CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB
CURSO DE ENGENHARIA DE COMPUTAÇÃO

BRUNO LOPES RIBEIRO SILVA

SISTEMA DE CONTROLE DO TRIO AUTOMOTIVO POR MEIO DE SMS

Orientadora: MSc. Francisco Javier de Obaldia Díaz

Brasília

Dezembro, 2012

BRUNO LOPES RIBEIRO SILVA

SISTEMA DE CONTROLE VIA SMS DO ALARME AUTOMOTIVO

Trabalho apresentado à Banca examinadora do curso de Engenharia da Computação da FATECS – Faculdade de Tecnologia e Ciências Sociais Aplicadas – Centro Universitário de Brasília como requisito parcial para obtenção do título de Engenheiro da Computação.

Brasília

Dezembro, 2012

BRUNO LOPES RIBEIRO SILVA

SISTEMA DE CONTROLE VIA SMS DO ALARME AUTOMOTIVO

Trabalho apresentado à Banca examinadora do curso de Engenharia da Computação da FATECS – Faculdade de Tecnologia e Ciências Sociais Aplicadas – Centro Universitário de Brasília como requisito parcial para obtenção do título de Engenheiro da Computação.

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. Abiézer Amarília Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Prof. Francisco Javier de Obaldia Díaz, MSc.
Orientador

_____

Prof. Fabiano Mariath D`Oliveira, MSc.
Uniceub

_____

Prof. Antonio Barbosa Junior.
Uniceub

_____

Prof. Luíz Cláudio Lopes de Araujo
Uniceub

AGRADECIMENTOS

Inicialmente gostaria de agradecer a Deus por ter me dado tudo e de maneira especial pela minha família que sempre foi à base e que sempre estiveram ao meu lado, me dando apoio nos momentos difíceis e celebrando a cada sucesso, sem eles nada seria possível.

Agradeço também a minha esposa Nathália Ferreira Borba pela paciência e o apoio dado nesse período da minha vida.

Ao professor Francisco Javier que foi meu Orientador, deixo o meu muito obrigado porque foi imprescindível durante a implementação e escrita do projeto.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

ÍNDICE DE SIGLAS E ABREVIATURAS

ANATEL – Agência Nacional de Telecomunição
ASCII – American Standard Code for Information Interchange
BS – Basic Station
CAN – Controller for Area Network
CPU – Central Processing Unit
D – AMPS – Digital Advanced Mobile Phone System
EEPROM – Electrically-Erasable Programmable Read-Only Memory
ERB – Estação de Rádio Base
ES – Estação Móvel
GPRS – General packet radio service
GPS – Global Position System
GSM – Global System for Mobile Communications
IDE – Integrated Development Environment
IP – Internet Protocol
LCD – Liquid Crystal Display
M2M – Machine to Machine
MHz – Mega-Hertz
MO-SMS – Mobile Originated – Short Message Service
MS – Mobile Station
MSC – Mobile Station Controller
MT-SMS – Mobile Terminated – Short Message Service
PDU – Protocol Data Unit
PIC – Programmable Interface Controller
PIR – Passive Infrared Sensor
PWM – Pulse Width Modulation
RAM – Random Access Memory
RISC – Reduced instruction set computing
ROM – Read Only Memory
SATA – Serial ATA International Organization
SIM – Subscriber identity module
SMS – Short Message Service
SMSC – Short Message Service Center
TCP – Transmission Control Protocol
USB – Universal Serial Bus
USART – Universal Synchronous Asynchronous Receiver Transmitter

RESUMO

A partir de 1990, o uso da tecnologia GSM nas redes de celulares alcançou um grande sucesso, difundindo-se de forma e tornando-se realmente um sistema de comunicação global. Com a utilização dessa tecnologia o presente trabalho busca o controle e monitoramento remoto do alarme automotivo e de todas as funções por ele gerenciadas, como a abertura e fechamento dos vidros, travamento e destravamento de portas e notificação da quebra do bloqueio eletrônico criado pelo alarme automotivo. Assim, o projeto apresenta um sistema capaz de monitorar e controlar as funções da central de alarme e fazer comunicação com o celular por meio de SMS. Para isso, incluiu-se no sistema um Shield GSM com o módulo SIM900. Tudo isso foi feito por meio de um microcontrolador Atmega328 inserido em um Arduino Uno.

Palavras Chave: Arduino, GSM, Monitoramento, Celular, SMS.

ABSTRACT

From the 90s on , the use of GSM technology in mobile networks reached a success, spreading in a manner, really becoming a global communication system. Using this technology, the current work seeks the control and remote monitoring of the automotive alarm and all functions managed by it, such as the and closing of Windows, doors locking and unlocking and notification of a break in the electronic block created by the automotive alarm. Thus, the Project presents a system capable of monitoring and control the alarm central functions and communicate with the cell phone trough SMS. Thereunto, it was included in the system a GSM Shield with the SIM900 module. All that was done with an Atmega328 microcontroller inserted in an Arduino Uno.

Keys Words: Arduino, GSM, Monitoramento, Celular, SMS.

CAPÍTULO 1 – INTRODUÇÃO

Neste capítulo é feita a exposição das questões iniciais deste trabalho de conclusão de curso expondo tópicos como motivação, objetivos e a estrutura da monografia.

1.1. MOTIVAÇÃO

Segundo dados da ANATEL em 2012 o número de linhas ativas ultrapassa o número de habitantes no Brasil, chegando à marca de 250 milhões. Assim, ele deixou de ser aparelho telefônico e passou a assumir diversas outras funções, como o de "central de comando". A rede GSM, rede atual dessa tecnologia, oferece serviço de mensagens curtas (SMS – Short Message Service) que por utilizar um canal de comunicação em um curto período de tempo, sem causar impacto aos recursos da rede, alcançando grande utilização do ponto de vista técnico.

Outro fato motivador desse trabalho é o alarmante crescimento na estatística de roubo de acessórios automotivos sejam por abordagem direta ao condutor ou por arrombamento do veiculo no momento em que seu condutor o estaciona em qualquer local. Muitas vezes o segundo caso pode ser evitado se existirem ações rápidas e fatores que dificultem ou inibam a ação de arrombar o veículo, impedindo que o autor do roubo obtenha o sucesso desejado em sua ação.

Como a automação de processos sempre foi sonho da sociedade e com a evolução no campo eletrônico, os controladores programáveis passaram a ser a fundação para processos automatizados. Vários são os exemplos onde está presente uma unidade de controle (microcontrolador) que torna as atividades do dia-a-dia mais fácil.

E finalmente, com base nesse cenário, buscou-se com a integração de microcontroladores, *modems GSM*, comunicação serial e a rede GSM um mecanismo que ajude o condutor a tomar providências que agilizem sua ação no momento em que seu automotor sofre uma tentativa de arrombamento, assim como se buscou viabilizar a automação de processos por meio da ferramenta disponibilizada pelo celular.

1.2. OBJETIVOS GERAIS

O projeto tem como objetivo especificar e desenvolver um sistema que possibilita a interação entre um Arduino Uno e o alarme automotivo, permitindo o monitoramento e controle de suas funções, assim como a interação entre o modem e o Arduino tornado possível

o controle via SMS, por meio da rede GSM e das funções do trio-elétrico automotivo controladas pela central do alarme automotivo.

1.2.1. OBJETIVOS ESPECÍFICOS

- Integração do Arduino com o Modem;
- Integração do Arduino com os periféricos de entrada e saída;
- Definições das funções que serão controladas pelo Arduino;
- Controle do tráfego de dados através da comunicação serial.
- Configuração e controle do Modem através dos comandados AT integrados com a configuração do Arduino.

1.3. JUSTIFICATIVA

Tendo como gerador do projeto a grande acessibilidade da telefonia celular e com o objetivo de aproveitar essa vantagem, surgiu a ideia de se desenvolver um sistema capaz de integrar o alarme automotivo a um modem GSM.

1.4. APRESENTAÇÃO DO PROBLEMA

O desenvolvimento de um projeto como o aqui apresentado impõe os seguintes desafios: integrar e interagir o Arduino Uno com o shield SIM900, estabelecer a comunicação entre os dois dispositivos e controlar e monitorar todas essas fases, de forma que seja criado um sistema que seja capaz de monitorar as funções do alarme automotivo de modo que caso exista alguma intrusão o Arduino seja capaz de identificar o motivo. Além disso, o sistema deve ser capaz de ao identificar alguma intrusão e trata – lá de forma correta executando a ação para o qual foi previamente configurado.

1.5. ESTRUTURA DA MONOGRAFIA

Capitulo 1 – Trata da introdução do trabalho, versa sobre a motivação para o projeto, seus objetivos e a forma como a monografia será organizada.

Capítulo 2 – Apresenta a parte destinada ao referencial tecnológico do trabalho, descriminando as características relativas ao Arduino, rede GSM, descrição técnica sobre trio-automotivo e sobre o Processing e códigos AT.

Capítulo 3 – Destinado ao desenvolvimento, do qual constará a descrição detalhada sobre a execução do projeto, citando as ferramentas utilizadas, descrição detalhada das fases do projeto, explicação dos códigos, detalhes de implementação, dos testes e resultados obtidos.

Capitulo 4 – Parte destina a Montagem do protótipo onde foram feitos os testes e experimentos no protótipo do projeto.

Capítulo 5 – Parte destinada as conclusões obtidas nos projetos, suas projeções e possibilidade para continuidade do projeto.

CAPÍTULO 2 – REFERENCIAL TEÓRICO

Nesse capítulo, são apresentadas tecnologias que servem como base ao desenvolvimento do projeto, quer seja na utilização direta ou referenciando aspectos afins com o projeto, fornecendo escopo suficiente para que o protótipo possa ser construído**.**

2.1. MICROCONTROLADORES

Pode-se citar a versatilidade do microcontrolador como sua principal característica, pois em sua placa, além da CPU, possui elementos necessários para a execução de qualquer projeto: Memórias RAM e ROM, Timers (contadores de tempo), porta serial(responsável pela interface com o barramento gerador de um linha de comunicação serial), Contadores, PWM, canais de comunicação e conversores de sinais analógico-digitais A/D e digitais-analógicos D/A. Cita-se, também, como vantagem do microcontrolador a dificuldade para efetuar-se a cópia do código do software interno, dificultando assim, a ação de agentes não autorizados.

Por outro lado a CPU de microcontroladores, quando comparada á dos microprocessadores, não é tão robusta e possui um conjunto de instruções limitadas, baixa 15resence15e de clock e reduzido espaço de memória endereçável. Isto faz com que os microcontroladores possuam campo definido de aplicação.

Pela sua fácil manipulação, encontram-se microcontroladores, em várias situações: em semáforos, eletrodomésticos, balanças eletrônicas, calculadoras, telefones públicos, microterminais, controle para carregamento de baterias, controles de acesso. Devido a sua forma de acesso a periféricos, de forma padronizada e integrada, a sua própria programação, que é bem simples, dependendo da forma, o microprocessador pode ser ideado em assembler, C ou Processing (ambas, formas de programação, serão discutidas abaixo)

2.1.1. MICROCONTROLADOR ATMEGA328P

O microcontrolador Atmega328P é utilizado pelos dispositivos Arduino Uno, Arduino 2009 possui um microcontrolador de 8 bits. O chip é fabricado pela ATMEL. A arquitetura RISC (Reduced Instruction Set Computer ou Computador com um Conjunto Reduzido de Instruções, e representa uma linha de processadores que apresentam um conjunto simples e pequeno de instruções e que utilizam a mesma quantidade de tempo para executar as mesmas) com frequência de operação em 16 MHz. O referido dispositivo ainda possui memórias do tipo Flash, EEPROM e RAM, sendo que cada uma delas possui 32KB, 1KB e 2KB,

respectivamente. Na sua composição ainda estão vinte e três entradas e saídas digitais, com tensão de operação de varia de 1.8 a 5.5V e baixo consumo de energia [ATMEL].

2.1.2. ARDUINO

Plataforma de computação embarcada de fonte aberta, baseada em uma placa formada por entrada e saídas, que se desenvolve através de uma linguagem conhecida como Processing, utilizado principalmente para o desenvolvimento de objetos interativos independentes, ou conectados a softwares de um computador(Banzi,2011).

Características do Arduino:

- 6 Pinos de entrada Analógica(pinos 0-5);
- 14 Pinos Digitais de Entrada/Saída(pinos 0-13);
- 6 Pinos de saída Analógica(pinos 3,5,6,9,10 e 11);
- MicrocontroladorATmega328;
- Voltagem de Operação 5 volts;
- Voltagem de Entrada (recomendada) entre 7 a 12 volts;
- Voltagem de Entrada (limites) entre 6 a 20 volts;
- Corrente D/C por pino E/S: 40 mA;
- Corrente D/C por pino 3.3V: 50 mA;
- Memória Flash: 32 KB (Atmega328);
- SRAM: 2 KB(Atmega328);
- EEPROM: 1 KB(Atmega328);
- Clock Speed: 16 MHz.

A figura 2.1 mostra uma placa do Arduino Uno que será utilizada no projeto o Quadro 2.1 mostra quais pinos do Arduino que serão utilizados no projeto e a figura 2.2 mostra o esquemático de um Arduino Uno.

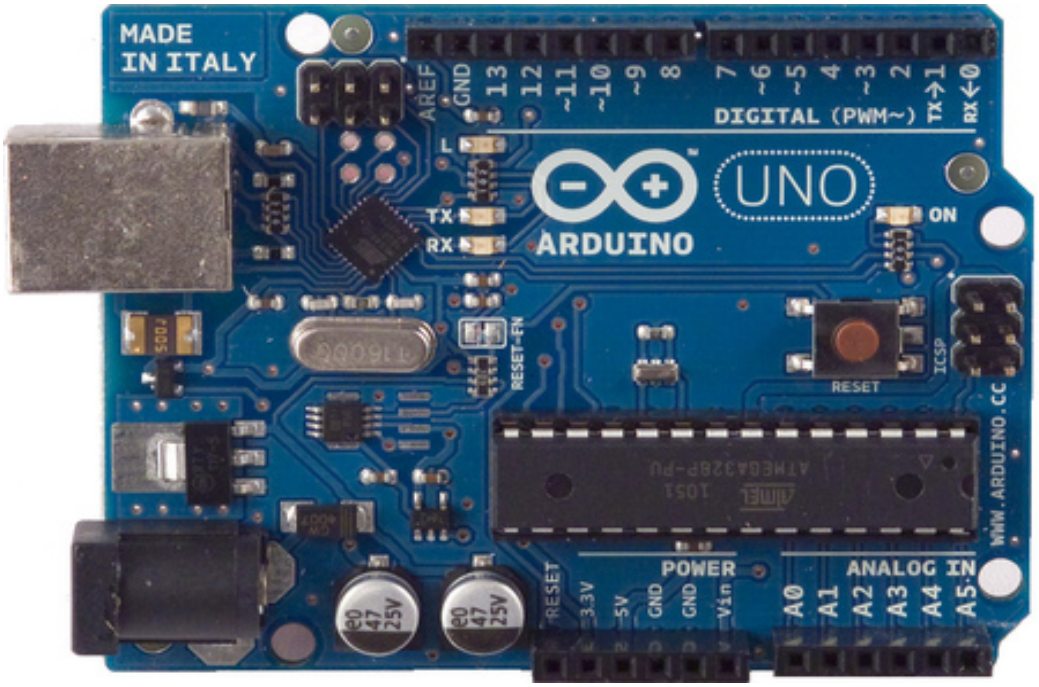

Figura 2.1 – ArduinoUNO
(Fonte: www.arduino.cc**)**

Quadro 2.1 – Pinos Utilizados no Projeto

Pino	Função
0	RX – Hardware Serial
1	TX – Hardware Serial
2	Não usado
3	Não usado
4	RX – Software Serial e pino de Status
5	TX – Software Serial
6	Não usado
7	Pino de Entrada do Push_Button
8	Pino de Entrada do Sensor de Presença
9	Pino usado para ligar o módulo
10	Pino do Motor de Passo
11	Pino do Motor de Passo
12	Pino do Motor de Passo
13	Pino do Motor de Passo
A0	Led que indicara que o alarme esta ligado
A1	Led que indicara intrusão pelas portas
A2	Led que indicara a quebra de vidros
A3	Led que indicara o estado das portas
A4	Led que indicara que o estado dos vidros
A5	Não Usado

