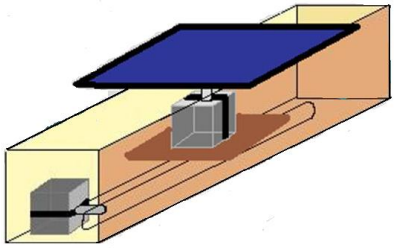
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

CENTRO DE CIÊNCIAS EXATS E TECNOLÓGICAS

ENGENHARIA DE COMPUTAÇÃO



PLATAFORMA MULTIFUNCIONAL

CURITIBA

2009

JORGE HENRIQUE WERNECK GOMES

LAURA WOBETO

LEONARDO ALVES FERREIRA

MARCELO JÚNIOR COSSETIN

PLATAFORMA MULTIFUNCIONAL

Documentação apresentada ao curso de Engenharia de Computação (Turma U - Matutino) do Centro de Ciências Exatas e Tecnológicas da Pontifícia Universidade Católica do Paraná, como critério de avaliação do Projeto Integrado I, sob a orientação do Prof. AFONSO FERREIRA MIGUEL, e Prof. GIL MARCOS JESS.

CURITIBA

2009

AGRADECIMENTOS

Somos muito gratos a todos aqueles que reservaram um tempo para nos ajudar durante esta caminhada, dentre esses professores, família, amigos, colegas de curso, colegas de trabalho, funcionários da PUC, entre outros. Em adição, gostaríamos de agradecer de modo especial ao :

Professor Afonso Ferreira Miguel, responsável pela verificação semanal da evolução do projeto, sempre disponibilizando-nos materiais de apoio para a conclusão, nos auxiliando nos momentos de dúvidas, perante as diversas dificuldades encontradas, sempre como uma visão mais abrangente, nos ajudando nas definições gerais desde software ao hardware, sempre nos lembrando que colhemos frutos do que plantamos.

Professor Gil Marcos Jess, responsável pela modulação do projeto de maneira inicial, nos mostrando caminhos otimizados para a conclusão, não se interessando apenas como um projeto em si, mas o que ele poderá se tornar futuramente.

Professor Edson José Pacheco, responsável pela no auxílio na parte de programação orientada a objeto, nos ajudando nas definições das linguagens C++ e C#, nos indicando também uma utilização mais otimizada para o editor gráfico Windows Form.

Á todos o nosso muito obrigado !

Jorge Henrique Werneck Gomes
Laura Wobeto
Leonardo Alves Ferreira
Marcelo Junior Cossetin

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

O projeto desenvolvido durante o 3° período do Curso de Engenharia da Computação, se resume em uma plataforma multifuncional (muito utilizadas em linhas de montagem), operando de forma a movimentar-se horizontalmente sobre um trilho e capaz de realizar o movimento rotacional de 360° em qualquer ponto eixo horizontal. Esta é composta por duas estruturas principais, o hardware e o software, para desenvolvimentos dessas estruturas foi-se necessário tanto um conhecimento da linguagem C ++ como um conhecimento físico para desenvolvimento da estrutura controlada pelo software.

Como a principal característica de um projeto é sua limitação no tempo, usamos de um cronograma desenvolvido pela equipe para otimizar o tempo e atingir os objetivos dentro do prazo estipulado. Assim dividimos em etapas o processo projetual, listadas abaixo:

-Ante-projeto (Plano de Trabalho, Especificação de requisitos)

-Projeto (Projeto (Lógico) Software, Projeto (Físico) Hardware, Projeto Mecânico)

-Implementação (Software, Hardware, Mecânico).

Por fim com todo o projeto em mãos passamos para etapa de testes, onde obtemos os resultados esperados, preparando então esta documentação a qual apresenta tanto os passos do desenvolvimento do projeto detalhadamente como os resultados desse processo.

2 JUSTIFICATIVAS

Uma plataforma em uma linha de montagem agiliza muito o trabalho e gera uma grande economia de tempo. As plataformas existentes no mercado de hoje quase todas apenas se locomovem em duas direções, para frente e para trás.

Para aumentar ainda mais a eficiência das plataformas em uma linha de montagem resolvemos desenvolver uma que não apenas locomova-se para frente e para trás, mas também que faça rotações, o que permitirá ao montador a ter acesso os dois lados da peça.

Com a plataforma multifuncional um empregado faz o trabalho de dois, o que acaba reduzindo o número de empregados e aumentando os lucros da empresa.

3 METODOLOGIA

Primeiramente foram montados os circuitos: Conversor RS232, dois módulos capazes de controlar motores de passo, e duas etapas de potencia diferentes como já foi mencionado. Todos os circuitos foram montados na ordem citada a cima e testados à medida que foram sendo construídos. Tudo foi feito em protoboard para depois serem produzidos em placas de maneira definitiva.

Com todos os circuitos devidamente testados, passamos para a parte da montagem da estrutura, que foi feita de maneira simultânea com o desenvolvimento do software.

A estrutura foi desenvolvida a partir de um trilho de impressora antiga que se adequava a proposta do projeto, o trilho e as placas com os circuitos foram fixados a outra estrutura que foi feita com madeira para a melhor observação do projeto.

Para o desenvolvimento do software, foi realizada primeiramente a parte manual que foi simples, pois não envolvia a parte do windowns form e características do C#, depois foi feita a parte automática que exigia todas as interações citadas à cima, o que levou a ser uma parte muito trabalhosa.

Por ultimo foi feita a documentação do projeto e com isso a sua conclusão.

4 RESPONSABILIDADES

Para que o projeto obtivesse este sucesso, fez-se necessário durante todo o seu desenvolvimento a participação ativa de todos os participantes do grupo e também dos professores, sendo exigido que cada um muita responsabilidade, seriedade e muita força de vontade em todos os eixos. Cada integrante teve a sua responsabilidade desempenhando-a com o máximo de comprometimento. Os professores estiveram aptos a responder todas as nossas dúvidas em relação ao projeto nos ajudando e ofertando-nos novas idéias.

Podemos contar com as estruturas da Pontifícia Universidade Católica do Paraná – PUCPR, sendo uma das responsabilidades essenciais, pois são nos laboratórios com os devidos equipamentos que conseguimos levar o projeto adiante.

5 OBJETIVO

O objetivo principal é proporcionar ao usuário manipulação da plataforma informando um número inteiro através de um software no qual distância deverá ser percorrida no eixo horizontal e um ângulo para rotação da plataforma. Para distância haverá um limite a ser seguido.

6 COMPONENTES

Antes de definir o projeto, estaremos passando especificações sobre os componentes usados para o desenvolvimento do projeto.

6.1 Resistor

Basicamente a função de um resistor para a física é a transformação de energia elétrica em energia térmica, porem acaba que por sua vez controlando a intensidade de corrente elétrica evitando que outros componentes estraguem.

Figura 1 : Resistor

6.2 Capacitor

Um capacitor é um componente elétrico passivo que pode armazenar energia em um campo elétrico, entre um par de condutores (camados "placas"). O processo de armazenamento de energia no capacitor é conhecido como "carregamento" e envolve cargas elétricas de igual magnitude, mas polaridade oposta, acumulando-se em cada uma das placas. A habilidade de um capacitor em armazenar carga é medida pela sua capacitância, em unidades de farads. Capacitores são frequentemente usados em circuitos elétricos e eletrônicos, como dispositivos de armazenamento. Eles também podem ser usados para diferenciar sinais de alta e baixa frequências. Esta propriedade os torna úteis como filtros em eletrônica. Na prática, os capacitores possuem resistências internas, vazamento de carga, indutância e outras propriedades não ideais, não encontradas em um capacitor teórico, ideal.

Figura 2 : Capacitor

6.3 Transistor

Um transistor funciona com o mesmo princípio de um registro de água em que, ele irá controlar o fluxo de elétrons:

- O transistor é constituído de três pinos chamados, emissor, coletor e base;
- O pino "base" é que controla o fluxo de corrente (como o registro de água) do emissor e coletor;
- Uma pequena corrente na base faz-se com que aumenta-se o fluxo entre o coletor e base. Voltando ao registro de água, um pequeno giro no registro, logo, aumenta-se muito a saída da intensidade de água.

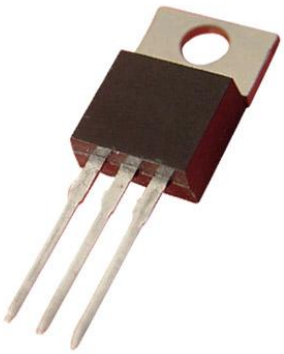


Figura 3 : Transistor

6.4 Circuitos Integrados

6.4.1 MAX 232

O MAX 232 é um circuito integrado conversor de nível, sua função é converter sinais TTL em RS232(Consiste em um padrão para troca serial de dados binários entre um DTE (terminal de dados)e um DCE (comunicador de dados.)) e virse-versa, além disso, ele fornece uma ótima rejeição de ruído e é mais resistente à descargas e curtos.

