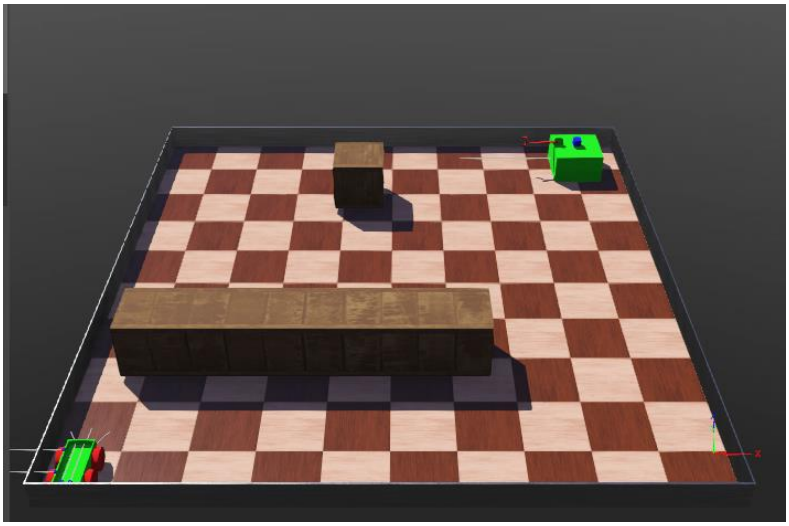
Please Edit this page with your details

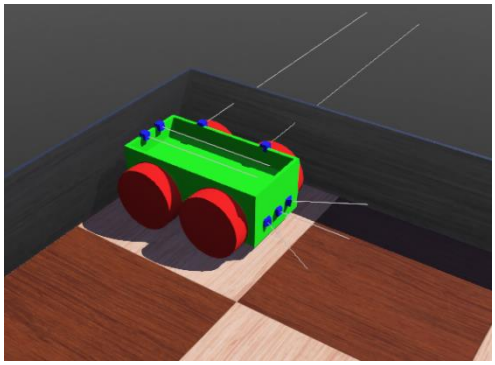
Introduction



This robot system simulated in this project is an automated robot system which the mobile robot can go to robot manipulator avoiding obstacles and fetch the package to required destination while avoiding obstacles. Mobile robot is a four wheeled differential drive robot and manipulator is a 3 dof (degree of freedom) immobile robot manipulator. Both robot controllers are written in MATLAB in SPA (Sense Plan Act) architecture. For avoiding obstacles Bug algorithm is implemented.

To test the automated system, above shown arena was created with a large obstacle and a small obstacle.

Mobile Robot



Following picture shows the mobile robot system designed in Webots simulator. To keep the box on top of robot without falling while moving, a wall was added around the top perimeter. Height of this wall is 0.05m (5cm). All the other design dimensions are as specified.