Fonte(Autor, 2012)

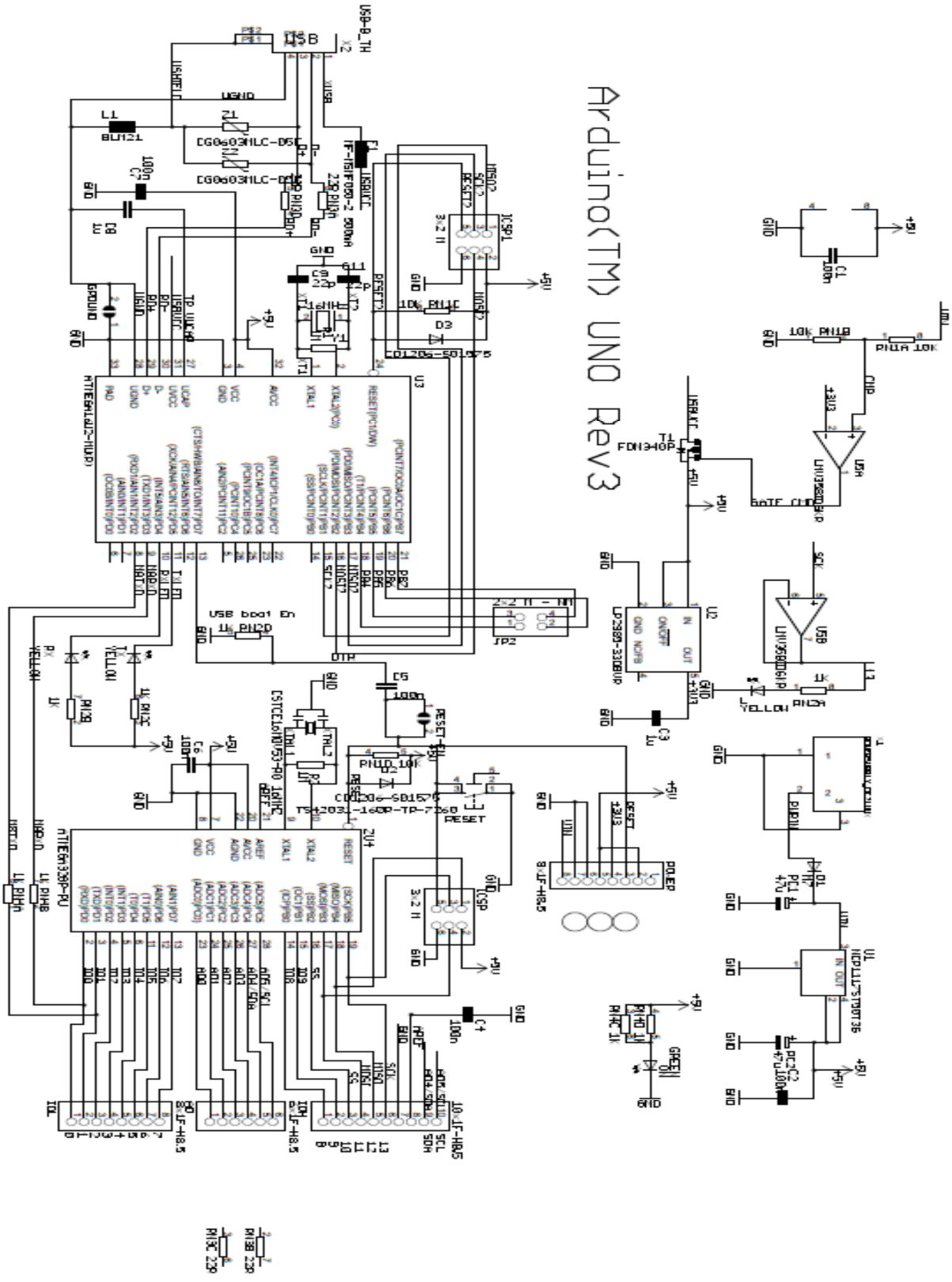

Figura 2.2 – Diagrama Arduino UNO
(Fonte: www.arduino.cc**)**

2.1.3. SHIELD

Shields são placas de circuitos contendo outros dispositivos (Por exemplos receptores GPS, displays de LCD, módulos de Ethernet etc), que ao se conectar no Arduino esse obtém funcionalidades adicionais. Shields também estendem pinos até o topo de suas próprias placas de circuito, para que você continue a ter acesso a todos eles. A utilização do Shield é opcional (Michael McRoberts, 2011).

2.1.3.1. SIM 900

Módulo GSM/GPRS quadband com pilha TCP/IP embutida de fácil soldagem e integração, até mesmo para prototipagem (board-to-board) e pode ser usado em aplicações onde a transmissão via tecnologia GSM/GPRS é necessária, seja por transmissão de voz, SMS, possui consumo reduzido de energia. Ideal para aplicações industriais, como M2M, telemetria, etc.

O SIM900 possui como principais características, a figura 2.3 mostra o SIM900.

- Quad-Band 850/ 900/ 1800/ 1900 MHz
- GPRS multi-slot class 10/8
- GPRS mobile station class B
- Compliant to GSM phase 2/2+Class 4 (2 W @850/ 900 MHz)
- Class1 (1 W @ 1800/1900MHz)
- Controle via AT 19resence19 (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)
- Baixo Consumo de Energia: 1.5mA(sleep mode)
- Temperatura de Operação: -40°C to +85 °C
- Suporta para se ativado via Software

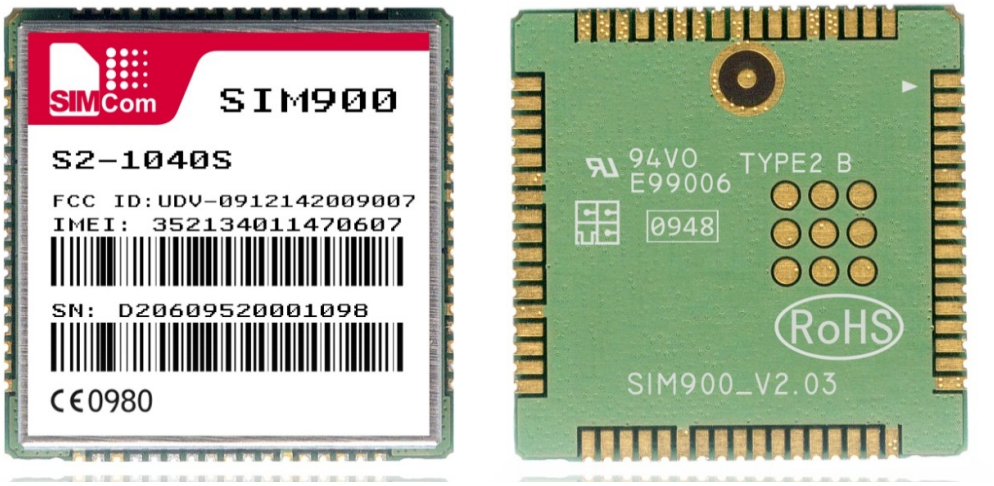

Figura 2.3 – SIM 900
(Fonte: www.iteadstudio.com**)**

2.1.3.2. COMANDOS AT

Antigamente chamados de Hayes Command Set, os comandos AT implementam uma linguagem de programação que também é utilizada para operar alguns modelos de modens remotamente. (SANTANA, 2007).

Uma linha de comando AT pode conter um ou mais comandos, utilizando delimitadores para separar cada comando. A linha possuirá o prefixo “AT” e o sufixo, ASCII de Carriage Return, o delimitador poderá ser um ponto e virgula ou um espaço. O modem emitirá uma mensagem, Result Code, no momento em que o comando é emitido, avisando assim ao terminal o resultado do comando requisitado.

Os serviços executados pelo módulo SIM900 são, chamada de voz, envio e recebimento de SMS, entre outros, e serão controlados por meio de instruções formadas por comandos AT.

2.2. COMUNICAÇÃO SERIAL

O aperfeiçoamento de técnicas de comunicação serial com a do microcontrolador o capacitam ainda mais para alcançar velocidades elevadas. Pode-se citar como exemplo as técnicas mais utilizadas: USB 2.0, Firewire, SCSI serial, SATAI e SATAII (PEREIRA, 2003).

O módulo interno de comunicação serial – USUART (Universal Synchoronous Asynchronous Receiver Transmitter), do PIC, é uma grande vantagem, tal protocolo possui dois modos de operação, síncrono e assíncrono. (LAVINHA, 2003)

Torna-se necessário que o tamanho de dados(intervalo de bits) tenha um padrão, possuindo, os dois lados, o mesmo valor. Tal velocidade, Baud Rate, é indicada em bits por segundo. O quadro 2.1 mostra parâmetros de configurações seriais

Quadro 2.2 – Parâmetros para configuração em C de comandos serial

OPÇÃO	DESCRIÇÃO
BAUD = valor	Especifica a velocidade da comunicação serial.
XMIT = pino	Especifica o pino de transmissão de dados.
RCV = pino	Especifica o pino de recepção de dados.
RESTART_WDT	Determina a função GET() ressete o watchdog enquanto aguarda a chegada de um caractere.
INVERT	Inverte a polaridade dos pinos de TX/RX, não pode ser utilizada com o hardware interno.
PARITY = x	Seleciona a paridade (x pode ser: N (sem paridade), E (paridade par) ou 0 (paridade impar).
FLOAT_HIGHT	A saída não vai em nível lógico 1. Utilizado com saídas coletor aberto.
ERRORS	Solicita o armazenamentodos erros de recepção na variável RS232_ERRORS, ressetando os flags de erro quando eles ocorrem.
BRGH10K	Permite a utilização das velocidades disponíveis com o bit BRGH em 1 em chips que tenham bugs nesta configuração do hardware.
ENABLE = pino	Especifica um pino para atuar como saída de habilitação durante uma transmissão. Utilizado no protocolo RS485.
STREAM =identificador	Associa a interface RS232 a um identificador de stream de dados.
BITS = x	Seleciona o número de bits de dados

(**Fonte: Pereira, 2003**).

2.3. TELEFONIA MÓVEL

O avanço tecnológico e as interações 22resence22e entre as partes do globo terrestre demandam uma comunicação rápida e com mobilidade, assim como a necessidade da transmissão de dados por meio de um sistema eficaz e confiável. Nesse cenário, foi criada a telefonia celular, com a finalidade de promover a comunicação entre dois pontos, uma estação móvel (ES) e outra estação móvel ou uma unidade fixa (Estação Rádio-Base, ERB ou BS).

A transmissão em uma rede de telefonia móvel ocorre no momento em que um usuário digita um numero de 8 dígitos (ligação local), 11 dígitos (ligação interurbana) ou 13 dígitos (ligação internacional). A MS – Mobile Station – buscará na banda um canal com o sinal forte e enviará o número para ERB mais próxima ao local que estiver utilizando aquele canal e retransmitirá-lo para a MSC (mobile station controller), que por sua vez, remetera os dados para central telefônica. Caso a parte de chamada esteja disponível, a conexão será feita e o resultado transmitido para a MSC, ponto em que a MSC atribuirá um canal de voz para ligação e conexão estabelecida. A ERB se ajustará de forma automática para dar continuidade à ligação pela busca de novas células.

A recepção ocorre no momento em que uma chamada é direcionada a uma estação móvel, o número enviado pela central telefônica para a MSC. A MSC, por meio de um sinal de consulta, localiza a Estação Móvel dentro de uma célula (rotina chamada de paging). Sendo a ES localizada, a MSC transmite um sinal de chamada e, quando a ES responde um canal de voz é atribuído dando inicio a comunicação de voz.

Podemos dividir em três grandes gerações a evolução da tecnologia de comunicação móvel por meio do celular. A primeira geração possibilita a comunicação de voz por meio de canais analógicos. A segunda geração, desenvolvida para proporcionar uma comunicação de voz de alta qualidade, possuía suporte a tecnologia digital. Nessa geração os principais sistemas desenvolvidos foram o D-AMPS –Digital Advanced Mobile Phone System-, e o GSM – Global System for Mobile Communication. A convenção de várias tecnologias proporcionou uma gama variada de serviços, onde é possível a comunicação tanto de dados como de voz digitalizada. (FOROUZAN, 2009)

2.3.1. GSM

Criado com o intuito de oferecer uma padronização na telefonia digital Européia, tornou-se rapidamente o padrão mais utilizado no mundo por sua capacidade.

O GSM – Global System for Mobile Comunication – utiliza duas bandas para a comunicação duplex. Cada uma possui largura de faixa de 25 MHz a menor centrada aproximadamente entre 902 MHz e a maior centrada em 947 MHz, conforme figura 2.4 que mostra que cada banda será subdividida em 124 canais que serão separados por faixas de segurança. A figura 2.4 mostra a faixa de funcionamento da rede GSM.

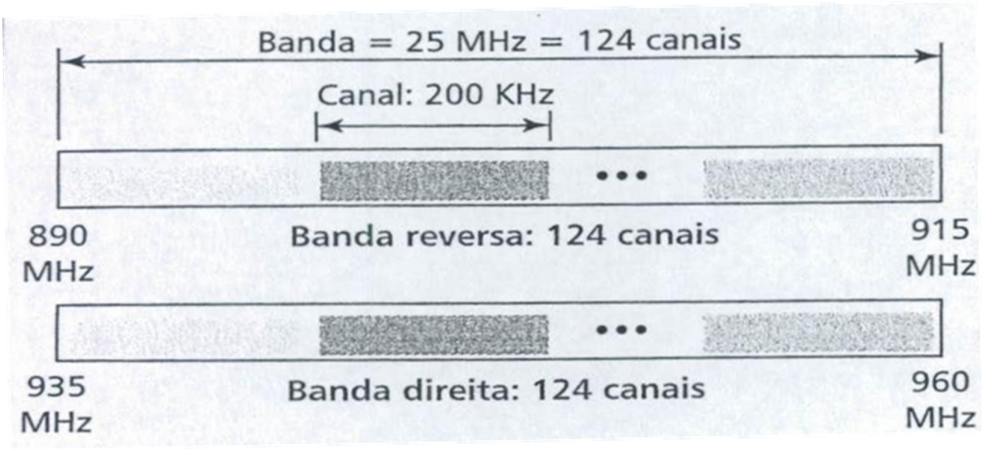

Figura 2.4–Banda de funcionamento da rede GSM

(Fonte: FOROUZAN**,** 2006**)**

A figura 2.5 mostra o sistema GSM e os canais digitalizados e comprimidos de voz formando um sinal digital de 13 Kbps, sendo transmitidos 156,25 bits por shot-time.

