

# **Universidade de Brasília**



## **Projeto final Eletrônica Embarcada**

### **Autores:**

Lucas dos Santos Barros de Sousa - 18/0022555

Matheus Oliveira Dias - 18/0025104

Victor Hugo Ciurlini 11/0021223

Brasília  
17 de dezembro de 2020

# Conteúdo

<b>1</b>	<b>Versão Completa</b>	<b>2</b>
1.1	Especificações do sistema . . . . .	2
1.2	Componentes . . . . .	4
1.3	Planejamento e solução . . . . .	5
1.3.1	Comunicação . . . . .	5
1.3.2	Gerenciamento . . . . .	5
1.3.3	Controle . . . . .	5
1.4	Funcionamento e simulações . . . . .	8
1.4.1	definição de periféricos e pinos . . . . .	8
1.4.2	Simulações e resultados . . . . .	9
1.5	Esquemático . . . . .	19
1.6	Diagrama de Fluxo . . . . .	21
<b>2</b>	<b>Versão Simplificada</b>	<b>22</b>
2.1	Novas especificações do sistema . . . . .	22
2.2	Funcionamento e simulações . . . . .	23
2.2.1	Fluxograma . . . . .	25
<b>3</b>	<b>Considerações Finais</b>	<b>26</b>

# 1 Versão Completa

## 1.1 Especificações do sistema

Dado um projeto, é necessário atender a alguns requisitos solicitados pelo cliente. neste caso, o cliente trata-se do professor da matéria. O produto final trata-se de um firmware responsável por controlar um Elevador para um edifício de 4 andares, controlado por um microcontrolador PIC16F1827, alimentado com 5V e trabalhando a uma frequência de 8 MHz. Esse controlador atua em um motor de corrente contínua, responsável pelo movimento de subida, descida e parada do elevador. Os requisitos do sistema são descritos no relatório, sendo eles:

- O elevador deve subir e descer quando solicitado;
- O controlador é responsável por otimizar o percurso do elevador;
- O controlador limita a velocidade máxima, a aceleração, a desaceleração e o tempo de parada do elevador;
- O controlador realiza a leitura de parâmetros, como corrente e temperatura do motor;
- A medição da corrente deverá ser sempre positiva, independente do sentido do elevador;
- O valor medido da corrente deve ser convertido em unidade de engenharia (mA);
- Caso a corrente ultrapassar um valor máximo pré-definido, o controlador deve desligar o motor. Isso ocorre caso o elevador ficar travado em um dos extremos do percurso ou a tensão de alimentação fornecida ficar muito baixa;
- O controlador fará a leitura do andar que se encontra o elevador através de sensores em cada andar, ativado por um ímã posicionado no elevador;
- A resposta dos sensores deve ser do tipo coletores abertos;
- Os sensores devem ser atendidos por interrupções;
- A velocidade máxima do elevador não pode ultrapassar 20 mm/s;

- Ao se aproximar do andar, o elevador deverá reduzir sua velocidade gradualmente e manter uma velocidade constante de 5 mm/s até sua parada total;
- O controlador enviará informações acerca da velocidade do elevador para dois LEDs, um vermelho (em movimento) e um verde (parado)

## 1.2 Componentes

Para que seja possível a validação do firmware, foi desenvolvido pelo professor uma maquete do elevador onde será gravado em um controlador PIC16F1827 o algoritmo e testado utilizando uma lista de estímulos. Os componentes são:

Componente	Quantidade
PIC16F1827-I / P	1
<b>AmpOP</b>	
LM7805CT	1
LM358N	3
<b>Relé</b>	
SRD-S-105D	2
<b>Diodo</b>	
1N4148	3
<b>AmpTRANS</b>	
BC337	2
BC548	1
<b>MOSFET</b>	
IRF9540	1
<b>Capacitores</b>	
100nF	1
1uF/6.3V	1
100uF/16v	1
47uf/6.3v	1
<b>Resistores</b>	
0.47/2W	1
150/250mW	2
220/250mW	7
1K/250mW	3
2K2/250mW	1
4K7/250mW	1
10K/250mW	1
22K/250mW	1
82K/250mW	3
<b>SENSITIVE HALL-EFFECT SWITCHES</b>	
A3144	4
<b>USB-TO-UART BRIDGE</b>	
CP2102	1

Optical Sensor	1
<b>LED</b>	
RED	1
GREEN	1
<b>Sensores</b>	
LM35DZ/NOPB	1
Optical Sensor	1
<b>Motor</b>	
Motor 3-6V, 100-200rpm, 260-470mA	1

### 1.3 Planejamento e solução

Ao estudar os requisitos apresentados, foi definido que a solução seria dividida em três macros diferentes:

- Comunicação;
- Gerenciamento;
- Controle;

Cada área está integrada com as outras, tendo entre si trocas de informações e parâmetros.

#### 1.3.1 Comunicação

Área responsável pela comunicação usuários - controlador e vice versa. Essa área é responsável por receber os dados dos usuários e transmitir esses dados para as demais funções. A comunicação também é responsável em transmitir ao usuário informações como a temperatura e a corrente que alimenta o motor. Essa comunicação é realizada através de 4 bytes, como solicitado no roteiro, e a uma velocidade de 19200 bits por segundo.

#### 1.3.2 Gerenciamento

#### 1.3.3 Controle

A área de controle é responsável única e exclusivamente pelo controle da velocidade do motor. Dados os requisitos do sistema descritos no primeiro capítulo, foi fornecido também a forma de onda que representa como o motor deve se comportar ao:

- Se afastar do andar de origem;

- No percurso até o destino final;
- Ao se aproximar do andar de destino;

Dado essas descrições, temos:

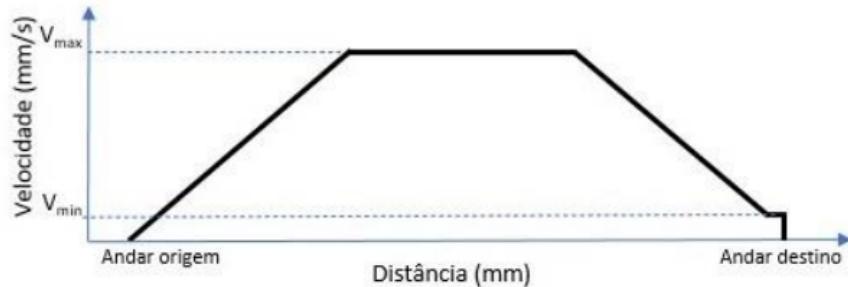


Figura 1: Forma de onda, controle da velocidade do elevador

Para atender as especificações, foi definido algumas variáveis a serem calculadas considerando as informações disponibilizadas:

- o **Enc** fornece 20 pulsos a cada 15mm de deslocamento;
- a altura de cada andar individualmente é de 60mm;
- a velocidade deve aumentar de forma gradual, implicando a necessidade de uma aceleração constante;
- Quando atingido a velocidade máxima, o elevador se move a 20mm/s;
- Ao se aproximar do andar de origem, o elevador desacelera com velocidade constante e, ao atingir uma velocidade de 5mm/s, o elevador deve manter uma velocidade constante até sua parada total.
- a parada total é indicada por um sensor de presença do tipo coletor ativo, como citado nas especificações do sistema.

