System Design (Draft)
MECHTRON 4TB6

Group 1, UWheeledChair,
Lisa Ji
Haoyu Lin
Yuntian Wang
Zichun Yan

December 25, 2023

Table 1: Revision History

Date	**Developer(s)**	**Change**
2023-12-18	Lisa Ji, Haoyu Lin Yuntian Wang, Zichun Yan	Revision 0: Draft

Contents

List of Tables

List of Figures

1 Introduction

This document provides the system design for the UWheeledChair project of Group 1.

1.1 Project Purpose

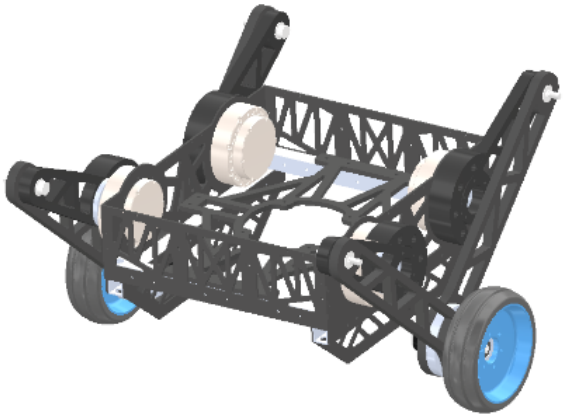

Figure 1: WBR Mechanical Platform

This project is to develop the software and control system for a fully-autonomous delivery robot, based on the existing assembly of a Wheeled Bipedal Robot (WBR) provided by the MacRobomaster Club (MacRM), as shown in Figure 1. The project will be referred to as the WBR project in the following documents.

For MacRM, WBR was constructed following the constraints defined in the rules, Committee (2023), of the 2024 RoboMaster University League (RMUL) Competition, whose host is SZ DJI Technology Co., Ltd. (DJI). Details of the constraints are shown in SRS (see Ji et al., 2023, Design Constraints). The hardware system is built under the competition constraints, that the robot shall be non-holonomic, able to balance itself on two wheels, and able to jump across obstacles. Here we are adapting it to our delivery robot project, and as the additional constraints for a delivery robot, it shall be able to plan and follow the delivery route, as well as automatically avoid obstacles on the way.

1.2 Document Purpose

The purpose of this document is to show the overall design of a system which meets the requirements stated in the SRS (Ji et al., 2023). This includes the development team's decisions and considerations for system components, and also how we adapted the mechanical model (WBR) built according to RoboMaster University League (RMUL) rules to our delivery robot project.

1.3 Project Scope

Since the mechanical and electronic hardware are fixed constraints, the WBR project mainly focuses on the software and control system, while MacRM is responsible for the mechanical design, provision, and maintenance.

1.4 Organization of Document

The following document will outline the overall system components, communication protocols, overall system behavior, operation and undesired error handling.

2 Project Overview

2.1 System Context

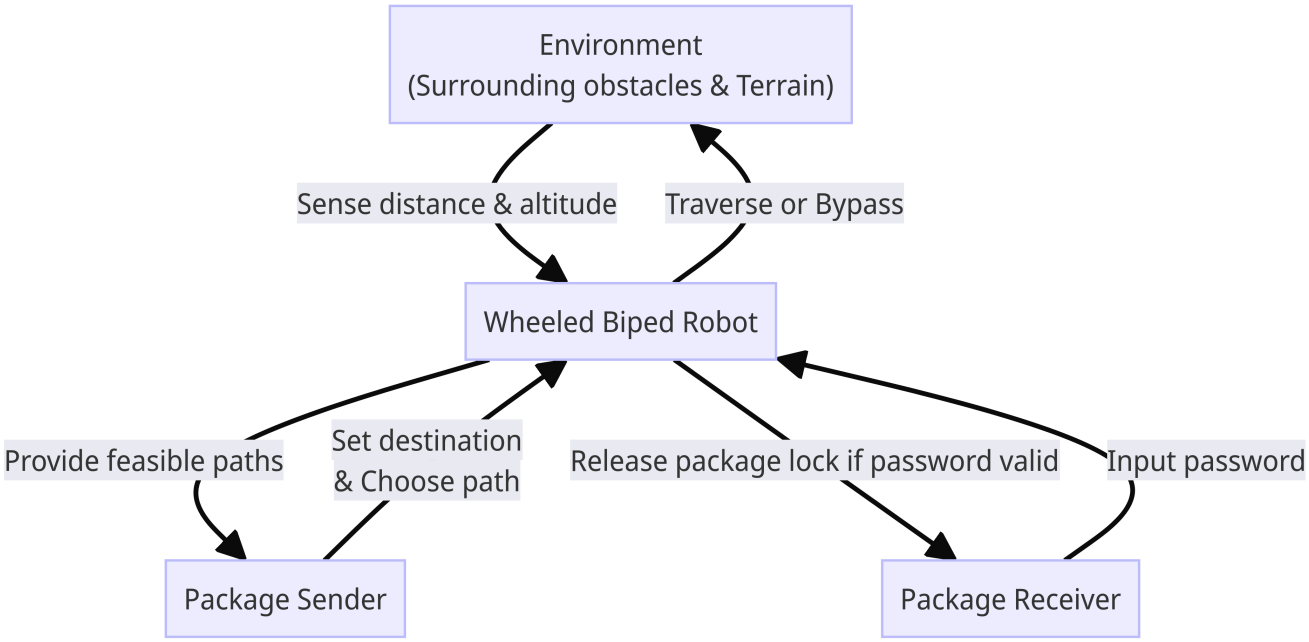

Figure 2: System Context Diagram

The following picture shows the design of the context diagram of the project, it illustrates the core interaction between the WBR and its immediate external entities. In this diagram, the WBR is the central system, the information regarding the condition of the environment is detected, such as the distance from the nearest obstacle, then the WBR takes the reaction on how to handle a certain situation. The package sender acts as the external entity, it sets the destination and picks a route for the WBR and the WBR provides it the feasible path and goes to complete the package delivery. The package receiver is another external entitle, it inputs password, in response to it, the WBR releases the package once it confirms the validity of the password.

2.2 Hardware Platform

2.2.1 Mechanical Platform

As mentioned in section 1.1, the WBR is constrained by the rules of 2024 RMUL. From page 26 to 27 of the robot specification manual, Committee (2023), our "Balancing Standard Robot" is constrained to have all ground-contacting surfaces aligned on the same line and cannot keep balance without dynamic adjustment. Additionally, to win the competition, the robot shall be light, fast, robust, and small.

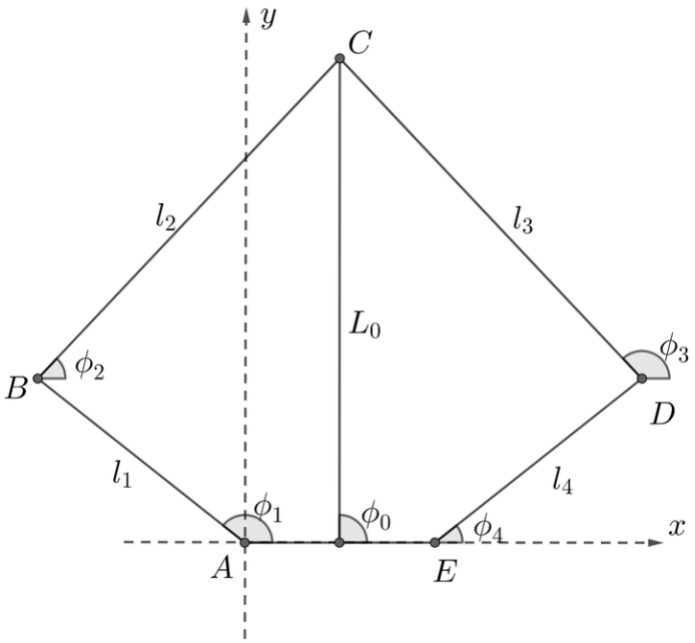

Figure 3: Sketch of WBR Leg Linkage, from Wang (2022)

Therefore, MacRM decided to put a five-bar linkage configuration on per side of the WBR, by following the paper Wang et al. (2021). Shape of each linkage, as illustrated in figure 3, is controlled by a pair of two hip-joint motors, which changes the overall robot posture as a result. The two motors contacting the ground are drive motors, which regulates the horizontal movement.

2.2.2 Electronic Platform

MacRM designed the electronic system according to the rules of RMUL as shown in figure 4. The rationale of choice of each specific component is out of scope of this document.

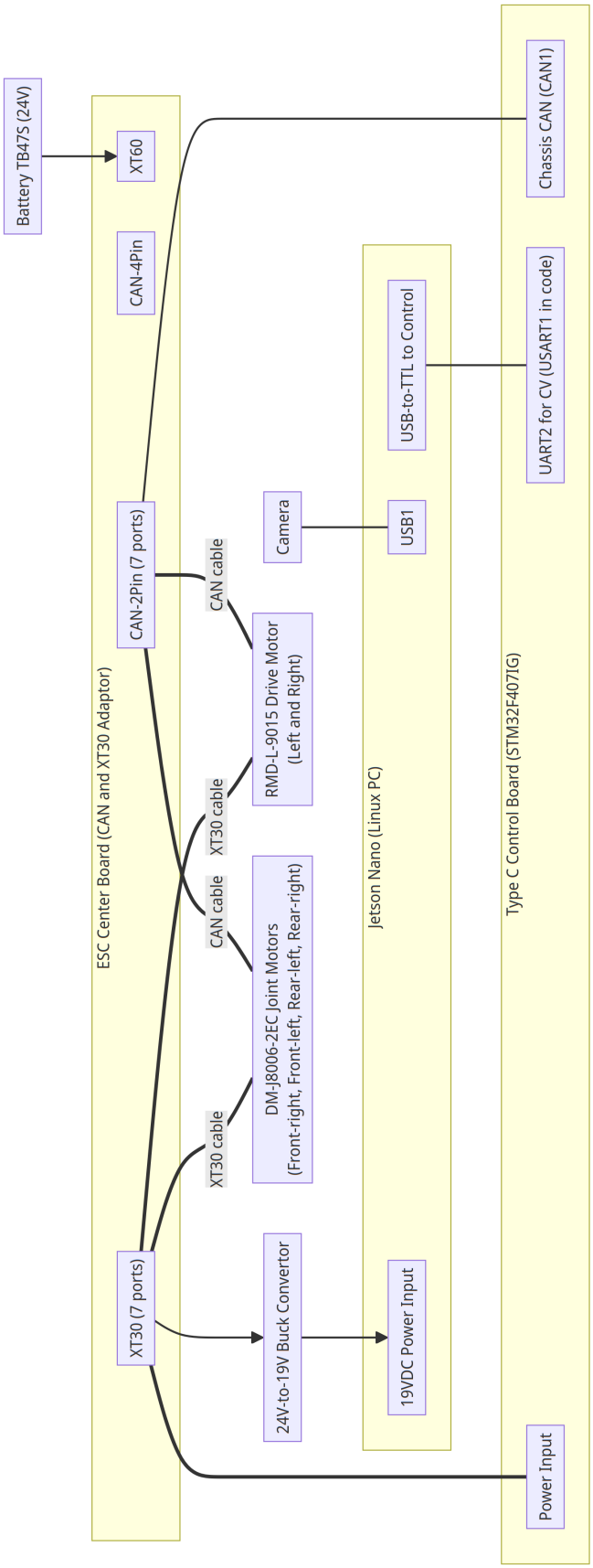

Figure 4: WBR Electronic System

3 System Variables

3.1 Variables

Table 2: Monitored Variables

Variable Name	**Type**	**Unit**	**Description**
m_CameraReading	2D list	N/A	Output from last visual odometry reading compare to land.
m_CurrentAcceleration	float	m/s^2	Current Acceleration of WBR
m_CurrentLocation	2D list	N/A	The current location WBR is at.
m_CurrentRotation	2D list	N/A	Current global Rotation in pitch yaw roll axis
m_CurrentSpeed	float	m/s	Current Speed of WBR
m_Destination	2D list	N/A	Final location that delivery need to go to.
m_GPSReading	2D list	N/A	Output from last GPS reading.
m_HipMotorAbsAngle	Digital Array	0-max (TBD)	Encoder value of four hip motors measuring absolute angle.
m_IMUReading	2D list	N/A	Output from last IMU reading.
m_InitiaLocation	2D list	N/A	m_Initial Location that WBR is currently at
m_NextLocation	2D list	N/A	The next location WBR need to go to.
m_Orientation	Analog Array	rad	Orientation of chassis to be sensed by IMU
m_TargetPitchAngle	Digital	m/s	Commanded roll angle of chassis platform by CV.
m_TargetRollAngle	Digital	m/s	Commanded roll angle of chassis platform by CV.
m_Time	Digital	s	Real-world time

Table 3: Controlled Variables

Variable Name	Type	Unit	Description
c_CurrentLocation	2D list	N/A	The current location WBR is at.
c_CurrentLocationVisual	2D list	N/A	Current global location of WBR.
c_DestinationArrived	Boolean	N/A	Boolean to determine if destination is arrived or not.
c_DriveMotorTorques	Digital Array	N m	Desired torque of drive motors
c_EmailSent	Boolean	N/A	Boolean to determine if email is sent or not.
c_EstimatedLocation	2D list	N/A	Output from SLAM module that get estimation from sensor fusion.
c_GeoFence	2D List	N/A	List and include all the point for geo boundaries.
c_HipMotorTorques	Digital Array	N m	Desired torque of hip motors
c_ObstaclesLocations	2D list	N/A	Collection of local obstacles locations in list.
c_OnDelivery	Boolean	N/A	Boolean to determine of delivery task is given or not.
c_ROSMap	2D List	N/A	Output that show all passable area.
c_TargetSpeed	float	m/s	Target speed that it need to go to.
c_TrajectoryPath	2D List	N/A	output that replace and insert into original path.

