Amateur Radio Direction Finding (ARDF)

Un paio di ani fà con alcuni colleghi della sezione ARI di Cernusco sul Naviglio, si parlava di ARDF (Amateur Radio Direction Finding) o caccia alla volpe, se volete.
Avevamo preparato un paio di 'volpi' e identificato un'area che sembrava essere perfetta per la caccia ma poi la pandemia e relative chiusure ci hanno costretto a cambiare i nostri piani.

ARDF è una gara di radio localizzazione dove i cacciatori armati di radio e antenne direzionali devono scovare una o più volpi che non sono altro che piccole trasmittente di piccola potenza che trasmettono in modo omnidirezionale un loro identificativo.

L'obbiettivo della caccia è trovare le volpi nel minor tempo possibile.

Le trasmittenti operano sulla stessa frequenza e trasmettono il loro identificativo una alla volta; quando una volpe trasmette le altre restano in attesa della loro finestra di trasmissione.

Immaginando una gara con 3 volpi, inizia a trasmettere per prima la volpe_1 per un minuto.
Passato il minuto la volpe_1 si mette in attesa e inizia a trasmettere la volpe_2.
Passato un altro minuto la volpe_2 si mette in attesa e inizia a trasmettere la volpe_3 e così via...

Le volpi trasmettono un identificativo in codice morse (CW) che identifica in modo univoco ogni singola volpe. L'identificativo è composto dalle lettere MO seguite da un numero di punti (in CW) che variano da 1 a 5 (nel caso di 5 volpi).

Perciò la volpe_1 ha come identificativo le lettere MOE la volpe_2 MOI (- - - - - .) , la volpe_3 MOS etc. In questo modo i primi 2 caratteri MO producono la trasmissione di 5 linee CW utili per aiutare l'orientamento delle antenne verso la direzione della volpe, mentre il punto (o i punti che seguono) identificano la volpe.

Le prime volpi realizzate, erano costruite attorno ad un microprocessore (pic12F675) ed un trasmettitore SAW per i 433 Mhz. Dopo un po' di prove sono emersi un paio di problemi che ci hanno indicato di rinunciare al HW usato in precedenza e usare qualcosa di diverso.

Per le nuove volpi, ho optato per un modulo NodeMCU Mini (D1) che permette di avere più flessibilità in termini di spazio per il programma, numero di pin utilizzabili e possibilità di comunicazione via WiFi.

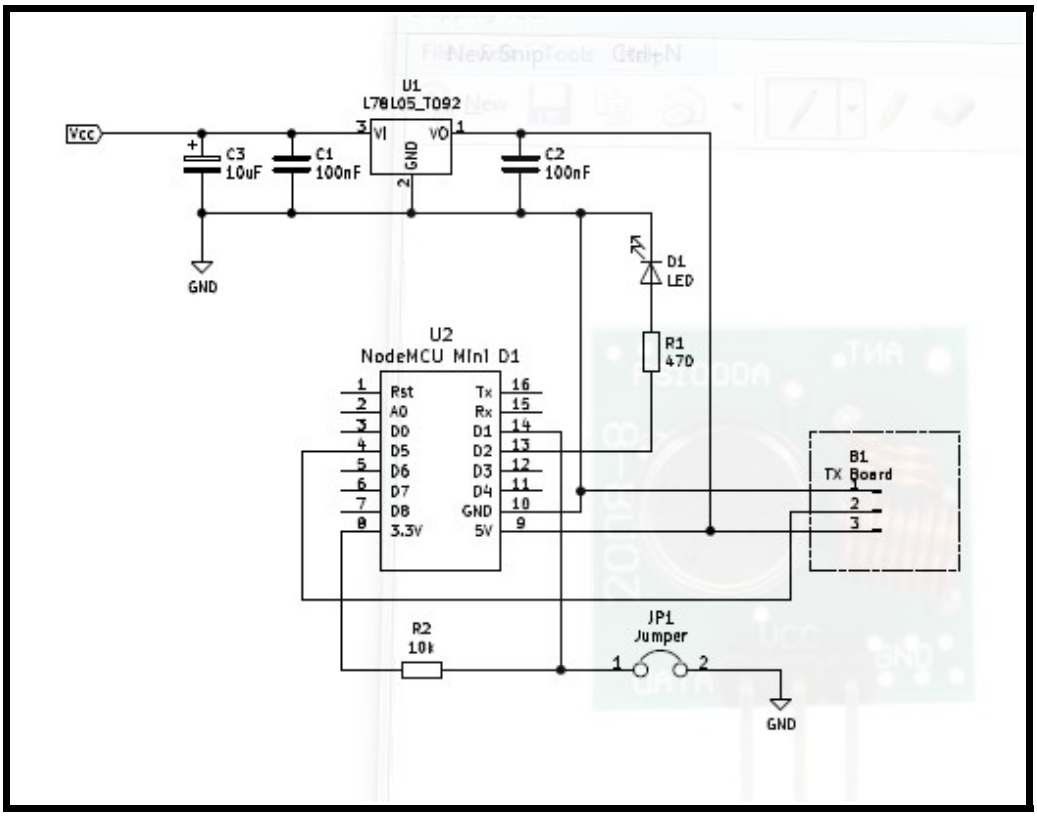
Questa scelta, oltre a quanto espresso in precedenza, permette di usare Arduino IDE per la stesura del programma e relativo caricamento del codice compilato nel modulo NodeMcu. Inoltre essendo la piattaforma Arduino conosciuta a molti è semplificata la possibilità di modificare alla bisogna il programma.

