CENTRO PAULA SOUZA
COMPETÊNCIA EM EDUCAÇÃO PÚBLICA PROFISSIONAL
Centro Estadual de Educação Tecnológica Paula Souza
GOVERNO DO ESTADO DE SÃO PAULO

ADENALDO OLIVEIRA DA SILVA

GUSTAVO FERREIRA DA SILVA

LUIZ GUSTAVO MOREIRA DA SILVA

AUTOMAÇÃO RESIDENCIAL COM ARDUINO

Trabalho de conclusão de curso apresentado a ETEC Francisco Garcia para obtenção do título de Técnico em Informática sob a orientação do Professor Pedro Ramires da Silva Amalfi Costa.

ETEC FRANCISCO GARCIA

CAJURU - SP

2º SEMESTRE/2013

Dedicamos este trabalho
aos nossos professores que
sempre nos apoiaram e colaboraram
com a realização deste. Também aos
nossos familiares pelo apoio e incentivo.

AGRADECIMENTO

Agradecemos aos professores e a todos os que colaboraram na elaboração deste projeto.

“Uma experiência nunca é um fracasso, pois sempre vem demonstrar algo”

(Thomas Edison)

RESUMO

Este projeto consiste em uma demonstração de uma forma de automação residencial, onde através de um sistema se controla toda a casa desde o portão da garagem até na luz do banheiro. O sistema rodará sobre uma placa eletrônica Arduino que gerencia toda a automação da casa e através de um *software* onde sistema é controlado. O software foi desenvolvido em uma linguagem de programação da Microsoft (*o Visual Basic. NET*), a interface é bem definida e simplificada, onde não se exige conhecimento de informática para utilizá-lo. Por se tratar de um protótipo, foi optado por utilizar-se apenas a porta USB, em uma conexão feita por trasmissão serial, onde os dados são transmitidos por um bit de cada vez. Através da interface, o usuário clica no botão da ação desejada e é enviado um caractere que corresponde a essa ação para a porta USB que transmite e passa para o Arduino, fazendo com que este execute o que foi pedido. Devido à praticidade, qualquer pessoa consegue automatizar o que quiser desde que tenha tempo livre, dedicação e persistência.

Palavras-chave: Arduino, Automação Residencial.

ABSTRACT

This project consists of a demonstration of a form of home automation, through a system where you control the whole house from the garage door to the bathroom light. The system will run on an electronic board that manages Arduino whole home automation and through a software system which is controlled. The software was developed in a programming language from Microsoft (Visual Basic. NET), the interface is well defined and streamlined, which does not require computer knowledge to use it. Since this is a prototype, it was chosen to use only the USB port on a connection made by serial transmission to where the data is transmitted one bit at a time. Through the interface, the user clicks on the button of the desired action and sent a character that corresponds to this action for the USB port that transmits and passes to the Arduino, causing it to perform what was asked. Due to the convenience, anyone can automate what you want provided you have free time, dedication and persistence.

Keywords: Arduino, Home Automation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AC – *Alternating Current*

ADK – *Android Development Kit*

BIT – *Binary Digit*

CD – *Compact Disc*

CLP – Controlador Lógico Programável

DC – *Direct Current*

KB - *Kilobytes*

LEDs – *Light Emitting Diodes*

mA – *Miliampère*

Mhz - *Megahertz*

USB – *Universal Serial Bus*

V – *Volts*

VB. NET – *Visual Basic. NET*

SUMÁRIO

1. INTRODUÇÃO

Este projeto consiste em uma demonstração de automação residencial, onde se pode controlar desde a iluminação e os ventiladores, até o portão de uma casa.

Para isto, foi utilizada a plataforma Arduino, que consiste em uma placa com um microcontrolador, que torna possível o controle através de uma comunicação com o computador com os outros dispositivos ligados a ela, como por exemplo, tendo um ventilador ligado à placa Arduino, pode-se controlar suas funções básicas, desde ligar e desligar, até controlar sua velocidade de rotação.

Para que essa automação fosse utilizada pelo usuário, foi desenvolvido um programa com uma interface gráfica, visando facilitar a interação e a usabilidade.

Este programa faz a leitura da interface e permite que o usuário possa interagir ajustando, por exemplo, o brilho da lâmpada e a velocidade de rotação de um ventilador. Estas informações são lidas e transmitidas para a placa através da interface de comunicação entre o computador e a placa.

2. O ARDUINO

Aqui será falado mais sobre o Arduino, sua história, suas versões e a aplicação neste projeto.

2.1 HISTÓRIA

O Arduino (figura 1 abaixo) surgiu em 2005 na Itália. Criado por um professor chamado Massimo Banzi com o intuito de ensinar programação e eletrônica para seus alunos de *design* para que eles pudessem utilizar em seus projetos de arte, aumentando a interação e utilizando-se da robótica. (MAGGIONI,RIOS, et al.,2012).

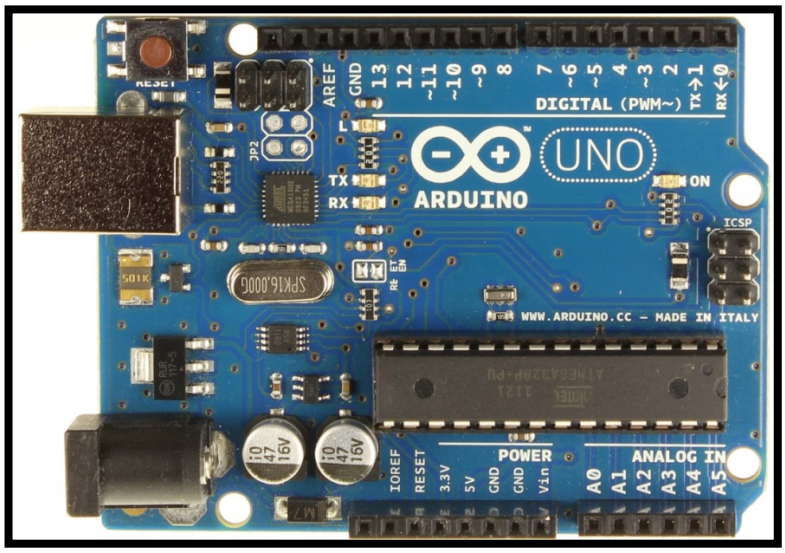

Figura 1 - Exemplo de placa Arduino, modelo Uno

Fonte: http://www.semageek.com/wp-content/uploads/2011/12/arduino-uno-est-passe-en-revision-r3-details-des-differences-02.jpg

No entanto, além de ser difícil ensinar eletrônica, trabalhar com microcontroladores era uma tarefa árdua. Pois além de exigir um bom conhecimento de eletrônica, ainda exigia outros investimentos, como por exemplo: gravadores para microcontroladores que precisavam ser comprados ou construídos, entre outros fatores, isso aumentava o tempo e o custo.

Segundo SANTIGO e TIENE, et al.(2012, p.3) “Pensando nisso, Massimo Banzi e David Cuartielles decidiram criar sua própria placa. O aluno David Mellies foi o responsável pela linguagem de programação do Arduino.”

O *Hardware* foi desenvolvido sobre uma licença livre, de forma que qualquer um pode criar a sua placa sem ter que pagar direitos autorais ou ser considerado crime.

Existem muitos clones, como por exemplo, o Illuminato, que possui 32 Entradas e Saídas e 10 LEDs que podem ser controlados para objetivos específicos e apenas com uma instrução desenvolvida pela *Liquidware*.

