UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

LUIZ CLAVER PEREIRA GRILO

DESENVOLVIMENTO DE UM SISTEMA DE AUTOMAÇÃO RESIDENCIAL BASEADO EM MICROCONTROLADOR

Campina Grande, Outubro de 2013

LUIZ CLAVER PEREIRA GRILO

DESENVOLVIMENTO DE UM SISTEMA DE AUTOMAÇÃO RESIDENCIAL BASEADO EM MICROCONTROLADOR

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Orientador: Prof. Dr. Péricles Rezende Barros

Campina Grande, Outubro de 2013

LUIZ CLAVER PEREIRA GRILO

DESENVOLVIMENTO DE UM SISTEMA DE AUTOMAÇÃO RESIDENCIAL BASEADO EM MICROCONTROLADOR

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Aprovado em / /

_____

Professor

Universidade Federal de Campina Grande

_____

Professor Dr. Péricles Rezende Barros

Universidade Federal de Campina Grande

AGRADECIMENTOS

Agradeço aos meus pais: Marcelo Bezerra Grilo e Tânia Maria Augusto Pereira, por terem me ensinado a viver! Com eles aprendi a gostar de estudar, viajar, apreciar e fazer música, respeitar as pessoas, não fazer coisas erradas e a fazer boas festas!

Aos meus irmão: Simone Augusto Pereira e Marcelo Grilo Jr, pela paciência. Em particular ao mais novo, por ter me aguentado nos diversos momentos em que estivemos juntos, na música no esporte e nos estudos.

Aos meus amigos, os quais não vou citar nomes, caso contrário posso me complicar depois. Com vocês aprendi o que são todos aqueles adjetivos bonitos que usamos nos poemas e nas letras de músicas. Como sei que nem todos vão ler esse relatório, então faço um especial agradecimento aos amigos que conheci na época da universidade. Alguns foram apenas colegas de disciplina ou de estudos. Mas fico feliz por ter conhecido pessoas que tenho a certeza e vontade de manter contato pelo resto da vida, todos vocês são parcela importante nessa conquista.

Finalmente agradeço a todos os que fizeram e fazem o curso de Engenharia Elétrica na UFCG. Aos professores, em especial o Prof. Péricles Rezende Barropor ter incentivado e apoiado a ideia de fazer um trabalho envolvendo automação residencial e arduino, ao secretariado em especial a Adail e Tchay, que tornam a nossa vida mais fácil e doce no tratar dos problemas.

Foi bom, foi divertido e que venha muito mais!

Muito obrigado a todos e até a próxima!

RESUMO

A automação residencial vem alavancando índices de 35% de crescimento no mercado nacional. Isso se deve ao barateamento dos produtos e à popularização de novas tecnologias que facilitam a implementação do projeto. A automação residencial pode ser subdivida em três segmentos: automação da segurança, controle multimídia e controle e automação de acionamentos de iluminação e outros circuitos. Um projeto completo contempla as três variações, mas uma vantagem desse tipo de sistema é a sua modularidade, possibilitando ao cliente a instalação sucessiva do sistema. Esse trabalho tem como objetivo desenvolver um sistema de automação residencial, contemplando automação da segurança, acionando alarmes através da leitura de sensores; e o controle de acionamentos de circuitos de iluminação e irrigação, tudo isso utilizando o microcontrolador Arduino como unidade de processamento central. Os principais objetivos são desenvolver um sistema que seja menos invasivo no processo de instalação na residência, que possa ser controlado localmente através do *smartphone* com sistema operacional *Android*, utilizando comunicação *bluetooth*, e que também possa ser controlado de forma remota através da internet e, finalmente, que seja um sistema de baixo custo. Um protótipo da instalação elétrica monofásica de uma casa foi desenvolvido para validar a arquitetura do sistema de automação proposto e servir como fonte de estudos para desenvolvimento e aperfeiçoamento em futuros projetos de automação residencial.

Palavras chave: Automação residencial; controle; *bluetooth; android;* instalações elétricas.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

Ao terminar o expediente no fim do dia, um indivíduo, antes de sair do trabalho, acessa a internet do seu *smartphone* e verifica através de câmeras a situação de sua casa. Aciona iluminação externa e interna e pode escolher alguns cenários de recepção, que incluem som ambiente ou temperatura padronizada. Ao chegar à casa, o portão se abre automaticamente ao reconhecer o veículo e registra a sua entrada. Já dentro da casa o indivíduo pode acionar o sistema multimídia e assistir a programação gravada da TV ou verificar as tarefas realizadas automaticamente ao longo do dia, como por exemplo, a irrigação do jardim, devido à detecção de um baixo nível de umidade no solo.

O cenário descrito anteriormente não é fictício e hoje em dia é possibilitado através de sistemas de automação residencial. Segundo dados da Associação Brasileira de Automação Residencial (Aureside), o crescimento do mercado de automação residencial no Brasil é de 35% ao ano. O crescimento se deve a redução nos custos das tecnologias e ao aumento do número de empresas e serviços especializados na área.

Os serviços de automação residencial podem ser divididos em três segmentos: automação da segurança, controle multimídia e automação e controle de acionamentos de iluminação e outros circuitos. Um projeto completo contempla as três variações, mas uma vantagem desse tipo de sistema é a sua modularidade, possibilitando ao cliente a instalação sucessiva do sistema. Por exemplo, é possível fazer um projeto de automação para controle de iluminação e acionamento de eletrodomésticos; em seguida, pode-se instalar um módulo de segurança contendo câmeras e controles de acesso; por fim, é possível instalar um sistema multimídia para controle da televisão e som ambiente.

O desenvolvimento tecnológico é outro fator que incentiva o uso de automação residencial. O avanço e barateamento dos microprocessadores e, principalmente, o desenvolvimento de tecnologias de comunicações, como a internet e os telefones móveis, possibilitam portabilidade e um controle descentralizado dos sistemas. Além disso, redes de comunicação sem fio possibilitam projetos menos invasivos a estrutura de construção da casa.

Esse projeto apresenta como proposta a implementação de um sistema de automação residencial de baixo custo, utilizando um microcontrolador Arduino como fonte central de processamento. O controle do sistema é feito através de um *smartphone* com sistema operacional *Android*, utilizando comunicação *bluetooth* e também através da internet, utilizando o protocolo TCP/IP. O controle convencional, através de

interruptores, para acionar circuitos de iluminação, por exemplo, continua coexistindo com os novos controles de automação, podendo ser necessária apenas a substituição de interruptores simples por interruptores paralelos, nos casos onde esses interruptores não estavam previstos.

O projeto visa contemplar a automação da segurança, verificando vários parâmetros de interesse através de sensores: detecção de chamas, vazamento de gás, e sensor de presença; e o controle e automação de acionamentos, possibilitando ao usuário o controle de iluminação, irrigação e acionamentos de cargas diversas, como por exemplo, intensidade luminosa de determinado ambiente.

Para validação e testes do sistema, foi desenvolvido um protótipo de uma instalação elétrica monofásica de uma residência. O protótipo contém três lâmpadas, simulando o circuito de iluminação; uma lâmpada dimerizavel, simulando o controle de intensidade luminosa; uma válvula elétrica, para simular um sistema de irrigação. Quatro interruptores simulam os acionamentos convencionais dos circuitos de iluminação e irrigação, uma tomada ainda é disponibilizada, para fornecer a alimentação necessária ao funcionamento do sistema. Circuitos de acionamento a relé e a chaveamento com TRIAC foram utilizados para auxiliar no processo de acionamento automático das cargas. Sensores, LEDS e buzinas são utilizados para interagir com o meio e fornecer sinalizações para o usuário.

2. ARQUITETURA DO SISTEMA

O sistema de automação residencial proposto visa automatizar o controle de tarefas cotidianas e possibilitar ao usuário que controle acionamento de cargas ou acompanhe o status do sistema através do *smartphone* ou da internet. Também é importante que o sistema seja pouco invasivo na instalação em uma residência e que possua componentes de baixo custo. A Figura 1 é um diagrama geral da arquitetura proposta para desenvolvimento do sistema.

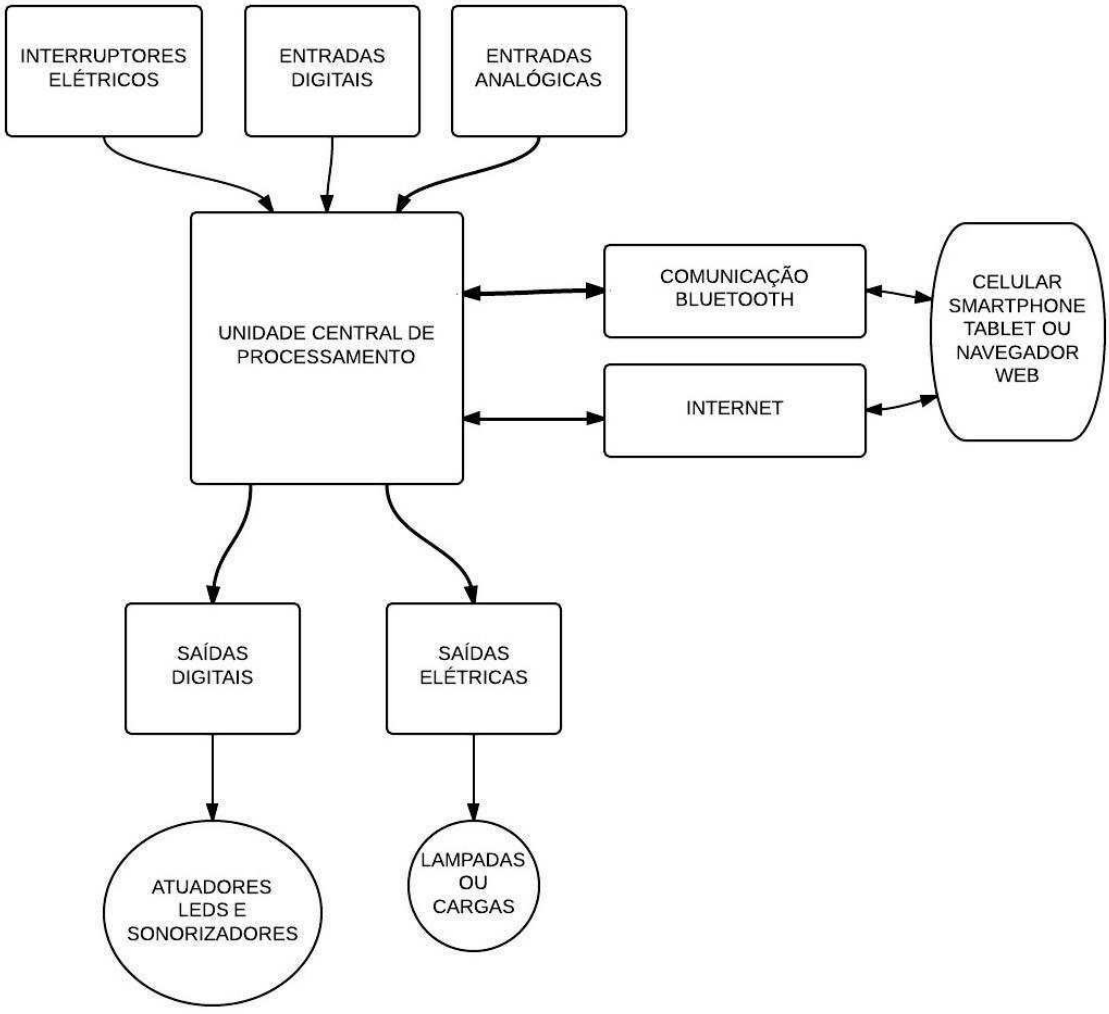

Figura 1 - Diagrama da arquitetura proposta para o sistema de automação residencial.

O sistema possui uma unidade central de processamento, onde os sinais de entrada de sensores e interruptores ou através de protocolo de comunicação são processados para gerar as respectivas saídas desejadas. Os sinais de saída são enviados para módulos de

acionamento de carga ou interfaces com o usuário, como LEDs e sonorizadores. A interface entre o *smartphone* e o sistema se dá através de um módulo de comunicação *bluetooth*. O sistema ainda possui hardware dedicado para se conectar à internet, de forma que seja possível receber e enviar comandos de um usuário remotamente.

O sistema se conecta aos interruptores da rede elétrica residencial existente, devendo a única adaptação a ser feita, caso necessário, seja a substituição dos interruptores simples por interruptores paralelos. A única instalação elétrica necessária para funcionamento do sistema é o fio de retorno dos interruptores, que deve ser conectado em paralelo com a placa de relés utilizada. A Figura 2 apresenta um detalhe da conexão elétrica entre os dois componentes, de forma que seja possível acionar as cargas através dos interruptores ou do *smartphone* de forma independente.

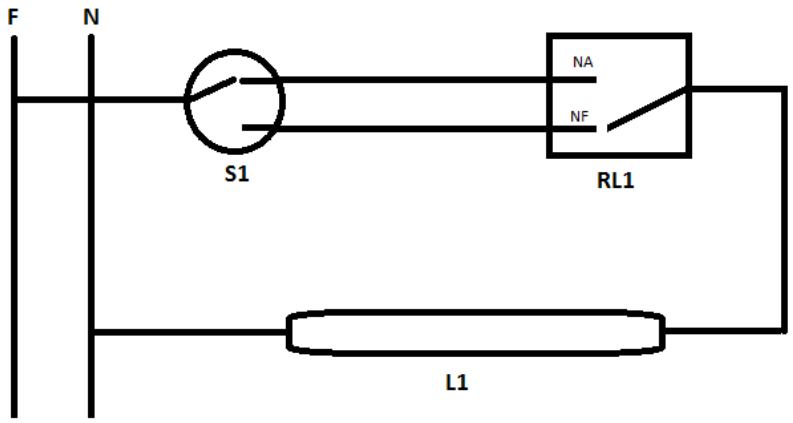

Figura 2 - Instalação elétrica necessária para acionamento das cargas através dos interruptores ou de comandos bluetooth via celular.

3. DESCRIÇÃO DETALHADA DO SISTEMA

Nesse capítulo serão apresentados a descrição de cada parte integrante do sistema de automação residencial desenvolvido e previamente apresentado na arquitetura geral. Também serão apresentados conceitos teóricos referentes a protocolos de comunicação e dispositivos utilizados no desenvolvimento do projeto.

3.1 Unidade central de processamento

No sistema de automação residencial desenvolvido, a unidade central de processamento é responsável controle dos sinais de entrada e saída digitais e analógicos, pelo processamento dos protocolos serial e SPI, pela execução das funções desenvolvidas para controle do relógio e das lógicas de acionamento de relés e sensores, e também pela geração de um servidor que processa as requisições HTTP recebidas via internet. O *hardware* escolhido para ser unidade central de processamento foi o sistema microcontrolador Arduino.