Per quanto riguarda i moduli trasmittenti ho riutilizzato gli stessi usati in precedenza.

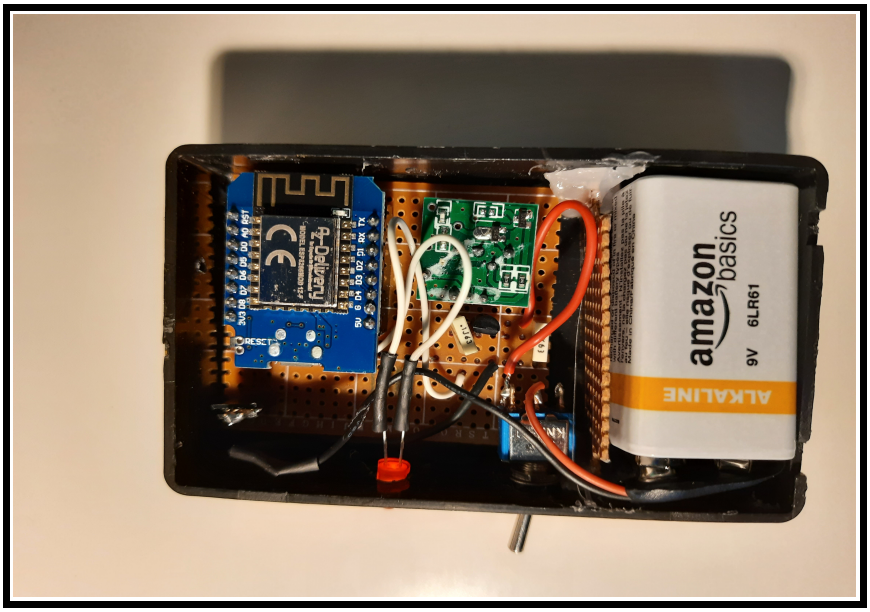
Uno dei problemi emersi durante i test dei primi prototipi era la scarsa sincronizzazione del tempo tra le volpi. Le volpi, per trasmettere nell'intervallo assegnato e non sovrapporsi, devono avere un orologio interno abbastanza preciso che indichi quando iniziare a trasmettere e quando restare in attesa. Se la durata della gara è di qualche minuto, questo problema è poco avvertibile ma se la durata è di qualche decina di minuti può diventa noioso, perchè le trasmissioni delle volpi possono essere sovrapposte.

