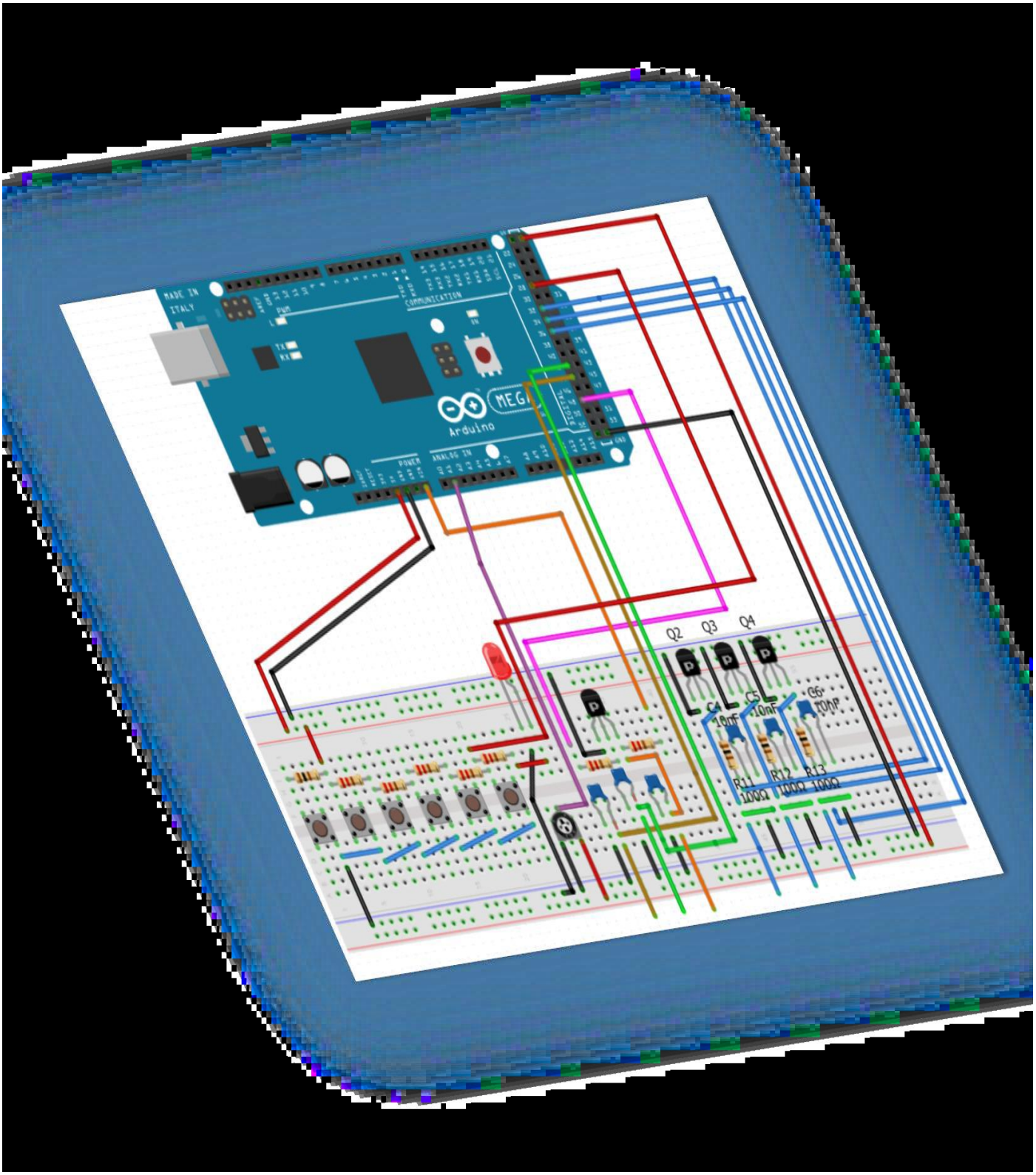
Manuale WINKEYER K3NG



Traduzione , grazie a Google, da IT9SAJ Carmelo => http://it9saj.altervista.org

INDICE pag.

Tastiera CW Arduino

1. Introduzione.

Un Arduino è un computer piccolo, relativamente economico e programmabile che può fare molto scrivendo il codice giusto. Un progetto Arduino è consigliato a chiunque sia interessato alla programmazione o ad armeggiare con l'elettronica. Puoi scegliere tra un'ampia varietà di progetti, ma io, come radioamatore, ho scelto il K3NG Arduino CW-Keyer. Con grande piacere ho portato a termine con successo questo progetto. I miei ringraziamenti a Ralph, PA1RB e Anthony Good, K3NG, non sarebbe stato certamente possibile senza il loro aiuto!

Forse questa guida può stimolare a lavorare con i progetti Arduino, è destinata al principiante con alcune conoscenze di base di elettronica. Acquisirai gradualmente la conoscenza di come gestire i microcontrollori e la programmazione associata. Si tratta di un progetto particolarmente divertente per le emittenti radiofoniche e per i radioamatori.

K3NG è il progettista dell'Arduino CW-Keyer, basato sul microcontrollore Arduino UNO.

Per i più avanzati, c'è un'ampia descrizione sul suo sito per la sua costruzione.

Tutta la sua programmazione è di tipo "open source", che offre la possibilità di apportare modifiche al programma a propria scelta.

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

Dato che ho l'Arduino MEGA 2560, fig.1, sono andato alla ricerca di un manuale basato su quest'ultimo. Dopo qualche sforzo ho finalmente trovato un sito di KF4BZT con il quale pensavo di poter fare qualcosa da principiante.

La differenza tra UNO e MEGA sta in una diversa assegnazione dei pin e KF4BZT lo adatta dalla descrizione di base di K3NG.

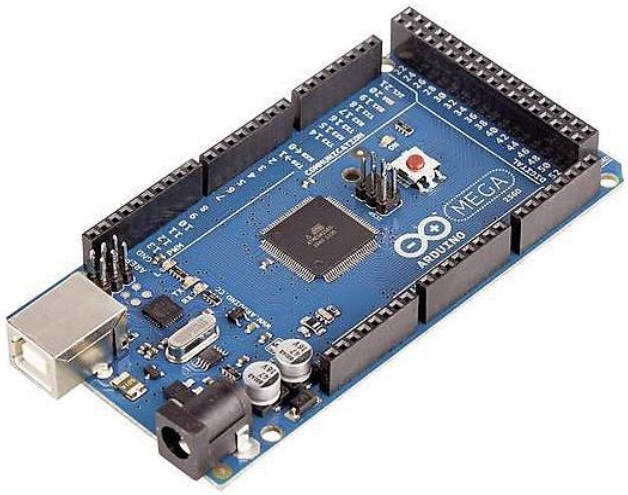


Figura 1, di Arduino MEGA 2560

Nel manuale di KF4BZT il CW-Keyer è costruito passo dopo passo, utilizzando l'Arduino MEGA 2560, lardellato di molte immagini (non) chiare.

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

Allora perché questo manuale si chiede. Bene, io da manichino sapevo poco del fenomeno Arduino e presto mi sono bloccato nella sua descrizione. Da un lato una spiegazione dettagliata, dall'altro mi sfugge la fase iniziale del "come comincio". Inoltre, è piuttosto sciatto sul suo sito con alcuni schemi, che contengono errori.

Prima di tutto, cosa offre questo keyer:

1.1 Caratteristiche del manipolatore K3NG CW.

Velocità CW regolabile da 1 a 999 WPM.

Fino a sei linee di tasti del trasmettitore selezionabili.

Programmazione e interfacciamento tramite porta USB ("interfaccia a riga di comando")

Interfaccia tastiera USB o PS2 per il funzionamento della tastiera CW senza computer.

Logging e Contest Program Interfaccia tramite emulazione del protocollo di interfaccia K1EL Winkey 1.0 e 2.0.

Versioni PTT opzionali con piombo, coda e tempi di sospensione regolabili.

Display LCD opzionale: modalità classica a 4 bit, display RGB Adafruit I2C o display LCD YourDuino I2C.

Fino a 12 memorie con macro.

Numeri seriali.

Tastiera CW (tramite un programma terminal server come Putty o il programma Arduino Serial).

Potenziometro velocità, optional - velocità regolabile anche con comandi

QRSS e HSCW.

Modalità Faro/Volpe.

Lambico A e B.

Modalità tasto diretto.

Modalità definitiva.

Modalità bug.

CMOS Super Keyer Iambic B Timing.

Pagaia inversa.

Modalità Hellschreiber (invia tastiera, macro di memoria, beacon)

Tempi di Farnsworth.

Tono laterale a frequenza regolabile.

Disattiva l'uscita sidetone/sidetone high/low per disabilitare l'oscillatore audio.

Modalità di comando per utilizzare la paletta per modificare le impostazioni, i promemoria dei programmi, ecc.

Compensazione chiavi.

Dah a questa regolazione del rapporto.

Imposta.

Esercizio di ricezione del nominativo.

Invia esercizio.

Impila la memoria.

AUTOSPAZIO.

Personalizzazione dello spazio delle parole.

Simboli preconfigurati e personalizzati.

Memorizzazione persistente della maggior parte delle impostazioni.

Design del codice modulare che consente la selezione delle funzioni e una facile modifica del codice.

Supporto per caratteri non inglesi.

Decodificatore di ricezione CW (SPERIMENTALE).

Coefficiente di velocità della manopola rotante.

Modalità risparmio.

Supporto per mouse USB.

Supporto per anello LED Mayhew

Pratica di spedizione dell'alfabeto.

QLF / Emulazione chiave diretta (NOVITÀ).

Interfaccia USB Keyboard HID (Human Interface Device) (Keyer = tastiera per il tuo computer) (NUOVO).

Di seguito sono riportati i pin digitali / PWM / analogici disponibili su tre Arduino comunemente usati:

Pin Arduino Nano.

Pin I/O digitali 14

Pin PWM 6

Pin I/O analogici 8

Pin di Arduino Yun.

Pin I/O digitali 20

Pin PWM 7

Pin I/O analogici 12

Arduino Mega 2560 Pin.

Pin I/O digitali 54

Pin PWM 15

Pin I/O analogici 16

Arduino UNO e Mega 2560.

Il vantaggio di Arduino Mega 2560 è che ha 256 KB di memoria flash (di cui 8 KB utilizzati dal boot loader). Questo ti dà lo spazio per tutte le possibilità offerte dal firmware.

Di seguito puoi vedere i pin Digital / PWM / Analog disponibili su Arduino MEGA 2560. Il Mega ti consente di fare molte più connessioni, più connessioni meglio è.

Non tutti i pin verranno utilizzati per questo, poiché ci sono modi per accoppiare le connessioni insieme attraverso un'unica linea, come la messa a terra e la linea da 5 volt per alimentare il circuito. Di seguito è riportato un confronto con Arduino UNO.

Pin Arduino Nano.

Pin I/O digitali 14

Pin PWM 6

Pin I/O analogici 8

Arduino Mega 2560 Pin.

Pin I/O digitali 54

Pin PWM 15

Pin I/O analogici 16

1.2 Specifiche di ARDUINO MEGA 2560.

Importanti sono i pin digitali e analogici e la velocità di clock (decodifica CW).

Specifiche tecniche:

Microcontrollore ATmega2560

Tensione di esercizio 5V

Tensione di ingresso (consigliata) 7-12V

Tensione di ingresso (limite) 6-20V

Pin I/O digitali 54 (di cui 15 con uscita PWM)

Pin di ingresso analogico 16

Corrente CC per pin I/O 20 mA

Corrente CC per pin 3,3 V 50 mA

Memoria flash 256 KB di cui 8 KB utilizzati dal bootloader

SRAM 8 KB

EEPROM 4KB

Velocità di clock 16 MHz

Lunghezza 101,52 mm

Larghezza 53,3 mm

Peso 37 g

Come puoi vedere sotto, fig.2, c'è una sezione digitale separata che inizia con il pin 22 fino al pin 53. La versione del pin analogico, in basso a sinistra, è etichettata da A0 a A15. L'alimentazione è sul pin 5 Volt con, tra l'altro, due pin di massa (GND).

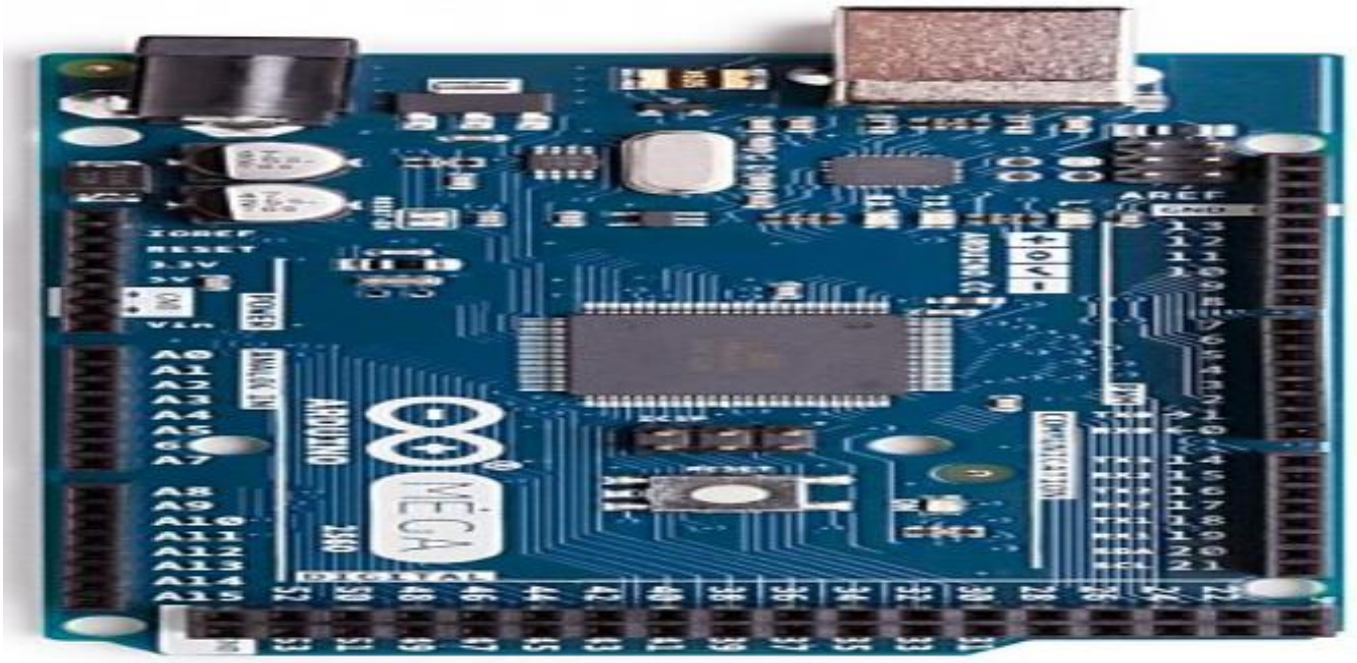


Figura 2, Di Arduino MEGA 2560

1.3 Arduino Basato su un ingresso, un circuito Arduino può eseguire azioni indipendenti emettendo segnali di uscita digitali e analogici.

