Cosmix ou compteurs de muons

valise II à seuil 100 mV

logiciel 3.1 installé

utilisateur

Clavier

LCD

Scintillateurs (muons et particules)

5 AI

I2C-//

1 AI : coïncidence
2 IT : C1 & C2

EEPROM

interne au CPU

Arduino
3.3

carte SD

SPI

I2C

RS232/9600 bauds

USB

RTC

RS232/4800 bauds

I2C

GPS

Baromètre

Home

```
#include <EEPROM.h>
#include <TinyGPS.h>
#include <Serial1.h>
#include <RTClib.h>
#include <SD.h>
#include <Wire.h>
#include <Adafruit_MCP23017.h>
#include <Adafruit_RGBLCDShield.h>
```

logiciel

```
// ------- //
// defines //
// ------- //
#define COINC_LINE 4
#define VERSION "m3.3s"
#define SERIALNUM "0-0"
```

```
// ---------- //
// Globals Objects //
// ---------- //


RTC_DS1307 RTC;
//RTC_Millis RTC_m;
TinyGPS gps;
Adafruit_RGBLCDShield lcd =
Adafruit_RGBLCDShield();
```

EEPROM

GPS

```
// GPS
/*
 * GPS RX: pin 19
 TX: pin 18
*/
#define RXPIN 19
#define TXPIN 18
#define GPSBAUD  4800
```

baromètre

```
// BMP085 pressure sensor
#define BMP085_ADDRESS 0x77  // I2C address of BMP085
```

```
/*
 * Interrupt pin  2 .
 * BMP085 SDA 20
 SCL 21
 */
```

```
// BMP085
const unsigned char OSS = 0;  // Oversampling Setting
```

```
// BMP085 Calibration values
int ac1;
int ac2;
int ac3;
unsigned int ac4;
unsigned int ac5;
unsigned int ac6;
int b1;
int b2;
int mb;
int mc;
int md;
// b5 is calculated in bmp085GetTemperature(...), this variable is also used in bmp085GetPressure(...)
// so ...Temperature(...) must be called before ...Pressure(...).
long b5;
```

```
void bmp085Calibration()

short bmp085GetTemperature(unsigned int ut)

long bmp085GetPressure(unsigned long up)

char bmp085Read(unsigned char address)

int bmp085ReadInt(unsigned char address)

unsigned int bmp085ReadUT()

unsigned long bmp085ReadUP()
```

RTC

```
#define RTC_DISPLAY_POS 15
```

SD

```
// SD

#define SD_LINE 10
```

```
/*
place #define MEGA_SOFT_SPI 1 in libraries/SD/utility/Sd2Card.h
 * SD card attached to SPI bus as follows:
 ** MOSI - pin 11
 ** MISO - pin 12
 ** CLK - pin 13
 ** CS  - pin 10
*/
```

```
// SD Card
const int chipSelect = SD_LINE;
```

clavier

```
// Buttons Pad
#define BUTTONSPAD_AI     0
#define BUTTON_UP     0x1
#define BUTTON_DOWN   0x2
#define BUTTON_LEFT   0x3
#define BUTTON_RIGHT  0x4
#define BUTTON_SELECT 0x5
#define BUTTON_NOTHING  0x0
#define BUTTON_NEXT   0x6
```

```
//=====================================================
// Stop Acquisition    (VIOLET BackGround color = Acq stopped)
// ---------------------------------
// Detach interrupts
// Close datafile
// Display final counts
//=====================================================
void stopAcquisition()
```

```
if(serialcom)Serial.println("stopped");
```

```
//=====================================================
// Start Acquisition (GREEN BackGround = Acq running)
// ---------------------------------
// Get initial time
// Open datafile
// attach interrupts
// Display counts
//=====================================================
void startAcquisition()
```

```
if(serialcom)Serial.println("started");
```