No Brasil, tem o Brasuíno que possui um microcontrolador extra que pode ser usado para criar circuitos avançados USB. Há também o Tatuino, desenvolvido pela Tato e com uma forte comunidade de usuários, possui as mesmas especificações do original, porém acompanha CD com códigos-exemplo.

Os circuitos podem ser montados em uma placa para protótipos (mais conhecida como *protoboard* ou *breadboard* ou ainda matriz de pontos). Devido a isso, ele é voltado para engenheiros iniciantes ou *hobbyistas*. Por ser uma solução tudo em um, de código aberto, mais de 500.000 placas foram vendidas até o ano de 2008. A facilidade e a ligeira rapidez na implementação vem atraindo cada vez mais simpatizantes para a plataforma.

Seu maior uso é no desenvolvimento de protótipos para implementar e controlar sistemas interativos, tanto a nível doméstico, comercial ou móvel. Algo semelhante ao que o CLP (Controlador Lógico Programável) faz na indústria.·.

Abaixo serão citados os principais componentes que constituem o Arduino. Lembrando que é um padrão aberto, portanto existem várias versões oficiais e clones do mesmo que podem ter componentes diferentes.

2.1.1 MICROCONTROLADOR

Pode ser entendido como um computador, porém com o objetivo de controlar apenas algo específico, por isso ele geralmente é encontrado embutido em um produto comercial.

Ele é dedicado, ou seja, feito para realizar apenas aquela função. Ao contrário dos computadores que podem realizar diversas tarefas.

Assim como o computador, ele geralmente executa um programa que é armazenado em uma memória. Porém, ao contrário dos computadores tradicionais que possuem uma memória que sempre muda o que está gravado conforme executa-se outras tarefas, o programa é armazenado numa memória em que é possível realizar apenas sua leitura. Para modificar algo, deve-se gravar novamente nela, apagando o que já estava gravado.

2.1.2 NÚMEROS BINÁRIOS (BITS)

Binário é um sistema numérico composto por apenas dois algarismos. (0 e 1). Um microcontrolador entende exclusivamente os denominados impulsos elétricos, que podem ser de nível alto ou baixo (Respectivamente 1 ou 0). Cada dígito corresponde a 1 *bit* (abreviação de *binary digit* ou dígito binário). 8 bits equivalem a 1 *byte*. Por isso, para que a placa entenda tudo o que foi escrito na linguagem do Arduino, é necessário converter esse código para binário antes de ser realizada a gravação. Este processo é feito pelo Ambiente de Desenvolvimento Integrado do Arduino.

2.1.3 PORTAS DE ENTRADA E SAÍDA

São através dessas portas que o Arduino executa tudo o que foi programado e gravado pelo usuário. Elas são divididas em dois tipos: Analógicas e Digitais, nas quais serão explicadas abaixo.

2.1.4 ENTRADAS E SAÍDAS ANALÓGICAS

Um sinal analógico é aquele que pode ser variado entre um determinado intervalo. Analógico vem de análogo (Analogia). Onde para ter-se um valor, precisa-se comparar tendo referência como outro. Por exemplo, no ponteiro do relógio analógico, fazer referência à posição do sol.

No Arduino, nessas entradas utiliza-se sensores e componentes em que ilustra-se melhor este conceito. Exemplo: Um potenciômetro que varia a resistência elétrica partindo de um ponto, em que vai se alterando a dimensão do mesmo, conforme ela aumenta a resistência também vai aumentando.

2.1.5 ENTRADAS E SAÍDAS DIGITAIS

Um sinal digital já se provém de números e destaca-se por ser preciso, não necessitando fazer analogia e nem dependendo de nenhum outro meio nem referência.

Ao contrário do analógico, ele é finito e definido apenas com aquela frequência em um determinado instante. Ou seja, repete-se, mas somente naquilo que foi determinado.

2.1.6 COMUNICAÇÃO SERIAL

No padrão de comunicação serial, todos os bits são transferidos em fila, ou seja, um por vez, isto em apenas uma “fila”.

O Arduino utiliza-se do padrão USB (*Universal Serial Bus* ou Barramento Serial Universal), que garante maior velocidade e facilidade, por poder ser plugado em qualquer computador que tenha uma conexão desse tipo e um sistema operacional compatível (*Windows, Linux, MacOS).* Também é através desse barramento que a placa é alimentada, pois as portas USB possuem uma tensão de 5 V.

2.2 ESPECIFICAÇÕES DO ARDUINO

Existem diversas versões do Arduino, as principais serão citadas abaixo:

2.2.1 ARDUINO UNO

O Arduino Uno (Um em italiano) é considerada a primeira versão da placa, constitui-se de menos portas que os mais avançados, tendo assim o melhor custo x benefício (Geralmente é encontrada com preços entre R$ 60,00 e 80,00), por

isso é mais voltada para os iniciantes. Levando em conta isso, ela foi a escolhida para ser utlizada nesse projeto. Seguem as especificações:

- 14 Portas Digitais que podem ser configuradas para entrada ou saída de dados.
- 6 Entradas Analógicas.
- Cristal Oscilador de 16 Mhz.
- Conexão USB.
- Soquetes ICSP.
- Botão de *Reset.*
- Saída de 5V e 3,3V.
- Possibilidade de alimentação externa (9V).

2.2.2 ARDUINO MEGA

O destaque do Arduino Mega (figura 2 abaixo) é a presença de mais portas, tornando-a ideal para projetos de grande porte e que utilizem diversos acessórios, como por exemplo, os famosos *shields* que são outras placas auxiliares que se encaixam sobre a principal.

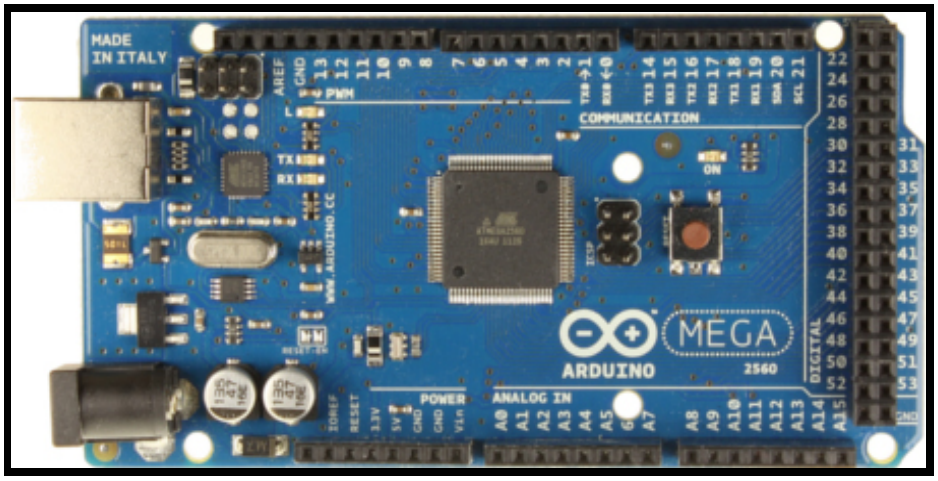

Figura 2 - Arduino Mega

Fonte: http://arduino.cc/en/uploads/Main/ArduinoMega2560_R3_Front_450px.jpg

As especificações são as seguintes:

- Microcontrolador Atmega 2560
- Tensão de operação: 5 V

- Tensão de Entrada (Limites): 7-12 V
- Tensão de Entrada (Recomendada): 6-20 V
- Portas de Entrada e Saída Digitais: 54 (14 podem ser controladas por largura de pulso, ou seja, pode-se variar o ciclo ou a tensão de operação).
- Portas de Entrada Analógicas: 16
- Corrente DC por Entrada/Saída: 40 mA
- Corrente DC para pino de 3,3 V: 50 mA
- Memória *Flash*: 256 KB (8 KB, são usados para o *bootloader*, ou seja para a inicialização da gravação do projeto).
- SRAM (Memória Estática): 8 KB
- EEPROM (Memória Apagável por Meios Elétricos): 4 KB
- Velocidade de *Clock*: 16 Mhz

2.2.3 ARDUINO ADK

O Arduino ADK (Figura 3 abaixo) é basicamente o mesmo do Mega, porém ele inclui uma interface USB para comunicação com celulares e *tablets* que rodam o sitema operacional *Android* do Google. As especificações são as mesmas do Arduino Mega.

