PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

CENTRO DE CIENCIAS EXATAS E DE TECNOLOGIA

ENGENHARIA DE COMPUTAÇÃO

PROJETO PLC

CURITIBA,

2010

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

CENTRO DE CIENCIAS EXATAS E DE TECNOLOGIA

ENGENHARIA DE COMPUTAÇÃO

Projeto apresentado como requisito para avaliação do programa de aprendizagem de Microprocessadores, do curso de graduação em Engenharia da Computação da Pontifícia Universidade Católica do Paraná, referente ao quinto período do mesmo. Com instrução do professore Afonso Ferreira Miguel. Tendo como alunos responsáveis pelo desenvolvimento: Carlos Alexandre Gouvêa da Silva, Edson Leonardo e Rafael Veiga.

CURITIBA,
2010

Abstract

THe PLC project aims to develop a PLC (Program Logic Control). Since a PLC is a device used to optimize systems, home automation, residential or industrial. With a PLC is possible to create several solutions in industrial automation systems, to reduce electric costs with industrial materials such as: relays, timers, counters, etc. And shorten the execution and assembly of electrical panels.
The PLC has been of great help in many areas of industry in general. Existing ones are used to automate industrial machinery in order to optimize its processes, in addition to promoting the safety of operators of machines. A practical example is a boxed food products, where a PLC embargoed. The PLC system must receive information, process and act on information.

Sumário

Índice de fotos

1. Resumo

O Projeto PLC tem como finalidade desenvolver um PLC (Program Logic Control). Sendo que um PLC é um equipamento utilizado para aperfeiçoar sistemas de automação residencial, predial ou industrial. Com um PLC é possível criar diversas soluções em sistemas de automação industrial, visando reduzir custos com matérias elétricos industriais, como por exemplo: reles, temporizadores, contadores, etc. E ainda diminuir o tempo na execução e montagem de painéis elétricos.

O PLC tem sido de grande ajuda em diversas áreas da indústria em geral. Os existentes são utilizados para automatizar maquinas industriais com a finalidade de aperfeiçoar seus processos, alem de promover a segurança de operadores de maquinas. Um exemplo pratico é uma encaixotadora de produtos alimentícios, onde existe um PLC embargado. O sistema PLC deverá receber uma informação, processar e atuar sobre a informação.

2. Objetivos

2.1 – Geral

Utilizando os conhecimentos adquiridos nas disciplinas de Eletrônica I e Microprocessadores I, construir um projeto que integre essas disciplinas e traga um entendimento prático e claro de cada recurso aprendido teoricamente. Criar um PLC, demonstrando suas características e aplicações praticas.

2.2 - Específicos

2.2.1.Trabalhar com circuitos processadores;

2.2.2. Desenvolvimento de um código em linguagem definida para atuação sobre processador Seeduino;

2.2.3. Trabalho e processamento de sinais de entrada para atuação em sistemas de controle.

3. Lista de componentes

1 REGULADOR DE TENSAO 7812 IC1

1 REGULADOR DE TENSAO 7805 IC2

2 DIODOS N4007 D1/D2

2 CAPACITORES ELETROLITICOS 1000uF C5/C6

1 CAPACITOR ELETROLITICO 2200uF C4

3 CAPACITORES CERAMICA 100nF C1/C2/C3

2 RELES 12V

LED’s AUTO BRILHO VERMELHO, AMARELO E VERDE

4. Utilidades

4.1 Lista de materiais, equipamentos e ferramentas

Fenolite;
Fios para conexão;
Multímetro digital;
Isopor;
Computador;
Transformador;
Seeduino processador Atmega;
2 sensores digitais;
Alicate bico e corte.

4.2. Softwares

Arduino Alpha – compilador código;
Eagle – layout placa de circuito impresso e esquemático;
Paint – desenho maquete rua

5. Descrição do Projeto

5.1. Passos iniciais

Para inicio do projeto teve-se inicio a um escopo geral do projeto, com levantamento dos equipamentos, circuitos e materiais necessários para o projeto. A aquisição da placa Seeduino foi a principal peça de desenvolvimento do projeto, pois essa será peça importantíssima para elaboração do código de implementação do projeto. Fez-se assim também a compra dos componentes do circuito regulador de tensão e isopor para maquete. O teste no transformador já possuído pela equipe sucedeu satisfatoriamente.

Figura 1 - Placa Seeduino

5.2. Maquete

Para demonstração prática do PLC, como já mencionado, foi desenvolvido um semáforo pratico. Para tal teve-se uso de isopor como base da maquete e sustentação do semáforo para as luzes do mesmo. Sobre o isopor base foi colocado um desenho desenvolvido em Paint onde indica a rua em si. Na própria base de isopor vários

furos foram dispostos para passagem dos fios e dois corte semi profundos para posicionar os sensores.