Com essas informações, foi definido intervalos de distância que atendiam aos requisitos para cada uma das etapas (aceleração, percurso, desaceleração, parada) e com esses dados foi possível definir o valor da aceleração constante utilizando a equação de Torricelli.

Foi definido que o Motor irá ter uma aceleração constante de 0 a 15mm do ponto de origem, mantendo sua velocidade constante pelo percurso até a distância de desaceleração e parada. O motor terá sua desaceleração iniciada a 15mm do andar de destino, sendo desses 15mm, 11 em desaceleração

constante e 4mm em velocidade constante igual a 5mm/s. Foi definido que a aceleração seria de  $52\text{mm/s}^2$ .

Cada Área foi designada para um integrante, porém não limitando o mesmo á aquela área, Quando necessário, houve interação dos membros para desenvolver as soluções que demandavam mais tempo e esforço. A divisão das tarefas ficou da seguinte forma:

- Comunicação: Lucas dos Santos Barros de Sousa
- Gerenciamento: Matheus Oliveira Dias
- Controle: Victor Hugo Ciurlini

Foi definido para o desenvolvimento do projeto que seria usado o gerenciador de repositórios GITHUB em conjunto com a ferramenta GitKraken, afim de facilitar a paralelização de tarefas.



## 1.4 Funcionamento e simulações

### 1.4.1 definição de periféricos e pinos

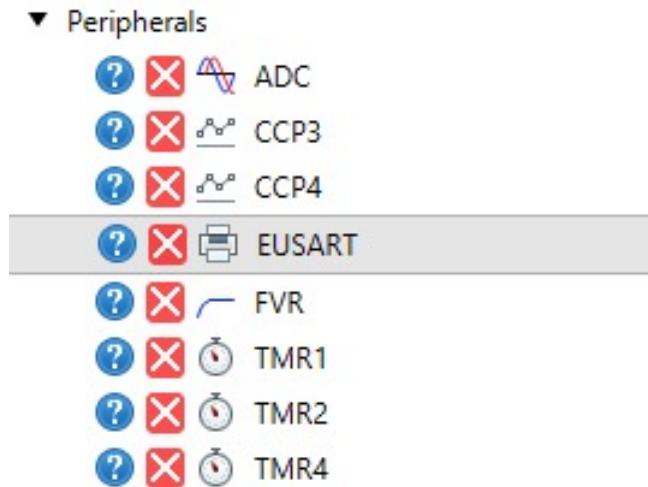


Figura 2: Periféricos utilizados

Os periféricos acima foram definidos a partir das necessidades para a solução.

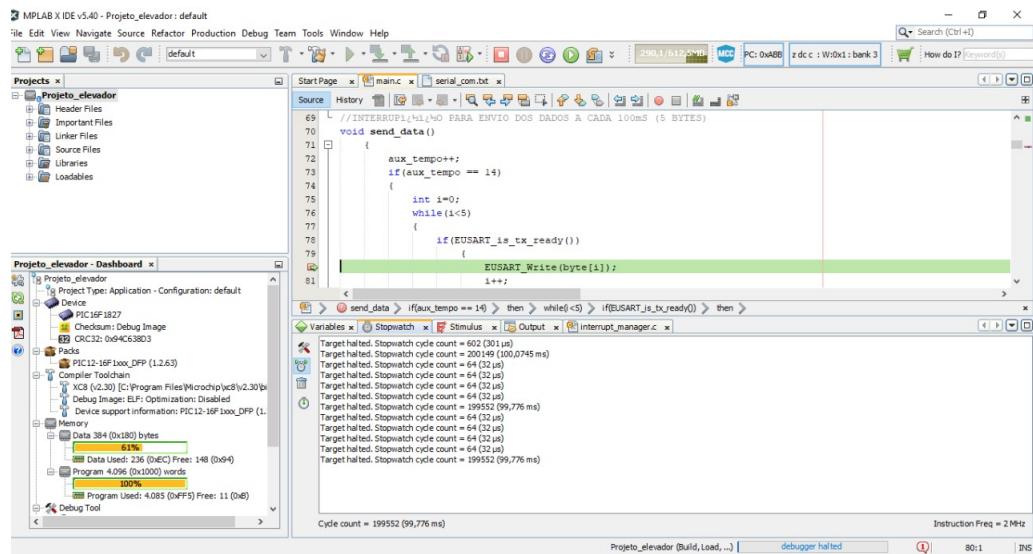
- ADC - Utilizado para mensurar a temperatura e a corrente no motor;
- CCP3 - Modo PWM, de 10 bits. Responsável pelo controle do motor;
- CCP4 - Modo de captura. Responsável por mensurar velocidade do motor;
- TMR1/TMR2 - Associados ao CCP3/CCP4. Garantem o funcionamento dos CCPs;
- EUSART - Recebimento e transmissão dos dados;
- FVR - será utilizado uma tensão de referência, pois a temperatura e a corrente estão contidos entre 0 a 1v. Para melhorar a precisão, usamos uma tensão de 1.024v;
- TMR4 - Responsável pelas interrupções e envio dos dados ;

Os pinos foram pré-definidos no relatório, de modo que, utilizando o MCC do mplab, sua configuração fica da seguinte forma:

Pin Module										
Easy Setup		Registers		Selected Package : SOIC18						
Pin Name	Module	Function	Custom Na...	Start High	Analog	Output	WPU	OD	IOC	
RA0	ADC	AN0	Im	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
RA1	ADC	AN1	Temp	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
RA2	Pin Module	GPIO	Dir	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>				
RA3	CCP3	CCP3		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>				
RA4	CCP4	CCP4		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
RA6	Pin Module	GPIO	LedG	<input type="checkbox"/>		<input checked="" type="checkbox"/>				
RA7	Pin Module	GPIO	LedR	<input type="checkbox"/>		<input checked="" type="checkbox"/>				
RB0	Pin Module	GPIO	S1	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>			<input type="button" value="ne..."/>
RB1	EUSART	RX		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="button" value="none"/>
RB2	EUSART	TX		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="button" value="none"/>
RB3	Pin Module	GPIO	S2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="button" value="ne..."/>
RB4	Pin Module	GPIO	S3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="button" value="ne..."/>
RB5	Pin Module	GPIO	S4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="button" value="ne..."/>

Figura 3: Configuração dos pinos

#### 1.4.2 Simulações e resultados



Na simulação acima é possível ver a taxa de transmissão de bytes. A informação é transmitida a uma taxa de 100 ms, cada byte demora, em média,  $32\mu s$ .

