CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

CAIO VARGAS MOI

ALARME ATIVADO POR UM SENSOR DE MOVIMENTO

Orientador: Prof. Luciano Duque

Brasília

Dezembro, 2014

CAIO VARGAS MOI

ALARME ATIVADO POR UM SENSOR DE MOVIMENTO

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

Brasília

Dezembro, 2014

CAIO VARGAS MOI

ALARME ATIVADO POR UM SENSOR DE MOVIMENTO

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

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

_____

Prof. Abiezer Amarilia Fernandes

Coordenador do Curso

Banca Examinadora:

_____

Prof. Luciano Duque, titulação.

Orientador

_____

Prof. nome, titulação.

Centro Universitário de Brasília (UniCEUB)

_____

Prof. nome, titulação.

Centro Universitário de Brasília (UniCEUB)

_____

Prof. nome, titulação.

Centro Universitário de Brasília (UniCEUB)

DECICATÓRIA

Dedico este trabalho à minha família (pai, mãe, irmãos, tios, primos e avós), que me deu todo o apoio para que eu fizesse o curso e o concluísse, à minha namorada, Mônica Barroso, que me apoiou e auxiliou sempre que necessário para a conclusão deste projeto, e a todos os meus amigos, que me apoiaram nos momentos de dificuldade da vida.

Caio Moi

AGRADECIMENTOS

Gostaria de agradecer, primeiramente, a Deus, por me guiar e me iluminar em todos os momentos de minha vida. Cada oportunidade e vitória, eu devo a Ele.

Agradeço também à minha família que me educou, me apoiou, me incentivou e me deu todas as condições necessárias para que eu pudesse chegar até aqui.

Agradeço à minha namorada, que me deu todo o apoio e incentivo para que eu concluísse a minha graduação.

Agradeço aos meus amigos, que, assim como minha família, sempre me apoiaram nos momentos difíceis da vida.

Aos colegas, que me auxiliaram neste projeto, me estendendo o braço sempre que solicitado.

A todos: Muito obrigado!

Caio Moi

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS E TABELAS

LISTA DE SIGLAS

AT	*Hayes AT Commands*
bps	bits por segundo
GLP	Gás Liquefeito de Petróleo
GSM	*Global System for Mobile Communications* (Sistema Global para Comunicações Móveis)
Hz	Hertz
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
kHz	Quilo Hertz
kbps	Quilo bits por segundo
LCD	*Liquid Crystal Display* (Visor de Cristal Líquido)
LED	*Light Emitting Diode* (Diodo Emissor de Luz)
MHz	Mega Hertz
PIR	*Passive Infrared* (Infravermelho Passivo)
ppm	Partes Por Milhão
SIM	*Subscriber Identification Module* (Módulo de Identificação do Assinante)
SMS	*Short Message Service* (Serviço de Mensagem Curta)
USB	*Universal Serial Bus* (Barramento Serial Universal)
V	Volts

RESUMO

Este trabalho foi feito com o intuito de apresentar um alarme com detecção de movimento por meio de um sensor PIR (infravermelho passivo) que, integrado a um GSM *Shield*, informa ao usuário, por meio de um SMS, que o mesmo teve sua propriedade invadida.

Para que o projeto seja possível, foi construída uma placa para que se acoplasse com um microcontrolador Arduino integrado a um GSM *Shield* e o sensor de movimento mencionado. O sensor detectará o movimento ativando um LED, um *buzzer* e acionando o envio do SMS a um número de celular cadastrado no código-fonte.

Palavras-chave: Arduino UNO. Módulo GSM. Microcontrolador. Alarme. Sensor PIR.

ABSTRACT

This work was done in order to present an alarm with motion detection by means of a PIR sensor (passive infrared) that integrated in a GSM Shield informs the user by means of a SMS, which had the same property their invaded.

To make the project possible, that a board was built to docked with an integrated Arduino microcontroller to a GSM Shield and the said motion sensor. The sensor detects the movement by activating an LED, a buzzer and triggering sending the SMS to a mobile number registered in the source code.

Keywords: Arduino UNO. GSM module. Microcontroller. Alarm. PIR sensor.

CAPÍTULO 1 – INTRODUÇÃO

Alarmes sensoriais servem para serem usados sempre que se desejar detectar a passagem de uma pessoa ou animal por um determinado local.

Conforme Armando Silva afirma em sua monografia "Alarme com Ativação por Sensor Presencial e Alerta Via SMS", por definição, sensores são dispositivos que respondem a estímulos físicos ou químicos, gerando um sinal positivo ou negativo ou gerando um valor medido.

Anda conforme o mesmo, o Sensor Infravermelho Passivo (PIR) possui calibragem para temperatura do corpo humano. Apesar de ser considerado um sensor de movimento, na verdade trata-se de um sensor de temperatura.

A radiação infravermelha possui frequência menor ou igual à frequência da luz, devido a isto ela praticamente não pode ser vista a olho nu. É uma radiação que não causa efeitos que possam prejudicar o corpo humano e é caracterizada como alta emissora de oscilações eletromagnéticas por segundo.

Todos os objetos (corpos) emitem certa quantia de luz infravermelha, essa luz infravermelha nada mais é do que as diferenças de temperatura no corpo. Quando o sensor detecta alguma diferença repentina de temperatura em algum corpo, significa dizer que o sensor "detectou movimento".

Segundo Thomazini (2005, p.37):

> Infravermelho Passivo: Trata-se apenas de um receptor de infravermelho com ajuste de sensibilidade. É utilizado principalmente para alarmes de intrusão, pois detecta o calor humano a uma distância razoável. O elemento sensitivo desse tipo de sensor é pirotérmico integrado.

Este projeto visa criar um protótipo de alarme utilizando o sensor infravermelho para detectar a alteração de temperatura (reconhecendo movimento) e enviar um SMS para o usuário ao detectar este movimento.

Para criar este protótipo será necessário utilizar uma base de *hardware* e em cima desta base, acrescentar periféricos que permitam chegar aos resultados esperados. No caso, será utilizado como base a plataforma Arduino e como periféricos serão utilizados um LED,

um módulo *GSM Shield*, *buzzer* e sensor de movimento (PIR). Este protótipo deverá funcionar da seguinte forma: o sensor de movimento detecta a passagem de alguém ou algo, o Arduino fará a ativação do LED e do *buzzer*, além de enviar um comando ao módulo GSM para que o dono da propriedade seja alertado sobre a invasão. Após isto, o usuário poderá fazer a desativação do alarme via SMS, bem como ativá-lo novamente.

1.1 Motivação

A cada dia que passa a tecnologia toma mais espaço na vida das pessoas, em todas as áreas, como por exemplo: telefonia, computadores portáteis, automóveis, etc.

Segundo o Jornal A Folha do Médio Norte, no artigo "A Tecnologia Cada Vez Mais Presente Na Vida Dos Cidadãos":

> "O mundo hoje é comandado pela tecnologia e cada vez mais se faz necessário conhecê-la e dominá-la. Em nossas casas, escritórios, linhas de produção das indústrias, veículos, enfim para todos os lados que olhamos lá está ela presente em nosso dia-a-dia."

Com a segurança não haveria de ser diferente. Existem muitos equipamentos tecnológicos diferentes na área da segurança tanto referentes a automóveis quanto à residências. Dentre estes equipamentos, o mais simples e comum é o alarme sensorial.

O alarme sensorial permite detectar intrusões de áreas, trazendo, assim, maior segurança para o seu usuário. Porém, de nada serão úteis sirenes e luzes se o usuário e/ou as instituições competentes não souber(em) que a propriedade foi invadida, esteja o proprietário presente ou ausente.

Estando ciente sobre a intrusão, o usuário poderá tomar as devidas providências e, desta forma, proteger a sua propriedade. Foi pensando nisto que este projeto foi criado. O projeto propõe o envio de um SMS para o usuário assim que o dispositivo detectar o movimento na casa para que o mesmo tome as devidas providências.

O projeto visa criar um protótipo funcional capaz de fazer com que invasores de propriedades desistam do delito, ou que sejam pegos em flagrante, visando proteção e segurança aos seus usuários. Para tal, foram utilizados componentes tais como: arduíno, *cellular shield* compatível, sensor de movimento e um *buzzer*.

O problema foi observado ao notar-se que a violência tem crescido cada vez mais no mundo. O protótipo criado é capaz de proteger não somente propriedades fechadas, mas também propriedades abertas (ex: canteiros-de-obras), tendo em vista que ocorrem alguns furtos nestas propriedades por terem pouca vigilância e itens fáceis de serem furtados e difíceis de serem contabilizados.

1.2 Objetivos do Trabalho

1.3.1. Objetivo geral

Implementar um protótipo funcional para propriedades que, por meio da detecção de movimento em um ambiente interno, acione um alarme que envia SMS ao proprietário de forma a tentar fazer com que invasores desistam de cometer o delito de invasão de propriedade ou com que invasores sejam pegos em flagrante cometendo o referido delito, além disto, o dispositivo deve também ser capaz de ser desativado via SMS.

1.3.2. Objetivos específicos

Para desenvolver o sistema de detecção de alerta e de tentativa de captura do invasor, são necessários que se atinjam os seguintes objetivos:

- Realizar pesquisas bibliográficas nas seguintes áreas:
  - Sensores
  - Linguagem de Programação
  - Arduino
  - Rede GSM
- Fazer pesquisas, também, no que tangem os seguintes pontos:
  - Solução que utilize um sensor de movimentos capaz detectar uma invasão;
  - Solução que seja capaz de enviar um SMS por meio de acionamento de um dispositivo;

  - Solução que seja capaz de receber um SMS para fazer o acionamento e a desativação do dispositivo.
- Especificar a solução encontrada, descrevendo os componentes utilizados e suas funções;
- Desenvolver um *hardware* baseado em Arduino que identificará uma intrusão e a alertará ao usuário.
- Incrementar o protótipo com sirene e luzes;
- Programação do microcontrolador;

  A programação implementada no microcontrolador permitirá que o sistema identifique a passagem de uma pessoa, dispara um alarme sonoro e envia um SMS ao proprietário que o mesmo teve sua propriedade invadida.
- Construir um protótipo funcional;

  É o protótipo que permitirá que o sistema funcione da forma como desejada. Ou seja, detectará o movimento, acionará o alarme e enviara o SMS ao proprietário. Este protótipo deverá ser de pequeno porte, para que tenha seu manuseio facilitado para a sua apresentação.
- Detectar invasões a propriedades;
- Enviar um SMS quando o dispositivo for acionado;
- Ativar e desativar o alarme via recebimento de SMS;
- Incrementar o sistema com um sensor de gás que também alerte o usuário via SMS.

1.3 Metodologia

Por meio de várias pesquisas (tanto entre conversas com colegas da área quanto com pesquisas no meio acadêmico) é possível chegar à conclusão de que a forma mais simples de se executar este projeto será utilizando um arduíno, conectado a um sensor de movimento PIR e a uma *Cellular Shield* para arduíno. Desta forma, foram necessários conhecimentos nos seguintes conteúdos para o desenvolvimento do projeto:

- Linguagem de programação C++
- Microcontroladores
- Arduino

- Sensor PIR (*Passive Infra Red*)
- Rede GSM
- SMS
- SIM Card

Por último, descreveremos a metodologia do desenvolvimento do projeto. Para que este projeto seja desenvolvido, é necessário que o mesmo seja dividido em 5 etapas, descritas a seguir:

- **PRIMEIRA ETAPA:** Primeiramente, é necessário fazer as pesquisas sobre o desenvolvimento do projeto. Como fazer o projeto, quais os conhecimentos necessários para sua execução, que recursos utilizar, quais componentes se encaixariam melhor no projeto, etc.
- **SEGUNDA ETAPA:** Decididos os métodos e procedimentos, é necessário, então, adquirir os componentes para a montagem do protótipo.
- **TERCEIRA ETAPA:** É nesta etapa que ocorrerá a montagem do projeto. Esta montagem deve acontecer de forma que o projeto seja simples e de fácil manuseio.
- **QUARTA ETAPA:** Com o protótipo montado é, então, necessário implementar o código-fonte para que o protótipo funcione.
- **QUINTA ETAPA:** A quinta e última etapa é a realização de testes com todos os componentes e o(s) código(s)-fonte juntos. Para o auxílio destes testes aconselha-se a utilização de um Osciloscópio (que indicará o recebimento do sinal e permitirá fazer os ajustes necessários), de uma trena (para medir o raio em que o dispositivo é acionado) e de um compasso (que permitirá medir a angulação do alcance do dispositivo).

1.4 Resultados Esperados

Com a metodologia proposta, espera-se que seja montado um protótipo capaz de detectar corretamente movimentos no ambiente e enviar um SMS ao usuário, informando a invasão de sua propriedade. O dispositivo deve ser simples e de fácil manuseio.

Em relação ao seu funcionamento, espera-se que o protótipo alcance os objetivos traçados no objetivo geral. O dispositivo deverá enviar o SMS sempre que acionado e, para fins de demonstração, não deverá ter um tempo muito longo de intervalo de detecção.

O SMS também somente deverá ser enviado caso ocorra detecção de movimento. E o dispositivo somente deverá ser acionado caso realmente haja movimento. O SMS também deverá ser um meio utilizado para desativação do alarme.

Espera-se que, quando o sensor de movimento tiver valor 1, ou seja, detectar o movimento, a programação e o sistema acionem o *buzzer* e enviem o SMS ao proprietário.

Espera-se que, ao receber o SMS, o alarme seja ativado ou desativado, a depender do SMS enviado.