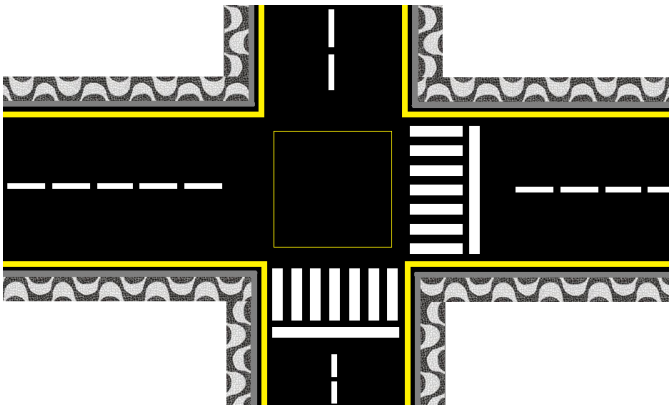


Figura 2 - Desenho Rua Paint

5.3. Circuitos elétricos

Após a compra do Seeduino e os componentes, tem-se a montagem do regulador de tensão com saída 5V para alimentação do Seeduino e 12V para alimentação do modulo de entrada dos sensores. Sendo que a fonte primaria utilizada é a da rede elétrica 127V e para sua utilização usou-se um transformador 0-15V conectado diretamente no regulador de tensão desenvolvido. Para controle dos sensores digitais tem-se feito um circuito simples de entrada composto somente por um rele, isso se deve, pois o sensor digital apenas disponibiliza apenas o sinal lógico 0, então como foi necessário a utilização do sinal 1 para reconhecimento do código através do Seeduino, o rele tem como finalidade mudar o sinal de 0 para 1.

