Project of a Self-Balancing Robot Using a PIC Microcontroller

Prof. Fernando Silveira Madani, Mauá Institute of Technology

Fernando Silveira Madani received the B.S (1998) in Mechatronics Engineering from the Univ. Paulista – Brazil, the M.S. (2002) and Ph.D. (2010) from the Aeronautical Institute of Technology (ITA) - Brazil. In 2002, he joined the faculty of the Dept. of Mechanical Engineering, Mauá Institute of Technology – Brazil, where he is currently as a full professor and Head of the Mechatronics Engineering program. His main research interests include, robotics, advanced manufacturing systems, embedded systems, and autonomous mobile robots. Since 2014 is an INEP (agency linked to the Ministry of Education) advisor, to promote the evaluation and improvement of undergraduate courses in mechatronics engineering in Brazil.

Mrs. Andressa Corrente Martins, Instituto Maua de Tecnologia

Andressa Martins is holds a master's degree in Aerospace Systems and Mechatronics with a focus on Robotics from the Aeronautics Institute of Technology and a degree in Control and Automation Engineering from the Universidade Paulista. Currently, she is a professor at the Mauá Institute of Technology. She has experience in the field of Control and Automation Engineering (Mechatronics), working mainly on the following topics: Intelligent Systems, Digital Twin, Industrial Automation, Instrumentation, Robotics, Mechanism Design, and Engineering Fundamentals.

Julia Meneses Roberto, Instituto Mauá de Tecnologia

I have technical training in mechatronics from Senai Almirante Tamandaré, completed in 2019, and I am currently a student on the Control and Automation Engineering course. Additionally, I work as a project monitor at Instituto Mauá de Tecnologia, where I have been working on the development of a unified module for instrumentation and monitoring of bridges and viaducts. Furthermore, I have experience in the field of robotics, resulting from my participation in academic competition teams. In this context, I highlight my work in projects related to autonomous robots, in which I was able to apply theoretical and practical knowledge acquired throughout my academic and professional career.

Marcelo Sacilotti Villas Boas

Control and Automation Engineering student in Instituto Maua de Tecnologia, interested in subjects like mechanics, electronics, programming and control, works as intern at Instituto Maua de Tecnologia in a autonomus vehicles research group.

Dr. Anderson Harayashiki Moreira, Instituto Mauá de Tecnologia

Graduated in Control and Automation Engineering from Instituto Mauá de Tecnologia (IMT) (2008). Master in Mechatronics Engineering from the Instituto Tecnológico de Aeronáutica (ITA) (2011). PhD in Mechatronics Engineering from the Instituto Tecnológico de Aeronáutica (ITA) (2017). He is currently a professor at the Instituto Mauá de Tecnologia. He develops activities and research in the area of mobile autonomous robotics, control systems, industrial robotics and microcontroller systems.

Alexandre Harayashiki Moreira

Alexandre Harayashiki Moreira received M.S. (2017) in multirobot system from UFABC Univ., Brazil. Since 2016, he has been a Member of Control & Automation Research group at Maua Institute of Technology focused on autonomous robots and smart cities.

Self-Balancing Robot Using a PIC Microcontroller

ABSTRACT

The project developed as part of the subjects Instrumentation, Microcontrollers, and Control Systems, studied in the fourth year of the Control and Automation Engineering course, aims to create a robot that simulates the behavior of an inverted pendulum. This implies designing a robot capable of autonomously balancing on two wheels, interpreting data provided by sensors, and taking actions based on that data. This project provides an opportunity for the practical application of the concepts covered in these subjects, emphasizing the integration of knowledge in a single project. An inverted pendulum is an unstable system since its center of mass is located above the pivot point, tending to fall. To keep the system in balance, it is necessary to incorporate key elements, including a measurement unit that combines a 3-axis accelerometer and a 3-axis gyroscope, stepper motors as actuators, and a PIC microcontroller. After the mechanical and electronic construction of the robot, the next step is processing sensor data through developed software to obtain more precise and stable readings. Then, a PID controller is implemented, responsible for adjusting the position of the support point and the applied force to maintain the robot's balance continuously. The angle at which the robot is in relation to the ground is visible through an application that connects to the system via a Bluetooth module. During the project's development, it was possible to recognize the importance of certain aspects to ensure the effective performance of the system. Firstly, there is the need for a robust physical structure that ensures the alignment of the robot's wheels and the correct distribution of its mass. Furthermore, it is essential to ensure the proper synchronization of the control loop with the previously defined sampling period to allow the controller to react consistently to possible disturbances. These findings play a crucial role in the successful development and operation of the system.

Keywords: Integrative project, multidisciplinary project, control, instrumentation and robotics.

INTRODUCTION

The present report describes the development of a project within the scope of the Control and Automation Engineering course. The project aims to create a robot capable of simulating the behavior of an inverted pendulum. The challenge of this system lies in its notorious instability, as the center of mass is above the pivot point, making it prone to fall.

The inverted pendulum is a classic example in control and automation, representing a dynamic system that challenges the stabilization and control capabilities of automated systems. In this context, the proposed project seeks to replicate this complexity by requiring the design of a robot with the ability to autonomously maintain its balance on two wheels.

The theory behind the inverted pendulum involves the dynamics of a rigid body in motion, where factors such as the location of the center of mass and the influence of gravity play crucial roles. To address this challenge, the project incorporates essential elements, including a sensor composed of a 3-axis accelerometer and a 3-axis gyroscope. This sensor is designed to provide data on the robot's tilt and acceleration, enabling the system to take corrective actions in real-time.

Additionally, stepper motors are strategically positioned to function as propellers, providing the necessary adjustments to maintain the robot's balance. The coordination of all these operations is carried out by a microcontroller, while software processes the sensor data, ensuring stable readings for decision-making in system control.

The implementation of a PID controller is of crucial importance, as it plays a fundamental role in adjusting the position of the support point and applied force, ensuring the continuous balance

of the robot. For visualization, the system has a user interface that allows monitoring the angle at which the robot is positioned relative to the ground, through an application connected to the system via a Bluetooth module.

When discussing integrated projects, one widely utilized approach is Project-Based Learning (PjBL) [5] [6]. Bacich and Moran [7] define PjBL as a methodology in which students engage with tasks and challenges to solve real-world problems or develop projects relevant to their lives beyond the classroom or workplace. Throughout this process, students address interdisciplinary issues, make decisions, and collaborate both independently and in teams.

Utilizing PjBL principles [8], the integrated project encompassing Instrumentation, Microcontrollers, Programming, and Control Systems has been structured into stages to enhance student comprehension and improve the application dynamics and feedback on results. These stages are divided into five parts:

1. Defining the problem and project theme.
2. Conducting theoretical research.
3. Constructing prototypes and conducting validation tests.
4. Delivering oral and practical presentations.
5. Writing a scientific report.

