**IMT - CEUN**

**Escola de Engenharia Mauá**

**Documentação – Projeto Microcontroladores**

**Curso: Tronco de Engenharia Elétrica**

**Turno: Diurno**

**Disciplina: Microcontroladores**

**Prof.: Rafael Corsi**

**Autores**

**13.00946-0 Thales Pizzotti Costi**

**13.03869-9 Stéfany Mazon**

**S. Caetano do Sul**

**05/12/2016**

**SUMÁRIO**

[1. Introdução 3](#_Toc468739169)

[2. Objetivos 3](#_Toc468739170)

[3. Metodologia 3](#_Toc468739171)

[4. Resultados 4](#_Toc468739172)

[ANEXO A – CÓDIGO DO PROJETO 6](#_Toc468739173)

# Introdução

Foi utilizada a teoria aprendida na matéria de Microcontroladores para realizar o controle de um levitador eletromagnético, projeto estipulado na matéria de Sistemas de Controle. Para isso utilizou-se da plataforma da Atmel para realizar o desenvolvimento do controle por meio de exemplos fornecidos pelo próprio software.

O projeto do microcontrolador foi necessário para coordenar a aferição do sensor de infravermelho com o controle da fonte linear por meio de um sinal PWM.

# Objetivos

Realizar o controle em malha fechada de um levitador eletromagnético para equilibrar um objeto metálico com sua devida referência.

# Metodologia

Para realizar o trabalho acima descrito utilizou-se o software Atmel Studio 7.0. A metodologia do projeto consistiu em primeiramente a codificação do PWM e do AD separadamente, e posteriormente juntamos os códigos.

Para o AD, utilizamos como tensão de referência 3V3 e o pino de entrada utilizado foi o PA17. Para verificar o funcionamento do mesmo, criamos um desenho no LCD para demonstrar a variação da tensão, isto é, a medida que variamos a tensão de entrada, o valor variava no LCD.

Para o PWM, utilizou-se do exemplo do mesmo por meio de um led para posteriormente, configurá-lo em um PIO. O I/O selecionado foi o pino 16 do PORT A e a frequência do PWM foi 5,0kHz.

Após verificá-los individualmente, seguiu-se para o código final realizando as devidas modificações como:

* Configurar a frequência de amostragem em 1,0kHz por meio da interrupção do Timer Counter (TC).
* Utilizar o controle do PWM junto de sua equação de diferenças dentro da interrupção do conversor analógico/digital (ADC).

# Resultados

De modo geral os objetivos do projeto foram alcançados. Conseguiu-se integrar o PWM com o conversor AD para controlar o levitador. O único problema que não consolidou para o controle perfeito foi que o hardware foi desenvolvido para controle de tensão por meio do PWM e a equação de diferenças elaborada por outro grupo foi modelada por corrente.

Com esse projeto pudemos sumarizar tanto os conceitos relacionados à matéria de eletrônica embarcada quanto aos tópicos estudados ao longo do curso, com a implementação do hardware para atuação do controle.

Figura 1 – Placa de desenvolvimento da Atmel utilizada no curso e no projeto.

