M5Paper 環境モニターソース

美都

2021 年 4 月 13 日

目次

1 メイン 2

2 バッテリーメーター 6
- 2.1 battery.h . . . 6
- 2.2 battery.cpp . . . 6

3 温湿度計 8
- 3.1 thermometer.hpp . . . 8
- 3.2 thermometer.cpp . . . 8

4 ネットからの情報取得 11
- 4.1 infoFromNet.hpp . . . 11
- 4.2 wifiid.h . . . 11
- 4.3 apikey.h . . . 11
- 4.4 infoiFromNet.cpp . . . 11

5 時計の表示 14
- 5.1 tokei.hpp . . . 14
- 5.2 tokei.cpp . . . 14

6 天気予報データの管理 16
- 6.1 tenki.hpp . . . 16
- 6.2 tenki.cpp . . . 17

7 天気予報の表示 19
- 7.1 drawtenki.hpp . . . 19
- 7.2 drawtenki.cpp . . . 19

8 SD カード内の jpeg ファイルをスキャン 22
- 8.1 scanfile.hpp . . . 22
- 8.2 scanfile.cpp . . . 22

9 メッセージボックス 24
- 9.1 messageArea.h . . . 24
- 9.2 messageArea.cpp . . . 24

10 ユーティリティー 26
- 10.1 util.h . . . 26
- 10.2 util.cpp . . . 26

1 メイン

```
#include <M5EPD.h>
#define LGFX_M5PAPER
#include <LovyanGFX.hpp>

#include "battery.h"
#include "drawtenki.hpp"
#include "infoFromNet.hpp"
#include "messageArea.hpp"
#include "scanfile.hpp"
#include "tenki.hpp"
#include "thermometer.hpp"
#include "tokei.hpp"
#include "util.h"

//#define MEMPRINT

static LGFX lcd;
MessageArea *mesg;
bool startedOnTimer;
bool isOperateMode = false;
bool mustCheckInfo = false;
time_t timeOfOperateModeStart = 0;
time_t timeOfLastUpdate = 0;

inline void printMem(const char *msg) {
#ifdef MEMPRINT
    Serial.printf("【%s】heap:%'d, psram:%'d\n", msg, ESP.getFreeHeap(),
                  ESP.getFreePsram());
#endif
}

void drawLcd() {
    drawBattery(960 - 120 - 5, 5, &lcd);

    {
        Tokei tokei(300, 100);
        tokei.drawDigitalTokei(&lcd, 630, 50);
    }

    {
        Thermometer t(200, 200);
        t.drawTempMeter(&lcd, 530, 180);
        t.drawHumMeter(&lcd, 750, 180);
    }

    {
        Tenki tenki = Tenki();
        DrawTenki drawTenki(&tenki, 455, 112);
        drawTenki.draw(&lcd, 495, 408);
    }

    //写真の表示。480*320がちょうど。
    //プログレッシブと最適化を無効にすること。
    // SDカードのルート直下のjpgファイルを対象とする。
```

```
55      JpegFiles jpgs;
56      char *filename = jpgs[random(jpgs.count())];
57      Serial.print("写真番号:");
58      Serial.println(filename);
59      lcd.drawJpgFile(SD, filename, 10, 110);
60      delay(400);
61  }
62
63  // ●分ピッタリまでの秒数
64  int rest_minute() {
65      time_t time = now();
66      int sec = (int)(time % 60);
67      return 60 - sec;
68  }
69
70  // シャットダウンを試みる。通電中はすり抜ける
71  void challengeShutdown() {
72      Serial.println("電源切ってみるテスト");
73      int rest_sec = rest_minute() - 4;
74      if (rest_sec <= 0) rest_sec = 50;
75      if (rest_sec < 30) rest_sec += 60;
76      if (rest_sec > 120) rest_sec = 90;
77      Serial.printf("電源断(%d秒)\n", rest_sec);
78      M5.shutdown(rest_sec);  // 一旦停止
79      Serial.println("電源切れませんでした。");
80  }
81
82  void checkInfoFromNetwork(bool always = false) {
83      time_t timenow = now();
84      time_t outdated = timenow - 10 * 60;
85      Tenki tenki;
86
87      if (!tenki.isEnable() || tenki.getDate(0) < outdated || timenow == 0 ||
88          mustCheckInfo) {
89          Serial.println("ネットワークの情報の取得開始");
90          GetInfoFromNetwork info;
91          info.setNtpTime();
92          updateSystemTime();
93          info.getWeatherInfo();
94          tenki.refresh();
95      }
96
97      char fmt[] = "%Y%m%d %H:%M:%S";
98      char s[50];
99      time_t t = tenki.getDate(0);
100     strftime(s, sizeof(s), fmt, gmtime(&t));
101     Serial.printf("teki-date:%s\n", s);
102     t = now();
103     strftime(s, sizeof(s), fmt, gmtime(&t));
104     Serial.printf("now-date:%s\n", s);
105
106     if (abs(tenki.getDate(0) - now()) > 3 * 3600 + 10 * 60) ESP.restart();
107
108 }
109
110 // RTCモジュールのタイマーより起動されたかを確認する。
111 // その後、RTC.begin()相当の処理を実行する。
112 void saveStartedOnTimer() {
```

```
113     Wire.begin(21, 22);
114     uint8_t rtcStatus = M5.RTC.readReg(0x01);
115     startedOnTimer = (rtcStatus & 0x0c) != 0 ? true : false;
116     M5.RTC.writeReg(0x00, 0x00);
117     M5.RTC.writeReg(0x01, 0x00);
118     M5.RTC.writeReg(0x0D, 0x00);
119 }
120
121 void setup() {
122     M5.begin(false, true, true, true, true);
123     Serial.println("システム初期化終了");
124     M5.BatteryADCBegin();
125     M5.SHT30.Begin();
126     SD.begin();
127     lcd.init();
128     lcd.setRotation(1);
129     mesg = new MessageArea(490, 50, 2, true);
130     randomSeed(analogRead(0));
131     saveStartedOnTimer();
132     checkInfoFromNetwork();
133
134     drawLcd();
135
136     if (startedOnTimer) {
137         challengeShutdown();
138     }
139     timeOfOperateModeStart = now();
140     timeOfLastUpdate = timeOfOperateModeStart;
141     isOperateMode = true;
142 }
143
144 void loop() {
145     M5.update();
146
147     if (isOperateMode) {
148         char buff[80];
149         if (strlen(mesg->getText(buff, 80, 0)) == 0) {
150             strcat(buff, "操作可能->");
151         }
152         if (strlen(buff) > 78) {
153             strcpy(buff, "操作可能->");
154         }
155         if (M5.BtnR.wasPressed()) {
156             strcat(buff, "R");
157             timeOfOperateModeStart = now();
158         }
159         if (M5.BtnL.wasPressed()) {
160             strcat(buff, "L");
161             timeOfOperateModeStart = now();
162         }
163         if (M5.BtnP.wasPressed()) {
164             strcat(buff, "P");
165             mustCheckInfo = true;
166             timeOfOperateModeStart = now();
167         }
168         mesg->setText(buff, 0)->flush()->draw(&lcd, 5, 490);
169     } else {
170         mesg->setText("", 0)->flush()->draw(&lcd, 5, 490);
```

```
171     }
172
173     isOperateMode = (now() - timeOfOperateModeStart < 50);
174
175     int elapsed = now() - timeOfLastUpdate;
176     if (rest_minute() > 55 && elapsed > 50) {
177         timeOfLastUpdate = now();
178         checkInfoFromNetwork();
179         mustCheckInfo = false;
180         drawLcd();
181         if (!isOperateMode) {
182             challengeShutdown();
183             isOperateMode = true;
184             timeOfOperateModeStart = now();
185         }
186     }
187     delay(400);
188 }
```

