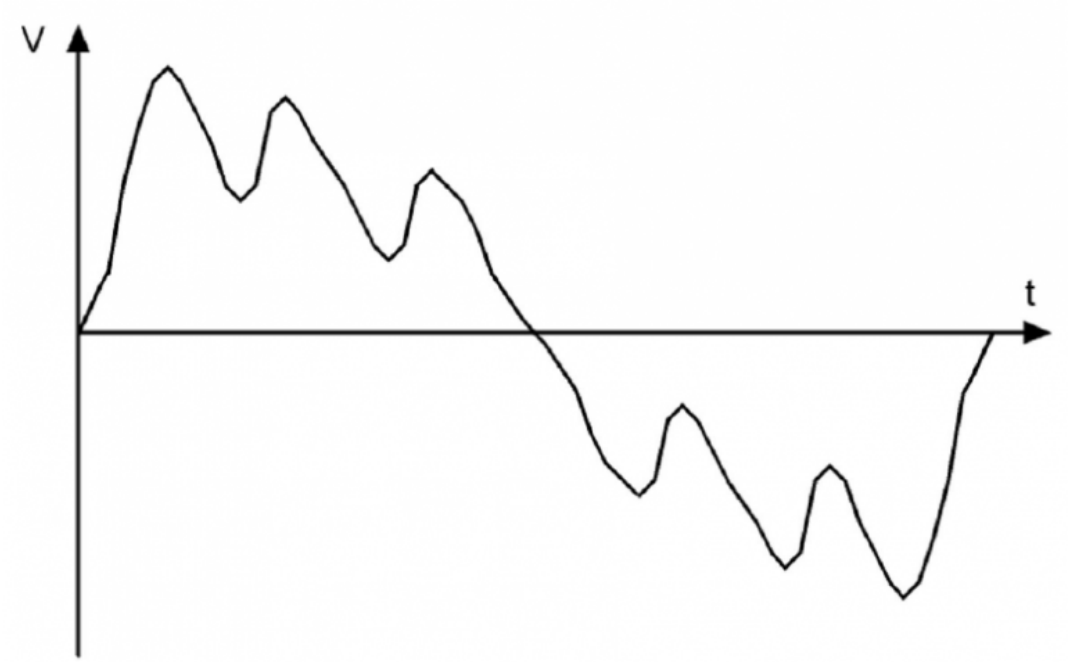
Laboratoire 4 : Convertisseur numérique analogique (ADC)

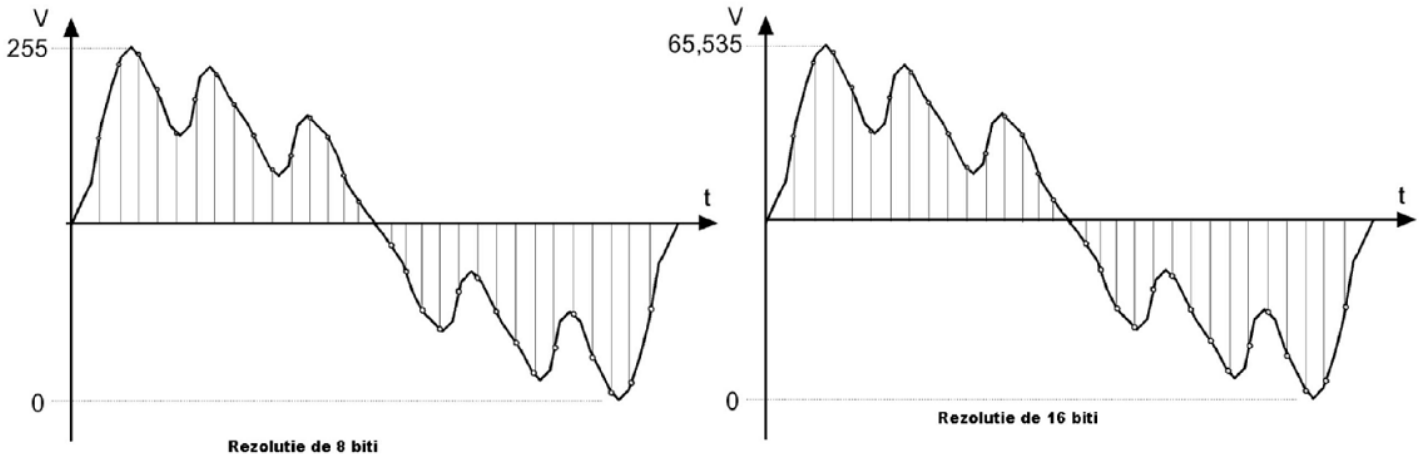
Ce laboratoire vise à vous familiariser avec le travail avec le convertisseur analogique-numérique présent dans le microcontrôleur Atmega328p.

