TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

# The Self-balancing E-bike - Construction and Control

**Diplomová práce**

| | |
|---|---|
| *Studijní program:* | N2612 – Elektrotechnika a informatika |
| *Studijní obor:* | 3906T001 – Mechatronika |
| *Autor práce:* | **Bc. Zaid Al-Dailami** |
| *Vedoucí práce:* | Ing. Lukáš Hubka, Ph.D. |

Liberec 2023

Tento list nahraďte

originálem zadání.

## Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

7. 1. 2023 Bc. Zaid Al-Dailami

## Acknowledgement

I am happy to express my deep gratitude to my supervisor Lukaš Hubka for the great support, motivation, and guidance. It is my honor to be guided by him, especially with his knowledge and modesty.

I would like also to truly, delightfully, and completely send my regards and thanks to all staff of TUL professors who taught, helped, and guided me through my studying journey, and also not to forget to thank the studying department for all facilities and kindness they provided me with.

Last but not least, all massive thanks to my parents, family, friends, and everyone who supported me in accomplishing this project.

# The Self-balancing E-bike - Construction and Control

## Abstrakt

Tato práce se primárně soustředí na řešení samovyvažování se modelu jízdního kola. Nejprve jsou představeny různé možnosti řešení problému samovyvažování. Jeden z mechanismů je vybrán a následně detailně popsán a implementován na reálném modelu jízdního kola se setrvačníkem. Dále je namodelován, navrhnut a vytvořen mechanický model prototypu včetně elektroniky. Nakonec je provedena kalibrace a prostudováno několik způsobů filtrace signálů.

**Klíčová slova:** samovyvažovací model jízdního kola, reakční kolo, gyroskop, obrácené kyvadlo, Kalmanův filtr; odhad postoje; IMU; AHRS; Kvaternion; stat-spcae model; PID regulátor; komplementární filtr; kalibrace gyroskop; kalibrace akcelerometru.

# The Self-balancing E-bike - Construction and Control

## Abstract

This project is mainly about a self-balancing bicycle. Firstly, the various possibilities of balancing a bicycle are discussed. Secondly, one mechanism of these possibilities is chosen to be further studied and applied in a real robot which is balancing a bicycle with a reaction wheel. Thirdly, a proposed prototype of a mechanical model and electrical components is modeled, designed, and investigated. Fourthly, calibration and different kinds of filtering like, complementary filters, and Kalman filters are deeply studied.

**Keywords:** self-balancing bicycle model, reaction wheel, gyroscope, inverted pendulum, Kalman Filter; attitude estimation; IMU; AHRS; quaternions; stat-space model; PID controller; complementary filter; calibrating gyro; calibrating accelerometer.

# Contents

# List of abbreviations

| | |
|---|---|
| **SBMB** | Self-balancing Motor Bike |
| **LED** | Lighe Emitting Diode |
| **MEMS** | Micro-electromechanical System |
| **IMU** | Instrumental Measurement Unit |
| **GPE** | Gyroscopic Precession Effec |
| **BLDC** | Brushless DC electric motor |
| **IC** | Integrated Circuit |
| **UAV** | Unmanned Aerial Vehicles |
| **EKF** | Extended Kalman Filter |
| **UKF** | Unscented Kalman Filter |
| **AHRS** | Attitude and Heading Reference System |
| **NDE** | North-East-Down |
| **KF** | Kalman Filter |
| **ErKF** | Error State Kalman Filter |
| **LQR** | Linear Quadratic Regulator |
| **PID** | Proportional Integrator Derivative |

# 1 Introduction

Two-wheeled vehicles (bicycles) have been used since 1817. Since that time, two-wheeled vehicles have gone through different stages of development. Additionally, in the late 19th century, Electrical motors have been implemented in two-wheeled vehicles. Such vehicles have been named motorbikes.

Even though driving motorbikes or bicycles could be interesting and adventurous for some people, it might be dangerous for others. In addition, many people who did not learn to drive bicycles in their childhood, find it extremely difficult, embarrassing, and even sometimes impossible for them to learn it when they get older. Therefore, a self-balancing robot might be somehow helpful for those who did not have the chance to learn when they were young, with the case that the two-wheeled vehicles could be modified to function as a learning device. Furthermore, hazardous accidents happen every year causing injuries, permanent disabilities, and death. Thus, more stable two-wheeled vehicles might decrease the number of accidents and so the number of injuries and death rate. Moreover, Moreover, from the modeling, practical and experimental control point of view, the self-balancing bicycle is a nice challenge for control engineers.

At the beginning of the 21st century, a lot of effort has been focused on the system of self-driving cars (four-wheeled vehicles). In the case of motorbikes, however, a self-driving system might not be useful unless there is a system that balances the two-wheeled vehicles. The two-wheeled vehicle might have their preferences over four-wheeled vehicles. These preferences can be summarized in size, energy consummation, crowding, and adventure.

Furthermore, engaging the new cutting-edge technology in our daily uses devices has been an essential thing to facilitate our life with less effort and, of course, less cost. One of these technologies is data acquisition, signal processing, calibration, and filtering. Without this accuracy of processing, sensors would have their own flaws, errors, and probably damages, to property or even personal life. Therefore, getting data from sensors is not as easy as it sounds, and it surely requires some techniques and approaches in order to to get accurate signal results that will ensure functionality and safety. Some of these techniques are filtering and combining results of two or three sensors for getting the targeted output precisely, usually known as sensor fusion,

One of the best chances to apply and translate electronics knowledge, mechanical knowledge, and coding knowledge into reality is to be engaged with a real practical mechatronics project. Such a project can combine all knowledge and backgrounds together in order to produce a mechatronics system with our eyes open to the practical issues, errors, and flaws that usually are not taken into account in theory.

The previous motivations have been a good spark for me to put more effort to make analytical research on the different possibilities of balancing two-wheeled vehicles, proposing a prototype of a self-balancing robot, designing an initial model for the robot, constructing a 3D prototype, study different possibles of getting clean precise output results of sensors, model the plant, design the controller and implement it to the whole system.

# 2 Possibilities of stabilizing bicycles

Each vehicle driven on the road must be somehow stabilized and running in a safe mood. Four-wheeled vehicles and three-wheeled vehicles are normally stabilized by the wheels, but for two-wheeled vehicles, this is not the case. Each two-wheeled vehicle must be, therefore stabilized either by the driver himself or by another mechanism. In order for the two-wheeled vehicle to be autonomous not only does it need a system for driving it safely and intelligently, but also it must have an engineering system that enables it to be stabilized. There are different possibilities by which the two-wheeled vehicle (bicycle) can be stabilized. Some of these engineering ideas are as follows:

## 2.1 A reaction wheel

The idea of a self-balancing robot can be implemented on bicycles with different mechanisms. The reaction wheel, however, is considered one of the most suitable methods due to its straightforwardness, functionality, and robustness. The reaction wheel mechanism relies on the principle of an inverted pendulum. An inverted pendulum is a pendulum with the exception that its center of mass is above its pivot and thus should be stabilized [1][2][3][4]. Figure 2.1 shows the idea of stabilizing the cart's inverted pendulum.

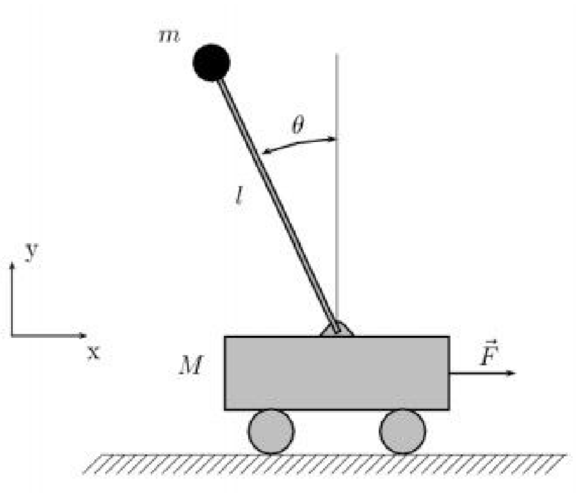


Figure 2.1: Cart inverted pendulum

The concept of the reaction wheel control is basically controlling a spinning motor that is connected to a wheel with calculated inertia. Then, the motor accelerates or decelerates with the possibility of rotating in both directions depending on the requirements. This motion of the wheel creates a reaction torque based on the law of conservation of momentum and thus enables to control of the bicycle. The advantages of using such a mechanism can be summarized in low price, simplicity, and absence of ground reaction. While the disadvantages are high energy consummation and limited torque amounts[1]. One efficient prototype, that harnesses and utilizes this idea, is called MURATA BOY[3][5]. It is shown in Figure 2.2.

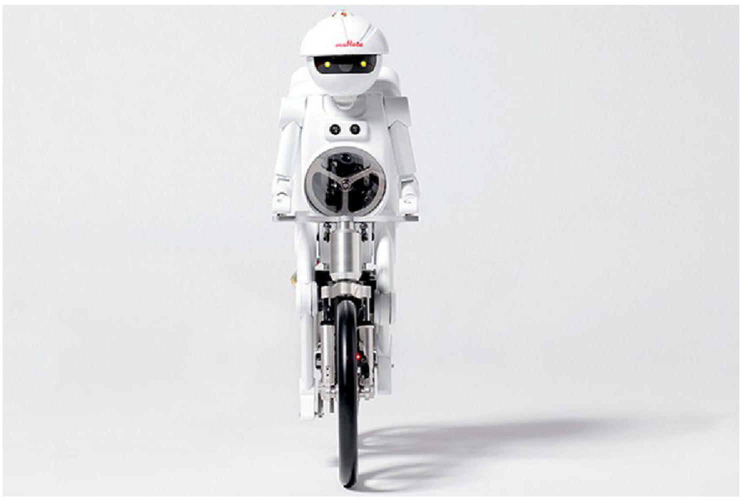


Figure 2.2: MURATA BOY prototype

## 2.2 Gyroscopic Stabilisation

Gyroscopes as systems or sensors(MEMS) are one of the most useful devices in the engineering world. They have different implementations such as in cameras, helicopters, balancing vehicles, and spacecrafts[6]. A simple gyroscope system with its components is illustrated in figure 2.3.

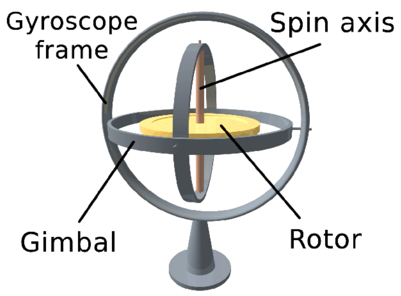


Figure 2.3: A simple gyroscope

Gyroscopes can as well be implemented for balancing bicycles in different forms and methods, some of which are:

### 2.2.1 A gyroscope, its axis is parallel to the driving wheels' axes

This stabilizing method is based on the gyroscopic precession effect (GPE). It basically uses two motors. The first motor is a high-speed BLDC motor for rotating the rotor of the gyroscope. The second motor is a DC servo-motor for moving the gimbal when required and so balancing the bicycle[7][8]. More illustrated details are in figure 2.4.

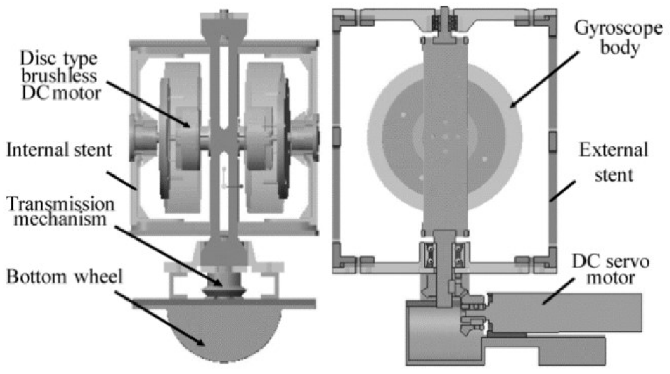


Figure 2.4: A balancing bicycle gyroscope

The advantages of using this method compared to the reaction wheel method, are more stability, efficiency, and accuracy. Furthermore, since the rotor rotates at a very high speed, it provides a very stable movement. In the case of changing the angle of the gimbal, the reaction balance torque will be very huge, and so very precise and balanced.[7]

This method has been implemented in the Gyro-X-1967 prototype as shown in figure 2.5[9]. Now this prototype is preserved in Lane Motor Museum

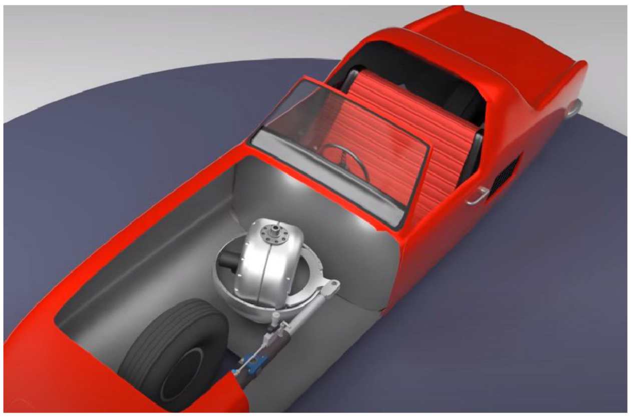


Figure 2.5: Gyro-X Car prototype

### 2.2.2 Two gyroscopes their axes are perpendicular to the driving wheels axes

This mechanism is based on two gyroscopes that rotate in different directions with constant velocity. When the vehicle tilts (rolls) to the right or to the left, there is a gyroscope sensor that sends a signal to the motor and so rotating the gimbal upward or downward. When the gimbal rotates, it rebalances the bicycle again[10][11]. Figure 2.6 shows the principal working of a couple of gyroscopes[12].

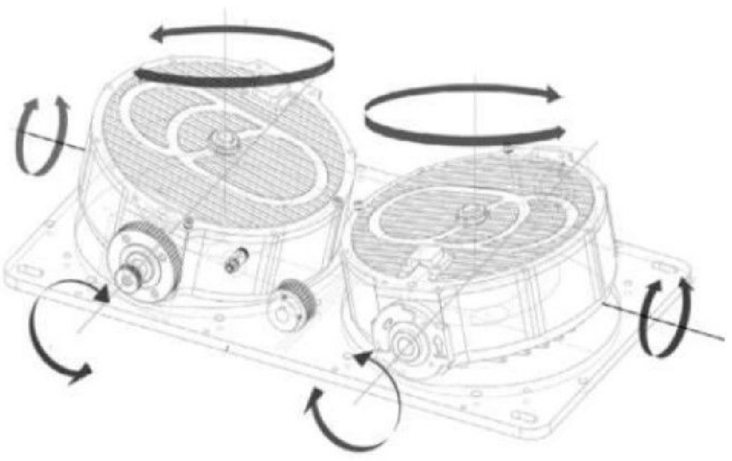


Figure 2.6: Couple gyroscopes

This method of stabilizing is considered the most accurate and stable among all methods mentioned in this project. Therefore, it has been implemented in a real-world self-balanced motorbike prototype called C1 by Lit Motors Company. The prototype is illustrated in figure 2.7[12]

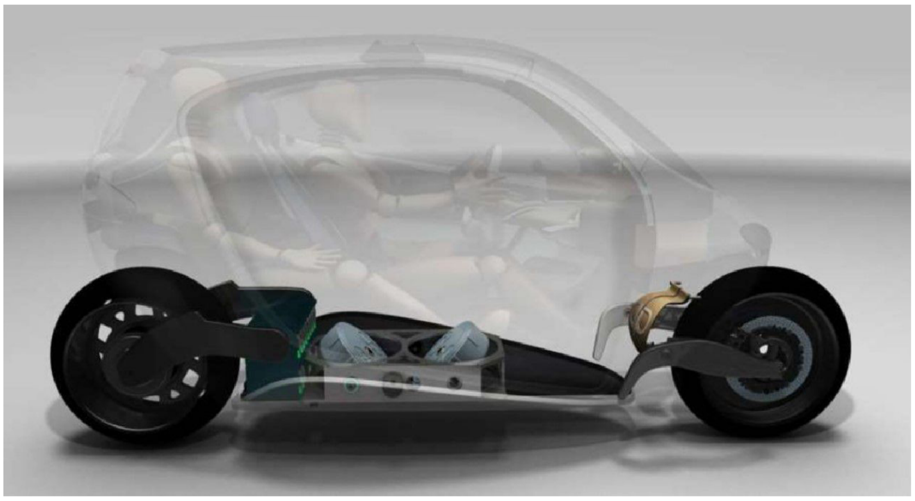


Figure 2.7: C1 prototype by Lit Motors

## 2.3 A torque applied on the steering handlebar

This mechanism basically balances the bicycle by an external force or torque that is applied directly to the steering handlebar. The external force can be in the form of an electric linear actuator[13]. The external torque can be in the form of a rotational motor.[14]

The advantages of such a mechanism are lightness and low power consumption. On the other hand, the disadvantages are a less stable system and a lack of robustness when the tilt angle is large. Figure 2.8 shows an actuator that balances the bicycle by balancing the steering handlebar.

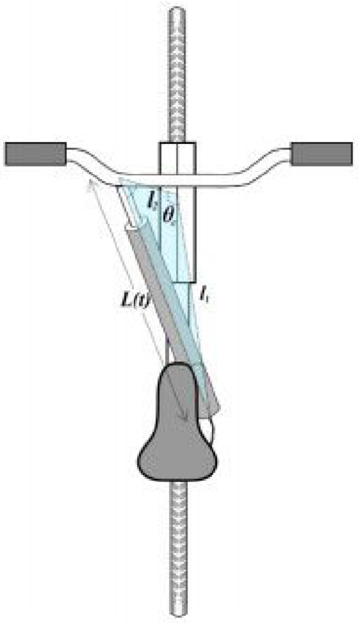


Figure 2.8: Balancing a bicycle by an actuator

# 3 Construction, prototyping, assembly

From the previous methods of motorbike controlling mentioned in chapter 2, the reaction wheel method is chosen to be further studied, analyzed, and performed. For this method, multiple electrical and mechanical components will be needed to do the job. The idea in this prototype is that a self-balancing motorbike or (SBMB) will be able to stabilize itself with the help of a reaction wheel that is connected to a DC motor. When the bike detects any kind of tilt or disturbance, SBMB should be able to detect these changes and upon them take proper action by the reaction wheel motor. Angle detection is achieved by fusing an accelerometer, gyroscope, and magnetometer. All these sensors are inserted in the MPU-9250. In addition, there are two other motors that are responsible for moving and steering. These motors are controlled by a smartphone via a Bluetooth module. Figure 3.1 shows the overall schema from the upper compact level point of view.

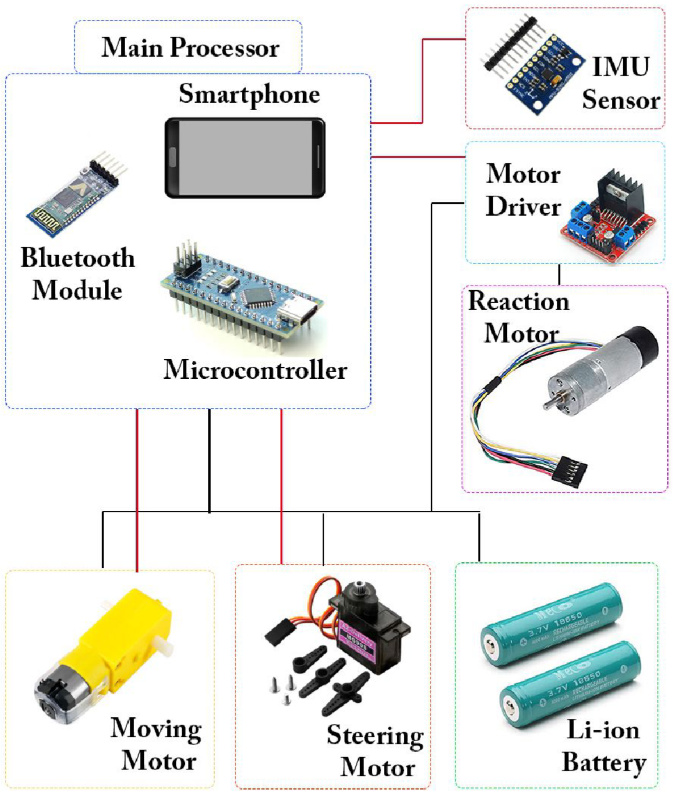


Figure 3.1: Scheme of electrical components

## 3.1 Assembly

The SBMB system is based on a controlling approach. In other words, it requires all essential components for successful feedback control. So, it needs a microcontroller, appropriate sensors, actuators, and a well-designed system that will facilitate the controlling approach namely the center of mass position. Figure 3.2 shows the chart or the circle of a successfully controlled mechatronics system.

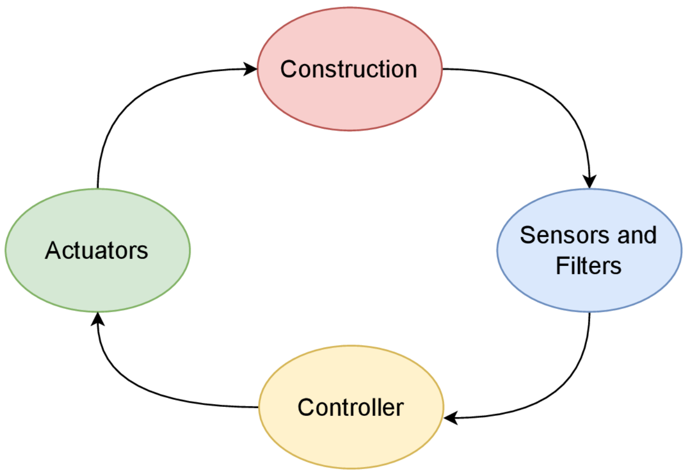


Figure 3.2: The Circle of a successful controlled mechatronics system

Moreover, to successfully accomplish such a system, a lot of mechanical and electrical components are needed. The multiple main electrical and mechanical components are illustrated in the following table.

Table 3.1: Electrical and mechanical components

| The component | The number |
|---|---:|
| Arduino nano | 2 |
| Li-ion batteries | 4 |
| Bluetooth module | 1 |
| Motors | 3 |
| Motor driver | 1 |
| Switch | 1 |
| MPU 9250 | 1 |
| Wheels | 2 |
| Reaction wheel | 1 |
| Bearings | 2 |
| Shafts | 2 |

## 3.2 Construction

Two 3D prototypes are designed for the purpose of construction, both are done in Inventor. The first one is in figure 3.3 and the second one is in figure 3.4

### 3.2.1 Prototype A

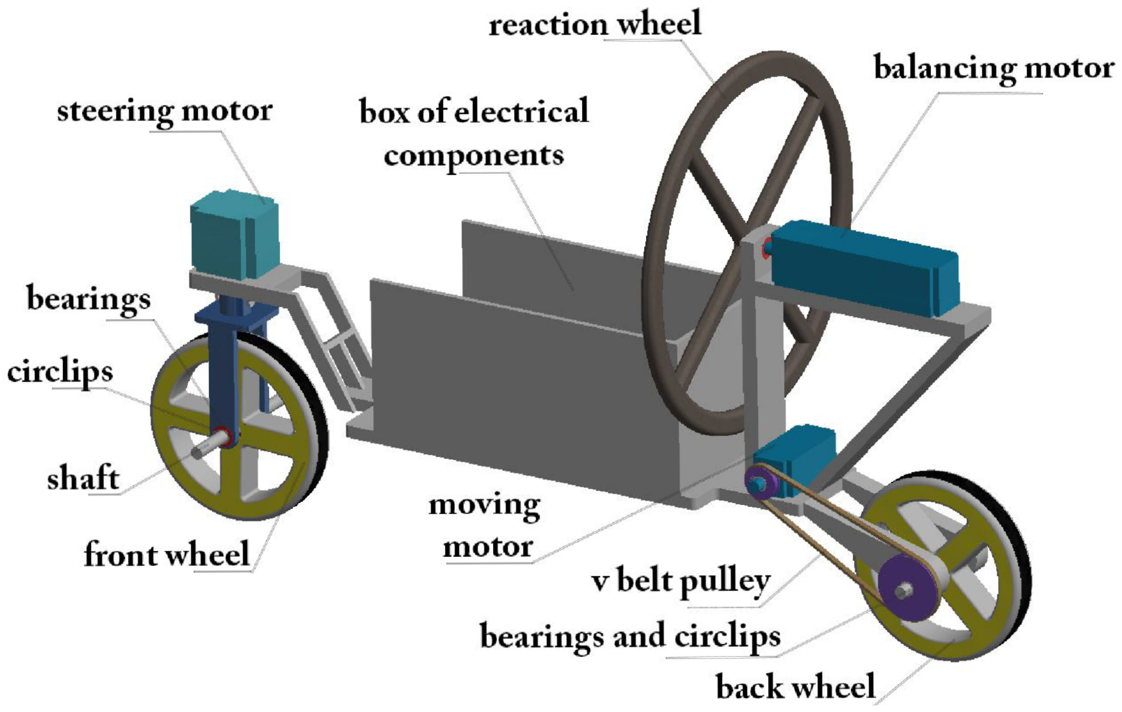


Figure 3.3: Bicycle 3D design A

### 3.2.2 Prototype B

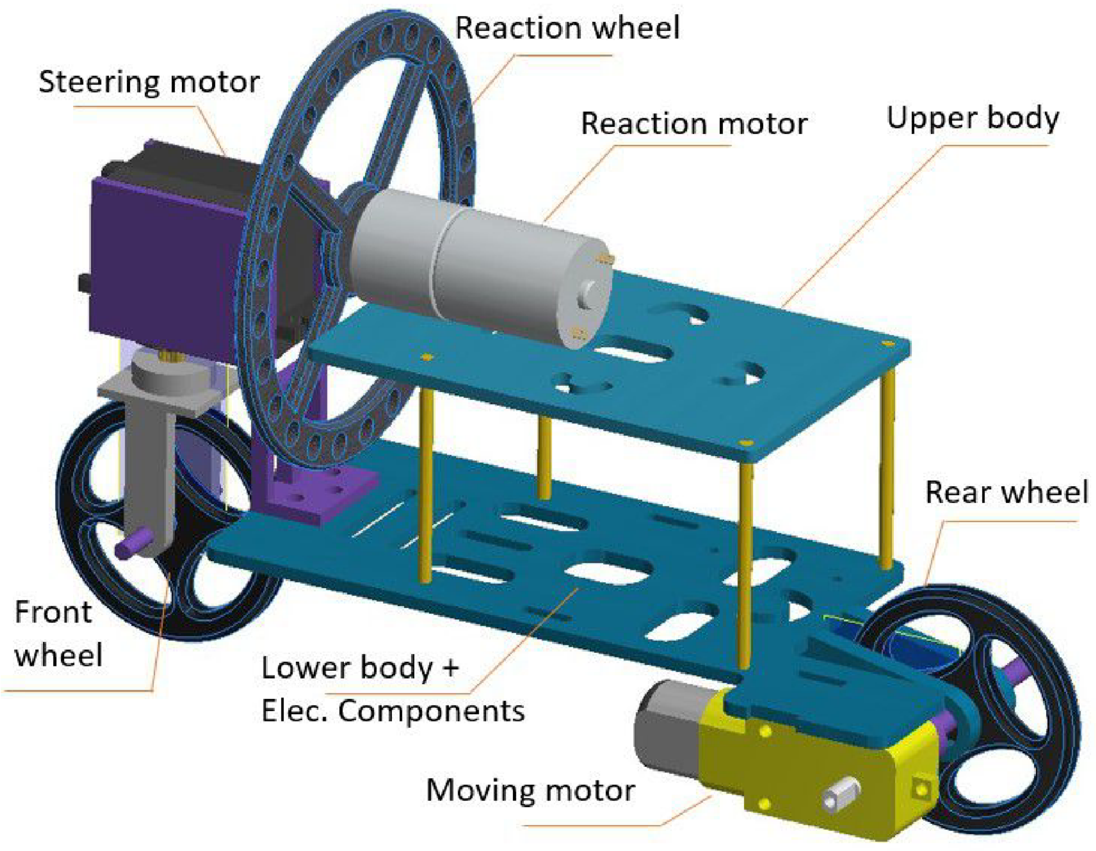


Figure 3.4: Bicycle 3D design B

# 4 Actuators and sensors

Actuators and sensors are the devices required in any machine to recognize what happens in the surrounding environment and therefore react to the action accordingly and properly.

