```
/*
 * LteCamera02.ino
 * LTE通信でHDRカメラ画像を定期的にDropboxにuploadする
 *
 * 常にHDR ONで写真撮影する
 *
 * バックアップとしてSDにも10分ごとに写真を記録
 * 日時でファイル名作成
 * 日付でサブディレクトリ作成してその中に記録
 * 露光調整・ホワイトバランス・ISO感度・JPEG品質設定追加
 *
 * Dropboxに写真upload
 * SDと同じファイル名・サブディレクトリにuploadする
 *
 * 必要なLibrary ArduinoHttpClient
 *
 * 注意
 * Memory 1536KBにすること　jpeg画像データ用に大量のメモリ使う
 *
 */
#include <SDHCI.h>
#include <stdio.h>  /* for sprintf */
#include <RTC.h>
#include <LTE.h>
#include <ArduinoHttpClient.h>
#include <Watchdog.h>
#include <Camera.h>

// 昼間の開始時間と終了時間　ISO感度自動にする時間
#define DAY_START_HOUR 6 // 開始時間 6:00
#define DAY_END_HOUR 18  // 終了時間 18:00
// Jpeg Quality
#define JPEG_QUALITY  80
// JPEG画像バッファのスライス数　値を小さくするとJPEG画像バッファが大きくなり高画質で撮影が可能
#define  JPEGBUFSIZE_DIVISOR  3

// APN name
#define APP_LTE_APN "so-net.jp" // replace your APN

/* APN authentication settings
 * Ignore these parameters when setting LTE_NET_AUTHTYPE_NONE.
 */
#define APP_LTE_USER_NAME "nuro" // SIMのusername
#define APP_LTE_PASSWORD  "nuro" // SIMのpassword

// APN IP type
#define APP_LTE_IP_TYPE (LTE_NET_IPTYPE_V4V6) // IP : IPv4v6

// APN authentication type
#define APP_LTE_AUTH_TYPE (LTE_NET_AUTHTYPE_CHAP) // Authentication : CHAP

/* RAT to use
 * Refer to the cellular carriers information
 * to find out which RAT your SIM supports.
 * The RAT set on the modem can be checked with LTEModemVerification::getRAT().
 */
#define APP_LTE_RAT (LTE_NET_RAT_CATM) // RAT : LTE-M (LTE Cat-M1)

// URL, path & port (for example: httpbin.org)
char server_content[] = "content.dropboxapi.com";
char server_api[] = "api.dropbox.com";
int port = 443; // port 443 is the default for HTTPS

// Dropbox APIのルート証明ファイル名
// content.dropboxapi.com
#define ROOTCA_FILE_CONT "CERTS/cont.dropboxapi_rootCA.crt"
// api.dropbox.com
#define ROOTCA_FILE_API "CERTS/api.dropbox_rootCA.crt"

// HTTP POSTで一度に送るバイト数max
#define POST_SIZE 1024

// 以下3つの情報はDropboxの短期アクセストークン(short-lived access token)を取得するのに使う
#define REFRESH_TOKEN "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"              // Dropbox リフレッシュトークン 永続的に使えるトークン
#define APP_KEY "xxxxxxxxxxx"  // アプリキー
#define APP_SECRET  "xxxxxxxxxxxxxxx" // アプリシークレット
// WatchDogのタイムアウト 最大40sec 単位msec
#define WATCHDOG_TIMEOUT        (40*1000)

// USB Console
#define BAUDRATE                (115200)

/*
 * 撮影動作スケージュル設定
 * 通信データ量削減のために夜間は撮影間隔を長くする
 * 夜間はISO感度を上げて露光時間を長くする　目に見える効果は上がらず
 */
struct sched_t {
        int period;     // 撮影間隔 sec
        int iso;        // ISO感度 0=AUTO CAM_ISO_SENSITIVITY_1600
        int exposure;   // 露光時間 0=AUTO 100usec単位
```

```
};
// 露光時間1secの設定値 100usec単位
#define EXPOSURE_1SEC   10000
struct sched_t schedule[24] = {
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 0H
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 1H
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 2H
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 3H
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 4H
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 5H
