UNIVERSIDADE FEDERAL DO PARANÁ
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
TE311 DC - OFICINA DE PROJETOS EM ENGENHARIA ELÉTRICA
PROF. DR. ÂNDREI CAMPONOGARA

ALESSANDRO FARAGO - GRR20202601
OTÁVIO A. W. COLARES - GRR20202591
PEDRO H. SARTORELLO - GRR20205776

PROTÓTIPO DE ALTÍMETRO EM ARDUINO PARA USO EM MINIFOGUETES DE COMPETIÇÃO

CURITIBA
2022

RESUMO

O objetivo do projeto é confeccionar um altímetro digital onde a altitude é mostrada e gravada em um cartão pelo dispositivo. destina-se a acompanhar as mudanças na pressão do ar que ocorrem junto com as mudanças na altitude que são convertidas usando um cálculo matemático. É importante salientar que a gravação de forma dinâmica dos dados em um cartão SD foi um dos pontos almejados pelos autores, que de forma ideal foi atingida, assim como seus interesse em codificar o dispositivo de microcontrole arduino para atuar como um altímetro obteve seu êxito, assim concretizando o projeto. O altímetro de bordo é um instrumento barométrico que consiste em uma cápsula aneróide com vácuo parcial interno para medição da pressão ambiente, instalada em uma câmara de tubo foguete onde existem alguns orifícios que possibilitam a variação de pressão que irá ser convertido. Diversas instrumentações e estruturas elétricas que seriam necessárias para a construção do altímetro foram realizadas por módulos acoplados a placas Arduino já existentes; neste cenário, a aplicação é de expertise variada juntamente com as ferramentas já disponíveis.

Palavras-chave: altímetro, foguetemodelismo, barômetro.

SUMÁRIO

1. INTRODUÇÃO

1.1. Objetivos Gerais

1.1.1. Objetivo geral

O presente projeto destina-se na confecção de um altímetro com micro controlador Arduino para implementação em foguetes de pequeno porte para competição. Assim, conforme se verifica na realidade, muitos modelos desse tipo de equipamentos são importados e apresentam erros durante o funcionamento, prejudicando o registro de dados de altitude de equipes de foguetemodelismo com os seus protótipos. Portanto, diante dos protótipos de altímetros envolvidos por outras equipes e competição destina seu presente projeto cuja confecção tem materiais compatíveis ao estudo de eletrônica embarcada utilizando um Arduino nano, sensor BMP280, com entrada para *protoboard*, acionamento sonoro via *buzzer* e modos de memória de cartão SD.

1.1.2. Objetivos específicos

O presente projeto tem os seguintes objetivos específicos:

• Desenvolver um circuito para a coleta de dados de diferentes lançamentos e que possuam dimensões compatíveis com foguetes de competição.

• Implementar um algoritmo em Arduino para a leitura do apogeu, que seja de forma independente do uso do computador, e armazenamento dos dados realizados com memória externa.

1.2 Estrutura do trabalho

O relatório está descrito em sete capítulos, dos quais descrevem de forma aprofundada o funcionamento e a dinâmica da construção adotada no projeto. Dessa forma, esse volume se inicia com a explicação física dos componentes usados na parte de fundamentação teórica, seguido pela descrição do projeto e seu funcionamento na seção de desenvolvimento do projeto. Assim, descreve-se o resultado encontrado e suas implicações, bem como, a descrição e conclusão

encontrada com o protótipo. Por fim, apresenta-se as referências bibliográficas utilizadas na confecção do relatório e as informações complementares a ele.

2. FUNDAMENTAÇÃO TEÓRICA

Tendo em vista a prática de foguetemodelismo e a dificuldade encontrada em registrar os dados de altitude dos protótipos de minifoguetes é destinado o presente projeto. O protótipo desenvolvido utiliza o conhecimento em sensores e microcontroladores, realiza a leitura da variação da pressão atmosférica, armazenamento dos dados e oferece um sistema de recuperação ao usuário do seu minifoguete durante voo.

Dessa forma, nesse capítulo, apresentada a fundamentação teórica, tem-se a descrição em etapas da dinâmica de funcionamento do protótipo desenvolvido. Primeiro o equipamento é descrito através de um diagrama de blocos, seguido pela explicação sobre o sensor barométrico e o bloco de processamento dos dados coletados. A seguir descreve-se o sistema de monitoramento utilizado para tratamento dos dados, por fim, o capítulo finalizado com uma breve introdução sobre minifoguetes.

2.1. Diagrama de blocos

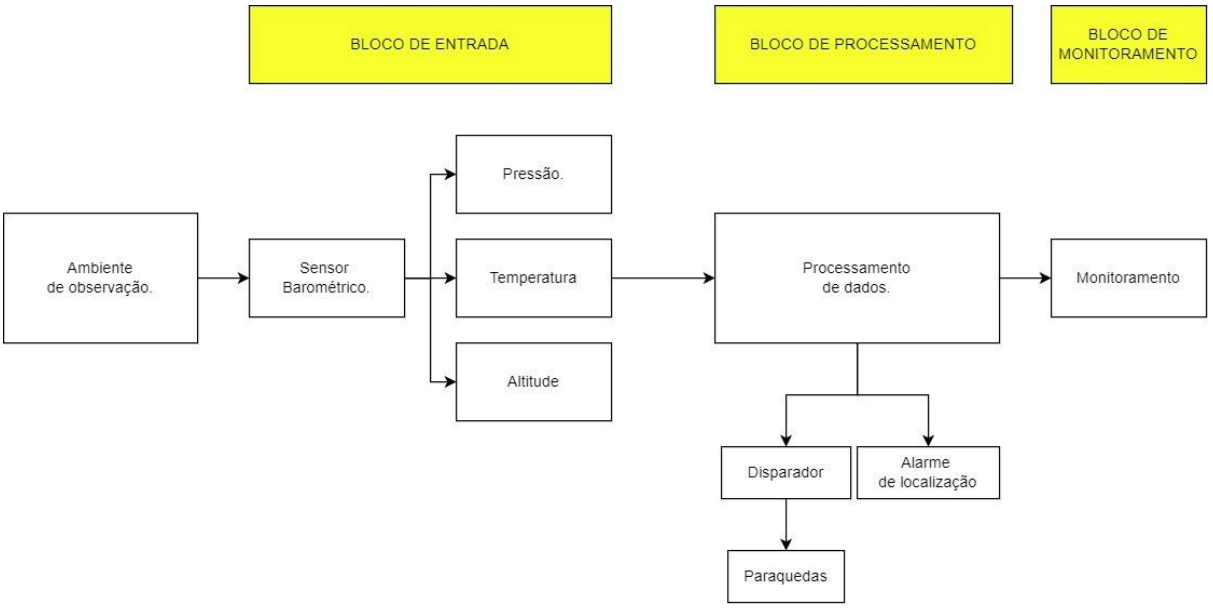
O diagrama de blocos ilustrado na Figura 2.1, representa todo o acompanhamento do sistema. Nele estão contidas todas as etapas necessárias para o funcionamento do altímetro. De acordo com a Figura 2.1, o ambiente de observação do presente projeto consiste em ser qualquer local a céu aberto que permita o lançamento do foguete.

No Bloco de Entrada, o sensor barométrico é o principal componente. Ele é o item que determina as medidas importantes do projeto, como a pressão atmosférica, temperatura e altitude.

No Bloco de Processamento, o microcontrolador Arduino Nano desempenha o papel de interpretar os dados obtidos pelo sensor e, assim, ativa o disparador para abrir o paraquedas e acionar o alarme de localização.

No Bloco de Monitoramento, é efetuada a leitura dos dados externamente, do qual é possível analisar e compreender o comportamento do protótipo de minifoguete do usuário durante o período de lançamento.

FIGURA 2.1.1 - DIAGRAMA DE BLOCOS



Fonte: O autor (2022)

2.2. Sensor barométrico

Como dispositivo de entrada, o sensor BMP280 é utilizado, ele é o sucessor do sensor BMP180, possui ganhos como menor consumo de energia, precisão e tamanho cerca de 63% menor, o que o torna comum em uso nos dispositivos móveis e portáteis, representado na Figura 2.2.1. O sensor possui tensão de operação de 3V, consumo de corrente de 2,7 mA, funciona com interfaces I2C ou SPI, faixa de medição de pressão: 300 kPa à 1100 hPa (cerca de 9000 metros acima do mar à -500 m abaixo do nível do mar), com uma precisão de 0.12 hPa para mais ou para menos. A faixa de medição da temperatura é entre -40 a 85 °C, com uma precisão de mais ou menos 1°C. Ele possui dimensões de 15 mm por 12 mm por 2,3 mm, sua principal vantagem e, por isso, é ótimo para projetos de minifoguetes, que necessitam de um menor espaço para os sensores.

