ISSN 0280–5316
ISRN LUTFD2/TFRT--5659--SE

Bluetooth in Control

Andreas Hörjel

Department of Automatic Control
Lund Institute of Technology
January 2001

Department of Automatic Control **Lund Institute of Technology** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER THESIS *Date of issue* January 2001 *Document Number* ISRN LUTFD2/TFRT--5659--SE
Author(s) Andreas Hörjel	*Supervisor* Johan Eker, Anton Cervin, Mats Andersson (ConnectBlue) *Sponsoring organisation*

Title and subtitle

Bluetooth in Control

Abstract

This master thesis evaluates control algorithms for control applications using Bluetooth in the loop. It is done in order to investigate whether Bluetooth can be used in hard real-time control loops or not. The control algorithms suggested in this work are evaluated on a real process, the Furuta pendulum, due to the unstable nature of the process. This will assure that the Bluetooth communication is pushed to its limits, where problems can be found. Problems encountered are bandwidth limitations, retransmissions in the low layers of Bluetooth and disturbances to the wireless communication.

Key words

Classification system and/ or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280–5316		*ISBN*
Language English	*Number of pages* 37	*Recipient's notes*
Security classification		

The report may be ordered from the Department of Automatic Control or borrowed through:
University Library 2, Box 3, SE-221 00 Lund, Sweden
Fax +46 46 222 44 22 E-mail ub2@ub2.lu.se

1. INTRODUCTION **4**
1.1 BACKGROUND 4
1.2 PROBLEM FORMULATION 5
1.3 ROADMAP 6
2. THE CONTROL APPLICATION **6**
2.1 HARDWARE 7
2.2 SOFTWARE 8
2.3 BLUETOOTH COMMUNICATION 10
2.4 THE CONTROLLER 13
2.5 THE REMOTE I/O 14
3. THE FURUTA PENDULUM **14**
3.1 THE FURUTA PENDULUM MODEL 15
3.1.1 Stabilization of the pendulum *16*
3.1.2 Swing Up *17*
3.1.3 Friction compensation *18*
4. BLUETOOTH CONTROL ALGORITHMS **18**
4.1. DELAY COMPENSATION 19
4.1.1 Static delay *19*
4.1.2 Stochastic delay *20*
4.2 BIT ERROR FILTER 21
4.2.1 Filter design *22*
4.2.2 Simulated Bit error filter *23*
4.3 CONTROL LAWS 24
5. BLUETOOTH CONTROL FUNCTIONS **25**
5.1 CLOCK SYNCHRONIZATION 25
5.1.1 Clock synchronization scheme *26*
5.2 TIME STAMPS 27
6. EXPERIMENTS **27**
7. GUI **29**
8. RESULTS **30**
8.1 Results with static delay compensation *30*
8.2 Results with stochastic delay Compensation *31*
8.3 Results simulated bit error filter *33*
8.4 Results with implemented bit error filter *35*
9. FUTURE WORK **35**
9.1 TIME-OUTS 35
10. CONCLUSIONS **36**

11. REFERENCES 37

1. Introduction

Bluetooth is a short distance wireless communication network technology. It was recently introduced to the market and is currently being evaluated for many future applications. One application is Bluetooth in control loops, where Bluetooth is used as a communication network in a distributed real-time control system.
The work in this report aims at evaluating control algorithms for a distributed real-time control system using Bluetooth technology.
Bluetooth is a future technology due to low cost and the wireless properties. The technology is well founded among many leading global companies, which makes Bluetooth a coming technology.

1.1 Background

Bluetooth is wireless communication that transmits over radio. The range of Bluetooth is 10 meters within which the receiver must be in order to receive information. The technique uses a frequency-hopping algorithm in order to avoid disturbance at a certain frequency. These properties allow a study of Bluetooth in control applications in order to reduce wiring.

Bluetooth is mostly known on the telecom market, e.g. for cellular phones, headsets, etc. However, the Bluetooth application possibilities are many; industry applications containing Bluetooth communication is a new area being investigated. Bluetooth could be used for monitoring, transferring, and logging of data or operator communication. These applications have little real-time requirements, which makes them easy to implement using Bluetooth. However, this work aims at making a study of Bluetooth in a hard real-time distributed control system. A hard real-time distributed control system is a system with distributed actuator sensor and controller nodes that communicate over a network. Bluetooth is not designed for hard real-time applications; it is designed for common voice and data transmission, and this is a barrier to overcome.

The Bluetooth control system studied in this work is shown in Figure 1.1.

Bluetooth in control loops creates some design problems for developers of control systems:

- Static delay in the control loop due to limited communication bandwidth. This is a predictable delay between the sampling of the process and the putting out of the control signal to the process.
- Stochastic delays due to retransmissions of Bluetooth data packets. This is an unpredictable delay.
- Information scrambling due to external disturbances.

These types of delays and disturbances can be fatal to an unstable process.

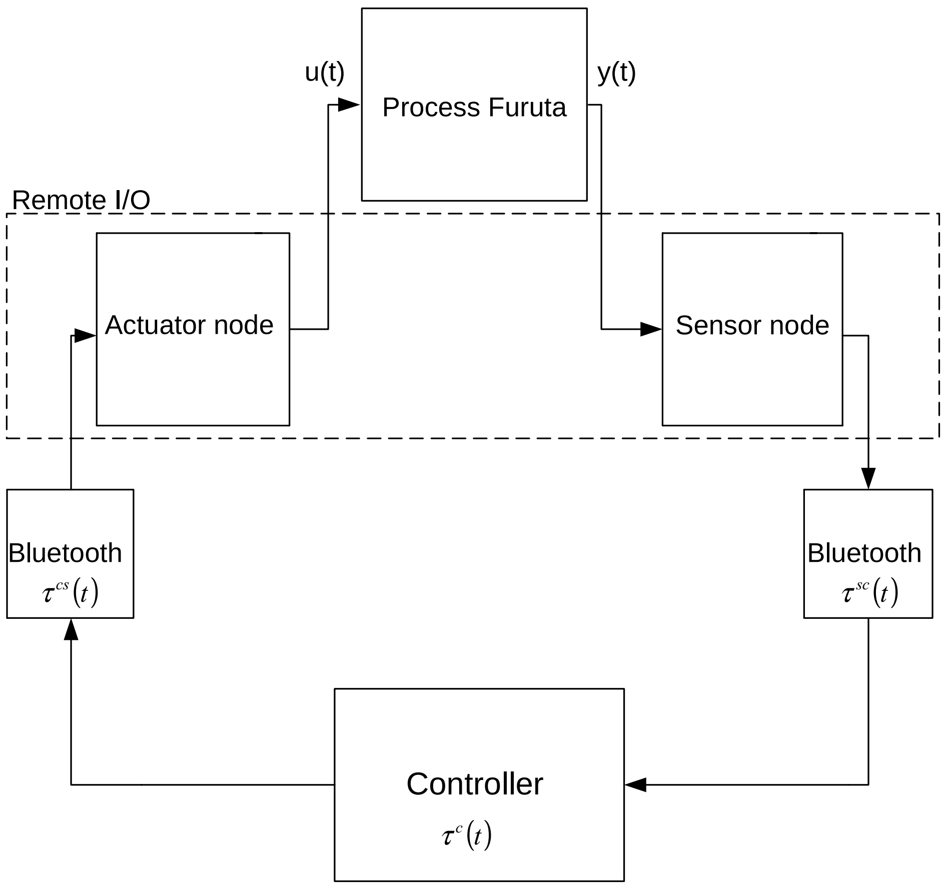


Figure 1.1 The general setup of the control application.

1.2 Problem formulation

The first subobjective is to control the unstable process, even though the control system is exposed to delays. The total delay is the total time it takes to complete one control loop, from the sampling of the system in the sensor node, through the control node, and back to the actuator node.
The round trip delay can thus be described as (see Figure 1.1):

$$\tau = \tau_{sc} + \tau_{ca} + \tau_{c} + \tau(t)$$

The delay $\tau(t)$ is the stochastic delay due to retransmissions in the loop. The other parts of the total delay are static. The static delays depend on the bandwidth limitations in the communication, which in this control loop are represented by RS232 and Bluetooth. RS23 is for communicating between the PC and the Bluetooth chips (see Figure 3.1).

The second subobjective is to reduce the effects of disturbances, with a filter, in order to assure stable control. External disturbances insert faulty information packets in the control loop, and the filter should attempt to correct the faulty packages.

A third subobjective is to control an unstable process to investigate unknown problems between the relation between Bluetooth and distributed control systems. An unstable process is used to push Bluetooth to the extreme to force unknown problems to reveal themselves.

1.3 Roadmap

The rest of the report is outlined as follows. Section 2 gives an introduction to the dynamics of the process and discusses stabilization, swing up and friction.

The main part of this work is presented in Sections 3 and 4. Section 3 defines the setup and problems with the setup when implementing Bluetooth in control loops. Section 4 defines control algorithms suitable for Bluetooth in control loops.

Section 5 explains the experimental fixes necessary to produce results to the control algorithms and experimental results and summarizes the results and conclusions of the tests. The demo interface is presented in Section 6.

2. The control application

The control application's general software/hardware setup is displayed in Figure 3.1. It shows four objects, two hardware and two software. The heart of the control application is the two software instances, Remote I/O and Control, which run on a PC. The Remote I/O software is separated from the Control software that runs on other hardware. The communication between these instances is sent over Bluetooth. This setup eliminates the need for cables between the process and the controller.

