Embodied Artificial Intelligence Project

1. Semester Master 2021

Group 26 - Frida Jul Pilegaard (frsch18) & Nicoline L. Thomsen (nthom18)

November 2021

1 Introduction

The objective is to follow a line to locate a tomato can in a model arena, and bring the can back the same way. The line may have various curvature and bends as well as areas of white between consecutive segments of line. The arena is in two levels which are connected by a ramp which needs to be traversed as part of the challenge both up and down.

The challenges of the arena are summarised as follow:

- Following a straight line
- Following a line which curves or has a 90° corner
- Following a straight dotted line
- Following a dotted line which curves
- Grasps a can and following the path with the can.
- Climbing a ramp and descending a ramp with the can

Using the principles of embodied AI, a mobile robot is designed using LEGO Mindstorms EV3 kit [1] to accomplish the task of retrieving the can.

2 System Description

The final design of the robot can be seen in figure 1. It has a length of 24.8 cm and a width of 18.2 cm om the widest point.

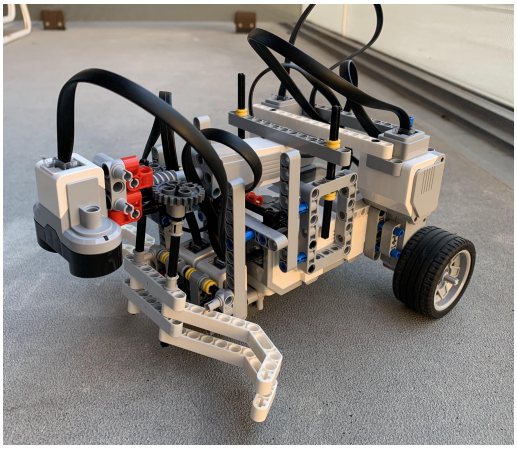

Figure 1: The final search and rescue robot design

The robot is equipped with two colour sensors for line detection, and an ultrasonic sensor for detection of the can. Two large motors are utilised to drive two wheels which are located at the back and a smaller motor is used for a gripper.

The behaviour the robot exhibits is controlled by a state machine, switching between states based on sensor input. These different states define the actions the robot should take for a given situation. See description of the state machine, and a more detailed explanation of the actions themselves in section 3.

To follow a straight or only softly curving line a classical PI controller is implemented and tuned to achieve the desired behaviour. The chosen method is to use the two colour sensors and the error is the difference between them. It is supported by additional behaviours to handle sharper corners. The robot makes a distinction on weather it is in a sharp turn/corner, or the line has stopped all together, by looking at when 'true' black was last detected. This is described in greater detail in section 3.1.

Following a dotted line is handled by driving straight when it is determined that the line is ending rather than a sharp turn.

The challenge of climbing and descending the ramp is handled by designing the physical structure of the robot so that it can handle a steeper slope. How this is accomplished is described in section 4. Grasping the can once found, is done using a front-mounted gripper, that holds the can in place. It is activated when the ultrasonic sensor detects an object closer than a certain threshold. The gripper is described further in section 4.2.

To return with the can the robot needs to turn around, and does so by turning left until the line is detected again, so the line-following can continue in the reversed direction.

