

UNIVERSIDADE FEDERAL DO ABC

**Relatório do programa de
Pesquisa Desde o Primeiro Dia (PDPD)**

Pedro Paulo Oliveira Barros

RA: 11201810085

Orientador: Prof. Dr. Diego Paolo Ferruzzo Correa

Engenharia Aeroespacial

CECS - UFABC

**Determinação experimental de consumo de energia
elétrica em foguete de competição**

São Bernardo do Campo - SP

Fevereiro 2019



MINISTÉRIO DA EDUCAÇÃO
Fundação Universidade Federal do
ABC Pró-Reitoria de Pesquisa

Av. dos Estados, 5001 - Santa Teresinha - Santo André/SP - CEP
09210-580 Bloco L - 3.º andar - Fone: (11) 3356.7619
propes@ufabc.edu.br

Carta de apresentação

Santo André, de de 20__.

À Pró-Reitora de Pesquisa,

Profª. Dra. Sônia Maria Malmonge

Encaminho o relatório do(a) aluno(a) (nome do aluno) referente ao projeto de pesquisa junto ao programa de Iniciação Científica na modalidade (PIC, PIBIC, PIBIC-AF, PDPD, PIBITI ou Jovens Talentos) no edital (número do edital).

(descrever o desempenho do aluno durante a pesquisa).

Nome e assinatura do Orientador:

Sumário

1. Introdução	3
2. Objetivos e Metas.....	5
2.1 Meta	5
2.2 Objetivos	5
3. Metodologia	6
3.1 Materiais	6
3.2 Métodos	7
4. Resultados	17
5. Discussão	17
6. Conclusões	19
7. Cronograma	19
8. Referências	20

1. Introdução

Foguetes foram um dos grandes responsáveis pelo conhecimento que temos do Universo e da Terra. A construção dos mesmos exige extrema eficiência e precisão.

Um dos componentes de um foguete de grande importância é seu sistema eletrônico.

Ele é o responsável pelo funcionamento dos sensores, sistemas de recuperação, GPS, comunicação, separação de fases, lançamento, entre diversos outros.

Esse sistema eletrônico necessita de, na maioria dos casos, energia elétrica para funcionar. No caso de foguetes experimentais e de competição construídos pela UFABC Rocket Design, essa alimentação elétrica é proveniente de baterias 9V reutilizáveis.

Esse trabalho, então, tratará da determinação experimental do consumo de energia elétrica em foguetes de competição.

A UFABC Rocket Design participa de diversas competições de foguetes experimentais por ano e os dados e informações de cada lançamento são de extrema importância para melhorias e correção de erros no foguete.

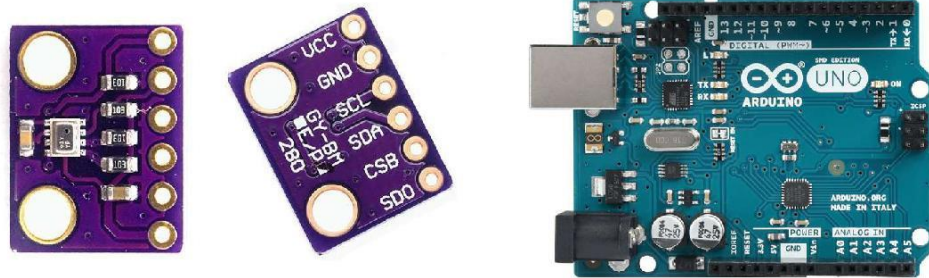


(Foto da equipe atrás de um de seus projetos, 2017; Foguete Boitatá HI)

Todos os foguetes da equipe tem seu sistema eletrônico, que é composto pelo microcontrolador Arduino e sensores, alimentado por baterias 9v reutilizáveis, porém ainda não foi feito um levantamento sobre a autonomia dessas baterias durante o lançamento.

Desse modo, esse projeto focará em determinar os parâmetros energéticos do sistema, para que assim seja possível estabelecer a autonomia dessas baterias.

Essa determinação será feita através de um sensor de corrente não invasivo, que medirá a corrente de um sistema composto por um sensor barométrico BMP-280 conectado a um microcontrolador Arduino UNO REV3.



(BMP-280)

(Arduino UNO REV3)

O Arduino é uma plataforma de prototipagem eletrônica de hardware livre e de placa única que faz uso da linguagem C++, uma linguagem de programação compilada multi-paradigma muito popular.

O BMP280 trabalha com uma tensão de operação de 3V. Sua faixa de medição de pressão é de 300 a 1100hPa, equivalente a +9000 a -500 metros e precisão de

$\pm 0.12\text{hPa}$ ($\pm 1\text{m}$). Seu tamanho também foi reduzido em relação ao seu antecessor (BMP180), por isso foi optado pela sua escolha.

O projeto também contará com o aprendizado de conceitos básicos do Arduino e da linguagem C++.

2. Objetivos e Metas

2.1 Meta

A meta dessa pesquisa é determinar o consumo energético de um sistema eletrônico acoplado a foguetes de competição utilizando o microcontrolador embarcado Arduino.

