Technical Design Paper Template for the Robotics Dojo Competition 2025

First A. Amos Oniare, Second B. Dismas Karimi, Third C. Gareth Kipkoech, and Fourth D. Fundi Brian (KNIGHTS)

I. INTRODUCTION

Robotics Dojo is a project-based robotics training program under the AFRICA-ai-JAPAN Project at the Jomo Kenyatta University of Agriculture and Technology with its primary objective being the encouragement of robotics research in Kenya. The program creates a robotics challenge each year in which students form teams and compete. The challenges are always evolving from previous years in an effort to build robots that can be beneficial in the real world environments such as industries.

This year's challenge was the design and development of an autonomous mobile robot that can collect and offload loads while navigating on uneven terrain, within a changing environment and use a camera for image detection and classification of potato diseases. The robot's performance is gauged on how accurately and fast it can perform navigation, image detection and collection and offloading of loads. This is an improvement of last year's challenge that only included autonomous navigation of a mobile robot.

II. DESIGN STRATEGY

The robot had to be able to do the following:

- Map of its environment and save the map.
- Autonomous navigation of its environment using the saved map.
- On board load transportation and dropping at a designated location.
- Image classification of healthy and diseased potatoes using a camera.

The above requirements were met by use of appropriate mechanical, electrical and software systems that worked together to accomplish a common goal. After careful consideration, the following elements that make up the robot were created:

- Robot frame
- Offloading mechanism
- Drivetrain and propulsion
- Power Supply
- Wheel mounts and wheels
- Control Scheme
- Environment detection devices
- Mapping, Localization and Navigation system
- Image detection system

Robot Frame

It represents the physical structure that supports all the components such as sensors, actuators, electronics and other mechanical elements of the robot. It was made from acrylic which is relatively simple to fabricate using a laser cutter and offers sufficient mechanical strength ensuring the robot has a rigid core that can sustain collisions without damage and can reduce vibration transmission that may influence the accuracy of sensor data.

Offloading mechanism

An arrangement was devised that could tilt the bed as well as open the gate using only one actuated link.The arrangement consisted of a bed-tilting mechanism and a gate actuation mechanism.

The bed tilting mechanism was a four-bar chain in double-rocker configuration. It is illustrated in figure A. The bed was the follower link, a servo arm being the crank and a coupler connecting rod.

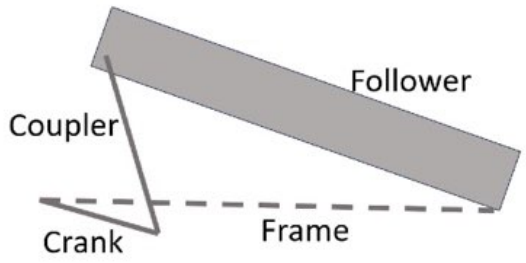


Figure A: Bed tilting mechanism

The gate forms part of a simple first class lever anchored on one side by a hinged link. It is illustrated in figure B. The pivot link and gate were rigidly connected, making them a single link.

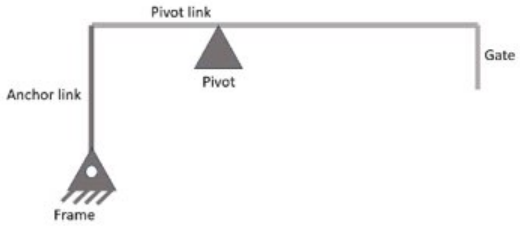


Figure B: Gate actuation mechanism

With this arrangement, it was possible to pivot the bed as well as raise the gate using only one actuated link.

Drivetrain and propulsion

This system was required to move the robot accurately and reliably on uneven terrain. It was made up of 4 motors with rotary encoders to provide odometry data and was powered by a lithium polymer (LiPo) battery via the drv8833 motor drivers. The number of motors was chosen to be four to increase traction in unstable surfaces and increase total propulsion power in moving the robot on inclined planes.

Wheel mounts and wheels.

The wheels chosen were of 85mm in diameter and with spikes for a firm grip to the ground. The wheels had a thickness of 36mm allowing bigger ground coverage for robot stability. These wheels were rotated using four motors which had maximum revolutions per minute (RPM) of 200. Hexagonal couplings were used to attach the wheels to the motors. The wheels-motors assembly was then mounted to the chassis of the robot using mounting brackets.

Control Scheme

This section was divided between the Mobile platform and navigation system of the robot. An STM32F4 microcontroller was used for the mobile platform because of its adequate input and output pins and relatively more interrupt pins compared to arduino mega. It was responsible for sending commands to the motors, reading encoder and IMU data and using a serial connection to send the data to the navigation system for processing.

The navigation system required a processor capable of hosting the Robot Operating System (ROS2) and the Raspberry Pi 4 was chosen because of its sufficient processing speed and relatively low power requirements. Its function was to receive data from the STM32F4 and Lidar, process them and send appropriate motor commands to the STM32F4.

Environment detection devices

The sensors used for the robot setup were quadrature encoders, MPU-6050, RPLIDAR A1 M8 and the PiCamera module 2. The encoders were used to track wheel rotation. This information would assist the robot to calculate its estimated position and trajectory. The MPU-6050 was used to measure the robot's orientation and movement by integrating a gyroscope and an accelerometer. To mitigate the odometry drift the encoder data and inertial measurement unit (IMU) data were combined using sensor fusion. The LIDAR was used to detect objects within the environment of the robot using an infrared (IR) laser scan. This data was used for simultaneous localization and mapping (SLAM) of the robot as well as during navigation to allow the robot to detect the introduction of new obstacles into its environment. The PiCamera was used to capture images of plant leaves displayed on a screen for classification using a machine learning model. The possible categories were "healthy", "early blight" and "late blight". The PiCamera color detection on an LED strip mounted in the game field for distinguishing payloads.