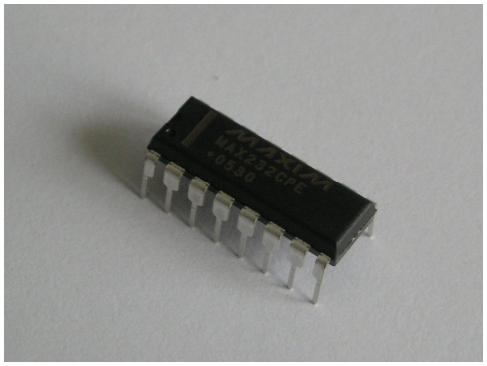


Figura 4 : MAX232

6.4.2 PIC16F629

O PIC12F629 nada mais é do que um microcontrolador - Os microcontroladores PIC têm famílias com núcleos de processamento de 12 bits, 14 bits e 16 bits e trabalham em velocidades de 0kHz (ou DC) a 48MHz, usando ciclo de instrução mínimo de 4 períodos de clock, o que permite uma velocidade de no máximo 10 MIPS. Há o reconhecimento de interrupções tanto externas como de periféricos internos. Funcionam com tensões de alimentação de 2 a 6V e os modelos possuem encapsulamento de 6 a 100 pinos em diversos formatos.

Figura 5 : PIC

6.4.3 ULN2803A

O ULN2803A nada mais é do que um circuito integrado que possue 8 disposições de DARLINGTON (com 8 saidas), o que nos permite montar uma etapa de potencia que controla até dois motores de passo com corrente inferior a 0,5A.

7 PROJETO – PLATAFORMA MULTIFUNCIONAL

O projeto é composto por apenas uma unidade física (hardware), sendo este controlado pela a parte lógica (software) do projeto.

A unidade física é composta por uma estrutura interna de uma impressora, dois motores de passos, e cinco circuitos previamente montados em placas, sendo estes dois circuitos controladores de motores de passo, que neste referenciamos como módulo M0, um conversor RS232 –TTL, e duas etapas de potência , sendo cada uma ofertadora de energia para cada um dos motores. A energização ocorre por três fontes independentes, sendo duas ligadas as etapas de potências e a outra ligada ao circuito conversor.

A unidade lógica foi desenvolvida em linguagem C++, porém como sua interface foi feita através do editor gráfico Windows Form, tivemos que fazer interações com a linguagem C#, pois esta é a linguagem base do Windows Form. A conexão entre a parte física e a lógica é feita através de um cabo serial, que se conecta com o computador e com a unidade física.

Na figura A1. encontra-se uma representação gráfica de como ficou a disponibilização do projeto.

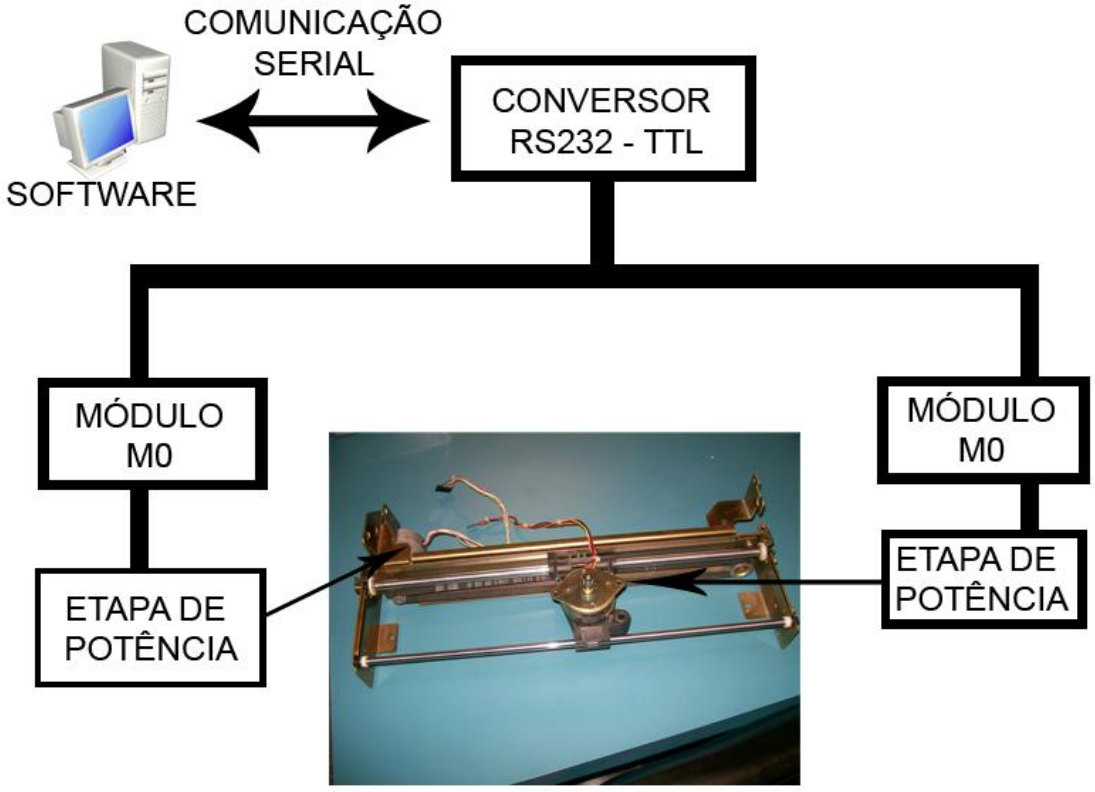


Figura 6 : Diagrama de Blocos PLATAFORMA MULTIFUNCINAL

7.1 Estrutura da Impressora

Para que não fosse necessária a criação de uma estrutura autentica, foi utilizada a estrutura de uma impressora antiga.

Esta estrutura se baseia em uma correia ligada a um eixo disponibilizado por um dos motores de passo, e outro eixo que tem funcionalidade apenas como um rolamento. Ao energizar o motor, a correia prática o movimento em torno dos eixos.

Outro artifício já encontrado na estrutura que podemos aproveitar foi a plataforma já integrada a correia, fazendo que o movimento da correia, fizesse a plataforma se locomova também.

Esta estrutura que pré define a distância que a plataforma poderá percorrer. Para a aplicação final, poderá ser desenvolvida uma estrutura abrangendo uma distância superior ao que o protótipo poderá percorrer.

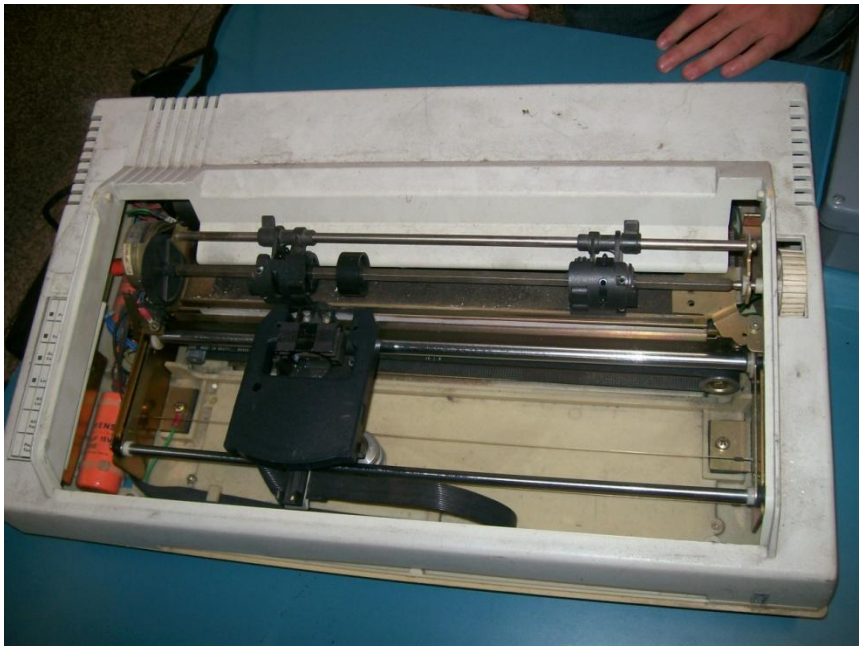


Figura 7 : Impressora

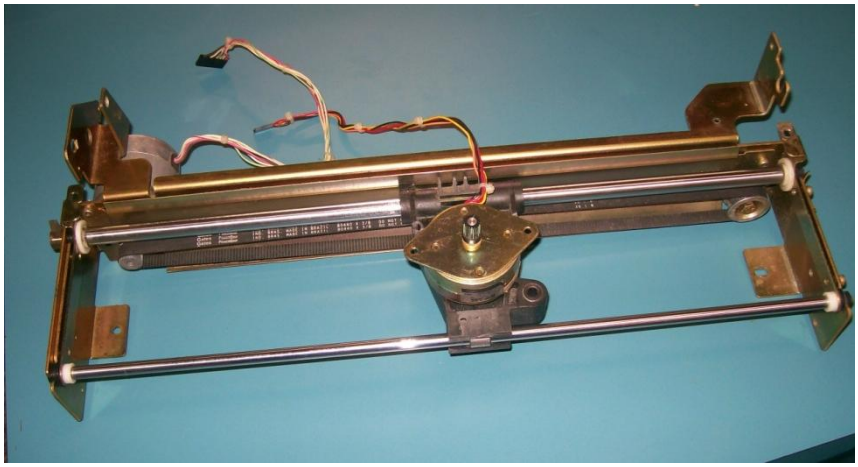


Figura 8 : Estrutura reaproveitada da impressora

Especificações ilustradas idem anexos nomeados de estrutura da impressora.

7.2 Motores de Passo

Cada motor de passo é responsável por um movimento, como já definido nos objetivos, um dos motores é responsável pelo o movimento horizontal e outro é responsável pela movimentação angular.

Estes possuem tipos de aplicações diferentes, ou seja, possuem características diferentes sendo um maior que o outro.

