Modelling and Controlling the Kinetic and Dynamic of a Bicycle

C. Tapken [(1)], P.Gloesekoetter [(1)] , A. Olivares [(3)] , G. Olivares [(2),]

[(1)] Fachbereich Elektrotechnik und Informatik.
Münster University, Germany.
[(2)] Departamento de Arquitectura y Tecnología de Computadores
Universidad de Granada. Granada, Spain
[(3).] Nazaries IT. Granada. Spain

ct500893@fh-muenster.de, peter.gloesekoetter@fh-muenster.de,
alberto.olivares@nazaries.com, gonzalo@ugr.es

Resumen. En este trabajo, se presenta el modelado y control posterior de una mini-bicicleta autónoma, que se utilizará para la enseñanza de sistemas de control. El modelo mecánico se construye inicialmente a partir de un diseño CAD y posteriormente se integra en Simulink, conjuntamente con los módulos de control. Se lleva a cabo el modelado del sistema completo, incluyendo las partes mecánicas, sensores, actuadores y la fricción de las ruedas con el suelo, consiguiéndose el mismo comportamiento que con la bicicleta física. A partir de las ecuaciones matemáticas que definen el comportamiento del sistema se diseña un controlador PID y un controlador LQG en el espacio de estados. Para verificar el modelo, los controladores diseñados se prueban también con los mismos parámetros en la mini-bicicleta física, obteniéndose un resultado idéntico.

Palabras Clave: Bicicleta autónoma**,** Control Digital, Control PID, control LQG, Simscape, Aprendizaje Experimental.

Abstract. In this paper, we present the modelling and subsequent control of an autonomous mini-bicycle, which will be used to teach control systems. The model is initially built from a CAD design and then integrated into Simulink, together with the control modules. The modelling of the complete system is carried out, including the mechanical parts, sensors, actuators and the friction of the wheels with the ground, achieving the same behavior as the physical bicycle. From the mathematical equations that define the behavior of the system, a PID controller and an LQG controller are designed in the state space. To verify the model, the designed controllers are tested, too. Identical parameters in the physical mini-bicycle lead to the same results in the model.

Keywords: Autonomous Bicycle, Digital-Control, PID controller, State space controller, LQG control, Simscape Multibody Simulation, Experimental Learning.

1 Introduction

The main aim of this paper is to design a bicycle model from a little existing physical bicycle, which can be used for education. Using a virtual model for teaching control systems has one main advantage; it is possible to test every controller and every configuration without damaging anything. In addition, there is the advantage that with a virtual model, a complete course with several students can experiment at a very low cost, and achieve the same learning experience than they would obtain with a physical system. Further on, teaching at home and distance learning capabilities can be mentioned. A bicycle is a good example to learn to control a system, because it is a very popular and well-known vehicle. Therefore, many people have an idea of how to control it. But it is not so easy to design a controller for this system, because it has no stable state of equilibrium and mathematical equations change with the speed of the bicycle. Due to the increased computing power of the past years, it makes sense to represent this model in a graphical fashion; this is called “multibody simulation”. This graphical simulation method has the advantage to verify the function of the controller intuitively; since it is not necessary to read and understand diagrams. Further, no mathematical equations are needed during set-up; the model can be built by drawing a CAD model and passing it to the multibody simulation software. After the creation of this virtual mechanical model, it is required to build a controller, as well as to obtain the complete and runnable model to check its controllability.

In this paper, it is shown how to set up the mathematical equations which are required to build a PID controller and a state space controller for this model. To verify the results, the controllers are verified with the physical bicycle. At the end of this paper it is shown how to develop the program for the physical bicycle and how to implement this program on an Arduino card. Therefore, it is also shown how to read all the sensors and how to control all the actuators. The bicycle model can be used to raise interest of the students, because it is not artificially made up and has direct practical reference. Additionally, it can also serve as the basis for the future construction of a real-size autonomous bicycle.

2 Physical bicycle

At first, the physical mini-bicycle is described [1]. It is a simple model, on which all the necessary parts are mounted so that it can drive autonomously. It has a motor for the driving, a servo for the steering, an encoder for controlling the speed, a sensor for measuring the inclination, an accumulator and an Arduino One card for the calculations. The already existing Arduino shield is necessary, since the Arduino board cannot drive the motor on its own. For reference: the distance between the axes is approximately 165 mm. Figure 1 shows a photograph of the model.

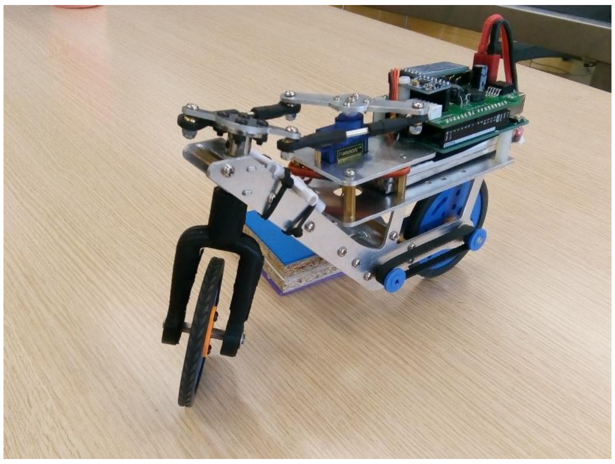

Figure 1. Physical model of the bicycle

2.1 Driving motor

In the physical model, a series DC motor with a gearbox is used. The gearbox is plugged directly to the motor shaft and on the other side of this motor there is an encoder, which will be described later in the article. It is a small motor that provides a mechanical power of about 0.3 watts only, due to the used battery voltage. The motor is mounted between the two wheels and the power transmission is handled by a timing belt.

2.2 Servo for the steering

The physical model uses a servo for steering. A servo is a combination of a motor, a gearbox, and the necessary control electronics. This kind of motor is used for example for model making, in automotive construction and in hard drives. In the physical bike a small one with a plastic gear is used. Furthermore, it is controlled with a pulse width modulation and the position can be directly specified with this signal while its integrated electronics regulate the position. The servo is mounted on top of the bike and it is connected to the handlebar with an adapter.