1.5 Organização

Esta monografia segue organizada da seguinte forma:

O capítulo 2 tratará do referencial teórico utilizado para o desenvolvimento deste projeto. Neste Capítulo teremos um detalhamento dos conhecimentos necessários para o desenvolvimento do protótipo.

O capítulo 3 tratará do detalhamento do projeto. Nele será descrito como foi feito o desenvolvimento do projeto, trazendo diagramas, *softwares*, testes, etc.

O capítulo 4 tratará dos testes feitos com o protótipo. Neste capítulo também se falará dos resultados obtidos no projeto.

E por último, o capítulo 5 trará as conclusões do projeto. Neste capítulo também se falará das ideias para desenvolvimento futuro.

CAPÍTULO 2 - REFERENCIAL TEÓRICO

Neste capítulo, serão descritos os conhecimentos prévios necessários os quais servem de base para o desenvolvimento deste projeto, tais como o *hardware*, a linguagem utilizada no projeto e os componentes.

Neste projeto fez-se uma mescla de dois projetos anteriormente apresentados no UniCEUB, o do aluno Armando Vitor de Oliveira Silva, que apresentou o projeto "Alarme com Ativação por Sensor Presencial e Alerta Via SMS", e do aluno Jefferson Silva Santos, que apresentou o projeto "Detector de Vazamento de Gás com Aviso por SMS".

Ambos os projetos supracitados foram utilizados como embasamento teórico para a elaboração e escrita deste novo projeto.

2.1 Microcontroladores

Nos dias de hoje, os microcontroladores fazem parte da maioria dos equipamentos que são utilizados no dia-a-dia de todos, como nos automóveis, nos eletrodomésticos, nos equipamentos industriais/comerciais etc. Um microcontrolador é um circuito integrado no qual tem implementado um microcomputador.

Segundo Gimenez (2002, p.4):

> Microcontrolador: dispositivo semicondutor em forma de circuito integrado, que integra todas as partes básicas de um microcomputador – microprocessador, memórias não voláteis, memórias voláteis, portas de entrada e saída. Ele é conhecido como um microcomputador implementado em um único circuito integrado. Geralmente, é limitado em termos de quantidade de memória, principalmente no que diz respeito à memória de dados.

Este projeto utiliza-se do microcontrolador Arduino, que será descrito no próximo item.

2.2 Arduino

Arduino é uma plataforma de *hardware* livre que possui suporte de entrada e saída. A linguagem de programação utilizada para o desenvolvimento em arduíno é baseada em C/C++, o que facilita o acesso e a compreensão dos novos usuários. É muito utilizado por pessoas que desejam desenvolver projetos independentes.

A Figura 2.1 ilustra o hardware utilizado no projeto, o Arduíno UNO:

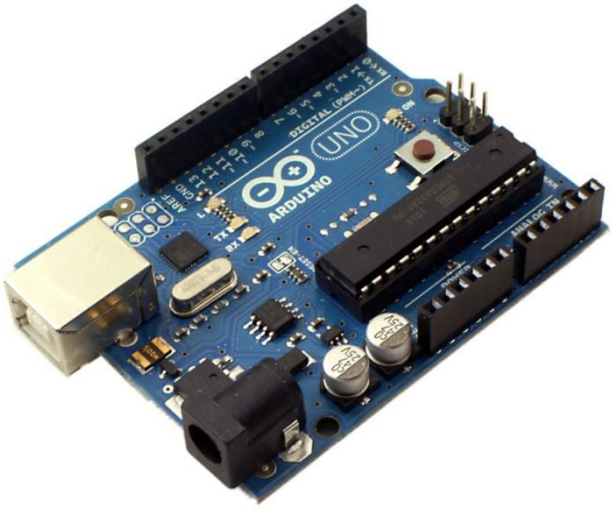


Figura 2.1 - Imagem de um Arduino UNO. (Fonte: http://www.liquidware.com/system/0000/3648/Arduino_Uno_Angle.jpg)

Conforme afirmado no próprio site do Arduino, o Arduino é implementado em uma linguagem conhecida como *Processing*. Esta linguagem é de código aberto e foi inicialmente desenvolvida para ensinar fundamentos da programação computacional em um contexto visual. Para implementar o programa é necessário a utilização do ambiente Arduino IDE, o qual será melhor detalhado mais adiante.

É uma ferramenta para tornar possível de se controlar os computadores além do que o seu computador *desktop*. É uma plataforma de desenvolvimento livre de computação física baseada em uma placa de microcontrolador simples e um ambiente de desenvolvimento para escrever *software* para a placa.

O Arduino pode ser utilizado para desenvolver objetos interativos, tendo entradas a partir de uma variedade de sensores ou interruptores e controle de uma variedade de luzes, motores e outras saídas físicas.

Projetos Arduino podem funcionar sozinhos, de forma independente, ou podem se comunicar com *software* rodando em um computador.

As placas podem ser montadas à mão ou compradas pré-montados e o IDE de código aberto pode ser baixado gratuitamente.

Pode ser executado em várias plataformas diferentes (Linux, Windows, Mac OS). Algumas de suas plataformas oferecem a possibilidade de carregar a programação no *hardware* via USB.

O *software* de sua plataforma é gratuito, o que eleva o interesse de pessoas que querem começar a fazer seus experimentos ou desenvolver novos projetos.

Em seu site (www.arduino.cc), o usuário encontra bibliotecas, exemplos, etc, para poder começar a desenvolver utilizando a plataforma.

Ainda conforme o site do Arduino, o microcontrolador na placa é programado usando a linguagem de programação Arduino e o ambiente de desenvolvimento Arduino IDE.

Placas de Arduino podem ser construídas à mão ou compradas pré-montadas, o que facilita o interesse de novos desenvolvedores de *hardwares*. Os *designs* de referência do *hardware* são disponíveis sob uma licença de código aberto, ou seja, o usuário é livre para adaptá-los às suas necessidades.

As portas do Arduino servem para que possamos ligar componentes ao *hardware*. A utilização destas portas é escrita no programa do próprio *hardware* (Arduino IDE). Estas portas serão melhores descritas no item 2.8.

Utilizando-se o Arduino, encontra-se uma maior facilidade de encontrar módulos que se comuniquem com a plataforma de desenvolvimento escolhida.

O Arduino apresenta o esquema elétrico mostrado na Figura 2.2.

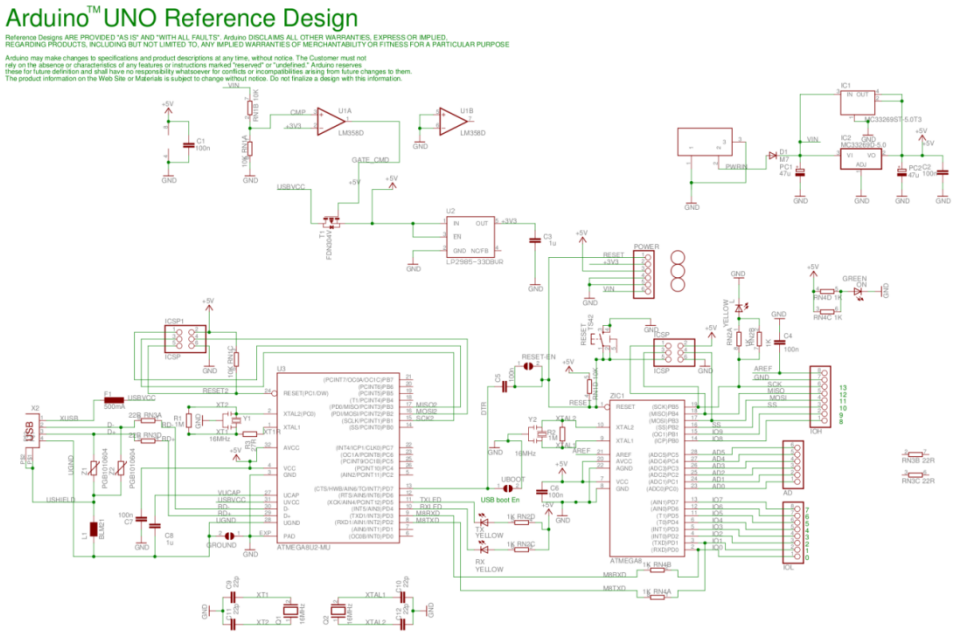


Figura 2.2 – Esquema elétrico do Arduino UNO. (Fonte: http://brittonkerin.com/annotateduino/arduino-uno-schematic.png)

Este esquema elétrico também pode ser encontrado, ampliado, no Anexo A.

Os principais componentes do Arduino são:

- 1 microcontrolador;
- 1 oscilador;
- 1 regulador linear de 5V;
- 1 saída USB;
- 14 pinos digitais de entrada ou saída (programáveis);
- 6 pinos de entrada analógica ou entrada/saída digital (programáveis);
- 5 pinos de alimentação (5V, GND e referência analógica);
- 1 pino de reset;
- 2 pinos para ligar o oscilador.

O microcontrolador, como já visto no item anterior, é responsável por fazer a integração dos componentes. O oscilador é apenas um relógio simples que envia pulsos de tempo em uma frequência especificada. A saída USB serve para enviar os programas a serem interpretados pelo Arduino, além de servir como fonte de energia para que o dispositivo possa funcionar.

Os pinos de entrada digitais podem ter valor 1 (recebe 5V) ou 0 (recebe 0V).

As entradas analógicas possibilitam a verificação da tensão que está na porta, permitindo a leitura (0 a 5V) em 1024 níveis (1024 bits), com o valor máximo 1023 (4.955V) e o valor mínimo é 0 (0V).

Os pinos de saída, tanto digitais quanto analógicos, enviam energia também podendo ter valor entre 1 e 0.

O módulo GSM é um módulo capaz de atuar como facilitador da comunicação que deverá ser provida pelo dispositivo. Ele será o responsável pelo envio das mensagens SMS ao usuário.

O *Cellular Shield* utilizado é o Módulo GSM/GPRS IComSat SIM900. Esta placa pode ser facilmente acoplada no Arduino UNO e, desta forma, é possível estender as funcionalidades GSM para a placa Arduino.

A Figura 2.3 ilustra o Módulo GSM utilizado no projeto:

Figura 2.3 - Módulo GSM/GPRS IComSat SIM 900. (Fonte: http://imall.iteadstudio.com/media/catalog/product/cache/1/image/9df78eab33525d08d6e5fb8d27136e95/i/m/im120417009_9_2.jpg)

Conforme o próprio manual do Módulo SIM900, o mesmo é capaz de executar todas as funcionalidades GSM, como fazer ligações, enviar SMS, receber SMS, acessar internet pela rede 3G, etc. O Quadro 2.1 mostra as especificações do Módulo GSM IComSat SIM900.

Quadro 2.1 – Especificações do Shield SIM900.

Quad-Band 850/900/1800/1900 MHz
GPRS multi-slot classe 10/8
GPRS estação móvel de classe B
Compatível com o GSM fase 2/2+
Classe 4 (2W@850/900MHz)
Classe 1 (1W@1800/1900MHz)
Controle via comandos AT (GSM 07.07, 07.05 e SIMCOM Comandos AT melhorado)
Baixo consumo de energia: 1,5 mA (modo de espera)
Temperatura de operação: -40 ° C a +85 ° C

Fonte: http://imall.iteadstudio.com/im120417009.html

Já a Figura 2.4 abaixo, mostra o esquema elétrico do módulo GSM.

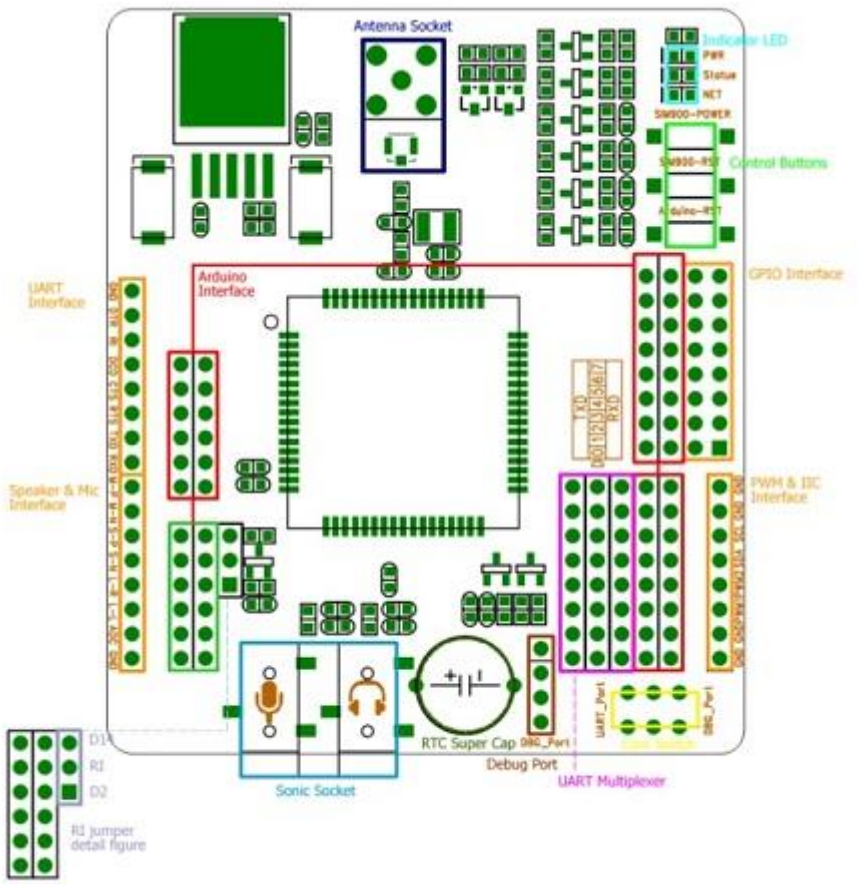


Figura 2.4 – Esquema elétrico do Módulo GSM. (Fonte: http://imall.iteadstudio.com/im120417009.html)

Este esquemático também pode ser encontrado, ampliado, no Anexo B.