Sensors used and their application

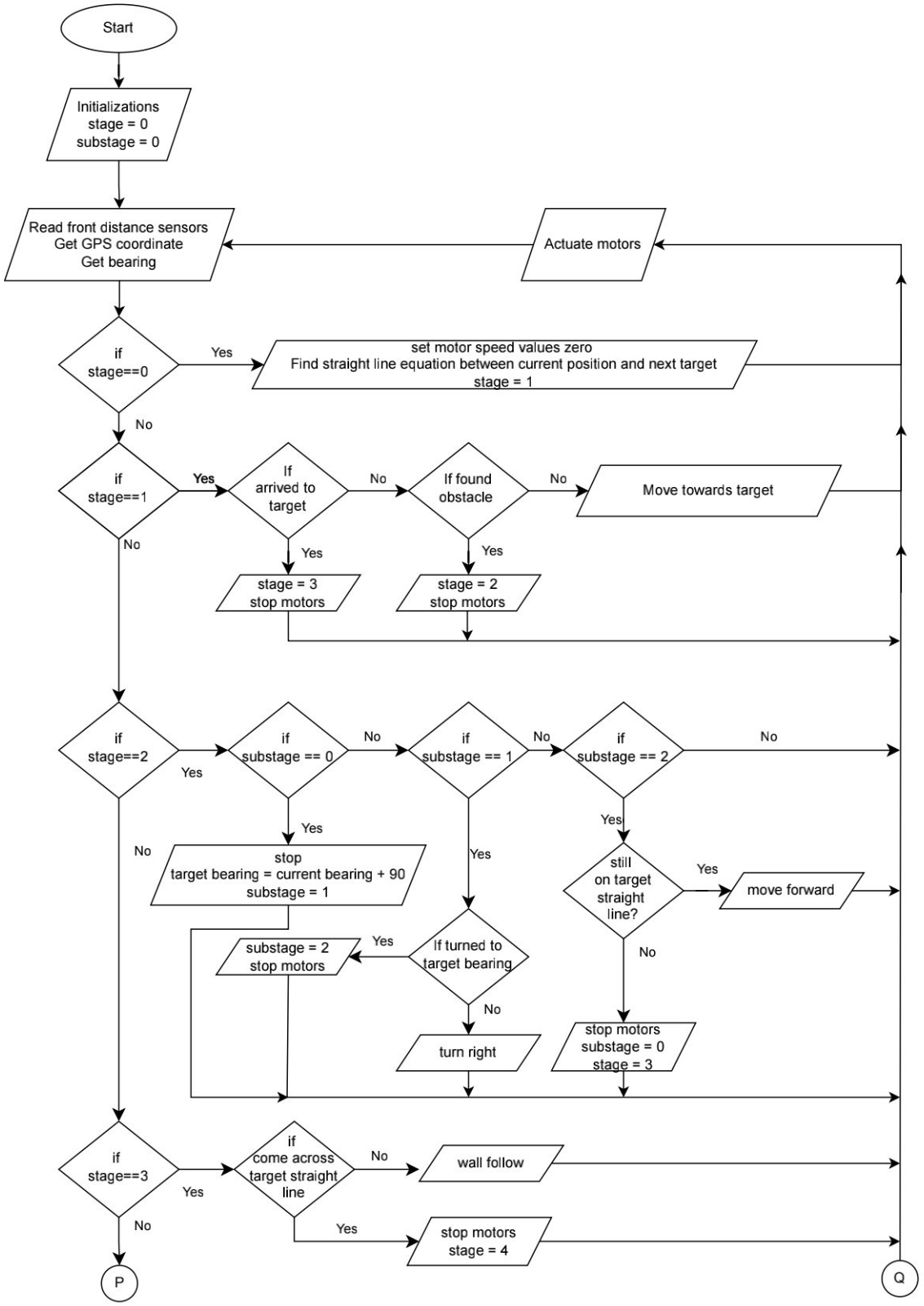
Following table shows the application of sensors.