The screenshot shows the MPLAB X IDE interface with the following details:

- Projects:** Projeto\_elevador
- Source Editor:** main.c, serial\_com.txt
- Code Snippet:**

```

182 }
183 }
184 void controle()
185 {
186     I_m = conv_I*ADC_GetConversion(0); // Realiza a leitura da corrente, converte para mA usando *conv
187     temp_mt = conv_temp*ADC_GetConversion(1); // Realiza a leitura da temperatura, converte para C usando *co
188     int count = 0;
189     int a = 52;
190     if (temp_mt > max_dutyValue) { // Caso sim, esse valor é substituído pelo valor máximo
191         max_dutyValue = 1023; // Valor da aceleração máxima para o motor (ci)cálculado por to
192         int min_dutyValue = 256; // Valor da aceleração mínima para o motor (ci)cálculado por to
193         int destiny = distancia*60; // Valor do FWD para v = 5 mm/s
194         int route = destiny - 40; // Valor do BACK para v = 5 mm/s
195         int dutyValue = 0; // Inicia dutyValue em zero
196     }
197 }
```
- Variables View:** Shows memory dump for variables like RCREG, TXREG, and dutyValue.
- Dashboard:** Shows project statistics: Data 384 (0x00) bytes used (61%), Program 4.096 (0x1000) words used (100%), and Program 4.085 (0xFFFF) Free (11 (0xb)).

Valor da corrente e temperatura sendo transmitidos, já convertidos, em formato float.

The screenshot shows the MPLAB X IDE interface with the following details:

- Projects:** Projeto\_elevador
- Source Editor:** main.c, serial\_com.txt
- Code Snippet:**

```

197 for(count = 0; count < 20; count++) { // Loop responsável pela aceleração do motor
198     //pulse+=1;
199     dutyValue+=a; // Adiciona valor de aceleração ao dutyValue
200     if(dutyValue > max_dutyValue) { // Compara se o valor de dutyValue não ultrapassou o máximo
201         PWN3_LoadDutyValue(max_dutyValue); // Caso sim, esse valor é substituído pelo valor máximo
202     } else {
203         PWN3_LoadDutyValue(dutyValue); // Envia dutyValue com o valor acrescido de aceleração do motor
204     }
205 }
206 count = 0;
207 for(count = 0; count < route ; count++) { // Loop responsável pela velocidade máxima constante
208     //pulse+=1;
209     PWN3_LoadDutyValue(max_dutyValue); // Envia max_dutyValue para o PWM
210 }
211
212 count = 0; // Reinicia o contador para o próximo loop
213 }
```
- Variables View:** Shows memory dump for variables like porta, INTERRUPT\_InterruptManager, PWN3\_LoadDutyValue, RCREG, and dutyValue.
- Dashboard:** Shows project statistics: Data 384 (0x00) bytes used (61%), Program 4.096 (0x1000) words used (100%), and Program 4.085 (0xFFFF) Free (11 (0xb)).

The screenshot shows the MPLAB X IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Production, Debug, Team, Tools, Window, Help. The title bar says "MPLAB X IDE v5.40 - Projeto\_elevador : default". The left sidebar shows the "Projects" tree with "Projeto\_elevador" selected, containing Header Files, Important Files, Unlink Files, Source Files, Libraries, and Loadables. The main workspace has two tabs: "main.c" and "serial\_com.txt". The "main.c" tab displays C code for a PWM module, specifically handling dutyValue and PWM3\_LoadDutyValue. The "Variables" tab on the right shows memory dump information for variables like porta, INTERRUPT\_Manager, PWM3\_LoadDutyValue, RCREG, TIREG, and\_dst, flia\_up, and dutyValue. The status bar at the bottom indicates "Projeto\_elevador [Build, Load, ...] debugger halted" and "200:1 IN5".

É possível ver nas simulações acima a variação do dutyValue, responsável pela velocidade do motor, variando de forma constante. Tal processo se repete para a rotina de desaceleração.

This screenshot shows the same MPLAB X IDE interface as the previous one, but with a different project configuration. The title bar now says "364/1/6129MB HPC: 0x0B2 rDC.C : W:0x01:bank 0". The "Variables" tab in the bottom right shows updated values for variables like porta, INTERRUPT\_Manager, PWM3\_LoadDutyValue, RCREG, TIREG, and\_dst, flia\_up, and dutyValue. The status bar at the bottom indicates "Projeto\_elevador [Build, Load, ...] debugger halted" and "80:1 INS".

The screenshot shows the MPLAB X IDE interface during a debug session. The top window displays the source code for `main.c` and `pwm3.c`. The bottom window shows the memory dump for the serial communication test file `serial_com.bx`. The memory dump table has columns for Name, Type, Address, and Value.

Name	Type	Address	Value
<code>porta</code>	SFR	0x4C	00100100
<code>INTERUPT_InterruptManager</code>	function	0x4	0xd7E
<code>PWM3_LoadDutyValue</code>	function	0x978	8480
<code>RCREG</code>	SFR	0x199	0x02
<code>TIREG</code>	SFR	0x19A	10011110
<code>fila_up</code>	int[10]	0x84	Out of Scope
<code>dutyValue</code>		0x6F	00000000
<code>and_atng</code>	unsigned char	0x159	20.4
<code>t_m</code>	float	0x155	61.2
<code>temp_mnt</code>	float	0x154	0.0
<code>and_origem</code>	int	0x10A	0x0000
<code>and_dst</code>	int	0x74	0x0002

Recebimento dos dados. O arquivo de simulação serial está enviando o valor hexadecimal de 0,2 pela porta serial. Nesse momento o registrador REG armazena esse valor.

The screenshot shows the MPLAB X IDE interface during a debug session. The top window displays the source code for `main.c` and `pwm3.c`. The bottom window shows the memory dump for the serial communication test file `serial_com.bx`. The memory dump table has columns for Name, Type, Address, and Value.

