Hybrid Model Predictive Control
based on
Wireless Sensor Feedback

Erik Henriksson
820505-7451

February 21, 2007


Abstract

This report presents an implementation of a control system of a hybrid system over a network using constrained model predictive control. The systems architecture includes the use of a wireless sensor network for measurements gathering as well as wireless TCP/IP links for communications with the controlled process. The system uses a reference governor approach were the process is stabilized using a local controller at the plant while a computer on the other side of the network-link runs an MPC algorithm which sends optimal set points back to the local controller. In the studied application the MPC optimization problem is solved on-line in real time using CPLEX and the Hybrid Toolbox for MATLAB/Simulink.

Keywords—Constrained model predictive control, reference governor, wireless sensor networks, networked control, hybrid systems, unbounded time-delays
Contents

1 Introduction 5
  1.1 Control over wireless networks 5
  1.2 Problem formulation 5
    1.2.1 The process 6
    1.2.2 The Wireless Sensor Network 6
    1.2.3 The Target 7
    1.2.4 The Host 7
  1.3 Related work 7
  1.4 Thesis outline 8

2 Plant model 10
  2.1 Theoretical model 10
  2.2 Derivation of parameters 11
    2.2.1 Estimation of $\alpha$ and $\beta$ 11
    2.2.2 Estimation of $T_{ss}(p)$ 12
    2.2.3 Practical estimation 12
    2.2.4 Estimation of $\gamma(\cdot)$ 13
  2.3 Sampled model 14
  2.4 Hybrid model 15
    2.4.1 Transforming into MLD model 16

3 Control design 17
  3.1 General idea for the MPC 17
    3.1.1 The Hybrid MPC algorithm 17
  3.2 Choosing reference variables and tuning the MPC 18
    3.2.1 Reference variables 19
    3.2.2 Choosing weights, references and horizon 20
  3.3 The used setup 21
    3.3.1 Expanded plant model 21
    3.3.2 MPC formulation used 21
    3.3.3 Weighting and referencing 22
## 4 Communication network

4.1 Issues to be handled ........................................... 24
4.2 Packet loss model ............................................. 24
4.3 Observers ......................................................... 25
   4.3.1 Luenberger observer ..................................... 25
   4.3.2 Non-linear extended Luenberger observer ............... 26
4.4 Handling packet loss .......................................... 27

## 5 Implementation

5.1 General Structure .............................................. 28
5.2 Wireless Sensor Network block ................................. 28
   5.2.1 General build up ......................................... 29
   5.2.2 Writing applications ...................................... 29
5.3 Host block ....................................................... 31
   5.3.1 Java level .................................................. 32
   5.3.2 MPC level .................................................. 32
5.4 WLAN block ..................................................... 33
5.5 Target block ..................................................... 33
   5.5.1 The local controller ...................................... 33
5.6 The process ...................................................... 34

## 6 Validation

6.1 Simulation of the suggested setup ............................. 35
6.2 Reachable set points .......................................... 35
   6.2.1 Example of non-reachable set point ..................... 38
6.3 Simulations with packet loss introduced ..................... 41
   6.3.1 Behavior with feedback packet loss ..................... 41
   6.3.2 Behavior with forward packet loss ....................... 44
   6.3.3 Behavior with multi directional packet loss ............ 44

## 7 Conclusions

7.1 Wireless Sensor Networks ..................................... 49
7.2 Hybrid MPC ..................................................... 49
List of Figures

1.1 Block diagram of control architecture .................. 6
1.2 Experimental setup at the Automatic Control Laboratory, University of Siena .......................... 7
2.1 Schematics of the process ...................... 10
2.2 Estimates of $\hat{f}(\cdot)$ (solid) and $\bar{f}(\cdot)$ (dashed) .............. 14
2.3 Estimates of $\gamma(\cdot)$ .......................... 15
3.1 Example of reference trajectories for the output variables 23
4.1 Gilbert Model automaton .......................... 25
5.1 The logical levels in the closed loop .................. 28
5.2 The hardware blocks in the closed loop .............. 29
5.3 Block diagram of the Host .......................... 31
5.4 Block diagram of the Target .......................... 34
6.1 Temperature simulation of used setup .............. 36
6.2 Position simulation of used setup .................. 36
6.3 Applied continues input in simulation of used setup ....... 37
6.4 Applied binary input in simulation of used setup ....... 37
6.5 Reachable set points .......................... 38
6.6 Temperature simulation of non-reachable set point ....... 39
6.7 Position simulation of non-reachable set point ....... 39
6.8 Applied continues input in simulation of non-reachable set point ....... 40
6.9 Applied binary input in simulation of non-reachable set point ....... 40
6.10 Nominal temperature step response performance ....... 41
6.11 Nominal position step response performance ....... 42
6.12 Applied continues input in nominal step response performance ....... 42
6.13 Applied binary input in nominal step response performance ....... 43
6.14 Temperature step response with 90% feedback packet loss ....... 43
6.15 Temperature step response with 90% forward packet loss ....... 44
6.16 Position step response with 90% forward packet loss ....... 45
6.17 Continues input in step response with 90% forward packet loss ....... 45
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.18</td>
<td>Binary input in step response with 90% forward packet loss</td>
<td>46</td>
</tr>
<tr>
<td>6.19</td>
<td>Temperature step response with 90% forward and 90% feedback packet loss</td>
<td>47</td>
</tr>
<tr>
<td>6.20</td>
<td>Position step response with 90% forward and 90% feedback packet loss</td>
<td>47</td>
</tr>
<tr>
<td>6.21</td>
<td>Continuous input in step response with 90% forward and 90% feedback packet loss</td>
<td>48</td>
</tr>
<tr>
<td>6.22</td>
<td>Binary input in step response with 90% forward and 90% feedback packet loss</td>
<td>48</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Control over wireless networks

The society of today is moving towards a wireless community where a large number of different mobile embedded systems interact with each other. Examples of this are modern handheld communications devices exchanging information over cellular and wireless local area networks as well as the concept with smart homes where a wireless communication infrastructure allows the large number of systems in a home to interact and exchange information and commands with each other.

Another large field of interest is in the broad spectra of applications in wireless automation where introducing these systems will give several advantages such as reduced cost due to less wiring at system installation and no need for rewiring if the system is upgraded. Other advantages are the increased flexibility in the placement of sensors and actuators and the possibility to move them during operation. This enables embedding intelligence in more devices and to easily exchange information between them. All in all wireless technology makes it possible to use more dynamic and adjustable control architectures.

However, using wireless links in feedback control applications raises new challenges on how to deal with network uncertainties such as packets lost during communication and variable time delays. The research in control over wireless networks is a young area without mature theory or tools, but with lots of current activity.

1.2 Problem formulation

The problem at hand is to control a miniature industrial heating process over a network. The forward communication to the process is done via wireless TCP/IP LAN (WLAN) and the measurements from the process are transferred via a wireless sensor network.
The general idea to solve this problem is to use the reference governor approach described in [1] were stability of the process is granted by a local controller at the plant which in turn gets its reference from an MPC algorithm on the other side of the communication network. This reduces the need of having large computational power at the plant and also enables having one central powerful computer running several reference governing MPCs at the same time. The proposed control loop architecture can be seen in Figure 1.1.

