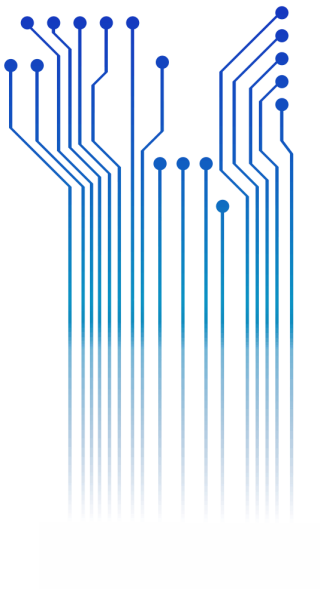
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

IGOR DANTAS ROCHA

TRABALHO DE CONCLUSÃO DE CURSO

DOMÓTICA VIA DISPOSITIVOS MÓVEIS UTILIZANDO A PLATAFORMA ARDUINO



Campina Grande
2016

IGOR DANTAS ROCHA

DOMÓTICA VIA DISPOSITIVOS MÓVEIS UTILIZANDO A PLATAFORMA ARDUINO

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: Domótica / Instrumentação Eletrônica

Orientador:
João Batista Morais dos Santos, D. Sc.

Campina Grande
2016

IGOR DANTAS ROCHA

DOMÓTICA VIA DISPOSITIVOS MÓVEIS UTILIZANDO A PLATAFORMA ARDUINO

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: Domótica / Instrumentação Eletrônica

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professor João Batista Morais dos Santos, D. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Dedico este trabalho em especial aos meus pais, por todo o apoio durante esta longa jornada.

AGRADECIMENTOS

Agradeço a Deus, em primeiro lugar, pela minha vida e pelo dom da perseverança, que me permitiu concluir este trabalho.

À minha família, por todo o carinho, amor e cuidado. Por terem me ensinado valores que pude levar por todos os lugares onde passei, ajudando-me a ser, além de um profissional melhor, uma pessoa melhor.

Aos amigos de berço, que apesar da distância, sempre se fizeram presentes em minha vida.

Ao GRUPO PROCAD, em especial a Eleazar Florido Cobos, Fracisco Javier Montiel Hijano e Javier Trigueros Heredia pela oportunidade de estagiar na Espanha durante três meses e trabalhar com o Arduino, objeto principal deste projeto.

À Pablo Alvarez Gallardo, grande amigo que ganhei durante o tempo em PROCAD, pelos ensinamentos diários na área de Eletrônica.

Ao meu professor orientador João Batista Morais dos Santos, pela disponibilidade, por toda atenção e suporte que me foi dado desde o primeiro momento.

Enfim, agradeço a todos que de alguma forma me ajudaram durante toda essa etapa.

"O insucesso é apenas uma oportunidade para recomeçar com mais inteligência"

Henry Ford.

RESUMO

A domótica é a área do conhecimento voltada ao desenvolvimento de soluções de automação residencial proporcionando, aos seus usuários, maior segurança, praticidade, economia e conforto. A origem do termo domótica vem da junção das palavras Domus, que em latim significa residência; e automática, palavra em grego que significa "que funciona por sí só". Este mercado tem crescido consideravelmente nos últimos anos, devido ao avanço da tecnologia e consequentemente o barateamento de ferramentas utilizadas para montar o sistema de automação residencial. O presente projeto tem como objetivo o desenvolvimento de um protótipo de um sistema de automação residencial utilizando a plataforma Arduino, controlada e monitorada através de dispositivos móveis. É utilizada uma maquete para simular uma residência. No modelo é implementado sistemas de controle de iluminação, irrigação, alarme, detecção de vazamento de gás e de chuva e consumo energético.

Palavras-chave: Domótica, Arduino, Dispositivos móveis, Sensores, Atuadores.

ABSTRACT

Domotics is the area of knowledge that studies the development of residential automation solutions, allowing its users to enjoy higher safety, easiness, economy and comfort along the housing experience. The origin of the term Domotica comes from the junction of the words Domus - "residence", from Latin -, and automatica - greek word that means "something that works by itself". This market has been considerably growing in the last few year, due to the advance in technology and the lower costs of hardware used to assemble residential automation system components. The objective of this project is to develop a prototype of a residential automation system using an Arduino based platform, controlled and monitored by means of mobile devices. A mockup was built to simulate a residence physical spaces. In the mockup it was used several control systems, such as lightning, irrigation, alarm, gas leakage detection, rain detection and electrical energy consumption.

Keywords: Home automation, Arduino, Mobile devices, Sensors, Actuators.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CLP	Controlador Lógico Programável
MDF	Medium Density Fiberboard
EIB	Conselho Nacional de Petróleo
KNX	Empresa de Pesquisa Energética
BACNet	Indústrias Nucleares Brasileiras
DALI	Digital Addresable Lighting Interface
PLC/BUS	Powerline Communication Bus
EHSA	European Home Systems Association
BCI	BatiBUS Club International
ABB	Asea Brown Boveri
ABI	Allied Business Intelligence
AURESIDE	Associação Brasileira de Automação Residencial
PWM	Pulse Width Modulation
UART	Universal asynchronous receiver/transmitter
USB	Universal Serial Bus
ICSP	In-Circuit Serial Programming
FTDI	Future Technology Devices International
GND	Ground
SRAM	Static Random Access Memory
EEPROM	Electrically-Erasable Programmable Read-Only Memory
A/D	Conversão Analógica/Digital
SPI	Serial Pheripheral Interface
TWI	Two Wire Interface
GPS	Global Positioning System
FEEC	Faculdade de Engenharia Elétrica e de Computação da Unicamp
IDE	Integrated Drive Eletronics
QR	Quick Response

IBM	International Business Machines
SIG	Special Interest Group
HP	Hewlett-Packard
NTT	Nippon Telegraph and Telephone Corporation
ISM	Industrial, Scientific, and Medical
LAN	Local Area Network
LDR	Light Dependent Resistor
LED	Light-emitting diode
GLP	Gás Liquefeito de Petróleo
IHM	Interface Homem-Máquina
SDK	Software development kit

SUMÁRIO

1 INTRODUÇÃO

A evolução tecnológica atual faz com que as pessoas aumentem ainda mais seus níveis de exigências a níveis de segurança, conforto e comunicação. O que há alguns anos seria utópico, hoje é mais comum do que se imagina.

É normal luzes acenderem automaticamente ao subirmos uma escada de um prédio ou ao entrar em um banheiro público. Ambientes que controlam suas temperaturas automaticamente. Com essa capacidade de "automatizar" o que antes era feito por nós mesmos, criou-se o conceito de "casas inteligentes".

Segundo Roque (2008), domótica é a utilização simultânea da eletricidade, da eletrônica e das tecnologias de informação no ambiente residencial, permitindo uma gestão local ou remota do ambiente.

O projeto trata-se de uma aplicação real de automação residencial, onde poderá ser visto diversas atividades automatizadas que costumeiramente seriam realizadas pelas pessoas.

1.1 MOTIVAÇÃO

Com o avanço da tecnologia, dispositivos eletrônicos como sensores e atuadores sofreram uma redução de preço considerável. Além disso, surgiram plataformas como o Arduino e o Raspberri Pi, conhecidas como plataformas DIY (expressão que vem do inglês que significa: *Do It Yourself*, ou "Faça Você Mesmo". Essas plataformas permitem a interligação de componentes que se comunicam ao mundo real de maneira simples e prática. Ambas possuem um microcontrolador, responsável por toda a parte de controle do sistema.

Com todas essas ferramentas a um preço acessível, tornou-se possível a criação de um sistema de automação residencial de baixo custo. Onde é possível utilizar estas ferramentas para segurança, conforto e economia do usuário.

Para a implementação de um sistema de automação residencial pode-se utilizar dispositivos para comunicação, sensores e dispositivos como microcontroladores,

controladores lógico programáveis (CLP) ou algum outro microprocessador. Neste projeto, em particular, é utilizada a plataforma Arduino e um smartphone para monitoramento e auxílio nas tarefas de controle. A comunicação entre o Arduino e o dispositivo móvel é feita através da tecnologia Bluetooth.

O Arduino é uma plataforma de hardware e software, com bibliotecas opensource, criado com o objetivo de permitir o desenvolvimento de controle de sistemas interativos de baixo custo. Tem sido utilizado nas mais diversas aplicações.

1.2 OBJETIVO

O presente trabalho tem como objetivo a implementação de um projeto de domótica em uma residência, utilizando a plataforma Arduino. Neste projeto será implementado controles de presença, temperatura e iluminação. Detectores de chuva e vazamento de gases serão instalados, para que seja possível monitorar e acompanhar o sistema. Para o monitoramento e auxílio nas tarefas de controle é utilizado um aplicativo em um dispositivo móvel. O aplicativo, que funcionará como a IHM do sistema, foi desenvolvido no *AppInventor*, ferramenta criada por alunos e professores do *Massachusetts Institute of Technology* (MIT).

Para validação dos resultados foi feita uma maquete em MDF (*Medium Density Fiberboard*) que representa uma residência. Nela foram instalados os diversos sensores e atuadores.

1.3 ORGANIZAÇÃO DO TRABALHO

O trabalho de conclusão de curso está dividido em quatro capítulos. O primeiro capítulo destina-se a parte introdutória. No segundo capítulo é feito todo o embasamento teórico do projeto, onde é abordado um pouco do surgimento da domótica, bem como é feita uma explanação sobre as principais ferramentas utilizadas no projeto. O terceiro capítulo destina-se a explanar um pouco sobre os componentes utilizados, bem como as técnicas utilizadas para o projeto de controle e monitoramento. É descrito também cada ambiente da residência e o aplicativo criado para o smartphone. No quarto capítulo, as conclusões e considerações finais são apresentadas, bem como

2 FUNDAMENTAÇÃO TEÓRICA

No decorrer deste Capítulo serão apresentados aspectos teóricos com ênfase no Arduino Mega2560, utilizado neste projeto. Também será apresentado o *AppInventor* (ambiente no qual foi criado o aplicativo para o sistema Android) e o *Bluetooth*, tecnologia de comunicação sem fio responsável pela interação do Arduino com o dispositivo móvel. Também será explanado um pouco da história da domótica e sua utilização nos dias atuais.

2.1 A DOMÓTICA

Dispositivos que funcionam com energia elétrica ou movidos a gás tornaram-se viáveis na década de 1900, com a introdução de energia elétrica, levando a introdução de aquecedores de água (1889), máquinas de lavar (1904), geladeiras e outros eletrodomésticos.

Existe uma grande variedade de protocolos em que uma "casa inteligente" pode ser distribuida. Cada linguagem fala com vários dispositivos conectados com objetivo de instrui-los para executar uma função. Alguns exemplos são o *European Installation Bus* (EIB), *Konnex* (KNX), *Building Automation and Control NETworks (BACNet)*, *Digital Addresable Lighting Interface* (DALI), *Powerline Communication Bus* (PLC/BUS), *Insteon* e *Zwave*. A maioria dos protocolos listados a seguir não são abertos. Todos têm uma API.

O transporte das informações tem envolvido conectividade direta de fios, tecnologia PLC e sem fio.

Será explanada a seguir um pouco sobre as duas tecnologias mais conhecidas utilizadas na automação residencial: A X-10 e a KNX.

2.1.1 X-10

Em 1975, surge a primeira tecnologia que seria usada na domótica, conhecida como X-10. O protocolo X-10, desenvolvido pela Pico Electronics of Glenrothes, na Escócia, utilizava a rede elétrica como canal de comunicação entre os diversos

dispositivos eletrônicos. Trata-se de uma tecnologia PLC (Power Line Carrier). A grande vantagem deste protocolo era que era possível o controle de dispositivos sem precisar modificar a infra-estrutura elétrica da residência. O X-10 é um protocolo aberto, assim sendo, qualquer fabricante pode incluir um módulo de comunicação X-10 nos seus produtos e oferecê-lo aos usuários.

Os pacotes transmitidos usando o protocolo X-10 consistiam de um código de quatro bits, seguido por outro código de unidade de quatro bits, finalmente seguido por um comando de quatro bits. Para a facilidade de configuração do sistema pelos usuários, o código-casa de quatro bits é selecionado como uma letra de A a P, enquanto o código de unidade de quatro bits é um número de 1 a 16.

Após o sistema ser instalado, cada dispositivo controlado é configurado para responder a um dos 256 endereços possíveis. Cada dispositivo reage aos comandos especificamente dirigidos a ele. Por exemplo, o protocolo pode transmitir uma mensagem que diz: "selecione A3", seguido por "ligar", que comanda o dispositivo endereçado por A3 a ser ligado. Várias unidades podem ser abordadas antes de dar o comando, permitindo que o envio de um único comando abordem várias unidades simultaneamente.

Não há restrições que se use mais de um "código-casa" dentro de uma única residência. Assim sendo, instalações que usam vários "códigos-casa" na mesma casa tem os dispositivos divididos em "zonas" separadas.

O protocolo X-10 é até hoje o mais utilizado no mundo, graças ao seu baixo custo, facilidade de uso e grande variedade de equipamentos. No mercado norte-americano esta tecnologia é bastante difundida, sendo comum encontra-la até em supermercados.

2.1.2 KNX

No Brasil, o KNX começou a ganhar força no final de 2010, com a forte crise europeia. Houve uma forte migração de europeus para o Brasil, inclusive de engenheiros. Os profissionais que trabalhavam com automação chegavam aqui, desenvolviam todo o projeto de automação predial em KNX, e só na hora de implantar descobriam que este protocolo era completamente desconhecido. Este foi o caso da SISINT, uma empresa portuguesa de engenharia, que chegou ao Brasil em 2011.