A plataforma Arduino foi desenvolvida no Instituto de Design de Interação Ivrea, na Itália, em meados de 2005, pelo professor Massimo Banzi. O objetivo de seu desenvolvimento era o de construir uma plataforma de *hardware* e *software* capaz de controlar e mensurar o mundo físico através do processamento de entradas e saídas, que abstraísse ao máximo a parte eletrônica, permitindo que o foco fosse mantido no desenvolvimento da ideia base do projeto. Denominou-se tal plataforma de Arduino (Figura 3), nome de origem germânica *Hardwin*, *hard* (forte) e *win* (amigo), e que em italiano tornou-se Arduino.

A plataforma possui a documentação aberta, disponibilizando todos os esquemáticos de *hardware* e bibliotecas para as mais diversas aplicações e uma IDE de programação em C/C++. Utiliza um microcontrolador de oito bits como unidade de processamento. Apresenta um design específico de forma que outros módulos (*Shields)* possam ser facilmente adicionados para agregar funcionalidades. As suas principais características são:

Figura 3 - Placa de desenvolvimento Arduino, com microcontrolador ATMEL de 8 bits.

- Microcontrolador ATmega328;
- 14 E/S digitais (das quais 6 são PWM);
- 6 entradas analógicas – 10 bits de resolução;
- Comunicação Serial, SPI e I2C on board;
- 16 MHz de clock;
- Reset por software;

3.1.1 O Microcontrolador ATMega328

O microcontrolador ATmega328 é a unidade central de processamento da placa de desenvolvimento Arduino. O seu funcionamento pode ser resumido através do diagrama funcional (Figura 4), que apresenta as estruturas de hardware presentes no dispositivo.

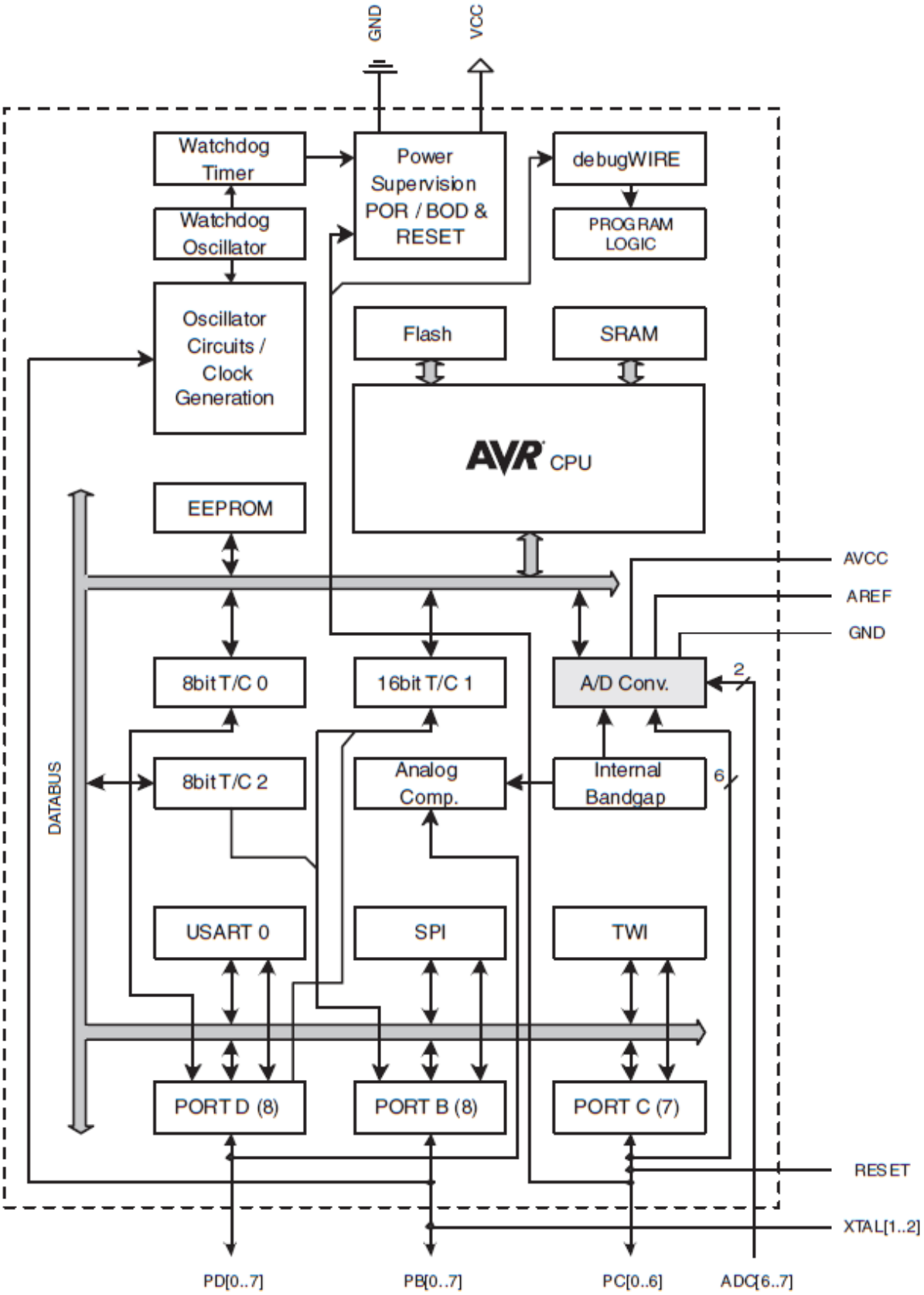

Figura 4 – Diagrama de blocos do microcontrolador Atmega328, unidade de processamento do Arduino Uno R3.

O núcleo AVR tem trinta e dois registradores de uso geral, estando todos conectados à Unidade Lógica Aritmética. O AVR permite que dois registradores independentes sejam acessados em uma única instrução, executada em um único ciclo de *clock*. Essa característica operacional resulta em uma arquitetura mais eficiente, além de permitir obter taxas de transferência de dados 10 vezes mais rápidas do que os microcontroladores CISC[1] convencionais.

Além dos registradores descritos posteriormente, há ainda três temporizadores / contadores com modos de comparação, interrupções interna e externa, serial programável USART, uma porta SPI, um *Watchdog Timer*[2] com oscilador interno, e cinco modos selecionáveis de economia de energia.

No modo de "*wait-state*" da CPU, os módulos: SRAM,Timer/Counters, USART, porta SPI e a interrupção continuam funcionando. No modo de economia de energia, o temporizador assíncrono continua funcionando, permitindo ao usuário manter um temporizador de base enquanto o resto do dispositivo encontra-se hibernando.

O modo ADC de redução de ruído interrompe a CPU e todos os módulos E/S, exceto, o temporizador assíncrono ADC, para minimizar o ruído de comutação durante as conversões ADC.

No modo de "*wait-state*", o oscilador está em funcionamento enquanto o resto do dispositivo hiberna, o que permite um *start-up* rápido e com baixo consumo de energia.

A AVR ATmega328 é suportada por um pacote completo de programas e ferramentas de desenvolvimento de sistemas, incluindo compiladores C, Macro Assemblers e depurador/simulador de programa. A plataforma Arduino consiste em um arcabouço de *hardware* e *software* auxiliares ao AVR para facilitar o desenvolvimento de soluções que utilizem este microchip da fabricante ATMEL.

A grande diferença entre os vários modelos de Arduino disponíveis no mercado reside na memória do microcontrolador, sendo essa uma característica determinante em seu desempenho. No microcontrolador ATMega328 há três tipos de memória: Flash, SRAM, EEPROM. A memória Flash é utilizada para o armazenamento do programa que

[1] CISC(*Complex Instruction Set Computer*) é uma linha de arquitetura de processadores capaz de executar centenas de instruções complexas diferentes. Sendo assim, extremamente versátil. Os processadores baseados em um conjunto de instruções complexas contêm microprogramação, ou seja, um conjunto de códigos de instruções, que são gravadas no processador, permitindo-lhe receber as instruções dos programas e executá-las, utilizando as instruções contidas na sua microprogramação.

[2] *Wacthdog Timer* é um dispositivo eletrônico temporizador que dispara um reset ao sistema, se o programa principal devido a alguma condição de erro deixar de fazer reset no *Watchdog timer*.

é carregado no microcontrolador para ser executado, bem como para o armazenamento do *bootloader*. O *bootloader* é responsável pelo gerenciamento da inicialização do microchip, e desempenha algumas tarefas programadas, como determinar quando reprogramar ou passar para a aplicação principal.

A memória SRAM (Static Random Access Memory) é a memória utilizada para armazenamento das variáveis de processamento. Quando o Arduino é desenergizado, as informações armazenadas nesta memória são perdidas. Diferentemente de EEPROM (Electrically Erasable Programmable Read-Only Memory), que é responsável pelo armazenamento de constantes ou dados de configuração, e apresenta longos ciclos de gravação e rápida leitura de dados.

O modelo Arduino Uno Versão 3 tem 32 kB de memória Flash, dos quais 0,5 kB são destinados ao bootloader, 2 kB de SRAM e uma EEPROM de 1 kB.

O ATMega328, assim como outros microcontroladores, oferece ainda um conversor A/D. Na conversão de um sinal analógico em digital, fazem-se amostragens, e representa-se a leitura realizada através de um valor equivalente binário, de acordo com a resolução do conversor A/D. A precisão do conversor é determinada pelo número de bits que o mesmo gera para representar as grandezas analógicas. O conversor A/D do ATmega é de 10 bits, representando 2^{10} níveis de tensão distintos. Assim, a resolução do conversor pode ser expressa por:

$$resolução = \frac{V_{ref}}{2^n}$$

em que V_{ref} é a tensão de referência [V]; e n é a número de bits.

Desse modo, sendo a opção default da tensão de referência V_{ref} = 5 V, para o Arduino, têm-se uma resolução de:

$$resolução = \frac{5}{2^{10}} = 4{,}88\ mV$$

Diz-se, portanto, que o Arduino pode detectar variações de 5 mV. Há, porém, a possibilidade de alterar-se o V_{ref}, através do pino de entrada A_{ref}. Desse modo, com a possibilidade de diminuição da resolução de referência, têm-se também uma diminuição

dos valores de tensão detectáveis e uma consequente diminuição do nível máximo de medição de tensão permitido, o que pode não ser desejável em determinadas aplicações.

O ATmega328 é um microcontrolador com elevada capacidade de processamento e um conjunto de periféricos diversificado, conferindo, dessa forma, uma solução altamente eficaz para muitas aplicações de controle embarcado. Quando utilizado na placa de desenvolvimento Arduino, se torna uma solução perfeita para ser a unidade central de processamento do nosso sistema de automação, visto que é portátil, apresenta conjunto de entradas e saídas necessárias e possui documentação aberta disponível para ser acessada e utilizada sem custos adicionais.

3.2 Descrição do *Software* do Arduino

A plataforma de *software* do Arduino, modelada na linguagem de programação *Processing*[3], denominada IDE (*Integrated Devolopment Environment*), trata-se de um programa executado em um computador, que permite a codificação de programas em linguagem específica, *sketches*, a serem carregados na plataforma de *hardware*.

Uma vez requerido o *upload* do *sketch* na placa Arduino, o código escrito em linguagem específica é traduzido na Linguagem C. Na sequência, o mesmo é compilado pelo AVR- GCC, responsável pela tradução final para a linguagem entendida e conhecida pelo microcontrolador, simplificando, dessa forma, a programação de microcontroladores.

Desse modo, o ciclo de programação de um Arduino pode ser resumido como:

1. Escreve-se o Sketch;
2. Compila-se o Sketch;
3. Liga-se a placa a uma porta USB do computador;
4. Envia-se o Sketch para a placa, através da USB;
5. A placa executa o programa;

O IDE pode ser executado em qualquer sistema operacional. É dividido em três partes: o Toolbar no topo, Sketch Windows no centro e a janela de mensagens na base (Figura 05).

[3] Processing é uma linguagem de programação de código aberto e ambiente de desenvolvimento integrado desenvolvido em 2001 pelo MIT (Massachusetts Institut of Tecnology), construído para comunidades de projetos visuais com o objetivo de ensinar noções básicas de programação de computador em um contexto visual.

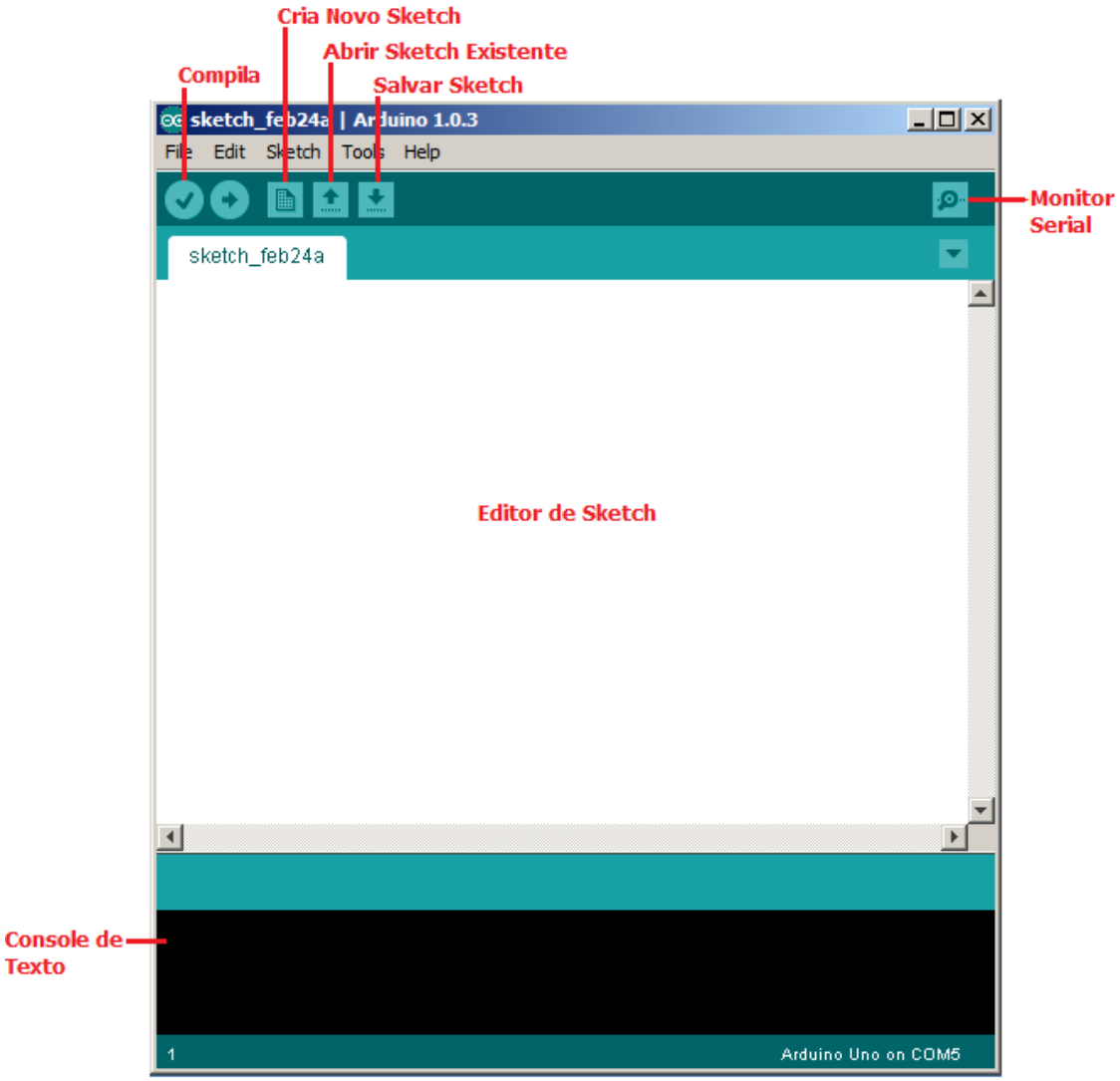

Figura 5 – Tela do software utilizado para escrever os sketchs e programar o Arduino.