Para formar um frame TDM serão necessários oito slotsmultiplexados, sendo que 26 frames formarão um multiframe. Com essas informações pode-se calcular a taxa de transmissão da rede GSM da seguinte forma:

Taxa de Transmissão do Canal = (1/20 ms) x 26 x 8 x 156,25 = 270,8 Kbps. Assim, cada canal digital terá uma taxa de transmissão de 270,8 Kilobytes por segundo que serão modulados numa portadora por meio da GMSK(variante da modulação FSK).

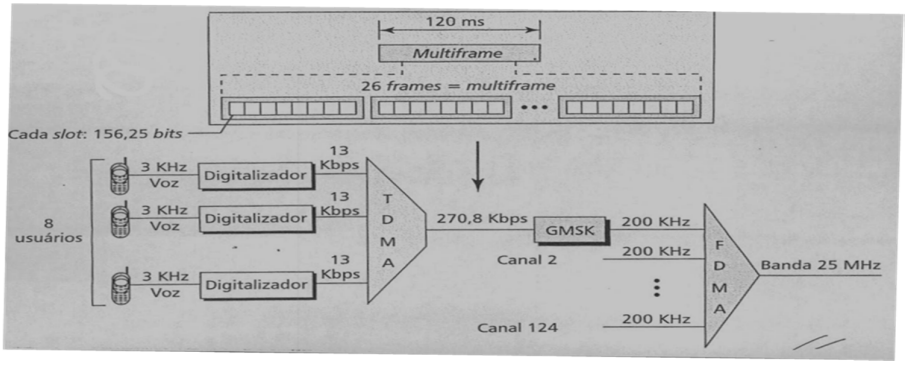

Figura 2.5 – Sistema GSM Típico

(Fonte: FOROUZAN**,** 2006**)**

Tendo como resultado uma onda analógica de 200 kHz. Finalmente, 124 canais analógicos de 200 kHz cada, multiplexados pela técnica FMDA, tendo como resultado uma banda de 25MHz.

A figura 2.6 ilustra os dados do usuário e a sinalização (overhead) dentro de um multiframe.(FOROUZAN,2006).

Os padrões mais utilizados de GSM são o GSM 900 Primário que utiliza frequência entre 890 Mhz e 960 MHz, GSM 900 Estendido que utiliza frequências entre 880 e 960 MHz diferenciando do anterior por possuir enlaces diretos e reversos de apenas 10 MHz, GSM 900 Ampliado com um espectro de frequências entre 876 e 960 Mhz com distância reduzida entre os enlaces (6 MHz), GSM 1800 com banda de 75 MHz de largura e faixa de 1,8 Ghz – implantado no Brasil para implementar a PCN – Personal Communication Networks, PCS 1900 com faixa de operação de 1,9 Ghz e espectro localizado entre 1850 e 1990 MHz sendo capaz de gerar 300 canais de RF(FORUZAN – 2006) .

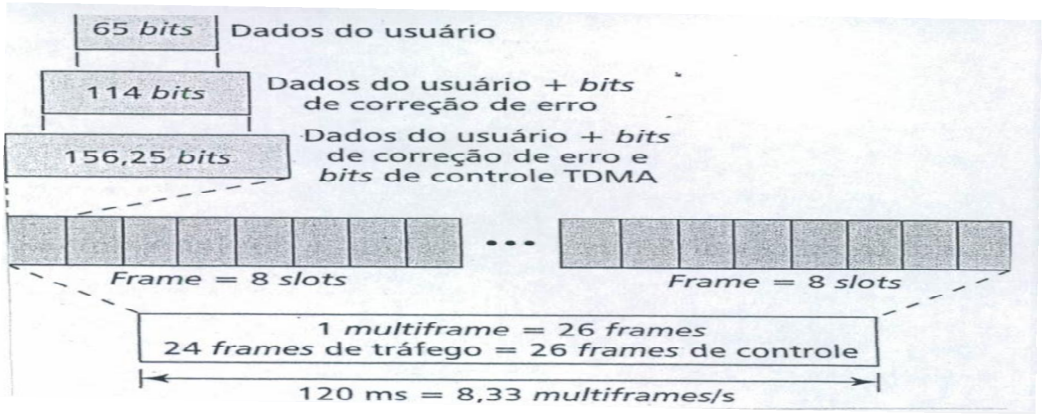

Figura 2.6 – Componentes do Multiframe

(Fonte: FOROUZAN, 2006)

2.3.2. SMS – SHORT MESSAGE SERVICE

Serviço oferecido a toda rede GSM permitindo a todo usuário enviar e receber mensagens alfanuméricas de até 160 caracteres. Aplicação do tipo armazena e encaminha (store-and-forward), sendo assim, não se envia diretamente SMS entre usuários, mas primeiramente ao SMSC(Short Message Service Store – Centro de Mensage), local de direcionamento aos destinatários, ocorrendo falha na entrega, o SMSC armazena a mesma e num período determinado a mesma e reenviada ao destinatário. Divide-se as mensagens em dois tipos:

MT-SMS (Mobile Terminated – Short Message Service): Mensagens enviadas do SMSC ao terminal móvel. Possui origem no SMSC e final na estação móvel.

MO-SMS (Mobile Originated – Short Message Service): Não são obrigatoriamente de uma estação móvel, mas são enviadas a uma SMSC.

Como SMS podem ser enviadas em modo texto ou PDU (Protocol Data Unit – Protocolos de Unidade de Dados), elas também serão classificadas de acordo com o formato de envio.

O modo texto estruturado de forma mais simples, necessita somente escrever diretamente os caracteres que se deseja enviar, porém não disponível em todos os dispositivos.

O formato PDU de estrutura complexa, consiste em octetos hexadirecionais que informam a SMSC, o tipo de endereço, nacional ou internacional, número do destinatário, comprimento do dados, entre outras. O conteúdo da SMS deve ser escrito em Hexadecimal seguindo a codificação ASCII.

Uma das funções do SMS, que será utilizada nesse trabalho, é a que utiliza o protocolo de aplicativo(WAP) sem fio, podendo configurar dispositivos móveis pelo ar, enviando itens e iniciar aplicativos com pacotes WAP, como por exemplo, configurar e controlar modems..(FOROUZAN,2006).

2.4. LINGUAGEM PROCESSING

Processing é uma linguagem de programação de código aberto e ambiente de desenvolvimento integrado (IDE), construído para as artes eletrônicas e comunidades de design visual com o objetivo de ensinar noções básicas de programação de computador em um contexto visual e para servir como base para cadernos eletrônicos. O projeto foi iniciado em 2001 por CaseyReas e Ben Fry, ambos ex-membros do Grupo de Computação do MIT Media Lab. Um dos objetivos declarados de processamento é atuar como uma ferramenta para não-programadores iniciados com a programação, através da satisfação imediata de um feedback visual

2.5. TRIO ELÉTRICO AUTOMOTIVO

O Trio-elétrico automotivo, conhecido basicamente como conjunto formado de travas elétricas, alarme, módulo de subida do vidro, e em alguns veículos possui um sistema para imobilização do motor.

2.5.1. ALARME

Tem como principal função proteger o interior do veículo contra furto, assim como, o próprio veículo. Para isso utiliza de vários sensores espalhados no interior do automotor:

- **Porta do motorista**: Interruptor (comutador) localizado na Coluna a esquerda (dobradiça do carro), tem o funcionamento baseado na abertura da porta que empurrado pela mola que fecha o circuito e envia um sinal elétrico para o módulo de controle(item 1 da figura 2.7);
- **Porta do passageiro**: Interruptor localizado na Coluna adireita (item 2 da figura 2.7);
- **Capô**: Interruptor localizado, normalmente, em local próximo a torre de amortecimento ou próximo ao painel que separa a cabine dos passageiros do compartimento do motor (item 5 da figura 2.7);
- **Portas traseiras**: Interruptores localizados na Coluna B direita e esquerda (itens 3 e 4 da figura 2.7)
- **Porta-Malas**: Interruptor instalado na tampa do Porta-Malas (item 2 da figura 2.7);
- **Vidros Traseiros**: Caso o carro possua desembaçador, utiliza-se tal como sensor no caso da quebra do vidro, caso contrário regula-se o sensor ultra-sônico para verificação da quebra do vidro traseiro(item 8 da figura 2.7).
- **Vidros (Portas Dianteiras e Portas Traseiras)**: Sensor ultra-sônico localizado geralmente atrás do espelho retrovisor, faz a vezes de verificação de quebra de vidro, assim como o monitoramento do interior do veículo, após a quebra de algum vidro(item 9 da figura 2.7).

O sistema funciona baseado em uma central localizada próxima a caixa de fusível do veículo, um módulo eletrônico responsável pelo recebimento, interpretação e definição se o buzzer (aviso sonoro) deve ou não ser acionado, após o recebimento de um sinal enviado por

um ou mais mecanismo acima mencionado. A figura 2.7 mostra a localização de cada um dos componentes descritos acima.

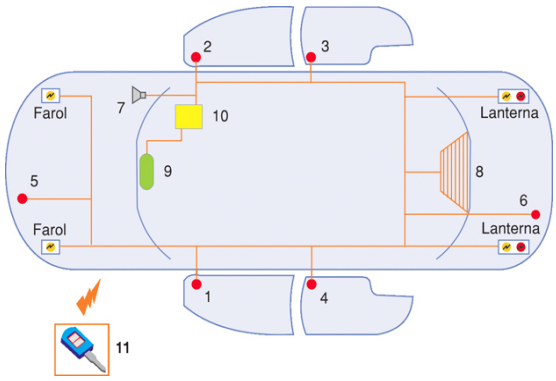

Figura 2.7 – Componentes do alarme e imobilizador do motor.
(Fonte: Saber Eletrônica, 2008**)**

2.5.2. IMOBILIZADOR DO MOTOR

Uma antena instalada próxima ao tambor de ignição do veículo conectada a um módulo eletrônico faz a leitura de um chip (transponder) que está instalado na chave do veiculo. A leitura é feita toda vez que a solicitação de partida é dada. Caso exista a tentativa de ligação com uma cópia mecânica da chave irá negar o pedido, pois não houve confirmação do transponder do sistema.

2.5.3. CONTROLE

Tem o funcionamento baseado em ondas de rádio, tem como objetivo primário ligar ou desligar o sistema de alarme.

2.6. TRAVAS ELÉTRICAS

Localizado nas portas do veiculo são módulos que contem pequenos motores e algumas engrenagens eletromecânicas (conforme itens 5,6,7,8 da figura 2.8), ficam responsáveis pelo travamento e destravamento de portas, assim como seu possível travamento mecânico.

O sistema torna possível, em alguns modelos o acesso a partes especificas do veiculo, como as portas, tampa do bagageiro, tampa do tangue de combustível. A figura 2.8 mostra os componentes e ligações do sistema de travas elétricas.

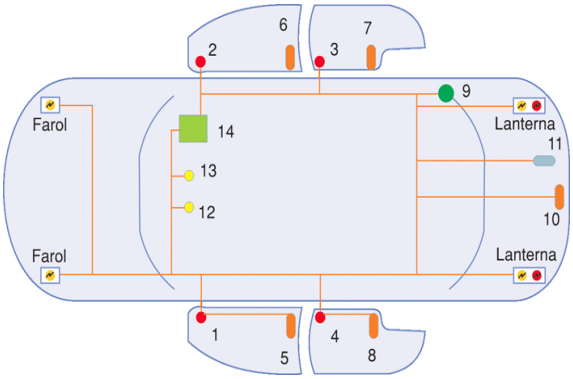

Figura 2.8 – Componentes das Travas Elétricas do Veículo

(Fonte: Saber Eletrônica, 2008**)**

2.7. ACIONAMENTO ELÉTRICO DOS VIDROS

Formado por motores elétricos o sistema de acionamento elétrico dos vidros é responsável pela abertura ou fechamento dos vidros, possuindo algumas características especificas, como citaremos a seguir:

Comfort Closing– fechamento dos vidros no momento em que se aciona o alarme e travas-elétricas.

Express Upand Express Down – subida ou descida do vidro com apenas um toque no interruptor de comando.

Pich Protection – Proteção contra esmagamento.

Internal Pressure Relief – Quando em subida algo impede o fechamento, o sentindo será alterado, evitando assim acidentes.

O sistema funcionará baseado em um conjunto formado pelo Módulo eletrônico, localizado próximo à caixa de fusíveis, máquinas de vidros, localizados nas portas e interruptores de acionamento, localizados de acordo com o modelo do veiculo. A figura 2.9 mostra a localização do módulo eletrônico(item 13), a máquina de vidros(itens 5,6,7 e 8), interruptores de acionamento(itens 1,2,3 e 4).

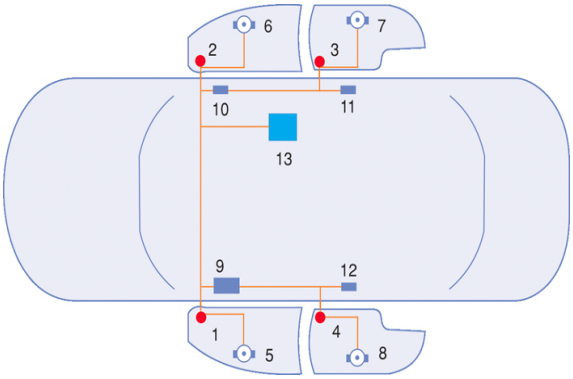

Figura 2.9 – Componentes do Acionamento elétricos do vidros

(**Fonte**: Saber Eletronica, 2008)

Para o projeto as funções do trio elétrico serão simuladas através dos seguintes dispositivos:

- Motor dos Vidros – Motor de Passo e Led Verde
- Travas elétricas – Led Verde
- Acionamento do Alarme – Led Vermelho
- Acionamento do Alarme pela abertura da porta – Push Button e Led Amarelo
- Acionamento do Alarme pela quebra dos Vidros – Sensor PIR e Led Amarelo

Tendo como base os conceitos aqui apresentados no próximo capítulo serão mostrados a integração de cada componente e como com a utilização do arduino em conjunto com o shield GSM é possível o controle da funções do trio elétrico automotivo através de um simples SMS.

CAPÍTULO 3 – DESENVOLVIMENTO DO PROTÓTIPO

Esse capítulo será destinado ao desenvolvimento, do qual constará a descrição detalhada sobre a execução do projeto, citando as ferramentas utilizadas, descrição detalhada das fases do projeto e explicação dos códigos.

3.1. DESCRIÇÃO DO SISTEMA

Busca-se o desenvolvimento de um sistema que tenha como pilares a automação automotiva com a mobilidade do celular. O projeto utiliza um microcontrolador Atmega328P programado em uma linguagem de alto nível (Processing), que conectado ao Shield GSM pode enviar os comandos ao microcontrolador por meio de SMS.

Para programar o sistema é necessária a ligação de um microcontrolador diretamente ao kit do alarme automotivo com 30resence30e, tendo a finalidade de controlar as funções que serão solicitadas pelo usuário por meio da rede GSM.

Para que o usuário possa fazer solicitações por meio de SMS e necessário que o microcontrolador se comunique com um dispositivo que permita a interface com o celular. Para que tal funcionalidade possa ser alcançada é necessário um modem que possua funções de conectividade com a rede GSM. Para interface entre o modem e o microcontrolador são utilizados comandos AT (comando de configuração padrão do modem). A figura 3.1 mostra a topologia do sistema.

