Universidade de Brasília

Faculdade de tecnologia
Departamento de Engenharia Elétrica

Protótipo de uma Caixa Preta para veículos terrestres, com ênfase no GPS e na alimentação da placa

Tássila Catarina Sigurace de Almeida

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia de Redes de Comunicação

Orientador
Prof. Dr. Ricardo Zelenovsky

Brasília, DF

2023

AGRADECIMENTOS

Dedico este trabalho a todos os meus amigos e familiares que me ajudaram e apoiaram a conclusão dele, bem como ao meu orientador que além de possuir um grande conhecimento da área acadêmica, é um ser humano com sensibilidade e simpatia para com os estudantes, sempre disposto a ajudar e apoiar nas maiores turbulências.

RESUMO

Este trabalho é uma etapa do desenvolvimento do projeto denominado Caixa Preta para Veículos Terrestres, tais como carros, caminhões e motos. Nesta etapa foi desenvolvido o acesso, o registro e a exibição dos dados do GPS empregado no projeto. Para tal, foram feitos ensaios com a placa, dentro de um veículo em movimento, recolhendo-se em tempo real dados de localização deste, e simulando a quais desses dados seriam salvos em um eventual acidente.

Além disso, aqui se apresentam soluções e ensaios para que o hardware funcione de forma independente com o emprego de super capacitores. É proposto ainda um circuito para permitir que o CPU desligue completamente todo o hardware. São relatadas algumas peculiaridades do Arduino Mega 2560, que é o processador usado deste projeto, quando opera com tensões próximas ao seu limite de operação. Tais ensaios foram feitos utilizando um osciloscópio, e baseados na tentativa de resolução dos seguintes objetivos: Alimentar um Arduino pela tomada de 12V de um carro (acendedor de cigarro); Garantir o funcionamento momentâneo do Arduino após o desligamento do carro; Permitir que o Arduino se auto desligue; Prever recursos para monitorar a alimentação e a carga do super capacitor.

E por fim, também se encontrou uma solução para a conexão dos botões (chaves) da placa no Arduino, sem que este perdesse grande parte de suas portas com isso. Para solucionar o problema das chaves propôs-se um divisor resistivo, conectado à porta do conversor analógico digital (ADC) do Arduino, permitindo o uso linear de uma faixa de tensões.

Palavras-chaves: Caixa preta (CXP), Sistema de posicionamento global (GPS), Super capacitor.

ABSTRACT

This work is a stage in the development of the project called Black Box for Land Vehicles, such as cars, trucks and motorcycles. At this stage, access, recording and display of GPS data used in the project was developed. To this end, tests were carried out with the plate, inside a moving vehicle, collecting real-time location data, and simulating which of these data would be saved in a possible accident.

Furthermore, solutions and tests are presented here to make the hardware work independently using super capacitors. A circuit is also proposed to allow the CPU to completely turn off all hardware. Some peculiarities of the Arduino Mega 2560, which is the processor used in this project, are reported when it operates with voltages close to its operating limit. Such tests were carried out using an oscilloscope, and based on an attempt to solve the following objectives: Powering an Arduino through the 12V socket of a car (cigarette lighter); Ensure the momentary operation of the Arduino after turning off the car; Allow the Arduino to turn off itself; Provide resources to monitor the power supply and charge of the super capacitor.

And finally, a solution was also found for connecting the buttons (switches) on the board to the Arduino, without losing a large part of its ports as a result. To solve the switch problem, a resistive divider was proposed, connected to the Arduino analog-to-digital converter (ADC) port, allowing the linear use of a range of voltages.

Keywords: Black box (CXP), Global positioning system (GPS), Super capacitor.

LISTA DE ABREVIATURAS E SIGLAS

CDR	*Crash Data Retrieval*
CSMU	*Crash-survivable memory unit*
CXP	*Caixa preta*
DD	*Decimal Degrees*
DLC	*Diagnostic Link Connector*
DMS	*Degrees-Minutes-Seconds*
DOP	*Dilution of Precision*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
FDR	*Flight data recorders*
GPGGA	*Global Positioning System Fix Data*
GPGSA	*GPS DOP and active satellites*
GPGLL	*Geographic Position global latitude e longitude*
GPGSV	*GPS Satellites in view*
GPRMC	*Recommended minimum specific GPS/Transit data*
GPS	*Global Positioning System*
GPVTG	*Track made good and ground speed*
IMU	*Inertial Measurement Units*
LCD	*Liquid Crystal Display*
LED	*Light-Emitting Diode*
MEMS	*Micro Electro Mechanical System*
NMEA	*National Marine Electronics Association*
PCB	*Printed Circuit Board*
QFN	*Quad Flat No leads*
SPP	*Serial Port Protocol*
SPI	*Serial Peripheral Interface*
SRAM	*Static random-access memory*

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

De acordo com Peter Ashford, em sua publicação em fevereiro de 2010 para a revista "*Avionics News*" [4], por volta de 1939, os franceses Paul Beaudouin e François Hussenot, fizeram os primeiros testes, do que hoje podemos chamar popularmente de "Caixa preta - CXP" ou *Flight Data Recorders* - FDR (Gravadores de dados de voo). Esta evoluiu para algo correspondente a registro de parâmetros específicos de desempenho da aeronave. Esses dados registrados por uma FDR são usados para investigação de acidentes e para analisar questões de segurança aérea, degradação de material e desempenho do motor. Inicialmente os registros das CXP eram feitos em filmes fotográficos, com o auxílio de raios de luz e um espelho que se inclinava de acordo com a magnitude dos dados. Este gravador de dados era usado não apenas em aeronaves, mas em trens e outros veículos. Segundo a BBC Brasil seu nome popularizado é devido a uma expressão da época que denominava de "caixa preta" um equipamento aéreo de função e interior desconhecidos. Como era uma nova invenção e muitos não sabiam de seu funcionamento, recebeu este apelido, mas as atuais são de cor laranja brilhante, para fornecer visibilidade nos destroços, de um acidente e são resistentes ao calor.

Já a edição de março de 2010 da "*Avionics News*"[5], fala sobre a importância da sobrevivência de uma FDR, onde os dados são salvos em gravadores de estado sólido, substituindo as antigas fitas magnéticas. Atualmente, os FDR nada mais são que matrizes empilhadas de chips de memória, sem peças móveis, por isso apresentam menos problemas de manutenção e maiores chances de sobrevivência. Diversas informações, como tempo, pressão, altitude, fluxo de combustível, entre outras, são gravadas na *Crash-Survivable Memory Unit* – CSMU (unidade de memória de sobrevivência em colisão), que é projetada e testada para sobreviver à altas e baixas temperaturas, à pressão hidráulica e à impactos grandes. A caixa preta em si é composta pela junção dos técnicos de voo e da gravação do áudio da cabine. Todas essas informações são fisicamente armazenadas na parte traseira do avião, fazendo com que a frente sirva de "zona de esmagamento", aumentando sua chance de sobreviver a acidentes. As mais modernas possuem um sistema de "auto ejeção" na hora do impacto, e outras, como em ônibus espaciais, usam a tecnologia de *downlinks* como alternativa à transferência de dados, no lugar de uma caixa preta.

Com relação a veículos autônomos, uma das empresas no mercado que possui algo semelhante a uma caixa preta, é a "*BOSCH Invented for life*". Segundo a marca, seu kit chamado "Kit básico DLC (*Diagnostic Link Connector* - Conector de Link de Diagnóstico) de recuperação de dados de falha", dispõe de um software CDR (*Crash Data Retrieval* - Recuperação de dados de falha), um módulo verde de microcontrolador CDR , fonte de alimentação, cabos e adaptadores, todos para ligação do módulo no veículo e em um computador, que seja baseado em windows.[8]

Figura 1.1: Kit básico DLC de recuperação de dados de falha.[8]

O módulo de recuperação de dados de falha geralmente já é instalado de fábrica no próprio veículo, como uma função do sistema de segurança do carro ou do airbag. Tal software fornece as informações de falhas sob a forma de um relatório CDR, que apresenta os seguintes parâmetros de dados: Velocidade do veículo; Status do freio; Ângulo do volante; Gravidade da falha (delta-V); Status do cinto de segurança; Acelerador do motor; Posição do pedal do acelerador; Posição da engrenagem de transmissão; Dados de implantação do airbag; e Informações de detecção de ocupantes.[8]

Desde a sua criação, para gravação de voos, até sua chegada em veículos autônomos, o surgimento de módulos de gravação de dados, ou caixas pretas, é de extrema importância para os tripulantes de um veículo que possam ter passado por algum acidente. Por conta disso, esse trabalho faz parte de uma evolução de projetos que visam desenvolver uma CXP brasileira. O ponto evolutivo onde este trabalho se encontra é descrito na seção seguinte.

1.1. Motivação

A ideia para uma CXP em um automóvel, surgiu com o intuito de auxiliar a perícia de acidentes de trânsito, que tem uma grande demanda no Brasil. Atualmente, a perícia para a reconstrução da dinâmica de um acidente de trânsito toma como base marcas físicas deixadas no local. São tipicamente buscados indícios físicos tais como marcas de frenagem no asfalto, deformação do próprio carro, danos na região onde aconteceu o acidente. Muitas vezes tais pistas não ficam acessíveis, como é o caso de chuva, acidente em estradas não pavimentadas e até ação maliciosa dos envolvidos no acidente.

Segundo a BBC Brasil, assim como a invenção da FDR foi de início vista como não comercial, devido a uma de suas implementações, a de gravação das vozes dos pilotos da cabine, que hoje é obrigatória e bem aceita, a CXP pode parecer um investimento desnecessários, porém, do mesmo modo que gravar as vozes da cabine ajudou nas investigações de muitos acidentes aéreos, uma caixa-preta para veículos autônomos ajudaria muito a perícia a determinar o ocorrido em um acidente envolvendo carros, ônibus em estradas, caminhões, entre outros. Além do que, para empresas de transporte de carga, isso seria ideal para ajudar na determinação de possíveis culpados de um incidente e para identificar motoristas ruins.

Este trabalho faz parte de uma evolução de trabalhos relacionados a CXP, alguns já concluídos e outros que continuam desenvolvendo e implementando aperfeiçoamentos ao projeto. Portanto, o ponto onde o atual trabalho se encontra, pode ser visto na lista cronológica a seguir:

- Vinícius de Oliveira Lima: "*Proposta de Plataforma Inercial para Auxiliar na Perícia de Acidentes de Trânsito*", aluno de mestrado de 2016.

- Hudson Pereira Ramos e Vanessa Oliveira Lucena: "*Proposta de plataforma inercial e simulador 3D para periciar acidentes de trânsito*", alunos de graduação de 2017.

1º Protótipo Caixa Preta versão 1 de 2017. Figura 3.1

- Gabriela da Silva Lopes: "*Caixa Preta para Veículos Automotivos*", aluna de graduação de 2018.

- Renato Estevam Nogueira: "*Perícia de acidentes de trânsito utilizando uma plataforma inercial com magnetômetro*", aluno de graduação de 2019.

2º Protótipo Caixa Preta versão 2 de 2019. Figura 3.2

- José Luiz G. Nogueira: "*Caixa Preta para Carros: Comparação de métodos de estimativa de inclinação usando acelerômetro, giroscópio e magnetômetro*", aluno de graduação de 2021.

- Paulo B. Teixeira Neto: "*Caixa Preta para carros: proposta para calibração, coleta e fusão de dados*", aluno de graduação de 2021.

- Tássila Catarina Sigurace de Almeida: "*Protótipo de uma Caixa Preta para veículos terrestres, com ênfase no GPS e na alimentação da placa*", aluna de graduação de 2023.

3º Protótipo Caixa Preta versão 3 de 2021. Figura 3.3

- Matheus Ribeiro: "*Projeto e fabricação de hardware para caixa preta automotiva e densímetro*", aluno de graduação 2022.

Para melhor visualização, temos a linha temporal ilustrada na Figura 1.2.

Figura 1.2: Evolução do Projeto Caixa Preta.

Como se vê, este projeto trabalhou com o 2º protótipo da caixa, onde ensaiou alguns aperfeiçoamentos do uso da bateria e chaves na placa, e testou algumas das soluções. Bem como a simulação do uso do GPS da CXP em um veículo em movimento.

1.2. Objetivo

Um dos intuitos do trabalho está relacionado com o estudo e soluções para a conexão dos botões (chaves) da placa no Arduino, sem que este perdesse grande parte de suas portas com isso. Outro foco, visa demonstrar soluções para as seguintes propostas relacionadas a sustentação de energia da caixa preta: Alimentar um Arduino Mega 2560 pela tomada de 12V de um carro (acendedor de cigarro); Garantir o funcionamento momentâneo do Arduino Mega 2560 após o

desligamento do carro; Permitir que o Arduino Mega 2560 se auto desligue; Prever recursos para monitorar a alimentação e a carga do super capacitor.

E mais relacionado ao GPS, tem-se a meta de entender o funcionamento deste na PCB (*Printed Circuit Board*), quando ele está em movimento dentro de um veículo, e parado fora deste, e como a última versão do software está atuando nesses testes.

1.3. Organização do Trabalho

Após a fabricação do protótipo caixa preta versão 1 de 2019, foram feitas pesquisas bibliográficas e de ensaios experimentais com a placa. Com isso, a estruturação deste trabalho para abordar tais temas foi feita da seguinte maneira:

1. INTRODUÇÃO é onde definimos o problema que se quer solucionar, e descrevemos quais aspectos e pontos deste, que serão focados para o estudo.

2. FUNDAMENTAÇÃO TEÓRICA é o local nos quais estão as descrições bibliográficas dos componentes da CXP versão 1 de 2019.

3. MATERIAIS E MÉTODOS é o capítulo onde se fazem as descrições do hardware e do software. O primeiro tema de hardware tem foco na solução com super capacitor para sustentar o funcionamento da CXP de forma a permitir a gravação dos dados em memória não volátil e também dos botões disponibilizados para o usuário interagir com a placa. Já o segundo tema faz a descrição da estruturação do software e descreve como ele foi usado para os testes do GPS.

4. DADOS OBTIDOS são todos os testes que foram feitos com a placa, em movimento ou não, e uma exposição de mapas gerados pela captura de dados do GPS, sendo feitas comparações e observações do que pode ou não funcionar para a melhoria do projeto.

5. CONSIDERAÇÕES FINAIS são as conclusões sobre o trabalho realizado, indicação de descobertas importantes, assim como sugestões para melhoras futura e para novos estudos sobre o tema.

2. FUNDAMENTAÇÃO TEÓRICA

A criação de uma caixa preta para veículos terrestres necessita coletar e armazenar alguns tipos de dados, para que estes possam ser acessados e utilizados por peritos em acidentes. Pensando nisso foi feita uma placa de circuito interno, para testes, capaz de coletar a localização do veículo, sua aceleração, velocidade e seu giro. A Figura 3.2 demonstra como a placa de testes foi construída, e a localização de seus componentes.

Para a coleta e testes de dados da Caixa Preta Versão 2.0, utilizou-se na PCB um Arduino Mega 2560, display LCD, uma interface *bluetooth*, memórias, LEDs e dois tipos de sensores, um deles é um módulo inercial, contendo acelerômetro e giroscópio, e o outro um receptor GPS. Todos esses componentes foram conectados à placa, para testar a competência deles, e a real necessidade de cada um, bem como foram reservados espaços extras, numa placa padrão, para a implementação e testes de outros, caso seja necessário. Todos eles são mais bem descritos a seguir.

2.1. Aquisição e Recepção de Dados

Aqui será descrito os dois tipos de sensores utilizados, a unidade de medição inercial e o GPS, que são as peças fundamentais do protótipo para a aquisição de dados, bem como alguns dos dispositivos usados para visualização desses dados, no momento dos testes, com o LCD e o *bluetooth*.

2.1.1 Unidade de Medição Inercial

A unidade de medição inercial, ou IMU (*Inertial Measurement Units*), escolhida para este projeto foi o modelo MPU-9250, descrita no datasheet da InvenSense, que nada mais é que um dispositivo de rastreamento de movimento de 9 eixos, pois este módulo multi-chip comporta duas matrizes integradas em um pacote QFN (*Quad Flat No leads*), uma com um giroscópio e um acelerômetro de 3 eixos cada, e a outro com magnetômetro de 3 eixos. A integração desses eixos proporciona uma melhor quantidade de dados, visto que as 3 medidas (aceleração, giro e campo magnético) podem ser usadas num filtro de fusão de dados para melhorar as estimativas importantes para a reconstrução do acidente.[14]

Figura 2.1: Placa que facilita o uso MPU-9250 que oferece, acelerômetro, giroscópio e magnetômetro, todos com 3 eixos. [21]

Acelerômetro

O acelerômetro de 3 eixos do MPU-9250 é um MEMS (*Micro Electro Mechanical System*), que funciona da mesma maneira que um acelerômetro discreto, porém em uma escala dimensional muito menor. Ou seja, é um sistema que indica a projeção da aceleração da gravidade nos 3 eixos do dispositivo. Portanto, quando um dos eixos está na vertical ele deve indicar 9,98m/s. Para estimar a aceleração, esse sistema utiliza o deslocamento de uma massa que, presa as molas, altera o valor de capacitores. Essa variação na capacitância é usada para estimar a aceleração, como ilustrado na Figura 2.2. Já na Figura 2.3, podemos observar uma imagem ampliada desse sistema em um MEMS.[9]

A aceleração que iremos encontrar com essas medidas, é uma aceleração dinâmica, e um grande exemplo desta são vibrações e choques, como acidentes de carro, em que a mudança de aceleração é repentina, comparada ao seu estado anterior, que é o objetivo de pesquisa da CXP.

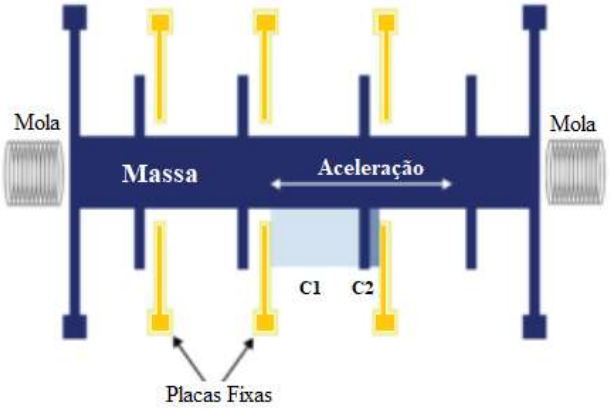


Figura 2.2: Funcionamento teórico de um acelerômetro MEMS.
Fonte: ZELENOVSKY e MENDONÇA (2019). [25]

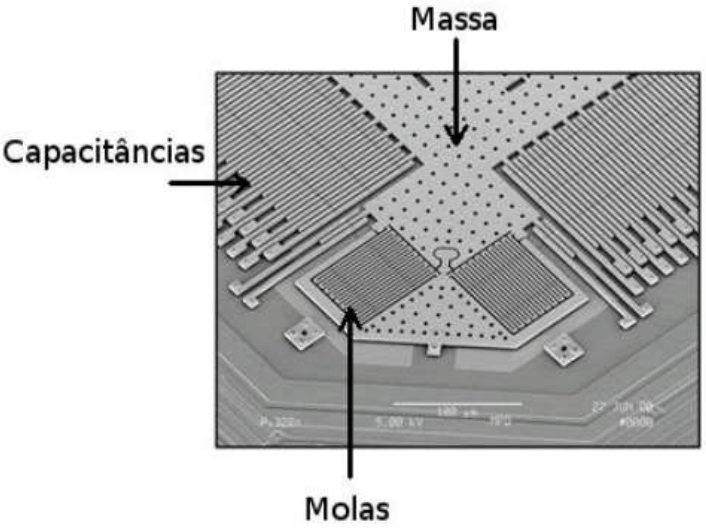


Figura 2.3: Fotografia de um acelerômetro MEMS.
Fonte: CALACHE (2013). [9]

Giroscópio

Um giroscópio de 3 eixos MEMS, apresenta elementos semelhantes ao acelerômetro, como massa de prova e molas, porém, ele irá calcular a velocidade angular do sistema, utilizando-se do Efeito Coriolis. A Figura 2.4 ilustra esse efeito, que consiste em aplicar uma força a uma massa, em uma dada direção, e nesta mesma massa aplicarmos um giro, isso irá gerar uma força perpendicular a velocidade inicial, chamada de força de Coriolis "*Fc*", que pode ser vista a seguir, onde "*W*" é a velocidade angular, e "*V*" a velocidade.

$$\vec{F_c} = -2\vec{W} * \vec{V}$$

No giroscópio MEMS, Figura 2.5, se utilizam dos mesmos princípios que o acelerômetro, com a diferença de se ter uma oscilação constante na horizontal, mantendo a massa sempre em movimento, e quando esta mover-se na direção das molas, ocorrerá um giro, e a diferença de

capacitância nos dará a estimativa da aceleração de Coriolis, e a partir desta que se encontra a velocidade angular. [25]

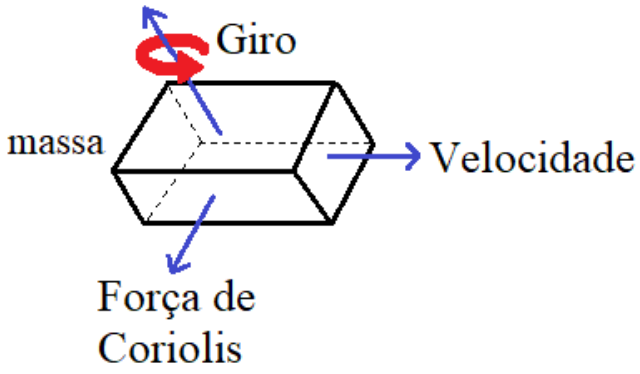


Figura 2.4: Força de Coriolis.