2.3 Sistemas de Controle

A definição de controle é manter as atividades planejadas para que se obtenha o desempenho esperado ou um objetivo alcançado.

Já os sistemas de controle, de forma simplista, são conjuntos de interconexões que visam um resultado esperado.

Conforme explicado por Norman Nise em seu livro “Engenharia de Sistemas de Controle”, os sistemas de controle fazem parte da sociedade moderna. Desde ônibus espaciais até a pulverização de água de irrigação, vemos a presença de sistemas de controle. É preciso

que haja controle até mesmo para que se irrigue as plantas sempre no mesmo horário automaticamente. Até mesmo o nosso corpo funciona como um sistema de controle, como quando, por exemplo, estamos fazendo alguma atividade física, o nosso corpo produz mais adrenalina, a adrenalina faz com que nosso coração aumente a velocidade dos batimentos cardíacos para que as nossas células recebam o oxigênio necessário a tempo de fazerem seu trabalho.

Segundo Dorf e Bishop (2001, p.2):

> Um sistema de controle é a interconexão de componentes formando uma configuração de sistema que produzirá uma resposta desejada do sistema. A base para análise de um sistema é forma pelos fundamentos fornecidos pela teoria dos sistemas lineares, que supõe uma relação de causa e efeito para os componentes de um sistema.

Ou ainda, segundo Nise (2002, p.2):

> Um sistema de controle consiste em subsistemas e processos (ou plantas) reunidos com o propósito de controlar as saídas dos processos. Por exemplo, uma caldeira produz calor como resultado do fluxo de combustível. Neste processo, subsistemas chamados válvulas de combustível e atuadores de válvula de combustível são usados para regular a temperatura de uma sala controlando a saída de calor da caldeira. Outros subsistemas, como os termostatos, que se comportam como sensores, medem a temperatura da sala. Na sua forma mais simples, um sistema de controle fornece uma saída ou uma resposta para uma dada entrada ou estímulo.

Esta visão é bem simplista, tendo em vista que hoje em dia, somos capazes de enviar mais de uma entrada e receber mais de uma resposta.

Segundo Ogata (1982, p.2):

> Em virtude de os processos modernos com muitas entradas e saídas tornarem-se mais e mais complexos, a descrição de um sistema de controle moderno exige um grande número de equações. A teoria de controle clássica, que trata apenas de sistemas de entrada simples-saída simples, tornou-se inteiramente impotente para sistemas de múltiplas entradas-múltiplas saídas. Desde 1960, aproximadamente, a teoria de controle moderna tem sido desenvolvida para competir com a complexidade crescente de processos modernos e requisitos rigorosos e estritos em precisão, peso e custo em aplicações militares.

Conforme Katsuhiko Ogata explica em seu livro “Engenharia de Controle Moderno”, para que se possa fazer o controle é preciso definir se o controle será em malha aberta ou malha fechada:

- Malha aberta: Um sistema de controle em malha aberta não permite que a saída gere um novo efeito na ação de controle.
- Malha fechada: Um sistema de controle em malha fechada permite que a saída realimente a entrada. Ou seja, a saída gerada volta a entrada do controlador permitindo que seja gerada uma nova saída.

Nise ainda explica que para se construir um sistema de controle é necessário que exista um motivo ou que seja trazido um benefício. Existem quatro razões principais para se construir um sistema:

- Amplificação e potência;
- Controle remoto;
- Facilidade de uso da forma de entrada;
- Compensação de perturbações.

No caso deste projeto, o controle nos traz o benefício de controlar um dispositivo remotamente por meio de um controle em malha fechada.

A Engenharia de controle e automação percorre inúmeras áreas de conhecimento e funções dentro destas áreas. É ela a responsável por determinar ou implementar os requisitos globais do sistema. Dentre estes requisitos estão as especificações de desempenho do sistema como um todo.

Nise ainda vai além e mostra que para a criação de um projeto de controle são necessários 6 passos:

- Transformar requisitos em um sistema físico;
- Desenhar um diagrama de blocos funcional;
- Criar um diagrama esquemático;
- Desenvolver um modelo matemático;

  Modelos matemáticos descrevem as características dinâmicas de um sistema. Após obter-se o modelo matemático, várias ferramentas podem ser utilizadas para análise e síntese.
- Reduzir o diagrama de bloco;

Esta fase pode ser pulada caso se esteja interessado somente no desempenha de um subsistema individual. Para os demais casos reduz-se os diversos subsistemas a um mais simples.

- Analisar o projeto.
  Para esta fase são usados sinais simples para teste e análise. Uma base de entrada de dados complexa não permite que os testes sejam feitos de forma adequada, tendo em vista que uma vez que ocorra um erro não será possível detectar exatamente qual é o erro ocorrido.

Ogata mostra ainda que existem vários tipos de sistemas de controle, dentre eles:

- Sistemas de controle de pressão;
- Sistemas de controle de velocidade;
- Sistemas de controle numérico;
- Sistemas de controle por computador;
- Sistemas de controle de tráfego;
- Sistemas de controle biológico;
- Sistemas de controle de estoque;
- Sistemas de controle comercial.

Neste projeto o tipo de sistema de controle que será utilizado será o controle de presença.

2.4 Sensor PIR

Um sensor de movimento deve ser capaz de identificar a presença de pessoas dentro do seu raio de ação e possibilitar que seja executada alguma ação. Depois de certo tempo (*delay*), a ser determinado na programação do dispositivo, os equipamentos ativados pelo sensor, são desativados.

Sensores são indispensáveis em residências, condomínios e indústrias que utiliza algum tipo de tecnologia inteligente para economizar energia, para garantir a segurança, para identificar se o carro está parado em frente ao portão eletrônico, etc.

Como explicado anteriormente, os sensores passivos infravermelhos funcionam como sensores de movimento. Neste projeto, esta é exatamente a função exigida deste componente.

Para que o mesmo funcione de forma correta é necessário verificar os ajustes manuais do mesmo quanto à sensibilidade, de forma que o mesmo somente acione o sistema, caso seja detectado uma maior diferença de temperatura.

Este projeto utiliza o sensor PIR BISS0001, capaz de detectar movimentos a distâncias ajustáveis de 3 a 7 metros e com uma amplitude de 140º.

Este sensor consegue amplificar os sinais e, dessa forma, consegue modular um sinal de saída em nível digital. Sua saída gera um sinal ligado/desligado para que possa fazer o mesmo (ligar ou desligar) um circuito de alta potência. As Figuras 2.5 e 2.6 ilustram o sensor de detecção de presença:

Figura 2.5 - Sensor PIR. (Fonte: http://www.huinfinito.com.br/22-58-large/modulo-sensor-de-movimento.jpg)

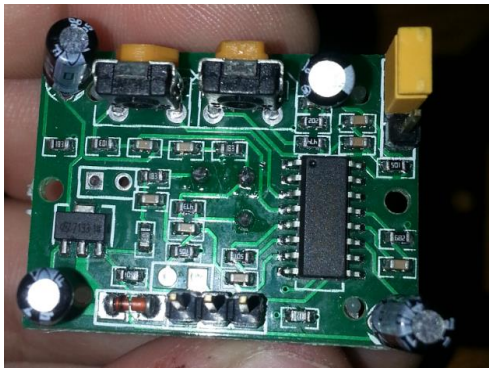


Figura 2.6 - Verso do sensor PIR. (Fonte: Autor)

As principais características do sensor podem ser encontradas no Quadro 2.2.

Quadro 2.2 - Principais características do Sensor PIR.

Tensão de entrada	DC 4.5 a 20V
Corrente estática	50µA
Sinal de saída	0,3V ou 5V (saída alta quando detecta movimento)
Ângulo de detecção	140º
Distância de detecção	Máx. 7m

Fonte: http://www.huinfinito.com.br/sensores/22-modulo-sensor-de-movimento.html

2.5 Sensor de Gás (MQ2)

Este projeto também conta complementação de um sensor de gás MQ-2. O módulo MQ-2 utiliza um semicondutor (SnO_2 – dióxido de estanho) como material sensível, pois possui menor condutividade no ar limpo.

As principais características do MQ-2 estão apresentadas no Quadro 2.3.

Quadro 2.3 – Principais características do MQ-2.

Alta sensibilidade aos gases GLP, propano e hidrogênio
Boa sensibilidade a gases combustíveis
Simples circuito de acionamento
Longa vida e baixo custo

Fonte: http://www.huinfinito.com.br/attachment.php?id_attachment=129

2.6 Linguagem de Programação C/C++

C++ é uma linguagem de programação de uso geral. É uma linguagem multi-paradigma. Por combinar características de linguagens de alto e de baixo níveis, é considerada uma linguagem de médio nível. É uma das linguagens mais populares e é bastante usada também nas universidades devido à grande base de utilizadores e ao grande desempenho.

A linguagem C++ é muito utilizada em terminais de autoatendimento bancários e em vários outros ramos comerciais. C++ permite que se possa aplicar eficazmente qualquer tarefa de programação virtual, pois seus aos atributos são orientados a objetos. Não é incomum ver C++ sendo usada para projetos como editores, bancos de dados, sistemas de arquivos pessoais, utilitários de rede, etc.

Esta linguagem surgiu na década de 80 para implementar uma versão distribuída do Unix, que antes disto era escrito em C.

Segundo VICTORINE (2006):

> Os programas em C tendem a ser bastante compactos e de execução rápida e podem ser desenvolvidos em partes separadas e depois unidos num produto final, que significa que bibliotecas de funções podem ser criadas e distribuídas para serem usadas sem que realmente se conheça o código fonte de cada uma delas.

Nos dias de hoje, muitos programadores dizem que a linguagem C++ define a essência da programação.

Conforme Herbert Schildt explica em seu livro “C++ guia para iniciantes”, C++ é a incrementação da linguagem C, por muito tempo foi chamada de “C com classes” por se tratar

da principal incrementação à linguagem C. Porém, certa vez, Rick Mascitti sugeriu a Bjarne Stroustrup (criador da linguagem) que se chamasse a linguagem de C++. O nome garantiu o lugar de C++ na história, pois todo programador em C reconheceria que se tratava de uma incrementação à linguagem C.

Segundo SCHILDT (2002):

> A programação orientada a objetos pegou as melhores ideias da programação estruturada e as combinou com diversos novos conceitos. O resultado foi uma maneira melhor e diferente de organizar um programa. NO sentido mais geral, um programa pode ser organizado em uma de duas maneiras: em torno de seu código (o que está acontecendo) ou em torno de seus dados (quem esta sendo afetado).

Esta linguagem serve como base para que se possa compreender melhor a linguagem de programação utilizada neste projeto, que é feita no Arduino IDE.

2.7 Rede GSM

A rede GSM "é uma tecnologia celular aberto, digital utilizado para a transmissão de serviços móveis de voz e dados" (GSM World, 2013).

O sistema GSM é capaz de realizar chamadas de voz, fazer transferência de dados a velocidades de até 9,6 kbps e, ainda, transmitir SMS.

A rede GSM começou a ser projetada na Europa em 1982 e hoje é o padrão mais popular de telefonia móvel no mundo. As redes GSM terrestres já cobrem mais de 90% da população mundial e está presente em até 129 países.

O GSM utiliza duas bandas para comunicação duplex. Cada banda tem uma largura de 25 MHz e se divide em 124 canais de 200 kHz. Cada canal de voz é digitalizado e comprimido para um sinal digital de 13 kbps. No GSM, o sinal e os canais de voz são digitais.

"O sistema GSM fez a passagem da tecnologia analógica, usada até então nos vários sistemas de funcionamento, para a tecnologia digital, com os ganhos inerentes em fiabilidade, segurança e robustez." (SÁ, 2007, p.199).

Com o sistema GSM é possível que as operadoras de telefonia celular façam restrições em aparelhos que elas mesmas vendem para uso em sua própria rede. É o chamado

bloqueio de aparelho e é implementado por uma característica do *software* do telefone. No Brasil, as operadoras são obrigadas a vender os aparelhos desbloqueados.

2.8 SMS e SIM Card

O SMS é um recurso dos telefones celulares. Surgido em 1992, revolucionou o mercado de telefonia móvel ao permitir que duas pessoas se comunicassem por mensagem através de seus celulares.

Este recurso passou a ser muito utilizado nos celulares. Segundo NIELSEN (2008):

> Durante o segundo trimestre de 2008, um assinante móvel típico EUA efetivou ou recebeu 204 chamadas por mês. Em comparação, o cliente médio móvel enviou ou recebeu 357 mensagens de texto por mês - um aumento de 450% em relação ao número de mensagens de texto circulou mensalmente, durante o mesmo período em 2006.

Por se tratar de um serviço de mensagem curta (*Short Message Service*), o SMS aceita que sejam utilizados somente até 160 caracteres. No Brasil, é divulgado pelas empresas de telefonia móvel como serviço de "torpedo", tendo recebido este nome devido à alta velocidade com que a mensagem era recebida pelo destinatário.

O *SIM Card*, também conhecido como cartão SIM, ou *chip* do celular, "é um *chip* de memória portátil usado principalmente em telefones celulares que operam em redes GSM. Estes cartões SIM possuem memória" (SANTOS, 2012).

Trata-se de um cartão de plástico que tem um *smart card* impresso junto com o número ID (identificador). O ID do cartão SIM é único em todo o mundo e é chamado de ICCID (*International Circuit Card Identification*).

O cartão SIM é um microcontrolador e sua função básica é autenticar o cliente, o que é feito através do código PIN (*Personal Identification Number*) do cartão.

