

mercredi 30 octobre 2024 20:05

Symbole disponible sur Kicad

Prix : 4.16\$

Empreinte - SnapMagic

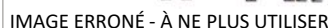
## Informations - Mouser

## IGM - Arduino : getting started

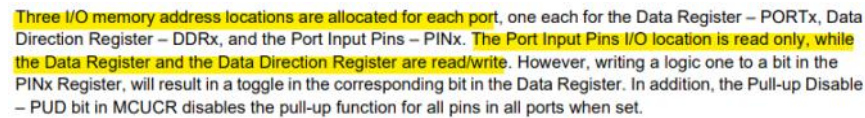
### Pins type - DANKC

## Fiches technique - ATMEGA328P-AU

**Pin layout (p.12):**



### I/O Pin Equivalent Schematic

**Table 28-13. Pin Name Mapping**

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/BSY	PD1	O	0: Device is busy programming, 1: Device is ready for new command
OE	PD2	I	Output Enable (Active low)
WR	PD3	I	Write Pulse (Active low)
BS1	PD4	I	Byte Select 1 ("0" selects Low byte, "1" selects High byte)
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1
PAGEL	PD7	I	Program memory and EEPROM Data Page Load
BS2	PC2	I	Byte Select 2 ("0" selects Low byte, "1" selects 2'nd High byte)
DATA	{PC[1:0]: PB[5:0]}	I/O	Bi-directional Data bus (Output when OE is low)

### 14.2.1 Configuring the Pin

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

## A24 - Inch2Cent Page 2

## 29.1 Absolute Maximum Ratings\*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except RESET with respect to Ground	-0.5V to $V_{CC}+0.5V$
Voltage on RESET with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0mA
DC Current $V_{CC}$ and GND Pins	200.0mA

## 29.2.8 ATmega328P DC Characteristics

Table 29-8. ATmega328P DC characteristics -  $T_A = -40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$ ,  $V_{CC} = 1.8V$  to  $5.5V$  (unless otherwise noted)

Symbol	Parameter	Condition	Min.	Typ. <sup>(2)</sup>	Max.	Units
$I_{CC}$	Power Supply Current <sup>(1)</sup>	Active 1MHz, $V_{CC} = 2V$		0.3	0.5	mA
		Active 4MHz, $V_{CC} = 3V$		1.7	2.5	
		Active 8MHz, $V_{CC} = 5V$		5.2	9	
		Idle 1MHz, $V_{CC} = 2V$		0.04	0.15	
		Idle 4MHz, $V_{CC} = 3V$		0.3	0.7	
		Idle 8MHz, $V_{CC} = 5V$		1.2	2.7	
	Power-save mode <sup>(3)</sup>	32kHz TOSC enabled, $V_{CC} = 1.8V$		0.8		$\mu\text{A}$
		32kHz TOSC enabled, $V_{CC} = 3V$		0.9		
	Power-down mode <sup>(3)</sup>	WDT enabled, $V_{CC} = 3V$		4.2	8	
		WDT disabled, $V_{CC} = 3V$		0.1	2	

- Notes:
- Values with "Minimizing Power Consumption" enabled (0xFF).
  - Typical values at  $25^{\circ}\text{C}$ . Maximum values are test limits in production.
  - The current consumption values include input leakage current.

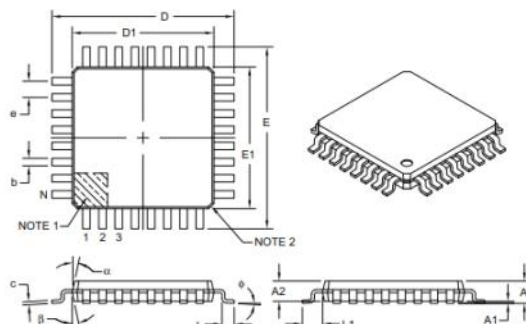
- Although each I/O port can source more than the test conditions (20mA at  $V_{CC} = 5V$ , 10mA at  $V_{CC} = 3V$ ) under steady state conditions (non-transient), the following must be observed:  
ATmega48A/PA/88A/PA/168A/PA/328P:  
1] The sum of all  $I_{OH}$  for ports C0 - C5, D0 - D4, ADC7, RESET should not exceed 150mA.  
2] The sum of all  $I_{OH}$  for ports B0 - B5, D5 - D7, ADC6, XTAL1, XTAL2 should not exceed 150mA.  
If  $I_{OH}$  exceeds the test condition,  $V_{OH}$  may exceed the related specification. Pins are not ensured to source current greater than the listed test condition.
- Although each I/O port can sink more than the test conditions (20mA at  $V_{CC} = 5V$ , 10mA at  $V_{CC} = 3V$ ) under steady state conditions (non-transient), the following must be observed:  
ATmega48A/PA/88A/PA/168A/PA/328P:  
1] The sum of all  $I_{OL}$  for ports C0 - C5, ADC7, ADC6 should not exceed 100mA.  
2] The sum of all  $I_{OL}$  for ports B0 - B5, D5 - D7, XTAL1, XTAL2 should not exceed 100mA.  
3] The sum of all  $I_{OL}$  for ports D0 - D4, RESET should not exceed 100mA.  
If  $I_{OL}$  exceeds the test condition,  $V_{OL}$  may exceed the related specification. Pins are not ensured to sink current greater than the listed test condition.