By tackling these theoretical and practical challenges, the project not only offers an opportunity for applying knowledge gained the disciplines but also demonstrates the integration of multiple engineering domains into a cohesive project. This report serves to document the progress and essential aspects of development.

Below is the project evaluation rubric:

Figure 1

Description	0 points	1 point	2 points	3 points
Theoretical concepts in the project	Failed to apply the theoretical concepts and the operation did not provide a correct result, for the most part.	Failed to apply the theoretical concepts, but most of the circuits showed correct results.	Failed to apply the theoretical concepts, but the operation showed correct results.	Managed to apply the theoretical concepts and circuits presented with correct results.
Project development	The practical project was not developed in any part.	The project was developed partially, lacking a conclusion regarding the operation of the circuit.	The project was developed partially, presenting the circuit definition processes.	The project was fully developed, presenting all the steps in detail.
Circuit implementation	Failed to implement the circuits for the most part.	Managed to implement most of the circuits, but many interventions were necessary.	Managed to implement most of the circuits, with few corrections.	Managed to implement all the circuits, practically autonomously.
Programming	Failed to program the circuits for the most part.	Managed to program most of the circuits, but a lot of interventions were necessary.	Managed to program most of the circuits, with few corrections.	Managed to program all the circuits, practically autonomously.
Result interpretation	Don't interpret the results in most or all of them.	Interpreted a few results.	Interpreted part of the results.	Interpreted all results.
Project presentation	Inadequate presentation, without minimal information about the development of the work and non-functional final circuit.	Presentation with little information about the development of the project and the final circuit partially works.	Presentation with the necessary information, but the final circuit partially works.	Presentation with all the necessary information and final circuit in full operation.
Integration between subjects	The project doesn't integrate the subjects.	The project was developed in a partially integrated way, but with fault.	The project was developed in a partially integrated way without fault.	The project was developed in an integrated manner.
Participation of team members	the work was carried out with the contribution of a small part of the team, or it was not carried out at all.	The work was carried out with the contribution of only half of the team.	The work was carried out with the contribution of most of the team members.	The work was carried out with the contribution of all team members.
Project documentation	The inadequate report, without following formatting, and little information about the project.	The inadequate report, without formatting, but with information about the project.	The adequate report, but with little information about the project.	The adequate report, with all the information about the project.

THEORETICAL RESEARCH

Here, we will cover the complete development of the project, including the mechanical assembly of components, the design and fabrication of the PCB, and the programming of the PIC microcontroller.

Components Used:

- 2 Nema 17 Stepper Motors
- 1 3D-Printed Motor Base with PLA
- 1 3D-Printed Electronics Base with PLA
- 2 3D-Printed Sides with PLA
- 2 3D-Printed Wheels with PLA
- 1 PIC 16F1619 Microcontroller
- 2 Motor Drivers
- 1 MPU6050 Accelerometer
- 1 Bluetooth Module
- 1 10kΩ Resistor
- 1 100 µF Capacitor
- 1 3S LiPo Battery
- 1 Slide Switch
- Screws and nuts

Mechanic Project

The mechanical system of the pendulum is relatively simple, consisting of a pair of parallel stepper motors and, above them, a structure for mounting the electronic components and the system's battery. This assembly configuration represents an inverted pendulum system.

The robot's mechanical design was developed using SolidWorks software, and the physical design was 3D-printed with PLA material.

For this system, which requires high torque and acceleration, several things were considered to arrive at a mechanical assembly that best meets these requirements.

In choosing the motors, NEMA 17 stepper motors were preferred over traditional DC motors, as they offer significant benefits. While DC motors would require an encoder or position sensor, along with a reduction gearbox that could introduce backlash and complicate precise control, stepper motors provide notable advantages. These motors offer precise angular movements and positioning, facilitate control through electrical pulses, eliminate the need for position feedback, maintain good torque at low speeds, and provide simplicity in control. These characteristics make stepper motors an effective choice for the precise control of the robot at low speeds.

Figure 2

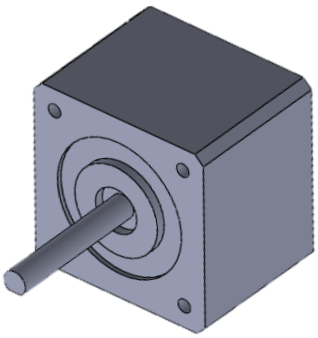
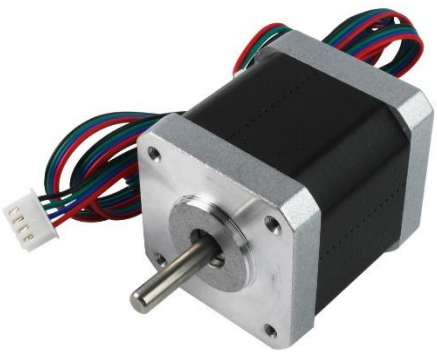


Figure 3



For the wheels, it was observed that larger diameter wheels offer greater torque due to the increased moment of inertia. Additionally, the radial acceleration of the system should also be high, and larger diameter wheels provide higher acceleration due to the expanded circumference of the wheel. With fewer motor rotations, we achieve a greater correction distance for the pendulum, contributing to a more efficient control of the system.

Figure 4

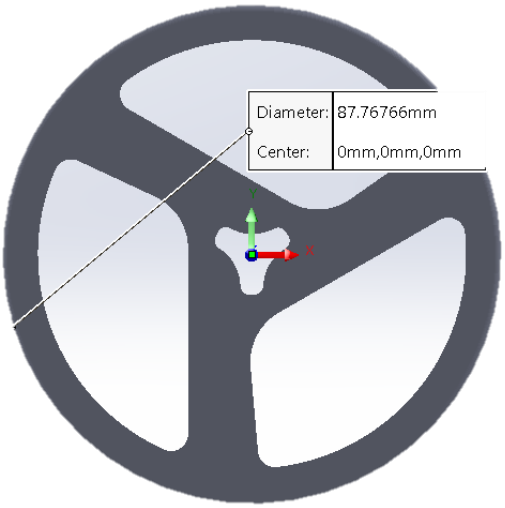


Figure 5

In the design of the structure, stability is achieved when the center of mass is directly aligned above the pivot point, ensuring a stable response to disturbances. In this context, the adopted solution was to position the mass as high as possible, aiming to emphasize control action as the predominant factor for system stability. This decision involves placing the electronic board and the battery at the highest point, creating an initial condition of controlled instability but allowing a controlled response to disturbances through control action.

