Chapitre3 : Les capteurs pour l'IoT

Un capteur est un dispositif transformant l'état d'un grandeur physique observée en une grandeur utilisable, telle qu'une tension électrique, une hauteur de mercure, une intensité ou la déviation d'une aiguille.

1. Classification des capteurs

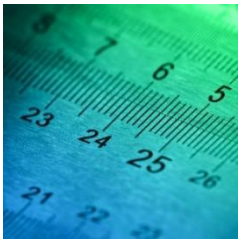
Les capteurs et leurs conditionneurs peuvent faire l'objet d'une classification par type de sortie :

- Capteurs analogiques,
- Capteurs numériques,
- Capteurs logiques.

1.1 Capteurs analogiques

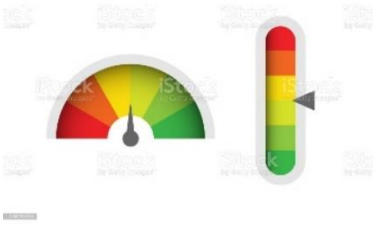
La sortie dans le capteur est une **grandeur électrique** dont la valeur est une fonction de la **grandeur physique** mesurée par le capteur. La sortie peut prendre une infinité de valeurs continues. Le signal des capteurs analogiques peut être du type :

- sortie tension(voltage) ;
- sortie courant(ampérage) ;
- règle graduée, cadran, jauge (avec une aiguille ou un fluide),etc.



Règle graduée

Cadran

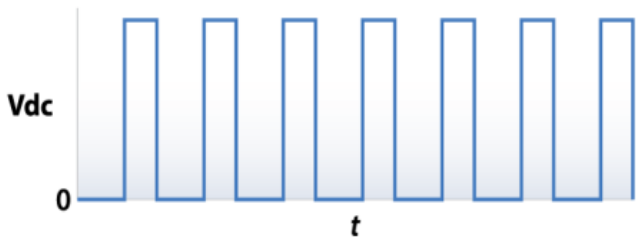


Jauge

1.2 Capteurs numériques

La sortie est une séquence d'états logiques qui, en se suivant, forment un nombre. La sortie peut prendre un grand nombre de valeurs discrètes. Le signal des capteurs numériques peut être du type :

- Train d'impulsions, avec un nombre précis d'impulsions ou avec une fréquence précise



- Code numérique binaire

- Bus de terrain

1.3 Capteurs logiques

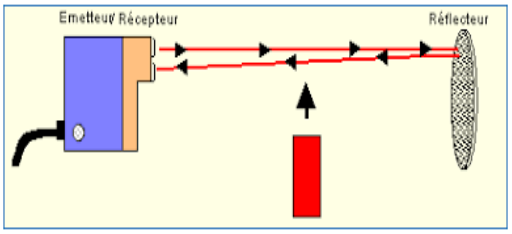
Capteurs logiques ou capteurs TOR. La sortie est un état logique que l'on note 1 ou 0. La sortie peut prendre ces deux valeurs. Il y a notamment 4 types de capteurs logiques :

- Courant présent ou bien absent dans un circuit ;
- Potentiel, souvent 5 V ou bien 0 V ;
- LED allumée ou bien éteinte ;
- signal pneumatique (pression normale/forte pression) ;
- etc.

Quelques capteurs logiques typiques : capteurs de fin de course ; capteurs de rupture d'un faisceau lumineux ; capteur de présence d'eau, divers capteurs de position.

Capteur de présence d'eau

Capteurs de fin de course



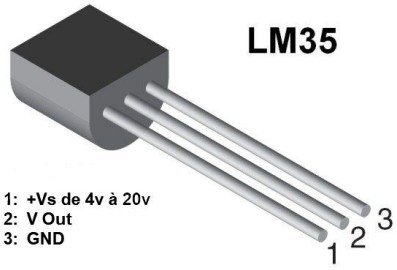
Capteurs de rupture d'un faisceau lumineux

capteurs de position

2. Capteur analogique de température - cas du LM35

Le capteur analogique de température LM35 peut mesurer la température ambiante dans un local. Il dispose de trois pattes : Broche1 : doit être connecter à une source de courant qui fournit un voltage de 4v à 20v.

