Pontifícia Universidade Católica do Paraná

CCET – Centro de Ciências Exatas e Tecnológicas

Engenharia de Computação

Alex Douglas Fukahori

Arthur Teixeira Brita

Felipe Cornehl

Hélio Pasko Rompkoyski

Projeto Integrado SmartGate

Documentação referente ao Projeto Integrado
desenvolvido no 4º Período do Curso de
Engenharia de Computação da Pontifícia
Universidade Católica do Paraná.
Orientadores: Afonso Ferreira Miguel e Gil Marcos Jess.

Curitiba, Dezembro de 2009

Sumário

1. Introdução

Para que haja maior comodidade na hora de entrar em alguns lugares, sem a necessidade de usar a força humana ou pressionar botões, será, desenvolvido o SmartGate que é um protótipo de cancela que através de um detector de metais percebe se o carro está para entrar ou não, caso positivo a cancela abre automaticamente.

A idéia inicial do projeto foi do aluno Felipe Cornehl, com implementações dos outros integrantes do grupo.

1.1 Objetivo

O SmartGate tem como objetivo abrir a cancela caso seja detectado algum metal próximo e fechar caso não seja detectado nenhum metal.

1.2 Descrição do projeto

No projeto foi montada uma maquete feita de madeira representando uma cancela. Na parte de baixo da estrutura foram embarcados os módulos e os circuitos necessários para fazer o motor obedecer aos controles via porta Serial (RS-232) de um computador para controlar a cancela.

Como o principal objetivo do projeto é abrir a cancela quando algum objeto metálico se aproxima, foi utilizado um detector de metais na parte de cima da estrutura, assim, possibilitando a identificação de um carro metálico, e após essa identificação é enviado um sinal ao computador que manda um comando ao motor servo, e que este abrirá ou fechará a cancela.

1.2.1 Fluxograma do funcionamento

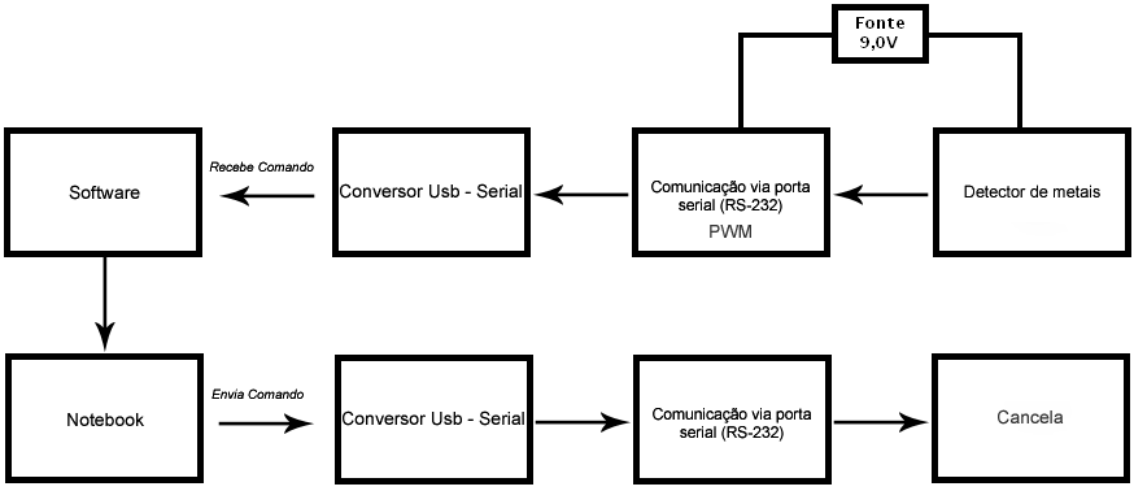

Figura 1: Módulo do controle.

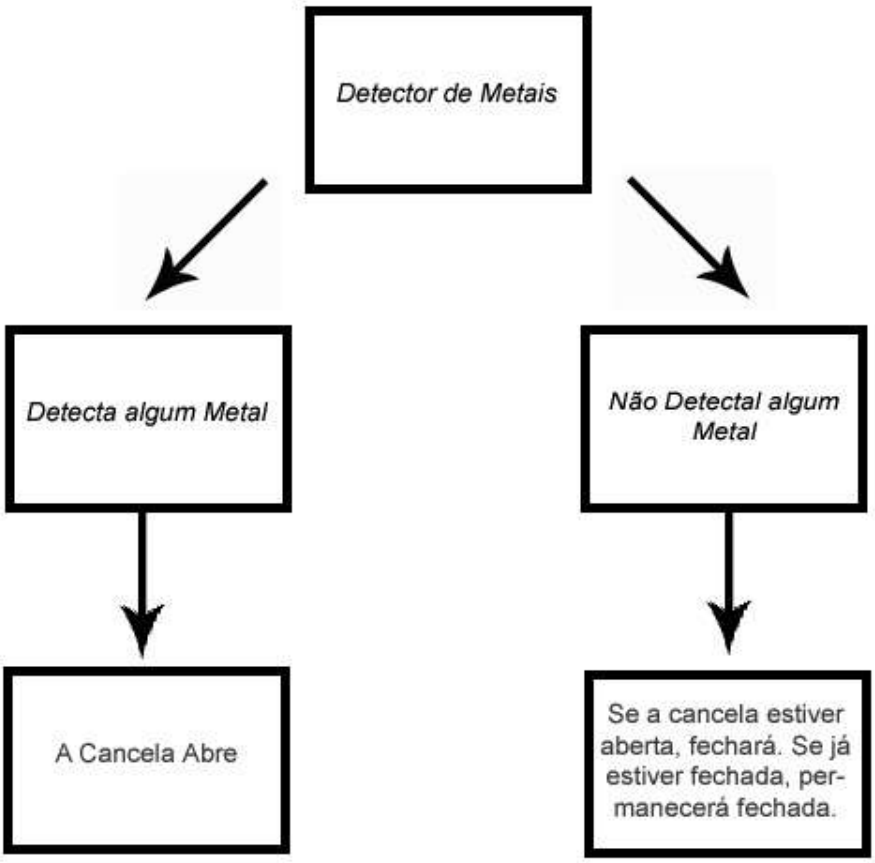

Figura 2: Módulo do funcionamento do Detector de Metais.

1.3 Problemas encontrados

- A bobina feita para o detector de metais tinha o campo magnético muito espalhado, impossibilitando uma detecção precisa;

- O conversor RS-232 para TTL do projeto anterior estava em mal estado e não funcionava adequadamente;

- O kit Altera que era utilizado para comparar a freqüência do oscilador do detector de metais com a freqüência da placa do kit altera (quando o comparador do kit altera detectava a oscilação do detector de metais ele enviava um, caso não, enviava zero). Porém depois de poucos minutos de uso, o kit altera sofria alto aquecimento e parava com o funcionamento do circuito;

- O software utilizado para controlar o motor servo não recebia os dados adequadamente através da porta serial;

- Soldas frias impossibilitaram o êxito dos circuitos;

- Fontes dos laboratórios em mal estado.

1.4 Métodos de resoluções utilizados

- Foi construída uma bobina com núcleo toroidal de ferrite seccionado;

- Foi montado outro circuito conversor RS-232 para TTL;

- Foi construído um detector de metais com comparador de freqüência embutido. Para isso foi utilizado um micro controlador PIC que faz a detecção da oscilação do campo magnético da bobina e compara com a freqüência do PIC, caso detectado a variação ele manda um caso não manda zero;

- O software foi corrigido;

- As soldas frias foram resolvidas após a verificação das placas e posteriormente ressoldando-as;

- Efetuou-se a troca da fonte.

2. Módulos desenvolvidos

2.1 Módulo para comunicação via Porta Serial (RS-232)

Inicialmente foi montado um conversor RS-232 - TTL para possibilitar a comunicação entre o computador e a cancela. O principal componente utilizado pelo circuito é o circuito integrado (veja o tópico 6) MAX232 que é responsável por converter o sinal recebido de uma porta Serial do computador em um sinal TTL ou CMOS (veja o tópico 6).

Lista de componentes utilizados para o conversor:

C1, C2, C3, C4 - Capacitor Eletrolítico de 1uF/16V;

C5 - Capacitor Eletrolítico de 100uF/16V;

R1, R2 - Resistor de 1Kohms 1/4W;

Q1 - Transistor BC548;

CI1 - MAX232;

CI3 - LM7805;

1x Conector RS-232 macho.

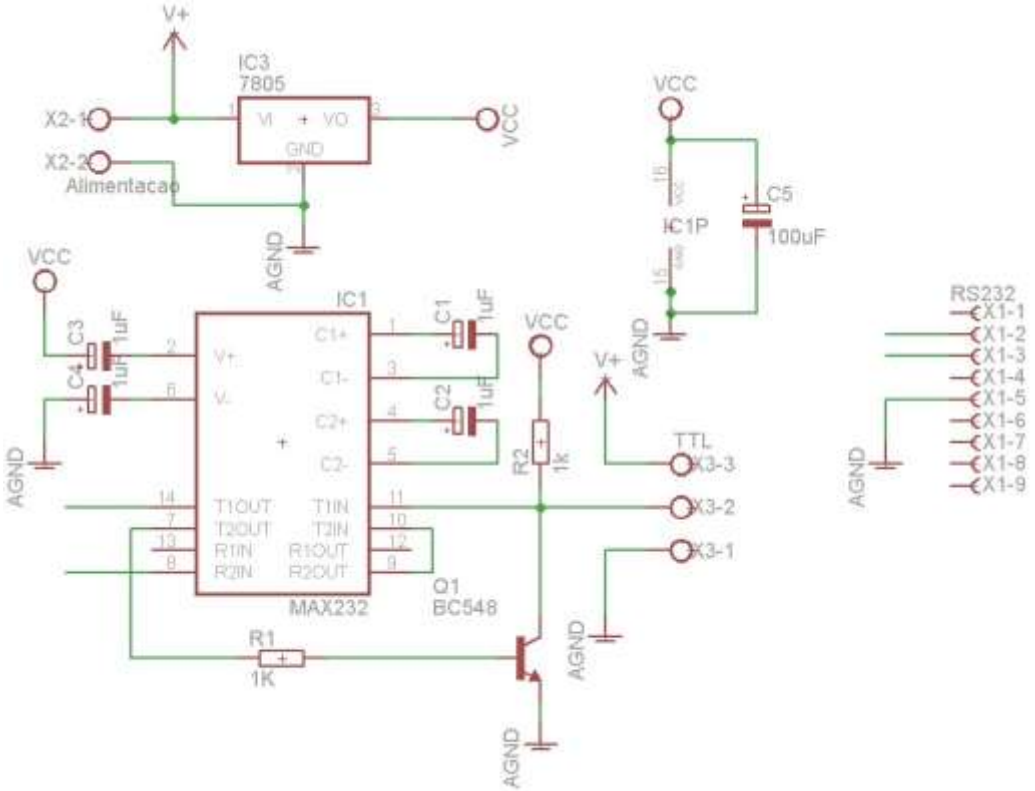

Figura 2.1- Esquemático do conversor RS232 e TTL.

