```
/*  Programme  "alertes aquaponiques par SMS"  pour Arduino Uno R3
avec  capteur température DS18B20, capteur débit  YFS201,
sondes vibrations  SW18010 et shield GSM GPRS SIM900.
 Version 1.5  - 24 avril 2020   -
https://www.truitesaquaponiques.com/2019/12/30/alerte-automatique-aquaponie/  */

//Bibliothèques
//Bibliothèque pour sonde température
#include <OneWire.h>

//Bibliothèque pour SIM900
#include <SoftwareSerial.h>

//Déclaration de variables

// Variables générales
const int SERIAL_PORT=9600;
unsigned long dateDernier1;
unsigned long dateDernier2;

unsigned long dateCourante;

// Variables liées à SIM900
//Configure RX sur pin 7 et TX sur pin 8
SoftwareSerial SIM900(7, 8);

// Variables propres à la sonde de température  DS18B20
// fil jaune du signal sur pin 11
const int DS18B20_PIN=11;
const int DS18B20_ID=0x28;
// Déclaration de l'objet ds sur pin DS18B20_PIN
OneWire ds(DS18B20_PIN);
// seuil de température maximale
unsigned char temperatureLimiteMax=23;
//seuil de température minimale
unsigned char temperatureLimiteMin=6;
// variable de stockage de la température
float DS18B20Temperature;

//Variables propres aux 2 sondes vibrations  SW18010
int BrocheCapteur1 =9;
int BrocheCapteur2 =10;

long mesureVibration1 = 0;
long mesureVibration2 = 0;
int nombreMesureVibration = 0;
long totalMesureVibration1 = 0;
long totalMesureVibration2= 0;
long mesureVibration1Temp = 0;
long mesureVibration2Temp= 0;
```

```
long totalMesureVibration1Bis = 0;
long totalMesureVibration2Bis = 0;

//Variables liées aux SMS
//déclaration de  chaines de caractères
String chaineSMS, chaineSMS1, message, message1;
// compteur du nombre de SMS envoyé
int nombreSMS = 0;
char c;
char str[200];

//Variables liées au capteur de debit  bassin
byte sensorInterrupt = 0;
// fil jaune du signal sur pin2
byte sensorPin = 2;
/* Pour le calcul du flux d'eau*/
//4,5 pulsations par s pour 1 litre par mn
float calibrationFactor = 4.5;
//variable pour compter les pulsations
volatile byte pulseCount;
//variable pour stocker le débit des bassins à truites
float debit1;
unsigned int flowMilliLitres;
unsigned long oldTime;

//Paramétrage
void setup()
{
// Initialisation du port de communication avec le PC
Serial.begin(SERIAL_PORT);
 //Arduino communique avec le moniteur serie à une fréquence de  9600 bauds
 Serial.begin(9600);
Serial.println("Début Initialisation du programme");

// Paramétrages liés au capteur de débit bassin
pinMode(sensorPin, INPUT);
digitalWrite(sensorPin, HIGH);
pulseCount = 0;
debit1 = 0.0;
flowMilliLitres = 0;
oldTime = 0;

///Paramétrages liés au shield SIM 900
//Arduino communique avec SIM900 GSM à une fréquence de  9600 bauds
SIM900.begin(19200);
//Donner du temps pour que le GSM se connecte : 20 secondes
 delay(30000);
SIM900.println("AT"); //Handshaking with SIM900
updateSerial();
SIM900.println("AT+CSQ"); //Signal quality test, value range is 0-31 , 31 is the best
```

```
updateSerial();
SIM900.println("AT+CCID"); //Read SIM information to confirm whether the SIM is plugged
updateSerial();
SIM900.println("AT+CREG?"); //Check whether it has registered in the network
updateSerial();
//Configuring TEXT mode
SIM900.println("AT+CMGF=1");
updateSerial();
//Les SMS reçus ne passent pas par la carte SIM. Ils sont perdus.