Broche2 : c'est le voltage fournit par le capteur qui permet de mesurer le degré de température en °C. Le facteur d'échelle de sortie du LM35 est de 10 mV par ° C. Par exemple pour 25 °C, il fournit une tension de sortie de 250 mV.



Broche3 : doit être connecter au pin Ground ou Terre de la carte ioT.

Remarque : pour certaines versions du capteur LM35 le branchement direct dans le 5 volt entraine un sur-chauffage du capteur d'où il est recommander de rajouter une résistance.

Exercice d'application : Ecrire un code Arduino permettant de lire à partir d'un capteur LM35 le degré de température ambiante en °C et de l'afficher chaque 3 secondes sur le moniteur série. Sachez que :

- Le port **V out** est lié au port analogique **A0**
- L'instruction qui permet la lecture d'une valeur analogique est : ***int valeur_brute = analogRead(A0);***
- L'instruction qui permet de Transforme la mesure du grandeur d'électricité en température est : ***float temperature_celcius = valeur_brute * (1.1 / 1023.0 * 100.0);***
- L'instruction de configuration (à écrire dans la fonction setup()) qui permet d'améliorer la précision de la mesure en réduisant la plage de mesure est :
  analogReference(INTERNAL); // Pour Arduino UNO
  analogReference(INTERNAL1V1); // Pour Arduino Mega2560

Correction :

```
/* Code Arduino pour le capteur LM35 (2°C ~ +110°C) avec une meilleure précision. */

void setup() {

 // Initialise la communication avec le PC
 Serial.begin(9600);

 // Améliore la précision de la mesure en réduisant la plage de mesure
// analogReference(INTERNAL); // Pour Arduino UNO
 analogReference(INTERNAL1V1); // Pour Arduino Mega2560
}

void loop() {

  // Mesure la tension sur la broche A0
 int valeur_brute = analogRead(A0);

 // Transforme la mesure (nombre entier) en température via un produit en croix
 float temperature_celcius = valeur_brute * (1.1 / 1023.0 * 100.0);

  // Envoi la mesure au PC pour affichage et attends 250ms
 Serial.println(temperature_celcius);

 delay(3000);
}
```

Exercice d'application : Modifier le code de l'exercice précédent pour afficher aussi la valeur brute de la grandeur d'électricité. Vérifier la formule de conversion de la grandeur d'électricité en grandeur de température.

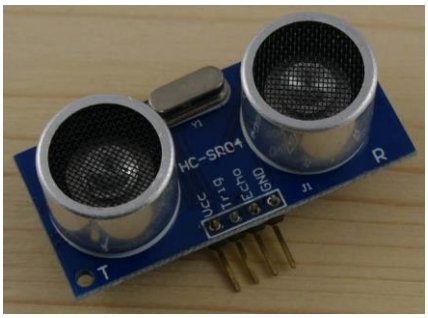
3. Capteur numérique – cas du capteur à ultrason «HC-SR04 »

Principe de fonctionnement :

Un capteur de distance à ultrason utilise le même principe qu'un capteur laser, mais en utilisant des ondes sonores (inaudible) au lieu d'un faisceau de lumière.

Ils sont bien moins chers qu'un capteur laser, mais aussi bien moins précis. Cependant, contrairement aux capteurs à infrarouge, la lumière ambiante et l'opacité de la surface en face du capteur ne jouent pas sur la mesure.

Le capteur HC-SR04 est un capteur à ultrason low-cost. Ce capteur fonctionne avec une tension d'alimentation de 5 volts, dispose d'un angle de mesure de 15° environ et permet de faire théoriquement des mesures de distance entre 2 centimètres et 4 mètres avec une précision de 3mm.



Capteur à ultrason HC-SR04

N.B. *Il existe des capteurs à ultrason bien plus haut de gamme (donc précis) mais plus cher.*

Etapes de mesure de distance avec le capteur à ultrason HC-SR04

Le principe de fonctionnement du capteur est entièrement basé sur la vitesse du son.

Voilà comment se déroule une prise de mesure :

1. On envoie une impulsion HIGH de 10µs sur la broche TRIGGER du capteur.
2. Le capteur envoie alors une série de 8 impulsions ultrasoniques à 40KHz (inaudible pour l'être humain)
3. Les ultrasons se propagent dans l'air jusqu'à toucher un obstacle et retournent vers le capteur.
4. Le capteur détecte l'écho et clôture la prise de mesure.