O motor responsável pela a movimentação na horizontal é mais robusto, pois além de suportar o peso do objeto a ser locomovido, ele também deve suportar toda a estrutura responsável pela a movimentação angular. Fica alojado atrás da estrutura da impressora, sendo seu eixo, um dos eixos no qual a correia rolará. E se tratando das características eletrônicas, o motor mais robusto está ligado a uma etapa de potência mais complexa, pois este motor necessita de uma corrente alta para que haja um funcionamento eficiente. Esta etapa de potência que fornece energia ao motor, é alimentada por uma fonte de 6 Volts que forneça no mínimo 1,2 Ampér de corrente, podendo mesmo com isso causar a movimentação no eixo horizontal.

O motor responsável pela movimentação angular é um motor mais leve, porém não menos eficiente. Para este caso o motor não precisaria suportar o peso de outro motor ou algo do tipo, pois isso se pode usar um motor menor para trabalhar o caso. Ele fica alojada na parte superior da plataforma fixada a correia, e fixada ao seu eixo, foi implantada plataforma que será o suporte onde os objetos serão colocados, e logo em seguida locomovidos. É alimentado através de uma etapa de potência simples, sendo este alimentado por uma fonte de 12 Volts fornecendo ate 0,5 Ampér.

7.3 Módulo M0

O módulo M0 foi desenvolvido a partir do micro controlador PIC16F629, que tem função de reconhecer dados enviados pela comunicação serial da linha TTL_SERIAL_BUS, processar e enviar para a etapa de potência.

Para a confecção deste circuito foram usados capacitores eletrolíticos de 100µF e 10µF e o circuito integrado PIC16F629, como mostra abaixo o diagrama em blocos:

7.4 Etapas de Potência

Este circuito alimenta os motores de passo, foram feitos duas etapas, uma de VALOR_AMPERES amperes para alimentar o motor da PLATAFORMA_OU_BASE e uma de QUANTOS_AMPERE ampéres para o outro motor de passo alimentar o outro motor. A etapa de potência é ativa ou desativada de acordo com o estimulo que seu transistor receber do módulo M0.

7.5 Conversor RS232 – TTL

O conversor RS232 – TLL é construído com o circuito integrado MAX232 que converte o sinal serial recebido do computador para o sinal serial TTL, e envia este sinal para o micro controlador PIC16F629 que processará as informações.

Para construir o conversor foi necessário um cicruito integrado MAX232, um transistor BC548, dois resistores de 1K , quatro capacitores eletrolíticos de 1µF x 16V e um de 100µF x 16V, um conector DB9 e um cabo de alimentação para alimentar o circuito.

7.6 Software Controlador

Para a construção do software primeiro foi decidido qual seria biblioteca gráfica a se usar, para o nosso caso escolhemos o Windows Form, pois já tínhamos um pequeno conhecimento sobre o compilador Visual Studio 2008, no qual é utilizado para criar interfaces do Windows Form.

Em seguida foi feita uma análise das necessidades de controle da plataforma. Como já citado, seus movimentos são horizontal e angular, portanto o usuário terá a necessidade de passar qual a distância ele deve percorrer em qual direção ele deve seguir, no eixo horizontal, e qual o ângulo que a plataforma deve atingir. Assim foram pensadas duas maneiras para que o usuário possa informar estes valores; Automática e Manual. O usuário deve escolher de qual maneira ele quer manipular a paltaforma através do conjunto MANIPULATION.

No controle manual o usuário deve passar exatamente os três valores, sendo estes: Distance (para distância), Way (para direção), e Angle (para ângulo). Todos os

valores já são pré definidos, ou seja, o usuário deve limitar a sua escolha aos valores que aparecerão na lista, ficando impossibilitado de escolher qualquer outro valor a não ser os anteriormente já pré selecionados pelo programador. Logo após a escolha aperta-se o botão OK, fazendo com inicia-se o movimento da plataforma.

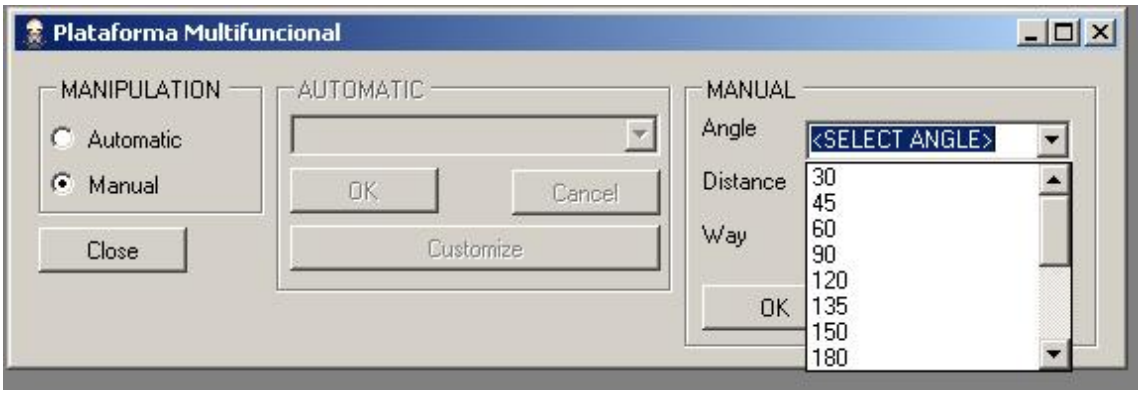


Figura 9 : Layout do Software (Janela Principal)

Figura A. Demonstração de como o usuário deve escolher os valores.

No controle automático anteriormente o usuário deve criar um padrão de movimento e para esse padrão ele deve dar um nome. Neste caso o usuário poderá criar vários padrões e ao chamar o modo automático, o usuário deve selecionar qual dos movimentos padrões com os seus respectivos nomes ele quer que a plataforma percorra (conforme Figura 9).

Para a criação deste movimento padrão o usuário deve primeiramente ativar a opção automático, em seguida clicar no botão CUSTOMIZE. Este botão o disponibilizará ao usuário uma janela como se fosse a opção manual, devendo proceder da mesma maneira, porém ao finalizar esta etapa deve estipular um nome ao conjunto de dados para que o programa salve o padrão, e possa disponibilizar ao usuário no layout principal. Ao clicar no botão OK do layout customize, o programa terá salvo o conjunto de dados, e disponibilizará ao usuário para que ele possa escolhe-lo na layout principal. Caso o usuário clique em cancelar no layout customize, ele apenas retornará ao layout principal.

Ao desenvolver o software, como observado, houve diversas interações de componentes que utilizamos no editor gráfico com o nosso código, causando com isso alguns erros já esperados pela diferença do código.

7.6.1 Código Fonte

Abaixo segue o código fonte do programa, referente a cada arquivo (.h) e (.cpp).

7.6.1.1 Dados.h

```
#ifndef dados_h
#define dados_h

#include <iostream>
#include <string>
#include <fstream>
#include <vector>

using namespace std;

class Dados
{
private:
	int angle;
	int distance;
	int way;
	char name[200];
	ifstream *bancoDados;
	int contadorElementos;
public:
	Dados();
	virtual ~Dados(void);
	Dados(int a, int d, int w);
	Dados(char *nome,int angulo, int distancia, int direcao);
public:
	void setAngle(int a);
	void setDistance(int d);
	void setWay(int w);
	void setName(char *n);
	char *getName();
	int getContadorElementos();
	int getAngle();
	int getDistance();
	int getWay();
	ifstream *getFile();
	//separa as strings do do arquivo texto
	string separaString(string *linha);
	//armazena os valores em um vetor contidos no arquivo
	vector<Dados*> loadFile();


};
#endif
```

