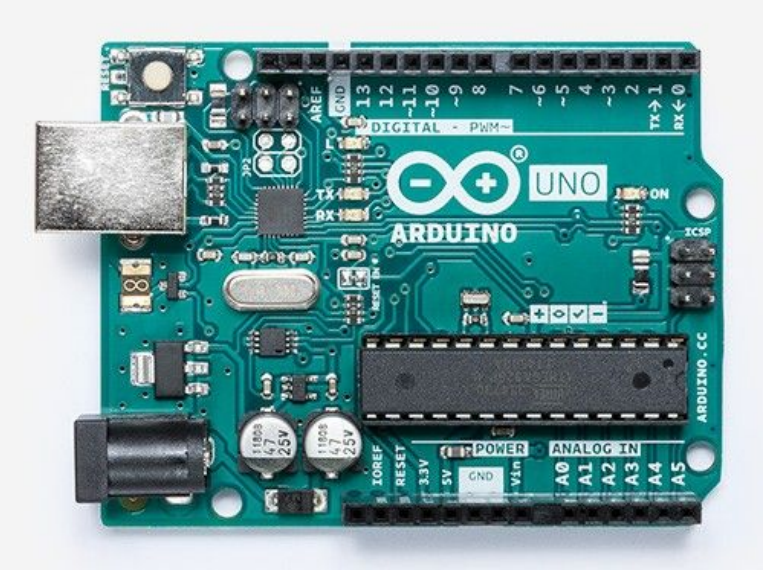
Cédric COGNARD
Paul VAN-ELSUE

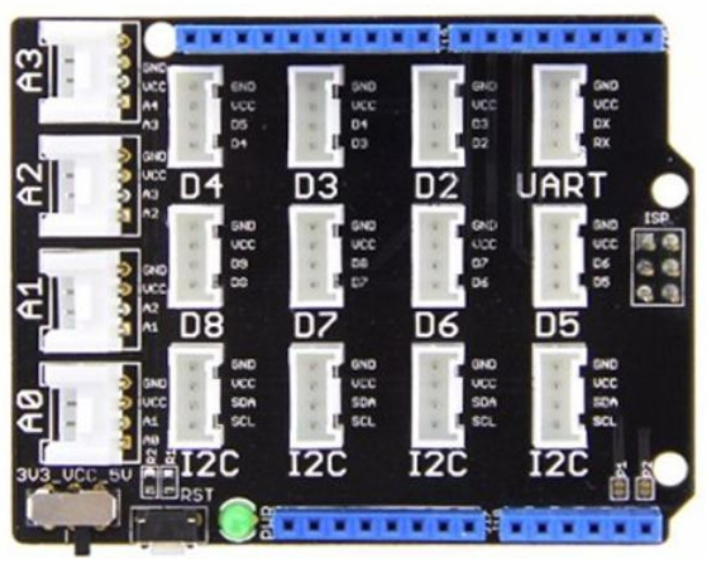
Rapport M2M

Cédric COGNARD
Paul VAN-ELSUE

Introduction



Pour ces TPs, nous allons utiliser une carte Arduino Uno Rev 3. Elle est architecturée autour d'un microcontrôleur Atmega328P de la société ATMEL. Ce microcontrôleur peut être couplé à un Shield Grove qui vient se plugger dessus : c'est ce que nous allons utiliser. Cette carte d'interface permet de faciliter la mise en oeuvre matérielle de périphériques d'E/S sur la carte Arduino.



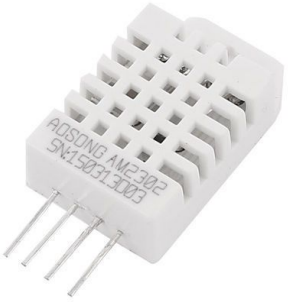
Dans ce TP nous allons apprendre à utiliser différents composants comme des capteurs, des led, des écrans, etc. De plus, nous allons créer un service web qui pourra communiquer avec notre microcontrôleur et donc afficher ses informations ainsi d'interagir avec ses composants.

La carte se programme en Arduino, qui est un langage fait pour la carte, un mélange de C et de C++.

1. Prise en main Arduino

a. Capteur température et humidité

Dans un premier temps nous allons utiliser le capteur **DHT22** qui va nous donner la température et l'humidité ambiante.



Ce composant s'utilise avec très peu de lignes de code, en effet il suffit de déclarer l'objet `DHT dht(DHT_PIN, DHTTYPE)` tout en haut de notre code et de placer un `dht.begin()` dans notre fonction `setup()` afin de pouvoir observer les valeurs captées.
Ainsi nous pouvons placer les mesures captées dans des variables de notre `loop` de la façon suivante :

```
h = dht.readHumidity();
t = dht.readTemperature();
f = dht.readTemperature(true);
```

b. Ecran LCD

L'écran LCD se branche sur un port **I2C** de notre microcontrôleur, ici nous souhaitons afficher les mesures de notre capteur sur l'écran. Pour l'initialiser il faut, comme pour le capteur, déclarer l'objet `rgb_lcd lcd` et aller modifier la fonction `setup()`. Il est possible d'afficher un message avant d'envoyer nos données, comme nous le faisons dans cet exemple:

```
lcd.begin(16, 2);
lcd.setCursor(0, 0);
lcd.print("Let me start");
```

```
lcd.setCursor(0, 1);
lcd.print("please !");
```

