Ateliers CREPP

Atelier Environnement Arduino

Club de Robotique et d'Electronique
Programmable de Ploemeur

16 mars 2022
13 pages

Table des matières

SECTION 1

PRÉAMBULE

- Document réalisé en LaTeX par Nicolas Le Guerroué pour le Club de Robotique et d'Electronique Programmable de Ploemeur (CREPP)
- Permission vous est donnée de copier, distribuer et/ou modifier ce document sous quelque forme et de quelque manière que ce soit.
- Version du 16 mars 2022
- Taille de police : 11pt

06.20.88.75.12

nicolasleguerroue@gmail.com

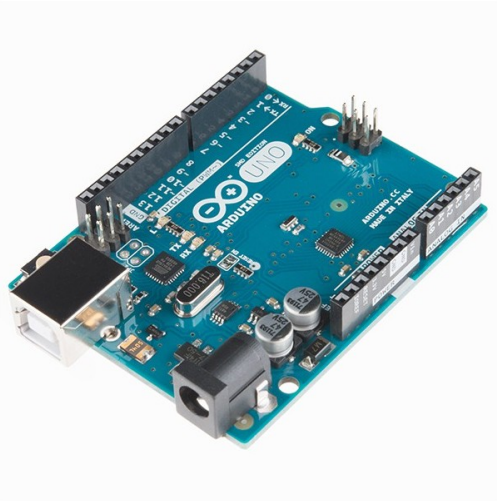
- **Dans la mesure du possible, évitez d'imprimer ce document si ce n'est pas nécessaire. Il est optimisé pour une visualisation sur un ordinateur et contient beaucoup d'images.**

Versions

octobre 2021	Fusion des supports d'ateliers
novembre 2021	Ajout de l'atelier sur les servomoteurs
décembre 2021	Ajout de l'atelier sur les moteurs pas-à-pas
janvier 2022	Ajout de l'annexe pour l'installation des bibliothèques ESP8266
février 2022	Ajout de l'annexe pour le serveur Web ESP8266 NodeMCU

Première partie

L'environnement Arduino



Présentation de l'environnement Arduino et de son langage

SECTION 2
INTRODUCTION

Ce document vise à présenter le projet Arduino et ses supports
Ce tutoriel a pour but également de présenter certaines possibilités d'Arduino en terme de langage et de ressources.
Bien évidemment, cette section n'est pas du tout exaustive.

Origines

Arduino est née en 2004 sous l'impulsion d'étudiants italiens souhaitant promouvoir l'accès à l'électronique. Ils se rencontraient fréquement dans un bar pour développer leur projet.
Aujourd'hui, Arduino c'est :

1. Un langage de programmation basé sur le C++
2. Une communauté
3. Un projet Open-Source

Supports

Arduino disposant d'une communauté assez vaste, de nombreux supports existent.
Nous avons notamment le site officiel d'Arduino à l'adresse suivante :
arduino.cc/Reference/en

Le langage Arduino est compatible avec les instructions du C++ dans la mesure ou le compilateur pour Arduino est g++. Ainsi, les types compososés comme les structures et classes sont supportés, tout comme le mot clé auto par exemple.

SECTION 3

PRÉSENTATION

Nous utilisons des cartes Arduino Uno, basées sur les microcontroleurs Atmega-328 du fabricant ATMEL.
Les microcontrôleurs sont des unités contenant dans un seul boitier une mémoire, un processeur et des interfaces entrées-sorties pour ne citer que ces élements.
Cela permet notamment de dialoguer avec des périphériques [1]

Le microcontroleur

Alimentation

Tension d'alimentation

Le microcontroleur doit être alimenté entre 1.8 V et 5.5 V.
Il existe deux façon d'alimenter la carte Arduino :

1. Via le port USB Le port USB délivre du 5V régulé avec un courant maximal de 500 mA (cas général)
2. Via la broche Vin (connectique Jack femelle) La carte Arduino possède un régulateur intégré de tension en 5 V, ce qui permet d'alimenter la carte entre 7V et 20V

Courants d'entrées-sortie

Fréquence d'horloge

La carte Arduino comporte un oscillateur de 16 MHz même si en interne du microcontroleur, un oscillateur de 8 Mhz est intégré. Cela donne une idée des performances maximales de l'Arduino.

Mémoire

1. Voire section Protocoles de communication

Ce microcontroleur dispose de clé **32 ko** de clé **mémoire flash**, c'est à dire la mémoire pour stocker le programme téléversé vers la carte.

Quand à la clé **mémoire vive (SRAM)**, elle est de **2 ko** et est utilisée pour les variables du programme en cours d'éxécution.
Cette mémoire peut être donc vite saturée lors de l'utilisation de grands tableaux par exemple.

Enfin, le possède une mémoire effacable electriquement, appelée **EEPROM** [2], lors de l'éxécution du programme.
Cette mémoire occupe **1 ko** et chaque registre de cette mémoire, pouvant stocker un nombre codé sur 8 bits (type byte ou char), peut être modifiée 100 000 fois avant son arrêt définitif.

Caractéristiques électriques

Le microcontrolleur dispose d'entrée sorties permettant d'intéragir avec des périphériques (Diodes electroluminescentes, capteurs, modules de communication. . .)
Les entrées sont deux types :

1. entrée **numérique** : la valeur lue sera perçue comme un niveau logique 0 ou 1 sur les broches allant de 1 à 13
2. entrée **analogique** : Un Convertisseur Analogique-Numérique 10 bits est intégrés sur les broches A0, A1, A2, A3, A4 et A5 De ce fait, le CAN est possède une résolution de 4.84 mV avec une référence de tension à 5V [3]

Astuce : Il est possible de configurer les broches analogique en broches digitales.