In questo caso, puoi utilizzare un microcontrollore Arduino MEGA 2560 dei molti tipi di microcontrollori Arduino per controllare un ricetrasmettitore con un tasto Morse o una tastiera, puoi anche rendere visibili i caratteri Morse come testo su un display. L'Arduino Mega 2560, con il chip Atmega 2560, ha una memoria programmabile di 256 KB e 70 connessioni programmabili.

1.4 Le parti più fondamentali.

Di cosa hai bisogno inizialmente per questo:

a. Una cosiddetta breadboard, fig.3a

Questo progetto richiede 2 breadboard. Nella figura 3b puoi vedere come sono disposti i collegamenti sotto la breadboard.

Figura 3a, parte anteriore della breadboard

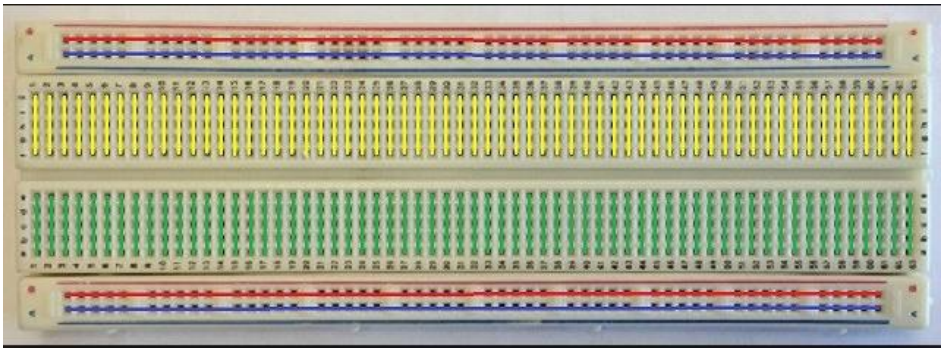


Figura 3b, retro della breadboard

b. Fili di collegamento, non ordinarne troppo pochi, inoltre sono relativamente economici, fig.4.

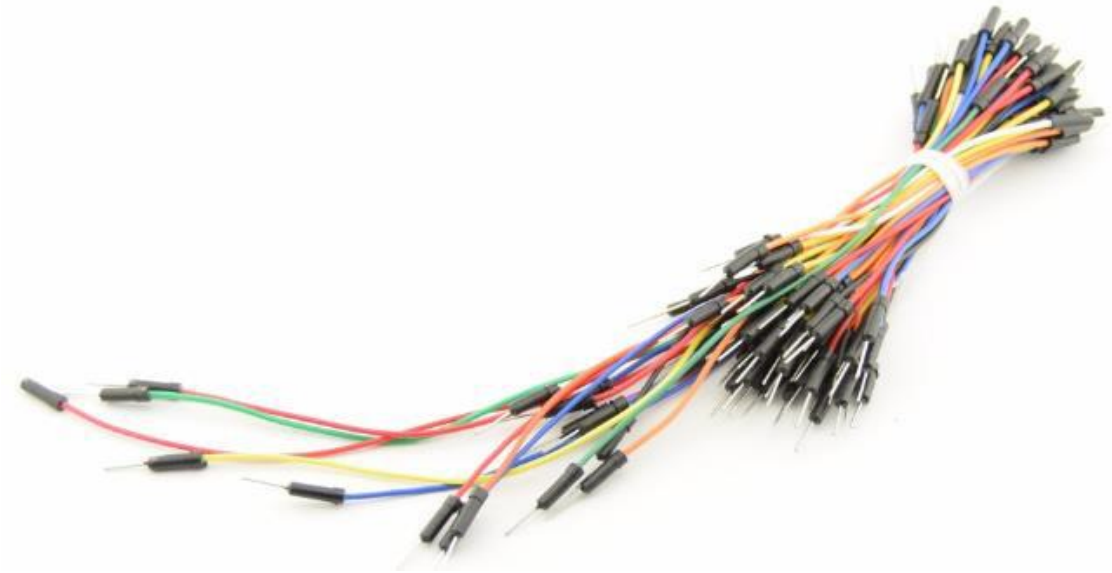


Figura 4, fili di collegamento

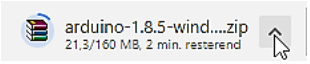
c. Inoltre, ovviamente, tutti i componenti comuni inclusi nei circuiti, vedere il capitolo 11, elenco dei componenti.

2. Installazione dell'IDE Arduino 1.8.5

Arduino IDE è un programma che ti permette di comunicare tra il tuo PC e il microcontrollore e di caricare il software necessario sul microprocessore. Quindi scaricherai prima questo programma.

Crea un file da qualche parte sul tuo Hard Drive, fig.5, con ad esempio il nome: Cartella di download temporanea K3NG e scarica il seguente file zip:

https://www.arduino.cc/en/Principale/Software



Quindi, crea una nuova cartella chiamata: C:/Arduino e copia il contenuto del file arduino-1.8.5 dalla cartella di download temporaneo nella cartella Arduino, fig.6.

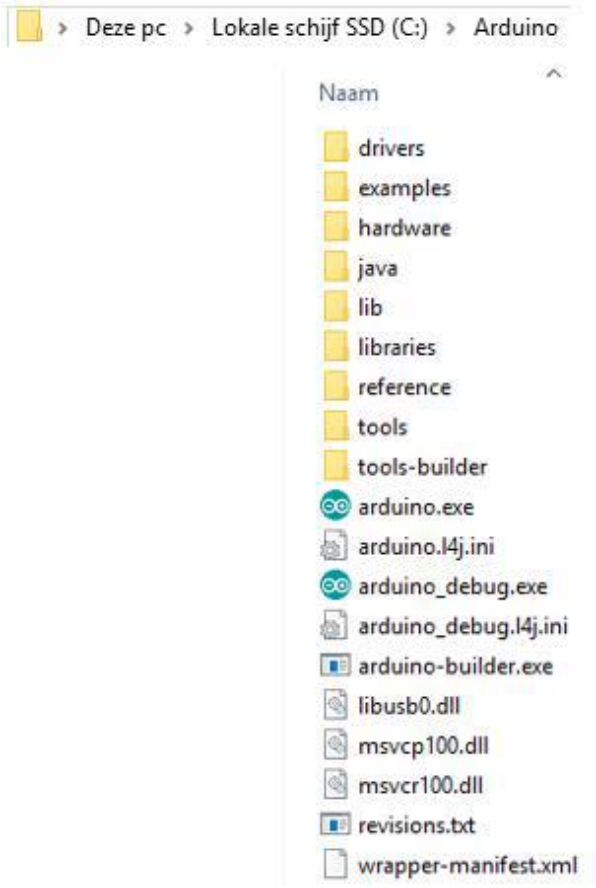


Figura 6, il contenuto di arduino-1.8.5

Ora avvia il programma con il file Arduino.exe.

Questo per quanto riguarda l'installazione dell'IDE di Arduino.

3. Lavorare con il programma Arduino IDE.

È giunto il momento di collegare la porta USB del tuo Arduino MEGA 2560 alla porta USB del tuo computer.

Avvia Arduino IDE e ad esempio apparirà una finestra Arduino simile, fig.7.

3.1 Configurazione dell'IDE Arduino

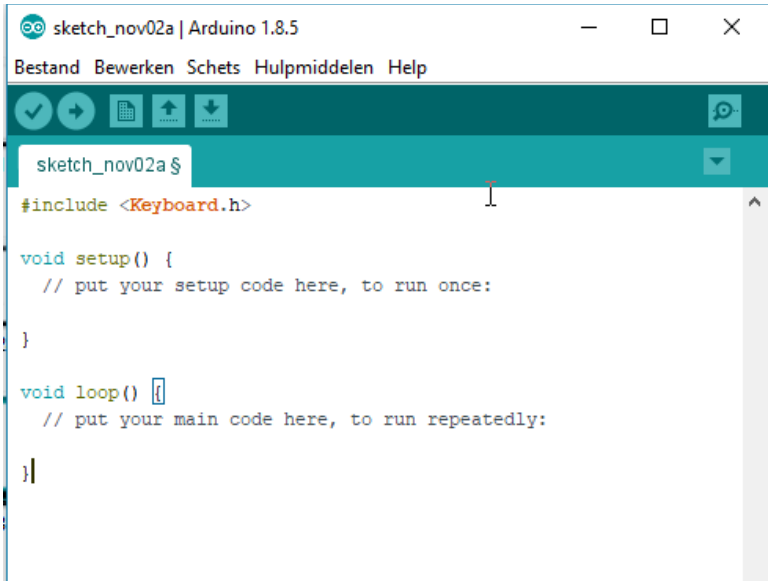


Figura 7, Arduino, una possibile schermata iniziale.

Per prima cosa scegli le seguenti opzioni, Fig. 8 e 9 da Strumenti.

Da Strumenti selezionare Arduino Mega, fig.8

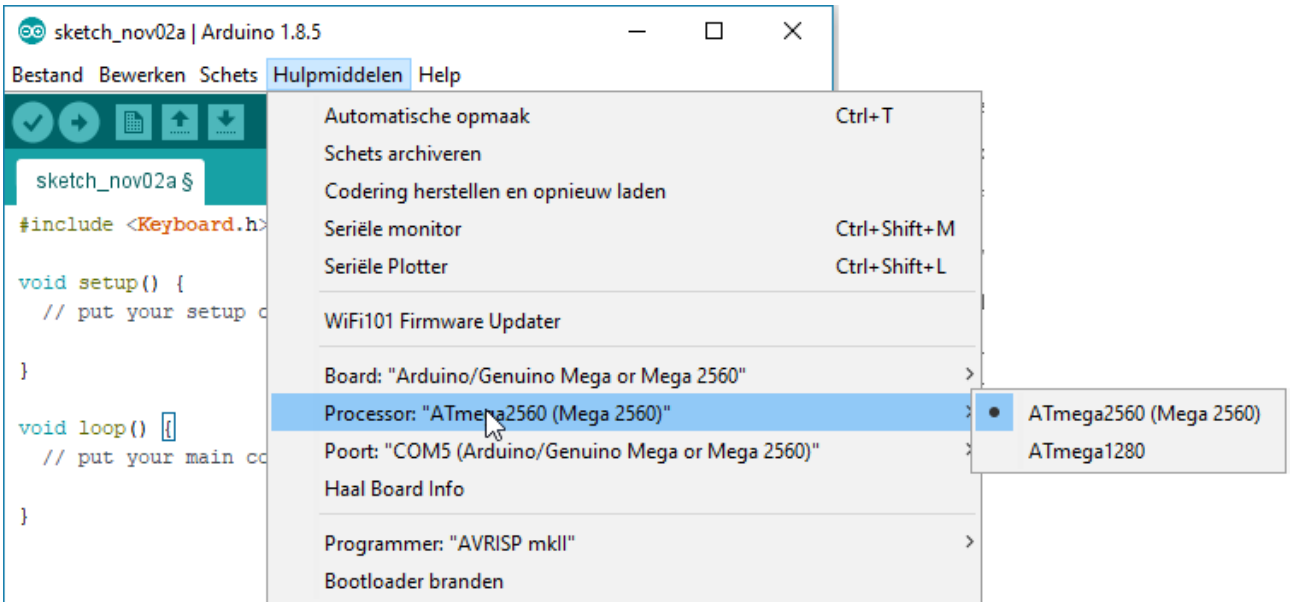


Figura 8, ATmega2560

Una porta è solitamente già determinata da Windows 10, fig.9. Se le cose continuano a non funzionare, controlla in Gestione dispositivi se Arduino è collegato alla porta USB corretta, fig.10. Puoi farlo tramite: Pannello di controllo > Gestione dispositivi, quindi inserisci il numero di porta corretto. Ovviamente puoi assegnare una porta diversa come nell'esempio seguente.

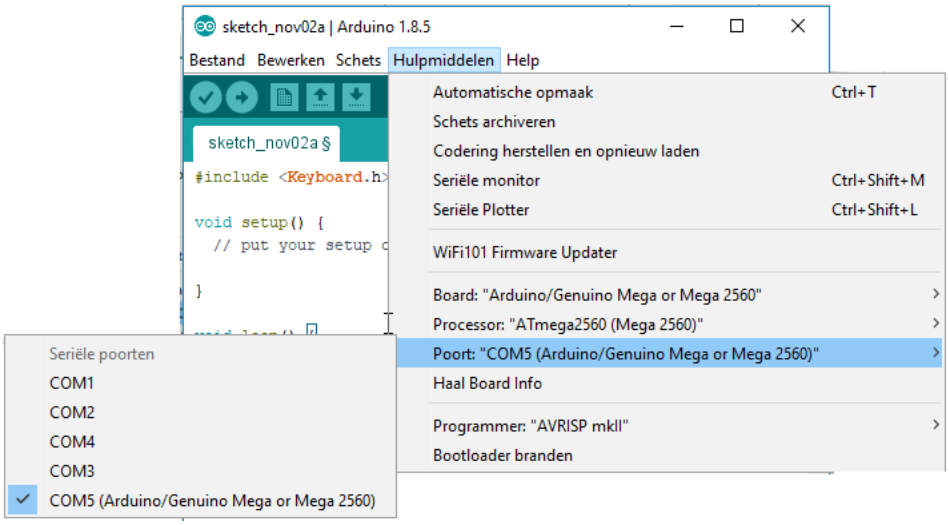


Figura 9, Assegnazione del numero di porta

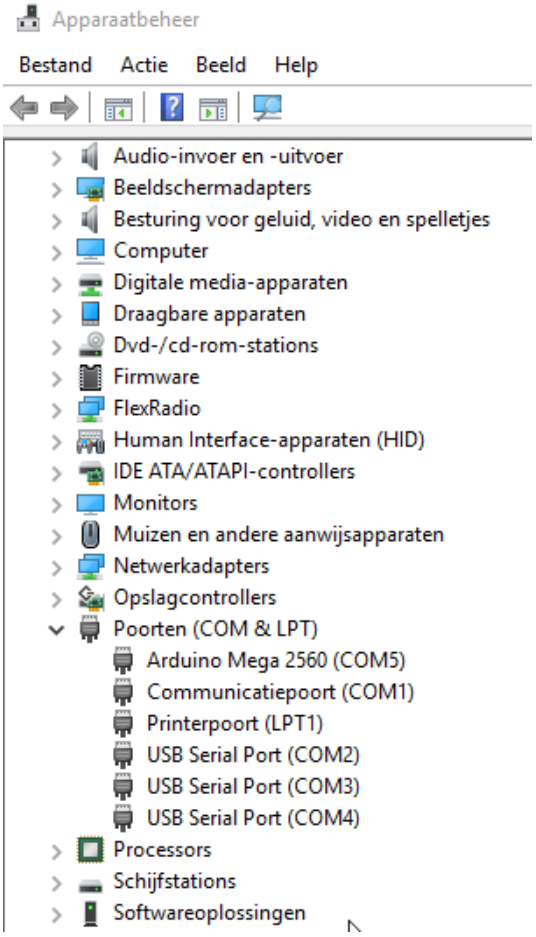


Figura 10, Gestione dispositivi

4. Schizzo lampeggiante.

Prima di iniziare con lo sketch (programma) K3NG, è consigliabile caricare prima lo sketch Blink. Arduino ha fornito una serie di programmi utili per questo in: Esempi. Con Blink puoi quindi verificare se anche Arduino MEGA funziona correttamente e potrebbero volerci solo 10 minuti. Ora disconnettiti dal tuo PC.

