

Yann Dos Santos

Arduino Robot

Matura travail en informatique avec Mathias Bolli

2020 Ecole cantonale de Frauenfeld

# Résumé des chapitres du dossier

## Chapitre 1: Le cerveau robot

Le premier chapitre est consacré la fondation de l’IDE Arduino et son fonctionnement. L’IDE sert de pont entre le programmeur et le robot. Il vous permet d’entrer du code dans la langue C pour le compiler, puis de l’envoi l’Arduino via le port en série.

Le premier exercice du dossier consiste à allumer la carte Arduino, à la connecter à l'ordinateur et à demander au robot d'afficher "Hello World" sur le moniteur série.

Tout ce que vous voulez commander au robot est d'abord compilé par l'IDE Arduino, puis envoyé au robot sous forme de bits via le port série. Le port série peut être utilisé pour envoyer et recevoir des données, la vitesse à laquelle les données sont envoyées est appelée "débit en bauds" (bits par seconde). Dans cet exercice, nous avons initialisé le port série avec la commande « **Serial.begin(9600)** » à une vitesse de 9600 bauds, puis nous avons utilisé la commande « **Serial.println("Hello world ! \n")** » pour envoyer le texte au moniteur série.

On verra plus tard que le code de l’Arduino est organisé en deux parties :

* La procédure « setup » : dans cette partie, le code ne s’exécutera qu’une fois au début du programme. C’est dans cette partie qu’on viendra faire les déclarations et initialiser le port série et les inputs du robot.
* La procédure « loop » : dans cette partie, le code se répétera indéfiniment. C’est ici que nous écrirons le programme principal, de manière général les robots veulent effectuer des tâches en boucle.

Pour utiliser et stocker des variables on doit d’abord les déclarer et préciser quel type on veut utiliser, le microcontrôleur va ensuite stocker ces données sous forme binaire dans sa RAM. Il existe beaucoup de formes de variables mais voici les plus courantes :

* Les « int » : ils contiennent un nombre entier situé entre -32768 et 32767, si notre valeur ne se trouve pas dans cet intervalle une erreur apparait.
* Les « char » : ils représentent un charactère sous forme de nombre grâce au standard ASCII (American Standard Code for Information Exchange)
* Les « float » : ils contiennent un nombre décimal arrondi situé entre ≈ - 3.4 ∙ 1038 et ≈ 3.4 ∙ 1038

Pour définir une variable on écrit « *type nom = valeur* » :

int a **=** 52**;**

Ou on peut d’abord définir le type de variable et ensuite lui assigner sa valeur. Cette méthode est plus pratique car on peut à tout moment modifier la valeur juste avec son nom au lieu de remonter tout le code pour la modifier.

int a**;**

a **=** 52**;**

On pourra ensuite faire des opérations avec ces variables comme l’addition, la soustraction, la division, la multiplication, le modulo, etc…

Dans chaque programme il est nécessaire d’utiliser des conditions, ce sont des parties de code qui s’exécuteront si un prérequis est remplis, par exemple un bouton pressé ou une certaine valeur de variable. On utilisera les fonction « if » et « else » qui vont d’abord évaluer si la condition entre les parenthèses est vraie et ensuite décider d’exécuter le code entre les accolades ou pas. Si la condition de la fonction « if » est fausse, le programme exécutera le code de la fonction « else ».

**if(**a **>** b**)** //Wenn die Bedingung wahr

**{** // Ausführen

Serial**.**print**(**"a ist grosser als b"**);**

**}**

**else** // Ansonsten

**{** // Ausführen

Serial**.**print**(**"b ist grosser als a"**);**

**}**

On a par la suite appris à utiliser des boucles avec la fonction « while », comme pour « if », le programme va évaluer si la condition entre les parenthèses est vraie, et tant qu’elle est vraie, le code entre les accolades s’exécutera, à l’inverse de « if » ou le code ne s’exécutais qu’une seule fois. Par exemple une boucle « *while (true)* » exécutera son code à l’infini.