Power Supply

A Lithium Polymer battery was chosen to power the Mobile platform due to its high energy density and relatively light weight. The relevant power requirements for each electrical device were calculated and used in determining the appropriate charge capacity of the battery. Switches were also used to disconnect the power source from the electronic components when the robot was not in use to reduce the likelihood of accidents caused by short circuits occurring. The navigation system was powered by an 18 W power bank that was connected to the Raspberry Pi which then powers the lidar.

Mapping, Localization and Navigation System

- Mapping [1]. This was possible by using the laser scan from the LIDAR to visualize the robot's environment. This was carried out using the slam toolbox and a file containing the mapping parameters such as scanning range, resolution, maximum loop distance among others needed to generate a clear map that the robot can later refer to. It was possible to create a good map because of utilizing the Extended Kalman Filter (EKF) sensor fusion algorithm to combine the IMU data with encoder data for accurate and reliable estimate of the robot's position and velocity. Therefore, even if the actual robot skidded then in RVIZ2, the skidding would not be registered as a robot's movement. The map was visualized in RVIZ2 and later, if satisfied with its clarity, saved and serialized for use in localization.
- Localization. For the robot to understand its location with reference to its surroundings then the saved map was used by loading it as a parameter during localization. Upon starting the localization launch file, the earlier created map would be loaded on to RVIZ2 and if not satisfied with the robot's analysis of its current location, then a pose estimate would be provided.
- Navigation. After localization, the robot was now ready to autonomously navigate through set way points or to just one set goal. With the assistance of the nav2 stack with tuned parameters different from the default, the local and global costmaps showed the robot's weight allocation to different obstacles in the environment and thus being able to

find an obstacle free path however long or short. A notable implementation is adoption of the Model Predictive Path Integral (MPPI) controller instead of the default Dynamic Window Approach (DWA) critic based controller after the former outshined the latter in multiple simulation runs.

The general steps taken in the design and development of the robot were as follows:

- A model and design of the robot frame and loading and offloading system was created using a CAD (Computer Aided Drawing) software while ensuring the robot dimensions were within the allowed limits i.e, 300 x 300 x 200 mm.
- Acrylic was chosen as the fabrication material and a laser cutter was used to cut the desired shapes making up the robot structure.
- The robot chassis was assembled using appropriate screws and hot glue where necessary.
- A Bill Of Materials (BOM) was created for purchasing electrical components and other consumables.
- An electrical circuit was designed and transferred onto a perforated board by soldering the electrical devices and connection header pins in place.
- The electrical and electromechanical components were connected to the circuit.
- Testing was done on the Mobile platform to ensure it was following instructions from a written algorithm.
- The sensing elements were added to the circuit and the Raspberry pi, connected to the lidar, was connected via serial to the STM32F4.
- Image detection and classification was implemented using the Raspberry Pi Camera module 2.
- Image data was used to coordinate the loading and offloading process.
- Mapping and Navigation was then developed, tested and improved.

III. VEHICLE DESIGN

This section discusses the design process in detail and explains the methodologies used during the fabrication, assembly and programming of the robot. The vehicle design was split in three: the mobile platform design, navigation system design and the computer vision system design. Improvements from last year's robot design was also noted and discussed.

1. *Mobile Platform*

This consists of the drivetrain and the loading and offloading system. Figure 1 shows the robot design that was created.

The 2D dimensioned views of the robot's frame are shown in Figure 1.

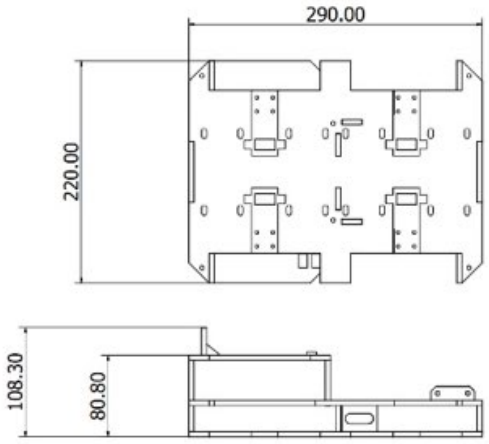


Figure 1: 2D dimensioned robot frame

The 2D dimensioned views of the motor holder and coupling assembly is shown in Figure 2.

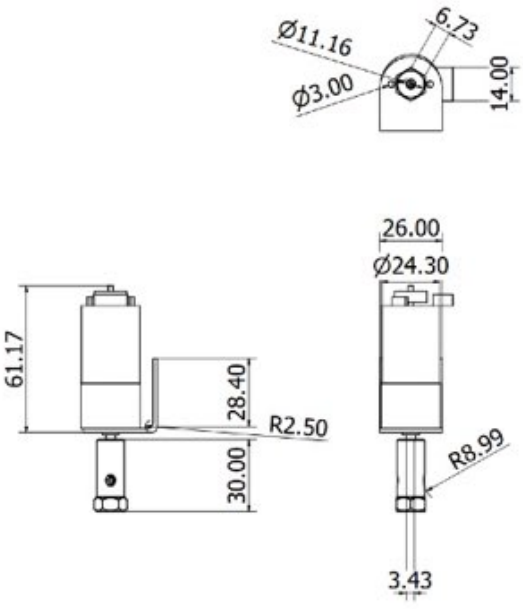


Figure 2: 2D dimensioned motor-coupler assembly

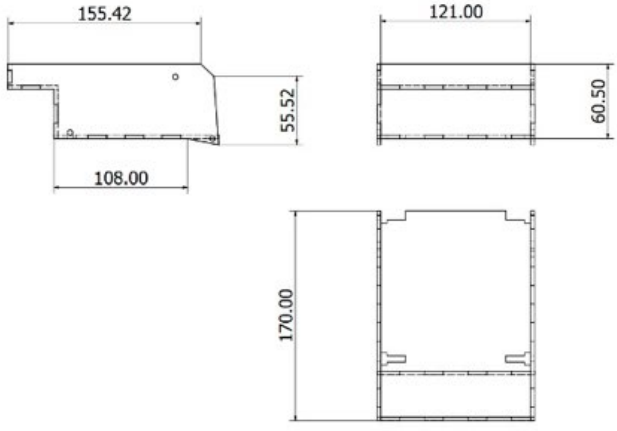


Figure 3: 2D dimensioned courier/bed

The 2D dimensioned views of the robot's bed used as courier for the colored cubes are shown in Figure 3.
A tagged 3D robot model is shown in Figure 4. It gives an overview of the outlook of the designed robot in Autodesk Inventor.