KNX é um protocolo aberto utilizado em âmbito mundial e direcionado a aplicações em automação residencial e predial. Quer se trate de uma residência, um condomínio residencial ou um complexo de edifícios de escritórios, a demanda por conforto e funcionalidade na gestão de sistemas de controle de ar condicionado, de iluminação e segurança é crescente e caminha lado a lado com a conscientização pelo uso eficiente da energia. O protocolo KNX surgiu com a meta de atender a estas necessidades (LUCIANA MENDONÇA, 2012).

A associação KNX foi fundada em 1999, com a consequente criação do protocolo KNX, em decorrência da união de três países e seus respectivos protocolos: Bélgica, com o EIB; Holanda, com a *European Home Systems Association* (EHSA); e a França, com o *BatiBUS Club International* (BCI).

O protocolo KNX é baseado na transmissão de dados sobre topologia em barramento e em sua arquitetura de comunicação, que possibilita a passagem do cabo de dados em paralelo aos cabos da rede elétrica, ou seja, compartilhando da mesma infraestrutura física, reduzindo custos tanto de material quanto de mão de obra para a instalação. (LUCIANA MENDONÇA, 2012).

Uma das características do KNX é a facilidade de integrar produtos de diferentes fabricantes dentro de uma mesma rede. Esta é uma grande vantagem para o usuário final, que, ao instalar um sistema de automação residencial em sua casa, não fica refém de um fornecedor específico. Empresas como *Siemens*, *Bosh*, *Schneider*, *Asea Brown Boveri* (ABB), *Toshiba*, *Rockwell* e Eberle já possibilitam a utilização de KNX nos seus equipamentos.

Com relação ao meio de comunicação, há quatro possibilidades:

- Um cabo único correndo todo o sistema sem qualquer problema de interferência;

- Sistema sem fio via rádio;

- Sistema de transmissão por rede elétrica;

- Transmissão via internet ou rede de computadores.

O KNX tem uma Topologia que suporta até 65.500 dispositivos divididos entre sensor e atuador. A linha vermelha é de alimentação de atuadores de maior carga e a verde alimenta sensores e envia dados. (KNX DO BRASIL, 2016).

2.1.3 SITUAÇÃO ATUAL DA DOMÓTICA

Em 2012, nos Estados Unidos, de acordo com a ABI (*Allied Business Intelligence*) Research, 1,5 milhão de sistemas de automação residencial foram instalados. De acordo com Li et. al. (2016), existem três gerações da domótica:

- Primeira geração: tecnologia sem fio com servidor proxy, como a *Zigbee*;

- Segunda geração: inteligência artificial controlando dispositivos elétricos, como por exemplo a *Amazon Echo*.

- Terceira geração: A geração do "amigo-robô", que interage com os seres humanos, tendo como exemplos a *Roomia* e o *Robot Rovio.*

A internet banda larga concedeu ao usuário o monitoramento e controle de sua residência de qualquer lugar que disponha de internet. Com o intenso avanço tecnológico, intensificado no século XXI, no qual um único aparelho, como o smartphone, pode incorporar inúmero serviços (telefonia, internet, gps), tornou-se ainda mais aplicável sistemas de automação residencial. Entretanto, apesar do cenário tecnológico bastante favorável para a domótica, com as novas tecnologias sendo incorporadas pelo mercado brasileiro, o mercado de construção civil ainda não absorveu essa tendência. Atualmente, os nossos veículos possuem mais tecnologia embarcada do que nossas próprias residências, mesmo considerando que estas tenham um preço bem mais elevado.

Em 2000, foi fundada no Brasil uma associação chamada Associação Brasileira de Automação Residencial (AURESIDE), que tem por objetivo difundir tecnologias, treinar e formar profissionais, fomentar no mercado a utilização de Automação Residencial e outros objetivos desta natureza.

Atualmente, os projetos de automação são feitas em cima das principais tecnologias dedicadas a automação residencial: X-10 e KNX. Atualmente há projetos de domótica desenvolvidos com Arduino, porém, por pequenas empresas que focam no baixo custo como grande vantagem para o consumidor final.

2.2 ARDUINO

O Arduino foi criado no norte da Itália por cinco pesquisadores, são eles: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, e David Mellis. Sendo assim, o objetivo desses pesquisadores era criar um dispositivo que fosse fácil de usar,

fácil de programar e o mais importante de tudo, que fosse barato, fazendo dele uma plataforma acessível para todos e adaptável para qualquer tipo de projeto por ser Open-Source.

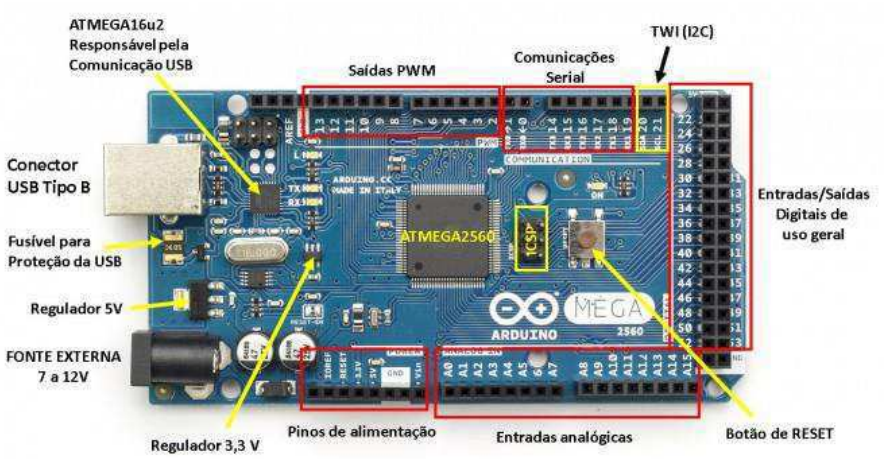
A plataforma Arduino surgiu no ano de 2005, na Itália, com o docente Massimo Banzi, que queria que seus alunos de Design utilizasse ferramentas de eletrônica e computação para criar seus próprios protótipos. Porém, com o que se tinha na época, essa não era uma tarefa fácil. As plataformas que existiam na época eram consideradas complexas para pessoas que não eram da área de eletrônica ou computação. Além do mais, as que existiam eram bastante caras. Daí surgiu a necessidade de criação de uma plataforma que qualquer pessoa com quase nenhum conhecimento na área pudesse manusear e criar seus próprios projetos. O objetivo dos criadores do Arduino era criar um dispositivo que fosse fácil de usar, fácil de programar e barato, fazendo dele uma plataforma acessível para qualquer tipo de pessoa. Sua primeira versão era vendida a trinta dólares, valor bem razoável quando comparado as ferramentas da época que tinham o mesmo propósito.

O Arduino é uma plataforma de computação embarcada, ou seja, um sistema que, através de hardware e software consegue interagir com o ambiente. O mesmo pode ser utilizado para desenvolver objetos interativos independentes, ou pode ser conectado a um computador, a uma rede, ou até mesmo à Internet para recuperar e enviar dados do Arduino e atuar sobre eles. Em outras palavras, ele pode enviar um conjunto de dados recebidos de alguns sensores para um site, dados estes que poderão, assim, ser exibidos na forma de um gráfico. (MCROBERTS, 2011).

2.2.1 CARACTERÍSTICAS DO HARDWARE

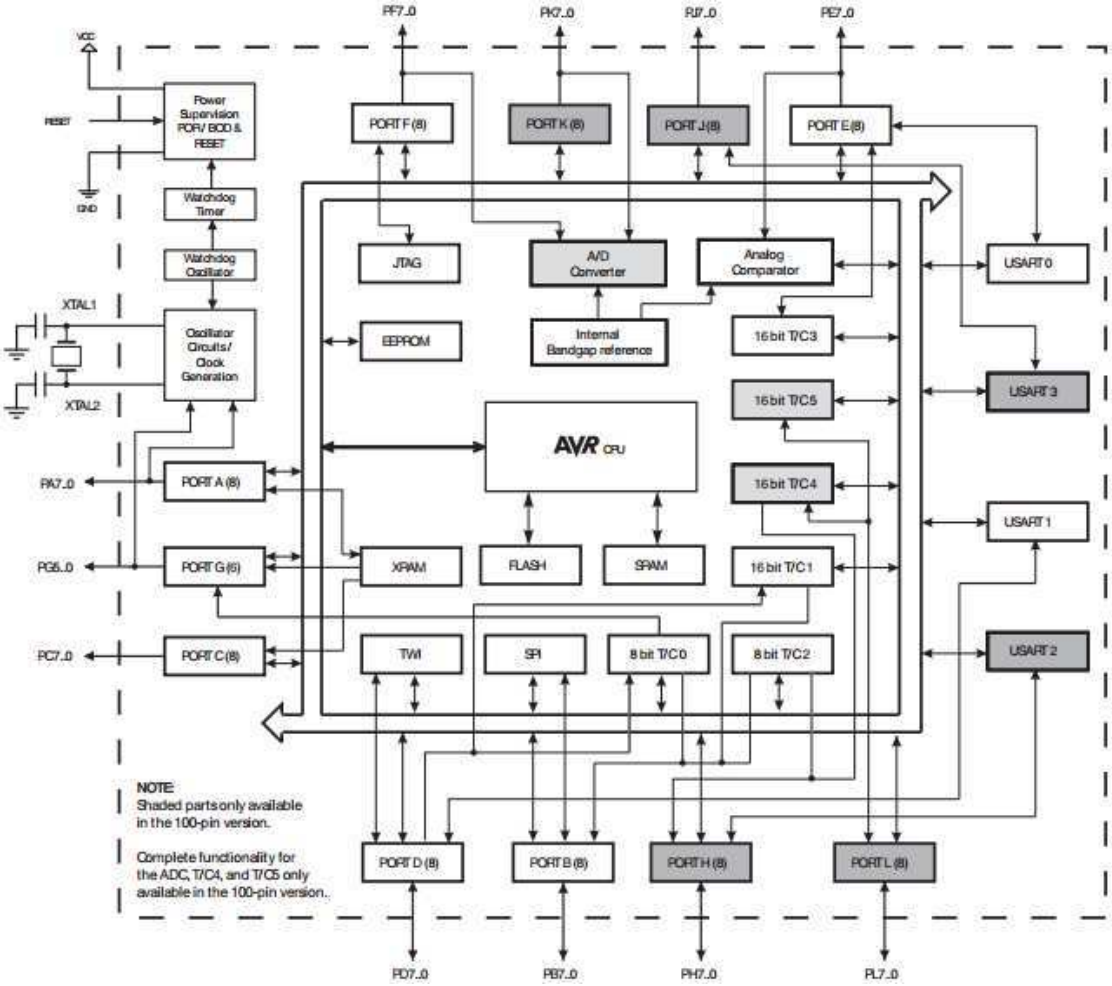
O Arduino Mega2560, mostrado na Figura 1, está baseado no microcontrolador ATMega2560. Há 54 pinos de entradas/saídas digitais (das quais 14 são saídas com *Pulse Width Modulation* - PWM), 16 entradas analógicas, quatro UART (*Universal asynchronous receiver/transmitter*), um cristral oscilador de 16MHz, conexão USB (*Universal Serial Bus*), conector de alimentação, conector ICSP (*In-Circuit Serial Programming*) e um botão de reset. Na Figura 2 é mostrado o diagrama de blocos do microprocessador ATMega2560.

Figura 1 - Arduino Mega2560 e um resumo de recursos.



Fonte: http://blog.arduino.cc/2011/01/05/nice-drawings-of-the-arduino-uno-and-mega-2560/

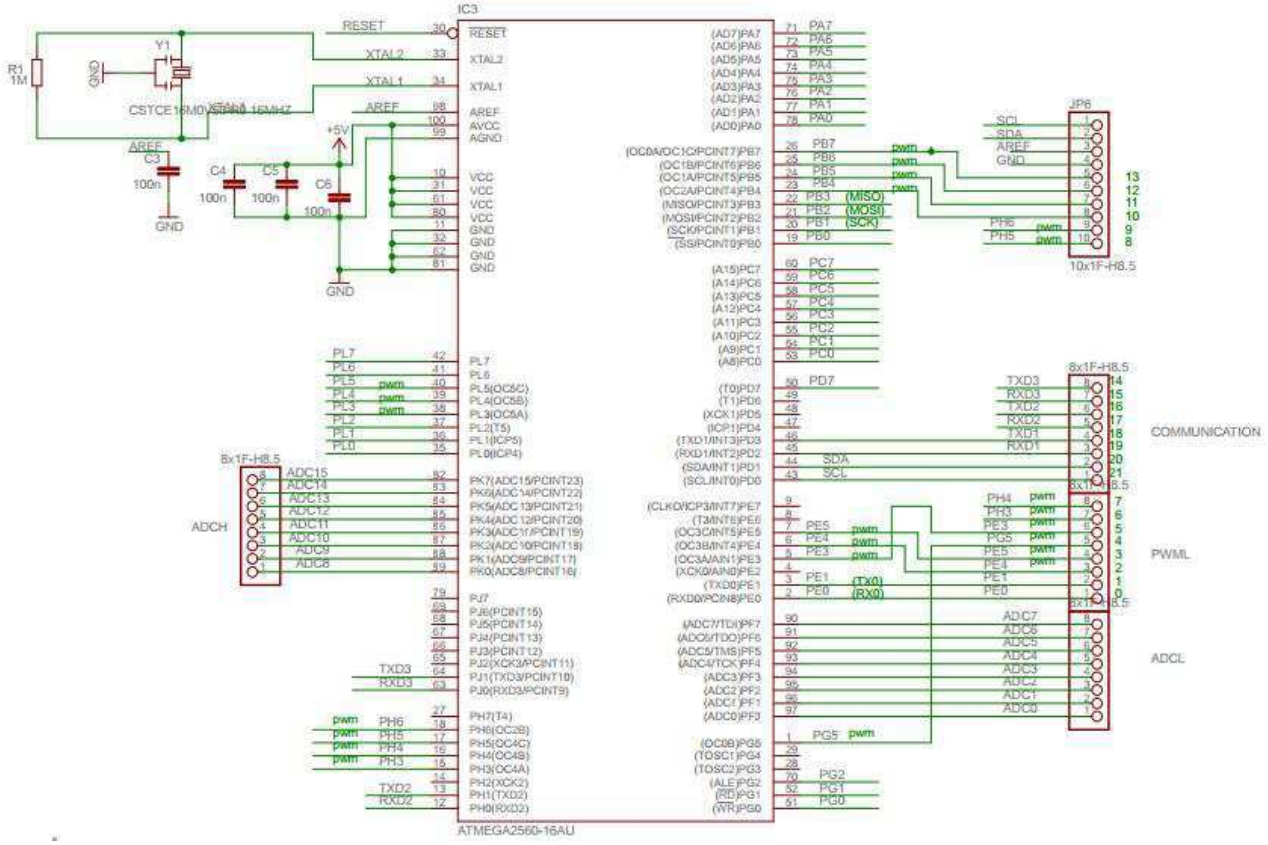
Figura 2 – Diagrama de blocos do microprocessador ATMega2560.



