En intressant CW keyer - del 2

Den behändiga lilla CW-keyern *Open CW keyer Mk2* och dess användning beskrevs i del 1 (QTC 4/2024). Den är bra för t.ex CW-paddle-träning och portabelbruk, och kan kopplas till vilken CW-sändare som helst.

I denna del tittar vi under skalet, både hårdvarumässigt och dess programvara, vilket öppnar för nya möjligheter, tack vare den flexibilitet som Andrew K3NG byggt in i sin öppna CW-programvara på Arduino [1]. Arduino är billiga små mikrodatorer. Det sitter en liten Arduino Nano som hjärnan inuti Open CW keyer Mk2.

Jag kan nog utlova att denna del blir lärorik för den som inte stött på Arduino tidigare.

Text: Poul Kongstad SA7CND

Bakgrund

Man kan lätt bygga eller för en billig penning köpa denna lilla CW-keyer som bygger på öppen gratis programvara, *Open CW keyer Mk2*. Keyern formar utsökt fin CW och har 2-12 CW-minnen. Man sänder med manipulator (”paddles”) och har god nytta av ”jambisk nyckling” (Iambic, se del 1) för att komma upp i hastighet efter en del träning.

I mitt fall är keyern tänkt för portabelt bruk eftersom min portabelrig IC706 saknar CW-minnen (men har keyer iofs). En liten 5 volt mobil powerbank räcker att driva keyern länge.

Jag har gjort en modifiering med fler CW-minnen, och för det krävs att man konfigurerar om programvaran något i keyern [2]. Men först tittar vi hur keyern är uppbyggd.

Bild 1 - Min modifierade Open CW keyer Mk2 *med knappar för fler CW-minnen.*

Elektroniken

Det ska sägas med det samma: Open CW keyer Mk2 innehåller en liten krets **Arduino Nano** (ATmega328), men den räcker. Minnesutrymmet är klart begränsat, 32 kbyte programminne, 2 kbyte arbetsminne och 1 kbyte EEPROM minne som inte försvinner när spänningen slås från. Se bild 2.
Se även separat faktaruta om en större Arduinomodell Mega och vilka fler keyer-funktioner som då blir möjliga.

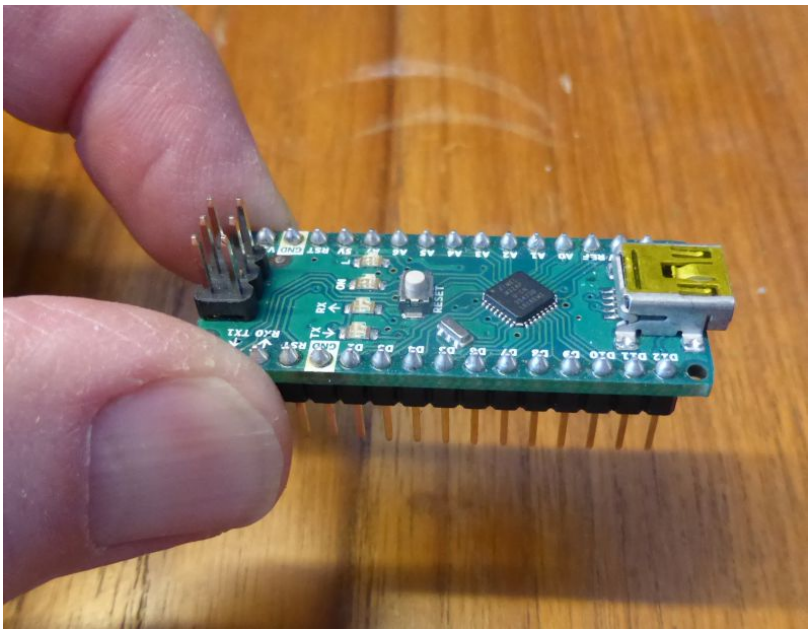


Bild 2. Denna lilla Arduino Nano 47x18 mm är ”hjärnan” i Open CW keyer Mk2.

En arduinokrets har många digitala och analoga *in- och utgångar* som kan användas för styrning och integration med annan hårdvara. Det finns även tilläggskort som kan anslutas till dessa stift. Med dessa in- och utgångar kan man från programvara:

- Digitala ingångar: läsa och status till/från, omkopplare m m
- Digitala utgångar: styra reläer, indikeringar osv. Pulsbreddsmodulering (PWM) för positionering och varvtal osv är även tillgängligt på 6 av dessa.
- Analoga ingångar: läsa av värden och spänningar i apparater.

