
LoRa Beacon Manuale di Installazione Vr. 4.1.x

1	<i>Introduzione.....</i>	3
2	<i>Note di montaggio HW.....</i>	8
2.1	Versione LoRa_Beacon_2020_vr4_1 (alias mini-Tracker)	8
2.2	Versione LoRa_Beacon_2020_vr3_pcs4 (alias iGate)	11
2.3	Carrier per moduli radio LoRa (LoRa carrier)	16
2.4	Note di montaggio relative ad entrambe le versioni di PCB carrier.	21
3	<i>Installazione SW Iniziale.....</i>	24
3.1	Setup ambiente di caricamento SW e caricamento Immagine SW iniziale	25
3.2	Setup iniziale del dispositivo LoRa_Beacon (qualsiasi versione)	30
3.3	Setup sottosistema LoRa (qualsiasi versione)	35
3.4	Setup sottosistema APRS (qualsiasi versione)	36
3.4.1	Setup sottosistema APRS per modalità iGate e Tracker	36
3.4.2	Setup sottosistema APRS per connessione ad un server di servizio	39
3.5	Setup APRS Compatibility (qualsiasi versione)	39
3.6	Schermata Dashboard (qualsiasi versione)	41
3.7	HW Setup Configuration	44
3.8	Setup syslog logging (qualsiasi versione)	45
3.9	Setup sottosistema MQTT (qualsiasi versione)	45
3.10	Setup sottosistema Beaconing Nativo (qualsiasi versione)	46
4	<i>Interfaccia di Debug Remoto</i>	49
4.1	Accesso alla IF di Remote Debug tramite Telnet Client	49
4.1.1	Comando “gps_status”.....	51
4.1.2	Comando “temperature”	52
4.1.3	Comando “wifi_scan”.....	52
4.1.4	Comando “display_config”.....	52
4.1.5	Comando “show_stats”.....	53
4.1.6	Comando “show_events”.....	53
4.1.7	Comando “log_display”.....	53
4.2	Accesso alla IF di Remote Debug tramite Web App	55
5	<i>Porting del SW LoRa Beacon su altre piattaforme HW</i>	58
5.1	Installazione SW su dispositivo TTGO T-beam-V1-2019	58
5.2	Installazione SW su dispositivo Heltec_wifi_lora32	61
6	<i>Installazione SW via OTA (Over-The-Air).....</i>	63
7	<i>Salvataggio e Caricamento della configurazione via OTA.....</i>	66
8	<i>Esempi di dove acquistare la componentistica</i>	71

1 Introduzione

Il presente documento descrive le modalità di montaggio HW e di installazione del SW dei dispositivi basati sul progetto LoRa_Beacon sviluppato nel contest della sperimentazione SARIMESH (rif. <http://www.sarmesh.net>)

L'obiettivo di questo progetto è quello di generare una piattaforma HW/SW utilizzabile per attività di **sperimentazione relativamente alla tecnologia LoRa applicata ad attività di tipo radioamatoriali** e quindi non necessariamente utilizzando le linee guida e le implementazioni inizialmente pensate per questa tecnologia in ambito IoT (Internet of Things).

Va la pena di ricordare che la tecnologia LoRa è nata principalmente per consentire la realizzazione di dispositivi di tipo “sensore” ovvero per delle applicazioni di tipo M2M (Machine-To-Machine communication) caratterizzate da una limitata esigenza di comunicazione in termini di quantità di dati trasmessi, ma con l'obiettivo di realizzare la massima indipendenza da fonti di alimentazione esterne, cercando di utilizzare delle batterie per l'alimentazione dei dispositivi con un obiettivo di autonomia dell'ordine degli anni.

Come risultato di questi requisiti funzionali il protocollo radio LoRa ha mirato a ridurre la potenza di trasmissione dei dispositivi a parità di **“link budget radio”** disponibile ovvero aumentando tale valore a parità di potenza trasmessa; la chiave per ottenere tale obiettivo è l'utilizzo di una metodica di trasmissione a spettro disperso che consentisse di ottenere un significativo “guadagno di processo” .

Quindi nell'ambito dell'IoT un tema che assume enorme importanza è proprio la riduzione del consumo energetico dei dispositivi, in previsione del loro utilizzo come sensori remoti non alimentati da fonti di energia esterne.

Nell'uso radioamatoriale ovviamente questo aspetto passa in secondo piano mentre assumono rilevanza soprattutto gli aspetti di massimizzazione delle prestazioni in termini di “link budget radio disponibile”.