1. Mesure de signaux analogiques

Afin de pouvoir mesurer des signaux analogiques dans un système informatique numérique, ils doivent être convertis en valeurs numériques discrètes. Un convertisseur analogique-numérique (Analog to Digital Converter - ADC) est un circuit électronique qui convertit une tension d'entrée analogique en une valeur numérique.

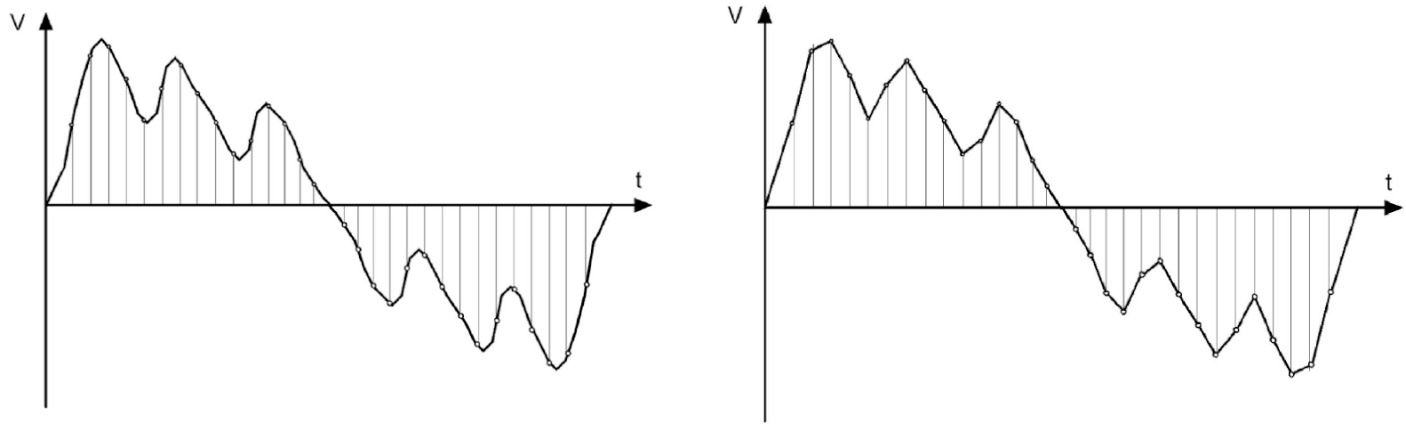


Une caractéristique importante d'un ADC est sa **résolution** . La résolution indique le nombre de bits que le convertisseur peut représenter la sortie binaire. les fournir à sa sortie dans l'intervalle de mesure. **Quantité de mesure** - la plus petite valeur pouvant être distinguée par l'ADC est le rapport entre la plage de tension d'entrée (la différence entre la tension maximale et la tension minimale pouvant être appliquée à l'entrée du convertisseur) et le nombre maximal de binaires représentables valeurs (2^N) .



Par exemple, si la résolution d'un convertisseur est de 10 bits, il peut fournir 2^10 = 1024 valeurs de sortie différentes. Si la plage de mesure est de 0 à 5V, la grandeur de mesure sera : (5V-0V)/1024 = 0,0048V soit 4,8mV.

Une autre caractéristique importante d'un convertisseur analogique-numérique est le taux d'échantillonnage (fr taux d'échantillonnage). Cela dépend du temps entre deux conversions successives et affecte la façon dont la forme d'onde originale sera rendue après le traitement numérique. Ci-dessous, nous voyons comment le signal échantillonné sera reconstruit après avoir traversé un convertisseur numérique-analogique (DAC). Comme on peut le voir, le signal reproduit n'est pas identique à l'original. Si le taux d'échantillonnage devait augmenter, le signal reproduit en joignant les points/échantillons numériques se rapproche de mieux en mieux de l'original.



