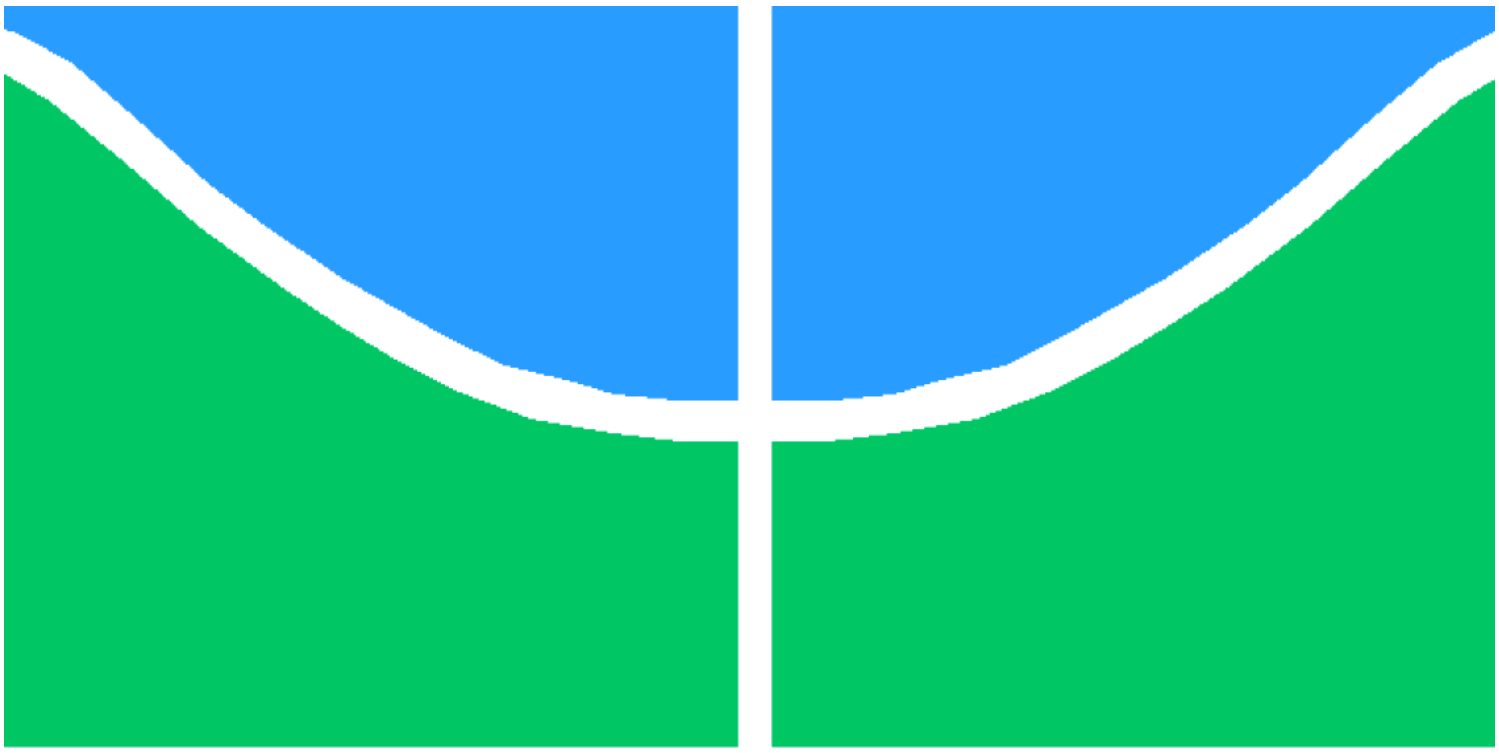


DISSERTAÇÃO DE MESTRADO

PROPOSTA DE PLATAFORMA INERCIAL PARA AUXILIAR NA PERÍCIA DE ACIDENTES DE TRÂNSITO

Vinícius de Oliveira Lima

Brasília, Dezembro de 2016

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROPOSTA DE PLATAFORMA INERCIAL PARA AUXILIAR NA PERÍCIA DE ACIDENTES DE TRÂNSITO

VINÍCIUS DE OLIVEIRA LIMA

ORIENTADOR: RICARDO ZELENOVSKY

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.DM – 638/16

BRASÍLIA/DF: DEZEMBRO - 2016

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DISSERTAÇÃO DE MESTRADO

PROPOSTA DE EMPREGO DE GIROSCÓPIO E ACELERÔMETRO NA PERÍCIA DE ACIDENTES DE TRÂNSITO

VINÍCIUS DE OLIVEIRA LIMA

DISSERTAÇÃO DE MESTRADO PROFISSIONAL SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

_____

RICARDO ZELENOVSKY, DOUTOR, PUC-RJ, Unb/ENE (ORIENTADOR)

_____

EDSON MINTSU HUNG, DOUTOR, Unb/ENE (EXAMINADOR INTERNO)

_____

MAURICIO SERCHELLI, DOUTOR, UNICAMP-SP, FÍSICA (EXAMINADOR EXTERNO)

BRASÍLIA, 21 DE DEZEMBRO DE 2016.

FICHA CATALOGRÁFICA

LIMA, VINÍCIUS DE OLIVEIRA

Proposta de emprego de giroscópio e acelerômetro na perícia de acidentes de trânsito [Distrito Federal] 2016.

v,68P., 210x297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2016).

Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia Departamento de Engenharia Elétrica.

I. ENE/FT/UnB

1

REFERÊNCIA BIBLIOGRÁFICA

LIMA, VINÍCIUS DE OLIVEIRA (2016). Proposta de emprego de giroscópio e acelerômetro na perícia de acidentes de trânsito [Distrito Federal] 2016. Dissertação de Mestrado em Engenharia Elétrica, Publicação PGENE.DM-638/16, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 68p.

CESSÃO DE DIREITOS

AUTOR: Vinícius de Oliveira Lima.

TÍTULO: Proposta de emprego de giroscópio e acelerômetro na perícia de acidentes de trânsito

GRAU/ANO: Mestre/2016.

científicos. O autor reserva outros direitos de publicação e nenhuma parte desta dissertação de mestrado pode ser reproduzida sem a autorização por escrito do autor.

_____

Vinícius de Oliveira Lima
Brasília – DF – Brasil.

À minha família, que sempre teve como pilar a educação.

Agradecimentos

A Deus por ter me colocar frente a obstáculos difíceis, mas necessários para o meu crescimento. É na dificuldade que damos o devido valor.

À minha família e, principalmente a minha esposa, pela paciência e compreensão. Foi um ano atípico de casamento e mestrado, mas sua parceria só fez mostrar que fiz a escolha certa. Espero sempre poder retribuir a altura.

Ao prof. Zelenovisky pela confiança e auxílio, qualidades nobres cada vez mais raras entre as pessoas.

Vinícius de Oliveira Lima

RESUMO

O presente trabalho é um estudo aos módulos de *airbag* veiculares, diante das importantes informações que esse dispositivo pode fornecer, sobre um acidente, principalmente aos peritos da área. Conhecido pelo termo técnico *EDR (Event Data Recorder)*, a variedade desse tipo de equipamento é grande e por isso esse trabalho teve como foco um modelo específico comum ao Renault/Scenic. Foi criado um programa para leitura do módulo objeto de estudo e feito alguns experimentos para acionar seu algoritmo de deflagração das bolsas. Além da escassa literatura no assunto, o resultado desses experimentos foi debatido e colocado em questão para criação de um equipamento que pudesse auxiliar os peritos na extração de dados, como a velocidade, no momento de um acidente. O equipamento desenvolvido, de baixo custo e simples, é capaz de capturar dados de aceleração e giro nos instantes que antecedem e sucedem a colisão. Com esses dados é possível traçar gráficos ou ainda visualizar o comportamento do veículo em uma simulação tridimensional.

ABSTRACT

The present work is a study in the matter of *airbag* modules, which can give important information about an accident, especially to forensics experts. Known as the technical name of EDR (Event Data Recorder), the variety of this equipment is huge and because of that, this work is dedicated to a Renault/Scenic typical module. A code was built to read it and some experiments were made to enable its deployment algorithm. Beyond the poor literature in the matter, the results of the experiments started a discussion that made the development of a new equipment that could be useful for forensics experts extract data like velocity in a accident, for example. The projected device is cheap and simple and captures acceleration and gyro seconds before and after the collision. With these values, we are able to plot graphics or even 3D simulations.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ABS	*Analogic to digital converter*
ACM	*Airbag Control Module*
ASCII	*American Standard Code for Information Interchange*
BCM	*Brake Control Module*
CAN	*Controller Area Network*
CPU	*Central Processing Unit*
CS	*Chip Select*
DI	*Data Input*
DO	*Data Output*
DTC	*Diagnostic Trouble Code*
ECU	*Eletronic Control Unit*
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
EWDS	*Erase and Write Disable*
EWEN	*Erase and Write Enable*
GPS	*Global Positioning System*

I2C	*Inter-Integrated Circuit*
MISO	*Master Input Slave Output*
MOSI	*Master Output Slave Input*
PCM	*Pulse code modulation*
NIV	Número de Identificação Veicular
OBD	*OnBoard Diagnostics*
PCM	*Powertrain Control Module*
SCL	*Signal Clock*
SDA	*Signal Data*
SS	*Slave Selection*
SOIC	*Small Outline Integrated Circuit*
TWI	*Two Wire Interface*
VIN	*Vehicle Identification Number*

1 INTRODUÇÃO

De uma forma geral, acidentes de trânsito que causam lesão corporal necessitam, por lei, da presença de equipe de perícia criminal, após o isolamento e preservação do local do fato [1]. Seu trabalho é analisar os vestígios materiais encontrados e, com base em fundamentos técnicos e científicos, reconstruir a cena do acidente. Essa análise é protocolizada em forma de laudo e encaminhada às autoridades competentes (delegado ou juiz). No Brasil, é competência da Polícia Técnica ou Científica realizar essa tarefa.

Dentre os vestígios mais comuns encontrados estão fragmentos desprendidos dos veículos, avarias decorrentes da colisão, posição de repouso final dos veículos envolvidos e o principal deles: marcas pneumáticas recentes ao evento. É com a presença desse último elemento, que na maioria dos casos os peritos criminais, a partir de cálculos físicos, determinam a velocidade e orientação dos veículos que deixaram esse rastro.

Ocorre que nem sempre esse elemento de interesse pericial está presente na cena, principalmente quando se tratam de marcas de frenagem. Para piorar, a tendência é que essas marcas sejam vestígios cada vez mais raros, haja a vista a presença cada vez maior dos sistemas de freios do tipo ABS. Nesse sistema, as rodas não travam totalmente com o acionamento do pedal de freio como no antigo. O ABS, na verdade, permite que elas girem e travem em frações de segundo, possibilitando maior eficiência que o condutor tenha maior dirigibilidade durante uma frenagem. Na prática isso resulta em pouco ou quase nenhum rastro. No Brasil desde de 2014 é obrigatória a presença do ABS em todos os novos veículos fabricados (nacionais e importados), conforme resolução 380/11 do CONTRAN [2].

O quer fazer, então, quando não há marcas pneumáticas como vestígios? Como estabelecer uma velocidade no laudo pericial? Os outros vestígios permitem estabelecer diferentes formas de cálculos da velocidade, porém não são muito precisas e por conta disso, na maioria das vezes, são descartadas pelos peritos criminais (são cálculos menos objetivos e com pouca aceitação na comunidade científica), deixando assim o laudo sem um valor de velocidade confiável, dado esse primordial.

Porém, a tecnologia presente nos veículos mais modernos viabiliza uma outra solução. Os automóveis modernos são divididos em diversos módulos conhecidos como *ECUs (Eletronic Control Unit)*, que são dispositivos capazes de controlar e tomar decisões sobre os mais diversos sistemas elétricos [3]. São vários os tipos de *ECU: PCM (Powertrain Control Module), SCM (Suspension Control Module), BCM (Brake Control Module), ACM (Airbag Control Module).* Eles se comunicam entre si por meio de protocolos específicos, sendo o padrão *CAN (Controller Area Network)* o mais comum deles.

Aqui, nosso interesse está no *ACM.* Essa é a unidade responsável por tomar as decisões sobre quando acionar os *airbags* e os pré-tensionadores dos cintos de segurança em situações de colisão. Essa decisão é feita com base na leitura de sensores espalhados no veículo, tais como acelerômetros, sensores de impacto e pressão, sensores de velocidade de giro da roda, giroscópios, sensores de presença e sensores de pressão dos freios[1]. Para cada um desses módulos, foi desenvolvido um algoritmo específico que monitora os valores medidos pelos sensores e, quando detecta algo que indique um impacto suficientemente grande, decide pela deflagração das bolsas e acionamento dos pré-tensionadores. Nos

[1] Os tipos e quantidade de sensores variam de modelo e marca. A forma mais básica é ter pelo menos um acelerômetro.

circuitos dessa unidade há a presença de memórias não voláteis, responsáveis por armazenar os dados lidos pelo algoritmo. Apesar não ter esse como objetivo, o módulo de *airbag* funciona, portanto, como uma verdadeira caixa-preta em acidentes de trânsito, guardando os dados da colisão, que são extremamente úteis para a perícia [4].

No mundo forense, esse tipo de dispositivo é conhecido como *EDR (Event Data Recorder)*. Criado inicialmente para auxiliar nas análises de medidas de segurança de carros de corrida [5], os EDRs hoje em dia são capazes, basicamente, de registrar e armazenar informações como diferença de velocidade (delta-V^2), se o cinto de segurança estava afivelado ou não, se o pedal de freio foi pressionado ou não, o giro do veículo no momento colisão, dentre outros parâmetros a depender do modelo. Portanto, a aquisição desses dados permitiria uma reconstrução (parcial) de cena de acidente de trânsito muito mais fidedigna.

1.1 MOTIVAÇÃO DO ESTUDO

O primeiro grande problema para os peritos da área reside no fato de não existir uma padronização rígida para a forma de armazenamento dos dados desses módulos. Cada montadora de automóveis segue uma formatação específica e ainda assim emprega módulos diferentes de acordo com os modelos de seus carros. Nos EUA a NHTSA (*National Highway Traffic Safety Administration*) criou uma norma que obriga aos veículos fabricados a partir de setembro de 2010, equipados com as caixas-pretas, a armazenarem no mínimo, 15 tipos de informações específicas em formatos padrões e, ainda, exige que o acesso a esses dados esteja disponível para equipamentos comerciais [6]. Dentre essas

[2] Definido na física como sendo igual à aceleração integrada no tempo.

informações está, por exemplo, a velocidade pré-colisão, uso do freio, uso dos cintos e estado da luz de aviso do *airbag*. A norma acrescenta ainda que é necessário prover os valores dos parâmetros armazenados durante os 5 segundos anteriores à colisão, a cada 0,5 segundo. Todavia a padronização criada pela NHTSA é intrínseca a cada tipo de dado e não na forma em que se dá o armazenamento propriamente dito. Por exemplo, essa norma estabeleceu que a aceleração longitudinal deve ser armazenada num intervalo de – 50 a + 50 g[3], com uma precisão de +/- 5 % e com uma resolução de 0,01 g; mas não especificou quantos bits serão necessários para cada amostra ou em que localização do arquivo deve estar armazenado. Como são muitas montadores e muitos tipos diferente de módulos, cada um armazena os dados como bem entende e, portanto, a leitura pura e simples da memória não é trivial. A tradução e interpretação dos dados binários armazenados nesses módulos, acaba sendo de domínio exclusivo das montadoras.

No mercado existe a disponibilidade de alguns equipamentos capazes de decodificar os bits das memórias e gerar um relatório com os dados da colisão. Esses equipamentos são conhecidos como *CDR* (*Crash Data Retrival*) e o principal deles é o desenvolvido pela multinacional *Bosch*. Esse aparelho já é amplamente utilizado na análise forense nos EUA. Todavia ele, além de caro, foi criado com o intuito de atender aquela frota e, portanto, possui pouca utilidade no Brasil. Os outros tipos de instrumentos *CDR* basicamente restringem-se aos desenvolvidos pelas próprias montadoras e, na maioria das vezes, não são comercializadas e seu uso é exclusivo de suas matrizes.

