UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS PONTA GROSSA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

WALDYR TURQUETTI GONÇALVES

ATIVIDADE PRÁTICA SUPERVISIONADA: PROGRAMAÇÃO DE UM CRUZAMENTO DE SEMÁFOROS USANDO O PIC16F877A

PONTA GROSSA
2020

WALDYR TURQUETTI GONÇALVES

ATIVIDADE PRÁTICA SUPERVISIONADA: PROGRAMAÇÃO DE UM CRUZAMENTO DE SEMAFOROS USANDO PIC16F877A

Atividade Prática Supervisionada, apresentado à disciplina Microcontroladores, do curso de Ciência da Computação da Universidade Tecnológica Federal do Paraná – UTFPR.
Orientadora: Prof. Dra. Cristhiane Gonçalves

PONTAGROSSA
2020

SUMÁRIO

1 INTRODUÇÃO

Esse trabalho em questão é referente a atividade pratica supervisionada da disciplina Microcontroladores, onde o objetivo é fazer uma aplicação utilizando a linguagem Assembly do PIC16F877A e emular tal utilizando o Protheus.

A aplicação desse trabalho em questão foi referente a um semáforo de cruzamento para veículos e pedestres, utilizando delays e para a sincronia dos semáforos dos veículos e pedestres.

2 PROPOSTA

Assim como um cruzamento normal, os veículos podem seguir quando os semáforos forem abertos, tanto em um sentido quanto no outro, porém só podem ir em frente ou virarem a direita, desse modo os semáforos iram abrir de 2 em 2, e ai todos os semáforos de pedestres iram abrir simultaneamente após os 4 semáforos de carro terem sido abertos. Como observado na Figura 1.

Figura 1 – Ilustração do Cruzamento [1]

3 METODOLOGIA

Para executar a ideia, foi usado o mplab 8.92 [2] para programar o código em assembly que seria usado no microcontrolador, e para emular o micro controlador usamos o Proteus 8.0 [3] como software de simulação. E a ideia de delay programado foi a ideia principal, para padronizar os momentos em que os semáforos serão abertos e fechados.

4 DESENVOLVIMENTO

Nesse trabalho em questão o desenvolvimento é dividido em duas partes, a primeira é a apresentação de como o circuito do cruzamento foi montado, e na segunda é apresentado o código assembly.

4.1 CIRCUITO

No circuito utilizamos um cristal de 4mhz ligados as portas OSC1 E OSC2 do PIC_A (PIC16F877A), e ligamos esse cristal a dois capacitores de 20pF.

É ligado esses 4 semáforos as seguintes portas:

Semáforo 1-Vermelho-RA0, Amarelo-RA1, Verde-RA2

Semáforo 2-Vermelho-RB0, Amarelo-RB1, Verde-RB2

Semáforo 3-Vermelho-RC0, Amarelo-RC1, Verde-RC2

Semáforo 4-Vermelho-RD0, Amarelo-RD1, Verde-RD2

E os semáforos de pedestres foram ligados as portas:

Semáforo de pedestre 1-Vermelho-RB6, Verde-RB4

Semáforo de pedestre 2-Vermelho-RA3, Verde-RD6

Semáforo de pedestre 3-Vermelho-RB5, Verde-RB3

Semáforo de pedestre 4-Vermelho-RC4, Verde-RB7

Semáforo de pedestre 5-Vermelho-RC3, Verde-RC6

Semáforo de pedestre 6-Vermelho-RD3, Verde-RC6

Semáforo de pedestre 7-Vermelho-RD4, Verde-RC7

Semáforo de pedestre 8-Vermelho-RD7, Verde-RD5

E por fim na porta ¬ MCLR (pino 1) é um pino de entrada, onde foi configurado um botão PULL-UP, que todo vez que esse botão é acionado resetamos o PIC_A, ou seja, toda vez que apertamos o botão RESET o programa volta para início do código assembly. Como observado na Figura 2.