Mais quel est le taux d'échantillonnage minimum pour reproduire sans perte un signal d'une fréquence donnée ? **Le théorème de Nyquist** stipule qu'un taux d'échantillonnage d' **au moins deux fois la fréquence du signal mesuré**est nécessaire pour cela, le théorème s'applique également à un signal composé de tout un spectre de fréquences, comme la voix humaine ou une chanson. Les limites maximales de l'audition humaine sont de 20 Hz à 20 kHz, mais les fréquences courantes pour la voix se situent dans la plage de 20 à 4 000 Hz, de sorte que les centraux téléphoniques utilisent un taux d'échantillonnage du signal de 8 000 Hz. Le résultat est une reproduction intelligible de la voix humaine, suffisante pour transmettre des informations dans une conversation ordinaire. Pour une reproduction fidèle du spectre audible, des taux d'échantillonnage plus élevés sont utilisés. Par exemple, l'enregistrement sur un CD a un taux d'échantillonnage de 44100Hz ce qui est plus que suffisant pour reproduire fidèlement toutes les fréquences audibles.

Selon la façon dont la conversion est effectuée, les convertisseurs analogique-numérique peuvent être de plusieurs types :

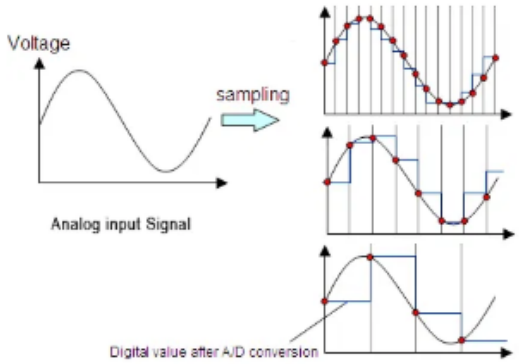
- ADC parallèle (Flash)
- ADC d'approximations successives
- ADC cu integrare (simple pente, double pente);
- ADC Sigma-delta (delta-sigma, 1-bit ADC sau ADC cu oversampling).

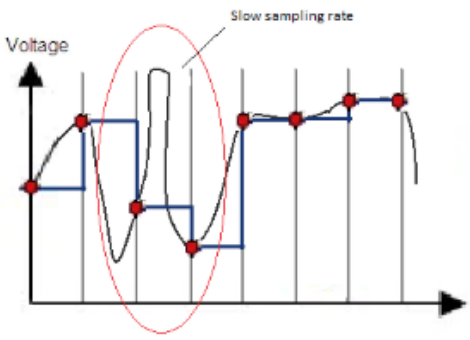
Résolution

The resolution represents the number of discrete values that can be produced over the range of analog values in a measurement period. The discrete values are usually in binary form, so a common way to represent resolution is in audio bit depth (the number of bits of information in each sample). For example, for a resolution of 8 bits, we can encode the analog input signal in 256 different levels, the values ranging from 0 to 255. Resolution can also be expressed in volts, each level representing a fraction of the voltage measurement range. For example, for a range of 0- 5V and an 8-bit resolution, the subdivision would be (5V – 0V) / (2 ^ 8).

Sampling Rate

The sampling rate represents the time at which the values of the analog input signal are read. The time is divided into discrete intervals, each sampling a value from the input signal. The Nyquist frequency tells us that it is mandatory for the sampling rate to be at least twice the frequency of the analog input signal so that the signal can be reconstructed afterwards. There appears a problem where frequencies above half the Nyquist frequency are sampled because they are incorrectly detected as lower frequencies. This process is called aliasing.