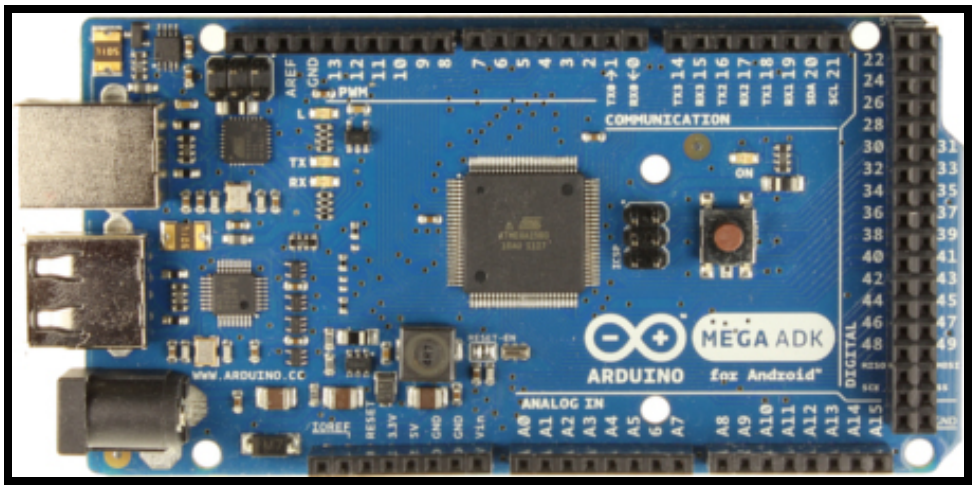

Figura 3 - Arduino ADK

Fonte: http://arduino.cc/en/uploads/Main/ArduinoADK_R3_Front_450px.jpg

2.2.4 ARDUINO LEONARDO

O Arduino Leonardo (figura 4 abaixo) foi projetado para usuários mais avançados. Tem praticamente as mesmas especificações do Arduino Uno, porém as funções de programação permitem utilizá-lo como teclado e/ou mouse e contra com um microcontrolador mais rápido que o do Uno.

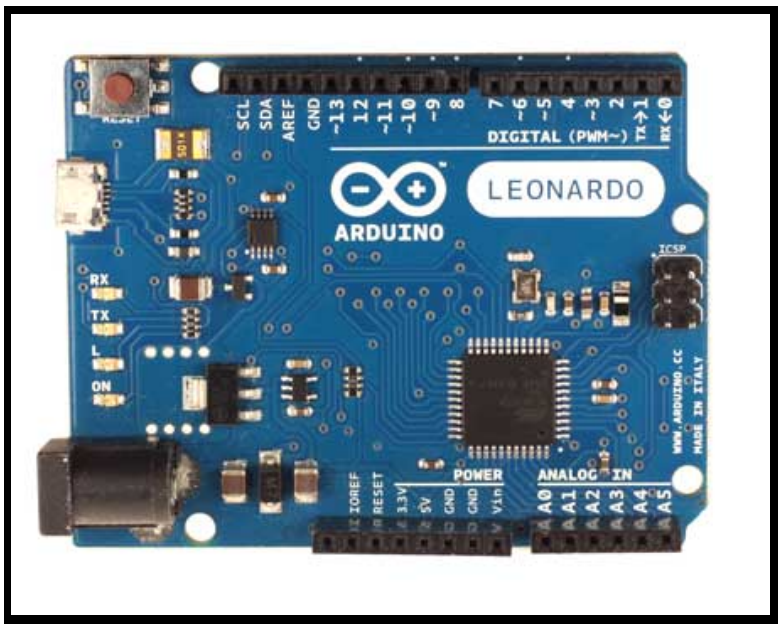

Figura 4 - Arduino Leonardo

Fonte: http://arduino.cc/en/uploads/Main/ArduinoLeonardoFront_2_450px.jpg

As especificações são as seguintes:

- Microcontrolador: ATmega32U4
- Tensão de operação: 5 V
- Tensão de entrada (Limites): 7-12 V
- Tensão de entrada (Recomendada): 6-20 V
- Pinos de Entrada/Saída Digitais: 20 (Na qual, 7 deles podem ser controlados por largura de pulso, ou seja, pode-se variar o ciclo ou a tensão).
- Entradas/Saídas Analógicas: 12
- Corrente DC por Pino de Entrada/Saída: 40 mA
- Corrente do Pino 3.3 V: 50 mA

- Memória *Flash*: 32kB(Atmega328) (0,5 KB são usados pelo *bootloader*, ou seja, para carregar o programa gravado, ou iniciar a gravação).
- SRAM: 2,5KB(ATmega2560)
- EEPROM: 1KB(ATmega2560)
- Velocidade do *Clock*: 16 Mhz

3. O PROJETO

O software foi desenvolvido incialmente utilizando o Ambiente de Desenvolvimento Integrado do Arduino (Figura 5 abaixo).