Per motivi di semplicità e per ridurre al minimo le dimensioni delle volpi ho scartato la possibilità di usare un chip esterno dedicato a questo scopo (tipo Real Time Clock o RTC). Considerando inoltre che il modulo NodeMcu ha un oscillatore a quarzo usato per generare i vari clock interni necessari al suo funzionamento, perchè non usarlo per implementare un orologio via SW...

Questo è lo schema elettrico della 'nuova' volpe; i componenti sono facilmente reperibili anche presso i vari venditori online.



Vista la semplicità dello schema non ho disegnato un circuito stampato ma ho realizzato la volpe usando una basetta mille fori alloggiata in una scatola di plastica con dimensione 10x6 cm



Con riferimento alla foto precedente sono visibili partendo da sinistra il modulo NodeMcu Mini ed il modulo radio a 433 Mhz.

Come dicevo in precedenza, il problema più grande è sincronizzare tra loro gli orologi implementati dal SW nel modulo NodeMcu. Questi moduli sono basati su di un microprocesore RISC a 32 bit con una velocità di clock di 80Mhz hanno una memoria RAM utilizzata per contenere i dati di 32Kb e una memoria FLASH che contiene il codice eseguibile di 4MB.
In aggiunta hanno una radio WiFi interna utilizzabile da programma.

Ho usato la funzionalità del WiFi integrato per la sola sincronizzazione del tempo tra le varie volpi. Dopo che la sincronizzazione è avvenuta, la radio viene spenta per limitare l'uso di corrente.
Con la radio spenta ho misurato un consumo di circa 30/35 mA, che permette alla volpe di restare accesa qualche ora usando una normale pila a 9V che ha una capacità di più o meno 200 mAh.
Nel caso di uso prolungato, ci sono batterie ricaricabili da 9V che hanno una capacità 3 o 4 volte superiore a una normale batteria 'generica'.

A causa del processo di sincronizzazione, una delle volpi componente il 'branco' deve essere identificata come capobranco (Master). Questa è la volpe il cui orologio interno verrà usato come riferimento da tutte le altre. Il jumper JP1 è usato per permettere al software di identificare quale delle volpi è la volpe Master. La sola volpe Master deve avere il jumper JP1 'chiuso' mentre tutte le altre volpi devono avere il jumper aperto. Le volpi utilizzano volutamente lo stesso codice per poter essere intercambiabili alla bisogna senza la necessità di riprogrammazione. Ad esempio nel caso di un guasto, una qualsiasi delle volpi può essere innalzata al livello di Master o viceversa.

L'orologio interno è implementato via SW usando un 'Ticker' che è risultato essere sufficientemente stabile per il nostro uso anche se lo stesso non utilizza interrupt HW (è un os_timer).

Nel programma, o sketch nel mondo Arduino, la funzione ***ticker_interrupt()*** implementa il nostro orologio. La funzione viene chiamata una volta al secondo e mantiene tre variabili **ore**, **minuti** e **secondi** che contengono il tempo trascorso dalla accensione della volpe (00:00:00).

L'orologio comincia a scandire il tempo alla accensione della volpe. Ora, per sincronizzarle tra loro dovrei accendere tutte le volpi manualmente nello stesso momento, che è un po difficoltoso, ma può essere una cosa semplice se fatto dal codice (programma).

Procedendo con ordine:

1. accendo una volpe che ha il Jumper inserito. Il SW controlla se JP1 è presente o no, e nel caso positivo elegge la volpe a Master.
2. A questo punto la volpe Master crea una rete WiFi con nome 'ARDF' e si mette in attesa di essere contatta dalle altre volpi del branco.
3. Le altre volpi, accese in sequenza, si connettono alla volpe Master, si fanno spedire il contenuto delle variabili **ore**, **minuti**, **secondi** e sincronizzano le proprie variabili (il loro orologio interno) con quelle spedite dalla volpe Master.
4. A questo punto ogni volpe ha il proprio orologio interno sincronizzato con quello della volpe Master, spegne la radio WiFi e quando richiesto inizia a trasmettere il proprio ID.

Vero, c'è una latenza tra la richiesta dell'ora ed il momento in cui la stessa viene ricevuta ma è contenuta in limiti accettabili (si usa un colloquio usando datagram UDP).

