Testing the structured method to design embedded control software

N.B. (Niek) Morsinkhof

BSc Report

Committee:
T.G. Broenink, MSc
dr.ir. J.F. Broenink
dr. C.G. Zeinstra

July 2019

032RAM2019
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

The aim of this bachelor assignment was to test the structured method, by applying the method to the development of the control software for a slider setup. This slider setup existed out of two rails which were positioned perpendicular on each other. each rail had a cylindrical shaped mass attached to it. The goal for the development was to let the masses move in such a way that it was possible to wrap a string from one mass to the other mass.

The structured method makes use of a iterative process, where the total design is first divided into different features. The features are first developed based on a simplified model of the physical system, and then these designs are tested with iterations, where at every iteration cycle detail is added to the model of the physical system.

Before the the structured method was applied to the development of the slider setup, an analysis of the physical setup, and of how the setup should operate, where made first. After this the structured method could be applied, where different features were defined, and different models on different detail levels were made. When the preparation of the structured method was finished, the development could begin with the iteration process.

At the end of the development process the design-goal was not entirely achieved since problem arose when the velocity of the masses was increased. However, it became apparent that a mistakes were made in the implementation of the structured method. Fixing this mistake enabled the masses to move at a higher velocity.

At the evaluation of the process conclusions could be drawn based on the mistakes that were made during the application of the method.

Contents

1 Introduction

1.1 Context

There are some challenges while designing a cyber-physical system. To come to a successful and reliable design it is important to remove errors as early as possible, since eliminating errors at the end stage of the design can be very costly. One approach to design a cyber-physical system regarding control software is the use of an iterative process. This iterative process first breaks down the control system into different control features, and each control feature is then developed with a loop in such a way that with every cycle more detail and complexity is added. The order of addition of the different details are important to make the process as efficient as possible in time effectiveness and in detecting errors.

1.2 Goal

The goal of this project is to evaluate the structured method (Broenink and Broenink (2019)) based on the application of the method to the setup in this project. Conducting this project should result in a better understanding of the benefits and drawbacks of applying this method to the development of embedded software control. The outcome of this research can be used together with other tests of the method to come to a more general conclusion of the benefits of this method.

The main research question in this project is stated as follows:

- **How applicable is the structured method approach for the design of embedded control software of a cyber-physical system?**

It can be hard to have an objective measure of how applicable a certain method is. The applicability of a method can be measured in a relative manner compared to other manners. The applicability can also be measured by analyzing the time efficiency of every step and order of the steps. This could be done by asking whether it would be more time efficient to deviate at any point in the process from the tested method. Finally it can be determined how effective the method is in the detection of errors.

To answer the main question different sub-questions will be answered:

- **Was it in any stage of the development more beneficial to the efficiency to deviate from the method?**
- **Is it likely that the same design errors which were spotted using the method, would also have been spotted without using this method?**
- **If these errors were spotted without the method, would it have cost more or less time to fix them without using the method?**

1.3 Project outline

First the structured method will be eludicated in chapter 2. After it is clear how the structured method should be applied to the development of software control, the setup that is used to test the method will be analyzed in section 3.1. It will be explained what the setup eventually has to do, and what the design will look like. In the next chapter the implementation of the

structured method begins. This implementation will be done according to the order described in the analysis. After the implementation there will be a brief evaluation about the implementation of the method, and this evaluation will be continued in the discussion. At the end the research questions will be answered, and recommendations will be made which may improve the method.

2 Analysis

2.1 The structured method

The method that will be tested in this project is a method to develop control software for a cyber-physical system. The idea of this method is to develop the control software in a time effective manner, by means of an iterative process where errors are spotted in an early stage.

In figure 2.1 it can be seen that the method consists out of an outer and an inner cycle. The outer cycle describes how different control features are added to the design, whereas the inner cycle represents the development of the features. Following 2.1 gives the following steps:

- Preparation
- 1) Design new feature and test the new feature.
- 2) Implement and test new feature.
  A) Design feature based on the ideal model.
  B) Combine the feature with a more detailed version of the system.
  C) Determine if the test passes. If if it passes, add more detail so go to B. If it does not pass redesign the feature and go to A.
- 3) Continue to the next feature, until all features are implemented.
- Evaluate and reflect on the cycle process.

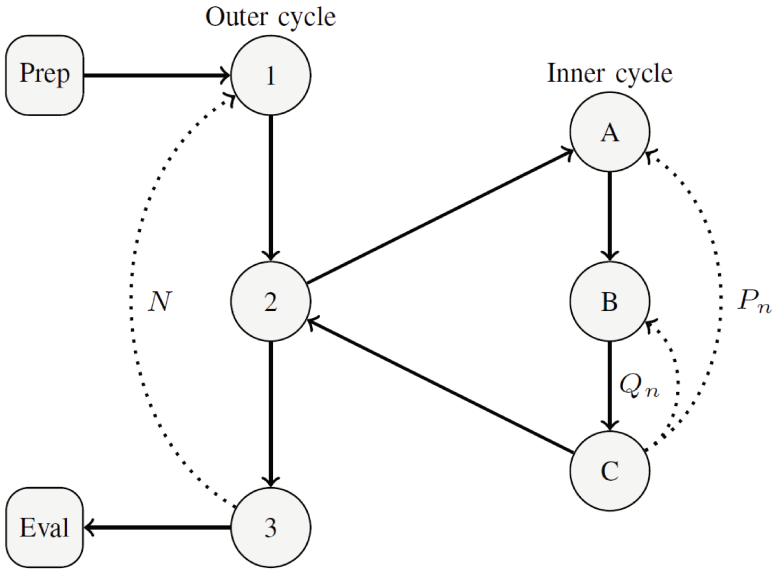

Figure 2.1: Schematic of the method with an inner and outer cycle. Broenink and Broenink (2019)

Preparation

Before the development process, models with different level of detail of the physical system should be available. the order of implementation of the details should be well thought off before the process. Details which are more critical and have the most influence on the behavior should be implemented first. In this way design errors are spotted relative early in the process. Steps that take a lot of effort to implement should be simulated as late as possible, which decrease the chance that these details are implemented at an unsuccessful cycle.

Outer cycle

The execution of the outer cylces require that the intermediate designs can be tested at the end of every cycle. The test can be based either on the prototype, or on simulations. When the test is done on the prototype it is important that previous features can quickly be deployed on the feature. This also shows that it is important to order the development of the features properly, since there may be features which depend on previous features. Whereas, other features operate more independently which means that these could be developed earlier in the process.

Inner cycle

The main idea behind the inner cylce is that the design is first based on an idealized version of the model, and detail is added at every successful test. By developing a feature with this addition of detail, the cause of a design error is easy to determine. Because of the multiple steps in detail, different sets of the prototype model can be created, which is represented in figure 2.2.

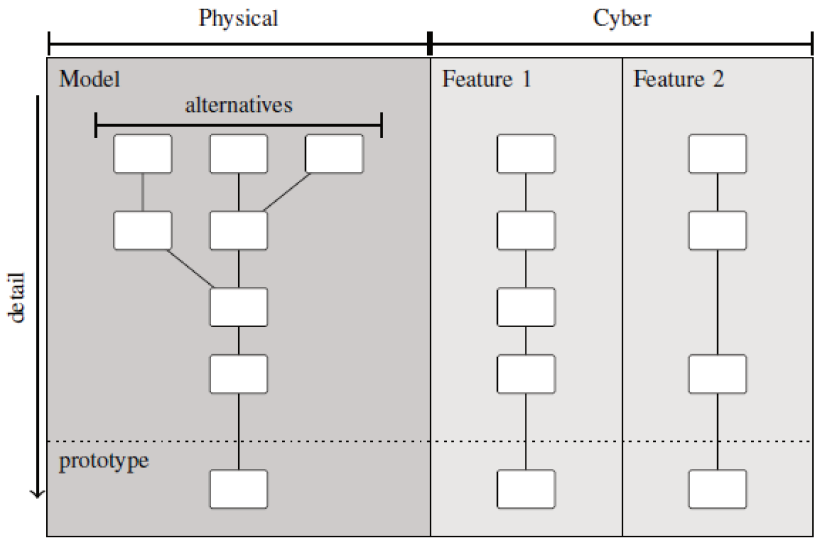

Figure 2.2: Set of models built up by using the structured method. Broenink and Broenink (2019)

2.2 The physical system

Before the method will be applied and tested, it must be clear which physical system will be used and what this physical system has to do. So, the next step is to analyze the slider setup, next to that the steps that the slider setup has to take have to be precisely described, in order to define features for the method. It should be noted that this is not yet part of the preparation of the method, but it is merely an description of the physical system, together with the description of the aim of this physical system.

