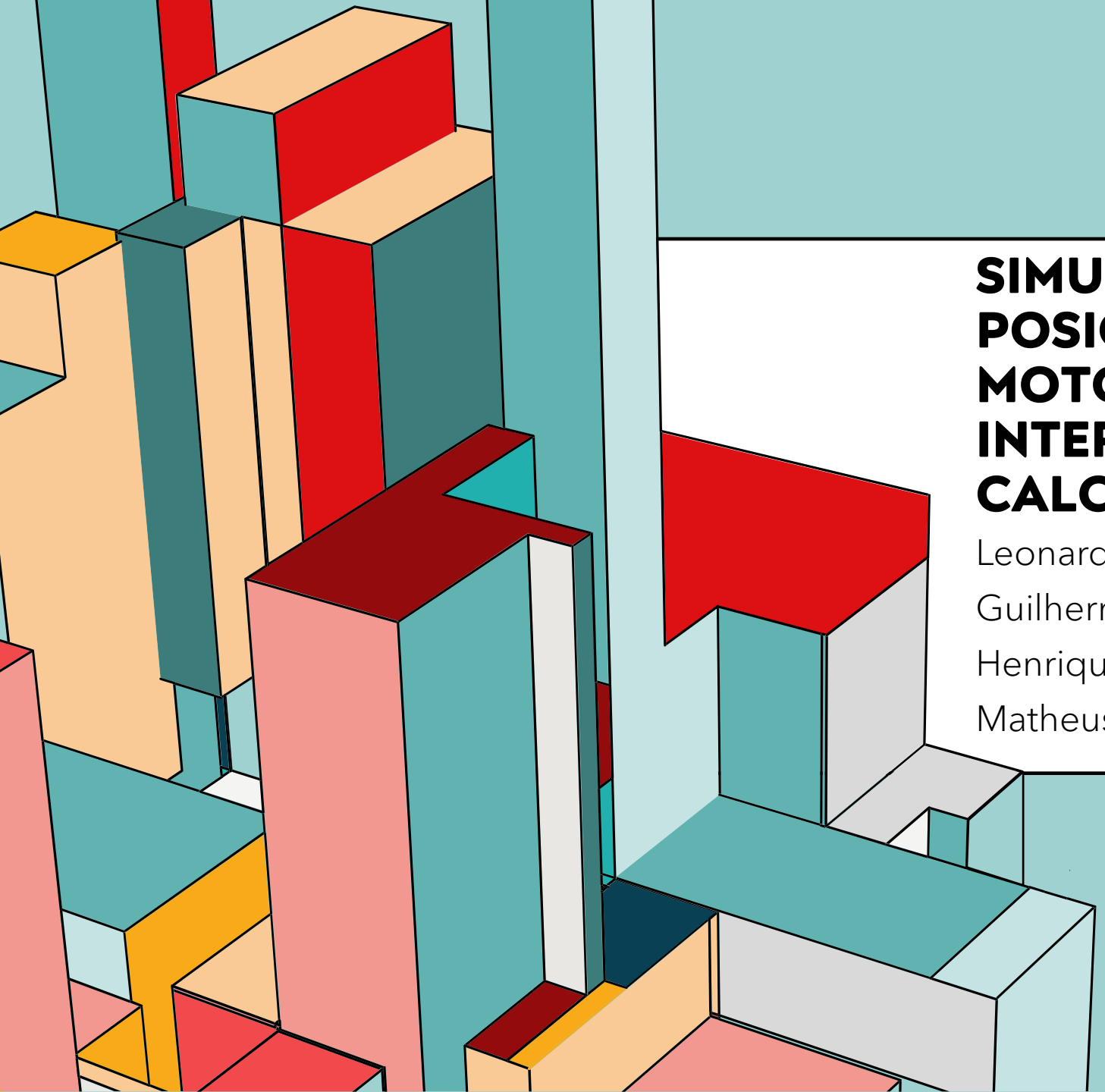


SIMULAÇÃO DE CONTROLE DE POSIÇÃO E VELOCIDADE DE UM MOTOR DC COM PARÂMETROS INTERATIVOS E PRÉ-CALCULADOS

Leonardo Oneda Galvani

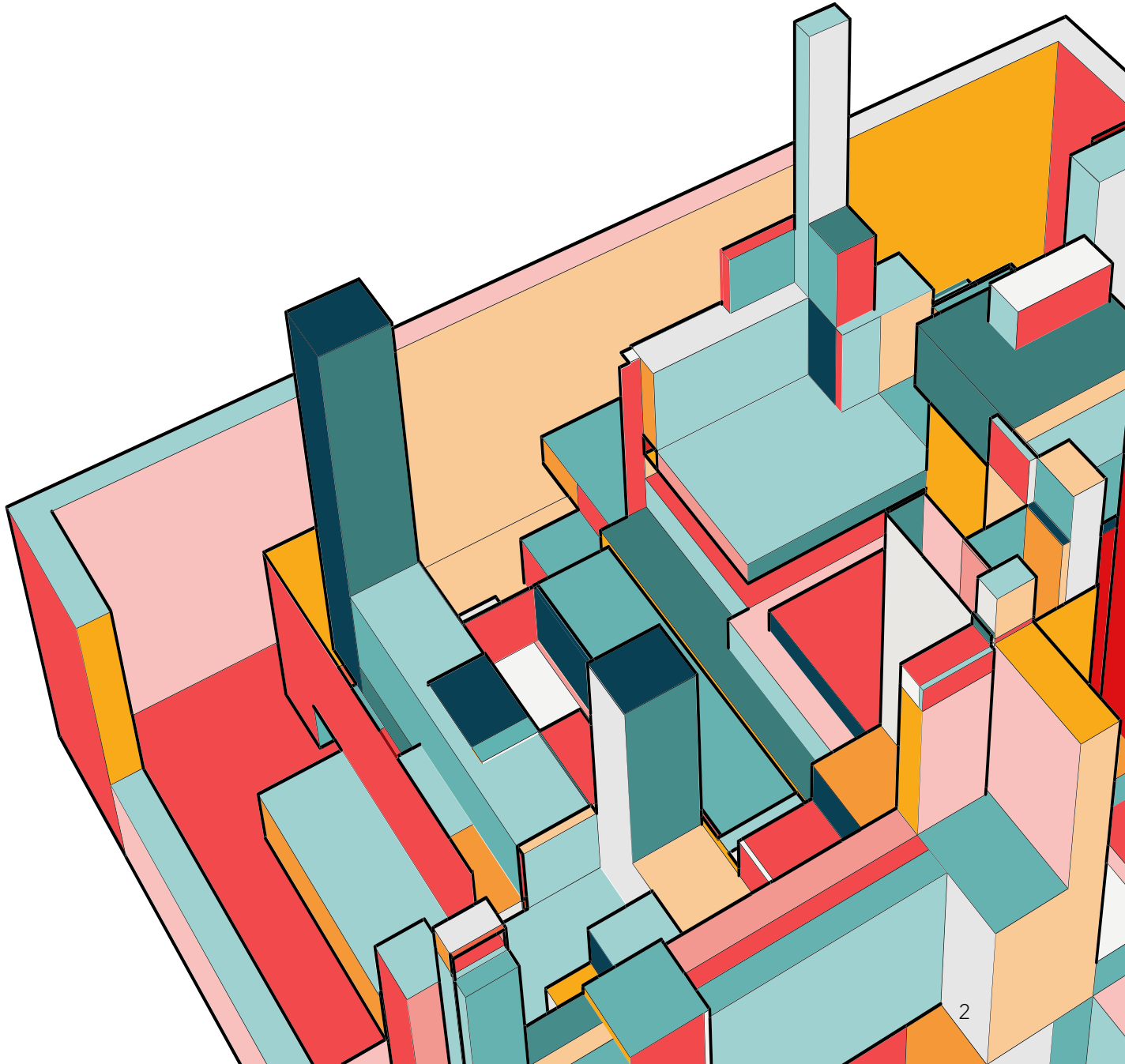
Guilherme Nami Bortolozi

Henrique Fortuna Accorinti

Matheus Ferreira Palú

A IDEIA

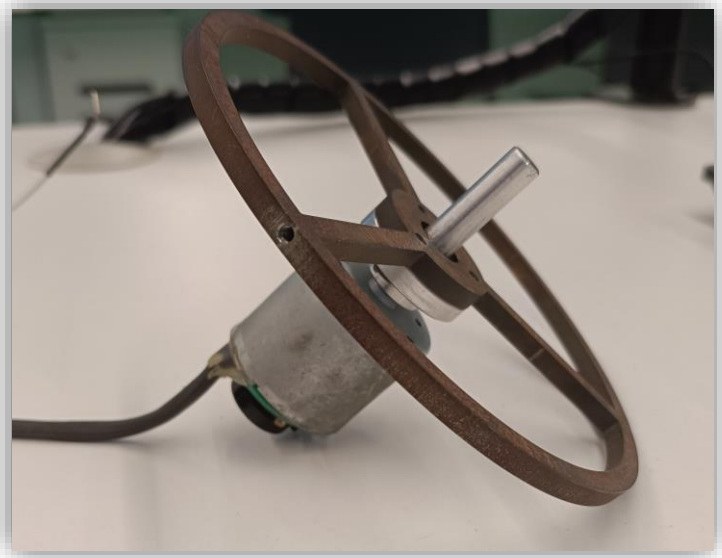
- A ideia principal é desenvolver um ambiente de simulação de malhas de controle fechadas diferentes no mesmo dispositivo e poder comparar e entender os efeitos de acordo com os valores introduzidos.
- Substituir software com licenças, como LabView ou Matlab.
- Analisar o desempenho com controladores projetados para a planta.

