National Institute of Applied Science and Technology

C.A.O

Project report

Elaboré par : MSEIBI souha

JABRI chaima

CHAABANI elaa

JLASSI wejden

Self-Balancing Robot

Publically defended on :27/04/2022, jury: ***Mrs. Olfa Boubaker***

Academic year 2021/2022

Table of contents

Table of figures

Table of Tables

Introduction

For the last few decades, one of the most popular benchmarks for research in control theory and robotics enthusiasts around the world has been "the wheeled inverted pendulum".

In fact, the instability of mobile inverted pendulum has surfaced in recent years and has attracted interest from control theory experimentation. Furthermore, engineers have applied the idea of the mobile inverted pendulum model to many problems such as designing walking gaits for humanoid robots, robotic wheelchairs and personal transport systems (Segway)

In view of this facts, developing a self-balancing robot is the ideal platform to put into practice what has been covered in control Systems lectures. It would also be so interesting to investigate the suitability and the performance of the PID regulator and to compare between the behavior in practice and the simulations.

It is in this perspective that our CAD project fits.

This report is divided into six parts. The first point concerns the difficulties of control of the system, then we will move to the mathematical model of the robot. Next, we will have a closer look at the electrical design. then, we will present the mechanical design. After that we will consider the software implementation and we will highlight the experimental results and finally we will end with conclusion and prospect.

I-Control difficulties

Self-balancing robots are having the inverted pendulum structure mounted over two wheels and they are characterized by the ability to balance itselves.[1]

It is an inherently unstable, high-order, multivariable, nonlinear, and strongly coupled system, and represents an underactuated mechanical system. For such an underactuated mechanical system, which has fewer control inputs than the generalized coordinates, it is necessary to indirectly control the underactuated generalized coordinates through dynamic coupling. Underactuation, while resulting in a smaller number of actuators and thus helping to reduce the manufacturing costs and failure rate, poses challenges to control design. Furthermore, unlike simpler systems like the pendulum on a cart system, which are restrained to a guided trajectory, the self-balancing robot moves on its own trajectory while balancing the pendulum. One of the difficulties of controlling the robot is to simultaneously control its linear motion, tilt motion, and yaw motion. [3]

II- Mathematical model:

The principle of position control is based on the famous inverted pendulum benchmark. Tilting the platform by an angle alpha, will cause the robot to move, until that the angle becomes zero again. The system can therefore be modeled as a mechanical subsystem which consists of the robot body and two wheels. The body can be modeled as an inverted pendulum with the mass concentrated at the center of gravity and the axis of rotation above the wheel axis.

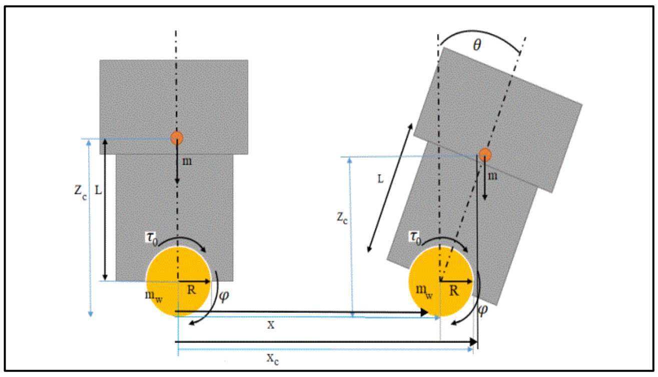


Figure 1:3D robot model

In order to develop a reliable and efficient control system for a two-wheeled balancing robot, it is essential to understand its parameters. This can be achieved through a mathematical model. The following table represents the set of robot physical parameters that allow us to model our system:

Table 1:Table of assignements

X	Horizontal position of the center of the Wheel compared to a defined origin
X_C	Horizontal position of the point m of the body part compared to the origin
φ	Angle of rotation of the wheel relative to the horizontal axis at t = 0
Z_c	Vertical position of the body
θ	Angle of rotation of the body part to vertical position
M	Body mass
m_W	Wheel mass
R	Wheel radius
L	Length between the body and the center of the wheel
I	The inertia of the body part
Iw	Wheel inertia

Dynamic Equation:

$$\begin{array}{l|l} x = R\,\varphi & \dot{x} = R\,\dot{\varphi} \\ xc = R\,\varphi + L \sin\theta & \dot{x}c = R\dot{\varphi} + L\dot{\theta}\cos\theta \\ zc = R + L\cos\theta & \dot{z}c = -L\,\dot{\theta}\sin\theta \end{array} \qquad (1)$$

Potential energy E_p:

$$Ep = mg\,(R + L\cos\theta) - mg(R+L) = mgL\,(\cos\theta - 1) \qquad (2.1)$$

The kinetic energy Ec :

(2.2)

$$Ec = \frac{1}{2} mw\,\dot{x}^2 + \frac{1}{2}\,Iw\,\dot{\varphi}^2 + \frac{1}{2} m\,\dot{x}c^2 + \frac{1}{2} m\,\dot{z}c^2 + \frac{1}{2} I\,\dot{\theta}^2$$
$$= \frac{1}{2}(Iw + mw\,R^2 + mR^2)\,\dot{\varphi}^2 + m\,R\,L\cos\theta\dot{\varphi}\,\dot{\theta} + \frac{1}{2}(I + m\,L^2)\,\dot{\theta}^2$$

Lagrangian derivation

$$\boxed{L = Ec - Ep = \frac{1}{2}(Iw + mw\,R^2 + mR^2)\,\dot{\varphi}^2 + m\,R\,L\cos\theta\dot{\varphi}\,\dot{\theta} + \frac{1}{2}(I + m\,L^2)\,\dot{\theta}^2 - mgL\,(\cos\theta - 1)} \qquad (2.3)$$

Par conséquent, les équations dynamiques pour la coordonnée **φ** et la coordonnée θ peuvent être dérivées comme suit :

For θ:

$$\frac{\partial L}{\partial \dot{\theta}} = m\,R\,L\cos\theta\,\dot{\varphi} + (1 + mL^2)\,\dot{\theta} \qquad (3.1)$$

$$\frac{\partial L}{\partial \theta} = -m\,R\,L\,\dot{\varphi}\sin\theta\ddot{\theta} + mgL\sin\theta \qquad (3.2)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = (I + m\,L^2)\,\ddot{\theta} + m\,R\,L\cos\theta\ddot{\varphi} - m\,g\,L\sin\theta = \chi \qquad (3.3)$$

