MANUAL DO CSLADDER MIC

1.0 INTRODUÇÃO

O CSLadder Mic, é uma ferramenta didática construída por Rocha (2012), como trabalho de conclusão de curso na formação de Engenharia de Computação da Pontifícia Universidade Católica de Minas Gerais no ano de 2012. Sobre orientação do professor Luiz Carlos Figueiredo.

O objetivo da ferramenta é auxiliar os estudantes a elaborar programas de CLP (Controlador Lógico Programável) na Linguagem Ladder, com utilização de microcontrolador em substituição ao CLP. Onde foi selecionado o microcontrolador ATmega328 da fabricante ATMEL, com a plataforma de desenvolvimento Arduino Duemilanove composta com compilador (Arduino alpha, *Open-Source*) e placa de gravação de microcontrolador. O recurso didático ainda possibilita a depuração do programa através do Ambiente de simulação.

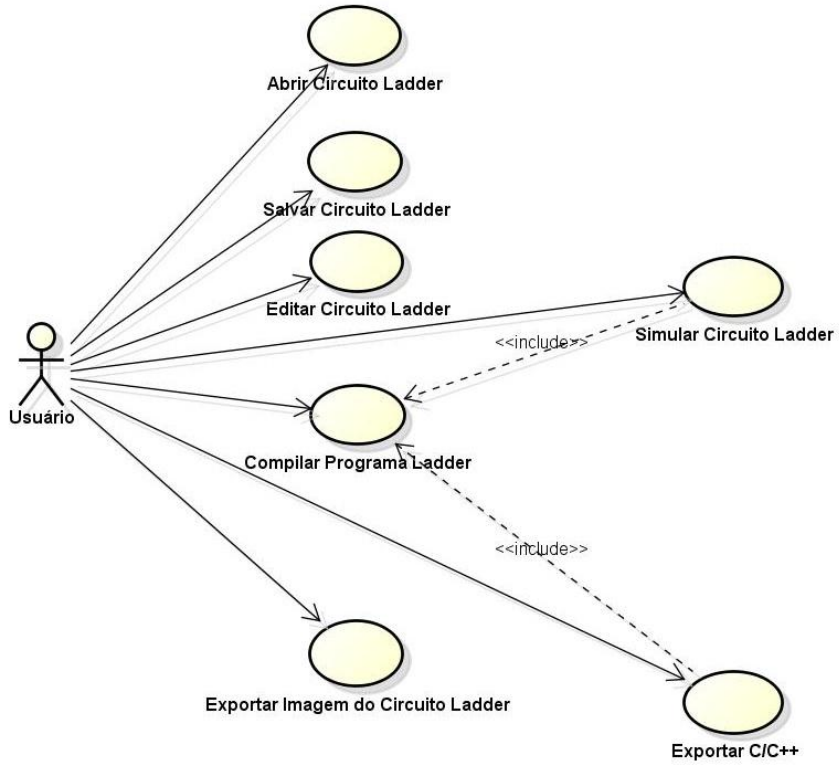
O Csladder Mic permite realizar edições na linguagem gráfica Ladder, gerando de forma automática (etapas de compilação) o código objeto na linguagem C/C++, para posteriormente ser compilado na *Integrated Development Environment* (IDE), do Arduino alpha.

As instruções da linguagem Ladder permitidas na versão 1.0 do CsLadder Mic são: Normal Aberto (NA), Normal Fechado (NF), *Output Terminal Energize* (OTE), *Output Terminal Energize Latch* (OTL) e *Output Terminal Unlatch* (OTU) (MORAES, 2007).

2.0 FUNCIONALIDADES

A seguir serão descritos as funcionalidades do CSLadder Mic. A Figura 1 apresenta o Diagrama de Caso de Uso do CSLadder Mic, exibindo as possibilidades de ação do usuário com o aplicativo.

FIGURA 1 – Caso de Uso do CSLadder Mic



Fonte: Elaborado pelo autor

2.1 Barra de ferramentas do Menu arquivo

A Figura 2 apresenta os botões de atalhos para o menu arquivo, a seguir serão descritas suas funcionalidades.

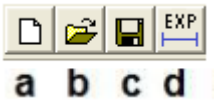
A Figura 2 item “a” ou no menu bar **arquivo→novo**, corresponde a criar um novo diagrama Ladder apagando todo conteúdo já editado.

A Figura 2 item “b” ou menu bar **arquivo/abrir**, corresponde a abrir um programa Ladder salvo.

Na Figura 2 item “c” ou menu bar **arquivo/salvar**, corresponde a salvar um programa Ladder. Visto que o CSLadder Mic salva seu diagramas Ladder na extensão LAD.

Na Figura 2 item “d” ou menu bar **arquivo/exportar BMP**, corresponde a exportar uma imagem com a extensão BMP do programa Ladder editado pelo usuário.

FIGURA 2 - Botões de manipulação de arquivos



Fonte: Elaborado pelo autor

2.2 Formas de Edição no CSLAdder MIC

2.2.1 Editor Gráfico Ladder

Este editor consiste em elaborar o diagrama Ladder através do mouse. Não permite ao usuário elaborar edições em que as conexões não apresentem um caminho possível a ser energizado da barra da esquerda até a da direita.

2.2.1.1 Considerações de Edição

A seguir serão descritos alguns padrões adotados para as edições.

cada matriz lógica, possui dimensões $4x9$. Sabendo que $M_{i,j}$, então tem-se a variação de $1 \leq i \leq 4$ e $1 \leq j \leq 9$, totalizando 36 células.

Por padrão, nas colunas ímpares apenas podem ser inseridas instruções ou conexão de continuidade e nas colunas pares somente são inseridas as conexões da linguagem Ladder.

O número de matrizes lógicas é limitado à quantidade de memória do computador. Cada matriz só poderá conter uma instrução de saída na célula $M_{1,9}$, por isso as células $M_{2,9}$, $M_{3,9}$ e $M_{4,9}$ destinadas para instruções de saída não terão funcionalidade.

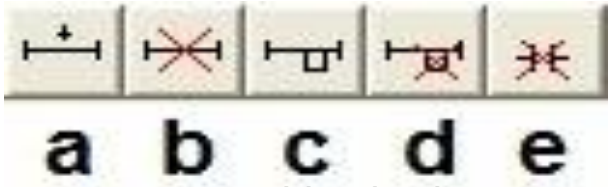
A memória e os pinos de entrada e saída do microcontrolador são limitados, por isso foi necessário criar uma padronização dos pinos digitais e limitar o número de instruções utilizadas no programa Ladder. O padrão utilizado será apresentado a seguir:

- **Entradas (EN)** - são entradas digitais do microcontrolador, variando de EN0 (pino digital 0) até EN7 (pino digital 7), totalizando 8 entradas;
- **Saídas (S)** - são saídas digitais do microcontroladores, variando de S8 (pino digital 8) até S13 (pino digital 13), totalizando 6 saídas;
- **Contadores (C)** - são os contadores Ladder da instrução CTU, variando de C0 (contador 0) até C7 (contador 7), totalizando 8 contadores;
- **Temporizadores (T) ou cronômetros** - são os temporizadores Ladder da instrução TON, variando de T0 (temporizador 0) até T4 (temporizador 4), totalizando 5 temporizadores;
- ***Flags* (F)** - são bobinas internas utilizadas para expandir o número de instruções Ladder NA, NF, OTE, OTU e OTL, variando de F0 (*flag* 0) até F15 (*flag* 15), totalizando 16 *flags*.