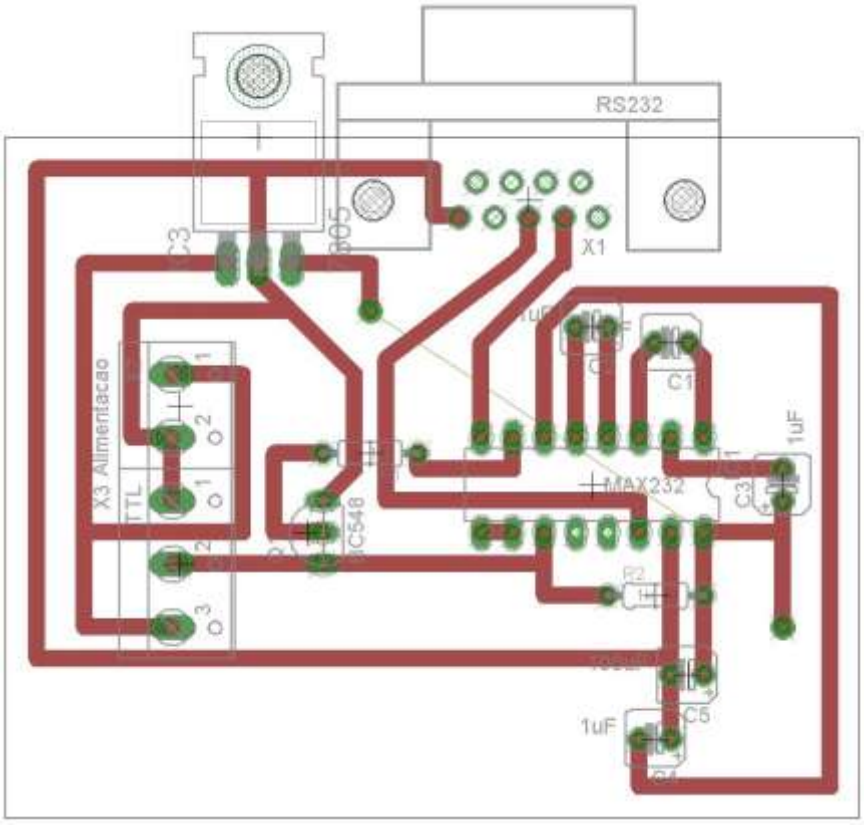

Figura 2.2 – Desenho do circuito Impresso para o Conversor RS232 e TTL.

2.2 PWM

O módulo PWM (Pulse Width Modulation) é responsável por controlar o servo motor. Neste circuito, a potência entregue é controlada a uma carga através do chaveamento da tensão e corrente em freqüências elevadas.

Lista de componentes utilizados:

CI1 – Circuito Integrado – PIC16F629;

CI2 – Regulador de tensão – 78L05;

C1 – Capacitor Eletrolítico – 100uF x 16V;

C2 – Capacitor Eletrolítico – 10uF x 16V.

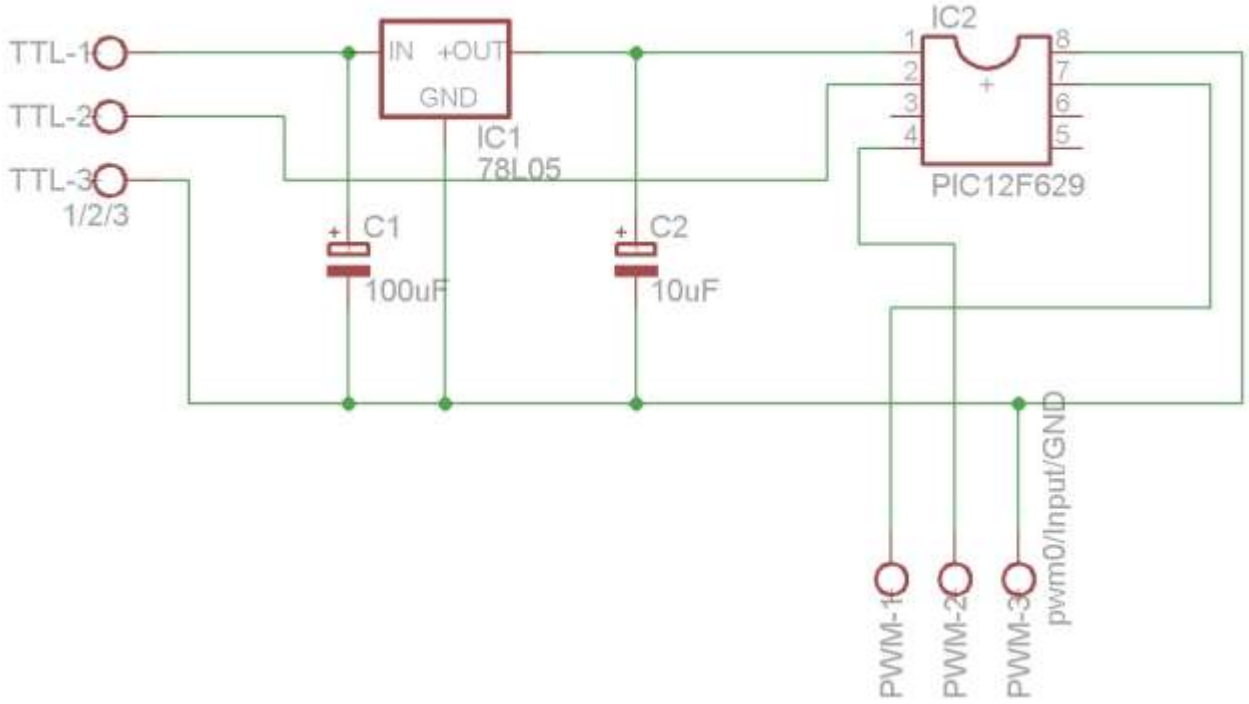

Figura 2.3 – Esquemático do PWM.

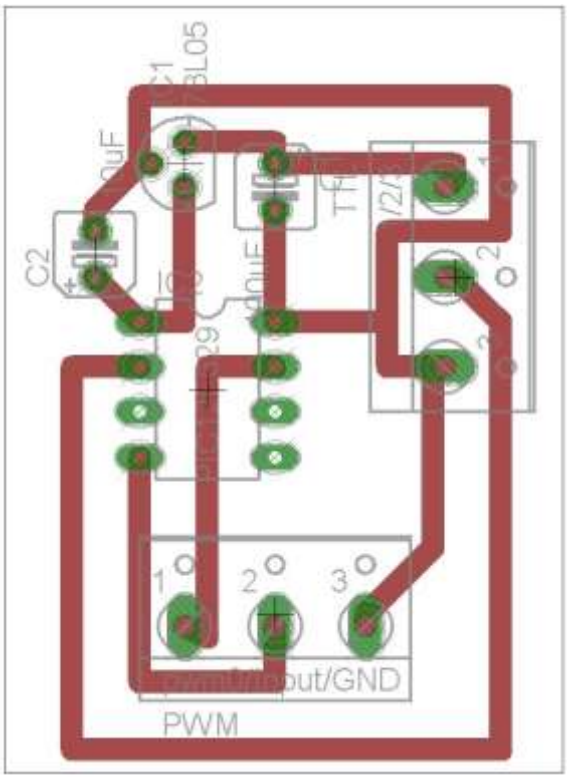

Figura 2.4 – Desenho do circuito impresso PWM.

2.3 Detector de metais

O detector de metais tem como princípio de funcionamento a variação da freqüência que é gerada na bobina e interpretada pelo PIC, e comparado com o próprio oscilador do PIC, e quando há esta variação ele manda zero ou um. Ou seja, quando ele detecta algum metal, é enviado um e zero quando não é detectado.

Antes de começar a funcionar como detector de metais, o circuito faz a calibragem, que seria analisar todo o campo magnético influente no local e ajustar para que haja as condições necessárias de variação sem qualquer margem de ruído e interferência.

Lista de componentes utilizados:

CI1 - Microcontrolador PIC12F683;

CI2 - regulador de tensão LM7805;

C1, C2 - Capacitor de 15pF;

C3 - Capacitor de 100nF ;

R1, R2 - Resistor de 330 ohms;

2x Leds;

2x Bornes de 3 pinos.

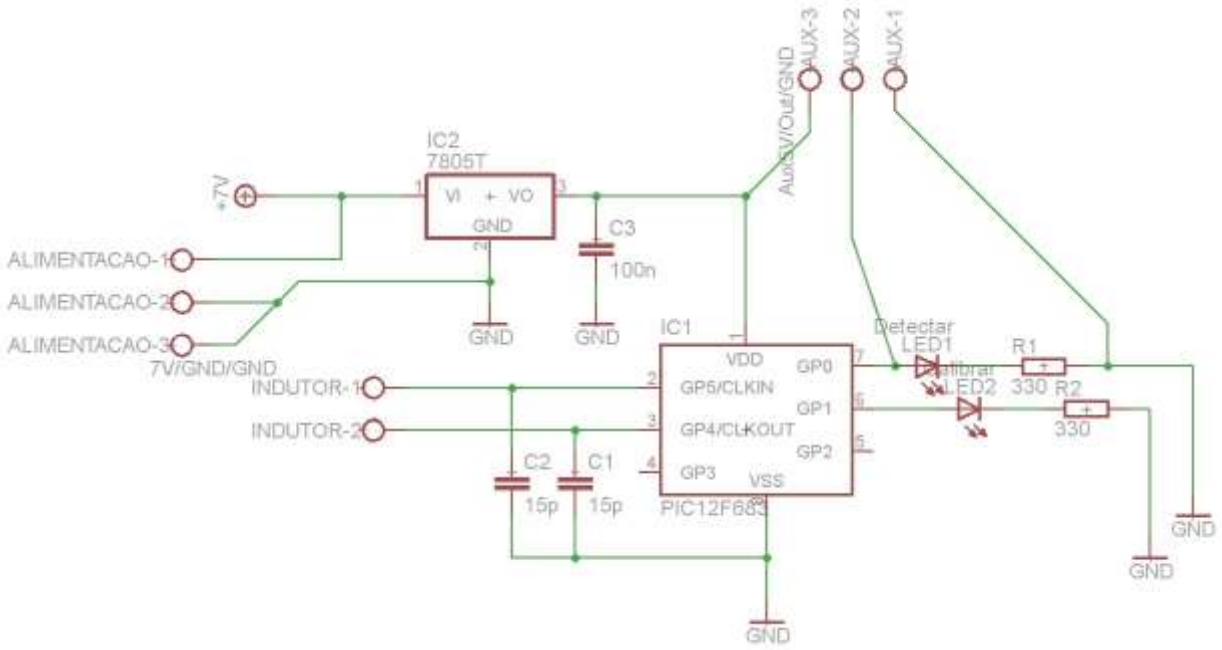

Figura 2.5 – Esquemático do circuito do Detector de metais.