SIM900.println("AT+CNMI=1,2,0,0,0");
updateSerial();
//effacer dernier SMS
SIM900.println("AT+CMGD=1,4\r");
updateSerial();

Serial.println("Fin Initialisation du programme");
Serial.println("");
}
//Programme principal exécuté en boucle
void loop()
{
 // "surveillance  de la pendule" toutes les 1 seconde pour la reception de SMS
 dateCourante = millis();
 if(dateCourante - dateDernier1 >= 1000)
   {
  if (SIM900.available())
   {
     char c = SIM900.read();
     Serial.print (c);
  //repérage du caractère "S" (S pour Sonde?) constituant le contenu du SMS
       if (c=='S')
        {
  //Préparation du message de réponse avec les valeurs des sondes
      message="";
  //Transformation du nombre 'DS18B20Temperature' en texte et ajout à message
        sprintf(str, "Temp : %d.%02d", int(DS18B20Temperature), (int((DS18B20Temperature)*100)%100));
        Serial.println(str);
     message=message+str + " ; ";
//Transformation du nombre 'debit' en texte et ajout à message
     sprintf(str, "debit : %d", int(debit1));
       Serial.println(str);
     message=message+str + " ; " ;
//Transformation du nombre 'mesureVibration1' en texte et ajout à message
     sprintf(str, "mesureVibration1 : %d", int(totalMesureVibration1Bis));
       Serial.println(str);
     message=message+str + " ; ";
//Transformation du nombre 'mesureVibration2' en texte et ajout à message
     sprintf(str, "mesureVibration2 : %d", int(totalMesureVibration2Bis));
       Serial.println(str);
     message=message+str + " ; ";
```

```
//Envoyer SMS
chaineSMS=message ;
 sendSMS();
//effacer dernier SMS
//SIM900.print("AT+CMGD=1,4\r");
//updateSerial();
//delay(2000);
// Reset la SIM900
SIM900.println("ATZ\r");
updateSerial();
//delay(2000);
c='0';
chaineSMS=String ();
      }
     }
      // Mise à zéro du compteur temps
      dateDernier1 = dateCourante;
  }

  //"surveillance  de la pendule" toutes les 5 mn pour les mesures sur capteurs
  dateCourante = millis();
    if(dateCourante - dateDernier2  >= 300000)
    {
  // Lire signal capteur de température et tester la valeur
  DS18B20Temperature = getTemperatureDS18b20();
  Serial.print(" Temperature  : ");
Serial.println( DS18B20Temperature);
// Tester dépassement seuils de température. Ajouter au contenu du message SMS.
// Vérifie que la mesure est différente de zéro(cause CRC)
 if(DS18B20Temperature >1)
 {
// si température au dessus de la limite supérieure
  if(DS18B20Temperature>temperatureLimiteMax)
  {
// mettre message d'alerte dans chaineSMS1
 chaineSMS1=String ("Alerte temperature haute; ");
// Ajouter chaineSMS1 dans chaineSMS
  chaineSMS+=chaineSMS1;
  }
// si température en dessous de la limite inférieure
  if(DS18B20Temperature<temperatureLimiteMin)
  {
// mettre message d'alerte dans chaineSMS1
  chaineSMS1=String ("Alerte temperature basse; ");
//Ajouter chaineSMS1 dans chaineSMS
  chaineSMS+=chaineSMS1;
  }
 }

// Lire signal débitmètre1 (arrivée d'eau bassin à truite) et tester
```

```
debit1= getValeurDebitmetre1();
Serial.print(" Debit1 :  ");
Serial.println(debit1);
// Tester débit sur débitmètre1 puis  ajouter  contenu message SMS ;
// si le débit est inférieure à 1 litre par minute
if(debit1<1)
 {
// mettre message d'alerte dans StringSMS1
 chaineSMS1=String ( " Alerte debit bassin nul; ");
// Ajouter chaineSMS1 dans chaineSMS
 chaineSMS+=chaineSMS1;
 }

// Lire signal des capteurs sur les deux pompes à air et analyser les résultats
  mesureVibration1=getcapteurVibrations();
    delay(50);
