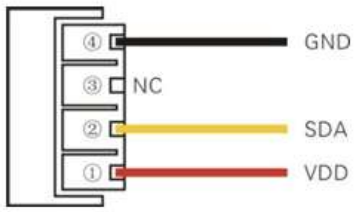
Capteur d’humidité et température (DHT22)

Le capteur **AM2302** (ou **DHT22**) mesure la **température** (-40°C à +80°C) et l’**humidité** (0 % à 100 %) avec une bonne précision. Il communique via une sortie numérique sur une seule ligne bidirectionnelle, simplifiant son intégration avec des microcontrôleurs comme les **STM32**. Alimenté en 3,3V à 6V, il nécessite une résistance pull-up sur la broche de données. Il est couramment utilisé dans la domotique et les stations météorologiques.

Table 1： AM2302 Pin assignments

Pin	Name	Description
①	VDD	Power (3.3V−5.5V)
②	SDA	Serial data, bidirectional port
③	NC	Empty
④	GND	Ground



PIC1： AM2302 Pin Assignment

Pour utiliser ce capteur nous avons commencé par faire les fonctions de configuration du pin PA0 soit en input ou en output avec resistance pull-up (capteur_temperature.c) pour communiquer avec le capteur DHT22, en suite nous avons configuré PB8 (SCL) et PB9 (SDA) pour assurer la connexion I2C avec l’écran LCD.

Nous avons fait une fonction timer pour initialiser TIM2 et l’utiliser pour créer des délais en microsecondes (timer.c).

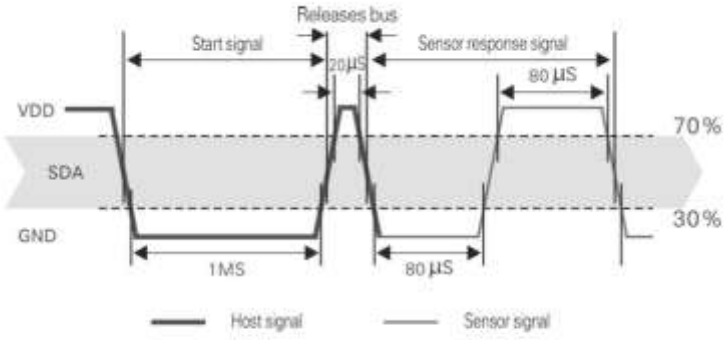
Après dans le main nous avons initialisé tous les périphériques :

```
/* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_I2C1_Init();
  lcd_init(&hi2c1,&lcdData); // initialisaer l'ecran LCD
  Timer_initialisation(); // initialiser le timer TIM2
```

Comme expliqué sur la datasheet et montre sur l'image ci-dessous nous devons envoyé un signal bas de 1 ms puis un signal haut de 20 microsecondes pour demander une mesure au capteur.
Pour générer le Delay en microsecondes nous avons fait une fonction '**delay_us**' à partir de timer **TIM2** et nous avons pris un PSC = 79 et ARR=9 ce qui permet d'avoir 10 us à chaque overflow.

```
TIM2->PSC = 79;/*80Mhz/80 => 1MHz (un tick par 1us)
TIM2->ARR = 9;/* overflow à 10us*/
```