2.3 Arduino

Arduino is an open source hardware and software for the construction of digital devices and interactive objects [2]. The physical bicycle uses an "Arduino Uno R3" card, with an "ATmega328P device". It is a microcontroller with clock speed of 16 MHz. In addition, this board contains six pins for analog input and 14 for digital input signals. Six of them can provide a PWM output. Because the output current is limited to 20 mA, a driver is needed; at the bicycle, a “TB6612FNG” device is used [3]. This is a “H-Bridge driver” consisting out of four transistors; it can be used to drive the motor in both directions and the speed can be controlled by pulse width modulation. This chip is mounted on an Arduino shield attached directly on the Arduino Uno card. In addition, there is a voltage transformer, a switch for the battery and some connector for the sensors and the motors on the shield.

2.4 Encoder

The physical bicycle uses an encoder which is mounted directly on the motor shaft. It consists of a magnetic disc and two Hall sensors; the sensor can be seen below.

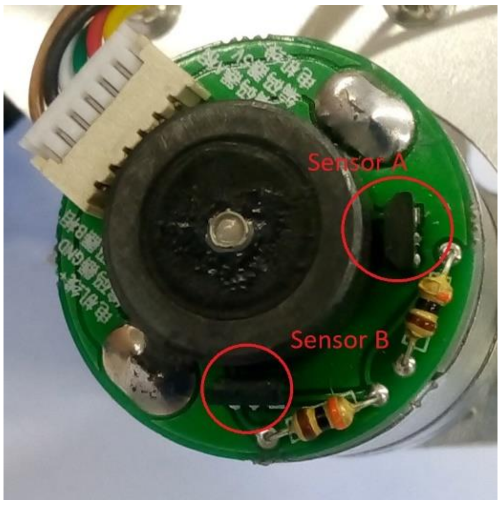

Figure 2. Encoder.

Because of the twisted mounting of the two Hall sensors it is possible to detect the direction of rotation. The sensors generate 13 pulses for a complete revolution, because there are several pairs of poles on the magnetic disk. The encoder delivers up to 1800 pulses per second in non-load operation mode.

2.4 Gyroscope and accelerometer

The physical bicycle uses a “MPU6050” integrated circuit [4] to measure the inclination. This is a six-axis motion tracking device that combines a three-axis accelerometer, a three-axis gyroscope and a microprocessor for the communication over I^{2}C bus. A gyroscope uses the principle of angular momentum to measure rotation rate of an axis and an accelerometer measures the inertia caused by the movement of masses during linear acceleration or deceleration. We make the fusion of both sensors to measure the inclination with sufficient precision.

3 Modelling process

The first step is the modelling process. To create a model that behaves in the same way as the physical model, it is necessary to model not only the dimensions or the volume of the frame but also the motor, the servo and the friction. For the building processes, the “Autodesk Inventor Professional 2018” [5] software is used to design a CAD model. This model can be exported to a multibody simulation software, which can, in turn, simulate the graphical model. “MATLAB Simscape” [6], a toolbox included in “MATLAB Simulink” [7], is used for the latter purpose.

3.1 Bicycle model

Within the next sections, it is presented how to build the CAD model of bicycle frame and how to make it runnable inside of the multibody simulation software. Also, it is shown how to measure the inclination and the speed of the bicycle, because these values a required to build a controller.

3.1.1 CAD drawing of a simple bicycle model

At first, the CAD drawing of the bicycle must be built. This model consists of about 14 different parts and it is important to create every part with the same dimensions, the same mass and the same density as the physical bicycle to get the same behaviour. Because the CAD program calculates the mass from the density and the dimensions, it is not essential to measure the weight. This has the advantage that it is not necessary to disassemble the physical model. In “Inventor”, the connection tool can be used to add the joints. The designed CAD model is depicted in figure 3.

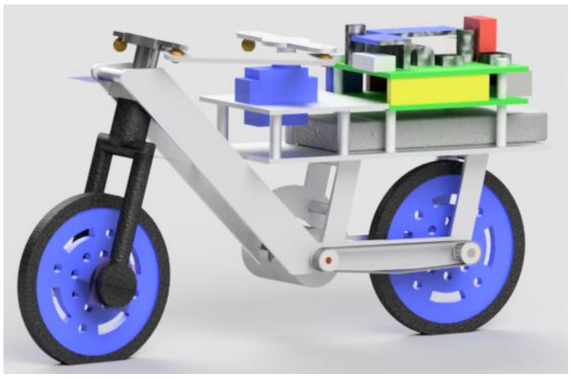

Figure 3. Inventor CAD model of the physical bike

In addition, a ground area must to be added so the bike can ride on it and the degrees of freedom are reduced to the normal use case of a bicycle.

3.1.2 Importing the model to Simulink

The bicycle model can be imported to “Simscape Multibody”, a toolbox from “MATLAB”, by using the “Simscape Multibody Link Inventor Plug-In” [8]. Because Simscape is a toolbox which makes part of Simulink, it can be used inside a Simulink model, as a simple block diagram. The resulting block diagram can be seen in figure 4.

