INSTITUTO FEDERAL DE SANTA CATARINA

CÂMPUS FLORIANÓPOLIS

DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

Cleisson Fernandes da Silva

**CONTROLADOR DIGITAL UTILIZANDO O ESPAÇO DE ESTADOS**

Florianópolis, 2018

**SUMÁRIO**

[1 INTRODUÇÃO 3](#_Toc531118317)

[2 CONTROLADOR DIGITAL 4](#_Toc531118318)

[2.1 A PLANTA 4](#_Toc531118319)

[2.2 VALIDAÇÃO DO MODELO 6](#_Toc531118320)

[2.3 ANÁLISE DO SISTEMA 7](#_Toc531118321)

[2.4 PROJETO DO CONTROLADOR 8](#_Toc531118322)

[2.5 PROJETO DO OBSERVADOR 10](#_Toc531118323)

[2.6 DISCRETIZAÇÃO 12](#_Toc531118324)

[2.7 SIMULAÇÃO 13](#_Toc531118325)

[2.8 IMPLEMENTAÇÃO 16](#_Toc531118326)

[2.9 COMPARAÇÃO 18](#_Toc531118327)

[3 CONCLUSÃO 18](#_Toc531118328)

[APÊNDICE A – Códigos Matlab 19](#_Toc531118329)

[APÊNDICE B – Códigos Microcontrolador 21](#_Toc531118330)

[BIBLIOGRAFIA 28](#_Toc531118331)

# INTRODUÇÃO

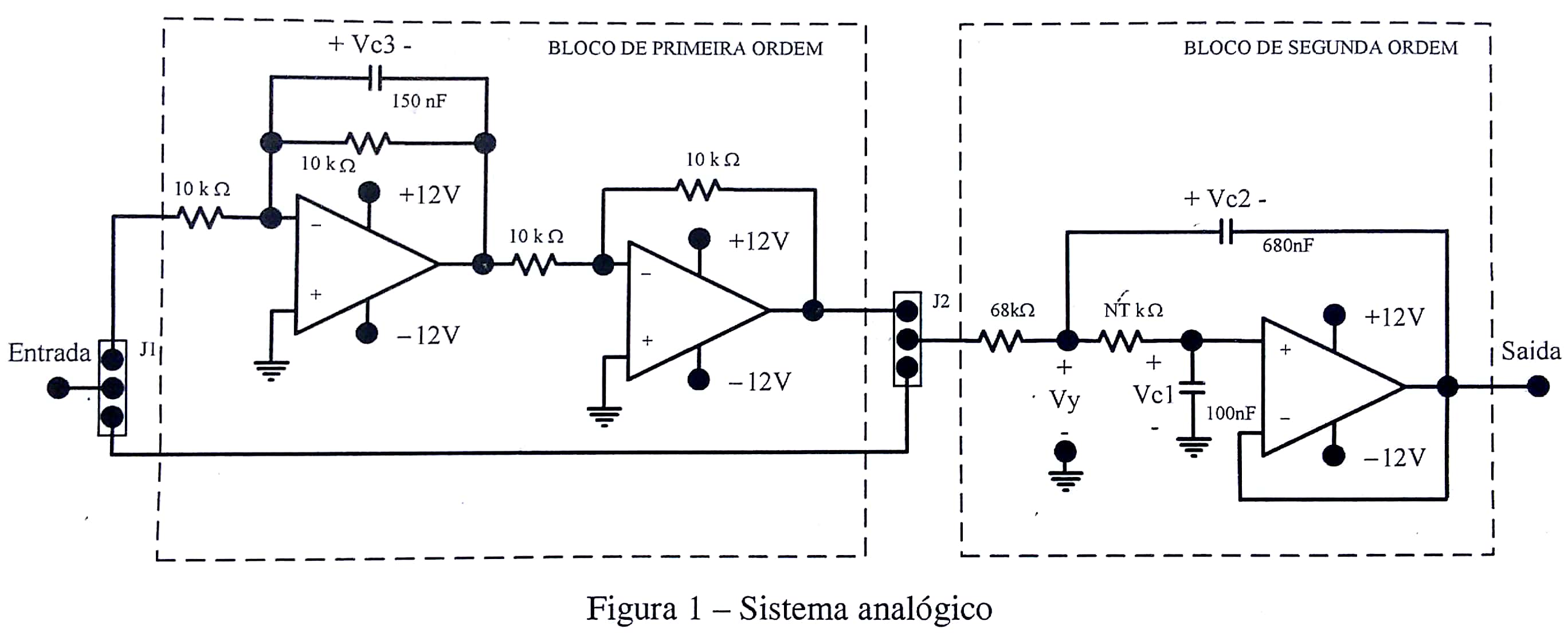
O uso de controladores digitais está cada vez maior. Isso se deve ao fato que controladores digitais são confiáveis, baratos e são capazes de assumir as funções de um controlador analógico. Isso é feito em grande parte porque é econômico, já que o hardware é substituído por software. Funções complexas também são muito mais fáceis de implementar e o registro e o monitoramento são fáceis com computadores. Os controladores digitais também são capazes de assumir muitas formas, como a de uma placa de microprocessador, um microcontrolador ou um controlador lógico programável. Os sistemas de controle digital são flexíveis, baratos, escaláveis ​​e adaptáveis. Portanto, eles são amplamente utilizados em implementações de múltiplos sistemas de controle.

# CONTROLADOR DIGITAL

## A PLANTA

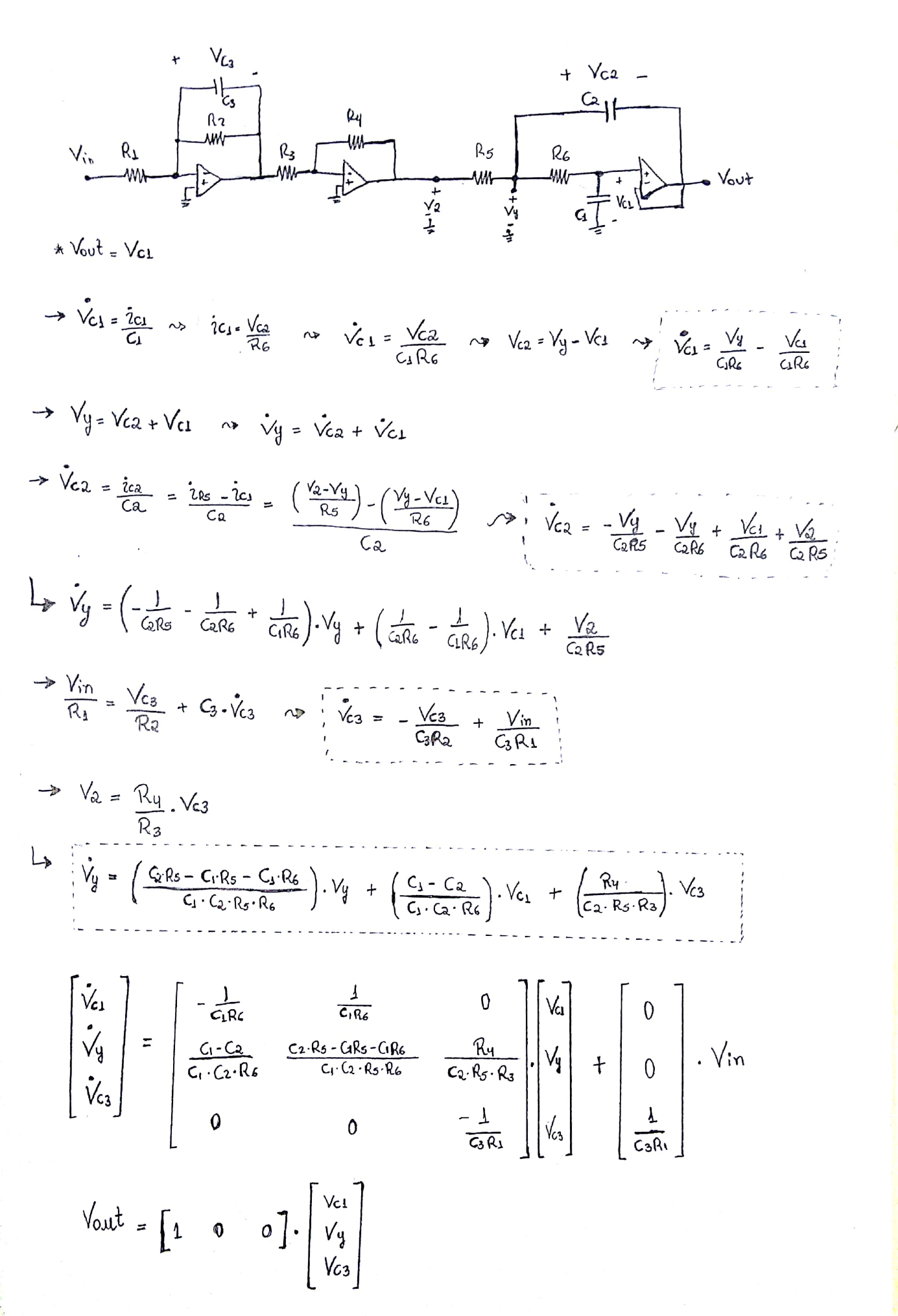
A planta a ser controlada está apresentada na Figura 1, onde NT é 24. Além disso o capacitor de 150 nF foi substituído por um capacitor de 15 nF. Essa planta é um circuito que contém um bloco de primeira ordem e um bloco de segunda ordem em sequência.