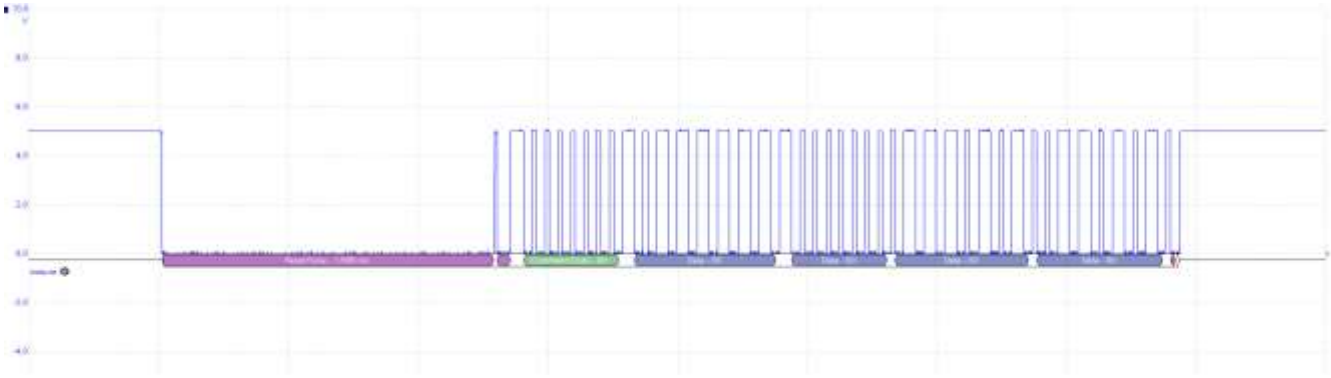


Pic7: Single bus decomposition of the timing diagram

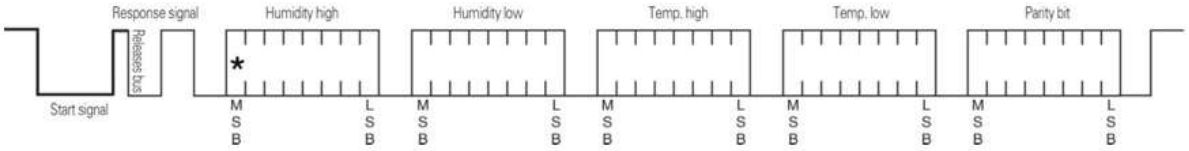
Dans le la While (1) :

```
/*etape 1 : communiquer avec le capteur */
      pin_output_config();// se mettre en sortie pour communiquer avec le
capteur
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_0, 0);//mise a 0 du pin PA0
      HAL_Delay(1); // delay 1ms
      HAL_GPIO_WritePin(GPIOA,GPIO_PIN_0, 1);//mise a 1 du pin PA0/
      delay_us(2); // delay 20us
```

Voici un exemple de tram visualisé avec le Pico-scope :



Maintenant nous récupérons les mesures brutes sur 5 octets avec la fonction **capteur_mesures ()** (capteur_temperature.c) le 1 er et 2 eme octets correspond à l'humidité, 3 et 4 à la température :



Pic5: AM2302 Single-bus communication protocol

```
/*etape 2 et 3 : récupérer les mesures*/
      pin_input_config();// se mettre en entrée pour récupérer les mesures
      delay_us(16);// delay de 2*80us, attente de la réponse du capteur
      capteur_mesures(mesures); // remplir le tableau mesures
      humidite_brut = (mesures[0] << 8) | mesures[1]; // mesures[0] =
humidity high (8bits) , mesures[1] =  humidity low (8bits)
      temperature_brut = (mesures[2] << 8) | mesures[3]; // mesures[2] =
temp high (8bits) , temp[3] =  humidity low (8bits)
```

À cette étape nous calculerons l'humidité en pourcentage et la température en °C avec la méthode indiqué sur la datasheet

```
      temp =(float) temperature_brut/10.0; //transformer la mesure en
temperature reelle
      humid =(float) humidite_brut/10.0; // transformer la mesure en
humidite reelle
```

Pour permettre l'affichage sur l'écran LCD nous transformons les float en caractères :

Note : pour utiliser cette fonction il faut activer la fonction (-u _ptintf_float) dans project>properties>C/C++ Build >Settings >MCU/MPU Settings

```
sprintf(temperature,"Temp : %.1f C",temp);//transformer du float en char pour les afficher dans le LCD
sprintf(humidite,"Humid : %.1f %%",humid);//transformer du float en char pour les afficher dans le LCD
```

Faut savoir aussi qu'il faut changer l'adresse de LCD dans la bibliothèque (lcd.c) mise à disposition selon l'écran utilisé par exemple dans notre cas (0x3E << 1).