2 バッテリーメーター

2.1 battery.h

```
#include <M5EPD.h>
#define LGFX_M5PAPER
#include <LovyanGFX.hpp>

// バッテリー残量を(x,y)に表示する。
int drawBattery(int x, int y, LGFX *lcd);
int get_rest_battery();
```

2.2 battery.cpp

```
#include "battery.h"

// バッテリー残量の取得
int get_rest_battery() {
    const int max_vol = 4350;
    const int min_vol = 3300;
    // M5.BatteryADCBegin();
    int voltage = M5.getBatteryVoltage();
    voltage = max(voltage, min_vol);
    voltage = min(voltage, max_vol);
    float rest_battery_raw =
        (float)(voltage - min_vol) / (float)(max_vol - min_vol);
    rest_battery_raw = max(rest_battery_raw, 0.01f);
    rest_battery_raw = min(rest_battery_raw, 1.f);
    return (int)(rest_battery_raw * 100);
}

// バッテリー残量計の表示
int drawBattery(int x, int y, LGFX *lcd) {
    LGFX_Sprite battery_meter(lcd);
    int rest_battery = get_rest_battery();

    // バッテリー矩形の表示
    battery_meter.setColorDepth(4);
    battery_meter.createSprite(120, 30);
    battery_meter.fillSprite(15);
    battery_meter.setColor(0);
    battery_meter.drawRect(10, 10, 45, 20);
    battery_meter.fillRect(55, 17, 5, 5);
    battery_meter.fillRect(10, 10, (int)((45 * rest_battery) / 100), 20);

    // バッテリー残量文字の表示
    battery_meter.setFont(&fonts::lgfxJapanMinchoP_20);
    battery_meter.setTextSize(1, 1);    // 縦,横　倍率
    battery_meter.setTextColor(0, 15);  // 文字色,背景
    battery_meter.setCursor(62, 10);
    battery_meter.printf("%d%%", rest_battery);

    lcd->startWrite();
    battery_meter.pushSprite(x, y);
    lcd->endWrite();
```

```
42
43      return rest_battery;
44  }
```

3 温湿度計

3.1 thermometer.hpp

```
#include <M5EPD.h>
#define LGFX_M5PAPER
#include <LovyanGFX.hpp>

class Thermometer {
   private:
    float temp;
    float hum;
    int sizex;
    int sizey;
    float radius;
    LGFX_Sprite face;
    LGFX_Sprite scale[2];
    LGFX_Sprite hand;

    void makeMeterFace(int min, int max, const char *unit);
    void makeScale();
    void makeHand();

   public:
    Thermometer(int sizex = 200, int sizey = 200);

    float get_temp();
    float get_hum();

    void drawTempMeter(LovyanGFX *lcd, int x, int y);
    void drawHumMeter(LovyanGFX *lcd, int x, int y);
    void drawString(LovyanGFX *lcd, int x, int y);
};
```