Per programmare le volpi è necessario installare un IDE Arduino; una ricerca con Google produce centinaia di tutorial e video a questo proposito; nel mio caso uso una versione 1.8.9 (non recentissima).

In aggiunta al IDE è necessario installare le librerie ESP8266WiFi e ESP_EEPROM. La prima è usata per controllare la parte WiFi la seconda per salvare i dati di configurazione in una EEPROM emulata in quanto il modulo NodeMcu non dispone di una eeprom usabile da programma.

E' necesario inoltre installare il supporto per ESP8266 nel IDE Arduino; anche in questo caso, cercando in Internet, ci sono centinaia di esempi.

Le volpi devono anche conoscere quanti sono i componenti del branco e quanti minuti di attesa devono aspettare dopo l'accensione prima di iniziare a trasmettere. Il numero di componenti serve per capire quando iniziare a trasmettere e quanto tempo attendere tra una trasmissione e l'altra.

Per evitare di dover riprogrammare le volpi quando si vogliono cambiare questi parametri, è possibile usare una APP per Android (no Apple) inclusa nel file zip allegato. E' scelta di chi poi utilizzera le volpi se avere o non avere la possibilità di cambiare dinamicamente (via APP) questi parametri.

```
// ---------- CONFIGURATION VARIABLES ----------
boolean use_fixed_parms = false;   // use fixed parms from the program
int     fox_max_number = 2;        // number of foxes in use
int     fox_start_minute = 1;      // minute at witch we start xmit
//---------------------------------------------------------------------
```

Con riferimento alle righe sovrastanti, estratte dal codice ***boolean use_fixed_parms*** indica al programma se abilitare o no l'uso dell APP.

Se ***use_fixed_parms = false*** decido di volere modificare il contenuto delle variabili dinamicamente. Le variabili aggiornabili via APP vengono salvate nella EEPROM emulata. A ogni ripartenza, il loro contenuto viene aggiornato con i dati letti dalla EEPROM.

Se ***use_fixed_parms = true*** decido di volere utilizzare il contenuto delle variabili in modo statico. In questo caso le variabili che seguono, ***fox_max_number*** e ***fox_start_minute*** devono essere impostate manualmente. La prima indica (da 1 a 5) quante volpi sono presenti nel branco; la seconda a quale minuto dopo l'accensione inizierà a trasmettere la prima volpe (Master) del branco. In aggiunta la variabile ***use_fixed_parms = true;*** deve essere impostata in questo modo.

Supponiamo che abbia scelto di usare staticamente queste variabili. Il mio branco sarà composto, ad esempio da 3 volpi e voglio che la trasmissione inizi dopo 2 minuti dalla accensione della prima volpe. Devo impostare le variabili come segue e ricompilare/ricaricare in ognuna delle volpi il codice aggiornato. In realtà è sufficiente rinfrescare il programma della sola volpe Master, ma se si vuole mantenere l'intercambiabilità tra le volpi, và fatto per tutte.

```
boolean use_fixed_parms = true;   // use fixed parms from the program
int     fox_max_number = 3;       // number of foxes in use
int     fox_start_minute = 2;     // minute at witch we start xmit
```

Se decido di volere variare dinamicamenye (via APP) il contenuto di queste variabili, devo impostara ***use_fixed_parms = false;*** e ricompilare/ricaricare in ognuna delle volpi il codice aggiornato.

Se in futuro devo cambiare il numero delle volpi o i minuti di attesa per la prima trasmissione, nel caso di uso statico delle variabili devo ricompilare e ricaricare in ognuna delle volpi il codice aggiornato. Se decido di usarle in modo dinamico (usando la APP per impostarle) non devo rifare il passo di ricompilazione/ricarica del codice.