For φ:

$$\frac{\partial L}{\partial \dot{\varphi}} = (Iw + mw\,R^2 + mR^2)\,\dot{\varphi} + mRL\,\dot{\theta}\cos\theta \qquad (4.1)$$

$$\frac{\partial L}{\partial \phi} = 0 \qquad (4.2)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}}\right) - \frac{\partial L}{\partial \phi} = (Iw + mw\,R^2 + mR^2)\,\ddot{\varphi} + m\,R\,L\cos\theta\ddot{\theta} - m\,R\,L\sin\theta\dot{\theta}^2 = \mu \qquad (4.3)$$

⇒ μ and χ are the torques for each coordinate. We can write these nonlinear dynamic equations in a second-order matrix style as follows:

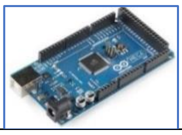
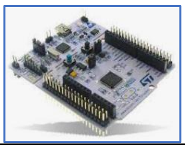
$$\boxed{\begin{pmatrix} \mathrm{Iw}+\mathrm{mw}\,R^2+\mathrm{m}R^2 & \mathrm{m\,R\,L}\cos\theta \\ \mathrm{m\,R\,L}\cos\theta & \mathrm{I}+\mathrm{m}\,L^2 \end{pmatrix}\begin{pmatrix} \ddot{\varphi} \\ \ddot{\theta} \end{pmatrix}+\begin{pmatrix} 0 & -\mathrm{m\,R\,L}\sin\theta\dot{\theta} \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \end{pmatrix}+\begin{pmatrix} 0 \\ -\mathrm{m\,g\,L}\sin\theta \end{pmatrix}=\begin{pmatrix} \mu \\ \chi \end{pmatrix}} \quad (5)$$

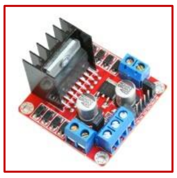
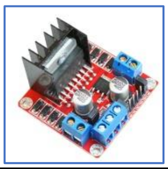
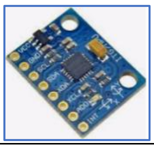
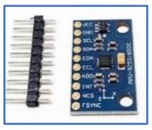
III- Electrical design:

In this part, we will focus on electronic components and devices necessary for the realization and design of our project as well as the wiring diagram.

III-1 Hardware choice:

Table 2:Hardware choice

Hardware	Possible solutions	Characteristics	Chosen solution
Microcontroller	**Arduino Uno**	• Less expensive • Has a lot of modules to simplify the work • Available	we already have an Arduino uno board and in order to minimize the cost of the project we opted for this solution
	Arduino Mega 	• A large number of I/O • Low computing power • Clutter	
	STM 32 	• More Professional • More powerful • It can work in real time with an embedded OS STM32 F4	
Motor	**DC Motor 6V**	• Low cost • Small size and easy installation • Complexity of implementing (Example: PID control)	With a view to optimize the energy consumption and minimize the cost we have chosen the DC 6v motor
	DC Motor with encoder	• Precise speed control • More precision • Return stability after an external perturbation • High cost	
	Stepper Motor	• More precise • Heavier, • Consumes more energy	

Motor Driver	**4-channel relay module**	• Switching current maximum: 10A • Cut-off voltage 250 V/AC, 30 V/DC max.	Since the motors used are low power, the L298N is preferred 
	L298N Module 	• Low cost • It can deliver up to 2A peak and 20W continuous.	
Sensor	**MPU 6050** 	• 3-axis gyroscope and a 3-axis accelerometer • Low cost	since we don't need a magnetometer, we chose the MPU6050
	MPU 9250 	• 9-axis which combines 3-axis gyroscope, 3-axis accelerometer and 3-axis magnetometer	

III-2 Electrical design:

Using the Fritzing software, to read the inclination of the robot we chose an MPU6050 sensor on which we carried out stationary tests where the sensor rests flat on a surface which has an inclination of about 0° and after which the calibration of the sensor is carried out.

We can see on the synoptic diagram produced on the easyeda software **[2]** in the **appendix I**, that the sensor uses the I2C protocol which involves the use of serial clock pin (SCL) which synchronizes the transfer of data and a serial data pin (SDA) over which the data is sent. The sensor is powered in our case at 5V.

To control the speed and direction of the motor using the driver which involves the use of PWM signal pin (5,6,9,10) which provides Pulse Width Modulation, this driver is powered at 12v. The electrical design is illustrated by the figure bellow:

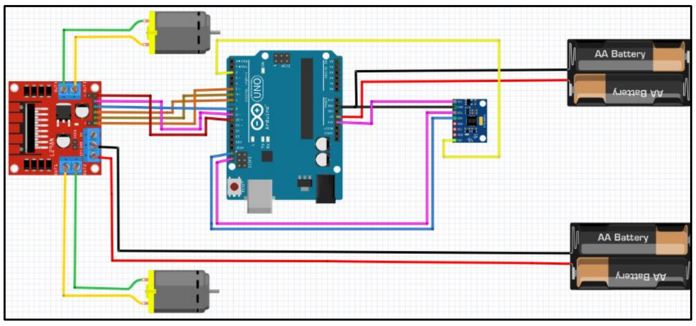


Figure 2:Electrical design

IV- Mechanical Design:

The design goal was to make the CAD model as similar as possible. To achieve this goal, we used CATIA software.

To ensure a stable base for the robot, we will need 2 light wooden bases, a lower base designed for the purpose of fixing the motor supports and also contains the driver and a 5V power supply, another higher base for aligning the axis of the gyroscope with the motor shaft and also contains our microcontroller, regulator and main power supply 12V.

Our chassis also requires nuts and 4 galvanized steel threaded rods,

And as steel is a metal alloy made mainly of iron and carbon, so it is a rust and corrosion resistant material and more durable but good conductor of electricity for this we add a layer of zinc which is called the operation of galvanizing.

The picture bellow shows the mechanical conception of our robot.

