



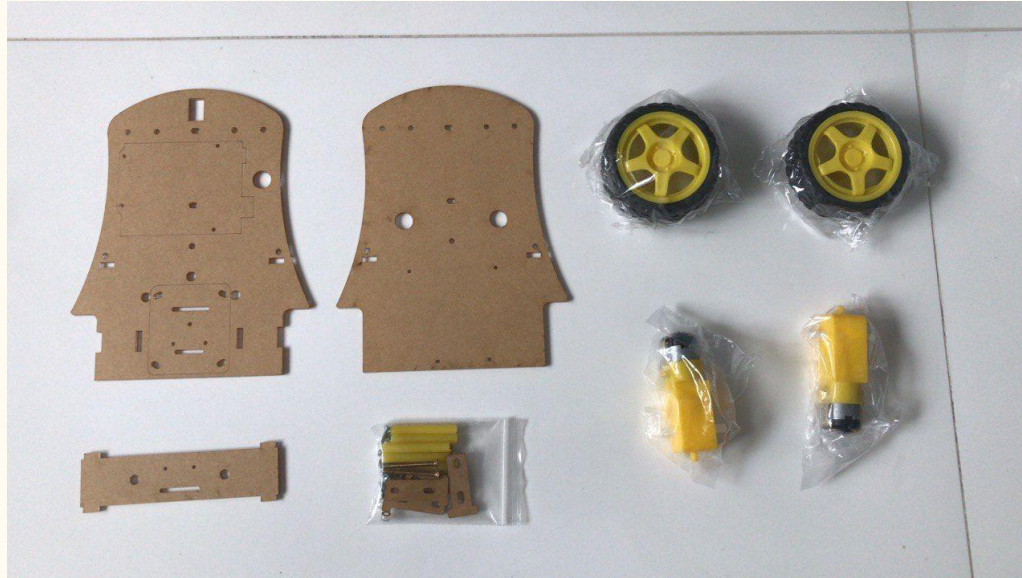
Balancing Robot

Gabriel Mendes, Jessé Alves and Matheus Villela

Physical Part of the Robot Finished!

Before:

Fig 1 - Chassis Parts

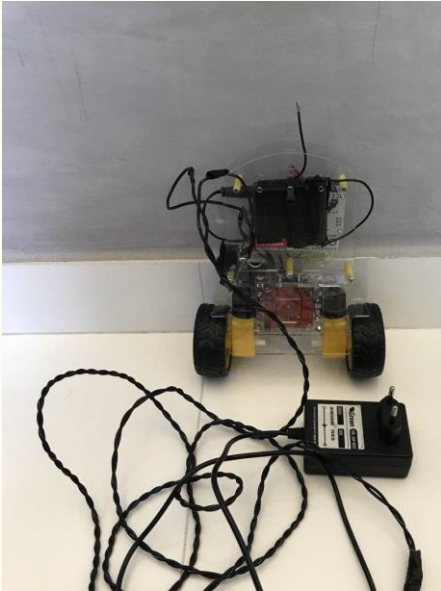


Source: Own

Total Robot Assembly Completed!

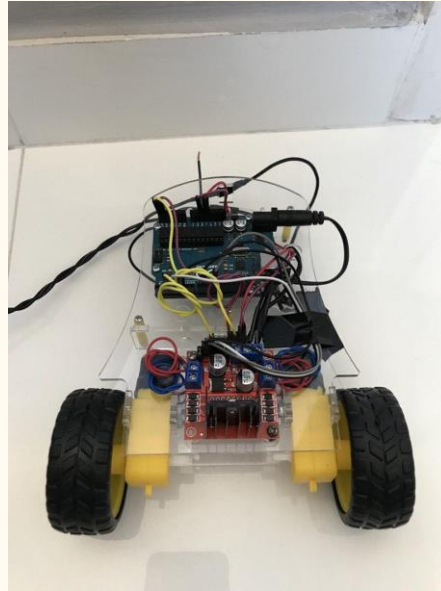
After:

Fig 2 - Battery module



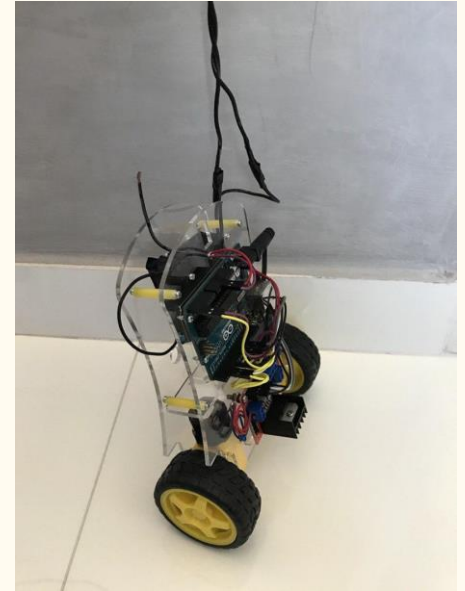
Source: Own

Fig 3 - Chassis Assembly



Source: Own

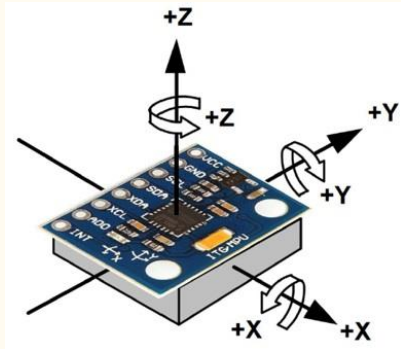
Fig 4 - Electrical connection



Source: Own

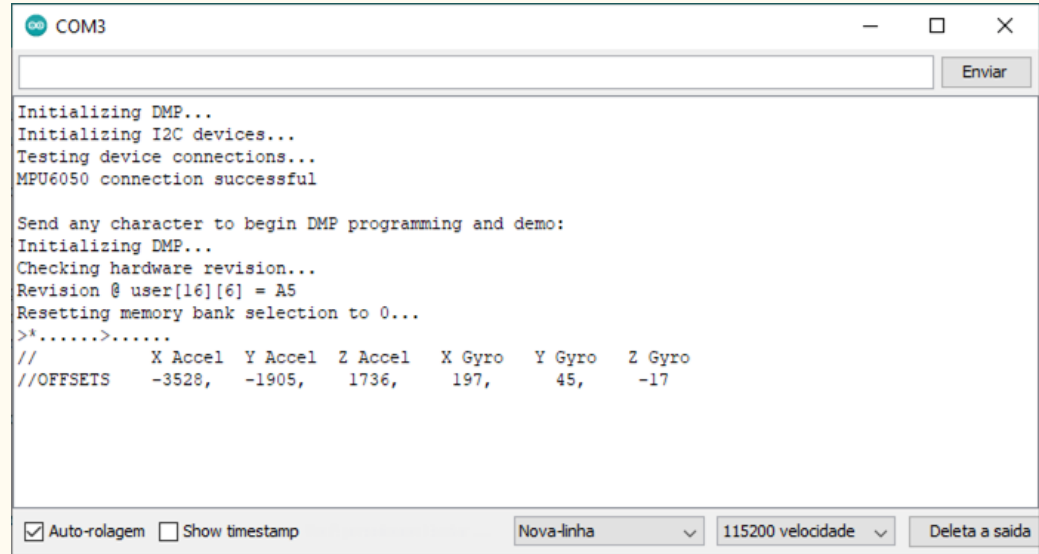
Communication With the MPU6050 Sensor

- I2C Communication
- Composed of an Accelerometer, Gyroscope and Temperature Sensor
- Element responsible for inserting feedback into the project
- Need for calibration



Source:

blog.arduinoomega.com

A screenshot of a serial monitor window titled 'COM3'. The window shows the following text:

```
Initializing DMP...
Initializing I2C devices...
Testing device connections...
MPU6050 connection successful

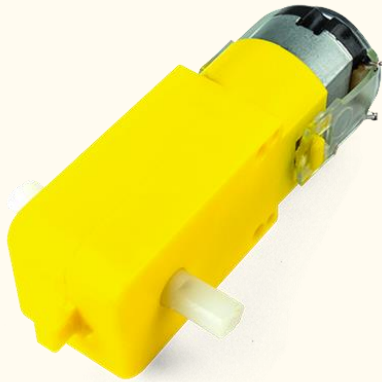
Send any character to begin DMP programming and demo:
Initializing DMP...
Checking hardware revision...
Revision @ user[16][6] = A5
Resetting memory bank selection to 0...
>^.....>.....
//          X Accel  Y Accel  Z Accel  X Gyro   Y Gyro   Z Gyro
//OFFSETS  -3528,   -1905,   1736,   197,    45,    -17
```

The bottom of the window has a status bar with checkboxes for 'Auto-rolagem' (checked) and 'Show timestamp' (unchecked). There are also dropdown menus for 'Nova-linha' and '115200 velocidade', and a 'Deleta a saída' button.