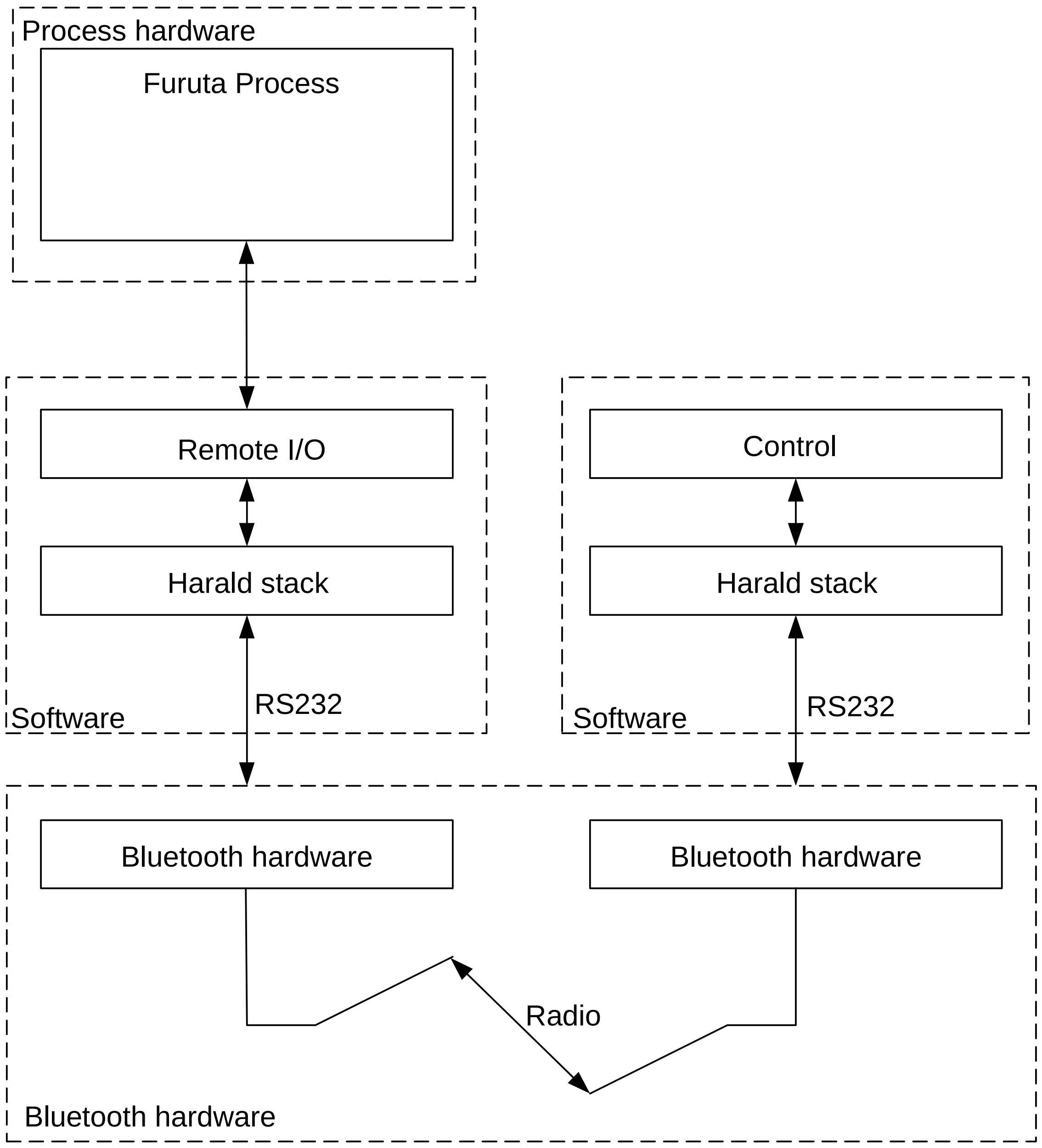


Figure 2.1 The general hardware software setup.

2.1 Hardware

The Furuta pendulum allows sampling of all four states necessary for controlling the process. This fact simplifies the control laws due to that all states are known and do not need to be estimated. The states are represented by voltages in the interval –10 to +10 V.

Two Bluetooth Application and Training Toolkits from Ericsson (version P3D) constitute the Bluetooth hardware. The Remote I/O and the Control tasks run on a computer that communicates with the toolkits through RS232 communication. The computer runs the Linux real-time operating system KURT [9].

2.2 Software

The Remote I/O and the Control task are implemented as Java programs. They use a Bluetooth stack, developed by Johan Eker [1]. The Bluetooth stack gives the tasks access to methods for establishing links, reading and writing data and etc. The Control and the Remote I/O tasks are designed specially for this work and contain all control laws and testing tools necessary.
The whole structure of the program is displayed in Figure 2.2.
A Bluetooth instance starts the control application by creating either a Remote I/O or a Control instance. The Remote I/O and the Control instances inherit from the BTUnit class. The Remote I/O, Control and Local objects are the only objects that are involved in the control loop, where the Remote I/O samples the process and Control calculates the new control signal. The Local class is a class that gives the Remote I/O and Control instances access to the Harald stack. The Container and Opcom classes are objects for creating a GUI. The Container object is a transport object that sends sampled states and the control signal to the Opcom class.

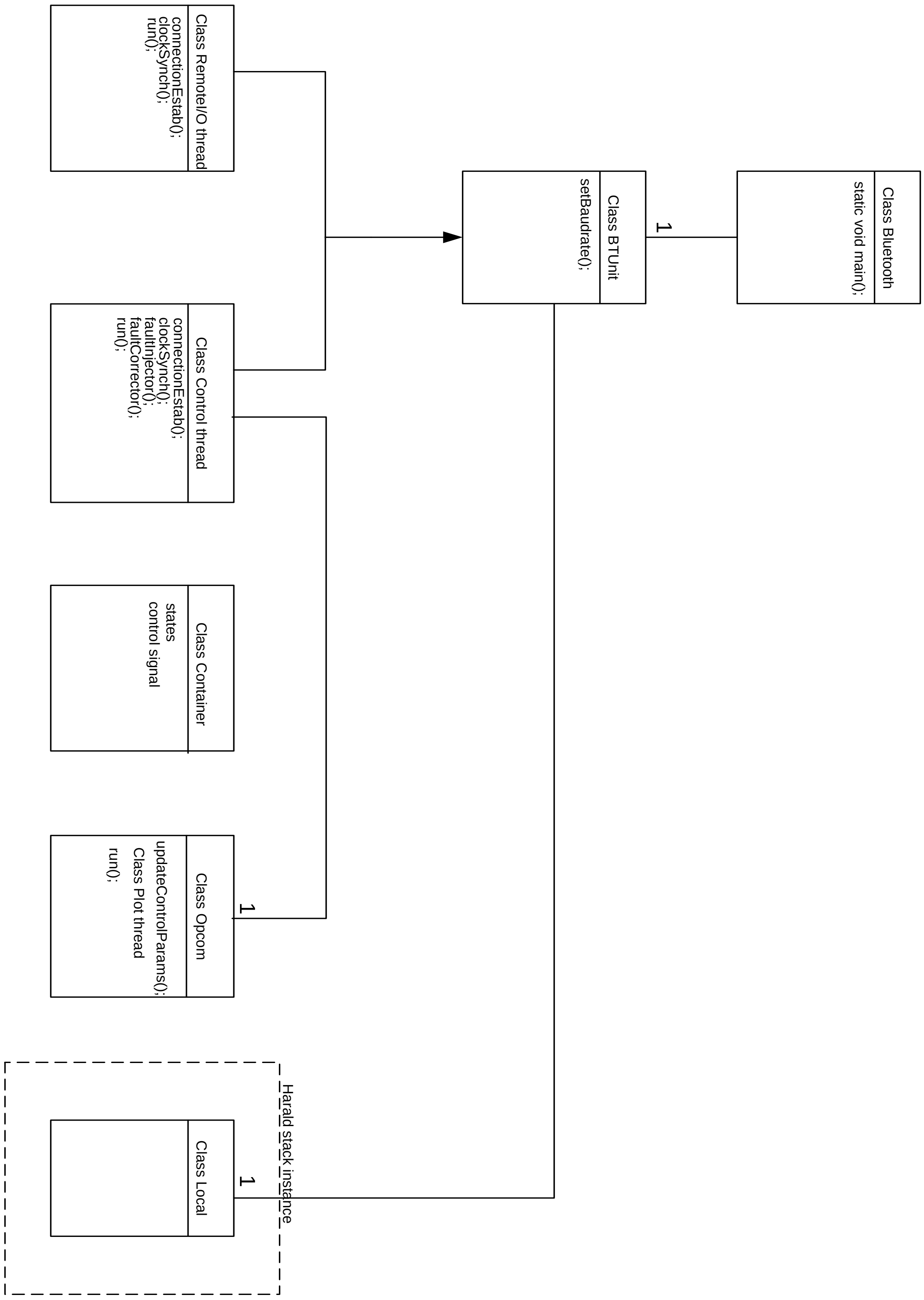


Figure 2.2 The control applications class diagram. The diagram displays the most important methods in each class, for more information look at source code.

2.3 Bluetooth communication

The Bluetooth communication setup is based on the hardware configuration of the Bluetooth Application and Training Toolkit. The Toolkit is comes with a RS232 port and an USB port. The RS232 port was used due to that the Harald software contained a RS232 driver but no USB driver. The setup of a communication chain where the user communicates through RS232 is not an optimal setup and is probably not a setup that will be used in the future. RS232 communication is much slower than the Bluetooth communication, which means that the RS232 communication limits the baud rate significantly. The USB port should be used in the future if available.

The limitations in the RS232 communication depend on configuration limitations on the RS232 port on the computer. The baud rate could only be configured to 115200 bits/s. The default baud rates of the Bluetooth chips are 56700 bits/s and they are configured to 115200 at each start up of the control application. Another problem was reading the RS232 receive buffer in the computer. The buffer was emptied with an interval of 10 ms, which resulted in a static delay in the communication chain of at least 50 ms. This delay is reduced by configuring the RS232 port on the computer to empty its buffer more frequently. The resulting delay was around 20 ms.
These alterations gave the communication chain properties sufficient for controlling the pendulum. The alterations resulted in a time diagram, see figure 3.3, with a total loop time of 18 ms. This can be compared to a loop time off 50 ms, which was obtained before the modifications were introduced.

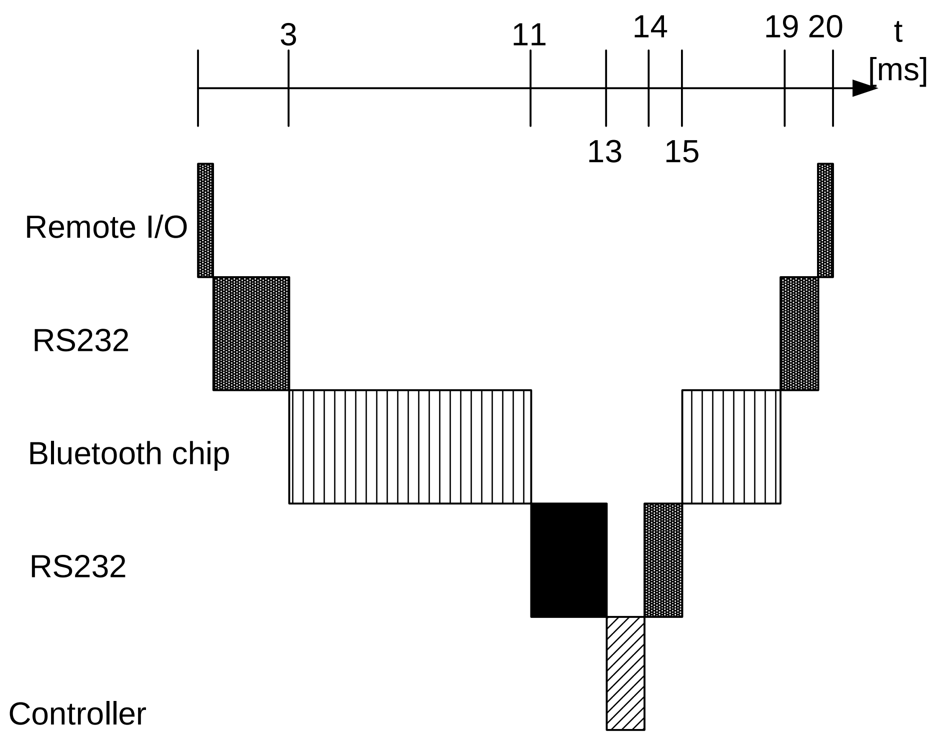
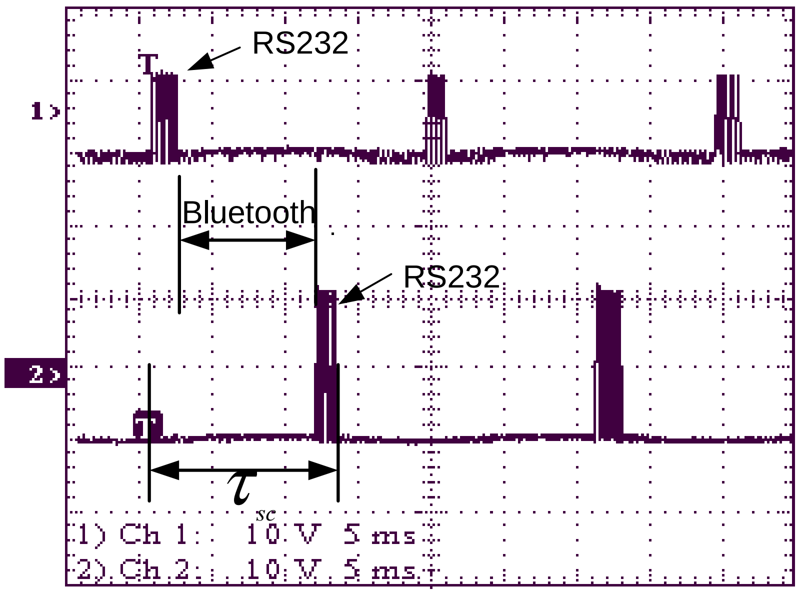


Figure 2.3 The timing diagram for the communication in the loop. The total duration of the timing diagram is equal to the total duration from sampling the process until receiving the control signal from the Control object.



