

Distributed Wireless Control Using Bluetooth

Johan Eker

Department of Electrical Engineering and Computer Science
University of California at Berkeley
337 Cory Hall, Berkeley, CA 94720-1770, US
johane@eecs.berkeley.edu

Anton Cervin

Department of Automatic Control
Lund Institute of Technology
Box 118, SE 221 00 Lund, Sweden
anton@control.lth.se

Andreas Hörjel

ConnectBlue
Stora Varvsgatan 11 N:1
SE 211 19 Malmö, Sweden
andreas.horjel@connectblue.se

Abstract

Bluetooth is currently emerging as one of the most promising personal wireless network technologies. The automation industry is also showing interest in using Bluetooth for more industrial applications. However, problems with unreliable links and time delays must be explicitly addressed in any wireless control application. This paper discusses two specific problems that may occur when using Bluetooth in a control loop: long random delays and bit errors. Simple solutions to these problems are presented. The ideas have been tried out, both in simulations and in experiments, on a rotating inverted pendulum controlled over a Bluetooth network.

1. Introduction

In a distributed control system the sensors, the actuators and the controller are physically displaced and communicate over a network. The main advantage of this setup is flexibility. It allows us to use the same control computer for several control loops and it relieves us from the potential problem of placing the controller close to the controlled process. It also allows us to build our control systems in a hierarchical fashion. The cost for using networks has decreased radically over the last years which has made distributed solutions much more appealing. However, a downside to using a distributed approach is that delays will be introduced in the control loops due to latencies in the network communication. The delays will lead to decreased phase margins in the control loops and potentially unstable systems.

In this paper a wireless Bluetooth radio network is used. Going wireless has some obvious advantages;

without cables we have a much greater freedom to physically distribute the nodes. It is possible to place actuators and sensors without worrying about the location of the control node. For example, it allows us to place the sensor and actuator on a mobile object and still run the controller from a stand-still platform. However, going wireless will also result in less dependable systems. The likelihood of a wireless link containing bit errors is several orders of magnitude larger than for a wired link. This means that the possibility of a message being corrupt or delayed due to retransmissions is much larger.

Two particular problems are studied in this paper. In the first case, we assume that we have a reliable connection, i.e. the underlying network protocol guarantees that there will be no corrupted messages. The cost for this is that messages may be retransmitted several times, causing lengthy delays. The implications for the controller design is that we must explicitly deal with the varying time delays. In the second case, we assume an unreliable connection. We will receive possibly scrambled messages, but we will have a short and nearly constant delay. The problem in this case is to detect and handle the bit errors. We use a model-based observer approach for error detection and correction.

The two problem cases have been simulated in MATLAB/SIMULINK and then implemented and run over a wireless Bluetooth link controlling a rotating inverted pendulum. To control the experiments, random delays and bit errors were artificially introduced by fault injection in the Bluetooth protocol stack. More details about the Bluetooth-pendulum implementation are available in [Hörjel, 2001].

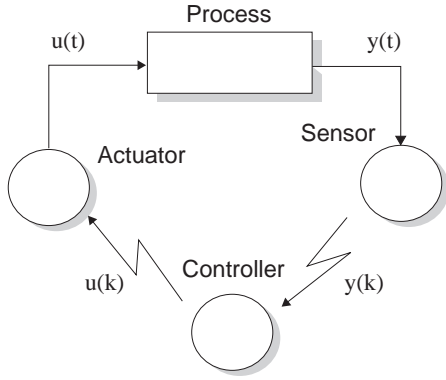


Figure 1 The distributed system configuration used in the paper. The communication between the sensor, the actuator, and the controller is over a wireless link.

2. The Setup

The distributed control configuration is shown in Figure 1. The feedback loop consists of three parts: the sensor, the controller, and the actuator. The sensor node is time-driven while the controller and actuator nodes are event-driven. The sensor samples the process periodically and sends the measurement values to the controller. Upon receipt, the controller calculates a new control signal and sends it to actuator node which outputs the value. In our setup, the sensor node and the actuator node are located in the same hardware unit, called the Remote I/O.