## 4.1 Actuators and motor drivers

### 4.1.1 Actuators

An actuator is a transducer that converts one form of energy to another. For example, the electric motor is considered to be an actuator because it converts the electric energy to mechanical transnational or rotational energy. There are many types of actuators as eclectic, pneumatic, and hydraulic.

In this project, only eclectic actuators have been used as a source of energy transformation. Three electric motors are used. The first motor is a servo motor for enabling the bike to turn left and right. The second motor is a dc motor for allowing the bike to move forward and backward. The third motor is a dc motor that is connected to the reaction wheel for the balancing process of the whole bike. The three motors are shown in figure 4.1[15]

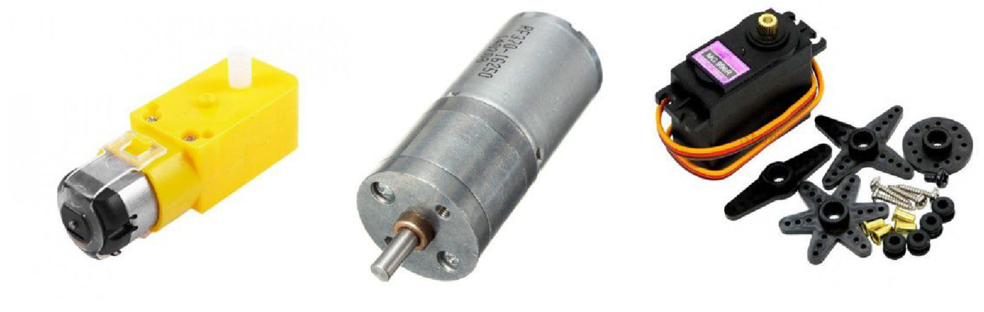


Figure 4.1: The drive motor, the reaction wheel motor, and the steering motor respectively

### 4.1.2 Motor drivers

Motor drivers are devices that are used for controlling motors. Different types of motors have different ways of driving. For DC motors, H-Bridge is preferably used due to its simplicity and low cost. The H-Bridge that is used for driving the dc motors is L293D. Figure 4.2 shows the principal working of H-Bridge and L293D. H-Bridge's functional principle is that it has electrical switches that are controlled by microcontrollers' PWM signals. And by these signals, the motor is driven directionally and quantitatively[16]

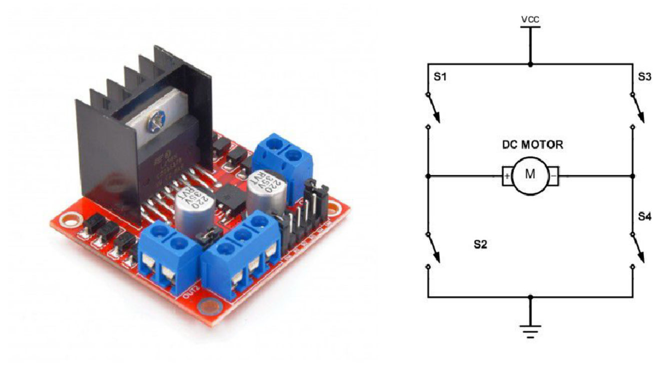


Figure 4.2: L293D motor driver, schema of H-Bridge working principal

## 4.2 Sensors and sensor fusion

Sensors are instruments that allow the detection of an expected change in the contiguous environment. The considerable development of technologies in ICs has led to a revolutionary contribution to the manufacturing sensors world. Nowadays, the trend and the most used sensors are smart and soft sensors, in which hardware sensing is neglected. In other words, a soft and smart sensor is an instrument that involves a microprocessor that is capable of doing mathematical operations so that it can produce the required output signals of interest with less noise.[17]

### 4.2.1 MEMS and MPU9250

MEMS or Microelectromechanical systems are small devices that combine electrical and mechanical components in the form of ICs to do advanced and completed microscopic tasks.[18]

IMU or Inertial Measurement Unit is a MEMS device that combines multiple sensors in order to determine the orientation of an object with respect to a specific frame. IMUs sensors can vary from one type to another depending on the sensors inserted, but most of them include sensors that can detect angular rate, the earth's magnetic field, and gravitational acceleration. The most important applications of IMUs are in planes, satellites, and UAVs[19].

MPU9250 sensor is an IMU sensor that includes 3 sensors, an accelerometer, a magnetometer, and gyrometer. Each sensor detects the change in three axes XYZ, so in total, this device outputs 9 signals. Every three signals are on one axis. Figure 4.3 shows how the MPU9250 sensor looks like.

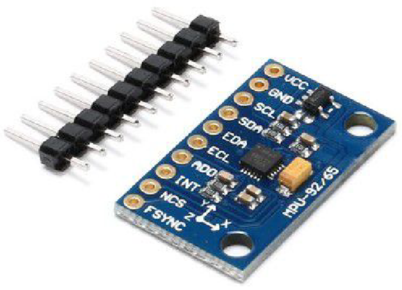


Figure 4.3: MPU9250 sensor

## Accelerometers

An accelerometer is an instrument that can measure linear or gravitational accelerations. There are different types of accelerometers like MEMS capacitive accelerometers, piezoresistive accelerometers, and piezoelectric accelerometers. The one that is used in MPU9250 is a MEMS capacitive accelerometer. Its working principle is based on this, a mass is placed between springs. The mass, depending on the action, moves to one side or another. A change, that occurs in the mass, will result in a change in the capacitance. This change is recorded as a G-Force. Figure 4.4 illustrates the working principle of MEMS capacitive accelerometers.[19][20]

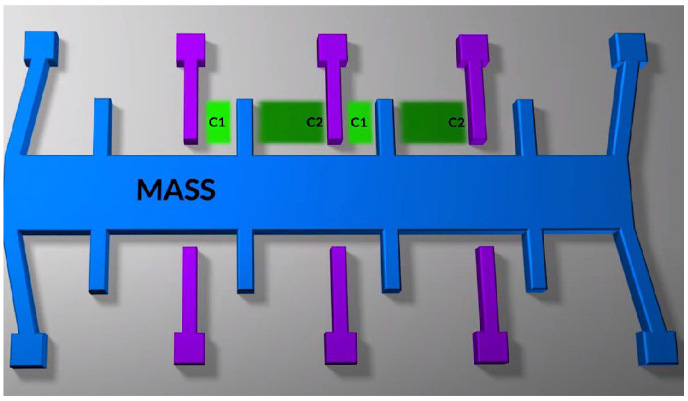


Figure 4.4: MEMS capacitive accelerometer working principal

The only drawback of using accelerometers alone is that the output results are not highly accurate and their noise is high. Therefore, for applications that need more precise angle detection for example roll and pitch, then other techniques should be used. Using an accelerometer alone can be used to find roll and pitch angles but only in 2D. The formulas for calculating roll and pitch angles by accelerometer readings are as follows:

$$\phi = \arctan 2 \frac{acc_y}{acc_z} \tag{4.1}$$

$$\theta = \arctan 2 \frac{-acc_x}{acc_y^2 + acc_z^2} \tag{4.2}$$

where $\phi$ is *roll angle*, $\theta$ is *pitch angle*, $acc_x$ $acc_y$ $acc_z$ are *accelerometer readings in X, Y, and Z respectively.* These Euler angles' equations are derived according to these documents.[21] [22] A Simulink model is done for calculating the roll and pitch angle. It is added to the appendix. In figure 4.5, a plot draws the results of Euler angles of the Simulink model. From the plot, it can be noticed that the noise is high and the accuracy is decent.

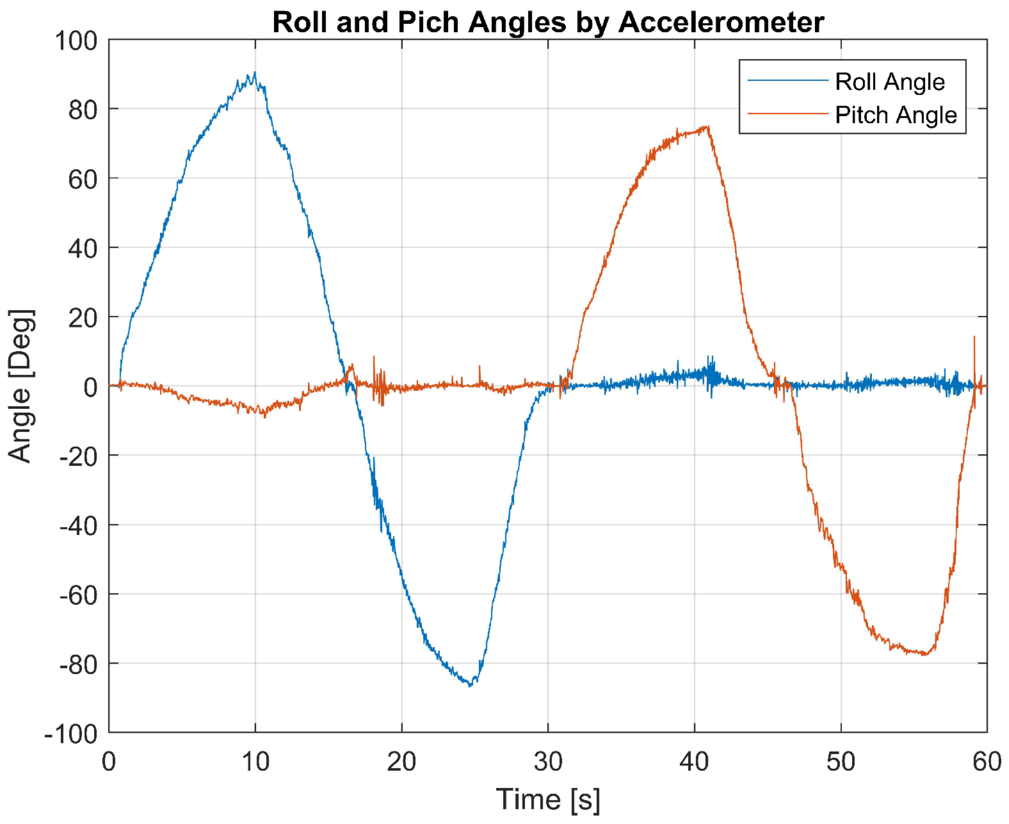


Figure 4.5: Roll and pitch angles by accelerometer

### Gyroscopes

Gyroscopes are instruments used to detect angular rate and therefore enabling to know angular velocity and displacement. Its working principle is based on the Coriolis effect as shown in Fig. 4.6 In which, when there is a mass moving in a particular direction with velocity illustrated by the red arrow, and an external angular rate is applied represented by the blue curved arrow, these two effects will produce the Coriolis force represented by the blue arrow which will result in a change in capacitance. This change is then processed and translated as the angular rate.[19][20][23]

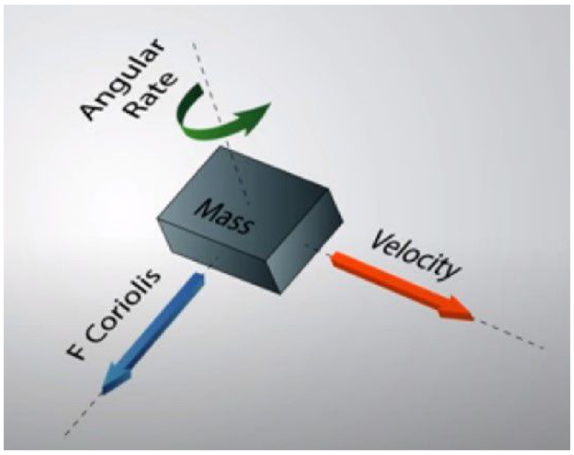


Figure 4.6: Gyro working principle

Gyroscopes give the angular rate. Therefore, the integration of it gives the angular displacement. In figure 4.8, the plot shows the roll and pitch angles. Since we are dealing with discrete integration, it keeps drifting, making it inefficient for long-term angle measuring. From the plot, it is noticed that the roll angle is drifting downward while the pitch angle is drifting upward.

#### Magnetometers

Magnetometers are scientific sensors that are used to detect the earth's magnetic field. One common method used in manufacturing magnetometers is hall effect magnetometers. The hall effect working principle simply states that, when we have a conductive plate and there is a current running, the current will flow straight from one side to another, but when the conductive plate comes across a magnetic field, the electrons will deflect to one side of the plate and the other side will be positively charged, and so a voltage will be created between the plates. Figure 4.7 shows the working principle of hall effect sensors. [24][25]

Values of roll and pitch got by accelerometer can be used with magnetometer signal in the 3 axes X, Y, and Z to get the value of Yaw. The equation of yaw in this paper[26] was used in my Simulink model. The results of the Simulink model can be noticed in figure 4.9. We can see no matter how we rotate the body in the yaw axis, the roll and pitch angle ate stable.

### 4.2.2 Sensors calibration

Sensors are the detective instruments that let the device recognize the change in the environment. Therefore, the accuracy of these devices is highly preferred. However,

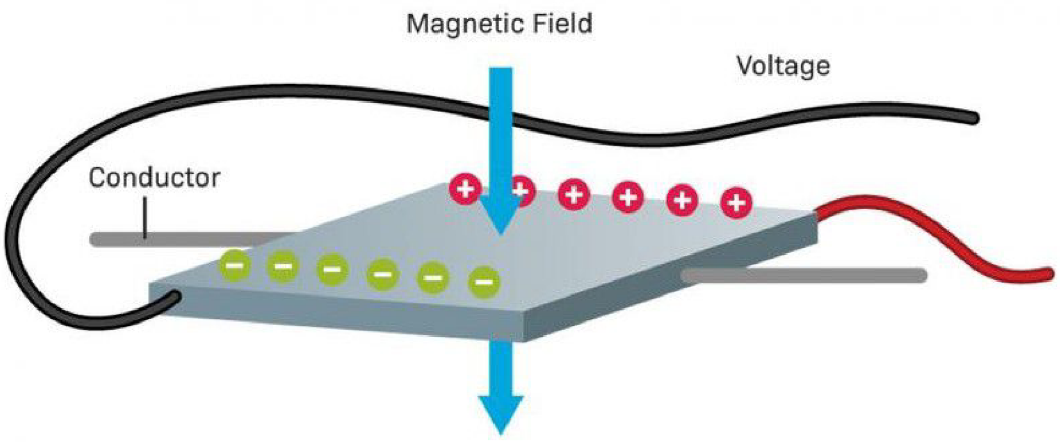


Figure 4.7: Hall effect principle

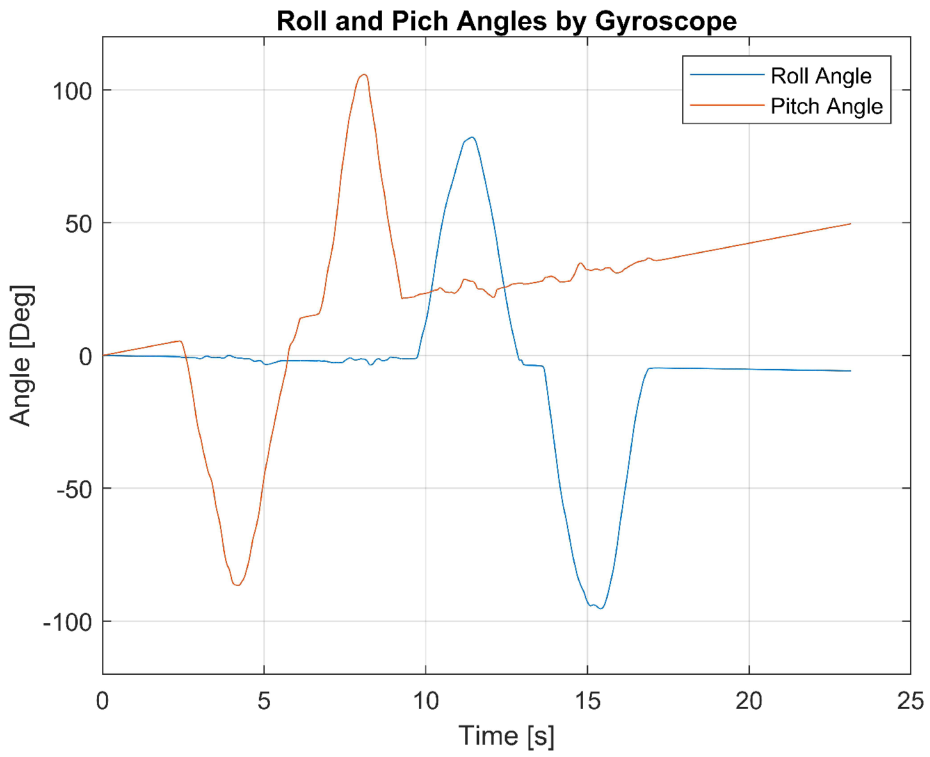


Figure 4.8: Roll and pitch angles by gyroscope

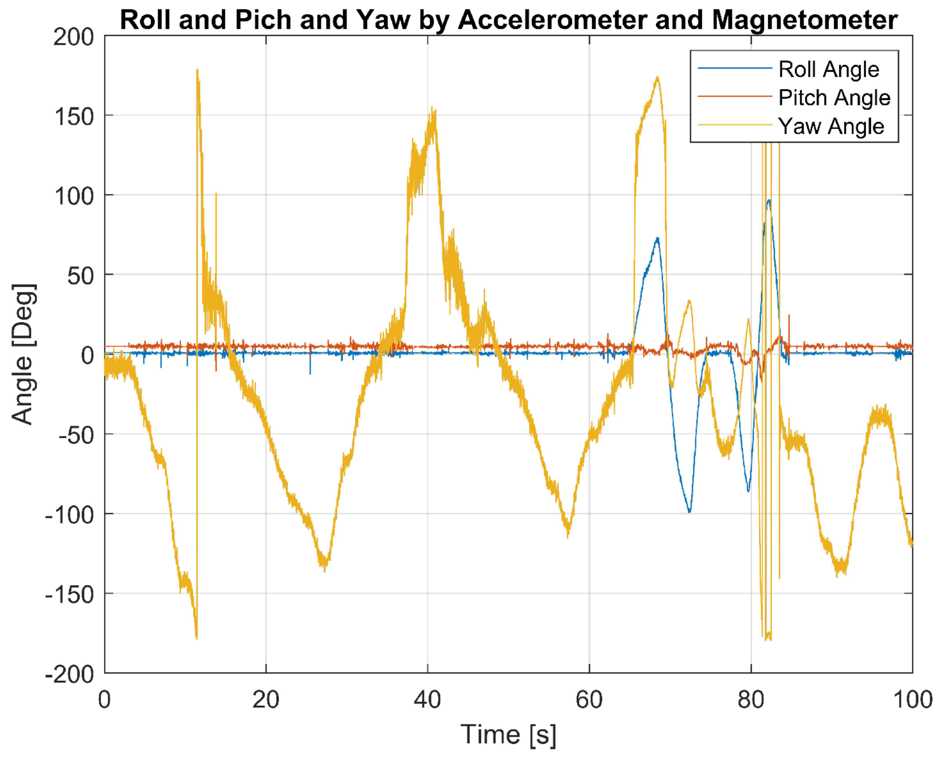


Figure 4.9: Roll and pitch angles by accelerometer and magnetometer

this is not usually the case, especially when the errors are prohibitive. So, sometimes a process that is called calibration is required for more precise output results.

## Accelerometer calibration

Ideally speaking, the three axes in the accelerometer should be aligned $90^0$ from each other, but due to manufacturing errors, it is common that an accelerometer will have non-orthogonality errors. In addition, no matter how much expensive the accelerometer is, it will have some bias. For these factors, calibration is needed for more trustworthy results. Not to forget mentioning that also removing bias can also be done by estimation, but calibration is more straightforward.[27][28] The equation used for calibration is as follows:

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \begin{pmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{31} \end{pmatrix} \begin{pmatrix} a_{Xraw} - b_x \\ a_{Xraw} - b_y \\ a_{Xraw} - b_z \end{pmatrix} \tag{4.3}$$

where $a_x\, a_y\, a_z$ are the *calibrated measurements*, $S$ matrix is *scale-factor* and nonorthogonaliy corrections, $a_{Xraw}$ $a_{Yraw}$ $a_{Zraw}$ are are the *raw measurements*, and

$b_x\, b_y\, b_z$ are the *bias corrections*.

For the calibration, multiple readings are taken from the accelerometer in different positions. The more reading there are, the more accurate is the calibration. One side note, when taking the measurement values, the sensor should be as steady as possible. This process is done within Matlab the code is in the appendix. The results are compared in figure 4.10 and figure 4.11. Figure 4.9 scatters the raw and the calibrated values in 3D, while figure 4.10 plots the data in XY plane.

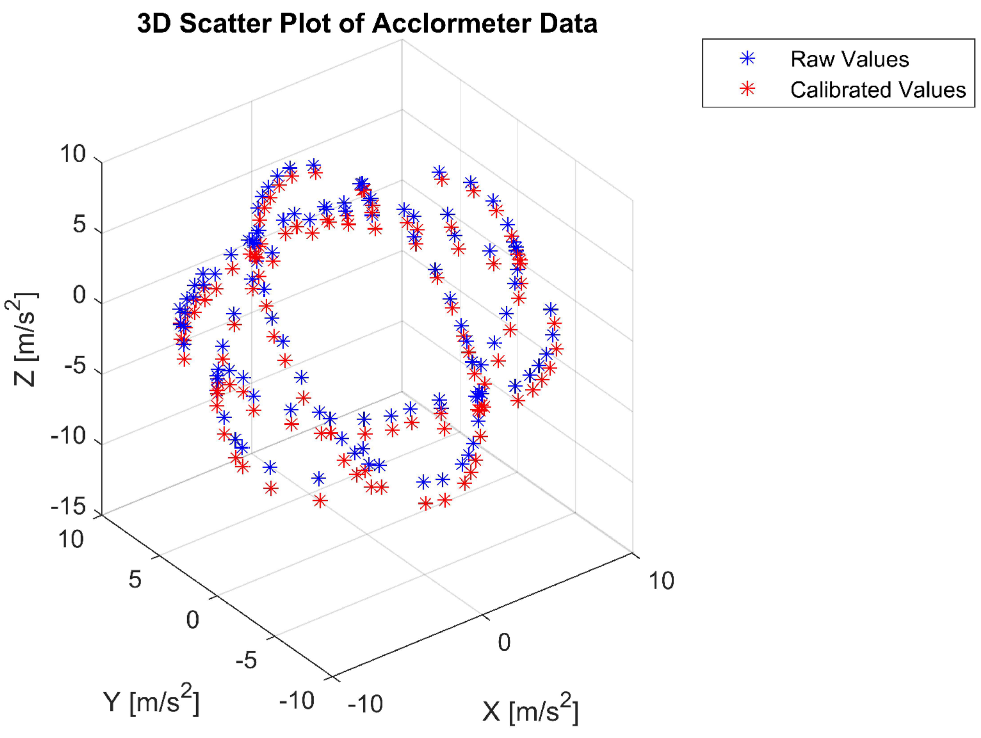


Figure 4.10: Comparison of raw and calibrated values in 3D

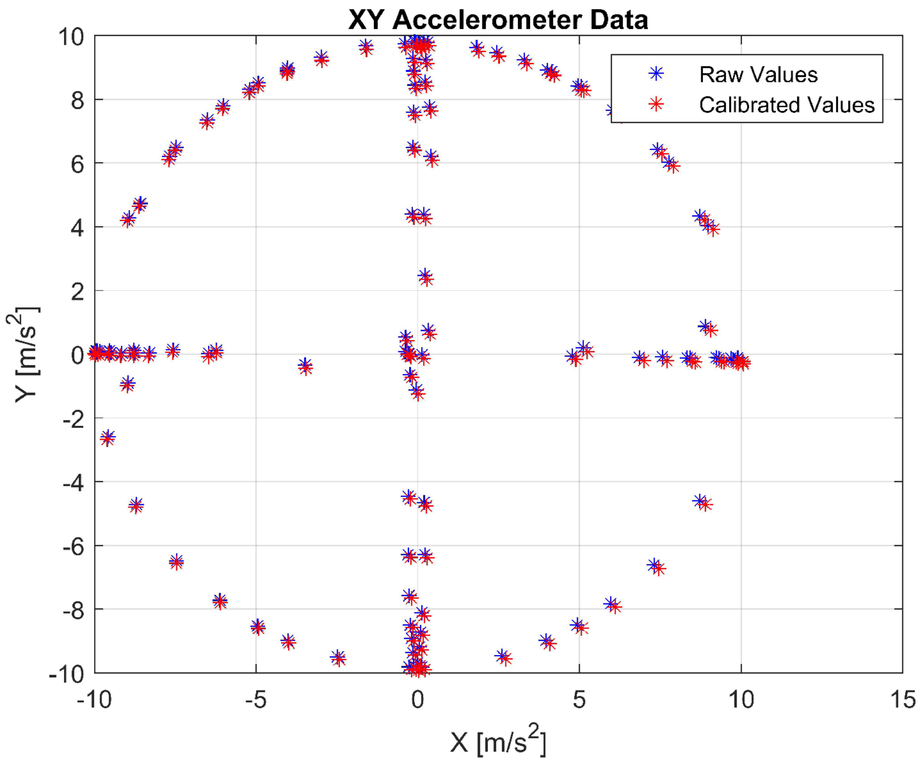


Figure 4.11: Comparison of raw and calibrated values in XY plane

### Magnetometer calibration

Calibrating the magnetometer is an essential step in any sensor. The errors are caused either by soft iron distortions or hard iron distortions. Soft iron distortion is usually caused by paramagnetic materials or ferrous metals like iron and steel. Hard iron distortions or biases are caused by magnets or high-current wires. Soft iron distortions or hard iron distortions are not the only reason why calibration is required. Calibration is also needed for axes' misalignment[29][30].