As mentioned in section 1.1 using wireless links causes new problems in control applications, therefore the work has focused on how to overcome these problems as well as others that have been raised during the work to control the process.

1.2.1 The process

The process, as can be seen in Figure 1.2, consists of a belt which via an actuator can be moved backwards or forwards. Over the belt four heating lamps are placed in a row. These lamps can be pairwise turned on and off, that is, lamps 1 and 3 are controlled by one switch and lamps 2 and 4 by another switch. These switches can be operated independently of each other.

1.2.2 The Wireless Sensor Network

To simulate the working pieces and to take temperature measurements the Berkeley type motes Tmote sky from Moteiv corporation [2] are used. These motes come equipped with sensors, microprocessor and radio communication device [9]. For communication with a regular PC these motes are also equipped with a USB interface.

The motes used for the measurements gathering are organized in a Wireless Sensor Network. This network is set up and organized by the motes in the net. From this net the data acquired by the motes can be read out with the help of a PC.
1.2.3 The Target

The local controller runs on a PC called the Target. This PC is placed at the process and provides the interface between the software and the plant using a DAQ board.

On the other side the local controller is interfaced with wireless TCP/IP ethernet. Via this ethernet connection it can be controlled by the Host PC.

1.2.4 The Host

The reference governing MPC is implemented in a PC called the Host. It is at this computer that the optimal control actions are calculated and sent to the Target, which in turn makes sure that they are actuated. It is at the Host that the data from the Wireless Sensor Network is read out and processed. It is also here that actions are taken to overcome lost data packets and to estimate unmeasurable states in the system.

1.3 Related work

In the area of predictive control over wireless networks [14] proposes a new dynamic predictive control scheme with a Kalman filter and CUMSUM based delay estimator. The delay estimate is here used in the predictive control law to explicitly suppresses the network induce delays. An alternative algorithm is presented in [15] using a gain scheduling algorithm based on LQR output feedback.
In [17] an idea to use the command sequences computed by predictive controllers to overcome packet loss and large delays is presented along with a synchronization algorithm.

A more specific example in the area of control over networks can be found in [16] which proposes a structural building monitoring system were motes are used for gathering of feedback data, calculation of control forces and issuing the actuator commands. It also presents an implementation of the system on a half-scale structure model.

As an introduction to MPC [1] gives an overview, historic aspects as well as some motivating examples. It discusses unconstrained and constrained optimal control together with feasibility, stability and convergence results. More in this area can be found in [18]. For the expansion into Hybrid MPC [7] gives an introduction to different classes of hybrid models as well as an explanation of Hybrid MPC. More on Hybrid MPC can be found in [19].

For persons interested in Mote programming [20] and [21] are recommended. Here the basics of programming motes is explained. Further reading in synchronization of wireless sensor networks can be found in [22],[23] and [24].

1.4 Thesis outline

The thesis is divided into seven chapters, each of which is described below.

Chapter 1, Introduction

Gives a motivation for the subject and comments some of the related work on the area. It also gives the formulation of the problem and presents the idea on how to solve it. Further it presents the hardware and the structure of the setup.

Chapter 2, Plant model

Here a theoretical model is derived. Methods for identification of parameters is presented as well as the parameters themselves. A sampled model is derived along with two different hybrid representations of the system.

Chapter 3, Control design

This chapter presents the MPC formulation used to control the system. The choice of reference variables are motivated and discussions are made around how to choose reference trajectories and choosing weights.
Chapter 4, Communication network
Here the models of packet loss in the wireless link is presented along with the observer used to estimate states and overcome missing data.

Chapter 5, Implementation
The implementational aspects of the problem are presented. The problem is divided into a number of different logical levels and each level is presented and discussed.

Chapter 6, Validation
This section shows simulations of the closed-loop with different rates of packet loss both in the forward and feedback link.

Chapter 7, Conclusions
Conclusions drawn from the work are presented and discussed.
Chapter 2

Plant model

2.1 Theoretical model

The schematics of the process with inputs and states is shown in Figure 2.1. From this the theoretical model has been derived, this is done in continues time for simplicity. Some basic assumptions have been made regarding the thermodynamical behavior of the lamps and the motes. The lamps are said to have no dynamics and the motes are said to be second order. This behavior has been validated through experiments.

First let us derive a simple model of the mote temperature with only the belt velocity as input variable. One then ends up with something like.

\[ \dot{T}_1 = -\alpha (T_1 - T_{ss}(p)) \]  
\[ \dot{T}_2 = -\beta (T_2 - T_1) \]  
\[ \dot{p} = v \]

\[ v, p, T_1, T_2 \in \mathbb{R} \]

were \( T_{ss}(p) \) is the steady-state temperature of the mote when in position \( p \).

Figure 2.1: Schematics of the process
CHAPTER 2. PLANT MODEL

If the possibility to pair-wise turn on and off the lamps is introduced, as described in section 1.2.1, one get two new binary input signals \( \{ u_1, u_2 \} \) representing the state of the lamp switches. The value of these variables clearly affects \( T_{ss}(p) \) and the general description can be seen in Equation 2.2.

\[
T_{ss}(p, u_1, u_2) = f_1(p)u_1 + f_2(p)u_2 + T_{amb}
\] (2.2)

Were \( T_{amb} \) is the ambient temperature which now has to be taken into account since all lamps can be turned off at the same time. \( f_1(p) \) and \( f_2(p) \) are the increase in steady-state temperature from lamp pair 1 resp. lamp pair 2 when in position \( p \) with the lamps turned on. By the nature of the process one can conclude that \( f_1(p), f_2(p) \) are static nonlinearities and that \( f_1(p) = f_2(p + l) \) were \( l = 0.15m \).

To get a more convenient problem a piece-wise affine approximation \( \chi(p) \) of the lamps influence is derived. This is done by partitioning \( p \in \mathbb{R} \) into \( \ell \) intervals \( \{ I_1, I_2, \ldots, I_\ell \} \). It is then said that \( f_1, f_2 \) are piece-wise affine in these intervals. That is,

\[
\chi_i(p) = \begin{cases} 
K_j p^i + h_j & \text{if } u_i = 1, \ p \in I_j \\
0 & \text{otherwise}
\end{cases}
\] (2.3a)

\[
\chi(p) = \chi_1(p) + \chi_2(p).
\] (2.3b)

Summing up the above reasoning and taking some hardware limitations into account the final model becomes.

\[
\dot{T}_1 = -\alpha(T_1 - \chi(p) - T_{amb})
\] (2.4a)

\[
\dot{T}_2 = -\beta(T_2 - T_1)
\] (2.4b)

\[
\dot{p} = \gamma(u_c)
\] (2.4c)

\[
u_c \in \mathbb{R}
\] (2.4d)

were \( \gamma(\cdot) \) is the static non-linear mapping between the process input \( u_c \) and the belt speed \( v \).