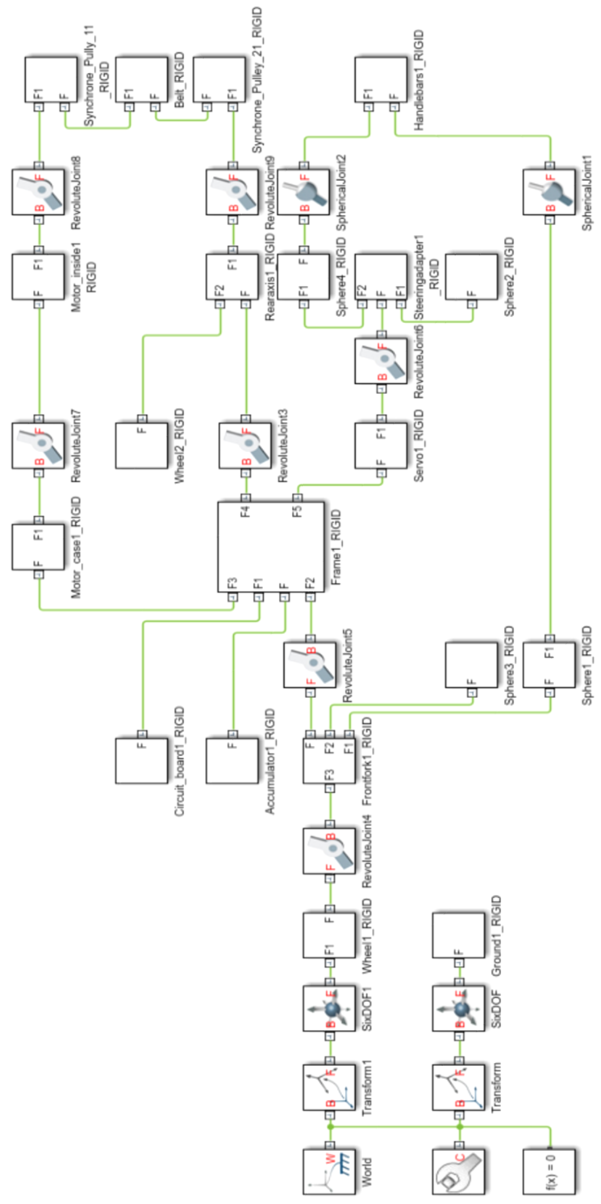

Figure 4. Simulink block diagram of the mini-bicycle model.

This model is runnable without any changes, but the problem with the connection of the wheel and the ground is still relevant, because MATLAB also has no joint for this connection. In this block diagram a joint with six degrees of freedom is used, which means that the bicycle can move in every direction and in every plane, which is not realistic. To fix the problem an additional library, called “Simscape Multibody Contact Forces Library” [9] can be employed. By using this library, it is possible to model every contact between a moving and a fix part; for example, the contact between a wheel and the ground can be modelled with the block “Spare to Plane Force”. Now, the model can move like a real bicycle. A still of the running model can be seen in figure 5.

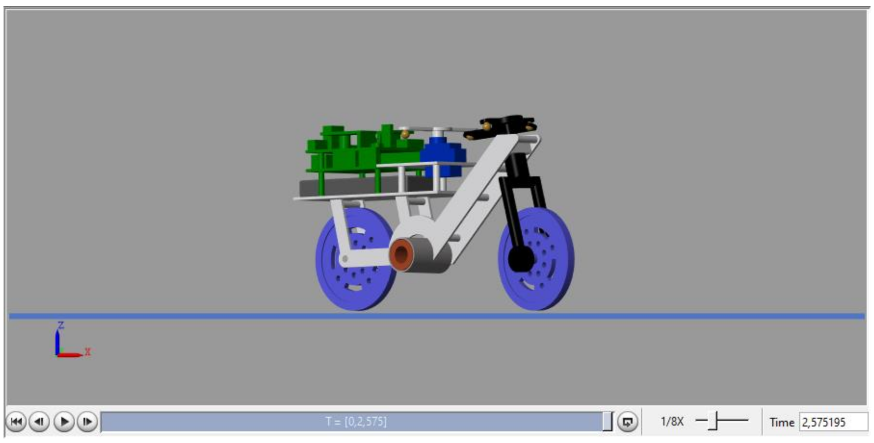

Figure 5. Simscape Multibody bicycle simple model.

The connection between the belt and the pulleys can be modelled with the CAD program, but Simscape interprets this connection as a rigid connection without movement. This is a problem since the movement of the belt is necessary, because it is the connection of the motor and the rear wheel. This problem is solved by manually building this connection with Simulink. After these manipulations, the model is runnable, except for the motor, which inclusion is discussed in detail in section 3.2.

3.1.3 Measuring the data

To control the bicycle model, it is important to measure the magnitudes which should be controlled, because these values must be returned to close the control loop. In this case, the necessary physical values are the speed of the bicycle and the inclination. Simscape has a “Transform Sensor” block, which can be used to measure physical values. This block measures the relationship between two frames. Additionally, it is possible to select which parameters should be measured. For example, the rotational and translational relationships, the velocities and the accelerations can be measured.

3.2 Driving motor

As described above, a series motor with a gearbox is used for the physical model. The following equivalent circuit diagram represents the behavior of the series motor in a simplified way [10], since for instance the brush voltage remains is constant.

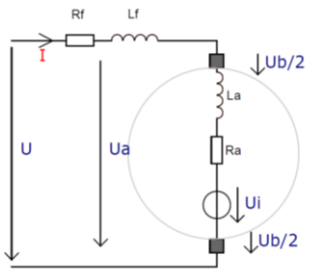

Figure 6. Simplified equivalent circuit diagram of series motor.

Figure 7 reveals the block diagram that has been deduced from the equivalent circuit diagram.

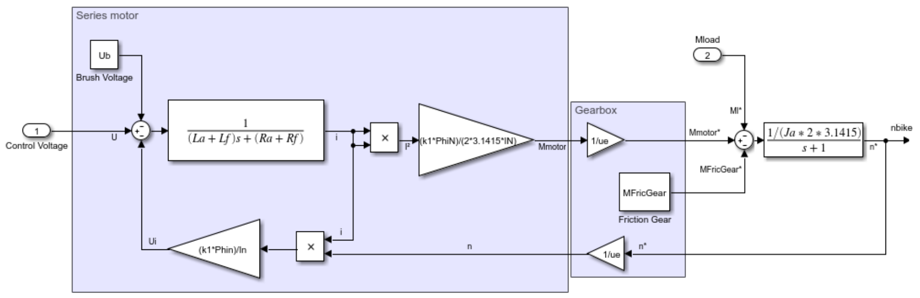

Figure 7. Block diagram complete series motor.