(p.308)

## Dimensions (p.636):

### 32-Lead Plastic Thin Quad Flatpack (PT) – 7x7x1.0 mm Body, 2.00 mm [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Dimension	Units	MILLIMETERS		
		MIN	NOM	MAX
Number of Leads	N		32	
Lead Pitch	e		0.80 BSC	
Overall Height	A	—	—	1.20
Standoff	A1	0.05	—	0.15
Molded Package Thickness	A2	0.95	1.00	1.05
Foot Length	L	0.45	0.60	0.75
Footprint	L1		1.00 REF	
Foot Angle	α	0°	3.5°	7°
Overall Width	E		9.00 BSC	
Overall Length	D		9.00 BSC	
Molded Package Width	E1		7.00 BSC	
Molded Package Length	D1		7.00 BSC	
Lead Thickness	c	0.09	—	0.20
Lead Width	b	0.30	0.37	0.45
Mold Draft Angle Top	β	11°	12°	13°
Mold Draft Angle Bottom	β	11°	12°	13°

- Notes:
- Pin 1 visual index feature may vary, but must be located within the hatched area.
  - Chamfers at corners are optional; size may vary.
  - Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
  - Dimensioning and tolerancing per ASME Y14.5M.
- BSC: Basic Dimension. Theoretically exact value shown without tolerances.  
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-074B

## Branchements

- XTAL

### Unused XTAL Pins

If XTAL pins are not in use, they should be tied to ground. This helps to prevent unintentional behavior during device start-up.

- RESET

## 29.5 System and Reset Characteristics

Table 29-11. Reset, Brown-out and Internal Voltage Characteristics<sup>(1)</sup>

Symbol	Parameter		Min.	Typ	Max	Units
V <sub>POT</sub>	Power-on Reset Threshold Voltage (rising)		1.1	1.4	1.6	V
	Power-on Reset Threshold Voltage (falling) <sup>(2)</sup>		0.6	1.3	1.6	V
SR <sub>ON</sub>	Power-on Slope Rate		0.01		10	V/ms
V <sub>RST</sub>	RESET Pin Threshold Voltage		0.2 V <sub>CC</sub>		0.9 V <sub>CC</sub>	V
t <sub>RST</sub>	Minimum pulse width on RESET Pin				2.5	μs
V <sub>HYST</sub>	Brown-out Detector Hysteresis			50		mV
t <sub>BOD</sub>	Min. Pulse Width on Brown-out Reset			2		μs
V <sub>BG</sub>	Bandgap reference voltage	V <sub>CC</sub> =2.7 T <sub>A</sub> =25°C	1.0	1.1	1.2	V
t <sub>BG</sub>	Bandgap reference start-up time	V <sub>CC</sub> =2.7 T <sub>A</sub> =25°C		40	70	μs
I <sub>BG</sub>	Bandgap reference current consumption	V <sub>CC</sub> =2.7 T <sub>A</sub> =25°C		10		μA

Notes:  
1. Values are guidelines only.  
2. The Power-on Reset will not work unless the supply voltage has been below  $V_{POT}$  (falling).

- AVCC (p.14)

### AVCC

$AV_{CC}$  is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to  $V_{CC}$ , even if the ADC is not used. If the ADC is used, it should be connected to  $V_{CC}$  through a low-pass filter.

Note that PC6...4 use digital supply voltage,  $V_{CC}$ .