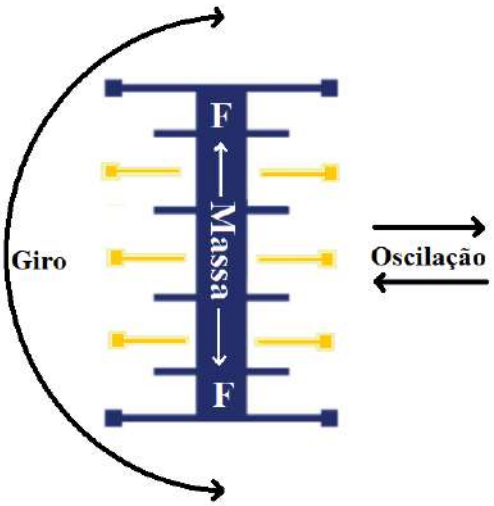


Figura 2.5: Giroscópio MEMS.

Fonte: ZELENOVSKY e MENDONÇA (2019). [25]

Magnetômetro

O magnetômetro MEMS utilizado na MPU-9250 é o AK8963, segundo o datasheet da InvenSense [13], e ele servirá para a detecção do campo magnético próximo ao IMU. De acordo com o datasheet da Asahi KASEI, ele se utiliza do efeito Hall para medir e estimar tais medidas, e é mais indicado para o uso de navegação pedestre na cidade para telefones celulares e outros aparelhos portáteis, como uma bússola eletrônica de 3 eixos.

O magnetômetro usado emprega três sensores de efeito Hall para oferecer medição de campo magnético em 3 eixos. Cada sensor Hall é ativado com uma corrente constante e a presença de um campo magnético perpendicular a esta provoca o surgimento de uma tensão que é proporcional à intensidade deste campo. Assim, é possível estimar a magnitude/força do campo magnético. Como mostrado na Figura 2.6, verifica-se a aparição de uma diferença de potencial, chamada de tensão Hall, que ocorre devido à força de Lorentz, que desvia as cargas elétricas da sua trajetória. Essa tensão é perpendicular à corrente e ao campo elétrico.[1]

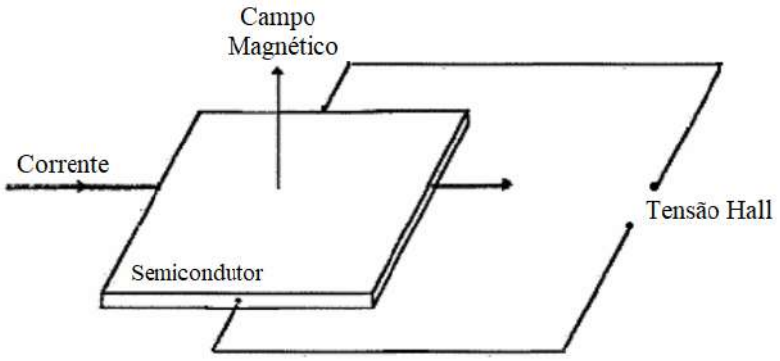


Figura 2.6: Força de Coriolis.

Fonte: ARAÚJO (2009). [1]

2.1.2 Sistema de Posicionamento Global

Segundo GREWAL, WEILL e ANDREWS (2007), o GPS (*Global Positioning System*) foi criado pelo departamento de defesa norte americano para o auxílio na navegação, mas com o tempo se tornou adequado para outras funções. No período em que o livro foi escrito, os autores informam que em um sistema ideal, o GPS teria um total de 24 ou mais satélites ativos, dispersos de forma uniforme em seis órbitas, como ilustrado na Figura 2.7. Com isso, teoricamente serão sempre visíveis, em qualquer parte da superfície terrestre, 3 ou mais satélites, sendo necessário para uma localização mais precisa dos receptores GPS, quatro ou mais. [18]

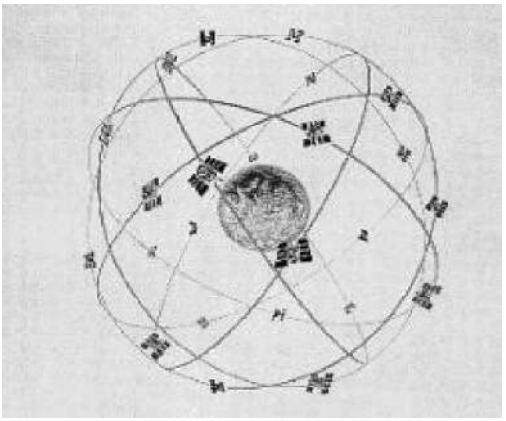


Figura 2.7: Forma simplificada de constelações de satélite para cobertura global. Fonte: LIMA (2005). [19]

O posicionamento do receptor GPS se dá através da recepção de sinais eletromagnéticos enviados pelos satélites em órbita da Terra (LIMA, 2005).[19] Quando um receptor se conecta a um satélite, este projeta uma esfera imaginária em torno de si, com as possíveis localizações do GPS. Como já foi dito, são necessários 4 ou mais satélites para uma posição precisa. Então, com o cruzamento de um número maior de esferas de satélites, o ponto onde todas de cruzam fica mais restrito e único (se os satélites estiverem não muito próximos), obtendo o local que se encontra o receptor. Esse processo é chamado de Trilateração, segundo o GISGeography. [15] As Figuras 2.8a, 2.8b e 2.8c ilustram isso.

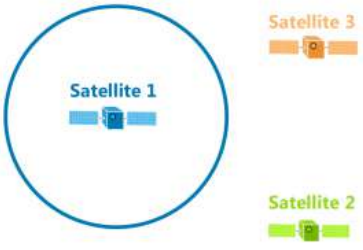


Figura 2.8a: Conexão com um satélite.

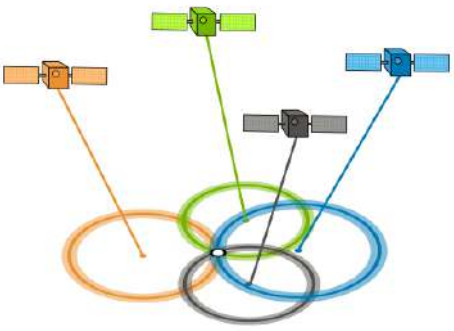


Figura 2.8b: Cruzamento de quatro satélites conectados.

Figura 2.8c: Esferas com o cruzamento dos quatro satélites

Fonte: GISGeography [15]

Com o tempo, a indústria de produtos eletrônicos evoluiu com a miniaturização dos GPS, visando o um uso em navegação pessoal, e por isso deveria ter um custo mais baixo e um consumo de energia menor, para que fossem adequados para o consumo doméstico e projetos pessoais.(LIMA, 2005) [18] Um desses produtos será o utilizado em nosso projeto, que é o modelo GPS - GY-NEO6MV2, ilustrado na Figura 2.9. Segundo sua ficha técnica ele possui um alto nível de capacidade de integração com opções de conectividade flexíveis, com um baixo custo e alto desempenho em um produto pequeno (16 x 12,2 x 2,4 mm).[12]

As informações fornecidas pelo GPS, são baseadas nas especificações da NMEA (*National Marine Electronics Association*), que detalham dados para dispositivos eletrônicos como eco-localizadores, giro-compassos e receptores GPS. Algumas dessas mensagens são: GPGGA, GPGSA, GPGLL, GPGSV, GPRMC e GPVTG, citadas na seção 4 e melhor descritas no Apêndice.

Figura 2.9: GPS - GY-NEO6MV2.[12]

Latitude e Longitude

De acordo com BOLSTAD (junho 2019), as coordenadas geográficas em uma esfera, são definidas pela Latitude, que mede as distâncias norte-sul, e Longitude que mede as distâncias leste-oeste, dos pontos sobre essa esfera. Os paralelos são linhas de mesma latitude, e estas são marcadas com o grau zero na linha horizontal do equador e vão até +90º para o Norte e -90º para Sul. Já as longitudes são definidas como meridianos, com seu ponto zero definido no meridiano de Greenwich e diferente dos paralelos, todos os meridianos se cruzam nos dois polos da Terra, e vão de 0º até +180º na parte Leste, e de 0º até -180º na Oeste; perceba que os pontos +180º e -180º se encontram no mesmo lugar, e, portanto, são o mesmo.[7] Isso tudo está ilustrado na Figura 2.10.

As duas formas mais usuais de se representar uma localização, são através das notações DMS (*Degrees-Minutes-Seconds*), exemplo.: 1º 50’ 23.0’’S, ou DD (*Decimal Degrees*), exemplo.: -15.839725, onde o sinal representa a direção (- Sul e Oeste, + Norte e Leste). A Figura 2.11 resume as conversões entre DMS e DD.[7]

Na Figura 2.10, apresenta-se de forma clara os meridianos e os paralelos. Vemos que, como era de se esperar, há uma separação proporcional entre as distâncias de latitude e longitude, na área que corresponde a linha do equador e seus paralelos mais próximos. À medida que nos aproximamos

dos pólos, pelo fato da Terra ser modelada por uma esfera, ocorre uma distorção, das distâncias dos meridianos, pelo fato destes se cruzarem nesses pontos, diferente dos paralelos. [7]

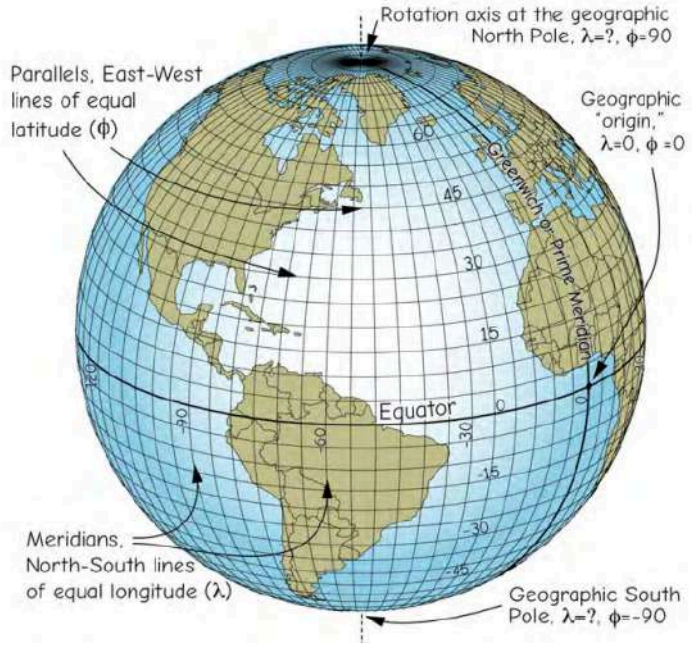


Figura 2.10: Convenções de Latitude e Longitude

DD from DMS

DD = D + M/60 + S/3600

e.g.

DMS = 32° 45' 28"

DD = 32 + 45/60 + 28/3600

= 32 + 0.75 + 0.0077778

= 32.7577778

DMS from DD

D = integer part

M = integer of decimal part x 60

S = 2nd decimal x 60

e.g.

DD = 24.93547

D = 24

M = integer of first decimal x 60

= 0.93547 x 60

= integer of 56.1282

= 56

S = 2nd decimal x 60

= 0.1282 * 60 = 7.692

so DMS is

24° 56' 7.692"

Figura 2.11: Conversão de Grau-minuto-segundo (DMS) para Grau-decimal (DD), e vice-versa

Fonte: BOLSTAD (junho 2019) [7]

2.2. Processamento e armazenamento de dados

Para o processamento das informações retiradas dos sensores, assim como sua visualização e gravação dos dados, se faz necessário incorporar mais alguns elementos no hardware da caixa preta. Para questões de teste, foram acrescentados LCD, bluetooth, memórias SRAM e EEPROM, e claro, a plataforma inercial MPU-9250.

2.2.1 LCD e Bluetooth

LCD (Liquid Crystal Display)

Como pode ser observado na Figura 3.3, o LCD utilizado é do formato 4x20, do modelo HD44780, da família de controladores LCD Hitachi. Com seu *back ligh*t azul e escrita branca, a função básica desse dispositivo é fornecer, de forma contínua, os dados que estão escritos em sua memória RAM, segundo ZELENOVSKY e MENDONÇA, 2019. [25] No projeto Caixa Preta ele é usado para facilitar a tarefa de desenvolvimento e depuração dos códigos.

Bluetooth

O módulo bluetooth utilizado em nossa CXP foi o modelo HC-05 Bluetooth SPP (*Serial Port Protocol*), com tamanho e projetos feitos para configuração serial sem fio, sendo usado aqui para o auxílio de outros dispositivos na comunicação dos dados da placa. Detalhamentos de suas

configurações são encontrados em seu datasheet.[11] Ele é muito importante, pois toda a comunicação com a CXP é feita via terminal serial.

2.2.2 Microcontrolador

De acordo com ZELENOVSKY e MENDONÇA, de 2019 [25], microcontroladores, como o nome diz, são dispositivos de controle digital, que oferecem recursos de entrada e saída de dados, possibilitando sistemas de tamanho pequeno e medidas de intervalos de tempo, não precisando realizar operações sofisticados com tais dados, diferentemente do que se propõe um microprocessador. A família utilizada no projeto foi a do Atmel AVR (usada no Arduino), e o microcontrolador escolhido foi o Arduíno Mega 2560, como visto na Figura 2.12,

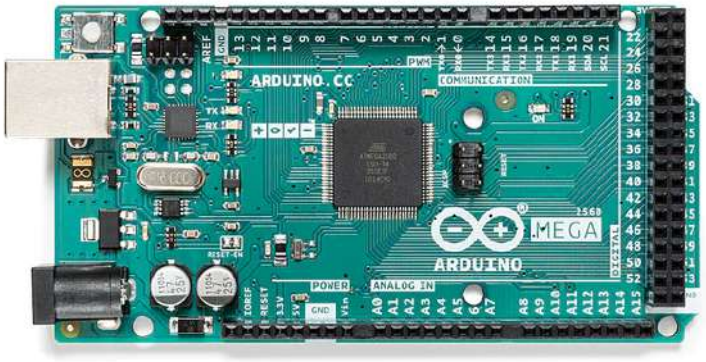


Figura 2.12: Arduino Mega 2560. [2]

2.2.3 Memória SRAM (*Static random-access memory*)

São memórias utilizadas para o armazenamento volátil de dados. Foi selecionada a SRAM 23LC1024 que oferece 128 KB. Com o emprego de duas dessas memórias chega-se a 256 KB. O acesso a essas memórias é feito via protocolo SPI (*Serial Peripheral Interface*). [25] Essas duas memórias são usadas para gravar continuamente o estado da CXP. Em caso de acidente, toda memória SRAM é copiada para uma memória EEPROM.

2.2.4 Memória EEPROM (*Electrically-Erasable Programmable Read-Only Memory*)

Para memória EEPROM foi selecionada a 24FC1025, que tem capacidade de 128 KB. Como são usados dois chips, chega-se a 256 KB, a mesma capacidade da SRAM. Enfatizando, em caso de acidente, os dados transitórios das SRAM são gravados nas EEPROM. Essa EEPROM é acessada via barramento I^2C (Inter-IC Bus). O I^2C é um tipo de protocolo para comunicação serial, necessitando apenas de dois fios para comunicação bidirecional, segundo ZELENOVSKY e MENDONÇA, de 2019. [25]

3. MATERIAIS E MÉTODOS

3.1. Hardware

O projeto de uma caixa-preta desenvolvida para veículos autônomos, assim como muitos outros desse tipo, passou por uma série de evoluções e mudanças tanto do Software, quanto do Hardware. A evolução da caixa preta, que chamaremos de “Versão 1” de 2017, para a “Versão 2” de 2019, com relação ao hardware, se deu de forma a criar uma PCB exclusiva, organizando os componentes e os fios, como pode ser visto na comparação das Figuras 3.1 e 3.2. Outra mudança foi na quantidade de espaços extras, deixados para a alocação de outros dispositivos, prevendo testes ou adaptações futuras, o acoplamento de um módulo bluetooth, e um número maior de memórias, sem contar com um melhor desenvolvimento dos botões e LCD para interação com o usuário, o que será melhor descrito mais à frente.

Com o tempo e o desenvolvimento da pesquisa, uma outra placa foi desenvolvida, com o tamanho menor de 10cm x 10cm, diferente da anterior que possuía o tamanho de 21cm x 13cm. A nova PCB também possui os elementos principais de uma CXP, como o MPU, GPS e o microcontrolador ATMmega, mas todos de uma forma desmembrada, para que coubessem no tamanho, e no custo-benefício da placa. Esta, como se vê na Figura 3.3, obteve sua criação e desenvolvimento no trabalho do aluno Matheus Rotta Ribeiro, e só está sendo citada aqui, como uma menção à evolução do projeto, ela não será abordada neste trabalho.

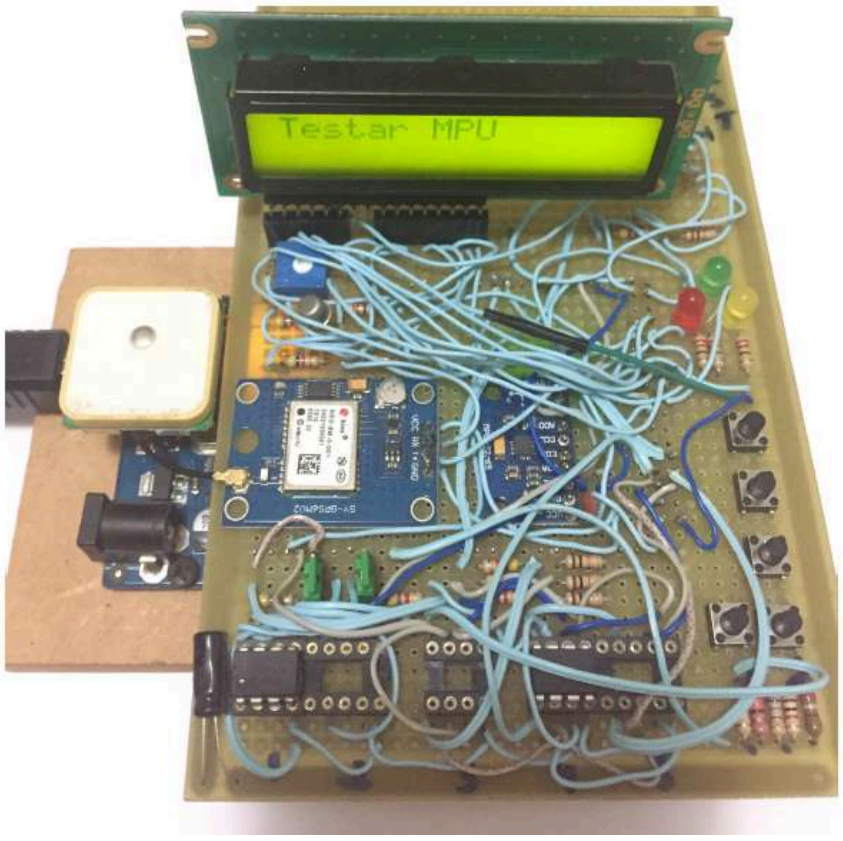


Figura 3.1: Primeira versão da Caixa Preta (Versão 1, de 2017).

