MORLAND Marine

VANBESIEN Alexis

UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

Bureau d'études : Microcontrôleur STM32

MORLAND Marine

VANBESIEN Alexis

Table des matières

MORLAND Marine

VANBESIEN Alexis

Capteur DHT22

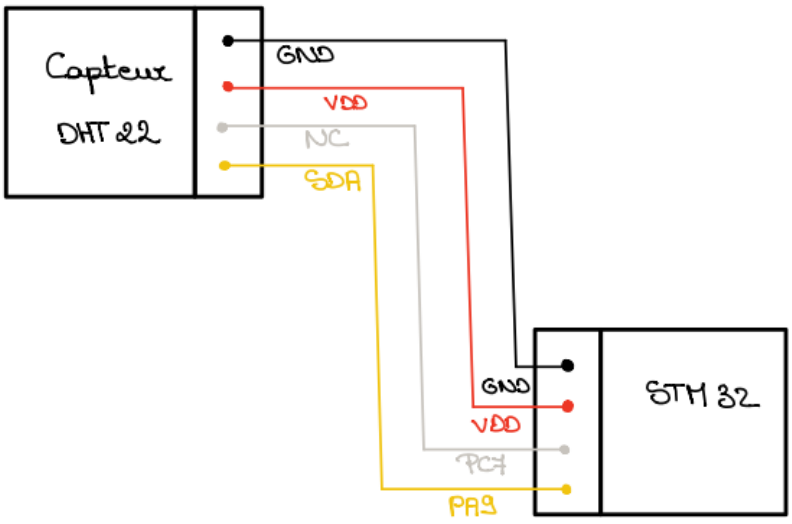
Il s'agit d'un capteur de température et d'humidité.

Caractéristiques :

	Température	Humidité
Résolution	0.1°C	0.1%RH
Intervalle de mesure	[-40; +80] °C	[0 ; 99.9]%RH

Schéma électrique :

C'est un capteur comportant quatre pins : GRND, VDD, NC (not connected) et SDA (Serial Data Bidirectionnel). Les informations du capteur DHT22 sont transmises via un simple bus de communication 1 fil qui est le SDA.



Le capteur envoie ainsi 40 bits de données à interpréter :

- bits 0 à 7 : Parity Bit

- bits 8 à 15 : Low Temperature
- bits 16 à 23 : High Temperature

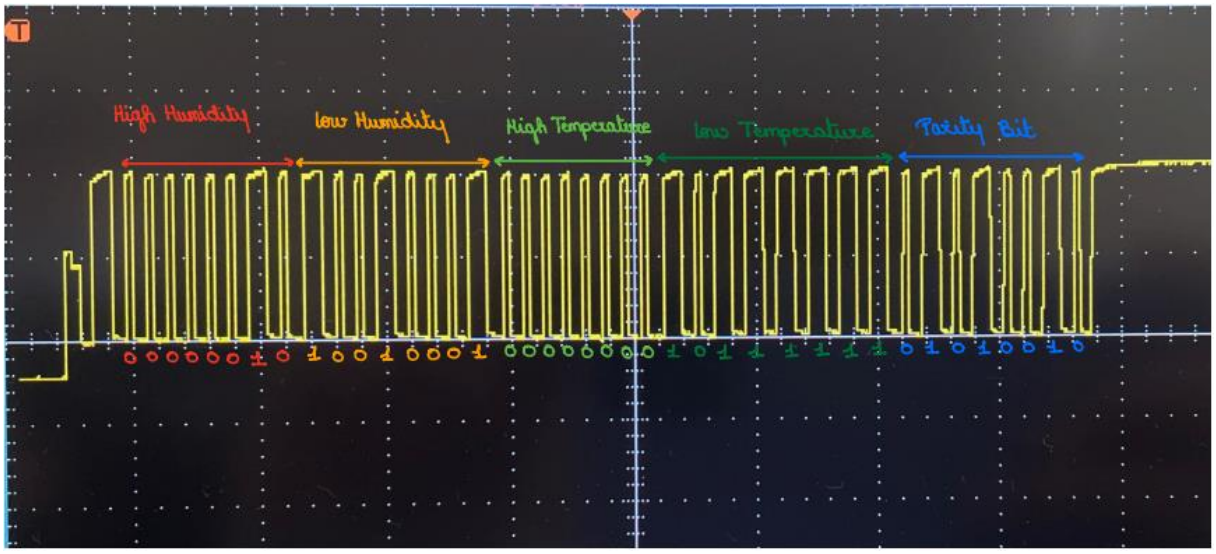
- bits 24 à 31 : Low Humidity
- bits 32 à 40 : High Humidity

Le Parity Bit doit être égal à la somme des bits n° 8 à 40. C'est une donnée permettant de vérifier la prise correcte de mesures par le capteur DHT22.

Les données de Low et High Humidity, après conversion des bites on obtient une certaine valeur. Cette valeur est alors divisée par 10 afin d'obtenir l'humidité réelle relevée par le capteur DHT22.

Par analogie, on procède à la même méthode pour connaître la valeur de température relevée par le capteur DHT22.

Durant notre BE, nous avons relevé à l'oscilloscope le signal du capteur DHT sur le pin SDA afin de vérifier son bon fonctionnement, et faire un calcul à la main de température et d'humidité.



	High Humidity + Low Humidity	High Temperature + Low Temperature	Parity Bit
Binaire	0000 0010 + 1001 0001	0000 0000 + 1011 1111	0101 0010
Valeur	$2^9 + 2^7 + 2^4 + 2^0 = 657$	$2^7 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 191$	$2^6 + 2^4 + 2^1 = 82$
Vérification	0000 0010 + 1001 0001 0000 00000 1011 1111 = 1 0101 0011 = 82		
Résultat	Humidité : 65.7%RH	Température : 19.1 °C	

Les mesures relevées et transmises par le capteur DHT22 sont en concordances.