Le signal sur la broche ECHO du capteur reste à HIGH durant les étapes 3 et 4, ce qui permet de mesurer la durée de l'aller-retour des ultrasons et donc de déterminer la distance.

N.B. *Il y a toujours un silence de durée fixe après l'émission des ultrasons pour éviter de recevoir prématurément un écho en provenance directement du capteur.*

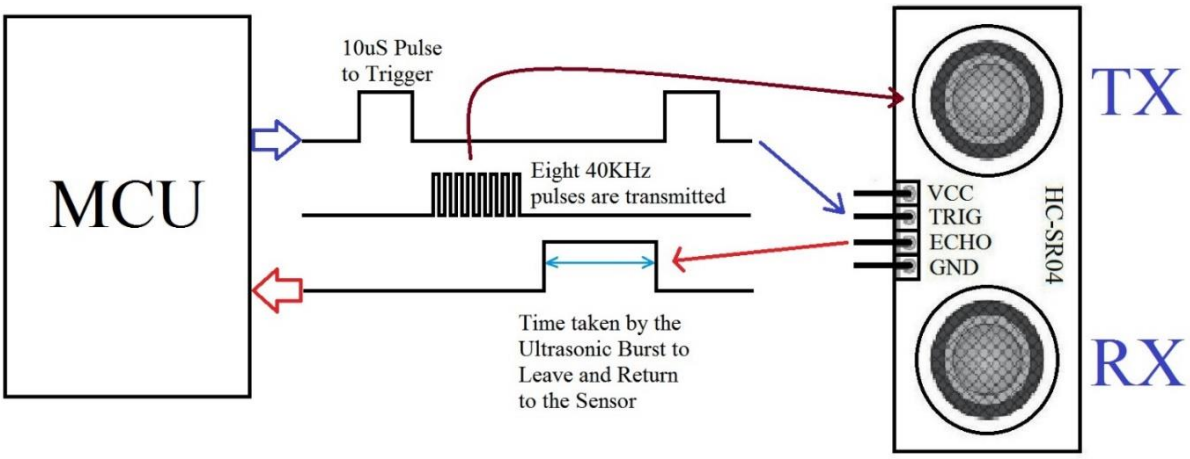
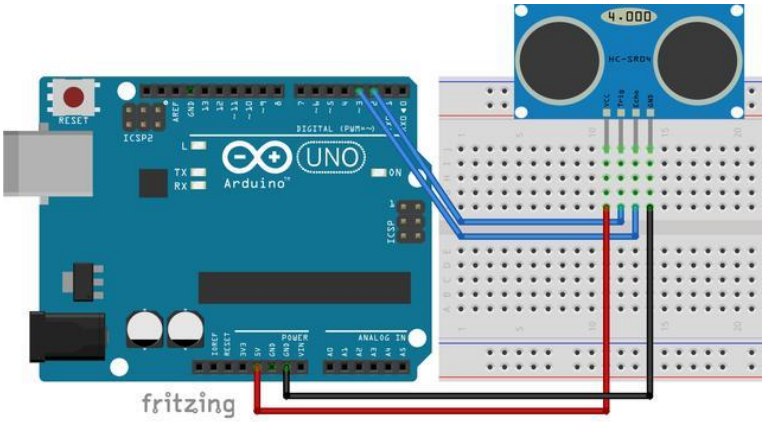


Illustration du signal TRIGGER et ECHO

Matériel nécessaire

Pour réaliser ce premier montage, il va nous falloir :

- Une carte Arduino UNO (et son câble USB),
- Un capteur HC-SR04,
- Une plaque d'essai et des fils pour câbler le montage.



Vue prototypage du montage

- L'alimentation 5V de la carte Arduino va sur la broche VCC du capteur.
- La broche GND de la carte Arduino va sur la broche GND du capteur.
- La broche D2 de la carte Arduino va sur la broche TRIGGER du capteur.
- La broche D3 de la carte Arduino va sur la broche ECHO du capteur.

