Mechanics End semester project

Two sided self-balancing robot

M.Ammar	387796
Abdul-Raheem	369247
Waleed	365843
Umer Ali	378563
Abdullah Ayaz	378499

Background ……..3

Purpose ……..3

Method ……..4

Model definition ……..5

Control Theory ……..6

LQR, Linear Quadratic Regulator ……..8

Arduino UNO ……8

Angular data, ψ and ψ˙ ……9

Movement - Motors and Driver shield ……10

Construction of physical demonstrator ……..11

Validation process ……..12

Software ……13

Results ……13

Validation of the model ……14

References……16

Background

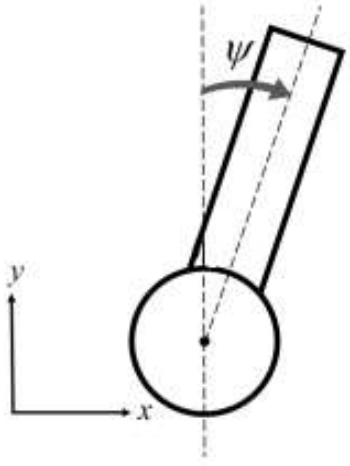
Inverted pendulum applications are plenty; for example the human body is an inverted pendulum balancing the upper body around our ankle joints in every step. In the recent decade Segway's, making use of bodily movements for steering, have emerged on the market. In common, these applications share that their centre of mass is located above their pivot points, thus requiring active control in order to balance. The open-source community is full of instructions and code snippets, many making use of the open source micro controller Arduino for control algorithms. An example is the open-source kit Balanduino as seen below.

Purpose

Prepare the demonstrator for a state-space controller to be implemented, that can control the angle deviation, ψ and position, x. As seen in the diagram below. A state-space description is based on a model of reality. Several assumptions and simplifications must be made. The core part of this project will be to create a model for that purpose and validate whether the model is good enough for implementation of a state-space control, this will be done by answering the question:

Using a manually tuned PID-controller, what conclusions can be made regarding the reliability of the model?

From answering this question it can be concluded if the model is accurate enough for future state space control.



Method

To fulfil the purpose the following method will be used:

• Derive dynamical equations based on theory of the inverted pendulum

• Form transfer functions for the angle deviation, ψ and position, x

• Find a controller that can control these two conditions

• Set up requirements for the demonstrator

• Design a demonstrator that fulfils these requirements, investigate the boundaries of the control signal Chosen error sources will be investigated:

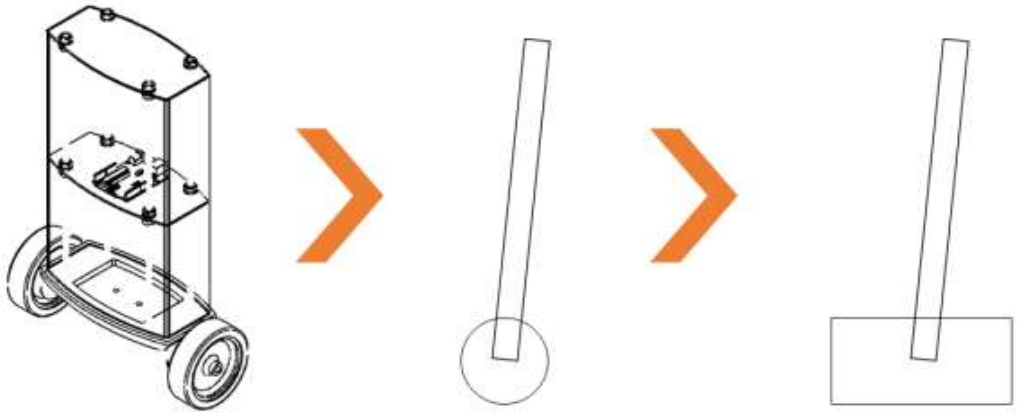
• Design a three dimensional model in CAD - Computer Aided Design that is as identical as possible to the physical demonstrator to acquire correct parameters needed for the simulated model

