Beacon per la banda dei 630 metri

Incuriosito da qualche articolo trovato in rete che riguarda la banda dei 630m e relative trasmissioni a bassa potenza, ho assemblato un piccolo trasmettitore/beacon per provare cosa/quanto è possibile fare con pochi watts su queste frequenze, ben consapevole dei limiti del caso viste le lunghezze d'onda in gioco ed il comportamento di tale banda di trasmissione.

La realizzazione nasce come beacon in WSPR a cui in seguito ho aggiunto altre modalità di trasmissione che sono riportate più avanti in queste note.

Qualche mese fa avevo realizzato un transverter seguendo un progetto di G3XBM ma pensando a trasmissioni principalmente in wspr, diventa poco pratico dedicare un trasmettitore della stazione a questo scopo.
Per chi fosse interessato, la documentazione relativa alla realizzazione del transverter è disponibile sul sito di ARI Cernusco. (www.aricernusco.it)

Ho quindi optato per un trasmettitore/beacon dedicato pilotato da un modulo che usa un ESP8266.
ESP8266 è un chip prodotto da una azienda cinese, apparso sul mercato qualche tempo fa, che include oltre ad un microcontrollore una radio WiFi e relativo supporto del protocollo TCP/IP.

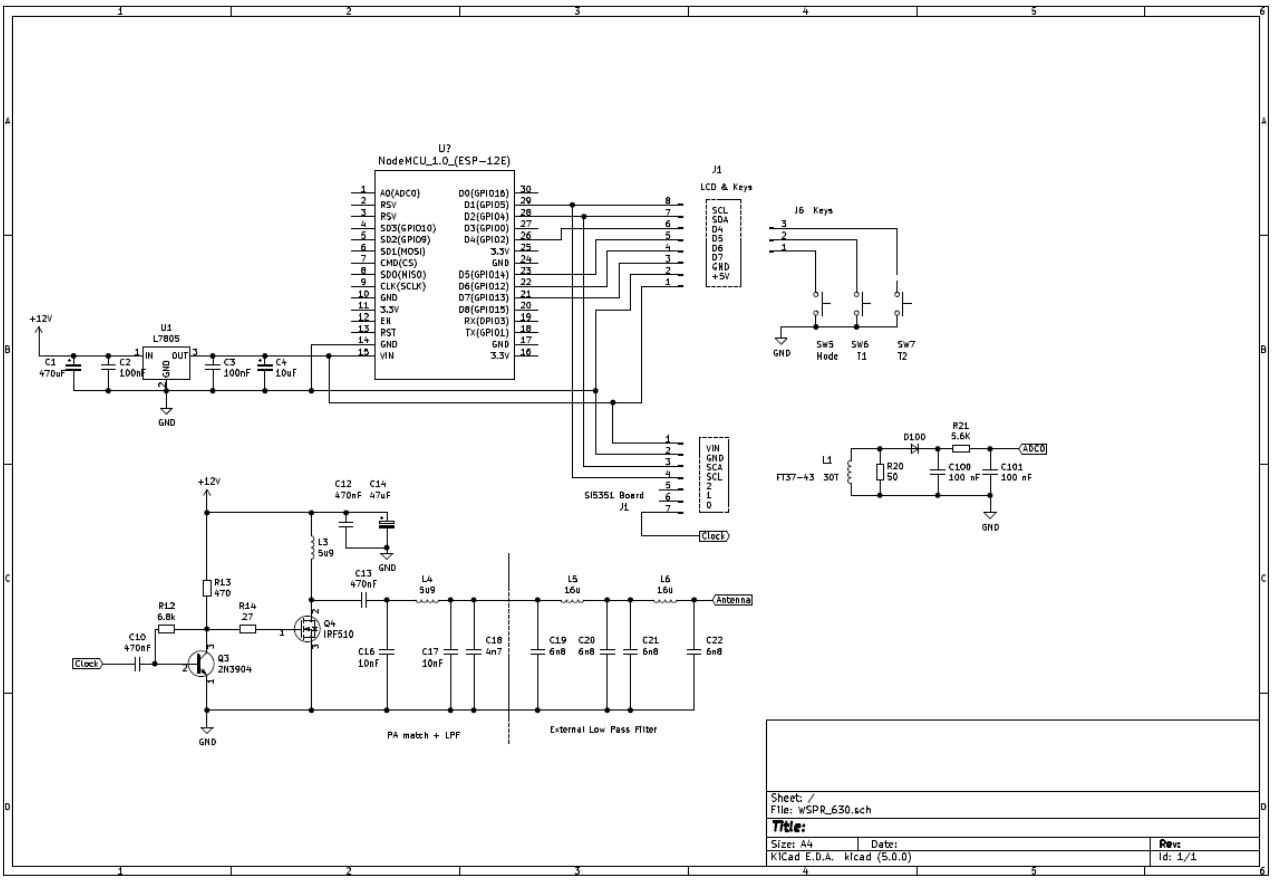
Nel mio caso ho usato un modulo NodeMCU V1 che monta un microcontrollore ESP8266-12E con clock a 80 Mhz, una memoria flash di 4Mb per contenere il programma, 11 pin digitali I/O, un ADC e varie tipologie di comunicazione incluse I2C, SPI e Seriale.
I pin realmente utilizzabili sono in realtà in numero minore perchè parte degli stessi è usato anche per altre funzioni e quindi non si ha un uso 'esclusivo' degli stessi.

La possibilità di avere un WiFi integrato, risolve anche il problema della sincronizzazione dell'ora per le trasmissioni in WSPR; sfruttando la connessione alla rete WiFi di casa si utilizza un qualsiasi server NTP (Network Time Protocol) per sincronizzare l'orologio sviluppato nel SW.
In aggiunta, la stesso WiFi è usato per la configurazione ed il controllo del beacon, anche remotamente, attraverso un browser.

