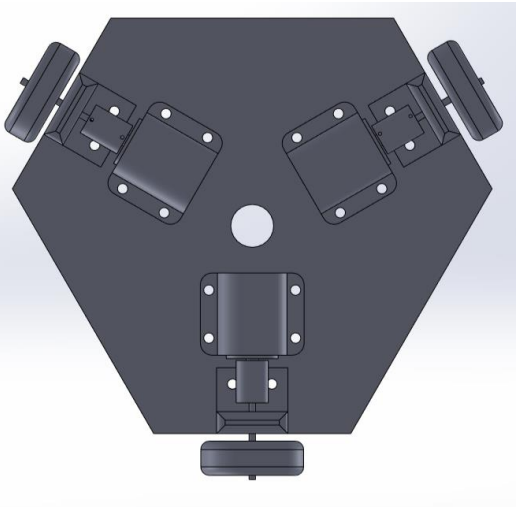
Mechanical Design

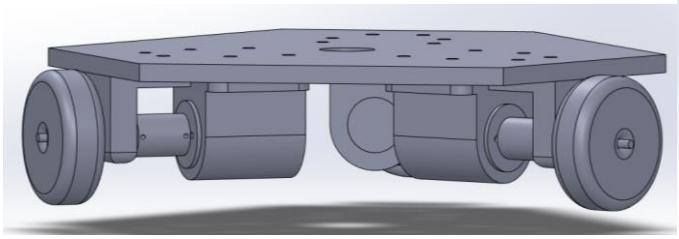
Initial Design :

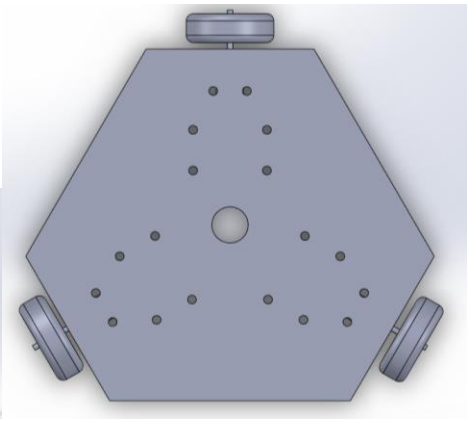


Detailed CAD Model design:

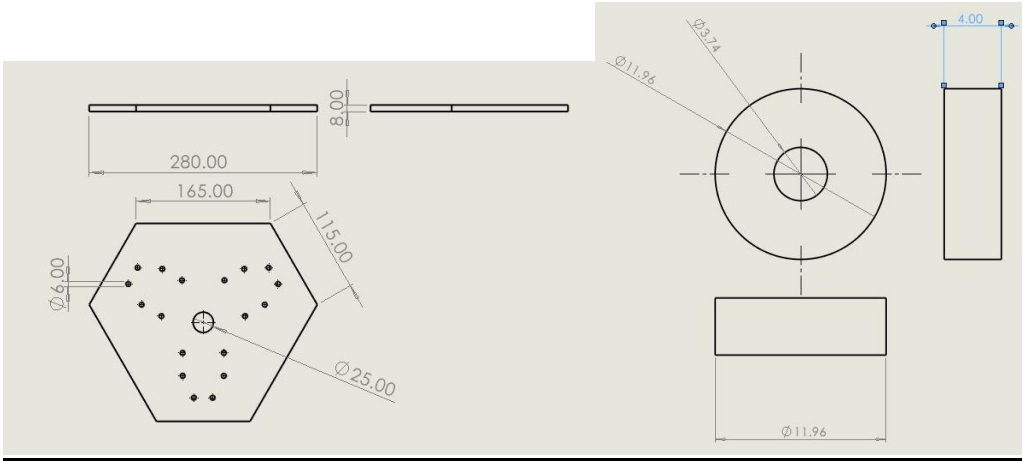


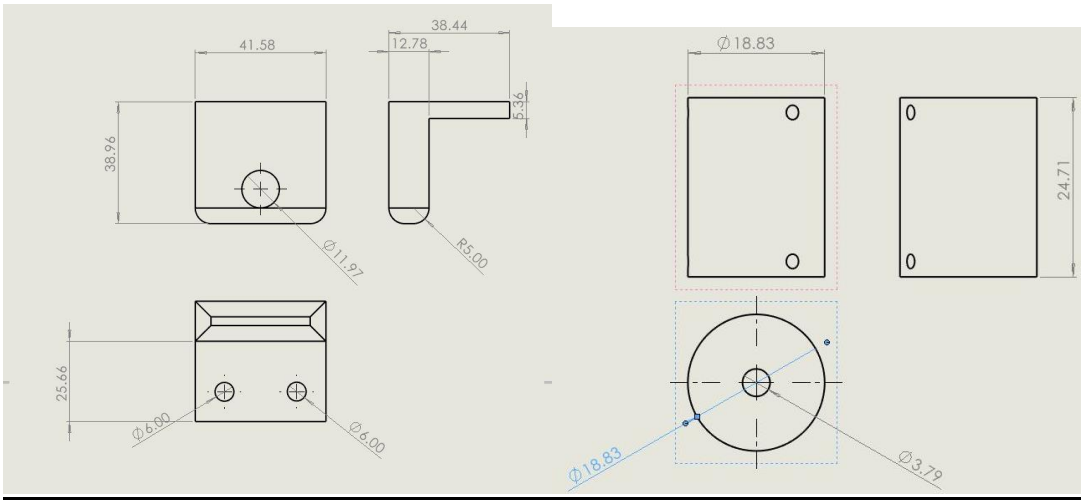


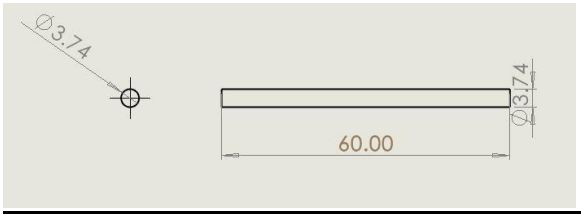


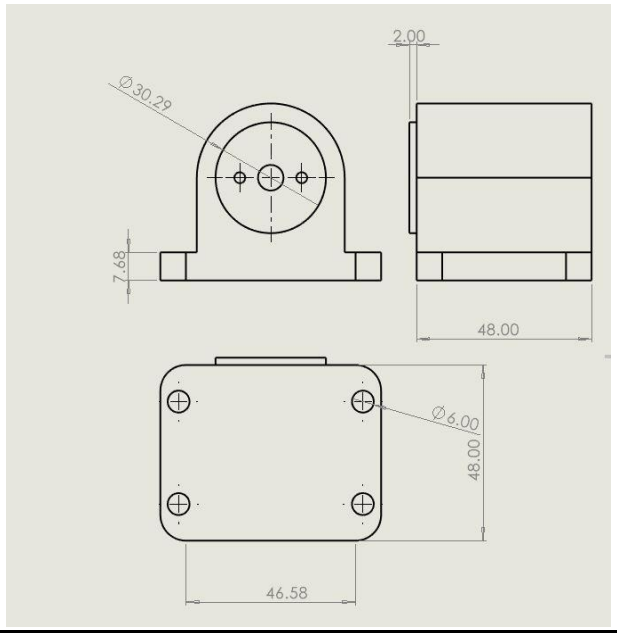


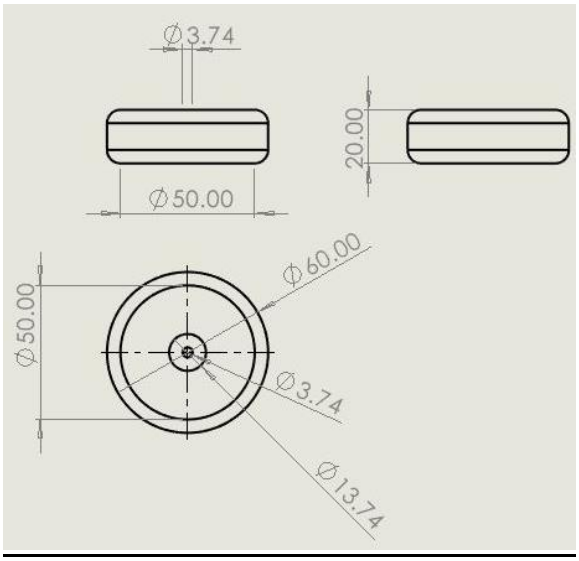
Technical Drawing of the Parts:











Key design Descriptions:

Base Plate material: Plexi-Glass with 8mm thickness

Plexi-Glass was the second best option as compared to the aluminum extrusion profiles due to its strength and ability to be cut into a suitable shape.