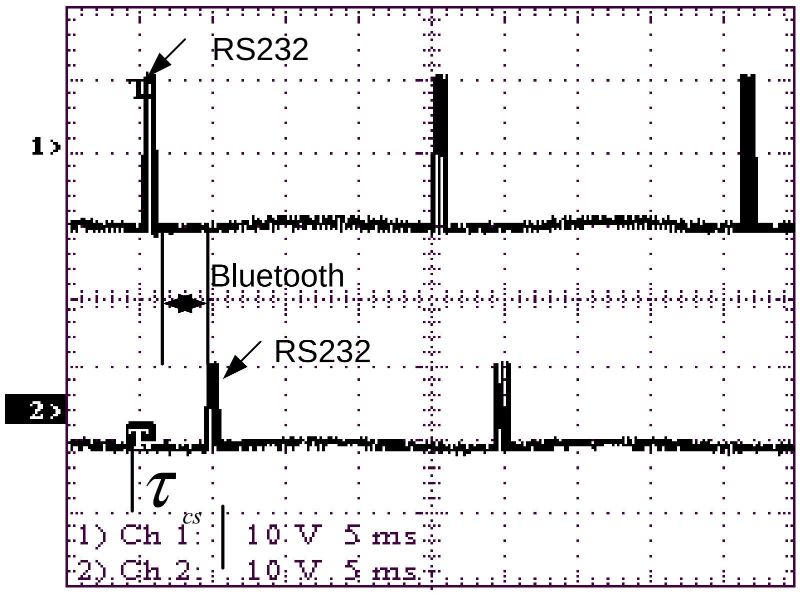


Figure 2.4 The test results sampling on the RS232 communication. The first picture displays measurements from when sending from Remote I/O to the Controller, where the measurements is obtained on Tx and Rx on the RS232 communication between the PC and the Bluetooth chip. τ is the delay from Remote I/O to Controller and the peaks show the serial communication over the RS232 cable. The second picture displays measurements when sending in the other direction. The difference in time between the communications peaks depends on the size of the package sent over RS232.

Information sent in the loop is e.g. controller settings, states and control signals. The communication uses a protocol, which specifies where to position the information in time and space, see Figure 2.5.

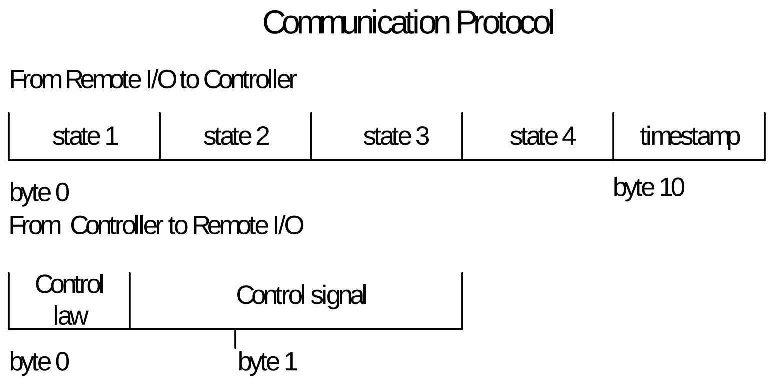


Figure 2.5 The communication protocol used in this work. State1-state4 are symbols for the sampled states from the process. The control law field specifies which control law to use in the system.

The Remote I/O is the master in the Bluetooth communication. The master in a Bluetooth network is the Bluetooth unit that controls the communication in the network. It is set to be master when the Bluetooth is initialized. The sampling in the Remote I/O is time-driven and the rest of the actions in the loop are event-driven, see figure 3.6.

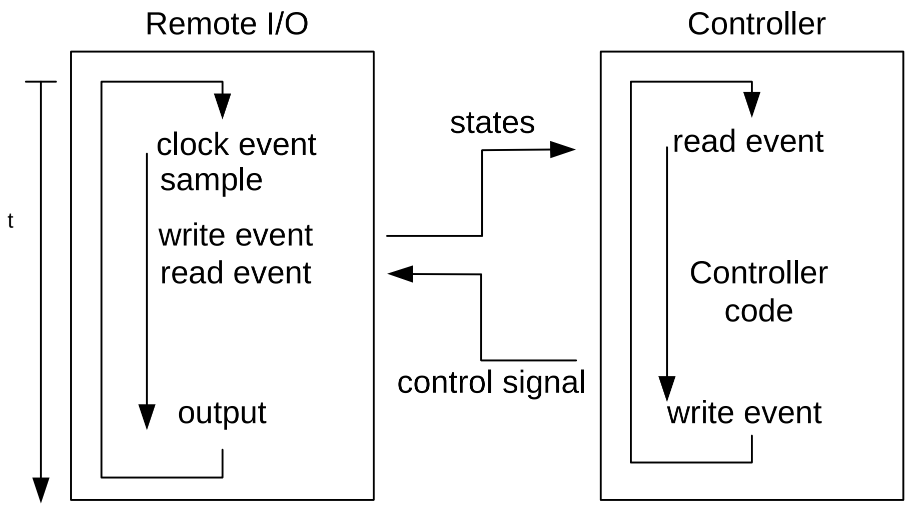


Figure 2.6 displays the interaction between the Remote I/O and the Controller. When the Remote I/O or the Controller receives a write or read event it continues executing the code as.

2.4 The Controller

The Controller is an instance of the data object Control (see Section 2.2) that contains the control laws. The Controller receives states from the Remote I/O and sends the calculated output control signal back to the Remote I/O. The Controller uses the communication protocol specified in Section Bluetooth communication to switch the control laws in run

time. The Controller allows switching of control laws when the pendulum is stabilised after swing up.
The controllers decides:

- What states to sample.
- How often the system should sample the process.
- Should the Remote I/O do further calculations. This depends on which control law that is used.

2.5 The Remote I/O

The Remote I/O samples the process and actuates the control signal to the process. The Remote I/O is discussed around two concepts, intelligent I/O and ordinary I/O. The ordinary I/O holds no intelligence, which means that the I/O only samples and puts out control signals. The ordinary I/O takes no notice of whether the control signal is correct or not. The ordinary I/O is easily implemented on an embedded microcomputer due to low calculation demands.
The intelligent I/O holds calculation possibilities and can alter the control signal depending on when it senses that something is incorrect with the control signal. The intelligent I/O is used for implementing stochastic delay compensation, due to its possibility to sense stochastic delays in the communication loop. Both the intelligent I/O and the ordinary I/O are implemented in the Remote I/O; they are started through the GUI.

3. The Furuta Pendulum

The Furuta pendulum was selected as a case study for this work on the basis of its unstable nature. This makes it suitable as a testbench for control experiments. If Bluetooth does not meet the communication demands for stable control of the Furuta pendulum it will fall down.

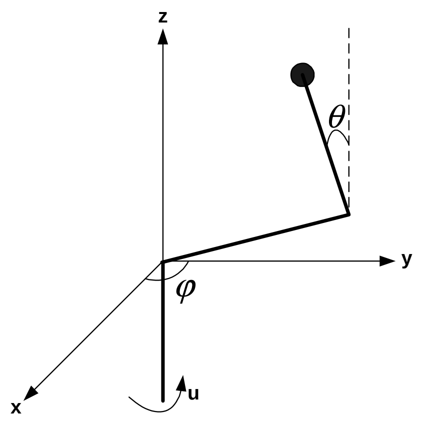


Figure 3.1 The Furuta pendulum.

3.1 The Furuta Pendulum Model

This section describes the mathematical model for the Furuta pendulum that was used when deriving the control laws.
The process consists of a rotating arm, to which a pendulum is attached (see Figure 3.1).

The angle θ is defined to be zero in an upright position and positive when the pendulum moves clockwise. The angle φ is defined to be positive when the pendulum is moving in counter clockwise. The pendulum's vertical axis is connected to a DC-motor that is controlled by the input u.

The complete derivation of the dynamics for the Furuta pendulum is excluded here. The derivation is based on Lagrange theory and can be read in Gäfvert [5].

Using the definitions made above, the equations of motion can be written:

$$\begin{aligned}
&\left(J_p + Ml^2\right)\left(\ddot{\theta} - \dot{\varphi}^2 \sin\theta\cos\theta\right) + Mrl\ddot{\varphi}\cos\theta - gl\left(M + {}^{m}\!/_{2}\right)\sin\theta = 0 \\
&Mrl\ddot{\theta}\cos\theta - Mrl\dot{\theta}^2\sin\theta + 2\left(J_p + ml^2\right)\dot{\theta}\dot{\varphi}\sin\theta\cos\theta \\
&+ (J + mr^2 + Mr^2 + (Jp + ml^2)\sin^2\theta)\ddot{\varphi} = u
\end{aligned} \tag{1}$$

Introducing

$$\begin{aligned}
a &= J_p + Ml^2 \\
b &= J + Mr^2 + mr^2 \\
c &= Mrl \\
d &= \lg(M + m/2)
\end{aligned}$$

the equations can be written:

$$\begin{aligned}
&a\ddot{\theta} - a\dot{\varphi}^2\sin\theta\cos\theta + c\ddot{\varphi}\cos\theta - d\sin\theta = 0 \\
&c\ddot{\theta}\cos\theta - c\dot{\theta}^2\sin\theta + 2a\dot{\theta}\dot{\varphi}\sin\theta\cos\theta + (b + a\sin^2\theta)\ddot{\varphi} = u
\end{aligned} \tag{2}$$

The coefficients for the pendulum used in the experiments are, see Svensson [6]:

$$\begin{aligned}
l &= 0.413\ \ m \\
M &= 0.01\ \ kg \\
Jp &= 0.0009\ \ kgm^2 \\
r &= 0.235\ \ m \\
J &= 0.05\ \ kgm^2 \\
m &= 0.02\ \ kg
\end{aligned}$$

3.1.1 Stabilization of the pendulum

The first objective in this work is to stabilize the pendulum in an upright position. The derivation of the linearized system is described in Johan Åkesson [2] and used here.