Observation signal de départ à l'oscilloscope :

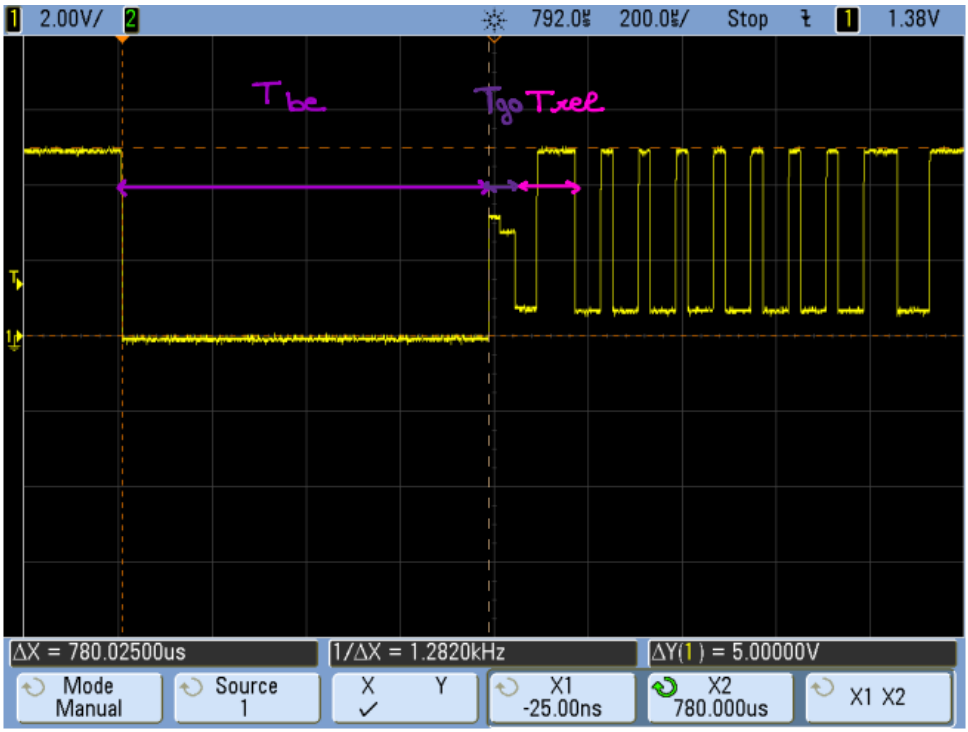
On note 3 périodes correspondant à l'envoie d'informations du master (ici STM32) au slave (ici DHT22).

- T be : Signal état bas de départ
- T go : Libération du bus de communication
- T rel : Réponse à l'état bas

Le reste du signal est étudié juste avant et est la réponse transmise du slave (DHT22) vers le master (STM32).

MORLAND Marine

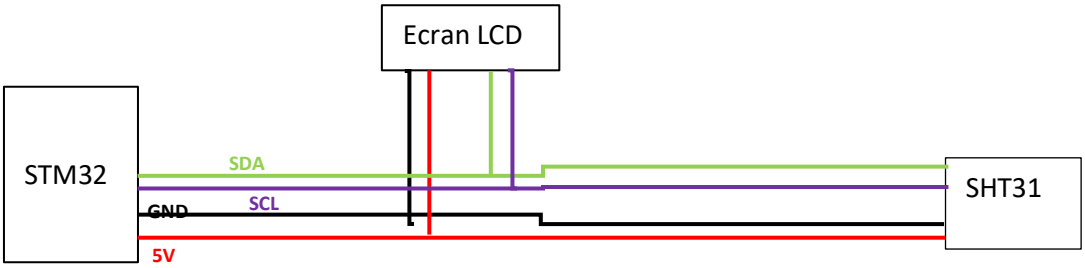
VANBESIEN Alexis



Capteur de température et d'humidité : SHT31

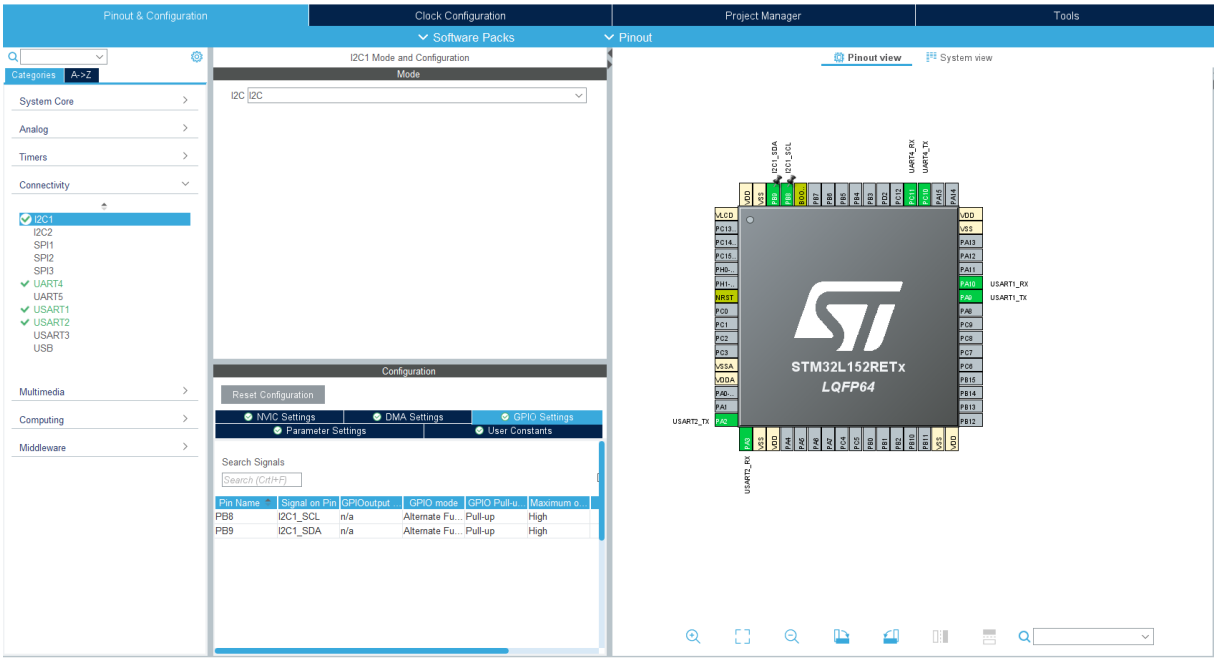
Dans ce projet, nous avons pour objectif de récupérer les informations fournis par le capteur de température et d'humidité SHT31 pour les afficher sur un écran LCD I²C en utilise l'IDE dédié à la STM32 : CubeIDE.