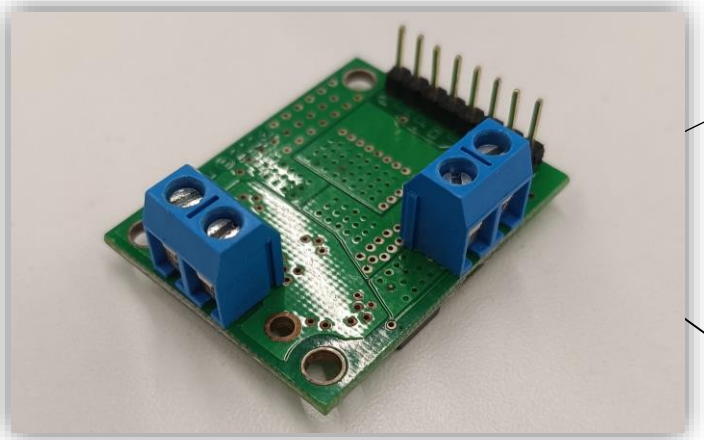


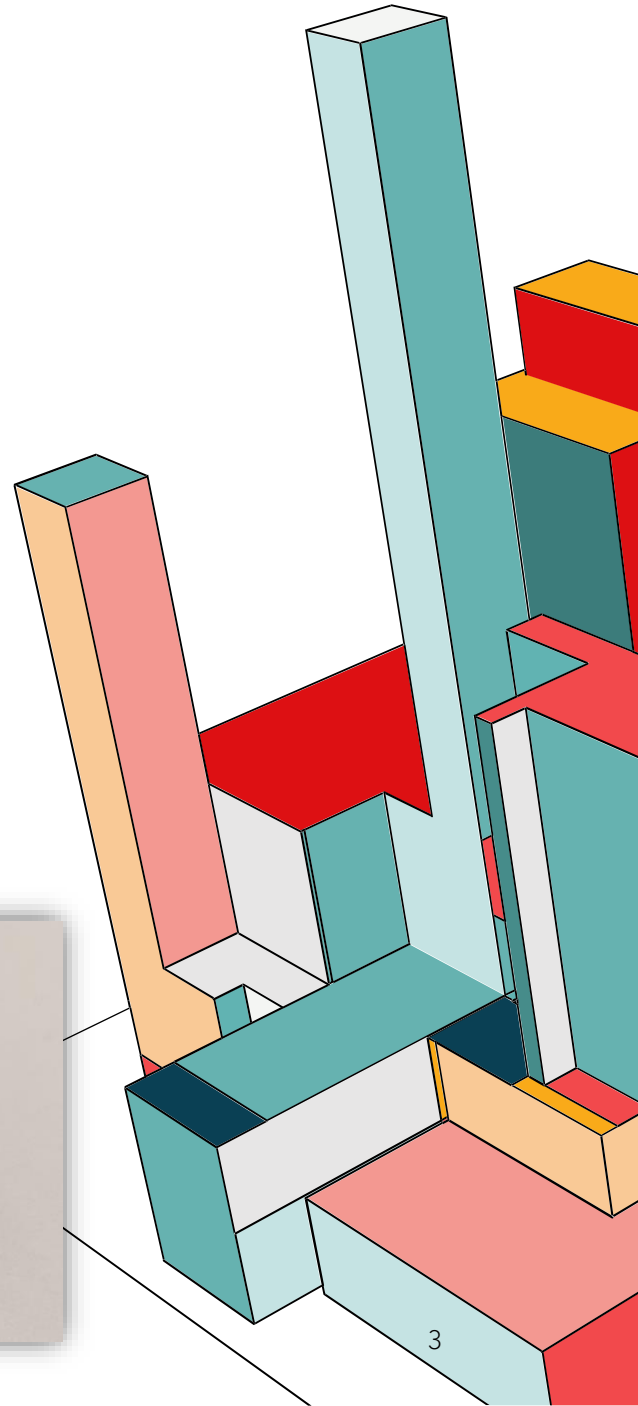
HARDWARE

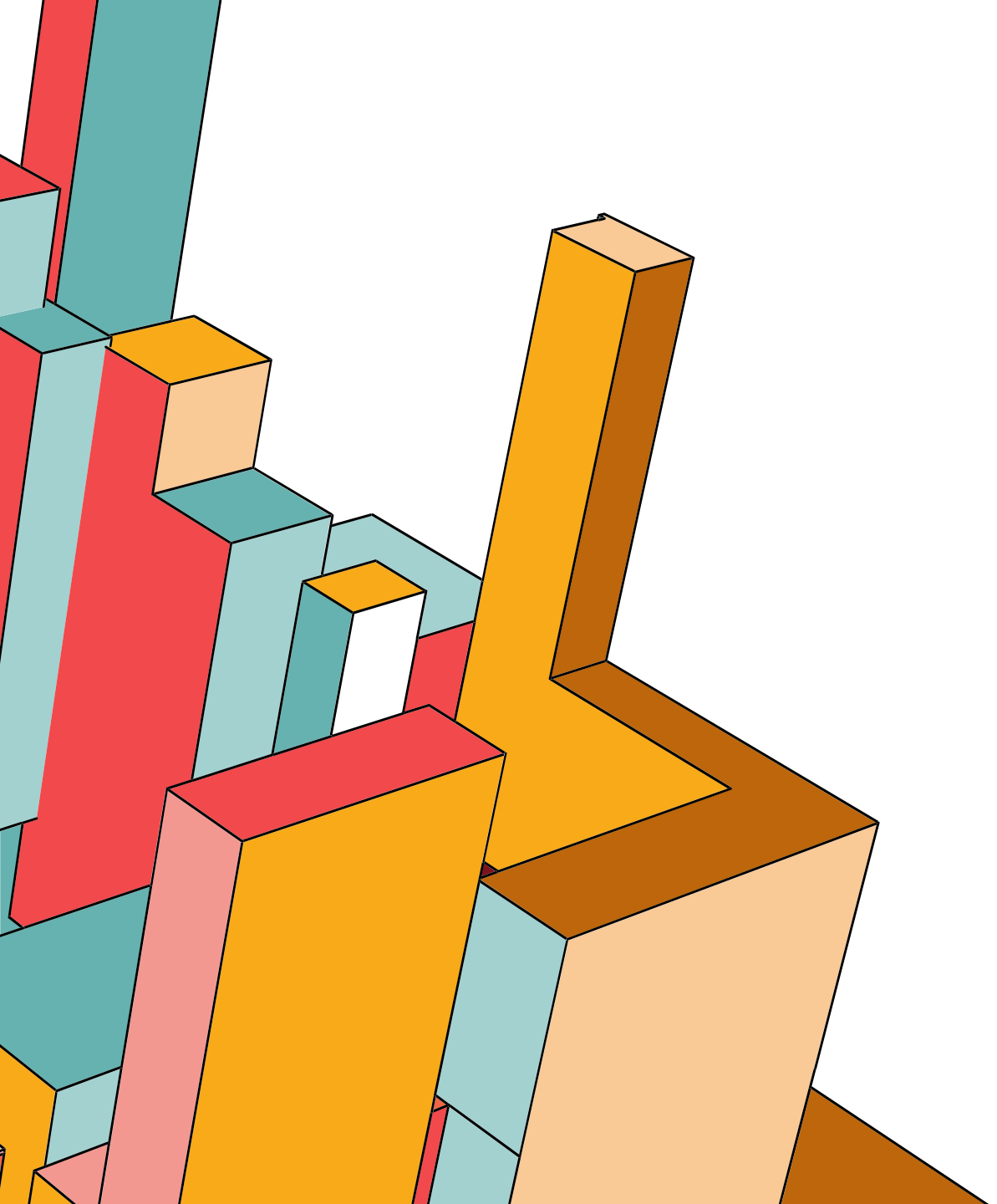
Sistema a ser controlado

- A interface terá como função controlar posição e velocidade de um motor DC, acoplado a uma roda de inércia. Para isso será necessário os seguintes componentes:
  - Motor DC
  - Encoder
  - Ponte H
  - microcontrolador
  - Comunicação Serial (Via cabo)







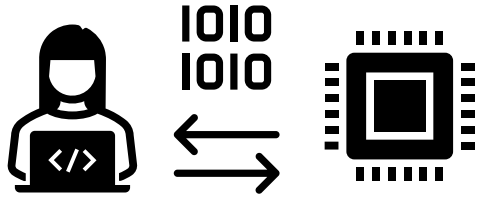


SOFTWARE

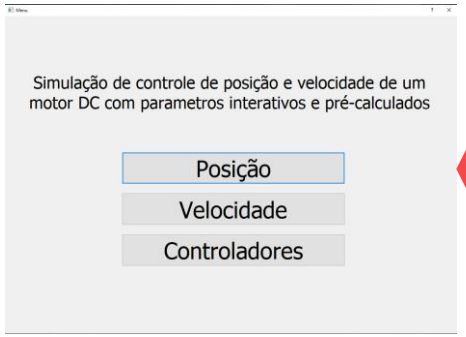
Base

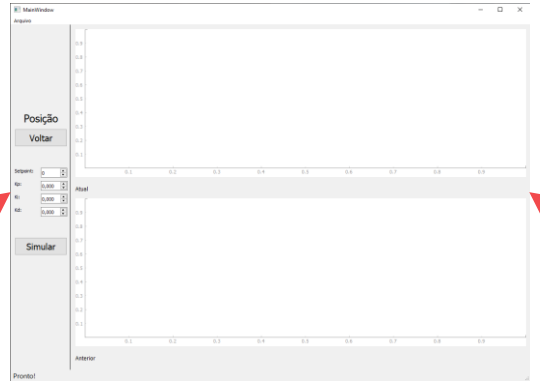
- O Software será separado em duas principais partes
  - Python (Orientado a objeto e interface gráfica)
  - C (Controle de posição ou velocidade)

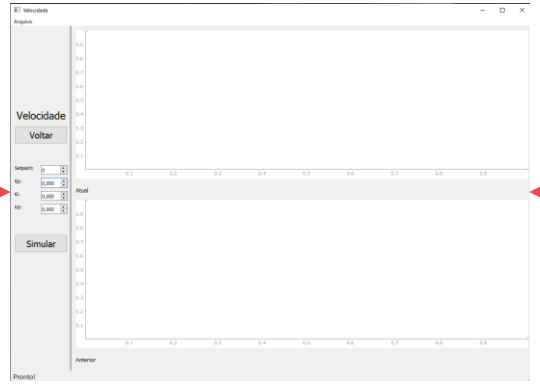
- A comunicação entre as duas partes será realizado com a comunicação serial.