Table 4: Enumerated Variables

Enumerated Name	Description	Range
y_Posture	Robot posture state	{e_Jump, e_Normal, TBD}
y_TargetPosture	Commanded target posture state of robot	{e_Jump, e_Normal, TBD}

3.2 Constants

Almost all constants are not yet determined, but will be after details of mechanical model is finalized, assembled, and measured by MacRM.

Table 5: Constants

Constant Name	Type	Unit	Value	Description
k_JumpTimeout	Digital	second	TBD	The maximum amount of time the WBR shall finish any stage of jump mode
k_PidGains	Digital Array	N/A	TBD	Gains for each PID controller feedbacks.
k_L1Length	Digital	m	TBD	Length of link l1 in figure 3. Note that linkages on left and right sides are symmetric.
k_L2Length	Digital	m	TBD	Length of link l2 in figure 3. Note that linkages on left and right sides are symmetric.
k_L3Length	Digital	m	TBD	Length of link l3 in figure 3. Note that linkages on left and right sides are symmetric.
k_L4Length	Digital	m	TBD	Length of link l4 in figure 3. Note that linkages on left and right sides are symmetric.
k_MaxEquivLegLength	Digital	m	TBD	The maximum allowable equivalent length WBR can reach
k_MinEquivLegLength	Digital	m	TBD	The minimum allowable equivalent length WBR can reach
k_MaxEquivLegLength	Digital	m	TBD	The maximum allowable equivalent length WBR can reach
k_PidGains	Digital Array	N/A	TBD	Gains for each PID controller feedbacks.
k_RawMap	3D List	N/A	TBD	Input image being read as RGB 3D List.
k_RobotMass	Digital	kg	TBD	Approximate mass of robot without cargo.
k_WindowSize	float	m	TBD	Windows size that we need to update path with

4 System Behaviour Overview

4.1 Normal Operation

To make the behaviour of the product to achieve the target task, the Finite State Machine is created to describe the behaviour with detailed description provided after the picture.

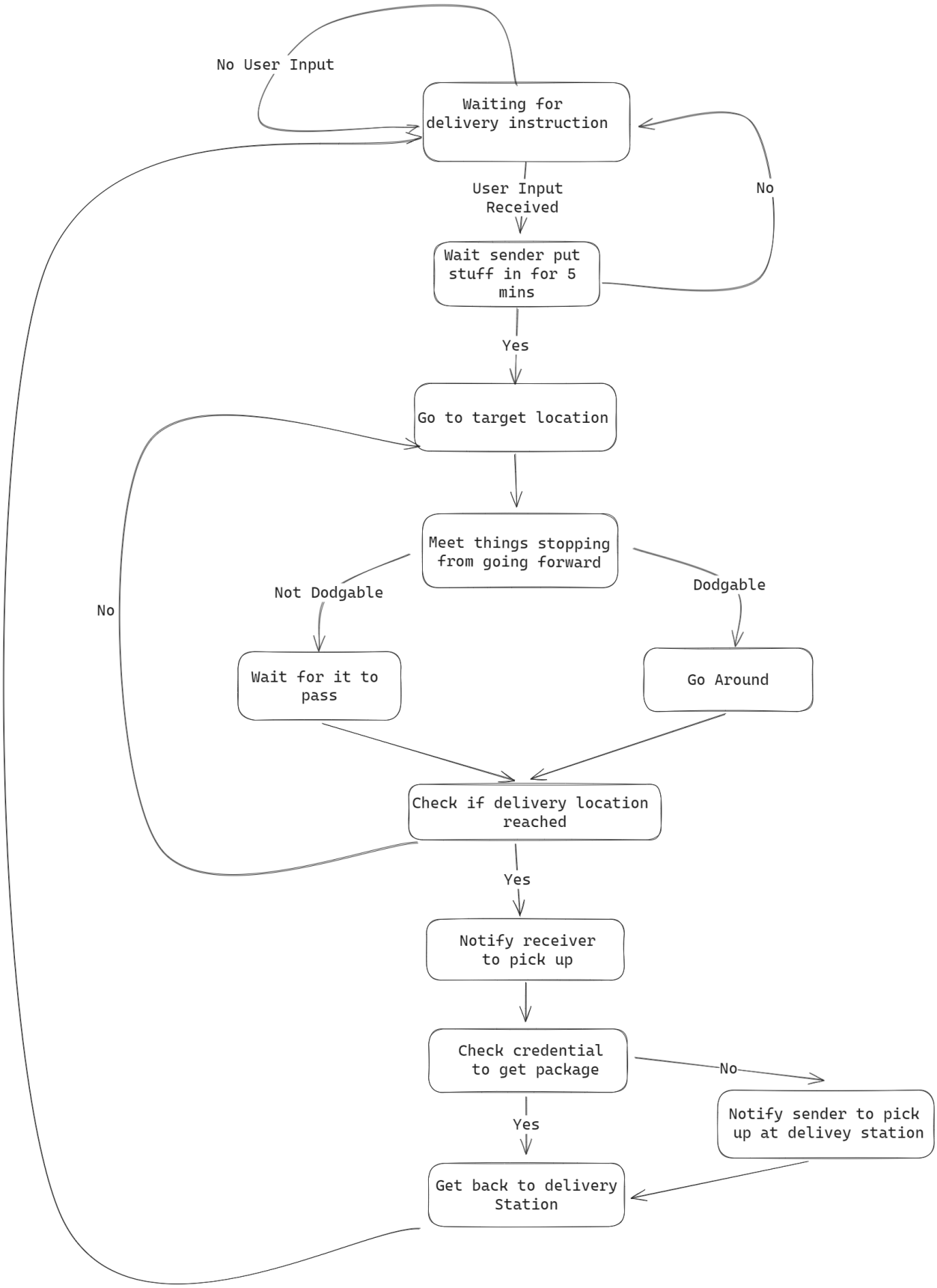

Figure 5: System Behaviour in Finite State Machine

4.2 Normal Operation State Descriptions

The Finite State Machine diagram briefly introduces the following general behaviours state. The description is given below:

Waiting for delivery instruction: It will need to stay in the delivery station to wait for new delivery instructions coming in from a client

Wait sender put stuff in for 5 mins: Upon receiving a delivery instruction, the robot initiates waiting status to allow sender side client to put stuff in delivery box for 5 mins. If there is nothing inside after 5 mins, it will return back the first status to waiting next delivery instruction

Go to target location: After sender side client put stuff in for delivery. Robot will start navigation to the target location.

Meet things stopping from going forward : When the sensor detects nearby obstacles, it communicates this information to the control system. The system temporarily interrupts the ongoing path travel task and initiates the execution of a special task.

Check if delivery location reached: When going closer to the delivery location or algorithm detect WBR reached target destination. It will go check if it actually reached or not.

Notify Reciver to pick up : After comfirming the target destination is reached, WBR will notify Reciver to pick package up.

Check credential to get package : This status will ensure only the person will credential can get the package, other wise it will bring the package back to original delivery station.

Get back to delivery Station : After finishing delivery task, it will go back to delivery station for next delivery task or wait there until previous failed delivery to be pick up.

4.3 Undesired Event Handling

The robot's undesired event handling is designed to ensure a high level of safety and operational integrity in various challenging scenarios. In the case of a collision or obstacle blocking, the robot promptly initiates an emergency stop to prevent any potential harm, followed by attempts to re-plan its path to navigate around the obstacle, prioritizing safety and adaptability. Loss of communication triggers the robot to make efforts to re-establish the link; if unsuccessful within a predefined time, it enters a safe mode, ceasing movements and alerting operators to prevent uncontrolled actions. In the event of a system fault, the robot logs the error, performs self-diagnosis, and initiates an emergency stop if necessary, providing detailed reports for maintenance teams to ensure quick troubleshooting and repairs. Human interference is addressed by the robot's safe response, avoiding collisions, initiating emergency stops when required, and alerting operators to prioritize

human safety. Loss of localization prompts the robot to re-establish its position; if unsuccessful, it enters a safe mode to prevent unpredictable behavior in the absence of accurate localization. These well-defined responses demonstrate a comprehensive approach to handling undesired events, emphasizing safety, adaptability, and effective communication with operators and maintenance teams.

Managing unexpected situations or errors during the operation of the Wheeled Biped Robot (WBP) is essential. Therefore, it is crucial to be aware of all potential events and to compile a comprehensive list summarizing these unexpected occurrences. All the possible unexpected events are listed below:

- **Collision or Obstacle Blocking:** occurs when the robot encounters physical obstacles or collides with objects and potentially impeding its movements, which potentially terminates its operation.
- **Loss of Communication:** Involves disruption between the robot and external environment.
- **System Fault:** This includes the malfunction or the failure of internal components or the error in software programs.
- **Human Interface:** Issues between robot and the human operators.
- **Loss of Localization:** This occurs when the robot losses its sense of position or orientation.

5 System Components

Below is a chart showing how the modules were decomposed. The following subsections act as the individual module guide.