2.2.1.2 Construção das ramificações

No aplicativo foi padronizado que um programa em LD deve possuir no mínimo uma matriz lógica. O usuário pode adicionar ou remover uma matriz lógica por vez, através dos botões apresentados na Figura 3 itens "a" e "b", respectivamente.

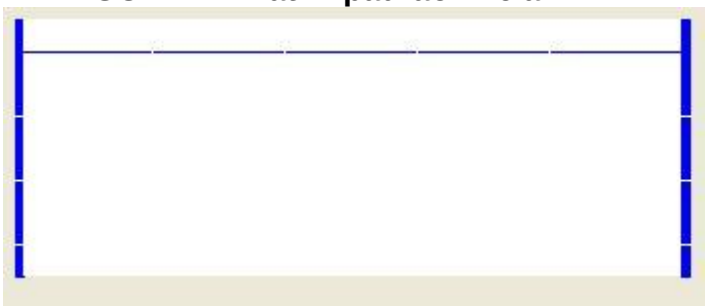
FIGURA 3 - Botões da ferramenta de edição



Fonte: Elaborado pelo autor

Inicialmente toda matriz a ser editada possui um conjunto de conexões de seguimento na primeira linha que leva a barra da esquerda à direita, como mostrado pela Figura 4.

FIGURA 4 - Matriz padrão inicial



Fonte: Elaborado pelo autor

Para construir as ramificações ou degraus, o usuário deve construir das linhas superiores às inferiores e em cada uma destas é realizada a construção das colunas da esquerda à direita. Já as desconstruções devem seguir o caminho inverso sendo realizado das linhas inferiores às superiores e para cada uma destas se destrói as colunas mais a direita.

A construção ou destruição de degraus é realizada passo-a-passo, sendo que o semicircuito que será construído ou destruído aparecerá selecionado ao usuário, para enfim decidir se deseja remover ou construir o circuito correto. Na Figura 3 itens "c" e "d", são apresentados os botões de construção e destruição da ramificação, respectivamente. O usuário deve escolher um deles e depois selecionar o local da matriz onde deseja realizar a edição.

Para facilitar a edição do programa Ladder, só é permitido ao usuário construir conexões em paralelo à esquerda e seguida de conexões em séries ou paralelas. A Figura 5 apresenta uma matriz Ladder que não pode ser construída e sua adaptação para ser construída no CSLadder Mic é representada pela Figura 6.

FIGURA 1 - Lógica em Ladder

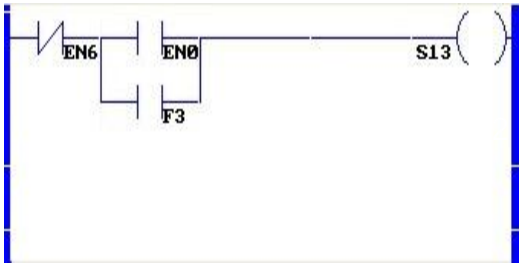
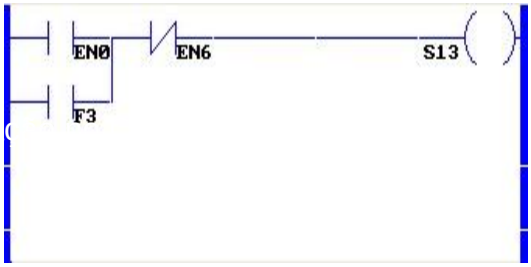


FIGURA 6 - Adaptação no CSLadder Mic



Fonte: Elaborado pelo autor

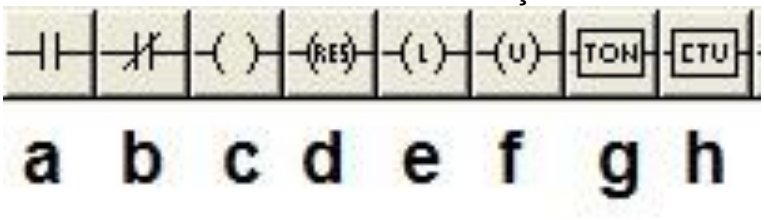
Na Figura 4 item "e" é utilizado para remover uma instrução.

É sugerido ao usuário construir primeiramente todas as ramificações de cada matriz e em seguida inserir as instruções Ladder na mesma.

2.2.1.3 Inserção de instrução

Para inserir uma instrução de entrada ou saída o usuário deverá selecionar a instrução na barra de ferramentas representa pela Figura 8 e direcionar ao local a ser inserido, assim será substituída uma conexão de continuidade pela instrução desejada.

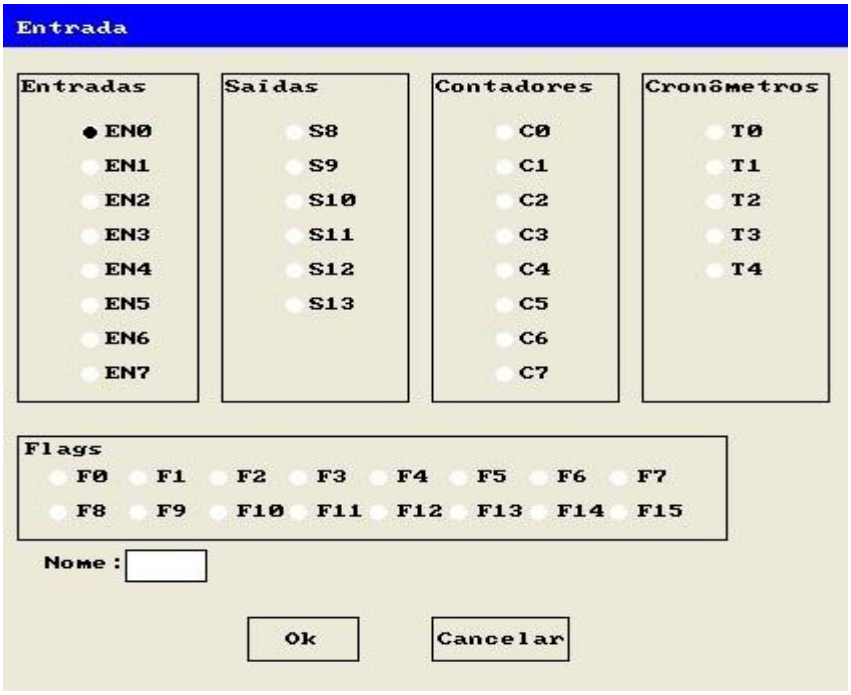
FIGURA 3 - Botões de instruções



Fonte: Elaborado pelo autor

A Figura 8 itens "a" e "b", apresentam as instruções NA e NF, respectivamente. Quando inseridas em local correto aparecerá à janela representada na Figura 9 onde são mostradas as opções de instruções.