Para piorar o cenário, no Brasil não existe qualquer legislação sobre o assunto. As concessionárias brasileiras são livres para armazenar ou não as informações de caixa-preta

[3] Neste trabalho a unidade "g" é utilizada como medida de aceleração, em que 1 g representa a aceleração da gravidade de 9,80 m/s².

em seus módulos, da forma como quiserem, sem ter que dar satisfação quando questionadas pela Justiça ou Polícia. Nos casos de acidentes fatais de maior repercussão, foi preciso retirar o módulo do veículo e enviá-lo às suas matrizes de origem com uma expectativa de resposta, o que toma muito tempo e não necessariamente é confiável. Para piorar ainda mais, uma literatura ou mesmo um debate no âmbito legislativo sobre este assunto não parece estar perto de acontecer.

1.2 ESBOÇO DO TRABALHO

Tendo esse problema em mente e dado o anseio da comunidade forense em algo que possa ser útil no auxílio da reconstrução de acidente trânsito, este trabalho visou estudar alguns dos módulos de *airbags* extraídos em locais de acidente em que houve o acionamento do algoritmo de deflagração das bolsas. Basicamente, o trabalho foi dividido em 3 partes: Ambientação (Capítulo 2); leitura, interpretação dos dados coletados e testes de acionamento do algoritmo (Capítulo 3), proposta de emprego de acelerômetro e giroscópio em veículos (Capítulo 4), simulação gráfica dos dados coletados (Capítulo 5) e conclusões (Capítulo 6).

A primeira parte é para ambientar o leitor na parte técnica relativa a acidentes e que será abordada em diante. São explanações sobre os protocolos utilizados e os termos técnicos.

Na segunda parte foi feita uma análise dos módulos questionados e estudou-se um tipo específico de memória, o Autoliv 550 56 90 00, oriundo de um *Renault/Scenic*. Realizou-se sua leitura e fez-se testes para entender seu funcionamento em uma simulação de colisão. Essas simulações consistiam em simplesmente deixar o módulo cair em queda livre sobre uma superfície rígida, com a intenção de gerar um impacto grande o suficiente a

ponto de acionar o algoritmo, para depois observar quais campos de bits foram alterados. A partir daí foi feita uma tentativa de interpretação em busca de informação útil para a perícia.

Diante da problemática já mencionada e da necessidade de algo que pudesse auxiliar os peritos da área, a terceira parte do trabalho mostra o desenvolvimento de um equipamento que pode ser acoplado nos automóveis, que fosse capaz de realizar leituras de aceleração e giro no momento de uma colisão. Este equipamento possui características semelhantes à dos módulos de *airbag* atuais, mas com a vantagem de ser mais simples e barato e até de auxiliar na compreensão do funcionamento desses sistemas.[4]

Além de traçar os gráficos dos resultados obtidos, no capítulo 5, trazemos uma inovação para área forense, que é a possibilidade de visualizarmos o comportamento do veículos nos instantes do acidente em forma de simulação 3D. Com auxílio da linguagem *Processing* (semelhante a *java*), foi possível criar um desenho 3D de um carro que girasse conforme as medições no equipamento desenvolvido. Os valores de aceleração no período gravado pelo dispositivo também podem ser visualizados nas simulações.

2 AMBIENTAÇÃO

2.1 O MÓDULO DE *AIRBAG* E OS TIPOS EXISTENTES

Como já foi brevemente descrito, o módulo de *airbag* nada mais é do que um circuito com um processador capaz de decidir quando, em uma possível colisão, é necessária a deflagração das bolsas e ativação dos pré-tensionadores dos cintos de segurança. Um algoritmo em tais sistemas é o responsável por deliberar sobre a deflagração, e decide com base nos dados recebidos de sensores que monitoram diversos

[4] Todos os códigos criados para este trabalho estão a disposição por meio do *email* do autor.

parâmetros, sendo a aceleração o principal deles. Um conversor analógico-digital discretiza os sinais colocando-os numa forma digital para serem interpretadas pelo processador [4].

Em uma colisão, o veículo envolvido é submetido a uma força de ação contrária ao seu movimento, provocando assim uma desaceleração brusca que, sendo maior do que a definida como limiar pelo algoritmo, faz com que o processador emita sinais para ativar as bolsas, os pré-tensionadores e também para armazenar os parâmetros da colisão em sua memória. Uma segunda colisão, após alguns minutos, não tem efeito no circuito, pois o algoritmo também provoca o travamento da leitura dos sensores e impede uma nova escrita na memória. Ou seja, o módulo para de funcionar até que seja submetido a uma inicialização, chamada geralmente de *reset* [20]. Na linguagem técnica, chamamos de código *clean*, aquele original de fábrica presente na memória de um módulo que não foi submetido ao acionamento do algoritmo de deflagração. Quando ocorre a deflagração, temos na memória o que chamamos de código *crash*.

As concessionárias no Brasil são contra o *reset* do aparelho. O que elas indicam, na verdade, é troca de todo o sistema de *airbag*, inclusive do módulo central por um novo. Porém, algumas oficinas detém o código *clean* de determinados módulos e o que elas fazem é, na verdade, simplesmente escrever esse código sobre o de *crash,* além de trocar as bolsas. Esse procedimento acaba por ser bem mais barato que o das concessionárias.

A localização dos módulos de *airbag* não segue um padrão rigorosamente definido, mas geralmente situam-se próximos do centro de massa dos automóveis: abaixo do console central ou do freio de mão. A Figura 2.1 mostra um exemplo da localização do módulo em um *Nissan/Frontier*. Já os sensores podem estar espalhados pela estrutura do veículo ou ainda inseridos diretamente na placa do módulo. Os mais comuns são os veículos com apenas *airbags* dianteiros, os quais possuem acelerômetros próximos ao para-choque

dianteiro e/ou inseridos no interior do módulo. Nesse caso, o circuito trabalha apenas com colisões frontais.

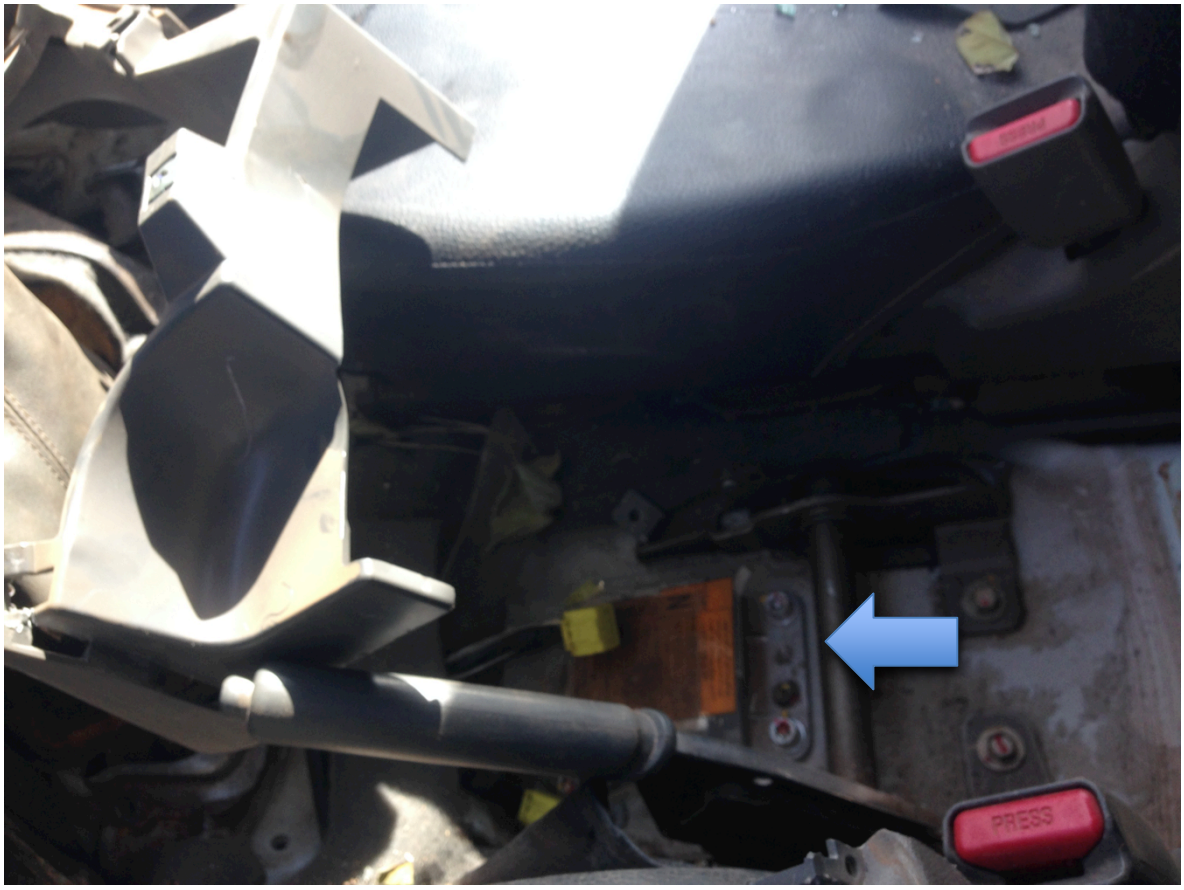


Figura 2.1: Localização do módulo de *airbag* de uma caminhonete *Nissan/Frontier* envolvida em capotamento.

2.2 MEMÓRIAS *EEPROMs*

A memória existente no circuito de tais módulos é uma *EEPROM* (*Eletrically Erasable Programmable Read-Only Memory*). É uma memória do tipo não volátil, o que permite com que a informação não se perca quando a alimentação é interrompida. Nesse trabalho a leitura dos módulos foi feita diretamente sobre essa memória, coletando-se assim os dados brutos armazenados.

Os modelos de *EEPROM* também não são padrão e, portanto, a forma de se extrair os dados varia de módulo para módulo. Alguns circuitos ainda não apresentam sequer essa memória de forma aparente, estando ela incorporada ao processador. Buscando a simplicidade e praticidade, este estudo trabalhou apenas com aquelas placas onde foi possível localizar a *EEPROM* e esta estava externa ao processador.

As *EEPROMs* são tipicamente encapsuladas no formato para montagem de superfície, denominado Soic8 (8 pinos). As fabricantes mais comuns encontradas nesse trabalho são *Atmel, FairChild e Microchip*. Para a correta leitura de cada memória foi necessário consultar o *datasheet* do componente e verificar parâmetros como número de bits de endereçamento e de dados, tensão de alimentação, valor do relógio e modos de escrita e leitura. Pela pesquisa nesse trabalho, percebeu-se que a maioria das *EEPROMs* presentes nos módulos analisados, possuíam 16 ou 8 bits de endereço/dados, e 5 V de alimentação. Para cada memória ainda existe um protocolo específico de acesso, como o *I2C* (*Inter-Integrated Circuit)* e o *SPI* (*Serial Peripheral Interface)*.

Figura 2.2: Exemplo de circuito do módulo de *airbag* de um automóvel *Renault/Scenic*.

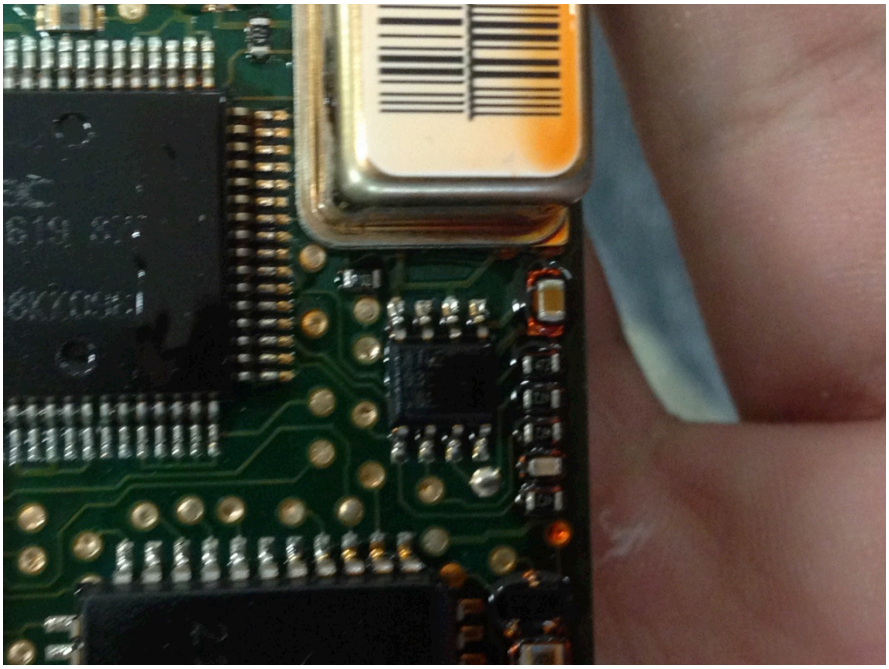


Figura 2.3: Mostra *EEPROM* do circuito, uma *Atmel* 93LC66.

Figura 2.4: Mostra acelerômetro do circuito, uma *Temic* Bax50.

As Figuras 2.2 a 2.4 apresentam como exemplo o circuito contendo uma memória *EEPROM* aparente e um acelerômetro, que foi retirado de um *Renault/Scenic* e que será objeto de estudo mais adiante.

2.3 FORMAS DE ACESSO À MEMÓRIA

São basicamente duas as formas de acesso à memória dos módulos: por meio de *dump* direto a memória ou por intermédio de *scanners* específicos como o citado da Bosch e das montadoras.

Esse segundo método tem a vantagem de ser mais prático, pois não necessita da retirada completa do módulo para sua leitura. Os scanners desse tipo conectam-se à rede CAN do veículo através da porta *OBD (On-Board Diagnostis*) cujo conector está situada geralmente abaixo do volante [16]. Por meio de protocolos de pergunta e resposta tais *scanners* são capazes de acessar o módulo de *airbag* e extrair suas informações. Outra vantagem é que a maioria desses equipamentos tem conexão com um *laptop* e com auxílio

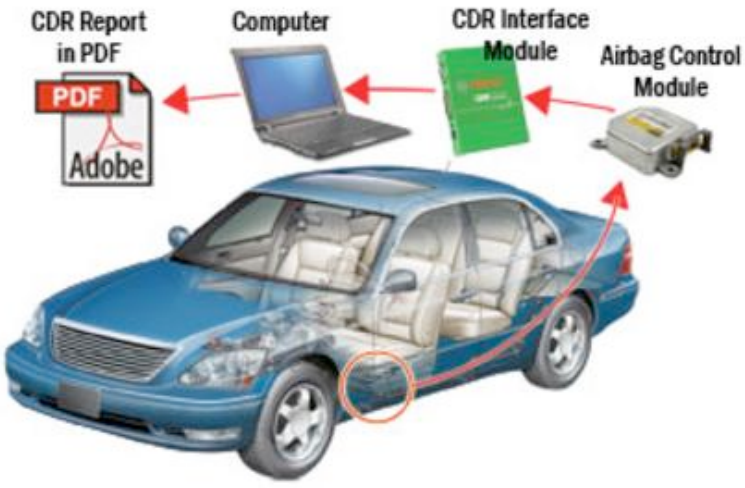


Figura 2.5: Mostra esquema de funcionamento do *CDR* da *Bosch* [8].

do *software* adequado, é capaz de já gerar um relatório com os dados do acidente, conforme ilustra Figura 2.5. Para o caso de um veículo colidido desprovido de energia, pode-se também fazer o acesso pela porta serial do próprio módulo.

Porém esses *scanners*, além de caros, muitas vezes não são capazes de interpretar as informações de veículos fabricados no Brasil. Isso ocorre porque os relatórios são gerados a partir de uma consulta na base de dados de NIVs (Número de Identificação Veicular ou em inglês *VIN – Vehicle Number Identification*) tipicamente americano. O aparelho da Bosch é o mais utilizado na investigação forense da América do Norte, mas como ele foi padronizado conforme a frota americana, os NIVs daqui, atualmente, possuem codificação diferente [8]. Já as oficinas e concessionárias brasileiras até possuem *scanners* com acesso ao módulo de *airbag*, mas as informações coletadas resumem-se aos códigos de erros (também conhecidos como *DTC – Diagnostics Trouble Codes*), os quais apresentam pouca ou quase nenhuma utilidade pericial. São códigos que informam, por exemplo, se a luz do *airbag* estava acesa ou se a bolsa está devidamente conectada. Mas dados importantes para este trabalho, tais como a aceleração, o giro e a velocidade no momento do impacto são desconhecidos. As montadoras até possuem equipamentos capazes de extrair esse tipo de informação, porém são exclusivos de suas matrizes no país de origem. Além disso, por serem exclusivos e fechados às marcas, não há como checar o grau de confiança dos resultados.