Enfin, pour que le code reste organisé, on peut utiliser des commentaires (texte en vert). Ces commentaires sont nécessaires pour documenter ce que fait le code, par exemple si c’est un gros projet c’est utile pour se souvenir de ce qu’on a fait ou si on doit donner le code à quelqu’un d’autre. Les commentaires ne sont jamais envoyés au programme.

Le programme suivant illustre tout ce que nous avons appris dans ce premier chapitre :

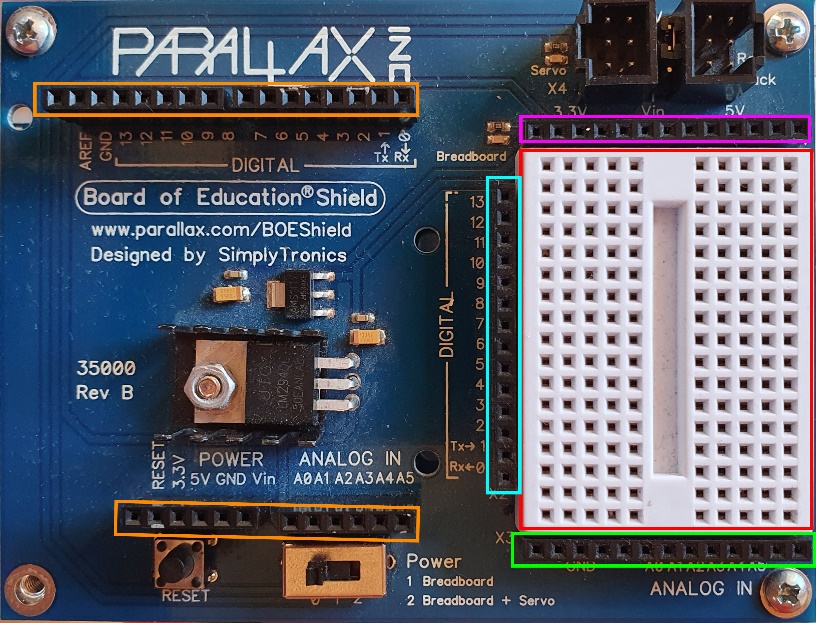
## Chapitre 2 : Construire et tester des indicateurs LED

Ce deuxième chapitre est consacré aux outils physiques du Boe-Bot comme des LEDs ou les servos moteurs (abrégés « servos).

Une LED, diode électroluminescente, est un semi-conducteur, qui émet de la lumière quand il est alimenté par un courant constant, en général 20mA. Ces LEDs seront utiles à signaler ce qu’il se passe à l’intérieur du dispositif aux personnes externes, par exemple une LED peut s’allumer quand le robot se met en marche et s’éteindre quand le robot n’effectue aucune action

Un servo est un moteur que nous allons piloter à l’aide de consigne de position ou de vitesse. Les consignes se trouveront dans notre code.

Pour commencer il a fallu assembler le Boe-Bot. L’Arduino va venir se fixer sous la PCB du Boe-Bot, appelé un « Shield », et va rediriger les entrées de l’Arduino sur le Shield du Boe-Bot. Sur ce Shield il y a ce qu’on appelle une plaque d’expérimentation (breadbord), cela va nous permettre de créer des circuits simples qui nécessitent aucune soudure. Autour de la breadbord il y a les connecteurs nécessaires à faire fonctionner un circuit : alimentation 5V, alimentation 3.3V, la tension des batteries et le connecteur « Ground » qui est le 0V du circuit. Il y a également des entrées/sorties digitales qui vont fonctionner en binaire (HIGH ou LOW : 5V ou 0V). Il y a aussi les entrées/sorties analogues qui vont convertir un signal de 0V à 5V en valeur numérique.



Alimentation

Breadboard

Entrées/Sorties digitales

Entrées/Sorties analogues

Broches de la carte Arduino

Dans la première activité nous allons utiliser des LEDs dans notre circuit. Pour alimenter correctement une LED nous devons utiliser une résistance qui va limiter le courant qui passe à travers la LED pour ne pas l’endommager.