7.6.1.2 Dados.cpp

```
#include "stdafx.h"
#include "Dados.h"
#include <iostream>
#include <string>
#include <fstream>
#include <vector>

using namespace std;

Dados::Dados(){
}
Dados::~Dados(){}

Dados::Dados(int a, int d, int w){
```

```
        this->angle=a;
        this->distance=d;
        this->way=w;
        this->contadorElementos=0;
}

Dados::Dados(char *nome,int angulo, int distancia, int direcao){
            strcpy(this->name,nome);
            this->angle=angulo;
            this->distance=distancia;
            this->way=direcao;
            this->contadorElementos=0;
}

void Dados::setAngle(int a){
            this->angle =a;
        }
void Dados::setDistance(int d){
            this->distance=d;
        }
void Dados::setWay(int w){
            this->way=w;
        }
void Dados::setName(char *n){
            strcpy(this->name,n);
        }
char *Dados::getName(){
            return this->name;
        }
int Dados::getContadorElementos(){
        return this->contadorElementos;
}

int Dados::getAngle(){
            switch (angle)
            {
            case 0:
                  //Quantidade de passos para fazer angulo de 30
                  return 2;
                  break;

            case 1:
                  //Quantidade de passos para fazer angulo de 45
                  return 3;
                  break;

            case 2:
                  //Quantidade de passos para fazer angulo de 60
                  return 4;
                  break;

            case 3:
                  //Quantidade de passos para fazer angulo de 90
                  return 12;
                  break;

            case 4:
                  //Quantidade de passos para fazer angulo de 120
                  return 8;
                  break;
```

```
        case 5:
              //Quantidade de passos para fazer angulo de 135
              return 9;
              break;

        case 6:
              //Quantidade de passos para fazer angulo de 150
              return 10;
              break;

        case 7:
              //Quantidade de passos para fazer angulo de 180
              return 24;
              break;

        case 8:
              //Quantidade de passos para fazer angulo de 210
              return 14;
              break;

        case 9:
              //Quantidade de passos para fazer angulo de 225
              return 15;
              break;

        case 10:
              //Quantidade de passos para fazer angulo de 240
              return 16;
              break;

        case 11:
              //Quantidade de passos para fazer angulo de 270
              return 18;
              break;

        case 12:
              //Quantidade de passos para fazer angulo de 300
              return 20;
              break;

        case 13:
              //Quantidade de passos para fazer angulo de 315
              return 21;
              break;

        case 14:
              //Quantidade de passos para fazer angulo de 330
              return 22;
              break;

        case 15:
              //Quantidade de passos para fazer angulo de 360
              return 48;
              break;

        default:
              return 0;
              break;
        }

}
```

```
int Dados::getDistance()
{
      switch (distance)
      {
      case 0:
            return 1480;
      case 1:
            return 1184;
      case 2:
            return 710;
      case 3:
            return 284;
      case 4:
            return 57;
      }
}
int Dados::getWay(){
            return way;
      }

      ifstream *Dados::getFile(){
            return this->bancoDados;
      }

string Dados :: separaString(string *linha){
      unsigned int comeco;
      unsigned int fim;

      string token;

      comeco = (unsigned int)  linha->find_first_not_of("\" , ");
      fim = (unsigned int)  linha->find_first_of("\" ,", comeco);

    token = linha->substr(comeco, fim - comeco);
            linha->erase(comeco, fim - comeco);

      return token;
}

vector<Dados*> Dados::loadFile(){

      ifstream *arquivo = new ifstream("BD.txt");
      //pegando valores dos pontos
                  //
                  Dados *d = new (Dados);

                  //variavel que recebe a linha d leitura do arquivo
                  string line;

                  //vetor  que  armazena  todos  os  dados  contidos  no
trajeto
                  vector<Dados*> v;


                  if (arquivo->is_open())
                  {
                        //leitura  de  ponto  a  ponto  do  arquivo  até  o
final do arquivo

                        //variaveis  para  armazenar  os  dados  após  a
separação
```

```
                        string nome;
                        string angulo;
                        string distancia;
                        string sentido;

                        //função que separa linha a linha do arquivo
                        getline (*arquivo,line);

                        //atribuição de valores para as variaveis de
armazenamento já separadas pela função separaString
                        nome = d->separaString(&line);
                        angulo = d->separaString(&line);
                        distancia =  d->separaString(&line);
                        sentido = d->separaString(&line);


                        //atribuição de valores das variaveis de
armazenamento as variaveis privadas da classe,
                        //sendo essas convertidas para os tipos de
variaveis existente nas classes, deixando de ser strings.
                        d->setName((char *)nome.c_str());
                        d->setAngle(atoi(angulo.c_str()));
                        d->setDistance(atoi(distancia.c_str()));
                        d->setWay(atoi(sentido.c_str()));

                        //atribuindo valores de cada linha a um
elemento do vetor
                        v.push_back(d);

                        //variavel contadora de elementos existente no
vetor
                        //incrementa a posicao do vetor
                        contadorElementos++;

                    }
                    //condição caso não consiga abrir o arquivo
                    else cout << "Unable to open file";
                    return v;
}
```

7.6.1.3 Serial.h

```
// Serial.h

#include <windows.h>

#ifndef __SERIAL_H__
#define __SERIAL_H__

#define FC_DTRDSR       0x01
#define FC_RTSCTS       0x02
#define FC_XONXOFF      0x04
#define ASCII_BEL       0x07
#define ASCII_BS        0x08
#define ASCII_LF        0x0A
#define ASCII_CR        0x0D
#define ASCII_XON       0x11
#define ASCII_XOFF      0x13

class CSerial
{
```

```
public:
      CSerial();
      ~CSerial();

      BOOL Open( int nPort, int nBaud );
      BOOL Close( void );

      int ReadData( void *, int );
      int SendData( const char *, int );
      int ReadDataWaiting( void );

      BOOL IsOpened( void ){ return( m_bOpened ); }

protected:
      BOOL WriteCommByte( unsigned char );

      HANDLE m_hIDComDev;
      OVERLAPPED m_OverlappedRead, m_OverlappedWrite;
      BOOL m_bOpened;

};

#endif
```

7.6.1.4 Serial.cpp

```
// Serial.cpp
#include "stdafx.h"
#include <iostream>
#include "Serial.h"



CSerial::CSerial()
{

      memset( &m_OverlappedRead, 0, sizeof( OVERLAPPED ) );
      memset( &m_OverlappedWrite, 0, sizeof( OVERLAPPED ) );
      m_hIDComDev = NULL;
      m_bOpened = FALSE;

}

CSerial::~CSerial()
{

      Close();

}

BOOL CSerial::Open( int nPort, int nBaud )
{

      if( m_bOpened ) return( TRUE );

      char szPort[15];
      char szComParams[50];
      DCB dcb;

      sprintf( szPort, "COM%d", nPort );
```

```
	m_hIDComDev = CreateFileA( szPort, GENERIC_READ | GENERIC_WRITE,
0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED,
NULL );
	if( m_hIDComDev == NULL ) return( FALSE );

	memset( &m_OverlappedRead, 0, sizeof( OVERLAPPED ) );
	memset( &m_OverlappedWrite, 0, sizeof( OVERLAPPED ) );

	COMMTIMEOUTS CommTimeOuts;
	CommTimeOuts.ReadIntervalTimeout = 0xFFFFFFFF;
	CommTimeOuts.ReadTotalTimeoutMultiplier = 0;
	CommTimeOuts.ReadTotalTimeoutConstant = 0;
	CommTimeOuts.WriteTotalTimeoutMultiplier = 0;
	CommTimeOuts.WriteTotalTimeoutConstant = 5000;
	SetCommTimeouts( m_hIDComDev, &CommTimeOuts );

	sprintf( szComParams, "COM%d:%d,n,8,1", nPort, nBaud );

	m_OverlappedRead.hEvent = CreateEvent( NULL, TRUE, FALSE, NULL
);
	m_OverlappedWrite.hEvent = CreateEvent( NULL, TRUE, FALSE, NULL
);

	dcb.DCBlength = sizeof( DCB );
	GetCommState( m_hIDComDev, &dcb );
	dcb.BaudRate = nBaud;
	dcb.ByteSize = 8;
	unsigned char ucSet;
	ucSet = (unsigned char) ( ( FC_RTSCTS & FC_DTRDSR ) != 0 );
	ucSet = (unsigned char) ( ( FC_RTSCTS & FC_RTSCTS ) != 0 );
	ucSet = (unsigned char) ( ( FC_RTSCTS & FC_XONXOFF ) != 0 );
	if( !SetCommState( m_hIDComDev, &dcb ) ||
		!SetupComm( m_hIDComDev, 10000, 10000 ) ||
		m_OverlappedRead.hEvent == NULL ||
		m_OverlappedWrite.hEvent == NULL ){
		DWORD dwError = GetLastError();
		if( m_OverlappedRead.hEvent != NULL ) CloseHandle(
m_OverlappedRead.hEvent );
		if( m_OverlappedWrite.hEvent != NULL ) CloseHandle(
m_OverlappedWrite.hEvent );
		CloseHandle( m_hIDComDev );
		return( FALSE );
		}

	m_bOpened = TRUE;

	return( m_bOpened );

}

BOOL CSerial::Close( void )
{

	if( !m_bOpened || m_hIDComDev == NULL ) return( TRUE );

	if( m_OverlappedRead.hEvent != NULL ) CloseHandle(
m_OverlappedRead.hEvent );
	if( m_OverlappedWrite.hEvent != NULL ) CloseHandle(
m_OverlappedWrite.hEvent );
	CloseHandle( m_hIDComDev );
	m_bOpened = FALSE;
```

```
	m_hIDComDev = NULL;

	return( TRUE );

}

BOOL CSerial::WriteCommByte( unsigned char ucByte )
{
	BOOL bWriteStat;
	DWORD dwBytesWritten;

	bWriteStat  =  WriteFile(  m_hIDComDev,  (LPSTR)  &ucByte,  1,
&dwBytesWritten, &m_OverlappedWrite );
	if( !bWriteStat && ( GetLastError() == ERROR_IO_PENDING ) ){
		if( WaitForSingleObject( m_OverlappedWrite.hEvent, 1000 ) )
dwBytesWritten = 0;
		else{
			GetOverlappedResult( m_hIDComDev, &m_OverlappedWrite,
&dwBytesWritten, FALSE );
			m_OverlappedWrite.Offset += dwBytesWritten;
			}
		}

	return( TRUE );

}

int CSerial::SendData( const char *buffer, int size )
{

	if( !m_bOpened || m_hIDComDev == NULL ) return( 0 );

	DWORD dwBytesWritten = 0;
	int i;
	for( i=0; i<size; i++ ){
		WriteCommByte( buffer[i] );
		dwBytesWritten++;
		}

	return( (int) dwBytesWritten );

}

int CSerial::ReadDataWaiting( void )
{

	if( !m_bOpened || m_hIDComDev == NULL ) return( 0 );

	DWORD dwErrorFlags;
	COMSTAT ComStat;

	ClearCommError( m_hIDComDev, &dwErrorFlags, &ComStat );

	return( (int) ComStat.cbInQue );

}

int CSerial::ReadData( void *buffer, int limit )
{

	if( !m_bOpened || m_hIDComDev == NULL ) return( 0 );
```

```
	BOOL bReadStatus;
	DWORD dwBytesRead, dwErrorFlags;
	COMSTAT ComStat;

	ClearCommError( m_hIDComDev, &dwErrorFlags, &ComStat );
	if( !ComStat.cbInQue ) return( 0 );

	dwBytesRead = (DWORD) ComStat.cbInQue;
	if( limit < (int) dwBytesRead ) dwBytesRead = (DWORD) limit;

	bReadStatus = ReadFile( m_hIDComDev, buffer, dwBytesRead,
&dwBytesRead, &m_OverlappedRead );
	if( !bReadStatus ){
		if( GetLastError() == ERROR_IO_PENDING ){
			WaitForSingleObject( m_OverlappedRead.hEvent, 2000 );
			return( (int) dwBytesRead );
			}
		return( 0 );
		}

	return( (int) dwBytesRead );

}
```

