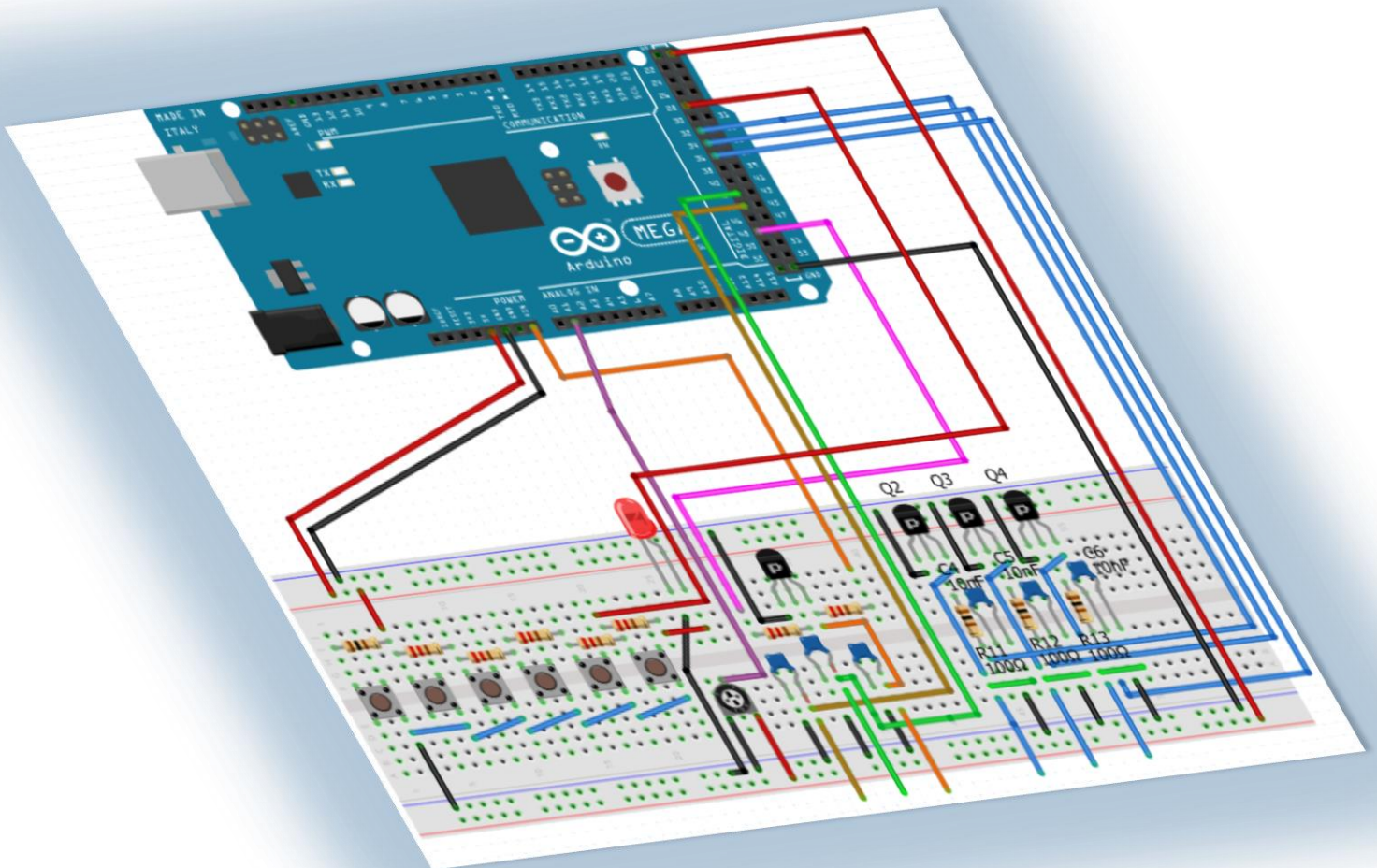


# Handleiding K3NG keyer voor de Arduino MEGA 2560

V 3.1





INHOUDSOPGAVE	pag.
<b>1. Inleiding</b> .....	<b>2</b>
1.1 Kenmerken van de keyer.....	3
1.2 Arduino MEGA 2560 specificaties.....	5
1.3 Arduino.....	7
1.4 De meest fundamentele onder delen .....	7
<b>2. Installeren van de Arduino IDE 1.8.5</b> .....	<b>9</b>
<b>3. Het werken met het Arduino IDE programma</b> .....	<b>10</b>
3.1 Arduino IDE configuratie.....	10
<b>4. Blink sketch</b> .....	<b>12</b>
4.1 Opstelling.....	12
4.2 Het programma maken en opstarten .....	13
<b>5. K3NG CW Keyer, UNO</b> .....	<b>16</b>
<b>6. Indelen van de K3NG CW keyer bestanden</b> .....	<b>17</b>
6.1. De bestanden op de juiste plek zetten... ..	18
<b>7. De K3NG Arduino MEGA 2560 opbouw</b> .....	<b>20</b>
7.1 <i>Beginnen met de opbouw van de K3NG keyer</i> .....	20
<b>8. Het stappenplan</b> .....	<b>21</b>
8.1 <i>Toevoegen van de knoppen</i> .....	21
8.2 <i>Toevoegen van de weerstanden</i> .....	22
8.3 <i>Toevoegen van een LED</i> .....	23
8.4 <i>Data van de K3NG keyer laden in Arduino IDE</i> .....	24
8.5 <i>K3NG basis programma aanpassen</i> .....	26
8.6 <i>Met potentiometer uitbreiden</i> .....	27
8.7 <i>Seinsleutel en luidspreker uitbreiding</i> .....	29
8.8 <i>Toetsenbord aansluiten</i> .....	30
8.9 <i>Display aansluiten</i> .....	32
8.10 <i>De verbinding met je transceiver</i> .....	34
8.11 <i>Geortz DSP CW Decoder</i> .....	36
<b>9. Hoe werkt de Decoder</b> .....	<b>40</b>
<b>10. Toetsenbord opdrachten</b> .....	<b>41</b>
<b>11. Componenten lijstje</b> .....	<b>42</b>
<b>12. Bijlagen, keyer_pin_settings en keyer_features_and_options.h</b> .....	<b>43</b>

## Arduino CW-Keyer

### 1. Inleiding.

Een Arduino is een kleine relatief goedkope programmeerbare computer die ontzettend veel kan doen door de juiste code te schrijven. Een Arduino project is aan te raden voor iedereen die geïnteresseerd is in programmeren of in het lekker knutselen met elektronica. Je kunt kiezen uit een breed scala van projecten maar ik, als zendamateer, heb mijn keuze laten vallen op de K3NG Arduino CW-Keyer. Met veel plezier heb ik dit projectje tot een goed einde gebracht. Mijn dank aan Ralph, PA1RB en Anthony Good, K3NG, zonder hun hulp was dat zeker niet gelukt!

Mogelijk kan deze handleiding het werken met Arduino projecten stimuleren, het is bedoeld voor de beginner met enige basis kennis van elektronica. Je doet er gaande weg kennis op met het omgaan van microcontrollers en de daarbij behorende programmering. Juist voor de zenden radioamateur is dit een bijzonder leuk project.

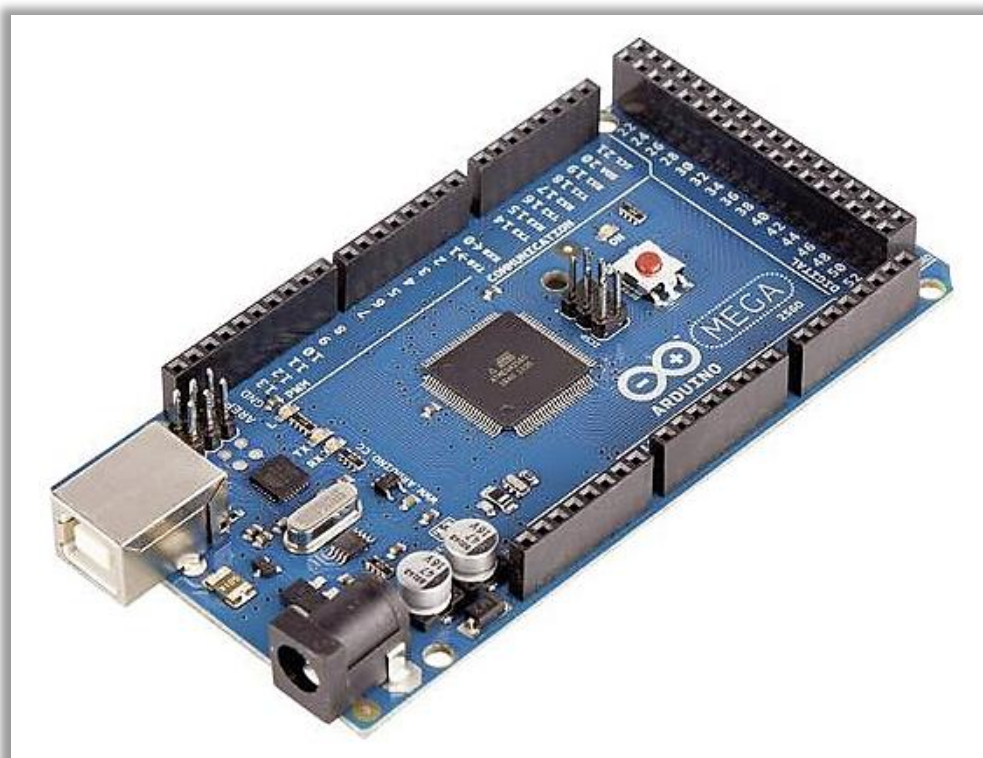
K3NG is de ontwerper van de Arduino CW-Keyer, gebaseerd op de Arduino **UNO** microcontroller. Voor de meer gevorderden is op zijn site een uitgebreide beschrijving aanwezig voor de bouw ervan.

Zijn hele programmering is van het type "open source" wat de mogelijkheid biedt om, naar eigen keuze, in het programma veranderingen aan te brengen.

<https://blog.radioartisan.com/arduino-cw-keyer/>

Omdat ik in het bezit ben van de **Arduino MEGA 2560**, fig.1, ben ik op zoek gegaan naar een handleiding gebaseerd op laatst genoemde. Na wat moeite heb ik uiteindelijk een site van KF4BZT gevonden waarmee ik als beginneling iets mee dacht te kunnen.

Het verschil tussen de UNO en de MEGA zit hem in een andere pinbezetting en KF4BZT past deze aan vanuit de K3NG basis beschrijving.



Figuur 1, de Arduino MEGA 2560

In de handleiding van KF4BZT wordt de CW-Keyer stapsgewijs opgebouwd, gebruik makend van de Arduino **MEGA 2560**, gelardeerd met vele (on)duidelijke plaatjes.

<https://kf4bzt.wordpress.com/2015/08/06/arduino-cw-keyer-project/>

Waarom dan deze handleiding vraag je je af. Wel, ik als dummy, wist nog maar weinig af van het Arduino fenomeen en liep al gauw vast in zijn beschrijving. Enerzijds een uitvoerige uitleg, anderzijds mis ik het begin stadium van "hoe begin ik nou". Bovendien is hij op zijn site nogal slordig met sommige schema's, waarin fouten zitten. Allereerst wat biedt deze keyer:

### 1.1 Kenmerken van de K3NG CW-keyer.

CW-snelheid instelbaar van 1 tot 999 WPM.

Maximaal zes selecteerbare zender sleutellijnen.

Programmering en interfacing via USB-poort ("command line interface")

USB of PS2 toetsenbord interface voor CW toetsenbord operatie zonder een computer.

Logging en Contest Program Interfacing via K1EL Winkey 1.0 en 2.0 interface protocol emulatie.

Optionele PTT-uitvoeringen met instelbare lead-, tail- en hangtijden.

Optioneel LCD Display - Klassieke 4-bits modus, Adafruit I2C RGB display of YourDuino I2C LCD Display.

Tot 12 geheugens met macro's.

Serienummers.

CW toetsenbord (via een terminal server programma zoals Putty of het Arduino Serial programma).

Snelheidspotentiometer, optioneel - snelheid ook verstelbaar met commando's

QRSS en HSCW.

Beacon / Fox modus.

Iambic A en B.

Rechte sleutel modus.

Ultimatische modus.

Bug modus.

CMOS Super Keyer Iambic B Timing.

Paddle reverse.

Hellschreiber modus (toetsenbord verzenden, geheugen macro, baken)

Farnsworth Timing.

Verstelbare frequentie-zijtoon.

Sidetone uitschakelen / sidetone hoge / lage uitgang voor het uitschakelen van audio oscillator.

Command modus voor het gebruik van de paddle om instellingen te wijzigen, programma herinneringen, enz.

Keying Compensation.

Dah naar Dit ratio aanpassing.

Belasting.

Callsign ontvangst oefening.

Verzend oefening.

Geheugen stapelen.

"Dead Operator Watchdog".

AUTOSPACE.

Wordspace aanpassing.

Pre-configured en Custom Prosigns, symbolen.

Blijvende opslag van de meeste instellingen.

Modular code design waardoor selectie van functies en eenvoudige code wijziging mogelijk is.

Niet-Engelse Karakterondersteuning.

CW Ontvangst Decoder (EXPERIMENTELE).

Snelheidscoëfficiënt van de draaiknop.

Slaapstand.

USB-muis ondersteuning.

Mayhew LED Ring Support

Alfabet Verzendpraktijk.

QLF / Straight Key Emulation (NIEUW).

USB-toetsenbord HID (Human Interface Device) Interface (Keyer = toetsenbord voor uw computer) (NIEUW).

---

Hieronder de Digitale / PWM / Analoge Pinnen die beschikbaar zijn op een drietal veel toegepaste Arduino's:

**Arduino Nano Pins.**

Digitale I / O Pins 14

PWM Pins 6

Analoge I / O-pins 8

**Arduino Yun Pins.**

Digitale I / O Pins 20

PWM Pins 7

Analoge I / O-pins 12

**Arduino Mega 2560 Pins.**

Digitale I / O-pinnen 54

PWM Pins 15

Analoge I / O-pins 16

**Arduino UNO en Mega 2560.**

Het voordeel van de Arduino Mega 2560 is, dat het 256KB flashgeheugen heeft, (waarvan 8KB wordt gebruikt door de opstartlader). Hierbij krijg je de ruimte voor alle mogelijkheden die de firmware biedt.

Hieronder zie je de *Digitale / PWM / Analoge* pinnen die beschikbaar zijn op de Arduino MEGA 2560. Met de Mega kun je veel meer verbindingen maken, hoe meer verbindingen, hoe beter. Niet alle pinnen zullen hiervoor gebruikt worden, aangezien er manieren zijn om verbindingen met elkaar te koppelen via een enkele lijn, zoals aarding en de 5 volt-lijn om de stroomkring te voeden. Hieronder een vergelijking met de Arduino UNO.

**Arduino Nano Pins.**

Digitale I / O Pins 14

PWM Pins 6

Analoge I / O-pins 8

**Arduino Mega 2560 Pins.**

Digitale I / O-pinnen 54

PWM Pins 15

Analoge I / O-pins 16

## 1.2 ARDUINO MEGA 2560 specificaties.

Belangrijk zijn de digitale- en analoge pennen, en de kloksnelheid (CW Decoding).

### Technische specificaties:

Microcontroller ATmega2560

Bedrijfsvoltage 5V

Ingangsspanning (aanbevolen) 7-12V

Ingangsspanning (limiet) 6-20V

Digitale I / O-pinnen 54 (waarvan 15 PWM-uitgang geven)

Analoge invoerpinnen 16

Gelijkstroomstroom per I / O-pin 20 mA

Gelijkstroomstroom voor 3,3 V pin 50 mA

Flash geheugen 256 KB waarvan 8 KB gebruikt door bootloader

SRAM 8 KB

EEPROM 4 KB

Kloksnelheid 16 MHz

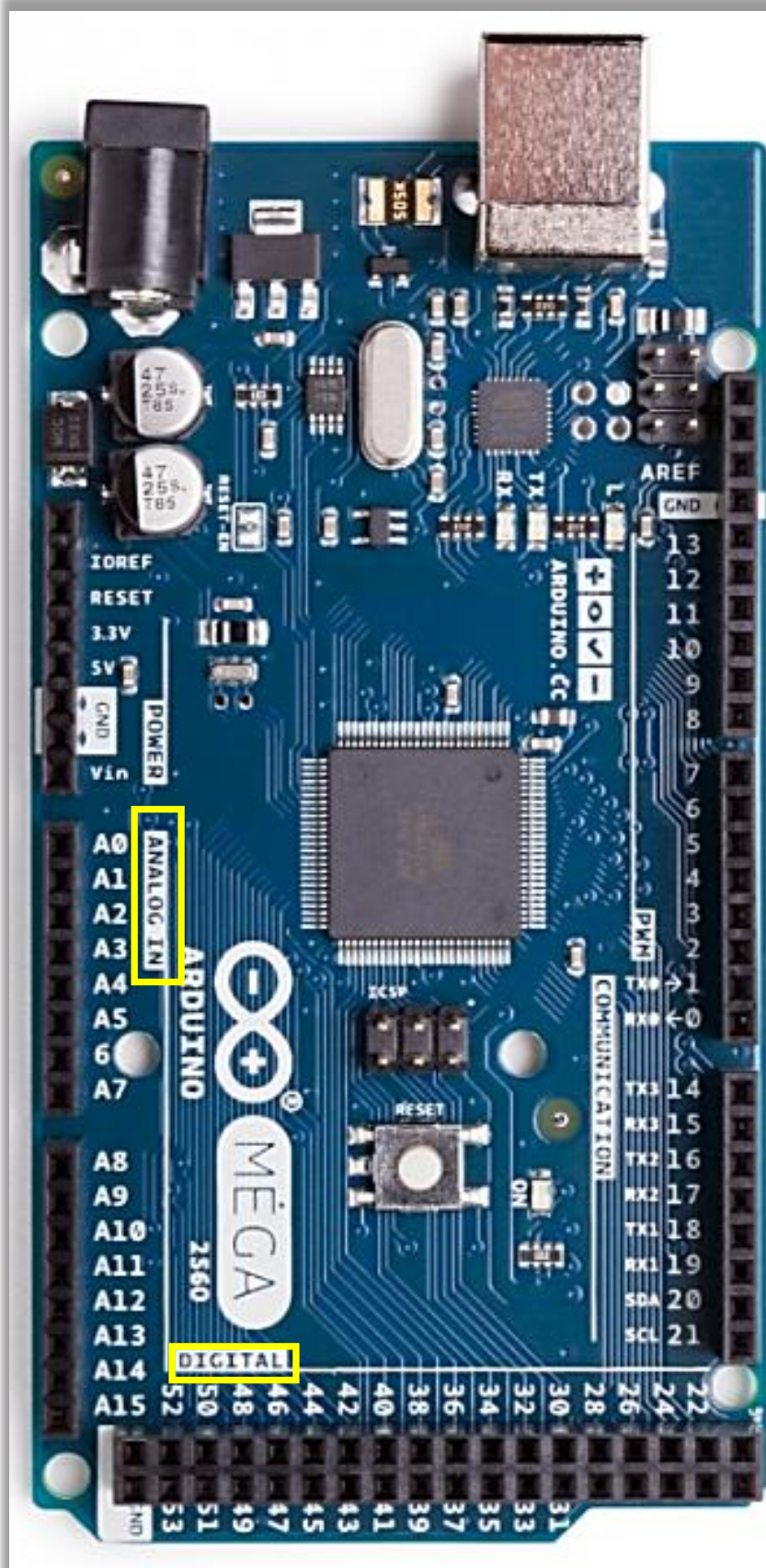
Lengte 101,52 mm

Breedte 53,3 mm

Gewicht 37 g

---

Zoals je hieronder, fig.2, kunt zien, is er een aparte digitale sectie die begint met pin 22 tot en met pin 53. De analoge pin uitvoering, linksonder, is gelabeld met A0 tot A15. De voeding is op de pen 5Volt met onder anderen twee massa (GND) pennen.