Per inizializzare il branco in entrambe i casi (variabili statiche o dinamiche)

1. Accendo la volpe Master e il led lampeggia per 10 volte, si spegne per un paio di secondi e poi si riaccende fisso.

   Il lampeggio è una indicazione dell'accensione e reset iniziale avvenuto con successo. Durante il periodo in cui il led resta spento, generalmente non più di un paio di secondi il codice imposta la rete WiFi e relativo access point con nome 'ARDF'. Se dopo il lampeggio iniziale il led resta spento, c'è un problema durante il settaggio della radio WiFi e relativa rete. Se tutto è OK fino a questo punto il led si riaccende indicando che la volpe Master è in attesa di essere contattata dalle altre volpi del branco per la sincronizzazione dell'ora.

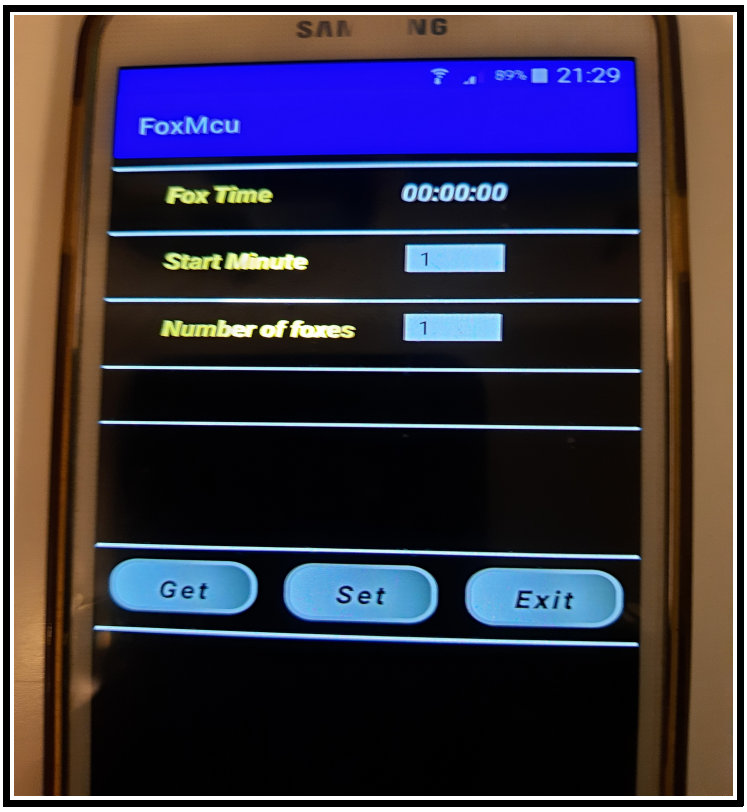
   A questo punto la volpe Master ha creato la rete WiFi che è usata per la sola sincronizzazione dell'ora e ha acceso in modo stabile il led.
2. Si accende una seconda volpe del branco, il led lampeggia come nel caso della volpe master e poi si spegne. Mentre il led è spento, la volpe contatta la volpe Master e sincronizza il proprio orologio. Il led si riaccende per un istante per indicare che la sincronizzazione è finita e poi si spegne. A questo punto la volpe si è sincronizzata ed è in attesa del momento in cui inizierà a trasmettere. Da notare che a questo punto il led della volpe Master è ancora acceso mentre la seconda volpe ha il led spento. Questo indica che la Volpe Master è in attesa di essere contattata da un'altra volpe del branco (led acceso) mentre la seconda volpe è sincronizzata e pronta.
3. Si accende una terza volpe. Stesso comportamento della volpe due, ma alla fine della sincronizzazione quando la volpe tre spegne il led indicando che è pronta anche la volpe Master spegne il led indicando che è pronta a trasmettere.

   A questo punto tutte le volpi sono pronte/sincronizzate e quando è passato l'intervallo iniziale impostato nella variabile ***fox_start_minute*** le volpi iniziano in sequenza la loro trasmissione.

Uso della APP per impostare le variabile di cui sopra.

Installare la APP su di un telefono Android; non necessità di versioni di Android recenti, funziona anche con versioni datate (non preistoriche :-)) .
La distribuisco ASIS (così com'è) non ha ricercatezze grafiche o cose simili o bells and wishes dicono gli americani.