3.2 thermometer.cpp

```
#include "thermometer.hpp"

Thermometer::Thermometer(int sizex, int sizey) : sizex(sizex), sizey(sizey) {
    M5.SHT30.Begin();
    radius = min(sizex, sizey) / 2 * 0.95;
    makeScale();
    makeHand();
};

void Thermometer::makeScale() {
    scale[0].setColorDepth(4);
    scale[0].setPsram(true);
    scale[0].createSprite(radius / 25, radius / 5);
    scale[0].fillSprite(0);
    scale[0].setPivot(scale[0].width() / 2, scale[0].height());
    scale[1].setColorDepth(4);
    scale[1].setPsram(true);
    scale[1].createSprite(radius / 40, radius / 7);
    scale[1].fillSprite(0);
```

```
    scale[1].setPivot(scale[1].width() / 2, scale[1].height());
}

void Thermometer::makeHand() {
    float height, width;
    height = radius * 0.8f;
    width = height * 0.1f;

    hand.setColorDepth(4);
    hand.setPsram(true);
    hand.createSprite(width, height);
    hand.fillSprite(15);
    hand.setColor(0);
    hand.fillTriangle(width / 2.f, 0, 0, height / 4.f, width, height / 4.f);
    hand.fillTriangle(0, height / 4.f, width, height / 4.f, width / 2.f,
                      height);
    hand.setPivot(width / 2., height);
}

void Thermometer::makeMeterFace(int min, int max, const char *unit) {
    face.setColorDepth(4);
    face.setPsram(true);
    face.createSprite(sizex, sizey);
    face.fillSprite(15);
    face.setColor(0);
    face.setFont(&fonts::lgfxJapanGothic_36);
    face.setTextColor(0, 15);
    face.setTextDatum(middle_center);
    float center[2] = {sizex / 2.0f, sizey / 2.0f};
    face.fillCircle(center[0], center[1], radius);
    face.fillCircle(center[0], center[1], radius * 0.95, 15);
    float angleInterval = 270.f / (float)(max - min);
    for (int i = min; i <= max; i += 2) {
        LGFX_Sprite *use_scale = (i % 10 == 0) ? &scale[0] : &scale[1];
        float angle = (270.f - 45.f) - (float)(i - min) * angleInterval;
        float angleRad = angle * 3.14159265f / 180.f;
        float startx =
            (radius - use_scale->height()) * cos(angleRad) + center[0];
        float starty =
            -1.0f * ((radius - use_scale->height()) * sin(angleRad)) +
            center[1];
        use_scale->pushRotateZoom(&face, startx, starty, 90.f - angle, 1.f,
                                  1.f);
        if (i % 10 == 0) {
            float charsize = (float)scale[0].height() / 36.f;
            float charx =
                (radius - use_scale->height() * 1.5f) * cos(angleRad) +
                center[0];
            float chary =
                -1.f * (radius - use_scale->height() * 1.5f) * sin(angleRad) +
                center[0];
            face.setTextSize(charsize);
            face.drawNumber(i, charx, chary);
        }
        face.drawString(unit, center[0], sizey / 5.f * 3.f);
    }
}

```

```
78  void Thermometer::drawTempMeter(LovyanGFX *lcd, int x, int y) {
79      makeMeterFace(0, 50, "℃");
80      float center[2] = {(float)sizex / 2.f, (float)sizey / 2.f};
81      float angle = 270.f - 45.f;
82      angle -= 270.f / 50.f * get_temp();
83      hand.pushRotateZoom(&face, center[0], center[1], 90.f - angle, 1.f, 1.f);
84      face.pushSprite(lcd, x, y);
85  }
86
87  void Thermometer::drawHumMeter(LovyanGFX *lcd, int x, int y) {
88      makeMeterFace(20, 80, "%");
89      float center[2] = {(float)sizex / 2.f, (float)sizey / 2.f};
90      float angle = 270.f - 45.f;
91      angle -= 270.f / 60.f * (get_hum() - 20.f);
92      hand.pushRotateZoom(&face, center[0], center[1], 90.f - angle, 1.f, 1.f);
93      face.pushSprite(lcd, x, y);
94  }
95
96  void Thermometer::drawString(LovyanGFX *lcd, int x, int y) {
97      M5.SHT30.UpdateData();
98      LGFX_Sprite meter(lcd);
99      meter.setColorDepth(4);
100     meter.setPsram(true);
101     meter.createSprite(250, 100);
102     meter.fillSprite(15);
103     meter.setColor(0);
104     meter.setTextColor(0, 15);
105     meter.setFont(&fonts::lgfxJapanMinchoP_36);
106     meter.setCursor(10, 10);
107     meter.printf("温度:%5.1f℃", this->get_temp());
108     meter.setCursor(10, 50);
109     meter.printf("湿度:%5.1f%%", this->get_hum());
110     meter.pushSprite(x, y);
111 }
112
113 float Thermometer::get_temp() {
114     M5.SHT30.UpdateData();
115     this->temp = M5.SHT30.GetTemperature();
116     return this->temp;
117 }
118
119 float Thermometer::get_hum() {
120     M5.SHT30.UpdateData();
121     this->hum = M5.SHT30.GetRelHumidity();
122     return this->hum;
123 }
```