Figuur 2, De Arduino MEGA 2560

### 1.3 Arduino

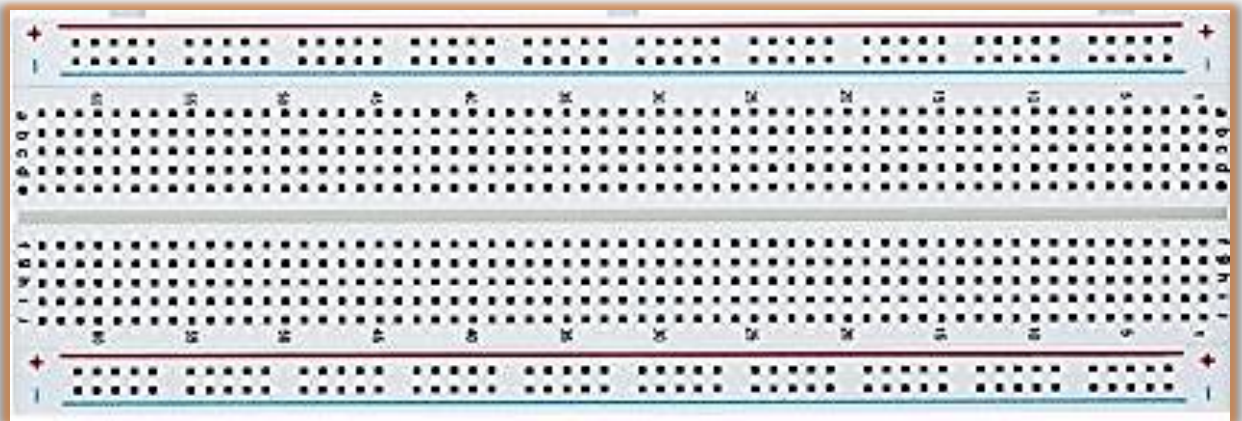
Op basis van een input kan een Arduino schakeling zelfstandige acties uitvoeren door het afgeven van digitale en analoge outputsignalen. Uit de vele soorten Arduino microcontrollers maak je in dit geval gebruik van een Arduino MEGA 2560 microcontroller om met een Morse sleutel of een toetsenbord een transceiver aan te sturen, ook kun je hiermee morsetekens als tekst zichtbaar maken op een display. De Arduino Mega 2560, met de Atmega 2560-chip, heeft een programmeerbaar geheugen van 256 KB en 70 programmeerbare aansluitingen.

### 1.4 De meest fundamentele onder delen.

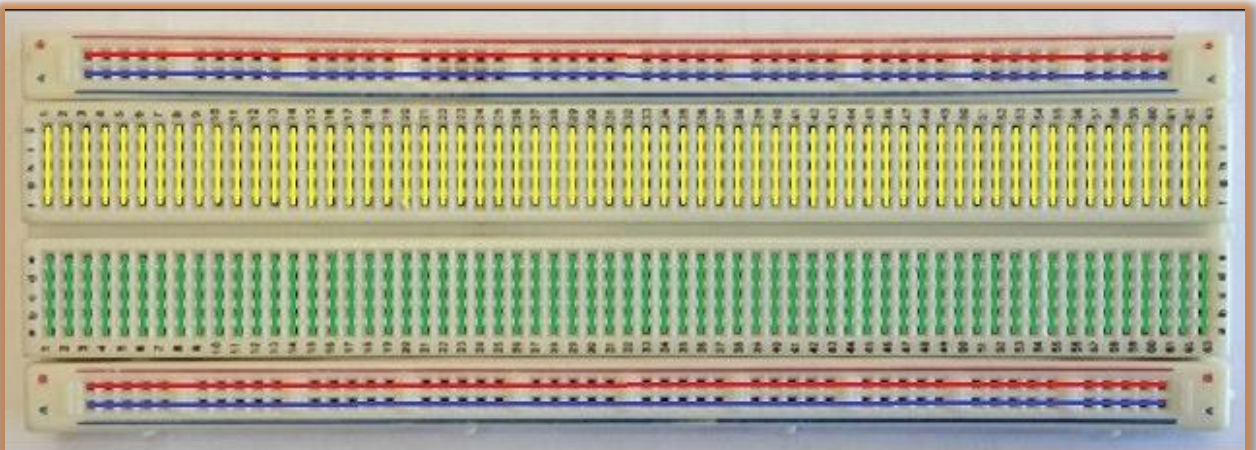
Wat heb je hiervoor in eerste instantie nodig:

- a. Een z.g. breadboard, fig.3a

Voor dit project zijn 2 breadboards nodig. In fig. 3b zie je hoe de verbindingen onder het breadboard zijn gelegd.



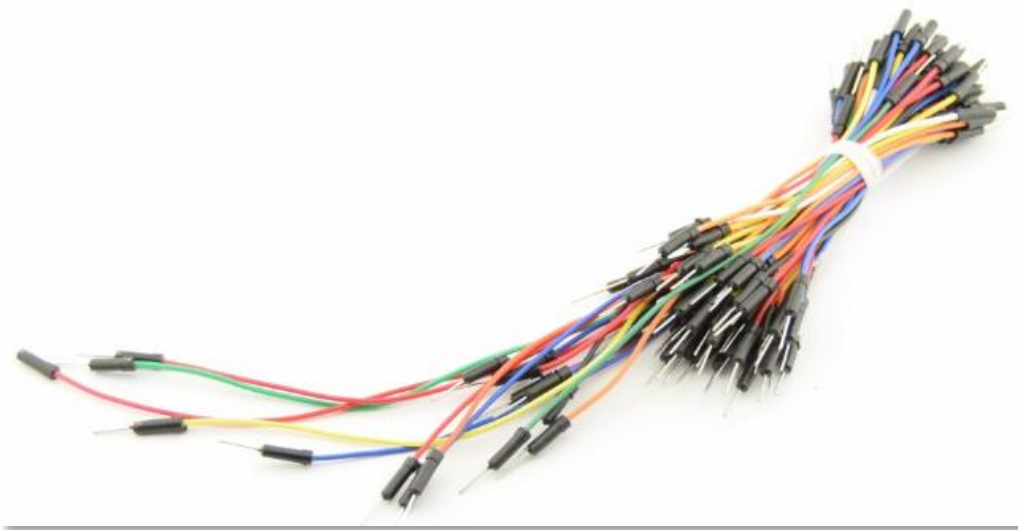
Figuur 3a, voorzijde breadboard



Figuur 3b, achterzijde breadboard



- b.** Verbindingsdraadjes, bestel er niet te weinig, bovendien zijn ze relatief goedkoop, fig.4.



*Figuur 4, aansluitdraden*

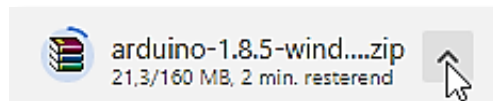
- c.** Daarnaast natuurlijk alle voorkomende componenten die in de schakelingen zijn opgenomen, zie hiervoor hoofdstuk 11, componenten lijstje.

## 2. Installeren van de Arduino IDE 1.8.5

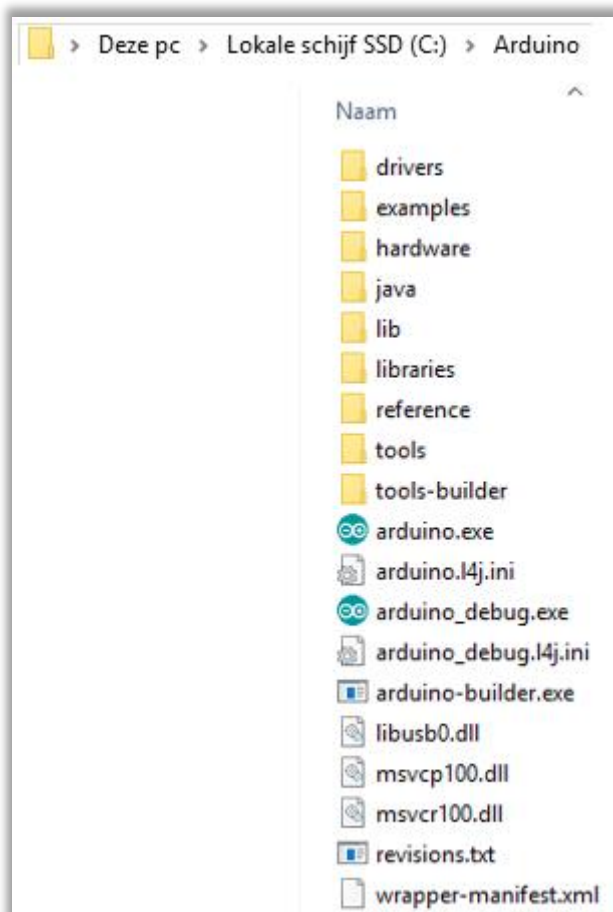
Arduino IDE is een programma waarmee je kunt communiceren tussen je PC en de microcontroller en de benodigde programmatuur naar de microprocessor uploaden. Dus dit programma ga je eerst downloaden.

Maak ergens op je Harde Schijf een bestand aan, fig.5, met bijvoorbeeld de naam: Tijdelijke downloadmap K3NG en download het volgende zip bestand:

<https://www.arduino.cc/en/Main/Software>



Maak vervolgens een nieuwe map aan met de naam: *C:/Arduino* en kopieer de inhoud van bestand *arduino-1.8.5* uit de tijdelijke download map naar je *Arduino* map, fig.6.



Figuur 6, de inhoud van arduino-1.8.5

Start het programma nu op met het bestand *Arduino.exe*.

Tot zover de installatie van Arduino IDE.



### 3. Het werken met het Arduino IDE programma.

Het wordt nu tijd de USB poort van je Arduino MEGA 2560 met de USB poort van je computer te verbinden.

Start Arduino IDE op en er verschijnt bijvoorbeeld een soort gelijk Arduino venster, fig.7.

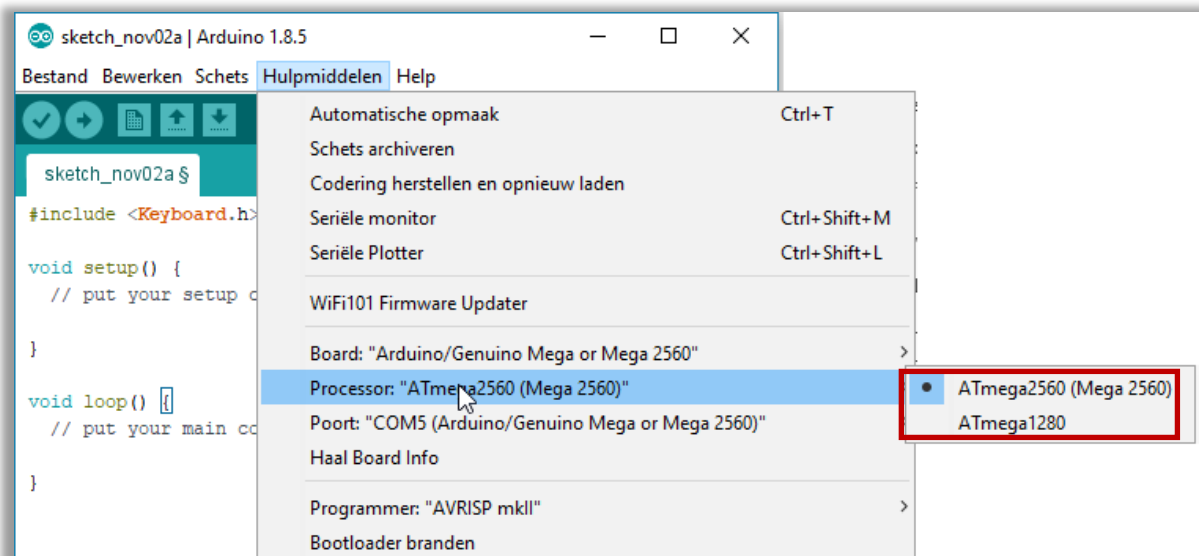
#### 3.1 Arduino IDE configuratie



Figuur 7, Arduino, een mogelijk start scherm.

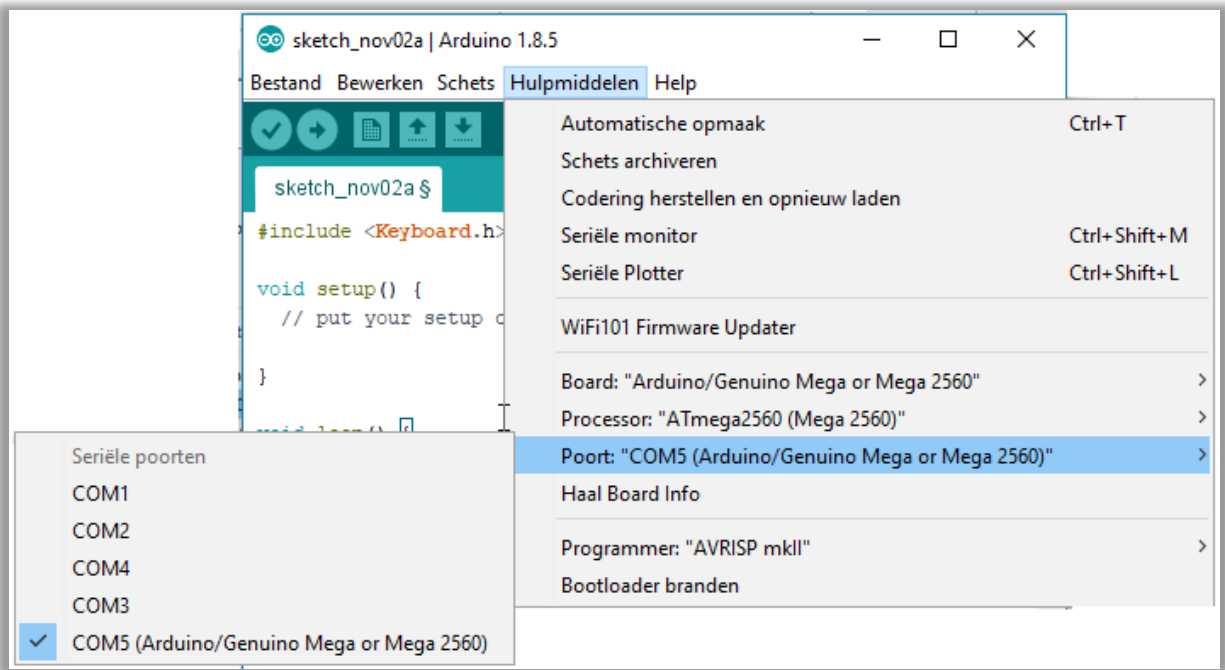
Kies eerst voor volgende opties, fig. 8 en 9 vanuit **Hulpmiddelen**.

Selecteer vanuit Hulpmiddelen de Arduino Mega, fig.8

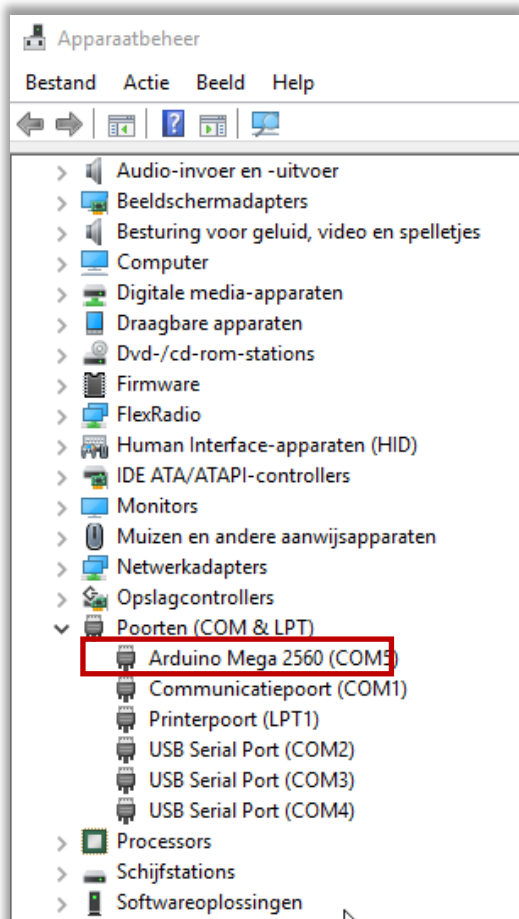


Figuur 8, ATmega2560

Een poort wordt meestal al door Windows 10 vastgesteld, fig.9. Mocht een en ander nog niet werken, kijk dan even onder *Apparaat beheer*, of Arduino op de goede USB poort is aangesloten, fig.10. Dit doe je via: *Configuratiescherm > Apparaatbeheer*, en geef dan alsnog het correcte poort nummer in. Bij jou kan natuurlijk een andere poort worden toegewezen als in onderstaand voorbeeld.



Figuur 9, toekennen van het poortnummer



Figuur 10, Apparaatbeheer

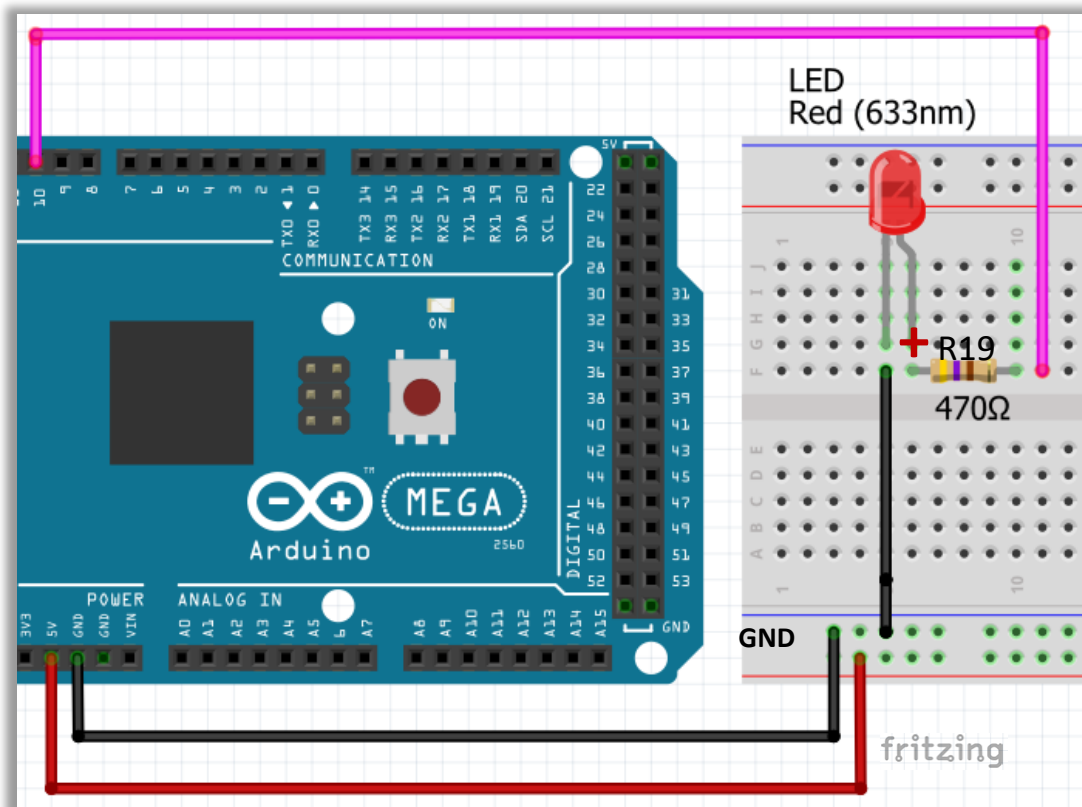
## 4. Blink sketch.

Voordat je met de K3NG sketch (programma) gaat beginnen is het raadzaam om eerst de sketch Blink te laden. Arduino heeft hiertoe een aantal nuttige programma's mee geven onder: [Voorbeelden](#). Met [Blink](#) kun je dan controleren of de Arduino MEGA ook goed functioneert en het kost je misschien maar 10 minuten de tijd. *Verbreek nu de verbinding met je PC.*

NB. Het is niet direct verplicht om de externe voeding (7-12 DC Volt) toe te passen, maar het biedt een voordeel bij grotere schakelingen en het toepassen van een display, zodat de werking stabiel wordt. De voeding via je USB aansluiting wordt dan automatisch uitgeschakeld, maar de inwendige voeding van de Arduino blijft 5Volt.

### 4.1 Opstelling.

Onderstaande fig. 11 laat de simpele opstelling zien van de Arduino Mega naar je breadboard. Kijk goed uit hoe je de onderdelen positioneert. Bij de meeste leds is de langere aansluitdraad de +.