7.6.1.5 Customize.h

```
#pragma once

#include <iostream>
#include <vector>
#include <fstream>

#include "Dados.h"

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Runtime::InteropServices;

using namespace std;


namespace INTERFACE_2 {

	/// <summary>
	/// Summary for Customize
	///
	/// WARNING: If you change the name of this class, you will need
to change the
	///		'Resource File Name' property for the managed
resource compiler tool
	///		associated with all .resx files this class depends
on. Otherwise,
	///		the designers will not be able to interact properly
with localized
	///		resources associated with this form.
```

```
	/// </summary>
	public ref class Customize : public System::Windows::Forms::Form
	{
	public:
		Customize(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~Customize()
		{
			if (components)
			{
				delete components;
			}
		}

	protected:
	private: System::Windows::Forms::Label^  label3;
	private: System::Windows::Forms::Button^  Cancel;

	private: System::Windows::Forms::Label^  label2;
	private: System::Windows::Forms::Label^  label4;
	private: System::Windows::Forms::Label^  label1;


	private: System::Windows::Forms::ComboBox^  cb_way;
	private: System::Windows::Forms::ComboBox^  cb_distance;
	private: System::Windows::Forms::ComboBox^  cb_angle;
	private: System::Windows::Forms::TextBox^  t_name;

	private: System::Windows::Forms::Button^  OK;

	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		void InitializeComponent(void)
		{
			System::ComponentModel::ComponentResourceManager^
resources = (gcnew
System::ComponentModel::ComponentResourceManager(Customize::typeid));
			this->t_name = (gcnew
System::Windows::Forms::TextBox());
			this->label3 = (gcnew
System::Windows::Forms::Label());
```

```
                    this->Cancel                    =                    (gcnew
System::Windows::Forms::Button());
                    this->label2                    =                    (gcnew
System::Windows::Forms::Label());
                    this->label4                    =                    (gcnew
System::Windows::Forms::Label());
                    this->label1                    =                    (gcnew
System::Windows::Forms::Label());
                    this->OK = (gcnew System::Windows::Forms::Button());
                    this->cb_way                    =                    (gcnew
System::Windows::Forms::ComboBox());
                    this->cb_distance                 =                  (gcnew
System::Windows::Forms::ComboBox());
                    this->cb_angle                   =                   (gcnew
System::Windows::Forms::ComboBox());
                    this->SuspendLayout();
                    //
                    // t_name
                    //
                    this->t_name->BackColor                                  =
System::Drawing::SystemColors::Control;
                    this->t_name->ForeColor                                  =
System::Drawing::SystemColors::Menu;
                    this->t_name->Location  =  System::Drawing::Point(131,
22);
                    this->t_name->Name = L"t_name";
                    this->t_name->Size = System::Drawing::Size(100, 20);
                    this->t_name->TabIndex = 2;
                    this->t_name->TextChanged           +=           gcnew
System::EventHandler(this, &Customize::t_name_TextChanged);
                    //
                    // label3
                    //
                    this->label3->AutoSize = true;
                    this->label3->BackColor                                  =
System::Drawing::Color::Transparent;
                    this->label3->Font                 =                 (gcnew
System::Drawing::Font(L"Microsoft       Sans       Serif",       8.25F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->label3->ForeColor                                  =
System::Drawing::SystemColors::ControlText;
                    this->label3->Location  =  System::Drawing::Point(37,
124);
                    this->label3->Name = L"label3";
                    this->label3->Size = System::Drawing::Size(29, 13);
                    this->label3->TabIndex = 1;
                    this->label3->Text = L"Way";
                    this->label3->Click                 +=                gcnew
System::EventHandler(this, &Customize::label3_Click);
                    //
                    // Cancel
                    //
                    this->Cancel->Cursor                                     =
System::Windows::Forms::Cursors::Hand;
                    this->Cancel->Location  =  System::Drawing::Point(317,
95);
                    this->Cancel->Name = L"Cancel";
                    this->Cancel->Size = System::Drawing::Size(75, 23);
                    this->Cancel->TabIndex = 0;
```

```
                    this->Cancel->Text = L"Cancel";
                    this->Cancel->UseVisualStyleBackColor = true;
                    this->Cancel->Click              +=              gcnew
System::EventHandler(this, &Customize::Cancel_Click);
                    //
                    // label2
                    //
                    this->label2->AutoSize = true;
                    this->label2->BackColor                              =
System::Drawing::Color::Transparent;
                    this->label2->Font              =              (gcnew
System::Drawing::Font(L"Microsoft       Sans       Serif",       8.25F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->label2->ForeColor                              =
System::Drawing::SystemColors::ControlText;
                    this->label2->Location  =  System::Drawing::Point(37,
89);
                    this->label2->Name = L"label2";
                    this->label2->Size = System::Drawing::Size(49, 13);
                    this->label2->TabIndex = 1;
                    this->label2->Text = L"Distance";
                    this->label2->Click              +=              gcnew
System::EventHandler(this, &Customize::label2_Click);
                    //
                    // label4
                    //
                    this->label4->AutoSize = true;
                    this->label4->BackColor                              =
System::Drawing::Color::Transparent;
                    this->label4->Font              =              (gcnew
System::Drawing::Font(L"Microsoft       Sans       Serif",       8.25F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->label4->ForeColor                              =
System::Drawing::SystemColors::ControlText;
                    this->label4->Location  =  System::Drawing::Point(27,
25);
                    this->label4->Name = L"label4";
                    this->label4->Size = System::Drawing::Size(84, 13);
                    this->label4->TabIndex = 1;
                    this->label4->Text = L"Program\'s Name";
                    this->label4->Click              +=              gcnew
System::EventHandler(this, &Customize::label4_Click);
                    //
                    // label1
                    //
                    this->label1->AutoSize = true;
                    this->label1->BackColor                              =
System::Drawing::Color::Transparent;
                    this->label1->Font              =              (gcnew
System::Drawing::Font(L"Microsoft       Sans       Serif",       8.25F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->label1->ForeColor                              =
System::Drawing::SystemColors::ControlText;
                    this->label1->Location  =  System::Drawing::Point(37,
54);
```

```
                    this->label1->Name = L"label1";
                    this->label1->Size = System::Drawing::Size(34, 13);
                    this->label1->TabIndex = 1;
                    this->label1->Text = L"Angle";
                    this->label1->Click              +=              gcnew
System::EventHandler(this, &Customize::label1_Click);
                    //
                    // OK
                    //
                    this->OK->BackColor                                    =
System::Drawing::Color::Transparent;
                    this->OK->BackgroundImageLayout                        =
System::Windows::Forms::ImageLayout::None;
                    this->OK->Cursor                                       =
System::Windows::Forms::Cursors::Hand;
                    this->OK->Location = System::Drawing::Point(317, 66);
                    this->OK->Name = L"OK";
                    this->OK->Size = System::Drawing::Size(75, 23);
                    this->OK->TabIndex = 0;
                    this->OK->Text = L"OK";
                    this->OK->UseVisualStyleBackColor = false;
                    this->OK->Click  +=  gcnew  System::EventHandler(this,
&Customize::OK_Click);
                    //
                    // cb_way
                    //
                    this->cb_way->DisplayMember = L"1";
                    this->cb_way->FormattingEnabled = true;
                    this->cb_way->Items->AddRange(gcnew       cli::array<
System::Object^  >(2) {L"Esquerda", L"Direita"});
                    this->cb_way->Location  =  System::Drawing::Point(115,
121);
                    this->cb_way->Name = L"cb_way";
                    this->cb_way->Size = System::Drawing::Size(135, 21);
                    this->cb_way->TabIndex = 0;
                    this->cb_way->Text = L"<SELECET WAY>";
                    //
                    // cb_distance
                    //
                    this->cb_distance->FormattingEnabled = true;
                    this->cb_distance->Items->AddRange(gcnew  cli::array<
System::Object^   >(5)  {L"45  cm",  L"36  cm",  L"27  cm",  L"18  cm",  L"9
cm"});
                    this->cb_distance->Location                            =
System::Drawing::Point(115, 86);
                    this->cb_distance->Name = L"cb_distance";
                    this->cb_distance->Size  =  System::Drawing::Size(135,
21);
                    this->cb_distance->TabIndex = 0;
                    this->cb_distance->Text = L"<SELECT DISTANCE>";
                    //
                    // cb_angle
                    //
                    this->cb_angle->FormattingEnabled = true;
                    this->cb_angle->Items->AddRange(gcnew     cli::array<
System::Object^   >(16)  {L"30",  L"45",  L"60",  L"90",  L"120",  L"135",
L"150",
                         L"180", L"210", L"225", L"240", L"370", L"300",
L"315", L"330", L"360"});
                    this->cb_angle->Location                               =
System::Drawing::Point(115, 51);
```

```
                    this->cb_angle->Name = L"cb_angle";
                    this->cb_angle->Size   =   System::Drawing::Size(135,
21);
                    this->cb_angle->TabIndex = 0;
                    this->cb_angle->Text = L"<SELECT ANGLE>";
                    //
                    // Customize
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6,
13);
                    this->AutoScaleMode                                =
System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(404, 177);
                    this->Controls->Add(this->Cancel);
                    this->Controls->Add(this->label3);
                    this->Controls->Add(this->t_name);
                    this->Controls->Add(this->cb_way);
                    this->Controls->Add(this->label2);
                    this->Controls->Add(this->cb_distance);
                    this->Controls->Add(this->OK);
                    this->Controls->Add(this->label1);
                    this->Controls->Add(this->label4);
                    this->Controls->Add(this->cb_angle);
                    this->Icon  =  (cli::safe_cast<System::Drawing::Icon^
>(resources->GetObject(L"$this.Icon")));
                    this->Name = L"Customize";
                    this->Text = L"Customize";
                    this->ResumeLayout(false);
                    this->PerformLayout();

            }
#pragma endregion
      private:    System::Void    OK_Click(System::Object^       sender,
System::EventArgs^  e) {
                        //pegando as variaveis da combo
                       int angleValue = cb_angle->SelectedIndex;
                        int      distanceValue      =      cb_distance-
>SelectedIndex;
                        int wayValue = cb_way->SelectedIndex, aux=0;
                        char           *           nameValue           =
(char*)(void*)Marshal::StringToHGlobalAnsi(t_name->Text), nome[100];
                        //Grava arquivo
                        FILE *Arquivo = fopen("Dados.txt", "r");

                        while(!feof(Arquivo))
                        {
                              fscanf(Arquivo,     "%s\n%i\n%i\n%i\n\n",
&nome, &aux, &aux, &aux);
                              if(strcmp(nome, nameValue)==0)
                              {
                                    MessageBox::Show("Este    nome    ja
existe", "ATENÇÃO");
                                    return;
                              }
                        }
                        //Form1::Form1();
                        Arquivo = fopen("Dados.txt", "a");
                        if(Arquivo==NULL)
                        {
                              MessageBox::Show("Erro     ao     gravar
posição", "ATENÇÃO");
```

```
                                return;
                            }


                            Dados d(angleValue, distanceValue, wayValue);
                            fprintf(Arquivo,          "%s\n%i\n%i\n%i\n\n",
nameValue, d.getAngle(), d.getDistance(), d.getWay());

                            fclose(Arquivo);
                             Close();
                             Application::Restart();

                             //Facha a janela

                        }
private:    System::Void    Cancel_Click(System::Object^        sender,
System::EventArgs^  e) {
                        Close();
                        }
private:   System::Void   t_name_TextChanged(System::Object^     sender,
System::EventArgs^  e) {
                    }
private:    System::Void    create_Click(System::Object^        sender,
System::EventArgs^  e) {
                        this->cb_angle->Enabled=true;
                        this->cb_distance->Enabled=true;
                        this->cb_way->Enabled=true;
                        this->OK->Enabled=true;
                        this->t_name->Enabled=true;

                    }
private:    System::Void    label2_Click(System::Object^        sender,
System::EventArgs^  e) {
                    }
private:    System::Void    label4_Click(System::Object^        sender,
System::EventArgs^  e) {
                    }
private:    System::Void    label3_Click(System::Object^        sender,
System::EventArgs^  e) {
                    }
private:    System::Void    label1_Click(System::Object^        sender,
System::EventArgs^  e) {
                    }
};
}
```