2.2 Derivation of parameters

2.2.1 Estimation of \( \alpha \) and \( \beta \)

To estimate \( \alpha \) and \( \beta \) one can do step response trials and try to fit a particular solution of a second order system to these data. This is done by the non-linear optimization problem described in Equation 2.5 were \( y \) is measured output, i.e. \( T_2 \) as in Equation 2.4b.
CHAPTER 2. PLANT MODEL

\[ \min_{c_i, \alpha, \beta \in \mathbb{R}} \parallel y - c_0 - c_1 e^{-\alpha t} - c_2 e^{-\beta t} \parallel_2^2 \]  \hspace{1cm} (2.5)

It is noticeable that only part of the step response is needed for the parameter estimation. This can be very useful, especially for slow systems.

2.2.2 Estimation of \( T_{ss}(p) \)

Once \( \alpha \) and \( \beta \) are known \( T_{ss}(p) \) can be estimated with step response trials at different positions \( p^* \). This is simply done by solving Equation 2.5 for each \( p^* \) using the estimated \( \hat{\alpha} \) and \( \hat{\beta} \). Using the estimated time constants now makes this a linear optimization problem which is easily solved.

By inspection on Equation 2.5 and Equation 2.4 one can observe the following. Given that the system is in steady state with the lamps turned off when a step is applied the following holds.

\[ T_{amb}(p^*) = \sum_{i=0}^{2} c_i \]  \hspace{1cm} (2.6)

\[ T_{ss}(p^*) = c_0 \]  \hspace{1cm} (2.7)

Looking at Equation 2.2 it can be seen that by applying the step with one of the inputs \( u_i \) at a time, one can, using the above equations, derive an estimate of the corresponding \( f_i \) according to Equation 2.8.

\[ \hat{f}_i(p^*) = T_{ss}(p^*) - T_{amb}(p^*) = -c_1 - c_2 \]  \hspace{1cm} (2.8)

When both \( \hat{f}_1 \) and \( \hat{f}_2 \) have been identified the relation between them as described in section 2.1 is utilized to shift \( \hat{f}_2 \) and form the mean of these two estimates, see Equation 2.9. This is done in order to reduce the variance of the estimates. From this joint estimate \( \hat{f}_1 \) and \( \hat{f}_2 \) are defined as in Equations 2.10

\[ \hat{f}(p) = \frac{\hat{f}_1(p) + \hat{f}_2(p + l)}{2} \hspace{1cm} l = 0.15m \]  \hspace{1cm} (2.9)

\[ \hat{f}_1(p) \triangleq \hat{f}(p) \]  \hspace{1cm} (2.10a)

\[ \hat{f}_2(p) \triangleq \hat{f}(p - l) \hspace{1cm} l = 0.15m \]  \hspace{1cm} (2.10b)

2.2.3 Practical estimation

Estimation of \( \alpha \) and \( \beta \)

\( \alpha \) and \( \beta \) are estimated using as series of step response trials. For each of these the optimization problem in Equation 2.5 is solved, results showed in Table 2.1.
Statistical analysis on the measurement data in Table 2.1 gives that $\hat{\alpha} = 85.5084 \times 10^{-3} \pm 19.8914 \times 10^{-3}$ (95%) and $\hat{\beta} = 6.0811 \times 10^{-3} \pm 2.3500 \times 10^{-3}$ (95%).

**Estimation of $\hat{f}_i$**

The a mote is placed at different locations $p^*$ and a step is applied by turning on one pair of lamps. Part of the step response is recorded and $\hat{f}_i(p^*)$ is then estimated according to the method in section 2.2.2. This is done repeatedly until enough measurements have been gathered to enable estimates of $\hat{f}_1(p)$ and $\hat{f}_2(p)$. Now using Equation 2.9 $\bar{f}$ is formed, showed in Figure 2.2. Also showed in Figure 2.2 is the piece-wise affine estimation of this curve, $\bar{f}$, which transformed into $\bar{f}_1(p)$ and $\bar{f}_2(p)$ in the same way as in Equations 2.10 and then used to compute the approximation $\chi(p)$ in Equation 2.3.

**2.2.4 Estimation of $\gamma(\cdot)$**

Since the system does not have any means to measure the velocity or the position of the belt during execution the only mean available to compute the position of the mote is to integrate $\gamma(u_c)$. Since this procedure is very sensitive to model errors the estimation of $\gamma(\cdot)$ needs to be of high accuracy. The method chosen for the estimation is the following:

1. Apply a constant input $u_c^*$ during the time interval $\delta t^*$.
2. Measure the distance $\delta p^*$ the mote has moved.
3. Calculate $\gamma(u_c^*) = \delta p^* / \delta t^* = v^*$
4. Redo from (1) until a sufficient number of data point have been collected.
5. Calculate the point-wise gain vector representing $\gamma(\cdot)$ using cubic splines to interpolate between data points.

<table>
<thead>
<tr>
<th>$T_{step}$</th>
<th>$\alpha^*$</th>
<th>$\beta^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.5 °C</td>
<td>81.6268 $10^{-3}$</td>
<td>4.4942 $10^{-3}$</td>
</tr>
<tr>
<td>-16.2 °C</td>
<td>70.8172 $10^{-3}$</td>
<td>5.4067 $10^{-3}$</td>
</tr>
<tr>
<td>19.9 °C</td>
<td>96.6727 $10^{-3}$</td>
<td>6.9373 $10^{-3}$</td>
</tr>
<tr>
<td>-14.2 °C</td>
<td>81.3030 $10^{-3}$</td>
<td>5.8640 $10^{-3}$</td>
</tr>
<tr>
<td>20.9 °C</td>
<td>85.9467 $10^{-3}$</td>
<td>5.8134 $10^{-3}$</td>
</tr>
<tr>
<td>-15.5 °C</td>
<td>81.8245 $10^{-3}$</td>
<td>5.7454 $10^{-3}$</td>
</tr>
<tr>
<td>16.8 °C</td>
<td>103.5111 $10^{-3}$</td>
<td>8.5445 $10^{-3}$</td>
</tr>
<tr>
<td>-13.5 °C</td>
<td>82.3653 $10^{-3}$</td>
<td>5.8434 $10^{-3}$</td>
</tr>
</tbody>
</table>

Table 2.1: $\alpha$ and $\beta$ estimation data points.
Figure 2.2: Estimates of $\hat{f}(p)$ (solid) and $\bar{f}(p)$ (dashed)

The resulting estimate of $\gamma(\cdot)$ is shown in Figure 2.3.

### 2.3 Sampled model

The above reasoning and derivations have all been done in continuous time. However since the system is computer implemented this model must be sampled. Using sampling and zero-order hold one get a model as in Equation 2.11.