NB. Non è subito obbligatorio utilizzare l'alimentatore esterno (7-12 Volt DC), ma offre un vantaggio con circuiti più grandi e l'uso di un display, in modo che il funzionamento diventi più stabile. L'alimentazione tramite la connessione USB viene quindi automaticamente disattivata, ma l'alimentazione interna dell'Arduino rimane di 5 Volt.

4.1 Configurazione.

La Fig. 11 di seguito mostra la semplice configurazione di Arduino Mega sulla breadboard.

Fai attenzione a come posizioni le parti.

Sulla maggior parte dei LED, il cavo più lungo è il " Catodo – " (lato diritto).

Figura 11, Configurazione Blink

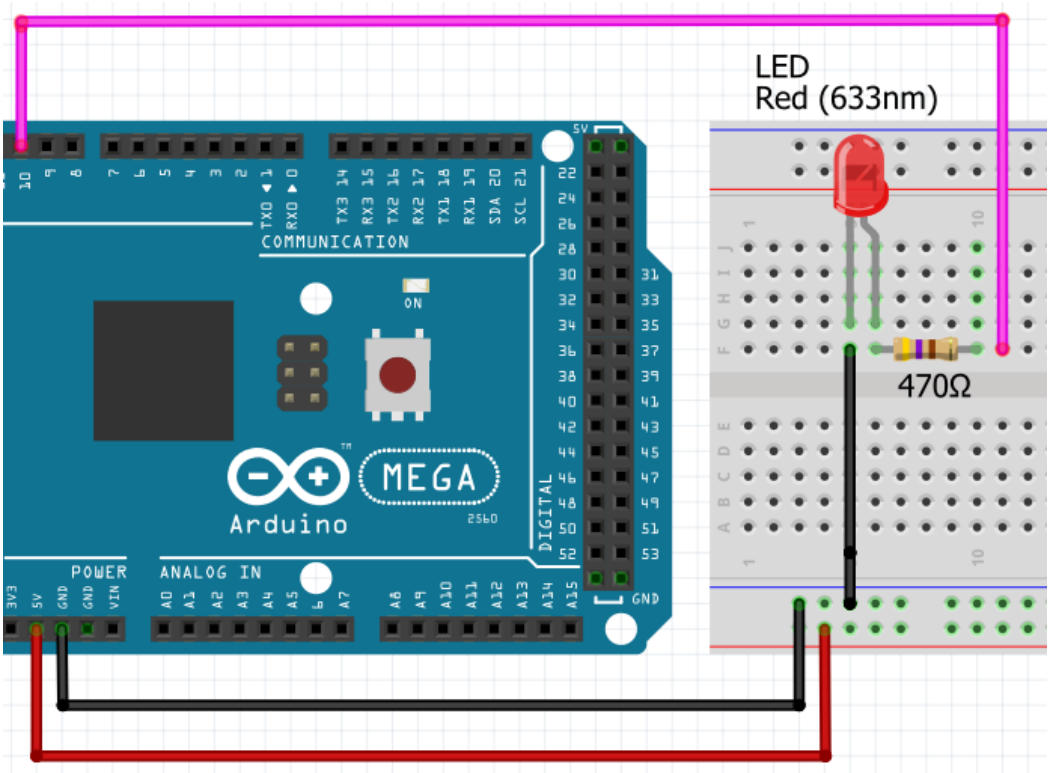
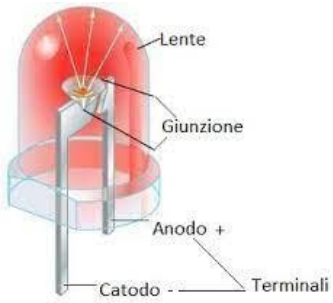


Figura 11, Configurazione vuota



Quando ti connetti alla breadboard, cerca di essere concentrato e di non avere fretta. È molto facile fare un errore. Se non stai attento, presto sarai pronto per il tuo prossimo Arduino MEGA 2560, come è successo a me.

4.2 Creazione e avvio del programma.

Prima di tutto, questo manuale non vuole darti una spiegazione del linguaggio di programmazione, ma con un piccolo sforzo puoi trovare abbastanza su internet per capire approssimativamente come funzionano i programmi. Inoltre, alcuni compiti contengono righe di commento con una spiegazione.

Avvia Arduino IDE e ora carica lo sketch Blink, fig.12.

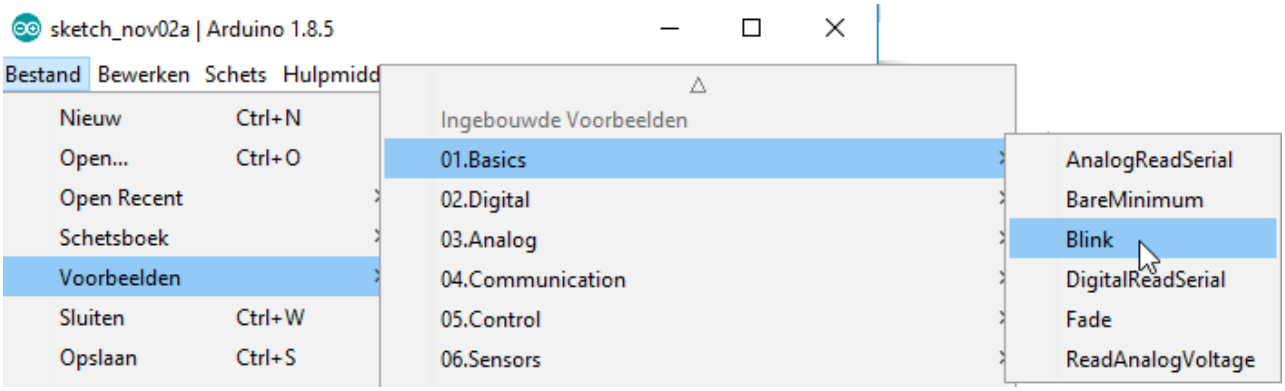


Figura 12, Selezione di Blink: Esempi 01.Nozioni di base > Blink

Una chiara spiegazione di questo programma può essere trovata su:

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

Il programma Blink apparirà ora nella finestra dell'IDE di Arduino. Nell'esempio seguente, Fig. 13, la variabile pin è impostata a 10, secondo la disposizione di Fig.10.

Qualsiasi cosa tra: /* …….. */ viene letto come testo e non compilato.

Anche le righe precedenti: // non vengono compilate.

