# Arduino Morsepieper.

| | | | |
|---|---|---|---|
| Naam activiteit: | **Arduino: Morse Pieper** | | |
| Type activiteit: |   Uitdagende Scouting Technieken; Internationaal | | |
| Speltakken: | Scouts, Explorers, Rover-Scouts, individueel of met kleine groepen | | |
| Tijd: | 45 – 90 minuten | Voorbereiding: | 30 minuten |
| | | Uitleg: | 5 minuten |
| | | Activiteit: | ➢ 30 minuten. |
| Benodigdheden: | 1. Arduino Nano<br>2. Een piezo pieper<br>3. PC met Windows 10.<br>4. De Arduino IDE software ( omgeving waarin je de Arduino kan programmeren)<br>5. de code voor het programma voor de Arduino nano. | | |

**Activiteit uitleg:**

Programmeer een Arduino nano zodat deze morse signalen gaat piepen.

Stap 1: Download en installeer de Arduino IDE ( programmeer software voor de Arduino)

De software om de arduino te programmeren is gratis.
Je kan de software downloaden van uit deze link:

https://www.arduino.cc/download_handler.php?f=/arduino-1.8.10-windows.exe

**Let op: dit is de link voor windows**!.

Er is ook software voor andere platformen, kijk daarvoor via deze link:
https://www.arduino.cc/en/Main/Software

Installeer de Arduino IDE software.

Stap 2: Kijk of de arduino IDE opstart.

Start de Arduino IDE.

Je zou dan zo iets moeten zien:


Dit is de Arduino IDE ( geïntegreerde ontwikkel omgeving) waarin je de programma's voor de arduino kan schrijven.

Stap 3: Het morse programma.

Ga naar de Arduino IDE.

Gooi eerst even een paar overbodige regels weg, deze:

**void setup() {**
**// put your setup code here, to run once:**

**}**

**void loop() {**
**// put your main code here, to run repeatedly:**

**}**
Je kan ze gewoon selecteren en verwijderen.

We hebben de code al aangepast voor de Arduino Nano, alleen je moet zelf even de tekst aanpassen die je de arduino nano wilt laten piepen. Deze is Geel gemarkeerd. Editen kan je in de Arduino IDE doen.

*Let op: lukt het knippen en plakken niet, dan kan je de code ook downloaden van deze site:*

*https://www.instructables.com/id/Arduino-Morse-Code/*

*en daarna een paar items aanpassen voor de nano, het betreft het groen gemarkeerde gedeelte en – uiteraard- het gele stukje tekst.*