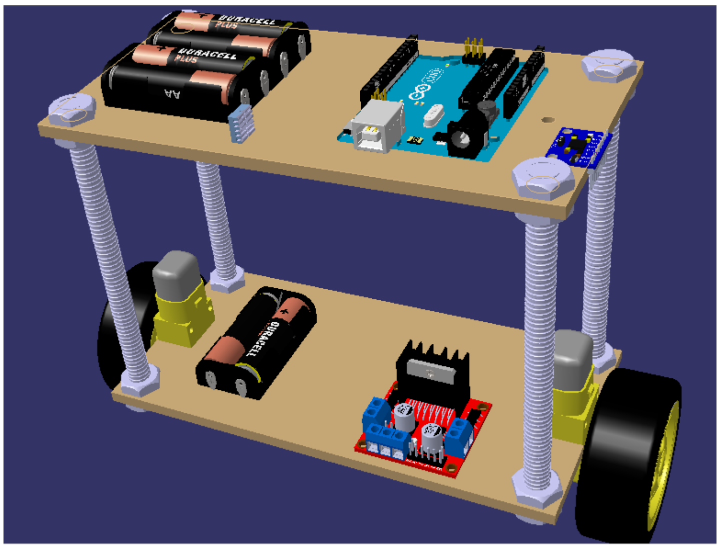


Figure 3:Mechanical design

The exploded view of our mechanical design is given in **Appendix II**

V- Software design:

The project is divided into two necessary parts, the HARDWARE part in which we discussed the material used, the mechanical design, and a soft part in which we will generate the code necessary for the robot as well as the simulations.

In the programming part we will therefore approach at the beginning the MATLAB simulation to determine the PID parameters and simulate the operation of the robot

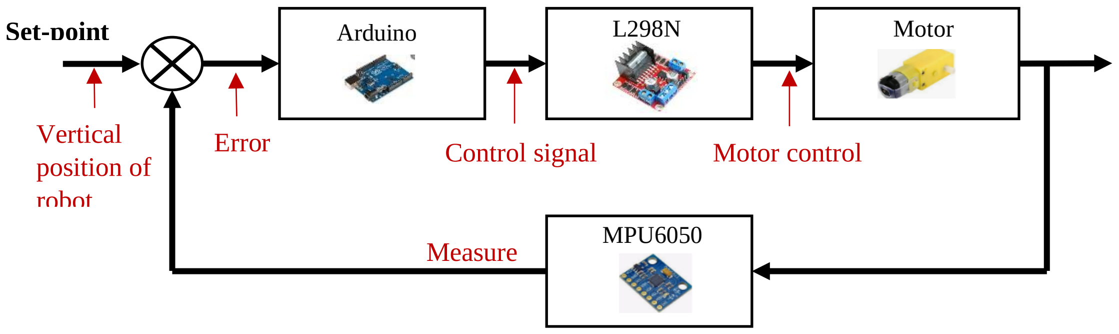
Thereafter we will present the flowchart of the Arduino code and finally we will present the LABVIEW interface

V.1 - MATLAB Simulation

V.1.1- PID regulator block Diagram:

In this loop, we find the mpu6050 sensor to measure the output which is the inclination of the robot's body

The Arduino Uno card is used as a comparator to adjust the error between the setpoint and the values received by the output measurement, as well as a corrector which controls the motors according to the error signal.



We have implemented the PID regulator to control the variables of our process and turn it into a more precise and stable status.

→ The command is both proportional to the error, proportional to the sum of the errors, and proportional to the variation of the error.

command = Kp * error + Ki * sum errors + Kd * (error - previous error)

Kp is the proportionality coefficient of the error.

Ki is the proportionality coefficient of the sum of the errors.

Kd is the proportionality coefficient of the error variation.

V.1.2- Building a self-balancing Robot with Simulink

In order to simulate the operation of our system, we have illustrated a 3d model of a two-wheeled self-balancing robot controlled using the PID controller and tuned manually.

- **Blocks' explanation:**

The solver: responsible for performing calculations

The world frame: considered to be the ground of every mechanical model in Simulink, and it gives access to the world coordinate frame.

The mechanism configuration: specify a uniform gravity for the entire mechanism

Chassis: represents the rods and the plates of the robot

Cart: represents the wheels of the robot

Rigid transform block: transform one solid block with respect to another solid block, and their pose with respect to each other becomes constrained, so they can move as one unit.

Revolute joint: will give a degree of freedom to the motion and this will allow the chassis to swing like a pendulum.

Prismatic joint: situated between the cart and the world frame which will allow it to move back and forth and eventually stabilizing the body in an upright pose given by the appropriate controller.

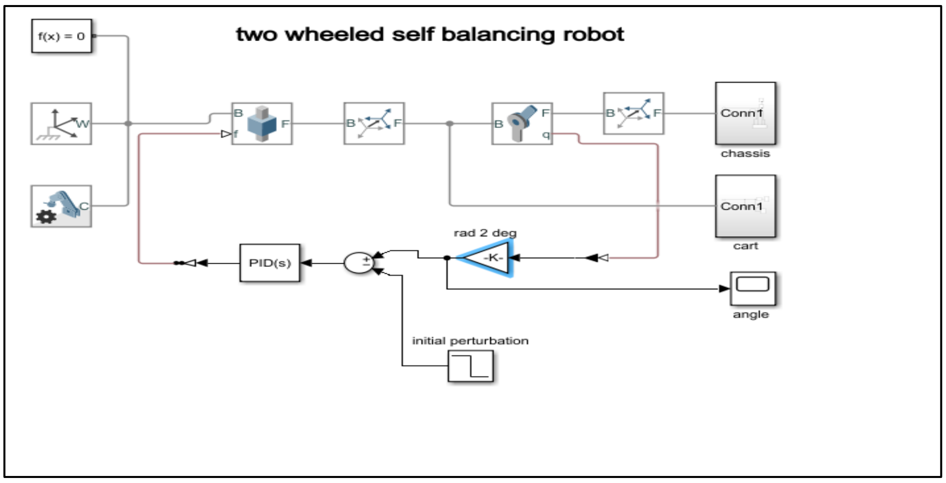


Figure 4:self-balancing Robot modulization with Simulink

In order to achieve the desired behavior and calculate the right input, we need the control loop. This control constitutes mainly from three parts, the first one is our system, and from it, we'll measure the inclination angle theta, the second part is the summation process, where we are going to compare theta with the desired output which is 180° and we feed the error to the third part of this loop which is the PID controller, which will give us the appropriate force to be applied in order to stabilize the robot.

We began by reading the angle inclination of the chassis provided by the position of the revolute joint. We also set the force in the prismatic joint to be provided by input, and the motion to be automatically computed.

Now, the PID parameters must be tuned to give us the desired behavior.