7.6.1.6 Form1.h

```
#pragma once

#include "Dados.h"
#include "serial.h"
#include "Customize.h"
#include <iostream>
#include <stdio.h>

#define PORTA 7

namespace INTERFACE_2 {

      using namespace System;
```

```
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	//using namespace Dados;

	/// <summary>
	/// Summary for Form1
	///
	/// WARNING: If you change the name of this class, you will need
to change the
	///              'Resource File Name' property for the managed
resource compiler tool
	///           associated with all .resx files this class depends
on.  Otherwise,
	///          the designers will not be able to interact properly
with localized
	///          resources associated with this form.
	/// </summary>
	public ref class Form1 : public System::Windows::Forms::Form
	{
	public:
		Form1(void)
		{
			InitializeComponent();

			char nome[10];
			int aux;
			String^ result;
			FILE *carrega = fopen("Dados.txt", "r");
			while(carrega!=NULL && !feof(carrega))
			{
				fscanf(carrega,   "%s\n%i\n%i\n%i\n\n",   &nome,
&aux, &aux, &aux);
				result = gcnew String(nome);
				cb_fileSave->Items->Add(result);
			}
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~Form1()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  OK_automatic;
	private: System::Windows::Forms::GroupBox^  groupBox1;
	private: System::Windows::Forms::RadioButton^  rb_manual;
	private: System::Windows::Forms::RadioButton^  rb_automatic;
	private: System::Windows::Forms::GroupBox^  gb_AUTOMATIC;


	private: System::Windows::Forms::Button^  Cancel_automatic;
	private: System::Windows::Forms::GroupBox^  gb_MANUAL;
	private: System::Windows::Forms::Button^  Customize_botao;
```

```
        private: System::Windows::Forms::Label^  label3;
        private: System::Windows::Forms::Label^  label2;
        private: System::Windows::Forms::Label^  label1;
        private: System::Windows::Forms::ComboBox^  cb_way;
        private: System::Windows::Forms::ComboBox^  cb_distance;
        private: System::Windows::Forms::ComboBox^  cb_angle;
        private: System::Windows::Forms::Button^  Cancel_manual;
        private: System::Windows::Forms::Button^  OK_manual;
        private: System::Windows::Forms::Button^  Close;
        private: System::Windows::Forms::ComboBox^  cb_fileSave;



        protected:

        private:
              /// <summary>
              /// Required designer variable.
              /// </summary>
              System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
              /// <summary>
              /// Required method for Designer support - do not modify
              /// the contents of this method with the code editor.
              /// </summary>
              void InitializeComponent(void)
              {
                    System::ComponentModel::ComponentResourceManager^
resources                          =                          (gcnew
System::ComponentModel::ComponentResourceManager(Form1::typeid));
                    this->OK_automatic              =              (gcnew
System::Windows::Forms::Button());
                    this->groupBox1               =               (gcnew
System::Windows::Forms::GroupBox());
                    this->rb_manual               =               (gcnew
System::Windows::Forms::RadioButton());
                    this->rb_automatic              =              (gcnew
System::Windows::Forms::RadioButton());
                    this->gb_AUTOMATIC              =              (gcnew
System::Windows::Forms::GroupBox());
                    this->cb_fileSave               =              (gcnew
System::Windows::Forms::ComboBox());
                    this->Cancel_automatic            =            (gcnew
System::Windows::Forms::Button());
                    this->Customize_botao             =            (gcnew
System::Windows::Forms::Button());
                    this->gb_MANUAL               =               (gcnew
System::Windows::Forms::GroupBox());
                    this->label3                =                 (gcnew
System::Windows::Forms::Label());
                    this->Cancel_manual              =             (gcnew
System::Windows::Forms::Button());
                    this->label2                =                 (gcnew
System::Windows::Forms::Label());
                    this->label1                =                 (gcnew
System::Windows::Forms::Label());
                    this->OK_manual               =               (gcnew
System::Windows::Forms::Button());
```

```
                this->cb_way                    =                   (gcnew
System::Windows::Forms::ComboBox());
                this->cb_distance                 =                 (gcnew
System::Windows::Forms::ComboBox());
                this->cb_angle                   =                  (gcnew
System::Windows::Forms::ComboBox());
                this->Close                     =                   (gcnew
System::Windows::Forms::Button());
                this->groupBox1->SuspendLayout();
                this->gb_AUTOMATIC->SuspendLayout();
                this->gb_MANUAL->SuspendLayout();
                this->SuspendLayout();
                //
                // OK_automatic
                //
                this->OK_automatic->Location                        =
System::Drawing::Point(7, 46);
                this->OK_automatic->Name = L"OK_automatic";
                this->OK_automatic->Size  =  System::Drawing::Size(75,
23);
                this->OK_automatic->TabIndex = 0;
                this->OK_automatic->Text = L"OK";
                this->OK_automatic->UseVisualStyleBackColor = true;
                this->OK_automatic->Click           +=          gcnew
System::EventHandler(this, &Form1::OK_automatic_Click);
                //
                // groupBox1
                //
                this->groupBox1->Controls->Add(this->rb_manual);
                this->groupBox1->Controls->Add(this->rb_automatic);
                this->groupBox1->Location                           =
System::Drawing::Point(12, 12);
                this->groupBox1->Name = L"groupBox1";
                this->groupBox1->Size   =   System::Drawing::Size(114,
70);
                this->groupBox1->TabIndex = 1;
                this->groupBox1->TabStop = false;
                this->groupBox1->Text = L"MANIPULATION";
                //
                // rb_manual
                //
                this->rb_manual->AutoSize = true;
                this->rb_manual->Location = System::Drawing::Point(6,
46);
                this->rb_manual->Name = L"rb_manual";
                this->rb_manual->Size   =   System::Drawing::Size(60,
17);
                this->rb_manual->TabIndex = 0;
                this->rb_manual->TabStop = true;
                this->rb_manual->Text = L"Manual";
                this->rb_manual->UseVisualStyleBackColor = true;
                this->rb_manual->CheckedChanged        +=       gcnew
System::EventHandler(this, &Form1::rb_manual_CheckedChanged);
                //
                // rb_automatic
                //
                this->rb_automatic->AutoSize = true;
                this->rb_automatic->Location                        =
System::Drawing::Point(6, 23);
                this->rb_automatic->Name = L"rb_automatic";
```

```
                    this->rb_automatic->Size  =  System::Drawing::Size(72,
17);
                    this->rb_automatic->TabIndex = 0;
                    this->rb_automatic->TabStop = true;
                    this->rb_automatic->Text = L"Automatic";
                    this->rb_automatic->UseVisualStyleBackColor = true;
                    this->rb_automatic->CheckedChanged      +=      gcnew
System::EventHandler(this, &Form1::rb_automatic_CheckedChanged);
                    //
                    // gb_AUTOMATIC
                    //
                    this->gb_AUTOMATIC->Controls->Add(this->cb_fileSave);
                    this->gb_AUTOMATIC->Controls->Add(this-
>Cancel_automatic);
                    this->gb_AUTOMATIC->Controls->Add(this-
>Customize_botao);
                    this->gb_AUTOMATIC->Controls->Add(this-
>OK_automatic);
                    this->gb_AUTOMATIC->Enabled = false;
                    this->gb_AUTOMATIC->Location                        =
System::Drawing::Point(132, 12);
                    this->gb_AUTOMATIC->Name = L"gb_AUTOMATIC";
                    this->gb_AUTOMATIC->Size = System::Drawing::Size(200,
109);
                    this->gb_AUTOMATIC->TabIndex = 2;
                    this->gb_AUTOMATIC->TabStop = false;
                    this->gb_AUTOMATIC->Text = L"AUTOMATIC";
                    //
                    // cb_fileSave
                    //
                    this->cb_fileSave->FormattingEnabled = true;
                    this->cb_fileSave->Location                         =
System::Drawing::Point(7, 19);
                    this->cb_fileSave->Name = L"cb_fileSave";
                    this->cb_fileSave->Size  =  System::Drawing::Size(187,
21);
                    this->cb_fileSave->TabIndex = 1;
                    this->cb_fileSave->SelectedIndexChanged    +=    gcnew
System::EventHandler(this, &Form1::filesave_SelectedIndexChanged);
                    //
                    // Cancel_automatic
                    //
                    this->Cancel_automatic->Location                    =
System::Drawing::Point(119, 47);
                    this->Cancel_automatic->Name = L"Cancel_automatic";
                    this->Cancel_automatic->Size                        =
System::Drawing::Size(75, 23);
                    this->Cancel_automatic->TabIndex = 0;
                    this->Cancel_automatic->Text = L"Cancel";
                    this->Cancel_automatic->UseVisualStyleBackColor     =
true;
                    this->Cancel_automatic->Click         +=         gcnew
System::EventHandler(this, &Form1::Cancel_automatic_Click);
                    //
                    // Customize_botao
                    //
                    this->Customize_botao->Location                     =
System::Drawing::Point(7, 75);
                    this->Customize_botao->Name = L"Customize_botao";
                    this->Customize_botao->Size                         =
System::Drawing::Size(187, 23);
```

```
                    this->Customize_botao->TabIndex = 0;
                    this->Customize_botao->Text = L"Customize";
                    this->Customize_botao->UseVisualStyleBackColor      =
true;
                    this->Customize_botao->Click         +=         gcnew
System::EventHandler(this, &Form1::Customize_botao_Click);
                    //
                    // gb_MANUAL
                    //
                    this->gb_MANUAL->Controls->Add(this->label3);
                    this->gb_MANUAL->Controls->Add(this->Cancel_manual);
                    this->gb_MANUAL->Controls->Add(this->label2);
                    this->gb_MANUAL->Controls->Add(this->label1);
                    this->gb_MANUAL->Controls->Add(this->OK_manual);
                    this->gb_MANUAL->Controls->Add(this->cb_way);
                    this->gb_MANUAL->Controls->Add(this->cb_distance);
                    this->gb_MANUAL->Controls->Add(this->cb_angle);
                    this->gb_MANUAL->Enabled = false;
                    this->gb_MANUAL->Location                           =
System::Drawing::Point(338, 12);
                    this->gb_MANUAL->Name = L"gb_MANUAL";
                    this->gb_MANUAL->Size   =   System::Drawing::Size(209,
137);
                    this->gb_MANUAL->TabIndex = 2;
                    this->gb_MANUAL->TabStop = false;
                    this->gb_MANUAL->Text = L"MANUAL";
                    //
