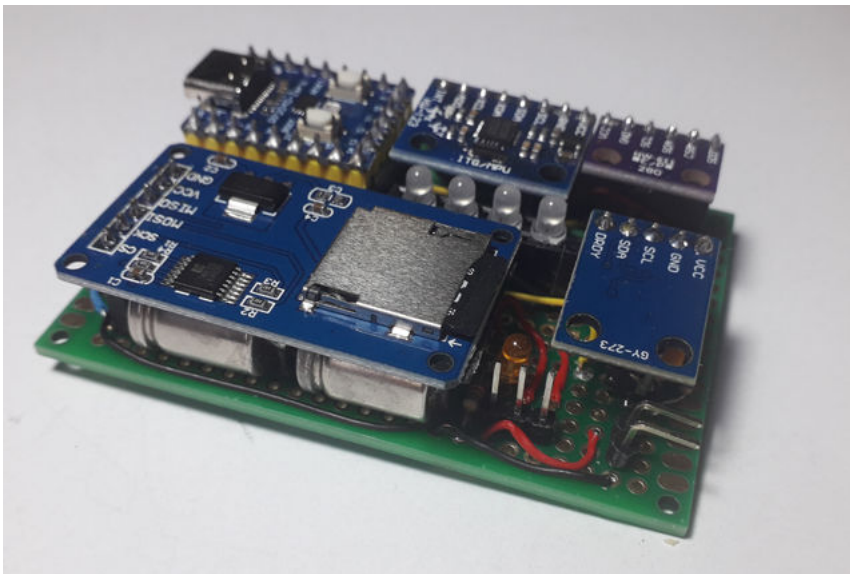


INSTRUMENTAÇÃO EMBARCADA PARA PESQUISA

MSc. Eng. Prof. Antonio Carlos Gasparetti

Este artigo apresenta o desenvolvimento de um sistema de instrumentação embarcada destinado à pesquisa em ambiente atmosférico. O projeto utiliza uma estrutura alimentada por bateria e incorpora sensores capazes de medir campos magnéticos em baixa e alta atmosfera, pressão barométrica, além de contar com um acelerômetro e giroscópio para compensação dos dados de movimentação. No núcleo do sistema está o microcontrolador RP2040 - ZERO, presente no Raspberry Pi Pico, que neste projeto gerencia a aquisição dos sinais e o armazenamento dos dados em um cartão SD. Essa plataforma possibilita o monitoramento preciso e a coleta contínua dos parâmetros ambientais, permitindo aplicações em estudos meteorológicos e experimentos científicos (RASPBERRY PI LTD, 2020, p.10; BOSCH SENSORTEC, 2018, p.7; INVENSENSE, 2013, p.12).dos em balões estratosféricos e satélites (NASA, 2020, p.18).

APLICAÇÕES DA INSTRUMENTAÇÃO EMBARCADA

A instrumentação embarcada tem se mostrado essencial para a realização de medições precisas e contínuas em diversos ambientes de forma autônoma.

Entre as principais aplicações, destacam-se:

Medidas de Anomalia de Campo Magnético

A capacidade de monitorar variações no campo magnético é fundamental para a detecção de anomalias, que podem indicar eventos geofísicos ou perturbações induzidas pela atividade solar. Tais medições possibilitam o estudo da interação entre a atmosfera terrestre e o vento solar, bem como o monitoramento de fenômenos como tempestades geomagnéticas (NOAA, 2021, p.22).

Coleta de Dados Climáticos

A integração de sensores de pressão e temperatura, como o BMP280, permite a obtenção de dados climáticos com alta resolução. Esses dados são cruciais para a análise de condições atmosféricas e para a realização de estudos relacionados às mudanças climáticas e à previsão do tempo. A coleta contínua de informações possibilita a identificação de padrões e tendências no clima (NOAA, 2021, p.22).

Comportamento dos Instrumentos em Situação Estratosférica

Em altitudes elevadas, as condições ambientais podem afetar o desempenho dos sensores. A instrumentação embarcada, ao integrar medições de campo magnético, aceleração e pressão, permite avaliar o comportamento dos instrumentos em condições estratosféricas. Esses dados são essenciais para a validação de modelos e para a calibração de sistemas utilizados em experimentos aeroespaciais, como os realizados em balões estratosféricos e satélites (NASA, 2020, p.18).

1. SISTEMA EM BLOCOS

O sistema é dividido em diversos blocos funcionais, que juntos garantem a aquisição e o processamento dos dados (figura 1).

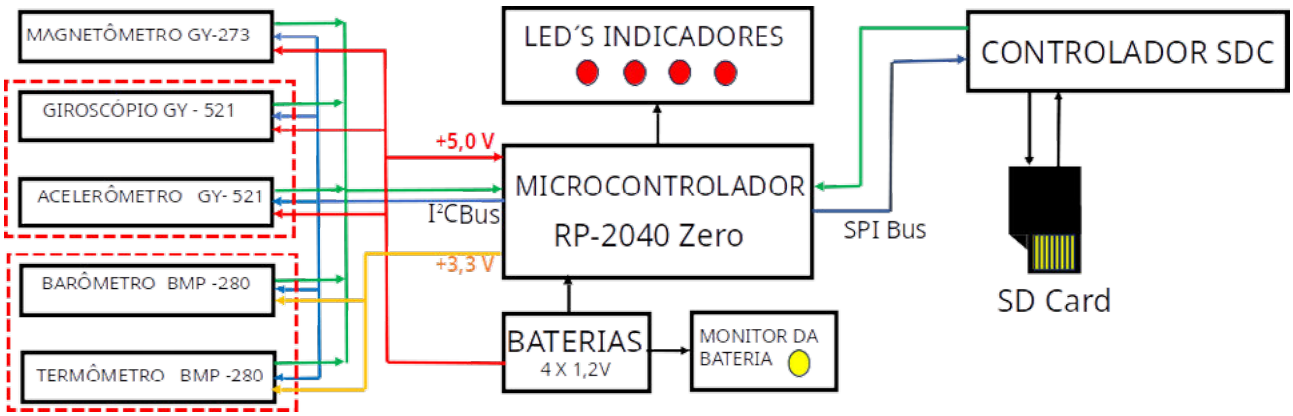


Figura 1 – Diagrama em Blocos do Sistema Embarcado

1.1. BLOCO DE ALIMENTAÇÃO

Fonte de Energia: Uma bateria recarregável que fornece energia ao microcontrolador, aos sensores e ao módulo SD. Existem duas linhas de alimentação, uma para +5 V para os dispositivos que operam em +3,3 V mas tem regulador de tensão próprio de +5 V para +3,3 V e outros dispositivos que trabalham com 3,3 V sem reguladores.

