

[Spresense メインボードによるタイムラプスカメラ DeepSleep 版]

マイクロ SD カードの接続までが終了したら、Spresense メインボードにカメラボードをフレキシブルケーブルで接続します。タイムラプスカメラのスケッチをリスト 1 に示します。なお、このスケッチは GPLv3 に基づき、再配布および改変が可能です。

リスト 1 の動作の流れを説明します。本スケッチでは、まず `setup()` 内でマイクロ SD カードの初期化、撮影枚数カウンタファイルからの撮影回数の復元、カメラの初期化を行います。次に `loop()` では、静止画を 1 枚撮影し、マイクロ SD カードに保存します。その後、撮影枚数カウンタファイルの 1 行目に現在の撮影回数を記録し、マイクロ SD カードおよびカメラの終了処理を行ったのち、Deep Sleep 状態に移行します。

Deep Sleep 状態では Spresense のほとんどの処理が停止し、指定した時間が経過すると RTC によって復帰します。復帰時には CPU が再起動され、プログラムは再び `setup()` から実行されます。そのため、`setup()` と `loop()` の処理を 1 回ずつ繰り返すことで、一定時間間隔での撮影を実現しています。Deep Sleep からの復帰時には RAM 上の変数の内容は保持されないため、現在が何枚目の撮影であるかという情報も失われてしまいます。そこで、本スケッチでは毎回撮影後に撮影枚数カウンタをファイルとしてマイクロ SD カードに保存しています。これにより、復帰後も撮影回数を正しく把握でき、連番のファイル名を継続して使用することが可能になります。

```
/*
 * Copyright (C) 2025 Hiroki Sato
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 3.
 *
 * This program uses Arduino SD library
 * Copyright (C) Yoshino Taro
 */
```

```
#include <Arduino.h>
#include <Camera.h>
#include <SPI.h>
#include <SPI1SD.h>
#include <LowPower.h>
#include <RTC.h>
```

```
/* =====
   ユーザー設定
   ===== */
```

```
/*
 撮影間隔 (ミリ秒)
  - 10 秒おき   : 10000
  - 1 分おき   : 60000
  - 10 分おき  : 600000
*/
```

```
const uint32_t INTERVAL_SEC = 10;
```

```
/*
  マイクロ SD カード電源制御を行うかどうか
  true  : D25 で PMOS ゲート制御
  false : SD カード常時通電
*/
```

撮影間隔を設定

マイクロ SD カードの
電源制御を設定(今回は未使用)

```
const bool USE_SD_POWER_CONTROL = false;
```

```
/*  
  PMOS ゲート制御ピン  
  H : SD カード電源 OFF  
  L : SD カード電源 ON  
*/
```

```
const int SD_POWER_PIN = 25;
```

```
/* ===== */
```

```
/* SPI5 を使った SD クラス */  
SpiSDClass SD(SPI5);
```

```
/* 撮影枚数カウンタ  
  DeepSleep 復帰後も継続するため SD に保存する */  
static uint32_t gCounter = 0;  
static const char COUNTER_FILE[] = "count.txt";
```

```
/* =====  
  カウンタ読み込み/保存関数の設定  
  ===== */
```

```
void loadCounter() {  
  // カウンタファイルが無ければ 0 スタート  
  if (!SD.exists(COUNTER_FILE)) {  
    gCounter = 0;  
    return;  
  }  
  
  SpiFile f = SD.open(COUNTER_FILE, FILE_READ);  
  if (!f) {  
    gCounter = 0;  
    return;  
  }  
  
  // 1 行目に書かれた数値を読む  
  gCounter = f.readStringUntil('\n').toInt();  
  f.close();  
}
```

```
void saveCounter() {  
  // 毎回書き直す (簡単で確実)  
  if (SD.exists(COUNTER_FILE)) {  
    SD.remove(COUNTER_FILE);  
  }  
  
  SpiFile f = SD.open(COUNTER_FILE, FILE_WRITE);  
  if (!f) return;  
  
  f.println(gCounter);  
  f.close();  
}
```

```
/* =====  
  setup()  
  ===== */
```

```
void setup() {  
  Serial.begin(115200);  
  while (!Serial);  
  
  Serial.println("=== Spresense Timelapse DeepSleep ===");  
  
  /* ---- LowPower / RTC 初期化 ---- */  
  LowPower.begin();  
  RTC.begin();  
  
  /* ---- 起動方法の判定 ----  
    DEEP RTC : DeepSleep からの復帰  
    それ以外 : 電源 ON / リセット */  
  bootcause_e bc = LowPower.bootCause();  
  if (bc == DEEP_RTC) {  
    Serial.println("Wakeup from DeepSleep");  
  } else {  
    Serial.println("Power On / Reset");  
  }  
  
  /* ---- SD 電源制御ピン設定 ---- */  
  if (USE_SD_POWER_CONTROL) {  
    pinMode(SD_POWER_PIN, OUTPUT);  
    digitalWrite(SD_POWER_PIN, HIGH); // SD 電源 OFF  
  }  
}
```