Figure 6

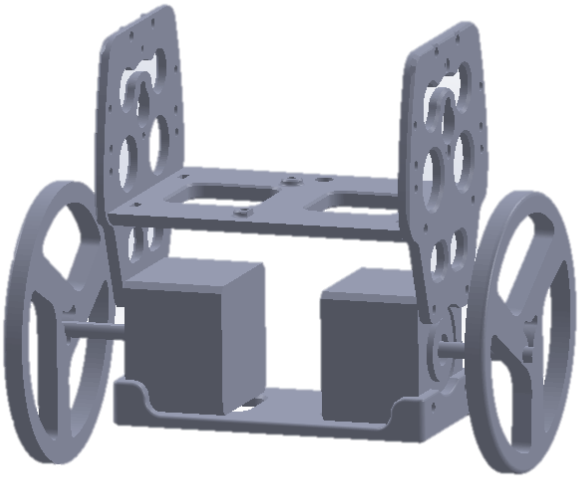
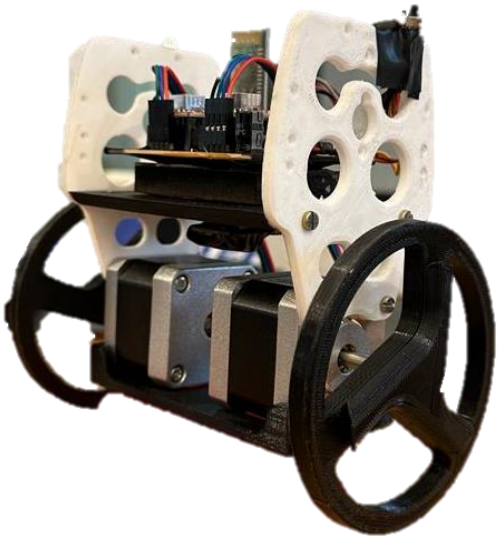


Figure 7

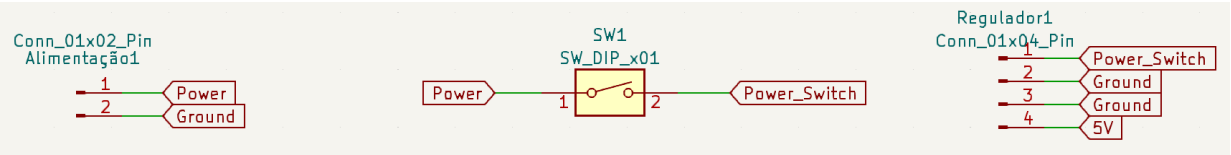


Electronic Project

The electronic circuit of the project was developed to meet all the requirements on a single printed circuit board. The board can be divided as follows:

For the project power supply, a Lithium-Polymer battery was used, providing a voltage around 11.1V and connected to the board via a JST-type connector. Additionally, a small slide switch was added, used to turn the system on and off. To power the components, a voltage regulator was implemented, reducing the input voltage to 5V.

Figure 8



For the motor control, an A4988 driver was used for each motor. They receive power from the battery, and a capacitor connected to the GND is used to smooth the power delivery and filter electrical noise, preventing fluctuations that may occur during the operation of the stepper motor.

Figure 9

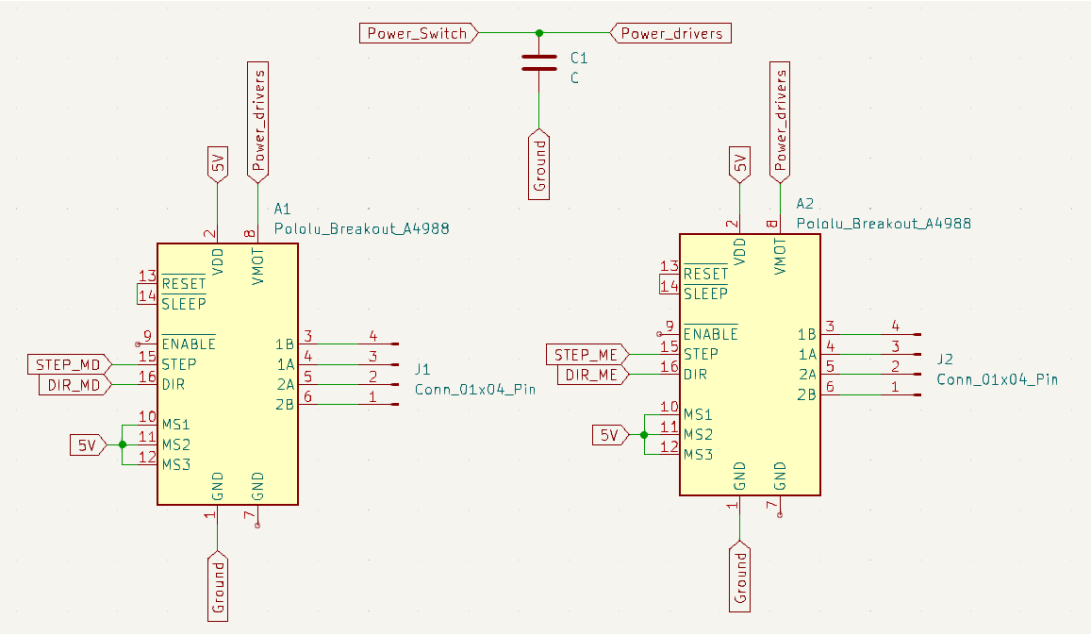
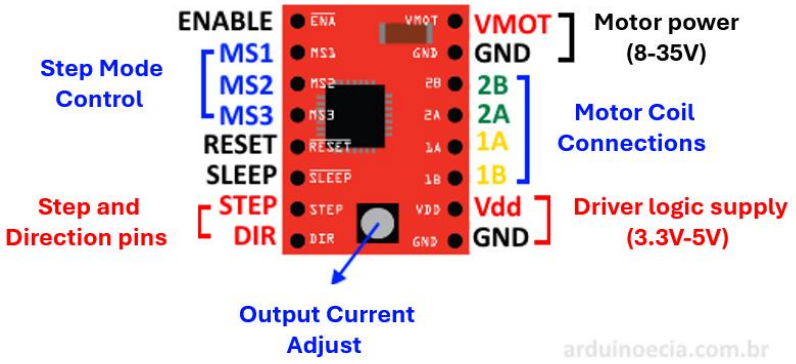


Figure 10



The drivers were configured in microstep mode, where each pulse on the STEP pin generates a movement in the motor of 1/16 of a step. It is known that the Nema17 motor rotates 1.8 degrees per step, so there is a resolution of 0.1125 degrees per pulse.

Figure 11

Pins			
MS1	**MS2**	**MS3**	**Microsteps Resolution**
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	Quarter step (1/4)
High	High	Low	Eighth step (1/8)
High	High	High	Sixteenth step (1/16)

The position data acquisition is performed through the MPU6050 accelerometer, a sensor that measures accelerations in various axes. The MPU6050 operates with I2C (Inter-Integrated Circuit) communication, facilitating integration with microcontroller systems. In addition to providing data on linear acceleration in the x, y, and z axes, the sensor also includes a gyroscope, expanding the capability to obtain information about the robot's orientation in relation to the rotation axes.