Figuur 11, Blink opstelling

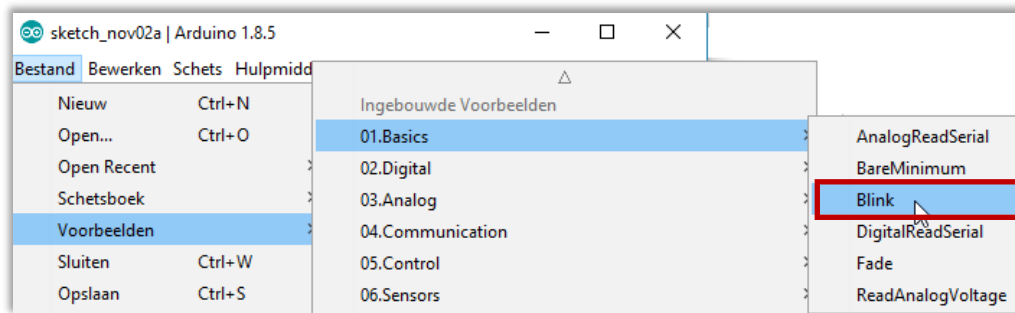


*Probeer, bij het aansluiten op je breadboard, geconcentreerd en niet te gehaast te werken. Het is heel makkelijk om een fout te maken. Als je niet voorzichtig te werk gaat dan ben je al gauw aan je volgende Arduino MEGA 2560 toe, zoals mij overkwam.*

## 4.2 Het programma maken en opstarten.

Voorop gesteld, deze handleiding is niet bedoeld om je uitleg te geven over de programmeertaal, maar met een beetje moeite is er op internet voldoende te vinden om globaal de werking van de programma's te begrijpen. Bovendien staat bij sommige opdrachten commentaar regels met een toelichting.

Start Arduino IDE op en laadt nu de sketch Blink, fig.12.



Figuur 12, Blink selecteren: Voorbeelden > 01.Basics > Blink

Een duidelijke toelichting op dit programma vind je op:

<https://www.arduino.cc/en/Tutorial-0007/BlinkingLED>

Het Blink programma verschijnt nu in het Arduino IDE venster. In onderstaande voorbeeld, fig. 13, is de pin variabele gesteld op 10, overeenkomstig de opstelling fig.10.

Alles wat tussen: `/* ..... */` wordt als tekst opgevat en niet gecompileerd.

Regels voorafgaande aan: `//` worden ook niet gecompileerd.

Voeg de regel toe: **`int LedPin = 10;`**

```
/*
Blink
Zet de led aan en herhaalt dit iedere seconde.

Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
the correct LED pin independent of which board is used.
If you want to know what pin the on-board LED is connected to on your Arduino
model, check the Technical Specs of your board at:
https://www.arduino.cc/en/Main/Products

modified 8 May 2014
by Scott Fitzgerald
modified 2 Sep 2016
by Arturo Guadalupi
modified 8 Sep 2016
by Colby Newman

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/Blink
*/
int led = 10;
// De setup stopt pas als je de reset knop op het Arduino bordje indrukt.

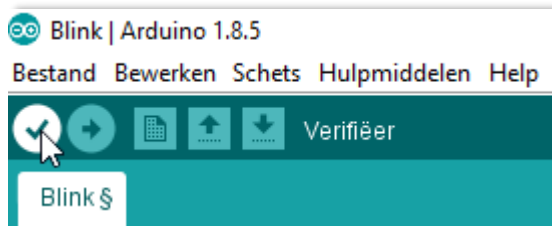
void setup() {
  // initialiseer digital pin als output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// met loop herhaal je het programma.
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // zet de LED aan (HIGH voltage aan)
  delay(1000); // wacht een seconde
  digitalWrite(LED_BUILTIN, LOW); // zet de LED uit met LOW
  delay(1000); // wacht een seconde
```

Figuur 13, het Blink programma

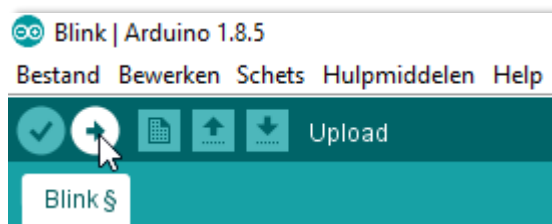
Staat alles goed op je breadboard, sluit dan de USB poort van je breadboard met de PC aan.

Klik op het "V" icoontje, fig.14, om het programma te compileren. Als je een fout melding krijgt, heb je een type foutje in het programma gemaakt.



Figuur 14, compileren.

Klik vervolgens op het "pijl" icoontje om het programma te uploaden naar je Arduino, fig.15.



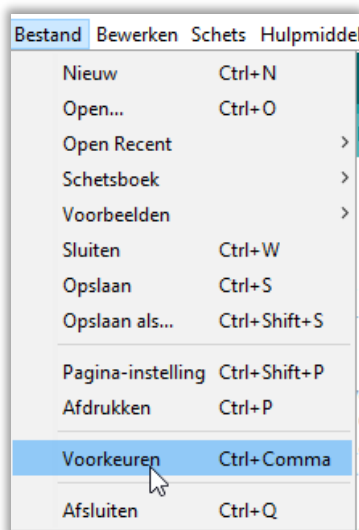
Figuur 15, uploaden

Als je bij het uploaden ook geen foutmelding krijgt, zal het ledje met de geprogrammeerde tijdsintervallen aan- en uit gaan. Zo, je hebt je eerste experiment succesvol afgerond. !!

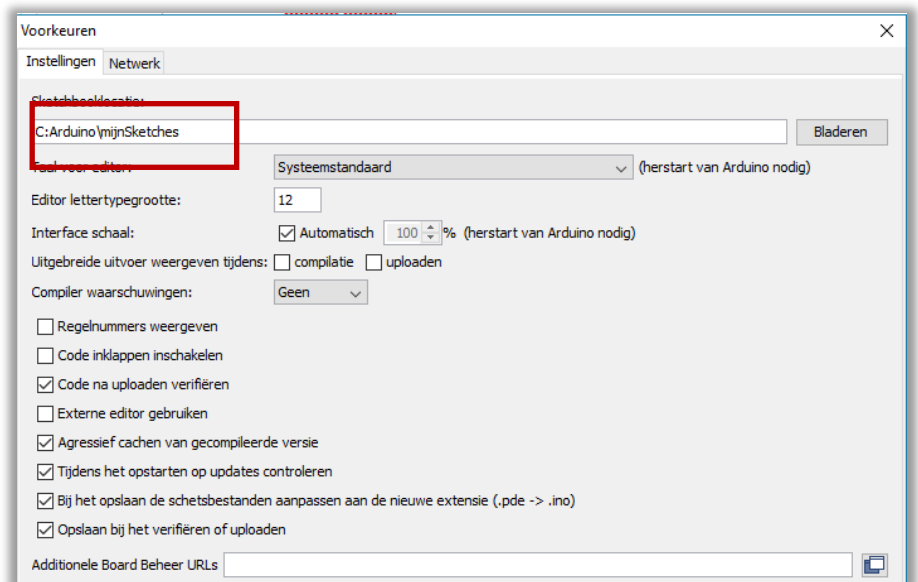
Hierna ga je met het "echte" project aan de gang. Om de kans op fouten te reduceren is de opbouw stapsgewijs beschreven.

Opslaan van je ontwerp programma's: ga naar **Bestand > Voorkeuren** fig.16 en 16a en sla in een zelfbenoemde map je sketches op.

Vergeet niet op **OK** te klikken.



Figuur 16, Map Opslaan bestanden



Figuur 16a, Voorkeuren

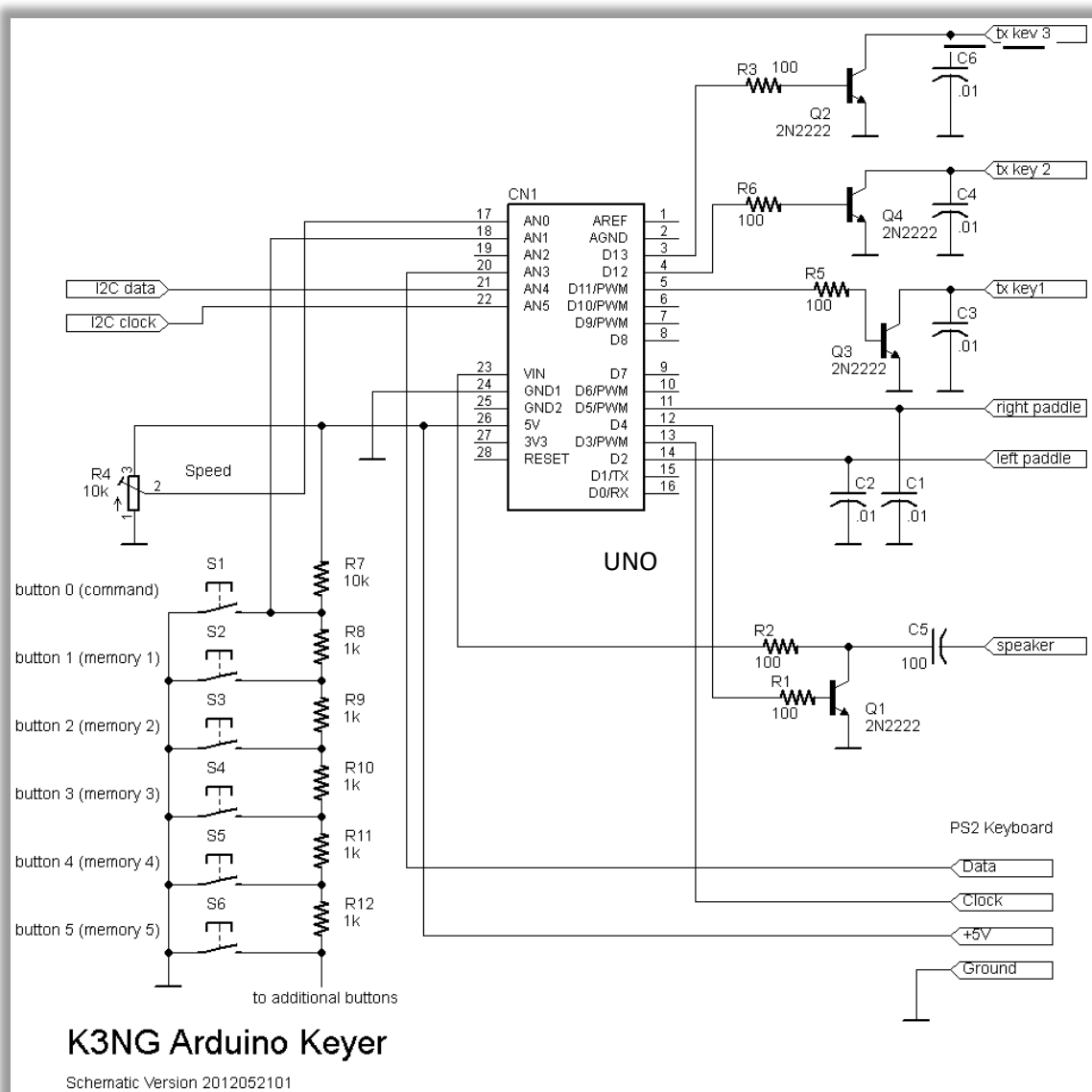


### 5. K3NG CW Keyer, UNO

Hieronder ziet je het originele schema, fig.17, van de Arduino K3NG CW keyer gebaseerd op de UNO en is te vinden onder:

[https://www.arduino.cc/nl/uploads/Main/arduino-mega2560\\_R3-sch.pdf](https://www.arduino.cc/nl/uploads/Main/arduino-mega2560_R3-sch.pdf)

In hoofdlijnen wordt dit schema in deze handleiding aangehouden, maar met gewijzigde poort bezetting ten behoeve voor de MEGA 2560 en verder een aanvulling voor een display en PS2 toetsenbord aansluiting.



Figuur 17, K3NGbasis schema

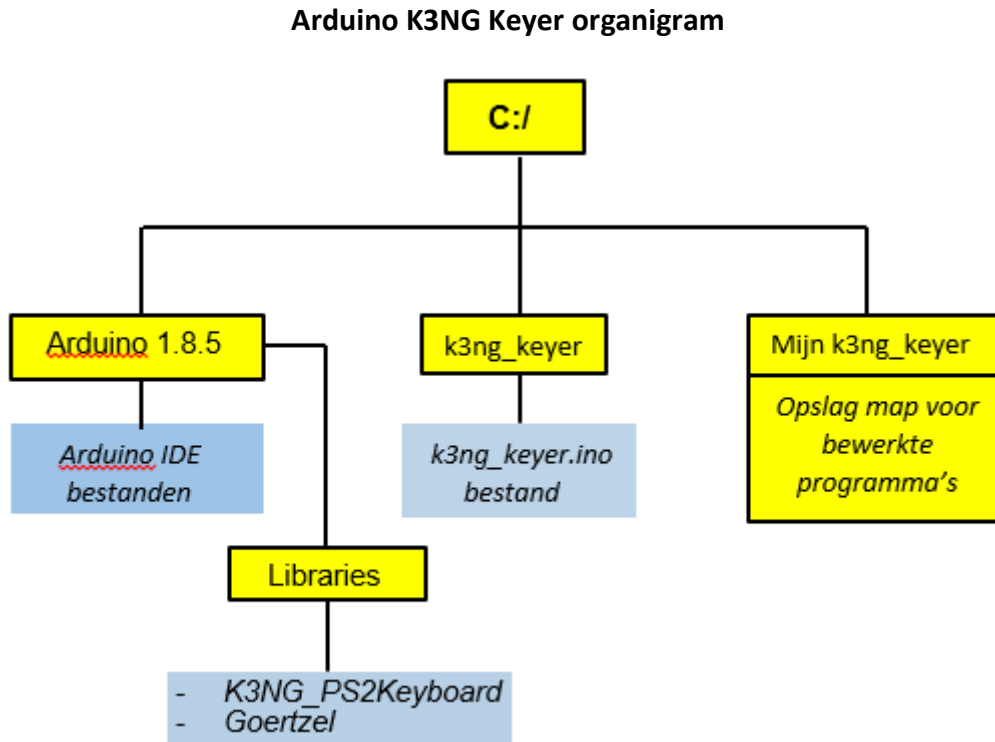


Hierbij de opmerking dat alle gebruikte C's, keramische schijf condensatoren mogen zijn, dus zonder polariteit.

## 6. Indelen van de K3NG CW keyer bestanden.

De map structuur is heel belangrijk voor een juiste werking van het project en daar is jammer genoeg weinig over te vinden op internet vandaar dat ik hieronder, met behulp van een organigram, fig.18, laat zien hoe je de Arduino structuur moet opzetten.

De gele rechthoeken tonen de mappen en de blauwe gekleurde rechthoeken de bijbehorende bestanden.

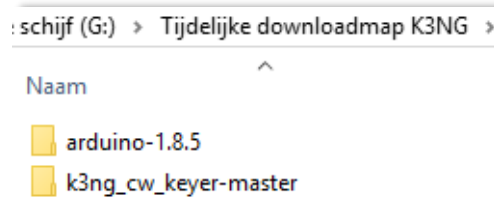
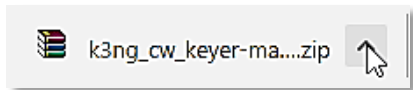


Figuur 18, Arduino K3NG organigram

Maak een nieuwe map aan met de naam: *k3ng\_keyer* onder *C/:*

Voor de K3NG keyer moet je onderstaande zip bestand downloaden en in je *Tijdelijke downloadmap* zetten. Het tussenstapje om het eerst in een tijdelijke map te zetten lijkt wat omslachtig, maar biedt het voordeel dat je bij een herinstallatie alle basis bestanden weer snel bij de hand hebt.

[https://github.com/k3ng/k3ng\\_cw\\_keyer](https://github.com/k3ng/k3ng_cw_keyer)



Figuur 19, tijdelijke download map

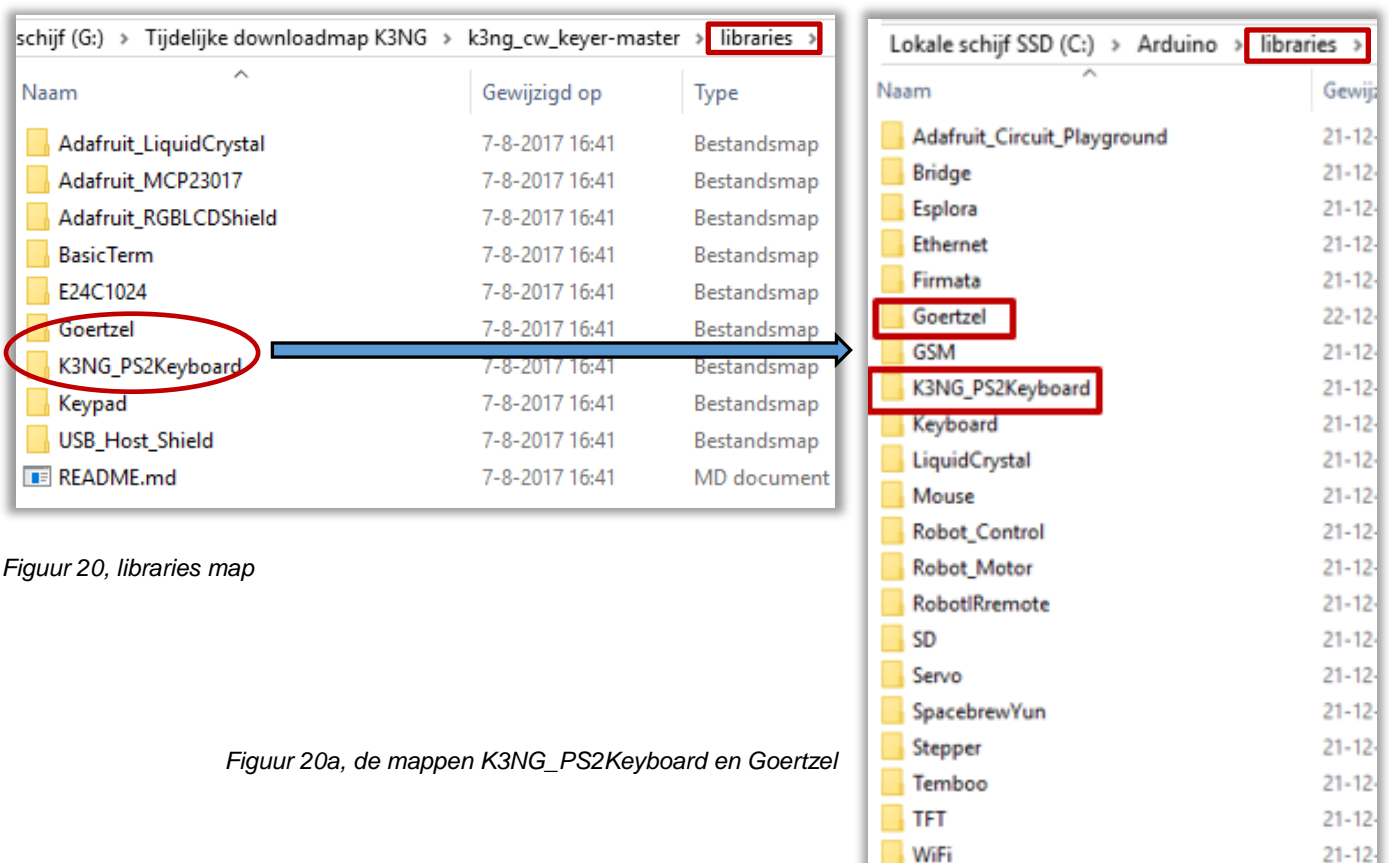
### 6.1 De bestanden op de juiste plek zetten.

Er van uitgaande dat de positie van Arduino IDE, uit het Blink project, nog op deze dezelfde plaats staat, ga je het volgende, overeenkomstig het organigram, doen:

Ga naar de map **libraries** via:

*Tijdelijke downloadmap > k3ng\_cw\_keyer-master > libraries*

Kopieer de map *K3NG\_PS2Keyboard* en *Goertzel* uit deze bibliotheek, fig.20 en plak deze in de map van de *Arduino* bibliotheek, fig.20a.

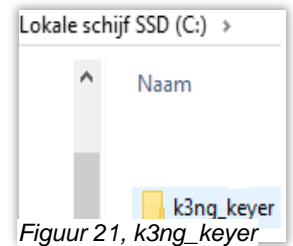


Figuur 20, libraries map

Figuur 20a, de mappen K3NG\_PS2Keyboard en Goertzel

Ter verduidelijking, de *PS2 keyboard* map wordt aan de bibliotheek toegevoegd, omdat je later je toetsenbord gaat aansluiten en Goertzel om Morse te decoderen op de display. Op deze manier kan Arduino IDE het benodigde bestand vinden en verder verwerken. (Met dank voor de hulp van Fred, VK2EFL).

