```
//------------------------------Station météo-----------------------------------------//

//-----Bibliothèque Température, Humidité, Pression : BME280-----//
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_BME280 bme;                    // use I2C interface

int temperature;
int hygrometrie;
int pression;


//-----Bibliothèque Capteur de luminosité-----//
#include "Digital_Light_ISL29035.h"
int luminosite;
int temps_defini1 = 10000;              // définition du temps désiré en ms soit 10 secondes
long debut_clignotement;
boolean variable2 = false;


//-----Bibliothèque Led RGB interruption-----//
#include <ChainableLED.h>
#include <stdlib.h>
#include <stdio.h>


ChainableLED leds(7, 8, 1);
int button1 = 2;                        // set the button1 pin
int button2 = 3;                        // set the button2 pin
```

```
volatile bool state1 = 1;                    // set button1 state     // State1 = 1 (éteint) et State1 !=1 (allumer)
volatile bool state2 = 1;                    // set button2 state     // State2 = 1 (éteint) et State2 !=1 (allumer)
int last_state_button1 = LOW;
int last_state_button2 = LOW;
volatile bool bascule=false;


//-----Durée de temps-----//
int temps_defini = 5000;                     // définition du temps désiré en ms soit 5 secondes
long debut_appui1;
boolean variable1 = false;
int delai_inter_capture;


//-----Mode de la sation météo-----//
boolean configuration = false;
boolean economique = false;
boolean activ_gps = false;


//-----Valeur borne capteur-----//
int MIN_TEMP_AIR = -10;
int MAX_TEMP_AIR = 60;

int HYGR_MINT = 0;
int HYGR_MAXT = 50;

float PRESSURE_MIN = 850 /1000.0F;
float PRESSURE_MAX = 1080 /1000.0F;
```

```
int LUMIN_LOW = 255;
int LUMIN_HIGH = 768;


//-----Bibliothèque Carte SD-----//
#include <SPI.h>
#include <SD.h>
File myfile;
String filename = "test";
String donne;


//-----Bibliothèque Horloge RTC-----//
#include <Wire.h>
#include <RTC.h>
static DS1307 RTC;


//-----Bibliothèque GPS-----//
#include <SoftwareSerial.h>
SoftwareSerial SoftSerial(4, 5);
unsigned char buffer[64];                 // buffer array for data receive over serial port
int count=0;                              // counter for buffer array


//----------------------------------DEBUT DU SETUP-----------------------------------------------------------------//

void setup()
{
  Serial.begin(9600);
```

```
//----------Setup Température, Humidité, Pression : BME280-------------------------//

  Serial.println(F("BME280 Sensor event test"));
  if (!bme.begin(0x76))
  {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1) delay(10);
  }


//---------------------Setup Carte SD-------------------------//

  Serial.print("Initializing SD card...");
   pinMode(10, OUTPUT);
  while (!SD.begin()) {
    Serial.println("initialization failed!");
    leds.setColorRGB(0, 255, 0, 0);
    delay(1000);
    leds.setColorRGB(0, 255, 255, 255);
    delay(2000);
    return;
  }
  Serial.println("initialization done.");


//--------------------Setup Horloge RTC-------------------------//

  RTC.begin();

  Serial.print("Is Clock Running: ");
```

```
  if (RTC.isRunning())
  {
  /*Serial.println(__DATE__);
  Serial.println(__TIME__);
     Serial.println("Yes");
     Serial.print(RTC.getDay());
     Serial.print(RTC.getMonth());
     Serial.print(RTC.getYear());
     Serial.print(RTC.getHours());
     Serial.print(RTC.getMinutes());
     Serial.print(RTC.getSeconds());*/
     if (RTC.getHourMode() == CLOCK_H12)
     {
        switch (RTC.getMeridiem())
        {
        case HOUR_AM:
           Serial.print(" AM");
           break;
        case HOUR_PM:
           Serial.print(" PM");
           break;
        }
     }
     Serial.println("");
     delay(1000);
  }
  else
  {
     delay(1500);
/*    Serial.println("No");
     Serial.println("Setting Time");*/
```

```
      RTC.setHourMode(CLOCK_H24);