4 ネットからの情報取得

4.1 infoFromNet.hpp

```
// ネットワークより取得する情報関連
// 時計・天気予報
#include <M5EPD.h>

#define WeatherFileName "/weather.jsn"

class GetInfoFromNetwork {
   private:
    bool wifiOn(void);
    void wifiOff(void);

   public:
    GetInfoFromNetwork();
    ~GetInfoFromNetwork();
    int isWiFiOn(void);
    int setNtpTime();
    String getWeatherQuery();
    bool getWeatherInfo();
};
```

4.2 wifiid.h

```
// wifiのssdiとパスワード
#define ssid "LAKINET"
#define password "mitomiilaki31412101218"
```

4.3 apikey.h

```
#define apikey "b957044d4e4e6a5b1bfb05fc0ecf9255"
```

4.4 infoiFromNet.cpp

```
#include "infoFromNet.hpp"

#include <HTTPClient.h>
#include <M5EPD.h>
#include <WiFi.h>
#include <sys/time.h>
#include <time.h>

#include "util.h"

// wifiid.hには、ssid,passwordの各defineを定義を文字列として記載すること。
// このファイルは、.gitignoreとする。
#include "wifiid.h"
// apikey.hには、apikeyのdefineの定義を文字列として記載すること。
// このファイルは、.gitignoreとする。
```

```
16 #include "apikey.h"
17
18 GetInfoFromNetwork::GetInfoFromNetwork() { wifiOn(); }
19
20 GetInfoFromNetwork::~GetInfoFromNetwork() { wifiOff(); }
21
22 bool GetInfoFromNetwork::wifiOn(void) {
23     WiFi.begin(ssid, password);
24     for (int i = 0; i < 10; i++) {
25         if (isWiFiOn()) return true;
26         delay(500);
27     }
28     return false;
29 }
30
31 void GetInfoFromNetwork::wifiOff(void) {
32     WiFi.disconnect(true);
33     WiFi.mode(WIFI_OFF);
34     return;
35 }
36
37 int GetInfoFromNetwork::isWiFiOn(void) {
38     return (WiFi.status() == WL_CONNECTED);
39 }
40
41 int GetInfoFromNetwork::setNtpTime() {
42     Serial.println("NTP時刻補正開始1");
43     if (!isWiFiOn()) return -1;
44     const long gmtOffset_sec = 0;
45     const int daylightOffset_sec = 0;
46     const char* ntpServer = "jp.pool.ntp.org";
47
48     struct timeval reset = {0, 0};
49     settimeofday(&reset, NULL);
50     configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
51     struct tm timeinfo;
52     time_t t = time(NULL);
53     int i = 0;
54     while (t < 1000000000L && i < 100) {  // システム時刻の設定完了を待つ。
55         t = time(NULL);
56         i++;
57         delay(100);
58     }
59
60     if (t < 1000000000L) return -1;
61     t += 9 * 3600;
62     gmtime_r(&t, &timeinfo);
63
64     char fmt[] = "%Y%m%d %H:%M:%S";
65     char s[50];
66     strftime(s, sizeof(s), fmt, gmtime(&t));
67     Serial.printf("設定時刻:%s(time_t:%ld, loop:%d)\n", s, t, i);
68
69     rtc_time_t rtcTime;
70     rtcTime.hour = (int8_t)timeinfo.tm_hour;
71     rtcTime.min = (int8_t)timeinfo.tm_min;
72     rtcTime.sec = (int8_t)timeinfo.tm_sec;
73     rtc_date_t rtcDate;
```

```
    rtcDate.year = (int8_t)timeinfo.tm_year + 1900;
    rtcDate.mon = (int8_t)timeinfo.tm_mon + 1;
    rtcDate.day = (int8_t)timeinfo.tm_mday;
    M5.RTC.setDate(&rtcDate);
    M5.RTC.setTime(&rtcTime);

    return 0;
}

const uint8_t fingerprint[20] = {0xEE, 0xAA, 0x58, 0x6D, 0x4F, 0x1F, 0x42,
                                 0xF4, 0x18, 0x5B, 0x7F, 0xB0, 0xF2, 0x0A,
                                 0x4C, 0xDD, 0x97, 0x47, 0x7D, 0x99};
#define OpenWeatherUrl "api.openweathermap.org"
#define City "Nagahama,JP"

String GetInfoFromNetwork::getWeatherQuery() {
    String url("/data/2.5/forecast?");
    url += "q=" City;
    url += "&appid=" apikey;
    url += "&lang=ja&units=metric";
    return url;
}

// 天気予報データをSDカードのWeatherFileNameに書き込む。
bool GetInfoFromNetwork::getWeatherInfo() {
    if (!isWiFiOn()) return false;
    HTTPClient http;
    File file;
    String url = String("http://") + String(OpenWeatherUrl) + getWeatherQuery();
    Serial.print("URL:");
    Serial.println(url);
    if (!http.begin(url)) return false;
    int retCode = http.GET();
    if (retCode < 0) goto http_err;
    if (retCode != HTTP_CODE_OK && retCode != HTTP_CODE_MOVED_PERMANENTLY)
        goto http_err;
    if (!SD.exists("/")) goto http_err;
    Serial.println("SD OK!");
    if (SD.exists(WeatherFileName)) SD.remove(WeatherFileName);
    file = SD.open(WeatherFileName, FILE_WRITE);
    if (!file) goto http_err;
    Serial.println("ファイルオープン完了");
    if (http.writeToStream(&file) < 0) goto file_err;
    file.close();
    Serial.println("weatherファイルへのjsonデータ書き込み完了");
    http.end();
    return true;

file_err:
    file.close();
http_err:
    http.end();
    return false;
}
```

