

Technical Info

Arduino: "Morse Voltmeter" Arduino: "Voltmètre Morse"

Door/par ON7DQ – Traduit par ONL11783

Ooit was ik begonnen met het leren programmeren van een PIC-processor in assembler. Dat ging niet zo vlot, er kwamen andere dringende zaken en de PIC vloog weer in de kast. Later leerde ik de Arduino kennen en toen de Oostendse radioclub een groepsaankoop deed, ben ik ook bezweken en dat was het begin van een lichte arduinoverslaving!

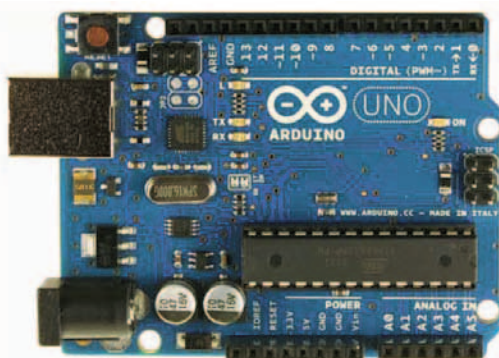


Fig. 1. De Arduino UNO / L'Arduino UNO

Wat is nu zo'n Arduino?

Je leest er alles over op <http://arduino.cc/>, maar hier volgt een korte samenvatting. Het is een klein ontwikkelbordje voor een 8-bits microprocessor (de Atmel ATmega 328). Je kan een aantal analoge en digitale inputs inlezen, en via de code in de processor kan je een aantal analoge of digitale outputs aansturen. Het meest populaire bordje is de Arduino UNO (zie **figuur 1**), kostprijs ongeveer € 25.

Eens je een project ontwikkeld hebt, is het natuurlijk zonde om je UNO-bordje definitief in een kast te verstoppen. Moet je dan een ander bordje kopen? Neen, want hiervoor bestaat gelukkig een (goedkope) oplossing. Je kan voor € 7,45 een inbouwkitje kopen dat de microprocessorchip en enkele belangrijke componenten bevat (zie <http://iprototype.nl/products/kits/barebone-arduino-kit>). Je zal immers meestal niet alle in- of uitgangen gebruiken. Dan maak je zelf een print met wat je nodig hebt, je programmeert de chip in de Arduino, en dan plaats je die chip in je eindproduct.

Hoe programmeer je de Arduino?

Hier schuilt de geniale uitvinding van de ontwerper(s): je hoeft helemaal geen assembler meer te leren en de ontwikkelomgeving is echt heel gebruiksvriendelijk gemaakt. Met een uitleg van een uurtje of zo kan je al aan de slag en een eerste werkend programma laten draaien. De ontwikkelomgeving of 'Integrated Development Environment (IDE)' werd geschreven in de taal 'Processing' en is multiplatform (dus niet enkel voor Windows). Je moet wel een beetje de basis van de taal C/C++ kennen of willen aanleren, en dan ben je vertrokken. Zie een voorbeeldje verderop.

Zowel de hardware als de software zijn open source. Dit wil zeggen dat je ook zelf je arduinobordje kan/mag ontwerpen. Alle gegevens zijn vrij te downloaden van de website, inclusief het schema en de eaglefiles om zelf je print te maken.

Een interessante variant van de Arduino is bijvoorbeeld de JeeNode (<http://jeelabs.com/products/jeenode>). Dit is een Arduino met een RF-module op 868 MHz (heeft echter geen eigen USB-aansluiting). Met een JeeNode kan je vanalles 'draadloos' doen, dus toch een soort radioamateurisme!

Il y a longtemps que j'avais entrepris l'apprentissage de la programmation d'un processeur PIC en assembleur. Ça n'allait pas de soi, j'avais autre chose à faire et finalement, le PIC a abouti rapidement au fond d'une armoire. Plus tard quand j'ai appris à connaître l'Arduino et que le radio club ostendais décida d'un achat groupé, je me suis laissé tenter et cela fut le début d'une "légère addiction"!

L'Arduino, c'est quoi?

Vous pouvez retrouver tous les détails sur <http://arduino.cc/>, mais voici un résumé succinct. C'est un petit circuit de développement

pour processeur 8 bits (l'Atmel ATmega 328). Via le code vous pouvez communiquer par toute une série d'entrées logiques et analogiques dans le processeur et de la même façon, il commande un nombre de sorties analogiques et digitales. La carte la plus populaire est l'Arduino UNO, (voir **fig. 1**), prix environ € 25.