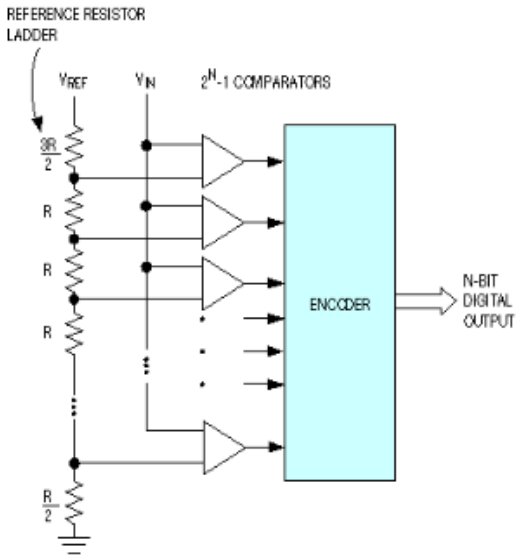




ADC Types

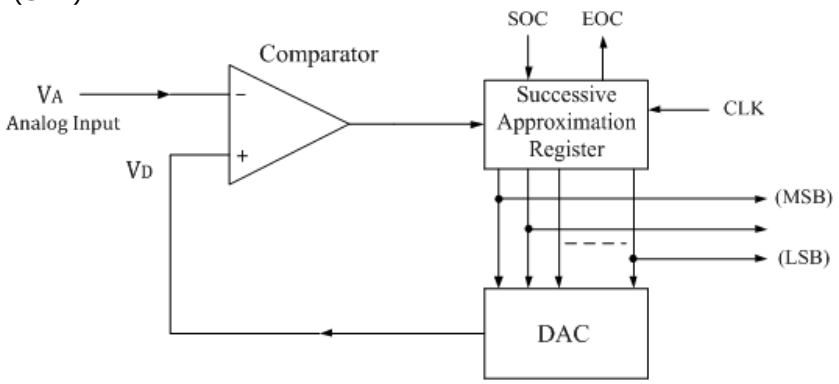
Direct Conversion (Flash ADC)

Has a bank of comparators sampling the input signal in parallel, each firing for their decoded voltage range. The comparator bank feeds a logic circuit that generates a code for each voltage range.



Successive Approximation

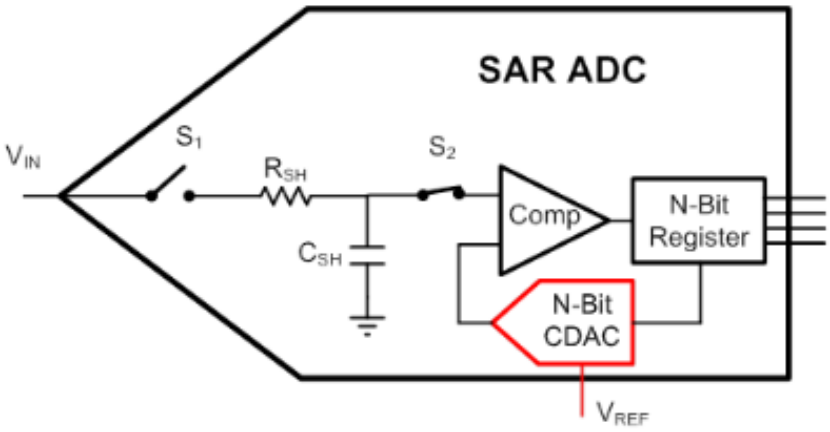
Uses a comparator to successively narrow a range that contains the input voltage. At each successive step, the converter compares the input voltage to the output of an internal digital to analog converter which might represent the midpoint of a selected voltage range. At each step in this process, the approximation is stored in a successive approximation register (SAR).



Successive approximation ADC`s are cheaper than direct conversion ADC`s, and have a smaller footprint on the IC, but they are also slower.

Sigma-Delta

Is a newer design that offers low conversion noise and good resolution. Sigma-delta convertors are appropriate for application that require high resolution, but not the fastest conversion. As stated above, an ADC conversion is not instant, it requires a certain amount of time to complete. This raises a issue: what will happen with the result of the conversion if the input signal will change during this conversion time. The conversion result might not correspond then to the real value of the signal. To solve this, an internal input buffer is used.



When the input signal is sampled, switch S1 is closed, S2 open, so that the Csh capacitor will hold a voltage equal to the input signal. S1 will then open, S2 will close and the conversion will begin. The two switches will hold this state until the conversion will end. Capacitor Csh will hold the (analog) value of the signal while the during the A/D conversion.