**Copy en paste de onderstaande code naar de Arduino IDE, tussen /* */**

```
/*
  Morse Code Project

  This code will loop through a string of characters and convert these to morse code.
```

```
  It will blink two LED lights and play audio on a speaker.
 */


//*****************************************************//
//   Type the String to Convert to Morse Code Here  //
//*****************************************************//
char stringToMorseCode[] = "Arduino Morse Code Project";

// Create variable to define the output pins
//adapted for Nano& piezo element plug n play by PE1OOM
int led12 = 13;      // blink an led on output 13
int led6 = 6;        // blink an led on output 6
int audio8 = 4;      // output audio on pin 4
int note = 1200;      // music note/pitch



/*
       Set the speed of your morse code
       Adjust the 'dotlen' length to speed up or slow down your morse code
             (all of the other lengths are based on the dotlen)
       Here are the ratios code elements:
         Dash length = Dot length x 3
         Pause between elements = Dot length
               (pause between dots and dashes within the character)
         Pause between characters = Dot length x 3
         Pause between words = Dot length x 7

*/
int dotLen = 100;     // length of the morse code 'dot'
int dashLen = dotLen * 3;    // length of the morse code 'dash'
int elemPause = dotLen;  // length of the pause between elements of a character
int Spaces = dotLen * 3;     // length of the spaces between characters
int wordPause = dotLen * 7;  // length of the pause between words
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output for LED lights.
  pinMode(led12, OUTPUT);
  pinMode(led6, OUTPUT);
}
// Create a loop of the letters/words you want to output in morse code (defined in string at top of code)
void loop()
{
  // Loop through the string and get each character one at a time until the end is reached
  for (int i = 0; i < sizeof(stringToMorseCode) - 1; i++)
  {
    // Get the character in the current position
       char tmpChar = stringToMorseCode[i];
       // Set the case to lower case
       tmpChar = toLowerCase(tmpChar);
       // Call the subroutine to get the morse code equivalent for this character
       GetChar(tmpChar);
  }

  // At the end of the string long pause before looping and starting again
  LightsOff(8000);
}
// DOT
```

```
void MorseDot()
{
  digitalWrite(led12, HIGH);    // turn the LED on
  digitalWrite(led6, HIGH);
  tone(audio8, note, dotLen);   // start playing a tone
  delay(dotLen);                // hold in this position
}
// DASH
void MorseDash()
{
  digitalWrite(led12, HIGH);    // turn the LED on
  digitalWrite(led6, HIGH);
  tone(audio8, note, dashLen);  // start playing a tone
  delay(dashLen);               // hold in this position
}
// Turn Off
void LightsOff(int delayTime)
{
  digitalWrite(led12, LOW);     // turn the LED off
  digitalWrite(led6, LOW);
  noTone(audio8);                     // stop playing a tone
  delay(delayTime);             // hold in this position
}
// *** Characters to Morse Code Conversion *** //
void GetChar(char tmpChar)
{
        // Take the passed character and use a switch case to find the morse code for that character
        switch (tmpChar) {
          case 'a':
                MorseDot();
                LightsOff(elemPause);
                MorseDash();
                LightsOff(elemPause);
                break;
          case 'b':
                MorseDash();
                LightsOff(elemPause);
                MorseDot();
                LightsOff(elemPause);
                MorseDot();
                LightsOff(elemPause);
                MorseDot();
                LightsOff(elemPause);
                MorseDot();
                LightsOff(elemPause);
                break;
          case 'c':
            MorseDash();
                LightsOff(elemPause);
                MorseDot();
                LightsOff(elemPause);
                MorseDash();
                LightsOff(elemPause);
                MorseDot();
                LightsOff(elemPause);
                break;
          case 'd':
                MorseDash();
                LightsOff(elemPause);
                MorseDash();
                LightsOff(elemPause);
                MorseDot();
```

```
            LightsOff(elemPause);
            break;
        case 'e':
            MorseDot();
            LightsOff(elemPause);
            break;
        case 'f':
          MorseDot();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            break;
        case 'g':
            MorseDash();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            break;
        case 'h':
          MorseDot();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            break;
        case 'i':
          MorseDot();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            break;
        case 'j':
          MorseDot();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            break;
    case 'k':
          MorseDash();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            break;
        case 'l':
          MorseDot();
```

