Vereniging voor Experimenteel Radio Onderzoek in Nederland

CTCSS Encoder

“Super Flexibel Ontwerpen met een Arduino Nano”

(voor minder als 2 euro weer QRV op de repeater)

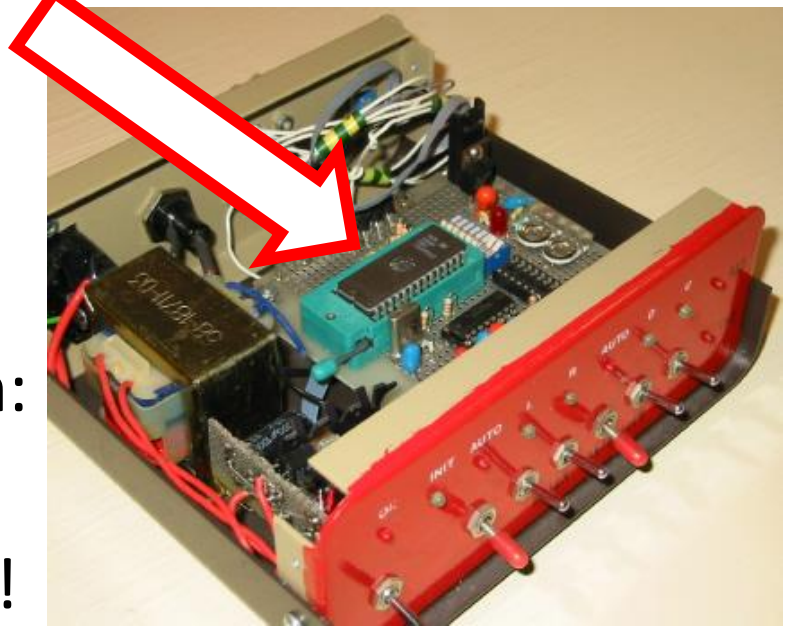
Henk Hamoen (PA3GUO)
March 2016

Thanks to Bart Wijsman (PA2BB) for corrections in the schematics

Agenda

- Introductie
- CTCSS toonslot op repeaters
- Arduino Nano
- CTCSS Tone opwekken
- Audio Uitgangs Filters
- Flexibel ontwerp
- Conclusies
- Vragen

Introductie



- Microcontroller in 20 jaar oude rotorbesturing vervangen: ST6 (RS232)→ Arduino (USB)
- Nòg eenvoudiger als verwacht !

- 20 jaar oude zelfbouw 145MHz zendontvanger
- Onbruikbaar sinds CTCSS

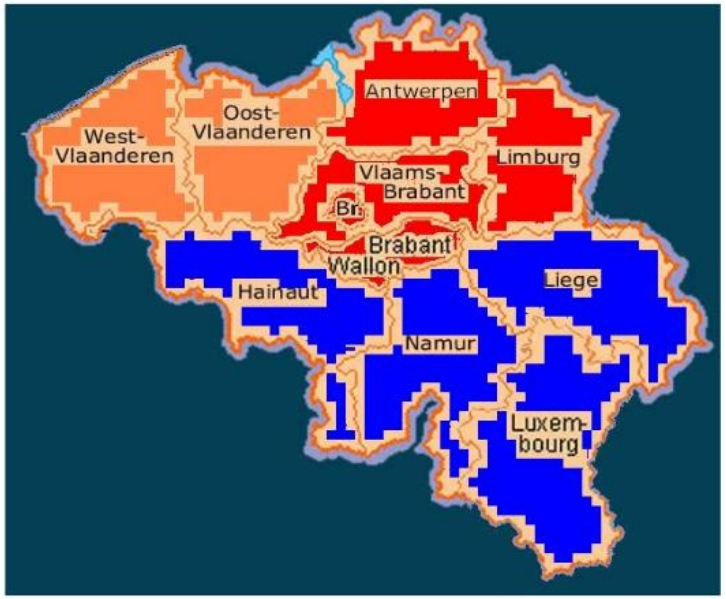
Kan ik dit ook met een Arduino oplossen ?

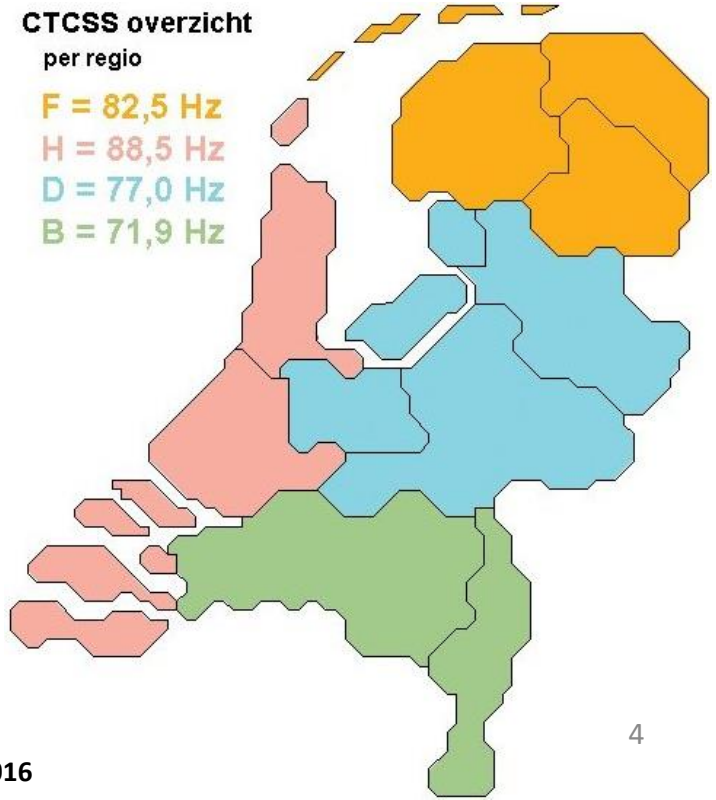
CTCSS toonslot op repeaters

- Eindhoven: 1 toon mee uitzenden (71.9 Hz)
- Nederland: 4 regios (B:71.9, D:77.0, F: 82.5, H: 88.5)
- Belgie: 3 regios (A (T): 131.8, C (C): 74.4, E (E): 79.7)
- Duitsland: beperkt (# (O):110.9, x (R):123.0, % (A):67.0)

Zone 1: A 131.8 Hz Antwerpen, Limburg, Vlaams-Brabant, Waals-Brabant en het Brussels gewest. (1)
Zone 2: C 74.4 Hz Henegouwen, Namen, Luik en Luxemburg.
Zone 3: E 79.7 Hz West-Vlaanderen en Oost-Vlaanderen.