électronique détecteur

MCP23017

Afficheur LCD

```
// AdaFruit LCD RGB 16x2 Shield
#define RED 0x1
#define YELLOW 0x3
#define GREEN 0x2
#define TEAL 0x6
#define BLUE 0x4
#define VIOLET 0x5
#define WHITE 0x7
#define BLANK_LINE "                "

#define WELCOME_BKGCOLOR WHITE
#define ACQ_STOPPED_BKGCOLOR VIOLET
#define ACQ_STARTING_BKGCOLOR RED
#define ACQ_RUNNING_BKGCOLOR GREEN
#define WARNING1_BKGCOLOR RED
#define WARNING2_BKGCOLOR YELLOW

#define CFG_ACQ_BKGCOLOR TEAL
#define CFG_GPS_BKGCOLOR BLUE
#define DISP_TEMP_BKGCOLOR YELLOW
#define DISP_GPS_BKGCOLOR BLUE
#define ABOUT_BKGCOLOR TEAL


#define TITLE_MENU_ACQ          ""
#define TITLE_MENU_CFG_ACQ      "Cfg Acquisition"
#define TITLE_MENU_CFG_GPS      "Cfg GPS"
#define TITLE_MENU_DISP_TP      "Temp/Press Info"
#define TITLE_MENU_DISP_GPS     "GPS Info"
#define TITLE_MENU_ABOUT        "COSMIX Info"
```

```
// Display
#define NB_CFG_ACQ_DIGI 4
#define MAX_DURATION 9999

#define NB_MENU 6

#define MENU_ACQ      0
#define MENU_CFG_ACQ  1
#define MENU_CFG_GPS  2
#define MENU_DISP_TP  3
#define MENU_DISP_GPS 4
#define MENU_ABOUT    5

#define RTC_UNSET 0
#define RTC_PC    1
#define RTC_LOCAL 2
#define RTC_GPS   3
```

guide logiciel

Home

librairies

constantes

variables

fonctions

évènements

main()

setup()

loop()

```
// ------- //
// defines //
// ------- //
#define COINC_LINE 4
#define VERSION "m3.3s"
#define SERIALNUM "0-0"

// BMP085 pressure sensor
#define BMP085_ADDRESS 0x77  // I2C address of BMP085
// AdaFruit LCD RGB 16x2 Shield
#define RED 0x1
#define YELLOW 0x3
#define GREEN 0x2
#define TEAL 0x6
#define BLUE 0x4
#define VIOLET 0x5
#define WHITE 0x7
#define BLANK_LINE "                "

#define WELCOME_BKGCOLOR WHITE
#define ACQ_STOPPED_BKGCOLOR VIOLET
#define ACQ_STARTING_BKGCOLOR RED
#define ACQ_RUNNING_BKGCOLOR GREEN
#define WARNING1_BKGCOLOR RED
#define WARNING2_BKGCOLOR YELLOW

#define CFG_ACQ_BKGCOLOR TEAL
#define CFG_GPS_BKGCOLOR BLUE
#define DISP_TEMP_BKGCOLOR YELLOW
#define DISP_GPS_BKGCOLOR BLUE
#define ABOUT_BKGCOLOR TEAL


#define TITLE_MENU_ACQ          ""
#define TITLE_MENU_CFG_ACQ      "Cfg Acquisition"
#define TITLE_MENU_CFG_GPS      "Cfg GPS"
#define TITLE_MENU_DISP_TP      "Temp/Press Info"
#define TITLE_MENU_DISP_GPS     "GPS Info"
#define TITLE_MENU_ABOUT        "COSMIX Info"

#define RTC_DISPLAY_POS 15

// Buttons Pad
#define BUTTONSPAD_AI     0
#define BUTTON_UP     0x1
#define BUTTON_DOWN   0x2
#define BUTTON_LEFT   0x3
#define BUTTON_RIGHT  0x4
#define BUTTON_SELECT 0x5
#define BUTTON_NOTHING  0x0
#define BUTTON_NEXT   0x6

// GPS
/*
 * GPS RX: pin 19
 TX: pin 18
*/
#define RXPIN 19
#define TXPIN 18
#define GPSBAUD  4800
// SD

#define SD_LINE 10

/*
place #define MEGA_SOFT_SPI 1 in libraries/SD/utility/Sd2Card.h
 * SD card attached to SPI bus as follows:
 ** MOSI - pin 11
 ** MISO - pin 12
 ** CLK - pin 13
 ** CS  - pin 10
*/



/*
 * Interrupt pin  2 .
 * BMP085 SDA 20
 SCL 21
 */

// Display
#define NB_CFG_ACQ_DIGI 4
#define MAX_DURATION 9999

#define NB_MENU 6

#define MENU_ACQ      0
#define MENU_CFG_ACQ  1
#define MENU_CFG_GPS  2
#define MENU_DISP_TP  3
#define MENU_DISP_GPS 4
#define MENU_ABOUT    5

#define RTC_UNSET 0
#define RTC_PC    1
#define RTC_LOCAL 2
#define RTC_GPS   3


// --------- //
// Constants //
// --------- //

// SD Card
const int chipSelect = SD_LINE;

// BMP085
const unsigned char OSS = 0;  // Oversampling Setting
```