Normalt ligger spänningsnivåerna för dessa in- och utgångar på 0 - 5 volt. Kretsen kan matas med 5-20 volt. Det ska sägas att en arduinokrets har många fler funktioner, t.ex serieport och TWI (two-wire interface) bus för 3-tråds förbindelse mellan flera yttre enheter (I2C).

Vår Arduino Nano har 14 digitala in/utgångar och 8 analoga ingångar. Analoga ingångar används i CW-keyern för att läsa av potentiometern för CW-hastighet och en för alla minnesknapparna.

Open CW keyer Mk2 **schema** [3] av Ondrej OK1CDJ är enkelt (se Bild **3**):

- Uppe till höger: SET-knappen för CW-kommandon och 2 knappar (S2, S3) för CW-minnen
- Nere till höger: Optokopplade utgångar till riggen (key, PTT)
- För övrigt ses Arduino-kretsen och en summer till vänster, samt kontakter och HF-avkoppling. JP1 är uttaget på baksidan som vid bygling spärrar mot omprogrammering (auto reset).

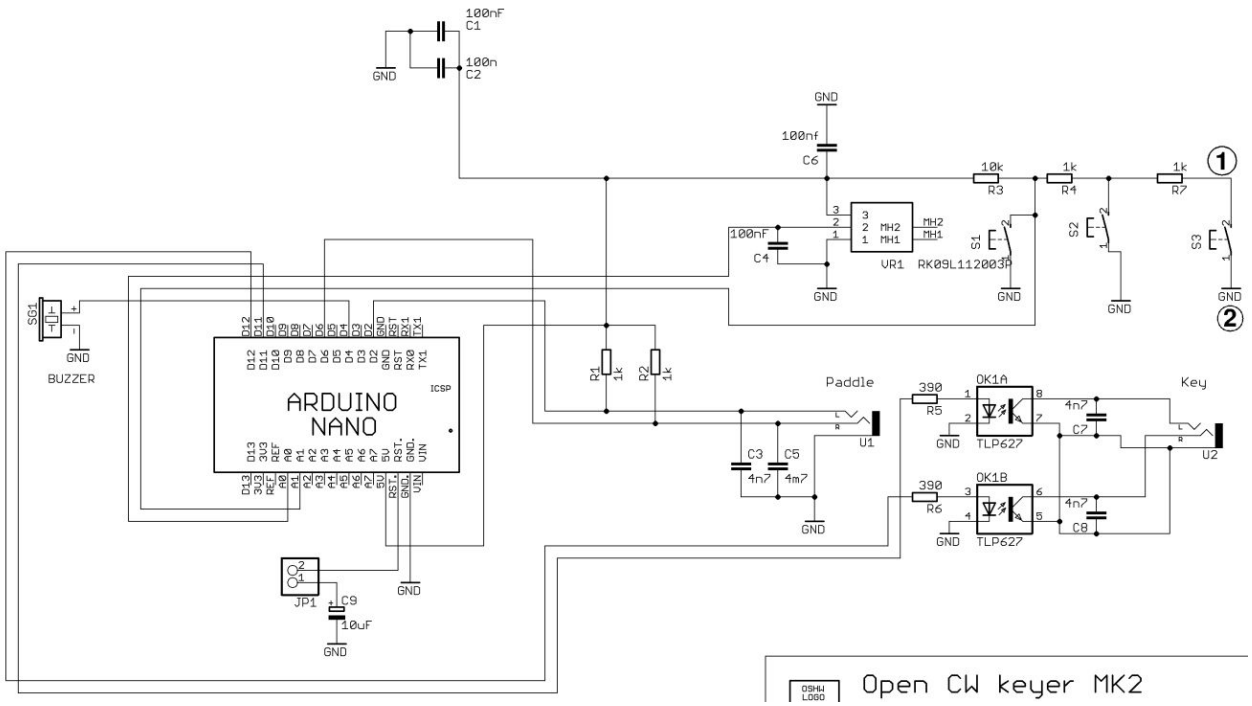


Bild 3. Kopplingsschema för Open CW keyer Mk2 - inte många komponenter behövs.
1 och 2 är anslutningspunkter för att bygga vidare med extra minnesknappar med sina motstånd.