2. Le convertisseur ADC de l'Atmega 328p

Le convertisseur analogique-numérique inclus dans le microcontrôleur Atmega328p est un CAN à approximations successives. Il a une résolution allant jusqu'à 10 bits et peut mesurer n'importe quelle tension dans la plage 0-5V à partir de huit entrées analogiques multiplexées (6 disponibles sur l'arduino).

Ce convertisseur peut être contrôlé par deux registres d'état et de contrôle (ADCSRA et ADCSRB) et un registre de bits de sélection de multiplexeur (ADMUX). Dans le premier cas, nous pouvons définir quand convertir, s'il faut rompre à la fin d'une conversion, etc. À l'aide du registre des multiplexeurs, on choisit le canal qui générera l'entrée pour le convertisseur et la tension de référence. De plus, en dehors de ces deux registres, nous avons le registre ADC où le résultat de la conversion (ADC) est écrit.

La relation entre la valeur dans le registre ADC et la tension mesurée est la suivante :

CAN = V_in * 1024 / V_ref

ou

V_in = CAN * V_ref / 1024

où *V_in* est la tension mesurée et *V_ref* est la tension choisie comme référence.

Tension de référence

En fonction de l'intervalle dans lequel varie le signal que nous lisons, nous pouvons sélectionner une autre tension de référence. Ceci est utile pour augmenter la résolution de lecture. L'ADC sur l'Atmega 328p nous donne comme tension de référence la tension d'alimentation (AVCC), une tension interne de 1.1V ou une broche à laquelle on peut connecter une référence de tension externe (AREF)

Prédiviseur

Notre ADC a besoin d'un signal d'horloge pour savoir combien de temps dure une conversion. Parce que le signal d'horloge du microcontrôleur est trop rapide, nous avons besoin d'un prédiviseur. La valeur la plus basse est 2 et la plus élevée est 128.

F_ADC = F_CPU / PRESCALER

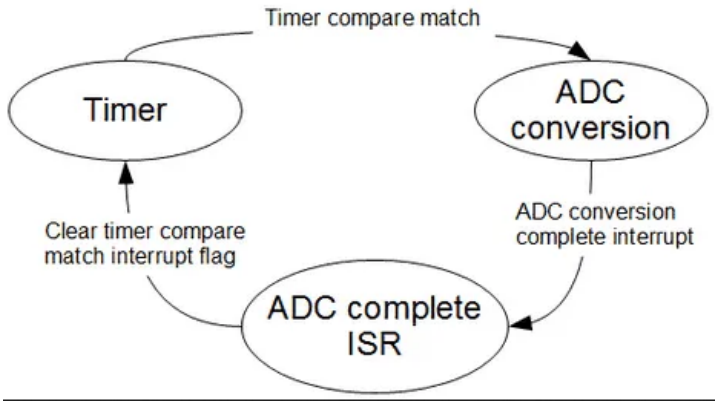
Le choix du prédiviseur dépend de la fréquence d'échantillonnage et de la précision souhaitée. Plus le prédiviseur est élevé, plus la fréquence ADC sera basse et plus la précision sera élevée. Plus d'informations peuvent être trouvées dans le chapitre 23.4 de la fiche technique .

Modes de fonctionnement

Le convertisseur peut fonctionner de plusieurs manières. Le mode le plus courant et celui qui est également disponible dans la bibliothèque arduino est **le mode de conversion unique** . Dans ce mode, une seule conversion sera effectuée et elle commencera par mettre ADSC à 1. Le registre sera mis à 0 automatiquement à la fin de la conversion.

Un autre mode de fonctionnement est **le mode de fonctionnement libre** dans lequel le convertisseur fonctionnera en continu, à la fin d'une conversion, la prochaine conversion démarrera automatiquement. A chaque fois, le résultat précédent sera écrasé. Pour démarrer une conversion, le registre ADSC sera mis à 1 et il ne sera pas mis à 0 automatiquement.

