

M5Paper 環境モニターソース

美都

2021 年 3 月 21 日

目次

1	メイン	2
2	バッテリーメーター	4
2.1	battery.h	4
2.2	battery.cpp	4
3	温湿度計	6
3.1	thermometer.hpp	6
3.2	thermometer.cpp	6
4	ネットからの情報取得	9
4.1	infoFromNet.hpp	9
4.2	wifid.h	9
4.3	apikey.h	9
4.4	infoFromNet.cpp	9
5	時計の表示	12
5.1	tokei.hpp	12
5.2	tokei.cpp	12
6	天気予報データの管理	14
6.1	tenki.hpp	14
6.2	tenki.cpp	14
7	天気予報の表示	16
7.1	drawtenki.hpp	16
7.2	drawtenki.cpp	16
8	ユーティリティー	18
8.1	util.h	18
8.2	util.cpp	18

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
13 static LGFX lcd;
14
15 void drawLcd(Tenki *tenki) {
16     drawBattery(960-120-5, 5, &lcd);
17     Thermometer t = Thermometer(200,200);
18     t.drawTempMeter(&lcd, 500, 100);
19     t.drawHumMeter(&lcd, 700, 100);
20
21     Tokei tokei = Tokei(300, 100);
22     tokei.drawDigitalTokei(&lcd, 100, 100);
23
24     DrawTenki drawTenki(tenki, 500, 200);
25     drawTenki.draw(&lcd, 950-500-5 , 330);
26
27     delay(500);
28 }
29
30 // ●分ピッタリまでの秒数
31 int rest_minute() {
32     rtc_time_t time;
33     M5.RTC.getTime(&time);
34     return 60-time.sec;
35 }
36
37 // シャットダウンを試みる。通電中はすり抜ける
38 void challengeShutdown() {
39     int rest_sec = rest_minute()-6;
40     if (rest_sec < 30) rest_sec += 60;
41     M5.shutdown(rest_sec); // 一旦停止
42 }
43
44 void checkInfoFromNetwork(Tenki *tenki, bool always=false) {
45     rtc_time_t time;
46     M5.RTC.getTime(&time);
47     time_t outdated = now() - 6*3600;
48
49     if (!tenki->isEnabled() || (time.hour%6==0 && time.min == 3) || tenki->getDate(0)<outdated)
50     {
51         Serial.println("ネットワークの情報の取得開始");
52         GetInfoFromNetwork info;
53         info.setNtpTime();
54         info.getWeatherInfo();
```

```

54         tenki->refresh();
55     }
56 }
57
58 void setup()
59 {
60     M5.begin(false, true, true, true, true);
61     M5.BatteryADCBegin();
62     M5.RTC.begin();
63     M5.SHT30.Begin();
64     SD.begin();
65     Serial.begin(115200);
66     lcd.init();
67     lcd.setRotation(1);
68
69     Tenki tenki;
70     checkInfoFromNetwork(&tenki);
71
72     drawLcd(&tenki);
73     challengeShutdown();
74 }
75
76
77 void loop()
78 {
79     delay((rest_minute()+1)*1000);
80     Tenki tenki;
81     checkInfoFromNetwork(&tenki);
82     drawLcd(&tenki);
83     challengeShutdown();
84 }

```

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].createSprite(radius/25, radius/5);
14     scale[0].fillSprite(0);
15     scale[0].setPivot(scale[0].width()/2, scale[0].height());
16     scale[1].setColorDepth(4);
17     scale[1].createSprite(radius/40, radius/7);
18     scale[1].fillSprite(0);
19     scale[1].setPivot(scale[1].width()/2, scale[1].height());
20 }
```

```

21
22 void Thermometer::makeHand() {
23     float height, width;
24     height = radius * 0.8f;
25     width = height * 0.1f;
26
27     hand.setColorDepth(4);
28     hand.createSprite(width, height);
29     hand.fillSprite(15);
30     hand.setColor(0);
31     hand.fillTriangle(width/2.f, 0, 0, height/4.f, width, height/4.f);
32     hand.fillTriangle(0, height/4.f, width, height/4.f, width/2.f, height);
33     hand.setPivot(width/2., height);
34 }
35
36 void Thermometer::makeMeterFace(int min, int max, const char* unit) {
37     face.setColorDepth(4);
38     face.createSprite(size_x, size_y);
39     face.fillSprite(15);
40     face.setColor(0);
41     face.setFont(&fonts::lgfxJapanGothic_36);
42     face.setTextColor(0, 15);
43     face.setTextDatum(middle_center);
44     float center[2] = {size_x/2.0f, size_y/2.0f};
45     face.fillCircle(center[0], center[1], radius);
46     face.fillCircle(center[0], center[1], radius*0.95, 15);
47     float angleInterval = 270.f / (float)(max-min);
48     for (int i = min ; i <= max ; i+=2) {
49         LGFX_Sprite *use_scale = (i%10==0) ? &scale[0] : &scale[1];
50         float angle = (270.f-45.f) - (float)(i-min) * angleInterval;
51         float angleRad = angle * 3.14159265f / 180.f ;
52         float startx = (radius - use_scale->height()) * cos(angleRad) + center[0];
53         float starty = -1.0f * ((radius - use_scale->height()) * sin(angleRad)) + center[1];
54         use_scale->pushRotateZoom(&face, startx, starty, 90.f-angle, 1.f, 1.f);
55         if (i%10==0) {
56             float charsize = (float)scale[0].height() / 36.f;
57             float charx = (radius - use_scale->height() * 1.5f) * cos(angleRad) + center[0];
58             float chary = -1.f * (radius - use_scale->height() * 1.5f) * sin(angleRad) +
                    center[0];
59             face.setTextSize(charsize);
60             face.drawNumber(i, charx, chary);
61         }
62         face.drawString(unit, center[0], size_y/5.f*3.f);
63     }
64 }
65
66
67 void Thermometer::drawTempMeter(LovyanGFX *lcd, int x, int y) {
68     makeMeterFace(0, 50, "°C");
69     float center[2] = {(float)size_x/2.f, (float)size_y/2.f};
70     float angle = 270.f - 45.f;
71     angle -= 270.f / 50.f * get_temp();
72     hand.pushRotateZoom(&face, center[0], center[1], 90.f - angle, 1.f, 1.f);
73     face.pushSprite(lcd, x, y);
74 }
75
76 void Thermometer::drawHumMeter(LovyanGFX *lcd, int x, int y) {
77     makeMeterFace(20, 80, "%");