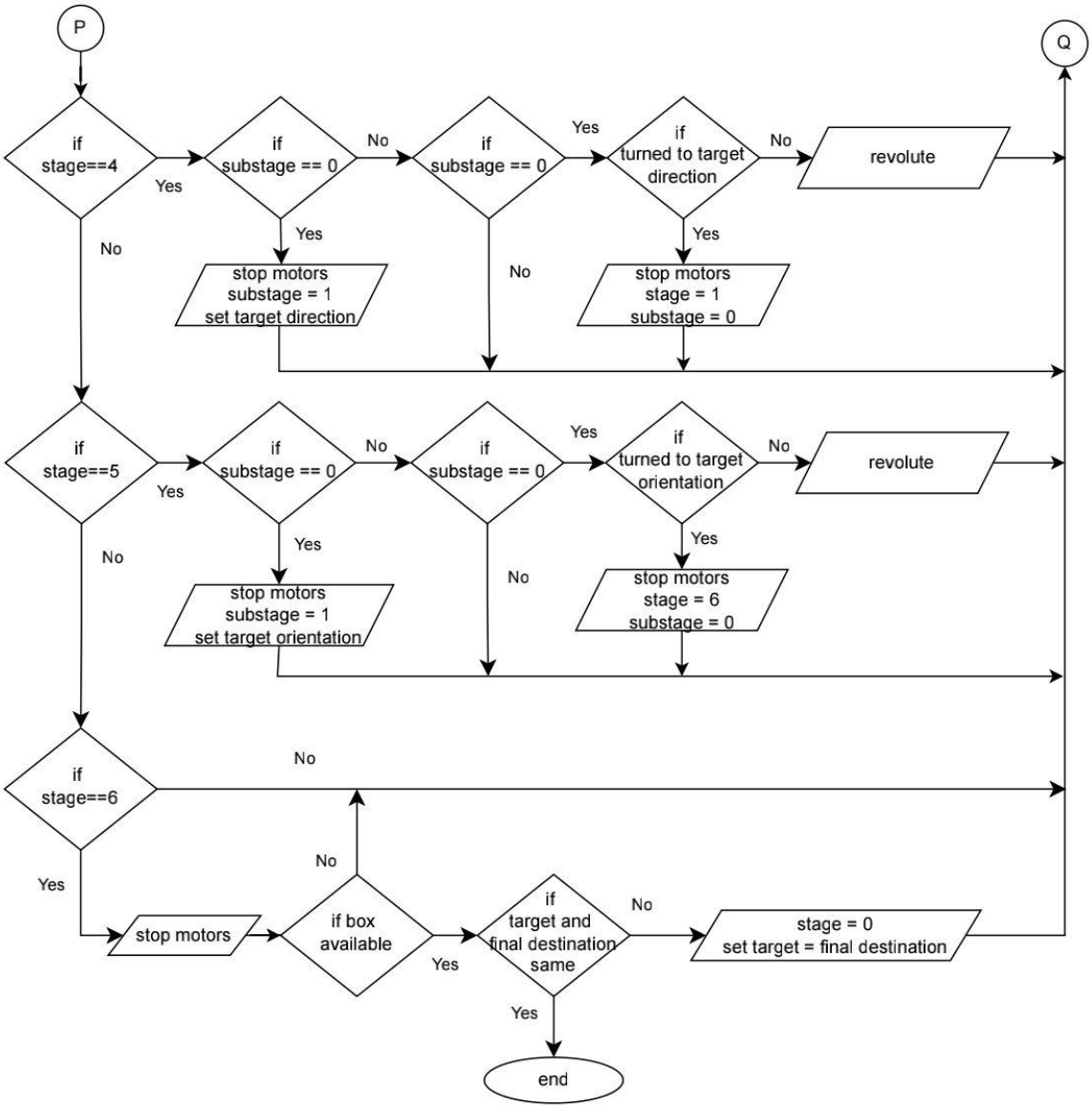
Sensor	Usage
3 Distance sensor at front	To identify obstacles
2 Distance sensors at left side	To follow wall
2 Distance sensors at top	To identify package (box)
GPS sensor	To localize
Compass	To localize
4 motors	For locomotion

Explanation on controller algorithm

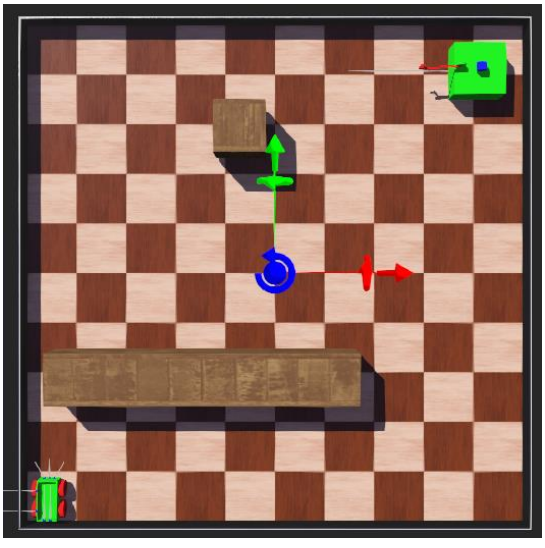
First robot travel towards the robot manipulator. Obstacles come across the path are avoided using Bug algorithm. Once the robot reaches manipulator, it waits till the package is loaded. After loading package, robot travels to final destination avoiding the obstacles.

Whole robot controller can be expressed as following flow diagram.





Movement and localization algorithm



Robot is localized using GPS coordinate obtained from GPS sensor. Heading of the robot is found using compass and the value is converted to bearing from North. The picture shows top view of the arena and the green arrow points North.