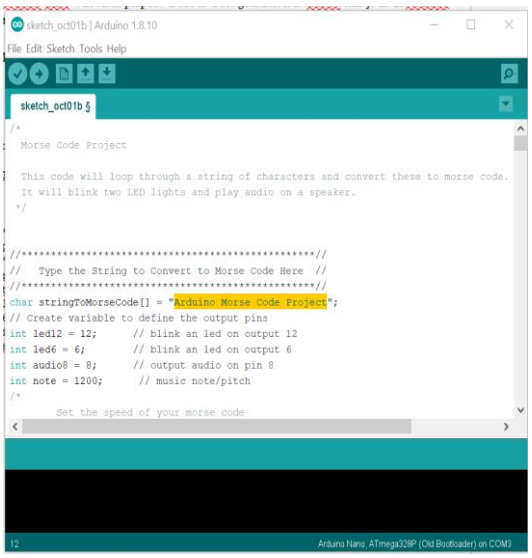
```
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            break;
    case 'm':
        MorseDash();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            break;
        case 'n':
        MorseDash();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            break;
        case 'o':
        MorseDash();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            break;
        case 'p':
        MorseDot();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            break;
        case 'q':
        MorseDash();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            break;
        case 'r':
        MorseDot();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            break;
        case 's':
        MorseDot();
            LightsOff(elemPause);
            MorseDot();
```

```
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            break;
        case 't':
          MorseDash();
            LightsOff(elemPause);
            break;
        case 'u':
          MorseDot();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            break;
        case 'v':
          MorseDot();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            break;
        case 'w':
          MorseDot();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            break;
        case 'x':
          MorseDash();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            break;
        case 'y':
          MorseDash();
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            break;
        case 'z':
          MorseDash();
            LightsOff(elemPause);
            MorseDash();
            LightsOff(elemPause);
            MorseDot();
```

```
            LightsOff(elemPause);
            MorseDot();
            LightsOff(elemPause);
            break;
        default:
            // If a matching character was not found it will default to a blank space
            LightsOff(Spaces);
    }
}
/*
    Unlicensed Software:

    This is free and unencumbered software released into the public domain.
    Anyone is free to copy, modify, publish, use, compile, sell, or
    distribute this software, either in source code form or as a compiled
    binary, for any purpose, commercial or non-commercial, and by any
    means.
    In jurisdictions that recognize copyright laws, the author or authors
    of this software dedicate any and all copyright interest in the
    software to the public domain. We make this dedication for the benefit
    of the public at large and to the detriment of our heirs and
    successors. We intend this dedication to be an overt act of
    relinquishment in perpetuity of all present and future rights to this
    software under copyright law.
    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
    EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
    MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
    IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR
    OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
    ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
    OTHER DEALINGS IN THE SOFTWARE.
    For more information, please refer to
*/
```

Na het knippen en plakken ziet er zo ongeveer uit: ( let op : zonder gele markering)