Guide

```
// -------- //
// Globals  //
// -------- //


// Config
bool continuousmode = true;
int duration = 0;
bool acqrunning = false;
char filename[15];
File dataFile;
char gpsfilename[15];
File gpsFile;
char serialnum[10];

// Event Counter
volatile long countPulse1=0;
volatile long countPulse2=0;
volatile long countCoinc=0;
volatile boolean event1=false;
volatile boolean event2=false;
volatile boolean eventc=false;
volatile long time0;

// module independant variables
short temperature;
long pressure;
// you can change the overall brightness by range 0 -> 255
int brightness = 255;
char cmode[2]={
  'D',char(243)};
char RTCmodeDisplay[4]={
  char(255),'\3','\4','\2'};
int RTCmode = RTC_UNSET;
char mess[80];
char longmess[200];
int longmessstart;
char cfg_acq_selindex = 0;
boolean acqdisplay = true;
boolean gpsmode = false;
boolean gpsset = false;
int gpstry = 0;
int selmenu = MENU_ACQ;
int localtime = 2;
char slatitude[15], slongitude[15];
float gpsaltitude = -999;
float latitude = -999;
float longitude = -999;
int lannee;
byte lemois;
byte lejour;
byte lheure;
byte laminute;
byte laseconde;
byte lescentiemes;
unsigned long lesmillis;
char lestatus[5];
char ascdate[100];
int about_index = 0;
DateTime now;

//ADKeyboard Module
int adc_key_val[5] ={
  50, 200, 400, 600, 800 };
int NUM_KEYS = 5;
int adc_key_in;
int key=-1;
int oldkey=-1;

// Serial comm data
char serialbuffer[20];
boolean stringComplete = false;
char datamess[200];
boolean serialcom = false;

byte newChar[8] = {
                  //4,14,31,0,0,0,0}; "^"
                  B00100,
                  B01110,
                  B11111,
                  B00000,
                  B00000,
                  B00000,
                  B00000,
                  B00000
                 };
byte newChar1[8] = {
                  //4,14,31,0,0,0,0}; "intersection"
                  B00000,
                  B01110,
                  B11011,
                  B10001,
                  B10001,
                  B10001,
                  B10001,
                  B00000
                 };
byte newChar2[8] = {
                  // antenne GPS
                  B00000,
                  B10001,
                  B10010,
                  B01100,
                  B01100,
                  B10011,
                  B10000,
                  B10000
                 };
byte newChar3[8] = {
                  //"PC"
                  B11000,
                  B10100,
                  B11000,
                  B10011,
                  B00100,
                  B00100,
                  B00011,
                  B00000
                 };
byte newChar4[8] = {
                  //"LC"
                  B10000,
                  B10000,
                  B10000,
                  B11011,
                  B00100,
                  B00100,
                  B00011,
                  B00000
                 };
byte newChar5[8] = {
                  //"é"
                  B00010,
                  B00100,
                  B01110,
                  B10001,
                  B11111,
                  B10000,
                  B01110,
                  B00000
                 };
byte newChar6[8] = {