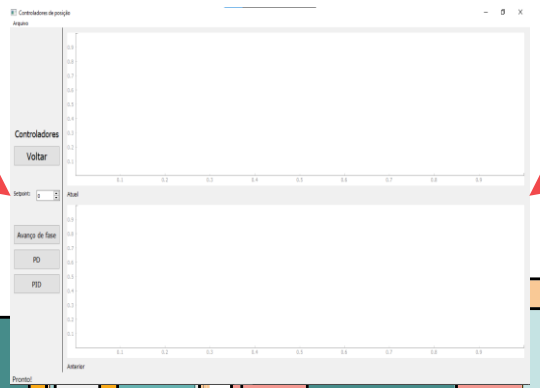


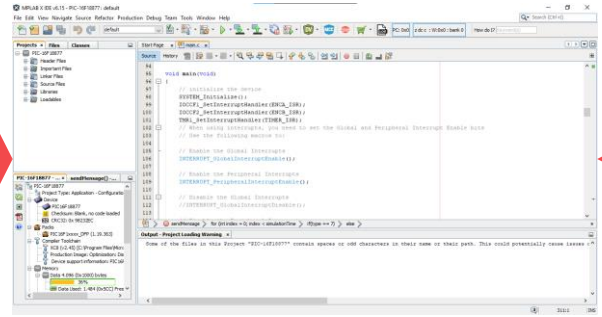
ESTRUTURA

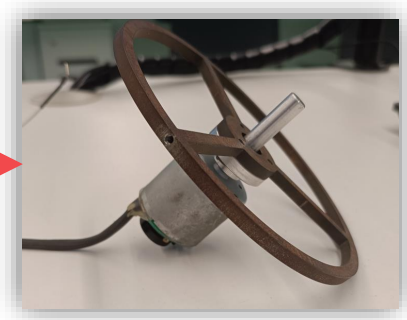

Simulação de controle de posição e velocidade de um motor DC com parametros interativos e pré-calculados
Posição
Velocidade
Controladores


Posição
Voltar
Simular


Velocidade
Voltar
Simular

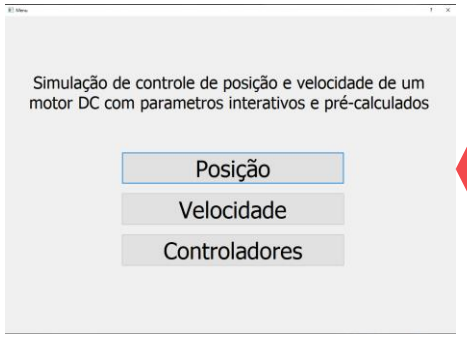

Controladores
Voltar
Avança de fase
PD
PID

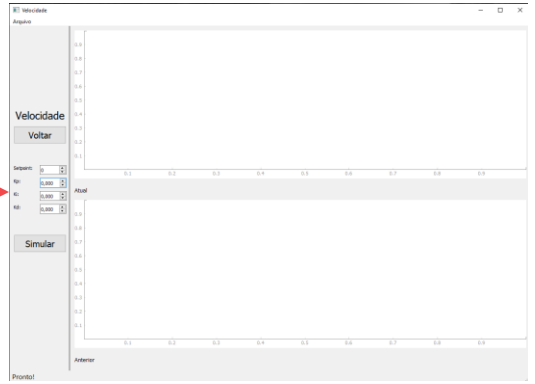


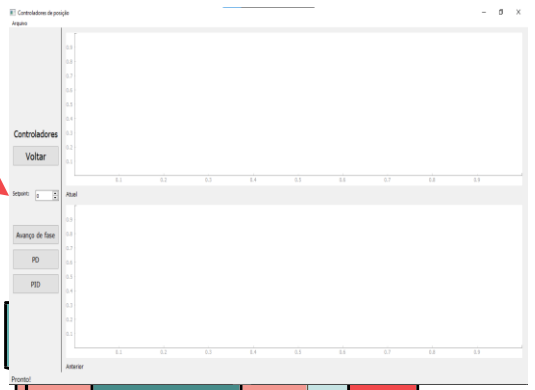


ESTRUTURA

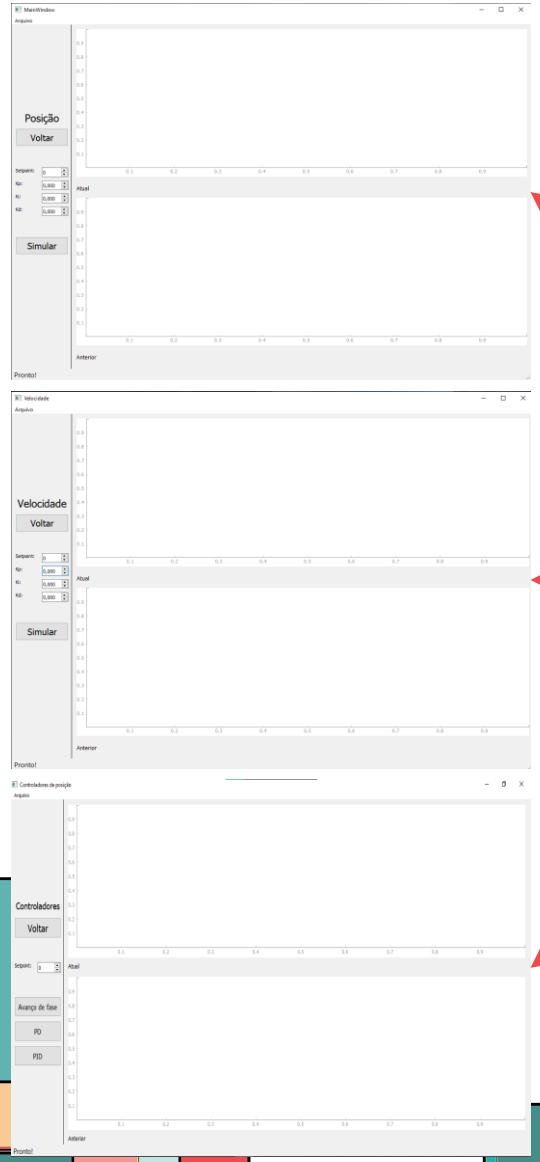
- Orientação a objeto
- Características importantes:
  - Botões para seleção de modo (Posição ou Velocidade)
  - Botões para voltar
  - Parâmetros:
    - Kp
    - Ki
    - Kd
    - SetPoint
  - Simular (pacote de informação via Serial)

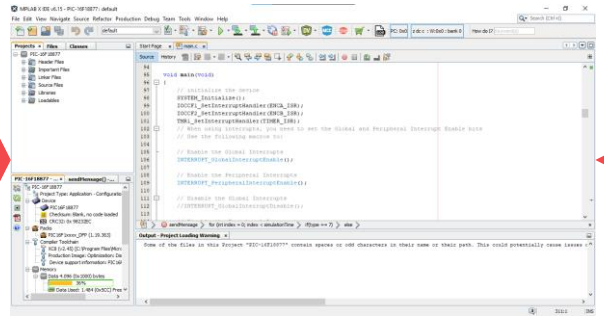


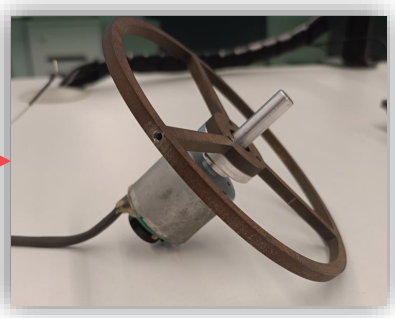




ESTRUTURA

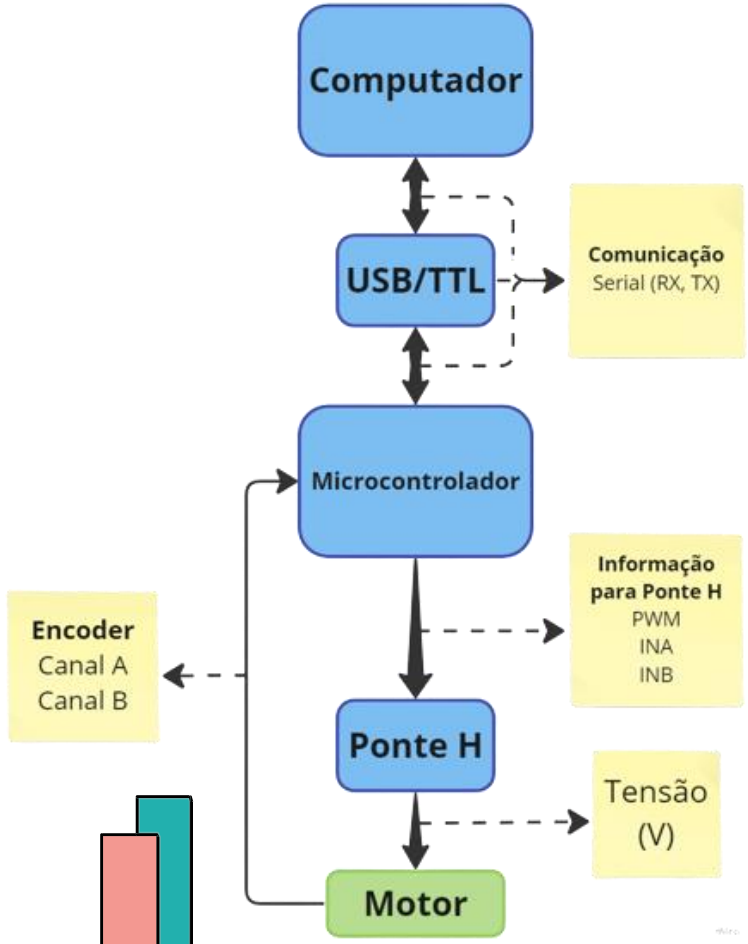






- Leitura do pacote
- Implementação da malha de controle
- Leitura simultânea dos dados
- Interpretação dos dados
- Envio via serial

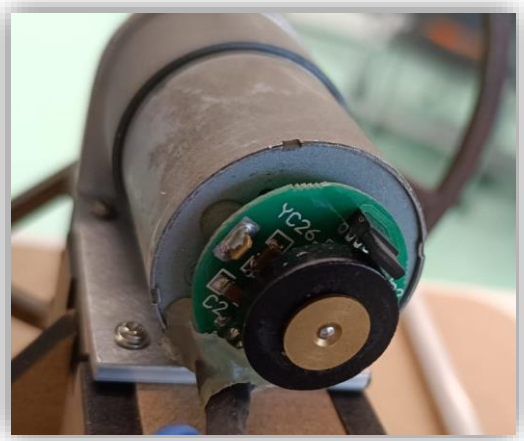
ESTRUTURA

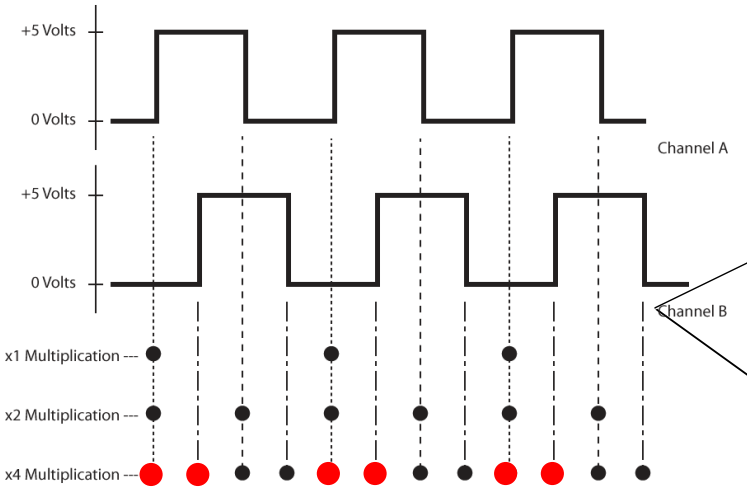

Computador
USB/TTL
Comunicação
Serial (RX, TX)
Microcontrolador
Encoder
Canal A
Canal B
Informação
para Ponte H
PWM
INA
INB
Ponte H
Tensão
(V)
Motor