3.2.3 Determination of the motor parameters

Every model requires the determination of the involved parameters. Datasheets and specifications are a good initial source to set a value for the parameters, but often they are missing or incomplete. For the determination of the moment of inertia, we employed a retardation test. For this test, the engine is accelerated to unloaded top-speed. After that, the engine power is switched off and through the time depending speed drop, the moment of inertia can be determined. Finally, the inductance needs to be determined. Since the motor is controlled by pulse width modulation, the motor's performance depends highly on its inductance. The inductance can be derived by measuring the motor's impulse response. For this test, the engine must stand still and the supply voltage of the motor must be pulsed. In that case the equivalent circuit diagram of the motor simplifies to a series connection of a resistor and an inductance. The inductance can, then, be deduced with the help of an oscilloscope.

3.3 PWM

The speed controller of the physical bicycle is built with a pulse width modulation. To use the same speed controller for the virtual model and for the physical bicycle, it is also necessary to build a pulse width modulation. The block diagram is reflected on figure 8.

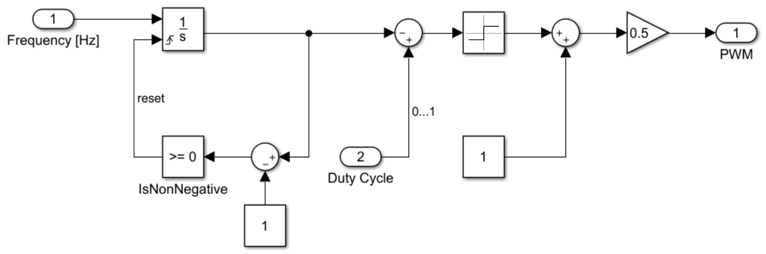

Figure 8. Block diagram of PWM modulation employed for speed control.

3.4 Steering servo

Because the servo position can be specified, and the electronics regulate the position, the servo can be modelled as a proportional element. However, no mechanical element can change the position without a time delay. As a first approximation, the servo is modelled as a "PT1 element" (a proportional element with a time constant). The servo has an important influence on the control, because it essentially determines the response time of the bicycle to the controller's output signals. One option is to determine the servo's delay by measuring its step response time. In our case, we measured the response time by taking a video from the motion. Then we derived the time constant during slow motion replay.

Once the servo delay is determined, all the parameters of the module are specified.

4 Controller design

It is necessary to design a control system because the bicycle is a system which has no stable state of equilibrium, i.e. without a controller this bicycle will always fall. It is also useful to set up a speed controller so that the bike can ride at a constant speed.

4.1 Theoretical foundations

To design a good controller, it is necessary to define the mathematical equations which compose it and which will, in turn, determine the value of the parameters.

4.1.1 Bicycle equations

The main difference between our bicycle and a conventional bicycle, ridden by humans, is that we can only use its inclination to maintain the balance, while humans can use their weight to change the balance point of the bike they are riding. This fact makes the necessary mathematics to be much simpler. Åström et al. [11] demonstrated that a very simple mathematical model can be used to express the behavior of a bicycle. This transfer function shows the mathematical connection between the inclination *φ(s)* and the steering angle *δ(s)*:

$$G_1(s) = \frac{\varphi(s)}{\delta(s)} = \frac{\frac{v \cdot a}{b \cdot h} s + \frac{v^2}{b \cdot h}}{s^2 - \frac{g}{h}}$$

Equation 1. Transfer function of the bike model.

Where g is the gravity acceleration, v is the bicycle speed and h is the height from de centre of the masses to the floor. The rest of parameters can be identified in Figure 9. In our case we suppose that λ =90º.

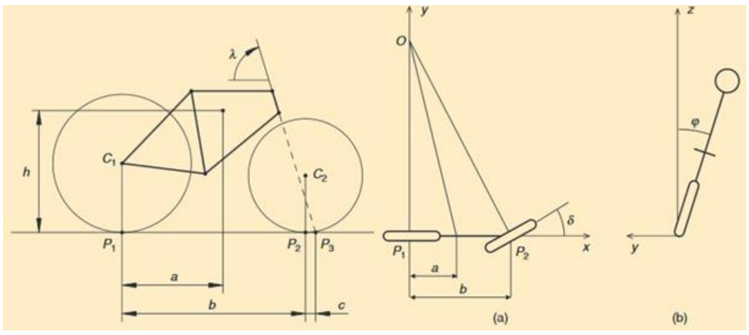

Figure 9. Bicycle geometry. Obtained from [2].

4.1.2 Servo equations

For the mathematical equation of the servo this first order transfer function from the modelling process can be used:

$$G_2(s) = \frac{K_{Servo}}{1 + T_{Servo} \cdot s}$$

Equation 2. Transfer function of the servo.

We obtained the gain K_{Servo} and the period T_{Servo} from the servo step response.

4.1.3 Steering

The servo is connected to the handlebar with an adapter, because of this, the angle of the servo isn't identical to the steering angle. Figure 10 illustrates the problem.

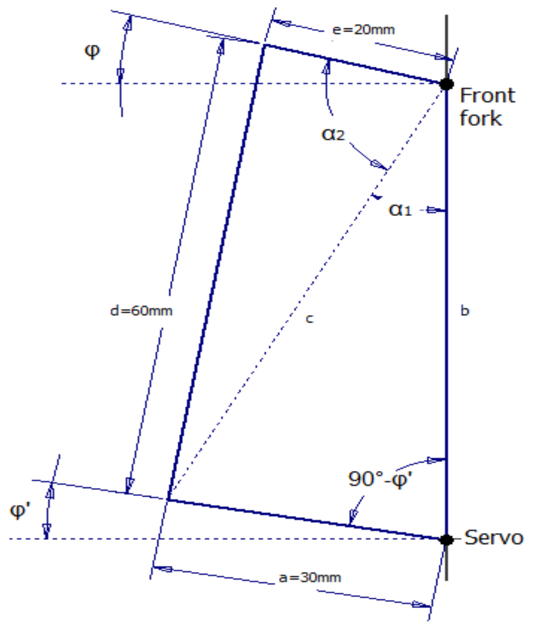

Figure 10. Adapter between servo and front fork.

