Projet PERI :
Dispositif météorologique

Rapport de projet

Paul MABILLOT & Adrien FERREIRA & Pierre MAHÉ

1er mai 2015

UPMC
PARIS UNIVERSITAS

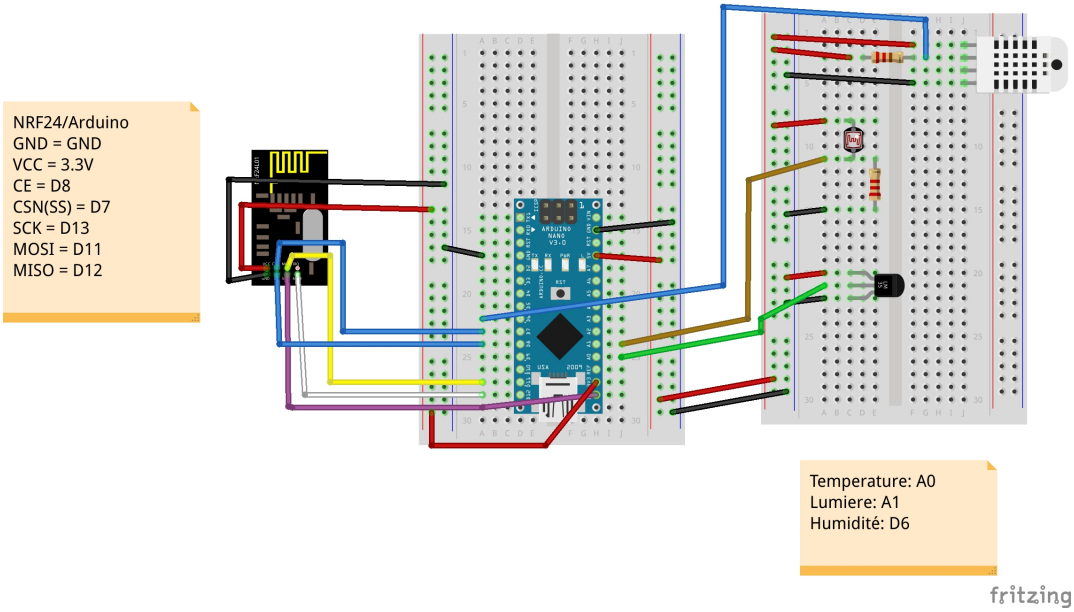
Table des matières

1 Description du projet

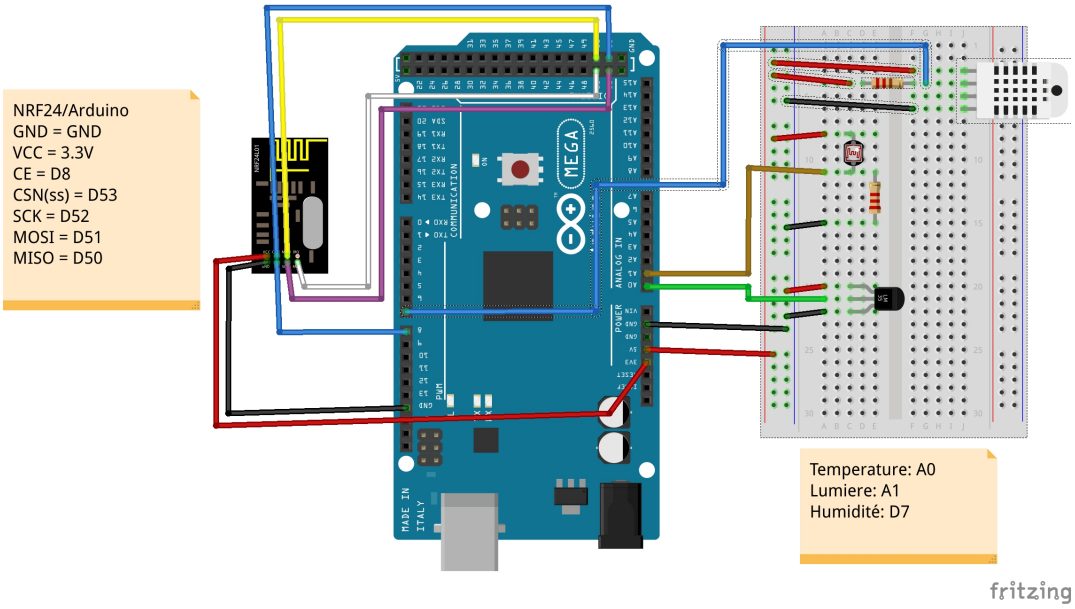
2 Montage

2.1 Arduino

Faire le montage suivant votre carte Arduino.