Figura 1 – Circuito da Planta a ser controlada



A modelagem da planta em espaço de estados se encontra na Figura 2. Os estados escolhidos foram Vc1, Vy e Vc3.

Figura 2 – Modelagem da planta no espaço de estados



## VALIDAÇÃO DO MODELO

Para validar o modelo calculado foi comparado em uma simulação a modelagem do circuito e um modelo em espaço de estados com as matrizes obtidas.

O circuito no Simulink está apresentado na Figura 3 e o resultado na Figura 4. Portanto, o resultado obtido pelo equacionamento do circuito descreve este corretamente.

Figura 3 – Circuito no Simulink para comparar os estados

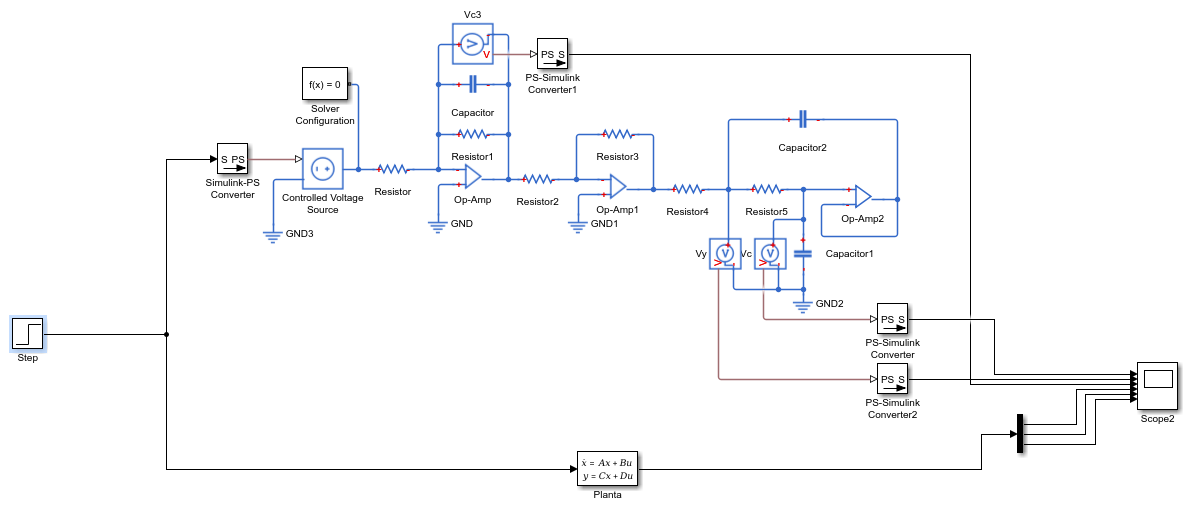
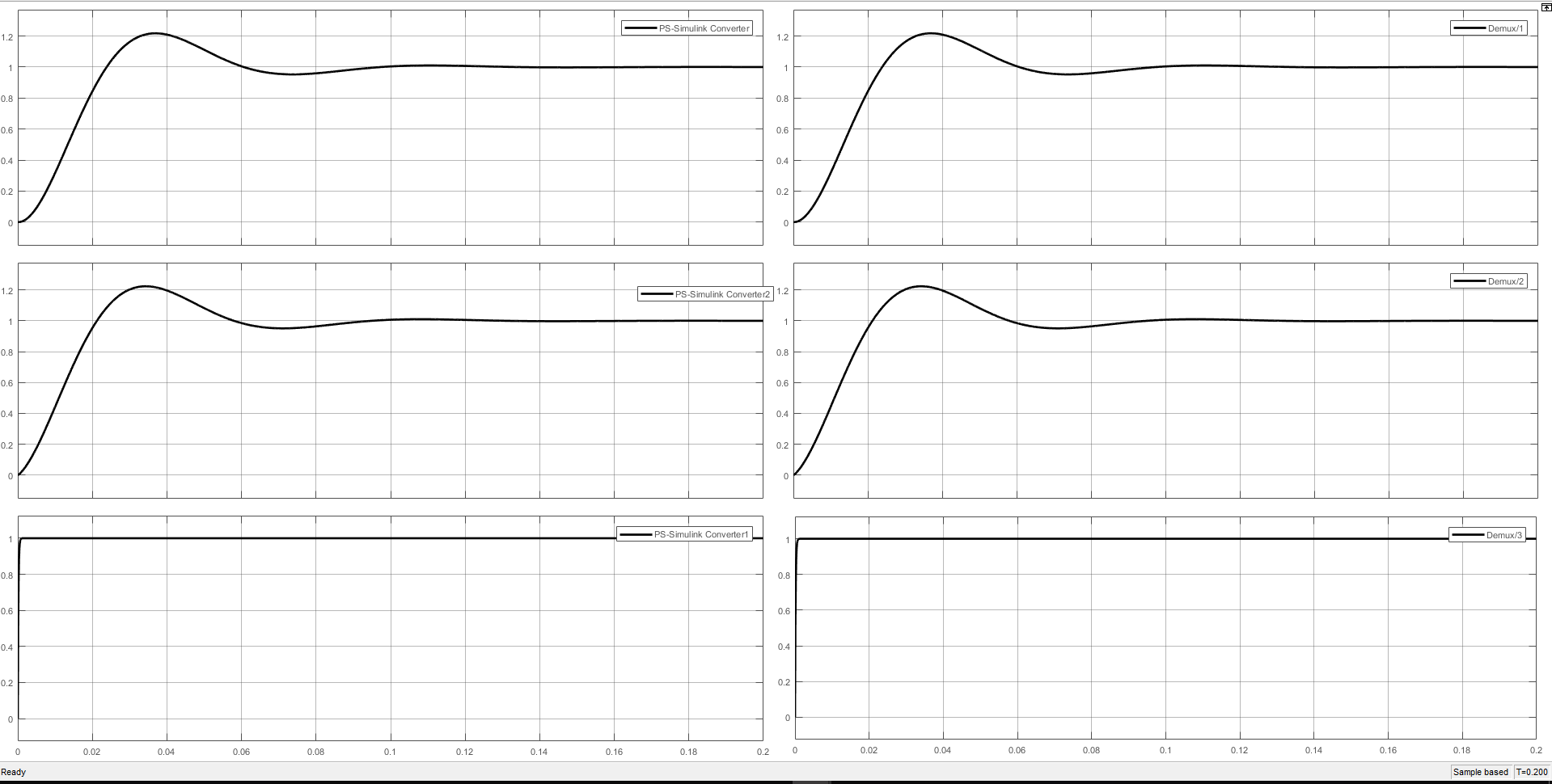


