Micro Controladores
Prática 2 - Semáforo de Carros e Pedestre com Display de 7 segmentos

Átila Camurça Alves

[1]Instituto Federal do Ceará (IFCE)

Abstract. *This practice has the goal to create a car and a pedestrian semaphore using the SanUSB tool [SanUSB 2011] and a PIC18F4550 microcontroller [Microchip 2006], where the pedestrian can solicitate his passage and view a 7 segments display which show the time left that he has to cross a highway.*

Resumo. *Esta prática tem como objetivo criar um semáforo de carros e pedestres usando a ferramenta SanUSB [SanUSB 2011] e microcontrolador PIC18F4550 [Microchip 2006], em que o pedestre pode solicitar sua passagem e visualizar um display de 7 segmentos que mostra o tempo que ele tem para fazer a travessia de uma auto estrada.*

Lista de códigos fonte

1. Introdução

Esta prática tem como objetivo criar um semáforo de carros e pedestres usando a ferramenta SanUSB [SanUSB 2011] e microcontrolador PIC18F4550 [Microchip 2006], em que o pedestre pode solicitar sua passagem e visualizar um display de 7 segmentos que mostra o tempo que ele tem para fazer a passagem. Imagine que este semáforo está em uma auto estrada de alta velocidade. Nesse cenário o sinal para os carros está sempre verde e para os pedestres está sempre vermelho. Quando um pedestre solicita a passagem através de um botão, o sinal para os carros sai do vermelho para o amarelo, logo em seguida para o vermelho, enquanto isso o sinal dos pedestres vai para o verde e um par de displays de 7 segmentos mostra o tempo que o pedestre tem para atravessar a auto estrada antes que o sinal fique verde para os carros novamente.

2. Material Necessário

- 5 LEDs
- 7 Resistores 390Ω
- 2 Displays de 7 segmentos
- 1 Cabo USB
- 1 Placa SanUSB
- 1 Protoboard

A programação é feita através da MPLAB-X IDE [Microship 2016] e a gravação é feita com o Gravador SanUSB [SanUSB 2016].

3. Iteração 1 - Semáforo simples

Inicialmente faremos um semáforo simples apenas dos carros, como mostra a Figura 1.

Código Fonte da iteração 1 no Apêndice A.

4. Iteração 2 - Semáforo simples para carros e pedestres

Em seguida faremos um semáforo tanto para os carros quanto para os pedestres. O semáforo deve estar sempre verde para os carros e vermelho para os pedestres. Ao acionar um botão o semáforo entra num processo em que o sinal alterna por um tempo.