A thickness of 8mm allows the platform to be able to withstand the weight of the components and the load that would be placed on top. Furthermore,

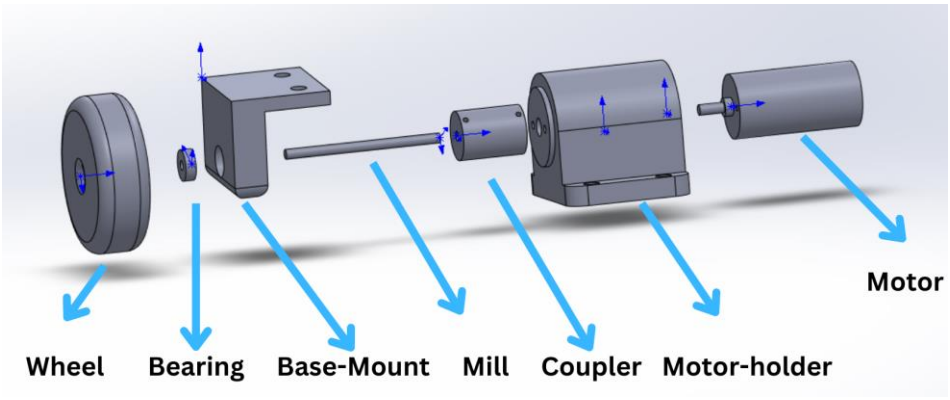
the chosen thickness is optimal for screwing the components atop the platform.

Omni Wheels: 60 mm Diameter

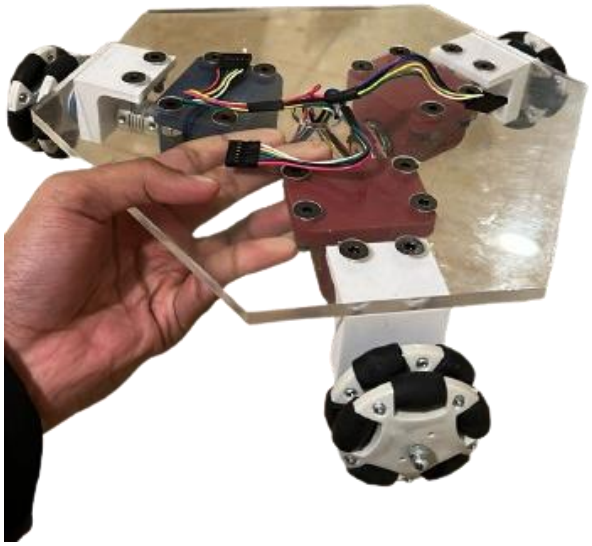
The holes in the Base are 6mm and M6 Countersunk screws are used with nuts for mounting the mount and the motor holder to the base.

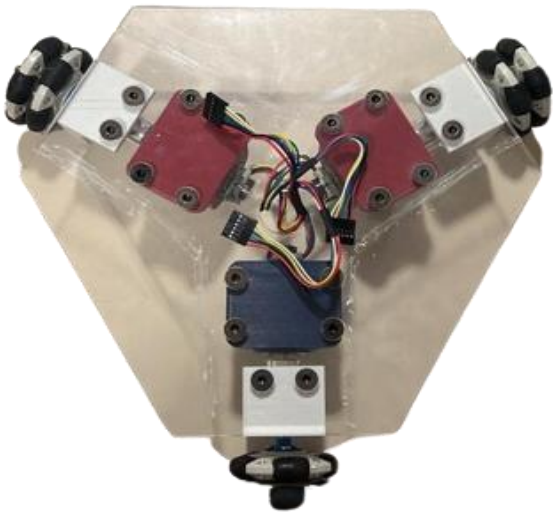
The Motor-holder was designed according to the specifications of the motor.(Refer to the technical drawing)

Exploded View of the DriveTrain:



The next step involved 3D printing the parts from the CAD model such as the Motor Holders, and the base Mounts followed by the assembly of all the components.





