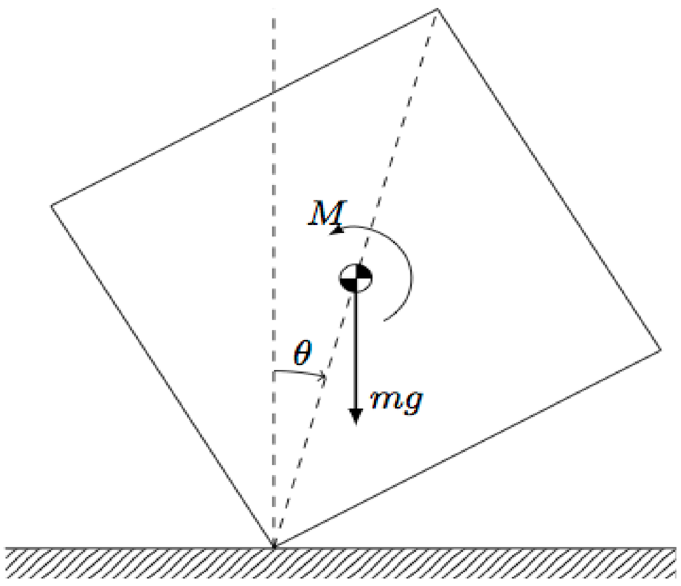
DEGREE PROJECT, IN MECHATRONICS , FIRST LEVEL
STOCKHOLM, SWEDEN 2015

StandingCube

THE SEARCH FOR A GREAT ARDUINO FRAMEWORK IN REGARDS TO SELF BALANCE

DAVID TENNANDER & VIKTOR GJUROVSKI



KTH ROYAL INSTITUTE OF TECHNOLOGY

SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENTC

StandingCube

The search for a great Arduino framework in regards to self balance

DAVID TENNANDER
VIKTOR GJUROVSKI

Bacherlor's Thesis in Mechatronics

Supervisor:	Daniel Frede
Examiner:	Martin Edin Grimheden
Approved:	2015-month-day

TRITA xxx yyyy-nn

Abstract

This thesis describes the development of a open source platform for developing self balancing applications on the Arduino chip. It consists of a theoretical part laying the ground work for the later part where the development and construction is described. As of this point the thesis is at it's 50 % stadium and there for do not have any results yet. Otherwise this sentence would give a short briefing of the discussion and conclusion of the result. The GitHub project can be followed at: `NYI`

Sammanfattning

Den stående kuben
Strävan efter ett underbart Arduina biblotek för själv balansering

Denna rapport beskriver utvecklingen av en open-source plattform för utväckling av självbalanserande system, med Arduino som bas. Rapporten består av en teoretisk del som lägger grunden till den andra delen där implementering och konstruktion av demonstrationssystemet beskrivs. Just nu är projektet i sin 50%-fas och har därför ännu inga resultat. Annars skulle denna menning presentera en sammanställning av rapportens diskussion och slutsats av resultatet. GitHub projektet går att följa på: `NYI`

Preface

We want to thank Daniel Frede for he's support and fine thinking!

David Tennander
Viktor Gjurovski
A sunny "Maskin", Mars, 2015

Contents

Nomenclature

Symbols

Symbols	**Description**
r	Radius (m)
x	Position $[m]$
$\dot{x}$	Velocity $[m/s]$
$\ddot{x}$	Acceleration $[m/s^2]$
$\hat{x}$	Estimate of position $[m]$
r	Radius with respect to contact point on surface $[m]$
θ	Angular deviation from equilibrium point $[rad]$
M	Motor torque
I	Moment of Inertia
σ	Standard deviation
ϵ	Measurement Error
Δt	Sampling interval
f_s	Sampling frequency

Abbreviations

Abbreviation	**Description**
CAD	Computer Aided Design
CAE	Computer Aided Engineering
IMU	Inertial measurement unit
MPU	Microprocessor unit
RMS	Root mean square
LQE	Linear-quadratic estimation
IIR	Infinite impulse response
RW	Reaction wheel
QS	Cube structure
SDA	Serial data line
SCL	Serial clock line

Chapter 1

Introduction

1.1 Background

Self balancing control systems is a growing part in today's technology as seen in the growing usage of Segway like verticals. But unfortunately the open-source world have not come up with a solid starting block, to ease the start up of solutions in this space. In the open source world big projects like *i2cdevlib* [Rowberg, 2015] trys to build a platform to gather communications protocols for different platforms. But a big reliable framework have not come up for IMU to Arduino projects.

1.2 Purpose

The purpose of this thesis is to find a framework for self balancing open-source projects. The aim is to create a library for supporting this type of projects and a demonstration of the framework will be a self balancing cube using reaction wheels.