2.2 Objetivos

Os objetivos são os listados a seguir:

- Entender conceitos básicos da teoria de foguetes;
- Aprender conceitos básicos de Arduino;
- Aprender conceitos básicos da linguagem de programação C++;
- Testar as condições de voo do foguete em laboratório;
- Avaliar a durabilidade de carga do sistema composto por uma bateria de 9V;
- Análise de dados e Validação experimental;

3. Metodologia

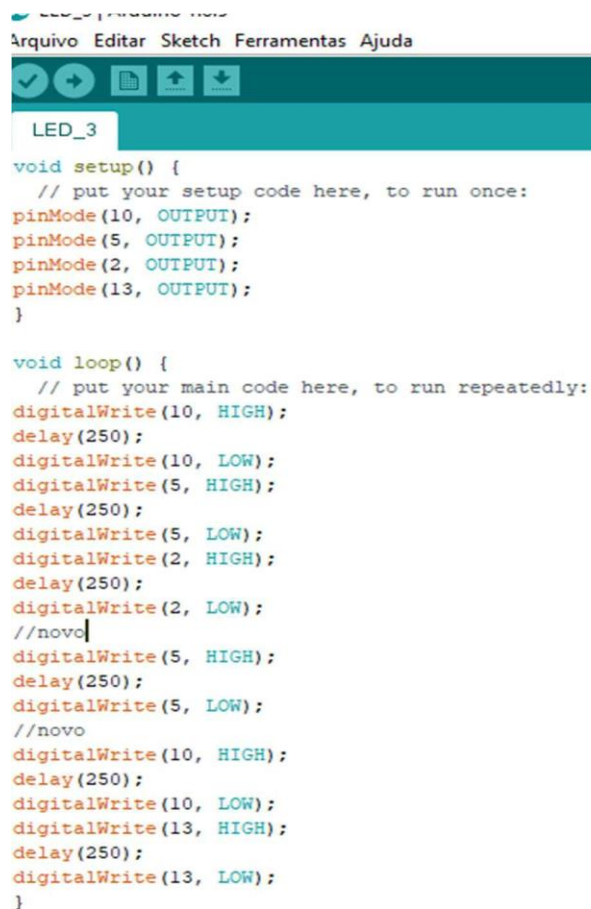
3.1 Materiais

- Computador pessoal
- Arduino UNO REV3
- Sensor de corrente não invasivo (SCT-013-000)
- Sensor de corrente invasivo (I2C INA219)
- Sensor barométrico (CI BMP-280)
- Acelerômetro (MPU 6050)
- Ferro de Solda
- LED's
- Protoboard 830 pontos
- Resistores
- Display LCD 16x2
- Fios Jumpers
- Bateria 9v DURACELL Copperhead 30mAh
- Adaptador de bateria 9v
- Plug P4

3.2 Métodos

A compreensão da linguagem C++ foi feita a partir de fóruns focados em Arduino, tutoriais e ebooks.

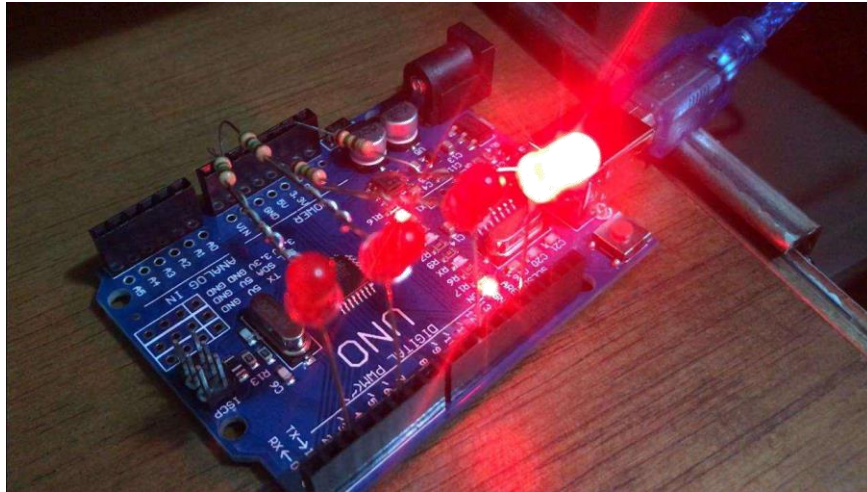
Para iniciação e adaptação a linguagem C++ e ao Arduino, foram utilizados LED's, resistores e a protoboard para a programação de diversos códigos. Um exemplo é o da imagem a seguir:



```
Arquivo  Editar  Sketch  Ferramentas  Ajuda
LED_3

void setup() {
  // put your setup code here, to run once:
  pinMode(10, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(13, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(10, HIGH);
  delay(250);
  digitalWrite(10, LOW);
  digitalWrite(5, HIGH);
  delay(250);
  digitalWrite(5, LOW);
  digitalWrite(2, HIGH);
  delay(250);
  digitalWrite(2, LOW);
  //novo
  digitalWrite(5, HIGH);
  delay(250);
  digitalWrite(5, LOW);
  //novo
  digitalWrite(10, HIGH);
  delay(250);
  digitalWrite(10, LOW);
  digitalWrite(13, HIGH);
  delay(250);
  digitalWrite(13, LOW);
}
```