Gerenciamento de Tensão: Um circuito de condicionamento, associado a um canal ADC0 do RP2040, realiza a medição da tensão da bateria, convertendo uma faixa de 0 a 2,5 V para a escala de 0 a 5 V, garantindo a integridade do sistema, bem como evitando a aplicação de tensão maior que 3,3 V na GP26 do microcontrolador

1.2. BLOCO DO MICROCONTROLADOR

RP2040 (Raspberry Pi Pico)

- Capacidades: Possui dois núcleos ARM Cortex-M0+ com clock de até 133 MHz, 264 KB de RAM e 2 MB de memória flash, além de diversas interfaces (I²C, SPI, ADC, PWM etc.) que possibilitam a conexão e o controle de múltiplos dispositivos.
- Configuração: No projeto, o RP2040 gerencia as comunicações I²C (para sensores) e SPI (para o cartão SD) e executa o programa em MicroPython ou, alternativamente, em C (para plataformas compatíveis com Arduino) (RASPBERRY PI FOUNDATION, 2020, p.10).
- A designação “RP2040 Zero” geralmente se refere a uma implementação ou placa que utiliza o microcontrolador RP2040 em um formato compacto e minimalista, similar ao conceito de placas “Zero” que priorizam a redução de custos e dimensões sem comprometer as funcionalidades essenciais. Essa variante pode ser empregada em aplicações que exijam baixo consumo energético, pequena dimensão e alta flexibilidade na integração com diversos sensores e dispositivos periféricos.

1.3. BLOCO DOS SENSORES

Magnetômetro (QMC5883L)

Responsável pela medição dos campos magnéticos; essencial para pesquisas em geofísica e monitoramento atmosférico (MOUSER, 2025, p.5).

Acelerômetro e Giroscópio (MPU6050)

Coleta os dados de aceleração e taxa de rotação, permitindo a compensação de movimentos e a estabilização das medições dos demais sensores (INVENSENSE, 2013, p.12).

Sensor de Pressão/Temperatura (BMP280)

Mede a pressão barométrica e a temperatura, dados fundamentais para análises meteorológicas e de altitude (BOSCH SENSORTEC, 2018, p.7).

As leituras do magnetômetro podem ser influenciadas pela inclinação do sensor e por movimentos dinâmicos, o que pode distorcer a

determinação da direção magnética. Por isso, é comum aplicar técnicas de compensação que utilizam dados do acelerômetro e, em alguns casos, do giroscópio para corrigir os efeitos de inclinação e aceleração. Essa fusão de dados – frequentemente realizada por algoritmos como o filtro complementar ou o filtro de Kalman – permite isolar a componente do campo magnético terrestre de interferências causadas pelo movimento e pela orientação do sensor, resultando em medições mais precisas da direção. (INVENSENSE, 2013, p.12; NASA, 2020, p.18)

1.4. BLOCO DE COMUNICAÇÃO

I²C

Utilizado para a comunicação com os sensores (magnetômetro, MPU6050 e BMP280) através dos pinos designados para dados (SDA) e clock (SCL).

SPI

Empregado para a interface com o cartão SD, garantindo alta velocidade na transferência dos dados coletados.

1.5. BLOCO DE ARMAZENAMENTO

Cartão SD

Armazena os dados coletados em formato CSV. O sistema foi projetado para suportar cartões SD com capacidade que pode chegar a dezenas de gigabytes, conforme a formatação FAT32, permitindo longos períodos de coleta sem a necessidade de transferência frequente dos dados.

1.6. BLOCO DE INDICADORES

LEDs de Erro

Três LEDs indicam, através de códigos binários, o estado da inicialização dos dispositivos. Por exemplo:

- 000 – Sem erros
- 001 – Erro no magnetômetro
- 010 – Erro no acelerômetro
- 011 – Erro no giroscópio
- 100 – Erro no cartão SD
- 101– Erro no Sensor de Pressão/Temperatura
- 110– Erro de calibração do Sensor de Pressão/Temperatura
- 111– Aviso de carga baixa na bateria

LED de Gravação

Um LED específico pisca durante a escrita dos dados no cartão SD, sinalizando as operações de armazenamento.

- 1000– Gravando no cartão SD do sistema está descrito na figura 2.

2. FUNCIONAMENTO DO SISTEMA

O fluxograma do sistema está descrito na figura 2.