The project consists of two parts, one engineering part and one research part. The engineering part focuses on the construction method and software development. While the research part will try to answer the question:

If we use a linearised control model for a non-linear balancing problem containing reaction wheels; What is the minimal overshoot possible for a stable system with a control system on an Arduino Uno chip? What forced angular deviation from the equilibrium point will create instability for this linearised system? Finally, will different magnitudes of overshoot have any affect on maximum stable angular deviation and if so, how?

1.3 Scope

The thesis aim is to build a platform to help developers construct balancing projects with reaction wheels.To prove our success a demonstration will also be built consisting of a cube balancing on its edge.

1.4 Method

This thesis project uses an Arduino Uno as micro processor. The framework will be built in `C` and `C++` witch is the standard for Arduino libraries [Ard,].

The construction and development of the cube will be done with use of CAD and simulation software. In this thesis the CAD software that will be used is Solid Edge and CS Eagle [Solid Edge, 2014] [EAGLE, 2014]. The simulation will be built in MatLab/Simulink [MATLAB, 2014].

Chapter 2

Theory

In this chapter the theoretical model for our project is derived and the theoretical background on witch the project stands on is presented.

2.1 Mechanical system

The mechanical system that describes balancing one surface of the cube in one dimension is shown in Figure 2.1. The dynamics of the system follows

$$I_o\ddot{\theta} - m_{tot}gr\theta = -M_{motor} \tag{2.1}$$

where I_o is the total moment of inertia MOI, including reaction wheels, RW, and cube structure(motors + cube frame), QS, around the contact point with the ground:

$$I_o = I_{RW} + m_{RW}r^2 + I_{QS} + m_{QS}r^2. \tag{2.2}$$

θ is the angular deviation from the cubes equilibrium point, m_{tot} is the total mass of the cube and M_{motor} is the torque from the DC motor. The torque from the DC motor follows

$$M_{motor} = k_2\phi * i_a \tag{2.3}$$

where $k_2\phi$ is the torque constant and i_a is the current running through the motor.

2.2 Control System

The system given by (2.1) is highly unstable at $\theta = 0$. To get the a stable equilibrium point at $\theta = 0$ as desired, a feedback controller needs to be implemented to move the system pools to the left hand plane, and there by making it stable.

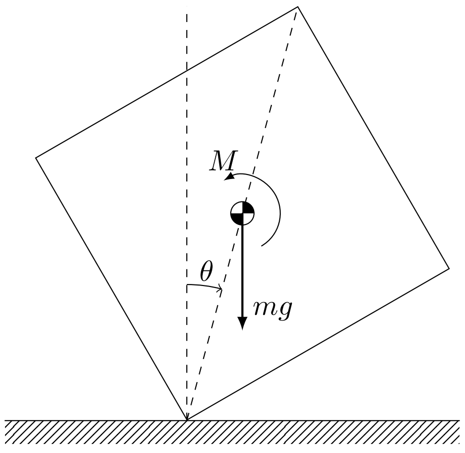


Figure 2.1. The 1-dimentional cube model.

2.2.1 Control System

In order to get a sufficiently fast and robust control system a state-space model is used. With the substitution

$$\bar{x} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \tag{2.4}$$

our open system described by (2.1) can be written as

$$\begin{aligned} \dot{\bar{x}} &= \mathbf{A}\bar{x} + \mathbf{B}M & (2.5) \\ \theta &= \mathbf{C}\bar{x}, & (2.6) \end{aligned}$$

where **A,B,C** follows

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ \frac{mgr}{I_{tot}} & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ -\frac{1}{I_{tot}} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{2.7}$$

If a controller, L, following

$$L = \begin{bmatrix} l_1 & l_2 \end{bmatrix} \tag{2.8}$$

is used in a feedback loop, as in Figure 2.2, we get the closed loop system:

$$\dot{\bar{x}} = (\mathbf{A} - \mathbf{BL})\bar{x} + \mathbf{B}l_0 r. \tag{2.9}$$

As equation (2.9) shows the closed system in Figure 2.2 can be stable if **L** is chosen so that the eigenvalues of $(\mathbf{A} - \mathbf{BL})$ all are negative. The system pools, p_1 and p_2, will be chosen to give a fast and stable dynamic. With consideration to the given

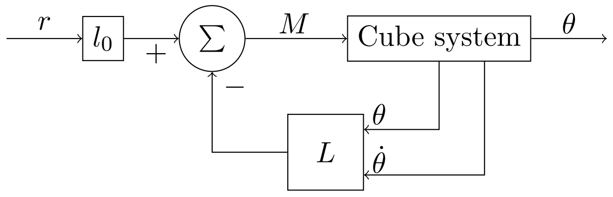


Figure 2.2. Feedback system for our control model.