- Analog input voltage (p.246/252)

Figure 24-1. Analog to Digital Converter Block Schematic Operation

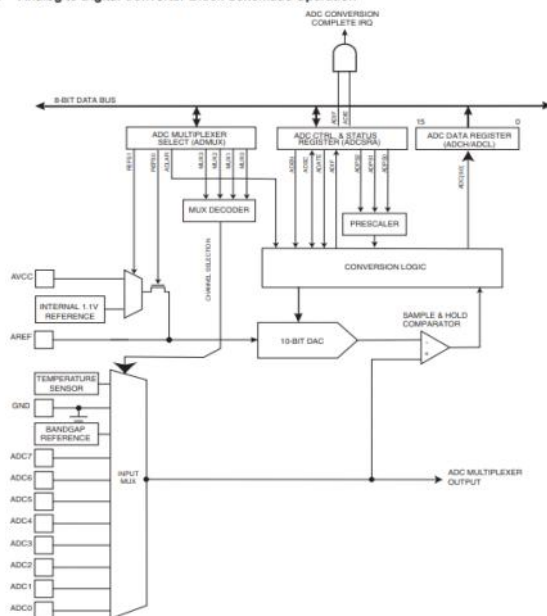
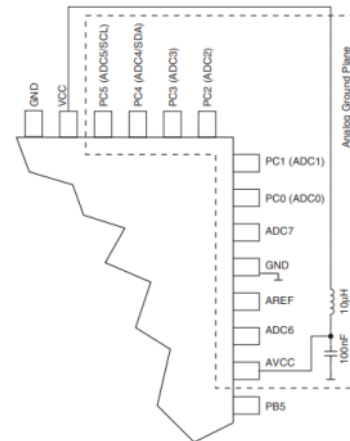


Figure 24-9. ADC Power Connections



The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 24-1 on page 247.

The ADC has a separate analog supply voltage pin,  $AV_{CC}$ .  $AV_{CC}$  must not differ more than  $\pm 0.3V$  from  $V_{CC}$ . See the paragraph "ADC Noise Canceled" on page 252 on how to connect this pin.

Internal reference voltages of nominally 1.1V or  $AV_{CC}$  are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

The Power Reduction ADC bit, PRADC, in "Minimizing Power Consumption" on page 51 must be disabled by writing a logical zero to enable the ADC.

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally,  $AV_{CC}$  or an internal 1.1V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.



#### 24.6.1 Analog Input Circuitry

The analog input circuitry for single ended channels is illustrated in Figure 24-8. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is

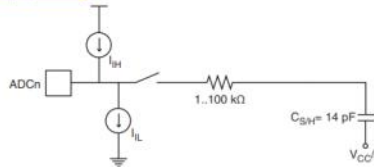
### ATmega48A/PA/88A/PA/168A/PA/328/P

selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10 kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedance sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

Signal components higher than the Nyquist frequency ( $f_{ADC}/2$ ) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 24-8. Analog Input Circuitry



#### ADC Conversion Result

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where  $V_{IN}$  is the voltage on the selected input pin and  $V_{REF}$  the selected voltage reference (see Table 24-3 on page 257 and Table 24-4 on page 258). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

#### • Bit 5:0 – ADC5D...ADC0D: ADC5...0 Digital Input Disable

When this bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC5...0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

Note that ADC pins ADC7 and ADC6 do not have digital input buffers, and therefore do not require Digital Input Disable bits.