Aggiungi la riga: 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
```

Figura 13, il programma Blink

Se tutto è corretto sulla breadboard, collega la porta USB della breadboard al PC.

Cliccare sull'icona “V”, fig.14, per compilare il programma. Se ricevi un messaggio di errore, hai commesso un errore di battitura nel programma.

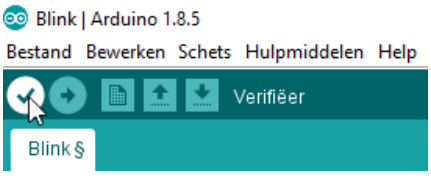


Figura 14, Compilazione.

Quindi fare clic sull'icona "freccia" per caricare il programma su Arduino, fig.15.

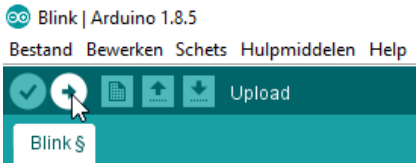


Figura 15, caricamento

Se non ricevi nemmeno un messaggio di errore durante il caricamento, il LED si accende e si spegne agli intervalli di tempo programmati. Quindi, hai completato con successo il tuo primo esperimento.

Successivamente, puoi iniziare con il progetto "reale". La struttura è descritta passo dopo passo per ridurre il rischio di errori.

Salvare i tuoi programmi di disegno: vai su File > Preferenze fig.16 e 16a e salva i tuoi schizzi in una cartella omonima.

Non dimenticare di fare clic su OK.

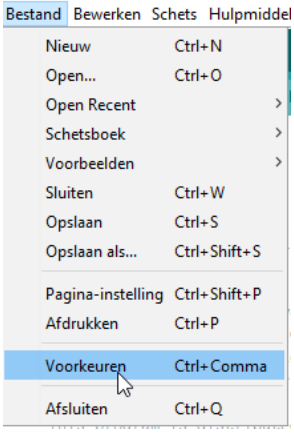


Figura 16, Cartella di salvataggio dei file

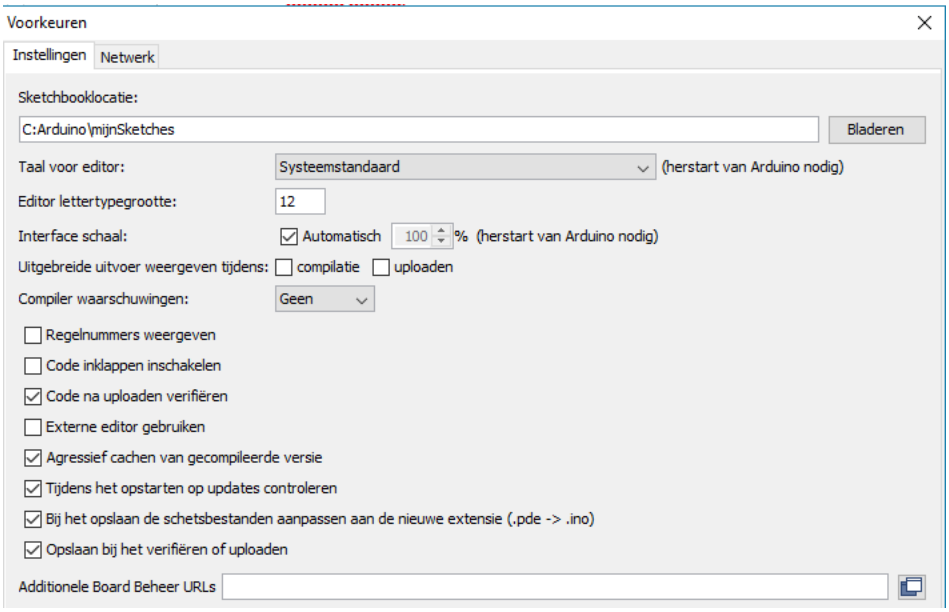


Figura 16a, Preferenze

5. Keyer K3NG CW, UNO

Di seguito è riportato lo schema originale, fig.17, del keyer Arduino K3NG CW basato su UNO e può essere trovato sotto:

In generale, questo schema viene utilizzato in questo manuale, ma con un'assegnazione delle porte modificata per il MEGA 2560 e un'aggiunta per un collegamento display e tastiera PS2.

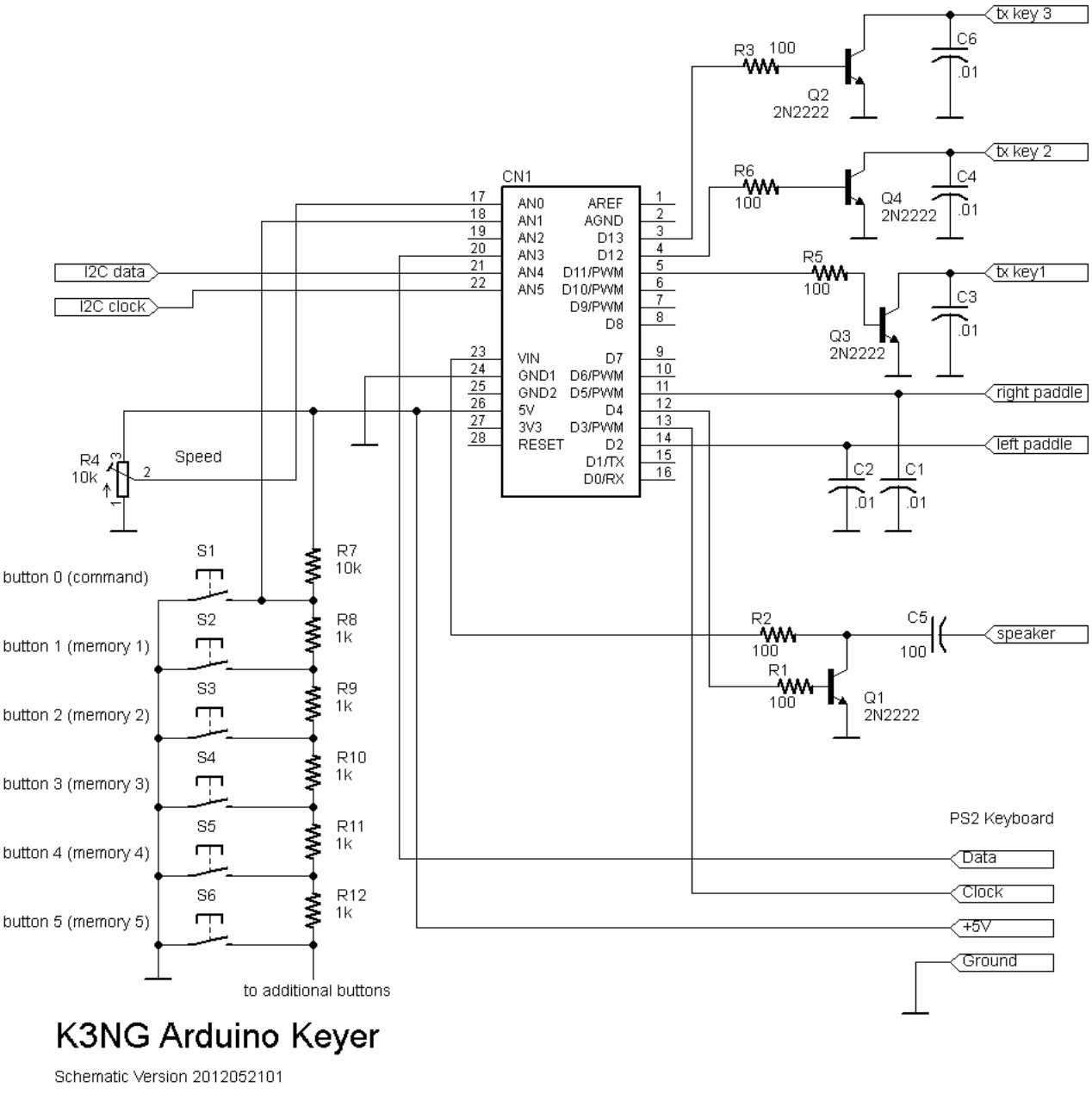


Figura 17, schema di base K3NG

Si noti che tutti i C utilizzati possono essere condensatori a disco ceramico, quindi senza polarità

6. Formatta i file del keyer K3NG CW.

La struttura della mappa è molto importante per il corretto funzionamento del progetto e purtroppo su internet c'è poco da trovare per questo motivo mostro di seguito, con l'aiuto di un organigramma, fig.18, come impostare Arduino struttura.

I rettangoli gialli mostrano le cartelle ei rettangoli blu i file associati.

Organigramma del keyer Arduino K3NG

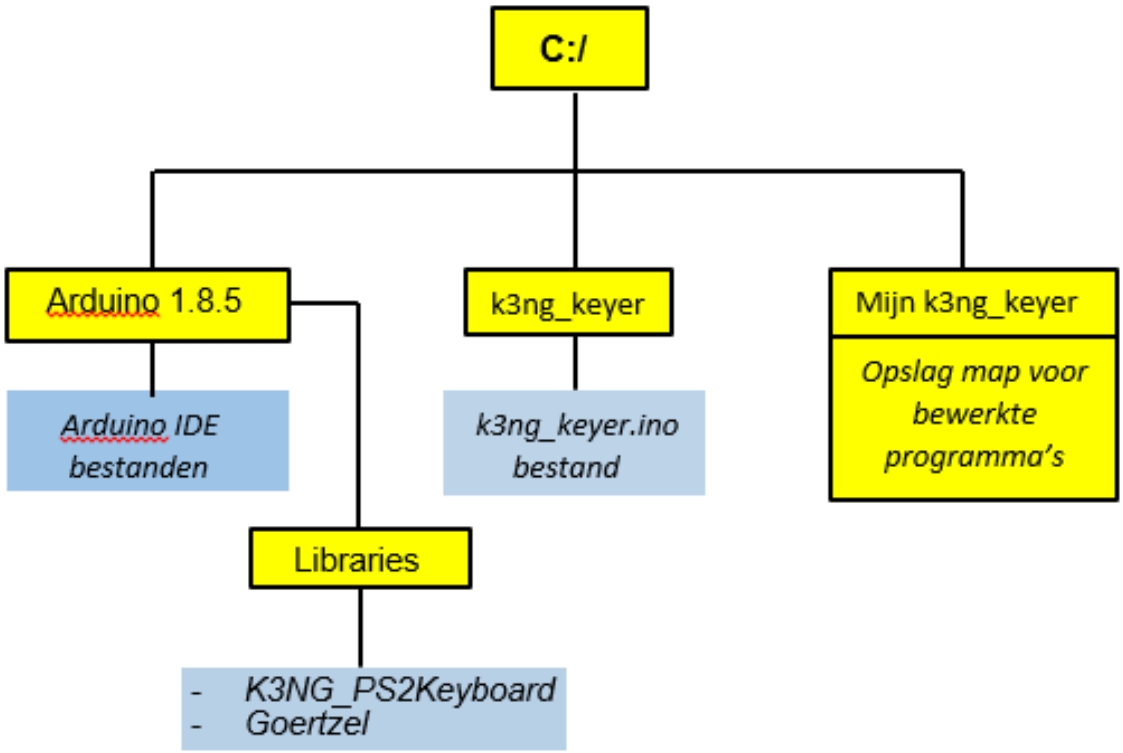


Figura 18, Organigramma di Arduino K3NG

Crea una nuova cartella chiamata: k3ng_keyer sotto C/:

Per il keyer K3NG è necessario scaricare il file zip sottostante e inserirlo nella cartella di download temporaneo. Il passaggio intermedio di inserirlo prima in una cartella temporanea sembra un po' macchinoso, ma offre il vantaggio di avere tutti i file di base rapidamente a portata di mano durante una reinstallazione.

https://github.com/k3ng/k3ng_cw_keyer

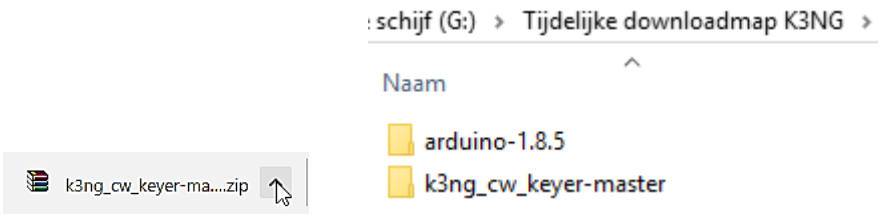


Figura 19, cartella di download temporanea

6.1 Mettere i file nel posto giusto.

Supponendo che la posizione dell'IDE Arduino, dal progetto Blink, sia ancora nello stesso posto, procedere come segue, secondo l'organigramma:

Vai alla cartella delle librerie tramite:

Cartella di download temporanea > k3ng_cw_keyer-master > librerie

Copiare la cartella K3NG_PS2Keyboard e Goertzel da questa libreria, fig.20 e incollarla nella cartella della libreria Arduino, fig.20a.

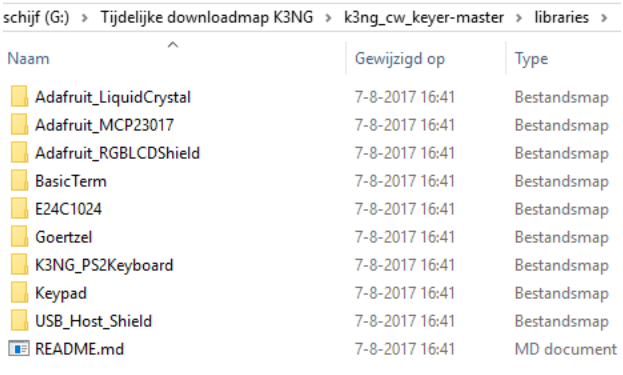


Figura 20, cartella delle librerie

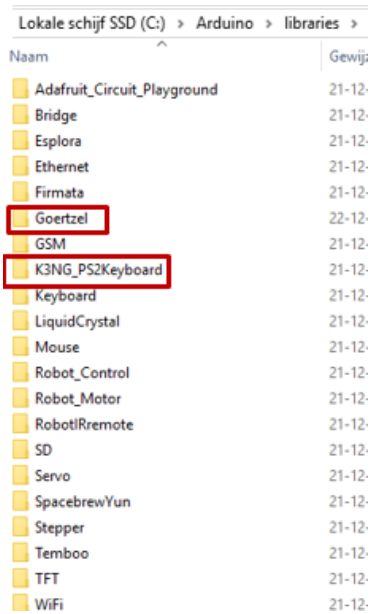


Figura 20a, mappa K3NG_PS2Keyboard in Goertzel

Per chiarire, la mappa della tastiera PS2 verrà aggiunta alla libreria, poiché in seguito collegherai la tua tastiera e Goertzel per decodificare Morse sul display. In questo modo, Arduino IDE può trovare il file necessario ed elaborarlo ulteriormente. (Grazie per l'aiuto di Fred, VK2EFL).

Quando inizierai a creare il programma keyer K3NG, nel prossimo capitolo, avrai bisogno del file k3ng_keyer.

Vai su k3ng_cw_keyer-master e copia da lì la cartella k3ng_ keyer e incollala in C:/, fig.21.

La cartella e il formato del file sono ora completi e ora puoi finalmente inizia con il progetto Arduino K3NG Keyer stesso.

Figura 21, manipolatore k3ng

7. Il K3NG Arduino MEGA 2560 a montaggio superficiale.

7.1 A partire dalla creazione dei componenti di Arduino K3NG keyer.

Scollega la connessione USB del tuo PC ogni volta che inizi a lavorare sulla breadboard.

Ora puoi finalmente iniziare a costruire il keyer fisico.

La breadboard che sto utilizzando corrisponde a quella mostrata a pagina 7, fig.3.

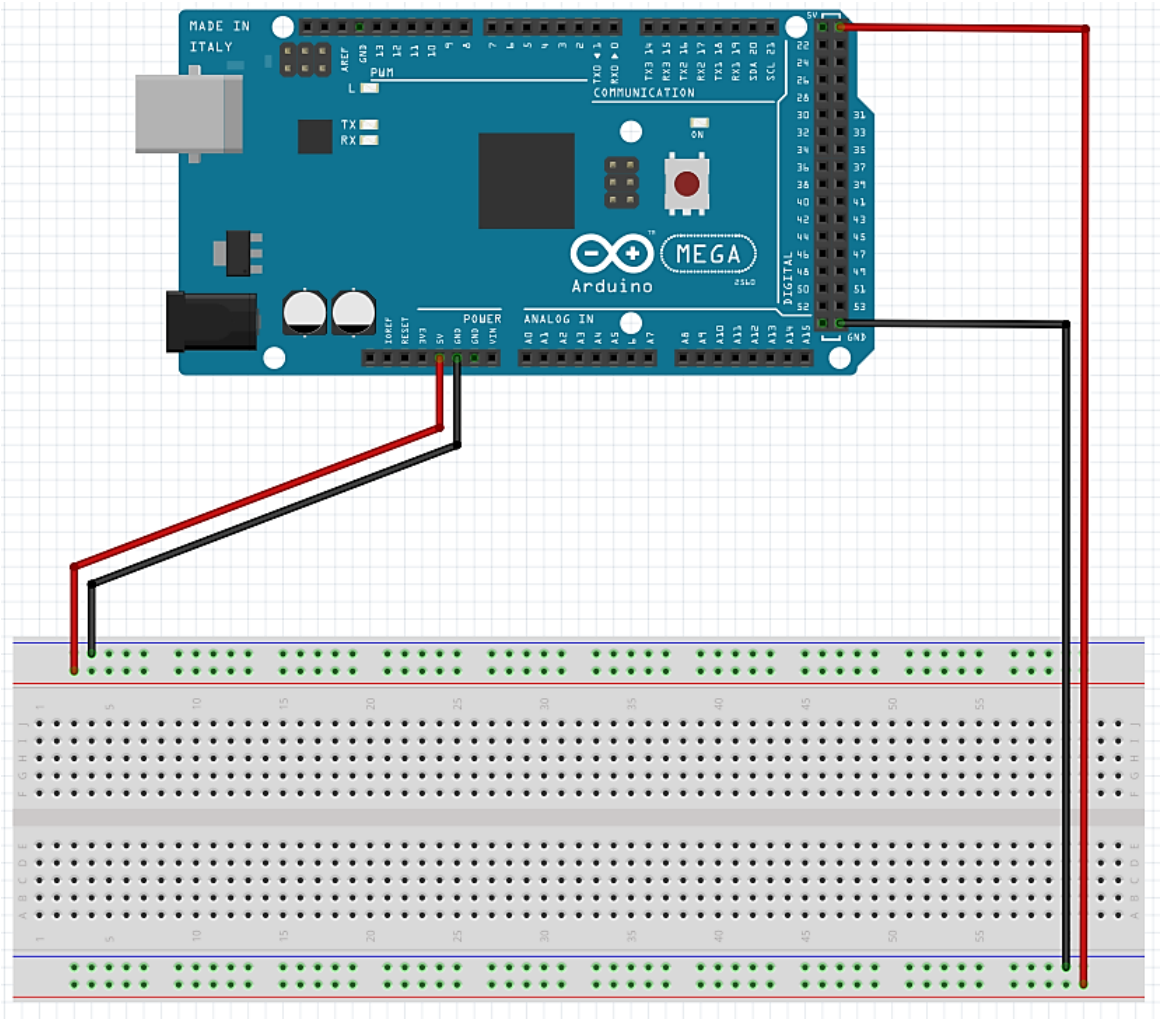


Figura 22, configurazione di base della breadboard con linee elettriche da Arduino MEGA

8. Il progetto passo dopo passo.

Passaggio 8.1: aggiunta dei pulsanti.

Come prima fase di questo progetto posizionare i pulsanti il più a sinistra possibile sulla scheda, fig.24. Sono in totale sei, di cui quello più a sinistra è il pulsante di comando per la programmazione. Gli altri pulsanti sono pulsanti di memoria. Puoi aggiungere più pulsanti di memoria in un secondo momento, se lo desideri. I pulsanti sono semplici interruttori on-off, collegati tra loro secondo lo schema, fig.23.

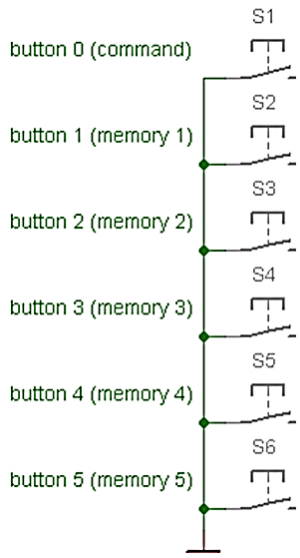


Figura 23, diagramma con pulsanti

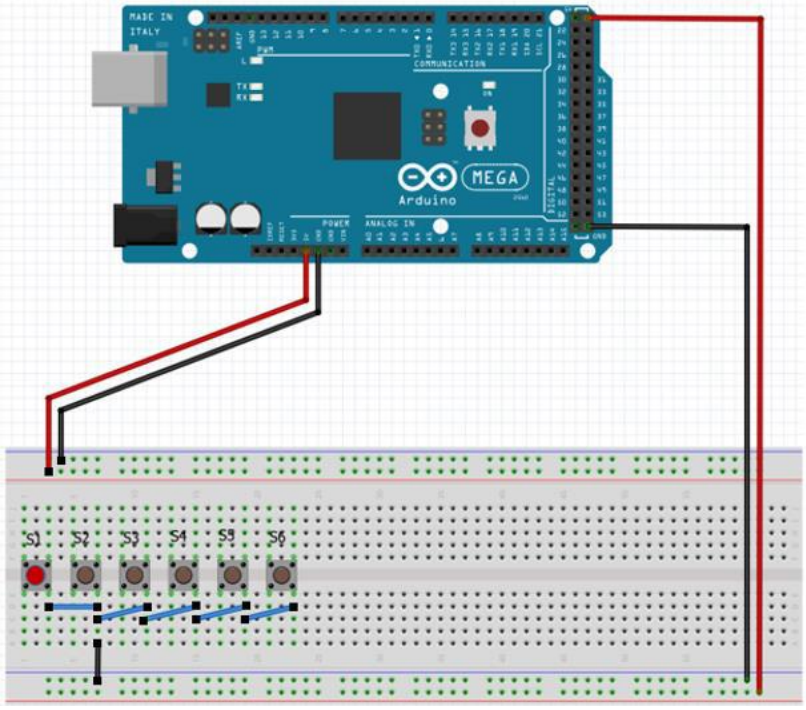


Figura 24, i pulsanti.

Passaggio 8.2: aggiunta dei resistori.

Posizionare le 6 resistenze sulla scheda, fig.26 e prestare attenzione alla posizione delle resistenze, si sbaglia facilmente.

La Fig, 25 mostra di nuovo il circuito.

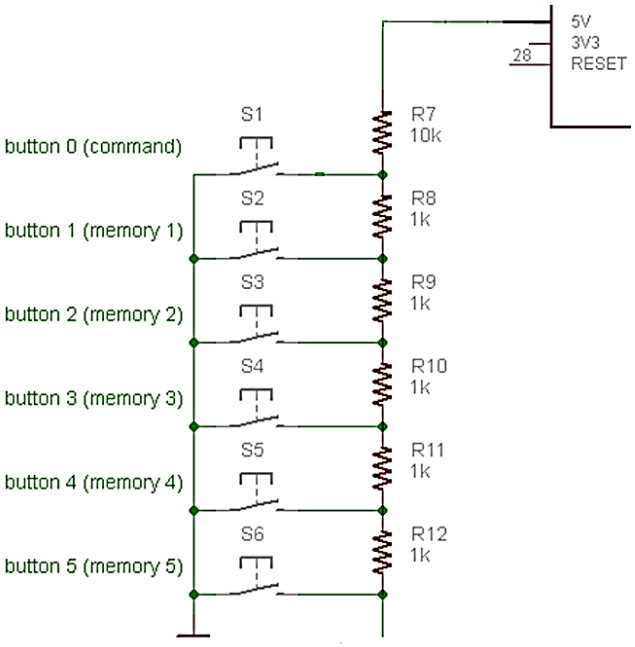


Figura 25, sei resistori.

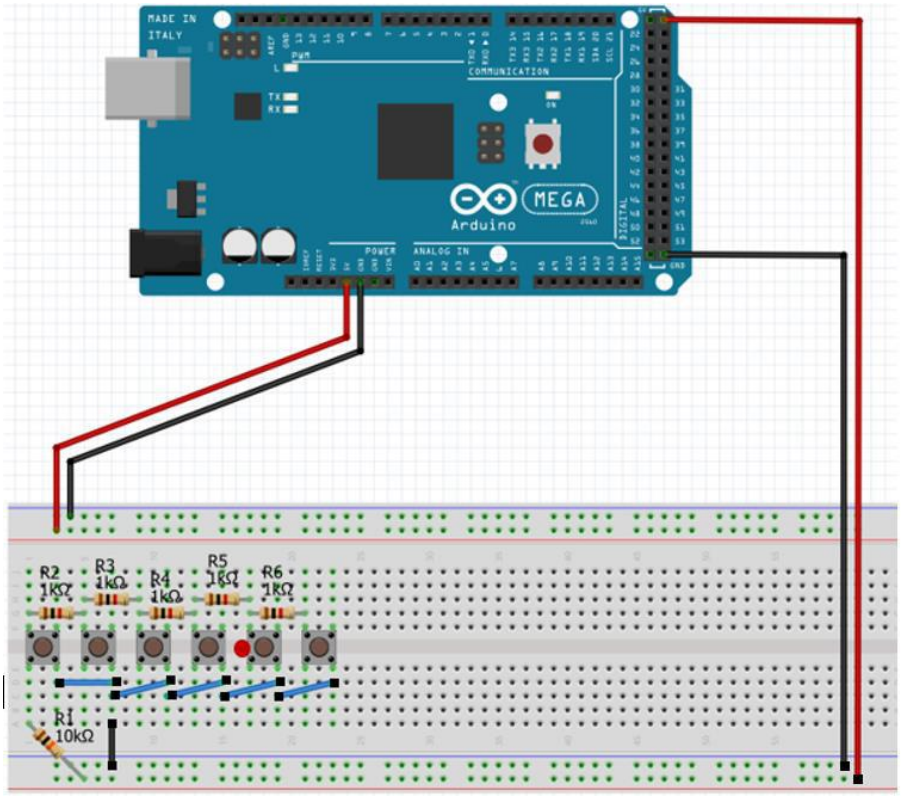


Figura 26, pulsanti con resistori.

Passaggio 8.3: aggiunta di un LED.

Questo passaggio può essere trovato anche sul sito Web di K3NG e fornisce un controllo se il pulsante di comando è stato premuto ed è incluso nei passaggi seguenti, fig.27 e 28.

