Écrire des données sur une carte SD

Arduino peut récupérer des données à l'aide de capteurs (humidité, température, intensité lumineuse, etc.). Il est intéressant d'avoir les données dans le moniteur série, mais si on veut pouvoir prendre des données sans être connecté à un ordinateur, nous aurons besoin d'un blindage nommé «data logging shield». Ce blindage (carte supplémentaire qui se connecte directement sur la carte Arduino) permet d'écrire des données sur une carte SD, donc de pouvoir créer un montage autonome. Une horloge interne, alimentée par une pile, permet de garder l'heure.

http://recitmst.qc.ca/arduino/ecrire-des-donnees-sur-une-carte-sd/

Gestion de l'horloge

Afin de lier les données du capteur à l'heure de la mesure, le blindage utilisé nous offre la possibilité d'utiliser une horloge interne. Voir les liens suivants pour plus de détails :

- http://tiptopboards.free.fr/arduino_forum/viewtopic.php?f=5&t=68
- http://www.worldofgz.com/electronique/gerez-lheure-date-arduino-rtc/

En résumé, nous avons besoin de télécharger la librairie RTClib, et de se faire un petit code (pris dans un lien ci-haut) comme celui-ci :

```
//Mise à l'heure RTC http://www.worldofgz.com/electronique/gerez-lheure-date-arduino-rtc/

#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 RTC; //Classe RTC_DS1307
void setup () {
 Serial.begin(57600); //Démarrage de la communication
 Wire.begin(); //Démarrage de la librairie wire.h
 RTC.begin(); //Démarrage de la librairie RTClib.h

 //Si RTC ne fonctionne pas
 if (! RTC.isrunning()) {
 Serial.println("RTC ne fonctionne pas !");

 //Met à l'heure à date à laquelle le sketch est compilé
 RTC.adjust(DateTime(__DATE__, __TIME__));
 //Cela fonctionne également :
 //RTC.adjust(DateTime("Dec 5 2012","12:00:00"));
```

```
 //RTC.adjust(DateTime(2014, 1, 21, 3, 0, 0));
 }
 }
void loop() {
 //Affichage de l'heure
 DateTime now = RTC.now();
 Serial.print(now.day(), DEC);
 Serial.print('/');
 Serial.print(now.month(), DEC);
 Serial.print('/');
 Serial.print(now.year(), DEC);
 Serial.print(' ');
 Serial.print(now.hour(), DEC);
 Serial.print(':');
 Serial.print(now.minute(), DEC);
 Serial.print(':');
 Serial.print(now.second(), DEC);
 Serial.println();
 delay(3000);
}
```

Ouvrir le moniteur série pour visualiser les données.

