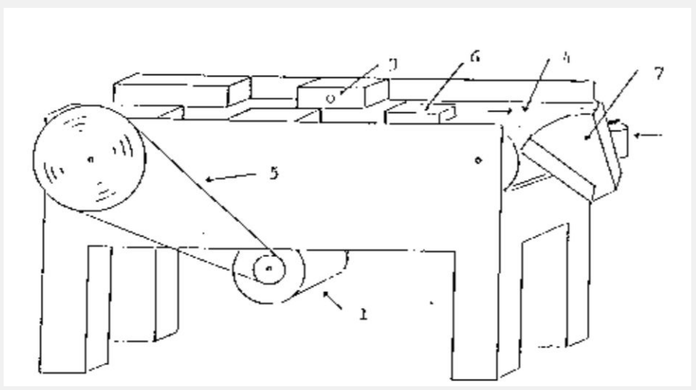
Computer Control Systems, Conveyer Belt Project

Omar Gallardo Puertas, Jonas Kohlschmidt

Project Overview

The task of the final project in the course “Computer Control Systems” was to enable a sorting machine to sort wooden sticks according to their length. The machine consists of a conveyer belt to transport the sticks, a moveable skid to slide the sticks to different end positions (e.g. boxes) and a sensor to measure the length of the sticks during their transport on the conveyer belt.



Picture 1: Schematic of the sorting machine

The conveyer belt and the skid could be moved with DC-motors, thus the object could be controlled with a Linux-process computer for communication by implementing digital controllers for each, the belt and the skid. Additionally a statistics program was implemented to give an overview over the processed sticks.

The control was implemented in C using the Real Time Application Interface (RTAI) for Linux. This facilitated running parallel threads for e.g. the skid, the belt and the sensor to enable a fast and reliable control of the machine.

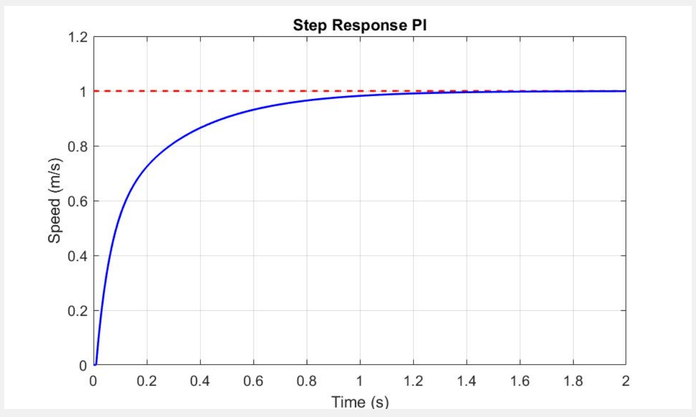
After control implementation the machine was able to sort the sticks almost error-free at full range of the conveyer belt speed (0.1 - 2.4 m/s).

Conveyer Belt Control

As mentioned above the belt could be set to different speeds to transport the wooden sticks. A PI-controller for setting the belt speed was designed and implemented. An integral controller was chosen to eliminate steady-state differences from reference signal to the actual speed to guarantee precise pace.

To design the controller a MATLAB SIMULINK model of the belt motor was used. The corresponding transfer function was multiplied with a constant for the belt’s tachometer. A conservative **phase margin (pm) of 110°** was selected to avoid any oscillation of the constantly moving belt. The **zero of the PI-controller was chosen to be five times lower than the 0dB frequency** determined through the bode plot of the transfer function and the pm. This constitutes the integral design part of the controller. For the proportional part the maximal possible phase margin was chosen. A sample time of 0.002s was found to be sufficient for both, the belt and the skid control.

To simplify the process of designing a suitable controller, a MATLAB script was conceived to automate this procedure with regards to the tuning knobs phase margin, PI-zero and sample time.