Viene aggiunto un LED per verificare se il pulsante di comando è premuto. Premere brevemente una volta il pulsante di comando (la porta digitale D28 va “alta”, circa 5 Volt) per far accendere il led, premere ancora brevemente, il led si spegne nuovamente. Prestare molta attenzione a come è posizionato il LED, il filo di collegamento più lungo è il + ed è collegato alla resistenza da 470 Ohm.

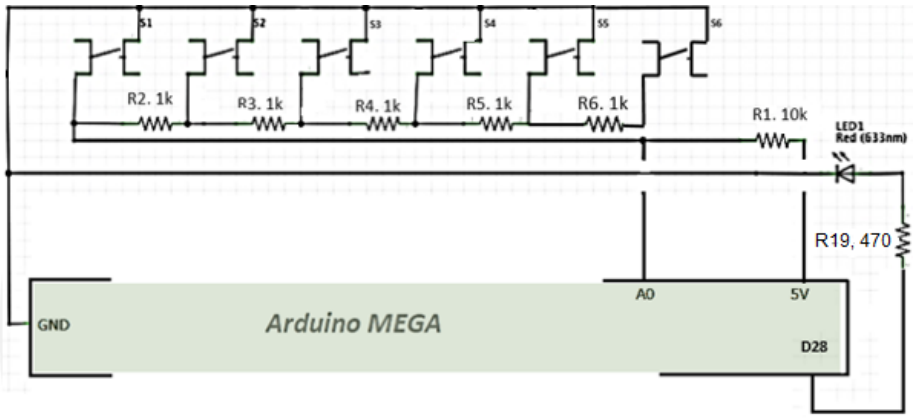


Figura 27, schematica con LED

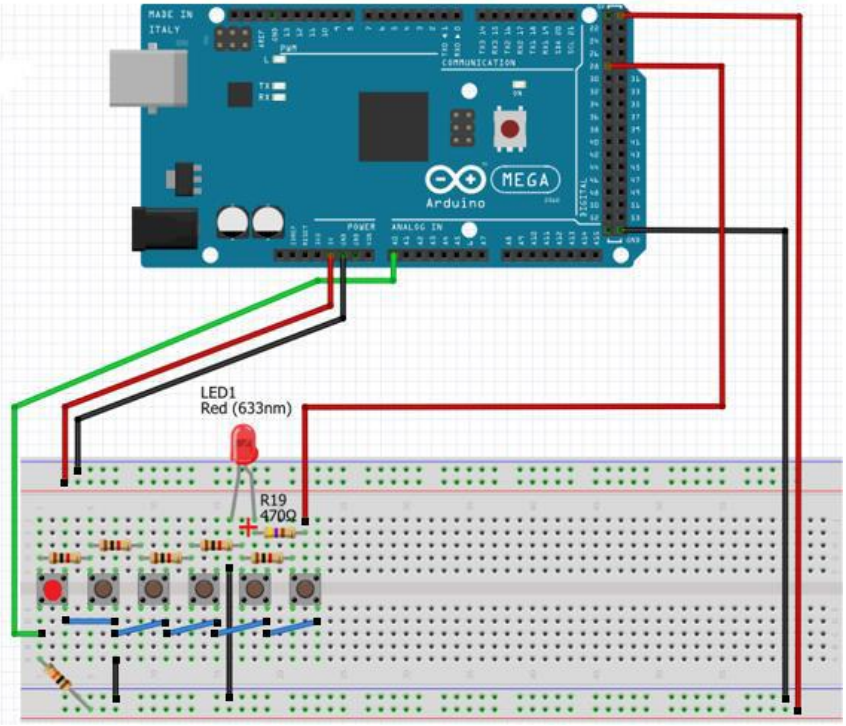


Figura 28, configurazione

Figura 29, nell'IDE Arduino: File/Apri

Figura 29, apri k3ng_keyer.ino

Figura 39a, file k3ng_keyer.ino di base

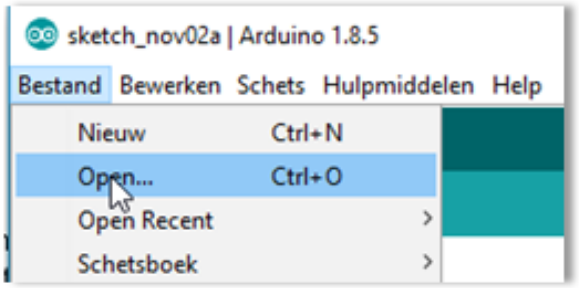
Passaggio 8.4: caricamento dei dati del keyer K3NG nell'IDE di Arduino

Apri il programma Arduino IDE.

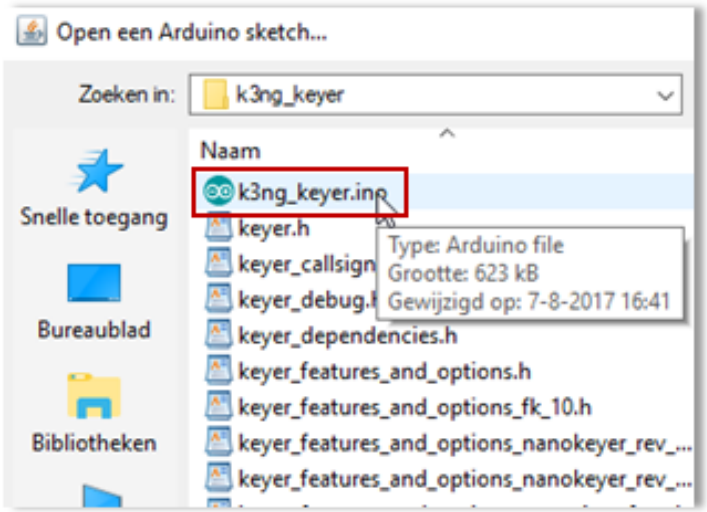
Per aprire il programma K3NG di base nell'IDE di Arduino, vai su Apri e fai clic sul file k3ng_keyer.ino dalla cartella:

SSD (C:) Arduino k3ngkeyer, figure 29 e 29a.

In esso troverai tutto il necessario h. file necessari per costruire il keyer.



Figuur 31, In Arduino IDE: Bestand /Openen



Figuur 31a, open k3ng_keyer.ino.

Figura 29, apri k3ng_keyer.ino

Figura 29a, file k3ng_keyer.ino di base

Tutti i file .h necessari ora appaiono nella barra verde orizzontale della finestra principale sottostante, che puoi vedere più chiaramente quando apri la finestra di espansione nascosta, fig.30.

Ora inizi dalla configurazione di base di K3NG e poi apporti le modifiche necessarie. Cosa fare con la cosiddetta h. i file possono/devono essere eseguiti, sarà chiarito nel resto del manuale.

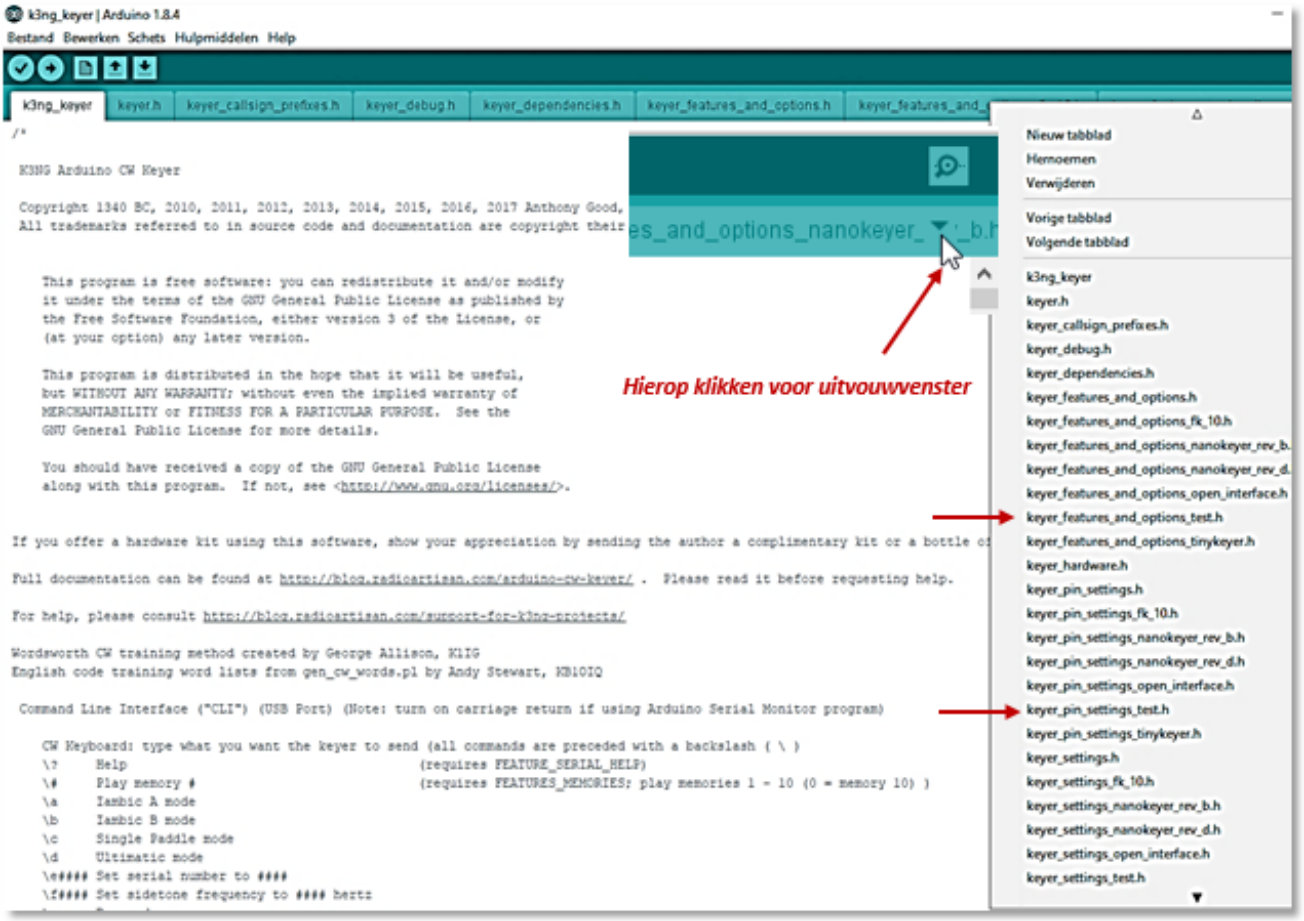


Figura 30, Finestra principale dell'IDE Arduino

I due h. i file sono gli strumenti necessari che utilizzerai per apportare modifiche al programma.

a. keyer_pin_settings.h

Qui è dove determini l'occupazione del programma.

b. keyer_features_and_options.h

Lì puoi abilitare o disabilitare alcune opzioni del programma K3NG di cui hai bisogno per il tuo progetto.

Passaggio 8.5: regolazione del programma base K3NG

keyer_pin_settings.h, che determina l'impostazione del pin.

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

Figura 31, impostazioni dei pin

Il pin analogico è stato scelto per: A0. (Attualmente Command_mode_active_led è impostato su uno stato inattivo). Il pin digitale per la porta 28, fig.31, che commuta il LED.

A questo punto devi attivare tre comandi, fig.32 e lo fai rimuovendo i caratteri //. Non dimenticarlo, altrimenti il programma non funzionerà.

keyer_features_and_options.h

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

Figura 32, attivazione delle funzioni.

Con "Modifica > Trova..." puoi trovare rapidamente un comando nel programma.

Una volta apportate le modifiche ai codici, puoi compilare il programma e, dopo aver collegato il tuo Arduino MEGA, caricarlo su Arduino Mega per il test.

Anche qui se sei sicuro che tutti i componenti siano collegati correttamente?

Non dimenticare di compilare e aggiornare il programma.

Dopo aver premuto il pulsante Command, il LED dovrebbe accendersi e premendo nuovamente questo pulsante si spegnerà.

Passaggio 8.6: estendere con il potenziometro.

Di seguito troverai lo schema, fig.33 e la configurazione della breadboard, fig.34, con il potenziometro allegato, che puoi utilizzare in seguito per impostare la velocità del segnale da 1 a 999 parole al minuto (WPM).

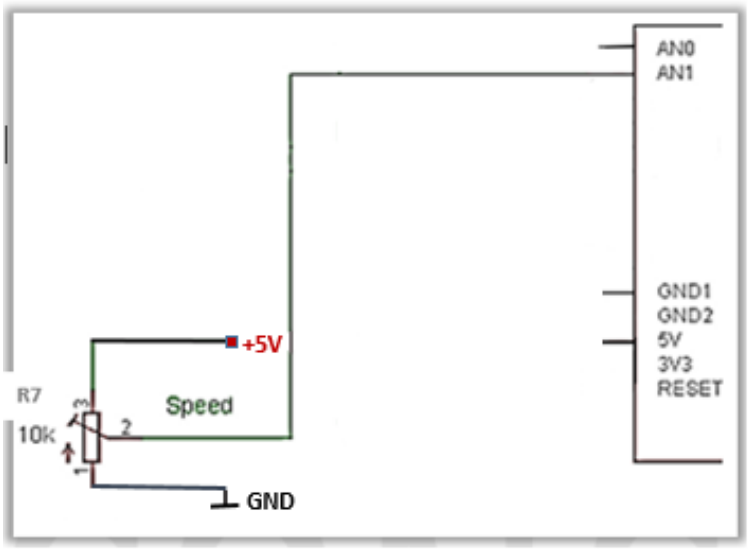


Figura 33, diagramma con potenziometro.

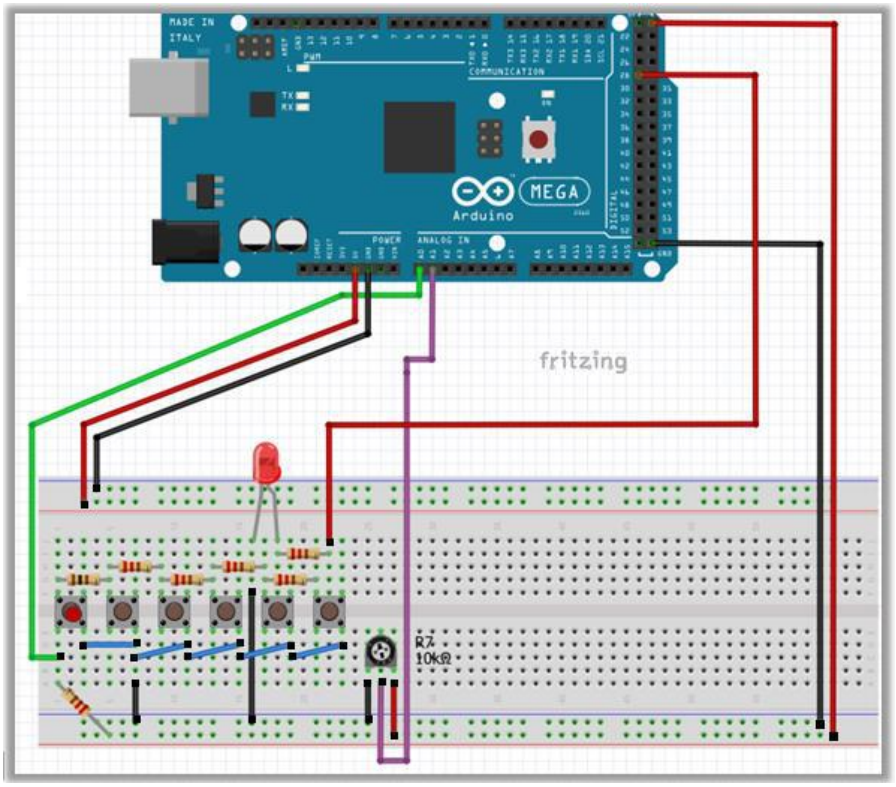


Figura 34, configurazione breadboard con potenziometro.

Affinché il potenziometro funzioni, è necessario eseguire le seguenti operazioni:

Selezionare keyer_pin_settings.h, che determina l'impostazione del pin. Poiché la porta analogica A0 è già occupata, scegliere la porta A1, fig.35.

keyer_pin_settings.h

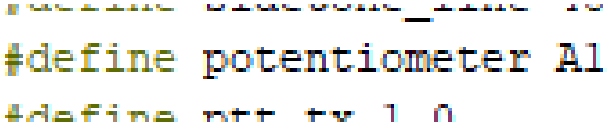


Figura 35, Porta analogica

Nel file keyer_features_and_options.h attivare i seguenti comandi, (// remove) fig.36/37

keyer_features_and_options.h

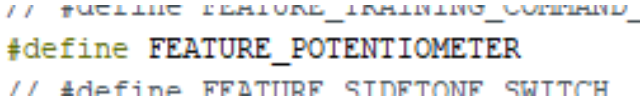


Figura 36, opzione potenziometro

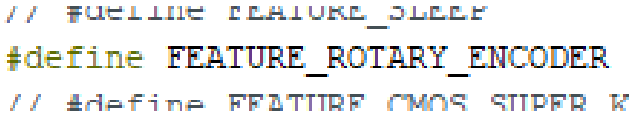


Figura 37, opzione Rotary Encoder

Compilando il programma alla fine di questo passaggio, il caricamento non ha senso qui.

Opmerking:

Quando compili per la prima volta, probabilmente visualizzerai il messaggio di seguito, fig.38.

Il testo rosso non ha alcun significato allarmante, il testo sottostante indica quanto spazio di archiviazione è stato utilizzato dello spazio di archiviazione totale disponibile.

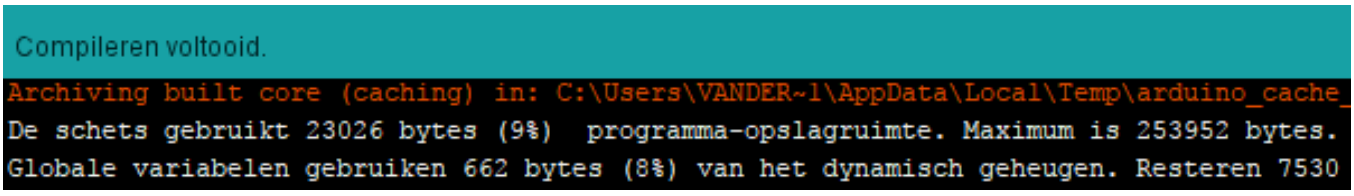


Figura 38, spazio di archiviazione utilizzato.

Passaggio 8.7: estensione chiave e altoparlante.

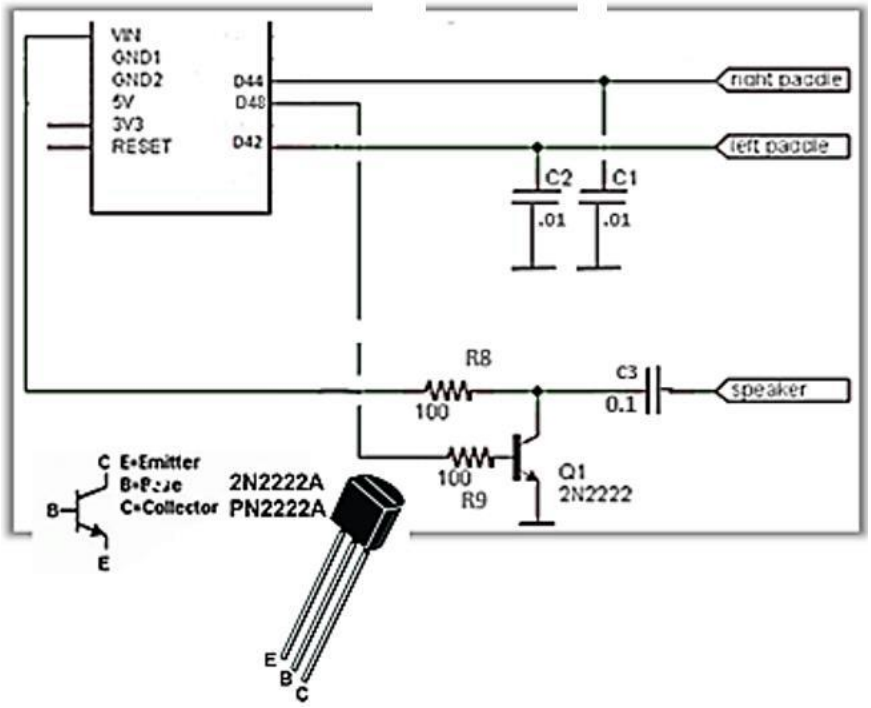


Figura 39, tasto e altoparlante

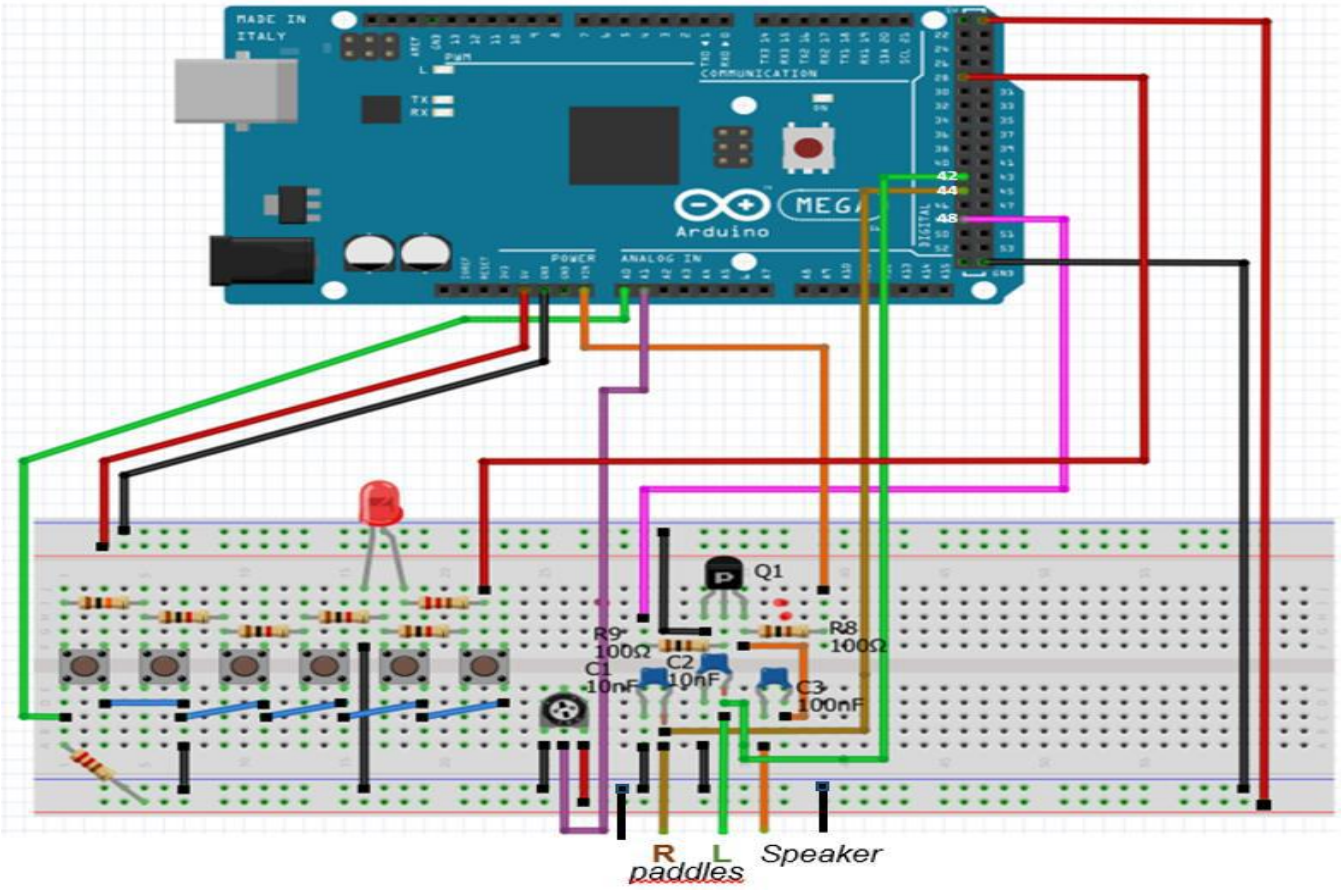


Figura 40, tasto di connessione, altoparlante

Con questo passaggio, figg.39 e 40, sarete in grado di convertire i caratteri della tastiera in segnali sonori utilizzando un piccolo altoparlante, forse di interesse solo per chi ha padronanza del CW.

Ma attenzione: nel passaggio 8.7 aggiungerai una tastiera con la quale i caratteri CW possono essere prodotti in modo udibile.

E diventa ancora più divertente se aggiungi un display nel passaggio 8.8 in cui puoi vedere cosa stai producendo in termini di caratteri CW.

Affinché i paddle dell'altoparlante e del keyer funzionino, non sono necessari comandi speciali nel file keyer_features_and_options.h.

Devi indicare nel file keyer_pin_settings.h quale impostazione pin scegli per il sidetone i paddle. Per i paddle sono stati selezionati l'altoparlante, pin 48, e pin 42 e 44, fig.41 e 41a.

Il transistor 2N2222 è un transistor di commutazione NPN standard e deve essere collegato secondo lo schema.

Le impostazioni dei pin chiave e sidetone, corrispondenti ai pin Arduino, sono:

keyer_pin_settings.h

