Rampa eletrônica de lava carros feito com pic16F628A

Gabriel Machado[1], Juliana Batista[1], Iago Costa das Flores[1]

[1]Faculdade de Computação e Engenharia Elétrica – Universidade Federal do Sul e Sudeste do Pará (UNIFESSPA)
Marabá – PA – Brazil

{univercomput,juliana.batista, gabriel.machado}@unifesspa.edu.br

Resumo. *Este relatório tem como objetivo, descrever de forma detalhada o funcionamento de um código escrito em linguagem C, com a finalidade de simular uma catraca elétrica de estacionamento, onde um dos motores é responsável por fazer o dispositivo abrir, junto de um led verde como indicativo, e outro motor para fechar a catraca, também com um led, na cor vermelha, como indicativo. O código em C, foi utilizado para programar o hardware descrito acima, ao qual foi esquematizado com o auxílio do programa Proteus.*

Palavras-chave—C, pic, programação.

1. Introdução

O projeto foi elaborado para simular uma catraca eletrônica, a lógica da aplicação foi implementada utilizando um laço de repetição infinito e 2 condicionais para analisar o estado dos botões a serem pressionados. O tempo que cada motor fica ativo é definido por um "delay" de 4s.

O código feito em C, foi escrito e compilado utilizando o programa MPLAB, após a compilação do script, um arquivo .hex foi gerado e utilizado no programa Proteus, onde o hardware para o funcionamento do programa foi esquematizado com 2 motores, 2 leds, um pic16f628a, 4 resistores, 2 botões, 1 cristal e 2 capacitores, após realizar as devidas ligações, a simulação no Proteus foi iniciada para que fosse possível observar o funcionamento do hardware programado em linguagem C.

2. Definição do Projeto

O projeto consiste em dois servos motores do tipo DC (corrente contínua), dois leds e dois botões. Cada botão aciona um motor e um led. Os motores irão girar em sentidos contrários quando acionados. Fazendo alusão a subida e descida da rampa. Outros equipamentos necessários para reproduzir fisicamente o projeto foram abstraídos, sendo usado apenas os componentes eletrônicos ligados diretamente ao pic16F628A.

Na figura 1 é possível ver nas linhas 2 até a linha 16 temos a configuração padrão para usar o pic16F628A juntamente com a definição das portas de entrada e saída utilizadas no projeto.

3. Fundamentação Teórica

A criação de programas para microcontroladores e microprocessadores pode até ser complexo de acordo com o crescimento da complexidade desenvolvida, os primeiros dispositivos programáveis eram escritos em linguagem de máquina, que basicamente eram códigos em binário, isso tornando a tarefa extremamente complexa resultado em custos elevados e um grande tempo para desenvolvimento dessas aplicações. A linguagem Assembly é de baixo nível e não possui nenhum comando, instruções ou função além daquelas definidas no conjunto de passos do processador utilizado, resultado em um trabalho maior para executar rotinas e operações que não fazem parte do conjunto conhecido, para isso temos as chamadas linguagens de alto nível, são criadas para permitir a programação utilizando comandos mais complexos que são propositalmente traduzidos para linguagens de baixo nível em seu processo de utilização.

Para fins de compreensão das outras maneiras de exercer aplicações com a família pic, temos uma nova atividade com o desafio de implementar na linguagem C como são os caminhos para estas simulações. Agora também é possível completar atividades mais complexas devido à versatilidade da linguagem de programação.

