View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositório Institucional do UniCEUB

CENTRO UNIVERSITÁRIO DE BRASÍLIA – UniCEUB
FACULDADE DE TECNOLOGIA E CIÊNCIAS SOCIAIS APLICADAS – FATECS
CURSO DE ENGENHARIA DA COMPUTAÇÃO
PROJETO FINAL DE GRADUAÇÃO

Leandro Borges Martins

Sistema Antifurto Integrado ao Monitoramento de Presença de Crianças no interior de veículos utilizando GPRS

Brasília, 2010

Leandro Borges Martins

Sistema Antifurto Integrado ao Monitoramento de Presença de Crianças no interior de veículos utilizando GPRS

Prof. Orientador: M.Sc. Maria Marony Sousa Farias

Monografia apresentada à Banca Examinadora da Faculdade de Tecnologia e Ciências Sociais Aplicadas do UniCEUB como um dos pré-requisitos para obtenção do título de Bacharel em Engenharia da Computação.

Brasília, 2010

Autoria: Leandro Borges Martins

Título: Sistema Antifurto Integrado ao Monitoramento de Presença de Crianças no interior de veículos utilizando GPRS

Monografia apresentada à Banca Examinadora da Faculdade de Tecnologia e Ciências Sociais Aplicadas do UniCEUB como um dos pré-requisitos para obtenção do título de Bacharel em Engenharia da Computação.

Brasília, 22 de Dezembro de 2010

Banca Examinadora

_____

Prof. Msc. Maria Marony
Orientador

_____

Prof. Msc. Flavio Klein
Examinador

_____

Prof. Msc. Fabiano Mariath
Examinador

_____

Prof. Dr. Gil Renato
Examinador

AGRADECIMENTOS

A Deus, por minha vida e saúde.

Ao meu pai João, pelo exemplo e apoio incondicional. Meu ídolo.

A minha mãe Herli, pelo carinho, serenidade e paciência. Minha fortaleza.

Aos amigos, sempre pacientes e constantes.

A professora Maria Marony, pelo apoio e orientação.

Aos professores e profissionais do Uniceub que contribuíram para o processo de formação profissional e intelectual.

RESUMO

Neste trabalho é apresentada uma proposta de um sistema de segurança veicular. A inovação apresentada neste projeto se deve ao fato de que o sistema proposto, não apenas detecta o furto do veículo através de uma mensagem SMS ou uma ligação ao proprietário de um automóvel, mas também, permite que seja detectado o possível esquecimento de uma criança ou animal no interior do veículo, evitando tragédias.

A solução desenvolvida foi implementada utilizando um micro controlador, placa GPRS e chip GSM. Através do GPRS, foi possível o envio de informações via SMS ou mesmo efetuar ligações. A ferramenta utilizada para desenvolver a aplicação foi o Arduino Development Enviroment que é baseada na linguagem C.

Palavras chaves: Arduíno, GPRS, Monitoramento, Alarme.

ABSTRACT

This study presents a proposal for a vehicle safety system. The innovation presented in this project is the fact that the system not only detects the theft of the vehicle by an SMS message or a call to the automobile's owner, but also allows to be detected a missing child or animal inside the vehicle, avoiding accidents or tragedies.

The solution was implemented using a microcontroller, GPRS and GSM chip card. Through the GPRS it is possible to make calls or send informations by SMS. The tool used to develop the aplication was Arduino Devlopment Enviroment wich is based on language C.

Palavras chaves: Arduíno, GPRS, Monitoring, Alarm.

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

Neste capítulo é feita a apresentação deste projeto descrevendo tópicos como motivação, objetivos e estrutura da monografia.

1.1 Motivação

A tensão da vida moderna tem levado pais, mães ou responsáveis a esquecer bebês e crianças pequenas no interior dos veículos. Em muitas cidades brasileiras, registram-se muitos casos em que crianças se queimam gravemente por ficarem horas expostos ao calor chegando em algumas vezes até o falecimento, algo que nenhuma consciência absolve.

Desta forma, surgiu a idéia de criar um dispositivo que monitore o interior de veículos, evitando assim o furto do automóvel e o esquecimento de crianças pequenas e/ou animais no interior dos mesmos, utilizando um microcontrolador, placa GPRS e chip GSM. A fim de evitar essas perdas irreparáveis, neste projeto é proposta a criação de um sistema interligado às portas, bancos e ignição do veículo, capazes de identificar um possível esquecimento de bebês e/ou animais, bem como o seu furto.

A inovação apresentada neste projeto se deve ao fato de que o sistema proposto, não apenas detecta o furto do veículo através de uma mensagem SMS ou uma ligação ao proprietário de um automóvel, mas

também, permite que seja detectado o possível esquecimento de uma criança ou animal no interior do veículo, evitando tragédias.

1.2 Objetivos

O objetivo geral do projeto é o desenvolvimento de uma aplicação capaz de realizar o monitoramento de presença de bebês e/ou animais no interior dos veículos, bem como o furto do automóvel. Tendo como resultado esperado, tem-se a notificação do proprietário do veículo sobre algum desvio do comportamento normal com o recebimento de mensagens SMS ou ligações. Sendo assim, os objetivos gerais do projeto são:

a. Identificar o esquecimento de bebês ou animais no interior dos veículos;

b. Acionar o alarme para que o motorista responsável tome as devidas ações.

c. Identificar um possível furto do veículo informando o proprietário.

A partir destes objetivos gerais, detalhou-se os seguintes objetivos específicos:

- Utilização de micro controlador que centralize os sinais recebidos pelos sensores;

- Utilização de placa GPRS para efetuar ligações e enviar mensagens SMS;

- Desenvolver uma placa para tratamento de tensão capaz de alimentar o sistema proposto;

- Desenvolver um sistema capaz de receber sinais de sensores, analisar os sinais e posteriormente gerar ações adequadas.

1.3 Estrutura da Monografia

Este projeto é constituído de cinco capítulos. Segue uma breve descrição.

Capítulo 1 – **INTRODUÇÃO –** aborda o que será tratado no projeto bem como a motivação e objetivo.

Capítulo 2 - **APRESENTAÇÃO DO PROBLEMA –** aborda o problema identificado e a solução proposta.

Capítulo 3 - REFERENCIAL TEÓRICO - aborda conceito e definições como, história e informações como um todo do arduíno. Assim como os dispositivos utilizados, como sensores de presença, de pressão e alarmes.

Capítulo 4 - PROTÓTIPO DESENVOLVIDO – descrição detalhada do desenvolvimento do projeto propriamente dito. Citando as ferramentas utilizadas, detalhes de implementação e resultados obtidos.

Capítulo 5 – RESULTADOS OBTIDOS E ANÁLISE DOS RESULTADOS - Descreve os testes feitos e ações tomadas.

Capítulo 6 – CONSIDERAÇÕES FINAIS – Descreve as conclusões obtidas nesse projeto.

Capítulo 7 – REFERÊNCIAS BIBLIOGRÁFICAS – Descreve o material consultado nesse projeto.

Capítulo 8 – ANEXOS – Descreve tabelas usadas como auxílio no desenvolvimento do projeto.

APÊNDICE

2 APRESENTAÇÃO DO PROBLEMA

Neste capítulo é descrita a apresentação do problema bem como a solução proposta.

2.1 Descrição do Problema

Um grande fator de gastos públicos ou privados é a enorme quantidade anual de acidentes automotivos, causados na maioria das vezes por falta de atenção. Não raro, os pequenos acidentes se transformam em grandes tragédias. Um exemplo desse fato é a ocorrência de esquecimentos, seja de crianças ou até mesmo animais dentro de veículos, onde não existe um monitoramento de presença capaz de identificar ou tomar alguma decisão, caso uma criança ou animal, seja esquecido dentro do carro.

No Brasil, um caso foi relatado no jornal o GLOBO de 13/04/2007, que um pai esqueceu seu bebê de um ano e quatro meses no carro ocasionando a morte da criança. Nesses casos, a lei autoriza o juiz a conceder perdão judicial, já que a aplicação de uma pena é considerada desnecessária pela Justiça, porque o pai já foi punido com a morte do filho.

Já nos Estados Unidos, somente em 2003, quarenta e duas crianças morreram de insolação por terem sido deixadas no interior do veículo e de

1996 a 2000, mais de 120 crianças faleceram. Além disso, mais de mil casos de lesão ou morte já foram documentados. Essas fatalidades ocorrem porque a temperatura dentro de um carro, independente da sua cor ou do assento, pode subir rapidamente de 35ºC para 65ºC em cerca de 20 minutos se em exposição ao sol e as crianças são incapazes de produzir suor necessário para manter a temperatura corporal, ocasionando graves danos a alguns órgãos ou mesmo a morte(PARENTHOOD,2010).

Tendo em vista esse assunto, propõe-se neste projeto, a utilização de sensores de movimento e pressão devidamente instalados no interior do veículo de forma a alertar seu dono sobre o possível esquecimento de crianças pequenas e/ou animais.

A outra frente do sistema, que também é considerada um problema é em relação ao furto de veículos. Neste caso, quando o alarme for acionado o proprietário receberá uma mensagem SMS ou uma ligação.

3 REFERENCIAL TEÓRICO

Neste capítulo é feita a apresentação deste projeto descrevendo tópicos como motivação, objetivos e estrutura da monografia.

3.1 Placa Arduino

A placa de laboratório Arduino Duemillanove é baseada no micro controlador ATmega328P e foi inicialmente utilizada nesse projeto devido à sua facilidade de uso, tanto para a programação do controlador quanto para o uso de suas portas de entrada e de saída. Possui 14 portas digitais de entrada e saída, dentre as quais 6 podem ser utilizadas como saídas PWM (Pulse Width Modulation), 6 entradas analógicas, cristal interno de 16 MHz, conexão USB para dados e alimentação, botão de reset, entre outros recursos. A simplicidade de utilização foi determinante para a escolha desta placa, visto que por haver um *bootloader*[1] pré gravado no controlador, o processo de gravação e edição é extremamente simples, não havendo necessidade de outros equipamentos para tanto. Para o uso, pode ser alimentada pela porta USB ou com um fonte DC de 7 a 12V [ARDUINO, 2010]. A figura 3.1 ilustra a placa arduino duemilanove.

[1] Bootloader é num gerenciador de inicialização.

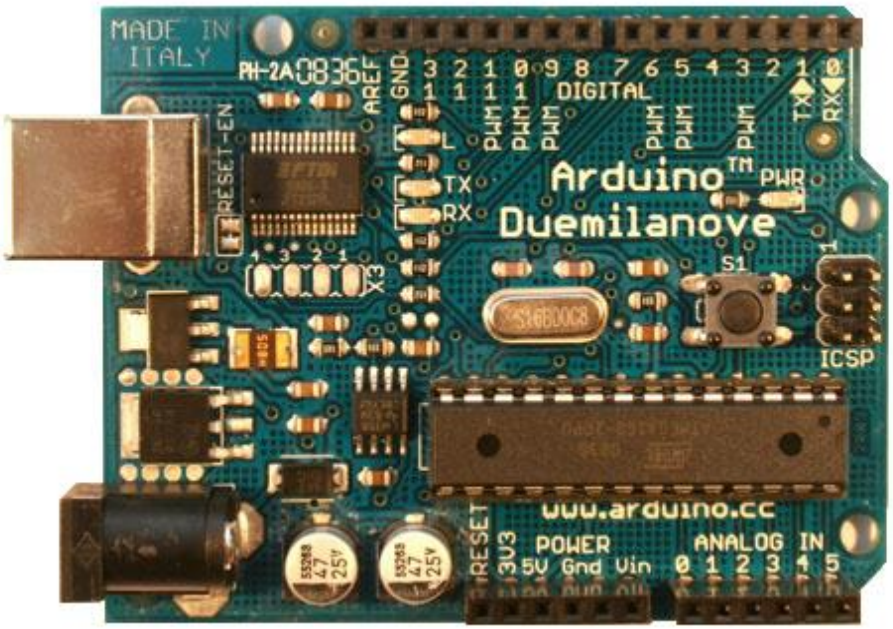


Figura 3.1 - Placa Arduino Duemillanove

Fonte: http://arduino.cc/en/uploads/Main/ArduinoDuemilanove.jpg

Na tabela 3.2 é descrito um resumo das características operacionais da placa.