Electronics:

The **Raspberry Pi 4** was chosen as the primary computing device due to the ease of connectivity with the required hardware together with the Intel RealSense D435i which would be used as the primary sensor. Moreover, this single board computer allows for the execution of Python scripts - the primary programming language being used for this project - and provides sufficient computing power to run the computer vision algorithms. The SSH feature of the Pi would be used to access it wirelessly.

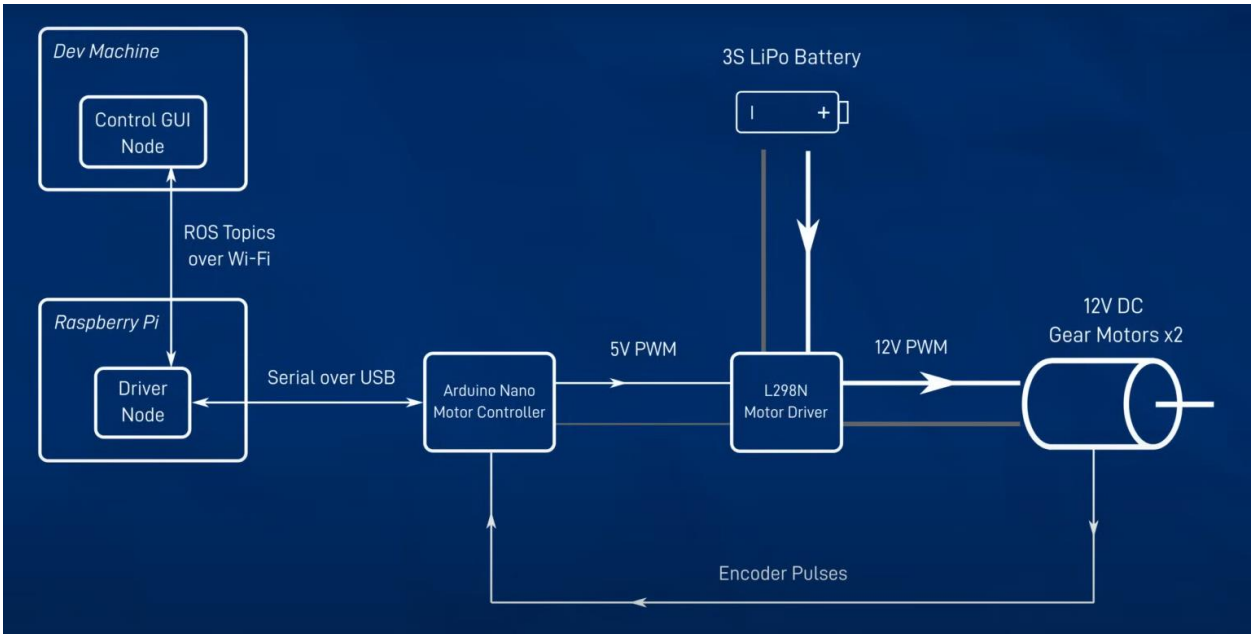
The **Intel RealSense D435i** would be used as the primary sensor for the system. This camera was primarily chosen due to its stereoscopic capability and depth perception. The camera contains a depth module that enables the creation of a depth map which would be used to measure the distances between the robot and potential obstructions in its immediate surroundings. A RGB module is also present that would be used to identify obstructions and create bounding boxes using machine learning algorithms. Furthermore, the camera comes with its own Python library

which streamlines the process of creating a depth map and applying filters.

For the Motor-Driver 2 - **L298N Motor Drivers** were selected to control the 3 motors. They provide the required current output for the motors along with the ability to control the motors with PWM pins provided inside of this motor-holder.

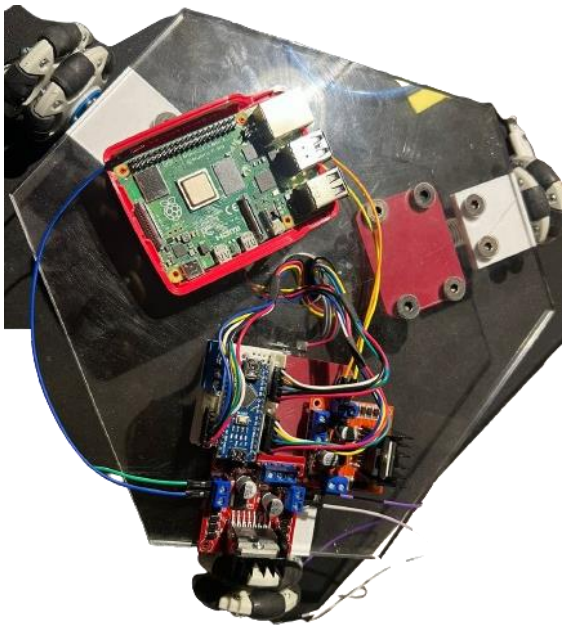
In order to ensure that the Raspberry Pi has sufficient computing capability to execute the algorithms, the **Arduino Nano** would be used to handle the motor control.