FIGURA 4 - Inserção de instrução NA e NF

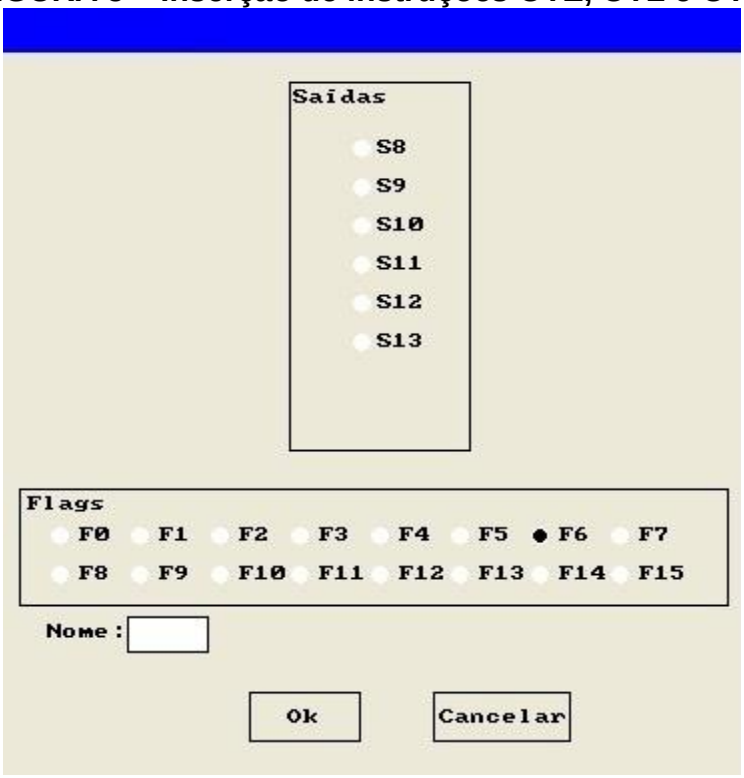


Fonte: Elaborado pelo autor

As instruções de entradas e saídas se refere aos pinos digitais do microcontrolador e seus respectivos estados lógicos. Os “Cronômetros” referem se ao estado do bit DN de um temporizador, os “Contadores” referem se ao bit DN de um contador e as “*Flags”* referem se ao estado lógico de uma *flag* utilizada no programa.

A Figura 8 itens "c" (referente à instrução OTE), "e" (L referente à instrução OTL) e "f" (U referente à instrução OTU), são as bobinas de saídas de uma matriz. Quando inseridas em local correto a janela da Figura 10 será exibida e também apresentará as opções de utilização para saídas digitais e *Flags*.

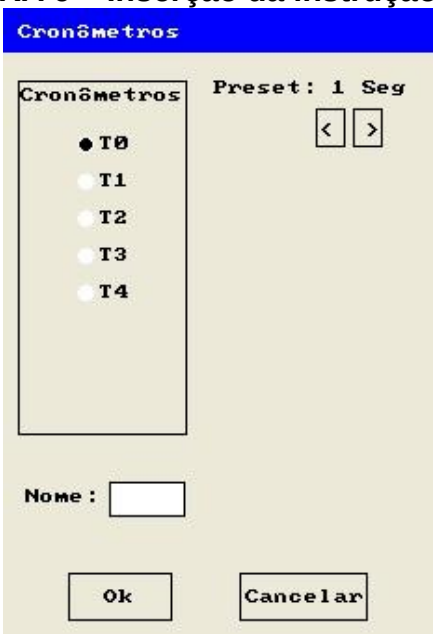
FIGURA 5 – Inserção de instruções OTE, OTL e OTU



Fonte: Elaborado pelo autor

Na Figura 8 o item "g" (TON) é utilizado para inserção de um temporizador em uma matriz. Quando inserido em local correto, será exibida a janela da Figura 31 que apresenta os temporizadores específicos e o usuário deverá definir o temporizador e o tempo em segundos do *Preset* do mesmo.

FIGURA 6 – Inserção da instrução TON



Fonte: Elaborado pelo autor

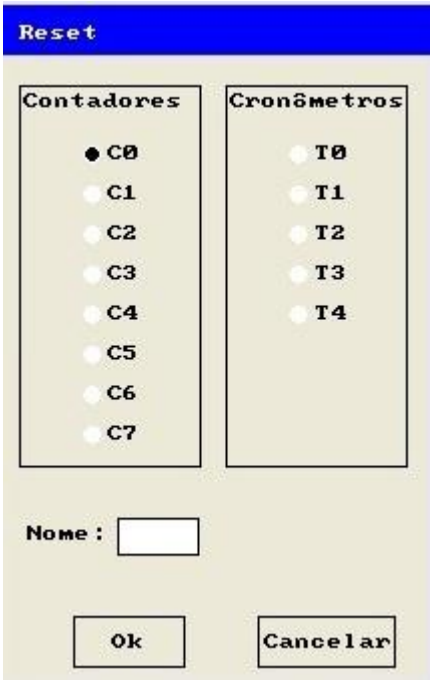
Na Figura 8 o item "h" (CTU) é utilizado para inserção de um contador em uma matriz. Quando inserido em local correto, aparecerá à janela da Figura 12 que apresenta os contadores específicos e o usuário deverá definir o contador e o valor do *Preset* do mesmo.

FIGURA 7 – Inserção da instrução CTU

Fonte: Elaborado pelo autor

Na Figura 8 no item "d" foi criada a instrução *Reset* (RES) de uma matriz, para que seja possível reiniciar um contador ou temporizador específicos. Quando inserido em local correto, aparecerá a janela da Figura 13 onde apresenta as opções para que os contadores ou cronômetros utilizados possam voltar ao estado inicial.

FIGURA 8 – Inserção de instrução RES



Fonte: Elaborado pelo autor

2.3 Ambiente de Simulação da linguagem Ladder

Este ambiente é uma das saídas do CSLadder Mic. Consiste em botões de entradas gráficos e painéis de visualização dos atributos das instruções LD. O Ambiente de Simulação, Depurador, *Debugger* ou Simulador possui um objetivo didático, pois simula o funcionamento do programa Ladder para que o usuário possa visualizar previamente as ações realizadas pelo seu programa para uma aplicação específica.

Na Figura 14 item “a” ou menu bar **Simulador/simular** é para que seja inicializada a simulação do programa Ladder. É realizada após a edição do usuário e a não ocorrência de erros.