Visto que o sensor possui baixo consumo de energia, a alimentação por baterias permite o funcionamento por longos períodos de tempo, por isso é indicado para projetos como minifoguetes, drones, relógios, estações meteorológicas, dispositivos com GPS, etc.

FIGURA 2.2.1 - SENSOR BMP280

Fonte: Usinainfo (2022)

Sobre seu funcionamento, os sensores barométricos são instrumentos científicos utilizados para medir pressão atmosférica, ou pressão barométrica, é a pressão exercida pela atmosfera sobre uma superfície, ou seja, o peso do ar. No caso do BMP280, é um sensor barométrico digital, que usa um chip para medir a temperatura do ar, conforme descrito em seu manual. O chip é sensível à pressão atmosférica que influencia sua capacidade de conduzir corrente. A mudança de volume de ar afeta sua corrente, assim, a pressão do ar é calculada pela variação dessa corrente. A unidade da Pressão pode ser calculada em atmosferas (atm), ou Pascal (Pa). Onde 1 atm é aproximadamente 101.325 Pa.

Para determinar a altitude é simples, ela é determinada pela medição da pressão do ar, enquanto a pressão do ar decresce e a altitude cresce. Isso acontece porque a densidade do ar é menor em altas altitudes.

A pressão e altura é calculada pela seguinte relação:

$$P = P_b\left[\frac{T_b+(h-h_b)}{T_b}\right]^{\frac{-g_o M}{R^* L_b}}$$

Em que P_b é a pressão atmosférica inicial, tomada como referência [pascal]; T_b é a temperatura de referência [kelvin]; L_b é a gradiente adiabática da temperatura [Padrão Internacional para Atmosfera: kelvin/metro]; h é a altura calculada com base na pressão [metro]; h_b é a altura inicial, tomada como referência [metro]; R^* é a Constante Universal dos Gases Ideais, cujo valor é 8,3144598

[J/(mol·K)]; g_0 é a aceleração da gravidade, cuja média estimada é 9,80665 m/s² e M é a massa molar do ar terrestre, cuja média estimada é 0,0289644 kg/mol.

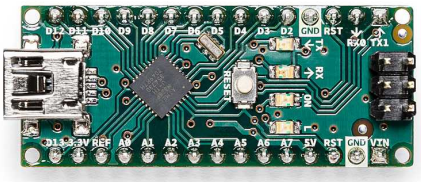
2.3. Bloco de processamento

O Arduino, segundo a própria desenvolvedora, é descrito como uma plataforma de código aberto que conta com hardware e software de fácil uso, destinado à elaboração de projetos interativos. McRoberts (2011) descreve o Arduino como um dispositivo programável capaz de se comunicar com elementos ligados a ele. Existem vários tipos de sensores e componentes compatíveis com Arduino, capazes de fornecer dados de temperatura, intensidade luminosa, umidade, entre outros.

O Arduino é encontrado em vários modelos atualmente. No site da desenvolvedora, podem ser encontradas informações sobre vários modelos, como: Arduino Uno, Arduino Nano, Arduino Mega, Arduino Leonardo, Arduino Micro, Arduino Zero, entre outros.

A placa de desenvolvimento escolhida para o projeto é a Arduino Nano, exemplificada na Figura 2.3.1. Ela possui o microcontrolador ATmega328, uma memória flash de 32 kB, sendo 2 kB destinados ao bootloader e conta com 8 entradas analógicas e 14 entradas digitais de comunicação.

FIGURA 2.3.1 - ARDUINO NANO



Fonte: Store Arduino (2022)

A programação para Arduino é feita em um ambiente de desenvolvimento integrado (Integrated Development Environment - IDE) próprio, o Arduino IDE, no qual podem ser desenvolvidos códigos em linguagem C e C + +. Todos os modelos de Arduino são compatíveis com essa mesma IDE.

O Arduino conta com entradas digitais e analógicas de comunicação. Uma entrada digital pode assumir apenas dois valores, sendo eles: low e high; ou, 0 ou 1. Essa entrada opera como um interruptor, através do qual um led pode ser ligado e desligado, por exemplo. Já uma entrada analógica pode assumir vários valores num intervalo de 0 a 1023, sendo comumente usada para receber dados de sensores.

O Arduino conta ainda com certos protocolos de comunicação, dentre os quais cabe destacar o I2C e o SPI.

O protocolo I2C (Inter Integrated Circuit) descreve um barramento de comunicação serial com apenas dois fios, os quais são o SDA (Serial Data) e SCL (Serial Clock). É muito utilizado em microcontroladores, pois tem como vantagem seu baixo custo e simplicidade.

Já o protocolo SPI (Serial Peripheral Interface) descreve um barramento de comunicação serial com quatro fios. Tais fios são: MISO (Master Input Slave Output), MOSI (Master Output Slave Input), SCK (Serial Clock) e SS (Slave Select). Esse sistema de comunicação consiste num gerador de sinal (mestre – master) e em um ou mais receptores desse sinal (escravo – slave), de forma que cada comunicação ocorre de forma individual entre o master e um dos slaves.

2.4. Bloco de monitoramento

O cartão de memória é um dispositivo capaz de fazer a leitura e a gravação de dados. Ele permite com que dados sejam escritos, apagados e escritos novamente por um vasto número de vezes. Dentre suas características, cabe destacar que o cartão de memória não necessita de alimentação para manter os dados gravados na memória, uma vez que armazenado o arquivo, o cartão pode ser removido do dispositivo sem que exista perda de dados. Outra características importante do cartão de memória é o seu tamanho reduzido. Embora existam vários formatos e fabricantes, cabe destaque ao cartão micro SD pelas suas dimensões diminutas e vasta capacidade de armazenamento.

Um adaptador micro SD é um dispositivo que permite a comunicação entre o Arduino e o cartão micro SD. A comunicação é realizada através do protocolo SPI.

Com o adaptador de cartão micro SD é possível armazenar os dados lidos do sensor de pressão atmosférica, o qual facilita a leitura dos mesmos, pois podem ser

facilmente acessados através de um computador ou qualquer outro dispositivo que tenha uma entrada para cartão micro SD.

O microcontrolador é embutido com código próprio para recuperação, do qual é acionado quando recebe uma leitura específica de altitude pré-programada. Dessa forma, após o apogeu ser registrado o dispositivo aciona um sinal digital por uma das portas digitais do microcontrolador. Este sinal aciona a queima de um skib elétrico comercial (o mesmo usado para o lançamento de fogos de artifício), gerando um impulso para fora na placa, do qual fecha o orifício entre a cavidade interna e externa do foguetemodelo. O paraquedas deve ser constituído de material leve e densidade baixa, a fim de produzir o impulso necessário para desaceleração do minifoguete durante e após a queda.

Quando é registrada a atitude final, o dispositivo inicia o acionamento sequencial de um bip, a fim de apresentar a localização do minifoguete para recuperação. Esses bips permanecem acionados até a bateria sem esgotar ou que seja pressionado botão de reiniciar localizado centralmente na placa do microcontrolador.

2.5. Sobre minifoguetes

A história dos foguetes surgiu muito antes dos lançamentos e voos das primeiras missões espaciais Apollo e Sputnik. Os primeiros protótipos de foguetes desenvolvidos com estrutura similar às que conhecemos hoje tinham finalidade bélica como os bastões de fogo chinês ou os baionetas descritas por Rogers Bacon, ambos do século 13. Com a evolução da tecnologia os foguetes foram aperfeiçoados até chegarmos aos módulos atuais de Newton a Tchaikovski, do canhão aos Vergeltungswaffe 2 alemães, assim, modelos matemáticos e os primeiros protótipos possibilitam hoje a confecção caseira com uso de materiais com baixa densidade e recicláveis que contribuem para a prática de foguetemodelismo, conforme descrito no artigo “Guia de foguetes - uma história pictórica de foguetes” (em tradução livre do inglês *Rockets Guide - A Pictoral History of Rockets*).

Nessa modalidade podemos subdividir os minifoguetes partes:

- **Nariz**: partes superior do foguete usado para proteger e reduzir arrasto aerodinâmico.

- **Tubo-foguete**: o tubo do qual estão inseridos os componentes eletrônicos e cargas úteis do foguete.
- **Tubo-guia**: tubo externo acoplado ao tubo-foguete que auxilia na estabilidade do foguete antes e durante o lançamento.
- **Tubo-motor**: tubo contra o pendente do foguete usado para impulsionar o foguete.
- **Empenas**: são as asas do foguete que auxiliam na sustentação e estabilidade parte onde a injeção dos gases durante a queima.