    totalMesureVibration1= totalMesureVibration1 + mesureVibration1;
    totalMesureVibration1Bis= totalMesureVibration1 + mesureVibration1;
    mesureVibration2=getcapteurVibrations();
    totalMesureVibration2= totalMesureVibration2 + mesureVibration2;
    totalMesureVibration2Bis= totalMesureVibration2 + mesureVibration2;
    nombreMesureVibration = nombreMesureVibration + 1;
     Serial.print("Total Mesure Vibration1 : ");
    Serial.println(totalMesureVibration1);
    Serial.print("Total Mesure Vibration2 : ");
     Serial.println(totalMesureVibration2);
     Serial.print ("nombre mesure sonde vibration : ");
     Serial.println (nombreMesureVibration);
     Serial.println ("");
 if(nombreMesureVibration==4)
 {
    if (totalMesureVibration1 == 0)
    {
      //mettre message d'alerte dans chaineSMS1
      chaineSMS1=String ( " Alerte airlift 1 defectueux; ");
      //Ajouter chaineSMS1 dans chaineSMS
      chaineSMS+=chaineSMS1;
     }
    if (totalMesureVibration2 == 0)
    {
       //mettre message d'alerte dans chaineSMS1
      chaineSMS1=String ( " Alerte airlift 2 defectueux; ");
       //Ajouter chaineSMS1 dans chaineSMS
      chaineSMS+=chaineSMS1;
     }
    nombreMesureVibration=0;
    totalMesureVibration1 = 0;
    totalMesureVibration2 = 0;
 }
```

```
// Tester contenu du message SMS et envoyer message créé
if(chaineSMS!=""  && nombreSMS<4)//Pour un message non vide et  moins de 4  SMS envoyés
  {
  Serial.println(chaineSMS);
// Envoyer le SMS
 sendSMS();
  nombreSMS = (nombreSMS + 1);// incrémenter le nombre de SMS envoyés
  }
// Téléphoner si nombre de message =4
if (nombreSMS==4)
{
// Appeler portable
appelTelephonique();
 nombreSMS = (nombreSMS + 1);// incrémenter le nombre de SMS envoyés
}

// Remettre les  messages à zéro
chaineSMS1=String ();
chaineSMS=String ();
dateDernier2 = dateCourante;
}
}
```

// Fonction acquisition température

```
float getTemperatureDS18b20()
{
byte i;
byte data[12];
byte addr[8];
float temp =0.0;
delay(5000);
//Il n'y a qu'un seul capteur, donc on charge l'unique adresse.
ds.search(addr);
// Cette action sert à surveiller si la transmission s'est bien passée
if (OneWire::crc8( addr, 7) != addr[7])
  {
  Serial.println("getTemperatureDS18b20 : <!> CRC is not valid! <!>");
  return false;
  }
// On vérifie que l'élément trouvé est bien un DS18B20
if (addr[0] != DS18B20_ID)
  {
  Serial.println("L'équipement trouvé n'est pas un DS18B20");
  return false;
  }
// Demander au capteur de mémoriser la température et lui laisser 850ms pour le faire (voir datasheet)
ds.reset();
ds.select(addr);
   ds.write(0x44);
   delay(850);
```

```
// Demander au capteur de nous envoyer la température mémorisée
ds.reset();
ds.select(addr);
ds.write(0xBE);
// Le MOT reçu du capteur fait 9 octets, on les charge donc un par un dans le tableau data[]
for ( i = 0; i < 9; i++)
  {
  data[i] = ds.read();
  }
// Puis on convertit la température (*0.0625 car la température est stockée sur 12 bits)
temp = ( (data[1] << 8) + data[0] )*0.0625;
//Serial.print("(DS18B20) =>\t temperature: ");
//Serial.println(temp);
return temp;
}

//Fonction " Compteur de mesures"
void pulseCounter()
{
pulseCount++;
}
```

//Fonction "Envoi de SMS"

```
void sendSMS()
{
//Commande AT pour mettre le SIM900 en mode SMS
SIM900.print("AT+CMGF=1\r");
delay(100);
// Utiliser la numérotation internationale (+33 et sans le 0)
SIM900.println("AT + CMGS = \"+33XXXXXXX\"");
delay(100);
// Acquisition du Message SMS à envoyer
SIM900.println(chaineSMS);
delay(100);