Il modulo è programmabile usando un linguaggio interpretato di nome Lua-script, o in C usando l'ambiente integrato di sviluppo (IDE) di Arduino.
Vista la facilità d'uso del secondo e le centinaia di librerie disponibili in rete ho usato per questa realizzazione l'IDE di Arduino.
Per chi volesse approfondire l'argomento ESP8266 NodeMcu rimando a ricerca in Internet dove sono disponibili tutte le informazioni del caso .

La parte trasmittente è costruita attorno ad un modulo SI5351 e un amplificatore a MOSFET in classe E copiato dal transverter menzionato in precedenza.

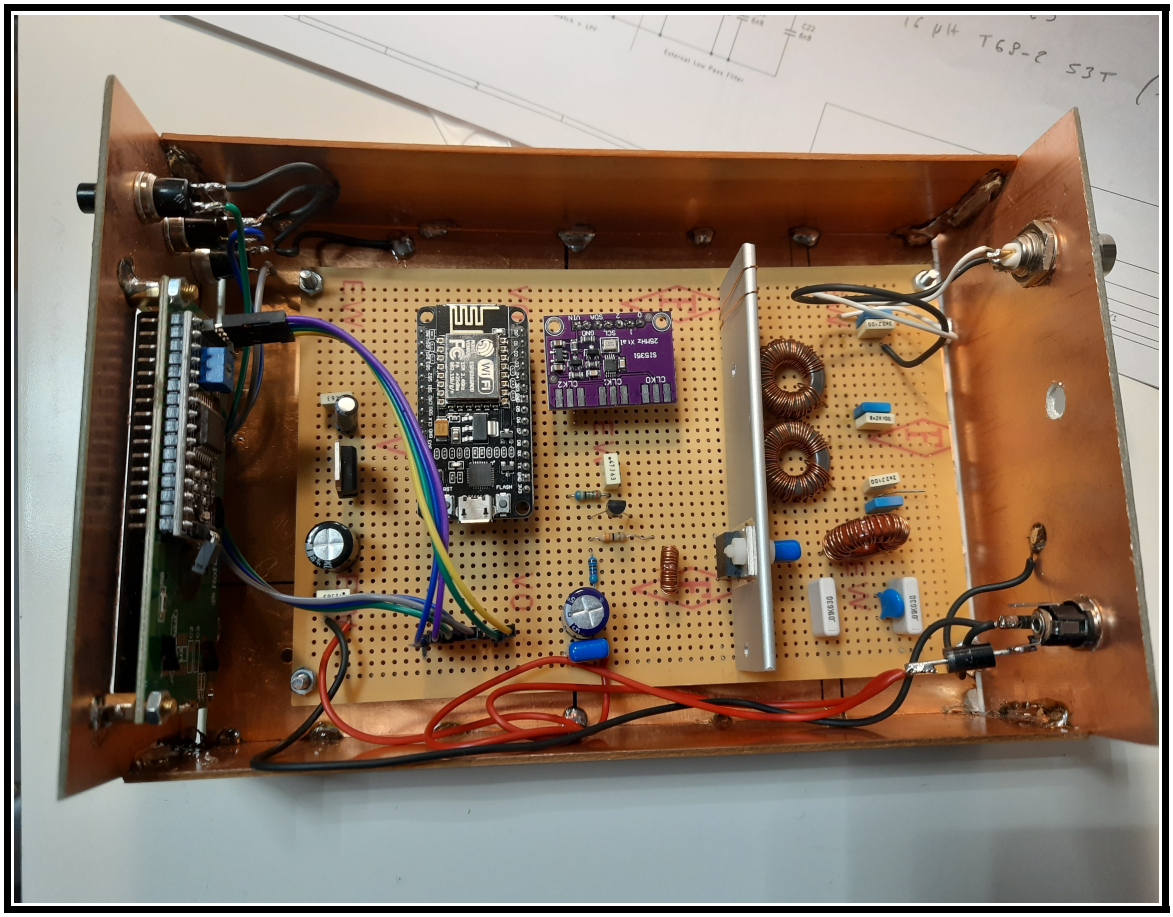
Questo è lo schema elettrico del beacon.



La parte dello schema che riguarda le componenti a radio frequenza non è particolarmente complessa e non mi dilungherei più di tanto nella sua descrizione.
A grandi linee, il segnale generato dal modulo SI5351 è squadrato e trasformato da Q4 in impulsi di ampiezza opportuna per poter pilotare il mosfet usato, nel nostro caso un IRF510, che necessita di 8V sul gate per andare in saturazione.
Per quanto riguarda le bobine, sono tutte realizzate su toroide; FT37-43 per L3, TO50-2 (rosso) per L4, L5 e L6.

Ho realizzato due prototipi dell'oggetto uno su una basetta millefori e un altro in aria su di un pezzo di circuito stampato (o Manhattan style come dicono i colleghi US) visto le basse frequenze in gioco.
In entrambe i casi non ho notato particolari problemi di auto oscillazioni o simili; nel caso G3XBM suggerisce di aggiungere una perlina di ferrite sul gate del mosfet e/o ritoccare il valore di R14.

Segue una foto della realizzazione su millefori.
Da notare nella parte sinistra i moduli NodeMCU e SI5351 e la parte del circuito relativa a Q3.
A destra il mosfet con relativo dissipatore e i filtri incluso il passa basso costruito attorno a L5/L6.