(1): *Sommige repeaters gebruiken nog de CTCSS-frequentie 67.0 Hz.*





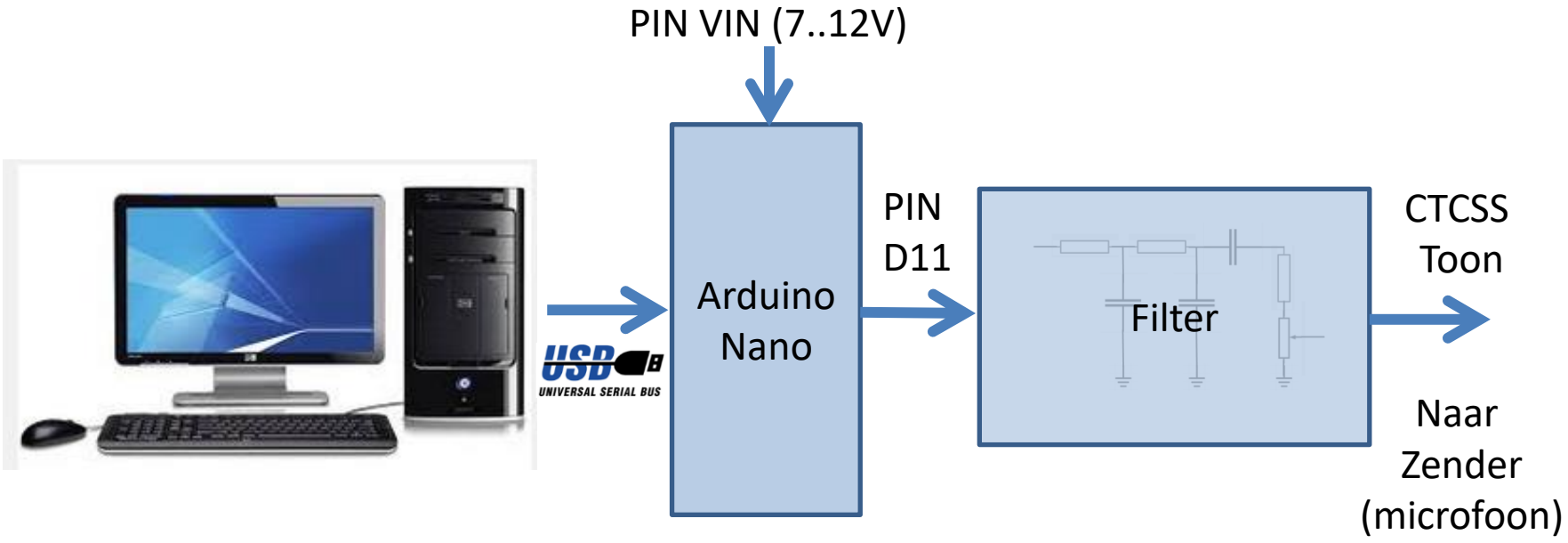
Arduino Nano

- Kosten: 1,80 euro (www.AliExpress.com)
- Afmeting: 1.8 x 4.5 cm
- Digitale IO: 11 (14)
- Analoge IO: 8

- Voeding: via USB *òf* 5V *òf* 7..12V
- Eenvoudig te (her-)programmeren

CTCSS toon opwekken

- SW: Arduino libraries: CTCSS (credits: Jan PE1CID)
- HW: Arduino + uitgangsfilter
- PC: Eénmalig programmeren

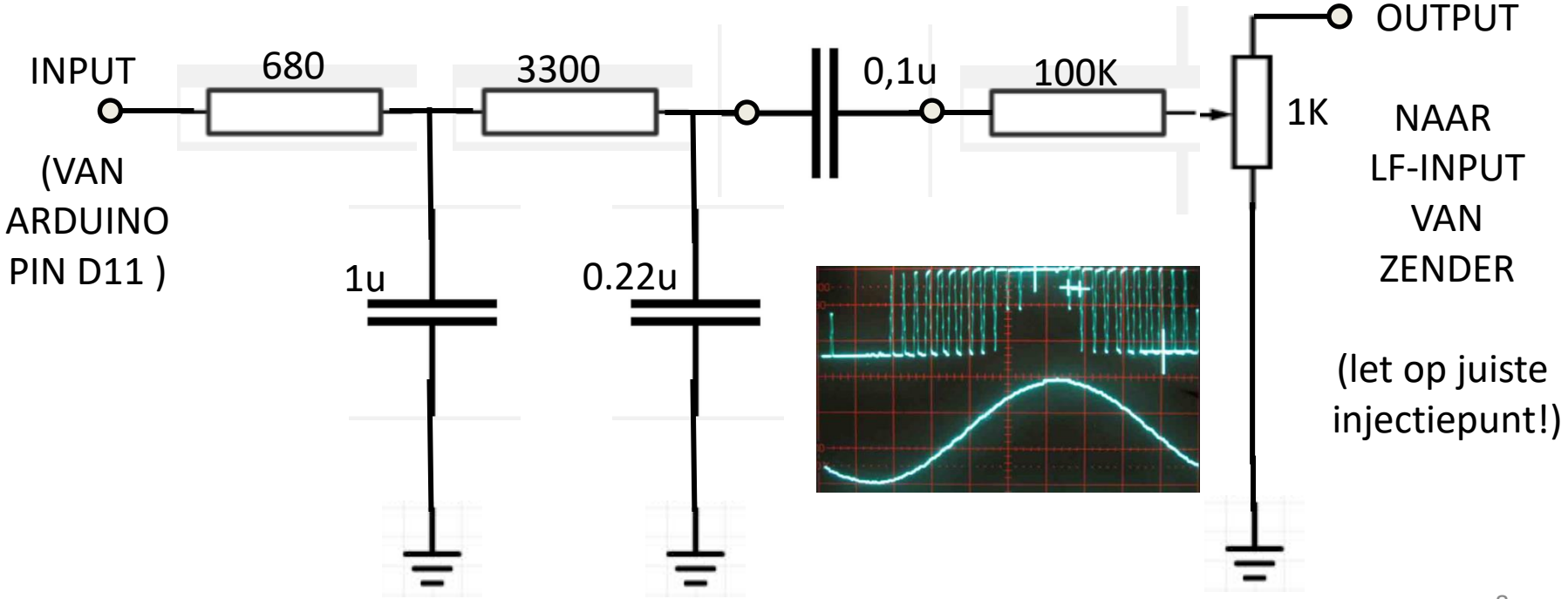


CTCSS toon opwekken: SW

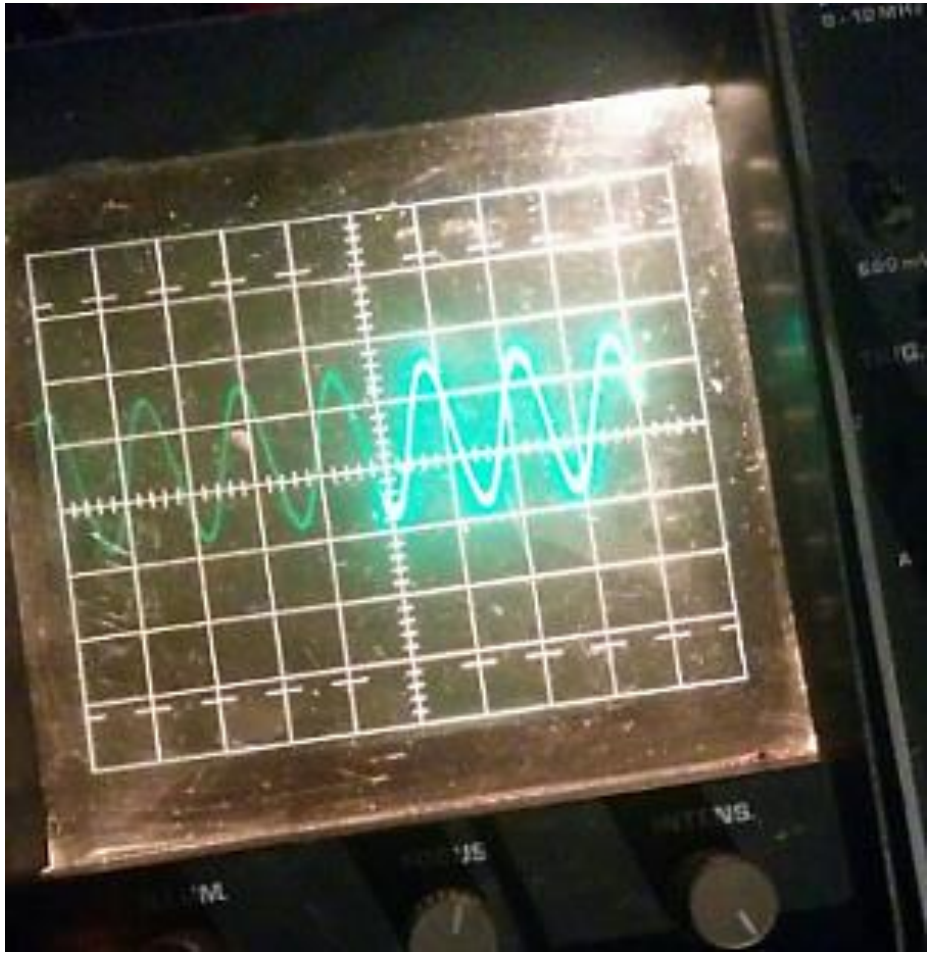
```
#include <CtcssToneId.h>
#include <CtcssTone.h>

CtcssTone.init();

CtcssTone.tone_on(tone_ctcss_B);  // start tone B (71.9 Hz)
CtcssTone.tone_off();             // stop tone
```