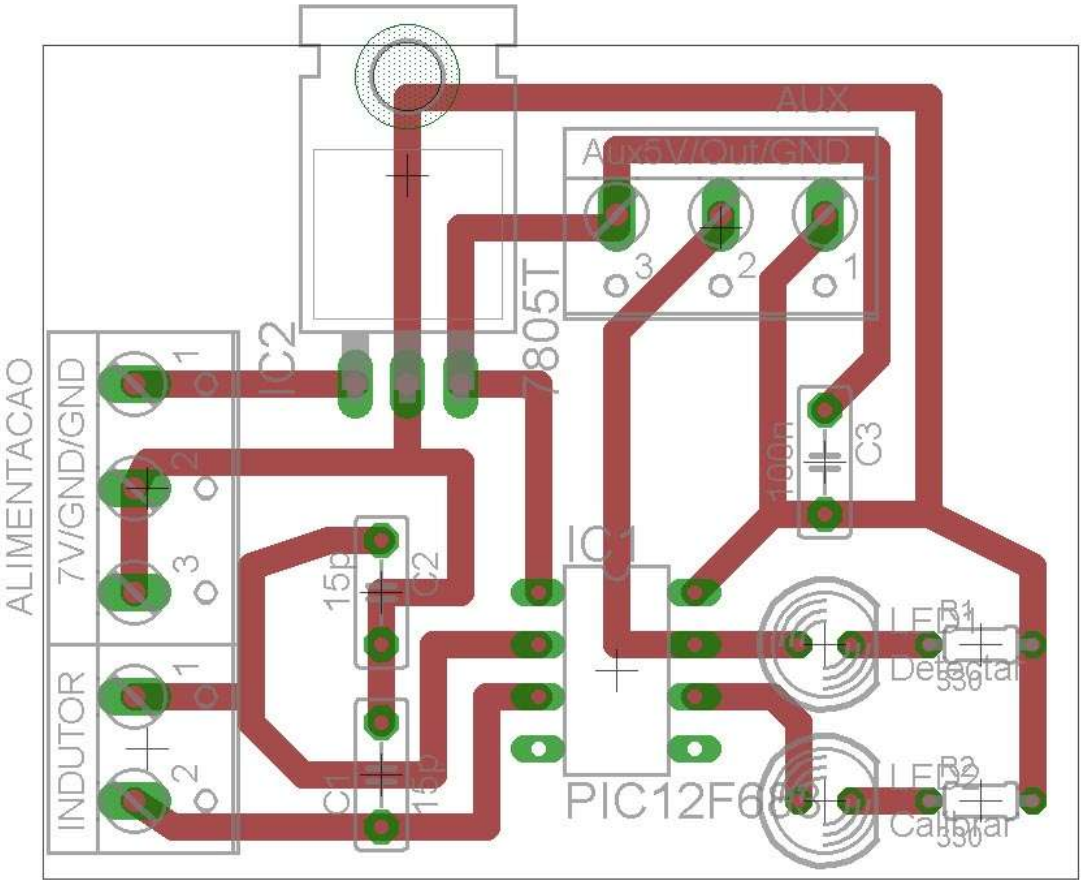

Figura 2.6 – Desenho do circuito impresso do Detector de Metais.

2.4 Circuito de senha para bloquear o funcionamento da cancela

Para possibilitar o bloqueio do funcionamento da cancela, foi desenvolvido um circuito que possibilita a inserção de uma seqüência de números no qual bloqueia a corrente do detector, fazendo com que a cancela não abre nem feche por um tempo pré-determinado.

O principal componente do circuito é o PIC12F675. Ele guarda a seqüência de senha no próprio EEPROM o microcontrolador tem integrado, assim, possibilitando a comparação com os valores digitados pelo teclado. Para distinguir os botões pressionados, foram colocados vários resistores de 1kohms em série no intervalo de cada botão, assim, ao pressionar os botões é enviada uma tensão diferente ao microcontrolador, possibilitando a diferenciação dos botões pressionados.

Lista de componentes utilizados:

CI1 – LM7805;

CI2 – Microcontrolador PIC12F675;

C1, C4 – Capacitor 100nF;

C2, C3 – Capacitor eletrolítico 220uF;

C5 – Capacitor 10nF;

R1 até R14 – Resistor 1Kohms;

R15 – Resistor 100Kohms;

R16, R17 – Resistor de 4,7Kohms;

S1 até S14 – Botão de pressão;

D1, D2 – Diodo 1N4148;

T1, T2 – Transistor BC547;

2x LED’s;

2x Relés.

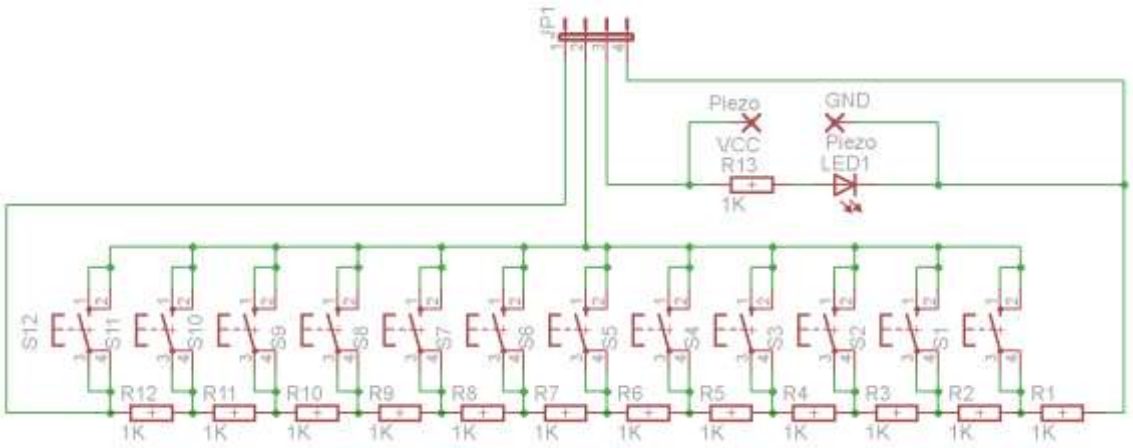

Figura 2.7 – Esquemático do circuito do Teclado.

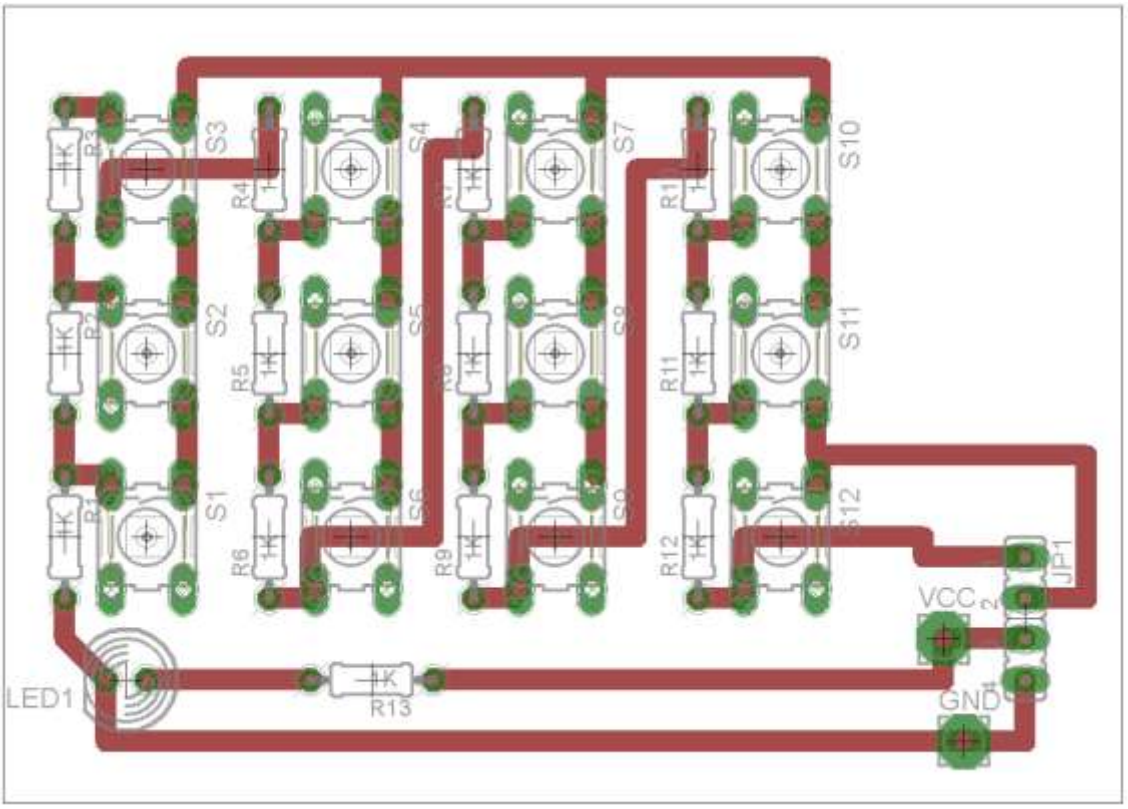

Figura 2.8 – Layout do circuito do Teclado.

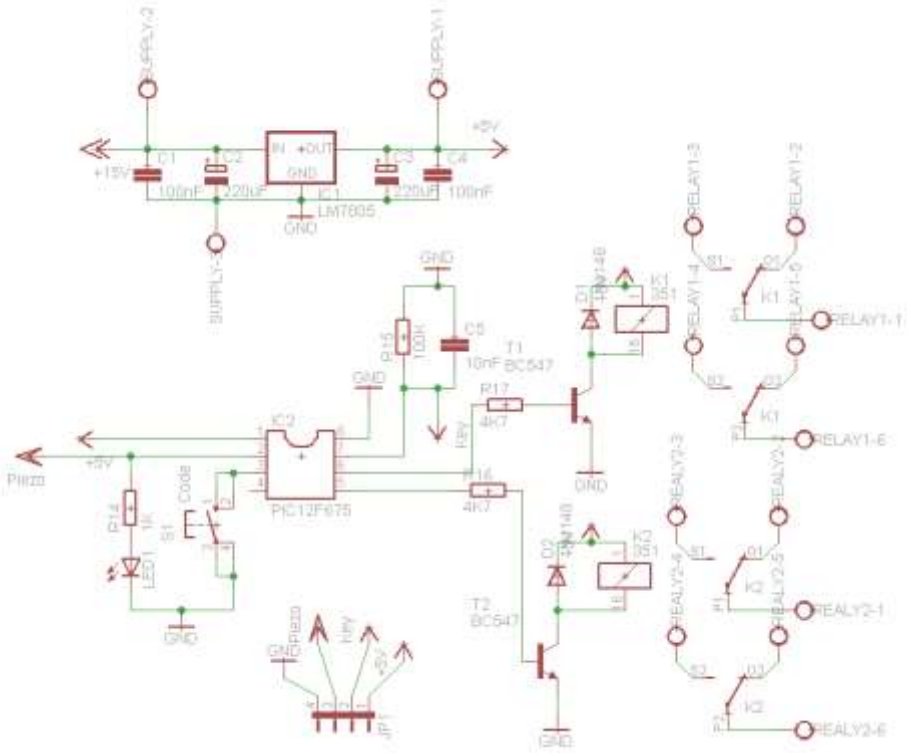

Figura 2.9 – Esquemático do circuito principal.

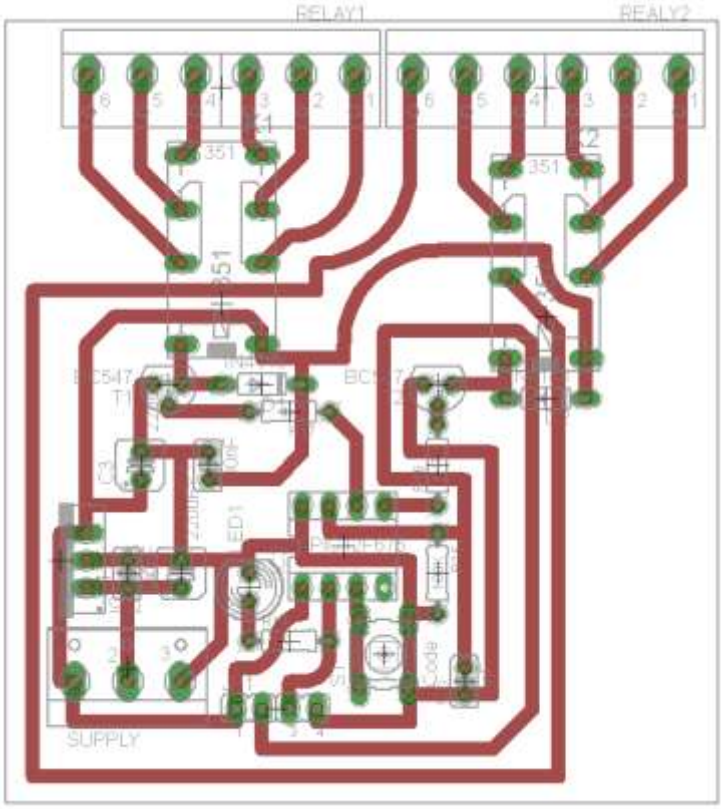

Figura 2.10 – Layout do circuito principal.