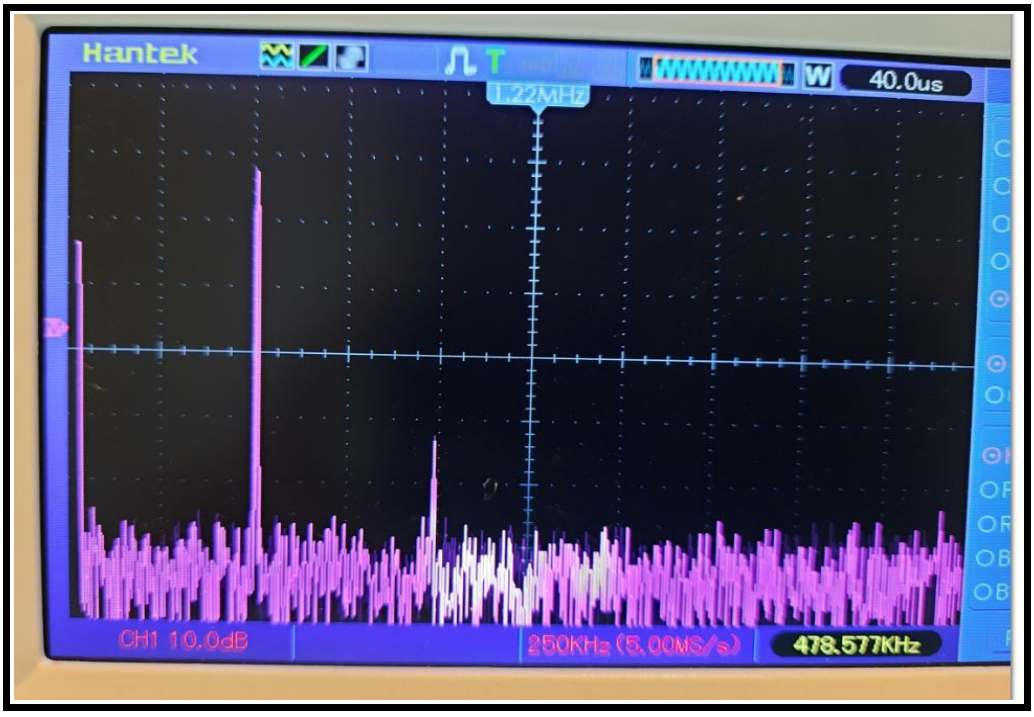
Mediando le misure fatte sui due prototipi, in output si possono ottenere diverse potenze in funzione della tensione di alimentazione (Vcc) del mosfet.
Nella tabella che segue i watts in uscita sono calcolati misurando la tensione picco pico (Vpp) su di un carico fittizio di 50 ohm variando Vcc.

Vcc	Vpp	Watts
12V	45	5
13V	50	6
14V	55	7,5
15V	60	9
16V	65	10
17V	71	12
20V	82	16

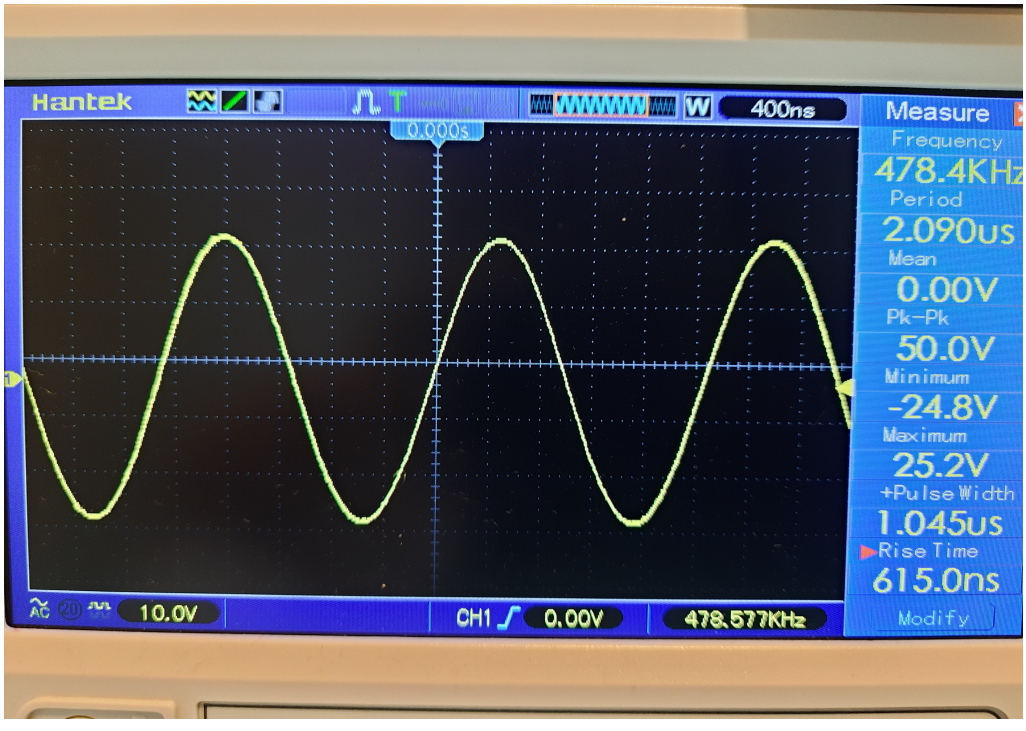
Per il mio uso 'normale' del beacon fornisco una tensione di alimentazione massima di 14 Volts ed il radiatore del mosfet è stato dimensionato di conseguenza; nel caso si vogliano più watts in uscita le dimensioni dello stesso vanno ovviamente aumentate.

Il regolatore 7805 (in contenitore TO220) alimenta i moduli NodeMCU e SI5351 più il display LCD.
L'assorbimento è di circa 130-150 milliampere a 14V e la potenza dissipata è di 1.35 watts; serve qundi un dissipatore adeguato.
Nel caso si voglia salire con la tensione di alimentazione suggerirei di separare l'alimentazione della parte logica da quella di alimentazione del mosfet per evitare di aumentare la potenza dissipata dal regolatore.

