EnvloggerSD 自律型環境データロガーシステム 仕様書

作成日: 2026年1月11日 **バージョン:** 1.1 (Final Release)

---

1. システム概要

本システムは、Raspberry Pi Picoを用いたスタンドアロン型の環境データロガーである。 外部電源に依存せず、乾電池のみで長期間稼働し、1時間ごとに「温度」「湿度」「気圧」を測定。正確な時刻情報と共にmicroSDカードへCSV形式で記録する。 省電力化のために、測定時以外はDeep Sleep（お休みモード）状態で待機する仕様である。

---

2. 機能要件

1. **環境測定機能:**
    - 温度（°C）、湿度（%）、気圧（hPa）を同時に測定できること。
    - *※重要事項:*湿度測定が可能な「BME280」を使用すること。（外見が酷似したBMP280は不可）
2. **正確な時刻管理機能:**
    - Picoの電源が落ちても時刻を保持できる外部RTC（Real Time Clock）モジュールを使用すること。
3. **データ記録機能:**
    - 測定データをmicroSDカードに保存すること。
    - 保存形式はCSV（カンマ区切り）。ファイルは月ごとに自動分割（例: `/sd/2026-01.csv`）。
    - 新規ファイル作成時に、自動でヘッダー行（Date,Temperature,Humidity,Pressure）を挿入すること。
4. **省電力動作機能:**
    - 1回の測定・記録が完了した後、自動的に1時間のDeep Sleepモードへ移行すること。
5. **状態通知機能:**
    - PicoオンボードLEDを用いて、動作状態（起動、成功、エラー）を目視確認できること。

---

3. ハードウェア構成仕様

3.1. 電源仕様 (厳守)

本システムは Raspberry Pi Pico のオンボードレギュレータを使用し、**VSYS** ピンから給電する。

- **推奨電源: DC 4.5V** （単3形アルカリ乾電池 × **3本** 直列）
- **許容電圧範囲: DC 1.8V ～ 5.5V**
    - *※警告:*単3アルカリ電池4本直列（6.0V）は、Picoの絶対最大定格（5.5V）を超えるため**使用不可**とする。
    - *※備考:*ニッケル水素充電池（エネループ等 1.2V）を使用する場合は、3本（3.6V）または4本（4.8V）での動作が可能。

3.2. 部品表 (BOM)

品目	仕様・備考	数量
マイコンボード	**Raspberry Pi Pico** (RP2040)	1

品目	仕様・備考	数量
環境センサ	**BME280** モジュール (I2C接続) ※必ずChip ID`0x60`(湿度対応版)を確認すること。	1
RTCモジュール	**DS3231** (I2C接続・電池バックアップ付き)	1
SDカードモジュール	SPI接続タイプ	1
microSDカード	32GB以下 (FAT32フォーマット済み)	1
電源ケース	**単3電池 × 3本用** 電池ボックス（スイッチ付き推奨）	1
基板	ブレッドボード互換ユニバーサル基板 (推奨)	1
接続部品	ピンヘッダ(オス)、ピンソケット(メス)、配線材、はんだ	一式

3.3. 配線図（ピンアサイン）

【重要】電源接続規則

- **電池入力(+)**: Picoの **39番ピン (VSYS)** に接続。（36番 3V3には接続しないこと）
- **モジュール給電**: Picoの **36番ピン (3V3 OUT)** から各センサへ供給。

機能	Pico ピン番号 (GPIO)	接続先モジュールとピン
I2C Bus	**1** (GP0)	BME280 **SDA**, DS3231 **SDA**
	2 (GP1)	BME280 **SCL**, DS3231 **SCL**
SPI Bus	**14** (GP10)	SDモジュール **SCK** (またはCLK)
(SDカード用)	**15** (GP11)	SDモジュール **MOSI** (またはDI)
	16 (GP12)	SDモジュール **MISO** (またはDO)
	17 (GP13)	SDモジュール **CS**
電源(+)	**36** (3V3 OUT)	全モジュールの **VCC** (またはVIN)
GND	**38** (GND) など	全モジュールおよび電池の **GND** (-)
電池入力(+)	**39** (VSYS)	電池ボックスの **赤線** (+)

4. ソフトウェア仕様

4.1. 開発環境

- **OSファームウェア:** MicroPython (Raspberry Pi Pico用最新版 `.uf2`)