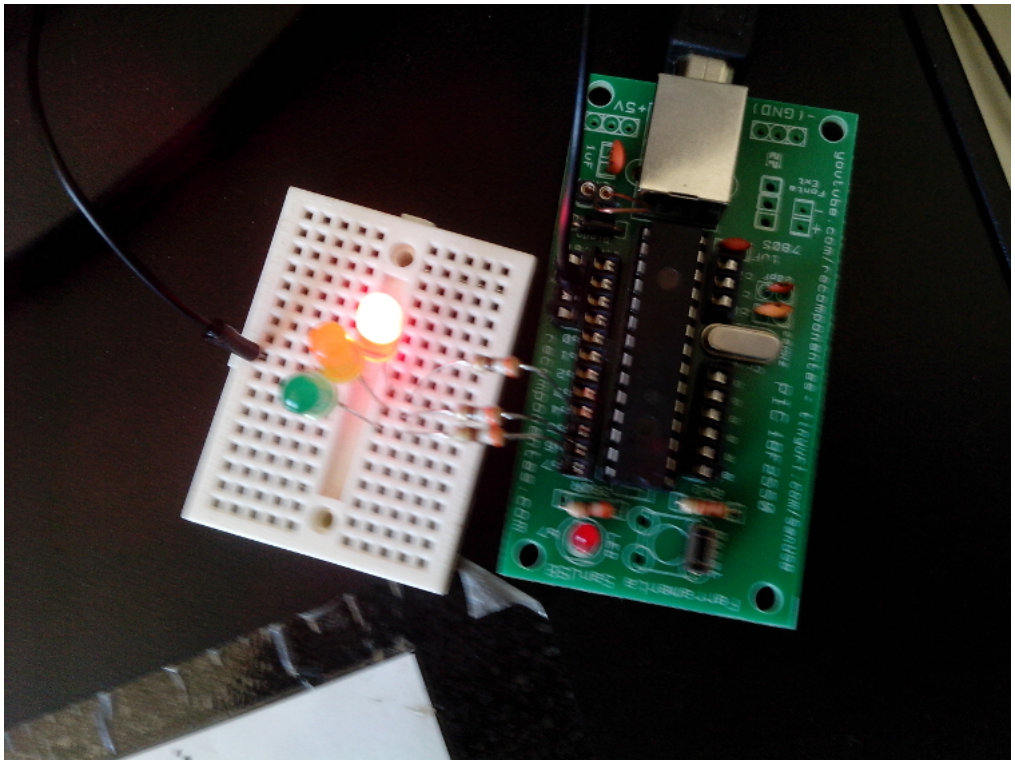

Figura 1. Semáforo Simples de Carros

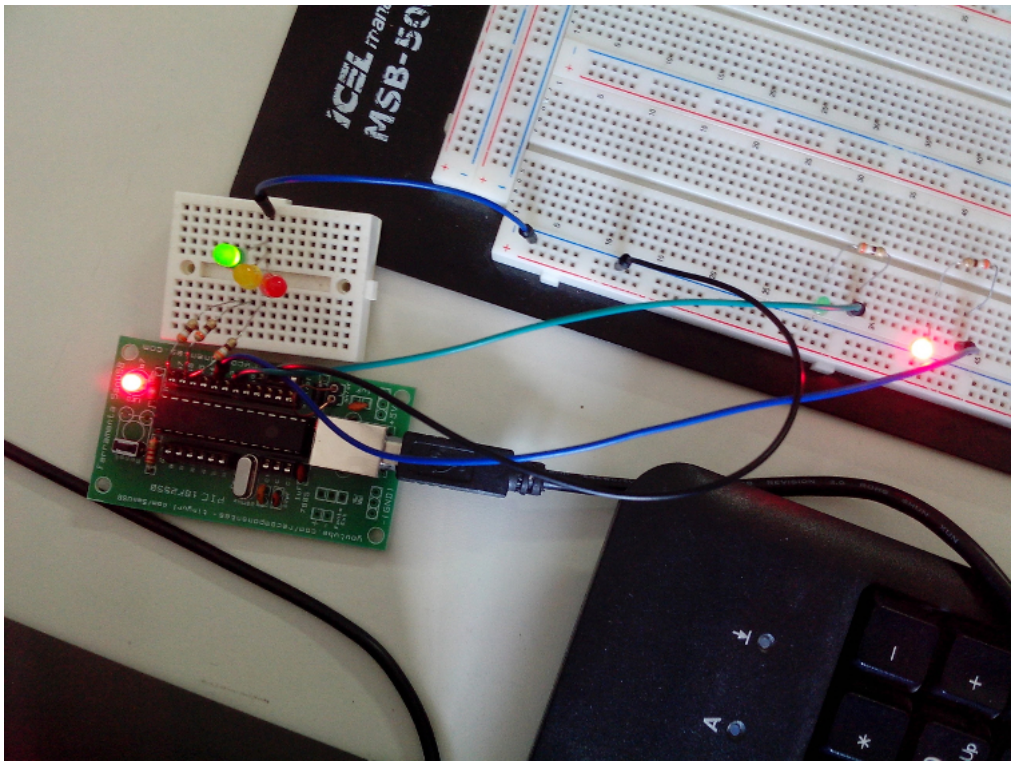

Figura 2. Semáforo simples para carros e pedestres

Código Fonte da iteração 2 no Apêndice B.

5. Iteração 3 - Display de 7 segmentos com contagem regressiva

Com o semáforo produzido é preciso usar um display de 7 segmentos que irá mostrar a contagem regressiva para que o sinal verde do pedestre vá para vermelho. Mas antes é preciso fazer apenas que o display conte de zero a nove indefinidamente.