Le capteur utilise lui aussi le bus I²C, il faut donc effectuer le branchement suivant :



Le protocole i²C fonctionne avec 2 câbles, le SDA (Serial Data) pour l'envoi de données, et le SCL (Serial Clock) qui sert d'horloge pour déterminer la fréquence de communication.

On a un maître, ici le microcontrôleur STM32 et des esclaves possédant une adresse unique, ici l'écran LCD et le capteur SHT31. On commence donc par initialiser les ports de la STM32 pour permettre la communication I²C avec le logiciel CubeMX (intégré à CubeIDE) :



Les 2 ports qui nous intéressent, sont les ports PB8 qui fera office de SCL et PB9 de SDA. Les autres ports ne sont pas importants dans notre cas, ils ont été initialisés afin de permettre le bon fonctionnement de notre BE ultérieur.

Une fois cette tâche effectué, CubeMX nous génère un code automatique dans TrueStudio que nous allons éditer. Tout d'abord, nous rajoutons la bibliothèque de l'écran LCD qui nous a été fournis dans le projet ; Dans cette bibliothèque nous pouvons notamment trouver 2 adresses : Celle de l'affichage 0x7C, et celle du RGB qui permet de colorer l'écran 0xC4.

Nous nous penchons ensuite sur l'écriture de la bibliothèque du capteur SHT31 en lisant la datasheet expliquant son fonctionnement. On obtient ainsi :

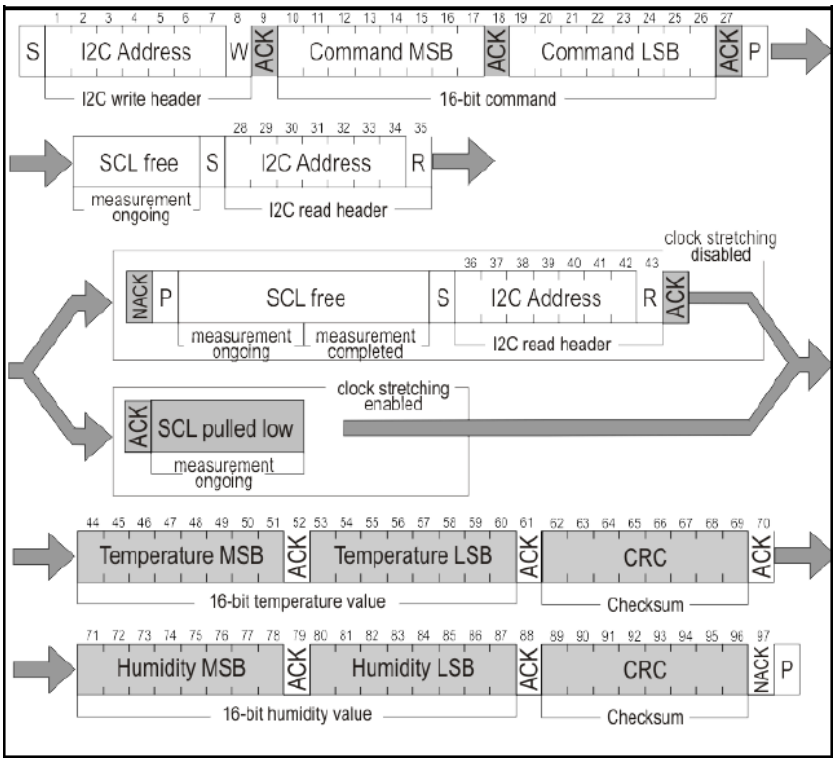
- Son adresse : 0x44

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

- Sa condition de Start : on choisira ici 0x2C06

Condition		Hex. code	
Repeatability	Clock stretching	MSB	LSB
High	enabled	0x2C	06
Medium			0D
Low			10
High	disabled	0x24	00
Medium			0B
Low			16
e.g. 0x2C06: high repeatability measurement with clock stretching enabled			

- La décomposition de l'échange en le maitre et l'esclave :



Ce dernier servira notamment à vérifier le bon fonctionnement de notre installation à l'aide d'un oscilloscope. On peut voir que la température et l'humidité sont écrit sur 16 bits chacun, avec un CRC chacun qui permet de vérifier si la valeur lue est fiable ou non.