Picture 2: Step response of the PI-controller

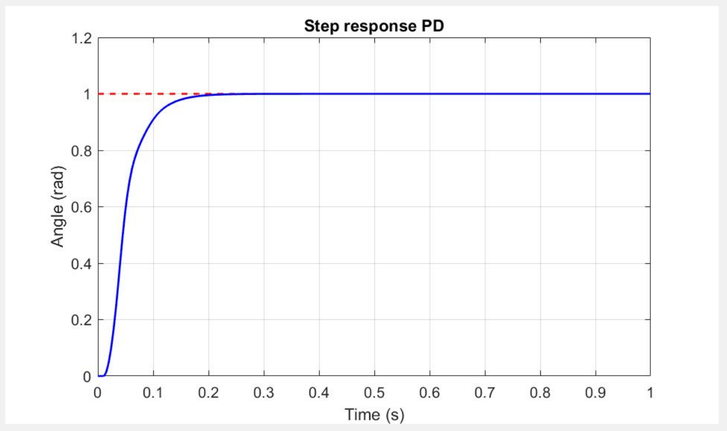
It can be appreciated that the step response of the belt system with the designed controller shows no steady state-error. The rather slow reaction of the system (~1.6s to reach the desired value) is not of importance since the speed of the belt is not subject to change during running.

Skid Control

The skid position, which sorts the sticks in different box compartments, was set using a PD-controller. This controller type was chosen in order to move the skid as quick as possible. The steady-state offset that goes hand in hand with PD-controllers can be tolerated since the box compartments provide some scope within the skid position.

Similar to the PI-controller, a SIMULINK model of a DC motor was modified (with an integrator, gear ratio and a potentiometer constant) to obtain a transfer function representing the change of skid position corresponding to the voltage input. The **pole of the transfer function** was determined in order to cancel it out with the PD-controller. This, in combination with the tuning factor **alpha=0.2**, gives the derivative part of the controller. The proportional part was designed using a phase margin on **70°**. In order to speed up the control, the proportional part was doubled.

Analogous to the design of the PI-controller, a MATLAB script was conceived to simplify the design procedure.



Picture 3: Step response of the PD-controller

It can be appreciated that the controller shows a quick reaction time despite using rather conservative tuning parameters. As this controller was sufficient handling the full range of the conveyer belt speed there was no need to implement faster, slightly overshooting controllers which were designed as well.

The PD-controller was converted from floating-point to fixed-point. Therefore a DC-limit- and a large-signal-analysis was carried out.

Implementation

The program was implemented in C using RTAI Linux. The communication with the sorting machine was enabled using the Linux computer device “Comedi”.

Five different parallel threads were used to

DTU Electrical Engineering
Department of Electrical Engineering
DTU Electrical Engineering
Department of Electrical Engineering
DTU Electrical Engineering
Department of Electrical Engineering

handle belt and skid in- and output, menu navigation, timing and sensor output simultaneously. To handle the real time execution of program semaphores were used to mutually exclude some of the threads

Menu

The menu was implemented to navigate through the different program functions in the terminal, e.g manually adjusting the length criterias for the wooden sticks. An overview is given in the picture below.

```
********************MENU**************************

 Select one of the options below

 1: Show paramater values
 2: Set skid position
 3: Set Belt speed
 4: Show Statistics
 5: Change length categories
 6: Close program

****************************************************
```

Picture 4: Menu navigation

The program also takes track of size and number of processed sticks. The statistics output is shown in the following picture.

```
************ STATISTICAL ASSESMENT *************

 Number of sticks per category

                    ____
          ____     |    |
         |    |    |    |
  ____   |    |    |    |
 |    |  |    |    |    |
 |    |  |    |    |    |
   0       0       0         0          0
 small   medium   large   too small  too large

 TOTAL: *0*

 Select one of the options below
 1: Reset parameters
 2: Return to main menu
```

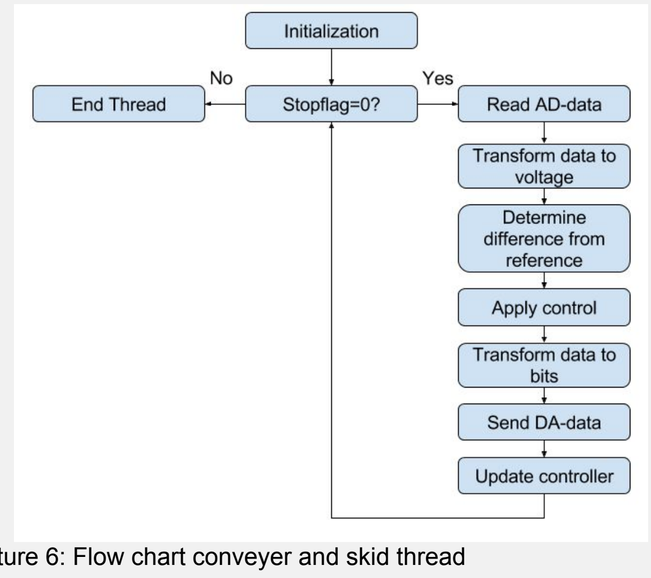
Picture 5: Statistical output

Conveyer Belt Thread

The presented PI-controller for the conveyer belt was implemented using simple initialization, control and update functions. The communication with the sorting machine consisted of reading and writing data to and from DA-/AD-converters. A flow chart depicted in picture 6 gives an overview over the logic used in the conveyer belt thread as well as in the skid thread.

