Capteur SHT31

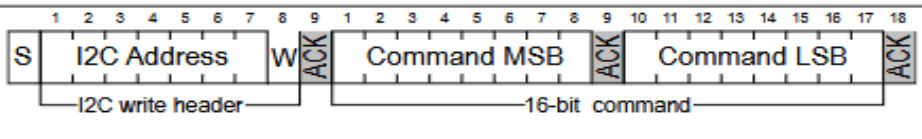
1- Spécification :

Adresse du capteur : afin de communiquer avec le capteur par le protocole I2C, son adresse est nécessaire, dans la datasheet du capteur on trouve l'adresse (0x44) :

SHT3x-DIS	I2C Address in Hex. representation	Condition
I2C address A	0x44 (default)	ADDR (pin 2) connected to logic low

Commandes : Pour définir le nombre de mesure faite par le capteur en MPS (Measurements Per Second), des commandes spécifiques sont envoyées depuis la carte STM32 vers le capteur SHT31. Dans notre cas, nous configurons le capteur pour effectuer une mesure par seconde avec une haute répétabilité. Pour cela, la commande 0x2130 est transmise au capteur.

Condition		Hex. code	
Repeatability	mps	MSB	LSB
High	0.5	0x20	32
Medium			24
Low			2F
High	1	0x21	30
Medium			26
Low			2D
High	2	0x22	36
Medium			20
Low			2B
High	4	0x23	34
Medium			22
Low			29
High	10	0x27	37
Medium			21
Low			2A
e.g. 0x2130: 1 high repeatability mps - measurement per second			



Durée de la mesure : Le temps nécessaire pour effectuer une mesure varie en fonction du mode sélectionné. Dans notre cas, nous considérons la valeur maximale de 15 ms (donc attendre un delay de 15ms après chaque demande de la mesure). Le tableau ci-dessous, extrait du datasheet, détaille les durées de mesure pour chaque mode

Measurement duration	$t_{MEAS,l}$	Low repeatability	-	2.5	4.5	ms	The three repeatability modes differ with respect to measurement duration, noise level and energy consumption.
	$t_{MEAS,m}$	Medium repeatability	-	4.5	6.5	ms	
	$t_{MEAS,h}$	High repeatability	-	12.5	15.5	ms	

2- Explication du Code

a) Variables utilisées :

- **uint8_t capteur_temp_adress :** Cette variable stocke l'adresse du capteur SHT31 (0x44). Un décalage d'un bit vers la gauche est nécessaire pour laisser la place au bit de lecture/écriture (W/R).
- **uint8_t commande[2] :** Ce tableau contient les commandes spécifiques au capteur SHT31. Ces commandes, envoyées par le maître (la carte STM32), permettent notamment de demander l'envoi des mesures de température, de configurer la vitesse de mesure, ou d'autres fonctionnalités du capteur.
- **uint8_t data[6] :** Ce tableau stocke les données reçues du capteur, comprenant les valeurs brutes de température et d'humidité sur 6 octets.
- **uint16_t temperature_brut :** Cette variable enregistre la valeur brute de la température renvoyée par le capteur. Elle sera ensuite convertie en une valeur réelle de température en degrés Celsius.
- **uint16_t humid_brut :** Similaire à temperature_brut, cette variable stocke la valeur brute de l'humidité avant sa conversion en pourcentage d'humidité.
- **float temp :** Contient la valeur réelle de la température, obtenue après la conversion de temperature_brut. Cette valeur est exprimée en degrés Celsius.
- **float humid :** Contient la valeur réelle de l'humidité, résultant de la conversion de humid_brut. Elle est exprimée en pourcentage d'humidité relative (%HR).
- **char temperature[50] :** Cette chaîne de caractères sert à stocker la valeur convertie de la température sous forme de caractères, afin de l'afficher sur l'écran LCD.
- **char humidite[50] :** Similaire à temperature, cette chaîne de caractères stocke la valeur de l'humidité, formatée pour être affichée sur l'écran LCD.

b) Fonctions utilisées :

- **HAL_I2C_Master_Transmit(&hi2c1,capteur_temp_adress,commande,2,5000);**
  Cette fonction permet d'envoyer des messages du maître (la carte STM32) vers l'esclave (le capteur de température) via le protocole I2C. Elle prend en entrée 5 paramètres :
  - **L'interface I2C utilisée :** L'adresse de l'I2C sur laquelle le message sera envoyé (dans ce cas, on utilise I2C1, donc hi2c1).
  - **L'adresse de l'esclave** : Spécifiée par capteur_temp_adress, cette variable doit être de type uint16_t. Si ce n'est pas le cas, il faut effectuer un cast explicite.
  - **Le contenu du message** : Un pointeur vers les données à envoyer, ici commande. Cette variable doit être de type uint8_t*. Si ce n'est pas le cas, un cast est nécessaire.
  - **La taille du message envoyé** : En octets, ici 2 octets, ce qui signifie que deux octets seront transmis.
  - **Le délai d'attente (timeout)** : Durée maximale d'attente en millisecondes avant que la fonction retourne un message d'erreur si l'esclave (le capteur) ne répond pas.