Als je met de opbouw van het K3NG keyer programma, in het volgende hoofdstuk, begint heb je het bestand *k3ng\_keyer nodig*. Ga naar *k3ng\_cw\_keyer-master* en kopieer daaruit de map *k3ng\_keyer* en plak deze in C:/, fig.21. De map- en bestanden indeling is hiermee compleet en je kunt nu eindelijk beginnen met het Arduino K3NG Keyer project zelf.



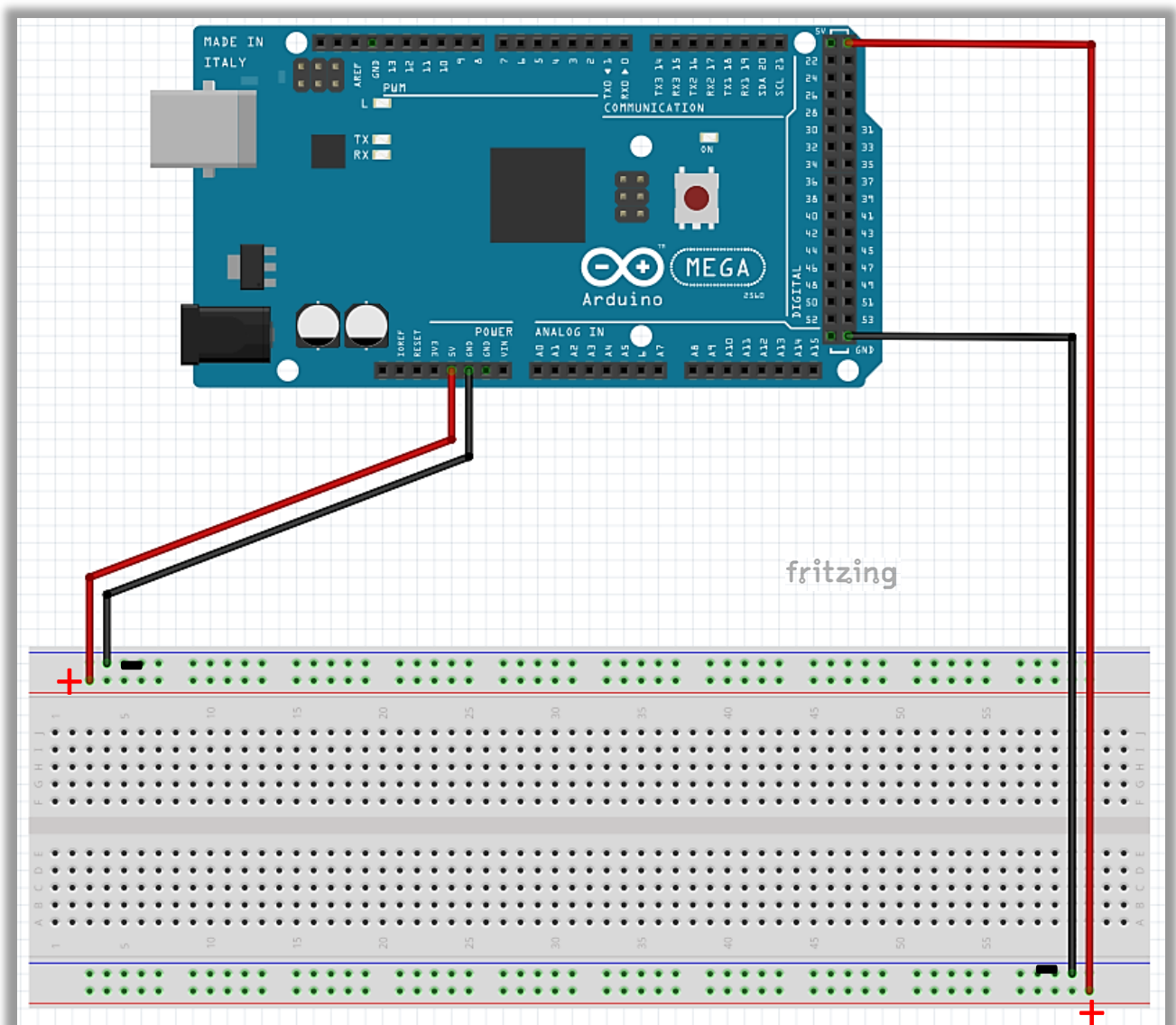
Figuur 21, k3ng\_keyer

## 7. De K3NG Arduino MEGA 2560 opbouw.

### 7.1 Beginnen met de componenten opbouw van de Arduino K3NGkeyer.

Verbreek elke keer voordat je aan je breadboard gaat werken de USB verbinding van je PC.

Je kunt nu eindelijk beginnen met de fysieke bouw van de keyer. Het breadboard dat ik gebruik komt overeen met het getoonde op pagina 7, fig.3.

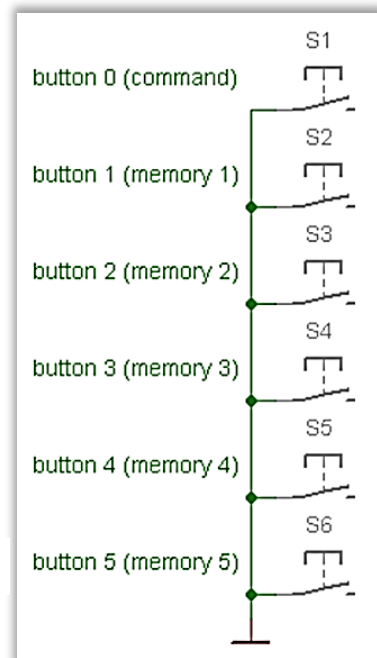


Figuur 22, basis breadboard opstelling met voedingslijnen vanaf de Arduino MEGA

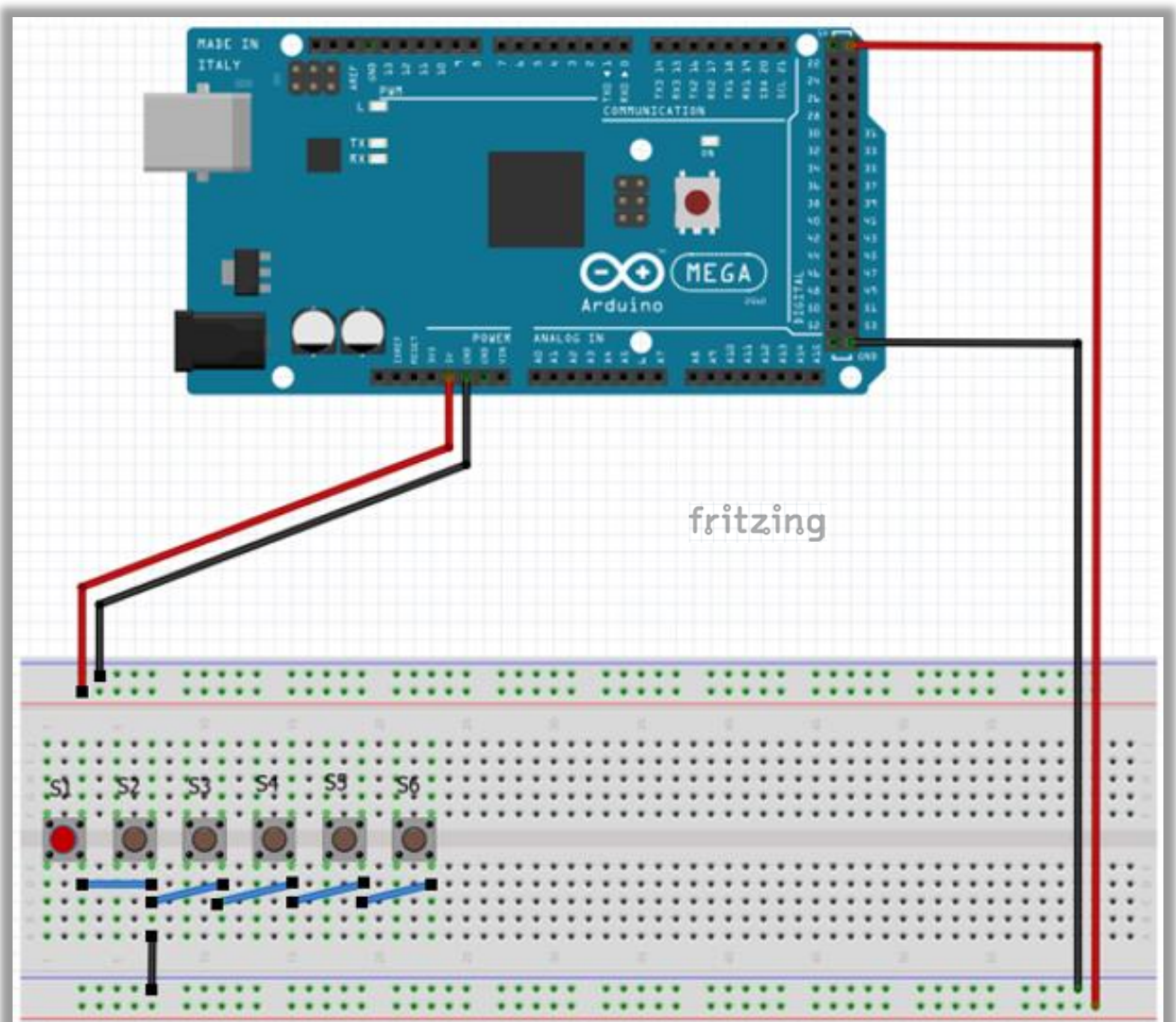
## 8. Het Stappenplan.

### Stap 8.1: Toevoegen van de knoppen.

Als eerste stap in dit project plaats je de knoppen zoveel mogelijk links op het board, fig. 24. Het zijn er in totaal zes, waarvan de meest linkse de **Commando** knop voor de programmering is. De andere knoppen zijn **Geheugen** knoppen. Je kunt later meerdere geheugen knoppen bijvoegen als je dat wilt. De knoppen zijn eenvoudige aan-uit schakelaartjes, met elkaar verbonden overeenkomstig het schema, fig.23.



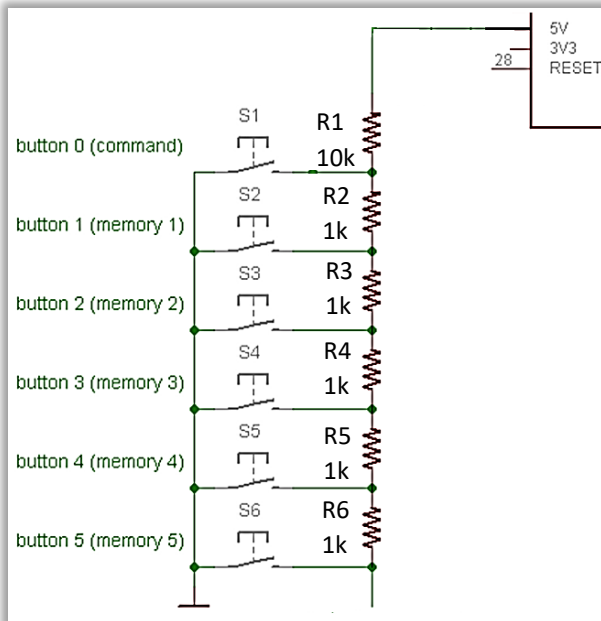
Figuur 23, schema met knoppen.



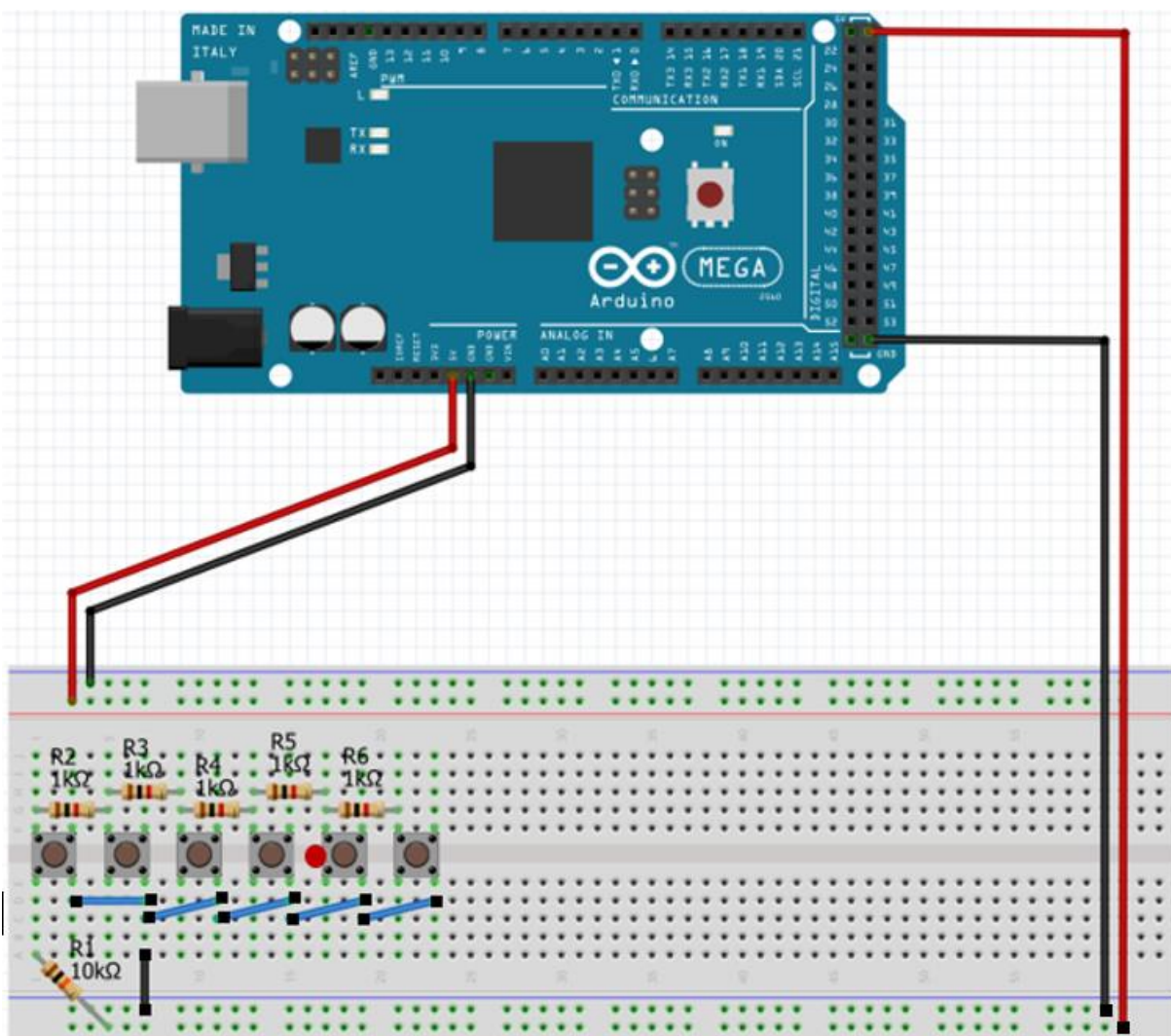
Figuur 24, de knoppen.

**Stap 8.2: Toevoegen van de weerstanden.**

Plaats de 6 weerstanden op het bord, fig. 26 en let goed op de positie van de weerstanden, een ver-gissing is zo gemaakt. Fig. 25 laat nog eens de schakeling zien.



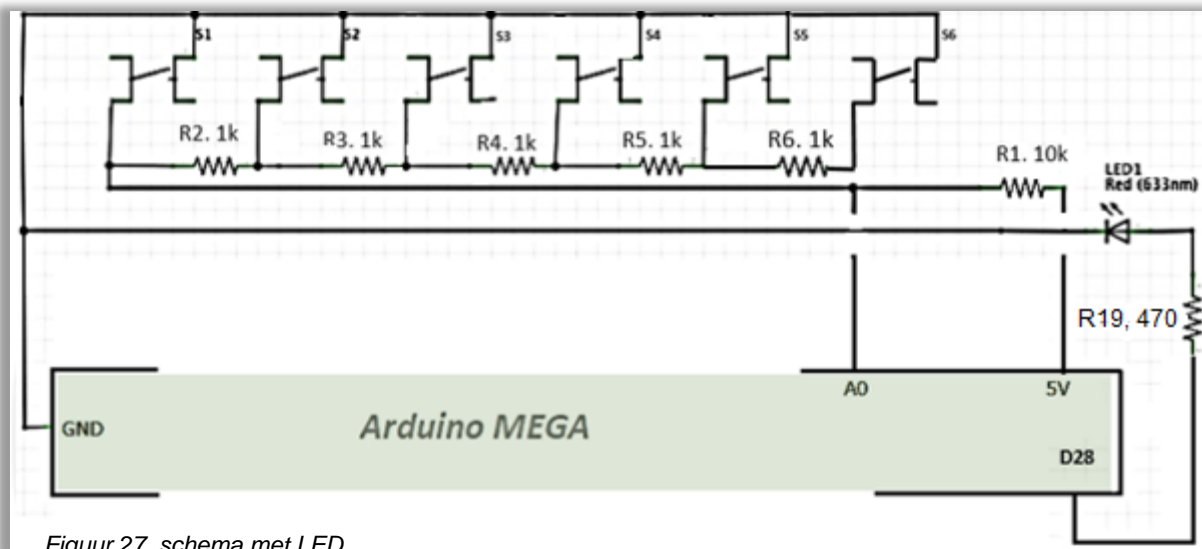
*Figuur 25, een zestal weerstanden.*



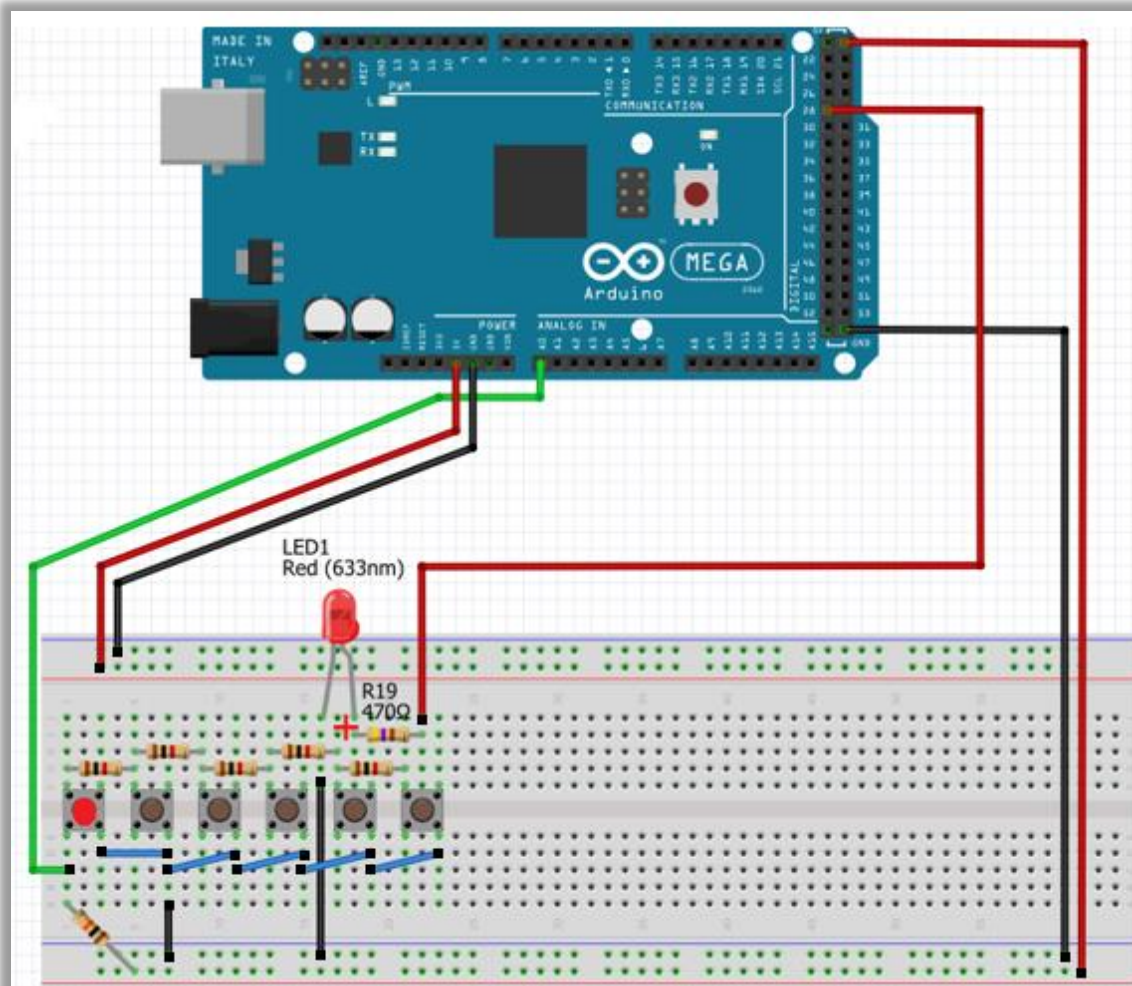
*Figuur 26, knoppen met weerstanden.*

**Stap 8.3: Toevoegen van een LED.**

Deze stap is ook op de K3NG website terug te vinden, en geeft een controle mogelijkheid of de Commando knop is ingedrukt en wordt in de volgende stappen meegenomen, fig.27 en 28. Om te controleren of de Commandoknop is ingedrukt wordt een ledje toegevoegd. Eenmalig de Commandoknop kort indrukken (digitale poort D28 wordt "hoog", ca. 5 Volt) laat het ledje branden, nogmaals kort indrukken, gaat het ledje weer uit. Let goed op hoe het ledje wordt geplaatst, de langste aansluitdraad is de + en is met de 470 Ohm weerstand verbonden.