Skid Thread

As stated the skid thread used the same control and communication logic as the conveyer thread. The controller was implemented using fixed-point values.



Picture 6: Flow chart conveyer and skid thread

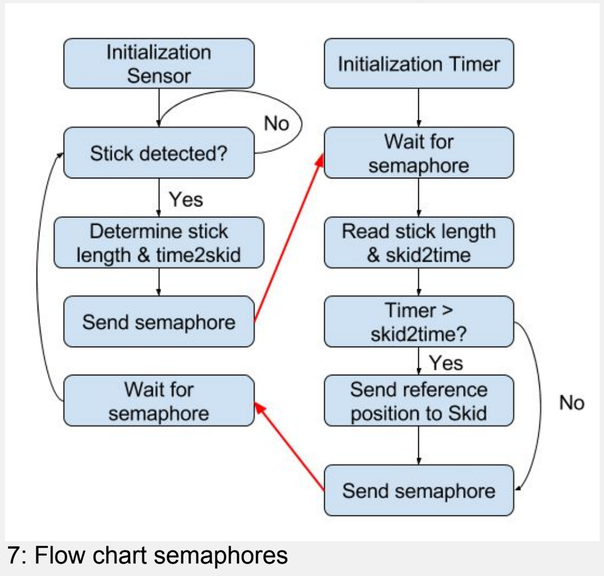
Sensor Thread

The sensor outputs a logical one whenever it detects a stick and a logical zero when not. It measures the time when a logical one is present. Using the conveyer speed the stick length is calculated and the stick is assigned to one of three length categories that can be altered in the menu. Additionally counters are increased for every detected category to provide statistical assessment.

Eventually the time the stick needs to reach the skid is calculated and a semaphore for mutual exclusion is sent to the timer thread which is hereby activated. The Sensor thread is stopped until it receives a semaphore from the timer thread and starts again.

Timer Thread

As stated the timer thread is paused until it receives a signal from the sensor thread. If activated, it reads the length category of the detected stick and the time the stick needs to reach the skid (=time2skid) from an array. It compares an internal timer with time2skid, if the time is reached it sets the reference position of the skid, if not it pauses and compares again the next cycle. For the case more sticks get detected before the timer reaches time2skid, they are stored in the array and processed one after another by the timer. The flow chart gives a overview over the semaphore logic.



Picture 7: Flow chart semaphores

Testing

Before testing the final program, small tests have been done during the development. This is because the parameters obtained in Matlab for each of the controllers are designed for ideal conditions. These small adjustments were done printing out the output data obtained (in volts) and comparing them to the measurements given by the oscilloscope. By this way it was possible to tune the desired positions of the Skid and also eliminate the offset of 0.15V from the belt.

Also, some tests were made regarding to the bricks detection. This is because the length of the sticks and the time that these ones took to reach the skid were calculated in function to the belt's speed.

Once every part of the system was tuned it was the moment of making the final tests. Although the system had to be executed only with the speed of 0.2 and 0.8 m/s, it was tried for every velocity from 0.2 to 2.4 m/s. Then, the next two problems were found for different speeds:

- Slow speed. When the belt were running below around 0.5 m/s, the skid turned too soon, so some new calculations had to be done in order to adjust this issue.
- High speed. When the belt were running over 1.5 m/s, the sticks length obtained from the sensor started to be larger than the real values. Therefore, the function made to calculate this length had to be changed.

By solving these two issues the testing procedure was concluded.

Conclusion

The sorting machine worked as expected for the asked speeds, even leaving less space than 1 mm between bricks. The only small problem found was running at speeds higher than 2 m/s, that for small distances between sticks it was not fast enough to go from the left to the right in time and vice versa.