3. Software

3.1 Funcionamento

O software basicamente utiliza a classe CSerial para enviar aos circuitos montados (RS-232) e ao motor servo as informações que estes devem receber.

O software também está preparado para receber o sinal enviado pelo detector de metais, caso seja detectado algum metal o software recebe o valor "1" e envia o comando ao motor para abrir a cancela. Caso não seja detectado algum metal o software recebe o valor "0" e envia o comando ao motor para fechar a cancela.

3.2 Código

Código "funcoes.h":

```
#ifndef FUNCOES_H
#define FUNCOES_H

unsigned int portaC, taxCom, cont1, sleepDelay;//Variáveis universais
void abreArq(void);//Protótipo de funcao para abrir arquivo de dados.

#endif
```

Código "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 ); }
	BOOL GetCTS();

protected:
	BOOL WriteCommByte( unsigned char );

	HANDLE m_hIDComDev;
	OVERLAPPED m_OverlappedRead, m_OverlappedWrite;
	BOOL m_bOpened;

};

#endif
```

Código “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 )
{
	//TRACE1(buffer);
	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 );

}

BOOL CSerial::GetCTS()
{
	DWORD lpModemStat;
	if( !m_bOpened || m_hIDComDev == NULL ) return FALSE;
	if(!GetCommModemStatus(m_hIDComDev,&lpModemStat))
		return FALSE;
	return (lpModemStat & MS_CTS_ON)?TRUE:FALSE;
}
```

Código “main.cpp”:

```
#include "stdafx.h"
#include "serial.h"
#include "funcoes.h"
#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <fstream>

#define clrscr system("CLS")

using namespace std;

int menu(void)//Menu para opções.
{
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
BACKGROUND_BLUE | BACKGROUND_INTENSITY);//Muda a cor do BackGround
recebendo os valores BACKGROUND_RED | BACKGROUND_GREEN |
BACKGROUND_BLUE | BACKGROUND_INTENSITY.
	//SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
128);//Muda cor da letra, o segundo valor inteiro variando de 0 E15.
	int opcao, ESCquit;
	cout<< "Digite a opcao desejada:" <<endl;
	cout<< "1. Editar a porta COM(atual="<<portaC<<");"<<endl;
	cout<< "2. Editar a taxa de
comunicacao(atual="<<taxCom<<");"<<endl;
	cout<< "3. Iniciar o programa do SmartGate;"<<endl;
	cout<< "4. Editar o valor do Delay necessario para o PIC
responder(atual="<<sleepDelay<<" ms);"<<endl;
```

```
	cout<< "5. Editar o valor do Contador de '1'
consecutivos(atual="<<cont1<<");"<<endl;
	cout<< "6. Sair." <<endl;
	cout<< "Opcao: ";
	cin>> opcao;//Receber o valor digitado pelo usuário.
	cout<<"\n\n";
	return opcao;//Retorna o valor digitado pelo usuário.
}

void abreArq(void)//Funcao para abrir arquivo de dados.
{
	FILE *arq;
	arq = fopen("dados.txt","r");//Abrindo arquivo para leitura .
	rewind (arq);
	fscanf(arq, "%i", &portaC);//Associando o primeiro valor do
arquivo para a variável portaC.
	fscanf(arq, "%i", &taxCom);//Associando o segundo valor do
arquivo para a variável taxCom.
	fscanf(arq, "%i", &sleepDelay);//Associando o segundo valor do
arquivo para a variável taxCom.
	fscanf(arq, "%i", &cont1);//Associando o segundo valor do
arquivo para a variável taxCom.
	if(arq!=NULL)//Testa se o arquivo foi aberto com sucesso.
	{
		SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
2);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
		cout<<"Arquivo foi aberto com sucesso."<<endl<<endl;
	}
	else
	{
		SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
4);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
		cout<<"Nao foi possivel abrir o arquivo."<<endl<<endl;
	}
}

int _tmain(int argc, _TCHAR* argv[])
{
	/*keybd_event(VK_MENU,0x38,0,0);//Código para FullScreen do
console. Obs.: Não funciona para WinVista ou superiores.
	keybd_event(VK_RETURN,0x1c,0,0);//Código para FullScreen do
console. Obs.: Não funciona para WinVista ou superiores.
	keybd_event(VK_RETURN,0x1c,KEYEVENTF_KEYUP,0);//Código para
FullScreen do console. Obs.: Não funciona para WinVista ou superiores.
	keybd_event(VK_MENU,0x38,KEYEVENTF_KEYUP,0);//Código para
FullScreen do console. Obs.: Não funciona para WinVista ou
superiores.*/

	int op=0, sai=1;//Variáveis da repetiçao.
	int bu=50, read=0;//Variáveil para o tamanho da String de
leitura da porta Serial e para chamar função de leitura.
	int contador=0;//Variável para contar os "1" consecutivos.
	char quitOp=' ';//Variável para sair durante o programa rodando.
	char teste[]="1";//String para testar o Buffer.
	CSerial RS;//Chamando Construtor.
	abreArq();//Chama a função para abrir o arquivo.
	do
	{
		op = menu();//Associando o valor retornado pelo menu.
```

```
            switch(op)//Switch do menu.
            {
                  case 1:
                        {
                              cout<< "Digite o numero da porta COM: ";
                              cin>> portaC;//Recebendo o valor da porta
COM.
                              cout<<"\n\n";//Frescura.
                              if(portaC>2 || portaC<1)//Teste de erro.
                              {

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
4);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                    cout<< "Porta COM incorreta, favor
refazer."<<endl<<endl;
                                    abreArq();//Chama a função para
abrir o arquivo.
                              }
                              break;
                        }
                  case 2:
                        {
                              cout<< "Digite a taxa de comunicacao: ";
                              cin>> taxCom;//Recebendo a taxa de
Comunicação.
                              cout<<"\n\n";//Frescura.
                              if(taxCom==0 || taxCom<0)//Teste de erro.
                              {

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
4);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                    cout<< "Taxa de comunicacao
incorreta, favor refazer."<<endl<<endl;
                                    abreArq();//Chama a função para
abrir o arquivo.
                              }
                              break;
                        }
                  case 3:
                        {
                              if(taxCom==0 || taxCom<0 || portaC>2 ||
portaC<1)//Teste de erro da porta e taxa de comunicação.
                              {

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
4);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                    cout<< "Erro ao verificar o valor a
porta e a taxa de comunicacao."<<endl<<endl;
                              }
                              else
                              {

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
2);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                    cout<< "Valor da porta e taxa de
comunicacao verificada com sucesso."<<endl<<endl;
                              }
```

```
                              RS.Open(portaC,taxCom);//Abrindo porta
Serial para iniciar a comunicação.
                              char comando0[20], comando1[20];//Strings
para os comandos Eserem enviados para a porta Serial.

                             if(RS.Open(portaC,taxCom))//Teste de erro.
                              {

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
2);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                    cout<< "Porta aberta com
sucesso."<<endl<<endl;
                                    Sleep(1000);//Pausa 1s para dar
tempo do usuário ler a mensagem.
                              }
                              else
                              {

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
4);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                    Beep(700, 5000);//Aciona o Beep com
frequencia=700 e tempo=5000ms.(pode ser usado printf("\a"); para a msm
utilidade)
                                    cout<< "Falha ao abrir a
porta."<<endl<<endl;
                                    Sleep(1000);//Pausa 1s para dar
tempo do usuário ler a mensagem.
                              }

                              sprintf(comando1,
"led.pwm.T=96\r");//Copiando o comando EString.
                              RS.SendData(comando1,
strlen(comando1));//Enviando o comando.
                              Sleep(sleepDelay);//Pausa em tempo=X ms.
                              do
                              {
                                    do
                                    {

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
6);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                          cout << "Pressione ESC para
sair."<<endl<<endl<<endl;
                                          if(kbhit())///Se alguma tecla
for precionado entra nesse caso.
                                          {
                                                quitOp=getch();
                                          }
                                          sprintf(comando0,
"led.in.get()\r");//Copiando o comando EString.
                                          RS.SendData(comando0,
strlen(comando0));//Enviando o comando.
                                          Sleep(sleepDelay);//Pausa em
tempo=X ms.
                                          char *Rbuffer= new
char[bu];//Alocando Memória para receber dados da porta Serial.
```

```
                                        read=RS.ReadData(Rbuffer,
bu);//Chama função que l□Eos dados recebidos pela porta Serial.

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
11);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                        cout<<"String Recebido pela
Porta Serial:"<<endl<<endl<<Rbuffer<<endl<<endl;//Imprime o buffer de
leitura na tela.
                                        char *procurar1 =
strpbrk(Rbuffer, teste);//Verifica que a primeira string possui algum
caracter igual □Ealgum caracter da segunda string.
                                        if(procurar1!=NULL)//Testa se
recebeu algum dado na porta Serial.
                                        {
                                              contador++;//Adiciona
+1 no contador de "1" consecutivos.
                                        }
                                        else
                                        {
                                              contador=0;//Zera o
contador de "1" consecutivos.
                                        }
                                        if(contador>=cont1)//Testa o
valor do contador pra saber se detectou o Metal.
                                        {

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
2);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                              cout<< "Metal
detectado."<<endl;
                                              sprintf(comando1,
"led.pwm0.on(4)\r");//Copiando o comando □EString.
                                              RS.SendData(comando1,
strlen(comando1));//Enviando o comando □EString.

      Sleep(sleepDelay);//Pausa em tempo=X ms.
                                              delete
[]Rbuffer;//Desalocando Memória que recebeu o dado da porta Serial.
                                              Sleep(2500);//Delay
para esperar o 'carro' passar.
                                              clrscr;//Limpa Tela.
                                        }
                                        else if(contador<cont1)
                                        {

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
4);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                              cout<< "Metal nao
detectado."<<endl;
                                              sprintf(comando1,
"led.pwm0.on(7)\r");//Copiando o comando □EString.
                                              RS.SendData(comando1,
strlen(comando1));//Enviando o comando □EString.

      Sleep(sleepDelay);//Pausa em tempo=X ms.
                                              delete
[]Rbuffer;//Desalocando Memória que recebeu o dado da porta Serial.
                                              clrscr;//Limpa Tela.
```

```
                                    }
                                }while(quitOp!=27);//Loop enquanto
o usuário não aperta ESC.

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
15);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                    cout << "Saindo..."<<endl;
                                    quitOp=0;//Zera o quitOp, s□Epara
garantir.
                                    RS.Close();//Fechando porta.
                                    break;
                              }while(RS.IsOpened());//Loop enqunato a
porta estiver aberta.
                              RS.Close();//Fechando porta.
                              if(RS.Close())//Teste se fechou a porta.
                              {
                                    cout<<"Porta fechada."<<endl;
                              }
                              break;
                        }
                  case 4:
                        {
                              cout<< "Digite o tempo de Delay para
enviar cada comando: ";
                              cin>> sleepDelay;//Recebendo o valor do
Delay.
                              cout<<"\n\n";//Frescura.
                              if(sleepDelay==0 ||
sleepDelay<100)//Teste de erro.
                              {

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
4);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                    cout<< "Delay muito pequeno, favor
refazer."<<endl<<endl;
                                    abreArq();//Chama a função para
abrir o arquivo.
                              }
                              break;
                        }
                  case 5:
                        {
                              cout<< "Digite a quantidade de vezes que
será necessário detectar o valor '1': ";
                              cin>> cont1;//Recebendo o valor do
contador.
                              cout<<"\n\n";//Frescura.
                              if(cont1==0 || cont1<0)//Teste de erro.
                              {

      SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
4);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                    cout<< "Valor incorreto, favor
refazer."<<endl<<endl;
                                    abreArq();//Chama a função para
abrir o arquivo.
                              }
                              break;
```

```
                        }
                case 6:
                        {
                                RS.Close();//Garantindo o fechamento das
portas.
                                sai=0;//Fechar a repetição do menu.
                                break;
                        }
                default:
                        {

        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
4);//Muda a cor do BackGround recebendo os valores BACKGROUND_RED |
BACKGROUND_GREEN | BACKGROUND_BLUE | BACKGROUND_INTENSITY.
                                cout<<"Opcao incorreta, favor escolher
outra opcao."<<endl<<endl;
                                Sleep(1000);//Pausa 1s para dar tempo do
usuário ler a mensagem.
                                clrscr;//Limpa Tela.
                                break;
                        }
                }
        }while(sai);//Repetição do menu.
        cout<<"Todos os direitors reservados para A.D.F. ;-)"<<endl;
        system("pause");//Frescura para mostrar a mensagem no fim, caso
não queira mostra coloque '>>NULL' ao lado do pause.
        return 0;
}
```