Por falta de alternativa, o *dump* direto da memória foi o método escolhido neste trabalho para leitura dos circuitos de *airbag*. Pela simplicidade selecionamos apenas aqueles que possuíam *EEPROMs* externas ao processador. Existem equipamentos próprios para leitura desse tipo de memória, mas que necessitam "desoldar" o componente da placa

e inseri-lo em uma *protoboard* específica para leitura. Em nosso estudo optou-se por conectar fios diretamente com a ajuda de um "jacaré", aos terminais da *SOIC* (*Small Outline Integrated Circuit*) e, por meio de um microcontrolador Arduino modelo Mega 2560 [9], realizou-se o controle da conexão com memória. Foi desenvolvido um código de acesso, baseado nos protocolos *I2C* ou *SPI*, em que o microcontrolador se comportava como mestre e a *EEPROM* como escravo. Este assunto será melhor abordado no subitem seguinte. Esse código é capaz de gerar um arquivo do tipo texto com o *dump* transcrito em formato hexadecimal. De posse deste arquivo, podemos utilizar outras ferramentas, como o Matlab, para manipular os dados coletados e tentar traduzí-los em algo inteligível.

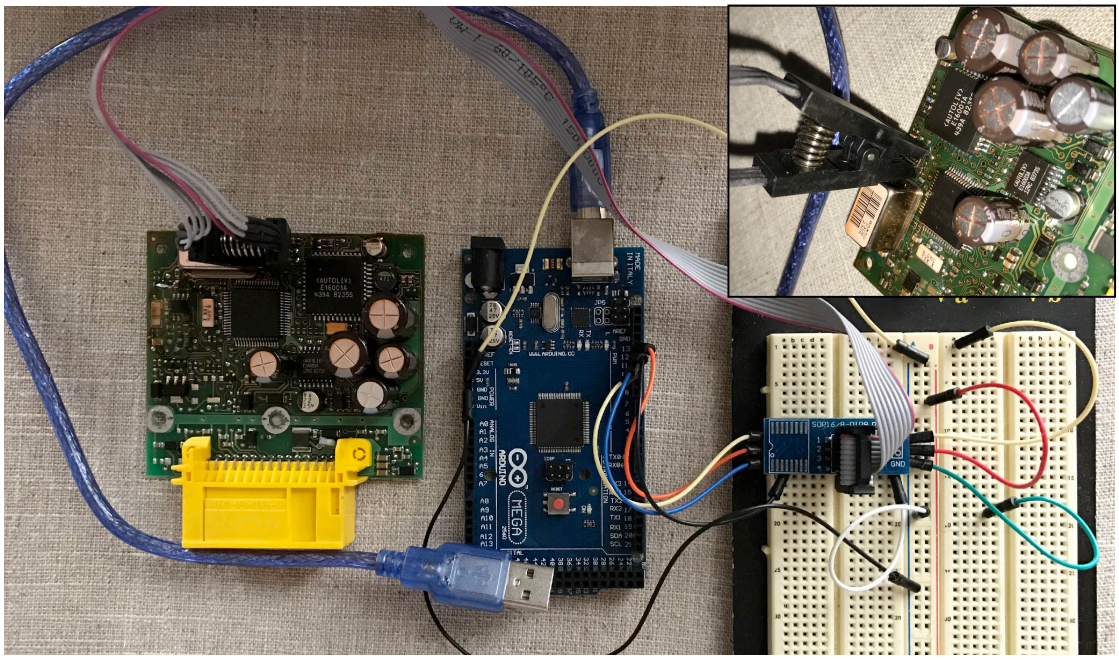


Figura 2.6: Conexão do Arduino com a memória de *EEPROM* do módulo do *Renault/Scenic*.

2.4 BARRAMENTOS SPI E I2C

Aqui iremos fazer uma breve descrição desses dois tipos de protocolos utilizados na conexão com as memórias. São barramentos de comunicação do tipo mestre-escravo, em

que apenas dois dispositivos podem conversar por vez, evitando assim colisões. Ambos são protocolos que trabalham em sincronia com um relógio, definido pelo elemento mestre.

O *SPI* [10] foi desenvolvido pela Motorola e sua comunicação é composta de 3 fios mais um linha de *clock (SLK)*. São elas: a linha de seleção do escravo (*SS – Slave Select*), a linha de transmissão de dados do mestre para o escravo (*MOSI – Master Output Slave Input*) e a linha de transmissão de dados do escravo para o mestre (*MISO – Master Input Slave Output*). O mestre que quer se comunicar com um determinado escravo, seleciona-o colocando sua linha SS (também conhecida como *CS – Chip Select*) em baixo nível. Em seguida, envia suas informações através de sua porta *DO (Data Output*), as quais são recebidas na porta *DI (Data Input*) do escravo selecionado. O mestre recebe informações do escravo pela porta *DI*, a qual está conectada com a *DO* deste dispositivo. Observe a Figura 2.7. Cada dispositivo possui um registrador que envia e recebe um bit ao mesmo tempo, a cada pulso de *clock* (subida ou descida), formando assim uma comunicação em anel. Em outras palavras, o mestre e o escravo enviam e recebem, simultaneamente, um bit. O resultado é que eles "trocam " bits, como ilustra a Figura 2.8.

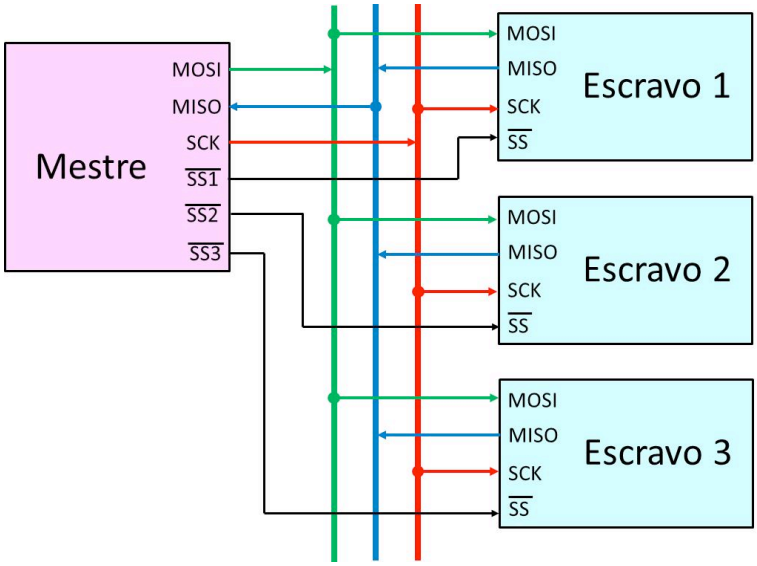


Figura 2.7: Esquema de arquitetura de conexão de um mestre com 3 escravos [10].

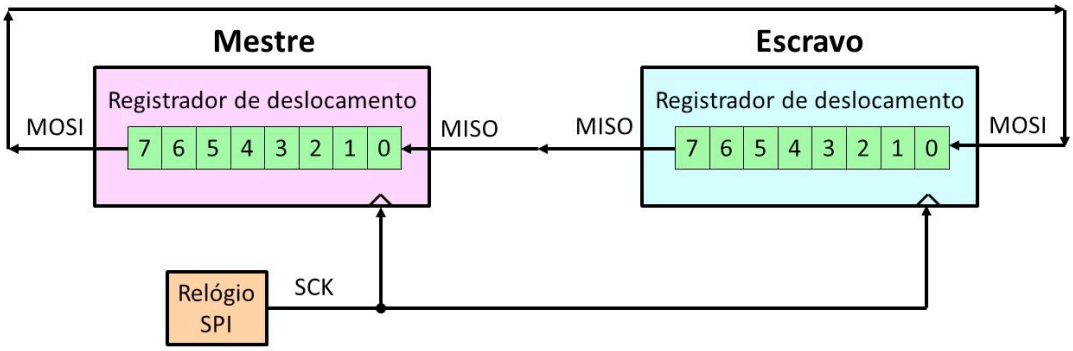


Figura 2.8: Comunicação em anel utilizada no *SPI* [10].

Já o barramento *I2C* [10], conhecido também por *TWI (Two Wired Interface*), foi desenvolvido pela Philips e trabalha apenas com 2 fios: o de dados/endereço (*SDA – Serial Data*) e o *clock* (*SCL – Serial Clock*). O protocolo utiliza circuitos a coletor (ou dreno) aberto e resistores *pull-up* em cada uma dessas linhas. Tanto o mestre como o escravo podem "puxar" as linhas para nível baixo (zero) ou deixá-las em nível alto (um). Assim se torna possível a comunicação de um único mestre com seus escravos. Aqui a escrita e a leitura são feitas pela mesma linha (SDA). O que irá diferenciá-las é o tipo de endereçamento do escravo, o que é feito com o último bit do endereço. Caso seja uma leitura (e consequentemente escrita no mestre) este bit é igual a 1, e o inverso caso seja uma escrita (leitura no mestre). A comunicação inicia-se com uma condição START (S) e termina com uma de STOP (P). Em seguida ao START, vem o endereço com o bit de leitura/escrita na posição final. Toda vez que o receptor da mensagem recebe os dados ele gera um ACK de confirmação ao transmissor. Após o primeiro ACK da comunicação, os dados da memória no presente caso estão prontos para serem enviados/recebidos. Quando o receptor quer parar de receber os dados, ele gera um NACK. Veja as Figuras 2.9 e 2.10.

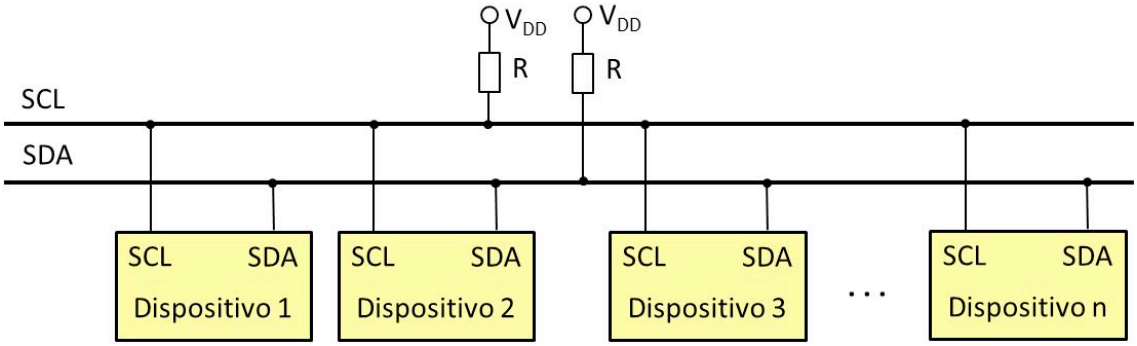
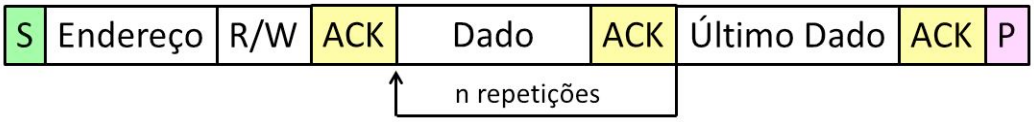


Figura 2.9: Conexão do protocolo I2C [10].



Fig

Figura 2.10: Exemplo de transação no I2C [10].

3 LEITURAS E TESTES COM OS MÓDULOS DE AIRBAG

3.1 ROTINAS DE LEITURA E ESCRITA

Neste trabalho, o estudo foi feito com diversos módulos apreendidos pela equipe de perícia da Seção de Delitos de Trânsito (SDT) da Polícia Civil do Distrito Federal (PCDF) retirados de veículos envolvidos em acidente de trânsito com vítima, que tiveram suas bolsas de *airbag* deflagradas. Houve cuidado para que se mantivesse em sigilo a identidade dos veículos cujos módulos foram analisados. Para operar os circuitos dos módulos, foi necessário retirá-los do carro, abrir sua caixa de proteção[5] e, dentre os componentes observados, identificar principalmente a memória *EEPROM* e o acelerômetro. Foram selecionados para estudo apenas aqueles que tinham a memória não volátil externa ao processador. O acelerômetro não é um componente simples de ser identificado, pois possui

[5] Em algumas das caixas havia parafusos especiais, que só puderam ser abertas com chaves adquiridas com as próprias concessionárias.

vários formatos e muitos dos componentes observados nas placas dos módulos não apresentam manual de forma pública. Ao contrário das EEPROMs, que apesar das várias marcas e modelos, não foi difícil conseguir os *datasheets* para identificação das configurações e forma de acesso de cada memória.

Como a variedade de módulos era grande, optou-se, neste trabalho, por definir um deles como objeto de estudo. Foi escolhido o da marca Autoliv modelo 550 56 90 00, ilustrado na figura abaixo (3.1) e também nas Figuras 2.2, 2.3, 2.4 e 2.6. A triagem foi feita com base, primeiramente, na quantidade de peças disponíveis: havia três equipamentos da Autoliv desse mesmo modelo, sendo dois de Renault/Scenic e um de Renault/Megane (modelo antigo). Como será detalhado adiante, esse modelo foi ainda o que o apresentou a leitura mais simples, com apenas 256 endereços. Além disso, foi um dos poucos em que foi possível identificar, de fato, um acelerômetro embutido no circuito (Figura 2.4). Apesar da identificação do acelerômetro, o Temic BAX50, não foi possível avançar mais do isso. Não se encontrou qualquer tipo de documentação relativa ao seu funcionamento, ou parâmetros de configuração. Inclusive foi feito contato com as empresas responsáveis pela fabricação do componente, mas sem êxito nas respostas.

Já a EEPROM encontrada desses módulos é comercialmente conhecida. Todos os módulos usavam o modelo 93C66, porém de diferentes fabricantes: *Atmel [11], Microchip [12] e Fairchild [13]*. Com auxílio do *datasheet* de cada uma delas, foi possível identificar a pinagem, a alimentação, e os códigos para estabelecer uma comunicação dessa memória. Esse modelo de *EEPROM* utiliza o protocolo *SPI*, com 256 endereços e trabalha com um sistema que define 8 bits para endereçamento e 16 para dados.

Figura 3.1: Módulo de *airbag* Autoliv modelo 550 56 90 00.

A conexão do Arduino com a *EEPROM* foi feita com auxílio de um "jacaré", estabelecendo, portanto, a relação mestre – escravo, conforme ilustrado anteriormente na Figura 2.5. Uma rotina para acesso à memória foi inserida no Arduino Mega capaz de imprimir relatórios com o conteúdo da memória em formato hexadecimal. Apesar de serem de marcas distintas, as diferenças de configuração das *EEPROMs* eram mínimas e, portanto, a mesma rotina foi utilizada para os três módulos.

No modelo 93C66 há 8 pinos distribuídos conforme se mostra na Figura 3.2. Os pinos Vcc e GND são de alimentação e terra respectivamente e, conforme manual, recebem os valores de 5 V e 0 V[6]. A memória permite que se dedique 8 ou 16 bits para dados, a depender da configuração do pino ORG. Para ORG no valor alto, teremos 16 bits de dados e 8, no valor baixo. Observou-se que as memórias questionadas já possuem, em seu

[6] Na verdade aqui está uma diferença entre os tipos de memória do modelo 93C66: o range de voltagem não é o mesmo, mas todos tem o 5 V dentre seus intervalos. Além disso, o Arduino Mega 2560 permite alimentar outros dispositivos com 5 V ou 3,3 V. Por isso optou-se pelo 5 V.

circuito, esse pino conectado ao Vcc, padronizando assim, os 16 bits para dados[7]. O *datasheet* ainda especifica que o pino de *clock* (SK) deve ser configurado para uma frequência de 1 MHz. O pino CS é o *Chip Select* do barramento *SPI*, o qual seleciona o dispositivo enquanto seu nível for baixo. Os pinos DI e DO são os que geram a entrada e saída de bits da memória, respectivamente, e, portanto, são conectados às saída e entrada, respectivamente, do microcontrolador Arduino. Por fim, o pino DC não é utilizado [11].