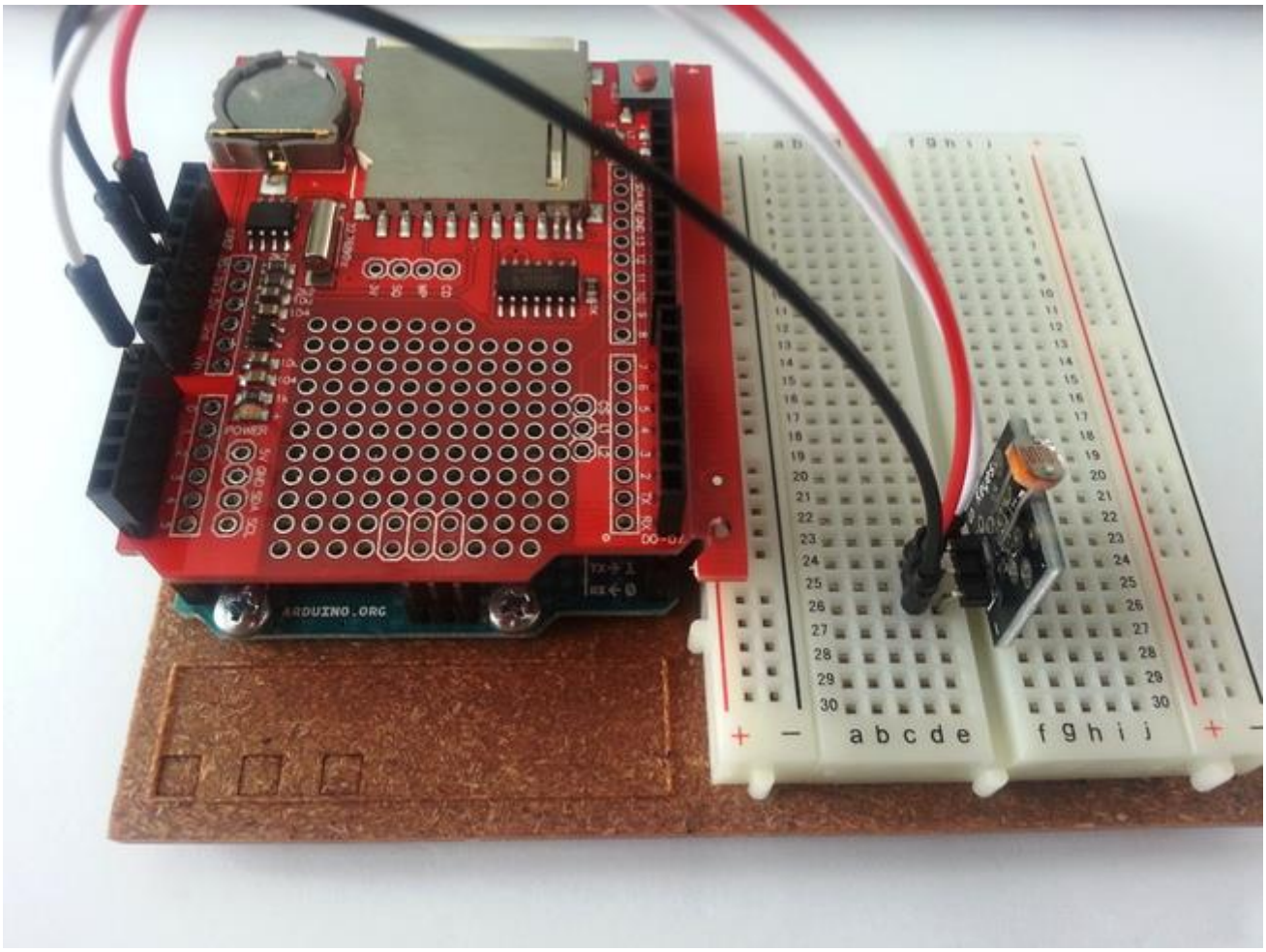
Gestion de la carte SD

Maintenant qu'on peut gérer l'heure, voyons comment inscrire nos données sur la carte SD.

Les détails dans les liens suivants:

- http://www.worldofgz.com/electronique/ecrire-et-lire-des-donnees-sur-une-carte-sd/
- http://tiptopboards.free.fr/arduino_forum/viewtopic.php?f=5&t=68

En résumé, nous devons avoir des données à inscrire, voici le montage utilisé (un capteur d'intensité lumineuse sur l'entrée analogique A0).



Ensuite, pour le programme, nous devons inclure deux librairies (déjà présente dans l'IDE d'Arduino), soit SPI.h et SD.h. Ensuite, après l'activation de la communication, nous devons ouvrir le fichier, écrire une ligne dans le fichier, fermer le fichier. Voici le code exemple :

```
/* Lien vers le code http://www.worldofgz.com/electronique/ecrire-et-lire-
des-donnees-sur-une-carte-sd/
Essais écriture carte SD
Gloup - 05/05/2015
www.worldofgz.com
*/
#include <SPI.h>
#include <SD.h>

File fichierSD;
int sensorValue;
void setup() {
 Serial.begin(57600);

 //Initialisation de l'instance
 if(!SD.begin(10)) {
 Serial.println(F("Initialisation impossible !"));
 return;
 }
 Serial.println(F("Initialisation OK"));
}
void loop() {
 sensorValue = analogRead(A0);
```

```
 Serial.println(sensorValue);

 //Ouverture du fichier
 fichierSD = SD.open("analog.txt", FILE_WRITE);

 //Test pour écriture
 if(fichierSD) {
 Serial.println(F("Ecriture en cours"));
 //Ecriture
 fichierSD.println(sensorValue);
 fichierSD.close();
 }

 delay(1000);
}
```

Un fichier analog.txt est créé sur la carte, et les valeurs d'intensité lumineuse y sont inscrites tant que le montage est alimenté.