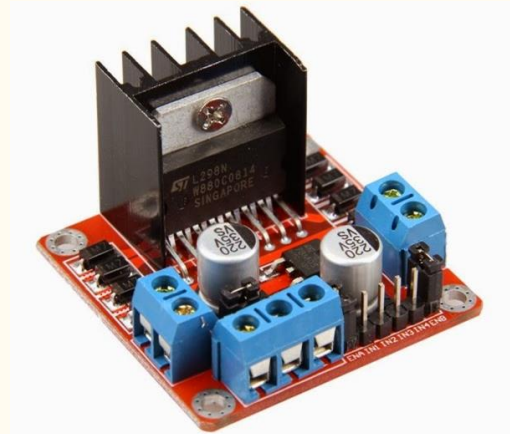
Source: Own

Activating Motors With the L298N Module

- Motor drive using PWM for speed adjustment
- Power supply through a battery module connected to the H Bridge, which will turn on the motors and, through a voltage regulator, turn on the Arduino.
- Use of motors with reduction gearboxes to maximize torque at the expense of speed



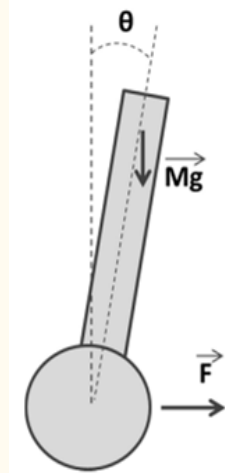
Source: RoboCore



Source: FilipeFlop

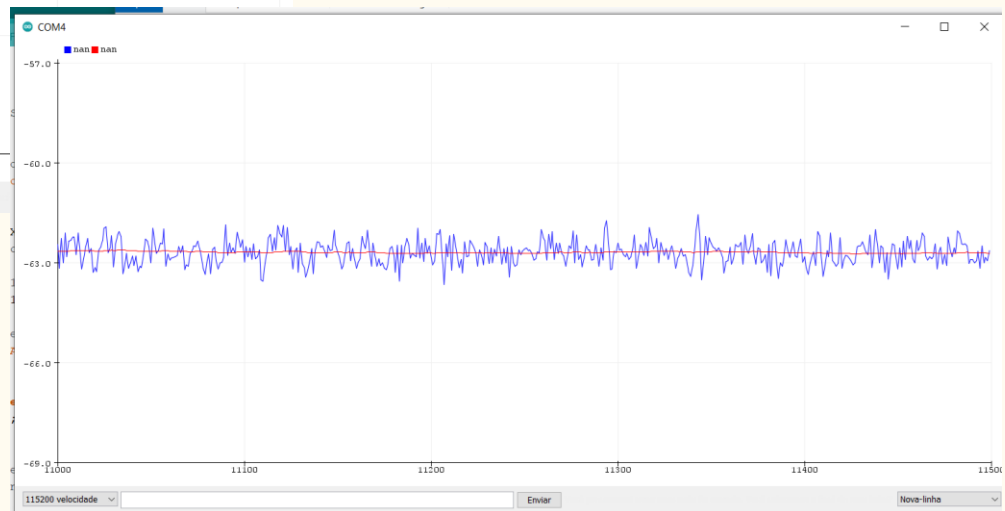
Sensor reading finished

- Testing each variable provided by the MPU6050 sensor: AccX, AccY, AccZ, GyroX, GyroY and GyroZ;
- Theta angle calculated and estimated using a Kalman Filter library on Arduino;



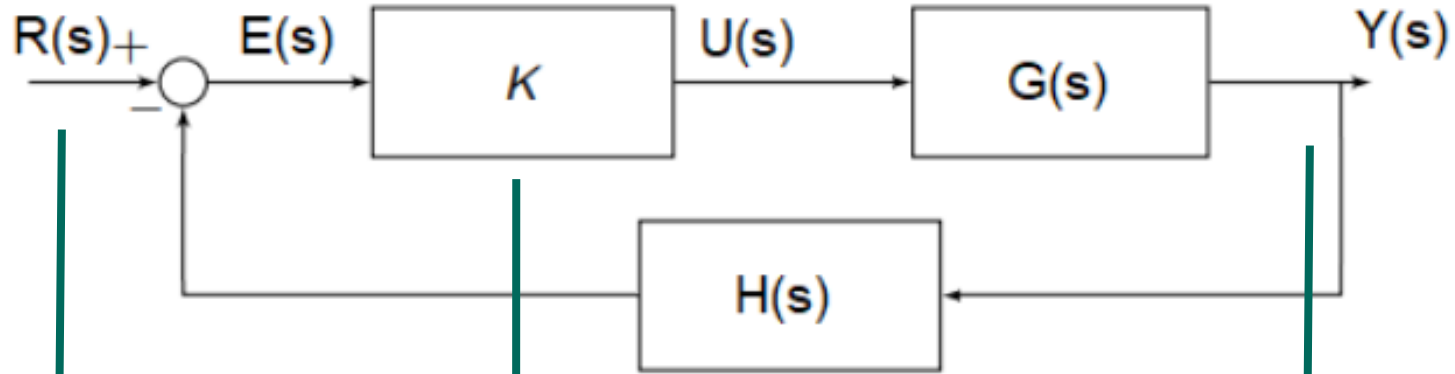
Source: Alex Pisciotta

Estimation Results



Control System Structure

Figura 11 - Sistema em Malha Fechada



Reference in 6°

discrete PID

MPU6050 sensor

Discrete PID control code implemented

$$u(t_k) = u(t_{k-1}) + \left(K_p + K_i \Delta t + \frac{K_d}{\Delta t} \right) e(t_k) + \left(-K_p - \frac{2K_d}{\Delta t} \right) e(t_{k-1}) + \frac{K_d}{\Delta t} e(t_{k-2}) \quad (24)$$