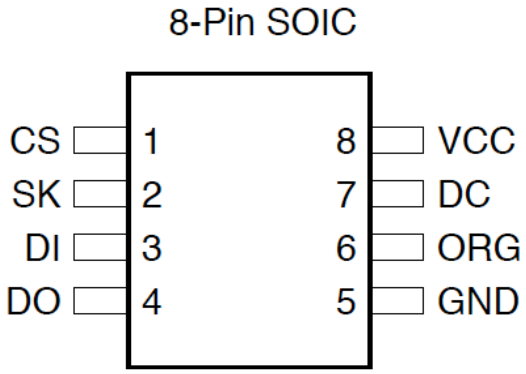


Figura 3.2: Pinagem da *EEPROM* 93C66.

Para se realizar a leitura da memória, o microcontrolador além de posicionar o CS em nível baixo, deve enviar o código 110 (direcionado ao pino DI da memória). O primeiro bit sempre representa o START da comunicação, enquanto que os dois seguintes definem o código de operação (Op Code), nesse caso, uma leitura. Após isso, deve-se enviar o endereço a ser lido (8 bits). A saída (DO) da EEPROM, portanto, envia os dados (16 bits) existentes no endereço selecionado, precedidos de um 0 (*dummy*). A leitura da memória é feita nos pulsos de subida do relógio. Para realizar a leitura completa da memória, criou-se um laço que chama a função de leitura 256 vezes, percorrendo assim todos os endereços da memória [11].

[7] Inclusive caso se force o ORG para 0V, desarma-se o barramento e a conexão deixa de funcionar.

Já a escrita na memória tem procedimento semelhante, mas com um detalhe importante. Para garantir integridade dos dados, toda vez que esse dispositivo é alimentado, ele entra no estado bloqueado para escrita ou apagamento (*EWDS – Erase/Write Disable State*). Portanto para realizar cada escrita (ou para se apagar dados) deve-se, primeiramente, mudar o estado para desbloqueado (*EWEN – Erase/Write Enable State*). Isso é feito enviando o código 100, seguido de 11XXXXXX[8]. Após um pulso de descida do CS, o estado de permissão para escrita é acionado. O modo de escrita é selecionado com o código 101, seguido do endereço a ser escrito (8 bits) e os dados (16 bits), enviados do Arduino para a *EEPROM*. O *datasheet* ainda recomenda um lapso de tempo suficientemente grande antes de iniciar o passo seguinte, para que não haja erro na escrita. Em nosso programa, uma espera de 200 microsegundos foi suficiente. Assim como na leitura, para realizar uma escrita completa da memória, é necessário um laço que chama a função de escrita 256 vezes, sobreescrevendo assim todos os endereços da memória [11].

Foram feitas outras rotinas para acesso a outros modelos de memórias de veículos diferentes. A maioria utilizava o protocolo SPI como meio de comunicação, diferindo entre si basicamente nos códigos de operação (Leitura e Escrita) e na pinagem existente. Como a maior parte das leituras geraram arquivos muito extensos, eles não serão disponibilizados aqui, mas serão debatidos no decorrer do trabalho.

3.2 TESTE DE ACIONAMENTO DO ALGORITMO DE DEFLAGRAÇÃO

Foi realizado um *dump* das memórias de alguns dos módulos apreendidos. O resultado é um arquivo com uma tabela de valores em hexadecimal. Todavia não foi

[8] "X"significa qualquer bit, 1 ou 0.

possível encontrar um padrão nesses valores que permitisse sua interpretação. Assim, foi decidido simular uma situação de acidente para um módulo e comparar os dados da memória, antes e depois do impacto, com um acelerômetro incorporado ao sistema.

O experimento aqui proposto consiste em deixar o módulo cair de uma determinada altura, a partir do repouso, ao ponto de que, ao tocar o chão, a força de impacto seja grande o suficiente para gerar uma aceleração no sentido contrário capaz de ativar o algoritmo de deflagração das bolsas de *airbag*. Para tornar o ambiente controlado e acompanharmos as alterações de aceleração no decorrer do experimento, acoplamos ao módulo um acelerômetro de nosso domínio que registra e grava os valores durante certo intervalo de tempo.

O acelerômetro utilizado foi o MPU 6050 [14]. Esse dispositivo, que também funciona como giroscópio, é comumente utilizado em plataformas microprocessadas para realizar medidas de aceleração (medidas em g) e de giro (medidas em graus/segundo) em cada um dos três eixos X, Y e Z. O MPU se comunica por meio do protocolo I2C. Como nosso interesse é apenas nas medições de aceleração, criamos uma rotina a qual imprime os valores de aceleração segundo os três eixos durante 3 segundos, a cada 2 ms. Com essa tabela de dados medidos, podemos utilizar o Matlab para gerar um gráfico da aceleração *versus* tempo em cada eixo.

É importante frisar que o que nos interessa é o eixo de sentido de deslocamento da queda do módulo. Todos os módulos de *airbag* apresentam em sua carcaça um desenho de seta que aponta para a parte dianteira do carro. Esse é o sentido do Eixo X e, portanto, temos que acoplar nosso MPU com o mesmo sentido, a fim de validar nossos resultados.

Veja na Figura 3.3. uma imagem com dois gráficos do MPU em queda, sendo um ilustrando os três eixos e o outro detalhando apenas o Eixo X (sentido de movimento da queda).

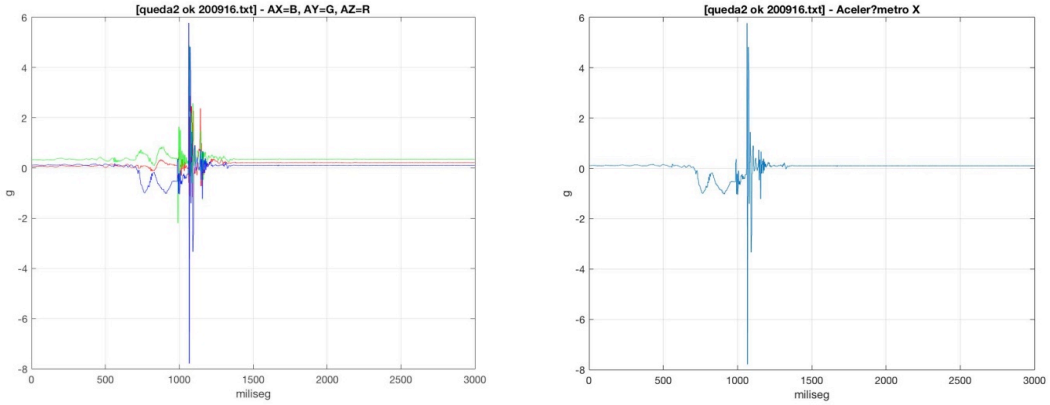


Figura 3.3: O gráfico da esquerda mostra os resultado dos três eixos do MPU sobre um impacto após ser submetido a uma queda livre e o gráfico da direita detalha o Eixo X.

O MPU foi soldado em uma placa de circuito, a qual foi encaixada na parte superior de uma placa Arduino Mega. A fim de manter o sentido de deslocamento do conjunto módulo – MPU - Arduino e evitar assim o máximo de rotações, a queda foi feita no interior de um cano PVC com diâmetro próximo do conjunto. O módulo foi alimentado com uma fonte[9] de 12 V e o Arduino com um cabo USB conectado a um computador. Ambos os cabos deveriam ser grandes o suficiente para permitir que a queda ocorresse sem que eles fossem tracionados ou ficassem presos. Observe na Figura 3.4 o material utilizado no experimento.

[9] Um teste foi feito em oficina para averiguar quais pinos do módulo que devem ser conectados ou aterrados para garantir sua alimentação. Descobriu-se ainda que duas duplas desse pinos deveriam ter uma pequena resistência, pois são os fios que levam sinal a cada uma das bolsas: motorista e passageiro.

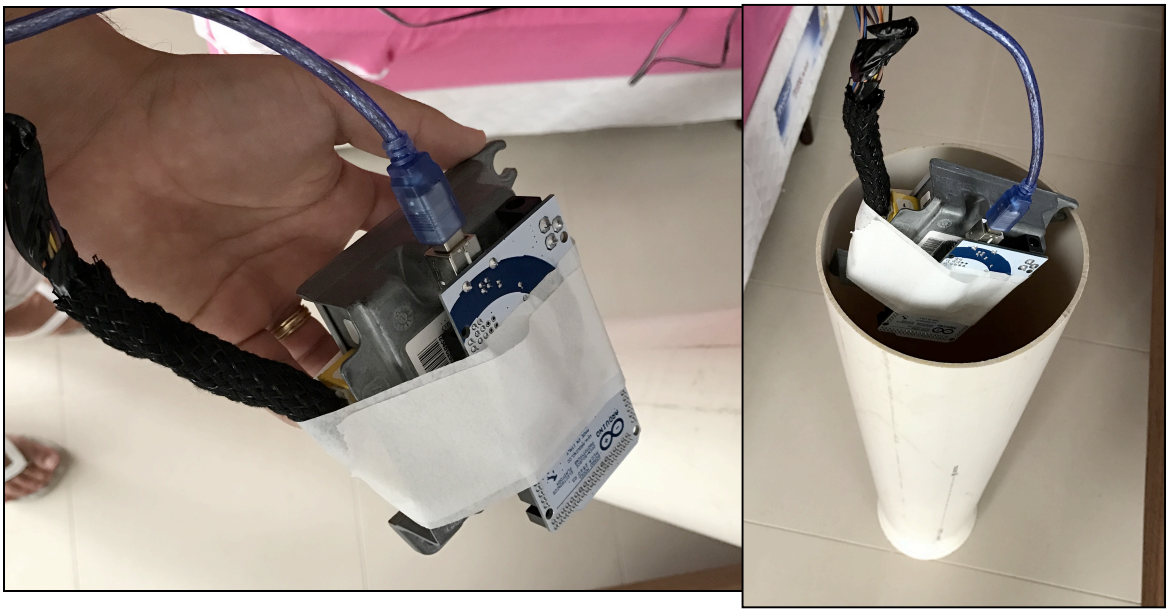


Figura 3.4: Mostra como o experimento foi feito.

Como mencionado anteriormente, o processador do *airbag* quando identifica que sua *EEPROM* possui um código de crash armazenado, ele bloqueia novas gravações na memória. Portanto para realizar cada experimento de queda, era necessário "resetar" a memória para o modo clean. O arquivo clean foi obtido em contato com oficinas especializadas em fazer justamente esse procedimento (limpar módulos "batidos"). Assim o dispositivo pôde ser reutilizado e dispensou a necessidade de se trocar o módulo de *airbag* após cada ensaio. O código clean adquirido está retratado na Figura 3.5.

O teste de queda do módulo com o MPU (e Arduino) acoplado foi repetido várias vezes e alguns dos resultados estão mostrados nas Figuras 3.6, 3.7 e 3.8. A queda foi preparada com cautela de forma a deixar os fios de alimentação (fonte 12 V e cabo USB) livres para que não houvesse trepidação do conjunto. Ao liberar a rotina do MPU (no computador) tínhamos 3 segundos para deixar o conjunto cair. Observou-se que a partir da altura de 0,5 m o algoritmo já era acionado. As leituras revelaram diferenças quando comparadas com o arquivo clean.

```
0000    0700    0400    1100    0A00    1300    0000    0000
0000    0000    0B00    1000    0000    0000    5802    5802
3806    F918    2003    0607    4D04    0330    FF1F    0D26
4203    0300    0008    0006    0606    0600    BA7C    FFE4
0000    00DB    A382    0010    1440    000A    07F6    6107
2CAA    0000    0007    E0C3    C804    0300    0000    0000
0106    9941    4334    5836    345F    342E    5632    3001
0101    0101    4C33    3138    3341    3235    3241    4130
3634    3633    005F    6910    80FF    002A    59FF    FFFF
FFFF    0A00    7380    5A35    3334    3835    3533    35FF
0103    0128    14A0    0000    0099    3C00    0000    FF7D
0F00    2626    2650    4B00    0E0E    0E20    0E07    6464
6464    5F40    FF64    5EB4    09FF    2010    0817    0817
0817    0A30    0710    1010    1010    1028    1964    0E0C
0101    0101    0202    0909    0909    FFFF    0909    0909
FFFF    0B0B    0B0B    FFFF    0B0B    0B0B    FFFF    1010
1012    1208    0303    0303    0303    0119    1908    0A19
1908    0103    0201    0000    0000    0043    FFFF    FFFF
0000    0004    FFFF    FF5A    2600    0000    0000    0000
0000    2800    0000    0000    0000    0000    0000    00FF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    4005
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FF58    3634    4334    4534    2020    2020    20FF    FFFF
0000    0000    0000    0000    0000    0064    6566    6768
696A    6B6C    6D6E    6F70    7172    7374    7576    7778
```

Figura 3.5: Código *clean* hexadecimal do Autoliv 550 56 90 00.

```
0000    0700    0400    1100    0A00    1300    0000    0000
0000    0000    0B00    1000    0000    0000    5802    5802
3806    F918    2003    0607    4D04    0330    FF1F    0D26
4203    0300    0008    0006    0606    0600    BA7C    FFE4
0000    00DB    A382    0010    1440    000A    07F6    6107
2CAA    0000    0007    E0C3    C804    0300    0000    0000
0106    9941    4334    5836    345F    342E    5632    3001
0101    0101    4C33    3138    3341    3235    3241    4130
3634    3633    005F    6910    80FF    002A    59FF    FFFF
FFFF    0A00    7380    5A35    3334    3835    3533    35FF
0103    0128    14A0    0000    0099    3C00    0000    FF7D
0F00    2626    2650    4B00    0E0E    0E20    0E07    6464
6464    5F40    FF64    5EB4    09FF    2010    0817    0817
0817    0A30    0710    1010    1010    1028    1964    0E0C
0101    0101    0202    0909    0909    FFFF    0909    0909
FFFF    0B0B    0B0B    FFFF    0B0B    0B0B    FFFF    1010
1012    1208    0303    0303    0303    0119    1908    0A19
1908    0103    0201    0000    0000    0043    FFFF    FFFF
0000    0004    FFFF    FF5A    2600    0000    0000    0000
0000    2800    0000    0000    0000    0000    0000    00FF
FFFF    FFFF    FFFF    FFFF    FFFF    001A    0600    5005
267A    0005    0704    8000    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FF58    3634    4334    4534    2020    2020    20FF    FFFF
0000    0000    0000    0000    0000    0064    6566    6768
696A    6B6C    6D6E    6F70    7172    7374    7576    7778
```

Figura 3.6: Leitura 1 do código de *crash* hexadecimal do Autoliv 550 56 90 00.

```
0000    0700    0400    1100    0A00    1300    0000    0000
0000    0000    0B00    1000    0000    0000    5802    5802
3806    F918    2003    0607    4D04    0330    FF1F    0D26
4203    0300    0008    0006    0606    0600    BA7C    FFE4
0000    00DB    A382    0010    1440    000A    07F6    6107
2CAA    0000    0007    E0C3    C804    0300    0000    0000
0106    9941    4334    5836    345F    342E    5632    3001
0101    0101    4C33    3138    3341    3235    3241    4130
3634    3633    005F    6910    80FF    002A    59FF    FFFF
FFFF    0A00    7380    5A35    3334    3835    3533    35FF
0103    0128    14A0    0000    0099    3C00    0000    FF7D
0F00    2626    2650    4B00    0E0E    0E20    0E07    6464
6464    5F40    FF64    5EB4    09FF    2010    0817    0817
0817    0A30    0710    1010    1010    1028    1964    0E0C
0101    0101    0202    0909    0909    FFFF    0909    0909
FFFF    0B0B    0B0B    FFFF    0B0B    0B0B    FFFF    1010
1012    1208    0303    0303    0303    0119    1908    0A19
1908    0103    0201    0000    0000    0043    FFFF    FFFF
0000    0004    FFFF    FF5A    269A    D100    0000    0000
0000    2828    2800    0000    0000    0000    0000    00FF
FFFF    FFFF    FFFF    FFFF    FFFF    001A    0800    1007
267A    0005    0502    8000    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FF58    3634    4334    4534    2020    2020    20FF    FFFF
0000    0000    0000    0000    0000    0064    6566    6768
696A    6B6C    6D6E    6F70    7172    7374    7576    7778
```

Figura 3.7: Leitura 2 do código de *crash* hexadecimal do Autoliv 550 56 90 00.

```
0000    0700    0400    1100    0A00    1300    0000    0000
0000    0000    0B00    1000    0000    0000    5802    5802
3806    F918    2003    0607    4D04    0330    FF1F    0D26
4203    0300    0008    0006    0606    0600    BA7C    FFE4
0000    00DB    A382    0010    1440    000A    07F6    6107
2CAA    0000    0007    E0C3    C804    0300    0000    0000
0106    9941    4334    5836    345F    342E    5632    3001
0101    0101    4C33    3138    3341    3235    3241    4130
3634    3633    005F    6910    80FF    002A    59FF    FFFF
FFFF    0A00    7380    5A35    3334    3835    3533    35FF
0103    0128    14A0    0000    0099    3C00    0000    FF7D
0F00    2626    2650    4B00    0E0E    0E20    0E07    6464
6464    5F40    FF64    5EB4    09FF    2010    0817    0817
0817    0A30    0710    1010    1010    1028    1964    0E0C
0101    0101    0202    0909    0909    FFFF    0909    0909
FFFF    0B0B    0B0B    FFFF    0B0B    0B0B    FFFF    1010
1012    1208    0303    0303    0303    0119    1908    0A19
1908    0103    0201    0000    0000    0043    FFFF    FFFF
0000    0004    FFFF    FF5A    2600    0000    0000    0000
0000    2800    0000    0000    0000    0000    0000    00FF
FFFF    FFFF    FFFF    FFFF    FFFF    001A    0500    5005
267A    0005    0504    8000    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF
FF58    3634    4334    4534    2020    2020    20FF    FFFF
0000    0000    0000    0000    0000    0064    6566    6768
696A    6B6C    6D6E    6F70    7172    7374    7576    7778
```

Figura 3.8: Leitura 3 do código de *crash* hexadecimal do Autoliv 550 56 90 00.