3.3 Módulo de expansão e conectividade com a internet

Como explicado, uma das vantagens do Arduino é a possibilidade de expansão das suas funcionalidades através de módulos adicionais, também conhecidos como *Shields*. Os *Shields* são placas de circuitos impresso que são encaixadas sobre a placa do Arduino e agregam recursos de *hardware* ao microcontrolador, portanto, apresenta como principal característica ter o mesmo design da placa do Arduino UNO. Exemplos de *Shields* desenvolvidos para o Arduino são (Figura 6):

- Arduino GSM Shield – agrega conexão através de rede GPRS.
- Arduino Bluetooth Shield – Agrega conectividade *bluetooth* ao microcontrolador.
- Arduino WiFi Shield – agrega conectividade à internet sem fio.
- Arduino Motor Shield – circuito para controle de cargas indutivas, como motores.

- Arduino Sensor Shield – placa desenvolvida para ler sinais de diversos sensores analógicos.
- Arduino Ethernet Shield – Conecta o microcontrolador a internet através de rede ethernet.

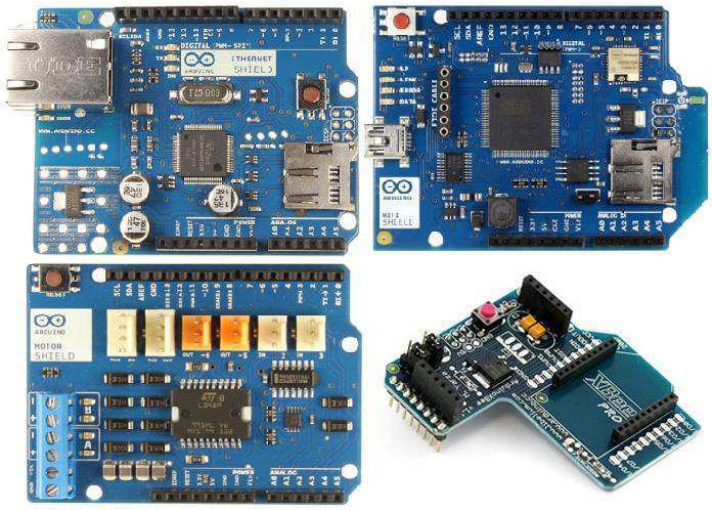

Figura 6 – Diversos *shields* para ser utilizado como módulo de expansão das funcionalidades de *hardware* do Arduino.

Para o projeto do sistema de automação residencial, foi escolhido o S*hield Ethernet* com o intuito de promover a conexão do sistema com a internet, utilizando protocolo de comunicação TCP/IP. A escolha do S*hield* Ethernet se justifica pois a maioria dos projetos de automação residencial provê acesso à internet, de forma que seja possível o usuário verificar o *status* do sistema ou enviar comandos de forma remota. Ele possui menor custo que os demais, apresenta conexão cabeada ethernet com conector padrão RJ45, o que implica em maior robustez ao sistema, e disponibilidade em estoque para aquisição imediata. As principais características desse *shield* são listadas a seguir:

- Controlador de ethernet: Wiznet W5100.
- Velocidade da conexão: 10/100 Mb.
- Conector padrão RJ45.
- Conexão com o Arduino via protocolo SPI.
- Conector micro-SD, para criar um servidor de arquivos através de cartão de memória.

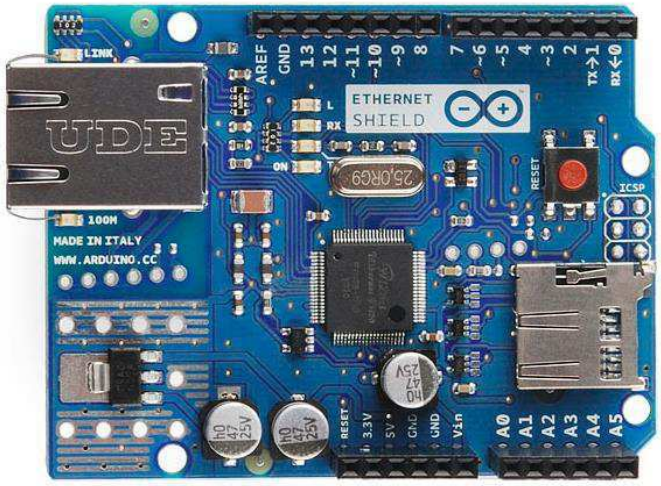

Figura 7 – Shield Ethernet, acoplado ao Arduino agrega a função de acesso à internet via TCP/IP.

Como explicado, o *shield ethernet* se comunica com a placa do Arduino através de protocolo de comunicação SPI (*Serial Peripheral Interface*). O SPI é um protocolo de dados serial síncrono, utilizado por microcontroladores para comunicação rápida com um ou mais dispositivos periféricos em curtas distâncias. Também pode ser usado para comunicação entre dois microcontroladores.

Em uma comunicação SPI, sempre há um mestre e um ou mais escravos, que se comunicam entre si através de uma conexão *full-duplex*. Os seguintes fios constituem a conexão SPI:

- MOSI (Master Out Slave In): Canal responsável pelo envio de dados do mestre para o escravo;
- MISO (Master In Slave Out): Canal responsável pelo envio de dados do escravo para o mestre;
- SCK (Serial Clock): Canal para envio do sinal de clock;
- SS (Slave Select): Canal utilizado pelo mestre para habilitar ou desabilitar a conexão com o escravo. Se na conexão existir mais de um escravo, todos compartilham os canais citados anteriormente, com exceção deste. Cada escravo possui seu SS. E quando este pino estiver em nível baixo, há conexão com o mestre e quando estiver em nível alto, o escravo ignora as informações do barramento.

Na comunicação SPI são utilizados dois registradores de deslocamento para a troca de informações, um pertencente ao mestre e o outro ao escravo. Os bits são deslocados e enviados um por vez, entre os registradores. À medida que um bit é

transmitido pelo canal MOSI, outro é recebido pelo canal MISO, e após todos serem enviados, os registradores do mestre e do escravo estarão com os dados trocados. A Figura 8 ilustra a troca de informações neste tipo de comunicação.

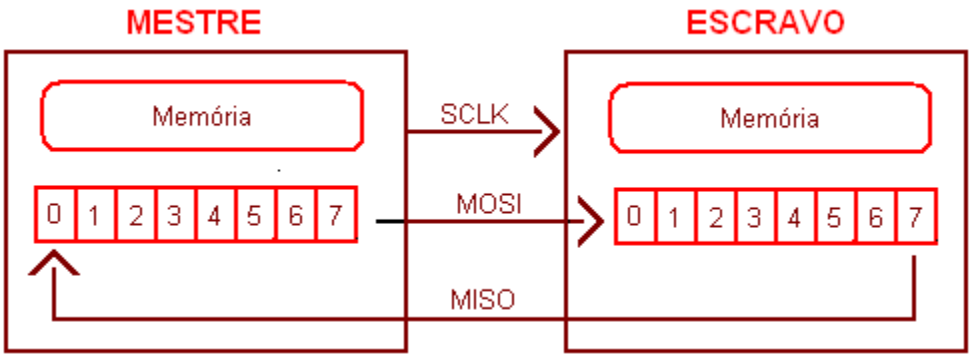

Figura 8 – Troca de informações entre mestre e escravo através de protocolo de comunicação SPI.

Para utilizar o *shield ethernet* com o Arduino Uno R3, é necessário adicionar duas bibliotecas de *software* disponibilizadas gratuitamente no site do Arduino, e que são instaladas junto com a IDE de programação dos *sketchs*. As bibliotecas *Ethernet.h* e *SPI.h*. tem ligação direta com as conexões apresentadas na Figura 9 e possuem funções predefinidas para definir IP, escrever pelo canal ethernet, definir um cliente, definir um servidor, ler e escrever comandos SPI, definir parâmetros de configurações do protocolo.

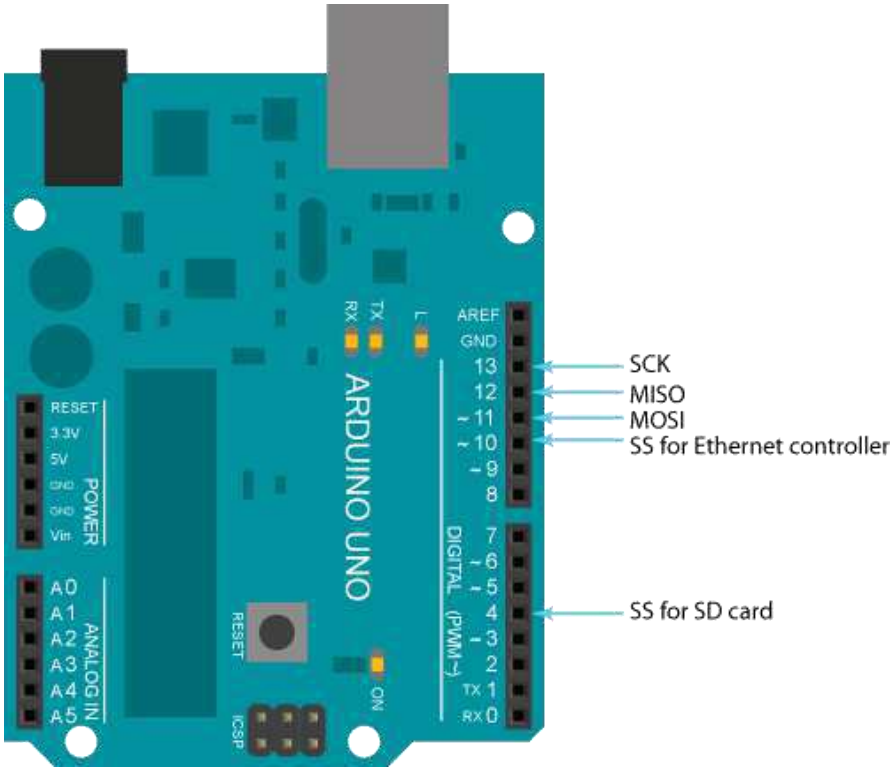

Figura 9 – Pinos utilizados pela comunicação SPI do shield de expansão ethernet.

3.4 Conexão com a Internet

Como descrito, uma das funcionalidades do projeto é promover conexão com a internet, de forma que seja possível o usuário verificar o status do sistema ou atuar de forma remota, para tomar algumas ações de controle. O meio que o Arduino se conecta a internet é através do *shield* ethernet apresentado anteriormente, contudo algumas adaptações são necessárias para correto funcionamento da conectividade com a internet.

No Arduino, com auxílio das funções da biblioteca *ethernet.h*, disponível na IDE instalada no PC, é possível criar um pequeno servidor HTTP com uma página contendo as informações que deseja-se apresentar ao usuário. Contudo, para correta configuração do servidor, é necessário que a rede possua IP fixo, e que o roteador da residência onde o sistema for instalado, seja configurado para liberar o acesso de forma remota ao servidor criado no Arduino. Também é no roteador local que espera-se que sejam implementadas verificações de login para segurança do acesso, visto que as limitações de memória do Arduino não permitem que sejam programados muitos recursos.

A Figura 10 apresenta uma arquitetura da conexão com a internet do sistema de automação residencial proposto.

Figura 10 – Topologia de acesso ao servidor do Arduino de forma remota via internet.

3.5 Controle sem fio via comunicação *bluetooth*

Outro recurso presente no projeto é a possibilidade de controlar os circuitos de acionamentos, através de um telefone celular, do tipo *smartphone,* utilizando comunicação sem fio. Esse recurso está presente em praticamente todos os sistemas de automação residencial encontrados no mercado, portanto é uma funcionalidade essencial a ser implementada. Para o sistema de automação residencial desenvolvido foi escolhido um *smartphone* com sistema operacional Android com suporte à comunicação *bluetooth.*

Por ser uma tecnologia que permite uma comunicação sem fios entre dispositivos, o *bluetooth* utiliza a faixa de frequência de 2.4GHz. Seu padrão é definido por um grupo de empresas (SIG), as quais possuem interesse no dispositivo, que também supervisiona o desenvolvimento das especificações, bem como promove e protege a marca.

O *bluetooth* tem por objetivo prover um meio de baixo custo, baixo consumo de energia e baixa complexidade de configuração para interligar dispositivos eletrônicos diversos, tais como telefones celulares, notebooks, desktops, câmeras digitais, impressoras e periféricos em geral. Por esta capacidade de interligação com dispositivos diversos, o *Bluetooth* foi selecionado como meio de comunicação para o envio dos dados deste projeto. Um segundo tipo de comunicação sem fio, diversas vezes utilizado em projetos de automação residencial é a *ZigBee.* Não foi utilizado *ZigBee* pois não é possível comunicação com *smartphones* ou computadores de forma simplificada e direta, como no caso do *bluetooth.*

Os dispositivos que possuem a tecnologia Bluetooth são divididos em três classes, de acordo com a potência do sinal que podem produzir. As classes são especificadas na tabela abaixo.

Tabela 1- Classes de dispositivos Bluetooth

Classe	Pot Max (mW)	Pot Max (Dbm)	Cobertura (m)
1	100	20	100
2	2.5	4	10
3	1	0	1

Dependendo do tipo, os dispositivos *bluetooth* podem fornecer determinados serviços para as aplicações, tais como envio e recebimento de arquivos, conexão a uma rede remota através do celular, entre outros. O serviço de interesse deste trabalho é o SPP (*Serial Port Profile*), que permite que um dispositivo estabeleça conexão com outro a partir da criação de uma porta serial virtual, possibilitando que ambos se comuniquem através de uma conexão serial sem fio. Tal serviço utiliza um protocolo de emular portas seriais denominado de RFCOMM.

A comunicação serial corresponde à transmissão e recepção de um bit de dados por vez. Existem dois tipos de comunicação serial, a síncrona e a assíncrona. Na comunicação síncrona, deve haver o envio do sinal de *clock* entre os dispositivos, já na comunicação serial assíncrona, não há necessidade de envio do sinal de *clock*, portanto as informações só necessitam dos canais de dados para serem trafegados. Na comunicação serial assíncrona, utilizada no serviço SPP dos dispositivos *bluetooth*, os dados são enviados em pacotes de bits, resultando em 8 bits de informação (um *byte*) mais os bits de controle. Os bits de controle são os seguintes:

- Start Bit: atua como indicador de início da transmissão, ativando um temporizador interno do receptor para gerar o mesmo clock que o transmissor;
- Stop Bit: sinaliza a parada, o término do envio do pacote de dado. Pode ser 1 ou 2 bits;
- Bit de Paridade: responsável pela conferência da paridade na informação. Este bit é opcional e verifica a ocorrência de erros no envio da informação.