3.3 Telas

Para que seja possível mudar a porta serial e outras opções referentes ao funcionamento do software de modo simples. É exibido logo que se inicia o software um menu com a opção de escolher a tarefa desejada:

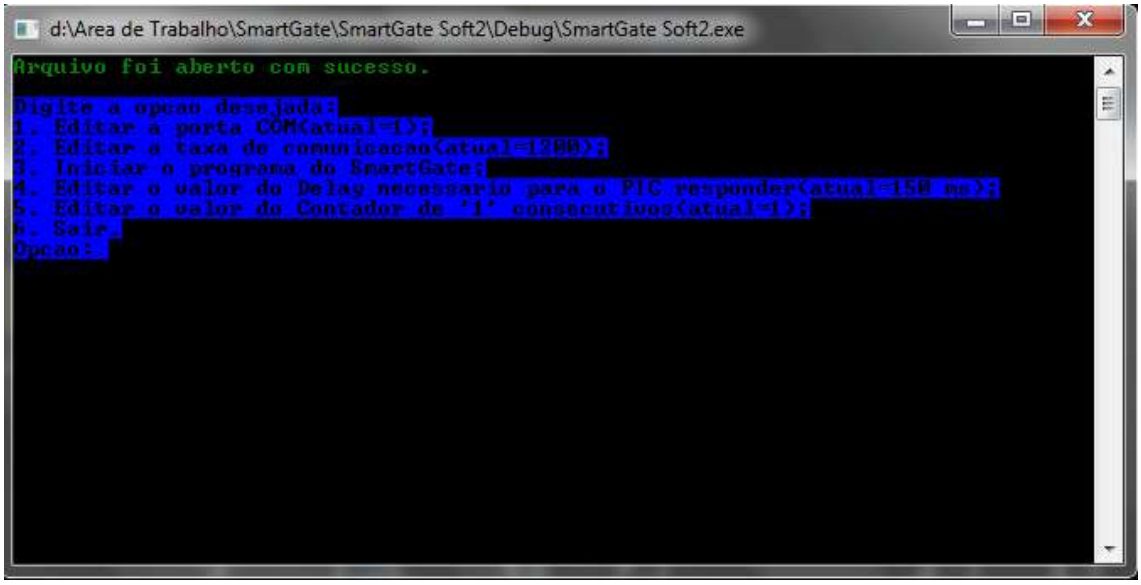

Figura 3.0 - Menu do Software.

Como o Software recebe o sinal do detector de metais, ele exibe na tela caso o metal seja detectado ou caso não:

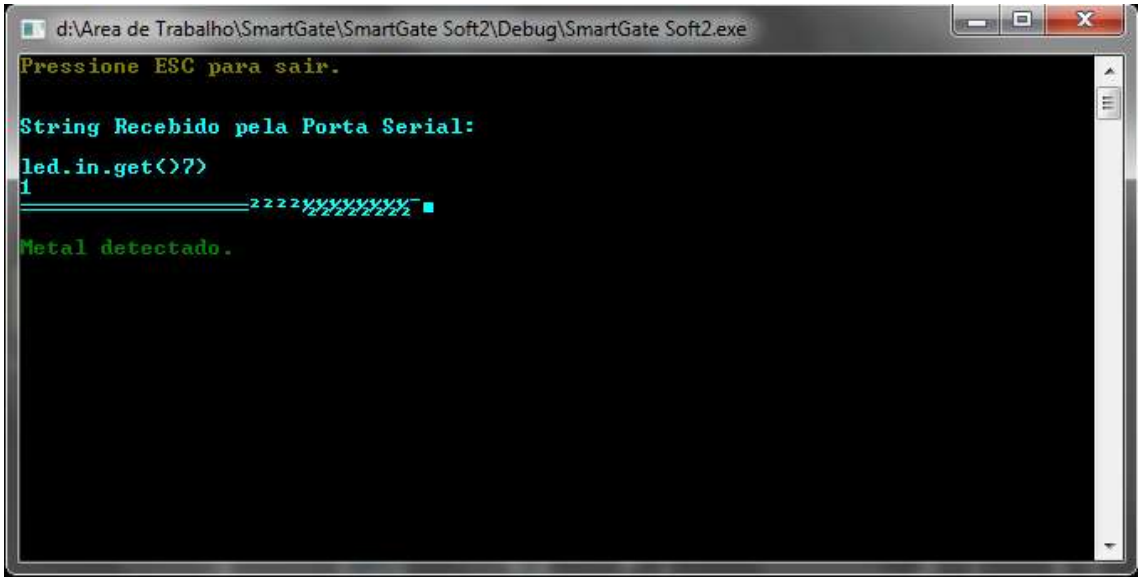

Figura 3.1 - Tela exibida pelo Software ao detectar algum metal.

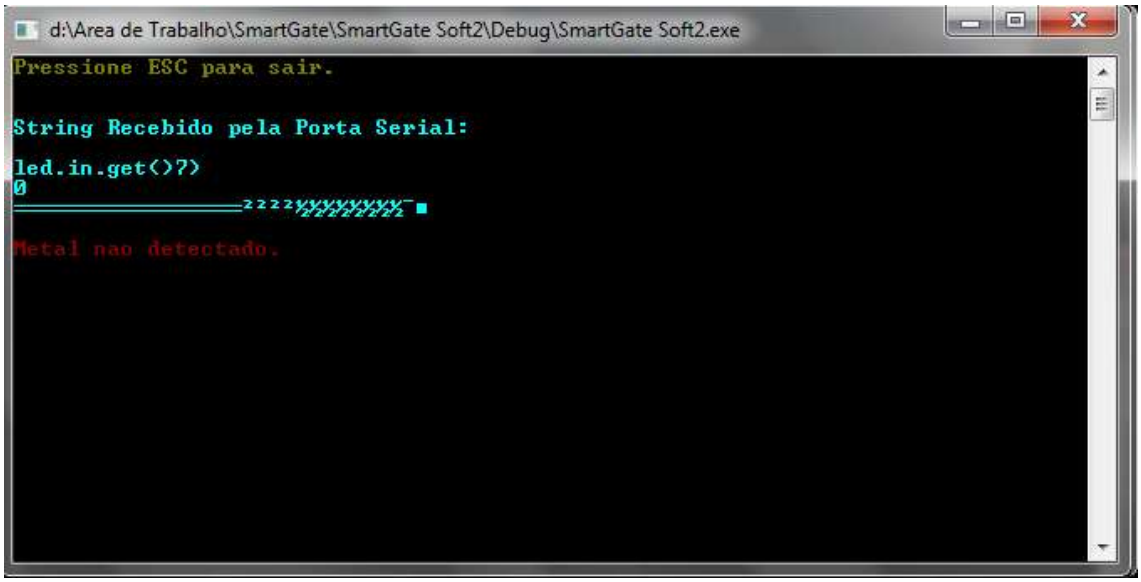

Figura 3.2 - Tela exibida pelo Software ao não detectar algum metal.

4. Conclusão

Até a data da entrega do projeto todas as metas foram estabelecidas, ou seja, todos os módulos desenvolvidos funcionaram conforme o esperado, assim a cancela abriu logo que algum metal fosse detectado.

5. Galeria de Fotos

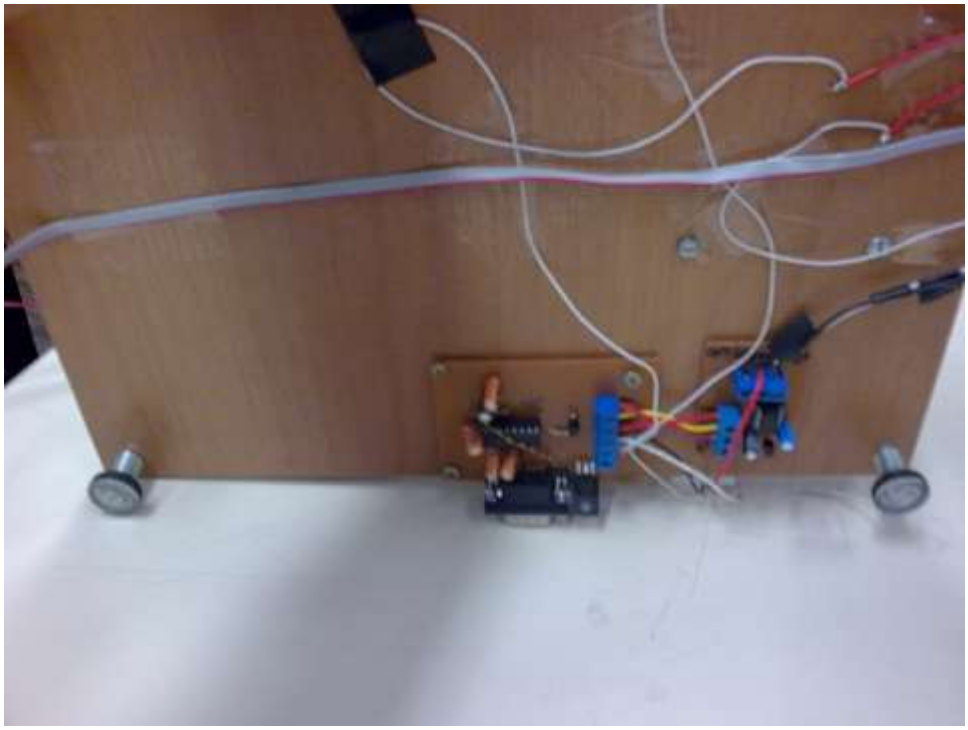

Figura 5.0 – Foto as placas embaixo da maquete.

