UNICEUB – CENTRO UNIVERSITÁRIO DE BRASÍLIA

FATECS – FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS

CURSO DE ENGENHARIA DE COMPUTAÇÃO

GUSTAVO CAETANO DE ALMEIDA

SISTEMA CONTROLADOR DE ILUMINAÇÃO DE AMBIENTES ATRAVÉS DE INTERFACE COMPUTADORIZADA

BRASÍLIA / DF

2º SEMESTRE DE 2010

GUSTAVO CAETANO DE ALMEIDA

SISTEMA CONTROLADOR DE ILUMINAÇÃO DE AMBIENTES ATRAVÉS DE INTERFACE COMPUTADORIZADA

Trabalho apresentado ao Centro Universitário de Brasília como pré-requisito para a obtenção de Certificado de Conclusão do Curso de Engenharia de Computação.

Orientador: José Julimá Bezerra Júnior

BRASÍLIA / DF

2º SEMESTRE 2010

GUSTAVO CAETANO DE ALMEIDA

SISTEMA CONTROLADOR DE ILUMINAÇÃO DE AMBIENTES ATRAVÉS DE INTERFACE COMPUTADORIZADA

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.
Orientador: Prof. José Julimá Bezerra Júnior

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiezer Amarilia Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Prof. José Julimá, Mestre em Engenharia Elétrica -Sistemas de controle.
Orientador

_____

Prof. Antonio Barbosa, Especialista em Engenharia de Software.
UniCEUB

_____

Prof. Luis Cláudio, Mestre em Matemática Pura.
UniCEUB

_____

Prof. Cléber da Silva Pinheiro, Doutor em Física.
UniCEUB

RESUMO

Trata-se de um sistema de automação de iluminação para qualquer tipo de ambiente, tendo como meta reduzir o desperdício de energia elétrica e facilitar o controle da iluminação de locais distantes, que não precise de pessoas o tempo todo. Após implementar um sistema envolvendo um circuito inteligente conectado ao computador, o usuário pode controlar áreas ou individualmente cada ponto de energia elétrica em um ambiente determinado. O objetivo é propor uma maneira eficaz de controle da iluminação de ambientes, usando para isso uma interface que gere a integração e interação do gestor com todos os pontos de luminosidade do ambiente. É utilizado *hardware* disponível no mercado e já comum a todas as residências e empresas (computador) conectado à plataforma Arduíno (que utiliza microcontrolador Atmel Atmega328), circuitos controladores, ligado pela porta USB à central de comutação (PC), que por sua vez tem uma interface simplificada para execução dos comandos feita em *software* construído na linguagem de programação “PHP”.

Palavras-chave: automação, Arduíno e iluminação.

ABSTRACT

This is an automation system for lighting any type of environment, aiming to reduce the waste of electricity and facilitate the control of light from distant places that does not need people all the time. After implementing a system involving an intelligent circuit connected to the computer, the user can control areas or each individual point of electrical energy in a given environment. The aim is to propose an effective way to control lighting environments, using an interface that manages the integration and interaction with all points of the ambient brightness. It's used commercially available hardware and easily available in residences and businesses (computer) connected to the Arduino platform (which uses Atmel Atmega328) circuit controllers, connected to the USB port commuting center (PC) and software built on the programming language "PHP".

Words Keys: Automation, Arduino and lighting.

Este projeto e monografia são dedicados especialmente, aos meus pais Orlando e Léia, minha irmã Karine e cunhado Flávio, meus amigos, Fábio Lima, Rodrigo Isidro e Bruno Matos, aos meus colegas de trabalho, ao meu orientador, José Julimá e a todos aqueles que tiveram paciência e me incentivaram durante todo o processo de elaboração deste projeto.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

AMP OP	Amplificador Operacional
AREF	*Analog Reference* - Pino de referência de tensão para entradas analógicas
ASP	*Active Server Pages*
AVR	Microcontrolador RISC de *chip* único
BPS	Bits Por Segundo
C	Linguagem de programação estruturada;
C++	Linguagem de programação orientada a objeto
CAD	*Computer-aided design*
CI	Circuito Integrado
COM	Porta de comunicação responsável pela entrada e saída de dados
DC	*Direct Current*
DIP	*Dual In-Line Package*
DTR	*Data Terminal Ready*
E/S	Entrada e Saída de dados
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
FIFO	*First In, First Out*
FTDI	*Future Technology Devices International*
GND	*Ground*
GNU	*GNU's Not Unix*
GPL	*General Public License*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
I2C	*Inter-Intergrated Circuit*
ICSP	*In-Circuit System Programming*
IDE	*Integrated Development Environment*
ISP	*In-System-Programming*
JFET	*Junction Field Effect Transistor*
LDR	*Light Dependent Resistor*
LGPL	*Library General Public License*
LED	*Light Emitting Diode*
MISO	*Master In Slave Out*
MOSFET	*Metal Oxide Semiconductor Field Effect Transistor*
MOSI	*Master Out Slave In*
NPN	Ligação lógica negativa
OSI	*Open Systems Interconnection*
PC	*Personal Computer*
PHP	*PHP: Hypertext Preprocessor*
PIR	*Passive InfraRed Sensor*
PIC	*Programmable Interface Controller*
PNP	Ligação lógica positiva
PPM	*Pulse Position Modulation*
PWM	*Pulse-Width Modulation*
RAM	*Random Access Memory*
RISC	*Reduced Instruction Set Computer*
SCK	*Serial Clock*
SCL	Entrada analógica do Arduíno.
SDA	*Serial Data Line*
SMD	*Surface Mountaing Device*
SMS	*Short Message Service*

SPI	*Serial Peripheral Interface*
SQL	*Structured Query Language*
SS	*Slave Select*
STK500	Protocolo de comunicação
TCP	*Transmission Control Protocol*
TTL	*Time To Live*
TWI	*Two-Wire Interface*
UART	*Universal Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*

UNIDADES DE MEDIDA

A	Ampère, unidade de corrente elétrica
KB	*KiloByte*
Hz	Hertz, unidade de freqüência
Ω	Ohm, unidade de resistência elétrica
V	Volts, unidade de tensão elétrica
W	*Watt*, unidade de potência

CAPÍTULO 1 – INTRODUÇÃO

1.1. – APRESENTAÇÃO DO PROBLEMA

Visando melhorar e racionalizar o consumo elétrico por residências e empresas, observa-se o crescimento da demanda por automação do sistema de iluminação de forma computadorizada. É possível encontrar diversas empresas atuando neste segmento. A maioria dessas empresas oferece solução completa de automação residencial/empresarial indo além do sistema de iluminação, como por exemplo, automatizando a abertura/fechamento de portas e janelas, sistemas de entretenimento (TV, *Home Theater* etc.), sistemas de irrigação de jardins, monitoramento de pacientes etc.

Segundo Couto de Moraes e Castrucci, em seu livro, intitulado “Engenharia de Automação Industrial”, a palavra *automation* foi inventada pelo *marketing* da indústria de equipamentos na década de 1960. O neologismo, sem dúvida, sonoro, buscava enfatizar a participação do computador no controle automático industrial. O que significa automação? Hoje entende-se por automação qualquer sistema, apoiado em computadores, que substitua o trabalho humano e que vise soluções rápidas e econômicas para atingir os complexos objetivos das indústrias e dos serviços (por exemplo, automação industrial, automação bancária). A automação implica na implantação de sistemas interligados e assistidos por redes de comunicação, compreendendo sistemas supervisórios e interfaces homem-máquina que possam auxiliar os operadores no exercício de supervisão e análise dos problemas que porventura venham a ocorrer. A automação na indústria decorre de necessidades tais como: maiores níveis de qualidade de conformação e de flexibilidade, menores custos de trabalho, menores perdas materiais e menores custos de capital; maior controle das informações

relativas ao processo, maior qualidade das informações e melhor planejamento e controle da produção. [11].

Este trabalho utiliza sensor de iluminação e sensor de presença para identificar o grau de iluminação do ambiente e a presença de pessoas no local. Dessa forma, torna-se possível programar as atividades do sistema de forma que as ações ocorram de forma automática, sem necessidade de intervenção humana. Os sensores estarão presentes em todos os ambientes em que o controle de iluminação é desejável.

Imagina-se um futuro onde os humanos habitarão casas inteligentes e, certamente, este tipo de sistema de controle de iluminação será item obrigatório para aumentar o conforto das famílias.

Como este projeto depende essencialmente das variações de iluminação ao longo do dia, é muito importante fazer uso de um sensor de iluminação de qualidade para não comprometer o projeto.

1.2. – MOTIVAÇÃO

O tema “automação” está inserido na sociedade há aproximadamente 50 anos e com o avanço da tecnologia e a consequente miniaturização e redução do custo dos componentes, esse tema passou a ser mais difundido e estudado por entusiastas, profissionais e acadêmicos.

A Associação Brasileira de Automação Residencial, em um artigo sobre “Projeto de Interiores e Automação”, afirma que a automação residencial promove a integração e racionalização dos diversos sistemas existentes em uma residência, relacionados à

comunicação, transmissão de dados, iluminação, climatização, segurança, áudio e vídeo, irrigação de jardim etc, gerando como benefícios: economia, conforto e segurança.

Possibilita ainda uma flexibilidade muito grande com relação à múltipla função de uma simples tomada, que pode ser para telefone e num momento seguinte funcionar como ponto de rede, sem a necessidade de passar novos cabos. [07]

Analisando estritamente a automação residencial no que tange à iluminação, Bolzani, em seu livro “Residências Inteligentes” diz que através da automação residencial, toda a iluminação de uma casa pode ser controlada além do interruptor convencional de parede. Sistemas inteligentes podem acentuar os detalhes arquitetônicos de uma sala ou criar ambientes especiais para a utilização do *home-theater* ou para a leitura de um livro, por exemplo. Economia de eletricidade é outra vantagem, pois a intensidade de luz é regulada conforme a necessidade e as lâmpadas não precisam operar com seus brilhos máximos como acontece normalmente. [08]

O curso de Engenharia de Computação possui diversas disciplinas que tratam de assuntos correlatos à automação e isso faz com que o aluno se sinta compelido a acompanhar as inovações na área. Dentre as inovações tecnológicas mais recentes nos últimos 5 anos, é possível afirmar que a plataforma de computação física microcontrolada Arduíno certamente esteja entre as mais revolucionárias, devido ao seu alto grau de facilidade de aprendizado, implementação e disponibilidade de recursos, que permite que leigos, entusiastas e profissionais gabaritados desenvolvam os mais variados dispositivos de automação.

Este projeto tem o claro propósito de controlar a iluminação de um ambiente através de um sistema informatizado e, para isto, serão utilizadas diversas ferramentas *open source*. A

predileção por este tipo de ferramentas para a execução deste projeto está diretamente relacionada à filosofia *open source*, que prega a disponibilização de recursos (*software* ou *hardware*) para a comunidade internacional a fim de que os mesmos possam estar em contínua evolução, independentemente de seu criador abandonar o projeto.

1.3. – OBJETIVOS DO TRABALHO

Este projeto tem como objetivo desempenhar estudos de *hardware* e *software* para desenvolver e implementar um controle automatizado de iluminação de ambientes através de interface computadorizada.

Para realizar este projeto, é necessário fazer uso de um dispositivo físico inteligente, que, no caso, é um ambiente de desenvolvimento integrado de codinome Arduíno em conjunto com sensores de presença, sensores de luminosidade, computador e ambiente gráfico computadorizado desenvolvido em linguagem de programação PHP (*PHP: Hypertext Preprocessor*).

Para maior comodidade do usuário, o mesmo poderá acender/apagar a luminária que desejar manualmente, bastando, para isso, proceder como é feito hoje: pressionando o interruptor de energia. Através da interface computadorizada, também é permitido configurar se os sensores de presença e sensores de iluminação deverão estar ativos ou não.

De forma a interpretar os comandos de acender/apagar a luminária e transformá-los em informações inteligíveis capazes de executar a tarefa, é utilizado um microcontrolador da família Atmel – o Atmega328. Esse microcontrolador também é responsável por enviar o *feedback* para o computador, a fim de exibir a informação de se o comando foi bem executado e se a(s) luminária(s) estão acesas ou apagadas.

1.4. – JUSTIFICATIVA E RELEVÂNCIA DO TEMA

O projeto em questão abrange diversas disciplinas ministradas ao longo do curso de Engenharia de Computação e algumas são ligadas diretamente ao tema. São elas: Microcontroladores e Microprocessadores, Circuitos Eletrônicos, Arquitetura de Computadores, Linguagem e Técnicas de Programação e Banco de Dados.

Trata-se de um tema atual e comum ao meio tecnológico. Segundo a AFRAC (Associação Brasileira de Automação Comercial), de uma forma geral, indústrias do segmento tecnológico crescem de forma contínua e o setor de automação segue a mesma tendência.

O setor de automação comercial movimenta uma média de R$ 1,4 bilhão por ano no país, incluindo mais de dois milhões de estabelecimentos que, juntos, empregam 25 milhões de pessoas. [02]

Os números são da AFRAC. Segundo previsão da entidade, o segmento deverá crescer 10% em 2010. [02]

Com o avanço da tecnologia, as facilidades estão à mão de todos, fazendo com que a concorrência nesta área da Engenharia seja cada vez maior, onde não só as grandes empresas têm a possibilidade de elaborar projetos de automação, mas qualquer pessoa que tenha o conhecimento das tecnologias já existentes no mercado.

1.5. – ESCOPO DO TRABALHO

O objetivo deste trabalho é controlar um sistema de iluminação de um ambiente à distância através de uma interface computadorizada. O controle possibilitará ao usuário o

acionamento manual ou automático das luminárias do ambiente controlado. Para isto, é utilizado um sensor de luminosidade conectado eletronicamente a uma plataforma microcontrolada e, conforme a escolha do usuário, as luminárias podem se apagar ou acender automaticamente conforme o nível de luminosidade no ambiente e/ou podem se apagar ou acender, obedecendo a horários pré-programados através da interface computadorizada. Essa interface possibilitará ao usuário controle sobre todo o sistema, incluindo agendamento de horários para acender/apagar as lâmpadas, acionamento do modo econômico e identificação *on-line* do *status* das luminárias.

A fim de ilustrar o funcionamento do sistema, foi elaborado um diagrama para ilustrar o funcionamento do sistema, conforme exibido na Figura 01.

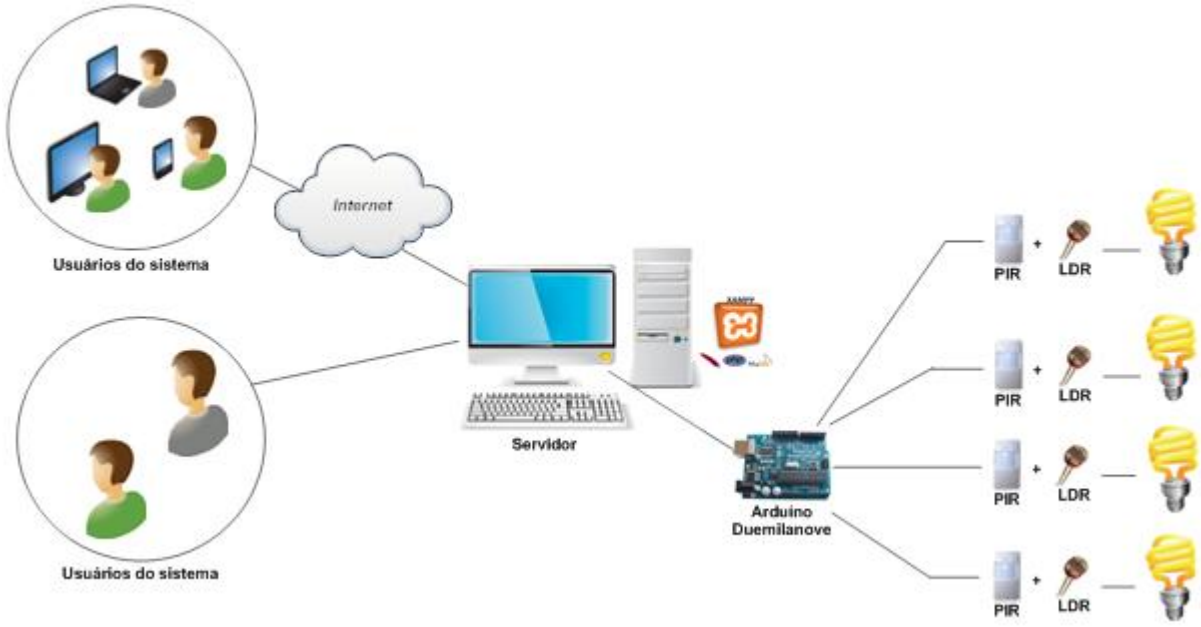

FIGURA 1.1 – TOPOLOGIA DO PROJETO

Através do diagrama exibido na Figura 1.1, é possível compreender facilmente que os usuários podem acessar a interface controladora (instalada no Servidor) localmente ou através da *internet*, e executar as ações desejadas. Os comandos das ações são enviados ao Arduíno, que interpreta o comando e com o auxílio dos sensores de presença e de luminosidade executa a ação demandada.

1.6. – RESULTADOS ESPERADOS

Para este projeto construiu-se um circuito para integração dos componentes eletrônicos a fim de possibilitar a automação e, para a interface computadorizada, foi desenvolvida uma interface web para controle/visualização do sistema.

O circuito conta com sensores de luminosidade, sensores de presença e componentes eletrônicos (como resistores, circuitos integrados, relés e diodos) que são conectados à plataforma Arduíno, que possui um microcontrolador integrado que é responsável pela execução de todos os comandos enviados através da interface web, bem como também enviar a sinalização de retorno proveniente do LDR (*Light Dependent Resistor)* informando se o comando foi bem executado ou não.

A interface desenvolvida é responsável por toda a interação entre o sistema automatizado e o usuário. Através da interface, o usuário tem um panorama completo de seu ambiente e pode, a seu critério:

- Pré-configurar horários para que as lâmpadas se acendam ou apaguem;
- Acender ou apagar as lâmpadas em tempo real;
- Identificar se as lâmpadas estão acesas ou apagadas.

1.7. – ESTRUTURA DA MONOGRAFIA

A estrutura desta monografia consiste de 05 capítulos que tratam os assuntos descritos abaixo:

Capítulo 1 – Capítulo introdutório, onde é apresentado o objetivo do projeto.

Capítulo 2 – Capítulo de referencial tecnológico, onde é apresentada a teoria utilizada no projeto, abordando *hardware* e *software*.

Capítulo 3 – Capítulo de desenvolvimento - Implementação de *software* e *hardware*.

Capítulo 4 – Capítulo da aplicação do projeto com resultados.

Capítulo 5 – Conclusão.

CAPÍTULO 2 – REFERENCIAL TECNOLÓGICO

2.1. - ARDUÍNO

O Arduíno é uma plataforma de computação física de código aberto, fácil de usar, extremamente poderosa e que vem ganhando espaço considerável junto a entusiastas, acadêmicos e profissionais do ramo. Ele é baseado em uma simples placa microcontroladora, e um ambiente para desenvolver o código para a placa.

O Arduíno captura informações do meio através de uma série de sensores que monitoram variações no ambiente, trata essa informação convertendo-a para grandezas digitais e, a partir, daí executa um comando pré-definido por seu desenvolvedor, tal como: controlar luzes, motores ou outras saídas físicas.

O *hardware* do Arduíno é de código aberto e suas placas podem ser montadas a um custo baixíssimo ou compradas; O código do ambiente de desenvolvimento também é de código aberto e sua utilização é regida sob a licença GPL (*General Public License)*, as bibliotecas microcontroladoras C/C++ sob LGPL (*Library General Public License*), e os esquemas e arquivos CAD (*Computer-aided design*) sob *Creative Commons Attribution Share-Alike*.

2.2. - MOTIVOS PARA A ESCOLHA DO ARDUÍNO

Apesar de existirem diversas plataformas microcontroladas disponíveis no mercado - como, por exemplo: *Parallax Basic Stamp*, *Netmedia's*, *Phidgets* e *MIT's Handyboard* - a plataforma Arduíno (ilustrada na Figura 2.1) despontou para ser escolhida como cérebro deste projeto devido a sua grande popularidade junto à comunidade acadêmica e a diversos outros fatores, como os citados abaixo:

- Custo/Benefício: A versão mais barata do módulo Arduíno pode ser montada à mão, e, inclusive, os módulos de Arduíno pré-montados custam menos de R$ 100,00.
- *Hardware* e *Software* de código aberto: Ambos possuem seus códigos, esquemas e planos publicados na comunidade *Open Source*, onde os interessados podem tomar conhecimento da tecnologia, aprimorá-la e continuar seu desenvolvimento mesmo que o criador da tecnologia perca o interesse pela mesma. Sem dúvida, um dos grandes pontos fortes da plataforma Arduíno é o fato de ela ser *Open Source*.
- Multiplataforma: O *software* de Arduíno roda em sistemas operacionais *Microsoft Windows*, *Apple Macintosh OSX* e *GNU (GNU's Not Unix)/Linux*. A maioria dos sistemas microcontroladores está limitada ao sistema operacional *Microsoft Windows*.
- Portabilidade: Outra característica importante do Arduíno é que você pode carregar um programa nele através do computador central via USB (*Universal Serial Bus)*, desconectá-lo do computador e desligá-lo. Ao ligar o Arduíno novamente, ele irá rodar o último programa carregado sem necessidade de que o mesmo seja novamente carregado através do computador central via USB. Isso significa que só

há necessidade de plugar o Arduíno ao computador central quando for desenvolver e eliminar erros do programa. Concluindo, caso queira apenas rodar o programa inserido dentro do Arduíno não há necessidade de plugá-lo ao computador, pois o código já estará na plataforma independentemente da quantidade de tempo que o Arduíno esteja desligado.