Para haver uma comunicação consistente, a comunicação serial assíncrona necessita de configurações prévias. Ambos os dispositivos precisam preestabelecer igualmente a taxa de envio (*Baud Rate*) dos bits, ou seja, quantos bits por segundos serão enviados na conexão, se há o bit de paridade, quantos *Stop Bits* serão utilizados e se há controle de fluxo da informação. A Figura 11 ilustra um típico pacote de dados enviado em uma conexão serial.

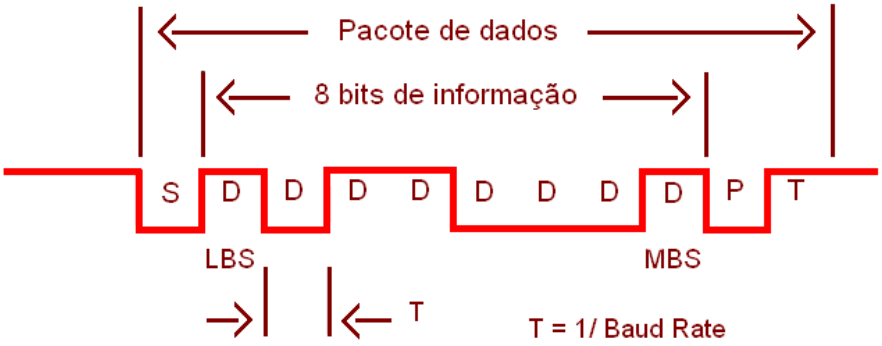

Figura 11 – Pacote de dados em uma comunicação serial assíncrona.

O *hardware* escolhido para implementar a comunicação *bluetooth* no Arduino foi o conversor b*luetooth* para serial desenvolvido pela empresa *Sure Electronics*, que apresenta as seguinte características:

- Frequência de operação: 2,4 GHz a 2,48 GHz;
- Protocolo b*luetooth* V2.0+EDR;
- Saída de potência classe 2;
- *Hardware* de suporte implementado;
- Tensão de operação: sinais TTL (5 V).

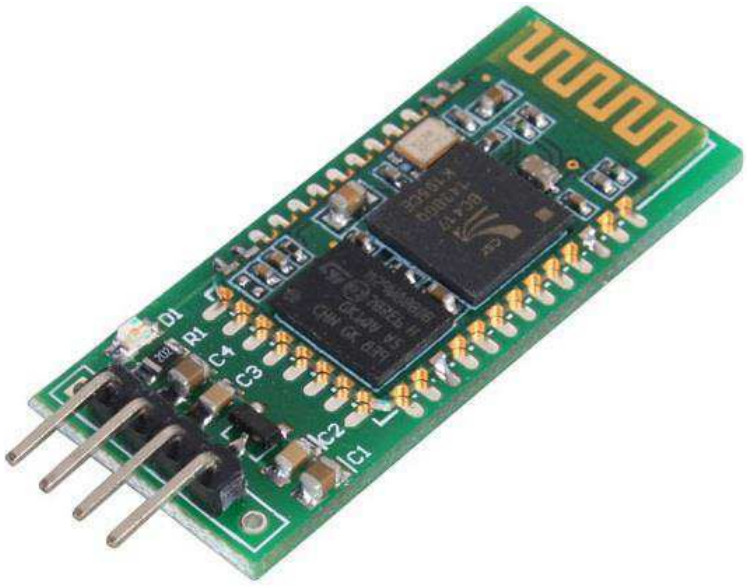

Figura 12 – Hardware *bluetooth* utilizado para comunicação entre Arduino e *smatphone*.

Esse dispositivo possui apenas dois sinais de saída, um pino para transmissão de dados serial, e outro para recepção de dados serial. Além disso, é necessário fornecer alimentação de 5 Vdc e um terra. Ele vem preconfigurado para operar a taxa de transmissão e recepção de 9600 bps, portanto configurações adicionais não são necessárias.

Para enviar e receber sinais no Arduino, basta que seja conectado o sinal TX do *chip bluetooth* a porta RX do Arduino (pino 0) e o sinal RX do *chip bluetooth* a porta TX do Arduino (pino 1), conforme Figura 12. Não é necessário incluir nenhuma biblioteca extra no *sketch* a ser compilado, bastando apenas configurar a porta serial do Arduino para operar em 9600 bps e utilizar as funções de envio e recepção via porta serial, de acordo com a lógica do programa.

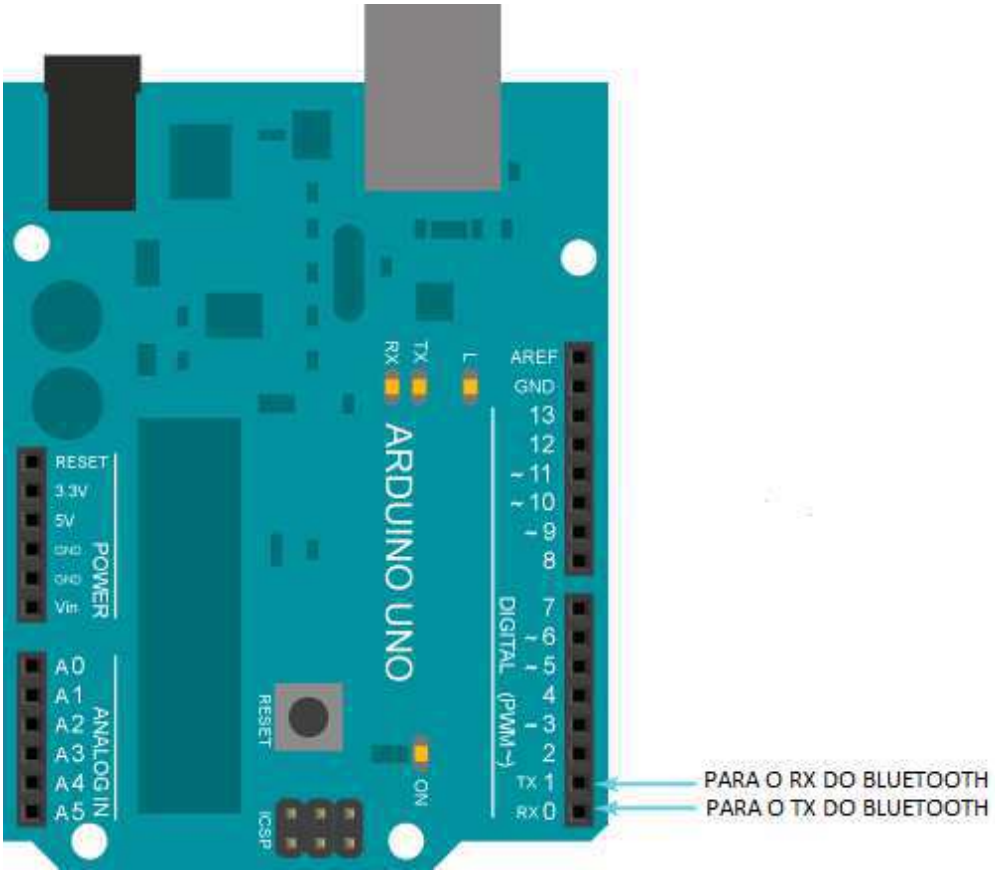

Figura 13 – Conexão entre o módulo *bluetooth* e o Arduino.

3.6 Smartphone Android e AppInventor

No segundo trimestre de 2013 foram vendidos 8 milhões de smartphones no Brasil, número 110% superior em relação ao mesmo período do ano anterior. Os dados foram divulgados pela consultoria IDC. De acordo com o estudo, pela primeira vez no país, a venda de smartphones superou a de aparelhos convencionais, dominando 54% do mercado local. Do total de smartphones comercializados, 90% estão equipados com o sistema operacional da Google, o Android.

Essa é uma das principais justificativas para a escolha do Android como sistema operacional adotado para desenvolvimento do aplicativo de controle do sistema. Outra importante informação é que o Android é um sistema aberto, programável em JAVA e que possui vasta documentação para elaboração de aplicativos, o que elimina custos associados com aquisição de *softwares* ou *hardware* específicos para programação dos aparelhos.

A ferramenta utilizada para desenvolver o aplicativo foi o *AppInventor*, um *applet* JAVA que roda direto do *browser,* desenvolvido pelo MIT (*Massachusetts Institute of Technology*). No *applet* é possível construir a interface gráfica do aplicativo, inserindo

botões, textos, figuras ou elementos de *hardware* (conectividade a internet, conexão *bluetooth*, etc) e, em seguida, constrói-se a lógica por trás de cada elemento, em uma programação fácil e intuitiva através do uso de blocos de funções. As Figura 14 e Figura 15 apresentam capturas de tela do módulo de construção de interface do AppInventor e do módulo de programação, respectivamente.

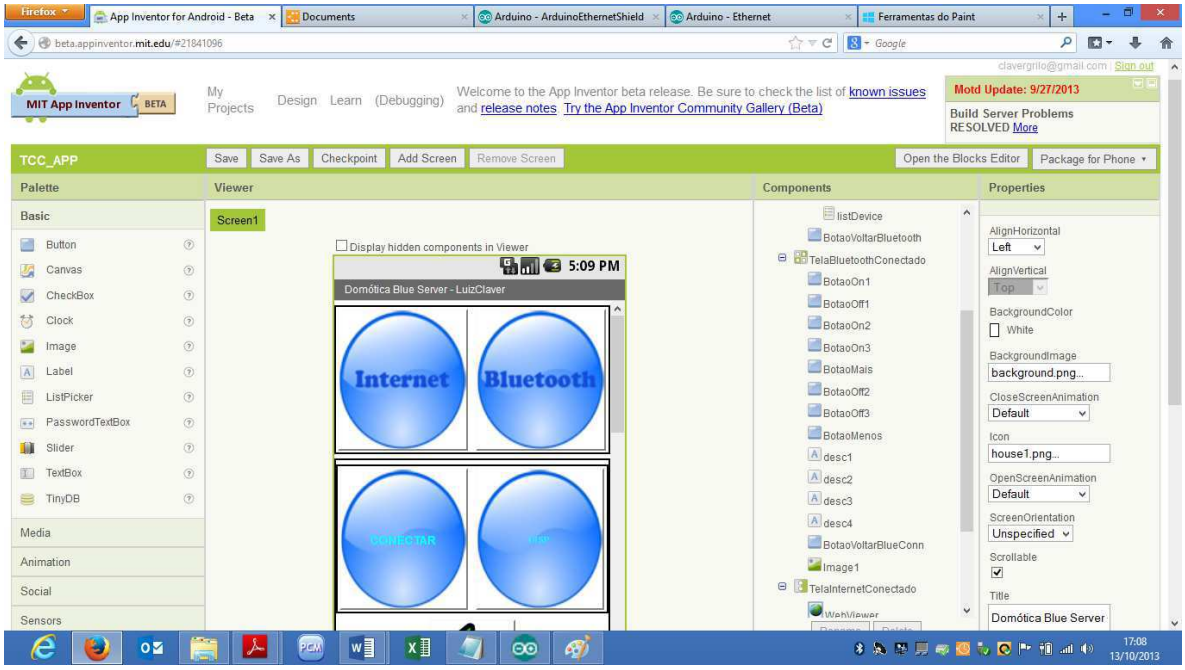

Figura 14 – Módulo de construção da interface gráfica do aplicativo no *AppInventor*.

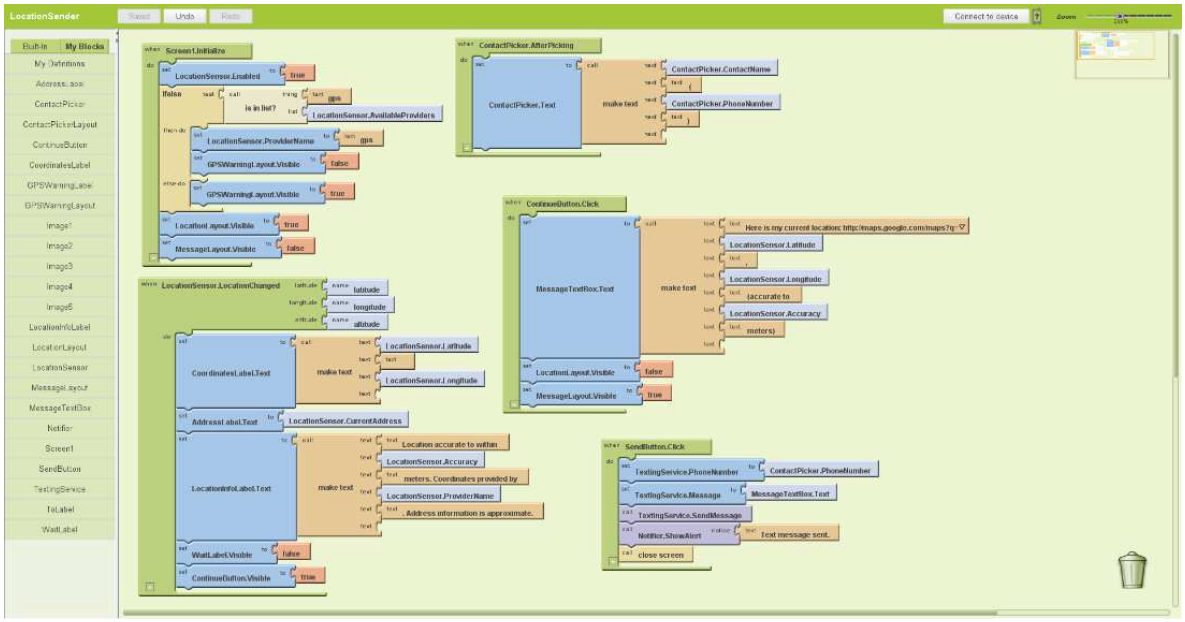

Figura 15 – Módulo de programação do *AppInventor*. Programação intuitiva através da utilização de blocos para construção da lógica.

3.7 Módulos de entrada e saída

Em um sistema de controle e automação residencial vários parâmetros são de interesse para serem monitorados. A temperatura ambiente, a umidade do solo para controle da irrigação, a presença de luminosidade para acionamento da iluminação externa, a detecção de presença para acionamento de circuitos de segurança. Para avaliar todos esses parâmetros são necessários diversos tipos de sensores.

No projeto do sistema de automação desenvolvido, foram contemplados três tipos de sensores. Sensor de presença infravermelho. É bastante utilizado em projetos de segurança e alarmes residenciais para identificar a presença de pessoas em ambientes. Ele emite um sinal infravermelho que é espalhado pela redoma convexa montada na parte superior. O sinal bate em um objeto e é refletido de volta a redoma que agrupa os feixes em um detector infravermelho. Existem ajustes de temporização e de sensibilidade, de forma que o sensor não capte qualquer variação de movimento instantânea. O sinal de saída do sensor é um sinal digital, 1 para detecção e 0 para não detecção de presença.