E' stato quindi deciso nella nostra sperimentazione di trascurare, almeno inizialmente, gli aspetti di minimizzazione dei consumi energetici, inessenziali ai fini dei nostri impieghi.

Il target di utilizzatori è stato quindi pensato per essere il radioamatore medio desideroso di sperimentare con poca spesa le nuove tecnologie HW/SW collegate in particolare a questa particolare tipologia di trasmissione radio.

La soluzione scelta si basa sul riutilizzo di “blocchi funzionali” HW e SW già disponibili allo scopo di semplificare e rendere economica la creazione di nuove configurazioni da sperimentare, con un minimo di sviluppi HW e SW.

La piattaforma HW è basata sul concetto di “carrier board”, ovvero di un circuito stampato destinato ad assemblare una serie di altri blocchetti funzionali a loro volta realizzati tramite piccoli circuiti stampati acquistabili sulle classiche piattaforme di e-commerce a prezzi molto convenienti.

Per agevolare l'intercambiabilità di alcuni moduli la soluzione è caratterizzata dalla possibilità di ospitare moduli HW anche fisicamente differenti a parità di funzione svolta: in

questo caso la carrier board viene dotata di diverse “impronte” atte a collegare diverse varianti di un certo blocco in maniera opzionale (ad es. montare diversi moduli GPS disponibili sul mercato, oppure diversi moduli LoRa sempre normalmente acquistabili sulle piattaforme di e-commerce tipo aliexpress).

Dal punto di vista della tipologia di montaggio si è cercato di utilizzare inizialmente la modalità “Pin in Hole” limitando l’uso di componentistica SMD solo a casi eccezionali. Anche questa scelta è stata ovviamente motivata dall’obiettivo di rendere il più agevole possibile il montaggio dei circuiti evitando di richiedere conoscenze specifiche o particolari abilità per il montaggio.

Dal punto di vista SW si è cercato di utilizzare una metodologia simile cercando di usare anche a questo livello “**blocchi funzionali SW**” già disponibili nell’ambito dell’open source, limitando i nuovi sviluppi allo stretto indispensabile.

Una nota particolare a tale riguardo è lo stile di programmazione utilizzato: essendo il target un tipico radioamatore si è cercato di evitare di usare uno stile di programmazione particolarmente “aulico” cercando di scrivere del codice agevolmente comprensibile anche da parte di persone che non siano dei bravi programmatori professionisti... l’obiettivo era e resta quello di incoraggiare anche gli inesperti a cimentarsi e l’utilizzo di tecniche di programmazione particolarmente avanzate è l’esatto opposto di ciò che serve per abbassare la soglia di ingresso a questo tipo di tematiche applicative.

L’aspetto SW è ovviamente strettamente legato alla scelta HW di una specifica tipologia di processore da utilizzare: la scelta è caduta sull’uso del **processore ESP32** che nell’ambito dei microcontrollori rappresenta una soluzione ottimale in quanto fornisce ad un costo estremamente contenuto una **piattaforma di processo molto potente** in grado di operare in modalità “**multiprocessing**” con caratteristiche anche di tipo “**real time**”, supportato da un **sistema operativo molto leggero ed efficiente (FreeRTOS)** e in grado di essere supportato da un **ambiente di sviluppo basato sulla piattaforma SW Arduino**.

Nella versione 4.x si è però deciso di abbandonare la piattaforma Arduino assumendo come nuova piattaforma di riferimento PlatformIO che nel frattempo si è affermata come la piattaforma più utilizzata per questa tipologia di applicazioni e peraltro estremamente più avanzata di Arduino e alla fine meno complessa come setup ed utilizzo.

Grazie a queste scelte la soluzione si presenta molto “user friendly” e consente di sviluppare agevolmente del nuovo SW senza richiedere una curva di apprendimento eccessivamente lunga a chi si voglia cimentare in tale direzione.

Un aspetto molto curato è stato quello della **gestione dei dispositivi**; si è cercato anche qui di evitare ai potenziali utilizzatori di dover necessariamente passare tramite l’installazione completa dell’ambiente di sviluppo SW (strada tipicamente richiesta dai progetti basati su microcontrollori) fornendo **una semplice interfaccia grafica**, agevolmente customizzabile, tramite la quale impostare molte delle funzionalità presenti, consentendo quindi di poter realizzare della sperimentazione, a livello delle funzionalità radio, senza dover necessariamente passare per la fase di sviluppo SW.

Nel seguito vengono presentate innanzitutto alcune note relative al montaggio di un prototipo delle due versioni HW attualmente disponibili, per poi passare alla descrizione del caricamento di una immagine SW sui prototipi stessi per finire con alcune note relative alla configurazione, tramite interfaccia grafica, dei principali parametri di funzionamento.