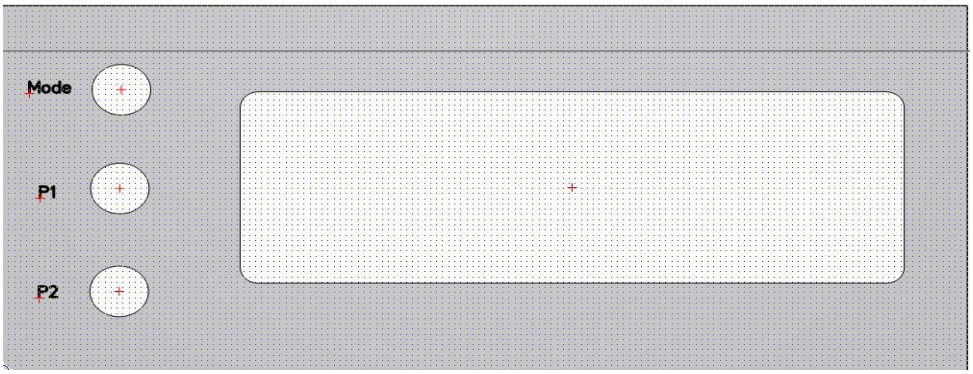
Il passa basso in uscita riduce la seconda armonica di 35/40 dB; questa è la FFT del segnale elaborato dall' oscilloscopio che mostra la fondamentale e la seconda armonica.



Questa è la forma d'onda misurata sul carico fittizio con alimentazione a 13 Volt e uscita di 6 Watts (50 Vpp).



Il beacon è controllabile usando 3 tasti ed un display lcd da 16 caratteri per 2 righe.
Con riferimento allo schema del beacon il taso Mode è connesso al pin D7 del modulo NodeMcu,
i tasti P1 e P2 rispettivamente ai pin D6 e D5. Il display LCD (I2C) è connesso ai pin D1 (SCL) e D2 (SDA)
Segue un esempio di un pannello frontale per lo stesso.



Al momento, il codice realizza trasmissioni in WSPR, OPERA, QRSS, DFCW e FSKCW.
E' possibile operare anche in CW usando sia tasto verticale che tasto Iambic.
Ho aggiunto quest'ultima funzionalità per scopi puramente di test; la portante viene generata/rimossa con la pressione/rilascio del tasto ma non ci sono meccanismi per ridurre i click generati (key shaping) in questo modo.

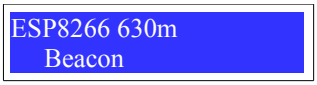
Per funzionare il beacon necessita di un collegamento a Internet attraverso il WiFi di casa perchè la configurazione ed eventuale controllo da remoto sono realizzate usando 'dialoghi' via http..

Come riportato in precedenza, per la sincronizzazione dell'ora si usano i servizi forniti da Server NTP in rete.
NTP è un protocollo utilizzato per sincronizzare gli orologi dei computer in una rete.
Consente ad un dispositivo di richiedere e ricevere l'ora da un server affacciato in rete/Internet che a sua volta riceve l'ora precisa da un orologio atomico.

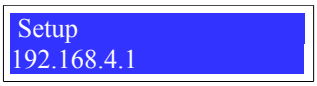
Il codice che controlla il beacon, come già riportato precedentemente, è scritto in C usando l'IDE di Arduino.
Ho usato alcune librerie o porzioni di codice delle stesse per pilotare per esempio il Modulo SI5351 e per codificare il messaggio WSPR.

Il beacon necessita di essere configurato con varie informazioni quali nome/password dell'access point (wifi di casa), callsign, locatore, etc prima di essere utilizzato.

Per configurare da zero il beacon, accendere lo stesso tenendo premuto il tasto Mode.
Apparirà questa prima schermata per qualche attimo



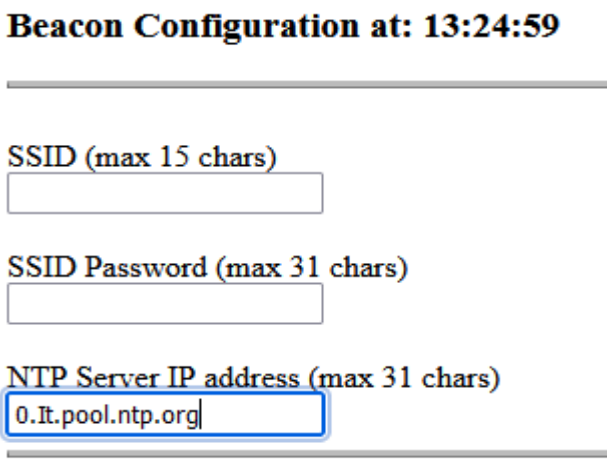
Seguita da questa che indica che il codice del beacon è entrato in modalità configurazione (Setup)



Il codice configura il beacon come un access point wifi con nome **BeaconSetup.**
Da un telefono/tablet/pc dovete temporaneamente connettervi a questa rete, la password per la connessione è **passw0rd** (con il numero 0 al posto della o).

Una volta collegati alla rete, da un browser digitare nella barra degli indirizzi **192.168.4.1**
(come riportato sul display LCD) che è l'indirizzo IP del nostro beacon nella rete **BeaconSetup**.

Apparirà una schermata con tutte le variabili da impostare per la configurazione del beacon.
Di seguito l'elenco delle stesse con relativa descrizione.



- **SSID**
  E' il nome dell'access point di casa a cui il beacon si connetterà durante il normale funzionamento.
  Lunghezza massima accettata è di 31 caratteri.
  Da notare che questa versione del codice del beacon non verifica se questa lunghezza viene superata né verifica il corretto contenuto dei campi impostati.
  Ponete attenzione a questa cosa per tutti i caratteri che andrete a digitare di seguito, incluso caratteri maiuscoli e/o minuscoli.

- **SSID Password**
  La password del vostro access point di casa.
  Lunghezza massima accettata è di 31 caratteri.

- **NTP Server IP address**
  Indirizzo del server NTP (Network Time Protocol) a cui il beacon si connetterà durante il funzionamento per sincronizzare l'orologio al suo interno.
  Quello riportato, ad esempio, è l'indirizzo di un NTP server Italiano accessibile in internet
  La lunghezza massima accettata è di 31 caratteri.