From this model the reader should notice two things. First that it is said that the state $x_3$ is considered as an output, even though it is not measurable. Secondly that the the input to the belt is said to be the true speed $v$. This choice of input is motivated by the fact that the positioning part of the model now becomes linear and that the system instead can be actuated by $u_c = \gamma^{-1}(v)$. 
CHAPTER 2. PLANT MODEL

Figure 2.3: Estimates of $\gamma(\cdot)$.

$$x(t+1) = \begin{pmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix} x(t) + \begin{pmatrix} b_{11} & 0 \\ b_{21} & 0 \\ 0 & b_{32} \end{pmatrix} \begin{pmatrix} \lambda(t) \\ v(t) \end{pmatrix}$$

(2.11a)

$$\lambda(t) = f_1(p)u_1 + f_2(p)u_2 + T_{amb}$$

(2.11b)

$$y(t) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} x(t)$$

(2.11c)

where $x = (T_1, T_2, p)^T$

2.4 Hybrid model

The above reasoning in this chapter has so far given a reasonable model for the process which is to be controlled. However the model in Equation 2.11 is still non-linear in $\lambda(t)$. To overcome this the Equations 2.3 are used to form $\hat{\lambda}(t) = \chi(p) + T_{amb}$. This estimate transforms the model into a set of affine models which together form a “piece-wise affine system” or PWA hybrid model of the system as in Equation 2.12.
\[ x(t + 1) = \begin{pmatrix} a_{11} & 0 & b_{11}K \\ a_{21} & a_{22} & b_{21}K \\ 0 & 0 & 1 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 0 \\ b_{32} \end{pmatrix} v(t) + \begin{pmatrix} H \\ H \end{pmatrix} \] (2.12a)

\[ y(t) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} x(t) \] (2.12b)

where \( K \) and \( H \) are determined by position and logical inputs. That is in interval \( I_j \), using same notation as in Equations 2.3, the truth table in Table 2.2 applies.

<table>
<thead>
<tr>
<th>( K )</th>
<th>( H )</th>
<th>( u_1 )</th>
<th>( u_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( T_{amb} )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( k_j^1 )</td>
<td>( h_j^2 + T_{amb} )</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( k_j^2 )</td>
<td>( h_j^1 + T_{amb} )</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( k_j^1 + k_j^2 )</td>
<td>( h_j^1 + h_j^2 + T_{amb} )</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2.2: Truth table of PWA hybrid dynamics.

### 2.4.1 Transforming into MLD model

The model described by Equation 2.12 and by the truth table in Table 2.2 can be transformed into an equivalent model of another class of hybrid models called MLD or 'Mixed Logical Dynamical' systems. The equivalency of these classes is presented in [10]. The transformation of a PWA system into MLD is on the form as presented in Equation 2.13. The reason for a transformation like this will be shown in chapter 3.

\[ x(t + 1) = Ax(t) + B_1 u(t) + B_1 \delta(t) + B_3 z(t) \] (2.13a)

\[ y(t) = Cx(t) \] (2.13b)

\[ E_2 \delta(t) + E_3 z(k) \leq E_1 u(k) + E_4 x(k) + E_5 \] (2.13c)

were \( u = (v, u_1, u_1)^T \) is the input vector and \( z(t) \in \mathbb{R} \) and \( \delta(t) \in \{0, 1\} \) are auxiliary variables used to describe the system. The matrices introduced are real valued and of proper dimension.
Chapter 3

Control design

3.1 General idea for the MPC

The overall idea for the control strategy is to use an MPC algorithm as a reference governor as described by [1]. Since the system is non-linear it is as mentioned in section 2.4 modeled as a set of affine models. This set is then transformed into an MLD model, as described in 2.4.1, for which there is a general MPC algorithm implemented in [6]. The idea is to use a this algorithm, described in section 3.1.1, with a moving horizon and solve the optimization problem on-line in real time.

3.1.1 The Hybrid MPC algorithm

The algorithm used is the one presented in [7] and implemented in [6]. The author wish to point out that the content in the current section is taken from [7].

Assume that the output $y(t)$ is to be tracked and that there are also desired references on input and auxiliary variables. Let $t$ be the current time and $x(t)$ is the current state. Consider the following optimal control problem
3.2 Choosing reference variables and tuning the MPC

Before choosing reference variables, set points to these variables and tune the controller one must first reflect over the control objective. The objective is, as in this application, often expressed in 'fuzzy' terms as in the list below. From this list it can be decided which variables that needs to be referenced in order to fulfil these objectives. This is done in section 3.2.1.
1. The piece should start at the beginning of the belt and end up at the end within a reasonable time.

2. The piece should be heated according to desired time vs temperature curve.

3. The movement of the piece should be ‘smooth’ to reduce wear on the process.

4. The piece should be moved as little as possible to reduce the use of the belt.

5. The temperature of the piece should stay within a ‘safe’ temperature range to prevent damage.

6. The piece should stay on the belt and not fall off.

3.2.1 Reference variables

Looking at the objectives in the above list one can separate them into three types of objectives. Objective 1 and 2 are objectives with respect to output, objective 3 and 4 are objectives with respect to inputs and finally objective 5 and 6 are objectives with respect to bounds on states.

In order to ensure that the piece starts at the beginning of the belt and ends up at the end of the belt the position must be referenced, otherwise there is no way to force the system to move in this way. The same holds when it comes to follow a desired temperature curve, if the temperature is not referenced it can not be controlled.

The two objectives related to input is of a more soft nature. The word ‘smooth’ is interpreted as to have a low acceleration. This in turn demands that the acceleration is in some way calculated and also referenced. This can be done by introducing another state, \( x_v \), into the system which remembers the previous continues input \( v \) and then introduce the auxiliary variable \( z_a \) to compute the acceleration. The dynamics of \( x_v \) and \( z_a \) are given by

\[
\begin{align*}
x_v(t + 1) &= v(t) \quad (3.3a) \\
z_a(t) &= \frac{v(t) - x_v(t)}{T_s} \quad (3.3b)
\end{align*}
\]

Objective 4 can be interpreted as to keep the input to the belt down and therefore also the value of the input must be referenced.

The two last objectives are different from the others in the sense that they only specify that the system at all times should be within a certain partition of the state-space. These objectives can be fulfilled by imposing upper and lower bounds on the states. Hence there is no need for reference variables since this can be implemented in the MPC constraints, see Equation 3.1b.
3.2.2 Choosing weights, references and horizon

Given which variables to be referenced one must now find suitable references and bounds as well as deciding the relative importance of these references by choosing their weights. In extension to this the control horizon $N$ must also be decided.

Value of the references

For the two input reference variables presented in section 3.2.1 the natural choice of reference is 0, which means that both the speed of the belt and the acceleration should be 0. This will guarantee both a 'smooth' behavior and a low usage of the belt.

When it comes to choosing appropriate references for the two outputs there is no natural choice since these are dependent on the desired performance of the plant. One could here imagine a large set of different trajectories that would be desirable depending on the purpose of the heating process.

Weighting

The choice of the weights is the key to fulfilling the control objectives. Since it may not be possible to fulfill all objectives at the same time these weights represent the relative trade-off between objectives, were a higher weight represents greater importance. With respect to these weights the MPC algorithm will calculate the input sequence that will give the 'optimal' trade-off between the objectives with respect to the weights.

Looking at the control objectives one can draw the conclusion that the most important objective is to make sure that the piece is heated according to the reference value. Almost as important is that the piece follow the position reference so that it will move according to the planned path. A lot less important then these two are the objectives of keeping the movement smooth and to move the belt as little as possible. This means that these two weights are several orders smaller than the weights on position and temperature.