Audio Uitgangs Filters

- Laagdoorlaat filters (credits: Joris PE1GLX)
  2 x RC: -3dB @234 Hz + -3dB@ 202 Hz (geen dure spoel !)
- DC ontkoppeling (C)
- Uitgangs nivo-regeling (instelbare R)

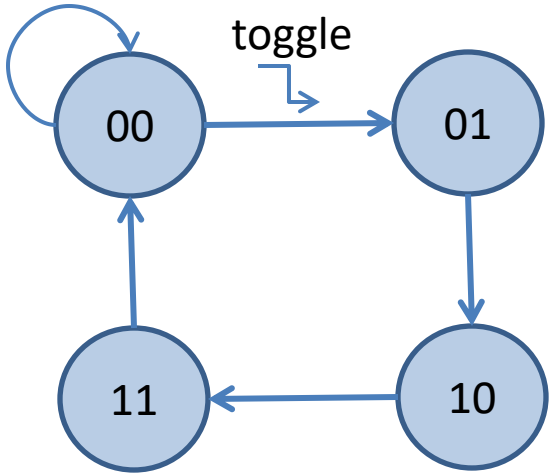


Resultaat: 71,9 Hz (Zuid NL)



MEER! 4 tonen voor heel NL

- Finite State Machine (FSM) nodig:
  - **4 states** (elke ‘state’ geeft een andere CTCSS tone)
  - 2 schakelaars bepalen in welke ‘state’ **opgestart** wordt
  - **‘toggle’** van state naar state met één druk-schakelaar
  - 2 leds geven de **actuele ‘state’** aan



4 tonen voor heel NL (SW)

Elke state geeft een andere CTCSS toon:

```
if      (state == state_1) {CtcssTone.tone_on(tone_ctcss_B);}
else if (state == state_2) {CtcssTone.tone_on(tone_ctcss_D);}
else if (state == state_3) {CtcssTone.tone_on(tone_ctcss_F);}
else if (state == state_4) {CtcssTone.tone_on(tone_ctcss_H);}
```

4 tonen voor heel NL (SW)

Opstarten in een bepaalde state (bepaald door stand schakelaars):

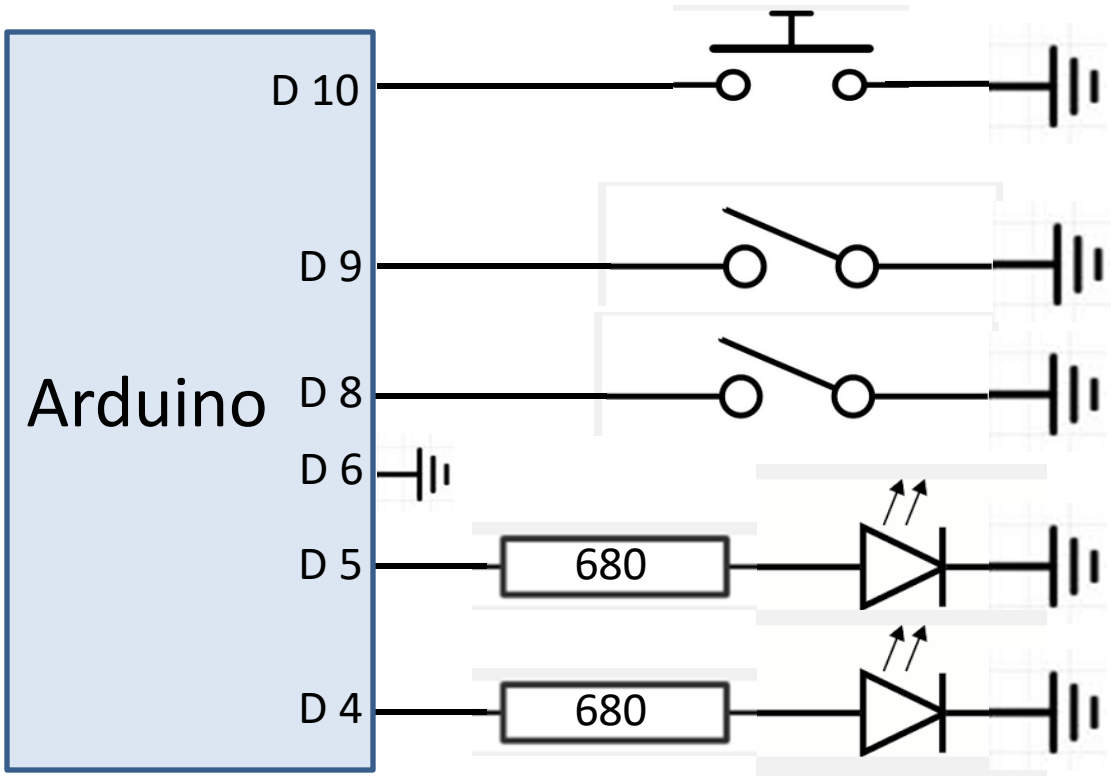
```
val_1 = digitalRead(switch_1); // LSB
val_2 = digitalRead(switch_2); // MSB

if      (val_2 == 0 && val_1 == 0) {state = state_1;}
else if (val_2 == 0 && val_1 == 1) {state = state_2;}
else if (val_2 == 1 && val_1 == 0) {state = state_3;}
else                               {state = state_4;}
```

4 tonen voor heel NL (SW)

Togglen van state naar state (met druk schakelaar):

```
// check if switch is pushed to go to next state
// -----------------------------------------
   val_t = digitalRead(toggle_switch);
   if (val_t == 0)
     {
           if (state == state_1) {state = state_2;}
      else if (state == state_2) {state = state_3;}
      else if (state == state_3) {state = state_4;}
      else                       {state = state_1;}
      delay(300); //  ← toggle speed
     }
```

4 tonen voor heel NL (HW)