Figuur 27, schema met LED



Figuur 28, opstelling



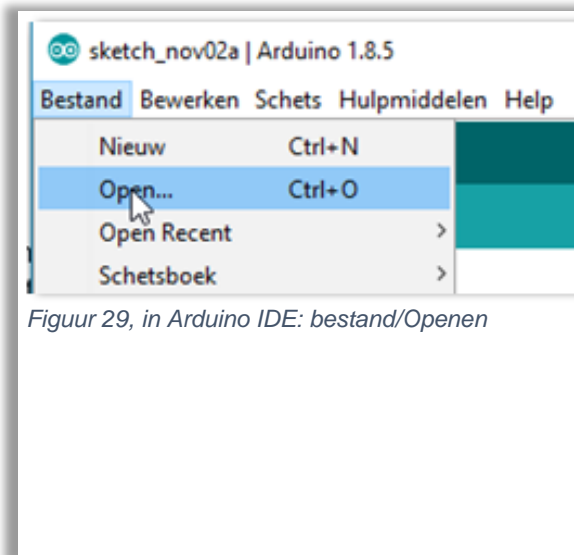
**Stap 8.4: De data van de K3NG keyer laden in Arduino IDE**

Open het Arduino IDE programma.

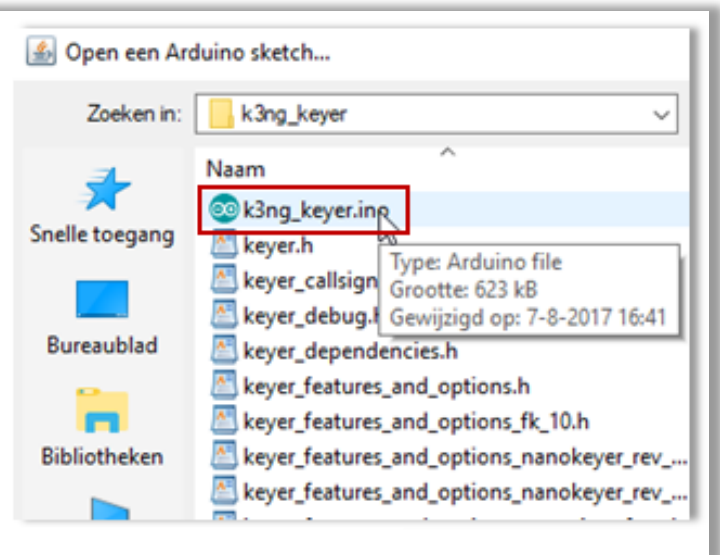
Om het basis K3NG programma in Arduino IDE te openen ga je naar **Open** en klik je op het **k3ng\_keyer.ino** bestand uit de map:

**SSD (C:) > Arduino > k3ngkeyer**, fig. 29 en 29a.

Daarin vind je alle noodzakelijke **.h** bestanden die nodig zijn om de keyer op te bouwen.



Figuur 29, in Arduino IDE: bestand/Openen

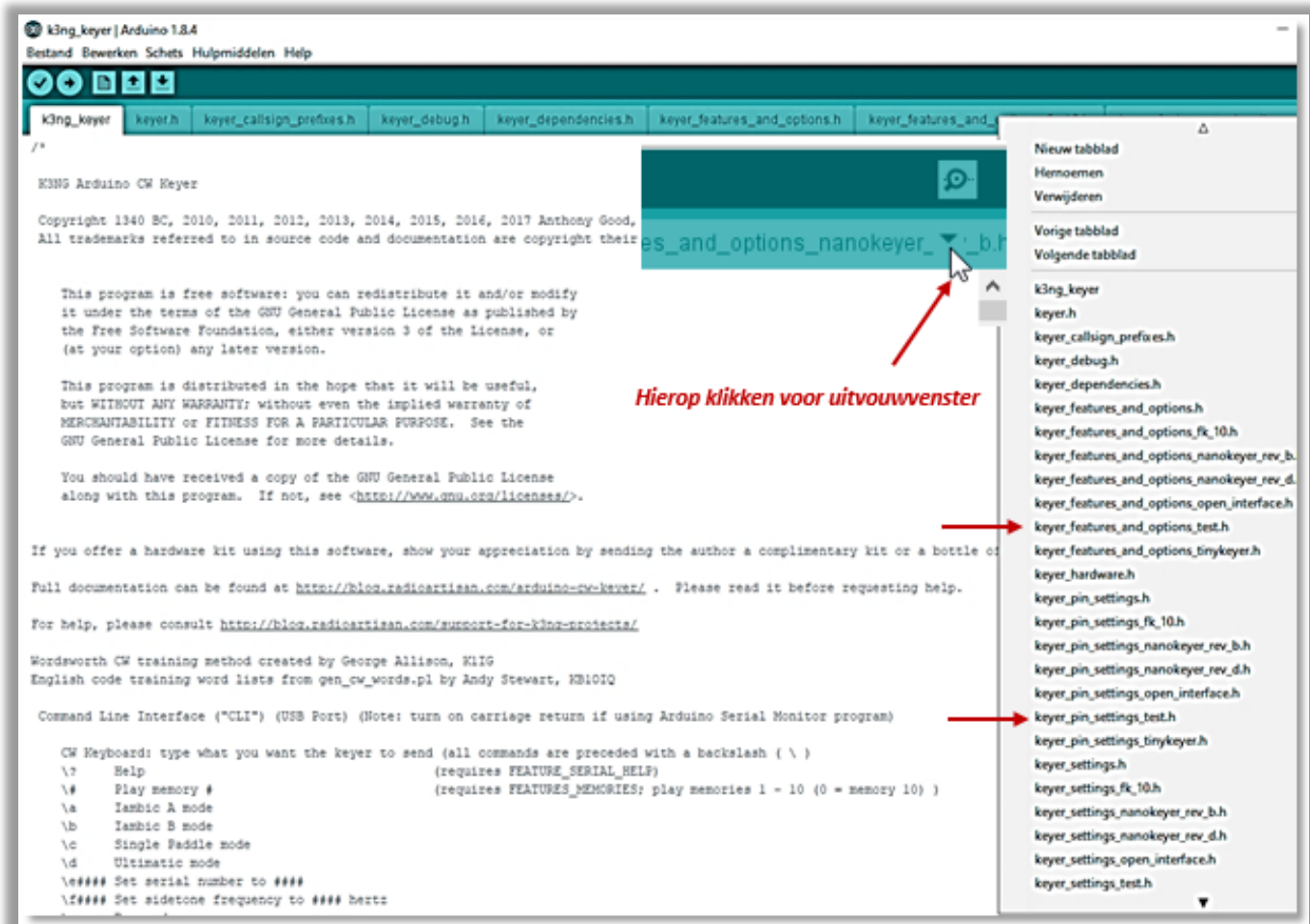


Figuur 39a, basis k3ng\_keyer.ino bestand

Figuur 29, open k3ng\_keyer.ino

In de onderstaande horizontale groene balk van het hoofdvenster verschijnen nu alle benodigde `.h` bestanden, die je wat duidelijker kunt zien als je het verborgen uitvouwvenster opent, fig.30.

Je gaat nu uit van de basis K3NG configuratie en breng daarna de nodige modificaties aan. Wat je met de zogenaamde `.h` bestanden kunt/moet doen, wordt je in het vervolg van de handleiding duidelijk gemaakt.



Figuur 30, Arduino IDE hoofd venster

De twee onderstaande `.h` bestanden zijn de noodzakelijke gereedschappen die je gaat gebruiken om een en ander in het programma aan te passen.

**a. `keyer_pin_settings.h`**

Daarin leg je de pin bezetting vast.

**b. `keyer_features_and_options.h`**

Daarin kun je bepaalde opties uit het K3NG programma, die je voor je project nodig hebt, aan- of uit zetten.

## Stap 8.5: Het K3NG basis programma aanpassen

*keyer\_pin\_settings.h*, waarmee je de pinsetting bepaalt.

```
#ifdef FEATURE_COMMAND_BUTTONS
  #define analog_buttons_pin A0
  #define command_mode_active_led 28
#endif //FEATURE_COMMAND_BUTTONS
```

Figuur 31, pin settings


Voor de analoge pin is gekozen voor: A0. (Momenteel is *Command\_mode\_active\_led* ingesteld op een niet-actieve staat). De digitale pin voor poort 28, fig.31, die de led schakelt.

In dit stadium moet je een drietal commando's activeren, fig.32 en dat doe je door de *//* tekens te verwijderen. Vergeet dat niet, anders zal het programma niet werken.

*keyer\_features\_and\_options.h*.

```
#define FEATURE_COMMAND_BUTTONS
// #define FEATURE_COMMAND_LINE_INTERFACE
#define FEATURE_MEMORIES
#define FEATURE_MEMORY_MACROS
```

Figuur 32, functie's activeren.

 Met de "Bewerken > Zoek..." vind je een commando snel terug in het programma.

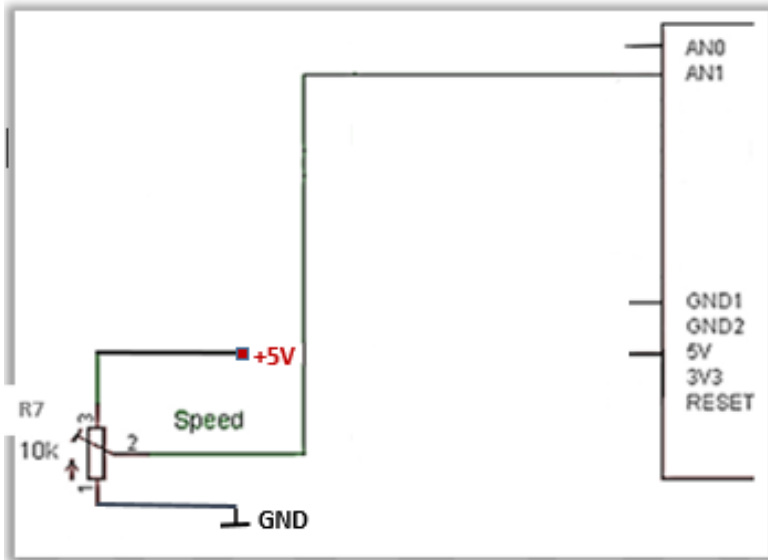
Zodra je de wijzigingen in de codes hebt aangebracht, kunt je het programma compileren en, na het aansluiten van je Arduino MEGA, uploaden naar de Arduino Mega voor het testen. Ook hier geldt: weet je zeker dat alle componenten correct zijn aangesloten?.

Vergeet niet het programma te compileren en updaten.

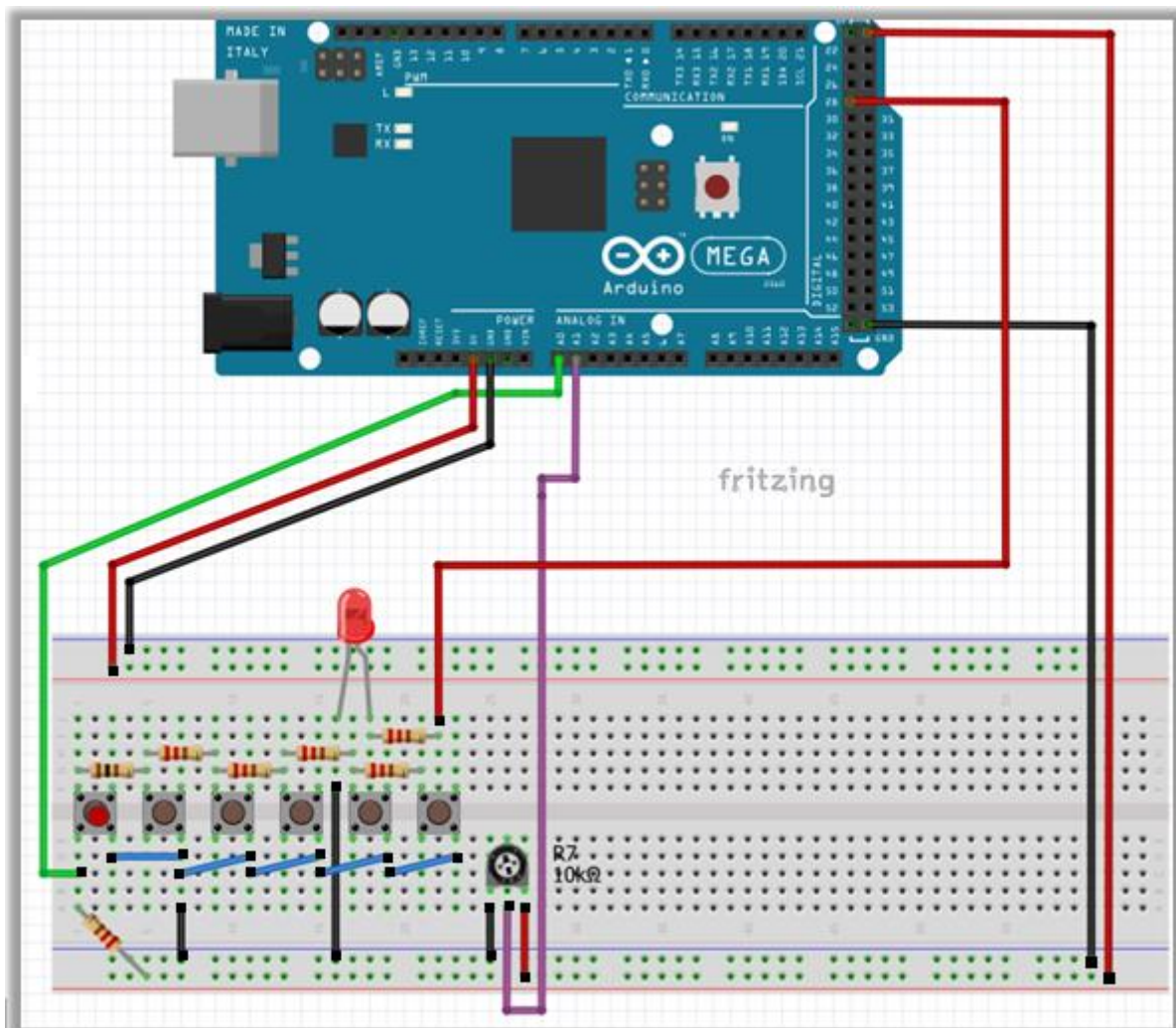
Na het drukken op de Commando knop, moet de led oplichten en nogmaals op deze knop drukken zal deze uitgaan.

**Stap 8.6: Met potentiometer uitbreiden.**

Hieronder vind je het schema, fig.33 en breadboard opstelling, fig.34, met de bijgevoegde potentiometer, waarmee je later de seinsnelheid kunt instellen van 1 - 999 woorden per minuut (WPM).



*Figuur 33, schema met pot meter.*



*Figuur 34, breadboard opstelling met potentiometer.*

Om de potentiometer te laten werken, moet je het volgende doen:

Selecteer *keyer\_pin\_settings.h*, waarmee je de pin setting bepaalt. Omdat de analoge poort A0 al is bezet, kies je hiervoor poort A1, fig.35.

#### *keyer\_pin\_settings.h*

```
// #define SIDETONE_PIN 10
#define potentiometer A1
#define ptt_tx 1.0
```

Figuur 35, Analoge poort

In het *keyer\_features\_and\_options.h* bestand de volgende opdrachten activeren, ( // weghalen ) fig. 36/37:

#### *keyer\_features\_and\_options.h*

```
// #define FEATURE_TRAINING_COMMAND
#define FEATURE_POTENTIOMETER
// #define FEATURE_SIDETONE_SWITCH
```

Figuur 36, Potentiometer optie

```
// #define FEATURE_SLEEP
#define FEATURE_ROTARY_ENCODER
// #define FEATURE_CMOS_SUPER_K
```

Figuur 37, Rotary Encoder optie

Als afsluiting van deze stap het programma compileren, uploaden heeft hier geen zin.

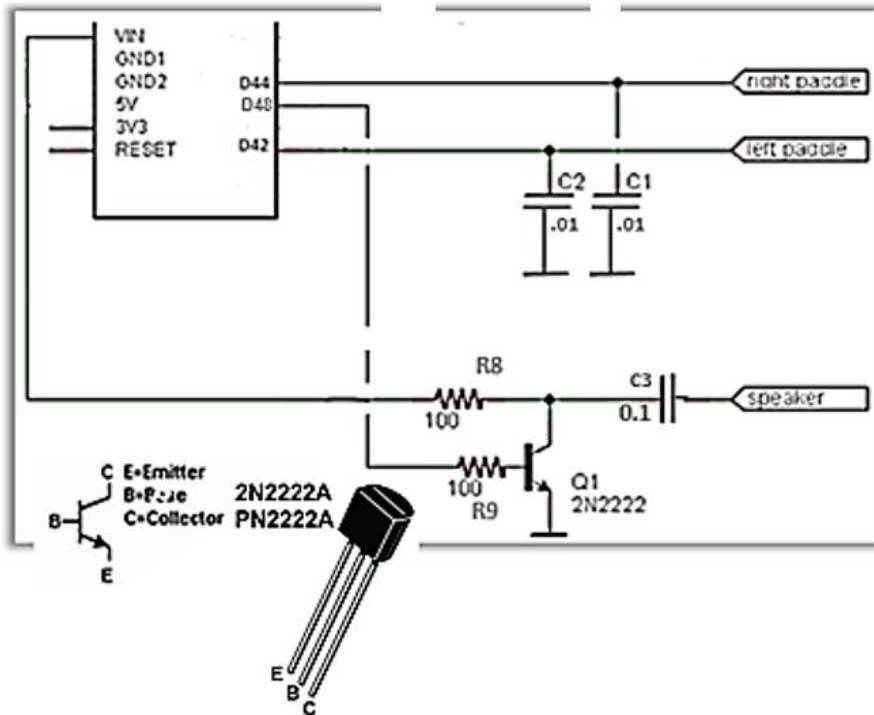
Opmerking:

Als je de eerste keer compileert zul je waarschijnlijk onderstaande bericht krijgen, fig. 38. De rode tekst heeft geen alarmerende betekenis, onderstaande tekst geeft aan hoeveel opslagruimte er gebruikt is van de totale beschikbare opslagruimte.

