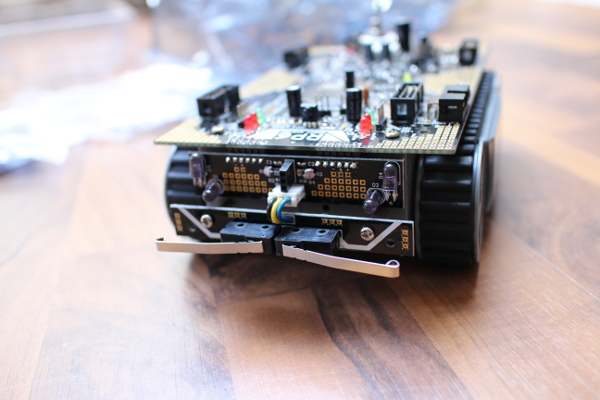
**TECHNICAL REPORT**

***Week 4 & 5 - Assignments***

Embedded Systems 3 (ES3)

**RP6 Robot**



**LECTURER:** Gerald Hilderink

**DATE:** 4/10/2019

**AUTHORS:** Sam Jarvis - 3734811

Yasser Alhazmi - 405156

*Abstract*

The following assignments implements the SHARP distance sensor in an RP6 driving along the wall scenario. The sensor operates on infrared light by flashing a beam of light and using the reflection of it to calculate distance. The assignment also introduces the PID controller system and is a natural continuation of previous weeks assignments.

Contents

[Week 6 1](#_Toc23718130)

[Introduction 1](#_Toc23718131)

[PID controllers 1](#_Toc23718132)

[Procedure 2](#_Toc23718133)

[Apparatus 2](#_Toc23718134)

[Conclusion 2](#_Toc23718135)

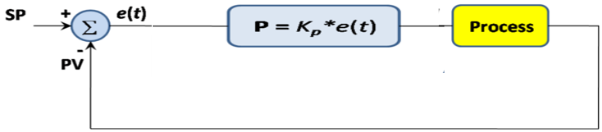
# Week 6

## Introduction

The assignment integrates the knowledge learned from weeks 4, 5, and 6. First, a transfer function must be implemented in order to convert the SHARP sensor output to distance (Week 4). Then, the noise generated by the SHARP sensors is eliminated using the moving average filter (Week 5). Finally, a PID controller (Week 6) that utilises the transfer function and moving average filter will facilitate the driving of RP6 along the wall.

Since the processes of calculating the transfer function and moving average filter were explained in previous weeks reports, this week focusses on PID controllers and the overall process.

### PID controllers

A proportional–integral–derivative (PID) controller is a feedback control system that uses the output of the previous iteration to further stabilize the system. It is frequently used in regulatory controllers, with some studies approximating their use in 97% of regulatory controllers.

The set point (SP) of a PID controller is the desired value or level of the system. The error value is the difference between the SP and process variable (PV). The PV is the value detected by a sensor after applying a process. There are three responses to error in a system within a PID controller:

* Proportional response (Kp): the speed of correction, a proportional ratio of the output based on error value.
* Integral response (KI): the speed of offset elimination one the system is stable, responsible for driving the steady state error to zero.
* Derivative response (KD): the speed of damping, the rate of change over time.

For example, if a PID controller has a KP of 2, SP of 10 and a PV of 2, then the error is 10 – 2 = 8. The controller output will be *error x KP = 8 x 2 = 16*. If in the process is implemented correctly the PV in the next iteration might be 4[[1]](#footnote-1), then the output of the error = 6 and controller output *= 6 x 2 = 12*, and so on.

The value of PID responses depends on the requirements of the project and characteristics of the sensor. The system engineer estimates it based on trial and error. Incorrect values might render the controller unusable and disturb the system instead of stabilizing it. For example, increasing KP too much will increase the oscillation and decrease the damping effect of the controller and the system might never reach the SP. In addition, depending on the project, a response value might be set to zero (not used). For example, due to KD’s high sensitivity to noise, control systems that interact with noisy signals might not use it at all to maintain the overall stability of the system. Eliminating the KI is also possible if maintaining a steady state (stable state with a small offset) is not critical for the system. All this shows that the embedded system engineer must choose the response values of a PID controller after thorough testing and experimenting.