- Ambiente de programação eficiente: O ambiente de programação do Arduíno é fácil de usar para principiantes e suficientemente flexível para que usuários avançados possam aproveitá-lo ao máximo.

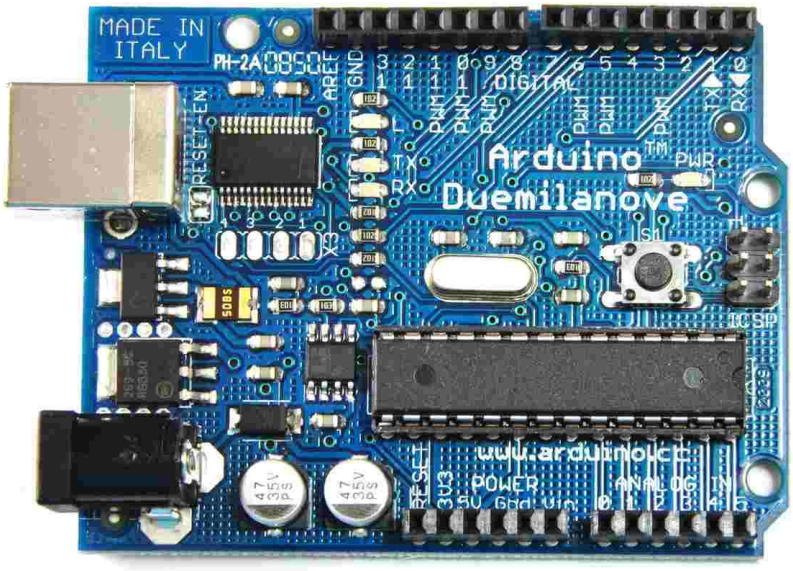

FIGURA 2.1 – ARDUÍNO DUEMILANOVE

FONTE: [ARDUINO *WEBSITE*, 2010]

2.3. - *HARDWARE* - O ARDUÍNO, SEUS COMPONENTES, INTERFACES E PINAGENS

2.3.1. - O ARDUÍNO DUEMILANOVE

Existem várias versões de plataformas Arduíno (Exemplo: Mini, LilyPad, Nano, Mega, Severino, Roboduíno, Duemilanove, Freeduíno, etc) todas elas, independentemente da versão, possuem um microcontrolador da fabricante Atmel® incorporado à plataforma.

A plataforma do Arduíno Duemilanove utilizada neste projeto é caracterizada pelo funcionamento de um microcontrolador Atmel® AVR (Microcontrolador RISC de *chip* único) ATmega328 operando à 5 V com 2 KB de RAM (*Random Access Memory*), 32 KB de memória Flash para armazenar programas e 1 KB de EEPROM (*Electrically-Erasable Programmable Read-Only Memory*) para armazenar parâmetros. A velocidade do *clock* é de 16 MHz, que é capaz de executar aproximadamente 300.000 linhas do código fonte C por segundo. A placa tem 14 pinos digitais de E/S e 6 pinos de entrada analógica. Possui um conector USB para se comunicar com o computador central e uma interface de alimentação elétrica DC (corrente contínua) para se conectar a uma fonte de energia externa de 6-20 V - uma bateria de 9 V, por exemplo. Possui, também, um chip da FTDI (*Future Technology Devices International*) responsável pela conversão de sinais do protocolo RS-232 para a tecnologia USB, e seis pinos ICSP (*In-Circuit System Programming*) para programação do microcontrolador com instruções AVR. [04].

Para facilitar a identificação dos componentes supracitados, é apresentada abaixo, na Figura 2.2, uma ilustração do Arduíno com os componentes devidamente numerados para identificação no Quadro 01.

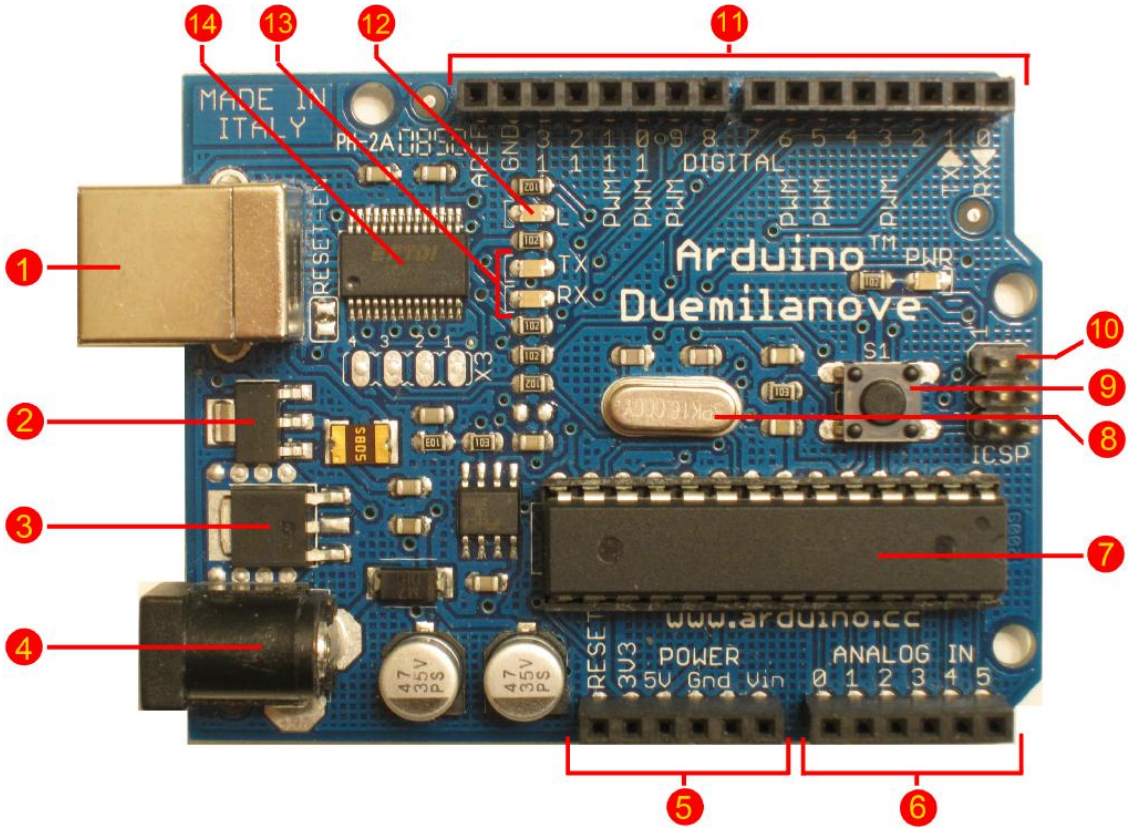

FIGURA 2.2 – ARDUÍNO DUEMILANOVE: IDENTIFICAÇÃO DE COMPONENTES

FONTE: [ARDUINO *WEBSITE*, 2010]

QUADRO 01 – IDENTIFICAÇÃO DE COMPONENTES DO ARDUÍNO

ID	**SIGNIFICADO**
01	Conector USB
02	Transistor
03	Regulador de tensão
04	Conector de energia elétrica
05	Pinos de força
06	Pinos analógicos
07	Microcontrolador Atmel ATmega328
08	Cristal oscilador de 16 MHz

09	Botão de *Reset*
10	ICSP
11	Pinos digitais
12	LED (*Light Emitting Diode*) do pino 13
13	LED's do dispositivo serial
14	Controlador FTDI USB

FONTE: ARDUINO *WEBSITE*, 2010

2.3.2. - O MICROCONTROLADOR ATMEL AVR ATMEGA328

O microcontrolador é um dos principais componentes em um projeto eletrônico. É praticamente um computador em um *chip*, pois contém processador, memória e periféricos de entrada/saída.

O Atmel AVR ATmega328 especificamente é um microcontrolador de arquitetura RISC (*Reduced Instruction Set Computer*), que, diferentemente da maioria dos demais microcontroladores disponíveis no mercado, utiliza memória flash para armazenar a programação.

O microcontrolador acoplado no Arduíno Duemilanove possui um formato de pacote DIP (*Dual In-Line Package*) de 40 pinos, que inclui endereços externos multiplexados e dados de controlador. Sua pinagem é idêntica à do Intel 8051 à exceção do *RESET*, que tem polaridade invertida. Os detalhes da pinagem podem ser visualizados na Figura 2.3.

FIGURA 2.3 – MAPEAMENTO DA PINAGEM DO ATMEGA328

FONTE: [ATMEL 8-BIT AVR DATASHEET, 2010]

2.3.3. - ENTRADAS E SAÍDAS

Cada um dos 14 pinos digitais do Duemilanove pode ser usado como entrada ou saída usando as funções pinMode(), digitalWrite(), e digitalRead(), da linguagem de programação *Processing*. Eles operam com 5 V. Cada pino pode fornecer ou receber um máximo de 40 mA e tem um resistor *pull-up* interno (desconectado por padrão) de 20-50 kΩ. Além disso, alguns pinos têm funções especializadas: [12]

- Serial: 0 (RX) e 1 (TX). Usados para receber (RX) e transmitir (TX) dados seriais TTL (*Time To Live*). Estes pinos são conectados aos pinos correspondentes do chip serial FTDI USB-to-TTL;
- PWM: 3, 5, 6, 9, 10, e 11. Fornecem uma saída analógica PWM (*Pulse-Width Modulation*) de 8-bit com a função analogWrite();

- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estes pinos suportam comunicação SPI (*Serial Peripheral Interface*), que embora compatível com o *hardware*, não está incluída na linguagem do Arduíno;
- LED: 13. Há um LED já montado e conectado ao pino digital 13. [12]

O Duemilanove tem 6 entradas analógicas, cada uma delas está ligada a um conversor analógico-digital de 10 bits, ou seja, transformam a leitura analógica em um valor dentre 1024 possibilidades (exemplo: de 0 a 1023). Por padrão, elas medem de 0 a 5 V, embora seja possível mudar o limite superior usando o pino AREF (*Analog Reference*)e um pouco de código de baixo nível. Adicionalmente, alguns pinos têm funcionalidades especializadas: [12]

- I2C: 4 (SDA) e 5 (SCL). Suportam comunicação I2C (TWI) usando a biblioteca Wire. [12]
- AREF: Referência de tensão para entradas analógicas. Usados com analogReference().[12]
- *Reset*: Envie o valor LOW para reiniciar o microcontrolador. Tipicamente utilizados para adicionar um botão de *reset* aos *Shield*s (placas que podem ser plugadas ao Arduíno para estender suas capacidades) que bloqueiam o que há na placa. [12]

2.3.4. - ALIMENTAÇÃO

O Arduíno Duemilanove pode ser alimentado pela conexão USB ou por qualquer fonte de alimentação externa. A fonte de alimentação é selecionada automaticamente. [12]

A alimentação externa (não-USB) pode ser tanto de uma fonte ou de uma bateria. A fonte pode ser conectada com um plugue de 2,1 mm (centro positivo) no conector de alimentação. Cabos vindos de uma bateria podem ser inseridos nos pinos GND (terra) e V_{in} (entrada de tensão) do conector de alimentação. [12]

A placa pode operar com uma alimentação externa de 6 a 20 V. Entretanto, se a alimentação for inferior a 7 V o pino 5 V pode fornecer menos de 5 V e a placa pode ficar instável. Se a alimentação for superior a 12 V o regulador de tensão pode superaquecer e avariar a placa. A alimentação recomendada é de 7 a 12 V. [12]

Os pinos de alimentação são:

- V_{in}: entrada de alimentação para a placa Arduíno quando uma fonte externa for utilizada. Você pode fornecer alimentação por este pino ou, se usar o conector de alimentação, acessar a alimentação por este pino; [12]
- 5 V: A fonte de alimentação utilizada para o microcontrolador e para outros componentes da placa. Pode ser proveniente do pino V_{in} através de um regulador *on-board* ou ser fornecida pelo USB ou outra fonte de 5 V; [12]
- 3V3: alimentação de 3,3 V fornecida pelo circuito integrado FTDI (controlador USB). A corrente máxima é de 50 mA; [12]
- GND (*ground*): pino terra. [12]

2.3.5. - *RESET* AUTOMÁTICO

Algumas versões anteriores do Arduíno requerem um *reset* físico (pressionando o botão de *reset* na placa) antes de carregar um *sketch* (o programa a ser compilado). O Arduíno Duemilanove é projetado de modo a permitir que isto seja feito através do *software* que esteja

rodando no computador conectado. Uma das linhas de controle de *hardware* (DTR - *Data Terminal Ready*) do FT232RL está conectada ao *reset* do ATmega328 via um capacitor de 100µF. Quando esta linha é colocada em nível lógico baixo, o sinal cai por tempo suficiente para reiniciar o chip. O *software* Arduíno usa esta característica para permitir carregar o programa simplesmente pressionando o botão ”*upload*” no ambiente Arduíno. Isto significa que o *bootloader* pode ter um *timeout* mais curto, já que a ativação do DTR (sinal baixo) pode ser bem coordenada com o início do *upload*. [12]

Estas configurações têm outras implicações. Quando o Duemilanove esta conectado a um computador rodando *Mac OS X* ou *GNU*/*Linux*, ele reinicia toda vez que a conexão é feita por *software* (via USB). No próximo meio segundo aproximadamente, o *bootloader* estará rodando no Duemilanove. Considerando que é programado para ignorar qualquer coisa a não ser um *upload* de um novo código, ele *intercepta* os primeiros *bytes* de dados enviados para a placa depois que a conexão é aberta. Se um *sketch* rodando na placa recebe configurações de uma vez ou outros dados ao iniciar, assegure-se que o *software* que esteja comunicando espere um segundo depois de aberta [12]

2.3.6. - PROTEÇÃO DE SOBRECARGA DO USB

O Arduíno Duemilanove tem um polifusível que protege a porta USB do computador contra curto-circuito e sobre-corrente. Apesar da maioria dos computadores possuírem proteção interna própria, o fusível proporciona uma proteção extra. Se mais de 500mA forem aplicados na porta USB, o fusível irá automaticamente interromper a conexão ate que o curto ou a sobrecarga seja removida. [12]

2.3.7. - FT232RL: COMUNICAÇÃO ENTRE A PLATAFORMA E O COMPUTADOR

A transferência dos dados para o microcontrolador é realizada através da emulação de uma conexão virtual RS-232 ou TTL. Como o Arduíno Duemilanove não possui interface de padrão RS-232, é necessário instalar programas e *drivers* relacionados à conversão RS-232 ou TTL para sinais USB, permitindo que *hardwares* de tecnologia antiga sejam compatíveis com computadores modernos.

O circuito integrado responsável pela conversão do protocolo RS-232 para a tecnologia USB é o FTDI FT232RL, que executa de forma eficiente e transparente toda a comunicação de dados entre o computador e o processador da plataforma, ATmega328, que apenas faz a gestão da comunicação de dados FIFO (*First In, First Out*) e das colisões de mensagens. [19]

Este chip permite comunicações em série com uma grande gama de velocidades: 1200, 2400, 4800, 9600, 19200, 38400,57600 e 115200 bps (*Bits* Por Segundo). Neste projeto, a comunicação de dados se dará com velocidade de 9600 bps.

2.3.8. - ICSP

O ICSP (programação serial Em-Circuito ou, do idioma inglês, *In-Circuit System Programming*) é uma tecnologia que permite que circuitos integrados que já estão montados e devidamente soldados nas placas sejam programados. No Arduíno, o ICSP permite que os microcontroladores ATmega328 sejam programados diretamente com instruções AVR sem a necessidade de se utilizar uma IDE (*Integrated Development Environment*) do Arduíno. Neste caso, é possível programar o ATmega328 da mesma forma que o fabricante programou o

Arduíno com o *bootloader*. O *bootloader* ajuda o Arduíno a entender seus esboços (*Sketches*) e executá-los, uma vez que tiverem sido carregados para o Arduíno.

O ICSP permite que proprietários de plataformas Arduíno façam upgrades em seus dispositivos quando a equipe de Desenvolvimento do Projeto Arduíno disponibiliza atualizações. O ICSP também permite que a própria equipe do Arduíno faça atualizações de última hora no *bootloader* da placa antes de enviá-la ao cliente, mesmo que a placa já esteja totalmente montada. Em outras palavras, o Arduíno é manufaturado sem *bootloader* e, como o *bootloader* é atualizado freqüentemente, a equipe do Arduíno aguarda a versão mais recente do *bootloader* para carregá-lo na plataforma e enviá-lo ao cliente. [19]

Concluindo, o ICSP pode ser utilizado para programar o Arduíno ou para usar o Arduíno como um programador ISP (*In-System Programming*).

2.3.9. - *BOOTLOADER*

O *bootloader* é um pequeno programa que roda na memória flash do microcontrolador AVR, que nunca é sobrescrito e é inicializado assim que a plataforma é energizada. O Arduíno precisa de um *bootloader* a fim de programar seus esboços (*sketches*) e transferir o código através de uma comunicação serial/USB. O trabalho do *bootloader* é ler dados do programa do UART (*Universal Asynchronous Receiver/Transmitter*) e escrevê-los no *flash* interno do microcontrolador. O AVR ATmega328 é fabricado e vendido sem o código na memória *flash*. O código pode ser carregado através dos 6 pinos reservados ao ISP, usando um programador ISP para microcontroladores AVR.

Para o Arduíno, o ISP é usado somente uma vez (no momento que é fabricado) - para carregar um pequeno *bootloader*. Na inicialização, o *bootloader* é executado e comunica-se

com o UART serial (Pinos TX e RX). A partir desse momento, o Arduíno pode ser programado através dos pinos seriais usando o protocolo STK500. Como os pinos seriais são conectados ao chip conversor USB para serial da FTDI, o Arduíno pode ser programado através da conexão USB. [19]

2.3.10. - AMPLIFICADOR OPERACIONAL

Segundo palavras de *David F. Stout* em seu livro, *Handbook of Operational Amplifier Circuit Design* publicado em 1976, os amplificadores operacionais são amplificadores de acoplamento direto, de alto ganho, que usam realimentação para controle de suas características. Eles são hoje encarados como um componente, um bloco fundamental na construção de circuitos analógicos. Internamente, são constituídos de amplificadores transistorizados em conexão série. [23]

Os amplificadores operacionais são usados em amplificação, controle, geração de formas de onda senoidais etc.. Com emprego na realização das funções clássicas matemáticas como adição, subtração, multiplicação, divisão, integração e diferenciação, os amplificadores operacionais são os elementos básicos dos computadores analógicos. São úteis ainda em inúmeras aplicações em instrumentação, sistemas de controle, sistemas de regulação de tensão e corrente, processamento de sinais, etc. [23]

O amplificador operacional, modelo LM358, ilustrado abaixo na Figura 2.4, presente na plataforma Arduíno, utiliza entradas bipolares PNP (ligação lógica positiva) que tem seus coletores conectados ao barramento de alimentação negativa. [14]

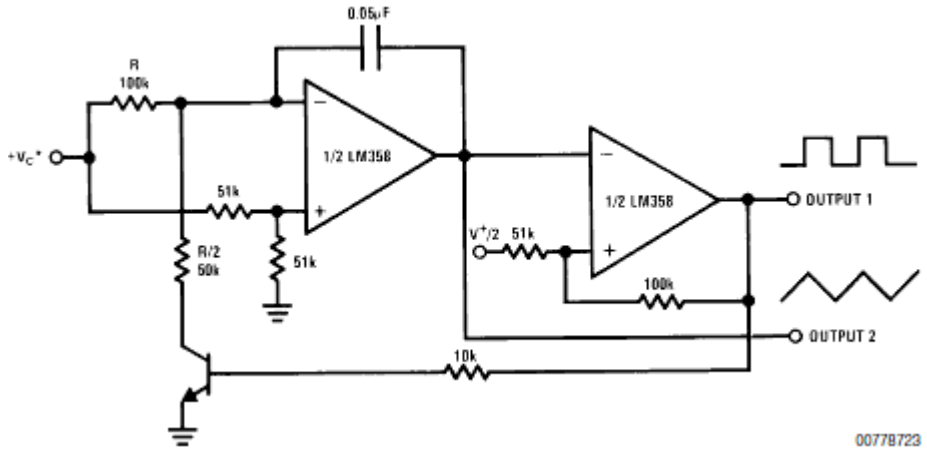

FIGURA 2.4 – EXEMPLO DE CIRCUITO INTEGRADO DO AMPLIFICADOR OPERACIONAL

FONTE: [MANCINI, 2002]

2.3.11. - CLOCK (OSCILADOR DE CRISTAL)

O cristal instalado no Arduíno o ajuda a tratar questões ligadas ao tempo. Por exemplo, em um projeto que tenha por objetivo acionar o interruptor de número 1 a cada 15 minutos e ligar os interruptores 2, 3 e 4 um após o outro com pausa de 20 minutos de um sobre o outro, o cristal do Arduíno é o elemento responsável por determinar o tempo exato do acionamento.

O código escrito na parte de cima do cristal é o 16.000H9H e transmite a informação de que o cristal opera a uma frequência de 16 MHz. OU seja, o Arduíno dispõe de um cristal que faz 16 milhões de ciclos por segundo.

O cristal é importante em qualquer equipamento que execute aplicações que dependam do tempo para serem bem executadas, tais como: microprocessadores, *webcams*, computadores, etc.

A precisão do cristal depende de sua especificação e, no caso do Arduíno, ele tem uma precisão de 100 ppm (*Pulse Position Modulation*). O que significa que o cristal tem uma margem de erro de 100 ciclos a cada 1 milhão de ciclos. Logo o máximo de erro/desvio que o cristal está submetido é de uma ordem de 30s a cada 1 ano.