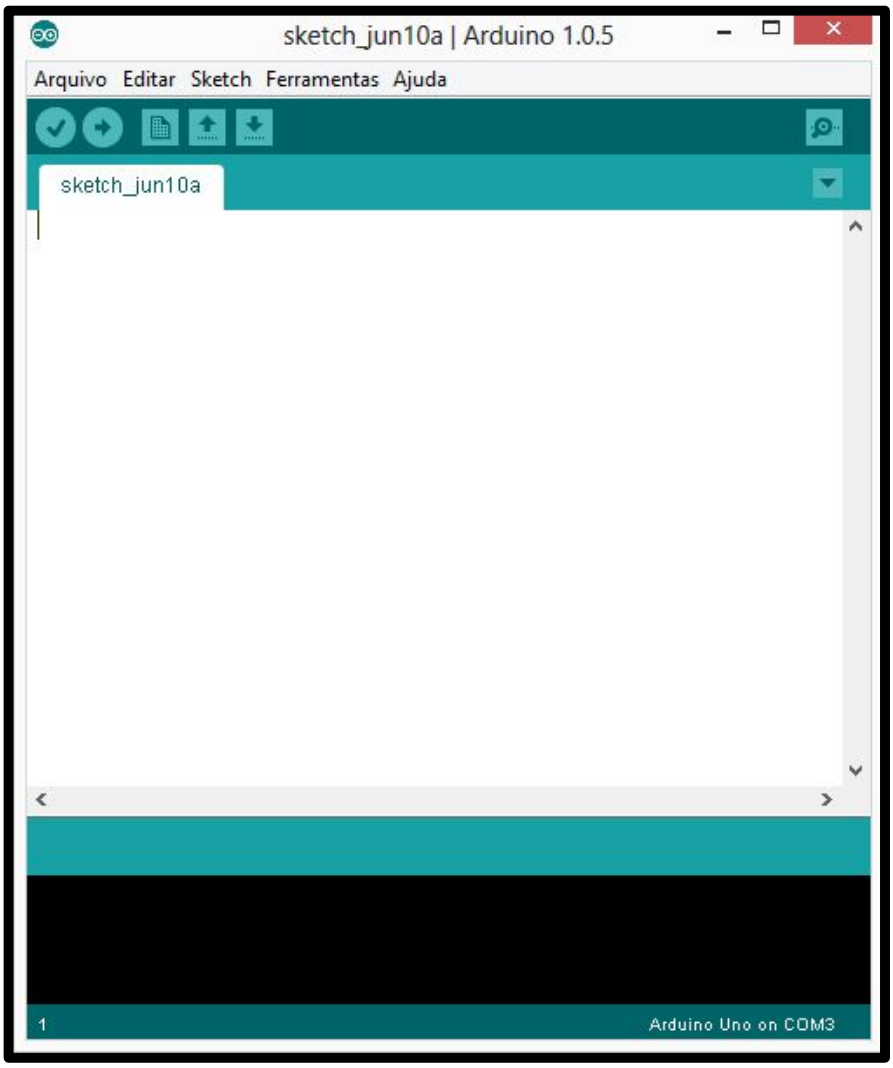

Figura 5 - Ambiente de Desenvolvimento Integrado do Arduino

Fonte: Autoria própria.

Exemplo de código:

```
int led = 13;
```

Nesta linha foi declarada uma variável do tipo inteiro com o nome de led e atribuido a ela o valor corresponde ao pino do Arduino que será utilizado, no caso o pino 13.

```
void setup() {
```

A função void setup é aquela utilizada para atribuir valor às variáveis, configurar os modos dos pinos do Arduino e incluir as bibliotecas que poderão ser utlizadas. Esta é executada apenas uma vez quando o Arduino é ligado ou quando é pressionado o botão de Reset.

pinMode(led, OUTPUT);

}

Nesta linha, é chamada a função pinMode que define uma entrada ou saida para a placa arduino, neste caso fala que o pino led (configurado como 13) está configurado como saída, ou seja, ele receberá um sinal digital ou analógico do Arduino.

void loop() {

A função void loop como o nome sugere, é um loop, ou seja esta função será sempre repetida enquanto o programa estiver em execução.

digitalWrite(led, HIGH);

Foi chamado a função digitalWrite que é a responsável por enviar ou não 5 V ao pino solicitado. Para enviar tem de se passado o parâmetro HIGH junto com a porta desejada (nesta linha está sendo mandado 5V para o pino 13 para que se acenda o led desejado).

delay(1000);

Espera 1000 milisegundos (ou 1 segundo). Ou seja, durante 1 segundo o LED permanecerá ligado.

digitalWrite(led, LOW);

A mesma função descrita anteriormente mas agora ele recebe como parâmetro um sinal digital baixo (LOW) do Arduino e desliga o LED.

delay(1000);

Espera 1000 milisegundos (ou 1 segundo). Ou seja, durante 1 segundo o LED permanecerá desligado.

}

Este projeto consiste basicamente de caracteres enviados pela porta serial do PC para o Arduino. Ou seja, digita-se tal caractere e ele ativa ou desativa o componente que estiver conectado a ele.

Para o controle do motor do portão, foi necessário inverter a polaridade do motor para que ele pudesse girar, ora em um sentido, ora em outro. Segue-se o mesmo princípio das luzes: para um caractere o motor girará para a esquerda e para outro ele girará para a direita (será explicado sobre essa funcionalidade nos próximos capítulos).

3.1 A INTERFACE DESENVOLVIDA

Visando facilitar a interação com o usuário, foi desenvolvida uma interface gráfica utilizando a linguagem de programação Visual Basic.NET (Figura 6 abaixo).

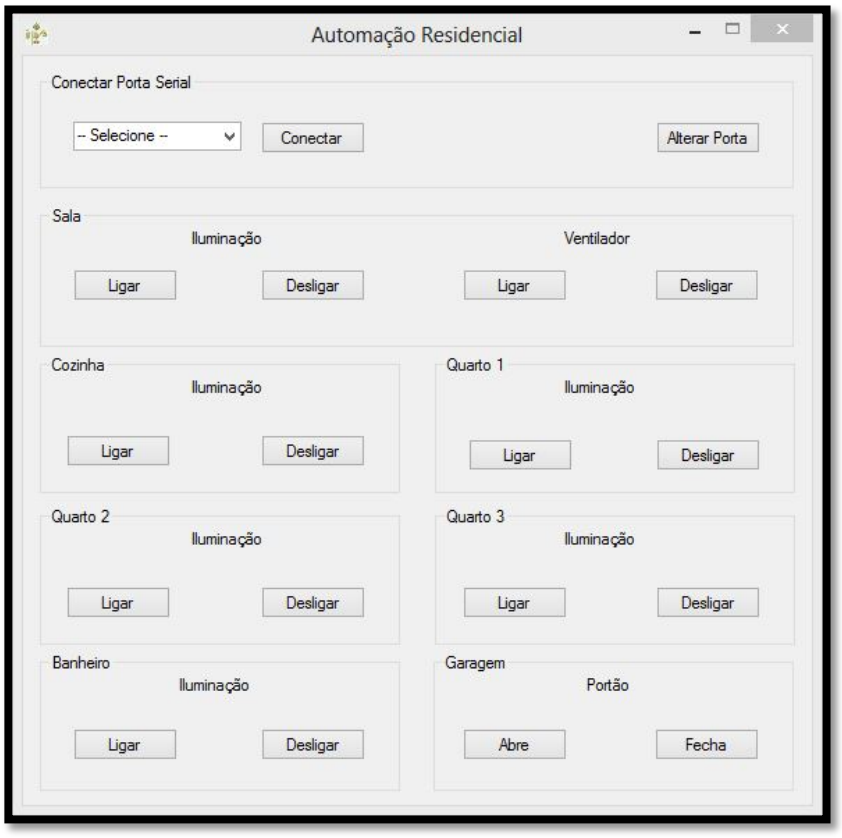

Figura 6 - Tela do Programa

Fonte: Autoria própria

Nessa interface, primeiramente o usuário escolhe a porta e estabelece a conexão. Depois é só clicar no botão da ação desejada. Por exemplo: Se o usuário quer abrir o portão, basta que ele clique no botão Abre que está localizado dentro de um Groupbox denominado Garagem. Com isso, economiza-se tempo, pois o usuário não precisará digitar valores em uma interface, e sim, apenas escolher a ação clicando no respectivo botão. Para desconectar, basta clicar no botão Alterar Porta.

3.1.1 OS CÓDIGOS E A COMUNICAÇÃO COM O ARDUINO

O Visual Basic trabalha com *Subs* que são nada mais do que procedimentos com tarefas a serem executadas pelo usuário.

Neste programa, como já citado acima, a primeira tarefa é conectar a porta *serial* para que possa ser estabelecida a comunicação com o Arduino. Abaixo será mostrado como isso acontece:

```
Imports System.IO.Ports

Public Class frmPrincipal

    Private Sub FrmPrincipal_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        CarregarPortas()

    End Sub
```

Imports refere-se que deve-se importar uma biblioteca que contém as portas utilizadas pelo sistema, que neste caso é a *serial*.

O *sub* FrmPrincipal_Load será executado quando o programa for aberto. Dentro dele, é chamado o *sub* CarregarPortas(), que terá como função listar as portas *seriais* que estiverem disponíveis para a comunicação.

```
    Private Sub CarregarPortas()


ListaPortas.Items.Clear()

ListaPortas.Items.Add("-- Selecione --")
```

```
ListaPortas.SelectedIndex = 0
For Each sp As String In My.Computer.Ports.SerialPortNames ListaPortas.Items.Add(sp)
Next
```

Neste *sub* há um *ComboBox* (já mencionado acima) chamado ListaPortas.