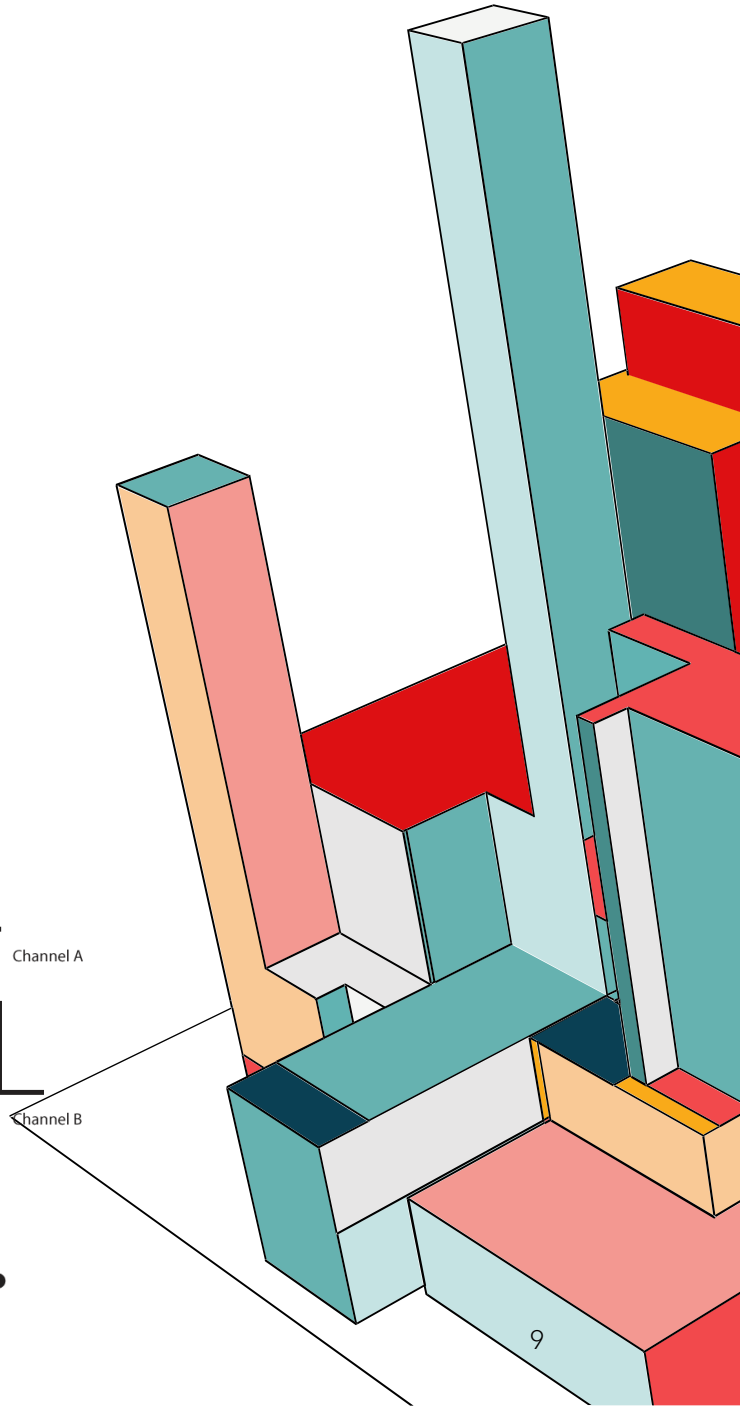
INSTRUMENTAÇÃO

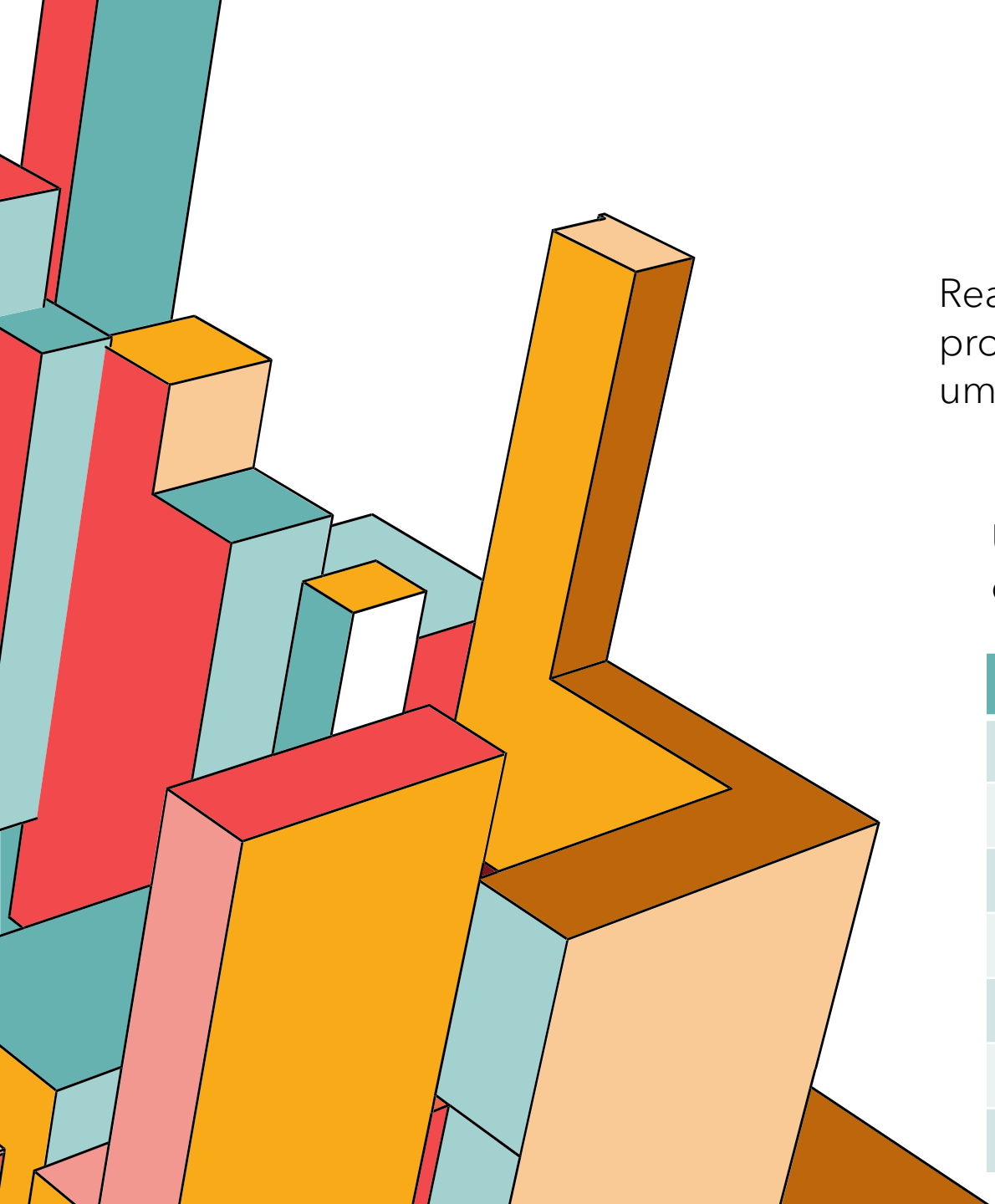
Sensor usado

- Para medir a posição e a velocidade do motor foi utilizado o mesmo sensor, um encoder de efeito hall.
- O encoder consiste em dois sensores de efeito hall estrategicamente posicionado que são acionado quando algum imã se aproxima, a qual este está posicionado no disco preto.
- Cada sensor define um canal, portanto no encoder existe canal A e canal B
- Por conta do uso de dois sensores é possível adquirir o sentido de rotação do motor, por meio da sua quadratura.









MICROCONTROLADOR

Realização da prototipação utilizando um arduino nano

- Pouca memória SRAM
- Não atende os requisitos

Utilizar o microcontrolador PIC com os periféricos necessários

Periférico	Característica
Timers	2x
PWM	1x
UART	1x
IO	3x
Int. Ext.	2x
Compatibilidade	MCC
Memória	>2KB

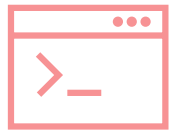
PIC16F1619 ✗

PIC16F18877 ✓

MÓDULOS

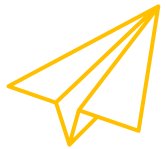
Receber dados

Leitura do pacote enviado pelo simulador pela UART



Controlar

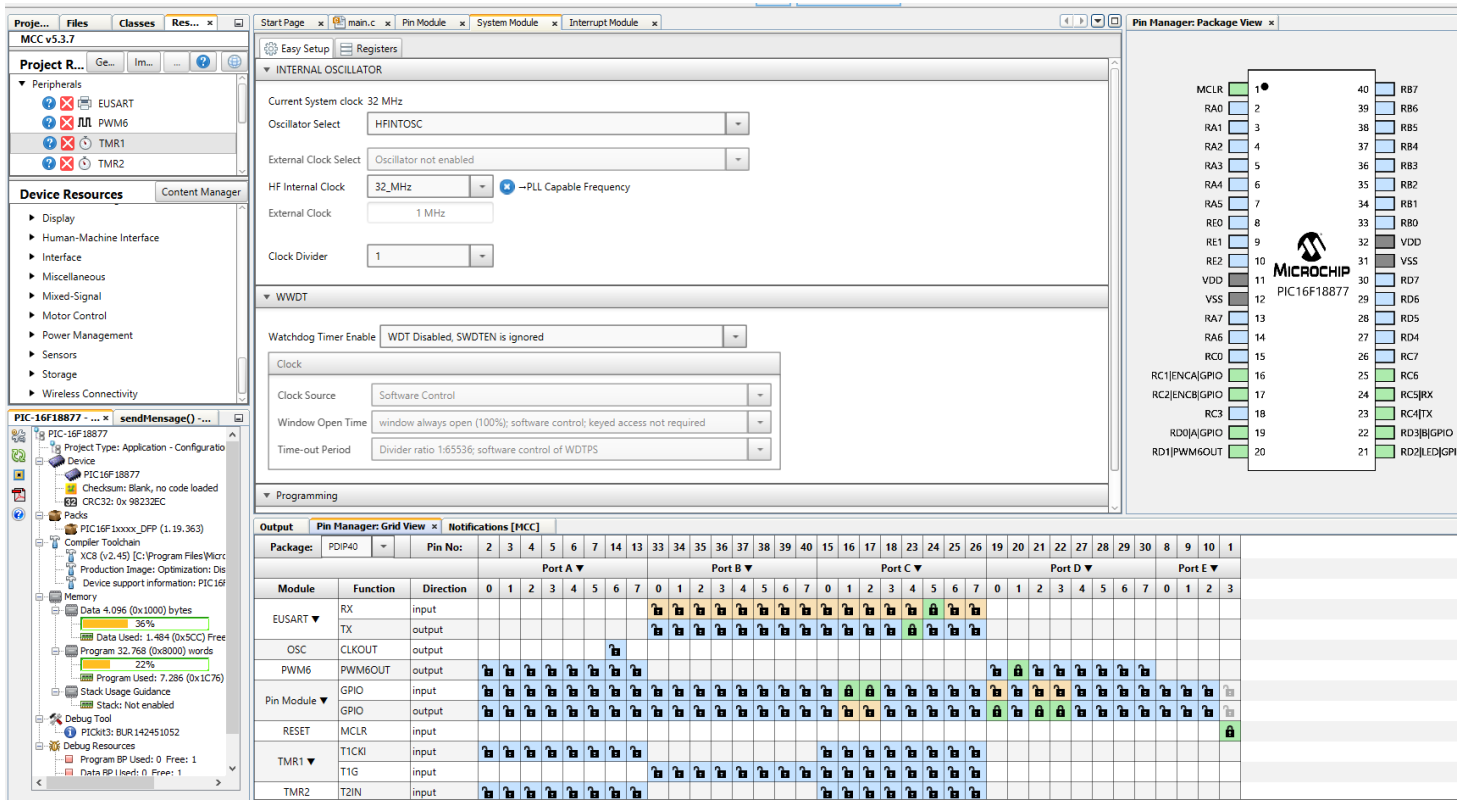
Controle da planta usando os dados, timers, PWM, IOs e a memória



Enviar dados

Envia pela UART todos os dados neccessários.