3 Approach

In this section it will be described to what control problem the structured method will be applied, to test the structured method. It will also be discussed how the implementation of the method will be evaluated.

3.1 The setup

To test the method a slider setup is used, which is shown in 3.1. There are two identical sliders which are positioned perpendicular relative to each other. Each slider has a rail, with a small cylindrical shaped mass, which can move along the rail. To move the masses along the rail, each mass is connected to a belt, and this belt is strained around two axles like in figure 3.2. the two axles can be driven by an electric DC motor, and this motor is controlled by a Raspberry Pi. In this report the lower slider and motor will be referred to as slider 1 and motor 1, and the upper slider and motor will be referred to as slider 2 and motor 2.

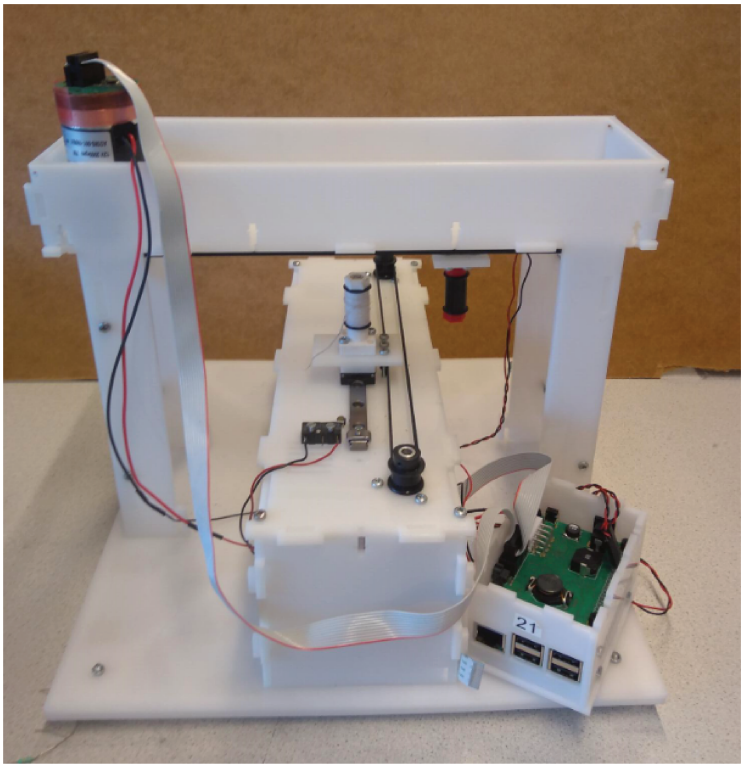

Figure 3.1: The slider system which is used in this project

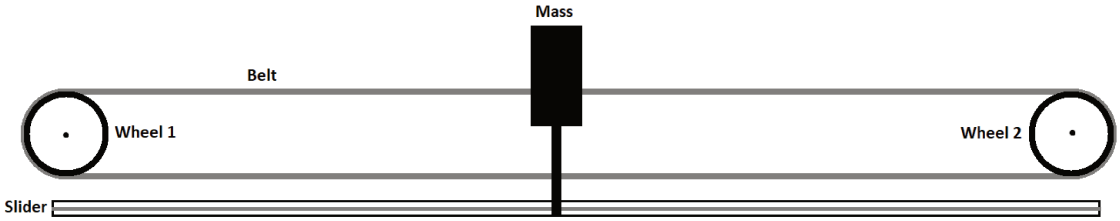

Figure 3.2: Schematic of one slider with mass attached to belt

Sensors

Each slider is equipped with an angle sensor. This sensor measures the angle of the axle of the electric motor. The output of these sensors will have a value between 0 and 1, representing the angle between 0 and 2π. For the feedback it is more convenient to have the total angle, since $2k\pi + \theta$ is of course not the same as $2n\pi + \theta$ for $k \neq n$, while the direct output does not make a distinction between the two.

Each rail is also equipped with a switch at the end of the rail. When the switch of rail 1 is activated, the output of pin 4 of the Raspberry Pi will be set from 0 to 1. After releasing the switch, pin 4 will be set to 0 again. This works the same for the switch on rail 2, only here the output pin is pin 18. These switches can be used for homing, such that the position of the masses can be determined.

3.2 The controller

Before the implementation of the method is started, it should first be clear what exactly the aim of the controller is, and what the approach of the controllers design will look like.

3.2.1 Aim of the controller

Eventually the setup has to be controlled such that one cylindrical shaped mass with a rope wrapped around it, can wrap the rope around the other cylindrical shaped mass. This implies that one mass has to circulate around the other mass, so that the rope unwraps around one mass and wraps around the other mass. It is also important that the distance between the masses remain constant since the length of the rope between the masses is constant too.

On the left of figure 3.3, a schematic of the top view of the setup is shown. Here we see mass 1 (M1) on the vertical rail and mass 2 (M2) on the horizontal rail, where the masses are connected to each other with a rope. On the right of the figure the desired orbit of mass 2 is drawn drawn as a circle where M1 is chosen as the center of a frame of reference. To realize the scenario on the right side of figure 3.3 the position of one mass has to move according to a sine while the other mass has to move according to a cosine.

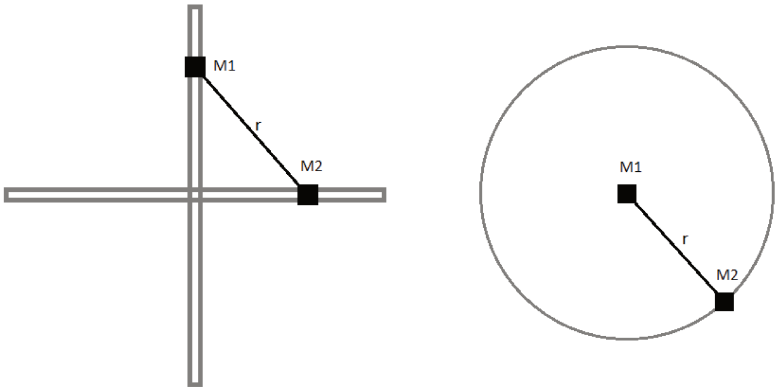

Figure 3.3: Left: Mass M1 on vertical slider, mass M2 on horizontal slider with rope in between with distance r. Right: M1 taken as origin in frame of reference with M2 orbiting around it.

3.2.2 Design of the controller

The idea of the design is that the position of the mass will be controlled with PID-control. There are two main scenarios which require there own positioning of the two masses.

The scenario at the start is that the information about the position is unavailable for the system. The only information available is the position relative to the initial position. However the initial position is unknown. In this first scenario the mass has to move to the switch at the end of the rail such that controller is able to position the mass in the correct initial position.

The second scenario is that the two masses have to move according to a sine and cosine, in order to make a circular movement with respect to each other. A schematic of the top level design can be seen in figure 3.4.

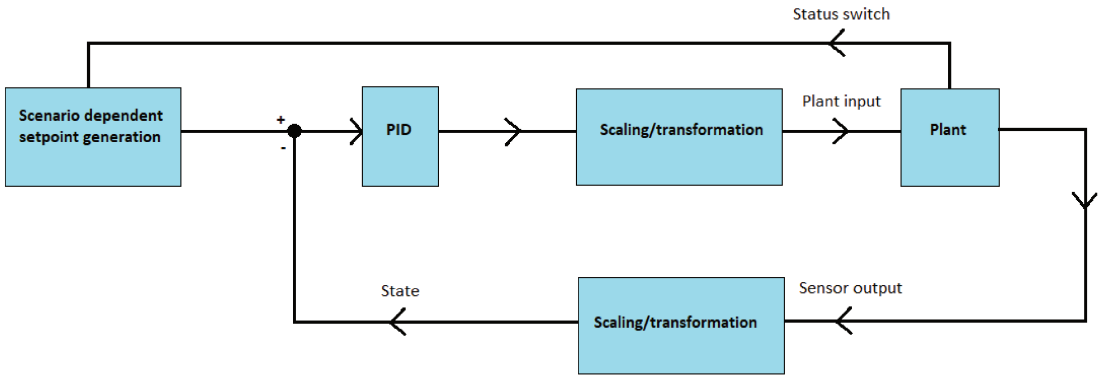

Figure 3.4: Block diagram of top level design

The first block determines the set-point based on the scenario of the system. The set-point generation determines the scenario based on the status of the switch of the setup.