(Foto do programa carregado para a placa; É possível ver os resistores)

Para maior entendimento da linguagem e de uso de componentes, foram feitos programas envolvendo um Display LCD. Exemplo a seguir:

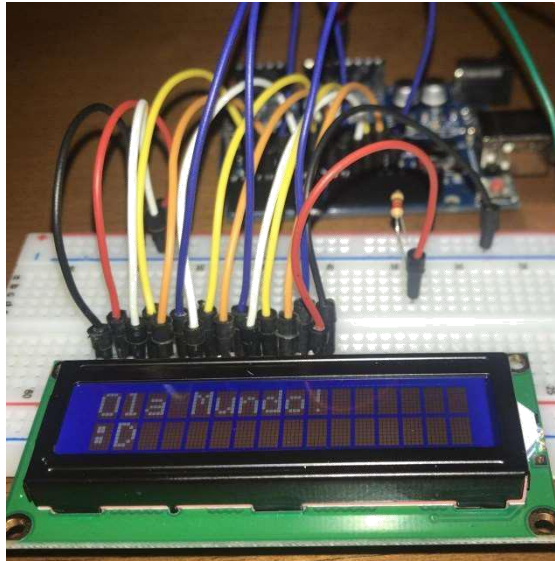
```
sketch_ola_mundo_lcd
#include <LiquidCrystal.h>

LiquidCrystal lcd(2,3,4,6,7,8,9,10,11,12,13);

void setup() {
  // put your setup code here, to run once:
  lcd.begin(16,2);

  lcd.print("Ola Mundo!");
  lcd.setCursor(0,1);
  lcd.print(":D");
}

void loop() {
  // put your main code here, to run repeatedly:
}
```



(Display LCD programado para exibir “Ola Mundo! :D”)

Também foram programados um barômetro e um acelerômetro, utilizados na medição de pressão, temperatura, aceleração e posição no espaço.

As alterações de pressão foram medidas a partir do sensor barométrico submetido a diferença de altitude entre 16 andares.

A variação de temperatura foi medida ao colocar o sensor na geladeira e congelador.

As alterações na posição, aceleração e angulação foram medidas através de simples movimentos no sensor.

Em certo momento, foi escrito um código para ler ambos o sensor barométrico e o acelerômetro. O programa envia as informações atualizadas dos dois sensores ao mesmo tempo para o monitor serial.



```
sketch_feb05b_Juntando_os_dois

#include <Wire.h>
#include "i2c.h"

#include "i2c_BMP280.h"
BMP280 bmp280;

#include <MPU6050_tockn.h>
#include <Wire.h>

MPU6050 mpu6050(Wire);

long timer = 0;

void setup()
{
    Serial.begin(9600);

    Serial.print("Probe BMP280: ");
    if (bmp280.initialize()) Serial.println("Sensor found");
    else
    {
        Serial.println("Sensor missing");
        while (1) {}
    }

    // onetime-measure:
    bmp280.setEnabled(0);
    bmp280.triggerMeasurement();

    Serial.begin(9600);
    Wire.begin();
    mpu6050.begin();
    mpu6050.calcGyroOffsets(true);
}

void loop()
{
    bmp280.awaitMeasurement();

    float temperature;
    bmp280.getTemperature(temperature);

    float pascal;
    bmp280.getPressure(pascal);

    static float meters, metersold;
    bmp280.getAltitude(meters);
```

```

metersold = (metersold * 10 + meters)/11;

bmp280.triggerMeasurement();

Serial.print(" HeightPT1: ");
Serial.print(metersold);
Serial.print(" m; Height: ");
Serial.print(meters);
Serial.print(" Pressure: ");
Serial.print(pascal);
Serial.print(" Pa; T: ");
Serial.print(temperature);
Serial.println(" C");

mpu6050.update();

if(millis() - timer > 1000){

    Serial.println("=====");
    Serial.print("temp : ");Serial.println(mpu6050.getTemp());
    Serial.print("accX : ");Serial.print(mpu6050.getAccX());
    Serial.print("\taccY : ");Serial.print(mpu6050.getAccY());
    Serial.print("\taccZ : ");Serial.println(mpu6050.getAccZ());

    Serial.print("gyroX : ");Serial.print(mpu6050.getGyroX());
    Serial.print("\tgyroY : ");Serial.print(mpu6050.getGyroY());
    Serial.print("\tgyroZ : ");Serial.println(mpu6050.getGyroZ());

    Serial.print("accAngleX : ");Serial.print(mpu6050.getAccAngleX());
    Serial.print("\taccAngleY : ");Serial.println(mpu6050.getAccAngleY());

    Serial.print("gyroAngleX : ");Serial.print(mpu6050.getGyroAngleX());
    Serial.print("\tgyroAngleY : ");Serial.print(mpu6050.getGyroAngleY());
    Serial.print("\tgyroAngleZ : ");Serial.println(mpu6050.getGyroAngleZ());

    Serial.print("angleX : ");Serial.print(mpu6050.getAngleX());
    Serial.print("\tangleY : ");Serial.print(mpu6050.getAngleY());
    Serial.print("\tangleZ : ");Serial.println(mpu6050.getAngleZ());
    Serial.println("=====\\n");
    timer = millis();
}
}