Podemos classificar em dois tipos os protótipos, segundo a Associação Brasileira de Foguetes, que resume a composição dos materiais e motores usados:

- **Minifoguetes experimentais**: são módulos construídos com material metálico ou com grande densidade com o objetivo de estudos práticos de foguetes passando de altitude de 12 km e/ou com motores de uso não comercial.
- **Foguetes Modelos**: são modelos feitos com materiais de baixa densidade como borracha e papelão com motores que voam até 12 km e usam motores comerciais com exceção de alguns modelos.

Os motores usados para o protótipo são classificados com a força de empuxo gerada e podem chegar ao ao altitudes específicas para minifoguetes, segundo National Association Rockets (NAR):

TABELA 2.5.1 - APOGEUS MÁXIMOS REGISTRADOS POR CATEGORIA DE MOTOR-FOGUETE:

CATEGORIA	RECORDE (m)	NAR (m)
Classe ⅛ A	49	84
Classe ¼ B	133	121
Classe ½ A	136	319
Classe A	185	347
Classe B	336	484
Classe C	444	673
Classe D	751	856
Classe E	723	1639

Fonte: *Brazilian Association of Rocketry* (2018).

Os dados de altitude da Tabela 2.5.1 referem-se a registros realizados com o altímetros de bordo dados do "Recorde brasileiro de minifoguetes" do Brazilian Association of Rocketry, dessa forma o presente projeto visa não apenas contribuir para a prática de foguetemodelismo, como também apresentar um modelo de altímetro de fácil operação e montagem.

3. DESENVOLVIMENTO

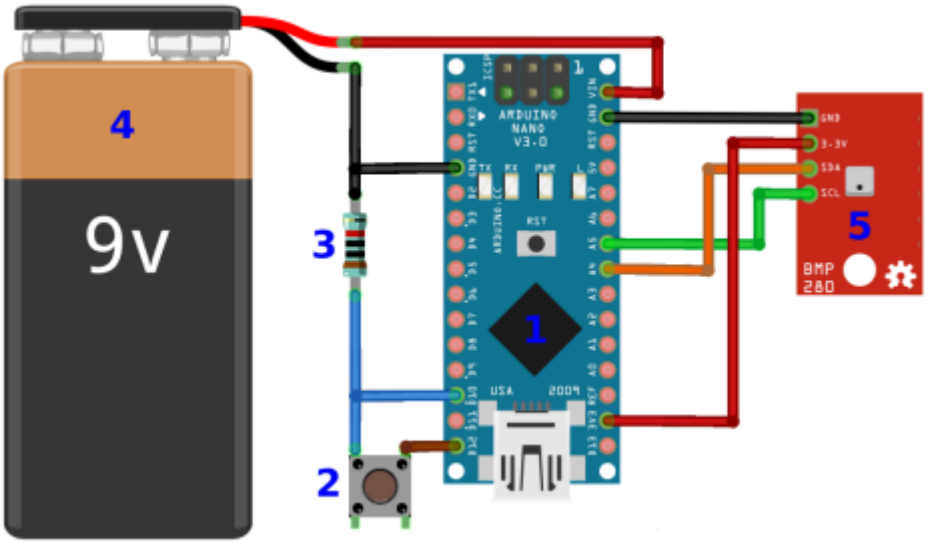
3.1. Confecção do projeto

3.1.1. Motivações

O presente projeto motiva-se no desenvolvimento de um altímetro portátil a foguetes de pequeno porte, pois, são os modelos mais comuns construídos pelas equipes desta modalidade no país. Dessa forma, o presente projeto é capaz de registrar os dados de altitude através do sensor barométrico BMP280 e armazenamento deles em um cartão de memória SD.

Este projeto também visa contribuir para o aperfeiçoamento do modelo desenvolvido anteriormente pela equipe de foguetes Tsiolkovsky da Universidade Tecnológica Federal do Paraná (UTFPR) do câmpus Francisco Beltrão, em um modelo h-Rocket, conforme apresentado na imagem abaixo, é composto por um botão, o microcontrolador Arduino, o sensor BMP280 e a bateria de 9 volts como alimentação. O projeto registra a leitura a cada meio segundo (2 Hz) e inicia a gravação dos dados quando detecta a aceleração de 10 metros por segundo ao quadrado em um período de 2 segundos. Além disso, ele é capaz de registrar o apogeu de até 6.000 metros de altitude e utiliza a memória interna do microcontrolador (denominada memória Electrically Erasable Programmable Read-Only Memory), apresentando o apogeu através dos piscar de um diodo emissor de luz (da sigla em inglês Light Emitter Diode) presentes na placa do microcontrolador com o aperto do botão sequencialmente.

IMAGEM 3.1.1: Esquemático eletrônico do altímetro h-Rocket



(1) Arduino Nano (pode ser substituído por outro Arduino ou similar);

(2) Botão de pressão;

(3) Resistência de 10 kΩ;

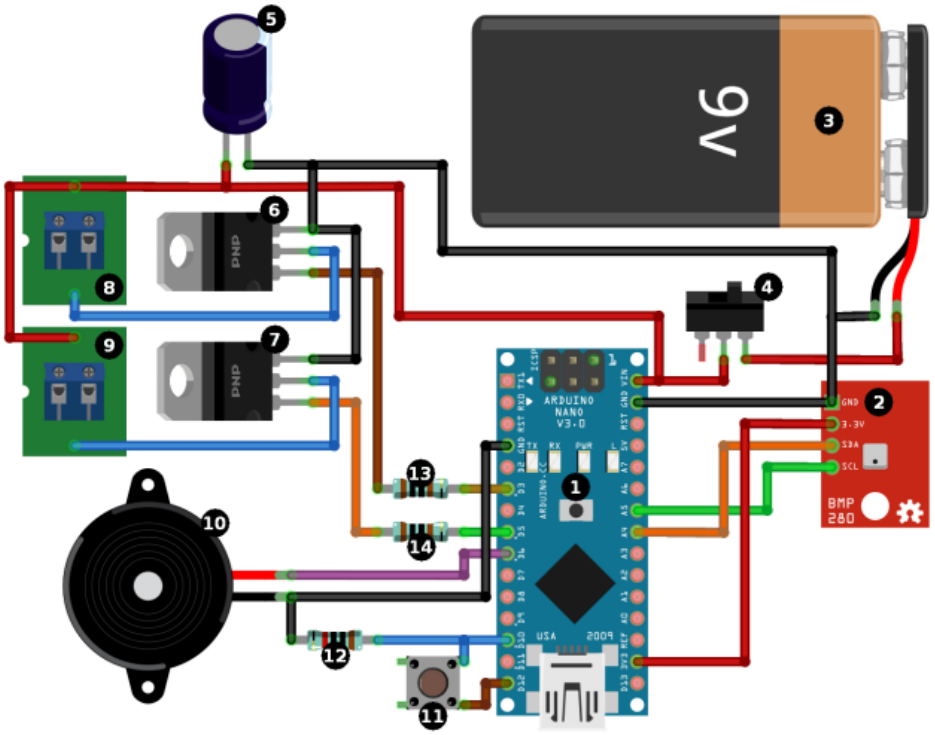
(4) Bateria de 9V (pode ser substitu´ıdo por outra bateria entre 6 e 20V);

(5) Barômetro BMP280;

Fonte: Equipe de foguetes Tsiolkovsky - UTFPR (desconhecido)

De maneira similar ao funcionamento do modelo r-Rocket desenvolvido pela mesma equipe da UTFPR, conforme apresentado na imagem abaixo, do qual opera e registra as altitudes dos protótipos de foguetes utilizando o mesmo sensor BMP280. Porém, nesse caso há uma evolução, pois, é apresentado um modelo com um sistema de recuperação embutido, ou seja, a possibilidade de se operar um paraquedas principal e outro auxiliar, além de possuir um equipamento sonoro. Assim como sua versão anterior, o r-Rocket também possui duas formas de se obter a leitura dos dados: primeiro somente do apogeu, pressionando sequencialmente o botão enquanto o equipamento estiver ligado e não operacional, ou por meio do programa do Arduino IDE. Em ambos os casos existe a possibilidade de apagar a memória do altímetro se pressionar o botão por um tempo pré-determinado.