Figura 4 – Resultado da comparação dos estados



## ANÁLISE DO SISTEMA

Uma das primeiras coisas necessárias para se fazer é analisar se o sistema de malha aberta (sem qualquer controle) é estável. Os polos da função de transferência da planta são os autovalores da matriz A do sistema. Utilizando a função *eig()* do Matlab encontramos os seguintes polos:

Por inspeção, pode ser visto que todos os polos estão no plano esquerdo e portando significa que o sistema em malha aberta é estável.

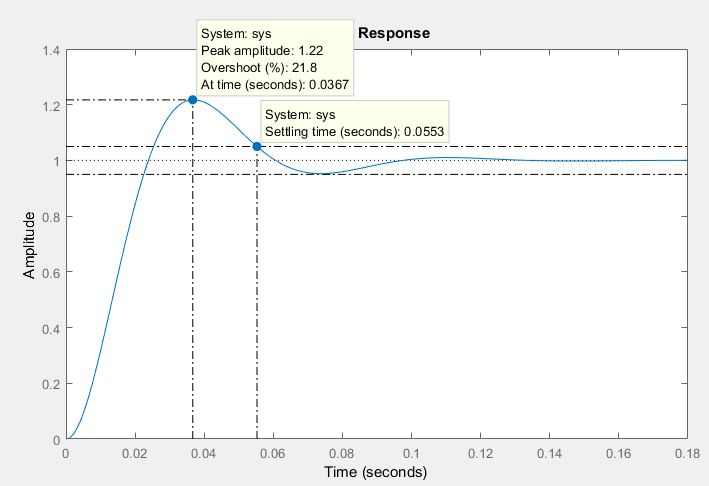
Outra coisa necessária é testar a controlabilidade e observabilidade do sistema.

Um sistema é controlável se sempre existir uma entrada de controle, u(t), que transfere qualquer estado do sistema para qualquer outro estado em tempo finito. O sistema é controlável se o rank da matriz de controlabilidade é o mesmo número que a quantidade de estados. Utilizando o função *crtb()* do Matlab encontramos que o rank da matriz de controlabilidade é igual à 3. Portanto o sistema é controlável.

Todas as variáveis de estado de um sistema podem não serem diretamente mensuráveis, por exemplo, se o componente estiver em um local inacessível. Nesses casos, é necessário estimar os valores das variáveis de estado internas desconhecidas usando apenas as saídas do sistema disponíveis. Um sistema é observável se o estado inicial, pode ser determinado com base no conhecimento da entrada do sistema, , e a saída do sistema, , durante algum intervalo de tempo finito. O sistema é dito observável se o rank da matriz de observabilidade é o mesmo número que a quantidade de estados. Utilizando o função *obsv()* do Matlab encontramos que o rank da matriz de observabilidade é igual à 3. Portanto o sistema é observável.

As características do sistema obtido está apresentada na Figura 5.

Figura 5 – Características da resposta ao degrau



Para cumprir os requisitos de projeto, as especificações da resposta ao degrau se tornam:

Utilizando as fórmulas de especificações para sistema de segunda ordem:

Foi estimado os valores de e como sendo:

## PROJETO DO CONTROLADOR

A planta é do tipo 0 e portanto o projeto de servossistema necessita de um integrador no ramo direto entre o comparador de erro e a planta para se ter erro nulo de regime permanente.

O controlador deste sistema foi feito utilizando o método de posicionamento de polos. Escolhendo uma matriz de ganho de realimentação de estado apropriada K, podemos colocar os polos de malha fechada em qualquer lugar que quisermos (porque o sistema é controlável). Podemos usar a da função *acker()* do MATLAB para encontrar o ganho de feedback do estado, K, que fornecerá os polos de malha fechada desejados.

A localização dos polos desejados foi feita utilizando um script Matlab. Primeiro foi escolhido o valor de e desejados e então foi utilizado o mesmo método da análise do sistema para encontrar os valores de e estimados. O script se encontra no Apêndice A.

Os polos escolhidos devem satisfazer as especificações mínimas estabelecidas. Os valores escolhidos foram:

O terceiro polo desejado é escolhido para que tenha uma dinâmica mais rápida que os dois primeiros. Foi definido que o terceiro polo seria 2 vezes mais rápido que a oscilação do sistema, portanto . Para o quarto polo considerou sendo o mesmo valor que o terceiro polo.

A estabilidade do sistema de malha fechada pode ser determinado encontrando os autovalores da matriz , que corresponde aos polos de malha fechada. Utilizando a função *eig()* do Matlab encontramos os seguintes polos:

Por inspeção, pode ser visto que todos os polos estão no plano esquerdo e portando significa que o sistema em malha frechada é estável.

Simulando o realimentação de estado com a configuração mostrada na Figura 6 obtemos o resultado apresentado na Figura 7.