Tabela 3.2 – Características da placa Arduino Duemillanove

Microcontrolador	ATmega328P
Voltagem Operacional	5V
Voltagem de Entrada	7 ~ 12V
Pinos Digitais (I/O)	14 (6 PWM)
Pinos Analógicos	6
Corrente por Pinos	40mA
Corrente Saída 3.3V	50mA
Memória Flash	32KB(2KB para Bootloader)
SRAM	2KB
EEPROM	1KB
Clock	16MHz

Fonte: http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove

A fonte de alimentação é um fator de extrema importância para o bom funcionamento da placa, pois uma alimentação inferior à 6V pode fazer

com que as portas de saída não consigam gerar as tensões e correntes esperadas, e uma tensão de entrada superior à 12V pode ocasionar em danos aos circuitos da placa. A seguir, a descrição dos pinos de alimentação.

VIN. É a entrada de tensão quando a placa Arduino é alimentada por uma fonte externa. Caso a alimentação venha diretamente do jack de conexão para alimentação, esse pino pode servir para fornecer alimentação à outros componentes.

5V. Fonte regulada utilizada para alimentar o microcontrolador e outros componentes da placa. Pode vir de uma entrada Vin, pela USB ou por outra entrada regulada a 5V.

3V3. Saída de alimentação de 3.3V, gerada pelo chip on board FTDI. A corrente maxima é de 50mA.

GND. Terra.

Outra característica favorável ao uso da tecnologia Arduino é sua facilidade de programação. Em um ambiente de programação próprio para as placas Arduino usa-se a programação em C/C++, com recursos exclusivos desenvolvidos com base na linguagem “Wiring”. Por já vir montada com um bootloader a placa Arduino Duemillanove permite a gravação direta no chip, sem necessidade de um equipamento de gravação. A comunicação com o computador de programação é feita através do protocolo STK500, e permite saídas em tela quando explicitadas na linha de código compilada e gravada no controlador [ARDUINO, 2010].

Sempre que é feita uma gravação do software no controlador da placa, automaticamente é feita re-inicialização do circuito, onde as memórias

internas são sobrescritas com as novas linhas. Embora a gravação force um reinício do sistema, essa ação não fica presa a esse evento, pois a placa dispõe de um botão de reset, o qual ao ser pressionado gera um reinício do circuito, limpando as variáveis internas da programação e levando o controlador ao seu estado inicial logo após a gravação de um novo código.

Entretanto, com o andamento do projeto, verificou-se a necessidade de utilização uma placa maior (com mais portas analógicas e digitais) a fim de utilizar também uma placa GPRS, sendo assim foi adquirida também a placa arduino mega que é similar a duemilanove. A figura 3.3 a seguir ilustra a placa arduino mega:

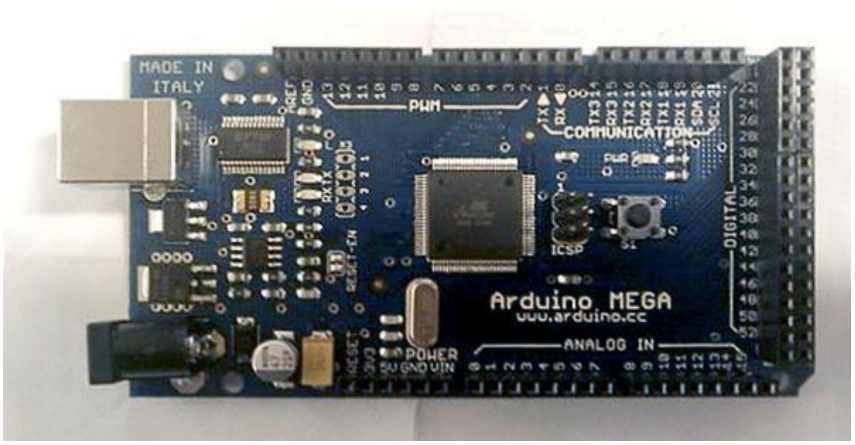


Figura 3.3 - Placa Arduino Mega

Fonte: http://blog.makezine.com/archive/2009/03/arduino_mega_spotted.html?CMP=OTC-0D6B48984890

No Quadro 3.4 estão detalhadas as principais diferenças entre as placas Arduino Mega e Arduino Duemilanove.

Tabela 3.4 – Principais diferenças entre as placas Duemilanove e Mega.

Arduino Duemilanove		Arduino Mega	
Microcontrolador	Atmega168/328	Microcontrolador	Atmega 1280
Tensão de Operação	5V	Tensão de Operação	5V
Tensão de Entrada	6-20V	Tensão de Entrada	6-20V
Pinos de I/O Digital	14	Pinos de I/O Digital	54
Pinos Analógicos	6	Pinos Analógicos	16
Pinos PWM	6	Pinos PWM	14
Corrente DC por pino de I/O	40mA	Corrente DC por pino de I/O	40mA
Corrente DC(3.3V)	50mA	Corrente DC(3.3V)	50mA

Tabela 3.4 – Principais diferenças entre as placas

Fonte: O autor.

3.2 Placa GPRS

No decorrer do projeto, foi verificada a possibilidade de se utilizar GPRS como modo de acionamento do sistema. Então, identificou-se o módulo disponível para arduíno da Hilo-Sagem que atende a necessidade.

GPRS – General Packet Radio Service ou Serviço de Rádio de Pacote Geral é uma tecnologia que aumenta as taxas de transferência

de dados nas redes GSM existentes que transporta informações através da comutação de pacotes a uma velocidade de até 144kbits/seg. A figura 3.5 a seguir ilustra a placa GPRS sendo ligada ao “pigtail” usado para ligar a antena.

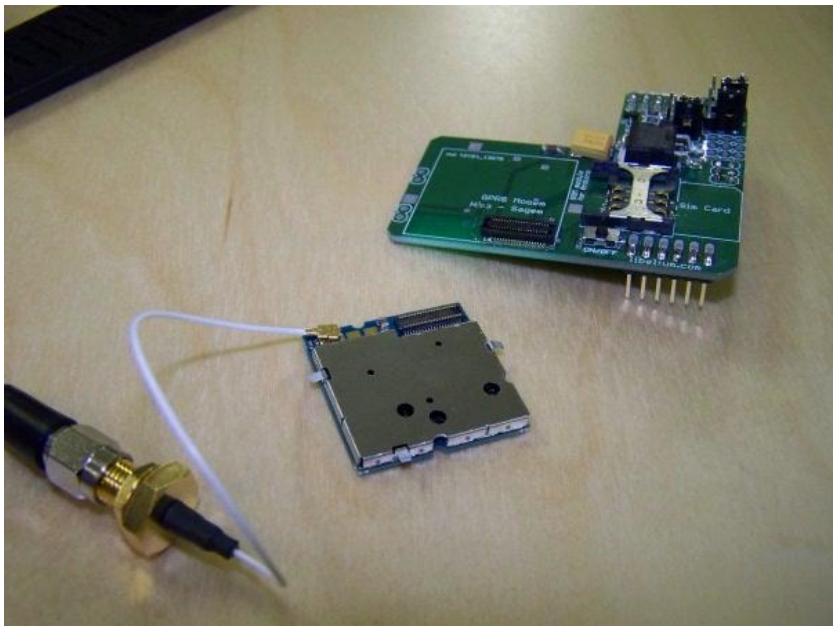


Figura 3.5 - Placa GPRS e Pigtail para ligar a antena.

Fonte: http://www.libelium.com/squidbee/index.php?title=Image:Gprs_hilo_3.jpg

3.3 Protoboard

Bandeja de trabalho na qual as conexões são feitas sem necessidade de solda. Possui conexão em série para as linhas, e através de “jumpers” fecha-se os circuitos entre linhas. A figura 3.6 a seguir, ilustra a protoboard:

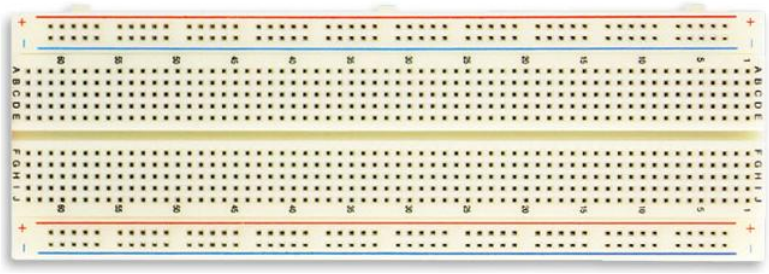


Figura 3.6 - Protoboard Minipa de 830 furos.

Fonte: http://www.reidasvalvulas.com.br/catalog/images/mp-830_g.jpg

Foi utilizado do modelo de 830 furos da Minipa, com dimensões de 165 x 54 x 8.5 mm. Abaixo, suas características técnicas.

- Número de Furos: 830.
- Composição (Base de Terminal): 2 x 315 Furos.
- Composição (Soquete de Distribuição): 2 x 100 Furos.
- Material do Corpo: Polímero ABS.
- Material do Contato: Bronze Fosforoso.
- Acabamento do Contato: Banho de Níquel.
- Bitola Fios de Conexão: 0,4mm ~ 0,7mm.
- Corrente Máxima: 1A RMS.
- Tensão Máxima: 250V RMS.
- Resistência de Contato (1kHz): 1mOaMHS Máximo.
- Rigidez Dielétrica: 1000V RMS por 60s.
- Dimensões: 165(A) x 54(L) x 8,5(P)mm.
- Peso: 85g.

3.4 Componentes Eletrônicos

Na composição do circuito inteiro, foram utilizados componentes eletrônicos comuns, que (à exceção de alguns) são facilmente encontrados em quaisquer lojas de componentes eletrônicos, e a preços justos. A Tabela abaixo exibe os componentes utilizados e as quantidades dos mesmos.

Tabela 3.7 – Componentes eletrônicos utilizados.

Componente	Grandeza	Quantidade
Resistor	2.2k	1
Resistor	1k	1
Resistor	10k	2
Resistor	100k	1
Capacitor	0.1uF	2
Capacitor	1uF	1
Capacitor	100pF	1
CI	CHN4558	1
LED	5V	2
Buzzer	5V	1

Tabela 3.7 – Principais componentes utilizados.

Fonte: O autor.

3.5 Resistor Sensível a Força

No projeto proposto, também foi utilizado um resistor sensível a força onde a resistência varia de acordo com a pressão aplicada. Quanto maior a pressão menor a resistência. Sem nenhuma pressão a resistência é maior que 1MΩ, sendo este sensor, sensível a forças na faixa de 100g a 10kg e nesse projeto é utilizado para detectar se existe ou não pressão nos bancos. A figura 3.8 abaixo ilustra o resistor.