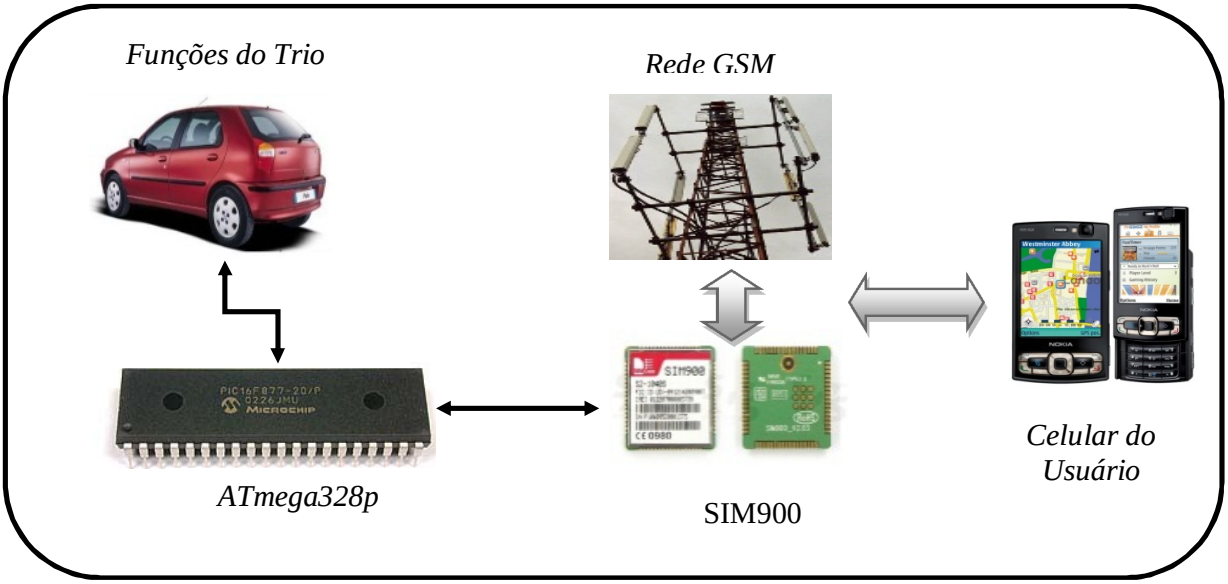

Figura 3.1 – Topologia do Sistema
(FONTE: Autor, 2012**)**

3.1.1. DEFINIÇÃO DAS FUNÇÕES DO SISTEMA

Para o desenvolvimento do sistema, primeiramente foi feita uma busca em projetos semelhantes para ver até que ponto seria possível à atividade do projeto com recursos disponíveis. Por exemplo, citaremos o projeto de Ivan Sampaio Nascimento que fala sobre "**Sistema de Alarme Automotivo que integra transdutor acústico/elétrico e celular**" (Nascimento, 2008), que tem como finalidade avisar o proprietário do veiculo, por meio de uma chamada telefônica, caso haja um evento sonoro no interior do véiculo.

As ações de controle aqui apresentadas serão controladas, monitoradas e ordenadas por meio de mensagens de texto.

Todo comando enviado pelo usuário retorna uma mensagem de confirmação de êxito da solicitação. O quadro 3.1 traz uma lista de todas as ações que serão executadas pelo sistema.

Quadro 3.1 – Funções do Sistema

Comandos	Ações
0001	Ativa o sistema
0002	Desativa o Sistema
0003	Fecha as Portas
0004	Abre as portas
0005	Fecha os vidros
0006	Abre os vidros
Alarme	Avisa ao usuário que o perímetro do alarme foi violado

(FONTE: Autor, 2012)

A validação do sistema foi feita por meio da utilização de um chip SIM900 presente em um Shield Icomsatv1.1 acoplado a um chip Atmega presente em um Arduino Uno Rev3.

A utilização de periféricos irão simular os dispositivos presentes em um veículo, pois devido a fatores orçamentários o projeto não pode ser implementado em um carro. Outro motivo que levou a não implantação do sistema em um carro foi à dificuldade de adequar o protocolo CANBUS, pois cada carro possui um protocolo diferente e isso também ocorre em carros do mesmo modelo e ano.

3.1.2. 3. CONFIGURAÇÃO DO SHIELD SIM900

Além da aquisição do Shield, foi necessário a aquisição de um SIM Card e a inserção do mesmo no Shield. Para o projeto foi adquirido um chip da operadora VIVO, pré pago, no valor de R$ 10,00(dez reais). Para SIM Card, não é necessário a inserção de PIN, será inserido em um soquete de acordo com a figura 3.2 .

Figura 3.2 Inserção do SIM no Shield
(**FONTE**: Autor, 2012)

Após essa inserção é necessária a colocação correta dos Jumpers TX e RX do shield de acordo com o arquivo GSM.cpp presente na biblioteca GSMShield elaborada pela própria fabricante do modem, pois como no programa, serão utilizadas duas entradas seriais; a primeira será a própria entrada serial da placa arduino, que é mais conhecida como Hardware Serial. Esta será utilizada para a comunicação do modem com o computador e no projeto foi utilizada para o acompanhamento do programa pela Serial Monitor. A outra entrada serial será através do Software Serial que e é configurada através de software, utilizando a biblioteca SoftwareSerial.h. A figura 3.3 mostra a correta alocação dos Jumpers.

Como a configuração da SoftwareSerial acontece por meio de software abaixo segue o trecho onde é chamada a biblioteca SoftwareSerial.h e logo em seguida escolhem-se os pinos do Arduino que serão usados como TX e RX, respectivamente.

```
#include <SoftwareSerial.h>
#include <String.h>
SoftwareSerial gsm(4, 5);     //TX e RX
```

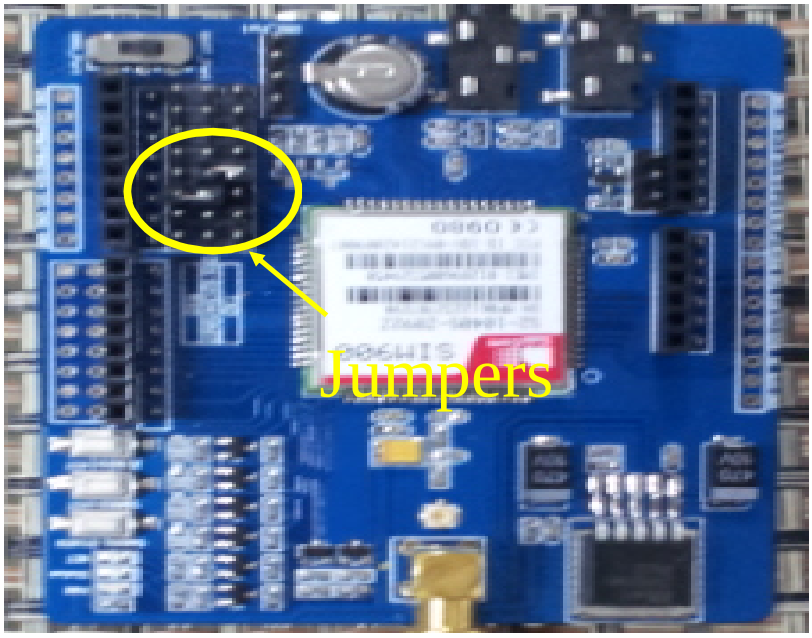

Figura 3.3 – Configuração dos Jumpers
(Fonte: Autor, 2012)

A software serial usada no projeto será conhecida com “gsm”.Todos os comandos utilizados na Hardware Serial são válidos para a SoftwareSerial.

Para o correto funcionamento do sistema é indispensável que a placa e o módulo funcionem com a mesma taxa de bits. A baud rate do modem vem configurada de fábrica em 9600.

A comunicação serial do arduino-tanto a Software Serial (gsm) e a Hardware Serial serão configuradas conforme o trecho de código abaixo:

```
 void setup(){
...
gsm.begin(9600);              //Serial baud rate
Serial.begin(9600);           //Serial baud rate
…
}
```

Outro ponto a ser observado é o tempo de inicialização do modem e que ele necessita que seu botão power seja pressionado ou que o pino 9 do Arduino esteja em alto logo seu funcionamento é vinculado ao pino 9 do Arduino e ao pino RX da Software Serial do Arduino que o usado é o pino 4. Então, criou-se a função poweron() que tem por objetivo verificar o estado do Pino 4. A função é mostrada a seguir:

```
…
void loop(){
.
.
.
poweron();
}
void poweron() {
pinState = digitalRead(statuspin);        //faz a leitura do valor do pino 4
if(pinState==LOW)
{
Serial.println("estou ligando...");
digitalWrite(9,HIGH);
delay(2000);
digitalWrite(9,LOW);
delay(3000);
}
};
```

3.1.3. COMANDOS AT

O módulo SIM900 possui dois modos de trabalho para tratamento de SMS o modo texto e o modo PDU. Optou-se pelo modo texto devido ao reduzido número de caracteres necessários para a sua utilização.

Inicialmente configura-se o modo de trabalho das mensagens, o comando AT+CMGF é responsável pela escolha:

- **AT+CMGF**=0- Modo PDU(padrão)
- **AT+CMGF**=1 –Modo Texto

A função CMMI, possível somente no modo texto, é a forma que a módulo ira tratar o recebimento da mensagem, que pode ser inteira, direto no terminal ou direciona – lá assim que a mesma é recebida pelo modem, sem a necessidade de instruir o modem para a leitura da mensagem. No projeto optar-se-á pelo recebimento direcionado através dos parâmetros: **AT+CNMI=3,3,0,0,0**. O comando enviará ao microntrolador a mensagem recebida no formato: +CMGR: "RECREAD", xxxx","","12/10/21,12:21:43-12" e enviará diretamente a Serial Software.

Para a criação da mensagem é necessária a utilização do comando **AT+CMGS** precedido do número destinatário. Fez-se a opção pela maneira que demande um número reduzido de caracteres do microcontrolador. O trecho do código a seguir mostra o procedimento para envio das SMS pelo Arduino.

```
{
…

gsm.println("AT+CMGS=\"+xxxxxxxx\"\r");   // Número de Envio da SMS
delay(1000);
gsm.println("Alarme Ativado");  // Começo da escrita da SMS
delay(1000);
gsm.println((char)26); //Carrie Return "Enter"
...
}
```

Com o objetivo de economia de espaço em memória optou-se pela utilização da mensagem e posterior exclusão do SMS do buffer. Para a tarefa utiliza-se o comando AT+CMGDA que possui funcionamento conforme quadro abaixo:

Quadro 3.3 – Comando AT+CMGDA

Opção	**Função**
"DEL READ"	Exclui as SMS lidas
"DEL UNREAD"	Exclui as SMS não lidas
"DEL SEND"	Exclui as SMS enviadas
"DEL UNSENT"	Exclui as SMS não enviadas
"DEL ALL"	Exclui todas as SMS
Estrutura:	AT+CMGDA="Opção"

(**FONTE**: Autor, 2012)

A estrutura de código a seguir é utilizada para a exclusão das mensagens, pois em diversas partes do Software é imprescindível sua utilização.

```
Void apaga_sms(){
Serial.println("|");
Serial.println("|");
```

```
Serial.println("------------------------- Excluindo SMS -------------------------");
Serial.println("|");
Serial.println("|");
gsm.println("AT+CMGDA=\"DEL ALL\"\r");//executa comando
delay(500);
 //espera resultado na serial
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);  //mostra retorno real
limpa_buffer();          //limpa buffer
Serial.println("|");
Serial.println("|");
Serial.println("------------------------- Excluindo SMS fim -------------------------");
};
```

Acima foram detalhados os comandos AT necessários para o desenvolvimento do projeto.

3.1.4. CONFIGURAÇÃO DO ARDUINO

Para implementação do projeto foi utilizado uma placa Arduino Uno Rev3 mostrado na Figura 2.2.

O código fonte do projeto é desenvolvido em um Software IDE – Integrated Development Environment que permite a criação de sketches para placas Arduino em uma linguagem simples, modelada a partir da linguagem Processing. A figura 3.4 mostra o ambiente de desenvolvimento, nessa tela temos acesso ao serial monitor por meio do botão indicado na figura que nos permitirá acompanhar o funcionamento do programa.

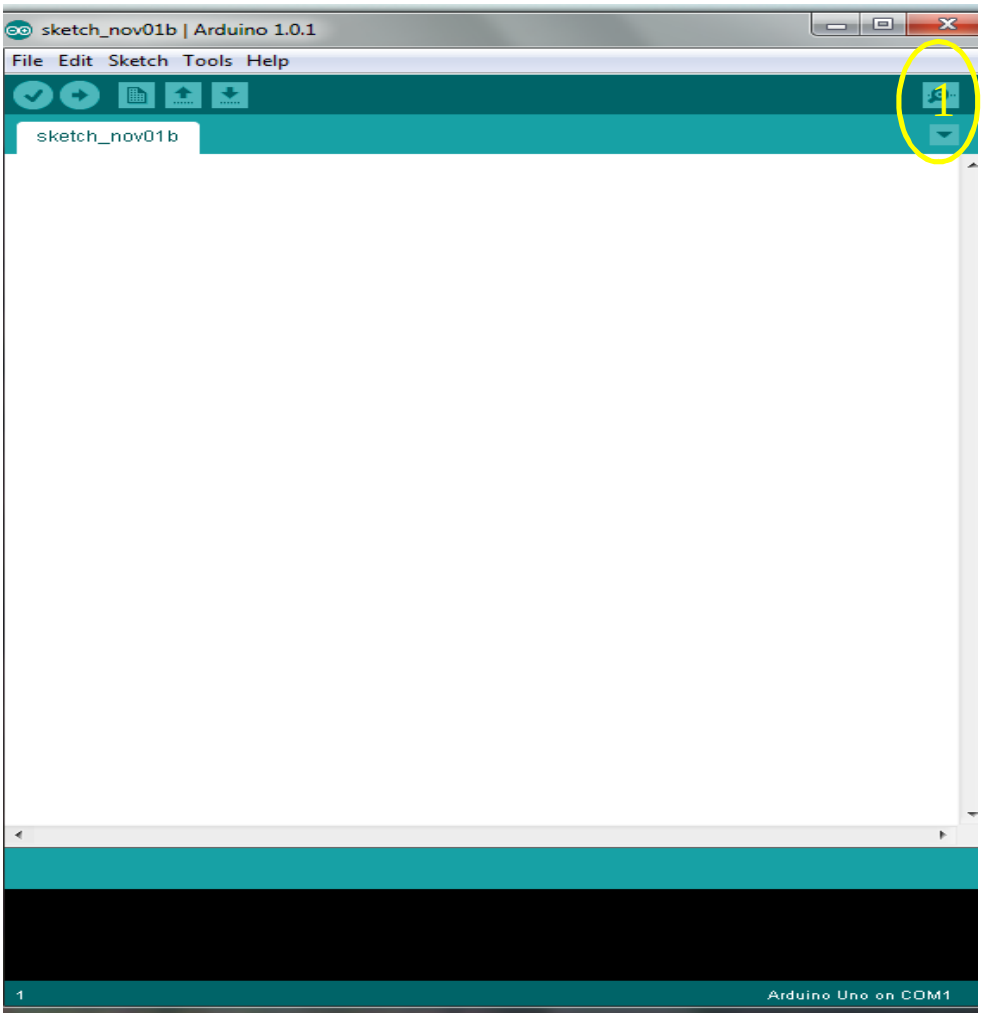

Figura 3.4 – Software IDE
(Fonte: Autor, 2012**)**

3.1.5. ESTRUTURA PRINCIPAL DO PROGRAMA

O programa é dividido em dois blocos principais e seguido de funções que serão executadas de acordo com a necessidade. O primeiro bloco o void setup() é configurado os pinos necessários para a execução dos periféricos e, a seguir, as velocidades de transmissão com que a Hardware Serial e a Software Serial irão se comunicar, serão configuradas, assim como, será definido o modo de leitura da SMS e o tipo de SMS que o módulo irá trabalhar. O trecho de código a seguir mostra como essa configuração foi feita.

```
Void setup()
{
pinMode(statuspin, INPUT);
```

```
pinMode(9, OUTPUT);
pinMode(ledligado, OUTPUT);
pinMode(ledbutton, OUTPUT);
pinMode(ledsensor, OUTPUT);
pinMode(buz,OUTPUT);
gsm.begin(9600);            // Serial baud rate
Serial.begin(9600);         // Serial baud rate
gsm.println("AT+CNMI = 3,3,0,0,0");
delay(2000);
gsm.println("AT+CMGF=1");
...
}
```

Concluindo a fase de configuração de pinos e de comandos iniciais do modem entra-se na função principal do código o void loop(); que ficará em espera aguardando se algum bit entrou pelas Portas Seriais, checando se alguma SMS chegou, verificando se o modem está ligado e fazendo a análise dos sensores. O trecho de código a seguir mostra essa ação.

```
Void loop()
{
if(Serial.available())          //verifica se existe algum bit na Hardware serial
{
gsm.print((unsigned char)Serial.read()); //escreve na S.Serial o que foi lido  na H.Serial
}
else  if(gsm.available())    //verifica se existealgum bit na Software Serial
{
Serial.print((unsigned char)gsm.read());Escrevena Hardware Serial
}

poweron();                   //Chama a função modulo_ligado
checksms();                 //Faz o tratamente da SMS
delay(500);
PUSH_BUTTON();        //Faz a leitura do Push_Button
delay(500);
```

```
SENSOR_PRESENCA(); //Faz a leitura do Sensor de presença
delay(500);
};
```

Nos itens a seguir é mostrado o tratamento dado a cada um dos sensores e atuadores presentes no trio elétrico do carro.