Figura 6 – Diagrama de blocos da realimentação de estados

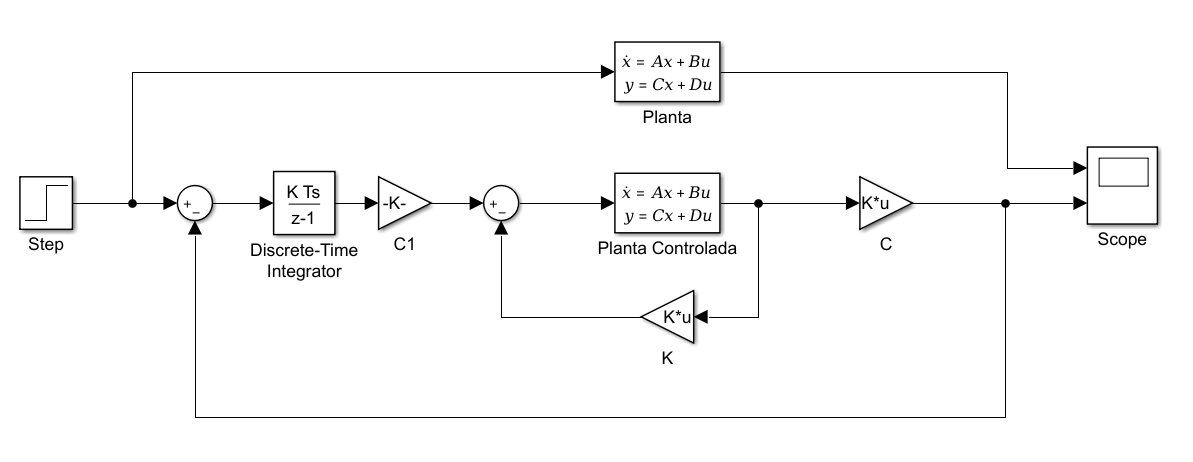
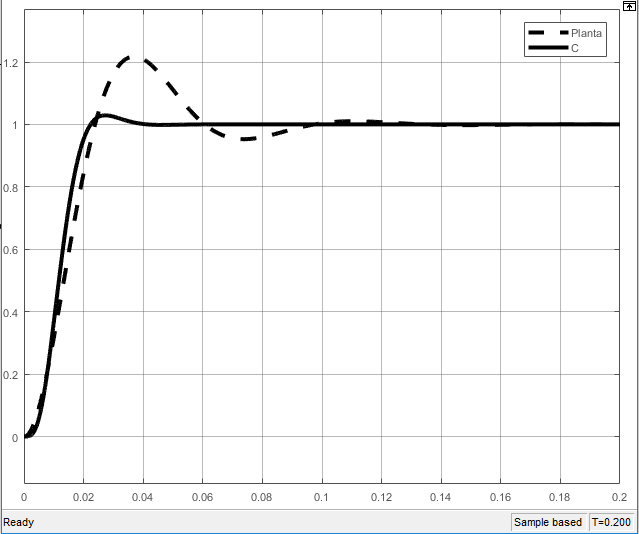


Figura 7 – Resultado da realimentação de estados



A curva da realimentação de estados apresentou um sobressinal de 2.9% e um tempo de acomodação de 19.96ms.

## PROJETO DO OBSERVADOR

Quando não podemos medir todas as variáveis de estado x (geralmente o caso na prática), podemos construir um observador para estimá-las, enquanto medimos apenas a saída.

O observador é basicamente uma cópia da planta; tem a mesma entrada e quase a mesma equação diferencial. Um termo extra compara a saída real medida com a saída estimada ; isso ajudará a corrigir o estado estimado e fará com que ele se aproxime dos valores do estado real .

Como queremos que a dinâmica do observador seja muito mais rápida do que o próprio sistema, precisamos posicionar os polos mais à esquerda do que os polos dominantes do sistema. Portanto os polos do observador foram definidos como:

Devido à dualidade entre controlabilidade e observabilidade, podemos usar a mesma técnica usada para encontrar a matriz de controle, substituindo a matriz pela matriz e tomando as transpostas de cada matriz.

Simulando o observador com a configuração mostrada na Figura 8 podemos comparar os estados estimados do observador com os estados reais. O resultado está apresentado na Figura 9. Podemos perceber que os dois primeiros estados foram estimados corretamente enquanto o terceiro estado sofreu um rápido desvio no começo. Apesar deste pequeno desvio, o terceiro estado é muito mais rápido que os outros estados e logo em seguida o observador segue o valor do terceiro estado real e portanto ele não afetará consideravelmente o restante do sistema.

Figura 8 – Diagrama de blocos do observador

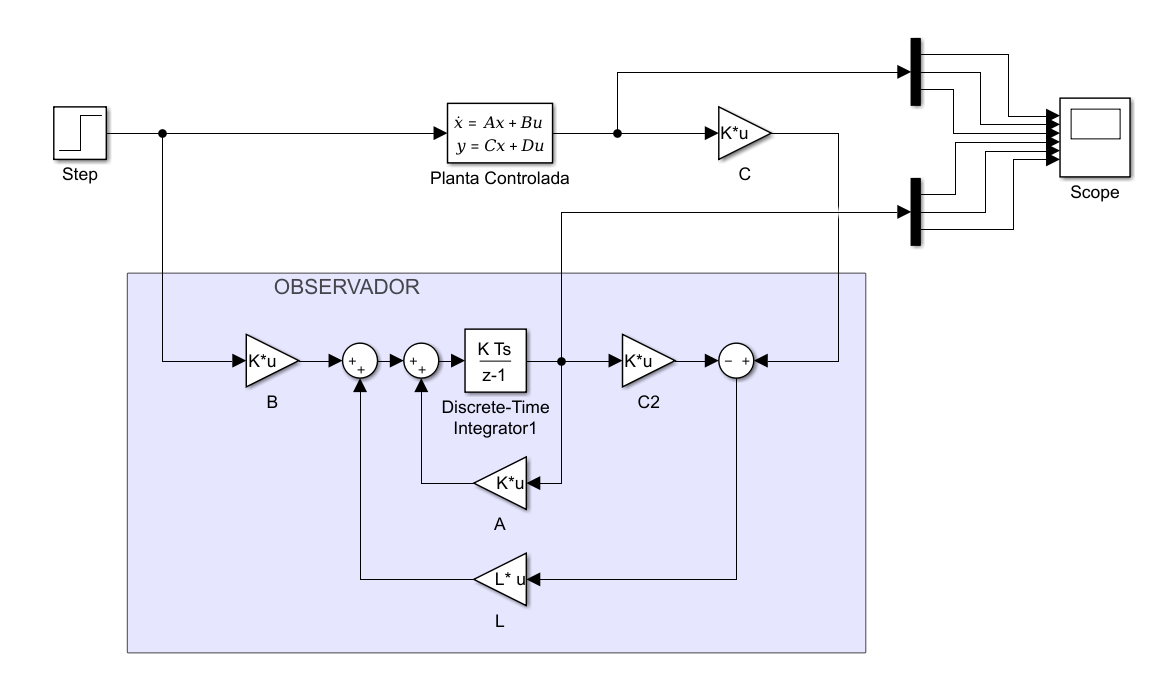
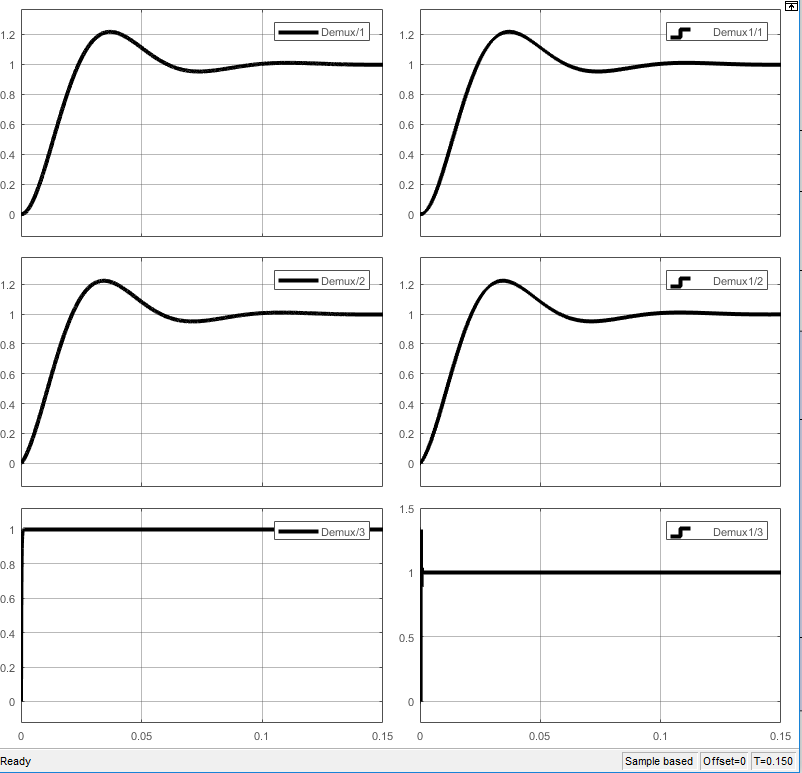


Figura 9 – Resultado da comparação dos estados com o observador



## DISCRETIZAÇÃO

Para implementar o controlador e o observador no microcontrolador é preciso discretizar os mesmos. Podemos aproximar esta discretização apenas transformando os integradores presentes em ambos os circuitos em integradores discretos. Utilizando o método de Euler como aproximação para a integração numérica temos que:

Onde é passo de cálculo e, no caso do microcontrolador, corresponde ao período de amostragem. O período de amostragem necessário para esta discretização é obtido de modo que a taxa de amostragem seja mais rápida do que a dinâmica do sistema.