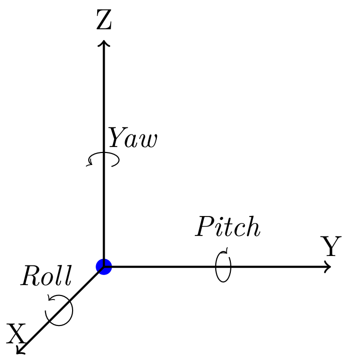


Figure 2.3. Cartesian system with yaw, pitch and roll

pair of pools l_1 and l_2 needs to be chosen to follow:

$$\begin{aligned}
\begin{vmatrix} \lambda & -1 \\ -\frac{mgr-l_1}{I_{tot}} & \lambda + \frac{l_2}{I_{tot}} \end{vmatrix} &= (\lambda - p_1)(\lambda - p_2) & (2.10)\\
&\Leftrightarrow \\
\lambda^2 + \frac{l_2}{I_{tot}}\lambda - \frac{mgr - l_1}{I_{tot}} &= \lambda^2 - (p_1 + p_2)\lambda + p_1 p_2 & (2.11)
\end{aligned}$$

From (2.11) the values for l_1 and l_2 can be derived to follow:

$$\begin{aligned}
l_1 &= -(p_1 p_2 I_{tot} + mgr) & (2.12)\\
l_2 &= (p1 + p2) I_{tot} & (2.13)
\end{aligned}$$

2.3 Attitude Control

In attitude control theory the controlling focuses on inertial navigation with respect to an inertial frame of reference. As seen in Figure 2.3 *yaw, pitch* and *roll* describe the rotation around the axis of the body in motion.

To give a precise position by electronic measurement, it is preferable to combine different kinds of sensors and use their different advantages. The most common way is to combine an accelerometer and a gyroscope on a inertial platform where the drift that occurs in an gyro over time is reduced by the accelerometer, while as the accelerometer is very sensitive to vibrations and mechanical noise which as the gyroscope is not, [King, 1998]. A measurement unit that combines these sensors is called an IMU, *Inertial measurement unit.*

The output data from sensors are voltage, combining this with resolution, range and other transformation parameters a real value output can be calculated.

2.3.1 Accelerometer

An accelerometer measures exactly what its name implies; acceleration $[m/s^2]$. There are mainly two different technologies used in accelerometers; the first measures the dynamic changes in a piezoelectric material which occurs when the material is deformed in any way. This will lead to a change in the conductivity of the material which is picked up by sensors and transformed in to an voltage output.

The second one consists of a structure of small pieces that can move freely relatively each other. Sensors measures changes in capacitance between the pieces and then returns an voltage output.

2.3.2 Gyroscope

A gyroscope measures movement around predefined axes in a Cartesian system, the value measured is thus angular rate per second $[rad/s$ or $°/s]$. The method to measure this varies, larger gyroscopes uses the law of conservation of angular momentum while the most common way in micro-electronics is to use piezoelectric materials just as with the accelerometer.

2.4 Communication Protocol

When communicating between devices a electronic architecture (both hardware and software) that transfers data is called a *bus.* The communication protocol is part of the bus and is either built serial or parallel. The communication used in this project is a serial communication, which is the process of sending one data bit at a time.

In this project the only communication protocol directly implemented and programmed is the I^2C-protocol, [Semiconductors, 2014]. The protocol uses two bus lines; one serial data line called SDA and a serial clock line called SCL. Both lines are bi-directional meaning data is sent both ways.

2.5 Kalman filter

2.5.1 Background

When sampling data of a dynamic system with an accelerometer and a gyroscope the sensors are very sensitive to disturbances. To get a stable signal from the sensors a linear quadratic estimation filter called Kalman filter is used. All theory regarding the Kalman filter in this thesis is based on the work published by: [MAYBECK, 1979] and [G. Welch, 2001].

The filter is recursive such as that it measures the multiple signals at time t_{n-1} and create an estimate of the next samples at t_n. The estimates are then compared to the real-time measurements at time t_n. This process is repeated for every sampling interval and uses a Bayesian (statistical) model to predict the prior state based on the previous state. It is important to remember that even though it is called the Kalman filter, it is not a filter at all, it is an estimator.

2.5.2 Measurement and State Model

In control theory one of the main problems faced when using sensors is the problem with designing an observer, where the mission is to describe an internal system with knowledge only of its inputs and output. The approach to solving this problem is a process model similar to the one described in equation 2.4 describing a state-space model representing a system of linear differential equations:

$$\bar{x}_t = \mathbf{A}\bar{x}_{t-1} + \mathbf{B}u_k + w_{t-1}, \tag{2.14}$$

$$\bar{y}_t = \mathbf{C}\bar{x}_t + v_k. \tag{2.15}$$

(2.14) is the state prediction model whereas (2.15) is the measurement model. The terms w_{t-1} and v_k represent the process and measurement error. To make the model work practically the assumption that they both are white, Gaussian distributed noises with different covariances, Q and R respectively is drawn.

