PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

CURSO DE ENGENHARIA DA COMPUTAÇÃO

ESTACIONAMENTO AUTOMATIZADO

CURITIBA

2010

ALEX DOS SANTOS XAVIER

JAZIEL DO CARMO DA SILVA

RAFHAEL W. DE SOUZA LEMES

ESTACIONAMENTO AUTOMATIZADO

Projeto integrado apresentado às disciplinas de Resolução de Problemas em Engenharia e Física III como parte do processo avaliativo do 3º período do curso de Engenharia da Computação da Pontifícia Universidade Católica do Paraná - Campus Curitiba.

Professores: Afonso Ferreira Miguel e Gil Marcos Jess

CURITIBA

2010

RESUMO

O projeto Estacionamento Automatizado, desenvolvido durante o terceiro período do curso de Engenharia da Computação da Pontifícia Universidade Católica do Paraná, foi desenvolvido com base no tema proposto para o período, o qual era usar os conceitos de movimento.

O projeto consiste em um estacionamento subterrâneo projetado para otimizar espaço. O sistema não necessita de manobristas já que o elevador pega o carro na entrada e o estaciona automaticamente ou busca-o conforme solicitado, sem a interferência humana, que ocorre somente na hora do pagamento.

Palavras-chave: Estacionamento, projeto, subterrâneo, otimizar, movimento.

ABSTRACT

The project Automated Park, developed over third semester of course of Engineer of Computer of Pontifícia Universidade Católica do Paraná, was built with base of suggested theme of semester, which was use concepts of movement .

The project is a underground parking projected for to optimize the places. The system doesn´t need valets, the elevator picks the car in the enter, and automatic park it, and retire the car as the program order, without humans interference, that happens only when the user will be pay for the service.

Key Words: Parking, project, underground, optimize, movement

SUMÁRIO

ÍNDICE DE FIGURAS

1. INTRODUÇÃO

O projeto estacionamento automatizado pretende resolver um dos problemas que mais afetam as grandes metrópoles, a falta de vaga para carros, totalmente automatizado o usuário não perderia mais tempo buscando uma vaga livre e a sua organização permitirá que em um curto espaço físico sejam criadas várias vagas, sendo necessária a intervenção humana apenas para o pagamento.

O sistema não necessita de manobristas, eliminando erros humanos como: colisões entre veículos e furtos em seu interior, é um sistema subterrâneo que otimiza o espaço acima do solo, possibilitando a sua instalação nas proximidades dos aeroportos ou em lugares sem espaço.

2. OBJETIVO

2.1 Geral

O usuário estacionará o seu carro no elevador e em um terminal do lado de fora ele digitara a placa do carro e uma senha pessoal para a sua identificação posterior;

O elevador conduzirá o carro para uma vaga livre, que “o estacionará” automaticamente;

Quando o usuário quiser a retirada do carro, no terminal ele digitará a placa do seu carro e sua senha pessoal;

O elevador irá recolher o carro do usuário, que será entregue na porta de saída;

2.2 Específico

1. Trabalhar com motores de passo para controle do eixo e elevador; e aprender a controlá-los.
2. Desenvolver placas e circuitos para comunicação com os motores, colocando em prática o aprendizado da parte teórica;
3. Desenvolver software para comunicação com as placas e motores e gerenciamento do estacionamento.
4. Entregar cd de dados contendo vídeos, fotos do projeto, códigos-fonte e documentação.

3. MATERIAIS UTILIZADOS

3.1 Hardware

- 4 Resistores 1kΏ;
- 4 Resistores 3k3Ώ;
- 4 Resistores de 330Ώ;
- 12 Resistores 4k7Ώ;
- 2 Placas de fenolite;
- 8 Transistores TIP 125;
- 8 Transistores TIP 122;
- 16 TransistoresBC548;
- 1Conector ATX;
- 1Conector 10 Pin;
- 1 Cabo Flat;
- 2 Conectores macho 10 Pin;
- 2 Conectores fêmea 10 Pin.
- Estanho;
- Sugador de solda;
- Arduino.

3.2 Software

- Microsoft Windows 7;
- Microsoft Visual Studio 2008;
- Software Eagle;
- Arduino Alpha.

3.3 Estrutura

- Madeira MDF;
- Eixos de duas impressoras;
- 2 motores de passo;
- Barra de metal;
- Parafusos;
- Alumínio;
- Tinta Esmalte Sintético branca, cinza.

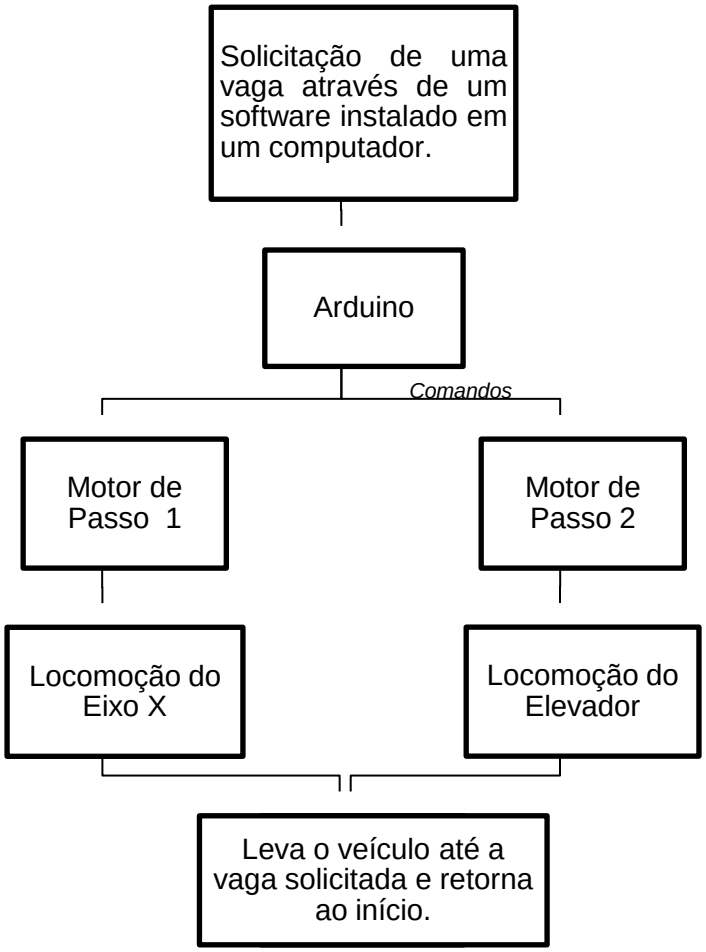
3.4 Equipamentos

- Notebook HP Pavilion;
- Uma fonte ATX;
- Protoboard;
- Arduino.

4. DESCRIÇÃO DO PROJETO

4.1 Descrição Geral

O funcionamento do projeto se dá pelo seguinte diagrama:



4.2 Descrição Detalhada

Através do software desenvolvido em C++, será solicitada uma vaga no estacionamento e será digitada uma senha, esses comandos serão enviados pela porta USB ao Arduino que é o responsável pela comunicação com os motores. O motor do eixo y busca o automóvel na primeira vaga do estacionamento e logo em seguida o motor do eixo x percorre o espaço solicitado até chegar à vaga . Quando o usuário solicita a retirada do automóvel o software realiza o mesmo processo descrito anteriormente e calcula o valor a ser pago na saída.