Name	Type	Address	Value
<code>porta</code>	SFR	0x4C	00100100
<code>INTERUPT_InterruptManager</code>	function	0x4	0xd7E
<code>PWM3_LoadDutyValue</code>	function	0x978	8480
<code>RCREG</code>	SFR	0x199	0x02
<code>TIREG</code>	SFR	0x19A	10011110
<code>fila_up</code>	int[10]	0x84	Out of Scope
<code>dutyValue</code>		0x6F	00000000
<code>and_atng</code>	unsigned char	0x159	00000000
<code>t_m</code>	float	0x155	0.0
<code>temp_mnt</code>	float	0x154	0.0
<code>and_origem</code>	int	0x10A	0x0000
<code>and_dst</code>	int	0x74	0x0002
<code>fila_down</code>	int[10]	0x134	0.0
<code>byte[0]</code>	unsigned char[5]	0x148	"\0\0\0\0\0"
<code>byte[1]</code>	unsigned char	0x148	00000000
<code>buf[0]</code>	unsigned char	0x149	00000000

MPLAB X IDE v5.40 - Projeto\_elevator : default

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

423.6 612.5 MB PC: 0xAC4 z dc c : W:0x00 : bank 3 How do I [keywords]

**Projects** Projeto\_elevator

**Variables**

Name	Type	Address	Value
PWM3_LoadDutyValue	function	0x978	8480
RCREG	SFR	0x199	0x02
TIREG	SFR	0x19A	10000000
fila_up	int[10]	0x84	Out of Scope
dutyValue	unsigned char	0x6F	00000000
and_atng	unsigned char	0x159	20.4
t_m	float	0x155	61.2
temp_mnt	float	0x15A	0x0002
and_origem	int	0x0A	0x0002
and_dst	int	0x74	0x0002
fila_down	int[10]	0x134	Out of Scope
byte	unsigned char[5]	0x148	"\0\0\0\0\0"
byte[0]	unsigned char	0x148	00000000
byte[1]	unsigned char	0x149	10000000
byte[2]	unsigned char	0x14A	10000000
byte[3]	unsigned char	0x14B	10000000
aux_tempo	int	0x14C	10000000

Projeto\_elevator - Dashboard

Project Type: Application - Configuration: default

Device PIC16F1827

Compiler Toolchain XC8 (v2.30) [C:\Program Files\Microchip\XC8\2.30\bin]

Memory Data 384 (0x180) bytes

Program Used: 236 (0x0EC) Free: 148 (0x94)

Program 4.096 (0x1000) words

Program Used: 4.085 (0xFFF5) Free: 11 (0x0B)

Debug Tool

Source History Output interrupt\_manager.c tmr4.c xtoff.c Umu8\_16.c sprmul.c

serial\_com.bx

```

1 // Arquivo para teste da comunicação serial
2 wait 10 ms
3
4 02
5
6 //wait 100 ms
7
8 0B

```

Projeto\_elevator [Build, Load, ...] debugger halted 81:1 INS

MPLAB X IDE v5.40 - Projeto\_elevator : default

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

410.1 612.5 MB PC: 0x8B z dc c : W:0x1 : bank 3 How do I [keywords]

**Projects** Projeto\_elevator

**Variables**

Name	Type	Address	Value
RCREG	SFR	0x199	0x02
TIREG	SFR	0x19A	10000000
fila_up	int[10]	0x84	Out of Scope
dutyValue	unsigned char	0x6F	00000000
and_atng	unsigned char	0x159	20.4
t_m	float	0x155	61.2
temp_mnt	float	0x15A	0x0002
and_origem	int	0x0A	0x0002
and_dst	int	0x74	0x0002
fila_down	int[10]	0x134	Out of Scope
byte	unsigned char[5]	0x148	"\0\0\0\0\0"
byte[0]	unsigned char	0x148	00000000
byte[1]	unsigned char	0x149	10000000
byte[2]	unsigned char	0x14A	10000000
byte[3]	unsigned char	0x14B	10000000
aux_tempo	int	0x14C	10000000

Projeto\_elevator - Dashboard

Project Type: Application - Configuration: default

Device PIC16F1827

Compiler Toolchain XC8 (v2.30) [C:\Program Files\Microchip\XC8\2.30\bin]

Memory Data 384 (0x180) bytes

Program Used: 236 (0x0EC) Free: 148 (0x94)

Program 4.096 (0x1000) words

Program Used: 4.085 (0xFFF5) Free: 11 (0x0B)

Debug Tool

Source History Output interrupt\_manager.c tmr4.c xtoff.c Umu8\_16.c sprmul.c

serial\_com.bx

```

1 // Arquivo para teste da comunicação serial
2 wait 10 ms
3
4 02
5
6 //wait 100 ms
7
8 0B

```

Projeto\_elevator [Build, Load, ...] debugger halted 80:1 INS

MPLAB X IDE v5.40 - Projeto\_elevator : default

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

Projects x Projeto\_elevator

Source History PWM3.c x serial\_com.btx

```

1 // Arquivo para teste da comunicação serial
2 wait 10 ms
3
4 02
5
6 //wait 100 ms

```

Variables x Stopwatch x Stimulus x Output x interrupt\_manager.c x tmr4.c x xotof.c x Umul8\_16.c x sprcmul.c x

Name	Type	Address	Value
RCREG	SFR	0x199	0x02
TIREG	SFR	0x19A	10011110
fila_up	int[10]	0x84	
dutyValue	unsigned char	0x6F	Out of Scope
and_atng	float	0x159	00000100
l_m	float	0x155	20.4
tempo_mt	float	0x15D	61.2
and_origem	int	0x0A	0x0002
and_dst	int	0x74	0x0002
fila_down	int[10]	0x134	
byte	unsigned char[5]	0x148	"\0\0\080\0080\0080\009e"
byte[0]	unsigned char	0x148	00000000
byte[1]	unsigned char	0x149	10000000
byte[2]	unsigned char	0x14A	00000000
byte[3]	unsigned char	0x14B	10001010
byte[4]	unsigned char	0x14C	10011110
	int	0x79	3

MPLAB X IDE v5.40 - Projeto\_elevator : default

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

Projects x Projeto\_elevator

Source History PWM3.c x serial\_com.btx

```

1 // Arquivo para teste da comunicação serial
2 wait 10 ms
3
4 02
5
6 //wait 100 ms

```

Variables x Stopwatch x Stimulus x Output x interrupt\_manager.c x tmr4.c x xotof.c x Umul8\_16.c x sprcmul.c x

Name	Type	Address	Value
RCREG	SFR	0x199	0x02
TIREG	SFR	0x19A	10011110
fila_up	int[10]	0x84	
dutyValue	unsigned char	0x6F	Out of Scope
and_atng	float	0x159	00000100
l_m	float	0x155	20.4
tempo_mt	float	0x15D	61.2
and_origem	int	0x0A	0x0002
and_dst	int	0x74	0x0002
fila_down	int[10]	0x134	
byte	unsigned char[5]	0x148	"\0\0\080\0080\0080\009e"
byte[0]	unsigned char	0x148	00000000
byte[1]	unsigned char	0x149	10000000
byte[2]	unsigned char	0x14A	00000000
byte[3]	unsigned char	0x14B	10001010
byte[4]	unsigned char	0x14C	10011110
	int	0x79	4

As simulações acima demonstram a transmissão de cada um dos 5 bytes. O vetor de bytes uint8\_t, está sendo transmitido de maneira serial, o registro de transmissão equivalente ao registro byte, posição a cada interação.