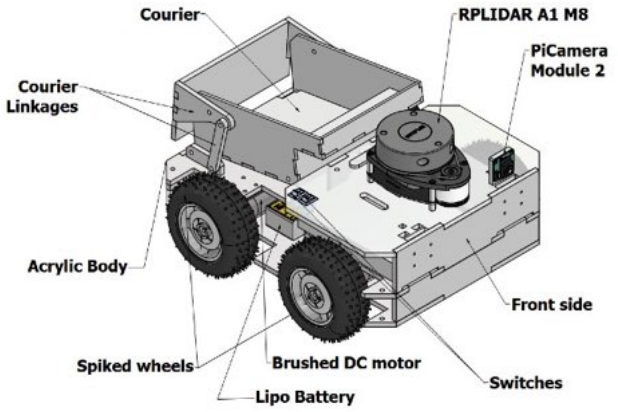


Figure 4: Tagged 3D robot model

The robot consists of 2 acrylic layers at its base, 3 layers for its front half and a loading and offloading mechanism at its rear half. Acrylic was chosen as the fabrication material because of the following:

- High impact resistance which is 10 times higher than glass.
- It is Lightweight.
- It has excellent dimensional stability.
- Innate weatherability and UV resistance.
- High optical clarity. (Makes it easier to identify problems in electrical connections within the body)
- Requires less energy when laser cutting as compared to other materials like aluminum.
- Its lower price compared to other durable alternatives like metals.

Laser cutting was chosen as the preferred method of manufacturing instead of other methods such as 3D printing because of the following advantages:

- Low lead time in fabrication
- High precision geometries and low tolerances.

The robot was designed to have a low height and to accommodate the heavy components in the lowest layer, with the aim of minimizing the height of the centre of mass.

Four motor driven wheels were chosen to allow the robot to have greater maneuverability and propulsion on uneven and unstable surfaces such as gravel and woodchips. The downside of using four wheels is that extra developments were needed to account for drift of the robot's odometry during turning on the spot using a differential drive system. This error necessitated incorporation of an IMU and correction algorithms. The implementation of this correction system is discussed in depth in the navigation system.

Once the robot frame design was completed, its electrical system was tackled, involving proper component selection and integration. Power calculations were performed for power supply sizing. The motor selection was informed by target robot acceleration.

The final robot was estimated to weigh approximately 4kg. A target maximum acceleration of $0.5m/s^2$ was selected. Given the above two considerations and assuming a worst case 50% efficiency, the required motor torque is given by:
$T = \frac{a \times m \times r}{N \times \eta} = \frac{0.5 \times 4 \times 0.0425}{4 \times 0.5} = 0.0425\,Nm$, where a is the target acceleration, m mass of the robot, r radius of the drive wheels and N the count of driven wheels
A target speed of $0.8m/s$ is selected. Given the wheel radius, the appropriate motor rotation count is given by:
$RPM = \frac{60v}{2 \times \pi \times r} = \frac{60 \times 0.8}{2 \times \pi \times 0.0425} = 179.751rpm$, where v is the target speed, and r is the driven wheel radius.

Typical current draw is calculated with the relation:
$I = \frac{T \times \omega}{V} = \frac{0.0425 \times 2\pi \times 179.751 \div 60}{10.5} = 0.0762A$, where T is the required torque, ω is the required angular velocity and V is the supply voltage.

The motors best matching the requirements above are found to be 12V, 200rpm with encoders.

The other components that make up the mobile platform include:

- DRV8833 - A Dual H-Bridge Motor Driver with 1.5 A peak current output per channel and under-voltage, over-current, and over-temperature protection.
- Hex Coupling - Used for connecting the motor shaft to the wheels.
- STM32F4 microcontroller - Used for low-level tasks like controlling motors, fetching data from encoders and IMU.
- 4 brushed DC motors with quadrature encoders- For propulsion.
- MG996R Servo Motor - Used for loading bed actuation.
- Lipo battery - Providing electrical power to the system.
- Perforated board - For mounting and soldering electronic components.
- Power Switch - Allows for the isolation of the electrical circuit from the power source when the robot is not in use.
- Buck converter - Steps down DC voltage from battery.
- Japan Solderless Terminals (JST) - Used for component connection to the main board.
- Male and female header rows - Used for modules and cable connection.
- Cable ties - For bundling and securing wires and electronics.
- Motor mounting brackets - For securing the motors to the robot frame

Once the components were selected and a circuit made, power calculations were applied to determine the correct rating of the Lipo battery and verify if the connections were safe and functional.

$P = IV\ (Watts)$ equation 'I'

Where; 'P' is Power in Watts
'I' is Current in Amperes
'V' is Voltage in Volts

The following chart is the representation of the electrical circuit with the current and voltage values of each component. Devices operating on the same voltage value were connected on the same voltage line and their current draws were summed as shown in Figure 5.