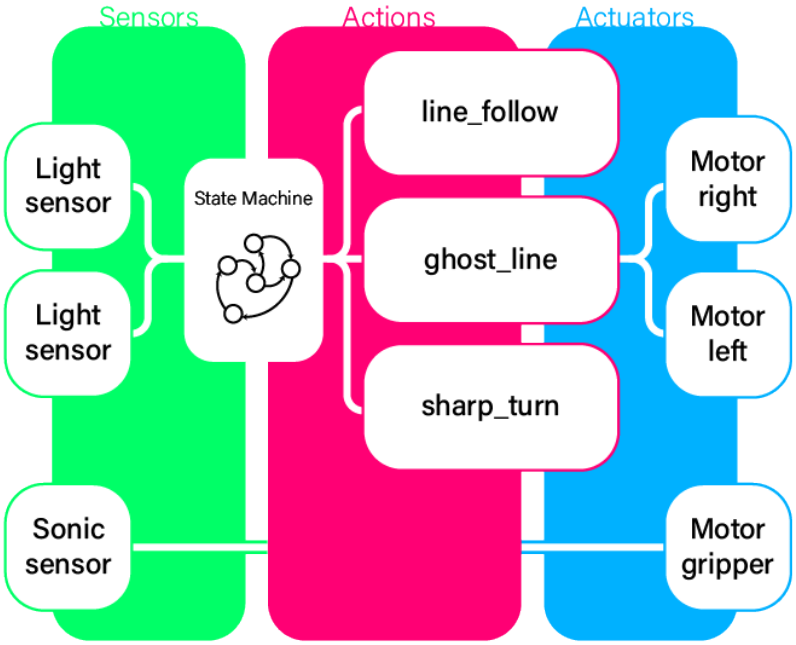

Figure 2: Information map between hardware and software

Figure 2 illustrates how information from the sensors is inputted to the finite state machine, that decides what action to take. Those actions, or states, send output commands to the actuators accordingly. The sonic sensor is connected directly to the motor of the gripper, this action is not controlled by the state machine. Once the ultrasonic sensor triggers the claw is closed. All other motion halt while the gripper is closing around the can.

3 Software

3.1 Action Description

When driving on straight line segments, it is desired to drive smoothly along that line. For this a PID controller is used, avoiding jagged motion. To achieve this two colour sensors are used, the sensors are used in the mode `COL-REFLECT`. The sensors are placed ahead of the wheels to allow the system to predict where the robot is headed. The sensors are placed as close as possible and the goal is for the point between the sensors to follow the centre of the line. Therefore the sensors need to detect the same amount of the line which would result in the same output from the sensors. The reference is for the error to have a difference of zero between the two sensors. It is important that the robot manages to close in on the line and gradually reduce the error so it eventually drive almost straight along straight lines. The PID controller is tuned so that it gradually decreases the error

as it oscillates around the line. Reducing the rise time and overshoot has been prioritised. Steady state error is not important in order to achieve this. In table 1 the parameters of the PID controller can be seen. The parameters have been determined by trial and error. It has a derivative (D) gain of 0, making this a PI controller. Though it will still be referred to as PID in the rest of this report.

P	I	D
0.3	0.1	0

Table 1: P, I and D gains for the controller

The control variable from the PID controller needs to be delegated to the two wheels. This control variable is converted to a difference in the two motor speeds using equation (1), where u_r and u_l are the force send to the right and left motor, r is the radius of the wheels, L is the length between the wheels and U_{PID} is the control variable from the PID controller. This equation is derived from [2].

$$(u_r - u_l) = \frac{L}{r} U_{PID} \tag{1}$$

r is measured to be 27.86 mm and L to 140.52 mm.

This difference is added or subtracted from a `base_speed` to turn the robot the appropriate amount.

Tuning the PID controller for smooth alignment on straight segments, means reducing that controller's ability to handle turns. Here, extra logic is implemented to handle events where both light sensors see white. Three such scenarios are illustrated on figure 3. A green arrow shows the desired direction of travel in the three cases, where the PID controller is insufficient for control.

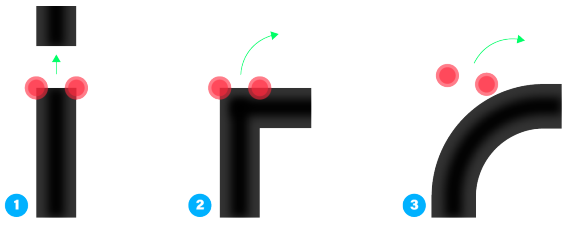

Figure 3: Scenarios that will be encountered during line-following

The most important distinction to make is the difference between scenario 1 and 2. Here both sensors will see the end of a line almost simultaneously. To know whether or not the line can be expected to continue straight ahead after the white segment, or if it is at a

corner in either direction, it is determined if any of the sensors saw 'true' black recently. True black is referring to the middle of the line where no white is visible to the sensor, thus being darker than at the edges. 'Recently' is within a time-out, a predefined number of samples which is chosen by tuning the system.

If it is determined that the line has ended, the robot drives straight ahead. If it is at a corner, or the robot has driven off the line, like in the case of scenario 3, the robot makes a turn around its rotation axis, with no added base speed. Thus the robot will not drive too far away from a line.

