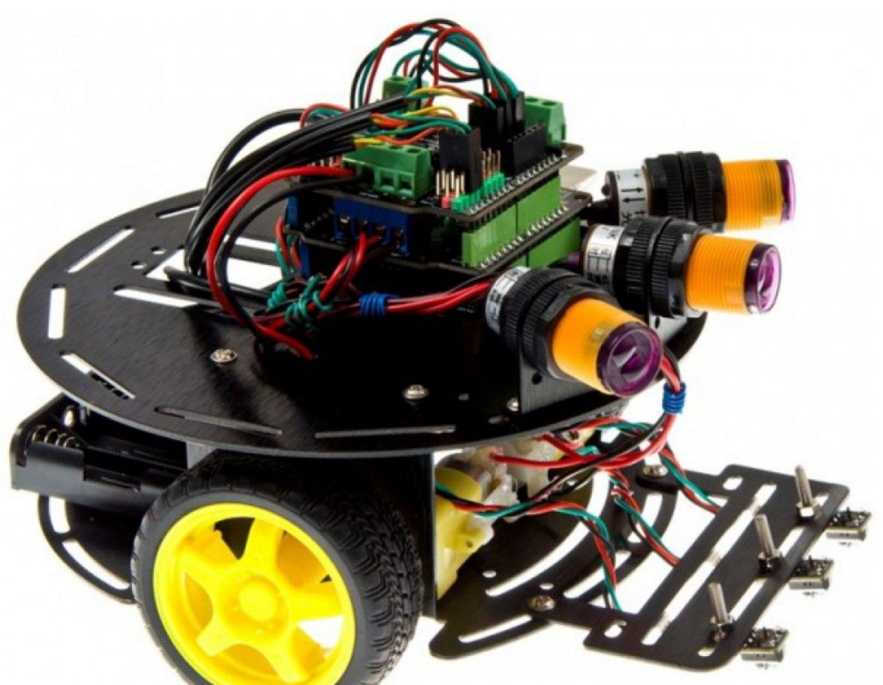
EEE3099S 2021

Milestone 1: Motion Control

Done By:
Luke Baatjes (BTJLUK001)
Jonathan Campbell (CMPJON005)



Date of Submission:
1 September 2021

Project Description:

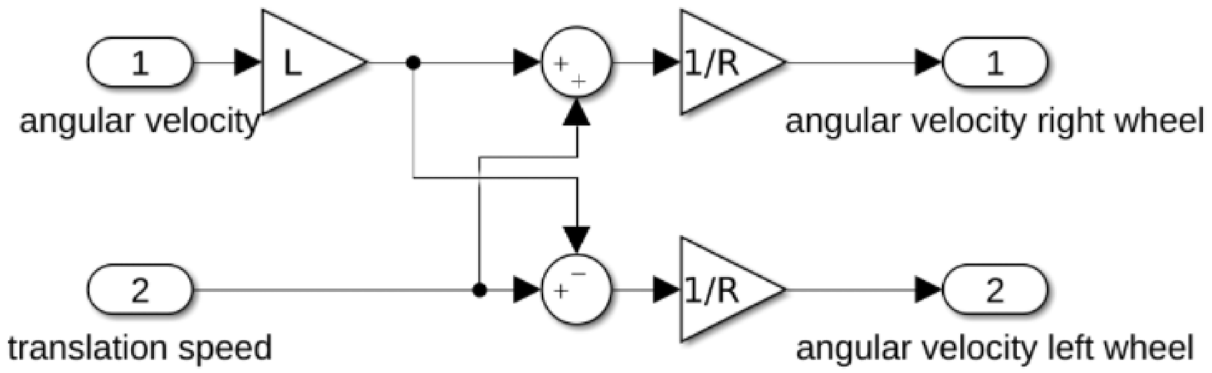
For this project we were tasked to build a line-following robot using Simulink. The first milestone is to learn how to control the robot using Simulink. To demonstrate this we need to move the robot in a straight line and stop within 10% of 1 meter. We also have to demonstrate the robot can rotate 90, 180, 270 degrees and stop within 10% of those angles.