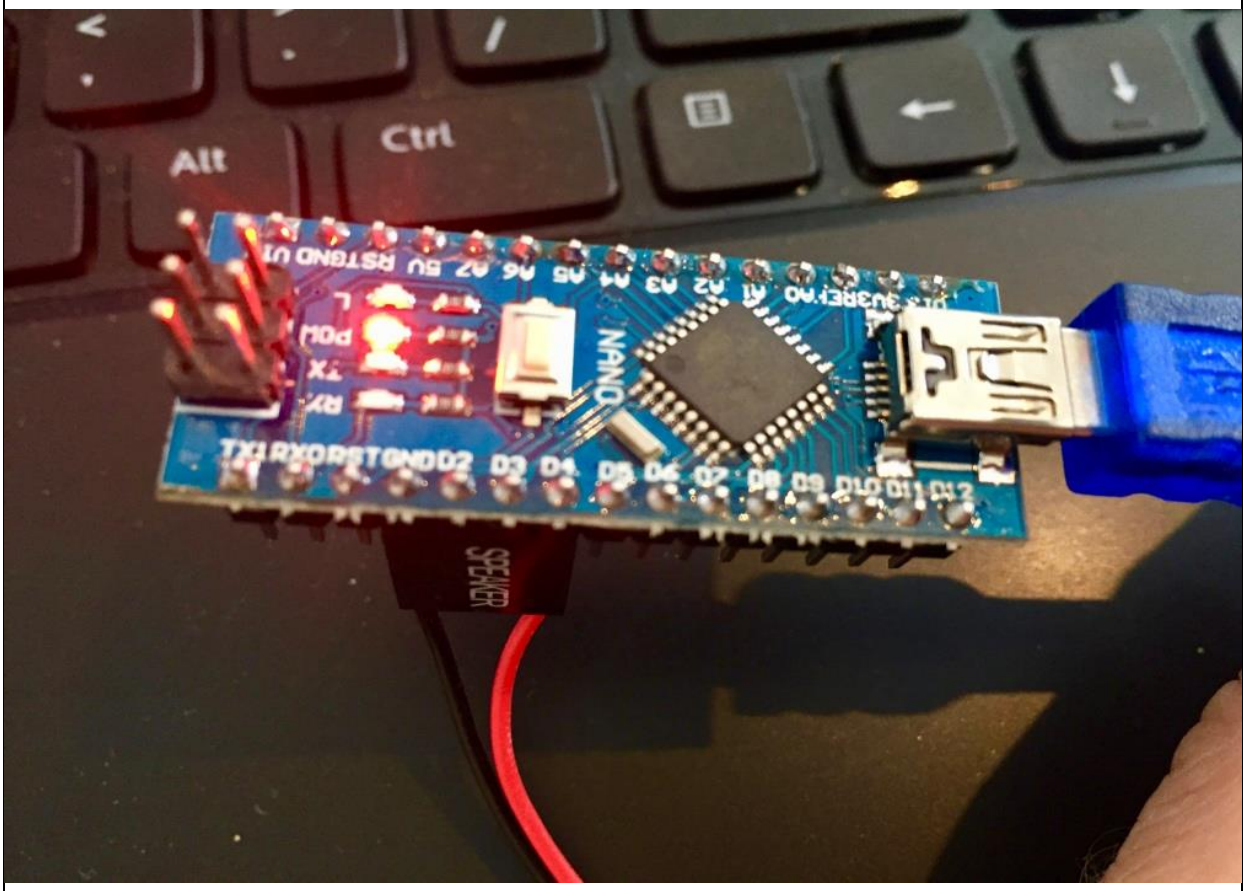



TIP: Na het aanpassen kan je het beste eerst even je programma opslaan via de File Save.

Stap 5: Programmeren van de Arduino Nano.

Voor dat we de Arduino nano programmeren moeten we eerst even de pieper op de nano zetten. Dit is het zelfde als bij de openingschallenge.

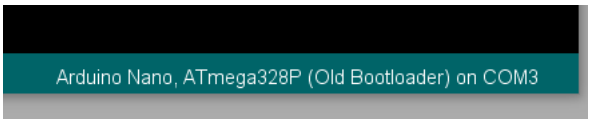
**<u>Dus de speaker is verbonden met Ground ( zwarte draad) en Pin D4 (rode draad).</u>**
Tevens gaat het onboard ledje L knipperen met de morse code. Hier hoef je niets voor te doen.




Sluit de USB kabel aan op de Arduino Nano. Als het programma van de opening er nog in zit dan gaat hij meteen weer piepen.

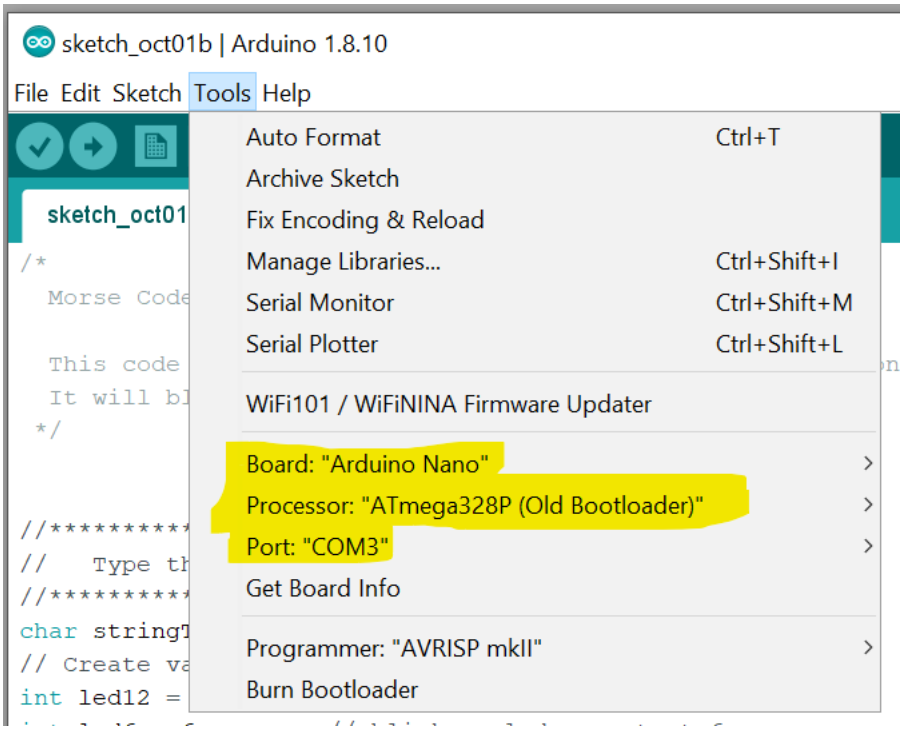
We gaan nu weer naar de Arduino IDE.