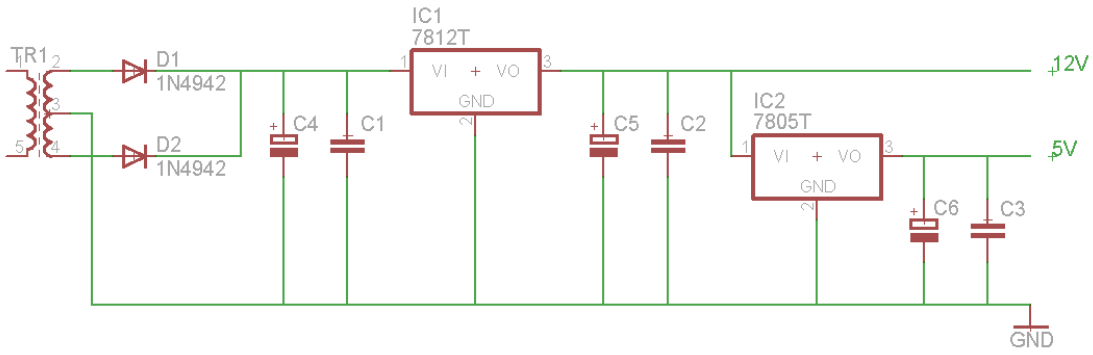


Figura 3 - Esquemático fonte

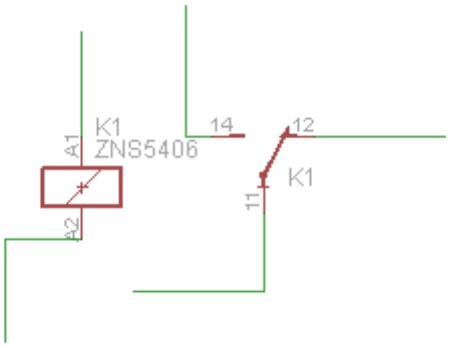


Figura 4 - Esquemático modulo de entrada

5.4. Esquemático Circuito Potencia e placas

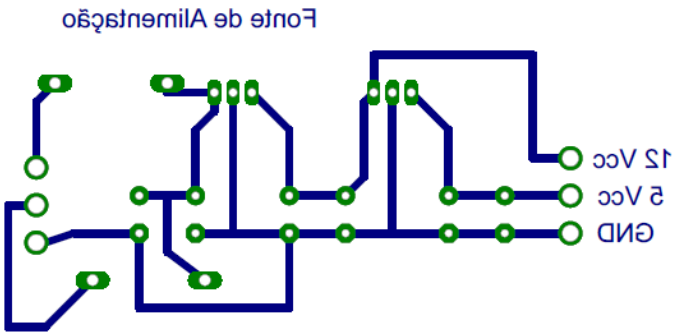


Figura 5 - Layout fonte

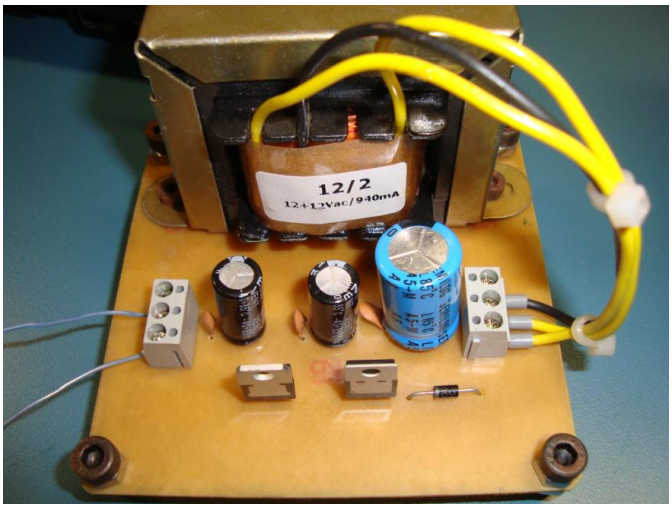


Figura 6 - Placa fonte

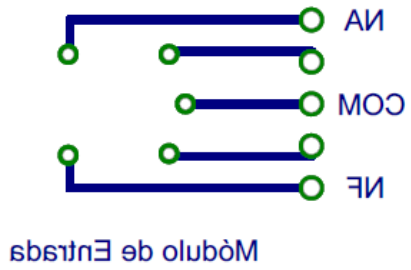


Figura 7 - Layout modulo de entrada

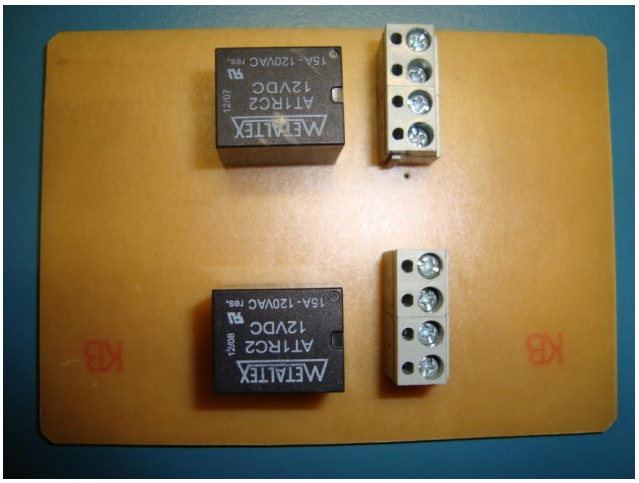


Figura 8 - Placa modulo de entrada

5.5. Desenvolvimento código

Para desenvolvimento do código utilizamos o próprio emulado de desenvolvimento de código disponibilizado pelo fabricante do Arduino (semelhante do Seeduino adquirido). No emulador do Arduino desenvolvemos o código em C, responsável por receber o sinal dos sensores e enviar as seqüências corretas dos led's a serem ascendidos.

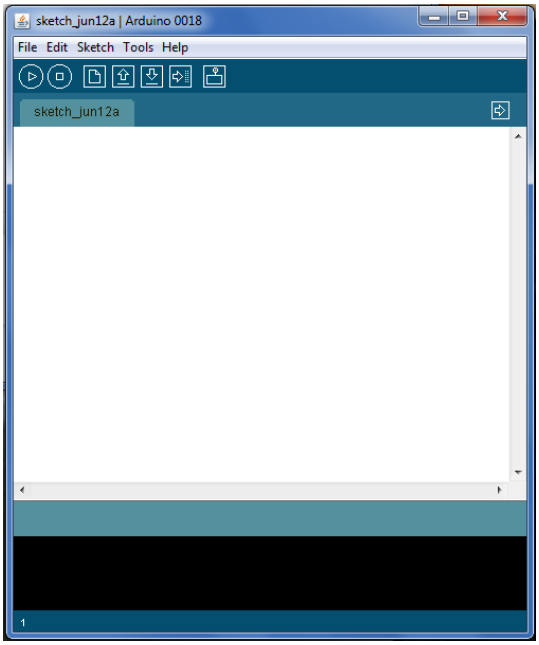


Figura 9 - Arduino Alpha

5.6. Código de controle

O código descrito abaixo desenvolvido em C, tem como estrutura base a um CLP. Tal estrutura teve que ser desenvolvida desta forma para que fosse atendido o objetivo inicial do projeto. O código trabalha com entradas e saídas externas, ficando em um loop infinito. A cada intervenção de um sinal de entrada enviado por um dos dois sensores digitais, o programa reage de tal forma que o mesmo recalcula um novo modo de mostrar ao usuário o led asceso. Ex: Se em

