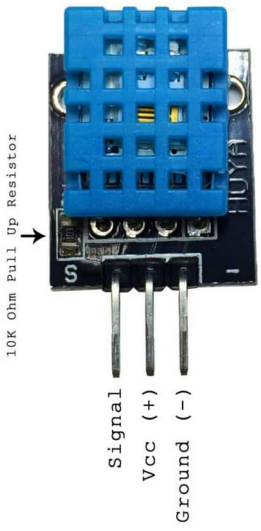
Journée 4 - TP2		
BSP UBoot et Linux embarqué, construire son propre système	*Date MAJ*	*24 août 2023*
Chiheb Ameur ABID	*Version*	*1.0*

Le capteur DHT11 est lui capable de mesurer des températures de 0 à +50°C avec une précision de +/- 2°C et des taux d'humidité relative de 20 à 80% avec une précision de +/- 5%. Une mesure peut être réalisée toutes les secondes.

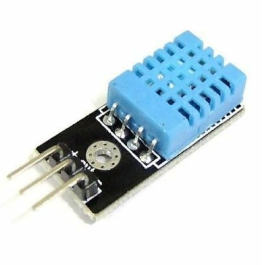
Les capteurs DHTxx communiquent avec le microcontrôleur via une unique broche d'entrée / sortie.

Le brochage du capteur est le suivant :

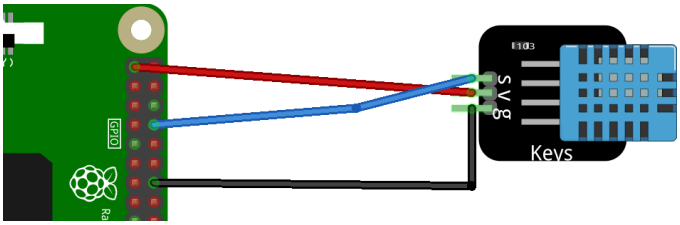
- La broche n°1 est la broche d'alimentation (5 volts ou 3.3 volts).
- La broche n°2 (S,DATA) est la broche de communication. Celle-ci doit impérativement être reliée à l'alimentation via une résistance de tirage de 4.7K ohms (il s'agit d'une sortie à collecteur ouvert).
- La broche n°3 (NC) n'est pas utilisée et ne doit pas être câblée.
- La broche n°4 est la masse du capteur (GND).



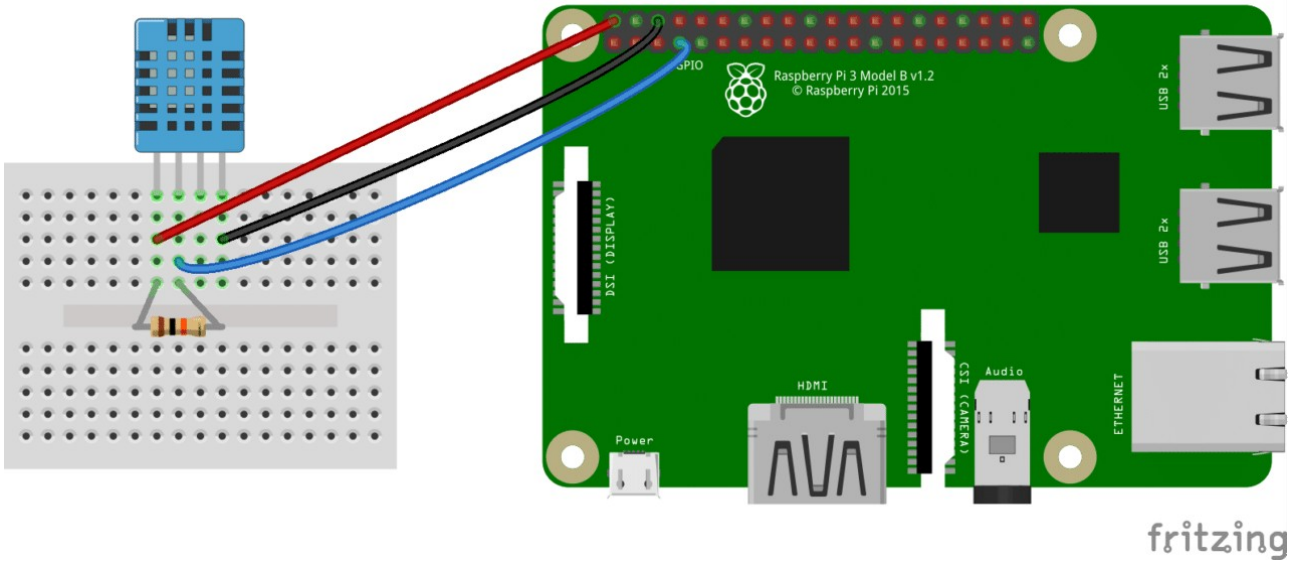
Pour le module DHT11 intégré à un PCB, les broches seront au nombre de trois, puisque le NC est retiré pour faciliter les choses. De plus, ce module inclut la résistant pull-up à la broche DATA.