• Investigate the accuracy of the sensor that delivers the angular data

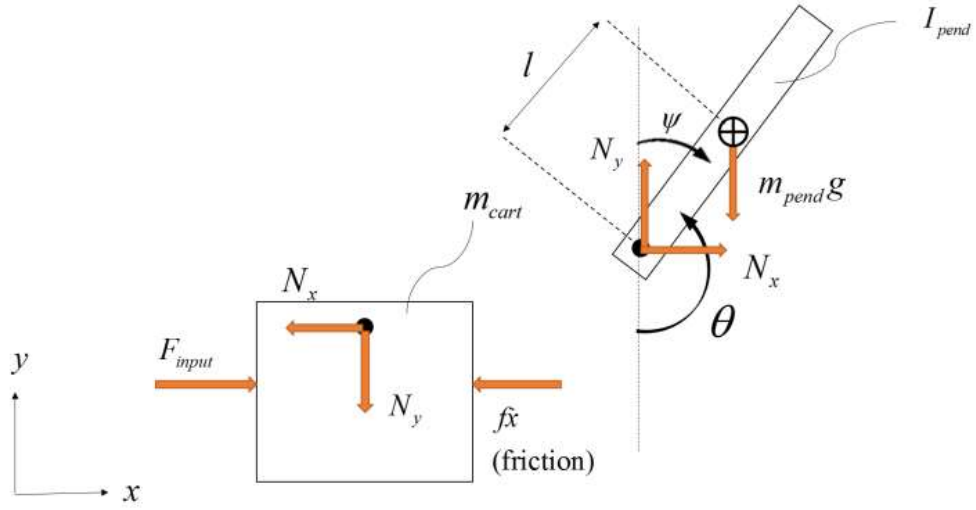
With an accurate model of the system and a functioning demonstrator this provides a platform for experiments in a simulated environment. The simulated model in comparison to the demonstrator will be validated by implementation of a PID-controller in both in order to compare impulse responses.

Model definition

The physical problem of the balancing robot is well described by the widely analysed inverted pendulum. It is commonly modelled as a rigid rod fastened by a frictionless joint to a rigid cart moving in one direction. The simplification that the wheel base can be seen as a cart sliding on a frictionless surface was made.



For convenience the pendulum's degrees of freedom are limited to one direction, the angle θ moving in the xy-plane



Control Theory

Calculation of the system's transfer function:

$$F_{input} = (m_{cart} + m_{pend})\ddot{x} + f\dot{x} + m_{pend}l\ddot{\theta}\cos\theta - m_{pend}l\dot{\theta}^2\sin\theta$$

$$(I_{pend} + m_{pend}l^2)\ddot{\theta} + m_{pend}gl\sin\theta = -m_{pend}l\ddot{x}\cos\theta$$

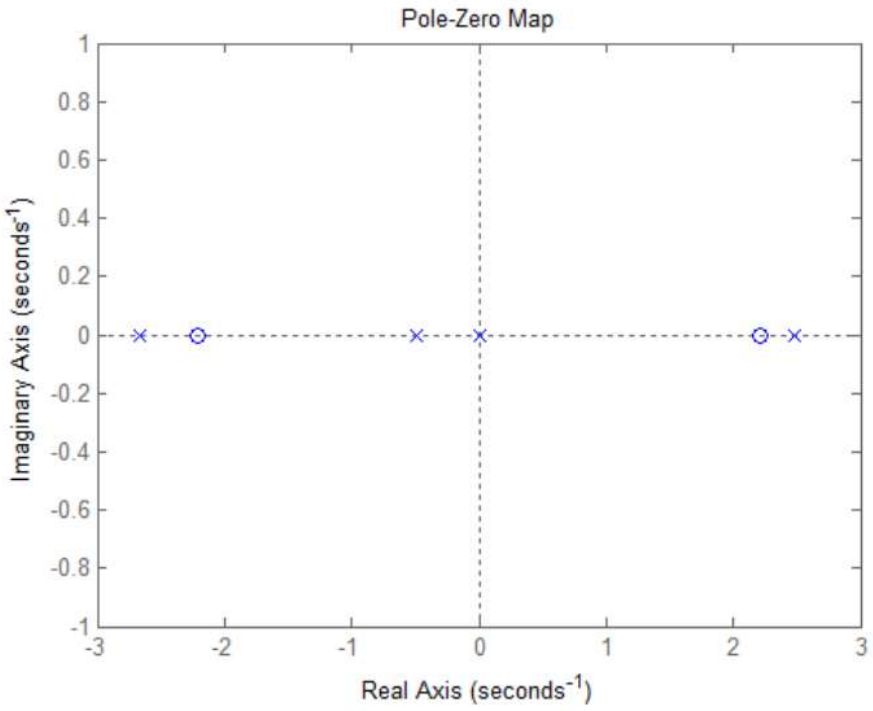
Where m_{cart} and m_{pend} represent the cart's and pendulum's weight respectively. Parameter, l, is the distance to the pendulum's center of mass, θ the angle between the pendulum and the vertical axis. The parameter, f is a friction parameter. Rewriting the equations as transfer functions renders:

$$\Psi(s) = \underbrace{\frac{\frac{m_{pend}l}{q}s}{s^3 + \frac{f(I_{pend}+m_{pend}l^2)}{q}s^2 - \frac{(m_{cart}+m_{pend})m_{pend}gl}{q}s - \frac{fm_{pend}gl}{q}}}_{G_\Psi(s)} U_{input}(s)$$

$$X(s) = \underbrace{\frac{\frac{(I_{pend}+m_{pend}l^2)s^2 - gm_{pend}l}{q}}{s^4 + \frac{f(I_{pend}+m_{pend}l^2)}{q}s^3 - \frac{(m_{cart}+m_{pend})m_{pend}gl}{q}s^2 - \frac{fm_{pend}gl}{q}s}}_{G_x(s)} U_{input}(s)$$

Where Ψ represents the angle deviation and X the position of the cart.

ed that for all positive values of the parameters l, I_{pend}, m_{cart}, m_{pend} and g the system in itself is unstable since it has a pole in the right half plane as can be seen in the pole-zero map.



This arrangement is viable, since an inverted pendulum is intuitively unstable. The differential equations are linearised. The system can then also be described in State Space form with the states being $\dot{x}$, $\ddot{x}$, $\dot{\psi}$ and $\ddot{\psi}$.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = A \begin{bmatrix} x \\ \dot{x} \\ \psi \\ \dot{\psi} \end{bmatrix} + B u_{input}$$

$$y = C u_{input}$$

Where u_{input} is the control effort in time domain. The system matrices A, B and C given by:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I_{pend}+m_{pend}l^2)f}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & \frac{m_{pend}{}^2 g l^2}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-m_{pend}lf}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & \frac{m_{pend}gl(m_{cart}+m_{pend})}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \frac{I_{pend}+m_{pend}l^2}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} \\ 0 \\ \frac{m_{pend}l}{I_{pend}(m_{cart}+m_{pend})+m_{cart}m_{pend}l^2} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

For complete calculation of the matrices A, B and C see Appendix A - Calculation of the system's transfer function. The system is controllable if the matrix S, defined as

$$S = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix}$$

has full rank. Likewise, the system is observable if the matrix O has full rank, where O is defined as:

$$O = \begin{pmatrix} C \\ CA \\ CA^2 \\ CA^3 \end{pmatrix}$$

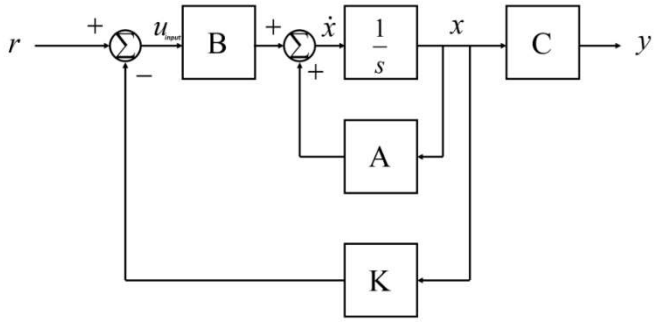
The rank of S and O confirms that the system is controllable and observable for any positive value of l, I_{pend}, m_{cart}, m_{pend} and g.

There are several ways to control this thesis' system if the position x, is disregarded. A possible control method would be a PID-controller . Since there are two states to be controlled, the deviation of the angle ψ and the position, x, cascaded PID-regulators can be used .

Another possibility is to implement some type of state space control. If the states of the system are known, the choice of in-signal can be based on the states inside the system leading to a state space feedback. The state space description holds information of the system's behaviour in past, present and future timespans.

LQR, Linear Quadratic Regulator

To control a linear system, LQR-control can be applied. The states of the dynamical system are described by a state space model including the matrices A, B and C



A cost matrix, Q, is introduced in order to weigh how fast each corresponding element in the state vector, x, need to reach their desired end values.