Applicationly speaking, it is good to keep the magnetometer far from magnets or magnetized metals. So that errors can be mitigated. For doing the calibration, firstly we take raw values from all sides. In other words, we rotate the sensor in $360^0$ multiple times in all axes directions. And then *magcal* function in MATLAB is used so that we can get the A matrix and the b vector. Then we can apply this equation for getting the calibrated values:

$$m_{calib} = A\ (m_{meas} - b_x) \tag{4.4}$$

where $m_{calib}$ is the *calibrated magnetometer vector*, $m_{meas}$ is the *measured magnetometer vector*, $b_x$ is the *hard iron corrections vector*, and $A$ is $3 * 3$ matrix that represents *soft iron, scale factor, and misalignment correction.*

After that, we can compare raw and calibrated values. The Matlab script for calibration is included in Appendix. The results of calibration are in Figures 4.12

and 4.13. It is noticeable that the calibrated values are centered around the zero value.

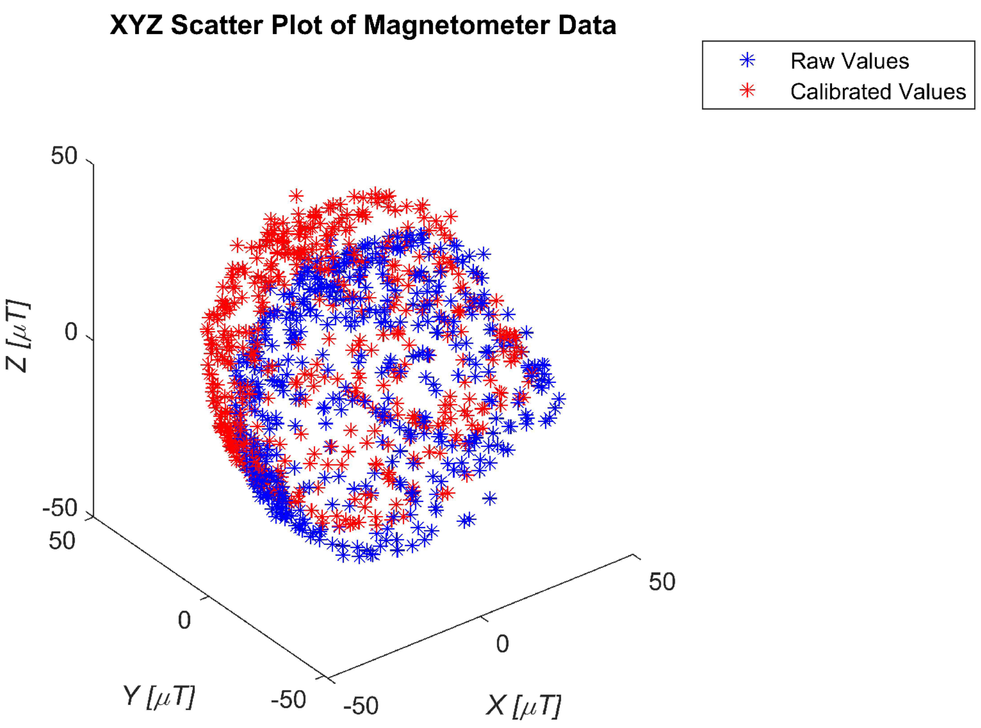


Figure 4.12: Comparison of raw and calibrated values in 3D

### 4.2.3 Sensor fusion

Sensor fusion or a multi-sensor system is a method that basically combines more than one sensor for a better perception and a more stable performance. Depending on results from a separate sensor cannot always be guaranteed. A single-sensor system can sometimes lead to catastrophic results, especially if the application field requires more sensitivity and precision. Therefore, sensor fusion is an essential step in such applications. In addition, research and development in this field have become a key goal for many companies. In other words, sensor fusion is a rapidly evolving field that will play a role in the cutting-edge technologies that will lead the world of manufacturing.[31][32] Figure 4.14 shows the fundamental concept of sensor fusion where multiple inputs are entered into the system. Then these data are processed to give output results.

The drawback of using individual sensors can be summarized in suffering from limited range, performance degradation under certain environmental conditions or limited field of view [33]. To look at it from a human point of view. Then, using individual sensors is like a human who has senses like taste, smell, touch, and hearing, but his brain cannot process and analyze what happens in the environment. So,

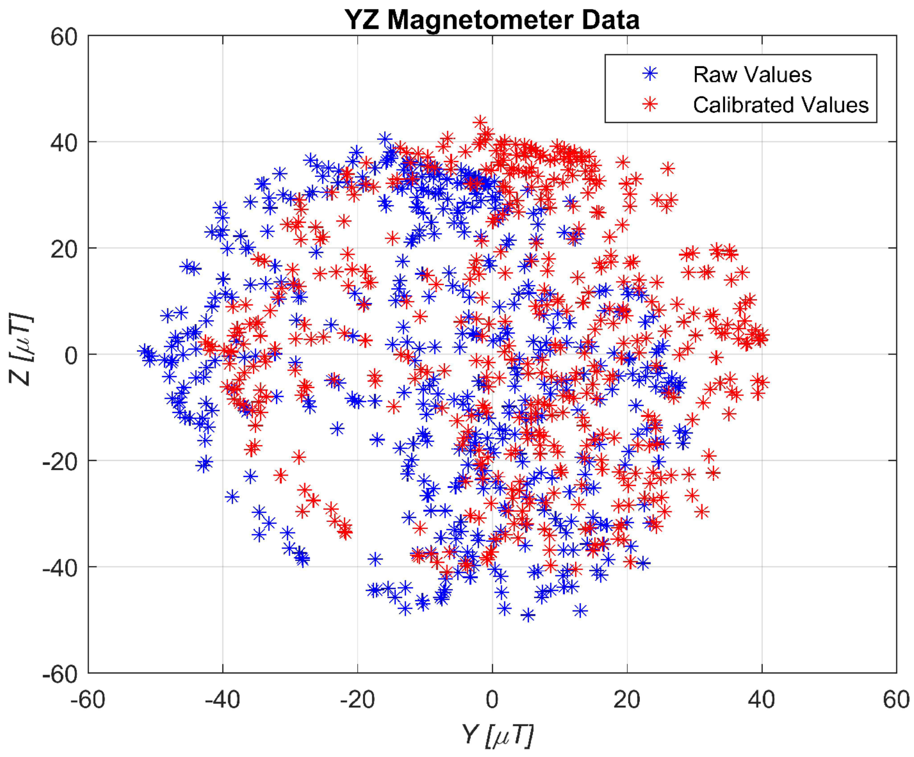


Figure 4.13: Comparison of raw and calibrated values in XY plane

I would describe sensor fusion as a progressive advanced approach that enables the device to perceive and sense the surroundings with higher efficiency and fewer errors.

The biggest advantage of sensor fusion is that it makes the device more intelligent, controllable, and reliable. However, using multiple sensors can be costly and complex to implement or understand.

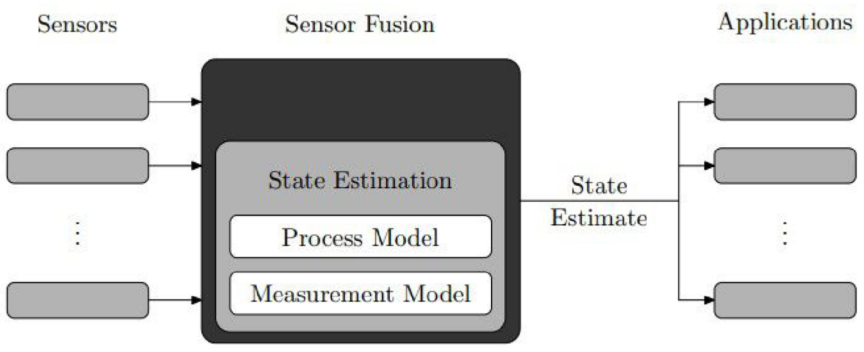


Figure 4.14: Basic concept of sensor fusion[32]

### Complementary filter

For simply advanced applications, the complementary filter is a good choice. Its basic idea is to compare two sensors and harness the best accurate measured values from any. As discussed and noticed before from graphs, individual use of sensors is not reliable. For example, the accelerometer alone is only good when the system is stable. So it is good when at rest which means for long time measurements. While the gyroscope is only good in the short time because it relies on integration and since we deal with digital systems, what is known as drift takes place as illustrated in figure 4.8. As a result, the trick is how to depend on an accelerometer for long-time measurements and also how to depend on a gyroscope for short-time measurements. And that is why the complementary filter exists.[34][35][36].

Since accelerometer measurements are highly affected by vibrations, a low pass filter will be needed to overcome or overpass the high-frequency signals, while for eliminating the drift effect, a high pass filter will be needed as illustrated in fig 4.15.[35]

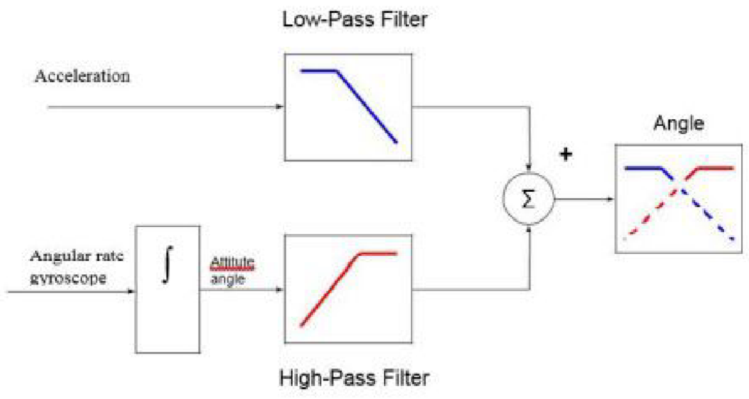


Figure 4.15: Block diagram of digital complementary filter system [36]

The equation used for the complementary filter is as follows:

$$\phi_{n+1} = \phi_{acc,n} \, . \, \alpha + (1-\alpha) \; (\phi_n + T \, . \, \phi_{gyro,n}) \tag{4.5}$$

where $\phi_{n+1}$ is the *current angle estimate*, $\phi_{acc,n}$ is the *measurement from the accelerometer*, $\alpha$ is a *constant* between (0,1), $\phi_n$ is the *previous angle estimate*, T is the *sample rate*, and $\phi_{gyro,n}$ is the *measurement form gyroscope.*

The $\alpha$ constant determines which sensor we tend to rely more on upon. When $\alpha$ is close to zero it means we trust the gyro measurements and when it its is close to the one we trust more accelerometer measurements. Typically $\alpha$ is a number that is close to 0 because the accelerometer is used only as a support for compensating the gyro drift.

Figure 4.20 shows the results got by the complementary filter. The Simulink model is included in Appendix.

### Kalman filter 2D

The Kalman filter is an optimal estimation algorithm for measuring estimate states of the system from some measurements[37]. Kalman filter is basically an observer that estimates a state than cannot be directly measured. It only depends on current and previous measurements. Therefore, one advantage of using the Kalman filter is that they only need low memory.[38] Kalman filters do not function as other sensors which only clean up the data. Rather, they combine, process, and estimate the data.[39] Kalman filter's main parts are prediction and estimation. Prediction and estimation equations are illustrated in figure 4.16

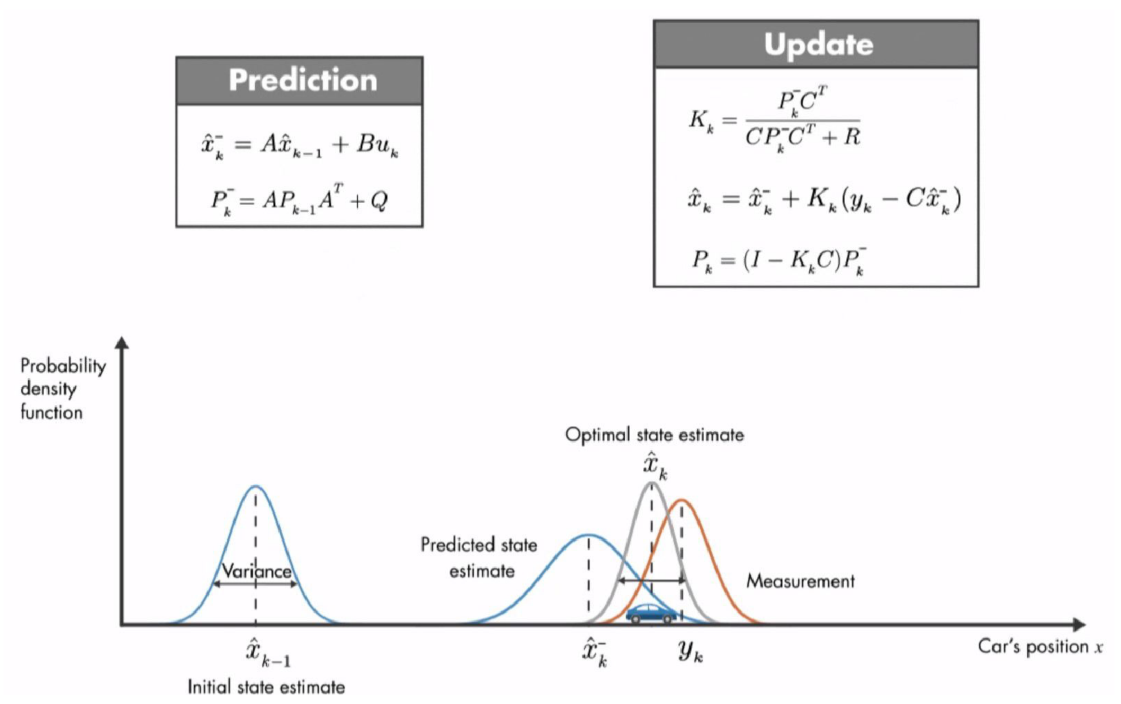
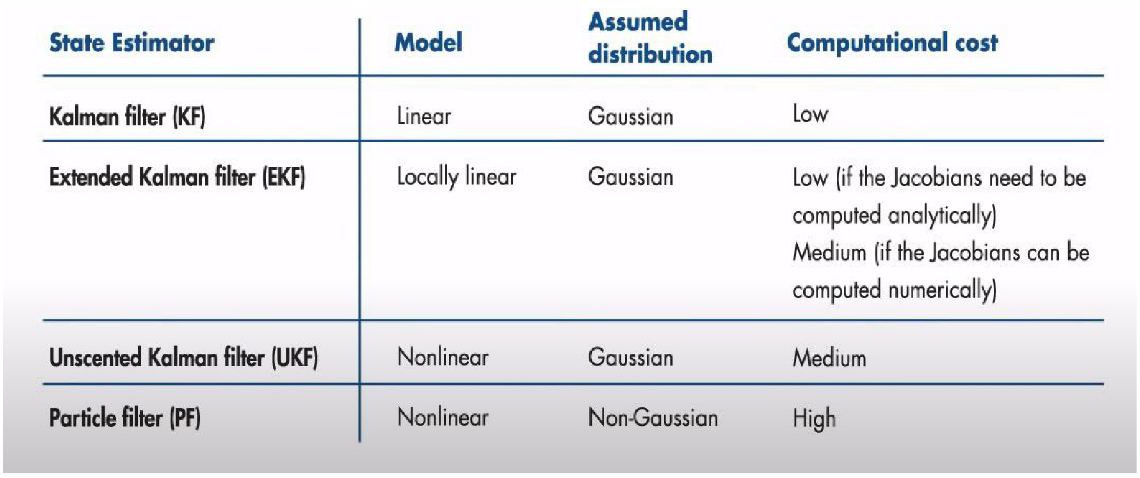


Figure 4.16: Kalman filter illustration. Prediction and estimation equations[39]

where $\hat{x_k}^-$ is the *priori estimate of $x$ at time step $k$*, $P_{k_k}^-$ is the *priori estimate of the error at time step $k$*, $Q$ is the *process variance*, $y_k$ is the *actual measured state*, $x_{k-hat}$ is the *posteri estimate of $x$ at time step $k$*, $P_k$is the *posteri estimate of the error at time step $k$*, $K_k$ is the *Kalman gain at time step $k$*, and $R$ is the *measurement variance.*

The graph in figure 4.16 explains the Kalman filter from the probability distribution point of view. Kalman filter procedure occurs in this manner. Firstly, the initial state estimate curve takes place. Then, the predicted state estimate in the blue curve occurs. After that, the measurement curve in orange is calculated. And finally, the optimal state estimate represented by the green curve is the multiplication result of the measurement Gaussian curve (in orange) and the Gaussian predicted state estimate curve (in blue).

There are many types and algorithms of Kalman filters. They vary in implementations depending on their applied use. However, they all share the same principle of prediction and estimation calculation. These types are either linear models or non-linear models. The more we tend to non-linear models, the more expensive the computational cost is. Figure 4.17 highlights some of Kalman filter types, use, assumed distribution, and costs.



| State Estimator | Model | Assumed distribution | Computational cost |
|---|---|---|---|
| Kalman filter (KF) | Linear | Gaussian | Low |
| Extended Kalman filter (EKF) | Locally linear | Gaussian | Low (if the Jacobians need to be computed analytically)<br>Medium (if the Jacobians can be computed numerically) |
| Unscented Kalman filter (UKF) | Nonlinear | Gaussian | Medium |
| Particle filter (PF) | Nonlinear | Non-Gaussian | High |

Figure 4.17: Some of Kalman filter types[39][40][41]

A stand-alone accelerometer or stand-alone gyroscope can be good to a certain level, but for a more 2D accurate system that can precisely give good angle results, the Kalman filter is an ideal choice where tilt angle, roll-pitch-yaw angles can be measured.[42] Roll, symbolized by $\phi$, is the rotation around the front-to-back axis (also known as north axis). Pitch, symbolized by $\theta$, is the rotation around the side-to-side axis (also known as the east axis). Yaw, symbolized by $\psi$, is the rotation around the vertical axis (also known as the down axis) as shown in figure 4.18. These angles are then fed back to the controller in order to do a certain controlling task in aviation, or automobile.

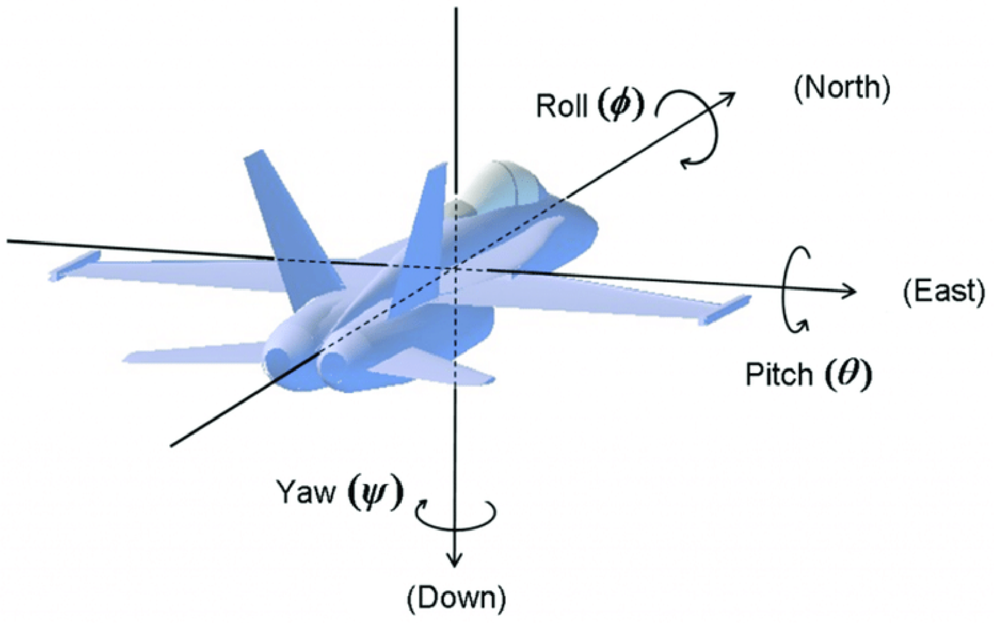


Figure 4.18: Roll pitch and yaw angles[43]

A Kalman filter for 2D orientation is implemented in Simulink. It is included in the appendix. It basically takes data from the gyroscope and accelerometer, merge them together, and processes them based on the Kalman filter approach. The graph in figure 4.20 shows the results of the roll angle.

### Kalman filter 3D

Fusing two sensors namely the accelerometer and gyroscope is a great success. However, it only applies to 2D applications. As soon as, the application starts to move in a third dimension, then it is affected and not accurate. Not only that but it is also disturbed by linear acceleration and any trivial vibrations. The problem point of view is that the system is only defined within two frames of reference. Therefore, the solution would be to add a third frame of reference namely the earth's magnetic field by a magnetometer.[44][38] One limitation of using Kalman filters is that it is only successful for linear systems, which is not usually the case. For non-linear systems, a modified version of the Kalman filter known as the Extended Kalman Filter is needed to be used.[45]

Moving from sensor fusion with two sensors to sensor fusion with three sensors is a huge jump that requires such an algorithm that would tackle the problem. This problem can be solved with EKF which is intended for non-linear systems. And for more accuracy and better behavior, the quaternion four-dimensional number is introduced.[46][47] The equation of this system is defined like this:

$$q = q_0 + q_1 * i + q_2 * j + q_3 * k \tag{4.6}$$

which can be understood as an extension of complex numbers, but in this case, it has one real number and three imaginary numbers.

*AHRS* or Attitude and Heading Reference System play a pivotal role in many applications like robotics, navigation, aviation, and human-machine analysis. Due to its precision, it is preferably used. It depends on 3 sensors, where each sensor is three axes X,Y, and Z.[48]

With the help of the Navigation Toolbox in MathWorks, multiple filters can be used for orientation detection based on quaternion like *ahrsfilter*. A code is done in Matlab for this detection for getting roll, pitch, and yaw. It is included in Appendix. The graph in figure 4.19 shows angles of roll pitch and yaw.

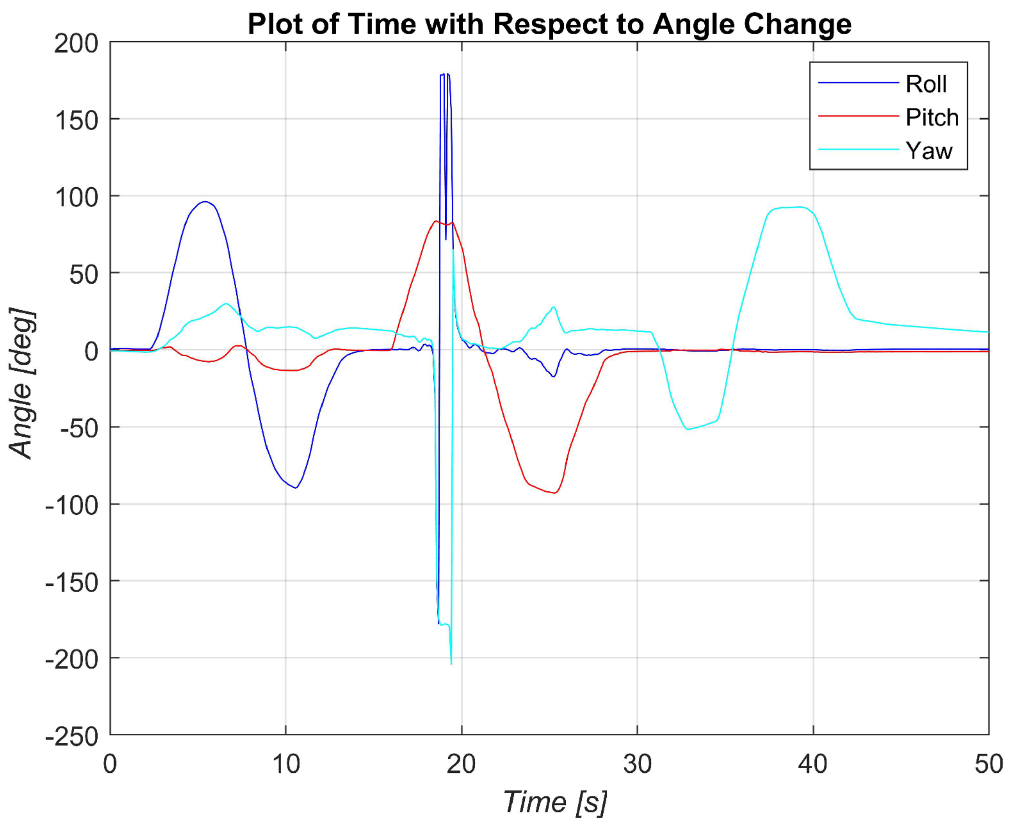


Figure 4.19: Orientations in MPU-9250 sensor[49]

### An experiment of different methods to detect angles

Figure 4.21 compares the experimental results of different methods for getting the roll angle. The experiment was done on an MPU 9250 sensor that is connected to a microcontroller and fixed on a box as shown in figure 4.20. The first method detects the angle with a stand-alone accelerometer, while the second method detects by a stand-alone magnetometer. The third method is achieved by a complementary filter, combining an accelerometer and magnetometer. The fourth and fifth ones are by Kalman filters. One is in 2D (Accelerometer + Gyroscope) and the other one is in 3D (Accelerometer + Gyroscope + Magnetometer).