3.3 ANÁLISE DOS RESULTADOS

Primeiramente, apenas observando o código *clean* (Figura 3.5), já podemos tecer algumas análises. Analisando-o, levantamos a hipótese de que o código parece ser dividido em três partes bem definidas. A primeira, corresponde às nove primeiras linhas, o que poderia significar o cabeçalho do arquivo. A segunda parte, da linha 10 a 13, possui todos os seus campos em FFFF, que nos leva a suspeita de que essa é a área em que os dados da colisão serão escritos. Por fim, a última parte, as duas últimas linhas poderiam ser uma espécie de fechamento incluindo um verificação como *checksum*. Todavia, não se encontrou elementos que justificassem essa hipótese. Fato é que o mapa hexadecimal apresenta 3 partes bem definidas.

```
>> Comp
-------------Arquivo clean.txt-----------
0000: 0000 0700 0400 1100 0A00 1300 0000 0000 0000 0000 0B00 1000 0000 0000 5802 5802
0001: 3806 F918 2003 0607 4D04 0330 FF1F 0D26 4203 0300 0008 0006 0606 0600 BA7C FFE4
0002: 0000 00DB A382 0010 1440 000A 07F6 6107 2CAA 0000 0007 E0C3 C804 0300 0000 0000
0003: 0106 9941 4334 5836 345F 342E 5632 3001 0101 0101 4C33 3138 3341 3235 3241 4130
0004: 3634 3633 005F 6910 80FF 002A 59FF FFFF FFFF 0A00 7380 5A35 3334 3835 3533 35FF
0005: 0103 0128 14A0 0000 0099 3C00 0000 FF7D 0F00 2626 2650 4B00 0E0E 0E20 0E07 6464
0006: 6464 5F40 FF64 5EB4 09FF 2010 0817 0817 0817 0A30 0710 1010 1010 1028 1964 0E0C
0007: 0101 0101 0202 0909 0909 FFFF 0909 0909 FFFF 0B0B 0B0B FFFF 0B0B 0B0B FFFF 1010
0008: 1012 1208 0303 0303 0303 0119 1908 0A19 1908 0103 0201 0000 0000 0043 FFFF FFFF
0009: 0000 0004 FFFF FF5A 2600 0000 0000 0000 0000 2800 0000 0000 0000 0000 0000 00FF
000A: FFFF FFFF FFFF FFFF FFFF FFFF FFFF 4005 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000B: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000C: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000D: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000E: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FF58 3634 4334 4534 2020 2020 20FF FFFF
000F: 0000 0000 0000 0000 0000 0064 6566 6768 696A 6B6C 6D6E 6F70 7172 7374 7576 7778

-------------Arquivo crash2.txt-----------
0000: 0000 0700 0400 1100 0A00 1300 0000 0000 0000 0000 0B00 1000 0000 0000 5802 5802
0001: 3806 F918 2003 0607 4D04 0330 FF1F 0D26 4203 0300 0008 0006 0606 0600 BA7C FFE4
0002: 0000 00DB A382 0010 1440 000A 07F6 6107 2CAA 0000 0007 E0C3 C804 0300 0000 0000
0003: 0106 9941 4334 5836 345F 342E 5632 3001 0101 0101 4C33 3138 3341 3235 3241 4130
0004: 3634 3633 005F 6910 80FF 002A 59FF FFFF FFFF 0A00 7380 5A35 3334 3835 3533 35FF
0005: 0103 0128 14A0 0000 0099 3C00 0000 FF7D 0F00 2626 2650 4B00 0E0E 0E20 0E07 6464
0006: 6464 5F40 FF64 5EB4 09FF 2010 0817 0817 0817 0A30 0710 1010 1010 1028 1964 0E0C
0007: 0101 0101 0202 0909 0909 FFFF 0909 0909 FFFF 0B0B 0B0B FFFF 0B0B 0B0B FFFF 1010
0008: 1012 1208 0303 0303 0303 0119 1908 0A19 1908 0103 0201 0000 0000 0043 FFFF FFFF
0009: 0000 0004 FFFF FF5A 269A D100 0000 0000 0000 2828 2800 0000 0000 0000 0000 00FF
000A: FFFF FFFF FFFF FFFF FFFF 001A 0800 1007 267A 0005 0502 8000 FFFF FFFF FFFF FFFF
000B: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000C: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000D: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000E: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FF58 3634 4334 4534 2020 2020 20FF FFFF
000F: 0000 0000 0000 0000 0000 0064 6566 6768 696A 6B6C 6D6E 6F70 7172 7374 7576 7778
>>
```

Figura 3.9: Resultado do programa de comparação entre os arquivos do *clean* e *crash*. Em destaque está os campos de mesmo endereços, mas com dados distintos.

Para determinar o que mudou do código clean para o crash, criamos um programa na plataforma Matlab que faz a checagem campo a campo e indica as diferenças. O resultado está na Figura 3.9, representando a comparação dos códigos das Figuras 3.5 e 3.7. Nele os campos em vermelho são os que contêm dados diferentes (valores em hexadecimal). Percebemos que só houve mudança nas linhas 9 e 10. Esse padrão repetiu-se para os demais testes de *crash*.

Comparando ainda dois códigos de crash entre si, percebemos algumas semelhanças (Figura 3.10). Como, por exemplo, na linha 10, em que os dados sempre começam com 001A, a partir da sexta coluna, e terminam com 8000 (na décima segunda coluna). Existe também as décima e décima primeira colunas, em que os valores são sempre 267A e 0005,

```
>> Comp
-------------Arquivo 051016 teste6 martelada ok.txt-----------
0000: 0000 0700 0400 1100 0A00 1300 0000 0000 0000 0B00 1000 0000 0000 5802 5802
0001: 3806 F918 2003 0607 4D04 0330 FF1F 0D26 4203 0300 0008 0006 0606 0600 BA7C FFE4
0002: 0000 00DB A382 0010 1440 000A 07F6 6107 2CAA 0000 0007 E0C3 C804 0300 0000 0000
0003: 0106 9941 4334 5836 345F 342E 5632 3001 0101 0101 4C33 3138 3341 3235 3241 4130
0004: 3634 3633 005F 6910 80FF 002A 59FF FFFF FFFF 0A00 7380 5A35 3334 3835 3533 35FF
0005: 0103 0128 14A0 0000 0099 3C00 0000 FF7D 0F00 2626 2650 4B00 0E0E 0E20 0E07 6464
0006: 6464 5F40 FF64 5EB4 09FF 2010 0817 0817 0817 0A30 0710 1010 1010 1028 1964 0E0C
0007: 0101 0101 0202 0909 0909 FFFF 0909 0909 FFFF 0B0B 0B0B FFFF 0B0B 0B0B FFFF 1010
0008: 1012 1208 0303 0303 0303 0119 1908 0A19 1908 0103 0201 0000 0000 0043 FFFF FFFF
0009: 0000 0004 FFFF FF5A 2600 0000 0000 0000 0000 2800 0000 0000 0000 0000 0000 00FF
000A: FFFF FFFF FFFF FFFF FFFF 001A 0500 5005 267A 0005 0504 8000 FFFF FFFF FFFF FFFF
000B: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000C: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000D: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000E: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FF58 3634 4334 4534 2020 2020 20FF FFFF
000F: 0000 0000 0000 0000 0000 0064 6566 6768 696A 6B6C 6D6E 6F70 7172 7374 7576 7778

-------------Arquivo crash2.txt-----------
0000: 0000 0700 0400 1100 0A00 1300 0000 0000 0000 0B00 1000 0000 0000 5802 5802
0001: 3806 F918 2003 0607 4D04 0330 FF1F 0D26 4203 0300 0008 0006 0606 0600 BA7C FFE4
0002: 0000 00DB A382 0010 1440 000A 07F6 6107 2CAA 0000 0007 E0C3 C804 0300 0000 0000
0003: 0106 9941 4334 5836 345F 342E 5632 3001 0101 0101 4C33 3138 3341 3235 3241 4130
0004: 3634 3633 005F 6910 80FF 002A 59FF FFFF FFFF 0A00 7380 5A35 3334 3835 3533 35FF
0005: 0103 0128 14A0 0000 0099 3C00 0000 FF7D 0F00 2626 2650 4B00 0E0E 0E20 0E07 6464
0006: 6464 5F40 FF64 5EB4 09FF 2010 0817 0817 0817 0A30 0710 1010 1010 1028 1964 0E0C
0007: 0101 0101 0202 0909 0909 FFFF 0909 0909 FFFF 0B0B 0B0B FFFF 0B0B 0B0B FFFF 1010
0008: 1012 1208 0303 0303 0303 0119 1908 0A19 1908 0103 0201 0000 0000 0043 FFFF FFFF
0009: 0000 0004 FFFF FF5A 269A D100 0000 0000 0000 2828 2800 0000 0000 0000 0000 00FF
000A: FFFF FFFF FFFF FFFF FFFF 001A 0800 1007 267A 0005 0502 8000 FFFF FFFF FFFF FFFF
000B: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000C: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000D: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
000E: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FF58 3634 4334 4534 2020 2020 20FF FFFF
000F: 0000 0000 0000 0000 0000 0064 6566 6768 696A 6B6C 6D6E 6F70 7172 7374 7576 7778
>>
```

Figura 3.10: Resultado do programa de comparação entre dois arquivos de *crash (Figuras 3.5 e 3.6)*. Em destaque estão os campos de mesmo endereços, mas com dados distintos.

respectivamente. Isso levanta a hipótese de que esses dados devam estar relacionados à colisão de uma forma geral. Percebemos, portanto, que basicamente apenas sete endereços estão relacionados às colisões de forma específica.

Apesar de todos os testes de queda serem feitos com o acompanhamento do acelerômetro MPU, não foi possível encontrar qualquer correlação entre os valores medidos pelo acelerômetro e os valores lidos a partir dos arquivos de *crash*. Isso porque, como pode ser observado, a mudança de bytes de uma colisão para outra foi de apenas sete campos. Poderíamos até traçar um gráfico com esses novos dados, mas como a quantidade de amostras diferentes em cada um é insignificante, fica inviável testar a hipótese de que esses poucos dados trariam as informações de aceleração do acidente.

Um outro teste foi feito traduzindo os bytes de um dos códigos de crash para o padrão ASCII [15]. Observando o resultado, o qual está na Figura 3.11, percebemos que não se desprende muita informação útil. Uma sequência de letras de “d” a x”” parece ser o fechamento do arquivo e as letras maiúsculas na terceira linha podem estar relacionadas ao modelo do módulo ou ao NIV do veículo. A tradução dos outros códigos

```
XX8ù M0ÿ
&Bº|ÿäÛ£@
öa,ªàÃÈAC4X64_4.V20L3183A252AA06463_iÿ*Yÿÿÿÿÿ
sZ53485535ÿ( <ÿ}&&&PK dddd_@ÿd^´        ÿ
0(d
                                ÿÿ                              ÿÿ


ÿÿ


ÿÿ
CÿÿÿÿÿÿÿZ&Ñ(((ÿÿÿÿÿÿÿÿÿÿÿ&zÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿX64C4E4
ÿÿÿdefghijklmnopqrstuvwx
```

Figura 3.11: Código ASCII do *Scenic (crash)*.

obtidos, inclusive do arquivo *clean* foi praticamente idêntica, mudando apenas um ou outro caractere. Apesar dessas incertezas, em outros tipos de módulo, foi possível identificar o NIV ou o número de modelo do módulo de forma clara. Isso ocorreu, por exemplo, com a leituras de um *Peugeot/306* (*EEPROM* 95080) e um *Hyundai/I30 (EEPROM 25640)*.

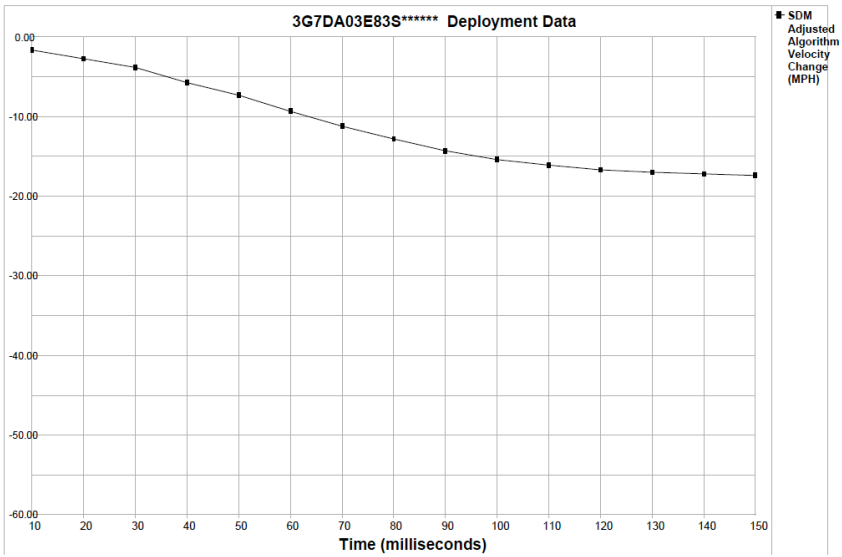
A conclusão do experimento é de que esse tipo de módulo não grava valores de aceleração em sua memória EEPROM. Possivelmente o único dado gravado de um acidente é de que as bolsas e pré-tensionadores foram acionados. Essa análise pode ser ampliada para outras marcas, pois foi confirmada, de maneira informal, por pessoal especializado de outras montadoras no Brasil. Surgiu daí a ideia de se propor um sistema que capturasse as diversas informações de uma acidente e depois permitisse a reconstrução de toda sua dinâmica. Tal sistema teria uma utilidade óbvia em viaturas policiais, que são expostas a um grande risco de acidentes e também como um módulo opcional que o proprietário poderia adicionar em seu veículo particular. Esse sistema é foco do capítulo seguinte.