Une fois un projet réalisé, serait-ce un péché de remiser définitivement le petit circuit UNO au fond de l'armoire? Dans ce cas, doit-on en acheter un nouveau? Non, car il existe heureusement une solution bon marché. Pour 7,45 Euro, on peut se procurer un kit d'intégration qui comprend le processeur et quelques composants importants (voir <http://iprototype.nl/products/kits/barebone-arduino-kit>). Dans la plupart des cas, vous n'utilisez pas toutes les entrées et sorties. Ensuite, vous dessinerez un circuit imprimé qui contiendra tout ce dont vous avez besoin, vous programmerez l'Arduino et placerez celui-ci sur le produit final.

Comment programmer l'Arduino?

C'est ici que réside la trouvaille géniale du développeur: il ne faut plus apprendre l'assembleur et l'environnement de développement est rendue particulièrement facile à utiliser. Connectez votre PC via le port USB et en moins d'une petite heure de prise en mains vous pourrez déjà vous mettre au travail et faire tourner votre premier programme fonctionnel. L'environnement de travail en 'IDE (Integrated Development Environment)' est écrit dans le langage 'processing' qui est multi plateforme, (ne convient pas seulement à Windows). Il faut bien maîtriser (un peu) la base du langage C/C++, ou l'apprendre, puis c'est parti! Voyez l'exemple plus bas.

Tant le hardware que le software sont 'open source'. Cela veut dire que vous avez le droit de créer votre propre carte Arduino. Toutes les données sont librement téléchargeables du site web, y compris les schémas et fichiers Eagle pour réaliser votre propre circuit.

Une variante intéressante est celle de JeeNode par exemple: <http://jeelabs.com/products/jeenode>. C'est un Arduino associé à un module RF à 868 MHz, (malheureusement dépourvu d'une connexion USB). Avec un JeeNode vous pouvez faire toute sortes de choses 'sans fil', c'est donc bien une espèce de radio amateurisme.

Wat ga je met de Arduino doen?

Aha! Je kan met de Arduino alles doen... Dat zegt men toch op het internet. De beste lokkertjes vind je op YouTube en heel dikwijls wordt verwezen naar het aansturen van een LED-kubus. Bekijk bijvoorbeeld deze eens: <http://www.youtube.com/watch?v=GUCX41pokZY>, of een van de vele varianten waarnaar je wordt doorverwezen.

Maar is er ook iets mee te doen voor radioamateurs? Tja, daar begint het natuurlijk. Iedere amateur heeft zo zijn eigen dromen, maar hoe begin je eraan? Er zijn natuurlijk enorm veel dingen reeds gemaakt en daar kan je naar hartenlust uit kopiëren en combineren. Zo bedacht ik de 'Morse Voltmeter' door een programma voor morsecode, dat ik kreeg via Ronny ON4RK, te combineren met de code voor de uitlezing van een potentiometer en aansturing van een LCD display, gevonden op het internet.

De hardware

Een echte voltmeter heb ik niet gemaakt, eerder een didactisch prototype. De ingelezen spanning is variabel tussen 0 en 5 volt door een potmeter aan te sluiten tussen massa en Vcc en de loper te verbinden met ingang A0 (een analoge ingang). De morse-uitgang is eenvoudig een 8 Ω luidsprekertje dat via een serieweerstand van 100 Ω verbonden wordt tussen pin 9 en massa. Dan nog een drukknopje aansluiten tussen pin 7 en massa, waarmee je de Arduino laat 'seinen'. Verder heb ik een (optionele) LCD display aangesloten, kwestie van de spanning ook zonder kennis van morse te kunnen aflezen. De aansluitingen van de LCD, en de uitleg over de gebruikte library heb ik gevonden op deze site: <http://arduino.cc/en/Tutorial/LiquidCrystal>

Zo kwam ik tot volgende verbindingen voor een standaard 14 pins LCD (zie tabel hier-naast).

Voor wie niet zo thuis is in elektronica, staat hier staat alles in nog veel meer detail: <http://learn.adafruit.com/character-lcds>

Het schema voor de LCD-aansluiting (**figuur 2**) in 4-bit mode is uiteraard voor heel wat andere projecten bruikbaar.