Avant d'utiliser l'écran pour afficher nos valeurs nous devons vérifier qu'il n'y pas de problèmes avec les mesures, pour cela nous procédons de la façon suivante :

```
if (isnan(h) || isnan(t) || isnan(f)) {
    lcd.clear();

    lcd.setCursor(0, 0);
    lcd.print("Failed to read");

    lcd.setCursor(0, 1);
    lcd.print("from DHT sensor!");
    }
```

Si tout se passe correctement nous pouvons afficher nos valeurs à l'écran via les variables vues précédemment :

```
lcd.clear();

lcd.setCursor(0, 0);
lcd.print("Humidity: ");
lcd.print(h);
lcd.print("%");

lcd.setCursor(0, 1);
lcd.print("Temp:   ");
lcd.print(t);
lcd.print("*C");
```

De cette façon nous pouvons voir ceci sur l'écran :

c. LED RGB

Nous voulons maintenant faire en sorte que cette led s'allume d'une certaine couleur en fonction de la température ambiante. Pour cela il faut encore et toujours le déclarer en indiquant `ChainableLED led(LED_PORT_A, LED_PORT_B, 1)` et aller l'initialiser dans la fonction `setup()` de notre programme avec un `init()`.
Notre variable `t` contenant la température ambiante va nous servir ici :

```
if(t < 16)
    led.setColorRGB(0, 0, 0, 255);
else if(t > 24)
    led.setColorRGB(0, 255, 0, 0);
else
    led.setColorRGB(0, 0, 255, 0);
```

Avec ce code, notre LED s'allume en bleu si la température est inférieure à 16°C, vert si la température est comprise en 24°C et 16°C puis rouge si elle dépasse les 24°C.

2. Utilisation du capteur BME280

a. Utilisation d'une librairie

Pour continuer sur l'utilisation des capteurs nous avons utilisé un BME280. Ce capteur fait la même chose que le DHT22 mais en étant plus performant, de plus il ajoute la possibilité de capter la pression (et donc de calculer la hauteur à laquelle se trouve le capteur si on pense à bien calibrer le calcul).

Pour commencer nous avons commencé par utiliser une librairie, facilitant énormément le travail. On y déclare un objet de type BME qu'on initialise dans le setup avec la méthode init et on récupères les valeurs avec de simples fonctions :

```
BME280 bme280;
[...]
if(!bme280.init())
  Serial.println("Error with BME280 !");
[...]
bme280.getTemperature();
bme280.getPressure();
bme280.calcAltitude(bme280.getPressure());
bme280.getHumidity());
```

b. Création de sa propre librairie

Pour continuer nous avons créé notre propre libraire. En effet le code d'une librairie est très lourd et l'Arduino n'a que peu de mémoire. C'est du au fait que beaucoup de fonctions sont données et certaines sont assez inutiles pour notre cas. C'était assez simple car il suffit de faire un copier coller de tout ce qui est important en partant de ce qu'on utilise et en copiant

les dépendances. Une fois que c'est fait on libère une mémoire très importante cela nous permet de passer à la suite sans avoir de problèmes de saturation.

3. Utilisation du Shield Ethernet Arduino

La dernière partie des TPs est la plus intéressante. C'est à ce moment que nous voyons comment assembler tout le travail vu précédemment pour en faire un vrai projet. Ici, nous allons dans un premier temps mettre en place un serveur web sur l'Arduino, on utilisera ensuite notre navigateur pour accéder aux fonctionnalités que nous développerons.
Ces fonctionnalités seront simples, nous afficherons les valeurs de nos capteurs que nous actualisons régulièrement puis nous mettrons un bouton pour contrôler une LED.

a. Préparation d'un serveur web

Pour la mise en place du serveur, c'est très simple. Nous avons récupéré le code d'exemple sur la page officielle d'Arduino et nous l'avons utilisé. Tout ce qui ne nous était pas utile a été supprimé et nous n'avions plus qu'à indiquer l'IP et l'adresse MAC du shield pour l'utiliser. Une connection sur *http://address_ip/* et nous avions accès au serveur (en mettant bien sur la bonne adresse IP). Pour envoyer du code HTML il suffisait de faire comme dans l'exemple, faire un print sur l'objet client.
Nous avons rencontré beaucoup de difficulté au début car il était impossible de communiquer avec le shield, cependant après avoir changé de carte tout a fonctionné directement. Il y a donc de très fortes chances que tous les problèmes soient dus au matériel.

b. Affichage des information sur la page web

Une fois le serveur web en place nous souhaitions afficher les informations de nos capteurs. Nous avons commencé par les configurer comme précédemment et à récupérer les informations dans des variables, encore une fois comme avant. Ensuite pour les afficher sur le serveur web nous faisons un print à notre objet client mais en mettant les variables en paramètres. Ensuite nous avons optimisé ça pour mettre directement les getters dans l'affichage du client, ce qui donne ça :

```
client.print("<div>Temperature : ");
client.print(bme280.getTemperature());
client.println(" °C</div>");
```

c. Commander la LED via la page web

Pour terminer nous avons voulu commander la LED depuis notre fameuse page web. C'est à ce moment que les choses sont devenues légèrement plus difficiles. Heureusement tout s'est quand même fait rapidement.