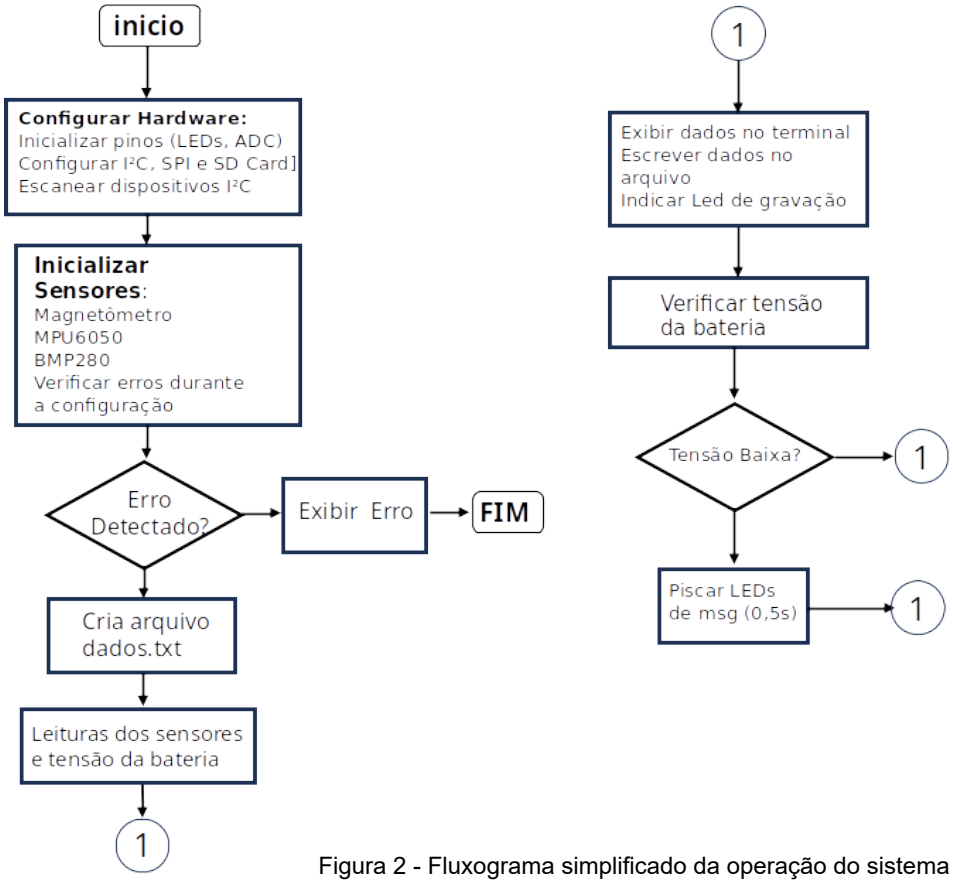


Figura 2 - Fluxograma simplificado da operação do sistema

O processo operacional do sistema segue etapas bem definidas:

2.1 - INICIALIZAÇÃO DOS PERIFÉRICOS

- Ao ligar, o microcontrolador configura as interfaces I²C e SPI.
- Cada sensor é inicializado individualmente. São realizadas leituras de identificação (como o chip ID do BMP280) e configurações de operação, tais como o ajuste de taxas de amostragem e modos de operação.
- Os LEDs de erro são acionados para indicar problemas na comunicação com os dispositivos. Se algum erro for detectado, o sistema exibe o código correspondente por 1 segundo e interrompe a execução para evitar a coleta de dados inválidos.

2.2 - AQUISIÇÃO DE DADOS

O sistema realiza leituras periódicas dos sensores:

- Magnetômetro: Captura os valores dos campos magnéticos nos eixos X, Y e Z.
- MPU6050: Coleta os dados de aceleração e giroscópio, essenciais para a compensação dos efeitos de movimento.
- BMP280: Adquire medições de pressão e temperatura.
- ADC: Monitora a tensão da bateria, permitindo ação corretiva caso a tensão caia abaixo do limiar definido (3,4 V).

2.3 - PROCESSAMENTO E ARMAZENAMENTO

Os dados são organizados em uma linha de CSV com o cabeçalho:

- MagX, MagY, MagZ, AccX, AccY, AccZ, GyrX, GyrY, GyrZ, BMP_T, BMP_P, ADC_V
- Cada linha representa um registro temporal das medições.
- Durante a escrita no cartão SD, o LED de gravação pisca, confirmando a operação.

2.4 - MONITORAMENTO CONTÍNUO E AÇÕES DE ALERTA

- Se a tensão da bateria for detectada abaixo de 4,0 V, os LEDs de erro piscarão em intervalos de 0,5 s para sinalizar a condição de baixa energia.

3 - SAÍDA DE DADOS

Os dados são gravados em um arquivo CSV no cartão SD, que deve estar formatado em FAT32 para compatibilidade.

Formato do Arquivo

- A primeira linha contém o cabeçalho com os nomes dos campos.
- Cada registro subsequente é composto por medições separadas por vírgulas, facilitando a importação para softwares de análise como Excel, MATLAB ou Python.

Capacidade Máxima

- A capacidade de armazenamento depende do cartão SD utilizado, podendo variar de alguns gigabytes até 32 GB ou mais, considerando que o sistema de arquivos FAT32 limita o tamanho máximo do arquivo individual a 4 GB.
- O sistema é adequado para longos períodos de coleta contínua, desde que haja energia suficiente e espaço no cartão.

4 - MONTAGEM DO SISTEMA

O diagrama de conexões está na figura 3.