5 時計の表示

5.1 tokei.hpp

```
/************************************************
 * 時計の表示
 ************************************************/

#include <M5EPD.h>
#define LGFX_M5PAPER
#include <LovyanGFX.hpp>

class Tokei {
   private:
    int year, month, day;
    int hour, min, sec;
    int dayOfTheWeek;
    int width, height;

    void getDateTime();
    int getDayOfTheWeek(int year, int month, int day);

   public:
    Tokei(int sizex = 200, int sizey = 200);
    void drawDigitalTokei(LovyanGFX *lcd, int x, int y);
};
```

5.2 tokei.cpp

```
/************************************************
 * 時計の表示
 ************************************************/

#include <M5EPD.h>
#define LGFX_M5PAPER
#include <LovyanGFX.hpp>
#include <time.h>

#include "tokei.hpp"
#include "util.h"

Tokei::Tokei(int width, int height) : width(width), height(height) {
    getDateTime();
    dayOfTheWeek = getDayOfTheWeek(year, month, day);
}

// 現在時刻を取得する。
void Tokei::getDateTime() {
    time_t t = now();
    struct tm  datetime;
    gmtime_r(&t, &datetime);

    year = datetime.tm_year + 1900;
    month = datetime.tm_mon + 1;
    day = datetime.tm_mday;
```

```
    hour = datetime.tm_hour;
    min = datetime.tm_min;
    sec = datetime.tm_sec;
}

// 曜日の計算。月曜日を0、日曜日を6とする。
int Tokei::getDayOfTheWeek(int year, int month, int day) {
    int y = year % 100;
    int c = y / 100;
    int ganma = 5 * c + c / 4;
    return (day + (26 * (month + 1)) / 10 + y + y / 4 + ganma + 5) % 7;
}

// デジタル時計を描画する
void Tokei::drawDigitalTokei(LovyanGFX* lcd, int x, int y) {
    // 描画領域区分比率
    const float tokei_ratio = 0.75f;
    // スプライト初期化
    LGFX_Sprite tokei;
    tokei.setPsram(true);
    tokei.setColorDepth(4);
    tokei.createSprite(width, height);
    tokei.fillSprite(15);
    // tokei.drawRect(0,0,width,height,0); // レイアウト検討用外枠
    // 時計時間部分表示
    char strTime[6];
    sprintf(strTime, "%02d:%02d", hour, min);
    tokei.setFont(&fonts::Font7);  // font高さ:48
    tokei.setTextColor(0, 15);
    float mag = (height * tokei_ratio) / 48.f;
    tokei.setTextSize(mag, mag);
    tokei.drawString(strTime, 0.f, height * (1 - tokei_ratio));
    // 時計日時部分表示
    const char* youbi_tbl[] = {"月", "火", "水", "木", "金", "土", "日"};
    const char* youbi = youbi_tbl[dayOfTheWeek];
    char strDate[30];
    sprintf(strDate, "%d年%2d月%2d日(%s)", year, month, day, youbi);
    tokei.setFont(&fonts::lgfxJapanGothic_36);
    mag = (height * (1 - tokei_ratio)) / 36.f;
    tokei.setTextSize(mag, mag);
    tokei.drawString(strDate, 0.f, 0.f);
    tokei.pushSprite(lcd, x, y);
}
```

6 天気予報データの管理

6.1 tenki.hpp

```
// 天気予報のJSONの解析及び値の提供

#include <ArduinoJson.h>
#include <M5EPD.h>

#include "util.h"

#ifndef TENKI
#define TENKI
struct Allocator {
    void *allocate(size_t size) { return ps_malloc(size); }
    void deallocate(void *ptr) { heap_caps_free(ptr); }
    void *reallocate(void *ptr, size_t new_size) {
        return ps_realloc(ptr, new_size);
    }
};

class Tenki {
   private:
    // DynamicJsonDocument json;
    BasicJsonDocument<Allocator> json;
    DeserializationError jsonErr;
    bool _isEnable;

   public:
    Tenki();
    // 情報が有効か否かを返す。
    bool isEnable() { return _isEnable; }
    // リストナンバーの最大値を返す。
    int getMaxListNo() { return json["list"].size() - 1; };
    // 指定時刻のデータのリストIDを取得する。
    int getListIdSpecifiedTime(time_t time);
    // 現在時刻のh時間後以降の最小時刻のデータのリストNo.を取得する
    int getListIdAfterHHour(int h) {
        time_t searchTime = now() + h * 3600;
        return getListIdSpecifiedTime(searchTime);
    }
    // 指定リストナンバーの時刻を取得する。
    time_t getDate(int i) {
        return json["list"][i]["dt"].as<time_t>() + 9 * 3600;
    };
    // 指定リストナンバーの天気を取得する。
    const char *getWeather(int i) {
        return json["list"][i]["weather"][0]["description"].as<const char *>();
    };
    // 指定リストナンバーの気温を取得する。
    float getTemp(int i) {
        return json["list"][i]["main"]["temp"].as<float>();
    };
    // 指定リストナンバーの風向を取得する
    const char *getWindDir(int i);
    // 指定リストナンバーの風速を取得する。
```

```
53      float getWindSpeed(int i) {
54          return json["list"][i]["wind"]["speed"].as<float>();
55      };
56      // 指定リストナンバーの風力階級を取得する
57      int getBeaufortScale(int i);
58      // 天気情報を最新のSDカードデータに更新する。
59      void refresh() {
60          _isEnable = false;
61          readJson();
62      };
63
64     private:
65      void readJson();
66  };
67  #endif
```