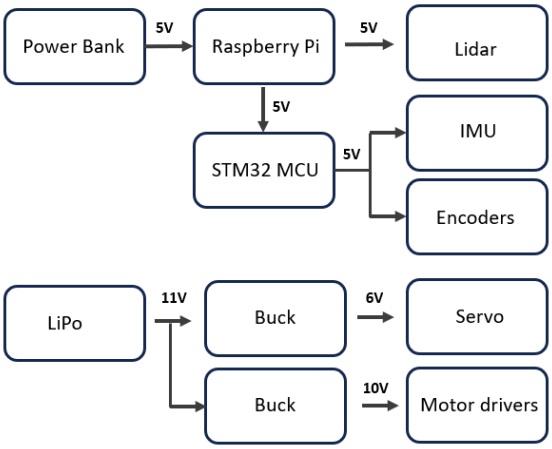


Figure 5: Electrical flow diagram

Two power sources were used, one for powering the Raspberry Pi and the other the motors. It was more convenient to use a power bank with a type C cable to power the raspberry pi. To determine if the Power Bank was capable of providing enough power, the total power drawn by the Raspberry Pi and the RPLidar was calculated as follows:

P = (1.5 x 5) + (1 x 5)= 12.5W

This value is well below the rated value of the power bank. The capacity of the power source for the rest of the components was determined by calculating the power requirements of each device and finding their sum.

Power from Lipo ≈ Peak power required by (1 Servo motor and 4 DC motors and encoders)

P ≈ (2.5 x 6) + 4(1.5 x 10) = 75 W

Therefore the 3S 2200mAh Lipo battery that was chosen had a Power rating of 24.4 Wh which can supply the peak current required by the system at 11 V.

Closed loop control of the motors was achieved by implementing the PID (Proportional Integral Derivative) control which is made up of the following elements:

- Proportional (P) Coefficient Kp - Its effect increases proportionally with magnitude of error. It increases responsiveness of the system but cannot effectively eliminate steady-state error. It was decided after testing to assign it a value of 200.
- Integral (I) Coefficient Ki - Its effect grows as the sum of errors over time increases. It therefore controls steady-state error. It was assigned a small value of 1, as much larger values increase system instability.
- Derivative (D) Coefficient Kd- Its effect increases when rate of change of error increases. It therefore controls overshoot by slowing down change as the final value is approached This coefficient was set to the value 120.
- Scale Ko - It scales the output of the controller by a factor. The value found appropriate was 50.

Challenges in Hardware Implementation

1. USB device enumeration failure - attempts to program the microcontroller over USB were initially unsuccessful. The STM32F4 would not be recognized as a USB device by the computer, regardless of the Operating System used. It was discovered that some Input-Output (IO) pins are inappropriate for use with the USB bootloader. The USB FS D+ and USB FS D- pins in particular had the greatest effect. Figure (i) shows the STM32F401 pinout and the affected pins.

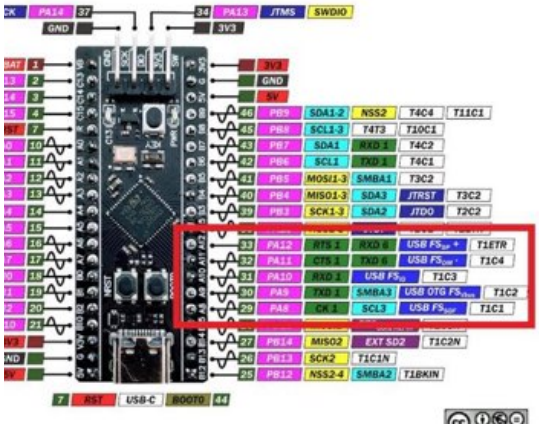


Figure (i): STM32F401 pinout with affected pins highlighted

The solution was to move IO devices from those pins to unaffected pins.

2. Motor jerk when running closed loop motor control - open loop motor control performed as expected, and reading encoder data alone by spinning the motors by hand was okay as well. However, activating both motors and encoder reading produced erratic behaviour. After extensive research and testing, it was discovered that motor wire length and shape was the contributor. To keep wiring tidy, excess motor and encoder cables had been coiled into loops. This unintentionally created strong electromagnetic coupling between the noisy motor power lines and the sensitive encoder signals. The result was EMI from inductive, capacitive, and antenna-like

effects, which corrupted encoder readings. As a workaround, motors had to be driven at a low PWM frequency of 500 Hz and lower. The figures (ii) and (iii) illustrate the cable management.

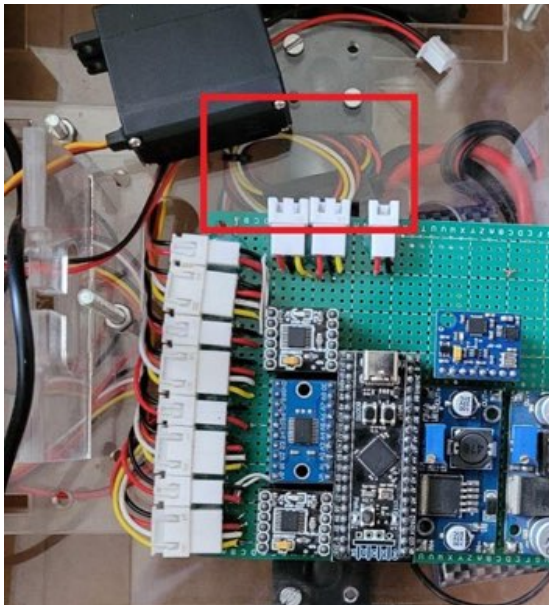


Figure (i): Wire loops present before.