Source: Own

```
/*Calculo do Erro*/
e2 = e1;
e1 = e0;
e0 = SetPoint - input;

/* Calculo dos Termos do Controle Discreto*/
double A0 = kp + ki*deltaTime + kd/deltaTime;
double A1 = -kp - 2*kd/deltaTime;
double A2 = kd/deltaTime;

/* Calculo do PID Discreto*/
output = output + A0*e0 + A1*e1 + A2*e2;
```

Source: Own

```
const int OUTA = 3;
const int OUTB = 9;
const int OUTC = 5;
const int OUTD = 11;

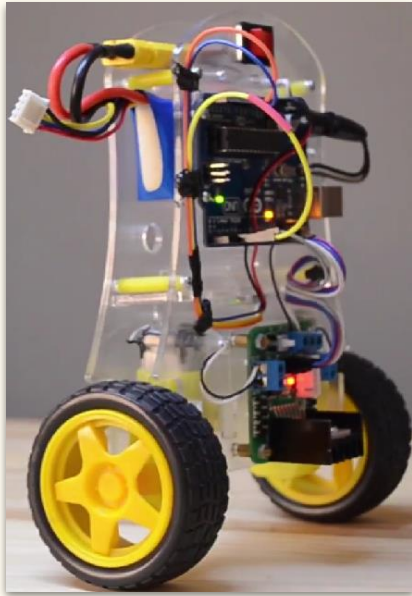
void int_motores(){
    pinMode(OUTA, OUTPUT);
    pinMode(OUTB, OUTPUT);
    pinMode(OUTC, OUTPUT);
    pinMode(OUTD, OUTPUT);
}

void PWMControleMotores(double comando){

    if(comando > 10){
        analogWrite(OUTA, 0); // Motor da direita p/ trás
        analogWrite(OUTB, abs(comando)); // Motor da direita p/ frente
        analogWrite(OUTC, 0); // Motor da esquerda p/ trás
        analogWrite(OUTD, abs(comando)); // Motor da esquerda p/ frente
    }else{
        analogWrite(OUTA, abs(comando)); // Motor da direita p/ trás
        analogWrite(OUTB, 0); // Motor da direita p/ frente
        analogWrite(OUTC, abs(comando)); // Motor da esquerda p/ trás
        analogWrite(OUTD, 0); // Motor da esquerda p/ frente
    }
}
```

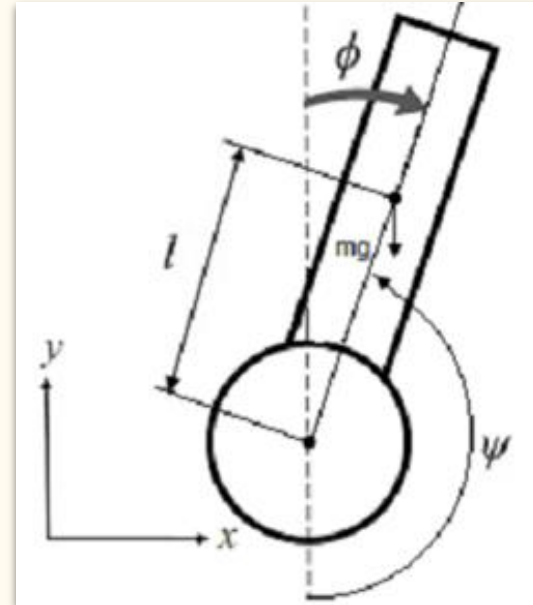
Robot Modeling

Fig 1 - Balancing Robot



Source: kpacitor.teachable.com

Fig 2 - Free Body Model



Source: Hanna Hellman & Henrik Sunnerman

Mathematical Model: Pendulum and Wheel

- From the free-body model of the pendulum in Figures 3 and 4, applying the equations of linear and angular motion, we will obtain the forces and the moment of inertia on the pendulum and from them we will arrive at the following equation:

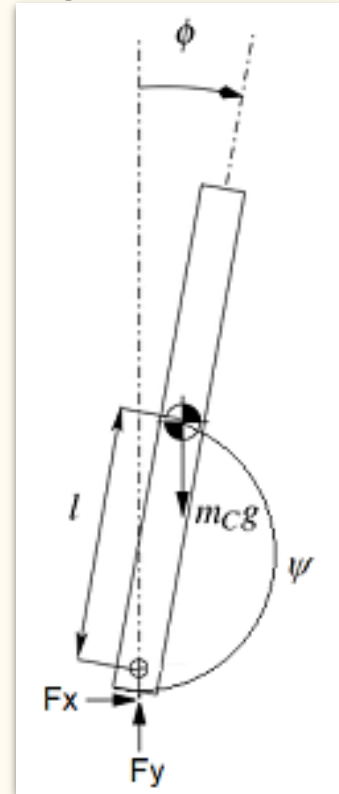
$$F_x = m_c * \frac{d^2x}{dt^2} + m_c * l * \frac{d^2\psi}{dt^2} * \cos(\psi) - m_c * l * \left(\frac{d\psi}{dt}\right)^2 * \sin(\psi) \quad (1)$$

$$F_y = m_c * g + m_c * l * \frac{d^2\psi}{dt^2} * \sin(\psi) + m_c * l * \left(\frac{d\psi}{dt}\right)^2 * \cos(\psi) \quad (\text{two})$$

$$J_c * \frac{d^2\psi}{dt^2} = -F_y * l * \sin(\psi) - F_x * l * \cos(\psi) \quad (3)$$

$$\psi = \pi + \phi \quad (4)$$

Fig 3 - Pendulum



Mathematical Model: Pendulum and Wheel

- From the free-body model of the pendulum in Figures 3 and 4, applying the equations of linear and angular motion, we will obtain the forces and the moment of inertia on the pendulum and from them we will arrive at the following equation:

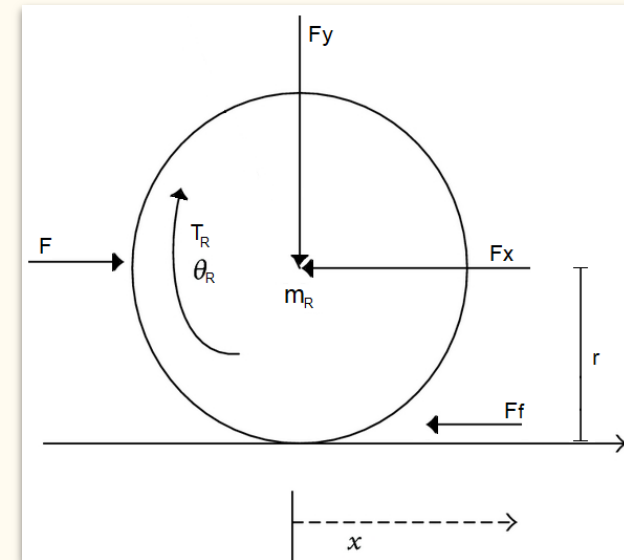
$$m_R * \frac{d^2x}{dt^2} = F - F_x - F_f \quad (1)$$

$$J_R * \frac{d^2\theta_R}{dt^2} = T_R + F_f * r \quad (\text{two})$$

$$T_R = n * T_a \quad (3)$$

$$\theta_R = \frac{\theta}{n} \quad (4)$$

Fig 4 - Wheel



Mathematical Model: DC Motor

- From the free-body model of the pendulum in Figure 3, applying the equations of linear and angular motion, we will obtain the forces and the moment of inertia on the pendulum and from them we will arrive at the following equation:

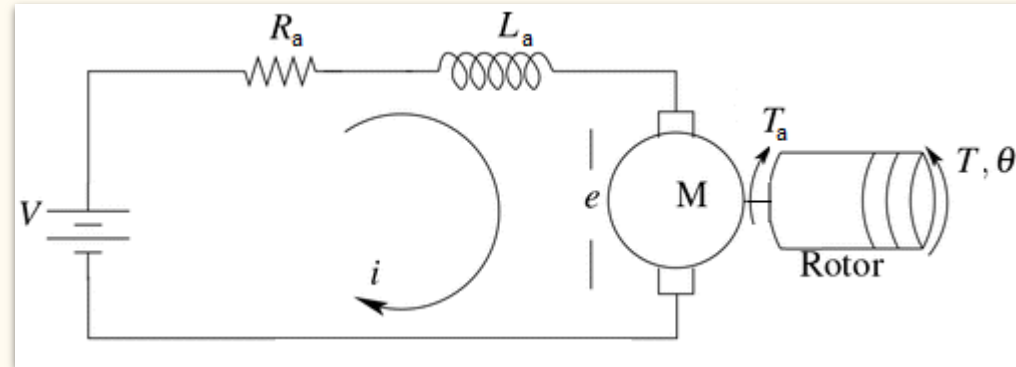
$$V - K * \frac{d\theta}{dt} = R_a * i + L_a * \frac{di}{dt}$$

(1)

$$J * \frac{d^2\theta}{dt^2} = K * i - T_a$$

(two
)