Programvaran

Det är rätt fantastiskt att Andrew K3NG gjort dels programkod för Arduino mikrodatorer med *många* CW-funktioner och dels ett antal förberedda konfigurationer för olika hårdvaror och kombinationer av funktioner.

En av dessa konfigurationer är just *Open CW keyer Mk2*!

Med Arduino IDE (utvecklingsmiljö) kan man konfigurera om vissa funktioner i denna Arduino Nano så att man *i stället för* funktionen ”CW-kommandon” får t.ex:

- Tillgång till kraftfullare kommandon (CLI, mycket användbart) via serieport-program som Putty
- Fler CW-minnen
- Tillgång till CW-macros för CW-minnena för lite mer avancerade morsefunktioner
- Med Putty i datorn kan keyern sända morse från tangentbordet till sändaren, ofta inkl ÅÄÖ.

Plan: Efter installation av Arduino IDE (utvecklingsmiljö, gratis) ska vi nu göra följande steg:

1. Lägga till 4 extra knappar för CW-minnen
2. Lägga in kommandogränssnitt (CLI) och macros för CW-minnen men då får vi ”offra” CW-kommandon och Winkeyer-körning pga minnesbegränsningar.

   Anm:

   a. *CW-kommandon är dock bra vid portabelkörning om man behöver ändra något CW-minnesinnehåll eller medhörning på plats utan dator med Putty.*

b. *Se separat faktaruta om du vill* spara *ursprungskonfigurationen som Open CW keyer Mk2 levererades med, innan du ändrar något i keyern.*

Arbetsgången med Arduino utvecklingsmiljö för Open CW keyer:

1. Gör först installation av Arduino IDE. Ladda ner installationsfilen för ditt operativsystem från [4] och följ installationsanvisningarna där.
2. Anslut din Open CW keyer Mk2 med USB-kabeln till datorn.
   Windows kan behöva drivrutin CH340 (341), kolla COM-port i Enhetshanteraren → Portar.
   På Linux brukar jag få ta ut USB-kabeln och sätta i den igen för /dev/ttyUSB0 ska ansluta
3. Skapa en mapp Arduino t.ex under dina Dokument på datorn.
4. Ladda ner och spara K3NG programpaket. Hämta paketet gratis:
   `https://github.com/k3ng/k3ng_cw_keyer/archive/refs/heads/master.zip`
   Öppna zip-filen k3ng_cw_keyer-master.zip.
   Kopiera ut följande från k3ng_cw_keyer-master till mappen Arduino:
   k3ng_keyer och *libraries* (med överskrivningar).
5. Starta Arduino IDE. Ställ in följande i menyerna:
   - Files - Settings: Scetchbook Location: ange din mapp Arduino enligt ovan
   - Tools - Board: Arduino Nano
   - Tools - Processor: ATmega328P (Old Bootloader)
   - Tools - Port: den COM-port som keyern är ansluten till, t.ex COM5 eller /dev/ttyUSB0
   - File - Open: k3ng_keyer → k3ng_keyer.ino.

Nu kan vi konfigurera om Open CW keyer som vi önskar. Följande steg förbereder alla modifieringarna.

6. Fortsätt i Arduino IDE:
   Öppna: keyer_settings_opencwkeyer_mk2.h
   Tips: det finns en liten pil uppe till höger varifrån alla filer kan öppnas, rulla med mushjul.
   Ändra
   `#define analog_buttons_number_of_buttons 3` till
   `#define analog_buttons_number_of_buttons 7`
   Det kan vara bra att markera dina ändringsrader med kommentar, t.ex *//dittCall*
   Save (Ctrl-s)
7. Öppna: keyer_hardware.h
   Välj Open CW keyer Mk2 genom att ta bort kommentarstecken //:
   `#define HARDWARE_OPENCWKEYER_MK2`
   Save (Ctrl-s)
8. Öppna: keyer_features_and_options_opencwkeyer_mk2.h
   Ändra följande rader till:
   `// #define FEATURE_COMMAND_MODE //CW commands`
   `#define FEATURE_COMMAND_LINE_INTERFACE //CLI commands`

```
#define FEATURE_MEMORY_MACROS
// #define FEATURE_WINKEY_EMULATION  //spar minne
#define FEATURE_SIDETONE_NEWTONE  //spar minne
#define FEATURE_AUTOSPACE
#define OPTION_NON_ENGLISH_EXTENSIONS
#define OPTION_PROSIGN_SUPPORT
```

Save (Ctrl-s).