To get to understand Matlab and Simulink we first did the onramp courses for them. We then decided to simulate the robots in simulink. This was done using the Robotics simulator blocks that have encoder simulators and robot simulators to help simulate the robot.

Once your robot can be controlled in simulation, you can use the same model to control the robotic platform. The robotic platform makes use of the RomeoV2, which is an equivalent of the Arduino Leonardo. Therefore, we use the Simulink Support Package for Arduino Hardware Library to communicate with the robotic platform.

Robot Kinematic Model and Descriptions:

The robot that we will be using is the Turtle 2 wheel drive robot platform with a smaller unpowered roller at the back to help with balance. With only 2 powered wheels with encoders we would have to use differential drive and maneuver the rover.
Differential drive robots work by controlling each wheel separately, this is done by having different motors on each wheel with encoders. If the robot wants to maneuver in the X-Y plane each wheel needs to be told there angular velocity. This is done by using the conversion block from the mobile robotics training library, this block converts the velocity and angle to the angular velocity for each wheel using this simplified diagram.



Taken from: Kinematics model of the wheel

To get angular velocity for each wheel we use these formulas form [6] and [7].

Encoder Mode:

The encoders that we are using for this project are light blocking sensors. These encoders. These sensors have an LED and a phototransistor, a disc with slits is attached to the wheel axis and passes in between the LED and phototransistor. When the wheel rotates the phototransistor senses the change in light. This is seen by the arduino as a tick, there are 20 ticks per one rotation of the wheel which works out to be 18 degrees each.

From the number of ticks we calculate the number of wheel rotations using equation [1]. We then note that the wheel's center travels a distance equal to the wheel's circumference (equation [2]) with one rotation of the wheel. Knowing this, the total distance travelled is calculated by using equation [3]. To implement this in simulink we add a gain block and equate its gain value to the circumference of the wheel divided by the tick count per rotation. The total encoder ticks needed to complete the calculation of distance is provided by the encoder.
The given radius R=0.031m with the axial length L=0.136m and ticks per Rotation=20m.

PWM Explanation:

PWM stands for Pulse Width Modulation. A PWM acts like a switch which constantly switches from a high value to a low value. What this does is control the speed at which the motor needs to be at by switching it on and off very pricey. If you take the average of the duty cycle you will find the average of the speed.

Distance Control Algorithm:

Distance control proved to be simpler to implement than angular position control. The design method used was an ad hoc design method which you could consider more agile than any other method.

Firstly, the left and right wheel encoder ticks were emulated (encoder block). Everytime the wheel rotates the encoder keeps track of the number of ticks (in our case it is 20 ticks. From the number of ticks we calculate the number of wheel rotations using equation [1]. We then note that the wheel's center travels a distance equal to the wheel's circumference (equation [2]) with one rotation of the wheel. Knowing this, the total distance travelled is calculated by using equation [3]. To implement this in simulink we add a gain block and equate its gain value to the circumference of the wheel divided by the tick count per rotation. The total encoder ticks needed to complete the calculation of distance is provided by the encoder.

Note that when the robot translates and rotates, we need to take into account that the robot's left wheel does not travel the same distance as its right wheel. Therefore, we are interested in the distance travelled by the center of the robot. To do this we calculate the average of the distance travelled by both wheels (equation [4]). The part described next is for completion of the simulation.

A constant block is used as the reference/target value of the distance. The actual distance travelled by the wheels is fed into a subtraction block that outputs the error signal. This error signal is fed into a proportional controller. The output of the controller is converted into an angular velocity which is fed into a lookup table block. This block generates a signal that the motor blocks interpret and output a signal to the robot simulator block.

The simulink simulation diagram can be seen in figure [1] of the appendix and the real hardware implementation diagram in figure [2] at the end of this report along with a comparison of the reference distance and the actual distance travelled by the center of the robot (figure [3]).