Création d'un fichier CSV

Le format CSV est intéressant, car il peut être ouvert facilement dans un tableur pour créer des graphiques, etc. Voici le code (commenté et expliqué plus bas) que nous utilisons pour notre premier fichier CSV contenant des données datées.

```
/*Exemple de code pour prendre des données et les écrire sur une carte SD
RÉCIT MST 2016
Inspiration http://www.worldofgz.com/electronique/ecrire-et-lire-des-
donnees-sur-une-carte-sd/
http://tiptopboards.free.fr/arduino_forum/viewtopic.php?f=5&t=68
http://forum.arduino.cc/index.php?topic=107367.0
*/
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include "RTClib.h"

RTC_DS1307 rtc;

File fichierSD;
int sensorValue;

void setup() {
 Serial.begin(9600);
//Pour la carte SD=====================
 //Initialisation de l'instance
 if(!SD.begin(10)) {//10 pour notre carte, peut etre different
 Serial.println(F("Initialisation impossible !"));
 return;
 }
 Serial.println(F("Initialisation OK"));
//Fin carte SD=========================

//Pour l'horloge=======================
 #ifdef AVR
 Wire.begin();
 #else
 Wire1.begin(); // Shield I2C pins connect to alt I2C bus on Arduino Due
```

```
 #endif
 rtc.begin();

 if (! rtc.isrunning()) {
 Serial.println("RTC ne fonctionne PAS!");
 // La ligne qui suit ajuste le RTC à la date et time du moment de
compilation(si connectee a ordinateur)
 rtc.adjust(DateTime(__DATE__, __TIME__));
 }
//Fin horloge============================
}

void loop() {
 sensorValue = analogRead(A0);//Le capteur est ici en A0

 DateTime now = rtc.now();

//On envoie dans le moniteur série l'info écrite
 Serial.print(now.year(), DEC);
 Serial.print('/');
 Serial.print(now.month(), DEC);
 Serial.print('/');
 Serial.print(now.day(), DEC);
 Serial.print(' ');
 Serial.print(now.hour(), DEC);
 Serial.print(':');
 Serial.print(now.minute(), DEC);
 Serial.print(':');
 Serial.print(now.second(), DEC);
 Serial.print(";");
 Serial.println(sensorValue);

//On nomme le fichier selon la date, donc toutes les donnees prises le meme
jour dans un seul fichier==========
 char datafile[13]; //tableau de 13 espaces pour le nom
 int jour=now.day();
 int mois = now.month();
 int annee= now.year();
 sprintf(datafile,"%02d%02d%04d.csv",jour,mois,annee); // %d pour un int
 datafile[13]='\0';//à mettre pour fermer convenablement le fichier
//Fin nommer fichier==================

//Ouverture du fichier=========================
 fichierSD = SD.open(datafile, FILE_WRITE);

 //Test pour écriture
 if(fichierSD) {
 Serial.println(F("Ecriture en cours"));
 //Ecriture
 maintenant();
 fichierSD.print(";");
 fichierSD.println(sensorValue);
 fichierSD.close();
 }
 //Fermeture du fichier avec une ligne de plus============

 delay(1000);//durée en ms entre les mesures
}

//Fonction qui écrit la date et l'heure dans le fichierSD
void maintenant() {
```

```
 DateTime now = rtc.now();

 fichierSD.print(now.year(), DEC);
 fichierSD.print('/');
 fichierSD.print(now.month(), DEC);
 fichierSD.print('/');
 fichierSD.print(now.day(), DEC);
 fichierSD.print(' ');
 fichierSD.print(now.hour(), DEC);
 fichierSD.print(':');
 fichierSD.print(now.minute(), DEC);
 fichierSD.print(':');
 fichierSD.print(now.second(), DEC);
}
```

Explication du code

À l'aide des commentaires présents dans le code, vous pouvez remarquer que nous utilisons un petit bout de code pour nommer le fichier selon la date (jour+mois+année), ce qui simplifiera la récupération des données après plusieurs utilisations de votre robot.

Ensuite, nous avons créé une fonction (maintenant) pour écrire la date+heure dans le fichier, ça rend le code plus lisible.

Selon le projet, vous pouvez ajuster le temps entre 2 mesures, ainsi que les données mises dans le fichier (on sépare les données avec des points-virgules « ; »).