The results that can be interpreted from figure 4.21 is that using an accelerometer alone can lead to noisy detected signals while using a gyro alone causes drifted results. The complementary filter and Kalman 2D results are quite similar. They are good in 2D applications, but as soon as there is a linear acceleration the results start to be noisy. Kalman 3D plot as shown in the figure is quite stable with less error and more precision even under linear acceleration.

The orientation and visualization of the 3D motion are also achieved in Simulink as shown in figure 4.22 with different animations. The code for the different methods as well as the orientation of the angles are included in the attachments.

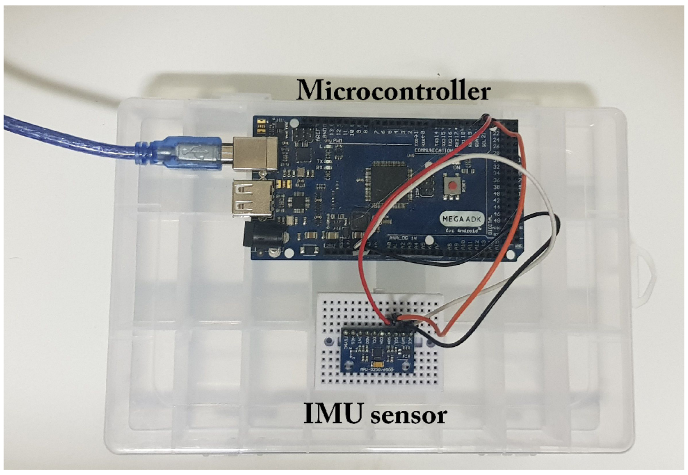


Figure 4.20: Orientations in MPU-9250 sensor

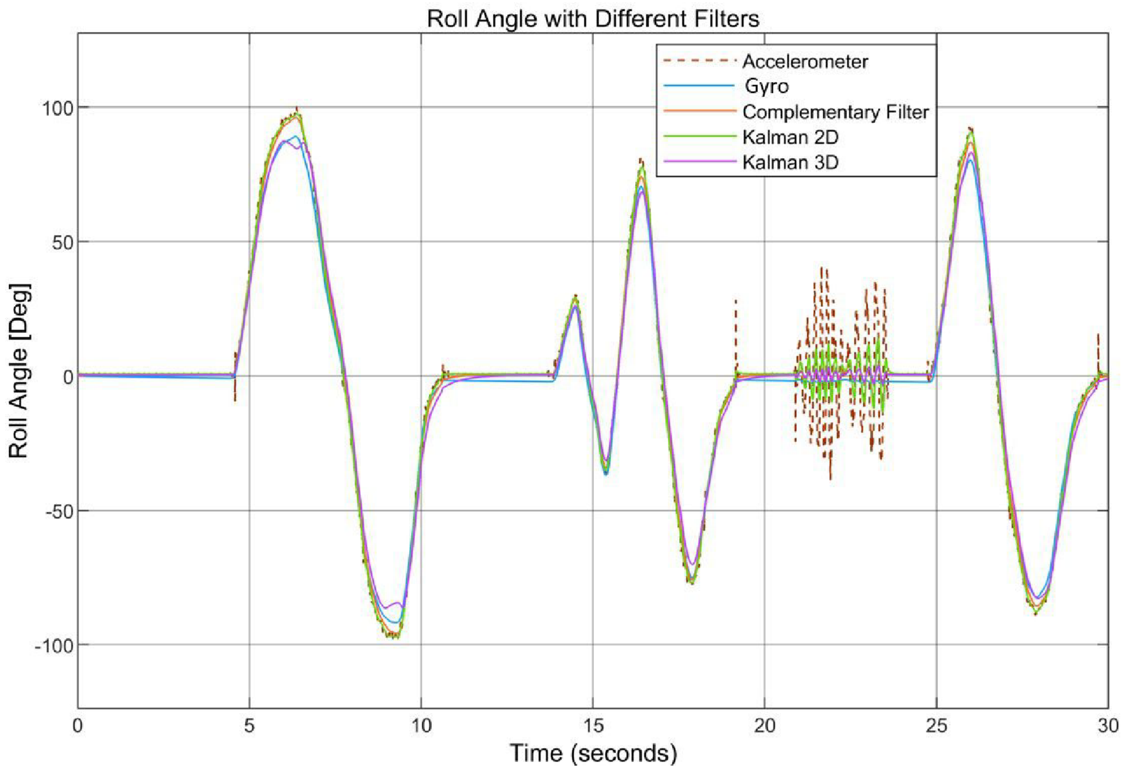


Figure 4.21: Orientations in MPU-9250 sensor

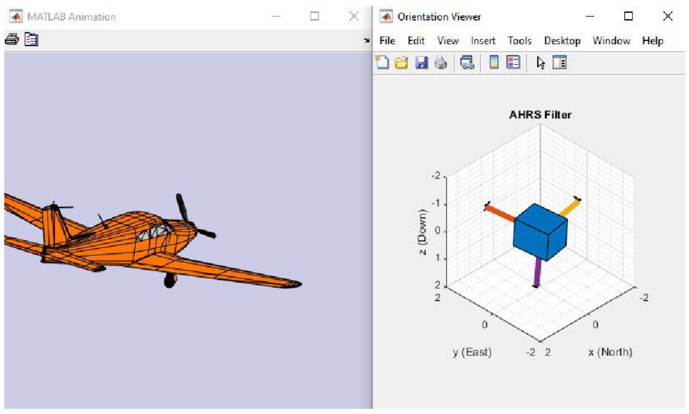


Figure 4.22: Orientations in MPU-9250 sensor

### 4.2.4 Worthwhile theoretical and practical notes

#### Alignment

Not all sensors in IMU are aligned. Therefore, it is important to first align the sensors before processing the data to the NDE coordinate system. As noticed in figure 4.23 the axis of the accelerometer-gyroscope is different from the magnetometer in MPU-9250. The accelerometer and the gyroscope axis need to be swapped and/or inverted to match the magnetometer axis.

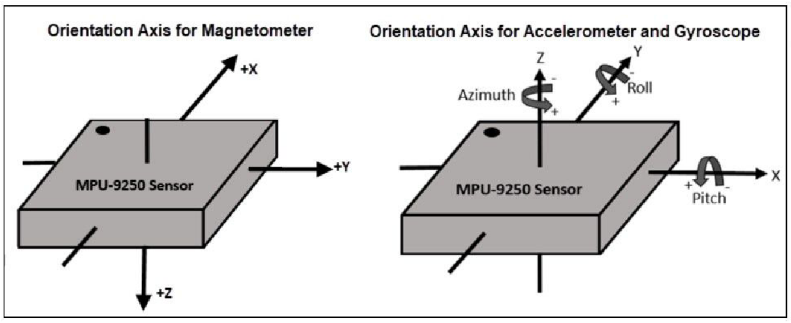

Orientation Axis for Magnetometer
+X
+Y
+Z
MPU-9250 Sensor
Orientation Axis for Accelerometer and Gyroscope
Z
Y
Azimuth
Roll
MPU-9250 Sensor
X
Pitch

Figure 4.23: Orientations in MPU-9250 sensor[49]

#### Discrete Euler

When working with microcontrollers, we need to discretize the system. The previous prediction and estimate equation is for continuous systems. Therefore, discretizing the system is an important step that should be taken into consideration. The following equation illustrates how to discretize an equation in a continuous system:

$$\dot{x}(t) = A\,x(t) + B\,u \tag{4.7}$$

is discretized like this:

$$x(K+1) = x\,(k) + T_s * (Ax\,(k) + Bu) \tag{4.8}$$

#### Quaternion over Euler angles

Quaternion system is preferred over Euler angles due to its more accuracy in 3D orientation visualization which is not vulnerable to bugs, gimbal lock, and other issues. The Euler method or a three-by-three matrix method is more intuitive to understand; however, it is in danger of gimbal lock. Gimbal lock is the case known when two axes of orientation are lined up which leads to one degree of freedom loss. Figure 4.24 shows the gimbal lock case. [50] All in all, the quaternion is more accurate but less intuitive and more computationally expensive.

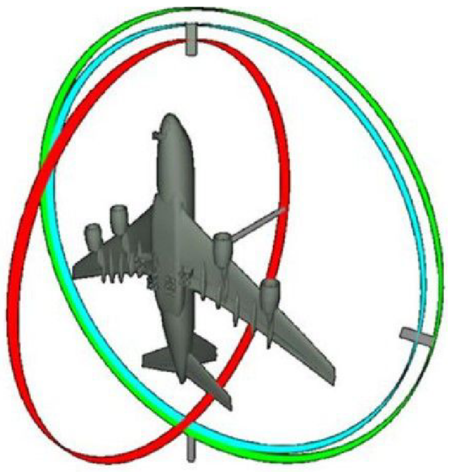


Figure 4.24: Gimbal lock case[50]

### EKF vs. ErKF

Kalman filters are sufficiently good for linear systems, but neither the environment model nor the process is usually linear. Therefore, in such cases, KF is replaced by EKF which is quite efficient for nonlinear systems. The *ahrsfilter* in Matlab is based on ErKF which is also used for non-linear systems. The mean difference between EKF and ErKF is that ErKF uses a technique where the error in the states is estimated using a Kalman filter, rather than the state itself. While in EKF the states are directly measured. Experiments have proven that ErKF is very robust method and effective.[51]

### Other notes

- Matrix operations are computationally costly.
- Filtering the sensors might be good before implementing them in KF.
- It is important to carefully initialize state estimate $x_{hat}$, error covarinace matrix $P$, and noise covariance matrices $R$ and $Q$.
- When programming Arduino in Simulink, there is an option of Running in IO, before the implementation and the code generation, which can save a lot of time instead of generating code each time with each modification.
- When working in Matlab or Simulink, all data mat files and functions must be in the same folder.

# 5 Control strategies and stability

## 5.1 PID controllers

### 5.1.1 Introduction

Proportional-integral-derivative is one of the most commonly used control strategies in the industry. Even though PID is a relatively old method compared with the new ones, it is still a leading control strategy that occupies 90-95% of the control industrial application. This dominance over the other control strategies can be summarized in this points[52]:

- PID is intuitively easy to grasp. It only requires a basic understanding of the working principle. In other words, it is not mathematically complicated.
- Its historical background, and early usage have led it to be a standard and a reference over time in the control theory.
- The appearance of digital control has facilitated the way into recognizable and remarkable improvements in the field. As the engineering main goal is to reach a better performance with less time, complexity, and price. Digital control has hugely helped PID in adaption, self-tuning, and gaining scheduling.
- It is always better to keep developing, modifying, and enhancing a strategy (well-known solution) in which you have solid knowledge, rather than starting in new strategy from scratch.

### 5.1.2 Fundamentals

A PID controller is basically a feedback controller strategy in which a desired value is given in order to be reached. Its main important factor is to precisely, roughly speaking, adjust/determine PID parameters to reach an optimum required value in a process known as parameters' tuning [53]. The PID controller, from how its name sounds, has three main parameters as shown in figure 5.1

So the controller part of the whole system can be mathematically represented by the following equation:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + k_d \frac{de(t)}{dt} \tag{5.1}$$

where $u(t)$ is the *output of PID*, $K_p$ is the *proportional gain*, $k_(i)$ is the *integration gain*, $k(d)$ is the *derivative gain*, $e(t)$ is the *error* which is the difference between output $y(t)$ and required value $\tau(t)$.[54]

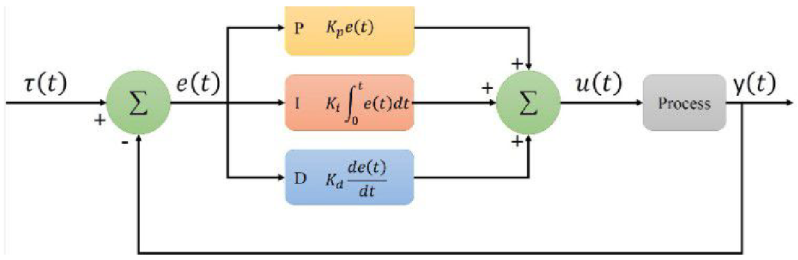


Figure 5.1: PID controller in time domain

Each term of the PID controller has a practical influence and a technical meaning[55]:

- The proportional part. It is proportional to the error. It speeds up the dynamic of response. In other words, it gets the system to the interned requirement as fast as possible, however, using it alone if not tuned properly can lead to instability.

- The integration part. It basically integrated the error, helping it gradually to reach zero offsets. Using it alone can make lagging and delays for the system.

- The derivative part. It basically restrains the system from moving too quickly in the beginning, and if it overpasses the required point, it will get it back. If not used properly, it can cause undesired high frequency.

Merging two or all of these terms together is just a task that requires knowledge of the system (plant) in order to be chosen properly. After selecting a specific two or three terms controller then it's a matter of tuning.

There are many tuning approaches that can be used in tuning PID controllers. The most known three ones are heuristic tuning, rule-based tuning, and Model-based tuning. Depending on the purpose, knowledge of the plant, and experience, the method can be chosen.[56]

## 5.2 State-space control

### 5.2.1 Introduction

State-space control, or what is often referred to in the literature as modern control theory, is a time-domain approach in which the system can have multiple inputs and multiple outputs. It is a more complicated approach that requires a deep understanding of the system as well as the approach. This type of approach can lead to a more stable system. With that being said, this approach is costly either due to the need for more sensors or due to its requirement for a more capable microcontroller to do the estimation of the non-sensored states by the observer.[57].

Mathematically speaking, the state-space representation can be defined in terms of matrices in this form:

$$\dot{x} = Ax + Bu \tag{5.2}$$

$$y = Cx + Du \tag{5.3}$$

where equation 5.2 is called *state equation*, and equation 5.3 is called *output equation*,
$x$ ... is the state vector
$\dot{x}$ ... derivative of the state vector with respect to time
$y$ ... output vector
$u$ ... input or control vector
$A$ ... system matrix
$B$ ... input matrix
$C$ ... output matrix
$D$ ... feed-forward matrix

### 5.2.2 Controllability and observability

A system is said to be fully controllable, if it is possible to transfer the system from any initial state $x(t_0)$ to any desired state $x(t)$ in specified finite time by a control vector $u(t)$. In other words, if any of the state variables is independent of the control input $u(t)$, there would be no possible way to bring the system to that desired state and so then the system is uncontrollable. To check controllability of any system then

- the rank of $Q_c$ should have the same value of the order of the matrix A.
- or determinant of of $Q_c$ should not be equal to 0.

$$Q_c = [B : AB : ... : A^{n-1}B] \tag{5.4}$$

If the system is in the controllable canonical form then, the system is always controllable. Likewise, A system is said to be fully observable, if every state $x(t_0)$ can be identified by measurement of outputs $y(t)$ over a finite time interval. For any system to be observable then

- the rank of $Q_o$ should have the same value of the order of the matrix A.
- or determinant of of $Q_o$ should not be equal to 0.

$$Q_o = [C : CA : ... : CA^{n-1}]^T \tag{5.5}$$

The practical importance of having an observable system is that if any of the states cannot be measured for any reason(cost, no possibility), then these states can be estimated by the observer if designed and parameterized accurately.[55][57][58]

### 5.2.3 Control strategies

A- Pole-placement method[59][55]

In the pole-placement approach, we have the choice freedom of the poles. The method is not very intuitive in presence of high-order systems. Since it is a state-feedback control then the control input is defined as:

$$u = -Kx \tag{5.6}$$

in which $u$ is the input, $k$ is the control vector, and $x$ is the state vector.

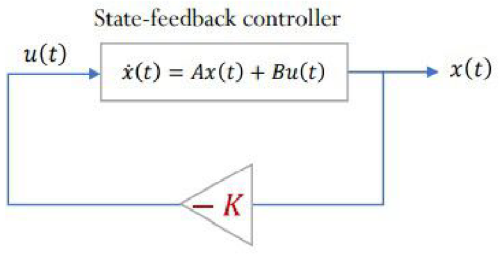


Figure 5.2: State_feedback control

B- LQR[59][55]

Linear quadratic regulator (LQR) is the optimal theory of the pole placement approach. It takes into consideration performance (states) and actuator effect. Its drawback is that it could be computationally expensive for high-dimensional systems.

In LQR approach, there are two matrices of cost functions which are Q and R. Q is the penalization given to the stats, while R is the penalization given to the inputs(actuators).

## 5.3 Modeling and controling

### 5.3.1 The plant

For creating the model we need to derive the equations of motion. This equation can be derived from Lagrangian mechanics or classic mechanics. The equation, that will be derived here, is based on classical mechanics.[60][61][62] Figure 5.3 can give a glimpse of how the equations will be driven. Basically for balancing the motorbike we need to formulate the equation of rotational motion around points A and B.

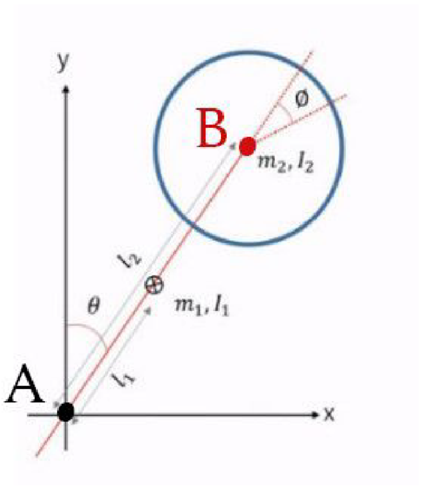


Figure 5.3: Free body diagram of the reaction wheel for balancing the wheel

The parameters of the model are defined like this:
$\theta$ ... angle of the bike
$\phi$ ... angle of the reaction wheel
$m_1$ ... mass of the bike
$m_2$ ... mass of the wheel
$I_1$ ... bike moment of inertia at the center of body
$I_2$ ... wheel moment of inertia at the center of wheel
$\tau_m$ ... torque of DC motor
$l_1$ ... distance to the center of body
$l_2$ ... distance to the center of wheel
$g$ ... acceleration of gravity
$v_0$ ... voltage applied v0 = U
$i$ ... electric current
$R_m$ ... motor resistance
$l_m$ ... motor inductance
$k$ ... electrical and mechanical constant of the motor
$\omega$ ...angular velocity of motor
$\tau_e$ ... electric torque of motor = $k.i$
$\tau_d$ ... static friction
$b$ ... viscous coefficient

The first equation will be the equation of motion around Point A:

$$[I_1 + m_1 l_1^2 + I_2 + m_2(l_1 + l_2)^2]\ddot{\theta} = [m_1 l_1 + m_2.(l_1 + l_2)]gsin(\theta) - \tau_m \tag{5.7}$$

The second equation will be the equation of motion around Point B:

$$I_2\ddot{\theta} + I_2\ddot{\phi} = \tau_m \tag{5.8}$$

The third part of the equations is the electrical equations and mechanical ones that represent the DC motor:

$$v_0 = R_m i + l_m \frac{di}{dt} + k\omega \tag{5.9}$$

$$\tau_e = \tau_m + b\omega + \tau_d \tag{5.10}$$

The system now based on the equations above can be represented with 4 states. But for sake of simplicity, the system is simplified and reduced to be represented by only 3 states which are, the position of bike ($\theta$), angular velocity of bike ($\dot{theta}$), and angular velocity of motor ($\dot{\phi} = \omega$). So after manipulating with equations (5.7-10), and after substituting:

$$I_t = [I_1 + m_1 l_1^2 + I_2 + m_2(l_1 + l_2)^2] \tag{5.11}$$

$$M = [m_1 l_1 + m_2.(l_1 + l_2)] \tag{5.12}$$

Then the main differential equations of the state-space model will be:

$$\ddot{\theta} = \frac{Mgl\sin(\theta)}{I_t} - \frac{kU}{R_m I_t} + \frac{k^2\omega}{R_m I_t} \tag{5.13}$$

$$\dot{\omega} = \frac{kU}{R_m I_2} - \frac{k^2\omega}{R_m I_2} - \frac{Mgl\sin(\theta)}{I_t} + \frac{kU}{R_m I_t} - \frac{k^2\omega}{R_m I_t} \tag{5.14}$$

so from equations (5.13-14), the equations of each state are as follows:

$$x_1 = \theta \Rightarrow \dot{x_1} = \dot{\theta} = x_2 \tag{5.15}$$

$$x_2 = \dot{\theta} \Rightarrow \dot{x_2} = \ddot{\theta} = \frac{Mgl\sin(x_1)}{I_t} - \frac{kU}{R_m I_t} + \frac{k^2 x_3}{R_m I_t} \tag{5.16}$$

$$x_3 = \omega \Rightarrow \dot{x_3} = \dot{\omega} = [\frac{k}{R_m I_2} + \frac{k}{R_m I_t}]U - [\frac{k^2}{R_m I_2} + \frac{k^2}{R_m I_t}]x_3 - \frac{Mgl\sin(x_1)}{I_t} \tag{5.17}$$

The system is non-linear and it can be stabilized around the equilibrium point.

## 5.3.2 linearization a non-linear system

Control theory can tackle linear systems ideally and perfectly. However, once the system is non-linear, then it is another case that would require other techniques. Since control theory is perfectly fine with linear systems, then we can linearize non-linear systems around the equilibrium point. And this is basically what is needed for balancing the bicycle, which is balancing the bicycle when the angle is just 0 or close.

The non-linear state-space system of the bicycle model, based on previous deriving, is represented in this form:

$$\dot{x} = \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} x_2 \\ \frac{Mgl\sin(x_1)}{I_t} - \frac{kU}{R_m I_t} + \frac{k^2 x_3}{R_m I_t} \\ (\frac{k}{R_m I_2} + \frac{k}{R_m I_t})U - (\frac{k^2}{R_m I_2} + \frac{k^2}{R_m I_t})x_3 - \frac{Mgl\sin(x_1)}{I_t} \end{pmatrix} \tag{5.18}$$

Then we equalize equation 5.18 to zero, to find the equilibrium point. After solving the equations then the point of equilibrium is when:

$$U^* = [0] \tag{5.19}$$

$$x^* = [0, 0, 0] \tag{5.20}$$

Now Jacobian matrix can be found from equation (5.18)

$$\delta\dot{x} = \begin{pmatrix} 0 & 1 & 0 \\ \frac{Mgl\cos(x_1)}{I_t} & 0 & (\frac{k^2}{R_m I_2}) \\ -\frac{Mgl\cos(x_1)}{I_t} & 0 & \frac{-k^2}{R_m I_t} - \frac{k^2}{R_m I_2} \end{pmatrix} \delta x + \begin{pmatrix} 0 \\ -\frac{k}{R_m I_t} \\ \frac{k}{R_m I_2} - \frac{k}{R_m I_t} \end{pmatrix} \delta u \tag{5.21}$$

After substituting $U^*$ and $x^*$ to equation(5.21), then the model state-space representation will be:

$$\delta\dot{x} = \begin{pmatrix} 0 & 1 & 0 \\ \frac{Mgl}{I_t} & 0 & \frac{k^2}{R_m I_2} \\ -\frac{Mgl}{I_t} & 0 & \frac{-k^2}{R_m I_t} - \frac{k^2}{R_m I_2} \end{pmatrix} \delta x + \begin{pmatrix} 0 \\ -\frac{k}{R_m I_t} \\ \frac{k}{R_m I_2} - \frac{k}{R_m I_t} \end{pmatrix} \delta u \tag{5.22}$$

For simplicity, and since all parameters of equation (5.22), are fixed constant values, then I would assume that:

$$a = \frac{Mgl}{I_t}; b = \frac{k^2}{R_m I_2}; c = -\frac{Mgl}{I_t}; d = \frac{-k^2}{R_m I_t} - \frac{k^2}{R_m I_2}; e = -\frac{k}{R_m I_t}; f = \frac{k}{R_m I_2} - \frac{k}{R_m I_t} \tag{5.23}$$

Then the equation (5.22) would be in this form:

$$\delta\dot{x} = \begin{pmatrix} 0 & 1 & 0 \\ a & 0 & b \\ c & 0 & d \end{pmatrix} \delta x + \begin{pmatrix} 0 \\ e \\ f \end{pmatrix} \delta u \tag{5.24}$$

And after substituting all parameter values we get:

$$\delta \dot{x} = \begin{pmatrix} 0 & 1 & 0 \\ 26.9187 & 0 & 0.0219 \\ -26.9187 & 0 & -0.0694 \end{pmatrix} \delta x + \begin{pmatrix} 0 \\ -0.5712 \\ 1.8130 \end{pmatrix} \delta u \tag{5.25}$$

The time-domain modeling as two states, three states, and four states is included in the attachments.

### 5.3.3 Discretizing a continuous system

Since our world is not continuously ideal and since we usually deal with digital systems like computers and microcontrollers, then discretization is a very important step. There are many techniques for how discretizing a system. One approach is to design a controller and then discretize it. Another approach is to discretize the plant and then design a discrete controller as shown in figure 5.4.

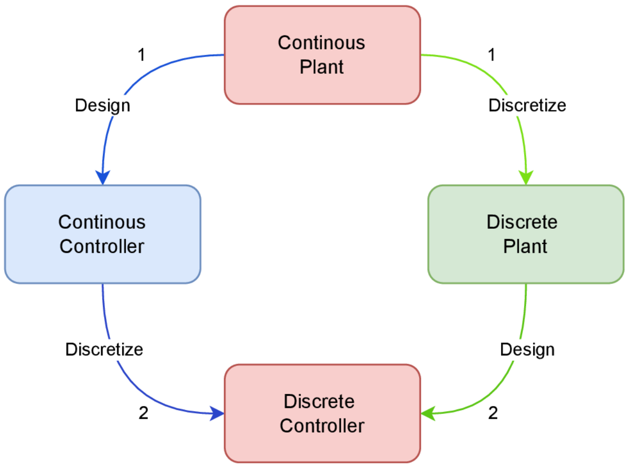


Figure 5.4: Approaches of discretizing a system

The discretization of the state space model is done according to the same principle in equation 4.8.