Primeiramente foi definido a taxa de amostragem sendo o mesmo valor de frequência que os polos do observador, por estes serem a dinâmica mais rápida do sistema. Portando:

Após isso o período de amostragem foi decrescido empiricamente até chegar em um valor considerado razoável para ser implementado no microcontrolador e que não afeta drasticamente no comportamento do controlador com a planta. O tempo de amostragem final foi :

## SIMULAÇÃO

Juntando o Controlador com a realimentação de estados e utilizando as estimativas de estados do Observador obtemos o sistema completo para ser implementado no microcontrolador. O Diagrama de blocos final está apresentado na Figura 10 e o seu resultado da simulação na Figura 11. A saída controlada apresentou sobressinal de 2.6% e um tempo de acomodação de .

Figura 10 – Diagrama de blocos do Controlador e Observador juntos

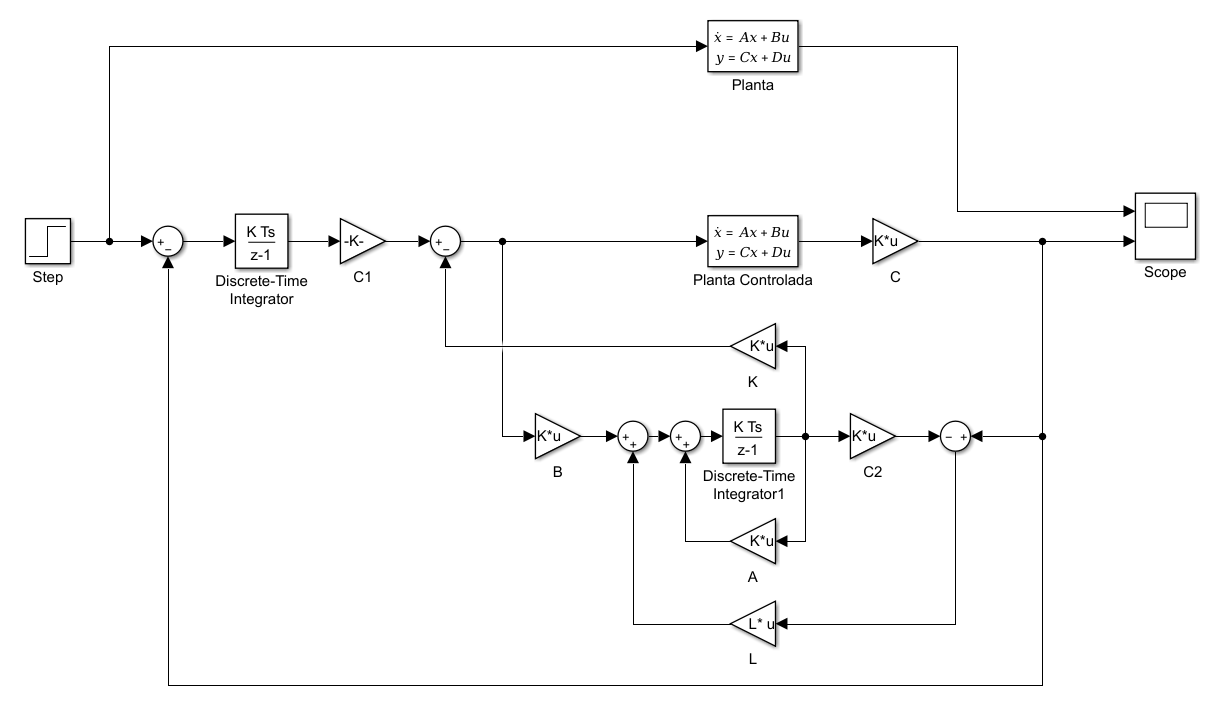
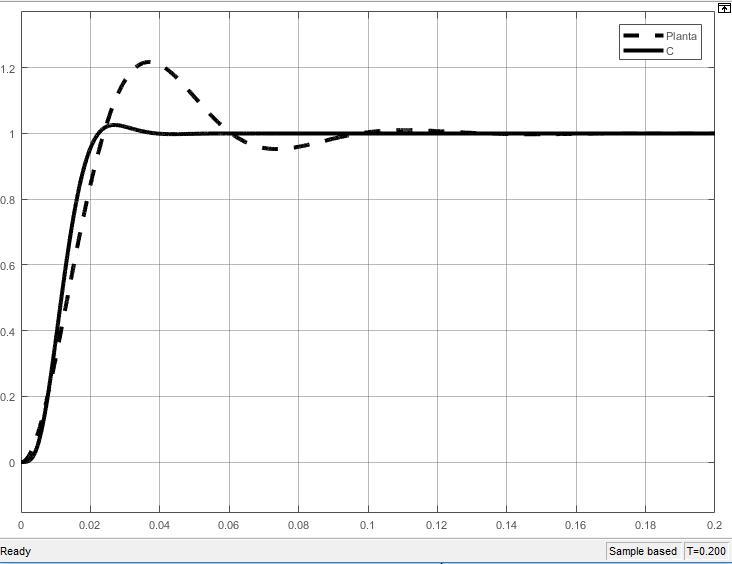


Figura 11 – Resultado do Controlador e Observador juntos



Um fator importante a se verificar é a ação de controle. O esquema para se medir a ação de controle está apresentada na Figura 12 e seu resultado na Figura 13. Como o valor do degrau no microcontrolador vai de 1V à 1,5V temos um degrau de amplitude de 0.5V. O valor máximo que podemos chegar é 3.3V e, partindo do 1V do degrau, temos 2.3V disponíveis para a ação de controle. De acordo com o resultado obtido, a ação de controle não ultrapassa os níveis permitidos.