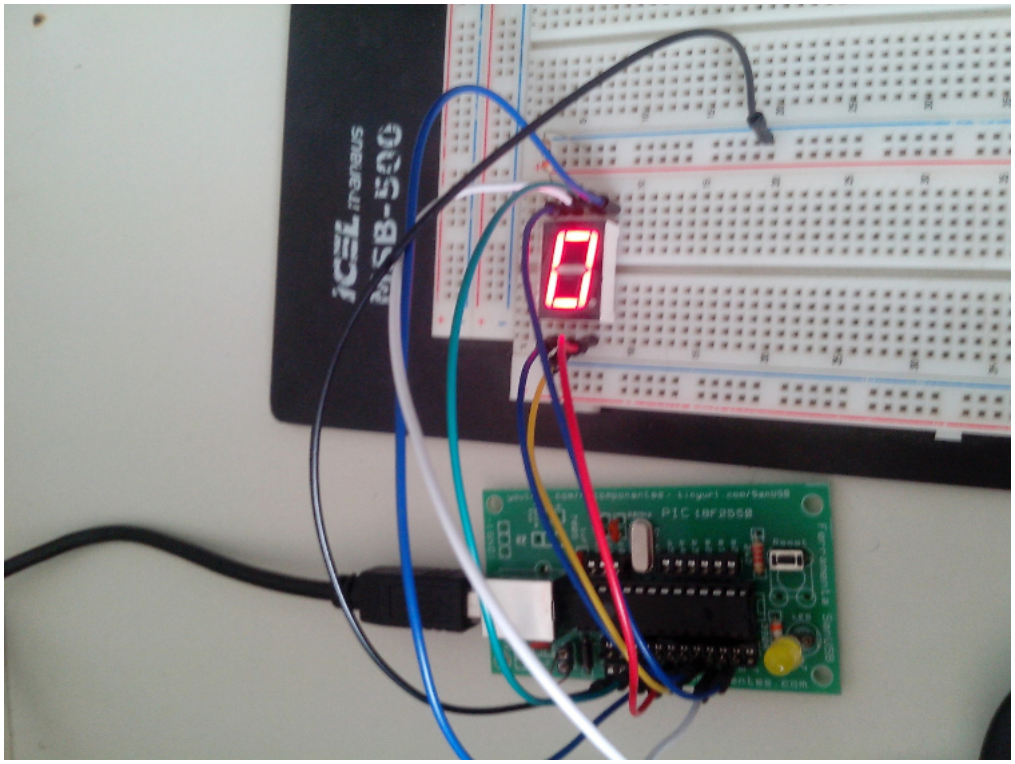

Figura 3. Display de 7 segmentos com contagem regressiva

Código Fonte da iteração 3 no Apêndice C.

6. Iteração 4 - 2 displays de 7 segmentos multiplexados

Como visto na Figura 3, um único display de 7 segmentos ocupa 7 pinos e um GND da placa. Sendo assim para conseguir usar 2 displays seriam necessários 14 pinos, o que torna o projeto mais complexo e caro.

Para resolver esse problema basta aplicar a multiplexação, fazendo com que os pinos sejam controlados por pinos chamados pinos de controle. Para ligar 2 displays colocamos os segmentos em curto e trocar o GND por dois pinos na placa. O estado do pino é alternado em espaços curtos de tempo de forma que dê a impressão que os displays estão acesos ao mesmo tempo.

Veja o circuito montado na Figura 4.