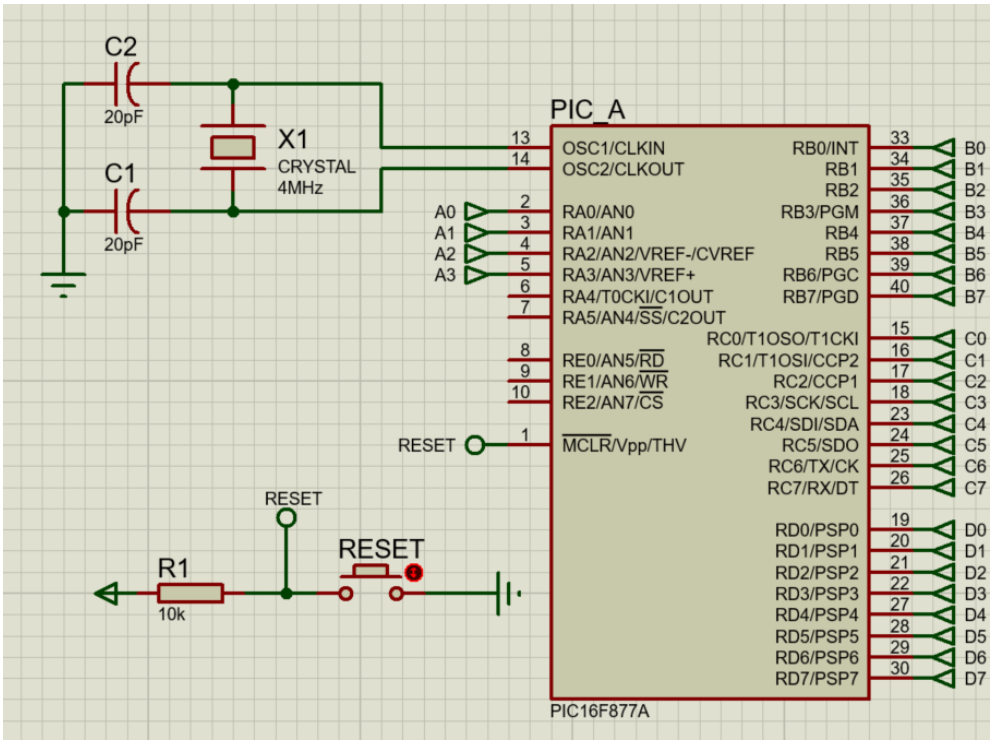

Figura 2 – As I/O e o Oscilador do PIC_A.

Foi usado 4 “Traffic Lights”, que são semáforos padrões usados no Proteus, e também 8 leds-green e 8 leds-red para fazer os semáforos de pedestres. Como observado na Figura 3.