Fonte: Datasheet do microprocessador ATMega2560.

Como se pode observar na Figura 2, o microprocessador possui um conversor A/D. As portas F e K (ambas de 0 a 7) servem como entradas analógicas para o conversor A/D, comunicando-se com as portas quinze portas analógicas da placa. Na Figura 3, é possível observar a conexão destas portas com as do Arduino Mega. Se o conversor A/D não estiver sendo utilizado, as portas F e K funcionam como portas I/O de 8 bits bi-direcional.

Figura 3 – Esquemático do microprocessador ATMega2560 na placa Arduino Mega 2560.



Fonte: Site oficial do Arduino.

O conversor do ATMega2560 é de 10bits, ou seja, pode-se representar 2^{10} níveis de tensão distintos, entre 0 e 5V. Deste modo, a resolução do conversor pode ser expressa por:

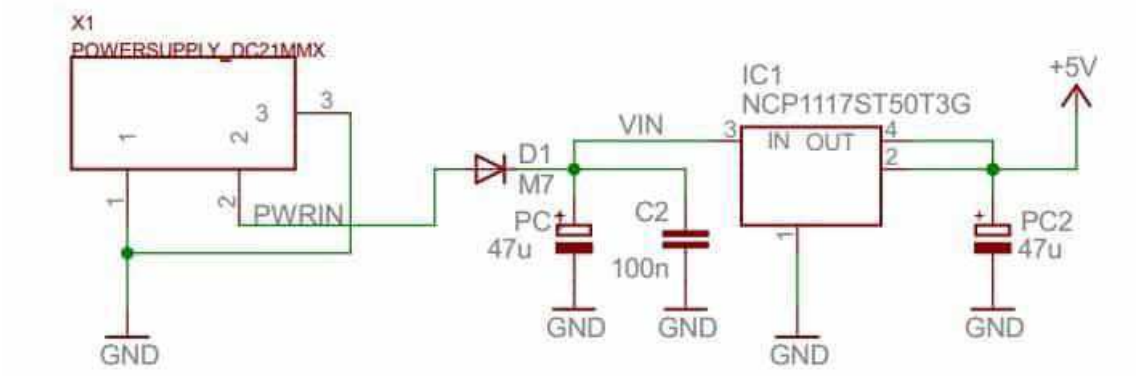
$$resolução = \frac{Vref}{2^n}$$

Na configuração default, sua faixa de valores é entre 0 e 5V. Como tem-se 10 bits, têm-se 1024 níveis de tensão distintos. Ou seja, é detectado variações de 5mV. Este valor pode ser modificado utilizando o pino AREF (modo EXTERNAL), que muda o valor de referência do pino de acordo com o valor aplicado no mesmo (pode-se aplicar

valores entre 0 e 5V). Também há os modos INTERNALL1V1 (referência de 1,1V) e o INTERNAL2V56 (referência de 5,6V).

A alimentação pode ser feita tanto pelo cabo USB como por uma fonte externa. A placa consegue operar com valores entre 6 e 20 volts. Com menos de 7V, o pino de 5V pode fornecer menos que os 5V. Se alimentarmos com mais de 12V, o regulador de tensão pode superaquecer. O aconselhável seria entre 7V e 12V. O circuito regulador para entrada externa é exibido na Figura 4. É utilizado neste regulador de tensão o CI OnSemi NCP117.

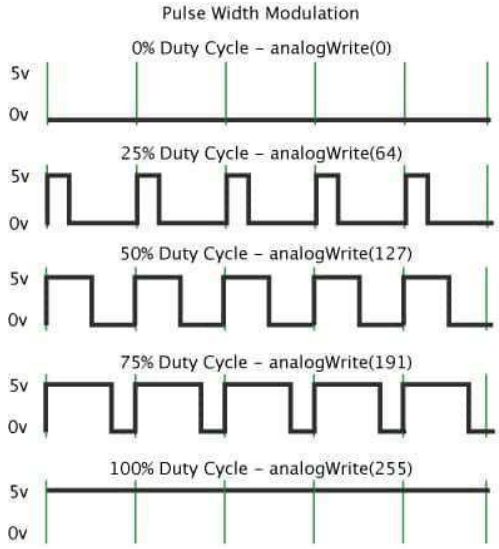
Figura 4 – Circuito regulador de tensão do Arduino.



Fonte: Site oficial do Arduino.

A placa Arduino Mega possui quinze portas que podem ser utilizadas como PWM (portas de 2 a 13 e 44 a 46), fornecendo saídas de 8bits de resolução utilizando a função analogWrite(). O PWM é uma técnica utilizada por sistemas digitais para variação do valor médio de uma onda periódica. Basicamente, mantem-se fixa a frequência de uma onda quadrada, variando o tempo em que ela fica em nível lógico alto. Esse tempo em que ela fica em nível lógico alto é chamado de *duty cicle*. Na figura 5, é exemplificado algumas modulações PWM:

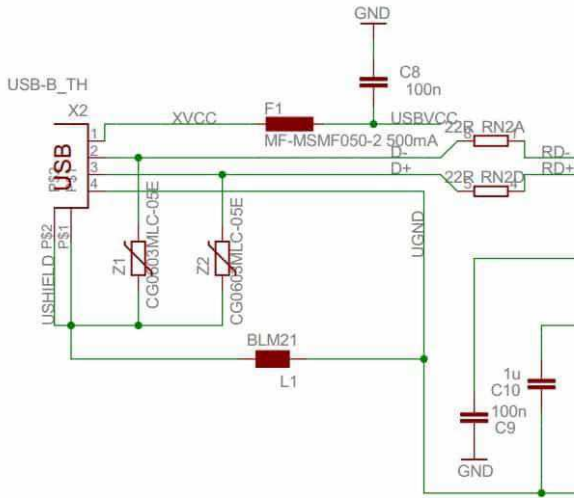
Figura 5 – Exemplos de modulação PWM



Fonte: Site oficial do Arduino.

Há quatro pinos de alimentação. O 5V que fornece cinco volts através do regulador da placa ou da porta USB. O 3,3V que é gerado pelo FTDI (*Future Technology Devices International*) chip da placa. Fornece uma corrente máxima de 50mA. O VIN que fornece a tensão de alimentação da placa quando utilizamos uma fonte externa e os pinos GND (terra). O Arduino possui um circuito que protege as portas USB de sobrecorrentes e curtos circuitos, fornecendo assim uma proteção extra. Caso seja aplicados mais de 500mA na porta USB, a ligação é desconectada automaticamente até haver ausência de sobrecargas e/ou curtos. A seguir, na Figura 6, pode-se observar o circuito de proteção da porta USB.

Figura 6 – Circuito de proteção USB do Arduino Mega 2560.



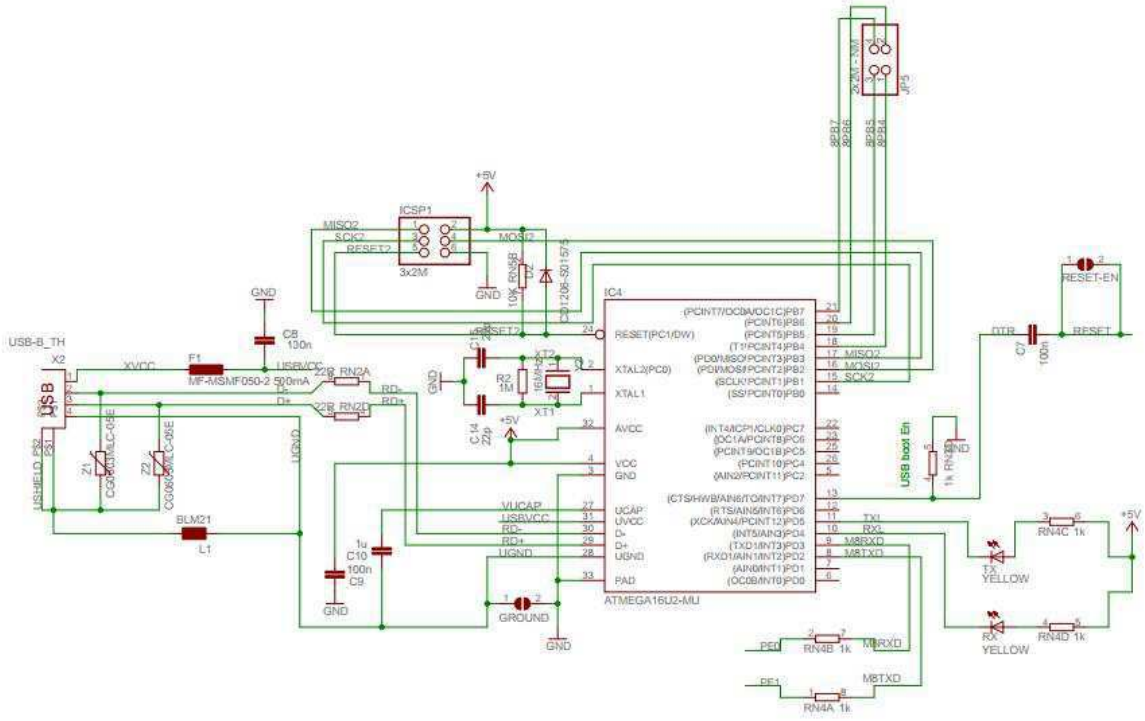
Fonte: Site oficial do Arduino.

Os dois varistores (Z1 e Z2) são responsável por evitar picos de SURGE e o surgimento de altas energias transientes. Os resistores RN2A e RN2D são responsáveis por limitar a corrente resultante de uma eventual descarga elétrica, protegendo os pinos do microcontrolador.

Há na placa um microcontrolador ATMEL ATMEGA16U2 que serve como interface USB para comunicação com o computador. Este microcontrolador é responsável como funciona a placa, possibilitando o upload do código binário gerado após a compilação do programa feita pelo usuário.

Neste microcontrolador, também estão ligados dois leds (TX e RX), controlados pelo software do microcontrolador, que indicam o envio e recebimento de dados. A conexão entre este microcontrolador e o ATMega2560 é feita pela serial desses microcontroladores. Na figura 7, pode-se observar o circuito de comunicação serial.

Figura 7 – Circuito de comunicação serial.



Fonte: Site oficial do Arduino.

Há vários modelos de Arduino no mercado. Uma das diferenças entre eles é com relação a memória. No microprocessador ATMega2560 temos a memória Flash que é utilizada para o armazenar o programa que é carregado no microcontrolador bem como para armazenar o bootloader, responsável pela inicialização do chip. A memória SRAM (*Static Random Access Memory*) é utilizada para armazenar as variáveis de

processamento. Quando o Arduino é desenergizado, estas informações deixam de existir, pois a SRAM é uma memória volátil. Já a memória EEPROM (*Electrically Erasable Programmable Read-Only Memory*) é responsável pelo armazenamento de dados de configuração e constantes. Por ser uma memória não-volátil, seus dados continuam armazenados mesmo com a desenergização. O microprocessador possui 256Kb de memória flash, 8Kb de SRAM e 4Kb de EEPROM.

O Arduino Mega possui 54 pinos digitais que podem ser utilizados como entradas ou saídas, operando a 5 volts. Cada pino pode fornecer ou receber até no máximo 40mA. Possuem um resistor de *pull-up* interno de 20-50kΩ.

Há 16 entradas analógicas na placa. Na condição default, o Arduino Mega mede entre 0 e 5V. Como ele oferece um conversor analógico/digital A/D de 10 bits, temos 1024 diferentes níveis de tensão nos pinos analógicos, detectando assim variações de até 5mV.

Placas externas, conhecidas como *shields*, podem ser acopladas ao Arduino, expandindo assim suas funcionalidades. Há *shields* para *Ethernet*, *Wi-fi*, *Xbee* ou para controlar sensores e atuadores, como a *Motor Shield*, que facilita o controle de motores.