The set-point generation gives the system orders on how to move the two masses. Eventually the system should execute the following tasks in the following order:

1. Move the mass of slider 2 in the direction of the homing switch.
2. When the mass switches the homing switch, reset its position to 0 and move the mass to the in ital position of the cosine at $t = 0$.
3. Move the mass on slider 1 in the direction of the homing switch.
4. When the mass switches the homing switch, reset its position to 0 and move to the initial position of the sine.
5. Hold the initial positions of the masses, so that there is time to connect the string from one mass to the other mass.
6. Start the circular movement.

After the homing, the position at the switch is defined as 0. So after the homing is done a reference position is made, which means that the slider system can be placed in a coordinate system, as is shown in figure 3.5. The slider is positioned in the third quadrant, because the sensor output gives a negative velocity when the masses move away from the switch. Since the position at the switches are defined as 0 for each slider, the slider has to be placed in the third quadrant.

In this figure $L1$ and $L2$ are the distances from the switch to the center of movement. This means that when the masses move in a relative circular movement with a radius r and angular velocity ω, they would move according to the following equations:

$$x = rsin(\omega t) - L2. \tag{3.1}$$

$$y = rcos(\omega t) - L1. \tag{3.2}$$

This implies the following initial positions:

$x_0 = r - L2$ and $y_0 = -L1$.

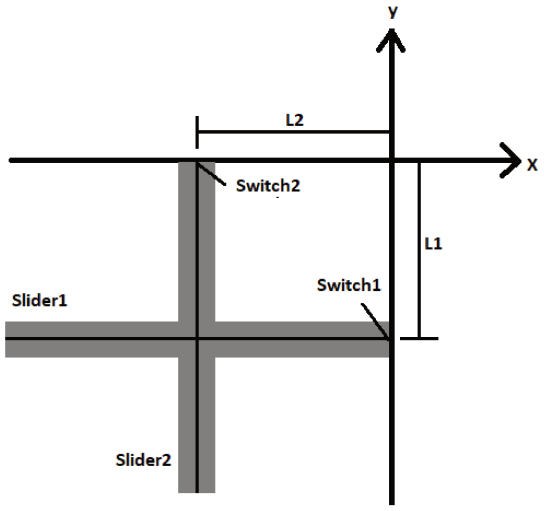

Figure 3.5: Slider system placed in coordinate system

Angle encoder

After a full rotation of the axle the angle sensor puts out 1 at 2π and one time step further, where the angle passes the 2π, the sensor puts out 0. This quick jump from 1 to 0 or vice versa in one time step can be filtered out by differentiating the output, and filter that output with a limit which is bigger than the maximum angular velocity but smaller than the derivative of the quick jump. After this limit the signal can be integrated to obtain the total angle. A schematic of this method can be found in figure 3.6, and the input and output of the schematic in 3.7.

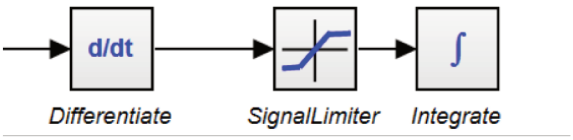

Figure 3.6: Blockdiagram to obtain total angle

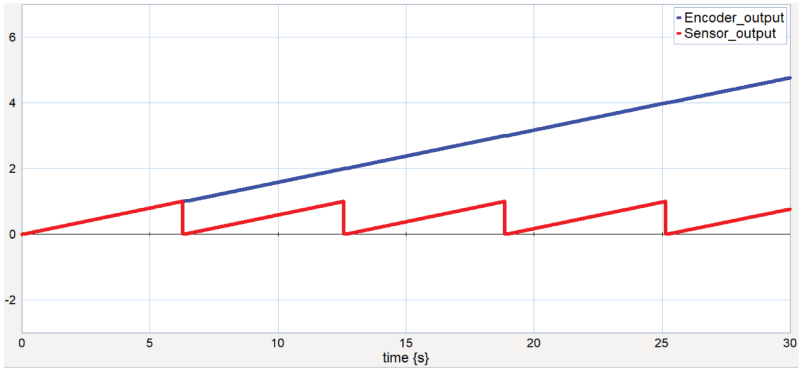

Figure 3.7: Sensor output(red) and encoder output(blue)

3.3 The evaluation

The aim of the evaluation is to answer the sub-questions from the introduction. To answer the questions it is important to keep an evaluation report during the iteration process, since at the time of the process the detailed insight is at its highest. The development of the control software will be done according to the structured method, but during the process it should be questioned for each step whether it was more efficient to deviate from the method. This evaluation for each step should help to answer the sub-questions.

After the method is applied to this particular project, the implementation of the method can be evaluated by placing all the B steps of one feature into a matrix, as in figure 3.8. Remember that the method aims to detect a design error as early as possible. Thus a good implementation of the method implies that for every redesign, or step P in figure 2.1, a minimum amount of steps of Q. So in an ideal implementation of the level of detail, the matrix representing all the iterations in the inner cycle would look like the matrix in figure 3.9. In this matrix the design error is in every cycle found in the first addition of detail, which means that the order of adding detail is chosen properly.

$$\begin{bmatrix} B_{1,1} & B_{2,1} & \cdots & B_{p,1} \\ B_{1,2} & B_{2,2} & \cdots & B_{p,2} \\ \vdots & \vdots & \ddots & \vdots \\ B_{1,q} & B_{2,q} & \cdots & B_{p,q} \end{bmatrix}$$

Figure 3.8: Matrix with all the B steps of the development of one feature

$$\begin{bmatrix} B_{1,1} & B_{2,1} & \cdots & B_{p,1} \\ & & & B_{p,2} \\ & & & \vdots \\ & & & B_{p,q} \end{bmatrix}$$

Figure 3.9: Matrix with all the B steps, of feature with ideal implementation of the method

4 Implementation of the structured method

In this chapter it is described how the structured method is applied to design the controller. The features and the different layers of detail will be described. After that, the approach of the evaluation will briefly be discussed.

4.1 Preparation

In this preparation the different features will be defined, and there order will be determined. After that the different details will be identified, which will then also be ordered. Based on the details, different models of the physical system can be made.

4.1.1 Features

In this section the total control of the setup will be divided into different features together with the order of the development of the features, since one feature may be build up another one.

The end goal of the design is to control the circular movement of the two sliders, such that a rope can be wrapped around a mass. To make this possible one mass has to move according to a sine, while the other mass has to move according to a cosine. Furthermore, the two masses have to oscillate around the same center, which requires the correct initial position for both masses. The correct initial position has to be managed by homing. In case it is not managed to design the homing feature, the masses can be given the correct initial position by hand. This is why the sine and cosine movement will be given the priority over the homing.

It is possible that the controller has to be redeveloped for relative high angular velocities. For this reason the sine/cosine movement will first be realized up to an angular velocity of 30 rad/s. The choice of this boundary of 30 rad/s is arbitrary, and will be changed if it turns out that major difficulties arise for higher or lower angular velocities. A controller which can move the masses at higher angular velocities will be treated as a separate feature.

In this case the features can all operate independently of each other. The masses can move according to a sine or cosine without homing and the homing can be developed without the development of the sine or cosine movement. So in this view there is not really a preference in order of the development of the features, apart from the slow and fast circular movement.

The order of the features will thus determined based on the importance of the features. As explained earlier in this section, the most important feature is the circular movement. After this it is decided that the homing is second most important, because it is preferred to have a fully automated system. This gives us the following order of features:

1. Slow sine/cosine movement
2. Homing
3. Fast sine/cosine movement

4.1.2 Detail

The most basic model that will be used is the linear model in figure 4.1. The controller is designed using PID-tuning. After a controller is designed using the most basic model, extra detail will be added to the model. The limit of the motor is probably the detail with the biggest influence. Next to the limit of the motor, there is also the elasticity of the belt that can play a role.

When a feedback loop is made to control the position the problem is that the position of the mass cannot be measured directly. This is not a problem when the elasticity of the belt does not have a lot of influence. however, if the elasticity of the belt does have a significant influence the error of the position measurement could also become significant.

Then there is the implementation of the discrete time of the model which correspondents to the sampling frequency of the raspberry pi. When the sampling frequency is too low, high frequency oscillations of the system could become undetectable. The final aspect is the prototype, here the control feature will be tested on the real setup. This leaves us with the following aspects:

1. Motor limits
2. Elasticity of the belt
3. Discrete time model
4. The prototype

4.1.3 Model of physical system

The models of the physical system will be represented in a bond-graph. The most ideal model that will be used is the model in figure 4.1. This is the model where the development begins. It consist out of an electrical, rotational and a translational domain. In the next detail level the motor limits are added behind the output of the PID-controller. Next, detail will be added to the translational domain in the form of a belt. This extension in detail can be seen in figure 4.2. Here three 0 junctions are added in the translational domain, these junctions behave as a spring damper system. It can also be seen that the integral to integrate the angular velocity in this model is discrete. At This point the model has reached its highest detail level. At that point all continues parts of the controller are replaced by their discrete version. These parts are among others, the PID-controller, integral blocks, and differentiating blocks. In the bond-graph model with belt there is also an impulse signal source added, which represents the switch signal. In the physical system the switch depends on the mass, but for the model an impulse signal which simulates the switch will do. This is because before the homing the position of the mass is random, so in the simulation the start time of the switch signal can be chosen randomly.