The control effort, u_{input} is given as: $u_{input} = -Kx$

where K is the gain matrix defined as: $K = R^{-1}B^T P$

and P is solved from the Riccatti Equation: $Q + A^T P + PA - PBB^T P = 0$

Since the state space system is observable and controllable, P has a unique solution so that the closed loop system poles are strictly in the right half plane

It is however necessary to make sure that for a specified change in input, for example an impulse on the robot, that the control effort uinput calculated by the controller is within certain bounds.

Arduino Uno board

The micro controller board used in this project is the Arduino Uno running on the ATmega328. The board is used for rapid prototyping of interactive applications through sensors monitoring the outside world. It has 14 digital in-/output pins and 6 analogue inputs. It operates on 5V and communicates via USB-port. The clock speed is 16 MHz and the flash memory is on 32 kB. Chosen components will have to be compatible with the Arduino Uno.

Angular data, ψ and $\dot{\psi}$

As seen in section Problem definition some kind of measurement unit is needed to determine the angle deviation, ψ and the angular velocity $\dot{\psi}$. A sensor that fulfils these requirements is the nine axis gyro/accelerometer, MPU-9150. The gyro can sense the angle velocity in three directions and the accelerometer senses angular acceleration. Only one axis is necessary for this application, although this IMU was available for this project and was therefore chosen.

The communication between the IMU and the processor is carried out through serial communication via protocol I2C at 400 kHz, with the accuracy of 16-bit which the Arduino can handle.

The MPU-9150 can track both slow and fast motions and it is programmable for different ranges depending on need. In this project the full scale range for the gyroscope is set to 250 °/s and the accelerometer to 2g. It runs on 3.3V (Vcc) which can be provided from the Arduino Uno board's pin with the same voltage. The MPU-9150 is connected to two of the Arduino Uno board's six analogue input pins. The gyro values will be read in quids (a number between 0 and 1023), this can be translated into angular velocity, $\dot{\psi}$.

As mentioned in the problem definition the angle ψ is also needed. It is calculated with:

$$\psi_{i+1} = \psi_i + \dot{\psi}_{i+1} dt$$

where dt is the time difference since the last loop.

State estimator - Kalman filter

As seen in the Equation above the gyro angle tends to drift after time. A state estimator used in this project is the Kalman Filter. As mentioned in the theory section of this thesis, controlling the system depends on the angle ψ to be known. With better accuracy on this value the probability of attaining a stable system increases. To increase the precision of the deviation angle, ψ, a Kalman filter can be implemented. Essentially, the Kalman filter is an algorithm that combines data and sorts out unreliable data by continuously estimating a prediction of the future, thus sorting out the data not consistent with this prediction. This project is accomplished with an open-source Kalman Filter code.

Minimising the error - Testing the accuracy of the angular data

The precision of the angular data is essential. Therefore it is necessary to experiment with the IMU together with the estimator, Kalman filter, to ensure that the output data is accurate.

The results from the experiments showed that the angular data from the sensor and the estimator follows sudden changes of tilt well, and the delay is small enough to be disregarded within the frames of this project. The magnitude of the noise was from statical tests concluded to be less than or equal to 0.03 degrees and is also disregarded.

Movement - Motors and Driver shield

Two DC-motors of type EMG30 were assigned for this project. The motors are equipped with encoders and a 30:1 reduction gearbox. The Table shows each motor's specifications from the datasheet

Rated voltage	12 V
Rated torque	0.147 Nm
Rated speed	170 rpm or 17.8 rad/s
Rated current	530 mA
No load speed	216 rpm or 22.62 rad/s
No load current	150 mA
Stall current	2.5 A
Rated output	4.22 W

According to the problem definition each motor should deliver at least 300 rpm, although the EMG30 delivers a rated speed of 170 rpm at the rated voltage 12V. Therefore these motors do not fulfil the recommendation regarding speed but they are used because they were the only available motors for this project with built-in encoders.

The motor driver used in this project is the Velleman VMA03. It is based on the L298P dual full bridge driver. It is design to fit the Arduino Uno as a shield. It runs on either external power supply or power from the Arduino board. The max current is 2.5 A for each channel.