Figure 6: Modular Decomposition Tree

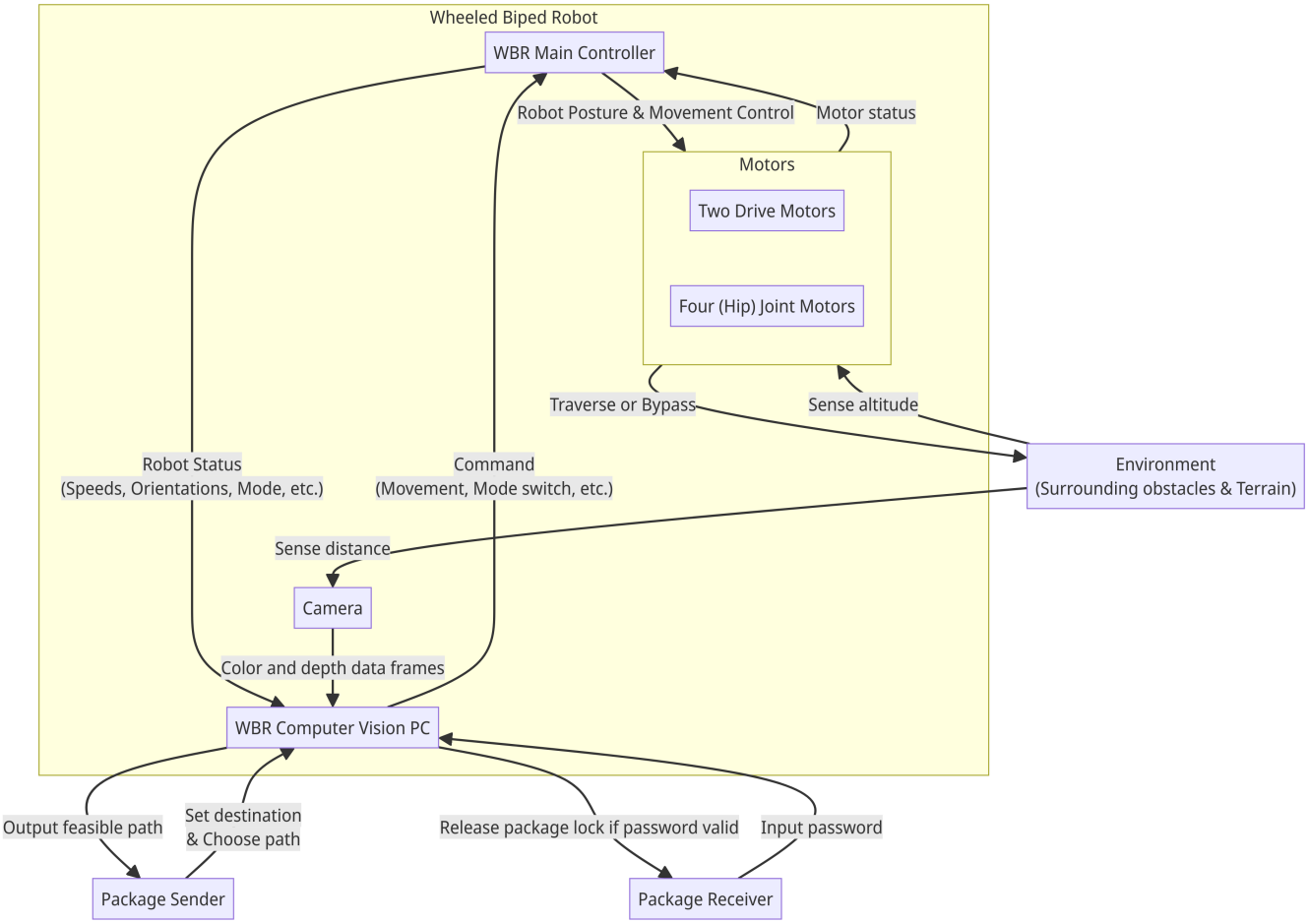

Figure 7: Complete System Diagram

5.1 Path Tracking & Planning Module

5.1.1 Global Path Planning: Trajectory Path Planning

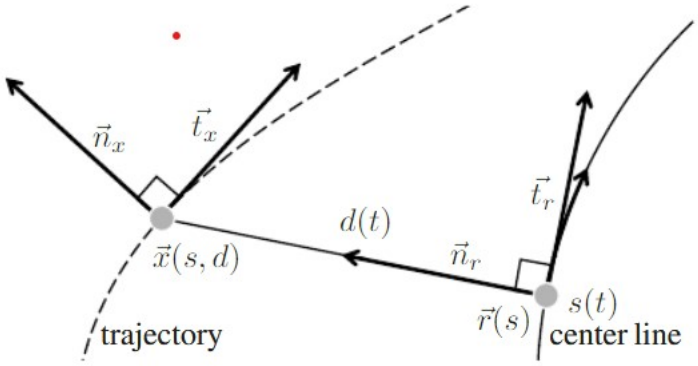

Fig. 2. Trajectory generation in a Frenét-frame

Figure 8: Optimal Trajectory Path Generation

5.1.1.1 Description

This module is the sub module for global path planning module. In order to get better path and less error during path tracking period, we need to choose trajectory path planning for WBR, Werling et al. (2010).

Using this algorithm we should able to get smooth movement on WBR. And the following expressions describe our trajectory path generation. x(s(t),d(t)) is the result Trajectory Path.

$$\vec{x}(s(t), d(t)) = \vec{r}(s(t)) + d(t)\vec{n_r}(s(t))$$

5.1.1.2 Inputs

Table 6: Input Variables of Trajectory Path Planning Module

Variable Name	Variable Type	Units	Range	Description
m_InitiaLocation	2D list	N/A	TBD	m_Initial Location that WBR is currently at
m_CurrentSpeed	float	m/s	TBD	Current Speed of WBR
m_CurrentAcceleration	float	m/s^2	TBD	Current Acceleration of WBR
m_Destination	2D list	N/A	TBD	Final location that delivery need to go to.

5.1.1.3 Outputs

Table 7: Output Variables of Trajectory Path Planning Module

Variable Name	Variable Type	Units	Range	Description
c_TrajectoryPath	2D list	N/A	TBD	Collection of points that WBR need follow alone the way.

5.1.1.4 Exception Handling

If destination is not in the range of area restriction (Geo Fence Area) system will give out warning to sender on python interface with. Then ask for re-enter the destination.
Any illegal input will give out warning and refuse to continue to next step.

5.1.1.5 Timing Constraints

This module will finish within 2 mins of input being given.

5.1.1.6 Initialization

Reset all to zero.

5.1.2 Global Path Planning : GPS Map Gathering and Transformation

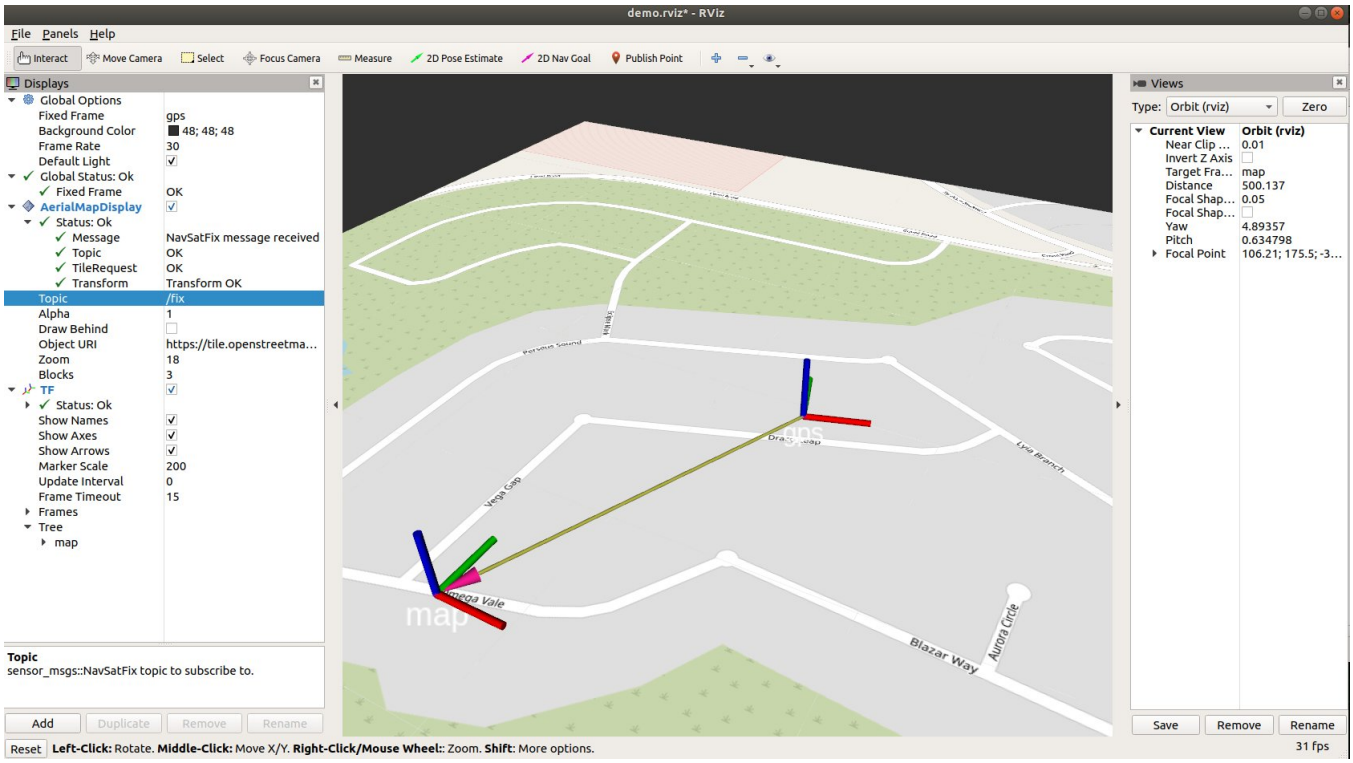

Figure 9: Map Gathering and Transformation

5.1.2.1 Description

This module is to transfer google map to grid map or other style of map that system can read for localization. We need to extra the image from google map to gather are that we want WBR to work within. Then output a grid map or other map format that can be used for localization with enough accuracy.

This module will be ran before put WBR into work. Since This is in the initialization process for the entire system.

5.1.2.2 Inputs

Table 8: Input Variables of GPS Map Gathering and Transformation

Variable Name	Variable Type	Units	Range	Description
k_RawMap	3D List	N/A	TBD	Input image being read as RGB 3D List.

5.1.2.3 Outputs

Table 9: Output Variables of GPS Map Gathering and Transformation

Variable Name	Variable Type	Units	Range	Description
c_ROSMap	2D List	N/A	TBD	Output that show all passable area.

5.1.2.4 Exception Handling

We will do a manual double check on the grid map generation to make sure there are no bad boundaries on the generated map.
At the same time, during WBR working condition, there will also be a Geo Fence applied to it on GPS that connect to internet in real-time.

5.1.2.5 Timing Constraints

Since this module need to be ran ahead of everything. The time constrain is very little, as long as it is able to finish in 10 mins, we will accept the time.

5.1.2.6 Initialization

Reset the output variable to zero and make sure there are enough storage for both map and image.

5.1.3 Local Path Planning : Local Obstacle Avoidance

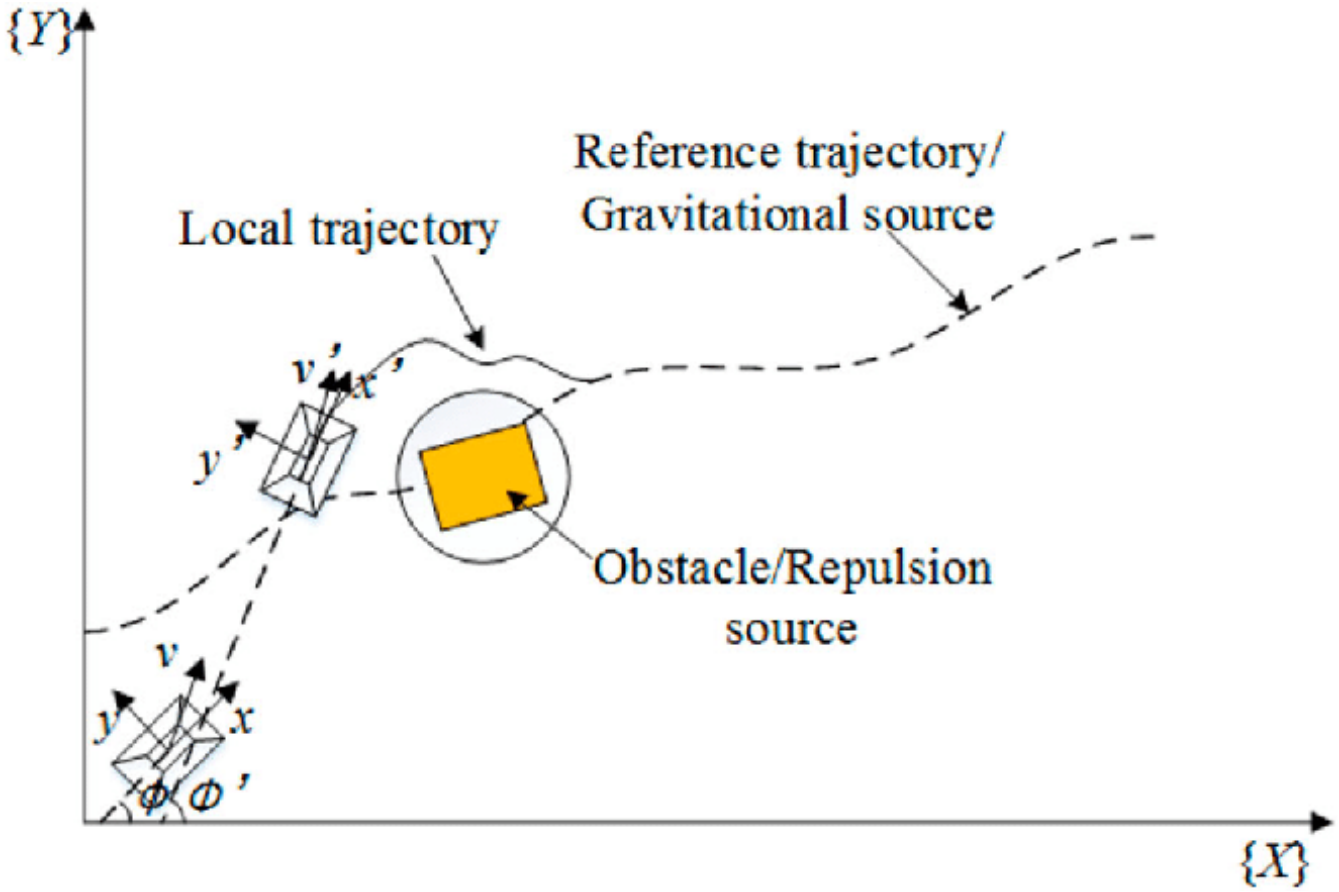

Figure 10: Local Obstacle Avoidance

5.1.3.1 Description

Local obstacle avoidance is a critical component in creating safe and reliable autonomous systems that can navigate complex and dynamic environments. Its successful implementation contributes to the overall efficiency and safety of robotic and autonomous applications in various fields. During delivery task, there are many unexpected obstacle that will show up, for example person, car, bike, cart, and animals. We need to make sure that all obstacle that are not on the original grid map will be dodged. The idea is gathered from Li et al. (2020) paper" Active vehicle obstacle avoidance based on integrated horizontal and vertical control strategy". Since this is an local obstacle avoidance for path planning, we will make a window within the size of environment sensing area and it need to react ahead of time.