Figure 5.5 shows the output results of all three states of the model when the initial condition is $[0.1, 0, 0]$, and sample time $T_s = 0.01s$. The figure shows a comparison of the continuous and the discrete state-space model of the self-balancing motorbike model after linearizing and around the equilibrium point. The model is attached with *state space model 3 states* folder.

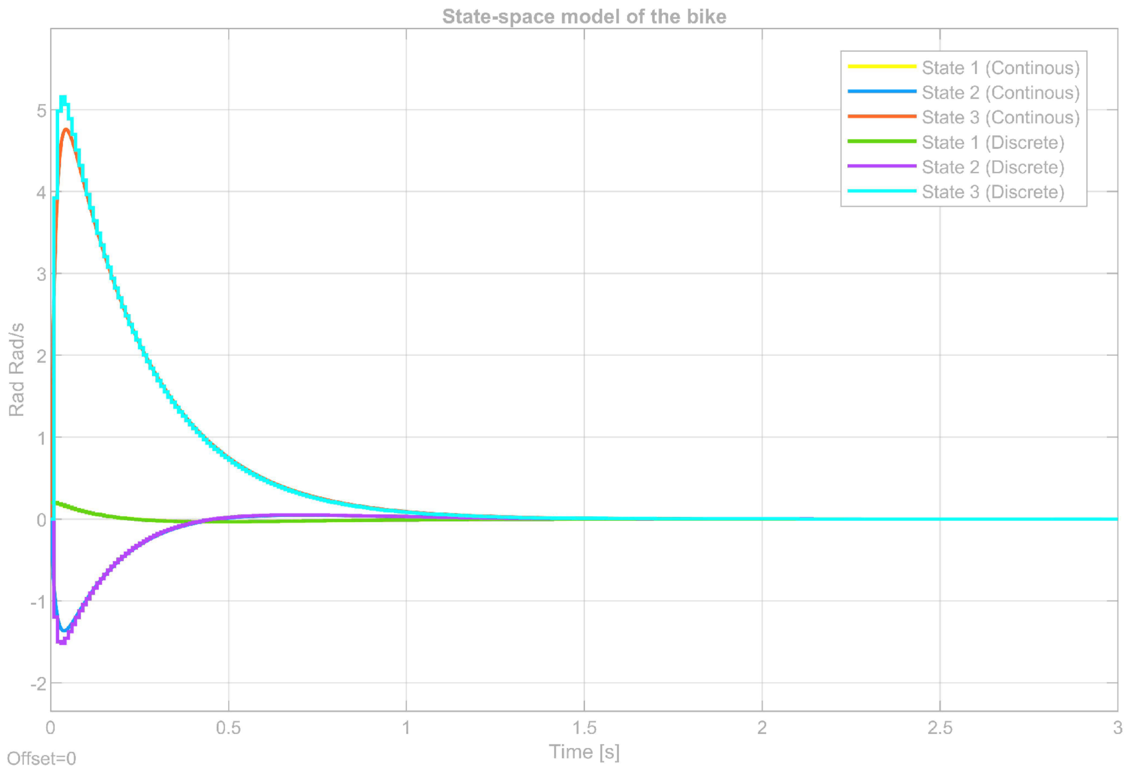


Figure 5.5: The output results of the continuous and discrete state-space model

### 5.3.4 Experimental results of the PID SBMB closed-loop control

The SBMB 3D model is shown in figure 6.1. The balancing of the SBMB prototype is accomplished in the Simulink environment as shown in figure 5.6. The closed-loop control includes multiple steps. Firstly, getting data from the IMU sensor. Secondly, including the calibrated model for the accelerometer and magnetometer. Thirdly, merging all data together with a 3D Kalman filter and getting quaternion data. Fourthly, converting from quaternion to roll, pitch, and yaw angles. In my case, only a roll angle was needed. Fifthly, feeding back the roll angle with the required value to the controller. Lastly, controlling the PWM of the motor will lead to controlling the reaction wheel motor voltage. And Controlling the voltage will lead to controlling the reaction wheel's angular velocity. And by all of that, we are in the end capable of giving required angular momentum that will stabilize the bike. The PID parameters that are used in the bicycle prototype are based more on experimental tuning. The PID values that were of the closed-loop control are as follows $K_p = 40$, $K_i = 0.001$, and $k_d = 0.1$ The potential of the prototype to control is around $5^0$ in both directions. The results of the required angle and the controlled one are shown in figure 5.7. The figure also in parallel shows the PWM value given to the motor at the same time of the balancing.

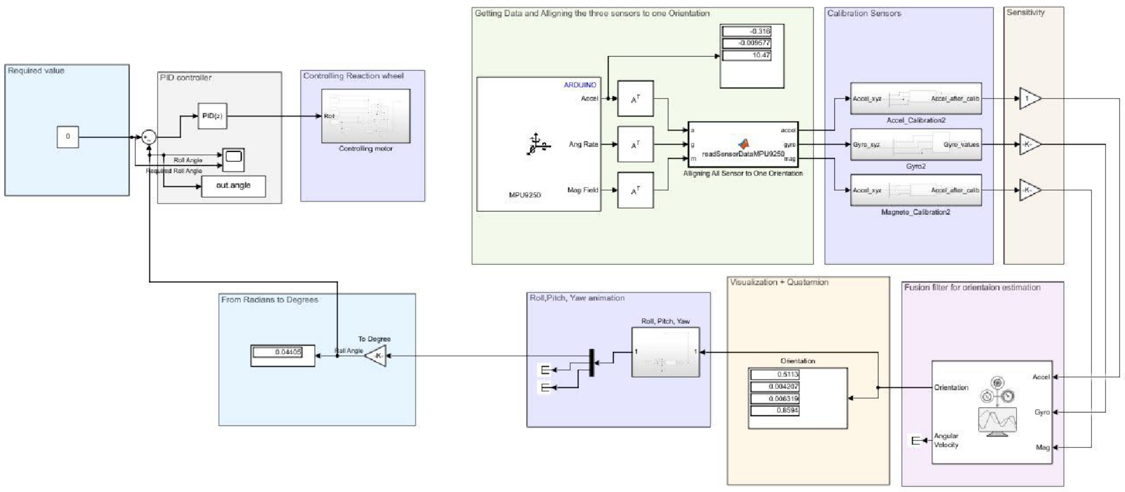


Figure 5.6: PID closed-loop control of the bike in Simulink

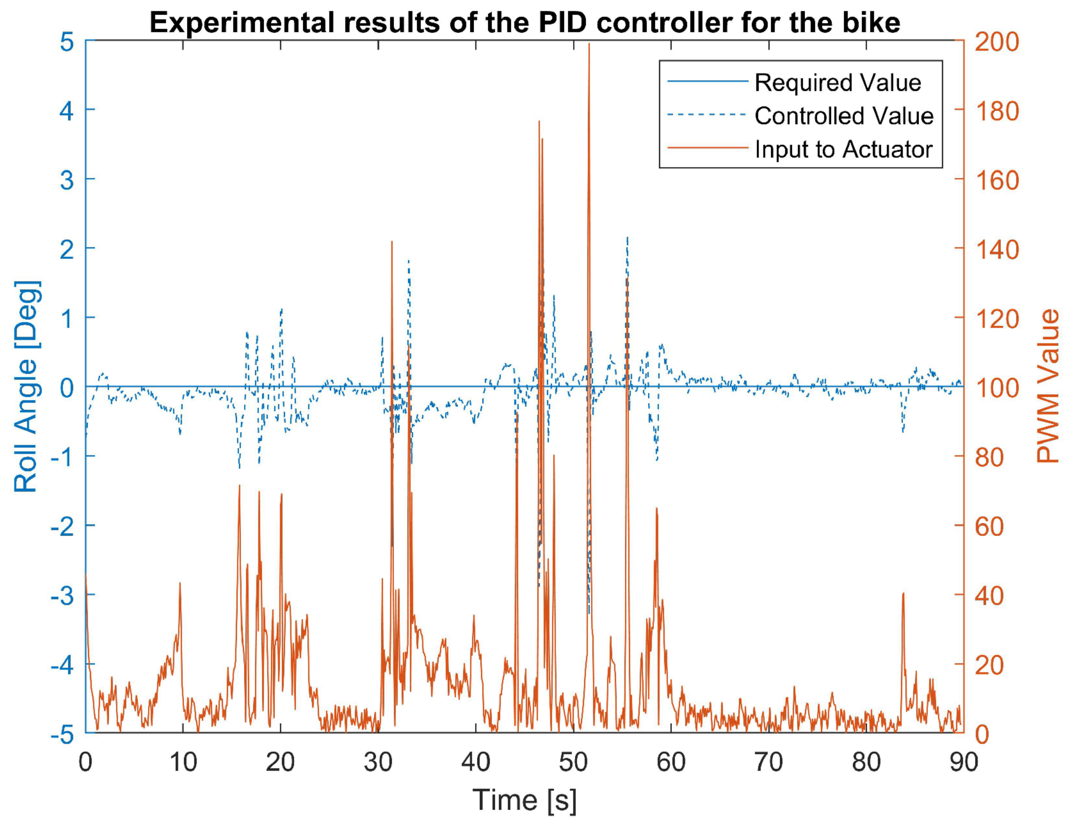


Figure 5.7: Results of the bike closed-loop PID control

# 6 Final product of the SBMB prototype

Figure 7.1 shows the final form of the SBMB prototype. The figure shows the mechanical and electrical aspects of the prototype. The mechanical side is represented by the body, reaction wheel, wheels, joints, bearings, and mechanism for steering. And the electrical components are microcontrollers, motors, IMU sensor, batteries, Bluetooth module, and motor driver.

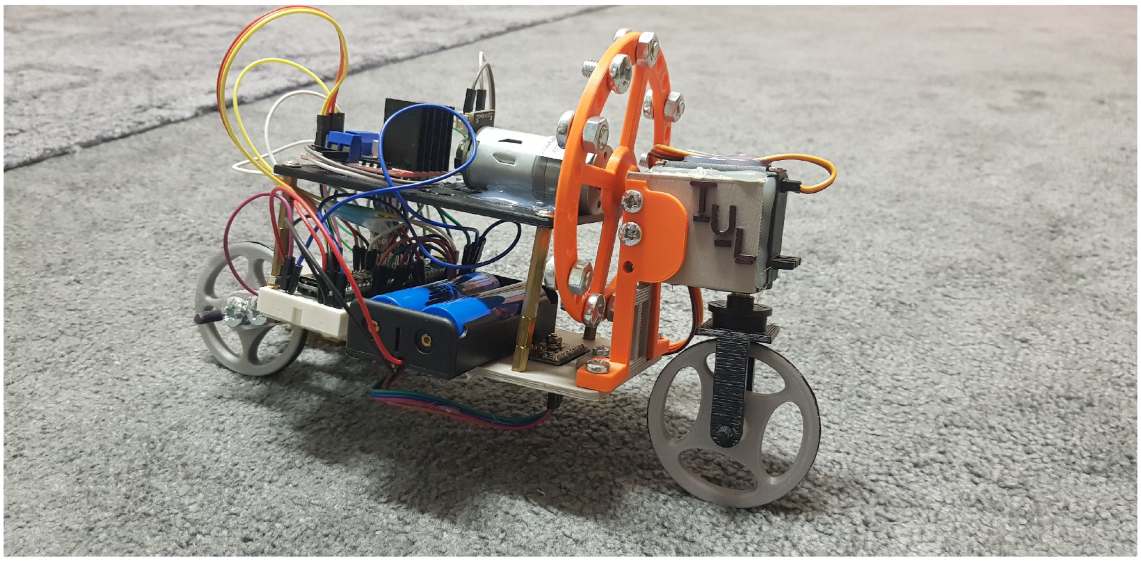


Figure 6.1: Final form of the SBMB prototype

# 7 Conclusion

## Summary

In this project, an analytical study of the different possibilities of balancing a two-wheeled vehicle has been accomplished. A proposed prototype is suggested with its different electrical and mechanical components.

The common different options, in which a bicycle can be automatically balanced discussed by this project are, by a reaction wheel, by the gyroscopic effect precession, or by a torque applied in the steering handlebar. The reaction wheel mechanism depends on the concept of the inverted pendulum. Its advantages are simplicity and robustness, while its disadvantage is more energy-consuming. GPE method can be applied in different forms and in different ways of balancing. Among all mechanisms mentioned, GPE method is considered the most robust but at the same time the most consuming method for energy. The torque applied to the steering handlebar method can be done by a rotational motor or an actuator. Its main advantage is, that it is the least method that consumes energy; however, it is not very stable.

The method that is chosen to continue further with is the reaction wheel method. To get a feedback signal from the sensors, an investigation of different methods and approaches to detecting the angle has been approached. The different methods are summarized in the following paragraph.

For detecting the angle, IMU 9250 sensor is selected. IMU sensor include 3 sensors which are an accelerometer, magnetometer, and gyro. Each sensor of those can detect in the directions x,y, and z. Therefore, there can be different scenarios and methods of harnessing and getting advantage of the sensors. For example, when using an accelerometer alone to detect the angle, then the results show a lot of noise in the short-term change. Secondly, when using gyro alone, the error keeps accumulating due to its dependence on discrete integration. So the drift effect is quite noticeable. To get the most out of each sensor then they can be combined.

Sensor fusion is a technique where more than one sensor can be merged to get more correct precise output results. For this application, three different methods are used. Firstly, a complementary filter between the gyro and accelerometer yields better results. The basic idea of the technique is to make the result depend on the accelerometer in long-term changes, and depend on the gyro in short-term changes. Secondly, Kalman filter between gyro and accelerometer. The difference between the complementary filter and the Kalman filter is that the $\alpha$ constant has to be chosen upfront in the complementary filter, while in the Kalman filter, it is automatically changed by the method. In my case, both of the methods have shown quite similar results. Merging two sensors is only enough for 2D applications for example a pendulum. However, when there is a need for 3D applications like drones, or balancing bicycles with a reaction wheel, then a third frame of reference is required.

The last sensor fusion method that is investigated is merging the three sensors together which are the accelerometer, gyro, and magnetometer. This method is accurately correct but costly. It can be used for AHRS applications and also for the bicycle balancing prototype. Results of the roll, pitch, and yaw angle are shown with different animations.

Of course not all sensors are ideal, so calibrations to the sensors are also achieved with different models.

The two 3D models are designed within Inventor. The models include the body of the vehicle, wheels, and other mechanical components. The model is printed then with 3D printing machines.

All in all, the bicycle prototype is controlled with a reaction wheel as an actuator of balancing which is driven by a DC motor, and the detection of the angle is accomplished with an IMU sensor using a 3D Kalman filter. To close the loop with the controller, a PID is chosen and used for the balancing and the bicycle is controlled by a mobile through the communication of a Bluetooth module. Modeling the plant with a state-space model and its LQR feedback controller is also investigated in this project.

### Further improvements and suggestions

A suggestion that can be investigated further is to control the reaction wheel with a motor that has a feedback signal, for example, of an encoder or tachometer. Then the states of the plant according to the model studied, are all available and the whole system can be controlled with LQR or other techniques and compared with PID strategies.

Also, for sake of simplicity or complexity, the plant model can be remodeled with two or four states depending on purpose and accuracy and therefore can be investigated and compared in reality.

Moreover, all parameters of the model are either taken as information from the supplier or approximated. So once the whole model is available with all signals, it would be good to do parameter estimation to all parameters to be more accurate of the model.

Furthermore, this whole process was done with Matlab and Arduino. So, it would be nice if the same techniques are investigated further with different software and hardware to make comparisons, and reach optimal results with more efficiency and less cost.

All in all, improvements of the whole system are still open in all its different aspects which are design, control, sensors, actuators, modeling, filtering, coding, theory, practice, and so on. So I would say for anyone who wants to push himself/herself forward, and continue further in this project with such a project, it is really attractive with challenge and enjoyment.

# References

[1] GUPTA, Nikhil Kumar; AMBIKAPATHY, A. *Self-Balancing Bicycle using Reaction Wheel*. Greater Noida, 2019. Available also from: https://ijesc.org/upload/302a95ee2491b0403a9bd03d1f0be541.Self-Balancing%20Bicycle%20using%20Reaction%20Wheel%20(3).pdf.

[2] BOUBAKER, Olfa. *Self-Balancing Bicycle using Reaction Wheel*. Tunis Cedex, 2012. Available also from: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6360606.

[3] ABDURRAHMAN1, Umar Tsani; SUKAMTO1, Pria; SOBARNAS1, Mohamad Anas; MUJIARTO2. *A Design for Self Balancing Scale Model Bicycle*. Indonesia, 2020. Available also from: https://iopscience.iop.org/article/10.1088/1742-6596/1764/1/012171/pdf.

[4] JEPSEN, Frank; SØBORG, Anders; PEDERSEN, Anders R.; YANG, Zhenyu. *Development and Control of an Inverted Pendulum Driven by a Reaction Wheel*. Esbjerg, 2009. Available also from: https://www.researchgate.net/publication/224591601_Development_and_Control_of_an_Inverted_Pendulum_Driven_by_a_Reaction_Wheel.

[5] *MURATA BOY* [online]. 2005. [visited on 2021-05-10]. Available from: https://corporate.murata.com/more_murata/robots/mboy.

[6] IVANOVA, Malinka Spasova; NAKOVA, Manuela Marinova; IVANOV, Ivan Nikolaev; DENISHEV, Krassimir Hristov. *Investigation and Analysis of Angular Gyroscope Microelectromechanical Systems*. Sofia, 2009. Available also from: https://www.academia.edu/27261573/Investigation_and_Analysis_of_Angular_Gyroscope_Microelectromechanical_Systems.

[7] JIN, Hongzhe; YANG, Decai; LIU, Zhangxing; ZANG, Xizhe; LI, Ge; ZHU, Yanhe. *A Gyroscope-Based Inverted Pendulum with Application to Posture Stabilization of Bicycle Vehicle*. Zhuhai, 2015. Available also from: https://ieeexplore.ieee.org/document/7419084.

[8] YETKIN, Harun; KALOUCHE, Simon; VERNIER, Michael; COLVIN, Gregory; REDMILL, Keith; OZGUNER, Umit. *Gyroscopic Stabilization of an Unmanned Bicycle*. Portland, Oregon, 2014. Available also from: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6859392.

[9] *Gyro-X-1967* [online]. 2009. [visited on 2021-05-10]. Available from: https://corporate.murata.com/more_murata/robots/mboy.

[10] METER, T.Van; SAHA, Dr. Mrinal. *Gyroscopic Stabilization of a Recumbent Bicycle*. Norman, OK, 2013. Available also from: https://s3images.coroflot.com/user_files/individual_files/534842_GFNWpAnYBNeyMdbq4psdCdyCi.pdf.

[11] WALCK, Chris Joel. *Stabilization of an inverted pendulum using control moment gyros*. 2013. Available also from: https://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=4155&context=etd. Master of Scince. Iowa State University, Iowa.

[12] *Gyroscopic stabilization of unstable vehicles* [online]. 2017. [visited on 2021-05-10]. Available from: https://www.slideshare.net/ParthPatel747/gyroscopic-stabilization-of-unstable-vehicles.

[13] FAWAZ, Ziad. *Design and Control of a Self-balancing Bicycle Using an Electric Linear Actuator*. 2019. Available also from: https://deepblue.lib.umich.edu/bitstream/handle/2027.42/148871/MastersThesis_FinalDraft%20(3).pdf?sequence=1. Master of Scince. University of Michigan-Dearborn. Supervised by Dr. Paul MUENCH.

[14] GALLASPY, J.M. *Gyroscopic stabilization of an Unmanned Bicycle*. 1999. Master of Scince. Auburn University, AL.

[15] *Introduction to Actuators Primary Knowledge*. May 2017. Available also from: https://nanohub.org/resources/26598/download/Actuators_PK_PG.pdf.

[16] ÖZER, Tolga; KIVRAK, Sinan; OĞUZ, Yüksel. *H Bridge DC Motor Driver Design and Implementation with Using dsPIC30f4011*. 2007. Available also from: https://www.researchgate.net/publication/317225711_H_Bridge_DC_Motor_Driver_Design_and_Implementation_with_Using_dsPIC30f4011.

[17] HALIT, Eren. *Sensors, The Engineering Handbook*. Pearth, Australia, 25 June 2004. Available also from: https://www.researchgate.net/publication/295072199_Sensors.

[18] EL-FATATRY, Ayman. *Inertial Measurement Units – IMU*. Chelmsford, UK, Feb 2004. Available also from: https://www.sto.nato.int/publications/STO%20Educational%20Notes/RTO-EN-AVT-105/EN-AVT-105-04.pdf.

[19] ILHAM, Faisal; TITO, Purboyo; ANTON, Ansori. *A Review of Accelerometer Sensor and Gyroscope Sensor in IMU Sensors on Motion Capture*. Indonesia, 10 Nov 2019. Available also from: https://www.sto.nato.int/publications/STO%20Educational%20Notes/RTO-EN-AVT-105/EN-AVT-105-04.pdf.

[20] DADAFSHAR, Majid. *Accelerometer and Gyroscopes Sensors: Operation, Sensing, and Applications*. 2014. Available also from: https://pdfserv.maximintegrated.com/en/an/AN5830.pdf.

[21] ARDAKANI, H. Alemi; BRIDGES, T. J. *Review of the 3-2-1 Euler Angles*. University of Surrey, UK, 15 April 2010. Available also from: http://personal.maths.surrey.ac.uk/T.Bridges/SLOSH/3-2-1-Eulerangles.pdf.

[22] PEDLEY, Mark. *Tilt Sensing Using a Three-Axis Accelerometer*. 03 June 2013. Available also from: https://www.nxp.com/files-static/sensors/doc/app_note/AN3461.pdf.

[23] VITTORIO, Passaro; ANTONELLO, Cuccovillo; LORENZO, Vaiani; MARTINO, De Carlo; CAMPANELLA, Carlo Edoardo. *Gyroscope Technology and Applications: A Review in the Industrial Perspective*. 07 October 2017. Available also from: https://www.researchgate.net/publication/320292459_Gyroscope_Technology_and_Applications_A_Review_in_the_Industrial_Perspective.

[24] D. B. PENGRA, y; STOLTENBERG, J.; DYCK, R. Van; VILCHES, O. *The Hall Effect*. 19 June 2015. Available also from: https://courses.washington.edu/phys431/hall_effect/hall_effect.pdf.

[25] *Magnetometer, Space Instrumentation and Observation*. 24 February 2011. Available also from: https://irsl.ss.ncu.edu.tw/media/course/106%E5%B9%B4%E7%AC%AC1%E5%AD%B8%E6%9C%9F%E4%B8%AD%E5%A4%AE%E5%A4%A7%E5%AD%B8%E5%A4%AA%E7%A9%BA%E6%B8%AC%E8%A8%88%E5%8F%8A%E6%93%8D%E4%BD%9CI/SIO_9.pdf.

[26] LAVENDER, Jiang. *Yaw Angle Estimation from Gyroscope and Magnetometer Based on Kalman Filter*. 03 May 2018. Available also from: https://www.researchgate.net/publication/340630361_Yaw_Angle_Estimation_from_Gyroscope_and_Magnetometer_Based_on_Kalman_Filter.

[27] STMICROELECTRONICS. *Parameters and calibration of a low-g 3-axis accelerometer*. 2014. Available also from: https://www.st.com/resource/en/application_note/an4508-parameters-and-calibration-of-a-lowg-3axis-accelerometer-stmicroelectronics.pdf.

[28] PEDRO, Batista; CARLOS, Silvestre; PAULO, Oliveira; BRUNO, Cardeira. *Accelerometer Calibration and Dynamic Bias and Gravity Estimation: Analysis, Design, and Experimental Evaluation*. 01 October 2011. Available also from: https://www.researchgate.net/publication/224180653_Accelerometer_Calibration_and_Dynamic_Bias_and_Gravity_Estimation_Analysis_Design_and_Experimental_Evaluation.

[29] SALEHI SARVENAZ Mostofi Navid, Bleser Gabriele. *A practical in-field magnetometer calibration method for IMUs*. 01 October 2011. Available also from: https://www.researchgate.net/publication/258449504_A_practical_in-field_magnetometer_calibration_method_for_IMUs.

[30] KOK, Manon; SCHON, Thomas B. *Accelerometer Calibration and Dynamic Bias and Gravity Estimation: Analysis, Design, and Experimental Evaluation*. 15 July 2016, available also from: https://ieeexplore.ieee.org/abstract/document/7470259.

[31] P.K., Varshney. *Multisensor Data Fusion*. 2000. Available also from: https://www.researchgate.net/publication/221047679_Multisensor_Data_Fusion.

[32] LUNDQUIST, Christian. *Sensor Fusion for Automotive Applications*. 2011. Available also from: http://liu.diva-portal.org/smash/get/diva2:451021/FULLTEXT01.pdf. Master of Scince. Linköping University, Sweden.

[33] GUSTAFSSON, F. *Statistical Sensor Fusion*. 2013. Available also from: https://stanfordasl.github.io/aa274a/pdfs/notes/lecture16and17.pdf.

[34] WALTER T. HIGGINS, JR. *A Comparison of Complementary and Kalman Filtering*. MAY 1975. Available also from: https://www.allaboutcircuits.com/uploads/articles/A_comparison_of_complementary_and_kalman_filtering.pdf.

[35] EUSTON MARK Paul Coote, Robert Mahony. *A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV*. 26 October 2008. Available also from: https://www.researchgate.net/publication/224339663_A_Complementary_Filter_for_Attitude_Estimation_of_a_Fixed-Wing_UAV.

[36] ARIFFIN NUR HAZLIZA Arsad Norhana, Bais Badaria. *Low cost MEMS gyroscope and accelerometer implementation without*. 01 Nov 2016. Available also from: https://www.researchgate.net/publication/315914729_Low_cost_MEMS_gyroscope_and_accelerometer_implementation_without_Kalman_Filter_for_angle_estimation.