N.B. *La plaque d'essai est ici totalement optionnelle. Si vous avez des fils mâles / femelles, vous pouvez directement câbler le capteur à la carte Arduino.*

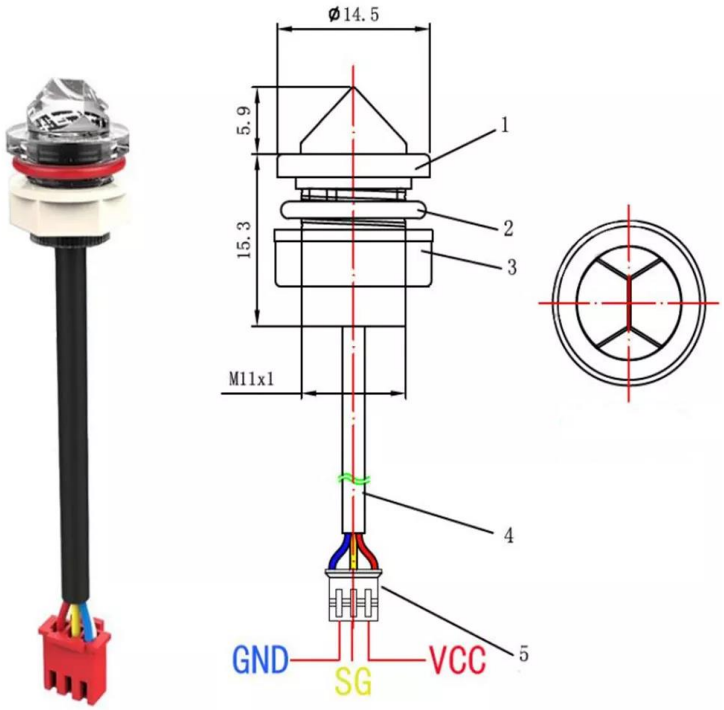
```
const byte TRIGGER_PIN = 2; // Broche TRIGGER
const byte ECHO_PIN = 3;    // Broche ECHO
/* Constantes pour le timeout */
const unsigned long MEASURE_TIMEOUT = 25000UL; // 25ms = ~8m à    340m/s
```

```
/* Vitesse du son dans l'air en mm/  ms */
const float SOUND_SPEED = 340.0 / 1000;
void setup() {
    Serial.begin(9600); /* Initialise le port série */
  /* Initialise les broches */
  pinMode(TRIGGER_PIN, OUTPUT);
  digitalWrite(TRIGGER_PIN, LOW); // La broche TRIGGER doit être à LOW au repos
  pinMode(ECHO_PIN, INPUT);
}
void loop() {   /* 1. Lance une mesure de distance en envoyant une impulsion HIGH de 10µs sur la broche TRIGGER */
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  /* 2. Mesure le temps entre l'envoi de l'impulsion ultrasonique et son écho (s'il existe) */
  long measure = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT);
  /* 3. Calcul la distance à partir du temps mesuré */
  float distance_mm = measure / 2.0 * SOUND_SPEED;
  /* Affiche les résultats en mm, cm et m */
  Serial.println(F("Distance: "));
  Serial.print(distance_mm);
  Serial.println(F("  mm"));
  Serial.print(distance_mm / 10.0, 2);
  Serial.println(F("  cm"));
  Serial.print(distance_mm / 1000.0, 2);
  Serial.println(F("  metre"));
  /* Délai d'attente pour éviter d'afficher trop de résultats à la seconde */
  delay(3000);
}
```

4. Capteur logique de présence de l’eau dans un réservoir (FS-IR82b)

L'indicateur de niveau optique FS-IR82b réfracte la lumière infrarouge émise par le tube de transmission interne et juge si le capteur est dans un état d'eau en fonction de la quantité de lumière photoélectrique reçue par le récepteur interne. Si seule une petite quantité de lumière est reçue ou n'est pas reçue parce que la lumière est réfractée dans l'eau, on considère qu'elle a un état d'eau.

Le type photoélectrique est affecté par le principe de fonctionnement car il s'agit de lumière infrarouge, mais n'a rien à voir avec des facteurs tels que la pression, la température, la densité, la corrosion, etc., et a peu d'effet sur le milieu mesuré. En raison de sa haute répétabilité et de sa grande précision de détection, il présente les caractéristiques d'une grande fiabilité et d'une longue durée de vie.