Dans un premier temps, nous avons créé un formulaire sur notre Arduino. Nous y mettons notre bouton et nous demandons à ce qu'une constante soit envoyée en POST lors du clic. On regarde les valeurs de retour et la nous n'avons pas réussi à traiter quoique ce soit. Après de nombreux essaies et une bonne heure de perdue nous n'arrivons toujours pas à expliquer pourquoi il nous est impossible de récupérer les valeurs en POST.

Mais nous n'avons pas baissé les bras et nous avons changé de méthode ! On a essayé d'envoyer les valeurs en GET et tout de suite il nous était possible de les lire ! Nous avons ensuite parsé un peu le résultat pour ne garder que les informations que l'on voulait, et on dit que lorsque cette constante envoyée par le client est reçue, on inverse le résultat de la LED.

C'était très fonctionnel et nous étions très contents du résultat.

Cependant, cela ne nous a pas suffit. La LED est une LED RGB, on peut donc l'illuminer de la couleur que l'on souhaite, alors pourquoi ne pas laisser le choix à l'utilisateur ? Après quelques recherches sur Internet nous avons vu qu'il était facile de faire un formulaire pour les couleurs, ce qui nous a donné ce code :

```
// Add three buttons to control the output led.
client.println("<section>");
client.println("<form method=\"GET\">");
client.println("<input type=\"color\" name=\"led_control\" class=\"form-control\">");
client.println("<input type=\"submit\">");
```

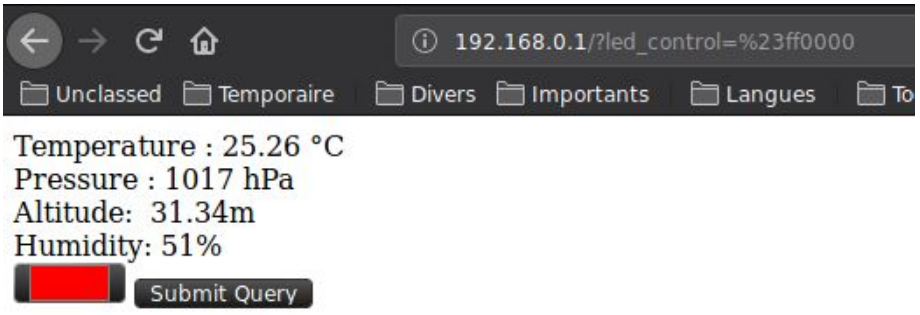
```
client.println("</form>");
client.println("</section>");
client.println("</html>");
```

Avant de récupérer le résultat nous avons un peu joué avec le formulaire et regardé comment sont envoyées les données. C'est simple, on reçoit le nom de la variable, un égal puis un valeur hexadécimale sous la forme *%xxxxxxxx* avec x étant un code héxa. Toujours en jouant avec le formulaire, nous avons vu que le rouge correspondait à *%00ff0000*, le bleu à *%0000ff00* et le vert à *%000000ff*. Les deux premiers 00 ne sont donc jamais utilisés, ni même le %. C'est probablement la transparence, nous avons donc choisi de l'ignorer. Ensuite nous récupérons donc les trois valeurs pour le RGB, que nous convertissons en décimal et nous demandons à notre LED de s'allumer sous cette couleur grâce à la librairie associée.

```
if(buffer.indexOf("led_control=") != -1) {
  color = "";

  for(int i = 0; i < 6; i++)
     color += buffer.charAt(buffer.indexOf("led_control=") + 15 + i);

  color_str =color.c_str();
  sscanf(color_str, "%02x%02x%02x", &r, &g, &b);

  led.setColorRGB(0, r, g, b);
}
```

Ce qui est génial, c'est que cette partie qui nous semblait difficile a fonctionné directement. Seule la conversion hexadécimal vers décimal nous a posé légèrement problème car nous avons souhaité le faire à la main sans résultats rapides, mais après une recherche Internet nous avons vu la fonction *sscanf* qui nous a facilité la vie.

Voici l'interface graphique finale :



Conclusion

Au final ces TPs ont été très intéressants mais aussi très amusants car cela nous a permis d'utiliser du matériel et de faire des choses plus concrètes. Très progressivement nous

avons pu utiliser une Arduino, des capteurs, communiquer avec la carte par Ethernet et contrôler une LED qui pourrait être n'importe quel actionneur. Nous pourrions très aisément remplacer ce travail pour par exemple demander à l'Arduino d'allumer une lumière d'une pièce d'une maison lorsqu'il fait sombre avec un capteur de luminosité, et afficher l'état des lumières de la maison sur une page web. Les possibilités sont immenses et très intéressantes.