Figura 12 – Diagrama para se medir a ação de controle

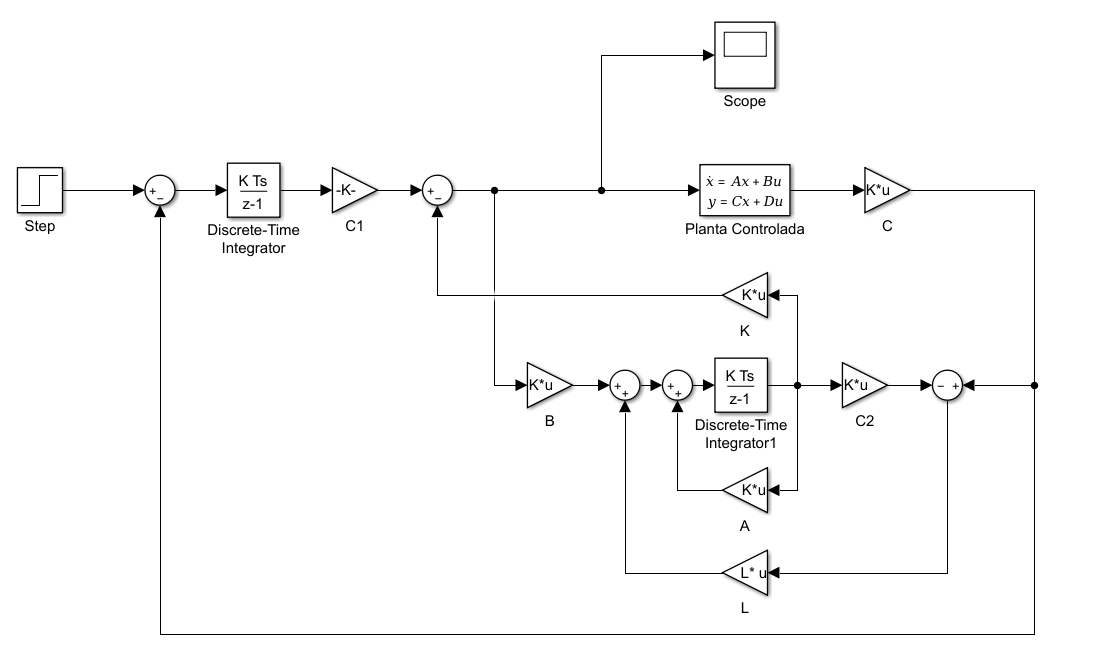
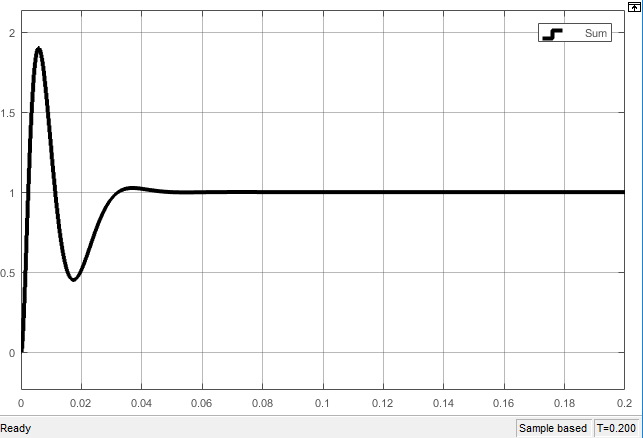


Figura 13 – Ação de controle



No Matlab podemos utilizar um bloco de função que é possível simular as operações executadas dentro do microcontrolador. O esquema de ligação está apresentado na Figura 14. Neste esquema foi adicionado um bloco de ruído para simular a implementação real que estará sujeito à vários tipos de ruídos. O resultado obtido se encontra na Figura 15. A saída controlada apresentou sobressinal de 3.4% e um tempo de acomodação de .

Figura 14 – Diagrama de blocos da simulação com o microcontrolador

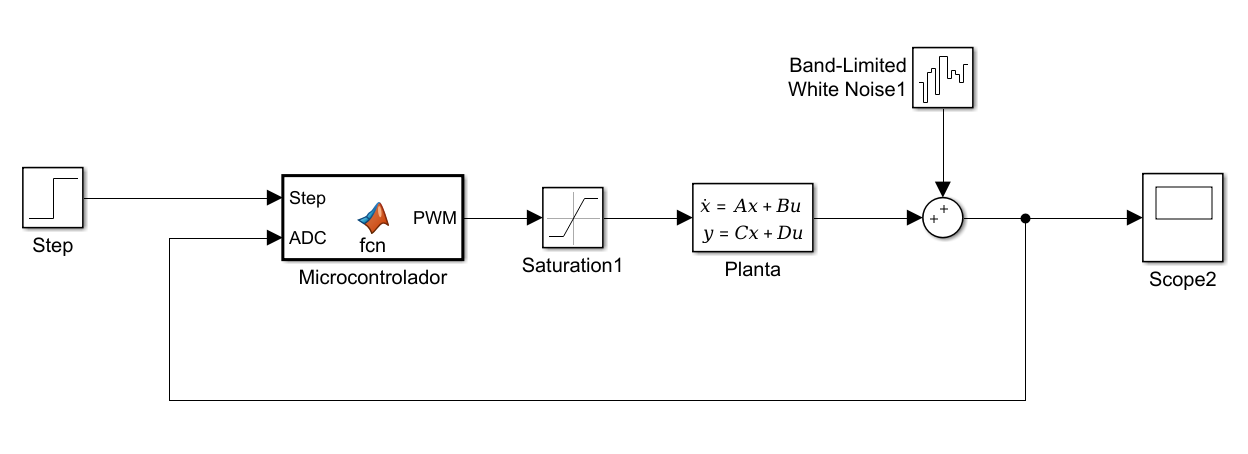
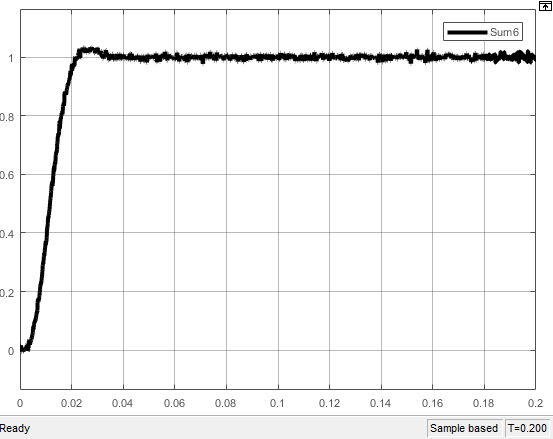


Figura 15 – Resultado da simulação com o microcontrolador



## IMPLEMENTAÇÃO

O controlador digital foi implementado no microcontrolador STM32F103C8T6 da STMicroeletronics. Como o microcontrolador não possui saída D/A, foi utilizado um PWM com uma tensão média correspondendo ao sinal desejado. O código utilizado se encontra no Apêndice B. Pela Figura 16 e a Figura 17 obtemos o valor de . Pela Figura 18 temos que o tempo de acomodação do sistema final é de cerca de 16.8 ms. Pelas capturas e valores obtidos podemos afirmar que o controlador projetado atende os requisitos de projeto.

Figura 16 – Captura da variação inferior

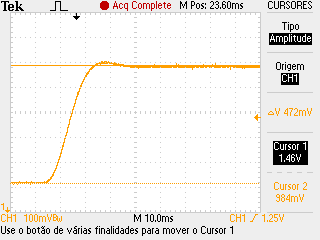


Figura 17 – Captura da variação superior

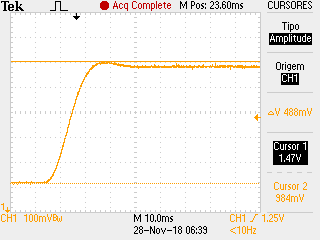
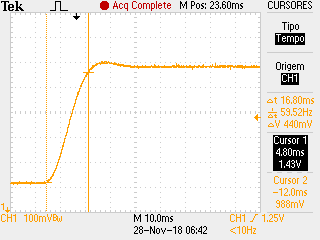
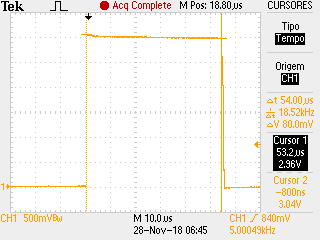


Figura 18 – Captura do tempo de acomodação



Na Figura 19 temos o tempo que o microcontrolador leva para processar e colocar o valor calculado na saída. Podemos ver que o tempo de processamento é de 54 μs.