A função *Clear*() é responsável por limpar as portas que estiveram na execução anterior antes de iniciar esta. A função *Add*, adiciona todas as portas de comunicação (Neste caso, havia apenas a COM3) que estiverem disponíveis. *SelectedIndex*, mostra que a palavra selecione, será a que aparecerá por padrão, portanto é o índice 0. A próxima linha executa um loop, ou seja, uma repetição para que as portas sejam sempre mostradas dentro do *ComboBox. ListaPortas.Items.Add (sp)*. Adiciona todas as portas seriais que estiverem disponíveis ao *ComboBox*. Abaixo, serão citados os botões que estão a direita do *ComboBox*, são respectivamente Conectar e Alterar Porta.

```
Private Sub Btn1_Click(sender As Object, e As EventArgs) Handles Btn1.Click
        If ListaPortas.SelectedIndex = 0 Then

            MessageBox.Show("Selecione uma PORTA!")

        Else

            ConectarPorta(ListaPortas.SelectedItem)

        End If

    End Sub
```

O Btn1 corresponde ao botão Conectar e este *sub* diz que quando for clicado, verificará se há alguma porta selecionada no *ComboBox*, se for igual a 0, ou seja, se não tiver nenhuma porta selecionada, será exibida uma caixa de mensagem (*MessageBox*), escrita: “Selecione uma PORTA!”), senão será executada a função ConectarPorta que será responsável por estabelecer a comunicação com o Arduino.

```
Private Sub Btn2_Click(sender As Object, e As EventArgs) Handles Btn2.Click

        ListaPortas.Enabled = True

        ListaPortas.SelectedIndex = 0


        Btn1.Enabled = True

        Btn1.Text = "Conectar"


        Btn2.Enabled = False


        PortSerial.Close()

    End Sub
```

Corresponde ao botão Alterar Porta. Quando este for clicado, reabilitará o *ComboBox* chamado ListaPortas, o botão Conectar e fechará a Porta Serial, permitindo assim que o usuário escolha outra porta de comunicação.

```
Private Sub ConectarPorta(Porta)



        ListaPortas.Enabled = False


        Btn1.Enabled = False

        Btn1.Text = "Conectando..."
```

```
Btn2.Enabled = True
```

Neste trecho desta função, o *ComboBox* que se refere a seleção das portas é desabilitado, assim como o botão conectar e é habilitado o botão de Alterar Porta (Já explicado acima).

```
Try
    If PortSerial.IsOpen Then
       PortSerial.Close()
    End If
```

Este Try como o nome sugere, trata-se de tentar conexão com a porta serial. Se a porta estiver aberta ele fechará a conexão e tentará de novo.

```
    PortSerial.PortName = Porta
    PortSerial.BaudRate = 9600
    PortSerial.Parity = Parity.None
    PortSerial.StopBits = StopBits.One
    PortSerial.DataBits = 8
    PortSerial.Handshake = Handshake.None
    PortSerial.ReadTimeout = 5000
    PortSerial.WriteTimeout = 5000
    PortSerial.Open()
    Btn1.Text = "Conectado"
```

Acima estão os parâmetros, da porta serial *PortName* é o nome que será atribuído a ela, BaudRate é a taxa em bits, no caso 9600, *Parity* é se terá ou não paridade. Paridade é muito útil na verificação de erros, mas nesse caso não está empregada. *StopBits* e DataBits referem-se a quantidade de bits que será utilizada, no caso 8 bits. As funções Timeout referem-se ao tempo de conexão em

milisegundos, no caso 5000 milisegundos. Ao final de tudo isso, a conexão estará aberta.

```
        Catch ex As Exception
            ListaPortas.Enabled = True
            ListaPortas.SelectedIndex = 0
            Btn1.Enabled = True
            Btn1.Text = "Conectar"
            Btn2.Enabled = False
        End Try
    End Sub
```

Caso o *Try* não seja satisfeito, este *Catch* será executado. De acordo com o código ele voltará e permitirá ao usuário tentar conectar novamente.

Feita a conexão com a porta *serial*, os demais botões do programa farão a função que estiver denominada neles.

Por exemplo:

```
Private Sub btnLigaQuarto1_Click(sender As Object, e As EventArgs) Handles btnLigaQuarto1.Click
        PortSerial.Write("2")
    End Sub
```

No botão LigaQuarto1, será ligado o LED correspondente a este quarto. A função *Write* escreverá o caractere “2” e enviará para o Arduino pela Porta *Serial*. Ao receber este comando, o Arduino ligará o LED.

O botão DesligaQuarto1 envia o caractere “3” pela Porta *Serial*. Ao receber este caractere, o Arduino desliga o LED.

A tabela abaixo especifica qual caractere cada botão envia, e o que o Arduino faz ao receber este comando:

Botão Ligar Iluminação Sala	Envia o caractere “1” via Porta *Serial* para o Arduino.	Arduino liga o LED correspondente à sala.
Botão Desligar Iluminação Sala	Envia o caractere “0” via Porta *Serial* para o Arduino.	Arduino desliga o LED correspondente à sala.
Botão Liga Ventilador Sala	Envia o caractere “v” via Porta *Serial* para o Arduino.	Arduino liga o ventilador da sala.
Botão Desliga Ventilador Sala	Envia o caractere “b” via Porta *Serial* para o Arduino.	Arduino desliga o ventilador da sala.
Botão Ligar Iluminação Cozinha	Envia o caractere “6” via Porta *Serial* para o Arduino.	Arduino liga o LED correspondente à cozinha.
Botão Desligar Iluminação Cozinha	Envia o caractere “7” via Porta *Serial* para o Arduino.	Arduino desliga o LED correspondente à cozinha.
Botão Ligar Iluminação Quarto 1	Envia o caractere “2” via Porta *Serial* para o Arduino.	Arduino liga o LED correspondente ao Quarto 1.
Botão Desligar Iluminação Quarto 1	Envia o caractere “3” via Porta *Serial* para o Arduino.	Arduino desliga o LED correspondente ao Quarto 1.
Botão Ligar Iluminação Quarto 2	Envia o caractere “4” via Porta *Serial* para o Arduino.	Arduino Liga o LED correspondente ao Quarto 2.
Botão Desligar Iluminação Quarto 2	Envia o caractere “5” via Porta *Serial* para o Arduino.	Arduino desliga o LED correspondente ao Quarto 2.
Botão Ligar Iluminação Quarto 3	Envia o caractere “8” via Porta *Serial* para o Arduino.	Arduino Liga o LED correspondente ao Quarto 3.
Botão Desligar Iluminação Quarto 3	Envia o caractere “9” via Porta *Serial* para o Arduino.	Arduino desliga o LED correspondente ao Quarto 3.
Botão Ligar Iluminação	Envia o caractere “q” via	Arduino liga o LED

Banheiro	Porta *Serial* para o Arduino.	correspondente ao Banheiro.
Botão Desligar Iluminação Banheiro	Envia o caractere “w” via Porta *Serial* para o Arduino.	Arduino desliga o LED correspondente ao Banheiro.
Botão Abrir Portão Garagem	Envia o caractere “e” via Porta *Serial* para o Arduino.	Arduino, envia sinais para a Ponte H que faz o motor girar para a esquerda, fazendo com que o portão abra.
Botão Fechar Portão Garagem	Envia o caractere “d” via Porta *Serial* para o Arduino.	Arduino, envia sinais para a Ponte H que faz o motor girar para a direita, fazendo com que o portão feche.