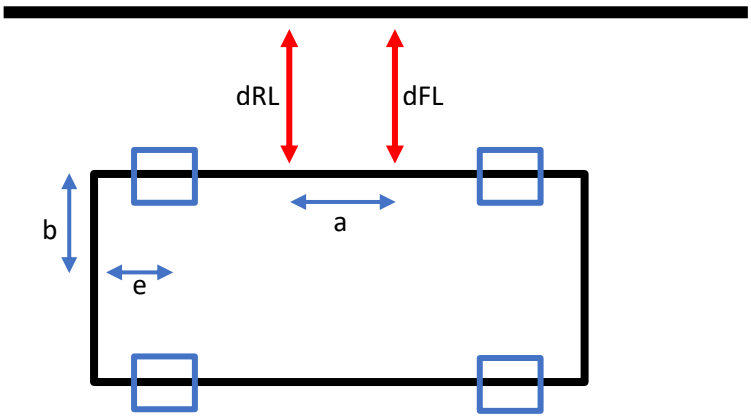
Before starting journey, controller finds the equation of the straight line which join starting point end destination point. Bearing of the destination from starting point is also calculated. Therefore robot always try to move on this straight line. When it come across an obstacle, robot turn right and wall follow around obstacle until robot come across the straight line. Then it goes on straight line.

Wall following algorithm

Following equations are used to estimate the wall.

$$\emptyset = \tan^{-1}\left(\frac{dRL - dFL}{a}\right)$$

$$d = \frac{dRL + dFL}{2} - dWallSide$$



Following equations are used to calculate left and right motor speed values.

$$\gamma = kWall * d$$

$$\alpha = \emptyset + \gamma$$

$$wL = \frac{v}{r}\left(\cos(\alpha) + \left(\frac{b}{e}\right) * \sin(\alpha)\right) \qquad wR = \frac{v}{r}\left(\cos(\alpha) - \left(\frac{b}{e}\right) * \sin(\alpha)\right)$$

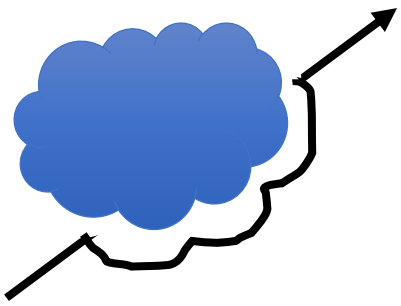
Following table gives the meaning of variables.

Variable name	Meaning
a	Distance between two proximity sensors
dRL	Reading of rear left proximity sensor
dFL	Reading of front left proximity sensor
dWallSide	Distance to be maintained between robot and wall
kWall	Gain of the wall following algorithm
v	Maximum linear velocity
r	Wheel radius
b	Half of the width of robot
e	As shown in the above figure

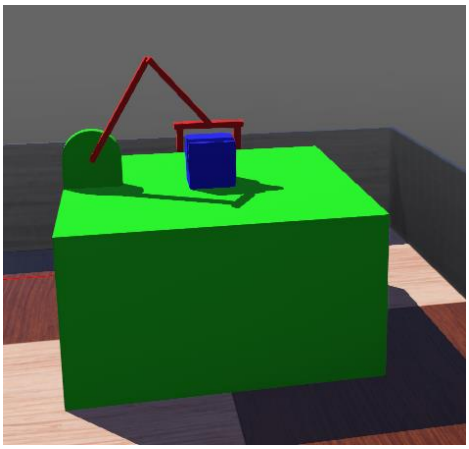
kWall is tuned by trial and error method. First acceptable range for kWall is calculated such that the wL and wR are not saturated. From calculation it was found that kWall should be in 0.001-0.0001 range.

Wall following algorithm (Bug algorithm)

Once the robot finds an obstacle in it's path robot turn 90^0 right from the current bearing. Then robot follows the wall of obstacle until it comes across it's original path. Then it follows original path.



Mobile Robot



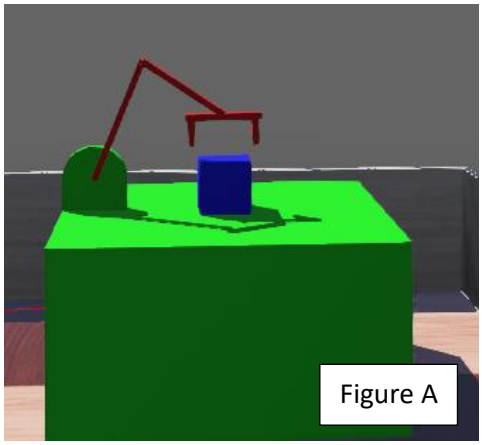
The following robot shows manipulator designed in Webots simulator. All the link lengths are as specified. Once the mobile robot arrives, manipulator picks the package (blue colored box) and loads it to mobile robot.

