M5Paper 環境モニターソース

美都

2021 年 3 月 26 日

目次

1 メイン 2

2 バッテリーメーター 5
- 2.1 battery.h . . . 5
- 2.2 battery.cpp . . . 5

3 温湿度計 7
- 3.1 thermometer.hpp . . . 7
- 3.2 thermometer.cpp . . . 7

4 ネットからの情報取得 10
- 4.1 infoFromNet.hpp . . . 10
- 4.2 wifiid.h . . . 10
- 4.3 apikey.h . . . 10
- 4.4 infoiFromNet.cpp . . . 10

5 時計の表示 13
- 5.1 tokei.hpp . . . 13
- 5.2 tokei.cpp . . . 13

6 天気予報データの管理 15
- 6.1 tenki.hpp . . . 15
- 6.2 tenki.cpp . . . 16

7 天気予報の表示 18
- 7.1 drawtenki.hpp . . . 18
- 7.2 drawtenki.cpp . . . 18

8 SD カード内の jpeg ファイルをスキャン 21
- 8.1 scanfile.hpp . . . 21
- 8.2 scanfile.cpp . . . 21

9 ユーティリティー 23
- 9.1 util.h . . . 23
- 9.2 util.cpp . . . 23

1 メイン

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

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

//#define MEMPRINT

static LGFX lcd;

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);
    printMem("バッテリー描画後のメモリ");

    printMem("時計描画前のメモリ");
    Tokei *tokei = new Tokei(300, 100);
    tokei->drawDigitalTokei(&lcd, 630, 50);
    printMem("時計描画後のメモリ");
    delete tokei;

    printMem("温度計描画前のメモリ");
    Thermometer *t = new Thermometer(200,200);
    t->drawTempMeter(&lcd, 530, 180);
    t->drawHumMeter(&lcd, 750, 180);
    printMem("温度計描画後のメモリ");
    delete t;

    printMem("お天気情報確保前のメモリ");
    Tenki *tenki = new Tenki();
    DrawTenki *drawTenki = new DrawTenki(tenki, 455, 112);
    drawTenki->draw(&lcd, 495, 408);
    printMem("お天気描画後のメモリ");
    delete drawTenki;
    delete tenki;

    //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。
    JpegFiles jpgs;
    char *filename = jpgs[random(jpgs.count())];
    Serial.print("写真番号:");
```

```
    Serial.println(filename);
    lcd.drawJpgFile(SD,filename, 10, 110);
    delay(500);
}

// ●分ピッタリまでの秒数
int rest_minute() {
    rtc_time_t time;
    M5.RTC.getTime(&time);
    return 60-time.sec;
}

// シャットダウンを試みる。通電中はすり抜ける
void challengeShutdown() {
    int rest_sec = rest_minute()-6;
    if (rest_sec < 30) rest_sec += 60;
    M5.shutdown(rest_sec); // 一旦停止
}

void checkInfoFromNetwork(bool always=false) {
    rtc_time_t time;
    rtc_date_t date;
    M5.RTC.getDate(&date);
    M5.RTC.getTime(&time);
    time_t outdated = now() - 6*3600;
    Tenki tenki;

    if (!tenki.isEnable() || (time.hour%6==0 && time.min == 3)
            || tenki.getDate(0)<outdated || date.year < 2020) {
        Serial.println("ネットワークの情報の取得開始");
        GetInfoFromNetwork info;
        info.setNtpTime();
        info.getWeatherInfo();
        tenki.refresh();
    }
}

void setup()
{
    M5.begin(false, true, true, true, true);
    M5.BatteryADCBegin();
    M5.RTC.begin();
    M5.SHT30.Begin();
    SD.begin();
    lcd.init();
    lcd.setRotation(1);
    randomSeed(analogRead(0));

    checkInfoFromNetwork();

    drawLcd();
    challengeShutdown();
}


void loop()
{
    delay((rest_minute()+1)*1000);
```

```
113     checkInfoFromNetwork();
114     drawLcd();
115     challengeShutdown();
116 }
```

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      return rest_battery;
43  }
```