        {  60, 0, 0 }, // 6H この時間から毎分撮影 ISO自動 露光時間自動
        {  60, 0, 0 }, // 7H
        {  60, 0, 0 }, // 8H
        {  60, 0, 0 }, // 9H
        {  60, 0, 0 }, // 10H
        { 180, 0, 0 }, // 11H この時間から3分毎撮影 ISO自動 露光時間自動
        { 180, 0, 0 }, // 12H
        { 180, 0, 0 }, // 13H
        {  60, 0, 0 }, // 14H この時間から毎分撮影 ISO自動 露光時間自動
        {  60, 0, 0 }, // 15H
        {  60, 0, 0 }, // 16H
        { 180, 0, 0 }, // 17H
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 18H この時間から10分毎撮影 ISO1600 露光時間1秒
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 19H
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 20H
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 21H
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 22H
        { 600, CAM_ISO_SENSITIVITY_1600, EXPOSURE_1SEC }, // 23H
};
// Dropbox API用 短期アクセストークン を保存する変数
String access_token;

// LTEモジュールライブラリのインスタンス定義
LTE lteAccess;
// LTE TLSクライントライブラリ２つのインスタンス定義
// content.dropboxapi.com アクセス用
LTETLSClient tlsClient;
HttpClient client = HttpClient(tlsClient, server_content, port);
// api.dropbox.com アクセス用
LTETLSClient tlsClient2;
HttpClient client2 = HttpClient(tlsClient2, server_api, port);
// SDライブラリのインスタンス定義
SDClass  theSD;

// LED ON/OFF関数定義
#define led0_on() digitalWrite(LED0, HIGH)
#define led0_off() digitalWrite(LED0, LOW)
#define led1_on() digitalWrite(LED1, HIGH)
#define led1_off() digitalWrite(LED1, LOW)
#define led2_on() digitalWrite(LED2, HIGH)
#define led2_off() digitalWrite(LED2, LOW)
#define led3_on() digitalWrite(LED3, HIGH)
#define led3_off() digitalWrite(LED3, LOW)
void led0_blink()
{
  digitalWrite(LED0, LOW);
  delay(500);
  digitalWrite(LED0, HIGH);
}
void led1_blink()
{
  digitalWrite(LED1, LOW);
  delay(500);
  digitalWrite(LED1, HIGH);
}

/**
 * カメラのエラーメッセージ出力
 */
void printError(enum CamErr err)
{
  Serial.print("Error: ");
  switch (err) {
      case CAM_ERR_NO_DEVICE:
        Serial.println("No Device");
        break;
      case CAM_ERR_ILLEGAL_DEVERR:
        Serial.println("Illegal device error");
        break;
      case CAM_ERR_ALREADY_INITIALIZED:
        Serial.println("Already initialized");
        break;
      case CAM_ERR_NOT_INITIALIZED:
        Serial.println("Not initialized");
        break;
      case CAM_ERR_NOT_STILL_INITIALIZED:
        Serial.println("Still picture not initialized");
        break;
      case CAM_ERR_CANT_CREATE_THREAD:
        Serial.println("Failed to create thread");
        break;
```

```
      case CAM_ERR_INVALID_PARAM:
        Serial.println("Invalid parameter");
        break;
      case CAM_ERR_NO_MEMORY:
        Serial.println("No memory");
        break;
      case CAM_ERR_USR_INUSED:
        Serial.println("Buffer already in use");
        break;
      case CAM_ERR_NOT_PERMITTED:
        Serial.println("Operation not permitted");
        break;
      default:
        break;
    }
}
void printClock(RtcTime &rtc)
{
  printf("%04d/%02d/%02d %02d:%02d:%02d¥n", rtc.year(), rtc.month(), rtc.day(),