// Fin de la commande AT avec un  ^Z, ASCII code 26
SIM900.println((char)26);
delay(100);
SIM900.println();
// Donner du temps pour envoyer le SMS
delay(20000);
// Reset la SIM900
SIM900.println("ATZ\r");
updateSerial();
//delay(2000);
}
```

//affichage communication SIM900 avec Arduino

```
void updateSerial()
{
  delay(500);
```

```
 while (Serial.available())
 {
//Forward what Serial received to Software Serial Port
     SIM900.write(Serial.read());
   }
   while(SIM900.available())
   {
  //Forward what Software Serial received to Serial Port
     Serial.write(SIM900.read());
   }
 }

  // Lire signal débitmètre1 (arrivée d'eau bassin à truite)
 float getValeurDebitmetre1()
 {
 /*Comptage pendant 1 seconde*/
 for (int i=0; i <= 2; i++)// Boucle à deux itérations car la première mesure de débit est fausse
   {
   delay(1000);
   if((millis() - oldTime) > 1000)
  {
  /* Désactivez l'interruption lors du calcul du débit et de l'envoi de la valeur à l'hôte*/
  detachInterrupt(sensorInterrupt);
  /*Parce que cette boucle peut ne pas se terminer à exactement 1 seconde d'intervalle, nous calculons
   le nombre de millisecondes qui se sont écoulées depuis la dernière exécution.
   Nous appliquons également le calibrationFactor pour mettre à l'échelle la sortie
  basée sur le nombre d'impulsions par seconde par unité de mesure (litres / minute en
  ce cas) provenant du capteur.*/
  debit1 = ((1000.0 / (millis() - oldTime)) * pulseCount) / calibrationFactor;
  /* Notez l'heure à laquelle ce traitement a été exécuté. Notez que parce que nous avons
   désactivé "interrompt", la fonction millis () ne sera pas incrémentée correctement
  à ce stade, mais il renverra toujours la valeur à laquelle il était défini
   juste avant la fin des interruptions.*/
  oldTime = millis();
  /* Divisez le débit en litres / minute par 60 pour déterminer le nombre de litres
  passé à travers le capteur dans cet intervalle de 1 seconde, puis multiplier par 1000 pour
  convertir en millilitres.*/
  flowMilliLitres = (debit1 / 60) * 1000;// Utile uniquement pour un calcul de quantité écoulée
  // Imprimer le débit obtenu pendant cette seconde, exprimé  en litres / minute
  Serial.print(" Mesure Débit: ");
  Serial.print(int(debit1));// Affiche la partie entière de la variable
  Serial.println(" L/min ");
  // Réinitialiser le compteur d'impulsions afin que nous puissions recommencer à incrémenter
  pulseCount = 0;
  //Activer à nouveau l'interruption maintenant que nous avons fini d'envoyer la sortie
  attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
  }

 }
 return debit1;
```

```
}
// fonction lecture du signal de la sonde vibrations
float getcapteurVibrations()
{
mesureVibration1Temp=0;
mesureVibration2Temp=0;
for(int i=0 ;i<10; i++)
{
delay(3000);
// pulseIn lit la durée d'une impulsion sur capteur 1
mesureVibration1=pulseIn (BrocheCapteur1, HIGH);
mesureVibration1Temp=mesureVibration1Temp+mesureVibration1;
// pulseIn lit la durée d'une impulsion sur capteur 2
mesureVibration2=pulseIn (BrocheCapteur2, HIGH);
mesureVibration2Temp=mesureVibration2Temp+mesureVibration2;
}
mesureVibration1=mesureVibration1Temp;
mesureVibration2=mesureVibration2Temp;
return mesureVibration1;
return mesureVibration2;
}

//fonction appel téléphonique
void appelTelephonique()
{
//  changer XX avec N° de téléphone sans le 0 de départ
  SIM900.println("ATD+ +33XXXXXXX;");
  updateSerial();
  delay(20000); // attendre 20 seconds...
  SIM900.println("ATH"); //hang up
  updateSerial();
}

/* Fin du programme */
```