[37] MATHWORKS. *Understanding Kalman Filters*. 2022. Available also from: https://www.mathworks.com/videos/understanding-kalman-filters-part-1-why-use-kalman-filters--1485813028675.html.

[38] MOHAMMED, Haider; ISLAM, Tariqul; ISLAM, Saiful; MAHMUD, Shajid. *Comparison of Complementary and Kalman Filter Based Data Fusion for Attitude Heading Reference SystemV*. 22 February 2017. Available also from: https://www.researchgate.net/publication/320726643_Comparison_of_Complementary_and_Kalman_Filter_Based_Data_Fusion_for_Attitude_Heading_Reference_System.

[39] RIKISENIA.L. *ATTITUDE DETERMINATION WITH QUATERNION USING EXTENDED KALMAN FILTER*. 2019. Available also from: https://thepoorengineer.com/en/angle-control-absolute/.

[40] SIMON, Dan. *Using Nonlinear Kalman Filtering to Estimate Signals*. 2006. Available also from: https://academic.csuohio.edu/simond/pubs/ESDNonlinear.pdf.

[41] WAN, Eric A.; MERWE, Rudolph van der. *The Unscented Kalman Filter for Nonlinear Estimation*. 2000. Available also from: https://groups.seas.harvard.edu/courses/cs281/papers/unscented.pdf.

[42] HERMAN, Veriñaz; EDGAR, Vela; RONALD, Ponguillo. *A Dual Extended Kalman Filter for Tilt Estimation*. 07 July 2017. Available also from: https://www.researchgate.net/publication/315999025_A_Dual_Extended_Kalman_Filter_for_Tilt_Estimation.

[43] ROMANI, Andrea Civita Simone Fiori Giuseppe. *A Mobile Acquisition System and a Method for Hips Sway Fluency Assessment*. Dec 2018. Available also from: https://www.researchgate.net/figure/An-illustration-of-the-three-angles-yaw-pitch-and-roll-returned-by-the-Hyper-IMU_fig1_329603549.

[44] RIKISENIA.L. *ATTITUDE DETERMINATION WITH QUATERNION USING EXTENDED KALMAN FILTER*. 2018. Available also from: https://thepoorengineer.com/en/attitude-determination/.

[45] ASCORTI, Leonardo. *An application of the extended Kalman filter to the attitude control of a quadrotor*. 24 July 2013. Available also from: https://www.politesi.polimi.it/handle/10589/80681. Master of Scince. POLITECNICO DI MILANO Corso di Laurea Magistrale in Computer Engeneering. Supervised by Prof. Matteo MATTEUCCI.

[46] RIKISENIA.L. *Quaternions and Rotations in 3-Space: The Algebra and its Geometric Interpretation*. 23 June 2001. Available also from: https://www.researchgate.net/publication/2378474_Quaternions_and_Rotations_in_3-Space_The_Algebra_and_its_Geometric_Interpretation.

[47] JIA, Yan-Bin. *Quaternions and Rotations*. 10 September 2013. Available also from: https://physique.cmaisonneuve.qc.ca/svezina/mat/note_mat/Quaternions-Yan-Bin_Jia-2019.pdf.

[48] KRZYSZTOF, Kolanowski; ALEKSANDRA, Swietlicka; MATEUSZ, Majchrzycki; KAROL, Gugała; KARO ´ N IGOR, Andrzej Rybarczyk. *Nine-Axis IMU sensor fusion using the AHRS algorithm and neural networks*. 2013. Available also from: https://dspace5.zcu.cz/bitstream/11025/11492/1/Krzysztof.pdf.

[49] MATHWORKS. *Estimating Orientation Using Inertial Sensor Fusion and MPU-9250*. 2022. Available also from: https://www.mathworks.com/help/fusion/ug/Estimating-Orientation-Using-Inertial-Sensor-Fusion-and-MPU-9250.html?s_eid=PSM_15028.

[50] TANNOUS, Halim. *Interactive and connected rehabilitation systems for e-health*. October 2018. Available also from: https://www.researchgate.net/publication/332394380_Interactive_and_connected_rehabilitation_systems_for_e-health.

[51] VENKATESH, Madyastha; VISHAL, Ravindra; SRINATH, Mallikarjunan; GOYAL, Anup. *Extended Kalman Filter vs. Error State Kalman Filter for Aircraft Attitude Estimation*. 08 August 2011. Available also from: https://www.researchgate.net/publication/269254094_Extended_Kalman_Filter_vs_Error_State_Kalman_Filter_for_Aircraft_Attitude_Estimation/citation/download.

[52] KNOSPE, Carl. *PID control*. March 2006. Available also from: https://www.researchgate.net/publication/3207700_PID_control.

[53] BANSAL, Hari om. *Tuning of PID Controllers using Simulink*. January 2009. Available also from: https://www.researchgate.net/publication/268802558_Tuning_of_PID_Controllers_using_Simulink.

[54] VISIOLI, Antonio. *Research Trends for PID Controllers*. January 2012. Available also from: https://www.researchgate.net/publication/285766672_Research_Trends_for_PID_Controllers.

[55] MODRLÁK, Osvald; HUBKA, Lukáš. *Automatic Control in Mechatronics*. 1st ed. Liberec: Technical University of Liberec, Faculty of Mechatronics, 2014. ISBN 978-80-7494-175-7. Available also from: https://dspace.tul.cz/handle/15240/7156?show=full.

[56] TOOLS, inca. *The Guide of PID tuning*. 2022. Available also from: https://www.incatools.com/pid-tuning/pid-tuning-methods/.

[57] NISE, Norman S. *Control Systems Engineering*. 7th ed. California State Polytechnic University, Pomona.: WILLEY, 2015. ISBN 2014037468. Available also from: https://oiipdf.com/control-systems-engineering-7th-ed-nise.

[58] TRIVEDI, Tapan. *Controllability and Observability*. January 2018. Available also from: https://www.researchgate.net/publication/340333667_Controllability_and_Observability.

[59] RAZMJOOY, Navid; A, Madadi; ALIKHANI, Hamid; MOHSENI, Mona. *Comparison of LQR and Pole Placement Design Controllers for Controlling the Inverted Pendulum*. July 2014. Available also from: https://www.researchgate.net/publication/340333667_Controllability_and_Observability.

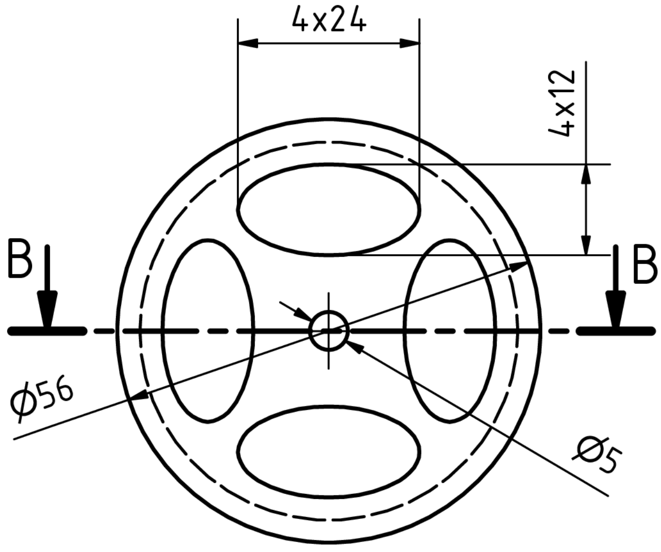
[60] B, Mrs.Vaishnavi; C.P, Mrs.Thamilselvi; J, Mr.Hiran Gabriel D; DR.KANCHANA; V, Mr.Balaganesh. *DEVELOPMENT OF INTELLIGENT SELF-BALANCING E-BIKE USING MACHINE LEARNING*. 2020. Available also from: https://www.sciencegate.app/source/305682.

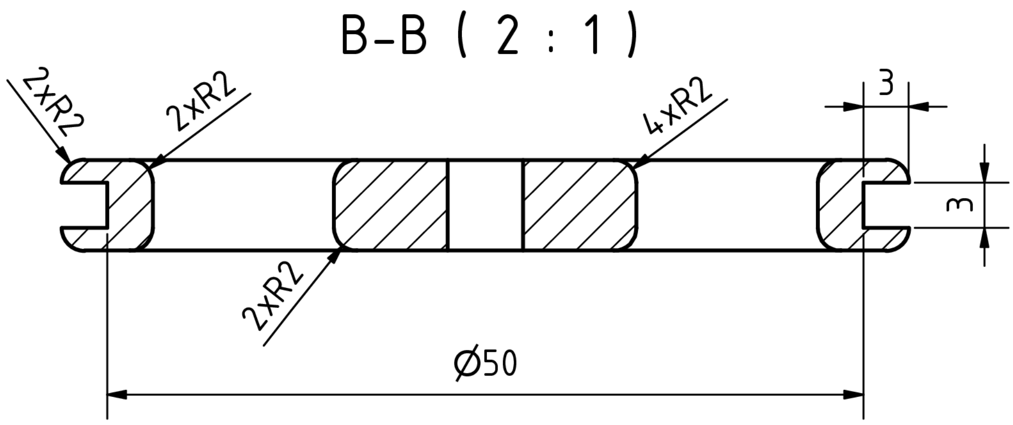
[61] YANG, Zhenyu. *Development and Control of an Inverted Pendulum Driven by a Reaction Wheel*. September 2009. Available also from: https://www.researchgate.net/publication/224591601_Development_and_Control_of_an_Inverted_Pendulum_Driven_by_a_Reaction_Wheel.

[62] BELASCUENNAHUEL, Gonzalo; AGUILAR, AguilarNahuel. *Design, Modeling and Control of a Reaction Wheel Balanced Inverted Pendulum*. Available also from: https://www.researchgate.net/publication/331271615_Design_Modeling_and_Control_of_a_Reaction_Wheel_Balanced_Inverted_Pendulum.

# Appendix

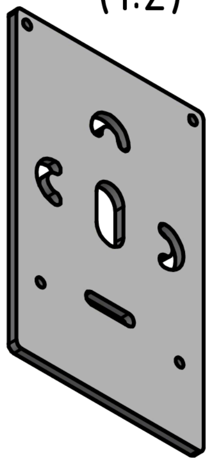
- 2D Drawings: Main components.
- Card Schematic and Connections.
- Main codes: Matlab, Simulink, and Arduino.

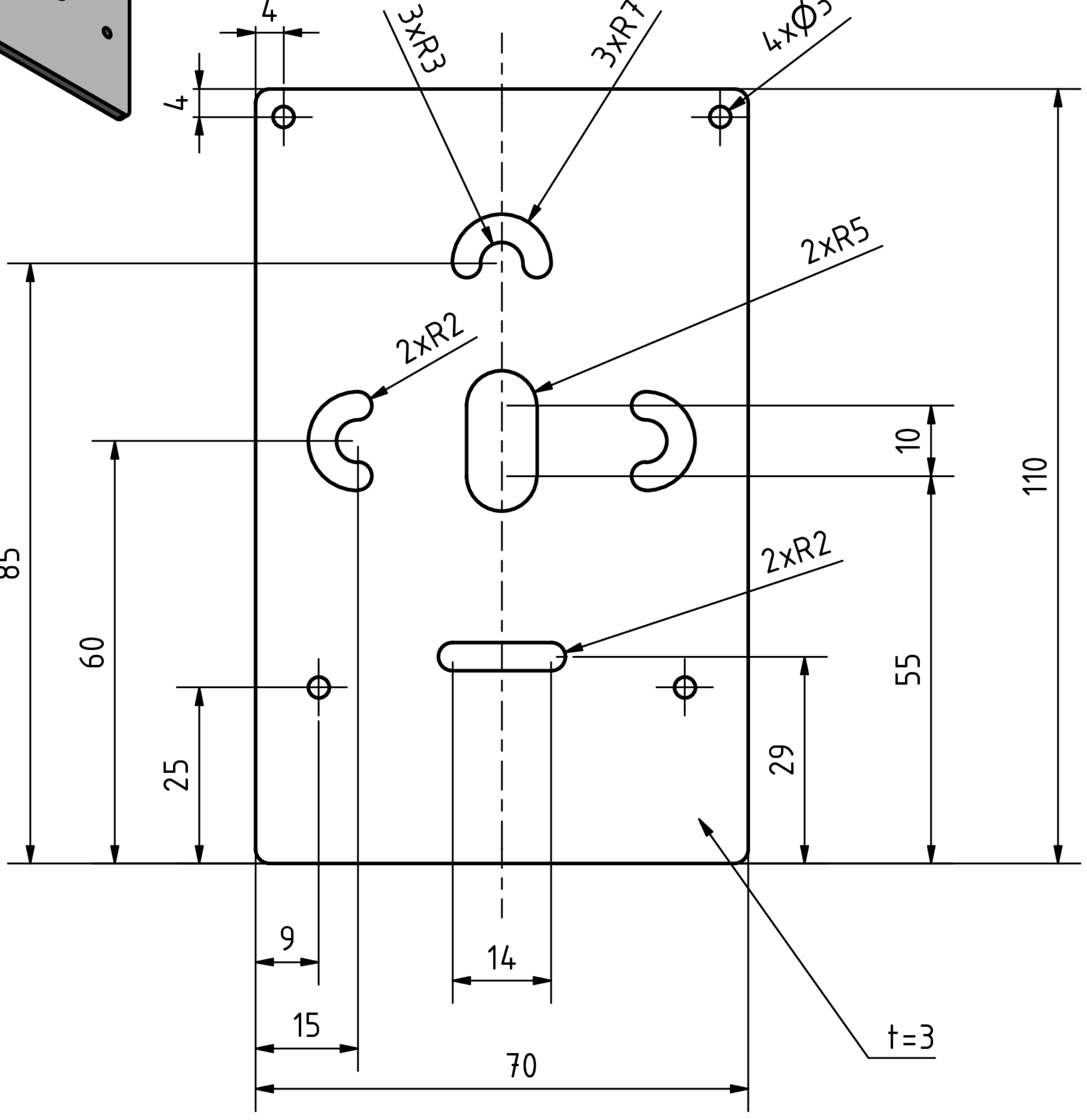




| | | | | | | |
|---|---|---|---|---|---|---|
| Materiál : ABS | | | | Přesnost ISO 2768-mK | Tolerování ISO 8015 | Promítání ISO E |
| | | | | TUL / FM | | |
| Měřítko | (1:1) | | Datum | Jméno | Front and Rear Wheel | |
| | | Nakreslen | 15.9.2022 | Z. Al-Dailami | | |
| | | Zkontrolován | | | | |
| | | Norma | | | | |
| | | | | | 01 | 1 |
| | | | | | | A4 |
| Stav | Změny | Datum | Jméno | | | |

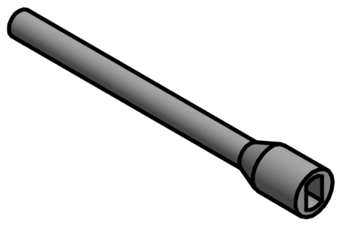
(1:2)

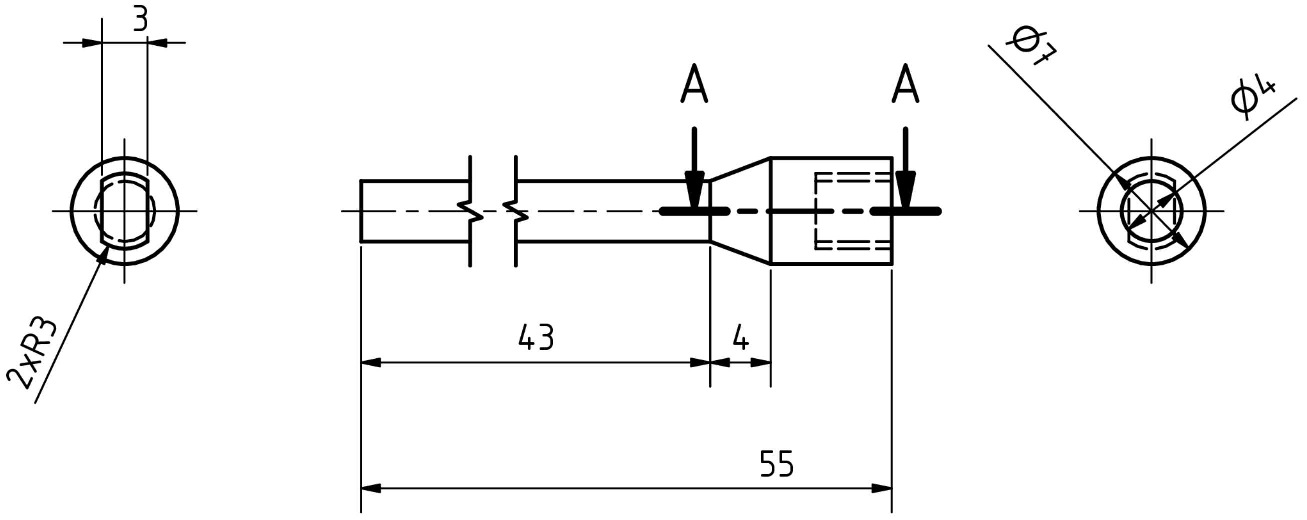




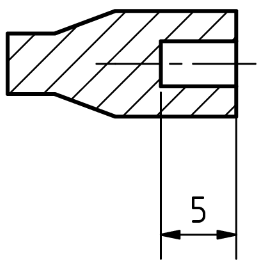
| | | | | | |
|---|---|---|---|---|---|
| Materiál : ABS | | | Přesnost ISO 2768-mK | Tolerování ISO 8015 | Promítání ISO E |
| | | | TUL / FM | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Měřítko | (1:1) | | | Datum | Jméno | |
| | | | Nakreslen | 15.9.2022 | Z. Al-Dailami | Upper Part |
| | | | Zkontrolován | | | |
| | | | Norma | | | |
| | | | | | | 02 |
| | | | | | | 1 |
| | | | | | | A4 |
| Stav | Změny | Datum | Jméno | | | |

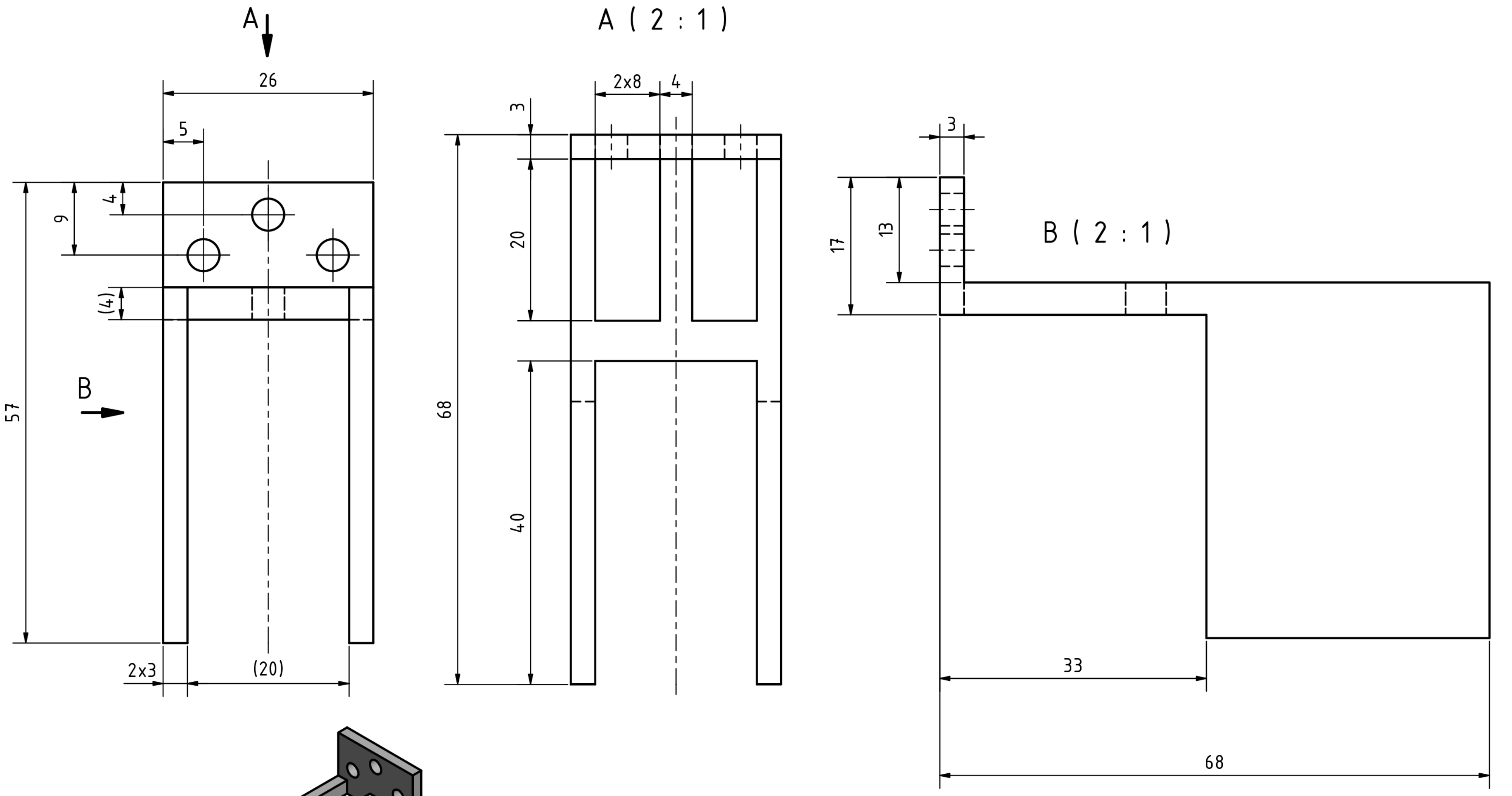


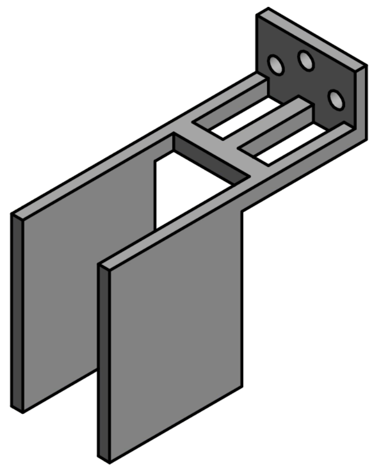


A-A ( 2 : 1 )



| | | | | | | |
|---|---|---|---|---|---|---|
| Materiál : ABS | | | | Přesnost ISO 2768-mK | Tolerování ISO 8015 | Promítání ISO E |
| | | | | TUL / FM | | |
| Měřítko | 2:1 | | Datum | Jméno | | |
| | | Nakreslen | 15.9.2022 | Z. Al-Dailami | Rear Wheel Shaft | |
| | | Zkontrolován | | | | |
| | | Norma | | | | |
| | | | | | 03 | 1 |
| | | | | | | A4 |
| Stav | Změny | Datum | Jméno | | | |





| | | | |
|---|---|---|---|
| Materiál : ABS | Přesnost ISO 2768-mK | Tolerování ISO 8015 | Promítání ISO E |
| | TUL / FM | | |

| Měřítko | (2 : 1) | | Datum | Jméno |
|---|---|---|---|---|
| | | Nakreslen | 15.9.2022 | Z. Al-Dailami |
| | | Zkontrolován | | |
| | | Norma | | |

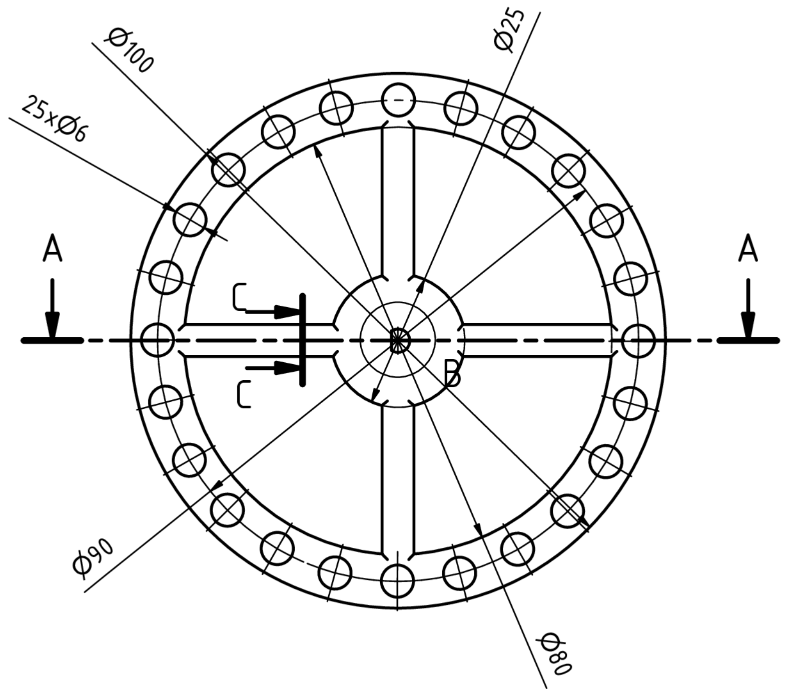
Steering Motor Support

04

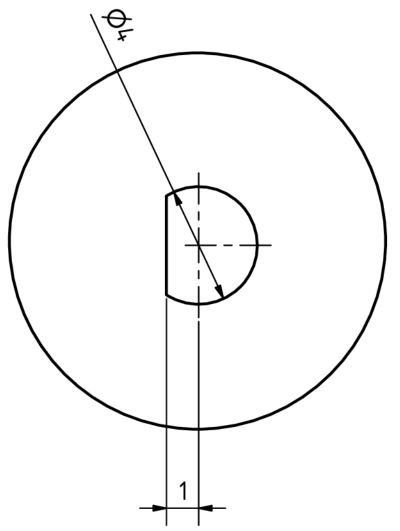
1

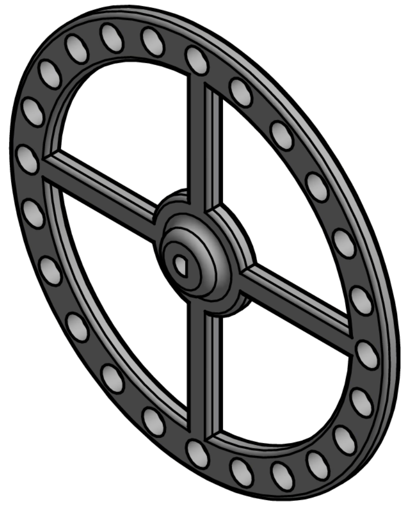
A3

| Stav | Změny | Datum | Jméno |
|---|---|---|---|

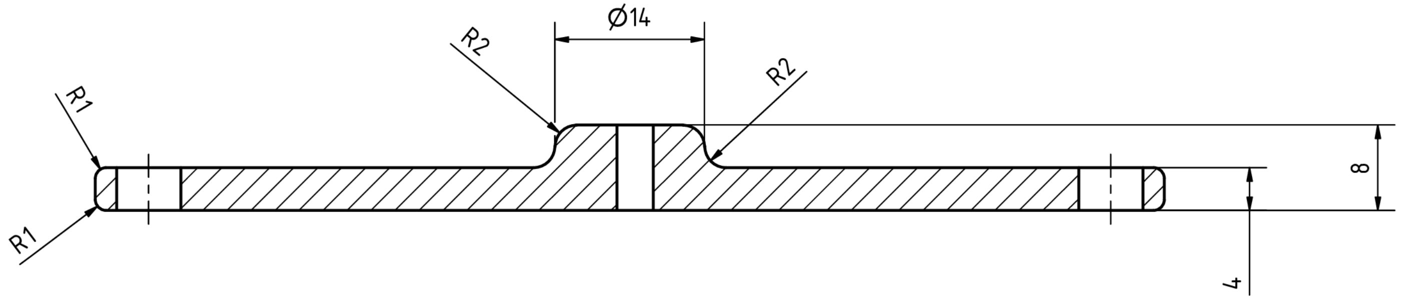


B ( 5 : 1 )