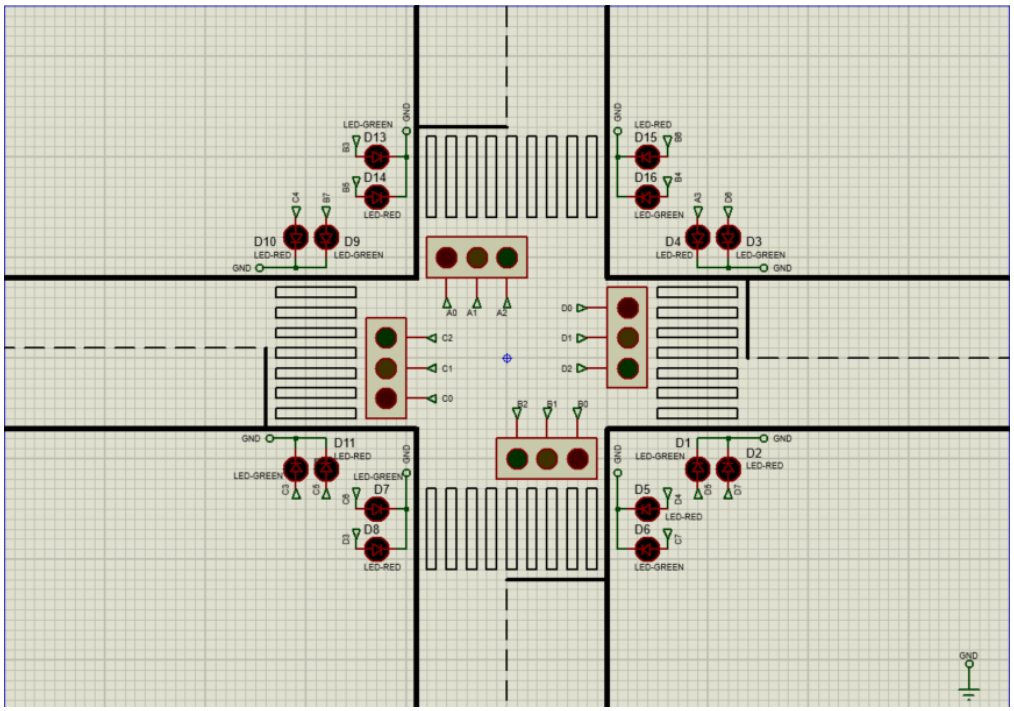

Figura 3 – Os semáforos de Veículos e os de Pedestres.

4.2 CÓDIGO

No início foi realizado os procedimentos padrões para a codificação em assembly, é definido o pic que será utilizado, as transições entre os bancos de memória, , as configurações do __CONFIG, depois separamos as variáveis para realizar os delays. Como é observado na Figura 4.

```
list p=16f877a
#include <p16f877a.inc>

;---PAGINAÇÃO DE MEMÓRIAS---
#DEFINE     BANK0   BCF     STATUS,RP0
#DEFINE     BANK1   BSF     STATUS,RP0

__CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON & _PWRTE_ON & _XT_OSC & _WRT_OFF & _LVP_OFF & _CPD_OFF
ERRORLEVEL  -302

    CBLOCK  0x20
            W_TEMP                  ; variable used for context saving
            STATUS_TEMP             ; variable used for context saving
            PCLATH_TEMP             ; variable used for context saving
            TEMP1                   ; For delay routine
            TEMP2                   ; For delay routine
            TEMP3                   ; For delay routine
            TEMP4                   ; For delay routine
    ENDC

    ;---INICIO DA APLICAÇÃO---
    ORG     0x000
    GOTO    MAIN
```

Figura 4 – Configurações Iniciais (Primeira Parte).

Então é iniciado o programa, colocando-o no início do bloco de memória (endereço 0x00) e logo ocorre o pulo para o início do código principal (MAIN). Em seguida é realizado os procedimentos de limpeza das portas do PIC_A, e por fim a configuração das portas (quais serão IN e quais serão OUT). Como é mostrado na Figura 5.

```
MAIN:
    BANK0
    CLRF    PORTA   ;LIMPA O PORTA
    CLRF    PORTB   ;LIMPA O PORTB
    CLRF    PORTC   ;LIMPA O PORTC
    CLRF    PORTD   ;LIMPA O PORTD
    CLRF    PORTE   ;LIMPA O PORTE

    BANK1
    MOVLW   B'00000000'     ;Todas os PinosB serão OUT;
    MOVWF   TRISA
    MOVLW   B'00000000'     ;Todas os PinosB serão OUT;
    MOVWF   TRISB
    MOVLW   B'00000000'     ;Todas os PinosC serão OUT;
    MOVWF   TRISC
    MOVLW   B'00000000'     ;Todas os PinosD serão OUT;
    MOVWF   TRISD
    MOVLW   B'00000111'     ;Todas os PinosE serão IN;
    MOVWF   TRISE

    BANK0

LOOPING:
```

Figura 5 – Configurações Iniciais (Segunda Parte).

4.2.1 LOOPING PRINCIPAL

A aplicação foi dividida em momentos, onde cada momento é um estado nos semáforos, ou seja, um momento seria quando para um lado está aberto e outro fechado para os veículos e tudo fechado para os pedestres, ou quando todos os semáforos de veículos estão fechados e os de pedestres estão abertos. E isso repetidamente pois está em um looping, conceito visto em [4] .