```

```

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

```


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 <HTTIClient.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.setColorDepth(4);
48     tokei.createSprite(width,height);
49     tokei.fillSprite(15);
50     //tokei.drawRect(0,0,width,height,0); // レイアウト検討用外枠
51     // 時計時間部分表示
52     char strTime[6];
53     sprintf(strTime, "%02d:%02d", hour, min);
54     tokei.setFont(&fonts::Font7); // font高さ:48
55     tokei.setTextColor(0,15);
56     float mag = (height*tokei_ratio) / 48.f;
57     tokei.setTextSize(mag, mag);
58     tokei.drawString(strTime, 0.f, height*(1-tokei_ratio));
59     // 時計日時部分表示
60     const char* youbi_tbl[] =
61         { "月", "火", "水", "木", "金", "土", "日" };
62     const char* youbi = youbi_tbl[dayOfTheWeek];
63     char strDate[30];
64     sprintf(strDate, "%d年%2d月%2d日(%s)", year, month, day, youbi);
65     tokei.setFont(&fonts::lgfxJapanGothic_36);
66     mag = (height*(1-tokei_ratio)) / 36.f;
67     tokei.setTextSize(mag, mag);
68     tokei.drawString(strDate, 0.f, 0.f);
69     tokei.pushSprite(lcd, x, y);
70 }

```

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 class Tenki {
9     private:
10         DynamicJsonDocument json;
11         DeserializationError jsonErr;
12         bool _isEnabled;
13
14
15     public:
16         Tenki();
17         // 情報が有効か否かを返す。
18         bool isEnabled() { return _isEnabled; }
19         // リストナンバーの最大値を返す。
20         int getMaxListNo() { return json["list"].size()-1; };
21         // 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
22         int getListIdAfterHHour(int h) ;
23         // 指定リストナンバーの時刻を取得する。
24         time_t getDate(int i) {
25             time_t time = json["list"][i]["dt"].as<time_t>() + 9*3600;
26             return time;
27         };
28         // 指定リストナンバーの天気を取得する。
29         const char *getWeather(int i) {
30             return json["list"][i]["weather"][0]["description"].as<const char *>();
31         };
32         // 指定リストナンバーの気温を取得する。
33         float getTemp(int i) { return json["list"][i]["main"]["temp"].as<float>(); };
34         // 指定リストナンバーの風向を取得する
35         const char *getWindDir(int i) ;
36         // 指定リストナンバーの風速を取得する。
37         float getWindSpeed(int i) { return json["list"][i]["wind"]["speed"].as<float>(); };
38         // 指定リストナンバーの風力階級を取得する
39         int getBeaufortScale(int i) ;
40         // 天気情報を最新のSDカードデータに更新する。
41         void refresh() { _isEnabled=false; readJson(); };
42
43     private:
44         void readJson() ;
45
46 };
47 #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         int width;
12         int height;
13
14     public:
15         DrawTenki(Tenki *tenki, int width, int height);
16         void draw(LovyanGFX *lcd, int x, int y) ;
17
18     private:
19
20 };
```

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
12 }
13
14 void DrawTenki::draw(LovyanGFX *lcd, int x, int y) {
15     LGFX_Sprite sprite;
16     sprite.setColorDepth(4);
17     sprite.createSprite(width, height);
18     sprite.fillSprite(15);
19     sprite.setTextColor(0, 15);
20     sprite.setColor(0);
21     sprite.setFont(&fonts::lgfxJapanGothic_36);
22     float textsize = (height / 4) / 36;
23     sprite.drawRect(0, 0, width, height);
24     for (int i=1; i <= 3 ; i++) {
25         sprite.drawLine(0, height/4*i, width, height/4*i);
26     }
27     sprite.setTextSize(textsize);
28     for (int i=0; i < 1 ; i++) {
```



```

29     int top = (height / 3) * i + 1;
30     int listNo = 4;
31     time_t dataTimet = tenki->getDate(listNo);
32     struct tm *dataTm = gmtime(&dataTimet);
33     char textbuf[100];
34     sprintf(textbuf, "%2d日%2d時", dataTm->tm_mday, dataTm->tm_hour);
35     int x = 1;
36     sprite.drawString(textbuf, x, top);
37     x += sprite.textWidth(textbuf) + 5;
38     sprite.drawString(tenki->getWeather(listNo), x, top);
39     x += sprite.textWidth("激しい雨") + 5;
40     sprintf(textbuf, "%2d℃", (int)tenki->getTemp(listNo));
41     sprite.drawString(textbuf, x, top);
42     x += sprite.textWidth("00℃") + 5;
43     sprite.drawString(tenki->getWindDir(listNo), x, top);
44     x += sprite.textWidth("南東") + 5;
45     sprite.drawNumber(tenki->getBeaufortScale(listNo), x, top);
46 }
47 sprite.pushSprite(lcd, x, y);
48 }

```

8 ユーティリティー

8.1 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 #endif
```

8.2 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 }
```