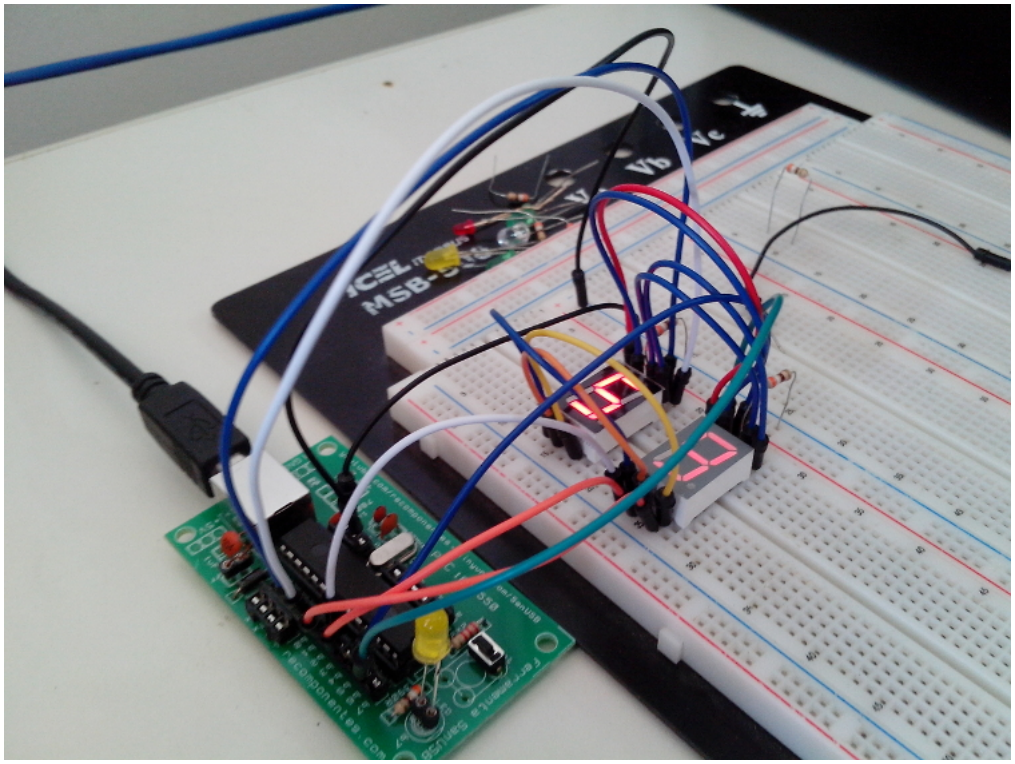

Figura 4. Multiplexação de 2 Displays de 7 segmentos

Código Fonte da iteração 4 no Apêndice D.

7. Iteração 5 - Semáforo para carros e pedestres com contagem regressiva

A partir do conteúdo já visto é possível criar um semáforo para carros e pedestres com contagem regressiva.

O circuito completo pode ser visto na Figura 5.