On récupère donc ces 32 bits (2x16 bits), et on effectue la conversion comme indiqué dans la datasheet :

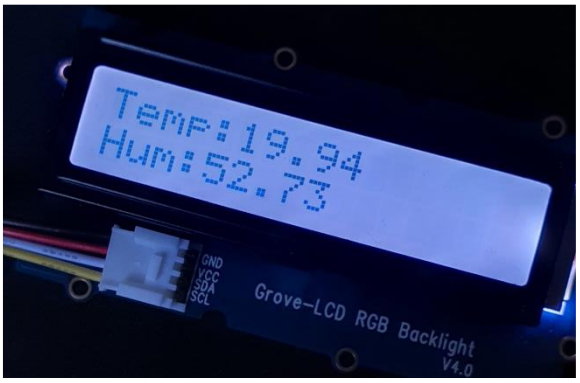
Relative humidity conversion formula (result in %RH):

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

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

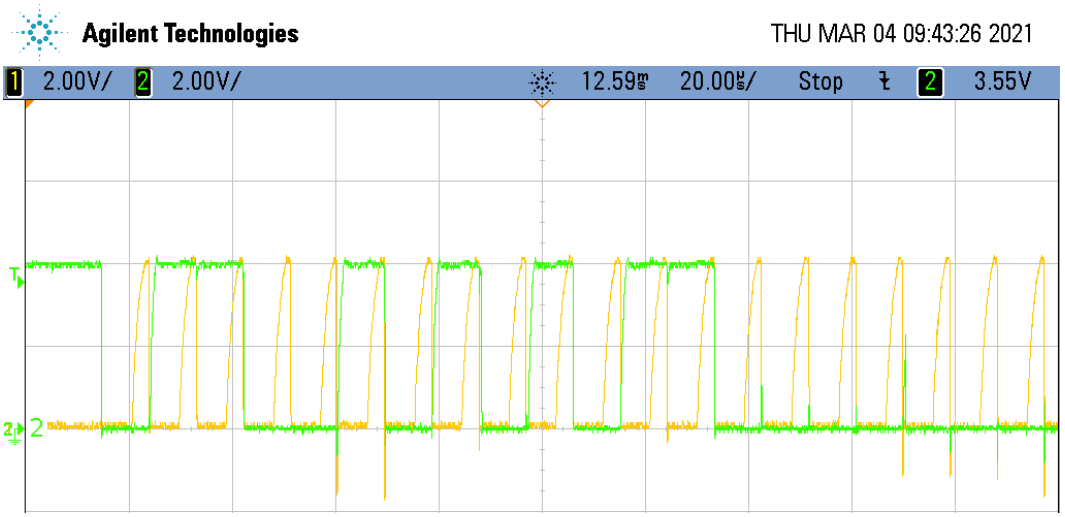
$$T[°C] = -45 + 175 \cdot \frac{S_T}{2^{16} - 1}$$

Une fois la conversion effectuée, on envoie ces informations sur l'écran LCD :



Pour s'assurer du bon fonctionnement, on va récupérer le signal SDA et SCL sur oscilloscope, l'étudier afin d'obtenir les bits correspondant à la température et à l'humidité, puis les convertir et comparer avec les valeurs de l'écran LCD.

Ici un morceau du signal étudié, la totalité sera mis en annexe dû à sa longueur. Ces valeurs n'ont pas été prises en même temps que la photo ci-dessus et ne correspondront donc pas à ce qui est affichés à l'écran, malgré tout, nous nous sommes assurés qu'elles correspondaient bien aux valeurs de température et humidité de là où nous nous trouvions au moment de la mesure



On obtient donc la suite de bits suivantes :

01100101010110000000011100011101100100011100100001011

T MSB A T LSB A CRC A H MSB A H LSB A CRC

T = 0x65C0 = 26048

S_{RH} = 0x768E = 30350

On convertit :

$$RH = 100\frac{S_{RH}}{2^{16}-1} = 46{,}31$$

$$T[°C] = -45 + 175\frac{S_T}{2^{16}-1} = 24{,}55\ °C$$

Comme dit précédemment, cela concordait avec les valeurs de la pièce où nous nous situions.

MORLAND Marine

VANBESIEN Alexis

Bureau d'études : Installation domotique

Qu'est-ce que la domotique ?

La domotique est une programmation de votre habitat au travers de fonctions spécifiques et de moyens de communication entre les propriétaires et celui-ci. Ce système doit être optimisé et automatisé.

Voici quelques aspects de mise en œuvre :

Les équipements électriques et les éléments intelligents sont reliés entre eux, ce qui leur permet de pouvoir interagir afin de réduire considérablement la consommation d'énergie électrique. Les principaux éléments sujets à cette automatisation sont le chauffage, la ventilation, l'éclairage. Ce système doit satisfaire et respecter un confort optimal au sein du domicile.

Domotique et économie d'énergie : comment ça fonctionne ?

L'habitat doit pouvoir gérer la consommation d'énergie en sécurité et devient donc un élément intelligent dans son ensemble. Afin d'y parvenir, on utilise un ordinateur* central et un réseau électrique qui vont permettre de relier les différents appareils à programmer.

Cet ordinateur central va donner des ordres et récupérer des données auprès de tous les appareils électriques et capteurs de différentes données. Il va aussi permettre à l'utilisateur de prendre connaissance de certaines données concernant son habitat.

Tout ce système est donc automatique, et doit permettre le moins d'interventions possible de l'Homme dans le processus. C'est ainsi que la maison devient intelligente, et est autogérée par la mise en place du système de domotique.

*Dans le cadre de notre BE l'ordinateur central sera un microcontrôleur STM32L152RE NUCLEO-64.

Pourquoi utiliser la domotique

Tous les avantages énumérés précédemment vont permettre de réduire les dépenses en énergie de la maison à hauteur d'environ 6 à 10 %.
Pour conclure, la domotique offre la possibilité de s'adapter au maximum aux besoins du foyer et au mode de vie. Dans un cadre plus particulier, la domotique peut aider les personnes dépendantes ou handicapées.