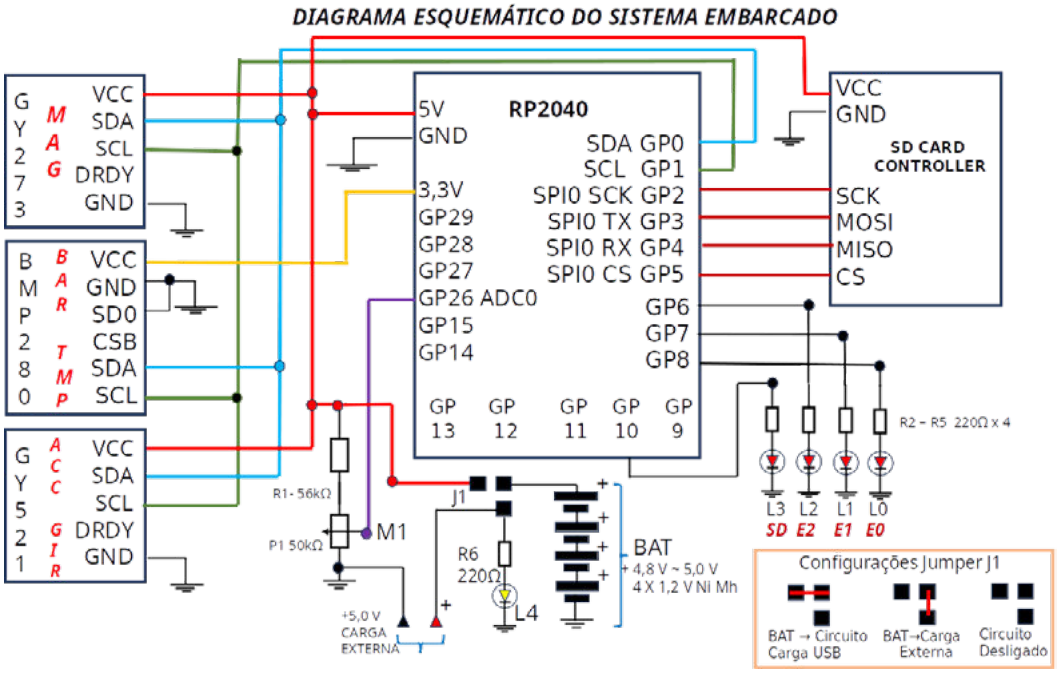


Figura 3 - Diagrama Esquemático do Sistema Embarcado

5 - SOFTWARE

O código fonte foi desenvolvido em micropython. A IDE utilizada é o Thonny (https://thonny.org/).

```
1   from machine import Pin, I2C, SPI, ADC
2   import utime
3   import time
4   import ustruct
5   import sdcard
6   import os
7
8   # ================================================
==========================
9   # Configuração dos LEDs
10  # LEDs de erro – exibem o código de inicialização nos pinos GP6, GP7
e GP8:
11  # 000 = sem erros
12  # 001 = erro no magnetômetro
13  # 010 = erro no acelerômetro
14  # 011 = erro no giroscópio
15  # 100 = erro no cartão SD
16  # 101 = Erro BMP 280
17  # 110 = Erro Calibração BMP 280
18  error_led_gp6 = Pin(6, Pin.OUT)  # Bit 2
19  error_led_gp7 = Pin(7, Pin.OUT)  # Bit 1
20  error_led_gp8 = Pin(8, Pin.OUT)  # Bit 0
21
22  # LED para indicação de gravação no cartão SD (pino GP10)
23  led_sd_write = Pin(10, Pin.OUT)
24
25  # ================================================
==========================
26  # Configuração dos barramentos
27  # I2C – utiliza GP0 (SDA) e GP1 (SCL)
28  i2c = I2C(0, scl=Pin(1), sda=Pin(0), freq=400000)
29
30  # SPI – utiliza GP2 (SCK), GP3 (MOSI), GP4 (MISO) e CS em GP5
31  spi = SPI(0, sck=Pin(2), mosi=Pin(3), miso=Pin(4))
32  cs = Pin(5, Pin.OUT)
33
34  # ADC – para leitura no canal (GP26)
35  adc = ADC(Pin(26))
36
```

```
37  # ================================================
==========================
38  # Endereços dos sensores I2C
39  ADDR_MAG = 0x0D   # Magnetômetro (ex.: QMC5883L)
40  ADDR_ACC = 0x68   # MPU6050 (acelerômetro e giroscópio)
41  ADDR_BMP = 0x76   # BMP280 (sensor de pressão/temperatura)
42
43  # ================================================
==========================
44  # Variável para armazenar o código de erro (inicialmente 000 = sem
erro)
45  error_code = 0  # 0b000
46
47  def set_error(new_code):
48      global error_code
49      # Se ainda não houver erro, registra o novo código
50      if error_code == 0:
51          error_code = new_code
52
53  def display_error_code(code):
54      # Mapeamento: bit2 -> GP6, bit1 -> GP7, bit0 -> GP8
55      error_led_gp6.value((code >> 2) & 1)
56      error_led_gp7.value((code >> 1) & 1)
57      error_led_gp8.value(code & 1)
58
59  # ================================================
==========================
60  # Inicialização – Verificação e configuração dos dispositivos
61
62  # Escanear dispositivos I2C
63  devices = i2c.scan()
64  if devices:
65      print("Dispositivos I2C encontrados:", [hex(d) for d in devices])
66  else:
67      print("Nenhum dispositivo I2C encontrado! Verifique as conexões.")
68
69  # ----- Magnetômetro -----
70  try:
71      who_am_i = i2c.readfrom_mem(ADDR_MAG, 0x0D, 1)
72      print("Magnetômetro identificado:", who_am_i)
73  except OSError as e:
74      print("Erro ao comunicar com o magnetômetro:", e)
75      set_error(0b001)
```

```
76
77      try:
78          i2c.writeto_mem(ADDR_MAG, 0x09, b'\x1D')  # 200Hz, full scale 8G, modo contínuo
79          i2c.writeto_mem(ADDR_MAG, 0x0A, b'\x00')  # Desativa interrupção
80          i2c.writeto_mem(ADDR_MAG, 0x0B, b'\x01')  # Restaura oversampling padrão
81          print("Magnetômetro configurado.")
82      except OSError as e:
83          print("Erro ao inicializar o magnetômetro:", e)
84          set_error(0b001)
85
86      # ----- Acelerômetro e Giroscópio (MPU6050) -----
87      try:
88          i2c.writeto_mem(ADDR_ACC, 0x6B, b'\x00') # Wake up do MPU6050
89      except OSError as e:
90          print("Erro ao inicializar MPU6050 (power):", e)
91          set_error(0b010)
92
93      try:
94          i2c.writeto_mem(ADDR_ACC, 0x1B, b'\x00')  # Configura giroscópio (escala ±250°/s)
95      except OSError as e:
96          print("Erro ao inicializar giroscópio:", e)
97          set_error(0b011)
98
99      try:
100         i2c.writeto_mem(ADDR_ACC, 0x1C, b'\x00')  # Configura acelerômetro (escala ±2g)
101     except OSError as e:
102         print("Erro ao inicializar acelerômetro:", e)
103         set_error(0b010)
104
105     if error_code == 0:
106         print("Acelerômetro e Giroscópio habilitados.")
107
108     # ----- BMP280 -----
109     try:
110         chip_id = i2c.readfrom_mem(ADDR_BMP, 0xD0, 1)
111         if chip_id[0] != 0x58:  # Valor esperado para BMP280
112             raise OSError("Chip ID incorreto")
113         # Configuração básica do BMP280:
114         # 0xF4: controle (oversampling e modo); 0xF5: configuração (tempo
```

```
de standby, filtro, etc.)
    i2c.writeto_mem(ADDR_BMP, 0xF4, b'\x2F')  # oversampling x1, modo normal
    i2c.writeto_mem(ADDR_BMP, 0xF5, b'\x0C')  # standby 125ms, sem filtro
    if error_code == 0:
        print("Medidor de Pressão e Temperatura Configurados.")

except OSError as e:
    print("Erro ao inicializar o BMP280:", e)
    set_error(0b101)



# Função para ler os coeficientes de calibração do BMP280
def bmp280_read_calibration():
    try:
        data = i2c.readfrom_mem(ADDR_BMP, 0x88, 24)
        calib = ustruct.unpack('<HhhHhhhhhhhh', data)
        return {
            'dig_T1': calib[0],
            'dig_T2': calib[1],
            'dig_T3': calib[2],
            'dig_P1': calib[3],
            'dig_P2': calib[4],
            'dig_P3': calib[5],
            'dig_P4': calib[6],
            'dig_P5': calib[7],
            'dig_P6': calib[8],
            'dig_P7': calib[9],
            'dig_P8': calib[10],
            'dig_P9': calib[11],
        }
    except Exception as e:
        print("Erro ao ler calibração do BMP280:", e)
        set_error(0b110)
        return None

bmp280_cal = bmp280_read_calibration()

# ----- ADC -----
try:
```

```
154         adc_value = adc.read_u16()  # Leitura de teste
155         Teste_V = (adc_value / 65535.0) * 8.9
156         print("ADC inicializado, leitura teste (V):", Teste_V)
157     except Exception as e:
158         print("Erro ao inicializar ADC:", e)
159
160     # ----- Cartão SD -----
161     def check_sd_card():
162         try:
163             sd = sdcard.SDCard(spi, cs)  # Inicializa o cartão SD
164             os.mount(sd, "/sd")         # Tenta montar o cartão SD
165             print("Cartão SD montado com sucesso.")
166             return True
167         except OSError as e:
168             print("Erro ao montar o cartão SD:", e)
169             return False
170
171     if not check_sd_card():
172         set_error(0b100)
173
174     # Apresenta o código de inicialização por 1 segundo e para a execução
se houver erro
175     display_error_code(error_code)
176     utime.sleep(1)
177     if error_code != 0:
178         # Se houver erro, mantém os LEDs indicando o problema e para o
programa
179         while True:
180             utime.sleep(1)
181
182     # ================================================
=======================
183     # Criação do arquivo CSV no cartão SD
184     filename = "/sd/dados.txt"
185     with open(filename, "w") as f:
186         # Incluímos também as leituras do BMP280 e ADC
187         f.write("MagX (G),MagY (G),MagZ (G),AccX (m/s²),AccY (m/s²),AccZ
(m/s²),GyrX (°/s),GyrY (°/s),GyrZ (°/s),BMP_T (°C),BMP_P (hPa),BAT_V\n")
188
189     # ================================================
=======================
190     # Funções de leitura dos sensores com conversões
191
```

```
192 def read_magnetometer():
193     data = i2c.readfrom_mem(ADDR_MAG, 0x00, 6)
194     x, y, z = ustruct.unpack('>hhh', data)
195     # Conversão para Gauss (supondo sensibilidade de 3000 LSB/G)
196     x = x / 3000.0
197     y = y / 3000.0
198     z = z / 3000.0
199     return x, y, z
200
201 def read_accelerometer():
202     data = i2c.readfrom_mem(ADDR_ACC, 0x3B, 6)
203     x, y, z = ustruct.unpack('>hhh', data)
204     # Conversão para m/s² (1g = 9.80665 m/s², 16384 LSB/g)
205     x = (x / 16384.0) * 9.80665
206     y = (y / 16384.0) * 9.80665
207     z = (z / 16384.0) * 9.80665
208     return x, y, z
209
210 def read_gyroscope():
211     data = i2c.readfrom_mem(ADDR_ACC, 0x43, 6)
212     x, y, z = ustruct.unpack('>hhh', data)
213     # Conversão para graus/s (supondo 131 LSB/(°/s))
214     x = x / 131.0
215     y = y / 131.0
216     z = z / 131.0
217     return x, y, z
218
219 def read_bmp280():
220     try:
221         data = i2c.readfrom_mem(ADDR_BMP, 0xF7, 6)
222         adc_P = (data[0] << 12) | (data[1] << 4) | (data[2] >> 4)
223         adc_T = (data[3] << 12) | (data[4] << 4) | (data[5] >> 4)
224
225         # Compensação de temperatura
226             var1 = (adc_T / 16384.0 - bmp280_cal['dig_T1'] / 1024.0) *
bmp280_cal['dig_T2']
227         var2 = ((adc_T / 131072.0 - bmp280_cal['dig_T1'] / 8192.0) * (ad-
c_T / 131072.0 - bmp280_cal['dig_T1'] / 8192.0)) * bmp280_cal['dig_T3']
228         t_fine = var1 + var2
229         temperature = t_fine / 5120.0  # em °C
230
231         # Compensação de pressão
```

```
232         var1 = t_fine / 2.0 - 64000.0
233         var2 = var1 * var1 * bmp280_cal['dig_P6'] / 32768.0
234         var2 = var2 + var1 * bmp280_cal['dig_P5'] * 2.0
235         var2 = var2 / 4.0 + bmp280_cal['dig_P4'] * 65536.0
236         var1 = (bmp280_cal['dig_P3'] * var1 * var1 / 524288.0 + bmp280_
cal['dig_P2'] * var1) / 524288.0
237         var1 = (1.0 + var1 / 32768.0) * bmp280_cal['dig_P1']
238         if var1 == 0:
239             pressure = 0
240         else:
241             pressure = 1048576.0 - adc_P
242             pressure = ((pressure - var2 / 4096.0) * 6250.0) / var1
243          var1 = bmp280_cal['dig_P9'] * pressure * pressure / 2147483648.0
244             var2 = pressure * bmp280_cal['dig_P8'] / 32768.0
245               pressure = pressure + (var1 + var2 + bmp280_cal['dig_P7']) /
16.0
246         # Converter pressão para hPa (de Pa)
247         pressure = pressure / 100.0
248         return temperature, pressure
249     except Exception as e:
250         print("Erro ao ler BMP280:", e)
251         return None, None
252
253 def read_adc_voltage():
254     raw = adc.read_u16()
255     voltage = (raw / 65535.0) * 8.9
256     return voltage
257
258 # ================================================
=======================
259 # Loop principal de leitura e gravação
260
261 while True:
262
263     try:
264         mag_x, mag_y, mag_z = read_magnetometer()
265     except Exception as e:
266         print("Erro ao ler magnetômetro:", e)
267         mag_x = mag_y = mag_z = 0
268
269     try:
270         acc_x, acc_y, acc_z = read_accelerometer()
271     except Exception as e:
```

```
272             print("Erro ao ler acelerômetro:", e)
273             acc_x = acc_y = acc_z = 0
274
275         try:
276             gyr_x, gyr_y, gyr_z = read_gyroscope()
277         except Exception as e:
278             print("Erro ao ler giroscópio:", e)
279             gyr_x = gyr_y = gyr_z = 0
280
281         bmp_temp, bmp_pres = read_bmp280()
282         adc_voltage = read_adc_voltage()
283
284         print(f"Mag: {mag_x:.2f} G, {mag_y:.2f} G, {mag_z:.2f} G | "
285               f"Acc: {acc_x:.2f} m/s², {acc_y:.2f} m/s², {acc_z:.2f} m/s² | "
286               f"Gyr: {gyr_x:.2f} °/s, {gyr_y:.2f} °/s, {gyr_z:.2f} °/s | "
287                   f"BMP: T={bmp_temp:.2f} °C, P={bmp_pres:.2f} hPa | ADC:
{adc_voltage:.2f}V")
288
289         led_sd_write.value(1)
290         with open(filename, "a") as f:
291             f.write(f"{mag_x:.2f},{mag_y:.2f},{mag_z:.2f},"
292                     f"{acc_x:.2f},{acc_y:.2f},{acc_z:.2f},"
293                     f"{gyr_x:.2f},{gyr_y:.2f},{gyr_z:.2f},"
294                     f"{bmp_temp:.2f},{bmp_pres:.2f},{adc_voltage:.2f}\n")
295         led_sd_write.value(0)
296
297           # Se a tensão cair abaixo de 3.4V, os LEDs de erro piscam a cada
0,5 s
298         adc_voltage = read_adc_voltage()
299         if adc_voltage < 4.0:
300             error_led_gp6.value(1)
301             error_led_gp7.value(1)
302             error_led_gp8.value(1)
303             utime.sleep(0.5)
304             error_led_gp6.value(0)
305             error_led_gp7.value(0)
306             error_led_gp8.value(0)
307             utime.sleep(0.5)
308         else:
309             utime.sleep(1)
```