Figure 12

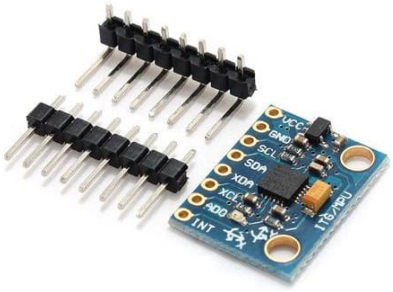
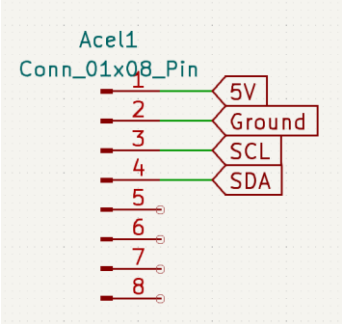


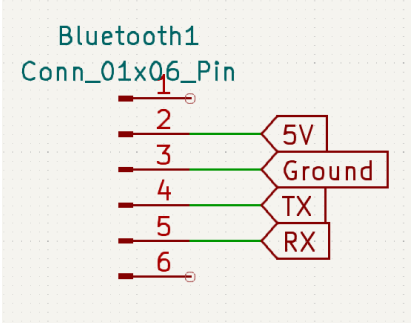
Figure 13



For serial communication, a Bluetooth module was utilized to enable effective interaction between the microcontroller and an Android device. This integration facilitates the transmission of relevant data, allowing a controlled variable to be displayed in real-time through a dedicated serial port reading application. The use of the module enables remote monitoring of the controlled system.

Figure 14

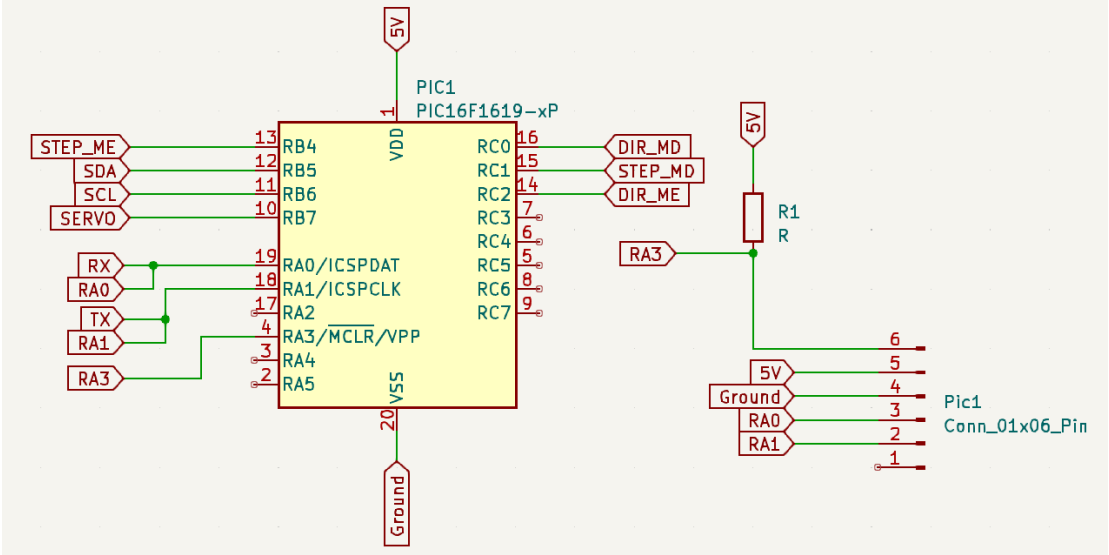
Figure 15



For the programming execution, the Pic16f1619 microcontroller was chosen. Additionally, a dedicated connection to the microcontroller programmer was incorporated to facilitate any reprogramming needs throughout the project development.

Figure 16

Figure 17



The electrical schematic was created using the KiCad software. Tests were initially conducted individually for each component using an Arduino Nano and then with the PIC16f1619. Subsequently, the assembly was mounted on a breadboard, and after testing and confirming the circuit, the PCB (Printed Circuit Board) file was developed and the board was milled using the W400 block. All components were soldered into place.

Figure 18

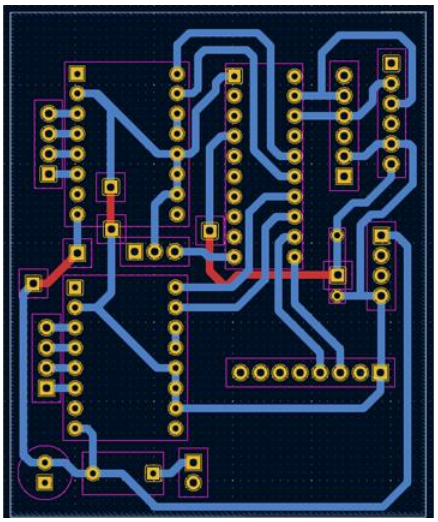
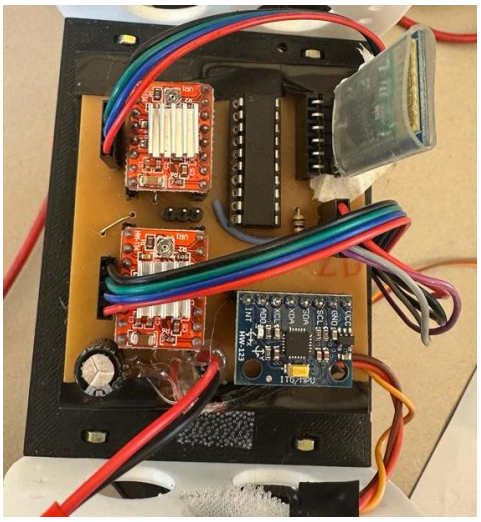


Figure 19



Programming

The programming of the PIC16F1619 was carried out in the Microchip's MpLab software. The MCC (Mplab Code Configurator) feature was employed to assist in the microcontroller configuration. Through MCC, it is possible to configure various settings, including internal and external clock speed, digital and analog input and output pins, PWM (Pulse Width Modulation), timer overflow interrupts, and other functionalities.

The code will be explained in parts as follows:

Reading the accelerometer and gyroscope module

The I2C protocol was used for communication with the MPU6050 module. For this purpose, a feature of MCC called MSSP was employed to configure the use of this communication protocol. To simplify usage, values representing the registers of this module and their addresses were defined.