3.2 Action Selection

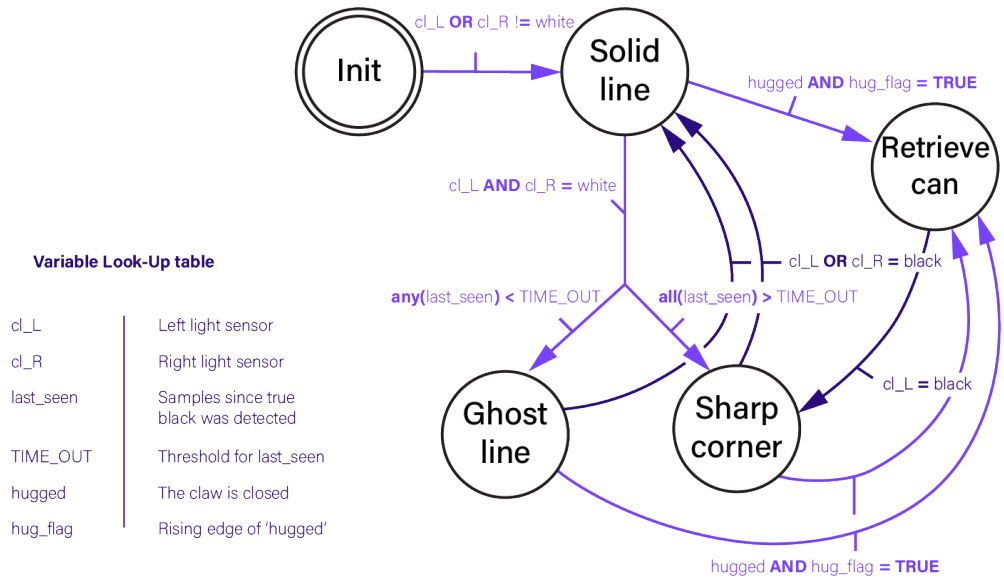

Figure 4: Finite state machine that controls action selection

In figure 4 the finite state machine that controls the action selection for the robot can be seen. The robot starts in the 'Init' state, being passive until one of the light sensors detects something other than white. In 'Solid line' the PID controller is deployed.

If both sensors detects white in this state, it is either because the line has ended or the robot is at a sharp turn. The difference is determined by whether or not one of the sensors have seen black within a given period.

If the claw begins to close, the state is changed to 'Retrieve can' regardless of the current state. The robot grabs the can, turns around and then gives control to 'Sharp corner', which will get the robot back on track.

4 Hardware Setup

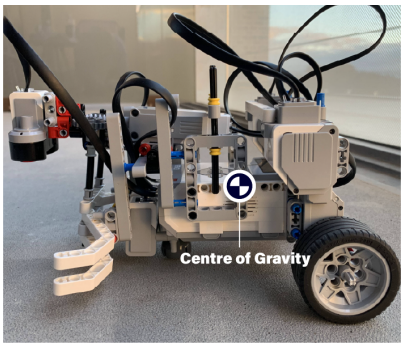

Figure 5: Photo of the robot with the centre of gravity marked

Apart from line following, which is addressed in section 3, the robot has to overcome some physical challenges, the most prominent to be able to drive up two ramps. The steepest and longest ramp has a slope of 30° degrees. To ensure that the robot does not keel over backwards when it is on the ramp, it has been designed with a low centre of gravity seen in figure 5. It is driven using two actuated wheels which are placed behind the centre of gravity, and balanced by a non-actuated spherical wheel in front. In order to successfully navigate the environment, it also has to be able to go through corridors with a minimum width of 29.5 cm, which limits the width of the robot. Therefore it has been designed to have a maximum width of 18.2 cm so it can comfortably navigate the narrow corridors.

4.1 Sensors

In order to sense its environment the LEGO robot is equipped with two colour sensors and an ultrasonic sensor. The colour sensors are used for line following and is pointed towards the ground in front of the robot. They are placed at a height of 7.2 mm above the floor to enable the robot to enter the ramp, and to sense a larger area of the mat. The sensors are not placed further from the mat because the data becomes unusable due to noise from the environment.

For line detection the colour sensors are placed side by side with 2.19 cm between their respective receptive centres. This width is chosen to so that each sensor can follow one side of the 2 cm line. The distance between the colour sensors and the wheels are 12.5 cm. Reduction of this distance would improve the PID line following, however it is not possible without changing the robot to have a different weight distribution. This has been shown to make it prone to keeling over on the ramp, and it is solved by additional behaviours in the code instead.