                  //"à"
                  B01000,
                  B00100,
                  B01110,
                  B00001,
                  B01111,
                  B10001,
                  B01111,
                  B00000
                 };

char *menu[NB_MENU] = {
                    TITLE_MENU_ACQ,
                    TITLE_MENU_CFG_ACQ,
                    TITLE_MENU_CFG_GPS,
                    TITLE_MENU_DISP_TP,
                    TITLE_MENU_DISP_GPS,
                    TITLE_MENU_ABOUT
                    };

// BMP085 Calibration values
int ac1;
int ac2;
int ac3;
unsigned int ac4;
unsigned int ac5;
unsigned int ac6;
int b1;
int b2;
int mb;
int mc;
int md;
// b5 is calculated in bmp085GetTemperature(...), this variable is also used in bmp085GetPressure(...)
// so ...Temperature(...) must be called before ...Pressure(...).
long b5;


// ---------------- //
// Globals Objects //
// ---------------- //


RTC_DS1307 RTC;
//RTC_Millis RTC_m;
TinyGPS gps;
Adafruit_RGBLCDShield lcd = Adafruit_RGBLCDShield();
```

```
void setup()
{
  boolean SD_OK=false;
  int address = 0;
  byte c;
  gpsset = false;
  Serial.begin(9600);
  RTC.begin();
  Serial1.begin(GPSBAUD);
  Wire.begin();
  bmp085Calibration();
  lcd.begin(20, 4);
  lcd.createChar(0, newChar);
  lcd.createChar(1, newChar1);
  lcd.createChar(2, newChar2);
  lcd.createChar(3, newChar3);
  lcd.createChar(4, newChar4);
  lcd.createChar(5, newChar5);
  lcd.createChar(6, newChar6);
  lcd.clear();
  lcd.setBacklight(WELCOME_BKGCOLOR);

  lcd.setCursor(0,0);
  sprintf(mess,"COSMIX  v%4s",VERSION);
  lcd.print(mess);
  lcd.setCursor(0,1);
  lcd.print("par le CENBG");
  delay(2000);
  do
    {
     c = EEPROM.read(address);
     serialnum[address] = c;
     address = address+1;
    }while (c);

  if (!RTC.isrunning()||(RTC.now().year()==2165))
  {
   char mess[50];
   sprintf(mess,"Horloge inactive");
    lcd.setCursor(0,1);
    lcd.print(mess);
   Serial.println(mess);
   delay(2000);
   RTC.adjust(DateTime(__DATE__, __TIME__));
   now = RTC.now();
    sprintf(mess,"Ajustement");
    lcd.setCursor(0,1);
    lcd.print(mess);
    delay(2000);
    lcd.setCursor(0,1);
    lcd.print(BLANK_LINE);
    sprintf(mess,"de la date \6");
    lcd.setCursor(0,1);
    lcd.print(mess);
    delay(2000);
    lcd.setCursor(0,1);
    lcd.print(BLANK_LINE);
    sprintf(mess,"%04d/%02d/%02d",now.year(),now.month(),now.day());
    lcd.setCursor(0,1);
    lcd.print(mess);
    delay(2000);
    lcd.setCursor(0,1);
    lcd.print(BLANK_LINE);
    sprintf(mess,"%02d:%02d:%02d",now.hour(),now.minute(),now.second());
    lcd.setCursor(0,1);
    lcd.print(mess);
    delay(2000);
    lcd.setCursor(0,1);
    lcd.print(BLANK_LINE);
    delay(2000);
   Serial.println(now.year());
   RTCmode = RTC_PC;
  }
  else
   RTCmode = RTC_LOCAL;


  // make sure that the default chip select pin is set to
  // output, even if you don't use it:
  sprintf(mess,"Init carte SD...");
  lcd.setCursor(0,1);
  lcd.print(mess);
  delay(1000);
  if(serialcom)Serial.print(mess);
  // SD Init
  SD_OK = SD.begin(chipSelect);
  sprintf(mess,(SD_OK)?"SD initialis\5e. ":"Echec SD        ");
  (SD_OK)?