5.1.3.2 Inputs

Table 10: Input Variables of Local Obstacle Avoidance

Variable Name	Variable Type	Units	Range	Description
c_TrajectoryPath	2D list	N/A	TBD	Collection of points that WBR need follow alone the way.
c_ObstaclesLocations	2D list	N/A	TBD	Collection of local obstacles locations in list.
m_CurrentSpeed	float	m/s	TBD	Current Speed of WBR
m_CurrentAcceleration	float	m/s^2	TBD	Current Acceleration of WBR
k_WindowSize	float	m	TBD	Windows size that we need to update path with

5.1.3.3 Outputs

Table 11: Output Variables of Local Obstacle Avoidance

Variable Name	Variable Type	Units	Range	Description
c_TrajectoryPath	2D List	N/A	TBD	output that replace and insert into original path.

5.1.3.4 Exception Handling

If SLAM update information slower than expected, the windows size will be increase at the same time. And If obstacle can not be avoid with Current Speed and acceleration, we will performance emergency stop.

5.1.3.5 Timing Constraints

This module need to finish in 100ms which is 10 Hz frequency that same as the frequency of LiDAR/Camera.

5.1.3.6 Initialization

Gather information from system and set windows size according to 10 Hz frequency.

5.1.4 Path Following : GPS,IMU, and Camera Localization

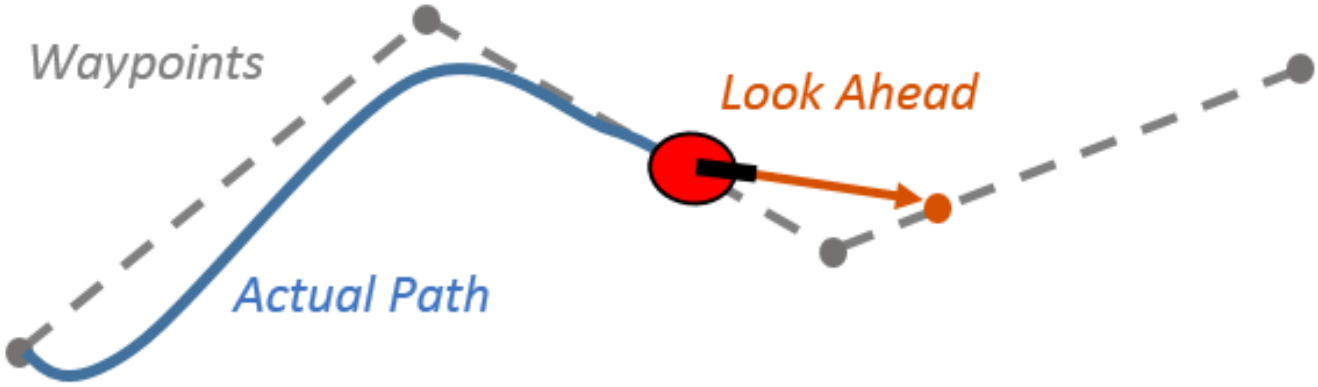

Figure 11: Path Following Concept

5.1.4.1 Description

This module runs with input from SLAM module that sensor fusion for GPS, IMU, and LIO/VIO have been applied. We take the final estimated location for the robot and plan the next movement based on current location.

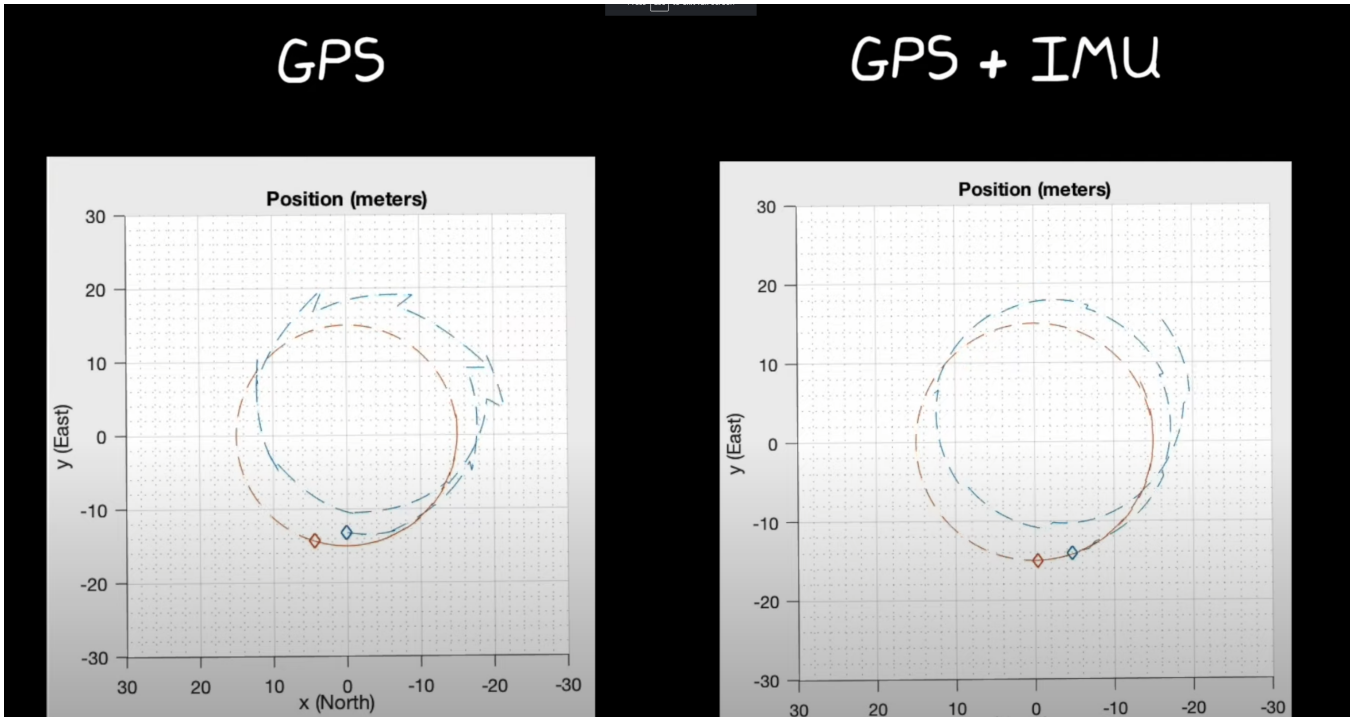

Figure 12: Sensor Fusion Result Comparison

5.1.4.2 Inputs

Table 12: Input Variables of GPS,IMU, and Camera Localization

Variable Name	Variable Type	Units	Range	Description
c_EstimatedLocation	2D list	N/A	TBD	Output from SLAM module that get estimation from sensor fusion.
c_CurrentLocation	2D list	N/A	TBD	The current location WBR is at.
m_NextLocation	2D list	N/A	TBD	The next location WBR need to go to.
m_CurrentSpeed	float	m/s	TBD	m_CurrentSpeed of WBR
m_CurrentAcceleration	float	m/s^2	TBD	m_CurrentAcceleration of WBR
m_CurrentRotation	2D list	N/A	TBD	Current global Rotation in pitch yaw roll axis

5.1.4.3 Outputs

Table 13: Output Variables of GPS,IMU, and Camera Localization

Variable Name	Variable Type	Units	Range	Description
c_TargetSpeed	float	m/s	TBD	Target speed that it need to go to.
m_CurrentRotation	2D list	N/A	TBD	target global Rotation in pitch yaw roll axis
m_CurrentLocation	2D list	N/A	TBD	The current location WBR is at.
c_DestinationArrived	Boolean	N/A	TBD	Boolean to determine if destination is arrived or not.

5.1.4.4 Exception Handling

If the robot is away from designated path by a lot. System will do a emergency stop and go back to previous point then set that point as temporary destination then recover to designated path.

5.1.4.5 Timing Constraints

Since this module need to work with SLAM, the required time to complete is 100ms and 10 Hz in frequency.

5.1.4.6 Initialization

Gather information from global and local path planning module.

5.2 Environment Sensing Module

5.2.1 Simultaneous Localization and Mapping : VIO or LIO Odometry with GPS

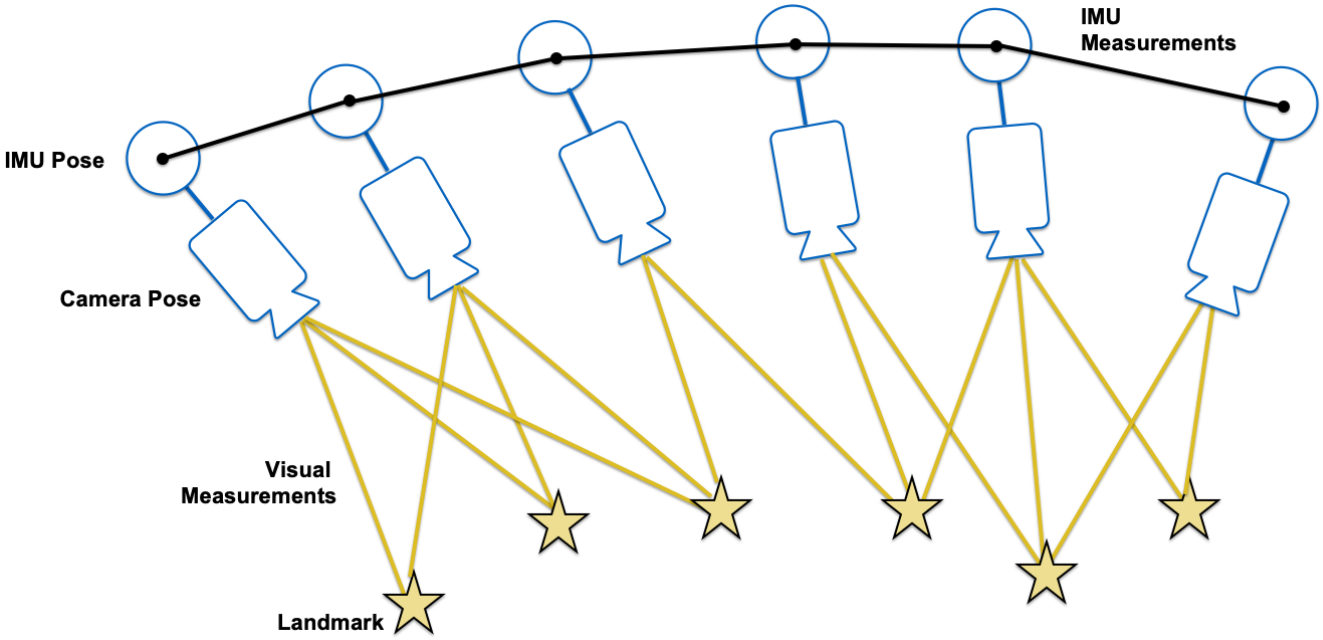

Figure 13: Visual IMU Odometry

5.2.1.1 Description

This module operates with inputs from the Simultaneous Localization and Mapping (SLAM) module, where sensor fusion for GPS, Inertial Measurement Unit (IMU), and Visual-Inertial Odometry (VIO) or Lidar-Inertial Odometry (LIO) has been applied. The final estimated location for the robot is obtained through SLAM, and based on this information, the module plans the next movement.

5.2.1.2 Inputs

Table 14: Input Variables of VIO or LIO Odometry with GPS

Variable Name	Variable Type	Units	Range	Description
m_IMUReading	2D list	N/A	TBD	Output from last IMU reading.
m_GPSReading	2D list	N/A	TBD	Output from last GPS reading.
m_CameraReading	2D list	N/A	TBD	Output from last visual odometry reading compare to land.
m_CurrentSpeed	float	m/s	TBD	Current Speed of WBR
m_CurrentAcceleration	float	m/s^2	TBD	Current Acceleration of WBR
m_CurrentRotation	2D list	N/A	TBD	Current global Rotation in pitch yaw roll axis

5.2.1.3 Outputs

Table 15: Output Variables of VIO or LIO Odometry with GPS

Variable Name	Variable Type	Units	Range	Description
estimated location	2D list	N/A	TBD	Output from SLAM module that get estimation from sensor fusion.

5.2.1.4 Exception Handling

If any sensor does not got read or disconnected during movement. That sensor will be abundant from sensor fusion.

