University of Sheffield

# COM2005-3005
# Bio-inspired Computing and Robotics


Robotics Lab Assignment

Ho Yin Angus Ng

Hyun Han

Matthew Kinton

Malvin Todorov

Leon Singleton

Robot: B4

Department of Computer Science

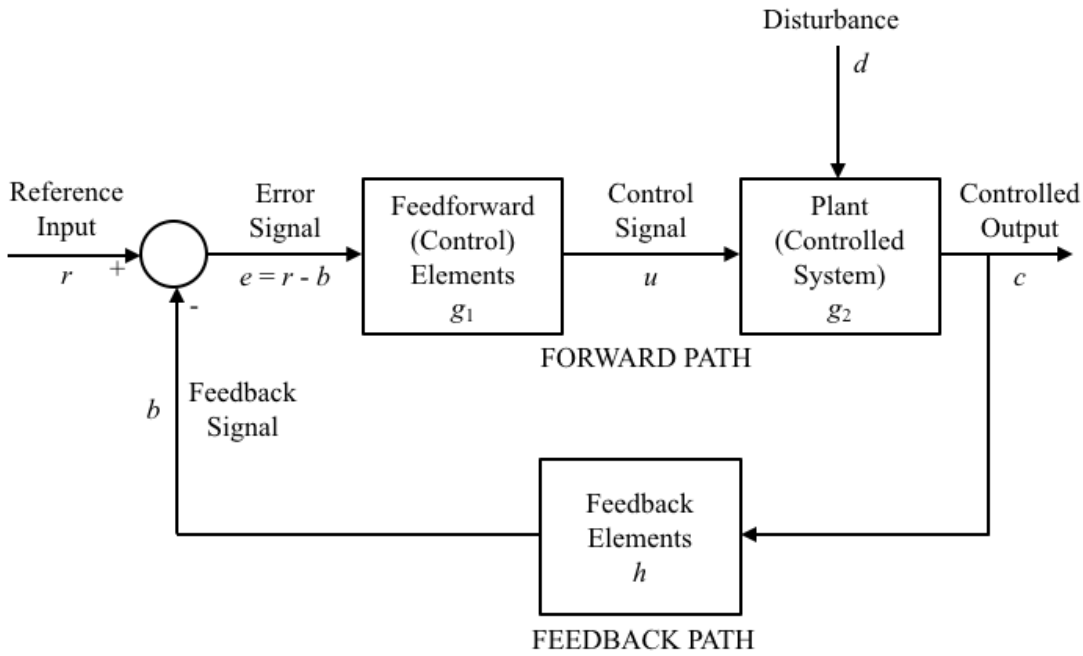
May 10, 2018

# Contents

# 1 Part I: Experiments with a PID Controller

## 1.1 General Principles

Recall from Prof. Moore's Lecture-2 on 'Autonomous Systems' that a classic negative-feedback control system is structured as follows:




Recall also that $g_1$ is commonly configured as a proportional-integral-derivative (PID) controller. Mathematically, this takes the form:

$$u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de}{dt}(t),$$

... where $K_P$ is the proportional gain, $K_I$ is the integral gain and $K_D$ is the derivative gain. A simple proportional controller would use $K_P$ only (i.e. $K_I = 0$ and $K_D = 0$).

In order to ensure that you fully understand these basic principles, please provide brief answers to the following five questions:

**Question 1:** *What is the function of the 'Reference Input'?* (Worth up to 2 marks)

| |
|---|
| The Reference Input is the desired outcome of the closed control loop. It is the target value for which the sensors read. For example, in a central heating system, the Reference Input is the desired room temperature. For the lego robot, the Reference Input could be a set distance away from obstacles in front of it. |

**Question 2:** *What is the significance of* $e = 0$*?* (Worth up to 2 marks)

The Error Signal $e$ is the difference between the Reference Input and the Process Variable from the Feedback Signal. Therefore, the greater the difference between the Reference Input and the Process Variable the greater the Error Signal and the significance of $e = 0$ is when the Reference Input is achieved. For example in the central heating system, the room temperature is now equal to the desired temperature. For the lego robot, the distance away from the obstruction is the target distance away from the obstruction.

**Question 3:** *What is the advantage of a closed-loop configuration over an open-loop configuration?* (Worth up to 4 marks)

In an open-loop system the system does not receive feedback from the sensors after the action has been executed. It acts entirely on the basis of an input and doesn't change depending on the feedback signal and it assumes the reference input has been achieved. For example a central heating system controller by a timer, heat is applied for a set amount of time regardless of the rooms temperature. The advantage of the closed-loop system then is that it receives feedback from the sensors after the action has been executed. If the reference input is not reached subsequent actions are performed determined by the current error until the reference input is reached. In the case of a central heating system, it looks at the current room temperature, and adjusts its heaters and coolers accordingly until the desired temperature is achieved.