Zoals je op de foto van **figuur 3** kan zien, wordt de kit extern gevoed met een 12 V adapter, er is dus geen PC nodig om de schakeling te gebruiken.

Qu'est-ce qu'on peut faire avec un Arduino?

Eh bien! Avec un Arduino on peut tout faire... Du moins c'est ce qu'on prétend sur internet. Vous découvrirez toutes sortes de tentations sur Youtube et souvent vous retrouverez la commande d'un cube à LED. Voyez par exemple: <http://www.youtube.com/watch?v=GUCX41pokZY>, ou une des nombreuses variantes vers lesquelles vous serez renvoyés.

Mais est-ce que les amateurs peuvent aussi l'utiliser? C'est là qu'on commence à se poser les bonnes questions... Chacun a ses propres rêves, par quel bout les prendre? On a naturellement réalisé énormément de choses dans lesquelles vous pouvez puiser à l'envie et c'est en les combinant que l'on trouvera son bonheur. C'est de cette façon que j'ai imaginé un 'voltmètre morse' en recevant un programme pour code Morse reçu de Ronny ON4RK, que j'ai combiné avec le script de lecture d'un potentiomètre qui commande un afficheur LCD, trouvé sur internet.

Le hardware

Je n'ai pas conçu un vrai voltmètre, mais un prototype didactique. La tension lue est réglable de 0 et 5 volts en raccordant un potentiomètre entre le Vcc et la masse et le curseur à l'entrée A0, une entrée analogique. La sortie morse est un simple petit haut-parleur de 8 Ω raccordé via une résistance de 100 Ω entre la borne 9 et la masse. Ensuite on raccorde encore un bouton-poussoir entre la borne 7 et la masse pour faire bipper l'Arduino. Ensuite j'ai connecté un afficheur LCD (optionnel), de façon à pouvoir lire la tension sans être télégraphiste. Les connexions du LCD, les bibliothèques utilisées et les explications, je les ai trouvées sur ce site: <http://arduino.cc/en/Tutorial/LiquidCrystal>

De cette façon je suis arrivé aux connexions suivantes pour un LCD standard à 14 bornes (voir tableau à côté).

Pour celui qui n'est pas familiarisé avec l'électronique, vous trouverez ici les explications avec encore plus de détails: <http://learn.adafruit.com/character-lcds>

Le schéma pour les connexions des LCD (**Figure 2**) en mode 4 bits est également utilisable pour bien d'autres projets.

Comme vous voyez sur la photo de la **figure 3**, le kit est alimenté avec un adaptateur 12V, vous pouvez donc vous passer du PC pour utiliser le circuit.

Arduino	LCD	pin nr N° pin
GND	GND	1
+5 V	Vcc	2
2	Enable	6
3	Data Bit 0 (DB0)	7
4	(DB1)	8
5	(DB2)	9
6	(DB3)	10
11	Read/Write (RW)	5
12	Register Select (RS)	4