IMAGEM 3.1.2: Esquemático eletrônico do altímetro r-Rocket



(1) Arduino Nano (pode ser substituído por outro modelo de Arduino ou similar);
(2) Sensor de pressão barométrica BMP280;
(3) Bateria de 9V (pode ser substituído por outra bateria entre 7 e 12V);
(4) Interruptor deslizante;
(5) Capacitor eletrolítico 470µF 16V ;
(6-7) Transistor MOSFET TIP122;
(8-9) Conector Borne de dois terminais;
(10) Alto falante tipo Buzzer 5V ;
(11) Botão de pressão;
(12) Resistor de 10kΩ;
(13-14) Resistor de 1kΩ;

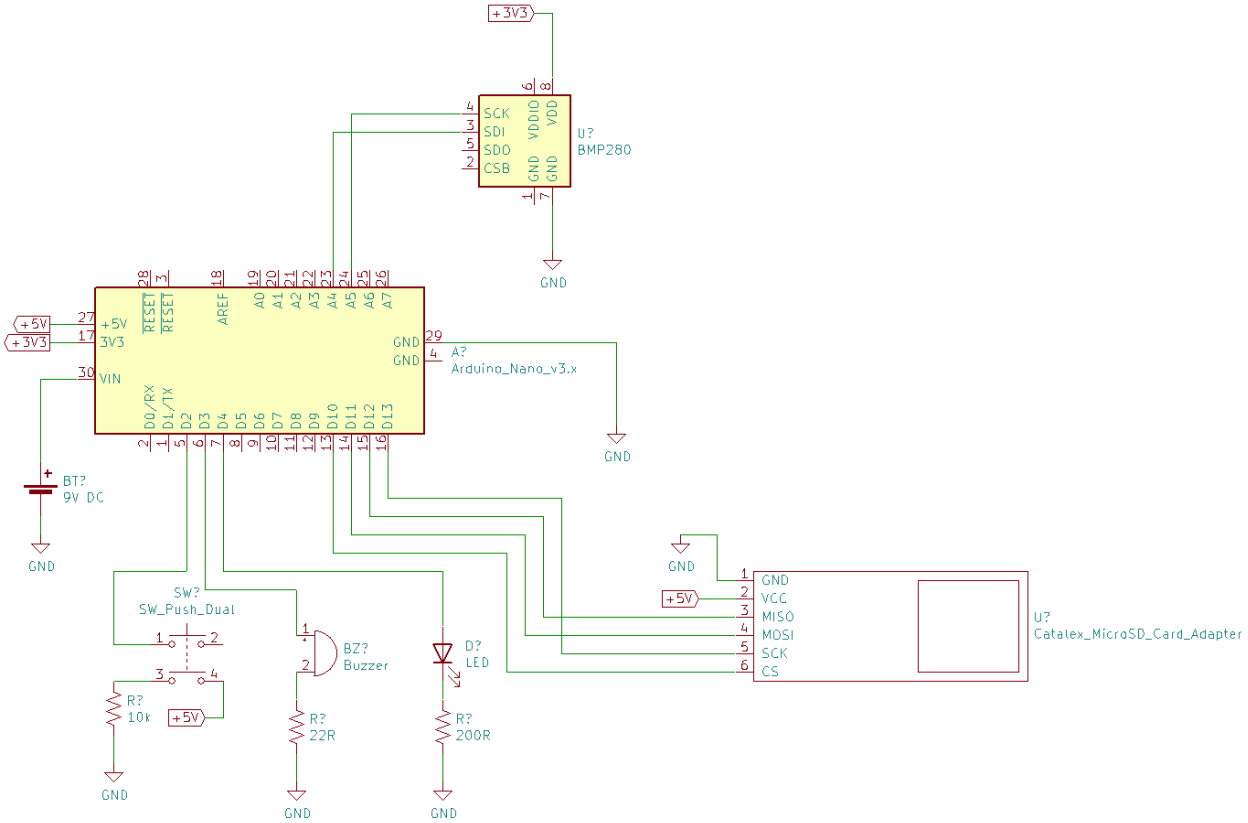
Fonte: Equipe de foguetes Tsiolkovsky - UTFPR (desconhecido)

3.1.2. O protótipo

Como modelo intermediário destina-se o presente projeto, pois este projeto não traz evoluções como o uso exclusivo da memória do microcontrolador. O presente projeto apresenta a coleta de dados através do mesmo sensor BMP280, é composto por um sistema de recuperação com paraquedas principal e dispositivo

sonoro, bem como, tem alimentação por uma bateria de 9 volts de tensão. A evolução consiste em dois pontos. Primeiro, o protótipo utiliza um cartão de memória do qual armazena os dados de cada voo de forma única e independente, em contraste com o h-Rocket e r-Rocket dos quais possuem a capacidade de armazenamento de um único lançamento sobrescrevendo os valores anteriores. Segundo, o sistema de recuperação e início da captação dos dados é ajustável para alturas pré-programadas, dos quais independem da aceleração do protótipo ou a altura mínima, podendo ser acionados os comandos a partir da realidade de cada usuário. Dessa forma, o projeto possui as seguintes como massa de 10,5 gramas sem bateria acoplada.

IMAGEM 3.1.2.1: ESQUEMÁTICO DO PROJETO



Fonte: Os autores (2022)

Conforme descrito anteriormente, no desenvolvimento do protótipo foi necessário o acesso a um computador com qualquer tipo de sistema operacional com plataforma compatível com a plataforma do Arduino IDE e programa também

compatível com a extensão (.csv) instalados no computador para efetuar a leitura dos dados, bem como, o conhecimento em linguagem em programação C++ para efetuar a alteração dos dados, como, por exemplo, a mudança na altitude mínima para começar a leitura dos dados.

3.1.3. Limitações do Projeto

As limitações do projeto observadas consistem em tópicos específicos como:

1. Se for deslocado horizontalmente, os valores lidos permanecem constantes, caso esteja em operação, ou não inicia a contagem, caso parta inicialmente em sentido horizontal.
2. Possui margem de erro de 1 metro de imprecisão na altitude, este erro é oriundo do próprio sensor BMP280, conforme descrito no manual do componente.
3. A leitura dos dados somente ocorre durante a bateria possuir gerar carga mínima para alimentar o circuito e a memória máxima do cartão SD não ser excedida o seu limite.
4. O microcontrolador faz a captação dos dados pela diferença de pressão, isto é, o equipamento não funciona em ambientes fechados mesmo em deslocamento.
5. O altímetro somente pode ser operado em foguetes de pequeno porte cujas dimensões sejam maiores ou iguais ao diâmetro do projeto.

3.1.4 - Ambiente de observação

Em consonância com as limitações, verifica-se que o ambiente de observação consiste em qualquer lugar dentro da superfície do planeta Terra, que apresente uma taxa de variação de pressão atmosférica praticamente linear. Somente foram avaliados as formas de operação sobre um ambiente simulado em bancada e com o projeto montado, com ambiente seco e com baixa luminosidade, sendo ela artificial, com baixíssima ou nenhuma turbulência, exceto quando testado dentro em um sistema simulado, e em temperatura ambiente, aproximadamente 25°C. Quaisquer simulações ou condições de funcionamento que diferem da testada

podem acarretar em mau funcionamento ou dados que diferentes dos valores obtidos em validação.

3.1.5 - Resultados Esperados

Portanto, espera-se que os seguintes resultados sejam obtidos com o projeto:

1. Algoritmo capaz de capturar os dados das altitudes dos pequenos foguetes, de forma similar a do altímetro r-Rocket.
2. Algoritmo deve apresentar apogeu e sistema de recuperação similar ao do r-Rocket fisicamente.
3. Altímetro deve ter dimensões compatíveis com protótipo de foguetes de no mínimo 25 milímetros de diâmetro.
4. Dados coletados devem ser armazenados de forma independente entre si do qual não sejam sobrescritos e devem ser lidos em formato independente do programa do Arduino IDE.

No capítulo 4 verifica-se o método usado para avaliação em comparação e validação dos dados.

3.2. Bloco de código

3.2.1. Ligações dos elementos de hardware

O Arduino Nano conta com oito entradas analógicas e quatorze entradas digitais. As entradas analógicas são numeradas de acordo com a sequência: A0, A1, A2, …, A7; já as entradas digitais são numeradas de acordo com a sequência: D0, D1, D2, …, D13.

O sensor BMP280 é ligado ao Arduino através de quatro fios e utiliza o protocolo I2C de comunicação. Os dois fios de alimentação (VCC e GND) são ligados às entradas 3,3 V e GND do Arduino, respectivamente. Já os dois fios de comunicação (SCL e SDA) são ligados de forma respectiva às entradas A5 e A4 do Arduino.

O adaptador de cartão micro SD é ligado ao Arduino através de seis fios e utiliza o protocolo SPI de comunicação. Os dois fios de alimentação (VCC e GND)

são ligados às entradas 5 V e GND do Arduino, respectivamente, podendo ser usada a entrada 3,3 V de forma alternativa para a alimentação positiva. Já os quatro fios de comunicação são ligados ao Arduino da seguinte maneira: o fio CS é ligado à entrada D10, o fio MOSI é ligado à entrada D11, o fio MISO é ligado à entrada D12 e o fio SCK é ligado à entrada D13.