We plan on following the following diagram when we will wire the components together:



(Newans, *The Ultimate Guide to using motors in robotics (including ROS, Raspberry Pi)* 2022)

A power bank would be used to power the components along with buck-boost converters to deliver the power to the motors, the raspberry pi and the motor controllers. Further research is required to implement this system.



The only component that needs to be implemented is a mount for the camera.

Dynamic Modeling:

The diagram below shows the simplified aerial view of the robot. The robot legs are equidistant by 120 degrees. The $\theta_{1,2,3}$ are 90, 210 and 330 degrees respectively. The end goal pose and current pose was used to determine the acceleration required by the robot to move from current pose to the end goal pose. The force required in each wheel would be used to get the required linear and rotational acceleration for the robot to move.

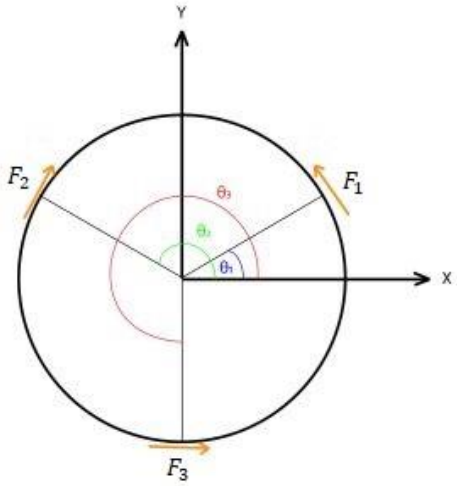


Fig 1.

The equations 1-3 show the use of the concept of equivalent systems to translate the forces and moments from the wheels to the center of the robot.

$$\Sigma F_x = F_{x,W=1} + F_{x,W=2} + F_{x,W=3} \qquad \text{Eq 1}$$

$$\Sigma F_y = F_{y,W=1} + F_{y,W=2} + F_{y,W=3} \qquad \text{Eq 2}$$

$$\tau_{R \to W=1,2,3} = F_W \times d_{radius\ of\ robot} \qquad \text{Eq 3}$$

The chassis of the robot was initially modeled as a circular disc and to simplify calculation, the following equation was used for the moment of inertia of the robot.

$$I = \frac{1}{2} M_R\, d^2 \qquad \text{Eq 4}$$

After assuming the shape of the chassis as a disc and applying Newton's 2nd law of motion, the following equations were derived for the relationship between the equivalent forces and torques applied on the robot and robot's linear and angular accelerations.

$$\alpha_{R\to W=1,2,3} = \Sigma \frac{\tau_W}{I} \qquad \text{Eq 5}$$

$$\Sigma F_{Rx} = M_R a_x \qquad \text{Eq 6}$$

$$\Sigma F_{Ry} = M_R a_y \qquad \text{Eq 7}$$

Controller Design:

A PID controller will be used to control the speed of the individual wheels to get to the desired robot pose. The main advantage of using this control technique is that it can be tweaked using P,I and D gains to get desired outputs through trial and error. The PID controller uses the error calculated from current pose and the desired pose. This error is then used in the following equation to get a desired signal which is added on top of the current signals to make sure that the error goes to zero.

$$u(t) = K_P e(t) + K_I \int_0^t e(t)dt + K_D \frac{d}{dt} e(t) \qquad \text{Eq 8}$$

K_P is the proportional gain which makes the system reduce error faster. If the required output was a certain value this gain multiplies with the output to increase the output to a greater value making sure that the desired pose is reached. The K_D gain helps in maintaining stability to the system, any vibrations would be smoothed out. The K_I gain helps in eliminating the errors which may be constant over time. Generally the K_I gain is kept very low since it may cause the system to reach instability since the constant errors keep getting added at the output of the system after each interval.