2.1 The Network

Bluetooth is currently emerging as one of the most promising personal wireless network technologies. It is primarily designed as a cable replacement between for example a cell phone and a PDA, but there is a growing interest to also use it in industrial applications. Up to eight Bluetooth units may be connected in a so called Piconet. The communication protocol is time divided, with a slot-length of $625 \mu\text{s}$. One unit acts as the master, and the slaves are only allowed to transmit if they were addressed by the master in the previous slot. The radio uses the Industrial-Scientific-Medical (ISM) frequency band, which ranges in Europe and the USA from 2.400 to 2.4835 GHz. The maximal data rate is 1 Mb/s, but the useful data rate is much lower. Interference is avoided by using a frequency-hop, spread spectrum technology, with 1600 hops per second. This gives relatively short packet length and good interference protection. For more information on Bluetooth see [Bluetooth Consortium, 2001].

2.2 Bluetooth in the Loop

The implementation structure is shown in Figure 2. In our setup the controller is the master and the Remote I/O is a slave. The theoretical minimum

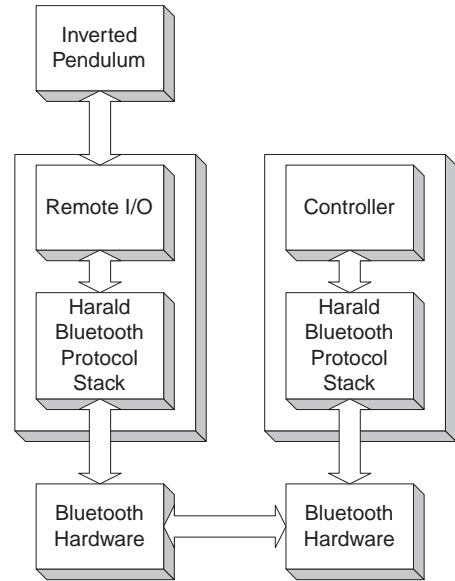


Figure 2 The implementation structure.

round-trip delay in Bluetooth is $2 \cdot 625 \mu\text{s}$. However, the length of the data packets, the Bluetooth protocol stack, and the communication between the Bluetooth hardware and the computer hardware bring the minimum round-trip delay up to 17 ms in our implementation. Still, this is fast enough to control the inverted pendulum, which is a process with fast and unstable dynamics.

3. The Pendulum Controller

The rotating inverted pendulum used in the experiments is shown in Figure 3. The objective is to control the pendulum angle θ and the arm angle ϕ to zero by applying the torque u to the rotating arm. The non-linear equations of motion can be written

$$\begin{aligned} \alpha \ddot{\theta} - \alpha \dot{\phi}^2 \sin \theta \cos \theta + \gamma \ddot{\phi} \cos \theta - \varepsilon \sin \theta &= 0 \\ \gamma \ddot{\theta} \cos \theta - \gamma \dot{\theta}^2 \sin \theta + 2\alpha \dot{\theta} \dot{\phi} \sin \theta \cos \theta \\ &+ (\beta + \alpha \sin^2 \theta) \ddot{\phi} = u \end{aligned}$$

Introducing the state vector

$$x = \begin{pmatrix} \theta & \dot{\theta} & \phi & \dot{\phi} \end{pmatrix}$$

and linearizing around the upright equilibrium a state-space description is given by

$$\frac{dx}{dt} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ a_{21} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ a_{41} & 0 & 0 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ b_2 \\ 0 \\ b_4 \end{pmatrix} u$$

The full state vector is directly measurable on the process. The sampling interval for digital control was

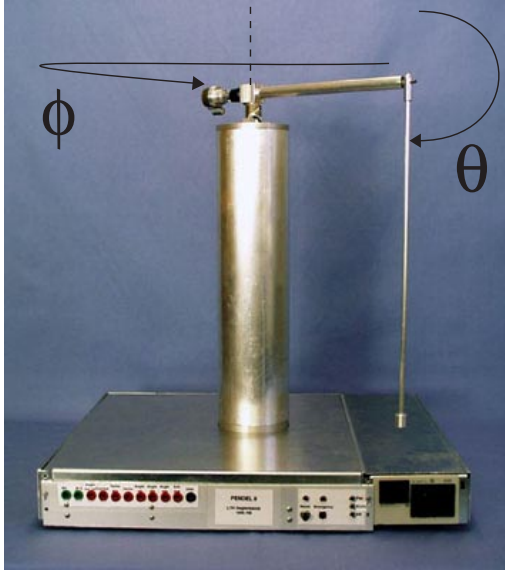


Figure 3 The rotating inverted pendulum used in the experiments. The objective is to stabilize the pendulum in the upright position $\theta = 0$.

chosen as $h = 60$ ms. Discrete state-feedback controllers were designed based on an linear-quadratic (LQ) formulation where the control should minimize the cost function

$$J = \lim_{T \rightarrow \infty} \frac{1}{T} \mathbb{E} \left\{ \int_0^T (x^T Q_1 x + u^T Q_2 u) dt \right\}$$

The cost J can also be measured and used to compare the performance of different control schemes for the pendulum.