Choosing horizon

When calculating the optimal input sequence the MPC does this with the given horizon $N$, that is, it computes the input trajectory that is optimal when looking $N$ sampling instants into the future. The length of this horizon clearly effects the performance of the controller since a longer horizon gives more degrees of freedom, hence the length of the horizon is an important tuning variable.
CHAPTER 3. CONTROL DESIGN

In general it can be said that a longer horizon gives a smoother behavior and a shorter a more aggressive controller. It is therefore advantageous to have a long horizon since it gives a better performance, a long horizon on the other hand expands the optimization problem and makes it more difficult to solve. The rule of thumb is to choose \( N \) as small as possible [7].

3.3 The used setup

This section presents the structures of the MPC algorithm used. This includes the use of an expanded model of the plant as well as examples of suitable weighs and some different reference trajectories.

3.3.1 Expanded plant model

Looking at Equation 2.12 and Table 2.2 one can see that the ambient temperature \( T_{amb} \) is modeled as a constant ‘input’ to the system. However it is an ‘input’ that can not be controlled. To overcome this when constructing the controller a fourth state is added which represents the ambient temperature. This state holds its initial condition and can only be changed by using reset maps.

When discussing control objectives in section 3.2.1 it was concluded that another state was needed in order to reference acceleration. This state is introduced as a fifth state holding the last continues input.

The new expanded system with five states is presented in Equation 3.4 and truth table in Table 3.1. It is the MLD representation of this system that the MPC controller is synthesized from.

\[
x(t + 1) = \begin{pmatrix}
a_{11} & 0 & b_{11}K & 1 & 0 \\
a_{21} & a_{22} & b_{21}K & 1 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{pmatrix} x(t) + \begin{pmatrix}
0 \\
0 \\
b_{32} \\
0 \\
0 \\
\end{pmatrix} v(t) + \begin{pmatrix}
H \\
H \\
0 \\
0 \\
0 \\
\end{pmatrix}
\]  

(3.4a)

\[
y(t) = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
\end{pmatrix} x(t)
\]

(3.4b)

where \( x = (T_1, T_2, p, T_{amb}, x_v)^T \)

3.3.2 MPC formulation used

From the new model of the system presented in section 3.3.1 the MPC formulation used is formulated. The objective function for this is presented in Equation 3.5a. This objective function is simplified by including the
static references on speed and acceleration directly into the function. The constraint set over which this function is to be minimized is presented in Equations 3.5b.

\[
\min_{\{u, \delta, z\}} J(\{u, \delta, z\}_0^{N-1}, x(t)) \triangleq q_\rho \rho^2 + \\
+ \sum_{k=0}^{N-1} q_v v(k)^2 + \frac{q_z}{T_s^2} (x_v(k|t) - v(k))^2 + \|Q_y(y(k|t) - y_v)\|_2 
\]

(3.5a)

\[
\begin{align*}
\text{s.t.} & \quad x(0|t) = x(t) \\
& \quad x(k+1|t) = Ax(k|t) + Bu(k) + B_2 \delta(k|t) + B_3 z(k|t) \\
& \quad y(k|t) = Cx(k|t) + Du(k) + D_2 \delta(k|t) + D_3 z(k|t) \\
& \quad E_2 \delta(k|t) + E_3 z(k|t) \leq E_1 u(k) + E_4 x(k|t) + E_5 \\
& \quad u_{\min} - 1 u_\rho \leq u(t + k) \leq u_{\max} + 1 u_\rho, \quad k = 0, 1, ..., N - 1 \\
& \quad x_{\min} - 1 x_\rho \leq x(t + k|t) \leq x_{\max} + 1 x_\rho, \quad k = 0, ..., N \\
& \quad y_{\min} - 1 y_\rho \leq y(t + k) \leq y_{\max} + 1 y_\rho, \quad k = 0, ..., N - 1 
\end{align*}
\]

(3.5b)

### 3.3.3 Weighting and referencing

After implementing the MPC algorithm described in section 3.3.2 a set of suitable weights, variable constraints and prediction horizon was decided via simulations of the system. One of the suitable sets found is presented below with weights and horizon in Equations 3.6 and the active constraints in Equations 3.7. Figure 3.1 shows an example of output reference trajectories.

\[
\begin{align*}
N & = 4 \\
q_\rho & = 10^3 \\
q_v & = 0.0076 \\
q_z & = 0.00076 \\
Q_y & = \begin{pmatrix} 0.875 & 0 \\ 0 & 0.01 \end{pmatrix}
\end{align*}
\]

(3.6a - 3.6c)

<table>
<thead>
<tr>
<th>K</th>
<th>H</th>
<th>(u_1)</th>
<th>(u_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(k_1^2)</td>
<td>(h_1^2)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(k_1^4)</td>
<td>(h_1^4)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(k_1^4 + k_2^4)</td>
<td>(h_1^4 + h_2^4)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.1: Truth table of extended hybrid dynamics.
Figure 3.1: Example of reference trajectories for the output variables

\[
\begin{pmatrix}
15 \\
15 \\
0 \\
15 \\
-\infty
\end{pmatrix} \leq \begin{pmatrix} x(t + k) \end{pmatrix} \leq \begin{pmatrix}
80 \\
80 \\
1.2 \\
35 \\
\infty
\end{pmatrix} \quad (3.7a)
\]

\[-0.1145 \leq v(t + k) \leq 0.1145 \quad (3.7b)\]

\[u_1(t + k), \ u_2(t + k) \in \{0, 1\} \quad (3.7c)\]
Chapter 4

Communication network

4.1 Issues to be handled

When dealing with wireless links one faces a multiple of problems such as unbounded varying time delays, loss of data packets, channel distortion etc. All of these problems must be overcome in order to successfully enable using wireless links in control applications.

4.2 Packet loss model

In order to simulate packet loss and thereby study the effectiveness of the suggested approach a model for the packet drops has to be used.

Two different models for packet loss has been used both of which are based on Markov chains. The models used is the Bernoulli model which is 0th-order Markov chain and the Gilbert model which is a 1st-order Markov chain, both of which are described in [11]. The Bernoulli model states that the state of the communication link at this instant is independent from previous instants. That is the probability of loosing the next package is independent if the current packet is lost or not. Essentially the model is described by a binary stochastic variable $X \in \{\text{received, lost}\}$ were $P(\text{lost}) = p$ and $P(\text{received}) = 1 - p$ were $p$ is the overall packet drop rate.

A more advanced model is the Gilbert model which states that the probability of loosing the next package is dependent on if the current packet was received or not. The model is best illustrated by the automaton in Figure 4.1 were $p_{rr} = 1 - p_{rl}$ is the probability to receive the next packet if the current packet was received and $p_{ll} = 1 - p_{lr}$ is the probability to lose the next package if the current package was lost. A formula to compute the steady-state or mean packet loss rate $p_L$ for the Gilbert model is given in Equation 4.1 taken from [11].

$$ p_l = \frac{p_{rl}}{p_{rr} + p_{rl}} \quad (4.1) $$
The numerical values of the probabilities introduced in this section are all related to the link quality of the network. Tuning these values enables simulation of a multiple of network conditions.

4.3 Observers

As most often is the case, not all of the states in the process are measurable, hence an observer is needed. Two different approaches have been tested. When designing the linear observers the state-space model in Equation 2.11 has been used.