By creating a diagram of the relationship between the angles, it can be shown that the steering adapter can be linearized as a proportional factor ($K_{steering}$).

4.2 PI controller for the velocity

A speed without large over or under swing is required, because the velocity has a big influence on the behaviour of the inclination control.

Initially it was necessary to obtain the parameters of the motor, and calculate its transfer function. Subsequently, the feedback loop is constructed with that transfer function and a discrete PI controller. The adjustment of the proportional and integral constants is done automatically with the "auto tune" option of the discrete PI module included in the Simulink library. A delay block to the closed loop is included, to consider the delay created by the calculation. Figure 11 shows the speed control feedback loop used to calculate PI controller parameters.

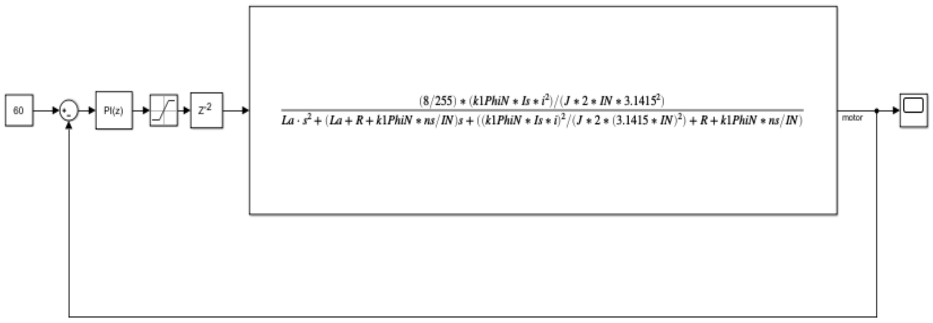
60
+
-
PI(z)
z^{-2}
$\frac{(8/255)*(k1PhiN*Is*i^2)/(J*2*IN*3.1415^2)}{La\cdot s^2+(La+R+k1PhiN*ns/IN)s+((k1PhiN*Is*i)^2/(J*2*(3.1415*IN)^2)+R+k1PhiN*ns/IN)}$
motor

Figure 11. Simulink control loop of the bicycle speed.

The designed PI controller can now be tested using the mechanical Simscape model. The resulting step response can be seen in figure 12.

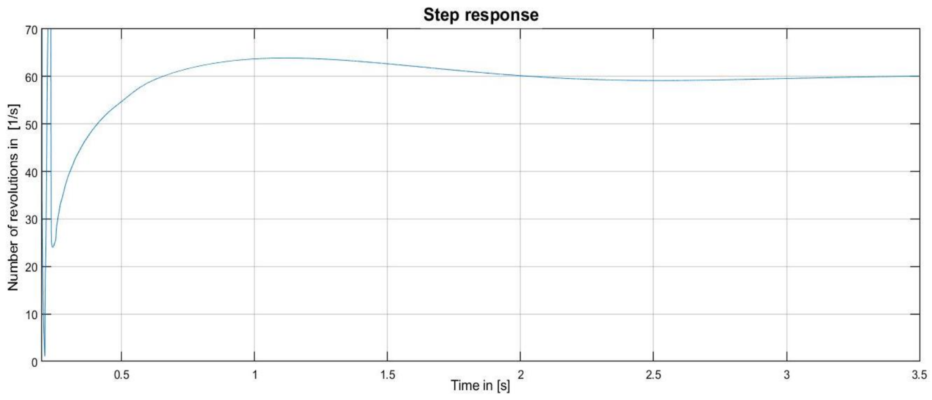

Figure 11. Step response of the driving motor.

4.3 PID controller for the inclination

Initially we designed a PID controller to regulate the inclination of the bicycle; for this, we have used the Simulink block diagram of figure 13. The numerical transfer functions of the steering servo and the bicycle are taken into account. Figure 14 shows the response of the inclination of the virtual bicycle with a 7 degrees inclination set point.

A video of the simulation can be seen in [12].

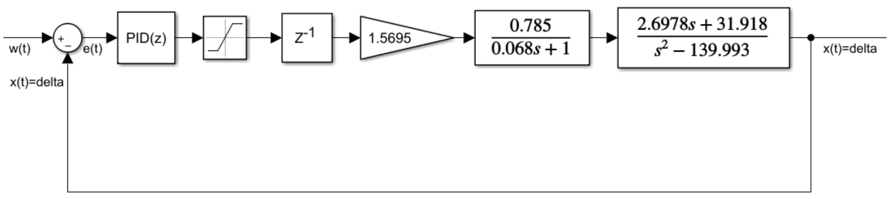

Figure 13. Feedback loop used for the calculation of the PID parameters for inclination control.

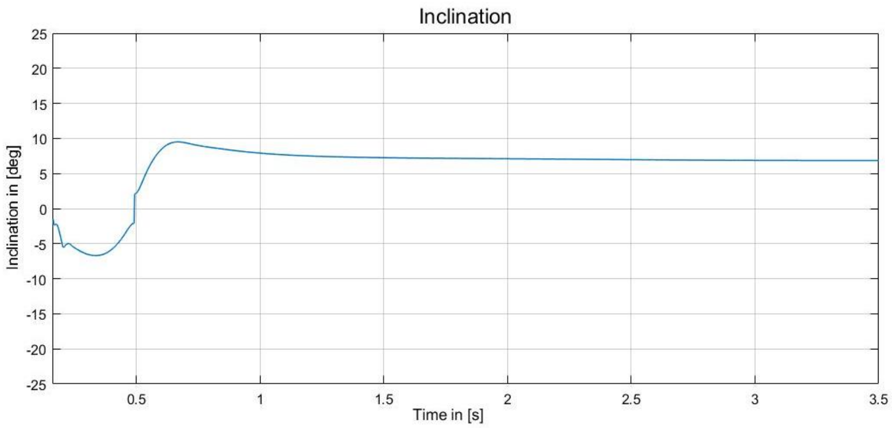

Figure 14. Inclination response for the virtual model of the bicycle for a set point of 7 degrees *(K_p=5.1; K_i=6.7; K_d=0.48).*