**Question 4:** *Give an example of a real-world negative-feedback control system and describe its operation.* (Worth up to 8 marks)

A central heating system with a thermostat is an example of a negative feedback control system. Initially the thermostat is set to the desired temperature, the reference input. Next, the thermostats thermometer (sensor) measures the real room temperature, this is the process variable given as the feedback signal. The error value is then calculated (desired temperature - real temperature) which is used by the thermostat to emit a control signal to the room heaters and coolers (process). If the error value is $> 0$ the heaters must be initiated as the room needs an increase in temperature. If the error signal is $< 0$ then the coolers must be initiated as the room needs a decrease in temperature. A value of 0 means the desired temperature is reached. As a PID Controller the rate of temperature change is determined by 3 constants (gains). Firstly, the proportional term $K_P$ gives an output value proportional to the error value. A high proportional term would result in a large rate of change of temperature. Oppositely a small proportional term results in a small rate of change. Secondly, the integral term $K_I$ is used to consider the history of the error, it accelerates the process towards the reference input. In case of a steady state error, the integral term would cause the heaters or coolers to be controlled to reach the desired temperature. Finally the derivative term $K_D$ provides a smoother transition in temperature as it aims to make the rate of change of temperature 0.

**Question 5:** *Assuming that $K_I = 0$ and $K_D = 0$, what would happen to your example system if (i) $K_P = 0$ or (ii) $K_P < 0$?* (Worth up to 4 marks)

As the integral term $K_I$, and the derivative term $K_D$ equal 0 they will have no effect on the outcome of the PID Controller. Therefore, the only value of interest is the proportional term $K_P$ and the control signal is therefore proportional to the error value. Given that $K_P = 0$, the control signal must also be 0 meaning that the process variable will never converge towards the reference input, it will stay at its initial reading. For the central heating example, no heaters or coolers will be applied as no change in state is determined. Given that $K_P < 0$, the process variable will diverge away from the reference input because the opposite action to the desired action would be applied. For the heating system if an increase in temperature is required the PID Controller would determine a decrease in temperature. Oppositely for a decrease in temperature, it would be determined that an increase in temperature is necessary.

### 1.2 Setting up the Lego EV3 Robot

You have already used the Lego EV3 system in your first-year Java Programming course (COM1003). The EV3 robots are controlled by sending commands over a wireless (Bluetooth) connection from a `Java` program running on a desktop computer. You will use the `ShefRobot` package which provides a simple wrapper of the `Lejos` EV3 library. The relevant resources (including `ShefRobot`) can be found at: `https://staffwww.dcs.shef.ac.uk/people/S.North/campus_only/com1003/robots/index.html`

For COM2005-3005 you will use the Lego EV3 robot in its basic 'tribot' configuration (two driving wheels and a rear trailing wheel) with additional sensors.

Your first practical task is to collect your robot, set up communications between your robot and your computer, and write a simple `Java` program to confirm your ability to control the robot. If you have any difficulties, contact one of the demonstrators. Tick off the following items (by editing the LaTeX source file) when they have been achieved:

- ☑ Robot collected
- ☑ Communications established between computer and robot
- ☑ Able to control robot using `ShefRobot`

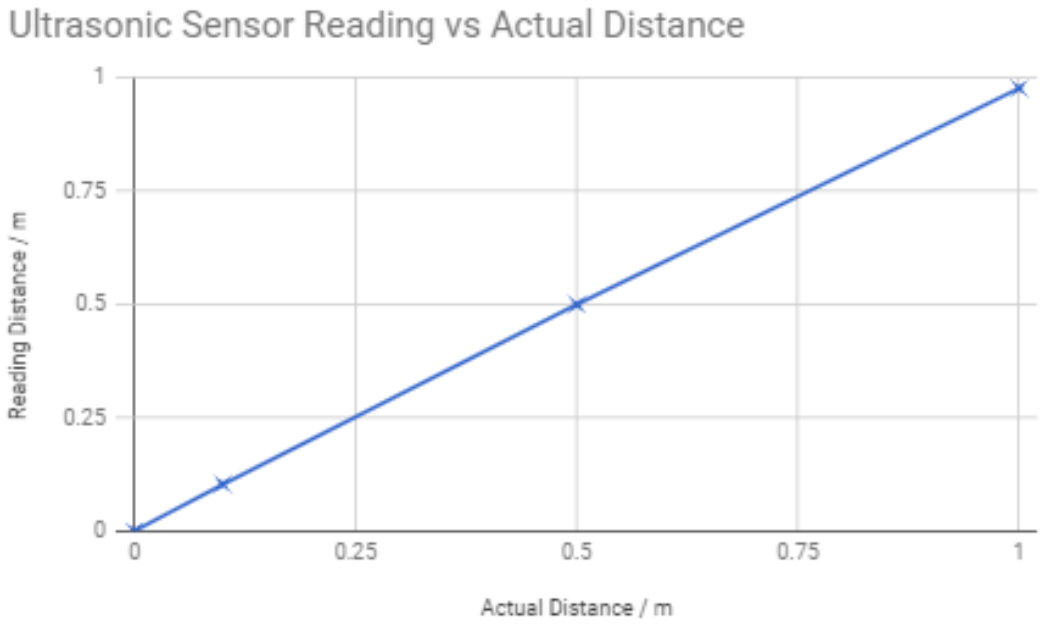
Next, mount one of the ultrasonic distance sensors on the front of the robot facing forwards, and test that you can read its sensor values continuously (this will require a processing loop):

- ☑ Sensor mounted
- ☑ Able to receive sensor values

**Question 6:** *How do the values from the ultrasonic sensor vary as a function of (i) distance to a surface and (ii) surface angle, and how stable are the readings over time?* (Worth up to 8 marks)

*(i)*

The readings of the ultrasonic sensor vary linearly as a function of distance to a surface. Increasing the distance from the surface increases the sensor reading as expected. However, sensor readings under $0.03m$ and over $1.7m$ are returned as $INFINITY$, this is a limitation with the sensor. In practice the sensor was not the exact fixed distance from the surface, therefore some degree of inaccuracy can be expected.