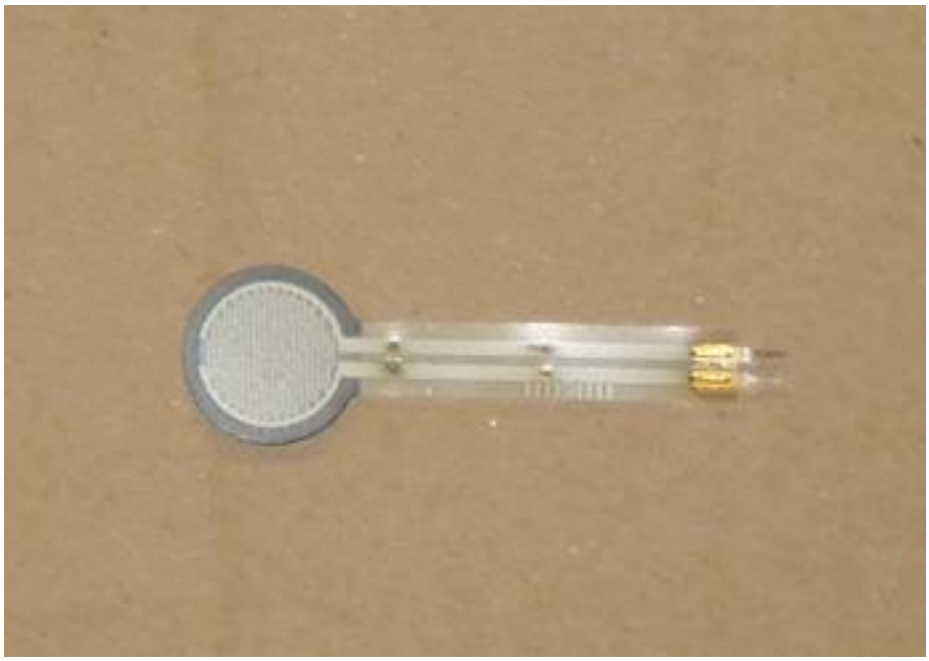


Figura 3.8 - Sensor de Pressão

Fonte: O Autor

3.6 Fontes de Alimentação

Foram usadas duas fontes distintas, uma 12V proveniente da fonte universal 26331, da TMS Microsistemas, como visto na Figura 22. Essa fonte possui entrada chaveada de 110/220Vac a 50/60Hz, com saídas também chaveadas de 1,5V, 3V, 4,5V, 6V, 7,5V, 9V e 12V. Sua carga máxima é de 700mA, e para o desenvolvimento do projeto suas configurações foram chaveadas para entrada de 220V e saída de 12V. A figura 3.9 a seguir ilustra a fonte de alimentação:

Figura 3.9 - Fonte TMS de alimentação 12V.

Fonte: http://www.commerce-center.com.br/imagem.php?imagem=11050001.jpg

A outra entrada de alimentação do sistema é via USB, para a placa MultiPIC, pois além de a mesma já estar preparada para essa porta de

conexão, a voltagem de entrada (5Vdc) é fornecida naturalmente pela porta USB.

Nota: Embora haja conexão USB para alimentação, nenhum dado é transferido entre o computador pessoal e o sistema, nem vice versa.

3.7 Sirene

A concepção inicial do projeto baseou-se na utilização de uma sirene externa, de alta freqüência e potência, e para isso optou-se pelo uso da sirene DNI 4042 Compact, da DNI Condutores Elétricos, que gera 120 dB a 0.25A. Contudo, devido ao alto grau de decibéis desse equipamento e pela demonstração acadêmica do projeto, esse dispositivo foi alterado para um menos potente. A escolha foi uma sirene, também conhecida como Buzzer, de montagem em placas de circuito impresso, da marca Kingstate modelo KPE-272, vide Figura 3.10. Abaixo, as especificações:

Freqüência de Operação: 3,5KHz

Voltagem de Operação: de 3 a 28 Vdc

Consumo: 6mA @ 12Vdc

Pressão Sonora: 88dB @ 12Vdc (a 30 cm)

Dimensões: 32mm (Diam) x 15mm (Alt)

Figura 3.10 - Buzzer PCB Piezoelétrica de 88 dB.

Fonte: http://www.futurlec.com/Pictures/BuzzerPCB1.jpg

3.8 Programação e Simulação de Circuito

A variedade de combinações possíveis que se obtém com o uso de um circuito micro-controlado é imensa. Ao invés de que sejam construídos enormes circuitos eletrônicos para se tratar um sinal de entrada, um CI pode centralizar as estruturas de comparação e decisão, tornando a montagem de circuitos eletrônicos extremamente mais simples e eficiente.

Com isso, foi elaborado um algoritmo de tratamento dos sinais captados pelos transdutores, que embora sejam apenas sinais elétricos, representam grandezas naturais e podem ser tratados com o uso de programação. Ambos os controladores abordados nesse projeto possuem ambientes de programação orientados à linguagem C adaptada, o que facilita o desenvolvimento da programação binária dos mesmos, e abaixo são citados os

programas utilizados para a geração desses códigos e sua simulação no projeto do circuito elétrico do projeto.

3.9 Softwares de Programação

A versão final do projeto foi estabelecida sob a plataforma ATmega, devido à sua facilidade de uso e utilizando programas como Arduino Development Enviroment e HyperTerminal.

3.10 O Software Arduino Development Enviroment

Trata-se de um ambiente desenvolvido em software livre que permite a programação, gravação e debug do código escrito para os controladores embarcados nas placas Arduino. A linguagem de programação utilizada nesse ambiente é o Wiring, uma variação do C/C++, e possui bibliotecas específicas para a programação de controladores. Reconhecem todas as estruturas do C e alguns recursos de C++, e uma vez gerado o código, é possível compilar o mesmo e gerar o código binário, que é facilmente gravado no controlador [BANZI, 2008]. A figura abaixo ilustra um dos programas básicos iniciais em

que se liga o módulo e aguarda o devido comando para que a ligação seja feita.

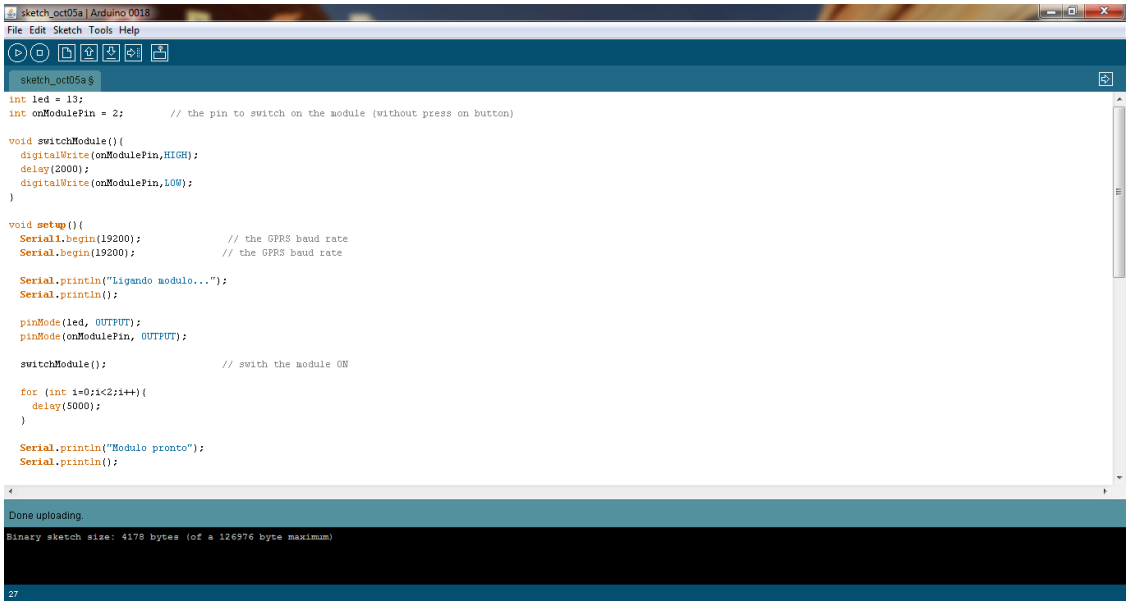


Figura 3.11 - Ambiente de programação Arduino.

Fonte: O Autor

3.11 O Software HyperTerminal

Trata-se de um programa que permite a realização de conexão com outros computadores, sites de Telnet, serviços online e computadores host, usando um modem. Em versões mais antigas do Windows o software proposto fazia parte do pacote, mas nas versões mais atuais como Windows Vista e Windows 7 o mesmo foi retirado da plataforma, entretanto, o mesmo possui uma versão disponível para download grátis e de uso pessoal.

Neste projeto fez-se necessário a utilização do HyperTerminal já que o “Serial Monitor” do ambiente arduino não atende a placa GPRS. A figura abaixo ilustra a utilização do HyperTerminal para testes iniciais da placa onde o código da figura anterior é utilizado e então o módulo inicializado. O HyperTerminal fica então aguardando novas instruções, que são indicadas pelos comandos AT, estes responsáveis por realizar ligações, enviar mensagens e até verificar a banda e nível de sinal disponível pela operadora. Na figura abaixo são usados dois comandos muito utilizados, o ATD que é responsável por discar o número desejado, e o ATH que cancela a ligação. A figura 3.12 a seguir, ilustra o HyperTerminal:

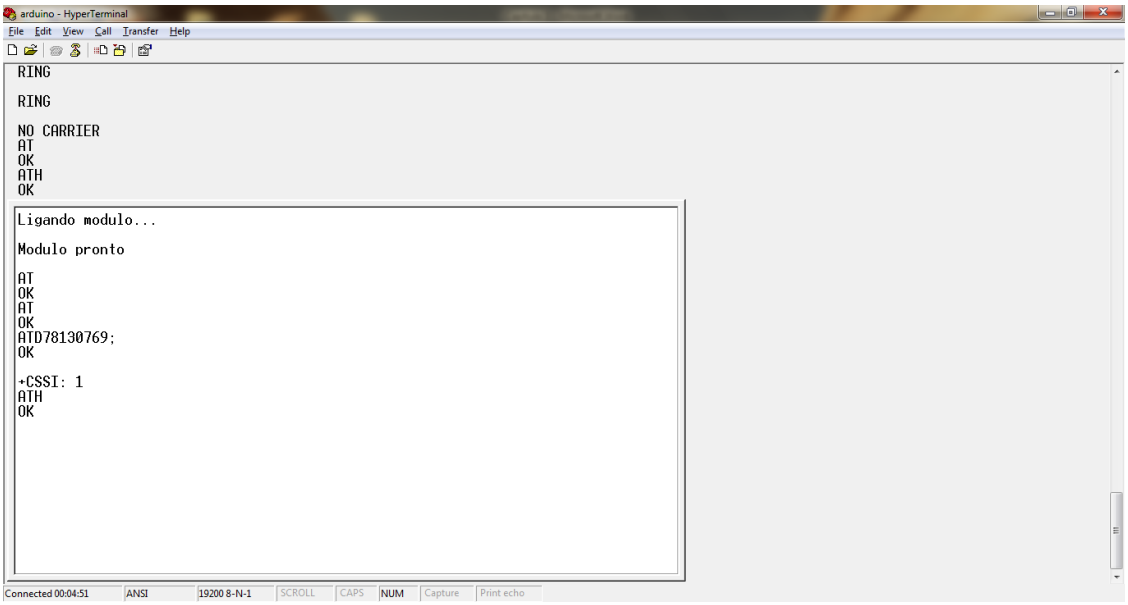


Figura 3.12 - Ambiente de programação HyperTerminal

Fonte: O Autor

4 Desenvolvimento Experimental do Protótipo

Para melhor entendimento e detalhamento da etapa de desenvolvimento, esse capítulo foi segmentado em duas seções principais, para que sejam descritas as seqüências históricas de desenvolvimento, os pontos de dificuldade na evolução do protótipo, as soluções apresentadas e a versão final do equipamento.