```
#define sidetone_line 48          // connect a speaker for sidetone
```

Figura 41, impostazione pin tono laterale

keyer_pin_settings.h

```
#define paddle_left 42
#define paddle_right 44
```

Figura 41a, paddles

Passaggio 8.8: collegare la tastiera.

Per prima cosa aggiungerai una tastiera (connessione), con la quale puoi già produrre caratteri morse udibili, senza usare la chiave. La tastiera che utilizzo è del tipo PS/2, un tipo maneggevole e facilmente reperibile. In Fig. 42 è visibile la vista frontale della parte del telaio e accanto a quella di Fig. 43, dove devono essere collegati i relativi pin. Dopo il controllo, il test dopo la compilazione e il caricamento.

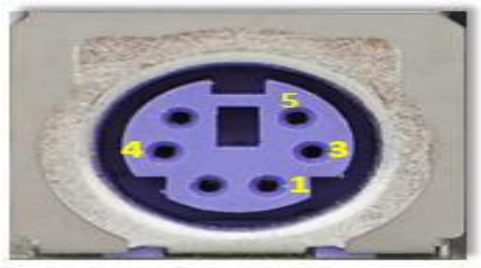


Figura 42, vista frontale della parte del telaio

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

Figura 43, designazioni delle penne.

Figura 46a, la voce di avviso

Collegare la tastiera è “un gioco da ragazzi” Secondo le figure 44 e 45 seguenti, attivare i seguenti comandi, la figura 46 mostra il cablaggio aggiunto.

file keyer_pin_settings.h: durata