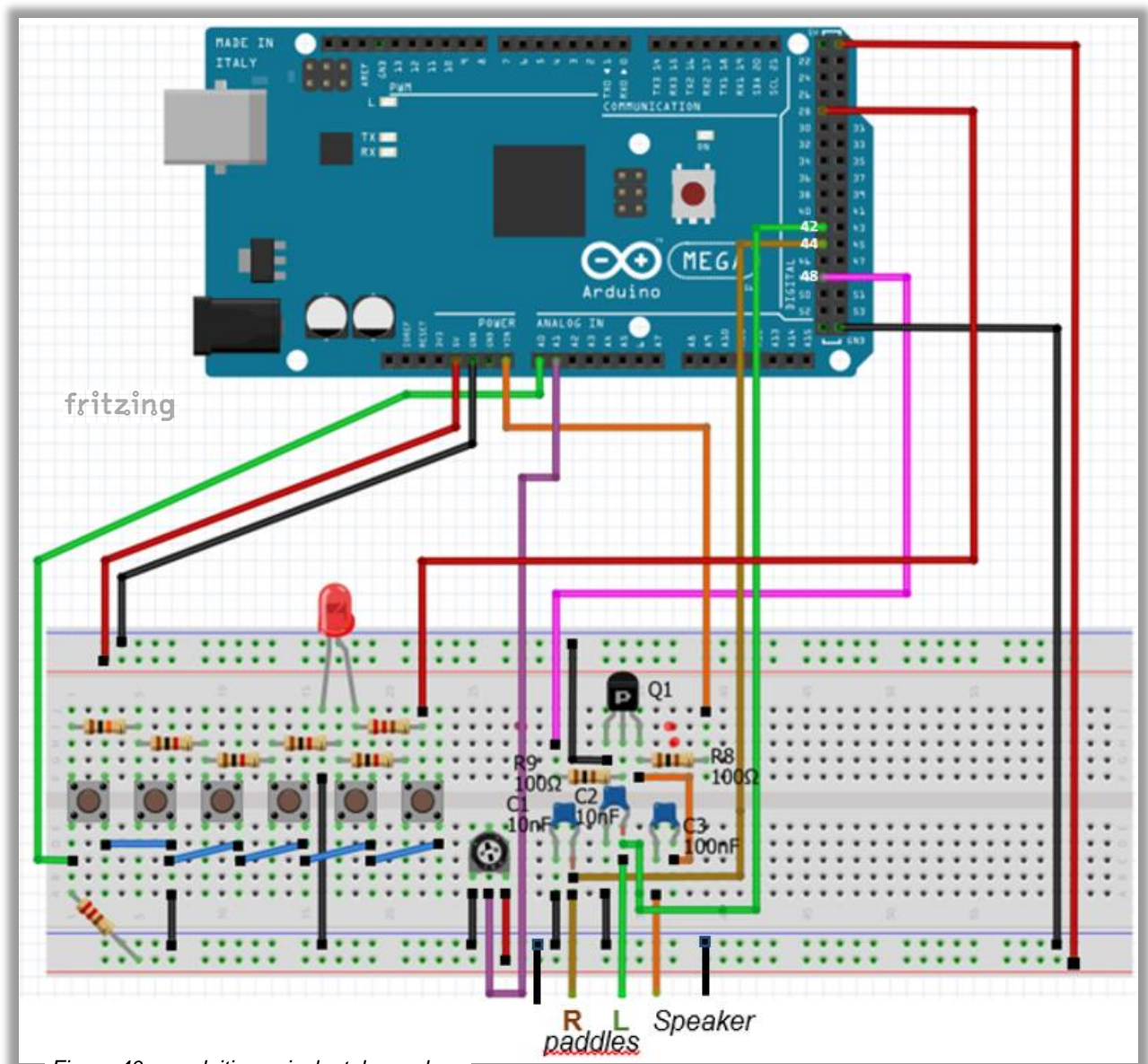
```
Compileren voltooid.
Archiving built core (caching) in: C:\Users\VANDER~1\AppData\Local\Temp\arduino_cache_
De schets gebruikt 23026 bytes (9%) programma-opslagruimte. Maximum is 253952 bytes.
Globale variabelen gebruiken 662 bytes (8%) van het dynamisch geheugen. Resteren 7530
```

Figuur 38, gebruikte opslag ruimte.

Stap 8.7: Seinsleutel en luidspreker uitbreiding.



Figuur 39, seinsleutel en luidspreker



Figuur 40, aansluiting seinsleutel, speaker

Met deze stap, fig.39 en 40, ben je in staat met de seinsleutel het seinschrift om te zetten in geluidsignalen met behulp van een luidsprekertje, misschien alleen interessant voor diegenen die het CW-en onder de knie hebben.

Maar let op: bij stap 8.7 ga je een toetsenbord toevoegen waarmee CW tekens hoorbaar geproduceerd kunnen worden.

En nog leuker wordt het als je in stap 8.8 een display gaat toevoegen waarbij je kunt zien wat je aan CW tekens produceert.

Om de luidspreker en keyer paddles te laten werken, zijn geen speciale commando's in het *keyer\_features\_and\_options.h* bestand nodig.

Wel moet je in het *keyer\_pin\_settings.h* bestand kenbaar maken welke pin setting je voor de sidetone en paddles kiest. Gekozen is voor de luidspreker, pin 48 en voor de paddles pin 42 en 44, fig.41 en 41a.

De 2N2222 transistor is een standaard NPN schakel transistor en moet worden aangesloten volgens het schema.

De seinsleutel- en sidetone pin settings, overeenkomstig de Arduino pinnen, worden:

*keyer\_pin\_settings.h*

```
#define CA_key_line_0 0
#define sidetone_line 48 // connect a speaker for sidetone
#define potentiometer A1 // Speed potentiometer (0 to 5 V) I
```

Figuur 41, side tone pin setting

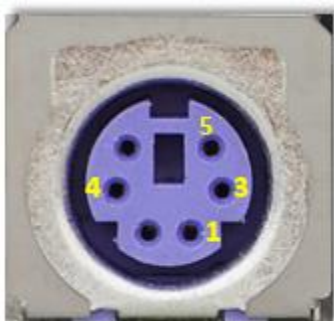
*keyer\_pin\_settings.h*

```
#define paddle_left 42
#define paddle_right 44
#define tx_key_line_1 32
```

Figuur 41a, paddles

**Stap 8.8: Het aansluiten van je toetsenbord.**

Als eerste ga je een toetsenbord (aansluiting) toevoegen, hiermee kun je dan al hoorbare morsetekens produceren, zonder de seinsleutel te gebruiken. Het keyboard dat ik gebruik, is van het type PS/2, een handzaam type dat makkelijk verkrijgbaar is. In fig. 42 is het vooraanzicht van het chassisdeel te zien en daarnaast in fig. 43, waar de bijbehorende pennen aangesloten moeten worden. Na controle de werking testen na compilatie en uploading.



Figuur 42, vooraanzicht chassisdeel

Contact	Naam	Functie
1	+DATA	Gegevens <b>A3</b>
2	Gereserveerd	Gereserveerd*
3	Massa	Massa <b>GND</b>
4	Voeding	+5 V DC (100 mA)
5	+CLK	Klok <b>CLOCK</b>
6	Gereserveerd	Gereserveerd*

Figuur 43, pen benamingen.

Het aansluiten van het toetsenbord is “een fluitje van een cent” Maak overeenkomstig de onderstaande figuren 44 en 45 de volgende commando’s actief, fig.46 toont de toegevoegde bedrading.

*keyer\_pin\_settings.h* bestand:

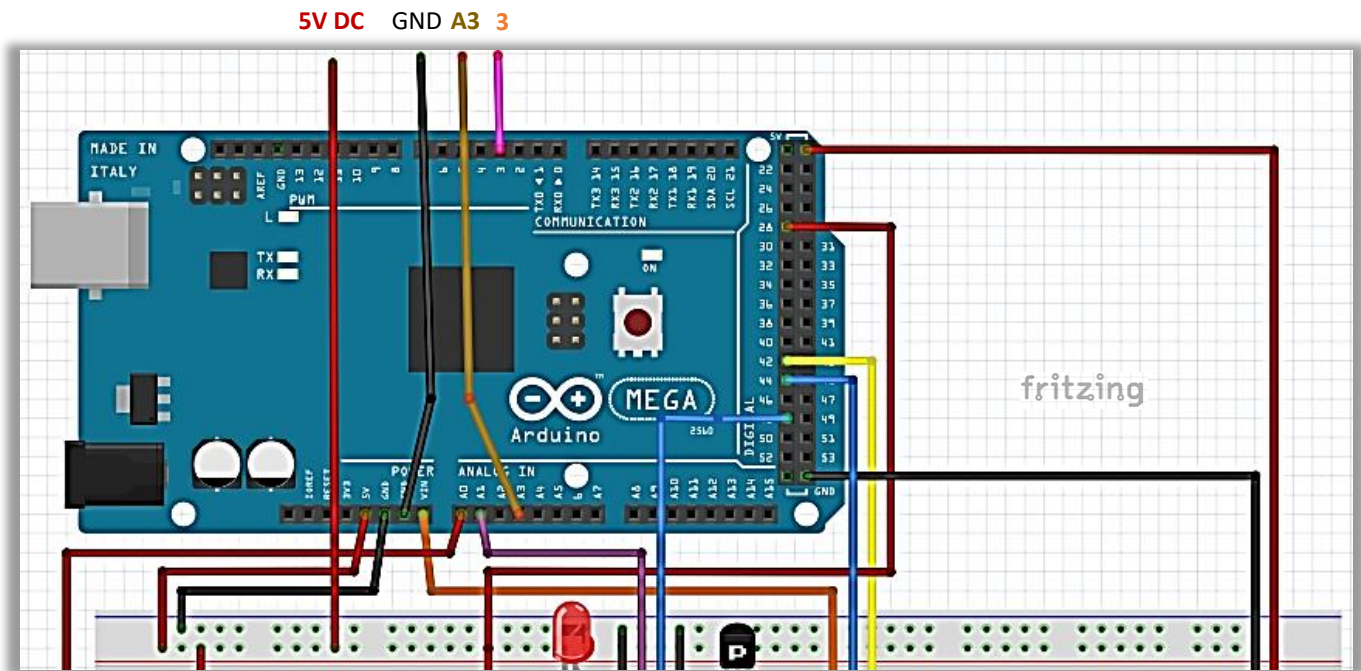
```
//ps2 keyboard pins
#ifdef FEATURE_PS2_KEYBOARD
  #define ps2_keyboard_data A3
  #define ps2_keyboard_clock 3 // this must be on an interrupt capable pin!
#endif //FEATURE_PS2_KEYBOARD
```

Figuur 44, pin settings bepalen.

*keyer\_features\_and\_options.h*

```
// #define FEATURE_HELL
#define FEATURE_PS2_KEYBOARD
// #define FEATURE_USB_KEYBOARD
```

Figuur 45, feature\_ps2\_keyboard activeren.



Figuur 46, GND (zwart), A3 (bruin) en 3 (roze) en de 5V aansluiting (rood)

Tijdens het compileren krijg je de volgende waarschuwing, fig. 46a, met betrekking tot je toetsenbord. Deze waarschuwing mag je negeren en overgaan tot het uploaden.

```
Compilieren voltooid.
WAARSCHUWING: Categorie 'AmateurRadio' in bibliotheek K3NG_PS2Keyboard is niet geldig. Wordt gewijzigd naar 'Uncategorized'
De schets gebruikt 27532 bytes (10%) programma-opslagruimte. Maximum is 253952 bytes.
Globale variabelen gebruiken 803 bytes (9%) van het dynamisch geheugen. Resteren 7389 bytes voor lokale variabelen. Maximum is 8192 bytes.
```

Figuur 46a, de waarschuwingsvermelding



**Stap 8.9: Het aansluiten van je display.**

Door het toepassen van een display wordt het project pas echt leuk, want je kunt nu ook zien wat je met je toetsenbord, en eventueel je sleutel, produceert aan morseteksten.

Om het een en ander wat overzichtelijker te maken heb ik gebruik gemaakt van een 2<sup>e</sup> breadboard die je ook moet je voorzien van een 5V en GND aansluiting. De display is een 4-bits 2-rij LCD. Later kan op het 2<sup>e</sup> breadboard dan ook de decodeer opstelling naast. De beide breadbordjes zijn dan goed gevuld. Wijzig in de *keyer\_pin\_settings.h* en *keyer\_features\_and\_options.h* bestanden het volgende, fig.47 en 48. Figuren 50 en 51 tonen de breadboard opstelling.

*keyer\_pin\_settings.h* bestand:

```
//lcd pins
#ifdef FEATURE_LCD_4BIT
#define lcd_rs 38
#define lcd_enable 31
#define lcd_d4 33
#define lcd_d5 35
#define lcd_d6 37
#define lcd_d7 39
#endif //FEATURE_LCD_4BIT
```

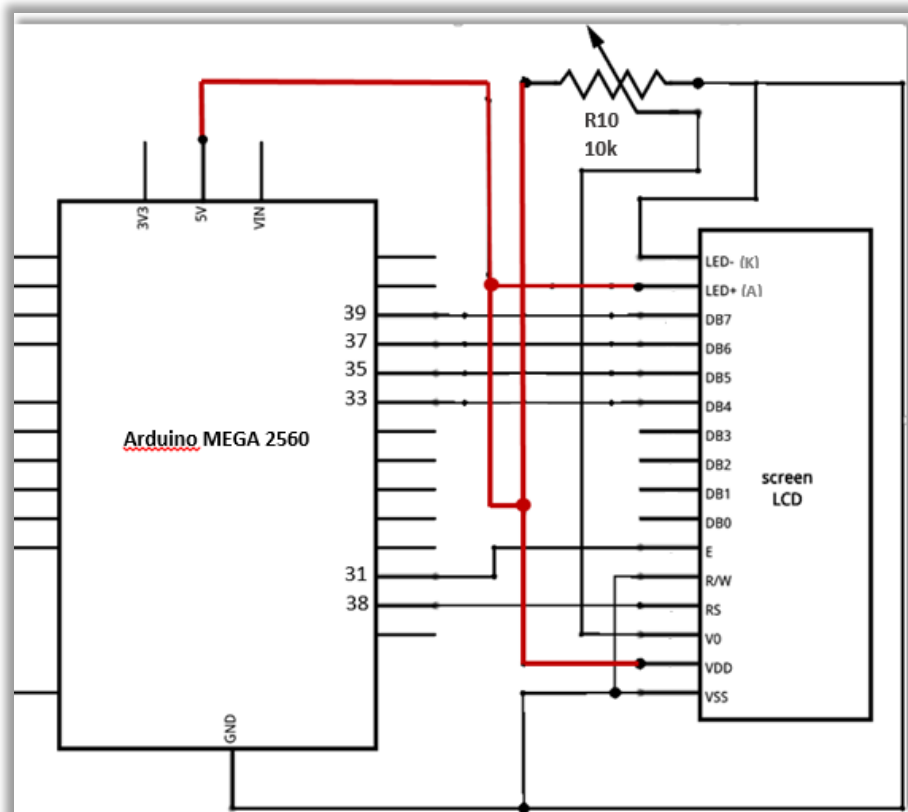
Figuur 47, LCD setting

*keyer\_features\_and\_options.h* bestand:

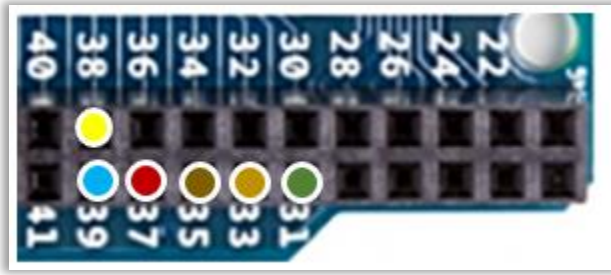
```
// #define FEATURE_DL2SBA_BANKSWITCH
#define FEATURE_LCD_4BIT
// #define FEATURE_LCD_ADAFRUIT_I2C
```

Figuur 48, feature LCD

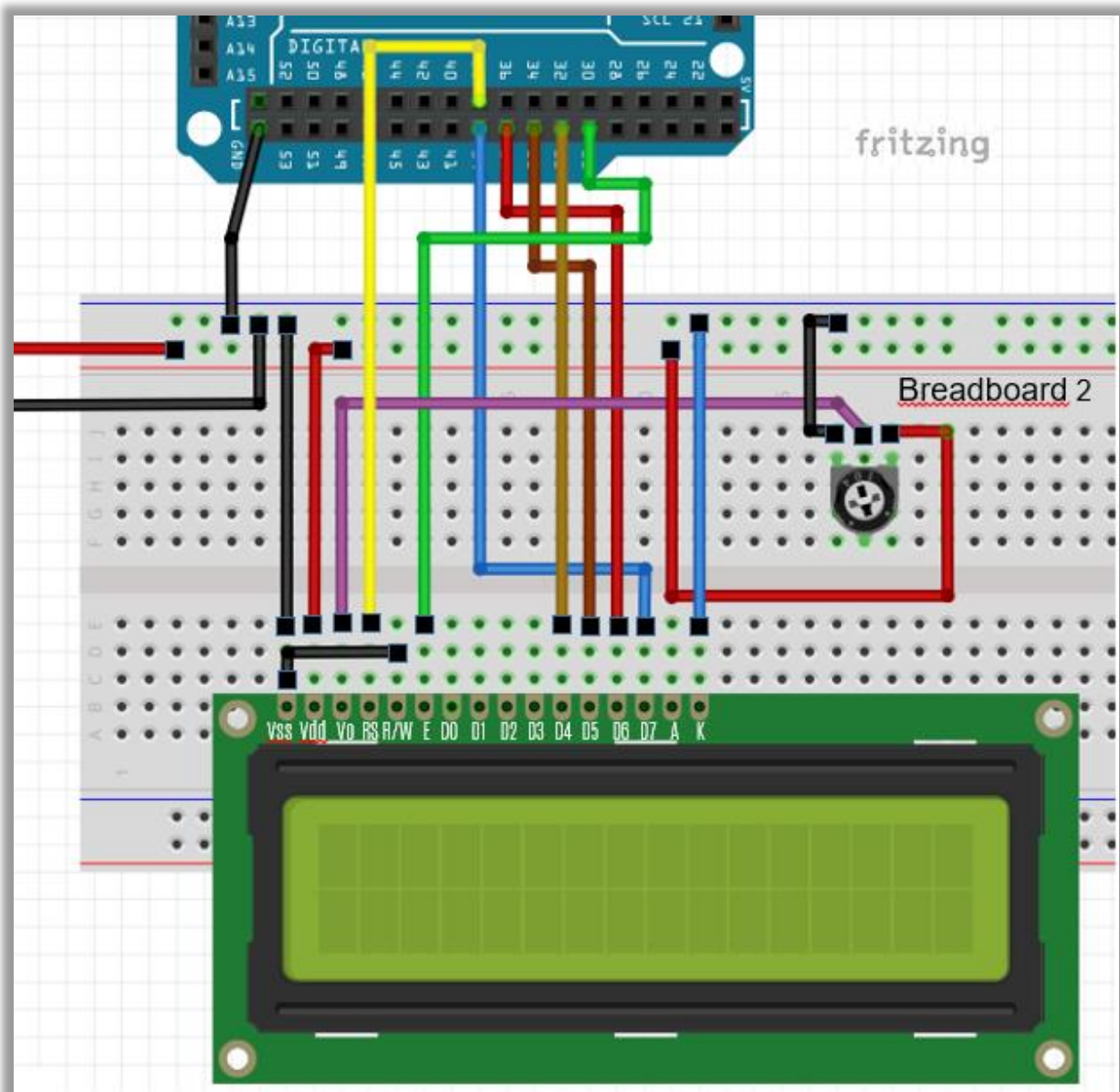
Als je het onderstaand schema aanhoudt, fig. 49, dan zal de display goed moeten werken. De potmeter R10 zodanig instellen dat de karakters zichtbaar worden.



Figuur 49, LCD schakeling



Figuur 50, detail pin setting



Figuur 51, LCD opstelling, 4-bits 2-rij LCD

### Stap 8.10: De verbinding met de Transceiver.

Bouw nu verder op je 1<sup>e</sup> breadboard.

Het K3NGontwerp heeft 3 TX aansluitingen, hetgeen betekent dat je er 3 transceivers op aan kunt sluiten, bijvoorbeeld een Kenwood, Icom en een Yaesu. Ik stel mij zo voor dat de drie transceivers op verschillende banden zijn ingesteld zodat, als de ene band “dicht” gaat, je met de andere transceiver op een andere band verder kunt contesten. Misschien een beetje ver gezocht, maar de mogelijkheid wordt wel geboden.

Om deze optie te activeren moet je de poorten wijzigen voor de `tx_key_line_1`, `tx_key_line_2` en `tx_key_line_3` die in het bestand [keyer\\_pin\\_settings.h](#) worden weergegeven. Ik heb die van mij op de digitale pin 32, 34 en 36 gezet, fig.52.