4.1 Histórico do Desenvolvimento

Partindo-se do pressuposto de que para atender a necessidade do projeto haveria necessidade de utilização de um circuito micro controlado, optou-se pelo uso do Arduíno. Inicialmente optou-se pela utilização do Arduino Duemilanove que possui micro controlador ATmega 328, mas devido a necessidade de utilização de mais portas de entrada/saída acabou-se optando por utilizar a placa Arduino Mega com micro controlador 1280. Neste passo foram feitos pequenos testes com a utilização de botões e leds.

4.2 A placa GPRS

No decorrer do projeto verificou-se a possibilidade de utilização da placa GPRS como modo de acionamento do alarme que foi adquirida pela Libellium, dando assim uma usabilidade e aplicabilidade maior ao sistema que quando disparado, é feito o acionamento da sirene pelo sistema original e paralelamente é realizada a ligação ou envio de mensagens dependendo da circunstância.

Entretanto, identificou-se um problema:

- A placa duemilanove possui apenas uma entrada serial, sendo que o Arduino se comunica com o computador e a placa GPRS via serial o que ocasionava em erro já que o arduino ao emitir uma mensagem por exemplo, não ficava claro a quem transmitir, computador ou placa GPRS. A figura 4.1 a seguir ilustra o problema:

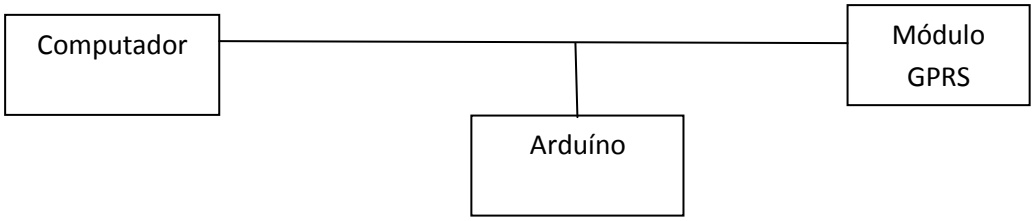


Figura 4.1 – Fluxo do problema

Fonte: O autor.

Para atender a esta necessidade foi adquirida a placa arduino mega que possui quatro entradas seriais, podendo assim ser especificado qual entrada serial deve ser usada para comunicação com o computador e placa GPRS como ilustra a figura 4.2 a seguir:

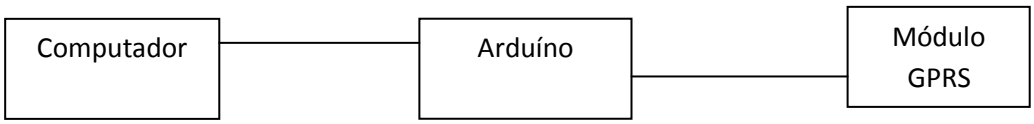


Figura 4.2 – Fluxo do Problema Resolvido

Fonte: O autor.

Então, após esta adaptação, iniciaram-se os testes. A figura 4.1 a seguir, ilustra a adaptação feita, onde foi estabelecido que a porta serial RX1(pino 19) e TX1(pino 18) fará a comunicação entre GPRS e Arduíno. A via RX1 é responsável por receber as informações da GPRS e a TX1 por transmitir instruções. A figura 4.3 ilustra a adaptação feita:

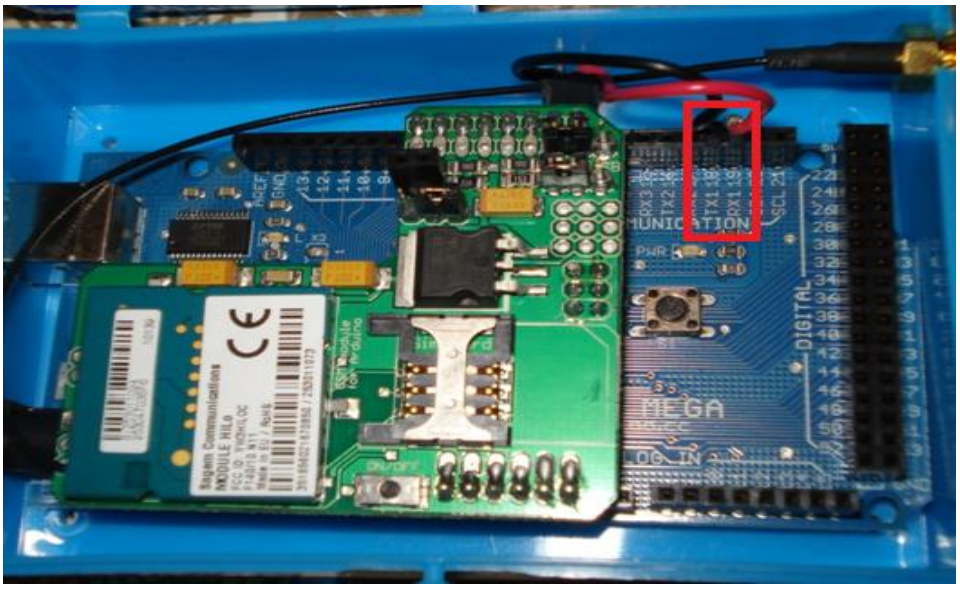


Figura 4.3 - Comunicação Arduino X GPRS

Fonte: O Autor

Sendo assim, a porta serial RX/TX que de acordo com a disposição original da GPRS seria responsável pela comunicação com a Arduíno foi atribuída para trocar informações com o computador através do software Arduino Develpment Enviroment. Essa modificação foi feita para que mesmo com o sistema em funcionamento, a Arduino enviasse informações ao computador, a fim de verificar possíveis erros e o andamento do processo.

Para um sistema em produção, esta alteração não seria necessária e a própria Arduino duemilanove com uma única porta RX TX poderia satisfazer a condição já que ai não seria necessária a comunicação com um computador e somente entre Arduino e GPRS. A figura 4.4 ilustra as vias RX(pino 0) e TX(pino 1), estas que são responsáveis por receber e transmitir informações com o computador.

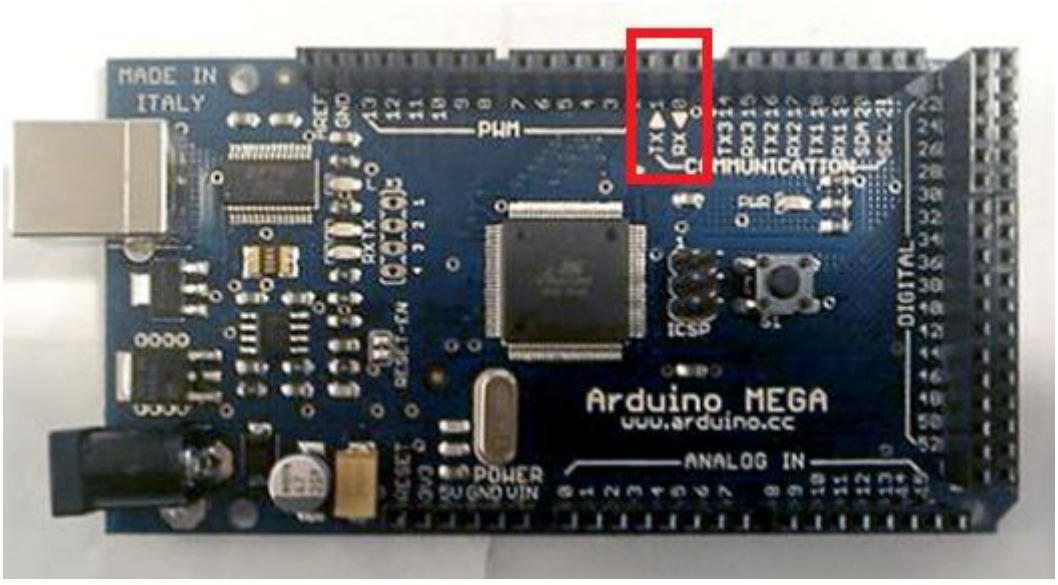


Figura 4.4 - Comunicação Arduino X Computador

Fonte: O Autor

A partir da adaptação feita foram feitos testes unificando placa GPRS e arduíno com auxílio do software HyperTerminal para efetuar ligações e enviar mensagens de texto através do programa a seguir capaz de enviar instruções a GPRS:

```
int led = 13;
int onModulePin = 2;       // pino configurado para inicializar modulo
/**
 *  função para inicializar o módulo onde configura-se o pino 2 como HIGH e após 2 segundos
 *  configura-se o pino como LOW.
 */
void switchModule(){  digitalWrite(onModulePin,HIGH);
 delay(2000);
 digitalWrite(onModulePin,LOW);
}
void setup(){
 Serial1.begin(19200);            // velocidade de comunicação gprs
 Serial.begin(19200);            // velocidade de comunicação arduino
 Serial.println("Ligando modulo...");
 Serial.println();
 pinMode(led, OUTPUT);
 pinMode(onModulePin, OUTPUT);
 switchModule();                // swith the module ON
 for (int i=0;i<2;i++){
  delay(5000);
 }
 Serial.println("Modulo pronto");
 Serial.println();
}
/**
 * o que for escrito no HyperTerminal será enviado como comando ao GPRS
 * e o que o GPRS obtiver como resposta será enviado ao HyperTerminal.
 */
void loop() {
 if (Serial.available()) {
  Serial1.print(Serial.read(), BYTE);
 }
 if (Serial1.available()) {
  Serial.print(Serial1.read(), BYTE);
 }
}
```

A figura 4.5 ilustra o teste realizado para efetuar uma ligação onde:

1 – Software HyperTerminal, responsável por enviar instruções ao GPRS. Na figura, o comando utilizado para ligação é “ATD78130769” onde ATD é o comando para realizar a ligação e posteriormente o número a ser discado.

2 – Celular recebendo a ligação após comando do HyperTerminal.

3 – Antena utilizada pelo GPRS.

4 – Arduino e GPRS acopladas devidamente após adaptações feitas.

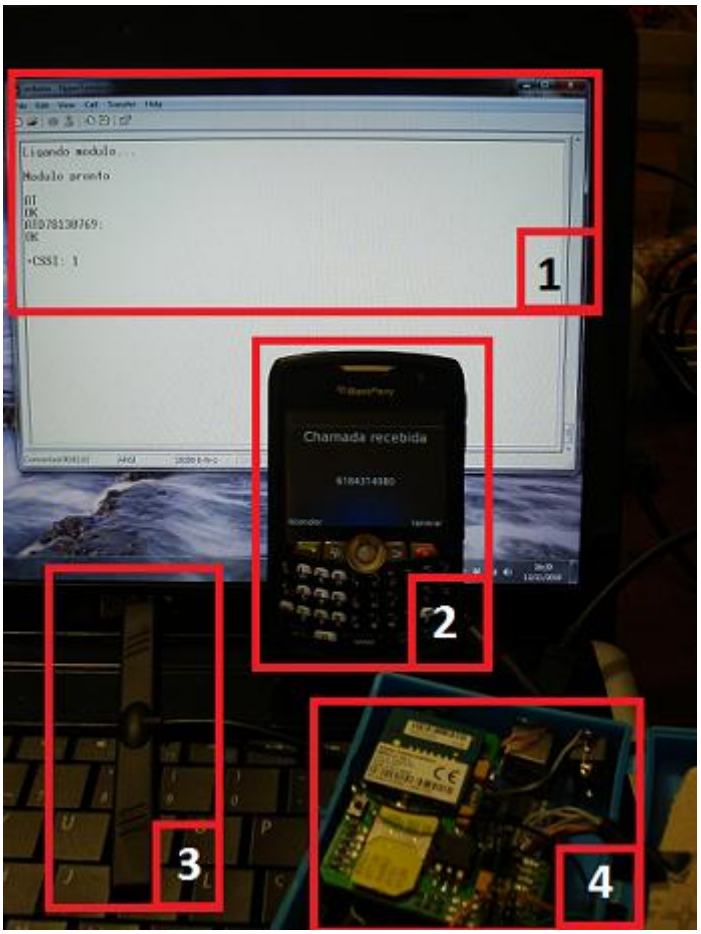


Figura 4.5 - Teste Arduino X GPRS

Fonte: O Autor

Então, a partir do teste de ligação feito com êxito, iniciaram-se os testes do envio de mensagem. A figura 4.6 ilustra o teste de envio de mensagens onde:

1 – Software HyperTerminal. Na figura, o comando utilizado para enviar a mensagem é “ATD+CMGS” seguido do número de destino.