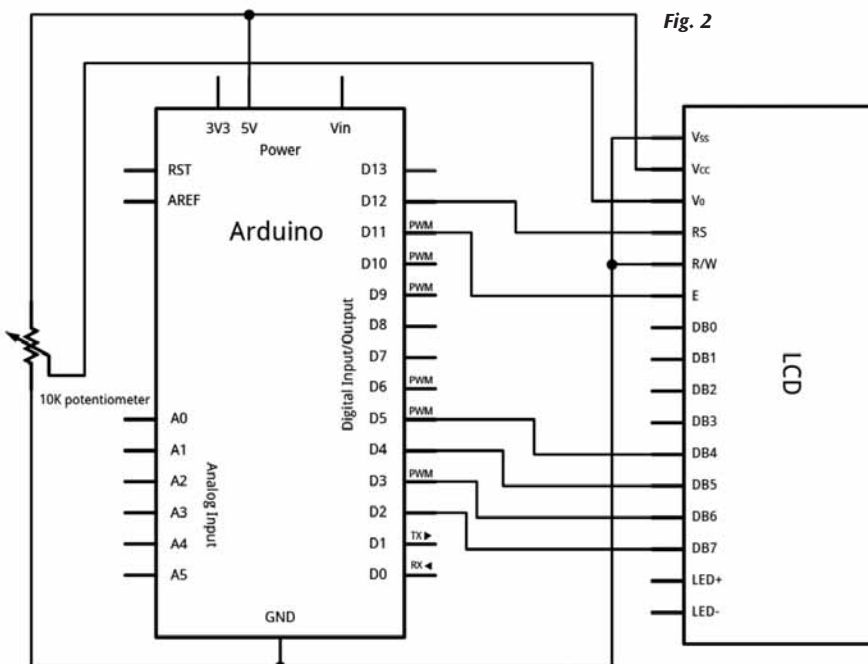


Fig. 2

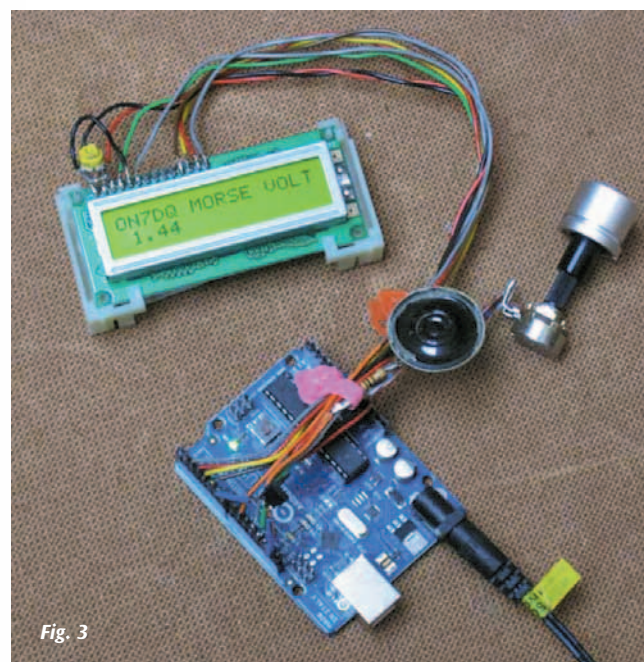


Fig. 3

De software

Een programma voor de Arduino wordt een 'sketch' genoemd. Hieronder de volledige listing voor de morse voltmeter, in blokjes gesplitst voor de uitleg. Overal waar in de code // staat, is dit commentaar die niet wordt gecompileerd.

Je kan de sketch hier downloaden:

http://users.khbo.be/on4hti/Morse_Voltmeter.pde

Een sketch bestaat in zijn eenvoudigste vorm uit drie gedeelten: voorbereidende declaraties, een setup() routine, en de loop(). loop() is het gedeelte dat continu blijft draaien. Wordt die loop() te onoverzichtelijk, dan is het natuurlijk beter om bepaalde stukken code in aparte functies (methodes) onder te brengen, zoals hieronder ook gebeurd is.

Deel 1: declaraties

```
#include <LiquidCrystal.h> // importeer de LCD library
// initialiseer de library met de juiste pinnummers voor de LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// geef namen aan de gebruikte pinnen;
// dit maakt de code leesbaarder
int ledPin = 13; // de interne LED van het arduino bordje
int spkPin = 9; // speaker 8Ω + serieweerstand 100Ω van pin 9
// naar massa
int knobPin = 0; // analogoog in A0
int switchPin = 7; // digital input pin voor drukknop

// variabelen voor de Morse routines:
#define VOLUME 100
// een array van "strings" maken,
// dit lijkt wat ingewikkeld maar het werkt!
char* letters[] = {
  ".-", "-.-.", "-.-.", "-.-.", ".-", ".-", ".-", ".-", ".-", ".-", // A-I
  "-.-.", "-.-.", "-.-.", "-.-.", "-.-.", "-.-.", "-.-.", "-.-.", // J-R
  "...", "...", "...", "...", "...", "...", "...", "... // S-Z
};
// nog een array van strings voor de cijfers
char* numbers[] = {"-----", "-----", ".----", "....-", "....-", "....-",
  "....-", "....-", "....-", "....-"};
int dotDelay = 100; // regelt de seinsnelheid

// variabelen voor de spanningsmeting:
int knobValue; // de inleeswaarde van de knob
// (geheel getal van 0-1023)
int volts; // de omgerekende spanning als geheel getal
// 0 – 50 = 0 - 5 Volt
double voltsDouble; // de spanning met 2 cijfers na de
// komma 0,00 – 5,00 Volt
```