3.1.6. VIDROS

No momento em que o microcontrolador identifica o valor 0005 ou 0006 ele começa a executar a rotina referente aos vidros. Como primeiro passo será verificado o estado do alarme e se o mesmo estiver ativado ele enviará uma SMS ao usuário informando que a ação não pode ser concluída, pois o alarme está ligado, caso contrário, ele entrará na rotina de abertura dos vidros. O código a seguir mostra a função para abertura ou fechamento dos vidros, o sistema informará ao usuário, se os vidros estão abertos ou se os vidros estão fechados.

```
validacodigo()
Serial.println("CODIGO  CADASTRADO");
if(digitalRead(ledligado) == 1){
gsm.println("AT+CMGS=\"+556196252800\"\r");  // Numero para o Envio da SMS
delay(1000);
gsm.println("O Alarme esta ligado");  // Começo da SMS
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{
if(digitalRead(ledvidros) == 1){
gsm.println("AT+CMGS=\"+556196252800\"\r");  //Numero de envio da SMS
delay(1000);
```

```
gsm.println("Os vidros ja 40rese fechados");  // Começo da escrita da SMS
delay(1000);
delay(200);
gsm.println((char)26);
leserial();             //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{
motor();
delay(1000);
gsm.println("AT+CMGS=\"+556196252800\"\r");  //Numero de envio da SMS
delay(1000);
gsm.println("Vidros Fechados");  // Começo da escrita da SMS
delay(1000);
gsm.println((char)26);
leserial();             //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
}
}
if(!strcmp(codcadastrado6, codremetente)){
Serial.println("O CODIGO E' IGUAL AO CADASTRADO");
if(digitalRead(ledligado) == 1){
gsm.println("AT+CMGS=\"+556196252800\"\r");  //Numero de envio da SMS
delay(1000);
gsm.println("O Alarme esta ligado");  // Começo da escrita da SMS
```

```
delay(200);
gsm.println((char)26);
leserial();             //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{
if(digitalRead(ledvidros) == 0){
gsm.println("AT+CMGS=\"+556196252800\"\r");  //Numero de envio da SMS
delay(1000);
gsm.println("Os vidros ja 41rese abertos");  //Começo da escrita da SMS
delay(1000);
delay(200);
gsm.println((char)26);
leserial();             //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{
motor_volta();
delay(1000);
gsm.println("AT+CMGS=\"+556196252800\"\r");  // Numero de Envio da SMS
delay(1000);
gsm.println("Vidros Abertos");  // Começo da escrita da SMS
delay(1000);
delay(200);
gsm.println((char)26);
leserial();             //processa dados e armazena no buffer
```

```
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
}
}
}
else{
Serial.print("Mensagem Invalida");
gsm.println("AT+CMGS=\"+556196252800\"\r");  // Numero de Envio da SMS
delay(1000);
gsm.println("Codigo Invalido");  //Começo da escrita da SMS
delay(1000);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
```

3.1.7. PORTAS

No momento em que o microcontrolador identifica o valor 0003 ou 0004 ele começa a executar a rotina referente às portas. Como primeiro passo será verificado o estado do alarme e se o mesmo estiver ativado ele enviará uma SMS ao usuário informando que a ação não pode ser concluída, caso contrario, ele entrará na rotina de abertura de portas. O código a seguir mostra a função para abertura ou fechamento dos das portas.

```
Serial.println("CODIGO CADASTRADO");
if(digitalRead(ledligado) == 1){
```

```
gsm.println("AT+CMGS=\"+556196252800\"\r");      // Numero de envoi da SMS
delay(1000);
gsm.println("O Alarme esta ligado");  // Começo da escrita da SMS
delay(200);
Serial1.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{
if(digitalRead(ledportas) == 0){
digitalWrite(ledportas,HIGH);
gsm.println("AT+CMGS=\"+556196252800\"\r");
delay(1000);
gsm.println("Portas Travadas ");  //Começo da escrita da SMS        delay(1000);
delay(200);
gsm.println((char)26);
leserial();                              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{
gsm.println("AT+CMGS=\"+556196252800\"\r");  // Numero de envio da SMS
delay(1000);
gsm.println("Portas ja 43rese Travadas ");  //Começo da escrita da SMS
delay(1000);
delay(200);
gsm.println((char)26);
```

```
leserial();             //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
}
}
if(!strcmp(codcadastrado4, codremetente)){
Serial.println(" CODIGO  CADASTRADO");
if(digitalRead(ledligado) == 1){
gsm.println("AT+CMGS=\"+556196252800\"\r");     // Começo da escrita da SMS
delay(1000);
gsm.println("O Alarme esta ligado");  // Começo da escrita da SMS
delay(200);
gsm.println((char)26);
leserial();             //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{
if(digitalRead(ledportas) == 1){
digitalWrite(ledportas,LOW);
gsm.println("AT+CMGS=\"+556196252800\"\r");  // Numero de envio da SMS
delay(1000);
gsm.println("Portas Destravadas ");  // Começo da escrita da SMS
delay(1000);
delay(200);
gsm.println((char)26);
leserial();             //processa dados e armazena no buffer
```

```
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{
gsm.println("AT+CMGS=\"+556196252800\"\r");  // Numero para envio da SMS,
delay(1000);
gsm.println("Portas ja 45rese destravadas ");  // Começo da escrita da SMS delay(1000);
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
}
```

3.1.8. ALARME

No momento em que o microcontrolador identifica o valor 0001 ou 0002 – ativar ou desativar – ele começa a executar a rotina referente ao alarme e chama as rotinas de fechamento de vidros e travamento das portas. O código a seguir mostra a função de acionamento do alarme.

```
If(!strcmp(codcadastrado1, codremetente)){
Serial.println(" CODIGO CADASTRADO");
digitalWrite(ledligado,HIGH);
digitalWrite(ledportas,HIGH);
delay(1000);
if((digitalRead(ledvidros) == 0)){
motor();
```

```
delay(1000);
}
delay(1000);
gsm.println("AT+CMGS=\"+556196252800\"\r");  // Numero de Envio da SMS
delay(1000);
gsm.println("Alarme Ativado");  // Começo da escrita da SMS
delay(1000);
gsm.println((char)26);
leserial();             //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
if(!strcmp(codcadastrado2, codremetente)){
Serial.println(" CODIGO CADASTRADO");
digitalWrite(ledligado,LOW);
digitalWrite(ledportas,LOW);
Serial1.println("AT+CMGS=\"+556196252800\"\r");  // Numero de envio da SMS
delay(1000);
Serial1.println("Alarme Desativado");  //Começo da escrita SMS
delay(1000);
msendbutton = false;
delay(200);
msendsensor = false;
delay(200);
Serial1.println((char)26);
leserial();             //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
```

Nesse momento passou a montagem do protótipo, interligando os periféricos necessários para a simulação (motor de passo, leds, sensor de presença e push button) ao conjunto formado pelo Arduino UNO e o Módulo SIM900 e foi testando cada funcionalidade em acordo com o que foi descrito acima, e em cada momento foi feita a verificação da tensão e corrente em cada componente do circuito. No próximo cápitulo mostraremos cada etapa dos testes realizados.

CAPÍTULO 4 – MONTAGEM, TESTES E EXPERIMENTOS.

4.1. INTRODUÇÃO

Este capítulo tem como finalidade mostrar os resultados obtidos com a implementação do sistema de controle e monitoramento do alarme via SMS.

Primeiramente, configura-se e verifica o envio e recebimento de SMS por meio do modem através de um PC.

Testou-se a comunicação serial entre os dois equipamentos feitos através de rotinas de programação e nesse mesmo momento foram testados os comandos AT que serão utilizados no projeto.

Em seguida foi testado o Arduino Uno Rev3 e todos os periféricos necessários, de forma independente, para simulação e a seguir o programa é carregado para que seja possível depurar qualquer erro.

Concluídas as fases acima, partiu-se para os testes do protótipo, nessa fase foi testado o envio e recebimento de SMS e verificado se a ação solicitada foi executada.

4.2. CONFIGURAÇÃO E MONTAGEM DO AMBIENTE DE TESTE

4.2.1. MONTAGEM DO PROTOTIPO

Como mencionado anteriormente, cada função do trio automotivo será simulada através de um dispositivo com funcionamento semelhante. O quadro 4.1 descrimina os itens necessários para a implementação do projeto, assim como o sua função.

Quadro 4.1 – Componentes do projeto e a sua função

Componentes	Função
Arduino Uno Rev3	Controle do Sistema
Shield SIM900 IcomSat v1.1	Habilitar a Rede GSM
Led Vermelho	Alarme Ligado
Motor de Passo + Led Verde	Vidros
Sensor Pir + Led Amarelo	Sensor de Presença do Carro
Push Button + Led Amarelo	Pinos da Portas e Capo do Carro
Protoboard	Suporte e ligação dos Componente
Fios Jumpers	Ligação dos Componentes
Led Verde	Trava das Portas

(**FONTE:** Autor,2012)

A figura 4.1 a seguir mostra uma foto do protótipo depois de montado, nele podemos observar os componentes ligados, no centro temos o arduino uno com o shield SIM900 acoplado, ele será responsável pelo controle do sistema e pelo recebimento das instruções por meio de SMS, resolvi dividir o Prototipo em duas protoboards para facilitar a visualização, na placa da esquerda temos o leds que irão representar o Alarme Ligado(vermelho), Portas travadas(led verde), Vidros Fechados(Led verde), Ativação de sensor de portas-push button – (led amarelo) e ativação do sensor de vidros-sensor PIR – (led amarelo). Na placa esquerda temos o motor de passo com o Driver ULN2003APG que representara a subida ou descida dos vidros, o push button que representara a sensor de portas, o sensor PIR que representará o sensor de pressão de ar e uma fonte de energia.

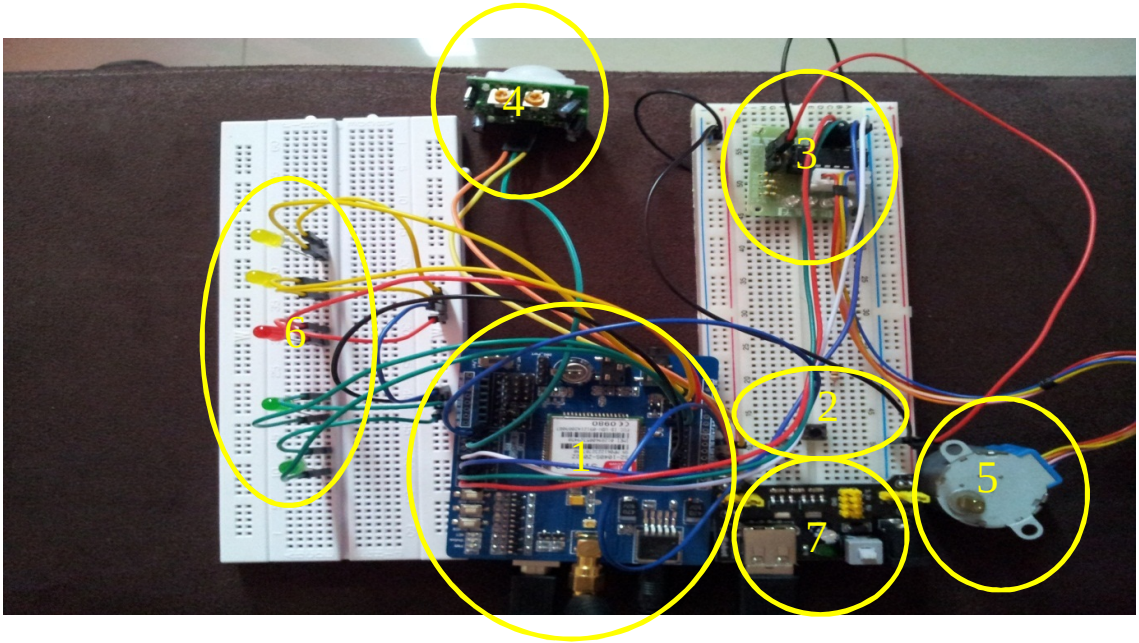

Figura 4.1-Protótipo do Sistema

(**FONTE**: Autor, 2012)

1. Arduino Uno + Shield SIM900;
2. Push Button;
3. Shield ULN2003APG;
4. Sensor PIR;
5. Motor de Passo;
6. Leds;
7. Fonte Externa de Protoboard.

4.2.2. ARDUINO + SHIELD SIM900

O ambiente de teste estabelecido nessa etapa foi feito utilizando um Arduino Uno, um Shield GSM SIM 900, um Celular e o IDE de programação.

Os códigos utilizados nessa fase serviram para verificar o funcionamento do Arduino em conjunto com o Shield e de que forma a SMS seria enviada e recebida pelo microcontrolador através do SIM900.

Primeiramente utilizou-se um código fornecido como exemplo pela biblioteca SIM900.h que tem como principal função verificar se o shield está ligado e enviar uma SMS ao usuário com o seguinte corpo: “Arduino SMS”.