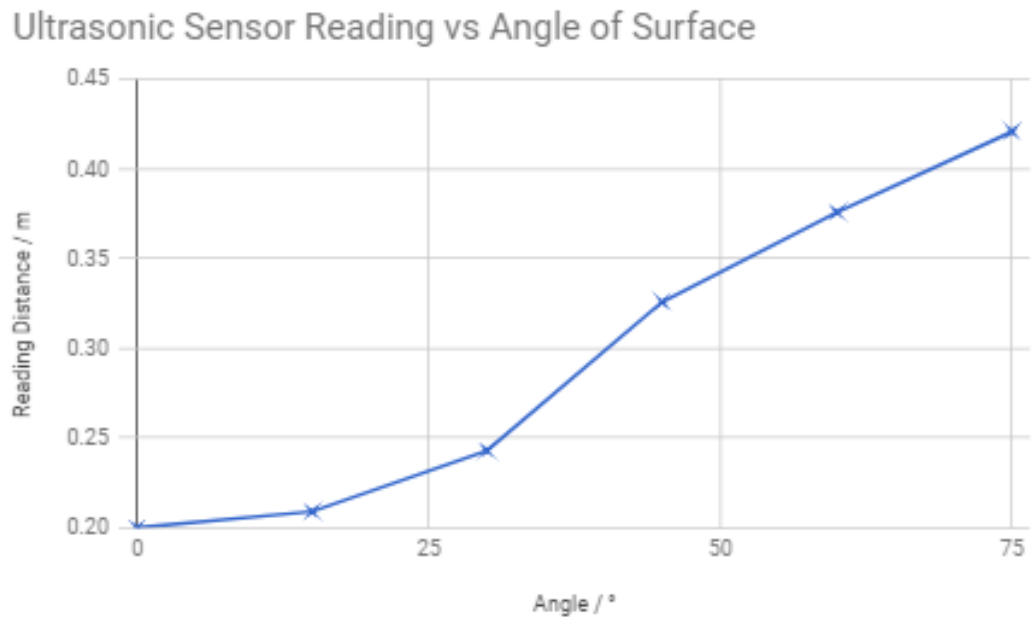

| Distance from Surface ($m$) | Average Distance Calculated ($m$) | Variance ($m^2$) |
|---|---|---|
| 0 | Infinity | 0 |
| 0.1 | 0.103 | $9.00x10^{-6}$ |
| 0.5 | 0.500 | $4.60x10^{-7}$ |
| 1 | 0.977 | $5.21x10^{-7}$ |

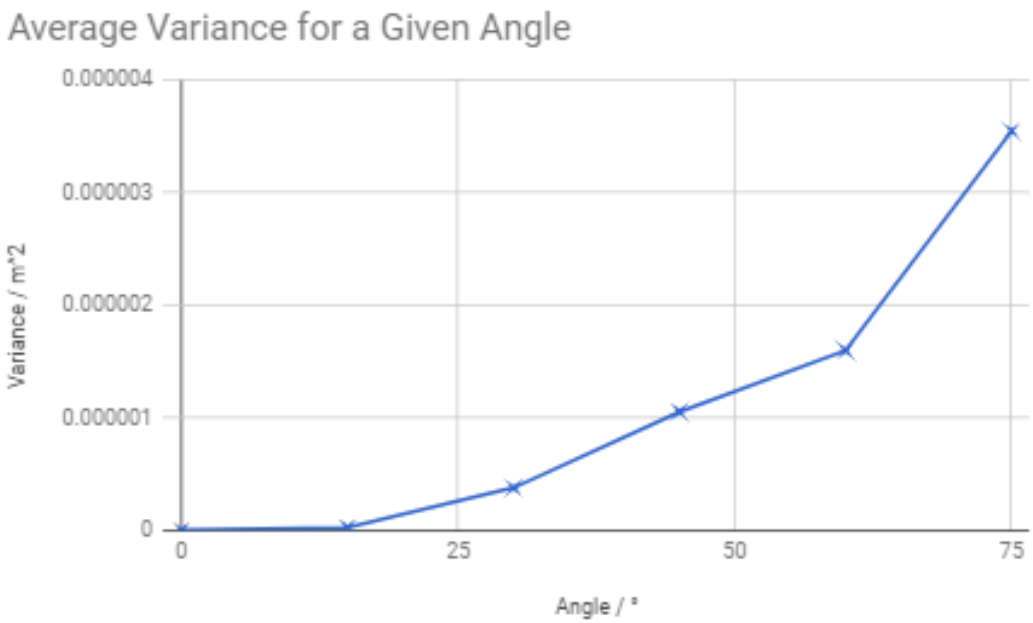

We took 400 readings at 0, 0.1, 0.5, and 1 metres from a 90° surface and took the

average to get an accurate sensor reading. The average variance is $1.33x10^{-4}$, therefore the sensor readings are very stable. However increasing the distance displayed an increase in variance, meaning that instability grows with distance. This is not a big problem if a robot is moving towards a target since its accuracy readings will increase as it gets closer but would therefore be slightly problematic if the robot is moving away from a target.