Voordat we de code uploaden gaan we eerst kijken of er verbinding is met de Arduino. Staat er onderin de Arduino IDE zoiets als dit:

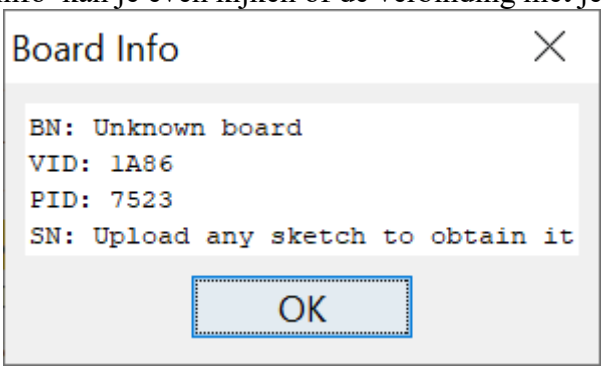



Dan is de Arduino herkend en kan geprogrammeerd worden.

Staat er nog niets dan moeten we even verder kijken. Ga naar Tools, en controleer bij Board of de arduino Nano is ingesteld, zo niet kies dan uit de lijst de arduino nano. let op: **De arduino nano gebruikt een oude bootloader dus gebruik ook die instelling, indien nodig pas de instelling aan**. ( de upload faalt met de nieuwe bootloader ) Check ook op welke compoort de arduino staat bij Port. Het ziet er ongeveer zo uit:




Met de optie get board info kan je even kijken of de verbinding met je nano er is:




Als dit allemaal goed staat dan kan je de “sketch” zoals het programma wordt genoemd laden op de arduino.

Dat gaat met deze knop: (rood omcirkeld):


Tijdens het uploaden knipperen er een paar ledjes en daarna hoor en zie je de Arduino je morse tekst afspelen.

Je morse pieper is klaar voor gebruik.

Stap 6: Aanpassen van de code.

Als je goed naar het programma kijkt, kan je ook de snelheid aanpassen en bijvoorbeeld ook de lengte van de morse tekens aanpassen.

Hoe snel kan de arduino morse piepen?
Kan de radiozendamateur dat bijhouden?

Probeer het maar uit!

**AVG, Privacy, datagebruik en veiligheid:**

**AVG en Privacy: Geen Impact**

**Datagebruik: Download vanaf het internet van de Arduino Software.**

**Veiligheid: geen bijzonderheden.**

De LJJO bedankt CHRIS WEATHERFORD voor deze code.
Het originele project van Chris vindt je hier: https://www.instructables.com/id/Arduino-Morse-Code/

Hier vindt je ook meer uitleg over de code en de pieper die Chris heeft gebouwd.
Chris gebruikt een oudere Arduino ( UNO) maar dat maakt niet zo veel uit.