

M5Paper 環境モニターソース

美都

2021 年 3 月 25 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	infoFromNet.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	17
7.1	drawtenki.hpp	17
7.2	drawtenki.cpp	17
8	SD カード内の jpeg ファイルをスキャン	19
8.1	scanfile.hpp	19
8.2	scanfile.cpp	19
9	ユーティリティー	21
9.1	util.h	21
9.2	util.cpp	21

1 メイン

```
1  #include <M5EPD.h>
2  #define LGFX_M5PAPER
3  #include <LovyanGFX.hpp>
4
5  #include "battery.h"
6  #include "thermometer.hpp"
7  #include "infoFromNet.hpp"
8  #include "tokei.hpp"
9  #include "tenki.hpp"
10 #include "drawtenki.hpp"
11 #include "util.h"
12 #include "scanfile.hpp"
13
14 // #define MEMPRINT
15
16 static LGFX lcd;
17
18 inline void printMem(const char* msg) {
19 #ifdef MEMPRINT
20     Serial.printf("【%s】 heap:%'d, psram:%'d\n", msg, ESP.getFreeHeap(), ESP.getFreePsram());
21 #endif
22 }
23
24 void drawLcd() {
25     drawBattery(960-120-5, 5, &lcd);
26     printMem("バッテリー描画後のメモリ");
27
28     printMem("時計描画前のメモリ");
29     Tokei *tokei = new Tokei(300, 100);
30     tokei->drawDigitalTokei(&lcd, 630, 50);
31     printMem("時計描画後のメモリ");
32     delete tokei;
33
34     printMem("温度計描画前のメモリ");
35     Thermometer *t = new Thermometer(200, 200);
36     t->drawTempMeter(&lcd, 530, 180);
37     t->drawHumMeter(&lcd, 750, 180);
38     printMem("温度計描画後のメモリ");
39     delete t;
40
41     printMem("お天気情報確保前のメモリ");
42     Tenki *tenki = new Tenki();
43     DrawTenki *drawTenki = new DrawTenki(tenki, 455, 112);
44     drawTenki->draw(&lcd, 495, 408);
45     printMem("お天気描画後のメモリ");
46     delete drawTenki;
47     delete tenki;
48
49     //写真の表示。480*320がちょうど。
50     //プログレスと最適化を無効にすること。
51     //SDカードのルート直下のjpgファイルを対象とする。
52     JpegFiles jpgs;
53     char *filename = jpgs[random(jpgs.count())];
54     Serial.print("写真番号:");
```

```

55     Serial.println(filename);
56     lcd.drawJpgFile(SD,filename, 10, 110);
57     delay(500);
58 }
59
60 // ●分ピッタリまでの秒数
61 int rest_minute() {
62     rtc_time_t time;
63     M5.RTC.getTime(&time);
64     return 60-time.sec;
65 }
66
67 // シャットダウンを試みる。通電中はすり抜ける
68 void challengeShutdown() {
69     int rest_sec = rest_minute()-6;
70     if (rest_sec < 30) rest_sec += 60;
71     M5.shutdown(rest_sec); // 一旦停止
72 }
73
74 void checkInfoFromNetwork(bool always=false) {
75     rtc_time_t time;
76     M5.RTC.getTime(&time);
77     time_t outdated = now() - 6*3600;
78     Tenki tenki;
79
80     if (!tenki.isEnabled() || (time.hour%6==0 && time.min == 3) || tenki.getDate(0)<outdated) {
81         Serial.println("ネットワークの情報の取得開始");
82         GetInfoFromNetwork info;
83         info.setNtpTime();
84         info.getWeatherInfo();
85         tenki.refresh();
86     }
87 }
88
89 void setup()
90 {
91     M5.begin(false, true, true, true, true);
92     M5.BatteryADCBegin();
93     M5.RTC.begin();
94     M5.SHT30.Begin();
95     SD.begin();
96     lcd.init();
97     lcd.setRotation(1);
98     randomSeed(analogRead(0));
99
100     checkInfoFromNetwork();
101
102     drawLcd();
103     challengeShutdown();
104 }
105
106
107 void loop()
108 {
109     delay((rest_minute()+1)*1000);
110     checkInfoFromNetwork();
111     drawLcd();
112     challengeShutdown();

```


2 バッテリメーター

2.1 battery.h

```
1 #include <M5EPD.h>
2 #define LGFX_M5PAPER
3 #include <LovyanGFX.hpp>
4
5 // バッテリー残量を(x,y)に表示する。
6 int drawBattery(int x, int y, LGFX *lcd) ;
7 int get_rest_battery() ;
```