*(ii)*

To test the stability of the ultrasonic sensor as surface angle is varied we performed two separate experiments. The first one involved changing the vertical angle of a surface and the second one involved changing the horizontal angle of a surface. For both experiments the distant from the surface was kept at a constant $50cm$ and the robot took readings for $10s$. Based on the results we gathered no clear conclusion can be observed for both these two experiments, the results we gathered follow no clear pattern and fluctuate repeatedly from having a greater error percentage as the angle increases to a lower error percentage as the angle increases. The actual error percentages we observed as the angle was changed were relatively small which we means we can therefore conclude that changing the angle of the surface has little effect on the ultrasonic sensors accuracy reading and that it is relatively stable. One case which should be overlooked in this conclusion is the test case for 120° as its readings resulted in error percentage readings that were very far from the error percentage readings of the other testing cases.






The readings of the ultrasonic sensor increase with increasing surface angle. Given a fixed distance of $0.2m$ and a surface at $0°$ in the $y$ axis (perpendicular to the sensor) the sensor reads $0.2m$. As this angle is increased the readings of the sensor also increase, so increasing the angle reduces the sensors accuracy. The readings taken are very stable with an average variance of $1.32x10^{-6}$. However, variance does increase with angle, so a greater angle causes greater instability.

### 1.3 Basic PID Controller

The next step is to program the robot to position itself *autonomously* at a set distance from a vertical surface (such as a wall). In order to do this you will need to implement a closed-loop negative feedback system incorporating a PID controller. The 'reference input' $r$ corresponds to the desired distance between the robot and the surface, the 'feedback signal' $b$ corresponds to the output of the ultrasonic distance sensor, and the 'control signal' $u$ corresponds to the value that is sent to the motors. The PID controller should be of the form $u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de}{dt}(t)$ (as indicated earlier).

**Note:** *You will have to measure dt (the loop time) using the system clock.*

**Note:** *Be aware that the* `Lejos` *EV3 library has separate commands to make the motors go forwards or in reverse. This means that you will have to convert +ve and −ve control signals into suitable motor commands.*

Once you have implemented the code, you should follow these steps:

1. Set $r = 50cm$.
2. Set $K_P = 1$, $K_I = 0$ and $K_D = 0$.
3. Place the robot 100 cm from a vertical surface with the ultrasonic sensor facing towards it.

4. Run the code.

If all is well, then your robot should move slowly towards the surface slowing down and eventually stopping as it approaches the 50 cm point.

**Note:** *If your robot appears to do the opposite of what you expect, try setting $K_P = -1$. If that solves the problem, then you have implemented a positive-feedback rather than a negative-feedback loop. This means that you need to swap the polarity of the signal being sent to the motors (and reset $K_P = 1$).*

Tick the following when your robot behaves correctly (by editing the LaTeX source file, as before):

☑ Robot behaves as expected

**Question 7:** *With the code still running, what happens if, after the robot has arrived at the 50 cm point, you manually push it towards the surface?* (Worth up to 2 marks)

After the robot has arrived $50cm$ from the reference point $r$ and it is manually moved towards the surface the robots wheels start moving backwards accelerating the robot towards the reference point. The further away from the reference point the faster the wheels spin as the error $e$ is multiplied by $K_P$. Therefore, as it gets nearer the reference point, the wheels speed slows down until it stops at the reference point once more.

Now implement an outer loop that swaps $r$ between 30 cm and 50 cm every 10 seconds. As a result, your robot should now change its position at regular intervals. These step changes in $r$ will make it easier to observe the robot's behaviour for different values of $K_P$, $K_I$ and $K_D$.

**Question 8:** *With $K_I = 0$ and $K_D = 0$, what happens when you experiment with different values of $K_P$? In particular, (i) what is a 'good' value for $K_P$ (and why), (ii) what value of $K_P$ causes the robot to start to oscillate continuously backwards and forwards around the target $r$, and (iii) when it starts to oscillate continuously, what is the oscillation period? Hint: start with $K_P = 1$, then increase it gradually.* (Worth up to 5 marks)

*(i)*
A good value of $K_P$ is one that causes the robot to reach the Reference Point as quickly as possible but without overshooting. As $K_P$ is proportional to the error, a $K_P$ too high will cause the robot to overshoot and oscillate, if it is really high the robot will never converge on the reference point. However, a $K_P$ too low will take too long to reach the target. We found a good value of $K_P$ to be 5, the robot quickly reaches the reference point and does not overshoot.
*(ii)*
Increasing the $K_P$ above 5 starts to make the robot overshoot, only when $K_P$ reaches 9 does the robot oscillate continuously and never converge on its Reference Point.
*(iii)*
When $K_P = 9$ the robot oscillates with an oscillation period of 1.4 seconds.

The value of $K_P$ that causes the robot to oscillate backwards and forwards continuously

is known as the 'ultimate gain' $K_U$, and the oscillation period is designated as $T_U$.

**Note:** *Be aware that $T_U$ is a measure of time, not frequency. I.e. if the robot oscillates backwards and forwards at a rate of five times a second, then $T_U = 1/5 = 0.2$ secs.*

**Question 9:** *What is the relationship between the 'good' value of $K_P$ that you discovered by experimentation and the value of $K_U$ that you measured?* (Worth up to 5 marks)

The relationship between the 'good' $K_P$ (5) value and the value of $K_U$ (9) is that our $K_P$ is roughly half of the $K_U$. This is inline with the Ziegler-Nichols method. The ultimate gain is a value of $K_P$ in which the robot oscillates indefinitely, neither converging or diverging. So increasing the $K_P$ beyond 5 up to 9 brings the robot closer to its ultimate gain.