    rtc.hour(), rtc.minute(), rtc.second());
}
/**
 * システムリセットをWatchDogで行う
 */
static void system_reset()
{
  led0_on();
  led1_on();
  led2_on();
  led3_on();
  sleep(3);
  Watchdog.start(100);
  while(1) delay(1000);
}
/*
 * 先に定義したREFRESH_TOKEN APP_KEY APP_SECRET を使い,
 * Dropboxの短期アクセストークンを取得する
 */
boolean dropbox_access_token_get()
{
  int statusCode;
  String response;
  // HTTP POST実行
  Serial.println("Dropbox access token request");
  String contentType = "application/x-www-form-urlencoded";
  String postData = "refresh_token=" + String(REFRESH_TOKEN) ¥  // リフレッシュトークン
    + "&grant_type=refresh_token&client_id=" + String(APP_KEY) ¥  // アプリキー
    + "&client_secret=" + String(APP_SECRET); // アプリシークレット
  Serial.println(postData);
  // Dropbox API oauth2 token
  client2.post("/oauth2/token", contentType, postData);

  // ステータスとレスポンスを取得
  statusCode = client2.responseStatusCode();
  response = client2.responseBody();
  Serial.print("Status code: ");
  Serial.println(statusCode);
  Serial.print("Response: ");
  Serial.println(response);
  // レスポンスからaccess tokenを抽出
  int i = response.indexOf(":");
  if (i < 0) return false;
  access_token = response.substring(i + 3);
  i = access_token.indexOf("¥"");
  if (i < 0) return false;

  // access tokenを取得出来たので保存する
  access_token = access_token.substring(0, i);
  Serial.println("Access token: " + access_token);

  return true;
}
/*
 * Dropbox APIのHTTP POSTを行い
 * 撮影画像をDropboxにアップロードする
 * 引数のfile_pathをフルパスで指定すればサブディレクトリが無い場合でも自動的に作られる
 */
boolean dropbox_upload(uint8_t *img_buff, int img_len, char* file_path)
{
  int statusCode;
  String response;
  // HTTP POSTのヘッダー作成
  Serial.println("making POST request");
  client.beginRequest();
  // Dropbox API files upload
  if (client.post("/2/files/upload") != 0) {
    return false;
  }
  client.sendHeader("Authorization: Bearer " + access_token);  // 短期アクセストークン設定
  client.sendHeader("Dropbox-API-Arg: {¥"autorename¥":false,¥"mode¥":¥"overwrite¥",¥"mute¥":false,¥"path¥":¥"/" +
String(file_path) + "¥",¥"strict_conflict¥":false}"); // ファイルパス設定
  client.sendHeader("Content-Type: application/octet-stream"); // コンテンツとしてバイナリを指定
```

```
client.sendHeader("Content-Length: " + String(img_len)); // コンテンツの長さを正しく設定する必要あり
// BODY送信を開始
client.beginBody();
// 画像データ(バイナリ)を分割して送る
const uint8_t *post_ptr = img_buff;
while(img_len > 0) {
  int post_len;
  if (img_len >= POST_SIZE) {
    post_len = POST_SIZE;
    img_len -= POST_SIZE;
  } else {
    post_len = img_len;
    img_len = 0;
  }
  // 送信には35秒程度かかるのでWatchDogをクリアしながら行う
  Watchdog.kick();
  // 分割されたデータを送信
  client.write(post_ptr, post_len);
  post_ptr += post_len;
}
// BODY送信終了
client.endRequest();
Serial.print("POST END");
// WatchDog
Watchdog.kick();

// ステータスとレスポンスを取得
statusCode = client.responseStatusCode();
response = client.responseBody();
Serial.print("Status code: ");
  Serial.println(statusCode);
  Serial.print("Response: ");
  Serial.println(response);
  return true;
}
/*
 * microsSDに撮影画像を保存する