2.2.2 SOFTWARE DO ARDUINO

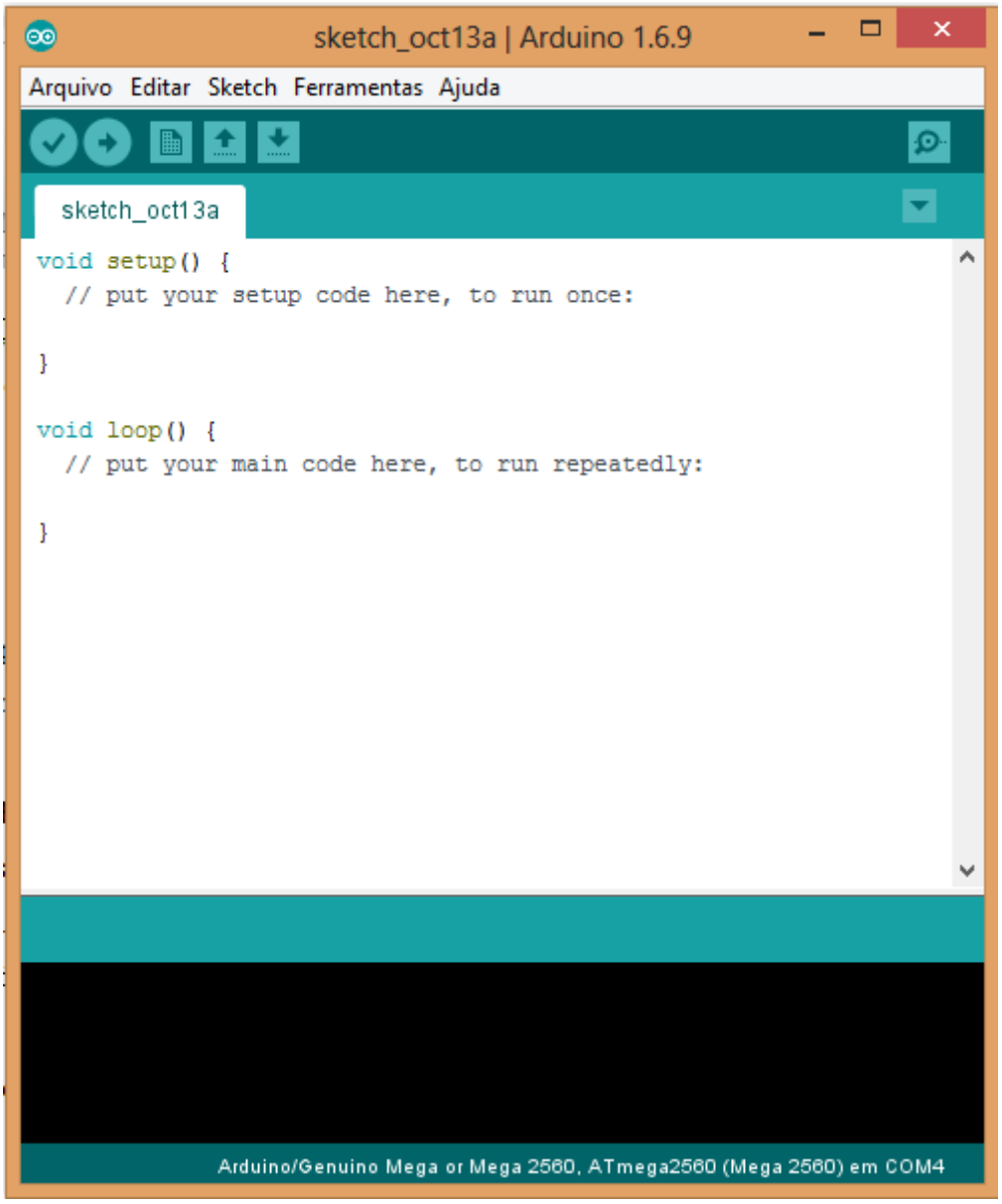
A programação do Arduino é feita através do seu próprio software. O ATMega 2560 possui um bootloader que permite o envio de códigos sem a necessidade de utilizar qualquer item externo. A comunicação é feita através do protocolo original.

O ambiente de desenvolvimento Arduino, o qual é mostrado na Figura 8, contém um editor de texto para escrever o código que permite opções de recortar/colar e localizar/substituir texto, uma área de alertas e possíveis comunicações de erros e vários menus. No canto inferior direito nos é informado a versão do Arduino bem como a porta de comunicação que está sendo utilizada. Através do software é possível criarmos novos programas (que são chamados de *sketches*) e passa-los para o Arduino. As barras de ferramentas nos permitem importar bibliotecas, fazer upload de programas, utilizar programas-exemplo, acessar toda a parte de configuração, menu de ajuda, ferramentas de edição e acessar o Monitor Serial. Os *sketches* são salvos com a extensão .ino. O software se encontra na versão 1.6.9.

O Monitor Serial nos permite que dados sejam enviados e recebidos com muita facilidade. Ao ocorrer envio ou recebimento de dados (que são transmitidos através do

chip ATMega16U2 e a porta serial), os LED (*Light-emitting diode*) referentes a Rx e Tx acendem. A biblioteca *SoftwareSerial* permite que você use outros pinos digitais para enviar e receber dados. O ATMega 2560 suporta comunicação SPI (*Serial Pheripheral Interface*) e TWI (*Two Wire Interface*). Para facilitar o uso da TWI, o Software possui a biblioteca *Wire*.

Figura 8 - Arduino software



Fonte: Elaborado pelo autor.

2.3 APP INVENTOR

O App Inventor é uma ferramenta que foi desenvolvida pela Google e atualmente é mantida pelo MIT que possibilita a criação de aplicativos para smartphones com sistema operacional Android, sem que seja necessário conhecimentos avançados em programação.

O grande diferencial do *App Inventor* é conseguir criar seu aplicativo fugindo das linhas de programação habituais. A programação ocorre de forma bastante simples e intuitiva graças ao recurso *drag and drop* (arrastar e soltar). A ferramenta permite desde manuseio de opções gráficas (botões, *labels*, imagens, planos de fundo) como a possibilidade de utilizar hardwares do próprio smartphone, como acelerômetro, microfone, câmera, GPS, *Bluetooth* e *Wi-fi.*

Segundo Wolber et al. (2011) o objetivo do usuário é unir esses blocos em quebra-cabeças, de modo a ir construindo sua aplicação. Essa característica de unir é bem particular, pois a ferramenta não permite que você una blocos que são "incompatíveis", diminuindo a chance de ocorrência de erros.

Por mais simples que o *App Inventor* possa parecer, ele ainda exige conhecimentos básicos de programação. Dependendo do grau de conhecimento na área do usuário, é possível criar aplicações com um certo grau de complexidade.

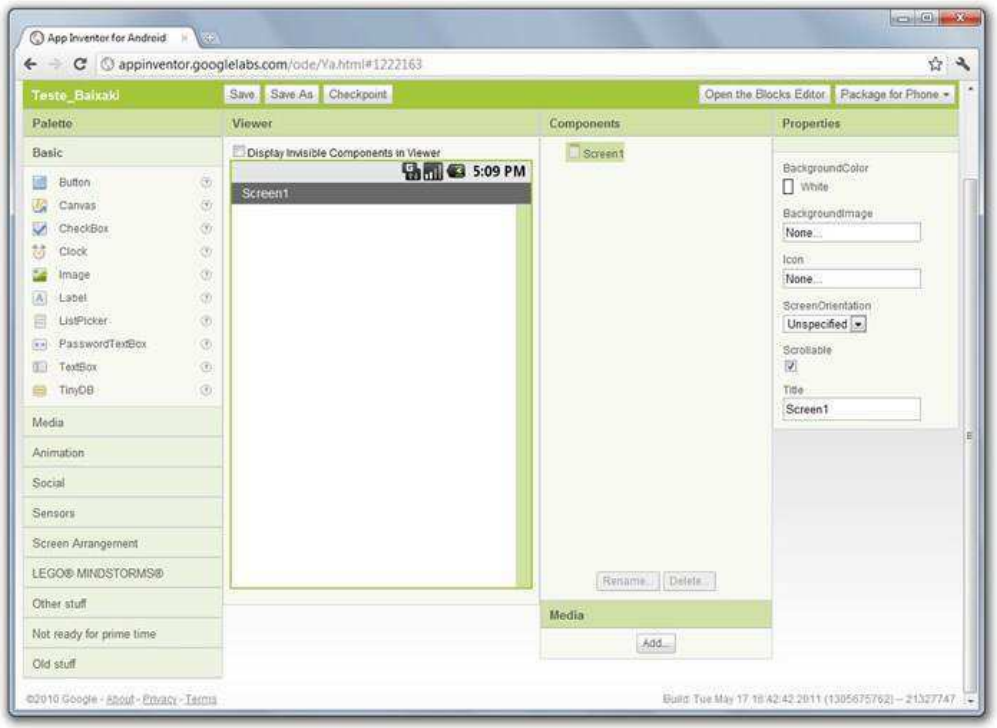
A primeira versão *trial* do App Inventor foi lançada em julho de 2010, e só era possível o acesso através de convite. No fim do mesmo ano, foi lançado a versão pública. Inicialmente, a plataforma era mantida pela *Google*, porém, ela foi desativada em 2011. O projeto foi salvo pelo MIT, que recebeu uma doação da Google para ajudar o MIT *Center for Mobile Learning*, um centro de aprendizado para desenvolvimento em dispositivos móveis.

Não é necessário instalar nada em seu computado para fazer uso da ferramenta. A aplicação funciona diretamente pelo navegador. A única coisa necessária é ter uma conta na Google. Todos os seus projetos são guardados no que é conhecido por armazenamento em nuvem, e podem ser acessados de qualquer computador em qualquer lugar, desde que este computador possua um navegador e acesso a internet.

O professor e pesquisador Eduardo Valle da Faculdade de Engenharia Elétrica e de Computação (FEEC) da Unicamp, completou no ano de 2016 a tradução da ferramenta para o português, que já se encontra disponível no site oficial.

A interface do App Inventor pode ser visualizada na Figura 9. Com uma interface simples e bastante fácil de usar, ela é organizada de modo muito similar às IDEs comuns, com algumas diferenças que visam a maior didática e facilidade de manuseio da plataforma, permitindo principalmente o acesso da ferramenta aos jovens que não são da área de programação, fazendo com que novas ideias consigam ser colocadas em prática.

Figura 9 - Tela inicial dos projetos do App Inventor.



Fonte: Elaborado pelo autor.

O aplicativo é composto por duas seções: o *App Inventor Designer* e o *App Inventor Blocks Editor.*

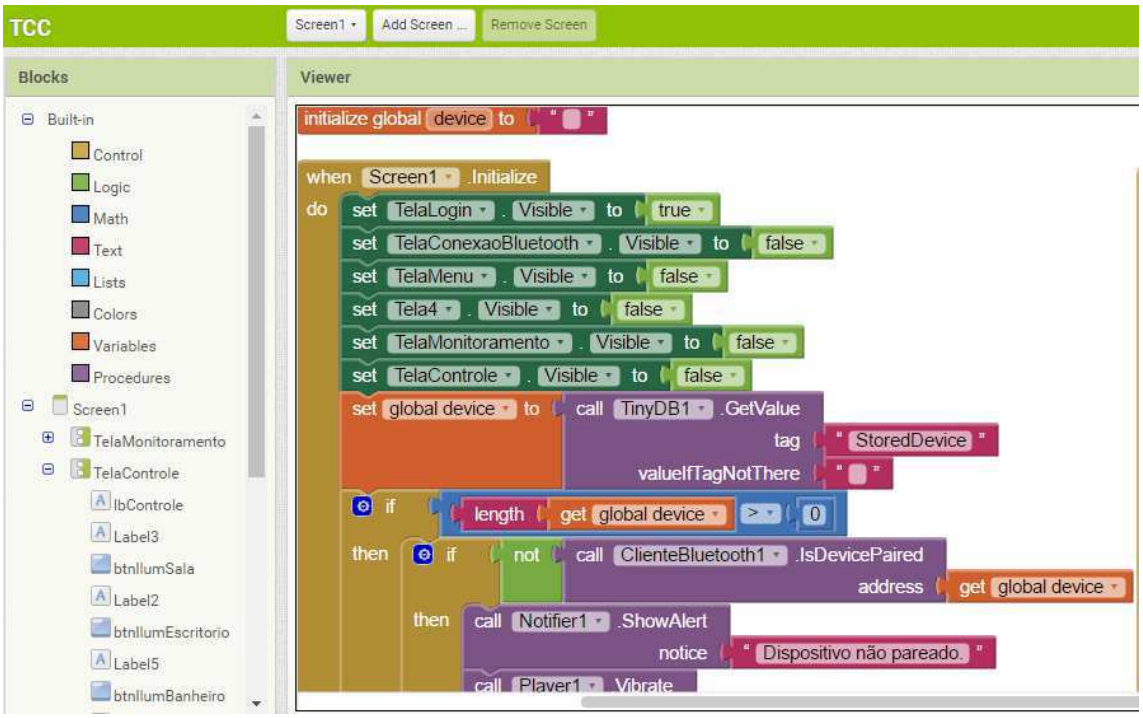
No *App Inventor Designer*, é basicamente onde você "desenha" o seu aplicativo. Utilizando a técnica de arrastar-soltar, conseguimos adicionar ao nosso aplicativo botões, imagens, *checkboxes*, textos e inúmeros outros recursos que nos ajudará a definir um layout para a aplicação a ser desenvolvida. Na aba *Palette*, também podemos encontrar ferramentas que dá suporte para utilizar os hardwares já existentes no próprio smartphone, como GPS, acelerômetro, microfone e *bluetooth*.