4.3 Circuitos Elétricos

4.3.1 Desenho da Placa Principal

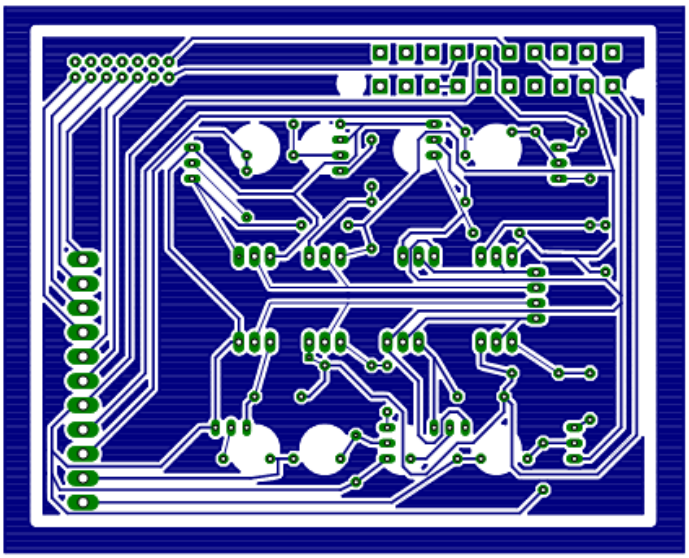


Figura 1- Circuito impresso da placa principal.

4.3.2 Desenho da Placa Secundária

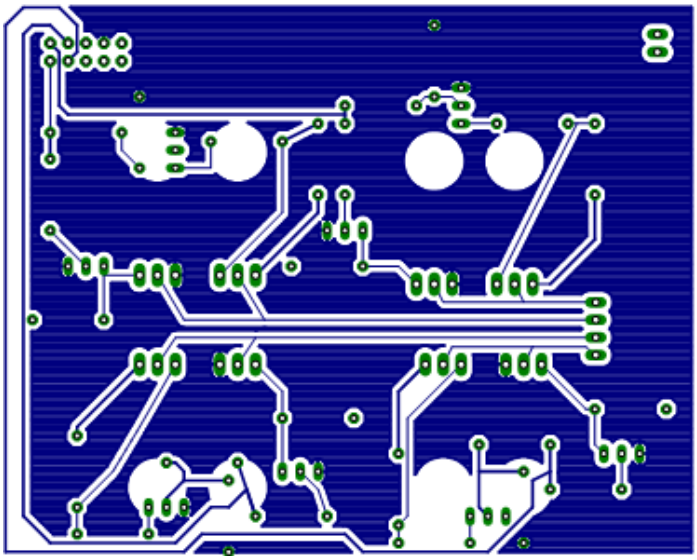


Figura 2- Circuito impresso da placa secundária.

4.3.2 Esquemático do Driver SM-DRV

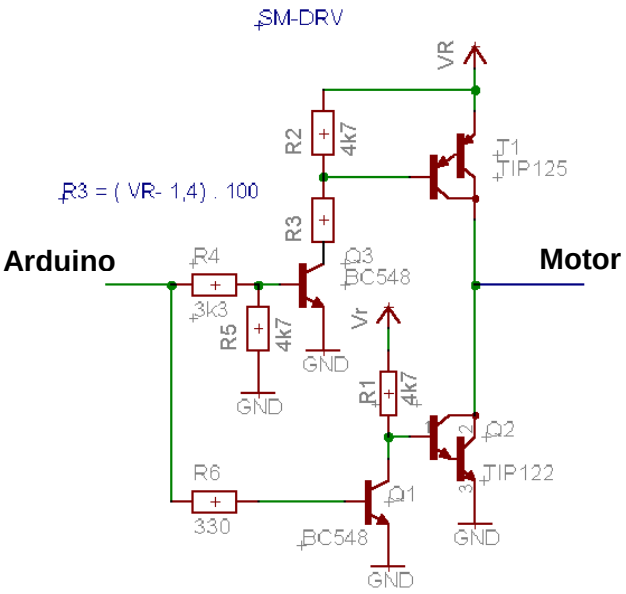


Figura 3 – Diagrama do esquemático do driver SM-DRV.

4.4 Software

4.4.1 Classes C++

4.4.1.1 Alocador.h

```
#pragma once
#include "Carro.h"
#include "Vaga.h"

class Alocador
{
private:
	Vaga aloca_vaga[3][3];
	char posicao_vaga[2];
	char comparacao_senhas[6];
	Vaga caixa_eletronico;
public:
	void alocar_vaga();//metodo que tem como função alocar uma vaga para o carro cada vez que e incializada
	void desalocar_vaga();//desaloca um carro de uma vaga, levando em consideração o numero da vaga e a senha de acesso
	Alocador(void);
	~Alocador(void);
	void acessa_variavel();//metodo que verifica se o carro pode ser alocado
```

```
      void pagamento(int i, int j);//metodo que contabiliza a hora de
entrada e saida e da o valor de pagamento correspondente

};
```

4.4.1.2 Carro.h

```
#pragma once

#include "Form2.h"
class Carro
{
private:
      char senha_carro[6];
      int k;//inicalizado no construtor com -1, o atributo serve para
fazer o casamento da comparação de senhas do alocador com a senha da
classe carro

public:
      int pega_medida_carro();//metodo que verifica se o carro pode
ser alocado ou não
    char senha_usuario();//faz o usuário escrever sua senha
    char retorna_senha_usuario();//somente retorna a senha do usuário
      int k_recebe_menos1();
      Carro(void);
      ~Carro(void);
};
```

4.4.1.3 Vaga.h

```
#pragma once
#include "Carro.h"
#include <time.h>
#include <stdio.h>
#include <iostream>
using namespace std;

class Vaga
{
private:
      bool estado_vaga;//identifica se a vaga esta cheia ou vazia,
vaga vazia = false, vaga cheia = true
      int tempo_vaga[2];//contem a quantidade de tempo que um carro
permanece em uma vaga
    Carro recebe_senha_carro;
public:
      int muda_valor_k();
      void muda_estado_vaga();//muda o estado da vaga
      int verifica_tempo_vaga();// se o atributo estado vaga esta em
true,o metodo começa a contar o tempo em que o carro esta na vaga
      bool verifica_vaga();//metodo que verifica se a vaga esta cheia
ou vazia
      char acessa_atributos();
      char retorna_senha();
      int retorna_tempo_hora();//retorna o tempo da hora do sistema
operacional para
      int retorna_tempo_minuto();
```

```
	Carro retorna_recebe_senhas_carro();
	Vaga(void);
	~Vaga(void);
};
```

4.4.1.4 Form1.h

```
#include "Alocador.h"

extern Alocador sistema;

#pragma once


namespace Park {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	/// <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();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~Form1()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::Label^  label1;
	private: System::Windows::Forms::Button^  button2;
```

```
	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)
		{
			this->button1 = (gcnew
System::Windows::Forms::Button());
			this->label1 = (gcnew
System::Windows::Forms::Label());
			this->button2 = (gcnew
System::Windows::Forms::Button());
			this->SuspendLayout();
			//
			// button1
			//
			this->button1->Location = System::Drawing::Point(80,
92);
			this->button1->Name = L"button1";
			this->button1->Size = System::Drawing::Size(119, 28);
			this->button1->TabIndex = 0;
			this->button1->Text = L"Guardar Carro";
			this->button1->UseVisualStyleBackColor = true;
			this->button1->Click += gcnew
System::EventHandler(this, &Form1::button1_Click);
			//
			// label1
			//
			this->label1->AutoSize = true;
			this->label1->Location = System::Drawing::Point(77,
29);
			this->label1->Name = L"label1";
			this->label1->Size = System::Drawing::Size(139, 13);
			this->label1->TabIndex = 1;
			this->label1->Text = L"Estacionamento Automático";
			//
			// button2
			//
			this->button2->Location = System::Drawing::Point(81,
138);
			this->button2->Name = L"button2";
			this->button2->Size = System::Drawing::Size(118, 28);
			this->button2->TabIndex = 2;
			this->button2->Text = L"Retirar Carro";
			this->button2->UseVisualStyleBackColor = true;
			this->button2->Click += gcnew
System::EventHandler(this, &Form1::button2_Click);
			//
			// Form1
			//
```

```
                this->AutoScaleDimensions = System::Drawing::SizeF(6,
13);
                this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(284, 264);
                this->ControlBox = false;
                this->Controls->Add(this->button2);
                this->Controls->Add(this->label1);
                this->Controls->Add(this->button1);
                this->Name = L"Form1";
                this->Text = L"Estacionamento Automático";
                this->Load += gcnew System::EventHandler(this,
&Form1::Form1_Load);
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion
      private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
                        sistema.alocar_vaga();
                 }
      private: System::Void Form1_Load(System::Object^  sender,
System::EventArgs^  e) {
                 }
      private: System::Void button2_Click(System::Object^  sender,
System::EventArgs^  e) {
                        sistema.desalocar_vaga();
                 }
      };
}
//Nota:Form1 é a interface grafica inicial do software, nela contem os
dois metodos que fazem toda a parte
//de alocação e desalocação das vagas
//alocar e desalocar vagas, são metodos criados na classe alocador.
```