MORLAND Marine

VANBESIEN Alexis

Bureau d'étude

Sujet de réalisation : Optimisation de l'éclairage, de la température et de l'humidité dans une pièce

Pour cela nous avons déterminé un cahier des charges et une liste de matériel précise.

Cahier des charges :

- Optimisation de l'éclairage : Lecture de la luminosité dans la pièce grâce à une photorésistance. Détermination de valeurs seuils de luminosité afin d'établir un ordre. Réception des données. Et envoie d'ordre de positionnement des volets en fonction de la luminosité. Ici les volets sont modélisés par un servomoteur.

- Optimisation du chauffage : Lecture de la température grâce au capteur SHT31. Détermination de certaines valeurs de températures remarquables. Réception des données. Envoie d'une commande au radiateur modélisé par un servomoteur.

- Optimisation de l'humidité : Lecture de l'humidité avec le capteur SHT31. Détermination de valeurs spécifiques d'humidité. Réception des données. Envoie d'une action à effectuer au ventilateur afin de réguler l'humidité. Ce ventilateur est représenté par un moteur à courant continu (MCC).

Liste du matériel :

- Les capteurs :

    - SHT31 : Température et Humidité
    - Écran LCD : protocole de communication I2C
    - Servomoteurs : Radiateur et Volets
    - MCC : Ventilateur
    - Photorésistance : Luminosité

- Microcontrôleur : STM32L152RE – NUCLEO 64

- Environnement de Développement Intégré - IDE : STM32CubeIDE

- Autres :
-
    - Résistance 10 kOhms
    - Module d'alimentation : 3.3 V – 5 V
    - Fils de liaison
    - Plaquette de protection ou Shield

MORLAND Marine

VANBESIEN Alexis

Photorésistance

Il s'agit d'un dipôle dont la résistance varie en fonction de l'éclairement E qu'elle reçoit d'une source de lumière. La partie sensible du capteur est une piste de sulfure de cadmium en forme de serpentin. L'énergie lumineuse reçue déclenche une augmentation de porteurs de charges libres dans ce matériau, de sorte que sa résistance électrique évolue.

Servomoteur

Il s'agit d'un moteur électrique se caractérisant par sa très grande capacité de régulation du positionnement suivant un angle. Ce moteur utilise la rétroaction afin de déterminer la position de l'arbre. En plus d'être un moteur, il est composé d'un capteur de position du rotor. Ce capteur permet de détecter la position de l'arbre moteur. Il est alors possible de contrôler la position de l'arbre moteur suivant l'angle ordonné. Le servomoteur est en général utilisé pour contrôler la position des objets, les faire pivoter ou bien déplacer un bras, une jambe, des robots, ou bien encore de déplacer différents capteurs.

La période entre deux impulsions doit être de 20 ms ou on peut dire que la fréquence de l'impulsion doit être de 50 Hz.

Le servomoteur est composé de 3 pins :

- Fil de terre noir
- Fil d'alimentation 5V rouge
- Fil de signal PWM jaune

Les servos peuvent aller de 0 à 180 degrés selon la largeur de l'impulsion. C'est un processus de modulation de largeur d'impulsion, MLI ou PWM :

- 1 millisecondes et correspond à 0 degré
- 1,5 millisecondes et correspond à 90 degrés
- 2 millisecondes et correspond à 180 degrés

Moteur à courant continu

Une machine à courant continu est une machine électrique. Il s'agit d'un convertisseur électromécanique permettant la conversion bidirectionnelle d'énergie entre une installation électrique parcourue par un courant continu et un dispositif mécanique. Selon la source d'énergie.

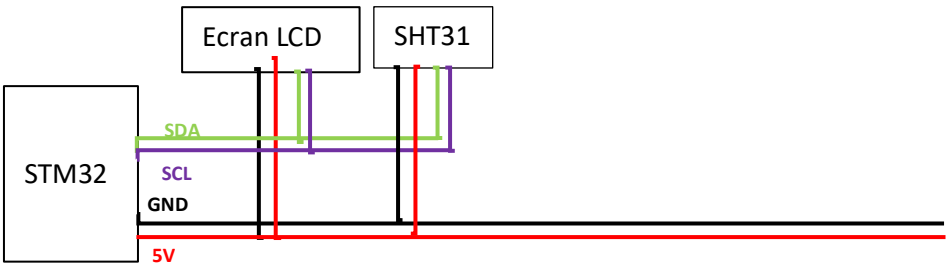
- En fonctionnement moteur, l'énergie électrique est transformée en énergie mécanique.
- En fonctionnement générateur, l'énergie mécanique est transformée en énergie électrique (elle peut se comporter comme un frein). Dans ce cas elle est aussi appelée dynamo.

A l'aide d'un signal PWM, nous pouvons faire varier sa vitesse grâce à la valeur efficace de tension transmise, plus le rapport cyclique sera petit, plus la vitesse sera grande.