Pour les cartes Nano et One les branchements sont similaire. À noter que nous n'utilisons pas de led externe mais celle intégrée au circuit.

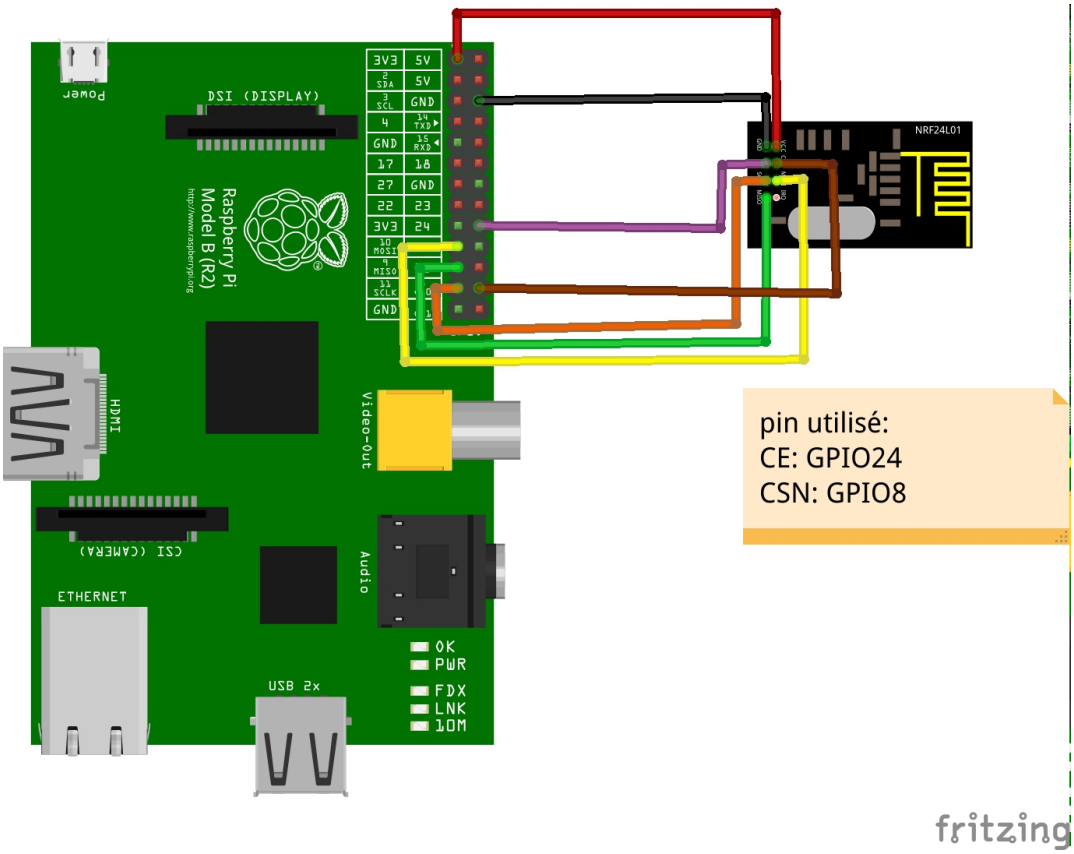


Pour l'Arduino Méga le montage est un peu différent. Branchez le pin CSN(SS) au pin D53, le SCK au pin D52, le MOSI au pin D51 et le MISO au pin D50.