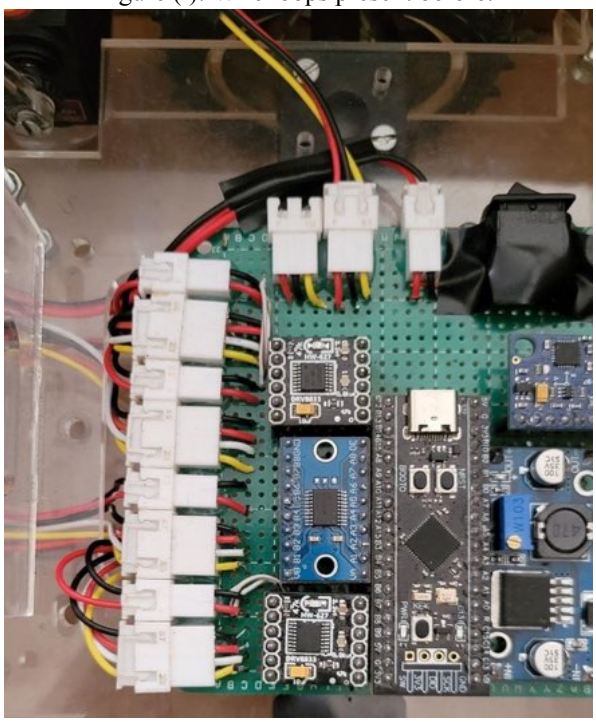


Figure (ii) Wire loops removed.

After removing the loops and shortening the cables, the interference was eliminated, and clean encoder signals were maintained even with PWM frequencies above 30 kHz.

2. *Navigation System*

In the design of the robot navigation system, multiple critical steps were covered, including data conversion from encoders and IMU, SLAM mapping, and path planning. A comprehensive robot navigation system framework to realize the navigation capability was designed. This framework implemented distributed communication through the ROS system, enabling collaboration between SLAM mapping and navigation path planning. A high-level overview of the system is shown in Figure 6.

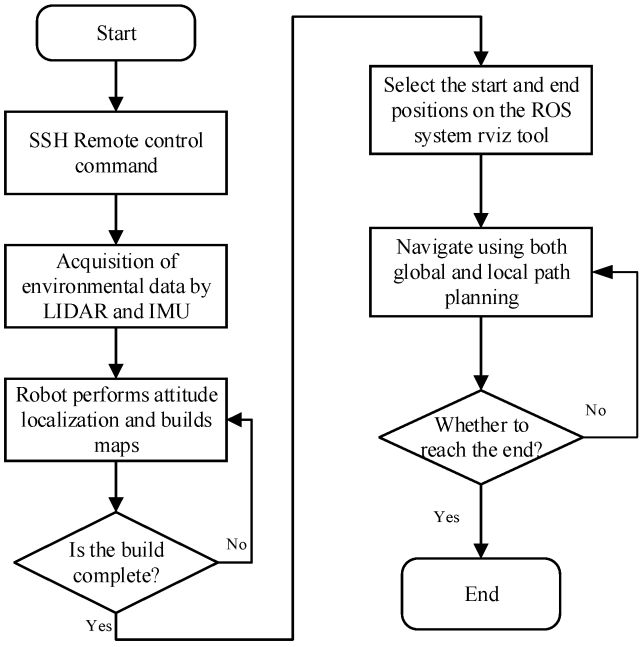


Figure 6: Navigation System Flow Chart

In the previous Robotics Dojo competition, encoder data was the only source of odometry data for the robot. The position of the robot was calculated based on counts of wheel rotations and the differential robot motion model. This is a simple method for robot localization but it only works accurately when the robot's wheels do not slip. This year's competition features uneven terrain and loose surfaces making it increasingly difficult to achieve motion without any slip, which throws off the localization. To solve this problem, an Inertial Measurement Unit (IMU) was introduced to complement the encoders. Data from both streams was used together to provide accurate and reliable robot localization even with wheel slip. This method of using data from more than one sensor for a single purpose is known as sensor fusion and it has the following advantages:

- Enhances accuracy of resultant data.
- Increases robustness of the system as it will continue functioning even if one sensor fails.
- Provides resilience against intermittent/missing data

The Extended Kalman Filter (EKF)[2] was used to make the system resilient to sensor noise for steady more meaningful readings. EKF is a development on the Kalman Filter which allows it to process non-linear functions of a system, better suited for real world applications. The basic principle of operation of this filter is as illustrated in figure 7:

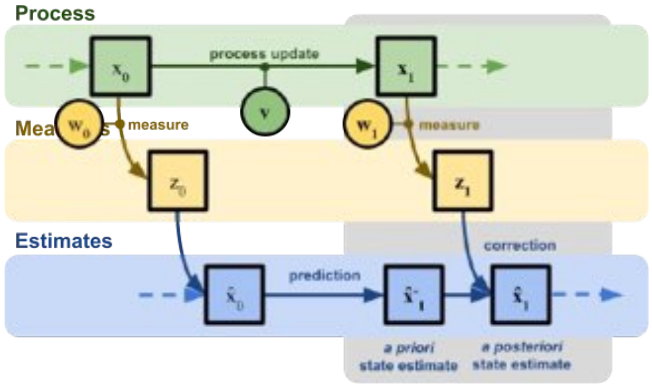


Figure 7: Extended Kalman Filter Illustration

The process represents the system being monitored such as a moving body, the measurements stand for the data collected from sensors and estimates are the results from the filter's model of the system. From a high-level perspective, the algorithm follows a 2-step cycle: (1) Predict, (2) Update.

In the prediction step, the system's state is projected forward using a non-linear motion model, creating the priori state estimate. In the update step, incoming sensor measurements are compared to the predicted state, and the state estimate is corrected to create the posteriori state estimate.

Using the XYZ coordinate frame with Z pointing up from the floor, the robot's state (position and orientation) can be described by just 3 values: X position, Y position and Yaw angle (rotation angle about the Z axis).

The state estimate is a function of the previous state, the previous velocities and some noise term. Mathematically stated[3]:

$$state_t = state_{t-1} + dt \times velocities_{t-1} + noise$$

In matrix form, and substituting the differential drive model we get:

$$\begin{bmatrix} x_t \\ y_t \\ \gamma_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \gamma_{t-1} \end{bmatrix} + \begin{bmatrix} \cos(\gamma_{t-1}) & 0 \\ \sin(\gamma_{t-1}) & 0 \\ 0 & dt \end{bmatrix} \begin{bmatrix} v_{t-1} \\ \omega_{t-1} \end{bmatrix} +$$

$state_t$ provides the current state estimate.