- **Demande d'interruption externe** . Dans ce mode, une conversion commencera sur un front positif d'une broche d'entrée.
- **Modes de comparaison analogique** . De cette façon, l'ADC peut comparer deux signaux analogiques. Nous n'utiliserons pas cette fonctionnalité en laboratoire. Plus de détails au chapitre 23 de la fiche technique .
- **Minuterie de modes** . Pour un contrôle plus précis du moment où la conversion se produit, un événement de minuterie peut être utilisé pour démarrer une nouvelle conversion.



enregistrer

ADMUX - Registre de sélection de multiplexeur ADC

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7:6 - **REFS1:0 : Bits de sélection de référence** → Sélectionne la tension de référence
- Bit 5 - **ADLAR ADC : résultat d'ajustement à gauche** → mode d'alignement des 10 bits. (aligné à gauche ou aligné à droite)
- Bits 3:0 - **MUX3:0 : Bits de sélection de canal analogique** → Sélectionne le port d'entrée à partir duquel la conversion est effectuée

ADCSRA - Contrôle ADC et registre d'état A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN : ADC Enable** → Démarre le convertisseur
- **Bit 6 – ADSC : ADC Start Conversion** → Utilisé pour démarrer une conversion
- **Bit 5 - ADATE : ADC Auto Trigger Enable** → Démarre la manière dont les conversions ADC démarrent automatiquement en fonction d'une source spécifiée dans ADCSRB
- **Bit 4 – ADIF : ADC Interrupt Flag** → Mis à 1 lorsqu'une conversion est terminée. Cet indicateur est effacé automatiquement lorsqu'une routine de gestion des interruptions est appelée.
- **Bit 3 – ADIE : ADC Interrupt Enable** → Démarre les interruptions pour ADC
- **Bits 2:0 – ADPS2:0 : ADC Prescaler Select Bits** → Setare prescaler

ADCSRB - Registre de contrôle et d'état ADC B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 2:0 – ADTS2:0 : Source de déclenchement automatique ADC** → La source à partir de laquelle générer une nouvelle conversion. Dépend de l'ADATE de l'ADCSRA

Exemple

Installation:

```
ADUX = 0 ;
/* ADC1 - canal 1 */
ADMUX |= (1 << MUX0);
/* AVCC avec condensateur externe sur la broche AREF */
ADMUX |= (1 << REFS0);
```

```
ADCSRA = 0 ;
/* définit le prédiviseur à 128 */
ADCSRA |= (7 << ADPS0);
/* activer ADC */
ADCSRA |= (1 << ADEN);
```

Lire:

```
/* lancer la conversion */
ADCSRA |= (1 << ADSC);
/* attend la fin de la conversion */
while (!(ADCSRA & (1 << ADIF)));
uint16_t resultat = CAN ;
```

3. ADC dans Arduino

La bibliothèque arduino nous fournit une fonction simple pour utiliser l'ADC, à savoir **analogRead()** .

```
void loop() {
  val = analogRead(A0); // lit la broche d'entrée
  Serial.println(val); // valeur de débogage
  delay(100);
}
```

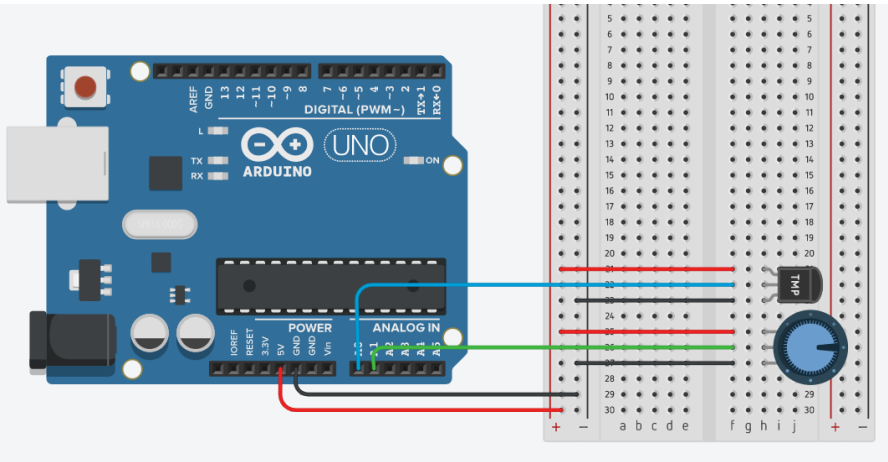
Cette fonction reçoit une broche en paramètre et se bloquera jusqu'à ce que la lecture de la valeur sur cette broche soit terminée. Pour les applications simples, cela suffit, mais pour les applications plus complexes où nous voulons continuer le traitement tout en faisant la conversion, nous devrons utiliser un code spécifique pour notre microcontrôleur.