9. Provkompilera för att se om denna konfiguration rymms i Arduino Nano minne:
   Sketch - Verify/Compile (Ctrl-R).
   Det ska rapporteras med vit text att hela minnet inte är förbrukat (< 100%)
10. Kontrollera att PRG på keyerns baksida inte är byglad.
    Ladda upp den nya konfigurationen till Open CW keyer:
    Sketch - Upload (Ctrl-U)
    När det är klart svarar keyern med ”HI”.

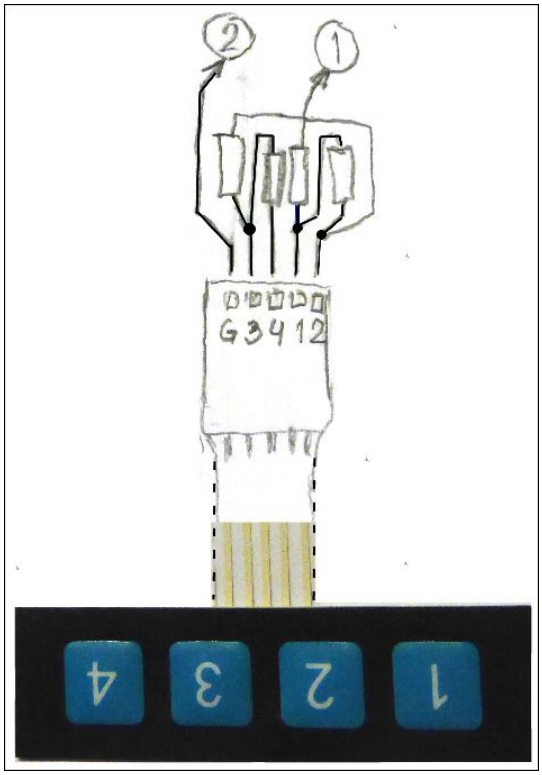
Nu är keyern redo för modifieringarna nedan, och i stället för CW-kommandon och Winkey har vi nu kommandogränssnitt CLI och CW minnes-macros i denna konfiguration.

1. En modifiering: fler knappar för CW-minnen

Keyern har 10 användbara CW-minnen, och kompletteras nu med en tryckknapps-sats med 4 knappar [5] för CW-minne 3-6. (Resterande CW-minnen kan nås med kommandotolk CLI och macros, se nedan).

Utbyggnaden med fler knappar görs genom att ”förlänga” kedjan med 1kΩ motstånd (R4, R7) och knappar (S2, S3) i schemat bild 3 med fler 1kΩ motstånd och knappar i en kedja enligt bild 4-5.

Nu är vi inne på en lite känslig sak - hur fult jag bygger… Så här gjorde jag, och det fungerar (än). En av varje motståndsledare fick sticka in i kontakten. Använd krympslang som isolering där det går. Vit tejp runtom. In i burken - glöm.