Figura 19 – Tempo de processamento



## COMPARAÇÃO

|  |  |  |  |
| --- | --- | --- | --- |
|  | Projetado | Simulado | Prático |
| Mp (%) | 5 | 2.6 | 3.39 |
| ts (ms) | 20 | 20 | 16.8 |

# CONCLUSÃO

Foi implementado com sucesso um controlador digital para a planta em questão. O sistema com o controlador apresenta valores de tempo de acomodação e valor de sobressinal menor.

# APÊNDICE A – Códigos Matlab

Script com a Modelagem, cálculos do Controlador e Observador:

clear;

clc;

R1 = 10e3;

R2 = 10e3;

R3 = 10e3;

R4 = 10e3;

R5 = 68e3;

R6 = 24e3;

C1 = 100e-9;

C2 = 680e-9;

C3 = 15e-9;

A = [(-1/(C1\*R6)) 1/(C1\*R6) 0;

(C1 - C2)/(C1\*C2\*R6), (-(C1\*R5 + C1\*R6 - C2\*R5)/(C1\*C2\*R5\*R6)) (R4)/(R3\*C2\*R5);

0, 0, -1/(C3\*R2)];

B = [ 0;

0;

1/(C3\*R1)];

C = [1, 0, 0];

D = 0;

sys = ss(A,B,C,D);

sys\_obsv = rank(obsv(A,C));

sys\_ctrb = rank(ctrb(A,B));

Mp = 0.218;

ts = 55.3e-3;

zeta = -log(Mp)/sqrt(pi^2 + log(Mp)^2);

wn = -log(0.05\*sqrt(1-zeta^2))/(ts\*zeta);

Mp\_desejado = 0.05;

ts\_desejado = 20e-3;

zeta\_desejado = -log(Mp\_desejado)/sqrt(pi^2 + log(Mp\_desejado)^2);

wn\_desejado = -log(0.05\*sqrt(1-zeta\_desejado^2))/(ts\_desejado\*zeta\_desejado);

sigma = zeta\_desejado\*wn\_desejado;

s1 = -sigma + 1i\*wn\_desejado\*sqrt(1-zeta\_desejado^2);

s2 = s1';

s3 = -2\*sigma;

s4 = s3;

%% Controlador de estados

Z = zeros(rank(A),1);

Ac = [A Z; -C 0];

Bc = [B; 0];

Cc = [C 0];

P = [s1 s2 s3 s4];

K = acker(Ac,Bc,P);

%% Observador

sl = -40\*sigma;

Pl = [sl sl sl];

L = acker(A',C',Pl)';

%% Discretização

T\_desejado = 1/((2\*pi\*abs(sl)));

T = 2e-4;

Função para o bloco do microcontrolador:

function PWM = fcn(Step, ADC)

A = single([-416.67, 416.67, 0; -355.39, 333.77, 21.626; 0, 0, -6666.7]);

B3 = single(6666.7);

K = single([-0.99525, 2.106, -0.86308, 106.05]);

L = single([13165, 1.1583e+05, -2.5599]);

Ts = single(0.0002);

%%

global i

global e

global n1

global n2

global n3

global m1

global m2

global m3

%%

rawValue = single((4095\*ADC)/3.3); % Converte valor de tensão para variável

c = single((4499\*rawValue)/4095); % Converte para o domínio de Pulso

setValue = single((4499\*Step)/3.3);

enew = setValue - c;

inew = (Ts\*e + i);

n1new = (Ts\*m1 + n1);

n2new = (Ts\*m2 + n2);

n3new = (Ts\*m3 + n3);

k = (K(1)\*n1new + K(2)\*n2new + K(3)\*n3new);

x = (inew\*K(4) - k);

PWM = double((3.3\*x)/4499); % Converte do domínio do Pulso para Tensão

m1 = single((c-n1new)\*L(1) + (n1new\*A(1,1)) + (n2new\*A(1,2)) + (n3new\*A(1,3)));

m2 = single((c-n1new)\*L(2) + (n1new\*A(2,1)) + (n2new\*A(2,2)) + (n3new\*A(2,3)));

m3 = single((c-n1new)\*L(3) + (n1new\*A(3,1)) + (n2new\*A(3,2)) + (n3new\*A(3,3)) + x\*B3);

e = enew;

i = inew;

n1 = n1new;

n2 = n2new;

n3 = n3new;

test = double([enew]);

# APÊNDICE B – Códigos Microcontrolador

Main.c:

/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

**#include** "stm32f1xx\_hal.h"

**#include** <arm\_math.h>

**#include** "dwt.h"

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private variables ---------------------------------------------------------\*/

ADC\_HandleTypeDef hadc1;

TIM\_HandleTypeDef htim1;

TIM\_HandleTypeDef htim2;

TIM\_HandleTypeDef htim3;

/\* USER CODE BEGIN PV \*/

/\* Private variables ---------------------------------------------------------\*/

GPIO\_PinState state;

uint32\_t counter = 0;

int32\_t setValue = 1000;

**volatile** uint32\_t rawValue;

//volatile int32\_t x[3];

//volatile int32\_t y[3];

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**static** **void** **MX\_TIM2\_Init**(**void**);

**static** **void** **MX\_TIM1\_Init**(**void**);

**static** **void** **MX\_TIM3\_Init**(**void**);

**static** **void** **MX\_ADC1\_Init**(**void**);

**static** **void** **MX\_NVIC\_Init**(**void**);

**void** **HAL\_TIM\_MspPostInit**(TIM\_HandleTypeDef \*htim);

/\* USER CODE BEGIN PFP \*/

/\* Private function prototypes -----------------------------------------------\*/

/\* USER CODE END PFP \*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\*

\* @retval None

\*/

**int** **main**(**void**)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration----------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_TIM2\_Init();

MX\_TIM1\_Init();

MX\_TIM3\_Init();

MX\_ADC1\_Init();

/\* Initialize interrupts \*/

MX\_NVIC\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_TIM\_PWM\_Start(&htim2, TIM\_CHANNEL\_1);

HAL\_TIM\_Base\_Start\_IT(&htim1);

HAL\_TIM\_Base\_Start\_IT(&htim3);

HAL\_ADC\_Start\_IT(&hadc1);

DWT\_Enable();

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

**void** **SystemClock\_Config**(**void**)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct;

RCC\_ClkInitTypeDef RCC\_ClkInitStruct;

RCC\_PeriphCLKInitTypeDef PeriphClkInit;

/\*\*Initializes the CPU, AHB and APB busses clocks

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.HSEPredivValue = RCC\_HSE\_PREDIV\_DIV1;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL9;

**if** (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

/\*\*Initializes the CPU, AHB and APB busses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

**if** (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

PeriphClkInit.PeriphClockSelection = RCC\_PERIPHCLK\_ADC;

PeriphClkInit.AdcClockSelection = RCC\_ADCPCLK2\_DIV6;

**if** (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInit) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

/\*\*Configure the Systick interrupt time

\*/

HAL\_SYSTICK\_Config(HAL\_RCC\_GetHCLKFreq()/1000);

/\*\*Configure the Systick

\*/

HAL\_SYSTICK\_CLKSourceConfig(SYSTICK\_CLKSOURCE\_HCLK);

/\* SysTick\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(*SysTick\_IRQn*, 0, 0);

}

/\*\*

\* @brief NVIC Configuration.