MCC

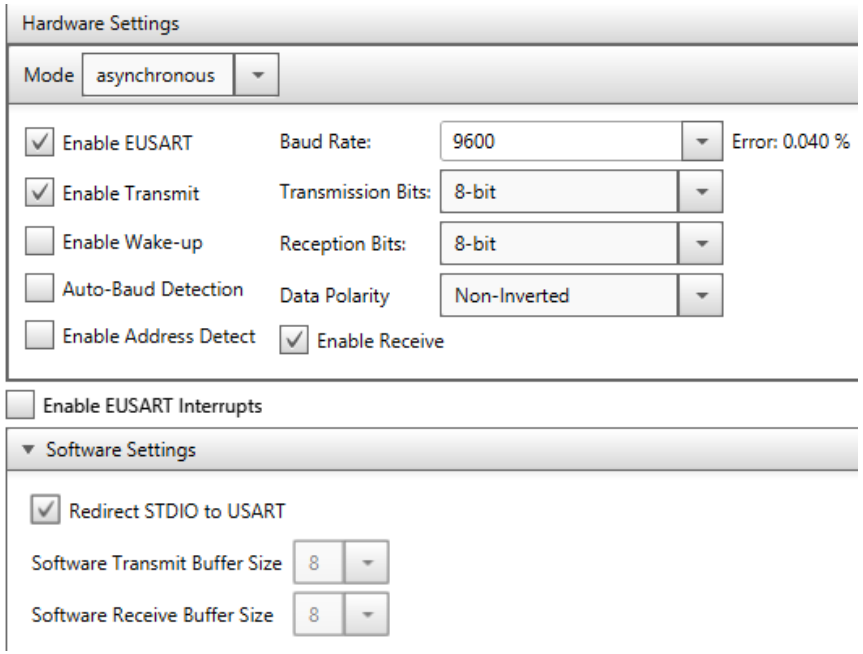


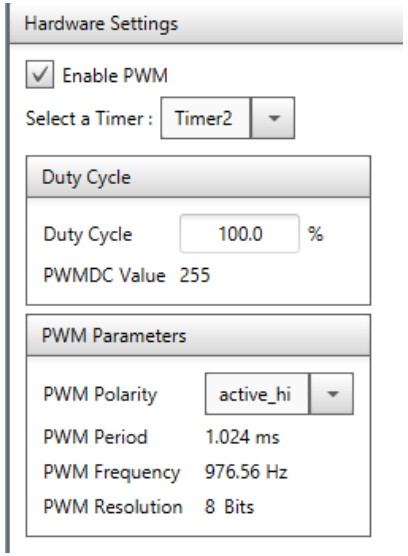
Easy Setup | Registers

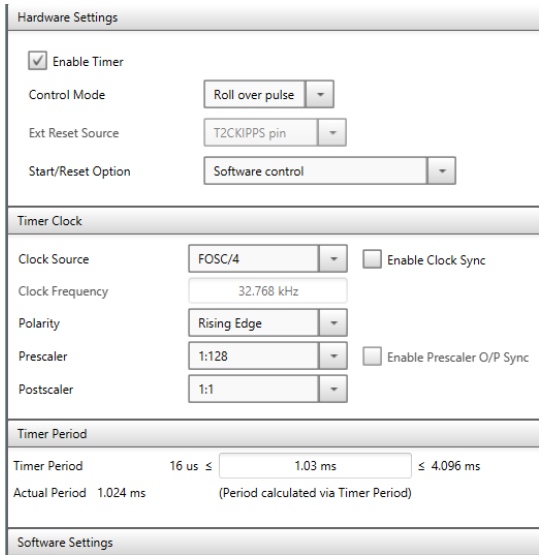
Selected Package : PDIP40

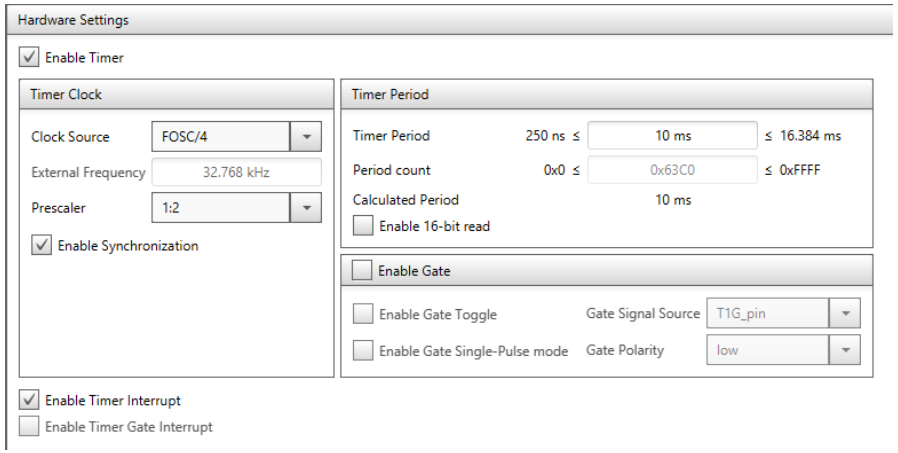
Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	OD	IOC
RC1	Pin Module	GPIO	ENCA	☐	☐	☐	☑	☐	positive
RC2	Pin Module	GPIO	ENCB	☐	☐	☐	☑	☐	positive
RC4	EUSART	TX		☐	☑	☑	☐	☐	none
RC5	EUSART	RX		☐	☐	☐	☐	☐	none
RD0	Pin Module	GPIO	A	☐	☐	☑	☐	☐	
RD1	PWM6	PWM6OUT		☐	☐	☑	☐	☐	
RD2	Pin Module	GPIO	LED	☐	☐	☑	☐	☐	
RD3	Pin Module	GPIO	B	☑	☐	☑	☐	☐	

MCC


Hardware Settings
Mode asynchronous
Enable EUSART
Baud Rate: 9600
Error: 0.040 %
Enable Transmit
Transmission Bits: 8-bit
Enable Wake-up
Reception Bits: 8-bit
Auto-Baud Detection
Data Polarity Non-Inverted
Enable Address Detect
Enable Receive
Enable EUSART Interrupts
Software Settings
Redirect STDIO to USART
Software Transmit Buffer Size 8
Software Receive Buffer Size 8


Hardware Settings
Enable PWM
Select a Timer : Timer2
Duty Cycle
Duty Cycle 100.0 %
PWMDC Value 255
PWM Parameters
PWM Polarity active_hi
PWM Period 1.024 ms
PWM Frequency 976.56 Hz
PWM Resolution 8 Bits


Hardware Settings
Enable Timer
Control Mode Roll over pulse
Ext Reset Source T2CKIPPS pin
Start/Reset Option Software control
Timer Clock
Clock Source FOSC/4
Enable Clock Sync
Clock Frequency 32.768 kHz
Polarity Rising Edge
Prescaler 1:128
Enable Prescaler O/P Sync
Postscaler 1:1
Timer Period
Timer Period 16 us ≤ 1.03 ms ≤ 4.096 ms
Actual Period 1.024 ms (Period calculated via Timer Period)
Software Settings


Hardware Settings
Enable Timer
Timer Clock
Clock Source FOSC/4
External Frequency 32.768 kHz
Prescaler 1:2
Enable Synchronization
Timer Period
Timer Period 250 ns ≤ 10 ms ≤ 16.384 ms
Period count 0x0 ≤ 0x63C0 ≤ 0xFFFF
Calculated Period 10 ms
Enable 16-bit read
Enable Gate
Enable Gate Toggle
Gate Signal Source T1G_pin
Enable Gate Single-Pulse mode
Gate Polarity low
Enable Timer Interrupt
Enable Timer Gate Interrupt

RECEBER DADOS

Leitura e interpretação

- Leitura da UART até o caracter “\n”
- Separção do conteúdo por meio do separador comum “,”
- Tranforma o texto em inteiro
- Tratamento e atribuição

```
void readMensage(int *TYPE, long *SETPOINT, float *KP, float *KI, float *KD){
    int i = 0;
    unsigned char receivedString[30];
    int value[5];
    while(1){
        if(EUSART_is_rx_ready()){
            unsigned char receivedChar = EUSART_Read();
            if(receivedChar == '\n' || receivedChar == '\r'){
                receivedString[i] = '\0';
                break;
            }
            else{
                receivedString[i] = receivedChar;
                i++;
            }
        }
    }
    char *token;
    token = strtok(receivedString,",");
    int j = 0;
    while(token != NULL){
        value[j] = atoi(token);
        token = strtok(NULL,",");
        j++;
    }
    *TYPE = value[0];
    *SETPOINT = value[1];
    *KP = value[2]/1000.0;
    *KI = value[3]/1000.0;
    *KD = value[4]/1000.0;
    receivedString[0] = '\0';
}
```