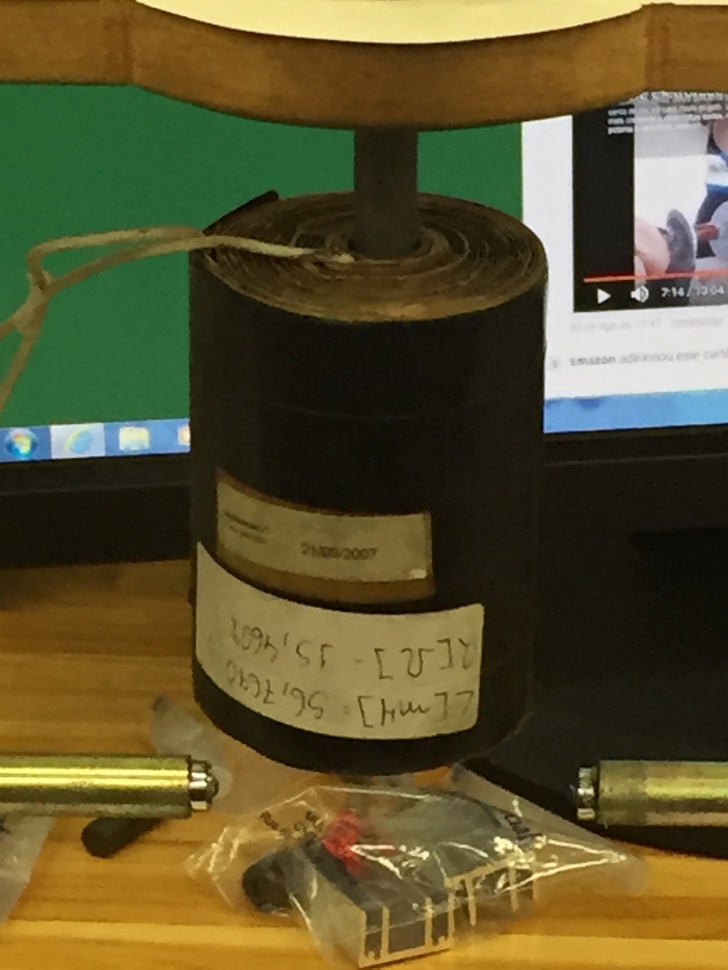


Figura 2 – Bobina e sensor de infravermelho utilizados no levitador.

# ANEXO A – CÓDIGO DO PROJETO

#include "asf.h"

#include "stdio\_serial.h"

#include "conf\_board.h"

#include "conf\_clock.h"

#include "smc.h"

#include "arm\_math.h"

/\*\* Chip select number to be set \*/

#define ILI93XX\_LCD\_CS 1

//AD

struct ili93xx\_opt\_t g\_ili93xx\_display\_opt;

#define pi 3.14926

#define PIN\_PUSHBUTTON\_1\_MASK PIO\_PB3

#define PIN\_PUSHBUTTON\_1\_PIO PIOB

#define PIN\_PUSHBUTTON\_1\_ID ID\_PIOB

#define PIN\_PUSHBUTTON\_1\_TYPE PIO\_INPUT

#define PIN\_PUSHBUTTON\_1\_ATTR PIO\_PULLUP | PIO\_DEBOUNCE | PIO\_IT\_FALL\_EDGE

#define PIN\_PUSHBUTTON\_2\_MASK PIO\_PC12

#define PIN\_PUSHBUTTON\_2\_PIO PIOC

#define PIN\_PUSHBUTTON\_2\_ID ID\_PIOC

#define PIN\_PUSHBUTTON\_2\_TYPE PIO\_INPUT

#define PIN\_PUSHBUTTON\_2\_ATTR PIO\_PULLUP | PIO\_DEBOUNCE | PIO\_IT\_FALL\_EDGE

//PWM

#define BOB\_PWM\_FREQ 10000

#define BOB\_PWM\_PERIOD 1000

#define BOB\_PWM\_CH 2

#define BOB\_PWM\_PIN 16

#define BOB\_PWM\_PIO PIOA

#define BOB\_PWM\_PIN\_MASK (1 << 16)

pwm\_channel\_t g\_pwm\_channel\_bob;

//AD

/\*\* Size of the receive buffer and transmit buffer. \*/

#define BUFFER\_SIZE (100)

/\*\* Reference voltage for ADC,in mv. \*/

#define VOLT\_REF (3300)

/\* Tracking Time\*/

#define TRACKING\_TIME 1

/\* Transfer Period \*/

#define TRANSFER\_PERIOD 1

/\* Startup Time\*/

#define STARTUP\_TIME ADC\_STARTUP\_TIME\_4

/\*\* The maximal digital value \*/

#define MAX\_DIGITAL (4095)

//PWM

/\*\* PWM frequency in Hz \*/

#define PWM\_FREQUENCY 5000

/\*\* Period value of PWM output waveform \*/

#define PERIOD\_VALUE 100

/\*\* Initial duty cycle value \*/

#define INIT\_DUTY\_VALUE 0

#define a1 0.67032

#define b1 0.0155161441837773

#define b2 -0.014710475073314

#define STRING\_EOL "\r"