4.3.1 Luenberger observer

The simplest observer to use is a reduced order observer as described in [4]. This observer is described by Equation 4.2 were \( y(t + 1) \) is the current measurement. The reconstruction error of the states also fulfills Equation 4.3.

\[
\begin{align*}
\hat{x}(t + 1 \mid t + 1) &= \Phi \hat{x}(t \mid t) + \Gamma u(t) + \\
&\qquad + K \left[ y(t + 1) - C(\Phi \hat{x}(t \mid t) + \Gamma u(t)) \right] \quad (4.2) \\
\tilde{x}(t + 1 \mid t + 1) &= (I - KC)\Phi \tilde{x}(t \mid t) \quad (4.3)
\end{align*}
\]

By looking at the output estimation error as in [4] one can form the expression in Equation 4.4 from which one can see that if \( (I - CK) = 0 \) there is no error in the output estimation.

\[
\begin{align*}
y(t + 1) - C\hat{x}(t + 1 \mid t + 1) &= C\tilde{x}(t + 1 \mid t + 1) = \\
&= C(I - KC)\Phi \tilde{x}(t \mid t) = (I - CK)C\Phi \tilde{x}(t \mid t) \quad (4.4)
\end{align*}
\]
CHAPTER 4. COMMUNICATION NETWORK

Forming \((I - CK)\) with the system described in Equation 2.11 one get.

\[
\begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix} - \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
k_{11} & k_{12} \\
k_{21} & k_{22} \\
k_{31} & k_{32}
\end{pmatrix} = \begin{pmatrix}
1 - k_{21} & -k_{22} \\
-k_{31} & 1 - k_{32}
\end{pmatrix} \equiv \begin{pmatrix}
0 & 0 \\
0 & 0
\end{pmatrix}
\]

\[
\Rightarrow K = \begin{pmatrix}
k_{11} & k_{12} \\
1 & 0 \\
0 & 1
\end{pmatrix}
\]

Again using the same system, now forming the prediction error system matrix of Equation 4.3 one get.

\[
(I - KC)\Phi = \begin{pmatrix}
a_{11} - k_{11}a_{21} & -k_{11}a_{22} & -k_{12} \\
0 & 0 & 0 \\
0 & 0 & 0
\end{pmatrix}
\]

Thus the state estimation error is converging to zero given that \(|a_{11} - k_{11}a_{21}| < 1\), arbitrarily the choice of \(k_{12}\).

Analysis of results

The above reasoning might lead to believe that the states can be estimated with no error, however this is not the case. Looking at Equation 4.2 and the system in Equation 2.11 one can see that he use of the above Luenberger observer requires both measuring the position and knowing the value of \(z(t)\). Since neither one of these can be measured they have to be open-loop estimated. The position must be estimated by integrating \(\gamma(\cdot)\) described in 2.2.4 and \(z(t)\) is estimated by inserting the position estimate in Equation 2.2.

4.3.2 Non-linear extended Luenberger observer

In order to get better performance from the above presented observer it was extended into a non-linear version. The estimation of \(\lambda(t)\) is done by using \(\dot{\lambda}(t)\). This new observer is described in Equations 4.5.

\[
\dot{x}(t+1|t+1) = \Phi \dot{x}(t|t) + \xi(t) + K[y(k+1) - C(\Phi \dot{x}(t|t) + \xi(t))]
\]

\[
\xi(t) = \begin{pmatrix}
b_{11}(\chi(p(t|t)) + T_{amb}) \\
b_{21}(\chi(p(t|t)) + T_{amb}) \\
T_s v
\end{pmatrix}
\]

were \(T_s\) is the sampling time.
4.4 Handling packet loss

When packets containing measurements are lost the best estimation method found is to let the observer evolve in open loop. This is intuitively correct since the best guess is that the system will evolve according to the model.

When packets are lost in the forward link the approach used is to zero-order hold the last known input at the Target. This is based on the assumption that the last known reference is the best guess of what would have been received if the current packet had not been lost.
Chapter 5

Implementation

5.1 General Structure

The general structure of the problem can be divided into 11 main levels were problems occur inside the levels but also in the boundaries between them. These levels and their relations to each other are depicted in Figure 5.1. These main levels can then in turn be grouped into 5 larger independent blocks, as described in chapter 1 based on which hardware platform they are implemented. Figure 5.2 show these hardware layers as grey dashed boxes enclosing the control loop blocks, shown in Figure 1.1, which are implemented in them.

The entire control loop is run on an 8Hz sampling frequency with these five separate hardware platforms synchronized.

5.2 Wireless Sensor Network block

The Wireless Sensor Network or WSN block consists of the three levels measurement point, wireless link and base station. It is in the link level

![Diagram](image)

Figure 5.1: The logical levels in the closed loop
and at the base station that most of the problems handled by the rest of the control loop occur. In the link there is shadowing, fading and other phenomena resulting in that information sent from the measurement point does not reach the base station, i.e. packets are lost. At the base station the problems occur if there are too many measurement points transmitting at the same time which causes the base station to congest, that is not being able to receive all packets sent to it.

5.2.1 General build up

The WSN block is built up by one or several motes acting as the measurement point which are transmitting their measurements via radio to a mote acting as the base station, which in turn is connected to the Host PC via USB. The base station runs a generic TinyOS application which listens to all packets sent on its monitoring frequency and group id. The measurement points run a specially written measurement application. The development of this application is described in section 5.2.2.

5.2.2 Writing applications

The motes run on the TinyOS [12] real time operating system and the applications are written in the programming language nesC [21]. To compile the software and download it onto the motes the software Cygwin [13] has been used.

Writing applications in nesC has proven to be hard since it is hardware close programming. Instead code refactoring of the numerous applications distributed with TinyOS is used. Still a substantial amount of time has been spent on different approaches before a well functioning application could be written. Below the most developed applications are described.

Delta application

The Moteiv [2] distribution of TinyOS contains an application called Delta. The general purpose of this application is to provide data to a Java appli-
cation which visualizes the sensor network architecture and displays some sensor data. Among the data that is read out is the temperature, which in the current experimental setup is the interesting measurement.

The reason for using the Delta application was that it is compatible with the Cygwin compiler command ‘lowpower’. This command appends code to the application compiled which enables synchronization in the wireless network. This would be desirable since all the motes in the network then will have the same global time. Thus making it possible to maximize network sleep time and have global time stamping of data.

After code modifications in the Delta application, were uninteresting measurements were taken out and sequence numbering and time stamping of packets were added, the application was compiled with the ‘lowpower’ option and the code uploaded on a number of motes.

When running this system on slow sampling frequency, i.e. 0.5 Hz, the system worked well and synchronization was achieved. When increasing the sampling frequency to 3 Hz the network performance deteriorated rapidly with large packet loss as a result. This is believed to have been caused by too low accuracy in the time synchronization. That is the motes ’woke up’ at the agreed global time value, however the value of global time was not the same at all motes. This lead to that motes broadcasted when no other mote were there to listen. Hence, packets were lost.

Since the only benefit of using the Delta application was its compatibility with the ‘lowpower’ compiler option, which had proven to have too low accuracy, it was decided that the development of the Delta application was to be abandoned and that development of a general TinyOS distribution application should be started, see section 5.2.2.