Na Figura 14 item “b” ou menu bar **Simulador/parar simulação**

FIGURA 14 – Ferramenta de início de simulação

Fonte: Elaborado pelo autor

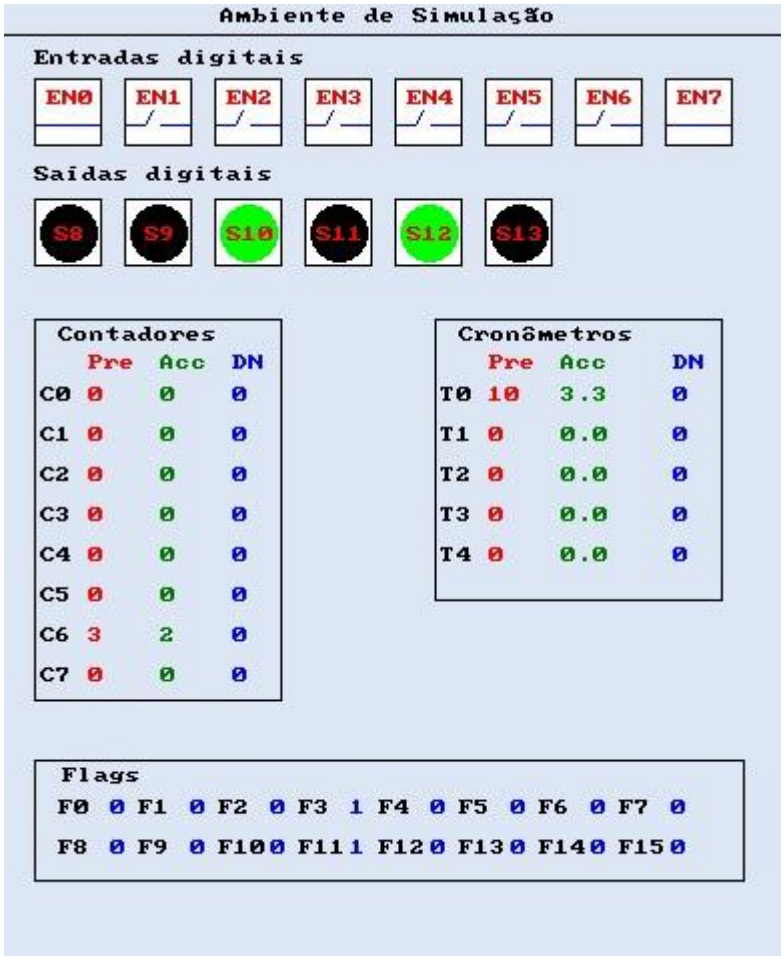
A Figura 15 apresenta o painel de simulação interativo.

A seguir será descrita a funcionalidade de cada componente do ambiente de simulação.

- **Entradas digitais -** EN0 até EN7 são botões virtuais que simulam chaves ligadas às entradas digitais disponíveis. A alteração do estado da chave é realizada pelo usuário através do clique do mouse. Quando a chave está fechada o estado da entrada é verdadeiro e quando aberta é falso;
- **Saídas digitais** - S8 até S13 são bobinas visuais, que simulam *led's* virtuais ligados às saídas digitais disponíveis. A alteração de estado dos *led's* é realizada através da lógica antecipadamente programada, sendo que quando o *led* está com a cor verde o estado da saída é verdadeiro ou alto e quando preto é falso ou baixo;
- **Contadores** - C0 até C8 são contadores visuais, que através de uma tabela simulam os valores dos atributos internos da instrução CTU. A alteração dos valores dos atributos é realizada através da lógica antecipadamente programada, sendo que em vermelho possui o valor do *Preset*, verde valor do *Accum* e azul o estado lógico do bit DN;
- **Cronômetros** - T0 até T4 são temporizadores visuais, que através de uma tabela, simulam os valores dos atributos internos da instrução TON. A Alteração dos valores dos atributos é realizada através da lógica antecipadamente programada, sendo que em vermelho possui o valor do *Preset*, verde valor do *Accum* e azul o estado lógico do bit DN;
- ***Flags*** - F0 até F15 são bobinas internas visuais que simulam o estado do bit das *flags.* A alteração é realizada através da lógica antecipadamente

programada, sendo que em azul possui seu estado lógico em1 quando verdadeiro e 0 quando falso.

FIGURA 9 – Painel de Simulação



Fonte: Elaborado pelo autor

2.4 Compilação Ladder

A Compilação do programa Ladder é realizada no momento de simulação do usuário, exportação do código C/C++ e ainda o usuário pode fazer a compilação prévia do programa através do botão da Figura 16.

FIGURA 10 – Botão de compilação

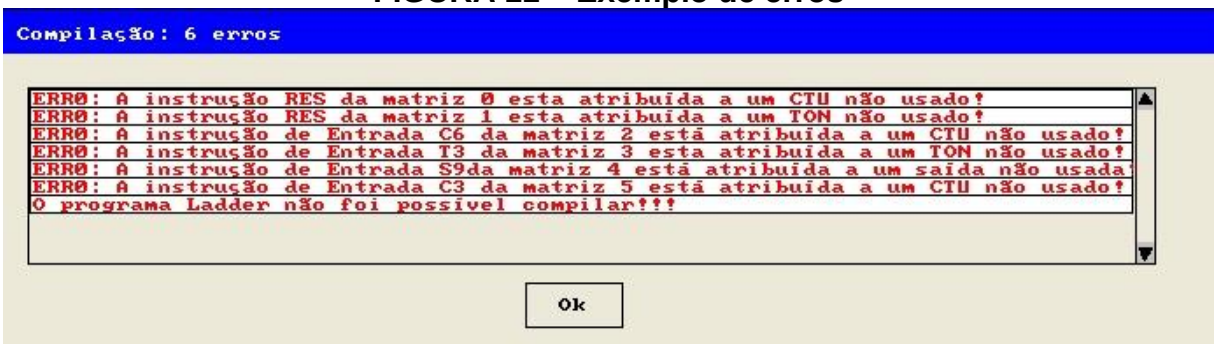
Fonte: Elaborado pelo autor

Os possíveis erros de implementação Ladder são:

- Instrução RES de reinício de um contador ou temporizador não utilizado;
- Multiplicidades de Temporizadores e Contadores, como instrução de saída;
- *Flags* e Saídas digitais utilizadas como NA ou NF, não utilizadas como instruções de saídas (OTE, OTL, OTU).

Na ocorrência de erros é exibido ao usuário um conjunto de mensagem com o número total de erros e suas descrições como exibido pela Figura 17.

FIGURA 11 – Exemplo de erros



Fonte: Elaborado pelo autor

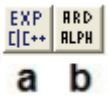
2.5 Geração de Código C/C++ para plataforma Arduino *Alpha*

Esta geração é um dos objetivos do CSLadder Mic, que gera o código objeto C/C++ equivalente ao programa fonte Ladder. Esta geração ocorre se caso não haja erro nos diagramas Ladder.

Para Exportar o código C/C++ é só clicar no botão da Figura 18 item “a” ou menu bar **compilador/Exportar C/C++**, onde será salvo no destino especificado o arquivo com a extensão PDE que é a utilizada pela IDE Arduino *alpha*.

Após o arquivo PDE ter sido exportado e se caso o compilador Arduino *Alpha* já estiver instalado no computador e todas as extensões PDE’s do Sistema operacional forem direcionadas para abrir o compilador Arduino *Alpha,* com o clique do botão da Figura 18 item “b” ou menu bar **compilador/Arduino Alpha** pode abrir o código objeto C/C++ na plataforma Arduino *Alpha*.

FIGURA 18 – Ferramenta de exportação de código C/C++



Fonte: Elaborado pelo autor

As entradas digitais foram padronizadas utilizando o resistor *pull-up,* onde em nível alto se deve ligar ao *ground* e o adverso é estar desconectado.

É necessário a utilização da biblioteca *csladdermic.h* , no apêndice A apresenta esta com suas implementações *csladdermic.cpp*, contida no apêndice B.