## 1.4 PID Tuning

There are several methods available for tuning the parameters of a PID controller. The *manual* method involves setting $K_P = 0.5K_U$, $K_I = 0$ and $K_D = 0$. Next, $K_I$ is gradually increased to improve the convergence. Then $K_D$ is gradually increased to improve the responsiveness.

A more formal tuning approach is known as the *Ziegler-Nichols* method. Once $K_U$ and $T_U$ have been determined, the PID gains may be set according to the following heuristic:

| | $K_P$ | $K_I$ | $K_D$ |
|---|---|---|---|
| **P** | $0.5K_U$ | | |
| **PI** | $0.45K_U$ | $0.54K_U/T_U$ | |
| **PID** | $0.6K_U$ | $1.2K_U/T_U$ | $3K_UT_U/40$ |

**Question 10:** *What values of $K_P$, $K_I$ and $K_D$ are given by the Ziegler-Nichols method for a P, PI and PID controller (based on the values of $K_U$ and $T_U$ you measured previously)?* (Worth up to 5 marks)

$K_U = 9$
$T_U = 1.4$
$P, K_P = 4.5$
$PI, K_P = 4.05, K_I = 3.47$
$PID, K_P = 5.4, K_I = 7.71, K_D = 0.945$

Confirm the effectiveness of the Ziegler-Nichols method by setting the values for $K_P$, $K_I$ and $K_D$ in your controller, and tick the following:

☑ The robot moves quickly to the target and stops with minimal overshoot

**Question 11:** *Using the values for $K_P$, $K_I$ and $K_D$ given by the Ziegler-Nichols method, what happens when you decrease the sampling rate (e.g. by introducing a 'wait' function in the loop)? What is the significance of your observations?* (Worth up to 5 marks)

Applying a 'wait function' decreased the robots sampling rate causing the robot to adjust its position less often. This caused the robot to adjust more severely and also begin to overshoot from the reference point, decreasing the sampling rate further caused the robot to oscillate and diverge from the reference point. This is because the robots 'reaction time' is increased, the difference in values between the current readings and the previous readings is large causing a severe adjustment. If the robot had passed the reference point between the two readings an even more severe change in direction was necessary, again a small enough sampling rate ensured the robot never converged. The significance of the sampling rate then is that the quality and accuracy of the control decreases as the sampling rate decreases. In contrast we found a sampling rate that was too high could overload the robot with too many differing readings. A sampling rate of $100Hz$ meant the robot was stable and consistent.

### 1.5 PID-based Steering

Finally (for Part I), reposition your ultrasonic sensor so that it faces sideways from the robot.

Now re-program your robot with a PID controller that maintains a fixed distance from a wall as the robot travels along parallel to it. This will require the speed to be set at some fixed value, and the PID-based control loop will handle the steering. Use the Ziegler-Nichols method to determine appropriate values for $K_P$, $K_I$ and $K_D$ (this can be done while the robot is stationary and changing the target distance a few centimetres once every 10 seconds, as before).

**Note:** *It is particularly important that the sensor is mounted well forward of the driving wheels, otherwise it will not be sensitive to changes in direction when the robot is stationary.*

**Question 12:** *What values for $K_U$ and $T_U$ did you measure, and what are the resulting values for $K_P$, $K_I$ and $K_D$ given by the Ziegler-Nichols method?* (Worth up to 5 marks)

$K_U = 7$
$T_U = 1$
$P, K_P = 3.5$
$PI, K_P = 3.15, K_I = 3.78$
$PID, K_P = 4.2, K_I = 8.40, K_D = 0.525$

Confirm the effectiveness of these values for $K_P$, $K_I$ and $K_D$, and tick the following:

- ☑ The robot travels along a wall maintaining a constant distance from it

### 1.6 Summary

You have successfully completed Part I of the assignment, and you should now have all of the skills necessary to move on to Part II.

# 2 Part II: Maze-Running Competition

## 2.1 Objective

The aim of Part II of the assignment is to combine the theoretical knowledge you have obtained in the lectures with the practical experience you have acquired in Part I to design and implement a robot that is capable of competing in a simple maze-running competition.

The challenge is to navigate a corridor (that will be set up in the robot arena) in the fastest time possible and without touching the walls. The precise layout will not be revealed until the final lab session. However, you will be able to practice in the arena beforehand. The two walls of the corridor are painted different colours so that, if necessary, your robot can determine if it is going in the correct direction.

You need to decide which control principle(s) to use and how to set up your robot's sensors and actuators. Also, since it's a competition, you will need to to think about how your robot manages 'risk', i.e. is it better to be slow-and-careful or reckless-but-fast? You will probably need to experiment with various alternatives before deciding on a final approach.

**Note:** *You will need to adopt good working practices for (i) organising your team and (ii) software version control. The latter is important as robots are notorious for failing after a so-called 'improvement', so being able to revert easily to an earlier version will save a lot of pain and grief.*

**Note:** *Each Lego kit contains two ultrasonic sensors.*

**Note:** *The maze will be a simple winding corridor. There will be no dead-ends, no loops and no junctions. However, there may be chicanes and small breaks in the wall.*

## 2.2 The Competition

In order to give every team an equal chance, the competition will be organised into a number of leagues, each consisting of several teams (and their robots). Marks will be awarded based on each robot's performance **within its league**. This will mean that each team will be competing against teams that have had an equal amount of time devoted to the development of their respective robots. The competition will take place during the final lab session.