2.2 battery.cpp

```
1 #include "battery.h"
2
3 // バッテリー残量の取得
4 int get_rest_battery() {
5     const int max_vol = 4350;
6     const int min_vol = 3300;
7     //M5.BatteryADCBegin();
8     int voltage = M5.getBatteryVoltage();
9     voltage = max(voltage, min_vol);
10    voltage = min(voltage, max_vol);
11    float rest_battery_raw = (float)(voltage - min_vol) / (float)(max_vol - min_vol);
12    rest_battery_raw = max(rest_battery_raw, 0.01f);
13    rest_battery_raw = min(rest_battery_raw, 1.f);
14    return (int)(rest_battery_raw * 100);
15 }
16
17 // バッテリー残量計の表示
18 int drawBattery(int x, int y, LGFX *lcd) {
19     LGFX_Sprite battery_meter(lcd);
20     int rest_battery = get_rest_battery();
21
22     // バッテリー矩形の表示
23     battery_meter.setColorDepth(4);
24     battery_meter.createSprite(120, 30);
25     battery_meter.fillSprite(15);
26     battery_meter.setColor(0);
27     battery_meter.drawRect(10, 10, 45, 20);
28     battery_meter.fillRect(55, 17, 5, 5);
29     battery_meter.fillRect(10, 10, (int)((45*rest_battery)/100), 20);
30
31     // バッテリー残量文字の表示
32     battery_meter.setFont(&font::lgfxJapanMinchoP_20);
33     battery_meter.setTextSize(1, 1); // 縦,横 倍率
34     battery_meter.setTextColor(0, 15); // 文字色,背景
35     battery_meter.setCursor(62, 10);
36     battery_meter.printf("%d%%", rest_battery);
37
38     lcd->startWrite();
39     battery_meter.pushSprite(x, y);
40     lcd->endWrite();
41 }
```

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

3 温湿度計

3.1 thermometer.hpp

```
1  #include <M5EPD.h>
2  #define LGFX_M5PAPER
3  #include <LovyanGFX.hpp>
4
5  class Thermometer {
6      private:
7          float temp;
8          float hum;
9          int sizex;
10         int sizey;
11         float radius;
12         LGFX_Sprite face;
13         LGFX_Sprite scale[2];
14         LGFX_Sprite hand;
15
16         void makeMeterFace(int min, int max, const char* unit);
17         void makeScale();
18         void makeHand();
19     public:
20         Thermometer(int sizex=200, int sizey=200);
21
22         float get_temp();
23         float get_hum();
24
25         void drawTempMeter(LovyanGFX *lcd, int x, int y );
26         void drawHumMeter(LovyanGFX *lcd, int x, int y);
27         void drawString(LovyanGFX *lcd, int x, int y);
28 };
```

3.2 thermometer.cpp

```
1  #include "thermometer.hpp"
2
3  Thermometer::Thermometer(int sizex, int sizey)
4      : sizex(sizex), sizey(sizey) {
5      M5.SHT30.Begin();
6      radius = min(sizex, sizey)/2*0.95;
7      makeScale();
8      makeHand();
9  };
10
11 void Thermometer::makeScale() {
12     scale[0].setColorDepth(4);
13     scale[0].setPsram(true);
14     scale[0].createSprite(radius/25, radius/5);
15     scale[0].fillSprite(0);
16     scale[0].setPivot(scale[0].width()/2, scale[0].height());
17     scale[1].setColorDepth(4);
18     scale[1].setPsram(true);
19     scale[1].createSprite(radius/40, radius/7);
20     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, "°C");
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);

```



```

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

```

4 ネットからの情報取得

4.1 infoFromNet.hpp

```
1 // ネットワークより取得する情報関連
2 // 時計・天気予報
3 #include <M5EPD.h>
4
5 #define WeatherFileName "/weather.jsn"
6
7 class GetInfoFromNetwork {
8     private:
9
10         bool wifiOn(void);
11         void wifiOff(void);
12     public:
13         GetInfoFromNetwork();
14         ~GetInfoFromNetwork() ;
15         int isWiFiOn(void);
16         int setNtpTime() ;
17         String getWeatherQuery() ;
18         bool getWeatherInfo() ;
19 };
```