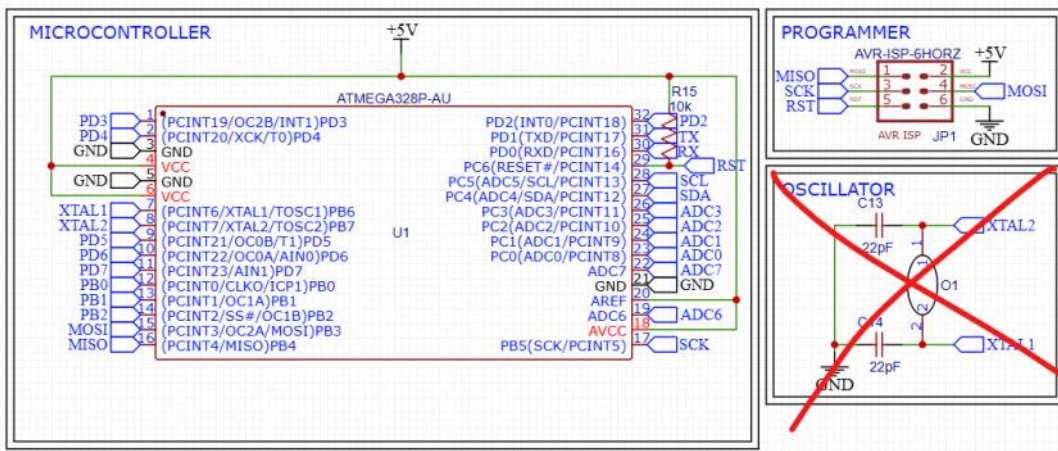
- Digital

#### I/O Numériques

Les fonctions ci-dessous permettent de configurer et d'interagir avec les broches numériques. On remarque que à chaque fois, on précise le numéro de la broche tel qu'il est annoté sur notre circuit.

- `pinMode(pin, mode)` : permet de configurer le comportement (entrée ou sortie) de la broche passée en paramètre.  
Exemple : `pinMode(13, OUTPUT);`
- `digitalWrite(pin, value)` : écrit une valeur (HIGH ou LOW) sur une broche configurée en tant que sortie (OUTPUT).  
Exemple : `digitalWrite(13, HIGH);`
- `digitalRead(pin)` : lit la valeur présente sur la broche spécifiée en paramètre (HIGH ou LOW).  
Exemple : `val = digitalRead(13);`

#### Branchements nécessaires (Miriam):



Minimum:

- Pins de prog



- Les branchements du programmer prennent des pins comme ceux-là et le schéma [ici](#).  
[Branchements \(p.9\)](#)
- MOSI, MISO, SCK, RST
- Recommande un bouton avec un filtre passe-bas pour RST (reset le board au besoin)  
[Branchements \(p.7-8\)](#) Recommande:
- Pins de debug
  - TX et RX (UART)
  - Fonctionne un peu comme serial.print de l'arduino

# Programmation

lundi 25 novembre 2024 19:10

Emplacement du code : C:\Users\elo34\Documents\PlatformIO\Projects\Inch2Cent\src

1. [Programmation ATMEGA328P](#)
2. [Concept](#)

# 1. Programmation ATMEGA328P

jeudi 5 décembre 2024 22:07

## Informations fournies par Miriam:

Utilisation de Microchip ou VS Code pour programmer ou Arduino IDE.

Manipulations afin de setup la plateforme :

Arduino IDE : <https://docs.arduino.cc/built-in-examples/arduino-isp/ArduinoISP/>

Il faut intégrer les bibliothèques Arduino dans les bons fichiers pour les deux logiciels (comme on fait sur VS Code).

Programmation de base en C/C++ comme Arduino

Fuse programming = donner certaines informations à ton atmega


VS Code
<a href="#">Platformio - Atmel AVR</a> <a href="#">Programming AVR Chips Using Visual Studio Code</a>
Localisation du projet : C:\Users\elo34\Documents\PlatformIO\Projects\Inch2Cent
Type de configuration : AVRISP mkII

**Microchip (NOT FONCTIONNAL)**

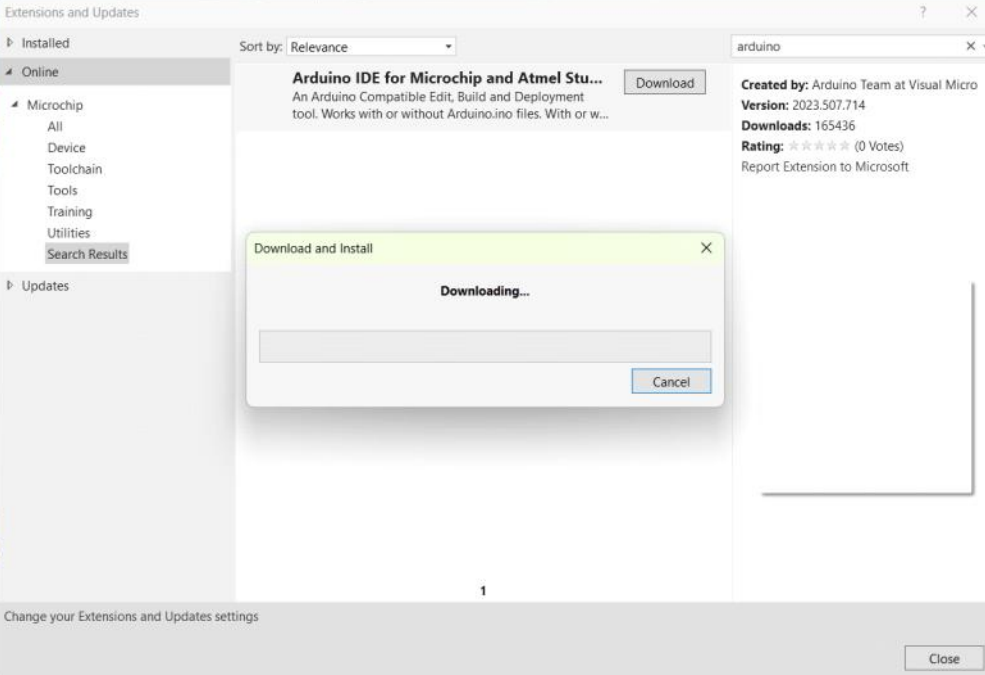
[Microchip Studio for AVR](#)