Angle Control Algorithm:

The angular position was a bit more difficult to implement than the distance algorithm. Once again we approach the design with an ad hoc method.

The angle control algorithm was not that much different to the distance control algorithm in that it starts off needing to calculate the distance travelled in the same manner as the distance control algorithm, however, after the gain block that calculates the distance travelled we need to have another gain block to calculate the angle that each wheel has moved through. The angle conversion equation (equation [5]) can be found in the theory section. Another part that was changed was the implementation of a proportional integral controller instead of just a proportional controller. This was done to reduce the oscillation when reaching the steady state.

It should now be emphasized that the encoder ticks for the left and right wheel differ by a factor of -1. Therefore, we need to take the absolute value of the angle from each wheel using an absolute value block in simulink. Thereafter, we follow the same procedure in completing the simulation, as we have done for the distance control algorithm, with the exception that now we use an angle as a reference in the constant block. The simulink simulation diagram (figure [4]) and its associated hardware implementation diagram (figure [5]) can be found in the appendix at the end of this report if needed along with a comparison of the reference angle and the actual angle the wheels move through (figure [6]).

Appendix A: Formulas

$Number\ of\ wheel\ rotations\ =\ total\ encoder\ ticks\ /\ tick\ count\ per\ rotation$
[1]

$circumference_{wheel}\ =\ 2\ *\ pi\ *\ r_{wheel}$
[2]

$distance_{total}\ =\ number\ of\ wheel\ rotations\ *\ circumference_{wheel}$
[3]
$distance_{center\ of\ robot}\ =\ (distance_{total\ left\ wheel}\ +\ distance_{total\ right\ wheel})\ /\ 2$
[4]

$angle_{moved\ through}\ =\ distance_{respective\ wheel}\ *\ (2\ /\ axleLength)$
[5]

$$angeVelocity_{right\ wheel} = (velocity + axleLength * angularVelocity)/\, r_{wheel}$$

[6]

$$angeVelocity_{left\ wheel} = (velocity - axleLength * angularVelocity)/\, r_{wheel}$$

[7]