Segundo Lorenzoni (2006, p. 60):

> A memória do cartão SIM é do tipo EEPROM e nela ficam armazenados não só o número de telefone ID único e universal, mas todas as configurações de dados, como informações do assinante, agenda de contatos, preferências, configurações, serviços contratados, SMS e outras informações. Além disso, a troca da linha telefônica para um novo aparelho hoje é simples, basta colocar o SIM *Card* em uso no novo celular comprado.

2.9 Pinagem

A pinagem utilizada pelo sistema está inserida no código fonte, como veremos no capítulo 3.

Toda a pinagem aqui descrita é considerada de fundamental importância para que o protótipo funcione corretamente.

Para o Arduino, será utilizada a pinagem mostrada no Quadro 2.4.

Quadro 2.4 - Pinagem no Arduino

PINO	**USO**
5V	Alimentação do Sensor PIR
GND	Aterramento do Sensor PIR
8	Entrada digital do Sensor PIR
13	Saída digital do LED
GND	Aterramento do LED
11	Saída digital do *buzzer*
GND	Aterramento do *buzzer*
Vin	Alimentação do Módulo GSM

Fonte: Autor.

Já os Quadros 2.5 e 2.6, apresentam a pinagem do *Cellular Shield*, onde VDD* = 3,0V. Os *jumpers* da placa devem ser conectados em D2 (TXD) e D3 (RXD).

Quadro 2.5 - Pinagem do Módulo GSM.

Interface	Pinagem	Descrição
GPIOs	1	VDD*
	2	GND
	3	GPIO1
	4	GPIO2
	5	GPIO3
	6	GPIO4
	7	GPIO5
	8	GPIO6
	9	GPIO7
	10	GPIO8
	11	GPIO9
	12	GPIO10
	13	GPIO11
	14	GPIO12
	15	GND
	16	VDD*
UART	1	GND
	2	DTR
	3	RI
	4	DCD
	5	CTS
	6	RTS
	7	TXD
	8	RXD

Fonte: http://imall.iteadstudio.com/im120417009.html

Quadro 2.6 – Continuação da Pinagem do Módulo GSM.

Interface	Pinagem	Descrição
IIC&PWM	1	GND
	2	GND
	3	IIC_SCL
	4	IIC_SDA
	5	PWM2
	6	PWM1
	7	GND
	8	GND
Debug_Port	1	GND
	2	PERKEY
	3	DBG_RXD
	4	DBG_TXD

Fonte: http://imall.iteadstudio.com/im120417009.html

Na Figura 2.7, pode ser vista a pinagem do Sensor PIR.

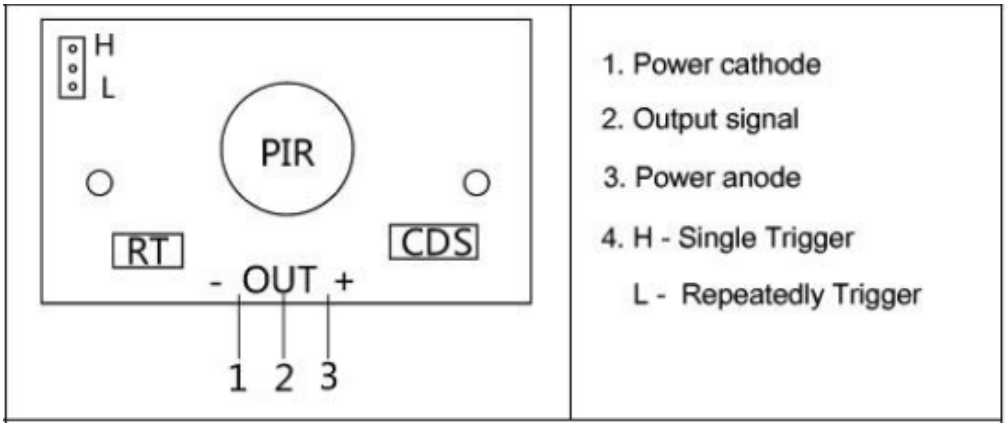


Figura 2.7 - Pinagem do Sensor PIR. (Fonte: http://www.huinfinito.com.br/attachment.php?id_attachment=58)

Onde:

1) Alimentação do Sensor PIR;

2) Saída do Sensor PIR;

3) Aterramento do Sensor PIR.

CAPÍTULO 3 – DESENVOLVIMENTO DO PROJETO

Neste capítulo, será descrito como é feito o desenvolvimento do projeto, trazendo diagramas, *softwares*, testes, etc.

3.1 Desenvolvimento do Projeto

O desenvolvimento do projeto deu-se de forma a seguir as etapas descritas no primeiro capítulo desta monografia. Primeiramente foram feitas pesquisas sobre os temas relevantes ao projeto, como demonstrado no capítulo anterior.

Logo após, definiu-se o tipo de modelo o qual seria utilizado, no caso, optou-se pelo montagem de um protótipo baseado em Arduino UNO, tendo em vista atender os objetivos de simplicidade e fácil manuseio do protótipo, obedecendo o fluxograma mostrado na Figura 3.1.

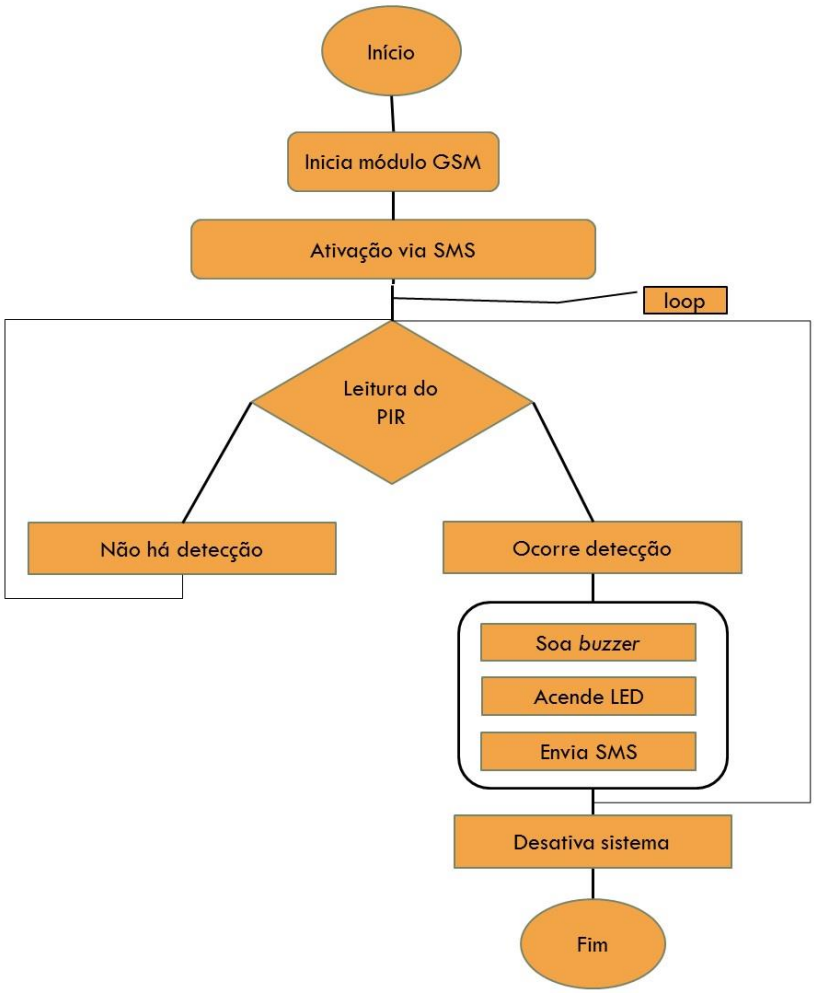


Figura 3.1 - Fluxograma do Modelo Proposto. (Fonte: Autor)

O dispositivo funciona da seguinte forma: Ao ser ligado o dispositivo, o microcontrolador (por meio do código implementado, o qual será detalhado mais adiante no item 3.4) deverá inicializar o módulo GSM, após a inicialização do módulo o sistema aguarda o recebimento de um SMS para que ative o sistema, entrando então em um loop de leitura do sensor de forma que ao identificar (detectar) movimento, o mesmo deverá acender o LED, soar o *buzzer* e enviar o SMS, tudo simultaneamente. Após estes procedimentos o dispositivo realiza nova leitura do sensor. Caso na leitura não seja detectado movimento, o dispositivo não deverá fazer nada e realizar nova leitura do sensor. O dispositivo também poderá ser desativado pelo próprio usuário via SMS, após o recebimento de SMS informando que o sistema fez detecção de movimento. O dispositivo tem um tempo de espera (*delay*) de 5 segundos a cada leitura.

Para que funcione de tal forma, é necessário que obedeça ao esquema do diagrama de *hardware* mostrado na Figura 3.2.

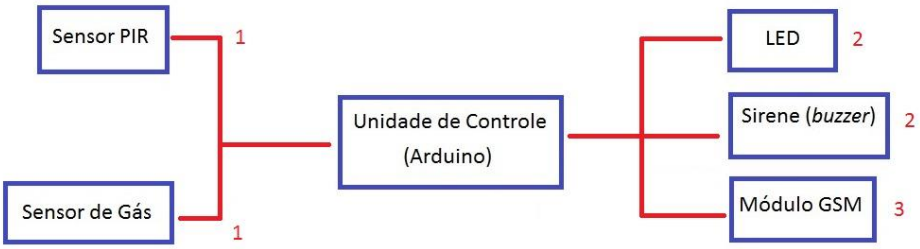


Figura 3.2 – Diagrama de blocos de *hardware*. (Fonte: Autor)

O diagrama de blocos acima descreve o funcionamento do sistema. O sensor PIR informa à unidade de controle que detectou algum movimento. Ao receber tal informação, a unidade de controle liga o LED e a sirene e, logo após, envia um SMS por meio do módulo GSM.

3.2 Sistema Proposto

O protótipo é montado de forma que sejam necessários poucos componentes, visando à simplificação da construção do mesmo. Optou-se pela utilização do Arduíno por ser de fácil aprendizagem com relação a sua programação e por ser possível acoplar uma *shield*, desenvolvida especificamente para se trabalhar com Arduino, capaz de realizar todas as funções GSM necessárias ao projeto.

A princípio, a modelagem seria feita utilizando-se outros componentes (3 microcontroladores PIC, Tela de LCD e Teclado Matricial), porém estes esbarraram nos objetivos de simplicidade do projeto, o que fez com que se optasse por uma modelagem diferente.

Definiu-se então que, para a montagem do projeto, seriam utilizados os seguintes componentes:

- Arduino UNO
  Definiu-se que o Arduino UNO seria a plataforma base para a construção do protótipo.

- GSM *Shield*

  O módulo GSM é o facilitador definido para a comunicação que deverá ser provida pelo dispositivo. Ele será o responsável pelo envio das mensagens SMS ao usuário.

- Sensor de Movimento (PIR)

  Neste projeto, o sensor responsável pela detecção do movimento é o sensor PIR. Este dispositivo pode ser facilmente encontrado no mercado. É o mais comum para este tipo de aplicações.

- *Buzzer*

  Também conhecido como sirene, está no projeto para que faça barulho ao ser acionado o dispositivo, de forma a chamar a atenção do invasor e tentar fazê-lo desistir do delito.

- LED

  O LED tem a mesma função que a sirene. Porém, ao invés de emitir som, acende luz de forma a também chamar a atenção do invasor. No protótipo, o LED tem apenas uma função demonstrativa. Na residência a ideia seria fazer com que as luzes da casa se acendessem ou que se acendessem outras luzes específicas e várias para executar a função de impedir que o invasor prossiga no cometimento do delito. O LED tem outra função além de sinalizar a intrusão, esta função é mostrar que o dispositivo está funcionando.

- Fios de cobre

  Os fios de cobre foram utilizados para fazer as ligações entre os componentes.

- *Jumpers*

  Já os *jumpers*, foram utilizados para permitir a conexão dos componentes.

Todos estes componentes são interligados e se comunicam conforme mostrado no diagrama de blocos na Figura 3.2.

3.3 Montagem do Protótipo

Após a decisão dos componentes a serem utilizados, foi feita a montagem do protótipo. A montagem correu sem problemas. A Figura 3.3 ilustra a montagem final do protótipo.

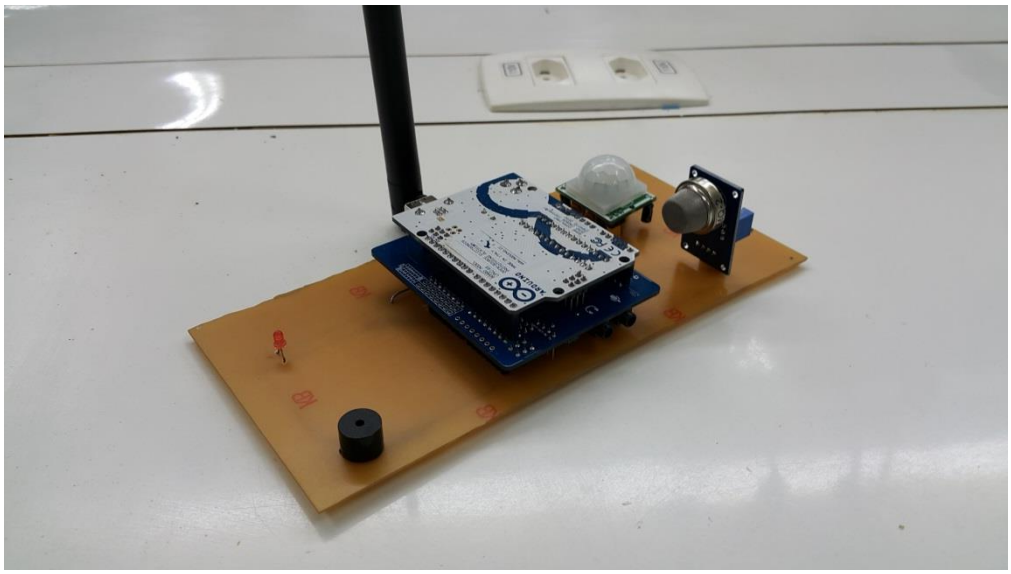


Figura 3.3 - Montagem Final do Protótipo. (Fonte: Autor)

Na montagem do modelo, foi utilizado o cabo USB para a alimentação do sistema. O *Shield* tem encaixe perfeito com a placa do Arduino UNO. Já o sensor PIR está conectado por meio de cabos e dos conectores encaixados no sensor.