Figura 16 – Sensor de presença infravermelho.

Como descrito anteriormente, um sensor de umidade do solo é útil para automatização do processo de irrigação em uma casa. O sensor contém duas barras condutoras, separadas por um espaçamento. Quando a umidade do solo é muito baixa, não existe contato entre as duas barras condutoras e o circuito não é fechado. Caso a umidade do solo aumente, passa a existir condutividade entre as duas barras e o sinal de saída aumenta, a medida que a condutividade aumenta. Portanto, a saída desse sensor é

um sinal analógico que varia de 0 V a 5 V e é linearmente proporcional à quantidade de umidade do solo: quanto maior a umidade no solo, maior a tensão medida.

Para ler o valor do sensor no Arduino existem duas possibilidades. A primeira consiste em conectar o sensor a uma entrada analógica e medir o valor atual de tensão e decidir qual ação tomar a medida que o valor aumenta ou diminui. A segunda maneira é escolher um valor limite, a partir do qual seria necessário, por exemplo, ligar a válvula de irrigação, e construir um circuito de acionamento externo que enviasse um sinal para o Arduino indicando que o limite foi atingido.

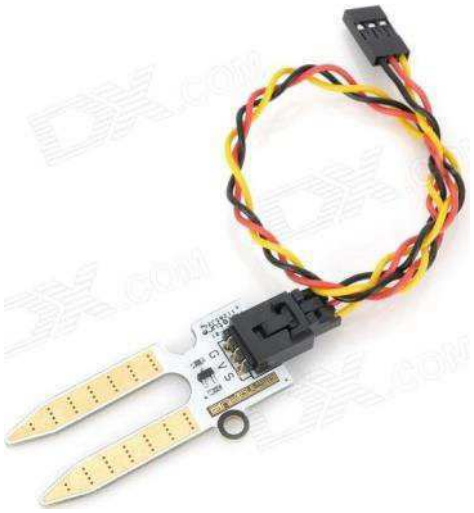

Figura 17 – sensor de umidade do solo.

O terceiro tipo de sensor utilizado também tem relação direta com a segurança da residência. É um conjunto de sensor para detectar a presença de gás e chamas. A detecção de gás é de suma importância em uma residência para verificar o vazamento de gás de cozinha e prevenir problemas maiores como um incêndio. Também pode ser utilizado em conjunto com um sensor detector de chamas para constatar a presença de fogo e fumaça e acionar alarmes de incêndio.

A saída de ambos os sensores é um valor analógico variando de 0 V a 5 V, proporcional a intensidade de chama ou de gás no ambiente. Ambos os sensores também possibilitam que se use uma saída digital quando um determinado valor for medido, ou seja, através dos *trimpots* configura-se um limiar para que a partir daquele valor, seja enviado um sinal ativo da detecção de chama ou de gás no ambiente.

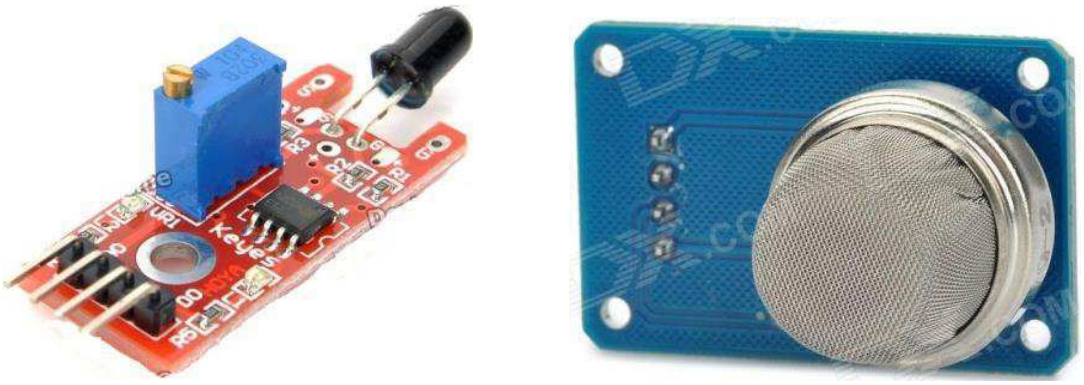

Figura 18 – Sensor detector de chama e sensor detector de gás.

Além dos sensores como elementos de entrada, são essenciais a existência de interruptores para acionamento de circuitos de iluminação ou irrigação, por exemplo. No sistema de automação residencial proposto, pretende-se utilizar interruptores paralelos, que permitem a utilização do esquema *three-way,* para conexão de dois circuitos para acionar uma carga de forma simultânea.

Figura 19 – Interruptor paralelo ou three-way, necessário para fazer a instalação elétrica com acionamento em dois pontos distintos de forma simultânea.

Uma das formas mais simples de automação residencial é o controle automático, ou diferenciado, de cargas do tipo liga / desliga. A principal carga que se encaixa nesse perfil é a carga de iluminação. Outros tipos de cargas que podem ser automatizadas através de controles ON / OFF são eletrodomésticos ou motores de portões de garagem.

O principal elemento de circuito para o controle de uma carga, como descrito acima, é o relé. Esse componente recebe um sinal de controle a nível TTL e aciona seus contatos eletromecânicos, podendo acionar uma carga de corrente alternada na tensão de 220 V e até 10 A, mantendo isolado de forma magnética o circuito de acionamento em tensão CC, do circuito da carga em tensão CA.

O circuito de acionamento de um relé é tal como apresentado na Figura 20. Para cada carga que se deseja controlar o acionamento, é necessário um novo circuito. O circuito U3 é uma proteção redundante para desacoplar opticamente o circuito de acionamento e o circuito da carga.

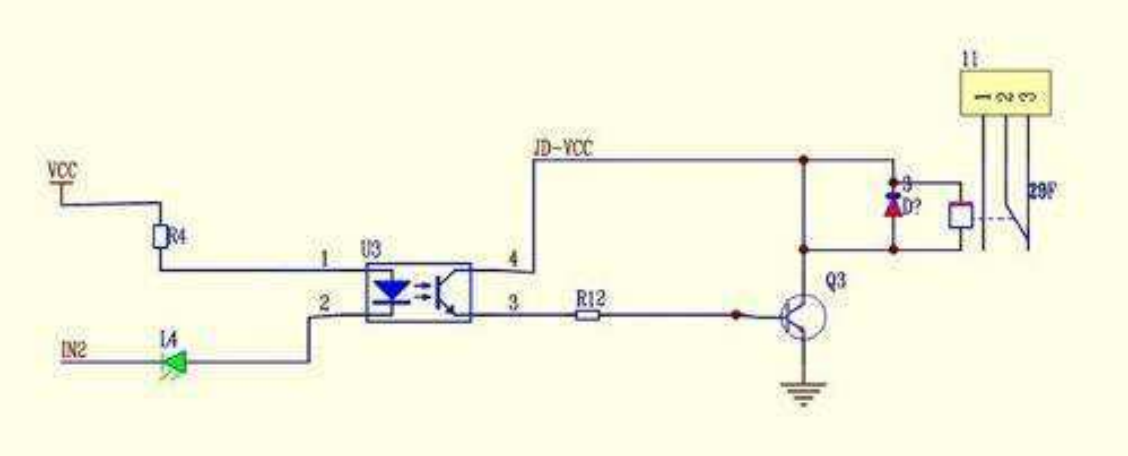

Figura 20 – Circuito de Acionamento de um relé para controle de carga.

Apesar de ser um circuito simples de montar e testar, existem placas que podem ser compradas prontas e já apresentam montagem industrial e foram previamente testadas. O preço dessas placas, quando compradas em sites de importação de eletrônicos sai mais em conta do que comprar os componentes e confeccionar uma placa de circuito impresso no Brasil. Uma placa comprada em um desses sites pode ser vista na Figura 21.

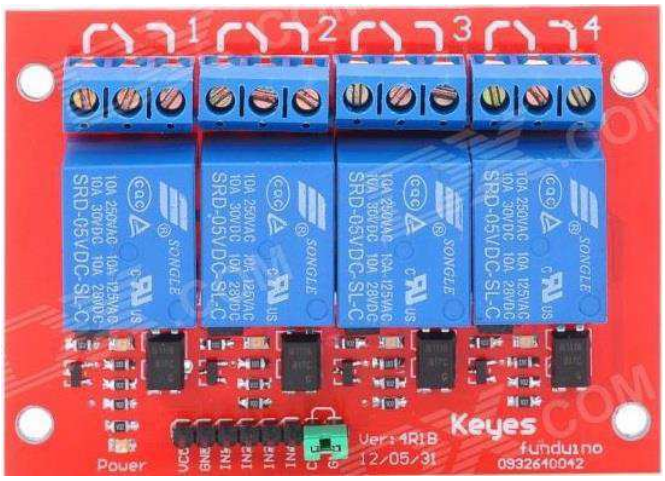

Figura 21 – Placa com quatro relés acionado por sinal TTL – 5 V, com saídas 220 Vca 10A.

Outro módulo que está presente em sistemas de automação comercial são os *dimerizadores*. Em alguns sistemas esse circuito se apresenta como *pulsadores*. É um circuito que diminui a quantidade de potência entregue a carga, basicamente com o objetivo de diminuir o consumo de energia. Em alguns tipos de circuitos é utilizado um *triac* para liberar mais ou menos tensão para carga, de acordo com o acionamento. Em outros são utilizados sistemas pulsados, controlados por microcontroladores, mais comuns em lâmpadas fluorescentes que não permitem dimerização.

No sistema desenvolvido foi utilizado um circuito com *triac*, conforme a Figura 22. O acionamento da potência entregue a carga é feito em uma interface no aplicativo no *smartphone*. O objetivo desse circuito é controlar a iluminação ambiente e consequentemente, a potência entregue a uma lâmpada halógena, que tem a possibilidade de ser dimerizável.

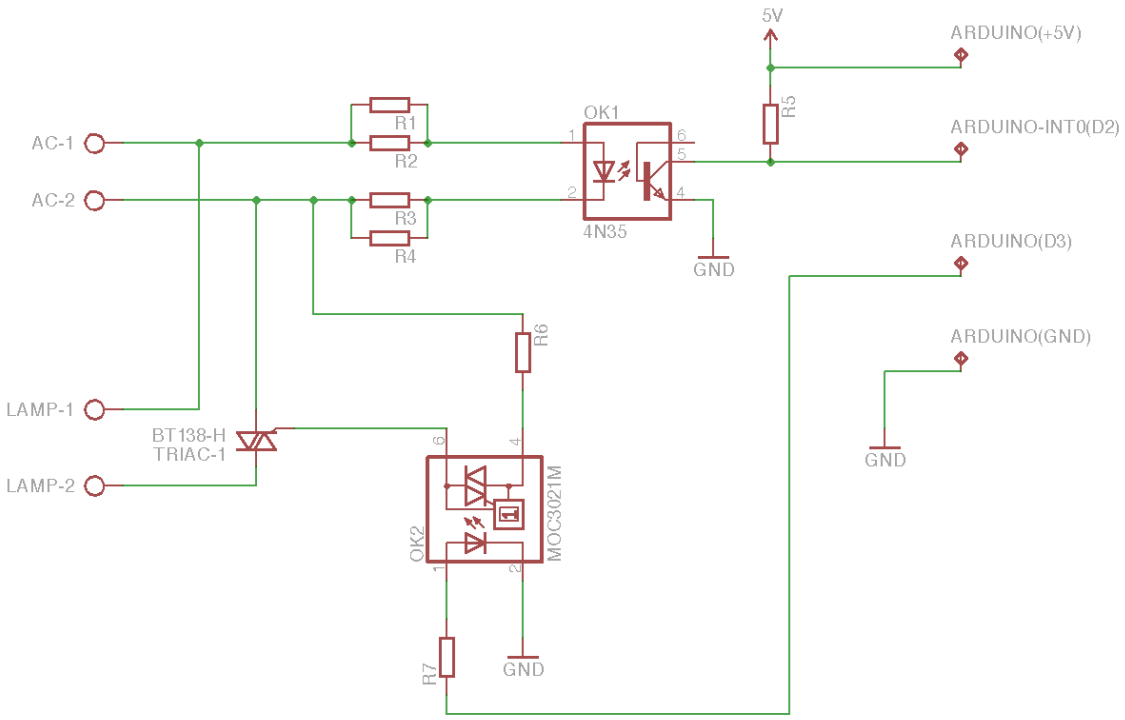

Figura 22 – Circuito para promover dimerização com o Arduino e um *triac*.

Para o caso do circuito de controle de potência, foi necessário projetar e montar uma placa externa, pois não foi encontrado no mercado um circuito que pudesse ser adquirido pronto. O circuito montado foi o apresentado na Figura 22. Na Figura 23, pode ser vista uma fotografia da placa após a montagem e testes de funcionalidade.

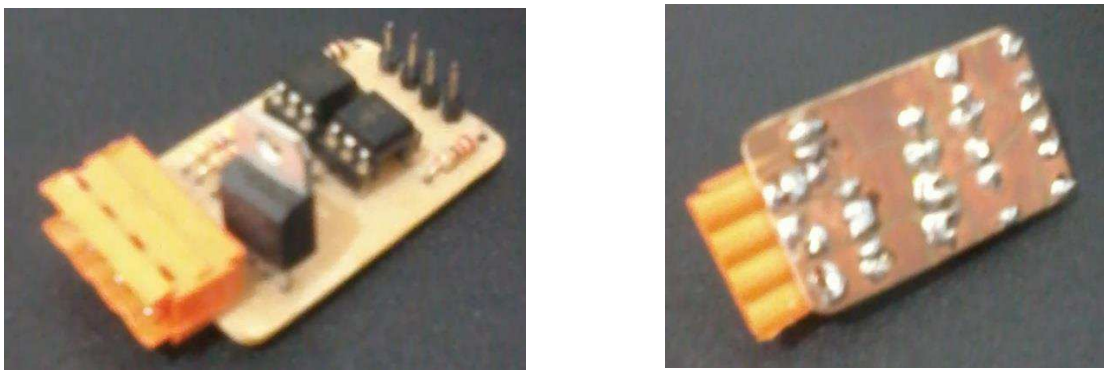

Figura 23 – fotografia da placa de controle de potência de cargas.

4 DESENVOLVIMENTO DO PROTÓTIPO EXPERIMENTAL

Com o intuito de validar o sistema de automação residencial proposto, foi pensada, projetada e construída uma plataforma experimental de testes. Essa plataforma visa simular uma instalação monofásica de uma residência com seus circuitos de instalação elétrica e algumas cargas, no caso do protótipo construído, apenas cargas de iluminação. Os sistemas de hardware descritos anteriormente foram instalados no protótipo de forma que seja possível programar o Arduino à medida que modificações fossem necessárias.

Nesse capítulo são apresentados os diagramas elétricos e aspectos construtivos do protótipo desenvolvido para simular a instalação elétrica de um residência e testar o sistema de automação residencial.