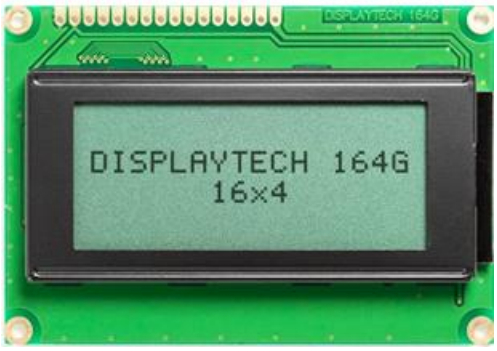


Bleu	Jaune	Rouge

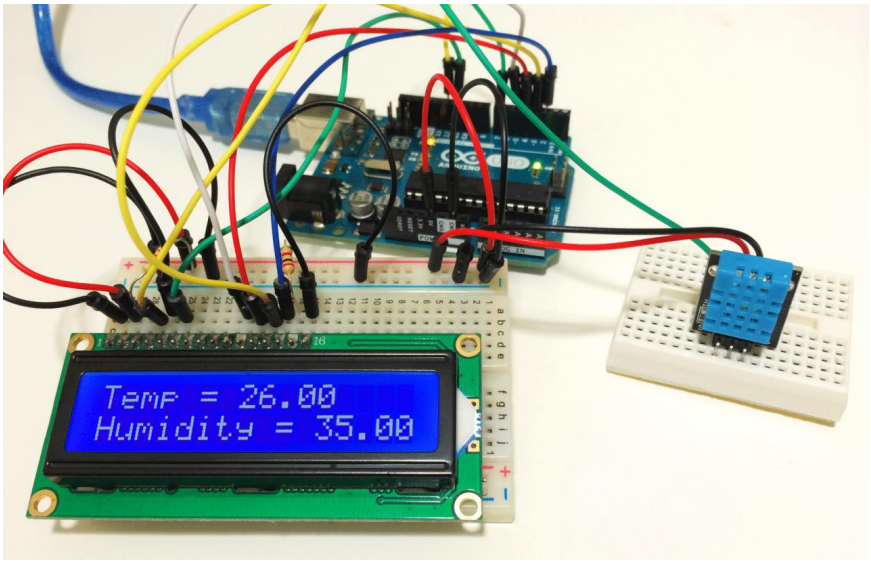
Exercice d’application : Sachant que ce capteur est logique, écrire le code Arduino permettant de lire à partir de ce capteur la présence (0) ou l’absence d’eau (1) puis afficher sur le moniteur série un message significatif.

5. Les Afficheurs LCD

Parfois on a besoin d'afficher les données collectées auprès des capteurs sur un afficheur juste à côté du boitier du système IOT. Sur le marché des afficheurs LCD sont offerts à bas prix. Essentiellement, il y a deux types d'afficheurs : LCD 16x2 et LCD 16x4.



Exemple : Le montage matériel suivant montre bien l'affichage sur un écran LCD la température et l'humidité collecté auprès d'un capteur DHT22 relié à la carte Arduino.



Le tableau qui suit montre le rôle de chacune des 16 broches :

Broche	Nom	Niveau	Fonction
1	Vss	-	Masse
2	Vdd	-	Alimentation positive +5V
3	Vo	0-5V	Cette tension permet, en la faisant varier entre 0 et +5V, le réglage du contraste de l'afficheur.
4	RS	TTL	Selection du registre (Register Select)
5	R/W	TTL	Lecture ou écriture (Read/Write)

6	E	TTL	Entrée de validation (Enable) active sur front descendant. Le niveau haut doit être maintenue pendant au moins 450 ns à l'état haut.
7	D0	TTL	Bus de données bidirectionnel 3 états (haute impédance lorsque E=0)
8	D1	TTL	
9	D2	TTL	
10	D3	TTL	
11	D4	TTL	
12	D5	TTL	
13	D6	TTL	
14	D7	TTL	
15	A	-	Anode rétroéclairage (+5V)
16	K	-	Cathode rétroéclairage (masse)

La technologie TTL (Transistor-Transistor logic) est normalisée pour une tension d'alimentation de 5 V . Un signal TTL est défini comme **niveau logique bas entre 0 et 1,4 V** , et comme **niveau logique haut entre 2,4 V et 5 V** (ces niveaux varient légèrement entre les différentes séries).Les broches 15 et 16 ne sont présentes que sur les afficheurs LCD avec rétroéclairage.