To detect the tomato can, an ultrasonic sensor is placed in front of the robot facing downwards at a height, which allows the can to go under it. When the can is underneath the sensor, it will signal the gripper to grasp the can.

4.2 Gripper

The robot is equipped with an actuated gripper which uses a medium LEGO motor, seen in figure 6. The gripper design follows the tutorial in [3]. The claws have been exchanged so that it can wrap around the can in a suitable way, and thereby hold it steady during the trip back through the arena. The gripper is placed in front of the robot to allow it to follow the line to where the can is located. The ultrasonic sensor is right above the grasping area pointed down towards the ground. When the can is underneath it, the sensor reading will fall below the threshold which indicates that an object is sufficiently close, which in turn activates the gripper. The gripper is placed at a height which makes it grasp around the centre of mass of the tomato can to be able to push it around effectively without the risk of tipping it over.

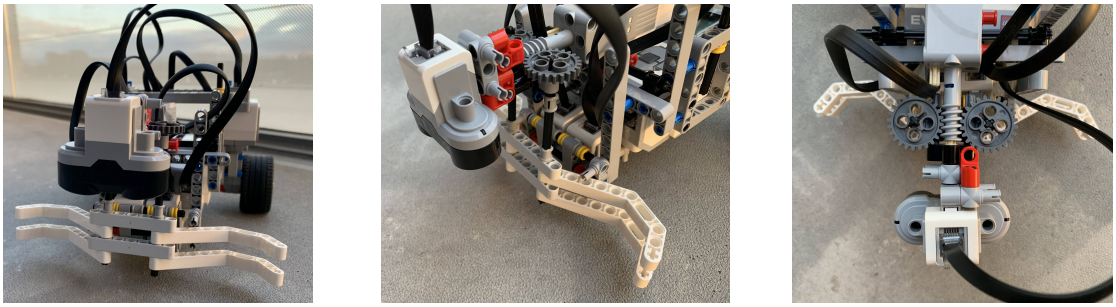

Figure 6: The claw mounted on the robot

5 Testing of system

5.1 Light Intensity Values

In order to determine the threshold values for when the colour sensors are detecting 'true' black and 'true' white, a number of tests were performed testing different areas of both black and white on the lower- and upper level. To ensure all true white readings are detected as white, the threshold is set to the lowest value detected on a completely white surface. For black the value is set slightly higher than the highest detected value, to ensure that the black line is detected even though it might not have a reading where the sensor is exactly above black. Table 2 shows the light intensity values for black and white.

	White	Black
Roof	43	7
Ground floor	43	4

Table 2: Light intensities measured on the arena. Minimum for white, maximum for black

5.2 Performance Test

To tests the robot's ability to deal with the various challenges of the arena, a number of tests have been conducted, using isolated pieces of the track to test on. These include straight lines, 90°curves, 180°curves, 90°corners, straight dotted lines, 90°curved dotted lines and 180°curved dotted lines. These tests were performed both with and without the can grasped in front of the robot. The tests without the can were repeated with the robot starting at an approximately 45°angle to the line. All tests have all been graded pass/fail and have been performed 10 times. The success rate of the tests are summarised in table 3. For solid lines the robot performs well, but there is a difference to be had when the robot is placed in a bad position at an angle, or when it is pushing the can. Following the dotted line is only successful when the robot meets it head on, and it is incapable of handling dotted lines that curve, especially with the load of the can. If it is at an angle before meeting the dotted line, it will also fail.

	Sharp corner	90°curve	180°turn	Straight
Solid	**80**	**100**	**100**	**100**
Dotted	-	50	0	100
Solid at an angle	**100**	**60**	**80**	**100**
Dotted at an angle	-	0	0	0
Solid with can	**100**	**60**	**90**	**100**
Dotted with can	-	0	0	100

Table 3: Success rate [%] for experiments following the line

Additionally the robot was tested driving up the ramp without the can and down the ramp with the can. The results of these tests can be seen in table 4. Driving up and down the ramp is successful in most cases, but it is not as robust as it would have been preferred. Especially on the way down, where it is at the mercy of gravity, there is a higher uncertainty for whether or not it will complete the task.