- **HAL_I2C_Master_Receive(&hi2c1,capteur_temp_adress,data,6,5000) ;**

  Cette fonction permet de recevoir les messages que l'esclave (le capteur de température) a répondu au maître (la carte STM32) via le protocole I2C. Elle prend en entrée 5 paramètres

  - **L'interface I2C utilisée :** L'adresse de l'I2C sur laquelle le message sera renvoyé (dans ce cas, on utilise I2C1, donc hi2c1).
  - **L'adresse de l'esclave** : capteur_temp_adress
  - **Où stocker le message reçu** : Un pointeur vers la variable où stocker les données envoyées de la part du capteur, ici le tableau data. Cette variable doit être de type uint8_t*. Si ce n'est pas le cas, un cast est nécessaire.
  - **La taille du message reçu :** La longueur des données attendues, ici 6 octets, car le capteur renvoie 6 octets de données.
  - **Le délai d'attente (timeout)** : Durée maximale d'attente en millisecondes avant que la fonction retourne un message d'erreur si l'esclave (le capteur) ne répond pas.

- **sprintf(temperature,"Temp : %.1f C",temp);**

  Cette function permet de transformer la variable temp (du type float) en chaine de caractère (temperature) afin d'afficher la valeur de la température sur l'écran LCD. Idem pour la valeur de l'humidité.

  Pour faire appel à cette fonction il est nécessaire d'inclure la bibliothèque stdio.h et dans le réglage du projet il faut coucher la case Use float with printf from newlib, cela ce fait dans properties -> C/C++ build -> Settings -> MCU/MPU Settings

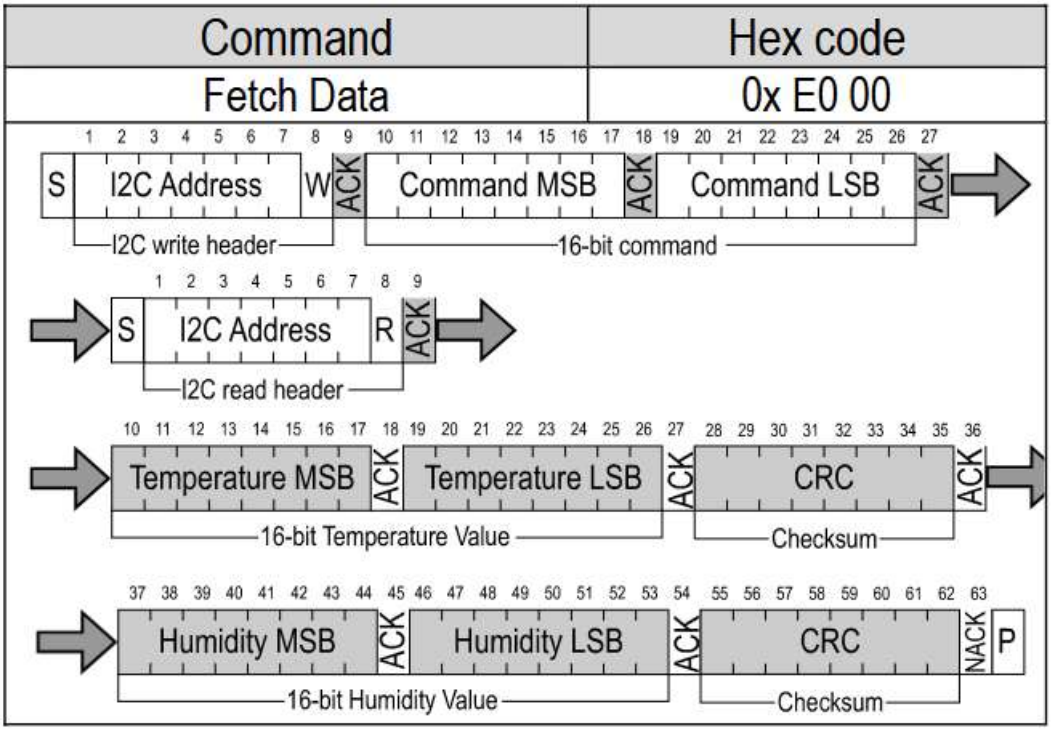
- **clearlcd** () : pour effacer le contenue de l'écran LCD
- **lcd_print**(); pour afficher des messages sur l'écran LCD
- **lcd_position() :** pour choisir la ligne et la colonne sur laquelle on fait l'affichage
- **HAL_Delay () :** pour générer des delay en ms

c) Récupération des mesures :