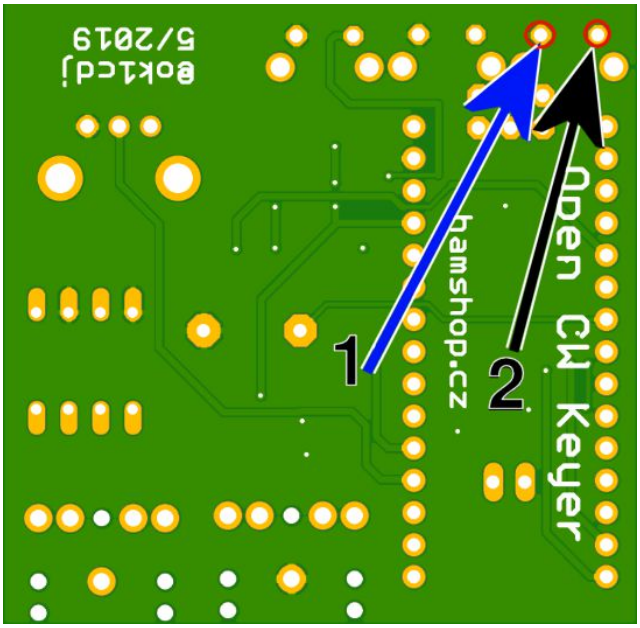


Bild 4 t.v. Jag lyckades tränga ihop de 4 små 1kΩ motstånden på knappsatsens kontakt.
1 och 2 går till schemats motsvarande punkter (bild 3), G är gemensam jordning.
Bild 5 t.h. Undersidan på Open CW keyer Mk2 kretskort. Den extra motståndskedjan och knapparna ansluts till 1 (till R7 högersida) och 2 (jord) med en tunn lång ledning bakom fronten.

”Tangentbordet” kan klippas ner lite med sax så att det passar på lådan. Tangentbordets flatkabel kan tas ut genom att fila ner 1 mm på en bit av bakplanet. Se bild 2 i del 1 av artikelserien.

Tryckknapparna [5] har en självhäftande undersida, så när man pillat in kablaget i lådan, kan de fästas på lådans ovansida. Dra av skyddstejpen på undersidan och klistra fast på lådan. Märk knapparna t.ex 3, 4, 5, 6 och skruva ihop lådan.

Nu är det dags att prova alla minnena. CW-minnen och annat programmeras nu endast från kommandon (CLI nedan). Men hastighetsratten gäller fortfarande.

2. CW keying från tangentbord

Med de ändringar i konfiguration vi gjode i K3NG CW keyer kan vi nu prova ett antal nya saker.

Man kan i Open CW keyer Mk2 nu sända CW från tangentbord genom seriekommunikation med t.ex Putty [6] i datorn via USB-kabeln, ofta även med svenska CW-tecken. I Windows behövs drivrutin ”CH340”.

Det du sänder visas på skärmen. Du får ”type-ahead”, dvs kan skriva fortare än keyern sänder.

Förslag till Putty-inställningar:
- Session
  Windows: COMx 115200 Serial (ange com-port nr x för keyern, se Enhetshanteraren-Portar)
  Linux: /dev/ttyUSB0 115200 Serial (se terminalkommando dmesg)

- Keyboard: VT100+
- Window: 180 x 77
- Translation: ISO-8859-1
- Serial Data bits 8, Stop bits 1, Parity None, Flow control None.

Man kan gå mellan paddles, minnen och tangentbord för att sända telegrafi effektivt.

Vissa trafiktecken får man med dessa tecken från tangentbordet:

= Åtskillnadstecken	**>** SK, Klart slut (QSO)	**(** KN Enbart Call svara
+ Avslutningstecken	**&** Vänta (15+ sek)	

3. Köra med CLI (kommandotolk)

En annan av ändringarna ovan är att vi (av platsbrist i Nano-minnet) skalat bort CW-kommandon men i stället fått tillgång till ett kommandospråk (CLI) för att konfigurera keyern och dess minnen. Dessa kommandon används typiskt hemma innan man ger sig ut portabelt. Tyvärr ryms inte både CW-kommandon och CLI samtidigt i Arduino Nano minne, bara en åt gången.

CLI-kommandon skrivs med små bokstäver. SET-knappen används då inte, utan varje kommando inleds i stället med \ backslash.

Med kommando **\s** visas en **sammanställning** av inställningarna.

Användbara **CLI-kommandon**:

\f###	Medhörning frekvens Hz	Ex: 600
\o	Medhörning av/på (toggle)	
\a \b	Iambic-A resp. iambic-B nyckling	(när paddlarna trycks ihop, se del 1)
\i	PTT enable / disable	Kvittens: *bipp* aktiverad, *bopp* ej aktiverad.
\w##	Sätt CW-hastighet (## WPM)	
\e#	Sätt contest serienummer till #	För ändringar. 001-ställs vid uppstart
\<	Sänd serienummer utan uppräkn.	**\(** Sänd förkortat serienummer utan uppr.
\>	Sänd serienummer och räkna upp	**\)** Sänd serienummer förkortat. Räkna upp
\1	Sänd CW-minne nr 1	CW-minne nr 0 - 9
\p1	Programmera CW-minne 1 (ex)	# = 0-9. (minne 10 nås med 0 i CLI)
|####	Tid mellan repeterade CW-minnen	Millisek. Håll minnesknapp + vä. paddle
\!##	Börja repetera CW-minne nr ##	
\\	*Avbryt* pågående sändning direkt	

4. Macros i CW-minnena

I CW-minnen kan nu även macros användas. De är små funktioner som enbart läggs in i CW-minne från CLI-kommandon.