Tabela 1 - Caracteres que são enviados para o Arduino

O código completo deste projeto pode ser encontrado em Anexo 1 no final deste trabalho.

3.2 A PARTE ELETRÔNICA

Abaixo, serão descritos todos os componentes eletrônicos utilizados e a função que desempenharam no projeto.

3.2.1 DIODO

O diodo (figura 7 abaixo) é um componente eletrônico que tem como função permitir que a corrente circule apenas em um sentido, mas para isso ele precisa estar polarizado. Ou seja, precisa-se ter uma tensão de pelo menos 0,7 V aplicada sobre ele.

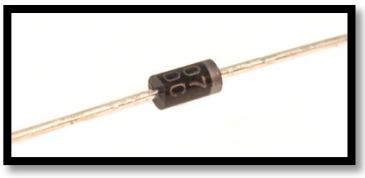

Figura 7 - Diodo 1N4007

Fonte: http://loja.multcomercial.com.br/ecommerce_site/arquivos4689/arquivos/1351010257_1.jpg

Neste projeto, foi utilizado um diodo exatamente igual ao da figura acima, no controle do motor. Pois quando se inverte, por exemplo, o sentido de rotação, ele gera um alto pico de tensão que pode danificar o Arduino. Portanto, o diodo é o responsável por realizar o bloqueio dessa tensão reversa e impedir de circular esta alta corrente.

3.2.2 TRANSISTOR

O *Transistor* (figura 8 abaixo) é um componente semelhante ao diodo, porém possui três terminais ao invés de dois. O terceiro terminal é denominado Base ou Porta (Dependendo do tipo de Transistor escolhido). Neste projeto, foi utilizado o TIP122, que consiste num transistor com base. Ou seja, através de uma corrente que circula por esse terminal que se determina se um dispositivo (Neste caso, o motor que equivale ao ventilador), será ou não acionado. Observação: Apenas se usa para controlar cargas de corrente contínua. Para cargas de corrente alternada se faz uso de outros componentes específicos.

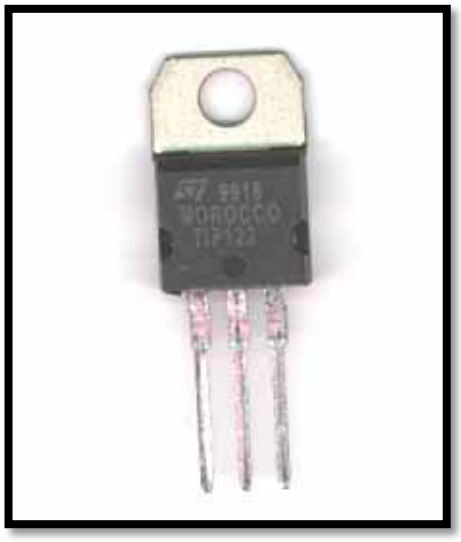

Figura 8 - Transistor TIP122

Fonte: http://www.lakesidepinballparts.com/tip122.jpg

3.2.3 RESISTOR

O resistor (figura 9 abaixo) é um componente eletrônico que tem como finalidade limitar uma tensão elétrica e controlar uma corrente. Conforme o nome sugere, ele “resiste” à passagem da corrente, ou seja, faz oposição. Neste projeto foram implementados resistores para controlar a corrente dos LEDs e também abaixar a tensão, já que o Arduino nos fornece 5V e os LEDs são de 2,1 V (Amarelo) e 3 V (Amarelo Auto Brilho).

Figura 9 – Resistor

Fonte: http://www.infoescola.com/wp-content/uploads/2010/02/resistor-eletronica.jpg

3.2.4 LED

Por se tratar apenas de uma demonstração, foi optado por se usar LEDs (*Light Emitting Diode* ou Diodo Emissor de Luz). Como o nome sugere, o LED também é um diodo, porém quando está corretamente colocado no circuito, este emite uma luz.

Figura 10 – LED

Fonte: http://www.cyberconductor.com/images/5mm20yellow20led.jpg

3.2.5 CAPACITOR

O capacitor é um componente eletrônico que tem a função de acumular cargas elétricas, ou seja, depois de alimentado ele vai carregando até a sua capacitância máxima. Neste projeto foi utlizado um capacitor entre o motor e o Arduino, para filtrar a corrente e evitar que ela queime o Arduino e também para evitar interferências geradas pelo motor em outros equipamentos próximos, quando a rotação deste for invertida.

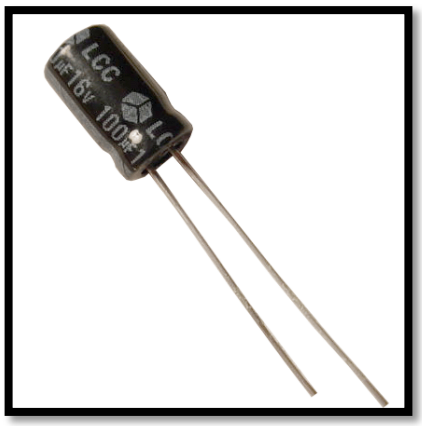

Figura 11 – Capacitor

Fonte: http://ww2.justanswer.com/uploads/teccar/2011-02-08_002532_capacitor.gif

3.2.6 MOTORES DE CORRENTE CONTÍNUA

O motor de corrente contínua pode ser entendido como uma bobina montada entre dois ímãs , esta bobina é móvel e é chamada de rotor. Ao circular uma corrente geram-se forças de repulsão entre por exemplo, o polo norte do rotor e o polo norte de um dos ímãs, esta força faz com que ele dê uma meia volta para ser atraído pelo polo sul do outro ímã. Todavia, existe um componente chamado de comutador, que tem como finalidade inverter os polós dos ímãs, gerando a repulsão novamente, mais meia volta e ela pararia, pois seria atraído pelo polo do outro ímã, porém o comutador é acionado novamente, fazendo com que ele sempre gire e não pare até que a corrente elétrica seja cortada.

Portanto, a força dependerá da tensão aplicada que determina a corrente que circula e a força do campo magnético que é gerado.

A velocidade é determinada pela força. Geralmente os fabricantes determinam uma faixa de tensão específica de acordo com a força desejada.

Neste projeto, foram utilizados dois motores. Um para simular um portão e outro para simular um ventilador.

Para que o motor do portão pudesse abrir ou fechar, foi utilizado um circuito integrado denominado de Ponte H. Que pode ser entendida como quatro chaves que abrem ou fecham. (Figura 12 abaixo).

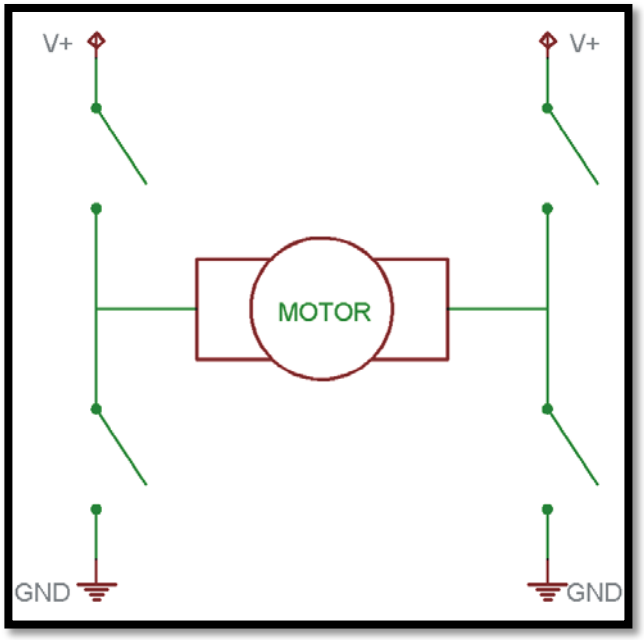

Figura 12 - Ponte H

Fonte: http://robolivre.org/uploads/thumbnails/_avatar_con_ponte-h1361045205_large.png

Para que o motor funcione, a chave V+ da esquerda superior e a segunda GND direita inferior, precisam estar fechadas. Ou a segunda V+ direita superior com a primeira esquerda inferior. Em cada uma dessas duas combinações o motor gira para um lado.