All states are measurable, which makes linear feedback control suitable. The state vector of the system is

$$x = (\theta \ \dot{\theta} \ \varphi \ \dot{\varphi})^T$$

System (2) is linearized around, the upright equilibrium point:

$$x = (0 \quad 0 \quad 0 \quad 0)^T$$

The linear system can be written:

$$\dot{x} = Ax + Bu = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dfrac{bd}{ab-c^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \dfrac{-cd}{ab-c^2} & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ \dfrac{-c}{ab-c^2} \\ 0 \\ \dfrac{a}{ab-c^2} \end{bmatrix} \tag{3}$$

For digital control design, a sampled data description is used. The sampled system is written

$$x_{n+1} = \Phi_1 x_n + \Gamma_1 u$$

where the matrixes Φ_1 and Γ_1 depend on the sampling interval *h.* The control law used to stabilize the pendulum is

$$u(k) = -Lx(k)$$

The feedback L vector can be calculated in many ways. Here LQ design is used. This design method is straightforward to apply. LQ design allows two design parameters, a Q matrix and R matrix. Q and R are penalty matrixes for the states *x* and the input *u*. Suitable Q and R matrixes were found to be:

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 60 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$R = 80$$

3.1.2 Swing Up

Swing up of the pendulum is a nice demonstration feature. The pendulum is swung from a downward position to an upright position using energy control. The idea is to fill the system with energy until the pendulum is positioned in the upright position. When in the upright position the pendulum is caught and stabilized. The strategy of swing up is described in Åström and Furuta [4]: Below the algorithm is briefly described.

Swing up uses a simplified model of the Furuta pendulum, which neglects the arm dynamics. This model is derived from an inverted pendulum on a linear rail and not a Furuta pendulum, which is a sufficient approximation:

$$\begin{aligned} Jp\ddot{\theta} &= Mgl\sin\theta - Mla\cos\theta \\ \ddot{\varphi} &= a \end{aligned} \qquad (4)$$

Introduce the normalizations

$$\omega_0 = \sqrt{\frac{Mgl}{Jp}}$$
$$u = \frac{a}{g}$$

and the equations can be written:

$$\begin{aligned} \ddot{\theta} &= \omega_0^2 \sin\theta - \omega_0^2 u\cos\theta \\ \ddot{\varphi} &= ug \end{aligned}$$

The control law for energy control is:

$$u = sat\left(k\left(E_n - E_o\right)sign\left(\dot{\theta}\cos\theta\right)\right) \qquad (5)$$

Sat is saturation for the energy control law and k is proportional design parameter. E_n describes the present energy of the system and E_0 describes the desired energy of the system and is thereby zero in a upright position. E_n is calculated using the same definitions above:

$$E_n = \frac{E}{Mgl} = \frac{\dot{\theta}^2}{2\omega_0} + \cos\theta - 1$$

The control law (8) swings the pendulum to an upright position, where it must also be caught. This is done with a modified control law that uses feedback from the arm angle more gently. The design method used to derive this control law is LQ design with large punishments on the θ states and smaller punishment on the φ states.

3.1.3 Friction compensation

Friction is a problem that shows up when trying to control the pendulum. Typically, friction will cause the arm to move periodically around a position φ_n. The friction is regarded as a Columbs friction with stiction in the arm. The friction in the pendulum pivot point is neglected. Friction compensation for the Furuta pendulum is briefly described in this report, for more information see Henrik Olsson [8].

The friction model can be written:

$$F_f(\dot{\varphi}, u) = \begin{cases} F_c^+ & \dot{\varphi} > 0 \\ F_s^+ & \dot{\varphi} = 0, \quad u > F_c^+ \\ u & \dot{\varphi} = 0, \quad F_s^- < u < F_s^+ \\ F_s^- & \dot{\varphi} = 0, \quad u < F_s^- \\ F_c^- & \dot{\varphi} < 0 \end{cases}$$

The friction model is simplified by:

$$F_s^+ = F_c^+$$
$$F_s^- = F_c^-$$

The control law is then altered to compensate for the friction by adding F_f to the control law in use.

$$u = -Lx + F_f$$

The friction compensation improves the control performance and reduces the periodic movement of the pendulum arm. For more detailed information see Johan Åkesson [2].

4. Bluetooth control algorithms

In this section, control algorithm designed for distributed hard real-time control application using Bluetooth as a communication network are described. These algorithms should improve the control performance when using Bluetooth despite the special properties of retransmissions, delays and external disturbance.

4.1. Delay compensation

Delay compensation aims at reducing the effects of delays in the communication loop. Two kinds of delays are discussed in this section, static delays and stochastic delays. Static delays are represented by delays of equal size at all time. Stochastic delays are represented by delays that vary in size in time.

4.1.1 Static delay

A distributed system using Bluetooth will always contain delays when communicating from one node to another. A control system using Bluetooth as described in this work will always contain a static delay (see figure 4.1). This delay can be regarded as static within a unique pair of nodes if not exposed to external disturbances. A control system exposed to a delay is less stable, due to loss of phase margin.

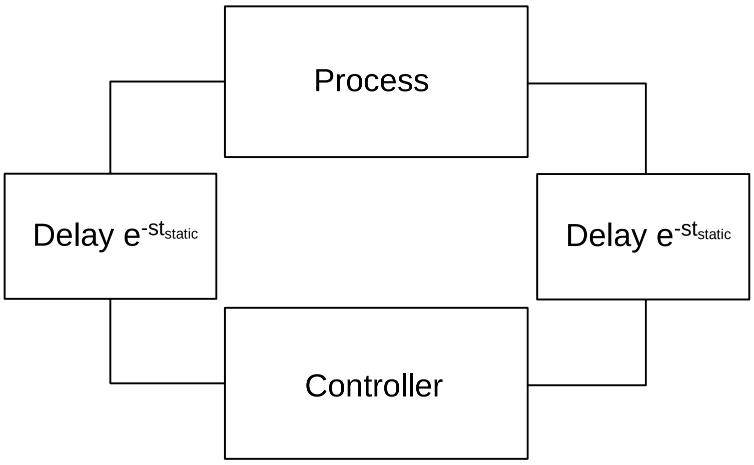


Figure 4.1 A system using Bluetooth with static delay. The static delay is due to that the process and the controller communicates with Bluetooth.

The phase margin loss can be written:

$$Delay = e^{-\tau s}$$
$$Phasem\arg in \quad reduction : \arg(e^{-\tau w_c}) = -\tau\omega_c$$

where ω_c is the cross-over frequency of the system.

Introducing a new state, the previous control signal $u_{n-1,}$ in the control law, reduces the effect of static delay, see Karl Johan Åström and Björn Wittenmark [7]. The new control law can be written:

$$u = -Lx \Rightarrow u_{static} = -L\begin{bmatrix} x \\ u_{n-1} \end{bmatrix}$$

The extended description of the system is:

$$\begin{bmatrix} x_{n+1} \\ u_n \end{bmatrix} = \begin{bmatrix} \Phi_1 & \Gamma_2 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_n \\ u_{n-1} \end{bmatrix}_n + \begin{bmatrix} \Gamma_1 \\ 1 \end{bmatrix} u$$

This system was used to calculate the new feedback vector L. The method used was a discrete LQ-design with a static delay of 21 ms. The static delay increases when the code in the application is enlarged.

4.1.2 Stochastic delay

Disturbance triggers retransmissions in the Bluetooth communication, which results in stochastic delays, Johan Nilsson [3], in the loop see figure 4.3. The delays are said to be stochastic due to the unpredictability of the retransmissions endurance and place in time. A retransmission could range from 0 seconds (no retransmission) up to total shut down.

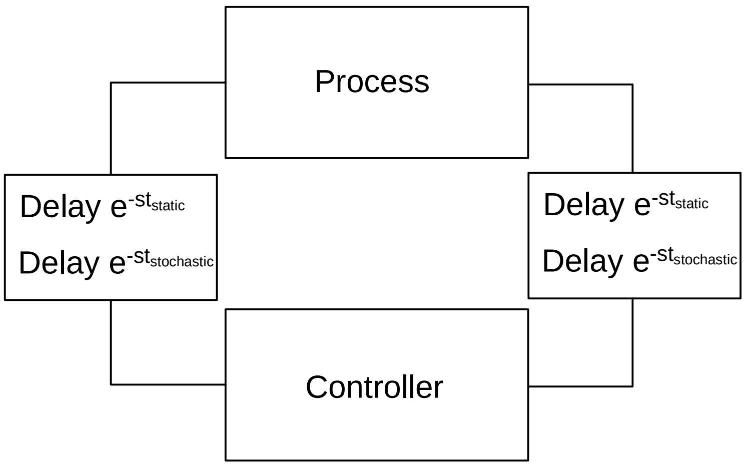


Figure 4.2 A system with stochastic delays. The stochastic delays are due to that the process and the controller communicate with Bluetooth.

The control law must be altered in order to assure stable control when the system is exposed to stochastic delays. The derivation of this control law is not explained in this work, but is explained in Johan Nilsson [3]. It can gently be described as a dynamic version of the static delay control law. An approximate version of the optimal stochastic controller was used in this work:

$$u = -Lx \Rightarrow u_{stochastic} = -L_0 \begin{bmatrix} x \\ u_{n-1} \end{bmatrix} - \frac{dL}{d\tau}(\Delta\tau)x \begin{bmatrix} x \\ u_{n-1} \end{bmatrix}$$

The L-vector for each delay is plotted and a trend can be estimated from which *dL/dτ*-vector can be estimated.

L_0 is the feedback gain for the nominal delay in the system where *Δτ* =0. The stochastic control law becomes active and dynamic when *Δτ* is nonzero. Figure describes how the *dL/dτ* is estimated in order to get a linear model of the stochastic control law.

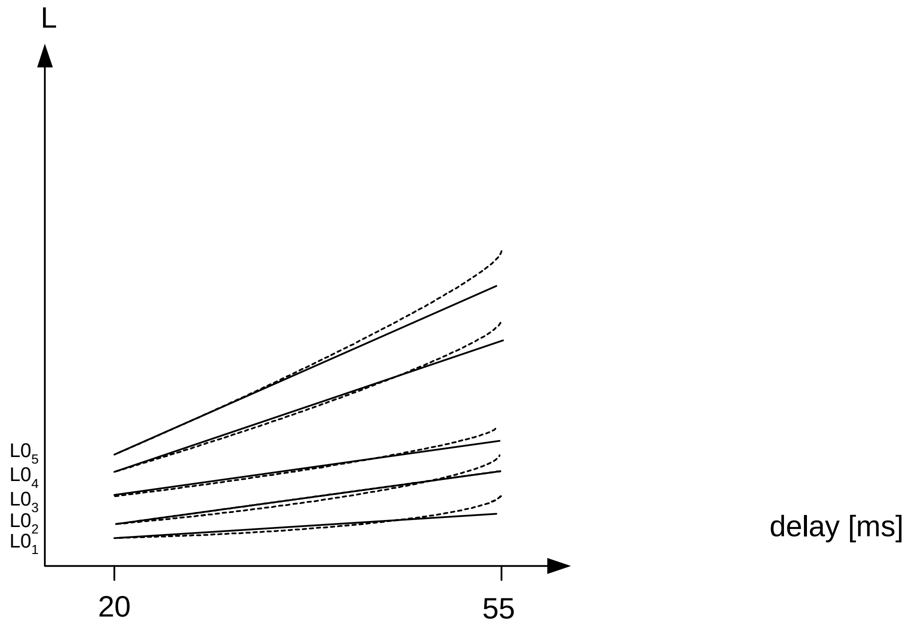


Figure 4.3 The plotted L vectors (dotted line) and the dL/dτ (full line) estimations for the L vectors. The estimations are done in order to get a linear function of the control law.