2.3.12. - TRANSISTOR (MOSFET)

Segundo Newton C. Braga, em seu livro Curso Básico de Eletrônica publicado em 2009, Transistores de Efeito de Campo MOS (*Metal Oxide Semiconductor*), ou MOSFETs (*Metal Oxide Semiconductor Field Effect Transistor*), são dispositivos derivados dos transistores de efeito de campo comuns, mas com algumas mudanças na sua estrutura. Conforme ilustrado na Figura 2.5, no MOSFET pode se observar uma fina camada de óxido de metal (que dá nome ao dispositivo) que isola o substrato da região de comporta, em lugar da junção encontrada no JFET (*Junction Field Effect Transistor*). [09]

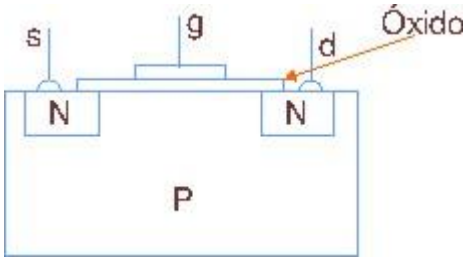

FIGURA 2.5 – MOSFET

FONTE: [BRAGA, 2009]

No entanto, o funcionamento do MOSFET é o mesmo: uma tensão aplicada no terminal de comporta provoca variações ou controla a corrente que flui entre o dreno e a fonte. Isso significa que os MOSFETs podem ser usados nas mesmas aplicações que o JFET, mas com algumas vantagens.

Os MOSFETs são comumente encontrados em circuitos de áudio e alta freqüência como, por exemplo, em amplificadores, pequenos transmissores, alarmes, etc. Em muitos equipamentos e projetos as funções destes componentes estão embutidas em circuitos integrados.

2.3.13. - REGULADOR DE TENSÃO

A série de reguladores de tensão MC33269 faz parte da família de reguladores de tensão positivo, de baixa saída, corrente média e de reguladores reparados e ajustáveis, especificamente projetados para serem usados em aplicações de baixa tensão de entrada. Estes dispositivos oferecem ao projetista do circuito uma solução econômica para o regulamento de tensão da precisão, enquanto mantém as perdas de potência em um nível mínimo. [15]

2.3.14. – ACESSÓRIOS (*SHIELD*S)

A plataforma Arduíno permite a conexão com módulos de *hardware* adicionais, denominados "*shield*s" , que acrescentam funcionalidade à plataforma. Estes acessórios, por assim dizer, integram fisicamente as mais diversas tecnologias à plataforma, como por exemplo, conexão *Ethernet*, *WiFi*, *Bluetooth*, *GPS, MP3, MIDI*, telas *Touchscreen*, módulos de reconhecimento de voz, etc. Entretanto, nenhum destes módulos será utilizado neste projeto.

2.4. - DEMAIS COMPONENTES FÍSICOS

2.4.1. - SENSOR DE LUMINOSIDADE

O LDR, ilustrado na Figura 2.6, é um componente que tem uma resistência que muda de acordo com a variação de luminosidade incidente sobre ele. A relação entre resistência e

intensidade de luz é inversamente proporcional, ou seja, sua resistência aumenta conforme a incidência de luz diminui sobre o dispositivo e vice-versa.

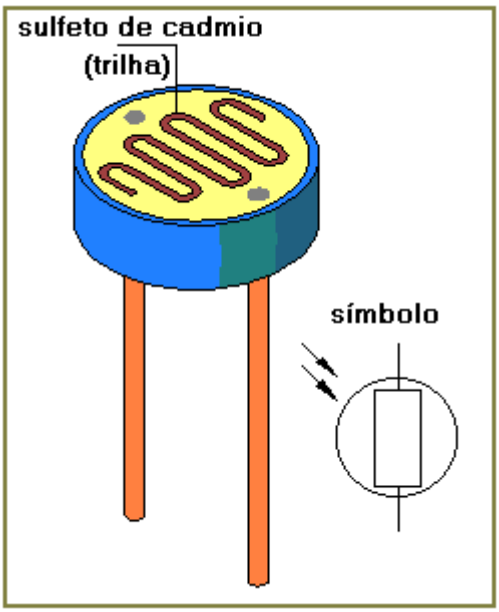

FIGURA 2.6 – LDR E SEU SÍMBOLO

FONTE: [NETTO, 2001]

O LDR é um sensor fotocondutivo fabricado com uma camada de material semicondutor - sulfato de cádmio. Devido à natureza do material empregado em sua fabricação, apresenta curvas de sensibilidade semelhantes à do espectro visível. Sua principal indicação, devido à característica citada, é, portanto, para aplicações onde esta característica visual se torna necessária, como por exemplo, controle por intensidade luminosa em sistemas de iluminação. [10]

Estes sensores também podem ser empregados para medir potências luminosas desde micro *watts* até mili *watts*, além de possuir um custo bastante reduzido. Sendo assim, o LDR é o sensor mais empregado na área de brinquedos eletrônicos e sistemas automáticos de iluminação urbana. [10]

2.4.2. - SENSOR DE PRESENÇA

Este projeto conta com 04 sensores da fabricante PPA, modelo *Sensit Solid*, com infravermelho passivo e relé de estado sólido (imune a campos magnéticos) idênticos aos exibidos na Figura 2.7.

FIGURA 2.7 – SENSOR DE PRESENÇA

O sensor possui as seguintes características físicas:

- Compensação de temperatura;
- 02 níveis de sensibilidade de detecção;
- Lente de Fresnel translúcida branca;
- Protetor plástico do piroelétrico;
- Fabricado com tecnologia SMD (*Surface Mountaing Device*); [20]

O sensor possui ainda as seguintes características técnicas:

- Duplo elemento piroelétrico;
- Detecção: 12 m de alcance x 110º de abertura;

- Tempo de acionamento: 2 segundos;
- Tempo de estabilização: 60 segundos;
- Compensação de temperatura: 10º a 45º;
- Tensão de alimentação: 10 ~ 18V DC;
- Consumo: 15 mA (12V)
- Dimensões
- 70 mm x 105 mm x 39 mm (Largura x Altura x Profundidade) [20]

O sensor de presença é um componente de extrema importância para atingir um dos objetivos deste projeto, que é a economia de energia. No modo econômico, ele é o responsável por enviar sinais ao Arduíno, que tratará a informação identificando se há ou não presença e acionará ou não as lâmpadas.

2.4.3 - ULN2003A

O dispositivo eletrônico ULN2003A, ilustrado na Figura 2.8, é um circuito integrado, que é um componente microeletrônico constituído por um conjunto de transistores, diodos, resistências e condensadores, fabricados ao mesmo tempo num mesmo processo, sobre uma substância comum semicondutora de silício designada comumente por *chip*. Os circuitos integrados revolucionaram o mundo da eletrônica e estão presentes em praticamente todos os dispositivos do gênero.

O ULN2003 é apenas um drive, constituído de 7 transistores Darlingtons NPN (Ligação lógica negativa), portanto não tem alimentação positiva. Ele apenas "chaveia" o GND para dispositivos externos, e estes dispositivos podem ser alimentados até a tensão suportada pelo coletor dos transistores internos do ULN2003 quando em estado de corte. O

pino 9 é a junção dos catodos dos 7 diodos de proteção, reversamente polarizados, caso ele seja usado para chavear cargas indutivas, como bobinas de relês.

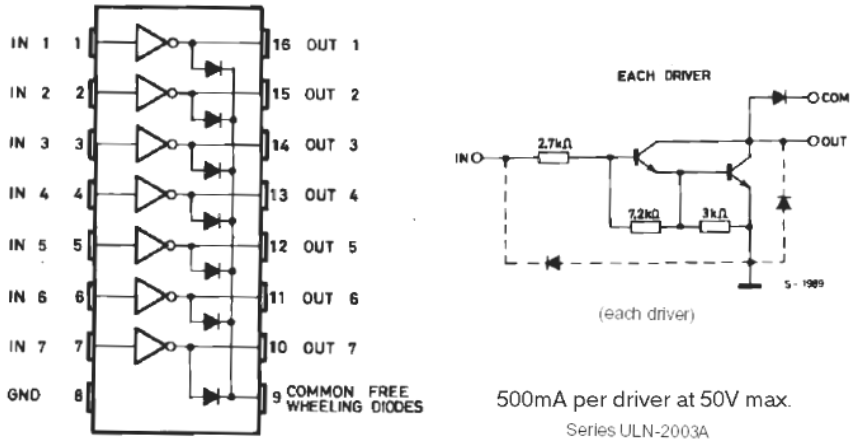

FIGURA 2.8 – CIRCUITO INTEGRADO ULN2003A

FONTE: [CI 2003A DATASHEET]

2.4.4. - RELÉ

O relé é um dispositivo comutador eletromecânico de extrema importância no universo da tecnologia – no ramo da robótica, domótica, eletrônica, mecatrônica e afins. Está presente na maioria dos dispositivos que fazem parte do dia a dia dos cidadãos e são muito comuns em eletrodomésticos, automóveis, sistemas telefônicos, dispositivos médicos e sistemas de automação atuando no controle eletrônico para desempenhar seu papel, seja ele ligar motores ou lâmpadas.

O tipo mais comum de relé eletromecânico não é o que está sendo abordado neste projeto. É o relé de parede que é utilizado em interruptores de energia para acender e apagar lâmpadas. A diferença com este tipo de relé de parede é que ele necessita de intervenção

humana para ser acionado, enquanto o relé presente neste projeto não necessita de intervenção humana para ser acionado.

O relé utiliza-se de princípios eletromagnéticos para operar corretamente. O interior de um indutor é composto de um indutor (bobina de cobre) que gera um campo magnético ao ser energizado com um pulso elétrico. A outra parte do indutor é composta de braços metálicos que fazem os contatos físicos da comutação. Quando o relé está desativado ou nenhum pulso elétrico lhe é fornecido, seus braços estão em uma posição que é conhecida como normalmente aberta (NA). Quando o relé está ligado ou um pulso elétrico lhe é enviado, o braço metálico se move em direção ao outro contato físico do relé. O braço físico se move à medida que o campo magnético gerado o impulsiona em direção ao indutor.

2.5. - AMBIENTE DE DESENVOLVIMENTO PARA O ARDUÍNO

A plataforma Arduíno inclui ainda um meio de desenvolvimento que permite escrever programas usando uma linguagem denominada *Processing*, que é uma linguagem derivada de um projeto de alunos do MIT (Instituto de Tecnologia de Massachusetts - EUA), que tem como grande objetivo facilitar o acesso de programadores ao mundo da programação de baixo nível (exemplo de linguagem: *Assembly*), sem necessidade alguma de conhecer a linguagem de máquina, que trabalha diretamente com os registradores dos processadores. A linguagem de programação de alto nível adotada pelo projeto *Processing* é uma versão simplificada da linguagem C/C++.

Processing é uma linguagem de programação de código aberto para as pessoas que querem criar imagens, animações, e interações. Desenvolvido inicialmente para servir como um *sketchbook* do *software* e para ensinar os fundamentos da programação de computador

dentro de um contexto visual, *Processing* igualmente evoluiu em uma ferramenta para gerar trabalho profissional de qualidade em sua forma final. Atualmente, existem dezenas de milhares de estudantes, artistas, desenhadores, pesquisadores e entusiastas que utilizam a linguagem *Processing* para a aprendizagem, a prototipificação e a produção. [21]

Para desenvolver aplicações utilizando a linguagem *Processing*, que funcionem em um microcontrolador, é utilizado um conjunto de abstrações, de codinome *Wiring*. Muitos se referem a este conjunto como linguagem *Wiring* ou linguagem Arduíno, mas na verdade são somente funções e bibliotecas escritas em C/C++

Segundo referências acerca da linguagem de programação do Arduíno disponíveis no *site* da referida plataforma, os programas em Arduíno podem ser divididos em três partes principais: estrutura, valores (variáveis e constantes) e funções. A linguagem Arduíno é baseada em C/C++. A referência acerca da linguagem de programação no que tange à sua aplicabilidade neste projeto se encontra no anexo desta monografia.

2.5.1. - APLICATIVO DE DESENVOLVIMENTO

O Arduíno é um ambiente multiplataforma e, como dito anteriormente, permite ser programado através dos três principais Sistemas Operacionais: *Microsoft Windows*, Apple *Macintosh iOS* e *GNU\Linux*. Abaixo será detalhado o procedimento para instalação do aplicativo de desenvolvimento no Sistema Operacional *Microsoft Windows*, bem como informações sobre a interface, código e parâmetros.

2.5.2. - INSTALAÇÃO – ARDUÍNO PARA *WINDOWS*

Primeiramente deve-se baixar o ambiente para o Arduíno que pode ser encontrado no seguinte site: *http://www.arduino.cc/en/Main/Software*, em *download* clique em *Windows* e baixe o arquivo arduino-0019.zip (ou mais novo). Será necessário um programa capaz de descompactar o arquivo (exemplos: *WinZip*, *WinRAR* etc.). Certifique-se de preservar a estrutura da pasta. Dê um duplo clique na pasta para abri-la, haverá uns arquivos e sub-pastas, clique no aplicativo arduíno, este será seu ambiente de desenvolvimento. [12]

Conecte a placa ao computador através do cabo USB, o LED verde na placa nomeado por PWR deve ascender, ele indica que a placa está ligada. O arduíno seleciona automaticamente a fonte de alimentação adequada. [12]

Quando se conecta a placa, o *Windows* deverá iniciar o processo de instalação do driver. No *Windows Vista*/*Seven*, o *driver* deve ser baixado e instalado automaticamente. No *Windows XP*, o assistente Adicionar Novo *Hardware* será aberto: [12]

- Quando o *Windows* perguntar se pode se conectar ao *Windows Update* para procurar o *software*, selecione não. Clique em Avançar. [12]
- Selecione personalizar, logo após selecione instalar e clique em Avançar. [12]
- Certifique-se de procurar o melhor *driver*, desmarque a pesquisa de mídia removível; selecione Incluir este local na pesquisa e procure os *drivers* /FTDI USB *Drivers* diretórios de distribuição do Arduíno. Clique em Avançar. [12]
- O assistente irá procurar o *driver* e em seguida, dizer que um *hardware* foi encontrado. Clique em Concluir. [12]

- O assistente de novo *hardware* abrirá novamente, faça todos os passos da mesma maneira, desta vez, uma porta serial USB será encontrada. [12]

2.5.3. - INTERFACE DO APLICATIVO DE DESENVOLVIMENTO

Para desenvolver um programa a ser executado no Arduíno é necessário utilizar a interface de desenvolvimento da plataforma. A interface foi desenvolvida em Java e é muito intuitiva. Para facilitar o entendimento, a funcionalidade de cada botão presente na interface de desenvolvimento é explicada abaixo, com o auxílio da Figura 2.9.

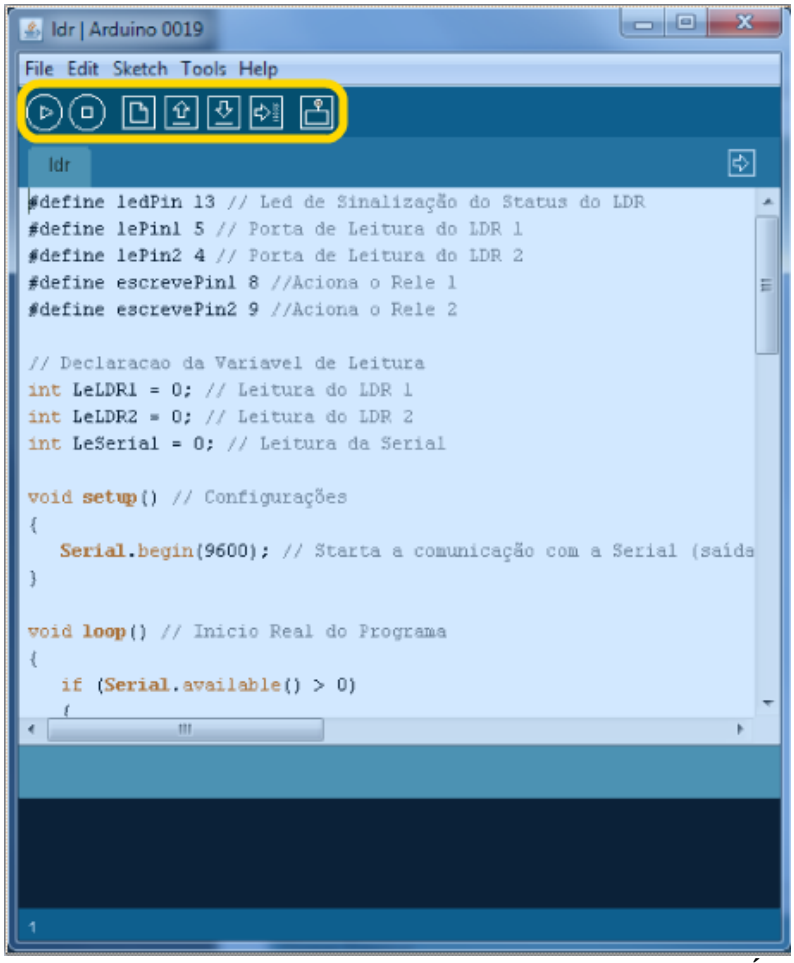

FIGURA 2.9 – AMBIENTE DE DESENVOLVIMENTO ARDUÍNO

Legenda dos ícones (na ordem da esquerda para a direita):

 Compilar;

 Parar processo de compilar;

 Novo;

 Abrir;

 Salvar;

 *Upload*;

 Leitura da interface serial

2.5.4. - LEITURA DA INTERFACE SERIAL

Através da interface de desenvolvimento, é possível realizar a leitura e envio de dados utilizando a interface serial, conforme ilustrado na Figura 2.10.

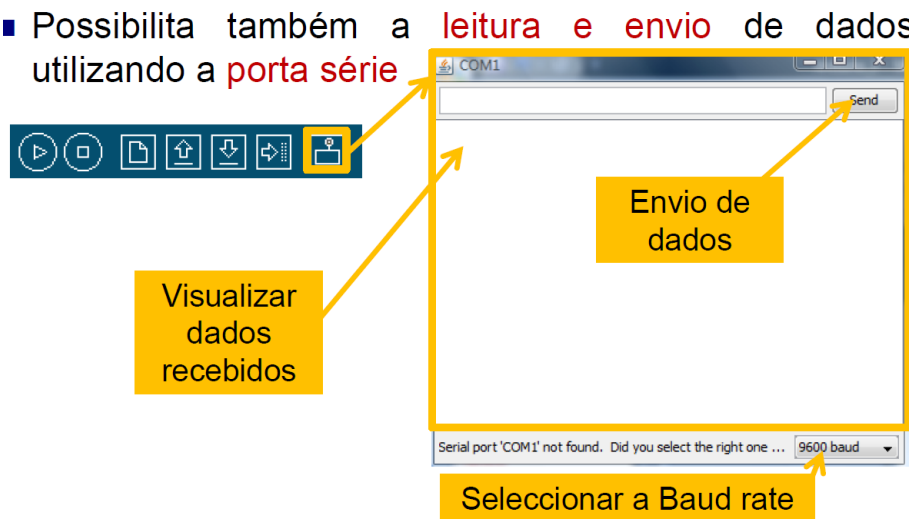

FIGURA 2.10 – LEITURA/ENVIO DE DADOS UTILIZANDO PORTA SERIAL

FONTE: [SANTOS, 2009]

2.5.5. - CICLO DE DESENVOLVIMENTO

O ciclo de desenvolvimento para o Arduíno é bem definido e, conforme pode ser observado na Figura 2.11, não difere do ciclo de desenvolvimento de microcontroladores estudados ao longo do curso, como o PIC, por exemplo.

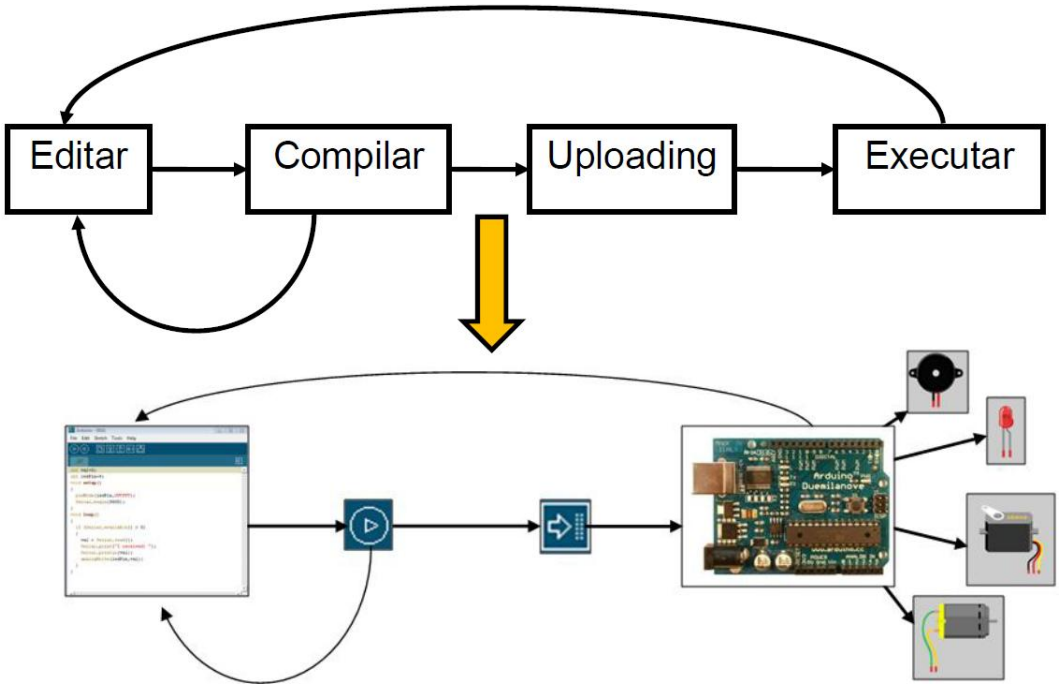

FIGURA 2.2 – CICLO DE DESENVOLVIMENTO PARA PLATAFORMA ARDUÍNO

FONTE: [SANTOS, 2009]

2.5.6. - ESTRUTURA DO *SKETCH*

A estrutura do código para desenvolver um programa para o Arduíno é idêntica à ilustrada na Figura 2.12, e como pode ser observado, é semelhante à estrutura de qualquer código em C.