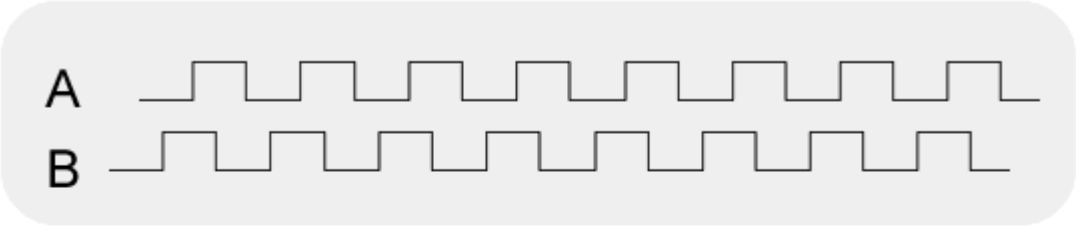
CONTROLAR

Leitura Encoder

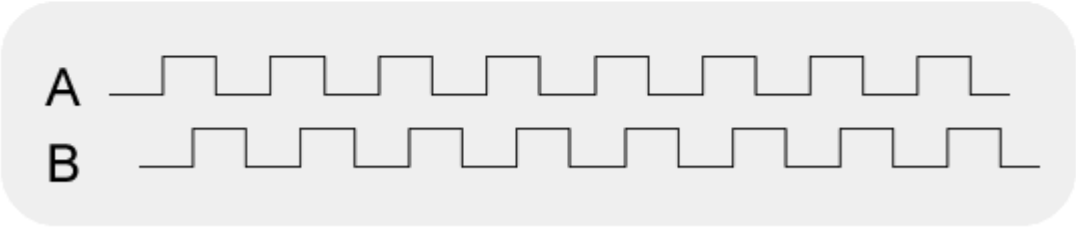
- Interrupção externa nos dois canais do encoder
- Leitura digital
- Variavel global

```
void ENCA_ISR(void){
    if (ENCB_GetValue() == 1) {
        encoder++;
    }
    else{
        encoder--;
    }
}

void ENCB_ISR(void){
    if (ENCA_GetValue() == 0) {
        encoder++;
    }
    else{
        encoder--;
    }
}
```



B leads A



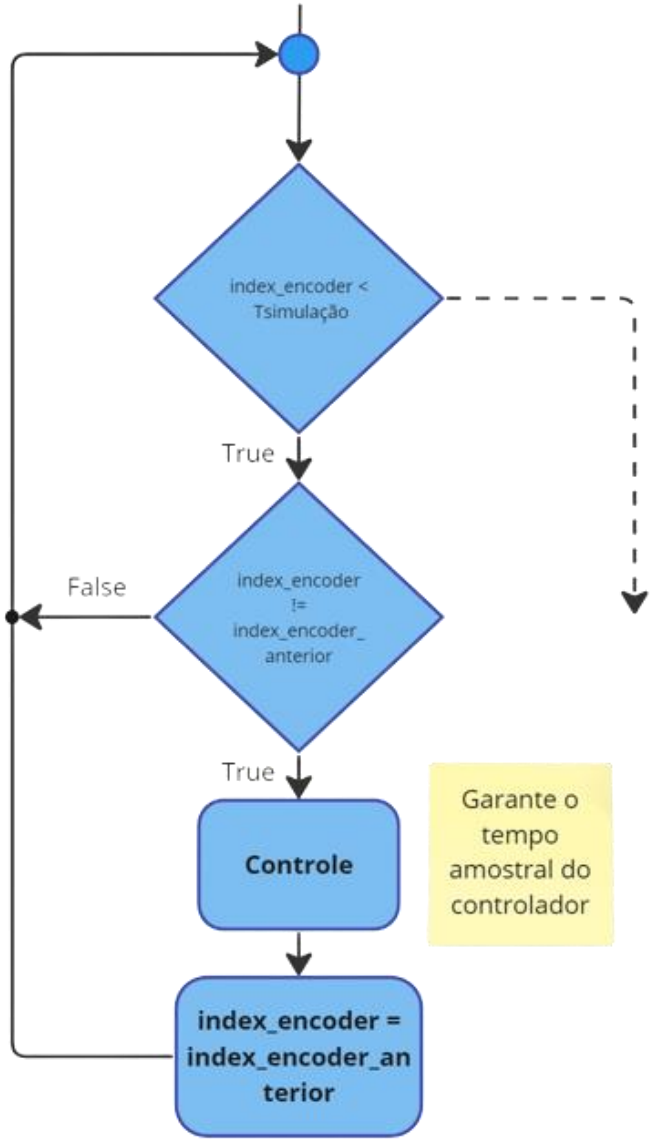
A leads B

CONTROLAR

Estouro de timer para controle discreto

- configurando o timer para o estouro coincidir com o tempo amostral do controle discreto de 10ms
- O estouro causa uma interrupção e a ISR dela é definida
- O importante dela é ser rápida, portanto simples

```
void TIMER_ISR(void){
    index_encoder++;
}
```



CONTROLAR

Controle

- Dependendo do modo selecionado pelo usuário, como controle interativo de posição ou velocidade, ou algum controlador projetado ele executará uma seguinte parte
- Usando os valores fornecidos pelo usuário ou usando a equação de diferenças
- Utiliza um saturador
- Aplica no motor a saída do controlados

```
void PID(){
    while(index_encoder < simulationTime/deltaT){
        if(index_encoder != index_encoder_anterior){
            if(type == 1){
                vel = getVelocity();
                error = (float) (setPoint*nPulseTurn/60000.0) - vel;
                correction += kp * error + ki * error_integrativo - kd * (vel - lastVel);
                error_integrativo += error;
                error_integrativo = constrain(error_integrativo,-600.0,600.0);
                lastVel = vel;
                Data[index_encoder] = vel * 60000.0 / ((float) nPulseTurn);
                velPulse_ms = (encoder - lastPulse)/10.0;
                lastPulse = encoder;
            }
            else{
                if(type == 0){
                    pos = getPosition();
                    error = (float) ((setPoint*nPulseTurn/360.0) - pos);
                    correction = kp * error + ki * error_integrativo - kd * (float) (pos - lastPos);
                    error_integrativo += error;
                    error_integrativo = constrain(error_integrativo,-360.0*nPulseTurn/360.0,360.0*nPulseTurn/360.0);
                    lastPos = pos;
                    Data[index_encoder] = pos * 360.0 / ((float) nPulseTurn);
                }
                else if(type == 2){//AF
                    pos = getPosition();
                    error = (float) ((setPoint*nPulseTurn/360.0) - pos);
                    correction = 0.136 * error - 0.1307 * error_anterior + 0.9548 * correction;
                    error_anterior = error;
                    Data[index_encoder] = pos * 360.0 / ((float) nPulseTurn);
                }
                index_encoder_anterior = index_encoder;
                correction = constrain(correction,-255.0,255.0);
                moveMotor((int) correction);
            }
        }
        moveMotor(0);
}
```

CONTROLAR

Controle

- Atribui, dependendo do sinal da saída do controlador, os estados dos IOs para a ponte H
- Tratamento da zona morta do motor
- Atribui o valor a saída PWM

```
void moveMotor(int m){
    if(m > 0){
        A_SetHigh();
        B_SetLow();
        m += 30;
//        m = constrain(m,20,255);
    }
    else if (m < 0){
        A_SetLow();
        B_SetHigh();
        m -= 30;
//        m = constrain(m,-255,-20);
    }
    PWM6_LoadDutyValue(abs(m));
}
```

ENVIAR DADOS

Envio

- Com os dados armazenados ocorre o envio do pacote de dados pela UART

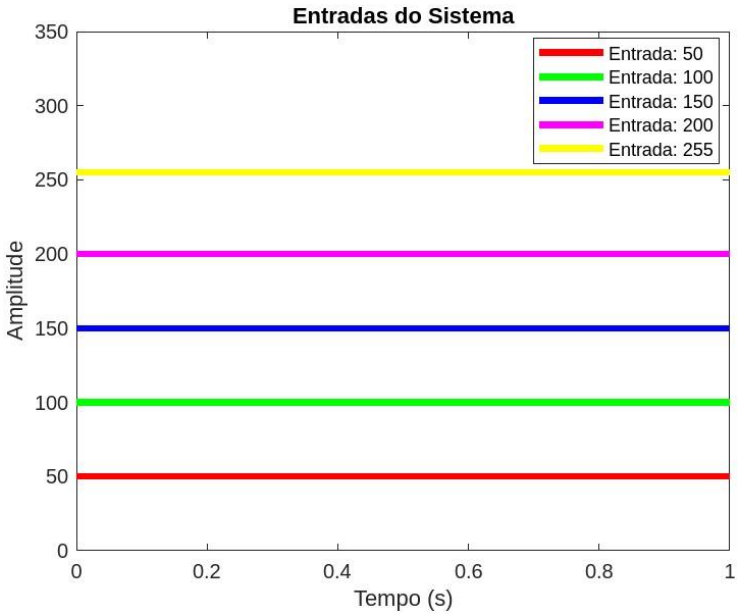
```
void sendMensage(){
    for (int index = 0; index < simulationTime/deltaT; index++) {
        if(type == 7){
            printf("%i\n",Data[index]);//Ensaio
        }
        else{
            printf("%i,%i",Data[index],Time[index]);//Posicao
        }

        if (index < simulationTime/deltaT - 1) {
            if(type != 7){
                printf(",");
            }
        }
        __delay_ms(1);
    }
    printf("\n");
}
```