4.2 Bit error filter

A problem that can occur in the loop is bit errors in the Bluetooth packets sent in the air due to disturbance. Disturbances are common in an industrial environment, which makes bit errors a valid problem to study.
Bit errors can either be filtered by the Bluetooth baseband layer or by the user. If the baseband layer filters the Bluetooth packets, it orders retransmissions until the target receives intact packets. Retransmission works by CRC filtering in the baseband layer and if the CRC doesn't match, a retransmission is ordered by the target and the transmission procedure restarts. The retransmission scheme can result in large delays in the loop if the disturbances are heavy. Choosing a certain Bluetooth packet type initializes the retransmission scheme.
Another method of dealing with bit errors and reducing the delays is to disable the retransmission scheme, which may result in bit errors in the target. However, the bit

errors can be filtered and the Bluetooth packet can be used, even though it initially contained bit errors. This scheme removes the retransmission delays and may prove more successful in stabilizing the process. It does not know in advance if it receives a Bluetooth packet containing bit errors or not. The filter must therefore decide whether the packet contains errors or not, which isn't a trivial problem.

4.2.1 Filter design

The bit error filter is designed as an observer that runs concurrent with the process and checks the validity of the Bluetooth packet the target receives. The target is the controller and the packet contains the sampled states of the system. Concurrent in the controller is the observer estimating process states.

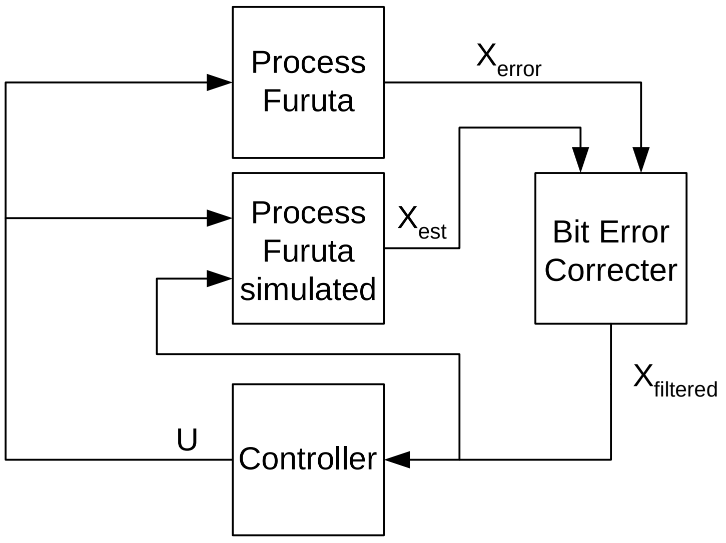


Figure 4.4 The bit error filter model. The filter consists of the BEC and simulated process.

Bit faults are corrected by comparing the estimated states with the sampled states. The comparison locates the bit error and flips the faulty bit if a faulty bit seems to exist. The sampled states are represented by voltage span of 10-(-10) volts displayed by a bit array of 12 bits. The states are thereby sent as a bit array with a value span of 0-4096. As the states are converted to a value represented by a bit array of 12 bits, faulty bits can easily be detected by looking at the difference between the sampled and the estimated states. The estimated states must be converted to the same value span, 12 bits, in order to do a correct comparison.

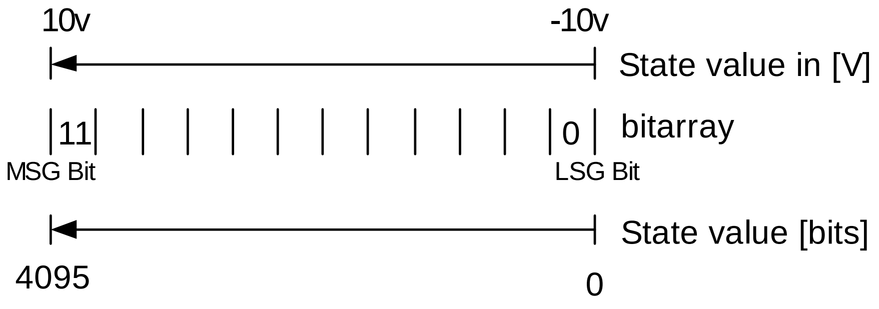


Figure 4.5 Sampled state and its bit array representation

The observer design uses a linearized model of the Furuta pendulum with a static delay compensation state (u_{n-1}). The observer is fed back with the corrected X_{filtered} states to keep equal to the real process. The observer equations are given by:

$$\begin{bmatrix} x_{n+1} \\ u_n \end{bmatrix} = \begin{bmatrix} \Phi_1 & \Gamma_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_n \\ u_{n-1} \end{bmatrix}_n + \begin{bmatrix} \Gamma_1 \\ 1 \end{bmatrix} u + K(Cx_{filtered} - Cx_n)$$

Since all states are measurable, K is a 5 by 4 matrix.

The bit error corrector locates the faulty bit by looking at the difference Δ between the sampled state and the estimated state. The difference indicates which bit that is incorrect, due to that the bit location is related to the flipped bits value ($2^n{}_{\text{bit}}$).

$$|x_{est} - x_{error}| = \Delta = 2^{n_{bit}}$$

$$n_{bit} = round\left(\frac{\log(|x_{est} - x_{error}|)}{\log(2)} \right)$$

The observer will estimate poorly if exposed to sequential bit errors. The filter can therefore only filter certain bits on each state. This accuracy on each state is marked by a threshold position in the bit array. The filter never flips bits under the threshold.

4.2.2 Simulated Bit error filter

The filter design was evaluated in Matlab, with the Simulink model shown in Figure 4.9.

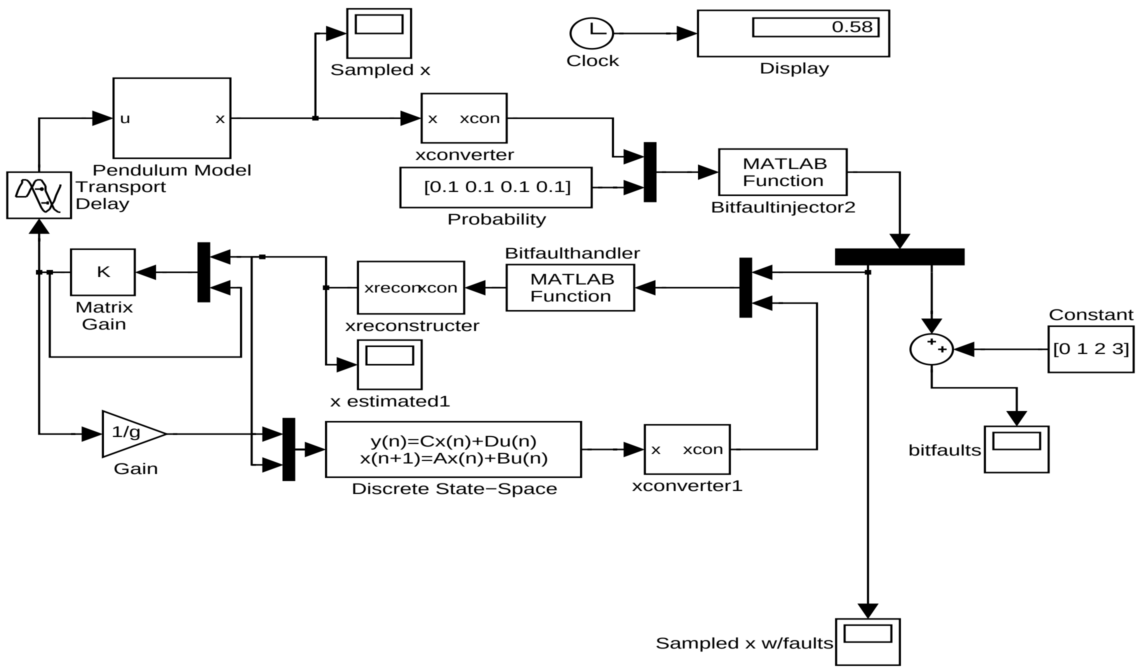


Figure 4.6 The filter design in simulink.

The process and the simulated process execute concurrent in time. The bit error filter compares the sampled states and the estimated state and produces correct states. A bit fault injector injects stochastic bit errors in each state in each packet sent from the process to the controller. The filter doesn't know in advance where and when to find a bit error. The convert functions convert states to fit the bit array model and the reconstruct model transforms the converted states back to normal states with the right unit of measurement.
The model simulates Bluetooth properties with the transport delay block between sampling and actuating.

4.3 Control laws

The control application contains six different control laws, which are shifted in runtime. The following control laws are available:

$$u_{catch} = -L_{catch}\left[x\right]$$
$$u_{stable} = -L_{stable}\left[x\right]$$
$$u_{static} = -L_{static}\begin{bmatrix} x \\ u_{n-1} \end{bmatrix}$$
$$u_{stochastic} = -L_{stochastic}\begin{bmatrix} x \\ u_{n-1} \end{bmatrix} - \frac{d\tau}{dL}(\Delta\tau)\begin{bmatrix} x \\ u_{n-1} \end{bmatrix}$$
$$u_{disturbanc} = -L_{disturbanc\ e}\begin{bmatrix} x \\ u_{n-1} \end{bmatrix}$$
$$u_{swingup} = sat(k(E_n - E_0)sign(\dot{\theta}\cos\theta))$$

The control law for stochastic delays are divided between the Remote I/O and the Control tasks. This is due to that $\Delta\tau$ must be calculated in the Remote I/O. The Catch and Stable control laws are used when the pendulum is caught in a upright position.

5. Bluetooth control functions

This section describes some functions necessary in a distributed control systems using Bluetooth, clock synchronization and time stamps.

5.1 Clock synchronization

In order for the clocks in the system to not differ, a clock synchronization method is executed in the beginning of initialization of the controller and the remote I/O. The clock needs to be synchronized in order to recognize delays in the system, due to that the Control and Remote I/O tasks runs on separate hardware with separate clocks. If the native clocks are not synchronized they cannot correctly decide at which time a data packet is received. The implemented clocks on the Remote I/O and the Control tasks are cyclic clocks with a resolution of a millisecond and cycle length of 65536 ticks.

The clock synchronization scheme is derived from a model described in Johan Nilsson [3]. This clock synchronization is built around two assumptions, which Bluetooth meets:

- The time to send a data packet from master unit to receiving it in the slave unit is equal to the time to send and receiving the same data packet in the reverse order.
- The difference in time between the native clocks is small.

5.1.1 Clock synchronization scheme

The scheme is implemented such that the master unit will contain the system clock and a slave unit synchronizes its clock to the master unit.

$$
\begin{aligned}
&M = master_{unit}\\
&S = slave_{unit}\\
&t_i = absolte\ \ time\\
&t_i^S = absolute\ \ time\ \ in\ \ S\\
&t_i^M = absolute\ \ time\ \ in\ \ M\\
&\delta = time\ \ deviation\ \ from\ \ t_i
\end{aligned}
$$

The absolute time in the master unit is regarded as the reference time:

$$
\begin{aligned}
t_i^M &= t_i\\
t_i^S &= t_i - \delta^S
\end{aligned}
$$

The slave unit's offset from the master unit is:

$$t_i^M - t_i^S = \delta^S = \delta$$

The scheme starts with a time request from the master unit, and the slave unit answers with a time response. The start time and the stop time are registered.