2 – Celular recebendo SMS após comando do HyperTerminal.

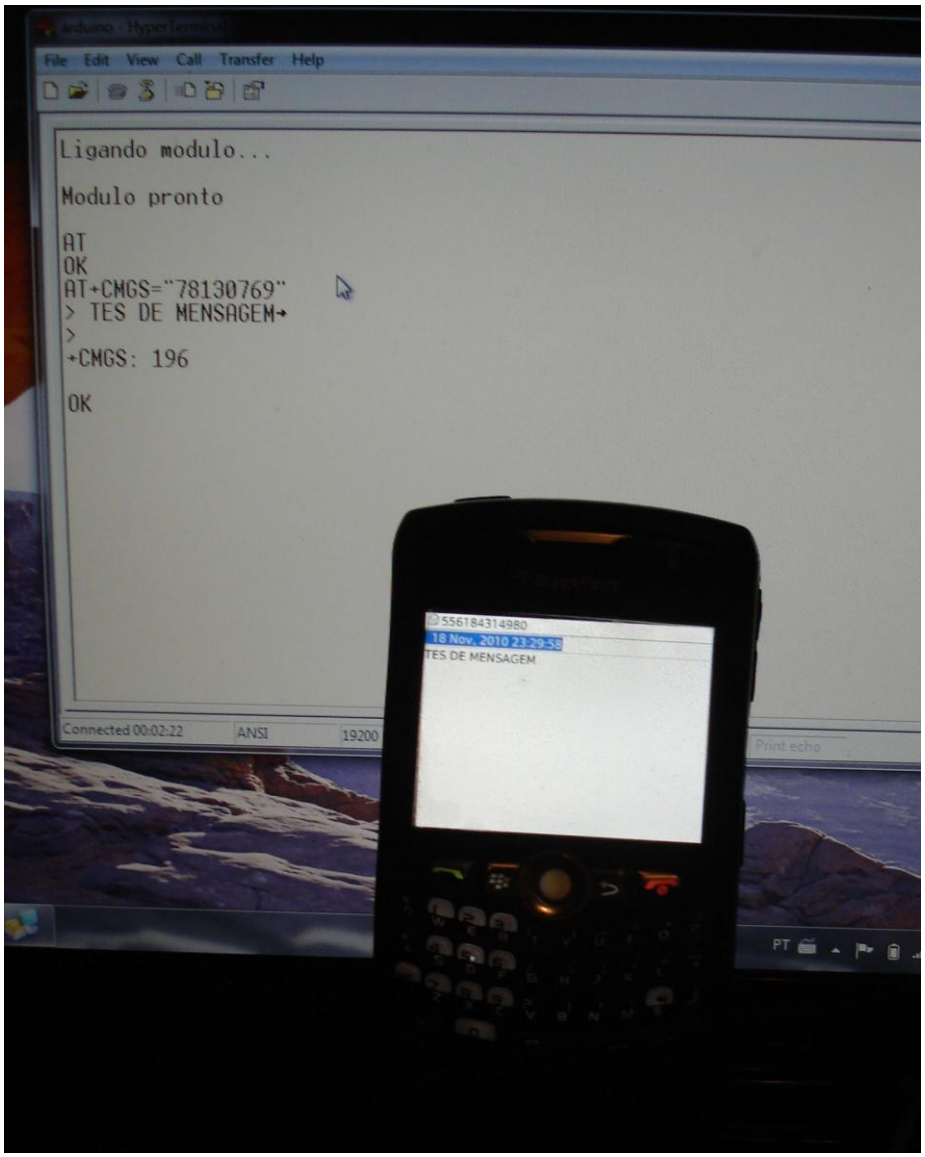


Figura 4.6 - Teste Arduino X GPRS SMS

Fonte: O Autor

Após alguns testes feitos sem êxito, verificou-se que a mensagem necessitava de um comando de término e não apenas um comando de ENTER para finalizá-la. O comando que identifica a finalização é o “Ctrl+Z”.

4.3 Alimentação do Protótipo

O passo seguinte ocorreu com a intenção de desenvolver o protótipo da forma mais próxima da realidade, para tal, puxou-se um fio da bateria para alimentar a placa. Então, surgiu outro problema, a bateria fornece 12V e a Arduino trabalha com 5V sendo necessário a confecção de uma simples placa caseira para que fosse feita essa transformação.

Então, foi feito uma placa onde a parte de alimentação é constituída por um regulador de tensão e por uma etapa de potência. O regulador de tensão é o 7805 que é responsável por reduzir e estabilizar as tensões fornecidas pela fonte aos níveis corretos para o micro-controlador. Já a etapa de potência é constituída por um amplificador operacional TL084, um resistor e um transistor de alta potência TIP120. Na figura 4.7 é apresentado o diagrama esquemático utilizado na construção da etapa de alimentação.

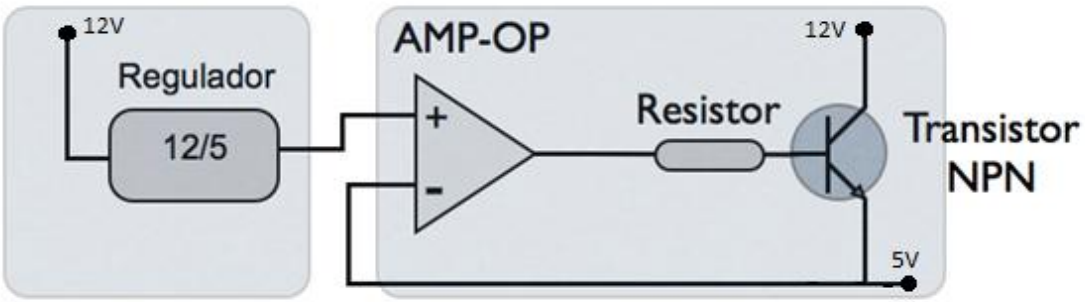


Figura 4.7 - Diagrama esquemático.

Fonte: O Autor

Na figura 4.8 estão apresentados os componentes como foram ligados na placa fisicamente.

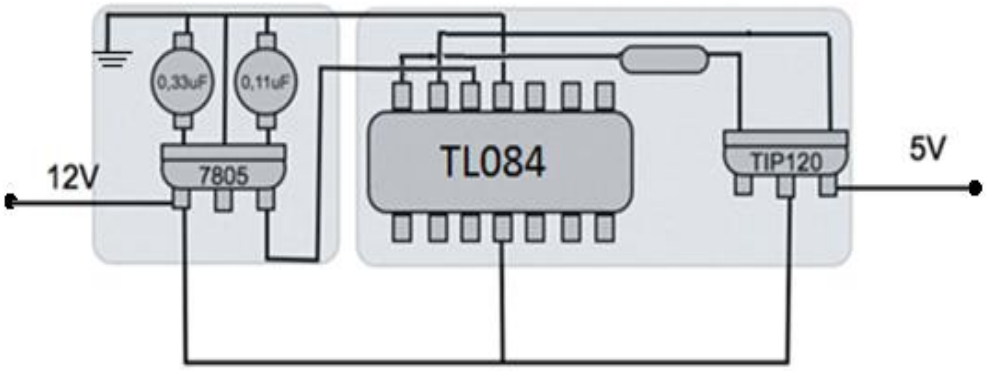


Figura 4.8 - Componentes Ligados

Fonte: O Autor

Na figura 4.9, está apresentado a placa confeccionada onde:

1 – Regulador de Tensão 7805;

2 – Amplificador Operacional TL084;

3 – Transistor TIP120.

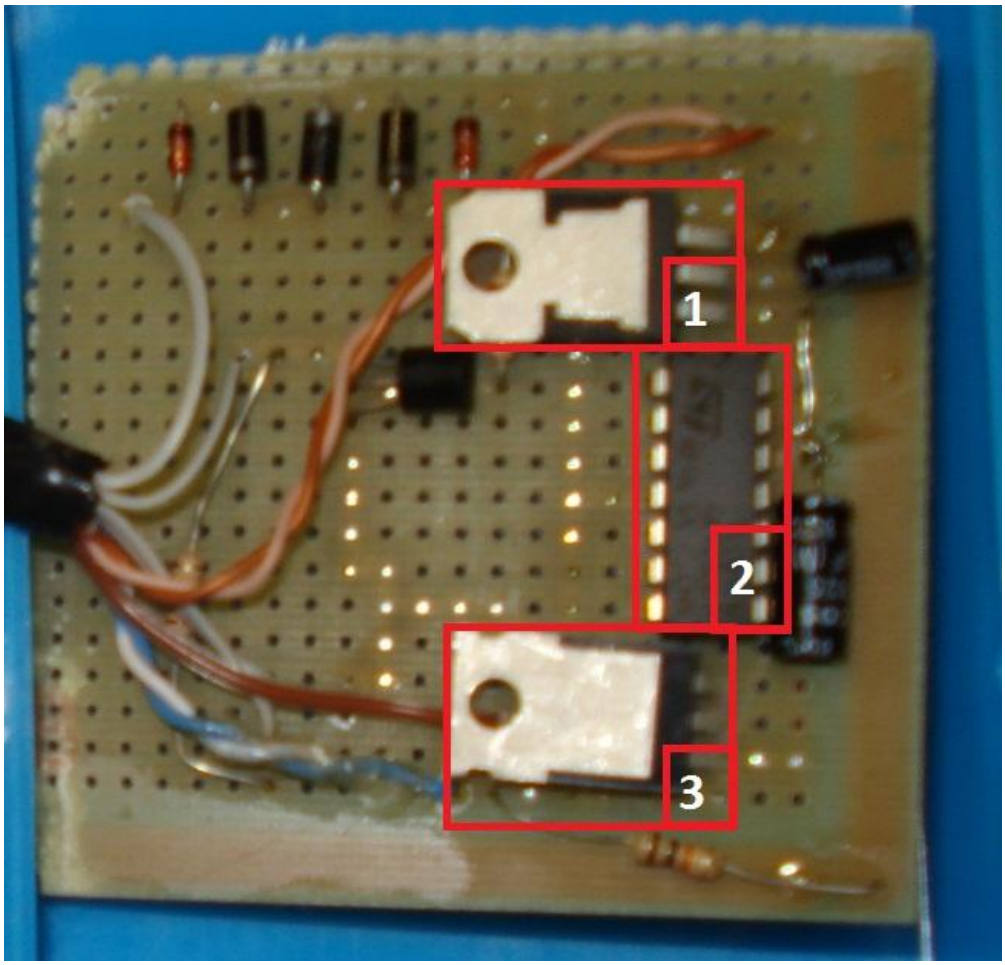


Figura 4.9 – Placa Confeccionada

Fonte: O Autor

Para realização dos testes da placa confeccionada, foi utilizada uma fonte 12V, como as que são utilizadas por computadores. Sendo assim, obteve-se êxito ao entrar com 12V e sair com 5V.