#define STRING\_HEADER "-- PWM LED Example --\r\n" \

"-- "BOARD\_NAME" --\r\n" \

"-- Compiled: "\_\_DATE\_\_" "\_\_TIME\_\_" --"STRING\_EOL

/\*\* PWM channel instance for LEDs \*/

pwm\_channel\_t g\_pwm\_channel\_led;

//AD

/\*\* adc buffer \*/

static int16\_t gs\_s\_adc\_values[BUFFER\_SIZE] = { 0 };

#define PIN\_ADC\_IN 17

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* GLOBAL \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int adc\_value\_old;

float v;

int vet[30];

float rad;

float bob\_dutycycle;

float bob\_dutycycle\_ant = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* HANDLER \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void push\_button\_handle(uint32\_t id, uint32\_t mask)

{

adc\_start(ADC);

}

/\*\*

\* \brief ADC interrupt handler.

\*/

void ADC\_Handler(void)

{

uint32\_t tmp;

uint32\_t tmp\_0 = 0;

uint32\_t status ;

uint16\_t erro;

status = adc\_get\_status(ADC);

/\* Checa se a interrupção é devido ao canal 17 \*/

static float rad\_antes = 0;

tmp = adc\_get\_channel\_value(ADC, 0);

erro = +2606 - tmp;

ili93xx\_set\_foreground\_color(COLOR\_WHITE);

ili93xx\_draw\_filled\_rectangle(9, 39, ILI93XX\_LCD\_WIDTH,55);

v=3.3\*((float)tmp)/4095.0;

rad=2\*pi\*((float)tmp)/4095.0;

ili93xx\_draw\_line(120,160,120+54\*arm\_cos\_f32(rad\_antes),160+54\*arm\_sin\_f32(rad\_antes));

ili93xx\_set\_foreground\_color(COLOR\_BLACK);

sprintf(vet, "Tensao: %f", v);

ili93xx\_draw\_string(10, 40, vet);

ili93xx\_draw\_line(120,160,120+54\*arm\_cos\_f32(rad),160+54\*arm\_sin\_f32(rad));

rad\_antes = rad;

//PWM

bob\_dutycycle = 1000 -(350 + (a1\*bob\_dutycycle\_ant + b1\*erro + b2\*tmp\_0)/3);

pwm\_channel\_update\_duty(PWM, &g\_pwm\_channel\_bob.channel, bob\_dutycycle);

tmp\_0 = erro;

bob\_dutycycle\_ant = bob\_dutycycle;

}

void TC0\_Handler(void)

{ static b = 0;

volatile uint32\_t ul\_dummy;

ul\_dummy = tc\_get\_status(TC0,0);

adc\_start(ADC);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* CONFIGs \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void configure\_lcd()

{

/\*\* Enable peripheral clock \*/

pmc\_enable\_periph\_clk(ID\_SMC);

/\*\* Configure SMC interface for Lcd \*/

smc\_set\_setup\_timing(SMC, ILI93XX\_LCD\_CS, SMC\_SETUP\_NWE\_SETUP(2)

| SMC\_SETUP\_NCS\_WR\_SETUP(2)

| SMC\_SETUP\_NRD\_SETUP(2)

| SMC\_SETUP\_NCS\_RD\_SETUP(2));

smc\_set\_pulse\_timing(SMC, ILI93XX\_LCD\_CS, SMC\_PULSE\_NWE\_PULSE(4)

| SMC\_PULSE\_NCS\_WR\_PULSE(4)

| SMC\_PULSE\_NRD\_PULSE(10)

| SMC\_PULSE\_NCS\_RD\_PULSE(10));

smc\_set\_cycle\_timing(SMC, ILI93XX\_LCD\_CS, SMC\_CYCLE\_NWE\_CYCLE(10)

| SMC\_CYCLE\_NRD\_CYCLE(22));