Le montage à une carte Raspberry du module intégré à un PCB sera donc le suivant :



Voici un exemple de montage du module DHT11 sans PCB :



.

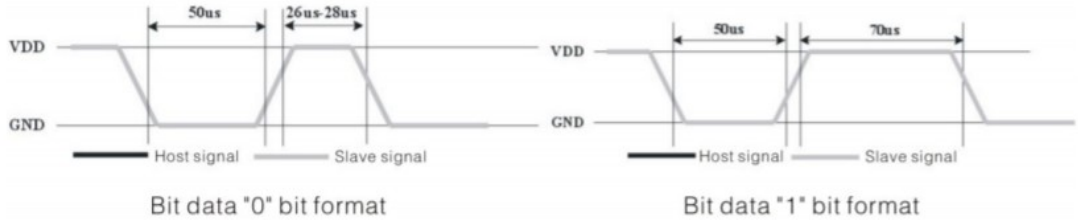
La communication avec un capteur DHT11 se fait en 3 étapes :

• Tout d'abord, le microcontrôleur maitre réveille le capteur en plaçant la ligne de données à LOW pendant au moins 18ms pour le DHT11. Durant ce laps de temps, le capteur va se réveiller et préparer une mesure de température et d'humidité. Une fois le temps écoulé, le maitre va libérer la ligne de données et passer en écoute. Le timeout pour recevoir une réponse est de 1000 us.

• Une fois la ligne de données libérée, le capteur répond au maître (pour montrer qu'il est bien réveillé) en maintenant la ligne de données à LOW pendant 80µs puis à HIGH pendant 80µs.

• Le capteur va ensuite transmettre une série de 40 bits (5 octets). Les deux premiers octets contiennent la mesure de l'humidité. Les deux octets suivants contiennent la mesure de la température et le cinquième octet contient une somme de contrôle qui permet de vérifier que les données lues sont correctes.

Le DHT envoie la valeur de l'humidité et celle de température sur 40 en série en commençant par le bit le plus signficatif. Chaque bit de données commence par le niveau de tension basse 50us et la longueur du signal de niveau de tension élevé suivant détermine si le bit de données est à "0" ou "1" comme l'illustre la figure ci-après.



Une manière pour identifier les bits 1 et 0 est la suivante :

- Si la durée de mise à 1 est inférieure à celle de mise à 0, alors il s'agit du bit '0'
- Sinon, c'est-à-dire la durée de mise à 1 est supérieure à celle de mise à 0, alors il s'agit du bit '1'

Le format d'une mesure envoyée par le capteur est illustrée par l'exemple suivant :

Example .: 40 data is received:

0011 0101	0000 0000	0001 1000	0000 0000	0100 1101
High humidity 8	Low humidity 8	High temp. 8	Low temp. 8	Parity bit

Calculate:

0011 0101+0000 0000+0001 1000+0000 0000= 0100 1101

On se propose d'écrire un driver permet de renvoyer la température et l'humidité capturées à partir du capteur DHT11.

1) Développer les fonctions suivantes :

- Réveiller le capteur en mettant la ligne data à 0 pendant 20ms, puis on remet le signal à 1

```
void SendStartSignal(); //
```

- La fonction WaitForLow() renvoie le temps d'attente en microsecondes pour que la ligne de données passe à '0' par le DHT11.

```
int WaitForLow() ;
```

- La fomction WaitForHigh() renvoie le temps d'attente en microsecondes pour que la ligne de données passe à '0' par le DHT11.


```
int WaitForHigh();
```

- Calculer la somme des 4 premiers octets reçus pour la vérification de la validité des données.

```
u8 CalculateParity(u8 HumidityHigh, u8 HumidityLow,u8 TemperatureHigh, u8 TemperatureLow) {
        return (HumidityHigh + HumidityLow + TemperatureHigh
                        + TemperatureLow);
}
```

- La fonction qui effectue la récupération des données à partir du DHT11 dans une variable codée sur 64 bits qui sera transmise à ProcessData() pour extraire la valeur de la température et de l'humidité.

```
void Measure() ;
```

- La fonction **ProcessData()**, appelée par la fonction **Measure()**, extrait la température et l'humidité à partir de la trame dans deux variables globales.

```
Void ProcessData(u64 Data);
```