2.2 Raspberry Pi

Faire le montage suivant sur votre carte Raspberry Pi.
Dans cet exemple, nous avons utilisé une Raspberry Pi modèle B mais pour les modèles B+ et 2 la rétrocompatibilité est assurée, il suffit de procéder au même branchement que le schéma.



Attention : Le module nRF24 est fragile, il n'accepte qu'une alimentation 3.3V, ne surtout pas le brancher sur le pin 5V au risque de l'endommager.

3 Codage

3.1 Structure de données

Cette structure est commune aux deux cartes, elle permet de structurer les données envoyées.

```
typedef struct t_payload{
  uint16_t cmd;
  uint16_t timeStamp;
  uint16_t valLight;
  uint16_t valTemperature;
  uint16_t valHumidity;
} payload;
```

Pour garantir l'interopérabilité, il est impératif de définir explicitement la taille des types manipulés. **Attention : certains compilateurs rajoutent des octets de bourrage pour aligner la taille de la structure en mémoire. Veillez donc à accorder les options des deux compilateurs (Arduino et Raspberry).** L'option `packed` peut être utilisée comme directive pour le compilateur.

L'endianess de l'Arduino et la Raspberry étant identique il n'a pas été nécessaire d'opérer à des conversions, attention cependant au portage sur d'autres systèmes. `cmd` utile pour définir le type de traitement à réaliser (cf : Protocole). Les autres champs sont présents pour stocker les données à proprement parler.

3.2 Arduino

Pour envoyer les données à intervalle régulière, vous devez installer la bibliothèque *SimpleTimer* (téléchargeable sur le site officiel Arduino).
Il faut tout d'abord armer un timer avec un temps (en milliseconde) et un pointeur de fonction (sans paramètre) dans le `setup`. Dans le `loop`, appelez la fonction `run`.

```
SimpleTimer timer;
timer.setInterval(DELAY, fctnCallback);
timer.run()
```

Pour communiquer avec le module nRF24L01+, vous devez installer une seconde bibliothèque. *lien du dépôt*. Seul les fichiers `RF24.cpp` et `RF24.h` sont à copier dans le dossier `librairies` de l'IDE sketch.

3.2.1 Initialisation

La séquence d'instruction à utiliser pour initialiser le module nRF24 est :

```
radio.begin();
radio.setPALevel(RF24_PA_MAX);  // la force d'emission du signal
radio.setDataRate(RF24_1MBPS);  // le debit de donnees

// Identifiant arbitraire du canal de emission
radio.openWritingPipe(0x0000000001LL);
// Identifiant du canal de reception pour le pipe 1
radio.openReadingPipe(1, 0x0000000002LL);
```

Attention : Il n'est possible d'écouter que sur 6 canaux simultanément. Attention à correctement synchroniser les canaux utilisés par les nœuds. Si deux modules écrivent sur le même canal, des collisions vont apparaître.

3.2.2 Lecture

Avant de pouvoir envoyer des données grâce au module nRF24, il faut l'activer en lecture grâce à l'instruction : `startListening()`. Pour savoir si une donnée est en attente dans le buffer de réception, il faut utiliser l'instruction `available()` non-bloquante. Pour récupérer la valeur, il faut utiliser `read`.

3.2.3 Écriture

Il n'est pas possible d'écouter et d'émettre simultanément. De ce fait, avant une écriture, il faut prendre soin d'appeler `stopListening()` pour arrêter l'écoute. Ensuite, utilisez `write()` pour envoyer les données. Ne pas oublier de réactiver l'écoute ensuite.