**SurgeTelos application**

In the general distribution of TinyOS there is an application called SurgeTelos. It is similar to Delta but is written a bit differently so that it can utilize some more of the general built in functionality in nesC and TinyOS. This made it a well suited platform for writing an application which would have the desired performance.

Firstly SurgeTelos was stripped from unnecessary sensor readings and other parts of code which would not be used. Then time stamping and sequence numbering was introduced. After this the data packet structure was refactored so that it only contained the necessary information. This resulted in an application which could run at 3Hz sampling frequency with no packet loss. This was a great performance increase in comparison with the Delta application. The tradeoff however that the time stamping of packets now only was in local time.
Mote ID
Temperature
Position
measurements from
motes and target
VU1U2
Set target

Observer
1
Mote to
read

Hybrid MPC

Figure 5.3: Block diagram of the Host

Mortar application

Tries to run the modified SurgeTelos faster than 4Hz failed. After extensive testing it was found that this was due to the Tmotes onboard humidity and temperature sensor SHT11.

Consulting the sensor data sheet available at [3] it was found that the built in ADC conversion was of high accuracy with the drawback of a high conversion time of about 265ms. It was also found that the when calling the sensor via the TinyOS general distribution sensor driver to read temperature it also read humidity, even if this was not requested. To improve sensor performance the driver had to be rewritten to lower the accuracy and also to only read temperature.

With the new driver implemented in the revised SurgeTelos a new application could be created, namely Mortar or ’My Own Reading Temperature Application Revised’. With this final application sampling frequencies of 8Hz can be achieved with high reliability.

5.3 Host block

The host block is run on a Laptop with a 1.2 GHz pentium mobile processor with 632 MB of RAM running Widows XP Professional. For the MPC MATLAB version 7.1 R14SP3 and Hybrid Toolbox version 1.1.0 are used. To solve the MPC optimization problems CPLEX version 9.0 is used.

The diagram of the Host block is depicted in Figure 5.3.
5.3.1 Java level

Between the WSN block and the MPC level there is a Java level working as an interface between the two, represented in Figure 5.3 by the Simulink block *Measurements from mote and target*.

The base station mote is, as mentioned before, connected to the Host PC via its USB. In order to read out the information that the base station is receiving into MATLAB a virtual COM-port software and Java are used.

First the virtual COM-port software is used to associate each mote with a COM-port and give it a unique address. After this it is possible to use a Java program distributed with TinyOS to read the byte vector from each packet and export it into MATLAB. That is, the virtual COM-port software takes the electrical signals from the USB-port and presents them in *RS-232* standard which can be read by Java.

After this the TinyOS distributed Java program can read the information from the base station mote, extract the packets and present it as a byte vector were all bytes have been transformed into decimal number format. This vector can then be read out in MATLAB. However there is a fault in the conversion. The byte values sent from the motes are positive numbers but in the byte vector read out from java negative numbers appear. This is due to that Java internally assumes that the binary representation is in 2-complements and hence the conversion is wrong. This can be corrected by doing inverse 2-complements conversion and then doing a regular binary to decimal conversion.

5.3.2 MPC level

The MPC level consists of the two Simulink blocks *Observer* and *Hybrid MPC* in Figure 5.3. In the first an observer as described in 4.3.2 is implemented and in the second an MPC controller as described in chapter 3 is implemented.

The MPC algorithm is implemented using the Hybrid Toolbox for MATLAB [6]. In order to use the Hybrid Toolbox the model in Equation 2.12 section 2.4 must be implemented in the special programming language HYSDEL [8] standing for ’Hybrid System DEscription Language’, a language used to describe affine hybrid dynamical systems. This code can then be compiled using a HYSDEL compiler which will return the described system as an MLD model as described in section 2.4 Equations 2.13. Since the compiler is run in MATLAB the MLD model will be available to the user as a structure object containing the matrices describing it.

Knowing the MLD model for the system the controller can easily be synthesized using the library of MATLAB functions available in the Hybrid Toolbox [6]. The ’hybrid controller object’, which is on a form suitable for optimization software, can then be inserted in the block *Hybrid MPC*, also
available in [6], in Figure 5.3.

The optimization problem resulting from the MPC consists of in total 141 optimization variables, 93 continues and 48 binary. These variables are optimized over 585 mixed-integer linear inequalities. The average execution time to solve the optimization problem using CPLEX is 17 ms with a worst-case computation time of around 125 ms.

5.4 WLAN block

The WLAN block consists of the levels Transmitter, Wireless link and Receiver. The Transmitter is implemented on the Host PC by an Intel(R) PRO/Wireless LAN 2100 3A Mini PCI Adapter and the Receiver is a network common resource implemented on a D-Link DWL-2100AP.

After the control action has been computed the new reference values are sent to the target via this link. The new references for the local controller are set by updating the scalar values in the reference register on the target.

Discussions have been made to use UDP instead of TCP as the transfer protocol. The reason for this is that UDP is faster and that re-transmitted data is out of date when it arrives to the Target. However it has not been implemented since the information about that packets have been lost may be useful in future applications. One could for example adapt the control algorithm to work in a safer set if it is detected that the percentage of packets lost is high.

5.5 Target block

The Target block is implemented on a Pentium 133Mhz PC running the MathWorks software xPC target. This is a real time kernel programmable from MATLAB using SIMULINK. A National Instruments PCI-6024E DAQ-board is used to interfaced the process with the software. This card enables the Target PC to send the electrical signals controlling the hardware actuators.

The block diagram of the Target layer can bee seen in Figure 5.4.

5.5.1 The local controller

The local controller provides an easy to use interface towards the DAQ and hardware for the Host block to use. This includes functions with commands for computing and actuating the process output \( u_c = \gamma^{-1}(v) \) given a speed reference \( v \) and actuate commands for turning on and off the lamps given the \( u_1, u_2 \) references. In this block the position of the belt is also estimated using real-time kernel reliable integration of the belt speed, compare section 2.2.4. Another important feature of the local controller is its zero-order hold for
last received input. This functionality handles the event that communication between Host and Target is lost. In this case the local controller holds the last known process input.

5.6 The process

The belt is moved using a belt roller with an encapsulated 24V DC-motor which in turn is controlled using a DC-servo amplifier. The switching of the lamps is managed by using two relays, one for each pair of lamps, to turn on and off their supply current.
Chapter 6

Validation

For all simulations this chapter the MPC setup used is the one described in section 3.3 with the weights as in Equation 3.6 and boundaries as in Equation 3.7. In the simulations below where it is stated that packets are lost these losses are induced by the software simply by neglecting received packages. This enables repeatability of the experiments as well as a highly controlled loss rate.

6.1 Simulation of the suggested setup

To show the behavior of the setup described in section 3.3 the system is simulated with the resulting behavior as in Figure 6.1 and Figure 6.2, control signals as in Figure 6.3 and Figure 6.4. One can see that the system reaches the position set points, Figure 6.2, fast and with high accuracy. The temperature, Figure 6.1, on the other hand is different. The tracking of the positive reference slope is quite good with very small overshoot, on the negative slope however the tracking is very bad. This phenomena is due to that the system can increase temperature by turning on the lamps and thereby increase the momentary value of $T_{ss}$ higher than the set point and thereby heat the Mote faster. On the negative flank the only mean possible to reach the set point is to turn off the lamps and let the mote cool down by itself.