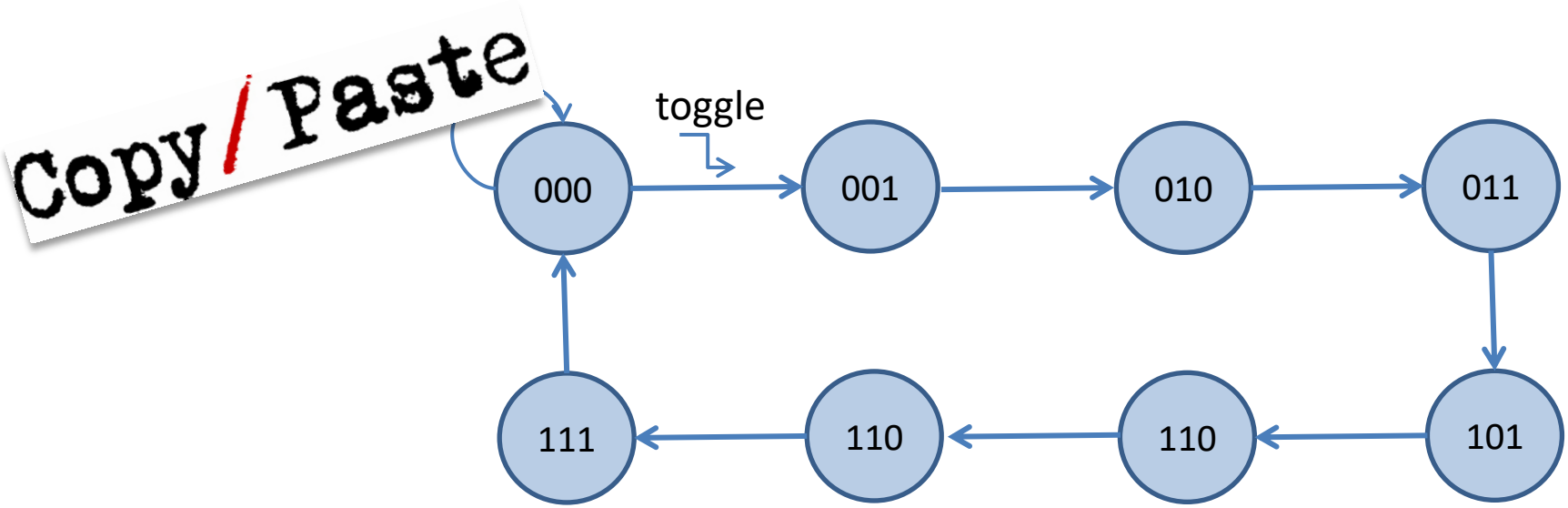
Toggle state

Opstart state
(00, 01, 10, 11)

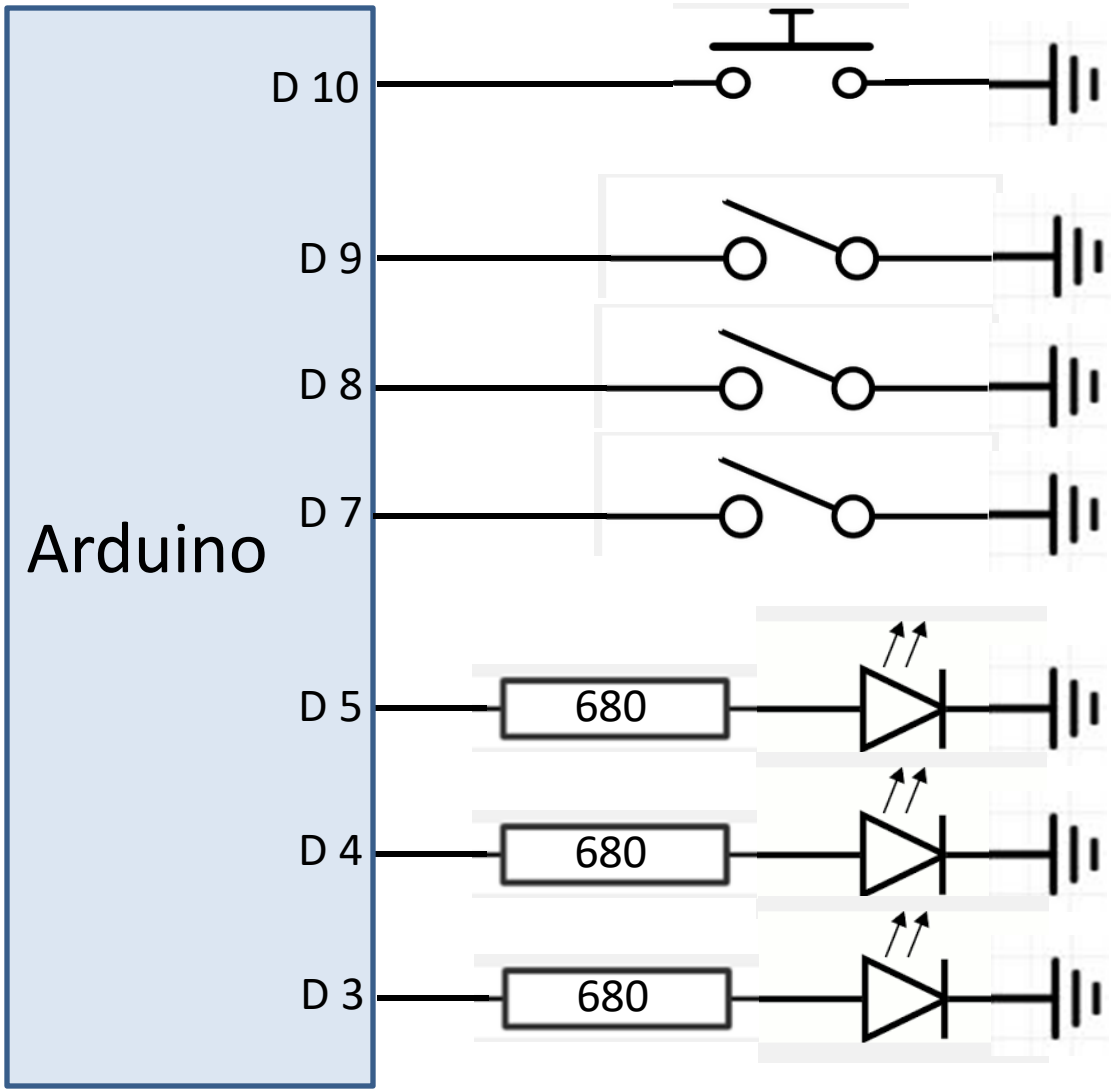
Actuele state
(00, 01, 10, 11)

MEER: 8 tonen !

- Finite State Machine (FSM) uitbreiding nodig:
  - **4 → 8 states** (4 Nederland, 2 België, 2 Duitsland)
  - 000, 001, 010, 011, 100, 101, 110, 111 (**3 bits**)
  - **2 → 3 schakelaars** bepalen in welke ‘state’ opgestart wordt
  - **2 → 3 leds** voor de actuele ‘state’ (toon)



8 tonen (HW)



Toggle state

Opstart state

(000, 001, 010, 011, 100, 101, 110, 111)

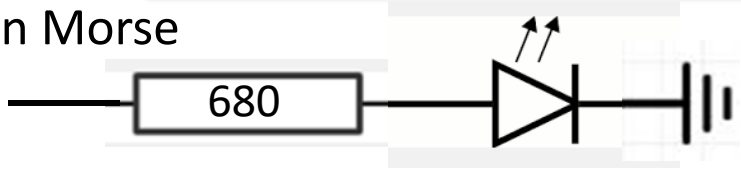
Actuele state

(000, 001, 010, 011, 100, 101, 110, 111)