4.4 State space controller for the inclination

Furthermore, a state space controller is built. This controller is based on a state space representation and the states of the controlled system are estimated with an observer. These estimated states are the feedback variables of the controller. At first, a state space representation of the system to be controlled must be set up. The differential equation can be defined in an observable canonical form by using the transfer functions in equations 1 and 2 [13].

$$\underline{\dot{x}}(t) = \begin{bmatrix} 0 & 0 & \frac{g}{h \cdot T_{servo}} \\ 1 & 0 & \frac{g}{h} \\ 0 & 1 & -\frac{1}{T_{servo}} \end{bmatrix} \cdot \underline{x}(t) + \begin{bmatrix} \frac{K_{steering} \cdot K_{servo} \cdot v^2}{b \cdot h \cdot T_{servo}} \\ \frac{K_{steering} \cdot K_{servo} \cdot v \cdot a}{b \cdot h \cdot T_{servo}} \\ 0 \end{bmatrix} \cdot \underline{u}(t)$$

$$\underline{y}(t) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \underline{x}(t) + 0 \cdot \underline{u}(t)$$

Equation 3. Space state representation of the bicycle.

Now the controller can be designed with the state space representation above. Because the states of the system don't represent physical magnitudes that are conceivable, the state space controller is designed as a linear-quadratic-gaussian regulator (LQG), which is really formed by a Kalman observer and a LQR control with the feedback of the estimated states. This is useful, because the controller can be calculated automatically and no poles must be specified. The feedback constants have been calculated with Matlab, using the order ***lqr***, and testing with different values of the Q and R parameters. The Kalman filter block by Simulink is also used. It is also necessary to adjust the covariance matrices Q and R of this filter

This controller has been designed, again, as a time discrete controller so the Arduino is able to run it. The resulting block diagram from this controller and the step response can be seen in figures 15 and 16 respectively. In this case an initial alteration of the signal is appreciated, possibly due to the initial slip of the rear wheel, produced when trying to get the final speed very quickly.

A video of the simulation can also be seen under [14].

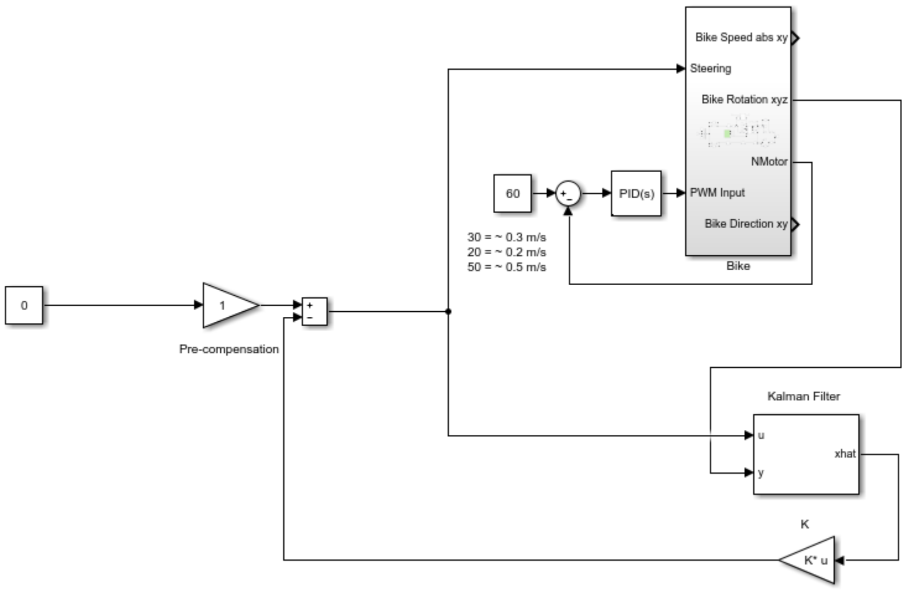

Figure 15. Block diagram of the LQG control.

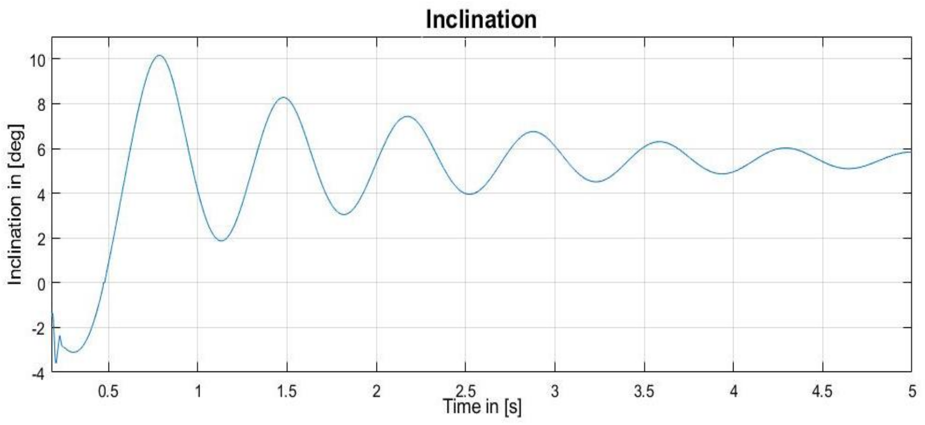

Figure 16. Diagram of the virtual bicycle inclination with the LQG controller (setup of inclination = 5 deg).

4.5 Comparison between the controllers

Now the controllers can be compared using the steering diagrams as a basis. The PID controller produces the best result, because the set point is reached earlier, but the first overshoots of both controllers are nearly equal, since they are due to the fact that initially the speed of the bicycle is not yet regulated (a minimum speed is required to maintain balance).