**2 Getting Started**  
[Creating a new project](#)  
Project Type used : GCC C++ Executable Project (C/C++)  

[Maker and DIY Solutions](#)  
Creating Arduino Sketches in Atmel Studio (p.13)

  
Atmel-4243  
9-From-M...

Impossible de download. Certaines personnes ont le [problème](#) depuis juin.

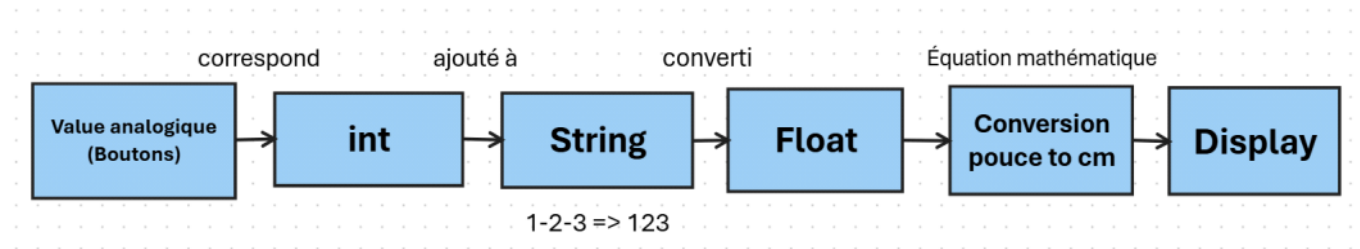




## 2. Concept

jeudi 5 décembre 2024

22:07



1. [Lecture des boutons](#)
2. [Jumeler les chiffres en nombre](#)
3. [Conversion](#)
4. [Display 7 segments](#)
5. [Vider la liste de chiffres](#)
6. [DEL d'incertitude](#)
7. [NOT USED - Modifier le range des boutons en fonction de VCC](#)

### Lecture des boutons

#### [Multiple pushbuttons on one Arduino Input](#)

1. Identifier la pin avec ce qu'elle représente.
2. Initialiser buttonValue = 0;
3. `buttonValue = analogRead(pin);`
4. Boucle de if afin de vérifier le bouton sélectionné  
`if (buttonValue >= 1009 && buttonValue <= 1020)`

#### Déterminer le range pour les if

1. Calculer la tension attendue avec la résistance mise
  2. Calculer le nombre entre 0 et 1023 qui est équivalent
    - $V_{th} / V_{max} * 1023$
  3. Ajouter entre 5 et 20 de chaque côté de la valeur pour faire le range
- \*\* Plus il y a de boutons, plus le range est petit

Voir Conception.xls pour les ranges sélectionnés

NOT USED

Les ranges changent constamment à cause de la tension d'entrée qui varie (batterie se vide)

Remplacer par un code qui calcule les ranges selon VCC : [Modifier le range des boutons en fonction de VCC](#)

### Jumeler les chiffres en nombre

#### Utilisation de String

##### [ARDUINO DOCS - String\(\)](#)

- `String Number = String(char);`
- Append : += (ajout à la fin du String)
  - `myString1 += data;`

##### [ARDUINO DOCS - toFloat\(\)](#)

- Convertir String to a float
- String doit commencer par un digit. Sinon, retourne zéro.
- La fonction arrondit à deux chiffres après la virgule et ignore tous les caractères qui ne sont pas des chiffres ou le point.