4 PROPOSTA DE EMPREGO DE PLATAFORMA INERCIAL EM VEÍCULOS

Como citado em diversos momentos no decorrer do trabalho, vários são os empecilhos para se extrair informação significativa dos atuais equipamentos de *airbag*. Os scanners internacionais que realizam este tipo de serviço são capazes de gerar relatórios com as informações da colisão, sendo a aceleração (e consequentemente a velocidade) a mais importante delas. Consultando alguns desses relatórios, observou-se que, dentre os dados gerados está o gráfico de diferença de velocidade (mph) pelo tempo (ms). Sabemos

da área de Cálculo que a diferença de velocidade é a integral da aceleração em um intervalo de tempo. Portanto temos que esse tipo de scanner já converte os valores armazenados no módulo de aceleração em termos de diferença de velocidade[10]. Na Figura 4.1, temos um exemplo de parte de um relatório da Bosch de um GM/Astek (veículo mexicano) que mostra o gráfico de diferença de velocidade durante 150 milisegundos [16]. Para esse tipo de módulo, o padrão é que os 50 ms sejam o período antes da colisão e os 100 ms finais, pós-colisão. Portanto é razoável pensar que existem módulos que possuem em seu código de deflagração pontos de um gráfico de aceleração.

BOSCH CDR CRASH DATA RETRIEVAL



Time (milliseconds)	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150
Adjusted Algorithm Velocity Change	-1.56	-2.66	-3.75	-5.73	-7.27	-9.24	-11.22	-12.75	-14.29	-15.39	-16.04	-16.70	-16.92	-17.14	-17.36

Figura 4.1. Mostra gráfico diferença de velocidade pelo tempo em parte de um relatório de *CDR* da Bosch.

[10] De forma geral o dado da diferença de velocidade não é um parâmetro útil para a perícia, pois esse dado analisado de forma isolada não agrega valor. Porém como, geralmente, após uma colisão, a velocidade final do veículo é zero, o resultado da integral da aceleração medida, para esse caso, no tempo, será a própria velocidade absoluta.

Pensando nisso este trabalho propõe um novo equipamento para ser acoplado aos veículos e que desempenhe a mesma função dos módulos tradicionais, mas com a vantagem de ser de fácil manuseio e acesso, além de baixo custo e principalmente com grande resolução. Com esse novo dispositivo, a leitura das informações no momento da colisão é feita de maneira bem simples e eficiente e, além das óbvias vantagens forenses, serve como ponto de partida no estudo dos módulos de *airbag* convencionais (pelo menos daqueles que gravam dados de colisão). Esse equipamento seria extremamente útil para perícia, se fosse acoplado, por exemplo, nas viaturas policiais, que, com frequência, envolvem-se em acidentes graves.

Buscou-se simplicidade e funcionalidade na construção do protótipo para os ensaios. Por isso foi selecionada uma placa microcontroladora Arduino Mega 2560 [9], o sensor utilizado anteriormente, MPU-6050 (acelerômetro e giroscópio) [14] e uma memória externa EEPROM, modelo 24LC1025, de 128 KB [17] e um Relógio de Tempo Real (Real Time Clock, RTC) que usa o chip DS-1307 [18]. O preço ficou baixo, pois o Arduino Mega custa aproximadamente 70 reais, enquanto o MPU, por volta de 20 reais, a memória 20 reais e o RTC em torno de 10 reais[11]. Assim, o custo estimado do protótipo é de R$ 120,00.

[11] Quando da publicação deste trabalho.

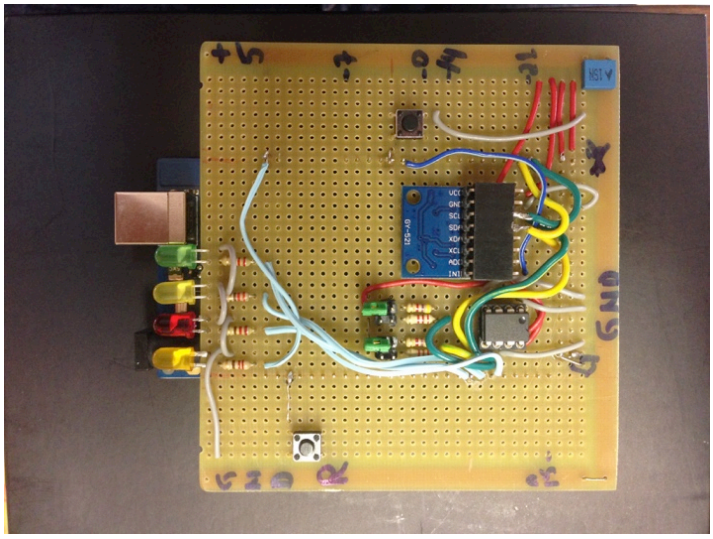


Figura 4.2. Protótipo para avaliação do conceito.

O sistema faz uso do barramento I2C, sendo que o Arduino Mega é o mestre deste barramento e os demais elementos, a saber, o MPU, a EEPROM e o Relógio são os escravos. O projeto prevê alimentação de 12 V (mesma existente em veículos) e com dimensões máximas de 3 x 12 x 12 cm, como mostrado na Figura 4.2. É importante frisar que, assim como nos módulos de *airbag* convencionais, a disposição do equipamento proposto no veículo deve ser referenciada com o sentido dos eixos do MPU. Portanto, a sugestão é que o eixo X do MPU seja colocado sobre o eixo longitudinal do veículo e apontado para a direção de deslocamento, ou seja, para sua frente.

O esquema elétrico do protótipo é apresentado na Figura 4.3. É fácil a identificação dos três periféricos básicos:

- MPU-6050: acelerômetro e giroscópio;

- RTC DS1307: relógio de tempo real e

- 24LC1025: EEPROM de 128 Kbytes

É preciso notar que o Arduino Mega se comunica com todos os dispositivos por meio de um barramento serial a dois fios (*SDA* e *SCL*) que segue o protocolo *I2C*. Nessas

linhas *SDA* e *SCL*, há a presença de resistores que funcionam como *pull-ups*, exigidos pelo protocolo [8]. Existem duas opções de resistores de *pull-up* que devem ser selecionadas pelos *jumpers* JP1 e JP2, de acordo com a velocidade do barramento.

Pelas figuras (4.2 3 4.3) ainda observamos que no dispositivo proposto há a presença de 4 *leds* e 2 botões. Os *leds* servem para sinalização ao usuário. Por enquanto, a proposta é a de que o *led* verde (D3) fique aceso enquanto o sistema está operando,

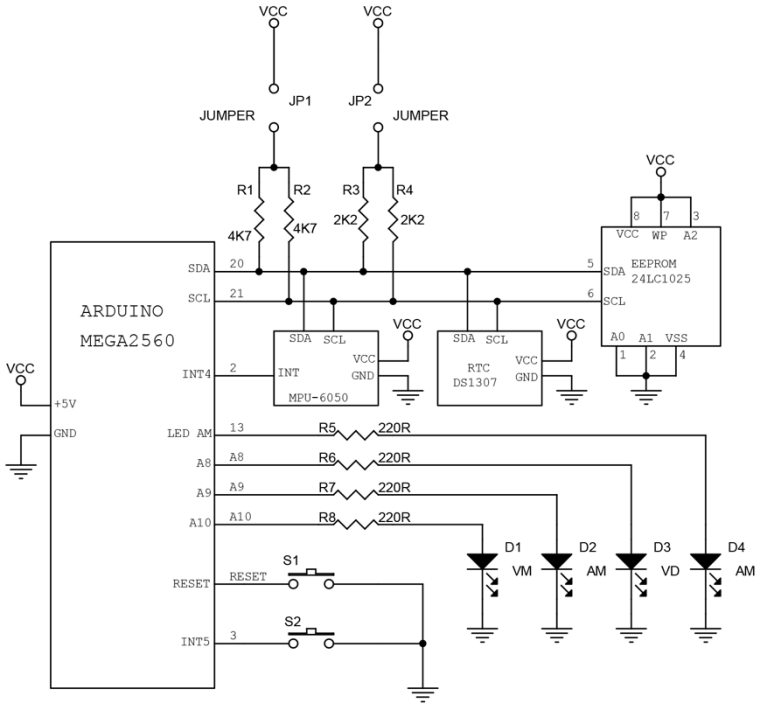


Figura 4.3. Esquema de comunicação do projeto.

o amarelo (D2), quando algum erro for detectado e o vermelho (D1) para indicar que já ocorreu uma colisão. O *led* D4 é usado para sinalização extra e teve grande emprego durante a fase de desenvolvimento. O botão S1 é o de *Reset* e o S2 também é voltado para o desenvolvimento do protótipo.

O *MPU*-6050, como já descrito, é um sensor do tipo *MEMS (Micro Electro Mechanical System*) capaz de realizar leituras de aceleração e giroscópio em cada um dos três eixos (X, Y e Z). Possui conversor analógico-digital de 16 bits para cada eixo, possibilitando assim grande precisão nos resultados de cada canal. Cada leitura completa do MPU possui 12 bytes, ou seja, 2 bytes (HIGH e LOW) para cada eixo de aceleração e giro. Os quatro intervalos de operação (escalas) em que ele trabalha vão de ±2 g, ± 4 g, ± 8 g e ± 16 g para a aceleração e ± 250 °/s, ± 500 °/s, ± 1000 °/s e ± 2000 °/s para o giroscópio.

4.1 LÓGICA DE REGISTRO DOS DADOS NA MEMÓRIA

Para construir tal sistema de registro, era necessária uma memória não-volátil onde se pudessem escrever continuamente os dados de aceleração e rotação. É claro que os dados mais recentes vão sobreescrevendo os mais antigos. Por ocasião de um eventual acidente, grava-se ainda o correspondente à metade da memória e se encerra a gravação. Assim, o sistema proposto é capaz de fornecer dados antes e depois do acidente.

Foi selecionada a memória 24LC1025 que é uma *EEPROM* com capacidade de 128 Kbytes (131.072 bytes). Ela trabalha com páginas de 128 bytes, sendo que gasta, no máximo, 5 ms para gravar cada página. Assim, o total de páginas é de 1024. A primeira página, denominada de página 0, como será explicado mais adiante, foi reservada para gravar as informações de configuração do sistema.

É importante esclarecer a questão da temporização, pois se trabalha com uma taxa de dados relativamente alta. A porta *I2C* foi configurada para operar na taxa limite do Arduino que é de 400 kHz (*SCL*). Nessa velocidade, gastam-se, aproximadamente 3,3 ms para se transmitir 128 bytes para a memória. Terminada a transmissão, a memória inicia a fase de gravação da página que dura, no máximo, 5 ms. Esta fase não demanda atenção do

processador, por isso ele pode se voltar para as leituras de dados do *MPU*. Tem-se então, sob a ótica da memória, 3,3 ms (transmissão) mais 5,0 ms (gravação). Isto resulta em 8,3 ms. Podemos arredondar para 10 ms, para assim garantir uma boa margem de segurança. Em 10 ms, o *MPU*-6050, que opera na taxa de 500 leituras por segundo, gera apenas 120 bytes (menos que uma página, que tem 128 bytes). Assim, se conclui que o sistema proposto atende à taxa especificada.

Excluindo-se a página 0, que é a de configuração, sobram 130.944 bytes para gravar as leituras do *MPU* (12 bytes por leitura). O interessante é que 130.944 é múltiplo de 12, o que simplifica muito ao projeto. Portanto, temos que a memória é capaz de guardar 10.912 (130.944/12) leituras de aceleração e giro.

Para favorecer a resolução, o *MPU* foi colocado para trabalhar com a máxima taxa de amostragem, que é de uma leitura completa (3 eixos de aceleração e 3 eixos de rotação) a cada 2 ms. Como visto no parágrafo anterior, a memória consegue armazenar 10.912 leituras completas, o que corresponde então a um intervalo de 21,824 segundos.

A lógica proposta é que a *EEPROM* seja gravada constantemente, sobreescrevendo a porção mais antiga quando seu limite é atingido. Porém, quando for detectado um acidente, prossegue-se a gravação por mais 10,91 segundos e depois se trava o sistema, permitindo apenas sua leitura ou o comando de apagar a memória. Assim, ficam gravados os dados correspondentes aos 10,91 segundos que antecederam ao acidente e aos 10,91 segundos que sucederam o acidente. Como mencionado anteriormente é característica da *EEPROM* manter a integridade de seus dados quando a alimentação é interrompida.

A caracterização de um acidente é dada pela ultrapassagem de valores limites para aceleração e rotação. Durante a fase de configuração, é possível especificar um limite de aceleração para cada um dos eixos do acelerômetro e também um limite de rotação para

cada um dos eixos do giroscópio. Quando pelo menos um desses limites é ultrapassado, o sistema reconhece um acidente e inicia a gravação por mais 10,91 segundos. Em seguida atualiza a página zero, para indicar qual sensor identificou o acidente e o ponteiro para os dados do instante do acidente.

4.2 DADOS DE CONFIGURAÇÃO DO SISTEMA

Como já afirmado, a página zero (de tamanho 128 bytes) foi reservada para armazenar as configurações do sistema e os dados que caracterizaram a colisão (acidente). Por enquanto, foram empregadas apenas 45 posições, listadas na Tabela 1. Existe, portanto, espaço para ampliações futuras. O byte de endereço zero usa as letras 'T' ou 'F' para indicar se o dispositivo já foi acidentado. O byte seguinte, da mesma forma, indica se o MPU passou no auto-teste (*self-test*) [19]. As posições de 2 a 15 trazem os parâmetros da calibração. As posições 16 e 17 indicam as escalas usadas durante a operação do dispositivo. Os limiares a serem usados para caracterizar um acidente estão indicados nas posições de 18 a 29. O ponteiro para a leitura que detectou o acidente está nas posições 30, 31 e 32. As 6 posições seguintes (33 a 38) indicam qual sensor detectou o acidente. Finalmente, as próximas 6 posições trazem as informações da data e hora do acidente. As demais posições estão vazias e são preenchidas com o byte 0xFF.

Tabela 1. Lista de alguns comandos da Página 0 do protótipo.

Bytes	Tipo	Descrição
0	Char	(T) TRUE → pronta para uso, (F) FALSE → acidentada
1	Char	(T) TRUE → passou no self test, (F) FALSE → falhou no self test
2	Byte	Acelerômetro, escala usada na calibração
3	Byte	Giroscópio, escala usada na calibração
4, 5	Int	Eixo X, resultado da calibração do acelerômetro
6, 7	Int	Eixo Y, resultado da calibração do acelerômetro

8, 9	Int	Eixo Z, resultado da calibração do acelerômetro
10, 11	Int	Eixo X, resultado da calibração do giroscópio
12, 13	Int	Eixo Y, resultado da calibração do giroscópio
14, 15	Int	Eixo Z, resultado da calibração do giroscópio
16	Byte	Acelerômetro, escala usada na operação
17	Byte	Giroscópio, escala usada na operação
18, 19	Int	Limiar de disparo, eixo X, acelerômetro (valor absoluto)
20, 21	Int	Limiar de disparo, eixo Y, acelerômetro (valor absoluto)
22, 23	Int	Limiar de disparo, eixo Z, acelerômetro (valor absoluto)
24, 25	Int	Limiar de disparo, eixo X, giroscópio (valor absoluto)
26, 27	Int	Limiar de disparo, eixo Y, giroscópio (valor absoluto)
28, 29	Int	Limiar de disparo, eixo Z, giroscópio (valor absoluto)
30, 31, 32	Long	Ponteiro para o bloco de 6 leituras onde ocorreu o disparo
33	Char	Acel Eixo X: (T) TRUE → disparou, (F) FALSE → não disparou
34	Char	Acel Eixo Y: (T) TRUE → disparou, (F) FALSE → não disparou
35	Char	Acel Eixo Z: (T) TRUE → disparou, (F) FALSE → não disparou
36	Char	Giro Eixo X: (T) TRUE → disparou, (F) FALSE → não disparou
37	Char	Giro Eixo Y: (T) TRUE → disparou, (F) FALSE → não disparou
38	Char	Giro Eixo Z: (T) TRUE → disparou, (F) FALSE → não disparou
39	Char	Ano (0 → 99)
40	Char	Mês (1 → 12)
41	Char	Dia (1 → 31)
42	Char	Hora (0 → 23)
43	Char	Minuto (0 → 59)
44	Char	Segundo (0 → 59)
45	Char	0xFF
...	Char	0xFF
127	Char	0xFF

4.3 COMUNICAÇÃO COM O DISPOSITIVO PROPOSTO

Para facilitar seu acesso, o sistema proposto é acessado via porta USB. Ele se comporta como um terminal serial que responde a comandos enviados pelo computador. Incialmente, alguns comandos simples foram propostos. É claro que o conjunto de comandos será expandido por ocasião de seu emprego profissional. Cada comando é

caracterizado por uma letra, que pode ser ou não seguida por parâmetros. A Tabela 2 apresenta apenas alguns dos comandos básicos.