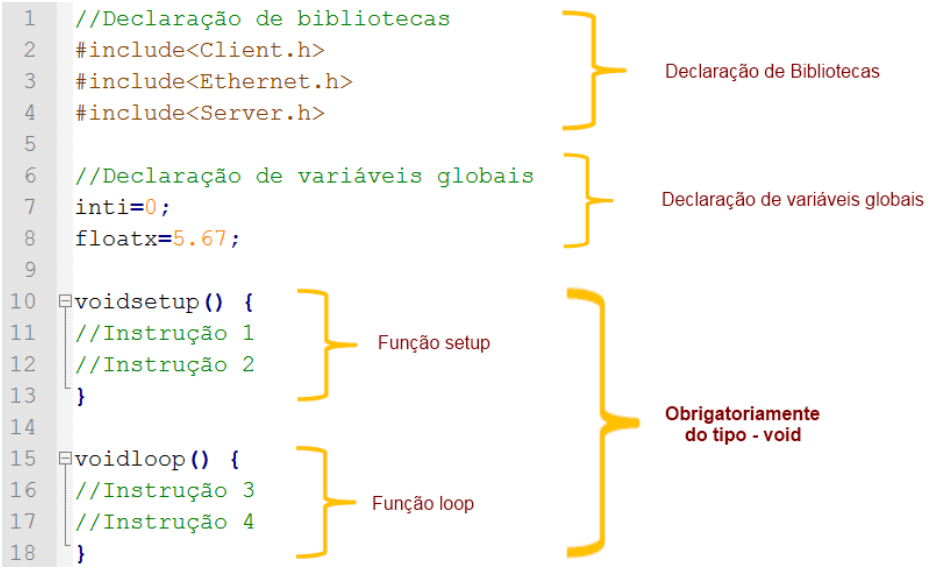

FIGURA 2.3 – ESTRUTURA DO *SKETCH*

2.5.7. - REFERÊNCIA DAS FUNÇÕES DA LINGUAGEM ARDUÍNO

Para desenvolver o código inserido dentro do Arduíno é necessário fazer uso de elementos de estrutura, elementos de função e variáveis próprias do Processing. É possível visualizar todos os elementos utilizados para desenvolver o código nas Figuras 2.13, 2.14 e 2.15.

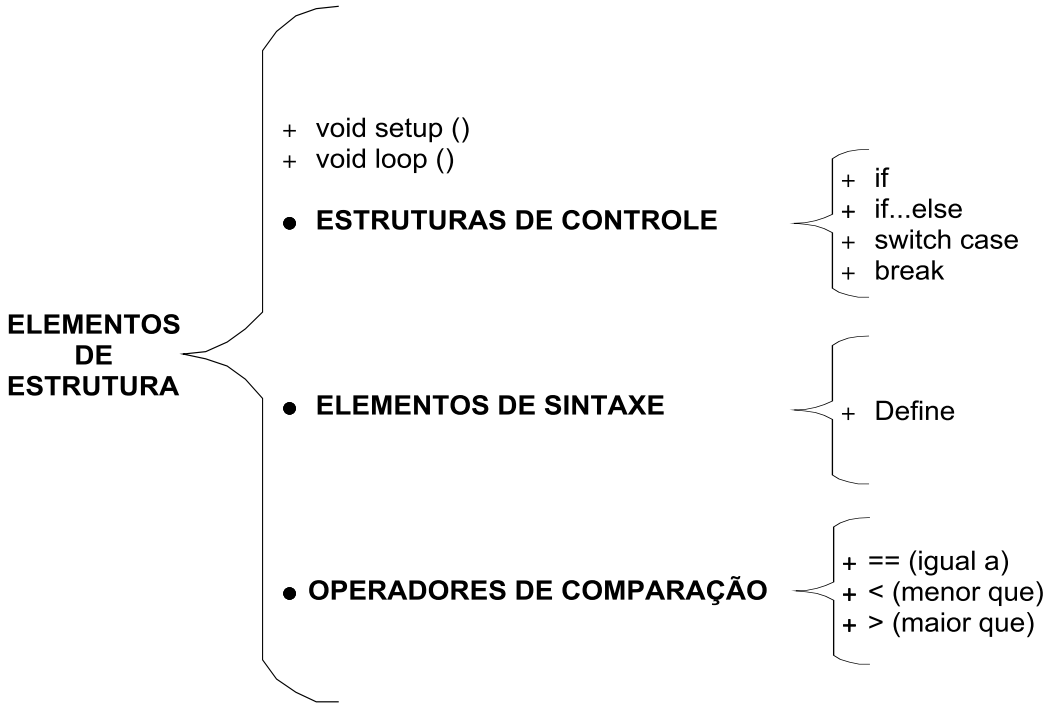

FIGURA 2.4 – ELEMENTOS DE ESTRUTURA DA LINGUAGEM *PROCESSING*

FONTE: [ARDUINO *WEBSITE*, 2010]

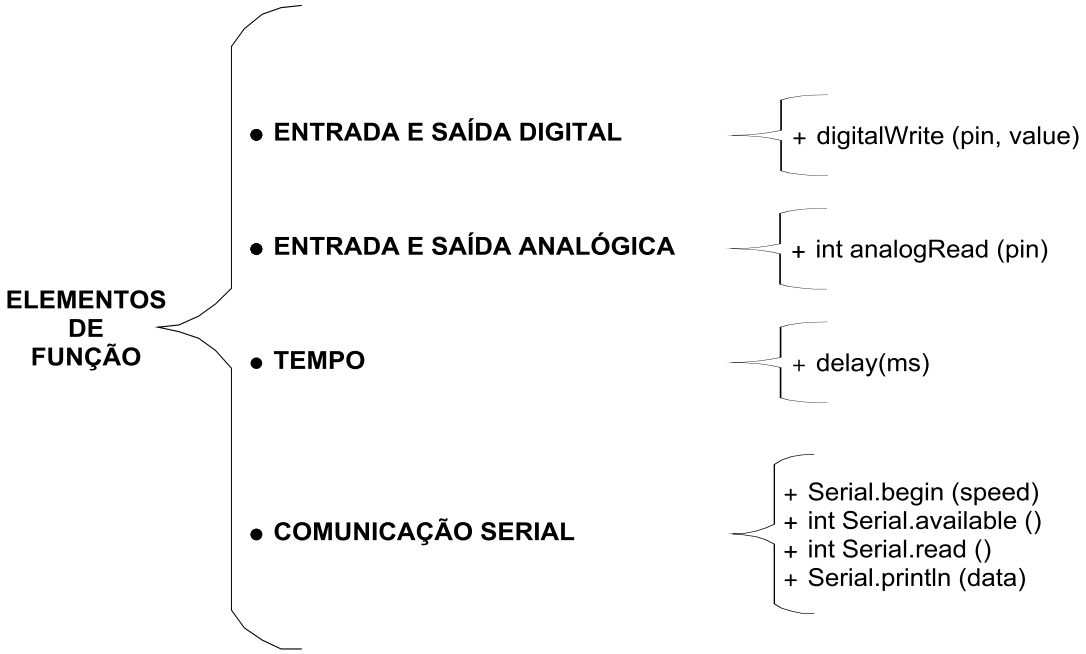

FIGURA 2.5 – ELEMENTOS DE FUNÇÃO DA LINGUAGEM *PROCESSING*

FONTE: [ARDUINO *WEBSITE*, 2010]

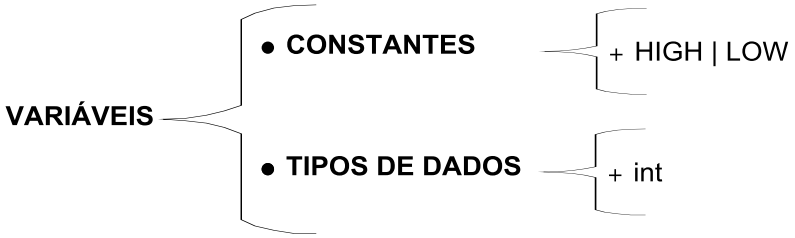

FIGURA 2.6 – VARIÁVEIS DA LINGUAGEM *PROCESSING*

FONTE: [ARDUINO *WEBSITE*, 2010]

Todas os elementos de função e estruturas utilizadas para desenvolver o código inserido no Arduíno encontram-se referenciadas no Anexo 01, com detalhes acerca de sua funcionalidade.

2.6. - SISTEMA *WEB*

2.6.1. - XAMPP

O XAMPP [13] (Figura 2.16), pacote integrado de servidores que incluem o Apache [25] e o MySQL entre outros, é um sistema altamente portável, que pode ser executado sem a necessidade de instalação, ou seja, pode ser utilizado até mesmo através de um dispositivo de armazenamento portável (*Pendrive*).

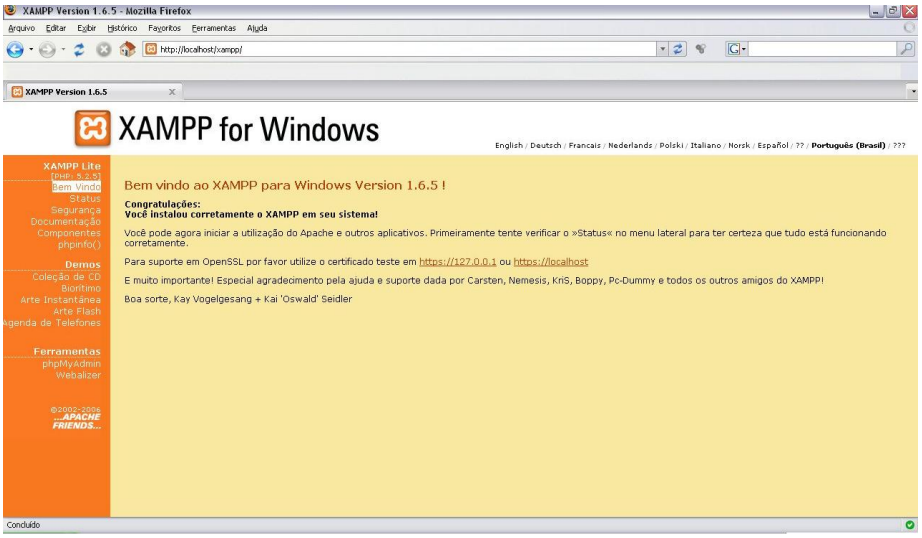

FIGURA 2.7 – XAMPP

2.6.2. - PHP

O PHP é uma linguagem de programação de uso geral criada especialmente para trabalhar em ambientes *Web*. Diferentemente de outras linguagens como o Javascript, o PHP é executado no servidor. Com isso seu código fonte nunca é revelado ao usuário final. E, diferentemente de linguagens como C e Pascal, o PHP não é uma linguagem compilada e sim interpretada, dispensando declarações rígidas de variáveis e permitindo uma mistura de códigos em HTML (*HyperText Markup Language*) e PHP.

Criado em 1994, a partir do sucesso do *Personal Home Page Tools* de *Rasmus Lerdof*., o PHP chamou atenção de *Zeev Suraski* e, em 1997, se tornou um pré-processador de hipertexto chegando à versão 3. Em 2000, o PHP atingiu a versão 4 trazendo consigo tecnologias da *Zend Technologies*, que permitiram funções mais complexas e maior interação com o usuário. [18]

O PHP já se mostrou superior em quesitos como simplicidade de conexão a bancos de dados, desempenho e gerenciamento de memória, além de ser distribuído sob licença GPL e de rodar em inúmeras plataformas. Essa licença torna o PHP *open source* e traz novas vantagens como o grande número de colaboradores distribuídos pelo mundo e o custo extremamente baixo de implementação, sendo apenas necessário investir no *hardware* e utilizar um conjunto completo de ferramentas gratuitas e open source.

Em sua versão mais recente - o PHP5 - foi introduzido um novo modelo de objetos e, com isso, foram implementadas muitas funções e características firmando o PHP como linguagem orientada a objetos, e o colocando a novos níveis de concorrência com linguagens mais robustas, como o Java [01].

2.6.3. - HTTP

Como protocolo mais utilizado na troca de informações da *internet*, o HTTP (*Hypertext Transfer Protocol*) efetua a comunicação transmitindo e recebendo arquivos na linguagem HTML. O canal de comunicação é estabelecido entre o *software* cliente, muitas vezes representado pelo navegador e o servidor *web*. [18]

Em síntese, este protocolo da camada de aplicação do modelo OSI (*Open Systems Interconnection*), estabelece uma conexão TCP (*Transmission Control Protocol*) por uma porta específica, e envia uma requisição como, por exemplo, “GET / HTTP 1.1”, que requisitaria a página padrão ao servidor *web*. Essa requisição é então respondida com uma mensagem no formato HTML, que é então interpretada pelo navegador e apresentada ao usuário. Essa resposta contém cabeçalhos que permitem ao navegador identificar se a requisição foi um sucesso ou algum erro ocorreu.

2.6.4. - APACHE

O Projeto do Servidor Apache é uma tentativa de desenvolver e manter um projeto de servidor de código aberto para sistemas modernos, produzindo um servidor seguro, eficiente e expansível, que esteja em sintonia com os padrões atuais. [03]

O Servidor Apache é gratuito e desenvolvido por voluntários membros do Grupo Apache. Atualmente, o Apache possui 54% do mercado de servidores *Web*, dominando o mercado desde o ano de 1996 [16], fazendo dele o servidor mais popular.

O servidor *web* é a ferramenta principal na atual estrutura da *web*. Ele é responsável por receber requisições de navegadores, processar e enviar os resultados de volta ao navegador que fez a requisição. Esse processamento pode contar com inúmeras ações, tais como: buscar arquivos HTML, seguir referências para imagens, convocar execução de *scripts* PHP, e muitas outras.

O Apache possui uma estrutura modularizada com um módulo central (conhecido como "*core*") responsável pelo recebimento e encaminhamento de requisições. Os outros módulos são responsáveis pelas outras funções, inclusive o módulo responsável pela interação com o PHP. [03]

2.6.5. - MYSQL

Trata-se de um banco de dados de código aberto sendo atualmente, o mais popular de sua categoria. Sua arquitetura permite que seja extremamente rápido e simples de usar e sua distribuição gratuita é um grande atrativo para programadores e empresários que desejam publicar sites na *internet*.

Com uma estrutura robusta, embora simplificada pela ausência de funções nativas do aplicativo SQL *Server* da *Microsoft*, o MySQL é totalmente capaz de responder e atender soluções *web* de pequeno a grande porte, com a vantagem de utilizar menos recursos do *hardware*, em comparação a servidores comercias. Mesmo com essas simplificações, o banco implementa a linguagem SQL (*Structured Query Language*) amplamente utilizada na realização de buscas em banco de dados. [18]

2.6.6. – REFERÊNCIAS DAS FUNÇÕES PARA O SISTEMA *WEB*

2.6.6.1 – PHP

Da mesma forma que foi necessário fazer uso de funções da linguagem de programação *Processing* para desenvolver o código para o Arduíno, agora é necessário fazer uso de funções da linguagem PHP para desenvolver a interface *web* e tratar as informações oriundas do Arduíno para que o sistema funcione corretamente.

As funções relacionadas para a construção da interface e do controle da iluminação estão relacionadas no Anexo 02.

2.6.6.2 – SQL

A linguagem utilizada para a criação do banco de dados e da tabela de alocação de cômodos e agendamento de horários é a SQL. As funções utilizadas para atingir este propósito estão referenciadas no Anexo 02.

CAPÍTULO 3 – DESENVOLVIMENTO

3.1. – APRESENTAÇÃO GERAL

O objetivo deste capítulo é apresentar o desenvolvimento da solução de automação. O desenvolvimento está dividido em três partes distintas: circuito eletrônico, código do circuito e interface controladora informatizada.

O circuito eletrônico é o responsável pela interligação física entre o dispositivo a ser controlado – luminárias – e o computador. O circuito é composto de diversos elementos – resistor, sensor de presença, sensor de iluminação, relé, circuito integrado e o Arduíno. Através do circuito que está sendo apresentado, é possível controlar até 08 luminárias. Caso haja necessidade de controlar mais luminárias, será necessário integrar outro Arduíno Duemilanove ou substituir o presente por outro que possua mais pinos, como o Arduíno *Mega*, por exemplo.

O Arduíno Duemilanove pode integrar à sua estrutura física, dispositivos (*Shield*s) com suporte à tecnologia *Ethernet* que permitem configurá-lo para atuar como um servidor *web*. Entretanto, para a realização deste projeto, foi proposta a utilização de um microcomputador de uso pessoal para fins econômicos (uma vez que pressupõe-se que o usuário já possua um PC - *Personal Computer*), e a integração do Arduíno Duemilanove com *shield*s não faz parte do escopo deste projeto. Desta forma, o Arduíno Duemilanove fica conectado ao PC através de uma interface USB para se comunicar com o servidor *web* que está instalado neste mesmo computador.

O código do circuito é a parte responsável por interpretar e executar corretamente os comandos enviados através da interface controladora. O código é escrito em linguagem de programação *Processing* e está embarcado no Arduíno.

A interface controladora informatizada é desenvolvida utilizando-se as tecnologias PHP, MySQL e Apache. Sendo que essas três tecnologias são instaladas através de um pacote denominado XAMPP. A interface controladora é a responsável por enviar comandos ao microcontrolador e informar o *status* de ação bem sucedida ou não, bem como o status de leitura dos dispositivos físicos.

Através da união destas três partes, é possível que o operador interaja com o sistema de automação de forma eficiente. O projeto prevê que o sistema seja capaz de executar as funções de acordo com o exposto no Quadro 02.

QUADRO 02 – POSSIBILIDADES DE AÇÕES A SEREM EXECUTADAS PELO SISTEMA

Modo de Operação	Sensor de presença	Acionamento da luminária	Ação
Modo Automático	Ativado	Luminária específica	Programar luminária(s) com temporizador para ligar/desligar a(s) lâmpada(s) em X minutos.
		Conjunto de luminárias	
	Desativado	Luminária específica	Programar luminária(s) sem temporizador para ligar/desligar a(s) lâmpada(s) em X minutos.
		Conjunto de luminárias	
Modo Manual	Ativado	Luminária específica	Acionar luminária(s) com temporizador para ligar/desligar a(s) lâmpada(s) em X minutos.
		Conjunto de luminárias	
	Desativado	Luminária específica	Acionar luminária(s) sem temporizador para ligar/desligar a(s) lâmpada(s) em X minutos.
		Conjunto de luminárias	
Modo Leitura	Ativado	Luminária específica	Leitura de estado da lâmpada e presença no ambiente.

	Desativado	Conjunto de luminárias	Leitura de estado da lâmpada no ambiente.

O sistema irá permitir que o usuário faça a monitoramento de seu ambiente e o controle de forma totalmente flexível. Além de o usuário contar com a interface para poder executar comandos de seu interesse remotamente, poderá monitorar o estado de cada cômodo de seu ambiente. Ou seja, poderá utilizar a interface para, à distância, ter conhecimento se há alguma presença no ambiente ou se as lâmpadas estão acesas ou apagadas. A interface também irá mostrar ao usuário se o sistema está sendo operado em modo manual ou automático independente de alguma ação ser executada.

3.1.1. – FLUXOGRAMA GERAL

Aqui é apresentado o fluxograma do sistema, que pode ser visualizado na Figura 3.1. Através do fluxograma, é possível observar a coerência de sua estrutura com as informações contidas no Quadro 02. O referido quadro e o fluxograma são essenciais para o desenvolvimento correto do código de programação inserido no microcontrolador.