O projeto conta com o uso de um buzzer (elemento responsável por emitir sons) do tipo ativo, que necessita apenas ser alimentado para que comece a sonorizar. Esse elemento é ligado à entrada D3 e GND do Arduino e em série com um resistor de 22 ohms para limitar a passagem de corrente elétrica pela porta digital, medida em cerca de 20 mA.

Um botão foi inserido no projeto para que fosse possível realizar alguns comandos no altímetro. O botão é ligado à porta digital D3, ao GND através de um resistor de 10 kΩ e à entrada de 5 V do Arduino. A ligação ocorre de tal forma que, enquanto o botão não é pressionado, a entrada D3 permanece ligada em série com o resistor de 10 kΩ ao GND, e a entrada de 5 V permanece isolada. Ao pressionar o botão, a porta D3 recebe um sinal de nível lógico alto da entrada 5 V.

O projeto conta ainda com um LED (do inglês light-emitting diode - diodo emissor de luz) que simula a abertura do paraquedas. Esse LED é ligado à entrada D4 em série com um resistor de 200 ohms e à entrada GND do Arduino. Quando acionada a entrada que realiza a ignição do dispositivo que projeta o paraquedas para fora da cápsula do foguete, o LED acenderá indicando que ocorreu a abertura do paraquedas.

A alimentação do Arduino Nano pode ser feita de duas maneiras: através da porta USB de 5 V ou de uma bateria com um intervalo de tensão de 9 V a 12 V. Para usar a alimentação por bateria, basta ligar o pólo positivo da bateria à entrada VIN do Arduino e o pólo negativo da bateria à entrada GND do Arduino.

3.2.2. Código do altímetro

O código desenvolvido para o altímetro conta com várias bibliotecas, responsáveis por permitir a comunicação entre o sensor BMP280 e o Arduino, bem como pelo correto funcionamento do adaptador de cartão micro SD. Conta também com algumas funções criadas para tornar o código mais eficiente.

3.2.2.1. Bibliotecas usadas no código

Durante o desenvolvimento do código, foi necessário inserir algumas bibliotecas ao mesmo. São elas: SD.h (biblioteca responsável pela comunicação com o cartão micro SD), SPI.h (biblioteca responsável pela comunicação SPI com o adaptador de cartão micro SD), Wire.h (biblioteca responsável pela comunicação I2C com o sensor BMP280), string.h (biblioteca para manipulação de strings - sequência de caracteres), math.h (biblioteca que contém várias funções matemáticas) e Adafruit_BMP280.h (biblioteca responsável pela comunicação com o sensor BMP280).

3.2.2.2. Tarefas realizadas pelo código

O código do altímetro foi desenvolvido visando realizar uma série de tarefas. Inicialmente, são coletados dados de pressão atmosférica, temperatura e altitude através do sensor BMP280; em seguida, esses dados são armazenados em um cartão micro SD; logo após, ocorre o acionamento do paraquedas; em seguida, é iniciado o alarme de localização; e por fim, é indicada a altitude de apogeu alcançada pelo foguete.

3.2.2.3. Funções presentes no código do altímetro

Buscando tornar o código do altímetro mais eficiente, algumas funções foram criadas para realizar tarefas simples, sendo algumas delas repetidas durante o funcionamento. Essas funções são explicadas a seguir.

3.2.3.1 Emitir uma sequência de bips

A seguinte função na Figura 3.2.3.1 recebe um número inteiro como parâmetro, e emite o valor desse número em uma sequência de bips ao acionar o buzzer através da entrada D3 do Arduino. Ao terminar a sequência de bips, a função não retorna nenhum valor. Caso o parâmetro seja o número zero, o bip emitido será

longo. Para os demais números inteiros positivos, será emitida uma sequência de bips curtos.

Figura 3.2.3.1 - Função para emitir uma sequência de bips

```
void buzzerBeep(int beep) { // Função para emitir mensagens por bip
  int i = beep;
  for(; i>0; i--) {
    digitalWrite(buzzer, HIGH);
    delay(200);
    digitalWrite(buzzer, LOW);
    delay(200);
    if(i == 1) {
      break;
    }
  }
  if(i == 0) {
    digitalWrite(buzzer, HIGH);
    delay(500);
    digitalWrite(buzzer, LOW);
    delay(200);
  }
  delay(500);
}
```

Fonte: Os autores.

3.2.3.2. Alarme de localização

A seguinte função na Figura 3.2.3.2, quando chamada, emite bips ininterruptamente até que o botão seja pressionado ou até que toda a energia da bateria de alimentação seja consumida. Se o botão for pressionado durante a execução desta função, uma outra função será chamada para indicar a altura do apogeu alcançado pelo foguete através de uma sequência de bips.

Figura 3.2.3.2 - Função para emitir um alarme de localização

```
void buzzerAlarm() { // Função para localizar o dispositivo
  while(1) {
    digitalWrite(buzzer, HIGH);
    delay(80);
    digitalWrite(buzzer, LOW);
    delay(80);
    if(digitalRead(pushButton) == HIGH) { // Indica a altura do apogeu
      apogeeBeep();
    }
  }
}
```

Fonte: Os autores.

3.2.3.3. Altura do apogeu

A seguinte função na Figura 3.2.3.3 recebe como parâmetro a altitude máxima alcançada pelo foguete (apogeu) obtida através do sensor BMP280. Essa função obtém o número inteiro das unidades de milhares, centenas, dezenas e unidades de metros do apogeu e, com o auxílio da função que emite bips, informa a altitude alcançada pelo foguete durante o voo.

Supondo que a altitude alcançada pelo foguete durante o voo seja de 1024 metros, a função obterá o número de milhares de metros (1 no exemplo) e passará esse valor como parâmetro para a função que emite bips, um bip curto será emitido e ocorrerá uma pausa curta. Logo após, a função obterá o número de centenas de metros (0 no exemplo) e passará esse valor como parâmetro para a função que emite bips, um bip longo será emitido e ocorrerá uma pausa curta. Logo após, a função obterá o número de dezenas de metros (2 no exemplo) e passará esse valor como parâmetro para a função que emite bips, dois bips curtos serão emitidos e ocorrerá uma pausa curta. Por fim, a função obterá o número de unidades de metros (4 no exemplo) e passará esse valor como parâmetro para a função que emite bips, quatro bips curtos serão emitidos e ocorrerá uma pausa curta. Ao final dessa rotina de bips, a função volta a repetir a sequência novamente, de forma ininterrupta, até o desligamento ou reinício do altímetro.

Figura 3.2.3.3 - Função para emitir a altura do apogeu por bips

```
void apogeeBeep() { // Indica a altura do apogeu por bips
  while(1) {
    delay(500);
    float height = apogee; // Recebe o apogeu
    height /= 1000.0;  // Obtem a parte inteira de milhar
    buzzerBeep((int)height); // Envia a parte inteira de milhar para bip
    height -= floor(height); // Subtrai a parte inteira de milhar
    height *= 10.0; // Obtem a parte inteira de centena
    buzzerBeep((int)height); // Envia a parte inteira de centena para bip
    height -= floor(height); // Subtrai a parte inteira de centena
    height *= 10.0; // Obtem a parte inteira de dezena
    buzzerBeep((int)height); // Envia a parte inteira de dezena para bip
    height -= floor(height); // Subtrai a parte inteira de dezena
    height *= 10.0; // Obtem a parte inteira de unidade
    buzzerBeep((int)height); // Envia a parte inteira de unidade para bip
    delay(500);
  }
}
```

Fonte: Os autores.

3.2.3.4. Início do dispositivo

A seguinte função na Figura 3.2.3.4 é executada uma vez ao iniciar Arduino e emite um bip longo para indicar que o altímetro está em funcionamento.

Figura 3.2.3.4 - Função para bip de inicialização do altímetro

```
void buzzerStart() { // Indica que o dispositivo está em funcionamento
  digitalWrite(buzzer, HIGH);
  delay(500);
  digitalWrite(buzzer, LOW);
  delay(500);
}
```

Fonte: Os autores.

3.2.4. Coleta de dados

Os blocos de código descritos a seguir localizam-se dentro da função void setup, que é executada apenas uma vez durante a inicialização do Arduino. Nessa

função são inseridas as principais funcionalidades do código elaboradas para serem lidas apenas uma vez durante o funcionamento do microcontrolador.

Ao iniciar o Arduino, um bip longo é emitido para indicar o funcionamento do mesmo. Após, o botão deve ser acionado para dar prosseguimento na rotina do código. O bloco de código responsável por essas etapas pode ser visualizado na Figura 3.2.4.1.