6.2 tenki.cpp

```
1   // 天気予報JSONデータの解析　及び　値の提供
2
3   #include "tenki.hpp"
4
5   #include <M5EPD.h>
6   #include <time.h>
7
8   #include "infoFromNet.hpp"
9   #include "util.h"
10
11  Tenki::Tenki() : json(22528), _isEnable(false) { readJson(); }
12
13  void Tenki::readJson() {
14      if (!SD.exists(WeatherFileName)) return;
15      File f = SD.open(WeatherFileName);
16      if (!f) return;
17      jsonErr = deserializeJson(json, f);
18      if (jsonErr == DeserializationError::Ok) _isEnable = true;
19      return;
20  }
21
22  // 指定リストナンバーの風力階級を取得する
23  int Tenki::getBeaufortScale(int i) {
24      static const float speed_tbl[] = {0.2,  1.5,  3.3,  5.4,  7.9,  10.7,
25                                        13.8, 17.1, 20.7, 24.4, 28.4, 32.6};
26      float speed = getWindSpeed(i);
27      int scale = 0;
28      while (scale < 12 && speed > speed_tbl[scale]) {
29          scale++;
30      }
31      return scale;
32  }
33
34  // 指定リストナンバーの風向を取得する
35  // 内部文字列のポインターを返す。
36  const char* Tenki::getWindDir(int i) {
37      static const char* dir[] = {"北", "北東", "東", "南東",
38                                  "南", "南西", "西", "北西"};
39      int deg = json["list"][i]["wind"]["deg"];
```

```
40      int idx = ((2 * deg + 45) / 45) % 8;
41      return dir[idx];
42  }
43
44  // 指定時刻のデータのリストIDを取得する。
45  // ない場合は、時刻以降の最も近いデータのIDとする。
46  int Tenki::getListIdSpecifiedTime(time_t time) {
47      int listNo = 0;
48      int max = getMaxListNo();
49      while (listNo <= max && getDate(listNo) < time) {
50          listNo++;
51      }
52      return listNo;
53  }
```

7 天気予報の表示

7.1 drawtenki.hpp

```
// 天気予報の表示
#include <M5EPD.h>
#define LGFX_M5PAPER
#include <LovyanGFX.hpp>

#include "tenki.hpp"

class DrawTenki {
   private:
    Tenki *tenki;
    LGFX_Sprite screen;
    int width;
    int height;
    float columnX[4];
    float rowCenterY[3];

   public:
    DrawTenki(Tenki *tenki, int width, int height);
    void draw(LovyanGFX *lcd, int x, int y);

   private:
    void drawFrameBorder();
    void calcColumnCoordinate();
    void drawTenkiInfo(int lineNo, int listNo);
    int getListNoFor1stLine();
    int getListNoFor2ndLine() { return tenki->getListIdAfterHHour(24); };
    int getListNoFor3rdLine() { return tenki->getListIdAfterHHour(48); };
};
```