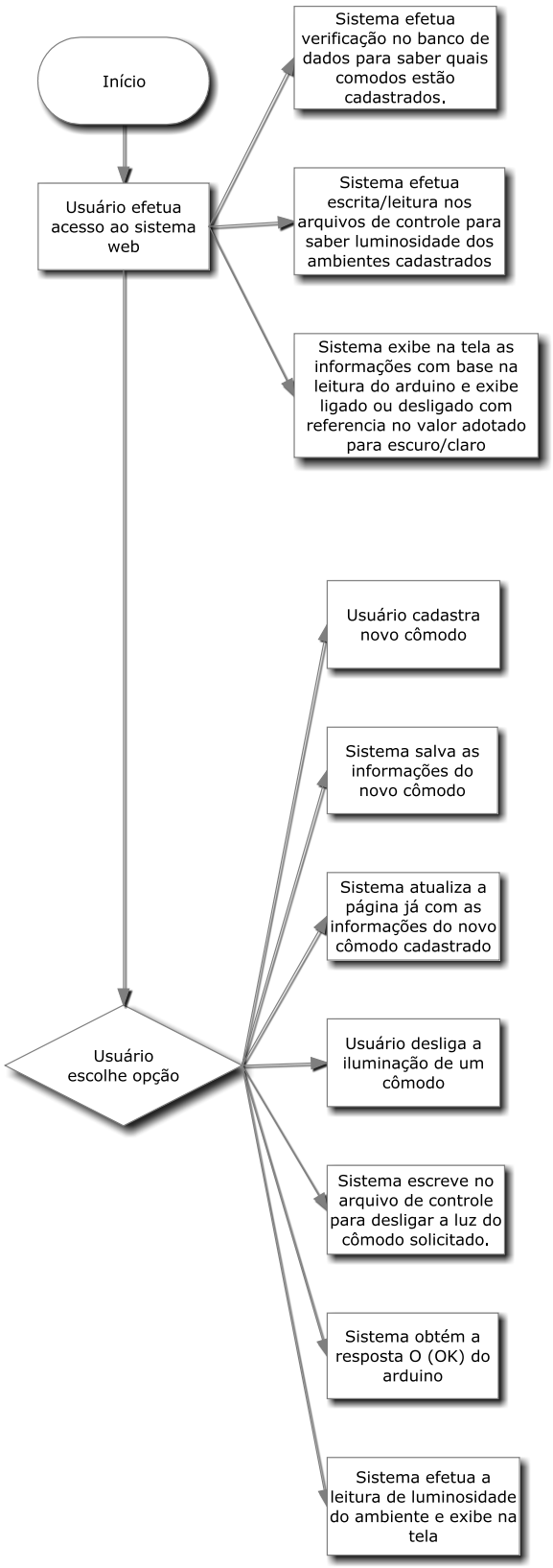

FIGURA 3.1 – FLUXOGRAMA DO SISTEMA

O termo ‘fluxograma’ designa uma representação gráfica de um determinado processo ou fluxo de trabalho, efetuado geralmente com recurso a figuras geométricas normalizadas e a setas unindo-as. Através desta representação gráfica, é possível compreender de forma rápida e fácil a transição de informações entre os elementos que participam no processo. [17]

3.1.2. – VISÃO GERAL DO PROJETO

O Arduíno é o responsável pela execução de todas as ações vislumbradas no Quadro 02. É o cérebro do sistema e, por isto, todos os sensores, resistores e circuitos integrados estão ligados a ele. Segue abaixo, na Figura 3.2, uma visão geral do projeto de forma simplificada exibindo as ligações realizadas entre os elementos do sistema.

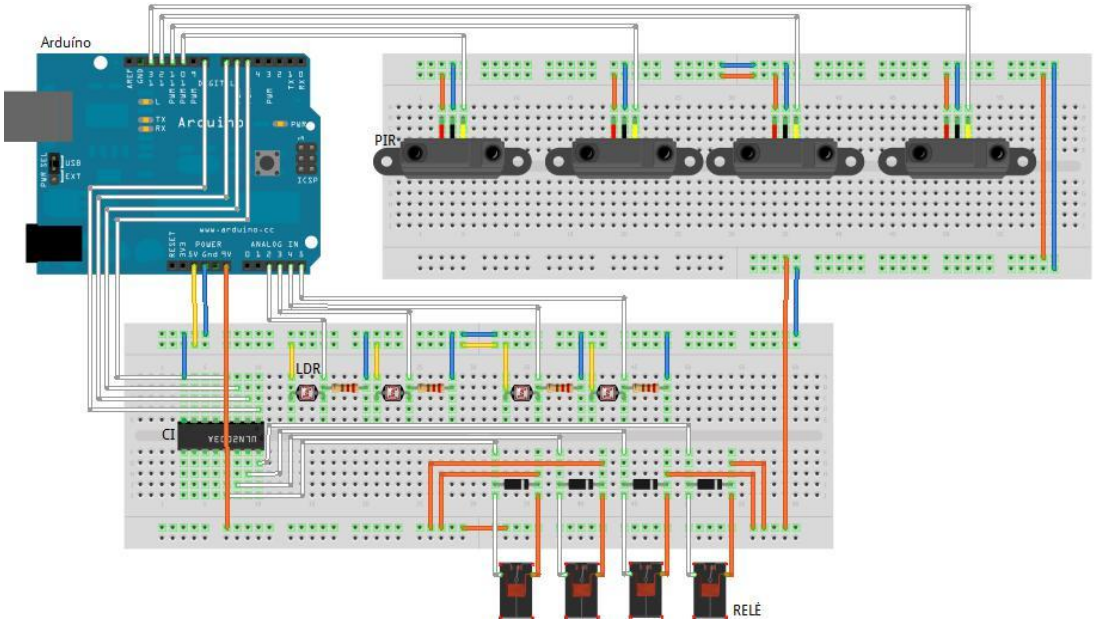

FIGURA 3.2 – DESENHO DO PROJETO NA PROTOBOARD

O Arduíno executa as ações, comandadas através da interface *web*, de acordo com os sinais de entrada ligados às portas analógicas 2, 3, 4 e 5 (LDR) e às portas digitais 10, 11, 12 e

13 (sensor de presença). As instruções para acender/apagar as lâmpadas saem das portas digitais 5, 6, 7 e 8 do Arduíno, que são ligados a relés – responsáveis pelo acionamento físico das lâmpadas.

3.2. – RELÉ

O circuito elétrico do projeto está desenhado de tal forma que a(s) lâmpada(s) está(ão) conectada(s) ao(s) relé(s) através dos contatos COMUM e N/A (Normal Aberto). Desta forma, quando houver necessidade de ligar uma lâmpada, o circuito será fechado e a mesma será ligada. De forma inversa acontecerá quando for necessário desligar uma lâmpada – o circuito será aberto.

Os Relés estão ligados a diodos retificadores que cumprem o papel de evitar que a corrente volte em sentido contrário.

Para o correto funcionamento do projeto, é necessário que os relés estejam ligados a uma fonte de 12 V. Neste projeto, os relés recebem a alimentação do Arduíno, mas poderiam receber alimentação elétrica de outra fonte que não o Arduíno.

3.3. – SENSOR DE ILUMINAÇÃO

Os sensores de luminosidade estão conectado às portas analógicas 2, 3, 4 e 5 do Arduíno.

A forma de ligação dos sensores é única a todos os sensores. Cada um dos sensores está ligado de forma serial a um resistor único (o qual tem sua outra extremidade ligada ao pólo negativo – terra – do Arduíno) e a uma porta analógica específica do Arduíno. A outra extremidade do sensor está ligada ao pólo positivo da fonte de alimentação do Arduíno.

O LDR, que é um transdutor de entrada, tem como principal função atuar no controle da passagem de corrente elétrica para o Arduíno, de forma que quando o ambiente está escuro, a resistência aumenta e passa pouca corrente para o Arduíno. Por outro lado, quando o ambiente recebe muita luz, a resistência do LDR diminui e passa muita corrente para o Arduíno. O Arduíno recebe a corrente elétrica (energia luminosa convertida em energia elétrica) tratada pelo LDR e de acordo com a quantidade de corrente elétrica recebida executa uma ação.

Através do uso de um aparelho para medição do nível de luminosidade do ambiente – Luxímetro - foi possível elaborar um quadro comparativo entre o nível de luminosidade do ambiente e o valor de leitura do LDR no momento. Esses dados são de extrema importância para referenciar o valor de resistência do LDR utilizado no código de programação para determinar em qual momento a lâmpada deve ser acesa ou apagada. As informações comparativas podem ser visualizadas no Quadro 03.

QUADRO 03 – VALORES DE LUMINÂNCIA DO LDR E LUXÍMETRO

HORÁRIO	DESCRIÇÃO	OBSERVAÇÃO	LUXÍMETRO (lx)	LDR (mA)
12 h ~ 14 h	Luminosidade alta	Sol a pino (sem iluminação artificial)	175 ~ 375	130 ~ 182
17 h ~ 18 h	Luminosidade moderada	(sem iluminação artificial)	105 ~ 135	70 ~ 84
19 h ~ 20 h	Luminosidade baixa	Com iluminação artificial	25 ~ 35	35 ~ 45

Os sensores de iluminação são referenciados na programação desenvolvida como “LeLDR1”, “LeLDR2, “LeLDR3” e “LeLDR4”.

No trecho do código exibido na Figura 3.3, é possível observar que “LeLDR1” é utilizado para receber o valor da corrente elétrica tratada pelo sensor de iluminação e executar uma ação conforme programado no código. No caso abaixo, é criada uma estrutura de programação ‘*case’* que irá identificar se o valor da corrente elétrica está abaixo do valor 35 e caso esteja irá ligar um LED ou lâmpada.

```
(...)...
case 49:
      //Efetuo somente a leitura do LDR
      LeLDR1 = analogRead(lePin1); // Efetua a leitura da Porta do LDR e salva na variavel
      Serial.println(LeLDR1); // Exibe o valor da variavel
      break;
case 50:
      //Efetuo a Leitura do LDR e caso escuro liga rele e caso claro desliga rele
      LeLDR1 = analogRead(lePin1); // Efetua a leitura da Porta do LDR e salva na variavel
      Serial.println(LeLDR1); // Exibe o valor da variavel
      if (LeLDR1 < 35)  // O valor 35 foi considerado um ambiente escuro
      {
        digitalWrite(ledPin, HIGH); // Liga o Led 13 para dizer que
                                    // o ambiente está escuro, onde
                                    // será enviado o sinal para ligar
                                    // a luz se necessário
        digitalWrite(escrevePin1,HIGH); // Liga o Rele
      }
(...)...
```

FIGURA 3.38 – TRECHO DE CÓDIGO DO ARDUÍNO

3.4. – SENSOR DE PRESENÇA

O sensor de presença atuará apenas como dispositivo de leitura, que será utilizada pelo Arduíno para execução de comandos.

Os sensores estão ligados nos pinos digitais do Arduíno, de números 10, 11, 12 e 13 do Arduíno. Da mesma forma que os relés, os sensores de presença estão ligados à fonte de energia de 12 V do Arduíno e ao pino terra.

Os sensores de presença são referenciados na programação desenvolvida como “LePIR1”, “LePIR2, “LePIR3” e “LePIR4”.

3.5. – CONTROLE DO ARDUÍNO

O microcontrolador possui o controle de todas as ações ocorrentes no sistema, mas para isso, é necessário desenvolver uma programação capaz de identificar e tratar todas as informações do sistema.

3.5.1. – PROGRAMAÇÃO DE CONTROLE DO ARDUÍNO

A programação inserida dentro do Arduíno, utilizada para controlar o sistema, foi desenvolvida em linguagem *Processing* (a qual é baseada em C e C++). Para que o programa seja inteligível tanto para o Arduíno quanto para um humano, é necessário definir e nomear as portas do Arduíno no programa de forma semelhante à exibida na Figura 3.4.

```
#define ledPin 13       // Led de Sinalização do Status do LDR
#define lePin1 5        // Porta de Leitura do LDR 1
#define lePin2 4        // Porta de Leitura do LDR 2
#define escrevePin1 8   //Aciona o Rele 1
#define escrevePin2 9   //Aciona o Rele 2

int LeLDR1 = 0;         // Leitura do LDR 1
int LeLDR2 = 0;         // Leitura do LDR 2
int LeSerial = 0;       // Leitura da Serial
```

FIGURA 3.49 – TRECHO DO CÓDIGO NO ARDUÍNO

3.5.2. – CÓDIGOS DE CONTROLE

Para que o projeto funcione de forma efetiva, é necessário utilizar códigos de controle a fim de informar ao *hardware* qual ação deve ser executada e em qual ambiente. Estes códigos encontram-se listados no Quadro 04.

QUADRO 04 – CÓDIGOS DE CONTROLE

CÓDIGO	DESCRIÇÃO	AMBIENTE
1	Lê luminosidade do ambiente	Ambiente 1
2	Lê luminosidade do ambiente e desliga ou liga dependendo da luminosidade	
3	Liga lâmpada	
4	Desliga lâmpada	
5	Lê luminosidade do ambiente	Ambiente 2
6	Lê luminosidade do ambiente e desliga ou liga dependendo da luminosidade	
7	Liga lâmpada	
8	Desliga lâmpada	
A	Lê luminosidade do ambiente	Ambiente 3
B	Lê luminosidade do ambiente e desliga ou liga dependendo da luminosidade	
C	Liga lâmpada	
D	Desliga lâmpada	
E	Lê luminosidade do ambiente	Ambiente 4
F	Lê luminosidade do ambiente e desliga ou liga dependendo da luminosidade	
G	Liga lâmpada	
H	Desliga lâmpada	
$	Liga/Desliga Modo Econômico	TODOS

3.5.3. – INSERÇÃO DO ALGORITMO NO ARDUÍNO

Inserir o código dentro da plataforma Arduíno é uma tarefa extremamente simples. Inicialmente, é necessário instalar os *drivers* para comunicação entre o Arduíno e o computador. Como dito anteriormente, o Arduíno possui um CI FTDI que faz a conversão do protocolo RS-232 para o padrão USB e, dessa forma, é necessário apenas conectar o Arduíno através da conexão USB ao computador e instalar os *drivers* que acompanham o dispositivo. Após o término da instalação, o computador irá apresentar uma porta de comunicação intitulada COM – que é uma interface RS-232 virtual emulada para se comunicar com o dispositivo através da interface USB.

Após o término da instalação, uma porta COM irá aparecer no ‘Gerenciador de Dispositivos’ do Microsoft Windows de forma semelhante à exibida na Figura 3.5

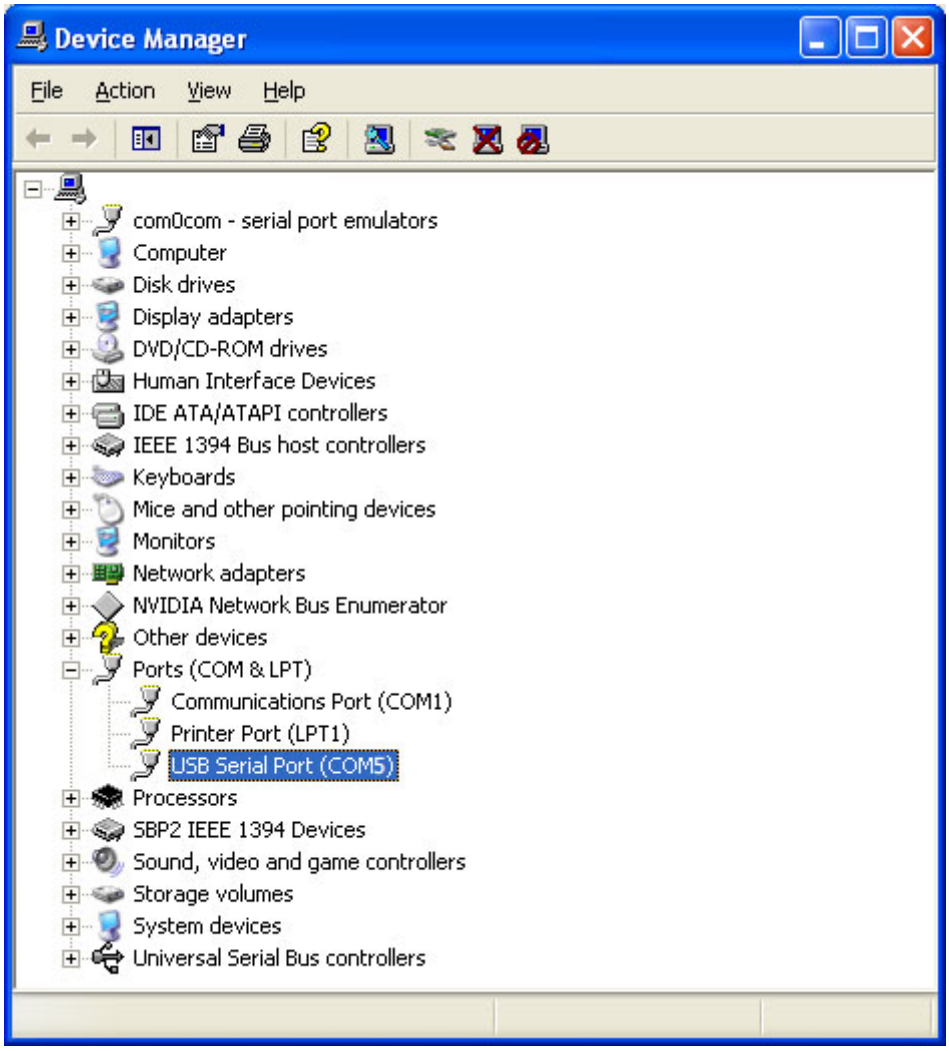

FIGURA 3.5 – PORTA VIRTUAL ‘COM’

Especificamente neste caso, a porta criada foi a COM5, mas o sistema operacional poderia ter atribuído outro número para a porta COM, como por exemplo, COM6, COM7 e assim sucessivamente. De posse da numeração da porta COM, deve-se ajustar o *script*

‘*ardu_script.php*’ informando corretamente o número da porta de comunicação, de forma semelhante à exibida na Figura 3.6.

```php
<?php
echo "\n\n INICIO DO SCRIPT \n\n";
// Para manter a porta aberta
// Conecta na porta
$port = fopen('com5', 'w+');
sleep(3);
echo "\n\n Porta Iniciada \n\n";

$file_entrada = "st_controle.txt";
$file_saida = "ardu_out.txt";
```

FIGURA 3.6 – AJUSTE DA PORTA ‘COM’ NO CÓDIGO

Para inserir o código no Arduíno é necessário utilizar a própria interface de programação desenvolvida pela equipe do Arduíno. A Figura 3.7 exibe a tela da interface de desenvolvimento e destaque para o botão de ‘*upload’*.

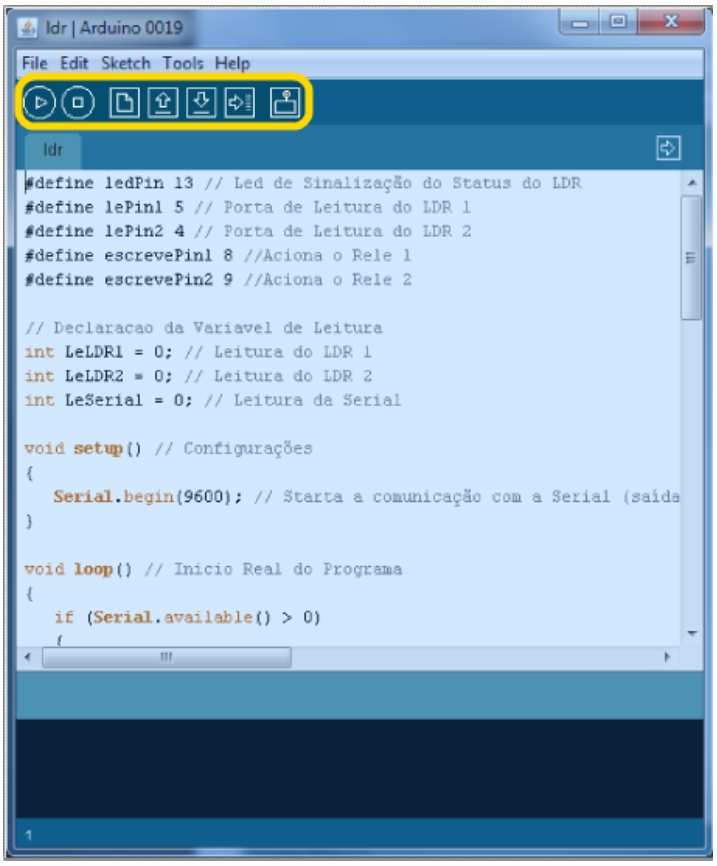

FIGURA 3.7 – AMBIENTE DE DESENVOLVIMENTO ARDUÍNO

 *Upload*;

Como dito anteriormente, após a inserção do código no Arduíno não é necessário reconectar o dispositivo ao PC para carrega o código novamente. O Arduíno ficará conectado ao PC neste projeto porque há necessidade de comunicação com um servidor *web*, que no caso é o PC. Em futuras implementações neste projeto, a ligação física entre o Arduíno e o PC poderá ser desfeita através da implementação de um Arduíno *Web Server*.

3.6. – APRESENTAÇÃO DO CIRCUITO

Aqui são descritas todas as ligações entre os componentes eletrônicos para execução do projeto. Para desenhar o circuito montado na *protoboard* (Figura 3.8), foi utilizado o

software Fritzing – que é um excelente *software* para desenhar circuitos eletrônicos. Este *software* também gera automaticamente uma visão esquemática do circuito. Entretanto, os resultados visuais não foram satisfatórios para efeito didático e o esquema do circuito (Figura 3.9) foi redesenhado utilizando o *Adobe Photoshop*, uma vez que *softwares* renomados como o *Proteus*, *Eagle*, *Multisim*, *Smartdraw* e outros ainda não possuem o Arduíno duemilanove em sua base de dados para aplicação.