4.2 wifiid.h

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

4.3 apikey.h

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

4.4 infoFromNet.cpp

```
1 #include <M5EPD.h>
2 #include <WiFi.h>
3 #include <HTTPClient.h>
4 #include "infoFromNet.hpp"
5 #include <time.h>
6
7 #include "util.h"
8
9 // 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。
10 // このファイルは、.gitignoreとする。
11 #include "wifiid.h"
12 // apikey.hには、apikeyのdefineの定義を文字列として記載すること。
13 // このファイルは、.gitignoreとする。
14 #include "apikey.h"
15
```

```

16 GetInfoFromNetwork::GetInfoFromNetwork() {
17     wifiOn();
18 }
19
20 GetInfoFromNetwork::~GetInfoFromNetwork() {
21     wifiOff();
22 }
23
24 bool GetInfoFromNetwork::wifiOn(void) {
25     WiFi.begin(ssid, password);
26     for (int i = 0 ; i < 10; i++) {
27         if (isWiFiOn()) return true;
28         delay(500);
29     }
30     return false;
31 }
32
33 void GetInfoFromNetwork::wifiOff(void) {
34     WiFi.disconnect(true);
35     WiFi.mode(WIFI_OFF);
36 }
37
38 int GetInfoFromNetwork::isWiFiOn(void) {
39     return (WiFi.status() == WL_CONNECTED) ;
40 }
41
42 int GetInfoFromNetwork::setNtpTime() {
43     if (!isWiFiOn()) return -1;
44     const long gmtOffset_sec = 9 * 3600;
45     const int daylightOffset_sec = 0;
46     const char * ntpServer = "jp.pool.ntp.org";
47
48     configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
49     struct tm timeinfo;
50     if (!getLocalTime(&timeinfo)) return -1;
51
52     rtc_time_t rtcTime;
53     rtcTime.hour = (int8_t)timeinfo.tm_hour;
54     rtcTime.min = (int8_t)timeinfo.tm_min;
55     rtcTime.sec = (int8_t)timeinfo.tm_sec ;
56     rtc_date_t rtcDate ;
57     rtcDate.year = (int8_t)timeinfo.tm_year + 1900;
58     rtcDate.mon = (int8_t)timeinfo.tm_mon + 1;
59     rtcDate.day = (int8_t)timeinfo.tm_mday ;
60     M5.RTC.setDate(&rtcDate);
61     M5.RTC.setTime(&rtcTime);
62     return 0;
63 }
64
65
66 const uint8_t fingerprint[20] =
67     { 0xEE,0xAA,0x58,0x6D,0x4F,0x1F,0x42,0xF4,0x18,0x5B,0x7F,0xB0,0xF2,0x0A,0x4C,0xDD,0x97,0
        x47,0x7D,0x99 };
68 #define OpenWeatherUrl "api.openweathermap.org"
69 #define City "Nagahama,JP"
70
71 String GetInfoFromNetwork::getWeatherQuery() {
72     String url("/data/2.5/forecast?");

```

```

73     url += "q=" City;
74     url += "&appid=" apikey;
75     url += "&lang=ja&units=metric";
76     return url;
77 }
78
79 // 天気予報データをSDカードのWeatherFileNameに書き込む。
80 bool GetInfoFromNetwork::getWeatherInfo() {
81     if (!isWiFiOn()) return false;
82     HTTPClient http;
83     File file;
84     String url = String("http://") + String(OpenWeatherUrl) + getWeatherQuery();
85     Serial.print("URL:");
86     Serial.println(url);
87     if (!http.begin(url)) return false;
88     int retCode = http.GET();
89     if (retCode < 0) goto http_err;
90     if (retCode != HTTP_CODE_OK && retCode != HTTP_CODE_MOVED_PERMANENTLY) goto http_err;
91     if (!SD.exists("/")) goto http_err;
92     Serial.println("SD OK!");
93     if (SD.exists(WeatherFileName)) SD.remove(WeatherFileName);
94     file=SD.open(WeatherFileName, FILE_WRITE);
95     if (!file) goto http_err;
96     Serial.println("ファイルオープン完了");
97     if (http.writeToStream(&file) < 0) goto file_err;
98     file.close();
99     Serial.println("weatherファイルへのjsonデータ書き込み完了");
100    http.end();
101    return true;
102
103 file_err:
104     file.close();
105 http_err:
106     http.end();
107     return false;
108 }

```

5 時計の表示

5.1 tokei.hpp

```
1  /*****
2   * 時計の表示
3   *****/
4
5  #include <M5EPD.h>
6  #define LGFX_M5PAPER
7  #include <LovyanGFX.hpp>
8
9  class Tokei {
10     private:
11         int year, month, day;
12         int hour, min, sec;
13         int dayOfTheWeek;
14         int width, height;
15
16         void getDateTime();
17         int getDayOfTheWeek(int year, int month, int day) ;
18
19     public:
20         Tokei(int sizex=200, int sizey=200);
21         void drawDigitalTokei(LovyanGFX *lcd, int x, int y);
22 };
```