Figura 3.2.4.1 - Inicialização do altímetro

```
Serial.println(F("-------------[ Altímetro ]-------------"));
Serial.println(F("[>] Dispositivo inicializado"));
buzzerStart(); // Indica que o dispositivo está em funcionamento

Serial.println(F("[>] Pressione o botão para continuar"));
while(digitalRead(pushButton) == LOW) { // Aguarda pressionar o botão
  delay(50);
}
```

Fonte: Os autores.

O bloco de código da Figura 3.2.4.2 é responsável por verificar o funcionamento do sensor BMP280 e do adaptador de cartão micro SD. Se o sensor BMP280 não for encontrado, uma sequência de dois bips será emitida regularmente até que o sensor seja localizado. Caso o cartão micro SD não seja localizado, uma sequência de três bips será emitida regularmente até que o cartão seja inserido no adaptador, ou até que o adaptador entre em funcionamento.

Figura 3.2.4.2 - Teste de funcionamento do sensor BMP280 e do adaptador de cartão micro SD

```
while(!bmp.begin(0x76)) { // Aguarda o sensor de pressão
  if(!errorMessageBMP) {
    errorMessageBMP = true;
    Serial.println(F("[!] Sensor de pressão não encontrado"));
  }
  buzzerBeep(2); // 2 bips para sensor de pressão não encontrado
}
while(!SD.begin()) { // Aguarda o SD card
  if(!errorMessageSD) {
    errorMessageSD = true;
    Serial.println(F("[!] SD card não encontrado"));
  }
  buzzerBeep(3); // 3 bips para SD card não encontrado
}
```

Fonte: Os autores.

O bloco de código da Figura 3.2.4.3 é responsável por gerar um nome único para o arquivo a ser gravado no cartão micro SD. Esse nome será composto pelo primeiro número inteiro positivo disponível, acrescido da extensão (.csv), podendo esse arquivo ser aberto diretamente em uma planilha após a gravação. O nome do arquivo será o primeiro disponível na sequência: 1.csv, 2.csv, etc.

Figura 3.2.4.3 - Criando um nome único para o arquivo

```
for(sequence=1;; sequence++) { // Cria um nome único para o arquivo
  if(SD.exists((String)sequence + ".csv")) {
    continue;
  }
  else {
    fileName = (String)sequence + ".csv";
    Serial.println("[>] Nome do arquivo: " + fileName);
    break;
  }
}
```

Fonte: Os autores.

Os blocos de código descritos a seguir localizam-se dentro da função void loop, que é executada inúmeras vezes durante o funcionamento do Arduino. A palavra em inglês loop pode ser traduzida como laço em português e tem como

significado um código que é repetido por inúmeras vezes. Nessa função são inseridos os blocos de código elaborados para fazer uma mesma rotina por várias vezes durante o funcionamento do microcontrolador.

O bloco de código da Figura 3.2.4.4 é um dos mais importantes presentes no código. Ele pode ser dividido em dois blocos, nos quais um deles é executado apenas durante o primeiro loop e o outro é executado tanto no primeiro loop quanto nos demais.

Figura 3.2.4.4 - Leitura e armazenamento dos dados lidos pelo sensor BMP280

```
dataFile = SD.open(fileName, FILE_WRITE); // Abre o arquivo de registro de dados
if (dataFile) {
  if (firstLoopWrite) { // Executado apenas durante o primeiro loop
    firstLoopWrite = false;
    dataFile.println("Tempo(ms);Altitude(m);Pressão(Pa);Temperatura(°C)"); // Escreve o nome das colunas do arquivo
    startAltitude = bmp.readAltitude(1013.25); // Registra a altitude inicial
    buzzerBeep(1); // 1 bip aguarda para iniciar gravação
    while(bmp.readAltitude(1013.25) < startAltitude + 1.5) { // Aguarda enquanto a altitude não excede em 1.5 m o referencial
      delay(1);
    }
    startTime = millis(); // Registra o tempo inicial
    Serial.println(F("[>] Iniciando a gravação dos dados"));
  }
  dataFile.print(millis() - startTime); // Escreve o tempo atual em ms
  dataFile.print(";");
  dataFile.print(bmp.readAltitude(1013.25) - startAltitude); // Escreve a altitude atual em m
  dataFile.print(";");
  dataFile.print(bmp.readPressure()); // Escreve a pressão atmosférica atual em Pa
  dataFile.print(";");
  dataFile.println(bmp.readTemperature()); // Escreve a temperatura atual em °C
  dataFile.close(); // Fecha o arquivo
}
else {
  if(!errorMessage) {
    errorMessage = true;
    Serial.println(F("[!] Falha ao abir arquivo"));
  }
}
```

Fonte: Os autores.

O bloco de código que é executado apenas uma vez é responsável por gravar o cabeçalho das colunas de dados do arquivo, registrar a altitude de referência, iniciar a coleta dos dados lidos pelo sensor BMP280 quando a altitude variar em um metro e cinquenta centímetros acima do referencial e dar início a contagem de tempo de gravação dos dados.

Já o bloco de código que é executado por várias vezes durante o funcionamento do altímetro é responsável por abrir o arquivo para gravação no cartão de memória micro SD, verificar se o arquivo está disponível para gravação, fazer a leitura e gravação dos dados fornecidos pelo sensor BMP280 e fechar o arquivo de gravação. Esse processo de abrir o arquivo, gravar e fechar o arquivo

ocorre durante todo o tempo de funcionamento do altímetro, de forma que se ocorrer algum problema durante a gravação, os dados gravados permanecem acessíveis no cartão micro SD.

3.2.5. Paraquedas e alarme de localização

Na sequência do código, ainda dentro da função void loop, tem-se o bloco de código contido na Figura 3.2.5.1 responsável por registrar a altitude de apogeu alcançada pelo foguete durante o voo.

Figura 3.2.5.1 - Registro da altitude de apogeu

```
if(apogee < bmp.readAltitude(1013.25) - startAltitude) { // Registra a altitude de apogeu
  apogee = bmp.readAltitude(1013.25) - startAltitude;
}
```

Fonte: Os autores.

O bloco de código da Figura 3.2.5.2 utiliza a altitude de apogeu para abrir o paraquedas quando o foguete descer trinta metros abaixo dessa altitude. Uma variável de controle recebe um valor indicando que o foguete já passou pelo apogeu e está retornando ao solo nesse momento.

Figura 3.2.5.2 - Abertura do paraquedas

```
if(bmp.readAltitude(1013.25) - startAltitude < apogee - 30.0){
  // Ativa o paraquedas ao se afastar do apogeu em 30 m
  flagApogee = true; // O dispositivo passou pelo apogeu
  digitalWrite(parachute, HIGH); // Aciona o paraquedas
}
```

Fonte: Os autores.

O bloco de código da Figura 3.2.5.3 é responsável por iniciar um timer para parar a gravação dos dados cinco segundos após acionar o paraquedas. Logo após finalizada a gravação, é dado início ao alarme de localização do altímetro.

Figura 3.2.5.3 - Finalização da gravação dos dados e início do alarme de localização

```
if(bmp.readAltitude(1013.25) < startAltitude + 20.0 && flagApogee){
  // Ativa o timer para parar a gravação a uma altitude menor que 20 m
  if(firstLoopReturn) { // Inicia o timer para parar a gravação
    firstLoopReturn = false;
    returnTime = millis();
  }
  if(millis() - returnTime > 5000) {
    // Limite de tempo para parar a gravação após retorno ao solo
    Serial.println(F("[>] Gravação finalizada"));
    digitalWrite(parachute, LOW); // Desliga a saída do paraquedas
    buzzerAlarm(); // Inicia o alarme de localização
  }
}
```

Fonte: Os autores.

O último bloco de código visualizado na Figura 3.2.5.4 representa uma camada de segurança caso o sensor BMP280 deixe de funcionar durante o voo do foguete. Após passados cento e vinte segundos do início da gravação, essa é interrompida e é dado início ao alarme de localização, pois estima-se que a viagem do foguete dure apenas alguns segundos, sendo muito provável que após cento e vinte segundos esse já se encontre em solo.