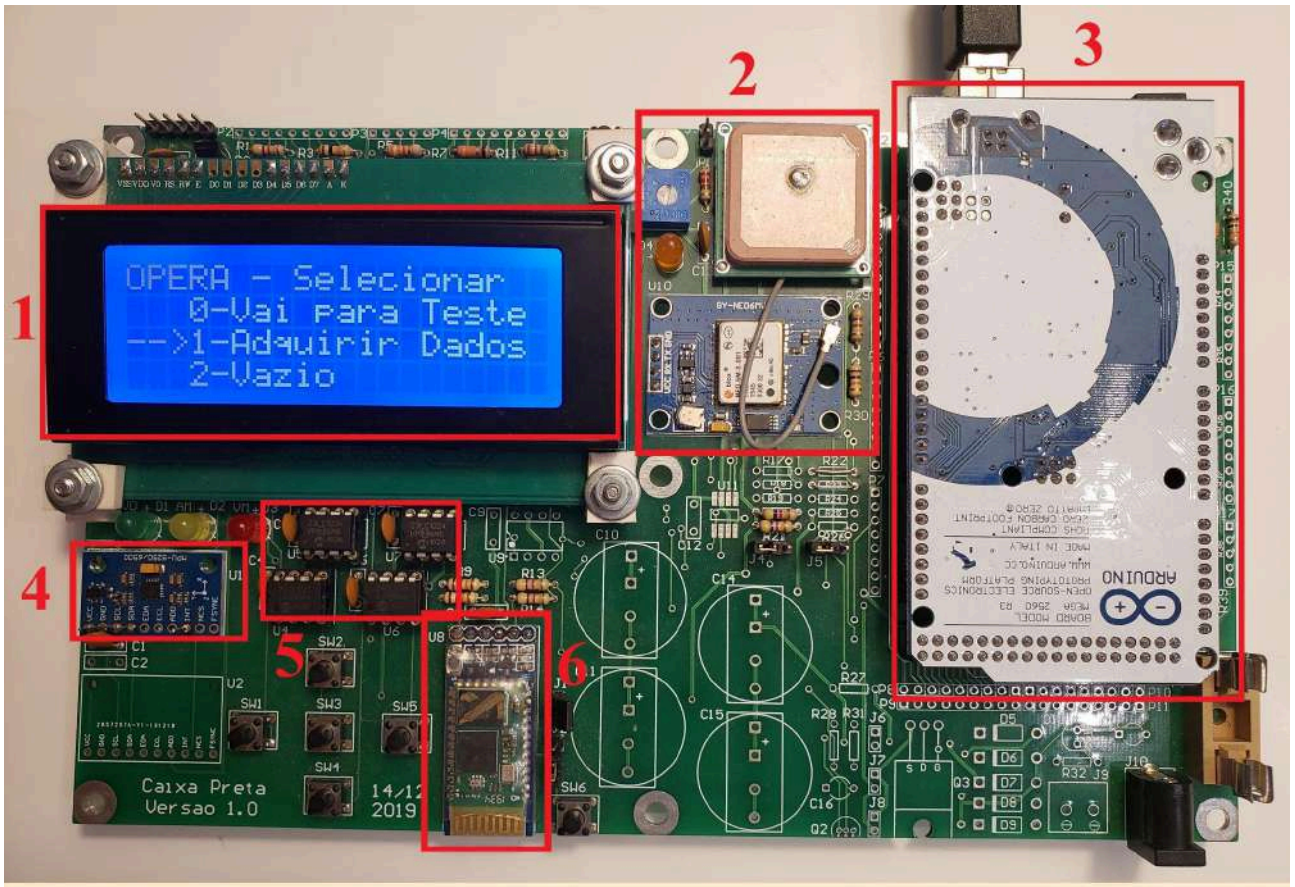


Figura 3.2: Protótipo Caixa Preta já com PCB (Versão 2, de 2019).

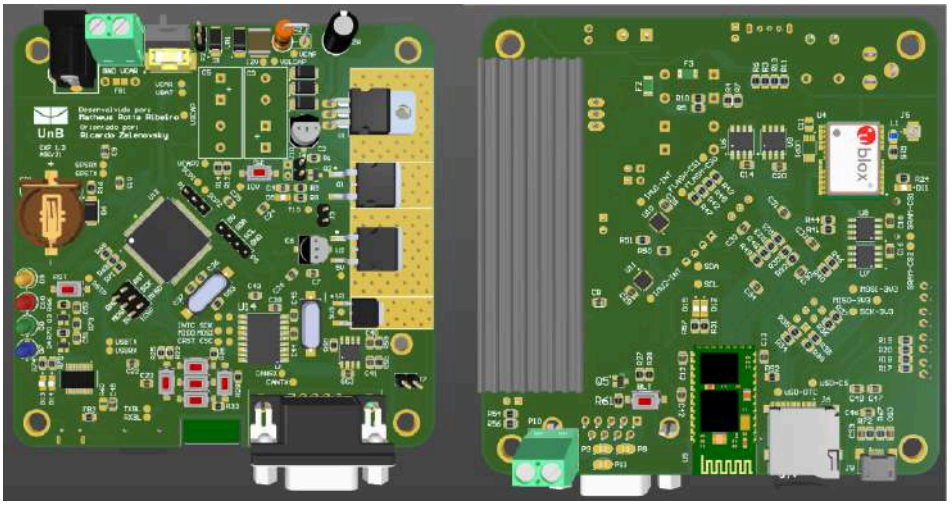


Figura 3.3: Protótipo 3D com projeto avançado, incluindo a integração no PCB de um Arduino Mega 2560 (Versão 3, de 2020).

É importante esclarecer novamente que este trabalho foi feito usando a placa versão 2, que está na Figura 3.2. Como pode ser visto nesta Figura anterior, a versão 2 da placa foi desenvolvida com dispositivos "*thru-hole*" e ainda houve a reserva de “áreas” para permitir redundância de componentes ou ensaios adicionais. Isso tudo foi previsto para facilitar medições e testes com os diversos componentes e ainda permitir ponderar as reais necessidades de cada um.

De acordo com a especificação do projeto, o uso rotineiro da Caixa Preta será via um terminal serial usando um canal Bluetooth. Porém, para a fase de desenvolvimento, é interessante ter um mostrador ágil, capaz de mostrar estados importantes do software em teste. Por isso foi incorporado um display LCD (dispositivo 1 da Figura 3.2), que permite ao programador sinalizar estados de seu programa ou apresentar valores de suas variáveis, sem a necessidade de um terminal serial. Na versão final, este mostrador não estará presente, porém seu conector permanecerá disponível.

A comunicação principal será feita via o módulo Bluetooth (dispositivo 6) e implica na necessidade de um terminal serial bluetooth que deve estar disponível num computador ou celular. Assim, usando esse terminal serial o usuário pode comandar a Caixa Preta, ler os dados armazenados e realizar todo o teste ou configuração que se façam necessários. Há a previsão do desenvolvimento de um aplicativo em Python.

Indo propriamente para os sensores que captam os dados importantes para o projeto, temos o módulo GPS (dispositivo 2) e o módulo MPU-9250 (dispositivo 4). O módulo GPS fornece a localização geográfica, a velocidade instantânea, além de informação de data e hora. O módulo MPU-9250 oferece medições em 9 eixos, sendo 3 medições para aceleração, 3 medições para o giro e 3 medições para o campo magnético.

Os dados gerados por esses módulos precisam ser tratados e gravados de uma forma compreensiva para o usuário da placa, tendo em vista que esta está sendo criada para facilitar a perícia de acidentes de trânsito. O tratamento de tais dados é feito por meio do Arduino Mega 2560, que os grava nas memórias, indicadas como dispositivo 5. São dois chips de memória SRAM 23LC1024, cada um com capacidade de 128 KB, sendo então o total de 256 KB. Os dois chips de memória EEPROM 24LC1025 fazem um total 256 KB, permitindo assim gravar de forma permanente todos os dados das SRAM.

De acordo com o projeto da Caixa Preta, durante a operação, todas as informações são gravadas de forma circular na SRAM. Isto significa que os dados recentes vão sobrescrevendo os mais antigos. Se for identificado um acidente, o sistema ainda grava dados em metade da SRAM e depois copia toda a SRAM para a EEPROM. Assim, em caso de acidente, o perito tem acesso a informações que antecederam e sucederam ao acidente. Isso é feito pois as EEPROM têm uma capacidade limitada de ciclos de escrita e apagamento, além de serem lentas quando comparadas com as SRAM. No caso específico da 24LC1025, o fabricante indica o limite de 10^6 ciclos.

3.1.1 Sustentação de Energia da Caixa Preta

Em uso rotineiro, a Caixa Preta é alimentada pelos 12V do automóvel. Toda vez que o carro é ligado ou desligado, a Caixa Preta também é ligada ou desligada. Porém, em caso de acidente, há a possibilidade de o circuito elétrico do carro ser danificado e cortar essa alimentação. Por isso, há a necessidade de se prever um sistema de backup de energia, que seja capaz de sustentar o sistema durante o tempo necessário para a coleta de dados após o acidente e ainda a cópia desses dados para a EEPROM.

Aqui será descrito este sistema de sustentação de energia. O texto aqui apresentado faz uso do artigo denominado "Estudo_01 - Arduino Embarcado em um carro", produzido e disponibilizado na Internet pelo orientador deste trabalho.

Já na versão 1 de 2017, a ambientação do projeto era embarcar a caixa preta em um veículo e alimentá-la com os 12V fornecidos pelo acendedor de cigarros. Com isso se observou a necessidade de um sistema de proteção, que pudesse manter essa voltagem dentro de limites aceitáveis, pois quando o carro estivesse ligado, ruídos do alternador e dos motores elétricos poderiam perturbar a alimentação. O desligamento do carro leva a duas situações:

a. O carro está sendo desligado de forma rotineira. Neste caso a caixa preta deve se auto-desligar.

b. O carro sofreu um acidente e a energia foi cortada. Neste caso a caixa preta precisa de 5 segundos para terminar a captura dos dados após acidente, gravar todos os dados na EEPROM e depois se auto-desligar.

O uso de baterias recarregáveis traria alguma complexidade para o projeto, pois demandaria por circuitos de carregamento e algum recurso para se verificar o tempo de vida das baterias. Por isso, a solução foi o uso de um super capacitor. Para implementação deste sistema de alimentação se considerou as seguintes especificações:

1) *Alimentar um Arduino pela tomada de 12V de um carro (acendedor de cigarro);*
2) *Garantir o funcionamento momentâneo do Arduino após o desligamento do carro;*
3) *Permitir que o Arduino se auto desligue;*
4) *Prever recursos para monitorar a alimentação e a carga do super capacitor.*

A Figura 3.4 apresenta a solução para a alimentação e cada uma das especificações acima é comentada a seguir.

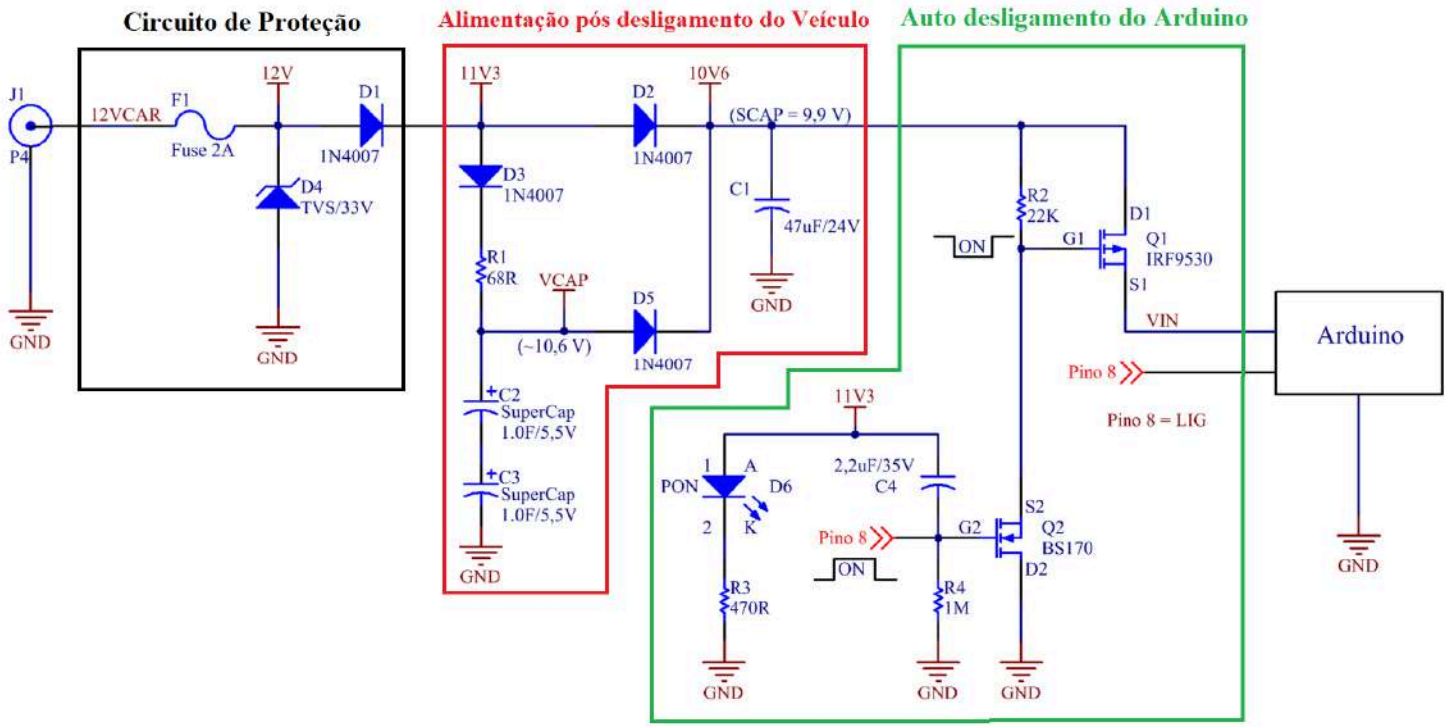


Figura 3.4: Circuitos solução dos problemas apresentados para o super capacitor.

1) *Alimentar um Arduino com os 12V de um carro (circuito de proteção)*

Como já dito anteriormente, a energia fornecida pelo acendedor de cigarros é muito ruidosa, com isso fez-se necessária a construção de um circuito protetor entre o Arduino e a voltagem fornecida pelo carro. Para garantir que não haja problemas decorrentes de picos de tensão, acrescentamos um supressor de transiente chamado diodo TVS D4 (*Transient Voltage Suppressor*), que possui 15V unidirecional, adequado para o projeto pois as tensões aqui trabalhadas não são altas nem velozes. Uma segunda preocupação é com relação aos picos negativos de tensão, que serão prevenidos com a utilização de um diodo retificador D1 (1N4007) em série com a alimentação, pois com o surgimento de picos negativos, ele entra em polarização reversa. Por fim acrescentou-se um fusível, também em série, para o caso de curto-circuito. O detalhamento da solução está demonstrado na Figura 3.4, no "Circuito de Proteção" em preto.

2) *Garantir o funcionamento momentâneo do Arduino após o desligamento do carro*

A solução imaginada para tal, foi o projeto de um super capacitor que deveria ter uma tensão máxima de 12V. No mercado nacional só foram encontrados capacitores de 1,0F/5,5V, que usados em série, no caso dois deles, chegamos a um equivalente de 0,5F/11,0V. Com a utilização do diodo retificador D1 (1N4007), mencionado na solução anterior, será fornecida ao super capacitor uma tensão de 11,3V (considerando queda de tensão de 0,7 V durante a condução). Para que a voltagem fornecida, após o circuito de proteção, ficasse abaixo do valor de 11V do super capacitor, colocou-se mais um diodo retificador D3 (1N4007), que reduziu para 10,6V a tensão fornecida ao super capacitor.

Como detalhado no circuito destacado em vermelho da Figura 3.4, foi acrescentado um resistor R1 de 68Ω, para evitar a queima do fusível, nas situações em que o super capacitor está completamente descarregado quando o carro liga e também dar um certo limite para a corrente que carrega o super capacitor, evitando valores acima de 11V.

Os diodos D3 e D5 funcionam como a carga e descarga do super capacitor. Quando o carro está fornecendo a tensão de 12V, D3 é ativado no modo de polarização direta e permite que a tensão de 10,6 V carregue o super capacitor, enquanto isso D5 fica cortado (anodo e catodo em 10.6 V), o que isola a saída do supercapacitor. Quando o carro é desligado, D3 entra em corte, por ser reversamente polarizado, enquanto D5 é diretamente polarizado e com uma queda de 0,7V de tensão, este permite a alimentação do Arduino pelo super capacitor, com uma tensão inicial de 9,9V. A tensão vai caindo em uma exponencial que tende a zero, à medida que o Arduino drena corrente do supercapacitor, como mostra a Figura 3.5 que foi obtida com o uso de um osciloscópio. Os diodos D2 e D5 trabalham de forma complementar levando adiante a tensão do carro ou a tensão do supercapacitor. Além disso, o diodo D2 está posicionado para limitar a variação de tensão quando o super capacitor é usado como alimentação. A tensão entregue ao Arduino é, aproximadamente, de 9,9V, pouco importando se ela foi gerada pelos 12V do carro ou pelo supercapacitor. O capacitor eletrolítico C1 de 47µF/24V ajuda a filtrar variações da tensão entregue ao Arduino.

Com relação ao teste feito para verificar quanto tempo o super capacitor é capaz de sustentar a alimentação do Arduino, mostrado na Figura 3.5, é necessário pontuar que foram usados dois canais pelo osciloscópio, um deles conectado à tensão do capacitor (VCAP), em azul (canal A), e o outro canal conectado ao pino 13 do Arduino, que está ligado a um led, em vermelho (canal B), o que permitiu determinar o exato instante em que a CPU parou de funcionar. O ensaio foi feito fornecendo por um longo período, a tensão de 12V ao circuito, e em seguida desligando-a. Neste instante há uma queda abrupta em VCAP, e depois um decaimento exponencial. O limite para o fornecimento de tensão é até 5,4V, após esse valor o Arduíno apresenta um comportamento errático que poderia

prejudicar o resto do circuito. Para evitar isso, usou-se a tensão limite de 6V, e como o ensaio foi feito sem o circuito de auto desligamento do Arduíno, há tempo suficiente para este usar do super capacitor e se desligar. Se conseguiu alimentação por um tempo de 15 segundos, bem superior ao mínimo desejado que era de 5 segundos.

Um breve parêntese sobre o comportamento errático. Este comportamento se dá pelo fato do Arduíno possuir um reset forçado por *Brownout*, ou seja, é o reset por falta de energia. Acontece que quando o Arduino entra em reset, ele para de drenar energia do supercapacitor e por isso a tensão fornecida por este dispositivo aumenta um pouco (perto de 5,7V). Este pequeno aumento na tensão do supercapacitor é suficiente para cancelar o reset por Burnout e o Arduino volta a funcionar provocando a queda na tensão do supercapacitor e entrando novamente em reset por Burnout. Isto se repete várias vezes até o supercapacitor ser descarregado e cair para 5,1V.