6.2 Reachable set points

As mentioned in section 3.2.2 all set points are not reachable. In fact only a small part of the state space is reachable. Looking at the output partition of the state space it can actually be decided which combinations of position and temperature that are feasible for this problem.

The partition reachable is derived using the reasoning as follows. In position $x^*$ there are four possible steady state temperatures to the system,
CHAPTER 6. VALIDATION

Figure 6.1: Temperature simulation of used setup

Figure 6.2: Position simulation of used setup
Figure 6.3: Applied continues input in simulation of used setup

Figure 6.4: Applied binary input in simulation of used setup
these are

\[ T_{ss}(x^*, u_1 = 0, u_2 = 0) = T_{amb} \]  
\[ T_{ss}(x^*, u_1 = 0, u_2 = 1) \]  
\[ T_{ss}(x^*, u_1 = 1, u_2 = 0) \]  
\[ T_{ss}(x^*, u_1 = 1, u_2 = 1) \]

Now, if one introduces ‘fast’ switching of the binary inputs one should be able to in ‘mean’ reach a steady state point anywhere between the points defined in Equation 6.1. That is, the by ‘fast’ switching reachable points are defined by the set \( \{ (x, T) : T_{amb} \leq T \leq T_{ss}(x, 1, 1), x_{min} \leq x \leq x_{max} \} \). The partition of the output space that is inside this set is shown as the grey area in Figure 6.5.

### 6.2.1 Example of non-reachable set point

If the MPC is given a non-reachable set point it will try to get the best possible fit to this point according to the weighting of the referenced variables as said in section 3.2.2. In Figure 6.6 and Figure 6.7 the system behavior when trying to deal with one set of non-reachable set points is shown, control inputs as in Figure 6.8 and Figure 6.9. The reference step is applied at time 0.
Figure 6.6: Temperature simulation of non-reachable set point

Figure 6.7: Position simulation of non-reachable set point
Figure 6.8: Applied continues input in simulation of non-reachable set point

Figure 6.9: Applied binary input in simulation of non-reachable set point
6.3 Simulations with packet loss introduced

The simulations done to illustrate packet loss are all done by studying the effects on a given reference step. Before introducing packet loss a simulation of the nominal step response is shown in order to illustrate the nominal performance. The simulation of the nominal system is shown in Figure 6.10 and Figure 6.11, control inputs in Figure 6.12 and Figure 6.13. As one can see from the figures the system performs nicely and converges towards the set points.

6.3.1 Behavior with feedback packet loss

Now packet loss in the feedback link is introduced. The system behavior to this is dependent on the accuracy of the plant model. Simulations will therefore be misleading since the model used in the observer is perfect. However, one can illustrate the behavior of the theoretical performance for a perfect model. This is done in Figure 6.14 which shows the systems temperature tracking performance when 90% of all packets are lost. The dashed red curve is here representing the nominal behavior and the blue solid curve is representing the actual behavior. As one can see the two curves are almost identical.
Figure 6.11: Nominal position step response performance

Figure 6.12: Applied continues input in nominal step response performance
Figure 6.13: Applied binary input in nominal step response performance

Figure 6.14: Temperature step response with 90% feedback packet loss
6.3.2 Behavior with forward packet loss

Introducing forward packet loss, which can be simulated with a representative result, induces a number of problems. The first and most obvious is that the system is not actuated according to the controller command. Secondly the observer does not know that the packet has been lost in the forward transfer and therefore assumes that the sent output also was actuated. This will lead to wrongful state estimation. Given a 90% drop rate the system will evolve according to Figure 6.15 and Figure 6.16, were the red dotted curve is representing the nominal behavior and the blue solid curve is representing the actual behavior. As one can see the temperature response only differs from the nominal by having a bit larger overshoot and settling time. The position response on the other hand has a large overshoot and relative large settling time. The effects of forward loss is probably better illustrated in the control input signals to the system as seen in Figure 6.17 and Figure 6.18, nominal output shown in dotted red, signal sent by controller in dashed black and actuated signals in solid blue.

6.3.3 Behavior with multi directional packet loss

When combining both forward and feedback packet loss the system will be affected by the cross coupling of the both these two phenomena. As mentioned in section 6.3.2 the forward loss will cause error in the state
CHAPTER 6. VALIDATION

Figure 6.16: Position step response with 90% forward packet loss

Figure 6.17: Continues input in step response with 90% forward packet loss
prediction since the input that the observer assume is received in fact is not. Dealing with only forward packet loss this problem is overcome by the measurements received from the plant which can be used to correct the wrongful prediction. However when both forward an feedback data are lost at the same time this correction can not be done and hence the state estimation error becomes larger. Again as mentioned in section 6.3.1 the observer in the simulated setup is perfect and therefor the effects of this multi directional packet loss will be almost the same as for only having forward packet loss. Figure 6.19 and Figure 6.20 show a simulation were the packet loss rate is 90% in both directions. Figure 6.21 and Figure 6.22 show control outputs with nominal output in dotted red, computed output in dashed black and the applied output in solid blue.
Figure 6.19: Temperature step response with 90% forward and 90% feedback packet loss

Figure 6.20: Position step response with 90% forward and 90% feedback packet loss
Figure 6.21: Continues input in step response with 90% forward and 90% feedback packet loss

Figure 6.22: Binary input in step response with 90% forward and 90% feedback packet loss
Chapter 7

Conclusions

7.1 Wireless Sensor Networks

This report presents an example where a Wireless Sensor Networks successfully has been used for feedback in a control loop. This shows that the problems with dropped data packets in both forward and feedback communication can be handled with quite a good result using standard techniques.

The Tmotes themselves are up to being used in control systems. To avoid jitter in the sampling time, care has to be taken when programming them to ensure that the code is schedulable.

The biggest problem when using WSN in control loop is the WSN to PC interface. This is something that has to be improved since it currently is the biggest bottleneck in the control loop. There is for example currently no timeout when packets are lost, hence all the code used to extract packets from the net gets stuck when a packet is dropped. This can probably be overcome using threads and optimizing the packet readout from the network.

7.2 Hybrid MPC

When controlling complex hybrid problems as the one described in this report Hybrid MPC has several advantages in comparison with traditional control laws. The most obvious advantage is that it offers a high level of abstraction in the way that constraints on states, inputs and outputs are handled automatically by the algorithm. In the setup used it has also proven to be easy to tune. The only possible drawback is that it is quite computationally heavy. However it has proven to be fast enough for this application.

Problems with high worst-case computation time can be overcome by imposing computational time constraints on the optimization solver, when the optimization algorithm times out the best solution found is applied.
Bibliography

[1] A. Bemporad, *Short Course on Model Predictive Control and Hybrid System*


[17] A. Bemporad, Predictive Control of Teleoperated Constrained Systems with Unbounded Communication Delays


[22] R. Solis, V.S. Borkar, P.R. Kumar, A New Distributed Time Synchronization Protocol for Multihop Wireless Networks