5.2.1.5 Timing Constraints

This module need to be finished within 100ms as well which is the next reading will come up from sensors.

5.2.1.6 Initialization

Set everything to zero.

5.2.2 Simultaneous Localization and Mapping : Local Obstacle Update

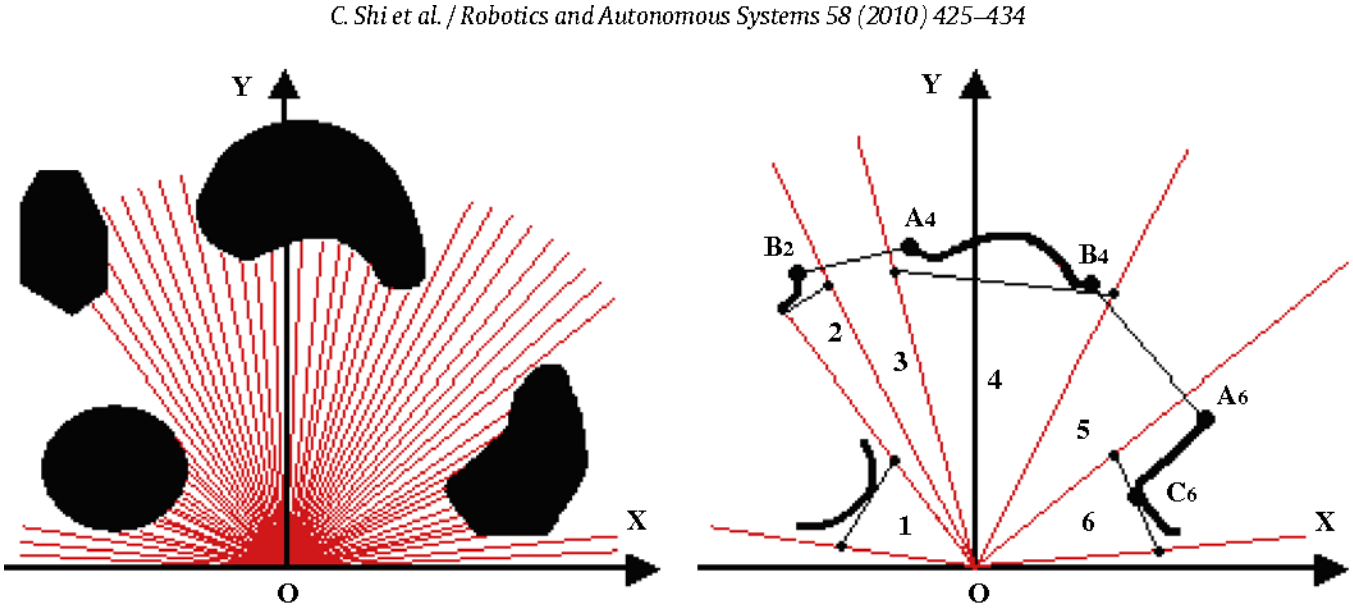

Figure 14: Local Obstacle Avoidance

5.2.2.1 Description

The Local Obstacle Update module is an integral component of the autonomous navigation system, tasked with continuously monitoring and updating the representation of obstacles within the immediate environment of the robot. This module operates in conjunction with sensors, such as lidar or cameras, to dynamically assess the surroundings and adapt the obstacle map in real-time.

5.2.2.2 Inputs

Table 16: Input Variables of Local Obstacle Update

Variable Name	Variable Type	Units	Range	Description
c_CurrentLocation	2D list	N/A	TBD	The current location WBR is at.
m_CameraReading	2D list	N/A	TBD	Output from last visual depth reading.
current location	2D list	N/A	TBD	The current location WBR is at.
m_CurrentRotation	2D list	N/A	TBD	Current global Rotation in pitch yaw roll axis

5.2.2.3 Outputs

Table 17: Output Variables of Local Obstacle Update

Variable Name	Variable Type	Units	Range	Description
c_ObstaclesLocations	2D list	N/A	TBD	Collection of local obstacles locations in list.

5.2.2.4 Exception Handling

Obstacle that is on the way and too close will trigger emergency stop. And if obstacles are blocking entire possible path, it will also stop and wait for 5 mins.

5.2.2.5 Timing Constraints

This module need to run at the same time as the same speed as SLAM and local avoidance module. So it need to be finished in 100ms.

5.2.2.6 Initialization

Set everything to 0.

5.2.3 Simultaneous Localization and Mapping : Area Restriction

Figure 15: Geo Fence

5.2.3.1 Description

The Area Restriction module is designed to confine the movement of WBR within predefined geographical boundaries. This module plays a crucial role in scenarios where it is essential to restrict the robot's access to specific areas, ensuring compliance with operational regulations or safety constraints. The primary function of this module is to monitor and control the robot's navigation, preventing it from entering restricted zones or areas with potential hazards.

5.2.3.2 Inputs

Table 18: Input Variables of Area Restriction

Variable Name	Variable Type	Units	Range	Description
c_ROSMap	2D List	N/A	TBD	Output that show all passable area.
c_GeoFence	2D List	N/A	TBD	List and include all the point for geo boundaries.

5.2.3.3 Outputs

Table 19: Output Variables of Area Restriction

Variable Name	Variable Type	Units	Range	Description
c_ROSMap	2D List	N/A	TBD	Output that show all passable area.

5.2.3.4 Exception Handling

Human correction will be applied for double check on the generated map.

5.2.3.5 Timing Constraints

Since this function runs right after map transformation and need human correction. So there is barely any time constraints for the generation pats.

5.2.3.6 Initialization

Gather information from map transformation module

5.3 Control Module

The control module is the heart of the system as it will be the primary system to interact with the surroundings in real-time. It has major functions to perform depending on the input from the CV system: Normal Movement Control and Jumping Control.

As illustrated in figure 16. Control system runs in a while loop that never stops until system shut down. For each loop, it runs Motion Planner, Forward Kinematics, Jump manager, and VMC modules in series. It also runs CV interface module in parallel.

5.3.1 Description

For normal situations when robot is moving on the ground, we used the open source code from Liu (2024), whose system design is illustrated in Wang (2022). This section gives an overview of the open source design.

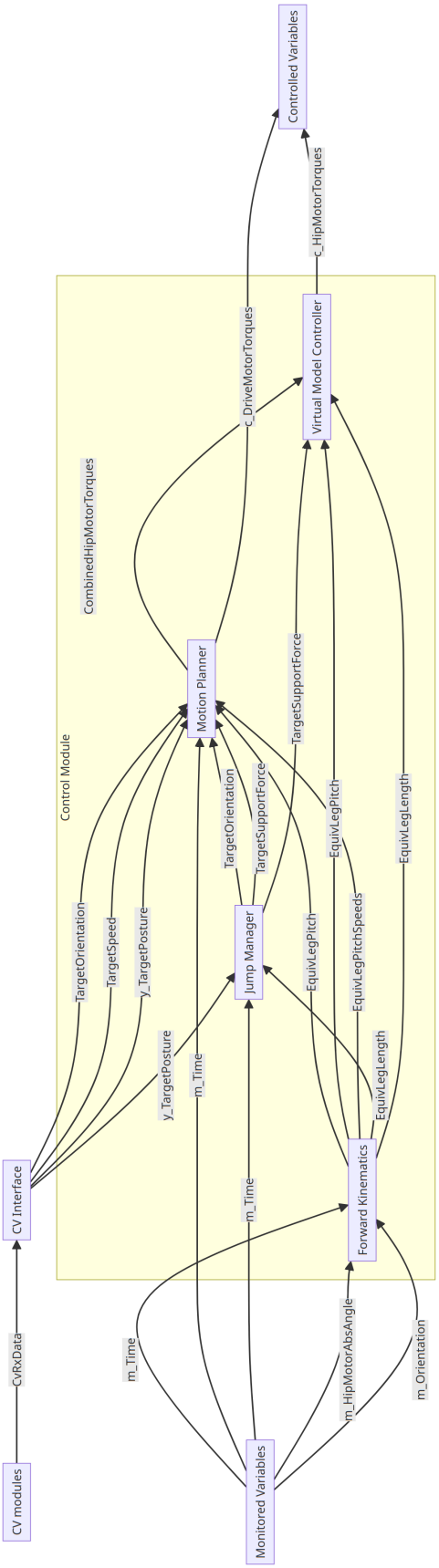

Figure 16: Control Model Decomposition

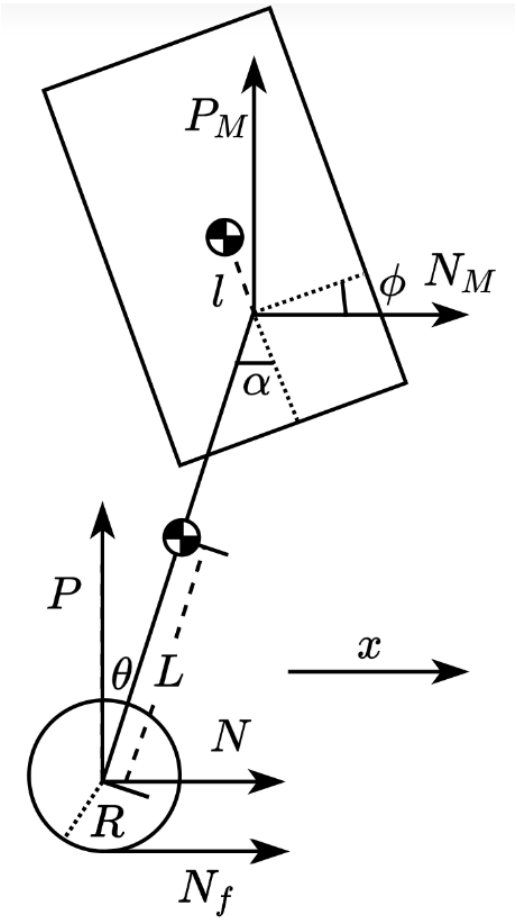

Figure 17: WBR Control Model Side View, from Wang (2022)

The symbols shown in Figure 17 are defined in table 20. The supporting leg here is the segment connecting center of a side of the chassis to the wheel center and is called "equivalent leg". The preliminary assumption for normal movement analysis is that hip angle is fixed, so that length of this equivalent leg does not change, and we can apply inverted-pendulum model to let robot balance to stable position. Left and right side of the robot are calculated separately because they may have different leg length when operating.

The main difference between normal movement control and jumping control is that the latter situation requires consideration of change in leg length. In normal movement control situation we will make sure change in leg lengths are slow that we can assume they are fixed at each time instance.

Symbols	Description	Positive Direction	Unit
θ	Angle between equivalent leg and vertical direction	As shown in the figure	rad
x	Displacement in drive wheel	As shown in the figure	m
ϕ	Angle between robot body and horizontal direction	As shown in the figure	rad
T	Output torque by drive motor	Same as θ	$N \cdot m$
T_p	Combined output torque by the two hip motors	Same as α	$N \cdot m$
N	Horizontal portion of supporting force by drive wheel	As shown in the figure	N
P	Vertical portion of supporting force by drive wheel	As shown in the figure	N
N_M	Horizontal portion of supporting force by equivalent leg	As shown in the figure	N
N_f	Friction force on drive wheel by the ground	As shown in the figure	N
R	Radius of drive wheel	N/A	m
L	Distance from mass center of equivalent leg to drive wheel	N/A	m
L_M	Distance from mass center of equivalent leg to rotational center of robot body	N/A	m
l	Distance from mass center of robot body to rotational center of it	N/A	m
m_w	drive wheel mass	N/A	kg
m_p	equivalent leg mass	N/A	kg
M	Robot body mass	N/A	kg
I_w	Rotational momentum of drive wheel around its mass center	N/A	$kg \cdot m^2$
I_p	Rotational momentum of equivalent leg around its mass center	N/A	$kg \cdot m^2$
I_M	Rotational momentum of robot body around its mass center	N/A	$kg \cdot m^2$

Table 20: WBR Control Model Side View Symbols

And the following expressions describe our aggregated nonlinear system. x specifies all the control system inputs, and u specifies all the outputs.

$$\vec{x} = \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix}, \vec{u} = \begin{bmatrix} T \\ T_p \end{bmatrix}, \dot{\vec{x}} = f(\vec{x}, \vec{u})$$

Our objective is to let robot approach upright posture with zero torque to maintain in ideal situation.

$$\vec{x} = \begin{bmatrix} 0 \\ 0 \\ x \\ 0 \\ 0 \\ 0 \end{bmatrix}, \vec{u} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In order to approach this ideal state, the open source Wang (2022) chose to use an LQR controller, so do we. Because details are extensively stated on the referred website, it is not stated here. The main contribution of our project is to adapt this controller software to our specific hardware system, and also implement Jumping control to it. The main idea of the LQR control is to construct system input to be,$\vec{u} = -K\vec{x}$, where the gain matrix K is calculated as an result of LQR optimization, based on given form of cost function, and it only depends on the equivalent leg length when robot is operating.