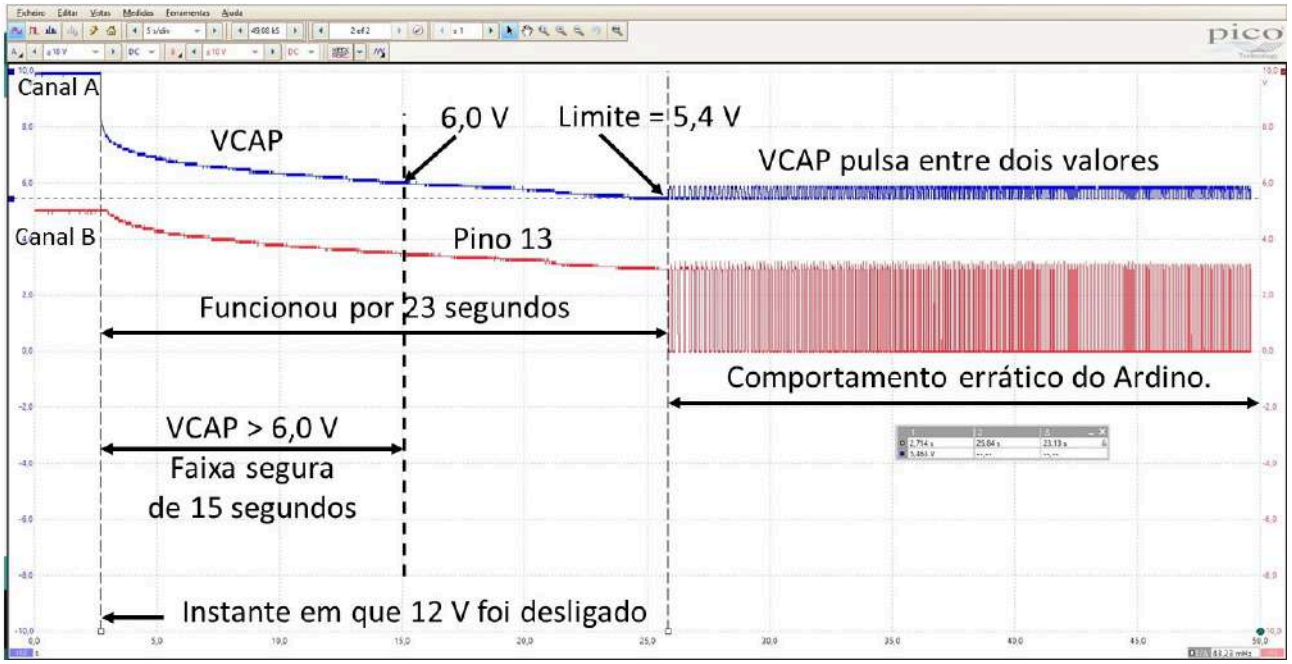


Figura 3.5: Ensaio para verificar quanto tempo o super capacitor é capaz de sustentar a alimentação do Arduino.

3) *Permitir que o Arduino se auto desligue*

Além de desenvolvermos um circuito que permita o Arduino se auto desligar, este precisa ser ligado novamente quando o carro voltar a funcionar, ou seja, deve ser energizado sempre que isso acontecer. Na Figura 3.4 pode-se ver o componente Q1, o MOSFET IRF9530, que aqui é o responsável pelo auto desligamento. Sendo um MOSFET do tipo P, está posicionado entre a alimentação e o Arduino, que seria a carga. Entre a porta (G1) e o dreno (D1) foi usado o resistor R2 de 22kΩ, para evitar disparos indevidos quando a porta estiver desconectada, e funciona como resistor de pullup, colocando G1 em nível alto, quando a porta não é acionada. Controlando Q1, temos o MOSFET Q2, de tipo N, portando posicionado entre a carga e o terra, com seu resistor R4 de 1MΩ funcionando como pulldown entre G2 e D2. Conectado à porta de Q2 temos o pino 8 do Arduino, que pode ligar ou desligar a alimentação de acordo com o seguinte esquema:

- Pino 8 = HIGH → Q2 satura → G1 = LOW → Q1 satura e alimenta o Arduino e
- Pino 8 = LOW → Q2 corta → G1 = HIGH (ação de R2) → Q1 corta e desliga o Arduino.

Apesar de tudo isso, é importante lembrar que quando todo o sistema estiver desenergizado, e o carro ligar, é necessário algo que faça com que a porta do MOSFET Q2 seja colocada em nível alto

de forma temporária, pois o pino 8 se encontrará desligado neste momento. Para tal, recorreu-se a um circuito auxiliar composto por um capacitor C4 de 2,2uF, e o resistor R4 de 1MΩ. Como C4 está conectado a tensão de 11,3V, assim que o carro liga, ele curto-circuita essa alimentação a G2, que satura Q2 e em seguida Q1, ligando o Arduino. Com o decorrer do tempo, a tensão em G2 vai caindo com a carga de C4. Assim, a primeira ação do programa é colocar o pino 8 em nível alto e garantir a alimentação, independente da carga do capacitor.

Ao ser ligado, o Arduino faz sua inicialização e entrega o controle ao programa do usuário após 0,9s. Estando C4 completamente descarregado quando o carro é ligado, o capacitor demora 3,8s carregando e mantendo Q2 em condução. Quando a tensão em G2 chega em 2 Volts o transistor Q2 corta e não mais alimenta o Arduino. Esses tempos estão ilustrados na Figura 3.6. Então, se C4 estiver completamente descarregado, o intervalo de tempo para o programa se ligar é mais que o suficiente. Se, no entanto, C4 já possuir alguma carga (carro ligando e desligando de forma rápida), esse tempo pode ser diminuído, então colocou-se um led D6 e um resistor R3 de 470Ω, para descarregar o capacitor de forma bem rápida, o que não se conseguia antes por conta do valor alto de R4. Com a Figura 3.7, pode-se notar que poderíamos ter dispensado o led D6, porém este também funciona como um informativo de que o carro está ligado, PON (Power On), pois o próprio pino 8, quando o Arduino está desligado, permite a descarrega de C4.

Figura 3.6: Tensão em G2 e em C4, quando se liga o carro (simulação).

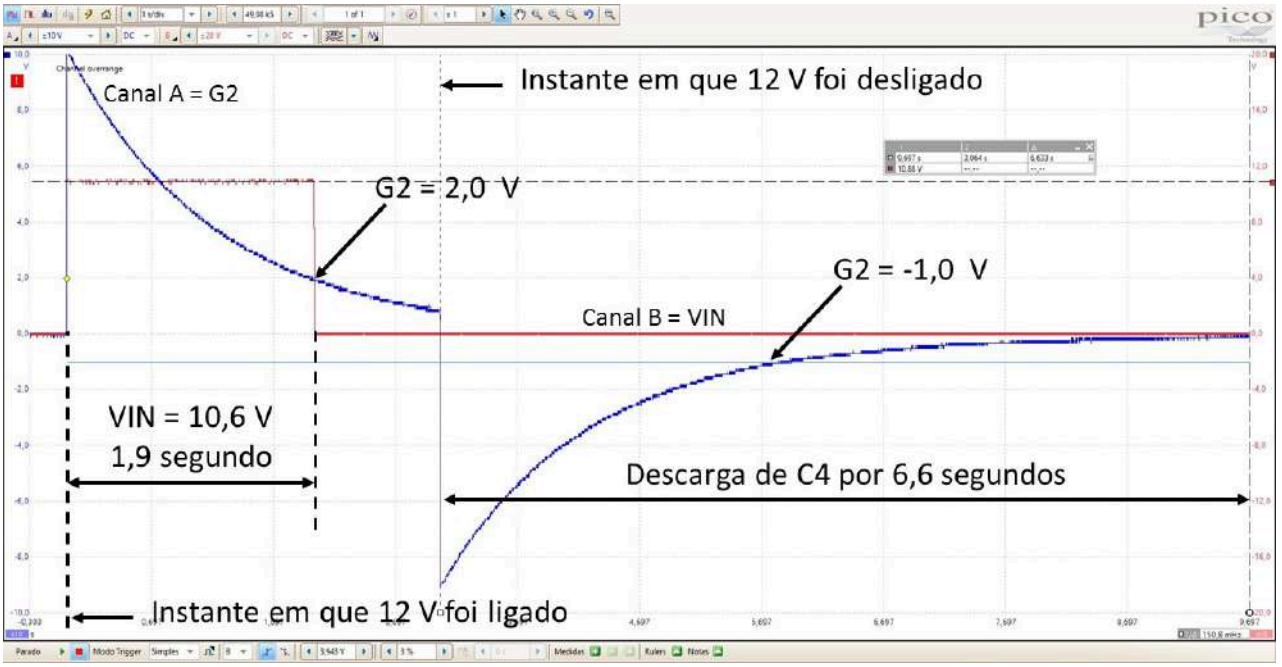


Figura 3.7: Ensaio de ligar e desligar a tensão de 12V para verificar o comportamento de G2 e C4 (Vin é a tensão entregue ao Arduino).

A Figura 3.7 nos mostra um dos ensaios de ligar e desligar a tensão de 12V para verificar o comportamento de G2 e os tempos de carga e descarga de C4, sendo a linha azul (Canal A) posicionada em G2, e alinha vermelha (Canal B) indica a tensão de alimentação (Vin) do Arduino. Na primeira parte, à esquerda, vemos o tempo de 1,9 segundos de funcionamento do Arduino, antes da tensão ser cortada pela carga de C4 que leva ao corte de Q2 e Q1 (Q2 corta quando a tensão em G2 cai abaixo de 2 V). Este é um tempo que representa metade do que foi calculado anteriormente (3,8s, vide Figura 3.6), porém ainda é suficiente para inicializar o Arduino. Pois o programa do usuário assume o controle 0,9 segundos após a energização. Quando a alimentação de 12V é cortada, C4 passa a descarregar e por isso surge a tensão negativa. Pode-se ver que após 3s, a tensão em C4 está em 1V, e que após 6,6s ele já está praticamente descarregado.

Como foi dito anteriormente, o controle de Q2 e a consequente alimentação do Arduino é realizada pelo pino 8, que precisa ser colocado em nível lógico alto, quando o programa do usuário inicia. O ensaio, cuja listagem está abaixo, foi feito para verificar o comportamento ao ligar e desligar o Arduino. Os resultados estão na Figura 3.8. Assim que a alimentação de 12V é ligada, o programa (após 0,9s) assume o controle e coloca o pinto 8 em nível alto, o que garante a alimentação do Arduino. Depois de 5 segundos, por ação do programa, este pino é colocado em nível baixo e o circuito se auto desliga. Depois de algum tempo, a tensão de 12V é novamente aplicada e o ensaio se repete.

```
void setup() {
        pinMode(8,OUTPUT);          //Pino 8 como saída
        digitalWrite(8,HIGH);       //Pino 8 = HIGH ==> Ligar Q2
        pinMode(13,OUTPUT);         //Pino 13 como saída
        digitalWrite(13,HIGH);      //Acender led da placa
        delay(5000);                //Esperar 5 segundos
        digitalWrite(8,LOW);        //Pino 8 = LOW ==> Desligar Q2
        while(1);                   //Laço de segurança
        }
void loop() {}
```

Na Figura 3.8, a linha em azul (canal A) indica a tensão na porta G2, e a linha em vermelho (canal B), indica a tensão de alimentação entregue ao Arduino. Assim que o programa do usuário começa a rodar, o pino 8 é acionado (pino 8 = H) antes da curva da tensão em G2 chegar a 2V. Depois de 5 segundos, o Arduino se auto desliga (pino 8 = L). Depois de algum tempo a alimentação de 12V é desligada e depois novamente ligada e o Arduino volta a funcionar se auto desligando novamente após 5 segundos. É de se notar que o capacitor C4 se descarregou rapidamente pelo pino 8 do Arduino.

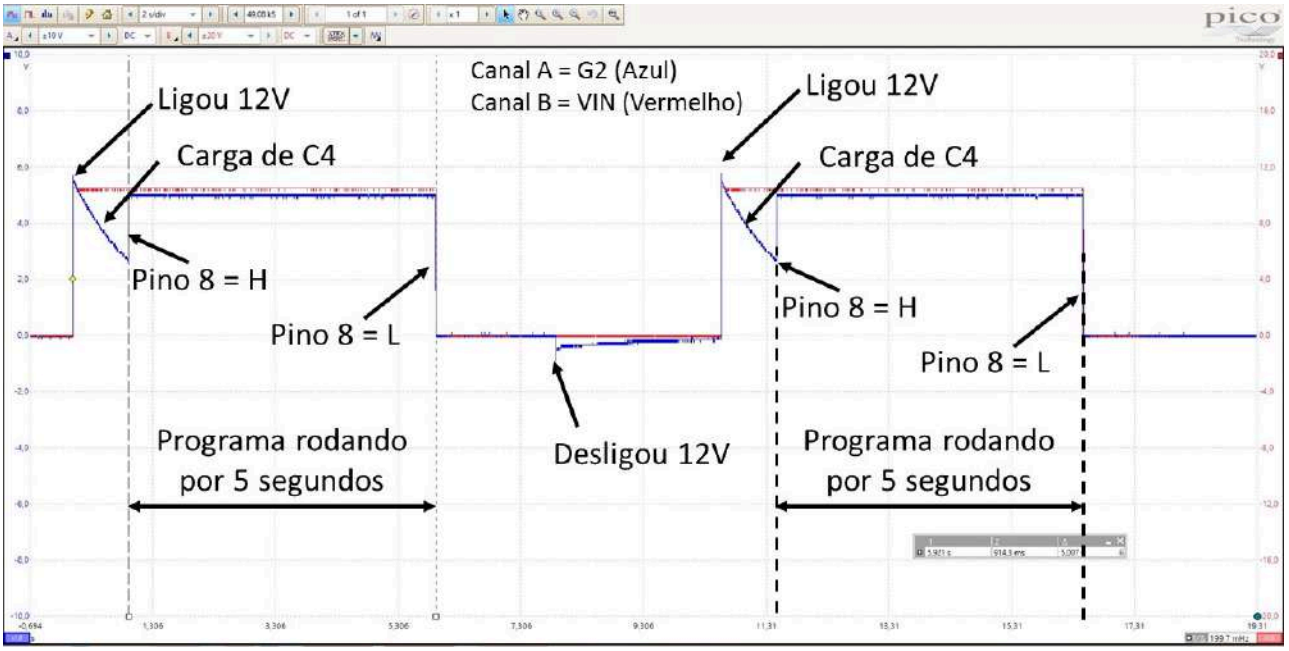


Figura 3.8 Verificação da partida automática com a presença da alimentação de 12V e o funcionamento do auto desligamento.

4) *Prever recursos para monitorar a alimentação e a carga do super capacitor*

Para o monitoramento das tensões de alimentações do carro e do super capacitor, foram empregados divisores resistivos, como mostrado na Figura 3.9. Esses divisores vão gerar diversas tensões:

VCAR indica se o carro está ligado;

VCAR-INT gera uma interrupção (pelo flanco de descida) quando o carro é desligado;

VCAP indica a tensão sobre o super capacitor e

VCAP-COMP faz uso do comparador analógico do Arduino para indicar que a tensão no super capacitor chegou a um nível crítico.

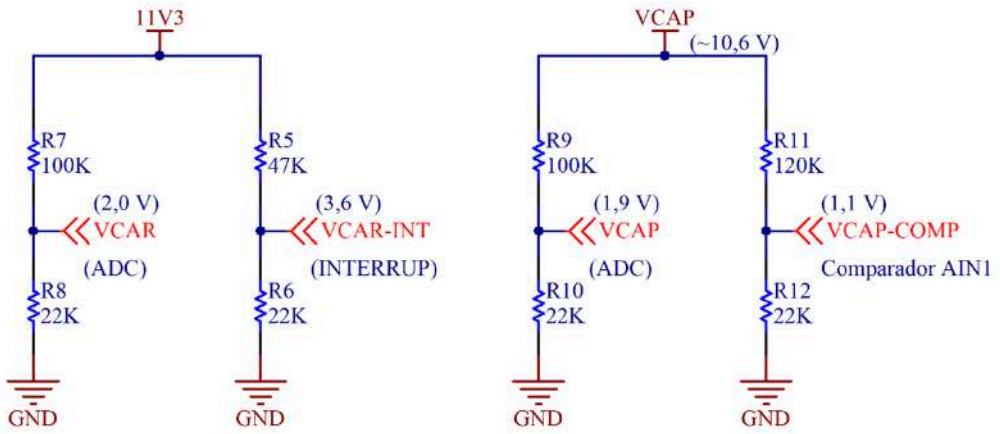


Figura 3.9: Circuitos para monitoramento da alimentação e geração de alertas.

• Informar se o carro está ligado (se a alimentação de 12V está presente)

Aqui são utilizados os resistores R7 e R8, com sua saída VCAR (tensão do carro) conectada a uma das entradas do conversor analógico digital do Arduino (ADC). Esses resistores foram calculados para gerar uma tensão de 2 V quando o carro está ligado. O cálculo do divisor se encontra logo abaixo.

$$VCAR = 11,3(R8/(R7+R8))) = 11,3(22k/(100k+22k)) = 2V \rightarrow ADC$$

• Alertar quando o carro for desligado

Conectando a saída VCAR_INT, divisor resistivo de R5 e R6, a uma das entradas de interrupção por flanco de descida do Arduino, pode-se saber quando o carro é desligado. Com a presença dos 12 V, este divisor gera uma de 3,6 V, o que é suficiente para manter nível lógico alto. Quando o carro é desligado, a saída deste divisor vai para nível baixo e provoca uma interrupção. O cálculo do divisor se encontra logo abaixo.

$$VCAR\_INT = 11,3(R6/(R5+R6))) = 11,3(22k/(47k+22k)) = 3,6V \rightarrow Interrupção$$

• Medir a tensão atual sobre o super capacitor

Assim como foi feito para saber se o carro estava ligado, também é necessário conhecer a tensão que está chegando no super capacitor. Este divisor foi calculado para que na presença de 10,6 V, sua saída esteja em 1,9 V. É claro que deve ser usado o conversor AD do Arduino para acompanhar esta tensão. O cálculo do divisor se encontra logo abaixo.

$$VCAP = 10,6(R10/(R9+R10))) = 10,6(22k/(100k+22k)) = 1,9V \rightarrow ADC$$

• Indicar quando a tensão sobre o super capacitor cair abaixo de um limite

Quando o circuito é alimentado pelo super capacitor, é importante saber quando este está chegando ao seu limite. Como já vimos na Figura 3.5, este limite é de 5,4 V. O comparador analógico do Arduino pode gerar uma interrupção quando a tensão na sua entrada cair abaixo de 1,1 V. Para deixar uma certa folga e considerando os valores comerciais, o divisor resistivo foi calculado para chegar a 1,1 V quando a tensão sobre o super capacitor for de 7,1 V. Em outras palavras, quando a tensão no super capacitor cair para 7.1 V, o comparador provoca uma interrupção de alerta (pouca bateria). O cálculo do divisor se encontra logo abaixo.

$$VCAP\_COMP = 7,1(R12/(R11+R12))) = 7,1(22k/(120k+22k)) = 1,1V \rightarrow Comparador$$

3.1.2 Solução para os Botões

O conteúdo aqui explanado também foi baseado no "Estudo_02 - Pequeno teclado usando apenas um pino do Arduino", produzido e disponibilizado pelo orientador deste trabalho.

Para evitar o uso de uma grande quantidade de pinos do Arduino, a proposta foi por um divisor resistivo controlado por chaves, cuja saída é lida pelo ADC do Arduino. A primeira ideia surgiu com o emprego de uma série de resistores idênticos, como mostrado na Figura 3.10.

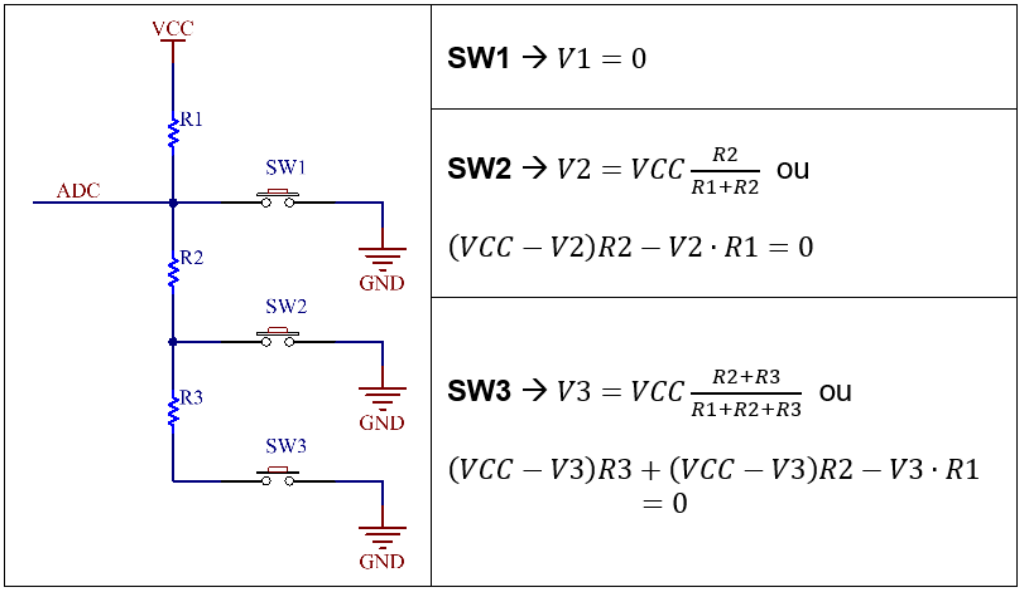


Figura 3.10: Solução para a conexão de 3 chaves ao ADC do Arduino. A sigla V1, representa a tensão no ADC quando a tecla 1 é acionada, e assim por diante.