Na coluna *Viewer*, é onde o usuário pode organizar cada um dos objetos que foram selecionados na aba *Palette*. O *Viewer* simula uma tela de celular, para passar a ideia de como ficará a aplicação.

A terceira coluna é chamada de *Components*. Lá podemos encontrar todos os itens que já foram adicionados ao novo aplicativo em desenvolvimento. Nesta coluna, podemos renomear os componentes para uma maior facilidade na hora de programar, bem como adicionar sons, fotos e vídeos da sua máquina para o aplicativo. Ao selecionar um item nesta coluna, seus detalhes aparecerão automaticamente na coluna *Properties*. Lá é possível modificar cada item que já foi inserido na sua aplicação. Cada modificação que é feita no objeto é mostrada instantaneamente na coluna *Viewer*.

Ao terminar a parte de "montagem" do seu novo aplicativo, é necessário atribuir funções a cada objeto inserido. Isto é feito na seção *App Inventor Blocks Editor*, ilustrada na Figura 10.

Figura 10 - Seção App Inventor Blocks Editor



Fonte: Elaborada pelo autor.

O *App Inventor Blocks Editor* é onde a programação em si acontece. Atribuir ações a cada objeto que foi inserido na seção *App Inventor Designer*. A interface de códigos é substituída por blocos que se encaixam, criando as ações em cada objeto. Esta parte da plataforma é dividida em duas partes: *Blocks* e *Viewer*.

Na seção *Blocks* é encontrado todos os objetos que foram inseridos em seu programa, bem como passos de execução, na seção *Built-in*. Blocos *built-in* estão disponíveis independentemente de quais componentes estão em seu projeto. Há blocos

de controle, lógicos, matemáticos, de texto, listas, cores, variáveis e de procedimento. Além destes blocos, cada componente presente no projeto tem seu próprio conjunto de blocos específicos para seus eventos, métodos e propriedades.

Na seção *Viewer* é onde os comandos são estruturados como peças de um quebra-cabeça. Apenas funções compatíveis se encaixam. Quando há a união destes blocos tem-se uma ação completa.

O *App Inventor* permite que você visualize sua aplicação mesmo de dois modos: criando uma máquina virtual no seu computador ou utilizando seu próprio smartphone. No primeiro caso, é necessário instalar uma máquina virtual baseado em Java, permitindo assim a criação de um sistema Android genérico. Com isso, tem-se uma tela de celular no monitor do seu computador para testar os aplicativos.

Outra opção é utilizando o próprio smartphone. É necessário instalar o aplicativo MIT AI2 Companion no seu dispositivo móvel. Conectado a internet, ele consegue simular o aplicativo através de um código de seis caracteres gerado na plataforma, ou através da leitura de um QR Code. As alterações feitas no Designer e no Blocks aparecem em tempo real. Com essa capacidade a vida do programador é bastante facilitada, tendo em vista que é possível checar os resultados das ações de imediato.

2.4 BLUETOOTH

Será utilizada a tecnologia Bluetooth para fazer a comunicação entre a plataforma Arduino e o dispositivo móvel, tendo em vista sua facilidade de uso e baixo custo de compra de componentes.

Bluetooth é uma especificação industrial para a comunicação em curta distância de redes sem fio com um baixo custo e alta operabilidade (KOBAYASHI, 2004).

O Bluetooth se tornou um grande conhecido dos usuários pela facilidade de transferir dados através de dispositivos. Antes da popularização da internet móvel para celulares, a maneira mais simples e rápida de transferir arquivos de um dispositivo móvel para outro era por essa tecnologia. Também é bastante utilizado em fones de ouvido e impressoras.

Tudo começou em 1998, quando cinco empresas (a Ericsson, a Nokia, a IBM, a Intel e a Toshiba) formaram um consórcio chamado Bluetooth SIG (*Special Interest Group*) com o objetivo de expandir e promover o conceito Bluetooth e estabelecer um

novo padrão industrial. Atualmente, já fazem parte deste consórcio empresas como 3Com, Compaq, Dell, HP (Hewlett-Packard), Lucent, Motorola, NTT (*Nippon Telegraph and Telephone Corporation*) DoCoMo, Philips, Samsung, *Siemens* e *Texas* (KOBAYASHI, 2004).

O protocolo opera na faixa de licença-livre ISM (Industrial, Scientific, and Medical) em 2,45 gigahertz. Alcança velocidades de até 723,1 kbit/s.

As redes bluetooth usam a topologia piconet. Cada piconet contém no mínimo dois e no máximo oito dispositivos em rede.

Um aparelho dotado de tecnologia Bluetooth pode comunicar-se com um ou mais aparelhos simultaneamente. As piconets são formadas por um membro mestre e por diversos escravos, dos quais até sete podem ocupar o espaço de ativos, enquanto que os restantes são caracterizados como escravos estacionados (podendo chegar até 255). Cada um destes ganha um "endereço de estacionamento" de 8 bits.

Os dispositivos de Bluetooth possuem quatro estados de operação: Mestre, Escravo Ativo, Escravo Passivo e Em Espera. O Mestre controla a piconet. O Escravo ativo participa ativamente da piconet, monitorando ou participando. O Escravo Passivo faz parte da piconet, porém em um modo de baixa prioridade, monitorando a rede ocasionalmente. O estado Em Espera não está conectado a uma piconet, aguradando pedidos, ele não está sincronizado com a rede.

O fato da tecnologia Bluetooth não precisar de cabos e fios é uma grande vantagem. Os usuários não precisam de nenhum acessório adicional para poder utilizar a tecnologia dos seus celulares. Outra grande vantagem é o fato da tecnologia não requerer uma linha clara de visão entre os dispositivos, ou seja, os dispositivos não tem que estar voltados um para o outro. É possível transferir dados via Bluetooth de um aparelho para outro até em locais separados por obstáculos. A energia utilizada para esta tecnologia é muito baixa, tornando-a uma ferramenta ideal para dispositivos eletrônicos. Porém, a grande vantagem da tecnologia Bluetooth se dá pela sua facilidade de uso. Com apenas alguns cliques no seu smartphone ou computador é possível ativar o seu Bluetooth e começar a trocar dados com outro usuário. É uma tecnologia livre e não requer nenhum taxas a serem pagas.

Há algumas desvantagens no Bluetooth. A taxa de transferência é relativamente baixa quando comparado a uma LAN (*Local Area Network*) *wireless*. Se seu objetivo é transferir grandes quantidades de dados, esta tecnologia não é a ideal. A necessidade de proximidade com o outro usuário também é um obstáculo em alguns casos.

Embora a tecnologia Bluetooth seja sem fio, ela tem um propósito bem diferente das tecnologias wireless utilizadas atualmente.

3 MATERIAIS E MÉTODOS

A seguir será apresentado toda a parte de aquisição de material, montagem do sistema na maquete, a parte de programação em Arduino e a confecção do aplicativo para Android desenvolvido no App Inventor.

3.1 DESCRIÇÃO DO SISTEMA DE CONTROLE

O projeto tem como objetivo aplicar técnicas de domótica em uma maquete que simula um ambiente residencial. Implementou-se vários sistemas de controle na residência, foram eles:

- Controle de iluminação de um ambiente (acionado via smartphone);
- Automatização da iluminação (banheiro);
- Controle de temperatura (quarto);
- Controle de segurança (cofre);
- Monitoramento de vazamento de gases (cozinha);
- Monitoramento de detecção de chuvas.

Foram utilizados sensores de pequeno porte, para adequar-se a maquete. Estes sensores poderiam, em sua maioria, serem utilizados em um ambiente residencial sem nenhum problema.

A comunicação entre o dispositivo móvel e o Arduino é feita através de bluetooth. Através dessa comunicação é possível o acionamento de lâmpadas e bem como o monitoramento do sistema.

3.2 COMPONENTES UTILIZADOS

Foram utilizados diversos componentes no desenvolvimento do projeto, tais como: sensores, relés, módulo bluetooth, motor e Arduino. A seguir será mostrado cada elemento utilizado, uma lista de material com os custos de cada componente e uma breve abordagem sobre cada item.

3.2.1 SENSORES DE PRESENÇA INFRAVERMELHO

O sensor de presença, mostrando na Figura 11, é bastante utilizado em estabelecimentos residenciais e comerciais. Sua principal finalidade é a economia de energia, visto que evita que as luzes sejam esquecidas ligadas. Seu funcionamento é bastante simples: ele emite um sinal infravermelho que se espalha, sendo refletido pelos objetos e retornando ao redoma que agrupa os feixes em um detector infravermelho.

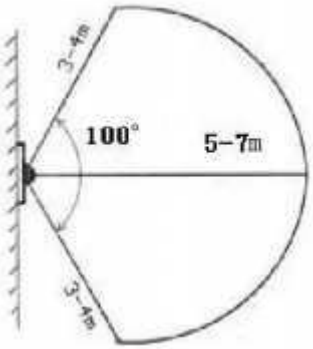
O módulo utilizado neste projeto foi o PIR DYP-ME003 da Elec Freaks, que possui um HC—SR01 *Body Sensor Module*. Consegue operar entre -15°C e 70°C, com uma detecção de distância ajustável entre 3 e 7 metros. Sua tensão de operação é entre 4,5V e 20V. Tem como 3,3V nível alto e 0V como nível baixo. Quando inativo, consome uma corrente menor que 50μA. Possui um ajuste de retardado no tempo em que o sinal fica no nível lógico alto variando entre 5s e 300s. O ajuste de distância e tempo são regulados por dois potenciômetros existentes no módulo. O alcance do sensor é representado na Figura 12.

Seu modo de ligação é bastante simples. Há 3 terminais, VCC para alimentação, GND para o terra e um terceiro para emitir o sinal de saída.

Figura 11 - Sensor de presença infravermelho.

Fonte: Site de compras AliExpress.

Figura 12 – Alcance do sensor de presença infravermelho.



Fonte: Datasheet do módulo DYP-ME003 da Elecfreaks.

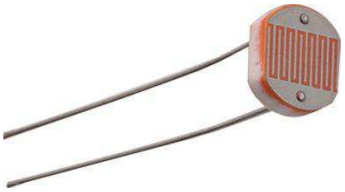
3.2.2 SENSOR LDR

O *Light Dependent Transistor* – LDR, mostrado na Figura 13, como seu próprio nome diz, é um resistor que varia o valor de sua resistência em função da luminosidade que incide sobre o sensor. Quanto maior a iluminação detectada pelo sensor, menor será sua resistência. É possível comprovar este fato utilizando um multímetro quando houver variância na iluminação que incide sobre o sensor.

Este eletrodo é feito de um composto químico semicondutor conhecido por sulfeto de cádmio. Quando os fótons incidem sobre ele, há liberação de elétrons nas moléculas. Se há mais elétrons disponíveis, a resistência elétrica diminui.

Na instalação do sensor, foi utilizado um resistor de 10kohms para servir como divisor de tensão. Com isso é possível calcular o valor da tensão de saída sobre o LDR.

Figura 13 - Sensor de luminosidade LDR.



Fonte: Site de compras AliExpress.

3.2.3 MÓDULO SENSOR DE GÁS INFLAMÁVEL E FUMAÇA MQ-2

O sensor MQ-2, mostrado na Figura 14, é capaz de detectar diversos tipos de gases inflamáveis, tais como: GLP, Metano, Propano, Butano, Hidrogênio e Gás Natural. Também é capaz de detectar fumaça. Tem uma detecção de concentração de GLP que varia entre 200 e 5000ppm. Sua tensão de operação é de 5V. O módulo utiliza o comparador duplo diferencial LM393. Quando a concentração de gás ultrapassa a pré-definida pelo potenciômetro, a saída digital D0 (que é ligada a um LED presente no módulo) fica em estado alto e, caso esteja abaixo do nível, encontra-se em estado baixo.

Sua temperatura de operação é entre -20°C e 50°C. O tempo de pré-aquecimento do sensor é de 24 horas.

Na nossa implementação, este sensor está conectado ao Arduino através de sua saída analógica A0, conectada a uma entrada analógica do Arduino, que possui um conversor A/D. O sensor foi testado e monitorado pelo Monitor Serial, acompanhando seus valores para definição do limite mais adequado.

Sua pinagem é composta por uma saída analógica, uma digital, VCC e GND. Possui também um LED indicador para tensão.

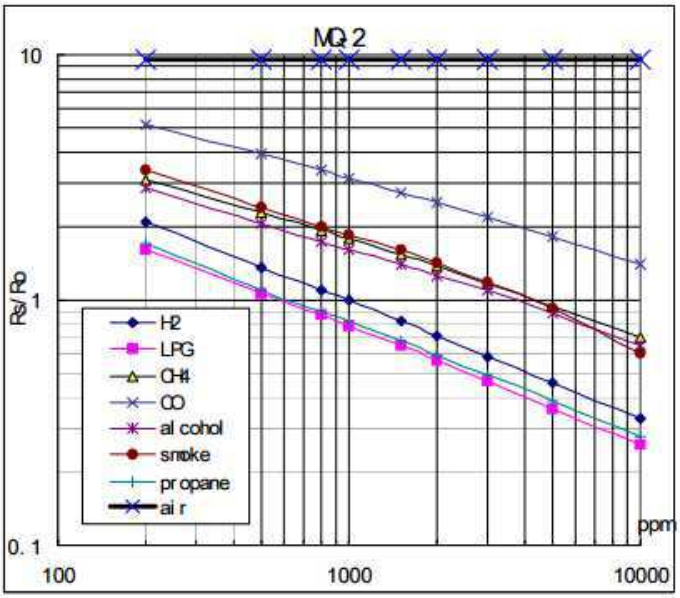
Figura 14 – Sensor MQ-2

Fonte: Site de compras Aliexpress.