MORLAND Marine

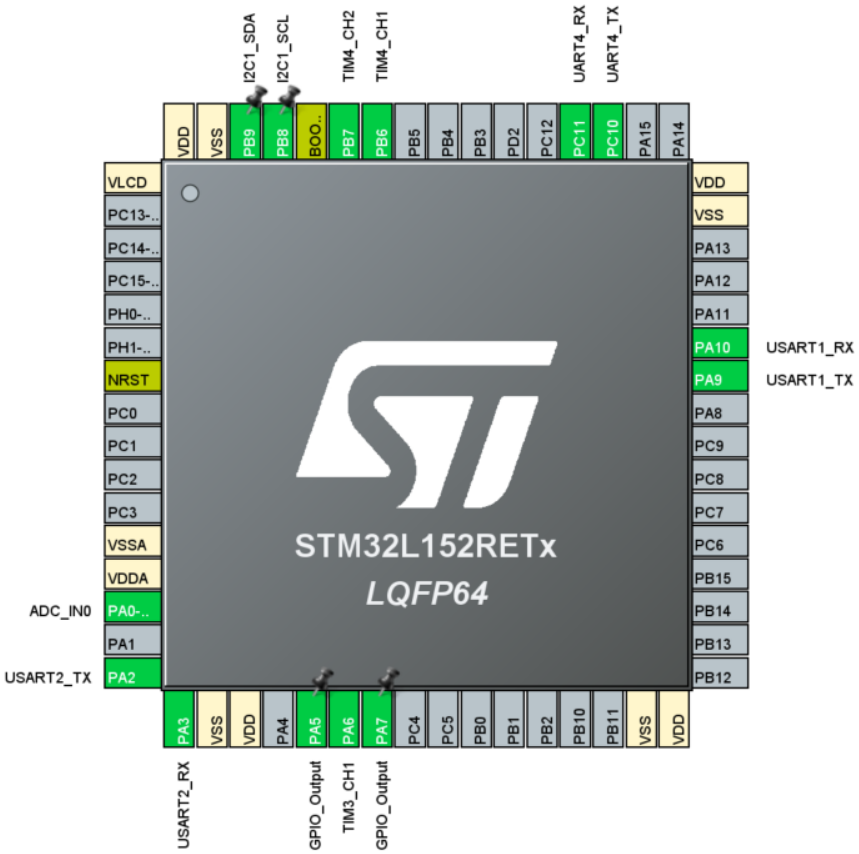
VANBESIEN Alexis

On utilisera le schéma de câblage suivant :



L'écran LCD et le SHT31 auront donc le même fonctionnement que précédemment, nous utiliserons ici 2 fonctionnalités : la lecture d'une tension, et l'utilisation d'un signal PWM.

On cherche donc à lire la tension aux bornes de la photorésistance, qui nous permettre donc programmer différents scénarios en fonction de l'intensité lumineuse de la pièce. Nous commençons par initialiser le pin PA0 en ADC_IN :



On vient initialiser la lecture à l'aide de :

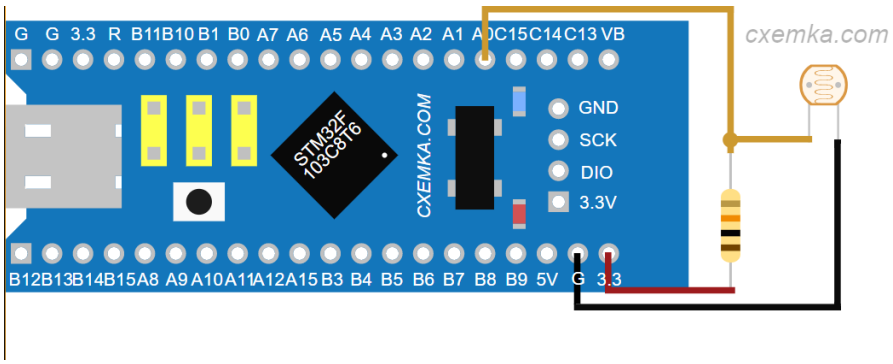
```
MX_USART1_UART_Init();

MX_ADC_Init();
```

Ensuite pour y faire la lecture :

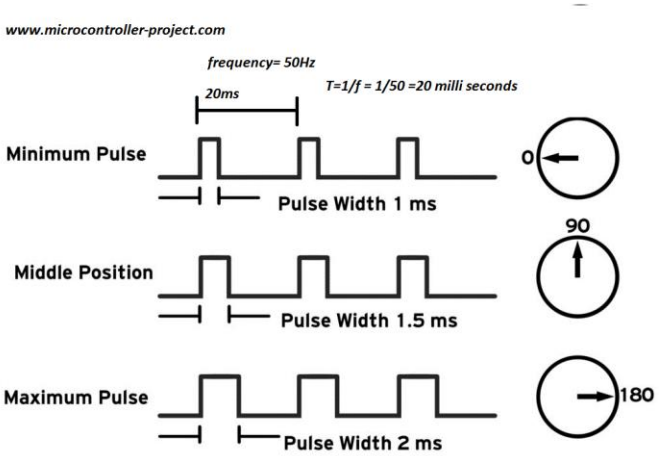
```
HAL_ADC_Start(&hadc);
HAL_ADC_PollForConversion(&hadc, HAL_MAX_DELAY);
raw = HAL_ADC_GetValue(&hadc);
```

La function HAL_ADC_Start va autoriser la conversion, qui être effectué à l'aide de la fonction HAL_ADC_PollForConversion. La valeur lue va être stocké dans un registre puis lu à l'aide de la fonction HAL_ADC_GetValue. Le branchement sera simple effectué comme ceci :



Nous pouvons aussi rajouter un condensateur de l'ordre de 100pF pour lisser la valeur de tension et ainsi éviter les fortes variations qui pourraient nuire à la mesure.