The built-in encoders on the motors are Hall sensors, and they will use two digital pins on the Arduino. They have a resolution of 360° per rotation

Control force limits

Calculation of the system's transfer function is in fact an applied force that comes from the two DC motors. It can be described as:

$$u_{input} = \frac{2M_{motor}}{r}$$

Where M_{motor} is the mechanical torque from one motor and r is the radius of the wheels. The torque can be expressed as:

$$M_{motor} = K_2 \Phi I_A$$

Where $K_2\Phi$ is a torque constant of the DC-motor, and I_A is the armature current.

Applying Kirchhoff's law on the circuit of one DC-motor: $U_A = R_A I_A + K_2 \Phi \omega$

Where U_A is the armature voltage applied to each motor, R_A is the sum of all the resistance in the motor and ω is the angular velocity of the shaft. R_A was measured to be 8.5 Ω

Parameter $K_2\Phi$ was theoretically calculated using the measured resistance and no load speed and no load current from the data sheet as well as an armature voltage of 12V. $K_2\Phi$ was calculated to be 0.474 Nm/A.

$$M_{motor} = \frac{K_2 \Phi}{R_A} (U_A - K_2 \Phi \omega)$$

$$u_{input} = \frac{2K_2 \Phi}{R_A r} (U_A - K_2 \Phi \omega)$$

It is assumed that ω at the moment of a_m impulse on the system, which is the moment that requires the largest control force, is 0 rad/s. Equation above with ω=0 shows the proportional relation between the control force u_{input} and the armature voltage of the DC-motor, UA.

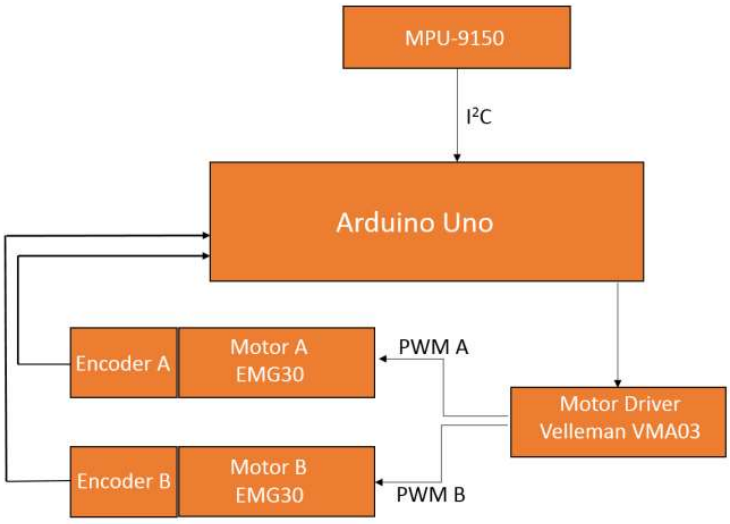
$$u_{input} = \frac{2K_2\Phi}{R_A r}U_A$$

The proportional relation between uinput and UA is a factor of 1.9. This means that the limit of the control effort for the motors running on 12V is a force of 22N.

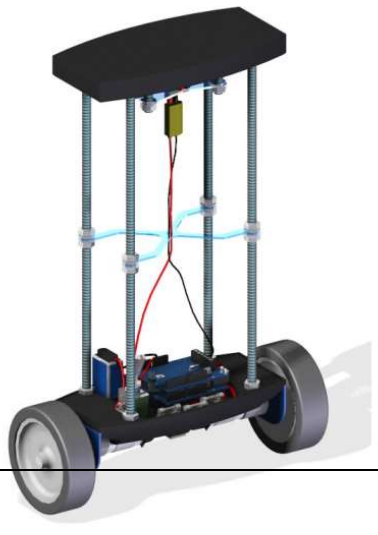
Construction of physical demonstrator

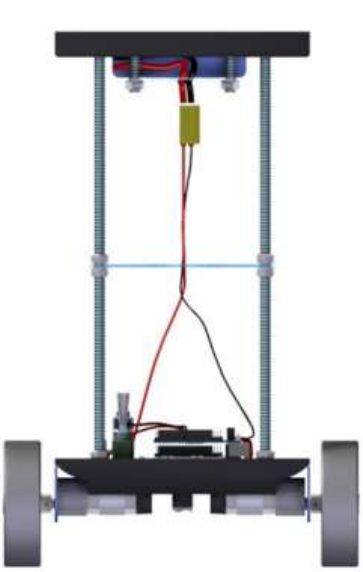
The demonstrator was designed both physically and in CAD. The used CAD program is Solid Edge[14]. The theoretical model is dependent on the exact values of the m_{cart}, m_{pend} and I_{pend}. Solid Edge assists in finding these parameters easy through the application Physical properties.