Ramp up	Ramp down (with can)
70	60

Table 4: Success rate [%] for experiments driving on the ramp

6 Discussion

Placement of Colour Sensors

The colour sensors are placed in front of the LEGO brick facing down towards the mat. A fairly small change in the wheels will result in a rather large change in the position of the colour sensors, when the wheels and sensors are placed far apart. Because of this placing the sensors closer to the wheels might result in smoother motion. However, because moving the sensors back would require that the brick itself is lifted to make room, it is ultimately chosen to place them in front to keep the centre of gravity low. The undesired movements are reduced by tuning the PID controller and having additional architecture to handle sharp turns.

Colour Sensor Accuracy and Variation

The colour sensor outputs varies greatly depending on the lighting of the location, this poses a challenge as there is different lighting conditions on the top and bottom of the arena. In this iteration of the code the same limits are used for the entire arena, but a possible way of making the system less prone to errors could be to have two different sets of thresholds, depending on which part of the track the robot is on. A gyroscope could be used to determine when the robot is going up and down the ramp, and thereby determining what part of the track it is on. Alternatively the robot could perform a number of calibrations throughout the tracks to adjust for changes in lighting.

Dynamic Base Speed

To increase the overall speed of the robot, it is interesting to consider using different base speeds with the PID controller. This base speed could be described by a linear, or non-linear, function that is dependent on the error, being a high values for lower errors, and slow down for high errors so there is time to correct them. The error changes quickly however, resulting in jerked motion due to the sudden acceleration. An alternative is to let the function be dependent on the time with low errors, or using the value of the integration term of the PID controller. This will ensure the robot can utilise segments of the line that are straight, to gain speed.

The downside of this approach lies in the method for handling sharp turns and ended lines, this is described in section 3. This method relies on a constant time-out value, that is predefined by tuning. The value of the time-out is dependent on the speed at which the robot drives, so using the proposed method of a dynamic base speed, this time-out value should be dynamic as well. It greatly increases complexity, but for decreasing completion time it could be beneficial. It is however not implemented in this project.

Using a PID Controller For All Control

Tuning a PID controller that can drive smoothly along a line, while also handling sharp turns, is a complex and time-consuming task, and was therefore not done in this project.

However if the controller was able to handle both tasks with a certain robustness, it would simplify the system as a whole. With a 2-sensor configuration, the reference is the difference between the values of those sensors. This means that when the robot reaches the end of a line, it will by design, provided it was aligned with the line, drive straight forward. This would omit the need for extra logic to handle that scenario.

Not Detecting Curving Dotted Lines

When the robot is trying to drive on a dotted line which has some curvature, the robot drives forward until it reaches white, but detecting that it curves and therefore activating the sharp turn in either direction. This often results in the sensors going between the lines and the robot turning 180 ° until it meets the solid line in the opposite direction. This could perhaps be solved by implementing a small forward motion in the turn behaviour, or starting each turn by driving forward a little but both these solutions might result in issues while navigating curving lines or corners

7 Conclusion

Using the principles of embodied AI a robot has been designed with accompanying software architecture which aims to accomplish the task of retrieving a tomato can from a arena. Hardware and software has been combined to attempt to overcome the challenges in the environment.

The built robot and the developed control software manages to follow solid lines almost every time without the can. It does well when it is straight and in both the tested curves and it has a decent success rate for corners. When the robot is holding the can the success rate is much lower in the two types of curves. When at an angle the robot manages to correct the movement to a straight line or in a sharp turn, it does however not do so each time with the two types of turns. Dotted lines are only managed both with and without the can when the robot comes straight onto the line, and half the time without the can in a 90° curve. All other dotted line scenarios has a success rate of 0%. Ascending the ramp is successful in 70% of cases and descending with the can in 60%. Unfortunately mainly due to the inability to navigate a curving dotted line the robot cannot run the entire track.

8 Bibliography

[1] LEGO. Lego® mindstorms® ev3. https://www.lego.com/da-dk/product/lego-mindstorms-ev3-31313. Online; accessed 29 November 2021.

[2] Steven M LaValle. A differential drive. http://planning.cs.uiuc.edu/node659.html. Online; accessed 28 November 2021.

[3] Filipp Sudanov. Lego mindstorms ev3 simple gripper using medium servo motor - building instruction. https://www.youtube.com/watch?v=ULsRBRowNB0&ab_channel=FilippSudanov. Online; accessed 29 November 2021.