 */
boolean file_save(uint8_t *img_buff, int img_len, char *dirname, char *file_path)
{
  // ディレクトリ作成
  if (!theSD.exists(dirname)) {
    if (!theSD.mkdir(dirname)) {
      Serial.println("SD.mkdir() ERROR");
      return false;
    }
  }
  // SDに同一ファイル名があれば削除
  theSD.remove(file_path);
  // 画像をファイル記録
  File myFile = theSD.open(file_path, FILE_WRITE);
  if (myFile.write(img_buff, img_len) != (size_t)img_len) {
    myFile.close();
    return false;
  }
  myFile.close();
  return true;
}

/**
 * 各種初期化
 */
void setup()
{
  char apn[LTE_NET_APN_MAXLEN] = APP_LTE_APN;
  LTENetworkAuthType authtype = APP_LTE_AUTH_TYPE;
  char user_name[LTE_NET_USER_MAXLEN] = APP_LTE_USER_NAME;
  char password[LTE_NET_PASSWORD_MAXLEN] = APP_LTE_PASSWORD;
  CamErr err;
  int err_cnt = 0;

  /* Open serial communications and wait for port to open */
  Serial.begin(BAUDRATE);
  while (!Serial) {
    ; /* wait for serial port to connect. Needed for native USB port only */
  }
  // LED初期化
  pinMode(LED0, OUTPUT);
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  led0_on();
  led1_off();
  led2_off();
  led3_off();
  //
  Serial.println("");
  Serial.println("Starting LTE HDR Camera system.");
  Serial.println("=========== APN information ===========");
  Serial.print("Access Point Name  : ");
  Serial.println(apn);
  Serial.print("Authentication Type: ");
```

```
Serial.println(authtype == LTE_NET_AUTHTYPE_CHAP ? "CHAP" :
               authtype == LTE_NET_AUTHTYPE_NONE ? "NONE" : "PAP");
// 撮影スケジュール表示
Serial.println("Schedule");
Serial.println("Hour Period ISO Exposure");
for(int i=0; i<24; i++) {
      printf("%2d %3d %d %d¥r¥n", i, schedule[i].period, schedule[i].iso, schedule[i].exposure);
}
/*
 * Watchdog 初期化と開始
 * timeout 最大40sec
 */
Serial.println("Watchdog start");
Watchdog.begin();
Watchdog.start(WATCHDOG_TIMEOUT);
/* SD初期化 */
while (!theSD.begin()) {
  // SDカードがマウントされるまで待つ
  Serial.println("Insert SD card.");
}
/*
 * LTEモデム初期化リトライループ
 */
while (true) {
  /* モデム電源ONとRF機能有効化 */
  if (lteAccess.begin() != LTE_SEARCHING) {
    Serial.println("Could not transition to LTE_SEARCHING.");
    Serial.println("Please check the status of the LTE board.");
    for (;;) {
      sleep(1);
    }
  }
  /* APNへの接続開始
   * The connection process to the APN will start.
   * If the synchronous parameter is false,
   * the return value will be returned when the connection process is started.
   */
  if (lteAccess.attach(APP_LTE_RAT,
                       apn,
                       user_name,
                       password,
                       authtype,
                       APP_LTE_IP_TYPE) == LTE_READY) {
    Serial.println("attach succeeded.");
    break;
  }
  // WatchDog
  Watchdog.kick();
  /* If the following logs occur frequently, one of the following might be a cause:
   * - APN settings are incorrect
   * - SIM is not inserted correctly
   * - If you have specified LTE_NET_RAT_NBIOT for APP_LTE_RAT,
   *   your LTE board may not support it.