4.1 Construção do protótipo experimental

A construção de um protótipo para testar o correto funcionamento do sistema de automação residencial proposto tem como objetivo verificar a funcionalidade do sistema percebendo os automatismos atuando conforme programado no Arduino e o usuário atuando nos sistemas de controle para obter as saídas esperadas.

O protótipo portanto deveria ter como características:

- Portabilidade: não ser muito grande, podendo ser transportado para ser utilizado em apresentações.
- Baixo Custo: como não é o meio fim, os custos com o protótipo deveria ser o menor possível, devendo ser utilizado material reaproveitado de outras montagens.
- Simplicidade: com o objetivo de observar o correto funcionamento do sistema de automação, o protótipo deve ser simples o bastante para fornecer entradas e saídas, não sendo fonte de trabalho extra ou fonte de possíveis erros de projeto.

Baseado nos requisitos mínimos de funcionamento, o protótipo contém três lâmpadas fluorescentes para simular o circuito de iluminação de uma residência. Contém uma lâmpada halógena que pode ser dimerizável, para testar o controle de potência entregue as cargas. Contém ainda quatro interruptores elétricos, para ligar e desligar as cargas da forma convencional e uma tomada, para alimentar o circuito DC do Arduino e dos componentes do sistema de automação. A Figura 24 apresenta um esquemático proposto para a construção do protótipo. Na vista superior pode ser visto o espaço reservado para as lâmpadas e também para a instalação do sistema de automação.

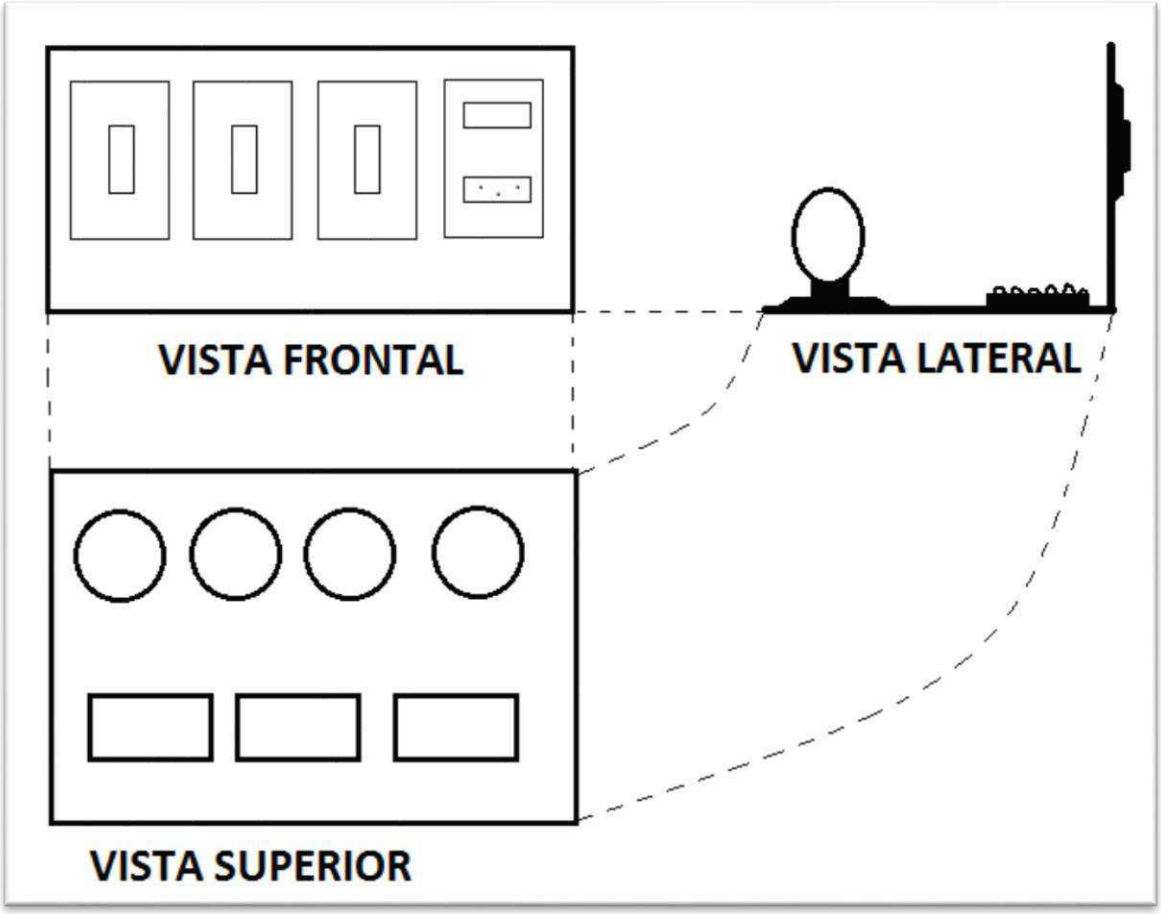

Figura 24 – Vistas do sistema de protótipo construído para testar o sistema de automação residencial proposto.

Após o levantamento dos componentes necessário para a montagem, foi feita a compra e em seguida a montagem do protótipo. Foram usadas duas chapas de madeira para fazer a base horizontal e vertical. Foi utilizado fio de 1 mm para as conexões elétricas e foram fincados com pregos as placas de circuitos do sistema de automação residencial. As fotografias vistas na Figura 25 exemplificam a evolução da montagem do projeto e o resultado final. Na Figura 26 é apresentado o diagrama multifilar da instalação elétrica do projeto.

Figura 25– Construção do protótipo e versão final já com as conexões elétricas efetuadas conjuntamente com o sistema de automação residencial.

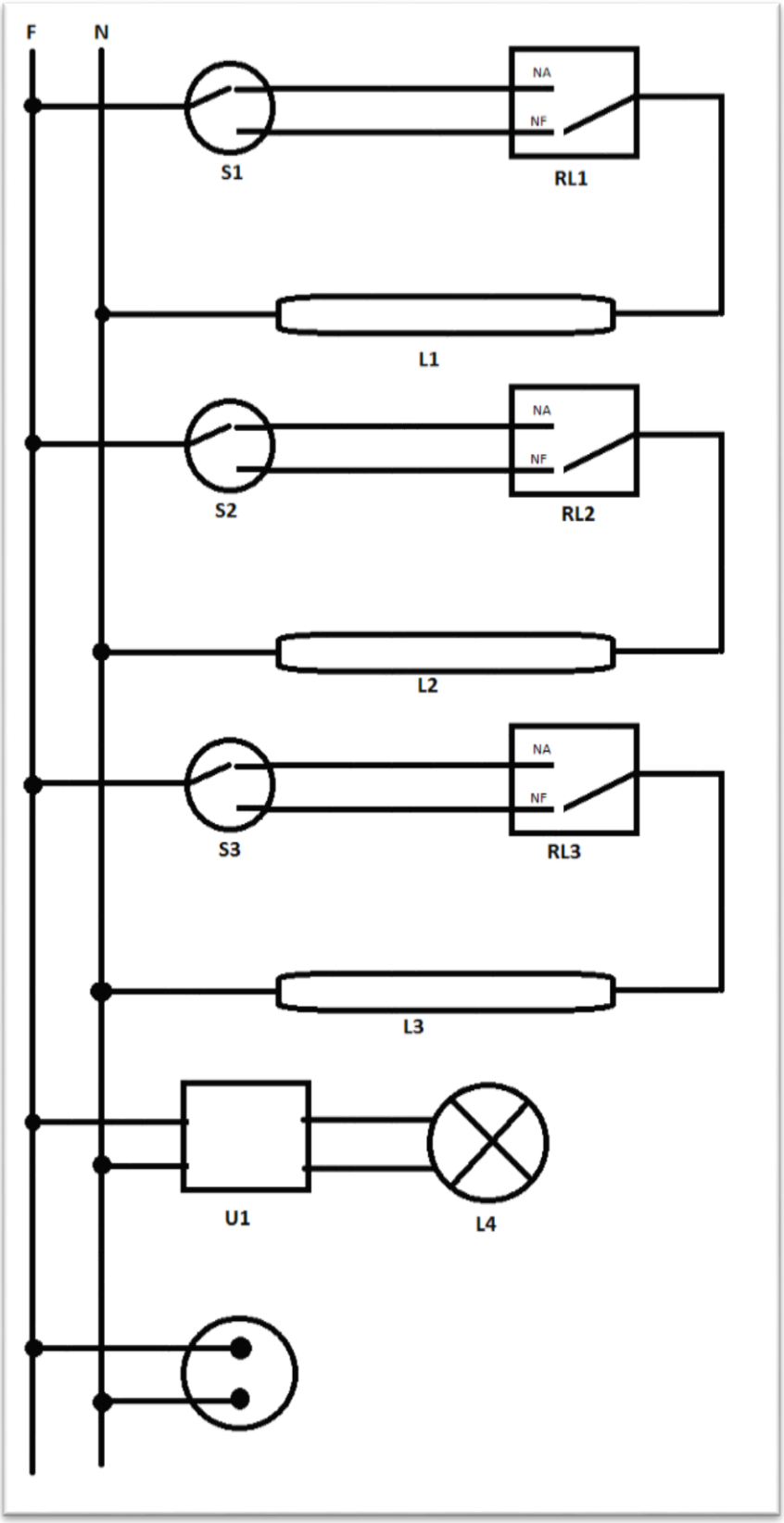

Figura 26 – Diagrama multifilar da instalação elétrica do protótipo, onde RLn são os relés do módulo de acionamento de cargas e U1 é o circuito de controle de potência entregue a carga, apresentado anteriormente.

5 TESTES E RESULTADOS

Nesse capítulo é apresentado a concepção e teste de cada parte do sistema de forma modular. Serão apresentados algoritmos utilizados para o implementação de cada módulo de controle do sistema e em anexo os códigos em linguagem proprietária do Arduino para executar as funções propostas.

5.1 Aplicativo *Domotic Blue Server*

Como explicado anteriormente foi desenvolvido um aplicativo para o sistema operacional Android, de forma que fosse possível controlar o sistema através da conexão *bluetooth* ou internet. Foi utilizado o *applet AppInventor*, do MIT, que utiliza programação em blocos e roda através de um *web browser*, para realizar a programação do aplicativo. O programa contém basicamente cinco telas:

- Tela Inicial – Assim que o usuário acessa o programa a tela inicial é apresentada e duas opções de ação são permitidas, a escolha de controlar o sistema através do *bluetooth* ou da internet. A seleção é feita clicando-se no botão com a opção desejada, conforme Figura 27.
- Tela *Bluetooth* – Selecionada a opção de controle via *bluetooth*, o usuário é redirecionado para outra tela, onde deve escolher a qual dispositivo *bluetooth* deseja se conectar. Nesse instante, caso o *hardware bluetooth* do telefone não esteja ativado, é requisitado que ele seja ligado. A Figura 28 apresenta uma captura dessa tela. Nesse momento o elemento *voltar* é introduzido em formato de "seta", de forma que é possível que o usuário retorne a tela anterior ao clicar nesse objeto.
- Tela de Controle *Bluetooth* – Após conectado corretamente ao dispositivo *bluetooth* as opções de controle do sistema são apresentadas. Basicamente as opções se resumem a ligar o desligar uma das três lâmpadas fluorescentes do protótipo de testes ou aumentar e diminuir a potência entregue a lâmpada halógena. Novamente o objeto *voltar* é apresentado, dessa vez, caso clique sobre o objeto o usuário é levado para tela inicial (Figura 29).
- Tela Controle Internet – Ao clicar na opção internet na tela inicial, o usuário é redirecionado para a página do servidor que roda no Arduino, conectado a rede

local em sua residência. Essa página é apresentada em formato *web* utilizando um dos elementos disponíveis no *AppInventor*, conforme Figura 30.

O Aplicativo é compilado diretamente no *browser* de acesso a internet, e pode ser baixado para o computador e instalado em qualquer dispositivo Android que seja de versão 2.3 ou superior. Também pode ser instalado em *tablets* Android, contudo a formatação de tela pode variar.

Figura 27 – Tela inicial do aplicativo.

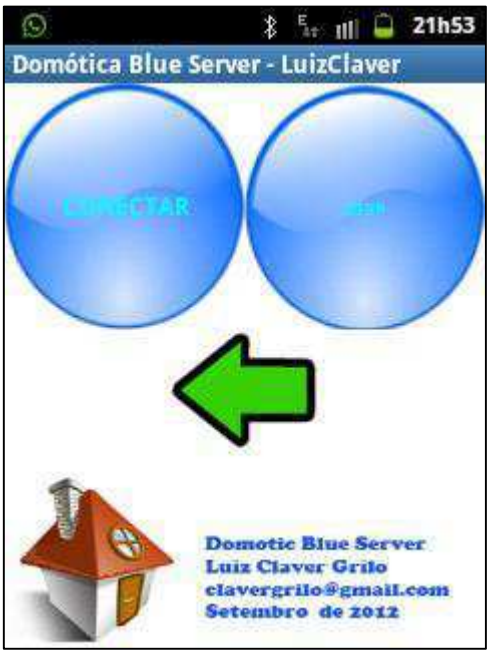

Figura 28 – Tela de seleção da conexão com hardware bluetooth ou retorno para tela anterior.

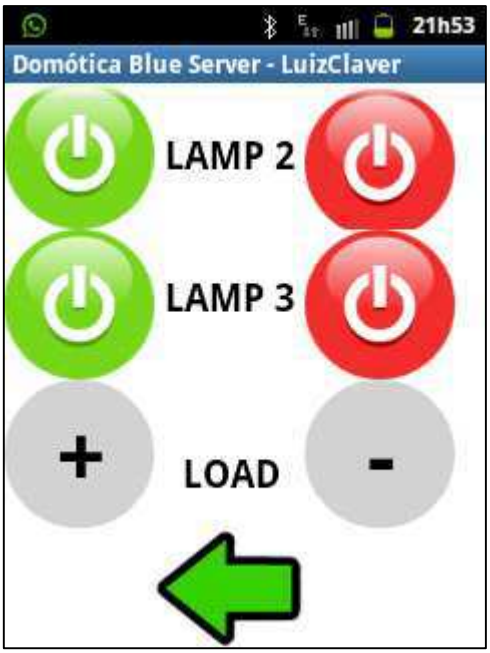

Figura 29 – Tela de controle dos dispositivos de instalação elétrica do protótipo, três lâmpadas (ação liga / desliga) e uma carga luminosa, (ação aumenta ou diminui intensidade luminosa).

Figura 30 – Captura da tela de controle do sistema via internet.

5.2 Módulos de Controle e Lógica no Arduino

Como explicado anteriormente, o Arduino funciona como uma placa de desenvolvimentos que contém toda a lógica de necessária para processar as entradas digitais, analógicas e protocolos de comunicação e acionar os sinais de saída da forma

esperada pelo usuário. Alguns controles são feitos através da interação com o usuário, utilizando meios como *smartphone* ou internet, e outros controles são feitos de forma automática, utilizando leituras provenientes de sensores. Essa seção visa explicar a programação de cada módulo de controle no Arduino.