2. Référence : arduino.cc/Reference/EEPROM
3. Il est également possible de changer la tension de référence de la carte Arduino arduino.cc/Reference/en/language/function/analog-io/analogreference

SECTION 4 LE LANGAGE

Les types

Par défaut les types sont signés, c'est à dire que la plage de valeur pour un nombre codé sur n bits est compris entre $\frac{-2^n}{2}$ et $\frac{2^n+1}{2}$ Pour définir un type non signé, c'est à dire pour agrandir la plage positive, il suffit d'ajouter le mot clé **unsigned** avant les types concernés.

Les types supportés

1. **byte** *[1 octet]*
   Désigne la plus petite unité de mémoire allouable, permettant de stocker un nombre entier compris entre -127 et + 127.

2. **unsigned byte** *[1 octet]*
   Idem mais la plage de valeur strictement positive

3. **int** *[2 octets]*
   Permet de stocker un nombre entier compris entre -32536 et +32536

4. **unsigned int** *[2 octet]*
   Idem mais la plage de valeur strictement positive

5. **float** *[4 octets]*
   Permet de manipuler des réels

6. **double** *[4 octets]*
   Idem, mais ce type est plus précis que le type float et demande plus de ressources au micro-controleur.

7. **char** *[1 octets]*
   Ce type est utilisé pour stcoker et traiter des caractères de la table ASCII.

8. **String** *[4 octets]*
   String est un type élaboré qui permet de traiter des chaines de caractères.

Les types non supportés

Les types **vector**, **array** et **tuple** ne sont pas supportés par le langage Arduino.

Les fonctions

Les fonctions mathématiques

En ce qui concerne les fonctions mathématiques, les fonctions trogonométriques sont incluses.

SECTION 5

LES BROCHES D'INTERRUPTION

Dans certains cas, il est souhaitable de récuperer la valeur d'une broche à tout moment du programme, même quand celui ci est occupé dans une tâche et même dans une fonction de temporisation [1].
Pour remédier à ce problème, on peut utiliser les **broches d'interruption** qui permettent de récuperer la main sur l'ensemble du programme lorsque'un évènement survient sur une broche.

Concrètement, lorsque un évènement e survient sur la broche b, la fonction f est appelée, quelque soit l'état du programme principal.

Prenons le cas d'un bouton qui doit changer l'état d'une LED à n'importe quel moment du programme.

```
int ledPin = 13;    //Led interne
int BOUTON = 2;  //Bouton relié à la broche 2 avec une résistance de charge

volatile int state = LOW;  //Etat courant de la LED

void setup() {

  Serial.begin(9600);//Vitesse de communication à 9600 bit/s

  pinMode(ledPin, OUTPUT);                //Led mise en sortie
  pinMode(BOUTON, INPUT_PULLUP);    //Bouton mis en entrée

  attachInterrupt(digitalPinToInterrupt(BOUTON), onEvent, CHANGE);  //Appel de
 la fonction onEvent à chaque changement de front du bouton
  Serial.println("Init");
}

void loop() {

 delay(5000); //Pause du programme principal
```

1. Voir delay(), delayMicroseconds()

```
}

void onEvent() {

  state = !state; //Inverse l'état de la LED

  if(state){
  Serial.println("ON");
  }else{
  Serial.println("OFF");
  }
   digitalWrite(ledPin, state); //Met à jour l'état de la LED
}
```

Code d'exemple

Ici, quelque soit l'action effectuée dans la fonction loop, dès qu'un front montant est détecté sur la broche BOUTON (2), la fonction onEvent() sera exécutée et changera l'état de la LED à chaque front

Mode d'interruption

Il existe différents modes pour les broches :

- RISING : front montant
- FALLING : Front descendant
- CHANGE : Front montant et descendant

Chronogrammes d'interruption

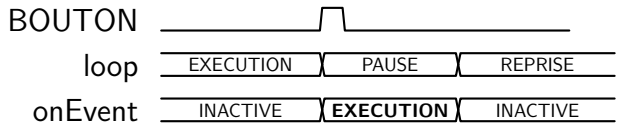


FIGURE 5.1 – Exemple avec mode RISING

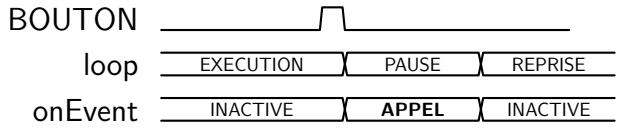


FIGURE 5.2 – Exemple avec mode FALLING

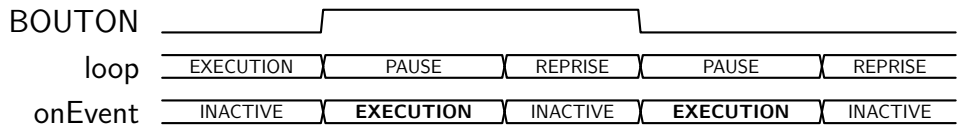


FIGURE 5.3 – Exemple avec mode CHANGE

SECTION 6

SYNTHÈSE ARDUINO

Matériel

1. Mémoire Flash : 32 ko
2. Mémoire Vive (SRAM) : 2 ko
3. Fréquence d'horloge : 16 MHz

Électriques

1. Impédance d'entrée : $> 1MOhm$
2. Courant de sortie par broche : 40 mA maximum
3. Courant de sortie pour toutes les broches entrée-sorties : 200 mA