In our design the goal pose would be determined by the computer vision algorithms. The goal pose would consist of the distance and the angle to which the target moves. The figure below shows how the controller would get the desired target pose using PID and feedback loop.

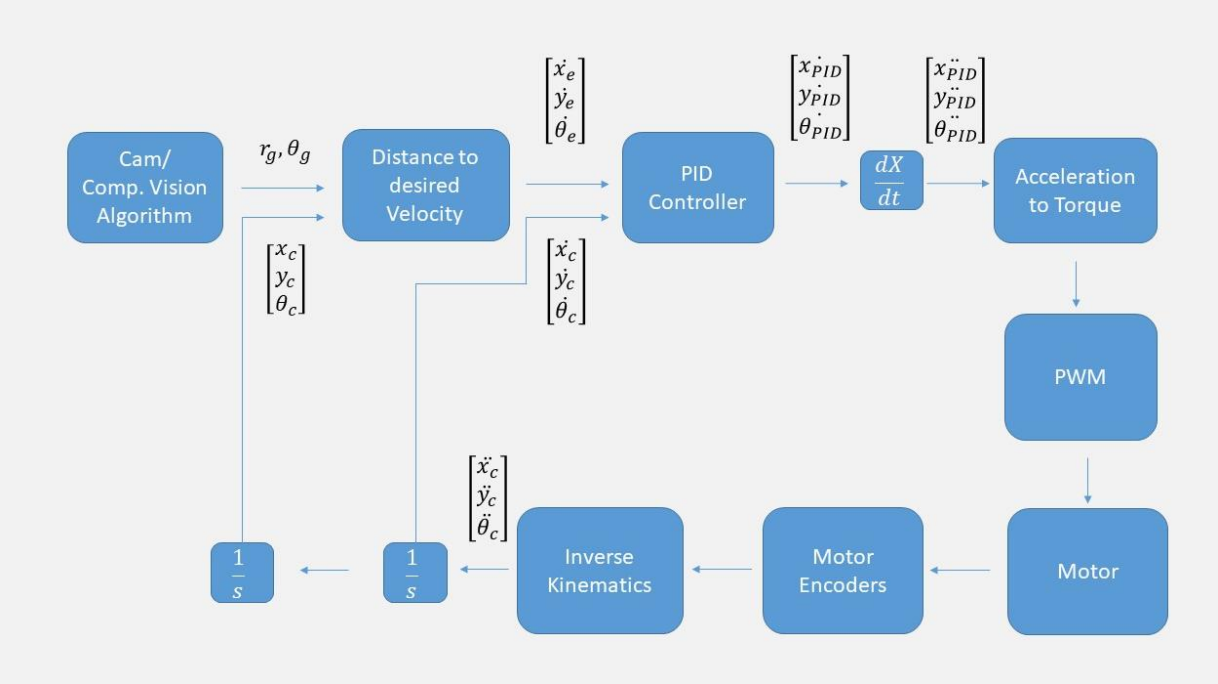


Fig 2.

Software

The programming language primarily used to create the software is Python. OpenCV, Pyrealsense, and numPy were the main libraries used.

The flow of the software is the following:

1. Firstly, a depth frame and a RGB frame is captured by the camera. In order to denoise the input, appropriate filters are applied to the depth map which include a decimation, spatial, and temporal filter.
2. Secondly, a central point is calculated within the captured frames along with tolerance limits. For the robot to be able to follow the person accurately, the person must be within the tolerance limits that would be defined as points on the x-axis. In order to check if the subject is within the limits, a machine learning model is used to recognize the person and create a bounding box around them. Then, it would be checked whether the bounding box is within the tolerance limits. There would be two possibilities in this scenario:
    a. The person is within the tolerance limit. In this case, the robot would keep moving in the same direction unless prompted to stop.
    b. The person is outside the tolerance limit. In this case, distance between the center point of the frame and the

person would be calculated. Using that, further calculations would be performed and the robot would be rotated to realign the person within the tolerance limits. The process would be repeated until the person is within the limit.