Para não utilizar em todos os projetos pode se criar uma pasta como o nome “csladdermic” dentro da pasta “libraries” da IDE Arduino, adicionando os *csladdermic.cpp e csladdermic.h,* não necessitando adicionar toda vez em um novo projeto.

3.0 Exemplo de exercícios de automatização implementados no CSLadder Mic

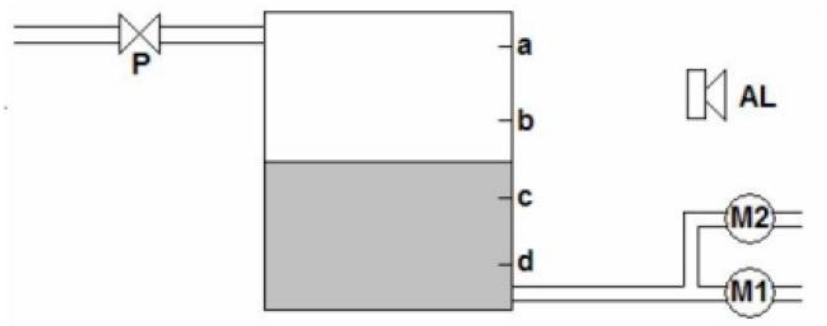
A seguir descreveremos alguns problemas de automatização que foram implementados no CSLadder MIC.

3.1 Sistema de Reservatório

> Implementar um programa em Ladder para o CLP em um sistema de reservatório composto de uma válvula de entrada P, duas bombas (acionadas por M1 e M2), um alarme AL e quatro sensores de nível (a, b, c, d), conforme ilustrado na Figura 19. As condições de funcionamento são as seguintes: se o nível for 'a', então fecha-se a válvula P. Se o nível for inferior a 'b', então abre-se a válvula P. Acima de 'b', M1 e M2 bombeiam. Abaixo de 'b', somente M1 bombeia. Abaixo de 'c', soa o alarme AL. Em 'd', nenhuma das bombas deverá funcionar (SILVEIRA; SANTOS, 2009, p.114-115).

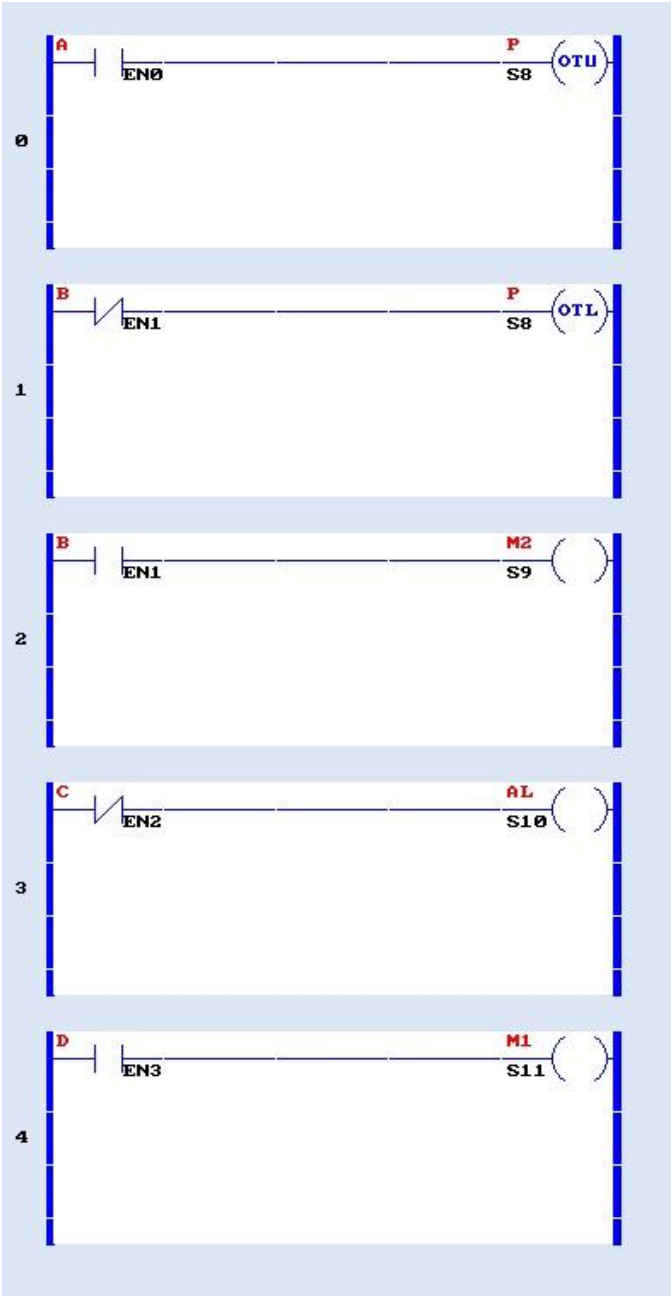
A Figura 20 apresenta a solução em LD para o problema.

FIGURA 19 - Sistema de Reservatório



Fonte: SILVEIRA; SANTOS, 2009.

FIGURA 20 - Programa fonte em Ladder para o Sistema de Reservatório



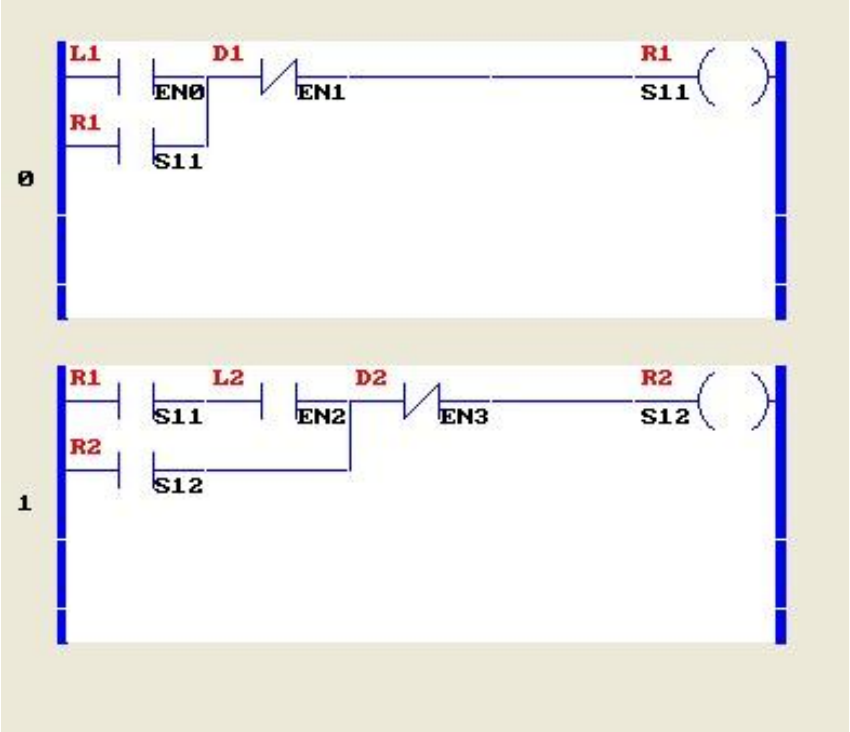
Fonte: Elaborado pelo autor

3.2 Relés com Dependência

> Elaborar um programa CLP para controlar dois relés (R1 e R2) de tal maneira que R1 pode atuar de forma independente e R2 só pode atuar se R1 estiver ligado, mas pode continuar ligado após o desligamento de R1. Os relés são ligados pelas botoeiras L1 e L2, e são desligados pelas botoeiras D1 e D2 (SILVEIRA; SANTOS, 2009, p.115).