**Note:** *If there is time at the end of the league battles, the winning robot in each league will compete again to find the overall champion. This 'championship' contest will not count towards the marks for the assignment, but a (small) prize may be awarded.*

As you can imagine, running such an event with a large number of teams/robots requires very precise time management, so we will be issuing a strict timetable for the final lab session. This should appear on MOLE the week before. It is essential that you prepare carefully for your designated time slot (otherwise you may lose marks - see below).

The rules for the competition are as follows:

1. Any number of practice runs may be made (subject to the availability of the arena).

2. Each team must register their arrival at the arena (with their robot) 10 mins prior to their league's designated time slot, after which no further technical development will be permitted.
3. Each team will be called forward in turn to place their robot on the starting line.
4. An 'official' run for each team's robot will be timed by the lab demonstrators.
5. If a robot fails to complete a run within a **3 minute time limit**, the demonstrators will record the furthest position it reached.
6. It is permitted for a designated team member to rescue their robot and place it back on the course at the place where things went wrong **twice** per run without penalty, but the clock will keep running. A third rescue is not permitted, instead the run will be terminated and the position reached will be recorded.
7. Failure to arrive at the arena at the designated time will earn a penalty (see below) and may mean that you do not get a run.

Marks will be awarded as follows:

- 5 marks will be awarded for a run completed within the 3 minute time limit.
- 5 marks will be awarded for a 'clean' run. (i.e. no walls touched and no rescues).
- 10 marks will be awarded to the team with the best run within their league, 9 marks to the second best team, and so on. A team who fails to arrive at the arena at their designated time will be ranked last in their league.

**Question 13:** *What was the result of your official attempt?* (Worth up to 20 marks)

| Completed | Wall Touches | Rescues | Time | Distance | Rank |
|---|---|---|---|---|---|
| Y | 3 | 3 | 71s | 10.5 meters | 1st |

## 2.3 Your Team's Solution

**Question 14:** *In terms of your robot's software architecture, what principles did you explore, and what was the final approach taken (include a system diagram)?* (Worth up to 15 marks)

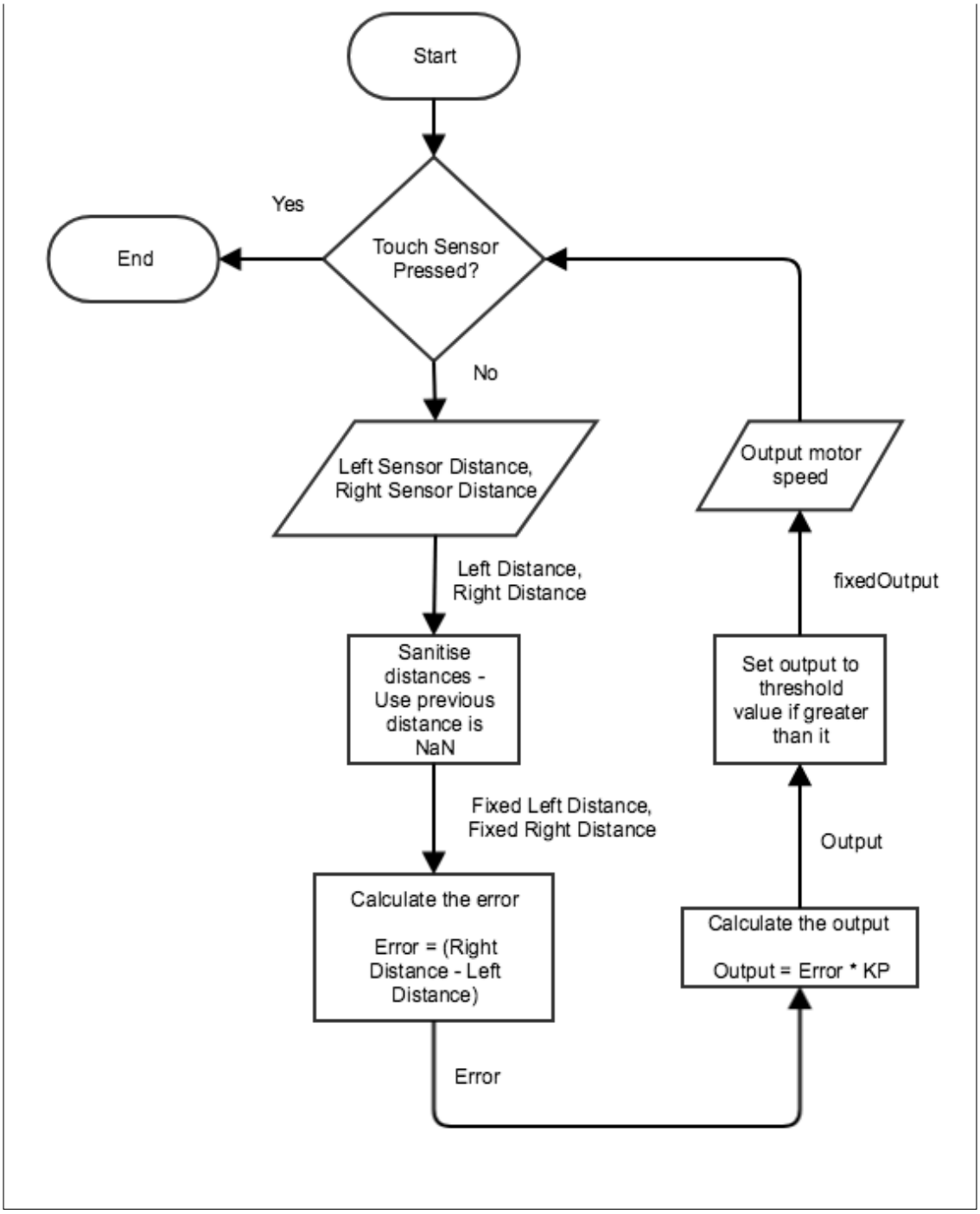
Our final system uses one while loop with the controller calculating the output that runs until the touch sensor is pressed, this allows us to disable our robot without having to restart it everytime we want to make adjustments to parameters within our code. The loop has a sampling rate of $100Hz$, which means the robot does not get overwhelmed with readings but enough readings are taken for smooth movement. Additionally we used $System.nanotime()$ to get the most accurate deltaTime possible. Using $System.currentTimeMillis()$ proved to be unreliable in testing. Before the controller calculates the output the validity of each sensors readings are checked as occasionally $NaN$ or $Infinity$ values are read. If either of these values are read the previous sampled value is used instead, as the readings are taken every $10ms$ the previous value is accurate enough to what the actual reading should be.