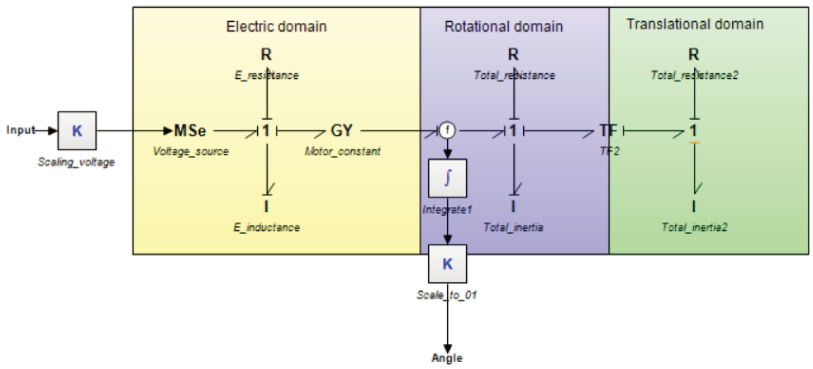

Figure 4.1: Bond-graph the ideal model of the physical system

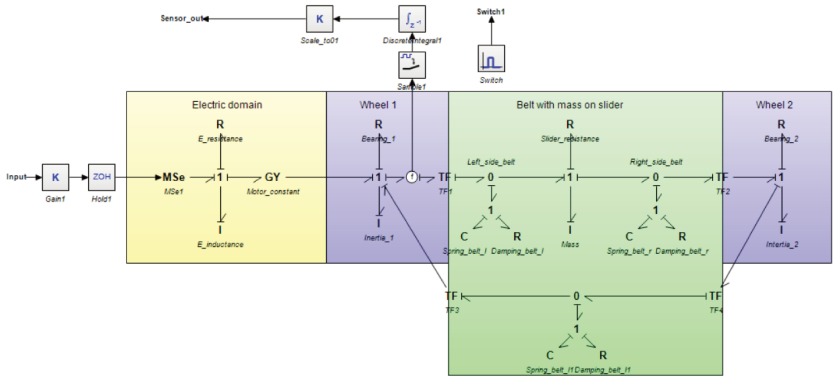

Figure 4.2: Bond-graph model of the physical system with belt

4.2 The iteration process

Slow circular movement motor 1

Step 1:

The initial position does not matter in this case, it is only important that the circular movement is stable and that the offset does not shift during the movement.

Test: The controller is tested with a sine input with an amplitude of 3cm, and it should be stable at an angular velocity of 30 rad/s. Also the overshoot cannot be too high since this may cause problems when the two masses are connected with a rope.

Step 2:

A1: The controller is designed using PID-tuning, based on the simplified model.

- B1,1: The motor limits are added to the model. The maximum voltage the motor can provide is 24 volts.
- C1,1: The controller works up to an angular velocity of approximately 28 rad/sec. Although the system is still stable beyond this angular velocity, the error of the first sine wave begins to have an overshoot of more than 1mm. This may be a problem for the system when the two masses will be connected with a string.

A2: It is possible that the error in the first wave of the sine is caused by a rise time which is to low. To compensate for this the proportional gain is increased to 3.

- B2,1: The limits of the motor are added to the model.
- C2,1: The error of the first wave is reduces to an acceptable level. There is still a small error in the amplitude when the sine is settled but this error is acceptable.
- B2,2: The belt is now added to the model.
- C2,2: There is no significant difference compared to the test without the limit of the motor.
- B2,3: Now the model is transformed to a discrete model.
- C2,3: There seems to be no change with regard to the previous model.
- B2,4: Finally the model is tested on the real system.

C2,4: The mass moves stable over the rail, although the movement seems to be somewhat jerky. For angular velocities above 8 rad/s the offset of the sine starts to shift after approximately 30 seconds. Furthermore the shift is clearly detectable with the eye however the measurement does not indicate a significant shift in the offset. This is problematic since the correction of the error of the controller depends on the measurement. If the measurement is not correct the controller will likely not correct the error. For relative low angular velocities, which we will define as 8 rad/s, the controller works. For higher angular velocities the controller has to be adapted.

A3: To correct the shift from the offset, the integrating gain is increased, as this gain decreases the steady state error. The proportional gain could also be increased to reduce the steady state error. however, the movement is already jerky and this is something that should be reduced somewhat, since it could be that this jerky behavior results in very abrupt short movements, which could cause the shift in the offset. The Kp is therefore set to 2.5.

B3,1: The motor limitations are added to the model.

C3,1: The tested model behaves stable, also the error in the amplitude is still acceptable.

B3,2: The behavior of the belt is added to the model.

C3,2: For this controller there is also no significant difference compared to the test without the belt.

B3,3: The discrete time is implemented into the model.

C3,3: There is no difference compared to the previous test.

B3,4: The controller will be tested on the real system.

C3,4: The mass behaves more stable which is to say that it is less jerky than with the previous controller. However the offset starts to move again, so increasing to 5 seems to have had little effect.

A4: It is tried to lower the proportional gain even more to reduce the jerky movement somewhat more, which is still there.

B4,1: The motor limitations are added.

C4,1: The tested model behaves stable, also the error in the amplitude is still acceptable.

B4,2: The behavior of the belt is added to the model.

C4,2: For this controller there is also no significant difference compared to the previous test.

B4,3: The discrete behaviour is added.

C4,3: There is no different behavior.

B4,4: The controller is tested on the real system.

C4,4: For an angular velocity of 8 rad/s the movement is stable and the center of the sine does not shift. For higher angular velocities up to 16 rad/s the movement remains stable, although the jerky behavior increases. The main problem for this higher velocity is that the the center of the sine shifts and the masses crashes into the end of the rail eventually. In addition to this problem comes the issue that the shift is not detectable in the measurement. According to the position measurement there is a minor change in the center of the sine, however this does not account for the 1cm change or more in the center of the sine which is detectable with the naked eye on the prototype.

For now it will be decided that higher angular velocities than the 8rad/s reached in the slow movement feature, will be included in the fast movement feature.[1]

Step 3:

The first feature works as desired up to an angular velocity of 8 rad/s. Now the next feature homing will be developed. The different detail steps can be found in figure 4.3, and the cycles are shown in the matrix in figure 5.1.

Slow circular movement motor 2

Because of the similarities of the the sliders, the result of the first feature is directly applied to the prototype of the second slider. The input of the controller was $r(cos(\omega t) - 1)$, since this makes the initial input value equal to the initial value of the angle sensor, which is 0. In he development of the controllers for higher angular velocities the two sliders will be taken separately.

Initial position M1

Step 1:

To have a proper working system it is important that the two sliders have the correct initial position. First the mass has to move to the switch, the switch detects the mass the position will be reset and the mass has to move to the initial position.

Test: To test this feature, the same models are used as the models for the previous feature. The mass should first move to the switch, where the position will be set to 0. After that the mass has to move to its initial position. The assumption is made that the masses cannot touch each other.

Step 2:

A1: For this feature virtual switches are needed to make the system do all the tasks in the right order. The PID-controller is made a little more aggressive than the controller for the sine and cosine such that a possible steady state error is as low as possible. The proportional gain is made a little bit higher than the derivative gain. The input of this controller will be a slope since a constant position as input would probably result in a too big overshoot which causes the mass to slam in the end of the rail.

B1,1: The controller is tested on the with motor limits.

C1,1: The behavior of the model is stable and the mass moves as expected. There is a small steady state error of less than 1 mm, which will not be considered as significant.

B1,2: The controller is tested on the model with the behavior of the belt.

C1,2: There is no difference compared to the simulation without the limit of the motor.

B1,3: The discrete time is implemented in the model.

C1,3: There is no difference with regard to the previous test.

[1] This was a mistake and will be discussed in the discussion

B1,4: The controller is tested on the real system.

C1,4: There is some unstable behavior in the first 2 seconds in the form of oscillation, but the mass moves to the switch. At the point where the mass touches the switch and the mass should turn around, it starts to oscillate for a brief moment before it stays put at the switch because it has activated the switch multiple time. The controller has to be adapted to prevent both the oscillation and the problem at the switch.