lcd.setBacklight(ACQ_RUNNING_BKGCOLOR):lcd.setBacklight(ACQ_STARTING_BKGCOLOR);
  lcd.setCursor(0,1);
  lcd.print(mess);
  if(serialcom)Serial.println(mess);

  sprintf(gpsfilename,"GPS.TXT");
  gpsFile = SD.open(gpsfilename, FILE_WRITE);
  delay(1000);
  lcd.setBacklight(WELCOME_BKGCOLOR);
  lcd.setCursor(0,1);
  lcd.print(BLANK_LINE);

  // see if the card is present and can be initialized:
  // acquisition auto start
  acqrunning = false;
  continuousmode = true;
  serialbuffer[0]=0;
  InitCounters();
  selmenu = MENU_ACQ;
  acquisition_display(BUTTON_NEXT);
  acquisition_display(BUTTON_RIGHT);
}
```

```
#include <EEPROM.h>
#include <TinyGPS.h>
#include <Serial1.h>
#include <RTClib.h>
#include <SD.h>
#include <Wire.h>
#include <Adafruit_MCP23017.h>
#include <Adafruit_RGBLCDShield.h>
```

```
void float2string(float f,char *s,int precision)
{
  if ((long(f)==0)&&(f<0))
    sprintf(s,"-%ld.%lu",long(f),(f>=0)?long((f-long(f))*pow(10,precision)):long((long(f)-
f)*pow(10,precision)));
  else
    sprintf(s,"%ld.%lu",long(f),(f>=0)?long((f-long(f))*pow(10,precision)):long((long(f)-
f)*pow(10,precision)));
  return;
}

void InitCounters()
{
  countPulse1=0;
  countPulse2=0;
  countCoinc=0;
  event1 = false;
  event2 = false;
  eventc = false;
  return;
}

uint8_t serialcommandanalysis()
{
  char buttonsstring[] ="start stop  reset stat  mod   gps   *idn  s/n   ";
  char command[10];
  char data[10];
  boolean request = false;
  sscanf(serialbuffer,"%s %s",command,data);
  if(command[strlen(command)-1]=='?')
  {
    request = true;
    command[strlen(command)-1]=0;
  }
  char *s = strstr(buttonsstring,command);
  switch((s-buttonsstring)/6)
  {
  case 0:
    selmenu = MENU_ACQ;
    selectdisplay(false,BUTTON_NEXT);
    return (acqrunning)?BUTTON_NOTHING:BUTTON_RIGHT;
  case 1:
    selmenu = MENU_ACQ;
    selectdisplay(false,BUTTON_NEXT);
    return (acqrunning)?BUTTON_RIGHT:BUTTON_NOTHING;
  case 2:
    selmenu = MENU_ACQ;
    selectdisplay(false,BUTTON_NEXT);
    return BUTTON_LEFT;
  case 3:
    Serial.println(acqrunning?"STATUS running":"STATUS stopped");
    return BUTTON_NOTHING;
  case 4:
    if(request)
    {
      if(continuousmode)
        Serial.println("MOD Infini");
      else
      {
        sprintf(mess,"MOD %04i",duration);
        Serial.println(mess);
      }
    }
    else
    {
      if(!acqrunning&&atoi(data)>0)
      {
        duration = atoi(data);
        continuousmode = false;
      }
      else
      {
        continuousmode = true;
      }
      selectdisplay(false,BUTTON_NEXT);
    }
    return BUTTON_NOTHING;
  case 5:
    if(request)
    {
      if(gpsmode)
        Serial.println("GPS Requis");
      else
        Serial.println("GPS Optionnel");
    }
    else
    {
      if(!acqrunning&&strstr(data,"req"))
      {
        gpsmode = true;
      }
      else
      {
        gpsmode = false;
      }
      selectdisplay(false,BUTTON_NEXT);
    }
    return BUTTON_NOTHING;
  case 6:
    serialcom = false;
    Serial.flush();
    sprintf(mess,"COSMIX %s",VERSION);
    Serial.println(mess);
    return BUTTON_NOTHING;
  case 7:
    sprintf(mess,"S/N %s",serialnum);
    Serial.println(mess);
    return BUTTON_NOTHING;
  }
  return BUTTON_NOTHING;
}
```