Ett exempel är serienummer som en del av signalrapporten vid contest.

Användbara **macros för CW-minnen**:

\i1#	Kopiera in minne nr # här (i minne 1)	# = 1 - 9, 0 = 10
\#	Hoppa över till minne nr # och spela upp därifrån	

\e	Sänd serienummer som siffror	Räknar sen upp
\c	Sänd serienummer med förkortade siffror	Räknar sen upp
\n	Räkna ner serienummer utan att sända det	För repetition nr
\u \v	Aktivera PTT alt. Deaktivera PTT	
\w##	Sätt CW normalhastighet till ## WPM	
\y#	Öka CW-hastighet med # WPM	T.ex i rapport 5NN
\z#	Minska CW-hastighet med # WPM	
\+	Följande två tecken bildar ett Prosign (sammansatt tecken)	Ex: **\+**VE (Verified)

Tabellen ovan var alltså macros till CW-minnen, inte CLI-kommandon till keyern.

Exempel 1: Lägg in serienummer i CW-minne; det räknas upp automatiskt varje gång det sänts:

- **p**35NN **\e \n\e** (sänder samma serienr två gånger med nerräkning emellan)
  eller om 5NN ska sändas fortare + kortare serienr: **\p**3**\y9**5NN**\z9 \c \n\c**

Exempel 2: Keyern kan ha fler minnen även om de inte kan aktiveras direkt med en knapp.
Exempel:

- **p**9 ditt call, ex: \p9SA7CND
- **p**2CQ CQ CQ de **\i9 \i9** CQ K (hämtar ”dolt” minne 9)

Avslutning

Artikelseriens del 1 behandlade användning av denna lilla Open CW keyer Mk2 i originalutförande, ”out of the box”..

I denna del 2 fick du fler tekniska funktioner i denna lilla CW-keyer som baseras på en Arduino Nano. K3NG programpaket för Open CW keyer kan anpassas för olika hårdvaror och med olika funktioner genom att ändra några rader i K3NG-konfigurationen genom Arduino utvecklingsmiljö (IDE).

Skulle det visa sig att man föredrar CW-kommandon (som endast kräver paddles) framför CLI-kommandon (som fordrar dator/serieprogram) kan man ändra tillbaks i Arduino:
i keyer_features_and_options_opencwkeyer_mk2.h, ändra till (t.ex för mobilt bruk):
#`FEATURE_COMMAND_MODE`
//#`FEATURE_COMMAND_LINE_INTERFACE`
och sen ladda om programvaran.

Eller så vill du kanske prova andra funktioner i K3NG Open CW keyer’s utbud.

Som vanligt, lycka till och 73 de Poul, sa7cnd@ssa.se **.-.-.**

Referenser

1. Orientering av K3NG keyer funktioner: https://blog.radioartisan.com/arduino-cw-keyer/
2. K3NG Arduino CW keyer dokumentation: https://github.com/k3ng/k3ng_cw_keyer/wiki/

3. OK1CDJ schema och kretskort: https://github.com/ok1cdj/OpenCWKeyerMK2
4. Arduino utv.miljö: https://www.arduino.cc/en/software
   Man får gärna skicka ett bidrag och ange e-post. Annars välj ”Just Download”.
5. Komponenter:
   1 st Membran-tangentbord 4-knappar:
   https://www.electrokit.com/produkt/tangentbord-membran-1x4/
   4 st små metallfilmsmotstånd:
   https://www.electrokit.com/produkt/motstand-metallfilm-0-125w-1-1kohm-1k/
6. Seriekommunikationsprogram PuTTY, Download PuTTY: https://putty.org/

Ordlista

★ CLI - Command Language Interface - möjlighet att köra textkommandon direkt

★ PuTTY - mångsidigt och beprövat kommunikationsprogram för seriekommunikation, SSH, telnet m m. Finns gratis på https://putty.org och Microsoft app store

★ WinKeyer - extern hårdvara (USB-sticka med CW-key-kabel), en morse keyer som styrs från dator med kommandon och som formar morsetecken från text.