Deel 2: de setuproutine (moet in iedere sketch staan, desnoods leeg)

```
void setup()
{
  // zet het aantal rijen en kolommen voor de LCD:
  lcd.begin(16, 2);

  // Zet een berichtje op de LCD
  // (helaas max. 16 tekens op een rij in dit geval)
  lcd.print("ON7DQ MORSE VOLTMETER");
  // bij uitvoering staat die "METER" niét op het LCD, zie foto

  // de werking van de verschillende pins instellen
  pinMode(ledPin, OUTPUT);
  pinMode(spkPin, OUTPUT);
  pinMode(switchPin, INPUT);

  // om zonder weerstanden te kunnen werken,
  // schrijf een HIGH naar een input!
```

Le software

Un programme pour Arduino s'appelle un 'sketch'. Plus bas vous retrouverez le listing complet du voltmètre Morse, découpé en blocs pour l'explication. Partout où vous retrouverez //, c'est du commentaire qui ne sera pas compilé.

Vous pouvez télécharger le sketch ici:

http://users.khbo.be/on4hti/Morse_Voltmeter.pde

Un sketch, dans sa forme la plus simple, est composé de trois éléments: les déclarations préalables, la routine de setup() et la boucle loop(). Loop(), c'est la partie qui tourne sans arrêt. Si cette loop() devient trop peu claire, il vaut mieux la scinder en morceaux séparés le code (méthodes), ce qui est les cas ci-dessous.

Partie 1: déclarations

```
#include <LiquidCrystal.h> // importez la librairie LCD
// initialiser la librairie avec les numéros de borne exacte pour le LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// donnez des noms aux bornes utilisées;
// cela rend le code plus lisible
int ledPin = 13; // la LED interne du circuit Arduino
int spkPin = 9; // haut-parleur 8Ω + résistance série
// 100Ω de borne 9 vers la masse
int knobPin = 0; // analogique en A0
int switchPin = 7; // entrée digitale pour bouton poussoir

// variables de la routine Morse:
#define VOLUME 100
// créer un array de "strings",
// cela semble compliqué mais ça fonctionne!
char* letters[] = {
  ".-", "-.-.", "-.-.", "-.-.", ".-", ".-", ".-", ".-", ".-", ".-", // A-I
  "-.-.", "-.-.", "-.-.", "-.-.", "-.-.", "-.-.", "-.-.", "-.-.", // J-R
  "...", "...", "...", "...", "...", "...", "...", "... // S-Z
};
// encore un array de chaînes pour les chiffres
char* numbers[] = {"-----", "-----", ".----", "....-", "....-", "....-",
  "....-", "....-", "....-", "....-"};
int dotDelay = 100; // règle la vitesse de transmission

// variables pour la mesure de tension:
int knobValue; // valeur de lecture du bouton
// (nombre entier de 0-1023)
int volts; // la valeur de la tension calculée
// 0 – 50 = 0 - 5 Volt
double voltsDouble; // la tension avec 2 chiffres après la
// virgule 0,00 – 5,00 Volt
```

Partie 2: la routine de setup (on doit la faire apparaître dans chaque sketch, à la rigueur vide)

```
void setup()
{
  // Initialisez le nombre de lignes et de colonnes du LCD:
  lcd.begin(16, 2);

  // Message d'accueil LCD
  // (hélas max. 16 caractères sur une ligne dans ce cas)
  lcd.print("ON7DQ MORSE VOLTMETER");
  //pendant l'exécution "METER" n'apparaît pas sur le LCD, voir photo

  // définir l'action des diverses bornes
  pinMode(ledPin, OUTPUT);
  pinMode(spkPin, OUTPUT);
  pinMode(switchPin, INPUT);

  // pour pouvoir fonctionner sans résistances,
  // écrivez un HIGH sur une entrée!
```

```
digitalWrite(switchPin,HIGH);
// dit hangt een interne pull-up weerstand aan de pin

// controle bij debuggen,
// de spanning wordt ook naar de PC gestuurd
Serial.begin(9600);
}
```