```

(Printscreen do programa que lê os dois sensores)

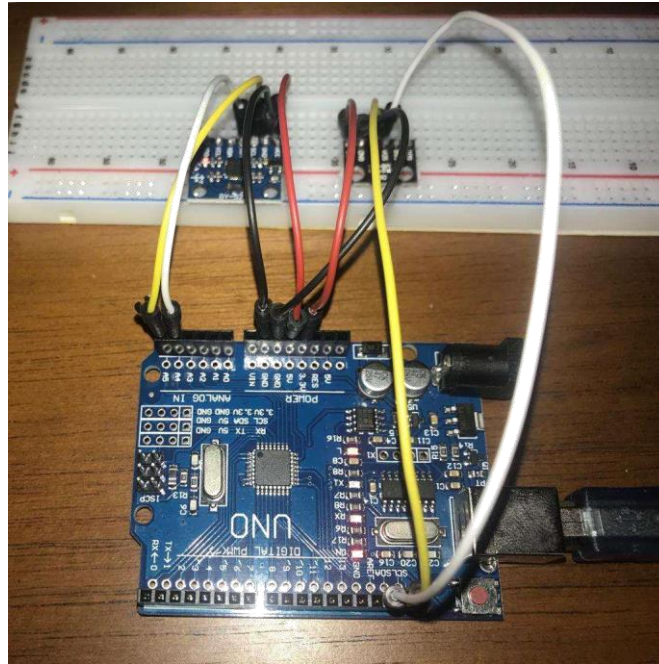
```

temp : 29.02
accX : -0.02   accY : 0.01   accZ : 0.98
gyroX : 0.00   gyroY : -0.03   gyroZ : 0.16
accAngleX : 0.48   accAngleY : 1.02
gyroAngleX : 4.31   gyroAngleY : 5.95   gyroAngleZ : -2.87
angleX : 0.61   angleY : 1.20   angleZ : -2.87
=====

HeightPT1: 787.52 m; Height: 787.75 Pressure: 92214.00 Pa; T: 28.73 C
HeightPT1: 787.51 m; Height: 787.39 Pressure: 92218.00 Pa; T: 28.59 C
HeightPT1: 787.51 m; Height: 787.48 Pressure: 92217.00 Pa; T: 28.63 C
HeightPT1: 787.49 m; Height: 787.30 Pressure: 92219.00 Pa; T: 28.67 C
HeightPT1: 787.46 m; Height: 787.21 Pressure: 92220.00 Pa; T: 28.69 C
HeightPT1: 787.43 m; Height: 787.12 Pressure: 92221.00 Pa; T: 28.70 C
HeightPT1: 787.40 m; Height: 787.03 Pressure: 92222.00 Pa; T: 28.71 C
HeightPT1: 787.39 m; Height: 787.39 Pressure: 92218.00 Pa; T: 28.71 C
HeightPT1: 787.39 m; Height: 787.30 Pressure: 92219.00 Pa; T: 28.73 C
HeightPT1: 787.39 m; Height: 787.48 Pressure: 92217.00 Pa; T: 28.73 C
HeightPT1: 787.39 m; Height: 787.30 Pressure: 92219.00 Pa; T: 28.73 C
HeightPT1: 787.38 m; Height: 787.30 Pressure: 92219.00 Pa; T: 28.74 C
HeightPT1: 787.36 m; Height: 787.21 Pressure: 92220.00 Pa; T: 28.74 C
HeightPT1: 787.36 m; Height: 787.39 Pressure: 92218.00 Pa; T: 28.73 C
=====

```

(Monitor serial mostrando os dados recebidos pelos sensores)



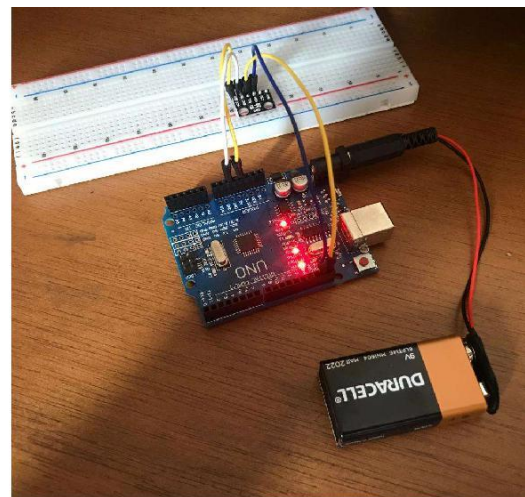
(Conexão dos “jumpers” entre os dispositivos; MPU6050 à esquerda e BMP-280 à direita)

Para que o Arduino pudesse ser alimentado por uma bateria 9v, foi necessário soldar

um adaptador de bateria 9v em um plug P4.