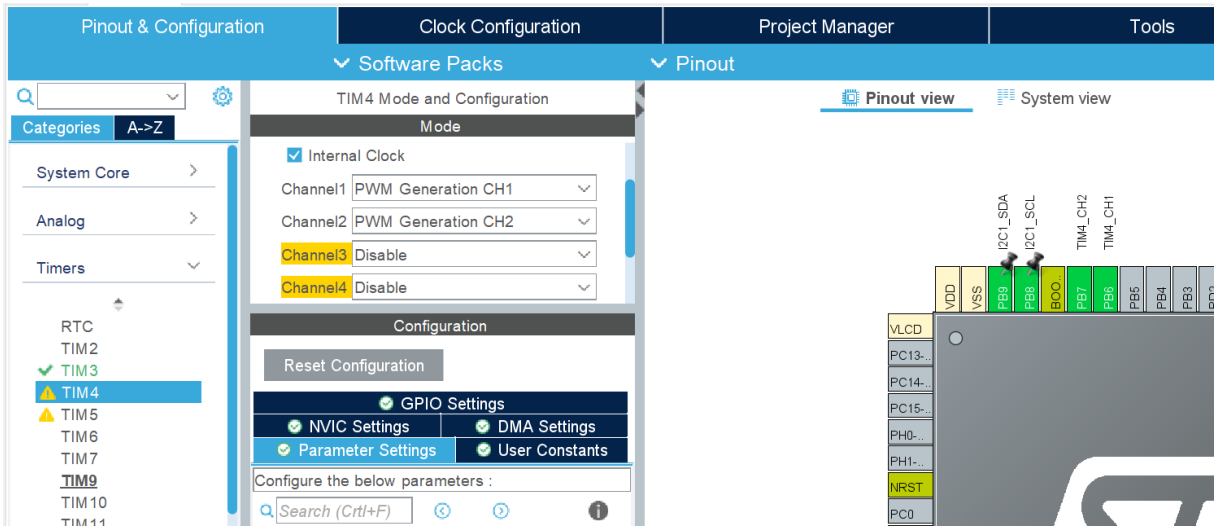
Nous passons ensuite à l'utilisation d'un signal PWM. C'est un signal carré périodique dont on peut faire varier le rapport cyclique comme ceci :



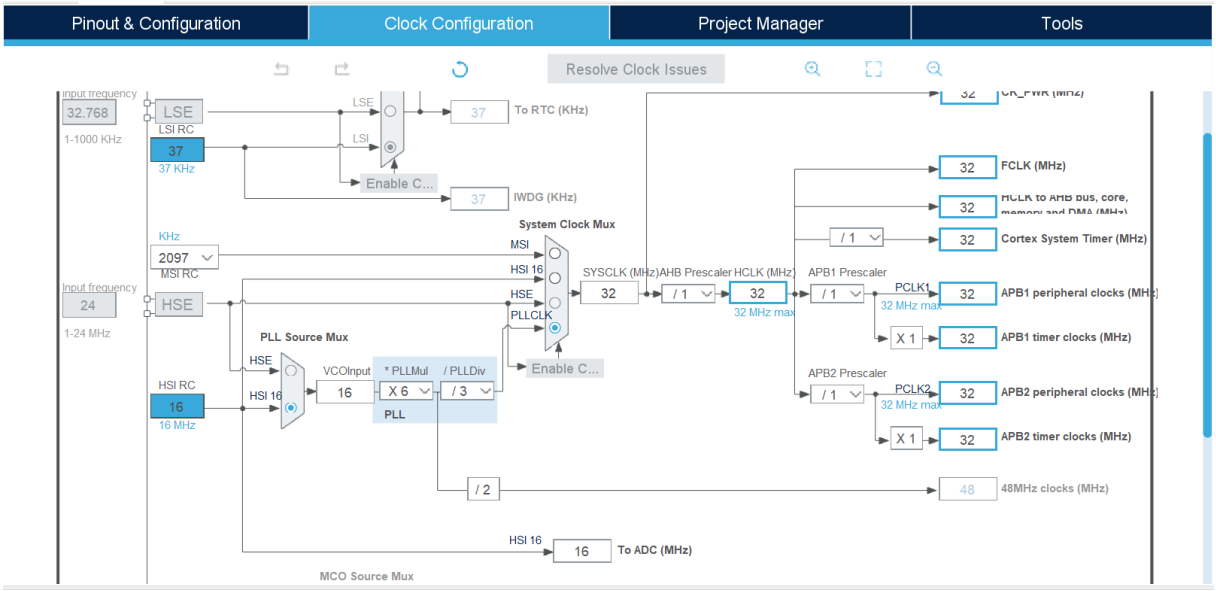
Nous allons l'utiliser de 2 manières différentes :

- Nous allons faire varier la valeur efficace de tension du signal pour gérer la vitesse du moteur à courant continu
- Faire varier le rapport cyclique afin de gérer la position du servomoteur.

On commence donc par initialiser un pin en TIMN_CHX (PWM Generation Output) et nous allons utiliser 2 Channel sur le même timer :



Nous réglons la valeur d'horloge interne à son maximum, 32 MHz dans le cas de la STM32L152RE :



Les 2 moteurs fonctionnent de manière optimale à 50Hz, il faut donc procéder à quelques calculs pour pouvoir les faire fonctionner :

$$Counter\ Frequency\ =\ Frequency\ Required\ *\ Pwm\ Resolution$$

$$TimerPrescallerValue = \frac{Timer\ input\ clock}{Counter\ Frequency} - 1$$

Avec donc :