determinada pista o sinal esta verde porem não esta passando nenhum carro, e na outra pista este sinal vermelho porem existe um carro em cima do sensor da rua parado. Então o código decide que este sinal deve ser trocado, e assim o faz.

```
#include <stdio.h>
#include <stdlib.h>
#define nModules 62 // Número de módulos do CLP

        // Declaração type do Módulo
        const int ca   = 1; // Contato Aberto
        const int cf   = 2; // Contato Fechado
        const int temp = 3; // Temporizador
        const int out  = 4; // Saída

        // Delay para o temporizador
        const int tempoVd = 40000;  // 20 segundos - tempo luz verde
        const int tempoSs =  2000;  //  1 segundo  - intervalo de tempo para uma nova leitura do estado do sensor
        const int tempoPd =  1000;   //  0.5 segundos - intervalo de tempo entre: acionamento e temporizador,
                                    //desacionamento e temporizador (com isso o led pisca num intervalo de 1s)

        // Pinos do Seeeduino
        const int Sensor1 = 0;
        const int Sensor2 = 1;
        const int Vermelho1 = 11;
        const int Amarelo1  = 12;
        const int Verde1    = 13;
        const int Vermelho2 = 8;
        const int Amarelo2  = 9;
        const int Verde2    = 10;
        const int Pedestre1 = 7;
        const int Pedestre2 = 6;

        // Estado do sensor
        int sensor1State = 0;
        int sensor2State = 0;

void setup() {
        // Declara os pinos dos sensores como entrada
        pinMode(Sensor1, INPUT);
        pinMode(Sensor2, INPUT);
        // Declara os pinos dos leds como saída
        pinMode(Vermelho1, OUTPUT);
        pinMode(Vermelho2, OUTPUT);
        pinMode(Verde1, OUTPUT);
        pinMode(Verde2, OUTPUT);
```

```
        pinMode(Amarelo1, OUTPUT);
        pinMode(Amarelo2, OUTPUT);
        pinMode(Pedestre1, OUTPUT);
        pinMode(Pedestre2, OUTPUT);
        Serial.begin(9600);
}

void loop(){

        // Modulo
        typedef struct {
           unsigned int type;     // type     - tipo do módulo, pode ser: ca, cf, temporizador, out
           unsigned int state;    // state    - estado do módulo, se está sendo executado ou não
           unsigned int pin;      // pin      - pino seeeduino
           unsigned int delay_ms; // delay_ms - tempo do temporizador em milisegundos
           unsigned int led;      // led      - liga ou desliga
    } Module;

    // Declaração dos módulos do CLP
 Module M0, M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15, M16, M17, M18, M19, M20,
M21, M22, M23, M24, M25, M26, M27, M28, M29, M30, M31, M32, M33, M34, M35, M36, M37, M38, M39, M40,
M41, M42, M43, M44, M45, M46, M47, M48, M49, M50, M51, M52, M53, M54, M55, M56, M57, M58, M59, M60,
M61;

        // ---------------------------------
        //    Primeira parte do programa
        // ---------------------------------
        // Descrição:
        //   * Semáforo 1 - luz vermelha
        //   * Semáforo 2 - luz verde
        //   * Pedestre 1 - luz verde
        //   * Pedestre 2 - luz vermelha
        // ---------------------------------

        // Acionar luz vermelha semáforo 1
        M0.type = out;
        M0.pin = Vermelho1;
        M0.state = 0;
        M0.delay_ms = 0;
        M0.led = 1;

        // Acionar luz verde semáforo 2
        M1.type = out;
        M1.pin = Verde2;
        M1.state = 0;
        M1.delay_ms = 0;
        M1.led = 1;

        // Acionar luz vermelha semáforo pedestre 2
        M2.type = out;
        M2.pin = Pedestre2;
```

```
M2.state = 0;
M2.delay_ms = 0;
M2.led = 1;

// Desacionar luz vermelha semáforo pedestre 1
M3.type = out;
M3.pin = Pedestre1;
M3.state = 0;
M3.delay_ms = 0;
M3.led = 0;

// Temporizador 1 - Controla o tempo que os módulos (M0 até M3) ficam acionados
M4.type = temp;
M4.pin = 0;
M4.state = 0;
M4.delay_ms = tempoVd;
M4.led = 0;

// Contato Aberto Temporizador 1 - desabilita o temporizador
M5.type = ca;
M5.pin = 0;
M5.state = 0;
M5.delay_ms = 0;
M5.led = 0;

// ------------------------------------------------
//    Segunda parte do programa
// ------------------------------------------------
// Descrição:
//   * Semáforo 1 - luz vermelha
//   * Semáforo 2 - luz amarela
//   * Pedestre 1 - luz vermelha intermitente
//   * Pedestre 2 - luz vermelha
// ------------------------------------------------

// Desacionar luz verde semáforo 2
M6.type = out;
M6.pin = Verde2;
M6.state = 0;
M6.delay_ms = 0;
M6.led = 0;

// Acionar luz amarela semáforo 2
M7.type = out;
M7.pin = Amarelo2;
M7.state = 0;
M7.delay_ms = 0;
M7.led = 1;

// ------------------------------------------------------
//   Semáforo do pedestre 1 piscando (módulo 8 ao 27)
```

```
// ------------------------------------------------------

// Aciona luz vermelha semáforo pedestre 1 - primeiro pisca
M8.type = out;
 M8.pin = Pedestre1;
 M8.state = 0;
 M8.delay_ms = 0;
 M8.led = 1;

 // Temporizador pedestre
 M9.type = temp;
 M9.pin = 0;
 M9.state = 0;
 M9.delay_ms = tempoPd;
 M9.led = 0;

 // Desaciona luz vermelha do semáforo do pedestre 1
 M10.type = out;
 M10.pin = Pedestre1;
 M10.state = 0;
 M10.delay_ms = 0;
 M10.led = 0;

 // Temporizador pedestre
 M11.type = temp;
 M11.pin = 0;
 M11.state = 0;
 M11.delay_ms = tempoPd;
 M11.led = 0;

 // Acionar luz vermelha semárofo do pedestre 1 - segundo pisca
 M12.type = out;
 M12.pin = Pedestre1;
M12.state = 0;
M12.delay_ms = 0;
 M12.led = 1;

 // Temporizador pedestre
 M13.type = temp;
 M13.pin = 0;
 M13.state = 0;
 M13.delay_ms = tempoPd;
 M13.led = 0;

 // Desaciona luz vermelha do semáforo do pedestre 1
 M14.type = out;
 M14.pin = Pedestre1;
 M14.state = 0;
 M14.delay_ms = 0;
 M14.led = 0;
```

```
// Temporizador pedestre