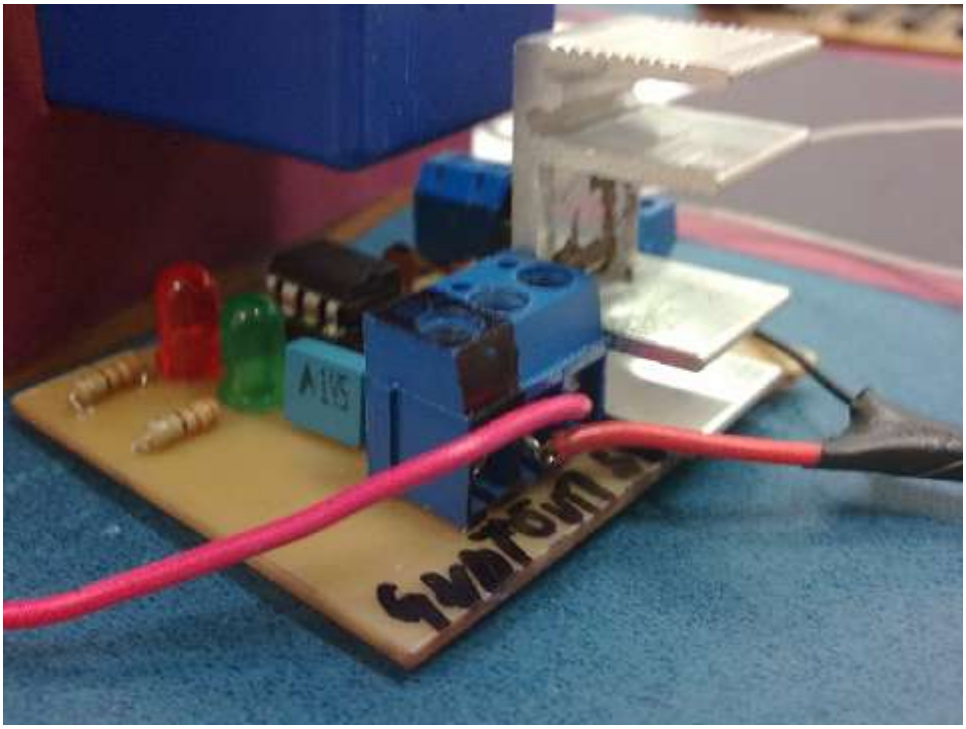

Figura 5.1 – Foto mostrando a placa do Detector de metais.

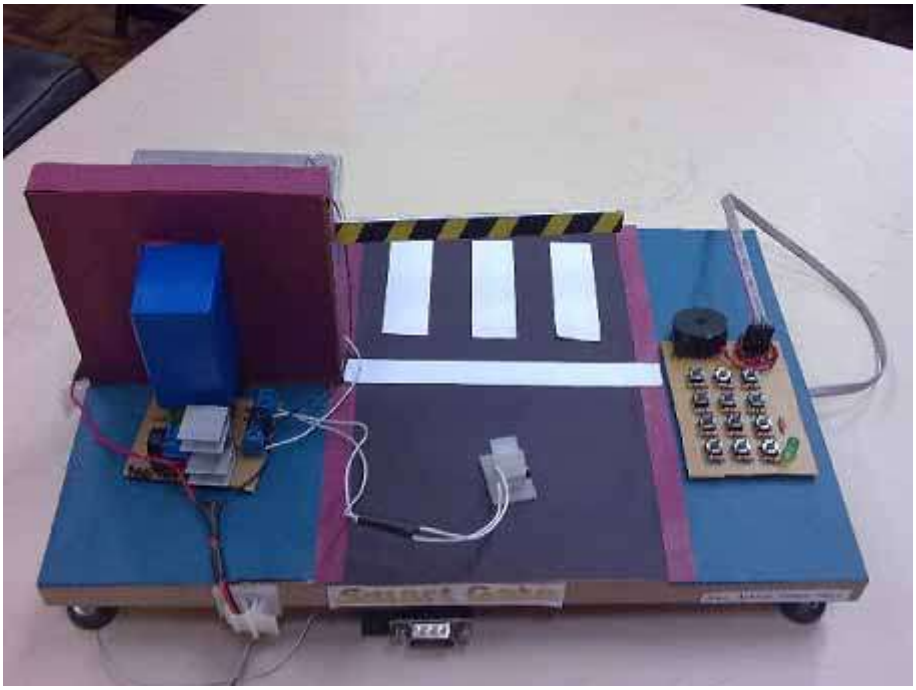

Figura 5.2 – Foto da maquete numa visão geral.

6. O que é?

6.1 Corrente elétrica

A corrente elétrica é o movimento ordenado de partículas eletricamente carregadas. Vamos explicar a corrente elétrica a partir de um condutor metálico (um fio elétrico, por exemplo). Dentro desses condutores há muitos elétrons livres descrevendo um movimento caótico, sem direção determinada. Ao aplicar-se uma diferença de potencial entre dois pontos do metal (ligando as pontas do fio a uma bateria, por exemplo), estabelece-se um campo elétrico interno e os elétrons passam a se movimentar numa certa ordem, constituindo assim a corrente elétrica.

A corrente elétrica é definida como a razão entre a quantidade de carga que atravessa certa secção transversal (corte feito ao longo da menor dimensão de um corpo) do condutor num intervalo de tempo. A unidade de medida é o *Coulomb por segundo* (C/s), chamado de *Ampère* (A) no SI em homenagem ao físico e matemático francês André-Marie Ampère (1775-1836).

Fonte: UFPA.

6.2 Diodo

Um diodo é o tipo mais simples de semicondutor. De modo geral, um semicondutor é um material com capacidade variável de conduzir corrente elétrica. A maioria dos semicondutores é feita de um condutor pobre que

teve impurezas (átomos de outro material) adicionadas a ele. O processo de adição de impurezas é chamado de dopagem**.**

Um semicondutor com elétrons extras é chamado material tipo-N, já que tem partículas extras carregadas negativamente. No material tipo-N, elétrons livres se movem da área carregada negativamente para uma área carregada positivamente.

Um semicondutor com elétrons em buraco extras é chamado material tipo-P, já que ele efetivamente tem partículas extras carregadas positivamente. Os elétrons podem pular de buraco em buraco, movendo-se de uma área carregada negativamente para uma área carregada positivamente. Como resultado, os próprios buracos parecem se mover de uma área carregada positivamente para uma área carregada negativamente.

Um diodo é composto por uma seção de material tipo-N ligado a uma seção de material tipo-P, com eletrodos em cada extremidade. Essa combinação conduz eletricidade apenas em um sentido. Quando nenhuma tensão é aplicada ao diodo, os elétrons do material tipo-N preenchem os buracos do material tipo-P ao longo da junção entre as camadas, formando uma zona vazia. Em uma zona vazia, o material semicondutor volta ao seu estado isolante original - todos os buracos estão preenchidos, de modo que não haja elétrons livres ou espaços vazios para elétrons, e assim a corrente não pode fluir.

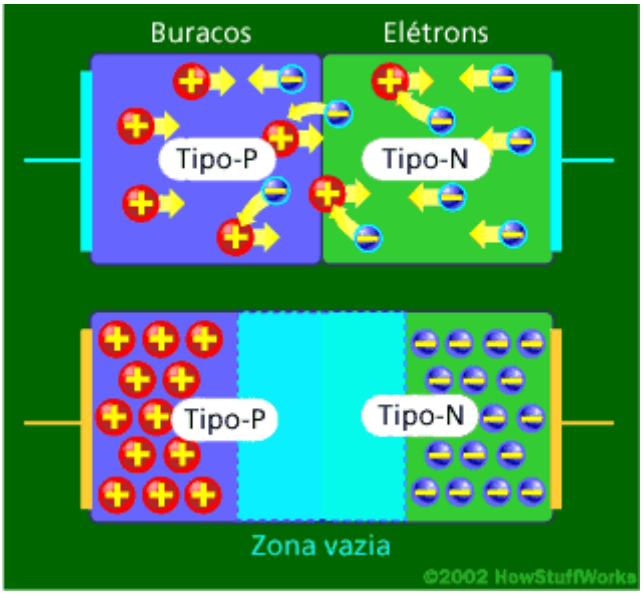

Figura 6.0 – Por dentro de um diodo.

Para se livrar da zona vazia, você precisa que elétrons se movam da área tipo-N para a área tipo-P e que buracos se movam no sentido inverso. Para fazer isto, você conecta o lado tipo-N do diodo ao terminal negativo do circuito e o lado tipo-P ao terminal positivo. Os elétrons livres no material tipo-N são repelidos pelo eletrodo negativo e atraídos para o eletrodo positivo. Os buracos no material tipo-P se movem no sentido contrário. Quando a diferença de potencial entre os eletrodos é alta o suficiente, os elétrons na zona vazia são retirados de seus buracos e começam a se mover livremente de novo. A zona vazia desaparece e a carga se move através do diodo.

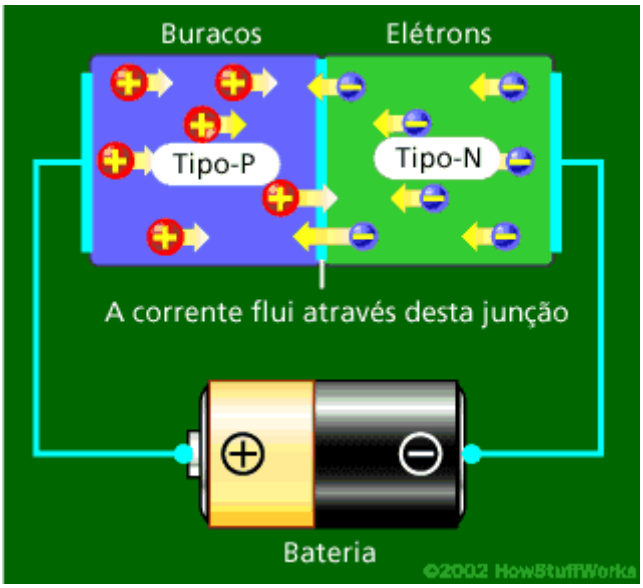

Figura 6.1 – Energia Elétrica fluindo pelo diodo.

Se você tentar mover a corrente no sentido oposto, com o lado tipo-P conectado ao terminal negativo do circuito e o lado tipo-N conectado ao pólo positivo, a corrente não fluirá. Os elétrons negativos no material tipo-N são atraídos para o eletrodo positivo. Os buracos positivos no material tipo-P são atraídos para o eletrodo negativo. Nenhuma corrente flui através da junção porque os buracos e os elétrons estão cada um se movendo no sentido errado. A zona vazia então aumenta.

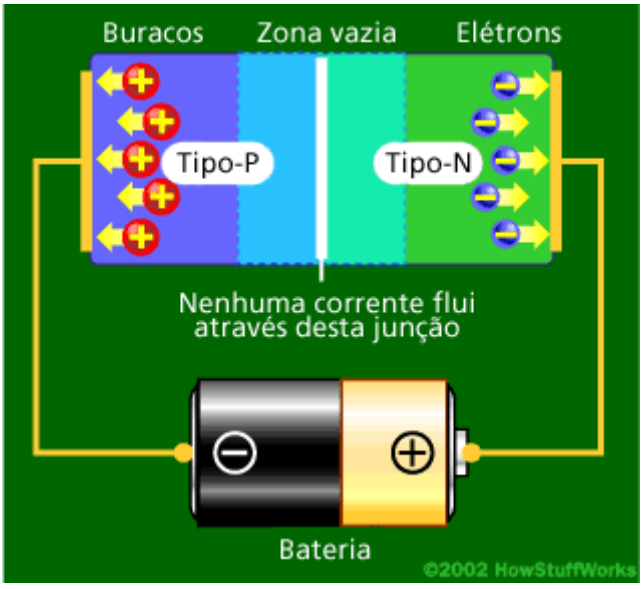

Figura 6.2 – Diodo não fluindo corrente elétrica.

Figura 6.3 – Diodos existentes no mercado.