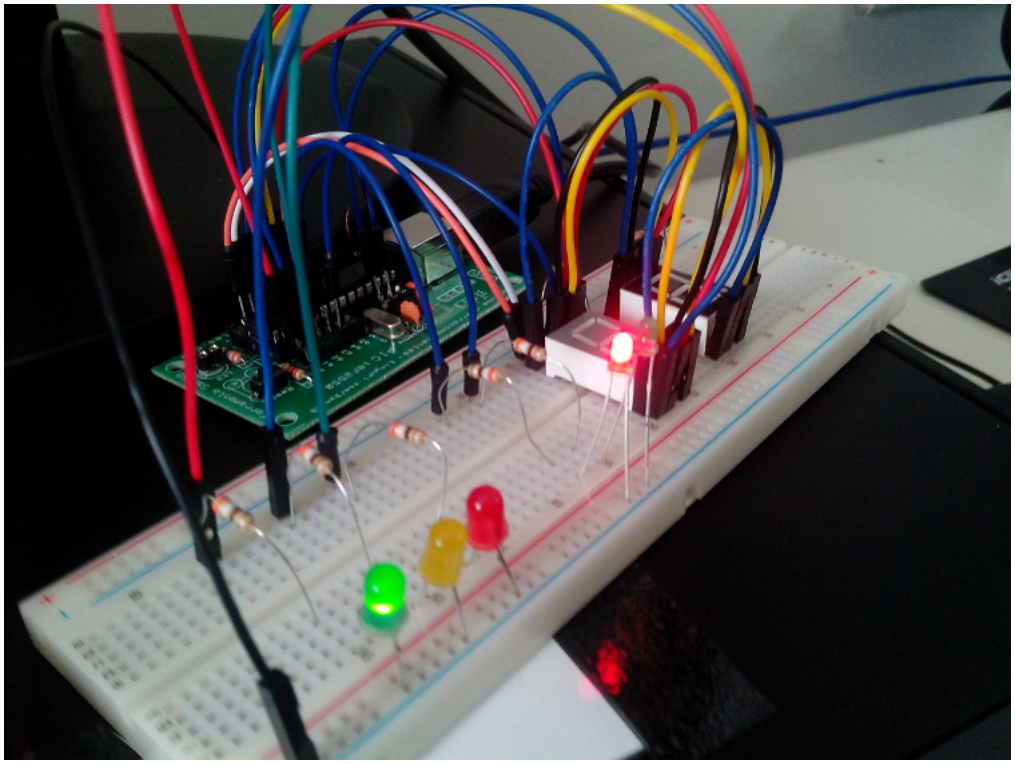

Figura 5. Semáforo para carros e pedestres com contagem regressiva

Código Fonte da iteração 5 no Apêndice E.

Referências

[Microchip 2006] Microchip (2006). PIC18F2455/2550/4455/4550 Data Sheet. http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf. [Online; accessed 24-August-2016].

[Microship 2016] Microship (2016). MPLAB X Integrated Development Environment (IDE). http://www.microchip.com/mplab/mplab-x-ide. [Online; accessed 24-August-2016].

[SanUSB 2011] SanUSB, G. (2011). Aplicações Práticas de Eletrônica e Microcontroladores em Sistemas Computacionais. https://www.dropbox.com/s/0e8r2sh94x9enof/2%20-%20Apostila_MPLABX.pdf?dl=0. [Online; accessed 27-July-2016].

[SanUSB 2016] SanUSB, G. (2016). Gravador da plataforma SanUSB. https://github.com/SanUSB-grupo/Gravador-Linux. [Online; accessed 24-August-2016].

A. Código fonte Iteração 1

```
#include "SanUSB48X.h"

void interrupt interrupcao() {}

void main(void) {
    clock_int_48MHz();
    while (1) {
        nivel_alto(pin_b7);
        tempo_ms(3000);
        nivel_baixo(pin_b7);

        nivel_alto(pin_b6);
        tempo_ms(1000);
        nivel_baixo(pin_b6);

        nivel_alto(pin_b5);
        tempo_ms(3000);
        nivel_baixo(pin_b5);
    }
}
```

Listing 1: Iteração 1

B. Código fonte Iteração 2

```
#include "SanUSB48X.h"
void interrupt interrupcao() {}

#define verde_carro pin_b7
#define amarelo_carro pin_b6
#define vermelho_carro pin_b5

#define vermelho_pedestre pin_b3
#define verde_pedestre pin_b1

int flag_pedestre = 0, i;

void tempo(int tempo) {
    for (i = 0; i < tempo; i += 100) {
        tempo_ms(100);
        if (! entrada_pin_e3) {
            flag_pedestre = 1;
        }
    }
}

void main(void) {
    clock_int_48MHz();
        while (1) {
        nivel_alto(vermelho_pedestre);
        nivel_baixo(vermelho_carro);

        nivel_alto(verde_carro);
        tempo(10000);
        nivel_baixo(verde_carro);

        if (flag_pedestre) {
            nivel_alto(amarelo_carro);
            tempo_ms(1000);
            nivel_baixo(amarelo_carro);

            nivel_alto(vermelho_carro);
            nivel_baixo(vermelho_pedestre);

            nivel_alto(verde_pedestre);
            tempo_ms(5000);
            nivel_baixo(verde_pedestre);
            flag_pedestre = 0;
        }
    }
}
```

Listing 2: Iteração 2

C. Código fonte Iteração 3

```
#include "SanUSB48X.h"
#define zero   0b0111111
#define um     0b0000110
#define dois   0b1011011
#define tres   0b1001111
#define quatro 0b1100110
#define cinco  0b1101101
#define seis   0b1111101
#define sete   0b0000111
#define oito   0b1111111
#define nove   0b1101111

unsigned char seg[10] = {
    zero, um, dois, tres, quatro,
    cinco, seis, sete, oito, nove
};
int i;

void interrupt interrupcao() {}

void main(void) {
    clock_int_48MHz();
    TRISB = 0b00000000;
    while (1) {
        for (index = 9; index >= 0; index--) {
            PORTB = seg[index];
            tempo_ms(1000);
        }
    }
}
```