M15.type = temp;
M15.pin = 0;
M15.state = 0;
M15.delay_ms = tempoPd;
M15.led = 0;

// Acionar luz vermelha do semáforo do pedestre 1 - terceiro pisca
M16.type = out;
M16.pin = Pedestre1;
M16.state = 0;
M16.delay_ms = 0;
M16.led = 1;

// Temporizador pedestre
M17.type = temp;
M17.pin = 0;
M17.state = 0;
M17.delay_ms = tempoPd;
M17.led = 0;

// Desaciona luz vermelha do semáforo do pedestre 1
M18.type = out;
M18.pin = Pedestre1;
M18.state = 0;
M18.delay_ms = 0;
M18.led = 0;

// Temporizador pedestre
M19.type = temp;
M19.pin = 0;
M19.state = 0;
M19.delay_ms = tempoPd;
M19.led = 0;

// Acionar luz vermelha do semáforo do pedestre 1 - quarto pisca
M20.type = out;
M20.pin = Pedestre1;
M20.state = 0;
M20.delay_ms = 0;
M20.led = 1;

// Temporizador pedestre
M21.type = temp;
M21.pin = 0;
M21.state = 0;
M21.delay_ms = tempoPd;
M21.led = 0;

// Desaciona luz vermelha do semáforo do pedestre 1
M22.type = out;
```

```
M22.pin = Pedestre1;
M22.state = 0;
M22.delay_ms = 0;
M22.led = 0;

// Temporizador pedestre
M23.type = temp;
M23.pin = 0;
M23.state = 0;
M23.delay_ms = tempoPd;
M23.led = 0;

// Acionar luz vermelha do semáforo do pedestre 1 - quinto pisca
M24.type = out;
M24.pin = Pedestre1;
M24.state = 0;
M24.delay_ms = 0;
M24.led = 1;

// Temporizador pedestre
M25.type = temp;
M25.pin = 0;
M25.state = 0;
M25.delay_ms = tempoPd;
M25.led = 0;

// Desaciona luz vermelha do semáforo do pedestre 1
M26.type = out;
M26.pin = Pedestre1;
M26.state = 0;
M26.delay_ms = 0;
M26.led = 0;

// Temporizador pedestre
M27.type = temp;
M27.pin = 0;
M27.state = 0;
M27.delay_ms = tempoPd;
M27.led = 0;

// ----------------------------
//    Fim pisca do pedestre
// ----------------------------

// Contato Aberto Temporizador Pedestre
M28.type = ca;
M28.pin = 0;
M28.state = 0;
M28.delay_ms = 0;
M28.led = 0;
```

```
// ---------------------------------
//    Terceira parte do programa
// ---------------------------------
// Descrição:
//   * Semáforo 1 - luz verde
//   * Semáforo 2 - luz vermelha
//   * Pedestre 1 - luz vermelha
//   * Pedestre 2 - luz verde
// ---------------------------------

// Desaciona luz vermelha do semáforo do pedestre 2
M29.type = out;
M29.pin = Pedestre2;
M29.state = 0;
M29.delay_ms = 0;
M29.led = 0;

// Aciona luz vermelha do semáforo do pedestre 1
M30.type = out;
M30.pin = Pedestre1;
M30.state = 0;
M30.delay_ms = 0;
M30.led = 1;

// Desacionar luz amarela do semáforo 2
M31.type = out;
M31.pin = Amarelo2;
M31.state = 0;
M31.delay_ms = 0;
M31.led = 0;

// Acionar luz vermelha do semáforo 2
M32.type = out;
M32.pin = Vermelho2;
M32.state = 0;
M32.delay_ms = 0;
M32.led = 1;

// Desacionar luz vermelha do semáforo 1
M33.type = out;
M33.pin = Vermelho1;
M33.state = 0;
M33.delay_ms = 0;
M33.led = 0;

// Acionar luz verde do semáforo 1
M34.type = out;
M34.pin = Verde1;
M34.state = 0;
M34.delay_ms = 0;
M34.led = 1;
```

```
// Temporizador 1 - Controla o tempo que os módulos (M29 até M34) ficam acionados
M35.type = temp;
M35.pin = 0;
M35.state = 0;
M35.delay_ms = tempoVd;
M35.led = 0;

// Contato Aberto Temporizador 3
M36.type = ca;
M36.pin = 0;
M36.state = 0;
M36.delay_ms = 0;
M36.led = 0;

// ------------------------------------------------
//    Quarta parte do programa
// ------------------------------------------------
// Descrição:
//   * Semáforo 1 - luz verde
//   * Semáforo 2 - luz amarela
//   * Pedestre 1 - luz vermelha
//   * Pedestre 2 - luz vermelha intermitente
// ------------------------------------------------

// Desacionar luz verde do semáforo 1
M37.type = out;
M37.pin = Verde1;
M37.state = 0;
M37.delay_ms = 0;
M37.led = 0;

// Acionar luz amarela semáforo 1
M38.type = out;
M38.pin = Amarelo1;
M38.state = 0;
M38.delay_ms = 0;
M38.led = 1;

// Pisca do Pedestre

// Acionar luz vermelha do semáforo do pedestre 2
M39.type = out;
M39.pin = Pedestre2;
M39.state = 0;
M39.delay_ms = 0;
M39.led = 1;

// Temporizador pedestre
M40.type = temp;
M40.pin = 0;
```

```
M40.state = 0;
M40.delay_ms = tempoPd;
M40.led = 0;

// Desaciona luz vermelha do semáforo do pedestre 2
M41.type = out;
M41.pin = Pedestre2;
M41.state = 0;
M41.delay_ms = 0;
M41.led = 0;

// Temporizador pedestre
M42.type = temp;
M42.pin = 0;
M42.state = 0;
M42.delay_ms = tempoPd;
M42.led = 0;

// Acionar luz vermelha do semáforo do pedestre 2
M43.type = out;
M43.pin = Pedestre2;
M43.state = 0;
M43.delay_ms = 0;
M43.led = 1;

// Temporizador pedestre
M44.type = temp;
M44.pin = 0;
M44.state = 0;
M44.delay_ms = tempoPd;
M44.led = 0;

// Desaciona luz vermelha do semáforo do pedestre 2
M45.type = out;
M45.pin = Pedestre2;
M45.state = 0;
M45.delay_ms = 0;
M45.led = 0;

// Temporizador pedestre
M46.type = temp;
M46.pin = 0;
M46.state = 0;
M46.delay_ms = tempoPd;
M46.led = 0;

// Acionar luz vermelha do semáforo do pedestre 2
M47.type = out;
M47.pin = Pedestre2;
M47.state = 0;
M47.delay_ms = 0;
```

```
M47.led = 1;

// Temporizador pedestre
M48.type = temp;
M48.pin = 0;
M48.state = 0;
M48.delay_ms = tempoPd;
M48.led = 0;

// Desaciona luz vermelha do semáforo do pedestre 2
M49.type = out;
M49.pin = Pedestre2;
M49.state = 0;
M49.delay_ms = 0;
M49.led = 0;

// Temporizador pedestre
M50.type = temp;
M50.pin = 0;
M50.state = 0;
M50.delay_ms = tempoPd;
M50.led = 0;

// Acionar luz vermelha do semáforo do pedestre 2
M51.type = out;
M51.pin = Pedestre2;
M51.state = 0;
M51.delay_ms = 0;
M51.led = 1;

// Temporizador pedestre
M52.type = temp;
M52.pin = 0;
M52.state = 0;
M52.delay_ms = tempoPd;
M52.led = 0;

// Desaciona luz vermelha do semáforo do pedestre 2
M53.type = out;
M53.pin = Pedestre2;
M53.state = 0;
M53.delay_ms = 0;
M53.led = 0;

// Temporizador pedestre
M54.type = temp;
M54.pin = 0;
M54.state = 0;
M54.delay_ms = tempoPd;
M54.led = 0;
```