4. Coping with Varying Delays

Delay in the control loop always has an adverse effect on control performance. For example, a constant delay of one sample decreases the phase margin of the system by 11 to 34 degrees, assuming that the sampling interval has been chosen according to the rule of thumb $0.2 \leq \omega h \leq 0.6$ [Åström and Wittenmark, 1997]. Here, h is the sampling interval and ω is the cross-over frequency of the closed-loop system.

The case of constant delay is easy to cope with in the control design. The sampled-data description of the plant including the control delay will simply be of higher order, and design can be made using standard techniques. The control law has the form

$$u(k) = -L \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}.$$

where L is a constant feedback gain vector, $x(k)$ is the state vector and $u(k-1)$ is the old control signal.

The case of randomly varying delays is more difficult. A simple but non-optimal solution is to do static delay compensation for the mean delay.

The optimal LQ-controller for random sensor-to-controller and controller-to-actuator delays was derived in [Nilsson *et al.*, 1998]. The controller performs dynamic delay compensation by adjusting the feedback gain according to the actual delays. The optimal control law has the form

$$u(k) = -L(\tau_k^{sc}) \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} \quad (1)$$

where the feedback gain vector L is now a function of the sensor-to-controller delay τ_k^{sc} in the current sample.

In Figure 4, the cost J as a function of the amount of stochastic delay has been computed for different control schemes. When the delay becomes very long, the dynamically compensating controller is still stable ($J < \infty$), while the other control schemes become unstable.

4.1 Dynamic Delay Compensation with Intelligent I/O

Dynamic delay compensation works better the shorter the controller-to-actuator delay is compared to the sensor-to-controller delay. If the controller is located *at* the actuator, the delay in the current sample can be known exactly and the correct gain can be applied. Modifying the setup is of course not a real solution, but using an I/O with only very limited computational power, this ideal behavior can be emulated closely. The idea is illustrated on the pendulum controller.

The optimal feedback gain depends on the distribution of the round-trip delay. For the case of heavy

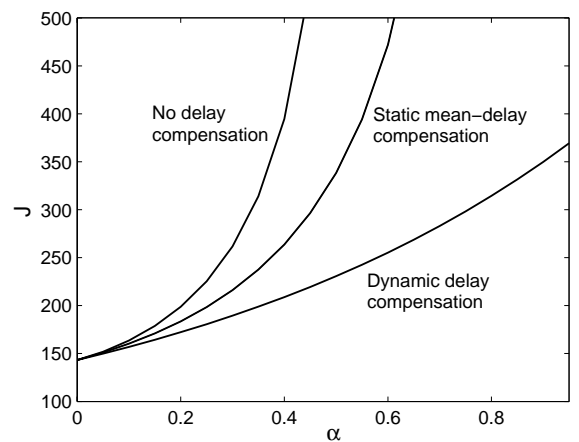


Figure 4 Values of the cost function J for the pendulum controller under different schemes. In this plot, the control delay is uniformly distributed on $[0, \alpha h]$.

disturbances and frequent retransmissions, we assume the exponentially decaying delay distribution shown in Figure 5.

The optimal gain vector is computed using the formulas in [Nilsson *et al.*, 1998], and the result is shown in Figure 6. We notice that the optimal gain can be closely approximated by a linear function of the delay τ . Approximating Eq. 1, we write

$$u(k) \approx -L_0 \begin{pmatrix} x(k) \\ u(k-1) \end{pmatrix} - \frac{dL}{d\tau} \begin{pmatrix} x(k) \\ u(k-1) \end{pmatrix} \tau$$

In the controller node, $x(k)$ and $u(k-1)$ are known, but τ is still unknown. However, the controller can precompute

$$\hat{u}(k) = -L_0 \begin{pmatrix} x(k) \\ u(k-1) \end{pmatrix}$$

and

$$\lambda(k) = -\frac{dL}{d\tau} \begin{pmatrix} x(k) \\ u(k-1) \end{pmatrix}$$

These two scalars are then sent to the Remote I/O. If the I/O keeps track of the round-trip delay τ , it can do the simple adjustment

$$u(k) = \hat{u}(k) + \lambda(k)\tau$$

before applying the control signal to the process.