4. Metodologia

Como projeto agora foi alterado com mais versatilidade o objetivo é implementar uma solução prática em utilizando microcontroladores da família pic na linguagem de programação C, com hardware e software, então com auxílio de alguns programas como o MPLAB o mesmo projeto foi seguido com os mesmo objetivos propostos em ser uma forma de utilizar motores para atividades específicas, com seu botões para início e retorno das tarefas, isso sendo indicadas pelos leds. Portanto foram aplicados passos em C para ser inseridos no simulador buscando demonstrar como está o funcionamento das aplicações, são utilizados leds, motores, resistores e outros itens para completar essa tarefa, quando o motor recebe o pulso que vem da placa é possível observar os leds e o motor entrando em ação, com o fim do timer requerido na aplicação, temos o fim da ação e também é possível reverter essa ação com o outro motor.

```
1   //Created on August 4, 2021, 12:58 PM
2   #include<xc.h>
3   #pragma config FOSC = INTOSCIO // Oscillator Selection bits (HS oscillator: Hi
4   #pragma config WDTE = OFF      // Watchdog Timer Enable bit (WDT disabled)
5   #pragma config PWRTE = OFF     // Power-up Timer Enable bit (PWRT disabled)
6   #pragma config MCLRE = OFF     // RA5/MCLR/VPP Pin Function Select bit
7   //(RA5/MCLR/VPP pin function is MCLR)
8   #pragma config BOREN = OFF     // Brown-out Detect Enable bit (BOD disabled)
9   #pragma config LVP = OFF       // Low-Voltage Programming Enable bit //(RB4/PG
10  #pragma config CPD = OFF       // Data EE Memory Code Protection bit (Data mem
11  #pragma config CP = OFF
12  #define _XTAL_FREQ 4000000 //Define a utilizacao do clock interno de 4 Mhz
13  #define MOTOR1 RB0
14  #define MOTOR2 RB1
15  #define bot1 RA0
16  #define bot2 RA1
```

Figura 1. Configurações iniciais

Na figura 02 temos a função main onde inicializa as portas de entrada e saída que irão ser utilizadas. Nota-se dois if um para verificar o acionamento de cada botão e manipular o funcionamento dos motores e leds.

```
int main() {
    TRISA = 0x03; // Define portas RA0 e RA1 como entrada
    TRISB = 0x00; //Define todas as portas B como saida
    PORTB = 0x00;
    PORTA = 0x00;
    while (1) //Loop infinito
    {
        if (bot1 == 0) { //botao 1 sem corrente
            MOTOR1 = 1; // motor 1 acionado
            __delay_ms(4000); // delay 4 segundos
            MOTOR1 = 0; // motor 1 desligado
            bot1 = 1; // botao 1 com corrente
        }
        if (bot2 == 0) { //botao 2 sem corrente
            MOTOR2 = 1; // motor 2 acionado
            __delay_ms(4000); // delay 4 segundos
            MOTOR2 = 0; // motor 2 desligado
            bot2 = 1; // botao 2 com corrente
        }
    }
    return 0;
}
```

Figura 2. main do programa

5. Arquitetura

Na parte da arquitetura temos o código fonte escrito e compilado com a ajuda da IDE MPLAB X v5.35. Juntamente com a simulação montada e simulada no software Proteus.

O código foi escrito em c com uso da função delay_ms do pic16F628a para realizar a contagem do tempo de subida e descida da rampa eletrônica.

Na figura 5 podemos observar a montagem do circuito eletrônico com o PIC16F628A com os botões, motores e leds. Usou-se alguns resistores para manter a corrente adequada para os motores, leds e acionamento dos botões.