Para se chegar a esta solução, primeiramente foi feita uma montagem no Proteus. O Proteus é um *software* no qual se pode desenhar placas de circuito impresso antes de se montar, de fato, um protótipo. Ou seja, o Proteus permite simular os circuitos eletrônicos a serem implementados. As Figuras 3.4 e 3.5 mostram a montagem feita no Proteus para o desenvolvimento do protótipo:

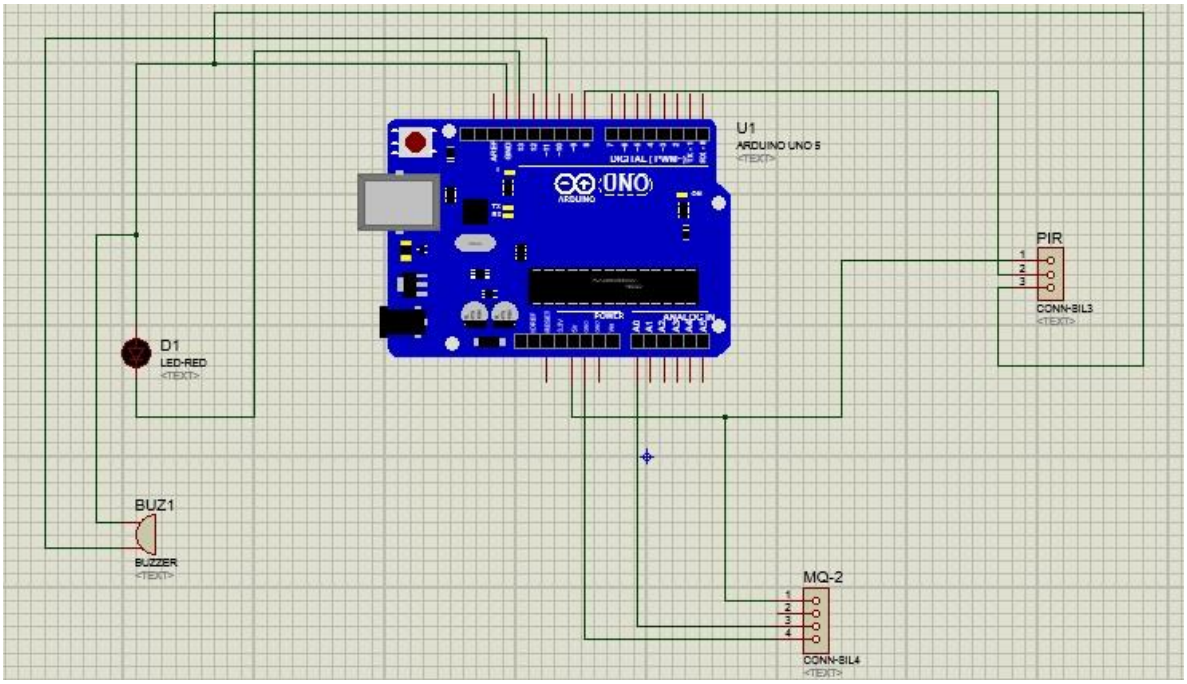


Figura 3.4 – Montagem do modelo no ISIS Proteus. (Fonte: Autor)

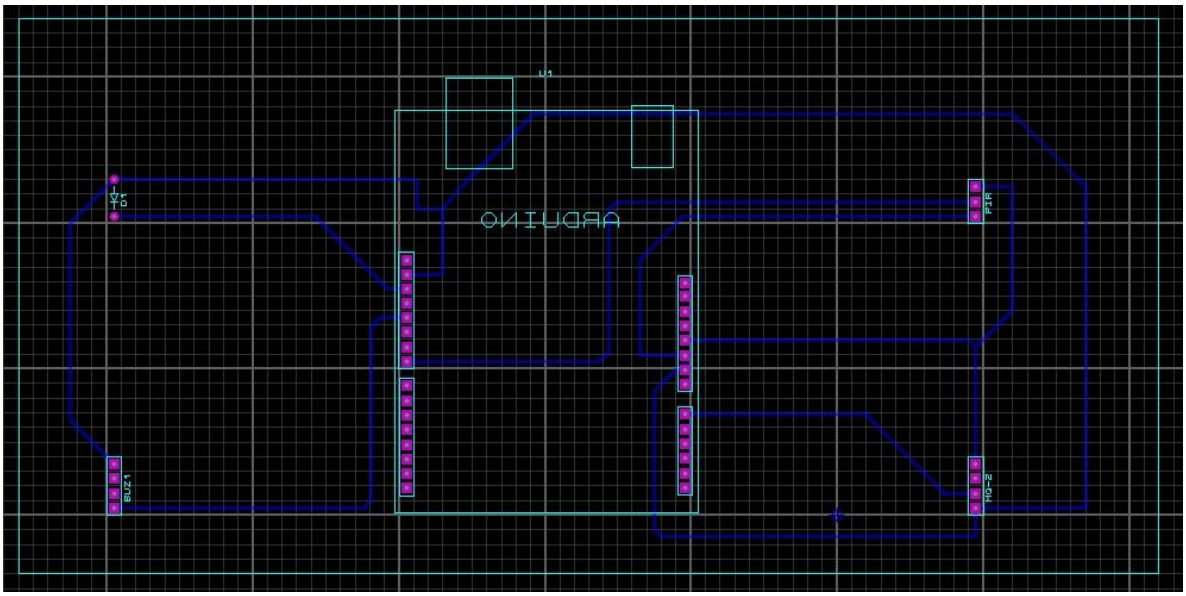


Figura 3.5 – Montagem do modelo no ARES Proteus. (Fonte: Autor)

3.4 Código-Fonte

Para facilitar a explicação do código, faremos algumas divisões no mesmo. O código, na íntegra, encontra-se no Anexo C.

No código-fonte, ocorreram vários problemas relacionados à biblioteca. Nas bibliotecas encontram-se os comandos AT necessários para que sejam enviados os SMS. Como existem duas bibliotecas chamadas “GSM.h”, teve-se certa dificuldade em descobrir qual seria a mais apropriada para uso no projeto.

Na primeira parte, são mostradas as bibliotecas, definições, pinagens e variáveis do código fonte:

```
#include <GSM.h>

#define PINNUMBER ""


GSM gsmAccess;

GSM_SMS sms;


boolean notConnected = true;


int LRed = 13;

int pinSpeaker = 11;

int PinPIR = 8 ;

int PinGAS = 0;

int valGAS;

boolean pirState = HIGH;

boolean val = false;

boolean desliga = true;

int count=0;

char senderNumber[20];
```

```
char c;
```

Entre as definições, está o código PIN do *SIM Card*, como no caso do projeto o reconhecimento do PIN está desativado, não é necessário que se insira o PIN no programa. Entre as variáveis pode ser encontradas as pinagens do LED (pino 13), do *buzzer* (pino 11), do sensor PIR (pino 8) e do sensor de gás (pino analógico 0).

Também estão definidos os valores iniciais dos booleanos “notConnected” (variável que indica a conexão ao módulo GSM) como verdadeiro, “val” (variável que indica o acionamento do alarme) como falso, “desliga” (variável que indica a desativação – por meio de SMS - do alarme) como verdadeiro e “pirState” (variável que define o estado do sensor PIR) como alto.

Além disto, estão definidas as variáveis inteiro ”valGAS” (armazana o valor lido pelo sensor de gás), inteiro “count” (um contador criado para contabilizar as tentativas de conexão ao módulo GSM), caractere “sendNumber” (vetor onde é armazenado o número do qual foi recebido o SMS) e caractere “c” (para montar a mensagem que chegou via SMS).

Já a segunda parte ilustra a função “setup” do programa, que é a função que inicializa o sistema.

```
void setup () {
    pinMode(LRed, OUTPUT);
    pinMode(PinPIR, INPUT);
    pinMode(pinSpeaker,OUTPUT);
    Serial.begin(9600);
    Serial.println ("Sistema inicializado!");
    while (notConnected) {
        if (gsmAccess.begin (PINNUMBER)==GSM_READY) {
            notConnected = false;
            Serial.println ("Conectado ao sistema GSM!");
            count=0;
        }
        else {
            Serial.println ("Nao conectado");
```

```
            count++;
            Serial.print ("Tentativa ");
            Serial.print (count);
            Serial.println (" falhou.");
            delay (500);
        }
    }
    Serial.println ("GSM inicializado!\n");
}
```

Nesta função, define-se os modos da pinagem (entrada ou saída), o sistema inicializa a porta serial e inicia o GSM *Shield*. Caso a conexão com o módulo GSM ocorra sem problemas, o sistema imprime na porta Serial a frase “Conectado ao sistema GSM!” e em seguida imprime “GSM inicializado!”. Caso contrário, o sistema imprime “Não conectado”, informando assim que a conexão ainda não aconteceu e contabiliza o número de tentativas, imprimindo-as na porta serial.

Na terceira parte, temos a função “loop” da programação. Nesta função devemos colocar tudo aquilo que deverá acontecer sempre que o sistema for acionado.

```
void loop () {
    Receive_loop ();
    while (val) {
        digitalWrite(LRed, HIGH);
        SMS_loop ();
        playTone(100,160);
        delay (5000);
        Serial.println("Movimento detectado!");
    }
    if (desliga) {
        Serial.println("Alarme desativado via SMS!");
```

```
        }
    else {
        Serial.println("Esta tudo ok.");
        delay (1000);
        }
}
```

A função loop, aciona a função de loop de recebimento (veremos esta função mais a frente) caso esta função retorne um valor verdadeiro para “val”, faz a leitura do sinal de saída do sensor PIR e quando o sinal lido é igual a alto, chama as funções “SMS_loop” (função de loop para envio do SMS) e “playTone” (função responsável por fazer o *buzzer* soar).

Além das funcionalidades descritas, o sensor possui um *delay* (atraso) de 5 segundos para fazer uma nova leitura e informar que o movimento foi detectado. Além disso, a função determina que caso o valor de “val” seja verdadeiro e o valor de “desliga” seja falso, aciona o alarme. Caso “val” seja falso e “desliga” também o seja, significa que não há movimento na propriedade. Sempre que “desliga” for verdadeiro, significa dizer que o sistema está desativado.

A quarta parte ilustra a função “SMS_loop”. Esta função é responsável por fazer o envio do SMS.

```
void SMS_loop () {
    sms.beginSMS("+556191055994");
    Serial.println ("Mensagem enviada ao numero: +556191055994");
    sms.print("Alerta! Propriedade invadida! Disque: 190.");
    sms.endSMS();
    Serial.println("\n COMPLETE! \n");
}
```

A função faz a leitura do número para o qual deve mandar a mensagem, que é pré-determinada no código, e envia uma mensagem predefinida ao usuário, imprimindo na porta Serial que a mensagem foi enviada para o número escrito e, após o envio, finaliza o envio e imprime uma mensagem de que o envio foi completado.

A quinta parte ilustra a função “Receive_loop”. Esta função lê as mensagens recebidas e descarta as que não interessam ao sistema.

```
void Receive_loop () {
    if (sms.available()) {
        Serial.println("Mensagem recebida de:");
        sms.remoteNumber(senderNumber, 20);
        Serial.println(senderNumber);
        if (sms.peek()!='#' && sms.peek()!='*') {
            Serial.println("SMS descartado!");
            sms.flush();
        }
        else {
            if (sms.peek()=='#') {
              val = digitalRead(PinPIR);
              GAS_loop ();
              desliga = false;
              Serial.println("Alarme ativado!");
            }
            else {
               if (sms.peek()=='*') {
                 val = false;
                 desliga = true;
                 Serial.println("Alarme desativado!");
               }
            }
        }
        Serial.print("Mensagem: ");
```

```
        while (c=sms.read()) {
            Serial.print(c);
        }
        Serial.println("\nFinal da Mensagem");
        sms.flush();
        Serial.println("Mensagem deletada!\n");
    }
    delay(500);
}
```

Esta função lê as mensagens recebidas e descarta as que não interessam ao sistema.

A função inicia verificando se há SMS disponível. Sempre que há mensagem, o sistema informa na porta serial o seu recebimento, lendo e mostrando o número do qual a mensagem foi recebida, e, ao final, imprime o SMS na porta serial após se fazer o tratamento da mensagem (exceto nos casos em que as mensagens não tenham conteúdo interessante ao sistema – ativação ou desativação do alarme).

Logo após imprimir o número do qual recebeu o SMS, a função faz o tratamento da mensagem, decidindo, então, o que será feito com a partir do recebimento da mensagem.

Para fazer tal tratamento, a função verifica os caracteres iniciais da mensagem recebida e decide o que fazer da seguinte forma: as mensagens que iniciam com o caractere "#" ativam o alarme, enquanto as mensagens que iniciam com o caractere "*" desativam o alarme. Qualquer mensagem que não iniciar com um destes dois caracteres, não interessa ao sistema.