#if ((!defined(SAM4S)) && (!defined(SAM4E)))

smc\_set\_mode(SMC, ILI93XX\_LCD\_CS, SMC\_MODE\_READ\_MODE

| SMC\_MODE\_WRITE\_MODE

| SMC\_MODE\_DBW\_8\_BIT);

#else

smc\_set\_mode(SMC, ILI93XX\_LCD\_CS, SMC\_MODE\_READ\_MODE

| SMC\_MODE\_WRITE\_MODE);

#endif

/\*\* Initialize display parameter \*/

g\_ili93xx\_display\_opt.ul\_width = ILI93XX\_LCD\_WIDTH;

g\_ili93xx\_display\_opt.ul\_height = ILI93XX\_LCD\_HEIGHT;

g\_ili93xx\_display\_opt.foreground\_color = COLOR\_BLACK;

g\_ili93xx\_display\_opt.background\_color = COLOR\_WHITE;

/\*\* Switch off backlight \*/

aat31xx\_disable\_backlight();

/\*\* Initialize LCD \*/

ili93xx\_init(&g\_ili93xx\_display\_opt);

/\*\* Set backlight level \*/

aat31xx\_set\_backlight(AAT31XX\_AVG\_BACKLIGHT\_LEVEL);

ili93xx\_set\_foreground\_color(COLOR\_WHITE);

ili93xx\_draw\_filled\_rectangle(0, 0, ILI93XX\_LCD\_WIDTH,

ILI93XX\_LCD\_HEIGHT);

/\*\* Turn on LCD \*/

ili93xx\_display\_on();

ili93xx\_set\_cursor\_position(0, 0);

}

//PWM

/\*\*

\* \brief Configure the Console UART.

\*/

static void configure\_console(void)

{

const usart\_serial\_options\_t uart\_serial\_options = {

.baudrate = CONF\_UART\_BAUDRATE,

#ifdef CONF\_UART\_CHAR\_LENGTH

.charlength = CONF\_UART\_CHAR\_LENGTH,

#endif

.paritytype = CONF\_UART\_PARITY,

#ifdef CONF\_UART\_STOP\_BITS

.stopbits = CONF\_UART\_STOP\_BITS,

#endif

};

/\* Configure console UART. \*/

sysclk\_enable\_peripheral\_clock(CONSOLE\_UART\_ID);

stdio\_serial\_init(CONF\_UART, &uart\_serial\_options);

}

void configure\_botao(void)

{

pmc\_enable\_periph\_clk(PIN\_PUSHBUTTON\_1\_ID);

pio\_set\_input(PIN\_PUSHBUTTON\_1\_PIO, PIN\_PUSHBUTTON\_1\_MASK, PIN\_PUSHBUTTON\_1\_ATTR);

pio\_set\_debounce\_filter(PIN\_PUSHBUTTON\_1\_PIO, PIN\_PUSHBUTTON\_1\_MASK, 10);

pio\_handler\_set(PIN\_PUSHBUTTON\_1\_PIO, PIN\_PUSHBUTTON\_1\_ID,PIN\_PUSHBUTTON\_1\_MASK, PIN\_PUSHBUTTON\_1\_ATTR ,push\_button\_handle);

pio\_enable\_interrupt(PIN\_PUSHBUTTON\_1\_PIO, PIN\_PUSHBUTTON\_1\_MASK);

NVIC\_SetPriority((IRQn\_Type) PIN\_PUSHBUTTON\_1\_ID, 0);

NVIC\_EnableIRQ((IRQn\_Type) PIN\_PUSHBUTTON\_1\_ID);

}

static void configure\_tc(void)

{

uint32\_t ul\_sysclk = sysclk\_get\_cpu\_hz();

pmc\_enable\_periph\_clk(ID\_TC0);

tc\_init(TC0, 0, TC\_CMR\_CPCTRG | TC\_CMR\_TCCLKS\_TIMER\_CLOCK5);

tc\_write\_rc(TC0,0,32768\*0.001);

tc\_enable\_interrupt(TC0,0,TC\_IER\_CPCS);