Faktaruta

A. Spara och återställ konfiguration

Innan du laddar ner ny programvara till OpenCWkeyer_Mk2, kan du spara dess programvara och inställningar vid leverans. Detta görs med kommandot avrdude. Du kan behöva trycka på SET-knappen när du ansluter USB-kabeln mellan keyern och datorn. Man kan på Linux kolla att keyern anslutits till /dev/ttyUSB0 med kommando dmesg. På Windows kan drivrutin CH340 behövas, kolla COM-port i Enhetshanteraren.

Programmet avrdude finns gratis på github: `https://github.com/avrdudes/avrdude`

Kommandon för att spara programvaran och inställningarna för OpenCWkeyer_Mk2.
På Windows ersätts /dev/ttyUSB0 med COMx (com-port).

```
avrdude -c arduino -p atmega328p -P /dev/ttyUSB0 -b 57600 -U
flash:r:downloaded_code.hex:i -v
```

```
avrdude -c arduino -p atmega328p -P /dev/ttyUSB0 -b 57600 -U
eeprom:r:downloaded_eeprom.hex:i -v
```

Återställ den sparade programvaran och inställningarna till OpenCWkeyer_Mk2:

```
avrdude -c arduino -p atmega328p -P /dev/ttyUSB0 -b 57600 -U
flash:w:downloaded_code.hex -v
```

```
avrdude -c arduino -p atmega328p -P /dev/ttyUSB0 -b 57600 -U
eeprom:w:downloaded_eeprom.hex -v
```

B. Arduino mikrocontrollers

Fler funktioner i en K3NG Open keyer är möjliga med en större krets **Arduino Mega** (ATmega2560, 110x53 mm) med 256 kbyte programminne, 8 kbyte arbetsminne och 4 kbyte EEPROM. En sådan inklusive anslutningskort är t.ex ”AZDelivery AZ-MEGA2560-Board Bundle med Prototype Shield kompatibel med AZ-MEGA2560-Board” som fås förmånligt på amazon.se. Den kräver en större låda än den lilla Open CW keyer Mk2.

Om man önskar kan alla dessa funktioner och många fler väljas med Arduino *Mega* i K3NG CW keyer:

- Keyern kan köras och konfigureras från *direktanslutet* tangentbord och ge morse ut till sändaren
- Keyern kan köras från program som matar ut WinKeyer kommandon (ex:N1MM+)
- *Både* kommandorad (CLI, praktiskt) och CW-kommandon (portabelt) tillgängliga
- Träningsmoduler för CW
- Farnsworth och Wordsworth keying, dvs längre tecken- och ordmellanrum för träning
- Enkel avkodare för CW alt. Hellschreiber
- Välja nyckla en av 6 riggar
- HSCW (high-speed) och QRSS (slow-slow CW)
- Webbsida för konfigurering och CW-minnen (t.ex mobiltelefon)
- Sammankoppling av 2 keyers över nätverk, m m.

C. Exempel på statusutskrift med Open CW keyer CLI

```
\s
Iambic A
Buffers: Dit On Dah On
WPM: 23
Command Mode WPM: 16
Sidetone: On 600 hz
Dah to dit: 3.00
Weighting: 50
Keying Compensation: 0 mS
Serial Number: 4
Potentiometer WPM: 24 (Activated) Range: 13 - 40 WPM
Autospace Off
Autospace Timing Factor: 2.00
Wordspace: 7
TX: 1
Quiet Paddle Interrupt: Off
Mill Mode: Off
PTT Buffered Character Hold: Off
```

```
PTT: Disabled
TX Inhibit: Off
TX Pause: Off
Paddle Echo: On
Paddle Echo Timing Factor: 1.75
Tx1 lead time: 0
Tx1 tail time: 10
PTT hang time: 0.00 wordspace units
Memory repeat time: 3000
Memory 1:QRZ K
Memory 2:DE \I9
Memory 3:CQ CQ DE \I9 \I9 \I9 CQ K
Memory 4:\Y95NN\Z9 \E \N\E K
Memory 5:JO76KX
Memory 6:73 TU EE
Memory 7:\Y95NN\Z9 \C \N\C K
Memory 8:TEST \I9
Memory 9:SA7CND
Memory 10:{empty}
Memory 11:{empty}
Memory 12:{empty}
```