Segue la parte della configurazione che riguarda la modalità WSPR-2

WSPR Call (max 7 chars)
IW2XYZ

WSPR Locator (max 4 chars)
JN45

WSPR dbm (max 2 chars)
10

WSPR Minute (max 2 chars)
10

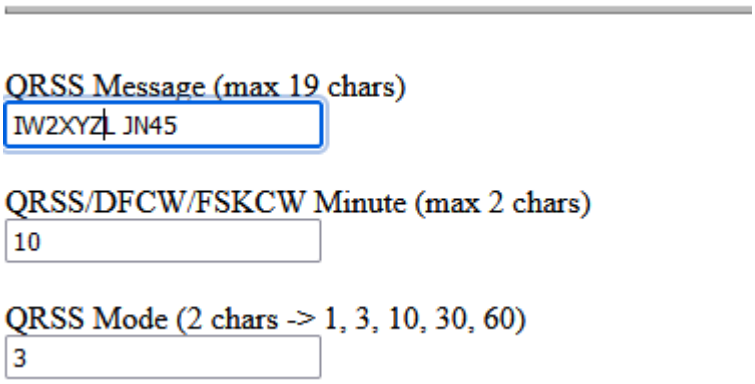
- **WSPR Call**
  E' il nominativo (il vostro) che verrà usato nella composizione del messaggio WSPR.
  La lunghezza massima accettata è di 7 caratteri che devono essere digitati in **maiuscolo**.

- **WSPR Locator**
  E' il vostro locatore che verrà usato nella composizione del messaggio WSPR.
  La lunghezza massima accettata è di 4 caratteri che devono essere digitati in **maiuscolo**.

- **WSPR dbm**
  E' il valore della potenza usato del beacon espressa in dbm.
  Deve rispettare i valori di potenza imposi dal protocollo WSPR; vi rimando alla documentazione dello stesso per conoscere i valori accettati .
  Nell'esempio 10 corrisponde a 0.010 Watts (10 milliwatts).
  La lunghezza massima accettata è di 2 caratteri.

- **WSPR Minute**
  E' il minuto a cui il beacon inizierà a trasmettere il messaggio WSPR.
  Nell'esempio il valore 10 indica che la trasmissione inizierà al minuto 0, 10, 20, 30, 40 ,50 dell'ora.
  Considerando che la trasmissione di un messaggio WSPR-2 impiega poco meno di 2 minuti per essere trasmesso, la cosa si svilupperà in 2 minuti di trasmissione (al minuto 10, 20 etc) seguiti da 8 minuti di pausa.
  Se ad esempio il valore impostato fosse uguale a 15, la trasmissione inizierebbe al minuto 0, 15, 30, 45; avremo i soliti di 2 minuti di trasmissione seguiti da una pausa di 13 minuti.

Segue la parte della configurazione che riguarda la modalità QRSS/DFCW/FSKCW



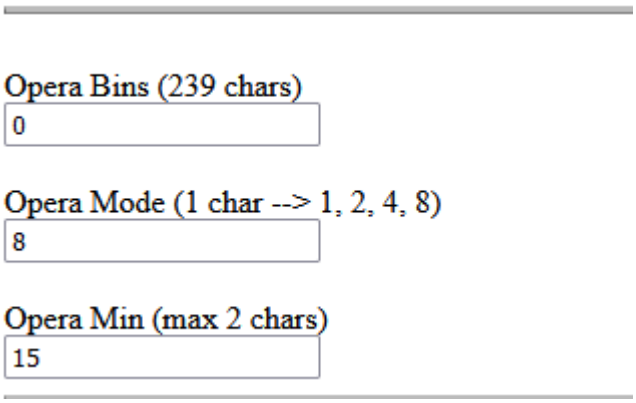
- **QRSS Message**
  E' il testo del messaggio che verrà trasmesso in QRSS/DFCW e FSKCW; nell'esempio IW2XYZ JN45.
  Può essere un qualsiasi testo alfanumerico senza caratteri speciali (punti, vergole etc).
  Il solo carattere speciale che si può utilizzare nel testo è la barra (/) e la lunghezza massima accettata è di di 19 caratteri

- **QRSS/DFCW/FSKCW Minute**
  Valgono le stesse considerazioni del WSPR.
  Và valorizzato considerando il tempo necessario per la trasmissione del messaggio che varia in base alla scelta nel campo successivo e la modalità di trasmissione scelta.
  Da notare che normalmente la partenza di una trasmissione in queste modalità non deve necessariamente essere sincronizzata con un minuto specifico.
  Ho scelto di iniziare la trasmissione in un momento preciso dell'ora per avere la possibilità di sommare o sovrapporre più ricezioni dello stesso messaggio ricevuto usando un programma.
  L'idea non è nuova, in internet ho trovato qualche vecchio documento (Coherent CW) che tratta di argomenti simili che mi hanno incuriosito.
  Vero che nei nostri giorni WSPR e OPERA hanno potenzialità enormi per quanto riguarda la ricezione di segnali debolissimi e relativo SNR, ma ho comunque voluto tenere aperta una porta a questa vecchia/nuova sperimentazione.
  La lunghezza massima accettata è di 2 caratteri.

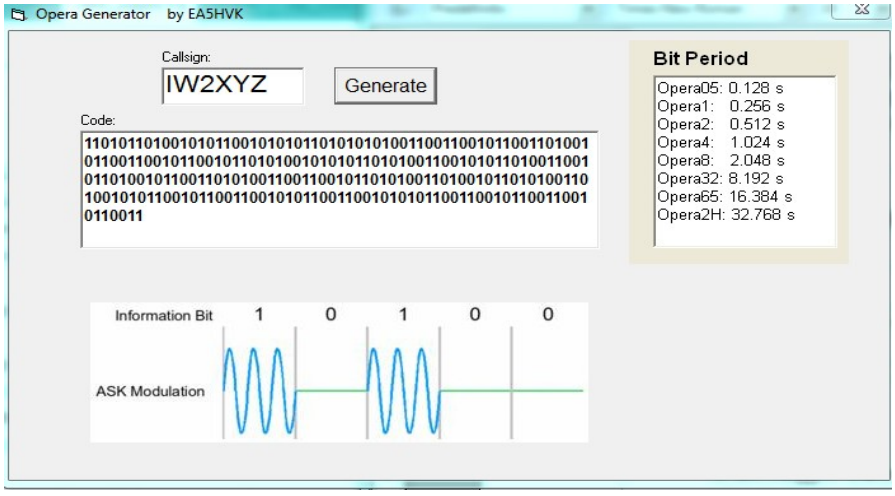
- **QRSS Mode**
  E' la modalità di trasmissione QRSS.
  Il valore identifica la lunghezza di un punto espressa in secondi; per esempio 3 indicherà che un punto avrà una lunghezza di 3 secondi mentre una linea avrà lunghezza di 9 secondi.
  E' valido per tutte le tre modalità di trasmissione.