Figura 3.2.5.4 - Interrupção da gravação após 120 segundos

```
  if (millis() - startTime > 120000) {
    // Parar a gravação se exceder o limite de tempo
    Serial.println(F("[>] Gravação finalizada"));
    digitalWrite(parachute, LOW); // Desliga a saída do paraquedas
    buzzerAlarm(); // Inicia o alarme de localização
  }
}
```

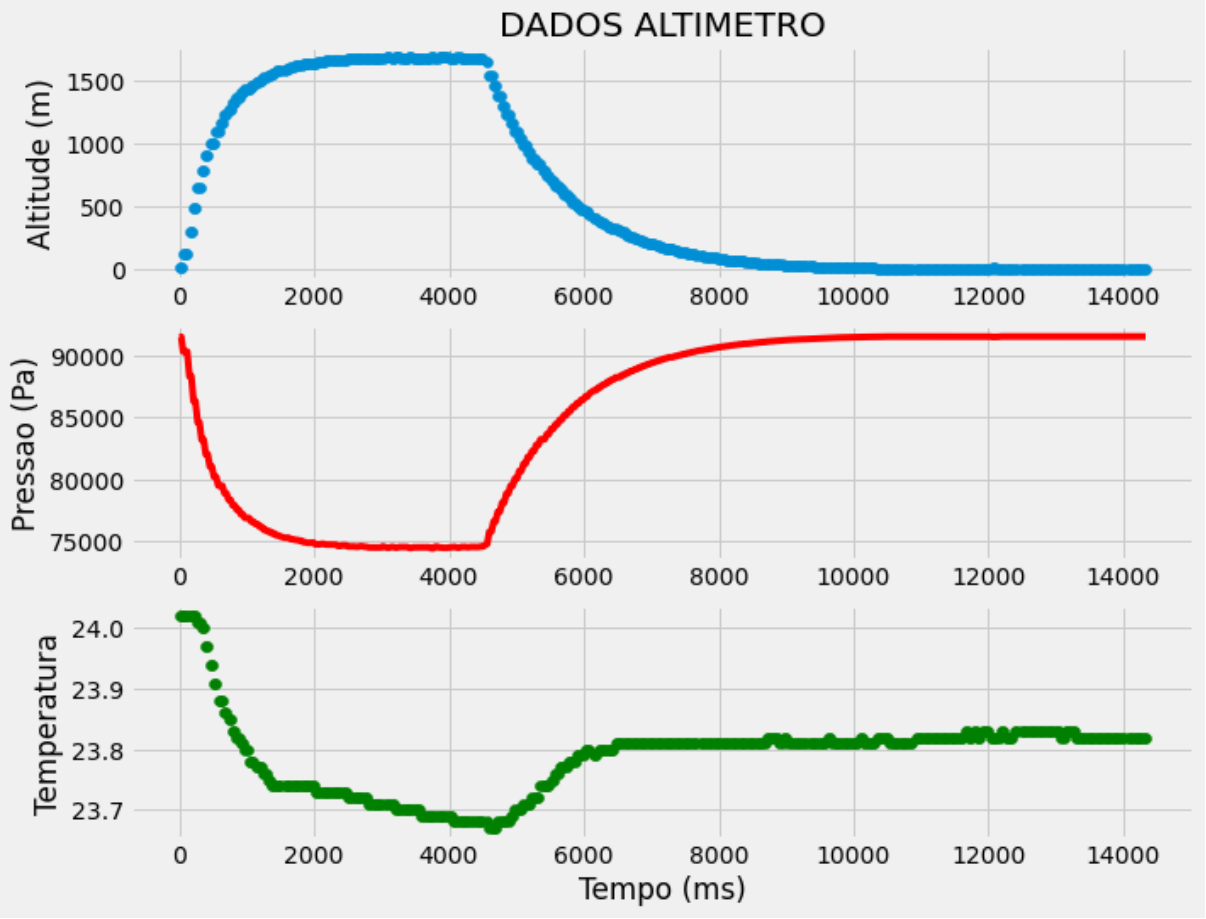
Fonte: Os autores.

4. TESTES E VALIDAÇÃO DOS DADOS

Por ser um protótipo de bancada, não foi possível testá-lo em um lançamento real. Por isso, os testes foram feitos variando a pressão do local, para simular uma mudança de altitude, e variando pequenas altitudes para verificar a precisão de suas medições.

Na Figura 4.1 é apresentado, como referência, dados do altímetro h-Rocket, desenvolvido pela equipe Tchaikovski da Universidade Tecnológica Federal do Paraná, campus Francisco Beltrão, no qual é possível verificar o comportamento gráfico do lançamento através da relação entre pressão e altitude.

Figura 4.1: Dados de simulação dados reais

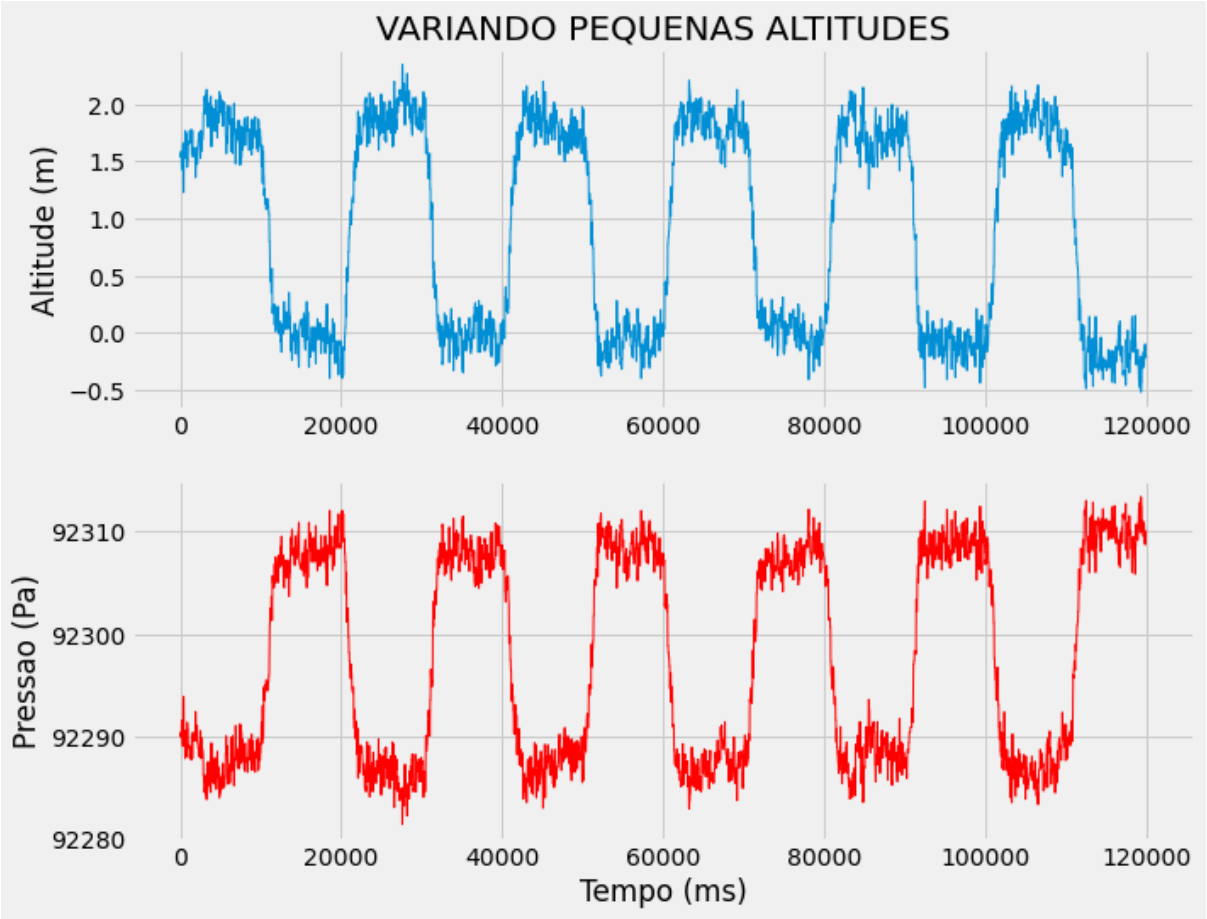


Fonte: Os autores (2022)

4.1. Variando pequenas altitudes

No primeiro teste, são marcados dois referenciais (0 e 2 m), ligamos o protótipo em zero metros e variamos sua altitude de zero a dois metros a cada 10 segundos. No Figura 4.1.1 é possível perceber o mecanismo de segurança do altímetro, o qual só é ativado, e portanto inicia a gravação, quando percebe uma variação de 1.5 m do referencial.