- `myString.toFloat();`

<https://forum.arduino.cc/t/how-do-you-convert-a-float-to-string-solved/237090>  
<https://forum.arduino.cc/t/solved-keypad-number-input-and-store/57566>  
<https://www.best-microcontroller-projects.com/arduino-string-to-float.html#:~:text=The%20simplest%20way%20to%20convert%20an%20Arduino%20string,a%20parameter%20and%20returns%20the%20corresponding%20float%20value.>

## Conversion

L'ensemble des valeurs entrées sur le clavier sont en pouce dans le but de les convertir en centimètre.

1 po = 2.54 cm

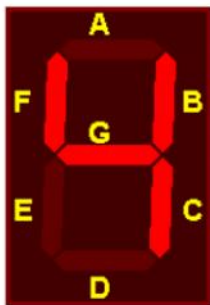
Limitation : [Conversion possible](#)

*Les cas d'affichage pour les conversions en fonction des limitations*

Si lengthToConvert > 390	Retourne 1000	Afficher sur le display --- Allumer DEL
Si lengthToConvert < 390 && lengthToConvert > 39	Retourne la valeur arrondie (aucune décimale) ex : 112.3 => 112	Afficher sur le display le nombre arrondi sans décimale Allumer DEL
Si lengthToConvert < 39	Retourne la valeur arrondie jusqu'à 2 décimale selon la valeur	Afficher sur le display le nombre arrondi avec décimale

## Display 7 segments

[Afficheur LEDS 7seg](#)



On se rend compte qu'il existe plusieurs positions du commutateur SW1 pour lesquels un même segment peut devoir être allumé. Par exemple le segment B doit être allumé quand il faut afficher les chiffres 1, 2, 3, 4, 7, 8 et 9. Le tableau qui suit résume quels segments de l'afficheur doivent être allumés en fonction de la position de SW1 (case avec un x = segment allumé).

Position SW1	A	B	C	D	E	F	G
1		x	x				
2	x	x		x	x		x
3	x	x	x	x			x
4		x	x			x	x
5	x		x	x		x	x
6	x		x	x	x	x	x
7	x	x	x				
8	x	x	x	x	x	x	x
9	x	x	x	x		x	x

### [Comment utiliser un AFFICHEUR 7 segments](#)

- Les branchements DIG permettent d'activer et désactiver chaque digits.
- Les branchements en lettre permettent de contrôler le même segment dans tous les digits.

- Multiplexage : Afficher un digit à la fois avec une rotation rapide pour donner l'impression que tous les chiffres s'affichent en même temps.

1. Identifier chaque pin avec ce qu'elle représente.

2. Setup()

1. Initialiser les pins qui sont présentées en 1. dans setup()

- `pinMode(pin, OUTPUT);`

2. Désactiver tous les digits pour éviter qu'ils s'allument

- Faire en sorte que la différence de potentiel est 0.
- `digitalWrite(digit, HIGH);`

3. Loop()

1. Définir les fonctions utiles à l'affichage

Possible de prendre un nombre à plusieurs digits et de le séparer en chiffre.

- Diviser le nombre en plusieurs variables.

Le code utilisé dans la vidéo.



imake-code  
-afficheur...

Gérer les décimales dans l'affichage (elle n'est pas toujours au même emplacement)



Gestion de l'affichage

- Le display est allumé pendant 15 secondes à l'aide d'une fonction while(). On prend le temps avant avec millis, puis on vérifie dans le while si le temps présent est 15 secondes plus tard ou non.
- L'activation de la DEL d'incertitude se fait dans l'activation des différents digits de l'affichage.

## Vider la liste de chiffres

Effectuer le RESET

NOT USED

Clear the string or the char before using it again : [How to clear of contents of string in Arduino?](#)

All you would need to do to "clear" this string is to assign a null to the first character.  
For example: `foo[0] = '\0';`

`str.remove(0);`

Doit réinitialiser la longueur.

Pour vérifier si String est empty => `str.empty()`;  
Si retourne TRUE => str est empty

Comment procéder pour déclencher le RESET?