Appendix B: Screenshots

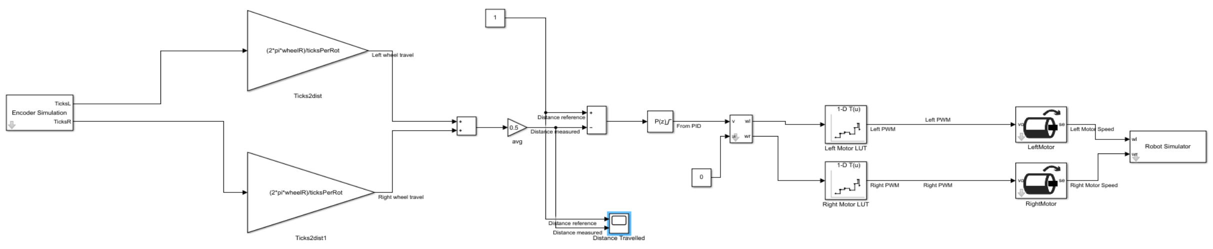


Figure 1: 1m travel simulink simulation file diagram

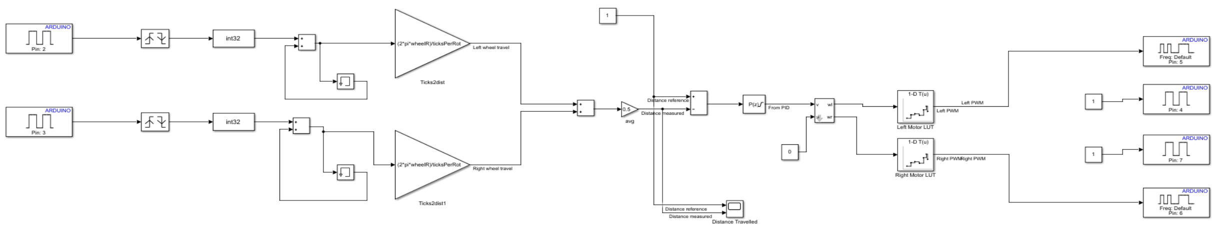


Figure 2: 1m travel simulink real implementation file diagram

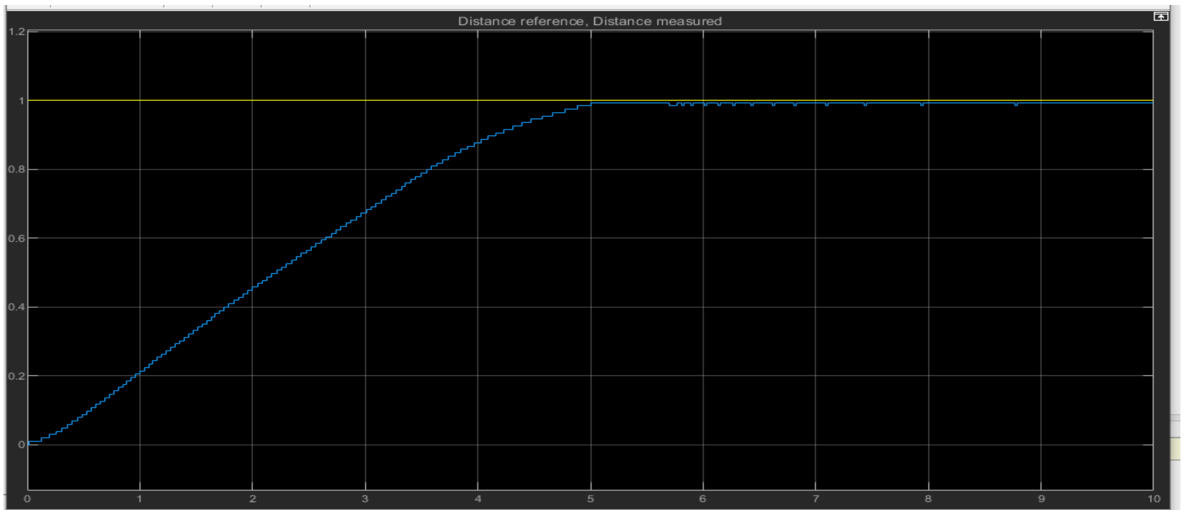


Figure 3: Comparison between reference/target 1m and actual distance