(1)



(2)

1 - Foto do plug P4 já conectado à bateria 9v

2 - Sensor barométrico conectado ao arduino com o programa já carregado e alimentado por uma bateria 9v

A biblioteca para medição da corrente elétrica foi escrita pela EmonLib e contém o programa compatível com o sensor de corrente SCT-013 100A:

```
current_only
// EmonLibrary examples openenergymonitor.org, Licence GNU GPL V3

#include "EmonLib.h"           // Include Emon Library
EnergyMonitor emon1;          // Create an instance

void setup()
{
  Serial.begin(9600);

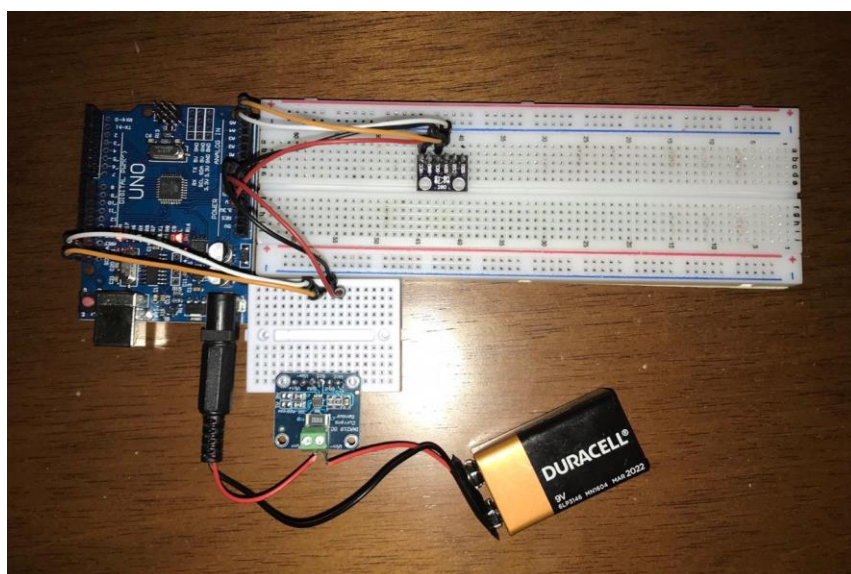
  emon1.current(1, 111.1);      // Current: input pin, calibration.
}

void loop()
{
  double Irms = emon1.calcIrms(1480); // Calculate Irms only

  Serial.print(Irms*230.0);     // Apparent power
  Serial.print(" ");
  Serial.println(Irms);        // Irms
}
```

(Printscreen do programa de medição de corrente na IDE do Arduino)

Após diversas tentativas de medir a corrente do sistema com o sensor de corrente não-invasivo SCT-013, foi notado que este sensor não possuía sensibilidade suficiente para medir uma corrente tão baixa quanto a do Arduino. Por esse motivo, foi adquirido um sensor de corrente invasivo INA219. Desse modo foi possível fazer a medição da corrente do sistema em tempo real enquanto ligado ao computador.



Esses foram os resultados conseguidos através da observação de hora em hora por 26 horas:

Com USB		
Horas	mA	V
22	13,4	8,48
23	12,9	8,46
0	12,3	8,45
1	12	8,43
2	11,2	8,43
3	11,1	8,42
4	12	8,4
5	12,7	8,39
6	11,9	8,38
7	12,4	8,37
8	12,6	8,35
9	11,9	8,34
10	13,8	8,33
11	13,5	8,32
12	11,4	8,31
13	13,3	8,3
14	13,1	8,29
15	12,4	8,28
16	13,6	8,27
17	12,2	8,26
18	13,5	8,26
19	13,2	8,25
20	12,4	8,25
21	13	8,24
22	12,3	8,23
23	12,1	8,22
0	14	8,21

Como o Arduino estava ligado ao computador, esse o alimentou durante a obtenção dos resultados, por esse motivo pode ser vista a oscilação na corrente do sistema. Desse modo, esse teste não foi bem-sucedido.