1° Momento: O 1° momento é quando a passagem dos veículos na vertical está liberada e a passagem dos veículos na horizontal e pedestres estão travadas. Sendo assim é defino onde a saída será 1 e onde será 0 através bits que são movidos para o registrador W através da instrução MOVLW e em seguida utilizar a instrução MOVWF para passar os valores dos bits para o endereço da porta em questão (Esse processo é mais detalhado no datasheet do PIC16F877A [5]) por fim é repetido 4 vezes a chamada do delay de 5 segundos para totalizar 20 segundos. Como observado na Figura 6.

```
LOOPING:

;**************************
MOVLW   B'00001100'     ;1° Momento
MOVWF   PORTA
MOVLW   B'01100100'
MOVWF   PORTB
MOVLW   B'00110001'
MOVWF   PORTC
MOVLW   B'10011001'
MOVWF   PORTD
CALL    DELAY5S
CALL    DELAY5S
CALL    DELAY5S
CALL    DELAY5S
;**************************
```

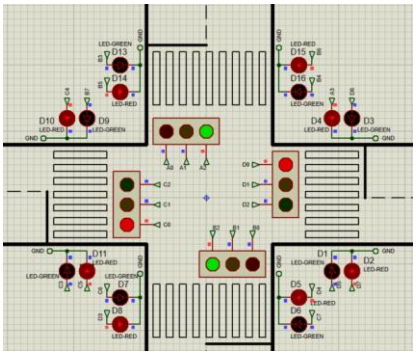

Figura 6 – Primeiro Momento.

2° Momento: O 2° momento é quando a passagem dos veículos na vertical está prestes a ficar fechada (Amarela) e a passagem dos veículos na horizontal e pedestres estão travadas. Nesse momento como o a luz amarela fica um tempo bem menor comparado a luz verde e vermelha é usado apenas 3 vezes o delay de 100 milissegundos totalizando 300 milissegundos. Como visto da Figura 7.

```
;*************************
MOVLW   B'00001010'     ;2° Momento
MOVWF   PORTA
MOVLW   B'01100010'
MOVWF   PORTB
MOVLW   B'00110001'
MOVWF   PORTC
MOVLW   B'10011001'
MOVWF   PORTD
CALL    DELAY100M
CALL    DELAY100M
CALL    DELAY100M
;*************************
```

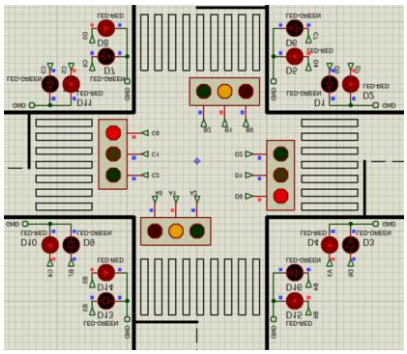

Figura 7 – Segundo Momento.

3° Momento, 6° Momento e 8° Momento : Esses três momentos são idênticos. Seriam os momentos em que o todos os semáforos tanto os de veículos como os de pedestres estão fechados por um breve período de tempo (300 mS), esses momentos são necessários para assim que um par de semáforos fechar o outra par não abra imediatamente, seria mais para uma medida de segurança. Para esse momento foi criado uma função “TUDOVERMELHO” Demonstrado na Figura 8.

```
CALL TUDOVERMELHO       ;3° Momento


CALL TUDOVERMELHO       ;6° Momento
CALL ATIVAPEDESTRE      ;7° Momento
CALL TUDOVERMELHO       ;8° Momento

GOTO LOOPING
```

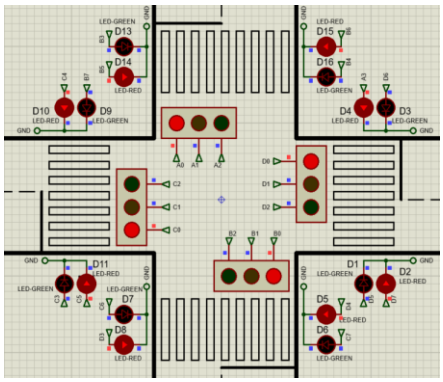

Figura 8 – Terceiro, Sexto e Oitavo Momento.