4.4.1.5 Form2.h

```
#pragma once

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


namespace Park {

      /// <summary>
      /// Summary for Form2
      ///
      /// 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 Form2 : public System::Windows::Forms::Form
	{
	public:
		Form2(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}


	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~Form2()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::TextBox^  textBox1;
	protected:
	private: System::Windows::Forms::Label^  label1;
	private: System::Windows::Forms::Button^  button1;

	protected:



	protected:

	protected:


	protected:

	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>
		System::ComponentModel::Container ^components;
		char* senha;
	public:
		char* getSenha();
#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)
		{
			this->textBox1 = (gcnew
System::Windows::Forms::TextBox());
```

```
                this->label1 = (gcnew
System::Windows::Forms::Label());
                this->button1 = (gcnew
System::Windows::Forms::Button());
                this->SuspendLayout();
                //
                // textBox1
                //
                this->textBox1->Location = System::Drawing::Point(56,
88);
                this->textBox1->Name = L"textBox1";
                this->textBox1->PasswordChar = '*';
                this->textBox1->Size = System::Drawing::Size(177,
20);
                this->textBox1->TabIndex = 0;
                this->textBox1->UseSystemPasswordChar = true;
                this->textBox1->TextChanged += gcnew
System::EventHandler(this, &Form2::textBox1_TextChanged);
                //
                // label1
                //
                this->label1->AutoSize = true;
                this->label1->Location = System::Drawing::Point(74,
62);
                this->label1->Name = L"label1";
                this->label1->Size = System::Drawing::Size(140, 13);
                this->label1->TabIndex = 1;
                this->label1->Text = L"Digite a senha, 6 caracteres";
                this->label1->Click += gcnew
System::EventHandler(this, &Form2::label1_Click);
                //
                // button1
                //
                this->button1->DialogResult =
System::Windows::Forms::DialogResult::OK;
                this->button1->Location = System::Drawing::Point(100,
114);
                this->button1->Name = L"button1";
                this->button1->Size = System::Drawing::Size(81, 28);
                this->button1->TabIndex = 2;
                this->button1->Text = L"OK";
                this->button1->UseVisualStyleBackColor = true;
                this->button1->Click += gcnew
System::EventHandler(this, &Form2::button1_Click);
                //
                // Form2
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(6,
13);
                this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(282, 165);
                this->ControlBox = false;
                this->Controls->Add(this->button1);
                this->Controls->Add(this->label1);
                this->Controls->Add(this->textBox1);
                this->Name = L"Form2";
                this->Text = L"Form2";
                this->Load += gcnew System::EventHandler(this,
&Form2::Form2_Load);
                this->ResumeLayout(false);
```

```
                this->PerformLayout();

            }
#pragma endregion
      private: System::Void Form2_Load(System::Object^  sender,
System::EventArgs^  e) {
                  }
      private: System::Void textBox1_TextChanged(System::Object^
sender, System::EventArgs^  e) {
                  }
      private: System::Void label1_Click(System::Object^  sender,
System::EventArgs^  e) {
                  }
      private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {

                        senha = new char[6];
                        if(textBox1->Text->Length == 0)
                        {
                              MessageBox::Show("valor naum pode ser
nulo");
                              senha[0] = 'a';
                              return;
                        }

                        for(int i = 0; i < textBox1->Text->Length;
i++)
                        {
                        senha[i] = (char)textBox1->Text[i];
                        if(textBox1->Text->Length < 6)
                        {
                             MessageBox::Show("Poucos Digitos !");
                           senha[i] = 'a';
                           return;
                        }
                        else if(textBox1->Text->Length > 6)
                        {
                              MessageBox::Show("Muitos digitos!");
                              senha[i] = 'a';
                              return;
                        }
                        }
                  }
      };
}
```

4.4.1.6 Form3.h

```
#pragma once

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


namespace Park {
```

```
	/// <summary>
	/// Summary for Form3
	///
	/// 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 Form3 : public System::Windows::Forms::Form
	{
	public:
		Form3(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~Form3()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Label^  label1;
	private: System::Windows::Forms::TextBox^  textBox1;
	private: System::Windows::Forms::Button^  button1;
	protected:

	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>
		System::ComponentModel::Container ^components;
	private:
		char *posicao;
	public:
		char *get_posicao();
#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)
		{
			this->label1 = (gcnew
System::Windows::Forms::Label());
			this->textBox1 = (gcnew
System::Windows::Forms::TextBox());
```

```
                this->button1 = (gcnew
System::Windows::Forms::Button());
                this->SuspendLayout();
                //
                // label1
                //
                this->label1->AutoSize = true;
                this->label1->Location = System::Drawing::Point(42,
52);
                this->label1->Name = L"label1";
                this->label1->Size = System::Drawing::Size(211, 13);
                this->label1->TabIndex = 0;
                this->label1->Text = L"Informe a vaga em que o carro
se encontra";
                this->label1->Click += gcnew
System::EventHandler(this, &Form3::label1_Click);
                //
                // textBox1
                //
                this->textBox1->Location = System::Drawing::Point(67,
79);
                this->textBox1->Name = L"textBox1";
                this->textBox1->Size = System::Drawing::Size(152,
20);
                this->textBox1->TabIndex = 1;
                this->textBox1->TextChanged += gcnew
System::EventHandler(this, &Form3::textBox1_TextChanged);
                //
                // button1
                //
                this->button1->DialogResult =
System::Windows::Forms::DialogResult::OK;
                this->button1->Location = System::Drawing::Point(103,
105);
                this->button1->Name = L"button1";
                this->button1->Size = System::Drawing::Size(78, 30);
                this->button1->TabIndex = 2;
                this->button1->Text = L"OK";
                this->button1->UseVisualStyleBackColor = true;
                this->button1->Click += gcnew
System::EventHandler(this, &Form3::button1_Click);
                //
                // Form3
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(6,
13);
                this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(284, 264);
                this->ControlBox = false;
                this->Controls->Add(this->button1);
                this->Controls->Add(this->textBox1);
                this->Controls->Add(this->label1);
                this->Name = L"Form3";
                this->Text = L"Form3";
                this->Load += gcnew System::EventHandler(this,
&Form3::Form3_Load);
                this->ResumeLayout(false);
                this->PerformLayout();

            }
```

```
#pragma endregion
      private: System::Void label1_Click(System::Object^  sender,
System::EventArgs^  e) {
                  }
      private: System::Void Form3_Load(System::Object^  sender,
System::EventArgs^  e) {
                  }
      private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
                  posicao = new char[2];
                        if(textBox1->Text->Length > 2)
                        {
                              MessageBox::Show("vaga não existente");
                        }

                        for(int i = 0; i < textBox1->Text->Length;
i++)
                        {
                              posicao[i] = (char)textBox1->Text[i];
                        }
                  }
      private: System::Void textBox1_TextChanged(System::Object^
sender, System::EventArgs^  e) {
                  }
      };
}
```