A2: To prevent oscillation the controller is made a little bit less aggressive by reducing the proportional gain to 2, and the derivative gain is increased to 3. To prevent that the mass stays put at the switch a small constant is added to the slope of the input when the mass moves back. The idea is that because of the small constant the mass will have a quick short movement away from the switch, such that the mass will only hit the switch once. This constant will be set at 1 mm at the first try.

B2,1: The controller is tested on the model with motor limitations.

C2,1: The behavior of the model is stable and the mass moves as expected. There is a small steady state error of less than 1 mm, which will not be considered as significant.

B2,2: The controller is tested on the model with the belt.

C2,2: There is no difference compared to the simulation without the limit of the motor.

B2,3: The discrete time is implemented in the model.

C2,3: There is no difference with regard to the previous test.

B2,4: The controller is tested on the real system.

C2,4: The controller seems to work as desired, which is to say it behaves as the simulation of this design. However testing the controller multiple times on the system shows that occasionally the mass oscillates at the switch for a brief moment before it stays put at the switch. The test is considered unsuccessful and an attempt will be made to eliminate the oscillation at the switch.

A3: The controller will be made less aggressive to prevent oscillation, so the proportional gain will be set to 1.5.

B3,1: The controller is tested on the model with motor limits.

C3,1: The behavior of the model is stable and the mass moves as expected. There is a small steady state error of less than 1 mm, which will not be considered as significant.

B3,2: The controller is tested on the model with the belt.

C3,2: There is no difference compared to the previous simulation.

B3,3: The discrete time is implemented in the model.

C3,3: There is no difference with regard to the previous test.

B3,4: The controller is tested on the real system multiple times.

C3,4: Testing the controller 10 times shows that there is no oscillation at the switch anymore.

Step 3: The feature behaves as desired. The feature is successful implemented together with the previous feature. In figure 4.3 all different detail steps are shown, and in figure **??** the cycles are shown.

Fast circular movement

The final feature is the faster circular movement of the masses. Initially the fast movement was defined as an angular velocity above the 30 rad/s. However, in the development of the slow movement feature an angular velocity of only 8 rad/s was realized. Thus, the fast movement will be redefined and the first goal will be 16 rad/s. Since the behavior of the two sliders start to differ for higher velocities the controllers of the two sliders will separately developed with PID-tuning.

Step 1:

For the fast movement feature it will be assumed that the mass has the correct initial position. A model is used where there is only a sine/cosine movement without homing.
Test: To test the feature the masses will be given the initial position of $x_{m1} = rsin(0) = 0$ and $x_{m2} = rcos(0) = r$. Eventually the masses have to have a stable movement at 16 rad/s, and it has to maintain its center of movement. When the test succeeds, the controller will be tested on higher velocities.

Motor1
Step 2:

A1: The proportional gain of this controller is made less aggressive, to decrease the overshoot. The derivative gain is made a little bit higher than the controller for slow movement. To create a stable offset, the integration gain is increased a little bit.

B1,1: The controller is tested on the model with motor limits.

C1,1: The model behaves as desired.

B1,2; C1,2; B1,3; C1,3: are executed respectively without any significant change.

B1,4: The controller is tested on the real system.

C1,4: The movement is stable, however after approximately 40 seconds the center starts to shift again.

A2: The proportional gain is increased to 1.2 for some more precision.

B2,1; C2,1:B2,2; C2,2; B2,3; C2,3: The simulations show a behavior as desired.

B2,4: The controller is tested on the real system.

C2,4: The movement of the mass maintains its center 10 seconds longer, after that it starts to shift again.

Multiple cycles are made with successful results in the simulation but with unsuccessful results at testing on the physical system. Redesigning the PID-controller did not result in any improvements on the physical system, which indicates that the error in the design has to be found somewhere else. This will be discussed at the end of this chapter.

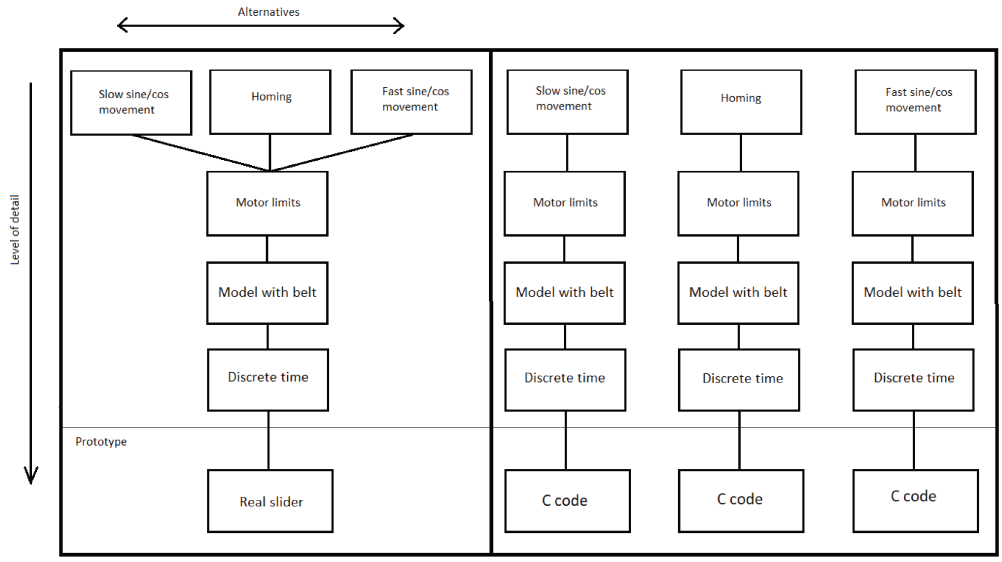

Figure 4.3: The model structure with all the separate features. These features can all be simulated together or individually at every detail level

4.2.1 Final design

The final design of the controller can be seen in figure 4.4. Like in figure 3.4 the design is divided into four parts, with the in and output part that is connected to motor 1 and 2. In the most left block a signal input ,virtual switch, can be seen. When this signal turns from 0 into 1 the system switches from the initial position setpoint the the sine cosine movement.

The two outputs of the switches from the sliders are connected with the setpoint blocks, but they are also connected to a switch called reset 1 and reset 2. When these reset switches are switched, the block after the reset has a constant output which is equal to the distance the mass has moved in the direction of the slider. This output is then subtracted from the measured position such that the position at the switch is defined as 0.

setpointM1:

The design of block setpointM1 is shown in figure 4.5, here M1 refers to motor 1 or slider 1. Five switches can be seen in the design, where switch 1 and 4 are break contacts and switch 2, 3 and 5 are make contacts. The switch-in is connected to the sliderswitch1 of figure 4.4,and the switchsine is connected to the virtualswitch of figure 4.4. Before the switch on the slider is switched, the setpoint of figure 4.5 has the value of Cspeedm11. This Cspeedm11 has a positive slope, which lets the mass of slider 1 move into the direction of the switch. When the mass touches the switch on the rail, switch1 is switched off, and switch 2 and 3 are switched on. From the moment switch 2 and 3 are switched on, a constant is integrated which results in a slope. A constant and an integral are used, since using a slope as input like Cspeedm11 would give the problem that the slope has to start rising exactly on the moment that the mass touches the switch. A small constant is added to the slope such that the mass has a quick movement away from the switch, this ensures that the mass does not stick at the switch. The signal limit, PositionM1, ensures that the slope stops at the initial position of the mass. After the mass has reached the initial position, the virtualswitch is set form 0 to 1 and the setpoint turns into a sine. The switchsin output is connected to the PID block. The diagram of SetpointM2 is exaclty the same except that the sine block is replaced by a cosine block. The two masses do not move to their initial position at the same time, but the masses move to their initial position after each other. This is realized by simply set a begin time for the second mass such that the first mass has

enough time to reach its initial position. This is of course done to prevent a collision between the two masses.