Guide

```
void loop()
{
 bool init=false;
 uint8_t buttons = readButtons();

 if(stringComplete)
 {
  /*   Serial.print("Incomming String  --> ");
   Serial.println(serialbuffer);*/
  buttons = serialcommandanalysis();
  stringComplete=false;
 }

 if(acqrunning&!continuousmode)
 {
  // si l'acquisition tourne et qu'elle n'est pas en mode continu, on envoie le stop
  // en simulant une pression sur le bouton droit du menu acquisition
  if((millis()-time0)>(60000*duration))
  {
    acquisition_display(BUTTON_NEXT);
    acquisition_display(BUTTON_RIGHT);
  }
 }
 if(selmenu==MENU_CFG_GPS)
 {
  config_GPS_refresh_display();
  delay(100);
 }
 if(selmenu==MENU_DISP_TP)
 {
  TP_display();
  delay(100);
 }

 if(selmenu==MENU_DISP_GPS)
  {
   GPS_display();
   delay(100);
  }

 if(selmenu==MENU_ABOUT)
 {
  about_display();
  delay(100);
 }

 if (buttons)selectdisplay(init,buttons);



 getGPSInfo();

 if (eventc or event1 or event2)
 {
  if (gpsmode && !gpsset)
       {
        InitCounters();
        if (selmenu==MENU_ACQ)
          {
       (gpstry%2)?lcd.setBackLight(WARNING1_BKGCOLOR):lcd.setBackLight(WARNING2_BKGCOLOR);
        sprintf(mess,"Attente GPS     ");
        lcd.setCursor(0,1);
        lcd.print(mess);
           }
        gpstry = gpstry +1;
        return;
       }
   char eventmess[8];
   sprintf(eventmess,"%1i,%1i,%1i",(event1)?1:0,(event2)?1:0,(eventc)?1:0);
   event1 = false;
   event2 = false;
   eventc = false;
   temperature = bmp085GetTemperature(bmp085ReadUT());
   pressure = bmp085GetPressure(bmp085ReadUP());
   if (selmenu==MENU_ACQ)
     {
  lcd.setBackLight(ACQ_RUNNING_BKGCOLOR);
      displaycount();
     }
   // write to file
   char slat[30];
   char slong[30];
   float2string(latitude,slat,6);
   float2string(longitude,slong,6);
   sprintf(datamess,"DATA %4d,%02d,%02d,%02d,%02d,%02d.%02d,(%s),%lu,%s,%s,%d,%d.%d,%lu,%s,%ld,
%ld,%ld"
       ,lannee,lemois,lejour,lheure,laminute,laseconde,lescentiemes,lestatus,lesmillis,
     slat,slong,
     (int)gpsaltitude,
     (int)((float)temperature/10.),(temperature<0)?(int)((-temperature)%10):(int)(temperature%10),
     (long)pressure,eventmess,countPulse1,countPulse2,countCoinc);
   if (dataFile)
   {
     dataFile.println(datamess+5); // suppress 'DATA ' keyword in mess
     dataFile.flush();
   }
   if(serialcom)Serial.println(datamess);
 }
 if (gpsFile)
    {
     char slat[30];
     char slong[30];
     float2string(latitude,slat,6);
     float2string(longitude,slong,6);
     sprintf(datamess,"%4d,%02d,%02d,%02d,%02d,%02d.%02d,(%s),%lu,%s,%s,%d,%d.%d,%lu"
       ,lannee,lemois,lejour,lheure,laminute,laseconde,lescentiemes,lestatus,lesmillis,
     slat,slong,
     (int)gpsaltitude,
     (int)((float)temperature/10.),(temperature<0)?(int)((-temperature)%10):(int)(temperature%10),
     (long)pressure);
     gpsFile.println(datamess);
     gpsFile.flush();
    }
 DateTime now = RTC.now();
}
```

```
void gestionINT01() {
  countPulse1 += 1 ;
  event1 = true;
  if(digitalRead(COINC_LINE))
  {
    countCoinc +=1;
    eventc = true;
  }
}

void gestionINT02() {
  countPulse2 +=1;
  event2 = true;
}

void serialEvent()
{
  int i;
  i = 0;
  if(!serialcom)serialcom=true;
  while (Serial.available())
  {
    // get the new byte:
    delay(1);
    char inChar = (char)Serial.read();
    // add it to the inputString:
    serialbuffer[i] = tolower(inChar);
    // if the incoming character is a newline, set a flag
    // so the main loop can do something about it:

    i = i + 1;
    if (inChar == '\n')
    {
      stringComplete = true;
      serialbuffer[i-1]=0;
      if (serialbuffer[i-2]=='\r')
        serialbuffer[i-2]=0;
    }
  }
}
```