The screenshot shows the MPLAB X IDE interface with the project 'Projeto\_elevador' open. The Variables tab in the debugger window is active, displaying memory dump information. The table shows memory addresses from 0x120 to 0x12E, with the value '2' at address 0x120. Other variables like origem\_up[0] through origem\_up[7] and and\_arq[0] through and\_arq[7] have values of 0. The status bar at the bottom indicates 'debugger halted'.

A imagem acima mostra a leitura da porta serial enviada pela comunicação com o valor igual a 00000110b, atribuindo a variável andar de origem sendo o 1 e o andar de destino sendo o andar 2.

This screenshot is identical to the one above, showing the MPLAB X IDE interface with the project 'Projeto\_elevador'. The Variables tab is active, displaying memory dump information. The value '2' is still present at address 0x120, indicating the correct interpretation of the serial data received.

Em relação ao armazenamento dos andares no vetor de fila, pode-se notar que como o andar 2 é o menor para a subida, ele já foi retirado e o seu valor foi atribuído à variável min\_origem\_up, responsável por indicar para onde o elevador deve seguir.

MPLAB X IDE v5.35 - Projeto\_elevador : default

Projects x Projeto\_elevador

Source History Default Output Start Page main.c pin\_manager.c interrupt\_manager.c abs.c

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

Search (Ctrl+F)

332/6576,1MB PIC:0x82C z DC C : W:0x29 : bank 0 How do I? keyword(s)

void sensor1() //trata interrupção do primeiro sensor
 {
 and\_ating=1; //atualiza variável usada no gerenciamento (de 1 a 4)
 and\_ating2=0; //atualiza variável usada na comunicação (de 0 a 3)
 }
}

void sensor2() //trata interrupção do segundo sensor

sensor1

Stimulus Variables Call Stack Breakpoints

Name	Type	Address	Value
temp			Out of Scope
aux			Out of Scope
sensor1	function	0x829	0xEF20
sensor2	function	0x8AB	0xF702
and_dst	int	0x74	2
and_ating	signed char	0x74F	1
fla_up	int[8]	0x80	
fla_up[0]	int	0x80	3
fla_up[1]	int	0x82	0
fla_up[2]	int	0x84	0
fla_up[3]	int	0x86	0
fla_up[4]	int	0x88	0
fla_up[5]	int	0x8A	0
fla_up[6]	int	0x8C	0
fla_up[7]	int	0x8E	0

Projeto\_elevador x Navigator

Projeto\_elevador

Device PIC16F1827

Checksum: Debug In CRC32: Hex file una

Packs

User program stopped

Digite aqui para pesquisar

Projeto\_elevador (Build, Load, ...) | 92:1 debugger halted | POR PTB2 13/12/2020 | 13:14

Ao ativar o sensor, ativa-se a interrupção atribuindo os valores de controle e de comunicação, cada qual com o inteiro indicando o andar associado ao sensor. Neste caso, o andar atingido recebe o valor 1.

MPLAB X IDE v5.35 - Projeto\_elevador : default

Projects x Projeto\_elevador

Source History Default Output Start Page main.c pin\_manager.c interrupt\_manager.c abs.c

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

Search (Ctrl+F)

451/2576,1MB PIC:0x800 z dc c : W:0x2 : bank 0 How do I? keyword(s)

void sensor2() //trata interrupção do segundo sensor
 {
 and\_ating=2; //atualiza variável usada no gerenciamento (de 1 a 4)
 and\_ating2=1; //atualiza variável usada na comunicação (de 0 a 3)
 }
}

void sensor3() //trata interrupção do terceiro sensor

sensor2

Stimulus Variables Call Stack Breakpoints

Name	Type	Address	Value
temp			Out of Scope
aux			Out of Scope
sensor1	function	0x829	0xEF20
sensor2	function	0x8AB	0xF702
and_dst	int	0x74	2
and_ating	signed char	0x74F	2
fla_up	int[8]	0x80	
fla_up[0]	int	0x80	3
fla_up[1]	int	0x82	0
fla_up[2]	int	0x84	0
fla_up[3]	int	0x86	0
fla_up[4]	int	0x88	0
fla_up[5]	int	0x8A	0
fla_up[6]	int	0x8C	0
fla_up[7]	int	0x8E	0

Projeto\_elevador x sensor20 ...

Projeto\_elevador

Device PIC16F1827

Checksum: Debug In CRC32: Hex file una

Packs

User program stopped

Digite aqui para pesquisar

Projeto\_elevador (Build, Load, ...) | 98:1 debugger halted | POR PTB2 13/12/2020 | 13:15

Sensor 2 é acionado e o andar atingido recebe o valor 2, que é o andar de origem, fazendo com que o elevador pare e a variável de valor mínimo de subida é atualizada.

The screenshot shows the MPLAB X IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Production, Debug, Team, Tools, Window, Help. The status bar at the bottom right shows 'debugger halted', '282:1', 'IN5', 'POR PTB2 13/12/2020'. The main window has tabs for Output, StartPage, main.c, pin\_manager.c, interrupt\_manager.c, abs.c. The Projects pane shows 'Projeto\_elevador' with files Header Files, Important Files, Unk File, Source Files, Libraries, Loadables. The Source pane displays C code:

```

278     }
279     LATAbits.LATA2 = 1; //como vai só subir a partir disso, define o sentido como subida
280     do //enquanto não atingir o maior andar solicitado
281     {
282         if(and_ating==min_origem_up) //se atingir o andar minimo da fila atualize o andar minimo e a distancia
283         {
284             min_up(); //min_up();
285             distancia=abs(and_ating-min_origem_up);
286         }

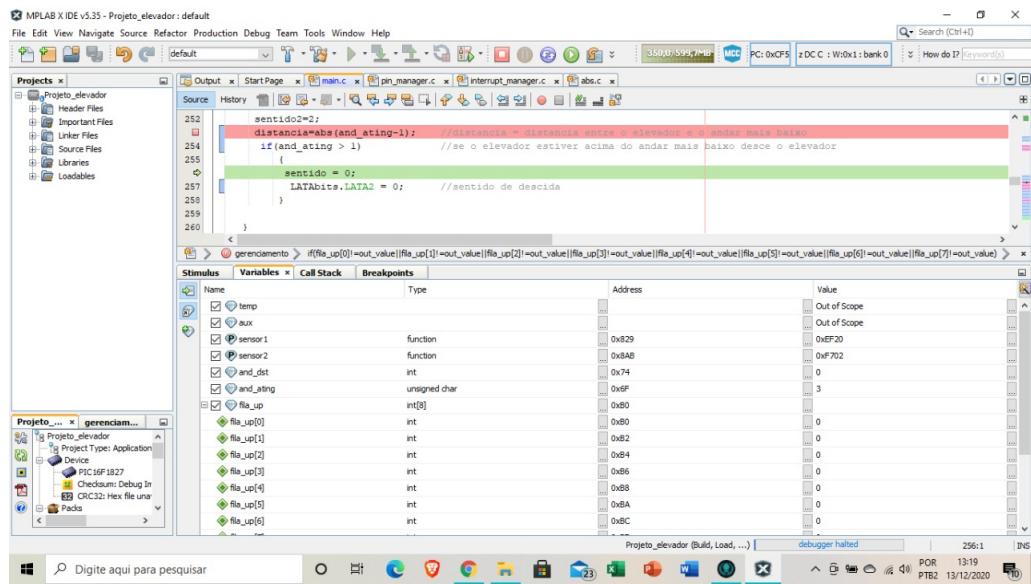
```