2.5.3 The Priori and Posteriori States

The Kalman filter uses two estimates of states in a looping cycle. The first is the *priori* state estimate, $\hat{x}_k^-$ and second one is the *posteriori* state estimate $\hat{x}_k$.

The *estimated error* at these two states can be estimated by deriving the state estimates from the actual value as; $x_t - \hat{x}_t^-$ and $x_t - \hat{x}_t$. Since the estimate errors are defined the *priori* and *posteriori* estimate error covariances, P_t^- and P_t respectively can also be defined.

Once the two state estimates are defined it is time to introduce the *Kalman gain* in the expression that computes the posterior state estimate:

$$\hat{x}_t = \hat{x}_t^- + K(y_t - C\hat{x}_t^-). \tag{2.16}$$

K is the Kalman gain and acts as a weighting factor, $C\hat{x}_t^-$ is the *posteriori measurement prediction.* If the measurement prediction is good the value $(y_t - C\hat{x}_t^-)$ will be minimized and the state estimate will remain the same. The Kalman gain, K is updated for every cycle and is defined as:

$$K_t = P_t^- C^T [C P_t^- C^T + R]^{-1}, \tag{2.17}$$

$$\begin{tikzcd}
\text{Time Update} \arrow[r, bend left] & \text{Measurement Update} \arrow[l, bend left]
\end{tikzcd}$$

Figure 2.4. The Kalman filter cycle.

where it is noticeable that if the measurement covariance, R is small the Kalman gain goes towards C^{-1} and will thus remain the same, showing the same behaviour as the posteriori state estimate $\hat{x}_t$ (2.16).

2.5.4 The Kalman Filter Cycle

The filter cycle is illustrated in 2.4 where the time update phase could be described as a prediction phase while the measurement update would be described as a correction phase.

The time update equations are based on the previous estimations and creates a transition from $t-1$ to t by updating the priori values:

$$\hat{x}_t^- = \mathbf{A}\hat{x}_{t-1}^- + \mathbf{B}u_t, \tag{2.18}$$

$$P_t^- = \mathbf{A}P_{t-1}^-\mathbf{A}^T + Q \tag{2.19}$$

The cycle continues to the measurement phase and updates the the Kalman gain (2.17), then the posteriori states estimate (2.16) and finally the estimate error covariance:

$$P_t = (I - K_tC)P_t^-, \tag{2.20}$$

where I is an identity matrix. For more profound description and derivation of the equation see [MAYBECK, 1979] and [G. Welch, 2001].

Chapter 3

Construction and Development

In this chapter the construction and development is described and the technical solutions are presented.

3.1 Problem Formulation

To construct the demonstration a few things needed to be done:

- Developing the controller
- Sizing the electronics to deliver the needed torque to the reaction wheel,
- Building the cube structure to hold all components with as low weight as possible.

3.2 Software

3.2.1 The controller

This section will describe how the controller is programmed and built.

3.2.2 Sensors

This section will describe how the sensors is implemented.

3.2.3 Motors

This section will describe how the motors is controlled from the controller.

3.3 Electronics

This section will describe how the electronic system is built.

3.4 Hardware

This section will describe how the QS is designed and built.

3.5 Results

Here we will present the result of our hard labour.

Chapter 4

Discussion and conclusions

In this chapter the result described in 3.5 will be discussed and conclusion of the project will be drawn.

4.1 Discussion

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.2 Conclusions

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 5

Recommendations and future work

In this chapter recommendations for further work can be found.

5.1 Recommendations

NYI

5.2 Future work

NYI

Bibliography

[Ard,] Arduino, frequently asked questions. http://arduino.cc/en/Main/FAQ. Accessed: 2015-03-08.

[EAGLE, 2014] EAGLE (2014). *version 7.2.0.* CadSoft Computer, Pembroke Pines, Florida.

[G. Welch, 2001] G. Welch, G. B. (2001). An introduction to the kalman filter. http://www.cs.unc.edu/~tracker/media/pdf/SIGGRAPH2001_CoursePack_08.pdf.

[King, 1998] King, A. D. (1998). Inertial navigation - forty years of evolution. Technical report.

[MATLAB, 2014] MATLAB (2014). *version 8.4.0 (R2014b).* The MathWorks Inc., Natick, Massachusetts.

[MAYBECK, 1979] MAYBECK, P. S. (1979). Stochastic models, estimation and control. Technical report.

[Rowberg, 2015] Rowberg, J. (2015). i2cdevlib. https://github.com/jrowberg/i2cdevlib.

[Semiconductors, 2014] Semiconductors, N. (2014). I2c-bus specification and user manual. http://www.nxp.com/documents/user_manual/UM10204.pdf.

[Solid Edge, 2014] Solid Edge (2014). *ST7.* Siemens PLM Software, Plano, Texas.