4.4.1.7 Form4.h

```
#pragma once

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


namespace Park {

      /// <summary>
      /// Summary for Form4
      ///
      /// 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 Form4 : public System::Windows::Forms::Form
      {
      public:
            Form4(void)
            {
```

```
                InitializeComponent();
                //
                //TODO: Add the constructor code here
                //
            }

        protected:
            /// <summary>
            /// Clean up any resources being used.
            /// </summary>
            ~Form4()
            {
                if (components)
                {
                    delete components;
                }
            }
        private: System::Windows::Forms::ListBox^  listBox1;
        private: System::Windows::Forms::Label^  label1;
        private: System::Windows::Forms::Button^  button1;
        protected:

        private:
            /// <summary>
            /// Required designer variable.
            /// </summary>
            System::ComponentModel::Container ^components;
    private:
            char* posicao;
        public:
            char* pega_posicao();

#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)
            {
                this->listBox1 = (gcnew
System::Windows::Forms::ListBox());
                this->label1 = (gcnew
System::Windows::Forms::Label());
                this->button1 = (gcnew
System::Windows::Forms::Button());
                this->SuspendLayout();
                //
                // listBox1
                //
                this->listBox1->FormattingEnabled = true;
                this->listBox1->Items->AddRange(gcnew cli::array<
System::Object^  >(9) {L"00", L"01", L"02", L"10", L"11", L"12",
L"20",
                    L"21", L"22"});
                this->listBox1->Location =
System::Drawing::Point(157, 64);
                this->listBox1->Name = L"listBox1";
                this->listBox1->Size = System::Drawing::Size(36, 30);
                this->listBox1->TabIndex = 0;
                this->listBox1->SelectedIndexChanged += gcnew
System::EventHandler(this, &Form4::listBox1_SelectedIndexChanged);
```

```
                //
                // label1
                //
                this->label1->AutoSize = true;
                this->label1->Location = System::Drawing::Point(34,
28);
                this->label1->Name = L"label1";
                this->label1->Size = System::Drawing::Size(294, 13);
                this->label1->TabIndex = 1;
                this->label1->Text = L"selecione a posição em que o
carro se encontra e aperte OK";
                this->label1->Click += gcnew
System::EventHandler(this, &Form4::label1_Click);
                //
                // button1
                //
                this->button1->DialogResult =
System::Windows::Forms::DialogResult::OK;
                this->button1->Location = System::Drawing::Point(145,
100);
                this->button1->Name = L"button1";
                this->button1->Size = System::Drawing::Size(59, 32);
                this->button1->TabIndex = 2;
                this->button1->Text = L"OK";
                this->button1->UseVisualStyleBackColor = true;
                this->button1->Click += gcnew
System::EventHandler(this, &Form4::button1_Click);
                //
                // Form4
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(6,
13);
                this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(358, 164);
                this->ControlBox = false;
                this->Controls->Add(this->button1);
                this->Controls->Add(this->label1);
                this->Controls->Add(this->listBox1);
                this->Name = L"Form4";
                this->Text = L"Estacionamento Automático";
                this->Load += gcnew System::EventHandler(this,
&Form4::Form4_Load);
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion
      private: System::Void label1_Click(System::Object^  sender,
System::EventArgs^  e) {
                 }
      private: System::Void Form4_Load(System::Object^  sender,
System::EventArgs^  e) {
                 }
      private: System::Void
listBox1_SelectedIndexChanged(System::Object^  sender,
System::EventArgs^  e) {
                 }
      private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
                  posicao = new char[2];
```

```
                    for(int i = 0; i < listBox1->Text->Length; i++){
                          //if(listBox1->Text->Length == 0)
                          //{
                              //  MessageBox::Show("Vaga não
informada!");
                          //}
                    posicao[i] = (char)listBox1->Text[i];
                    }
                }
      };
}
```

4.4.1.8 Motor.h

```
#ifndef MOTOR
#define MOTOR

class Motor
{
public:
      Motor();
      bool abrirPorta();//Configura o acesso a porta, retorna false se
não conseguiu abrir.

};

#endif
```

4.4.1.9 Alocador.cpp

```
#include "StdAfx.h"
#include "Form4.h"
#include "Alocador.h"
#include "Vaga.h"
#include "Carro.h"
#include <iostream>
using namespace std;
using namespace System::Windows::Forms;
using namespace Park;

Alocador sistema;

Alocador::Alocador(void)
{
}

Alocador::~Alocador(void)
{
}
void Alocador::alocar_vaga()
{
      int i,j;
      //o metodo alocador percorre uma matriz 3x3 verificando se o
valor boleano de cada aloca_vaga[i][j]
      //e falso, caso isso aconteça, e chamado um metodo que verifica
a condição de entrada do carro - acessa_atributos
```

```
	//se tudo ocorreu como o esperado, o metodo libera uma senha e o estado de aloca_vaga[i][j] passa de false para true
	//e retirado o tempo de sistema, que fica sendo como a hora inicial que o carro foi alocado
	for(i = 0; i < 3; i++){
		for(j = 0; j < 3; j++)
		{
			if(aloca_vaga[i][j].verifica_vaga() == false){
				if(aloca_vaga[i][j].acessa_atributos() == 'a')
				{
					return;
				}
				aloca_vaga[i][j].muda_estado_vaga();
				aloca_vaga[i][j].verifica_tempo_vaga();
				MessageBox::Show("o carro esta alocado na vaga " + i.ToString() + j.ToString(),"Estacionamento Automatico");
					return;
			}
		}
	}
	MessageBox::Show("O estacionamento está lotado!","Estacionamento Automatico");
}
void Alocador::desalocar_vaga()
{
	//o sistema faz a verificação se ha algum carro alocado, se não existir, da a mensagem que não ha carros na vaga
	int contador = 0;//isso impede que o usuario acesse o metodo sem uma utilidade
	if(aloca_vaga[0][0].verifica_vaga() == false)
	{
	contador = contador + 1;
	}
	if(aloca_vaga[0][1].verifica_vaga() == false)
	{
	contador = contador + 1;
	}
	if(aloca_vaga[0][2].verifica_vaga() == false)
	{
	contador = contador + 1;
	}
	if(aloca_vaga[1][0].verifica_vaga() == false)
	{
	contador = contador + 1;
	}
	if(aloca_vaga[1][1].verifica_vaga() == false)
	{
	contador = contador + 1;
	}
	if(aloca_vaga[1][2].verifica_vaga() == false)
	{
	contador = contador + 1;
	}
	if(aloca_vaga[2][0].verifica_vaga() == false)
	{
	contador = contador + 1;
	}
	if(aloca_vaga[2][1].verifica_vaga() == false)
	{
	contador = contador + 1;
	}
```

```
      if(aloca_vaga[2][2].verifica_vaga() == false)
      {
      contador = contador + 1;
      }
      if(contador == 9)
      {
    MessageBox::Show("Não ha carros guardados no
estacionamento!","Estacionamento Automatico");
      return;
      }
      ////////////////////////////////////////////////////////////////
////////////////////////////
      //desalocador - verifica atraves de dois dados fornecidos pelo
usuário, numero da vaga e senha
      //se o carro pode ser retirado da vaga, se todos os dados
estiverem corretos, o metodo muda o estado de
      //aloca_vaga[i][j] de true para false, isso significa que agora
o vaga esta vazia denovo
      //o metodo caba chamando um outro, pagamento, que calcula o
tempo de saida menos o tempo de entrada do carro
      // e da o valor em dinheiro a ser pago
      Form4 tela;
      if(tela.ShowDialog() == DialogResult::OK)//abre o aplicativo
Form4
      {
            char* posicao = tela.pega_posicao();
            for(int k = 0; k < 2; k++)
            {
                  Alocador::posicao_vaga[k] = posicao[k];//pega a
posição da vaga pela listbox
            }
      }
      int i, j, vaga;
      vaga = atoi(posicao_vaga);
      i=vaga/10;//pega os valores da vaga e joga o primeiro para o
valor na horizontal e o outro na vertical, formando a posição que o
carro esta alocado
      j=vaga%10;
      Form2 compara;
      if(compara.ShowDialog() == DialogResult::OK)//faz a comparação
entre a senha digitada na alocação e a senha da desalocação
      {
            char* senha = compara.getSenha();
            for(int i = 0; i < 6; i++)
            {
                  Alocador::comparacao_senhas[i] = senha[i];
            }
      }
      int H;
      for(H = 0; H <= 5; H++)
      {
            if(comparacao_senhas[H] !=
aloca_vaga[i][j].retorna_senha())
            {
                  MessageBox::Show("nao foi possivel liberar o carro,
numero da vaga ou senha esta incorreta!","Estacionamento Automatico");

                  aloca_vaga[i][j].muda_valor_k();
                  return;
            }
```

```
	}
	Alocador::pagamento(i,j);
	aloca_vaga[i][j].muda_valor_k();
	aloca_vaga[i][j].muda_estado_vaga();
}
void Alocador::pagamento(int i, int j)
{
	//pagamento - metodo que faz a contagem de tempo que o carro
permaneceu alocado, o valor em dinheiro
	// e contado por hora, para testar esse metodo e preciso alterar
o tempo do sistema quando for retirar o carro
	int p,contador = 2;
	int valor_total[2];

	caixa_eletronico.verifica_tempo_vaga();
	valor_total[0] = ((caixa_eletronico.retorna_tempo_hora()) -
(aloca_vaga[i][j].retorna_tempo_hora()));
	valor_total[1] = ((caixa_eletronico.retorna_tempo_minuto()) -
(aloca_vaga[i][j].retorna_tempo_minuto()));
	for(p = 0; p < 23; p++)
	{
		if(valor_total[0] == p)
		{
			contador = contador + p;
			break;
		}
	}
	MessageBox::Show("tempo que o carro permaneceu alocado " +
valor_total[0].ToString() + ":" + valor_total[1].ToString() + "\n" +
"valor total a pagar " + contador.ToString()+ " reais"); }
```