Nesta solução, por simplicidade, vamos sugerir o uso de resistores idênticos, com valor nominal de 10 kΩ. Se VCC =5,0 V, é fácil calcular os valores para V1, V2 e V3, como mostrado na Figura 3.11. Podemos ver que a proposta deve funcionar, porém as tensões V2 e V3 estão "próximas". Seria possível conseguir uma melhor separação, aproveitando a excursão de 0,0 V até 5,0 V.

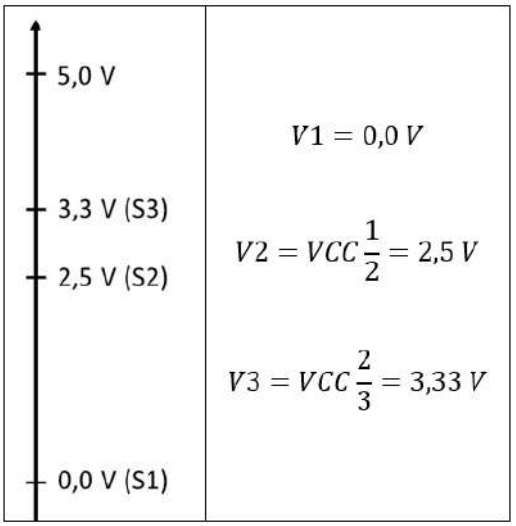


Figura 3.11: Gráfico mostrando a tensão gerada pelo acionamento de cada uma das 3 chaves.

Esta solução com resistores idênticos é simples, porém a margem de segurança entre as teclas vai diminuindo quando se aumenta sua quantidade. A Figura 3.12 apresenta as tensões geradas para o caso de se empregar 7 chaves. É possível ver que as tensões geradas pelo acionamento das chaves S5, S6 e S7 estão muito próximas.

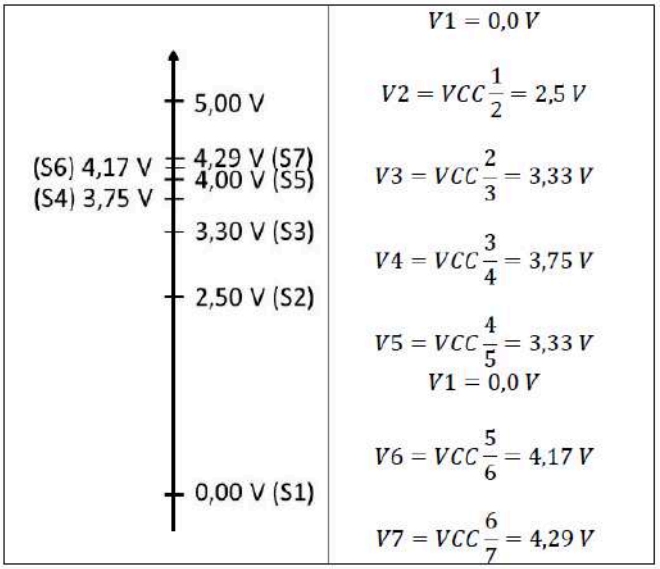


Figura 3.12: Tensões geradas nos acionamentos de cada chave para o caso de 7 chaves e resistores idênticos.

Para solucionar o problema das chaves vamos propor um divisor resistivo que permite o uso linear da faixa de 0 a 5 V , gerando tensões que sejam facilmente identificadas com o ADC operando em 8 bits. Assim, a faixa de tensão de 0 até 5V será dividida em 8 faixas, mostradas na Figura 3.13. O acionamento de cada chave deve gerar a tensão central de cada faixa, como mostrado abaixo. A chave SW1, quando acionada gera a tensão V1, a chave SW2, quando acionada gera a tensão V2 e assim por diante. A tensão V8 é gerada quando nenhuma das chaves é acionada.

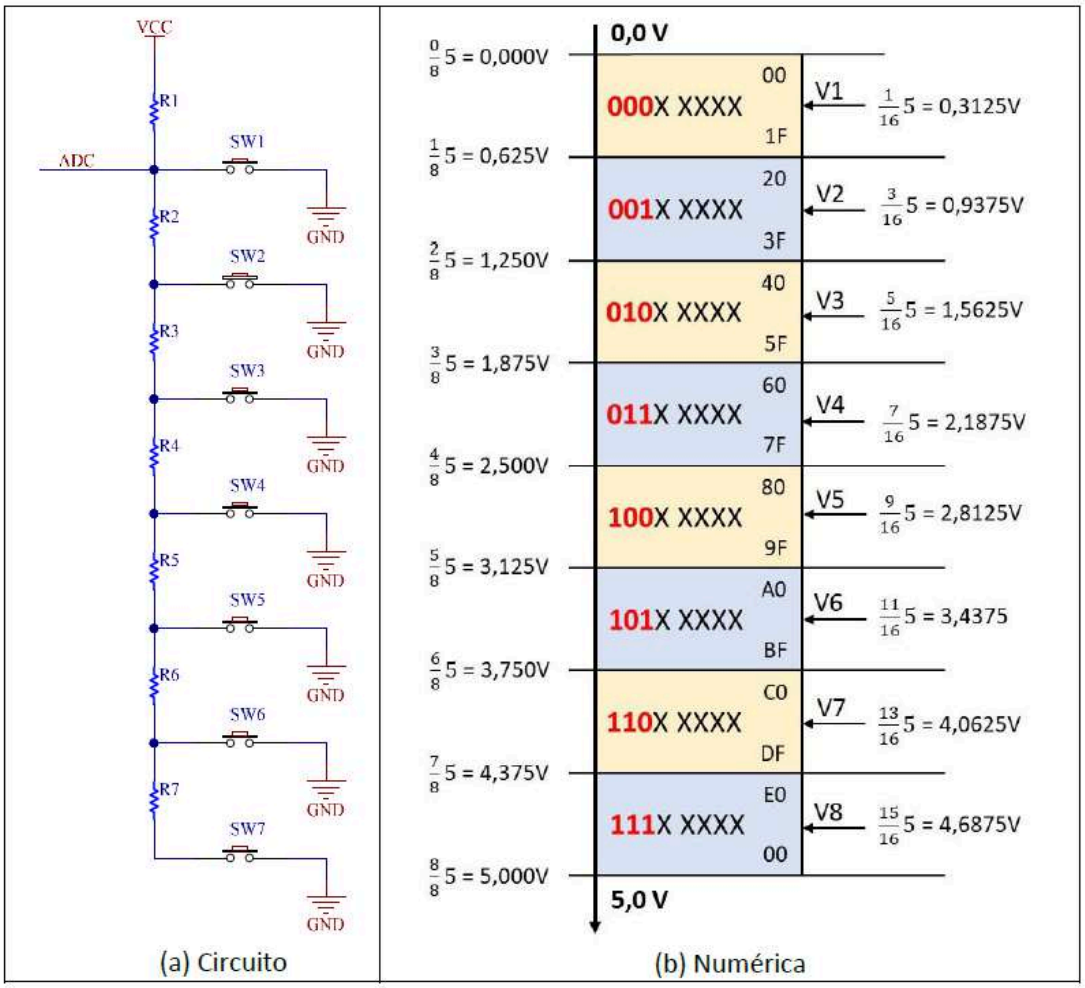


Figura 3.13:Tensões no ADC ao acionar uma das 7 chaves. A chave SWi quando acionada deve produzir a tensão Vi (SW1, quando acionada gera a tensão V1 e assim por diante).

Com essas especificações, o valor de cada resistor pode ser calculado com um sistema de 7 equações e 7 incógnitas, como mostrado abaixo.

$$V1 = 0$$
$$V2 = VCC\frac{R2}{R1+R2} \rightarrow (VCC - V2)R2 - V2 \cdot R1 = 0$$
$$V3 = VCC\frac{R2+R3}{R1+R2+R3} \rightarrow (VCC - V3)R3 + (VCC - V3)R2 - V3 \cdot R1 = 0$$
$$V4 = VCC\frac{R2+R3+R4}{R1+R2+R3+R4} \rightarrow (VCC - V4)R4 + (VCC - V4)R3 + (VCC - V4)R2 - V4 \cdot R1 = 0$$

…

Sob a forma matricial, tem-se o equacionamento abaixo, onde R1 foi arbitrado em 10 kΩ.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & VCC-V2 & -V2 \\ 0 & 0 & 0 & 0 & VCC-V3 & VCC-V3 & -V3 \\ 0 & 0 & 0 & VCC-V4 & VCC-V4 & VCC-V4 & -V4 \\ 0 & 0 & Vcc-V5 & Vcc-V5 & Vcc-V5 & Vcc-V5 & -V5 \\ 0 & Vcc-V6 & Vcc-V6 & Vcc-V6 & Vcc-V6 & Vcc-V6 & -V6 \\ Vcc-V7 & Vcc-V7 & Vcc-V7 & Vcc-V7 & Vcc-V7 & Vcc-V7 & -V7 \end{bmatrix} \cdot \begin{bmatrix} R7 \\ R6 \\ R5 \\ R4 \\ R3 \\ R2 \\ R1 \end{bmatrix} = \begin{bmatrix} 10K \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

A solução resulta na tabela abaixo, como também se apresentam os valores comerciais mais próximos.

	Calculados	Comerciais
R1	10.000,00	10.000
R2	2.307,69	2.400
R3	2.237,76	2.200
R4	3.232,32	3.300
R5	5.079,37	5.100
R6	9.142,86	9.100
R7	21.333,33	22.000

A tabela a seguir apresenta uma comparação entre as tensões geradas pelos resistores com os valores ideais e com as tensões obtidas quando se usam valores comerciais. A coluna “V” indica a tensão na saída do divisor resistivo. A coluna “Hexa” indica o valor retornado pelo ADC operando em 8 bits, convertendo a tensão gerada pelas chaves. A coluna “3 Bits” apresenta apenas os 3 bits mais significativos do código gerado pelo ADC. Pode-se ver que as chaves são facilmente identificadas por esses 3 bits. Em suma, com o operando em 8 bits, basta separar os 3 bits mais significativos e já se tem o código da chave acionada.

	Valores ideais				Valores comerciais				
	R	V	Hexa		R	V	Hexa		3 Bits
1	10.000,00	0,0	0		10.000	0,0	00		000
2	2.307,69	0,9375	30		2.400	0,9677	32		001
3	2.237,76	1,5625	50		2.200	1,5753	51		010
4	3.232,32	2,1875	70		3.300	2,2067	71		011
5	5.079,37	2,8125	90		5.100	2,8261	91		100
6	9.142,86	3,4375	B0		9.100	3,4424	B0		101
7	21.333,33	4,0625	D0		22.000	4,0758	D1		110

É interessante comentar que tal solução facilitou muito o emprego dos botões. Foram usados apenas 5 botões, organizados em uma cruz, como mostrado abaixo, sendo que na porção à direita estão os nomes usados no programa. Eles podem ser identificados na Figura 3.2. Assim, ficou fácil padronizar a interação do usuário com a Caixa Preta.

	↑				sw_cima	
←	Seleção	→		sw_esq	sw_sel	sw_dir
	↓				sw_baixo	

Figura 3.14:Ilustração da organização dos 5 botões (teclas) usados na Caixa Preta.

A cada 10ms o teclado é lido (com o uso do ADC) e se o mesmo valor se repete após 5 leituras o teclado é considerado estável (sem rebotes). Quando o valor correspondente a uma tecla é lido, ao mesmo tempo que o valor anterior a este for de nenhuma tecla acionada, então se aceita a nova tecla e seu código (valor) é colocado na fila do teclado. O programa principal, de tempos em tempos, consulta a fila do teclado para ver se alguma ação foi realizada.

3.2. Software

O código foi construído em linguagem de programação C++, utilizando a plataforma "Arduino 1.8.19 (Windows Store 1.8.57.0)" disponível no site da empresa em "Documentation Arduino Mega 2560".[3] O primeiro núcleo do código foi desenvolvido pela equipe inicial e depois, com os trabalhos em sequência, este núcleo foi sendo expandido e aperfeiçoado de acordo com as necessidades específicas de cada trabalho. Cada equipe documentou a parte por ela desenvolvida e uma biblioteca e sua documentação foi centralizada pelo orientador.

Neste tópico iremos apresentar brevemente as porções da biblioteca que foram desenvolvidas e aperfeiçoadas com este trabalho. Serão detalhadas em especial as funções de programação atribuídas ao GPS e explicando como são feitos alguns ensaios na placa. Com relação aos arquivos da biblioteca que foram úteis ao presente trabalho, temos:

CXP-21-9: inicia os códigos das outras funções abaixo, executadas em laço **Opera** e **Teste**;

Teste: pacote com diversas rotinas para testar o hardware da Caixa Preta;

Opera: operação da Caixa Preta para atingir os objetivos deste trabalho. Será detalhado na seção 4, onde se indica como são feitos os ensaios;

Defs: pasta com todas as constantes (#defines) usadas na CXP;

Globs: declaração de todas as variáveis globais;

MPU: conjunto de funções para configurar e acessar a plataforma inercial (MPU-9250);

Flash: conjunto de funções para ler e escrever na memória 24LC1025 via porta I2C e contém também o conjunto de funções para acessar o barramento I2C;

SRAM: conjunto de funções para ler e escrever na memória 23LC1024 via porta SPI (*Serial Peripheral Interface*);

EEPROM: conjunto de funções para ler e escrever na memória EEPROM interna da CPU do Arduino Mega;

GPS: pacote de funções para acessar o GPS e separar as informações necessárias;

TWI: (*Two Wire Interface*) pacote básico de funções para acessar a porta SPI;

Strings: biblioteca que gera strings de texto para serem impressas no LCD ou na porta serial;

LCD: biblioteca para acesso ao display LCD (20 x 4);

Serial: biblioteca para acesso às duas portas seriais (UART e Bluetooth);

Timers: rotinas que configuram e facilitam o uso da base de tempo da CXP;

Tec_Led: rotinas para acessar as teclas e controlar os leds.

Aleat: Funções para gerar números pseudo-aleatórios, demarcadas pela sigla "rrand", definidos no código.

CXP-21-9

O pacote CXP-21-9 contém a função de inicialização e o laço principal, que no Arduino são denominados de setup(){...} e loop(){...}. Aqui é feita a inicialização de todos os periféricos: GPS, comunicação serial, MPU, botões e leds, timers, LCD etc. Após isso, o programa oferece duas opções ao usuário: **Opera e Teste**. O **Modo Opera** é usado para ensaiar versões da CXP. Ele admite diversas opções e com isso, cada um dos alunos trabalhou e ensaiou suas versões de software, sem interferir com as demais. O **Modo Teste** permite selecionar opções para testar e ensaiar individualmente todos os dispositivos usados na CXP. Importante lembrar que o programa Arduino 1.8.19, exige que o nome da pasta que engloba todas as abas de funções deve ser o mesmo desta apresentada (CXP-21-9).

Opera

Quanto ao Modo Opera (na função opera()), o usuário tem a possibilidade de selecionar sua versão de operação ou um dos modos de operação previamente preparados. A listagem abaixo dá uma ideia das opções que podem estar disponíveis. Assim, cada equipe pode escolher uma opção que esteja vazia e ensaiar suas rotinas para operar a CXP. Essas opções podem ser selecionadas apenas com os botões e o LCD, ou então, de forma mais confortável usando uma das portas seriais (UART ou Bluetooth).

```
// OPERA - Mensagens do modo de Operação

char *opera_msg[]={ "0-Vai para Teste",      //0 -  Ir para o Modo de Teste
                    "1-Adquirir Dados",     //1 – Versão em teste
                    "2-Vazio",              //2
                    "3-vazio",              //3
                    "4-Vazio",              //4
                    "5-Calibra Fab",        //5 – Calibração de fábrica
                    "6-Calibra Mag",        //6 – Calibração do Magnetômetro
                    "7-Vazio",              //7
                    "8-Vazio",              //8
                    "9-Vazio"};              //9
```

A opção 5, denominada de Calibração de Fábrica, grava parâmetros importantes para o funcionamento da CXP e que o usuário não terá acesso. Ela é feita uma única vez, armazenando os dados na EEPROM (4 KB) interna da CPU do Arduino. Ela realiza as seguintes operações:

• Armazena os dados de aceleração de gravidade, padrão e do local onde foi feita a calibragem;
• Espera um longo tempo para o MPU configurar e depois é calcula a média de uma grande quantidade de medidas para determinar o erro intrínseco de cada eixo;
• Realiza o Self Test e armazena todos os resultados

A listagem abaixo apresenta alguns parâmetros que são armazenados nesta EEPROM.

```
// CONFIGURAÇÃO DE FÁBRICA
//Aceleração da Gravidade
#define G_PADRAO  9.80665     //1g padrão
#define G_BSB     9.7808439   //Ac. gravidade em Brasília
const char *CF_HOJE = "20/04/20";  //Data para Configuração de Fábrica
const char *CF_BSB = "Brasilia";   //Data para Configuração de Fábrica
```

A opção 6, a Calibração do Magnetômetro, por enquanto, precisa de um auxílio do Matlab. Tal calibração é necessária para remover ou compensar distorções vindas de materiais que não geram seu próprio campo magnético (distorções *soft-iron*), e aquelas advindas de objetos que geram seu próprio campo magnético (distorções *hard-iron*). [22]. No programa usado neste trabalho, tal calibração não foi desenvolvida a ponto de funcionar de forma correta, porém há uma melhor dissertação com testes e propostas para o funcionamento desta, no projeto do aluno Paulo B. Teixeira Neto.

Com a opção 1, Adquirir Dados, foram feitos os ensaios deste trabalho, onde se simulou o comportamento da CXP, como se estivesse conectada a bateria de um carro e ocorresse um acidente. Seria o laço principal da CXP, caso a placa estivesse finalizada e comercializada. Ao entrar neste modo, são rodadas todas as rotinas de inicialização e em seguida é feito o *self test* do MPU (definido no manual do MPU-9250), sucedido do zeramento de todas as SRAM. A partir deste momento, a caixa preta vai recebendo dados do MPU (100 leituras por segundo) e do GPS (um dado de localização por segundo), e os armazena na memória SRAM a partir do endereço zero.

O acionamento do botão central SW3 (SW_SEL) simula o instante de um acidente. Quando isto é feito, há a sinalização de um acidente e a aquisição continua por mais um período (programável). Se durante este “período pós acidente” o botão for novamente acionado, tudo é

reiniciado. Decorrido o período programado, todos os dados são enviados pelas duas portas seriais (UART e Bluetooth). São enviadas primeiro as informações da plataforma inercial e depois as informações do GPS. Esses dados podem ser armazenados em um arquivo para processamento. Algumas versões do software da CXP enviam esses dados diretamente para o Matlab para processamento e análise.

Abaixo está uma representação dos dados enviados pela CXP após um acidente, onde temos os marcadores:

#[m ..... m]# → início e fim dos dados do MPU;

#[g ..... g]# → início e fim dos dados do GPS;

#[l ..... l]# → início e fim dos dados relativos à calibração local (realizada quando a CXP é ligada);

#[f ..... f]# → início e fim dos dados relativos à calibração de fábrica (realizado uma única vez);

```
#[m
290720 = data
181733 = hora
2402 = temperatura
20046 = self test acel e giro
20046 = self test mag
5040 = ponteiro MPU na hora da batida
229888 = ponteiro GPS na hora da batida
+21331 +20046 +2 +250 +2111 +544 +232 +17905 +124 +112 +62
+21331 +181 +181 +183 +171 -100 +2005 -680 +1940 +3255 +3230
+20046 +20046 +2 +250 +100 +5 +1  +2  +3  +4  +5
259 = quantidade de linhas
00000 00003C30 00808 01056 13060 65368 00104 65446 65535 65535 65535
00001 00003C42 00816 01028 13216 65325 00116 65444 65535 65535 65535
00002 00003C54 00716 01032 13032 65295 00122 65445 65535 65535 65535
...
12717 00003BFA 00584 00988 13172 65193 00133 65477 65535 65535 65535
12718 00003C0C 00604 00972 13252 65194 00134 65480 65535 65535 65535
12719 00003C1E 22222 22222 22222 22222 22222 22222 65535 65535 65535
m]#
#[g
290720 = data
181733 = hora
229888 = ponteiro GPS na hora da batida
256    = quantidade de linhas
0000000000 +0000246784 A 21/05/20 12:35:24 1548.63020S 04748.65727W 1.203K 0.649
1065.9M [04 6.54 3.24 5.68] 15246
...
0000000255 +0000246656 A 21/05/20 12:35:23 1548.63031S 04748.65770W 0.687K 0.371
1066.1M [06 6.54 3.24 5.68] 13464
g]#
#[l
21331
20046
…
123316.00
l]#
#[f
+21331
20/04/20
...
-00006 -00018 -00004 +00180 -00543 +00188
f]#
```

A memória SRAM tem espaço para armazenar 12.720 registros da plataforma inercial. Se esta opera na taxa de 100 leituras por segundo, há, portanto, espaço para se gravar 127 segundos de dados. Podendo ser 64 segundos antes do acidente e 63 segundos após o acidente. Não necessariamente todo este espaço precisa ser ocupado a cada acidente.