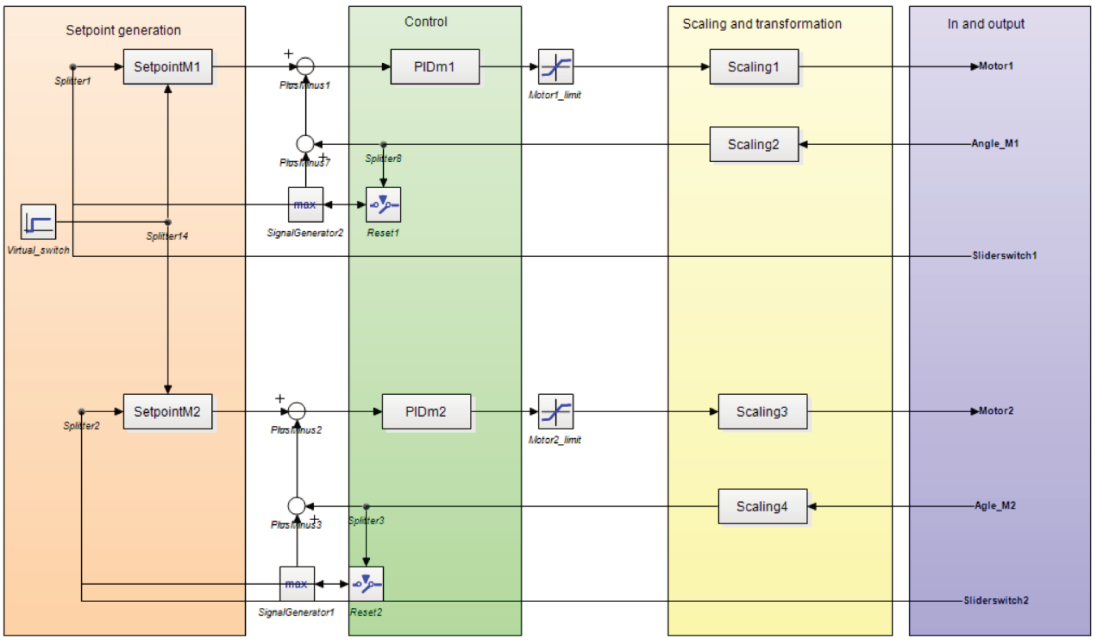

Figure 4.4: Final design of the controller

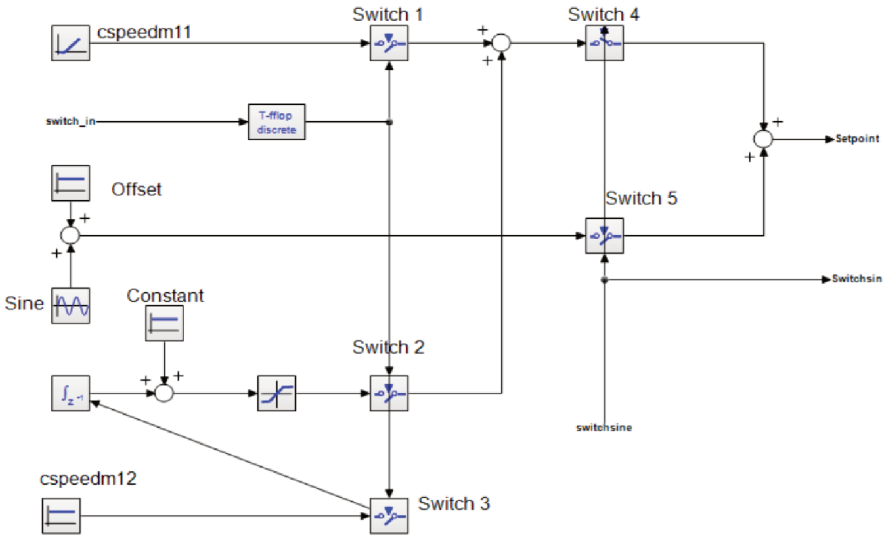

Figure 4.5: Block diagram of setpointM1

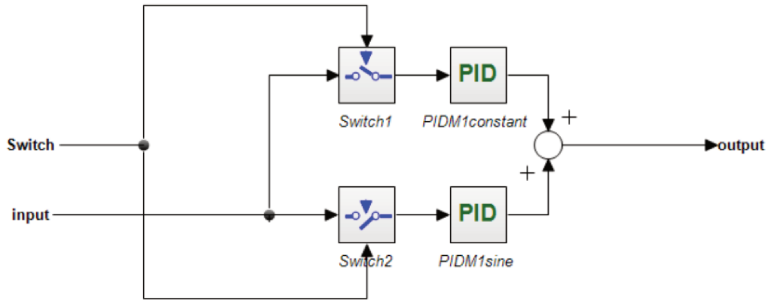

Figure 4.6: Block diagram of block PIDm1

PIDM1:

The next block is block PIDm1, the block diagram of this part is shown in figure 4.6. There are two inputs and one output. The input "Switch" is connected to the virtual switch, which sets the homing procedure to the circular movement. When this switch is set from 0 to 1 the switch before the PIDM1constant breaks, and the switch before PIDM1sine is turned on. This is because the homing needs a different of PID-controller than the circular movement. The "input" is connected to the output of the plusminus1 of figure 4.4.

Scaling:

At last there is the scaling and transformation. The scaling1 block only multiplies the input with constant, to realize the correct voltage on the prototype. From measurements it determined that a constant of 0.071 results in the most accurate input voltage. The block Scaling2 is represented in figure 4.7. This block diagram contains the new angle encoder, together with the a constant equal to 2π, and one constant equal to the radius in cm to obtain the position.

Figure 4.7: Block diagram of block Scaling2

4.3 Discussion

It seems that the core of the problem is that the measurement of the position does not match with the position which is observed. It is observed that the center of the sine and cosine movements shift, while this is not detected in the measurement, as mentioned earlier. The same behavior of this shifting is neither observed in the simulation. This strongly suggests that an important detail regarding the angle sensors is left out. The method of encoding the angle, explained at the end of section 3, is not taken into account in the simulations. However, it was noticeable that at every rotation there is a small dip in the measurement plot of the position. differentiating the output of the angle sensor causes a very high positive or negative number, depending on the direction of rotation. This number is limited, by a limit which is higher than the maximum angular velocity. The number that comes out the signal limiter is significantly smaller than the huge number at every end of rotation, however this output of the signal limiter has still a deviation from the correct angular velocity. Even though this error is small, the error builds up after every rotation. This is possibly the cause of the discrepancy between the angle measurement and the actual angle.

Implementing the original angle encoder in the model together with the sensor limitation, and simulating the model for a sine with an amplitude of 3 cm and an angular velocity of 20 rad/s indicates that there is indeed a shift in the movement. Whereas the output of the angle encoder does not indicate a shift. This is exactly what is observed during the tests on the physical system. This simulation clearly points out that the angle encoder is the cause of the shifting center of the sine and cosine movement.

A different angle encoder is used to eliminate the error. This time code is used to obtain the total angle. The code works according to the following steps:

1. If (inputsample-previous_inputsample)$>$ 0.5 then: rotation = rotations - 1.
2. If (inputsample-previous_inputsample)$<$ -0.5 then: rotation = rotations + 1.
3. Output = Input + rotations.

This encoder gives the correct total angle without small errors for every rotation. The original angle encoder is replaced by the new encoder and the same simulation is done. The simulation shows that the new encoder does not cause any shift. Applying the new design to the prototype with an angular velocity of 20 rad/s proves that the new encoder works and the original encoder was the cause of the shift. There is no shift observed, and in addition to that the jerky behavior that was mentioned before also has decreased significantly. It is possible that the small dips of the previous encoder caused this jerky behavior.

The implementation of the limit of the angle sensor result in an extra model. This means a that the new set of models looks like the schematic in figure 4.8

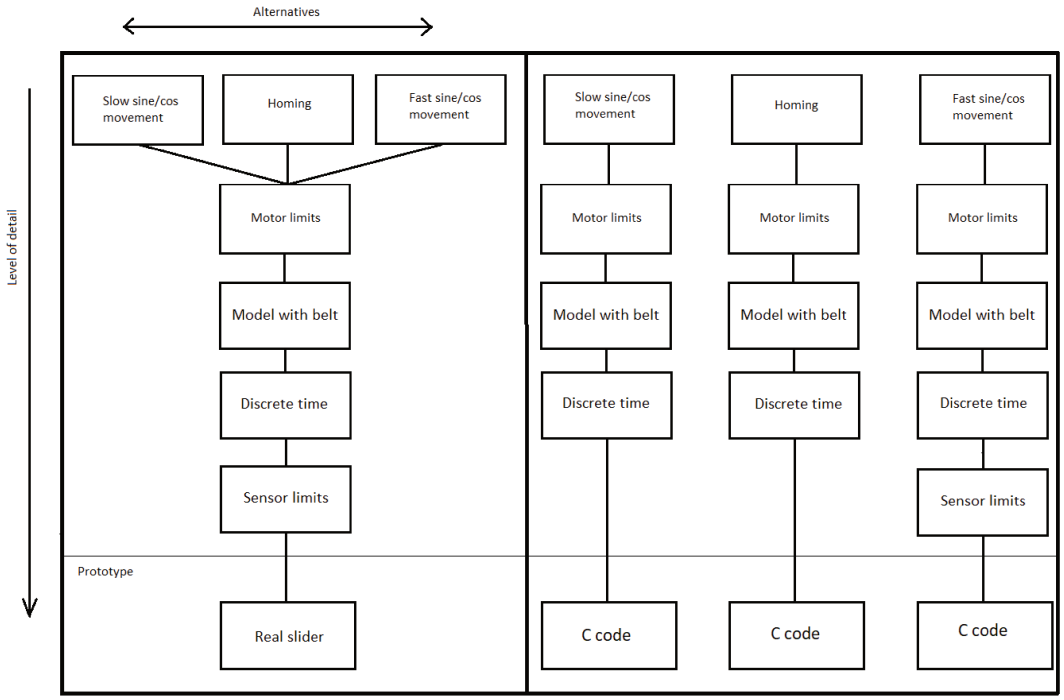

Figure 4.8: New set of models after the implementation of the new detail aspect

5 Evaluation

In this section the application of the method will be evaluated, and the method itself will be evaluated.