4.2 Simulation

The suggested delay compensation scheme was simulated against the nonlinear pendulum model. The pendulum was disturbed by white process noise and the round-trip delay in the control loop varied between 20 and 55 ms according to the distribution shown in Figure 5. The simulation result is shown in Figure 7. This was compared to a controller which was designed for a constant delay of 20 ms (the nominal round-trip delay). The result of this simulation is shown in Figure 8. For this particular control problem, the dynamic delay compensation scheme was able to reduce the cost J by about 30 %.

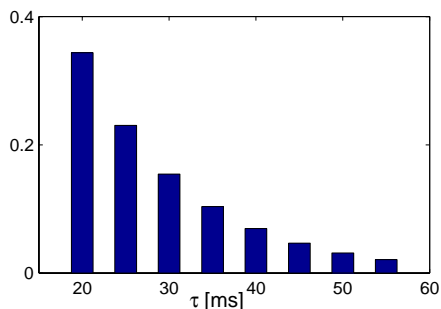


Figure 5 The assumed delay distribution.

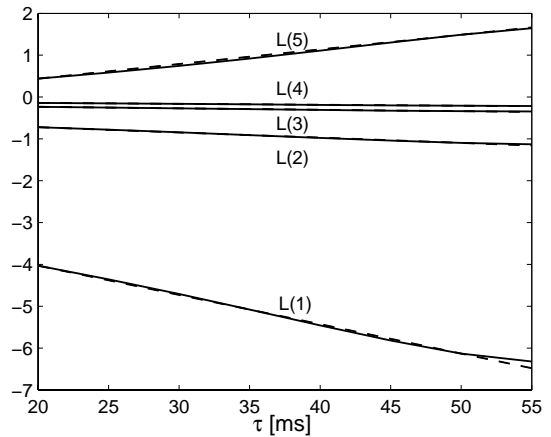


Figure 6 Optimal gain vector (full lines) and linear approximation (dashed lines).

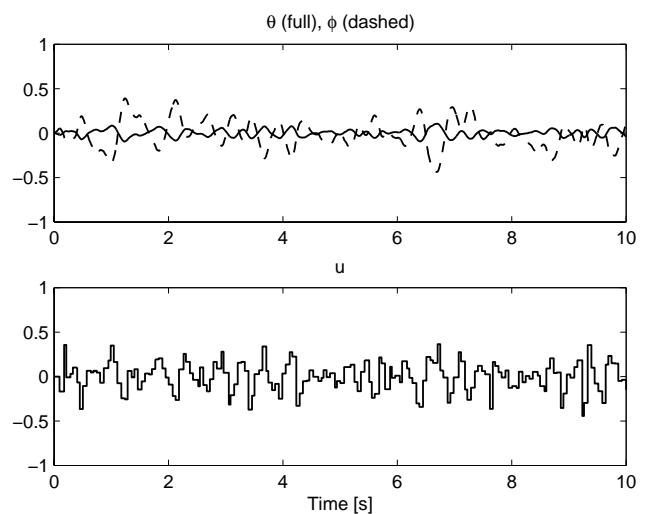


Figure 7 Simulation of the pendulum controller with random communication delays and dynamic delay compensation.

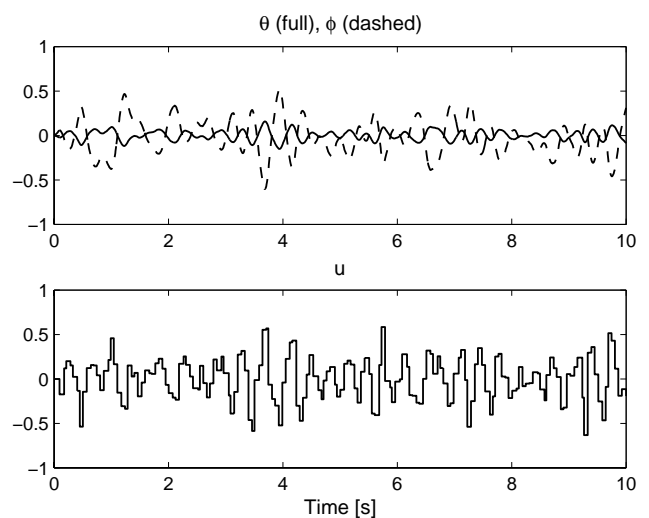


Figure 8 Simulation of the pendulum controller with random communication delays and static delay compensation.