5.3.2 Inputs

Table 21: Input Variables of General Control Module

Variable Name	Variable Type	Units	Range	Description
m_TargetRollAngle	Digital	m/s	-1 to 1	Commanded roll angle of chassis platform by CV.
m_TargetPitchAngle	Digital	m/s	-1 to 1	Commanded roll angle of chassis platform by CV.
y_Posture	Digital	N/A	{e_Jump, e_Normal, TBD}	Commanded posture by CV.
m_TargetSpeed	Digital	m/s	-1 to 1	Commanded velocity by CV.
m_HipMotorAbsAngle	Digital Array	0-max (TBD)	TBD	Encoder value of four hip motors measuring absolute angle.
m_Orientation	Analog Array	rad	TBD	Orientation of chassis to be sensed by IMU
m_Time	Digital	s	TBD	Real-world time

5.3.3 Outputs

Table 22: Output Variables of General Control Module

Variable Name	Variable Type	Units	Range	Description
c_DriveMotorTorques	Digital Array	N m	TBD	Desired torque of drive motors
c_HipMotorTorques	Digital Array	N m	TBD	Desired torque of hip motors

5.3.4 Internal Variables

Table 23: Internal Variables of General Control Module

Variable Name	Variable Type	Units	Range	Description
EquivLegPitch	Digital Array	rad	TBD	Angle between equivalent left/right leg and horizontal direction. Calculated by forward Kinematics of figure 3.
EquivLegPitchSpeeds	Digital Array	rad	TBD	Derivative of EquivLegPitch
DriveWheelDisplacements	Digital Array	rad	TBD	Ground displacement by drive wheels
DriveWheelSpeed	Digital	rad	TBD	Derivative of DriveWheelDisplacement
EquivLegLength	Digital Array	rad	TBD	Length of equivalent left/right legs which is distance from rotational center of robot body to drive wheel
PrevTime	Digital	s	TBD	Time at previous interrupt

5.3.5 Exception Handling

Input and output validation will be the main form of exception handling. All mandatory values will be checked to ensure they are not empty, and not out of specified limit. If any input or output is out of specified limit, encapsulate it to the maximum or minimum.

In addition, if pitch of robot body, i.e. m_ϕ, is greater than a value, all motors shall be shut down to prevent undetermined damage when system has tripped over.

5.3.6 Timing Constraints

Operate as fast as possible so that velocity measurement can be as precise as possible. This is the highest priority task of control board.

5.3.7 Initialization

Reset all variables to zero.

5.4 Control Submodule: Forward Kinematics Module

5.4.1 Description

Convert hip motor angles to end-effector (drive motor) configurations according to figure 3.

5.4.2 Inputs

Table 24: Input Variables of Forward Kinematics Module

Variable Name	Variable Type	Units	Range	Description
m_HipMotorAbsAngle	Digital Array	0-max (TBD)	TBD	Encoder value of four hip motors measuring absolute angle.
m_Orientation	Analog Array	rad	TBD	Orientation of chassis to be sensed by IMU
m_Time	Digital	s	TBD	Real-world time

5.4.3 Outputs

Table 25: Output Variables of Forward Kinematics Module

Variable Name	Variable Type	Units	Range	Description
EquivLegPitch	Digital	rad	TBD	Angle between equivalent leg and horizontal direction. ϕ_0 in figure 3.
EquivLegPitchVelocity	Digital	rad	TBD	Derivative of EquivLegPitch
EquivLegLength	Digital Array	rad	TBD	Length of equivalent left/right legs which is distance from rotational center of robot body to drive wheel

5.4.4 Exception Handling

Input and output validation will be the main form of exception handling. All mandatory values will be checked to ensure they are not empty, and not out of specified limit. If any input or output is out of specified limit, encapsulate it to the maximum or minimum.

5.4.5 Timing Constraints

Same as section 5.3.6.

5.4.6 Initialization

Reset all input variables to zero.

5.5 Control Submodule: Jump Manager

5.5.1 Description

Jump Manager dynamically change control strategy and parameters to perform jumping action once commanded by the decision-making modules. This module is established to monitor the physical

state of WBR and the real world time to govern the jumping procedure of WBR. The ultimate goal is to execute a safe and effective jump that significantly contributes to the robot's travel. The figure 18 shows the whole process of jump action of WBR, including the process before it receives the jump command and after if fully completes the jump action and resumes initial position.

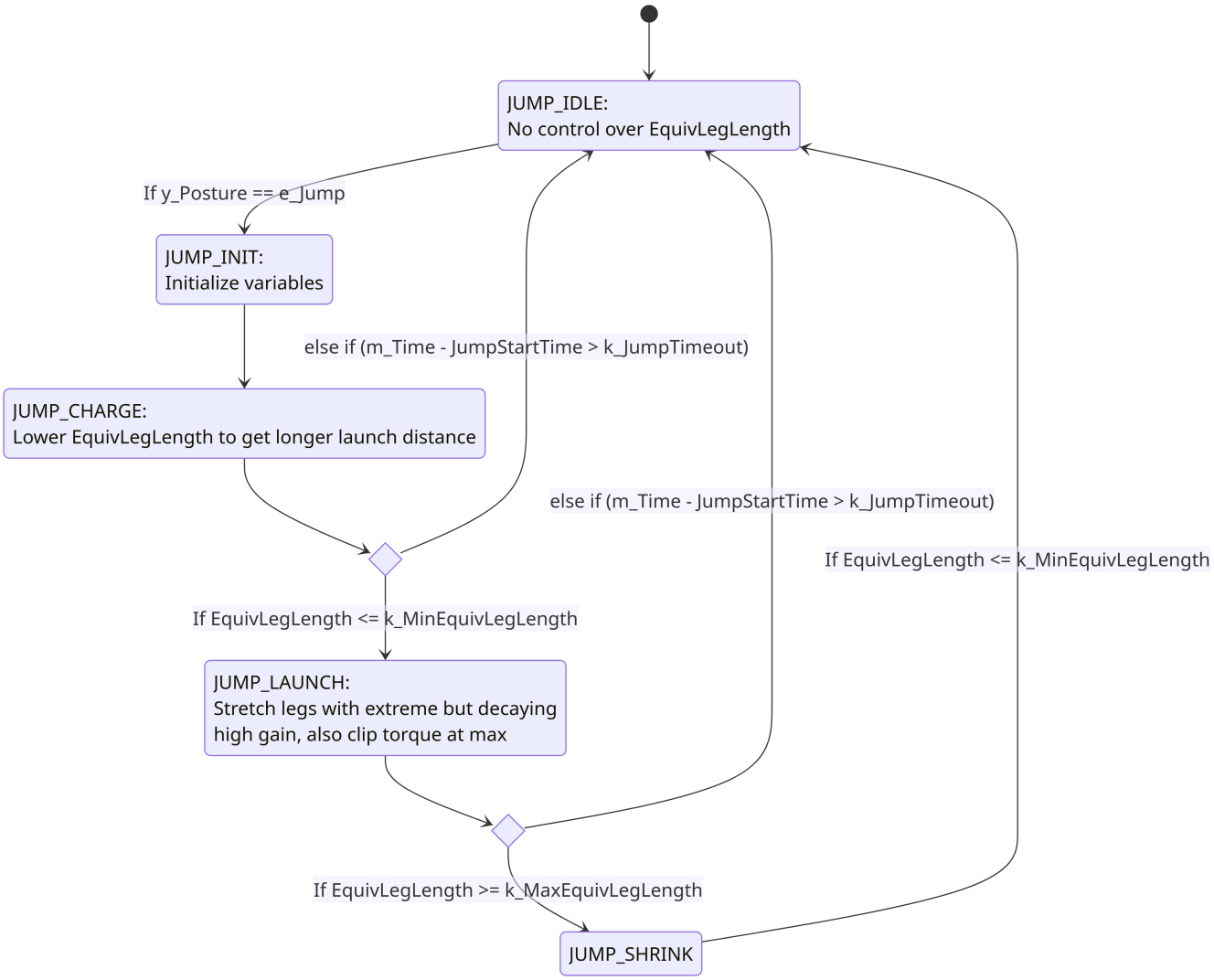

Figure 18: Jump FSM

5.5.2 Jump State Descriptions

Before WBR receives any jump command, it stays in JUMP_IDLE state meaning it has nothing to do with jump process at the moment. Once there is a jump command, it starts to prepare and initialize all the variables that required to monitor and control during the jump action. By inputting **m_Time**, the system continuously checking if in each state, the elapse time is greater than **k_JumpTimeout**, which represents the maximum allowable duration for the jump action.

This precautionary measure ensures that the system avoids being caught in an error state. By inputting and monitoring the variable **EquivLegLength** to take this physical state of WBR as a reference to deduce which stage the robot is in. To compare the **EquivLegLength** with the lowest acceptable height CoM, it can be inferred that if the robot finishes charging energy and will need to stretch legs and increase **c_DriveMotorTorques** and **c_HipMotorTorques** for a jump. To compare **EquivLegLength** with **k_MinEquivLegLength**, the system will determine whether to decrease **c_DriveMotorTorques** and **c_HipMotorTorques** to bring the WBR back to a horizontal position.

JUMP_IDLE: WBR stays idle when there is no command received.

JUMP_CHARGE: WBR lower the CoM to a predetermined level for the purpose of getting a longer launch distance. This primarily aimed at accumulating more energy for a higher jump. It is important to ensure CoM remains within the robot's bearing capacity, and it should not exceed the lowest acceptable CoM level.

JUMP_LAUNCH: Stretching legs but with decaying gain. The intentional reduction in gain serves as a buffer, mitigating large torques to prevent potential damage to the components.

JUMP_SHRINK: Lower CoM to its initial position.

5.5.3 Inputs

Table 26: Input Variables of Jumping Control Module

Variable Name	Variable Type	Units	Range	Description
m_Time	Digital	s	TBD	Real-world time
EquivLegLength	Digital Array	rad	TBD	Length of equivalent legs of left and right side which is distance from rotational center of robot body to drive wheel
y_TargetPosture	Digital	N/A	{e_Jump, e_Normal, TBD}	Commanded target posture state of robot

5.5.4 Outputs

Table 27: Output Variables of Jumping Control Module

Variable Name	Variable Type	Units	Range	Description
TargetOrientation	Digital Array	rad	TBD	Commanded target orientation of chassis
TargetSupportForce	Digital Array	N	TBD	Array of desired force and momentum going to apply to each leg

5.5.5 Internal Variables

Table 28: Internal Variables of Jumping Control Module

Variable Name	Variable Type	Units	Range	Description
IsJumpTheAir	Boolean	N/A	N/A	Indicating WBR is currently in the air or not.
JumpState	Enum	N/A	N/A	representing the current state of the jump procedure
PrevTime	Digital	s	TBD	Time at previous interrupt

5.5.6 Exception Handling

Input and output validation will be the main form of exception handling. All mandatory values will be checked to ensure they are not empty, and not out of specified limit. If any input or output is out of specified limit, encapsulate it to the maximum or minimum.

Input variables will be checked if they are in the correct data type, as specified in the table above. If any of the data type is wrong, the system promptly reports the discrepancy as an issue.

5.5.7 Timing Constraints

Operate as fast as possible so that jump stage control can be as precise as possible.

5.5.8 Initialization

Reset all variables to zero.