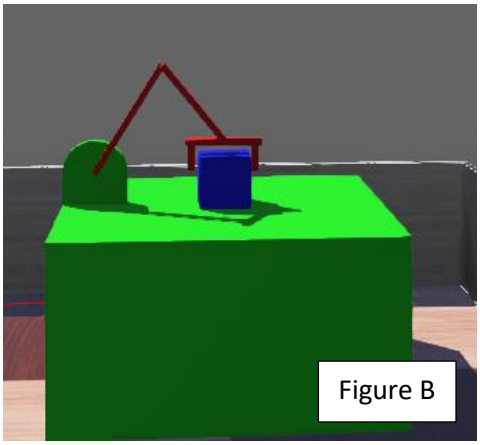
Sensors used and their application

Following table shows the application of sensors.

Sensor	Usage
3 motors	To rotate links
3 encoders	To find rotated angle
1 distance sensor	To identify the arrival of mobile robot
Connector (gripper)	As gripper to hold package

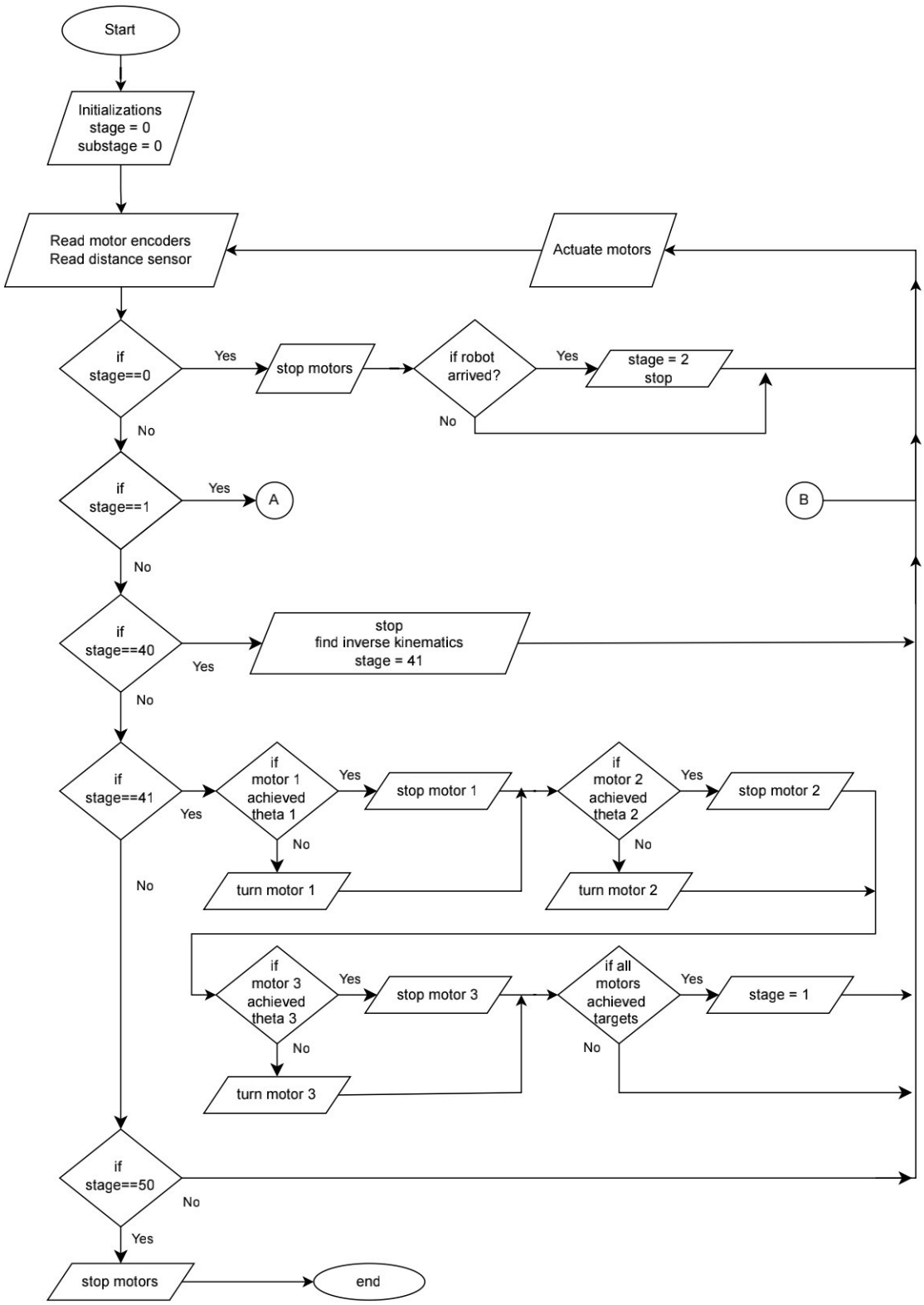
Explanation on controller algorithm

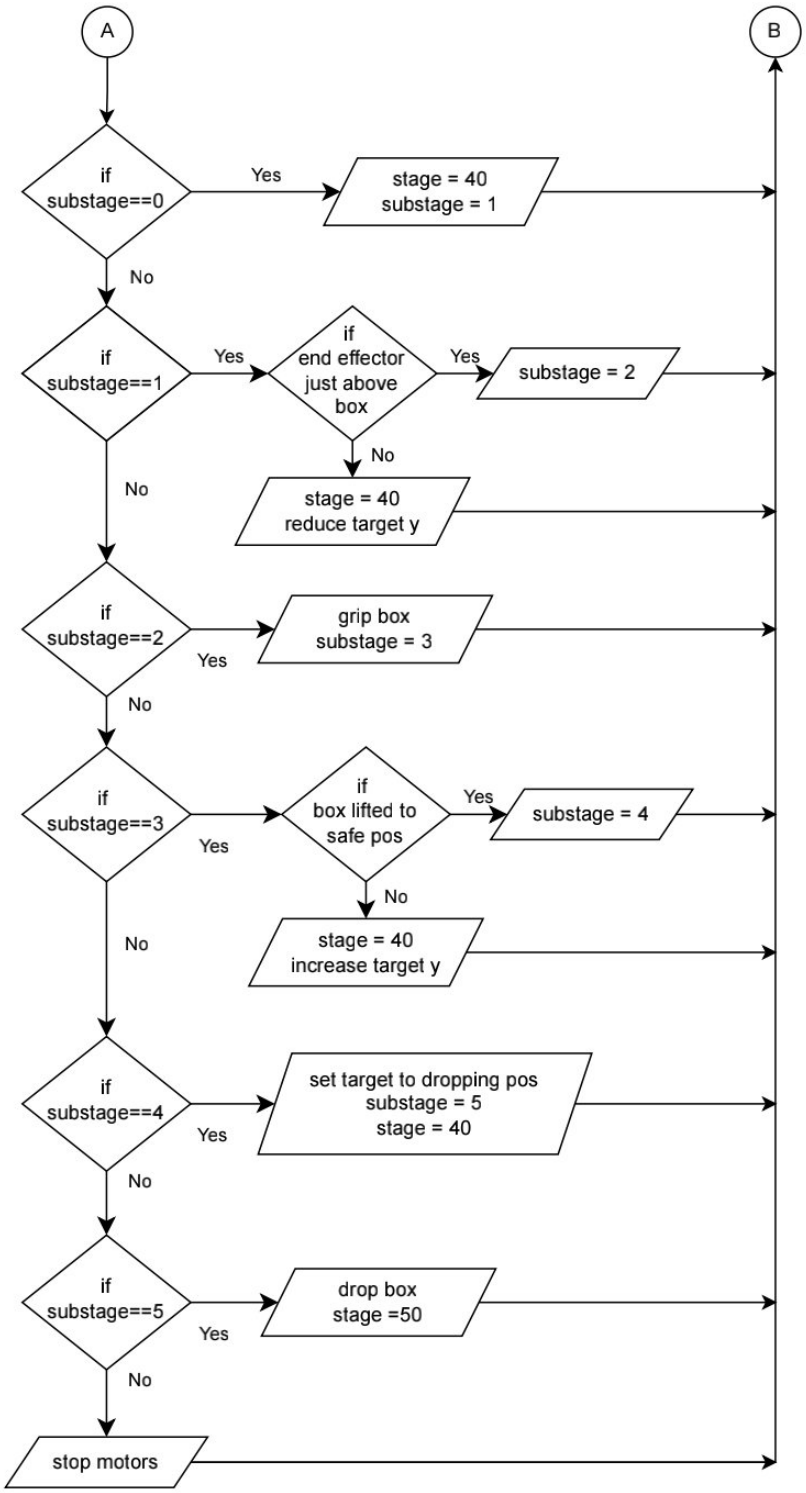




Manipulator waits until the mobile robot arrives. Once the mobile robot, end effector of manipulator is moved in pre-planned trajectory. To reduce the number of calculations and the path is free of obstacles trajectory of end effector is planned in joint space. But the package should slide into gripper without colliding. To achieve this first end effector comes above the