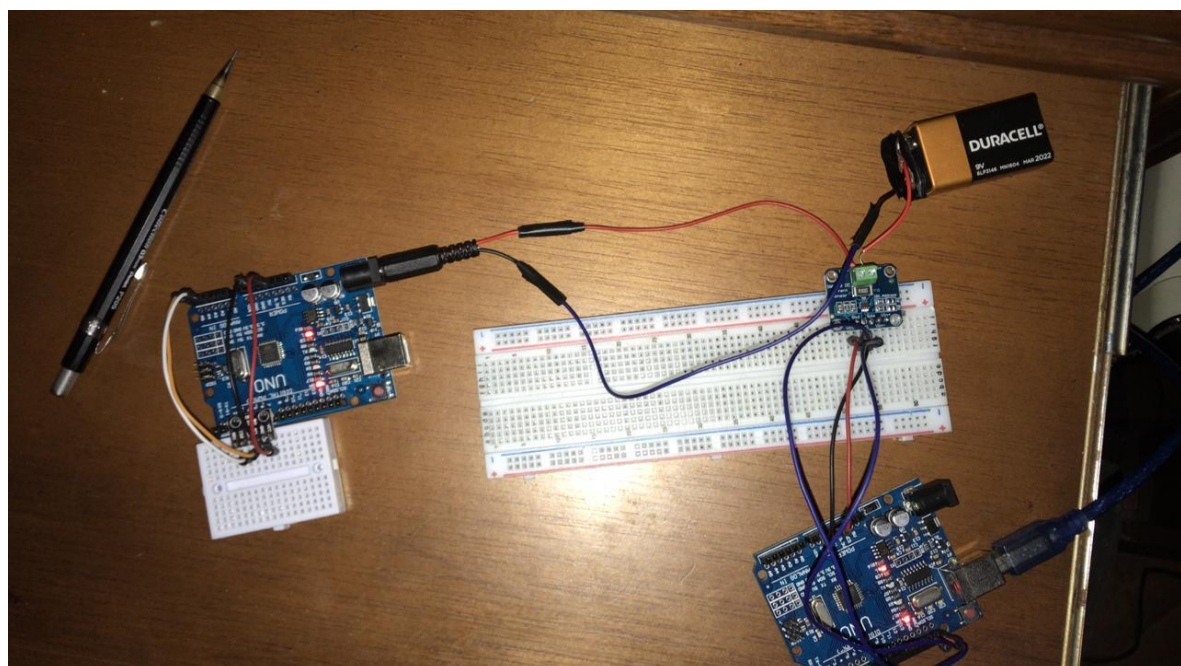
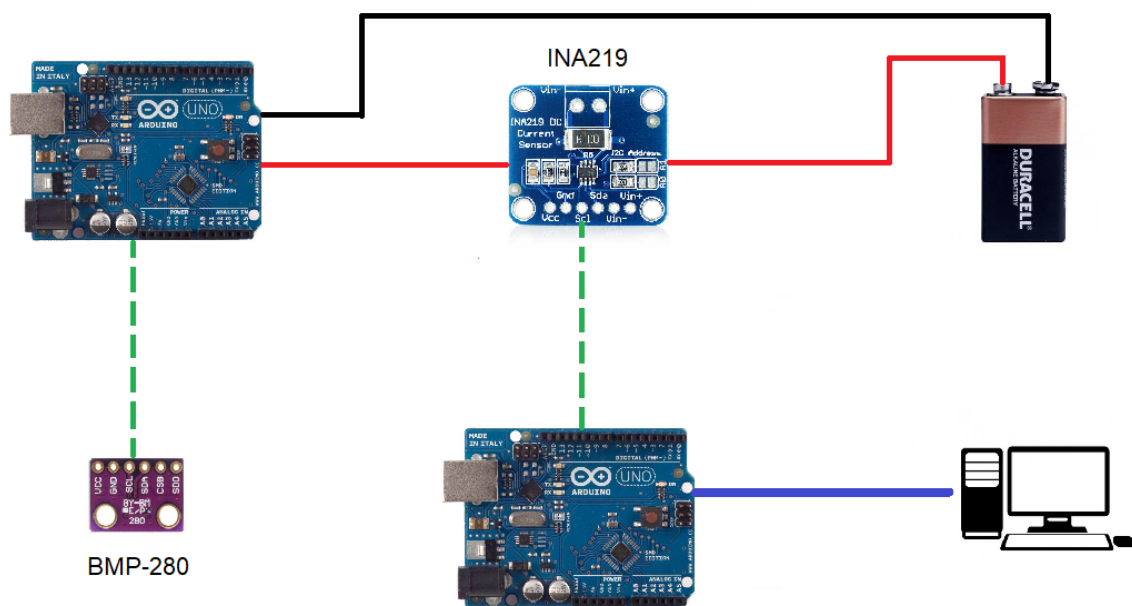
Para evitar esse problema, o teste foi refeito com uma checagem da corrente e voltagem inicial do sistema e bateria, outra checagem as 19 horas de experimento e uma ao ponto em que a energia foi cortada do sistema. Os resultados foram os seguintes:

Sem USB		
Horas	mA	V
0	14,5	9,16
19	12,1	7,2
22	0	5,5

De acordo com a medição inicial, o sistema deveria ser autônomo por aproximadamente 21 horas e 30 minutos e, de acordo com os testes empíricos, ele durou entre 21 horas e 30 minutos e 22 horas. (A checagem de vida da bateria foi feita de meia em meia hora a partir das 19 horas de experimento).

Para uma medição em tempo real das condições elétricas do sistema sem que o mesmo fosse comprometido, foi adquirido um novo Arduino, para um sistema misto. Um Arduino rodaria o programa do BMP-280 e seria alimentado pela bateria 9v. O outro rodaria o programa do INA219 e leria a corrente do sistema do primeira Arduino e a voltagem da bateria 9v enquanto mostrava essas informações em tempo real no monitor serial da IDE do Arduino.