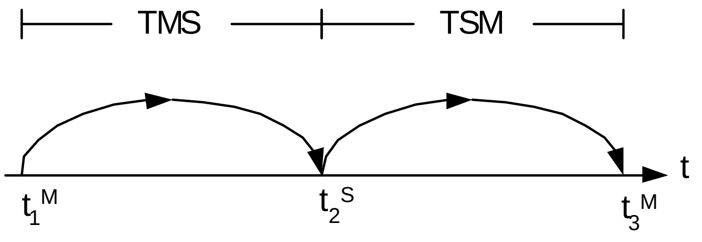


Figure 5.1

According to the time diagram (see figure 4.5.) it follows that:

$$
\begin{aligned}
TMS &= t_2^M - t_1^M = \left(t_2^S - \delta\right) - t_1^M\\
TSM &= t_3^M - t_2^M = t_3^M - \left(t_2^S - \delta\right)
\end{aligned}
$$

According to the assumptions,

$$TMS = TSM \Rightarrow \delta = E\left\{\frac{2t_2^S - t_1^M - t_3^M}{2}\right\}$$

The E operator stands for the expected value of δ. The expected value of δ has been implemented as a mean value of 9 consecutive estimations of δ.

5.2 Time stamps

Time stamps are implemented on every package sent in the control application. The stamps are used for recognizing delays in the system. Each time a unit receives a data packet from a remote unit it can control the validity of the received packet through the time stamp. If the data packet is too old it is useless in the control application. A condition for the time stamps to be relevant is that the native clocks in the system are synchronized. Without the clock being synchronized, a comparison is without value.

The protocol for sending timestamps is displayed in figure 5.2. Each timestamp is positioned last in all packages sent.

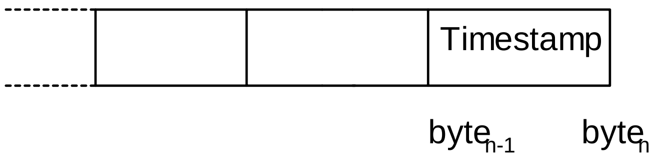


Figure 5.2

6. Experiments

The static and stochastic delay compensation and the bit error filter were tested on the Furuta pendulum. These three experiments were carried out in order to validate the control algorithms. Each experiment had to be carried out with its own special properties:

- Static delay compensation: a static delay of 21 ms.
- Stochastic delay: a static delay of 21 ms and stochastic delays. The stochastic delays are introduced in the control loop with a assumed exponential probability distribution shown in Figure 6.1.

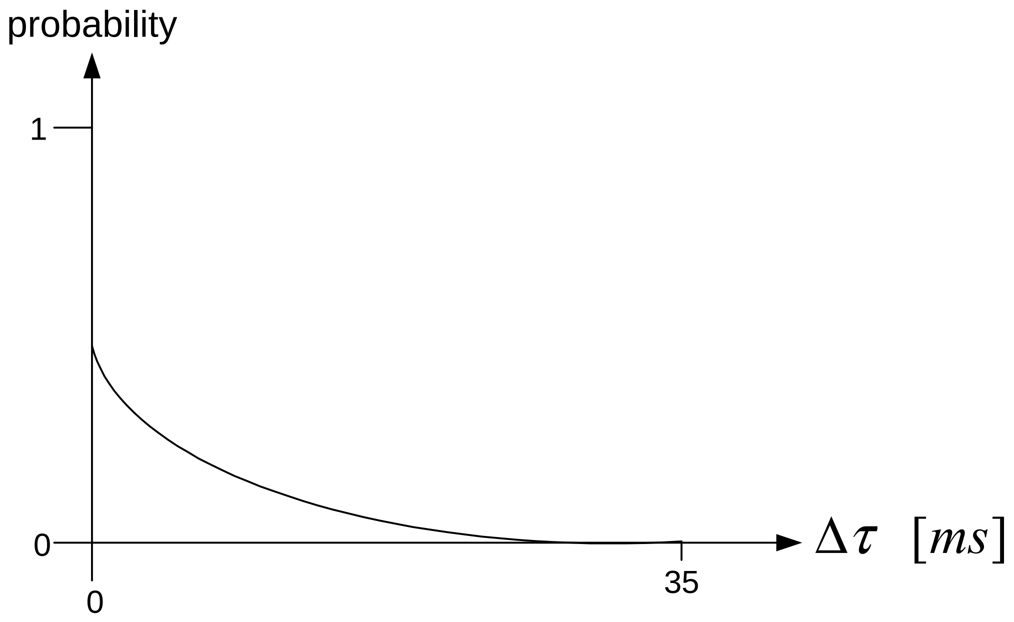


Figure 6.1Exponential distribution of the stochastic delay. It is most likely that a short delay is inserted into the loop.

- Bit error filter: bit errors are injected into the sampled states, and a static delay of 21 ms.

It is important the realize that the static delay of 21 ms is a natural delay inserted into the loop due to the use of Bluetooth, while the bit errors and the stochastic delays are artificially inserted into the loop. The fault injector does this.

The fault injector is an instrument for testing the control laws and ideas. A fault injector is used in the loop in order to have control over which fault that is inserted into the control loop. There is no easy way of exposing the system to fault caused by natural disturbance. The fault injector is discussed as a unit but is distributed in the control loop. One part is situated in the Remote I/O and the rest in the Controller.

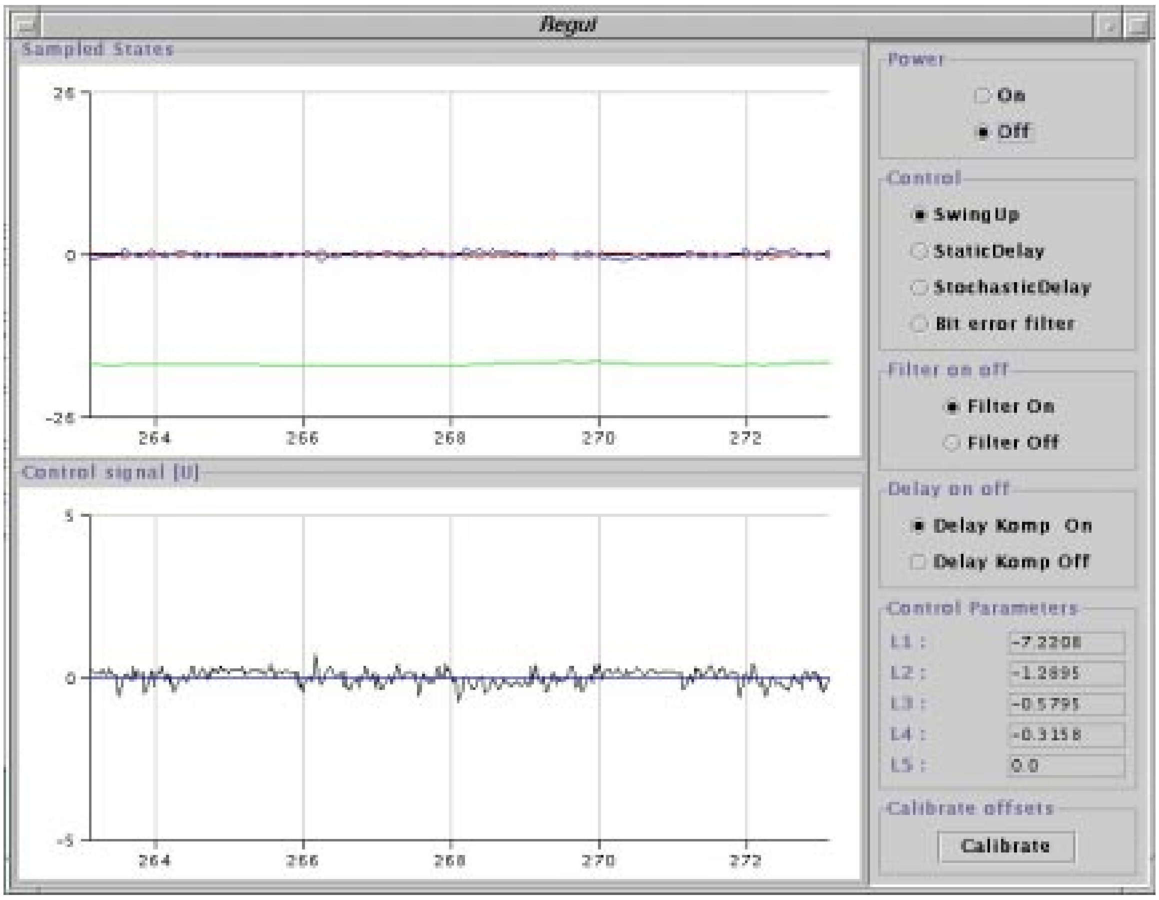
The bit error injector is located in the Controller and injects bit faults in the sampled states from the Remote I/O. The Controller receives the sampled states from the Remote I/O, which send the states to the fault injector. The fault injector then injects faults in the states and sends them to the bit error corrector (see figure 4.7).

The delay injector is located in the Remote I/O and injects delays into the control loops. This is done between sampling and putting out the control signal in order to simulate delays in the system. The static delay is not simulated in the control loop due to that it exists naturally. The stochastic delays are injected by updating a delay variable in the Remote I/O before putting out the control signal simulates the stochastic delays.

7. GUI

The GUI (Graphical User Interface) is a demo feature for easy understanding and control of the process. The GUI gives the user of the demo easy access to all features of the pendulum demo.

The layout of the GUI is shown in Figure 7.1:



Power: On [✓ Off]

Control: [✓ SwingUp] StaticDelay StochasticDelay Bit error filter

Filter on off: [✓ Filter On] Filter Off

Delay on off: [✓ Delay Komp On] Delay Komp Off

Control Parameters

L1 :	-7.2208
L2 :	-1.2895
L3 :	-0.5795
L4 :	-0.3158
L5 :	0.0

Calibrate offsets

Figure 7.1 The layout of the GUI..

The power panel gives the user possibility to start and stop the demo. The off and on buttons are connected to the Controller, which checks the power state every time a loop begins. If off is set the Controller becomes inactive and the whole communication loop is stopped and thereby the whole demo.
The Control panel gives the user easy access to enable different control laws at any time in the demo. It is important to know that control laws are automatically switched when the pendulum has fallen down. The GUI doesn't reflect this, which means that controls shouldn't be switched until the pendulum is stabilised.
The filter panel gives the user easy access to the state of the filter. The filter is shut off whenever the filter off button is set, and this will inevitable lead to that the pendulum will

fall down. The filter is turned on when the filter on button is set; this gives the pendulum stable characteristics although exposed to injected faults.
The delay compensation panel allows the user to switch the state of the delay compensation. These buttons are related to both static and stochastic delays. These buttons should therefore be observed whenever switching delays.
The L panel updates the feedback gains when a control law is switched in the GUI.
The calibrate panel is used when the power off is set in order to allow calibration of offsets to measured states.