NVIC\_EnableIRQ(ID\_TC0);

tc\_start(TC0,0);

}

void configure\_adc(void)

{

pmc\_enable\_periph\_clk(ID\_ADC);

adc\_init(ADC, sysclk\_get\_cpu\_hz(), 6400000, STARTUP\_TIME);

adc\_configure\_timing(ADC, TRACKING\_TIME , ADC\_SETTLING\_TIME\_3, TRANSFER\_PERIOD);

adc\_configure\_trigger(ADC, ADC\_TRIG\_SW, 0);

adc\_enable\_channel(ADC, 0);

adc\_enable\_ts(ADC);

NVIC\_EnableIRQ(ADC\_IRQn);

adc\_start(ADC);

adc\_enable\_interrupt(ADC, ADC\_ISR\_EOC0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* MAIN \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)

{

sysclk\_init();

board\_init();

//AD

configure\_lcd();

configure\_botao();

// PWM

/\* Configure the console uart for debug information \*/

configure\_console();

/\* Output example information \*/

puts(STRING\_HEADER);

/\* Configure PIOA16 \*/

pmc\_enable\_periph\_clk(ID\_PIOA);

gpio\_configure\_pin(PIO\_PA16\_IDX, PIO\_PERIPH\_C | PIO\_DEFAULT);

/\* Enable PWM peripheral clock \*/

pmc\_enable\_periph\_clk(ID\_PWM);

/\* Disable PWM channels for LEDs \*/

pwm\_channel\_disable(PWM, BOB\_PWM\_CH);

/\* Set PWM clock A as PWM\_FREQUENCY\*PERIOD\_VALUE (clock B is not used) \*/

pwm\_clock\_t clock\_setting = {

.ul\_clka = BOB\_PWM\_FREQ \* BOB\_PWM\_PERIOD,

.ul\_clkb = 0,

.ul\_mck = sysclk\_get\_cpu\_hz()

};

int bob\_pwm\_prescale;

bob\_pwm\_prescale = pwm\_init(PWM, &clock\_setting);

/\* Initialize PWM channel for Bobina \*/

/\* Period is center-aligned \*/

g\_pwm\_channel\_bob.alignment = PWM\_ALIGN\_CENTER;

/\* Output waveform starts at a high level \*/

g\_pwm\_channel\_bob.polarity = PWM\_HIGH;

/\* Use PWM clock A as source clock \*/

g\_pwm\_channel\_bob.ul\_prescaler = PWM\_CMR\_CPRE\_CLKA;

/\* Period value of output waveform \*/

g\_pwm\_channel\_bob.ul\_period = BOB\_PWM\_PERIOD;

/\* Duty cycle value of output waveform \*/

g\_pwm\_channel\_bob.ul\_duty = 50;

g\_pwm\_channel\_bob.channel = BOB\_PWM\_CH;

// configura CPRD

pwm\_channel\_init(PWM, &g\_pwm\_channel\_bob);

/\* Disable channel counter event interrupt \*/

pwm\_channel\_disable\_interrupt(PWM, BOB\_PWM\_CH, 0);

NVIC\_DisableIRQ(PWM\_IRQn);

NVIC\_ClearPendingIRQ(PWM\_IRQn);

NVIC\_SetPriority(PWM\_IRQn, 0);

NVIC\_EnableIRQ(PWM\_IRQn);

/\* Enable PWM channels for LEDs \*/

pwm\_channel\_enable(PWM, BOB\_PWM\_CH);

PIOA->PIO\_ODR = (1 << 17) ;

configure\_adc();

configure\_tc();

/\*\* Draw text, image and basic shapes on the LCD \*/

ili93xx\_set\_foreground\_color(COLOR\_BLACK);

ili93xx\_draw\_string(10, 20, (uint8\_t \*)"14 - ADC");

ili93xx\_draw\_filled\_circle(120,160,60);

ili93xx\_set\_foreground\_color(COLOR\_WHITE);

ili93xx\_draw\_filled\_circle(120,160,55);

}