Tabela 2. Lista de alguns comandos de interface com o usuário.

Cmdo.	Param.	Descrição
?	-	Enviar configuração. (página 0)
I	-	Inverter estado acidentado (T → F ou de F→T)
R	-	Zerar toda a EEPROM
L	-	Ler toda a memória
X	n	Limiar acelerômetro eixo X
Y	n	Limiar acelerômetro eixo Y
Z	n	Limiar acelerômetro eixo Z
x	n	Limiar giroscópio eixo X
y	n	Limiar giroscópio eixo Y
z	n	Limiar giroscópio eixo Z

Apresentaremos alguns exemplos. Quando recebe o comando "?", o dispositivo envia os dados da página zero, aquela que armazena todas as configurações. O comando "R" indica ao sistema para apagar toda a EEPROM (gravar 0xFF). O comando "I" indica para inverter o estado de acidentado (T) para não acidentado (F) ou ao contrário. Como resposta ao comando "L", o dispositivo envia todo o conteúdo da memória e tem como resposta uma tabela com seis colunas de valores de aceleração e giro, nos eixos X, Y e Z, respectivamente. Os outros seis comandos permitem especificar os limites para caracterizar acidentes.

Apenas este pequeno conjunto de comandos já permite o emprego do dispositivo proposto e a leitura dos dados para posterior análise. As Figuras 4.4 e 4.5 mostram exemplos dos comandos "?" e "L", respectivamente.

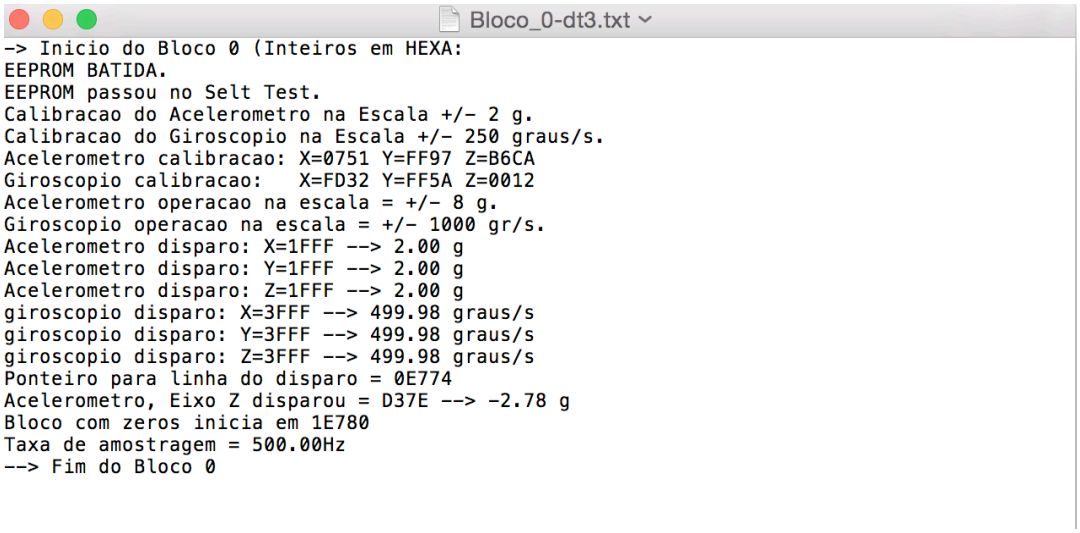


Figura 4.4. Exemplo de resposta da Página 0.

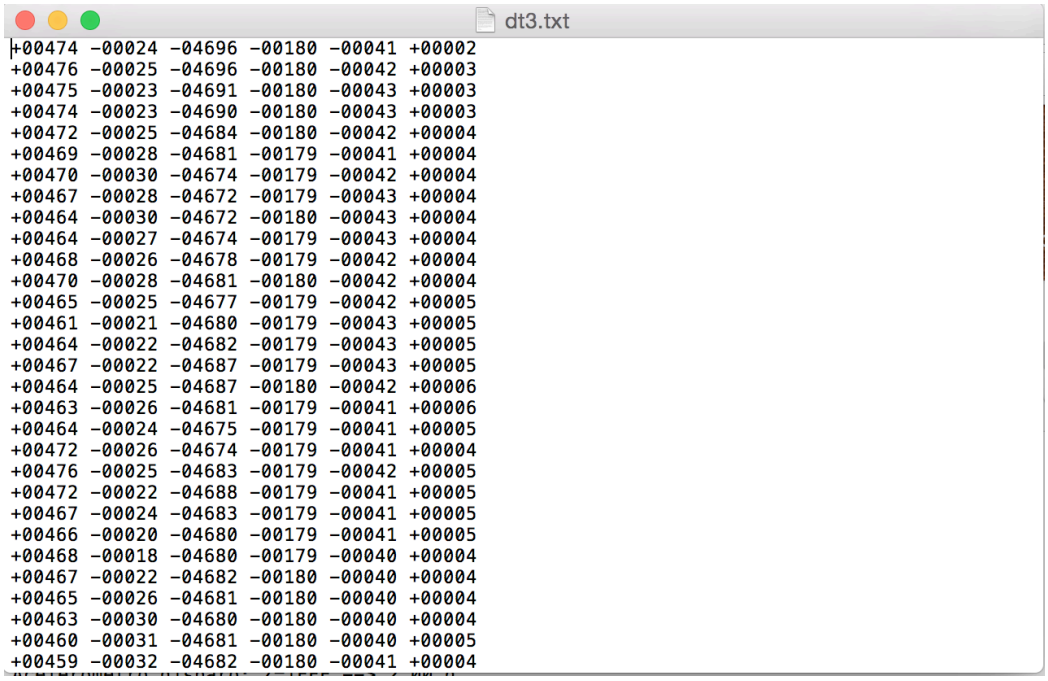


Figura 4.5. Resultado da leitura da memória do equipamento. A resposta são 6 colunas dos 3 eixos de aceleração e giro respectivamente. A resposta não contém descontados os valores de *offset*.

4.4 SEGURANÇA NO ARMAZENAMENTO DOS DADOS

Os dados de aceleração e velocidade são gravados sequencialmente na memória *EEPROM*. Cada valor de leitura tem 16 bits, ou seja, é composta por 2 bytes, denominados *High* e *Low*. Assim, uma leitura completa que é composta por 3 eixos do acelerômetro e 3 eixos do giroscópio, precisa de 12 bytes, como mostrado na Tabela 3. Todavia, a gravação

em sequência é preocupante, pois se a gravação na *EEPROM* perder 1 único byte (como ocorreu em alguns ensaios), todos os demais dados ficarão deslocados e serão interpretados de forma errada.

Tabela 3. Forma como os 12 bytes do MPU são armazenados.

Acelerômetro						Giroscópio					
Eixo X		Eixo Y		Eixo Z		Eixo X		Eixo Y		Eixo Z	
H	L	H	L	H	L	H	L	H	L	H	L

Para fornecer segurança no armazenamento é proposto o seguinte método de marcação dos dados: Reduzir a precisão de 16 bits para 14 bits, e usar esses dois bits extras para marcar leituras do acelerômetro e do giroscópio. A palavra que agora é de 14 bits é quebrada em duas palavras de 7 bits e, para completar o byte, o bit mais à direita (o menos significativo) é feito igual a 1 no caso dos dados do acelerômetro e igual a 0, no caso dos dados do giroscópio. Desta forma, na memória deverão estar grupos de 12 bytes, onde os 6 primeiros são ímpares (último bit igual a 1) e os 6 últimos são pares (último bit igual a 0). Enquanto este padrão for repetido, os dados são confiáveis. Caso haja algum erro, devem-se recusar os dados errados e avançar para o próximo bloco confiável. A sugestão é interpolar os dados que foram perdidos.

4.5 LÓGICA PARA LEITURA E GRAVAÇÃO DE DADOS

Como já foi citado, a memória *EEPROM* gasta 5 ms para gravar um byte. Porém ela oferece um modo de gravação de página que gasta, no máximo, os mesmos 5 ms para gravar 128 bytes. Assim, por premência de tempo, decidiu-se realizar a gravação por páginas. É preciso comentar que o número 128 não é múltiplo de 12, com isso, leituras consecutivas poderão ficar em páginas diferentes. Outro ponto importante é que a leitura

dos dados do MPU e a gravação na *EEPROM* não podem ter uma sincronização fina. O tempo de gravação varia muito. O manual especifica apenas o limite máximo que é de 5 ms [17].

Por isso, a solução foi trabalhar com dois buffers de 128 bytes (tamanho da página) cada um. Assim, enquanto um buffer está sendo preenchido com as leituras, o outro está sendo gravado. O programa trabalha com dois estados: Estado 1 e Estado 2. No Estado 1, grava-se o primeiro buffer enquanto preenche o segundo. No Estado 2, que é o oposto, grava o segundo buffer enquanto preenche o primeiro. Isto está mostrado na Figura 4.6. Para facilitar a partida do programa, foi criado o Estado 0, que preenche o primeiro buffer com os parâmetros de configuração e desvia para o Estado 1. O ponteiro do endereço a ser gravado vai de 128 a 130.944 e volta a 128. Assim, preserva a primeira página da memória, que é a de configuração. Ao final da gravação, após um acidente, a página 0 é atualizada com os dados relativos ao acidente.

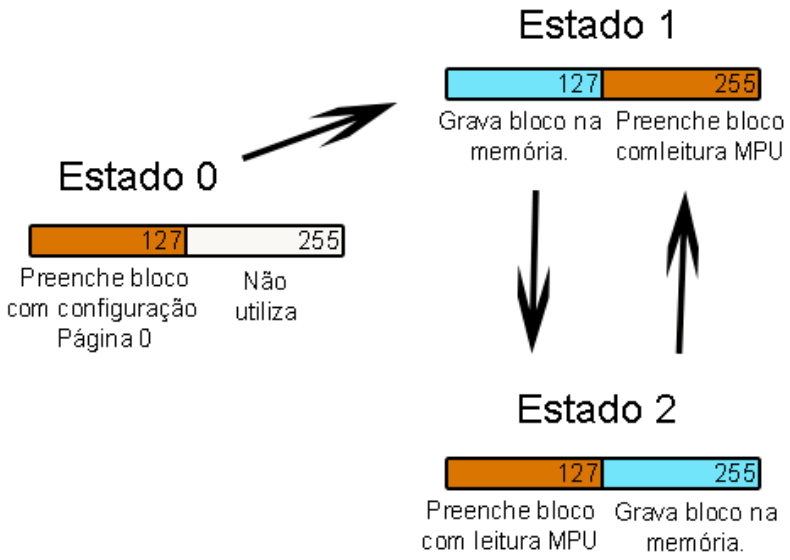


Figura 4.6. Diagrama de estados para leitura do MPU e escrita na memória EEPROM.

Ao final da gravação também é escrito um bloco de valores iguais a zero. Isso é feito no intuito de facilitar a visualização do início das medições, que se encontra após esse bloco, pois a gravação é feita de forma cíclica. Em outras palavras, após esse bloco de zeros temos os dados mais antigos coletados pelo sensor e, antes do bloco, temos os mais recentes.

4.6 GARANTIA DE ALIMENTAÇÃO

O sistema foi previsto para empregar os 12 V da alimentação veicular. Porém, é comum que a alimentação do veículo seja interrompida durante um acidente. Para solucionar este problema, não é difícil implementar uma bateria de *back-up* de 9 V. Essa bateria só alimentaria o sistema na falta da alimentação principal. Ela seria capaz de sustentar o sistema por algumas horas, o que é mais que suficiente para finalizar a gravação dos dados do acidente.

Esse recurso de *back-up* de alimentação pode ser mais uma informação útil para a perícia, pois pode indicar o instante exato em que o sistema de alimentação do veículo entrou em colapso.

4.7 MANEJO DOS RESULTADOS

O sistema proposto se comunica através de comandos e respostas enviados pela porta USB. O comando “L” permite que se obtenha um mapa hexadecimal de toda a memória. De posse deste mapa, é possível empregar diversos programas para analisar os dados do acidente. Por exemplo, com o Matlab, é possível traçar gráficos com os valores instantâneos de aceleração e de velocidade angular. Com emprego da integral discreta, traçam-se gráficos de velocidade e posição angular.

Considerando que a colisão é exatamente o ponto de disparo do sistema e, ainda, considerando que o veículo não mais se movimentou após atingir sua posição de repouso final, teremos precisamente as velocidades do veículo segundo os três eixos, nos instantes anteriores e posteriores à colisão.

Como a quantidade de valores adquiridos é considerável (por volta de 10.900 amostras) e detalhada, é possível, ainda, montar uma simulação gráfica do acidente que mostra de forma 3D o movimento do veículo, a partir da plataforma *Processing*. Para tanto, é necessário um pré-processamento para gerar um arquivo texto (*.txt*) com a tabela de dados de aceleração e giro segundo cada eixo. Nessa simulação podemos visualizar o movimento de giro do veículo nos momentos que antecedem e sucedem a colisão e, ainda, os valores de aceleração alterando-se conforme esse movimento.

Os detalhes das simulações gráficas estão descritos no Capítulo 5.

5 SIMULAÇÃO 3D DOS RESULTADOS DO PROTÓTIPO

Para auxiliar na reconstrução da dinâmica do acidente, é possível, em posse dos dados gerados pelo protótipo descrito no capítulo anterior, visualizar as informações de giro e aceleração de forma gráfica. Com auxílio do *Matlab*, traçamos os gráficos de cada eixo, observados na Figura 5.1. Para atingir esse objetivo foi necessário construir um programa capaz de fazer os devidos descontos de *offset* (calibração) para cada valor medido pelo MPU. Logo, o programa tem como entrada dois arquivos *txt*, um da configuração do sistema (Página 0) e outro com os dos dados de leitura propriamente ditos.

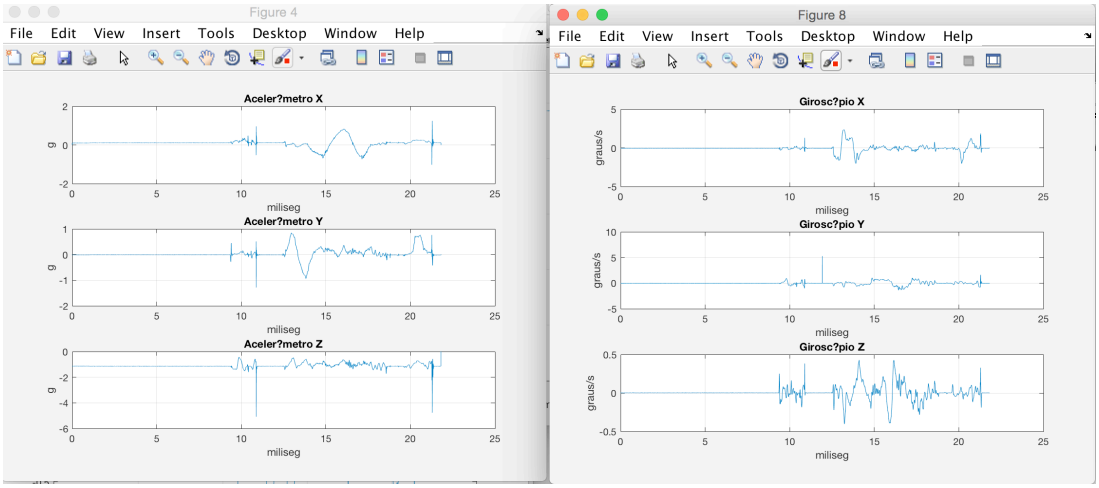


Figura 5.1. Mostra um exemplo dos gráficos gerados pelo Matlab com os valores de cada eixo de aceleração e giro coletados pelo equipamento.

Com esses arquivos é possível, ainda, realizar uma simulação 3D do comportamento do veículo quando do acidente. Para isso, foi utilizado a linguagem *Processing* de plataforma própria (aberta), muita utilizada por novos programadores a se familiarizarem com o ambiente programação. Baseada em *java*, essa linguagem apresenta sintaxe simples em um contexto visual gráfico e amplo de utilidades.