   * - Rejected from LTE network
   */
  Serial.println("An error has occurred. Shutdown and retry the network attach process after 1 second.");
  lteAccess.shutdown();
  sleep(1);
  // LTEモデム初期化に連続して失敗したらシステムリセットする
  err_cnt++;
  if (err_cnt > 10) system_reset();
}
// WatchDog
Watchdog.kick();
led0_blink();
/*
 * LTEネットワークから時刻を得てRTCに設定
 */
RTC.begin();
unsigned long currentTime;
while(0 == (currentTime = lteAccess.getTime())) {
  sleep(1);
}
RtcTime rtc(currentTime);
Serial.print("Time from LTE: ");
printClock(rtc);
RTC.setTime(rtc);
/*
 * Dropbox API用のルート証明をmicroSDから読み込んでLTETLSClientに設定
 */
// content.dropboxapi.comアクセス用
File rootCertsFile = theSD.open(ROOTCA_FILE_CONT, FILE_READ);
tlsClient.setCACert(rootCertsFile, rootCertsFile.available());
rootCertsFile.close();
// api.dropbox.comアクセス用
rootCertsFile = theSD.open(ROOTCA_FILE_API, FILE_READ);
tlsClient2.setCACert(rootCertsFile, rootCertsFile.available());
rootCertsFile.close();
/*
 * Dropbox API用の短期アクセストークン取得
 */
err_cnt = 0;
```

```
while(!dropbox_access_token_get()) {
  Serial.println("ERROR: access token can not get");
  // 連続して失敗したらシステムリセット
  if (err_cnt++ > 5) system_reset();
  sleep(5);
  Watchdog.kick();
}
// WatchDog
Watchdog.kick();
led0_blink();
/*
 * カメラ設定を行う
 */
Serial.println("Prepare camera");
// メモリ節約のため, Videoストリームで利用するバッファの数をゼロにする
err = theCamera.begin(0);     // Video buffer disableにする
if (err != CAM_ERR_SUCCESS) {
  printError(err);
}
// 自動露光調整に設定
Serial.println("Set Auto exposure ON");
err = theCamera.setAutoExposure(true);
if (err != CAM_ERR_SUCCESS) {
    printError(err);
}
// 自動ホワイトバランス調整に設定
Serial.println("Set Auto white balance ON");
err = theCamera.setAutoWhiteBalance(true);
if (err != CAM_ERR_SUCCESS) {
    printError(err);
}
// 自動ホワイトバランスのモード設定　自動モードに設定
Serial.println("Set Auto white balance parameter AUTO");
err = theCamera.setAutoWhiteBalanceMode(CAM_WHITE_BALANCE_AUTO);
if (err != CAM_ERR_SUCCESS) {
    printError(err);
}
// HDR(High Dynamic Range)機能オン
Serial.println("HDR ON");
theCamera.setHDR(CAM_HDR_MODE_ON);
// JPEG画質設定 10単位で設定する　80に設定
Serial.print("Set JPEG quality ");
err = theCamera.setJPEGQuality(JPEG_QUALITY);
if (err != CAM_ERR_SUCCESS) {
    printError(err);
}
Serial.println(theCamera.getJPEGQuality());

// 静止画のフォーマット設定 JPEGに設定
Serial.println("Set still picture format");
err = theCamera.setStillPictureImageFormat(
   CAM_IMGSIZE_QUADVGA_H,
   CAM_IMGSIZE_QUADVGA_V,
   CAM_IMAGE_PIX_FMT_JPG,      // <---- JPEGに設定
   JPEGBUFSIZE_DIVISOR);  // jpgbufsize_divisor 値を小さくするとJPEG画像バッファが大きくなり高画質で撮影が可能
if (err != CAM_ERR_SUCCESS) {
    printError(err);
}
led0_off();
}

/**
 * メインループ
 */
void loop()
{
  CamErr err;
  // WatchDogリセット
  Watchdog.kick();
  led1_on();
  // RTCから現在時刻取得
  RtcTime rtc = RTC.getTime();
  printClock(rtc);
  /*
   * 2時間毎にDropbox API用の短期アクセストークンを取得する
   */
  if ((rtc.hour() % 2 == 0) && (rtc.minute() == 0)) {
    int err_cnt = 0;
    while(!dropbox_access_token_get()) {
      Serial.println("ERROR: access token can not get");
      // 連続して失敗したらシステムリセット
      if (err_cnt++ > 5) system_reset();
      sleep(5);
      Watchdog.kick();
    }
  }

  // 撮影タイミングか調べる
  int sec = rtc.second() + rtc.minute()*60;