4° Momento: O 4° momento é quando a passagem dos veículos na horizontal está liberada e a passagem dos veículos na vertical e pedestres estão travadas, funcionando de um jeito bem parecido com o 1° momento. Visto na Figura 9.

```
;*************************
MOVLW   B'00001001'     ;4° Momento
MOVWF   PORTA
MOVLW   B'01100001'
MOVWF   PORTB
MOVLW   B'00110100'
MOVWF   PORTC
MOVLW   B'10011100'
MOVWF   PORTD
CALL    DELAY5S
CALL    DELAY5S
CALL    DELAY5S
CALL    DELAY5S
;*************************
```

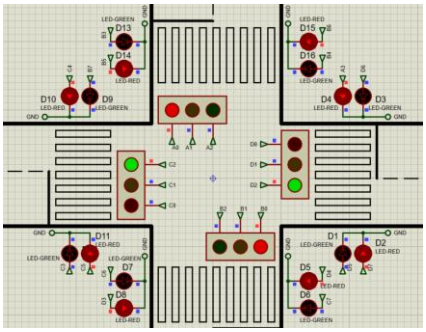

Figura 9 – Quarto Momento.

5° Momento: O 5° momento é quando a passagem dos veículos na horizontal está prestes a ficar fechada (Amarela) e a passagem dos veículos na vertical e pedestres estão travadas. Equivalente ao 2° momento, demostrado na Figura 10.

```
;*************************
MOVLW   B'00001001'     ;5° Momento
MOVWF   PORTA
MOVLW   B'01100001'
MOVWF   PORTB
MOVLW   B'00110010'
MOVWF   PORTC
MOVLW   B'10011010'
MOVWF   PORTD
CALL    DELAY100M
CALL    DELAY100M
CALL    DELAY100M
;*************************
```

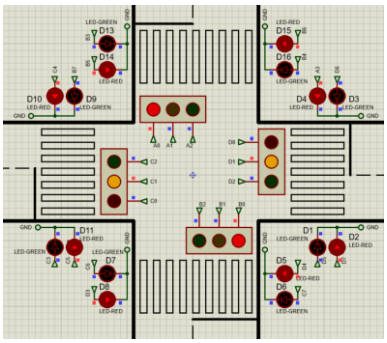

Figura 10 – Quinto Momento.

7° Momento: O 7° momento é quando a passagem dos veículos na horizontal e na vertical estão fechadas e a dos pedestres estão abertas por 10 segundos, esse momento é feito através da chamada de função “ATIVAPEDESTRE” e depois dos 10 segundos os faróis de pedestres começam a piscar demostrando que o farol já irá fechar e o semáforo dos veículos vão se abrir novamente.

```
CALL TUDOVERMELHO       ;6° Momento
CALL ATIVAPEDESTRE      ;7° Momento
CALL TUDOVERMELHO       ;8° Momento
```

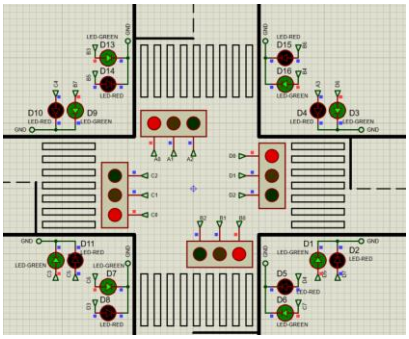

Figura 11 – Sétimo Momento.

4.2.2 CHAMADA DE FUNÇÕES

Nesse trabalho em questão foi criado apenas duas chamadas de funções “CALL”, sendo a “TUDOVERMELHO” onde é a função que representa os momentos 3, 6 e 8 visualizado na Figura 8. E a função “ATIVAPEDESTRE” que representa o momento 7 da aplicação visualizado na Figura 11.

A função “TUDOVERMELHO” e a função onde todos os faróis dos veículos e também todos os dos pedestres ficam vermelhos, como foi visualizado na Figura 8. Sua construção é simples, assim que o “CALL TUDOVERMELHO” é feito o programa enviado para o endereço de TUDOVERMELHO, em seguida é setado os bits para a condição ser satisfeita e chamado três vezes o delay de 100 ms (totalizando 300 ms) e por fim é retornado para onde a função foi chamada. O código pode ser visualizado na Figura 12.

```
;*******************************************************************************
TUDOVERMELHO:               ;Função que deixa todos os Semáforos com apenas o VERMELHO ATIVO
    MOVLW   B'00001001'
    MOVWF   PORTA
    MOVLW   B'01100001'
    MOVWF   PORTB
    MOVLW   B'00110001'
    MOVWF   PORTC
    MOVLW   B'10011001'
    MOVWF   PORTD
    CALL    DELAY100M       ;Totalizando 300mS
    CALL    DELAY100M
    CALL    DELAY100M
    RETURN
;*******************************************************************************
```

Figura 12 – FUNÇÃO “TUDOVERMELHO”.