Flexibel ontwerp

State	Tone	Regio
00	71.9 Hz	B
01	77.0 Hz	D
10	82.5 Hz	F
11	88.5 Hz	H

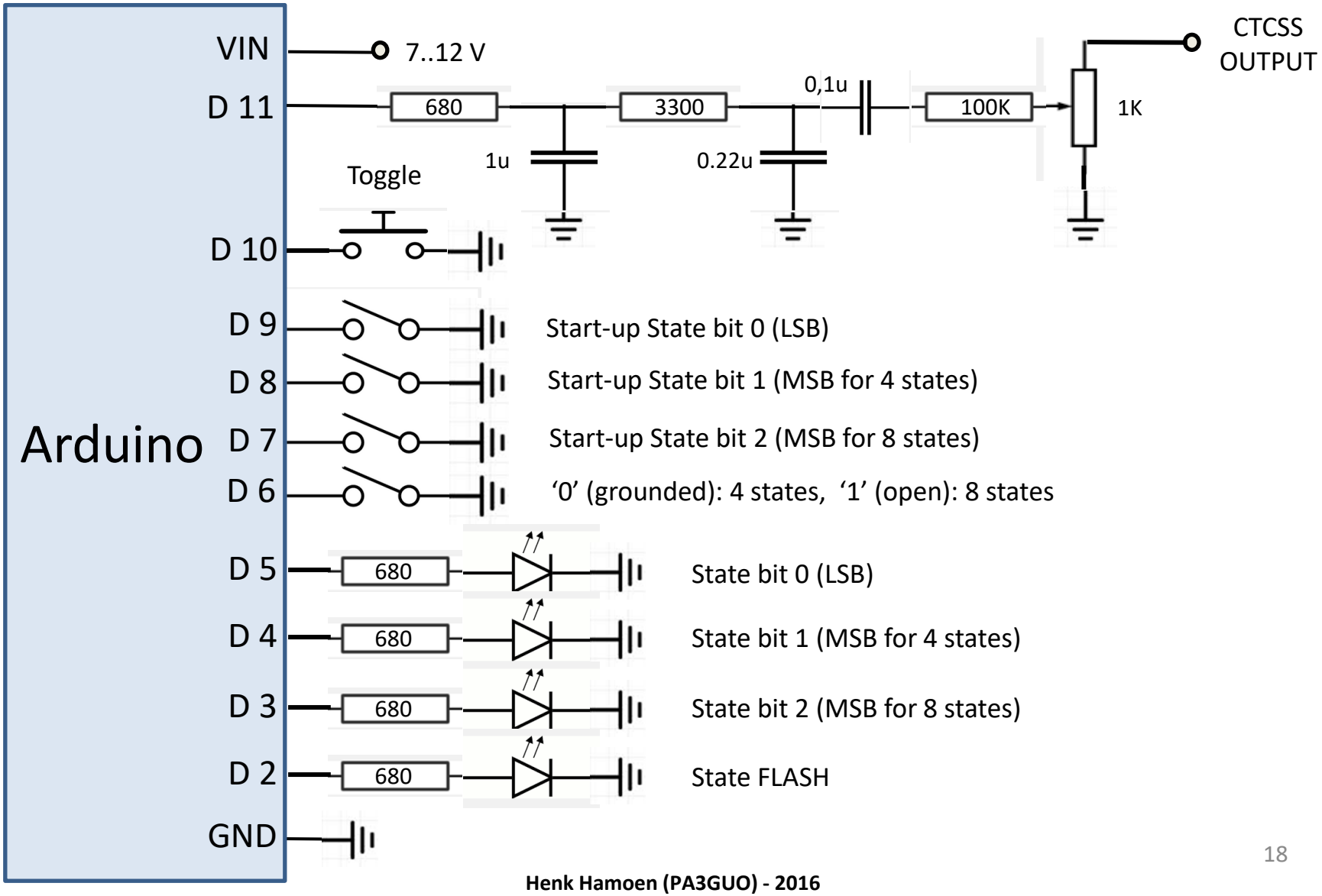
- Hardware
  - Extra: Flash LED die ‘state’ aankondigt in Morse *(1 ipv 3 LEDS)*



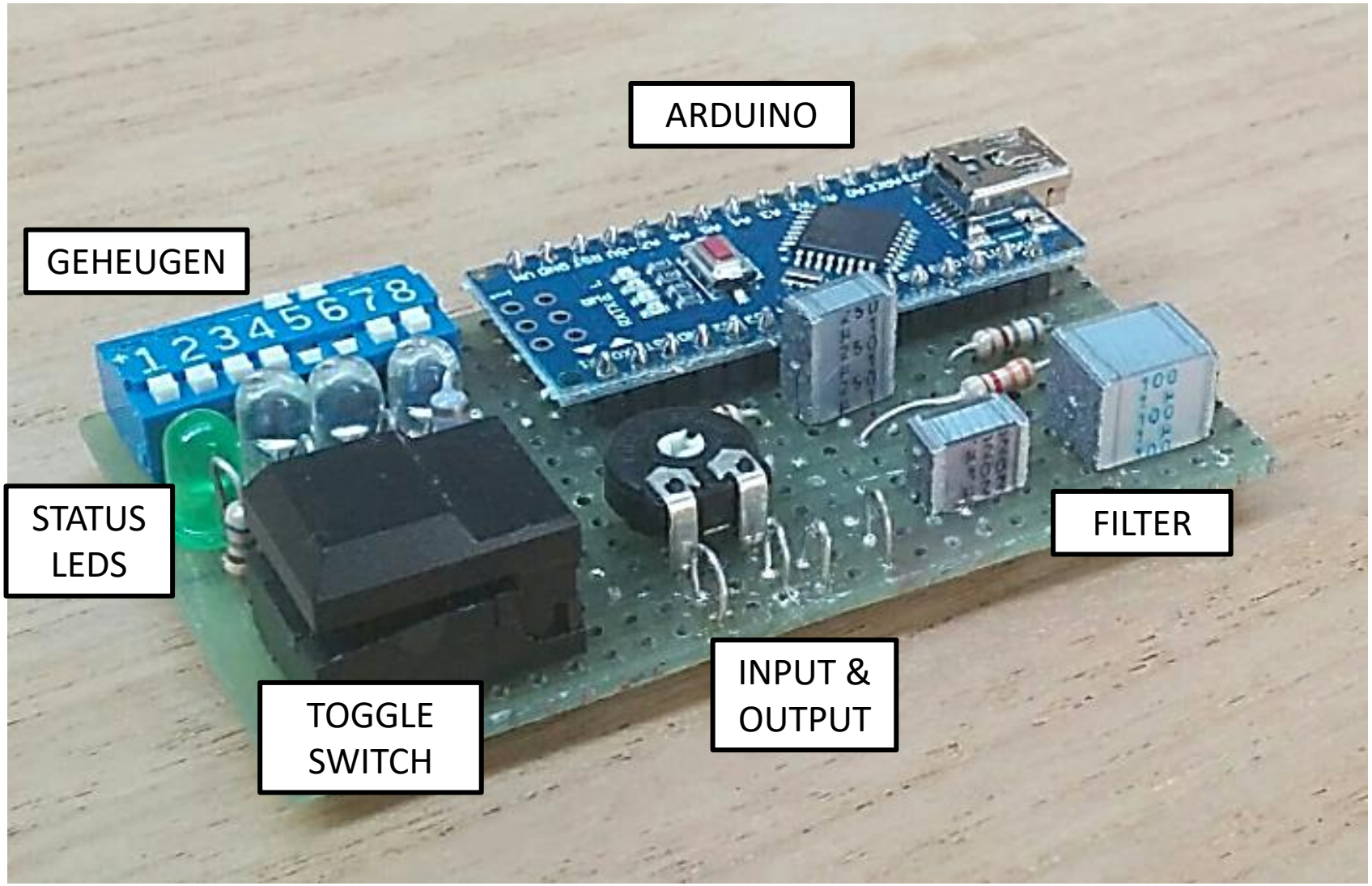
  - Extra: 1 schakelaar: 4 (alleen Nederland) òf 8 tonen (incl. België & DL) *(snellere bediening; niet onnodig schakelen; minder vaak ‘togglen’)*
  - Optie: Géén LEDs en géén toggle schakelaar: één CTCSS toon na opstarten *(kleinst mogelijke configuratie)*

- *Software*
  - *Je kan natuurlijk later altijd nieuwe codes in de Arduino programmeren*
  - *Of: een ander gedrag!*