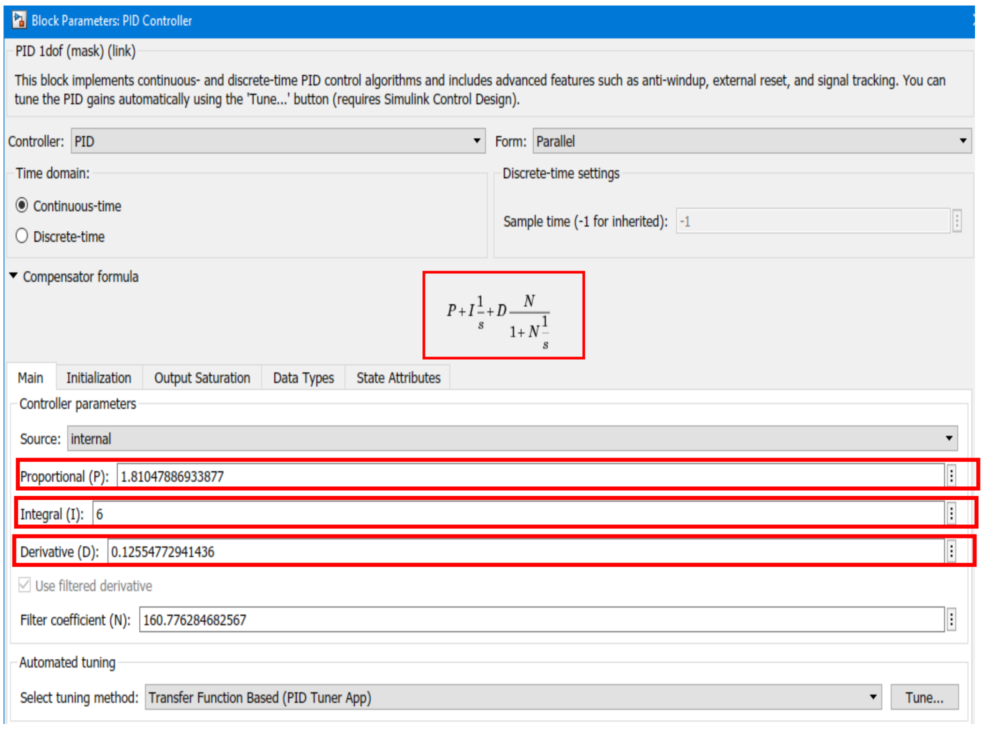
The main parameters to consider are the proportional, the integral and the derivative gains.

First, we set the proportional gain to 2, while the others to 0. Our system became stabilized. However, the oscillations continue to increase and the robot will fall eventually. We can reduce the oscillation by increasing the derivative gain. We set it for 0.01. We can see that the oscillations are reduced and the system is stable.

To manage the steady state error more efficiently we increase the integral gain.

However, the integral gain and the proportional gain make the system faster, as a consequence increase the oscillations. That is why we need to increase the integral gain to keep the robot in place.

After some testing and tuning, we have settled on the corresponding values for our gains to obtain the required behavior:


Block Parameters: PID Controller

PID 1dof (mask) (link)

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: PID
Form: Parallel

Time domain:
● Continuous-time
○ Discrete-time

Discrete-time settings
Sample time (-1 for inherited): -1

Compensator formula

$$P+I\frac{1}{s}+D\frac{N}{1+N\frac{1}{s}}$$

Main | Initialization | Output Saturation | Data Types | State Attributes

Controller parameters
Source: internal
Proportional (P): 1.81047886933877
Integral (I): 6
Derivative (D): 0.12554772941436
☑ Use filtered derivative
Filter coefficient (N): 160.776284682567

Automated tuning
Select tuning method: Transfer Function Based (PID Tuner App)
Tune...

Figure 5:PID parameters

V.2 – Software code Implementation:

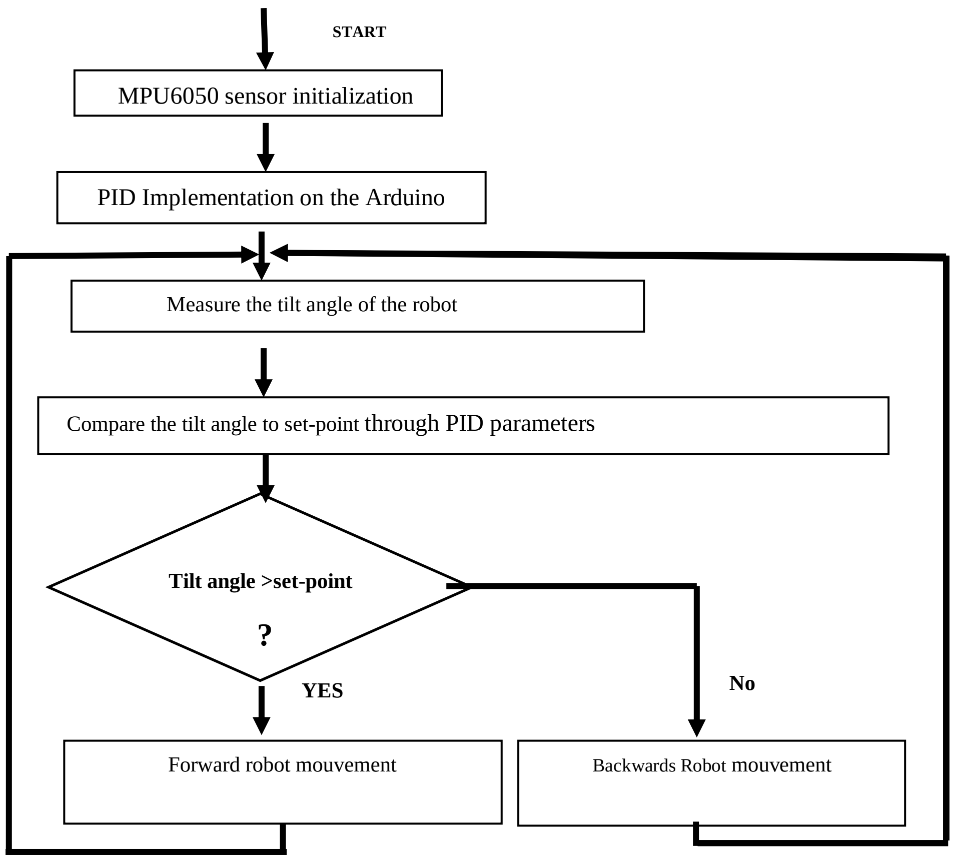
- The implementation of the software code was an important part for our project. Indeed, we spent a considerable amount of time on this component of the project.