package as shown in figure A and then carefully lowered in a vertical path into the position shown in figure B. Then the box is gripped. Then the box is lifted in same straight line. Then manipulator moves to unloading position and load the package to mobile robot.

This algorithm is shown in the following flow chart.





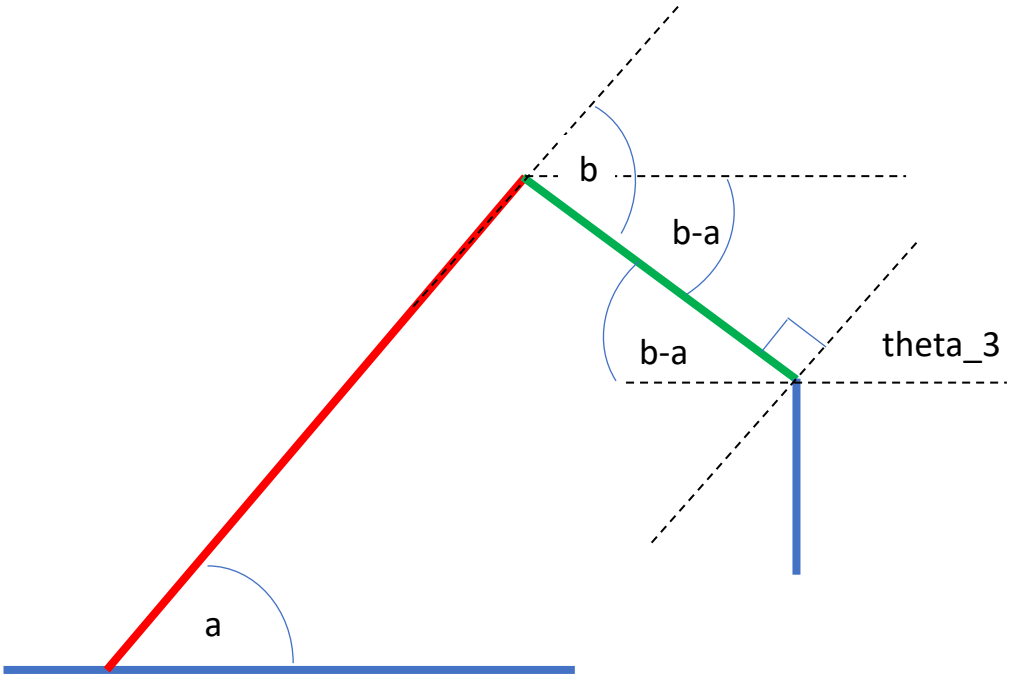
Simulating action of gripper

To simulate the gripping action of manipulator, a connector node is used.

Inverse kinematics

In this application, third/last link should be vertical. Therefore the 3 dof system can easily be converted to 2 dof system by reducing the link length of last link from the y coordinate. Then the inverse kinematics should be found to 2 dof system. Solutions for first two angles can be found using MATLAB as explained in the article https://www.mathworks.com/help/symbolic/derive-and-apply-inverse-kinematics-to-robot-arm.html

Once first two angles are found, final angle can be found as follows.



According to above diagram theta_3 is $(\pi/2)-(b-a)$. But when implemented in Webots angle a should be $(\pi-a)$. Then theta_3 becomes, $(3\pi/2)-b-a$. To match with Webots encoder another 2π should be subtracted. Therefore theta_3 is equal to $(-\pi/2)-b-a$.

Referred resources

Webots Nodes & APIs - https://cyberbotics.com/doc/reference/nodes-and-api-functions

Wall follow algorithm - https://youtu.be/DkOXpZtEZwg

Bug algorithm - https://youtu.be/HmkX9aiWISU

2 link planer manipulator inverse kinematics - https://www.mathworks.com/help/symbolic/derive-and-apply-inverse-kinematics-to-robot-arm.html