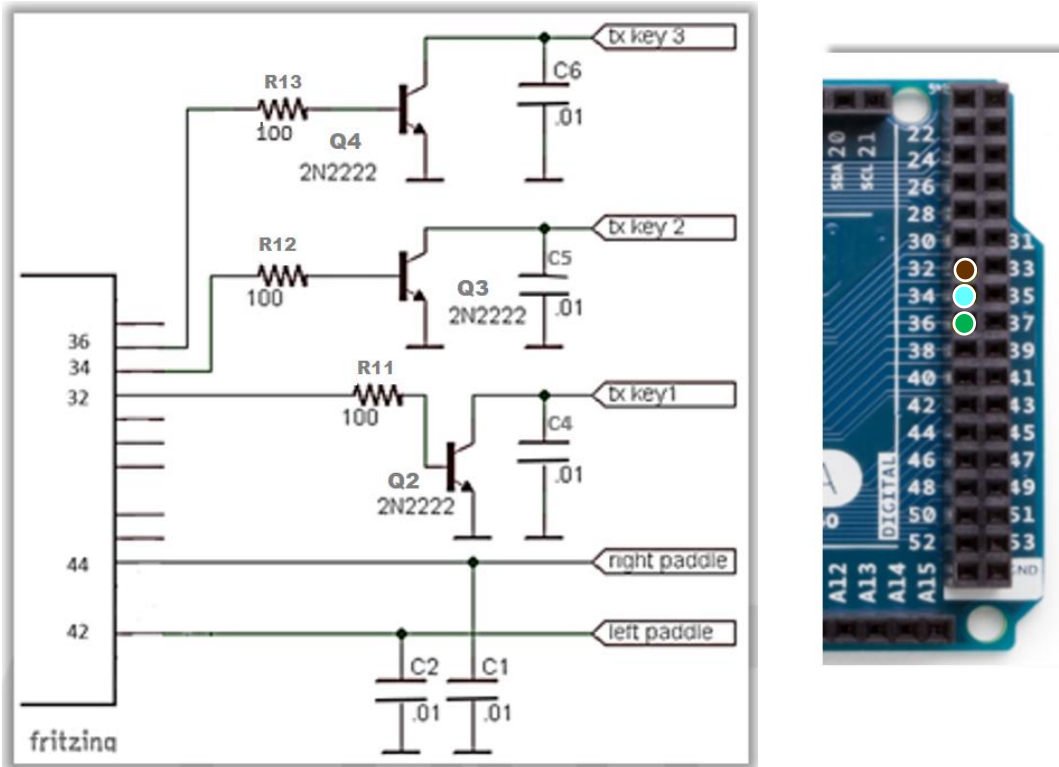
```
#define paddle_right 44
#define tx_key_line_1 32 // (high = key down/tx on)
#define tx_key_line_2 34
#define tx_key_line_3 36
#define tx_key_line_4 0
```

Figuur 52, de drie TX uitgangen geactiveerd.

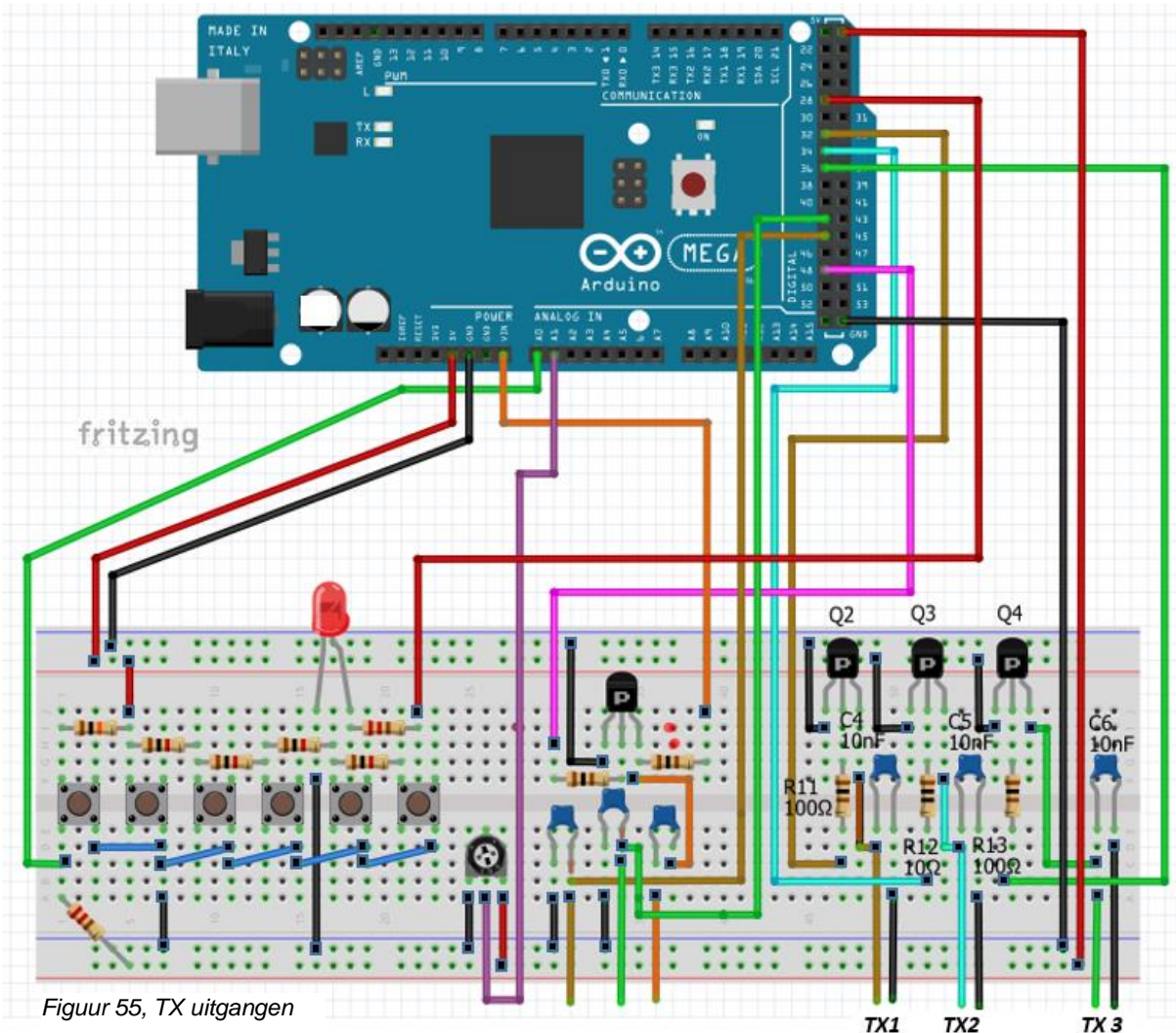
De CW-ers kunnen nu de zender “sleutelen”, waarbij zij de terugkomende morse tekst van het tegenstation op het gehoor kunnen “lezen”.

Met het toetsenbord kun je de zender ook in CW mode aan sturen, maar terugkomende morsesignalen zullen voor de “niet CW-ers” niet ontcijferd kunnen worden, maar daar ga je in het volgende hoofdstuk wat aan doen.

Figuren 53 en 54 en 55 laten het schema en toegevoegde bedrading op het breadboard zien.



Figuur 53, TX aansluitschema



Figuur 55, TX uitgangen

### Stap 8.11: GEORTZ DSP CW DECODER.

Het "sleutel" hoofdstuk is nu afgesloten en het wordt tijd dat je nu eens gaat kijken naar het decodeer gedeelte waarmee morse gedecodeerd wordt naar leesbare tekst op je display.

Dit deel is in grote lijnen overgenomen van de KF4BZT site, waarin hij beschrijft dat ook daar een omzetting moet plaats vinden van de UNO naar de MEGA 2560 controller. Hij maakt gebruik van de basis code van OZ1JHM dat op deze website staat:

<http://www.skovholm.com/cwdecoder>

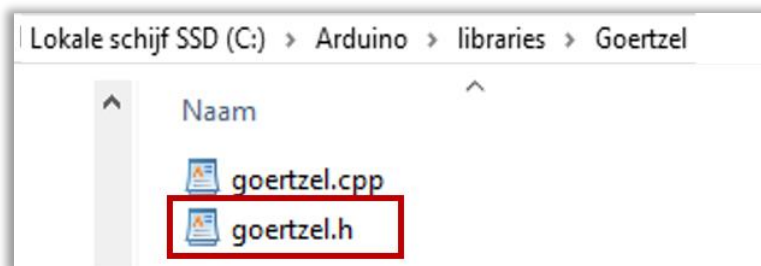
Het decoderen is gebaseerd op het Goertzel-algoritme en je kunt de informatie vinden op:

[https://en.wikipedia.org/wiki/Goertzel\\_algorithm](https://en.wikipedia.org/wiki/Goertzel_algorithm)

Het hele gebeuren is een dusdanig complex mathematisch gebeuren dat ik mij daarin niet in heb verdiept. Er is een bestand voor de Arduino in de bibliotheek opgenomen, genaamd *goertzel.h*.

Deze zorgt voor alle decodeeralgoritmen die nodig zijn om dit goed te laten werken. Daartoe moeten een aantal instellingen aangepast worden, zodat de Arduino Mega met deze code overweg kan.

Dit bestand is te vinden in de Goertzel map, zoals getoond in fig.56.



Figuur 56, map Goertzel

Open het *goertzel.h* bestand met bijvoorbeeld WordPad waarin je enige interessante informatie over de bemonsteringsfrequentie en bandbreedte-informatie kunt vinden

In het eerste deel van het bestand, grijs gedrukt, vind je nadere toelichting van OZ1JHM over de settings. Er zijn instellingen waar aandacht aan moet worden besteed, als de decoding hier niet goed werkt. Volgens OZ1JHM zou de Target Frequentie moeten werken met 558 Hz of misschien 744 Hz.

Verander een en ander in de rood omlijnde rechthoek, fig.57 om de codes geschikt te maken voor de Arduino Mega. Maar onthoud dat een hoge GOERTZ\_SAMPLES waarde veel tijd in beslag neemt, zodat je een compromis moet vinden.

```
#ifndef GOERTZEL_H
#define GOERTZEL_H

/*
Goertzel formula audio detector
This code comes from http://www.skovholm.com/cwdecoder , http://www.skovholm.com/decoder11.ino
Hjalmar skovholm Hansen, OZ1JHM <hjh@skovholm.com>

Notes from the original code author OZ1JHM (with edits from Goody K3NG)

GOERTZ_SAMPLING_FREQ will be 8928 on a 16 mhz without any prescaler etc., because we need the
tone in the center of the bins
you can set GOERTZ_TARGET_FREQ to 496, 558, 744 or 992
then GOERTZ_SAMPLES_INT the number of samples which give the bandwidth
which can be (8928 / GOERTZ_TARGET_FREQ) * 1 or 2 or 3 or 4 etc
int is 8928/558 = 16 * 4 = 64 samples

try to take GOERTZ_SAMPLES = 96 or 128 ;o)

48 will give you a bandwidth around 186 hz
64 will give you a bandwidth around 140 hz
96 will give you a bandwidth around 94 hz
128 will give you a bandwidth around 70 hz

BUT remember that a high GOERTZ_SAMPLES will take a lot of time so you have to find a compromise
*/

// Arduino Due (84 Mhz clock)
// #define GOERTZ_SAMPLING_FREQ 46872.0
// #define GOERTZ_SAMPLES 252 //168 //84

// Arduino Uno, Mega (16 Mhz clock)
#define GOERTZ_SAMPLING_FREQ 8928.0
#define GOERTZ_SAMPLES 64

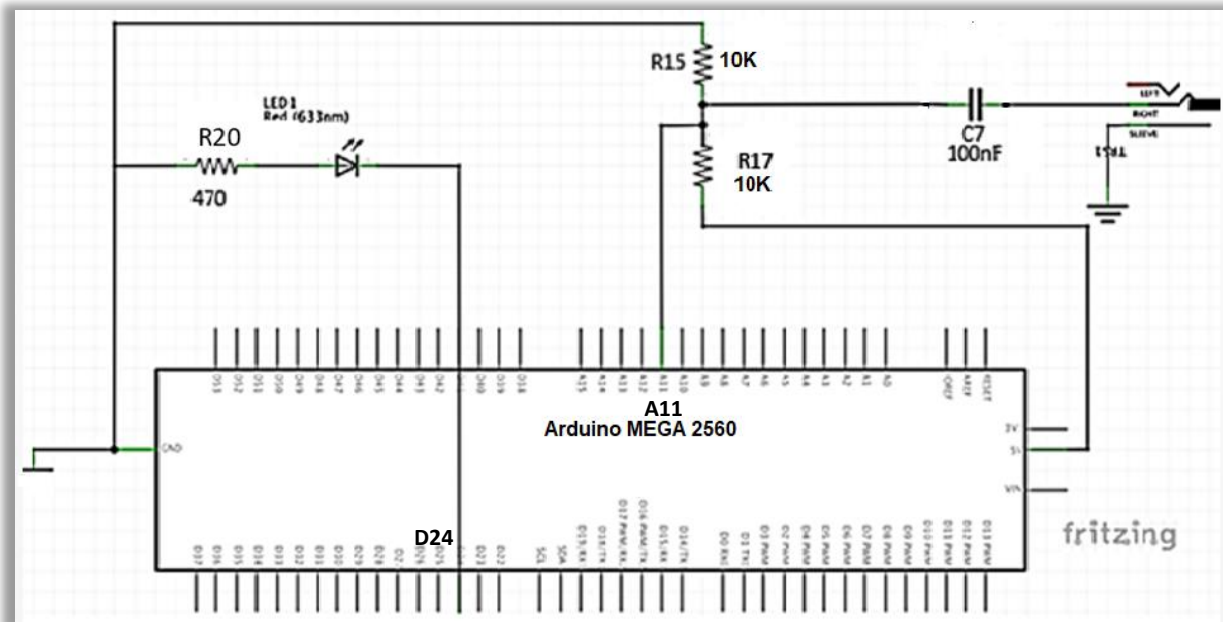
#define GOERTZ_NOISE_BLANKER_INITIAL_MS 6
#define GOERTZ_TARGET_FREQ 558.0

#define GOERTZ_MAGNITUDE_LIMIT_LOW 100
#define GOERTZ_MAGNITUDE_THRESHOLD0.6 //0.6
#define GOERTZ_MOVING_AVERAGE_FILTER 6
```

Figuur 57, Goertz instellingen

In het *keyer\_pin\_settings.h* bestand moet de pinsetting aangepast worden. Je kiest voor pin A11, en voor *cw decoder indicator* pin 24, fig.58, 59 en 60.

Als laatste in het *keyer\_features\_and\_options.h* de option voor *audio detector* en *cw\_decoder* activeren, fig.61.



Figuur 58, aanpassen voor de MEGA

Kies voor *define cw\_decoder\_pin* het getal "nul".

*keyer\_pin\_settings.h*

```
#ifndef FEATURE_CW_DECODER
#define cw_decoder_pin 0
#ifdef OPTION_CW_DECODER_GOERTZEL_AUDIO_DETECTOR
#define cw_decoder_audio_input_pin A11 // this must be an analog pin!
#endif //OPTION_CW_DECODER_GOERTZEL_AUDIO_DETECTOR
#define cw_decoder_indicator 24 //ledje
#endif //FEATURE_CW_DECODER
```

Figuur 59, pinsetting voor de decoder.

*keyer\_features\_and\_options.h*

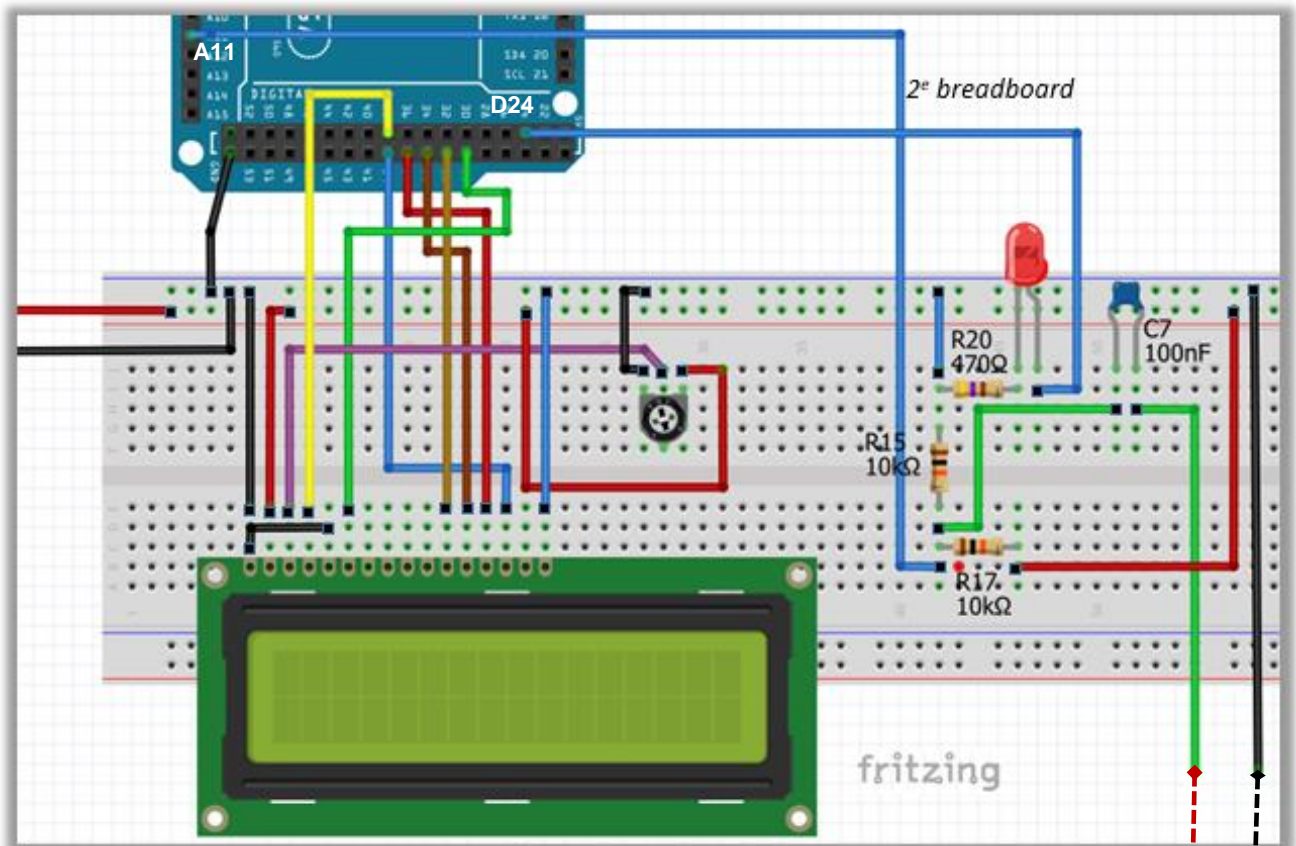
```
// #define OPTION_CW_KEYBOARD_ITALIAN
// #define OPTION_CW_KEYBOARD_GERMAN
#define OPTION_CW_DECODER_GOERTZEL_AUDIO_DETECTOR
// #define OPTION_INVERT_PADDLE_PIN_LOGIC
```

Figuur 60, Goertzel Audio detector

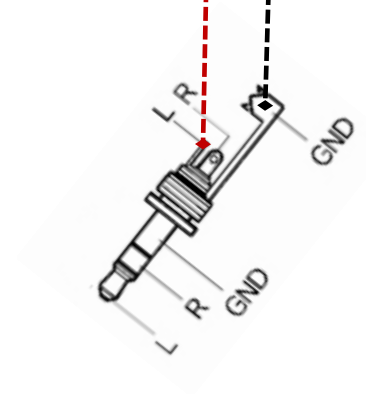
```
// #define FEATURE_LCD_DISPLAY
#define FEATURE_CW_DECODER
// #define FEATURE_SLEEP
```

Figuur 61, feature\_cw\_decoder

NB. Je kunt deze schakeling naast de display schakeling, op je 2<sup>e</sup> breadboard kwijt. Het ledje dient ervoor om de afstemming op het morsesignaal makkelijker te maken en die sluit je aan op A11, fig. 58 en 62.



Figuur 62, decodeer opstelling, 2<sup>e</sup> breadboard



NB:

Bij veel figuren zul je de naam Fritzing tegenkomen en je zult je misschien afvragen wat daarvan de betekenis is.

Fritzing is de naam van een programma



waarmee je schakelingen en breadboard opstellingen mee ontwerpt.