Avec la loi d’Ohm on peut facilement calculer la valeur de la résistance nécessaire pour alimenter une LED de 2V sans dépasser 20mA :

Par sécurité nous utiliserons une résistance de 220 Ω pour ne pas endommager la LED. Le circuit que nous allons utiliser est composé de 2 résistances de 220 Ω chacune et de 2 LEDs rouges. Chaque résistance va être reliée à une LED et à une sortie digitale de l’Arduino.

Pour changer l’état d’une broche, il faut utiliser la commande digitalWrite (pin, état). « Pin » étant le numéro de la broche à laquelle la LED est connectée, et « état » représente la tension de la broche (LOW 0V, HIGH 5V).

Le programme ci-dessous permet d’allumer une LED à la fois toutes les 500 millisecondes.

void setup**()** **{**

pinMode**(**12**,** OUTPUT**);** // Ausgangspins 12 und 13

pinMode**(**13**,** OUTPUT**);**

**}**

void loop**()** **{**

digitalWrite**(**12**,** 0**);** // Pin 12: 0V

digitalWrite**(**13**,** 1**);** // Pin 13: 5V

delay**(**500**);** // Warten 0,5s

digitalWrite**(**12**,** 1**);** // Pin 12: 5V

digitalWrite**(**13**,** 0**);** // Pin 13: 0V

delay**(**500**);** // Warten 0,5s

**}**

Les servomoteurs sont contrôlés par des impulsions carrées. À l’intérieur du servomoteur il y a un potentiomètre qui va servir à régler le moteur pour qu’à une impulsion de 1500µs le moteur soit arrêté. Pour cela il suffit d’écrire un programme qui indique au robot d’envoyer des pulsions de 1500µs aux servomoteurs, et ensuite il faut régler le potentiomètre jusqu’à ce que le moteur ne tourne plus. Les rangées d’impulsion que le robot peut traduire vont de 1300µs à 1700µs, 1500µs étant le paramètre ou le robot ne bougera plus. 1300µs et 1700µs sont les valeurs à laquelle le moteur tournera à pleine vitesse mais définit le sens de rotation des moteurs.

Voici le code utilisé pour régler le potentiomètre :

#include <Servo.h>

Servo servoLeft**;**

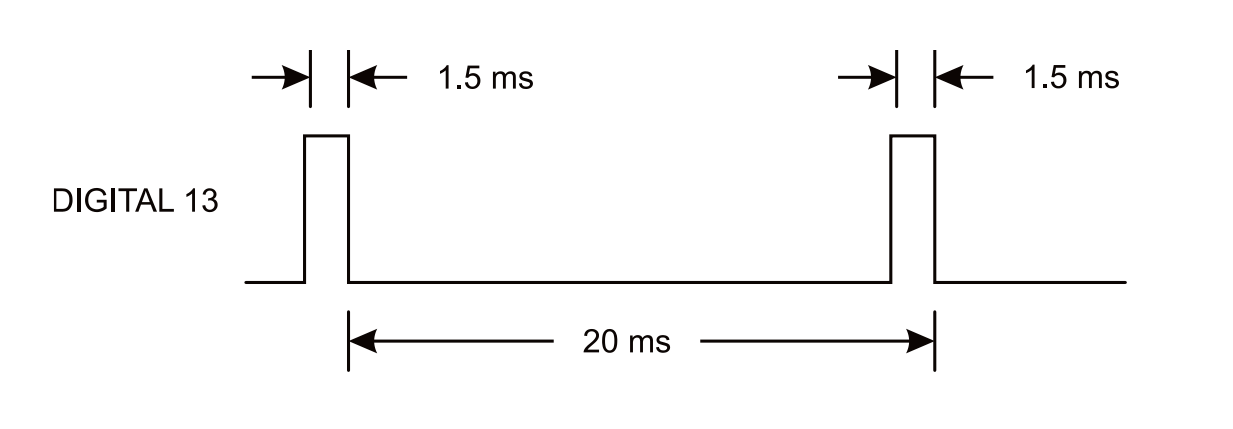
void setup**()** **{**

servoLeft**.**attach**(**13**);**

servoLeft**.**writeMicroseconds**(**1500**);**

**}**

Ce graphique montre le signal envoyé aux servomoteurs



## Chapitre 3 : Assemblages et tests du Boe-Bot

Ce chapitre est consacré à la construction du robot et quelques tests basiques. Une fois que toutes les pièces seront assemblées le robot sera prêt à rouler quand on le lui ordonnera.