5.6 Control Submodule: Motion Planner Module

5.6.1 Description

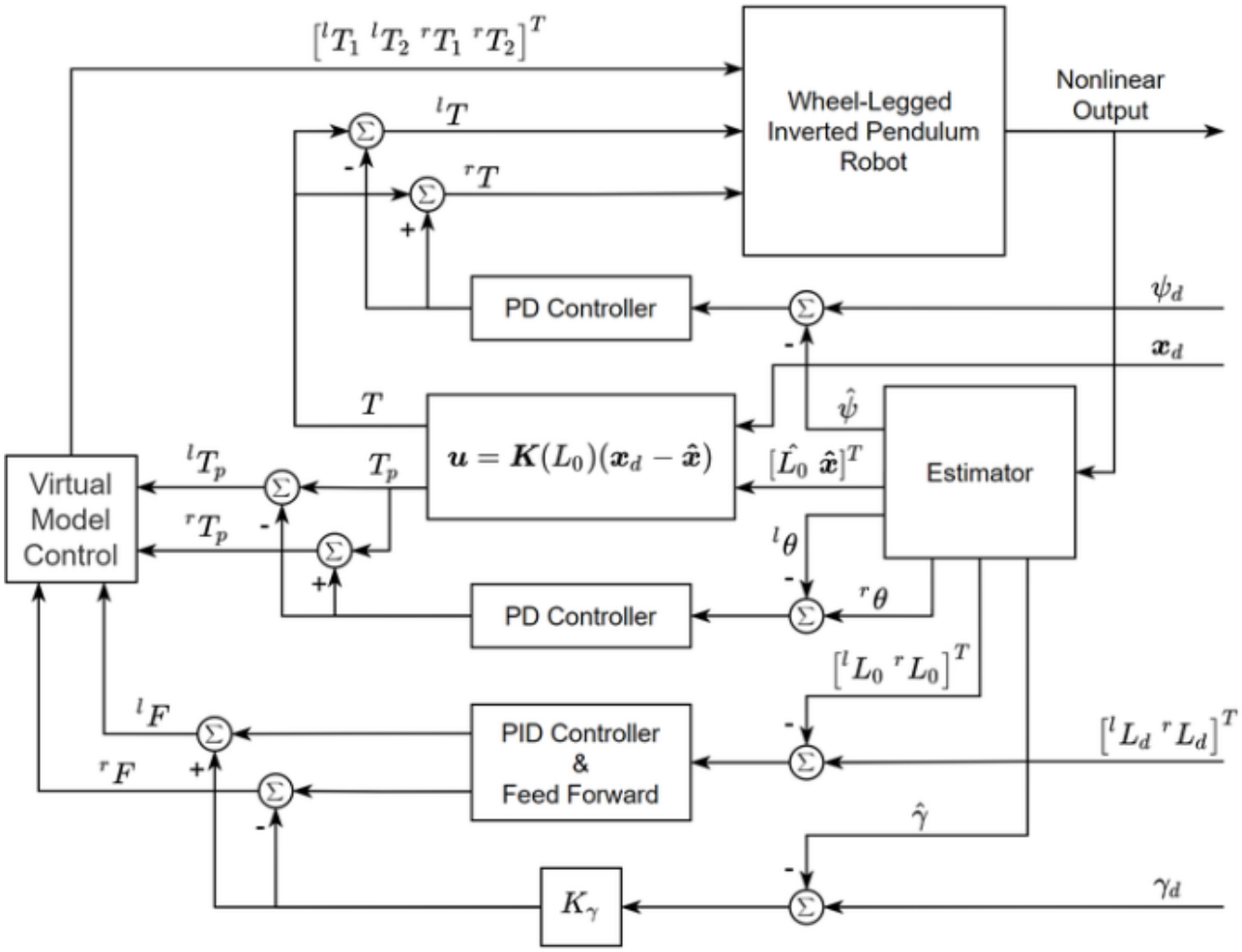

Figure 19: Motion Planner Design from Wang (2022)

Symbols	Description	Unit
ϕ_d	Robot body pitch target angle	rad
γ_d	Robot body roll target angle	rad
$[{}^l\theta, {}^r\theta]$	EquivLegPitch	rad
T	Output torque by drive motor	N*m
T_p	Combined output torque by the two hip motors	N*m
x_d	Current system status (main input)	N/A
$[{}^lL_b, {}^rL_b]$	Current left and right equivalent leg length provided by forward kinematics	m

Table 29: Main Symbols in Motion Planner Design

Integration of motion planner and VMC is shown in Figure 19. We directly use the control system implemented by Wang (2022), so here we only give an overview.

The estimator observes the IVP plant and provides current system status to the controllers. Three PID controller takes care of slight adjustment to the pitch and roll angle of robot body, and angle difference between left and right equivalent legs, respectively. The target for angle difference between equivalent legs is always set to zero.

The main LQR block, $\vec{u} = K(L_0)(\vec{x_d} - \vec{\hat{x}})$, constructs the output u as a linear combination of input x, where the gain is dependent on the equivalent leg length. Its outputs, planned drive motor torques and combined hip motor torques are passed into IVP and VMC for further control, respectively. VMC converts combined hip motor torques into individual hip motor torques. IVP directly outputs the drive motor torques.

5.6.2 Inputs

Table 30: Input Variables of Motion Planner

Variable Name	Variable Type	Units	Range	Description
m_Time	Digital	s	TBD	Real-world time
m_HipMotorAbsAngle	Digital Array	0-max (TBD)	TBD	Encoder value of four hip motors measuring absolute angle.
m_Orientation	Analog Array	rad	TBD	Orientation of chassis to be sensed by IMU
EquivLegPitch	Digital	rad	TBD	Angle between equivalent leg and horizontal direction. ϕ_0 in figure 3.
EquivLegPitchSpeeds	Digital Array	rad	TBD	Derivative of EquivLegPitch

5.6.3 Outputs

Table 31: Output Variables of Motion Planner

Variable Name	Variable Type	Units	Range	Description
c_DriveMotorTorques	Digital Array	N m	TBD	Desired torque of drive motors
CombinedHipMotorTorques	Digital Array	N m	TBD	Desired combined torque of hip motors

5.6.4 Internal Variables

Table 32: Internal Variables of Motion Planner

Variable Name	Variable Type	Units	Range	Description
DriveWheelDisplacements	Digital Array	rad	TBD	Ground displacement by drive wheels
DriveWheelSpeed	Digital	rad	TBD	Derivative of DriveWheelDisplacement
EquivLegLength	Digital Array	rad	TBD	Length of equivalent left/right legs which is distance from rotational center of robot body to drive wheel
PrevTime	Digital	s	TBD	Time at previous interrupt

5.6.5 Exception Handling

Input and output validation will be the main form of exception handling. All mandatory values will be checked to ensure they are not empty, and not out of specified limit. If any input or output is out of specified limit, encapsulate it to the maximum or minimum.

Input variables will be checked if they are in the correct data type, as specified in the table above. If any of the data type is wrong, the system promptly reports the discrepancy as an issue.

5.6.6 Timing Constraints

Operate as fast as possible so that motion planning can be as precise as possible.

5.6.7 Initialization

Reset all variables to zero.

5.7 Control Submodule: VMC Module

5.7.1 Description

Virtual Model Control (VMC) is a motion control language that uses simulations of imagined mechanical components to create forces, which are applied through real joint torques, thereby creating the illusion that the virtual components are connected to the robot, Pratt et al. (1997). For our particular use, VMC converts combined hip motor torques into individual hip motor torques. It controls the system with the assumption of ideal virtual model as shown in figure 3, and we shall apply the static force transformation formula $\vec{\tau} = J^T F$ to left and right side of the robot separately, where τ consists of desired front and back hip motor torques. Note that we have to assume acceleration and speed of all links are small enough to ignore. Derivation of the theorem can be found in chapter 5.10 of Siciliano et al. (2009). J^T is transpose of Jacobian matrix of linkage configuration, and F consists of desired external forces and momentum to generate, which are supporting force and momentum on the drive wheel end of equivalent leg.

5.7.2 Inputs

Table 33: Input Variables of VMC Module

Variable Name	Variable Type	Units	Range	Description
CombinedHipMotorTorques	Digital Array	N m	TBD	Desired combined torque of hip motors
TargetSupportForce	Digital array	N	TBD	Array of desired force and momentum going to apply to each leg.
EquivLegPitch	Digital Array	rad	TBD	Angle between equivalent left/right leg and horizontal direction. Calculated by forward Kinematics of figure 3.
EquivLegLength	Digital Array	rad	TBD	Length of equivalent left/right legs which is distance from rotational center of robot body to drive wheel

5.7.3 Outputs

Table 34: Output Variables of VMC Module

Variable Name	Variable Type	Units	Range	Description
c_HipMotorTorques	Digital Array	$N \cdot m$	TBD	Desired torques of hip motors

5.7.4 Exception Handling

Input and output validation will be the main form of exception handling. All mandatory values will be checked to ensure they are not empty, and not out of specified limit. If any input or output is out of specified limit, encapsulate it to the maximum or minimum.

Input variables will be checked if they are in the correct data type, as specified in the table above. If any of the data type is wrong, the system promptly reports the discrepancy as an issue.

5.7.5 Timing Constraints

Operate as fast as possible so that it does not block other control sub-modules by much.

5.7.6 Initialization

Reset all variables to zero.

5.8 CV Interface Module

5.8.1 Description

CV Interface is the custom communication interface between CV and control board designed by us. We choose to use the 1 Mbps UART. This section explains our custom communication protocol. All the code, detailed implementation, and explanation are recorded in Team (2023).

The figure 20 shows the general sequence CV and control board will follow to establish a communication session, synchronize timestamp, exchanging miscellaneous information, and communicating with robot movement (auto-move) and camera gimbal rotation (auto-aim) decisions. The camera gimbal rotation feature may not be implemented in this project, since it depends on whether we have time to add the camera gimbal module. If the mechanical structure of camera gimbal is not going to be added, we will fix the camera to the chassis platform.The drawback is the data input will not be stable for SLAM to process, but it is good enough for this project.

Design ideas of the sequence diagram in figure 20:

- Unit of all timestamps is ms, format is uint16_t. It's defined by timestamp = raw_timestamp - sync_time. if raw_timestamp overflow and becomes smaller than sync_time, timestamp = raw_timestamp - sync_time + 0x10000 to avoid negative number. Exception are TranDelta and TranDelta_MA, which are int16_t, because it may become negative when control board's tick updates slower than CV
- Reason for CV to request for CvSyncTime at around the 8th step: CV may send multiple ACKs to multiple set-mode requests that are stuck in the buffer before CV starts processing. CV will memorize the CvSyncTime of the last ACK, while control board may not, since some messages may be lost caused by turn around time of DMA controller. Therefore, to correlate, control board must have latest CvSyncTime stored and sent to CV when requested
- TranDelta history should be cleared every time Control board boots up.
- For consistency, all timestamps within package headers should be synchronized, i.e. offseted by SyncTime, even before the synchronization process, where . For example, for control board, the timestamp should be CtrlTimestamp(Tx) - CtrlSyncTime. For CV, the timestamp should be CvTimestamp(Tx) - CvSyncTime.
- For all msgs control receives from CV, update TranDelta_MA and its history. Newest TranDelta = CtrlTimestamp(Rx) - CtrlSyncTime - MsgTimestamp, except for the ACK msg where calculation is different as illustrated in the diagram below

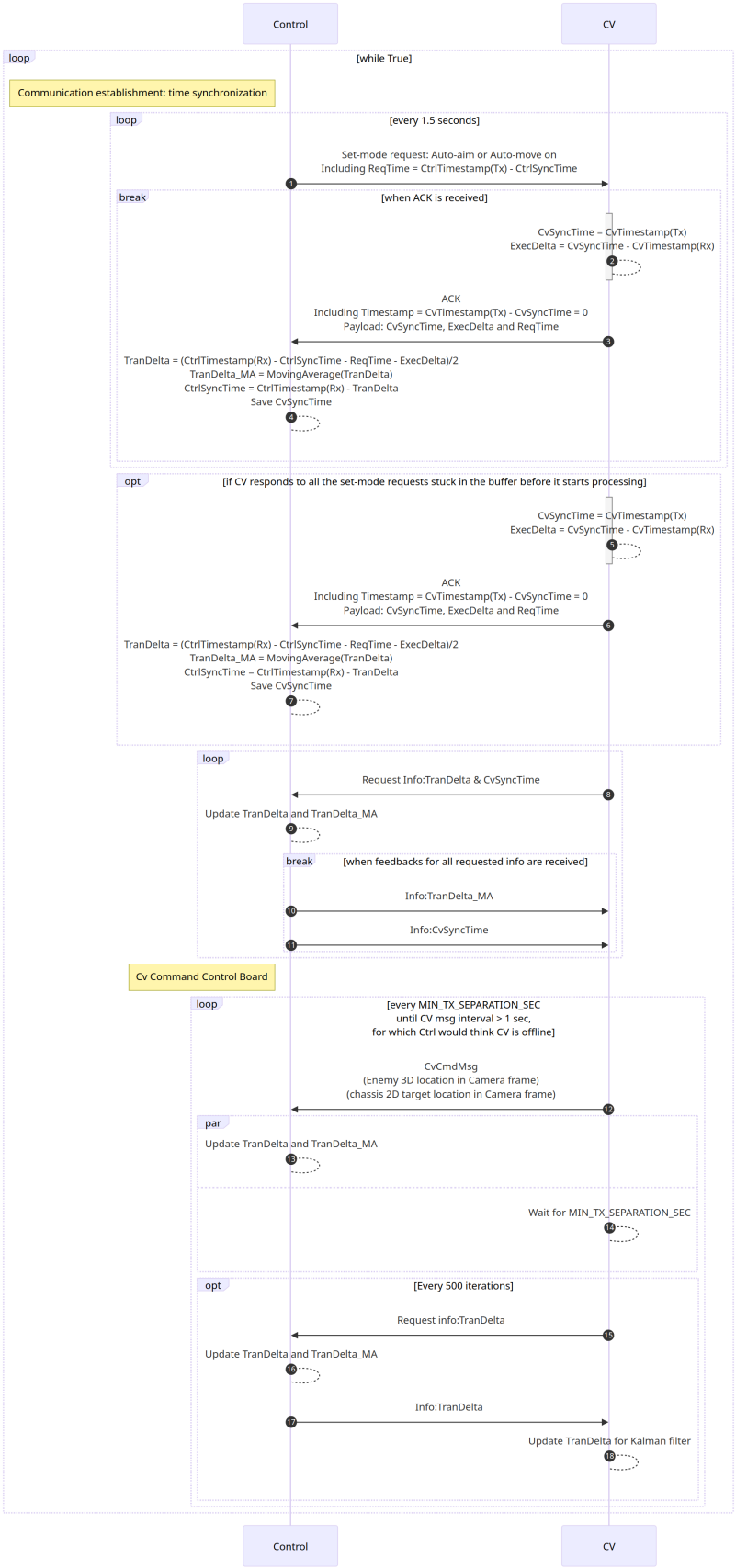

Figure 20: CV Interface General Sequence Diagram

Individual package format will be recorded in Team (2023), and it will not be included here for convenience. However, the general data structure is shown in table 35. To reduce processor work load of control board being interrupted at each incoming bit and the end of each packet, all packet types are designed to have the same size, and unused data field are to padded with 0xFF for package validation. With fixed package size, the control board can use DMA controller that processes the individual bit transmission in the background.