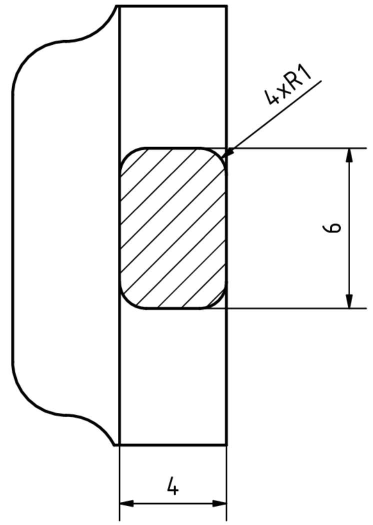




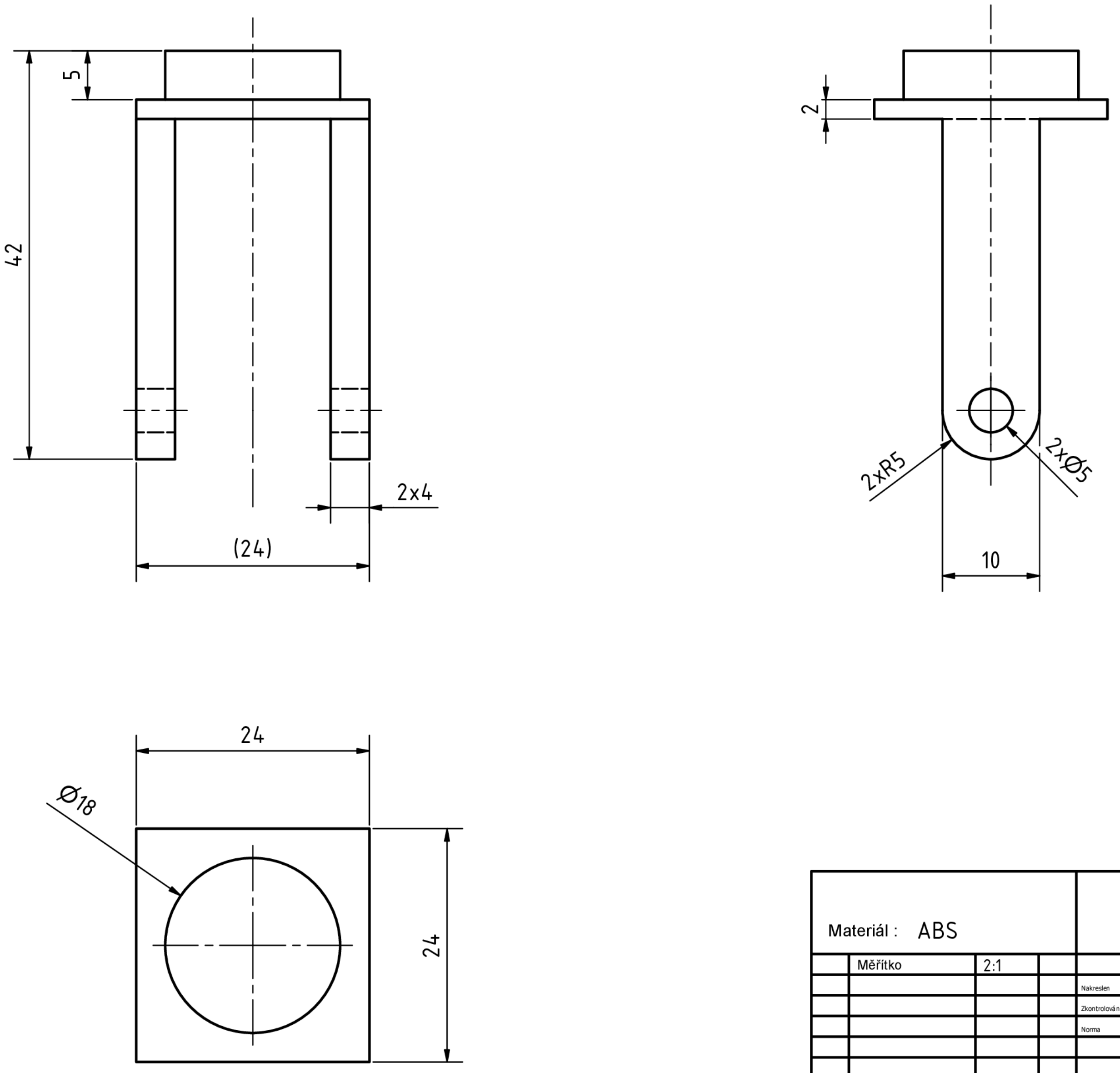
A-A ( 2 : 1 )

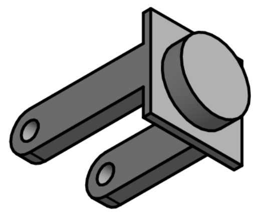


C-C ( 5 : 1 )

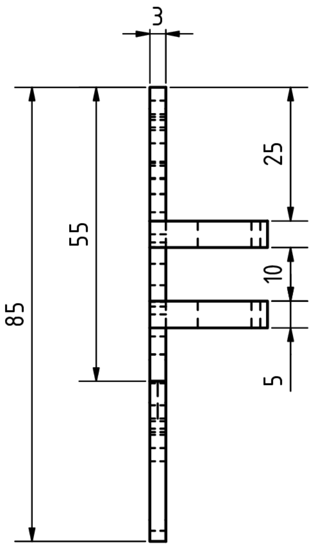


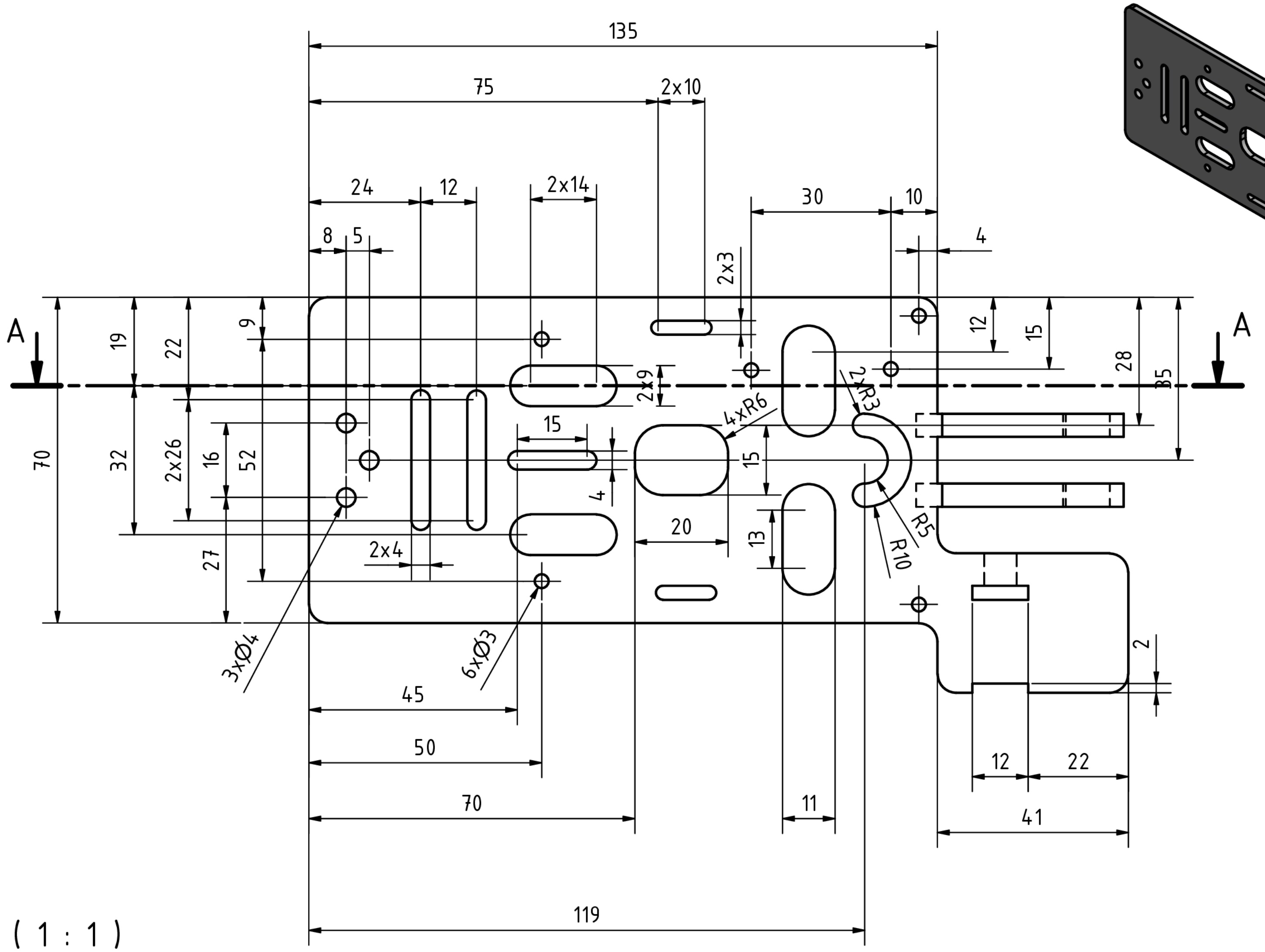
| | | | |
|---|---|---|---|
| Materiál : ABS | | | Přesnost ISO 2768-mK · Tolerování ISO 8015 · Promítání ISO E |
| | | | TUL / FM |
| Měřítko (1:1) | Datum | Jméno | Reaction Wheel |
| Nakreslen | 15.9.2022 | Z. Al-Dailami | |
| Zkontrolován | | | |
| Norma | | | |
| | | | 05 · 1 · A3 |
| Stav · Změny · Datum · Jméno | | | |

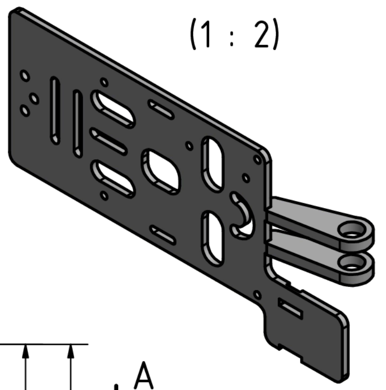




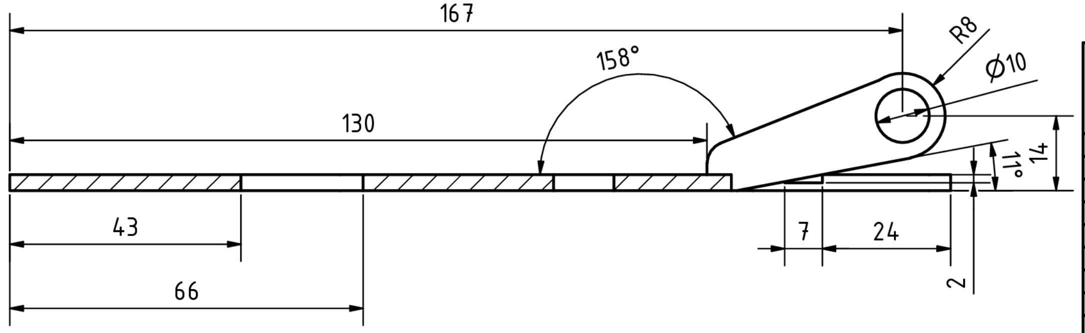
| | | | | |
|---|---|---|---|---|
| Materiál : ABS | | | Přesnost ISO 2768-mK | Tolerování ISO 8015 | Promítání ISO E |
| | | | TUL / FM | | |
| Měřítko | 2:1 | | Datum | Jméno | Front Wheel Support |
| | | Nakreslen | 15.9.2022 | Z. Al-Dailami | |
| | | Zkontrolován | | | |
| | | Norma | | | |
| | | | | | 06 | 1 |
| | | | | | | A3 |
| Stav | Změny | Datum | Jméno | | |





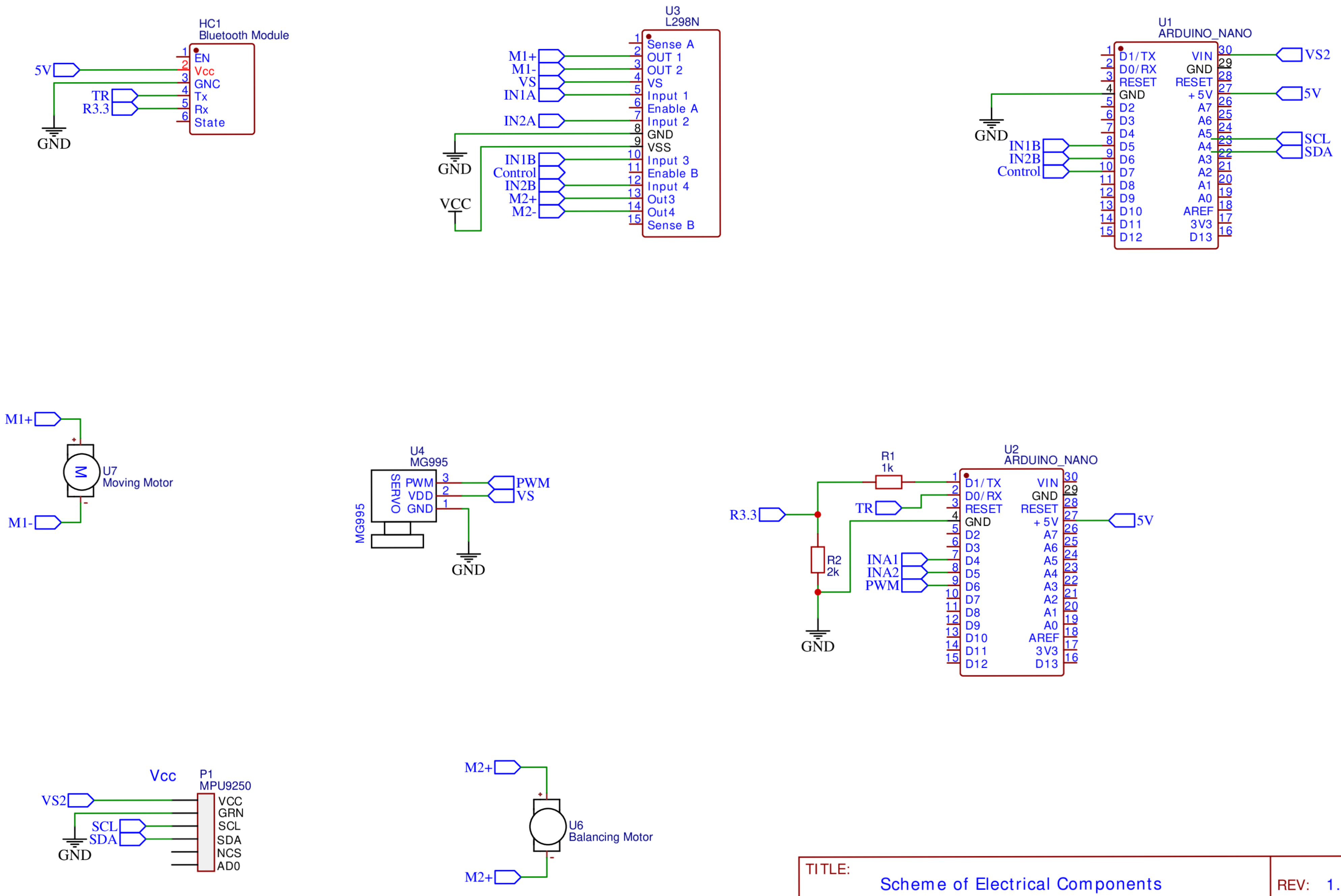


A-A ( 1 : 1 )



All Non-demensioned Radiuses R4

| | | | | |
|---|---|---|---|---|
| Materiál : ABS | | | Přesnost ISO 2768-mK | Tolerování ISO 8015 / Promítání ISO E |
| Měřítko 1:1 | | Datum | Jméno | TUL v Liberci |
| | Nakreslen | 15.9.2022 | Z.Al-Dailami | Lower Part |
| | Zkontrolován | | | |
| | Norma | | | |
| | | | | 07 / 1 / A3 |
| Stav / Změny / Datum / Jméno | | | | |



TITLE: Scheme of Electrical Components

REV: 1.0

Company: TUL FM

Sheet: 1/1

Date: 2022-09-24

Drawn By: Zaid Al-Dailami

## I. Calibrating IMU-9250 Magnetometer in MATLAB

```
clear all
clc
% Zaid Al-Dailami
% 22 8 2022
% Calibrating the Magnetometer

%%
a = arduino('COM3', 'MegaAdk', 'Libraries', 'I2C');
fs = 100; % Sample Rate in Hz
imu =
mpu9250(a,'SampleRate',fs,'OutputFormat','matrix');


%% Getting raw values so we can calibrate it
% move the sensro 360 in all direction for getting raw
data
tic;
stopTimer = 50;
i = 1
while(toc<stopTimer)
    [accel,gyro,mag] = read(imu);
    mag_x = mean(mag(:,1));
    mag_y = mean(mag(:,2));
    mag_z = mean(mag(:,3));
    mag_xx(i) = mag_x;
    mag_yy(i) = mag_y;
    mag_zz(i) = mag_z;
    i=i+1
end

mag_xyz_raw = [mag_xx', mag_yy',mag_zz']

% we then sace data:
%save raw_mag mag_xyz
% saving data as text file so it can be useed in
magnetometer program or
% directly we can use magcal function in matlab
%dlmwrite("raw_values.txt",mag_xyz)
```

```
%% Loading the data, and comparing raw values with
calibrated ones:

load raw_mag
mag_xyz_raw = mag_xyz;


[A,B,EXPMFS] = magcal(mag_xyz_raw);

Calibrated_Val = (mag_xyz_raw - B)*A;


% 2D plot:
% X,Y:
figure
plot(mag_xyz_raw(:,1),mag_xyz_raw(:,2),'b*');
hold on;
plot(Calibrated_Val(:,1),Calibrated_Val(:,2),'r*');
legend("Raw Values","Calibrated Values");
title("XY Magnetometer Data")
xlabel("{\it X [\muT]}")
ylabel("{\it Y [\muT]}")
grid on
xlim([-60 60])
ylim([-60,60])

% 3D plot:
figure
plot3(mag_xyz_raw(:,1),mag_xyz_raw(:,2),mag_xyz_raw(:,3
),"*b")
hold on
plot3(Calibrated_Val(:,1),Calibrated_Val(:,2),Calibrate
d_Val(:,3),"*r")
xlabel("{\it X [\muT]}")
ylabel("{\it Y [\muT]}")
zlabel("{\it Z [\muT]}")
legend("Raw Values","Calibrated Values");
title("XYZ Scatter Plot of Magnetometer Data")
```

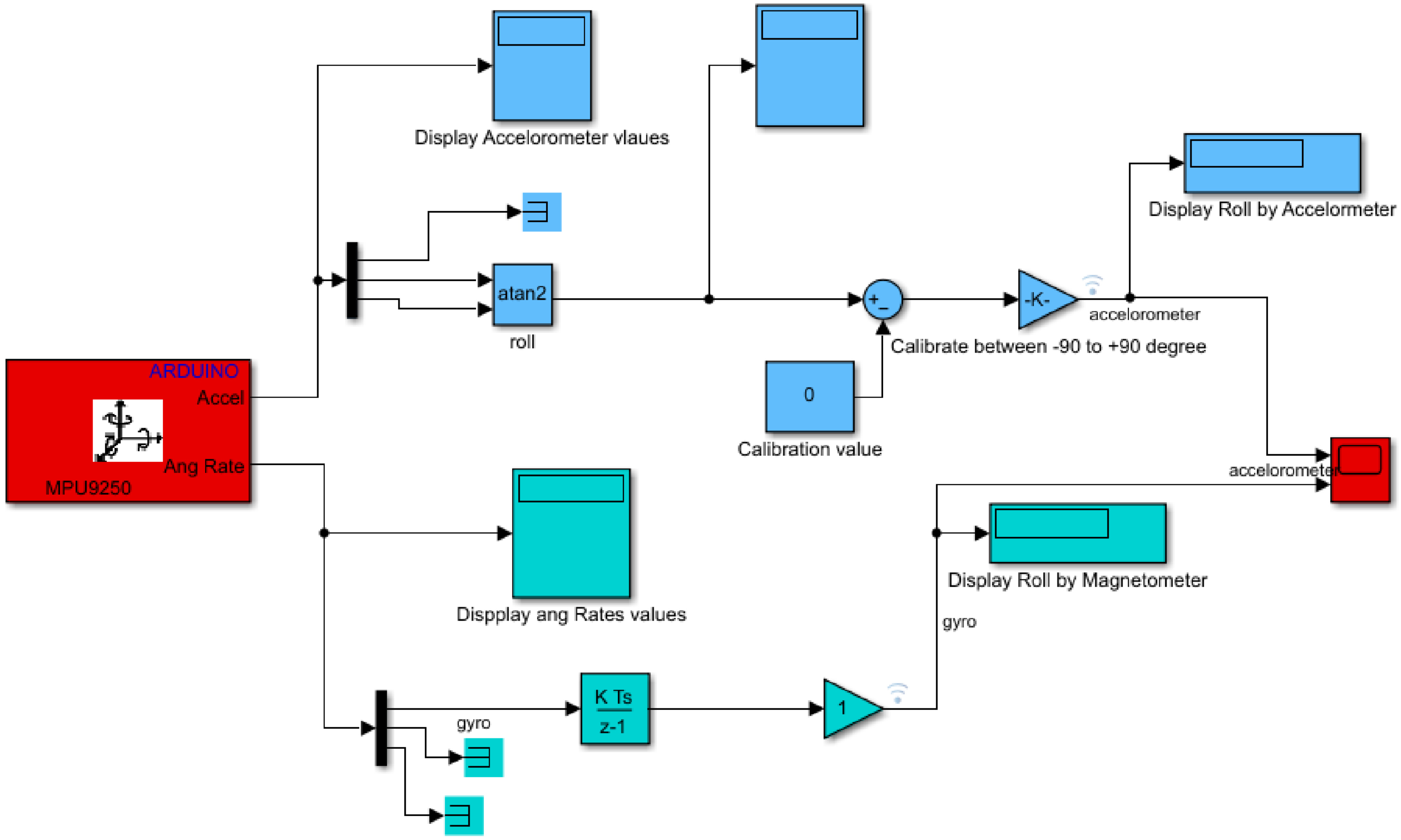
## II. Calibrating IMU-9250 Accelerometer in MATLAB

```
clear all
clc
% Zaid Al-Dailami
% 22 8 2022
% Calibrating the Accelorometer

%%
a = arduino('COM3', 'MegaAdk', 'Libraries', 'I2C');
fs = 100; % Sample Rate in Hz
imu =
mpu9250(a,'SampleRate',fs,'OutputFormat','matrix');


%% For getting Raw values one by one (value vy value
with changing position of IMU):
tic;
stopTimer = 1;
accelReadings=[];
while(toc<stopTimer)
    [accel,gyro,mag] = read(imu);
    accelReadings = [accelReadings;accel];
end
accel = mean(accelReadings)

%% B and M are values of matrices got from magneto
program after entering Raw values:
% B and M are the values of of matrices after
calibration
B = [-0.060484;0.100788;0.925356];
M = [1.010725,-0.001955, 0.000323;...
     -0.001955,0.998625,0.001641;...
     0.000323,0.001641,1.055832];

%% Comparing raw values and calibrated ones:
%load raw values
load raw_values

% Getting calibrated values
for i=1:length(raw_values)
    cal_values(:,i)= M*(raw_values(i,:)'-B);
end
cal_values = cal_values';

% 2D plots:
% X,Y:
plot(raw_values(:,1),raw_values(:,2),'b*');
hold on;
plot(cal_values(:,1),cal_values(:,2),'r*');
legend("Raw Values","Calibrated Values");
title("XY Accelerometer Data")
xlabel("X [m/s^2]")
ylabel("Y [m/s^2]")
grid on
% Y,Z:
figure
plot(raw_values(:,2),raw_values(:,3),'b*')
hold on
plot(cal_values(:,2),cal_values(:,3),'r*')
legend("Raw Values","Calibrated Values")
title("YZ Accelerometer Data")
xlabel("Y [m/s^2]")
ylabel("Z [m/s^2]")
grid on

% 3D plots:
figure
plot3(raw_values(:,1),raw_values(:,2),raw_values(:,3),'
b*');
hold on;
plot3(cal_values(:,1),cal_values(:,2),cal_values(:,3),'
r*');
title("3D Scatter Plot of Acclormeter Data")
legend("Raw Values","Calibrated Values");
xlabel("X [m/s^2]")
ylabel("Y [m/s^2]")
xlabel("X [m/s^2]")
zlabel("Z [m/s^2]")
grid on
```