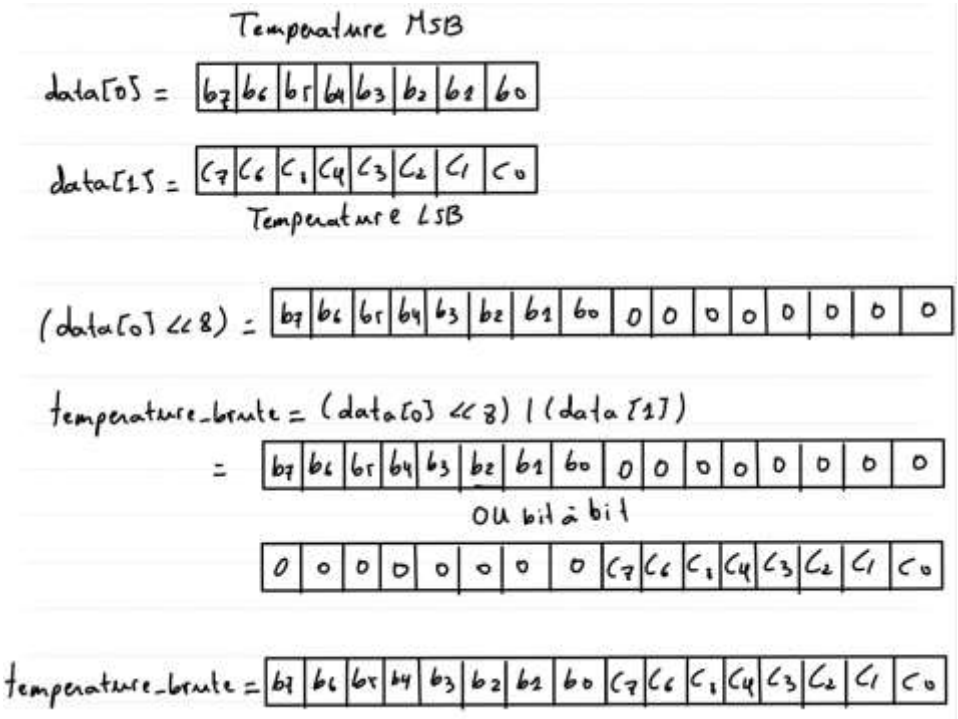
À chaque demande de mesure, le capteur SHT31 envoie un bloc de 6 octets d'un seul coup :

- Octets 1 et 2 : Ces deux octets contiennent la valeur brute de la température. Le premier octet représente la partie entière (Temperature MSB), tandis que le second correspond à la partie décimale (Temperature LSB).
- Octet 3 : Un octet de CRC (Checksum) utilisé pour vérifier si la valeur de température envoyée par le capteur correspond bien à celle reçue par la carte STM32. Il permet de détecter et éventuellement corriger les erreurs de transmission entre le capteur et le microcontrôleur.
- Octets 4 et 5 : Ces deux octets représentent la valeur brute de l'humidité. Le quatrième octet correspond à la partie entière (Humidity MSB), et le cinquième à la partie décimale (Humidity LSB).
- Octet 6 : Un second octet de CRC, cette fois ci pour la valeur de l'humidité.

Dans le datasheet on trouve le schéma suivant :



Pour récupérer la valeur totale de la température et la stockée dans la variable temperature_brute , la partie entière (Température MSB) est décalée de 8bits à gauche, cela permet d'avoir une valeur sous forme décimale sur 16 bits .La même chose est faite pour la mesure de l'humidité.