Als je er meer van wilt weten, kun je het (freeware) programma downloaden van de website:

<http://fritzing.org/download/>



## 9. HOE WERKT DE DECODER

Stem je ontvanger af op het CW gedeelte van de HF band. Zorg ervoor dat je de LED hebt aangesloten tussen de massa en de pin die je hebt gekozen.

Stem af op een station tot de LED begint te knipperen bij het ontvangen van CW. Je moet goed afstemmen om het leesbaar te krijgen, zodat de LED ziet dat de juiste gegevens worden verzonden.

Speel een beetje spelen met de bovenstaande instellingen in het *goertzel.h* bestand om te zien of je een betere decoding kan krijgen. De bandbreedte instelling van de transceiver is (CW mode) 1000Hz, de decoding is bij mij dan optimaal.

NB.

Eerlijkheidshalve moet ik zeggen dat je niet al te veel moet verwachten van de decoding. De decodeermethode is nog in een experimentele fase. Voor de beginnende CW operator is de boven vermelde decoding op de display toch erg leuk om te volgen en dan voornamelijk als er sprake is van een "net" regelmatig seinschrift niet verstoord door nabij gelegen CW signalen.

Mogelijk kun je ook proberen op bakenstations af te stemmen, die een mooi regelmatig seinschrift uitzenden.

Verdere informatie over de bediening en aanvullende informatie van de CW-keyer kun je vinden op de site:

<https://blog.radioartisan.com/arduino-cw-keyer/>

Onderstaande groep is heel actief en hulpvaardig, je kunt er al je vragen kwijt over de K3NG-CW keyer en andere K3NG projecten.

<https://groups.yahoo.com/neo/groups/radioartisan/info>

Nuttige aanvullende sites:

<https://www.arduino.cc/en/Main/Software>

[https://github.com/k3ng/k3ng\\_cw\\_keyer](https://github.com/k3ng/k3ng_cw_keyer)

<https://learn.sparkfun.com/tutorials/how-to-read-a-schematic>

<https://www.arduino.cc/en/Main/Software>

<http://fritzing.org/home/>

<https://makerzone.mathworks.com/resources/getting-started-with-arduino-mega-2560-hardware/>

<http://www.skovholm.com/cwdecoder>

<http://www.justradios.com/uFnFpF.html>

## 10. Toetsenbord opdrachten

### Speciale PS2 Toetsenbord opdrachten

F1 through F12-----	play memories 1 through 12
Up Arrow-----	Increase CW Speed 1 WPM
Down Arrow-----	Decrease CW Speed 1 WPM
Page Up-----	Increase sidetone frequency
Page Down-----	Decrease sidetone frequency
Right Arrow-----	Dah to Dit Ratio increase
Left Arrow-----	Dah to Dit Ratio decrease
Home-----	reset Dah to Dit Ratio to default
Tab-----	pause sending
Delete-----	delete the last character in the buffer
Esc-----	stop sending and clear the buffer
Scroll Lock -----	Merge the next two characters to form a prosign
Shift-----	Scroll Lock – toggle PTT line
CTRL-A-----	Iambic A Mode
CTRL-B-----	Iambic B Mode
CTRL-C-----	Single Paddle Mode
CTRL-D-----	Ultimatic Mode
CTRL-E-----	Set Serial Number
CTRL-G-----	Bug Mode
CTRL-H-----	Hellschreiber Mode (requires FEATURE_HELL)
CTRL-I-----	TX Line Disable/Enable
CTRL-M-----	Set Farnsworth Speed (requires FEATURE_FARNSWORTH)
CTRL-N-----	Paddle Revers
CTRL-O-----	Sidetone On/Off
CTRL-T -----	Tune
CTRL-U-----	PTT Manual On/Off
CTRL-W-----	Set WPM
CTRL-Z-----	Autospace On/Off
SHIFT-F1, F2, F3...	Program memory 1, 2, 3... (programmeren van de geheugen toetsen)
ALT-F1, F2, F3...	Repeat memory 1, 2, 3...
CTRL-F1, F2, F3...	Switch to transmitter 1, 2, 3...

## 11. Componenten lijstje

Benodigde onderdelen:

onderdeel	benaming, waarden	aantal
	Arduino MEGA, 2560	1x
	Breadboard	2 x
	Toetsenbord PS/2	1x
S 1,2,3,4,5,6	Aan/uit drukknopjes, 6x6 mm.	6 x
	verbindingsdraadjes	veel
Led 1, 2	Rood (633nm)	2x
R1,11,15,17	10 k $\Omega$	4x
R2,3,4,5,6	1 k $\Omega$	5x
R19, 20	470 $\Omega$	2x
R7,10	10 k $\Omega$ , pot. meter	2x
R 8,9,11,12,13,16	100 $\Omega$	6x
C1,2,4,5,6	0.01 $\mu$ F (10nF)	5x
C3,C7	0.1 $\mu$ F (100nF)	2x
Q 1,2,3,4	2N2222A (NPN)	4x

## 12. Bijlagen

### Keyer\_pin\_settings.h

```

/* Pins - you must review these and configure ! */
#ifndef keyer_pin_settings_h
#define keyer_pin_settings_h

#define paddle_left 42
#define paddle_right 44
#define tx_key_line_1 11 // (high = key down/tx on)
#define tx_key_line_2 12
#define tx_key_line_3 0
#define tx_key_line_4 0
#define tx_key_line_5 0
#define tx_key_line_6 0
#define sidetone_line 48 // connect a speaker for sidetone
#define potentiometer A1 // Speed potentiometer (0 to 5 V) Use pot from 1k to 10k
#define ptt_tx_1 0 // PTT ("push to talk") lines
#define ptt_tx_2 0 // Can be used for keying fox transmitter, T/R switch, or keying slow boatanchors
#define ptt_tx_3 0 // These are optional - set to 0 if unused
#define ptt_tx_4 0
#define ptt_tx_5 0
#define ptt_tx_6 0
#define tx_key_dit 0 // if defined, goes active for dit (any transmitter) - customized with tx_key_dit_and_dah_pins_active_state and tx_key_dit_and_dah_pins_inactive_state
#define tx_key_dah 0 // if defined, goes active for dah (any transmitter) - customized with tx_key_dit_and_dah_pins_active_state and tx_key_dit_and_dah_pins_inactive_state

#ifdef FEATURE_COMMAND_BUTTONS
#define analog_buttons_pin A0
#define command_mode_active_led 28
#endif //FEATURE_COMMAND_BUTTONS

/*
FEATURE_SIDETONE_SWITCH
Enabling this feature and an external toggle switch adds switch control for playing cw sidetone.
ST Switch status is displayed in the status command. This feature will override the software control of the sidetone (\o).
Arduino pin is assigned by SIDETONE_SWITCH
*/

#ifdef FEATURE_SIDETONE_SWITCH
#define SIDETONE_SWITCH 8
#endif //FEATURE_SIDETONE_SWITCH

//lcd pins
#ifdef FEATURE_LCD_4BIT
#define lcd_rs 38
#define lcd_enable 31
#define lcd_d4 33
#define lcd_d5 35
#define lcd_d6 37
#define lcd_d7 39
#endif //FEATURE_LCD_4BIT

```

```
#ifndef FEATURE_LCD1602_N07DH
  #define lcd_rs 8
  #define lcd_enable 9
  #define lcd_d4 4
  #define lcd_d5 5
  #define lcd_d6 6
  #define lcd_d7 7
#endif //FEATURE_LCD1602_N07DH

//ps2 keyboard pins
#ifndef FEATURE_PS2_KEYBOARD
  #define ps2_keyboard_data A3
  #define ps2_keyboard_clock 3 // this must be on an interrupt capable pin!
#endif //FEATURE_PS2_KEYBOARD

// rotary encoder pins and options - rotary encoder code from Jim Balls M0CKE
#ifndef FEATURE_ROTARY_ENCODER
  #define OPTION_ENCODER_HALF_STEP_MODE // Half-step mode?
  #define rotary_pin1 0 // CW Encoder Pin
  #define rotary_pin2 0 // CCW Encoder Pin
  #define OPTION_ENCODER_ENABLE_PULLUPS // define to enable weak pullups.
#endif //FEATURE_ROTARY_ENCODER

#ifndef FEATURE_LED_RING
  #define led_ring_sdi A10 //2 //Data
  #define led_ring_clk A9 //3 //Clock
  #define led_ring_le A8 //4 //Latch
#endif //FEATURE_LED_RING

#ifndef FEATURE_ALPHABET_SEND_PRACTICE
  #define correct_answer_led 0
  #define wrong_answer_led 0
#endif //FEATURE_ALPHABET_SEND_PRACTICE

#ifndef FEATURE_PTT_INTERLOCK
  #define ptt_interlock 0 // this pin disables PTT and TX KEY
#endif //FEATURE_PTT_INTERLOCK

#ifndef FEATURE_STRAIGHT_KEY
  #define pin_straight_key 52
#endif //FEATURE_STRAIGHT_KEY

#ifndef FEATURE_CW_DECODER
  #define cw_decoder_pin 0
  #ifndef OPTION_CW_DECODER_GOERTZEL_AUDIO_DETECTOR
    #define cw_decoder_audio_input_pin A11 // this must be an analog pin!
  #endif //OPTION_CW_DECODER_GOERTZEL_AUDIO_DETECTOR
  #define cw_decoder_indicator 24
#endif //FEATURE_CW_DECODER

#if defined(FEATURE_COMPETITION_COMPRESSION_DETECTION)
  #define compression_detection_pin 13
#endif //FEATURE_COMPETITION_COMPRESSION_DETECTION

#if defined(FEATURE_SLEEP)
```

```
#define keyer_awake 0
#endif

#if defined(FEATURE_CAPACITIVE_PADDLE_PINS)
  #define capacitive_paddle_pin_inhibit_pin 0 // if this pin is defined and is set high, the capacitive paddle pins will switch to
normal (non-capacitive) sensing mode
#endif

#ifdef FEATURE_4x4_KEYPAD
  #define Row3 33
  #define Row2 32
  #define Row1 31
  #define Row0 30
  #define Col3 37
  #define Col2 36
  #define Col1 35
  #define Col0 34
#endif

#ifdef FEATURE_3x4_KEYPAD
  #define Row3 33
  #define Row2 32
  #define Row1 31
  #define Row0 30
  #define Col2 36
  #define Col1 35
  #define Col0 34
#endif

#else

  #error "Multiple pin_settings.h files included somehow..."

#endif //keyer_pin_settings_h
```

**keyer\_features\_and\_options.h**

```

// compile time features and options - comment or uncomment to add or delete features
// FEATURES add more bytes to the compiled binary, OPTIONS change code behavior

#define FEATURE_COMMAND_BUTTONS
// #define FEATURE_COMMAND_LINE_INTERFACE // Command Line Interface functionality
#define FEATURE_MEMORIES // on the Arduino Due, you must have FEATURE_EEPROM_E24C1024 and
E24C1024 EEPROM hardware in order to compile this
#define FEATURE_MEMORY_MACROS
// #define FEATURE_WINKEY_EMULATION // disabling Automatic Software Reset is highly recommended (see documen-
tation)
// #define FEATURE_BEACON
// #define FEATURE_TRAINING_COMMAND_LINE_INTERFACE
#define FEATURE_POTENTIOMETER // do not enable unless you have a potentiometer connected, otherwise noise
will falsely trigger wpm changes
// #define FEATURE_SIDETONE_SWITCH // adds switch control for the sidetone output. requires an external toggle switch
(assigned to an arduino pin - see keyer_pin_settings.h).
// #define FEATURE_SERIAL_HELP
// #define FEATURE_HELL
#define FEATURE_PS2_KEYBOARD // Use a PS2 keyboard to send code - Change keyboard layout (non-US) in
K3NG_PS2Keyboard.h. Additional options below.
// #define FEATURE_USB_KEYBOARD // Use a USB keyboard to send code - Uncomment three lines in
k3ng_keyer.ino (search for note_usb_uncomment_lines)
// #define FEATURE_CW_COMPUTER_KEYBOARD // Have an Arduino Due or Leonardo act as a USB HID (Human Inter-
face Device) keyboard and use the paddle to "type" characters on the computer -- uncomment this line in ino file: #include
<Keyboard.h>
// #define FEATURE_DEAD_OP_WATCHDOG
// #define FEATURE_AUTOSPACE
// #define FEATURE_FARNSWORTH
// #define FEATURE_DL2SBA_BANKSWITCH // Switch memory banks feature as described here: http://dl2sba.com/in-
dex.php?option=com\_content&view=article&id=131:nanokeyer&catid=15:shack&Itemid=27#english
#define FEATURE_LCD_4BIT // classic LCD disidefplay using 4 I/O lines
// #define FEATURE_LCD_ADAFRUIT_I2C // Adafruit I2C LCD display using MCP23017 at addr 0x20
// #define FEATURE_LCD_ADAFRUIT_BACKPACK // Adafruit I2C LCD Backup using MCP23008 (courtesy Josiah Rit-
chie, KE0BLL)
// #define FEATURE_LCD_YDv1 // YourDuino I2C LCD display with old LCM 1602 V1 ic
// #define FEATURE_LCD1602_N07DH // http://linksprite.com/wiki/index.php5?title=16\_X\_2\_LCD\_Key-
pad\_Shield\_for\_Arduino
// #define FEATURE_LCD_SAINSMART_I2C
#define FEATURE_CW_DECODER
// #define FEATURE_SLEEP // go to sleep after x minutes to conserve battery power (not compatible with Ar-
duino DUE, may have mixed results with Mega and Mega ADK)
#define FEATURE_ROTARY_ENCODER // rotary encoder speed control
// #define FEATURE_CMOS_SUPER_KEYER_IAMBIC_B_TIMING
// #define FEATURE_HI_PRECISION_LOOP_TIMING
// #define FEATURE_USB_MOUSE // Uncomment three lines in k3ng_keyer.ino (search for note_usb_uncom-
ment_lines)
// #define FEATURE_CAPACITIVE_PADDLE_PINS // remove the bypass capacitors on the paddle_left and paddle_right
lines when using capactive paddles
// #define FEATURE_LED_RING // Mayhew Labs Led Ring support
// #define FEATURE_ALPHABET_SEND_PRACTICE // enables command mode S command - created by Ryan, KC2ZWM
// #define FEATURE_PTT_INTERLOCK
// #define FEATURE_QLF
// #define FEATURE_EEPROM_E24C1024

```

```
// #define FEATURE_STRAIGHT_KEY
// #define FEATURE_DYNAMIC_DAH_TO_DIT_RATIO
// #define FEATURE_PADDLE_ECHO
// #define FEATURE_STRAIGHT_KEY_ECHO
// #define FEATURE_AMERICAN_MORSE
// #define FEATURE_4x4_KEYPAD // code contributed by Jack, W0XR - documentation:
// https://github.com/k3ng/k3ng_cw_keyer/wiki/380-Feature:-Keypad
// #define FEATURE_3x4_KEYPAD // code contributed by Jack, W0XR - documentation:
// https://github.com/k3ng/k3ng_cw_keyer/wiki/380-Feature:-Keypad

// #define FEATURE_COMMAND_LINE_INTERFACE_ON_SECONDARY_PORT // Activate the Command Line interface
// on the secondary serial port
#define OPTION_PRIMARY_SERIAL_PORT_DEFAULT_WINKEY_EMULATION // Use when activating both FEAT-
// URE_WINKEY_EMULATION and FEATURE_COMMAND_LINE_INTERFACE
// // simultaneously. This will make Winkey emulation be the default at boot up;
// // hold command button down at boot up to activate CLI mode

// #define OPTION_SUPPRESS_SERIAL_BOOT_MSG
#define OPTION_INCLUDE_PTT_TAIL_FOR_MANUAL_SENDING
#define OPTION_EXCLUDE_PTT_HANG_TIME_FOR_MANUAL_SENDING
// #define OPTION_WINKEY_DISCARD_BYTES_AT_STARTUP // if ASR is not disabled, you may need this to discard
// errant serial port bytes at startup
// #define OPTION_WINKEY_STRICT_EEPROM_WRITES_MAY_WEAR_OUT_EEPROM // with this activated the unit will
// write non-volatile settings to EEPROM when set by Winkey commands
// #define OPTION_WINKEY_SEND_WORDSAPCE_AT_END_OF_BUFFER
#define OPTION_WINKEY_STRICT_HOST_OPEN // require an admin host open Winkey command before doing
// any other commands
#define OPTION_WINKEY_2_SUPPORT // comment out to revert to Winkey version 1 emulation
#define OPTION_WINKEY_INTERRUPTS_MEMORY_REPEAT
// #define OPTION_WINKEY_UCXLOG_9600_BAUD // use this only with UCXLog configured for Winkey 9600 baud
// mode
// #define OPTION_WINKEY_2_HOST_CLOSE_NO_SERIAL_PORT_RESET // activate this when using Winkey 2 emula-
// tion and Win-Test
// #define OPTION_WINKEY_FREQUENT_STATUS_REPORT // activate this to make Winkey emulation play better
// with RUMlog and RUMped
#define OPTION_WINKEY_IGNORE_LOWERCASE // Enable for typical K1EL Winkeyer behavior (use for Skook-
// umLogger version 1.10.14 and prior to workaround "r" bug)
// #define OPTION_REVERSE_BUTTON_ORDER // This is mainly for the DJ0MY NanoKeyer http://nano-
// keyer.wordpress.com/
#define OPTION_PROG_MEM_TRIM_TRAILING_SPACES // trim trailing spaces from memory when programming in
// command mode
#define OPTION_DIT_PADDLE_NO_SEND_ON_MEM_RPT // this makes dit paddle memory interruption a little
// smoother
// #define OPTION_MORE_DISPLAY_MSGS // additional optional display messages - comment out to save
// memory
// #define OPTION_N1MM_WINKEY_TAB_BUG_WORKAROUND // enable this to ignore the TAB key in the Send CW
// window (this breaks SO2R functionality in N1MM)
// #define OPTION_WATCHDOG_TIMER // this enables a four second ATmega48/88/168/328 watchdog ti-
// mer; use for unattended/remote operation only
// #define OPTION_MOUSE_MOVEMENT_PADDLE // experimental (just fooling around) - mouse movement will
// act like a paddle
// #define OPTION_NON_ENGLISH_EXTENSIONS // add support for additional CW characters (i.e. Å, Æ, P, etc.)
// #define OPTION_KEEP_PTT_KEYED_WHEN_CHARS_BUFFERED // this option keeps PTT high if there are characters
// buffered from the keyboard, the serial interface, or Winkey
```



```
// #define OPTION_DISPLAY_NON_ENGLISH_EXTENSIONS // LCD display suport for non-English (NO/DK/DE) characters
- Courtesy of OZ1JHM
// #define OPTION_UNKNOWN_CHARACTER_ERROR_TONE
// #define OPTION_DO_NOT_SAY_HI
// #define OPTION_PS2_NON_ENGLISH_CHAR_LCD_DISPLAY_SUPPORT // makes some non-English characters from
the PS2 keyboard display correctly in the LCD display (donated by Marcin sp5iou)
// #define OPTION_PS2_KEYBOARD_RESET // reset the PS2 keyboard upon startup with 0xFF (contributed by Bill,
W9BEL)
// #define OPTION_SAVE_MEMORY_NANOKEYER
#define OPTION_CW_KEYBOARD_CAPSLOCK_BEEP
// #define OPTION_CW_KEYBOARD_ITALIAN
// #define OPTION_CW_KEYBOARD_GERMAN
#define OPTION_CW_DECODER_GOERTZEL_AUDIO_DETECTOR
// #define OPTION_INVERT_PADDLE_PIN_LOGIC
// #define OPTION_ADVANCED_SPEED_DISPLAY //enables "nerd" speed visualization on display: wpm, cpm (char per
min), duration of dit and dah in milliseconds and ratio (contributed by Giorgio, IZ2XBZ)
// #define OPTION_PROSIGN_SUPPORT // additional prosign support for paddle and straight key echo on display, CLI,
and in memory storage
// #define OPTION_RUSSIAN_LANGUAGE_SEND_CLI // Russian language CLI sending support (contributed by Павел
Бирюков, UA1AQC)
#define OPTION_DO_NOT_SEND_UNKNOWN_CHAR_QUESTION
// #define OPTION_CMOS_SUPER_KEYER_IAMBIC_B_TIMING_ON_BY_DEFAULT
// #define OPTION_SIDETONE_DIGITAL_OUTPUT_NO_SQUARE_WAVE

// #define OPTION_WORDSWORTH_CZECH
// #define OPTION_WORDSWORTH_DEUTSCH
// #define OPTION_WORDSWORTH_NORSK
```