\* @retval None

\*/

**static** **void** **MX\_NVIC\_Init**(**void**)

{

/\* TIM1\_UP\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(*TIM1\_UP\_IRQn*, 0, 0);

HAL\_NVIC\_EnableIRQ(*TIM1\_UP\_IRQn*);

/\* ADC1\_2\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(*ADC1\_2\_IRQn*, 0, 0);

HAL\_NVIC\_EnableIRQ(*ADC1\_2\_IRQn*);

/\* TIM3\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(*TIM3\_IRQn*, 0, 0);

HAL\_NVIC\_EnableIRQ(*TIM3\_IRQn*);

}

/\* ADC1 init function \*/

**static** **void** **MX\_ADC1\_Init**(**void**)

{

ADC\_ChannelConfTypeDef sConfig;

/\*\*Common config

\*/

hadc1.Instance = ADC1;

hadc1.Init.ScanConvMode = ADC\_SCAN\_DISABLE;

hadc1.Init.ContinuousConvMode = *DISABLE*;

hadc1.Init.DiscontinuousConvMode = *DISABLE*;

hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;

hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;

hadc1.Init.NbrOfConversion = 1;

**if** (HAL\_ADC\_Init(&hadc1) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

/\*\*Configure Regular Channel

\*/

sConfig.Channel = ADC\_CHANNEL\_0;

sConfig.Rank = ADC\_REGULAR\_RANK\_1;

sConfig.SamplingTime = ADC\_SAMPLETIME\_13CYCLES\_5;

**if** (HAL\_ADC\_ConfigChannel(&hadc1, &sConfig) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

}

/\* Fs = Fcpu/((Prescaler+1)\*(Period+1)) \*/

/\* TIM1 init function \*/

/\* Tempo de Amostragem\*/

**static** **void** **MX\_TIM1\_Init**(**void**)

{

TIM\_ClockConfigTypeDef sClockSourceConfig;

TIM\_MasterConfigTypeDef sMasterConfig;

htim1.Instance = TIM1;

htim1.Init.Prescaler = 7;

htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim1.Init.Period = 1799;

htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim1.Init.RepetitionCounter = 0;

htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**if** (HAL\_TIM\_Base\_Init(&htim1) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

**if** (HAL\_TIM\_ConfigClockSource(&htim1, &sClockSourceConfig) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

}

/\* TIM2 init function \*/

/\* PWM Output\*/

**static** **void** **MX\_TIM2\_Init**(**void**)

{

TIM\_MasterConfigTypeDef sMasterConfig;

TIM\_OC\_InitTypeDef sConfigOC;

htim2.Instance = TIM2;

htim2.Init.Prescaler = 2;

htim2.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim2.Init.Period = 4499;

htim2.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim2.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**if** (HAL\_TIM\_PWM\_Init(&htim2) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (HAL\_TIMEx\_MasterConfigSynchronization(&htim2, &sMasterConfig) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sConfigOC.OCMode = TIM\_OCMODE\_PWM1;

sConfigOC.Pulse = 400;

sConfigOC.OCPolarity = TIM\_OCPOLARITY\_HIGH;

sConfigOC.OCFastMode = TIM\_OCFAST\_DISABLE;

**if** (HAL\_TIM\_PWM\_ConfigChannel(&htim2, &sConfigOC, TIM\_CHANNEL\_1) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

HAL\_TIM\_MspPostInit(&htim2);

}

/\* TIM3 init function \*/

/\* Step Alternator \*/

**static** **void** **MX\_TIM3\_Init**(**void**)

{

TIM\_ClockConfigTypeDef sClockSourceConfig;

TIM\_MasterConfigTypeDef sMasterConfig;

htim3.Instance = TIM3;

htim3.Init.Prescaler = 255;

htim3.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim3.Init.Period = 48124;

htim3.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim3.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

**if** (HAL\_TIM\_Base\_Init(&htim3) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

**if** (HAL\_TIM\_ConfigClockSource(&htim3, &sClockSourceConfig) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_UPDATE;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

**if** (HAL\_TIMEx\_MasterConfigSynchronization(&htim3, &sMasterConfig) != *HAL\_OK*)

{

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

}

/\*\* Configure pins as

\* Analog

\* Input

\* Output

\* EVENT\_OUT

\* EXTI

\*/

**static** **void** **MX\_GPIO\_Init**(**void**)

{

GPIO\_InitTypeDef GPIO\_InitStruct;

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_13, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_12, *GPIO\_PIN\_RESET*);

/\*Configure GPIO pin : PC13 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_13;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

/\*Configure GPIO pin : PA12 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_12;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

**void** **HAL\_TIM\_PeriodElapsedCallback**(TIM\_HandleTypeDef \*htim){

**if**(htim->Instance == TIM1){

HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_12,*GPIO\_PIN\_SET*);

//HAL\_GPIO\_TogglePin(GPIOA,GPIO\_PIN\_12);

HAL\_ADC\_Start\_IT(&hadc1);

}

**if**(htim->Instance == TIM3){

**if**(setValue == 1363)

setValue = 2045;

**else**

setValue = 1363;

}

//HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_12,GPIO\_PIN\_SET);

}

float32\_t A[3][3] = {{-416.67, 416.67, 0}, {-355.39, 333.77, 21.626}, {0, 0, -6666.7}};

float32\_t B3 = 6666.7;

float32\_t K[4] = {-0.99525, 2.106, -0.86308, 106.05};

float32\_t L[3] = {13165, 115830, -2.5599};

float32\_t c, e, enew, i, inew, k, x;

float32\_t n1, n1new, n2, n2new, n3, n3new;

float32\_t m1, m1new, m2, m2new, m3, m3new;

float32\_t Ts = 0.0002;

int32\_t xx;

**void** **HAL\_ADC\_ConvCpltCallback**(ADC\_HandleTypeDef\* hadc) {

rawValue = HAL\_ADC\_GetValue(&hadc1);

c = (4499\*rawValue)/4095;

enew = ((float32\_t)setValue - c);

inew = (Ts\*e + i);

n1new = (Ts\*m1 + n1);

n2new = (Ts\*m2 + n2);

n3new = (Ts\*m3 + n3);

k = (K[0]\*n1new + K[1]\*n2new + K[2]\*n3new);

x = (inew\*K[3] - k);

xx = x;

**if**(x > 4499)

x = 4499;

**else** **if** (x < 0)

x = 0;

\_\_HAL\_TIM\_SET\_COMPARE(&htim2, TIM\_CHANNEL\_1, (uint32\_t)x);

m1 = ((c-n1new)\*L[0] + (n1new\*A[0][0]) + (n2new\*A[0][1]) + (n3new\*A[0][2]));

m2 = ((c-n1new)\*L[1] + (n1new\*A[1][0]) + (n2new\*A[1][1]) + (n3new\*A[1][2]));

m3 = ((c-n1new)\*L[2] + (n1new\*A[2][0]) + (n2new\*A[2][1]) + (n3new\*A[2][2])) + x\*B3;

e = enew;

i = inew;

n1 = n1new;

n2 = n2new;

n3 = n3new;

HAL\_GPIO\_WritePin(GPIOA,GPIO\_PIN\_12,*GPIO\_PIN\_RESET*);

}

# BIBLIOGRAFIA

OGATA, Katsuhiko. Discrete-time control systmens. 2.ed. New Jersey: Prentice Hall, 1995. 745p., il. ISBN 0-13-034281-5.