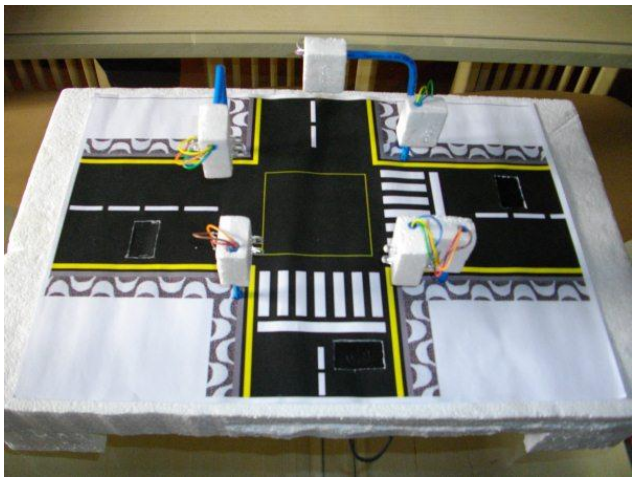
```
 // Acionar luz vermelha do semáforo do pedestre 2
 M55.type = out;
 M55.pin = Pedestre2;
 M55.state = 0;
 M55.delay_ms = 0;
 M55.led = 1;

 // Temporizador pedestre
 M56.type = temp;
 M56.pin = 0;
 M56.state = 0;
M56.delay_ms = tempoPd;
 M56.led = 0;

 // Desaciona luz vermelha do semáforo do pedestre 2
 M57.type = out;
 M57.pin = Pedestre2;
 M57.state = 0;
 M57.delay_ms = 0;
 M57.led = 0;

 // Temporizador pedestre
 M58.type = temp;
 M58.pin = 0;
 M58.state = 0;
 M58.delay_ms = tempoPd;
 M58.led = 0;
 // -- Fim pisca do pedestre

 // Contato Aberto temporizador pedestre
 M59.type = ca;
 M59.pin = 0;
 M59.state = 0;
 M59.delay_ms = 0;
 M59.led = 0;

 // Desacionar luz amarela do semáforo 1
 M60.type = out;
 M60.pin = Amarelo1;
 M60.state = 0;
 M60.delay_ms = 0;
 M60.led = 0;

 // Desacionar luz vermelha do semáforo 2
 M61.type = out;
 M61.pin = Vermelho2;
 M61.state = 0;
 M61.delay_ms = 0;
 M61.led = 0;

// Array com todos os módulos do CLP
```