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);
```

```
21      scale[1].setPivot(scale[1].width()/2, scale[1].height());
22  }
23
24  void Thermometer::makeHand() {
25      float height, width;
26      height = radius * 0.8f;
27      width = height * 0.1f;
28
29      hand.setColorDepth(4);
30      hand.setPsram(true);
31      hand.createSprite(width, height);
32      hand.fillSprite(15);
33      hand.setColor(0);
34      hand.fillTriangle(width/2.f, 0, 0, height/4.f, width, height/4.f);
35      hand.fillTriangle(0, height/4.f, width, height/4.f, width/2.f, height);
36      hand.setPivot(width/2., height);
37  }
38
39  void Thermometer::makeMeterFace(int min, int max, const char* unit) {
40      face.setColorDepth(4);
41      face.setPsram(true);
42      face.createSprite(sizex, sizey);
43      face.fillSprite(15);
44      face.setColor(0);
45      face.setFont(&fonts::lgfxJapanGothic_36);
46      face.setTextColor(0, 15);
47      face.setTextDatum(middle_center);
48      float center[2] = {sizex/2.0f, sizey/2.0f};
49      face.fillCircle(center[0], center[1], radius);
50      face.fillCircle(center[0], center[1], radius*0.95, 15);
51      float angleInterval = 270.f / (float)(max-min);
52      for (int i = min ; i <= max ; i+=2) {
53          LGFX_Sprite *use_scale = (i%10==0) ? &scale[0] : &scale[1];
54          float angle = (270.f-45.f) - (float)(i-min) * angleInterval;
55          float angleRad = angle * 3.14159265f / 180.f ;
56          float startx = (radius - use_scale->height()) * cos(angleRad) + center[0];
57          float starty = -1.0f * ((radius - use_scale->height()) * sin(angleRad)) + center[1];
58          use_scale->pushRotateZoom(&face, startx, starty, 90.f-angle, 1.f, 1.f);
59          if (i%10==0) {
60              float charsize = (float)scale[0].height() / 36.f;
61              float charx = (radius - use_scale->height() * 1.5f) * cos(angleRad) + center[0];
62              float chary = -1.f * (radius - use_scale->height() * 1.5f) * sin(angleRad) +
                  center[0];
63              face.setTextSize(charsize);
64              face.drawNumber(i, charx, chary);
65          }
66          face.drawString(unit, center[0], sizey/5.f*3.f);
67      }
68
69  }
70
71  void Thermometer::drawTempMeter(LovyanGFX *lcd, int x, int y) {
72      makeMeterFace(0, 50, "℃");
73      float center[2] = {(float)sizex/2.f, (float)sizey/2.f};
74      float angle = 270.f - 45.f;
75      angle -= 270.f / 50.f * get_temp();
76      hand.pushRotateZoom(&face, center[0], center[1], 90.f - angle, 1.f, 1.f);
77      face.pushSprite(lcd, x, y);
```

```
}

void Thermometer::drawHumMeter(LovyanGFX *lcd, int x, int y) {
    makeMeterFace(20, 80, "%");
    float center[2] = {(float)sizex/2.f, (float)sizey/2.f};
    float angle = 270.f - 45.f;
    angle -= 270.f / 60.f * (get_hum() - 20.f);
    hand.pushRotateZoom(&face, center[0], center[1], 90.f - angle, 1.f, 1.f);
    face.pushSprite(lcd, x, y);
}

void Thermometer::drawString(LovyanGFX *lcd, int x, int y) {
    M5.SHT30.UpdateData();
    LGFX_Sprite meter(lcd);
    meter.setColorDepth(4);
    meter.setPsram(true);
    meter.createSprite(250, 100);
    meter.fillSprite(15);
    meter.setColor(0);
    meter.setTextColor(0, 15);
    meter.setFont(&fonts::lgfxJapanMinchoP_36);
    meter.setCursor(10,10);
    meter.printf("温度:%5.1f℃", this->get_temp());
    meter.setCursor(10,50);
    meter.printf("湿度:%5.1f%%", this->get_hum());
    meter.pushSprite(x, y);
}

float Thermometer::get_temp() {
    M5.SHT30.UpdateData();
    this->temp = M5.SHT30.GetTemperature();
    return this->temp;
}

float Thermometer::get_hum() {
    M5.SHT30.UpdateData();
    this->hum = M5.SHT30.GetRelHumidity();
    return this->hum;
}
```

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 <M5EPD.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include "infoFromNet.hpp"
#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。
#include "apikey.h"