4.4 A Detecção do Sinal do Alarme

Após testes de comunicação bem sucedidos entre a placa do micro controlador e a placa GPRS, implementou-se a captura do sinal de acionamento do alarme. Desta forma, fez-se uma conexão do sinal da buzina do veículo, a qual geralmente é acionada pela maior parte dos alarmes, tornando o sistema apto para a grande maioria dos automóveis. No entanto, o sinal da buzina é acionado em 12V quando ligada e quando não, em 0V. Tornou-se necessário então, a utilização de um circuito regulador para baixar este sinal para 5V, valor de tensão utilizado pelo micro controlador. Esta adaptação,fez-se também foi feita na placa confeccionada como mostra a figura 4.3.

Durante testes realizados, observou-se que em eventuais acionamentos da buzina poderiam acionar erroneamente o alarme. Para tal, fez-se um tratamento no sinal da buzina, via software, onde se verificou a quantidade de vezes e o intervalo de acionamento da mesma. A função do programa que faz este tratamento, “alarme_disparou”, é mostrada no trecho de programa a seguir como descrito na função.

```
/**
 * Checa se o alarme disparou.
 * Quando o alarme é disparado, este envia um sinal para a buzina que varia no
 * tempo dentro de um determinado período, como a onda a seguir, onde p = período:
 *
 * (12V) |  ---------
 *       |  |       |
 * (0V)  |  |       |_____
 * ----------------------------------
```

```
 *          |  |- p/2 -|
 *          |  |------ p ------|
 *          |
 *
 * Esta função detecta se o alarme foi disparado comparando o valor do sinal atual
 * com o valor do sinal meio período atrás. Caso eles sejam iguais, o laço termina
 * sumariamente e a função retorna falso. Caso sejam diferentes, o contador de laços
 * é incrementado e os sinais da onda comparados são avançados em meio período e a
 * comparação é novamente feita. Após determinado número de laços sucessivos, o
 * laço é finalizado e a função retorna verdadeiro.
*/
boolean alarme_disparou(){
  boolean alarme = false; //atribui alarme desligado
  boolean sinal_anterior; //último sinal lido
  int qtd_laco = 0;//quantidade de vezes que o sinal foi detectado
  int alarme_count = 4;// limite para quantidade de vezes de acionamento da buzina
  int periodo = 2000;  //periodo da onda
  if (sinal_buzina()) {   //verifica se a buzina foi acionada
          delay(periodo/4);   // Avança para o meio da onda
          while (true){
           sinal_anterior = sinal_buzina();   // Guarda o sinal atual
                  delay(periodo/2);
                  if (sinal_buzina() == sinal_anterior){
                          alarme = false;
                          break;
                  }

                  if (qtd_laco >= alarme_count){  //laco que contabiliza a quantidade de vezes
que a buzina foi acionada, caso 4 vezes o sistema GPRS é acionado.
                          alarme = true;
                          break;
                  }
                  qtd_laco++;
          } return alarme;
  }
}
```

4.5 Versão Final do Protótipo

A versão final do protótipo é elaborada com o acoplamento da placa Arduino Mega, GPRS e placa de alimentação. Foram adicionados dois botões,

do tipo *push bottom*[2], para o fechamento da porta e detecção de chave de ignição. Na figura 4.10 é mostrado o circuito onde:

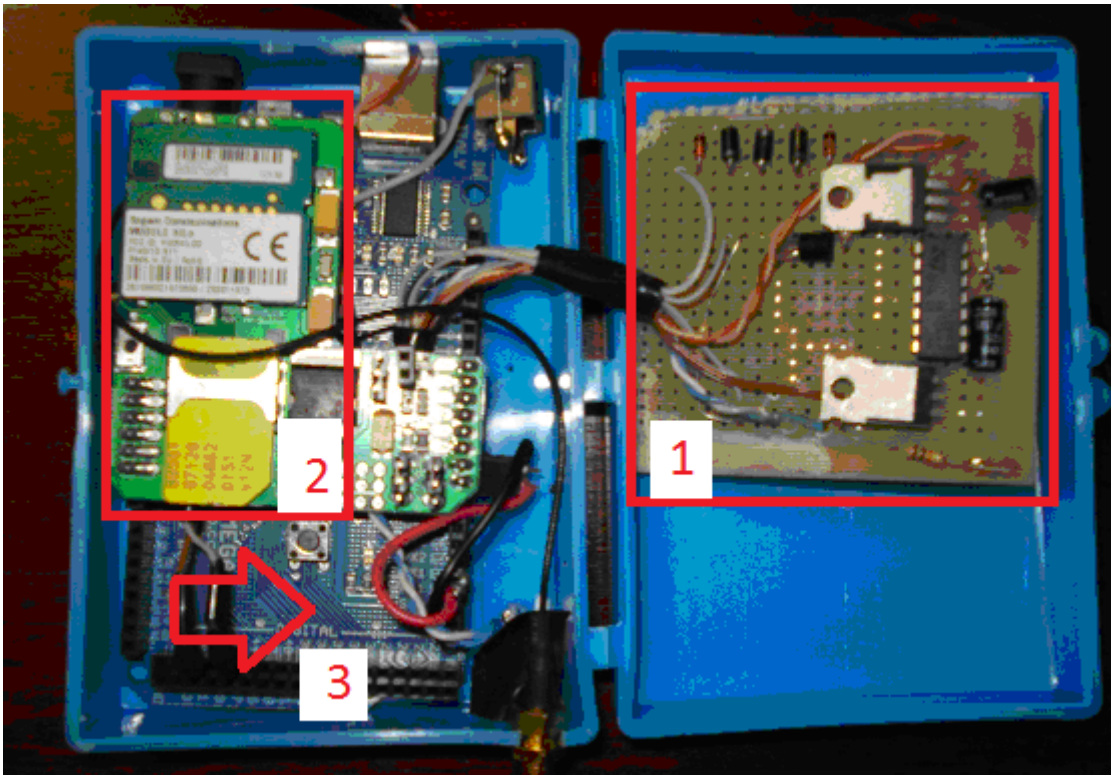


Figura 4.10 - Componentes Ligados

Fonte: O Autor

1- Circuito da Placa de alimentação;

2- Placa GPRS;

3- Placa Arduíno Mega.

[2] Push Bottom é um botão sensível a pressão.

5 - Resultados Obtidos e Análise dos Resultados

Para a fase de testes, a primeira constatação é a de que a placa GPRS funciona de forma satisfatória e atende as necessidades. O foco deste trabalho é a análise de todo um sistema de alarme e monitoração, e não o GPRS ou Arduino em si, embora tenha sido gasto uma enorme quantidade horas no desenvolvimento e aperfeiçoamento dos mesmos. Nos subitens abaixo, os resultados obtidos durante os testes de funcionamento foram detalhados.

5.1 Desenvolvimento da Programação em Linguagem C

Com o uso da Linguagem C é construído um protocolo de funcionamento para o controlador ATMEGA1280, e esse código, embora curto, permite que as informações sejam coletadas na entrada do sistema, tratadas dentro do controlador e posteriormente, que as saídas sejam acionadas.

Observou-se que, com o uso da plataforma Arduino, o tempo de desenvolvimento é menor, por se tratar de um ambiente de fácil compreensão, com uma grande variedade de funções específicas para o controlador em questão, que facilitam muito o trabalho de desenvolvimento e debug do código.

Com o uso da saída em tela via porta USB, à uma taxa de comunicação de 19200 bits/s, é possível observar as leituras realizadas na escala de referência, como também visualizar a ativação das saídas.

5.2 Testes Realizados

Nesta etapa, o ambiente de teste proposto foi configurado da seguinte forma:

- Utilização do veículo Toyota Corola ano 2004;

- Foram configurados todos os passos do sistema e em seguida o mesmo foi acoplado ao veículo.

Configurado o ambiente, iniciou-se a fase de testes onde foram estabelecidos dois casos de usabilidade, um considerando o esquecimento de uma criança no interior do veículo, e outro considerando o furto do automóvel.

5.3 Verificar Esquecimento

Para efetuar os testes quanto ao possível esquecimento de criança, foram consideradas as seguintes hipóteses e condições:

Atores: Criança, Veículo.

Descrição: Este teste tem como objetivo verificar a ausência de chave na ignição, portas fechadas e identificar peso no banco

Pré-condições: Ignição desligada, portas fechadas, identificação de peso no banco.

Requisitos: Carro, Arduino, Placa de Alimentação, GPRS, botões de identificação e pressão.

Cenário principal:

F1 – Identificar se a chave do veículo está na ignição;

F2 – Verificar se as portas estão fechadas;

F3 – Verificar se existe peso no banco com o sensor de pressão;

F4 – Caso exista peso no banco, enviar SMS a número pré-determinado.

A figura 5.1 demonstra o fluxo das fases.

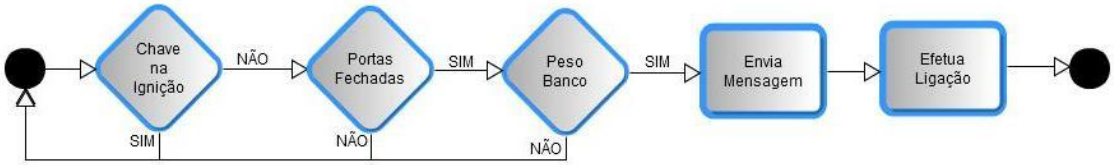


Figura 5.1 - Fluxo para Verificar Esquecimento

Fonte: O Autor

Exceções possíveis:

E1 – Caso as três condições não sejam satisfeitas a mensagem não será enviada.

Pós-condição: Após o envio da mensagem o sistema aguarda próxima detecção ou instrução.

Nesta etapa, foram utilizados dois botões, um para simular a presença ou não de chave na ignição e outro para o fechamento das portas com o intuito de não alterar o sistema original do veículo podendo causar maiores ônus posteriores.

Desta forma, obteve-se êxito testando os sensores disparados em diferentes ordens, mas sempre obedecendo a ordem pré-determinada como ilustrado pela figura.

5.4 Verificar Acionamento do Alarme

Então, iniciou-se o teste do seguinte. O sistema original dispara com a sirene e o pisca alerta. Com o carro parado em seu estado inicial, provocou-se o acionamento do alarme proposto. Para tal, se considerou as seguintes hipóteses e condições:

Atores: Carro

Descrição: Após o acionamento do alarme, enviar SMS e posteriormente ligar para o proprietário com número pré-determinado, funcionando paralelamente ao sistema original do veículo;

Pré-condição: Veículo parado em seu estado normal;

Requisitos: Carro, Arduíno, Placa de Alimentação, GPRS, Alarme original do veículo e botão de pressão;

Cenário Principal: Enviar SMS e posteriormente ligar para o proprietário do veículo com o acionamento do alarme.

F1 – Acionamento do alarme;

F2 – Verificar peso no banco;

F3 – Enviar mensagem informando que um possível furto pode estar ocorrendo.

F4 – Efetuar ligação.

Exceções possíveis:

E1 - Ao ligar e desligar o alarme, é enviado um sinal a mesma sirene que dispara o alarme e o sistema não deve enviar mensagens ou efetuar ligações.