Após isso, utilizou-se outro código exemplo que tem como função o teste de comandos AT, nessa etapa analisa-se e escolhem-se os comandos AT que serão utilizados no projeto. A seguir o código utilizado nessa etapa.

```
#include “SIM900.h”
#include <SoftwareSerial.h>
intnumdata;
charinseria[40];
inti = 0;
void setup(){
Serial.begin(9600);
Serial.println(“GSM Shield testing.”);
If(gsm.begin(9600))
Serial.println(“\nstatus=REDY”);
Else Serial.println(“\nstatus=IDLE”;)
};
void loop()
{
serialhwread();     //lê um novo bte na Hardware Serial e escreve na Software serial
serialswread();    //lê um novo byte na Software Serial
}
voidserialhwread(){
i=0;
```

```
if(Serial.avaliable () > 0){
while(Serial.avaliable () >0){
inSerial[i]=(Serial.read());
delay(10);
                    i++;
}
inSerial[i]=´\0´;
if(strcmp(inseria, “/END”)){
Serial.println(“_”);
51resenc[0]=0x1a;
51resenc[1]=´\0´;
gsm.SimpleWriten(51resenc);
}

//Envia o comando AT salvo usando a porta serial
If(!strcmp(inseria, “TEST”)){
Serial.println(“SIGNAL QUALITY”);
gsm.SimpleWriteln(“AT+CSQ”);
}
else{
Serial.println(inserial);
gsm.SimpleWriteln(inSerial);
}
inSerial[0]=\’0’;
}
}
voidserialswread(){
gsm.SimpleRead();
}
```

4.2.3. MOTOR DE PASSO

Nessa fase da montagem optou-se, primeiramente, pelo uso do motor de passo em conjunto com o Arduino, após verificar que o seu funcionamento estava correto, insere-se o Shield e testou seu controle por meio de SMS.

O motor escolhido para a simulação dos vidros foi o 28BYJ – 48 que possui quatro passos. Para o correto funcionamento do motor a amperagem e a tensão em cada ponto tem que estar correta, conforme especificação. A amperagem em cada pino do Aduino é baixa para o funcionamento do motor sendo necessário a utilização de um Drive, esse intermédio é feito através do Drive ULN2003APG ,que serve para fornecer correntes elétricas muito maiores do que as que o chip Atmega do Arduino pode oferecer.

A figura 4.2 mostra o motor de passo e o drive utilizado.

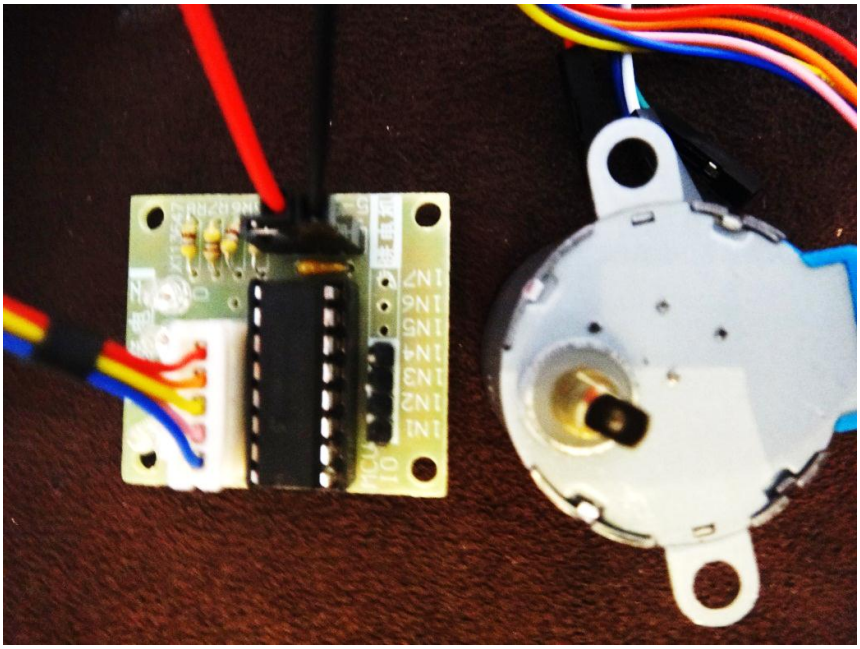

Figura 4.2 – Motor de Passo e ULN2003APG

(**FONTE**: Autor, 2012)

O código a seguir foi utilizado para os testes do motor, caso a função *myStepper.step* tenha seu valor positivo o motor gira em sentido anti-horário se o seu valor for negativo o motor gira em sentido horário.

```
#include <Stepper.h>
constintpassosVoltaCompleta = 2048;  // numero de passos por volta completa
int v = 0;
void loop(){
SteppermyStepper(passosVoltaCompleta,10,12,13,11);
myStepper.setSpeed(1);
```

myStepper.setSpeed(12);
myStepper.step(3000);

4.2.4. SENSOR PIR

No projeto utiliza-se o sensor PIR para simular a entrada no interior do veiculo, por quebra dos vidros. Normalmente, em sistemas de alarme comum se utiliza um microfone que envia um sinal elétrico decorrente da pressão do deslocamento do ar no interior do veículo. Devido à dificuldade na simulação do deslocamento de ar, optou-se pelo sensor de presença que possui a função de enviar um sinal elétrico caso o infravermelho seja cortado.

O sensor PIR possui um alcance de 7 m e um ângulo de atuação de até 100°. Como o sistema tem como premissa a leitura ou não dos dados de um sensor, a tese para utilização do sensor presente no veiculo será igual.

Nos primeiros testes verificou-se que a sensibilidade do sensor atrapalhava no funcionamento do software sendo necessário focar o infravermelho, pois seu ângulo de atuação é de 100º, então, qualquer movimento ao redor do sensor estava disparando o sistema e travando o Arduino, necessitando do reinicio forçado da placa.

Verificou-se que o shield do sensor necessita de um tempo de 20 segundos para que o carregamento dos capacitores esteja completo.

A figura 4.3 mostra o sensor já conectado ao Arduino. Assim que o LED verde é acionado indica que o feixe foi interrompido.

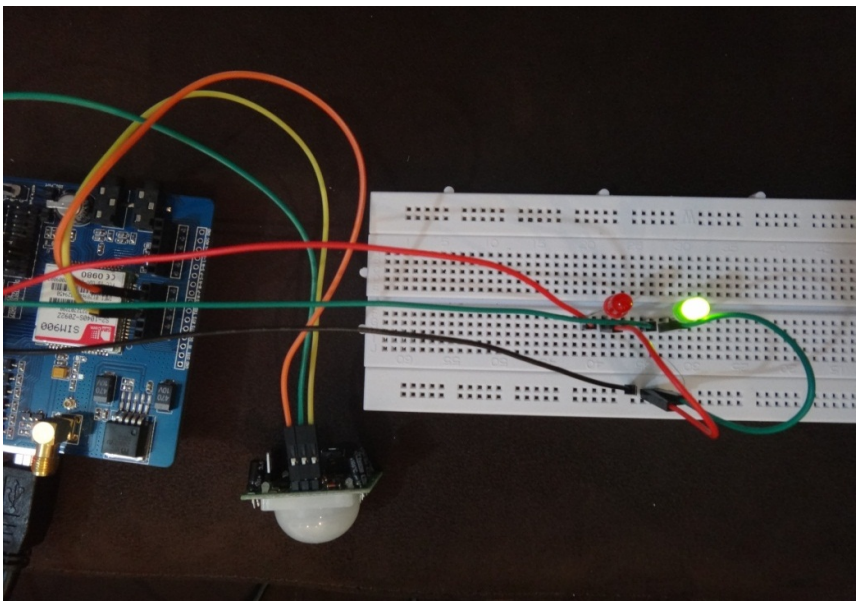

Figura 4.3 – Sensor PIR em funcionamento

(**FONTE**: Autor, 2012)

O código a seguir foi utilizado para os testes com o sensor pir:

```
Int ledVermelho = 13;          //Pino para o LED
Int ledVerde = 12;             //Pino para o LED
Int PIRPin = 7;                //Pino OUT do PIR
int PIR = 0;                   //Variável que representa o valor fornecido pelo sensor PIR
void setup() {
pinMode(ledVermelho, OUTPUT);
pinMode(ledVerde, OUTPUT);
pinMode(PIRPin, INPUT);
}
void loop(){
PIR = digitalRead(PIRPin);     // Faz leitura do OUT do sensor
if (PIR == HIGH) {             // Testa se a variável "PIR" é alta(5v),
digitalWrite(ledVermelho, HIGH);     // Liga led Vermelho
digitalWrite(ledVerde, LOW); // Pino do led Verde torna-se o GND(terra) do led Vermelho
}
else{
digitalWrite(ledVermelho, LOW);
digitalWrite(ledVerde, HIGH);       // Led Verde ON
```

4.2.5. ALARME E TRIO-ELÉTRICO

Assim que os sensores foram calibrados, foi verificada a necessidade de condicionar o uso ou não dos dados fornecidos pelas leituras do Sensor PIR e do PUSH_BUTTON. Então se condicionou que as leituras somente seriam utilizadas se o LED vermelho estivesse ativado.

Mas antes deste passo foi necessário um estudo do comportamento dos sensores em conjunto com o Arduino. Para esta fase criou-se o código a seguir, que tem como função acionar um LED via porta serial e retornar o valor da leitura do sensor, caso o LED esteja ligado.

```
// TESTE 32: Dois Sensores mais motor de passo
```

```
#include <Stepper.h>
constintledsensor = 13; //pino para abertura de portas
constintledalarme = 12; // pino para o LED Vermelho
constintledbutton = 11; //pino para o Led Verde
constintledportas = 10; //pino para o LED Amarelo
constintledvidros = 4; //
constint BUTTON = 8; // pino da entrada na qual o botão de pressão está conectado ao LED vermelho
constint SENSOR = 7; // pino da entrada na qual o botão de pressão está conectado ao LED verde
constintpassosVoltaCompleta = 2048;  // numero de passos por volta completa
intvalbutton = 0;          //val será utilizada para armazenar o estado do pino de entrada
intold_valbutton = 0;      //variáriavel para o BUTTON
intstatebutton = 0;       //55resence de estado do BUTTON
intvalsensor = 0;          //val será utilizada para armazenar o estado do pino de entrada
intold_valsensor = 0;      //variáriavel para o Sensor de presença
intstatesensor = 0;       //55resence de estado do Sensor de presença
booleanmsend = false;      //Boleano para o envio da SMS
int n;       //inteiro para a leitura serial(55resence do Switch_Case)
int x;      //inteiro para a leitura serial
void setup() {
pinMode(ledvidros, OUTPUT);
pinMode(ledsensor, OUTPUT);
pinMode(ledalarme,OUTPUT);       //Avisa ao Arduino que LED é uma saída
pinMode(ledportas, OUTPUT);      //Avisa ao Arduino que o LED é uma saída
pinMode(ledbutton, OUTPUT);      //Avisa ao Arduino que o LED é uma saída
pinMode(SENSOR, INPUT);     //Avisa ao Arduino que o SENSOR é uma Entrada
pinMode(BUTTON, INPUT);     //Avisa ao Arduino que o BUTTON é uma entrada
Serial.begin(9600);        //Seta a velocidade de inicio da Serial
Serial.println("Teste Serial");
}
void loop() {
if(Serial.available() > 0) {
n = Serial.read();
```

```
x = digitalRead(ledalarme);
switch(n) {
case ‘1’: //Ligar
if(x == LOW){
digitalWrite(ledalarme, HIGH);
digitalWrite(ledportas, HIGH);
delay(1000);
if(digitalRead(ledvidros) == 0){
motor();
digitalWrite(ledvidros, HIGH);
}
Serial.println(“Alarme Ligado”);
}
break;
case ‘2’: //desligar
if(x == HIGH){
digitalWrite(ledalarme, LOW);
digitalWrite(ledportas,LOW);
msend = false;
Serial.println(“Alarme Desligado”);
delay(1000);
}
break;
case ‘3’: //ligar
if(x == HIGH){
Serial.println(“Alarme Ligado”);
}
else{
if(digitalRead(ledportas == 0)){
digitalWrite(ledportas, HIGH);
Serial.println(“Portas Fechadas”);
}
}
delay(1000);
```

```
break;
case ‘4’: //desligar
if(x == HIGH){
Serial.println(“Alarme Ligado”);
}
else{
if(digitalRead(ledportas)==1){
digitalWrite(ledportas,LOW);
Serial.println(“Portas Destravadas”);
}
delay(1000);
}
break;
case ‘5’: //desligar
if(x == HIGH){
Serial.println(“Alarme Ligado”);
}
else{
if(digitalRead(ledvidros) == 1){
Serial.println(“vidros fechados”);
}
else{
motor();
digitalWrite(ledvidros, HIGH);
delay(1000);
}
}
break;
case ‘6’: //desligar
if(x == HIGH){
Serial.println(“Alarme Ligado”);
}
else{
if(digitalRead(ledvidros) == 0){
```

```
Serial.println("vidros Abertos");
}
else{
motor();
digitalWrite(ledvidros, LOW);
delay(1000);
}
}
break;
// default: //codigoinvalido
// Serial.println("Codigo Invalido");
}
}
SENSOR_PRESENCA();
delay(1000);
PUSH_BUTTON();
};
//#############################################################################

void SENSOR_PRESENCA(){
valsensor = digitalRead(SENSOR); //lê e armazena o valor da entrada
if((valsensor == HIGH)&&(old_valsensor == LOW)){  //verifica se a saída está em HIGH(botào pressionado)
statesensor = 1-statesensor;
delay(10);
}
old_valsensor == valsensor;
if(statesensor == 1){
digitalWrite(ledsensor,HIGH); //Acende o LED
}
else{
digitalWrite(ledsensor,LOW); //caso contrário o LED permanece apagado
}
delay(1000);
```

```
smssensor_presenca();
};
//#############################################################################

void PUSH_BUTTON(){
valbutton = digitalRead(BUTTON); //lê e armazena o valor da entrada
if((valbutton == HIGH)&&(old_valbutton == LOW)){  //verifica se a saída está em HIGH(botào pressionado)
statebutton = 1-statebutton;
delay(10);
}
old_valbutton == valbutton;
if(statebutton == 1){
digitalWrite(ledbutton,HIGH); //Acende o LED
}
else{
digitalWrite(ledbutton,LOW); //caso contrário o LED 59resence59e apagado
}
delay(1000);
smspush_button();
};
//#############################################################################
void smssensor_presenca() {
if(digitalRead(ledalarme) == 1){
if(digitalRead(ledsensor) == 1 &&msend == false ){
Serial.println("INTRUSAO PELA QUEBRA DE VIDROS");
msend = true;
}
if(digitalRead(ledalarme) == 0){
msend = false;
}
}
};
//#############################################################################
```

```
voidsmspush_button(){
if(digitalRead(ledalarme) == 1){
if(digitalRead(ledbutton) == 1 &&msend == false ){
Serial.println("INTRUSAO PELA PORTA");
msend = true;
}
if(digitalRead(ledbutton) == 0){
msend = false;
}
}
};
//#############################################################################

void motor(){
SteppermyStepper(passosVoltaCompleta,A1,A3,A2,A0);
myStepper.setSpeed(1);
myStepper.setSpeed(12);
myStepper.step(3000);
};
voidmotor_volta(){
SteppermyStepper(passosVoltaCompleta,A1,A3,A2,A0);
myStepper.setSpeed(1);
myStepper.setSpeed(12);
myStepper.step(-3000);
}
```

Concluído o teste dos sensores e atuadores montou-se o sistema e se fez a medição da tensão de entrada e saída de cada sensor e atuador, assim como foi feita a aferição da corrente em cada componente do circuito.

Após isso, foram feitos testes no código do sistema para verificação de suas funcionalidades e seus pontos críticos. A figura 4.4 mostra a tela do Serial Monitor no momento em que a SMS é recebida, tratada e a SMS de Status é enviada ao usuário do sistema.