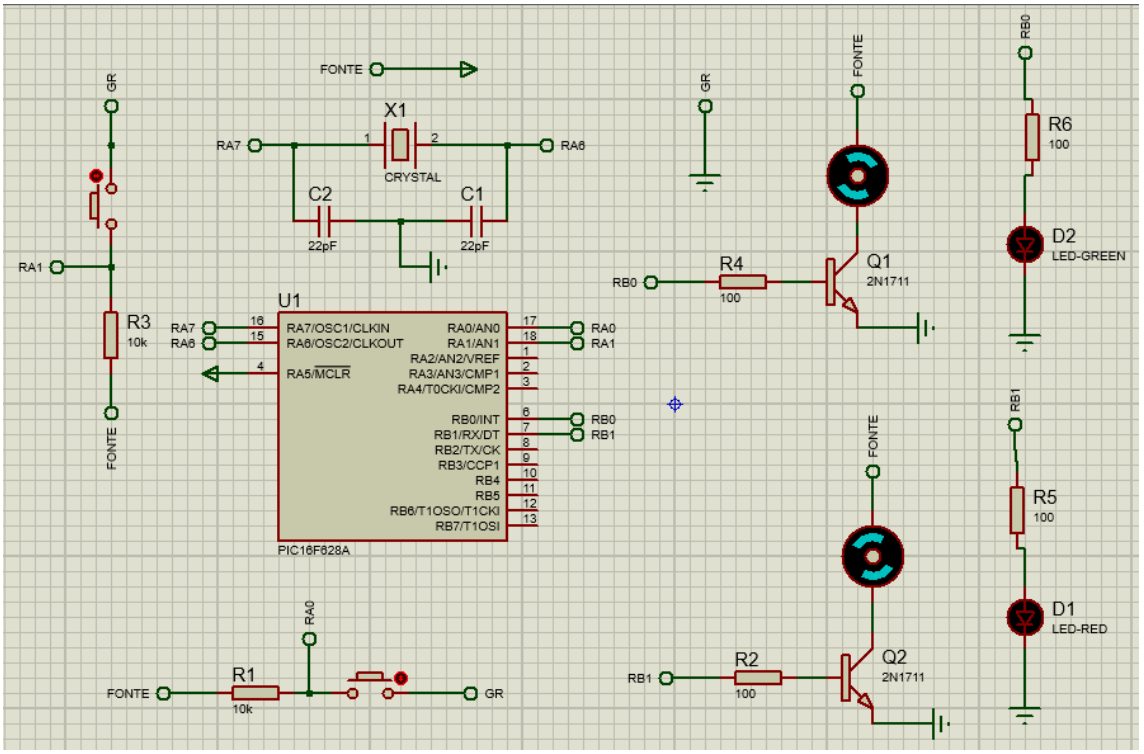

Figura 3. Arquitetura do projeto na simulação

6. Conclusão

Durante o desenvolvimento do projeto, não foram encontradas grandes dificuldades, devido à simplicidade do código escrito em C. Não se fez necessário o uso de nenhuma biblioteca além da biblioteca padrão “xc.h”.

Todo o código do programa foi implementado utilizando um laço de repetição e 2 condicionais, apesar da simplicidade do código, foi possível desenvolver um projeto que atendesse às exigências do roteiro proposto. O funcionamento é bastante eficiente, tendo em vista que a aplicação final pode ser utilizada em uma situação real.

7. Referências

José, D. S. **Desbravando o PIC - Ampliado e atualizado para PIC16F628A.** 7ª ed. Ed.Érica, 2003.

8. Apêndice

```
//Created on August 4, 2021, 12:58 PM
#include<xc.h>
#pragma config FOSC = INTOSCIO // Oscillator Selection bits (HS
oscillator: High-speed crystal/resonator on RA6/OSC2/CLKOUT and
RA7/OSC1/CLKIN)
#pragma config WDTE = OFF      // Watchdog Timer Enable bit (WDT
disabled)
#pragma config PWRTE = OFF     // Power-up Timer Enable bit (PWRT
disabled)
#pragma config MCLRE = OFF     // RA5/MCLR/VPP Pin Function Select
bit
//(RA5/MCLR/VPP pin function is MCLR)
#pragma config BOREN = OFF     // Brown-out Detect Enable bit (BOD
disabled)
#pragma config LVP = OFF       // Low-Voltage Programming Enable
bit //(RB4/PGM pin has digital I/O function, HV on MCLR must be
used for //programming)
#pragma config CPD = OFF       // Data EE Memory Code Protection
bit (Data memory code protection off)
#pragma config CP = OFF
#define _XTAL_FREQ 4000000 //Define a utilizacao do clock interno
de 4 Mhz
#define MOTOR1 RB0
#define MOTOR2 RB1
#define bot1 RA0
#define bot2 RA1
int main() {
   TRISA = 0x03; // Define portas RA0 e RA1 como entrada
   TRISB = 0x00; //Define todas as portas B como saida
   PORTB = 0x00;
   PORTA = 0x00;
   while (1) //Loop infinito
   {
       if (bot1 == 0) { //botao 1 sem corrente
```

```
            MOTOR1 = 1; // motor 1 acionado
            __delay_ms(4000); // delay 4 segundos
            MOTOR1 = 0; // motor 1 desligado
            bot1 = 1; // botao 1 com corrente
        }
        if (bot2 == 0) { //botao 2 sem corrente
            MOTOR2 = 1; // motor 2 acionado
            __delay_ms(4000); // delay 4 segundos
            MOTOR2 = 0; // motor 2 desligado
            bot2 = 1; // botao 2 com corrente
        }
    }
    return 0;
}
```