4.4.1.10 AssemblyInfo.cpp

```
#include "stdafx.h"

using namespace System;
using namespace System::Reflection;
using namespace System::Runtime::CompilerServices;
using namespace System::Runtime::InteropServices;
using namespace System::Security::Permissions;

//
// General Information about an assembly is controlled through the
following
// set of attributes. Change these attribute values to modify the
information
// associated with an assembly.
//
[assembly:AssemblyTitleAttribute("Park")];
[assembly:AssemblyDescriptionAttribute("")];
[assembly:AssemblyConfigurationAttribute("")];
[assembly:AssemblyCompanyAttribute("")];
[assembly:AssemblyProductAttribute("Park")];
[assembly:AssemblyCopyrightAttribute("Copyright (c)  2010")];
[assembly:AssemblyTrademarkAttribute("")];
[assembly:AssemblyCultureAttribute("")];

//
```

```
// Version information for an assembly consists of the following four
values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the value or you can default the Revision and
Build Numbers
// by using the '*' as shown below:

[assembly:AssemblyVersionAttribute("1.0.*")];

[assembly:ComVisible(false)];

[assembly:CLSCompliantAttribute(true)];

[assembly:SecurityPermission(SecurityAction::RequestMinimum,
UnmanagedCode = true)];
```

4.4.1.11 Carro.cpp

```
#include "StdAfx.h"
#include "Form2.h"
#include "Carro.h"
#include <iostream>
using namespace std;
using namespace System::Windows::Forms;
using namespace Park;

Carro::Carro(void)
{
      k = -1;
}

Carro::~Carro(void)
{
}
int Carro::pega_medida_carro()//metodo que retorna um valor inteiro 1
se as condições das propriedades do carro forem satisfeitas
{
      char resposta;
      //usuario agora digita s ou naum para a resposta de veiculos
      //cout << " estacionamento para veiculos pequenos, o seu carro e
pequeno?s ou n? " << endl;
      if(MessageBox::Show("estacionamento para veiculos pequenos, o
seu carro e pequeno?", "Pergunta", MessageBoxButtons::YesNo) ==
DialogResult::Yes)
      {
            resposta = 's';
      }
      if(resposta == 's'){//se a resposta for nao o programa passa a
seguinte mensagem
            return 1;
      }
      else{
            MessageBox::Show("Não foi possivel liberar carro para a
alocação!","Erro");
```

```
            return 0;// se todas as condições forem satisfeitas o
programa retorna 1 que sera usado pelo outro metodo senha_carro.
      }
}
char Carro::senha_usuario()
{
      int condição_senha;
      condição_senha = Carro::pega_medida_carro();// a variavel
condição_senha recebe o conteudo do metodo pega_medida
      if(condição_senha == 1)//se a variavel for igual a 1, o atributo
senha_carro recebe a senha que o usario digitar
      {
            //MessageBox::Show("Agradecemos sua preferencia em
contratar nossos serviços","Estacionamento Automatico");
            Form2 dlg;
            if((dlg.ShowDialog() == DialogResult::OK))
            {
                  char* senha = dlg.getSenha();
                  for(int i = 0; i < 6; i++)
                  {
                        Carro::senha_carro[i] = senha[i];
                        if(Carro::senha_carro[i] == 'a')
                        {
                              return 'a';
                        }

                  }
            }
            return Carro::senha_carro[6];
      }
      else if(condição_senha == 0)
      {
            return 'a';
      }

}
char Carro::retorna_senha_usuario()
{
      return Carro::senha_carro[k = k + 1];
}
int Carro::k_recebe_menos1()
{
      k = -1;
      return k;
}
```

4.4.1.12 Form2.cpp

```
#include "StdAfx.h"
#include "Form2.h"

char* Park::Form2::getSenha()
{
      return senha;
}
```

4.4.1.13 Form3.cpp

```
#include "StdAfx.h"
#include "Form3.h"

char* Park::Form3::get_posicao()
{
	return posicao;
}
```

4.4.1.14 Form4.cpp

```
#include "StdAfx.h"
#include "Form4.h"

char* Park::Form4::pega_posicao()
{
return posicao;
}
```

4.4.1.15 Park.cpp

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

#include "stdafx.h"
#include "Form1.h"
#include "Form2.h"
using namespace Park;

[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;
}
```

4.4.1.16 Vaga.cpp

```
#include "StdAfx.h"
#include "Vaga.h"
#include <iostream>
using namespace std;
using namespace System::Windows::Forms;

Vaga::Vaga(void)
{
	Vaga::estado_vaga = false;