The timestamp field in each package is used to identify the unavoidable control loop delay, so that the control system can predict the intended target action based on the delayed and received command from CV.

Data Field	Data	Description
Header	0x3E, 0x3E	To align the start of data package.
Timestamp	Byte 0, Byte 1	Timestamp value in ms, with type uint16_t, LSB first.
Message type	Byte 0	Custom enumerated value less than 0xFF.
Message Payload	Bytes	Format of payload depends on message type.
Unused data field	Fill with 0xFF	Placeholder data for package validation.

Table 35: CV Interface Data Frame

5.8.2 Inputs

Table 36: Input Variables of CV Interface Module

Variable Name	Variable Type	Units	Range	Description
m_Time	Digital	s	TBD	Real-world time
CvRxData	Digital Array	N/A	N/A	Byte array of incoming data from CV

5.8.3 Outputs

Table 37: Output Variables of CV Interface Module

Variable Name	Variable Type	Units	Range	Description
y_TargetPosture	Digital	N/A	{e_Jump, e_Normal, TBD}	Commanded target posture state of robot
TargetOrientation	Digital Array	rad	TBD	Commanded target orientation of chassis
TargetSpeed	Digital	m/s	TBD	Commanded target movement speed of chassis

5.8.4 Exception Handling

Handling of all sequential exceptions are shown in Figure 20. For example, if one side of communication host does not receive within a certain time, it would reset or completely shut down the communication session.

If any one side of communication host received corrupted data package, keep trying to restart the session.

5.8.5 Timing Constraints

The intervals between subsequent CV commands are not fixed, but they should be fast enough that robot can avoid colliding into an obstacle from an arbitrary distance once it is detected. The least typical frequency is 10Hz.

5.8.6 Initialization

Reset all variables and buffers to zero.

5.9 User Interaction Module

5.9.1 User Interface : Python Application UI

5.9.1.1 Description

This is for the python application UI for delivery task scheduling. It is a simple module just to let sender schedule delivery and check current location.

5.9.1.2 Inputs

Table 38: Input Variables of Python Application UI

Variable Name	Variable Type	Units	Range	Description
c_CurrentLocationVisual	2D list	N/A	TBD	Current global location of WBR.
i_Destination	2D list	N/A	TBD	Final location that delivery need to go to.

5.9.1.3 Outputs

Table 39: Output Variables of Python Application UI

Variable Name	Variable Type	Units	Range	Description
c_OnDelivery	Boolean	N/A	TBD	Boolean to determine of delivery task is given or not.

5.9.1.4 Exception Handling

When user input any illegal input, system and UI will both prompt warning and stop from next action for the user.

5.9.1.5 Timing Constraints

Since this is an UI so as long as it looks smooth for the human eye. The time constraints is fine. For scheduling the time constraint is 10 secs.

5.9.1.6 Initialization

Set everything to 0

5.9.2 User Interface : Delivery Location Update

5.9.2.1 Description

This Delivery Location Update module update current location of WBR to visual aid which is our User Interface.

5.9.2.2 Inputs

Table 40: Input Variables of Delivery Location Update

Variable Name	Variable Type	Units	Range	Description
c_CurrentLocation	2D list	N/A	TBD	Current global location of WBR.

5.9.2.3 Outputs

Table 41: Output Variables of Delivery Location Update

Variable Name	Variable Type	Units	Range	Description
c_CurrentLocationVisual	2D list	N/A	TBD	point on map.

5.9.2.4 Exception Handling

If current location arrive late or just go missing. The system will keep WBR current visual location, the dot to last spot.

5.9.2.5 Timing Constraints
The time constraint for this module is 1s per update.

5.9.2.6 Initialization
Gather current location from BRW.

5.9.3 Notification : Email Notification

5.9.3.1 Description
This module publish the email notification to receiver the delivery has arrived and ready to pick up.

5.9.3.2 Inputs

Table 42: Input Variables of Email Notification

Variable Name	Variable Type	Units	Range	Description
c_DestinationArrived	Boolean	N/A	TBD	Boolean to determine if destination is arrived or not.

5.9.3.3 Outputs

Table 43: Output Variables of Email Notification

Variable Name	Variable Type	Units	Range	Description
c_EmailSent	Boolean	N/A	TBD	Boolean to determine if email is sent or not.

5.9.3.4 Exception Handling
For any reason if email notification can not be sent, it will wait for 10 mins and go back to original station.

5.9.3.5 Timing Constraints
For this module, the email need to arrive to target inbox in 1 min.

5.9.3.6 Initialization

Set email sent to False.

5.10 Connection between Requirement and Design

The designs are constructed to fulfill the requirements that are outlined in Software Requirement Specification Document. The table with requirements and their corresponding design actions are presented.

Design Actions	Requirements
Motion Control	RM1, RM3, PR1
Obstacle Detection	ES1, ES3
Computer Vision Integration	RM2, PR2
Terrain Sensing and Adaptation	ES2, SR3
Wireless Communication	OER2, MSR1
User Interface Design	UID1, UID2
Weather-Resistant Housing	OER2, PR4
Emergency Stop Mechanism	SR2, PR3
Object Recognition Algorithm	ES3, SR1
Upgradeable Software Architecture	PR3, MSR2

Table 44: Requirement Traceability Matrix

5.11 Module Traceability

Module	**Requirements**
Path Tracking and Planning Module	RM1, RM2, PR1, PR2, OER1
Environment Sensing Module	ES1, ES2, ES3, OER1
Motion Planner Module	RM3, PR4, SR3
Forward Kinematics Module	RM1
Jump Module	RM3, PR1
VMC Module	RM1
CV Interface Module	RM2
User Interaction Module	UID1, UID2

6 Timeline

Assigned Task	Designer	Deadline
ES3	Lisa Ji, Yuntian Wang, Zichun Yan	11.30
RM4	Haoyu Lin	12.31
RM1	Haoyu Lin	12.31
OER1	Lisa Ji, Yuntian Wang, Zichun Yan	12.31
ES1	Haoyu Lin	1.31
ES2	Haoyu Lin	1.31
PR2	Lisa Ji, Yuntian Wang, Zichun Yan	1.31
PR4	Lisa Ji, Yuntian Wang, Zichun Yan	1.31
CR1	Lisa Ji, Yuntian Wang, Zichun Yan	1.31
MSR1	Lisa Ji, Yuntian Wang, Zichun Yan	1.31
MSR2	Lisa Ji, Yuntian Wang, Zichun Yan	1.31
RM2	Haoyu Lin	2.29
RM3	Haoyu Lin	2.29
PR1	Lisa Ji, Yuntian Wang, Zichun Yan	2.29
SR2	Lisa Ji, Yuntian Wang, Zichun Yan	2.29
SR3	Lisa Ji, Yuntian Wang, Zichun Yan	3.31
UID1	Lisa Ji, Yuntian Wang, Zichun Yan	3.31
UID2	Lisa Ji, Yuntian Wang, Zichun Yan	3.31
SR1	Lisa Ji, Yuntian Wang, Zichun Yan	3.31
PR3	Lisa Ji, Yuntian Wang, Zichun Yan	3.31

7 Appendix A

7.1 Naming Conventions

The following naming conventions are observed in this document:

- k_ : constant value
- m_: monitored variable
- c_ : controlled variable
- e_ : enumerated values
- y_ : enumeration
- i_ : input variable (individual component)
- o_: output variable (individual component)
- d_: data variable (data in communications packet)
- s_: Data Structures
- t_: Data Types

The first letter of the constant shall be lower case, and all subsequent starting characters are upper case, for example, "k_SomeTextHere".
Previous values shall be represented by a subscript "-x" where x represents how far in the past, for example, "k_SomeTextHere-3".

7.2 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	unit	SI
m	length	metre
kg	mass	kilogram
s	time	second
°C	temperature	centigrade
J	energy	Joule
W	power	Watt ($\mathrm{W} = \mathrm{J\,s^{-1}}$)
m/s	speed	meter/second
m/s^2	acceleration	meter per second square
A h	electric charge	ampere hour
cm	length	centimeter
V	voltage	volt
N m	torque	newton per meter
rad	angle	radian

7.3 Abbreviations and Acronyms

symbol		description
CoM	=	Center of Mass
CV	=	Computer Vision
DJI	=	SZ DJI Technology Co., Ltd.
DMA	=	Direct Memory Access
FSM	=	Finite State Machine
IMU	=	Inertial Measurement Unit
IVP	=	Inverted Pendulum
LQR	=	Linear Quadratic Controller
LSB	=	Least Significant Bit
MacRM	=	MacRobomaster Club
N/A	=	Not Applicable
PID	=	Proportional–integral–derivative
RMUL	=	RoboMaster University League
TBD	=	To be declared
UART	=	Universal Asynchronous Receiver or Transmitter
VMC	=	Virtual Model Control
WBR	=	Wheeled Bipedal Robot

References

RoboMaster Organizing Committee. *RoboMaster 2024 University Series Robot Building Specifications Manual V1.0*. RoboMaster Organizing Committee, Shenzhen, China, 1st edition, October 2023. Available at https://terra-1-g.djicdn.com/b2a076471c6c4b72b574a977334d3e05/RM2024/RoboMaster%202024%20University%20Series%20Robot%20Building%20Specifications%20Manual%20V1.0%20(20231031).pdf.

Lisa Ji, Haoyu Lin, Yuntian Wang, and Zichun Yan. System requirements specification. https://github.com/macrobomastercontrolteam/Infantry_WheeledBiped/blob/main/Deliverables/SRS/SRS.pdf, 2023.

Xu Li, Yibo Yang, and Jianchun Wang. Active vehicle obstacle avoidance based on integrated horizontal and vertical control strategy. *Automatika*, 61:448–460, 07 2020. doi: 10.1080/00051144.2020.1778215.

Dingchuan Liu. Webots simulation of a wheeled bipedal robot using model based lqr. https://github.com/LiuDingchuan/Graduate_Project/tree/main, 2024.

J. Pratt, P. Dilworth, and G. Pratt. Virtual model control of a bipedal walking robot. In *Proceedings of International Conference on Robotics and Automation*, volume 1, pages 193–198 vol.1, 1997. doi: 10.1109/ROBOT.1997.620037.

Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer London, 2009. doi: 10.1007/978-1-84628-642-1.

MacRobomaster Control Team. Cv interface. https://github.com/macrobomastercontrolteam/CV_Interface, 2023.

Hongxi Wang. Robomaster wheeled bipedal robot control system design. https://zhuanlan.zhihu.com/p/563048952, 2022.

Shuai Wang, Leilei Cui, Jingfan Zhang, Jie Lai, Dongsheng Zhang, Ke Chen, Yu Zheng, Zhengyou Zhang, and Zhong-Ping Jiang. Balance control of a novel wheel-legged robot: Design and experiments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6782–6788, 2021. doi: 10.1109/ICRA48506.2021.9561579.

Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame*, pages 987 – 993, 06 2010. doi: 10.1109/ROBOT.2010.5509799.