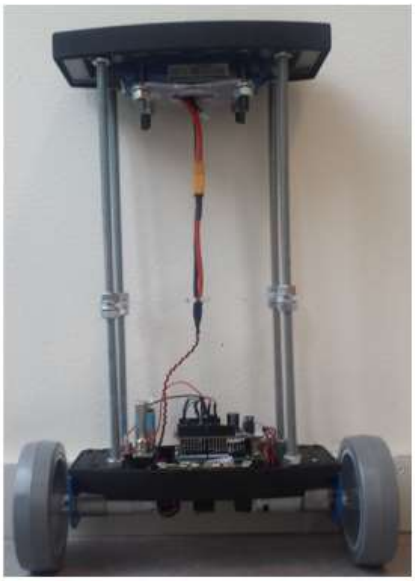
The battery is the component with the greatest value of density. The placement of the battery at the top is to make sure that the center of mass is located above the pivot point. All the other components are placed close to each other to avoid interference that comes with longer wires.



To secure a stable foundation for the robot a bottom plate was designed. Its purpose is to fasten the motor brackets and also to align the gyroscope axis with the motor axis.







Validation process

The derived model that forms the basis for the LQR-controller contains several assumptions and simplifications. It has been justified why these have been made, but it is not possible to determine if they will affect the stability of the system before implementing the controller to the demonstrator.

If the controller is implemented directly and it turns out that there are errors in the model and the demonstrator fails to balance, it is difficult to determine if the model is close to work or not.

Therefore, a validation process with a PID controller will be made. This is because the demonstrator can be manually tuned to achieve balance, though it can only control the angle deviation as mentioned in the Theory chapter of this thesis. It is then possible to compare impulses from the demonstrator to simulated impulses with the same PID-controller. If they behave similarly, the transfer function for the angle deviation, $G_{\psi}(s)$ is assumed to be valid. Since the transfer function for the position, $G_x(s)$ is based on the same parameters and assumptions, it can also be assumed valid.

If it proves to be a noticeable difference between the real and the simulated impulses, it is assumed that it can be an indication of what the problem in the model is.

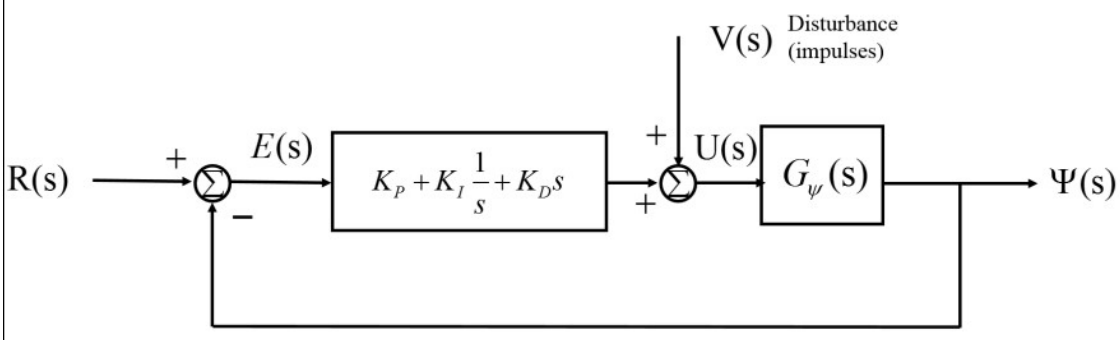
The validation process will be performed in these steps

1. Implement a manually tuned PID-controller on the demonstrator

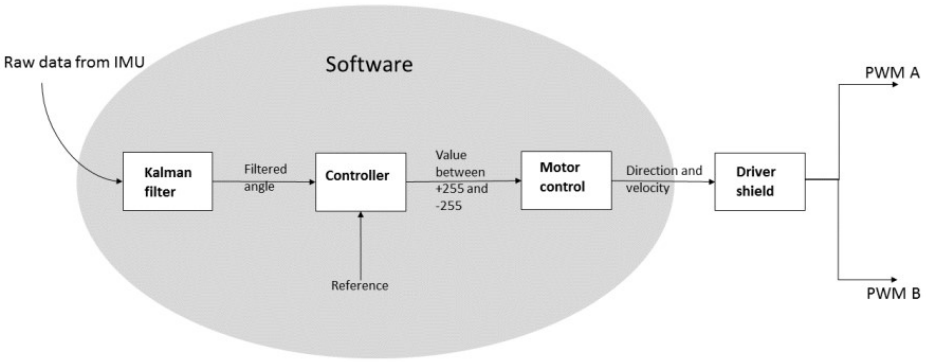
2. Perform impulse experiments and store the serial data from the IMU