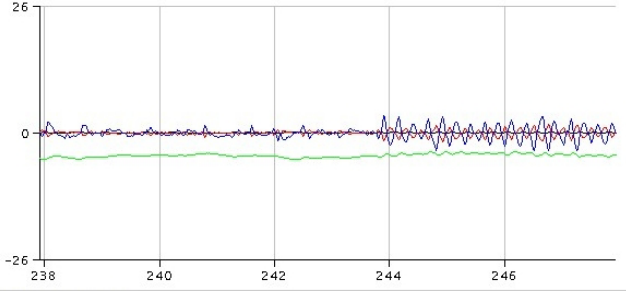


Figure 9 Experiment on the Bluetooth-pendulum setup with random communication delays. At time 244, the dynamic delay compensation is turned off, and the pendulum starts to oscillate. A large part of the increase in the cost is due to unmodeled friction.

4.3 Experiment

The dynamic delay compensation was also tried on the distributed Bluetooth-pendulum setup. The random delays were not due to actual disturbances but were injected in the control loop on purpose. The result of an experiment is shown in Figure 9. Due to unmodeled friction, the pendulum easily started to oscillate when the delay compensation was turned off.

5. Coping with Faulty Messages

Another problem that might occur is bit errors in the Bluetooth packets due to disturbances. These errors are currently handled by the Bluetooth baseband layer. When an error is encountered in the baseband layer, the packet is wasted and a retransmission is requested. This retransmission may need several tries before the packet can be delivered error free. The retransmission scheme can result in large delays in the loop if the disturbance is heavy. The delays will lead to decreased control performance, as explained in the previous section. An alternative solution is to reconstruct the information from the faulty packages, and hence avoid the long random delays.

Our solution to the bit-error problem is to run an observer (a pendulum model) in the controller node, in parallel with the real process. When samples are received from the sensor node, the potentially disturbed measurement value is compared bitwise to the output from the pendulum model. If there is an error in one of the most significant bits, there will be a sudden jump in the measurement compared to the observer output. The idea is illustrated in Figure 10.

5.1 Filter design

The bit error filter is designed as a standard state-observer that runs concurrent with the process and checks the validity of the Bluetooth packets when

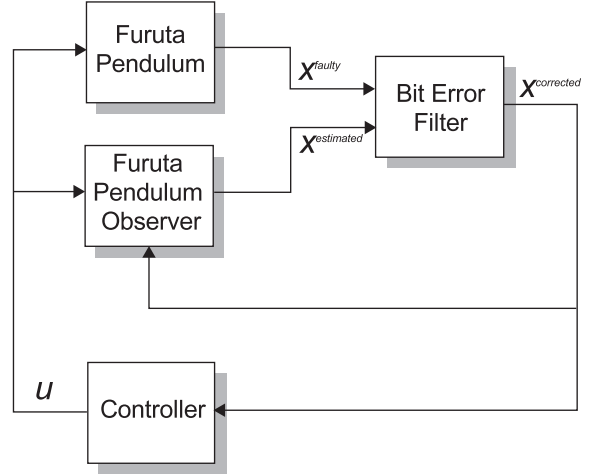


Figure 10 The block diagram for the bit-error filter. The bit-error corrector uses estimates from the observer to detect and correct bit errors from the process.

they arrive at the controller node. The packets contain the 4 sampled states of the system, represented as 4 times 12 bits. The bit error corrector locates a faulty bit by first computing the difference Δ between the sampled state and the estimated state. The difference indicates if and which bit that is incorrect. Assuming that there is a single bit error, the bit position is related to a bit value $2^{n_{bit}}$ as

$$\Delta = |x^{faulty} - x^{estim}| \approx 2^{n_{bit}} \quad (2)$$

The potential faulty bit is thus computed as

$$n_{bit} = \text{round} \left(\frac{\log(\Delta)}{\log(2)} \right) \quad (3)$$

The bit is then flipped if n_{bit} is above some threshold value. Tuning of the threshold values is critical to the performance of the filter. There will always be a difference between the sampled state and the observer state, so we should only attempt to correct the sudden large errors which indicate faulty bits and which are not just modeling uncertainty and measurement noise. In our experiments, the threshold values had to be set at different levels for the different process states, some measurements being noisier than the others.

5.2 Simulation

The bit-error filter was first tried out in simulations. In one particular simulation, a single bit error was injected into 5 % of the packets containing the θ measurement. The filter was then tuned to remove errors in the 8 most significant of the 12 bits. The simulation of the bit-error correction is shown in Figure 11. Without the filter, the control deteriorated completely, as shown in the simulation in Figure 12.