  The software was coded using Arduino IDE. Before starting the coding, it was essential to calibrate the sensor. In fact, it is necessary to determine the offset values in order to set the reference values for our robot.

  To do this, we have implemented an algorithm to determine these values by calculating the average of 1000 values read by the sensor MPU6050.

- The most important part of the code is the control loop. It is an infinite loop that reads the data, processes it, and then adjusts motor voltages.

  The structure of the Arduino program is shown by the Following flowchart:


START
MPU6050 sensor initialization
PID Implementation on the Arduino
Measure the tilt angle of the robot
Compare the tilt angle to set-point through PID parameters
Tilt angle >set-point
?
YES
No
Forward robot mouvement
Backwards Robot mouvement

IV.3 – LabVIEW interface:

The development of the LabVIEW code is divided into two parts: the front panel and the block diagram.

➢ The front panel:

Thus, in this interface we have the values read by the gyroscope and the accelerometer of the 3 axes and the three evolution curves of the position and acceleration values.

Similarly, we have given the possibility to the user to choose a single axis at a time in order to view the values read.

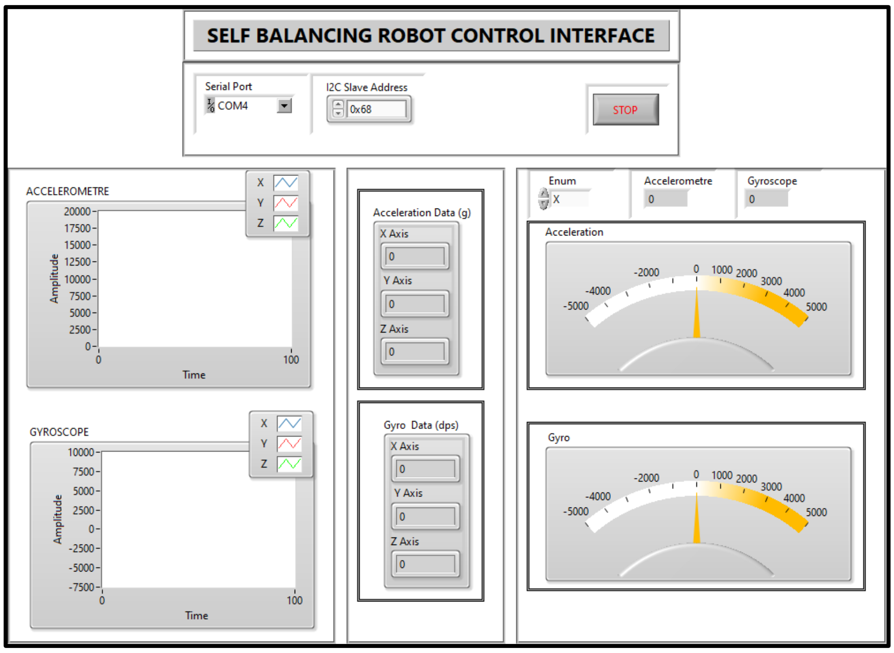


Figure 6:LabVIEW interface front panel

➢ The block diagram:

The block diagram presents the programming part of the LabVIEW code. The picture below shows the structure of the code

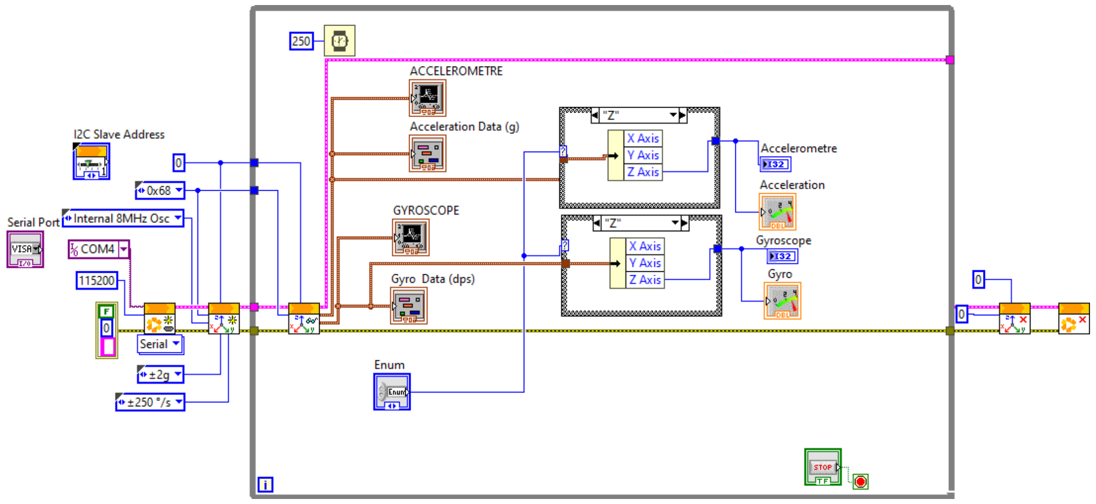
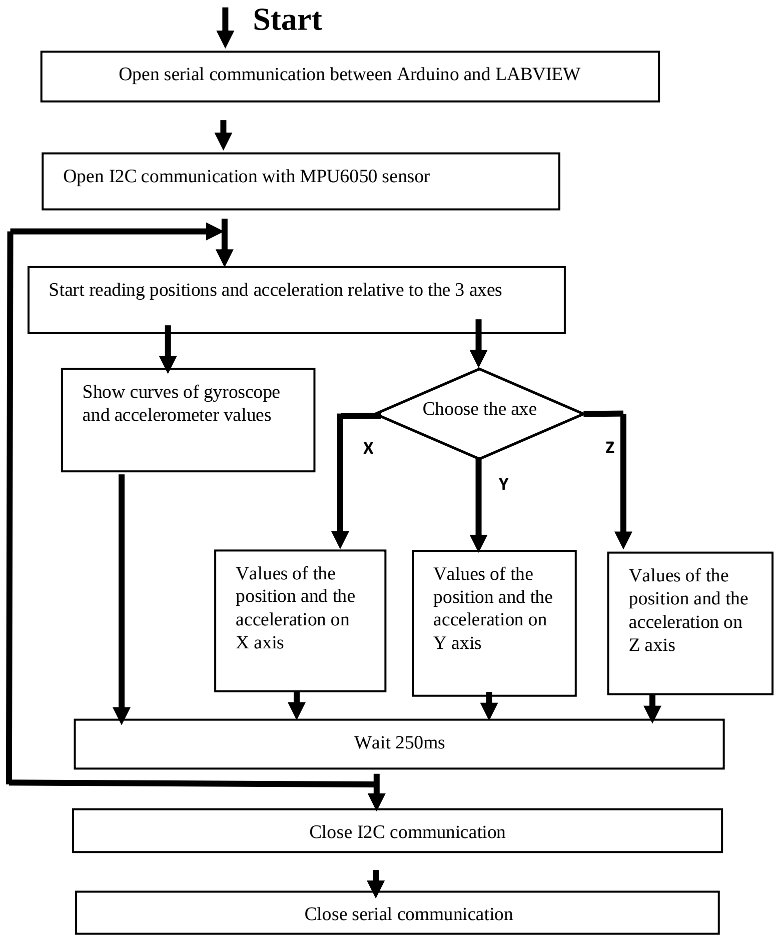