A figura 5.2 ilustra o fluxo de funcionamento deste segundo teste:

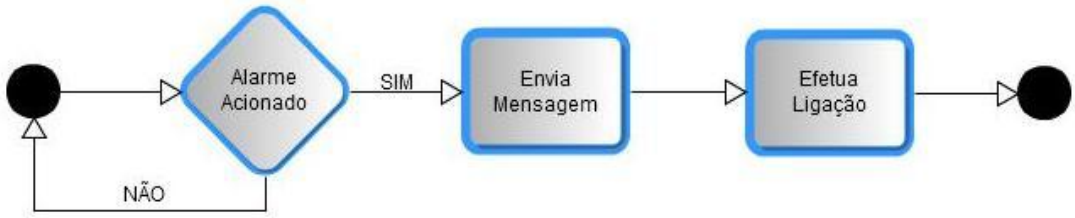


Figura 5.2 - Fluxo para Verificar Acionamento do Alarme

Fonte: O Autor

Exceção: Independente da verificação do peso nesta parte o sistema só realizará a função caso o alarme original do carro tenha sido acionado.

Pós-condição: Após efetuar ligação o sistema aguarda próxima detecção ou instrução.

Considerando esse caso de teste, não obteve-se êxito no sistema com as mensagens sendo enviadas e posteriormente sendo feita a ligação.

O sistema proposto em questão previa a detecção do acionamento do alarme quando a buzina do mesmo disparasse. Entretanto, a partir do erro ocorrido, verificou-se com o auxílio de um multímetro, que no fio na buzina só se recebia 12V quando a mesma era pressionada propositalmente, mas quando acionado o alarme, 0V. Então, constatou-se que o sistema de alarme original do veículo funciona a parte da buzina e possui uma sirene própria localizada do lado esquerdo do porta-malas. A partir daí, identificou-se o fio responsável por mandar 12V a esta sirene, onde posteriormente localizou-se o mesmo na barra inferior da porta do motorista juntamente com outros fios. Sendo assim, após a identificação deste fio, fez-se uma adaptação para o mesmo ser ligado a Arduino como mostra a figura 5.3 a seguir:

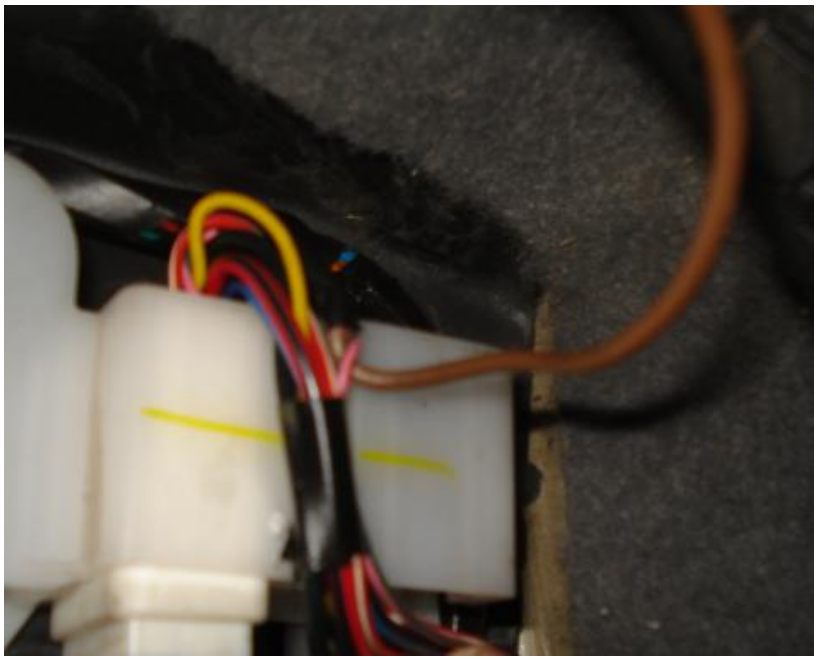


Figura 5.3 – Fio puxado da buzina.

Fonte: O Autor

Com estas alterações realizadas, os testes foram reiniciados obtendo-se êxito quando acionava-se o sistema, seja enviando mensagem ou realizando uma ligação obedecendo o fluxo como demonstrado na figura 5.4:

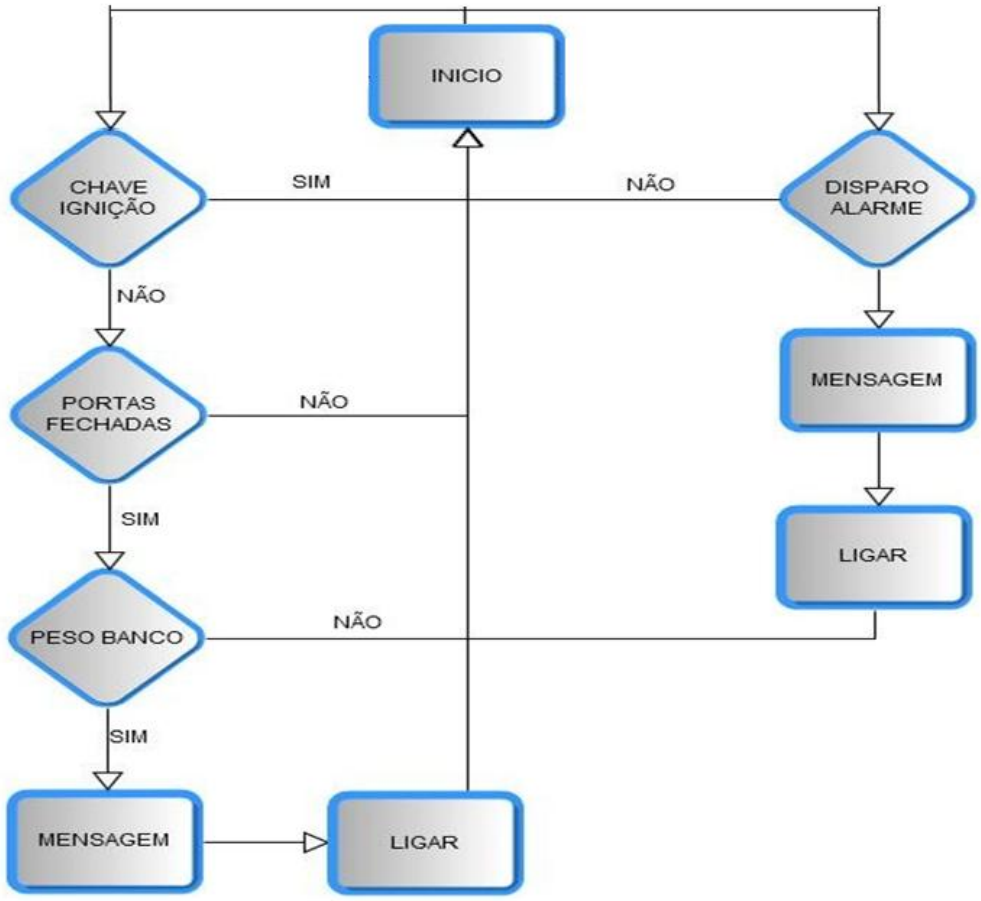


Figura 5.4 – Fluxo do sistema

Fonte: O Autor

6 Considerações Finais

Nesta sessão, são descritas as impressões do projeto a cerca de todo o processo evolutivo do desenvolvimento, bem como os resultados finais obtidos com a conclusão do protótipo.

6.1 Conclusão

Este projeto teve como finalidade o desenvolvimento de um sistema capaz de identificar possíveis esquecimentos de crianças no interior de automóveis além de possíveis furtos. Para isso, elaborou-se um projeto utilizando micro controlador e placa GPRS que ao analisar os sinais expostos pelo veículo, tomam uma ação, seja para o envio de mensagem ou mesmo uma ligação.

Paralelamente ao desenvolvimento dessa interface entre estes componentes, fez-se a codificação para analisar os sinais obtidos na linguagem C, que é suportada pelo Arduíno. Portanto, elaborou-se uma solução completa de recepção e análise de sinais.

Então, de acordo com o estudo e os testes realizados neste projeto, verificou-se que uma solução de alarme paralela aos sistemas originais dos veículos, funcionando juntamente sirenes, envios de mensagens e realização

de ligações. Demonstrado então, que a solução proposta neste projeto, tem grande disponibilidade e usabilidade. Para o perfeito funcionamento, além dos hardwares já citados, é necessária também a aquisição de um chip GSM com uma empresa de telefonia móvel.

Considerando a utilização das redes de telefonia, não foi constatado nenhum problema ou indisponibilidade dos serviços durante a fase de implementação, desenvolvimento e testes.

6.2 Dificuldades Encontradas

Durante o desenvolvimento, diversas situações negativas tiveram de ser transpostas, tanto no que se diz respeito ao conhecimento quanto à obtenção de materiais ou utensílios para confecção do modelo de protótipo. As dificuldades relacionadas ao conhecimento foram sanadas com pesquisas a literatura existente principalmente na internet. Houve uma maior necessidade de estudo, sobre o funcionamento e funcionalidades da placa GPRS, sobre a qual se possui ainda pouco material disponível. Porém, a maior dificuldade quanto a GPRS é o fato de não haverem revendedores no Brasil, apenas revendedores na internet e fora do país. Não obstante a dificuldade de adquirir a placa, o preço foi uma grande dificuldade unindo-se valor do produto e imposto pago já que a mercadoria veio de Zaragoza, ESP.

Pode-se assim, concluir que o objetivo deste projeto foi atingido com sucesso.

Como proposta para trabalhos futuros, sugere-se a melhoria do processo com a junção e análise dos demais sinais fornecidos pelo veículo. Além disso, seria interessante possibilitar uma forma de comunicação, por exemplo, com um sistema de rastreamento.

7 - Referências Bibliográficas

JUNIOR, Almir W. L. **Eletricidade & Eletrônica Básica.** 3ª Edição: Alta Books Editora, Rio de Janeiro, 2009

BANZI, Máximo **Getting Started with Arduino.** 3ª Edição: O'Reilly Media / Make, Outubro 2008

Mecatrônica Atual, Transdutores Piezoelétricos - Acesso em: 13.set.2010
http://www.mecatronicaatual.com.br/secoes/leitura/216

Regulador de Tensão – Acesso em 10.Set.2010

http://ivairijs.vilabol.uol.com.br/regulador1.html

Regulador de Tensão – Acesso em 10.Set.2010

http://www.guiadohardware.net/termos/regulador-de-tensao

Datasheets – Acessados diversas vezes entre Setembro - Outubro

http://www.datasheetcatalog.org/datasheet/stmicroelectronics/2301.pdf

http://www.datasheetcatalog.org/datasheet/stmicroelectronics/4128.pdf

ARDUINO – Acessados diversas vezes entre Agosto - Novembro

http://arduino.cc/en/Guide/Windows

http://www.arduino.cc/en/Tutorial/HomePage

http://arduino.cc/en/Main/ArduinoBoardMega

http://www.eletronica.com/arduino-tutorial/

GPRS - 10.Ago.2010

http://www.libelium.com/squidbee/index.php?title=New_GPRS_module_for_Arduino_%28Hilo_-_Sagem%29