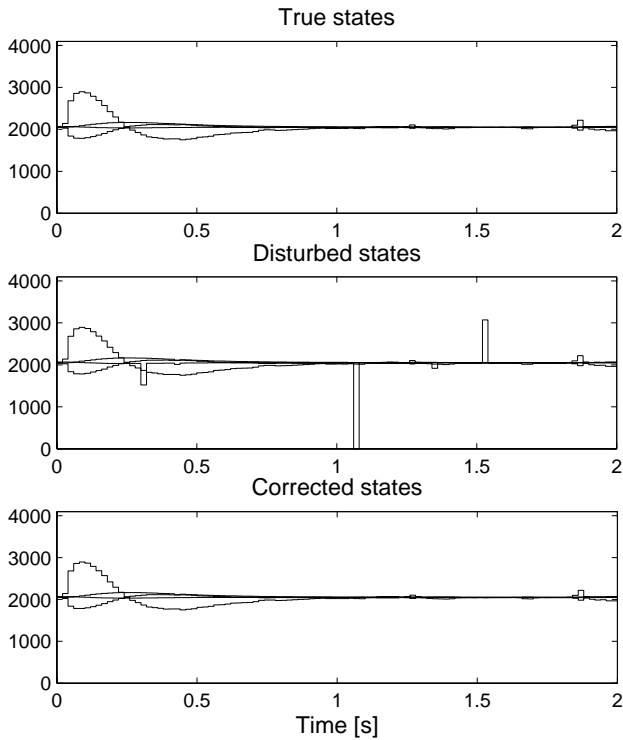


Figure 11 Simulation of the pendulum controller with random bit errors in the θ measurements. The large bit errors are detected and corrected, and the estimated states are very close to the true states.

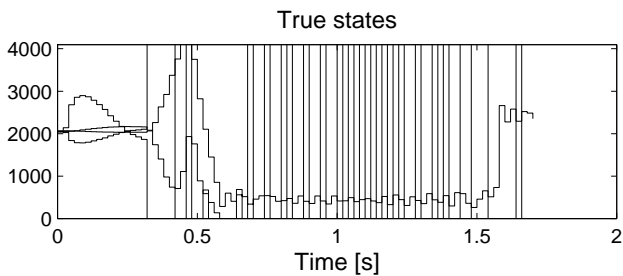


Figure 12 Simulation of the pendulum controller with random bit errors but without the bit-error correction. The performance deteriorates completely.

5.3 Experiment

The bit-error corrector was also tried on the Bluetooth-pendulum setup. Now, the round-trip delay in the loop was constant and equal to 20 ms. A bit error was injected in 5 % of the θ measurement values. The result of an experiment with and without error correction is showed in Figure 13. When the bit-error filter was turned off, the pendulum soon fell down.

6. Conclusion

A distributed wireless control system has been described, where Bluetooth is used to communicate

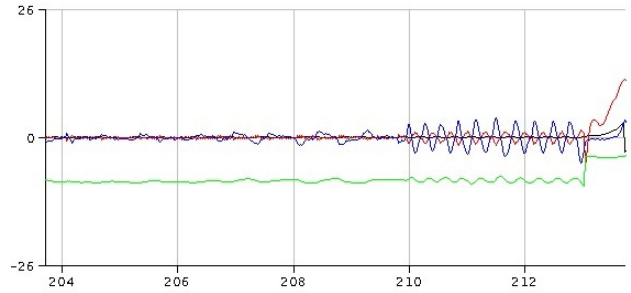


Figure 13 Experiment on the Bluetooth-pendulum setup with random bit errors in the θ measurements. At time 210, the bit-error correction is turned off, and the pendulum soon falls down due to the large erroneous control signals.

between the sensor node, the controller node, and the actuator node. Two specific problems when using Bluetooth have been identified and simple solutions have been presented. An intelligent I/O can be used to compensate for random delays that might occur due to retransmissions. An observer-based bit-error filter might be used to deal with faulty packages. The ideas have been tried out, both in simulations and in real experiments, on a rotating inverted pendulum.

7. References

- Åström, K. J. and B. Wittenmark (1997): *Computer-Controlled Systems*, third edition. Prentice Hall.
- Bluetooth Consortium (2001): “The Bluetooth specification 1.1.” <http://www.bluetooth.com>.
- Hörjel, A. (2001): “Bluetooth in control.” Master Thesis TFRT-5659 ISRN LUTFD2/TFRT-5659-SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.
- Nilsson, J., B. Bernhardsson, and B. Wittenmark (1998): “Stochastic analysis and control of real-time systems with random time delays.” *Automatica*, **34**:1, pp. 57–64.