5.2.1 Controle dos Relés

Um dos principais controles possíveis no sistema é ligar ou desligar cargas através do smartphone, utilizando comunicação bluetooth, ou através da internet. Os comandos recebidos pelo Arduino, seja por um ou pelo outro meio de comunicação, são processados e acionam ou não as saídas digitais da placa, que por sua vez comutam os contatos de um relé específico na placa de relés utilizada. O algoritmo para o controle via comunicação bluetooth é apresentado na Figura 31 e a tradução em código para o *sketch* Arduino é apresentada nos anexos.

5.2.2 Automatismos através dos sensores

Como explicado anteriormente, o sistema possui alguns sensores para monitorar parâmetros de interesse. Os sensores tem sua saída em formato analógico, portanto as portas do conversor analógico para digital do Arduino foram utilizadas para ler os dados e converter em um número que pudesse ser utilizado em uma lógica de automatismo.

Como exemplo podemos citar o sensor de medição da umidade do solo. Esse sensor apresente um sinal de saída que varia de 0 V a 5 V, de forma que, quanto maior a umidade, maior a tensão medida. Acontece que a saída desse tipo de sensor varia bastante, devido a diversos fatores, como por exemplo flutuação na tensão de referência do microcontrolador. Portanto no código do processamento da leitura analógica é utilizado um filtro por software para ter um valor médio da medida no tempo e enfim acionar a saída digital necessária, no caso um dos relés para acionar uma válvula. A Figura 32 apresenta o código utilizado no Arduino.

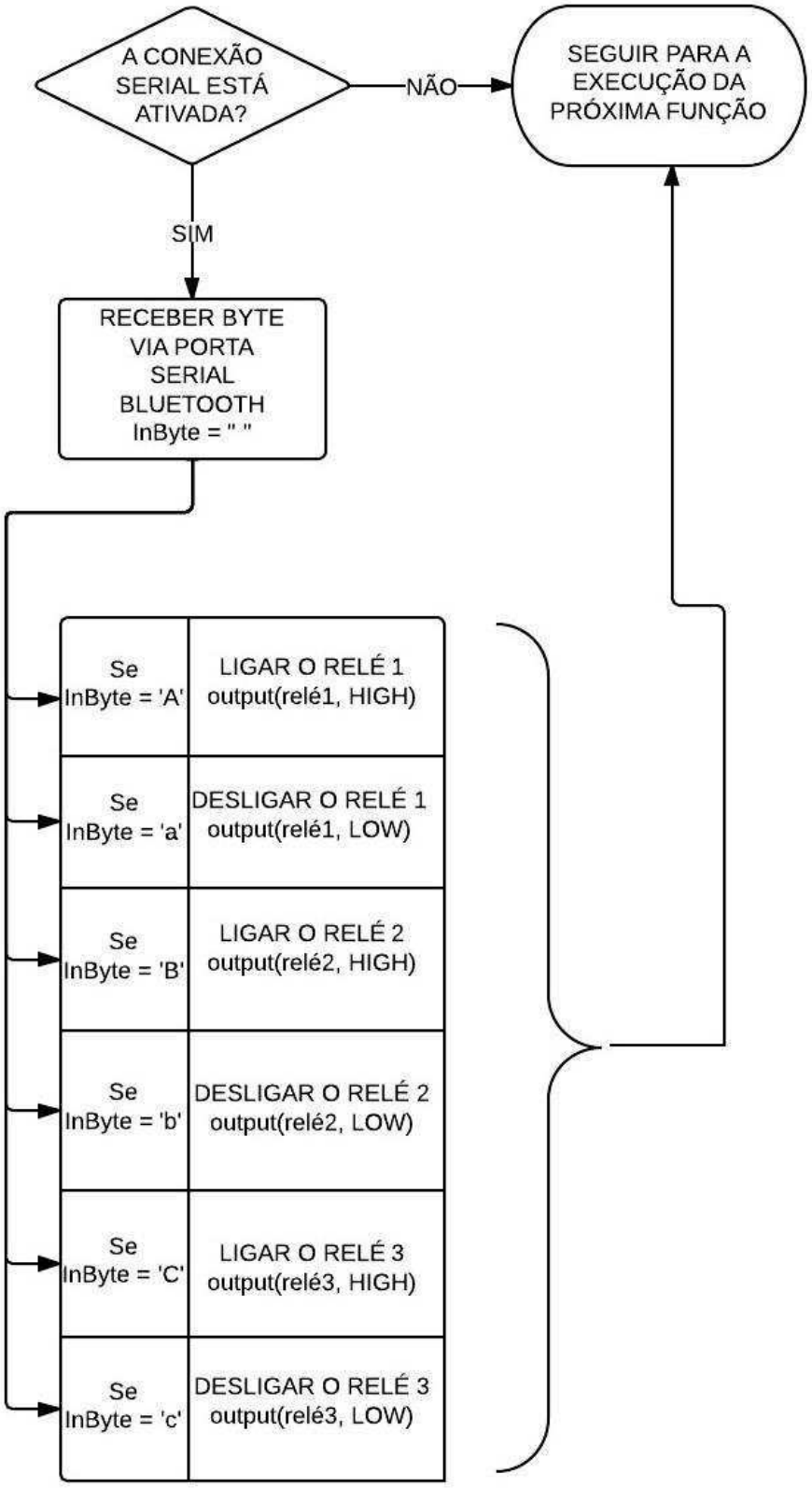

Figura 31 – Algoritmo para processamento dos dados recebidos na comunicação *bluetooth* e acionamento das saídas digitais.

```
/*
FUNÇÃO:        SensorJardim()
PARÂMETROS:    Nenhum
RETORNO:       Nenhum
DESCRIÇÃO:     Primeiramente a função calcula uma média de 50 valores medidos pelo sensor
               de forma que a medição analógica fique mais suave, uma espécie de filtro por software
               Após isso verifica se o valor médio medido na porta analógica A2 é menor que um valor base
               se for menor, aciona um contato de relé ligado a uma válvula que por sua vez
               aciona o sistema de irrigação da casa.
               O valor medido na porta analógica é a umidade do solo.
               Quando é um valor alto significa que o solo está umido, caso contrário seco.
*/
void SensorJardim(void) {

if(samplesJardim <= 50){                       //verifica se ja atingiu o numero mínimo de amostragens do sinal
accJardim = analogRead(sensorIn3) + accJardim;//vai acumulando o valor amostrado pra depois calcular a média
samplesJardim = samplesJardim + 1;             //vai atualizando o número de amostras medidas
}
else{                                          //quando medir 50 amostras,
mediaJardim = accJardim/samplesJardim;         //calcula a média dos valores amostrados
samplesJardim = 0;                             //reseta o contador de amostras
accJardim = 0;                                 //reseta o acumulador de amostras

if(mediaJardim < 100)  estadoRele4 = true;   //IMPLICA EM UMIDADE MUITO BAIXA, ENTÃO ATIVA O RELÉ PRA ACIONAR
A VÁLVULA DE IRRIGAÇÃO
else                estadoRele4 = false;    //CASO CONTRÁRIO A UMIDADE ESTÁ SUFICIENTE, ENTÃO DESLIGA A
IRRIGAÇÃO.

digitalWrite(releJardim, estadoRele4);
        }
}
```

Figura 32 - Código para o sketch do Arduino.

5.2.3 Página Web do Servidor Implementado

A página web do servidor implementado no Arduino tem a restrição básica de ocupar a menor quantidade de memória possível. Isso tem implicação direta na quantidade de comandos *client.println("");* utilizados. Portanto um estudo detalhado de somente o que foi necessário foi colocado na página. O código HTTP da página é apresentado nos anexos, e a Figura 30, apresentada anteriormente apresenta uma captura de tela da página.

O servidor implementado no Arduino faz uso da biblioteca disponibilizada de forma gratuita na instalação da IDE. Essa biblioteca possui as seguintes funções que foram utilizadas:

- *Ethernet.begin(mac, ip, gateway, subnet):* configura uma conexão ethernet com os respectivos valores de IP, *gateway e subnet* salvos nas variáveis.
- *EthernetClient client = server.available():* duas funções utilizadas em conjunto. Uma para criar um objeto cliente e outra para verificar se o cliente está disponível para se conectar ao servidor.
- *client.connected() e client.available():* funções que retornam 1 ou 0, se o cliente está conectado e se ele está disponível para escrita ou leitura, respectivamente.

- *char c = client.read():* Ler dados do cliente. Essa função lê apenas um byte, por isso precisa ser executada em *loop* para ler a quantidade de bytes necessário para conter uma mensagem de requisição.
- *client.println(" "):* função utilizada para escrever dados para o cliente conectado. Aqui são enviados, por exemplo, as linhas de código da página do servidor.

6 CONCLUSÃO E TRABALHOS FUTUROS

Foi realizado um projeto de um sistema para automação residencial utilizando microcontrolador como unidade central de processamento. Sistemas de automação residencial são sistemas modulares, hoje em dia são em sua maioria construídos utilizando-se tecnologia dos modernos processadores ou dos robustos CLPs, buscou-se realizar uma montagem intermediaria fazendo proveito dos benefícios da portabilidade dos microcontroladores para obter um sistema intermediário entre a performance dos processadores e a robustez dos CLP.

Foi desenvolvido um protótipo para testar e validar o sistema de automação desenvolvido. O protótipo teve como objetivo simular a instalação elétrica simplificada de uma residência e permitir que os sinais básicos de entrada e saída fossem aplicados ao sistema desenvolvido no microcontrolador a fim de verificar o correto controle e funcionamento das funções pensadas e programadas no Arduino.

A programação e montagem do sistema ocorreu de forma positiva e o objetivo esperado foi alcançado. Um sistema completo para auxiliar no estudo e desenvolvimento de tecnologias para automação residencial foi desenvolvido e pode ser usado como modelo para estudos mais avançados. Espera-se que recursos presentes em projetos de automação residencial similares, possam ser acrescentados ao projeto com auxílio da estrutura desenvolvida em futuros trabalhos. Controle de equipamentos via sinais infravermelho, inclusão de um sistema em tempo real para inserção de um *log* de eventos ou programação de ações baseadas no tempo, sejam frutos de trabalhos futuros.

O presente trabalho aborda vários aspectos de um projeto de engenharia elétrica, ao se debater sobre temas como eletrônica e microeletrônica, instalações elétricas prediais, arquitetura de sistemas e programação de processadores e construção de softwares específicos. Também contempla a montagem experimental, resultando no desenvolvimento de um produto que pode ser usado como modelo para desenvolvimento de outros produtos e tecnologias e no levantamento para construção de protótipos e atividades experimentais, tão importantes no desenvolvimento e formação de um profissional de engenharia elétrica.

7 BIBLIOGRAFIA

COSTA, E. M. M.; LIMA, A. M. N. *Sistemas dinâmicos a eventos discretos: fundamentos básicos para a moderna automação industrial*. Salvador: EDUFBA, 2005.

GRILO, L. C. P.; GRILO, M. B.; PEREIRA, E. G. *Desenvolvimento de um sistema para medição de temperatura e umidade com comunicação sem fio num secador solar*. Congresso Nacional de Engenharia Mecânica (CONEM). São Luis - MA, 2012.

JOBSTRAIBIZER, F. *Criação de aplicativos para celulares com Google Android*. São Paulo: Digerati Books, 2009.

McROBERTS, M. *Arduino básico*. Trad. Rafael Zanolli. São Paulo: Novatec, 2011.

PRUDENTE, F. *Automação predial e residencial: uma introdução*. 1. ed. São Paulo: LTC, 2011.

Revista VEJA. *Um lar que obedece ao dono*. Edição 2299, 12 de dezembro de 2012, p. 176-178.

SURE ELECTRONICS. *Bluetooth serial converter UART interface 9600bps user's guide*. P.R.Chine, Sure Electronics Co. Ltd, 2008.

THOMAZINI. D.; ALBUQUERQUE, P. U. B. *Sensores Industriais: fundamentos e aplicações*. 7. Ed. rev. e atual. São Paulo: Érica, 2010.