$$Timer\ input\ clock\ =\ 32\ MHz$$

$$Frequency\ Required\ =\ 50Hz$$

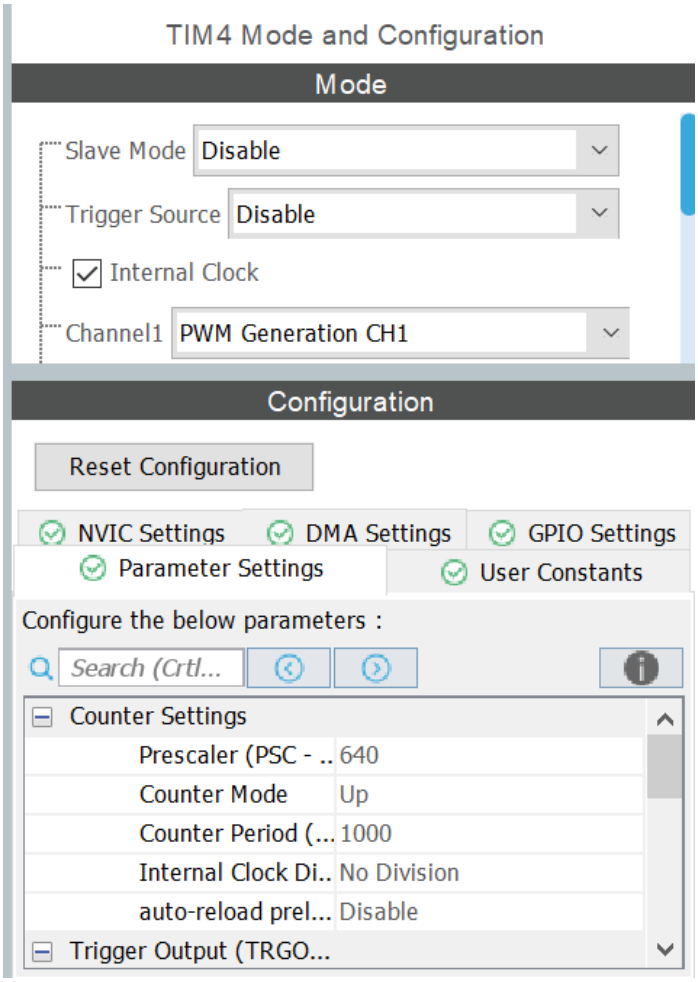
$$Pwm\ Resolution\ =\ 1000$$

On réalise les différents calculs :

$$Counter\ Frequency\ =\ 50\ 000$$

$$Timer\ Prescaller\ Valuer\ =\ 639$$

On rentre ces paramètres dans CubeIDE :



Au niveau du code, il faut donc initialiser les différents timer :

```
MX_TIM4_Init();
MX_TIM3_Init();
```

Le TIM4 sera utilisé pour les 2 servomoteurs, l'un représentant un radiateur, l'autre représentant l'ouverture/fermeture d'un rideau, et le TIM3 pour la machine à courant continue qui symbolisera notre ventilation.

On lance donc le signal avec les commandes suivantes :

```
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1); //servo moteur température-
HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2); //servo moteur luminosité
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1); //mcc ventilation
```

On peut ensuite moduler la largeur d'impulsion avec la fonction :

__HAL_TIM_SET_COMPARE(&timnX, TIM_CHANNEL_X, int largeur);

On règle donc la valeur de la largeur d'impulsion en fonction soit de la température pour le radiateur de manière à avoir constamment 24°C :

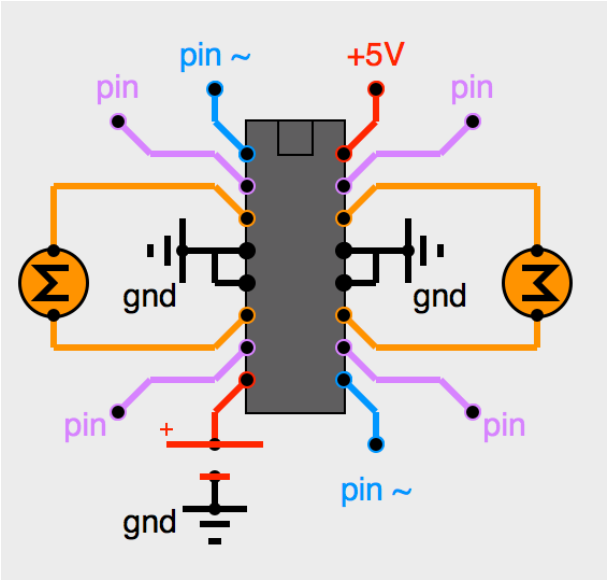
```
if(temp <= 22)
{
	__HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_1,100);
}
else if(temp <= 24 && temp > 22.3)
{
	__HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_1,75);
}
else if (temp <= 26 && temp > 22.3)
{
	__HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_1,50);
}
else if (temp > 26)
{
	__HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_1,25);
}
```

Soit en fonction de l'intensité lumineuse dans le cas des rideaux pour les avoir fermés lors de la nuit et lorsque la luminosité est trop forte :

```
if (raw <= 400)
		{
		  __HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_2,125);
			HAL_Delay(100);
		}
		else if(raw > 500 && raw < 1200 )
		{
			__HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_2,75);
			HAL_Delay(100);

		}
		else if(raw >= 1300)
		{
			__HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_2,25);
			HAL_Delay(100);
		}
```

Pour le moteur à courant continu, l'installation est différente au vu de sa demande en courant qui ne peut être fourni par la STM32. On utilise donc une source de tension externe, et un contrôleur L293D :



On doit donc utiliser 3 pins :

- 2 pins qui vont servir à définir le sens de rotation du moteur
- Le pin~ PWM qui va gérer la vitesse du moteur

On initialise donc les 2 pins de rotation :

```
HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,GPIO_PIN_SET); //sens de rotation-
HAL_GPIO_WritePin(GPIOA,GPIO_PIN_7,GPIO_PIN_RESET);
```

On règle ensuite la ventilation en fonction de l'humidité :

```
if(hum < 60)
         {
                  __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,0);
         }
         else if(hum >= 60 && hum < 70)
        {
                  __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,60); //ventilation à
40%
        }
        else if (hum >= 70 && hum < 80)
        {
              __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,80); //ventilation à 60%
        }
        else if (hum >= 80)
        {
              __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,100); //ventilation à
80%
        }
```

MORLAND Marine

VANBESIEN Alexis