Segue la parte della configurazione che riguarda la modalità OPERA

Questa modalità di trasmissione è stata sviluppata da EA5HVK ed implementa una trasmissione seriale dei dati con correzione degli errori. Vi rimando al sito in internet dell'autore dove potete trovare ulteriori informazioni (www.rosmodem.wordpress.com) ed il programma **Opera Generator** che servirà di seguito per la configurazione della modalità OPERA. (rosmodem.wordpress.com/tag/opera/)



- **Opera Bins**
  Il messaggio trasmesso in modalità beacon da OPERA contiene il solo callsign al quale sono aggiunte altre informazioni per la trasmissione a prova di errore del messaggio stesso.
  Dopo questa ultima manipolazione e preparazione dei dati il messaggio ha una lunghezza di 239 bits.
  Al momento non ho trovato porzioni di codice per codificare l'intero messaggio partendo dal callsign, per cui ho usato il programma Opera Generator per creare il messaggio finale.
  Dopo aver fatto partire il programma ed aver inserito il proprio callsign, premendo il tasto Generate il programma mostra i 239 bits che ci servono (Code).
  A questo punto con un copy/paste inserite i 239 caratteri richiesti nel campo Opera Bins.

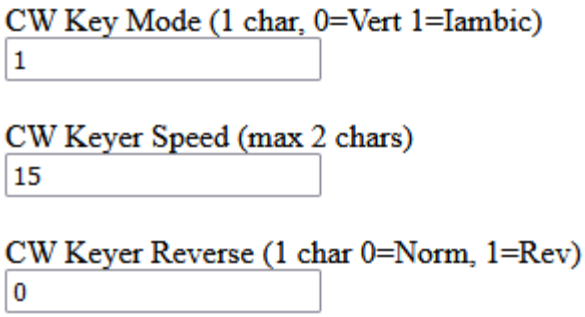


- **Opera Mode**
  Il protocollo Opera prevede 4 'tipi' di messaggio che aumentano di conseguenza il tempo necessario alla trasmissione del messaggio stesso come mostrato qui sopra.
  In 630 metri si usa normalmente Opera 8 tempo necessario per la trasmissione 239 * 2.048 sec = 8.15 minuti.
  La lunghezza massima accettata è di 2 caratteri.

- **Opera Min**
  Valgono le stesse considerazioni fatte per WSPR e QRSS.

Segue la parte della configurazione che riguarda la modalità CW.

Come riportato sopra, ho aggiunto questa modalità per fini di test.
L'HW minimale impiegato ed il codice non si prendono cura di eventuali click (key shaping) generati indirettamente dalla manipolazione con il taso CW; la portante viene semplicemente emessa e rimossa seguendo la manipolazione stessa.

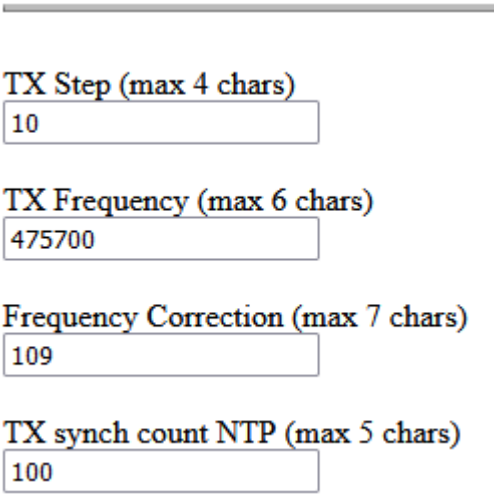


- **CW Key Mode**
  Identifica il tipo di tasto CW che si vuole utilizzare per la trasmissione
  0 -> Tasto Verticale **1**-> Tasto Iambic
  La lunghezza massima accettata è di un carattere.

- **CW Keyer Speed**
  Il codice implementa un semplice keyer la cui velocità espressa in WPM si imposta riempendo questo campo .
  La lunghezza massima accettata è di 2 caratteri.

- CW Keyer Reverse
  Inverte, nel caso, i paddle del tasto (punto/linea o linea/punto).
  0 ->punto/lineae **1**-> linea/punto

Seguono altri parametri di configurazione del sistema.



- **TX Step**
  Imposta il numero di Hz con cui viene incrementata/decrementata la frequenza di trasmissione in modalità CW/TUNE premendo i tasti P1 o P2 quando in modalità F STEP (Frequency Step).
  Lunghezza massima accettata è di 4 caratteri

- **TX Frequency**
  Cambia la frequenza di trasmissione del beacon.
  La frequenza di trasmissione per quanto riguarda le varie modalità è pre-impostata dal codice quando viene selezionata una modalità specifica come segue

  WSPR -> 475700 QRSS -> 476200 OPERA -> 478500

  E' possibile cambiare la frequenza usando i tasti P1 e P2 o usando l'interfaccia via browser impostando questo campo con la frequenza di trasmissione desiderata. Questa modifica non viene preservata cambiando la modalità di trasmissione.

  La lunghezza massima accettata è di 6 caratteri.

- **TX Frequency Correction**
  E' usato per impostare un valore di correzione per compensare il valore dell'oscillatore SI5351.
  Il valore del quarzo usato nel modulo SI5351 può essere spostato rispetto alla frequenza dichiarata a causa delle tolleranze introdotte durante la costruzione dello stesso e quindi la frequenza generata puo essere incorretta. Per ovviare a ciò, valorizzando questo campo è possibile forzare l'oscillatore ad aumentare o diminuire la frequenza generata.