A tabela abaixo indica o mapeamento da memória (SRAM ou EEPROM) onde se vê que há espaço separado para o MPU e para o GPS. Isto porque os formatos desses dados são bem distintos.

Mapa da SRAM:

Finalidade	Faixa Hexa	Bytes	Qtd msg	Tempo
MPU	0 0000 → 3 7E5F	228.960	12.720	127 seg
Configuração	3 7E60 → 3 7FFF	416	-	-
GPS	3 8000 → 3 FFFF	32.768	256	256 seg

Teste

Abaixo estão as opções para o modo de teste. É apresentada apenas uma lista com os nomes para que se possa ter uma pequena noção do que está disponível. Esses nomes são auto explicativos e como não foram o foco do presente trabalho, eles não serão detalhados.

1-LEDs → verificar o funcionamento dos leds;
2-LCD → Escrever mensagens no LCD para comprovar seu funcionamento;
3-Teclado → comprovar o funcionamento dos 5 botões da CXP;
4-TWI (I2C) → Testes com o I2C;
5-Acel e giro → Ensaios e testes com acelerômetro e giroscópio;
6-Magnetômetro → Ensaios e testes com o magnetômetro;
7-SRAM → Testes de leitura e escrita;
8-FLASH → Testes de leitura e escrita;
9-GPS: Tudo → Testes de acesso ao GPS, lista todas as mensagens enviadas pelo GPS;
10-GPS: Interpreta → Testes com o GPS, selecionando apenas as mensagens úteis ao projeto;
11-GPS:U-Center → Envia mensagens do GPS para o aplicativo “U-Center":
12-MPU → Configurar a testar comunicação com o MatLab:
13-Bluetooth → testar comunicação Bluetooth
14-BT - Cmds AT → configurar com comandos AT a interface Bluetooth.

4. DADOS OBTIDOS

4.1. Aquisição de Dados do GPS

Além da parte de alimentação com super capacitor e botões, o presente trabalho teve como objetivo testar a funcionalidade do GPS. Era importante colocar a Caixa Preta em movimento e simular acidentes, para então gravar as informações inerciais e os dados do GPS. Assim, se comprovou o funcionamento do GPS e se testou a importação desses dados na plataforma do Google Maps, para que se pudesse avaliar a precisão da placa. Alguns dos testes se deram com a placa parada dentro de um local fechado, porém próximos a uma janela ou porta que desse para a rua, e os outros foram feitos dentro do carro, com este em movimento, no percurso mostrado nas imagens abaixo. A Figura 4.1 mostra o percurso onde seriam feitos os ensaios dentro de casa (ponto fixo de medição), e fora de casa no carro (trajetória do carro), em vermelho.

Figura 4.1: Vista aérea do percurso que seria feito pelo carro em Águas Claras

Os ensaios finais foram organizados em três formas. O primeiro com a placa parada, o segundo com ela em movimento e simulando o acidente com o carro parado ao final do percurso, e o último ensaio com o carro em movimento e a simulação da batida no meio do percurso, mas com este continuando o seu trajeto.

Esses ensaios (testes), assim como visto na seção 3.2 sobre o software, foram feitos no **Modo Opera**, selecionando a opção "**1-Adquirir Dados**", que possui nela, uma etapa de obtenção simultânea dos dados do GPS, enquanto o veículo não está acidentado. A indicação de um acidente é simulada pelo acionamento do botão central (SW3) quando estamos na opção "1-Adquirir Dados". Neste modo, como a finalidade é o ensaio, o acionamento de SW3 inicia a aquisição de dados por

127 segundos (preenche toda a memória) e depois envia todos os dados pela porta serial. Assim, o usuário tem a opção de iniciar a gravação no instante adequado. Em suma, com o acionamento de SW3 o usuário indica o instante em que deve iniciar a aquisição dos dados. São gravados dados da plataforma inercial e do GPS. Os dados do GPS aqui utilizados para a construção de mapas, foram os de latitudes e longitudes, sendo as outras informações apenas visualizadas nos dados brutos.

Nosso interesse foi apenas pelos dados do GPS. Outras equipes usaram este modo para gerar dados inerciais para tentar recompor os movimentos sofridos pela Caixa Preta.

Gerando Mapas pelo Google My Maps

O formato das coordenadas do Google Maps, para se gerar os mapas é em graus decimais (DD), ou seja -15.839725, -48.029164. As mensagens de dados que conseguimos pelo GPS estão citadas no Apêndice deste trabalho. E após o acionamento no botão SW3, que simula a aquisição de dados, e o fim dos 127 segundos (preenche toda a memória), a forma como as mensagens do GPS são apresentadas na etapa de enviar todos os dados pela porta serial é vista no exemplo a seguir:

```
#[g
110922
004730
231680
255
0000000000 +0000247936 A 004526 110922 1550.38737 S 04801.75931 W 0.657 K 0.355  1213.6 M 04 5.68 5.21 2.28 34524
0000000001 +0000248064 A 004527 110922 1550.38770 S 04801.75936 W 2.105 K 1.137  1215.0 M 04 5.68 5.20 2.28 36324
0000000002 +0000248192 A 004528 110922 1550.38752 S 04801.75949 W 1.085 K 0.586  1215.7 M 04 5.68 5.20 2.28 38124
0000000003 +0000248320 A 004529 110922 1550.38746 S 04801.75960 W 1.294 K 0.699  1215.6 M 04 5.67 5.19 2.28 39924
0000000004 +0000248448 A 004530 110922 1550.38771 S 04801.75960 W 0.477 K 0.258  1216.7 M 04 5.67 5.19 2.28 41724
…
0000000253 +0000247680 A 004936 110922 1550.38406 S 04801.75778 W 1.637 K 0.884  1211.9 M 04 5.00 4.41 2.34 32688
0000000254 +0000247808 A 004937 110922 1550.38466 S 04801.75782 W 1.559 K 0.842  1211.8 M 04 4.99 4.41 2.34 34488
g]#
```

Tais dados são detalhados na seção 3.2, na parte de “Opera”, e no Apêndice. Os destacados em vermelho são os dados da latitude e sua direção (N/S), e em azul temos as longitudes (W/E). Importante perceber que os dados adquiridos, salvos na SRAM e mostrados na serial, estão em um formato DDMM.MMMMM para a latitude e DDDMM.MMMMM para longitude, onde D equivale a *Degrees*, e M *Minutes*. Pelo fato de haver minutos após a vírgula, se faz necessário a conversão destes para se obter o formato DD (*Decimal Degrees*), como é mostrado abaixo, utilizado na construção dos gráficos.

exemplo:

	Decimal Degrees = degrees + minutes/60
DDMM.MMMMM	*Decimal Degrees* = 15 + (50.44648 / 60)
1550.44648	*Decimal Degrees* = 15.840774666666

Contudo, são necessários alguns ajustes para se usar o Google Maps na sua função de criar mapas, a chamada *My Maps*.[16] O procedimento para tal foi dividido em cinco partes:

1. Obter os dados de localização com uma numeração, seguida por vírgula com a Latitude e uma última vírgula com a Longitude (ambos com seus correspondentes positivos ou negativos, de

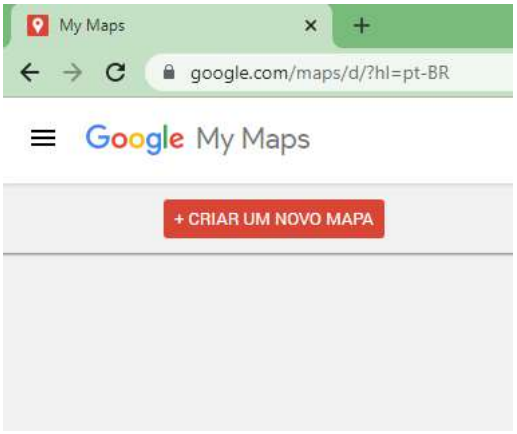
acordo com sua direção), sem espaços entre elas, para o formato .csv, ou separá-los em colunas no Excel.

Exemplo:

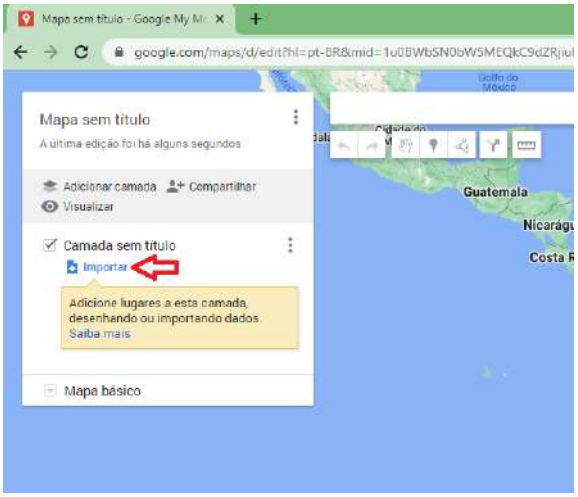
1,-15.839443206,-48.029048919
2,-15.839437484,-48.029052734 ou
3,-15.839428901,-48.029052734

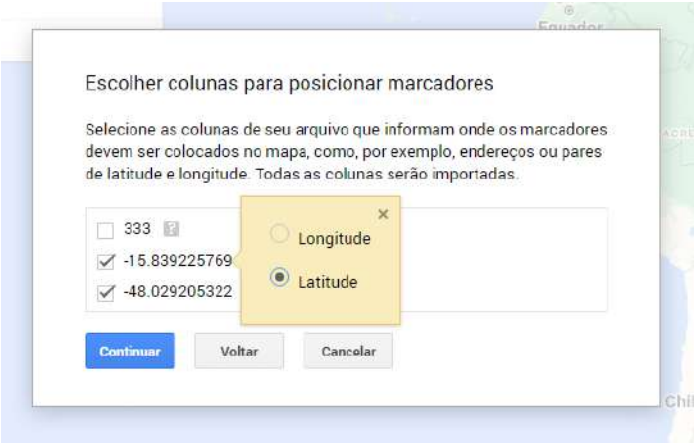
	A	B	C
1	0	-15,504683100	-48,019145700
2	1	-15,504696900	-48,019205200
3	2	-15,504707200	-48,019266400
4	3	-15,504712400	-48,019314500

2. Salvar tais dados em um arquivo csv, xlsx, kml, ou gpx.
3. Acessar o My Maps e criar um novo mapa.



4. Importar o arquivo e selecionar as opções correspondentes a Latitude, Longitude e numeração.





5. Gerar e salvar o mapa.

Ensaios com a Caixa Preta-CXP

Os ensaios em movimento foram feitos com o Protótipo Caixa Preta Versão 2 de 2019, no painel do carro, ligado a um computador, com uma pessoa dirigindo e outra operando os equipamentos, ilustrado na Figura 4.2, e o ensaio parado foi feito com a placa na janela de um apartamento, como mostra a Figura 4.3 abaixo.

Figura 4.2: Placa no painel do carro.

Figura 4.3: Placa na janela do apartamento.

A forma como os ensaios são feitos, utilizando o painel de LCD e o monitoramento pelos botões, apresenta os seguintes passos:

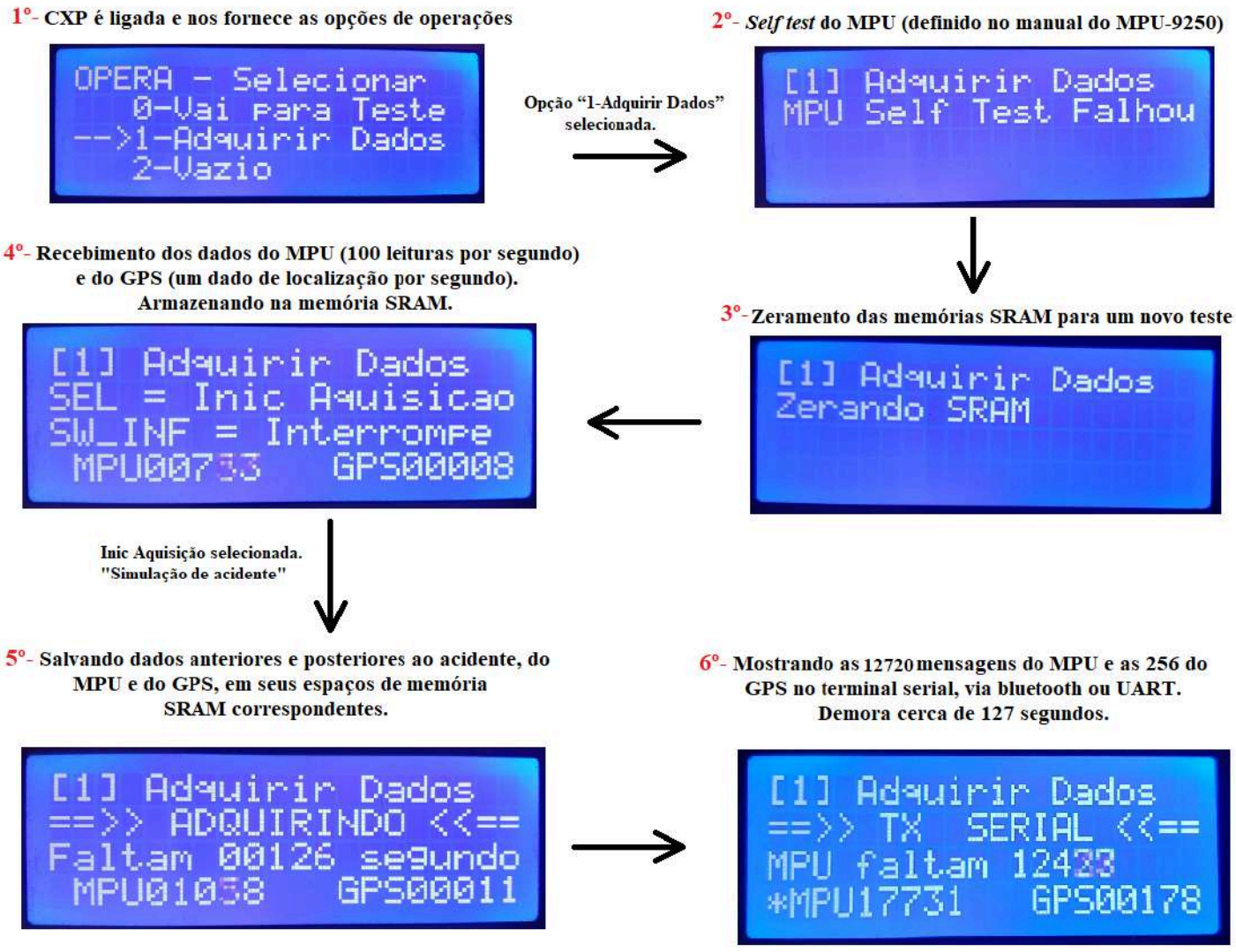


Figura 4.4: Passos dos ensaios para a simulação de um acidente, visto no LCD.

Como o fim da 6º etapa apresentada, o programa da CXP volta a 3º etapa, zerando a SRAM, e começando novamente a aquisição de dados de forma cíclica, até ser acionada a chave SW3 (SEL) de novo. Para sair da opção “1-Adquirir Dados” , na 4º e 5º etapa , basta apertar a sequência de botões: SW1 (esquerda), SW2 (cima) e SW5 (direita). Tal comando volta o programa para a 1º etapa, que é a mesma de quando a CXP liga. No entanto, se a mesma sequência for acionada na 6º etapa, a caixa preta primeiro encerrará a apresentação dos dados na serial, e depois voltará a 1º etapa.

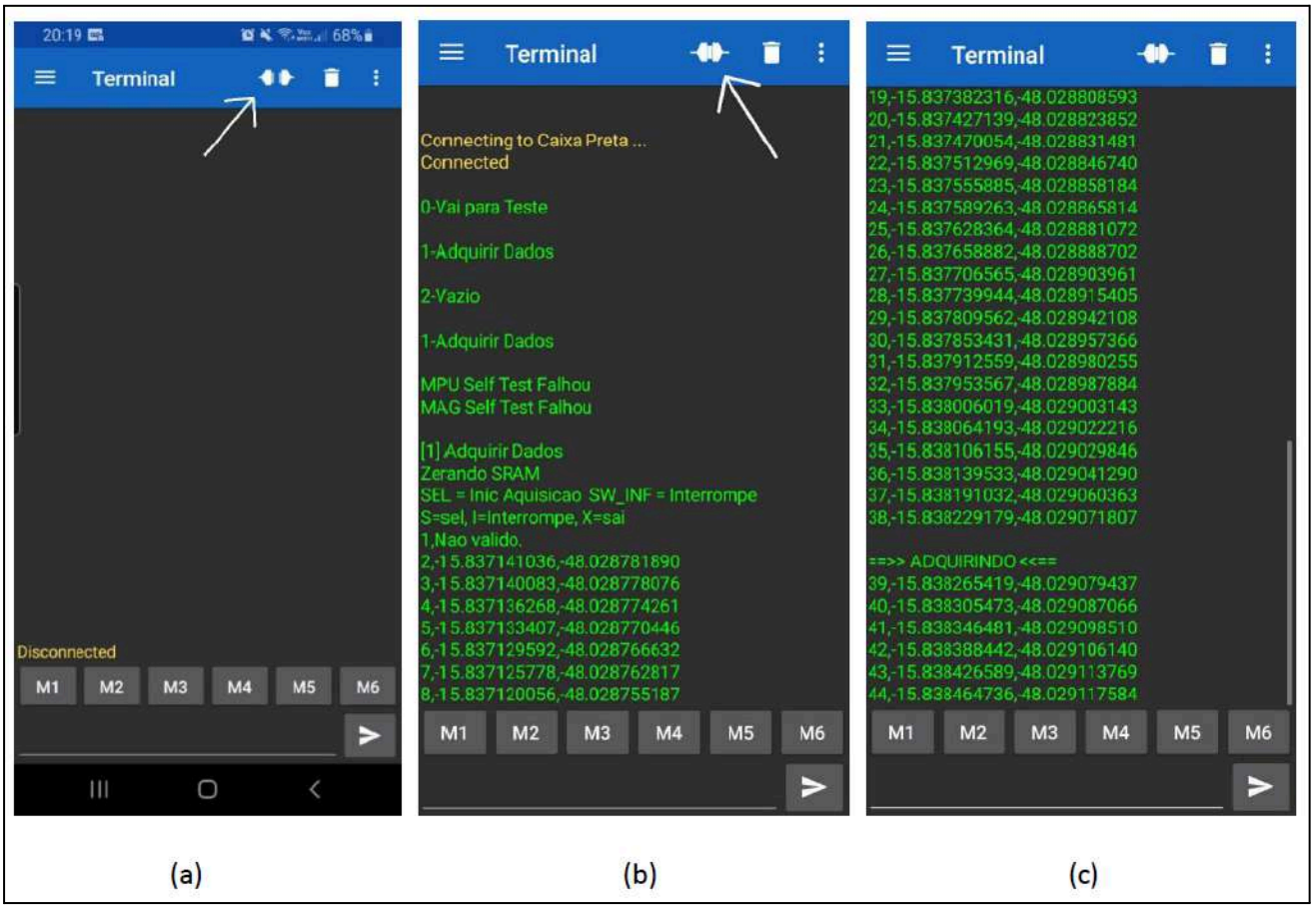
Para facilitar o entendimento de quais dados estão sendo usados para a montagem dos mapas, vamos elencá-los em três fases:

I - Recebimento: Dados de latitude e longitude que estão sendo recebidos do GPS, gravados e regravados na SRAM, enquanto não ocorre um acionamento das chaves. (4º passo da Figura 4.4);
II - Adquirindo: Dados dos 127s após o acionamento da chave para “simulação de acidente” (5º passo da Figura 4.4);
III - Serial: Dados inerciais e do GPS, que foram gravados na SRAM e apresentados no terminal serial, via Bluetooth ou UART (6º passo da Figura 4.4).