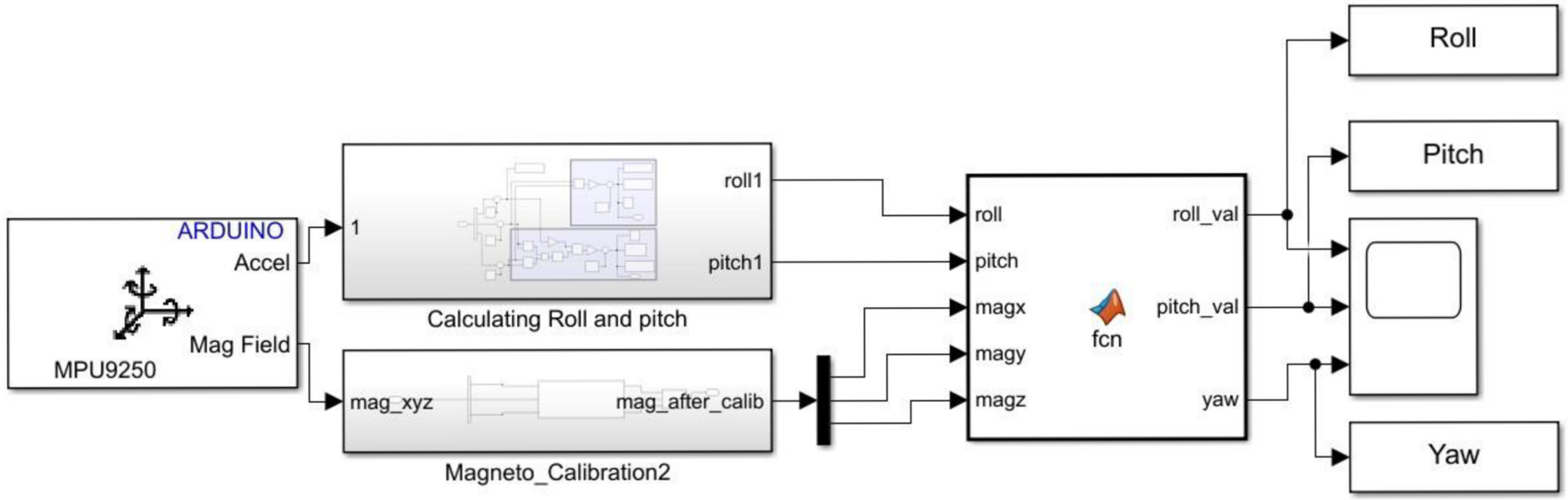
## III. Kalman Filter 3D in MATLAB Visualization and Data Plotting

```
clear all
clc
% Zaid Al-Dailami
% 22 8 2022
% IMU 9250 in Matlab


%% Setting the arduino:
% ardinosetup
a = arduino('COM3', 'MegaAdk', 'Libraries', 'I2C');
fs = 100; % Sample Rate in Hz
imu =
mpu9250(a,'SampleRate',fs,'OutputFormat','matrix');


%% Getting Data from IMU 9250, alligning orientaion of
Accelerometer and Gyroscpe with Magnetometer,
% Entering matrices that are used for calibration to
get more accurate results

% Values of B and M for calibrating accelerometer:
(Caluclated)
B = [-0.060484;0.100788;0.925356];
M = [1.010725,-0.001955, 0.000323;...
     -0.001955,0.998625,0.001641;...
     0.000323,0.001641,1.055832];

% Values of A and B for calibrating magnetometer:
(Calculated)
A = [1.0004,0.0157,0.0156;
    0.0157,1.0492,0.0909;
    0.0156,0.0909,0.9610];
b = [1.2579  -10.3320   -5.5533];

% GyroscopeNoise and AccelerometerNoise is determined
from datasheet and EX
GyroscopeNoiseMPU9250 = 0.01/1; % GyroscopeNoise
(variance value) in units of rad/s
AccelerometerNoiseMPU9250 = 0.00061; %
AccelerometerNoise(variance value)in units of m/s^2
viewer = HelperOrientationViewer('Title',{'AHRS
Filter'});
FUSE = ahrsfilter('SampleRate',imu.SampleRate,
'GyroscopeNoise',GyroscopeNoiseMPU9250,'AccelerometerNo
ise',AccelerometerNoiseMPU9250);
FUSE.LinearAccelerationNoise = 0.025;
FUSE.MagnetometerNoise       = 0.1;

stopTimer = 50;
i = 1;
tic;
while(toc < stopTimer)
    [accel,gyro,mag] = readSensorDataMPU9250(imu);
    % Swapping x and y for Accel
    accel_x = accel(:,2);
    accel_y = accel(:,1);
    accel_z = accel(:,3);
    % Chagning Polariy of x and y
    accel2 = [-accel_x,-accel_y,accel_z];
    % Calibrating Accelerometer
    accel3= M*(accel2'-B);
    accel3 =accel3';

    % Swapping x and y for Gyro
    gyro_x = gyro(:,2);
    gyro_y = gyro(:,1);
    gyro_z = gyro(:,3);
    gyro2 = [gyro_x,gyro_y,-gyro_z]*0.8;

    % Calibrating Magnetometer
    mag2 = (mag - b)*A;
    mag_x = mag2(:,1);
    mag_y = mag2(:,2);
    mag_z = mag2(:,3);
    mag2 = [mag_x,mag_y,mag_z]*0.02;

    rotators = FUSE(accel3,gyro2,mag2);
    %Convert quarternion into Euler angles
    eulFilt= euler(rotators,'ZYX','frame');
    eulFilt_Value(i,:) = mean(eulFilt);
    i = i+1;

    %3D Animation of the imu 9250 Sensor
    for j = numel(rotators)
        viewer(rotators(j));
    end

end


%%
t = linspace(0,stopTimer,length(eulFilt_Value(:,1)));

% Plotting Roll Angle:
figure
plot(t,(eulFilt_Value(:,3)*180/pi));
xlabel("{\it Time [s]}");
ylabel("{\it Roll [Deg]}");
title("Plot of Time with Respect to Roll Change")
grid on

% Plotting Pitch Angle
figure
plot(t,(eulFilt_Value(:,2)*180/pi)-6);
xlabel("{\it Time [s]}");
ylabel("{\it Pitch [deg]}");
title("Plot of Time with Respect to Pitch Change")
grid on


% Plotting Yaw Angle:
figure
plot(t,(eulFilt_Value(:,1)*180/pi)-116+38+4);
xlabel("{\it Time [s]}");
ylabel("{\it Yaw [deg]}");
title("Plot of Time with Respect to Yaw Change")
grid on


% Plotting Roll Pitch Yaw together:
figure
plot(t,(eulFilt_Value(:,3)*180/pi),'b');
hold on
plot(t,(eulFilt_Value(:,2)*180/pi)-6,'r');
hold on
plot(t,(eulFilt_Value(:,1)*180/pi)-116+38+4,'c');
grid on
legend("Roll","Pitch","Yaw")
xlabel('{\it Time [s]}');
ylabel('{\it Angle [deg]}');
title("Plot of Time with Respect to Angle Change")
```

## IV. Roll, pitch, and yaw by

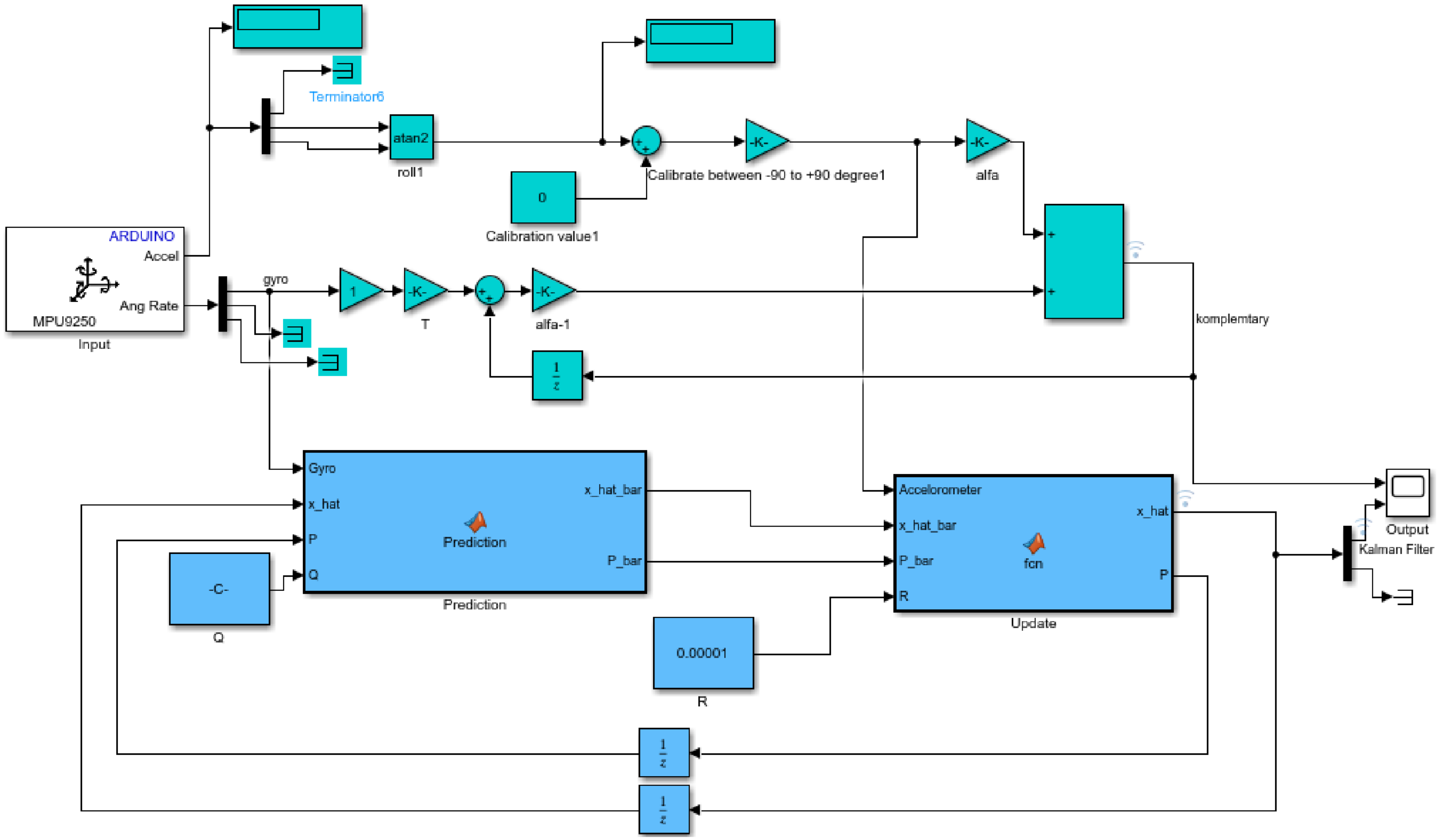

Display Accelorometer vlaues
atan2
roll
Calibrate between -90 to +90 degree
-K-
accelorometer
Display Roll by Accelormeter
ARDUINO
Accel
Ang Rate
MPU9250
0
Calibration value
accelorometer
Dispplay ang Rates values
Display Roll by Magnetometer
gyro
K Ts
z-1
gyro
1

## V. Roll, pitch, and yaw by Accelerometer and magnetometer



```
function [roll_val,pitch_val,yaw] =
fcn(roll,pitch,magx,magy,magz)

Xm = magx*cos(pitch) - magy*sin(roll)*sin(pitch) +
magz*cos(roll)*sin(pitch);
Ym = magy*cos(roll) + magz*sin(roll);
yaw = atan2(Ym,Xm) * 180 /pi;
roll_val = roll*180/pi;
pitch_val = pitch*180/pi;
```

## VI. Roll angle by Complementary Filter and Kalman Filter 2D



Cyan = Complementary filter

Light blue = Kalman Filter

## VI. Roll angle by Complementary Filter and Kalman Filter 2D

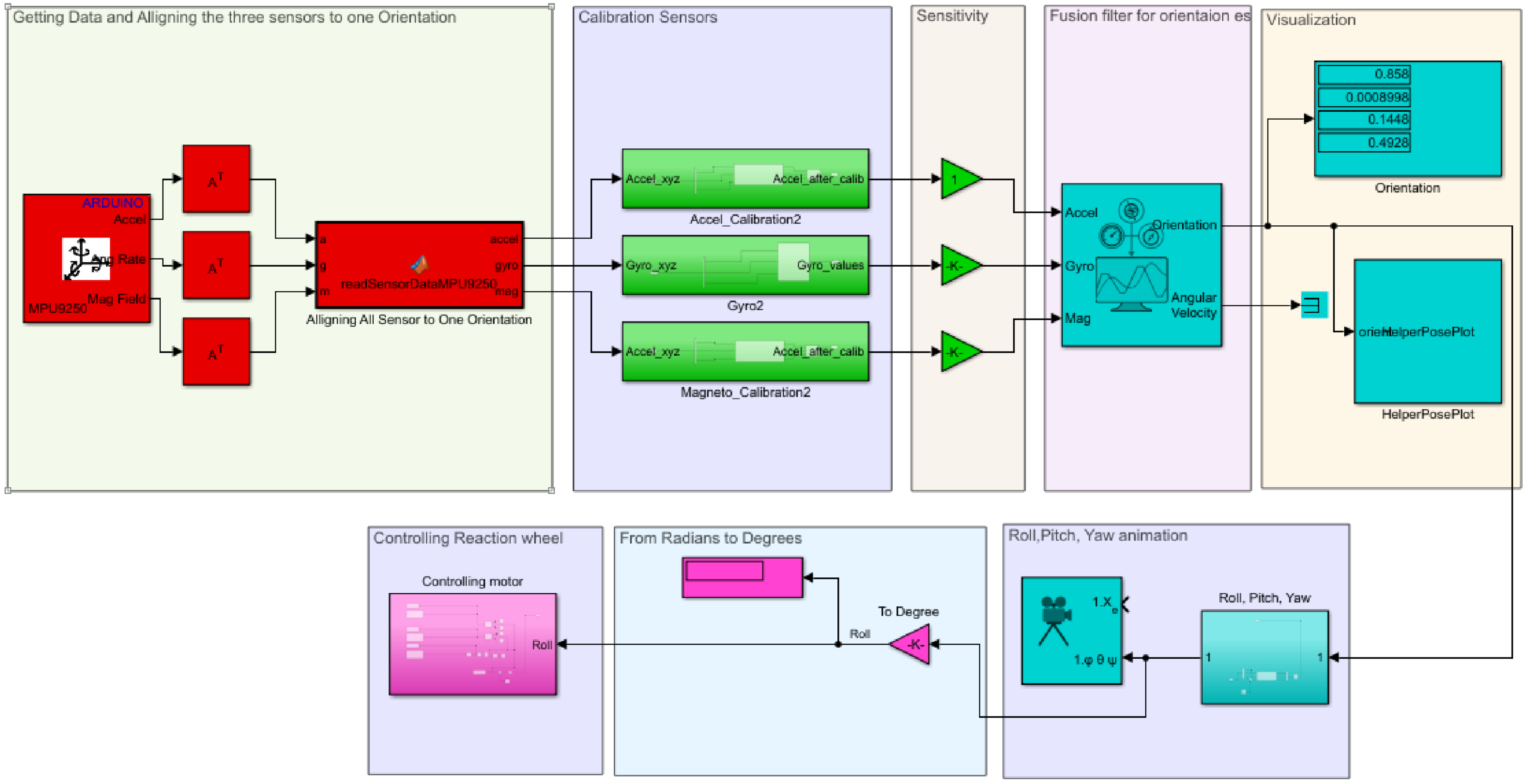
Prediction funtion:

```
function [x_hat_bar,P_bar] = Prediction(Gyro,x_hat,P,Q)

t = 0.01

A = [1 -t; 0 1]

B = [t; 0]

C = [1 0]

x_hat_bar = A*x_hat + B*Gyro

P_bar = A*P*A'+Q*t
```

Update function:

```
function [x_hat,P] = fcn(Accelorometer,x_hat_bar,P_bar,R)

C = [1 0]

K = P_bar*C'/ (C*P_bar*C'+R)

x_hat = x_hat_bar + K*(Accelorometer-C*x_hat_bar)

P = P_bar - K*C*P_bar
```

## VII. Kalman Filter 3D, orientation, visualization, and controlling motor



## VIII. Controlling steering and moving motor in Arduino

```
// Zaid Al-Dailami
// Controlling moving and steering motor
// 1.3.2022

int pos = 90;    // variable to store the servo position
char val;
const int m1a = 7;
const int m1b = 8;


#include <Servo.h>
Servo myservo;  // create servo object to control a servo

//const int pwm1 = 9;

void setup(){
  //pinMode(13,OUTPUT);
  pinMode(m1a, OUTPUT);
  pinMode(m1b, OUTPUT);
  myservo.attach(9);  // attaches the servo on pin 9 to the servo
object
  Serial.begin(9600);

}

void loop()

{ //digitalWrite(13,HIGH);
  while (Serial.available() > 0)
  {
  val = Serial.read();
  Serial.println(val);
  }
    //digitalWrite(13,HIGH);
```

```
if( val == 'B') // Backward
 {
   digitalWrite(m1a, LOW);
   digitalWrite(m1b, HIGH);
 }

else if(val == 'F') // Forward
 {
   digitalWrite(m1a, HIGH);
   digitalWrite(m1b, LOW);
 }

else if (val == 'R' && pos <=140) //Right
  { pos = pos + 1;
    myservo.write(pos);
    Serial.println(pos);
    delay(5);

  }

else if (val == 'L' && pos>= 40) //Left
  { pos = pos - 1;
   myservo.write(pos);
   Serial.println(pos);
   delay(5);
  }

  else if(val == 'I') //Forward Right
 {
   digitalWrite(m1a, HIGH);
   digitalWrite(m1b, LOW);
   if (pos <= 140)
   {
     pos = pos + 1;
```

## VIII. Controlling steering and moving motor in Arduino

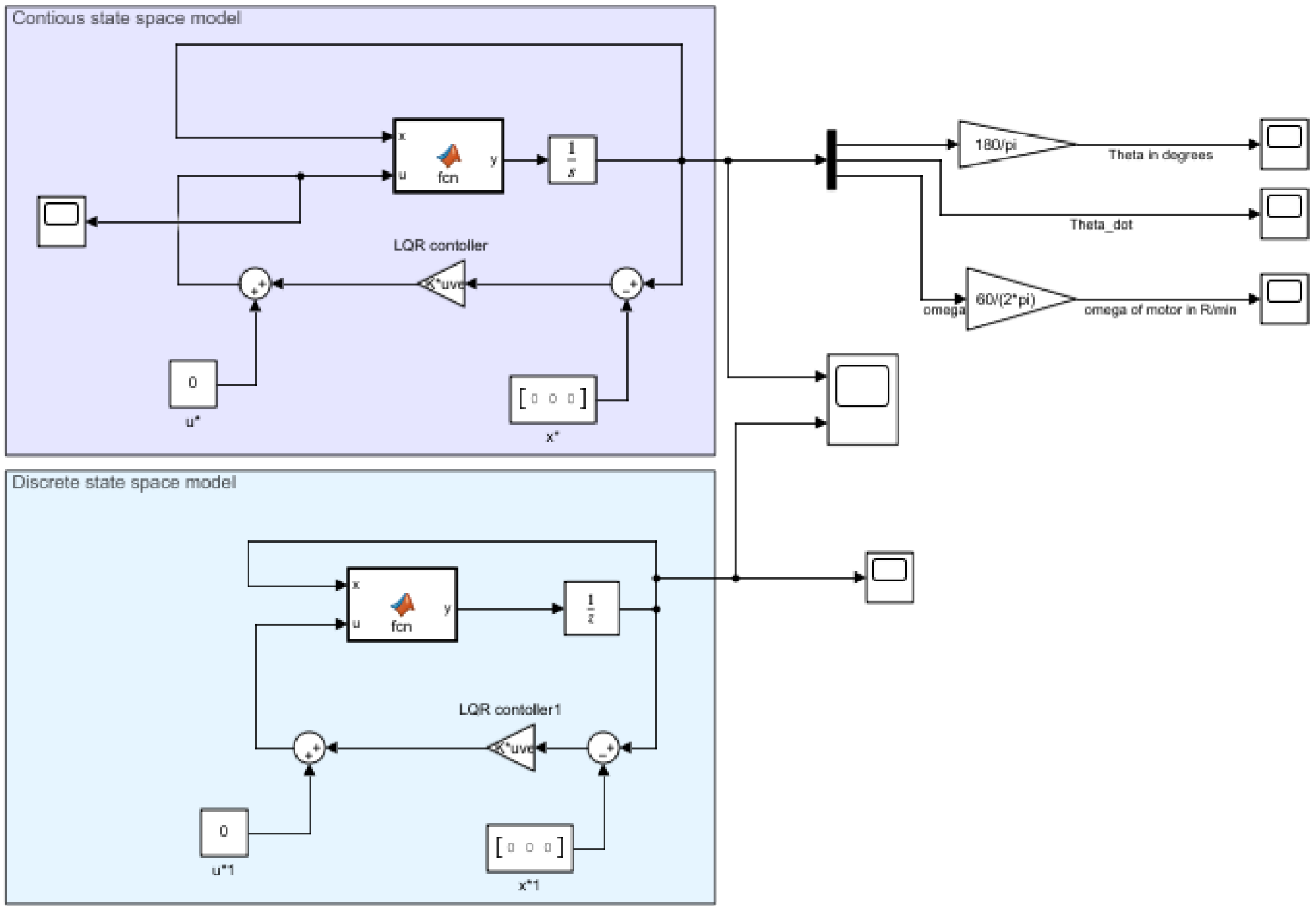
```
     myservo.write(pos);
     Serial.println(pos);
     delay(5);
   }
  }

  else if(val == 'J') //Backward Right
  {
  digitalWrite(m1a, LOW);
  digitalWrite(m1b, HIGH);
   if (pos <= 140)
   {
     pos = pos + 1;
     myservo.write(pos);
     Serial.println(pos);
     delay(5);
   }
  }
 else if(val == 'G') //Forward Left
  {
   digitalWrite(m1a, HIGH);
   digitalWrite(m1b, LOW);
    if (pos >= 40)
    {
     pos = pos - 1;
     myservo.write(pos);
     Serial.println(pos);
     delay(5);
    }
  }
 else if(val == 'H') //Backward Left
  {
  digitalWrite(m1a, LOW);
  digitalWrite(m1b, HIGH);
    if (pos >= 40)
    {
      pos = pos - 1;
      myservo.write(pos);
      Serial.println(pos);
      delay(5);
    }
  }


  else{
   digitalWrite(m1a, LOW);
   digitalWrite(m1b, LOW);
   delay(200);
 }
}
```

## IX. State-space model controlled (Continuous and discrete)

```
clear all
clc
%%

% Mechanical Parameters
L1 = 0.03; %m Length From Cart Center of Mass to Point
A
L2 = 0.04; %m Length From Wheel Center of Mass to Point
A
m1 = 0.5; %kg Mass of Cart
m2 = 0.1; %kg Mass of Reaction Wheel
m3 = 0.08; %kg Mass of Motor
I1 = 0.0001; %kg*m^2 Moment of Inertia of the Cart
I2 = 46.263 * 1e-4; %kg*m^2 Moment of Inertia of
Reaction Wheel
I3 = 0.004;  %kg*m^2 Moment of Inertia of Motor
g  = 9.81;   %accelration of gravity
I11 = m1 * L1^2;
I22 = m2 *(L1+L2)^2;
I33 = m3 *(L1+L2)^2;
MGL =  ((m1*L1)+(m2*(L1+L2))+(m3*(L1+L2)))*g;
It = I1 + m1*L1^2 + I2 + m2*(L1+L2)^2 + I3 +
m3*(L1+L2)^2 ;



%Electrical Parameters for the chosen DC motor:
% Free-run speed at 6 V: 1363 RPM
% Free-run current at 6 V: 80 mA
% Stall current at 6V: 900 mA
% Stall torque at 6V: 0.8 kg·cm
Free_Run_Speed = 1363*2*pi/60; %Rad/s
Nominal_Voltage = 6; %V
Stall_Current = 0.9; %A
Free_Run_Current = 80*1e-3; %A
Stall_Torque = 0.008; %kg.m
R = Nominal_Voltage/Stall_Current; %Inner Resistence of
motor
```

```
CM = (Nominal_Voltage - (R*Free_Run_Current)) /
Free_Run_Speed; %Konstant of the Motor(Electrical and
Mechanical)


a = MGL/It
b = CM^2/(R*It)
c = CM/(R*It)
d = -CM^2/(R*I2) - CM^2/(R*It)
e = CM/(R*I2) + CM/(R*It)

A = [0            1         0;
     a*cos(0)     0         b;
    -a*cos(0)     0         d]
B = [0; -c; e]
C = eye(3)
D = [0;0;0]


controllbility = ctrb(A,B);
if det(controllbility)==0
    fprintf("System is not contollable")
else
    fprintf("System is contollable")
end


%Checking stability:
stability = eig(A)

%LQR Feedback Controller:
Q = diag([10 1 1]);
RR = 1;
K = lqr(A,B,Q,RR)


x_star = [0 0 0]
u_Star = 0
```

## IX. State-space model controlled (Continuous and discrete)



Contious state space model
x
u
fcn
y
1/s
LQR contoller
K*uvec
0
u*
[0 0 0]
x*
Discrete state space model
x
u
fcn
y
1/z
LQR contoller1
K*uvec
0
u*1
[0 0 0]
x*1
180/pi
Theta in degrees
Theta_dot
omega
60/(2*pi)
omega of motor in R/min