To navigate the maze our robot maintains a fixed motor speed of 300. This is a good tradeoff between speed and accuracy. Also, our controller does not have a fixed reference point, rather it is the midpoint between the two walls of the maze. In testing we found that hugging one wall meant the robot could not fit through tight spaces and had little room for error. Therefore, the error is $rightSensorDistance - leftSensorDistance$, when the two are equal the robot must be in the centre of the two walls.

To calculate the output we found that $I$ and $D$ in the PID Controller added inaccuracy to the robot, the robot needed an instantaneous reaction without looking at past errors or predicting future errors therefore our final P value was 0.4 where $I$ and $D$ equal 0. Occasionally the sensor would misread extreme values or when turning a severe corner the sensor would read an unusually high value, to combat this we used a threshold for the output of 150. This meant that the robot could react quick enough to change in most cases, but did not react too quickly and then overshoot.

Once there are two cases that determine the left and right motor speed:

- $Output > 0$
  $Leftmotorspeed = fixedSpeed + absolute(output)$
  $Rightmotorspeed = max(0, fixedSpeed - absolute(output))$
- $Output < 0$
  $Leftmotorspeed = max(0, fixedSpeed - absolute(output))$
  $Rightmotorspeed = fixedSpeed + absolute(output)$




**Question 15:** *What was your robot's final physical configuration (include a photo)?* (Worth up to 5 marks)

As can be seen in the pictures below there were two types of setup we experimented with, the first case case was positioning the sensors so that they were aligned at 90° $7cm$ in front of the robot. The sensors were mounted in front of the robot so that the robot was responding to changes in the maze slightly in front of it. This meant that the robot would start turning the corners in time and not crash into a wall ahead of

it.