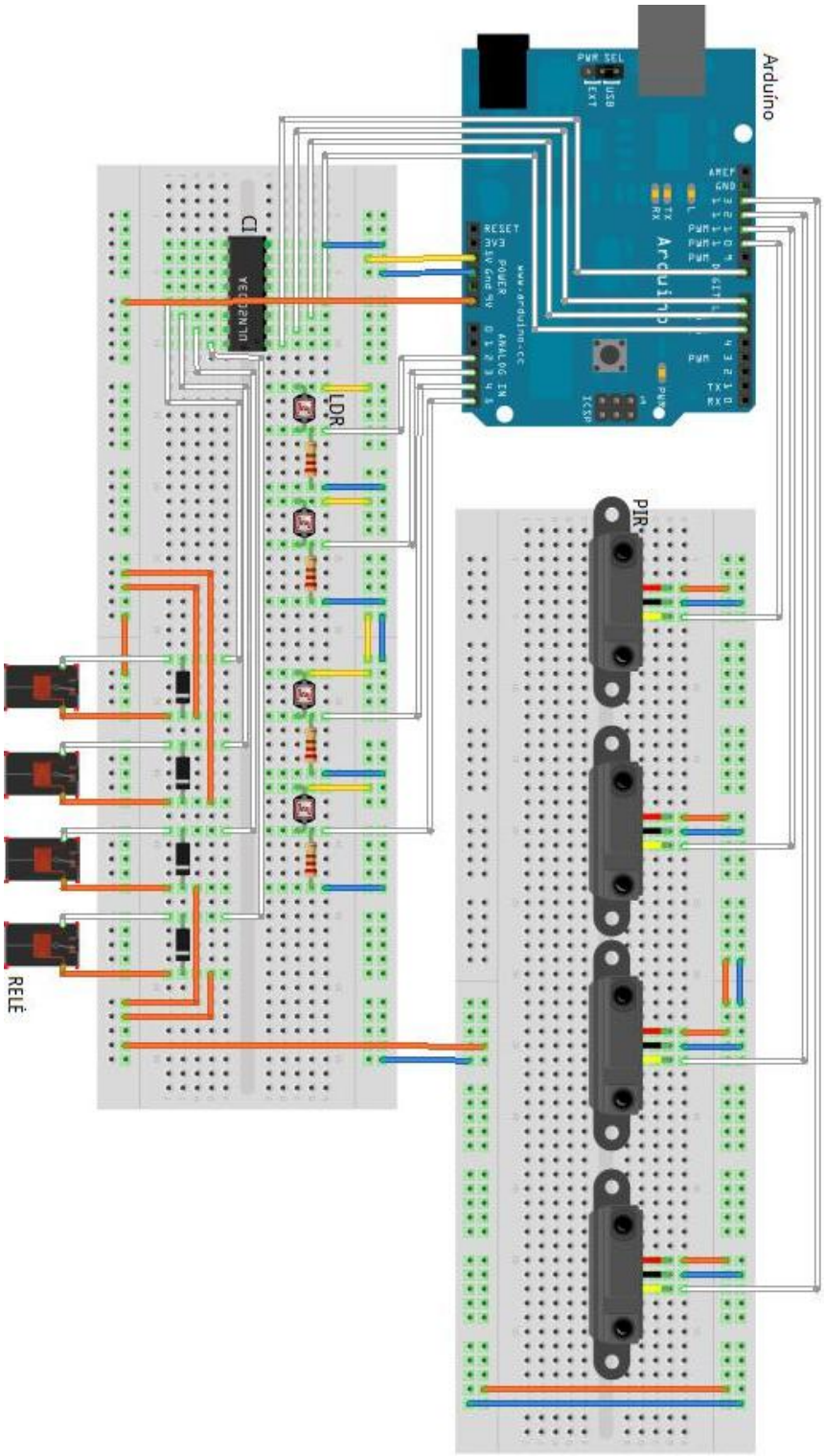

FIGURA 3.8 – DESENHO DO PROJETO NA PROTOBOARD

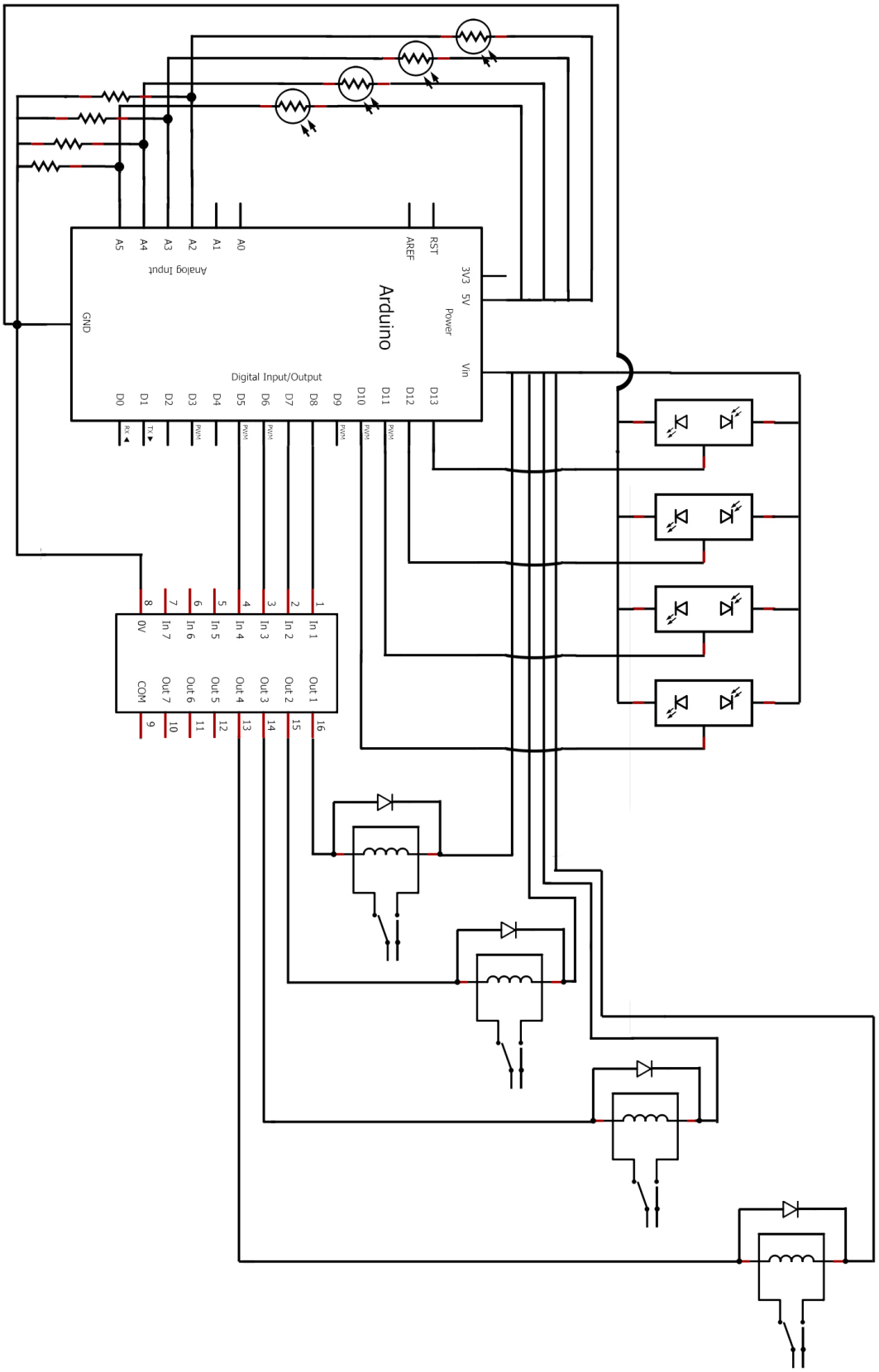

FIGURA 3.9 – CIRCUITO ELÉTRICO DO PROJETO

O Arduíno está ligado a todos os elementos eletrônicos dos circuitos, direta ou indiretamente. Abaixo, segue detalhes das ligações entre o Arduíno e o circuito integrado, sensores, relés, diodos etc.

- Os Pinos A2, A3, A4 e A5 do Arduíno tratam sinais analógicos e estão ligados a resistores (resistores cerâmicos e fotoresistores) para efetuar a leitura do nível de luminosidade dos ambientes. Uma das extremidades do resistor cerâmico está ligada ao pino terra, enquanto a outra está ligada à extremidade do fotoresistor e ao pino analógico do Arduíno. Da mesma forma, uma das extremidades do fotoresistor está ligada ao resistor cerâmico e ao pino analógico e o outro terminal está ligado à fonte de alimentação do Arduíno, de 9 V.
- Os pinos D5, D6, D7 e D8 do Arduíno estão ligados ao circuito integrado ULN2003A nos pinos ‘In 4’, ‘In 3’, ‘In 2’ e ‘In 1’, respectivamente. Recapitulando, o ULN2003A é responsável por elevar a tensão para 12 V – tensão mínima necessária para acionar o Relé.
- Os pinos ‘In 4’, ‘In 3’, ‘In 2’ e ‘In 1’ do circuito integrado ULN2003A estão ligados internamente aos pinos ‘Out 4’, ‘Out 3’, ‘Out 2’ e ‘Out 1’.
- Os pinos ‘Out 4’, ‘Out 3’, ‘Out 2’ e ‘Out 1’ circuito integrado ULN 2003A estão ligados cada um a um relé (que está conectado a um diodo retificador de forma paralela) para acionar os dispositivos de iluminação.

Após realizada a correta ligação do circuito, será necessário apenas inserir o código desenvolvido em linguagem *Processing* no Arduíno e conectá-lo a um PC para que o sistema funcione de forma correta.

3.7. – INTERFACE DE GERENCIAMENTO

A interface de gerenciamento remota é composta por uma aplicação *web*, que está publicada em um servidor. Através da *internet* é possível acessar a interface de gerenciamento do sistema de iluminação de forma remota. Acessando a interface é possível visualizar as informações da residência.

A interface foi desenvolvida utilizando-se linguagem de programação PHP, banco de dados MySQL e servidor de aplicação Apache. Todas as ferramentas necessárias para testes da aplicação foram instaladas através do pacote XAMPP, que engloba todos esses aplicativos.

3.7.1. – CÓDIGO DA INTERFACE

A interface foi desenvolvida utilizando-se PHP e MySQL visando permitir o desenvolvimento da aplicação de forma rápida e fácil.

O *script*, de codinome '*ardu_script.php*', interage com o Arduíno através da porta COM especificada, efetua a leitura do arquivo '*st_controle.txt*' (arquivo onde é escrita a solicitação do sistema *web*) para saber o que escrever para o Arduíno. O *script* também efetua a leitura do arquivo '*ardu_out.txt*' para determinar o valor de leitura do LDR, com o valor ø (ZERO) informando que a lâmpada está ligada.

Na Figura 3.10 é exibido trecho contido na programação desenvolvida para elaborar a interface *web*. Este trecho de código está contido no arquivo '*ardu_script.php*':

```php
<?PHP
// Conecta na porta
$port = fopen ('COM5', 'w+');
fwrite ($port, '9');             // Leio a primeira vez para saber luminosidade
// Obtem Codigo Luminosidade
$codLuz = fgets ($port);         // Efetua a leitura da porta
if ($codLuz < 32)                // Se a luminosidade for menor que 32 (escuro)...
                                 // ...então manda acionar a luz
{
   fwrite ($port, '1');          // escreve na porta 1 para acionar o rele
   $ack = fgets ($port);         // Obtem a resposta 0 = ok
   if ($ack == '0')
   {
      echo "OK";
   }
}
```

FIGURA 3.10 – TRECHO DO CÓDIGO DA INTERFACE *WEB* – *'ARDU_SCRIPT.PHP'*

O *script* '*index.php*' aponta para o arquivo '*my_layout.phtml*', o qual contém a página da interface de gerenciamento em si, chamando o *framework* do *layout* e as chamadas de funções para o arquivo '*core.php*'.

O *script* '*core.php*' contém as funções que efetuam chamadas no arquivo '*ardu_script.php*', e parte do seu código pode ser visualizado na Figura 3.11.

```php
<?php

function info_porta($codigo = 1)
{
    $file_saida = "st_controle.txt";
    $file_resposta = "ardu_out.txt";
    $fp = fopen($file_saida,"w");   // logo depois de ler zera o arquivo
  fwrite($fp,$codigo);
  fclose($fp);
sleep(4);
```

FIGURA 3.11 – TRECHO DO CÓDIGO DA INTERFACE *WEB* – *'CORE.PHP'*

Para armazenar as informações sobre os ambientes que estão sendo controlados, como por exemplo, nome do ambiente, status de leitura do LDR, status da lâmpada etc, é utilizado o banco de dados MySQL, que é de fácil entendimento e administração de dados.

Na Figura 3.12 pode ser visualizado trecho do código desenvolvido em SQL para criação de uma tabela:

```
CREATE TABLE IF NOT EXISTS `comodos` (
`id_comodos` int(11) NOT NULL auto_increment,
`nome_comodo` varchar(50) default NULL,
`hora_ligar` time default NULL,
`hora_desligar` time default NULL,
`is_timer` tinyint(1) default NULL,
PRIMARY KEY  (`id_comodos`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=7
```

FIGURA 3.12 – TRECHO DO CÓDIGO DO BANCO DE DADOS

3.7.2. – INICIALIZANDO O SISTEMA

Para inicializar o sistema é necessário que o servidor de páginas *web* e o banco de dados sejam inicializados. Para isso deve-se executar o pacote de serviços XAMPP e inicializar os serviços clicando no botão ‘*Start’*, que se encontra ao lado de cada um dos módulos de serviço. Há também a opção de habilitar a funcionalidade de ‘serviço’ para que os módulos sejam inicializados em conjunto com o Sistema Operacional. A tela do Painel de Controle do XAMPP é exibida na Figura 3.13.

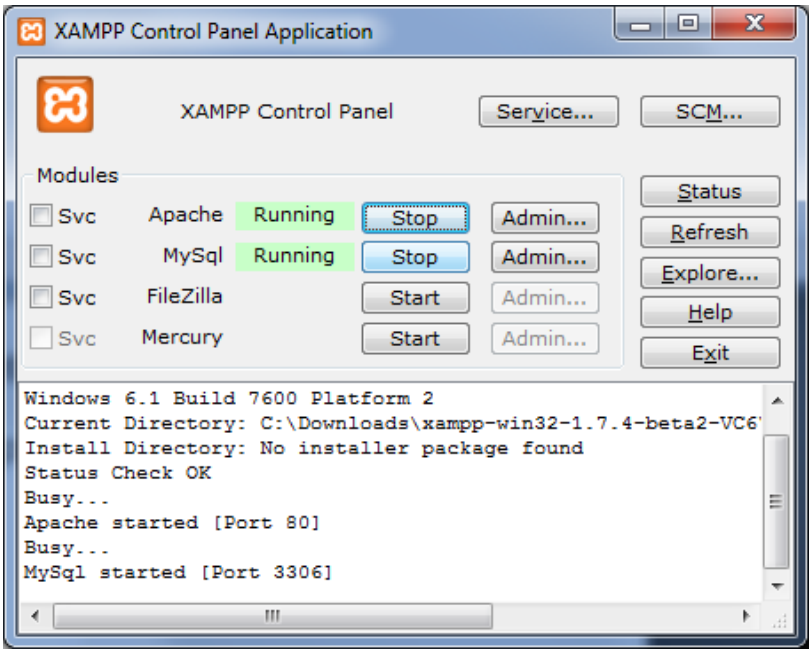

FIGURA 3.13 – PAINEL DE CONTROLE DA APLICAÇÃO XAMPP

Feito isto, é necessário executar um *script* pré-configurado que irá navegar até os diretórios corretos e acionar o executável do PHP para iniciar a interface de gerenciamento de automação. Veja a estrutura do *script* na Figura 3.14.

```
"C:                                    //Navegar até o Drive C:
CD xampp-win32-1.7.4-beta2-VC6         //Navegar até o diretório xampp...
CD XAMPP                               //Navegar até o diretório XAMPP...
CD HTDOCS                              //Navegar até o diretório HTDOCS
CD FINAL_GUSTAVO                       //Navegar até o diretório FINAL...
C:\xampp-win32-1.7.4-beta2-VC6\xampp\php\php.exe ardu_script.php    //Executar o script
```

FIGURA 3.14 – *SCRIPT* PARA INICIAR INTERFACE *WEB*

3.7.3. – AMBIENTE DE GERENCIAMENTO

Observe nas imagens abaixo que a interface de gerenciamento é simples e objetiva.

Na Figura 3.15 é possível observar o status da lâmpada. Ou seja: se ela está apagada ou acesa.

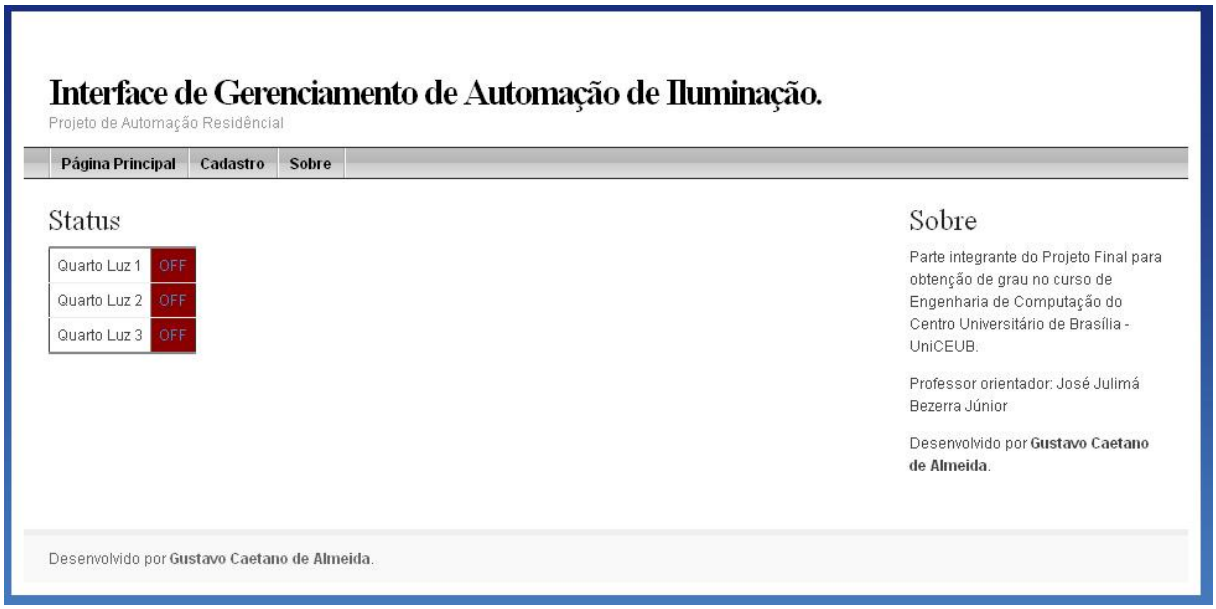

FIGURA 3.105 – INTERFACE DE GERENCIAMENTO (TELA INICIAL)

Observe na Figura 3.16 que, ao clicar em ‘Cadastro’, é apresentada a opção de programar o horário que a lâmpada deve ser ligada ou desligada.

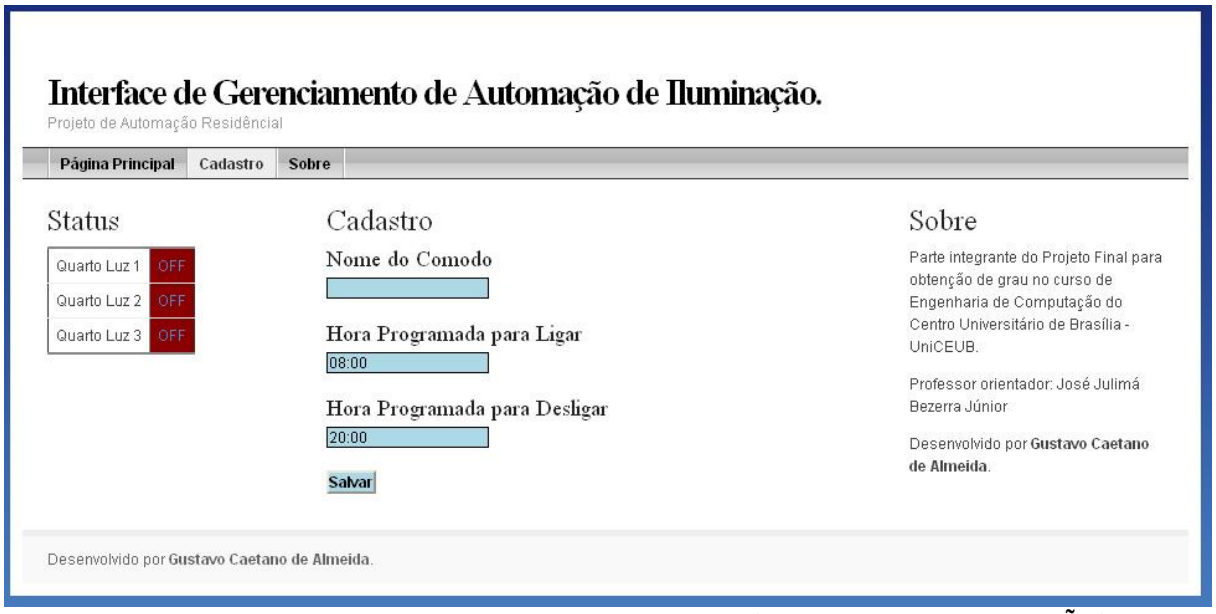

FIGURA 3.16 – INTERFACE DE GERENCIAMENTO (TELA DE PROGRAMAÇÃO)

CAPÍTULO 4 – APLICAÇÃO DA SOLUÇÃO COM RESULTADOS

4.2. – DIFICULDADES ENCONTRADAS

Apesar de o Arduíno ser uma excelente plataforma para execução de projetos de automação e afins e ter servido muito bem ao propósito do projeto, o próprio Arduíno acabou se tornando o foco das principais dificuldades encontradas durante a execução do projeto.

Foi muito difícil encontrar literatura oficial sobre a plataforma Arduíno em todos os seus aspectos: *hardware*, programação, projetos etc. Praticamente não existe literatura oficial no idioma português do Brasil. A grande fonte de informação oficial é disponibilizada no idioma inglês e para documentar o projeto foi necessário comprar livros em formato digital (*ebooks*).

Outro ponto que foi determinante para aumentar as dificuldades encontradas é a falta de *software* de desenho de esquemas eletrônicos que suportam o *hardware* do Arduíno Duemilanove.