Deel 3: de oneindige lus ... of loop()

```
void loop()
{
// lees de spanning op pin A0 (analog in), waarde 0 - 1023
knobValue = analogRead(knobPin);
// herbereken naar een andere bovengrens, hier bvb. 5 Volt
volts = map(knobValue, 0, 1023, 0, 50);

// zend de waarde uit in morse,
// tenminste ALS op de drukknop gedruwd is
if (digitalRead (switchPin) == LOW){
flashSequence(numbers[volts/10]); // cijfer voor de punt
flashSequence(".-.-"); // punt
flashSequence(numbers[volts%10]); // cijfer na de punt
}
// bereken de waarde in double formaat
voltsDouble = volts/10.0;

// zet waarde ook op het LCD display
lcd.setCursor(1, 1);
lcd.print(voltsDouble);
// zend waarde ook naar seriele poort
Serial.println(voltsDouble);
// een beetje wachten ... en dan wordt opnieuw gemeten
delay(100);
}
```

Dan volgen de morseroutines: deze heb ik gekregen van ON4RK, maar zijn ook niet zo moeilijk te begrijpen.

```
// deze functie roep je op met als argument
// een "string" tekens om te seinen
void flashSequence(char* sequence){
int i = 0;
while (sequence[i]!= NULL) {
flashDotOrDash(sequence[i]);
i++;
}
delay(dotDelay * 3); // gap between letters
}

void flashDotOrDash(char dotOrDash) {
digitalWrite(ledPin, HIGH); // zet de ingebouwde LED aan
// = "visual morse"
analogWrite(spkPin, VOLUME); // zet de toon aan
if (dotOrDash == '.') { // als het een punt is
delay(dotDelay); // wacht 1 tel
}
else { // anders is het een streep
delay(dotDelay * 3); // wacht dan 3 tellen
}
digitalWrite(ledPin, LOW); // zet de LED weer uit
analogWrite(spkPin, 0); // zet de toon weer uit
delay(dotDelay); // wacht 1 tel tussen punten en strepen
}
```

Zo, dat is alles... Veel succes bij het experimenteren met Arduino en benieuwd wat jullie er eventueel allemaal mee doen.

```
digitalWrite(switchPin,HIGH);
// ceci connecte une résistance pull-up interne à l'entrée

// contrôle lors du debug,
// la tension est envoyée vers le PC également
Serial.begin(9600);
}
```

Partie 3: la boucle infinie ... ou loop()

```
void loop()
{
// lire tension sur pin A0 (analog in), valeur 0 - 1023
knobValue = analogRead(knobPin);
// recalculez vers valeur maximum, ici p.ex. 5 Volt
volts = map(knobValue, 0, 1023, 0, 50);

// envoyez la valeur en morse,
// Si le bouton poussoir est enfoncé
if (digitalRead (switchPin) == LOW){
flashSequence(numbers[volts/10]); // chiffre des unités
flashSequence(".-.-"); // point décimal
flashSequence(numbers[volts%10]); // chiffres après le point
}
// calculez la valeur en format double
voltsDouble = volts/10.0;

// placez également la valeur sur l'afficheur LCD
lcd.setCursor(1, 1);
lcd.print(voltsDouble);
// envoyez également la valeur vers le port sériel
Serial.println(voltsDouble);
// un instant d'attente ... et puis recommencez la mesure
delay(100);
}
```

Suivent les routines morse que j'ai reçues de ON4RK, mais qui également ne sont pas difficiles à comprendre.

```
// on appelle cette routine avec comme argument
// une chaîne "string" de signes émettre
void flashSequence(char* sequence){
int i = 0;
while (sequence[i]!= NULL) {
flashDotOrDash(sequence[i]);
i++;
}
delay(dotDelay * 3); // temps mort entre les lettres
}

void flashDotOrDash(char dotOrDash) {
digitalWrite(ledPin, HIGH); // allumez la LED incluse
// = "visual morse"
analogWrite(spkPin, VOLUME); // déclenchez la tonalité
if (dotOrDash == '.') { // si c'est un point
delay(dotDelay); // attendez un temps
}
else { // sinon c'est une barre
delay(dotDelay * 3); // on attend 3 temps
}
digitalWrite(ledPin, LOW); // éteindre la LED
analogWrite(spkPin, 0); // Couper la tonalité
delay(dotDelay); // attendez un temps entre points et barres
}
```

Voilà, tout est dit! Beaucoup de succès avec l'expérimentation avec Arduino et je reste à l'écoute de tout ce que vous en avez fait!