Fig 5 - DC Motor Circuit



Source: <https://journals.sagepub.com/>

Model Linearization

- To linearize the model, the following considerations were made:

$$\left\{ \begin{array}{l} \phi \approx 0 \\ \psi = \pi + \phi \\ \sin(\psi) = \sin(\pi + \phi) \approx -\phi \\ \cos(\psi) = \cos(\pi + \phi) \approx -1 \\ \phi^2 \approx 0 \\ \phi * \frac{d^2\phi}{dt^2} \approx 0 \\ (\frac{d\phi}{dt})^2 \approx 0 \end{array} \right.$$

(1)

(two
)

(3)

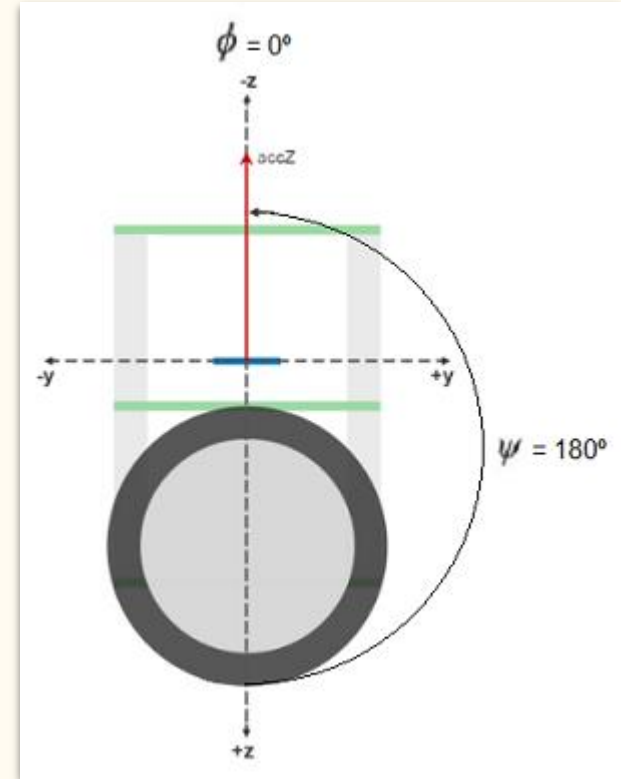
(4)

(5)

(6)

(7)

Fig 6 - Robot Angles



Source:

<https://www.instructables.com/>

State Space Model

- With the equations presented previously and linearization applied, we defined the following model:

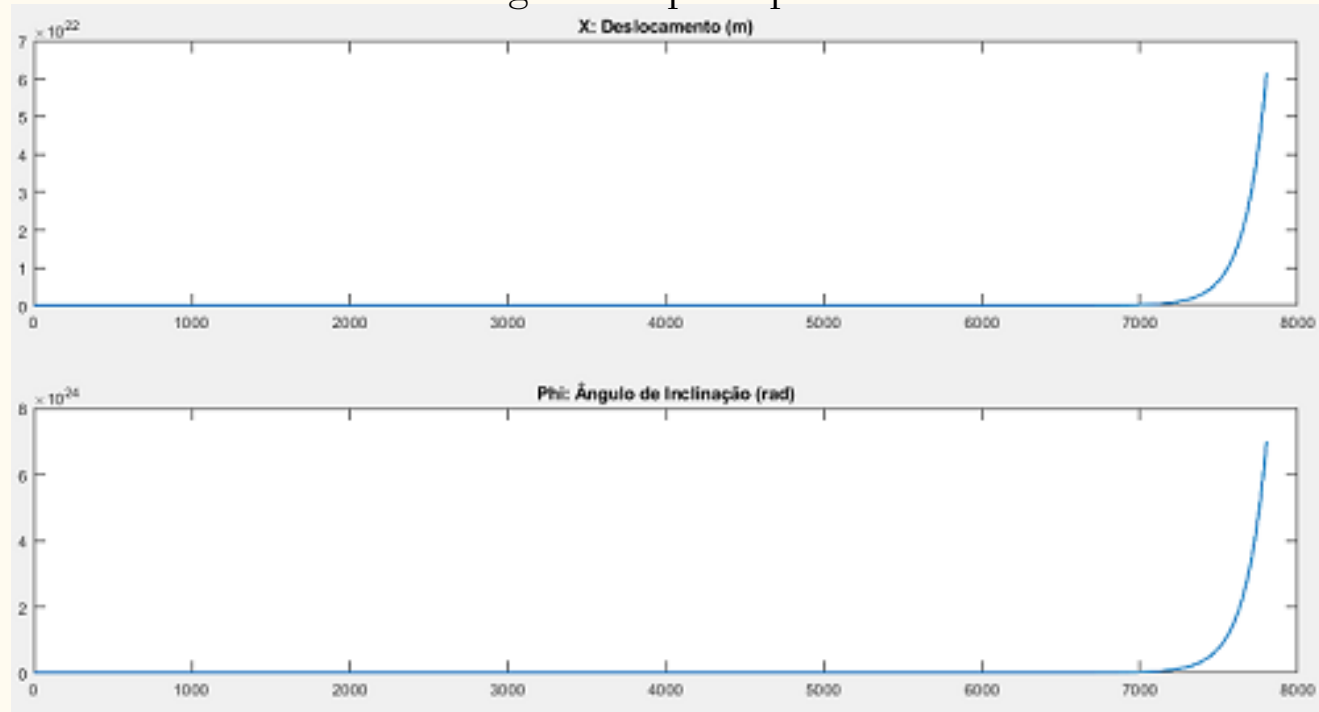
$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} \approx \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{(J_c + m_c * l^2) * n^2 * K^2}{R_a * q} & \frac{n * r^2 * m_c^2 * g * l^2}{q} & 0 \\ 0 & 1 & 0 & 1 \\ 0 & -\frac{n^2 * K^2 * m_c * l}{R_a * q} & \frac{m_c * g * l * p}{q} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{(J_c + m_c * l^2) * n^2 * r * K}{R_a * q} \\ 0 \\ \frac{n^2 * r * K * m_c * l}{R_a * q} \end{bmatrix} [V]$$