Figura 4.1.1: Variação da pressão para pequenas altitudes



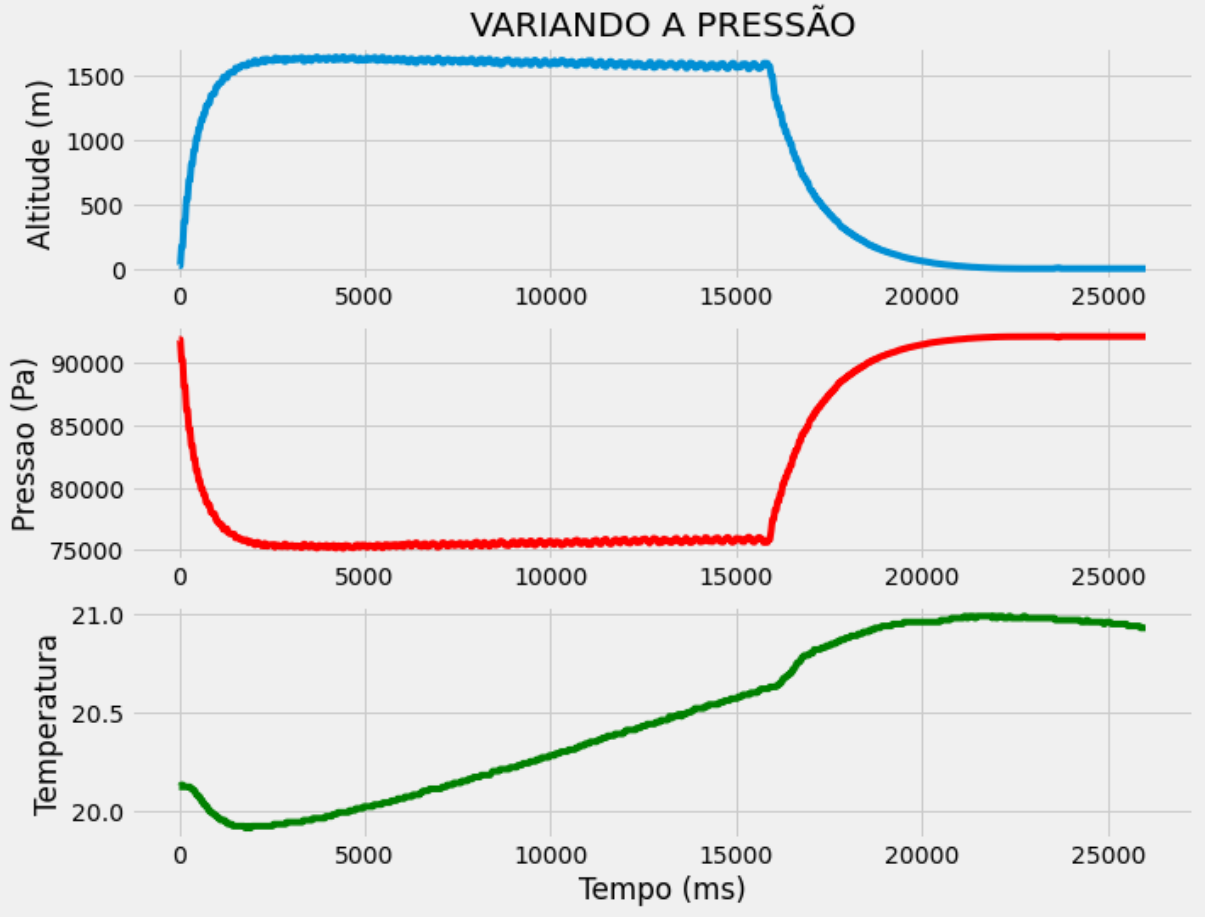
Fonte: Os autores (2022)

Analisando os dados lidos pelo sensor, constatamos que em média o sensor tem variação de ± 5.7 cm em relação a altura real. Note que nesse teste, variamos baixas altitudes, em um lançamento real, o foguete atinge altitudes maiores que 2 metros.

4.2 Variando a pressão

Visto que o sensor estima a diferença de altitude em relação ao seu referencial, calculando a mudança de pressão do meio, apresenta-se a Figura 4.2.1. Este teste tem como objetivo variar a pressão do local, simulando o lançamento do altímetro. Nele, o principal objetivo é verificar se a variação da pressão será inversamente proporcional à mudança de altitude.

Figura 4.2.1: Variação da pressão, altitude e temperatura em função do tempo



Fonte: Os autores (2022)

Como citado anteriormente, quanto menor a variação de pressão do local, maior será sua altitude. Na figura fica claro essa relação.

5. CONCLUSÃO

Conforme descrito no capítulo anterior o funcionamento do protótipo desenvolvido no presente relatório realiza o objetivo geral do qual é a construção de um altímetro utilizando Arduino. Dessa forma, verifica-se que o resultado obtido com o protótipo foram atingidos por meio de aplicações de técnicas e engenharia para o cumprimento dos objetivos.

Para implementar a coleta de dados de altitude foi utilizado um sistema com sensor barométrico BMP280 com um Arduino nano e uma memória Micro SD. O arranjo foi programado de forma simplificada em linguagem de programação C++ dos quais os dados captados através do sensor sejam registrados em arquivos únicos referente a cada lançamento e no formato de tabela para facilitar o tratamento.

Através do projeto final montado em protoboard, pode-se verificar de que suas dimensões são compatíveis com o mini foguetes e competição tanto em tamanho quanto em peso apesar do uso de placa com conexões do qual é meramente conveniente para o funcionamento do circuito e apresentação do projeto na disciplina oficina de projetos TE 311 no curso engenharia elétrica da Universidade Federal do Paraná. Nele foi utilizado uma "*protoboard" de* 830 furos que pode ser substituída por um arranjo mais robusto utilizando uma placa de fenolite perfurada. Dessa forma, as dimensões finais do projeto consistiram ao invés de 165 mm por 55 mm por 10 mm da marca MB102 referentes a uma placa de ensaio de conexões elétricas com 830 furos podemos ser reduzida para 18,5 x 54,5 x 10 milímetros em uma placa de circuito impresso pro próprio.

No quesito de devolver um algoritmo para a leitura do apogeu destaca-se a sequência na metodologia utilizada com o uso de bibliotecas próprias para a comunicação de memória do micro SD com Arduino e a comunicação via I2C do sensor barométrico com o microprocessador também foram utilizadas as bibliotecas para manipulação de sequência de caracteres e funções matemáticas além de funções próprias desenvolvidas pelos autores Nesse contexto, um projeto armazena os dados do início ao fim do acionamento do sensor e registrar registro apogeu dentro desse intervalo.

Como descrito no parágrafo anterior os dados coletados são armazenados de uma memória externa em formato específico para estruturação de tabelas. Assim, há a possibilidade de tratar os dados com qualquer software, cuja leitura seja realizada no formato "*Comma-separated Value",* bem como, é possível gravar e armazenar uma quantidade significativa de dados sem consumo de memória interna do Arduino de diversos lançamentos separados em arquivos Independentes entre si.

Com tudo o projeto também foi embutido com um sistema de recuperação do qual pode ser acionado via injeção de um paraquedas liberado através de um "*skib"* elétrico e também o uso de um bipe usado para sinalizar o apogeu do último lançamento e avisos quando está preparado para lançamento e recuperação após ele.

6. REFERÊNCIA

DAS, Debashis. *How Does the BMP280 Digital Pressure Sensor Work and how to Interface it with Arduino?*. Disponível em: https://circuitdigest.com/microcontroller-projects/interfacing-bmp280-sensor-with-arduino. Acesso em: 12 ago. 2022.

MIDE. *Air Pressure at Altitude Calculator.* Disponível em: https://www.mide.com/air-pressure-at-altitude-calculator. Acesso em: 12 ago. 2022.

MCROBERTS, M. Arduino básico. Disponível em: https://edisciplinas.usp.br/pluginfile.php/4287597/mod_resource/content/2/Ardu%C3%ADno%20B%C3%A1sico%20-%20Michael%20McRoberts.pdf . São Paulo: Novatec Editora, 2011. Acesso em: 30 jul. 2022.

BMP280 Sensor. Disponível em: https://www.usinainfo.com.br/sensor-de-pressao-arduino/sensor-de-pressao-e-temperatura-bmp280-4902.html

ARDUINO. Disponível em: https://store.arduino.cc/products/arduino-nano. Acesso em: 09 ago. 2022.

NATIONAL ASSOCIATION OF ROCKETRY UNITED STATES. *Model Rocketry Sporting Code*. Disponível em: http://nar.org/pdf/pinkbook.pdf . Acesso em: 12 ago. 2022.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION. *Model Rockets*. Disponível em: https://www.nasa.gov/sites/default/files/atoms/files/rockets-guide-20-history.pdf. Acesso em: 15 ago. 2022.

BRAZILIAN ASSOCIATION OF ROCKETRY. Recordes Brasileiros de Minifoguetes. p. 2 (2018). Disponível em: http://ftp.demec.ufpr.br/foguete/Recordes/2021-06-02_Recordes-BAR-18_resumo.pdf. Acesso em: 12 ago. 2022.

TUTORIAL ALTÍMETRO LAE-P: http://ftp.demec.ufpr.br/foguete/Divulgacao/2016_Tutorial_do_altimetro_LAE-P_Fabi'o-Matos_2016-03-20.pdf . Disponível em 20 mar 2016. Acesso em: 22 ago 2022.

Datasheet BMP280. Disponivel em: https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf Acesso em: 05 set 2022.