In addition, the disturbance reaction of the control loop is important; this can be determined by further simulations, but the resulting diagrams aren't shown here for the sake of clarity. Due to the D component, the PID controller is more aggressive than the state space controller, and as a result, the PID controller makes many more steering movements when a noise variable is switched on. However, there are no big differences in the inclination. This means that the disturbance reaction of the state space controller is a slightly better.

5 Controlling the physical bicycle

To complete this project, the designed controllers are ported to the Arduino card to test the controller set. This is useful to verify the model and the functional bicycle can be used to raise interest of the students, because it allows to show that this task is not only a constructed model without any practical reference.

5.1 Basics

It is possible to program the Arduino card directly from Simulink. This has the advantage that the designed controller can be used without big changes. However, the block diagram must be supplemented to read the data from the sensors and to set the actuating variables. The standard Arduino Simulink library has its limitations, though. For instance, it is not possible to read the data from the accelerometer and the gyroscope with the available blocks.

5.2 Reading the sensors

To read the data from the accelerometer and the gyroscope, the library “Rensselaer Arduino Support Package Library” [15] [16], can be used. This library is designed with blocks, which call a program in C code. The C code can include other C code; for example, exiting Arduino libraries. Furthermore, the library also contains a block to read encoders; however, this block computes pulses, so it can be used to calculate angles and not the speed. Some changes in the C code, which is called by the block, are used to solve this problem.

5.3 Filtering the accelerometer and gyroscope data

The physical bicycle uses an accelerometer and a gyroscope to measure the inclination, but the accelerometer is very noisy and the gyroscope can drift. A gyroscope uses the principle of angular momentum to accurately measure rotation rate about an axis, and an accelerometer measures the stresses caused by the movement of masses during linear acceleration or deceleration. We use the fusion of both sensors to measure the inclination with sufficient precision. To combine the sensors a complementary filter or a Kalman filter can be used. The instructions of the library which is mentioned in section 5.2, show that the following block diagram for the complementary filter can be used.

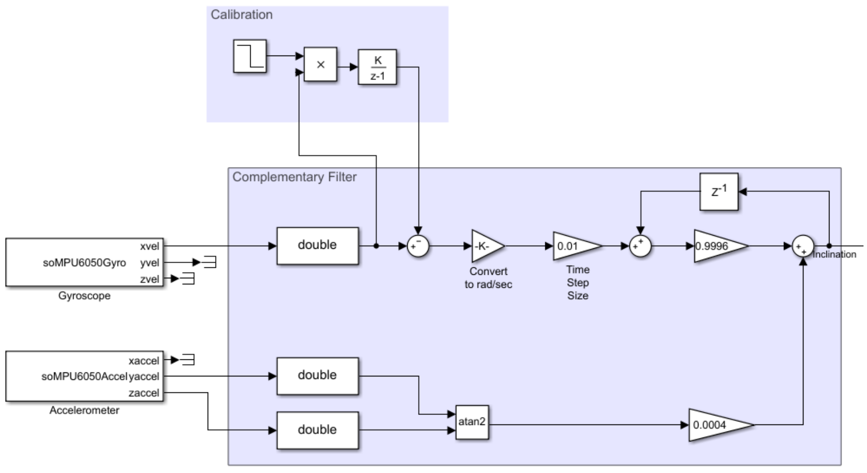

Figure 12. Block diagram of the complementary filter.

This block diagram includes also a calibration, because the gyroscope has always a bias and this amount depends on each device and can change with the temperature. So, the gyroscope must be calibrated after every start. In this project, the values gathered during the first second are averaged; the resulting value is subtracted from the current value. So, it is necessary that the bicycle is in upright position during the first second. The weight of the individual sensors is set experimentally, leading to a satisfactory result.

The Kalman filter was discarded in this occasion as it needs more computation power (we use an Arduino One Card).

5.4 Writing the actuators

To make the model runnable it is also important to write the calculated actuating variables to the actuators. In the case of the physical bicycle these are the PWM signal for the motor and the angle for the servo. The Simulink standard library has the right blocks for these actuators. The PWM block needs only an input from 0 to 255 so it can be connected to the controller, and the servo block needs only the angle in degrees, so the controller can be connected to this block after converting the angle to degrees. These elements are reflected on figure 16.

5.5 Controlling the physical bicycle

Now the controller can be ported to the Arduino card. At first the speed controller is tested. The controller produces similarly good results in the physical model as in the simulation with the same parameters, but this result cannot be measured because the bicycle must move for the correct result and a cable connection also affects the signal, because the cable is pulled behind. After this step, the controllers of the inclination must be tested.

5.5.1 PID controller for the inclination

The PID controller, which is designed in section 4.3, can be added directly to this block diagram, because it is a time discrete controller. With this controller, the physical bicycle works without changing the parameters. However, this controller generates many steering movements. This can be explained by the disturbance behaviour of the PID controller, because the D component reacts very quickly to the noise. This is because many disturbances occur when the bike is moving. In this case it is also not possible to measure the data to create a diagram. Below the complete Simulink block is shown; this block diagram includes also the controller for the speed and the control of the motor driver. A video of the simulation can be seen under [17].

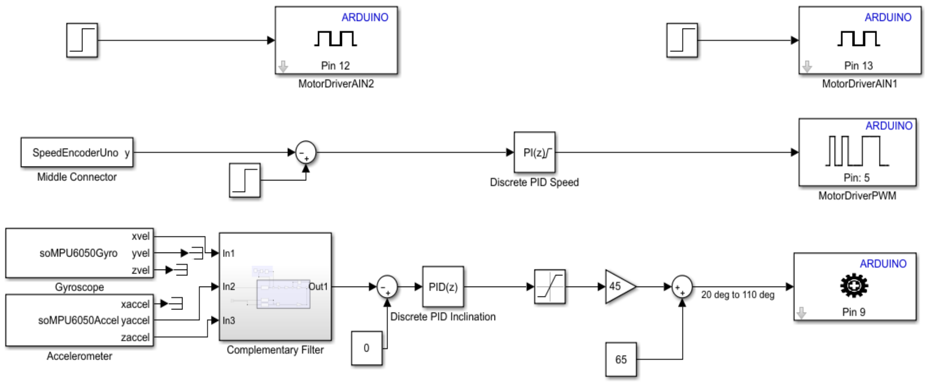

Figure 13. Block diagram of the PID controller.