}
Vaga::~Vaga(void)
{
}
void Vaga::muda_estado_vaga()
```

```
{
      if(estado_vaga == false)// se a vaga estiver cheia tempo_inicio
recebe o metodo verifica_tempo_vaga
      {
            estado_vaga = true;
      }
      else
      {
            estado_vaga = false;
      }
      }
int Vaga::verifica_tempo_vaga()
{
  struct tm *tempo_sistema;
  time_t currentTime;
  /* Pega a hora atual do sistema. */
  currentTime = time(NULL);
  /* Converte-o em uma estrutura tm. */
  tempo_sistema = localtime(&currentTime);
  tempo_vaga[0] = tempo_sistema->tm_hour;/* Apresenta a hora atual no
formato horas:minutos:segundos */
  tempo_vaga[1] = tempo_sistema->tm_min;
  MessageBox::Show("Hora: " + tempo_sistema->tm_hour.ToString() + ":"
+ tempo_sistema->tm_min.ToString());
  return 0;
}
bool Vaga::verifica_vaga()
{
      return estado_vaga;
}

char Vaga::acessa_atributos()
{
      return recebe_senha_carro.senha_usuario();
}
char Vaga::retorna_senha()
{
      return recebe_senha_carro.retorna_senha_usuario();
}
Carro Vaga::retorna_recebe_senhas_carro()
{
      return recebe_senha_carro;
}
int Vaga::muda_valor_k()
{
      return recebe_senha_carro.k_recebe_menos1();
}
int Vaga::retorna_tempo_hora()
{
      return tempo_vaga[0];
}
int Vaga::retorna_tempo_minuto()
{
      return tempo_vaga[1];
}
```

4.4.1.17 Motor.cpp

```
#include <Motor.h>
#include <Windows.h>

Motor::Motor()
{

}

bool Motor::AbrirPorta()
{
   HANDLE   hCom; // Handle para a Porta Serial (identificador).
   char *NomePorta = "\\\\.\\COM13";  //COM1, COM2...COM9 ou portas
virtuais "\\.\COMx".
 hCom = CreateFile(
                   NomePorta,   //Nome da porta.
                   GENERIC_READ | GENERIC_WRITE,   //Para leitura e
escrita.
                   0,       //(Zero) Nenhuma outra abertura será
permitida.
                   NULL,  //Atributos de segurança. (NULL) padrão.
                   OPEN_EXISTING,  //Criação ou abertura.
                   0,         //Entrada e saída sem overlapped.
                   NULL     //Atributos e Flags. Deve ser NULL para
COM.
                   );

   if(hCom == INVALID_HANDLE_VALUE)
      return false;   //Erro ao tentar abrir a porta especificada.
}
```

4.4.2 Software Arduino

```
/*************************************************************/
/* Código para receber informação pela porta serial.         */
/* Desenvolvido por Alex dos Santos Xavier                   */
/* 3° Período - 2010/ Engº de Computação/ PUC-PR             */
/*************************************************************/

#include <Stepper.h>
#include <MotorPasso.h>

// Passa a quantidade de passos que o motor possui por volta.
#define STEPS 200

// cria a classe Stepper, especificando o numero de
// passos do motor e os pinos que o controlaram
//Stepper stepper(STEPS, 8, 9, 10, 11);
MotorPasso teste(48,7,6,5,4);
int go;// Variavel que recebera informação da porta serial

void setup()
{
  //Envia informação PARA o pc
  Serial.begin(9600);//Inicia a velocidade de comunicação
```

```
  //pinMode(7, OUTPUT);
  //
  pinMode(6, OUTPUT);

  // configura a velocidade do motor para 15 RPMs
  //stepper.setSpeed(15);
  teste.ajustaVelocidade(15);
}

void loop()
{
  if(Serial.available())//se o usb está desponivel então
  {
    go = Serial.read();//leia informação
    if (go == 97)//"a"
    {
      go = -30;
     //stepper.step(go);//Gira o motor em x passo
      teste.movimenta(go);
    }
    if (go == 104)//"h"
    {
      go = 30;
      //stepper.step(go);//sendo negativo inverte o sentido
      teste.movimenta(go);
    }
    if(go == 111)//"o"
    {
      digitalWrite(7, HIGH);
      digitalWrite(6, LOW);
      delay(2000);



      digitalWrite(7, LOW);
      Serial.println("rodou");
    }
    if(go == 99)//"c"
    {
      digitalWrite(7, LOW);
      digitalWrite(6, HIGH);
      delay(2000);

      digitalWrite(6, LOW);
    }
    //Parte do código para testar informações lidas pelo arduino
    //util para descobrir o numero correspodente de determinado
    //botão:
    Serial.println("informacao lida:");//imprime
    Serial.println(go, DEC);
  }