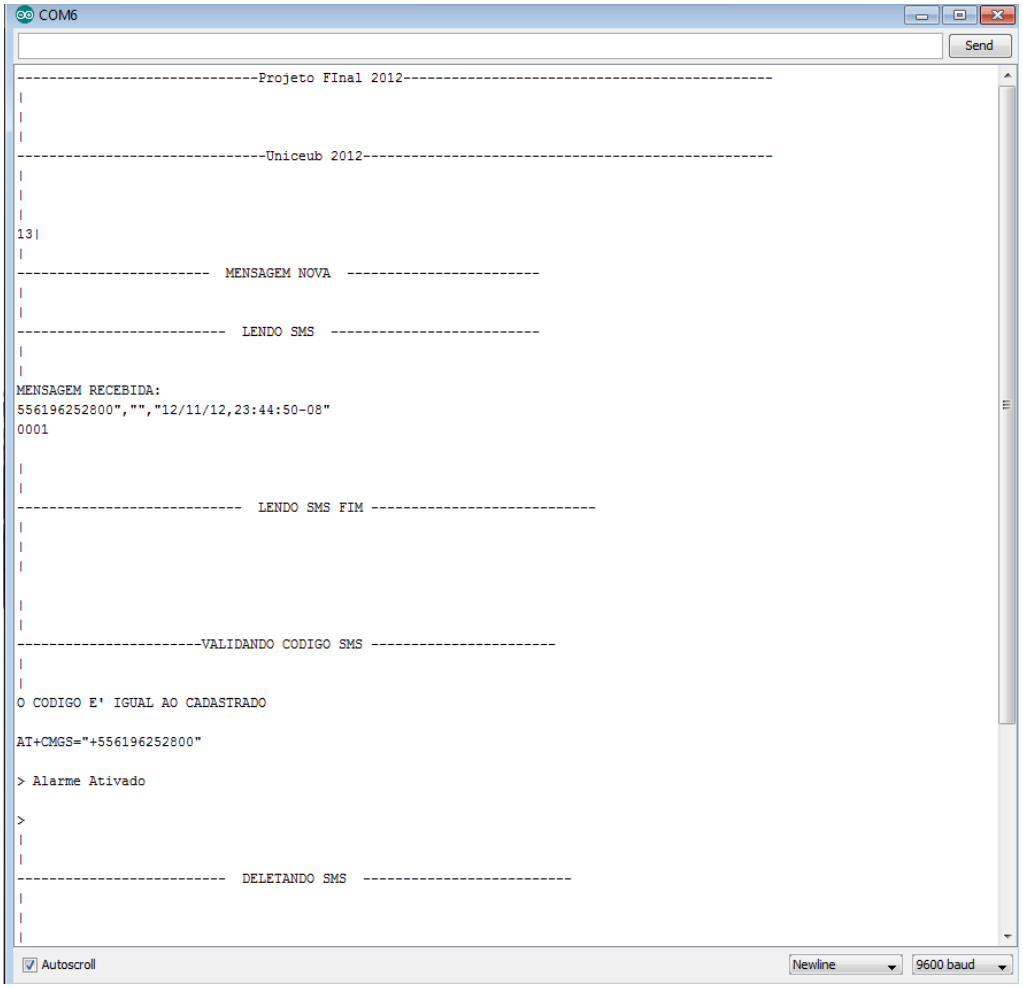

Figura 4.4 – Serial Monitor
(**FONTE**: Autor, 2012)

4.3. DIFICULDADES ENCONTRADAS

Este item descreve as dificuldades encontradas, explicando a causa da troca de componentes e suas soluções.

4.3.1. CELULAR

Inicialmente pensou-se na utilização de um celular com suporte a comandos AT para a comunicação entre o modem e o microcontrolador. Atualmente, poucos celulares possuem seu funcionamento baseado em comandos AT e a porta serial foi substituída pela USB e os celulares que possuem suporte aos comandos AT foram descontinuados pelos seus fabricantes. Logo, para utilização do aparelho seria necessário primeiro encontrar um que

possibilite o uso de comandos AT e ainda tenha a entrada serial como forma de conexão do aparelho, impossibilitando a produção do protótipo. Outro ponto considerado e que no caso do aparelho ser encontrado o projeto ficaria dependente deste modelo de aparelho, tornando menos maleável as atualizações tecnológicas.

Outro problema encontrado com a utilização de um celular para a comunicação foi que cada modelo de celular tem funções especificas de comandos AT delimitadas pelo fabricante do aparelho, o que delimitaria o projeto dependendo do modelo escolhido.

O fato de o celular limitar o projeto e restringir a atualização do protótipo fez com que fosse feita a escolha de um módulo GSM. Módulo que possui o foco ligado em aplicações de controle, de aplicações industriais e aplicações similares às apresentadas no projeto.

4.3.2. LEITURA DOS DADOS SERIAIS

Outro ponto de grande dificuldade foi à leitura dos dados seriais, pois se o controle de dados no buffer da entrada serial, tanto a software serial como a hardware serial, não fosse feito o retorno dos dados poderia ser inesperado, pois o código tem como premissa retornar o valor da String na tela do monitor serial, então muito “lixo” seria escrito na tela causando um retorno diferente do esperado.

4.3.3. CONTROLE DO NÍVEL DE CORRENTE DO SISTEMA

Um ponto de grande dificuldade foi o controle da corrente no sistema, pois se a fonte de alimentação externa for abaixo de 2 ampères o sistema entra no loop para testar se o módulo GSM está ligado e não consegue entrar no loop principal do programa que consiste na entrada de bits pela portas seriais.

4.3.4. TROCA DO PIC PELO ARDUINO

Outro grande problema enfrentado no projeto foi a utilização de um PIC 16F877A, que apesar de ser um microcontrolador muito versátil a forma de programação (em C ou Assembler) dificultava demais o uso por um novo usuário. O Arduino já segue em direção contrária, apesar da linguagem ser baseada em C/C++ ela é intuitiva. Outro ponto que pesou pela escolha do Arduino foi à vasta comunidade na Internet que o utiliza, pois como seus projetos e códigos são livres e sua principal funcionalidade é a prototipagem e a experimentação, conclui-se que é uma escolha adequada.

4.3.5. IMPLEMENTACÃO DO SISTEMA NO VEÍCULO

A grande dificuldade encontrada pelo projeto foi á implementação do sistema em um carro. Abaixo são listados os motivos que levaram a simular o sistema:

- Falta de Orçamento para o projeto.
- Cada carro possui uma forma diferente de interpretação dos protocolos CAN, que são de propriedade das montadoras, sendo necessário o contato com elas e o fornecimento de informações necessárias para a implementação do sistema.
- Além do fator acima citado existe a possibilidade da inserção de um shield que possibilitaria ao Arduino a comunicação com os protocolos CAN, mas seria necessário a aquisição de um modelo especifico de acordo com o carro.
-

4.3.6. CUSTOS DO PROJETO

O quadro 4.2 mostra o custo do projeto, buscou-se uma solução barata e com equipamentos que são facilmente encontrados tanto no mercado nacional, a maioria dos equipamentos foi comprada no mercado internacional o que reduziu o custo do projeto em mais da metade do valor.

Equipamento	Local de Compra	Valor no mercado Internacional ($)	Valor no Mercado Nacional (R$)
Arduino Uno Rev 3	DealExtreme	14,58	69,90
Shield IcomSat 1.1	Ebay	39,50	199,90
Protoboard 850 pontos	DealExtreme	3,90	14,9
Protoboard 830 pontos	DealExtreme	2,90	12,90
Motor de Passo	DealExtreme	3,50	34,99
Pacote com Leds	DealExtreme	0,99	4,50
Pacote com fios Jumper	DealExtreme	2,00	24,50
Sensor PIR	Ebay	4,40	25,90
Push Buton	DealExtreme	0,45	1,70
Fonte Externa de Protobord	DealExtreme	7,9	24,90
Total do Projeto(Reais)	-	177,70	414,09

Quadro 4.2 – Custos do Projeto

Fonte(Autor, 2012)

CAPÍTULO 5 - CONCLUSÕES

Este projeto tem como finalidade realizar um protótipo que alerte o usuário no caso de disparo do alarme automotivo, possibilitando a interação do usuário com as funções existentes em trios-elétricos que acionam vidros elétricos, travas elétricas e alarme sonoro, através de um celular e um Arduino em conjunto com um módulo GSM.

Com a simulação realizada, foi possível visualizar que é possível o controle de atuadores e a leitura de sensores por meio de comandos enviados via SMS, sendo assim, possível a implementação do sistema em qualquer ambiente em que o funcionamento seja feito por meio de atuadores e sensores, como é o caso do trio elétrico automotivo.

O sistema foi projetado para informar a partir do dado lido qual sensor foi ativado e existindo a possibilidade da inserção de novos sensores. Assim o usuário receberá informações precisas sobre os fatos.

Conclui-se que o objetivo de construir um protótipo para realizar a comunicação entre o veículo e seu usuário através de um módulo GSM, foi realizado com sucesso e foi criado um protótipo adaptável, podendo ser modificado e levado para outras situações como: alarmes residenciais, alarmes comercias, acionamento remoto de dispositivos, dentre outros.

A função do microcontrolador é importante e por possuir diversos pinos de entrada e saída possui possibilidades de evolução e integração com o veiculo, incluindo novas funções: como o controle do ar condicionado, dos faróis e a possibilidade de acionar o veiculo remotamente. Automatizando ainda mais as funções de controle de seu carro.

5.4. PROPOSTAS FUTURAS

5.4.1. INTERLIGAÇÃO COM MAIS SENSORES DO VEÍCULO

Como o projeto mostra a integração na leitura dos sensores por meio de SMS é possível adicionar outros sensores presentes no veículo, sendo possível o usuário possuir em seu celular informações presentes no computador de bordo do veiculo, como: nível de gasolina, autonomia do carro, consumo, temperatura externa e interna no carro dentre outras.

5.4.2. CRIAÇÃO DE SOFTWARE PARA CELULAR

A constante evolução dos celulares, da linguagem Java e a constante evolução de sistemas como o Android e o IOS tornam possível a criação de diversas aplicações para celulares, o que facilita a utilização do sistema pelo usuário.

Pode ser criado software para que a funções do sistema se tornem intuitivas levando à utilização do sistema por qualquer usuário que possua um celular ou smartphone.

Uma analise de como essa implementação deverá ocorrer deve ser realizada, verificando a tarifação da operadora, se é necessário a utilização de um servidor ou a utilização de pacotes de GPRS.

5.4.3. INTERLIGAÇÃO COM MAIS MÓDULOS

Uma expansão física no sistema, acrescentando um módulo GPS, permitiria ao usuário obter uma localização do automóvel, através de programas já disponíveis no mercado, como exemplo o Google Maps, sendo necessário o devido cuidado na instalação do sistema para que possa operar com uma bateria recarregável ou através da bateria do carro, e tendo o cuidado de evitar ruídos no sistema, elaborando um sistema eficaz de blindagem para a proteção do sistema.

Outra possibilidade e a inserção de um módulo bluetooth, no projeto o que permitiria o controle das travas e vidros do carro quando o usuário estivesse próximo ao carro facilitando a entrada no carro sem a necessidade de envio de uma SMS.

5.4.4. APLICAÇÕES EM OUTRAS ÁREAS

Como dito anteriormente devido a maleabilidade do projeto o mesmo pode ser utilizado em outras gamas de conhecimento, como exemplo podemos citar a saúde, a indústria ou qualquer outro sistema que tenha o funcionamento vinculado ao uso de sensores ou atuadores.

REFERÊNCIAS BIBIOGRÁFICAS

Livros:

BANZI, Massimo. Primeiros Passos com o Arduino. 1ª edição. Brasil: Editora Novatec, 2011.

BOYLESTAD, Robert; NASHELSKI, Louis. Dispositivos Eletrônicos e Teoria de Circuitos. 8ª edição. Rio de Janeiro: Editora Prentice Hall Brasil, 2004. 649 p.

COULOURIS, George; KINDBERG, Tim; DOLLIMORE, Jean. Sistemas Distribuídos: Conceitos e Projeto. 4ª edição. São Paulo: Editora Bookman, 2007. 784 p.

FOROUZAN, Behrouz A. Comunicação de Dados e Redes de Computadores. 3ª edição. Brasil: Editora McGraw Hill , 2006

GUIMARÃES, ALEXANDRE de Almeida. Eletrônica Embarcada Automotiva. 1ª edição. São Paulo: Editora Érica, 2007. 326 p.

LAVINIA, Nicolás C.;SOUZA, David J. Conectando o PIC 16F877A: recursos avançados. São Paulo: Érica, 2003.

MARGOLIS, Michel. Cookbook Arduino, Estados Unidos da América: Editora O'Reilly, 2012.'

MCROBERTS, Michel. Arduino Básico. Brasil, São Paulo. Ed. Novatec, 2012

NICOLOSI, Denys Emílio C. Microcontrolador 8051 Detalhado. 5ª edição. São Paulo: Editora Érica, 2004. 227 p.

PEREIRA, Fábio. *PIC* Programação em C. São Paulo: Ed. Érica, 2003

SOUZA, David José;LAVINIA, Nicolas César. Conectando o PIC Recursos Avançados. São Paulo: Ed. Érica, 2003.

SOUZA, David José;Desbravando o PIC – Ampliado e atualizado para o PIC 16f628A. São Paulo: Ed. Érica, 2003.

THOMAZINI, Daniel; ALBUQUERQUE, Pedro Urbano Braga de. Sensores industriais: fundamentos e aplicações. São Paulo: Érica, 2005.

Internet:

Arduino – Arduino UNO.

Disponível em: <http://arduino.cc/en/Main/ArduinoBoardUno>.

Acesso em: 25 de junho de 2012.