En utilisant le code du chapitre précédant, on peut faire se déplacer le robot grâce à la fonction « <servo>.writeMicroseconds(x) ». Grâce à cette fonction et aux deux roues, on peut faire avancer, reculer, tourner le robot dans la direction et la vitesse désirée, nous sommes totalement libres de la manière dont le robot va avancer. Et grâce aux batteries, une fois que le code sera injecté dans la RAM du robot, il n’aura plus besoin d’être relié par câble à un ordinateur.

Ensuite nous apprendrons à créer un circuit simple qui émettra un son quand le robot démarrera. Pour créer un son, il nous faut utiliser un « transducteur piézo-électrique », c’est un objet constitué d’un quartz à l’intérieur qui vibrera à une certaine fréquence donnée pour produire un son plus au moins aigu. Grâce à la fonction « tone(pin, fréquence, durée) » de l’Arduino il sera possible de choisir la fréquence et la durée du son avec la fonction « delay(ms) ».

Voici le code utiliser pour produire le son :

void setup**()** **{**

tone**(**4**,** 3000**,** 1000**);**

delay**(**1000**);**

**}**

void loop**()** **{**

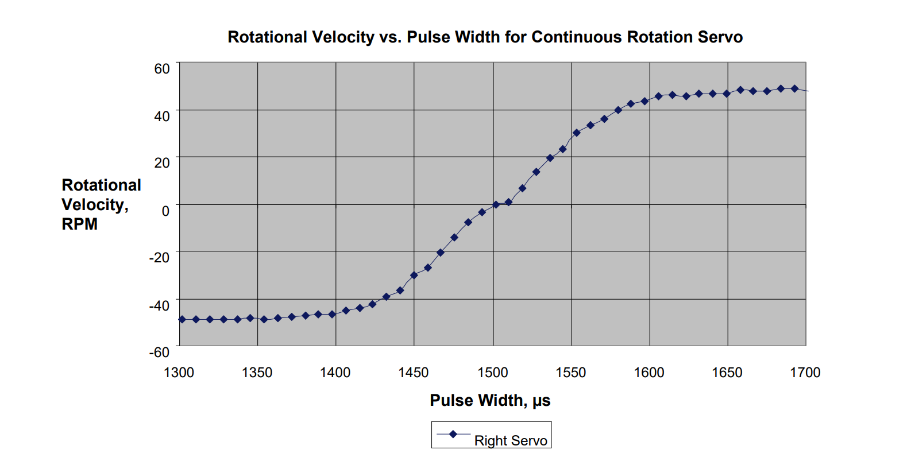
Serial**.**println**(**"Waiting for reset..."**);**

delay**(**1000**);**

**}**

Ici la fonction tone produira un son de 3000Hz[[1]](#footnote-1) pendant 1 seconde. Le robot attendra ensuite encore 1 seconde avant de passer à la partie loop du code.

Dans la partie loop, le robot attends simplement que l’utilisateur appuie sur le bouton physique « reset » sur le shield du robot. Ce bouton aura pour but d’ordonner au robot de reprendre le code depuis le tout début, dans ce cas appuyer sur le bouton reset fera en sorte que le robot émette à nouveau un son.

Par la suite, nous apprendrons à contrôler la vitesse des servos. Les servos sont pilotés par une modulation de largeur d’impulsion (PWM : pulse with modulation). C’est une impulsion de fréquence fixe, l’impulsion définit la vitesse du moteur. L’inconvénient est que la vitesse de rotation de la roue ne suit pas de façon linéaire la longueur de l’impulsion. La vitesse en fonction de la longueur de l’impulsion est assez linéaire entre 1400µs et 1600µs, mais une fois ces valeurs dépassées, la courbe tends vers une vitesse maximale et minimale. Il est important de prendre ce décalage en compte quand on veut définir une vitesse précise à notre robot.

1. Le Hertz (Hz) est une mesure de fréquence et correspond au nombre de répétitions d’un signal pendant une seconde. [↑](#footnote-ref-1)