Présentement	• Le RESET s'effectue automatiquement après le 15 secondes d'affichage. On ne peut pas déclencher le RESET volontairement, on doit attendre.
À vérifier	• Voir s'il est possible d'implanter un bouton de RESET. • Possiblement lorsque le '.' est sélectionné après 'e'. À voir si le while n'empêche pas la lecture de nouveaux boutons en même temps. • Nécessite peut-être un retravail des fonctions pour le permettre

## DEL d'incertitude

En fonction des limitations : [Conversion possible](#)

Les cas d'affichage sont présents [ici](#).

La DEL est activé si aucune décimale du nombre à afficher est présente. Aussi, si l'écran tombe en erreur (---), elle s'allume. Ces cas sont présents lorsque decimalPosition est égal à 0 ou 3.

Pour ce faire, digitalWrite() est utilisé. La commande d'affichage est intégrée dans l'affichage du display. Ainsi, à chaque fois qu'un digit allume (au 2 millisecondes), la DEL allume aussi.

## NOT USED - Modifier le range des boutons en fonction de VCC

Fonctions nécessaires :

- Déterminer VCC
- Déterminer la tension IO pour chaque résistance
  - $V_{IO} = \frac{V_{CC} \cdot R_{10}}{R + R_{10}}$
- Déterminer la valeur analogique pour cette tension IO
  - $Analog\ value = \frac{V_{IO}}{1023} \cdot V_{CC}$

## NOT USED

Pour déterminer VCC à ce moment-ci : [Measure VCC/Battery Voltage Without Using I/O Pin on](#)

Basé sur ce code:

```
ADMUX = (0x01 << REFS0)      /* AVCC with external capacitor at AREF pin */
      | (0 << ADLAR)          /* Left Adjust Result: disabled
*/
      | (0x0e << MUX0)        /* Internal Reference (VBG) */;
```

Start the ADC and calculate the result in the main while(1).

```
float Vcc_value = 0 /* measured Vcc value */;
uint16_t ADC_RES_L = 0;
uint16_t ADC_RES_H = 0;

while(1) {
    if (ADCSRA & (0x01 << ADIF)) /* check if ADC conversion complete */
    {
        ADC_RES_L = ADCL;
        ADC_RES_H = ADCH;
        Vcc_value = ( 0x400 * 1.1 ) / (ADC_RES_L + ADC_RES_H * 0x100) /* calculate the Vcc value */;
    }
}
```

- Étant donné que ce sont toujours les mêmes références et inputs channel, ADMUX peut être mis dans setup()
- La deuxième section du code peut être mise dans une fonction. Si l'on veut déterminer VCC à un moment précis, il faut retirer la loop while. Sinon, le code va rester pris dans une boucle

sans fin.

# Créer de nouveaux fichiers pour les fonctions

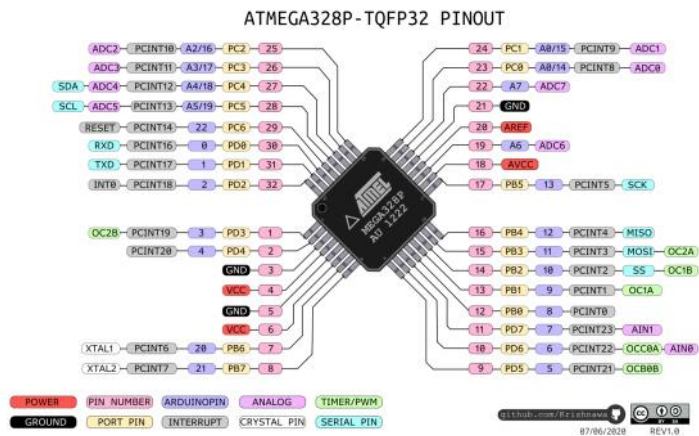
mardi 3 décembre 2024 19:34

[Utilisez plusieurs fichiers - Apprenez à programmer en C++ - OpenClassrooms](#)



# Programmer un chip avec Arduino.h

samedi 21 décembre 2024 16:22



Le numéro de pin de la chip ne correspond pas au numéro de pin à mettre dans le code.

Physiquement, une pin peut être nommée 9, mais dans Arduino c'est plutôt DI5 (la bulle mauve dans l'image).

Pour nommer les pins dans le code, il faut se fier à cette image.

# Spécifications projet

lundi 25 novembre 2024 20:01

## Limitations

- Conversion possible
  - Si l'on veut un nombre précis (avec une décimale) => rester sous 39 po (100 cm)
  - Si l'on veut un nombre sous 4 digits avec arrondissement (possible à mettre sur le display) => gros max 390 po (990.6 cm)