Foi utilizado um circuito integrado que trabalha com lógica digital, isso evita possíveis curtos-circuitos, pois numa lógica digital, só funciona se as condições forem verdadeiras. Caso contrário, o motor apenas não funcionará.

3.3 A PROGRAMAÇÃO DO ARDUINO

Abaixo será mostrado o código com as respectivas explicações:

```
int ledcozinha = 6;

int ledquarto1 = 8;

int ledquarto2 = 11;

int ledsala = 12;

int ledquarto3 = 13;
```

```
int ledbanheiro = 7;

int motorPin1 = 4;

int motorPin2 = 5;

int Vent = 9;

int armazena = 0;
```

Neste trecho foram declaradas as variáveis que serão usadas para representar os pinos do Arduino. Int significa que a variável é do tipo inteira, o valor atribuído corresponde ao pino respectivo do Arduino. A variável armazena é utilizada para guardar o caractere digitado pelo usuário que dependendo de qual for realizará a função desejada.

```
void setup (){
  Serial.begin(9600);
  pinMode (ledcozinha, OUTPUT);
  pinMode (ledquarto1, OUTPUT);
  pinMode (ledquarto2, OUTPUT);
  pinMode (ledsala, OUTPUT);
  pinMode (ledquarto3, OUTPUT);
  pinMode (motorPin1, OUTPUT);
  pinMode (motorPin2, OUTPUT);
}
```

Nesta função void setup, todos os pinos estão como *OUTPUT*, ou seja, configurados como saída. (Mais detalhes nos próximos trechos).

A função Serial.begin, define que a porta *serial* será utilizada. E o número indica a taxa de transmissão em *bits*, no caso, 9600 *bits*.

```
void loop(){
    if (Serial.available()){
```

```
armazena = Serial.read();
```

A função void loop como o nome sugere, é um *loop*, ou seja esta função será sempre repetida enquanto o programa estiver em execução.

A variável armazena vai receber o caractere digitado pelo usuário, que será transmitido do computador para a porta *serial*, o Arduino fará a leitura e executará o que o usuário deseja.

```
if (armazena==54){
  digitalWrite (ledcozinha,HIGH);
}
```

O Arduino trabalha com a tabela ASCII *(American Standard Code for Information Interchange* ou Código Americano Padrão para troca de informações). Ou seja, cada caractere digitado no teclado possui um código de 8 *bits* nessa tabela. No entanto, este valor está em decimal. Portanto o valor 54 corresponde ao caractere 6.

A função digitalWrite quer dizer que o Arduino enviará um sinal digital para a porta especificada, no caso a ledcozinha, que conforme citado no primeiro trecho, corresponde ao pino de número 6. *HIGH* significa que este sinal terá nível alto, ou seja 5V.

```
if (armazena==55){
digitalWrite (ledcozinha,LOW);
}
```

Nesta outra, o nível está como *LOW*, ou seja, ao inserir o caractere 7 (equivalente a 55 na tabela ASCII), o Arduino enviará 0V para o pino 6 e desligará o LED.

```
if(armazena=='e'){
  digitalWrite(motorPin1, 1);
  digitalWrite(motorPin2, 0);
  }
  else if(armazena=='d'){
  digitalWrite(motorPin1, 0);
  digitalWrite(motorPin2, 1);
}
else {
  digitalWrite(motorPin1, 0);
  digitalWrite(motorPin2, 0);
}
```

Conforme já mencionado, a ponte H escolhida para este projeto trabalha com uma lógica digital, neste trecho acima foi seguida esta lógica: Se o usuário inserir a letra “e”, será enviado 1 (5 V) ao pino 4 do Arduino que enviará ao respectivo pino de saída da ponte H, o mesmo com o pino 5 (motorPin2), porém a este será enviado 0 (0 V), isso fará o motor girar para a esquerda.

Mas, se o usuário inserir “d”, será enviado 0 (0 V) ao motorPin1 (pino 4), e 1 (5 V) ao motorPin2 (pino 5) fazendo assim, o motor girar para a direita.

Caso o usuário não insira nada, o motor ficará parado, pois serão enviados sinais digitais de 0 a ambos os pinos.

```
if (armazena=='v'){
  analogWrite(Vent,255);
}
```

```
if (armazena=='b'){
  analogWrite(Vent, 0);
}
}
```

Este trecho refere-se ao ventilador. Por se tratar de um motor, o Arduino deve enviar um sinal analógico, para que este seja acionado. Inserindo “v”, envia-se 255 que faz com que o motor gire na velocidade máxima. Ao inserir “b”, envia-se 0 que faz com que o motor pare. Valores intermediários podem ser inserios para um controle de velocidade, porém devido as limitações do motor escolhido, este controle não foi possível de ser feito.

O código completo deste projeto pode ser encontrado em Anexo 2 no final deste trabalho.

4. CONCLUSÃO

Através das experiências obtidas neste projeto, pode-se afirmar que com um baixo custo e com alguma criatividade, pode-se desenvolver algo que além de simples, de fácil uso, permite e contribui para agilizar o dia-a-dia de todos. Além disso, é muito gratificante pensar que foi tudo elaborado e construído pelo próprio usuário e para o usuário. O Arduino realmente, permite utlizar-se tão mais da criatividade do que pensar apenas em altos custos e grandes conhecimentos que levam tempo para serem adquiridos e assimilados totalmente. Todo o projeto de código-aberto tem essa finalidade: Ser para todos. Independente se a pessoa for rica, pobre, com pouco conhecimento, basta ter dedicação e empenho. Além disso, o melhor é saber que há uma comunidade que realmente se preocupa em sanar as dúvidas e oferecer o melhor de tudo: Conhecimento. Quando este é compartilhado, realmente se criam de tudo, mesmo que seja baseado em algo já existente. Com a partilha do conhecimento, todos podem, todos vão criar as suas soluções e o mundo realmente será de todos e para todos.

5. REFERÊNCIAS

ALMEIDA, Eduardo. **Motores DC.** Disponível em: http://www.arduino-projetos.com.br/2011/06/motores-dc.html/. Acesso em 20 de Maio de 2013.

BRAGA, Newton C. **Motores de Correte Contínua. – Revista Eletrônica Total – Edição Nº 147.** p. 30-33. – São Paulo - SP – Editora Saber, 2011.

MCROBERTS, Michael. **Arduino Básico.** – São Paulo – SP – Editora Novatec, 2011

PEREIRA, Felipe. **Curso sobre Arduino Parte 1. – Revista Saber Eletrônica – Edição Nº 454.** p. 12-15. – São Paulo - SP – Editora Saber, 2011.

_____. **Curso sobre Arduino Parte 2. – Revista Saber Eletrônica – Edição Nº 455.** p. 22-26. – São Paulo - SP – Editora Saber, 2011.

RIOS, Jefferson et al. **Introdução ao Arduino**. 2012. Faculdade de Computação – Universidade Federal de Mato Grosso do Sul, Mato Grosso do Sul, 2012.

