Fall Semester 2015

Line following robot

Group 2

2. Semester IT-Technology

Group members: Benjamin Nielsen - Henrik Jensen - Martin Nonboe - Nikolaj Bilgrau

Supervisor: Jesper Kristensen - Steffen Vutborg

Uddannelse i virkeligheden

IT-technology
Sofiendalsvej 60
9200 Aalborg SW
http://www.ucn.dk/

Title:

Line following robot

Project Period:

2. Semester | Spring semester 2016

Projectgroup:

Group 2

Medvirkende:

Benjamin Nielsen
Henrik Jensen
Martin Nonboe
Nikolaj Bilgrau

Supervisor:

Jesper Kristensen
Steffen Vutborg

Pages: TBD

Appendices: TBD

Completed TBD

Introduction

Project is written by:

Benjamin Nielsen	Henrik Jensen
Martin Nonboe	Nikolaj Bilgrau

Table of Contents

Glossary

ADC Analog-digital conversion

PID Proportional-integral-derivative controller

Requirements specification 1

The following section will describe the specific requirements that have been decided to fulfil to the general requirements as shown in the project description.

- Project must include light sensors
- Implement motor control
- Should make use of the Pic32 MCU
- Software will be written in MPLAPX
- Autonomous operation
- The product must make use of feedback concept e.g. a PID algorithm
- A function & performance test is to be conducted

Hardware section 2

2.0.1 Hardware diagram

2.1 Description of the hardware structure and functionality

2.1.1 Selection of sensor

Name	QRE1113 board	OPB706A	OPB704
Max sensor distance	3mm	1.27mm	3.8mm
Forward current	50mA	20mA	40mA
Mounting	On print	THT	In casing
Price	19.43DKK	26.90DKK	42.55DKK
Notes			

Table 2.1: Table of a selection of sensors

TBD Beskriv sensorer og hvorfor vi har valgt denne

2.1.2 OPB704 Sensor

Sensor choice for the line following robot will be the OPB704. The choice was logical because of the availability. It was confirmed in the brainstorming process that the sensor was reliable and had a wide range of applications. 3D printing the mount was ideal for the purpose of the sensor and the task ahead.

TBD Beskriv hardware struktur og funktion samt alle underdele

2.1.3 Analog-to-digital converter (ADC)

The purpose of the ADC is to convert the analog data from the sensors to digital data that can be manipulated by a computer - this allows data received from the bluetooth transmitter on the robot to be processed into readable data more easily, this is great for showing how the sensors are reacting to the environment. The sensors themselves cannot discern what they actually need to read, the sensors just read anything they can see and send that signal.
Analog signals can have a significant amount of noise - since any received noise is interpreted as part of the signal, so a digital signal is not only more easy to work with, it will also have less chaotic frequencies. This will make for more accurate readings on the tachometer on the robot, which allows even more finely tuned monitoring of the robot and its working processes.

2.1.4 ADC diagram

TBD

Our take on the ADC

TBD

Software section 3

3.0.1 Software diagram

TBD Softwarediagram

3.1 Description of the software structure and functionality

TBD Softwarebeskrivelse og underafsnit

3.1.1 Description of the PID controller

A PID controller continuously calculates an error value as the difference to a reference point and a measured process variable.
PID is an abbreviation for a proportional-integral-derivative controller, it is a control loop feedback mechanism. The controllers job is to minimize the error value for the given devices running time. In the case of this project the reference point is the line and the PID will power up the engines to steer accordingly to said reference point.

$$\mathrm{F}(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de(t)}{dt}$$

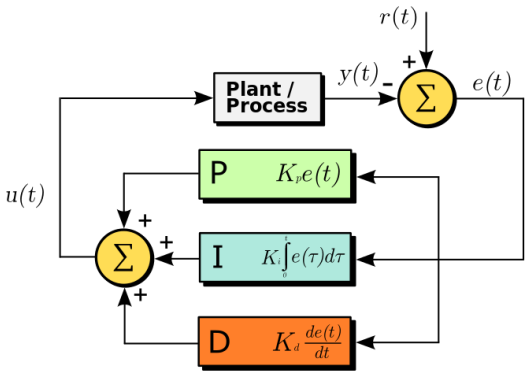

Figure 3.1: Block diagram, showing the idea of PID controller. Credits:

Proportional control(P)

The proportional term creates an output value that is proportionally related to the current error value, this value can be tuned by multiplying the error by a constant K_p. A high proportional gain results in a large change in the output for a given change in the error.

$$P_{\text{out}} = K_p\, e(t)$$

If the proportional gain is too high, the system can become unstable. Contrarily, a small gain will result in the device adjusting too slowly, which decreases overall efficiency and in the case of this project, it will end up being detrimental to the steering accuracy.

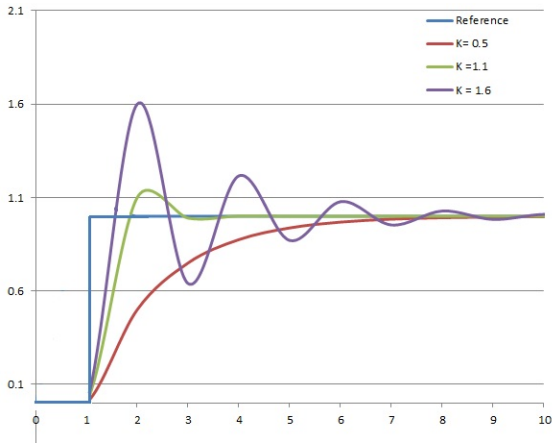

Figure 3.2: K_p with 3 values. (K_i, K_d held constant) Credits:

Integral control(I)

The integral controller is contributing proportionally to both the magnitude of the error and the duration of the error.
The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously.