4.2 Mapas dos Ensaios

Dados passados para um terminal serial no celular pelo Bluetooth

Como o desenvolvimento de um software próprio, em Python, ainda não foi feito, os ensaios com o Bluetooth se utilizaram de um aplicativo encontrado na internet, chamado “Serial Bluetooth Terminal”, disponível no Google Play Store [17] e instalado em um celular Android. As imagens a seguir, da Figura 4.5, demonstra como as informações aparecem no terminal serial e suas fases de dados correspondentes da Figura 4.4.



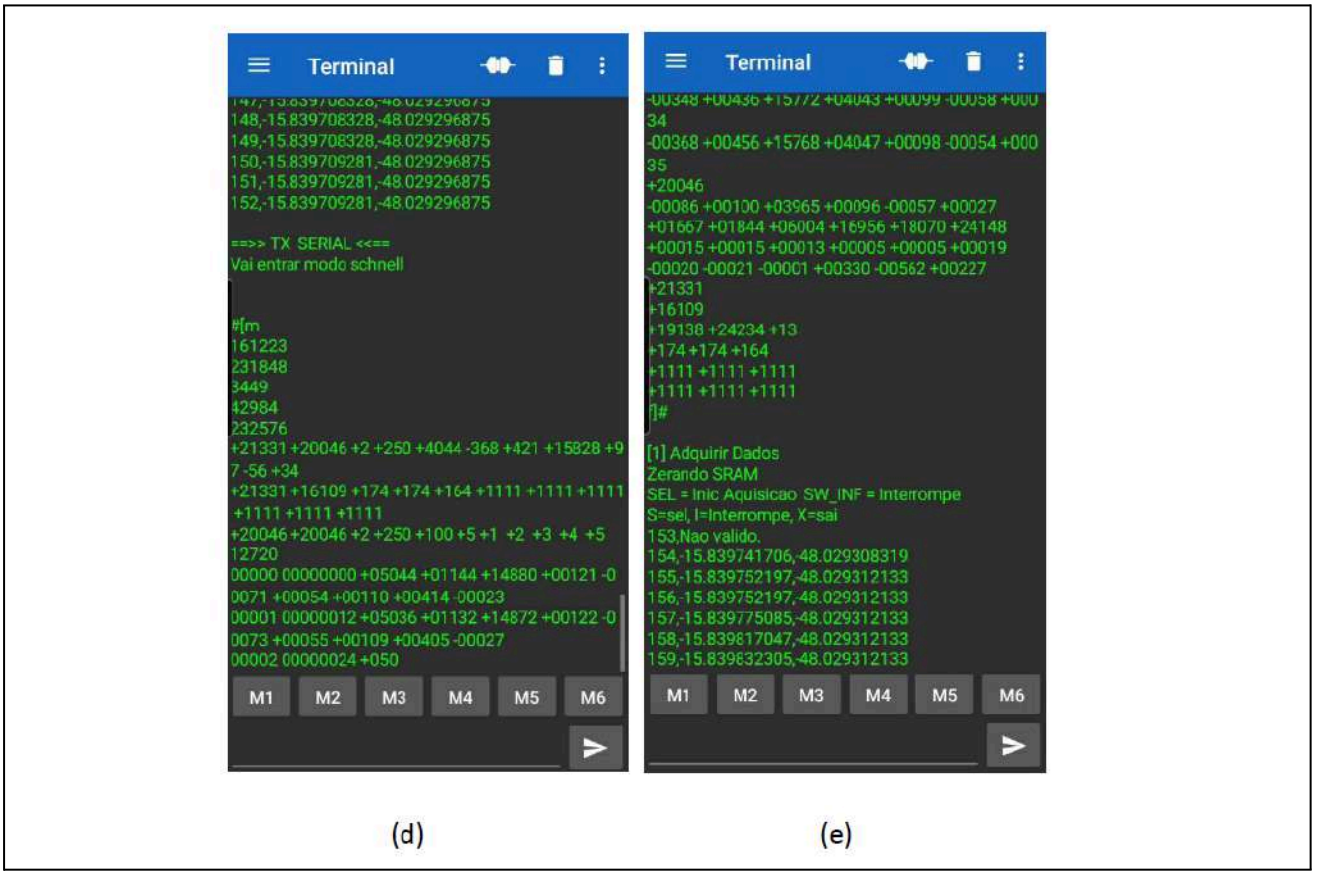


Figura 4.5: Passos dos ensaios para a simulação de um acidente, visto no terminal serial de um celular via Bluetooth. (a) Aplicativo não conectado a CXP; (b) Aplicativo conectado via Bluetooth, seleção da opção "1-Adquirir Dados" e início da fase **I-Recebimento** (4º passo da Figura 4.4); (c) Acionamento da chave SW3 que simula um acidente, é o início da fase **II-Adquirindo** (5º passo da Figura 4.4); (d) Apresentação dos dados (MPU, GPS, calibração local e de fábrica) salvos, fase **III-Serial** (6º passo da Figura 4.4); (e) Final dos dados salvos e apresentados na serial, zeramento da SRAM e retorno a fase **I-Recebimento.**

A utilização do Bluetooth foi de grande auxílio em ensaios iniciais, com o carro em movimento, feitos para testar a funcionalidade do código e dos próprios equipamentos de GPS - GY-NEO6MV2 e modelo HC-05 Bluetooth SPP. Como os arquivos gerados na serial, referentes ao MPU, GPS, calibração local e calibração de fábrica, são muito grandes (no mínimo 262.144 bytes de dados), o aplicativo utilizado nos ensaios não conseguia salvar um arquivo texto, com todas as mensagens salvas no terminal serial.

Com a criação de um app exclusivo, um aperfeiçoamento do programa deverá ser feito, a fim de apresentar tais mensagens (#[m…m]# … #[f…f]#), uma por vez, somente quando selecionadas (para fim de testes). Portanto, os mapas aqui gerados, foram feitos através do terminal UART.

Dados passados para um terminal serial de um computador pela porta UART

A construção dos mapas foi feita somente com os dados de latitude e longitude do GPS, sendo dividida em três ensaios, onde cada um pode ou não receber as três fases para a montagem de mapas, citadas na unidade 4.1. Ou seja, no Ensaio 1, testamos a CXP parada na janela de um apartamento, como mostrado na Figura 4.3, e geramos um mapa referente aos dados entregues na fase **I - Recebimento**. Já nos Ensaio 2.1 e 2.2, estaremos com a CXP no painel do carro, Figura 4.2, e seus trajetos, figura 4.1, serão feitos do início ao fim na fase **I - Recebimento** , correspondente ao 4º passo da Figura 4.4, e no final do percurso, com o carro parado, é acionada a chave SW3, entrando na fase **II - Adquirindo**, seguida da **III - Serial** . O ensaio 2.1 foi o primeiro feito no carro, com o GPS começando a ter uma estabilidade, e o ensaio 2.2 foi o último feito no carro, com o GPS ligado a mais tempo, estando mais estável. Por fim temos o ensaio 3, que passa metade do trajeto em movimento na fase **I - Recebimento**, ocorrendo então o acionamento da chave SW3, entrando na fase **II - Adquirindo**, com o carro continuando seu movimento, seguida da fase **III - Serial**.

Para ser melhor exemplificados, se elencou os ensaios com suas fases da seguinte forma:

Ensaio 1 - CXP parada na janela de um apartamento.

1º mapa gerado pela fase "**I - Recebimento"**

Ensaio 2.1 - Percurso total com "acidente" (acionamento de SW3) ao final do trajeto e o carro parado. Primeiro ensaio feito no carro.

1º mapa gerado pela fase "**I - Recebimento**".
2º mapa gerado pela fase "**II - Adquirindo**".
3º mapa gerado pela fase "**III - Serial**".

Ensaio 2.2 - Percurso total com "acidente" (acionamento de SW3) ao final do trajeto e o carro parado. Último ensaio feito no carro.

1º mapa gerado pela fase "**I - Recebimento**".
2º mapa gerado pela fase "**II - Adquirindo**".
3º mapa gerado pela fase "**III - Serial**".

Ensaio 3 - Acionamento de um "acidente" no meio do trajeto e carro continuando em movimento. Segundo ensaio feito no carro.

1º mapa gerado pela fase "**I - Recebimento**".
2º mapa gerado pela fase "**II - Adquirindo**".
3º mapa gerado pela fase "**III - Serial**".

A seguir será feita uma discussão sobre os dados apresentados, seguida da exposição das imagens dos ensaios, com os mapas equivalentes de cada fase, pois estas são muito grandes. O trajeto, Figura 4.1, feito pelo carro durante os ensaios, tem cerca de 1,91Km, segundo o Google Maps, e demorou cerca de 5,73 minutos para ser percorrido, em uma velocidade de 20km/h. Importante mencionar que a localização dos pontos geográficos é visualizada através do "alfinete" azul, mostrado ao lado. 

No Ensaio 1, Figura 4.6, o mapa criado com a fase I-Recebimento, obteve um tempo médio de 50 segundos de demora para a chegada de dados válidos, estando a placa na janela, que é um ambiente quase aberto. Em ambientes fechados a conexão com um satélite tende a demorar muito mais, ou às vezes não é feita. Os ensaios 2.1, 2.2 e 3, obtiveram uma conexão relativamente rápida, pelo fato da placa estar localizada no painel do carro, e este fora da garagem coberta.

Fazendo uma comparação entre as fases I-Recebimento, dos ensaios 2.1 e 2.2, nota-se que após estar ligado por mais tempo, o GPS fica mais preciso em seus dados, até mesmo em locais fechados, que é o caso do viaduto do metrô. Já nessa mesma fase no ensaio 3, é possível perceber o ponto exato onde o acionamento da chave SW3 foi feito, seguido da fase II-Aquisição, que praticamente completa o percurso, e a "soma" delas, pode ser vista na fase III-Serial, que claramente representa os dados anteriores e posteriores ao "acidente".

As fases II-Aquisição, dos ensaios 2.1 e 2.2, são praticamente um ponto no mapa, levando em conta a alta concentração de dados em uma determinada região, com poucos desses sendo erros. Agora com a fase III-Serial desses ensaios, é presumível a descoberta de 1,18km de dados salvos anteriores a um possível acidente.

No geral, os dados de localização do GPS estão bem precisos com a realidade dos percursos por onde ele passou. O custo benefício do GPS - GY-NEO6MV2, utilizado neste trabalho, foi aprovado, levando em conta que estamos nas fases de testes da construção da CXP.

Figuras dos ensaios citados:

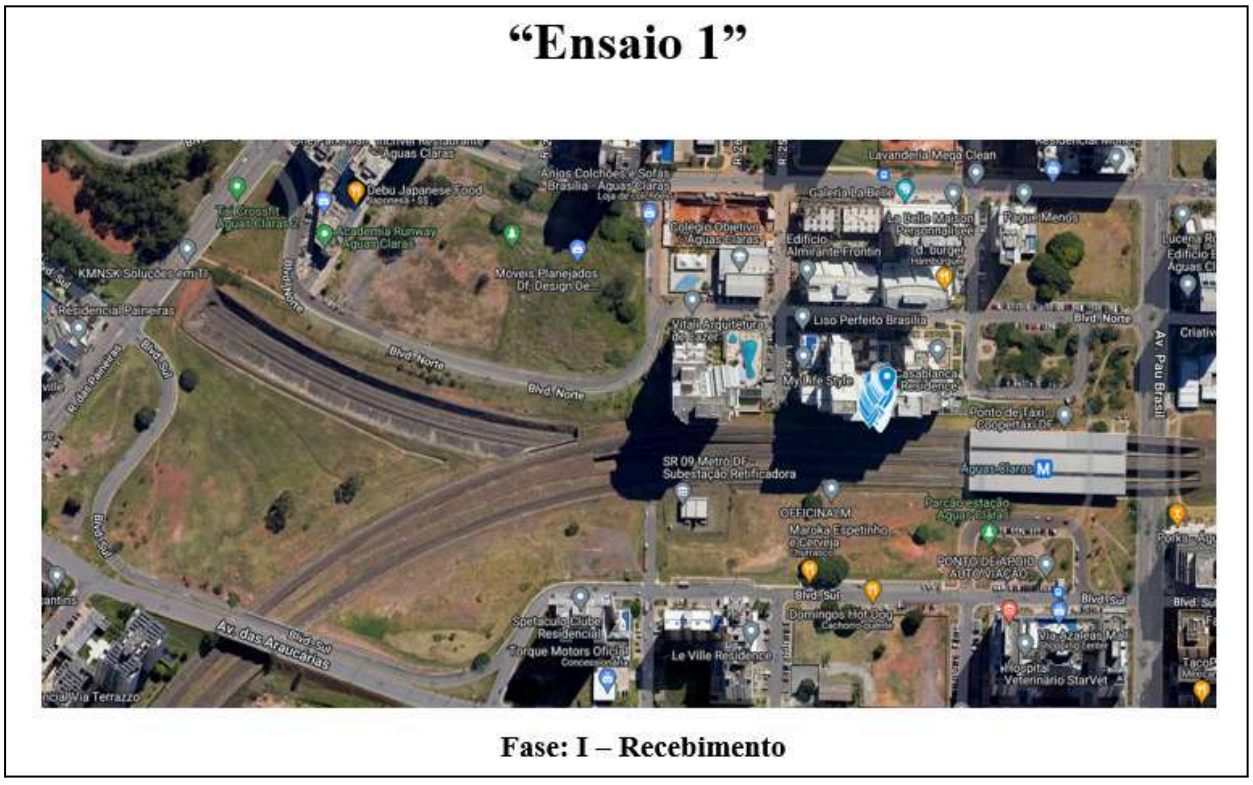


Figura 4.6: Ensaio 1 com a Fase: I - Recebimento.

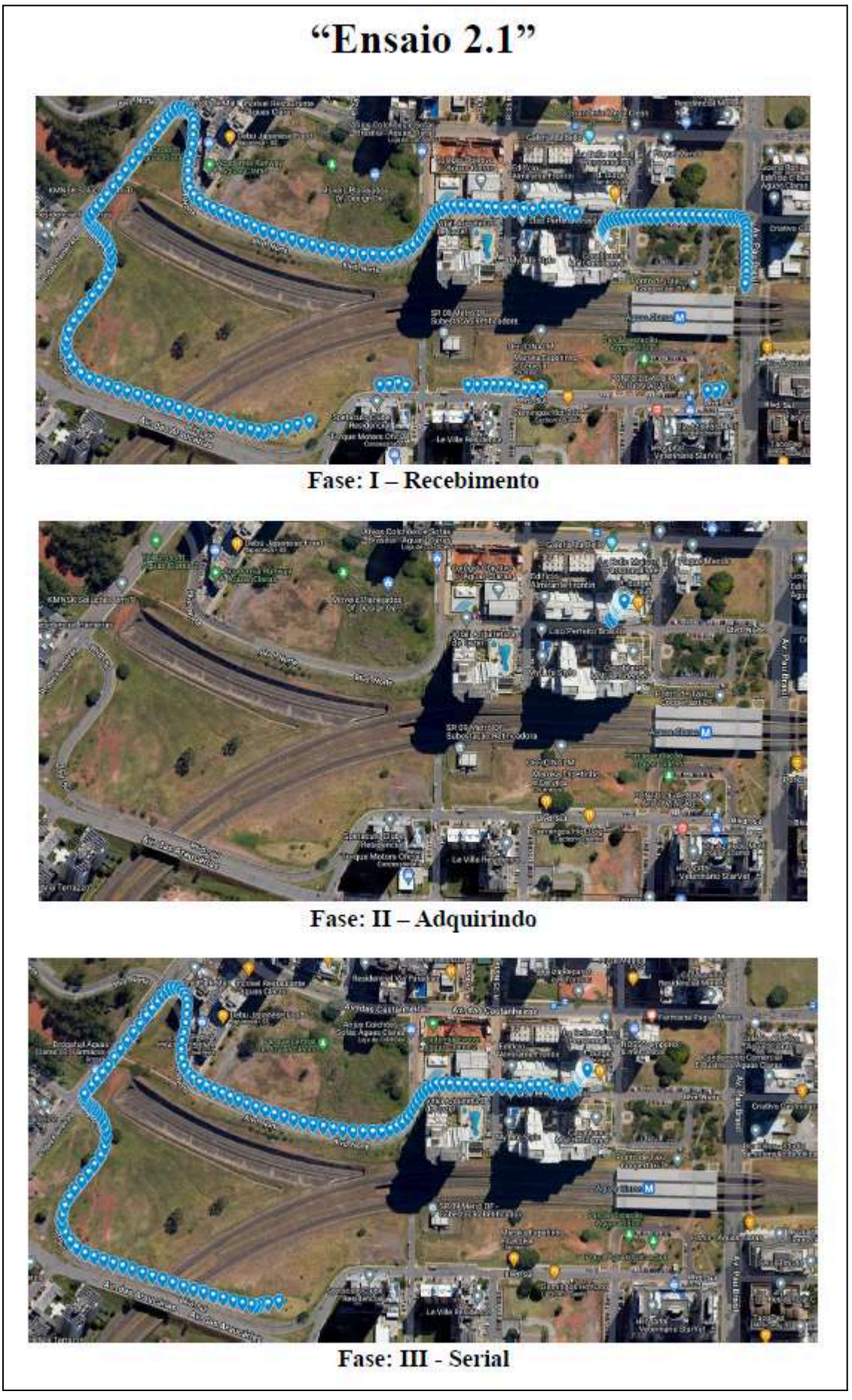


Figura 4.7: Ensaio 2.1 com as Fases: I - Recebimento, II - Adquirir e III - Serial

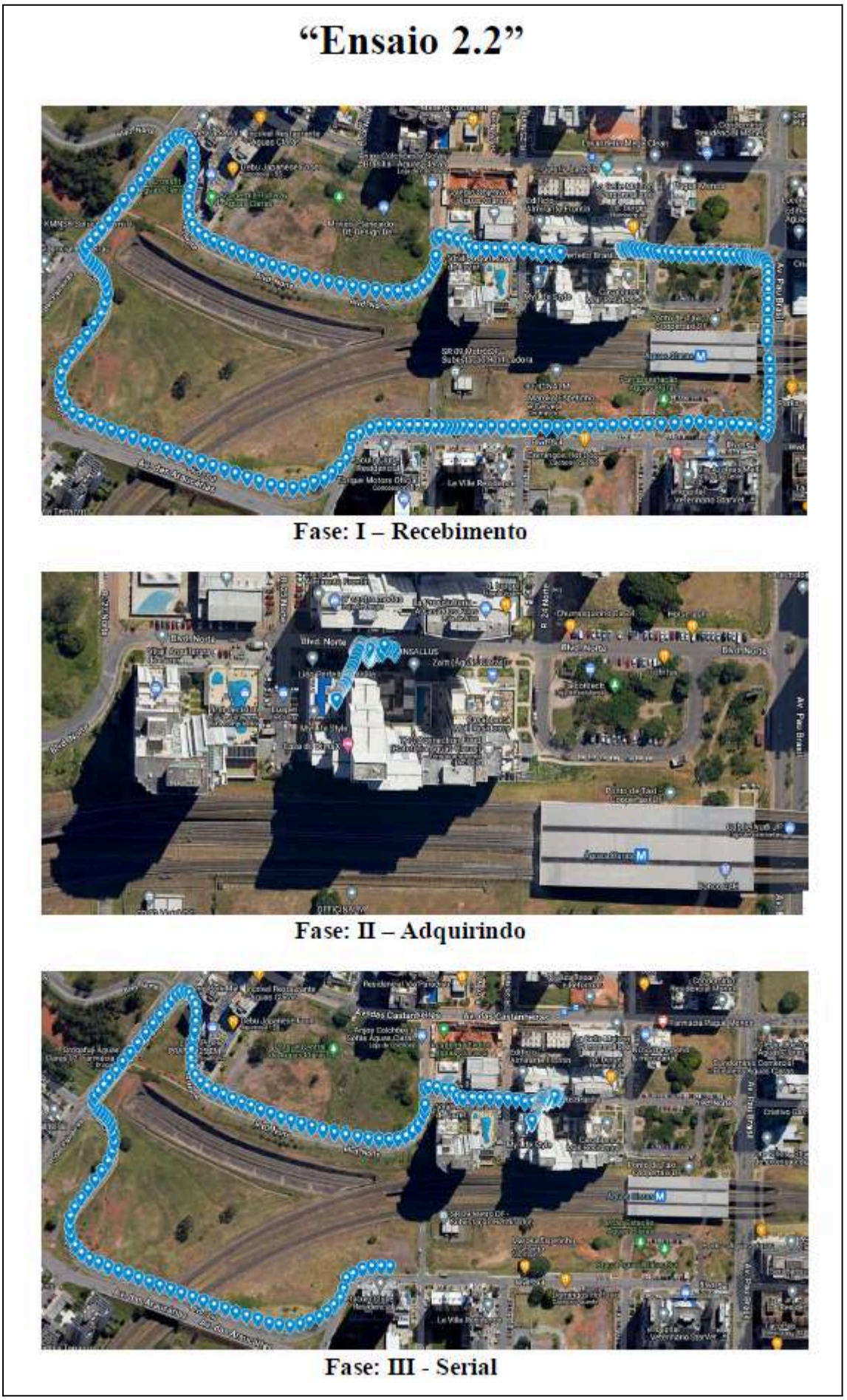


Figura 4.8: Ensaio 2.2 com as Fases: I - Recebimento, II - Adquirir e III - Serial