Au niveau du câblage, les informations à afficher peuvent être envoyées en parallèle donc plusieurs fils sont utilisés, ce qui rend plus difficile le brochage et sa programmation avec le mode 8 bits ou mode 4 bits.

5.1 Mode 8 bits

Dans ce mode de 8 bits, les données sont envoyées à l'afficheur sur les broches D0 à D7. On place la ligne RS à 0 ou à 1 selon que l'on désire transmettre une commande ou une donnée. Il faut aussi placer la ligne R/W à 0 pour indiquer à l'afficheur que l'on désire effectuer une écriture. Il reste à envoyer une impulsion d'au moins 450 ns sur l'entrée E, pour indiquer que des données valides sont présentes sur les broches D0 à D7. L'afficheur lira la donnée sur le front descendant de cette entrée.

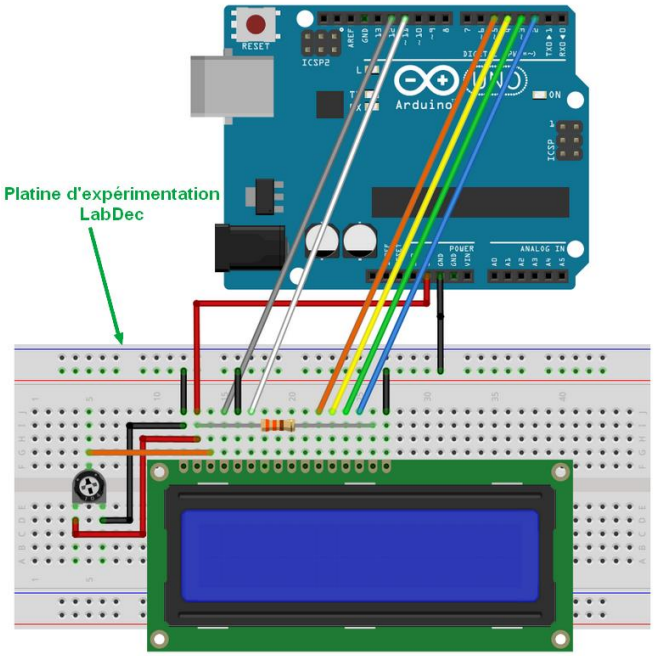
Si on désire au contraire effectuer une lecture, la procédure est identique, mais on place cette fois la ligne **R/W** à 1 pour demander une lecture. Les données seront valides sur les lignes D0 à D7 lors de l'état haut de la ligne **E**.

5.2 Mode 4 bits

Il peut, dans certains cas, être nécessaire de diminuer le nombre de fils utilisés pour commander l'afficheur, comme, par exemple lorsqu'on dispose de très peu de broches d'entrées sorties disponibles sur

un microcontrôleur. Dans ce cas, on peut utiliser le mode quatre bits de l'afficheur LCD. Dans ce mode, seuls les 4 bits de poids fort (D4 à D7) de l'afficheur sont utilisées pour transmettre les données et les lire. Les données sont écrites ou lues en envoyant séquentiellement les quatre bits de poids fort suivi des quatre bits de poids faible. Une impulsion positive d'au moins 450 ns doit être envoyée sur la ligne E pour valider chaque demi-octet.

Remarque : Aux fils de données s'ajoutent quelques fils de contrôle qui permettent d'organiser l'échange des informatieons entre l'Arduino et l'afficheur : A : (+5V) / K : (masse) / R/W : (masse)




```
#include <LiquidCrystal.h>
// initialize the library by associating any needed LCD interface pin
const int RS = 12, E = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(RS, E, d4, d5, d6, d7);
void setup() {
  lcd.begin(4, 20);    // set up the LCD's number of columns and rows:
 lcd.print("hello, DSIR");  // Print a message to the LCD.
}
void loop() {
    lcd.setCursor(0, 1);  // set the cursor to column 0, line 1
  lcd.print(millis()/1000);    // print the number of seconds since reset:
}
```