Sempre que o alarme é ativado, o sistema faz a leitura do sensor PIR, inicia a função "GAS_loop" (que será descrita logo a seguir), torna falso o booleano "desliga" (o que faz com que o sistema passe a ficar ativo) e imprime na porta serial que o alarme está ativado.

Sempre que recebe o comando de desativação, ele torna o booleano "desliga" verdadeiro e o mesmo ocorre com o booleano "val", ou seja, o alarme está desativado e não fará leitura do sensor PIR. Para identificar isto, o sistema imprime na porta serial "Alarme desativado!".

A sexta parte é a função "GAS_loop". Esta função é responsável por fazer a detecção do gás.

```
void GAS_loop() {
```

```
    valGAS = analogRead(PinGAS);
    Serial.print("Leitura do Sensor = " );
    Serial.println(valGAS);
    if (valGAS > 300) {
      digitalWrite (LRed, HIGH);
      playTone(100,160);
      sms.beginSMS("+556191055994");
      Serial.println ("Mensagem enviada ao numero: +556191055994");
      sms.print("Alerta! Gás detectado! Disque: 193.");
      sms.endSMS();
      Serial.println("\n GAS COMPLETE! \n");
    }
    delay(1000);
}
```

A função faz a leitura do valor indicado, analogicamente, pela variável “PinGAS” e recebida pela variável “valGAS”. Após, imprime na porta serial o valor (em partes por milhão) armazenado nesta última variável.

Sempre que este valor for maior do que 300 ppm, o sistema acende o LED, faz soar o *buzzer* (chamando a função “playTone”) e envia um SMS, com mensagem predefinida e usuário previamente determinado. Após, imprime na porta serial que o aviso do gás foi completado. Esta função possui um *delay* de 1 segundo entra as leituras.

Por fim, a última parte traz a função “playTone”. Esta função determina a frequência e a duração do toque da sirene, respectivamente.

```
void playTone(long duration, int freq) {
    duration *=1000;
    int period = (1.0/freq)*1000000;
    long elapsed_time = 0;
    while (elapsed_time < duration) {
```

```
            digitalWrite (pinSpeaker,HIGH);
            delayMicroseconds(period/2);
            digitalWrite(pinSpeaker,LOW);
            delayMicroseconds(period/2);
            elapsed_time += (period);
        }
    }
```

Esta função recebe os valores de duração e frequência, ao ser chamada por outra função.

Para a implementação, deve ser feito o carregamento do código fonte no sistema e colocá-lo para rodar.

3.5 Implementação

Como foi escolhido o Arduino UNO como plataforma de montagem, a implementação se torna muito mais simples e fácil. Basta conectar o Arduino ao computador, via cabo USB e fazer o *upload* (a carga) do código no mesmo.

A partir deste momento o sistema executara as funções conforme o fluxograma apresentado no início do capítulo.

CAPÍTULO 4 – TESTES

Este capítulo tratará dos resultados obtidos e dos testes feitos neste projeto.

4.1 Introdução

Para que seja possível fazer os testes de maneira mais eficaz e de melhor visualização, dividem-se os testes em cenários. O Diagrama de Blocos da Figura 4.1 ilustra esta divisão.

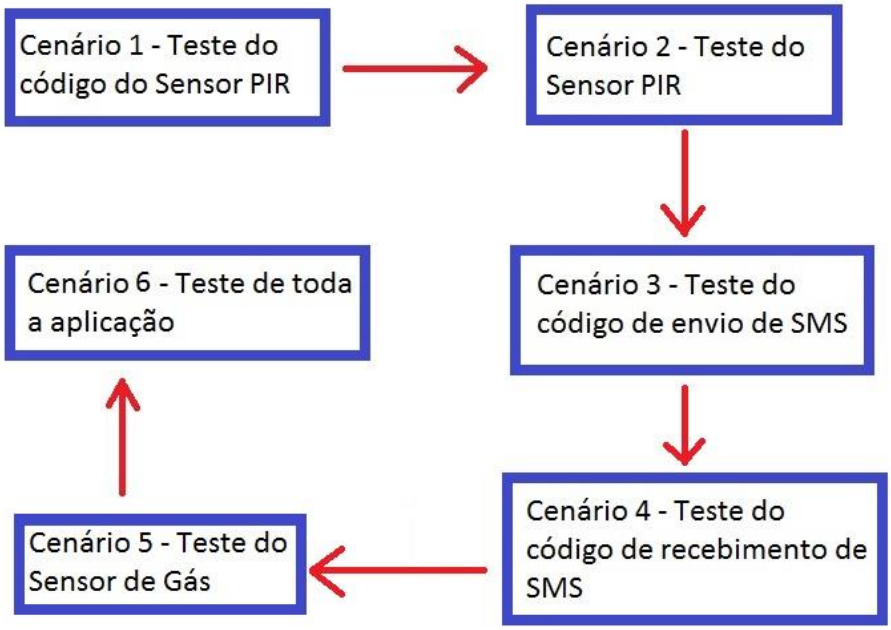


Figura 4.1 – Diagrama de Blocos dos Cenários de Teste

Cada um dos cenários do diagrama representa um teste feito para assegurar o funcionamento do bloco.

Após o desenvolvimento do código, deu-se início a fase de testes. Nesta etapa, dividiu-se o código para que se pudesse testá-lo separadamente e encontrar de forma mais fácil o erro, caso houvesse erro.

4.2 Cenário 1 – Teste do código do Sensor PIR

Este cenário tem como objetivo fazer com que o sensor PIR funcione corretamente, verificando se a lógica e a programação escritas estão corretas, ou seja, verificar se existe algum erro na programação feita para detectar movimento.

Primeiramente fez-se o teste do código do sensor de movimento. Para tal, fez-se a montagem apenas do Arduino com o Sensor PIR, o LED e o *buzzer*. Utilizando-se o seguinte código:

```
int LRed = 13;
int PinPIR = 8 ;
boolean pirState = HIGH;
boolean val;
int pinSpeaker = 11;

void setup () {
    pinMode(LRed, OUTPUT);
    pinMode(PinPIR, INPUT);
    pinMode(pinSpeaker,OUTPUT);
    Serial.begin(9600);
}

void loop () {
    val = digitalRead(PinPIR);
    if (val) { //TENTAR substituir por while
       digitalWrite(LRed, HIGH);
       playTone(100,160);
       delay (5000);
       digitalWrite(LRed, LOW);
       Serial.println("Movimento detectado!");
    }
    else {
        Serial.println("Esta tudo ok.");
    }

}

void playTone(long duration, int freq) {
```

```
    duration *=1000;
    int period = (1.0/freq)*1000000;
    long elapsed_time = 0;
    while (elapsed_time < duration) {
        digitalWrite (pinSpeaker,HIGH);
        delayMicroseconds(period/2);
        digitalWrite(pinSpeaker,LOW);
        delayMicroseconds(period/2);
        elapsed_time += (period);
    }
}
```

No início o dispositivo não reconhecia os movimentos de forma correta. O sensor identificava movimento mesmo quando não havia. Chegou-se a esta conclusão após a realização de testes com o osciloscópio. O osciloscópio provou que o sensor detectava sinal mesmo não havendo movimento. Era certo que não havia movimento pelo fato de que o sensor estava dentro de uma caixa fechada, em local escuro.

4.3 Cenário 2 – Teste do Sensor PIR

Este cenário tem como objetivo fazer com que o sensor funcione corretamente, analisando o *hardware*, ou seja, analisar quais condições fazem com que o dispositivo acione ou não.

Neste novo cenário, percebeu-se que o que estava ocorrendo era que o sensor estava ajustado com um alto nível de sensibilidade. Após o ajuste do nível de sensibilidade do sensor, o sensor passou a comportar-se de forma esperada, reconhecendo movimento somente quando realmente havia movimento.

Neste projeto o sensor está ajustado para alcançar uma distância de aproximadamente 5,6 metros e uma amplitude de 140º. Para se chegar a estes valores, foi preciso colocar o sensor no chão e aos poucos ir se aproximando do sensor até que o mesmo detectasse o movimento. Encontrado o local onde o movimento foi detectado, fez-se, com a ajuda de uma trena, a medição da distância do sensor até o local, como demonstrado na Figura 4.2.

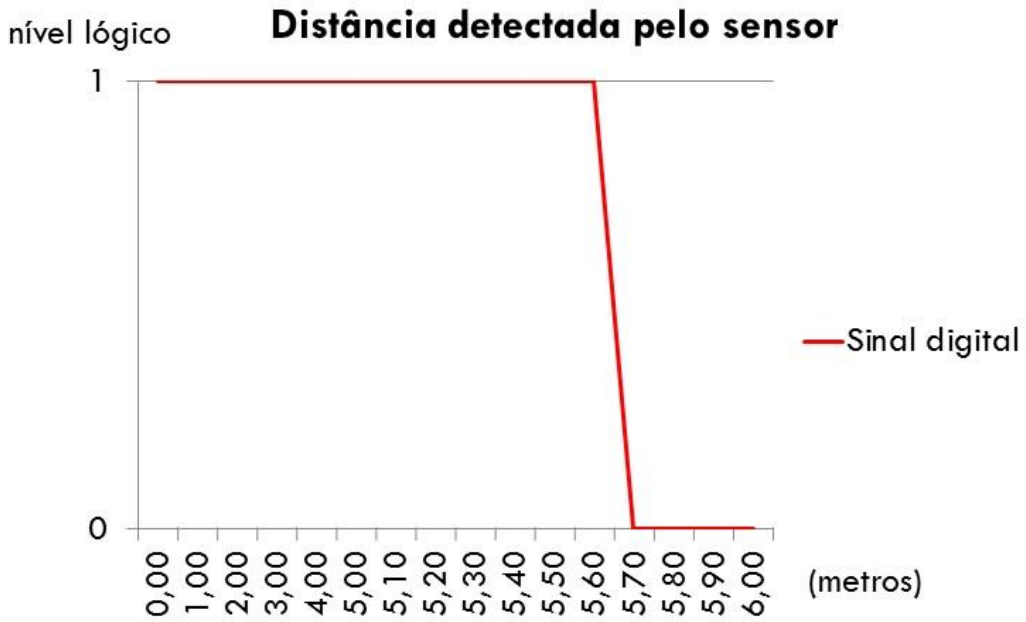


Figura 4.2 – Gráfico da distância detectada pelo Sensor PIR.

Já para encontrar a amplitude do raio de ação, foi colocado um transferidor abaixo do sensor e então foi possível, aproximando-se por trás do sensor, encontrar a angulação captada pelo sensor, como mostrado na Figura 4.3.

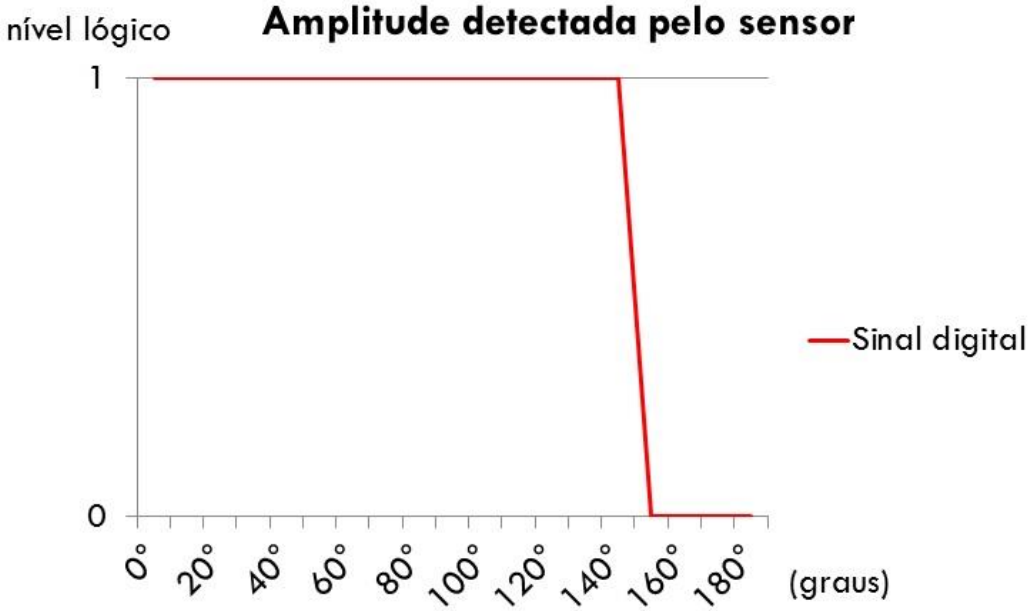


Figura 4.3 – Gráfico da amplitude detectada pelo Sensor PIR.

4.4 Cenário 3 – Teste do código de envio de SMS

Este cenário tem como objetivo fazer com que o módulo GSM funcione corretamente, verificando se a lógica e a programação escritas estão corretas, ou seja, verificar se existe algum erro na programação feita para o envio de SMS.

Testada a parte do sensor, fez-se então o teste do envio do SMS. Para tal, fez-se o *upload* do código-fonte no arduino, com o módulo GSM acoplado, e esperado o recebimento do SMS no número de telefone definido no código.