Sua curva característica de sensibilidade pode ser observada na Figura 15. Para a calibração do dispositivo foi utilizado o código do Anexo A da SandBox Electronics. Basicamente o código traça curvas aproximadas de acordo com as curvas de sensibilidade da Figura 14. Após isso, através de algumas funções (explicitadas no código em Anexo) ele calcula a razão entre Rs e Ro. Com este valor em mãos, pode-se voltar ao gráfico e encontrar o valor ppm correspondente.

O limite inferior de explosividade é de 18000ppm. Após a calibração, encontrou-se o valor dado pelo conversor A/D referente à 2500ppm. Qualquer valor acima deste, será acionado o alarme da área da cozinha.

Figura 15 – Curva característica de sensibilidade do sensor MQ-2

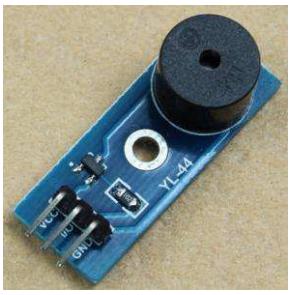


Fonte: Datasheet do sensor MQ-2.

3.2.4 MÓDULO BUZZER

O *buzzer*, mostrado na Figura 16, é um componente eletrônico composto por duas camadas de metal e uma camada de cristal piezoelétrico. Este componente consegue emitir um sinal sonoro ao ser energizado. O *buzzer* foi utilizado para simular um sinal sonoro de alarme. O *buzzer* opera em frequências de até 2kHz. É utilizada a função tone() do Arduino para emitir sinal sonoro na frequência desejada. Um buzzer está em um módulo YL-44, com 3 pinos: VCC, GND e I/O. Sua tensão de operação é de 5V. A ligação do outro *buzzer* que não está no módulo é um pouco distinta e será retratada adiante.

Figura 16 – Módulo Buzzer



Fonte: Site de compras AliExpress.

3.2.5 MICRO SERVO MOTOR

O Micro servo 9g Tower Pro SG90, mostrado na Figura 17, será o servo utilizado neste projeto. Ele possui uma tensão de operação entre 3V e 7,2V, com um ângulo de rotação de 180°. Possui uma velocidade de 0.12seg/60° trabalhando a 4,8V sem carga, com um torque de 1,2kg·cm. Sua temperatura de operação é entre -30°C e 60°C. Normalmente, estes servos são utilizados em aeromodelismo e projetos de robótica por seu peso reduzido (9 gramas) que não compromete o torque e o desempenho do motor. Neste projeto o servo será utilizado para fechar uma janela caso esteja chovendo.

Figura 17 – Micro Servo 9g

Fonte: Site de compras AliExpress.

3.2.6 SENSOR DE ÁGUA FUNDUINO

O sensor de água Funduino, mostrado na figura 18, é de alta sensibilidade. Aceita alimentação entre 3.3V e 5V, absorvendo menos de 20mA. Ele consegue detectar tanto gotas de água, bem como níveis de profundidade. Sinal de saída variando entre 0V e 4.2V. Funciona com interface analógica.

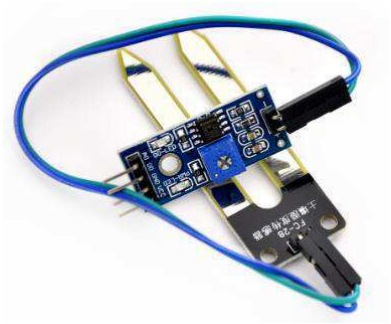
Figura 18 – Sensor de água Funduino

Fonte: Site de compras Aliexpress.

3.2.7 SENSOR DE UMIDADE FC-28

O sensor FC-28, mostrado na figura 19, tem como finalidade detectar as variações de umidade no solo. Utiliza, assim como o MQ-2, o comparador LM393. Sua tensão de funcionamento varia entre 3.3V e 5V. Possui um potenciômetro no seu módulo que se torna nível alto quando o valor na porta analógica é maior que o valor setado pelo potenciômetro. A placa é revestida em ambos os lados com um tratamento de níquel contra oxidação, melhorando assim sua condutividade, desempenho e duração.

Figura 19 – Sensor de umidade FC-28

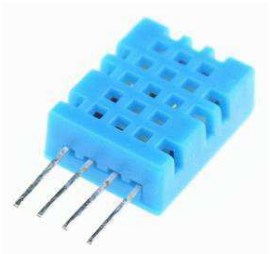


Fonte: Site de compras AliExpress.

3.2.8 SENSOR DE TEMPERATURA E UMIDADE – DHT11

O sensor DHT11, exibido na figura 20, é um sensor de temperatura e umidade que permite fazer leituras de temperaturas entre 0°C e 50°C e umidade entre 20% e 90%. O elemento sensor de temperatura é do tipo NTC e o sensor de umidade é do tipo HR202. Há um circuito interno que faz a leitura dos sensores e se comunica com o microcontrolador através de um sinal serial de uma via. Possui um tempo de resposta de aproximadamente dois segundos. Tensão de alimentação entre 3V e 5V.

Figura 20 – Sensor de temperatura e umidade DHT11



Fonte: Site de compras Aliexpress.

3.2.9 COOLER 12V

Cooler, palavra em inglês que significa “refrigerador”, é um componente muito utilizado principalmente para refrigerar componentes eletrônicos. Normalmente é encontrado acoplado com um dissipador. Neste projeto é utilizado um Cooler da HXS com tensão de alimentação de 12V, com uma velocidade máxima de 1800rpm. O cooler utilizado no projeto é mostrado na figura 21.

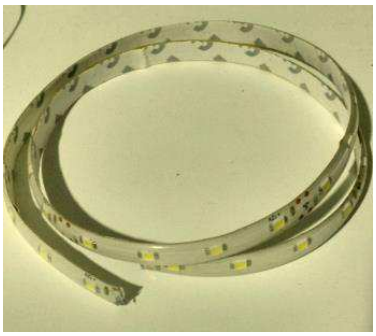
Figura 21 – Cooler

Fonte: Site de compras Aliexpress.

3.2.10 FITAS DE LED 12V

No nosso projeto, utilizamos fitas de LED de 12V, ilustradas na figura 22, para iluminarmos a sala e os banheiros. Alimentada por fontes de tensão de 12V.

Figura 22 – Fitas de LED

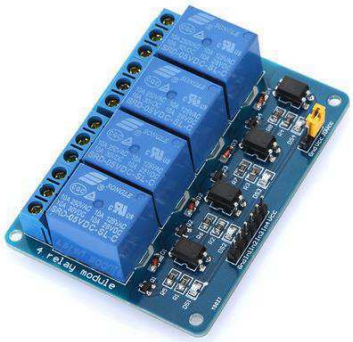


Fonte: Elaborado pelo autor.

3.2.11 MÓDULO RELÉ DE QUATRO CANAIS

É utilizado no acionamento das fitas de LED o módulo relé SRD-05VDC-SL-C de quatro canais, mostrado na figura 23. Permite controlar cargas até 220V, com tensão de operação de 5V. Consome uma corrente entre 15mA e 20mA. Trabalha com nível baixo ativo. Há LEDs indicadores de status para cada um dos canais. Possui seis pinos, um para o VCC, um para o GND e quatro para controle de cada um dos canais.

Figura 23 – Módulo Relé de quatro canais.



Fonte: Site de compras Aliexpress.

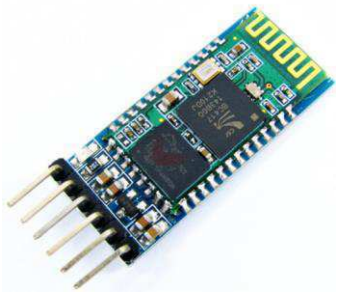
3.2.12 MÓDULO BLUETOOTH HC-05

O módulo HC-05, ilustrado na figura 24, oferece uma forma fácil e barata de comunicação com o Arduino, podendo funcionar tanto em modo Mestre como em Escravo.

O módulo pode ser alimentado com 3.3V ou 5V, graças ao regulador de tensão encontrado nele. Seu alcance é de até dez metros.

O HC-05 utiliza o firmware Linvor 1.8. Consome uma corrente de 35mA quando pareado, e de 8mA quando está apenas ligado. Funciona em temperaturas entre -40°C e 105°C.

Figura 24 - Módulo Bluetooth HC-05



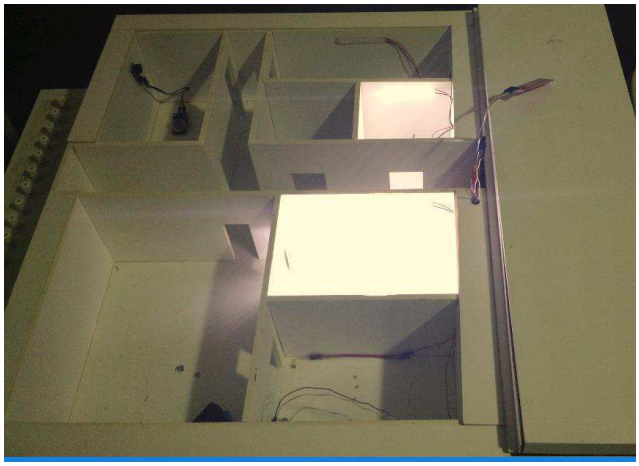
Fonte: Site de compras Aliexpress.

3.3 DESCRIÇÃO DOS AMBIENTES

Nesta seção será apresentada quais os locais onde cada sensor e atuador estão, bem como o sistema de controle implementado nos ambientes.

O ambiente é uma residência, dividida em uma sala, uma cozinha, um banheiro, um lavabo, dois quartos, um cofre e um escritório. A maquete é mostrada na Figura 25.

Figura 25 – Maquete em MDF.



Fonte: Elaborado pelo autor.

3.3.1 BANHEIRO

No banheiro é implementado um sistema de iluminação automática, ou seja, as luzes serão acesas quando a presença de alguém for detectada pelo sensor.

O sistema de controle é composto por um sensor de movimento que irá detectar a presença de alguma pessoa quando a mesma entrar no banheiro, enviando assim um sinal para o Arduino. Com essa informação, o Arduino envia um sinal para que seja acesa a iluminação do banheiro. A luz permanecerá ligada por um período T pré-determinado. Cada vez que um movimento é detectado, o temporizador responsável pela contagem desse período T é zerado.

O sensor utilizado é o HC-SR01 Body Module Sensor, que está acoplado no módulo PIR DYP-ME003. A iluminação é feita por fitas de LED. O sensor envia o sinal para o Arduino, que está conectado às fitas de LED através de um módulo relé. Ao enviar este sinal ao relé, os LEDs são acesos.

3.3.2 LAVABO

O lavabo é iluminado por uma fita de LED, que pode ser acionada pelo dispositivo móvel. Um sinal é enviado ao Arduino através do smartphone, e este sinal é repassado ao relé, energizando a fita de LED.

3.3.3 COFRE

No cofre é implementado um sistema de alarme.

O sistema para controle de segurança é composto basicamente por um módulo laser e um LDR.

O laser incide sobre o módulo, gerando um valor analógico bastante elevado. Caso essa luz pare de incidir sobre o LDR, sua resistência diminuirá, consequentemente, diminuindo o valor que é retornado ao Arduino pela sua saída analógica. Ao ser detectado essa diminuição significativa no sinal enviado, um buzzer é ativado como medida de segurança para informar uma ação indevida. Esta medida de segurança pode ser desativada através do smartphone, para entrada de pessoas autorizadas e/ou manutenção.

3.3.4 QUARTOS

No quarto, há um sistema de monitoramento de temperatura e umidade, utilizando o sensor DHT11. É possível acompanhar estes valores em tempo real pelo aplicativo no smartphone. Quando a temperatura passa dos 29°C, é enviado um sinal para o Arduino para o acionamento do Cooler, que tem por objetivo resfriar o ambiente. Este acionamento automático pode ser desativado pelo smartphone.

3.3.5 COZINHA

Vazamento de gases é um problema incômodo e perigoso. Detectar com rapidez o ocorrido é imprescindível, para evitar possíveis riscos de explosões. Na cozinha há a presença de um sistema de monitoramento de gases. O sensor MQ-2 é responsável pelo monitoramento. Caso o valor lido pelo sensor ultrapasse o valor pré-estabelecido, o sistema acionará o buzzer. O mesmo só parará de emitir sons quando o valor volte para abaixo do valor definido pelo sistema de controle.

3.3.6 SALA

Na sala há seis fitas de LED responsáveis pela iluminação do ambiente. As fitas de LED podem ser acionadas pelo smartphone. Pelo tamanho do cômodo, os componentes responsáveis pelo controle foram instalados lá (arduino e o módulo relé).

Há também um sensor de água que, ao detectar uma chuva, aciona um servo motor que irá fechar a janela da sala, impedindo que a água entre na residência.

3.3.7 ESCRITÓRIO

No escritório, assim como no lavabo, há uma fita de LED responsável pela iluminação, que também pode ser acionada pelo aplicativo no dispositivo móvel.