4.3. – AVALIAÇÃO DO PROJETO

O projeto é de fácil implementação e de custo baixo, o que incentiva amadores a executarem o projeto em suas residências ou em outros locais de interesse por conta própria. No caso de implementação do projeto em algum ambiente que possua muitos pontos de iluminação, pode-se estudar a substituição do Arduíno Duemilanove por outro, como o Arduíno *Mega* que possui 4x a quantidade de conexões do Arduíno Duemilanove ou pode se estudar o empilhamento dos Arduínos utilizando conexão USB.

Um ponto importante que deve ser ressaltado é o fato de que, independente da quantidade de pontos de iluminação que seja controlada, não há necessidade de mudança dos componentes citados no projeto por outros mais caros, por exemplo.

CAPÍTULO 5 – CONSIDERAÇÕES FINAIS

5.1. – CONCLUSÃO

O objetivo do projeto foi automatizar sistemas de iluminação e gerenciá-los remotamente através de uma interface computadorizada e para atingir o objetivo foi utilizado *hardware* e *software* open source de baixo custo de aquisição e de fácil implementação.

O projeto se mostrou eficaz no seu propósito, que foi permitir o gerenciamento remoto do sistema de iluminação e permitir a redução do consumo de energia relacionado ao gasto com iluminação, uma vez que consta no projeto o inovador “modo econômico”, que pré-define que as lâmpadas ficarão acesas somente quando houver movimento no ambiente.

A utilização do Arduíno propiciou uma fácil integração com todos os dispositivos do sistema e obteve os mesmos resultados que teriam sido obtidos caso o projeto tivesse sido executado com o uso de outros microcontroladores, que teriam de ser esquematizados com diversos outros componentes para atingir o mesmo propósito do Arduíno.

Os resultados obtidos foram satisfatórios e o sistema se mostrou útil, podendo ser implementado em qualquer ambiente. Atingiu o objetivo de toda automação: propiciar comodidade e conforto para as pessoas.

5.2. – SUGESTÕES PARA TRABALHOS FUTUROS

À medida que o projeto foi sendo desenvolvido, surgiram várias idéias para agregar ao projeto e foram identificadas tecnologias para fazê-lo com o mesmo objetivo, mas utilizando alguns componentes diferentes.

É possível aproveitar boa parte da estrutura do projeto e acrescentar outros dispositivos a serem controlados, como por exemplo: temperatura do ambiente, abertura/fechamento de portas e janelas, envio de mensagens através de e-mail ou SMS (*Short Message Service*) para alertar algo, alarme anti-furto, etc.

Uma forma de executar este projeto de forma diferente seria utilizar um *shield* com suporte à tecnologia Ethernet para configurar um *Webserver* embarcado no arduíno para eliminar a necessidade de o sistema ficar ligado a um computador de forma contínua, ou utilizar *shield*s de comunicação sem fio para implementar a conexão entre os dispositivos de iluminação e o Arduíno, eliminando, obviamente, a necessidade de reformas físicas na estrutura do ambiente para receber as várias malhas de fios necessárias para implementação do projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

[01] ACHOUR, Mehdi et al.“**PHP Manual**”. Disponível em: <http://www.php.net/manual/en/index.php>. Acesso em: 10 abr. 2010.

[02] AFRAC. “**Automação comercial: R$ 1,4 bi no Brasil**”. Disponível em: <http://www.baguete.com.br/noticias/hardware/01/04/2010/automacao-comercial-r-14-bi-no-brasil>. Acesso em: 16 nov. 2010.

[03] APACHE. Disponível em: <http://www.apache.org/>. Acesso em: 25 set. 2010.

[04] ARDUINO. “**Arduino Duemilanove**”. Disponível em: <http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove>. Acesso em: 11 set. 2010.

[05] ARDUINO. “**Language Reference**”. Disponível em: <http://arduino.cc/en/Reference/HomePage>. Acesso em: 06 set. 2010.

[06] ATMEL DATASHEET. “**8-bit *AVR*® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash**”. Disponível em: <http://www.atmel.com/dyn/resources/prod_documents/doc8271.pdf>. Acesso em: 06 jul. 2010.

[07] AURESIDE. “**Projeto de Interiores e Automação**”. Disponível em: <http://www.aureside.org.br/artigos/default.asp?file=01.asp&id=72>. Acesso em: 18 nov 2010.

[08] BOLZANI, Caio Augustus Morais. **Residências Inteligentes.** 1. ed. São Paulo, Brasil: Editora Livraria da Física, 2004.

[09] BRAGA, Newton C. **Curso Básico de Eletrônica.** 5. ed. São Paulo, Brasil: Nova Saber, Julho, 2009. 208p

[10] CAMARGO, Roberto Abdeunur. **Luz e Cena: Processo de Comunicação Co-Evolutivos**, São Paulo. Tese de Doutorado. Programa de Comunicação e Semiótica: PUC-SP, 2006

[11] DE MORAES, Cícero Couto. CASTRUCCI, Plínio de Lauro. **Engenharia de Automação Industrial.** 2. ed. São Paulo, Brasil: Editora LTC, 2001. 295p

[12] FONSECA, Érika Guimarães Pereira da Fonseca. BEPPU, Mathyan Motta. **Apostila Arduino**, Rio de Janeiro: Universidade Federal Fluminense, Outubro, 2010.

[13] HALL, Mark. Computer World: "**MySQL Breaks Into the Data Center**". Disponível em: <http://www.computerworld.com/databasetopics/data/software/story/0,10801,85900,00.html>. Acesso em: 11 set. 2010.

[14] MANCINI, Ron. **Op Amps for Everyone: Design Reference/Advanced Analog Products (SLOD006B)**. Estados Unidos da América: Texas Instruments, August, 2002.

[15] MOTOROLA. "**Motorola MC33269 Datasheet**".

[16] NETCRAFT "**August 2010 Web Server Survey**". Disponível em: <http://news.netcraft.com/ archives/2010/08/11/august-2010-web-server-survey-4.html>. Acesso em: 19 set. 2010.

[17] NUNES, Paulo; Artigo: **Conceito de Fluxograma**. Disponível em: <http://www.knoow.net/ cienceconempr/ gestao fluxograma.htm>. Acesso em: 15 ago. 2010.

[18] OLIVEIRO, Carlos Antonio José. **Faça um site – PHP4 com Base de Dados MySQL Orientado por Projeto**, São Paulo: Editora Érica, 2001

[19] OXER, Jonathan. BLEMINGS, Hugh. **Practical Arduino: Cool Projects for Open Source Hardware (Technology in Action)**. Estados Unidos da América: Apress, 2009.

[20] PPA. "**Manual do Sensor de Presença**".

[21] REAS, Casey. FRY, Ben. **Getting Started with Processing**. Estados Unidos da América: O'Reilly Media, Junho, 2010. 208p.

[22] SANTOS, Nuno Pessanha. **Apostila Introdução ao Arduíno**, Lisboa, Portugal: Escola Naval: Departamento de Engenheiros Navais – Ramo de Armas e Eléctronica. Disponível em: <http://www.isegi.unl.pt/docentes/vlobo/escola_naval/MFC/Slides%20 Arduino.pdf>. Acesso em: 15 out. 2010.

[23] STOUT, David. **Handbook of Operational Amplifier Circuit Design**, Estados Unidos da América: Editora McGraw-Hill Inc., Junho, 1976

[24] TEXAS INSTRUMENTS. "**CI 2003A Datasheet**". Disponível em: <http://www.datasheetcatalog .org/datasheet/texasinstruments/uln2003a.pdf>. Acesso em: 14 out. 2010.

[25] XAMPP. Disponível em: <http://www.apachefriends.org/en/xampp.html>. Acesso em: 17 set. 2010.

APÊNDICE A – PROGRAMA INSERIDO NO ARDUÍNO

```
#define ledPin 13 // Led de Sinalização do Status do LDR

#define lePin1 1 // Porta de Leitura do LDR 1
#define lePin2 2 // Porta de Leitura do LDR 2
#define lePin3 3 // Porta de Leitura do LDR 2
#define lePin4 4 // Porta de Leitura do LDR 2

#define escrevePin1 8 //Aciona o Rele 1
#define escrevePin2 9 //Aciona o Rele 2
#define escrevePin3 10 //Aciona o Rele 3
#define escrevePin4 11 //Aciona o Rele 4

#define sensorPin1 4 //Sensor Amb 1
#define sensorPin2 5 //Sensor Amb 2
#define sensorPin3 6 //Sensor Amb 3
#define sensorPin4 7 //Sensor Amb 4


// Declaracao da Variavel de Leitura
int LeLDR1 = 0; // Leitura do LDR 1
int LeLDR2 = 0; // Leitura do LDR 2
int LeLDR3 = 0; // Leitura do LDR 3
int LeLDR4 = 0; // Leitura do LDR 4

boolean ModoEco = false; // Controle do Modo/Economico
int PirMov = 0; // PIR
```

```
int LuxEscuro = 35;


int LeSerial = 0; // Leitura da Serial


void setup() // Configurações
{
  Serial.begin(9600); // Starta a comunicação com a Serial (saída de informações)
  pinMode(sensorPin1,INPUT); //setando sinal como entrada pino do sensor movimento
  digitalWrite(sensorPin1,HIGH); //setando *pull-up evitando ocilação sinal
}


void loop() // Inicio Real do Programa
{
  if (Serial.available() > 0)
  {
    LeSerial = Serial.read();
    switch (LeSerial) {
      case 36: // $
          // Efetua a verificacao para colocar em modo Economico
          if (ModoEco)
          {
            ModoEco = false;
            Serial.println('0'); // retorna o ack
          }
          else
          {
            ModoEco = true;
            Serial.println('1'); // retorna o ack
          }
```

```
    break;

//////////////////////

case 49: // 1
    //Efetuo somente a leitura do LDR 1
    LeLDR1 = analogRead(lePin1); // Efetua a leitura da Porta do LDR e salva na variavel
    PirMov = digitalRead(sensorPin1); // Verifico se existe movimento no amb 1
    if (ModoEco)
    {
     if (PirMov==HIGH)
     {
       // Existe Alguem no Ambiente
     }
    }
    Serial.println(LeLDR1); // Exibe o valor da variavel
    break;
case 50: // 2
    //Efetuo a Leitura do LDR e caso escuro liga rele e caso claro desliga rele
    LeLDR1 = analogRead(lePin1); // Efetua a leitura da Porta do LDR e salva na variavel
    Serial.println(LeLDR1); // Exibe o valor da variavel
    if (LeLDR1 < LuxEscuro)  // O valor 35 foi considerado um ambiente escuro
    {
     if (ModoEco)
     {
       PirMov = digitalRead(sensorPin1); // Verifico se existe movimento no amb 1
       if (PirMov==HIGH)
       {
         // Existe Alguem no Ambiente
```

```
            digitalWrite(ledPin, HIGH); // Liga o Led 13 para dizer que o ambiente está escuro, onde será enviado o sinal para ligar a luz se necessário
            digitalWrite(escrevePin1,HIGH); // Liga o Rele
           }
          }
         }
         else
         {
          digitalWrite(ledPin, LOW); // desliga o Led 13 para dizer que o ambiente está escuro, onde será enviado o sinal para desligar a luz se necessário
          digitalWrite(escrevePin1,LOW); // Desliga o Rele
         }
         delay(1000); // Pausa delay(milisegundos)
         break;
      case 51: // 3
         // Somente LIGA o rele independente do LDR
         ModoEco = digitalRead(sensorPin1); // Verifico se existe movimento no amb 1
         if (ModoEco)
         {
          PirMov = digitalRead(sensorPin1); // Verifico se existe movimento no amb 1
          if (PirMov==HIGH)
          {
           digitalWrite(escrevePin1,HIGH); // Liga o Rele
           digitalWrite(ledPin, HIGH);
          }
          Serial.println('!');
         }
         break;
      case 52: // 4
         // Somente DESLIGA o rele independente do LDR
```

```
      digitalWrite(escrevePin1,LOW); // Desliga o Rele
      digitalWrite(ledPin, LOW);
      Serial.println('!'); // retorna o ack
      break;

//////////////////

case 53: // 5
      //Efetuo somente a leitura do LDR 2
      LeLDR2 = analogRead(lePin2); // Efetua a leitura da Porta do LDR e salva na variavel
      if (ModoEco)
      {
        if (PirMov==HIGH)
        {
          // Existe Alguem no Ambiente
        }
      }
      Serial.println(LeLDR2); // Exibe o valor da variavel
      break;
case 54: // 6
      //Efetuo a Leitura do LDR 2 e caso escuro liga rele e caso claro desliga rele
      LeLDR2 = analogRead(lePin2); // Efetua a leitura da Porta do LDR e salva na variavel
      Serial.println(LeLDR2); // Exibe o valor da variavel
      if (LeLDR2 < LuxEscuro)  // O valor 35 foi considerado um ambiente escuro
      {
        if (ModoEco)
        {
          PirMov = digitalRead(sensorPin2); // Verifico se existe movimento no amb 1
          if (PirMov==HIGH)
```

```
            {
              // Existe Alguem no Ambiente
              digitalWrite(ledPin, HIGH); // Liga o Led 13 para dizer que o ambiente está escuro, onde será enviado o sinal para ligar a luz se necessário
              digitalWrite(escrevePin2,HIGH); // Liga o Rele
            }
          }
        }
        else
        {
          digitalWrite(ledPin, LOW); // desliga o Led 13 para dizer que o ambiente está escuro, onde será enviado o sinal para desligar a luz se necessário
          digitalWrite(escrevePin2,LOW); // Desliga o Rele
        }
        delay(1000); // Pausa delay(milisegundos)
        break;
    case 55: // 7
        // Somente LIGA o rele 2 independente do LDR
        if (ModoEco)
        {
          if (PirMov==HIGH)
          {
            digitalWrite(escrevePin2,HIGH); // Liga o Rele 2
            digitalWrite(ledPin, HIGH);
          }
        }
        Serial.println('!');
        break;
    case 56: // 8
        // Somente DESLIGA o rele 2 independente do LDR
```

```
      digitalWrite(escrevePin2,LOW); // Desliga o Rele
      digitalWrite(ledPin, LOW);
      Serial.println('!'); // retorna o ack
      break;
//////////

case 65: // A
      //Efetuo somente a leitura do LDR 2
      LeLDR3 = analogRead(lePin3); // Efetua a leitura da Porta do LDR e salva na variavel
      if (ModoEco)
      {
       if (PirMov==HIGH)
       {
         // Existe Alguem no Ambiente
       }
      }
      Serial.println(LeLDR3); // Exibe o valor da variavel
      break;
case 66: // B
      //Efetuo a Leitura do LDR 2 e caso escuro liga rele e caso claro desliga rele
      LeLDR3 = analogRead(lePin3); // Efetua a leitura da Porta do LDR e salva na variavel
      Serial.println(LeLDR3); // Exibe o valor da variavel
      if (LeLDR3 < LuxEscuro)  // O valor 35 foi considerado um ambiente escuro
      {
       if (ModoEco)
       {
         PirMov = digitalRead(sensorPin3); // Verifico se existe movimento no amb 1
         if (PirMov==HIGH)
         {
```

```
            // Existe Alguem no Ambiente
            digitalWrite(ledPin, HIGH); // Liga o Led 13 para dizer que o ambiente está escuro, onde será enviado o sinal para ligar a luz se necessário
            digitalWrite(escrevePin3,HIGH); // Liga o Rele
           }
          }
         }
         else
         {
          digitalWrite(ledPin, LOW); // desliga o Led 13 para dizer que o ambiente está escuro, onde será enviado o sinal para desligar a luz se necessário
          digitalWrite(escrevePin3,LOW); // Desliga o Rele
         }
         delay(1000); // Pausa delay(milisegundos)
         break;
    case 67: // C
         // Somente LIGA o rele 2 independente do LDR
         if (ModoEco)
         {
          if (PirMov==HIGH)
          {
           digitalWrite(escrevePin3,HIGH); // Liga o Rele 2
           digitalWrite(ledPin, HIGH);
          }
         }
         Serial.println('!');
         break;
    case 68: // D
         // Somente DESLIGA o rele 2 independente do LDR
         digitalWrite(escrevePin3,LOW); // Desliga o Rele
```

```
      digitalWrite(ledPin, LOW);
      Serial.println('!'); // retorna o ack
      break;

//////////////////

case 69: // E
      //Efetuo somente a leitura do LDR 2
      LeLDR4 = analogRead(lePin4); // Efetua a leitura da Porta do LDR e salva na variavel
      if (ModoEco)
      {
       if (PirMov==HIGH)
       {
         // Existe Alguem no Ambiente
       }
      }
      Serial.println(LeLDR4); // Exibe o valor da variavel
      break;
case 70: // F
      //Efetuo a Leitura do LDR 2 e caso escuro liga rele e caso claro desliga rele
      LeLDR4 = analogRead(lePin4); // Efetua a leitura da Porta do LDR e salva na variavel
      Serial.println(LeLDR3); // Exibe o valor da variavel
      if (LeLDR4 < LuxEscuro)  // O valor 35 foi considerado um ambiente escuro
      {
       if (ModoEco)
       {
         PirMov = digitalRead(sensorPin4); // Verifico se existe movimento no amb 1
         if (PirMov==HIGH)
         {
```

```
            // Existe Alguem no Ambiente
            digitalWrite(ledPin, HIGH); // Liga o Led 13 para dizer que o ambiente está escuro, onde será enviado o sinal para ligar a luz se necessário
            digitalWrite(escrevePin4,HIGH); // Liga o Rele
           }
          }
         }
         else
         {
          digitalWrite(ledPin, LOW); // desliga o Led 13 para dizer que o ambiente está escuro, onde será enviado o sinal para desligar a luz se necessário
          digitalWrite(escrevePin4,LOW); // Desliga o Rele
         }
         delay(1000); // Pausa delay(milisegundos)
         break;
    case 71: // G
         // Somente LIGA o rele 2 independente do LDR
         if (ModoEco)
         {
          if (PirMov==HIGH)
          {
           digitalWrite(escrevePin4,HIGH); // Liga o Rele 2
           digitalWrite(ledPin, HIGH);
          }
         }
         Serial.println('!');
         break;
    case 72: // H
         // Somente DESLIGA o rele 2 independente do LDR
         digitalWrite(escrevePin4,LOW); // Desliga o Rele
```

```
            digitalWrite(ledPin, LOW);
            Serial.println('!'); // retorna o ack
            break;

        default:
          // Se nao acontecer nenhuma das opções acima cai no default
          // default eh opcional
        break;
      }
    }
}
```

APÊNDICE B – CÓDIGO DA INTERFACE *WEB*

Código do *script* ardu_script.php

```
<?php
echo "\n\n INICIO DO SCRIPT \n\n";
// Para manter a porta aberta
// Conecta na porta
$port = fopen('com4', 'w+');
sleep(3);
echo "\n\n Porta Iniciada \n\n";

$file_entrada = "st_controle.txt";
$file_saida = "ardu_out.txt";

$var_controle = "";

while ($var_controle != 'CLOSE')
{
            // pega as informações do arquivo 'st_controle.txt'
  if (file_exists($file_entrada))
  {
    $fd       = fopen ($file_entrada, "r");
    if (filesize ($file_entrada) != 0)
    {
          $status   = fread ($fd, filesize ($file_entrada));
               fclose ($fd);
               $fp = fopen($file_entrada,"w"); // logo depois de ler zera o arquivo
               fwrite($fp,"");
```

```
            fclose($fp);
          }
}
else
{
                echo "\n\n !!! ARQUIVO NAO EXISTE !!! \n\n";
                $var_controle = 'CLOSE';
 $status = "";
}

switch ($status)
{
                        case '1' :
                          // Escrevo o código do arquivo para efetuar a leitura da luminosidade
                          fwrite($port,'1\n');
                          sleep(1);
                                // Efetua a leitura da porta
                                $codLuz = fgets($port);
                                // Agora vou escrever no arquivo para o sistema
                                $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
                          fwrite($fp,$codLuz);
                          fclose($fp);
                          echo "\n !!! ARDUINO OK - ".$codLuz."!!! \n";
                        break;
                        case '2' :
                                // Escrevo o código do arquivo para efetuar a leitura da luminosidade
                          fwrite($port,'2\n');
                          sleep(1);
                                // Efetua a leitura da porta
```

```
            $codLuz = fgets($port);
            // Agora vou escrever no arquivo para o sistema
            $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz."!!! \n";
break;
case '3' :
            // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'3\n');
  sleep(1);
            // Efetua a leitura da porta
            $codLuz = fgets($port);
            // Agora vou escrever no arquivo para o sistema
            $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;
case '4' :
            // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'4\n');
  sleep(1);
            // Efetua a leitura da porta
            $codLuz = fgets($port);
            // Agora vou escrever no arquivo para o sistema
            $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
```

```
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;

///////

case '5' :
  // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'5\n');
  sleep(1);
        // Efetua a leitura da porta
        $codLuz = fgets($port);
        // Agora vou escrever no arquivo para o sistema
        $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;
case '6' :
        // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'6\n');
  sleep(1);
        // Efetua a leitura da porta
        $codLuz = fgets($port);
        // Agora vou escrever no arquivo para o sistema
        $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;
```

```
case '7' :
        // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'7\n');
  sleep(1);
        // Efetua a leitura da porta
        $codLuz = fgets($port);
        // Agora vou escrever no arquivo para o sistema
        $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;
case '8' :
        // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'8\n');
  sleep(1);
        // Efetua a leitura da porta
        $codLuz = fgets($port);
        // Agora vou escrever no arquivo para o sistema
        $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;

///////

case 'A' :
  // Escrevo o código do arquivo para efetuar a leitura da luminosidade
```

```
  fwrite($port,'A\n');
  sleep(1);
            // Efetua a leitura da porta
            $codLuz = fgets($port);
            // Agora vou escrever no arquivo para o sistema
            $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;
case 'B' :
            // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'B\n');
  sleep(1);
            // Efetua a leitura da porta
            $codLuz = fgets($port);
            // Agora vou escrever no arquivo para o sistema
            $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;
case 'C' :
            // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'C\n');
  sleep(1);
            // Efetua a leitura da porta
            $codLuz = fgets($port);
            // Agora vou escrever no arquivo para o sistema
```

```
            $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;
case 'D' :
            // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'D\n');
  sleep(1);
            // Efetua a leitura da porta
            $codLuz = fgets($port);
            // Agora vou escrever no arquivo para o sistema
            $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;

///////

case 'E' :
  // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'E\n');
  sleep(1);
            // Efetua a leitura da porta
            $codLuz = fgets($port);
            // Agora vou escrever no arquivo para o sistema
            $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
```

```
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;
case 'F' :
        // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'F\n');
  sleep(1);
        // Efetua a leitura da porta
        $codLuz = fgets($port);
        // Agora vou escrever no arquivo para o sistema
        $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;
case 'G' :
        // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'G\n');
  sleep(1);
        // Efetua a leitura da porta
        $codLuz = fgets($port);
        // Agora vou escrever no arquivo para o sistema
        $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;
case 'H' :
        // Escrevo o código do arquivo para efetuar a leitura da luminosidade
```

```
  fwrite($port,'H\n');
  sleep(1);
            // Efetua a leitura da porta
            $codLuz = fgets($port);
            // Agora vou escrever no arquivo para o sistema
            $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! \n";
break;

///////

case '$' :
            // Escrevo o código do arquivo para efetuar a leitura da luminosidade
  fwrite($port,'$\n');
  sleep(1);
            // Efetua a leitura da porta
            $codLuz = fgets($port);
            // Agora vou escrever no arquivo para o sistema
            $fp = fopen($file_saida,"w"); // abre o arquivo de resposta
  fwrite($fp,$codLuz);
  fclose($fp);
  echo "\n !!! ARDUINO OK - ".$codLuz." !!! $$$\n";
break;

///////

case '0' :
```

```
                                    $var_controle = 'CLOSE';
                                    //Fecho a porta
                                    echo "\n\n FECHANDO A PORTA \n\n";
                                    fclose($port);
                            break;
                            default :
                            break;
    }
    sleep(1);
}
echo "\n\n FIM DO SCRIPT \n\n";
?>
```