http://arduino.cc/en/Main/ArduinoBoardUno - Acessado em 10/2013

http://arduino.cc/en/Main/ArduinoEthernetShield - Acessado em 10/2013

8 ANEXOS

```
/*********************************************
PROGRAMA FINAL DO TCC - DomoticGrilo          *
LUIZ CLAVER PEREIRA GRILO                     *
clavergrilo@gmail.com                         *
*********************************************/
#include <DS1302.h>
#include <SPI.h>
#include <Ethernet.h>

//*****************DEFINIÇÃO DOS PINOS DO ARDUINO ************************************
#define sensorIn1     A0  //A0 - ENTRADA ANALÓGICA 1 (SENSOR DE CHAMAS)  <- INPUT
#define sensorIn2     A1  //A1 - ENTRADA ANALÓGICA 2 (SENSOR DE GÁS)     <- INPUT
#define sensorIn3     A2  //A2 - ENTRADA ANALÓGICA 3 (SENSOR DE UMIDADE) <- INPUT
                          //0 - TX BLUETOOTH                            <- INPUT
                          //1 - RX BLUETOOTH                            -> OUTPUT
#define releLamp1     2   //2 - RELÉ 1 - LÂMPADA 1                      -> OUTPUT
#define releLamp2     3   //3 - RELÉ 2 - LÂMPADA 2                      -> OUTPUT
                          //4 - ETHERNNET SHIELD (definir como OUTPUT)  -> INPUT
#define releLamp3     5   //5 - RELÉ 3 - LÂMPADA 3                      -> OUTPUT
#define releJardim    6   //6 - RELÉ 4 - VÁLVULA DE IRRIGAÇÃO           -> OUTPUT
#define alarme        7   //7 - ALARME - LED + BUZZER                   -> OUTPUT
#define triacIn       8   //8 - TRIAC - INTERRUPT IN                    <- INPUT
#define triacOut      9   //9 - TRIAC - PWM OUTPUT                      -> OUTPUT PWM
                          //10- ETHERNNET SHIELD                        -> INPUT
                          //11- ETHERNNET SHIELD                        -> INPUT
                          //12- ETHERNNET SHIELD                        -> INPUT
                          //13- ETHERNNET SHIELD                        -> INPUT
//DEFINIÇÕES DAS AÇÕES
#define action_none 0
#define action_mypin_ON 1
#define action_mypin_OFF 2

//************** DEFINIÇÃO DAS VARIÁVEIS GLOBAIS UTILIZADAS NO PROGRAMA ************

char    inByte;             //ARMAZENA O BYTE CHEGANDO NA SERIAL
boolean estadoRele1 = false; //ARMAZENA O ESTADO DO RELÉ 1
boolean estadoRele2 = false; //ARMAZENA O ESTADO DO RELÉ 1
boolean estadoRele3 = false; //ARMAZENA O ESTADO DO RELÉ 1
boolean estadoRele4 = false; //ARMAZENA O ESTADO DO RELÉ 1
boolean estadoSensorIn2 = false;
boolean estadoSensorIn1 = false;
long accJardim, accGas, accIn1;
int samplesJardim, samplesGas, samplesIn1;
int mediaJardim, mediaGas, mediaIn1;

//DEFINIÇÕES DA REDE
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xBE }; //ENDEREÇO DO MAC DEFINIDO PELO
USUÁRIO
byte ip[] = { 192, 168, 0, 177 };                    //ENDEREÇO DE IP, DEFINIDO PELO
USUÁRIO
byte gateway[] = { 192, 168, 0, 1 };                 //internet access via router
(opcional)
byte subnet[] = { 255, 255, 255, 0 };                //subnet mask  (opcional)
EthernetServer server(3333);                         //PORTA DO SERVIDOR. NECESSÁRIA.
String readString = String(30);                      //string para receber as ações de
ligar/desligar os pinos

//PINOS DE SAÍDA QUE ESTÃO SENDO USADOS E PARA O QUE ESTÃO SENDO USADOS
int pinOutRelay = 7;
int inByte = 0;           //BYTE RECEBIDO PELO BLUETOOTH PARA FAZER A COMPARAÇÃO E ATUAR
NAS SAÍDAS.

//COMANDOS DE CHEGADA DO CLIENT / QUANDO CLICA NOS BOTÕES NA PÁGINA ENVIA ESSES COMANDOS
PARA O ARDUINO INTERPRETAR
String pinOutRelay_OFF = "GET /?out=5&status=0 HTTP/1.1";
String pinOutRelay_ON = "GET /?out=5&status=1 HTTP/1.1";

//VARIÁVEL PARA GUARDAR A AÇÃO ATUAL, PARA INFORMAR AO USUÁRIO O ESTÁGIO DO(S) PINO(S)
EM CONTROLE
int current_action;
//***********************CONFIGURAÇÕES DE SETUP INICIAL*****************************
void setup()
{
  //INICIA A PORTA ETHERNET COM AS CONFIGURAÇÕES ESCOLHIDAS
  Ethernet.begin(mac, ip, gateway, subnet);
  current_action = 0;

  // INICIA A PORTA SERIAL A 9600 bps
  Serial.begin(9600);

  //DEFINE TODOS OS PINOS ONDE ESTÃO LIGADOS OS RELÉS, COMO SAÍDA
  pinMode(releLamp1, OUTPUT);   digitalWrite(releLamp1, LOW);
  pinMode(releLamp2, OUTPUT);   digitalWrite(releLamp2, LOW);
```

```
  pinMode(releLamp3, OUTPUT);   digitalWrite(releLamp3, LOW);
  pinMode(releJardim, OUTPUT);  digitalWrite(releJardim, LOW);
  //DEFINE TODOS OS PINOS ONDE ESTÃO LIGADOS OS INTERRUPTORES COMO ENTRADA
}

//********************** PROGRAMA QUE FICA SE REPETINDO ******************************
void loop()
{
  EstadoReles();    //chama a função de acionamento dos relés por bluetooth serial

  SensorJardim();   //chama a função de leitura do sensor de umidade do solo e
acionamento de válvula de irrigação

  SensorGas();      //chama a função da leitura do sensor de gás

  SensorIn1();      //chama a função da leitura do estado de um terceiro sensor

  AtivaAlarme();    //chama a função pra ativar o alarme, se o sinal dos sensores
atingir determinado nível

  internet();        //chama a função relógio

  delay(100);
} //LooP

//************************ FUNÇÕES ****************************************
/*
FUNÇÃO:      EstadoReles()
PARÂMETROS:  Nenhum
RETORNO:     Nenhum
DESCRIÇÃO:   Verifica se o canal de comunicação serial está disponível. Caso esteja
recebe um byte pelo canal
             Faz 6 (seis) comparações para atualizar o estado dos relés, LIGADO ou
DESLIGADO
             Salva nas variáveis de estado (BOOLEAN) o estado atual do relé, para futura
conferência.
*/
void EstadoReles (void) {

    if (Serial.available() > 0) {            //SE A PORTA SERIAL ESTIVER DISPONÍVEL
    inByte = Serial.read();                 //LÊ O BYTE DISPONÍVEL NO CANAL DE
COMUNICAÇÃO DE ENTRADA

    if(inByte == 'A'){                      //SE RECEBEU O BYTE 'A' LIGA O RELÉ DA SAÍDA
1
      estadoRele1 = true;
      inByte = 0;    }

    if(inByte == 'a'){
      estadoRele1 = false;          //SE RECEBEU O BYTE 'a' DESLIGA O RELÉ DA SAÍDA 1
      inByte = 0;    }

    if(inByte == 'B'){
      estadoRele2 = true;       //SE RECEBEU O BYTE 'B' LIGA O RELÉ DA SAÍDA 2
      inByte = 0;    }

    if(inByte == 'b'){
      estadoRele2 = false;       //SE RECEBEU O BYTE 'b' DESLIGA O RELÉ DA SAÍDA 2
      inByte = 0;    }

    if(inByte == 'C'){
      estadoRele3 = true;     //SE RECEBEU O BYTE 'C' LIGA O RELÉ DA SAÍDA 3
      inByte = 0;    }

    if(inByte == 'c'){
      estadoRele3 = false;    //SE RECEBEU O BYTE 'c' DESLIGA O RELÉ DA SAÍDA 3
      inByte = 0;    }
  } // =Serial Available

   digitalWrite(releLamp1, estadoRele1);
   digitalWrite(releLamp2, estadoRele2);
   digitalWrite(releLamp3, estadoRele3);
}
//*****************************************************************************************
/*
FUNÇÃO:      SensorJardim()
PARÂMETROS:  Nenhum
RETORNO:     Nenhum
DESCRIÇÃO:   Primeiramente a função calcula uma média de 50 valores medidos pelo sensor
             de forma que a medição analógica fique mais suave, uma espécie de filtro
por software
             Após isso verifica se o valor médio medido na porta analógica A2 é menor
que um valor base
             se for menor, aciona um contato de relé ligado a uma válvula que por sua
vez
             aciona o sistema de irrigação da casa.
             O valor medido na porta analógica é a umidade do solo.
```

```
            Quando é um valor alto significa que o solo está umido, caso contrário
seco.
*/
void SensorJardim(void) {

  if(samplesJardim <= 50){                          //verifica se ja atingiu o numero
mínimo de amostragens do sinal
    accJardim = analogRead(sensorIn3) + accJardim;  //vai acumulando o valor amostrado
pra depois calcular a média
    samplesJardim = samplesJardim + 1;              //vai atualizando o número de
amostras medidas
  }
  else{                                             //quando medir 50 amostras,
    mediaJardim = accJardim/samplesJardim;          //calcula a média dos valores
amostrados
    samplesJardim = 0;                              //reseta o contador de amostras
    accJardim = 0;                                  //reseta o acumulador de amostras

    if(mediaJardim < 100)  estadoRele4 = true;    //IMPLICA EM UMIDADE MUITO BAIXA,
ENTÃO ATIVA O RELÉ PRA ACIONAR A VÁLVULA DE IRRIGAÇÃO
      else                 estadoRele4 = false;   //CASO CONTRÁRIO A UMIDADE ESTÁ
SUFICIENTE, ENTÃO DESLIGA A IRRIGAÇÃO.

    digitalWrite(releJardim, estadoRele4);
  }
}
//*****************************************************************************************
/*
FUNÇÃO:      SensorGas()
PARÂMETROS:  Nenhum
RETORNO:     Nenhum
DESCRIÇÃO:
*/
void SensorGas(void) {

  if(samplesGas <= 50){                    //verifica se ja atingiu o numero mínimo de
amostragens do sinal
    accGas = analogRead(sensorIn2) + accGas;  //vai acumulando o valor amostrado pra
depois calcular a média
    samplesGas = samplesGas + 1;              //vai atualizando o número de amostras
medidas
  }
  else{                                  //quando medir 50 amostras,
    mediaGas = accGas/samplesGas;               //calcula a média dos valores
amostrados
    samplesGas = 0;                         //reseta o contador de amostras
    accGas = 0;                             //reseta o acumulador de amostras

    if(mediaGas < 100)  estadoSensorIn2 = true;    //IMPLICA EM UMIDADE MUITO BAIXA,
ENTÃO ATIVA O RELÉ PRA ACIONAR A VÁLVULA DE IRRIGAÇÃO
      else                 estadoSensorIn2 = false;   //CASO CONTRÁRIO A UMIDADE ESTÁ
SUFICIENTE, ENTÃO DESLIGA A IRRIGAÇÃO.
  }
}

//*****************************************************************************************
/*
FUNÇÃO:      SensorIn1()
PARÂMETROS:  Nenhum
RETORNO:     Nenhum
DESCRIÇÃO:
*/
void SensorIn1(void) {

  if(samplesIn1 <= 50){                    //verifica se ja atingiu o numero mínimo de
amostragens do sinal
    accIn1 = analogRead(sensorIn1) + accIn1;  //vai acumulando o valor amostrado pra
depois calcular a média
    samplesIn1 = samplesIn1 + 1;              //vai atualizando o número de amostras
medidas
  }
  else{                                  //quando medir 50 amostras,
    mediaIn1 = accIn1/samplesIn1;               //calcula a média dos valores
amostrados
    samplesIn1 = 0;                         //reseta o contador de amostras
    accIn1 = 0;                             //reseta o acumulador de amostras

    if(mediaGas < 100)  estadoSensorIn1 = true;    //IMPLICA EM UMIDADE MUITO BAIXA,
ENTÃO ATIVA O RELÉ PRA ACIONAR A VÁLVULA DE IRRIGAÇÃO
      else                 estadoSensorIn2 = false;   //CASO CONTRÁRIO A UMIDADE ESTÁ
SUFICIENTE, ENTÃO DESLIGA A IRRIGAÇÃO.

  }
}
//*****************************************************************************************
/*
FUNÇÃO:      AtivaAlarme()
```

```
PARÂMETROS:  Nenhum
RETORNO:     Nenhum
DESCRIÇÃO:
*/
void AtivaAlarme(void) {

  if(estadoSensorIn1 && estadoSensorIn2)
    digitalWrite(alarme, HIGH);
  else
    digitalWrite(alarme, LOW);
}
//*****************************************************************************************
/*
FUNÇÃO:      Internet()
PARÂMETROS:  Nenhum
RETORNO:     Nenhum
DESCRIÇÃO:   Ativa o servidor com o cliente e envia a página WEB e recebe as requisições
ativando as ações esperadas.
*/
void Internet(void){
  current_action = 0;
  // CRIA UMA CONEXÃO COM O CLIENT
  EthernetClient client = server.available();
    if (client) {
      while (client.connected()) {
       if (client.available()) {
        char c = client.read();
        //LER CHAR POR CHAR DA REQUISIÇÃO HTTP E ARMAZENA OS CHARS EM UMA STRING
        if (readString.length() < 30) {
          readString = readString + c;
        }
        //SE A REQUISIÇÃO HTTP DO CLIENT TERMINOU, RECEBE ESSA INFORMAÇÃO
        if (c == '\n') {

         //Serial.print(readString);
         // COMPARA A STRING RECEBIDA DA REQUISIÇÃO HTTP, PARA VERIFICAR O QUE O USUÁRIO
DESEJA FAZER
         // GUARDA A INFORMAÇÃO DO QUE VAI SER FEITO EM current_action DE FORMA QUE É
POSSÍVEL SABER O ESTADO DO PINO
          if(readString.startsWith(pinOutRelay_ON))
              current_action = action_mypin_ON;

          else if(readString.startsWith(pinOutRelay_OFF))
              current_action = action_mypin_OFF;

          else
              current_action = action_none;

//
*****************************************************************************************
         // CRIA A PÁGINA HTTP COM CABEÇALHO PADRÃO QUE O USUÁRIO VAI VER
         // E ENVIA PARA ESSA PÁGINA AS INFORMAÇÕES ATUAIS DE ACORDO COM AS REQUISIÇÕES
DELE.
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html\n");
          client.println("<html><head>");
          client.println("<title>LuizClaver Server</title></head>");
          client.println("<body>");
          client.println("<h1>LuizClaver Server</h1>");

        //CRIAÇÃO DOS BOTÕES EM HTML. AS LINHAS DO MEIO MOSTRAM A MONTAGEM DA MENSAGEM
DE CONTROLE
        //QUE INDICA out=5&value=1 para ligado e value=0 para desligado.
          client.println();
          client.println("<form METHOD=get action=\"http://189.71.116.227:3333/\">");
//OBSERVAR QUE ESSE É O IP_FIXO DO ROTEADOR
          client.println("<input type=hidden name=\"out\" value=\"5\">");
//SE NÃO TIVER PLANO DE IP_FIXO, TEM QUE MUDAR TODA VEZ
          client.println("<input type=hidden name=\"status\" value=\"1\">");
//QUE O ROTEADOR REINICIAR, OU QUE A INTERNET REINICIAR
          client.println("<input type=submit value=\"LIGAR\">");

          client.println("<form METHOD=get action=\"http://189.71.116.227:3333/\">");
//DE FORMA SIMILAR AQUI O COMANDO DEPENDE DO IP
          client.println("<input type=hidden name=\"out\" value=\"5\">");
//COMO FUTURA SOLUÇÃO PODE-SE IMPLEMENTAR ALGO COM
          client.println("<input type=hidden name=\"status\" value=\"0\">");
//DHCP AUTOMÁTICO, QUE PEGA O IP DIRETO DA REDE.
          client.println("<input type=submit value=\"DESLIGAR\">");
          client.println("</form>");
          client.println();

         switch(current_action)
         {
         case action_mypin_OFF:
           digitalWrite(pinOutRelay, LOW);
           client.println("O PINO: ");
```

```
            client.print(pinOutRelay);
            client.print(" ESTA DESLIGADO");
            client.println("<br/><br/><a href='http://189.71.116.227:8081'>CAMERA</a>");
//NOVAMENTE A QUESTÃO DO IP E AGORA ACESSANDO UMA PORTA EXTRA
            client.println("</html>");
//CONFIGURADA TAMBÉM NO ROTEADOR PARA DAR ACESSO A CÂMERA IP
            break;

          case action_mypin_ON:
            digitalWrite(pinOutRelay, HIGH);
            client.println("O PINO: ");
            client.print(pinOutRelay);
            client.print(" ESTA LIGADO");
            client.println("<br/><br/><a href='http://189.71.116.227:8081'>CAMERA</a>");
//MAIS UMA VEZ O IP AQUI DEVE SER CONHECIDO PREVIAMENTE DO ROTEADOR
            client.println("</html>");
            break;

           case action_none:
            client.println("<br/><a href='http://189.71.116.227:8081'>CAMERA</a>");
//AQUI TAMBÉM
            client.println("</html>");
            break;
          }
//
*********************************************************************************************

           //LIMPA A STRING DE COMPARAÇÃO, PARA A PRÓXIMA REQUISIÇÃO HTTP DO CLIENT
           readString="";
           //FINALIZA A CONEXÃO COM O CLIENT
           client.stop();
         }
       }
     }
  }
  delay(1);
}
```