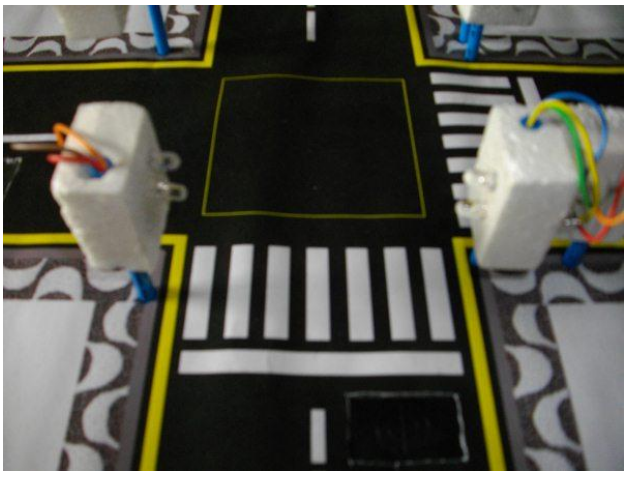
```
Module modules[62] = {M0, M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15, M16, M17,
M18, M19, M20, M21, M22, M23, M24, M25, M26, M27, M28, M29, M30, M31, M32, M33, M34, M35, M36, M37,
M38, M39, M40, M41, M42, M43, M44, M45, M46, M47, M48, M49, M50, M51, M52, M53, M54, M55, M56, M57,
M58, M59, M60, M61};

        // Loop infinito para o sinaleiro
        for(;;) {
          // Laço para executar todos os módulos do array
           for ( int i=0; i<nModules; i++ ) {

        // -- Variáveis Auxiliares -- //
        int ind = 0;              // armazena o indice do módulo anterior
        int tempof = 0;           // armazena o tempo em segundos do temporizador
        int tempoSemCarro1 = 0;   // tempo que o sensor1 fica desacionado
        int tempoSemCarro2 = 0;   // tempo que o sensor2 fica desacionado

        // Verifica o tipo do módulo que será executado: ca, cf, temp ou out
        switch (modules[i].type) {
           case 1: // Contato Aberto

              // Inicializa ind com o indice do módulo anterior
              // Se o módulo em execução for o primeiro (Module0), ind recebe o indice do ultimo módulo que foi
              //executado (Module61, indice 61)
            if ( i == 0 ) ind = nModules - 1;

              // Senão recebe o indice do módulo anterior
            else        ind = i - 1;

              // Altera o estado do módulo para 1 (módulo sendo executado)
              //modules[i].state = 1;

              // Altera o estado de um grupo de módulos que já foram executados para 0 até encontrar um
              // módulo do tipo ca
            while ( modules[ind].type != ca ) {
               modules[ind].state = 0;
               ind -= 1;
               // Se o indice do módulo anterior for igual a -1, isto significa que a variável ind deve receber o indice
               //do ultimo módulo do array
               if ( ind == -1 ) {
                  ind = nModules - 1;
               }
            }
            modules[i].state = 0;
            break;

           case 2: // Contato fechado
            modules[i].state = 1;
            break;

           case 3: // Temporizador
            // teste programa
```