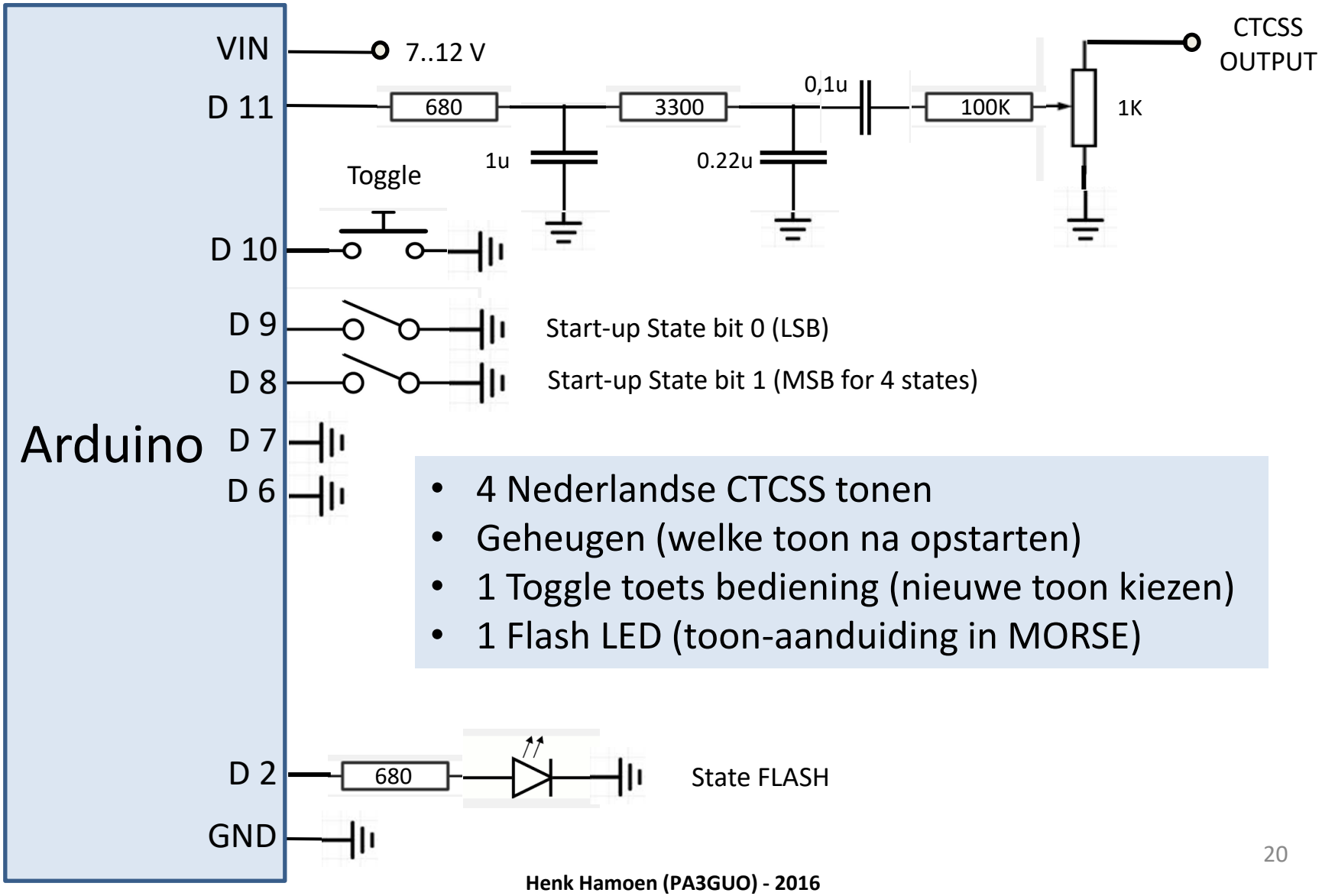
Flexibel ontwerp: alle funkties



Flexibel ontwerp: alle funkties

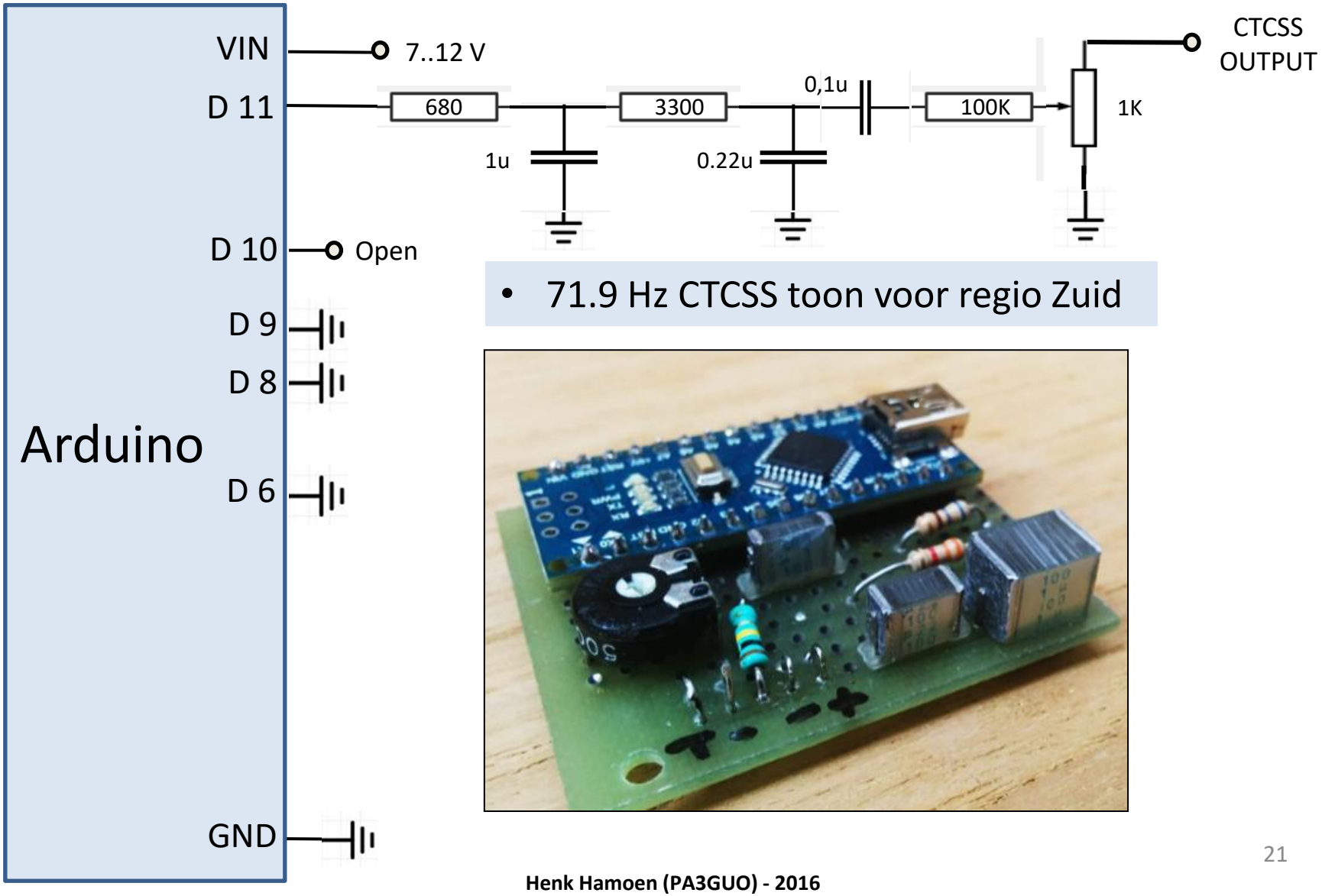


Flexibel ontwerp: NL + Flash



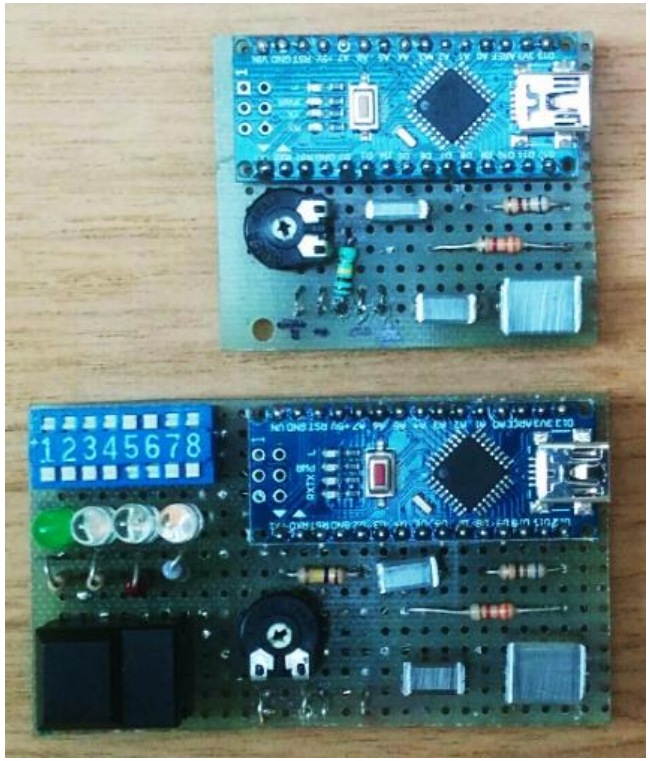
- 4 Nederlandse CTCSS tonen
- Geheugen (welke toon na opstarten)
- 1 Toggle toets bediening (nieuwe toon kiezen)
- 1 Flash LED (toon-aanduiding in MORSE)

Flexibel ontwerp: 1 toon



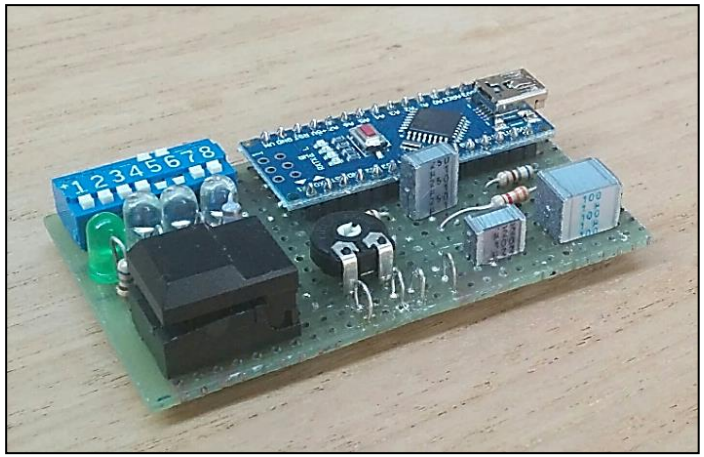
- 71.9 Hz CTCSS toon voor regio Zuid

Conclusies



- **Arduino's zijn goedkoop & klein**
  - *'Component' (net als een R of een IC)*
- **Arduino's hebben veel IO's**
  - *Je kan gratis een hoop 'extra' en/of 'redundante' functies inbouwen*
- **Oude zendontvangers kunnen weer 'online' komen**
  - CTCSS tonen genereren kan voor <2 euro
- **Met een Arduino Nano kan je Super Flexibel Ontwerpen !**

Vragen ?



Vraag 1: Is er een gratis kopie van de software ?

Antwoord: Natuurlijk (zie www.pa3guo.com)

Vraag 2: Kan dit niet veel mooier ?

Antwoord: Jazeker: zelfs met een LCD display etc.

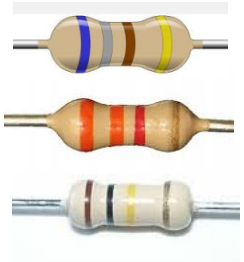
Vraag 3: Kan je ook CTCSS tonen zonder Arduino maken ?