```

```
GetInfoFromNetwork::GetInfoFromNetwork() {
    wifiOn();
}

GetInfoFromNetwork::~GetInfoFromNetwork() {
    wifiOff();
}

bool GetInfoFromNetwork::wifiOn(void) {
    WiFi.begin(ssid, password);
    for (int i = 0 ; i < 10; i++) {
        if (isWiFiOn()) return true;
        delay(500);
    }
    return false;
}

void GetInfoFromNetwork::wifiOff(void) {
    WiFi.disconnect(true);
    WiFi.mode(WIFI_OFF);
}

int GetInfoFromNetwork::isWiFiOn(void) {
    return (WiFi.status() == WL_CONNECTED) ;
}

int GetInfoFromNetwork::setNtpTime() {
    if (!isWiFiOn()) return -1;
    const long gmtOffset_sec = 9 * 3600;
    const int daylightOffset_sec = 0;
    const char * ntpServer = "jp.pool.ntp.org";

    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
    struct tm timeinfo;
    if (!getLocalTime(&timeinfo)) return -1;

    rtc_time_t rtcTime;
    rtcTime.hour = (int8_t)timeinfo.tm_hour;
    rtcTime.min = (int8_t)timeinfo.tm_min;
    rtcTime.sec = (int8_t)timeinfo.tm_sec ;
    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,0x97,0
      x47,0x7D,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 "tokei.hpp"

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

// RTCより現在時刻を取得する。
void Tokei::getDateTime() {
    rtc_time_t time;
    rtc_date_t date;

    M5.RTC.getTime(&time);
    M5.RTC.getDate(&date);

    year = date.year;
    month = date.mon;
```

```
27      day = date.day;
28      hour = time.hour;
29      min = time.min;
30      sec = time.sec;
31  }
32
33  // 曜日の計算。月曜日を0、日曜日を6とする。
34  int Tokei::getDayOfTheWeek(int year, int month, int day) {
35      int y = year % 100;
36      int c = y / 100;
37      int ganma = 5 * c + c / 4;
38      return (day+(26*(month+1))/10+y+y/4+ganma+5)%7;
39  }
40
41  // デジタル時計を描画する
42  void Tokei::drawDigitalTokei(LovyanGFX *lcd, int x, int y) {
43      // 描画領域区分比率
44      const float tokei_ratio = 0.75f;
45      // スプライト初期化
46      LGFX_Sprite tokei;
47      tokei.setPsram(true);
48      tokei.setColorDepth(4);
49      tokei.createSprite(width,height);
50      tokei.fillSprite(15);
51      //tokei.drawRect(0,0,width,height,0); // レイアウト検討用外枠
52      // 時計時間部分表示
53      char strTime[6];
54      sprintf(strTime, "%02d:%02d", hour, min);
55      tokei.setFont(&fonts::Font7); // font高さ:48
56      tokei.setTextColor(0,15);
57      float mag = (height*tokei_ratio) / 48.f;
58      tokei.setTextSize(mag, mag);
59      tokei.drawString(strTime, 0.f, height*(1-tokei_ratio));
60      // 時計日時部分表示
61      const char* youbi_tbl[] =
62          { "月", "火", "水", "木", "金", "土", "日" };
63      const char* youbi = youbi_tbl[dayOfTheWeek];
64      char strDate[30];
65      sprintf(strDate, "%d年%2d月%2d日(%s)", year, month, day, youbi);
66      tokei.setFont(&fonts::lgfxJapanGothic_36);
67      mag = (height*(1-tokei_ratio)) / 36.f;
68      tokei.setTextSize(mag, mag);
69      tokei.drawString(strDate, 0.f, 0.f);
70      tokei.pushSprite(lcd, x, y);
71  }
```

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 <M5EPD.h>
#include <ArduinoJson.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。
        float getWindSpeed(int i) { return json["list"][i]["wind"]["speed"].as<float>(); };
        // 指定リストナンバーの風力階級を取得する
        int getBeaufortScale(int i) ;
        // 天気情報を最新のSDカードデータに更新する。
        void refresh() { _isEnable=false; readJson(); };

```

```
53      private:
54          void readJson() ;
55
56  };
57  #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 <M5EPD.h>
4   #include <time.h>
5
6   #include "infoFromNet.hpp"
7   #include "tenki.hpp"
8   #include "util.h"
9
10  Tenki::Tenki() : json(22528), _isEnable(false) {
11      readJson();
12  }
13
14  void Tenki::readJson() {
15      if (!SD.exists(WeatherFileName)) return ;
16      File f = SD.open(WeatherFileName);
17      if (!f) return;
18      jsonErr = deserializeJson(json, f);
19      if (jsonErr == DeserializationError::Ok) _isEnable=true;
20      return;
21  }
22
23  // 指定リストナンバーの風力階級を取得する
24  int Tenki::getBeaufortScale(int i) {
25      static const float speed_tbl[] =
26          { 0.2, 1.5, 3.3, 5.4, 7.9, 10.7, 13.8, 17.1, 20.7, 24.4, 28.4, 32.6 };
27      float speed = getWindSpeed(i);
28      int scale = 0;
29      while (scale < 12 && speed > speed_tbl[scale]) { scale++; }
30      return scale;
31  }
32
33  // 指定リストナンバーの風向を取得する
34  // 内部文字列のポインターを返す。
35  const char *Tenki::getWindDir(int i) {
36      static const char* dir[] =
37          {"北", "北東",  "東", "南東", "南", "南西", "西", "北西"};
38      int deg = json["list"][i]["wind"]["deg"];
39      int idx = ((2*deg+45)/45) % 8;
40      return dir[idx];
41  }
42
43  // 指定時刻のデータのリストIDを取得する。
44  // ない場合は、時刻以降の最も近いデータのIDとする。
45  int Tenki::getListIdSpecifiedTime(time_t time) {
46      int listNo = 0;
47      int max = getMaxListNo();
48      while (listNo <= max &&  getDate(listNo) < time) { listNo++; }
49      return listNo;
```