Esquema representado a seguir:

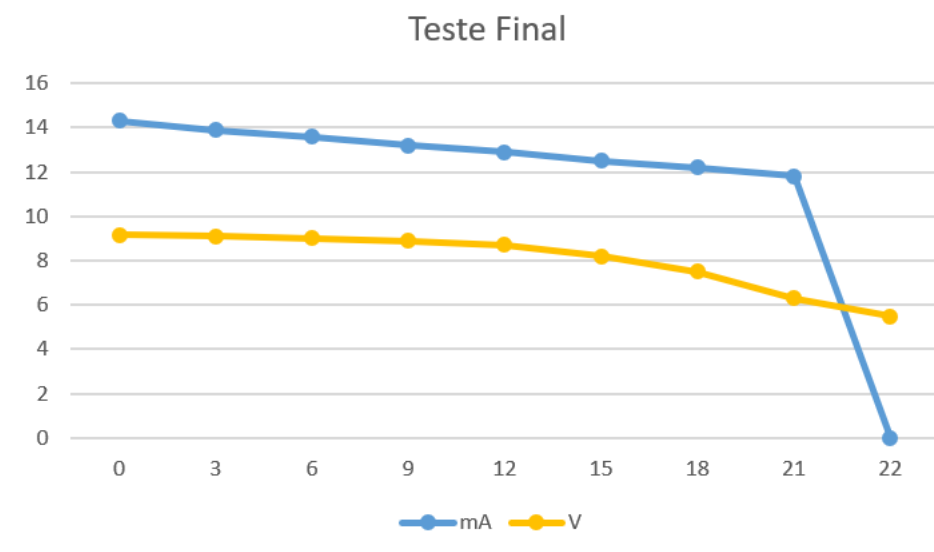


Esse sistema, porém, não resultou em dados plausíveis. Segue dados do monitor serial:

Dois Arduinos		
Horas	mA	V
0	27,2	1,01

Como esse método não foi funcional, foi refeito o teste com checagem de 3 em 3 horas da situação elétrica da bateria e do sistema. Os resultados foram coerentes e estão listados a seguir:

Teste Final		
Horas	mA	V
0	14,3	9,15
3	13,9	9,1
6	13,6	9
9	13,2	8,9
12	12,9	8,7
15	12,5	8,2
18	12,2	7,5
21	11,8	6,3
22	0	5,5



De acordo com esses resultados, foi firmado o resultado de que uma bateria 9v de 1300mAh dura entre 21 e 22 horas quando está alimentando um sistema composto por um Arduino e um sensor barométrico BMP-280.

4. Resultados

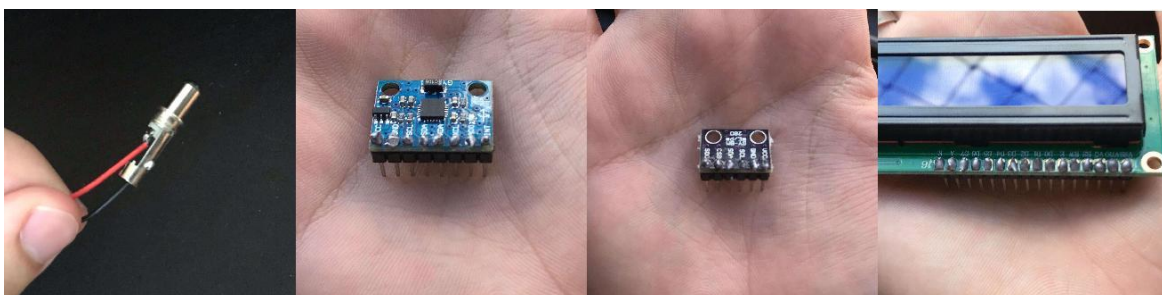
Conceitos básicos do Arduino, da linguagem C++ e da teoria de foguetes foram compreendidos. Também foram adquiridas noções de solda e a programação de sensores e componentes na placa foram bem-sucedidas.

Após diversos métodos de testes e aquisição de dados e a repetição dos funcionais, a duração de uma bateria 9v em um sistema embarcado composto por um Arduino e sensor de pressão BMP280 é de, em média, 21 horas e 30 minutos.

5. Discussão

Houve um imenso avanço na compreensão da linguagem C++, no manuseio do microcontrolador arduino e seus componentes e também em programação em geral. Dessa forma, a conclusão da pesquisa se tornou muito mais fácil, já que essas competências são de extrema utilidade para o projeto.

Para que fosse possível fazer algumas das conexões, certos componentes precisavam ser soldados. Com isso, foi adquirida habilidade em manusear o ferro de solda e o fio de estanho.



(Soldagem do Plug P4) (Soldagem no MPU6050) (Soldagem no BMP-280) (Soldagem no Display LCD)

Ao programar o acelerômetro e o barômetro, eles se mostraram funcionais e precisos. Mudanças mínimas na posição, pressão, angulação, altitude e temperatura foram capturadas pelos sensores.

Para medir a corrente do sistema, foi designado o sensor de corrente não-invasivo SCT-013. Após um longo período tentando compreender o porquê de o sensor não estar captando o campo elétrico, foi descoberto que o mesmo não era um bom instrumento para medir correntes tão baixas como a do Arduino, já que sua faixa de medição era de 0 a 100A. Desse modo foi necessário encontrar um sensor mais apto para o trabalho.