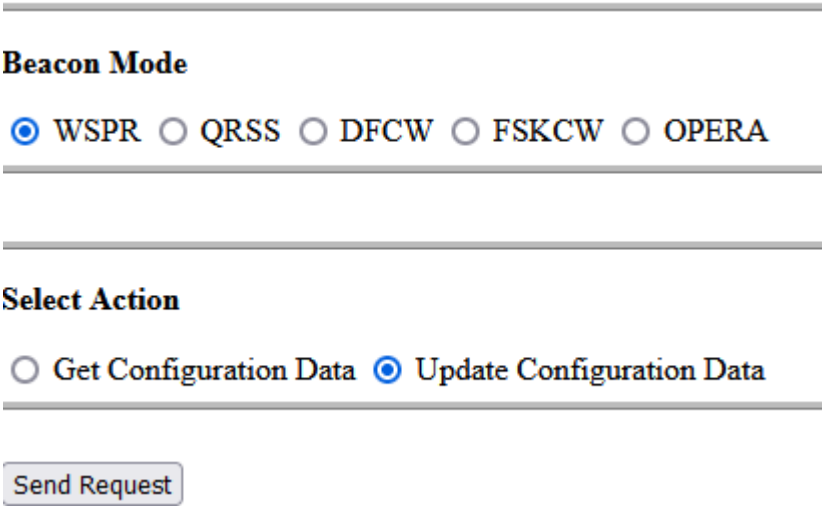
  La lunghezza massima accettata è di 7 caratteri.

- **TX Synch count NTP**
  Il beacon sincronizza il suo orologio interno usando il server NTP alla sua partenza.
  L'orologio è creato e mantenuto dal codice usando registri/divisori interni al chip ESP8266.
  In pratica, il codice viene interrotto ogni secondo da un interrupt la cui routine di gestione mantiene un orologio nella forma hh:mm:ss.
  Anche in questo caso la frequenza dei quarzi usati nel modulo NodeMcu posso avere delle tolleranze e di conseguenza l'orologio implementato nel codice potrebbe nel tempo diventare non più sincronizzato.
  Durante le mie prove non ho avuto problemi di questo tipo anche dopo 4 o 5 ore di funzionamento.

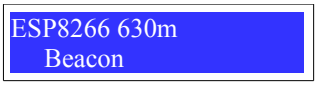
E' possibile/consigliato comunque forzare una re-sincronizzazione dell'orologio impostando in questo campo un valore che indica dopo quante trasmissioni deve essere innescato questo processo.
Nell'esempio 100 indica che dopo 100 trasmissioni deve essere innescato il processo di re-sincronizzazione.

La prima configurazione del beacon è a questo punto conclusa.
Ignorate al momento il parametro **Beacon Mode**

Selezionare il radio button **Update Configuration Data** e premere il pulsante **Send Request**
per salvare le informazioni digitate .
Il codice salverà la configurazione in un spazio di memoria nel ESP8266 destinato a EEPROM.



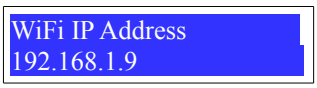
A questo punto il nostro beacon è stato configurato e possiamo farlo ripartire spegnendo e riaccendendo lo stesso.
Alla riaccensione comparirà quanto segue sul LCD



Dopo qualche secondo l'LCD coambierà in

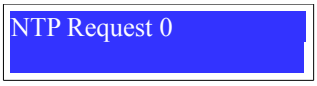
Il beacon si stà collegando all'acces point di casa il cui nome (SSID) è stato configurato in precedenza.
Ad esempio se in fase di configurazione ho indicato MY_SSID come nome, lo stesso è riportato nella seconda riga del display; a connessione avvenuta il display cambia in

Il processo di inizializzazione del WiFi continua e il nostro access point assegna un indirizzo IP dinamico al nostro beacon, indirizzo che viene riportato sul LCD come segue:



Il nostro beacon ha indirizzo IP **192.168.1.9** (è un esempio, cambia nella realtà).
Questo è l'indirizzo che dovremo usare per configurare e controllare il beacon successivamente; non sarà più necessario forzare in beacon in modo 'setup' come fatto in precedenza per eventuali modifiche future.
Da notare che questo è l'indirizzo IP assegnato dall'access point per questa sessione; se il beacon viene spento e riacceso la funzione di DHCP dell'access point potrebbero assegnare un indirizzo IP diverso.

A questo punto il codice del beacon sincronizza l'ora del suo orologio interno con quello ricevuto del server NTP che abbiamo identificato in precedenza in fase di prima configurazione.
La richiesta è inviata al server e il display LCD cambia in



A questo punto abbiamo spedito una richiesta di richiesta dell'ora al server e stiamo aspettando una risposta.
Nel caso il server non risponda, il codice rimanda per 20 volte la richiesta ed il numero, 0 in questo caso, viene incrementato di conseguenza.
Quando la risposta del server è arrivata il codice lo indica modificando il display in

A questo punti, abbiamo inizializzato la parte WiFi e abbiamo un ora valida e sincronizzata caricata nell'orologio del beacon; il contenuto del display cambia in:

Il processo di inizializzazione del beacon è completo.

Alla accensione il beacon si configura sempre in modalità WSPR e l'ora dell'orologio interno è mostrata e aggiornata sul display LCD.
Quando il minuto di partenza della trasmissione WSPR è uguale a quello configurato in precedenza, la trasmissione ha inizio ed il display cambia in

La scritta **TX** indica che il beacon ha una trasmissione attiva; scomparirà a trasmissione completata.
Da notare che durante la trasmissione l'ora sul display LCD non viene aggiornata; sarà re-aggiornata al termine della trasmissione.

Premendo il taso Mode si cambia la modalità di trasmissione tra i modi disponibili:

WSPR → OPERA → SET F → TUNE → CW → QRSS → CW_FSK → CW_DFC

Da notare che SET F e TUNE non sono modalità di trasmissione.