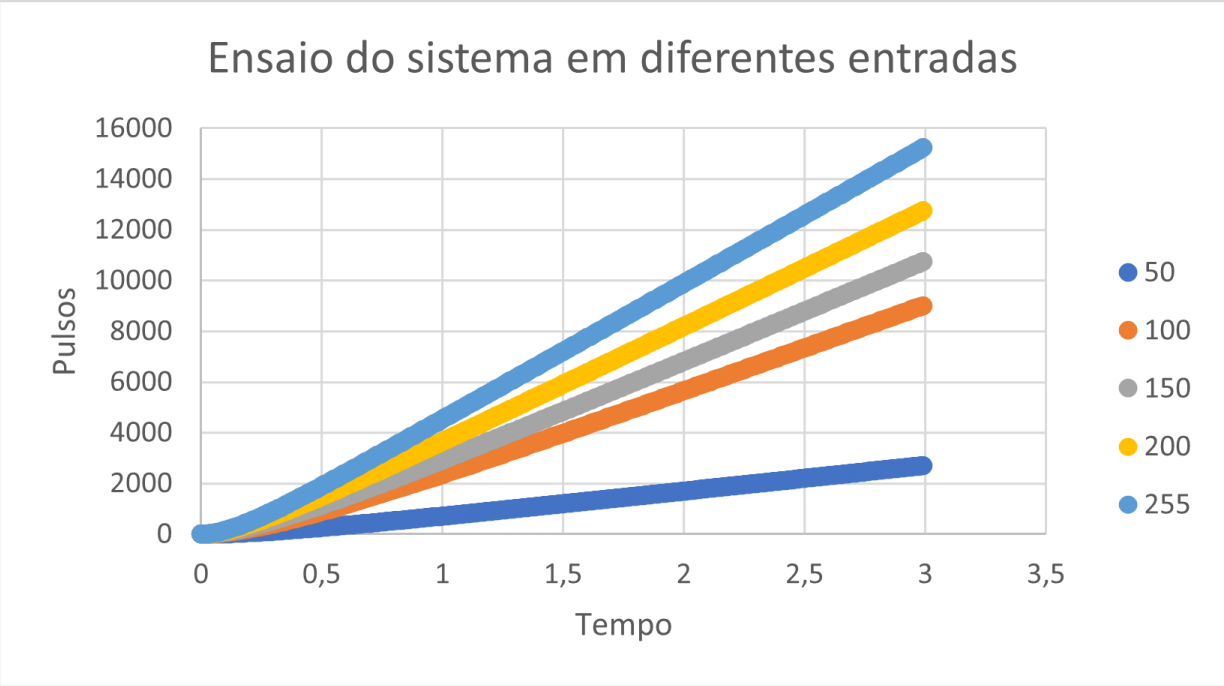
PROJETO DOS CONTROLADORES

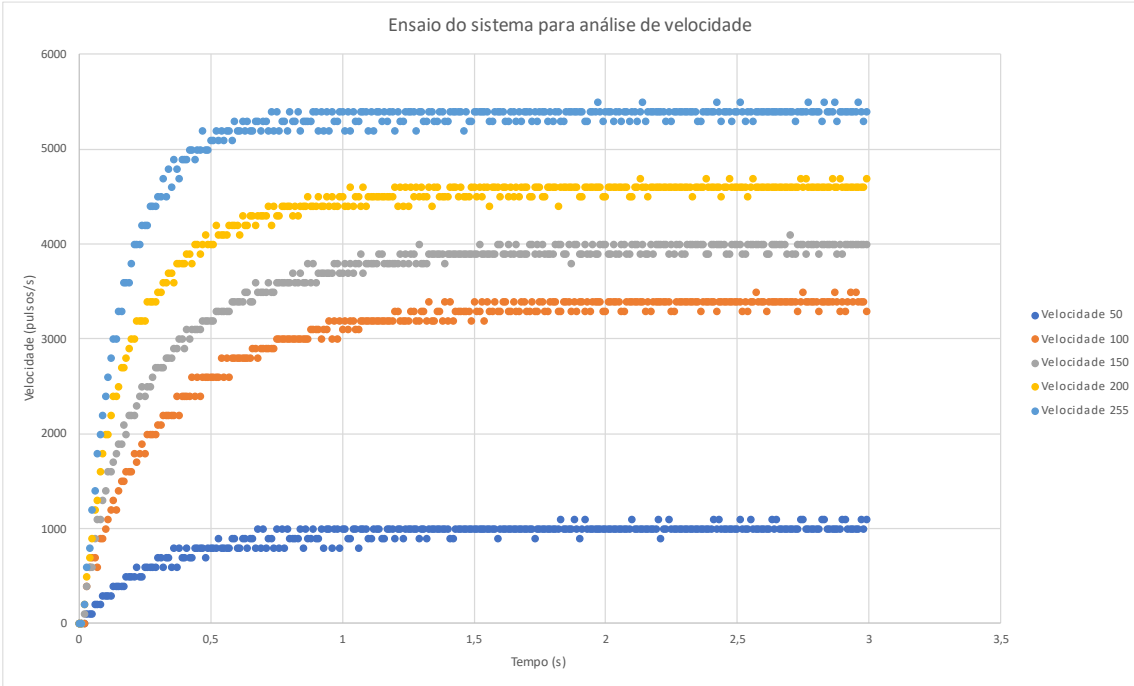
- No projeto foi proposto 3 projetos de controladores com o objetivo de comparar as suas eficiências e implementações, utilizando tanto métodos analítico como o software.
  - Avanço de fase
  - Proporcional – Derivativo
  - PID

- Primeira etapa consiste em obter o modelo do motor, para isso será usado o método do ensaio, com entradas definidas.



PROJETO DOS CONTROLADORES





PROJETO DOS CONTROLADORES

Velocidade

Analítico

Função de Transferência	Constante de tempo (s)	Ganho
$G_{50} = \frac{20}{0{,}3s + 1}$	0,3	20
$G_{100} = \frac{34}{0{,}3s + 1}$	0,3	34
$G_{150} = \frac{26{,}67}{0{,}27s + 1}$	0,27	26,67
$G_{200} = \frac{23}{0{,}19s + 1}$	0,19	23
$G_{255} = \frac{21{,}176}{0{,}18s + 1}$	0,18	21,176
$G_M = \frac{24{,}97}{0{,}248s + 1}$	0,248	24,97

MatLab

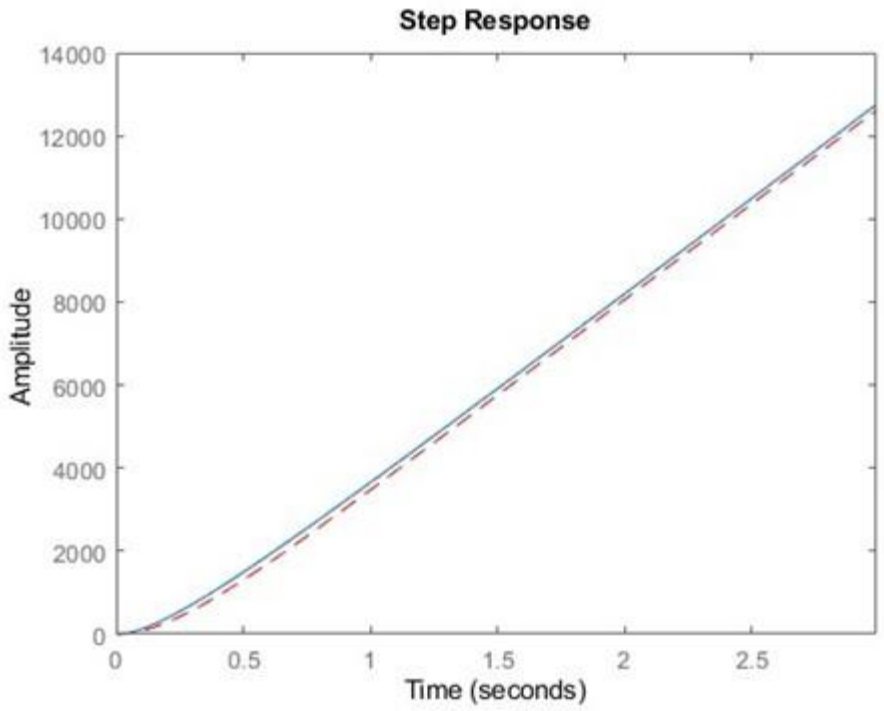
Função de Transferência	Constante de tempo (s)	Ganho
$G_{50} = \frac{20{,}128}{0{,}321s + 1}$	0,321	20,128
$G_{100} = \frac{33{,}697}{0{,}353s + 1}$	0,353	33,697
$G_{150} = \frac{26{,}26}{0{,}295s + 1}$	0,295	26,26
$G_{200} = \frac{22{,}776}{0{,}21s + 1}$	0,21	22,776
$G_{255} = \frac{21{,}04}{0{,}16s + 1}$	0,16	21,04
$G_M = \frac{24{,}7802}{0{,}2678s + 1}$	0,2678	24,7802

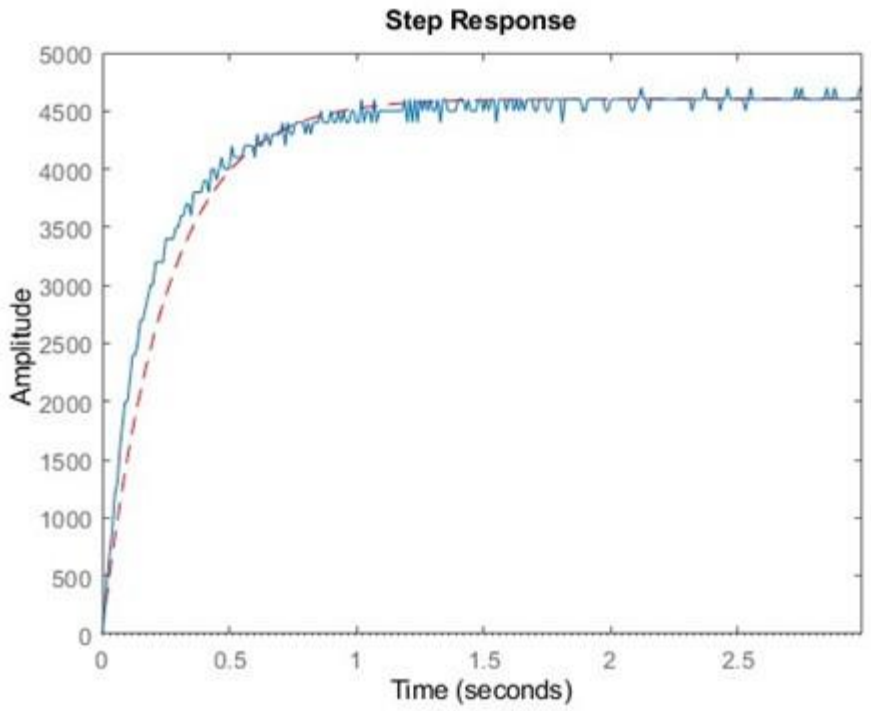
PROJETO DOS CONTROLADORES

Posição

Analítica			Software
Função de transferência	Cte. De tempo	Ganho	Função de transferência
$G_{50} = \frac{19,93}{s(0,3s+1)}$	0,3	19,93	$G_{50} = \frac{63,67}{s^2+3,342s+4,64*10^{-9}}$
$G_{100} = \frac{33,68}{s(0,31s+1)}$	0,31	33,68	$G_{100} = \frac{88,53}{s^2+2,748s+1,368*10^{-9}}$
$G_{150} = \frac{26,17}{s(0,27s+1)}$	0,27	26,17	$G_{150} = \frac{74,97}{s^2+2,978s+2,315*10^{-7}}$
$G_{200} = \frac{23,4}{s(0,2s+1)}$	0,2	23,4	$G_{200} = \frac{97,9}{s^2+4,43s+2,003*10^{-9}}$
$G_{255} = \frac{20,78}{s(0,16s+1)}$	0,16	20,78	$G_{255} = \frac{135,7}{s^2+6,521s+4,108*10^{-10}}$
$G_M = \frac{24,794}{s(0,248s+1)}$	0,248	24,794	$G_M = \frac{92,154}{s^2+4,002s+4,8*10^{-8}}$