Just before the main code loop, the configuration of the MPU6050 is performed. Initially, a command is given to restart it, ensuring that any previous configuration is cleared:

Figure 20

```
i2c_writelByteRegister(MPU6050_ADDRESS, 0x6B, 0x80);
__delay_ms(125);
i2c_writelByteRegister(MPU6050_ADDRESS, 0x6B, 0x01);
i2c_writelByteRegister(MPU6050_ADDRESS, 0x6B, 0x03);
```

After that, the unit scales for both the gyroscope and accelerometer are configured. The gyroscope is set to read up to 500 degrees/second, and the accelerometer is set to an acceleration of up to 2G, which is twice the acceleration due to gravity. A low-pass filter present in the module is also configured with a cutoff frequency of 10Hz, acting on the accelerometer signal. Additionally, a sampling rate of 100Hz is configured, the same as used in the main code loop.

Figura 21

```
i2c_writelByteRegister(MPU6050_ADDRESS, 0x1B, 0x01); // Config Gyro scale (500deg/seg)
i2c_writelByteRegister(MPU6050_ADDRESS, 0x1C, 0x00); // Config Accel scale (2g)
i2c_writelByteRegister(MPU6050_ADDRESS, 0x1A, 0x05); // Config Digital Low Pass Filter 10Hz
i2c_writelByteRegister(MPU6050_ADDRESS, 0x19, 0x09); // Set Sample Rate to 100Hz
i2c_writelByteRegister(MPU6050_ADDRESS, 0x38, 0x00); // Data ready interrupt enable
i2c_writelByteRegister(MPU6050_ADDRESS, 0x6B, 0x01);
```

After this, the gyroscope calibration is performed, which will be discussed next:

Figure 22

```
__delay_ms(1000);
printf("IMU calibrando, nao mexa");
calibrar();
__delay_ms(100);
printf("Pronto");
__delay_ms(1000);
```

The gyroscope has an inherent imprecision, wherein even with the IMU completely static, it detects an angular velocity. Therefore, a calibration period is necessary in which 100 measurements are taken with a 1ms interval. After this, an average is calculated and assigned to the variable x_gyro_offset, which will be subtracted from the gyroscope value later on.

Figure 23

```
void calibrar(){

    for (int i = 0; i < n_calibration; i++) {
        LerMPU();
        gyroX += x_gyro;
        __delay_ms(1);
    }
    x_gyro_offset = gyroX / n_calibration;


    LerMPU();

    angle = y_accel*v57;
    angle = angle/z_accel;
}
```

To read the IMU data, the subroutine below is used:

Figure 24

```
void LerMPU(){

    z_accel_h = i2c_read1ByteRegister(MPU6050_ADDRESS, MPU6050_ACCEL_ZOUT_H);
    z_accel_l = i2c_read1ByteRegister(MPU6050_ADDRESS, MPU6050_ACCEL_ZOUT_L);

    y_accel_h = i2c_read1ByteRegister(MPU6050_ADDRESS, MPU6050_ACCEL_YOUT_H);
    y_accel_l = i2c_read1ByteRegister(MPU6050_ADDRESS, MPU6050_ACCEL_YOUT_L);

    x_gyro_h = i2c_read1ByteRegister(MPU6050_ADDRESS, MPU6050_GYRO_XOUT_H);
    x_gyro_l = i2c_read1ByteRegister(MPU6050_ADDRESS, MPU6050_GYRO_XOUT_L);

    z_accel = ((z_accel_h << 8) | z_accel_l);
    y_accel = ((y_accel_h << 8) | y_accel_l);

    x_gyro = ((x_gyro_h << 8) | x_gyro_l);

}
```

In the case of this project, only 3 axes of the IMU are read: Y and Z from the accelerometer and X from the gyroscope. As the communication allows the transmission of only 8 bits at a time, two 8-bit variables are needed to form the 16-bit value. For example, the variable z_accel receives z_accel_h and z_accel_l. After that, a bit-shifting operation is performed to shift z_accel_h, representing the 8 most significant bits, to the left. Following this step, it is combined with z_accel_l to form z_accel.

Figure 25

```
void geraAngulo(){
    angle_gyro += ((x_gyro - x_gyro_offset)*2000)/6550;
    angle_accel = y_accel*v57;
    angle_accel = angle_accel/z_accel;

    // angle_gyro 0.001 graus/LSB
    // angle_accel  1 grau/LSB
    // angle 1 grau/LSB

    angle = ((angle_gyro) * 80 + angle_accel * 20 *1000)/(100*1000);

}
```

Above, you can see the `geraAngulo` subroutine. In its first line, it calculates the angle read by the gyroscope. The first step is to subtract the gyroscope reading by the offset calculated during the calibration. After that, to obtain the unit of degrees/second, the value should be divided by 65.5, as per the table. However, since this value will be numerically integrated later, it needs to be multiplied by 0.01, which is the sampling period in seconds, or simply divided by 100. Therefore, 65.5 * 100 = 6550.

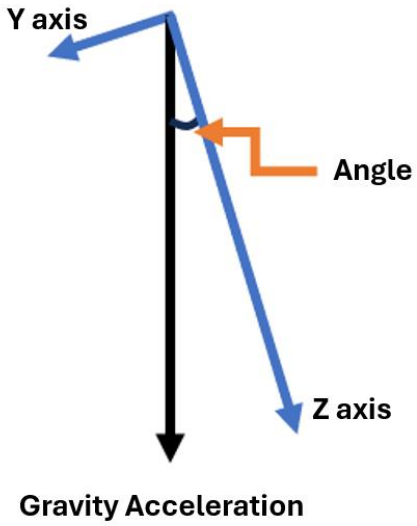
Figure 26

FS_SEL	Full Scale Range	LSB Sensitivity
0	± 250 °/s	131 LSB/°/s
1	± 500 °/s	65.5 LSB/°/s
2	± 1000 °/s	32.8 LSB/°/s
3	± 2000 °/s	16.4 LSB/°/s

The value 2 in the multiplier 2000 is a constant that needed to be added for unit adjustment, and the value 1000 in this multiplier had to be added because during the project development, it was found that this PIC model does not have a floating-point unit implemented. Calculations using variables like float and double demanded a lot of processing, making the execution of the control loop unfeasible. Therefore, strategies were employed to perform calculations only with int-type variables. In this case, the value was multiplied by 1000 to ensure that it does not become too small for addition. Thus, the unit of the variable `angle_gyro` is 0.001 degrees.