3. Insert the same PID-parameters to a simulated PID-controller with the transfer function for the deviation angle $G_{\psi}(s)$.

4. Compare the results and discuss the reliability of the theoretical model.



Software



Software for the demonstrator, assembled and created in the Arduino environment is presented schematically in Figure above. An open-source library for Kalman filter and a PWM-library was used in order to transport the raw data from the IMU all the way to how the motors behave, in terms of direction and PWM.

Results

Parameters	Values
m_{cart}	0.558 [kg]
m_{pend}	1.267 [kg]
I_{pend}	0.055 [kgm^2]
l	0.159 [m]

With these parameters inserted to the MATLAB model, the State Space system looks as:

$$\begin{pmatrix} \dot{x} \\ \ddot{x} \\ \dot{\psi} \\ \ddot{\psi} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.007 & 3.363 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.017 & 30.47 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \psi \\ \dot{\psi} \end{pmatrix} + \begin{pmatrix} 0 \\ 0.736 \\ 0 \\ 1,704 \end{pmatrix} u_{input}$$

$$y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ \psi \\ \dot{\psi} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} u_{input}$$

The unstable system can be stabilized by an LQR-controller. Using a cost matrix Q of:

$$Q = \begin{pmatrix} \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

where α and β are weight factors for the position and angle respectively. With their relative sizes the importance of the error and control effort for the respective states can be calibrated. [12] The control gain matrix, K , can be determined using for example the MATLAB function lqr():

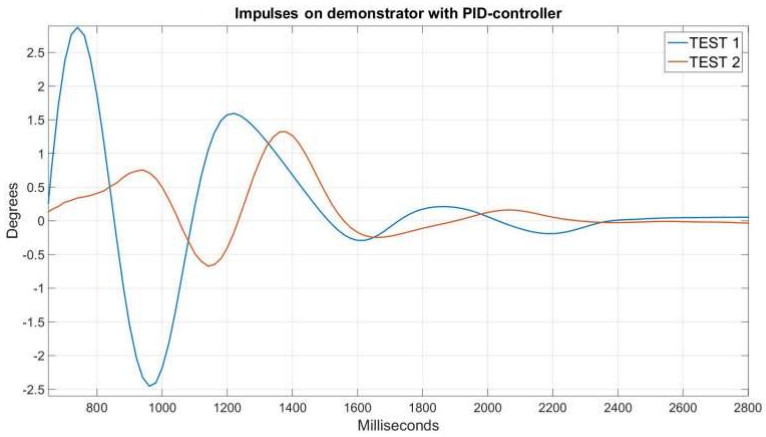
$$K = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 \end{pmatrix}$$

The proportional factor between the voltage given to the motors, U_A, and the control effort, u_{input}, was calculated to be 1.9. This results in a maximum control effort of ±22N that can be delivered from the motors.

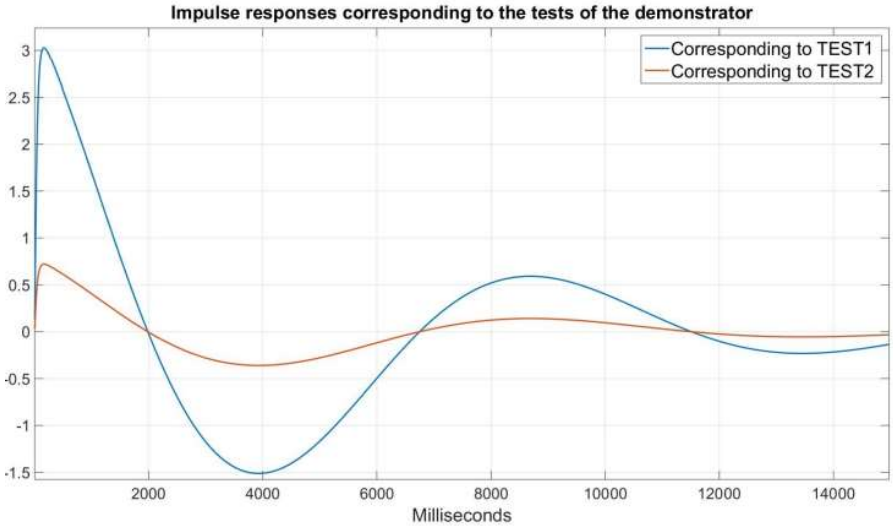
With these preparations an LQR-controller can be implemented for the model if it is ascertained that the control effort stays within the limits of ±22N.