3.3 Raspberry Pi

Pour communiquer avec la module nRF24L01+ depuis la Raspberry, utilisez les fonctions présentes dans les fichiers `RF24.cpp`, `RF24.h`, `bcm2835.c` et `bcm2835.h`.

3.3.1 Initialisation

La phase d'initialisation est similaire à celle de l'Arduino.

```
/* les deux premiers arguments sont les pins GPIO a utiliser pour CE
   et CS (definie BCM2835.h) la cadence du pin SPI.*/
RF24 radio(BCM2835_SPI_CS_GPIO24, 800000, BCM2835_SPI_SPEED_8MHZ);
radio.begin();
radio.setPALevel(RF24_PA_MAX);     // la force d'emission du signal
radio.setDataRate(RF24_1MBPS);     // le debit de donnees

// Identifiant du canal de reception pour le pipe 1
radio.openReadingPipe(1,0x0000000001LL);
// Identifiant arbitraire du canal de emission
radio.openWritePipe(0x0000000002LL);

radio.startListening();
```

3.3.2 Lecture, écriture

La manière de procéder et les fonctions à appeler sont les mêmes que pour l'Arduino.

4 Protocole de niveau applicatif

4.1 Communication Arduino - Raspberry

Pour pouvoir synchroniser les horloges des deux cartes, nous avons créé un protocole simple. Au démarrage l'Arduino envoie un message avec `cmd = INIT` et se met en attente de la réponse de la Raspberry. Quand elle reçoit ce paquet la Raspberry répond avec son estampille temporelle dans `timestamp` et `cmd = INIT`. L'Arduino sauvegarde cette valeur et l'utilisera pour dater ses envois (en l'incrémentant). De ce fait, si la carte Arduino venait à s'éteindre, elle cherchera à se resynchroniser au démarrage. Suite à cela, l'Arduino pourra envoyer ses valeurs de capteur avec `cmd = DATA` et les autres champs remplis en conséquence.

4.2 Stockage des données dans la Raspberry

Pour Stocker les données reçu la Raspberry Pi, notre programme écrit dans un fichier `XML`.
Dès que nous recevons un message, on écrit le timestamp et les valeurs associer.
Attention à la lecture concurrente (entre le programme C et le programme PHP), pensez à mettre des verrous lors de l'écriture et la lecture du fichier.

Structure du fichier :

```
<meteo>
  <!-- Premier echantillon -->
  <timestamp valeur="13"> <!-- avec  le temps en seconde -->
    <luminosite>15</luminosite> <!-- La valeur des capteurs -->
    <temperature>20</temperature>
    <humidite>15</humidite>
  </timestamp>
  <!-- Second echantillon -->
  <timestamp valeur="14">
    <luminosite>20</luminosite>
    <temperature>25</temperature>
    <humidite>10</humidite>
  </timestamp>
</meteo>
```

Cette structure permet de faciliter les traitements et d'être facilement extensible si vous voulez ajouter un autre capteur.

5 Serveur

5.1 Installation

-Installation Apache2 et PHP

```
sudo apt-get install apache2 php5 libapache2-mod-php5
```

-Installation Mysql (inutile dans notre cas)

```
sudo apt-get install mysql-server php5-mysql
```

-Répertoire du serveur

```
/var/www/
```

-Commandes Serveur :

```
$service apache2 stop : arreter serveur
$service apache2 start : demarrer serveur
$service apache2 restart : redemarrage serveur
```

5.2 Mise en place

5.2.1 Page d'accueil

Dans un premier temps nous allons créer la page d'accueil de notre station météo. Nous souhaitons y afficher les informations en temps réel ainsi qu'un graphique regroupant la température et l'humidité des trois derniers jours.

Dans la section *"station"* nous afficherons les données
Dans la section *"recap"* nous afficherons le graphique

```
<html>
  <head><title>Site meteorologique</title></head>

  <body>
  <section id="meteo">
    <section id="station"></section>

    <section id="recap"></section>
  </section>
  </body>
</html>
```

Notre Raspberry ne sera pas forcement connectée à internet, il est donc important d'importer les librairies pour assurer son bon fonctionnement, même sans connexion internet.
Pour cela que nous importons la bibliothèque jQuery et jqPlot. Le code suivant est à ajouter à la fin du *<head>*.