Utilizou-se o código-fonte a seguir:

```
#include <GSM.h>
#define PINNUMBER ""

GSM gsmAccess;
GSM_SMS sms;

boolean notConnected = true;
int c=0;
void setup () {
    Serial.begin (9600);
    Serial.println ("SMS Messages Sender");
    while (notConnected) {
        if (gsmAccess.begin (PINNUMBER)==GSM_READY) {
            notConnected = false;
            Serial.println ("Conectado ao sistema GSM!");
            c=0;
        }
        else {
            Serial.println ("Nao conectado");
            c++;
            Serial.print ("Tentativa ");
            Serial.print (c);
            Serial.println (" falhou.");
            delay (500);
```

```
            }
        }
        Serial.println ("GSM inicializado!");
}

void loop () {
        sms.beginSMS("06191055994");
        Serial.println ("Numero do destinatario: 6191055994");
        sms.print("Alerta! Propriedade invadida! Disque: 190.");
        Serial.println ("Mensagem enviada!");
        sms.endSMS();
        Serial.println("\n COMPLETE! \n");
}

int readSerial (char result[]) {
    int i=0;
    while (1) {
            while (Serial.available () > 0) {
                char inChar = Serial.read();
                if (inChar == '\n') {
                    result[i] = '\0';
                    Serial.flush();
                    return 0;
                }
                if (inChar !='\r') {
                    result[i] = inChar;
                    i++;
                }
            }
    }
}
```

A conexão do módulo GSM com o Arduino também apresentou problemas. O módulo possuía uma “gambiarra” que tinha a finalidade de não ter que dar *reset* na placa GSM toda vez que o dispositivo fosse ligado. Mas para que isto de fato funcionasse, era necessário que fosse criado o *reset* no código-fonte. Após perceber-se que a “gambiarra” no módulo estava impedido a inicialização do módulo, retirou-se a “gambiarra”. A partir deste momento o dispositivo passou a fazer a conexão de forma correta, porém precisando-se pressionar a tecla *reset* do módulo sempre que for ligado.

Outro problema encontrado foi o posicionamento dos *jumpers* no módulo GSM. Estava-se conectando-os em D5 (TXD) e D4 (RXD) quando na verdade era necessário que os conectasse em D2 (TXD) e D3 (RXD). Feito isso, o dispositivo passou a funcionar como se esperava, reconhecendo movimentos, ativando o *buzzer* e o LED e enviando SMS para o usuário.

4.5 Cenário 4 – Teste do código de recebimento de SMS

Este cenário tem como objetivo fazer com que o módulo GSM funcione corretamente, verificando se a lógica e a programação escritas estão corretas, ou seja, verificar se existe algum erro na programação feita para recebimento de SMS.

Testado o envio, testou-se o recebimento do SMS. Para que isto fosse possível, fez-se o *upload* do código-fonte e, com o uso de um celular, foram enviadas mensagens contendo (e não contendo) os caracteres de tratamento (“#” e “*”).

Utilizou-se o seguinte código:

```
#include <GSM.h>
#define PINNUMBER ""

GSM gsmAccess;
GSM_SMS sms;

boolean notConnected = true;

int LRed = 13;
int PinPIR = 8 ;
boolean pirState = HIGH;
boolean val = false;
int pinSpeaker = 11;
char senderNumber[20];
```

```
int count=0;

void setup() {
    Serial.begin(9600);
    Serial.println ("SMS Messages Receiver");
    while (notConnected) {
        if (gsmAccess.begin (PINNUMBER)==GSM_READY) {
            notConnected = false;
            Serial.println ("Conectado ao sistema GSM!");
            count=0;
        }
        else {
            Serial.println ("Nao conectado");
            count++;
            Serial.print ("Tentativa ");
            Serial.print (count);
            Serial.println (" falhou.");
            delay (500);
        }
    }
    Serial.println ("GSM inicializado!\n");
}

void loop () {
    char c;
    if (sms.available()) {
        Serial.println("Mensagem recebida de:");
        sms.remoteNumber(senderNumber, 20);
        Serial.println(senderNumber);
        if (sms.peek()!='#' && sms.peek()!='*') {
            Serial.println("SMS descartado!");
            sms.flush();
```

```
        }
        else {
           if (sms.peek()=='#')
             Serial.println("Alarme ativado!");
           else {
              if (sms.peek()=='*')
                Serial.println("Alarme desativado!");
           }
        }
        Serial.print("Mensagem: ");
        while (c=sms.read()) {
             Serial.print(c);
        }
        Serial.println("\nEND OF MESSAGE");
        sms.flush();
        Serial.println("MESSAGE DELETED\n");
     }
     delay(1000);
}
```

Nesta etapa, foram encontrados problemas com a antena do módulo GSM. A antena começou a apresentar problemas e, devido a isto, o sistema não consegui se conectar à rede GSM. Após a antena ser trocada, o problema foi resolvido e o dispositivo voltou a funcionar corretamente.

4.6 Cenário 5 – Teste do Sensor de Gás

Este cenário tem como objetivo fazer com que o sensor de gás funcione corretamente, verificando se a lógica e a programação escritas estão corretas, bem como o próprio *hardware* do sensor deve funcionar detectando se há ou não presença de gás.

Para testar a parte complementar do projeto, o Sensor de Gás, fez-se o *upload* do código-fonte e, com o auxílio de um isqueiro, permitiu-se que houvesse um vazamento de gás próximo ao sensor.

Fez-se uso do seguinte código:

```
int PinGAS = 0;
int valGAS;
int LRed = 13;
int pinSpeaker = 11;

void  setup() {
    pinMode (LRed, OUTPUT);
    pinMode(pinSpeaker,OUTPUT);
    Serial.begin(9600);
}

void loop() {
    valGAS = analogRead(PinGAS);
    Serial.print("Leitura do Sensor = " );
    Serial.println(valGAS);
    if (valGAS > 300) {
      digitalWrite (LRed, HIGH);
    }
    delay(1000);
}
```

Neste caso, pode-se encontrar uma série de resultados devido a leitura analógica feita pelo código, gerando o gráfico representado na Figura 4.4.

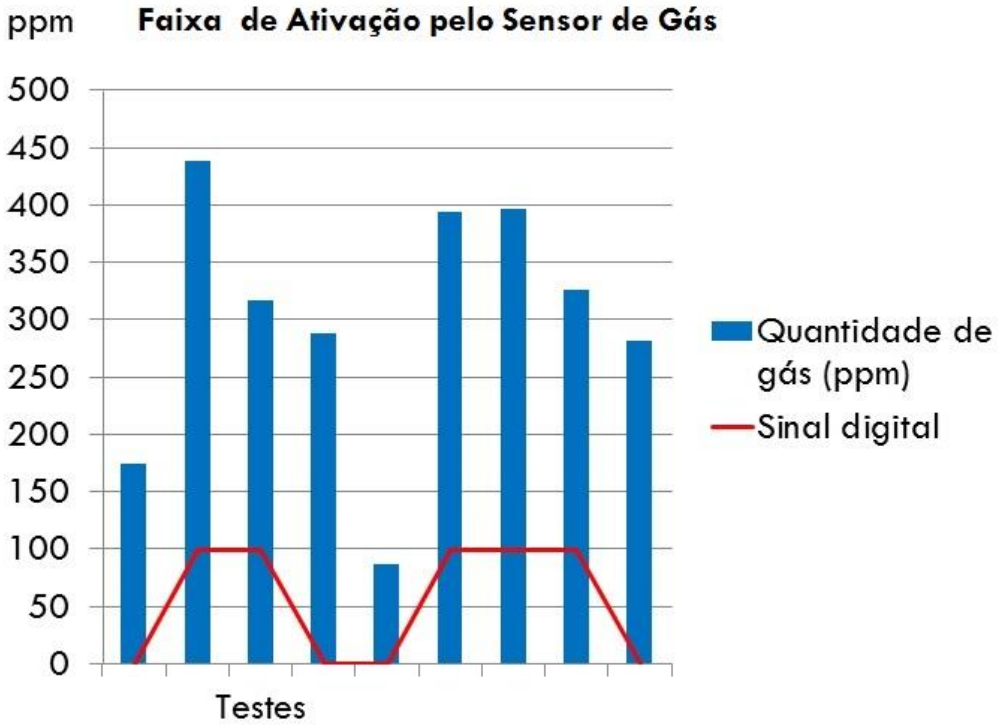


Figura 4.4 – Gráfico da faixa de ativação pelo Sensor de Gás.

Neste gráfico colocou-se uma variação de 1 a 100 para uma melhor visualização do sinal digital (acionamento ou não do alarme pelo sensor de gás).

Este gráfico ilustra que sempre que for feita uma leitura maior do que 300 partes por milhão (ppm) pelo sensor de gás, o alarme será acionado.

4.7 Cenário 6 – Teste de toda a aplicação

Este cenário tem como objetivo fazer com que a aplicação funcione corretamente, verificando, principalmente, se a lógica e a programação escritas estão corretas, ou seja, verificar se existe algum erro na programação feita para detectar movimento.

Para tal cenário, fez-se a união dos quatro códigos separados dando origem ao código-fonte descrito no capítulo 3. Durante o teste deste código correu tudo conforme o planejado, ou seja, todos os sensores e módulos funcionaram corretamente.

4.8 Resultados Obtidos

Após vários testes e correções de erros, foi possível fazer com que o dispositivo funcionasse da forma desejada, ou seja, detectando o movimento, ativando o dispositivo via

SMS, acendendo luzes e soando o *buzzer*, informando o usuário sobre a invasão de sua propriedade e, por fim, desativando o dispositivo via SMS.

Foram detectados pequenos pontos desfavoráveis no projeto, tendo em vista que em ambientes quentes o dispositivo fica muito sensível e é acionado muito facilmente.

Como pontos positivos foram observados o baixo custo, o elevado público, uma boa segurança e uma alta comodidade.

A Figura 4.5 ilustra o recebimento do SMS, trata-se de um *print screen* dado na tela do celular que recebeu o SMS.

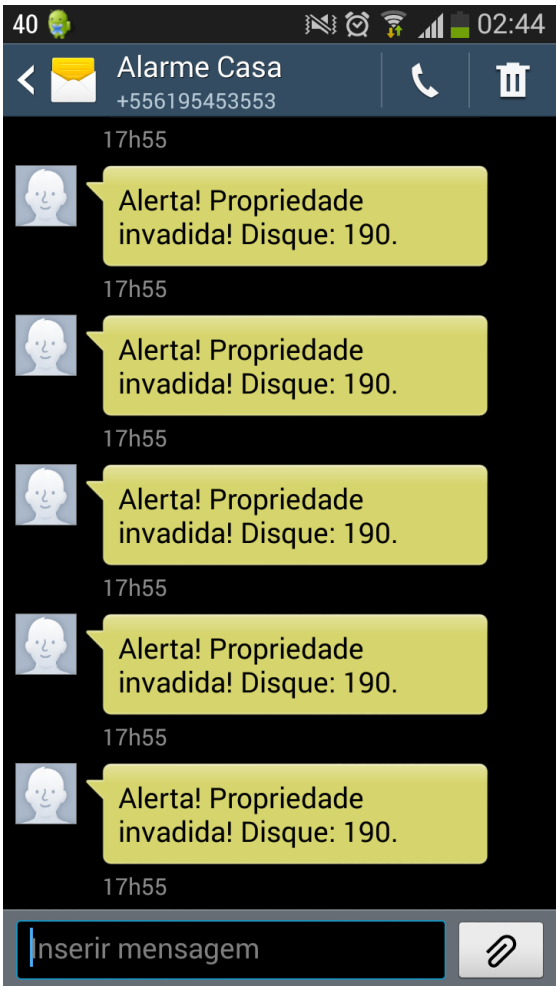


Figura 4.5 - Recebimento do SMS. (Fonte: Autor)

CAPÍTULO 5 – CONCLUSÕES E TRABALHOS FUTUROS

Este capítulo tratará das conclusões as quais se pode chegar com o projeto e sugestões de trabalhos futuros.

5.1 Conclusões

A pesquisa realizada permitiu que fosse possível alcançar os demais objetivos, tornando possível a realização da especificação, com descrição dos componentes utilizados e suas funções, da solução encontrada.

Houve o desenvolvimento de uma placa (*hardware*) para acoplamento do arduíno, nesta placa luzes e sirenes incrementam o protótipo.

A programação do microcontrolador foi feita com lógica que permitiu correto funcionamento do *hardware*. A programação do microcontrolador funcionou de forma correta, apesar de ter apresentado falhas no início do projeto.

Este projeto pode ser implementado em vários tipos de ambientes, como residências, automóveis, canteiro de obras, propriedades em geral.

O protótipo possui uma função complementar de reconhecimento de vazamento de gás na propriedade e efetivamente capaz de alertar o ocorrido, via SMS, ao proprietário.

Foi implementado um protótipo funcional capaz de executar perfeitamente suas funções, disparando o alarme quando acionado tanto pelo sensor de movimento quanto pelo sensor de gás e enviando SMS ao usuário em ambos os casos. Outra função executada perfeitamente pelo dispositivo é a ativação e desativação dos sensores via SMS.

Os objetivos deste projeto foram alcançados, pois foi criado um protótipo funcional simples e de fácil manuseio, atendendo a todas as especificações feitas, principalmente detectando movimento no local invadido e informando ao usuário sobre a invasão.

A metodologia planejada para o desenvolvimento do projeto mostrou-se correta para os objetivos fossem atingidos.

O projeto foi concluído com sucesso.

5.2 Trabalhos Futuros

Para complementação deste projeto, sugere-se que seja implementado o acionamento automático da segurança competente, seja ela pública ou privada, no momento em que for detectado movimento na propriedade.

O projeto também pode ser complementado de forma que o mesmo ative um sistema de filmagem, com tempo de duração pré-determinado pelo usuário, ao ser acionado o dispositivo.

Outra sugestão viável é a criação de um log on-line, contendo data e hora, de forma que o usuário possa saber a exata hora da intrusão, e até mesmo monitorar o sistema por meio do log.