Validation of the model

Several different impulses were introduced to the demonstrator, resulting in impulse responses plotted from serial angular data. Two representative tests are shown below. Both impulse responses show similar settling times of approximately 2.2 seconds.



Insertion of the manually tuned PID-parameters K_p=24, K_i=8 and K_d=15 into the simulated model resulted in the impulse response presented in Figure below. It is clear that a settling time of 12 seconds in the simulated system is not well enough and therefore the simulated model is currently deemed not to be an accurate representation of the demonstrator. Possible reasons for this will be discussed in the Discussion chapter of this thesis.


Impulse responses corresponding to the tests of the demonstrator
Corresponding to TEST1
Corresponding to TEST2
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
2000
4000
6000
8000
10000
12000
14000
Milliseconds

References

- AMCI Tech Tutorials. 2015. Stepper vs. Servo. Available at: http://www.amci.com/tutorials/tutorials-stepper-vs-servo.asp. [Accessed 13 May 2015].

- Arduino Uno. 2015. Arduino - ArduinoBoardUno . Available at: http://www.arduino.cc/en/main/arduinoBoardUno. [Accessed 22 April 2015]

- EMG30. 2010. Data sheet. Available at: http://www.robot−electronics.co.uk/htm/emg30.htm [Accessed 22 April 2015]

- Faragher, Ramsey. 2012. Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation. IEEE signal processing magazine.

- Glad, Torkel. Ljung, Lennart. 2006. Reglerteknik - Grundläggande teori. 4th edn. Studentlitteratur.

- InvenSense Inc. 2013. MPU-9150 Product Specification. Sunnyvale, USA. Revision 4.3. Available at: http://www.invensense.com/mems/gyro/documents/PSMPU-9150A-00v43.pdf. [Accessed 22 April 2015]

- Johansson, Hans. Lindahl, Per-Erik. Meyer, Robert. Grimheden, Martin. Sandqvist, William. Paulsson, Magareta. 2013. Elektroteknik. KTH

- Lauszus, Kristian. 2012. Kalman Filter. Available at: https://github.com/TKJElectronics/KalmanFilter [Accessed 20 Mars 2015]

- Loram, I, 2002. Human balancing of an inverted pendulum: position control by small, ballistic-like, throw and catch movements. Journal of Physiology, 540.3, 1111.

- Lövgren, Samir. 2015. Self balancing Robot. Available at: http://samirlovgren.se/pageid647. [Accessed 11 May 2015]. 29 CHAPTER 7. REFERENCES

- MathWorks Nordic. 2015. MATLAB - The Language of Technical Computing. Available at: http://se.mathworks.com/products/matlab/. [Accessed 21 April 2015]

- MathWorks. 2012. Control Tutorials for MATLAB and SIMULINK, Inverted Pendulum. Available at: http://ctms.engin.umich.edu/CTMS/index.php.exampleInverted Pendulumsection-SystemModeling. [Accessed 21 April 2015]

- Redhat. 2015. What is open source software. Available at: https://opensource.com/resources/what-open-source. [Accessed 13 May 2015].

- Siemens PLM Software. 2015. Solid Edge. Available at: http://www.plm.automation.siemens.com /en-us/products/solid-edge/. [Accessed 21 April 2015]

- Velleman. 2013. Motor and power shield for Arduino. Gavere, Belgium. Available at: http://www.velleman.co.uk/manuals/vma03.pdf [Acessed 29 Mars 2015]

- Jin, D. 2015. Development of a Stable Control System for a Segway. Available at: http://www.raysforexcellence.se/wp-content/uploads/2013/01/Dennis-Jin-Developmentof-a-stable-control-system-for-a-segway.pdf. [Accessed 02 May 2015]