Now the predicted sensor measurements is given by[3]:

$$\begin{bmatrix} y_t^1 \\ y_t^2 \\ . \\ . \\ y_t^n \end{bmatrix} = H_{nx3}^t \begin{bmatrix} x_t \\ y_t \\ \gamma_t \end{bmatrix} + \begin{bmatrix} \omega_1 \\ \omega_2 \\ . \\ . \\ \omega_n \end{bmatrix}$$

Where H is the measurement matrix, n is the number of sensor measurements at time t, and ω is the sensor noise term.

Once the estimated state and predicted sensor measurements at current time t are obtained, a weighted average is calculated to create an improved current state estimate. The successive steps in the EKF algorithm automatically determine the appropriate weights.

Robot_localization[4] a ROS2 package makes implementation of EKF in a project simple. The required inputs are the wheel encoder data, the IMU data, process noise covariances and velocity commands. The filter is applied to all sensor inputs before being combined to obtain a single reliable odometry output. The computation is done by a node named /ekf_node, which is subscribed to the /imu/data and the /diff_cont/odom topics. The node then publishes processed odometry data to the /odometry/filtered and /tf topics. The former is the current state estimate of the robot and the latter is the current transform of the robot base link.

Other than noise reduction, the EKF robustness against gaps in data collection due to, say, sensor interruption. This is because the prediction step, which utilizes the differential drive robot model, can extrapolate into the future. However, prediction for extensive durations leads to error accumulation, since sensor data, although noisy, still contains some information. This information is crucial for correction where predicted data is compared with noisy data to obtain a final state estimate that is both accurate and low in noise.

Robot navigation was achieved using the ROS2 Nav2 stack for path planning and control while slam toolbox was employed for SLAM.

3. *Computer Vision System*

This section was implemented using the Raspberry Pi Camera Module 2 which was the image source. The image data collected had the following functions:

- Classification of potato leaves as being healthy, having early blight or late blight.
- Coordinated the loading and offloading process

Classification of potato leaves was achieved using a package provided by Robotics Dojo with a trained model. The package creates a node subscribed to the /image topic and an inference engine whose result is annotated onto the source image. The labelled image is then published to the /inference_image topic which can be visualized in RVIZ under the Image display plugin.

The goal for Robotics Dojo was to make loading and offloading processes fully automated. The robot would have to discriminate loads based on color and deliver the load to a bay of matching color. During loading, two LED strips were placed in front of the loading bay to give a signal on the color of the load the robot has received such as lighting the red led strip when the load is of a red color. The robot then identifies and saves the led color, proceeds to navigate autonomously to the offloading bay and once again uses the camera to identify the bay with a matching color as the load it is carrying such as offload on the red bay if the load carried is red.

A simple workflow was implemented using the OpenCV[5] module in Python. OpenCV allows detection of

entities in an image such as color, shapes, faces and many more. The camera was run continuously to fetch real-time images. Using the Real-Time Streaming Protocol (RTSP), the images were sent over IP from the Raspberry Pi to the main computer for processing. This was adopted to relieve the raspberry pi off the heavy inference processing duties. The algorithm running on the main computer for simple real-time color detection was as follows:

1. An image is fetched from the RTSP stream and saved to memory
2. The image is converted from BGR to HSV color spaces
3. Color ranges in HSV format are defined
4. Using the color ranges, a mask is created for each color
5. The mask is dilated (a morphological transform) to reduce effects of image noise
6. Contours in the masked image are identified
7. Bounding rectangles corresponding to the contours are drawn on the image for tracking
8. Bounding rectangles are counted
9. Text is affixed for each bounding rectangle
10. The color with the largest count of bounding rectangles is published to a ROS topic /detected_color

Simple conditional checks were created to check for specific sequences in detected color. Servo actuation would be done only if a color was detected for 2 consecutive detection events after a time interval.

Challenges in Implementing Color Detection

An initial implementation of the algorithm had step 8 identify the largest contour by area. While initially promising it failed when the target color did not cover the full Field-of-View of the camera. The simple algorithm would, for instance, identify the color of the frame of the screen as well as the color shown on the screen itself. Of course this gave erroneous results that broke the algorithm downstream. However, it was noticed that for a given target color, many contours would be identified within the frame, along with one large contour for the frame on the screen. The team decided to shift strategy to counting bounding boxes, rather than ranking them based on area.

IV. EXPERIMENTAL RESULTS

The performance of the robot was tested and assessed in the following areas:

- Quality of map produced.
- Accuracy of colour detection.
- Coordination of the courier offloading based on image data.

The quality of the map produced was greatly dependent on the source of odometry data. A comparison was made on the maps obtained by using encoder data only and when using sensor fusion to combine encoder and IMU data. Two simple tests were done in simulation:

1. Turning in place 90 degrees clockwise from the initial position
2. Driving around a complete loop from the initial position

Figures 8 and 9 show the initial position of the robot at launch of SLAM .