5.2 tokei.cpp

```
1  /*****
2   * 時計の表示
3   *****/
4
5  #include <M5EPD.h>
6  #define LGFX_M5PAPER
7  #include <LovyanGFX.hpp>
8
9  #include "tokei.hpp"
10
11 Tokei::Tokei(int width, int height)
12     : width(width), height(height) {
13     getDateTime();
14     dayOfTheWeek = getDayOfTheWeek(year, month, day);
15 }
16
17 // RTCより現在時刻を取得する。
18 void Tokei::getTime() {
19     rtc_time_t time;
20     rtc_date_t date;
21
22     M5.RTC.getTime(&time);
23     M5.RTC.getDate(&date);
24
25     year = date.year;
26     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(&font::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(&font::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

```
1 // 天気予報のJSONの解析及び値の提供
2
3 #include <M5EPD.h>
4 #include <ArduinoJson.h>
5
6 #ifndef TENKI
7 #define TENKI
8 struct Allocator {
9     void* allocate(size_t size) { return ps_malloc(size); }
10    void deallocate(void *ptr) { heap_caps_free(ptr); }
11    void* reallocate(void* ptr, size_t new_size) { return ps_realloc(ptr, new_size); }
12 };
13
14 class Tenki {
15     private:
16         //DynamicJsonDocument json;
17         BasicJsonDocument<Allocator> json;
18         DeserializationError jsonErr;
19         bool _isEnabled;
20
21     public:
22         Tenki();
23         // 情報が有効か否かを返す。
24         bool isEnabled() { return _isEnabled; }
25         // リストナンバーの最大値を返す。
26         int getMaxListNo() { return json["list"].size()-1; };
27         // 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
28         int getListIdAfterHHour(int h) ;
29         // 指定リストナンバーの時刻を取得する。
30         time_t getDate(int i) {
31             time_t time = json["list"][i]["dt"].as<time_t>() + 9*3600;
32             return time;
33         };
34         // 指定リストナンバーの天気を取得する。
35         const char *getWeather(int i) {
36             return json["list"][i]["weather"][0]["description"].as<const char *>();
37         };
38         // 指定リストナンバーの気温を取得する。
39         float getTemp(int i) { return json["list"][i]["main"]["temp"].as<float>(); };
40         // 指定リストナンバーの風向を取得する
41         const char *getWindDir(int i) ;
42         // 指定リストナンバーの風速を取得する。
43         float getWindSpeed(int i) { return json["list"][i]["wind"]["speed"].as<float>(); };
44         // 指定リストナンバーの風力階級を取得する
45         int getBeaufortScale(int i) ;
46         // 天気情報を最新のSDカードデータに更新する。
47         void refresh() { _isEnabled=false; readJson(); };
48
49     private:
50         void readJson() ;
51
52 };
```

```
53 #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), _isEnabled(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) _isEnabled=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 // 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
44 int Tenki::getListIdAfterHHour(int h) {
45     time_t searchTime = now() + h * 3600;
46
47     int listNo = 0;
48     int max = getMaxListNo();
49     while (listNo <= max && getDate(listNo) < searchTime) { listNo++; }
50     return listNo;
51 }
```


7 天気予報の表示

7.1 drawtenki.hpp

```
1 // 天気予報の表示
2 #include <M5EPD.h>
3 #define LGFX_M5PAPER
4 #include <LovyanGFX.hpp>
5
6 #include "tenki.hpp"
7
8 class DrawTenki {
9     private:
10         Tenki *tenki;
11         LGFX_Sprite screen;
12         int width;
13         int height;
14         float columnX[4];
15         float rowCenterY[3];
16
17     public:
18         DrawTenki(Tenki *tenki, int width, int height);
19         void draw(LovyanGFX *lcd, int x, int y) ;
20
21     private:
22         void drawFrameBorder() ;
23         void calcColumnCoordinate();
24         void drawTenkiInfo(int lineNo) ;
25
26 };
```

7.2 drawtenki.cpp