L'oggetto nasce come beacon per frequenze pre-assegnate a questo scopo e quindi non ho previsto un controllo della frequenza 'agile' usando una manopola o cose simili.
Per cambiare la frequenza di trasmissione per il modo CW o TUNE, entrare nella modalità SET F e usare i tasti P1 o P2 per incrementare/decrementare la stessa .

In tutte le modalità di trasmissione, esclusa CW , quando la trasmissione non è attiva, è possibile cambiare la frequenza di emissione premendo i tasti P1 e P2.

In modalità TUNE premendo P1 il beacon emette emette una portante utile per fare il 'tuning' dell'antenna.
Il display cambia in

Il numero, 4 in questo caso, è un numero che indica l'intensità del segnale trasferito all'antenna; maggiore è questo valore, maggiore è la potenza trasferita.

Per questa funzione (facoltativa) ho usato un semplice misuratore di RF attorno al toroide L1.
Ho usato una trentina di spire per lo stesso su di un FT37-43; il filo che và all'antenna lo attraverso inducendo un valore di tensione che viene poi letto dall'ADC del module ESP8266 e mostrato sul display LCD.

E' possibile, **a trasmissione non attiva**, modificare i parametri di configurazione e modalità di funzionamento del beacon usando una interfaccia via browser per controllare il beacon stesso anche remotamente.
Questa funzione è attiva quando il beacon è in modalità **WSPR , QRSS, DFCW, FSKCW** e **OPERA.**

Aprite un browser da telefono/tablet/pc ed inserite nella barra degli indirizzi quello assegnato al nostro beacon dall'access point di casa; nel nostro esempio **http://192.168.1.9** (vedi sopra).
La pagina che verrà presentata è esattamente la stessa pagina vista durante la configurazione, dove a questo punto si potranno modificare tutti i parametri riportati nella pagina stessa.

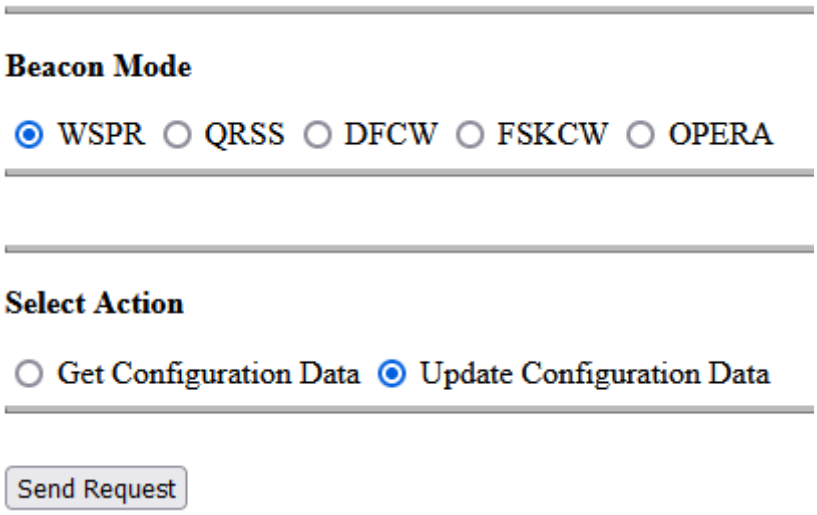
Da notare nella pagina alcune cose.

La prima riga nella pagina riporta l'ora dell'orologio interno al beacon, utile per verificare il sincrono dello stesso quando si controlla lo stesso remotamente via browser.

Beacon Configuration at: 17:45:22

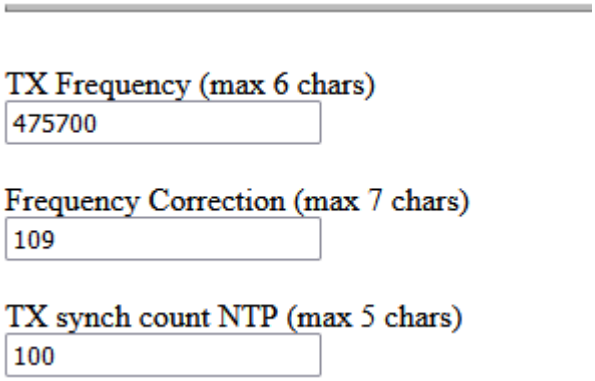
Selezionando il Radio Button **Get Configuration Data** e premendo il pulsante **Send Request** si chiede al beacon di mostrare i parametri di configurazione.

In questo modo si ha una ri-lettura degli stessi inclusa l'ora dell'orologio interno; non viene salvato alcun dato anche se modificato.



E' anche possibile impostare/modificare la modalità di trasmissione del beacon tra
WSPR → QRSS → DFCW → FSKCW → OPERA
selezionando il radio button opportuno e premendo il pulsante **Send Request.**
La trasmissione inizierà al tempo selezionato per la modalità selezionato usando i relativi parametri impostati in fase di configurazione (ad esempio minuto di inizio della trasmissione, lunghezza dei punti/linee in QRSS etc).

E' anche possibile modificare la frequenza di trasmissione del beacon digitando la stessa nel campo **TX Frequency** del pannello e premendo il pulsante **Send Request**



In allegato trovate il codice sorgente del beacon che rilascio con i termini della licenza GNU GPL; non posso garantire future manutenzione dello stesso.

Per la compilazione del codice con IDE Arduino in aggiunta alle librerie 'standard' di Arduino servono in aggiunta le seguenti, scaricabili via Sketch → Include Library → Manage Libraries

ESP32, ESP8266WiFi, ESP8266WebServer, ESP_EEPROM, LiquidCrystal_I2

Per la stesura del codice ho usato librerie disponibili in rete o porzioni di esse, per le quali valgono, le regole di copyright (GPL) fornite dai loro autori.
E' scritto in modo un po' disordinato, e a volte ridondante; sentitevi liberi di modificarlo/migliorarlo come meglio preferite.

I