As for the angle read by the accelerometer, the commonly used methodology is as follows:

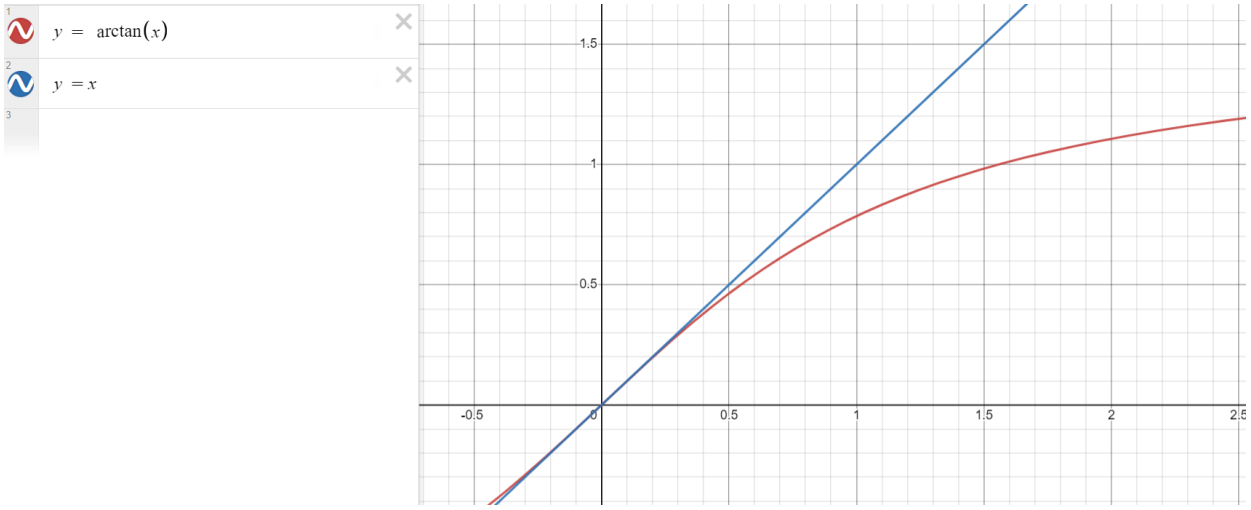
Figure 27



$$\boldsymbol{Angle} = \arctan\left(\frac{\boldsymbol{Y\ axis}}{\boldsymbol{Z\ aixs}}\right)$$

However, due to the previously mentioned floating-point issue, the use of the atan2 function, commonly employed in this type of situation, is not possible. Therefore, it was necessary to perform an approximation based on the analysis below:

Figure 28



For small angles (less than 30°), it can be considered that x = arctan(x) for generating an angle in radians. After this, it was necessary to multiply the angle by 180/pi to obtain it in degrees, resulting in a multiplier of approximately 57. However, directly multiplying it by the variable y_accel did not yield the expected result, likely due to the value 57 being an int, and y_accel being int32_t. Therefore, the variable v57 was created to ensure the success of this multiplication.

Conventionally, a complementary filter is implemented to combine the angles generated by the accelerometer and gyroscope. This filter is necessary because the accelerometer has the disadvantage of having a lot of noise, and the gyroscope has the disadvantage of accumulating error over time. Combining the two in a filter reduces these inaccuracies. The operation of this type of filter consists of a sum in which each of the angles has a weight, such that the sum of the weights is

always 1. Through tests, it was concluded that the best weight is 80% gyroscope and 20% accelerometer. Due to the floating-point issue, some adaptations were needed in the calculation.

Figure 29

```
angle = ((angle_gyro) * 80 + angle_accel * 20 *1000)/(100*1000);
```

Stepper motor control

To vary the motor speed, it's necessary to generate a square wave with a fixed duty cycle (50%) and a variable frequency. Below is the deduction of the constant that converts the desired rotation from RPM to the period applied in the square wave in microseconds.

$$\mathbf{0,1125}\frac{\boldsymbol{\cancel{degrees}}}{\boldsymbol{pulse}} \cdot \frac{\mathbf{1}}{\mathbf{360}}\frac{\boldsymbol{turn}}{\boldsymbol{\cancel{degrees}}} = \mathbf{3,125 \cdot 10^{-4}}\frac{\boldsymbol{turns}}{\boldsymbol{pulse}}$$

$$\frac{\mathbf{1}}{\mathbf{3,125 \cdot 10^{-4}}\frac{\boldsymbol{turns}}{\boldsymbol{pulse}}} = \mathbf{3200}\ \frac{\boldsymbol{pulses}}{\boldsymbol{turn}}$$

$$\mathbf{1}\frac{\boldsymbol{turn}}{\boldsymbol{minute}} \cdot \mathbf{3200}\frac{\boldsymbol{pulses}}{\boldsymbol{turn}} \cdot \frac{\mathbf{1}}{\mathbf{60}}\frac{\boldsymbol{minute}}{\boldsymbol{seconds}} \cdot \frac{\mathbf{1}}{\mathbf{1000000}}\frac{\boldsymbol{second}}{\boldsymbol{microsecond}}$$

$$= \mathbf{5,3333 \cdot 10^{-5}}\frac{\boldsymbol{RPM}}{\boldsymbol{microsecond}}$$

$$\frac{\mathbf{1}}{\mathbf{5,3333 \cdot 10^{-5}}\frac{\boldsymbol{RPM}}{\boldsymbol{microsecond}}} = \mathbf{18750}\frac{\boldsymbol{microsecond}}{\boldsymbol{RPM}}$$

As the value to be used in the code needs to be half of the period, we have:

$$\boldsymbol{halfPeriod} = \frac{\boldsymbol{n(RPM)}}{\mathbf{9375}}$$

The chosen method for generating the square wave was to use timer overflow interrupts. Timers 2, 4, and 6 of the PIC16F1619 allow their period to be changed during code execution, with the T2PR, T4PR, and T6PR registers defining the period, respectively.

The fact that the period register has only 1 byte in size led to the use of a combination of the 3 timers in the same code. Each timer has a different order of magnitude and is triggered one at a time, depending on the speed range desired.

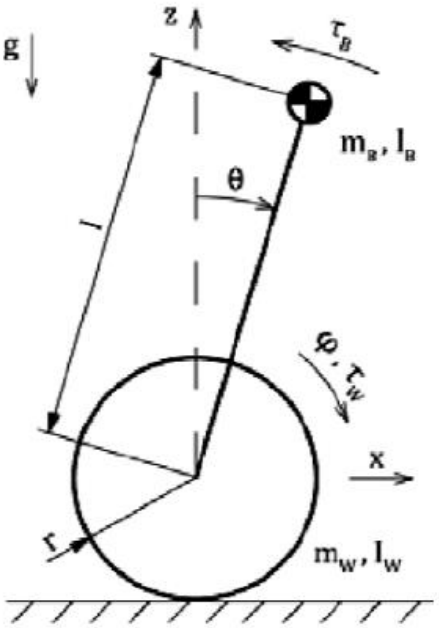
Below is the `moveMotor` function, responsible for implementing all the previously mentioned logic. Additionally, it has a minimum speed limiter to ensure that there is no division by zero in the period calculation.