```
1 // 天気予報の表示
2 #include <M5EPD.h>
3 #define LGFX_M5PAPER
4 #include <LovyanGFX.hpp>
5 #include <time.h>
6
7 #include "drawtenki.hpp"
8
9 DrawTenki::DrawTenki(Tenki *tenki, int width, int height)
10 : tenki(tenki), width(width), height(height) {
11     screen.setColorDepth(4);
12     screen.setPsrAm(true);
13     screen.createSprite(width, height);
14     screen.fillSprite(15);
15     screen.setColor(0);
16     screen.setTextColor(0,15);
17     screen.setFont(&font::lgfxJapanGothic_36);
18     float textsize = (((float)height-10.f) / 3.f) / 36.f;
19     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°C") + 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。 lineneno:0-2
48 void DrawTenki::drawTenkiInfo(int lineNo) {
49     const int hour[3] = { 12, 24, 48 };
50     screen.setTextDatum(middle_left);
51     int listNo = tenki->getListIdAfterHHour(hour[lineNo]);
52
53     //時間
54     time_t dataTimet = tenki->getDate(listNo);
55     struct tm *dataTm = gmtime(&dataTimet);
56     char textbuf[100];
57     sprintf(textbuf, "%2d日%2d時", dataTm->tm_mday, dataTm->tm_hour);
58     screen.drawString(textbuf, columnX[0], rowCenterY[lineNo]);
59     //天気
60     screen.drawString(tenki->getWeather(listNo), columnX[1], rowCenterY[lineNo]);
61     //気温
62     sprintf(textbuf, "%2d°C", (int)tenki->getTemp(listNo));
63     screen.drawString(textbuf, columnX[2], rowCenterY[lineNo]);
64     //風向、風力
65     const char *windDir = tenki->getWindDir(listNo);
66     int beaufortScale = tenki->getBeaufortScale(listNo);
67     sprintf(textbuf, "%s %d", windDir, beaufortScale);
68     screen.drawString(textbuf, columnX[3], rowCenterY[lineNo]);
69 }
70
71 void DrawTenki::draw(LovyanGFX *lcd, int x, int y) {
72     drawFrameBorder();
73     for (int i=0; i < 3 ; i++) {
74         drawTenkiInfo(i);
75     }
76     screen.pushSprite(lcd, x, y);
77 }

```

8 SD カード内の jpeg ファイルをスキャン

8.1 scanfile.hpp

```
1 // Jpegファイルの一覧を生成する。
2
3 #define blockSize 10
4 #define filenameSize 14 // 1+8+1+3+1(8.3形式に限る。 '/'が付加される)
5
6 struct FileNames {
7     char filenames[blockSize][filenameSize];
8     FileNames *prev;
9     FileNames *next;
10 };
11
12 class JpegFiles {
13     private:
14         FileNames *top;
15         FileNames *cur;
16         int _count;
17     public:
18         JpegFiles();
19         char *operator[](int i) ;
20         int count() { return _count; };
21
22
23     private:
24         FileNames *addBlock(FileNames *last) ;
25         void addFilename(const char*filename) ;
26         bool isJpegFile(const char* filename) ;
27
28
29 };
```

8.2 scanfile.cpp

```
1 // Jpegファイルの一覧
2
3 #include <M5EPD.h>
4 #include <SD.h>
5
6 #include "scanfile.hpp"
7
8 JpegFiles::JpegFiles() : top(NULL), cur(NULL), _count(0) {
9     if (!SD.exists("/")) return ;
10     File root = SD.open("/");
11     File entry = root.openNextFile();
12     while (entry) {
13         if (!entry.isDirectory()) {
14             if (isJpegFile(entry.name())) {
15                 addFilename(entry.name());
16             }
17         }
18         entry = root.openNextFile();
19     }
```

```

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                 {
67                     check = 1;
68                 } else {
69                     check = 0;
70                 }
71         }
72     }
73     return extBig[check] == '\0';
74 }

```

9 ユーティリティー

9.1 util.h

```
1 // ユーティリティー関数
2
3 #include "time.h"
4
5 #ifndef ENVIRONMENT_UTIL
6 #define ENVIRONMENT_UTIL
7 time_t now();
8
9 #endif
```

9.2 util.cpp

```
1 // ユーティリティー関数
2 #include <M5EPD.h>
3 #include <time.h>
4
5 time_t now() {
6     struct tm time;
7     rtc_time_t rtcTime;
8     rtc_date_t rtcDate ;
9
10    M5.RTC.getDate(&rtcDate);
11    M5.RTC.getTime(&rtcTime);
12
13    time.tm_year = rtcDate.year - 1900;
14    time.tm_mon = rtcDate.mon - 1;
15    time.tm_mday = rtcDate.day;
16    time.tm_hour = rtcTime.hour;
17    time.tm_min = rtcTime.min;
18    time.tm_sec = rtcTime.sec;
19
20    return mktime(&time);
21 }
```