Listing 3: Iteração 3

D. Código fonte Iteração 4

```
#include "SanUSB48X.h"
#define zero   0b0111111
#define um     0b0000110
#define dois   0b1011011
#define tres   0b1001111
#define quatro 0b1100110
#define cinco  0b1101101
#define seis   0b1111101
#define sete   0b0000111
#define oito   0b1111111
#define nove   0b1101111
#define pin_dezena pin_c1
#define pin_unidade pin_c0

unsigned char seg[10] = {
    zero, um, dois, tres, quatro,
    cinco, seis, sete, oito, nove
};
int i, z, dezena, unidade;

void interrupt interrupcao() {}

void main(void) {
    clock_int_48MHz();
    TRISB = 0b00000000;
    TRISC = 0b000;
    while (1) {
        for (i = 0; i < 99; i++) {
            for (z = 0; z < 100; z++) {
                dezena = i / 10;
                unidade = i % 10;

                nivel_baixo(pin_dezena);
                nivel_alto(pin_unidade);

                PORTB = seg[dezena];
                tempo_ms(5);

                nivel_baixo(pin_unidade);
                nivel_alto(pin_dezena);

                PORTB = seg[unidade];
                tempo_ms(5);
            }
        }
    }
}
```

Listing 4: Iteração 4

E. Código fonte Iteração 5

```
#include "SanUSB48X.h"
#define zero        0b0111111
#define um          0b0000110
#define dois        0b1011011
#define tres        0b1001111
#define quatro      0b1100110
#define cinco       0b1101101
#define seis        0b1111101
#define sete        0b0000111
#define oito        0b1111111
#define nove        0b1101111
#define apagado     0b00000000
#define verde_carro pin_a2
#define amarelo_carro pin_a3
#define vermelho_carro pin_a4
#define vermelho_pedestre pin_a1
#define verde_pedestre pin_a0
#define pin_dezena pin_c1
#define pin_unidade pin_c0

unsigned char seg[10] = {
    zero, um, dois, tres, quatro,
    cinco, seis, sete, oito, nove
};
int i, j, z, dezena, unidade, flag_pedestre = 0;

void interrupt interrupcao() {}

void tempo(int tempo) {
    for (j = 0; j < tempo; j += 100) {
        tempo_ms(100);
        if (!entrada_pin_e3) {
            flag_pedestre = 1;
        }
    }
}
```

Listing 5: Iteração 5a

```
void main(void) {
    clock_int_48MHz();
    TRISB = 0b00000000;
    TRISA = 0b000000;
    TRISC = 0b000;
    nivel_baixo(amarelo_carro);
    nivel_baixo(verde_carro);
    nivel_baixo(verde_pedestre);

    while (1) {
        nivel_alto(vermelho_pedestre);
        nivel_baixo(vermelho_carro);
        nivel_alto(verde_carro);
        tempo(5000);
        nivel_baixo(verde_carro);

        if (flag_pedestre) {
            nivel_alto(amarelo_carro);
            tempo_ms(1000);
            nivel_baixo(amarelo_carro);
            nivel_alto(vermelho_carro);
            nivel_baixo(vermelho_pedestre);
            nivel_alto(verde_pedestre);

            for (i = 20; i >= 0; i--) {
                for (z = 0; z < 100; z++) {
                    dezena = i / 10;
                    unidade = i % 10;
                    nivel_baixo(pin_dezena);
                    nivel_alto(pin_unidade);
                    PORTB = seg[dezena];
                    tempo_ms(5);
                    nivel_baixo(pin_unidade);
                    nivel_alto(pin_dezena);
                    PORTB = seg[unidade];
                    tempo_ms(5);
                }
            }
            nivel_baixo(verde_pedestre);
            PORTB = apagado;
            flag_pedestre = 0;
        }
    }
}
```

Listing 6: Iteração 5b