                    // label3
                    //
                    this->label3->AutoSize = true;
                    this->label3->Location   =   System::Drawing::Point(6,
73);
                    this->label3->Name = L"label3";
                    this->label3->Size = System::Drawing::Size(29, 13);
                    this->label3->TabIndex = 1;
                    this->label3->Text = L"Way";
                    this->label3->Click              +=              gcnew
System::EventHandler(this, &Form1::label3_Click);
                    //
                    // Cancel_manual
                    //
                    this->Cancel_manual->Location                       =
System::Drawing::Point(121, 106);
                    this->Cancel_manual->Name = L"Cancel_manual";
                    this->Cancel_manual->Size = System::Drawing::Size(75,
23);
                    this->Cancel_manual->TabIndex = 0;
                    this->Cancel_manual->Text = L"Cancel";
                    this->Cancel_manual->UseVisualStyleBackColor = true;
                    this->Cancel_manual->Click          +=          gcnew
System::EventHandler(this, &Form1::Cancel_manual_Click);
                    //
                    // label2
                    //
                    this->label2->AutoSize = true;
                    this->label2->Location   =   System::Drawing::Point(6,
46);
                    this->label2->Name = L"label2";
                    this->label2->Size = System::Drawing::Size(49, 13);
                    this->label2->TabIndex = 1;
                    this->label2->Text = L"Distance";
```

```
                    //
                    // label1
                    //
                    this->label1->AutoSize = true;
                    this->label1->Location   =   System::Drawing::Point(6,
19);
                    this->label1->Name = L"label1";
                    this->label1->Size = System::Drawing::Size(34, 13);
                    this->label1->TabIndex = 1;
                    this->label1->Text = L"Angle";
                    //
                    // OK_manual
                    //
                    this->OK_manual->Location = System::Drawing::Point(9,
105);
                    this->OK_manual->Name = L"OK_manual";
                    this->OK_manual->Size    =    System::Drawing::Size(75,
23);
                    this->OK_manual->TabIndex = 0;
                    this->OK_manual->Text = L"OK";
                    this->OK_manual->UseVisualStyleBackColor = true;
                    this->OK_manual->Click            +=            gcnew
System::EventHandler(this, &Form1::OK_manual_Click);
                    //
                    // cb_way
                    //
                    this->cb_way->DisplayMember = L"1";
                    this->cb_way->FormattingEnabled = true;
                    this->cb_way->Items->AddRange(gcnew       cli::array<
System::Object^  >(2) {L"Left", L"Right"});
                    this->cb_way->Location  =  System::Drawing::Point(61,
76);
                    this->cb_way->Name = L"cb_way";
                    this->cb_way->Size = System::Drawing::Size(135, 21);
                    this->cb_way->TabIndex = 0;
                    this->cb_way->Text = L"<SELECET WAY>";
                    //
                    // cb_distance
                    //
                    this->cb_distance->FormattingEnabled = true;
                    this->cb_distance->Items->AddRange(gcnew  cli::array<
System::Object^   >(5)  {L"45  cm",  L"36  cm",  L"27  cm",  L"18  cm",  L"9
cm"});
                    this->cb_distance->Location                          =
System::Drawing::Point(61, 48);
                    this->cb_distance->Name = L"cb_distance";
                    this->cb_distance->Size  =  System::Drawing::Size(135,
21);
                    this->cb_distance->TabIndex = 0;
                    this->cb_distance->Text = L"<SELECT DISTANCE>";
                    //
                    // cb_angle
                    //
                    this->cb_angle->FormattingEnabled = true;
                    this->cb_angle->Items->AddRange(gcnew     cli::array<
System::Object^   >(16)  {L"30",  L"45",  L"60",  L"90",  L"120",  L"135",
L"150",
                         L"180", L"210", L"225", L"240", L"370", L"300",
L"315", L"330", L"360"});
                    this->cb_angle->Location = System::Drawing::Point(61,
22);
```

```
                    this->cb_angle->Name = L"cb_angle";
                    this->cb_angle->Size   =   System::Drawing::Size(135,
21);
                    this->cb_angle->TabIndex = 0;
                    this->cb_angle->Text = L"<SELECT ANGLE>";
                    this->cb_angle->SelectedIndexChanged     +=     gcnew
System::EventHandler(this, &Form1::comboBox2_SelectedIndexChanged);
                    //
                    // Close
                    //
                    this->Close->Location   =   System::Drawing::Point(12,
88);
                    this->Close->Name = L"Close";
                    this->Close->Size = System::Drawing::Size(75, 23);
                    this->Close->TabIndex = 3;
                    this->Close->Text = L"Close";
                    this->Close->UseVisualStyleBackColor = true;
                    this->Close->Click              +=              gcnew
System::EventHandler(this, &Form1::Close_Click);
                    //
                    // Form1
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6,
13);
                    this->AutoScaleMode                                =
System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(558, 156);
                    this->Controls->Add(this->Close);
                    this->Controls->Add(this->gb_MANUAL);
                    this->Controls->Add(this->gb_AUTOMATIC);
                    this->Controls->Add(this->groupBox1);
                    this->Icon  =  (cli::safe_cast<System::Drawing::Icon^
>(resources->GetObject(L"$this.Icon")));
                    this->Name = L"Form1";
                    this->Text = L"Plataforma Multifuncional";
                    this->Load    +=    gcnew    System::EventHandler(this,
&Form1::Form1_Load);
                    this->groupBox1->ResumeLayout(false);
                    this->groupBox1->PerformLayout();
                    this->gb_AUTOMATIC->ResumeLayout(false);
                    this->gb_MANUAL->ResumeLayout(false);
                    this->gb_MANUAL->PerformLayout();
                    this->ResumeLayout(false);

            }
#pragma endregion
      private:   System::Void   label3_Click(System::Object^     sender,
System::EventArgs^  e) {
                    }
private:    System::Void    rb_automatic_CheckedChanged(System::Object^
sender, System::EventArgs^  e) {
                    this->gb_AUTOMATIC->Enabled = true;
                    this->gb_MANUAL->Enabled = false;
                    char *aux;
                    Dados d(0,0,0);
                    vector<Dados*> v=d.loadFile();
                    for(int
contador=0;contador<d.getContadorElementos();contador++){
                    }
              }
```

```
private:      System::Void      rb_manual_CheckedChanged(System::Object^
sender, System::EventArgs^  e) {
                    this->gb_MANUAL->Enabled = true;
                    this->gb_AUTOMATIC->Enabled = false;
              }
private:   System::Void   comboBox2_SelectedIndexChanged(System::Object^
sender, System::EventArgs^  e) {

              }
private:   System::Void   Comunicadora(char*   motor,int   direcao,   int
distance) {

                   CSerial Comunica;
                   Comunica.Open(PORTA,1200);
                   char comando[200];
                   if(Comunica.IsOpened())
                   {
                         sprintf(comando, "\r%s.free()\r",motor);
                         Comunica.SendData(comando, strlen(comando));
                         Sleep(150);
                         if(strcmp(motor, "base")==0)
                               sprintf(comando, "\r%s.T=16\r",motor);
                         else
                               sprintf(comando, "\r%s.T=100\r",motor);
                         Comunica.SendData(comando, strlen(comando));
                         Sleep(200);
                         sprintf(comando,     "\r%s.run(%d,%d)\r",motor,
direcao, distance);
                         Comunica.SendData(comando, strlen(comando));
                         Sleep(200);
                         //sprintf(comando,       "\r%s.free()\r",motor,
direcao, distance);
                         //Comunica.SendData(comando, strlen(comando));
                         //Sleep(200);
                   }

              }

private: System::Void Comunicadora_Cancel(char* motor) {

                   CSerial Comunica;
                   Comunica.Open(PORTA,1200);
                   char comando[200];
                   if(Comunica.IsOpened())
                   {
                         sprintf(comando, "\r%s.stop()\r", motor);
                         Comunica.SendData(comando, strlen(comando));
                         Sleep(200);
                         sprintf(comando, "\r%s.free()\r", motor);
                         Comunica.SendData(comando, strlen(comando));
                         Sleep(200);
                   }

              }
private:   System::Void   OK_manual_Click(System::Object^      sender,
System::EventArgs^  e) {
                    //Dados d;
                    int angleValue = cb_angle->SelectedIndex;
                    int distanceValue = cb_distance->SelectedIndex;
                    int wayValue = cb_way->SelectedIndex;
                    Dados d(angleValue,distanceValue,wayValue);
```

```
                    Comunicadora("base",d.getWay(),d.getDistance());
                    Sleep(500);
                    Comunicadora("plat",0,d.getAngle());

                }
private:    System::Void    Close_Click(System::Object^        sender,
System::EventArgs^  e) {
                    Application::Exit();
                }

private:  System::Void  Customize_botao_Click(System::Object^   sender,
System::EventArgs^  e) {
                    Form^ f = gcnew Customize();
                    f->Show();

                }
private:  System::Void  Cancel_manual_Click(System::Object^    sender,
System::EventArgs^  e) {
                    Comunicadora_Cancel("base");
                    Comunicadora_Cancel("plat");
                }
private:  System::Void  OK_automatic_Click(System::Object^    sender,
System::EventArgs^  e) {
                    char            *            nameValue            =
(char*)(void*)Marshal::StringToHGlobalAnsi(cb_fileSave->Text);
                    FILE *Abrir=fopen("Dados.txt", "r");

                    char nome[100];
                    nome[0]=0;
                    int angulo=0, distancia=0, direcao=0;
                    while(strcmp(nome, nameValue)!=0)
                          fscanf(Abrir,   "%s\n%i\n%i\n%i\n\n",   &nome,
&angulo, &distancia, &direcao);

                    Dados d(angulo,distancia,direcao);
                    Comunicadora("base",direcao,distancia);
                    Sleep(500);
                    Comunicadora("plat",0,angulo);

                    fclose(Abrir);
                }
private:  System::Void  t_name2_TextChanged(System::Object^    sender,
System::EventArgs^  e) {
                }
private:  System::Void  filesave_SelectedIndexChanged(System::Object^
sender, System::EventArgs^  e) {
                }
private:    System::Void    Form1_Load(System::Object^         sender,
System::EventArgs^  e) {
                }
private:  System::Void  Cancel_automatic_Click(System::Object^   sender,
System::EventArgs^  e) {
                }
};
}
```