Código do *script* ardu_hora.php

```
<?php

echo "\n\n INICIALIZANDO SCRIPT DE HORA \n\n";

require_once ( 'core.php' );

// Tratamento com o Banco de Dados
$mysql_con = mysql_connect('localhost', 'root', '');
if (!$mysql_con) {
    die('Não foi possivel Conectar : ' . mysql_error());
}
mysql_select_db("tcc_gustavo",$mysql_con) or die(mysql_error());

echo "\n\n CONEXAO COM BANCO EFETUADA \n\n";
```

```
while (1)
{

	$query = "SELECT * FROM comodos ORDER BY nome_comodo";
	$resultado = mysql_query($query,$mysql_con);
	while ($linha = mysql_fetch_array($resultado))
	{
		if ($linha['is_timer'] == 1)
		{
			if ((substr($linha['hora_ligar'],0,2) == date("H")) && (substr($linha['hora_ligar'],3,2)
== date("i")))
			{
				info_porta(chr($linha['id_comodos'] + 2));
				echo "Enviou o Ligar \n";
			}
			elseif ((substr($linha['hora_desligar'],0,2) == date("H")) &&
(substr($linha['hora_desligar'],3,2) == date("i")))
			{
				info_porta(chr($linha['id_comodos'] + 3));
				echo "Enviou o Ligar \n";
			}
		}
	}
	sleep(30);
}
?>
```

Código do *script* core.php

```
<?php


function info_porta($codigo = '1')
{
	$file_saida = "st_controle.txt";
	$file_resposta = "ardu_out.txt";
	$fp = fopen($file_saida,"w"); // logo depois de ler zera o arquivo
  fwrite($fp,$codigo);
  fclose($fp);


	sleep(4);


	if (file_exists($file_resposta))
	{
		$fd = fopen ($file_resposta, "r");
		if (filesize ($file_resposta) != 0)
		{
		$status = fread ($fd, filesize ($file_resposta));
     fclose ($fd);
     $fp = fopen($file_resposta,"w"); // logo depois de ler zera o arquivo
     fwrite($fp,"");
     fclose($fp);
    }
	}
	else
	{
		$status = 99;
	}
```

```
        return $status;
}
```

Código do *script* my_layout.phtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>Interface de Gerenciamento de Automa&ccedil;&atilde;o de Ilumina&ccedil;&atilde;o.</title>

<link href="css/my_layout.css" rel="stylesheet" type="text/css"/>
<!--[if lte IE 7]>
<link href="css/patches/patch_my_layout.css" rel="stylesheet" type="text/css" />
<![endif]-->

<script language="javascript">

        var show_div = true;
        var txt_div = '';

        function show_hide_div(pass)
        {
                if (show_div)
                {
                        show_div = false;
                        txt_div = 'hidden';
                }
```

```
                else
                {
                        show_div = true;
                        txt_div = 'visible';
                }

                var divs = document.getElementsByTagName('div');
                for(i=0;i<divs.length;i++)
                {
                        //if they are 'see' divs
                        if(divs[i].id.match(pass))
                        {
                                if (document.getElementById) // DOM3 = IE5, NS6
                                        divs[i].style.visibility=txt_div; //"hidden";// show/hide
                                else
                                        if (document.layers) // Netscape 4
                                                document.layers[divs[i]].display = txt_div; //'hidden';
                                        else // IE 4
                                                document.all.hideshow.divs[i].visibility = txt_div; //
'hidden';
                        }
                }
        }
</script>

</head>
<body>
<!-- skip link navigation -->
<ul id="skiplinks">
```

```
        <li><a class="skip" href="#nav">Skip to navigation (Press Enter).</a></li>
        <li><a class="skip" href="#col3">Skip to main content (Press Enter).</a></li>
</ul>

<?php

set_time_limit(60);

// Tratamento com o Banco de Dados
$mysql_con = mysql_connect('localhost', 'root', 'vertrigo');
if (!$mysql_con) {
   die('Não foi possivel Conectar : ' . mysql_error());
}
mysql_select_db("tcc_gustavo",$mysql_con) or die(mysql_error());

require_once ( 'core.php' );



        // Ligar ou desligar a luz de acordo com o Link

        switch ($_GET['func'])
        {
                case 'acao' :
                        $status = info_porta(chr($_GET['op']));
                        $_GET['func'] = '';
                break;
                case 'cadastra' :
                        $n_comodo = $_REQUEST["nome_comodo"];
                        $h_ligar = $_REQUEST["hora_ligar"];
```

```
$h_desligar = $_REQUEST["hora_desligar"];
if (($h_ligar == "") || ($h_desligar == ""))
{
	$is_timer = 0;
}
else
{
	$is_timer = 1;
}

$query = "SELECT id_comodos FROM comodos";
$resultado = mysql_query($query,$mysql_con);
while ($linha = mysql_fetch_array($resultado)) {
	$ultimo_id = $linha['id_comodos'];
}

if ($ultimo_id == '')
{
	$primeiro_id = 49;
}
else
{
	if ($ultimo_id == 53)
	{
		$primeiro_id = 65;
	}
	else
	{
		$primeiro_id = $ultimo_id + 4;
```

```
                    }
                }

                $query = "INSERT INTO comodos VALUES
('".$primeiro_id."','".$n_comodo."','".$h_ligar."','".$h_desligar."','".$is_timer."')";

                if(mysql_query($query,$mysql_con))
                {
                    echo "<script>alert('Comodo cadastrado com sucesso!!!');</script>";
                }
                else{
                    echo "<script>alert('Erro no cadastramento do comodo!!!');</script>";
                }
                $_GET['func'] = '';
            break;
            case 'eco' : // Verifico o Status do Modo Economico
                $status = info_porta('$');
                if (trim($status) == '0')
                {
                    $txt_eco = ""; // Modo Eco Desligado
                }
                else
                {
                    $txt_eco = "class=\"active\""; // Modo Eco Ligado

                }
                $status = '';
                $_GET['func'] = '';

            break;
```

```
}

$query = "SELECT nome_comodo, id_comodos FROM comodos ORDER BY id_comodos";
$resultado = mysql_query($query,$mysql_con);
$i = 0;
while ($linha = mysql_fetch_array($resultado)) {
        $arr_comodos[$i] = $linha['nome_comodo'];
        $arr_comodos_id[$i] = $linha['id_comodos'];
        $i++;
}

//print_r($arr_comodos_id);
$qtd_comodos = mysql_num_rows($resultado); // Para obter a quantidade de comodos cadastrados;

for ($i = 0; $i < $qtd_comodos; $i++)
{ echo "--->".$arr_comodos_id[$i];
        // Verificação dos Status
        $status_porta = info_porta(chr($arr_comodos_id[$i]));
        if ($status_porta > 150) // Significa que a luz esta ligada
        {
                $arr_comodo_color[$i] = "#006400";
                $arr_comodo_txt[$i] = "ON";
                $arr_comodo_op[$i] = $arr_comodos_id[$i] + 3; // Vai somar mais 3 para enviar a opção de desligar. Ex.: ID[1] + 3 = 4
                $arr_comodo_st[$i] = '1';
        }
        else
        {
                $arr_comodo_color[$i] = "#8B0000";
```

```
                $arr_comodo_txt[$i] = "OFF";

                $arr_comodo_op[$i] = $arr_comodos_id[$i] + 2; // Vai somar mais 2 para enviar a opção de ligar. Ex.: ID[1] + 2 = 3

                $arr_comodo_st[$i] = '0';

        }

        sleep(3);

    }


?>



<div class="page_margins">

    <div class="page">

        <div id="header" role="banner">

            <h1>Interface de Gerenciamento de Automa&ccedil;&atilde;o de Ilumina&ccedil;&atilde;o.</h1>

            <span>Projeto de Automa&ccedil;&atilde;o Resid&ecirc;ncial</span>

        </div>

        <!-- begin: main navigation #nav -->

        <div id="nav"  role="navigation">

            <div class="hlist">

                <ul>

                    <li><strong>P&aacute;gina Principal</strong></li>

                    <li><a href="#" onclick="show_hide_div('col3');">Cadastro</a></li>

                    <li><a href="#">Sobre</a></li>

                    <li <?php echo $txt_eco."> <a href=\"".$PHP_SELF."?func=eco\">Modo Econ&ocirc;mico</a>"; ?> </li>

                </ul>

            </div>

        </div>
```

```
<!-- end: main navigation -->
<!-- begin: main content area #main -->
<div id="main">
<!-- begin: #col1 - first float column -->
<div id="col1" role="complementary">
<div id="col1_content" class="clearfix">
<h2>Status</h2>
<table width="300px" border="1px">
<?php
for($i=0; $i < $qtd_comodos; $i++)
{
echo "
<tr>
<td>".$arr_comodos[$i]."</td>
<td bgcolor=".$arr_comodo_color[$i]."><a href=\"".$PHP_SELF."?func=acao&op=".$arr_comodo_op[$i]."\" >".$arr_comodo_txt[$i]."</a></td>
</tr>
";
}
?>
</table>
</div>
</div>
<!-- end: #col1 -->
<!-- begin: #col2 second float column -->
<div id="col2" role="complementary">
<div id="col2_content" class="clearfix">
<h2>Sobre</h2>
```

```
                                        <p>Parte integrante do Projeto Final para obten&ccedil;&atilde;o de
grau no curso de Engenharia de Computa&ccedil;&atilde;o do Centro Universit&aacute;rio de Bras&iacute;lia -
UniCEUB.</p>

                                        <p>Professor orientador: Jos&eacute; Julim&aacute; Bezerra
J&uacute;nior</p>

                                        <p>Desenvolvido por <b>Gustavo Caetano de Almeida</b>.</p>

                                </div>

                        </div>

                        <!-- end: #col2 -->

                        <!-- begin: #col3 static column -->

                        <div id="col3" role="main">

                                <div id="col3_content" class="clearfix" style="visibility: hidden;">

                                        <h2>Cadastro</h2>

                                        <form id="form_cadastro" method="post"
action="my_layout.phtml?func=cadastra">

                                                <div id="campo_comodo" style="width:300px;
height:60px;" onClick="style.backgroundColor='#FFFFE0';">

                                                        <h3>Nome do Comodo</h3><input
id="nome_comodo" name="nome_comodo" style="background-color: #ADD8E6; border: 1px solid black; "
type="text" maxlength="255" value=""/>

                                                </div>

                                                <div id="campo_hora_on" style="width:300px;
height:60px;" onClick="style.backgroundColor='#FFFFE0';">

                                                        <h3>Hora Programada para Ligar</h3><input
id="hora_ligar" name="hora_ligar" style="background-color: #ADD8E6; border: 1px solid black; "  type="text"
maxlength="5" value="08:00"/>

                                                </div>

                                                <div id="campo_hora_off" style="width:300px;
height:60px;" onClick="style.backgroundColor='#FFFFE0';">

                                                        <h3>Hora Programada para Desligar</h3><input
id="hora_desligar" name="hora_desligar" style="background-color: #ADD8E6; border: 1px solid black; "
type="text" maxlength="5" value="20:00"/>

                                                </div>

                                                <input id="saveForm" style="background-color:
#ADD8E6; font-weight: bold; font-size: 12px;" type="submit" name="submit" value="Salvar" />
```

```
                                    </form>
                                </div>
                                <div id="ie_clearing"> </div>
                                <!-- End: IE Column Clearing -->
                            </div>
                            <!-- end: #col3 -->
                        </div>
                        <!-- end: #main -->
                        <!-- begin: #footer -->
                        <div id="footer" role="contentinfo">
                            Desenvolvido por <b>Gustavo Caetano de Almeida</b>.<br />
                        </div>
                        <!-- end: #footer -->
                    </div>
</div>
<!-- full skip link functionality in webkit browsers -->
<script src="./yaml/core/js/webkit-focusfix.js" type="text/javascript"></script>
</body>
</html>
```

Código do *script* SQL

```
-- Servidor: localhost
-- Versão do Servidor MySQL: 5.0.51
-- Versão do PHP: 5.2.6

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

--
-- Banco de Dados: `tcc_gustavo`
```

```
--
CREATE DATABASE `tcc_gustavo` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci;
USE `tcc_gustavo`;

-- --------------------------------------------------------

--
-- Estrutura da tabela `comodos`
--

CREATE TABLE IF NOT EXISTS `comodos` (
  `id_comodos` int(11) NOT NULL,
  `nome_comodo` varchar(50) default NULL,
  `hora_ligar` time default NULL,
  `hora_desligar` time default NULL,
  `is_timer` tinyint(1) default NULL,
  PRIMARY KEY  (`id_comodos`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;
```

ANEXO 01 – REFERÊNCIA DA LINGUAGEM PROCESSING

ELEMENTOS DE ESTRUTURA	**DESCRIÇÃO**
setup()	A função setup() é chamada quando um programa começa a rodar. Use esta função para inicializar as sua variáveis, os modos dos pinos, declarar o uso de bibliotecas, etc. Esta função será executada apenas uma vez após a placa Arduino ser ligada ou ressetada.
loop()	Após criar uma função setup() que declara os valores iniciais, a função loop() faz exatamente o que seu nome sugere, entra em *looping* (executa sempre o mesmo bloco de código), permitindo ao seu programa fazer mudanças e responder. Use esta função para controlar ativamente a placa Arduino.
ESTRUTURAS DE CONTROLE	**DESCRIÇÃO**
if (condicional)	É usado juntamente com um operador de comparação, verifica quando uma condição é satisfeita.
Operadores de comparação	x == y (x é igual a y) x < y (x é menor que y) x > y (x é maior que y)
if / else	Permite um controle maior sobre o fluxo de código do que a sentença *if* básica, tornando possível que múltiplos testes sejam agrupados.
switch / case	Do mesmo modo que as sentenças *if*, as *switch* / *case* controlam o fluxo dos programas. *Switch/case* permite ao programador construir uma lista de "casos" dentro de um bloco delimitado por chaves. O programa checa cada caso com a vaiável de teste e executa o código se encontrar um valor idêntico. *Switch / case* é um pouco mais flexível que uma estrutura *if/else* de modo que o programador pode determinar se a estrutura switch deve continuar checando por valores idênticos na lista dos "casos" após encontrar um valor idêntico. Se a sentença *break* não é encontrada após a execução do bloco de código selecionado por um dos "casos", então o programa vai continuar a checar por mais valores idênticos entre os "casos" restantes. Se uma sentença *break* é encontrada o código sai da estrutura do mesmo modo que em uma construção *if/else if.*
break	*break* é usado para sair de um bloco do, *for*, ou *while*, se sobrepondo à condição normal de verificação. Também é usado para sair de uma sentença *switch*.
#define	#define é um componente muito útil da linguagem C que permite ao programador dar um nome a uma constante antes que o programa seja compilado. Constantes definidas no Arduino não ocupam espaço em memória no *chip*. O compilador substitui referências a estas constantes pelo valor definido no momento da compilação.
CONSTANTES	**DESCRIÇÃO**

HIGH	O significado de *HIGH* (em referência a um pino) pode variar um pouco dependendo se este pino é uma entrada (*INPUT*) ou saída (*OUTPUT*). Quando um pino é configurado como *INPUT* com pinMode, e lido com um digitalRead, o microcontrolador considera como *HIGH* se a voltagem for de 3 volts ou mais. Um pino também pode ser configurado como um *INPUT* com o pinMode, e posteriormente receber um *HIGH* com um digitalWrite, isto vai "levantar" o resistor interno de 20KΩ que vai manter a leitura do pino como *HIGH* a não ser que ela seja alterada para *LOW* por um circuito externo. Quando um pino é configurado como *OUTPUT* com o pinMode, e marcado como *HIGH* com o digitalWrite, ele está a 5 volts. Neste estado ele pode enviar corrente como por exemplo acender um LED que está conectado com um resistor em série ao terra, ou a outro pino configurado como *OUTPUT* e marcado como *LOW*.
LOW	O significado de *LOW* também pode variar dependendo do pino ser marcado como *INPUT* ou *OUTPUT*. Quando um pino é configurado como um *INPUT* com o pinMode, e lido com o digitalRead, o microcontrolador considera como *LOW* se a voltagem for de 2 volts ou menos. Quando um pino é configurado como *OUTPUT* com pinMode, e marcado como *LOW* com o digitalWrite, ele está a 0 volts. Neste estado ele pode "drenar" corrente como por exemplo para acender um LED que está conectado com um resistor em série ao +5 volts, ou a outro pino configurado como *OUTPUT* e marcado como *HIGH*.
pinMode()	Mudar o comportamento elétrico do pino para *INPUT* ou *OUTPUT* (respectivamente alta e baixa impedância)
FUNÇÕES	**DESCRIÇÃO**
digitalWrite()	Escreve um valor *HIGH* ou um *LOW* em um pino digital. Se o pino foi configurado como uma saída (output) com o pinMode(), sua voltagem será determinada ao valor correspondente: 5V (ou 3.3V nas placas de 3.3V) para *HIGH*, 0V (terra) para LOW. Se o pino está configurado como uma entrada (*input*) escrever um *HIGH* levantará o resistor interno de 20KΩ (tutorial de pinos digitais). Escrever um *LOW* rebaixará o resistor.
analogRead()	Lê o valor de um pino analógico especificado. A placa Arduino contém um conversor analógico-digital de 10 bits com 6 canais (8 canais no Mini e no Nano). Com isto ele pode mapear voltagens de entrada entre 0 e 5 volts para valores inteiros entre 0 e 1023. Isto permite uma resolução entre leituras de 5 volts / 1024 unidades ou 0,0049 volts (4.9 mV) por unidade. São necessários aproximadamente 100 µs (0.0001 s) para ler uma entrada analógica, portanto a velocidade máxima de leitura é de

	aproximadamente 10.000 vezes por segundo.
ESTRUTURA DE TEMPO	**DESCRIÇÃO**
delay(ms)	Suspende a execução do programa pelo tempo (em milisegundos) especificado como parâmetro.
COMUNICAÇÃO SERIAL	**DESCRIÇÃO**
Serial.begin(int velocidade)	Ajusta o taxa de transferência em bits por segundo (*baud*) para transmissão de dados pelo padrão serial. Para comunicação com um computador use uma destas taxas: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 57600, 115200. Você pode, entretanto, especificar outras velocidades por exemplo para comunicação através dos pinos 0 e 1 com um componente que requer uma taxa específica.
int Serial.available()	Obtém o número de bytes (caracteres) disponíveis para leitura através da porta serial.
int Serial.read()	Lê dados que estejam entrando pela porta serial.
Serial.println(data)	Envia dados para a porta serial seguidos por um *carriage return* (ASCII 13, or '\r') e por um caractere de linha nova (ASCII 10, or '\n').

ANEXO 02 – REFERÊNCIA DA LINGUAGEM PHP E SQL

PHP	
ELEMENTOS DE FUNÇÃO	**DESCRIÇÃO**
fopen	Abre um arquivo ou uma URL
fread	Leitura *binary-safe* de arquivo
fclose	Fecha um ponteiro de arquivo aberto
fwrite	Gravação em arquivos *binary-safe*
fgets	Le uma linha de um ponteiro de arquivo
file_exists	Checa se um arquivo ou diretório existe
filesize	Lê o tamanho do arquivo
sleep	Atrasa a execução

SQL	
ELEMENTOS DE FUNÇÃO	**DESCRIÇÃO**
SET	muda um parâmetro de tempo de execução
CREATE	Criar (banco de dados/tabela)

ANEXO 03 – ESQUEMÁTICO DO ARDUÍNO DUEMILANOVE