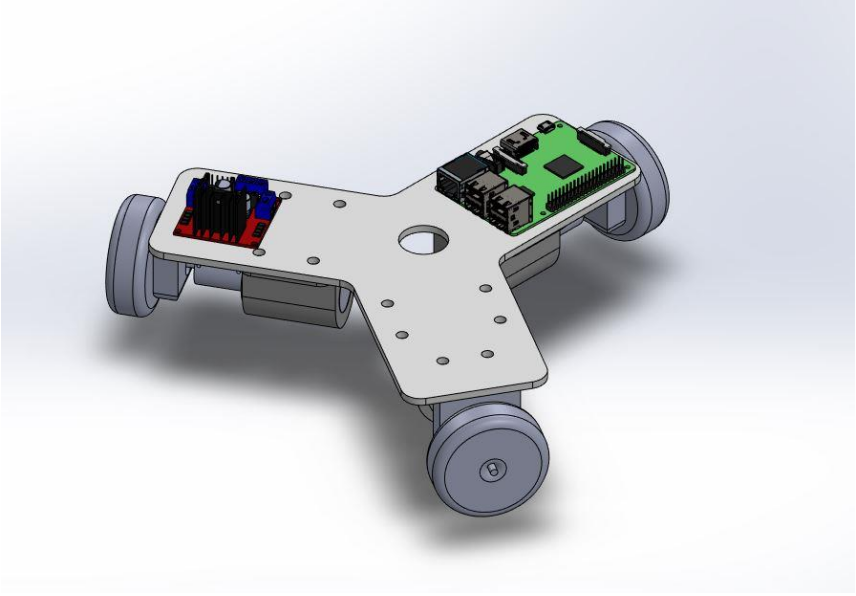
In case the person is not found within the field of view of the camera, the robot would rotate in order to locate the person. If not found, an alert would be made to caution the person.

3. Using the depth map, the distance between the person and the robot would be calculated and the robot would be moved an adequate distance. In case the distance between the robot and the person is below a certain threshold, the robot would be prompted to stop.

3. ENCOUNTERED PROBLEMS

1) Adjustment of the weight of the robot

In order to maximize operational efficiency, it is recommended to minimize the weight of the robot. Therefore, a new design was created that focuses on reducing the material at the base and provides enough space and sturdiness to hold all the electronic components onto it. Removing some of the plexiglass along the edges in between the wheels resulted in a significant reduction in the weight of the robot.



2) Use of the power bank while running the platform to make it autonomous

While the system is currently being tested with a cable attached to the platform for power we aim to make it completely autonomous. To solve this issue, the power would be derived from the power bank and the components would be powered independent of each other. A system needs to be derived in order to bring this into action.

3) Changing the computing device and camera

One of the main changes made was to change the choice of hardware. The initial plan involved using a Raspberry Pi Compute module along with stereoPi. Two RGB cameras were to be mounted on the system. There were two main issues with this selection:

1. The computing power of the Compute module is significantly lower compared to the Raspberry Pi 4 which could cause several problems while running the algorithms.
2. Using RGB cameras for distance calculation would have been inaccurate and computationally expensive. Using the Intel RealSense camera allows for accurate distance calculation by creating a depth map along with a simplified way of applying filters to the frames in order to denoise them.

The sub-tasks that were originally planned were finished within the allotted time period. There were a few periodic delays that detoured the progress of the project from the timeline such as the change in hardware mentioned above. However, the benefits of shifting to the new hardware, particularly the camera, expedited the pace of the project since tasks such as image processing and deriving formulas to calculate distance were no longer required.

4. TASKS TO BE COMPLETED BEFORE FINAL REPORT

The main tasks to be completed before the final report is to assemble different parts of the design namely mechanical, electrical, controls and pair it together with the software in order to test the entire system. One of the other tasks that needs to be done is to create a power circuit which will take current and voltage from a power-bank and accordingly give the required current to the motors and the Raspberry Pi. Furthermore, dynamic modeling also needs to be translated into Python code. Lastly, further optimization of the algorithm needs to be done and more scenarios - particularly involving obstruction detection - that the robot

might encounter need to be factored in order to maximize the functionality of the system.

5. REFERENCES

Newans, J. (2022). The Ultimate Guide to using motors in robotics (including ROS, Raspberry Pi). Retrieved from https://www.youtube.com/watch?v=-PCuDnpgiew