The controller output equals the accumulated error multiplied by the integral gain(Ki)

$$I_{\text{out}} = K_i \int_0^t e(\tau)\, d\tau$$

The integral term accelerates the movement of the process towards the reference point. Since the integral term correlates to accumulated errors from the past, it can cause the present value to overshoot the reference value.

Derivative control(D)

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_d. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, K_d.
TBD (billede skal rettes ind til teksten) The derivative term is given by:

$$D_{\text{out}} = K_d \frac{de(t)}{dt}$$

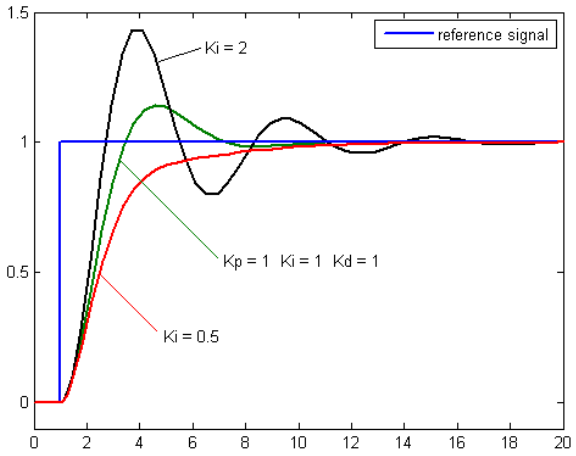

Figure 3.3: K_i shown with 3 values. Credits:

The derivative action predicts system behaviour and utilizes this to improve the settling time and stability of the system. An ideal derivative is not causal, so that implementations of PID controllers include an additional low pass filtering for the derivative term, to limit the high frequency gain and noise.

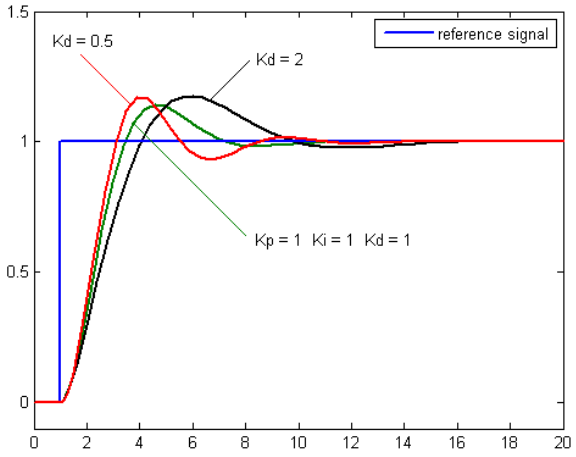

Figure 3.4: K_d shown with 3 values. Credits:

Loop tuning

Tuning the loop is the term used to describe the adjustments of the PID's control parameters (proportional band/gain, integral gain/reset, derivative gain/rate) to the optimal values for the given control scheme.
Stability is the first requirement; however systems can differ greatly, and different applications may have different requirements and these may even conflict with each other. For example, high speed and high accuracy often cancel each other out, because high speed may cause overshooting, while high accuracy is slow.
The ideal realistic behaviour is both as fast as possible, while also having minimum overshoot and oscillation.

Even though the process seems simple, with only three variables, it can be challenging to achieve, because it must satisfy the criteria despite being within the

limitations of PID control. While adjusting the PID can seem conceptually intuitive, and while most PIDs may perform acceptably with default controls, they may very well also have an unsatisfactory performance.
This can generally be fixed through optimisation and tuning, either through computer simulations or manual testing. In our case, we used manual tuning of the numbers.

Stability

If the parameters of the PID controller are set incorrectly the process input can become unstable. This means the controllers output becomes divergent, this can be limited by saturation and mechanical breaking.

Manual tuning

When a system must be online at all times a method for tuning is, to first set K_i and K_d values to zero. Increase K_p until the loop output oscillates, setting K_p at approximately half the value for a "quarter amplitude decay" type response.
Then increase K_i until any set off is corrected in sufficient time for the process. Adding too much K_i will however cause an instability. Finally, increase K_d, if required at all, until the loop is acceptably quick to reach its reference after a load disturbance.
A fast PID loop tuning process usually overshoots slightly to reach the reference point faster. In the case of systems that can't accept overshoot, an over-damped closed-loop system is best suited, which requires K_p setting significantly less than half that of the K_p setting that was causing the oscillation.
TBD (Vores fremgangsmåde med PID alt efter om "D" skal bruges)

Table 3.1: Manual tuning

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor change	Decrease	Decrease	No effect in theory	Improves if K_d is small

Table 3.1 explained

Table 3.1 gives an informative overview of what the different parameters does when tuned manually.
1. To decrease the rise time, use K_p .
2. To reduce the overshoot and settling time, use K_d .
3. To eliminate the steady-state error, use K_i .

Test 4

4.1 Test

Conclusion 5

TBD Konklusion

Appendices 6

6.1 Group collaboration agreement

6.1.1 Contact Information

Table 6.1: Contacts

Benjamin Nielsen	TBD	@: yipiyuk5@gmail.com
Henrik Jensen	Tlf: 28568934	@: henrik_kort@hotmail.com
Martin Nonboe	Tlf: 23827566	@: nonsens_4@hotmail.com
Nikolaj Bilgrau	TBD	@: nikolajbilgrau@gmail.com

6.1.2 Workflow

- Every friday after 12:00 is expected work consisting of three hours.
- If you aren't able of attending for scheduled study day. - Notice must be given to the project team.

6.1.3 Milestones and goals

TBD Milestones

6.1.4 Deadline

- Hand in June 7th.

List of references 7

TBD list of references PID:

List of Figures

Page

List of Tables

Page