4.2. 必須ライブラリ詳細

本システムの動作には、以下の外部ライブラリ（ドライバ）が必要である。 Picoのストレージ直下に、ファイル名を変更せずに配置すること。

ファイル名	概要・出典元（Author/Source）	入手先URL（参考）
`sdcard.py`	**MicroPython 公式ドライバ** MicroPythonチーム提供の標準ドライバ。	MicroPython GitHub
`bme280.py`	**Paul Cunnane / Peter Dahlebrg 版** MicroPythonコミュニティで標準的に使われるドライバ。 ※`read_compensated_data()`メソッドを持つもの。	robert-hh GitHub ※リンク先の`bme280_int.py` を `bme280.py` にリネームして使用。
`ds3231.py`	**Peter Hinch 版 (または互換版)** I2C接続RTCドライバ。 ※`get_time()`, `save_time()`メソッドを持つもの。	micropython-samples GitHub

4.3. 初期設定コード (RTC時刻合わせ)

RTCモジュールの時刻を合わせるためのスクリプト。初回のみ実行する。

```
# set_clock.py (初期設定用)
import machine
from machine import I2C, Pin
import ds3231
import time

# I2C設定
i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)
ds = ds3231.DS3231(i2c)

# ここに設定したい現在時刻を入力 (年, 月, 日, 曜日, 時, 分, 秒, サブ秒)
# 曜日: 0=月, 1=火... 6=日
# 例: 2026年1月11日 10時00分00秒
CURRENT_TIME = (2026, 1, 11, 6, 10, 0, 0, 0)

print("Setting Pico RTC...")
rtc = machine.RTC()
rtc.datetime(CURRENT_TIME)

print("Syncing to DS3231...")
# Picoの内蔵時計をDS3231に保存させる
ds.save_time()
time.sleep(0.5)

print("Done. Current DS3231 time:", ds.get_time())
```

4.4. メインプログラム (`main.py`)

本番運用用のコード。Deep Sleep機能を含む。

```
import machine
import time
import sdcard
import uos
from machine import I2C, SPI, Pin

# ドライバ読み込み（出典元のAPIに準拠）
import bme280
import ds3231

# --- ピン設定 ---
i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)
spi = SPI(1, baudrate=1000000, polarity=0, phase=0, sck=Pin(10),
mosi=Pin(11), miso=Pin(12))
cs_pin = Pin(13, Pin.OUT)
led = Pin("LED", Pin.OUT)

# --- 関数群 ---
def blink(times, interval=0.1):
    for _ in range(times):
        led.on()
        time.sleep(interval)
        led.off()
        time.sleep(interval)

def get_time_tuple():
    try:
        ds = ds3231.DS3231(i2c)
        return ds.get_time()
    except Exception:
        return None

def get_env():
    try:
        bme = bme280.BME280(i2c=i2c)
        raw_t, raw_p, raw_h = bme.read_compensated_data()
        temp = raw_t / 100
        pres = raw_p / 25600
        hum = raw_h / 1024
        return temp, hum, pres
    except Exception:
        return None, None, None

def write_log(filename, line):
    try:
        sd = sdcard.SDCard(spi, cs_pin)
        vfs = uos.VfsFat(sd)
        uos.mount(vfs, "/sd")
        full_path = "/sd/" + filename
        file_exists = False
        try:
            uos.stat(full_path)
```

```
            file_exists = True
        except OSError:
            file_exists = False

        with open(full_path, "a") as f:
            if not file_exists:
                f.write("Date,Temperature,Humidity,Pressure\n")
            f.write(line + "\n")
        uos.umount("/sd")
        return True
    except Exception as e:
        try: uos.umount("/sd")
        except: pass
        return False

# --- メイン処理 ---
blink(1)
time.sleep(1)

try:
    t = get_time_tuple()
    temp, hum, pres = get_env()

    if t is not None and temp is not None:
        now_str = "{:04d}-{:02d}-{:02d} {:02d}:{:02d}:{:02d}".format(t[0],
t[1], t[2], t[3], t[4], t[5])
        filename = "{:04d}-{:02d}.csv".format(t[0], t[1])
        log_str = f"{now_str},{temp:.2f},{hum:.2f},{pres:.2f}"

        if write_log(filename, log_str):
            blink(3)
        else:
            blink(10, 0.05)
    else:
        blink(2, 0.5)

except Exception:
    blink(10)

time.sleep(0.1)
machine.deepsleep(3600 * 1000) # 1時間スリープ
```

5. 実装・運用上の注意

1. **ソケット化の推奨:** 基板実装時は、メンテナンス性を考慮し、全モジュールをピンソケット経由で接続すること。
2. **設置環境:** 防水筐体に入れ、直射日光の当たらない通気性の良い場所に設置すること。
3. **電池交換:** 電池電圧が低下すると、SDカードへの書き込みエラーが発生しやすくなる。LEDが高速点滅し始めた場合は電池交換のサインである。