8. Results

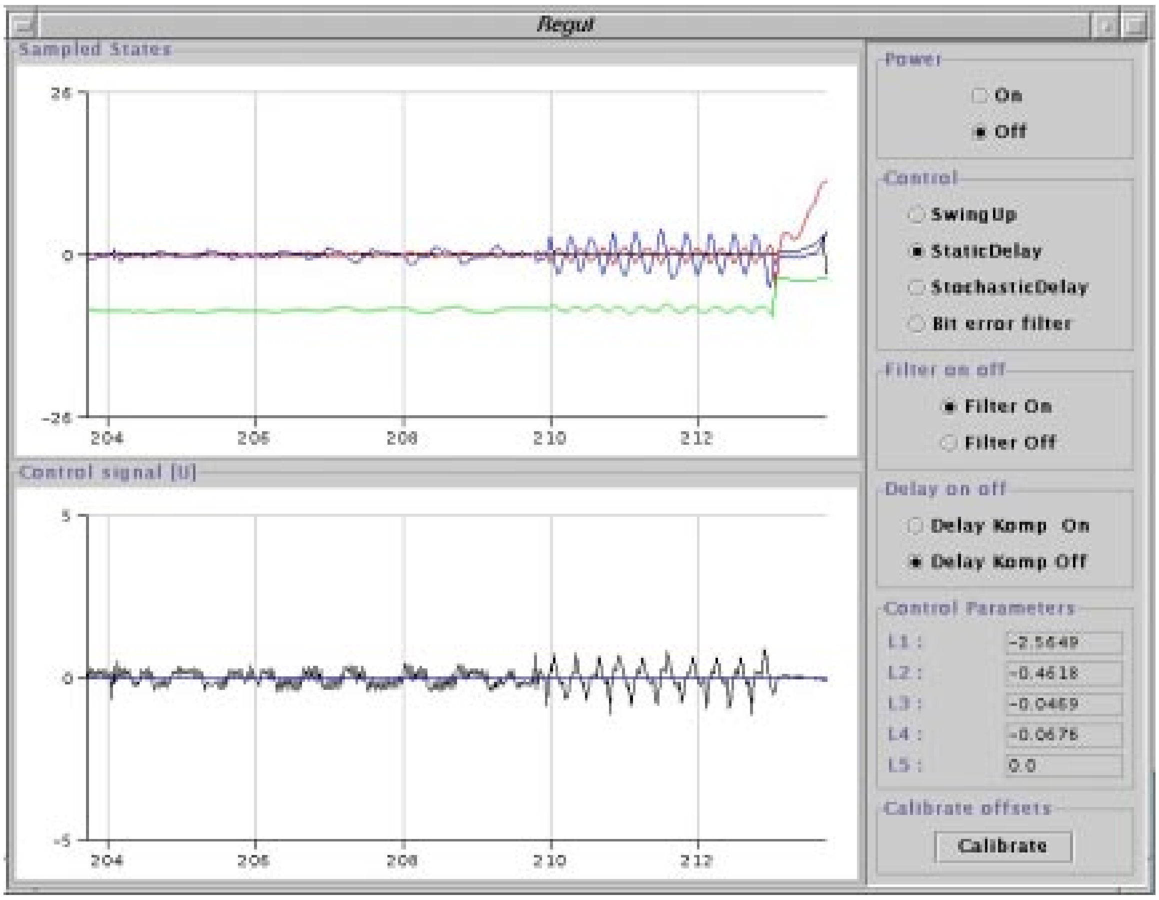
This section presents the results of the suggested control algorithms. The results are promising and could with further work be implemented in the industry.

8.1 Results with static delay compensation

The test of the static compensation was done with a extreme sampling interval in order to get a visual effect of the static delay compensation:

$$h=0.07 \text{ s}$$

The effect off not using the static delay compensation, when the system is exposed to a static delay, is that the pendulum becomes highly unstable. This fact is displayed in figure 8.1.



Power: On [✓ Off]

Control: SwingUp [✓ StaticDelay] StochasticDelay Bit error filter

Filter on off: [✓ Filter On] Filter Off

Delay on off: Delay Komp On [✓ Delay Komp Off]

Control Parameters

L1 :	-2.5649
L2 :	-0.4618
L3 :	-0.0469
L4 :	-0.0676
L5 :	0.0

Calibrate offsets

Figure 8.1 The effects of static delay compensation. One can clearly see where the static delay compensation is switched. The pendulum becomes highly unstable when the static delay compensation is turned off and finally falls down. This fact is displayed in the figure by the states in the figure. The amplitude off the states is larger when the static delay is turned off. By t=210, the delay comp is turned off.

The penalty matrixes Q and R for the new static control law chosen were:

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

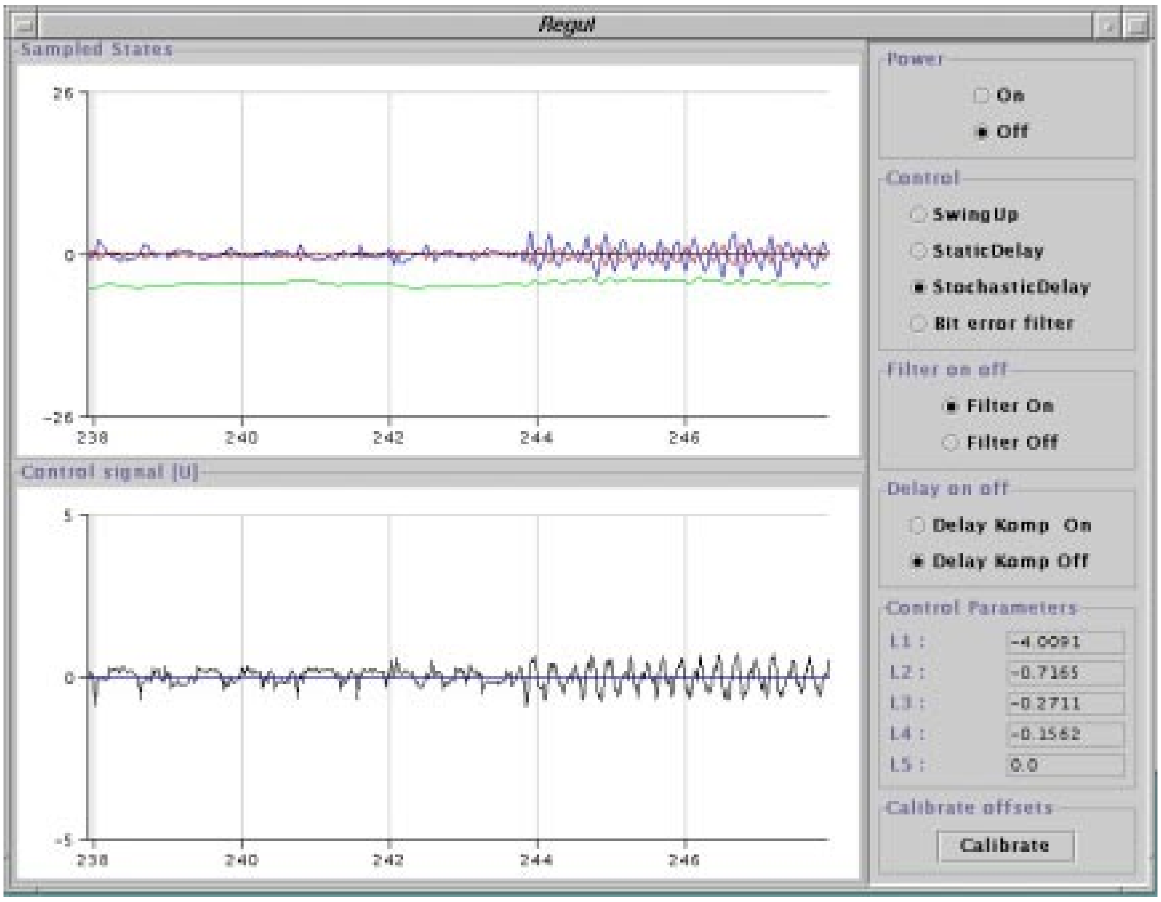
$$R = 20$$

8.2 Results with stochastic delay Compensation

The result of the stochastic delay compensation is positive. The test of the stochastic compensation was done with an extreme sampling interval in order to get a visual effect:

h=0.06 s

The effect of not using the stochastic delay compensation when the system is exposed to stochastic delays is that the pendulum becomes highly unstable or falls down. This fact is displayed in Figure 8.2.



Regul

Power: On [✓ Off]

Control: Swing Up, StaticDelay, [✓ StochasticDelay], Bit error filter

Filter on off: [✓ Filter On] Filter Off

Delay on off: Delay Komp On [✓ Delay Komp Off]

Control Parameters

L1 :	-4.0091
L2 :	-0.7165
L3 :	-0.2711
L4 :	-0.1562
L5 :	0.0

Calibrate offsets: Calibrate

Figure 8.2 The effects of stochastic delay compensation when the system is exposed to stochastic delays. The figure is taken from the GUI when the stochastic delay compensation is switched. One can clearly see that the pendulum becomes unstable and may fall down when the compensation is switched off. By t=244, the delay compensation is switched off.

The penalty matrixes *Q* and *R* for the stochastic control law were chosen as:

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 60 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R = 100$$

8.3 Results simulated bit error filter

The simulated bit error filter filtered the bit errors sufficiently to keep the process stable, although that there was no information when a bit error occurred in time and in which state. Figure 8.3 shows the performance if the system with filter and when exposed to static delay.

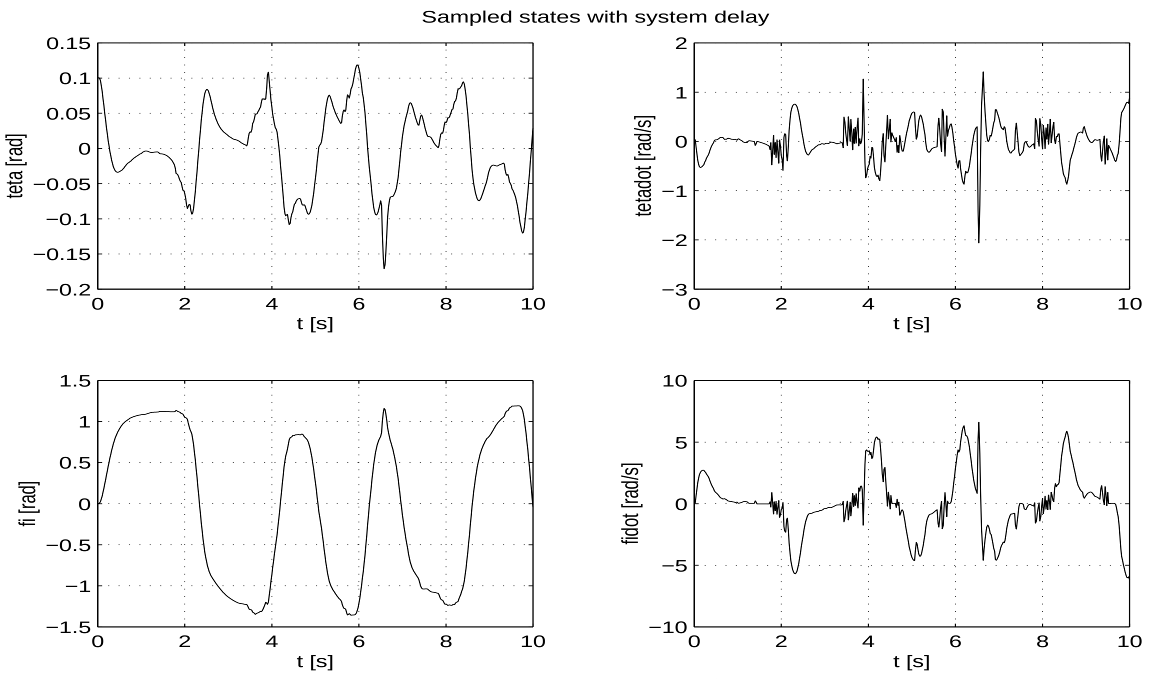


Figure 8.3 The sampled states from the process before the bit faults are injected.