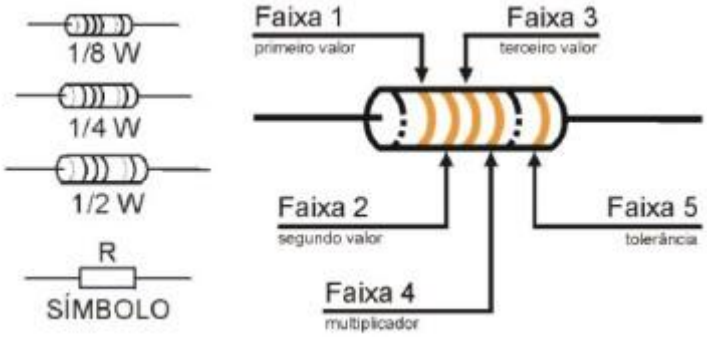
http://pt.wikipedia.org/wiki/Servi%C3%A7o_de_R%C3%A1dio_de_Pacote_Geral

Informações sobre o problema - 03.Dez.2010

http://www.parenthood.com/article-topics/summer_car_safety.html/full-view

8 Anexos

8.1 Tabela de Auxílio para Leitura de Resistores



Cores	Valores			Multiplicadores	Tolerência
	Faixa 1	Faixa 2	Faixa 3		
Prata	-	-	-	0,01	10%
Ouro	-	-	-	0,1	5%
Preto	0	0	0	1	-
Marrom	1	1	1	10	1%
Vermelho	2	2	2	100	2%
Laranja	3	3	3	1000	-
Amarelo	4	4	4	10000	-
Verde	5	5	5	100000	-
Azul	6	6	6	1000000	-
Violeta	7	7	7	-	-
Cinza	8	8	8	-	-
Branco	9	9	9	-	-
Nenhuma	-	-	-	-	20%

http://www.eletrica.info/tabela-de-resistores/

8.2 Funções do Arduíno

Estruturas de Funcionamento

void setup()
void loop()

Estruturas de Controle

if
if...else
for
switch case
while
do... while
break
continue
return
goto

Outras Sintaxes

;
{}
// (comentário de uma linha)
/* */ (comentário de várias linhas)
#define
#include

Operadores Aritméticos

= (operador de atribuição)
+ (adição)
- (subtração)
* (multiplicação)
/ (divisão)
% (módulo)

Operadores de Comparação

== (igual)
!= (diferente)
< (less than)

Funções Digitais I/O

pinMode(pino, modo)
digitalWrite(pino, valor)
int digitalRead(pino)

Analógicas I/O

analogReference(tipo)
int analogRead(pino)
analogWrite(pino, valor) - *PWM*

Avançadas I/O

shiftOut(pino de dados,pino do *clock*, bit de ordem,valor)
unsigned long pulseIn(pino, valor)

Tempo

unsigned long millis()
unsigned long micros()
delay(ms)
delayMicroseconds(us)

Matemática

min(x, y)
max(x, y)
abs(x)
constrain(x, a, b)
map(valor, do baixo, do alto, para baixo, para alto)
pow(base, expoente)
sqrt(x)

Trigonometria

sin(ângulo)
cos(ângulo)
tan(ângulo)

Tipos de Dados

void keyword
boolean
char
unsigned char
byte
int
unsigned int
word
long
unsigned long
float
double
string
array

Conversão

char()
byte()
int()
word()
long()
float()

Operadores Compostos

++ (incremento)
-- (decremento)
+= (adição composta)
-= (subtração composta)
*= (multiplicação composta)
/= (divisão composta)
&= (AND composto)
|= (OR composto)

Variáveis

HIGH | LOW

> (greater than) <= (less than or equal to) >= (greater than or equal to) **Operadores Booleanos** && (e) || (ou) ! (negao) **Ponteiros** * operador de deferência & operador de referência **Operadores Binários** & (AND) | (OR) ^ (XOR) ~ (NOT) << (deslocamento à esquerda) >> (deslocamento à direita)	**Números Randômicos** randomSeed(seed) long random(max) long random(min, max) **Bits e Bytes** lowByte() highByte() bitRead() bitWrite() bitSet() bitClear() bit() **Interrupções Externas** attachInterrupt(interrupção,função, modo) detachInterrupt(interrupção) **Interrupções** interrupts() noInterrupts()	INPUT | OUTPUT true | false integer constants floating point constants **Comunicação** Serial **Escopo de Variáveis e Qualificadores** variable scope static volatile const **Utilidades** sizeof() (sizeof operator)

APÊNDICE

```
int led = 13;
int onModulePin = 2;
int ledPin = 53;
int inPin_alarme = 46;
int inPin_peso_banco = 50;
int botao_vermelho = 45;  //ignicao
int botao_verde = 41;  //sensor porta
int alarme_count = 4;// limite para quantidade de vezes de acionamento da buzina
int periodo = 2000;//periodo da onda
unsigned long hora_ultima_mensagem;
unsigned long tempo_entre_alertas = 120000;
boolean primeira_alerta = true;
char* fone = "78130769";
boolean chave_na_ignicao_antes;
boolean porta_fechada_antes;


// funcao que inicializa o módulo GPRS que é utilizada na funcao seguinte.
void ligaModulo(){
          digitalWrite(onModulePin,HIGH);
           delay(2000);
           digitalWrite(onModulePin,LOW);
           delay(10000);
}

// iniciando o sistema e determia quais sao os pinos de entrada/saída.
//INPUT – Indica que o pino é de entrada
//OUTPUT – Indica que o pino é de saída
void setup() {
           pinMode(inPin_peso_banco, INPUT);
           pinMode(inPin_alarme, INPUT);
           pinMode(onModulePin, OUTPUT);

           Serial1.begin(19200);
           Serial.begin(19200);

           Serial.println("Ligando modulo...");
           Serial.println();

           ligaModulo();

           Serial.println("Modulo pronto");
           Serial.println();
}

//bloco lógico principal segmentado em funcões para melhor apresentacao/manutencao.
```

```
void loop(){
 if (alarme_disparou()) {
            Serial.println("Alarme disparou!");

            if (checa_se_hora_de_enviar_mensagem()) {
              envia_mensagem("Alarme disparado! Possivel furto!!!", fone);
              ligar(fone);
            }

 if (!chave_na_ignicao() && porta_fechada() && peso_banco()) {
            if (checa_se_hora_de_enviar_mensagem()) {
              envia_mensagem("Possivel crianca esquecida no carro!", fone);
              ligar(fone);
            }
 }

 if (porta_fechada() != porta_fechada_antes) {
            porta_fechada_antes = porta_fechada();
            if (porta_fechada()) {
              Serial.println("Porta fechada");
            } else {
              Serial.println("Porta aberta");
            }
 }

 if (chave_na_ignicao() != chave_na_ignicao_antes) {
            chave_na_ignicao_antes = chave_na_ignicao();
            if (chave_na_ignicao()) {
              Serial.println("Chave na ignicao");
            } else {
              Serial.println("Chave fora da ignicao");
            }
          }

}

boolean checa_se_hora_de_enviar_mensagem() {
          unsigned long tempo_desde_ultima_mensagem = millis() - hora_ultima_mensagem;
          if (tempo_desde_ultima_mensagem > tempo_entre_alertas || primeira_alerta) {
                    primeira_alerta = false;
                    hora_ultima_mensagem = millis();
                    return true;
          }
 return false;
}


/**
 *  Esta funcao, captura o sinal inicial da buzina que posteriormente é tratado
 *  na funcao alarme_disparou() e assim detectando se é ou nao um disparo acidental
 *  da buzina ou um disparo do alarme.
*/
```

```
boolean sinal_buzina(){
	return digitalRead(inPin_alarme);
}

/**
 *  Esta funcao, captura o sinal do botao de pressao que caso pressionado libera
 *  HIGH ou 1 e caso nao pressionado libera LOW ou 0.
*/
boolean peso_banco(){
	return digitalRead(inPin_peso_banco);
}

/**
 *  As funcões sinal_chave e sinal_porta capturam o sinal da chave e da porta respectivamente
 *  liberando HIGH ou 1 e caso nao pressionado libera LOW ou 0.
*/
boolean chave_na_ignicao(){
	return digitalRead(botao_vermelho);
}

boolean porta_fechada(){
	return digitalRead(botao_verde);
}

/**
 * Esta funcao é responsável por enviar mensagem ao proprietário, caso seja
 * detectado pressao no banco do carro.
*/
void envia_mensagem(char *mensagem, char *fone) {
	Serial1.print("AT+CMGS=");
	Serial1.print(34, BYTE);
	Serial1.print(fone);
	Serial1.println(34, BYTE);
	delay(1500);
	Serial1.print(mensagem);
	delay(500);
	Serial1.print(0x1A, BYTE);

	// debug
	Serial.println("Enviei a mensagem:");
	Serial.println(mensagem);
	Serial.print("para o fone : ");
	Serial.println(fone);

	// para dar tempo para o módulo processar
	delay(4000);
}

/**
 *  Esta funcao liga para o usuário, aguarda 12segs dentro do processo e encerra a ligacao.
 *  Esta funcao é chamada sempre que detectado o disparo do alarme, sendo detectada ou
 *  nao pressao entre os bancos.
```

```
*/
void ligar(char *fone){
        delay(1000);
        Serial1.print("ATDT");
        Serial1.print(fone);
        Serial1.println(";");
        Serial.print("ATDT");
        Serial.print(fone);
        Serial.println(";");

        // espera um tempo para a pessoa perceber que o fone esta tocando
        delay(30000);

        // desliga
        Serial1.println("ATH");
        Serial.println("ATH");

        // para dar tempo para o módulo processar
        delay(4000);
}

boolean alarme_disparou() {
        if (sinal_buzina()) {
          Serial.println("Sinal de alarme disparado!");
          delay(periodo);//Para verificar se o alarme não está sendo ligado/desligado
                if (sinal_buzina()) {
                  Serial.println("Eh alarme mesmo!");
                  return true;
                }
          Serial.println("Alarme falso!");
        }
        return false;
}

/**
 * Checa se o alarme foi disparado.
 * Quando o alarme é disparado, este envia um sinal para a buzina que varia no
 * tempo dentro de um determinado período, como a onda a seguir, onde p = período:
 *
 * (12V) |  ---------
 *       |  |       |
 * (0V)  |  |       |_______
 * ----------------------------------
 *       |  |- p/2 -|
 *       |  |------ p ------|
 *       |
 *
 * Esta funcao detecta se o alarme foi disparado comparando o valor do sinal atual
 * com o valor do sinal meio período atrás. Caso eles sejam iguais, o laco termina
 * sumariamente e a funcao retorna falso. Caso sejam diferentes, o contador de lacos
 * é incrementado e os sinais da onda comparados sao avancados em meio período e a
 * comparacao é novamente feita. Após determinado número de lacos sucessivos, o
```

```
 * laco é finalizado e a funcao retorna verdadeiro.
*/
boolean alarme_disparou_buzina_carro(){
            boolean alarme = false; //atribui alarme desligado
            boolean sinal_anterior; //último sinal lido
            int qtd_laco = 0;//quantidade de vezes que o sinal foi detectado
            if (sinal_buzina()) {//verifica se a buzina foi acionada
                Serial.println("Chegou sinal da buzina");
                    delay(periodo/4); // Avanca para o meio da onda
                    while (true){
                     sinal_anterior = sinal_buzina(); // Guarda o sinal atual
                                delay(periodo/2);
                                if (sinal_buzina() == sinal_anterior){
                                            alarme = false;
                                    Serial.println("Nao é alarme");
                                            break;
                                }

                                if (qtd_laco >= alarme_count){//laco que contabiliza a quantidade de
          vezes que a buzina foi acionada, caso 4 vezes o sistema GPRS é acionado.
                                            alarme = true;
                                    Serial.println("É alarme");
                                            break;
                                }
                                qtd_laco++;
                    } return alarme;
            }
}
```