```
//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
```

Figura 44, determinazione delle impostazioni dei pin.

keyer_features_and_options.h

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

Figura 45, Abilita feature_ps2_keyboard.

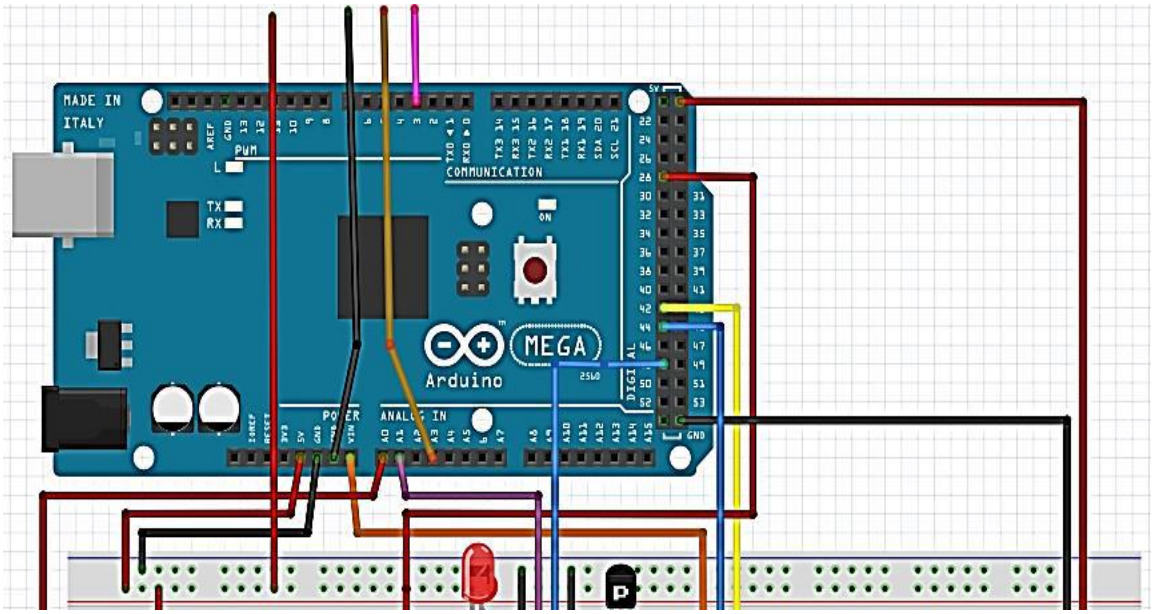


Figura 46, GND (nero), A3 (marrone) e 3 (rosa) e la connessione 5V (rosso)

Durante la compilazione verrà visualizzato il seguente avviso, Fig. 46a, relativo alla propria tastiera. Puoi ignorare questo avviso e procedere con il caricamento.

```
Compileren 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.
```

Figura 46a, la voce di avviso

Passaggio 8.9: collegamento del display.

L'uso di un display rende il progetto davvero divertente, perché ora puoi anche vedere ciò che produci con la tua tastiera, ed eventualmente la tua chiave.

Per rendere le cose un po' più chiare, ho usato una seconda breadboard che devi anche fornire con una connessione 5V e GND. Il display è un LCD a 4 bit a 2 righe. Successivamente, l'impostazione di decodifica può anche essere posizionata accanto alla 2a breadboard. Entrambe le breadboard vengono quindi ben riempite. Nei file keyer_pin_settings.h e keyer_featu-res_and_options.h, modificare quanto segue, fig. 47 e 48. Le figure 50 e 51 mostrano la configurazione della breadboard.

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
```

Figura 47, Impostazione LCD

keyer_features_and_options.h bestand:

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

Figura 48, Funzione LCD

Se si segue lo schema seguente, Fig. 49, il display dovrebbe funzionare correttamente.

Regolare il potenziometro R10 in modo che i caratteri diventino visibili.

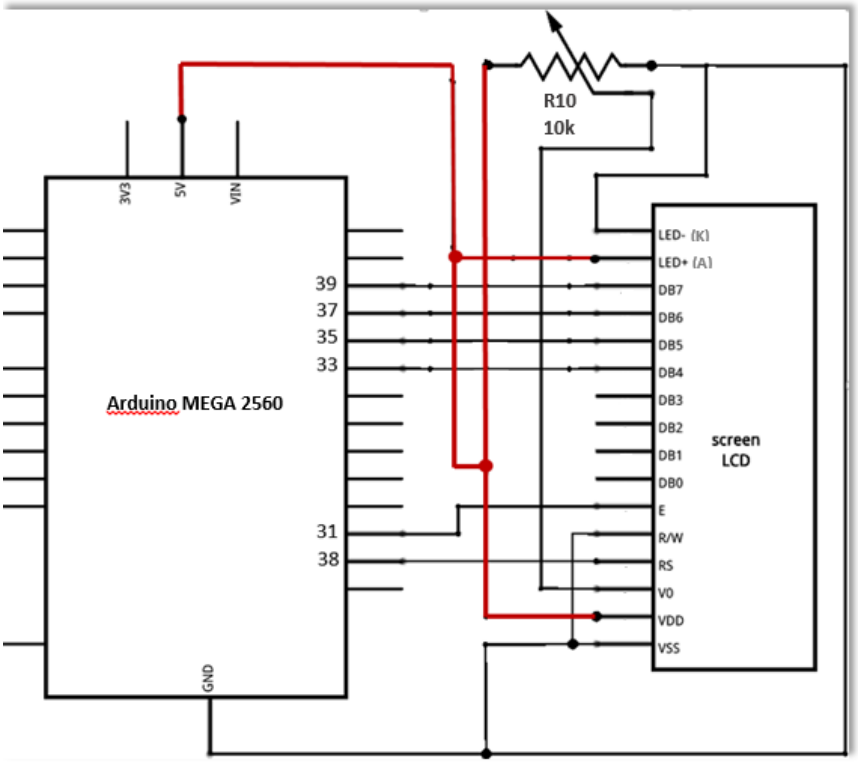
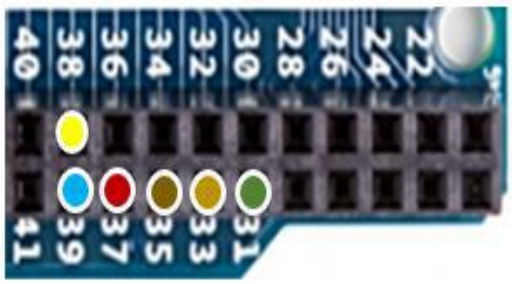


Figura 49, circuito LCD

Figura 50, impostazione del pin di dettaglio



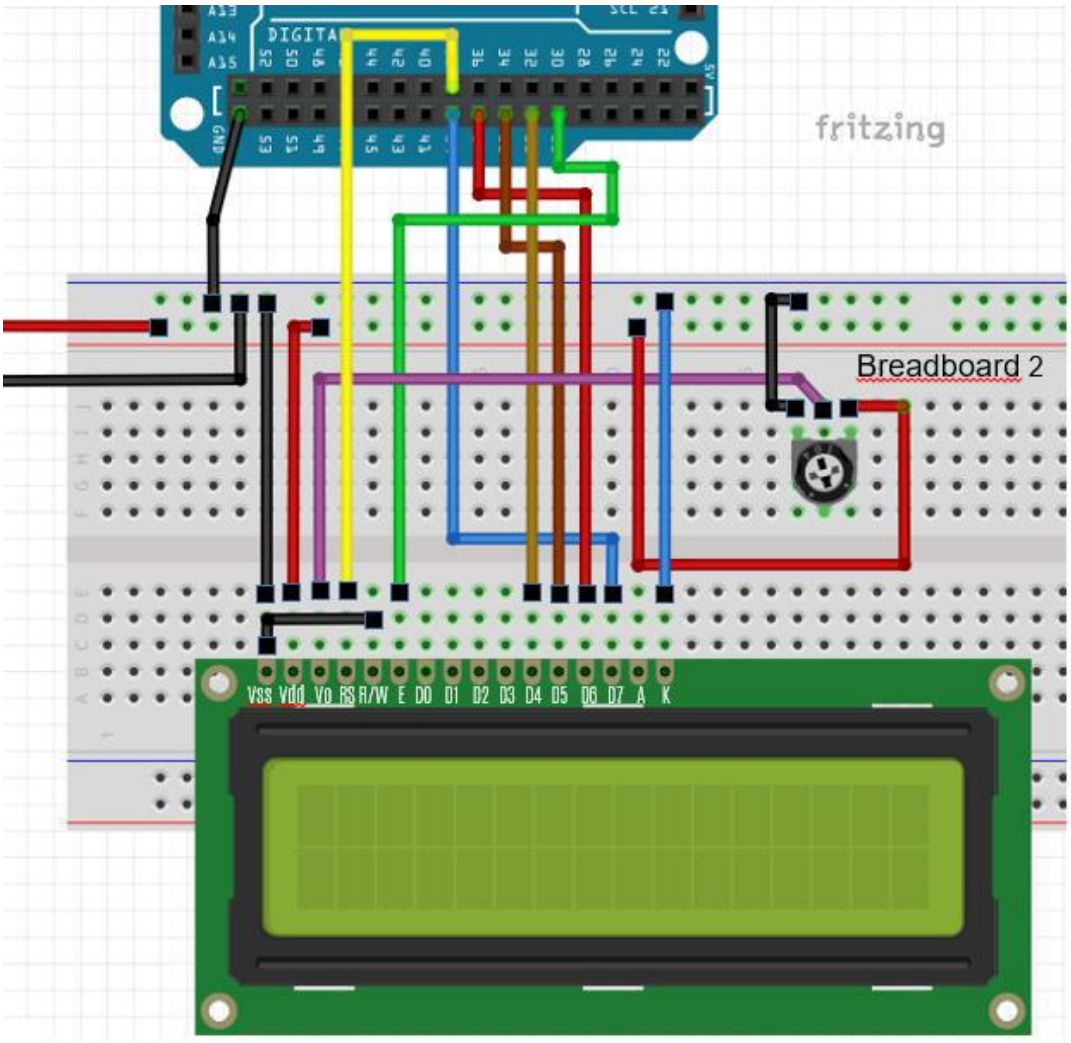


Figura 51, Configurazione LCD, LCD a 2 righe a 4 bit

Passaggio 8.10: la connessione al ricetrasmettitore.

Ora continua a costruire sulla tua prima breadboard.

Il design K3NG ha 3 connessioni TX, il che significa che puoi collegare 3 ricetrasmettitori, ad esempio un Kenwood, Icom e uno Yaesu.

Immagino che i tre ricetrasmettitori siano impostati su bande diverse in modo che se una banda va "chiusa", puoi continuare a contestare con l'altro ricetrasmettitore su un'altra banda. Forse un po' inverosimile, ma l'opportunità è offerta.

Per abilitare questa opzione è necessario modificare le porte per tx_key_line_1, tx_key_line_2 e tx_key_line_3 mostrate nel file keyer_pin_settings.h.

Ho messo il mio sui pin digitali 32, 34 e 36, fig.52.

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

Figura 52, le tre uscite TX attivate.

I CW-er possono ora "ticcare" il trasmettitore, per cui possono "leggere" il testo Morse di ritorno dalla stazione opposta a orecchio.

Con la tastiera puoi anche controllare il trasmettitore in modalità CW, ma i segnali Morse di ritorno non verranno decifrati per i "non CW", ma farai qualcosa al riguardo nel prossimo capitolo.

Le Figure 53, 54 e 55 mostrano lo schema e il cablaggio aggiunto sulla breadboard.