Pour transformer les valeurs brutes des mesures en valeurs réelles, on utilise les deux formules données par le datasheet :

Relative humidity conversion formula (result in %RH):

$$RH = 100 \cdot \frac{S_{RH}}{2^{16} - 1}$$

Temperature conversion formula (result in °C & °F):

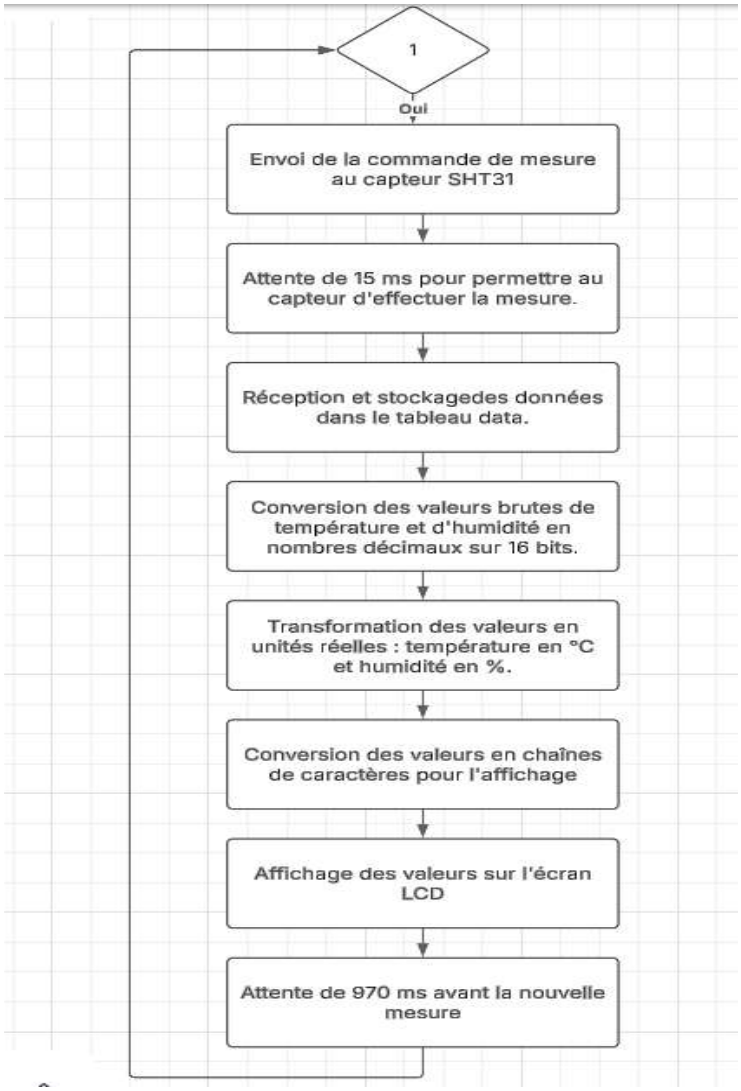
$$T[°C] = -45 + 175 \cdot \frac{S_T}{2^{16} - 1}$$

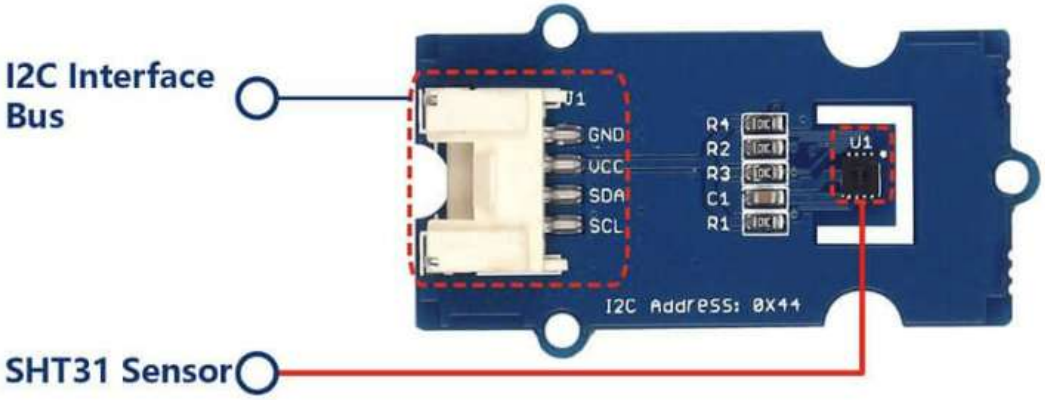
$$T[°F] = -49 + 315 \cdot \frac{S_T}{2^{16} - 1}$$

S_{RH} and S_T denote the raw sensor output for humidity and temperature, respectively. The formulas work only correctly when S_{RH} and S_T are used in decimal representation.

3- Algorithme

Le graphe suivant résume le fonctionnement du code




I2C Interface
Bus
SHT31 Sensor
U1
GND
VCC
SDA
SCL
R4
R2
R3
C1
R1
U1
I2C Address: 0X44