Observações:

1. Configurar a IDE Thonny para RP 2040
2. Utilizar o arquivo RPI_PICO-20241129-v1.24.1.uf2 para o boot
3. Utilizar o arquivo ADAFRUIT_QTPY_RP2040-20241129-v1.24.1.uf2 para o boot
4. Utilizar MicroPython driver for SD cards using SPI bus para SD-Card
5. Observar a execução no terminal da IDE para as mensagens

Ao executar o programa via IDE Thonny, se tudo estiver ok, o terminal deverá exibir as seguintes mensagens:

MPY: soft reboot

Dispositivos I2C encontrados: [‘0xd’, ‘0x68’, ‘0x76’]

Magnetômetro identificado: b’\xff’

Magnetômetro configurado.

Acelerômetro e Giroscópio habilitados.

Medidor de Pressão e Temperatura Configurados.

ADC inicializado, leitura teste (V): 5.383458

Cartão SD montado com sucesso.

Mag: 8.27 G, -6.74 G, 10.66 G | Acc: 0.19 m/s², -0.45 m/s², 10.49 m/s² | Gyr: -1.48 °/s, 0.86 °/s, 0.72 °/s | BMP: T=30.43 °C, P=935.45 hPa | ADC: 5.36V

6 - MONTAGEM

A montagem do sistema segue os seguintes passos:

Lista de Material

01 – Raspberry Pi Pico (RP2040)

01 – GY-273 Magnetômetro QMC5883L

01 – GY 521 Acelerômetro/ Giroscópio

01 – BMP 280 – HW 611 – Barômetro / Termômetro

01 – Módulo de Leitura/Escrita cartão SD

05 – Resistores Mini de 220 Ω

01 – Resistor Mini de 56k Ω

01 – Trimpot mini 50k Ω

04 – Led cristal Vermelho 3mm

01 – Led cristal Amarelo 3mm

01 – Conjunto com 4 baterias de 1,2V Ni Mh

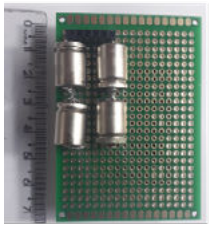
10 – Pinos terminais para jumper e bateria carga externa

04 – Barras de terminais fêmea para encaixe dos módulos

01 – Placa universal de 5 x7 cm dupla face com ilhas metalizadas

6.1 - MONTAGEM DO BANCO DE BATERIAS E CHASSIS PARA O RP 2040 E MÓDULOS

Foram utilizadas 4 células de 1,2V compondo uma unidade de 4,8V. A montagem foi realizada para aproveitar o espaço vazio do controlador de cartão SD. Foram distribuídos os conectores, os leds e o conjunto de carga e monitoração a fim de aproveitar o espaço disponível. O circuito pode ser otimizado com montagem smd podendo assim dispensar conectores e rearranjar o espaço para dimensões menores. Caso o sistema seja integrado a outros instrumentos a fonte de energia pode dispensar as baterias usadas neste caso, porém perdendo a autonomia de energia. A figura 4 (foto) mostra a montagem sugerida e na figura 5 os detalhes dos módulos.