7.2 drawtenki.cpp

```
// 天気予報の表示
#include <M5EPD.h>
#define LGFX_M5PAPER
#include <time.h>

#include <LovyanGFX.hpp>

#include "drawtenki.hpp"

DrawTenki::DrawTenki(Tenki *tenki, int width, int height)
    : tenki(tenki), width(width), height(height) {
    screen.setColorDepth(4);
    screen.setPsram(true);
    screen.createSprite(width, height);
    screen.fillSprite(15);
    screen.setColor(0);
    screen.setTextColor(0, 15);
    screen.setFont(&fonts::lgfxJapanGothic_36);
    float textsize = (((float)height - 10.f) / 3.f) / 36.f;
    screen.setTextSize(textsize);
```

```
21      calcColumnCoordinate();
22  }
23
24  // 各カラムの水平垂直座標の計算
25  void DrawTenki::calcColumnCoordinate() {
26      columnX[0] = 2.f;
27      columnX[1] = columnX[0] + screen.textWidth("88日88時") + 3;
28      columnX[2] = columnX[1] + screen.textWidth("激しい雨") + 3;
29      columnX[3] = columnX[2] + screen.textWidth("00℃") + 3;
30      rowCenterY[0] = height / 6.f;
31      for (int i = 1; i <= 2; i++) {
32          rowCenterY[i] = rowCenterY[i - 1] + height / 3.f;
33      }
34  }
35
36  // フレーム枠の表示
37  void DrawTenki::drawFrameBorder() {
38      screen.drawRect(0, 0, width, height);
39      for (int i = 1; i <= 2; i++) {
40          screen.drawLine(0, height / 3 * i, width, height / 3 * i);
41      }
42      for (int i = 1; i <= 3; i++) {
43          screen.drawLine(columnX[i] - 1, 0, columnX[i] - 2, height);
44      }
45  }
46
47  // 一行の天気情報を描画する。 lineno:0-2
48  void DrawTenki::drawTenkiInfo(int lineNo, int listNo) {
49      screen.setTextDatum(middle_left);
50
51      //時間
52      time_t dataTimet = tenki->getDate(listNo);
53      struct tm *dataTm = gmtime(&dataTimet);
54      char textbuf[100];
55      sprintf(textbuf, "%2d日%2d時", dataTm->tm_mday, dataTm->tm_hour);
56      screen.drawString(textbuf, columnX[0], rowCenterY[lineNo]);
57      //天気
58      screen.drawString(tenki->getWeather(listNo), columnX[1],
59                        rowCenterY[lineNo]);
60      //気温
61      sprintf(textbuf, "%2d℃", (int)tenki->getTemp(listNo));
62      screen.drawString(textbuf, columnX[2], rowCenterY[lineNo]);
63      //風向、風力
64      const char *windDir = tenki->getWindDir(listNo);
65      int beaufortScale = tenki->getBeaufortScale(listNo);
66      sprintf(textbuf, "%s %d", windDir, beaufortScale);
67      screen.drawString(textbuf, columnX[3], rowCenterY[lineNo]);
68  }
69
70  int DrawTenki::getListNoFor1stLine() {
71      time_t nowDateTime = now();
72      int nowhour = (nowDateTime / 3600) % 24;
73      time_t todayStart = nowDateTime / (24 * 3600) * (24 * 3600);
74      time_t searchDateTime;
75      time_t morning = 5 * 3600;
76      time_t evening = 18 * 3600;
77      if (nowhour < 5) {
78          searchDateTime = todayStart + morning;
```

```
79      } else if (nowhour >= 18) {
80          searchDateTime = todayStart + (24 * 3600) + morning;
81      } else {
82          searchDateTime = todayStart + evening;
83      }
84      return tenki->getListIdSpecifiedTime(searchDateTime);
85  }
86
87  void DrawTenki::draw(LovyanGFX *lcd, int x, int y) {
88      drawFrameBorder();
89      drawTenkiInfo(0, getListNoFor1stLine());
90      drawTenkiInfo(1, getListNoFor2ndLine());
91      drawTenkiInfo(2, getListNoFor3rdLine());
92      screen.pushSprite(lcd, x, y);
93  }
```

8 SD カード内の jpeg ファイルをスキャン

8.1 scanfile.hpp

```
// Jpegファイルの一覧を生成する。

#define blockSize 10
#define filenameSize 14  // 1+8+1+3+1(8.3形式に限る。'/'が付加される)

struct FileNames {
    char filenames[blockSize][filenameSize];
    FileNames *prev;
    FileNames *next;
};

class JpegFiles {
   private:
    FileNames *top;
    FileNames *cur;
    int _count;

   public:
    JpegFiles();
    char *operator[](int i);
    int count() { return _count; };

   private:
    FileNames *addBlock(FileNames *last);
    void addFilename(const char *filename);
    bool isJpegFile(const char *filename);
};
```

8.2 scanfile.cpp

```
// Jpegファイルの一覧

#include "scanfile.hpp"

#include <M5EPD.h>
#include <SD.h>

JpegFiles::JpegFiles() : top(NULL), cur(NULL), _count(0) {
    if (!SD.exists("/")) return;
    File root = SD.open("/");
    File entry = root.openNextFile();
    while (entry) {
        if (!entry.isDirectory()) {
            if (isJpegFile(entry.name())) {
                addFilename(entry.name());
            }
        }
        entry = root.openNextFile();
    }
    entry.close();
    root.close();
```

```
}

FileNames *JpegFiles::addBlock(FileNames *last) {
    FileNames *addBlock = (FileNames *)ps_malloc(sizeof(FileNames));
    if (!addBlock) return NULL;
    if (last) last->next = addBlock;
    addBlock->prev = last;
    addBlock->next = NULL;
    return addBlock;
}

void JpegFiles::addFilename(const char *filename) {
    int itemInd = _count % blockSize;
    if (itemInd == 0) {
        cur = addBlock(cur);
        if (!top) top = cur;
    }
    strcpy(cur->filenames[itemInd], filename);
    _count++;
}

char *JpegFiles::operator[](int i) {
    int itemInd = i % blockSize;
    int blockNo = i / blockSize;

    if (i >= _count) return NULL;
    FileNames *block = top;
    for (int b = 0; b < blockNo; b++) {
        if (!block) return NULL;
        block = block->next;
    }
    return block->filenames[itemInd];
}

bool JpegFiles::isJpegFile(const char *filename) {
    const char *extBig = ".JPG";
    const char *extSmall = ".jpg";

    int check = 0;
    for (int i = 0; filename[i] != '\0'; i++) {
        if (filename[i] == extBig[check] || filename[i] == extSmall[check]) {
            check++;
        } else {
            if (check > 0 && (filename[i] == extBig[check] ||
                              filename[i] == extSmall[check])) {
                check = 1;
            } else {
                check = 0;
            }
        }
    }
    return extBig[check] == '\0';
}
```