}
```

4.4.3 Botões

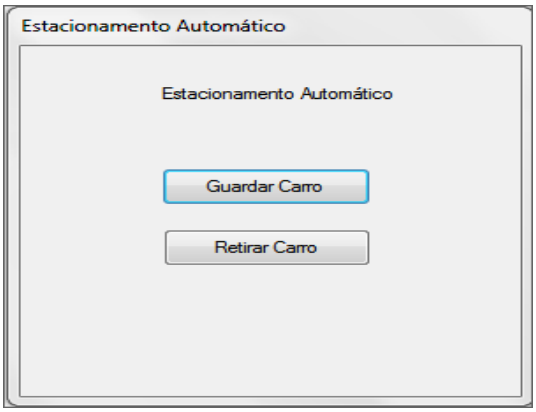


Figura 4- Botão do software.

5. PROBLEMAS ENCONTRADOS

Protótipo: O principal problema encontrado na construção do protótipo foi na fixação do elevador em cima do eixo x, pois quando fixava a parte de cima, criava-se atrito entre o carro e a barra de ferro e quando fixava a parte de baixo a parte de cima ficava “frouxa”. O problema foi parcialmente resolvido com uma barra de metal que fixa o meio do elevador às laterais do protótipo.

Durante o desenvolvimento do projeto optamos por não implementar a rampa que estacionaria o automóvel automaticamente devido a problemas no desenvolvimento da mesma, mas principalmente pela falta de tempo. Ponderamos que seria mais vantajoso deixar esse módulo pra trás, já que o mesmo estava atrasando todo o projeto.

Software de gerenciamento do projeto: A falta de experiência ao implementar o programa foi a principal dificuldade encontrada, em muitas partes do projeto, tivemos muita dificuldade com os métodos e funções do visual c++ que não conhecíamos, também por ser um programa de maior complexidade que os que fizemos em sala de aula, houve muitos erros de lógica que foram sendo corrigidos com pesquisas na internet e trocas de informações com outras pessoas do curso, além de ter que aprender como fazer um novo projeto para o aplicativo, projeto Windows Form Aplication, com uma linguagem muito mais avançada do que estamos acostumados.

Podemos afirmar que com o conhecimento que adquirimos em programação,o qual fomos desenvolvendo ao longo do projeto, faríamos o software de maneira mais otimizada e segura, usando conceitos como os de herança e banco de dados e funções como template, de tratativas de erro, entre outras.

Circuitos: O maior problema encontrado foi trabalhar com transistores, e a falta de experiência.

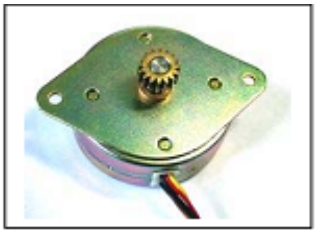
6. CONCLUSÃO

Trabalho em equipe, respeitar o cronograma que fizemos para cada etapa do projeto e principalmente ter uma grande perspectiva se aquilo que prometemos realmente poderá ser feito são conclusões que tiramos, tanto na etapa do pré – projeto quanto no seu termino.

Questões como divergência de idéias entre os participantes, horários de encontro para realização do trabalho e reuniões, tem que ser muito bem administradas pelo grupo e a compreensão mútua entre os participantes com certeza garante grande parte da conclusão do projeto.

7. CURIOSIDADES

7.1 Motores de passo



Motores de passo são motores DC, mas com a possibilidade de se controlar sua velocidade, direção e ângulo, pode-se girá-los num ângulo determinado com extrema precisão. Eles podem ser encontrados em dois tipos diferentes: magnético permanente e relutância variável. O motor de passo magnético permanente tem como característica “agarrar” quando se gira seu eixo, estando o motor desligado. Este tipo de motor possui tipicamente dois enrolamentos independentes, possuindo assim quatro fios, este motor é normalmente conhecido como motor bipolar. O segundo tipo de motor, o de relutância variável, tem como característica ter seu eixo “livre”, estes motores possuem quatro enrolamentos e um fio comum entre eles, tipicamente este motores possuem cinco ou seis fios, este são normalmente chamados de motores unipolares. As configurações mais comuns são motores de passo que dão uma volta completa (360º) em 48 e 200 passos. Este é um típico motor de passo unipolar. Ele possui cinco fios, sendo um deles o comum entre as quatro bobinas.

7.2 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.

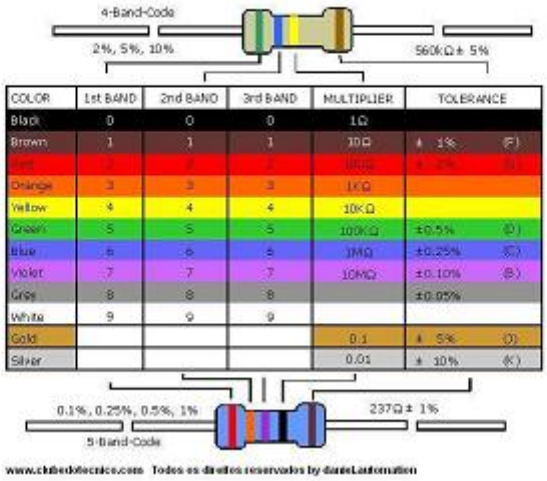


Figura 5 - Tabela de multiplicação dos resistores.

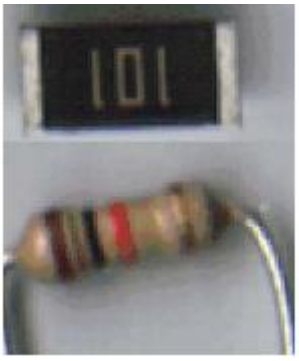


Figura 6- Imagem de um resistor SMD (acima) e um resistor de carbono (abaixo).

Fonte: Wikipédia

7.3 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 transistor 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.

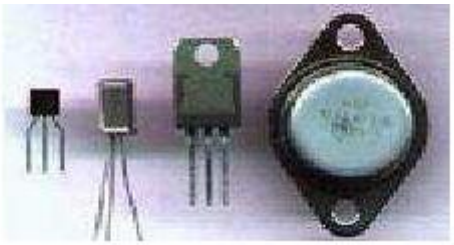


Figura 7 - Modelos de transistores existentes.

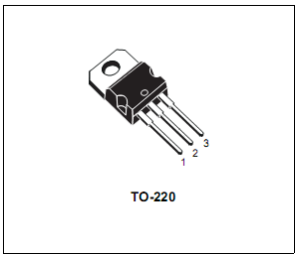


Figura 8 - Transistor TIP 122/125.

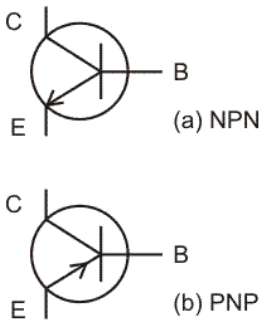


Figura 9: Símbolo de um transistor tipo NPN e outro PNP.

Fonte: Wikipédia, Guia do Hardware,datasheetcatalog,com.

7.4 Arduíno

Arduino, por vezes traduzida ao português como Arduíno, um computador físico baseado numa simples plataforma de hardware livre, projetada com um microcontrolador de placa única, com suporte de entrada/saída embutido e uma linguagem de programação padrão, na qual tem origem em Wiring, e é essencialmente C/C++. O objetivo do projeto é criar ferramentas que são acessíveis, com baixo custo, flexíveis e fáceis de usar por artistas e amadores. Principalmente para aqueles que não teriam alcance aos controladores mais sofisticados e de ferramentas mais complicadas.

Pode ser usado para o desenvolvimento de independentes objetos interativos, ou ainda para ser conectado a um computador hospedeiro. Uma típica placa Arduino é composta por um controlador, algumas linhas de E/S digital e analógica, além de uma interface serial ou USB, para interligar-se ao hospedeiro, que é usado para o programar e o interagir em tempo real. Ele em si não possui qualquer recurso de rede, porém é comum combinar um ou mais Arduinos deste modo, usando extensões apropriadas chamadas de shield. A interface do hospedeiro é simples, podendo ser escrita em várias linguagens. A mais popular é a Processing, mas outras que podem comunicar-se com a conexão serial são: Max/MSP, Pure Data, SuperCollider, ActionScript e Java

Atualmente, seu hardware é feito através de um microcontrolador Atmel AVR, sendo que este não é um requerimento formal e pode ser estendido se tanto ele quanto a ferramenta alternativa suportarem a linguagem Arduino e forem aceitas por seu projeto.[2] Considerando esta característica, muitos projetos paralelos se inspiram em cópias modificadas com placas de expansões, e acabam recebendo seus próprios nomes.