```
50 }
```

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 <LovyanGFX.hpp>
#include <time.h>

#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);
```

```
20      calcColumnCoordinate();
21  }
22
23  // 各カラムの水平垂直座標の計算
24  void DrawTenki::calcColumnCoordinate() {
25      columnX[0] = 2.f;
26      columnX[1] = columnX[0] + screen.textWidth("88日88時") + 3;
27      columnX[2] = columnX[1] + screen.textWidth("激しい雨") + 3;
28      columnX[3] = columnX[2] + screen.textWidth("00℃") + 3;
29      rowCenterY[0] = height/6.f;
30      for (int i = 1; i<=2; i++) {
31          rowCenterY[i] = rowCenterY[i-1] + height/3.f;
32      }
33  }
34
35  // フレーム枠の表示
36  void DrawTenki::drawFrameBorder() {
37      screen.drawRect(0, 0, width, height);
38      for (int i=1; i<=2; i++) {
39          screen.drawLine(0, height/3*i, width, height/3*i);
40      }
41      for (int i=1; i<=3; i++) {
42          screen.drawLine(columnX[i]-1, 0, columnX[i]-2, height);
43      }
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], rowCenterY[lineNo]);
59      //気温
60      sprintf(textbuf, "%2d℃", (int)tenki->getTemp(listNo));
61      screen.drawString(textbuf, columnX[2], rowCenterY[lineNo]);
62      //風向、風力
63      const char *windDir = tenki->getWindDir(listNo);
64      int beaufortScale = tenki->getBeaufortScale(listNo);
65      sprintf(textbuf, "%s %d", windDir, beaufortScale);
66      screen.drawString(textbuf, columnX[3], rowCenterY[lineNo]);
67  }
68
69  int DrawTenki::getListNoFor1stLine() {
70      time_t nowDateTime = now();
71      int nowhour = (nowDateTime / 3600) % 24;
72      time_t todayStart = nowDateTime / (24 * 3600) * (24 * 3600);
73      time_t searchDateTime;
74      time_t morning = 5 * 3600;
75      time_t evening = 18 * 3600;
76      if (nowhour < 5 || nowhour >= 18) {
77          searchDateTime = todayStart + (24 * 3600) + morning;
```

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

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

#include "scanfile.hpp"

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();
    }
```

```
20      entry.close();
21      root.close();
22  }
23
24  FileNames *JpegFiles::addBlock(FileNames *last) {
25      FileNames *addBlock= (FileNames*)ps_malloc(sizeof(FileNames));
26      if (!addBlock) return NULL;
27      if (last) last->next = addBlock;
28      addBlock->prev = last;
29      addBlock->next = NULL;
30      return addBlock;
31  }
32
33  void JpegFiles::addFilename(const char *filename) {
34      int itemInd =  _count % blockSize;
35      if (itemInd == 0) {
36          cur = addBlock(cur);
37          if (!top) top = cur;
38      }
39      strcpy(cur->filenames[itemInd], filename);
40      _count ++;
41  }
42
43  char *JpegFiles::operator[](int i) {
44      int itemInd = i % blockSize;
45      int blockNo = i / blockSize;
46
47      if (i >= _count) return NULL;
48      FileNames *block = top;
49      for (int b=0; b < blockNo; b++) {
50          if (!block) return NULL;
51          block = block->next;
52      }
53      return block->filenames[itemInd];
54  }
55
56  bool JpegFiles::isJpegFile(const char* filename) {
57      const char* extBig = ".JPG";
58      const char* extSmall = ".jpg";
59
60      int check=0;
61      for (int i=0; filename[i]!='\0'; i++) {
62          if (filename[i] == extBig[check] || filename[i] == extSmall[check]) {
63              check ++;
64          } else {
65              if (check > 0 && (filename[i] == extBig[check] || filename[i] == extSmall[check]))
                    {
66                  check = 1;
67              } else {
68                  check = 0;
69              }
70          }
71      }
72      return extBig[check] == '\0';
73  }
```

9 ユーティリティー

9.1 util.h

```
// ユーティリティー関数

#include "time.h"

#ifndef ENVIRONMENT_UTIL
#define ENVIRONMENT_UTIL
time_t now();

#endif
```

9.2 util.cpp

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

time_t now() {
    struct tm time;
    rtc_time_t rtcTime;
    rtc_date_t rtcDate ;

    M5.RTC.getDate(&rtcDate);
    M5.RTC.getTime(&rtcTime);

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

    return mktime(&time);
}
```