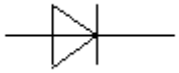

Figura 6.4 - Desenho esquemático de um diodo.

Fonte: HowStuffWorks

6.3 LED

LED’s é uma abreviação de “Light Emitting Diode” ou em português seria um Diodo Emissor de Luz, ele nada mais é do que um semicondutor que ao ser energizado ele emite uma luz. Ele é uma junção de semicondutores do tipo P e N, onde, o P é o positivo ou cátodo (falta de elétrons) e o N é o negativo ou o ânodo (excesso de elétrons), para mais detalhes sobre semicondutores a página onde está sendo explicado sobre os transistores nesse documento.

A cor do LED depende do cristal e da impureza de dopagem com que o componente é fabricado. O LED que utiliza o arseneto de gálio emite radiações infravermelhas. Dopando-se com fósforo, a emissão pode ser vermelha ou amarela, de acordo com a concentração. Utilizando-se fosfeto de gálio com dopagem de nitrogênio, a luz emitida pode ser verde ou amarela. Hoje em dia, com o uso de outros materiais, consegue-se fabricar LED’s que emitem luz azul, violeta e até ultravioleta. Existem também os LED’s brancos, mas esses são geralmente LED’s emissores de cor azul, revestidos com uma camada de fósforo do mesmo tipo usado nas lâmpadas fluorescentes, que absorve a luz azul e emite a luz branca. Com o barateamento do preço, seu alto rendimento e sua grande durabilidade, esses LED’s tornaram-se ótimos substitutos para as lâmpadas comuns, e devem substituí-las a médio ou longo prazo. Existem também os LED’s chamados RGB, e que são formados por três "chips", um vermelho (R de red), um verde (G de green) e um azul (B de blue), esses LED’s podem ser utilizados juntamente com um microcontrolador para variar as cores do modo que quiser.

Em geral, os leds operam com nível de tensão de 1,6 a 3,3V, sendo compatíveis com os circuitos de estado sólido. É interessante notar que a tensão é dependente do comprimento da onda emitida. Assim, os leds infravermelhos geralmente funcionam com menos de 1,5V, os vermelhos com 1,7V, os amarelos com 1,7V ou 2.0V, os verdes entre 2.0V e 3.0V, enquanto os leds azuis, violeta e ultravioleta geralmente precisam de mais de 3V, já a corrente consumida normalmente é de 20mA, mas dependendo o tipo de LED esse valor pode variar.O tempo de vida útil dele é de aproximadamente 100mil horas.

Figura 6.9 – Imagem de alguns LED's existentes no mercado.

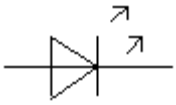

Figura 6.10 – Imagem do desenho esquemático de um LED.

Fonte: Wikipédia, HowStuffWorks.

6.4 Relé

Os relés são componentes eletromecânicos capazes de controlar circuitos externos de grandes correntes a partir de pequenas correntes ou tensões, ou seja, podemos acionar um relé com uma pilha e controlar um dispositivo (Motor, lâmpada e etc...) que precisa ser ligado em tensões mais altas (110 ou 220 volts, por exemplo).

O funcionamento dos relés é bem simples, quando uma corrente circula por uma bobina que se localiza dentro do relé, esta cria um campo magnético que atrai um ou vários contatos fechando ou abrindo circuitos que estiver ligado no relé. Quando tirar a corrente da bobina o campo magnético acaba, fazendo com que os contatos voltem para a posição original.

Os relés podem ter diversas configurações quanto aos seus contatos: podem ter contatos NA (normalmente aberto), NF (normalmente fechado) ou ambos, neste caso com um contato comum ou central (C). Os contatos NA são os que estão abertos enquanto a bobina não está energizada e que fecham, quando a bobina recebe corrente. Os NF se abrem quando a bobina recebe corrente, ao contrário dos NA. O contato C é o comum, ou seja, é ele que se move quando a corrente passa ou deixa de passar na bobina.

A principal vantagem dos Relés é que o circuito de carga está completamente isolado do circuito de controle, assim, podendo inclusive trabalhar com tensões diferentes entre o controle e a carga.

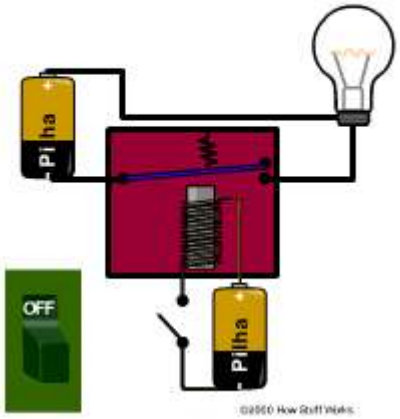

Figura 6.11 – Figura mostrando o estado de um relé desligado.

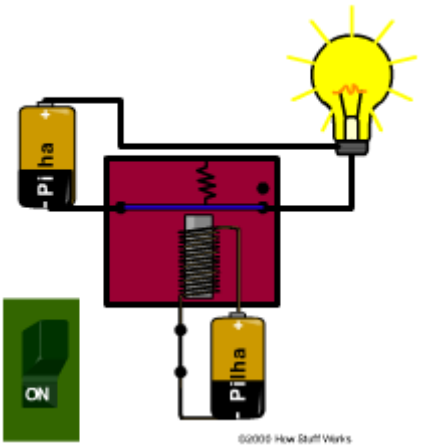

Figura 6.12 - Figura mostrando o estado de um relé ligado.

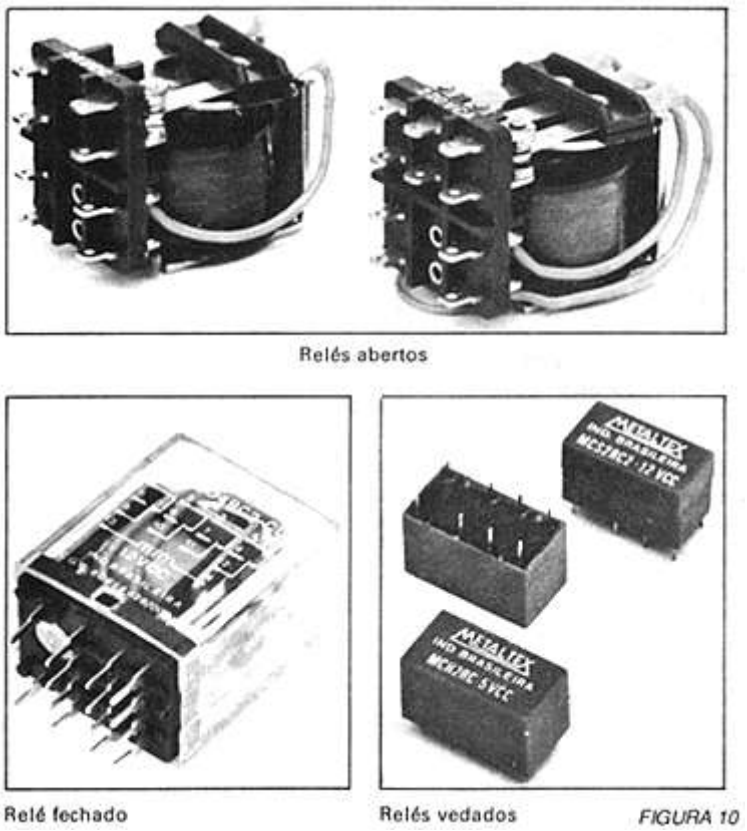

Figura 6.13 – Imagem de alguns relés existentes no mercado.

Fonte: Wikipédia, Angelfire e HowStuffWorks.

6.5 Resistor

Os resistores são componentes responsáveis por transformar energias elétricas em energia térmica através do efeito Joule. Ele é fabricado com matérias resistivo, como carbono, por exemplo. Um resistor tem umas faixas coloridas que podem mostrar os valores da resistividade e a sua tolerância desse resistor, alguns resistores são longos e finos, com o material resistivo colocado ao centro, e um terminal de metal ligado em cada extremidade. Este tipo de encapsulamento é chamado de encapsulamento axial. Resistores usados em computadores e outros dispositivos são tipicamente muito menores, freqüentemente são utilizadas tecnologia de montagem superficial (Surface-mount technology), ou SMT, esse tipo de resistor não possui terminais, já os resistores de maiores potências são produzidos mais robustos para dissipar calor de maneira mais eficiente, mas eles seguem basicamente a mesma estrutura.

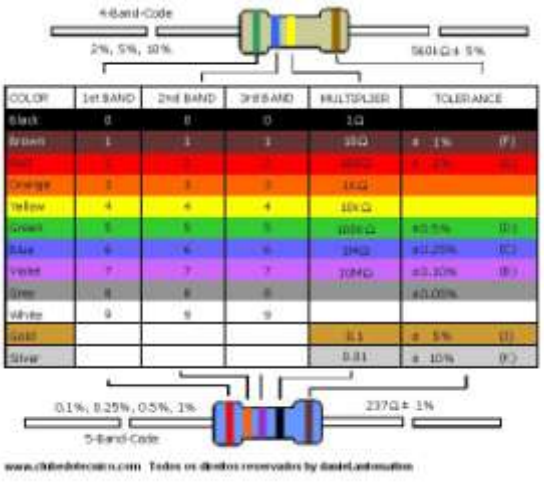

Figura 6.14 – Leitura das faixas do resistor.

Figura 6.15 – Imagem de um resistor SMD (cima) e um resistor de carbono (baixo).

Fonte: Wikipédia, HowStuffWorks.

6.6 Circuitos Integrados

Um circuito integrado, também conhecido por chip, é um dispositivo microeletrônico que consiste de muitos transistores e outros componentes interligados capazes de desempenhar muitas funções. Suas dimensões são extremamente reduzidas, os componentes são formados em pastilhas de material semicondutor.

A importância da integração está no baixo custo e alto desempenho, além do tamanho reduzido dos circuitos aliado à alta confiabilidade e estabilidade de funcionamento. Uma vez que os componentes são formados ao invés de montados, a resistência mecânica destes permitiu montagens cada vez mais robustas a choques e impactos mecânicos, permitindo a concepção de portabilidade dos dispositivos eletrônicos.

No circuito integrado completo ficam presentes os transístores, condutores de interligação, componentes de polarização, e as camadas e regiões isolantes ou condutoras obedecendo ao seu projeto de arquitetura.

No processo de formação do chip, é fundamental que todos os componentes sejam implantados nas regiões apropriadas da pastilha. É necessário que a isolação seja perfeita, quando for o caso. Isto é obtida por um processo chamado difusão, que se dá entre os componentes formados e as camadas com o material dopado com fósforo, e separadas por um material dopado com boro, e assim por diante.

Após sucessivas interconexões, por boro e fósforo, os componentes formados ainda são interconectados externamente por uma camada extremamente fina de alumínio, depositada sobre a superfície e isolada por uma camada de dióxido de silício.

Figura 6.16 – Imagem de um Circuito Integrado.

Fonte: Wikipédia, Guia do Hardware.

6.7 Transistor

O primeiro projeto surgiu em 16 de Dezembro de 47, onde era usado um pequeno bloco de germânio (que na época era junto com o silício o semicondutor mais pesquisado) e três filamentos de ouro. Um filamento era o pólo positivo, o outro o pólo negativo, enquanto o terceiro tinha a função de controle. Tendo apenas uma carga elétrica no pólo positivo, nada acontecia, o germânio atuava como um isolante, bloqueando a corrente. Porém, quando certa tensão elétrica era aplicada usando o filamento de controle, um fenômeno acontecia e a carga elétrica passava a fluir para o pólo negativo. Haviam criado um dispositivo que substituía a válvula, sem possuir partes móveis, ao mesmo tempo, muito mais rápidos. Este primeiro transistor era relativamente grande, mas não demorou muito para que este modelo inicial fosse aperfeiçoado.