Já a função “ATIVAPEDESTRE” é um pouco mais complexa, assim que ela é chamada o programa é enviado para o endereço ATIVAPEDESTRE e utilizado a variável TEMP4 para armazenar o valor D’5’ (5 Decimal) e setado os bits para satisfazer as condições dessa função por 2 vezes o delay de 5 s, ou seja, 10 s no total. Em seguida é iniciado o “LOOPPEDESTRE”, esse loop é utilizado para piscar as luzes vermelhas dos faróis dos pedestres indicando que o farol irá fechar definitivamente e os faróis dos veículos iram abrir. Esse loop ocorre 5 vezes, graças a instrução DECFSZ que decrementa 1 do TEMP4 e verifica se o TEMP4 está igual a 0, se ele for 0 o loop para e retorna para endereço da chamada da função, caso ao contrário faz um GOTO para o LOOPPEDESTRE e assim vai. Visualiza-se o código dessa função na Figura 13.

```
;*******************************************************************************
ATIVAPEDESTRE:
    MOVLW   D'5'
    MOVWF   TEMP4

    MOVLW   B'00000001'
    MOVWF   PORTA
    MOVLW   B'10011001'
    MOVWF   PORTB
    MOVLW   B'11001001'
    MOVWF   PORTC
    MOVLW   B'01100001'
    MOVWF   PORTD
    CALL    DELAY5S
    CALL    DELAY5S

LOOPPEDESTRE:
    MOVLW   B'00001001'
    MOVWF   PORTA
    MOVLW   B'01100001'
    MOVWF   PORTB
    MOVLW   B'00110001'
    MOVWF   PORTC
    MOVLW   B'10011001'
    MOVWF   PORTD
    CALL    DELAY100M
    MOVLW   B'00000001'
    MOVWF   PORTA
    MOVLW   B'00000001'
    MOVWF   PORTB
    MOVLW   B'00000001'
    MOVWF   PORTC
    MOVLW   B'00000001'
    MOVWF   PORTD
    CALL    DELAY100M

    DECFSZ  TEMP4, 1
    GOTO LOOPPEDESTRE
    RETURN
;*******************************************************************************
```

Figura 13 – FUNÇÃO “ATIVAPEDESTRE”.

4.2.3 DELAYS

O delay nada mais é do que uma maneira de gastar os ciclos de clock para que atrase a continuação do algoritmo. Há várias maneiras de gastar os ciclos de clocks, porém nesse trabalho foram utilizados apenas 1. No começo da função delay definimos valores para as TEMPs e ficamos decrementando esses TEMPs, como se fossem FOR dentre de FOR. No caso do “DELAY5S” são três TEMPs e três FORs, já no “DELAY100M” são dois TEMPs e dois FORs. Como observado na Figura 14.

```
;---DELAYS---
;*******************************************************************************
DELAY5S:                ;Função de Delay de 5s
    MOVLW   0X07    ;07
    MOVWF   TEMP1
    MOVLW   0X2F    ;47
    MOVWF   TEMP2
    MOVLW   0X03    ;03
    MOVWF   TEMP3
DELAY5SLOOP
    DECFSZ  TEMP1, 1
    GOTO    $+2
    DECFSZ  TEMP2, 1
    GOTO    $+2
    DECFSZ  TEMP3, 1
    GOTO    DELAY5SLOOP
    GOTO    $+1
    GOTO    $+1
    GOTO    $+1
    RETURN
;*******************************************************************************
;*******************************************************************************
DELAY100M:              ;Delay 100 mili routine
    MOVLW   0x1E    ;30
    MOVWF   TEMP1
    MOVLW   0x4F    ;79
    MOVWF   TEMP2
DELAYLOOP
    DECFSZ  TEMP1, 1
    GOTO    $+2
    DECFSZ  TEMP2, 1
    GOTO    DELAYLOOP
    GOTO    $+1
    NOP
    RETURN
;*******************************************************************************
```

Figura 14 – Código dos Delays de 5 s e de 100 ms respectivamente.