La tension de référence utilisée peut être définie avec la fonction analogReference()

4. Exercices

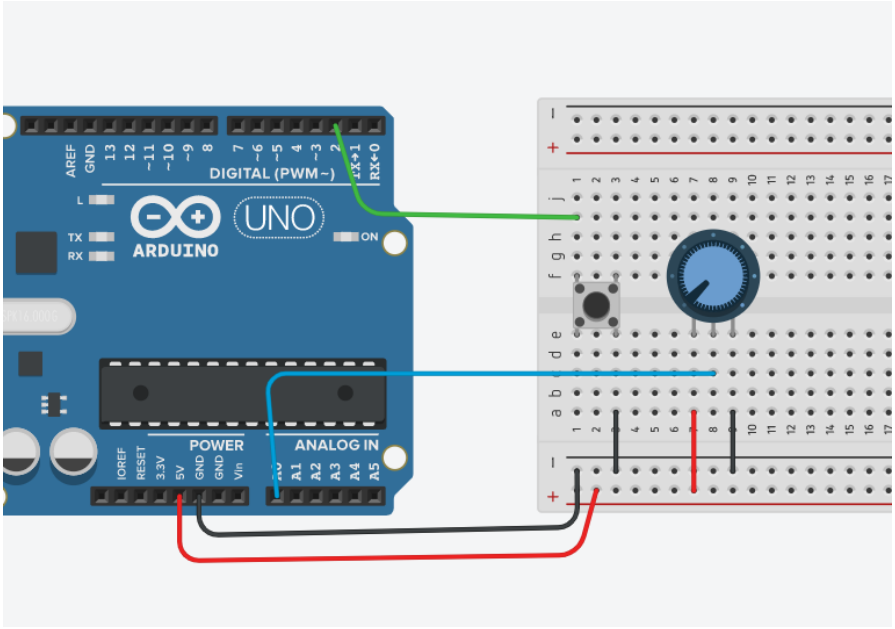
Tâche 0 Utilisez le code Arduino pour lire la valeur d'un potentiomètre et d'un capteur de température, puis envoyez les valeurs en série.

1. Transmettez à la console, en utilisant le port série disponible, la tension (calculée sur le microcontrôleur) de la sortie du potentiomètre et la valeur renvoyée par l'ADC (0-1023).
2. La valeur lue pour le capteur de température doit être exprimée en degrés Celsius. La tension de sortie du capteur varie linéairement avec la température. Vous pouvez faire la conversion expérimentalement ou en utilisant la fiche technique (Datasheet TMP36 [https://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf] / Datasheet LM35 [https://www.ti.com/lit/ds/symlink/lm35.pdf]).



RÉSOUDRE

Tâche 1 À l'aide du code AVR spécifique, lisez la valeur du potentiomètre uniquement lorsqu'un bouton est enfoncé.



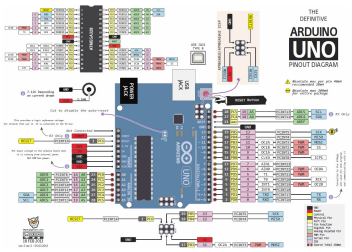
RÉSOUDRE

Tâche 2 À l'aide d'un code AVR spécifique, lisez la valeur du potentiomètre une fois toutes les 10 ms. Pour cet exercice vous devez utiliser l'ADC pour démarrer une conversion automatiquement selon un timer.

RÉSOUDRE

5. Ressources

- Fiche technique Atmega 328p
- Brochage Arduino UNO



- Responsable : Florin Stancu [mailto:florin.stancu@upb.ro]

RÉSOUDRE

6. Liens utiles

- Lecture analogique Arduino [https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/]
- Fiche technique TMP36 [https://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf]
- Fiche technique LM35 [https://www.ti.com/lit/ds/symlink/lm35.pdf]
- Fiche technique ATmega328p [https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf]
- Tinkercad TMP36 [https://www.tinkercad.com/things/cl7vDLkEZwF?sharecode=Ve7hoj8NLhZ_JSmV2jcoruc8lzp35D1E-pg1nTFhrgE]