Figure 7: LabVIEW interface block Diagram

The flowchart below summarizes the general structure of the code:


Start
Open serial communication between Arduino and LABVIEW
Open I2C communication with MPU6050 sensor
Start reading positions and acceleration relative to the 3 axes
Show curves of gyroscope and accelerometer values
Choose the axe
X
Y
Z
Values of the position and the acceleration on X axis
Values of the position and the acceleration on Y axis
Values of the position and the acceleration on Z axis
Wait 250ms
Close I2C communication
Close serial communication

IV.4 – Communication:

➢ I2C communication:

I2C is a bidirectional half-duplex synchronous serial bus. Several equipment, either masters or slaves can be connected to the bus. The connection is made via two lines:

• SDA (Serial Data Line): bidirectional data line,

• SCL (Serial Clock Line): bidirectional synchronization clock line

In this perspective, we have implemented this technology to ensure communication between the MPU6050 and the microcontroller.

➢ Arduino-LabVIEW communication:

The serial communication is ensured by the LINX module

VI-Experimental result:

VI.1- Final structure of the robot:

In this part we will present the final structure of the project and the curves obtained from the LabVIEW interface. The picture bellow shows the final assembly of the robot

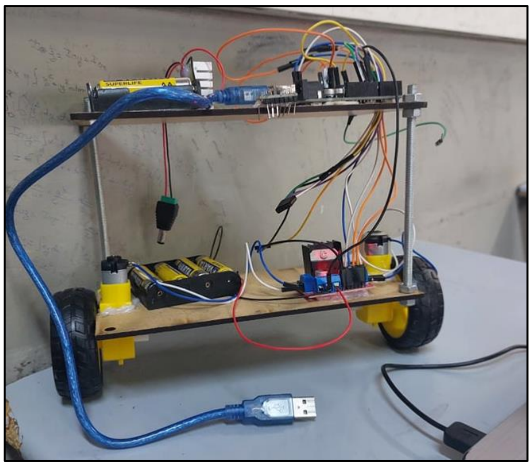


Figure 8:Final structure of the robot

- The two plates with the chassis are connected via screws and nuts.
- In order to have a good measurement of the angle of inclination of the chassis, the MPU model is laid flat in the center of rotation of the wheels.

VI.2- LABVIEW evolution curves:

Once the communication between the Arduino card and LabVIEW is ensured, we can observe the evolution of the values read by the gyroscope as well as the accelerometer on our interface.

the figures below represent an example of simulation during operation of the robot

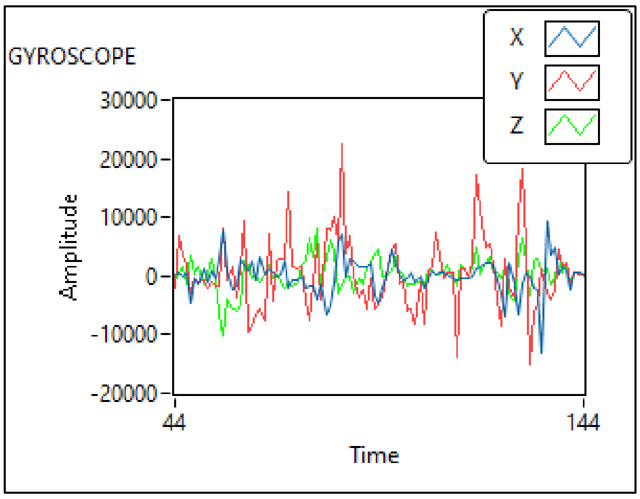


Figure 9:Gyroscope curves

⇒ These curves represent the evolution of the values read by the gyroscope for the three axes. As it is clear, the values read on the Y axis evolve in a considerable way which leads us to say that the position of robot always changes according the Y axis.

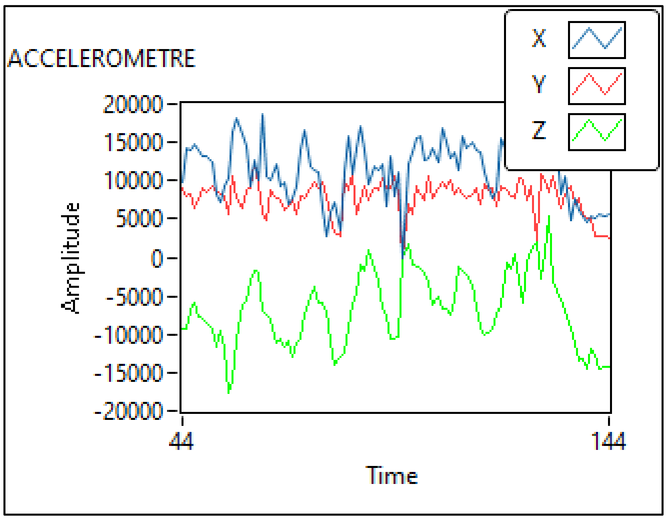


Figure 10:Accelerometer curves

- The curves below represent the evolution of the acceleration values along the three axes X, Y and Z we note that the amplitude of the highest acceleration is that of the X axis.
- In order to guarantee the stability of the robot we should focus on the X and the Y axes.
- The link of the video showing the operation of the robot with the LabVIEW interface is in the bibliography [4].