$$[y] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix}$$

Open Loop System Step Response

- Applying a voltage step of 1 V to the system.

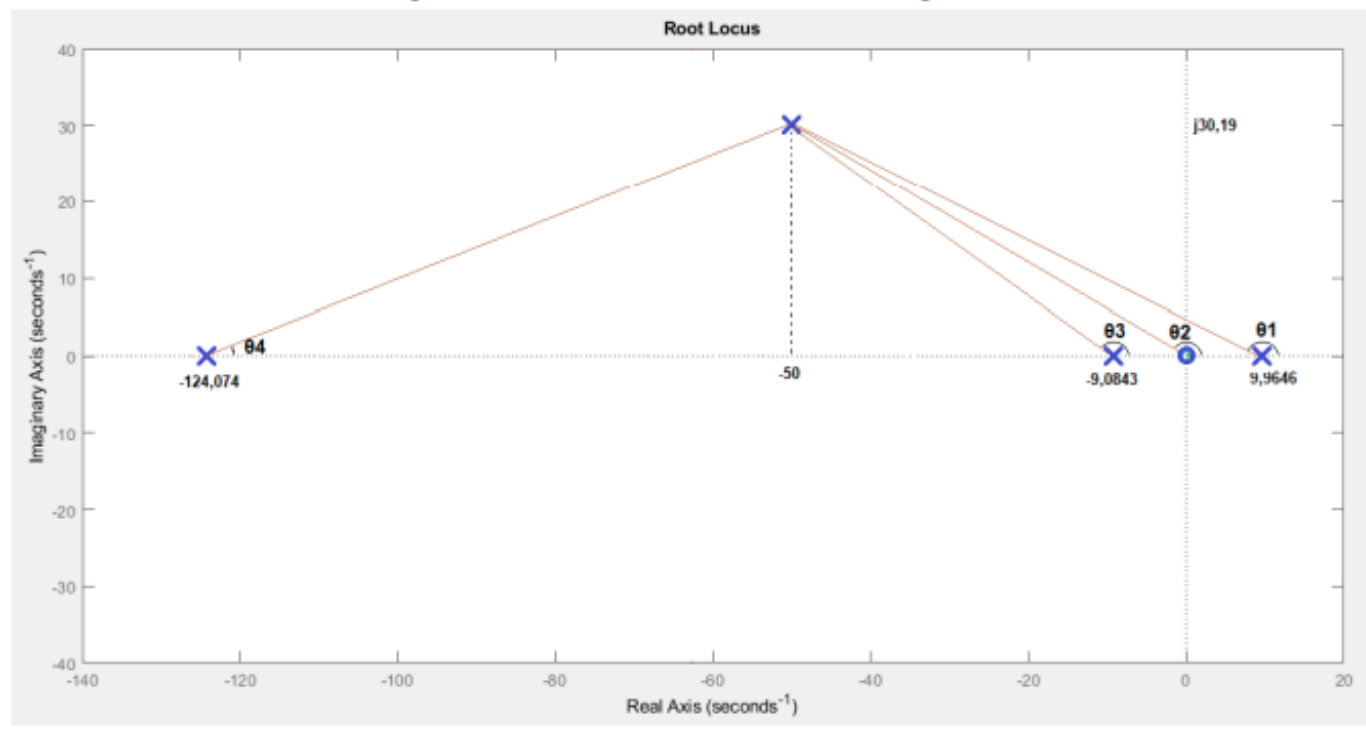
Fig 7 - Step Response



Source: Own Authorship

PID project

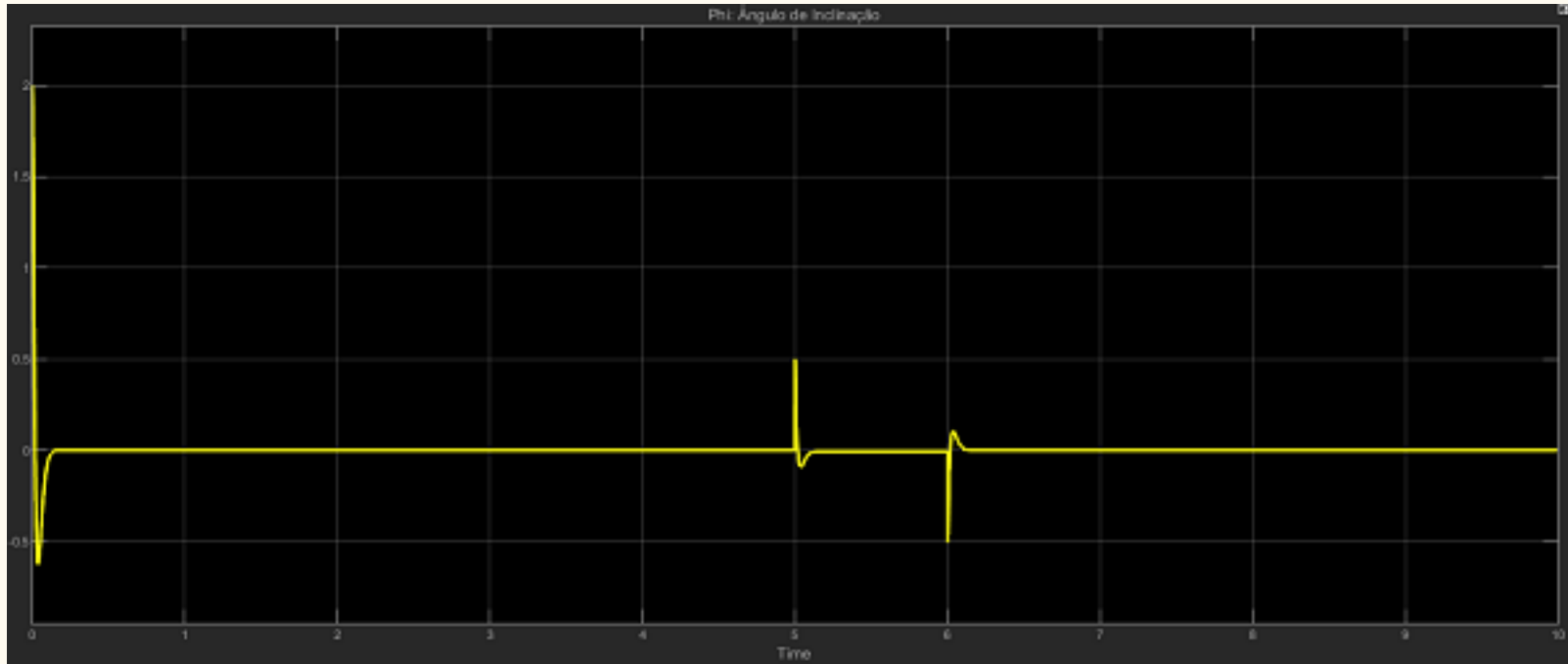
PID project by root location



Applying a Discrete PID Controller

- PID controller with $K_p = 18.43$, $K_i = 533.40$, $K_d = 0.1451$ and $\Delta t = 3$ ms.

Fig 8 - Closed Loop System



Source: Own Authorship

Practical Application

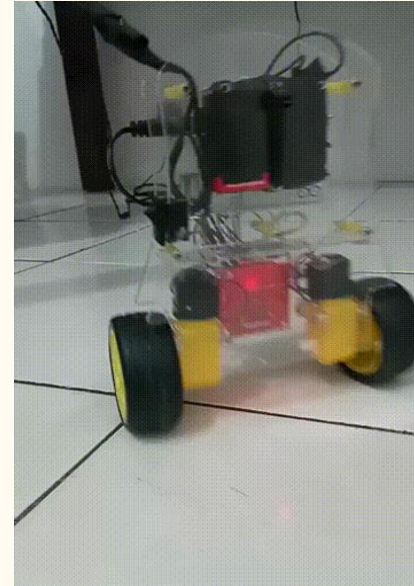
- Result obtained with the application of the PID Controller designed with the help of Matlab's PID Tuner.

Fig 9 - Side View



Source: Own Authorship

Fig 10 - Front View



Source: Own Authorship

Questions and suggestions



Thank you for your
attention