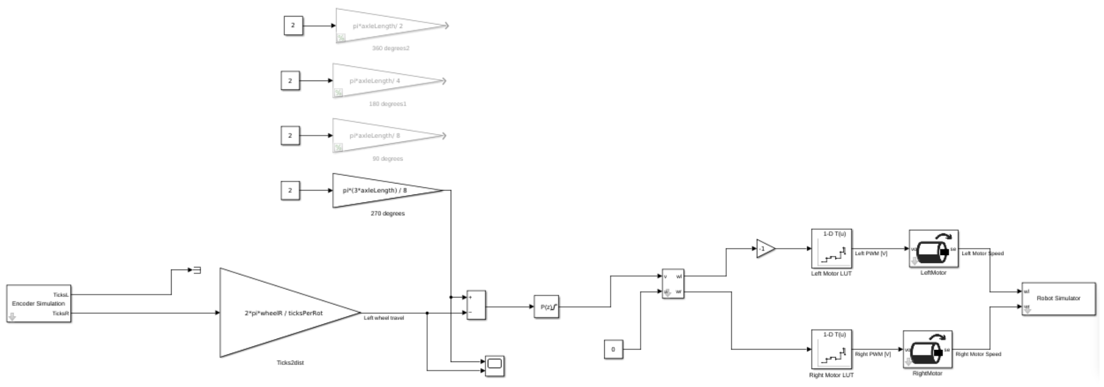


Figure 4: rotational simulink simulation file diagram

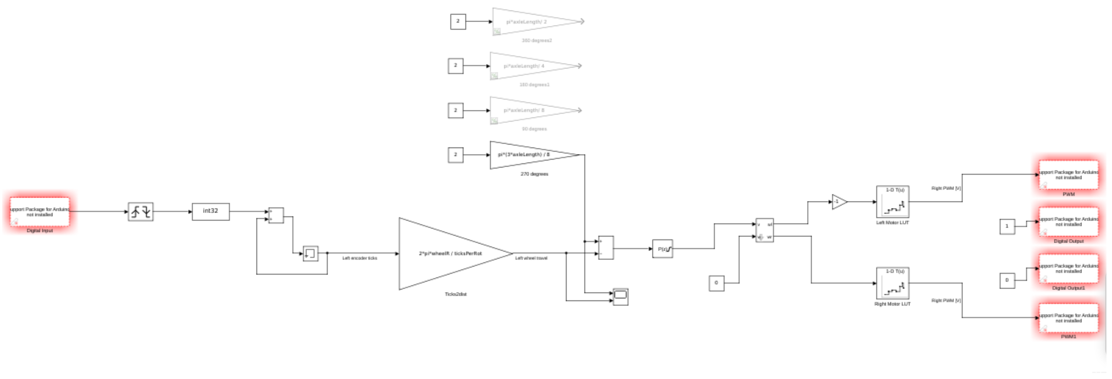


Figure 5: rotational simulink hardware file diagram

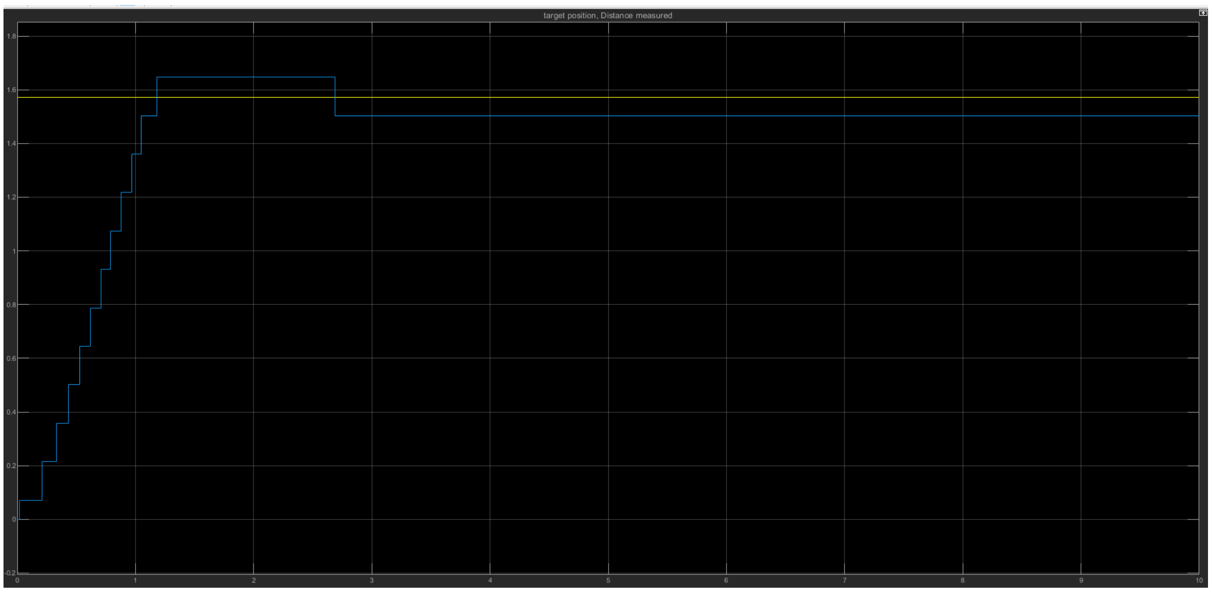


Figure 6: Comparison between reference/target angle (90) and actual angle