3.4 APLICATIVO SMART RESIDENCE

Foi criado um aplicativo para smartphone com sistema Android para monitorar e controlar os dispositivos eletrônicos que fazem parte do sistema de controle residencial, sendo a IHM (Interface Homem-Máquina) do sistema. Este aplicativo foi criado com o uso do App Inventor.

O aplicativo está dividido em cinco telas, sendo elas: Tela Login, Tela de conexão bluetooth, Tela Menu, Tela de Monitoramento, Tela de Iluminação e Tela de Controle.

3.4.1 TELA LOGIN

A tela inicial do aplicativo, mostrada na Figura 26, será onde o usuário entrará com seu respectivo login e senha para poder ter acesso ao aplicativo. A seção de Controle só poderá ser acessada pelos administradores.

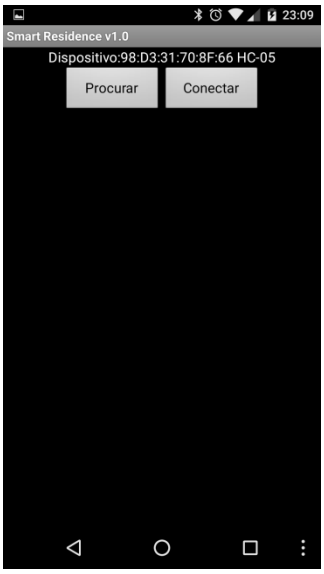
Figura 26 – Tela Login do aplicativo

Fonte: Elaborado pelo autor.

3.4.2 TELA DE CONEXÃO BLUETOOTH

A segunda tela, mostrada na figura 27, é responsável pela conexão do seu smartphone com o sistema de controle e é composta de dois botões: Procurar e Conectar. O primeiro, busca todos os sinais bluetooth disponíveis para pareamento. Após selecionar o sinal do sistema, clica-se em conectar para iniciar a conexão com o sistema. Quando a conexão é feita, é aberta automaticamente a Tela Menu.

Figura 27 – Tela de conexão bluetooth



Fonte: Elaborado pelo autor.

3.4.3 TELA MENU

A Tela Menu, observada na figura 28, é a tela que te dará acesso a todo o sistema. Ela é composta de três botões: Iluminação, Monitoramento e Controle. Ao clicar em algum dos botões, o usuário é redirecionado para a respectiva seção.

Figura 28 – Tela Menu do aplicativo

Fonte: Elaborado pelo autor.

3.4.4 TELA DE ILUMINAÇÃO

A Tela de Iluminação, ilustrada na figura 29, é onde o usuário poderá ligar ou desligar as lâmpadas. Há duas lâmpadas que podem ser acionadas pelo aplicativo: A lâmpada do Escritório e a da Sala. A tela é composta por botões de Liga/Desliga e ícones que mostram o status atual da lâmpada (se a lâmpada está ligada ou desligada).

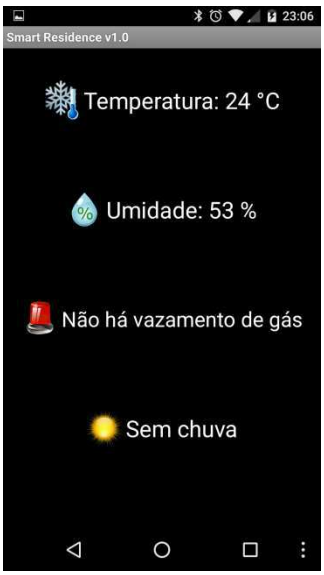
Figura 29 – Tela Iluminação do aplicativo

Fonte: Elaborado pelo autor.

3.4.5 TELA DE MONITORAMENTO

Na tela de monitoramento, mostrada na figura 30, pode-se encontrar os valores atuais de temperatura e umidade do quarto. Os valores são atualizados instantaneamente pelo Arduino e os dados são enviados ao aplicativo. Nesta tela também é possível verificar se está ou não chovendo. Há três estados: Sem chuva, chuva moderada e chuva forte. Também é possível, nesta tela, verificar se está ocorrendo vazamento de gás na cozinha.

Figura 30 – Tela Monitoramento do aplicativo



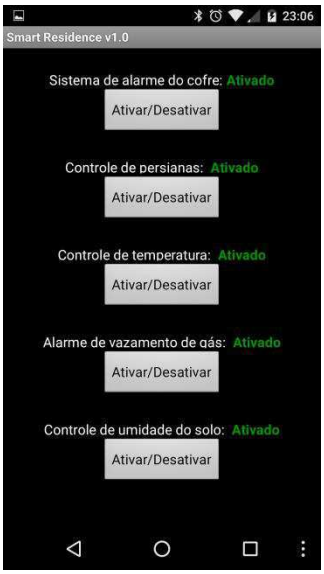
Fonte: Elaborado pelo autor.

3.4.6 TELA DE CONTROLE

Nesta tela, ilustrada na figura 31, pode-se ativar ou desativar os sistemas de controle existentes na casa. Há cinco deles: Controle de temperatura, Alarme de vazamento de gás, Alarme do Cofre, Controle de umidade do solo e Controle de persianas.

O controle temperatura aciona o cooler ao detectar no quarto uma temperatura acima de 29°C. O alarme de vazamento de gás é acionado quando a taxa de gás na cozinha está muito acima do valor normal. O alarme do cofre é acionado quando o feixe de luz do raio laser não incide sobre o sensor de luminosidade LDR. Já o controle de persianas aciona um motor que fechará a janela em caso de chuvas.

Figura 31 – Tela Controle do aplicativo



Fonte: Elaborado pelo autor.

4 CONCLUSÃO E TRABALHOS FUTUROS

Este capítulo final do trabalho de conclusão de curso está dividido em duas seções que apresentam as conclusões do trabalho realizado e a definição de trabalhos futuros que poderão dar continuidade aos resultados obtidos neste projeto.

4.1 CONCLUSÃO

A domótica vem crescendo bastante, trazendo aos seus usuários mais segurança, economia e conforto. Os preços dos materiais utilizado para criar um projeto de domótica estão cada vez mais baixos, devido ao avanço da tecnologia.

Neste projeto fez-se o levantamento e compra de todo material utilizado, criação do aplicativo para dispositivos móveis com Android e toda a instalação do projeto na maquete.

O Arduino foi o componente responsável por todo o controle do sistema, mostrando-se uma ferramenta com um bom desempenho para este tipo de tarefa.

O aplicativo responsável pelo controle e monitoramento pelo dispositivo móvel foi criado no AppInventor, uma ferramenta bastante simples para iniciantes. Porém, para projetos de grande porte, é recomendável a utilização de outro software, devido a sua limitação em alguns aspectos.

Disciplinas como Introdução a Programação, Técnicas de Programação, Instalações Elétricas, Arquitetura de Sistemas Digitais e Circuitos Elétricos foram de suma importância para o desenvolvimento das atividades realizadas pelo estagiário no decorrer do trabalho.

4.2 TRABALHOS FUTUROS

Em termos de trabalhos futuros, há muitas linhas que podem ser seguidas para dar continuidade a este projeto.

Primeiramente, seria interessante construir o aplicativo para o smartphone em uma plataforma mais robusta, como o Eclipse SDK (que utiliza a linguagem Java). Esta plataforma fornece uma maior gama de opções para incrementar o projeto. Outra possibilidade seria utilizar uma *Ethernet Shield* e criar uma IHM para ser acessada pela internet, que substituiria a tecnologia *Bluetooth* utilizada neste projeto.

Realizar estudos junto com a área de Engenharia Agrícola para aperfeiçoar o sistema de controle de irrigação, realizando ensaios para obter resultados ótimos para qual seria o melhor momento para irrigar as possíveis plantas e/ou jardins.

Utilizar um sistema de alertas via SMS com o Arduino (isto é possível utilizando o Arduino GSM GPRS *Shield*), para poder alertar o usuário durante uma possível ausência de internet. No caso de uma queda de energia, seria interessante a utilização de um dispositivo alternativo para a alimentação dos sensores, atuadores e microcontrolador (baterias, por exemplo).

Utilizar a ideia deste projeto em futuros laboratórios das disciplinas de Laboratório de Controle Digital, bem como na disciplina de Laboratório de Instalações Elétricas.

REFERÊNCIAS

BOLZANI, Caio Augustus Morais, Residências inteligentes. São Paulo, Ed. Editora e Livraria da Física, 2007.

AURESIDE, Associação Brasileira de Automação Residencial. A casa da Microsoft. Disponível em <http://www.aureside.org.br/temastec/default.asp file=concbasicos03.asp>. Acessado em: 09 de Out.. de 2016.

ROQUE, António. Introdução a domótica. Publicado na Revista O Electricista, nº1 Jul, Agos e Set de 2002. Disponível em: <http://www.antonioroque.com/textos.asp idCat=11&idArtigo=12>. Acessado em: 08 de Out. de 2016.

MCROBERTS, Michael. (tradução Rafael Zanolli), Arduíno Básico, São Paulo: Editora Novatec, 2011.

ARDUINO. Site Oficial. Disponível em: <http://www.arduino.cc> Acessado em: 29 de Out. 2013

BUILD YOUR SMART HOME, X-10 Protocol. Disponível em: <http://buildyoursmarthome.co/home-automation/protocols/x10/>. Acesso em: 17 de Out. de 2016.

KOBAYASHI, Carlos. A tecnologia Bluetooth e suas aplicações, 2004. Disponível em: <http://grenoble.ime.usp.br/movel/monografia_bluetooth.pdf>. Acessado em: 17 de Out. de 2016.