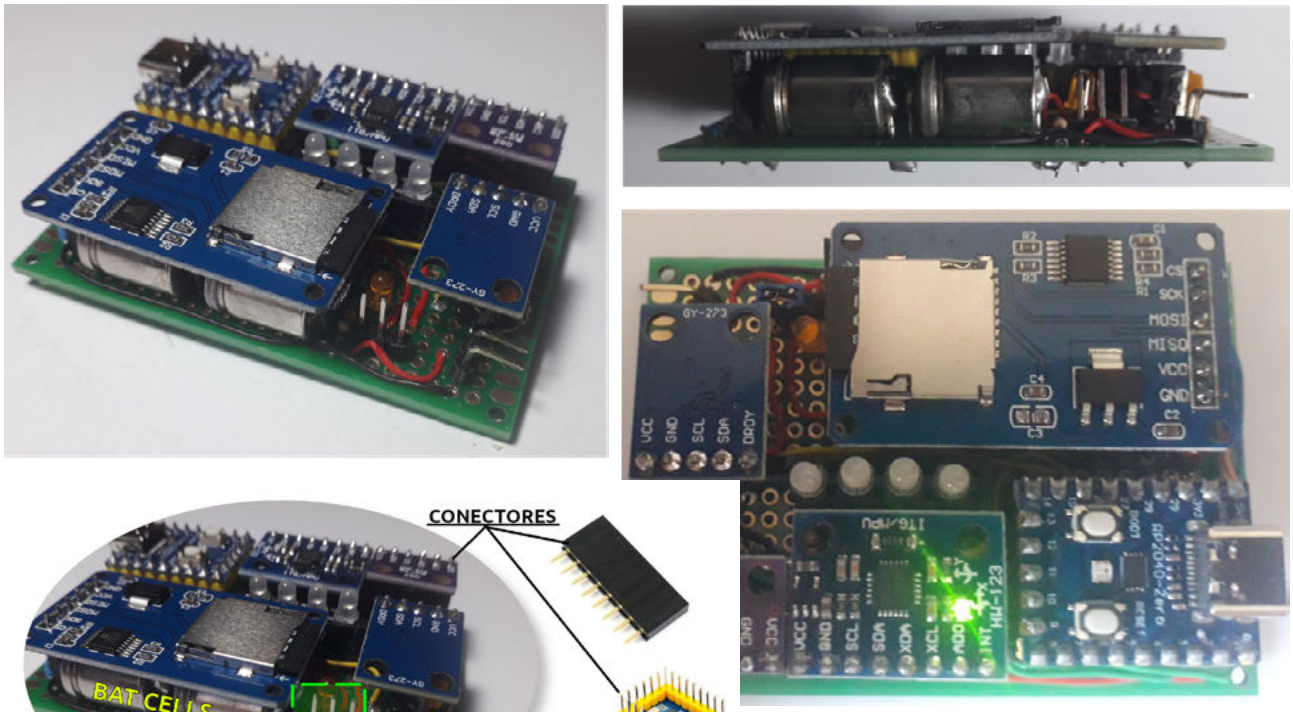


Figura 4 - Montagem sugerida

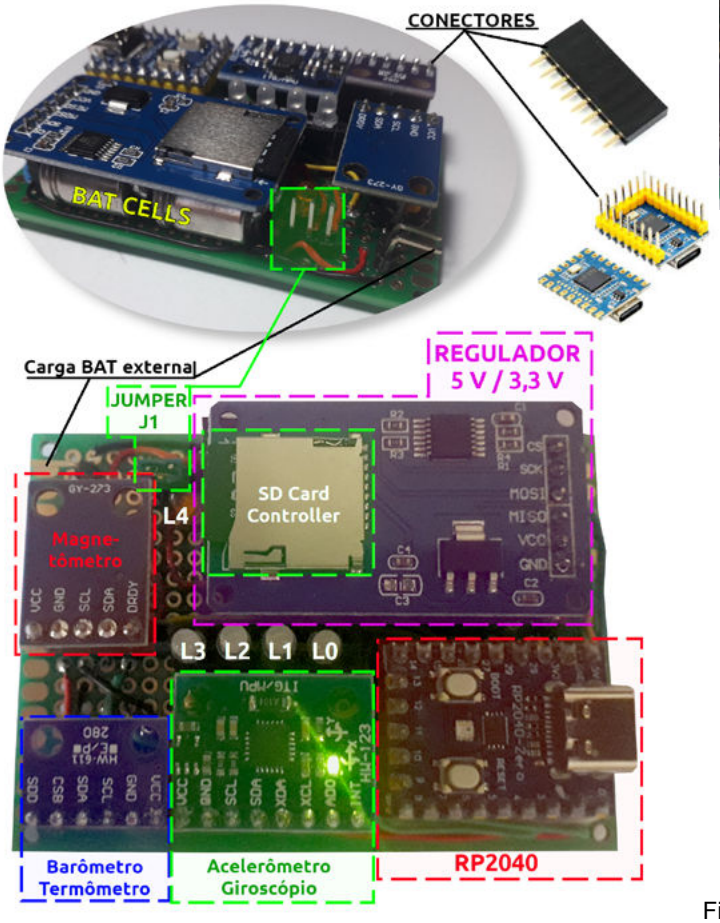


Figura 5: Detalhes dos módulos

6.2 - ENCAIXE E LIGAÇÃO DOS MÓDULOS

Os módulos devem ser conectados conforme diagrama da figura 3. Alguns cuidados e atenções importantes devem obrigatoriamente serem seguidas, para que não haja danos nos módulos e no microcontrolador:

- O sistema opera com a tensão de 3,3 V de forma comum, ou seja, tanto a alimentação quanto os sinais são de 3,3V. A opção de alimentar com 5,0 V advém do fato que os módulos utilizam regulador de tensão de 5V para 3,3V., EXCETO O BMP 280 que utiliza a tensão de 3,3V vinda do microcontrolador. Deve-se ter cuidado extremo em não inverter a alimentação, ou colocar 5V em qualquer PINO TANTO DO MICROCONTROLADOR quanto dos MÓDULOS, sob pena de dano irreversível!
- Ao montar a bateria verifique as soldagens para que não haja qualquer curto-circuito, ligações erradas ou ligação dos 5V da bateria em qualquer pino do sistema, exceto os VCC´s dos módulos, excluindo o BMP 280 que é alimentado pelo 3,3V do RP-2040. Deixe para alimentar o circuito com as baterias DEPOIS, de testar com o multímetro cada conexão e cada valor de tensão se estão nos pinos corretos!
- Cuidado para não trocar a posição dos módulos ao encaixar nos conectores. Erros nessa etapa com a alimentação ativa pode causar danos nos módulos ou no RP 2040.
    - Ligue os LEDs de erro aos pinos designados (GP6, GP7, GP8, GP10, com os resistores!

6.2.1 - INTEGRAÇÃO DO ADC

Conecte o pino de entrada analógica (ADC) à fonte de tensão da bateria, garantindo que o circuito de condicionamento converta a tensão corretamente. Ajuste M1 para aproximadamente 2,5V. O ajuste deverá ser calibrado ao rodar o software no IDE.

6.2.2 - ALIMENTAÇÃO

Certifique-se de que a bateria esteja corretamente conectada e que a tensão esteja dentro dos parâmetros especificados para o funcionamento dos dispositivos.

6.2.3 - UPLOAD DO PROGRAMA

Faça o upload do firmware (MicroPython) para o microcontrolador e teste a comunicação com os sensores, bem como o funcionamento dos LEDs e do cartão SD.

6.2.4 - VERIFICAÇÃO FINAL

Realize uma verificação do sistema, assegurando que todos os módulos respondam conforme o esperado e que o arquivo dados.txt seja criado e atualizado corretamente no cartão SD.

6.3 - CUIDADOS COM A ALIMENTAÇÃO E LIMITES DE OPERAÇÃO DOS SENSORES

- Para sensores que possuem regulador de tensão, observe a faixa de operação indicada (+5V).
- No caso do BMP280, que opera com 3,3 V, utilize um regulador específico para garantir a alimentação correta.
- Observe também os limites de temperatura de operação dos sensores, principalmente para aplicações em voos estratosféricos, onde condições extremas podem comprometer o desempenho dos componentes.

6.4 - ESPECIFICAÇÕES FINAIS DA MONTAGEM

- Peso Aproximado: 50 g
- Dimensões: 5 cm x 7 cm x 1,6 cm
- Consumo: ~ 40 mA
- Tensão Máxima de Operação: 5,0 V
- Tensão Mínima de Operação: 4,0 V
- Cartão: FAT16, 8G
- Fonte de Carga: Fonte externa de 5 V ou USB
- BATERIAS 1,2V x 4 (Ni-MH)
- Taxa de coleta de dados de todos os sensores até o cartão SD: 1 amostra por segundo
- Autonomia: 2 horas e 6 minutos

7 - TESTES E VALIDAÇÃO

Para a verificação do funcionamento adequado do sistema, foram realizados testes com monitoramento dos estados dos LEDs. Apresenta-se a tabela de mensagens:

Legenda

- P = piscante; 1 = aceso; 0 = apagado.

Códigos dos LEDs (Led4, Led3, Led2, Led1)

- P000: Operação normal
- 0000: Não está gravando no SD
- 0001: Falha no Magnetômetro
- 0010: Falha no Acelerômetro
- 0011: Falha no Giroscópio
- 0100: Falha no cartão SD
- 0101: Falha no Barômetro/Termômetro
- 0110: Falha na calibração do Barômetro/Termômetro
- XPPP: Bateria crítica

Observação: O LED laranja indica a presença de tensão externa na bateria, atuando no modo de carregador externo ou via USB (veja posição de Jumper 1).

CONSIDERAÇÕES FINAIS

Para baixar os códigos e arquivos mencionados neste artigo, acesse https://www.newtoncbraga.com.br/arquivos/incb026.zip

A instrumentação embarcada pode ser usada em balões de pesquisa estratosférica ou em foguetes de pesquisa. Tópicos como diferenças grandes de temperatura ou de radiação ou vibração não foram considerados neste projeto, devendo o interessado prover a mitigação de tais efeitos. A autonomia da fonte de energia pode ser estendida por baterias mais eficientes e gerenciamento das cargas através dos modos de economia ou estratégia de controle de consumo do sistema, a serem implementados tanto no hardware quanto no software.

Assista ao vídeo em que o autor explica a montagem e funcionamento do aparelho.

Referências

RASPBERRY PI LTD. Raspberry Pi Pico Datasheet. 2020. Disponível em: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf. Acesso em: 28 fev. 2025. p.10.

BOSCH SENSORTEC. BMP280 Datasheet. 2018. Disponível em: https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-10.pdf. Acesso em: 28 fev. 2025. p.7.

INVENSENSE. MPU-6000 and MPU-6050 Product Specification Revision 3.4. 2013. Disponível em: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf. Acesso em: 28 fev. 2025. p.12.

QMC5883L Datasheet. 2025. Disponível em: https://datasheet.chip-small.com/QMC5883L_QST/960367.pdf Acesso em: 28 fev. 2025. p.5.

SD CARD ASSOCIATION. Physical Layer Simplified Specification. 2009. Disponível em: http://www.sdcard.org/downloads/pls/. Acesso em: 28 fev. 2025.

NOAA. National Oceanic and Atmospheric Administration – Weather and Climate Data. 2021. Disponível em: https://www.noaa.gov. Acesso em: 28 fev. 2025. p.22.

NASA. Guidelines for Aerospace Instrumentation. 2020. Disponível em: https://www.nasa.gov. Acesso em: 28 fev. 2025. p.18.