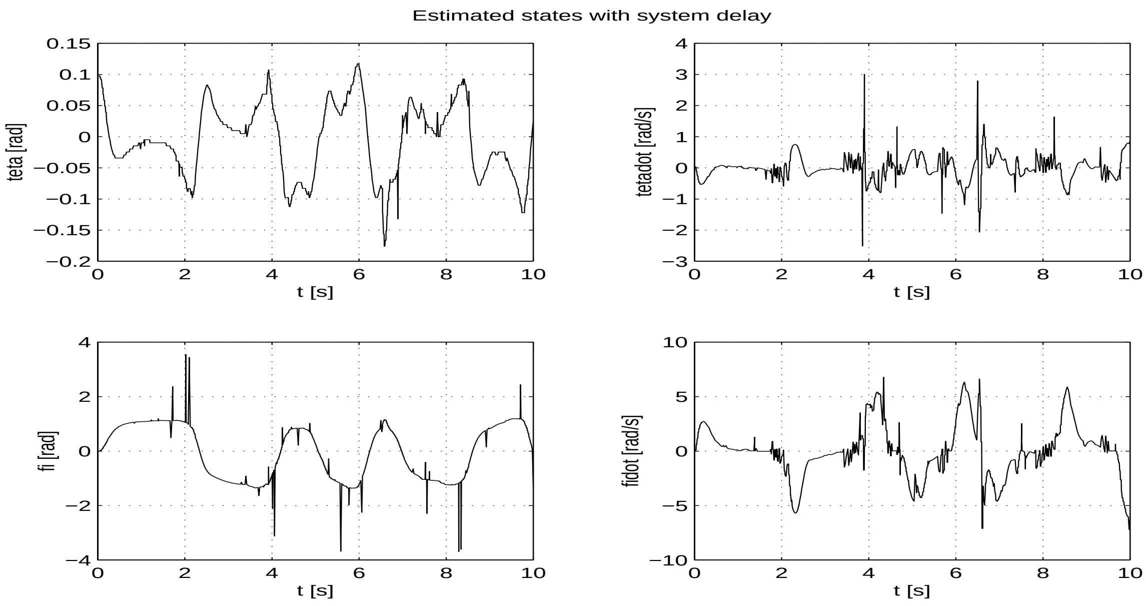


Figure 8.4 The estimated states the bit error corrector uses to compare the sampled states containing bit errors. These states are filtered; the bit faults shown are due to the accuracy of the filter. The filter does not correct bit fault under the specified accuracy specified in the application. State fi is plotted with a larger scale than in Figure 8.3.

One can see that some faults are present in the sampled states this is due to the bit fault filters accuracy. The bit errors are injected to each state with a probability of 0.1 for each state. Figure 8.5 displays the states with injected bit faults.

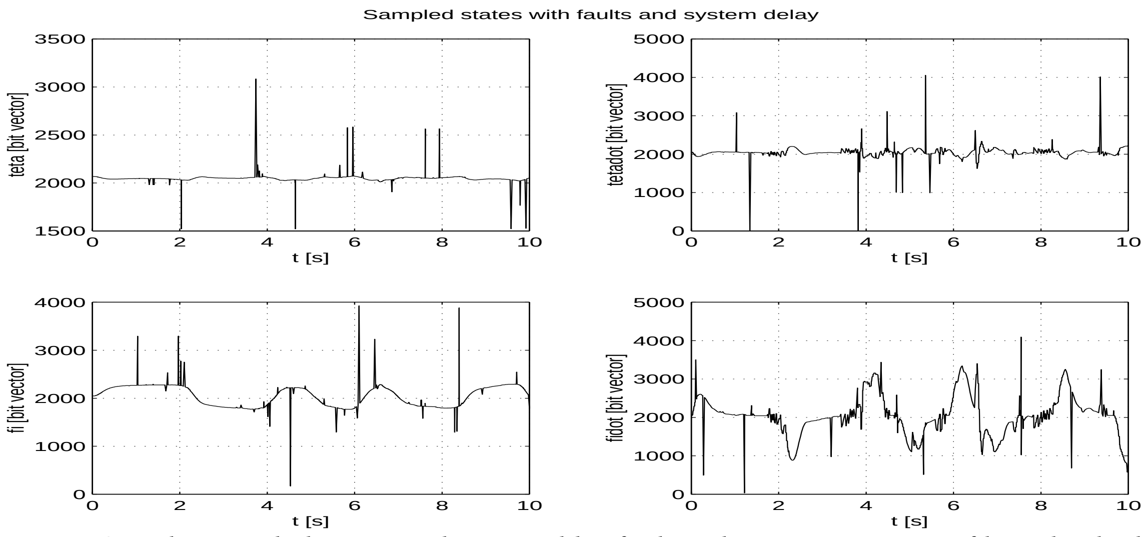


Figure 8.5 The sampled states with injected bit faults. These tastes are not filtered, which shows in the large bitfaults.

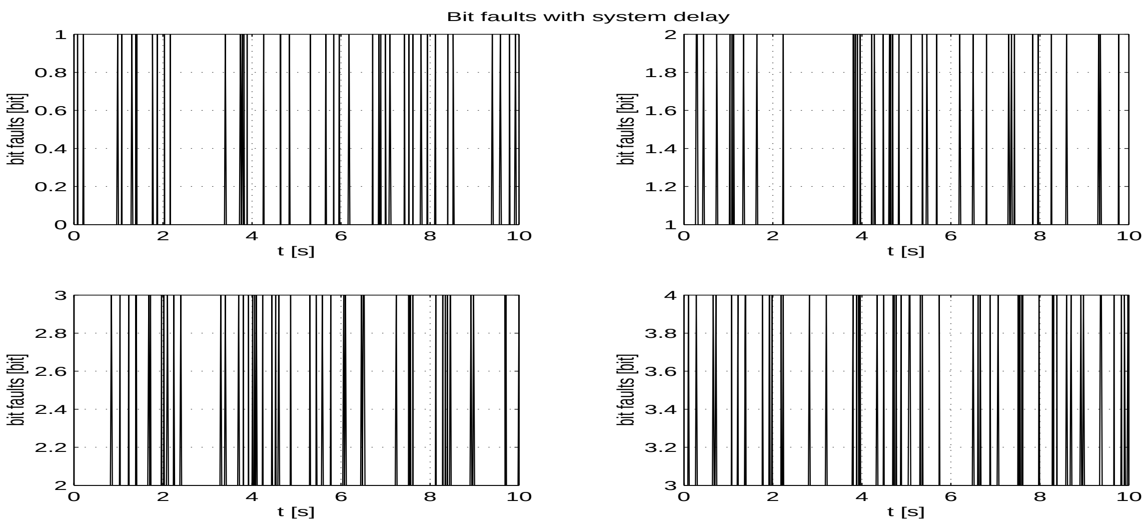


Figure 8.6 Bit faults injected into each state. Each black line is a bit fault injected to a state.

The filter works sufficiently when simulated in Matlab.

8.4 Results with implemented bit error filter

The bit error filter was implemented on the process with Bluetooth communication and tests. The same setting as in the simulated filter was used. The process is a bit unstable which is natural depending on the accuracy of the bit error corrector's accuracy, when ran with the filter. The results can be seen in the demo

9. Future Work

9.1 Time-outs

Time-outs are an idea not tested in this work, however the idea is to have some safety when the Bluetooth communication is completely shut down. Such a disruption can be fatal in some applications and must therefore be avoided.

One solution is to implement a watchdog timer on the communication channel in order to discover shut down in the communication. The watchdog timer should be implemented in the remote I/O due to that isolation of the process and the remote I/O in case of a communication breakdown.
The watchdog timers give the read and write methods a time window to work within. The communication is regarded as shut down if this time window is exceeded.
The remote I/O must contain a communication breakdown method in case of watchdog timer overflow. This method should secure graceful degradation of the process.

10. Conclusions

The static delay compensation is effective in distributed communication networks to fight bandwidth (loss off phase margin in control application) problems. The demo uses large samplings interval in order to get a visual effect in the demo, however static delay compensation gives effects at small sampling intervals. Static delay compensation should be used when control application is exposed to bandwidth limitations. Choosing USB communication with the Bluetooth chip can reduce this problem.

The stochastic delay compensation is effective in distributed communication networks to fight unpredictable delays and bandwidth problems (loss off phase margin in control application). The unpredictable delays are an effect off retransmissions in Bluetooth chips. The demo uses large samplings interval in order to get a visual effect, however stochastic delay compensation gives effects at small sampling intervals. Stochastic delay compensation should be used the control application is exposed to disturbance which gives retransmissions in the Bluetooth communication and bandwidth problems.

Static delay compensation does not demand an intelligent I/O, which is the case with stochastic delays. This is due to that stochastic delay demands calculation power in the Remote I/O. Calculation power of results in a more advanced I/O and thereby a more expensive I/O.

The bit error filter can be used to reduce retransmissions to shorten delay time in a system. The bit error filter works well making the pendulum stable although exposed to bit errors. The strength of the filter is that it doesn't need to have knowledge about when and where a bit fault occurs in order to work. The filter is an interesting idea applicable to hard real-time distributed systems. The filter could be developed further to filter multi bit errors. The idea is to flip several bits instead of single bit faults. This would make the filter more applicable to real process. Further development could be done be give the filter notice on when to suspect errors and maybe the location of the bit faults. Using the information from the CRC coding implemented in Bluetooth could do this. The bit error filter should be used when the control application is exposed to disturbance. It is important to know that this idea is built around that the Bluetooth chip does not filter any bit errors. Removing the retransmissions scheme could do this.

The conclusion is that much can be done in order to improve the sometimes-bad characteristics of Bluetooth in control applications. The suggested control algorithms are effective in their specific areas and each should be selected to fit the environment at site.

11. References

[1]. Eker Johan: *Harald stack.* Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.
[2]. Åkesson Johan: *Safe Manual Control of Unstable Systems*. Master thesis ISRN LUTFD2/TFRT --5646 --SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 2000.
[3]. Nilsson Johan: *Real-Time Control Systems with Delays* PhD thesis, ISRN LUTFD2/TFRT--1049--SE, February 1998, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, May 1996.
[4]. Åström K.J and K Furuta: *Swinging up a pendulum by energy control. Automatica (1999).*
[5]. Gäfvert Magnus: *Modeling the Furuta Pendulum* Technical Report, ISRN LUTFD2/TFRT--7574—SE, Dept. of Automatic Control,Lund Institute of Technology, August 1998.
[6]. Svensson J: Effects of friction on the Furuta Pendulum. Master thesis ISRN LUTFD2/TFRT-- 5593 --SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1998.
[7]. Karl Johan Åström and Björn Wittenmark: *Computer controlled systems—Theory and design.* Prentice-Hall, third edition 1997.
[8]. Henrik Olsson: *Control systems with friction.* Doctoral Dissertation ISRN LUTFD2/TFRT --1045-- SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, April 1996.
[9]. Kurt: http://ittc.ukans.edu/utime/