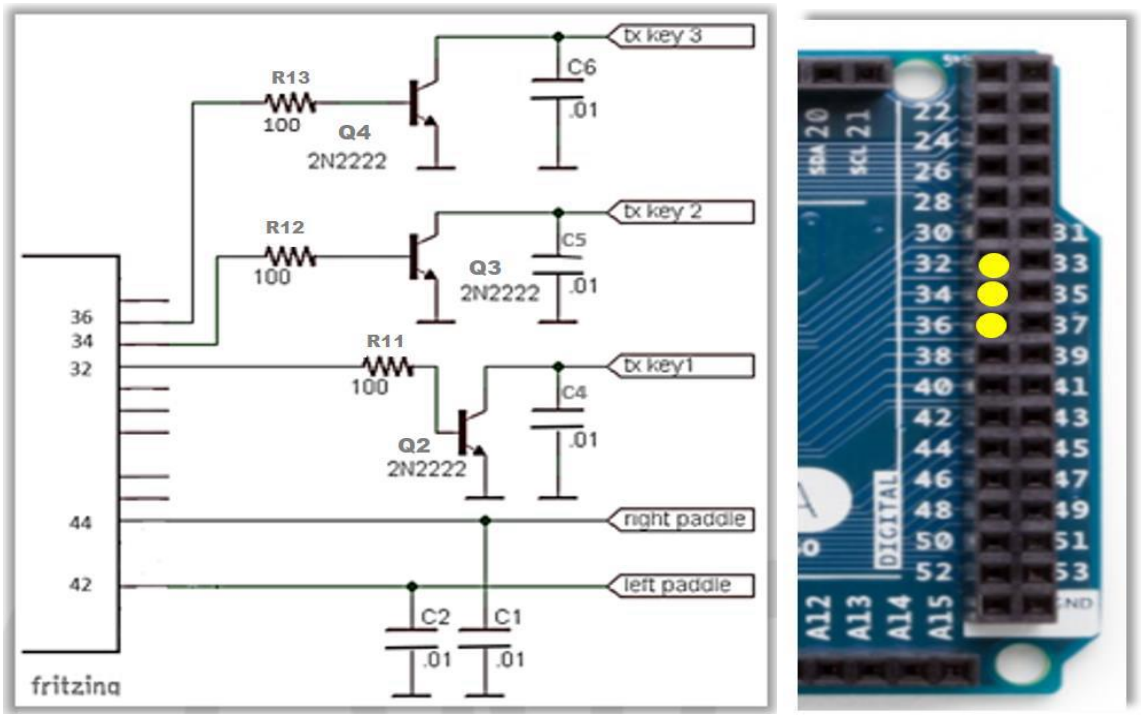


Figura 53, Schema di collegamento TX

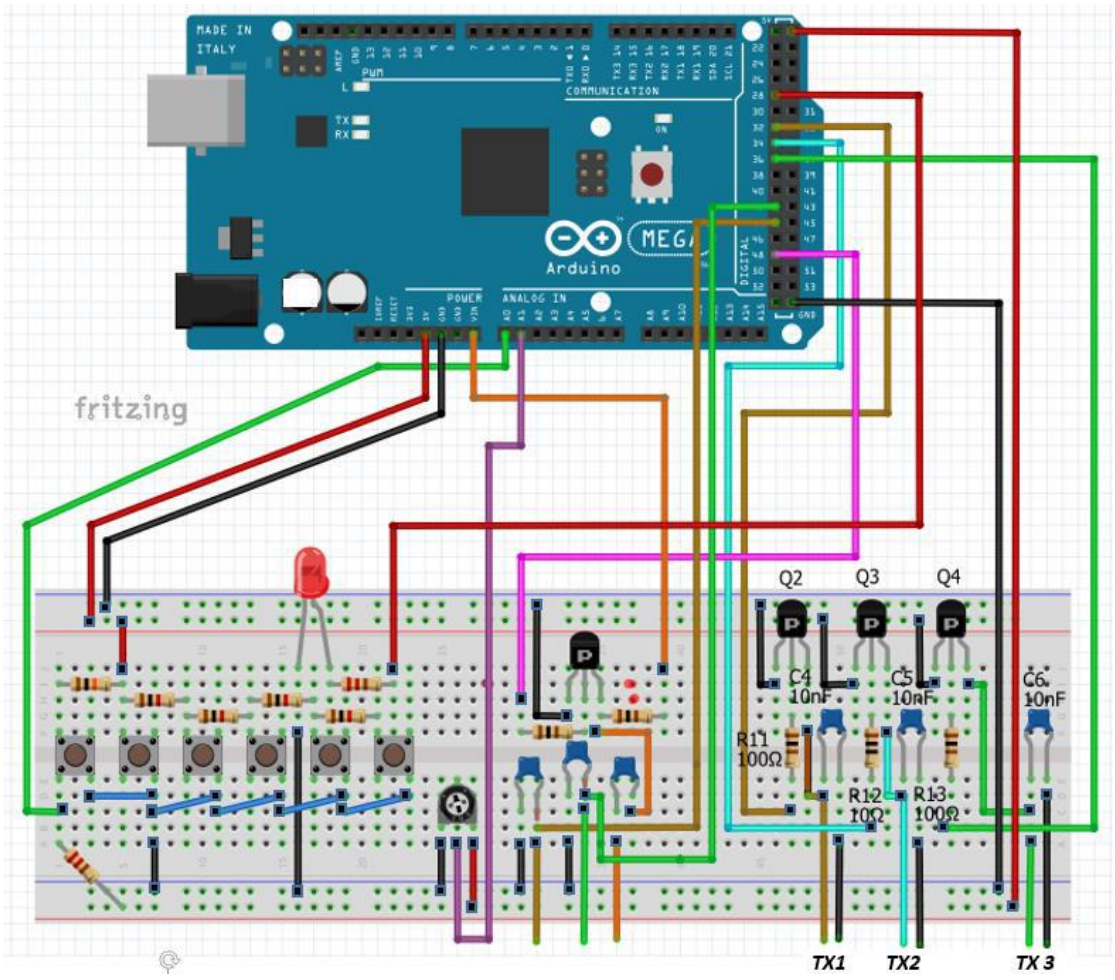


Figura 55, Uscite T

Passaggio 8.11: DECODER CW DSP GEORTZ

Il capitolo "chiave" è ora chiuso ed è ora di dare un'occhiata alla parte di decodifica che decodifica il codice Morse in testo normale sul tuo display.

Questa parte è ampiamente tratta dal sito KF4BZT, in cui descrive che deve avvenire anche una conversione dall'ONU al controller MEGA 2560. Utilizza il codice base di OZ1JHM che può essere trovato su questo sito:

http://www.skovholm.com/cwdecoder

La decodifica si basa sull'algoritmo di Goertzel e puoi trovare le informazioni su:

https://en.wikipedia.org/wiki/Goertzel_algorithm

L'intero evento è un evento matematico così complesso che non l'ho approfondito. Nella libreria è incluso un file per Arduino chiamato goertzel.h.

Questo si prende cura di tutti gli algoritmi di decodifica necessari affinché funzioni correttamente.

Per fare ciò, è necessario regolare una serie di impostazioni in modo che Arduino Mega possa gestire questo codice.

Questo file si trova nella cartella Goertzel, come mostrato in fig.56.

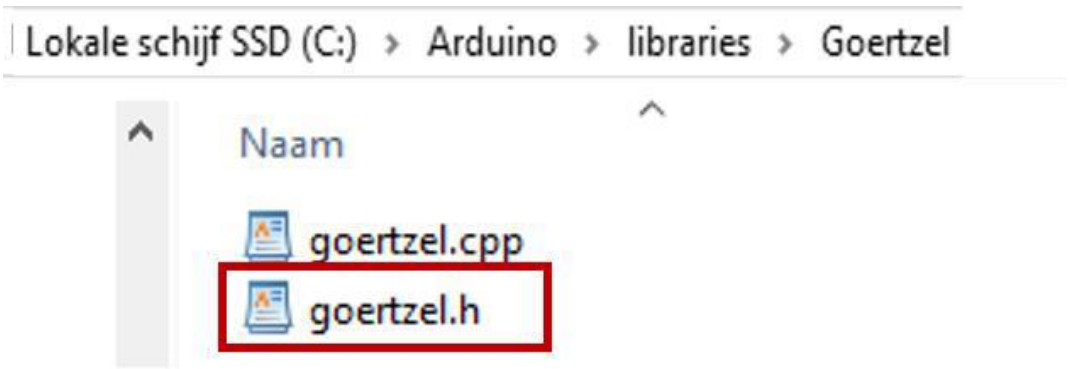


Figura 56, mappa Goertzel

Apri il file goertzel.h con ad esempio WordPad dove puoi trovare alcune informazioni interessanti sulla frequenza di campionamento e sulla larghezza di banda

Nella prima parte del file, stampata in grigio, troverai ulteriori spiegazioni da parte di OZ1JHM sulle impostazioni.

Ci sono impostazioni a cui prestare attenzione, se la decodifica qui non funziona correttamente.

Secondo OZ1JHM, la frequenza target dovrebbe funzionare a 558 Hz o forse 744 Hz.

Modificare alcune cose nel rettangolo rosso, fig.57 per rendere i codici adatti all'Arduino Mega.

Ma tieni presente che un valore GOERTZ_SAMPLES alto richiede molto tempo, quindi devi trovare un compromesso.

```
#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
  Hialmar 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
        init 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
```

Figura 57, Impostazioni di Goertz

Nel file keyer_pin_settings.h è necessario regolare il pinsetting. Scegli il pin A11, e per cw indicatore decoder pin 24, fig.58, 59 e 60.

Infine in keyer_features_and_options.h attivare l'opzione per rilevatore audio e cw_deco-der, fig.61.

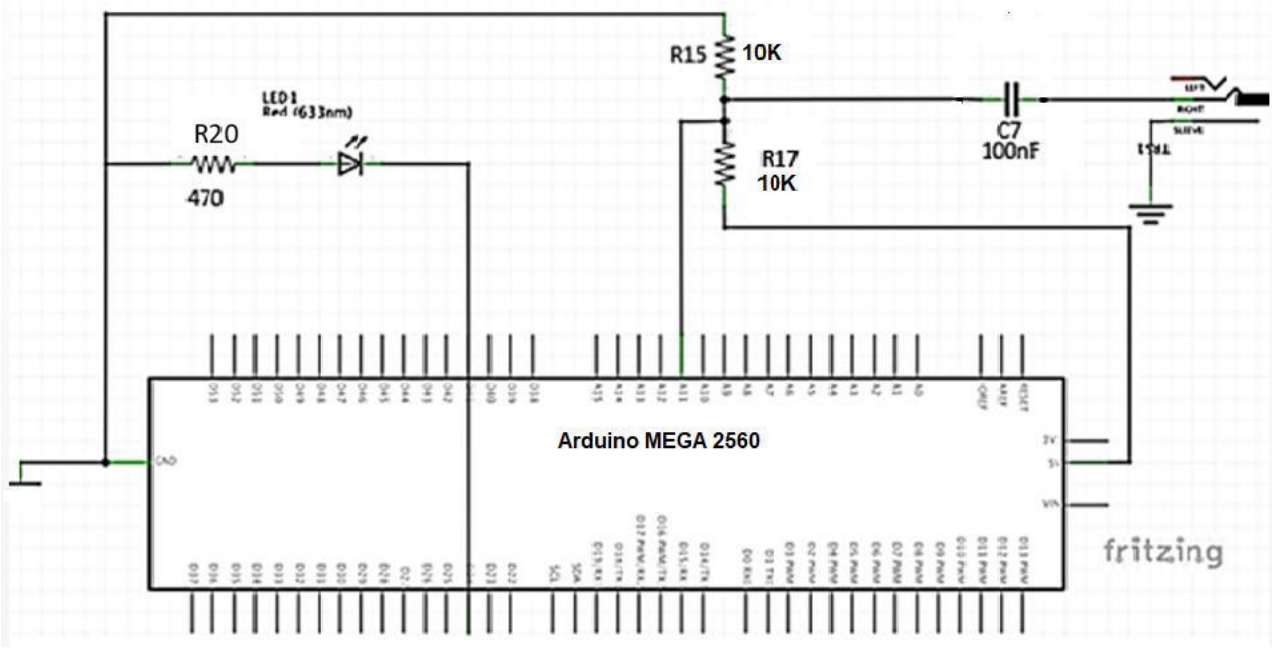


Figura 58, regolazione per il MEGA

Per definire cw_decoder_pin, scegli il numero “nul”.

keyer_pin_settings.h

```
#ifdef 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
```

Figura 59, impostazione dei pin per il 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
```

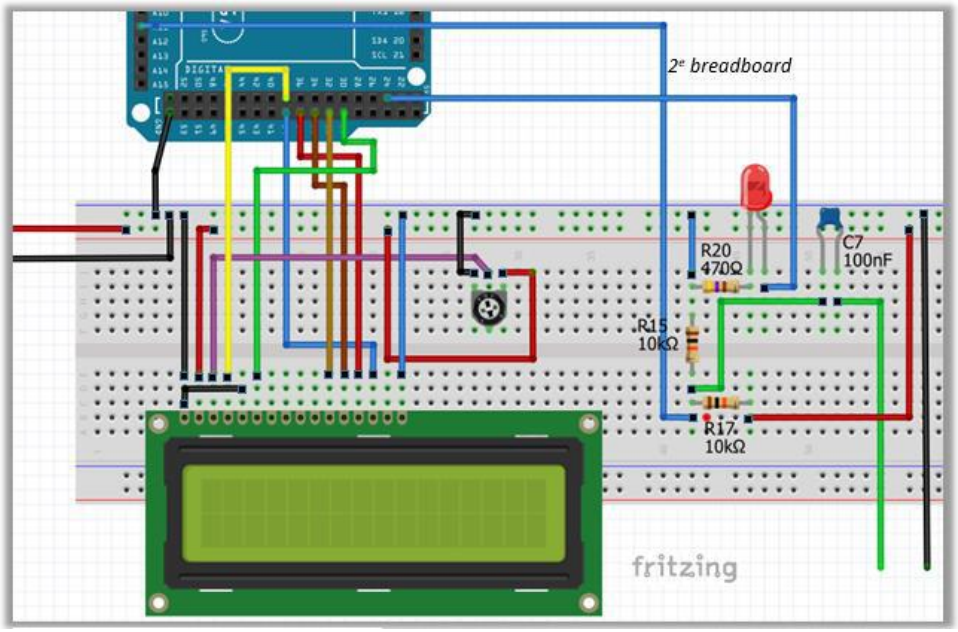
Figura 60, Rilevatore Goertzel Audio

```
#define FEATURE_CW_DECODER
```

Figura 61, feature_cw_decoder

NB. Puoi memorizzare questo circuito accanto al circuito di visualizzazione sulla tua seconda breadboard.

Il led serve per facilitare la sintonia sul segnale Morse e lo si collega ad A11, fig.58 e 62.



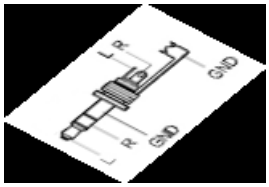


Figura 62, configurazione della decodifica, 2a breadboard

Nota:

Incontrerai il nome Fritzing in molti personaggi e potresti chiederti cosa significhi.

Fritzing è il nome di un programma che usi per progettare circuiti e arrangiamenti breadboard.

Se vuoi saperne di più, puoi scaricare il programma (freeware) dal sito web:

http://fritzing.org/download/

9. COME FUNZIONA IL DECODER

Sintonizzate il vostro ricevitore sulla parte CW della banda HF. Assicurati di aver collegato il LED tra la massa e il pin che hai scelto.

Sintonizzarsi su una stazione finché il LED non inizia a lampeggiare durante la ricezione in CW. È necessario sintonizzarsi per renderlo leggibile in modo che il LED veda che vengono inviati i dati corretti.

Gioca con le impostazioni di cui sopra nel file goertzel.h per vedere se riesci a ottenere una decodifica migliore. L'impostazione della larghezza di banda del ricetrasmettitore è (modalità CW) 1000 Hz, la decodifica è quindi ottimale per me.

NB. In tutta onestà, devo dire che non dovresti aspettarti troppo dalla decodifica. Il metodo di decodifica è ancora in una fase sperimentale. Per l'operatore in CW alle prime armi, la suddetta decodifica sul display è molto piacevole da seguire, soprattutto se c'è un segnale regolare "pulito" non disturbato dai segnali CW vicini.

Potresti anche provare a sintonizzarti sulle stazioni radiofaro, che trasmettono un piacevole script di segnale regolare.

Ulteriori informazioni sul funzionamento e informazioni aggiuntive del CW-keyer possono essere trovate sul sito:

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

Il gruppo qui sotto è molto attivo e disponibile, puoi porre tutte le tue domande sul

Keyer K3NG-CW e altri progetti K3NG.

Siti aggiuntivi utili:

https://github.com/k3ng/k3ng_cw_keyer

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

https://www.arduino.cc/en/Principale/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. Comandi da tastiera
Comandi speciali della tastiera PS2

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. Elenco dei componenti
Componenti richiesti:

elemento	**nome, valori**	**numero di**
Arduino MEGA, 2560	1x	
Breadboard	2 x	
Tastiera PS/2	1x	
S 1,2,3,4,5,6	SU / a partire dal premi i pulsanti, 6x6 mm.	6 x
fili di collegamento	carichi di	
Led 1, 2	Rood (633nm)	2x
R1,11,15,17	10 kΩ	4x
R2,3,4,5,6	1 kΩ	5x
R19, 20	470 Ω	2x
R7,10	10 kΩ, pot. meter	2x
R 8,9,11,12,13,16	100 Ω	6x
C1,2,4,5,6	0.01 µF (10nF)	5x
C3,C7	0.1 µF (100nF	2x
Q 1,2,3,4	2N2222A (NPN)	4x

12. Allegati

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
```

```
#ifdef 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
#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
// rotary encoder pins and options - rotary encoder code from Jim Balls M0CKE
#ifdef 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
#ifdef 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
#ifdef FEATURE_ALPHABET_SEND_PRACTICE
#define correct_answer_led 0
#define wrong_answer_led 0
#endif //FEATURE_ALPHABET_SEND_PRACTICE
#ifdef FEATURE_PTT_INTERLOCK
#define ptt_interlock 0 // this pin disables PTT and TX KEY
#endif //FEATURE_PTT_INTERLOCK
#ifdef FEATURE_STRAIGHT_KEY
#define pin_straight_key 52
#endif //FEATURE_STRAIGHT_KEY
#ifdef 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
#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 capactive_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 FEA-
TURE_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_WORDSPACE_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. À, Å, Þ, 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
```