Figure 30

```
void moveMotor(int vel){


  if(vel > 0){
    dirPin_ME_PORT = 1;
    dirPin_MD_PORT = 0;
  }
  else{
    vel = -vel;
    dirPin_ME_PORT = 0;
    dirPin_MD_PORT = 1;
  }

  if(vel < 1){
      vel = 1;
  }

  periodo = 9375/vel;

      if(periodo < 256){
          TMR2_StartTimer();
          TMR4_StopTimer();
          TMR6_StopTimer();

          T2PR = 9375/vel;
          interrupcao = 0;
      }

      else if((periodo > 255)&&(periodo < 2048)){
          TMR2_StopTimer();
          TMR4_StartTimer();
          TMR6_StopTimer();

          T4PR = 1172/vel;
          interrupcao = 1;
      }

      else if(periodo > 2047){
          TMR2_StopTimer();
          TMR4_StopTimer();
          TMR6_StartTimer();

          T6PR = 146/vel;
          interrupcao = 2;
      }

  }
```

Control
Modeling of a Balancing Robot

Figure 31



The mechanical system of the robot can be modeled as an inverted pendulum with wheels. In this model, two masses are considered: the mass of the entire robot structure excluding the wheels (mg) and the combined mass of the two wheels (mw). Additionally, the respective moments of inertia for each (Ig and Iw) are considered. The wheel has a radius (r), and the distance from the center of the wheels to the center of mass of the robot is represented by (l). The torque generated by the motor is denoted as (τB).

$$\boldsymbol{x = r\varphi}$$

$$\boldsymbol{x_{COG} = x + l \cdot sin(\theta)},$$
$$\boldsymbol{z_{COG} = x + l \cdot cos(\theta)},$$

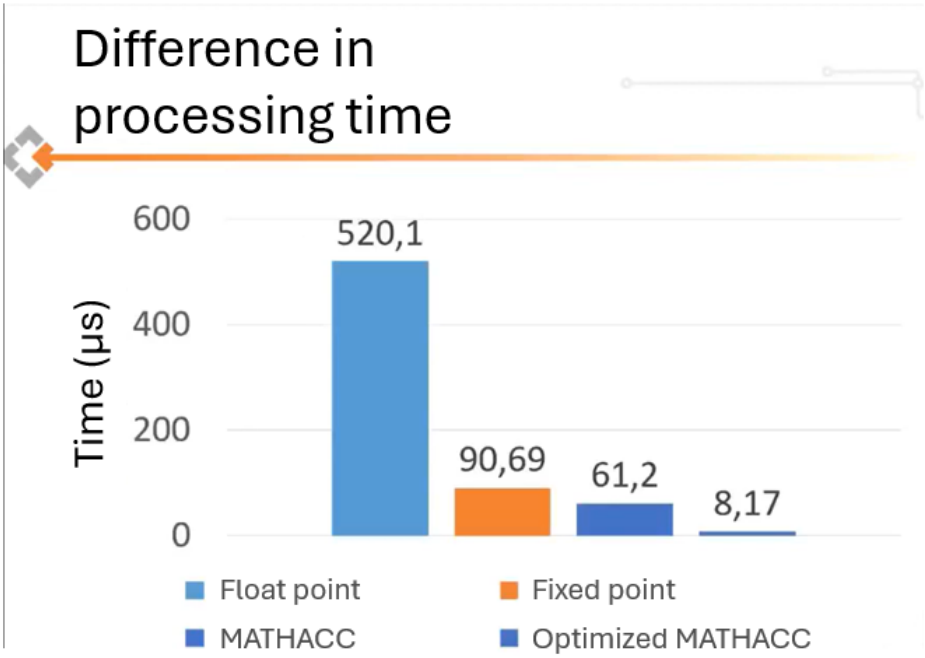
$$\boldsymbol{v_{COG} = \sqrt{\dot{x}_{COG}^{\,2} + \dot{z}_{COG}^{\,2}}}.$$

$$\boldsymbol{E_{kB} = \frac{1}{2} m_b v_{COG}^{\,2} + \frac{1}{2} I_b \dot{\theta}^2 = \frac{1}{2} m_b \left(\dot{x}^2 + 2\dot{x} l \cos(\theta)\,\dot{\theta} + l^2 \dot{\theta}^2\right) + \frac{1}{2} I_b \dot{\theta}^2}$$
$$\boldsymbol{E_{kW} = \frac{1}{2} m_W \dot{x}^2 + \frac{1}{2} I_w \dot{\varphi}^2 = \frac{1}{2} m_W \dot{x}^2 + \frac{1}{2} \frac{I_W}{r^2} \dot{x}^2}$$

Implementation of PID Control

For the robot control, the built-in PID module in the PIC was utilized, known as MATHACC. In the image below, a comparison can be observed between the manual implementation of PID and the use of the module in terms of processing time. This comparison was conducted by Professor Rodrigo Almeida, coordinator of the electronic engineering course at the Federal University of Itajubá.

Figure 32



Next, we will address how the module implements PID discretely. Everything begins with the transfer function of the controller in the Laplace domain.

$$\frac{C(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s$$

After that, a Z-transform was performed using the finite difference method. This method was chosen because it is not possible to perform the Z-transform using the table when the derivative part (Kd) of the PID is present.

$$\frac{C_{o(z)}}{E(z)} = \frac{\left(K_p + K_i T + \frac{K_d}{T}\right) + \left(-K_p{}^{o} - \frac{2K_d}{T}\right)z^{-1} + \left(\frac{K_d}{T}\right)z^{-2}}{1 - z^{-1}}$$

Then, the coefficients were grouped for the generation of K1, K2, and K3.

$$K1 = K_p + K_i T + \frac{K_d}{T}$$

$$K2 = -K_p - \frac{2K_d}{T}$$

$$K3 = \frac{K_d}{T}$$

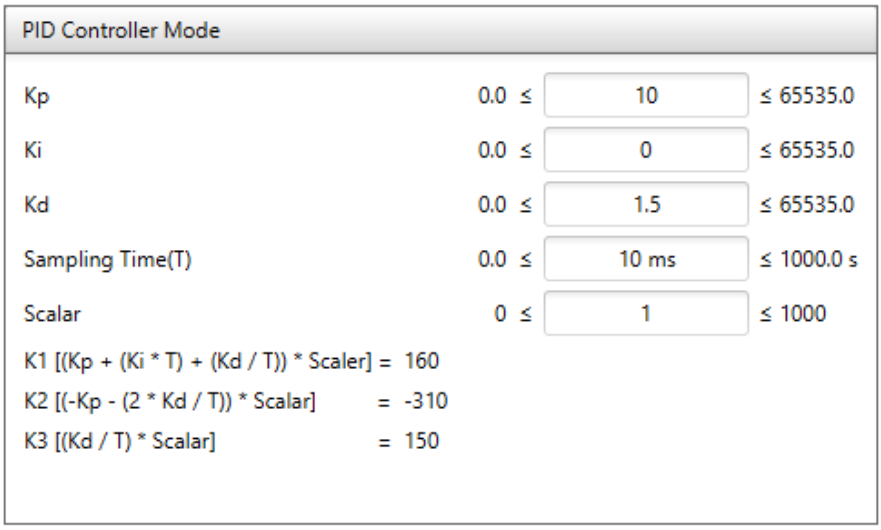
$$\frac{C_{o(z)}}{E(z)} = \frac{K1 + K2 \cdot z^{-1} + K3 \cdot z^{-2}}{1 - z^{-1}}$$