Como mencionado anteriormente, o equipamento criado é capaz de armazenar, em formato *txt*, os valores de aceleração e giro, nos três eixos, nos instantes anteriores e posteriores à colisão. Para visualizar o comportamento do veículo nesse período, criamos uma animação 3D de um carro de polícia, que se movimenta conforme os dados de giro armazenados. Além disso, apresenta-se de forma gráfica os valores de aceleração, aumentando ou diminuindo. Veja na Figura 5.2 um exemplo da tela do simulador.

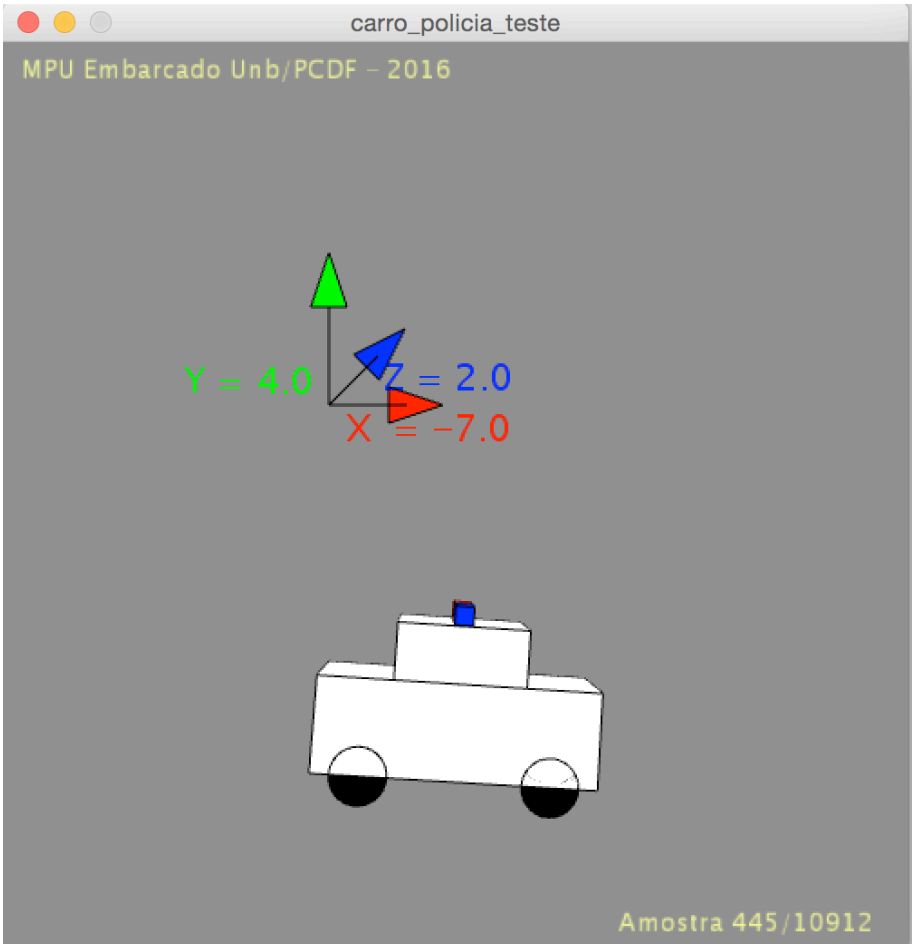


Figura 5.2. Mostra exemplo de tela do simulador. As setas indicam os valores da aceleração, enquanto o carro movimenta-se conforme os valores de giro. (Observação: Os valores de aceleração neste exemplo estão sem os descontos de calibração).

Assim como na montagem dos gráficos, para proceder a simulação, é preciso antes tratar o arquivo de entrada do programa. O arquivo deve estar dividido em 6 colunas de dados de aceleração e giro, nos eixos X, Y e Z, respectivamente, com espaçamento feito por *tab* entre eles. No capítulo anterior foi mencionado que o sistema possui um algoritmo que marca o início da memória com um bloco de zeros. Portanto, se quisermos visualizar, na simulação, a dinâmica a partir de seu dado mais antigo gravado, é necessário reorganizar os valores do arquivo texto, de forma a colocar primeiro aqueles sucessivos ao bloco de zeros e, em seguida, o restante.

O código prevê que os descontos de calibração (*offset*) sejam feitos. Para isso basta colocar nas devidas variáveis, as medições de calibração, encontradas na Página 0 (de configuração). No caso dos resultados de giro, além de descontar o *offset*, há ainda um passo a mais. Os dados coletados pelo MPU são registrados em graus por segundo, enquanto que a simulação necessita dos valores em radianos. Para realizarmos essa conversão, basta fazermos a diferença do valor coletado de velocidade ângular seguinte pelo atual e multiplicarmos pela taxa de amostragem, que é de uma amostra a cada 2 ms. Assim temos o valor em graus medido de uma amostra a outra. O passo seguinte é, simplesmente, converter para radianos.

Veja na figura seguinte, o momento exato de um capotamento em uma das simulações feitas com o dispositivo construído.

Com isso apresenta-se mais uma grande aplicabilidade desse equipamento para perícia nacional, com a inovação de permitir visualizar, de forma gráfica, a dinâmica de giro no momento do acidente. Como a plataforma de simulação é gratuita, o dispositivo torna-se ainda mais atraente.

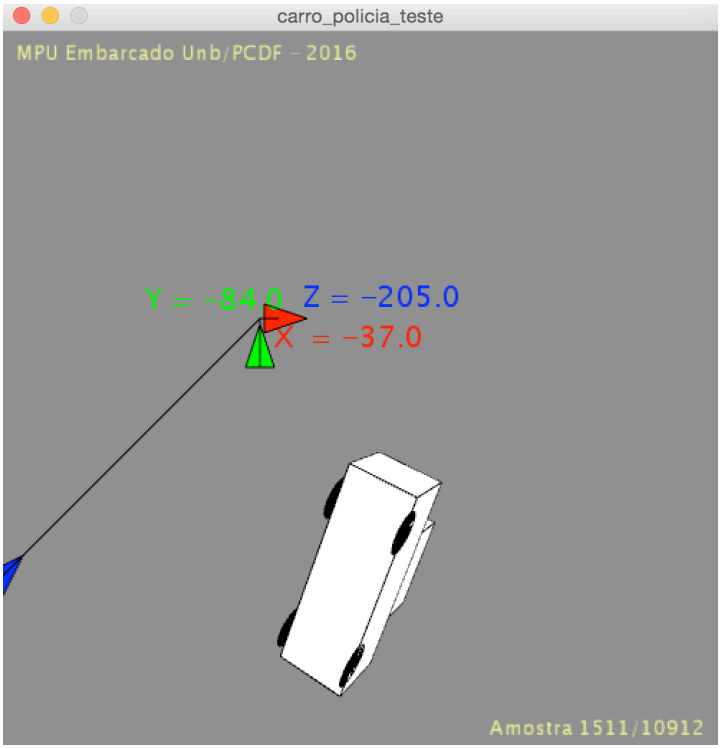


Figura 5.3. Mostra um exemplo de simulação de capotamento.

6 CONSIDERAÇÕES FINAIS E UTILIDADE PERICIAL

A velocidade é um dos principais elementos a ser desvendado nos laudos de exames em perícias de trânsito. Esse trabalho propõe um dispositivo de baixo custo, que teria grande utilidade para a área pericial de acidente de trânsito, e pretende contornar a dificuldade de se extrair este parâmetro variável dos atuais módulos de *airbag*. O equipamento proposto, quando acoplado a um veículo, grava dados de aceleração e velocidade angular no momento de uma colisão e ainda serve como auditoria da velocidade dos últimos segundos armazenados na memória.

O valor do giro coletado também seria importante para, além de poder projetar a dinâmica de movimentação do veículo, servir de comparação para verificar se o ponto de

repouso final encontrado pelos peritos no local do acidente, por exemplo, é compatível com os valores do giroscópio.

Sugere-se o acréscimo de um módulo *GPS* serial. Assim, além da velocidade no momento da colisão, o sistema forneceria, ainda, as coordenadas do veículo no instante do acidente. Esse tipo de sistema é simples, de baixo custo e, mais uma vez, engrandece o laudo pericial e, consequentemente, a investigação do crime relacionado.

O sistema proposto possui uma bateria de back-up para suprir energia na eventualidade de falha no sistema de 12 V do veículo. É possível gravar no bloco de configuração o instante do acidente ocorreu a falha de alimentação.

Um ponto importante a se considerar é o valor do limiar para caracterizar um acidente. A literatura nacional não é específica sobre este ponto. Ao que parece, o valor de 8 g é o que tem maior aceitação internacional [6]. Entretanto, cada fabricante usa um determinado valor. O sistema proposto se mostra, mais uma vez, de grande utilidade para a atividade forense, uma vez que possibilita encontrar qual é o valor de desaceleração que um determinado fabricante usa para ativar as bolsas e os pré-tensionadores. Por exemplo, consideremos um ensaio usando o módulo proposto configurado para disparar com uma desaceleração de 8 g instalado em um carro cujo módulo de *airbag* original esteja configurado para 4 g. Caso esse veículo envolva-se em colisão, que o faça disparar suas bolsas e ao mesmo tempo haja um corte de energia no sistema, teremos gravado em nosso equipamento o momento que houve o chaveamento da alimentação para a bateria (que provavelmente será algo em torno dos – 4 g). Se não houver o corte de energia, teremos no módulo projetado um valor de pico, onde ocorre a mudança de sentido do veículo (mudança de aceleração para desaceleração), saberemos assim, que o valor de disparo das bolsas no módulo original é algo próximo ou acima desse pico.

Vale reforçar que, dada a escassez de informações sobre os módulos de *airbag* e seu conteúdo (hardware e software), além da limitada literatura no assunto, o equipamento proposto serve como porta de entrada para o entendimento e estudo do funcionamento desses sistemas nos veículos fabricados no Brasil. É um eficiente aparelho para servir como base para desvendar o significado dos códigos coletados na memória dos módulos.

7 CONCLUSÕES

No Brasil existe uma grande dificuldade para se extrair informações de relevância pericial dos módulos de *airbag*. A literatura internacional afirma que esses dispositivos funcionam como caixa preta de veículos, armazenando informações como aceleração e velocidade de uma colisão que tenha acionado seu código de deflagração das bolsas e pré-tensionadores dos cintos de segurança [4]. Porém, nada há de concreto sobre os veículos que são fabricados no Brasil. As evidências levam ao questionamento se realmente gravam esse tipo de informação e, em caso positivo, como interpretar os bits coletados a partir de sua memória?

Tendo este problema como ponto de partida, o presente trabalho obteve êxito na construção de um equipamento que apresentasse ideia semelhante, mas com custo baixo e que pudesse gerar facilmente informações úteis na reconstrução de uma cena de acidente de trânsito. O sistema proposto faz uso de um processador Arduino e de um módulo MPU-6050 (acelerômetro e giroscópio) e de uma memória não volátil para armazenar dados dos 3 eixos de desaceleração e rotação durante um acidente.

O sistema é capaz de armazenar 20 s de informação o que representa 10.912 leituras de cada eixo X, Y e Z de aceleração e de giro. A grosso modo, são armazenados dados

sobre os 10 s que antecederam e sobre os 10 s que sucederam o acidente. Em posse dos valores de aceleração no eixo X (eixo de direção do veículo) é possível facilmente obter os valores de diferença de velocidade, parâmetro esse extremamente valioso nas perícias de trânsito.

O sistema proposto permite que os dados coletados sejam transpostos a um programa que reproduz, em três dimensões, o movimento de giro de veículo e o valor de aceleração medido a cada amostra. Assim temos uma poderosa ferramenta que auxilia no entendimento de como se deu o acidente.

Com isso reforça-se a grande aplicabilidade do dispositivo proposto para o mundo forense de acidentes de trânsito e como ele pode ajudar os peritos e experts do assunto.

8 TRABALHO FUTURO

Para trabalho futuro se propõe a implementação de ajustes que auxiliem ainda mais o trabalho da perícia, como a adição de outros componentes, por exemplo, o já citado módulo de GPS, e sensores, como um de verificação do estado do freio.

O equipamento trabalha a uma taxa de 1 amostra a cada 2 ms. Não se sabe ao certo a que taxa de amostragem os módulos de *airbag* tradicionais operam, portanto, é valido ampliar o algoritmo do protótipo para outras frequências de amostragem.

O teste real em uma grande variedade de veículos é válida para verificar o verdadeiro funcionamento do dispositivo. Parâmetros como limites de temperatura e umidade não foram testados e podem influenciar no bom funcionamento do equipamento.

Por fim considera-se útil, realizar ajustes mais finos nos programas criados, de forma a torná-los mais limpos e de simples entendimento do funcionamento do código.

Convém criar uma plataforma de interface gráfica para facilitar o acesso ao público em geral, mas principalmente os peritos da área.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BRASIL. Código Penal Brasileiro – Decreto de lei nº 3.689, de 3 de outubro de 1941. Art. 6º.

[2] BRASIL, CONTRAN (Conselho Nacional de Trânsito) – Resolução nº 380, de 2011. dispõe sobre a obrigatoriedade do uso do sistema antitravamento das rodas - ABS. Publicado no Diário Oficial em 3 de maio de 2011.

[3] Colaboradores da Wikipedia. Electronic Control Unit. Wikipedia, the free encyclopedia. Disponível em: < https://en.wikipedia.org/wiki/Electronic_control_unit> Acesso em 13 setembro 2016.

[4] W. Rosenbluth. Black Box data From Accident Vehicles, Methods of Retrival, Translation, and Interpretation. Ed. ASTM International. 2009.

[5] Colaboradores da Wikipedia – Event Data Recorder. Wikipedia, the free encyclopedia. Disponível em: < https://en.wikipedia.org/wiki/Event_data_recorder>. Acesso em 5 de novembro de 2016.

[6] Rosalyn G. Millman. 49 CFR Parts 552, 571, 585 and 595. National Highway Traffic Safety Administration, Departament of Transportation.

[7] Accident Investigation and Reconstruction Specialists, Inc – Bosch Crash Data Retrival (CDR) System Technician I and II Training. Disponível em: <http://www.airs-inc.net/bosch-cdr-tech-i-and-ii.html>. Acesso em 5 de novembro de 2016.

[8] Wikipedia – Bosch Diagnostics. Disponível em: <https://www.boschdiagnostics.com/cdr/>. Acesso em 21 de outubro de 2016.

[9] Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V. 2549Q–AVR–02/2014. Datasheet.

[10] Zelenovsky, R. e Mendonça, A. “Arduino Avançado”, MZ Editora, a ser publicado em 2018.

[11] Atmel. 3-Wire Serial EEPROMs, AT93C46, AT93C56, AT93C5. Datasheet. Rev.0172K–07/98.

[12] Microship. 93LC46/56/66, 1K/2K/4K 2.5V Microwire Serial EEPROM. Datasheet. DS21712B.

[13] Fairchild Semiconductor. FM93C66A, 4K-Bit Serial CMOS EEPROM, (MICROWIRE™ Synchronous Bus). Datasheet. July 2000.

[14] InvenSense Inc. MPU-6000 and MPU-6050 Product Specification Revision 3.4. Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013.

[15] Colaboradores da Wikipedia – ASCII. Wikipedia, the free encyclopedia. Disponível em: < https://en.wikipedia.org/wiki/ASCII>. Acesso em 10 de dezembro de 2016.

[16] Kawano, N. M. Bosch CDR Training Program, Relatórios CDR, case VIN 3G7DA03E83S. Seminário Nacional de. Perícia em Crime de Trânsito e Identificação de Veículos. Apresentação feita em 12 de maio de 2016. Cuiabá, Brasil.

[17] Michroship 24AA1025/24LC1025/24FC1025. DS21941B. Datasheet.

[18] Maxim Integrated DS1307 64 x 8, Serial, I2 C Real-Time Clock.

[19] Freescale Semiconductor. Implementing Auto-Zero Calibration Technique for Accelerometers. AN3447. Rev 0, 03/2007.

[20] C.Y. Chan. Fundamentals of Crash Sensing in Automotive Air Bag Systems. Ed. SAE International. 2000.