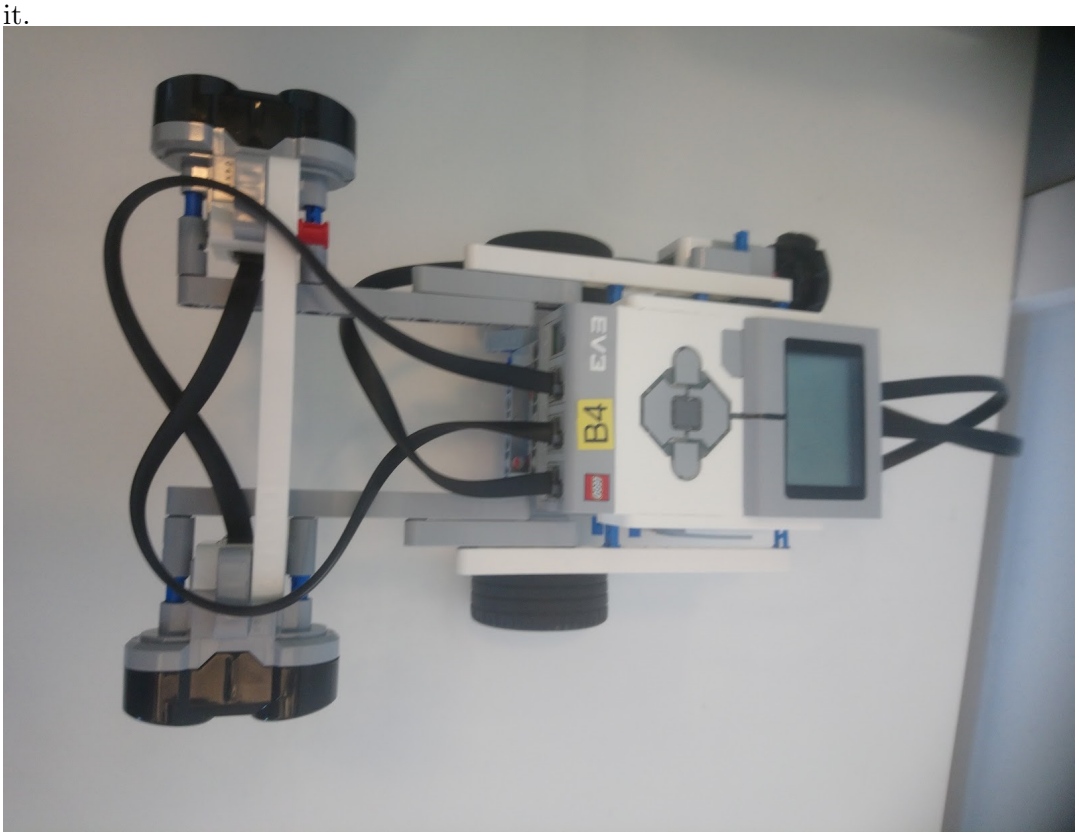



Figure 1: INITIAL DESIGN

An alternative approach we tried was angling the sensors at 45° in attempt to also be looking ahead of the maze and around corners. As predicted by looking at the stability of readings in question 6, this caused decreased the sensors reliability and decreased the performance of our robot. Ultimately then we used 90° sensors placed ahead of the robot.


Figure 2: ALTERNATIVE DESIGN

Another consideration we took into account was the width of the robot. At tight spaces in the maze the robot would touch the wall just because of its physical limitations. To combat this the 'arms' holding the sensors were stacked inwards in order to reduce the width of the robot as much as possible.
Finally, the robot used a touch sensor which drastically helped out testing. On press of the sensor the robot was stopped, without the use of this sensor we had to break out of the program manually and consequently restart the robot.

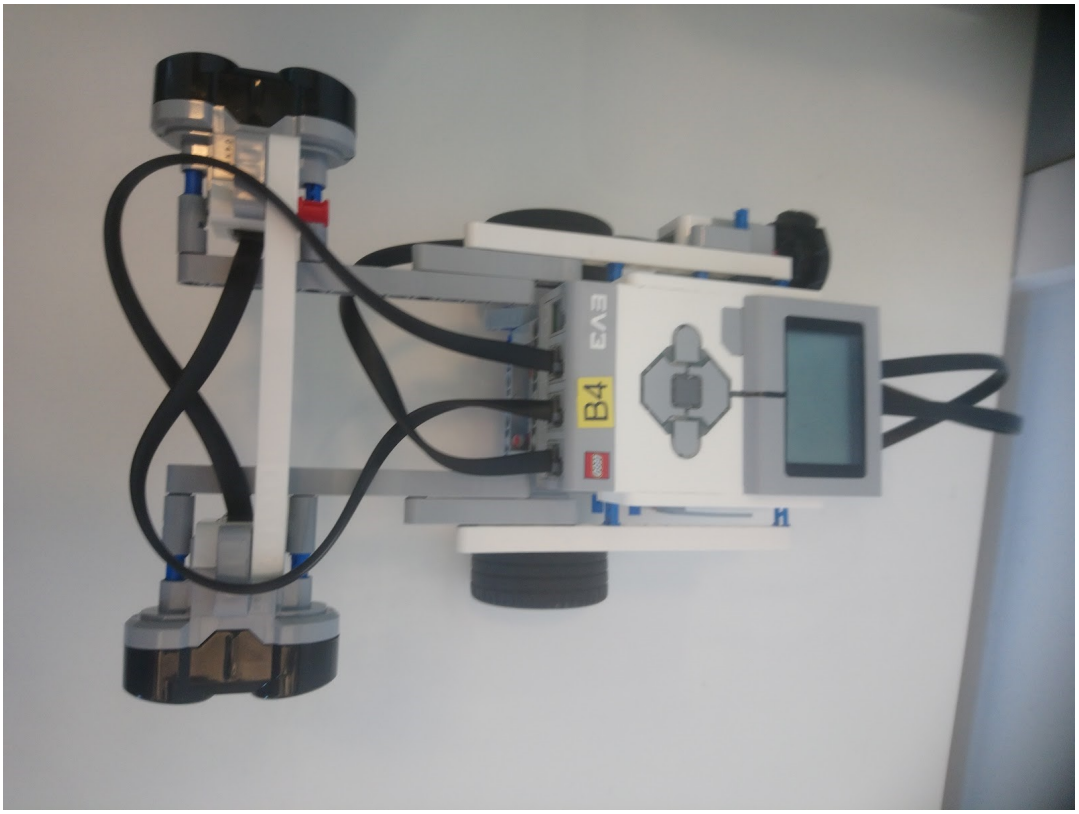



Figure 3: FINAL DESIGN

**Question 16:** *Are there any points you wish to make about your robot's performance in the competition? For example, if you had had more time, what might you have done differently?* (Worth up to 5 marks)

Our robot performed successfully in the maze running competition, we placed first in our league and competed in the grand finale where we placed fifth. There were however some issues which we encountered. Although we performed well we still hit the wall during our assessed run three times. Firstly, our robots speed was too fast, this meant that the robot did not have enough time to respond to sudden changes in the maze such as sharp corners. As a result we should perform further tests to find the optimal speed for all situations.

Finally, how did you organise your team? I.e. what was each member's role and responsibilities, and what was each person's contribution as a %? Please fill in the Table below:

| **Team Member** | **Role** | % |
|---|---|---|
| Ho Yin Angus Ng | - | 0 |
| Hyun Han | Assisted with robot experimentations and programming of solutions. | 22 |
| Matthew Kinton | Main programmer and answered lots of the questions | 27.5 |
| Malvin Todorov | Worked on the report and assisted with robot experimetations and programming. | 23 |
| Leon Singleton | Helped work on programming the the robot to solve all experimentations. Worked on collating test data and answering many questions in the report. | 27.5 |