La librairie jQuery :

```
<script src="js/jquery-1.11.2.min.js"></script>
```

La libraire de gestion de graphique jqPlot téléchargeable sur le site officiel :

```
<script type="text/javascript" src="js/jqPlot/jquery.jqplot.min.js"
  ></script>
<script type="text/javascript" src="js/jqPlot/plugins/jqplot.
  canvasTextRenderer.min.js"></script>
<script type="text/javascript" src="js/jqPlot/plugins/jqplot.
  canvasAxisLabelRenderer.min.js"></script>
```

5.2.2 Lecture de fichier

Les données envoyées par votre Arduino vers votre Raspberry (voir plus haut) sont sauvegardés dans le fichier *xml/fichier.xml* de votre serveur Apache. Nous allons écrire une fonction PHP permettant de récupérer la dernière valeur ajoutée dans ce fichier.

```
function getLastValue($path){
  $fp = fopen($path, "r");

  if (flock($fp, LOCK_SH)) {
    $xml = simplexml_load_file($path);
    flock($fp, LOCK_UN);
    $taille = sizeof($xml);
    $last = $xml->timestamp[$taille-1];
  }
  return $last['valeur'].' '.$last->temperature.' '.$last->humidite.'
    '.$last->luminosite;
}
```

La fonction *getLastValue*, qui prend en paramètre le chemin vers votre fichier XML, s'occupe d'ouvrir ce fichier en lecture seul.
Un verrou est alors placé pour éviter une écriture durant la lecture.
La fonction `simplexml_load_file` s'occupe de parser un fichier XML et retourne un objet facilement navigable.

Une fois l'objet obtenu on relâche le verrou sur le fichier.
La fonction retourne alors le timestamp, la température, l'humidité et la luminosité du dernier envoi.

5.2.3 Rafraîchissement automatique

Afin de rendre notre page dynamique nous souhaitons que les dernières valeurs envoyées par nos sondes soient directement affichés. Pour cela nous allons utiliser notre fonction PHP grâce à un appel Ajax toute les 1000ms.

```
function loop() {
  setTimeout(function(){
    $.ajax({
      type    : 'GET',
      url     : 'getLast.php',
      success: function(msg) {
        /* Traitement affichages */
      }
    });
    loop();
  }, 1000);
});
```

Le traitement des images est fait par la fonction `traitementLumino`.

— Lors du traitement, si l'humidité est supérieure à 50% alors il pleut.

— Si il est entre 21h et 7h, alors il fait nuit.

— Si la luminosité est forte, alors il fait soleil.

— Si la luminosité est moyenne, alors il fait légèrement couvert.

— Si la luminosité est faible, alors il fait sombre.

5.2.4 Le graphique des données

Nous affichons les données des 3 derniers jours sur notre graphique. Pour cela nous savons que chaque informations est reçu toute les 60s, il faut donc traiter 1440 événements par jour, soit dans notre cas 4320 événements (60*24*3) pour les trois derniers jours.

Contrairement à la récupération de données précédente, le graphique n'est généré qu'une fois, lors du rafraîchissement de la page par le navigateur web. Celle-ci étant trop longue pour la rendre dynamique.

La génération du graphique ce fait grâce à la fonction `jqplot` de la librairie `jqPlot` dont l'appel est présent dans le fichier `loadGraphe.php`. Dans ce fichier nous récupérons les données via PHP. Le javascript les traite pour permettre la génération.

```
1 <?php
2   include("gestionFile.php");
3   /* Recuperer les 4320 derniers timestamp dans le fichier XML */
4   $res = getAllValues("xml/fichier.xml");
5   $tabjs = php2js($res);
6 ?>
```