DTU Electrical Engineering
Department of Electrical Engineering
DTU Electrical Engineering
Department of Electrical Engineering
DTU Electrical Engineering
Department of Electrical Engineering

Fact sheet

PI-Controller for the Belt

Choosen sample time:

$$T_s = 0.002s$$

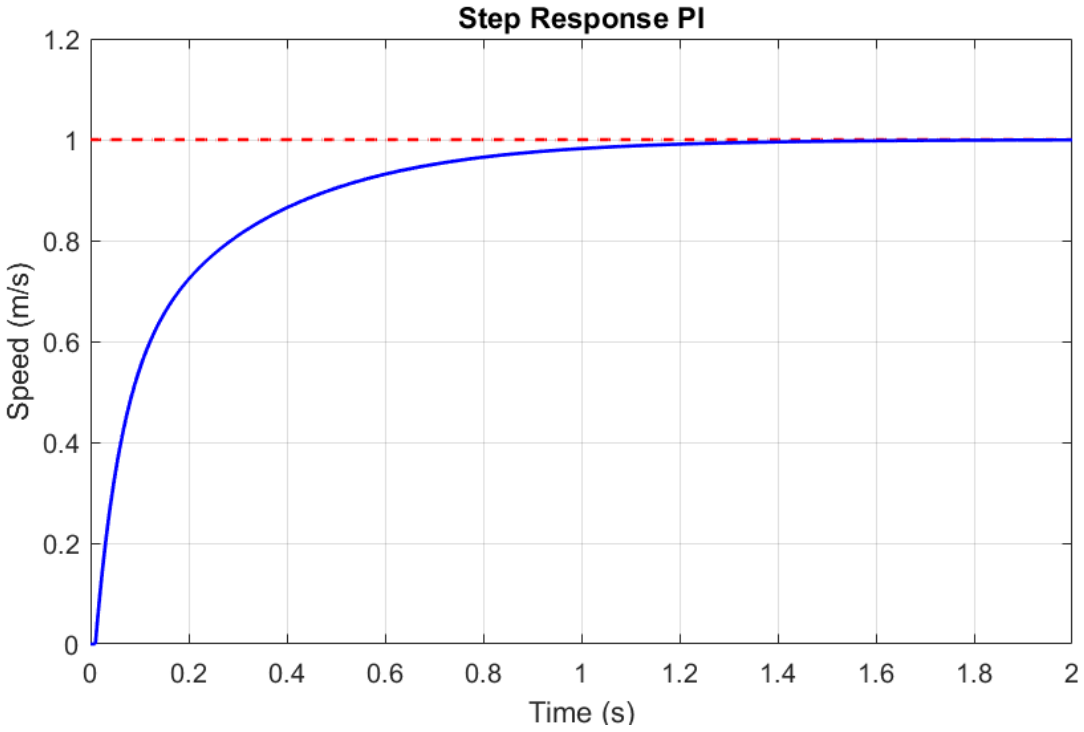
Tachometer constant:

$$T_a = \frac{2.4V}{1000rpm} = \frac{2.4 \cdot 60}{1000 \cdot 2\pi} = 0.0248$$

Controller transfer function:

$$G_{cd}(z) = \frac{0.9955 - 0.9841z^{-1}}{1 - z^{-1}}$$

Step response of the controller:



PD-Controller for the Skid

Choosen sample time:

$$T_s = 0.002s$$

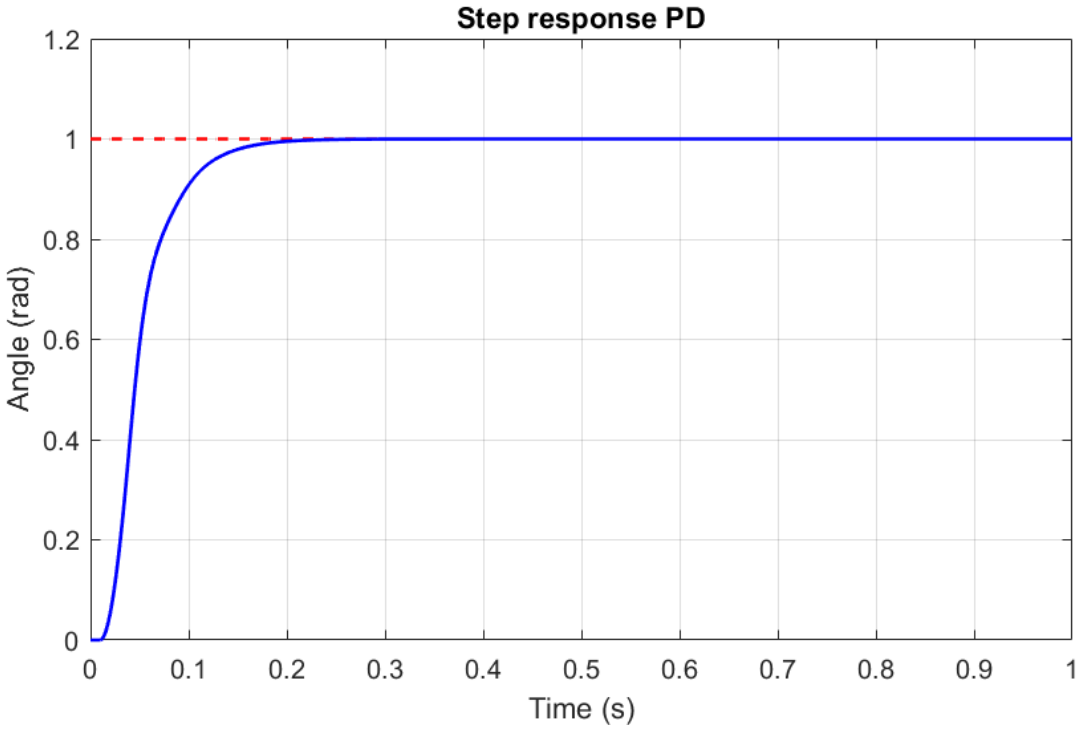
Potenciometer constant:

$$Pot = \frac{(15 - (-15)) \cdot 180}{350\pi} = 4.9111$$

Controller transfer function:

$$G_{cd}(z) = \frac{121.8 - 114.8z^{-1}}{1 - 0.7408z^{-1}}$$

Step response of the controller:



Fixed Point PD-Controller for the Skid

Parameters calculation:

$$-1 < \frac{b_0}{2^n} < 1 \longrightarrow n = 7, \quad b_{0fp} = \frac{b_0}{2^n} \cdot 2^{15} = 31181$$

$$-1 < \frac{-b_1}{2^n} < 1 \longrightarrow n = 7, \quad b_{1fp} = \frac{b_1}{2^n} \cdot 2^{15} = -29388$$

$$-1 < \frac{-a_1}{2^n} < 1 \longrightarrow n = 0, \quad a_{1fp} = \frac{a_1}{2^n} \cdot 2^{15} = -24274$$

Small signal analysis:

$$S_{AD} = \frac{h_{AD}}{2} D(z) \frac{1}{1 - z^{-1}}, \quad S_1 = \frac{h}{2} \frac{1}{1 - z^{-1}}, \quad S_2 = -\frac{h}{2} \frac{1}{1 - z^{-1}}, \quad S_3 = \frac{h}{2} \frac{b_0 - b_1 z^{-1}}{1 - a_1 z^{-1}} \frac{1}{1 - z^{-1}}$$

$$S_{DA}(z) = h_{DA} \cdot 1$$

where $D(z) = G_{cd}(z)$

$$max\Delta e_{ss} = \lim \left((1 - z^{-1}) \left(\frac{|S_{AD}(z)|}{|D(z)|} + \frac{\max|v(z)|}{|D(z)|} + \frac{|S_{DA}(z)|}{|D(z)|} \right) \cdot |H(z)| \right) \; for \; z \to 1$$

where $H(z) \sim 1$ for $z \to 1$

$$max\Delta e_{ss} = 0.5h_{AD} + 0.5h + 0.037h_{DA}$$

Large signal analysis:

$$x1 = \frac{1}{1 - a_1 z^{-1}} e_s = 3858 e_s \qquad x2 = \frac{b_0}{1 - a_1 z^{-1}} e_s = 431.32 e_s \qquad x3 = \frac{b_0}{1 - a_1 z^{-1}} e_s = 404.32 e_s$$

for a system running on 16 bits values in 2's complement:

$$min = -2^{15} = -32768, \qquad max = 2^{15} = 32768$$

for 1.4V of change from left to right position, the internal error is: $e_s = 287$

$$x2 = 431.32 e_s = 123788.84 > max$$

Internal error shifted **two bits down**:

$$e_s = \frac{287}{2^2} = 71.75$$

Now x2 will be within the limits:

$$x2 = 431.32 e_s = 30947.21 < max$$