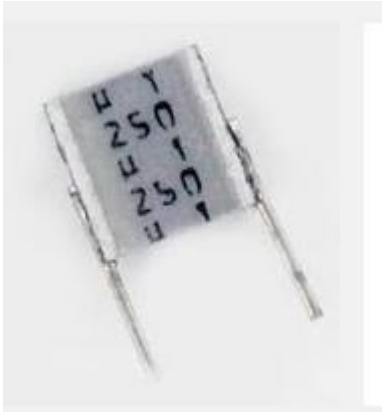
Antwoord: Absoluut (maar dit is wèl de leukste manier) !

BACK UP SLIDES

HW Components

Type	Value	Code
R	680 Ω	blue / grey / brown
R	3300 Ω	orange / orange / red
R	100 KΩ	brown / black / yellow
R	1 KΩ	1K or 102
C	0.1 µF	µ1
C	0.22 µF	µ22
C	1 µF	1 µ





SW Components

Sketchbook

- ***ctcss_encoder_pa3guo***
  - ctcss_encoder_pa3guo.ino
- ***libraries***
  - ***CtcssTone***
    - CtcssToneId.h
    - CtcssTone.h
    - CtcssTone.cpp

4 files

CTCSS Tonen

4 states mode

State	Tone	Regio
00	71.9 Hz	B
01	77.0 Hz	D
10	82.5 Hz	F
11	88.5 Hz	H

8 states mode

State	Tone	Regio
000	71.9 Hz	B
001	77.0 Hz	D
010	82.5 Hz	F
011	88.5 Hz	H
100	79,9 Hz	E
101	131.8 Hz	T
110	110.0 Hz	O
111	123.0 Hz	R

VHF Handbook 2009

CTCSS FREQUENCIES IN Hz TO BE USED FOR REPEATER ACCESS			
71.9 - B	100.0 - L	141.3 - V	203.5 - AF
74.4 - C	103.5 - M	146.2 - W	210.7 - AG
77.0 - D	107.2 - M	151.4 - X	218.1 - AH
79.7 - E	110.9 - O	156.7 - Y	225.7 - AI
82.5 - F	114.8 - P	162.2 - Z	233.6 - AJ
85.4 - G	118.8 - Q	167.9 - AA	241.8 - AK
88.5 - H	123.0 - R	173.8 - AB	250.3 - AL
91.5 - I	127.3 - S	179.9 - AC	
94.8 - J	131.8 - T	186.2 - AD	