REFERÊNCIAS

ARDUINO, Site do Projeto Arduino. **Arduino.** Disponível em: < www.arduino.cc/ >, acessado em 17/11/13.

BISHOP, Robert H.; DORF, Richard C. Sistemas de Controle. 8ª Edição. 1998. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora S.A. 2001.

FOROUZAN, B.A.; MOSHARRAF, F. *Redes de Computadores:* Uma Abordagem Top-Down. Porto Alegre: AMGH Editora Ltda, 2013.

GIMENEZ, Salvador P. *Microcontroladores 8051*. 1ª Edição. São Paulo: Person Education do Brasil Ltda., 2002.

GSM World. **GSM.** 2012. Disponível em: < http://www.gsma.com/aboutus/gsm-technology/gsm >. Acesso em: 17 nov. 2013.

LORENZONI, A. F. *Smart Cards – Java Card*. 2006. Trabalho de Conclusão de Curso (Graduação). Ciência da Computação. Centro Universitário Feevale do Rio Grande do Sul, Novo Hamburgo-RS, 2006.

MORETTO, Vanelirte. A Tecnologia Cada Vez Mais Presente Na Vida Dos Cidadãos. A Folha do Médio Norte, 2014, mar/2014. Disponível em: < http://www.afolhadomedionorte.com.br/a-tecnologia-cada-vez-mais-presente-na-vida-dos-cidadaos >. Acesso em: 15 mar. 2014.

NIELSEN WIRE. 2008. **SMS.** Disponível em: < http://www.nielsen.com/us/en/newswire/2008/in-us-text-messaging-tops-mobile-phone-calling.html >. Acesso em: 19 nov. 2013.

NISE, Norman S. *Engenharia de Sistemas de Controle*. 3ª Edição. 2000. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora S.A., 2002.

OGATA, Katsuhiko. *Engenharia de Controle Moderno*. 1970. Rio de Janeiro: Editora Prentice/Hall do Brasil Ltda. 1982.

SANTOS, J. S. *Detector de vazamento de gás com aviso por SMS*. 2012. 93 f. Monografia (Graduação). Engenharia da computação, UniCEUB, Brasília-DF, 2012.

SÁ, Rui. *Sistemas e Redes de Telecomunicações.* Lisboa: FCA – Editora de Informática Ltda., 2007.

SCHILDT, Herbert. *C++ guia para iniciantes*. Rio de Janeiro: Editora Ciência Moderna Ltda., 2002.

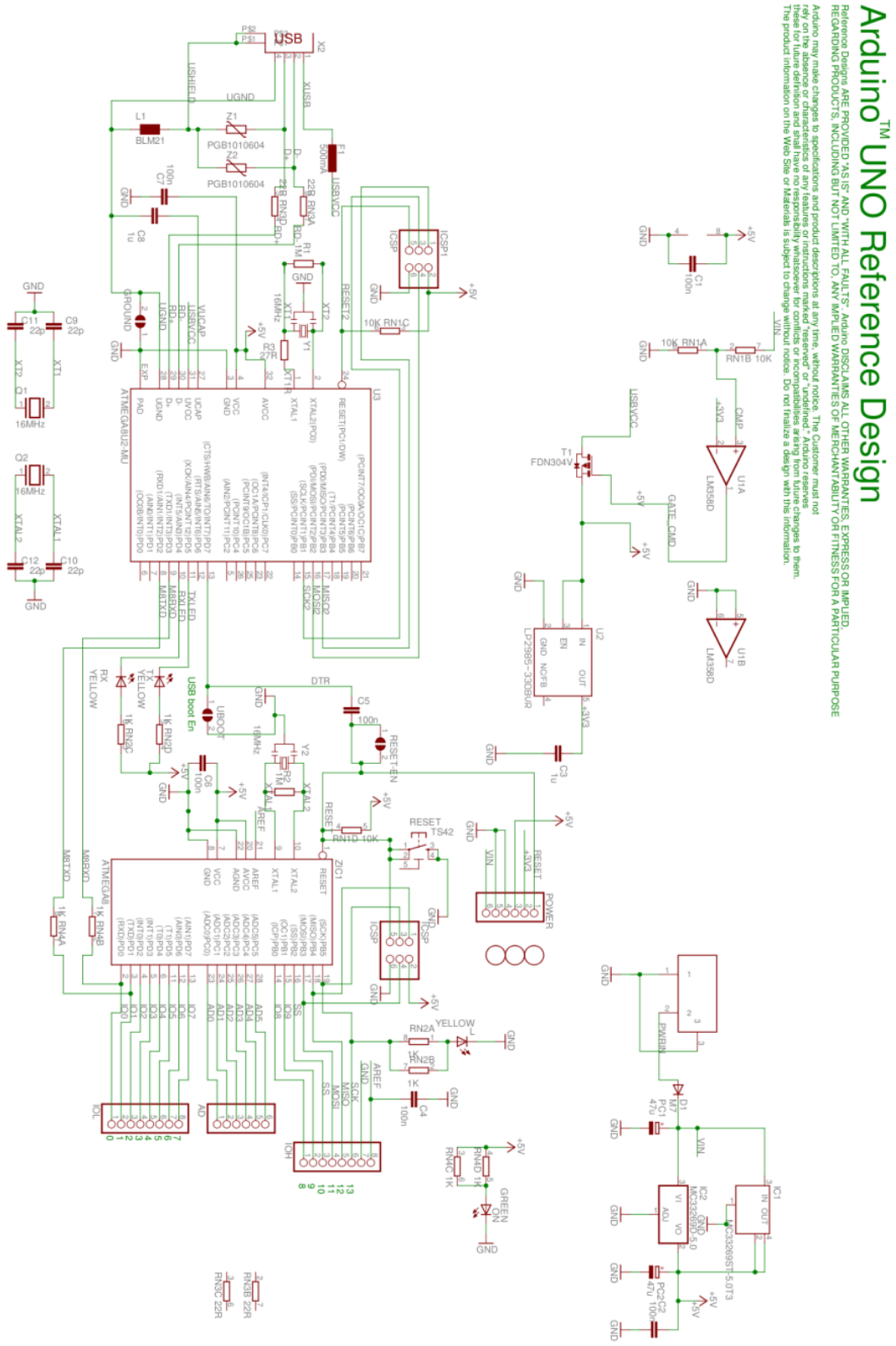
SILVA, A. V. O. *Alarme com Ativação por um Sensor Presencial e Alerta SMS*. 2012. 67 f. Monografia (Graduação). Engenharia da computação, UniCEUB, Brasília-DF, 2012.

SLIDESHARE. **Aprendendo a Programar em Arduino**. Disponível em: < http://www.slideshare.net/Miojex360/apostila-para-programar-arduino>. Acesso em: 03 nov. 2013.

THOMAZINI, D. ; ALBURQUERQUE, P. U. B. de. *Sensores Industriais:* Fundamentos e Aplicações. 1ª Edição. São Paulo: Érica Ltda., 2005.

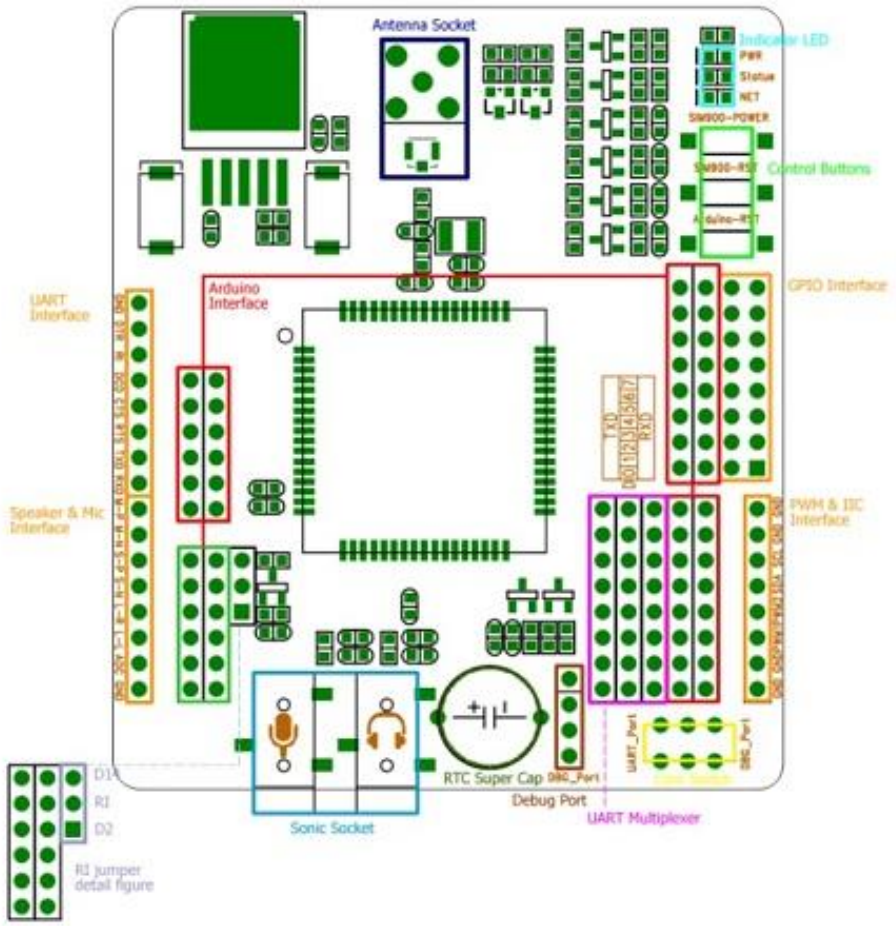
VICTORINE, V. M. *Treinamento em Linguagem C*. 2ª Edição. São Paulo: Person Education do Brasil Ltda., 2006.

ANEXO A – ESQUEMÁTICO ARDUINO UNO



Fonte: http://brittonkerin.com/annotateduino/arduino-uno-schematic.png

ANEXO B – ESQUEMÁTICO ICOMSAT 900



Fonte: http://imall.iteadstudio.com/im120417009.html

ANEXO C – CÓDIGO-FONTE COMPLETO DO PROTÓTIPO

```
#include <GSM.h>
#define PINNUMBER ""

GSM gsmAccess;
GSM_SMS sms;

boolean notConnected = true;

int LRed = 13;
int pinSpeaker = 11;
int PinPIR = 8 ;
int PinGAS = 0;
int valGAS;
boolean pirState = HIGH;
boolean val = false;
boolean desliga = true;
int count=0;
char senderNumber[20];
char c;

void setup () {
    pinMode(LRed, OUTPUT);
    pinMode(PinPIR, INPUT);
    pinMode(pinSpeaker,OUTPUT);
```

```
    Serial.begin(9600);
    Serial.println ("Sistema inicializado!");
    while (notConnected) {
        if (gsmAccess.begin (PINNUMBER)==GSM_READY) {
            notConnected = false;
            Serial.println ("Conectado ao sistema GSM!");
            count=0;
        }
        else {
            Serial.println ("Nao conectado");
            count++;
            Serial.print ("Tentativa ");
            Serial.print (count);
            Serial.println (" falhou.");
            delay (500);
        }
    }
    Serial.println ("GSM inicializado!\n");
}

void loop () {
    Receive_loop ();
    while (val) {
        digitalWrite(LRed, HIGH);
        SMS_loop ();
        playTone(100,160);
```

```
        delay (5000);
        Serial.println("Movimento detectado!");
    }
    if (desliga) {
        Serial.println("Alarme desativado via SMS!");
        delay (1000);
        }
    else {
         Serial.println("Esta tudo ok.");
         delay (1000);
         }
}

void SMS_loop () {
    sms.beginSMS("+556191055994");
    Serial.println ("Mensagem enviada ao numero: +556191055994");
    sms.print("Alerta! Propriedade invadida! Disque: 190.");
    sms.endSMS();
    Serial.println("\n COMPLETE! \n");
}

void Receive_loop () {
    if (sms.available()) {
        Serial.println("Mensagem recebida de:");
        sms.remoteNumber(senderNumber, 20);
        Serial.println(senderNumber);
```

```
if (sms.peek()!='#' && sms.peek()!='*') {
   Serial.println("SMS descartado!");
   sms.flush();
}
else {
   if (sms.peek()=='#') {
     val = digitalRead(PinPIR);
     GAS_loop ();
     desliga = false;
     Serial.println("Alarme ativado!");
   }
   else {
      if (sms.peek()=='*') {
        val = false;
        desliga = true;
        Serial.println("Alarme desativado!");
      }
   }
}
Serial.print("Mensagem: ");
while (c=sms.read()) {
     Serial.print(c);
}
Serial.println("\nFinal da Mensagem");
sms.flush();
Serial.println("Mensagem deletada!\n");
```

```
    }
    delay(500);
}

void GAS_loop() {
    valGAS = analogRead(PinGAS);
    Serial.print("Leitura do Sensor = " );
    Serial.println(valGAS);
    if (valGAS > 300) {
      digitalWrite (LRed, HIGH);
      playTone(100,160);
      sms.beginSMS("+556191055994");
      Serial.println ("Mensagem enviada ao numero: +556191055994");
      sms.print("Alerta! Gás detectado! Disque: 193.");
      sms.endSMS();
      Serial.println("\n GAS COMPLETE! \n");
    }
    delay(1000);
}

void playTone(long duration, int freq) {
    duration *=1000;
    int period = (1.0/freq)*1000000;
    long elapsed_time = 0;
    while (elapsed_time < duration) {
        digitalWrite (pinSpeaker,HIGH);
```

```
            delayMicroseconds(period/2);
            digitalWrite(pinSpeaker,LOW);
            delayMicroseconds(period/2);
            elapsed_time += (period);
        }
    }
```