Finally, we have the difference equation in the sample domain.

$$C_0[k] = C_0[k-1] + {}^0K_1 e[k] + {}^0K_2 e[k-1] + {}^0K_{3(2)} e[k-2]$$

Below, you can see the configuration of this module through the MCC of MpLab. The MCC itself performs the necessary calculations for discretization based on the values of Kp, Ki, Kd, and the sampling time.

Figure 33



The coefficients Kp, Kd, and Ki were manually adjusted using the trial-and-error method, but the BROBOT project provided some clues that helped save adjustment time. Among them is the fact that the integral part of the controller is not used, keeping Ki at 0, and also the proportion between the values of Kp and Kd. Once Kp was manually adjusted, Kd was calculated to maintain the same Kp/Kd ratio used in the BROBOT project, which was around 6.

The sampling time differed a bit from the BROBOT project, which uses 5 milliseconds. In this project, it was decided to use 10 milliseconds to ensure that the microcontroller could process the entire algorithm consistently. Even though it is double the time, there was no noticeable impact on the robot's reaction speed.

VALIDATION TESTS

During the development of the project, various issues arose, necessitating adaptations. For example, the PIC used was unable to perform calculations with floating-point variables, leading to the addition of multipliers to increase precision in integer calculations. Another issue was that the PIC did not function correctly when the PID algorithm was manually implemented in the code, resulting in constant crashes. Finally, the problem of timer registers used to drive the stepper motors

having a size of only one byte was encountered, requiring division into 3 timers with different intervals that alternated depending on the magnitude order of the desired speed at the moment.

At the end of the project, the robot operated as expected. It can maintain a nearly static vertical position, and when small disturbances are applied, it can recover and return to equilibrium. However, when larger disturbances are applied, it struggles to regain its position afterward, not due to a lack of motor power but rather due to a deviation caused during the integration of angular velocity to generate the angle. This results in a kind of "zero-point shift" that causes the robot to continuously move horizontally.

CONCLUSION

In conclusion, the development of a balancing robot project is quite challenging. Initially introduced as a challenge to replicate the dynamic complexity of an inverted pendulum, the project involved the integration of theoretical and practical knowledge from the disciplines of Instrumentation, Microcontrollers, and Control Systems.

Throughout the process, the importance of the theory behind the inverted pendulum was highlighted, with the simulation requiring the implementation of sensors, motors, and control algorithms. The interdisciplinary approach allowed overcoming technical challenges, from the PIC's limitation in handling floating-point calculations to the need for adaptations in timer registers to drive stepper motors.

The results obtained at the end of the project are promising, demonstrating the robot's ability to maintain a nearly static vertical position in the face of minimal disturbances. However, persistent challenges, such as the deviation in the integration of angular velocity and the consequent "zero-point shift" in more intense disturbances, offer opportunities for continuous improvement.

The collaborative effort of three simultaneous disciplines enriched the project approach, emphasizing the complexity and breadth of the required skills. This project not only consolidated the theoretical knowledge acquired but also demonstrated the practical application of this knowledge in a challenging context.

Ultimately, the initiative to develop a balancing robot not only illustrates the successful integration of multiple engineering domains but also emphasizes the importance of problem-solving and innovation to address complex challenges in automation and control.

REFERENCES

[1] P. Frankovský et al., "Modeling of Two-Wheeled Self-Balancing Robot Driven by DC Gearmotors," International Journal of Applied Mechanics and Engineering, Slovakia, pp. 739-747, Sep. 2017. Available: https://www.researchgate.net/publication/319618827_Modeling_of_Two-Wheeled_Self_Balancing_Robot_Driven_by_DC_Gearmotors. Accessed: Oct. 19, 2023.

[2] Equipe Embarcados, "Webinar Gravado: Implementação de Controle PID com PIC16F1619," 2017. Available: https://embarcados.com.br/controle-pid-com-pic16f1619/. Accessed: Oct. 10, 2023.

[3] Unknown. "PID Control on PIC16F161X by using a PID Peripheral," 2015. Available: https://ww1.microchip.com/downloads/en/AppNotes/90003136A.pdf. Accessed: Oct. 10, 2023.

[4] JJRobots, "B-Robot: The Balancing Robot." Available: https://jjrobots.com/projects-2/b-robot/. Accessed: Oct. 21, 2023.

[5] H. Hassan, C. Domínguez, J. -M. Martínez, A. Perles, J. -V. Capella and J. Albaladejo, "A Multidisciplinary PBL Robot Control Project in Automation and Electronic Engineering," in IEEE Transactions on Education, vol. 58, no. 3, pp. 167-172, Aug. 2015. Consult. [Online]. Available: https://ieeexplore.ieee.org/document/6895312 [Accessed 2022-12-08].

[6] A. Luis Baptista Soares, "Metodologias Ativas para uma Prática Educativa Inovadora", in VII Congresso Nacional de Educação, Maceió, Brazil, 2020-10-17. Maceió: Realize, 2020, pp. 1–6. Consult. [Online]. Available: https://editorarealize.com.br/editora/anais/conedu/2020/TRABALHO_EV140_MD4_SA19_ID2989_15062020165910.pdf [Accessed 2022-04-20].

[7] L. Bacich e J. Moran, Metodologias ativas para uma educação inovadora: uma abordagem téorico-prática. Porto Alegre: Penso, 2018. Consult. [Online]. Available: https://curitiba.ifpr.edu.br/wp-content/uploads/2020/08/Metodologias-Ativas-parauma-Educacao-Inovadora-Bacich-e-Moran.pdf. [Accessed 2022-11-20].

[8] E. H.J. Yew and K. Goh, "Problem-Based Learning: An Overview of its Process and Impact on Learning", Health Professions Education, vol. 2, no. 2, pp. 75–79, 2016-12-01. Consult. [Online]. Available: https://doi.org/10.1016/j.hpe.2016.01.004 [Accessed 2022-12-08].