The Variables pane shows memory dump for 'Projeto\_elevador' project, Device PIC16F1827, and Debug In CRC32: Hex file una Pads. It lists variables like temp, aux, sensor1, sensor2, and\_dst, and\_ating, fila\_up, and\_min\_origem\_up, origem\_up, aux, and\_min\_origem\_up. The value for 'and\_min\_origem\_up' is highlighted in blue.

Esse é o algoritmo responsável pela atualização do valor mínimo.

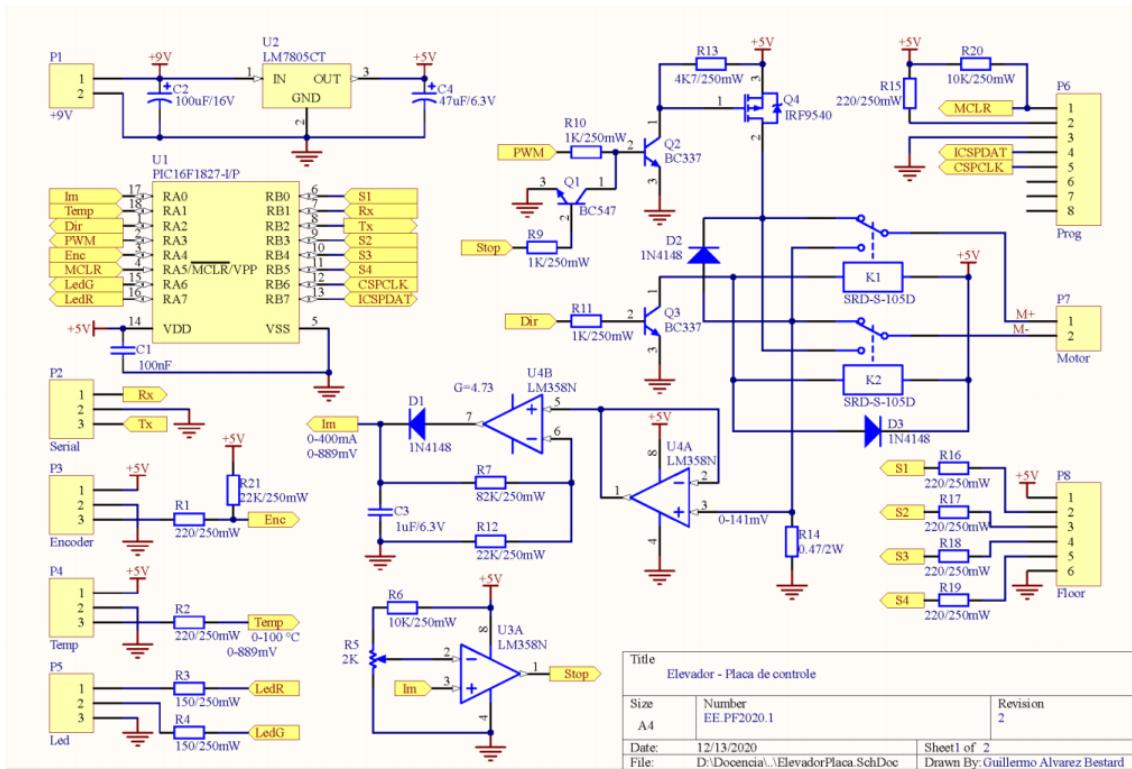
This screenshot shows the same MPLAB X IDE setup as the previous one, but the Variables pane now highlights the variable 'and\_min\_origem\_up' with a red border, indicating it is being watched or is the current focus. The rest of the interface and code are identical to the first screenshot.

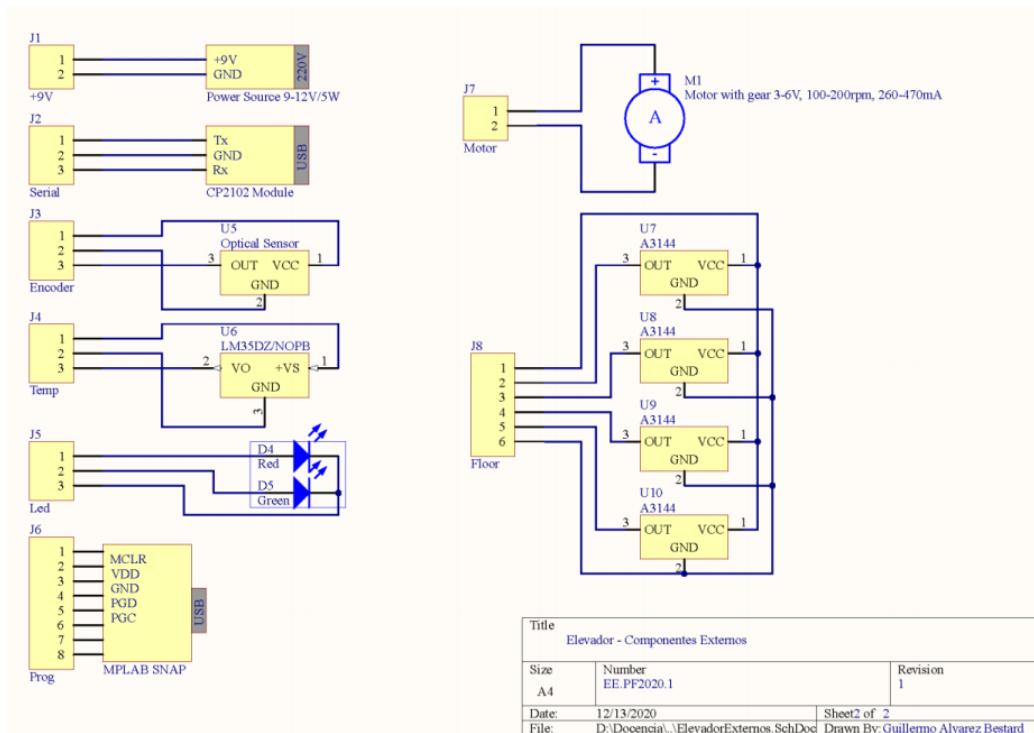
Na imagem acima, nota-se o valor do mínimo de subida sendo atualizado para 3.