Para representar melhor a ideia dos delays o fluxograma presentes na Figura 15 é essencial.

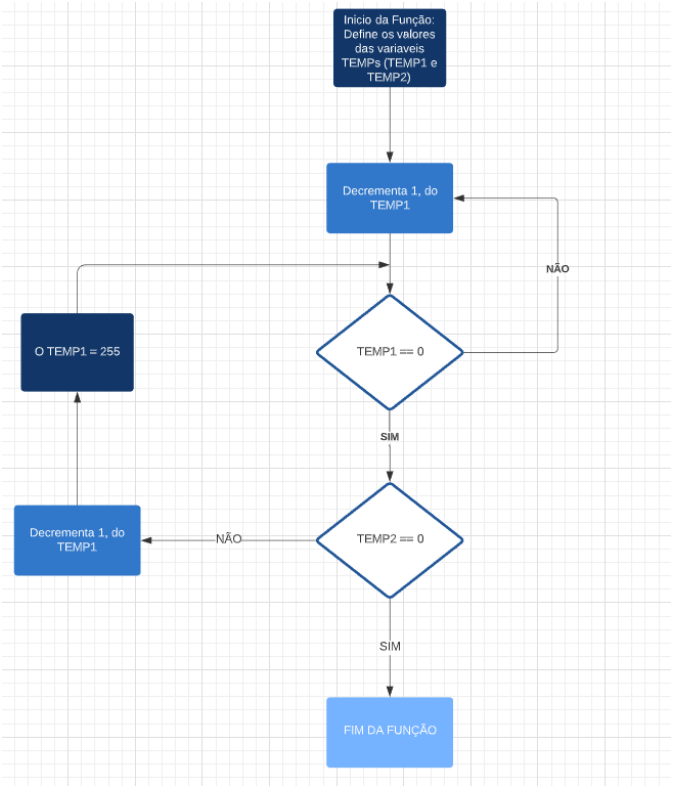

Figura 15 – Fluxograma para compreensão dos Delays.

5 CONSIDERAÇÕES FINAIS

A linguagem assembly é mais complicada do que as outras por ser baixo nível, sendo que um código simples de poucas linhas na linguagem C/C++ vire um código de mais de 100 linhas facilmente em assembly. Por esse fato foi um projeto que a lógica não foi um problema, mas sim a estrutura de dados foi o problema.

A aplicação realizada se aproxima de um caso real de semáforos, porém não é completamente igual, pelo fato de que um caso real tenha muito mais variáveis externas, e o fato também que esse projeto foi realizado para um caso especifico de semáforos de cruzamento.

Nessa aplicação em especifica o uso de Delays foi essencial para ocorrer a sincronização dos semáforos. Algo bem importante também para o projeto foram as chamadas de funções para deixar o código mais limpo e mais fácil de entender.

Por fim a atividade fez perceber as infinidades de aplicações possíveis de se fazer com os PICs, por possibilitarem fazer aplicações complexas e que realmente tenha alguma utilidade para o a sociedade e/ou para indústria.

REFERÊNCIAS

[1] Link da **Imagem do Semáforo de Cruzamento**: https://blogs.diariodepernambuco.com.br/mobilidadeurbana/2012/11/pedestre-sem-vez-na-rua/

[2] Link do **MPLAB**: https://www.microchip.com/development-tools/pic-and-dspic-downloads-archive

[3] Link do **Proteus**: https://www.labcenter.com/downloads/

[4] SOUZA, Devid José de. **Desbravando o PIC**. 8. ed. São Paulo: Erica, 2005. 262 p.

[5] MICROCHIP. **PIC16F87XA Data Sheet**. Usa: Microchip Technology Inc., 2003. 234 p.