      RTC.setDateTime(__DATE__, __TIME__);
/*    Serial.println("New Time Set");
      Serial.print(__DATE__);
      Serial.print(" ");
      Serial.println(__TIME__);*/
      RTC.startClock();
   }


//--------------Setup LED RGB Interruption----------------//

  pinMode(button1,INPUT);
  pinMode(button2,INPUT);
  //initialisation_interruption();

  delay(2000);
  if( digitalRead(button2) == 1)
  {
    configuration = true;
  }
}


//---------------FIN DU SETUP---------------------------------//



//--------------------------DEBUT DU PROGRAMME-------------------------------//

//-----------------Température, Humidité, Pression : BME280------------------//
```

```
int acquisition()
{
 Serial.print("Temperature = ");
 Serial.print(bme.readTemperature());                  // Température
 Serial.println(" *C");                               // Affiche la température en °C

 temperature = bme.readTemperature();

 Serial.print("Pression = ");
 Serial.print(bme.readPressure() / 1000.0F);           // Pression
 Serial.println("kPa");                               // Affiche la pression en kPa

 pression = bme.readPressure() / 1000.0F ;

 /*
 Serial.print("Approx. Altitude = ");
 Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));    // Altitude
 Serial.println("m");                                 // Affiche l'altitude en m
 */

 Serial.print("Humidité = ");
 Serial.print(bme.readHumidity());                    // Humidité
 Serial.println("%");                                 // Affiche le taux d'humidité en %

 hygrometrie = bme.readHumidity();


 if ( temperature < MIN_TEMP_AIR || temperature > MAX_TEMP_AIR || pression < PRESSURE_MIN
|| pression > PRESSURE_MAX || hygrometrie < HYGR_MINT || hygrometrie > HYGR_MAXT)
 {
```

```
   int debut_clignotement1 = millis();
   while((millis()-debut_clignotement1) < 10000)          // Si la température est inférieur à -10°C ou supérieur à 60°C  et .... alors...
   {
    Serial.println("Données reçues d'un capteur incohérentes -vérification matérielle requise");
    leds.setColorRGB(0, 255, 0, 0);  // LED rouge
    delay(1000);                                  // Sur une durée de 10 secs
    leds.setColorRGB(0, 0, 255, 0); //LED verte          // LED  intermittente rouge et verte(fréquence 1Hz, durée 2 fois plus longue pour le vert)
    delay(2000);
   }
  }
  return temperature,hygrometrie,pression;

}



//--------------------------Led RGB interruption-----------------------------//
/*
void initialisation_interruption()
{
  attachInterrupt(digitalPinToInterrupt(button1), basculer,CHANGE);
  attachInterrupt(digitalPinToInterrupt(button2), basculer,CHANGE);
}



void basculer() // ISR // Pas de Delay dans un ISR
{
  bascule =!bascule;


  if(bascule)
  {
```

```
   ledRGB();
  }
  else
  {
   void loop();
  }
}
*/


//-------------------------------------------------------------------------------------------------

boolean changement_de_mode()
{
  state1 = digitalRead(button1);                    // Bouton vert = state1
  state2 = digitalRead(button2);                    // Bouton rouge = state2

  if(state1 == 0 && last_state_button1 == LOW)              //-----> Mode Economique
  {                                          //-----> Bouton Vert ----> 5 secondes
   debut_appui1 = millis();                          // On y accède en appuyant pendant 5 secondes sur
le bouton vert
   variable1 = true;

   if(variable1 == true && state1 == 0 && last_state_button1 == HIGH)
   {
    while((millis() - debut_appui1) < temps_defini && last_state_button1 ==HIGH){}
    if((millis()- debut_appui1)>= temps_defini)
    {
     economique = !economique;
     return economique;
    }
```

```
   }
  }

  else if(state2 == 0 && last_state_button2 == LOW)            //-----> Mode Maintenance
  {                                                  //-----> Bouton rouge -----> 5 secondes