The application of the method to the development of the controller resulted in a controller that works for relative low angular velocities. To evaluate the method on its efficiency different steps will be judged on their relevance, and alternative steps or order of steps will be considered and compared to the method for this particular design.

First feature

An essential characteristic of this method is that detail is added after every successful test. Looking at the process of the first feature it is clear that most of the designs turned out to be insufficient at the test with the real system. The designs in these tests turned out to be insufficient because of the shift in the offset in the sine wave, however this effect was not indicated by the simulations. With this knowledge it can be argued that in repairing the error of this shift, it was redundant to simulate the redesigns. Since, it would be impossible to determine the effect of the adjustment of the controller on the shift of the offset based on the simulation, the necessity of these simulations is questionable. However, this is clearly due to the fact that the richest model used in this development did not fully describe the behaviour of the prototype.

Although the seemingly unnecessary simulations is not an argument against the method in this case as has been explained, there remains still one remark regarding efficiency. Comparing the tests of $B_{2,1}$ and $B_{2,2}$ there seems to be no significant difference. If one has the insight that this will be the case for the next design in A3, it could be decided to immediately add the features that are not likely to cause errors. Applying this idea to the development of this feature would result in a matrix with smaller sized columns than the matrix of feature 1 in figure 5.1:

$$\begin{bmatrix} B_{1,1} & B_{2,1} & B_{3,1} & B_{4,1} \\ & B_{2,2} & B_{3,2} & B_{4,2} \\ & & & \end{bmatrix}$$

Second feature

For this feature the same applies as for the first feature. At the development of this feature, errors where only detected at testing the design on the prototype. Like in the first feature, this was due to the large gap between the prototype and the model at the highest detail level.

Third feature

The initial problem of this feature was initially too, that the model deviated too much from the prototype to detect errors at the simulations. However in a rather late stage of the process it became apparent where the model fell short compared to the prototype. An important detail was left out in the iteration process, which brings up the question whether the method was executed in the appropriate way. One may say that no conclusions can be drawn when the method is not executed appropriately. however ending up with an inefficient process while leaving out a crucial step of the method, can be seen as an argument in favor of the method.

$$Slow\ sine/cosine\ movement \qquad InitialPosition \qquad Fast\ sine/cosine\ movement$$

$$\begin{bmatrix} B_{1,1} & B_{2,1} & B_{3,1} & B_{4,1} \\ & B_{2,2} & B_{3,2} & B_{4,2} \\ & B_{2,3} & B_{3,3} & B_{4,3} \\ & B_{2,4} & B_{3,4} & B_{4,4} \end{bmatrix} \qquad \begin{bmatrix} B_{1,1} & B_{2,1} & B_{3,1} \\ B_{1,2} & B_{2,2} & B_{3,2} \\ B_{1,3} & B_{2,3} & B_{3,3} \\ B_{1,4} & B_{2,4} & B_{3,4} \end{bmatrix} \qquad \begin{bmatrix} B_{1,1} & B_{2,1} & B_{3,1} & B_{4,1} & \dots \\ B_{1,1} & B_{2,2} & B_{3,2} & B_{4,2} & \dots \\ B_{1,1} & B_{2,3} & B_{3,3} & B_{4,3} & \dots \\ B_{1,1} & B_{2,4} & B_{3,4} & B_{4,4} & \dots \end{bmatrix}$$

Figure 5.1: Matrices representing the B steps of the different features

6 Discussion

In testing the structured method, the assumption was made that there would not be a major deviation from the structured method while applying it to the control problem. In order to draw a useful conclusion of the method it is of course preferable to stay as close to the method as possible during the development of the control software. In this case it could be said that the deviation from the method was too big to have a good judgment. However, conclusions can also be drawn by looking at the effects of the unintentional deviation from the method.

Lets have a closer look on what went wrong, and where the implementation of the method actually went wrong. Besides the mistake that was made at the very beginning of the implementation by leaving out important detail, there also was a crucial misstep during the iteration process. At the end of step C4,4 of the first feature, it was decided to stop the development of the feature at an achieved angular velocity which was significantly lower than the aim of 30 rad/s. It was not necessarily a big mistake to stop at a less result than was intended. However, it was the combination of this decision together with the fact that the matrix of this feature indicated that there was something wrong. A matrix with all the B steps, which has a large amount of steps Q for every column, is an indication for a bad implementation of the method. up to this point only redesigns of the PID-controller where considered, with a maximum column size for almost each unsuccessful test. But after a significant amount of redesign cycles with maximum sized columns, it would have been better to reconsider the implementation of the method. Instead, the choice was made to proceed to the next feature, with the same implementation of the method. Although the second feature was developed successfully, this choice resulted in a significant waste of time when developing the third feature.

Apart from the exclusion of detail during the implementation of the method, there was another aberration with regard to the structured method. Although, this did not cause any complications, it is worth to discuss in order to evaluate the structured method. According to the structured method, a feature has to be developed based on the models of the physical system and the models of all previous features. However, the development of the homing feature was not based on its previous feature. This was done because, as described earlier, the two features are independent of each other. The first feature could operate without the second feature, and the second feature could operate without the first feature. In this view it does not make any sense to test the homing feature on the model of the physical system with the model of the previous feature. Because of the independence, testing the homing feature on the model of the physical system alone was enough. Actually the same applies for the third with regard to the second feature.The second and third feature are independent just like the thirst and second feature.

The third feature is in some way built upon the first feature, however there is a difference compared to what the structured method describes. Of course one may say that the feature of the fast circular movement can only be developed after the development of the slow circular movement. However, the third feature is not really built upon the first feature, but the third feature is more an adaption or improvement of the first feature. So the difference is that the first feature is replaced by a better version, Whereas the structured method says that a feature should be added to the previous features.

7 conclusion and recommendations

7.1 Conclusion

In this bachelor assignment a the structured method was used to develop control software for a slider setup. At first the control software was developed successfully with the method up to a certain angular velocity. At the final stage of the development it was identified that the method was not optimally implemented, because an important detail layer was left out. Implementing this detail layer resulted in a redesign which worked for higher angular velocities up to at least 20 rad/s. Although the implementation of the method did not exactly go as desired, still some conclusions can be drawn with regard to the research questions.

Was it in any stage of the development more beneficial to the efficiency to deviate from the method?

As mentioned in the evaluation of feature 1, it may have been more efficient to simulate the effects of detail with little effect together. In this case it would have been more beneficial to even simulate all the detail together, since almost no error was detected in the simulation. However, it cannot be said that this deviation would be beneficial for other projects, since in this case the model lacked detail.

There has unintentionally been deviated from the method, in the sense that an important detail was left out in the process. This mistake in implementing the method shows that identifying and ordering the significant detail of the model is key for a correct implementation.

In the discussion it is also mentioned that the testing of the features was a deviation from the method. During the development of the features, the features were tested on the physical model without the implementation of the previous features. However, it would not have made a difference in the result of the design when this testing was done with the implementation of the previous features.

Is it likely that the same design errors which were identifying using the method, would also have been identified without using this method?

Regarding the idea to simulate detail features which have little effect, it would have been likely that the same errors would have been spotted. At no point in the iteration process errors where spotted with the simulation of the belt and the simulation of the discrete controller.

For what the exclusion of the detail of the angle sensor concerns, it is clear that proceeding with this deviation would not have resulted in identifying the error of the angle encoding.

Deviating in testing the features did not have any effect on identifying the errors.

If these errors were identified without the method, would it have cost more or less time to fix them without using the method?