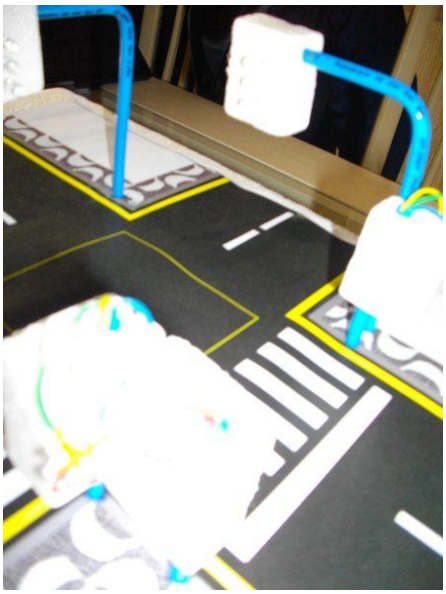
```
// sensor2State = digitalRead(Sensor2);
// Serial.print(sensor1tate,DEC);
// Serial.print(sensor2State,DEC);

// Altera o estado do módulo para 1 (módulo sendo executado)
modules[i].state = 1;

// Converte o tempo (milisegundos -> segundos)
tempof = (modules[i].delay_ms)/2000;

   // Se tempof = 20 significa que um semáforo está com o sinal verde ligado, portanto deve-se
   //verificar o estado dos sensores
if ( tempof == 20 ) {

  // Loop para verificar estado do sensor no intervalo de tempo do temporizador
    for ( int j=0; j<tempof; j++ ) {

       // Delay de 1s
       delay(tempoSs);

       // Lê estado dos sensores ( HIGH - desacionado, LOW - acionado )
       // Obs: as saídas são invertidas, por isso HIGH é desacionado e LOW acionado (apenas para os
       //sensores)
       sensor1State = digitalRead(Sensor1);
       sensor2State = digitalRead(Sensor2);

       // Atualiza as variáveis referentes ao tempo que os sensores estão desacionados
       // Para o sensor 1
       if ( sensor1State == HIGH )
                tempoSemCarro1 += 1;
       else
                tempoSemCarro1 = 0;

       // Para o sensor 2
       if ( sensor2State == HIGH )    tempoSemCarro2 += 1;
       else                           tempoSemCarro2 = 0;

       //  ------------------------------------------------   Acionamento dos Sensores   ----------------------------------
       //    O tempo de acionamento de um semáforo será afetado quando as três condições abaixo
       //forem satisfeitas:
       //       1 - Tiver decorrido um tempo de 5s desde que foi acionado a luz vermelha do semáforo x
       //       2 - Durante 3s não passar nenhum carro pelo semáforo y
       //       3 - Ter carro no semáforo x parado
       //    Se estas condições forem satisfeitas o sinal y amarela
       //    Obs: Após 15s desde que o sinal vermelho do semáforo x foi acionado, os sensores não
       //interferem no tempo de acionamento
       //-------------------------------------------------------------------------------------------------------------------

       // Condições para alterar o tempo de atuação de um semáforo
       // Verifica se está dentro do período de tempo para alterar o tempo de atuação
       if ( (j > 5) && (j < 15) ) {
```

```
                    //  ---------------------------------------------------------------------------------------------------------------
                    //  Verifica se:
                    //     1 - sensor x acionado
                    //     2 - sensor y desacionado
                    //     3 - semáforo x está com a luz vermelha acionada
                    //     4 - tempo sem passar carro no semáforo y igual a 3s
                    //  ---------------------------------------------------------------------------------------------------------------
                    if(  (modules[0].state  ==  1)  &&  (sensor1State  ==  LOW)  &&  (sensor2State  ==  HIGH)  &&
(tempoSemCarro2 >= 3) ) {
                       i = 4; // Pula para o Módulo 5 - Contato aberto temporizador 1 e logo após executa a
                              //segunda parte do programa
                       break;
                    }

                    if( (modules[32].state == 1) && (sensor1State == HIGH) && (sensor2State == LOW) &&
                          (tempoSemCarro1 >= 3) ) {
                       i = 35; // Pula para o Módulo 36 - Contato aberto temporizador
                       break;
                    } // end if
                  }
                }
              }

              //  Demais temporizadores
              else   delay(modules[i].delay_ms);
              break;

            case 4: // Saida
              modules[i].state = 1;
              // Verifica se o led deve ser ligado ou apagado
              if (modules[i].led == 1)
                 digitalWrite(modules[i].pin, HIGH); // Acende led semáforo
              else
                 digitalWrite(modules[i].pin, LOW);  // Apaga led semáforo
              break;
          }
        }
      }
}
```

6. Outras Imagens








LEVI'S

7. Conclusão

Assim, após a conclusão do projeto, chegou a conclusões que através dos conhecimentos pratico ou teórico adquiridos das disciplinas, pode-se desenvolver e implementar um projeto integrado. Além de agregar conhecimentos novos na resolução rápida e pratica de problemas que podem ocorrer em um sistema ou máquina.

Atendemos aos critérios exigidos para construção do CLP. Buscaram-se todas as informações teóricas possíveis e necessárias para um bom resultado no termino do mesmo. A fase mais difícil do projeto foi o desenvolvimento do código pro CLP, pois ele envolve um loop infinito onde que a interferência de um sinal externo (sensores), deve alterar automaticamente todo o processo de envio de sinais aos led’s.

8. Referencias Bibliográficas

Emulador arduino
http://www.arduino.cc/

CLP
http://www.impac.com.br/

SLC 500 System
http://www.rockwellautomation.com/