VI.3- Budget:

Table 3:Components cost

Elements	**Reference**	**Cost**
ARUIONO UNO	ARDUINO UNO	**0**
Motor + Wheel	6VMOTEUR DC 3-	**9 DT*2 =18 DT**
Gyroscope accelerometer module	MPU6050 - GY521 ACCELEROMETRE ET GYROSCOPE 3 AXES	**14.000 DT**
L298N module	L298N DC Motor Driver Module	**12.800 DT**
Batteries	Motor power supply battery	**0**
	1 PACK 4 BATTERIES Arduino power supply	**4.500 DT**
Coupler-battery	4*AA BATTERY COUPLER WITH JACK 5.5 CONNECTOR	**4.800 DT**
Total cost	**54.100 DT**	
Individual cost	**13.525 DT**	

Conclusion and prospect

This project was very instructive that we were able to put our technical knowledge into practice. We started our work with a mathematical modeling of the system, then we worked on the electrical design of the robot: material choice and wiring diagram.

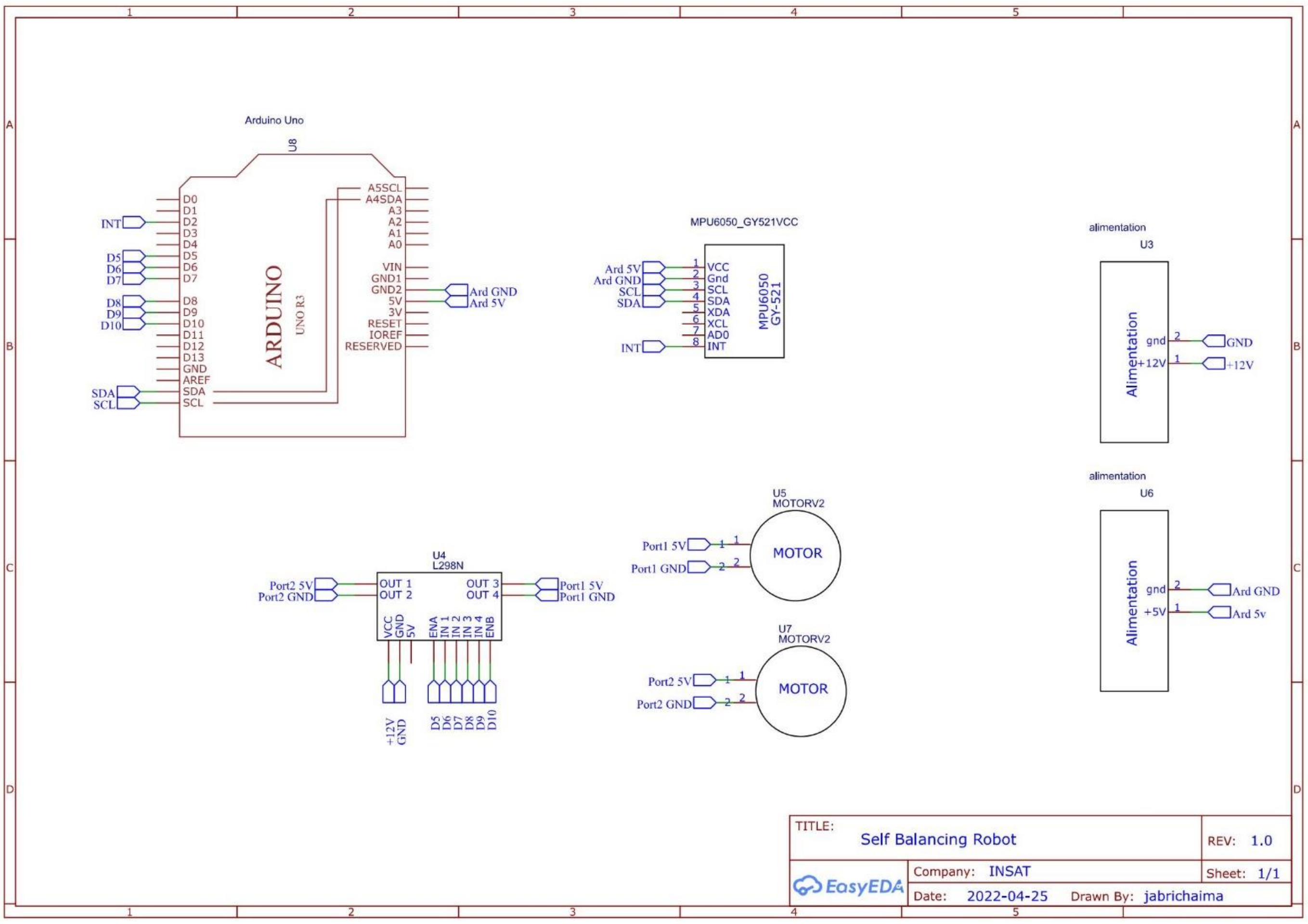
we studied the mechanical design: the dimensions as well as the positioning of the materials in order to guarantee the stability of the robot and finally we developed the code necessary for the operation and the control interface with LabVIEW.

During the realization of this project, we did not succeed to balance our robot in an optimal way. This is due to two main causes:

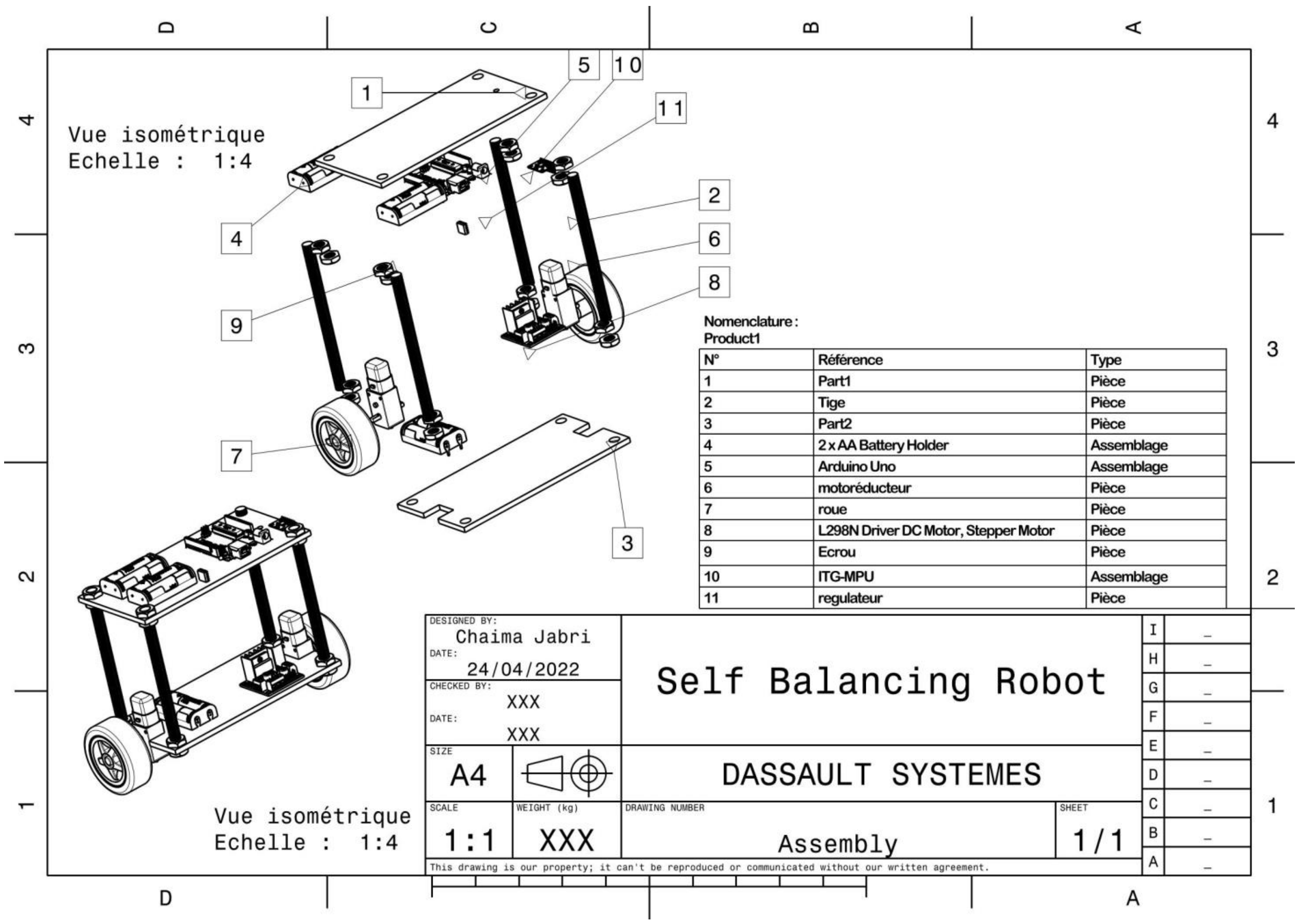
- The malfunction of the MPU6050 sensor: Erroneous values are being returned and it can be explained by a defect in the pin of clock SCL.
- The control method using PID regulator: In fact, it is capable to control the robot in the absence of the precise knowledge of the system parameters, but do not achieve optimal control.

As prospect, we aim add an LQR regulation based on the Riccati equation to our robot in order to enhance the control system.
Furthermore, we want to implement an artificial intelligence algorithm to ensure the stability of robot due an external disturbance.

Appendix I


Arduino Uno
U8
ARDUINO
UNO R3
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
GND
AREF
SDA
SCL
A5SCL
A4SDA
A3
A2
A1
A0
VIN
GND1
GND2
5V
3V
RESET
IOREF
RESERVED
INT
D5
D6
D7
D8
D9
D10
SDA
SCL
Ard GND
Ard 5V
MPU6050_GY521VCC
MPU6050
GY-521
VCC
Gnd
SCL
SDA
XDA
XCL
AD0
INT
Ard 5V
Ard GND
SCL
SDA
INT
alimentation
U3
Alimentation
gnd
+12V
GND
+12V
alimentation
U6
Alimentation
gnd
+5V
Ard GND
Ard 5v
U5
MOTORV2
MOTOR
Port1 5V
Port1 GND
U7
MOTORV2
MOTOR
Port2 5V
Port2 GND
U4
L298N
OUT 1
OUT 2
OUT 3
OUT 4
VCC
GND
5V
ENA
IN 1
IN 2
IN 3
IN 4
ENB
Port2 5V
Port2 GND
Port1 5V
Port1 GND
+12V
GND
D5
D6
D7
D8
D9
D10
TITLE: Self Balancing Robot
REV: 1.0
Company: INSAT
Sheet: 1/1
EasyEDA
Date: 2022-04-25
Drawn By: jabrichaima

Appendix II



Nomenclature :
Product1

N°	Référence	Type
1	Part1	Pièce
2	Tige	Pièce
3	Part2	Pièce
4	2 x AA Battery Holder	Assemblage
5	Arduino Uno	Assemblage
6	motoréducteur	Pièce
7	roue	Pièce
8	L298N Driver DC Motor, Stepper Motor	Pièce
9	Ecrou	Pièce
10	ITG-MPU	Assemblage
11	regulateur	Pièce

DESIGNED BY: Chaima Jabri	Self Balancing Robot		I _
DATE: 24/04/2022			H _
CHECKED BY: XXX			G _
DATE: XXX			F _
SIZE A4	DASSAULT SYSTEMES		E _
			D _
SCALE 1:1 / WEIGHT (kg) XXX	DRAWING NUMBER Assembly	SHEET 1/1	C _
			B _
			A _

This drawing is our property; it can't be reproduced or communicated without our written agreement.

Bibliography

[1]: DESIGN AND DEVELOPMENT OF SELF BALANCING ROBOT
Autor: 1:N.Meheswara Venkata Sai 2: S.Greeshma Srimani 3 G.Ajith Kumar Reddy

International Journal of Research in Engineering, IT and Social Science, ISSN 2250-0588, Impact Factor: 6.565, Volume 09, Special Issue 2, May 2019, Page 20-32

[2]: https://easyeda.com/editor?fbclid=IwAR1mp_eYU2CrgcVISBIAUTSOVC19WIfjCjvwGxIaQDkS601v2zrO3Rhj4FI

[3]: Optimal control of a two-wheeled self-balancing robot by reinforcement learning

Autor: Linyuan Guo,Syed Ali Asad Rizvi,Zongli Lin : **Robust and nonlinear control**

[4]: https://drive.google.com/file/d/1Ac84t8cAoO6BSk_vOToJkgeRSDR43c1U/view?usp=sharing