マイクロ SD カードの
電源制御を設定(今回は未使用)(続き)

マイクロ SD カードの
電源制御を行うピンを指定
(エラー! 参照元が見つかりま
せぬ)

撮影枚数カウンタの設定

撮影枚数カウンタファイルの読み込み関数

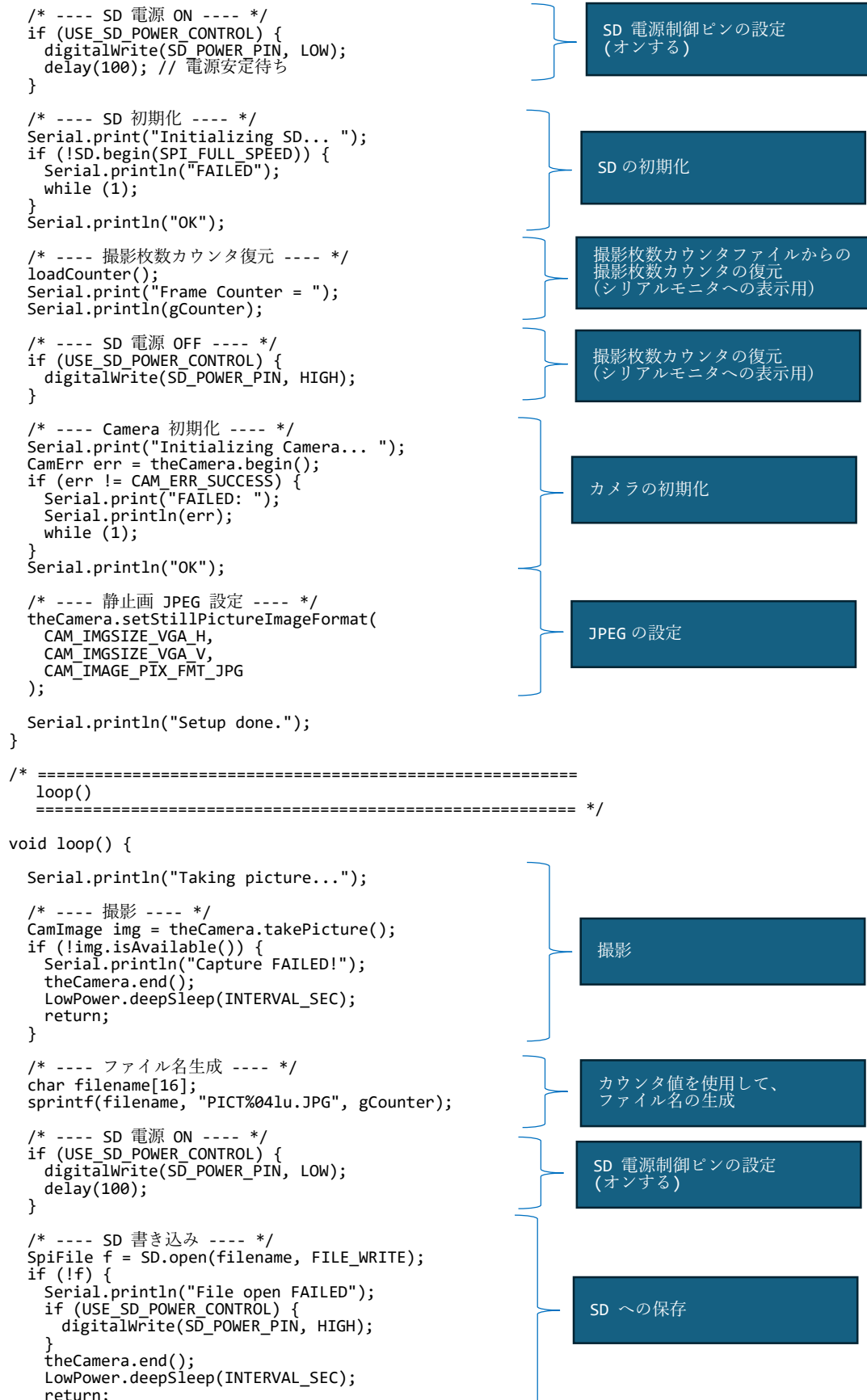
撮影枚数カウンタファイルの保存関数

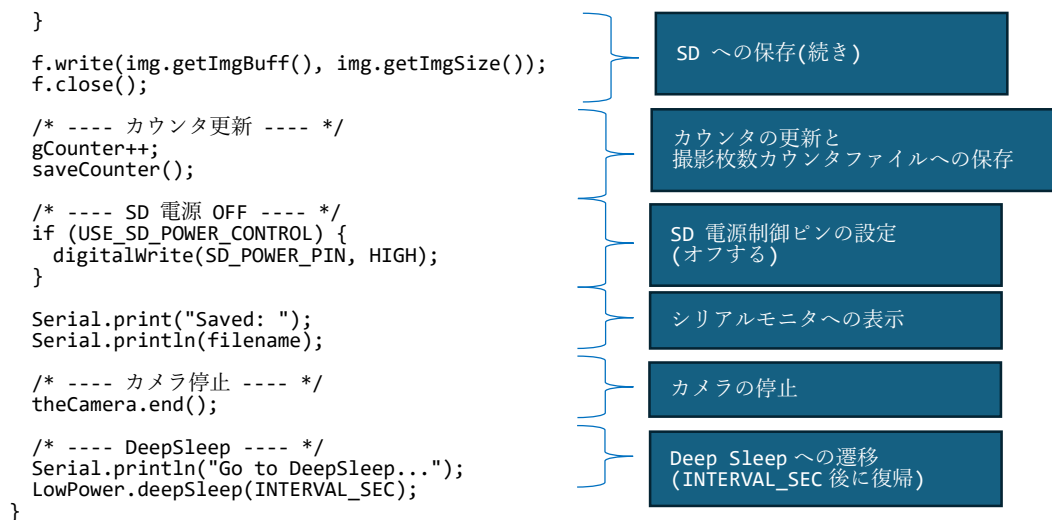
シリアルモニタの設定

RTC の初期化

起動方法の判定
(シリアルモニタへの表示用)

SD 電源制御ピンの設定
(いったんオフにする)





リストの最初の方にある変数 INTERVAL_SEC で撮影間隔を定義します。デモ用に 10 秒おき、1 分おきの設定例もコメント文に記載してあります。今回の様な植物の観測では 10 分おきで十分です。マイクロ SD カードの電源制御を行うかどうかは変数 USE_SD_POWER_CONTROL で定義します。現状は使用しないため false にしてあります。この電源制御は**エラー! 参照元が見つかりません**。の回路図では D25 で行っています。この電源制御をした場合、D25 により PMOS トランジスタが動作します。スケッチ内でマイクロ SD カードの電源をオン、オフしています。この電源制御をしない場合は、常時マイクロ SD カードの電源はオンになっています。本スケッチでは、電源制御の有無にかかわらず、setup() 内で一度マイクロ SD カードの電源をオフしてからオンする処理を行っています。これは、電源投入時点でマイクロ SD カードの状態が不定である可能性を考慮し、初期化を確実に行うための措置です。

尚、本スケッチでは消費電流と書き込み時間を抑える目的で、撮影解像度を VGA サイズ (640×480) に設定しています。