  int period = schedule[rtc.hour()].period;
  // 撮影するタイミングか？ 10secのウィンドウあり
```

```
if (sec % period > 10) {
  // 撮影しないで抜ける
  delay(100);
  led1_off();
  delay(900);
  return;
}
/*
 * 撮影開始
 */
Watchdog.kick();
/*
 * ISO感度と露光時間設定
 */
struct sched_t sched = schedule[rtc.hour()];
if (sched.iso == 0) {
  // 自動ISO感度調整に設定
  Serial.println("Set Auto ISO ON");
  err = theCamera.setAutoISOSensitivity(true);
  if (err != CAM_ERR_SUCCESS) {
      printError(err);
  }
} else {
  // ISO感度設定
  Serial.println("Set Auto ISO OFF");
  err = theCamera.setAutoISOSensitivity(false);
  if (err != CAM_ERR_SUCCESS) {
      printError(err);
  }
  Serial.print("Set ISO Sensitivity ");
  err = theCamera.setISOSensitivity(sched.iso);
  if (err != CAM_ERR_SUCCESS) {
      printError(err);
  }
  Serial.println(theCamera.getISOSensitivity());

}
if (sched.exposure == 0) {
  // 自動露光調整に設定
  Serial.println("Set Auto exposure ON");
  err = theCamera.setAutoExposure(true);
  if (err != CAM_ERR_SUCCESS) {
      printError(err);
  }
} else {
  // 自動露光調整OFF
  Serial.println("Set Auto exposure OFF");
  err = theCamera.setAutoExposure(false);
  if (err != CAM_ERR_SUCCESS) {
      printError(err);
  }
  // 露光時間設定
  Serial.println("Set Absolute exposure ");
  err = theCamera.setAbsoluteExposure(sched.exposure);
  if (err != CAM_ERR_SUCCESS) {
      printError(err);
  }
  Serial.println(sched.exposure);
}

// 静止画撮影API呼び出し
// このAPIは画像が得られるまで帰ってこない
Serial.println("call takePicture()");
  CamImage img = theCamera.takePicture();
  led1_blink();
  // 撮影画像が得られたか？
  if (img.isAvailable()) {
    int err_cnt = 0;
    // 画像有効なので画像をDropboxにPOSTする
    // 現在時刻でディレクトリ名を作る
    char dirname[32] = {0};
    sprintf(dirname, "%02d%02d%02d", rtc.year() % 100, rtc.month(), rtc.day());
    // 現在時刻でファイル名を作る
    char filename[32] = {0};
    sprintf(filename, "%02d%02d%02d_%02d%02d%02d.jpg", rtc.year() % 100, rtc.month(), rtc.day(), rtc.hour(), rtc.minute(),
rtc.second());
    // ファイルフルパス作成 microSDファイル記録とDropbox送信に使う
    char file_path[64] = {0};
    strcat(file_path, dirname);
    strcat(file_path, "/");
    strcat(file_path, filename);
    Serial.print("Save taken picture as ");
    Serial.println(file_path);

    // microSDへの記録は10分毎に間引きする
    if (rtc.minute() % 10 == 0) {
      Serial.println("Save to file on SD");
      err_cnt = 0;
      while(!file_save(img.getImgBuff(), img.getImgSize(), dirname, file_path)) {
        Serial.println("ERROR: file save failed");
        // 連続して失敗したらシステムリセット
        if (err_cnt++ > 5) system_reset();
```

```
        sleep(1);
        Watchdog.kick();
      }
      led1_blink();
    } // if (rtc.minute() % 10 == 0)
    Watchdog.kick();
    // 画像をDropboxにupload
    err_cnt = 0;
    while(!dropbox_upload(img.getImgBuff(), img.getImgSize(), file_path)) {
      Serial.println("ERROR: dropbox upload failed");
      // 連続して失敗したらシステムリセット
      if (err_cnt++ > 5) system_reset();
      sleep(5);
      Watchdog.kick();
    }
    Watchdog.kick();
    led1_blink();
  } else {
    // 撮影エラー
    Serial.println("Failed to take picture");
  } // if (img.isAvailable())
  // WatchDogリセット
  Watchdog.kick();
        led2_off();
  led1_off();
} // loop()
```