Because of the incomplete model that was used in the simulation, errors where almost only detected at testing the design on the physical model. Although errors where detected, the cause of the error remained unclear because of the incomplete model. Thus, using this deviation of the method cost more time to fix the error than the time it would have cost to fix the error when the method was applied in the proper way.

How applicable is the structured method for the design of embedded control software of a cyber-physical system?

The alternative steps with regard to the structured method in this project showed that deviating from the method had a negative result on the efficiency of the development of the control software. This also showed that it is required to have a thoroughly identification of the details of the

model to make the method applicable. Correcting the mistake that was made at the beginning eventually resulted in a successful design, which indicates that the method was indeed applicable for this particular method. For a more general conclusion about the structured method, more tests have to be done by developing cyber-pysical systems with the structured method.

7.2 Recommendations

After an attempt to apply the structured method, recommendations can be made to increase the success of the method.

- **Simulation of multiple details at once**

  As was the case in this the development during this project, it can become clear after a number of inner cycles that some additions in detail do not have a significant influence on the outcome of the simulation. In that case it is more efficient to identify these details, and simulate them together at once, instead of add and simulate them one at the time. In case there is an error in the simulation of the combined detail, one can strip away detail that most likely is involved in the error. When it is clear what design error caused the problem, redesign the feature.

- **Evaluation on different levels**

  In the discussion was described how a mistake was made by proceeding with a bad implementation of the method. In figure 2.1 it can be seen that the method suggest an evaluation at the end of the development of the feature, however a small evaluation can already been made after every inner cycle. After every unsuccessful test, a column is added to the matrix which represents the steps B for p and q. A small evaluation can be made after every unsuccessful test, by evaluating the matrix. If the size of the columns are relatively high, the implementation of the method should be reconsidered. Because, either the order of detail is not ideal or the most complete model lacks important detail. This prevents the risk that the next features are developed too using the less effective implantation of the method. So, changing the implementation at an early stage, if necessary, can save a significant amount of time.

  So, this implies that evaluations can be done on at least two different levels. After every inner cycle an evaluation can be made to reflect on the collection of columns. The reason for this evaluation is to check whether the method is implemented correctly. If the method is not implemented correctly, it can be decided to break the iteration loop and redesign the implementation.

  Evaluations can also be done after the development of each feature, so after each outer cycle. Even if the feature is designed successfully, one can still come to the conclusion that the implementation of the method was not optimal. In that case the implementation can be redesigned, which can save time in the development of the next features.

- **Independent features**

  Often the total behavior of the system, that is to say what the physical system has to do, can be subdivided into different tasks. It is evident to define these different tasks as different features. And as was the case in this project, these tasks or features could operate independently of each other. According to the structured method features should be tested on the model of the physical system with their previous features. However, when features can operate independently, this is not always necessary. However, if the definition of a feature should hold that it must be built upon another feature, then it could also be decided that independent features will be defined as sub-features. The combination of these sub-features can then be defined as a main feature.

A Demonstration instructions

The following steps describe a brief demonstration on how to use the setup.

1. Power the Rapsberry Pi of the setup, and connect the Pi with the laptop via the ethernet cable.
2. Open the 20-sim file. In the simulation model choose Tools->Real time toolbox -> C Code Generation. Choose the submodel called "control" and open with 20-sim 4C.
3. Select the target "Raspberry Pi Setup", after this it should automatically discover the setup. If not, reboot the Raspberry Pi.
4. In the connection I/O window set up the following connections:
   Input
   - Modelportname:Angle_M1 Targetportname: Pin: 1
   - Modelportname:Angle_M1 Targetportname: Pin: 5
   - Modelportname:Sliderswitch1 Targetportname: PIN 4 Pin: 4
   - Modelportname:Sliderswitch2 Targetportname: PIN 18 Pin: 18

   Output
   - Modelportname:Motor1 Targetportname:PWM output Pin: 2
   - Modelportname:Motor2 Targetportname:PWM output 2 Pin: 15
5. Before the starting the controller, set the frequency to 5khz. Furthermore make sure that the mass of slider 1 is not in the trajectory of the mass of slider 2.
6. The start time of the virtual switch mus have a value $t_0 = \frac{2k\pi}{\omega}$ with k as integer. This makes sure that the start position of the sine movement corresponds with the initial position of the homing. When the user plans to connect the two mass with the string, it is recommended to increase the k such that the start time of the virtual switch is above 60 seconds. This must give the user enough time to connect the string.
7. Press run to initiate the process.

B Parameters measurements setup

First the two electrical parameters will be determined, namely the resistance and inductance. The circuit in the electric motor behaves as an RL-circuit in series when the axle of the motor is clamped. The current through the circuit can be described according to equation B.1, with input voltage V.

$$i(t) = \frac{V}{R}(1 - e^{-\frac{R}{L}t}) \tag{B.1}$$

The current will be measured with the myDAQ when a constant voltage is applied to the setup while clamping the axle. From the current measurement the steady state current and the time constant can be determined. From this the resistance and the inductance can be derived.

Next three more parameters have to be determined. The motor constant, the total mechanical resistance, and the total inertia. The motor constant can be determined by measuring the current and the angular velocity. The torque and angular velocity can be described with equation B.2 and B.3, with motor constant k_m.

$$\tau = k_m I_{motor} \tag{B.2}$$

$$\omega = k_m V_{motor} \tag{B.3}$$

Here the motor voltage V_{motor} is equal to the input voltage V_{in} minus the steady state voltage over the resistor V_R.This brings us to equation B.4 for determining the motor constant. Note that this equation only holds for the steady state of the system, with a constant input voltage. Doing multiple measurements with different input voltages, should give the motor constant with good accuracy.

$$k_m = \frac{V_{in} - R_e I_{motor}}{\omega} \tag{B.4}$$

Next the total mechanical resistance and the total inertia will be determined. The total inertia I_t and the total mechanical resistanceR_m are connected to a one-junction with a torque as input. Like the 1-junction in the electrical domain, the flow of the junction in the mechanical domain can be described in the same manner, see equation B.5.

$$\omega(t) = \frac{\tau(t)}{R_m}(1 - e^{-\frac{R_m}{I_t}t}) \tag{B.5}$$

Combining equation B.5 with equation B.2, equation B.6 follows.

$$\omega(t) = \frac{k_m i(t)}{R_m}(1 - e^{-\frac{R_m}{I_t}t}) \tag{B.6}$$

By measuring the angular velocity and the steady state current, the mechanical resistance can be determined by looking at the steady state value of the angular velocity. After that the total inertia can be calculated by determining the time constant of the equation.
Measurements parameters model

The motor constant electrical resistance and mechanical resistance are determined according to section 3.1.1. The results of the two motors can be found in table B.1,B.2,B.3 and B.4. For the

electrical resistance, nine measurements have been done for each motor, and for the motor constant and mechanical resistance six.

still shows that the method is applicable to the method.

Input voltage (V)	**Measured resistance** (Ω)
3.2	9.6
4	11.7
4.8	7.6
5.3	9.7
5.8	9.5
6.3	9.3
7.1	8.1
7.4	9.8
7.9	9.6
average	9.4

Table B.1: Electrical resistance motor 1

Input voltage (V)	**Measured resistance** (Ω)
3.4	12.0
3.7	10.4
4.8	11.7
4.8	11.0
5.1	10.9
5.7	10.0
6.1	8.3
7.0	8.0
7.3	9.8
average	10.2

Table B.2: Electrical resistance motor 2

Input voltage (V)	**Measured resistance** (Ω)	Mechanical resistance($\frac{Ns}{m}$)
4.3	0.015	0.00044
4.2	0.042	0.00047
3.8	0.050	0.00040
3.8	0.048	0.00058
4.4	0.044	0.00029
4.2	0.037	0.00029
4.0	0.040	0.00037
Average	0.040	0.00040

Table B.3: Motor constant an mechanical resistance motor 1

Input voltage (V)	**Measured resistance** (Ω)	Mechanical resistance($\frac{Ns}{m}$)
3.8	0.052	6.7E-6
3.3	0.045	1.4E-5
3.5	0.050	8.6E-6
3.56	0.049	8.0E-6
3.8	0.052	8.3E-6
3.25	0.44	1.1E-5
3.6	0.048	1.0E-5
Average	0.049	10E-6

Table B.4: Motor constant an mechanical resistance motor 2

Bibliography

Broenink, J. and T. Broenink (2019), RAPID DEVELOPMENT OF EMBEDDED CONTROL SOFTWARE USING VARIABLE-DETAIL MODELLING AND MODEL-TO-CODE TRANSFORMATION.