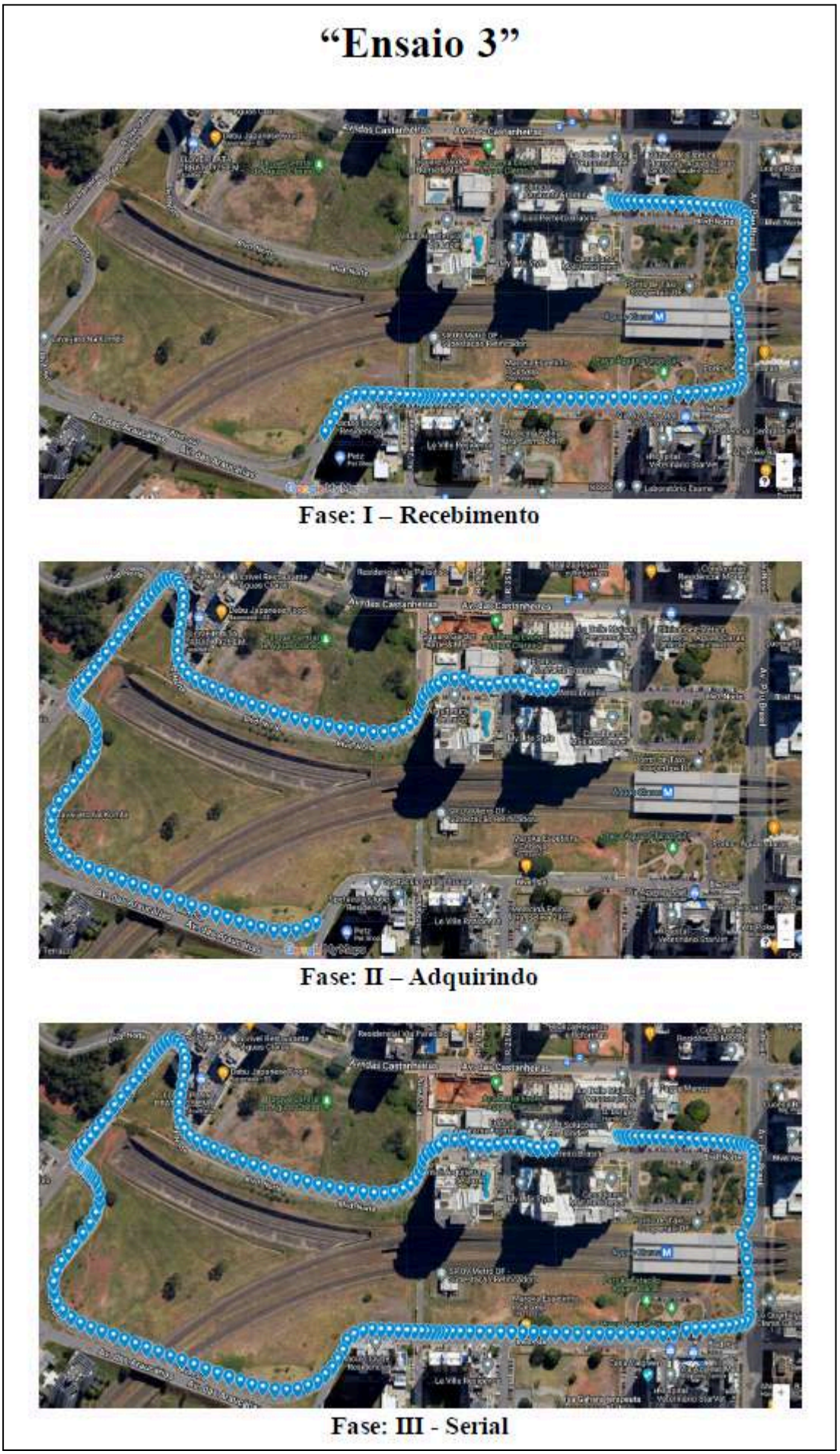


Figura 4.9: Ensaio 3 com as Fases: I - Recebimento, II - Adquirir e III - Serial

5. CONSIDERAÇÕES FINAIS

Diante do que foi desenvolvido neste trabalho, pode-se dizer que a sustentação de energia para a caixa preta funcionou muito bem nos ensaios feitos em casa e no carro (ligado por um conector USB ao acendedor de cigarros de 12V), pois os testes feitos no osciloscópio demonstraram que os diodos cumpriram seus papéis para reduzirem as tensões que chegavam para o super capacitor. O carregamento e descarregamento deste, que também tiveram auxílio de diodos, trabalhou muito bem, e conseguiu cumprir sua função de fornecer energia para o auto desligamento do Arduino Mega 2560.

Para o monitoramento da alimentação do Arduino, foram criados divisores resistivos, posicionados nas tensões do super capacitor e na de 11.3V (Figura 3.9). Os valores fornecidos por esses divisores forneceram alertas indicando se o carro estava ligado, quando este foi desligado e qual o nível de energia no super capacitor.

Com relação a conexão dos botões (chaves) da placa no Arduino, sem que este perdesse grande parte de suas portas com isso, a solução adotada foi: construir um divisor resistivo, controlado pelas chaves cuja saída é lida pelo ADC do Arduino. Tal desenvolvimento se mostrou muito satisfatório, tendo em vista que o uso linear da faixa de 0 a 5V, demarcado com tensões centrais para cada uma das chaves, fez com que a entrada do ADC conseguisse identificar estas com muita facilidade, operando em 8 bits. A solução ainda levou em consideração valores de resistências comerciais, que conseguiram manter os valores das tensões dentro das médias estipuladas.

A respeito dos ensaios feitos com o GPS, percebe-se um saldo positivo com relação à conexão da caixa preta com os satélites quando esta primeira está dentro do carro em movimento. A localização real da CXP corresponde aos dados de latitude e longitude recebidos por ela. Sendo problemática apenas em locais fechados, onde se perde a conexão. Já os ensaios feitos para recolher as mensagens salvas na SRAM, relativas a momentos antes e depois da simulação de um acidente, funcionaram bem, pois os dados coletados estavam de acordo com a realidade de onde o carro fez seu percurso, e foram informações suficientes para a geração dos mapas, que podem auxiliar na perícia.

5.1 Trabalhos Futuros

Para a continuação do desenvolvimento da proposta da caixa preta, é necessário serem feitos testes com esta ligada diretamente ao acendedor de cigarros de 12V, sem o auxílio de um conector USB, para que assim se possa comprovar em campo, a sustentação do super capacitor. Pois o objetivo final da caixa preta é estar conectada diretamente ao carro.

Também é necessário ser feita a junção dos dados de simulação do MPU e do GPS e estes testados, assim como a integração dos softwares. Como já foi citado, também há a proposta de criação de um aplicativo, em Python, para ser usado via Bluetooth, em ensaios, visto que esse é de suma importância para a fase comercial da caixa preta, onde ela não estará equipada com o LCD, e precisará fornecer os dados de forma simples para o usuário final.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ARAÚJO, Jefferson Ferraz D. F. "Construção de um magnetômetro Hall para caracterização de partículas magnéticas utilizadas em ensaios imunológicos". Orientador: Antonio Carlos Oliveira Bruno. Co-Orientadora: Sônia Renaux Wanderley Louro. 2009. Dissertação (Mestrado) - Curso de Física, Departamento de Física, PUC-Rio, Rio de Janeiro. 2009. Disponível em: <https://www.maxwell.vrac.puc-rio.br/14463/14463_4.PDF> Acesso em: 24 ago. 2022

[2] Arduino: Documentation Arduino Mega 2560. Disponível em: <https://store.arduino.cc/products/arduino-mega-2560-rev3> Acesso em: 22 ago. 2022

[3] Arduino: Ide arduino. Disponível em: <https://www.arduino.cc/en/Main/Software>

[4] ASHFORD, Peter. Technically Speaking Parte I: Flight Data Recorders:The Background on the 'Black Box' ". Revista Avionics News. Fevereiro de 2010. Documento eletrônico. Disponível em: <http://aea.net/AvionicsNews/ANArchives/TechSpeakFeb10.pdf> Acesso em: 21 ago. 2021

[5] ASHFORD, Peter. Technically Speaking Parte II: Flight Data Recorders: Built, Tested to Remain Intact After a Crash". Revista Avionics News. Março de 2010. Documento eletrônico. Disponível em: <http://aea.net/AvionicsNews/ANArchives/TechSpeakMar10.pdf> Acesso em: 21 ago. 2021

[6] BBC Brasil. "Quem inventou a caixa preta?". 16 de abril de 2014. Reportagem eletrônica. Disponível em: <https://www.bbc.com/portuguese/noticias/2014/04/140415_sp_caixa_preta_inventor_hb> Acesso em: 22 ago. 2021

[7] BOLSTAD, Paul: "GIS Fundamentals: A First Text on Geographic Information Systems". Editora XanEdu Publishing, Inc., 6ª Ed. Junho de 2019.

[8] BOSCH. Bosch: Kit básico DLC de recuperação de dados de falha. Disponível em: <https://cdr.boschdiagnostics.com/cdr/products/crash-data-retrieval-dlc-base-kit> Acesso em: 27 fev. 2023

[9] CALACHE, Danilo: Caracterização de um Acelerômetro Baseado em Sistemas Microeletromecânicos (MEMS). Trabalho de conclusão de curso (bacharelado em engenharia elétrica), Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2013. Disponível em: <http://www.lee.uerj.br/~jpaulo/PG/2013/PG-Acelerometro-MEMS-2013.pdf>

[10] Datasheet Asahi KASEI: AK8963 - 3-axis Electronic Compass, outubro 2013. Disponível em: <https://download.mikroe.com/documents/datasheets/ak8963c-datasheet.pdf> Acesso em: 24 ago.2022

[11] Datasheet HC-05-Bluetooth to Serial Port Module. Disponível em: <https://datasheetspdf.com/datasheet/HC-05.html> Acessado em 22 ago. 2022

[12] Datasheet GPS gy-neo6mv2. Disponível em: <https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/GY-NEO6MV2-GPS-Module-Datasheet.pdf>Acesso em: 22 ago. 2022

[13] Datasheet LCD. Disponível em: <http://pdf1.alldatasheet.com/datasheet-pdf/view/63673/HITACHI/HD44780.html> Acesso em: 22 ago. 2022

[14] Datasheet InvenSense Inc.: MPU9250. Disponível em: <https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf> Acesso em: 22 ago. 2022

[15] GISGeography: Trilateration vs triangulation – how gps receivers work. Disponível em: <https://gisgeography.com/trilateration-triangulation-gps/> Acesso em: 30 ago. 2022

[16] Google Maps, plataforma de criação de mapas "My Maps". Disponível em: <https://www.google.com/intl/pt-BR/maps/about/mymaps/> Acesso em: 01 ago. 2022

[17] Google Play Store. Serial Bluetooth Terminal. Versão "1.36". Aplicativo. Disponível em: <https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal&hl=pt_BR> Acesso em: 01 set. 2022

[18] GREWAL, Mohinder S.; WEILL, Lawrence R.; ANDREWS, Angus P. "Global positioning systems, inertial navigation, and integration". John Wiley & Sons, 2ª Ed. 2007.

[19] LIMA, Sandro Reginato Soares de. Integração GPS/INS utilizando sensores inerciais baseados em sistemas microeletromecânicos (MEMS). Orientador: Sílvio Rogério Correia de Freitas. Co-Orientadores: Cláudia Pereira Krueger e Günter Seeber. 2005. Dissertação (Doutorado) - Curso de Ciências Geodésicas, Departamento de Geomática, Universidade Federal do Paraná, Curitiba. 2005. Disponível em: <https://acervodigital.ufpr.br/handle/1884/2351> Acesso em: 29 ago. 2022

[20] LIMA, Vinícius De Oliveira: Proposta de plataforma inercial para auxiliar na perícia de acidentes de trânsito. 2016.

[21] LOPES, Gabriela da Silva: Caixa Preta para Veículos Automotivos. Trabalho de conclusão de curso (bacharelado em engenharia da computação), Universidade de Brasília, Brasília, 2018.

[22] NETO, Paulo B. Teixeira: Caixa Preta para carros: proposta para calibração, coleta e fusão de dados.Trabalho de conclusão de curso (bacharelado em engenharia da computação), Universidade de Brasília, Brasília, 2021.

[23] NMEA (*National Marine Electronics Association*) Disponível em : <https://www.nmea.org/> Acesso em: 23 ago. 2022

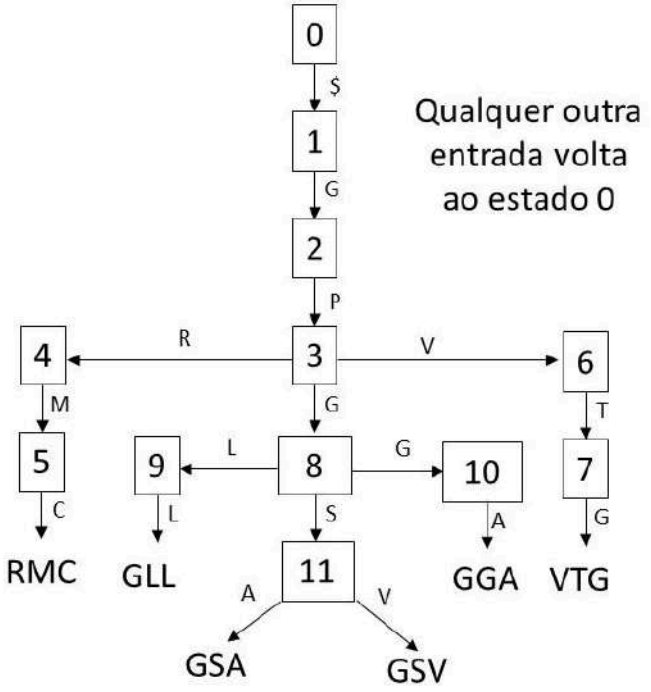
[24] RAMOS, Hudson Pereira; LUCENA, Vanessa Oliveira: Proposta de plataforma inercial e simulador 3d para periciar acidentes de trânsito. trabalho de conclusão de curso (bacharelado em engenharia mecatrônica), Universidade de Brasília, Brasília, 2017.

[25] ZELENOVSKY, Ricardo; MENDONÇA, Alexandre. "Arduino: Guia Avançado para Projeto". Editora Interciência, 1ª Ed. 2019.

APÊNDICE

Mensagens recebidas pelo GPS

Como mencionado na seção “2.1.2 Sistema de Posicionamento Global”, o módulo GPS utilizado, GPS - GY-NEO6MV2, nos fornece informações baseadas nas especificações da NMEA[23]. Os tipos de mensagens recebidas foram descobertas usando o diagrama de estados abaixo. A documentação dessas mensagens foi disponibilizada pelo orientador do projeto, e serão aqui citadas.



O resumo das mensagens de onde se retiraram informações é mostrado a seguir:

---------- GPRMC ----------

```
$GPRMC,hhmmss,status,latitude,N,longitude,E,spd,cog,ddmmyy,mv,mvE,mode*cs<CR><LF>
$GPRMC,083559.00,A,4717.11437,N,00833.91522,E,0.004,77.52,091202,,,A*57
```

0	1	2	3	4	5	6
$GPRMC,	hhmmss.sss,	Stat,	Lat:ddmm.mmmmm,	N/S,	Long:dddmm.mmmmm,	E/W,
$GPRMC,	083559.00,	A,	4717.11437,	N,	00833.91522,	E,
7	12	2	11	2	12	2

7	8	9	10	11	12	13	14
Speed:ddd.ddd,	Curso:ddd.ddd,	Data:ddmmyy	Mv,	mvE,	Modo	*Check	CR LF
0.004,	77.52,	091202,	,	,	A	*57	0xD 0xA
?8	?8	7	?8	?2	1	3	2

```
Tamanho = 80 bytes, vou usar tamanho 100.
Lido com Arduino: $GPRMC,131732.00,A,1548.62581,S,04748.65809,W,0.299,,260120,,,A*72
```

---------- GPGSA ----------

```
$GPGSA,Smode,FS{,sv},PDOP,HDOP,VDOP*cs<CR><LF>
$GPGSA,A,3,23,29,07,08,09,18,26,28,,,,,1.94,1.18,1.54*0D
```

Mostra identificador de até 12 satélites

0	1	2	3	4	5	6	7	8	9	10
$GPGSA,	Smode,	Fix,	Sat1,	Sat2,	Sat3,	Sat4,	Sat5,	Sat6,	Sat7,	Sat8
$GPGSA,	A,	3,	23,	29,	07,	08,	09,	18,	26,	28,
7	2	2	3	3	3	3	3	3	3	3

11	12	13	14	15	16	17	18	19
Sat9,	Sat10	Sat11,	Sat12,	PDOP,	HDOP,	VDOP	*Check	CR LF
18,	18,	18,	18,	1.94,	1.18,	1.54	*0D	0xD 0xA
3	3	3	3	5	5	5	3	2

Tamanho = 67 bytes, vou usar tamanho 100.
Lido com Arduino: $GPGSA,A,3,07,09,16,23,04,01,,,,,,,8.16,1.24,8.06*09

---------- GPVTG ----------

```
$GPVTG,cogt,T,cogm,M,sog,N,kph,K,mode*cs<CR><LF>
$GPVTG,77.52,T,,M,0.004,N,0.008,K,A*06
```

Mostra curso e velocidade

0	1	2	3	4	5	6
$GPVTG,	Curso T Cogt:ddd.dd,	Fix,	Curso M Cogm:ddd.dd,	Fix	Veloc Nós Sog:ddd.ddd,	Unidade
$GPVTG,	77.52,	T,	ddd.dd,	M,	0.004,	N,
7	7	2	7	3	8	2

7	8	9	10	11
Veloc kph Sog:ddd.ddd,	Unidade	Modo	*Check	CR LF
0.008	K,	A,	*06	0xD 0xA
8	2	2	3	2

Tamanho = bytes, vou usar tamanho 100.
Lido com Arduino: $GPVTG,,T,,M,0.079,N,0.146,K,A*2E

---------- GPGGA ----------

```
$GPGGA,hhmmss.ss,Latitude,N,Longitude,E,FS,NoSV,HDOP,msl,m,Altref,m,DiffAge,DiffStation*cs<CR><LF>
$GPGGA,092725.00,4717.11399,N,00833.91590,E,1,8,1.01,499.6,M,48.0,M,,0*5B
```

Mostra ...

0	1	2	3	4	5	6	7
$GPGGA,	hhmmss.sss,	Lat:ddmm.mmmmm,	N/S,	Long:dddmm.mmmmm,	E/W,	FS,	Nr sat:dd,
$GPGGA,	083559.00,	4717.11437,	N,	00833.91522,	E,	1,	8,
7	12	11	2	12	2	2	

8	9	10	11	12
HDOP,	Alt Msl:dddd.ddd,	uMsl	Altref:dddd.ddd,	Usep
1.18,	1.01,	M,	48.0,	M
5	9	2	9	2

13	14	15	16
Diffage,	DiffStation	*Check	CR LF
S,	0	*57	0xD 0xA
2	2	3	2

Tamanho = bytes, vou usar tamanho 100.
Lido com Arduino: $GPGGA,185326.00,1548.63054,S,04748.65655,W,1,06,2.06,1052.0,M,-11.8,M,,*4F