La APP ha nome FoxMcu e una volta selezionata si mostra come segue:



Dopo aver fatto il passo 1, la volpe Master ha il led acceso stabilmente; p**rima** di accendere le altre volpi :

1. Dal telefono cercare e connettersi alla rete WiFi con nome 'ARDF'.
2. Dalla APP premere il tasto Get (Leggi i dati). La APP si collega con la volpe Master e richiede alla stessa una copia del suo orologio interno, il minuti di attesa dopo l'accensione e il numero di volpi nel branco.
3. Nel caso si vogliano modificare questi due ultimi valori, dopo averli modificati premere il tasto Set (Scrivi i dati). La APP si collega alla volpe Master e spedisce i valori modificati. La volpe salve nelle EEPROM questi valori che saranno usati nelle prossime sincronizzazioni del branco.
4. Si procede dal passo 2 visto in precedenza per variabili statiche.

In breve, devo per prima cosa decidere se usare la modalità statica/dinamica per quanto riguarda il contenuto delle variabili usate dal programma. Devo in entrambe i casi aggiornare alla bisogna le righe che seguono ricompilare e caricare l'eseguibile in tutte le volpi.

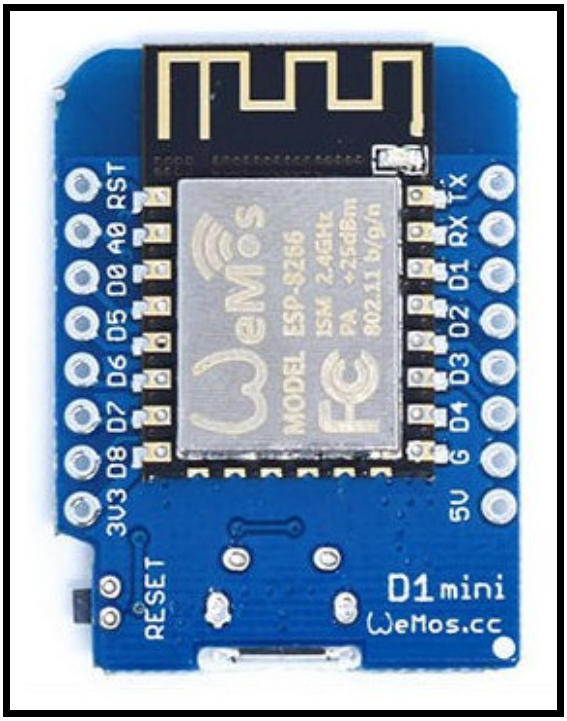
```
// ------------- CONFIGURATION VARIABLES ---------------
boolean use_fixed_parms = false;   // use fixed parms from the program
int     fox_max_number = 2;        // number of foxes in use
int     fox_start_minute = 1;      // minute at witch we start xmit
//-----------------------------------------------------------------------
```

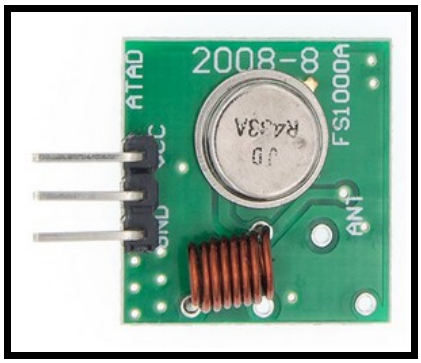
Le volpi vanno acce

se in sequenza, partendo dalla volpe Master, rispettando le tempistiche riportate sopra. Se non ci sono problemi, dopo che la volpe master ha acceso il led in modo stabile, è possibile accendere la volpe_2. Quando il led della volpe_2 ha finito di lampeggiare si può accendere la volpe_3 e così via.

Da notare che le volpi vanno accese rispettando questa sequenza ogni volta che vengono usate. Se per caso una delle volpi si spegne, la sequenza di avvio và ripetuta.

Esempio di moduli NodeMCu Mini D1 e TX saw a 433 Mhz usati per il progetto comperati da un noto venditore web.





Rilascio il sorgente del software usato per le volpi 'as-is' (cos'ì com'è).

Ovviamente per la trasmissione a 433 Mhz, vano rispettate leggi e regole vigenti.