## Procedure

The necessary apparatus for the week 6 project is similar to that of weeks 4 and 5.

### Apparatus

* 1x RP6
* 2x SHARP sensors

With all the equipment tested and verified, the transfer function, moving average filter and PID controller can be implemented to achieve the requirements of the final project.

First, the program starts by initializing essential RP6 components and I2C communication. The program then enters a while loop that does the following:

1. The SHARP sensor values are read.
2. These values are converted to a distance via the transfer function.
3. A moving average filter is applied to the distance values to eliminate noise and produce a stable signal for the PID controller.
4. The moving average of both sensor values is printed to the LCD.
5. The SP of the system is set to 15 cm.
6. KP and KD are both set to 1 at initialization, that is, they take no effect. However, these values can be controlled by the user while the RP6 is executing the PID controller.
7. The error, that is, the difference between the desired value and the result of the moving average filter is calculated.
8. The difference between the last iteration’s error and the current iteration’s error is calculated for KD to use at a later stage of the process.
9. The correction that the system needs to implement in order to reach the set point is calculated as follows: KP is multiplied by the error and KD is multiplied by the difference between the last error and the current error (step 8). These results are added together to produce the final correction value.
10. The PID controller implements this correction by subtracting it to the speed value of the left motor and adding it from the speed value of the right motor. These speed values are then input into the moveAtSpeed() function. Each motor, that is, the left tread and the right tread will move with the speed calculated by the PID controller.
11. The process is then repeated continuously until the RP6 is driving in a stable state along the wall at the desired distance from the wall, perhaps with a slight offset since KI is not implemented in this PID controller. However, accuracy is not essential for the safety or operation of this system, so this is not problematic.

The PID controller is implemented in code as this while loop.

Although oscillation could not be avoided, it was reduced to the minimum by choosing an appropriate KP and KD values.

## Conclusion

Although the PID controller is a robust control system, it was not implemented without error. Some errors were solved however, some were not solved and still exist in the RP6 system. First the solved problems will be discussed. Then, the unsolved problems, and finally, the improvements that could be implemented given more time will conclude this chapter.

The problems that were solved were:

* Radical correction despite KP and KD values of 1. This was caused by high correction speed values calculated by the PID controller. To further explain, if the normal speed of the right motor is 20, and the correction speed of the right moto is 95, then the PID controller will try to increase the right motor speed to 115. The RP6 cannot implement such a speed, it is out of range.

To solve this problem, an “if” statement was implemented. It checked whether the correction speed was higher than some value. This was tweaked, but ultimately it was set to 70. If the correction value was higher than this, it was set back to 70. This way, the maximum speed the motor could be was 90.

The problems that were not fixed:

1. The RP6 cannot find the desired distance if it does not start close to it. This can be caused by multiple factors and the conclusion is that it is caused by a combination of many since no single factor seemed to cause it. In other words, the RP6 can successfully follow a wall at the desired distance but, it is not effective at finding the desired distance.

In an attempt to fix this, the following solutions were implemented:

* The KP and KD values were tweaked however, this led to no success. The correction speed was already too fast and thus, a high KP value exacerbates the problem. A high KD value has little to no effect since the KP value is not causing overshoot.
* The difference between the error of the front and back SHARP sensors was used as the error as opposed to the error of either one sensor. The logic of this solution was flawed however, this was not known until after implementation and testing. This however, was extremely efficient at straightening the RP6 horizontally and thus it stabilized oscillation. It was not a successful solution though, since as long as the RP6 was horizontal relative to the wall, the error was zero. So, essentially, the desired distance was no longer considered.

Improvements (may also eliminate existing problems):

1. The main improvement is based upon an attempted fix for the unsolved problem. The difference between the error of each sensor was calculated and used as the error for the PID control loop. This efficiently corrected the horizontal offset of the robot but ignored the desired distance. If this could be implemented at the desired distance, it may prove to be robust and stable.

1. These are random numbers to show that we understand the concept. In reality, the value will be dependent on both the process output as well as the environmental changes detected by the sensor. [↑](#footnote-ref-1)