A Figura 21 apresenta a solução em LD.

FIGURA 12 - Programa fonte em Ladder para os Relés com Dependência



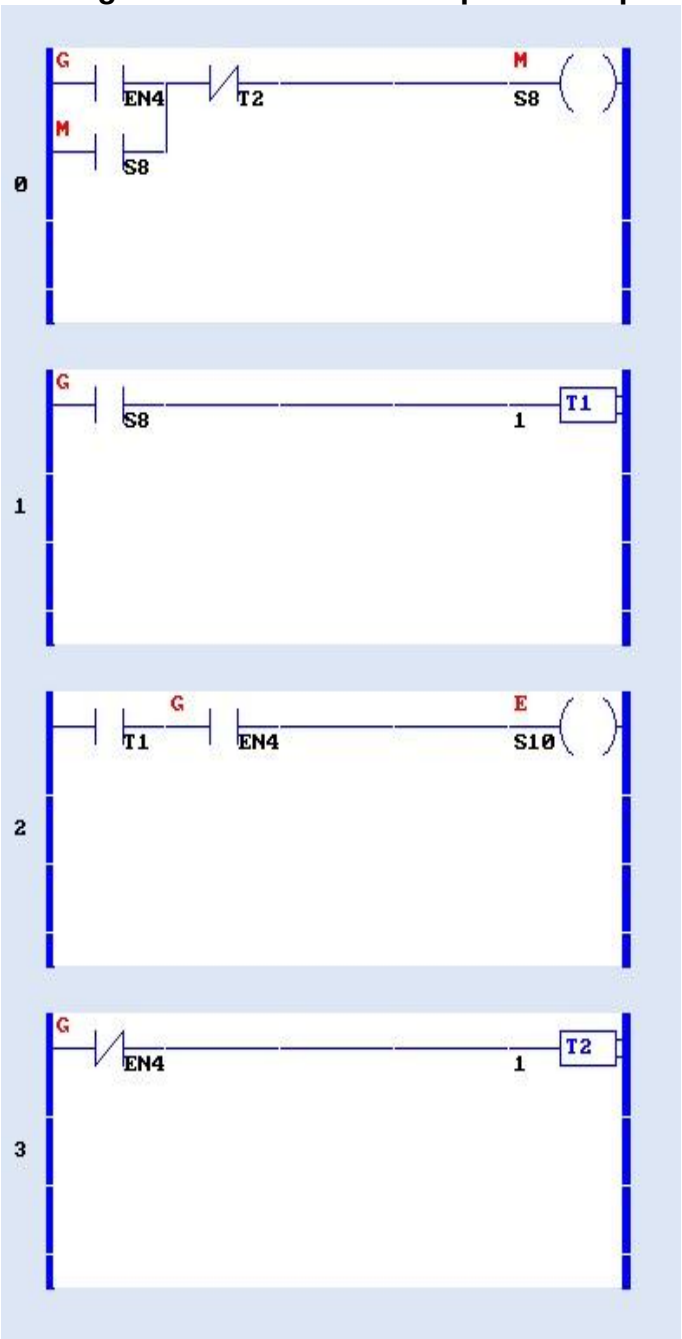
Fonte: Elaborado pelo autor

3.3 Máquina de Solda

> Em uma máquina de solda, há dois elementos controlados por um CLP: um contator (A) para fechamento do arco, e um relé (E) para avanço do motor do eletrodo. Quando o operador aciona o gatilho (G), a máquina deve entrar em funcionamento, atuando primeiramente o motor e 1 segundo após atuar o eletrodo. No momento em que o operador solta o gatilho, uma operação reversa deve ocorrer, ou seja, primeiramente desliga-se o eletrodo e após 1 segundo desliga-se o motor. Com base nestas informações elabore um programa CLP para realizar tal controle (SILVEIRA; SANTOS, 2009, p.116).

A Figura 22 apresenta a solução em Ladder para o problema proposto.

FIGURA 22 - Programa fonte em Ladder para a Máquina de Solda



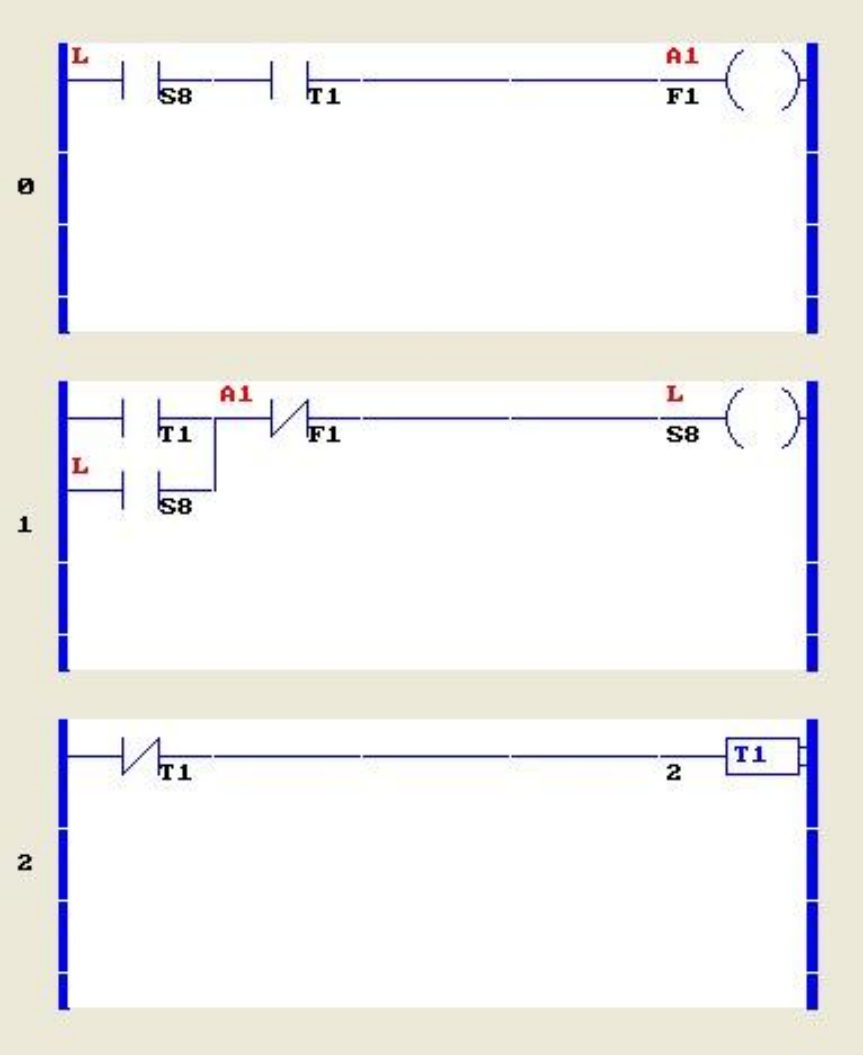
Fonte: Elaborado pelo autor

3.4 Lâmpada Intermitente

"Utilizando apenas um elemento temporizador, elabore um programa PLC capaz de acionar uma lâmpada de sinalização piscante com período de 2 segundos (SILVEIRA; SANTOS, 2009, p.116)".