Durante a década de 50, o transistor foi gradualmente dominando a indústria, substituindo rapidamente as problemáticas válvulas. Os modelos foram diminuindo de tamanho, caindo de preço e tornando-se mais rápidos. Alguns transistores da época podiam operar a até 100 MHz. Claro que esta era a freqüência que podia ser alcançada por um transistor sozinho, nos computadores da época, a freqüência de operação era muito menor, já que em cada ciclo de processamento o sinal precisa passar por vários transistores.

Mas, o grande salto foi à substituição do germânio pelo silício. Isto permitiu miniaturizar ainda mais os transistores e baixar seu custo de produção. Os primeiros transistores de junção comerciais foram produzidos partir de 1960 pela Crystalonics. A idéia do uso do silício para construir transistores é que adicionando certas substâncias em pequenas quantidades é possível alterar as propriedades elétricas do silício. As primeiras experiências usavam fósforo e boro, que transformavam o silício em condutor por cargas negativas ou condutoras por cargas positivas, dependendo de qual dos dois materiais fosse usado. Estas substâncias adicionadas ao silício são chamadas de impurezas, e o silício "contaminado" por elas é chamado de silício dopado.

O funcionamento e um transístor são bastante simples, quase elementar. É como naquele velho ditado "as melhores invenções são as mais simples". As válvulas eram muito mais complexas que os transistores e mesmo assim foram rapidamente substituídas por eles. Um transistor é composto basicamente de três filamentos, chamados de base, emissor e coletor. O emissor é o pólo positivo, o coletor o pólo negativo, enquanto a base é quem controla o estado do transistor, que como vimos, pode estar ligado ou desligado. Quando o transistor está desligado, não existe carga elétrica na base, por isso, não existe corrente elétrica entre o emissor e o coletor (temos então um bit 0). Quanto é aplicado certa tensão na base, o circuito é fechado e é estabelecida a corrente entre o emissor e o receptor (um bit 1).

Método de fabricação do transistor

Os materiais utilizados atualmente na fabricação do transistor são o Silício (Si), o Gálio (Ga) e alguns óxidos. Na natureza, o silício é um material isolante elétrico, devido à conformação das ligações eletrônicas de seus átomos, gerando uma rede eletrônica altamente estável.

O silício é purificado e passa por um processo que forma uma estrutura cristalina em seus átomos. O material é cortado em finos discos, que a seguir vão para um processo chamado de dopagem, onde são introduzidas quantidades rigorosamente controladas de materiais selecionados (conhecidos como impurezas) que transformam a estrutura eletrônica, introduzindo-se entre as ligações dos átomos de silício, recebe ou doa elétrons dos átomos, gerando o silício P ou N, conforme ele seja positivo (tenha falta de elétrons) ou negativo (tenha excesso de elétrons). Se a impureza tiver um elétron a mais, um elétron fica sobrando na estrutura cristalina. Se tiver um elétron a

menos, fica faltando um elétron, o que produz uma lacuna (que funciona como se fosse um buraco móvel na estrutura cristalina). Como resultado, temos ao fim desse processo um semicondutor.

O transistor é montado juntando uma camada P, uma N e outra P, criando-se um transistor do tipo PNP. O transistor do tipo NPN é obtido de modo similar. A camada do centro é denominada base, e as outras duas são o emissor e o coletor. No símbolo do componente, o emissor é indicado por uma seta, que aponta para dentro do transistor se o componente for PNP, ou para fora se for NPN.

Figura 6.17 – Imagem de alguns transistores existentes no mercado.

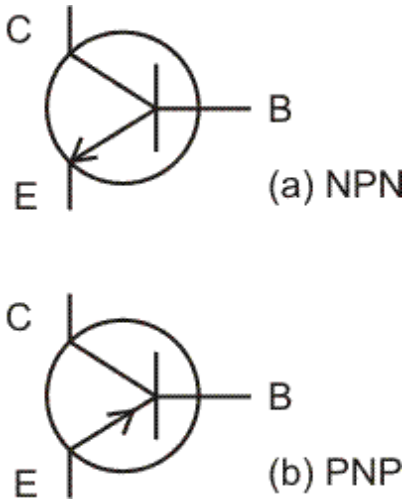

Figura 6.18 - Imagem do símbolo de um transistor do tipo PNP e outro do NPN.

Fonte: Wikipédia, Guia do Hardware.

6.8 Indutor

Indutor é um dispositivo elétrico passivo que armazena energia na forma de campo magnético, normalmente combinando o efeito de vários loops da corrente elétrica.

A capacidade de um indutor é controlada por quatro fatores:

- O número de espiras (mais espiras significam maior indutância)

- O material em que as bobinas são enroladas (o núcleo)
- A área da seção transversal da bobina (mais área significa maior indutância)
- O comprimento da bobina (uma bobina curta significa espiras mais estreitas - ou sobreposição - que significa maior indutância)

Um núcleo de ferro oferece ao indutor muito mais indutância do que o ar ou do que qualquer outro material ofereceria.

Construção:

Um indutor é geralmente construído como uma bobina de material condutor, por exemplo, fio de cobre. Um núcleo de material ferromagnético aumenta a indutância concentrando as linhas de força de campo magnético que fluem pelo interior das espiras.

Indutores podem ser construídos em circuitos integrados utilizando o mesmo processo que é usado em chips de computador. Nesses casos, normalmente o alumínio é utilizado como material condutor. Porém, é raro a construção de indutores em CI's: eles são volumosos em uma pequena escala, e praticamente restritos, sendo muito mais comum o uso de um circuito que utiliza um capacitor comportando-se como se fosse um indutor.

Pequenos indutores usados para freqüências muito altas são algumas vezes feitos com um fio passando através de um cilindro de ferrite.

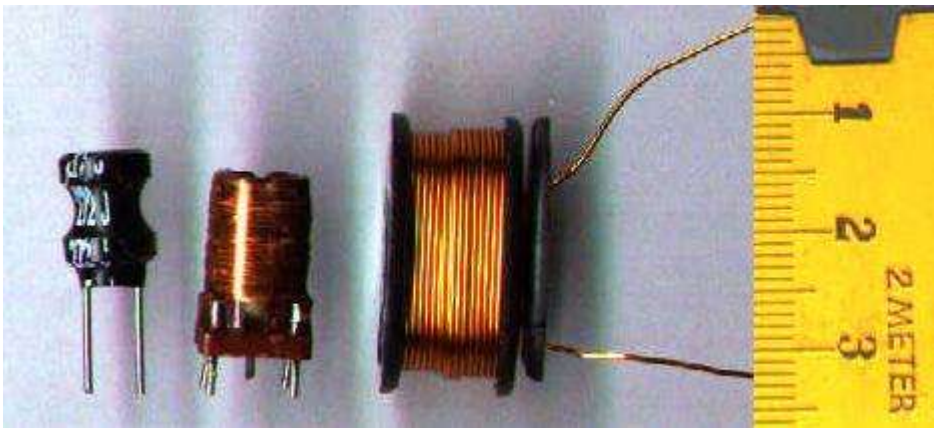

Figura 6.19 – Tipos de indutores.

Fonte: Wikipédia, HowStuffWorks.

6.9 Capacitor

Capacitor é um componente que armazena energia num campo elétrico, acumulando um desequilíbrio interno de carga elétrica.

Como a pilha, o capacitor possui dois terminais. Dentro do capacitor, os terminais conectam-se a duas placas metálicas separadas por um dielétrico. O dielétrico pode ser ar, papel, plástico ou qualquer outro material que não conduza eletricidade e impeça que as placas se toquem.

Em um circuito eletrônico, um capacitor é indicado da seguinte forma:

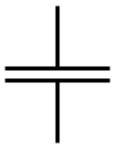

Figura 6.20 – Representação de um capacitor.

Capacitores são comumente usados em fontes de energia onde elas suavizam a saída de uma onda retificada completa ou meia onda.

Por passarem sinais de Corrente Alternada e bloquearem Corrente Contínua, capacitores são freqüentemente usados para separar circuitos Corrente alternados de corrente continua. Este método é conhecido como acoplamento AC.

Capacitores também são usados na correção de fator de potência. Tais capacitores freqüentemente vêm como três capacitores conectados como uma carga trifásica. Geralmente, os valores desses capacitores não são dados pela sua capacitância, mas pela sua potência reativa em var.

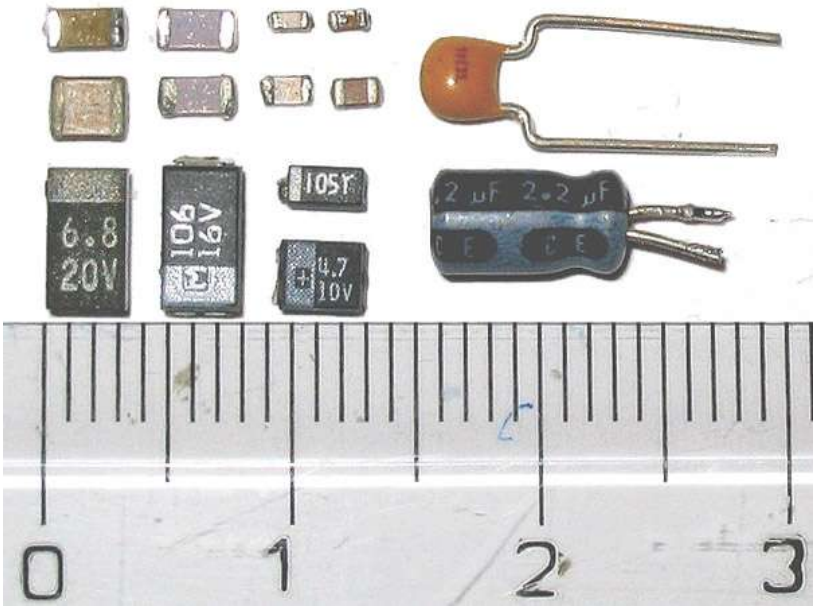

Figura 6.21 – Exemplos de capacitores.

Fonte: Wikipédia, HowStuffWorks.

6.10 Servo-Motor

Servo-motores são pequenas máquinas que apresentam movimento proporcional ao comando enviado.

Muitas Vezes esses motores permitem o posicionamento angular dentro de uma faixa de 180º, dependendo do ciclo de trabalho de um sinal PWM aplicado em sua entrada. Este sinal deve ter um período total de 20ms, com um ciclo ativo de 5 a 10% (0 a 180º respectivamente).

Como entrada, o servo geralmente possui três fios sendo o vermelho ligado à alimentação positiva, o preto ligado a terra e o último ligado ao sinal PWM. A tensão de alimentação do servo deve ser respeitada e observada na hora da aquisição.

Figura 6.22 – Exemplos de motor-servo.

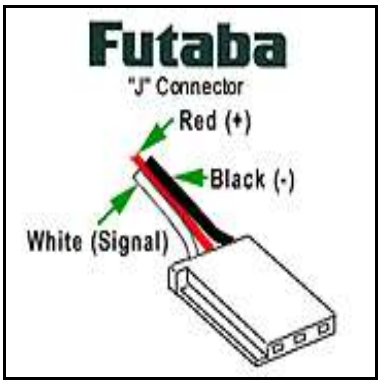

Figura 6.23 Fios de entrada de um motor-servo.

Fonte:

http://www.futaba-rc.com/servos

http://www.cs.uiowa.edu/~jones/step/