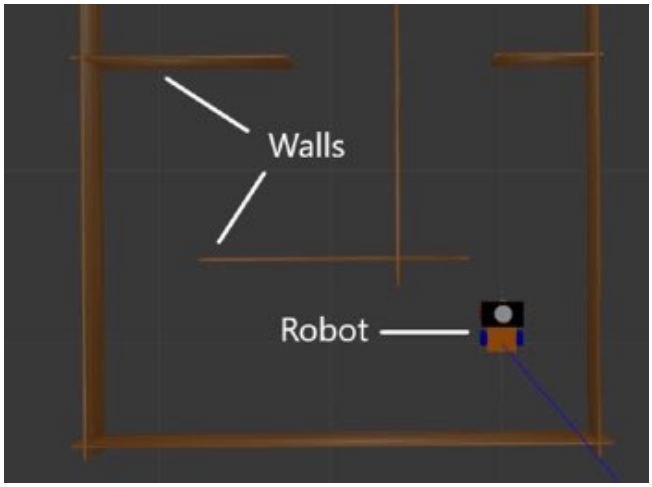


Figure 8: Virtual robot initial position

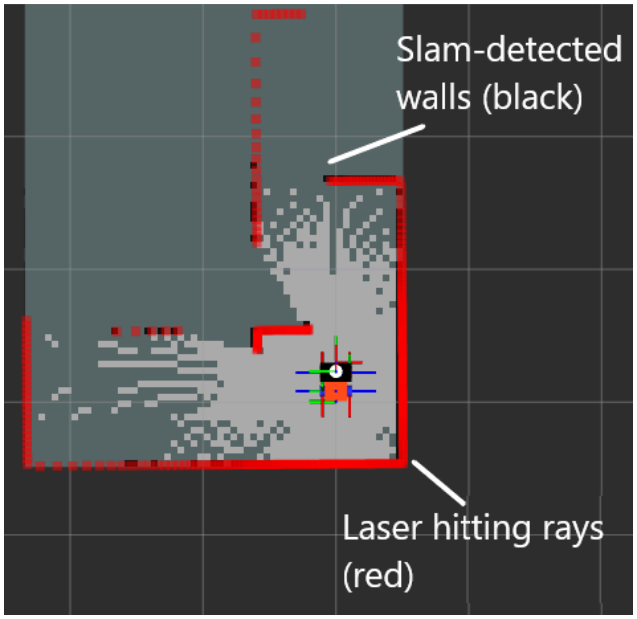


Figure 9: Rviz visualization at initial position

Figure 10 shows the result of turning in place without fused odometry.

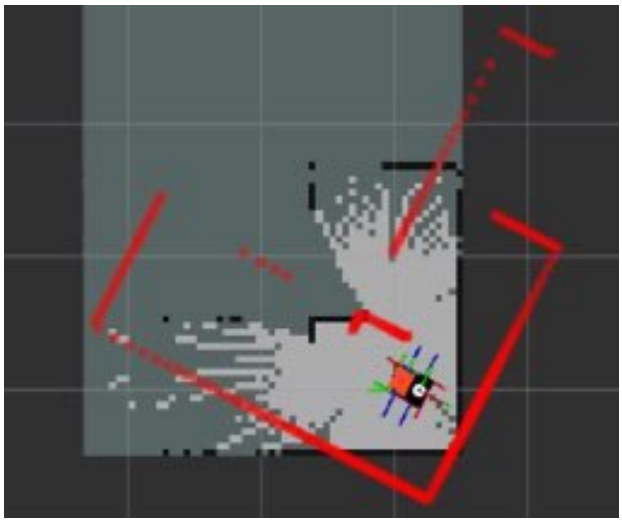


Figure 10: In-place turning without fused odometry

Figure 11 shows the result of turning in place with fused odometry

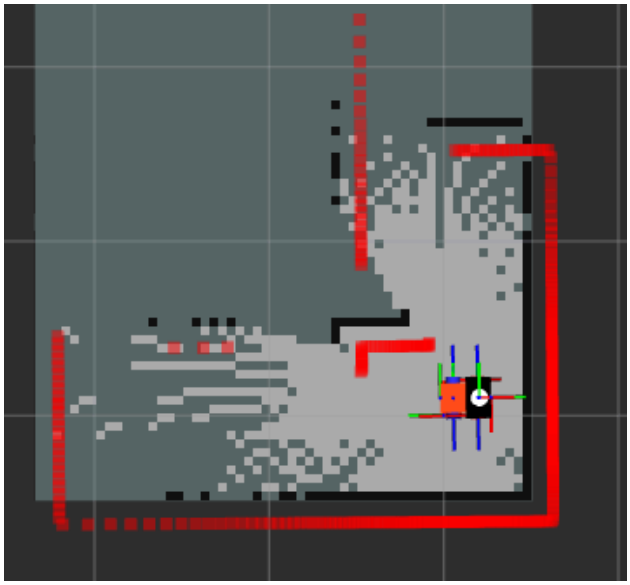


Figure 11: In-place turning with fused odometry

Figure 12 shows the generated map without fused odometry.

Figure 12: Map without fused odometry

Figure 13 shows the generated map with fused odometry.

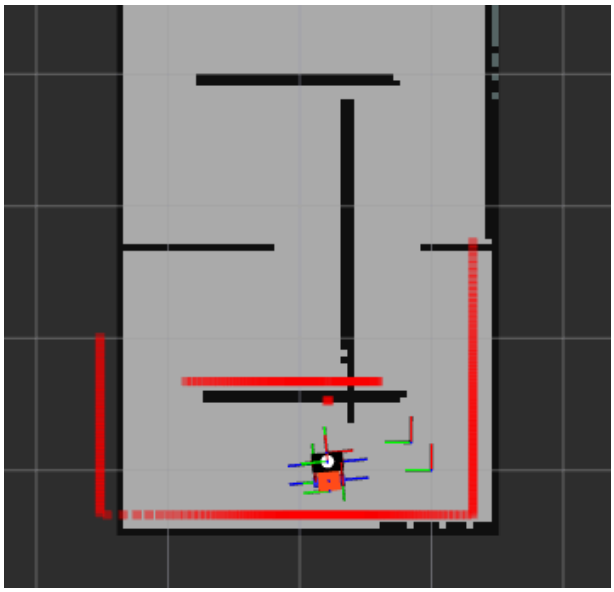


Figure 13: Map with fused odometry

Figure 14 shows the generated map without fused odometry with a modified driving style