A Figura 23 apresenta o LD para o problema proposto.

FIGURA 13 - Programa fonte em Ladder para a Lâmpada Intermitente



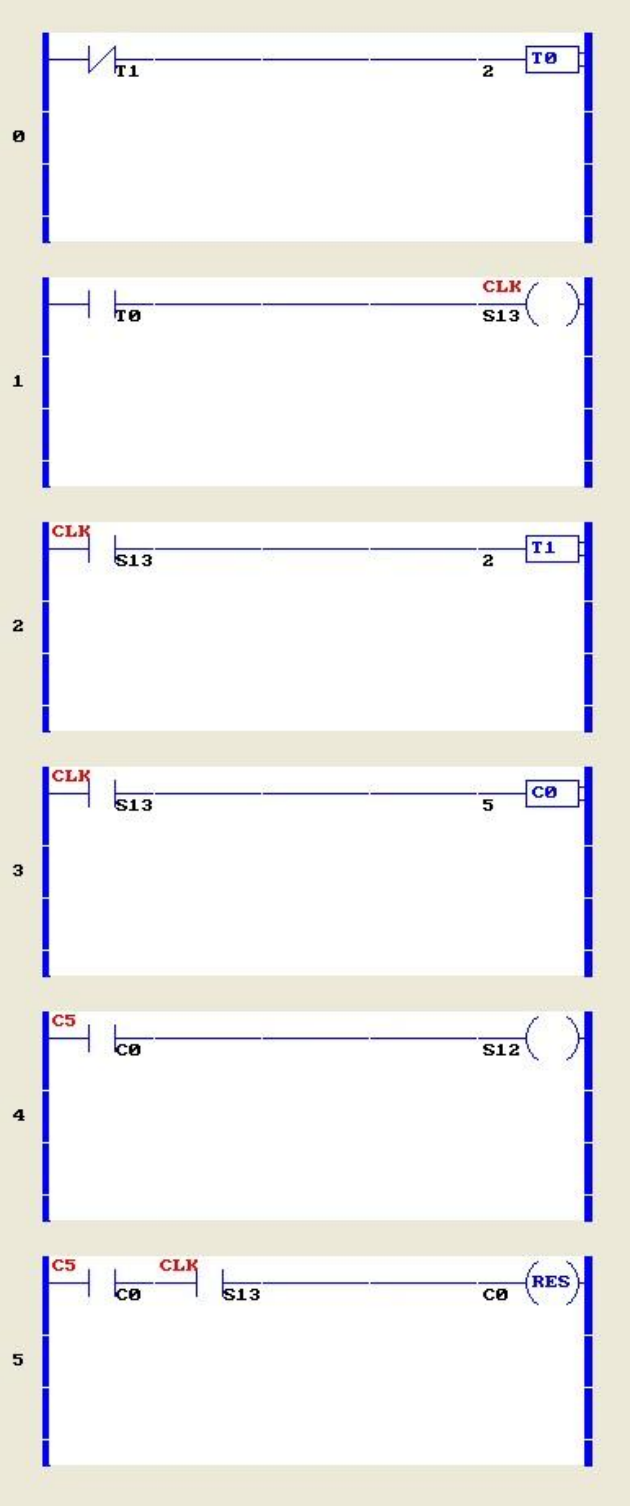
Fonte: Elaborado pelo autor

3.5 Contador de pulsos Módulo Cinco

> Utilizando-se dos recursos de contagem de CLP, elabore um programa capaz de acionar uma lâmpada sinalizadora sem que o números de pulsos recebidos em sua entrada for igual for múltiplo de 5 (cinco). Assim, no desfecho do quinto pulso a lâmpada acende, desligando-se no recebimento do sexto; novamente acende no décimo e desliga no décimo primeiro (SILVEIRA; SANTOS, 2009, p.116).

A Figura 24 apresenta o Ladder para o problema proposto.

FIGURA 14 - Programa fonte em Ladder para o Contador de Pulsos Módulo Cinco



Fonte: Elaborado pelo autor

4.0 REFERÊNCIAS

MORAES, Cicero C. **Engenharia de Automação Industrial**. 2. ed. Rio De Janeiro: LTCn, 2007. ISBN 85-216-1532-9.

ROCHA, Diego Silva Caldeira. **CSLADDER MIC:** compilador e simulador Ladder para microcontrolador ATMEGA328. 2012. 108 f. Monografia (Conclusão de curso) – Pontifícia Universidade Católica de Minas Gerais, Programa de Graduação em Engenharia de Computação, Belo Horizonte.

SILVEIRA, Paulo Rogério da; SANTOS, Winderson E. dos. **Automatização e Controle Discreto**. 3. ed. São Paulo: Érica, 2009. ISBN 9788571945913.

APÊNDICE A – BIBLIOTECA CSLADDERMIC.H

Este Apêndice apresenta o enunciado das classes da biblioteca *csladdermic.h*, utilizada para auxilio na geração de código objeto C/C++.

```
/*
BIBLIOTECA CSLADDERMIC.H
OBJETIVO: UTILZADA PARA GERAÇÃO DO CÓDIGO C++
AUTOR: DIEGO SILVA CALDEIRA ROCHA*/


#ifndef csladdermic_h
#define csladdermic_h
#include "WProgram.h" //BIBLIOTECA PADRÃO ARDUINO
#include "Bounce.h" // BIBLIOTECA PARA CONTROLE DE HISTERESE

//CLASSE Entrada PARA UTILIZAÇÃO DE ENTRADAS DIGITAIS
class Entrada{

private:
    bool sinal;//NIVEL ALTO OU BAIXO
    int pino;// VALOR DO PINO
    int Delay_read;//TEMPO DE HISTERESE

public:

    Entrada(int Pin, int D_r);//CONSTRUTOR
    void Setup();//CONFIGURAÇÃO INICIAL DA ENTRADA DIGITAL
    bool get_state();//RETORNA O SINAL DA ENTRADA
    int get_pino();//RETORNA O PINO UTILIZADO
    void Read(); //LEITURA DO SINAL DA ENTRADA

};

//CLASSE Saida PARA UTILIZAÇÃO DE SAIDAS DIGITAIS
class Saida{

private:
    bool sinal;//NIVEL ALTO OU BAIXO
    int pino;// VALOR DO PINO

public:
    Saida(int Pin);//CONSTRUTOR
    void Setup();//CONFIGURAÇÃO INICIAL DA SAÍDA DIGITAL
    bool get_state();//RETORNA O SINAL DA SAIDA
    int get_pino();//RETORNA O PINO UTILIZADO
    void Read();//LEITURA DO SINAL DA ENTRADA
    void set();//COLOCA A SAÍDA DIGITAL EM NÍVEL ALTO
    void reset(); //COLOCA A SAÍDA DIGITAL EM NÍVEL BAIXO
};

//CLASSE Flag PARA UTILIZAÇAO NA LÓGICA INTERNA
class Flag{
private:
    bool sinal;//NIVEL ALTO OU BAIXO
public:
    Flag(bool t);//CONSTRUTOR
    bool get_state();//RETORNA O SINAL DA FLAG
    void set(); //COLOCAO A FLAG EM NÍVEL ALTO
```

```
        void reset();//COLOCA A FLAG EM NÍVLE BAIXO
};

//CLASSE Ctu PARA UTILIZAÇAO DE CONTADORES
class Ctu{
private:
        bool sinal, //NIVEL ALTO OU BAIXO
                    sinal_ant;//GUARDA O SINAL ANTERIOR
        int presset,//ATE´QUAL VALOR SE DESEJA CONTAR
                    accum;//VALOR ATUAL DA CONTAGEM

public:
        Ctu(int PRST);//CONSTRUTOR
        bool get_state();//RETORNA O SINAL CONTADOR
        void set();//COLOCAO O CONTADOR EM NÍVEL ALTO
        void reset();//COLOCA O CONTADOR EM NÌVEL BAIXO
        void restart();//REINICIALIZA O CONTADOR
};

//CLASSE Ton PARA UTILIZAÇAO DE TEMPORIZADORES
class Ton{
private:
        bool sinal;//NIVEL ALTO OU BAIXO
        int presset;//ATE´QUAL VALOR SE DESEJA TEMPORIZAR

public:
        Ton(int PRST);//CONSTRUTOR
        void set();//COLOCAO O TEMPORIZADOR EM NÍVEL ALTO
        bool get_state();//RETORNA O SINAL DO TEMPORIZADOR
        void reset();//REINICIALIZA O TEMPORIZADOR

};#end
```