9 メッセージボックス

9.1 messageArea.h

```
// メッセージエリアを作成し、管理する。
#include <M5EPD.h>
#define LGFX_M5PAPER
#include <LovyanGFX.hpp>

#define MAX_LINE_CNT 5
#define MAX_LINE_LENGTH 80

class MessageArea {
   private:
    LGFX_Sprite mesg;
    int line_cnt;
    char line_buff[MAX_LINE_CNT][MAX_LINE_LENGTH + 1];
    bool changed_text;
    bool changed_sprite;
    bool flame;

   public:
    MessageArea(int width, int height, int line, bool flame = false);
    // ソース文字列の内容をバッファにセットする。line_noは0始まり
    MessageArea *setText(const char *source, int line_no);
    // バッファの内容を必要があれば、スプライトに書き込む
    MessageArea *flush(void);
    // スプライトに変更があれば、画面を描画する。
    // forceが指定されれば、必ず描画する。
    MessageArea *draw(LovyanGFX *lcd, int x, int y, bool force = false);
    // 現在のバッファの内容を返す。line_noは0始まり
    char *getText(char *buff, int buffSize, int line_no);

   private:
};
```

9.2 messageArea.cpp

```
// メッセージエリアを作成し、表示管理する。
#include "messageArea.hpp"

MessageArea::MessageArea(int width, int height, int line, bool flame)
    : line_cnt(line), changed_text(false), changed_sprite(false), flame(flame) {
    if (line_cnt > MAX_LINE_CNT) line_cnt = MAX_LINE_CNT;
    mesg.setPsram(true);
    mesg.setColorDepth(4);
    mesg.createSprite(width, height);
    mesg.setFont(&fonts::lgfxJapanGothic_36);
    float line_height =
        (float)(height - 2 * (line_cnt - 1) - 2) / (float)line_cnt;
    float text_size = (float)line_height / (float)mesg.fontHeight();
    mesg.setTextSize(text_size);
    mesg.setTextColor(0, 15);
    for (int i = 0; i < MAX_LINE_CNT; i++) {
        line_buff[i][MAX_LINE_LENGTH] = '\0';
```

```
        line_buff[i][0] = '\0';
    }
}

// ソース文字列の内容をバッファにセットする。line_noは0始まり
// バッファの内容に変更があるかどうかをチェックし、必要がない場合書き換えない。
MessageArea *MessageArea::setText(const char *source, int line_no) {
    if (line_no >= line_cnt) line_no = line_cnt - 1;
    if (strcmp(source, line_buff[line_no]) == 0) return this;
    strncpy(line_buff[line_no], source, MAX_LINE_LENGTH);
    line_buff[line_no][MAX_LINE_LENGTH] = '\0';
    changed_text = true;
    return this;
};

// バッファの内容を必要があれば、スプライトに書き込む
MessageArea *MessageArea::flush(void) {
    if (changed_text) {
        int line_height = mesg.fontHeight();
        mesg.fillSprite(15);
        int width = mesg.width();
        int height = mesg.height();
        if (flame) mesg.drawRect(0, 0, width, height, 0);
        for (int i = 0; i < line_cnt; i++) {
            int y = line_height * i + 2 * i + 1;
            mesg.drawString(line_buff[i], 1, y);
        }
        changed_text = false;
        changed_sprite = true;
    }
    return this;
}

// スプライトに変更があれば、画面を描画する。
// forceが指定されれば、必ず描画する。
MessageArea *MessageArea::draw(LovyanGFX *lcd, int x, int y, bool force) {
    if (changed_sprite || force) {
        mesg.pushSprite(lcd, x, y);
        changed_sprite = false;
    }
    return this;
}

// 現在のバッファの内容を返す。line_noは0始まり
char *MessageArea::getText(char *buff, int buffSize, int line_no) {
    strncpy(buff, line_buff[line_no], buffSize);
    if (buffSize <= strlen(line_buff[line_no])) buff[buffSize] = '\0';
    return buff;
}
```

10 ユーティリティー

10.1 util.h

```
// ユーティリティー関数

#include "time.h"

#ifndef ENVIRONMENT_UTIL
#define ENVIRONMENT_UTIL
time_t now();
void updateSystemTime();
#endif
```

10.2 util.cpp

```
// ユーティリティー関数
#include <M5EPD.h>
#include <sys/time.h>
#include <time.h>

#define MinTimet 1000000000L  // UNIX billennium 2001/9/9 01:46:40 UTC

void updateSystemTime() {
    struct tm setTime;
    struct timeval systime = {0, 0};
    rtc_time_t rtcTime;
    rtc_date_t rtcDate;

    Serial.println("システム時刻設定");
    settimeofday(&systime, NULL);
    M5.RTC.getDate(&rtcDate);
    M5.RTC.getTime(&rtcTime);

    setTime.tm_year = rtcDate.year - 1900;
    setTime.tm_mon = rtcDate.mon - 1;
    setTime.tm_mday = rtcDate.day;
    setTime.tm_hour = rtcTime.hour;
    setTime.tm_min = rtcTime.min;
    setTime.tm_sec = rtcTime.sec;

    systime.tv_sec = mktime(&setTime);
    systime.tv_usec = 0;

    settimeofday(&systime, NULL);
    Serial.println("システム時刻安定待ち");
    while (abs(time(NULL) - systime.tv_sec) > 100) {
        Serial.print(".");
        delay(100);
    }
    Serial.println("システム時刻設定終了");
}

time_t now() {
    time_t t = time(NULL);
```

```
40      if (t < MinTimet) {
41          updateSystemTime();
42          t = time(NULL);
43          if (t < MinTimet) t = 0;
44      }
45      return t;
46  }
```