Foi assim, então, que o sensor de corrente invasivo INA219 foi adquirido. Esse, diferentemente do SCT-013, tem uma precisão muito maior em medições de baixas correntes, uma vez que sua faixa é de 0 a 3A. Sendo assim, a medição da corrente e da voltagem do sistema se tornaram viáveis. Foram efetuados diversos testes com o sensor de corrente invasivo INA219, com configurações de conexões diferentes. Na teoria, o teste mais preciso seria o com dois microcontroladores Arduino (como representado na sessão “3.2 Métodos” pág. 15), porém, quando aplicado, não obteve resultados satisfatórios. Por esse motivo o método mais simples de observação de três em três horas do monitor serial foi escolhido como teste final.

6. Conclusões

Todo o aprendizado durante essa pesquisa está sendo e será muito importante para minha vida acadêmica e profissional. Os conhecimentos de programação na linguagem C++, soldagem de peças eletrônicas, fundamentos de elétrica e instalação de sensores no microcontrolador Arduino foram de extrema relevância, além de, principalmente, a responsabilidade adquirida que rege uma pesquisa científica e a maturidade que foi necessária desenvolver para desviar dos obstáculos, focar-se no objetivo e alcançar os resultados propostos.

7. Cronograma

- 1) Revisão Bibliográfica
- 2) Arduino e C++
- 3) Sensores Barométricos e de Corrente
- 4) Teste e Aquisição de Dados
- 5) Análise dos Dados
- 6) Relatório Parcial
- 7) Relatório Final

Atividades	2018			2019							
Meses	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago
1											
2											
3											
4											
5											
6											
7											

8. Referências

THOMSEN, Adilson. O que é Arduino?. 2 Set. 2014. Disponível em: <<https://www.filipeflop.com/blog/o-que-e-arduino/>>. Acesso em: 15 Out. 2018.

What is Arduino?. Disponível em: <<https://www.arduino.cc/en/Guide/Introduction#>> . Acesso em: 15 Out. 2018.

Structure of a program. Disponível em: <http://www.cplusplus.com/doc/tutorial/program_structure/>. Acesso em: 17 Out. 2018.

Variables and Types. Disponível em: <<http://www.cplusplus.com/doc/tutorial/variables/>>. Acesso em: 17 Out. 2018

CAVALIERI, Renan. Como acender e piscar um LED no Arduino. 1 Out. 2017. Disponível em: <<https://www.tecdicas.com/39/como-acender-e-piscar-um-led-no-arduino>>. Acesso em: 25 Out. 2018.

McROBERTS, Michael. Arduino Básico. Primeira Edição. Novatec Editora, 2011.

Use o BMP280 para medir temperatura, pressão e altitude. 26 Abr. 2017. Disponível em: <<https://www.arduinoecia.com.br/2017/04/bmp280-pressao-temperatura-altitude.html>>. Acesso em: 2 Dez. 2018.

NOGUEIRA, Rafael. Arduino: Projecto 28 – Estação Meteorológica com o BMP280. 29 Mai. 2016. Disponível em: <<https://aprenderrobotica.wordpress.com/2016/05/29/arduino-projecto-28-estacao-meteorologica-com-o-bmp280/>>. Acesso em: 2 Dez. 2018.

THOMSEN, Adilson. Tutorial: Acelerômetro MPU6050 com Arduino. 30 Set. 2014. Disponível em: <<https://www.filipeflop.com/blog/tutorial-acelerometro-mpu6050-arduino/>>. Acesso em: 3 Dez. 2018.

THOMSEN, Adilson. Medidor de corrente não invasivo com Arduino. 9 Nov. 2015. Disponível em: <<https://www.filipeflop.com/blog/medidor-de-corrente-sct013-com-arduino/>>. Acesso em: 17 Jan. 2019.

BOSCH SENSORTEC. Disponível em: <https://www.bosch-sensortec.com/bst/products/all_products/bmp280>. Acesso em: 13 Fev. 2019.

MPU-6050 DATASHEET. Disponível em: <<http://pdf1.alldatasheet.com/datasheet-pdf/view/517744/ETC1/MPU-6050.html>> . Acesso em 13 Fev. 2019.

DEMETRAS, Ezequiel. SCT-013 - Sensor de Corrente Alternada com Arduino, 30 Out. 2017. Disponível em: < <https://portal.vidadesilicio.com.br/sct-013-sensor-de-corrente-alternada/>> Acesso em 8 Mai. 2019

Sensor de corrente não invasivo SCT013 com Arduino. Disponível em: < <https://www.dobitaobyte.com.br/sensor-de-corrente-nao-invasivo-sct013-com-arduino/>> Acesso em 10 Mai. 2019

INA219 DATASHEET. Disponível em: < <http://www.ti.com/lit/ds/symlink/ina219.pdf>> Acesso em 12 Jul. 2019.

ADA, Lady. Adafruit INA219 Current Sensor Breakout. Disponível em: < <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ina219-current-sensor-breakout.pdf>> Acesso em 12 Jul. 2019.