APÊNDICE B – BIBLIOTECA CSLADDERMIC.CPP

Este Apêndice apresenta as implementações dos métodos das classes da biblioteca *csladdermic.h*, utilizada para auxilio na geração de código objeto C/C++.

```
/*
    BIBLIOTECA CSLADDERMIC.CPP

    OBJETIVO: IMPLEMENTAÇÃO DAS CLASSES DA CSLADDERMC.HI

    AUTOR: DIEGO SILVA CALDEIRA ROCHA

    */

    #include "WProgram.h"  //BIBLIOTECA PADRÃO DO ARDUINO
    #include <csladdermic.h>//CABEÇARIO
    #include "Bounce.h" // BIBLIOTECA PARA CONTROLE DE HISTERESE

    /*IMPLEMENTAÇÃO DOS MÉTODS DA CLASSE Entrada*/

    //CONSTRUTOR PAR. 1 VALOR DO PINO, PAR. 2 VALOR DE ATRASO DE HISTERESE;
    Entrada :: Entrada(int Pin, int D_r)
    {
          pino=Pin;//SET PINO
          D_r=Delay_read;//ATRASO DE HISTERESE
          sinal=false; // TODA ENTRADA INICIALMENTE ESTA EM NÍVEL BAIXO

    }

    //INSTALAÇÕES INICIAIS DE ENTRADA
    void Entrada::Setup()
    {
          pinMode(pino, INPUT);//colocando o pino como entrada
          digitalWrite(pino, HIGH);//colocando o pino para o funcionamento DO reistor pull up ;
    }

    //RETORNA O ESTADO DA ENTRADA
    bool Entrada :: get_state()
    {
          return sinal;
    }

    //RETORNA O PINO DE ENTRADA
    int Entrada:: get_pino()
    {
          return pino;
    }

    //FAZ A LEITURA DA ENTRADA DIGITAL
    void Entrada::Read()
    {

          //CONTROLE DE HISTERESE
          Bounce bouncer =  Bounce( pino,Delay_read );
          bouncer.update ( );
          if(bouncer.read()==0)//pull-up inverte a logica
```

```
                sinal=true;
        else
                sinal=false;
}

/* IMPLEMENTAÇÃO DOS MÉTODoS DA CLASSE Saida*/

//CONSTRUTOR PAR. 1 VALOR DO PINO
Saida :: Saida(int Pin)
{
        pino=Pin;
        sinal=false;
        digitalWrite(pino, LOW);
}

//INSTALAÇÕES INICIAIS DE ENTRADA
void Saida ::Setup()
{
        pinMode(pino, OUTPUT);
}

//RETORNA O ESTADO DA SAÍDA
bool Saida:: get_state()
{
        return sinal;
}

//RETORNA O PINO DE SAÍDA
int Saida:: get_pino()
{
        return pino;
}

//COLOCA UMA SAÍDA DIGITAL EM NIVEL ALTO
void Saida::set()
{
        digitalWrite(pino, HIGH);
        sinal=true;
}

//COLOCA UMA SAÍDA DIGITAL EM NÍVEL BAIXO
void Saida::reset()
{
        digitalWrite(pino, LOW);
        sinal=false;
}

/* IMPLEMENTAÇÃO DOS MÉTODoS DA CLASSE Flag*/

//CONSTRUTOR PAR. 1 ESTADO INICIAL DA FLAG
Flag :: Flag(bool t )
{
        sinal=t;

}

//RETORNA O ESTADO DA FLAG
bool Flag :: get_state()
{
        return sinal;
```

```
}

//COLOCA A FLAG EM NÍVEL ALTO
void Flag :: set()
{
	sinal=true;
}

//COLOCA A FLAG EM NÍVEL BAIXO
void Flag :: reset()
{
	sinal=false;
}

/* IMPLEMENTAÇÃO DOS MÉTODoS DA CLASSE Ctu */

//CONSTRUTOR PAR 1 VALOR FIM DA CONTAGEM
Ctu :: Ctu(int PRST)
{
	presset=PRST;
	sinal=false;
	sinal_ant=false;
	accum=0;
}

//RETORNA O ESTADO DO CONTADOR
bool Ctu:: get_state()
{
	return sinal;
}

//COLOCA O ESTADO DO CONTADOR EM NÍVEL ALTO
void Ctu::set()
{
	sinal_ant=true;
}

//COLOCA O ESTADO DO CONTADOR EM NÍVEL BAIXO
void Ctu::reset()
{
	if(sinal_ant==true&&accum<presset)
	{
		accum++;
		sinal_ant=false;
	}
	if(accum==presset)
		sinal=true;

}

//REINICIA A CONTAGEM DO CONTADOR
void Ctu::restart()
{
	accum =0;
	sinal=false;
	sinal_ant=false;
}
/*IMPLEMENTAÇÃO DOS MÉTODoS DA CLASSE Ton*/
Ton :: Ton(int PRST)
{
```

```
        presset=PRST;
        sinal=false;

}

//RETORNA O ESTADO DO SINAL DO TEMPORIZADOR
bool Ton:: get_state()
{
        return sinal;
}

//COLOCA O TEMPORIZADOR EM NÍVEL ALTO
void Ton::set()
{
        sinal=true;
}

//COLOCAR O TEMPORIZADOR EM NÍVEL BAIXO
void Ton::reset()
{
        sinal=false;
 }
```