7.6.1.7 PlataformaMultifuncional.cpp

```
// PlataformaMultifuncional.cpp : main project file.
```

```
#include "stdafx.h"
#include "Form1.h"
#include "Customize.h"

using namespace PlataformaMultifuncional;

[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
	// Enabling Windows XP visual effects before any controls are
created
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);

	// Create the main window and run it
	Application::Run(gcnew Form1());
	return 0;
}
```

8 PROBLEMAS E SOLUÇÕES

Por ser o primeiro projeto o grupo teve dificuldades na criação e confecção dos circuitos impressos os quais seriam utilizados na plataforma. Basicamente foram utilizados os circuitos encontrados nos módulos do professor Afonso. Os módulos foram de grande ajuda, porém estavam confusos e possuíam erros. A equipe estava atenta e não se perdeu muito tempo. O erro foi percebido no momento da montagem da placa conversora. Através do diagrama elétrico tínhamos os pontos exatos a serem ligados, porem ao se observar o desenho que seria projetado na placa, faltava a ligação de pontos indicados pelo diagrama. sabíamos que a projeção estava errada pois antes de tentarmos reproduzir as placas foram feitos inúmeros testes em proto-boards, e sabíamos que o diagrama elétrico estava correto e não o desenho projetado.

No momento da reprodução das placas deve-se atenção ao fato dos soquetes que viriam a ser usados. Como este seria o primeiro contato com devidos componentes ainda não sabíamos qual deveria ser o procedimento, porém colegas nos alertaram de deveria haver uma atenção maior quanto a temperatura que poderia ser trabalhado com cada um dos circuitos integrados, portanto procuramos a especificação referente a cada componente e vimos que realmente o RS232 e o PIC deveriam receber um soquete, pois não poderiam ser soldados diretamente na placa. Outra observação em relação aos módulos solicitados pelo professor Afonso, foi o fato de que alguns circuitos recebiam reguladores de tensão, isso é um método de prevenir perca de componentes ao ser aplicado uma tensão superior. porem, na intenção de redução de custos e minimização da placa resolvemos ser mais cautelosos sempre ao ligar o circuito, ajustando exatamente a fonte na tensão requisitada pelo circuito, abdicando assim o regulador de tensão.

No momento do desenvolvimento do software foram encontrados diversos problemas. um deles se deve ao fato que usarmos o modelo application de programação sendo este o Windows form. O Windows form trabalha com linguagem C# em seu código interno, e nós estávamos desenvolvendo as ações para o aplicativo através da linguagem C++. Para o instante inicial do desenvolvimento enquanto havíamos apenas interagido com valores inteiros para a parte manual, não tínhamos encontrado problemas, porem quando começou-se programar a parte Automática necessitávamos receber do usuário um valor sendo este informado por uma ferramenta existente no Windows form. chamada TextBox. A TextBox retorna um valor string ^, e o valor no

qual deveríamos receber do usuário é uma char *, para que pudesse interagir com perfeição perante a nossa classe de dados. Como solução do problema foi usado uma função que faz a conversão de C++ para C# e em outra situação foi feito o inverso.

9 CONCLUSÃO

Inicialmente não tinhamos conhecimento nenhum sobre como deveria ser desenvolvido o projeto, porem com a ajuda de materiais de apoio disponibilizado pelos os professores Gil Jess, e Afonso Miguel, conseguimos dar os primeiros passos, e conforme a evolução do projeto fomos nos esclarecendo cada vez mais quanto ao desenvolvimento de um projeto.

Escolheu-se um projeto simples por este ser exatamente o primeiro, e não sabermos bem como manipula-lo.

O grupo adquiriu uma carga grande de conhecimento e terá uma maior facilidade em lidar com projetos futuros.

Foi seguido um cronograma elaborado no inicio do semestre que nos ajudou a manter o ritmo e não ficarmos para trás.

Com a construção do hardware foi possível verificar o funcionamento da comunicação entre o computador e os motores de passos, entre as etapas existentes para o funcionamento.

Por fim com muito esforço obtivemos êxito e concluímos o estipulado.

10 REFERÊNCIAS BIBLIOGRÁFICAS

MIGUEL, Afonso. Aquisição de dados via porta serial com PIC12F629/675. Curitiba: 2009.

Comunicação com a porta serial. Disponível em:

< http://www.rogercom.com/>. Acesso em: 30 de março de 2009.

11 ANEXOS