Trio Elétrico Automotivo

Disponível em:< http://www.sabereletronica.com.br/secoes/leitura/994>

Acesso em: 16 de agosto de 2012

Controle remoto de travas Elétricas

Disponível em:< http://carros.hsw.uol.com.br/controle-remoto-de-travas-eletricas1.html>

Acesso em: 28 de agosto de 2012

Módulo SIM900

Disponível em :< http://imall.iteadstudio.com/im120417009.html>

Acesso em: 30 de setembro de 2012

APÊNDICE A – Código fonte de Projeto

```
#include <SoftwareSerial.h>
#include <String.h>
#include <Stepper.h>

SoftwareSerial gsm(4, 5); //TX, RX

#define BUF_LEN  84

int statuspin = 19;
int pinState = 0;
int x,y,i,j,a,b;

const int ledligado = 33;          //Sensor que indicara que o alarme esta ligado
const int ledbutton = 35;          //button que indicara intrusão pelas portas
const int ledsensor = 37;          //button que indicara a quebra de vidros
const int ledportas = 39;          //led que indicara que as portas estão fechadas
const int ledvidros = 41;          //led que indicara que os vidros estão fechados


const int BUTTON = 31;     // pino da entrada na qual o botão de pressão está conectado
const int SENSOR = 52;     // pino da entrada na qual o sensor de presença está conectado
int powerkey = 9;          //pino que liga o modulo

const int passosVoltaCompleta = 2048; //numero de passos por volta

int valsensor = 0;          //val será utilizada para armazenar o estado do pino de entrada
int old_valsensor = 0;      //variáriavel para o Sensor de presença
int statesensor = 0;        //variavel de estado do Sensor de presença

int valbutton = 0; //val será utilizada para armazenar o estado do pino de entrada
int old_valbutton = 0; //variáriavel para o BUTTON
int statebutton = 0; //variavel de estado do BUTTON
```

```
boolean msendbutton = false; //Boleano para o envio da SMS pela intrusão pelas portas
boolean msendsensor = false; //Boleano para o envio da SMS pela intrusão pelos vidros

char codcadastrado1[]= "0001";
char codcadastrado2[]= "0002";
char codcadastrado3[]= "0003";
char codcadastrado4[]= "0004";
char codcadastrado5[]= "0005";
char codcadastrado6[] ="0006";

char ncadastrado[]= "2800";


char codremetente[] = "";
char nremetente[] = "";


const int MAXDATA = 256;
char data [MAXDATA];

void setup()
{

pinMode(statuspin, INPUT);
pinMode(powerkey,  OUTPUT);
pinMode(ledligado, OUTPUT);
pinMode(ledbutton, OUTPUT);
pinMode(ledsensor, OUTPUT);
pinMode(ledportas, OUTPUT);
pinMode(ledvidros, OUTPUT);
pinMode(BUTTON, INPUT);
pinMode(SENSOR, INPUT);
Serial1.begin(9600);              // GPRS baud rate
```

```
Serial.begin(9600);               // GPRS baud rate
Serial1.println("AT+CNMI = 3,3,0,0,0");
delay(2000);
Serial1.println("AT+CMGF=1");
delay(2000);
Serial.println("------------------------------Projeto Final 2012--------------------------------------");
Serial.println("|");
Serial.println("|");
Serial.println("|");
Serial.println("------------------------------Uniceub 2012----------------------------------------");
Serial.println("|");
Serial.println("|");
Serial.println("|");
leserial();
limpa_buffer();

}

byte num_of_bytes;
byte retorno;
char comm_buf[256];
byte n;

void loop() {
if(Serial.available()){  //se tem algum dado na hardware serial
gsm.print((unsigned char)Serial.read());//escreve na software serial
}
else  if(gsm.available()){ //se tem algum dado na software serial
Serial.print((unsigned char)gsm.read()); //escreve na hardware serial
}
poweron(); //chama a função que verifica se o módulo esta ligado
checksms();//verifica se alguma sms foi recebida
push_button();//verifica o button
SENSOR_PRESENCA();//verifica o sensor de presença
```

```
};
//#############################################################################

void checksms(){

if(gsm.available() >0){
Serial.println("|");
Serial.println("|");
Serial.println("------------------------  MENSAGEM NOVA  ------------------------");
gsm.flush();
delay(1000);
lesms();
//validacodigo();
}
};
//#############################################################################
void lesms()
{
Serial.println("|");
Serial.println("|");
Serial.println("--------------------------  LENDO SMS  --------------------------");
Serial.println("|");
Serial.println("|");
delay(2000);
gsm.println("AT+CMGR=1"); //Lê a primeira SMS recebida
gsm.flush();
for (x=0;x < 255;x++){
data[x]='\0';
}
x=0;
do{
while(gsm.available()==0);
data[x]=gsm.read();
x++;
```

```
if(data[x-1]=='+'&&data[x-2]==""){ // remove o cabeçalho +CMGR: "REC
READ","xxxx","","12/05/21,12:21:43-12"
x=0;
}
}
while(!(data[x-1]=='K'&&data[x-2]=='O')); // enquanto não encontrar o ok não para
data[x-3]='\0'; //final da string depois do OK
Serial.println("MENSAGEM RECEBIDA:");
Serial.println(data); //mostra as SMS
limpa_buffer();
delay(2000);

Serial.println("|");
Serial.println("|");
Serial.println("----------------------------  LENDO SMS FIM ----------------------------");
Serial.println("|");
Serial.println("|");
Serial.println("|");
Serial.println("");
};
//#########################################################################
############
void validanumero()
{
  Serial.println("|");
  Serial.println("|");
  Serial.println("-----------------------VALIDANDO NUMERO DO USUARIO -------------------
----");
  Serial.println("|");
  Serial.println("|");
  a=0;
  b=0;
  for(a=8; a!=12; a++)
  {
```

```
   nremetente[b]=data[a];
   b++;
  }
  nremetente[4]='\0';
  Serial.println(nremetente);
  if(!strcmp(ncadastrado, nremetente)){
  validacodigo();
  }
  else{
  Serial1.println("AT+CMGS=\"+556196252800\"\r");  // Número de Envio da SMS
  delay(1000);
  Serial1.println("Alguem Tentou Acessar o Sistema");  // Inicio da escrita da SMS
  delay(1000);
  Serial1.println((char)26);
  leserial();               //processa dados e armazena no buffer
  retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
  Serial.println(comm_buf);
  limpa_buffer();
  delay(15000);
  apagasms();
  }
};
//#########################################################################

void validacodigo()
{
Serial.println("|");
Serial.println("|");
Serial.println("----------------VALIDANDO CODIGO SMS ----------------------");
Serial.println("|");
Serial.println("|");
j=0;
i=0;
for(i=41; i!=45; i++)
```

```
{
codremetente[j]=data[i];
j++;
}
codremetente[4]='\0';

if ((!strcmp(codcadastrado1, codremetente))||(!strcmp(codcadastrado2,
codremetente))||(!strcmp(codcadastrado3, codremetente))||(!strcmp(codcadastrado4,
codremetente))||(!strcmp(codcadastrado5, codremetente))
||(!strcmp(codcadastrado6, codremetente))){

if(!strcmp(codcadastrado1, codremetente)){

Serial.println("CODIGO CADASTRADO");
digitalWrite(ledligado,HIGH);
digitalWrite(ledportas,HIGH);
delay(1000);

if((digitalRead(ledvidros) == 0)){
motor();
delay(1000);

}

delay(1000);
gsm.println("AT+CMGS=\"+556196252800\"\r");  // n para o envio da SMS,
delay(1000);
gsm.println("Alarme Ativado");  // Inicio da escrita da SMS
delay(1000);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
```

```
delay(15000);
apagasms();
}
if(!strcmp(codcadastrado2, codremetente)){
Serial.println("CODIGO CADASTRADO");
digitalWrite(ledligado,LOW);
digitalWrite(ledportas,LOW);
gsm.println("AT+CMGS=\"+556196252800\"\r");  // n para o envio da SMS,
delay(1000);
gsm.println("Alarme Desativado");  // começo de escrita SMS
delay(1000);
msendbutton = false;
delay(200);
msendsensor = false;
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
msendbutton = false;
msendsensor = false;

}
if(!strcmp(codcadastrado3, codremetente)){

Serial.println("CODIGO CADASTRADO");
if(digitalRead(ledligado) == 1){

gsm.println("AT+CMGS=\"+556196252800\"\r");  //N de envio da SMS,
delay(1000);
gsm.println("O Alarme esta ligado");  //Começo de escrita da SMS
```

```
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();

}

else{

if(digitalRead(ledportas) == 0){
digitalWrite(ledportas,HIGH);
gsm.println("AT+CMGS=\"+556196252800\"\r");  //Numero de envio da SMS
delay(1000);
Serial1.println("Portas Travadas ");  // Começo de escrita da SMS
delay(1000);
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{

gsm.println("AT+CMGS=\"+556196252800\"\r");  //Numero de envio da SMS
delay(1000);
gsm.println("Portas ja estao Travadas ");  //Começo da escrita da SMS
delay(1000);
```

```
delay(200);
Serial1.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
}
}
if(!strcmp(codcadastrado4, codremetente)){

Serial.println("CODIGO CADASTRADO");
if(digitalRead(ledligado) == 1){

gsm.println("AT+CMGS=\"+556196252800\"\r");  // Numero de envio da SMS
delay(1000);
gsm.println("O Alarme esta ligado");  //Escrita da SMS
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();

}

else{

if(digitalRead(ledportas) == 1){
digitalWrite(ledportas,LOW);
```

```
gsm.println("AT+CMGS=\"+556196252800\"\r");  //Numero de envio da SMS,
delay(1000);
gsm.println("Portas Destravadas ");  // Escrita da SMS
delay(1000);
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{
gsm.println("AT+CMGS=\"+556196252800\"\r");
delay(1000);
gsm.println("Portas ja estao destravadas ");
delay(1000);
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
}
}

if(!strcmp(codcadastrado5, codremetente)){

Serial.println("CODIGO CADASTRADO");
if(digitalRead(ledligado) == 1){
```

```
gsm.println("AT+CMGS=\"+556196252800\"\r");
delay(1000);
gsm.println("O Alarme esta ligado");
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();

}

else{

if(digitalRead(ledvidros) == 1){
gsm.println("AT+CMGS=\"+556196252800\"\r");
delay(1000);
gsm.println("Os vidros ja estao fechado");
delay(1000);
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{
motor();
delay(1000);
```

```
gsm.println("AT+CMGS=\"+556196252800\"\r");
delay(1000);
gsm.println("Vidros Fechados");
delay(1000);
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
}
}
if(!strcmp(codcadastrado6, codremetente)){

Serial.println("CODIGO CADASTRADO");
if(digitalRead(ledligado) == 1){
gsm.println("AT+CMGS=\"+556196252800\"\r");
delay(1000);
gsm.println("O Alarme esta ligado");
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{

if(digitalRead(ledvidros) == 0){
```

```
gsm.println("AT+CMGS=\"+556196252800\"\r");
delay(1000);
gsm.println("Os vidros ja estao abertos");
delay(1000);
delay(200);
Serial1.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
else{
motor_volta();
delay(1000);
gsm.println("AT+CMGS=\"+556196252800\"\r");
delay(1000);
gsm.println("Vidros Abertos");
delay(1000);
delay(200);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
}
}
}

else{
```

```
Serial.print("Mensagem Invalida");
gsm.println("AT+CMGS=\"+556196252800\"\r");
delay(1000);
gsm.println("Codigo Não Cadastrado");
delay(1000);
gsm.println((char)26);
leserial();              //processa dados e armazena no buffer
retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
apagasms();
}
Serial.println("|");
Serial.println("|");
Serial.println("----------------------  VALIDANDO CODIGO SMS FIM ----------------------");
Serial.println("|");
Serial.println("|");
Serial.println("|");
Serial.println("");
delay(2000);

};
//#################################################
void push_button(){

valbutton = digitalRead(BUTTON); //lê e armazena o valor da entrada
if((valbutton == HIGH)&&(old_valbutton == LOW)){  //verifica se a saída está em
HIGH(botào pressionado)
statebutton = 1 - statebutton;
delay(10);
}
old_valbutton = valbutton;
```

```
if(statebutton == 1){
digitalWrite(ledbutton,HIGH); //Apaga o LED
}
else{

digitalWrite(ledbutton,LOW); //caso contrário o LED permanece apagado
delay(1000);
}
if(digitalRead(ledligado) == 1){

if((digitalRead(ledbutton) ==  1) && msendbutton == false ){
msendbutton = true;
delay(1000);
gsm.println("AT+CMGS=\"+556196252800\"\r");
delay(1000);
gsm.println("Intrusao Pela porta");
delay(1000);
gsm.println((char)26);
leserial();                          //processa dados e armazena no buffer
retorno = IsStringReceived("OK");          // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
limpa_buffer();
apagasms();
}
}
};
/*************************************************************
Rotina utilizada para rotação do Motor
Função myStepper.step
1 - Positiva - Sentido anti-horario
2 - Negatica - Sentido horario
*************************************************************/
```

```
void motor(){

Stepper myStepper(passosVoltaCompleta,34,38,40,36);
myStepper.setSpeed(1);
myStepper.setSpeed(10);
myStepper.step(3000);
digitalWrite(ledvidros,HIGH);
};
void motor_volta(){
Stepper myStepper(passosVoltaCompleta,34,38,40,36);
myStepper.setSpeed(1);
myStepper.setSpeed(10);
myStepper.step(-3000);
digitalWrite(ledvidros,LOW);
};
//#####################################################################
void SENSOR_PRESENCA(){
valsensor = digitalRead(SENSOR); //lê e armazena o valor da entrada
if((valsensor == HIGH)&&(old_valsensor == LOW)){  //verifica se a saída está em
HIGH(botào pressionado)
statesensor= 1 - statesensor;
delay(10);
}
old_valsensor = valsensor;
if(statesensor == 1){
digitalWrite(ledsensor,HIGH); //Apaga o LED
}
else{

digitalWrite(ledsensor,LOW); //caso contrário o LED permanece apagado
delay(1000);
}
if(digitalRead(ledligado) == 1){
if((digitalRead(ledsensor) ==  1) && msendsensor == false ){
```

```
msendsensor = true;
delay(1000);
gsm.println("AT+CMGS=\"+556196252800\"\r");
delay(1000);
gsm.println("Intrusao Pela Janela");
delay(1000);
gsm.println((char)26);
leserial();                          //processa dados e armazena no buffer
retorno = IsStringReceived("OK");            // verifica se o retorno é o esperado
Serial.println(comm_buf);
limpa_buffer();
delay(15000);
limpa_buffer();
apagasms();
}
}
};
//#####################################################################
void poweron() {
pinState = digitalRead(statuspin);
if(pinState==LOW)
{
Serial.println("estou ligando...");
digitalWrite(powerkey,HIGH);
delay(2000);
digitalWrite(powerkey,LOW);
delay(3000);
}
};
//#############################################################################
void apagasms(){
Serial.println("|");
Serial.println("|");
Serial.println("-------------------------  DELETANDO SMS  -------------------------");
```

```
Serial.println("|");
Serial.println("|");
gsm.println("AT+CMGDA=\"DEL ALL\"\r");//executa comando
delay(500);
//espera resultado na serial
leserial();              //processa dados e armazena no buffer
//retorno = IsStringReceived("OK"); // verifica se o retorno é o esperado
//Serial.println(comm_buf);  //mostra retorno real
limpa_buffer();          //limpa buffer
Serial.println("|");
Serial.println("|");
Serial.println("-----------------------  DELETANDO SMS FIM -------------------------");
Serial.println("|");
Serial.println("|");
Serial.println("|");
Serial.println("");
};
/************************************************************
* Metodo splitstring
************************************************************/
void splitString(char* data) {
Serial.print("Data entered splitString: ");
Serial.println(data);
Serial.print("FIM Data entered splitString: ");
char* parameter;
parameter = strtok (data,",");
while (parameter != NULL) {
//PROXIMO_METODO_COMA_AÇÃO(parameter);
Serial.println("parameter separado as partes");
Serial.println(parameter);
parameter = strtok (NULL,",");
}
};
/************************************************************
```

```
* Metodo para leitura da softserial
* buffer maximo de (num_of_bytes) caracteres
**********************************************************/
void leserial()
{
num_of_bytes = gsm.available(); //Pega quantidade de bytes ques estao na serial
while (gsm.available() > 0)    //Espera de até Available ser true
{
for(n=0; n<num_of_bytes; n++)
{
comm_buf[n] = gsm.read(); //armazena buffer e a cada mySerial.read carry on buffer interno
}
}
gsm.flush();              //limpa buffer de entrada da serial
};
/**********************************************************
* Metodo para limpeza do buffer
**********************************************************/
void limpa_buffer()
{
memset(comm_buf,'\0',BUF_LEN);
};
/**********************************************************
* Method checks received bytes
* compare_string - pointer to the string which should be find
* return: 0 - string was NOT received
* 1 - string was received
**********************************************************/
byte IsStringReceived(char const *compare_string)
{
//Serial.println("testando...");
char *ch;
byte ret_val = 0;
```

```
if(1) {
ch = strstr((char *)comm_buf, compare_string);
if (ch != NULL)
{
Serial.println("existe Valor");
ret_val = 1;
}
else
{
}
}
return (ret_val);
};

void apagasms1(){
Serial1.println("AT+CMGDA=\"DEL ALL\"\r");//executa comando
delay(500);
leserial();              //processa dados e armazena no buffer
limpa_buffer();          //limpa buffer

}
```

ANEXO I