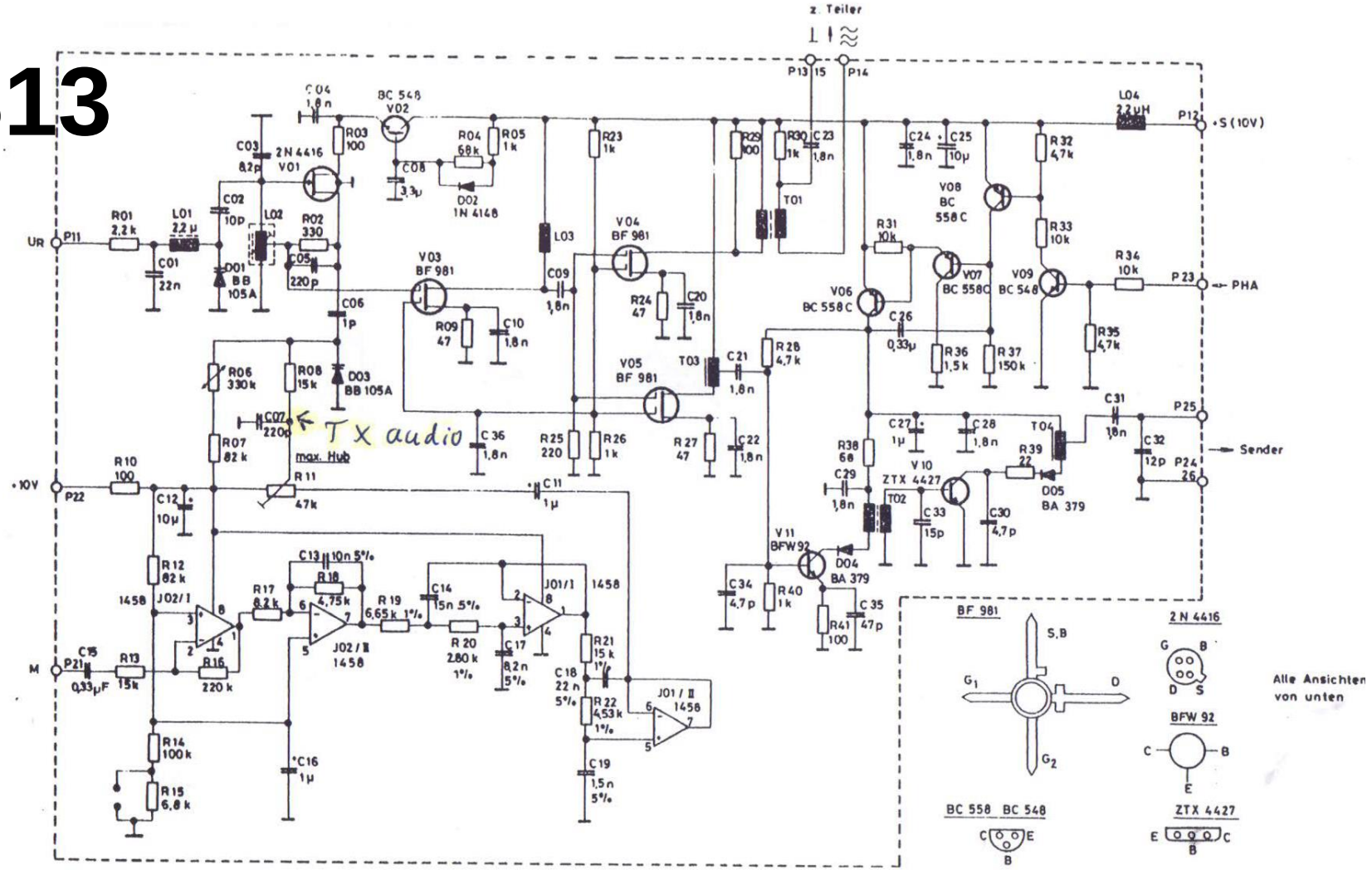
Table FM.2.1.

Credits: PE1ITR

Flash LED: Arduino SW

```
if (state == state_1)
    {delay(flash_start); // B: -...
     digitalWrite(led_m, HIGH);delay(flash_dash);digitalWrite(led_m, LOW);delay(flash_spacing);
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
    }
else if (state == state_2)
    {delay(flash_start); // D: -..
     digitalWrite(led_m, HIGH);delay(flash_dash);digitalWrite(led_m, LOW);delay(flash_spacing);
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
    }
else if (state == state_3)
    {delay(flash_start); // F: ..-.
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
     digitalWrite(led_m, HIGH);delay(flash_dash);digitalWrite(led_m, LOW);delay(flash_spacing);
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
    }
else
    {delay(flash_start); // H: ....
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
     digitalWrite(led_m, HIGH);delay(flash_dot); digitalWrite(led_m, LOW);delay(flash_spacing);
    }
```

T813



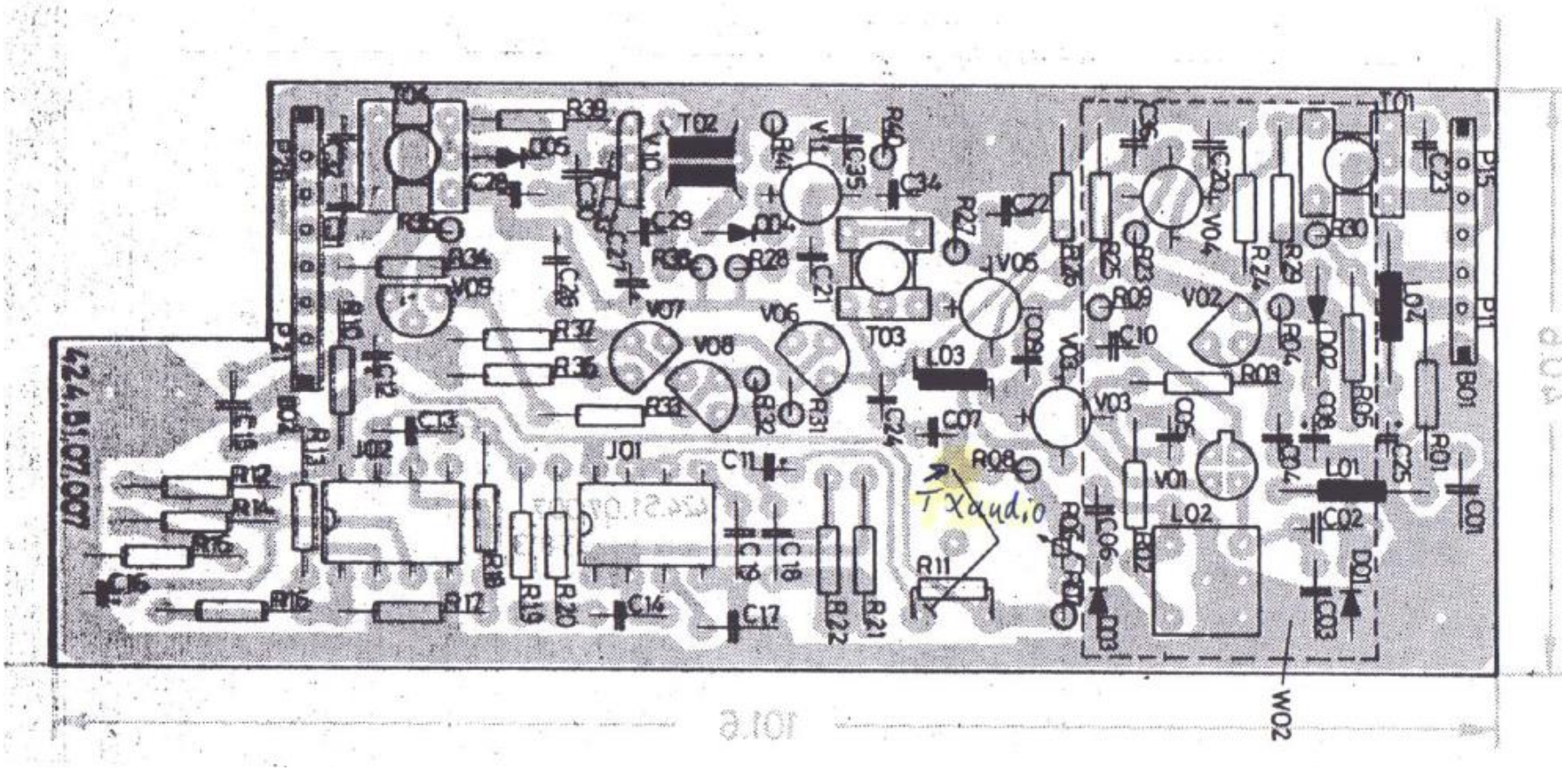
L02 : BV 39.424.104
L03 : BV 39.424.005
T01 : BV 39.424.801
T02 : BV 39.355.801
T03 : BV 39.424.803
T04 : BV 39.424.802

Credits: PE1FTV

Oszillator
Oscillator

T 813S

T813



Credits: PE1FTV