   debut_appui1 = millis();                                  // On y accède en appuyant pendant 5 secondes sur le bouton rouge
   variable1 = true;
   if(variable1 == true && state2 == 0 && last_state_button2 == HIGH)
   {
    while((millis() - debut_appui1) < temps_defini && last_state_button2 ==HIGH){}
    if((millis() - debut_appui1) >= temps_defini)
    {
     mode_maintenance();
    }
   }
  }
}



//-----------Différents modes de maintenance----------//



void mode_configuration()                                  //-----> Mode Configuration
{                                                  //-----> Led jaune continue
  leds.setColorRGB(0, 255, 255, 0);
  delay(10000);
  // CODE //
}
```

```
void mode_maintenance()                          //-----> Mode Maintenance
{                                              //-----> Led orange continue
  leds.setColorRGB(0, 255, 165, 0);
  delay(10000);
  // CODE //
}

void mode_standard_plus_eco()
{
  if(economique == true)
  {
   delai_inter_capture = 1200000;                  // Temps entre 2 cpatures (20min)
   leds.setColorRGB(0, 0, 0, 255);                //Led Bleue
   luminosite = NULL;

    if(activ_gps == true)
    {
     global_positioning_system();
     activ_gps = !activ_gps;
    }
    else
    {
     activ_gps = !activ_gps;
     clearBufferArray();
    }
  }
  else
  {
   delai_inter_capture = 600000;                  // Temps entre 2 cpatures (10min)
   leds.setColorRGB(0, 0, 255, 0);                // Led vert
   luminosite = analogRead(A0);                   // Capteur luminosité
```

```
   if(luminosite < LUMIN_LOW || luminosite > LUMIN_HIGH)
   {
    debut_clignotement = millis();
    while((millis() - debut_clignotement) < temps_defini1)
    {
     leds.setColorRGB(0, 255,0, 0);
     delay(1000);
     leds.setColorRGB(0, 0, 255, 0);
     delay(2000);
    }
   }
  }
 acquisition();                                  // Capture des la pression, de l'humidité et de la
Température
 global_positioning_system();                          // Recuperation des données gps
 carte_sd();



}



//------------------------------GPS---------------------------------//

unsigned char global_positioning_system()
{
  clearBufferArray();
   while(SoftSerial.available())          // reading data into char array
   {
    buffer[count++]=SoftSerial.read();     // writing data into array
    if(count == 64)break;
```

```
  }
 return buffer;
}

void clearBufferArray()                 // Fonction pour vide le tableau buffer
{
   for (int i=0; i<count;i++)
   {
      buffer[i]=NULL;
   }                                    // vider chaque case du tableau en lui assignant la valeur NULL
}



//----------------------Sauvegarde Carte SD-----------------------------//


void carte_sd()
{
 String date = String(RTC.getYear()) + String(RTC.getMonth()) + String(RTC.getDay());
 myfile = SD.open(date + "_0.LOG", FILE_WRITE);
 String donne = String(RTC.getHours()) + String(RTC.getHours()) + String(":") + String(temperature) +
String(hygrometrie) + String(pression) + String(luminosite);
 myfile.println(donne);
  if (myfile.size() >= 16384)
  {
   int i = 1;
    while (true)
    {
     if(!SD.exists(date + "_" + i + ".LOG"))
      {
       File second_file = SD.open(date + "_" + i + ".LOG", FILE_WRITE);
       second_file.write(myfile.read());
```

```
        second_file.close();
        break;
       }
     i++;
     if (i>50) break;
    }
  }
  myfile.close();
}


//----------------------VOID LOOP-------------------------------------//

void loop()
{
  if (configuration == true)
  {
   mode_configuration();
  }
  else
  {
   mode_standard_plus_eco();
  int fin_last_capture = millis(); //temps de la dernière capture
  while ((millis()-fin_last_capture) < delai_inter_capture)
  {
   changement_de_mode();
  }
  }

}
```

```
//-----------------------------------------------------------------
```