PROJETO DOS CONTROLADORES



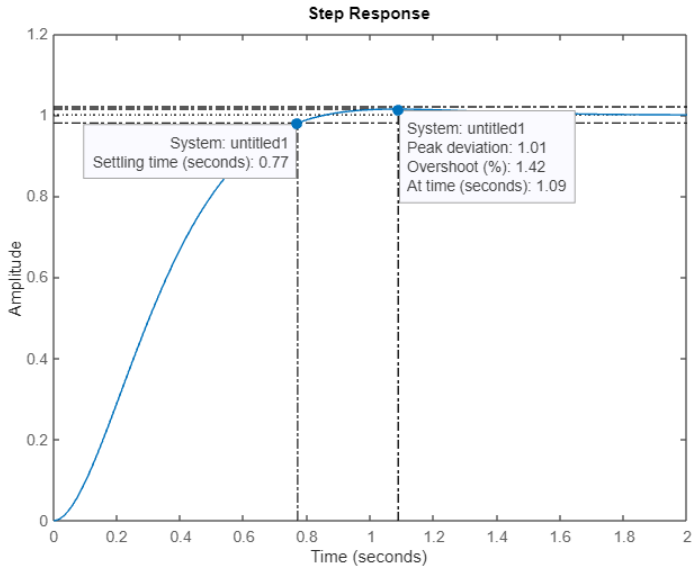


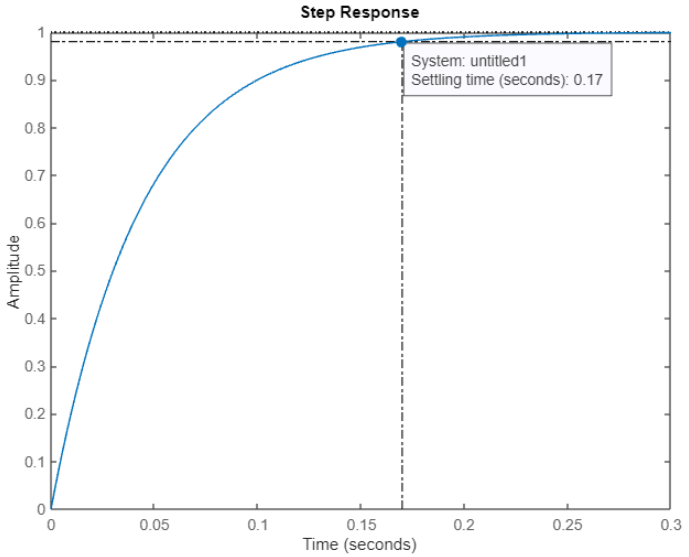
PROJETO DOS CONTROLADORES

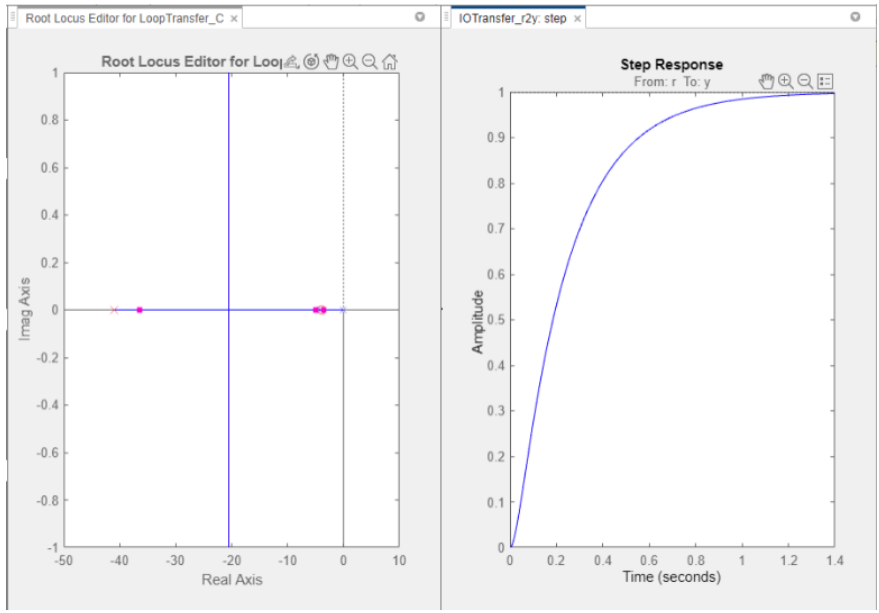
$$G_{AF}(s) = \frac{0{,}2632(s+4)}{s+8{,}46}$$

$$G_{PID} = \left(1 + \frac{1}{83375000 * s} + 0{,}2499 * s\right)$$

$$G_{PD}(s) = 0{,}17016 * \frac{(1+0{,}25s)}{(1+0{,}024s)}$$

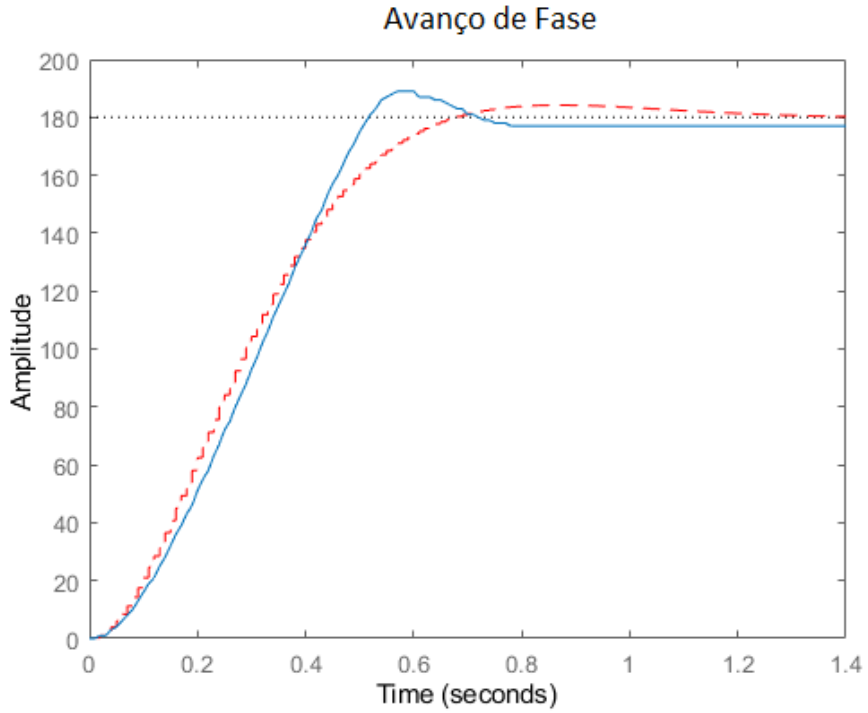






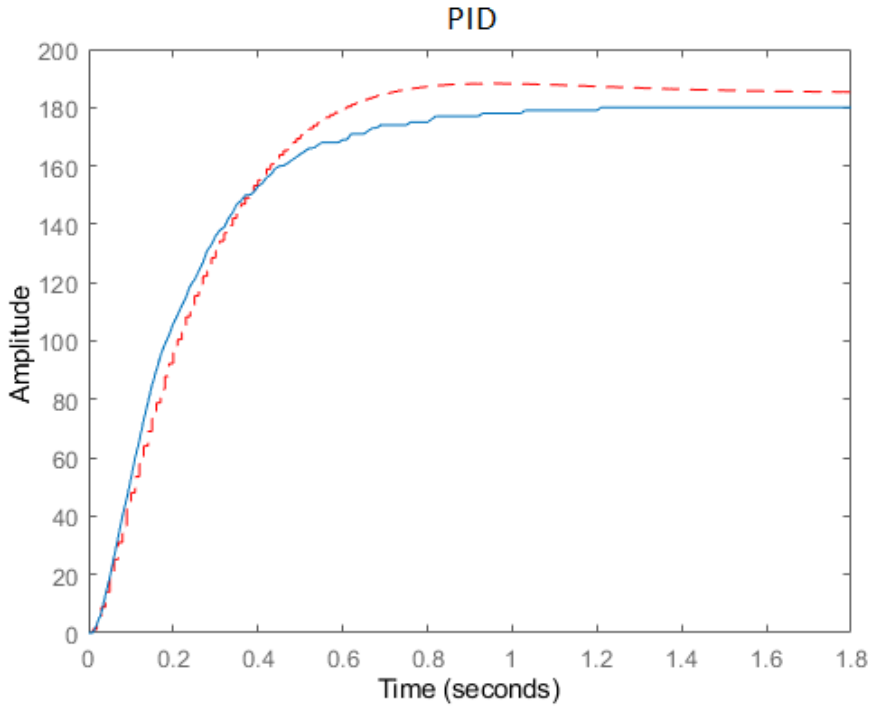
PROJETO DOS CONTROLADORES

$$y_{af}(kT) = 0{,}2632u[kT] - 0{,}2531u[(k-1)T] + 0{,}9189y_{af}[(k-1)T]$$



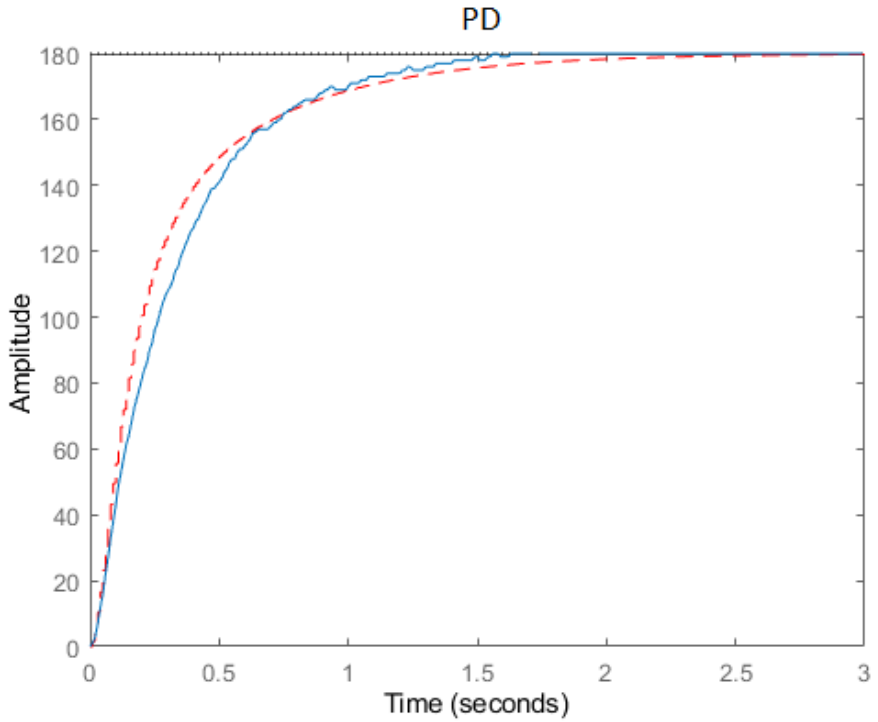
PROJETO DOS CONTROLADORES

$$y_{PID}(kT) = 1{,}509u[kT] - 2{,}94u[(k-1)T] + 1{,}431u[(k-2)T] + 1{,}607y_{af}[(k-1)T] - 0{,}6065y_{PID}[(k-2)]$$



PROJETO DOS CONTROLADORES

$$y_{PD}(kT) = 1{,}772u[kT] - 1{,}715u[(k-1)T] + 0{,}6592y_{PD}[(k-1)T]$$



OBRIGADO

Leonardo Oneda Galvani
Guilherme Nami Bortolozi
Henrique Fortuna Accorinti
Matheus Ferreira Palú