SEAKER, The. **Controll Leds With an Arduino From a C# Program**. Disponível em: http://www.instructables.com/id/Controll-Leds-With-an-Arduino-From-a-C-Program/. Acesso em 20 de Maio de 2013.

6. ANEXOS

Nesta parte do trabalho será apresentado o anexo de todos os códigos-fonte utilizados nas aplicações desenvolvidas.

6.1 ANEXO 1 – CÓDIGO DA APLICAÇÃO

```
Imports System.IO.Ports

Public Class frmPrincipal

    Private Sub FrmPrincipal_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        CarregarPortas()

    End Sub

    Private Sub CarregarPortas()

        ListaPortas.Items.Clear()

        ListaPortas.Items.Add("-- Selecione --")

        ListaPortas.SelectedIndex = 0


        For Each sp As String In My.Computer.Ports.SerialPortNames

            ListaPortas.Items.Add(sp)

        Next

    End Sub

    Private Sub ConectarPorta(Porta)

        ListaPortas.Enabled = False

        Btn1.Enabled = False

        Btn1.Text = "Conectando..."

        Btn2.Enabled = True

        Try
```

```
            If PortSerial.IsOpen Then
                PortSerial.Close()
            End If
            PortSerial.PortName = Porta
            PortSerial.BaudRate = 9600
            PortSerial.Parity = Parity.None
            PortSerial.StopBits = StopBits.One
            PortSerial.DataBits = 8
            PortSerial.Handshake = Handshake.None
            PortSerial.ReadTimeout = 5000
            PortSerial.WriteTimeout = 5000
            PortSerial.Open()
            Btn1.Text = "Conectado"
        Catch ex As Exception
            ListaPortas.Enabled = True
            ListaPortas.SelectedIndex = 0
            Btn1.Enabled = True
            Btn1.Text = "Conectar"
            Btn2.Enabled = False
        End Try
    End Sub
    Private Sub btnLigarSala_Click(sender As Object, e As EventArgs)
Handles btnLigarSala.Click
        PortSerial.Write("1")
    End Sub
```

```
        Private Sub btnDesligarSala_Click(sender As Object, e As EventArgs) Handles btnDesligarSala.Click
            PortSerial.Write("0")
        End Sub
        Private Sub btnLigaCozinha_Click(sender As Object, e As EventArgs) Handles btnLigaCozinha.Click
            PortSerial.Write("6")
        End Sub
        Private Sub btnDesligaCozinha_Click(sender As Object, e As EventArgs) Handles btnDesligaCozinha.Click
            PortSerial.Write("7")
        End Sub
       Private Sub Btn1_Click(sender As Object, e As EventArgs) Handles Btn1.Click
            If ListaPortas.SelectedIndex = 0 Then
                MessageBox.Show("Selecione uma PORTA!")
            Else
              ConectarPorta(ListaPortas.SelectedItem)
            End If
        End Sub
        Private Sub Btn2_Click(sender As Object, e As EventArgs) Handles Btn2.Click
            ListaPortas.Enabled = True
            ListaPortas.SelectedIndex = 0
            Btn1.Enabled = True
            Btn1.Text = "Conectar"
```

```
        Btn2.Enabled = False
        PortSerial.Close()
    End Sub
    Private Sub btnLigaQuarto1_Click(sender As Object, e As EventArgs) Handles btnLigaQuarto1.Click
        PortSerial.Write("2")
    End Sub
    Private Sub btnDesligaQuarto1_Click(sender As Object, e As EventArgs) Handles btnDesligaQuarto1.Click
        PortSerial.Write("3")
    End Sub
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles btnLigaQuarto3.Click
        PortSerial.Write("8")
    End Sub
    Private Sub btnLigaQuarto2_Click(sender As Object, e As EventArgs) Handles btnLigaQuarto2.Click
        PortSerial.Write("4")
    End Sub
    Private Sub btnDesligaQuarto2_Click(sender As Object, e As EventArgs) Handles btnDesligaQuarto2.Click
        PortSerial.Write("5")
    End Sub
    Private Sub btnDesligaQuarto3_Click(sender As Object, e As EventArgs) Handles btnDesligaQuarto3.Click
        PortSerial.Write("9")
```

```
        End Sub

        Private Sub btnLigaBanheiro_Click(sender As Object, e As EventArgs) Handles btnLigaBanheiro.Click

            PortSerial.Write("q")

        End Sub

        Private Sub btnDesligaBanheiro_Click(sender As Object, e As EventArgs) Handles btnDesligaBanheiro.Click

            PortSerial.Write("w")

        End Sub

        Private Sub btnAbrir_Click(sender As Object, e As EventArgs) Handles btnAbrir.Click

            PortSerial.Write("e")

        End Sub

        Private Sub btnFechar_Click(sender As Object, e As EventArgs) Handles btnFechar.Click

            PortSerial.Write("d")

        End Sub

        Private Sub btnLigaVent_Click(sender As Object, e As EventArgs) Handles btnLigaVent.Click

            PortSerial.Write("v")

        End Sub

        Private Sub btnDesligaVent_Click(sender As Object, e As EventArgs) Handles btnDesligaVent.Click

            PortSerial.Write("b")

        End Sub

    End Class
```

6.2 ANEXO 2 – CÓDIGO DO ARDUINO

```
int ledcozinha = 6;
int ledquarto1 = 8;
int ledquarto2 = 11;
int ledsala = 12;
int ledquarto3 = 13;
int ledbanheiro = 7;
int motorPin1 = 4;
int motorPin2 = 5;
int armazena = 0;
int Vent = 9;
void setup (){
  Serial.begin(9600);
  pinMode (ledcozinha, OUTPUT);
  pinMode (ledquarto1, OUTPUT);
  pinMode (ledquarto2, OUTPUT);
  pinMode (ledsala, OUTPUT);
  pinMode (ledquarto3, OUTPUT);
  pinMode (ledbanheiro, OUTPUT);
  pinMode (motorPin1, OUTPUT);
  pinMode (motorPin2, OUTPUT);
}
void loop(){

  if (Serial.available()){
    armazena = Serial.read();

    if (armazena==54){
      digitalWrite (ledcozinha,HIGH);
    }
    if (armazena==55){
    digitalWrite (ledcozinha,LOW);
```

```
    }
    if (armazena==50){
    digitalWrite (ledquarto1,HIGH);
  }
  if (armazena==51){
   digitalWrite (ledquarto1,LOW);
}
  if(armazena==52){
  digitalWrite(ledquarto2,HIGH);
  }
  if(armazena==53){
  digitalWrite(ledquarto2,LOW);
  }
  if(armazena==56){
    digitalWrite(ledquarto3,HIGH);
  }
  if(armazena==57){
    digitalWrite(ledquarto3,LOW);
}
  if(armazena==49){
    digitalWrite(ledsala, HIGH);
  }
  if(armazena==48){
    digitalWrite(ledsala, LOW);
  }
if(armazena=='e'){
  digitalWrite(motorPin1, 1);
  digitalWrite(motorPin2, 0);
  }
  else if(armazena=='d'){
  digitalWrite(motorPin1, 0);
  digitalWrite(motorPin2, 1);
```

```
}
else {
  digitalWrite(motorPin1, 0);
  digitalWrite(motorPin2, 0);
}
if (armazena=='q'){
  digitalWrite(ledbanheiro, HIGH);
   }
 if(armazena=='w'){
   digitalWrite(ledbanheiro, LOW);
}
  }
if (armazena=='v'){
  analogWrite(Vent,255);
}
if (armazena=='b'){
  analogWrite(Vent, 0);
}
}
```