Quando o elevador atinge o andar 3 que é o último andar de subida a variável do andar de destino é zerada. O elevador entra na condição de "parado" e como está em um andar acima do "1" (que definimos como o nível mínimo - de 1 a 4), modifica a "distância" e o sentido, para que o motor execute a descida do elevador.

## 1.5 Esquemático

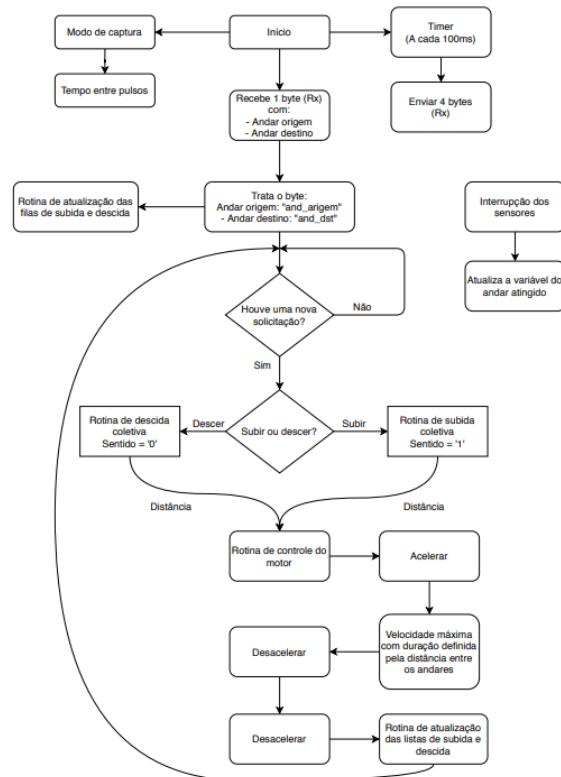


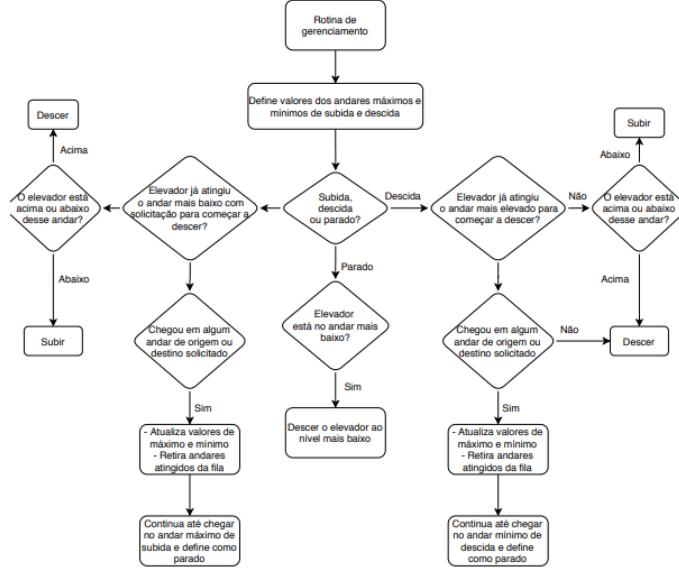


O esquemático do circuito, fornecido no relatório, é representado nas imagens acima.

## 1.6 Diagrama de Fluxo

Foi desenvolvido uma representação visual para o fluxograma de lógica do algoritmo, afim de melhor visualizar cada etapa da solução, indicando os requisitos do sistema apresentados no primeiro capítulo deste documento.





## 2 Versão Simplificada

Dado uma mudança do escopo, foi elaborado um segundo elevador com menos especificações e de funcionamento mais simples. Dessa forma, será apresentado a segunda solução desenvolvida pela equipe.

A metodologia da solução e a divisão da equipe permanece a mesma no papel, porém todos os integrantes tiveram participação na construção do segundo firmware.

### 2.1 Novas especificações do sistema

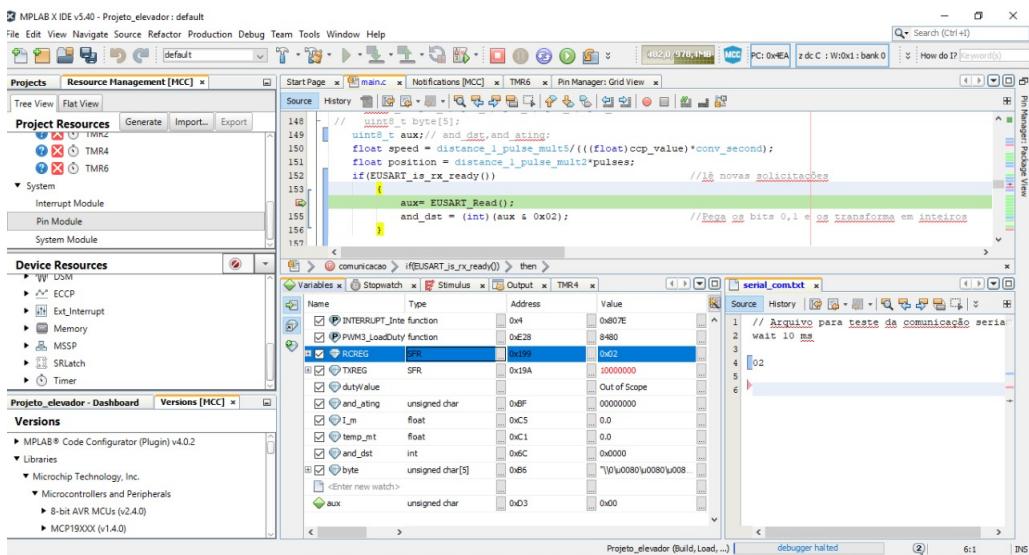
A fim de evitar redundância no documento, será citado apenas as especificações que diferem do modelo anterior, sendo elas:

- o Elevador terá uma espera de 2 segundos para o embarque/desembarque de passageiros;
- Após o andar solicitado ser atendido, o elevador deverá obrigatoriamente voltar ao andar térreo;
- haverá uma espera de 9.5 segundos antes de mudar a direção do motor por fins de segurança;
- A cor gradual dos LEDs indicando o movimento do elevador terá como parâmetro a posição do elevador, verde ao chegar no destino, vermelho

em movimento e o produto de ambos durante o intervalo de deslocamento;

- Haverá apenas um andar de destino por vez, sendo fornecido APENAS o andar de destino;
- As equações das medidas apresentadas na comunicação foram alteradas;

## 2.2 Funcionamento e simulações



A imagem acima apresenta a simulação, usando um estímulo representando o andar. Nota-se a informação sendo recebida pelo microcontrolador no registrador RCREG do periférico EUSART.

```

52 //INTERUPPI&gt;&gt; PARA ENVIO DOS DADOS A CADA 100ms (5 BYTES)
53 void send_data()
54 {
55     aux_tempo++;
56     if(aux_tempo == 14)
57     {
58         int i=0;
59         while(i<5)
60         {
61             if(EUSART_is_tx_ready())
62             {
63                 EUSART_Write(byte[i]);
64                 i++;
65             }
66         }
67     }
68 }

send_data > if(aux_tempo == 14) > then > while(<i> < 5) > if(EUSART_is_tx_ready()) > then >

Variables x Stopwatch x Stimulus x Output x serial_com.txt x TMR4 x
Target halted. Stopwatch cycle count = 199800 (99.994 ms)
Target halted. Stopwatch cycle count = 41 (20.5 µs)
Target halted. Stopwatch cycle count = 63 (31.5 µs)
Target halted. Stopwatch cycle count = 1865 (932.5 µs)
Target halted. Stopwatch cycle count = 1865 (932.5 µs)

Cycle count = 1865 (932.5 µs)

```

Pode-se aferir o tempo de envio de bytes pelo microcontrolador de aproximadamente 100 ms.

```

238
239
240     estado = 0;
241     //Aguarda 2 segundos para parada
242     PIE3bits.TMR6IE=1; // Ativa interrupção do timer utilizado para contagem da espera
243
244     acabou_delay=0;
245     TMR6_WriteTimer(0);
246     while (1)
247     {
248         if(acabou_delay == 1)
249         {
250             acabou_delay=0;
251             break;
252         }
253     }
254 }

PIE3bits.TMR6IE=0; // Desativa interrupção do timer utilizado para contagem da espera

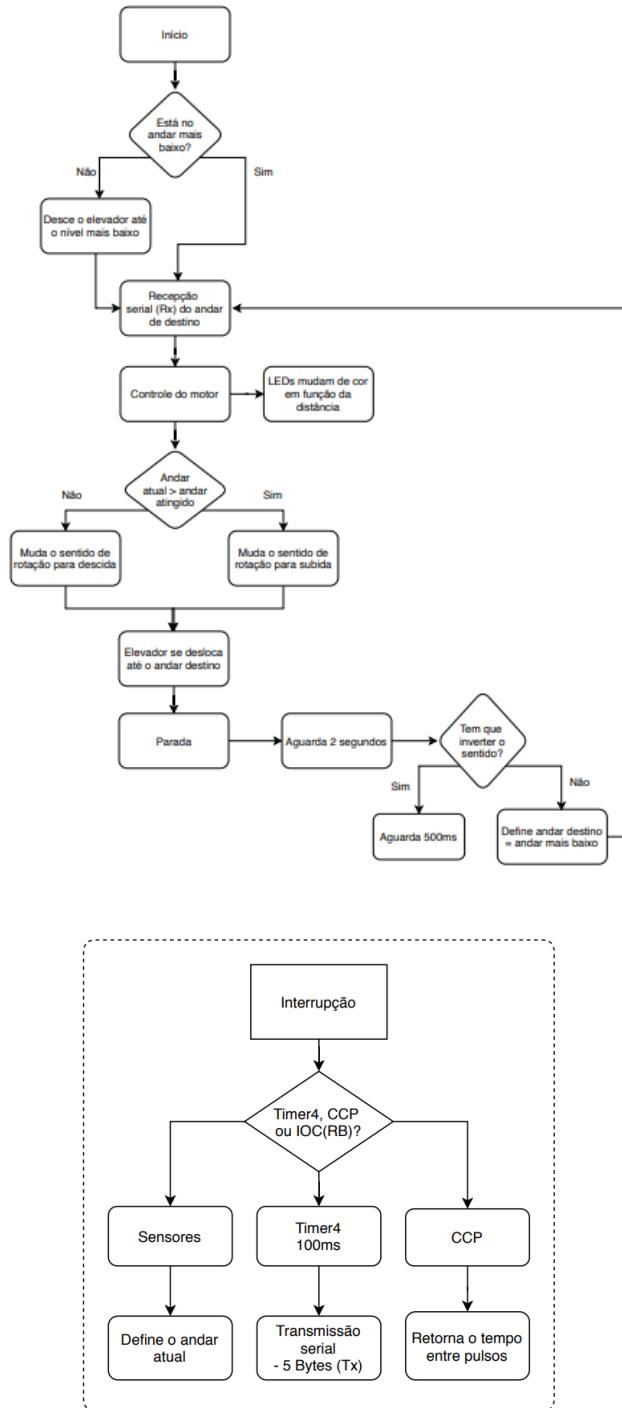
Variables x Stopwatch x Stimulus x Output x serial_com.txt x TMR4 x
Target halted. Stopwatch cycle count = 3996258 (1.998129 s)

Cycle count = 3996258 (1.998129 s)

```

Ao chegar no destino, o elevador tem o *delay* de 2 segundos, como solicitado. Foi utilizado uma interrupção para desenvolver a lógica para tal especificação.

### 2.2.1 Fluxograma



O novo fluxograma é apresentado pelo diagrama acima.

### **3 Considerações Finais**

Todas as informações contidas neste documento serão enviadas em anexo com o projeto. O envio constará:

- O Firmware desenvolvido contendo o arquivo de simulação;
- As imagens utilizadas no desenvolvimento do relatório;
- O documento do fluxograma no formato de PDF;

A lógica do algoritmo foi influenciado pelo trabalho [1], desenvolvido pelo atual professor da Universidade de Brasília, Faculdade de Tecnologia (FGA). deixamos aqui o nosso agradecimento ao seu trabalho.

### **Referências**

- [1] Daniel Mauricio Muñoz Arboleda. «Implementação e simulação de algoritmos de escalonamento para sistemas de elevadores usando arquiteturas reconfiguráveis». Em: (2006).