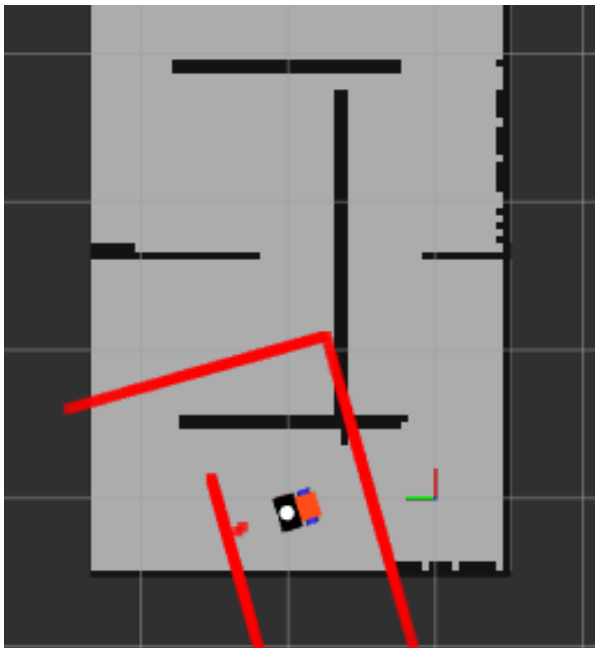
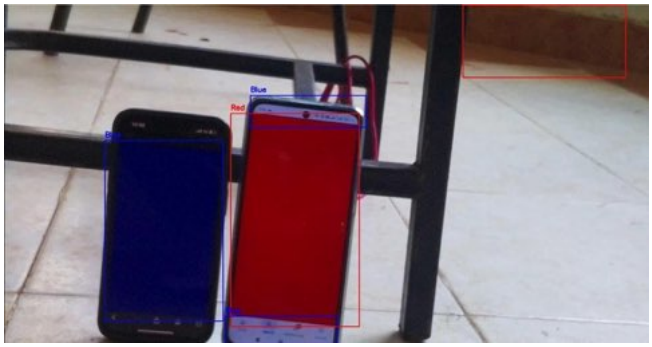


Figure 14: Map without fused odometry with modified driving style

It is quite clear from figures 10 and 11 that sensor fusion performs considerably better than odometry with wheel encoders alone. There still exists some translational drift in laser scan in figure 11 however in practice this is easier to compensate for than rotational drift. Figures 12 and 13 illustrate that EKF is indispensable as the map without fused odometry is unusable, and further cement the difference in significance between translational and rotational drift. An interesting finding shown in figure 14 was that the driving style also played a role. To obtain that result, in-place turning was avoided completely, instead opting to combine some linear velocity with all angular velocity. However, the alignment of the laser frame was still compromised at the tail end of the run when restoring the robot's orientation. This proves that although it is possible to map accurately without IMU data, it is not dependable, especially since autonomous robot navigation extensively uses in-place turning to clear up laser scan drift.

The results from the real robot closely resembled those obtained from simulation.

The Courier was designed to offload only when the robot had arrived at the offloading bay that was colored to match the color of the object loaded on the robot. OpenCV was used as a tool to classify image data into a specific range of colors such as red or blue. Additional functionality was added to this image controlled loading and offloading process to ensure that it functions as designed. OpenCV allows one to write a python script that displays an image from the pi camera with labelled rectangular contours encircling an identified image such as a color or an object. Two methods of color identification were implemented and compared. The first was selecting the largest contour by area and publishing the identified color as the label of the contour and the second was selecting the label with the most contours and publishing the label as the identified color. The described methods are illustrated in the following image:



From the image, it is observed that the red contours are two in number while the blue contours are three in number while the largest contour by area is the red contour, only if the larger red contour in the background is ignored.

As discussed in the Vehicle Design section, the latter method was chosen because of the following reasons:

- It was less prone to error. The first method could identify a contour larger than the object being observed in the background and send the wrong data. The second method would rely on contour concentration, such as an led, and send the led color as the correct color of the image.
- It was more reliable. An object with a consistent and bright color will trigger the correct color identification without being affected by background color identifications that may not have high intensity.

V. ACKNOWLEDGEMENT

The Robotics dojo competition development and implementation was greatly dependent on the 2025 Robotics dojo interns who created comprehensive video and written tutorials, as well as presentations on various topics. Acquisition of robotic parts was also made possible by the Japan International Cooperation Agency (JICA).

VI. REFERENCES

[1] E. Renard, ROS 2 from Scratch: Get Started with ROS 2 and Create Robotics Applications with Python and C++.* Birmingham, U.K.: Packt Publishing, 2024, pp. 343–344.

[2] Ribeiro, M. I. I. (2004). Kalman and extended Kalman filters: Concept, derivation and properties. *Institute for Systems and Robotics*, *43*(46), 3736-3741.

[3] AutomaticAdisson, Extended Kalman Filter with Python, 2025, Online https://automaticaddison.com/extended-kalman-filter-ekf-w

ith-python-code-example/.
[4] Tom Moore, Robot Localization Wiki, Online Article, 2024, https://docs.ros.org/en/melodic/api/robot_localization/html/index.html
[5] Raguraman, P., Meghana, A., Navya, Y., Karishma, S. K., & Iswarya, S. (2021). Color detection of RGB images using Python and OpenCv. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *7*(1), 109-112.

VII. APPENDIX–SITUATIONAL AWARENESS

A significant challenge in adoption of unmanned systems is user trust. Many applications to which robots may be of great benefit remain unautomated simply because of lack of trust. This highlights the need for communication between the designer and the ultimate consumer of the system. Therefore to bridge that gap, this team would do the following:

- Inform the user that the system is autonomous, what autonomy means and what they should expect.
- Explain the most relevant constituents of the system, especially components visible to the user.
- Communicate what the robot is designed to do, and what role the user plays in its successful operation.
- Discuss common misconceptions and Frequently Asked Questions about what the robot can and can't do.
- Cite internal and/or external testing and validation data that shows the product can complete the task, also listing confidence levels and uncertainties.
- List fault conditions, how to identify them, and resolve them if possible.
- Communicate that corner cases do exist and may not be accounted for.
- Emphasize that the robot is not sentient or innately malicious.