Apesar de o sistema poder ser montado pelo próprio usuário, os mantenedores atualmente possuem um serviço de venda do produto pré-montado, através deles próprios e também por distribuidores oficiais com pontos de venda mundiais.

O projeto iniciou-se na cidade de Ivrea, Itália, em 2005, com o intuito de interagir em projetos escolares de forma a ter um orçamento menor que outros sistemas de prototipagem disponíveis naquela época. Seu sucesso foi sinalizado com o recebimento de uma menção honrosa na categoria Comunidades Digitais em 2006, pela Prix Ars Electronica, além da marca de mais de 50.000 placas vendidas até outubro de 2008

Hardware

Sua placa consiste em um microcontrolador Atmel AVR de 8 bits, com componentes complementares para facilitar a programação e incorporação para outros circuitos. Um importante aspecto é a maneira padrão que os conectores são expostos, permitindo o CPU ser interligado a outros módulos expansivos, conhecidos como *shields*. Os Arduinos originais utilizam a série de chips *megaAVR*, especialmente os *ATmega8*, *ATmega168*, *ATmega328* e a *ATmega1280*; porém muitos outros processadores foram utilizados por clones deles.

A grande maioria de placas inclui um regulador linear de 5 volts e um oscilador de cristal de 16 MHz (Podendo haver variantes com um ressonador cerâmico), embora alguns esquemas como o *LilyPad* usam até 8 MHz e dispensam um regulador de voltagem embutido, por ter uma forma específica de restrições de fator. Além de ser microcontrolador, o componente também é pré-programado com um bootloader que simplifica o carregamento de programas para o chip de memória flash embutido, comparado com outros aparelhos que usualmente necessitam de um chip programador externo.

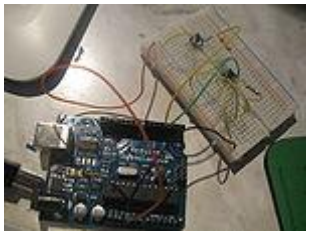


Figura 10 - Arduino conectado a uma protoboard

Conceitualmente, quando seu software é utilizado, ele monta todas as placas sobre uma programação de conexão serial RS-232, mas a maneira que é implementado no hardware varia em cada versão. Suas placas serial contém um simples circuito inversor para converter entre os sinais dos níveis RS-232 e TTL. Atualmente, existem alguns métodos diferentes para realizar a transmissão dos dados, como por placas programáveis via USB, adicionadas através de um chip adaptador *USB-para-Serial* como o FTDI FT232. Algumas variantes, como o Arduino Mini e o não oficial Boarduino, usam um módulo, cabo adaptador USB, Bluetooth ou outros métodos. Nestes casos, são usados com ferramentas microcontroladoras ao invés do Arduino IDE, utilizando assim a programação padrão AVR ISP.

A maioria dos pinos de E/S dos microcontroladores são para uso de outros circuitos. A versão *Diecimila*, que foi substituída pela *Duemilanove*, por exemplo, disponibiliza 14 pinos digitais, 6 das quais podem produzir sinais MLP, além de 6 entradas analógicas. Estes estão disponíveis em cima da

placa, através de conectores fêmeas de 0,1 polegadas (ou 0,25 centímetros). O modelo *Nano*, *Boarduino* e placas compatíveis com estas, fornecem conectores machos na parte de baixo da placa, para serem plugados em protoboards.

Software

O Arduino IDE é uma aplicação multi-plataforma escrita em Java na qual é derivada dos projetos *Processing* e *Wiring*. É esquematizado para introduzir a programação a artistas e a pessoas não familiarizadas com o desenvolvimento de software. Inclui um editor de código com recursos de realce de sintaxe, parênteses correspondentes e identação automática, sendo capaz de compilar e carregar programas para a placa com um único clique. Com isso não há a necessidade de editar Makefiles ou rodar programas em ambientes de *linha de comando*.

Tendo uma biblioteca chamada "Wiring", ele possui a capacidade de programar em C/C++. Isto permite criar com facilidade muitas operações de entrada e saída, tendo que definir apenas duas funções no pedido para fazer um programa funcional:

- *setup()* – Inserida no inicio, na qual pode ser usada para inicializar configuração, e
- *loop()* – Chamada para repetir um bloco de comandos ou esperar até que seja desligada.

Hardware oficial

O Arduino original é fabricado pela companhia italiana Smart Projects, porém a estadunidense SparkFun Electronics também possui algumas marcas comerciais sob a mesma licença.

Até hoje foram produzidas comercialmente 11 versões do dispositivo.

Modelo	**Descrição e tipo de conexão ao hospedeiro**	**Controlador**
Serial Arduino	Serial DB9 para programação	ATmega8
Arduino Extreme	USB para programação	ATmega8
Arduino Mini	Versão em miniatura do Arduino utilizando montagem superficial	ATmega168
Arduino Nano	Versão menor que o Arduino Mini, energizado por USB e conectada por montagem superficial	ATmega168
Lily Pad Arduino	Projeto minimalista para aplicações portáteis, utilizando montagem superficial	ATmega168
Arduino NG	USB para programação	ATmega8
Arduino NG plus	USB para programação	ATmega168
Arduino BT	Interface Bluetooth para comunicação	ATmega168
Arduino Diecimila	Interface USB	Atmega168 em um pacote DIL28

		(foto)
Arduino Duemilanove	Duemilanove significa "2009" em italiano. É energizado via USB/DC, com alternação automática	Atmega168 (Atmega328 para a versão mais nova)
Arduino Mega	Montagem superficial	Atmega1280 para E/S adicionais e memória

Fonte: Wikipedia.org

7.5 Placa Fenolite

É uma placa de plástico com cobre em uma de sua superfícies, é utilizada para a impressão de circuitos.

7.6 Eagle

Programa utilizado para o desenho de circuitos para posteriormente serem impressos na placa de fenolite.

8. ANEXOS

Figura 11 - Protótipo com os motores acoplados.

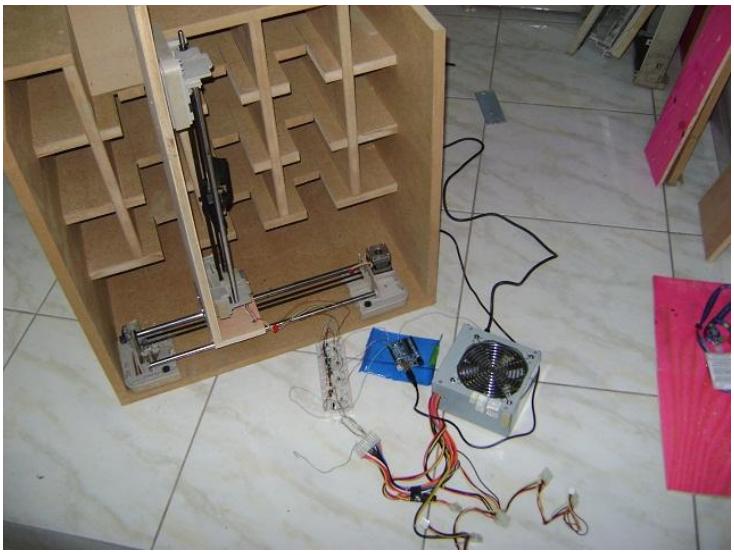


Figura 12 - Testes com motores X eY.

Figura 13 - Testes com o programa desenvolvido.

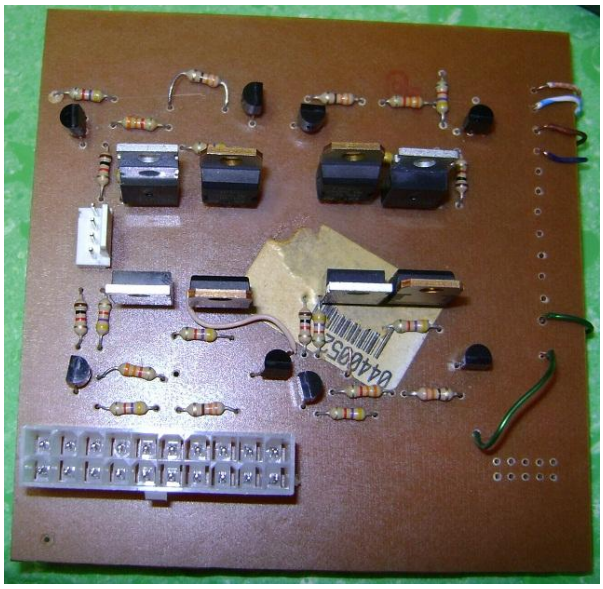


Figura 14 - Placa Principal.

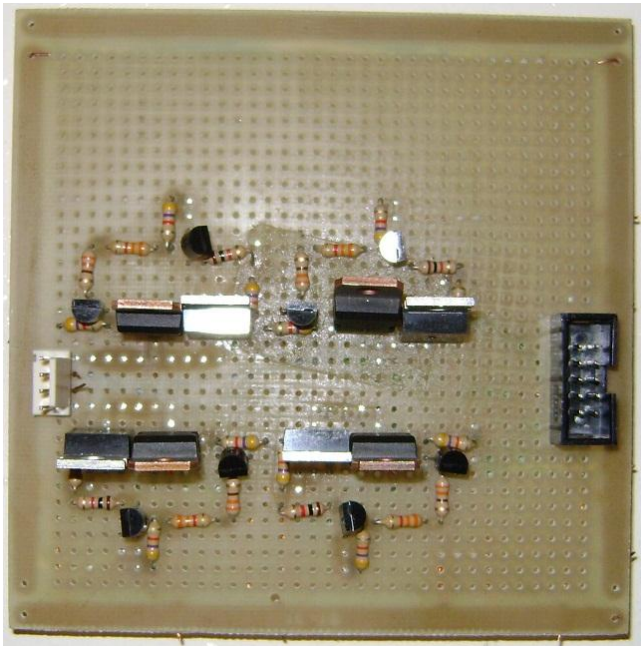


Figura 15- Placa Secundária

Figura 16 – Estacionamento Automatizado