APÊNDICE A – CÓDIGO DO ARDUINO

```
/*****************************************************************
                        TCC – IGOR DANTAS ROCHA
 *****************************************************************/

#include <SoftwareSerial.h>
#include <dht.h>
#include <Servo.h>

#define pinChuva A0
#define pinServo A1
#define pinLDRLavabo A2
#define pinAcionaLuzLavabo 5
#define pinAcionaIrrigacao 7
#define pinAcionaCooler 46
#define pinDHT A15
#define pinBuzzerCofre 22
#define pinBuzzerCozinha 53
#define pinLDRCofre A14
#define pinPIR 30
#define luzlaser 23
#define pinMQ2 A13
#define pinLuz1 38
#define pinLuz2 39
#define pinLuz3 40

Servo servo1; // Instanciar servo.
SoftwareSerial bluetooth(10, 11); // 10 = Rx 11 = Tx Conexão bluetooth
dht DHT;

int readBluetooth; //Variável que irá receber o comando enviado do Android
int pos = 0;

//**CRIAÇÃO DE VARIAVEIS AUXILIARES
int valorldr1;
int valorpir1;
int valorchuva;
int valormq2;
int valorldrlavabo;
String statusAlarme = "testando";
String statusChuva = "Sem chuva";
String statusluz1 = "desligada";
String statusluz2 = "desligada";
String statusluz3 = "desligada";
String statusCofre = "ativado";
String statusPersianas = "ativado";
```

```
String statusTemperatura = "ativado";
String statusGas = "ativado";
String statusUmidade = "ativado";


void setup() {
  Serial.begin(9600); //Inicia comunicação serial
  bluetooth.begin(9600); //habilita as portes selecionadas para se comunicarem por
serial.
  servo1.attach(pinServo);

  pinMode(pinChuva, INPUT);
  pinMode(pinAcionaIrrigacao, OUTPUT);
  digitalWrite(pinAcionaIrrigacao, HIGH);
  pinMode(luzlaser, OUTPUT);
  digitalWrite(luzlaser, HIGH);
  pinMode(pinLuz1, OUTPUT);
  digitalWrite(pinLuz1, HIGH);
  pinMode(pinLuz2, OUTPUT);
  digitalWrite(pinLuz2, HIGH);
  pinMode(pinLuz3, OUTPUT);
  digitalWrite(pinLuz3, HIGH);
  pinMode(pinPIR, INPUT);
  pinMode(pinDHT, INPUT);
  pinMode(pinLDRCofre, INPUT);
  pinMode(pinLDRLavabo, INPUT);
  pinMode(pinAcionaLuzLavabo, OUTPUT);
  pinMode(pinAcionaCooler, OUTPUT);
  pinMode(pinBuzzerCofre, OUTPUT);
  pinMode(pinMQ2, INPUT);
  //  pinMode(rele4, OUTPUT);
  //  digitalWrite(rele4, LOW);
}


void loop() {
  //    int valorChuva = analogRead(pinChuva);
  valorldr1 = analogRead(pinLDRCofre);
  valorpir1 = digitalRead(pinPIR);
  valormq2 = analogRead(pinMQ2);
  valorchuva = analogRead(pinChuva);
  valorldrlavabo = analogRead(pinLDRLavabo);

  //**VERIFICAÇÃO DOS VALORES NO MONITOR SERIAL;
  DHT.read11(pinDHT); //lê infos do sensor
  Serial.print("Umidade: ");
  Serial.println(DHT.humidity);
  Serial.print("    Temperatura: ");
  Serial.println(DHT.temperature);
  Serial.print("valor LDR: ");
```

```
  Serial.println(valorldrlavabo);
  Serial.print("valor pir ");
  Serial.println(valorpir1);
  Serial.print("valor statuscofre:");
  Serial.println(statusCofre);
  Serial.print("valor mq2:");
  Serial.println(valormq2);
  Serial.print("valor Chuva:"); //800 eh o valor maximo obtido.
  Serial.println(valorchuva);
  Serial.println(statusluz1 + ',' + statusluz2 + ',' + statusluz3);


//ENVIO DE DADOS PARA O APLICATIVO

  bluetooth.print(String((int)DHT.humidity) + ',' + String((int)DHT.temperature)+ ',' +
statusChuva + ',');
  bluetooth.print(statusluz1 + ',' + statusluz2 + ',' + statusluz3 + ',');
  bluetooth.print(statusCofre + ',' + statusPersianas + ',' + statusTemperatura + ',' +
statusGas + ',' + statusUmidade + ',');
  delay(500);

//TRATAMENTO DO SENSOR DE CHUVA
  if(valorchuva > 550) {
    statusChuva = "Chuva forte";
  } else {
      if (valorchuva > 100 && valorchuva <= 550) {
        statusChuva = "Chuva moderada";
      } else {
          if (valorchuva <= 100) {
          statusChuva = "Sem chuva";
          }
        }
    }

//TRATAMENTO LUZ LAVABO
  if(valorldrlavabo < 300) {
    digitalWrite(pinAcionaLuzLavabo, HIGH);
  } else {
    digitalWrite(pinAcionaLuzLavabo, LOW);
  }

//TRATAMENTO SISTEMA IRRIGAÇÃO
  if(statusUmidade == "ativado") {
    digitalWrite(pinAcionaIrrigacao, HIGH);
  }

//TRATAMENTO FECHAMENTO DE JANELA
  if (statusPersianas == "ativado") {
    if (valorchuva > 100) {
    abrirjanela();
```

```
    }
  }

//TRATAMENTO CONTROLE DE TEMPERATURA NO QUARTO
if (statusTemperatura == "ativado") {
  if(DHT.temperature > 27) {
      digitalWrite(pinAcionaCooler, HIGH);
  } else {
      digitalWrite(pinAcionaCooler, LOW);
  }
}

//TRATAMENTO CONTROLE DE SEGURANÇA COFRE
 if (statusCofre == "ativado") {
    if (valorldr1 < 800) {
      tocarsirene();
    }
 }

///TRATAMENTO VAZAMENTO DE GÁS
 if(statusGas == "ativado") {
    if (valormq2 > 2000) {
      tocarsirene2();
    }
 }

// RECEBIMENTO DE DADOS DO APLICATIVO
  if (bluetooth.available() > 0) { //Verifica se algo chegou via Bluetooth
    readBluetooth = bluetooth.read(); //Grava esse algo lido na variável
    if (readBluetooth == 'a') {
      digitalWrite(pinLuz1, !digitalRead(pinLuz1)); //Alterna estado da iluminação de led
da sala;
    }
    if (readBluetooth == 'b') {
      digitalWrite(pinLuz2, !digitalRead(pinLuz2)); //Alterna estado da iluminação de led
do escritório
    }
    if (readBluetooth == 'c') {
      digitalWrite(pinLuz3, !digitalRead(pinLuz3)); //variável para testes;
    }
    if (readBluetooth == 'e') {    // MUDANÇA DE STATUS CONTROLE DE JANELA
      if (statusPersianas == "ativado") {
        statusPersianas = "desativado";
      } else {
      if (statusPersianas == "desativado") {
        statusPersianas = "ativado";
      }
        }
    }
```

```
   if (readBluetooth == 'f') { // MUDANÇA DE STATUS CONTROLE DE
TEMPERATURA
     if (statusTemperatura == "ativado") {
       statusTemperatura = "desativado";
     } else {
     if (statusTemperatura == "desativado") {
       statusTemperatura = "ativado";
     }
       }
   }
   if (readBluetooth == 'd') { //MUDANÇA DE STATUS CONTROLE DE COFRE
     if (statusCofre == "ativado") {
       statusCofre = "desativado";
     } else {
     if (statusCofre == "desativado") {
       statusCofre = "ativado";
     }
       }
   }
   if (readBluetooth == 'g') {  //STATUS CONTROLE DE GÁS
     if (statusGas == "ativado") {
       statusGas = "desativado";
     } else {
     if (statusGas == "desativado") {
       statusGas = "ativado";
     }
       }
   }
   if (readBluetooth == 'h') { //STATUS CONTROLE DE IRRIGAÇÃO
     if (statusUmidade == "ativado") {
       statusUmidade = "desativado";
     } else {
     if (statusUmidade == "desativado") {
       statusUmidade = "ativado";
     }  //Alterna estado da iluminação de led da sala;
       }
   }
 }

 if(digitalRead(pinLuz1) == HIGH) { //BOOLEAN DA LUZ DA SALA
   statusluz1 = "desligada";
 }
 else {
   statusluz1 = "ligada";
 }

 if(digitalRead(pinLuz2) == HIGH) { //BOOLEAN DA LUZ DO ESCRITÓRIO
   statusluz2 = "desligada";
 }
 else {
```

```
    statusluz2 = "ligada";
   }

   if(digitalRead(pinLuz3) == HIGH) { //BOOLEAN VARIAVEL AUXILIAR
    statusluz3 = "desligada";
   }
   else {
    statusluz3 = "ligada";
   }


}

//FUNÇÕES AUXILIARES

void tocarsirene() { // ALARME TOCADO NO BUZZER

#define tempo 10
  int frequencia = 0;
  int Pinofalante = 10;

  //pinMode(Pinofalante,OUTPUT); //Pino do buzzer

  for (frequencia = 150; frequencia < 1500; frequencia += 1)
  {
    tone(pinBuzzerCofre, frequencia, tempo);
    delay(1);
  }
  for (frequencia = 1500; frequencia > 150; frequencia -= 1)
  {
    tone(pinBuzzerCofre, frequencia, tempo);
    delay(3);
  }
}

void tocarsirene2() { //ALARME TOCADO NO BUZZER

#define tempo 10
  int frequencia = 0;
  int Pinofalante = 10;

  //pinMode(Pinofalante,OUTPUT); //Pino do buzzer

  for (frequencia = 150; frequencia < 1500; frequencia += 1)
  {
    tone(pinBuzzerCozinha, frequencia, tempo);
    delay(1);
  }
  for (frequencia = 1500; frequencia > 150; frequencia -= 1)
  {
```

```
    tone(pinBuzzerCozinha, frequencia, tempo);
    delay(3);
  }
}

void abrirjanela() { //FUNÇÃO DE ABERTURA DE JANELA
  for (pos = 0; pos <= 90; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    servo1.write(pos);              // tell servo to go to position in variable 'pos'
    delay(25);                      // waits 15ms for the servo to reach the position
  }
}

void fecharjanela() {  //FUNÇÃO DE FECHAMENTO DE JANELA
  for (pos = 90; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    servo1.write(pos);              // tell servo to go to position in variable 'pos'
    delay(25);                      // waits 15ms for the servo to reach the position
  }
}
```

ANEXO A – CÓDIGO DE CALIBRAÇÃO DO SENSOR MQ-2

```
/******Demo for MQ-2 Gas Sensor Module V1.0*****
Support:  Tiequan Shao: support[at]sandboxelectronics.com

License: Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0)

Note:    This piece of source code is supposed to be used as a demonstration ONLY.
More sophisticated calibration is required for industrial field application.

                                        Sandbox Electronics    2011-04-25
************************************************************************/

#define         MQ_PIN                  (0)     //define which analog input channel you are
going to use
#define         RL_VALUE                (5)     //define the load resistance on the board, in
kilo ohms
#define         RO_CLEAN_AIR_FACTOR     (9.83)
//RO_CLEAR_AIR_FACTOR=(Sensor resistance in clean air)/RO,
                                        //which is derived from the chart in datasheet

/***********************Software Related Macros************************/
#define         CALIBARAION_SAMPLE_TIMES    (50)    //define how many samples
you are going to take in the calibration phase

#define         CALIBRATION_SAMPLE_INTERVAL  (500)   //define the time
interal(in milisecond) between each samples in the cablibration phase

#define         READ_SAMPLE_INTERVAL        (50)    //define how many samples
you are going to take in normal operation
#define         READ_SAMPLE_TIMES           (5)     //define the time interal(in
milisecond) between each samples in normal operation


/**********************Application Related Macros******************/
#define         GAS_LPG                 (0)
#define         GAS_CO                  (1)
#define         GAS_SMOKE               (2)

/*****************************Globals***********************************/
float         LPGCurve[3]  =  {2.3,0.21,-0.47};   //two points are taken from the curve.
                                        //with these two points, a line is formed which is
"approximately equivalent" to the original curve.
```

```
                                    //data format:{ x, y, slope}; point1: (lg200, 0.21),
point2: (lg10000, -0.59)
float       COCurve[3]  =  {2.3,0.72,-0.34};    //two points are taken from the curve.
                                    //with these two points, a line is formed which is
"approximately equivalent" to the original curve.
                                    //data format:{ x, y, slope}; point1: (lg200, 0.72),
point2: (lg10000,  0.15)
float       SmokeCurve[3] ={2.3,0.53,-0.44};    //two points are taken from the curve.
                                    //with these two points, a line is formed which is
"approximately equivalent" to the original curve.
                                    //data format:{ x, y, slope}; point1: (lg200, 0.53),
point2: (lg10000,  -0.22)
float  Ro  =  10;                   //Ro is initialized to 10 kilo ohms

void setup()
{
  Serial.begin(9600);                       //UART setup, baudrate = 9600bps
  Serial.print("Calibrating...\n");
  Ro = MQCalibration(MQ_PIN);                   //Calibrating the sensor. Please make
sure the sensor is in clean air
                                    //when you perform the calibration
  Serial.print("Calibration is done...\n");
  Serial.print("Ro=");
  Serial.print(Ro);
  Serial.print("kohm");
  Serial.print("\n");
}

void loop()
{
   Serial.print("LPG:");
   Serial.print(MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_LPG) );
   Serial.print( "ppm" );
   Serial.print("    ");
   Serial.print("CO:");
   Serial.print(MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_CO) );
   Serial.print( "ppm" );
   Serial.print("    ");
   Serial.print("SMOKE:");
   Serial.print(MQGetGasPercentage(MQRead(MQ_PIN)/Ro,GAS_SMOKE) );
   Serial.print( "ppm" );
   Serial.print("\n");
   delay(200);
}

/****************** MQResistanceCalculation ****************************
Input:   raw_adc - raw value read from adc, which represents the voltage
Output:  the calculated sensor resistance
Remarks: The sensor and the load resistor forms a voltage divider. Given the voltage
         across the load resistor and its resistance, the resistance of the sensor
```

```
        could be derived.
************************************************************************/
float MQResistanceCalculation(int raw_adc)
{
  return ( ((float)RL_VALUE*(1023-raw_adc)/raw_adc));
}


/***************************** MQCalibration ****************************
Input:   mq_pin - analog channel
Output:  Ro of the sensor
Remarks: This function assumes that the sensor is in clean air. It use
         MQResistanceCalculation to calculates the sensor resistance in clean air
         and then divides it with RO_CLEAN_AIR_FACTOR.
RO_CLEAN_AIR_FACTOR is about
         10, which differs slightly between different sensors.
************************************************************************/
float MQCalibration(int mq_pin)
{
  int i;
  float val=0;

  for (i=0;i<CALIBARAION_SAMPLE_TIMES;i++) {            //take multiple samples
    val += MQResistanceCalculation(analogRead(mq_pin));
    delay(CALIBRATION_SAMPLE_INTERVAL);
  }
  val = val/CALIBARAION_SAMPLE_TIMES;                   //calculate the average
value

  val = val/RO_CLEAN_AIR_FACTOR;                        //divided by
RO_CLEAN_AIR_FACTOR yields the Ro
                                        //according to the chart in the datasheet

  return val;
}


//*****************************  MQRead *****************************
Input:   mq_pin - analog channel
Output:  Rs of the sensor
Remarks: This function use MQResistanceCalculation to caculate the sensor resistence
(Rs).

         The Rs changes as the sensor is in the different concentration of the target
         gas. The sample times and the time interval between samples could be configured
         by changing the definition of the macros.
************************************************************************/
float MQRead(int mq_pin)
{
  int i;
```

```
 float rs=0;

 for (i=0;i<READ_SAMPLE_TIMES;i++) {
  rs += MQResistanceCalculation(analogRead(mq_pin));
  delay(READ_SAMPLE_INTERVAL);
 }

 rs = rs/READ_SAMPLE_TIMES;

 return rs;
}


/*****************************  MQGetGasPercentage **********************
Input:   rs_ro_ratio - Rs divided by Ro
        gas_id      - target gas type
Output:  ppm of the target gas
Remarks: This function passes different curves to the MQGetPercentage function which
        calculates the ppm (parts per million) of the target gas.
************************************************************************/

int MQGetGasPercentage(float rs_ro_ratio, int gas_id)
{
 if ( gas_id == GAS_LPG ) {
   return MQGetPercentage(rs_ro_ratio,LPGCurve);
 } else if ( gas_id == GAS_CO ) {
   return MQGetPercentage(rs_ro_ratio,COCurve);
 } else if ( gas_id == GAS_SMOKE ) {
   return MQGetPercentage(rs_ro_ratio,SmokeCurve);
 }

 return 0;
}


/*****************************  MQGetPercentage ************************
Input:   rs_ro_ratio - Rs divided by Ro
        pcurve      - pointer to the curve of the target gas
Output:  ppm of the target gas
Remarks: By using the slope and a point of the line. The x(logarithmic value of ppm)
        of the line could be derived if y(rs_ro_ratio) is provided. As it is a
        logarithmic coordinate, power of 10 is used to convert the result to non-
logarithmic
        value.
************************************************************************
**************/
int  MQGetPercentage(float rs_ro_ratio, float *pcurve)
{
 return (pow(10,( ((log(rs_ro_ratio)-pcurve[1])/pcurve[2]) + pcurve[0])));
}
```