5.5.2 State space controller for the inclination

In the same way as above, the state space controller can also be added directly to the block diagram. This controller works also without changing the parameters and the subjective result is slightly better. However, it is also not possible to record a diagram here. Figure 17 depicts the complete Simulink block, including the speed control block as well. A video of the simulation can be seen under [18].

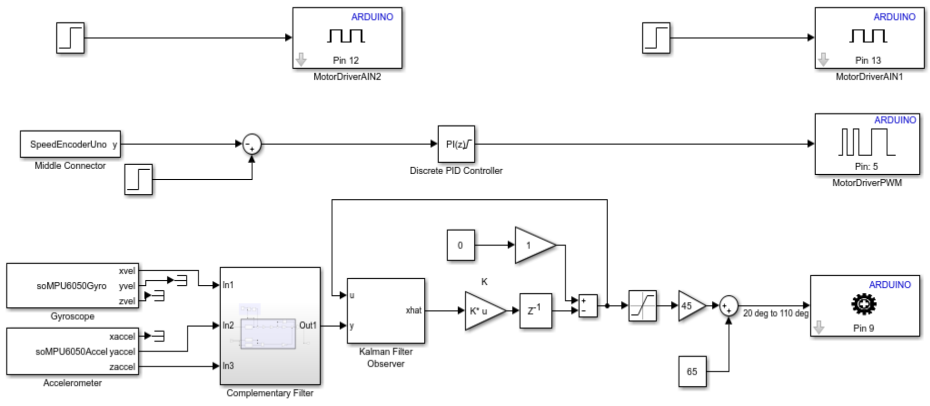

Figure 14. Block diagram of the state space controller.

6 Conclusion

The modelling of a complete autonomous mini-bicycle has been carried out, including the mechanical parts, sensors, actuators and the friction of the wheels with the ground, achieving the same behaviour as the physical bicycle.

We can conclude, then, that is it possible to create working controllers for mechatronics systems with the methods which are presented along this paper. Furthermore, the created model has shown satisfactory results leading to a matching between the behaviour of the controllers in the virtual model and in the physical bicycle.

Therefore, the created virtual model can be used for practical teaching of control technology, allowing the results to be ported directly to an Arduino board so that the students can observe the theoretical results in a real physical model.

References

[1] Minibalance. https://shop114407458.world.taobao.com/.
[2] Arduino Board Official Site. https://www.arduino.cc/ Access: 09.07.2018.
[3] TB6612FNG Toshiba Bi-CD Integrated Circuit. https://toshiba.semicon-storage.com/info/docget.jsp?did=10660&prodName=TB6612FNG. Last accessed: 09.07.2018.
[4] Invensense MPU 6000 Datasheet. https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf. Access: 09.07.2018.
[5] Autodesk Inventor Official Site. https://www.autodesk.com/products/inventor/overview Access: 04.07.2018.
[6] Matlab Simscape Official Site. https://es.mathworks.com/products/simscape.html Access: 04.07.2018.
[7] Matlab Simulink Official Site. https://www.mathworks.com/products/simulink.html Access: 04.07.2018.
[8] Matlab's SMI Port function. https://es.mathworks.com/help/physmod/sm/ref/smimport.html Access: 04.07.2018.
[9] Matlab Simscape Multibody Contact Forces Library. https://es.mathworks.com/matlabcentral/fileexchange/47417-simscape-multibody-contact-forces-library Access: 04.07.2018.
[10] P. Kurzweil, B. Frenzel and F. Gebhard, Physik Formelsammlung, pp.281-282, Wiesbaden, Vieweg, 2008.
[11] K. J. Åström, R. E. Klein and A. Lennartsson, "Bicycle Dynamics and Control", Control Systems Magazine, pp. 26-47, 01 01 2005.
[12] Video of the simulation of the PID controller of the inclination (I/II). https://youtu.be/dSf5DLQnZeE Access: 09.07.2018.
[13] J.Lunze, Regelungstechnik 1, pp. 88, 163, Springer Vieweg, 2016.
[14] Video of the simulation of the state space controller for the inclination (I/II) https://youtu.be/j3ycO2Lp8Mw Access: 09.07.2018.
[15] Rensselaer Arduino Support Package Library. https://es.mathworks.com/matlabcentral/fileexchange/62702-rensselaer-arduino-support-package-library--rasplib- Access: 04.07.2018.

[16] The Miniature Balancing Robot Kit for Education. Downloads page. https://minseg.com/pages/downloads Access: 04.07.2018.
[17] Video of the simulation of the PID controller for the inclination (II/II). https://youtu.be/_VZ02BhqNuk Access: 10.07.2018.
[18] Video of the simulation of the state space controller for the inclination (II/II). https://youtu.be/CNUgtSXgdqA Access: 10.07.2018.
[19] N. H. Getz and J. E. Marsden, "Control for an Autonomous Bicycle", 27.05.1995.
[20] L. Keo and M. Yamakita, "Controller Design of an Autonomous Bicycle with Both Steering and Balancer Controls", 10.08.2009.
[21] M. Sasaki, H. Tanaka and S. Ito, "Development of an Autonomous Two-Wheeled Vehicle Robot", 22.12.2011.
[22] A. Owczarkowski, D.Horla, P. Kozierski and T. Sadalla, "Dynamic Modeling and Simulation of a Bicycle Stabilized by LQR Control".
[23] Y. Feng, R. Dum Y. Xu, "Steering Angle Balance Control Method for Rider-Less Bicycle Based on ADAMS", 2017.
[24] J. Lowell, H. D. McKell, "The stability of bicycles", 7.11.1981.