Un tema collaterale trattato alla fine del documento è il tema del “debug” HW/SW, delle funzionalità presenti a tale fine nel progetto e delle modalità di testing dell’insieme HW/SW in caso di sviluppo di nuovo SW.

A partire dalla versione SW Vr 1.0.9.2 sono state introdotte una serie di piccole modifiche per rendere possibile l’utilizzazione della componente SW del progetto non solo su piattaforme HW SARIMESH, ma anche sui classici “schedini cinesi”, attualmente molto utilizzati per applicazioni LoRa e che montano ancora quasi sempre chips LoRa di prima generazione.

Le funzioni aggiunte hanno riguardato in particolare il supporto dei chips LoRa di prima generazione, introducendo una procedura per semplificare la fase di tuning della frequenza utilizzata in RX/TX in modo da consentire l’uso di questi dispositivi anche con valori di larghezza di banda inferiore a 62.5 KHz, e la possibilità di modificare da GUI una serie di parametri legati alla architettura HW dei circuiti, ovvero in particolare la possibilità di modificare da GUI i pins utilizzati dei vari chips presenti, in modo da adeguarsi alle specificità dei vari schedini.

Purtroppo il supporto di dispositivi HW di diverse sorgenti pone ovviamente il problema di avere a disposizione dei prototipi dei vari dispositivi su cui provare le varie versioni del SW; ovviamente questo richiede un impegno economico per procurarsi i vari campioni ed un onere di tempo di sviluppo e test per effettuare le varie regression test di una nuova versione SW sui vari dispositivi.

Con la versione 4.x abbiamo deciso quindi di interrompere il supporto degli schedini TTGO like per l’evidenza che esistono svariatissime versioni di tali schedini spesso abbastanza diversi tra loro; questo ovviamente non significa che le nuove versioni di SW non possono girare su questi dispositivi, ma semplicemente che da parte nostra ci limiteremo a testare al meglio tali SW sui campioni che abbiamo a disposizione, lasciando il test su ogni specifico dispositivo alle persone direttamente interessate, che potranno eventualmente condividere le loro esperienze tramite il sistema GITHUB.

Una ulteriore funzionalità introdotta a partire dalla stessa release SW sopra citata è il supporto, in aggiunta alla modalità di incapsulamento AX.25 per i pacchetti dati APRS, della modalità di incapsulamento utilizzata in altre implementazioni LoRa APRS attualmente esistenti e denominata OE_Style; in particolare i dispositivi che montano queste release di SW possono ricevere traffico APRS indifferentemente caratterizzati da incapsulamento AX.25 o OE_Style; il traffico LoRa uscente potrà, da interfaccia GUI, essere impostata per essere dotato dell’uno o dell’altro tipo di incapsulamento.

Questa feature consente di utilizzare in una stessa rete LoRa APRS sia dispositivi che utilizzano SW SARIMESH che dispositivi che utilizzano altre implementazioni LoRa APRS utilizzando l’incapsulamento OE_Style.

Con la versione SW 4.x sono state introdotte una serie di altre funzionalità tra cui citiamo in particolare l’utilizzabilità del meccanismo di compressione standard APRS per l’indicazione della “location” relativo ad ogni spot, abilitabile da GUI e con la possibilità di ricevere e trattare sia pacchetti con location compressa che pacchetti con location in chiaro.

La suddetta modalità è una delle varie modifiche che sono state introdotte allo scopo di minimizzare la lunghezza di pacchetto risultante e poter comunque avere la possibilità di tracciamento dei percorsi radio con riporto delle condizioni di ricezione sui nodi radio o parte dei nodi radio attraversati.

Una ulteriore funzionalità introdotta in Vr. 4.x è la funzione di “Agile Beacons” ovvero una modalità di generazione e invio di pacchetti beacon con delle modalità settabili da GUI allo scopo di evidenziare e rilevare in maniera mirata alcuni parametri radio.

In particolare accanto alla classica modalità base di invio di pacchetti a intervallo di tempo costante, sono affiancate una modalità di invio a “spazio percorso” prefissato e selezionabile da GUI, ed una modalità definita “ad evento” nella quale vengono definite delle condizioni di velocità, orientamento e spazio percorso che triggerano la generazione di un pacchetto di beacon.

Una ulteriore funzione introdotta è la possibilità di escludere un insieme di “zone” geografiche dall’invio di beacon principalmente per scopi di privacy o per mitigare eventuali problemi di congestione radio in particolari zone quali ad es. un centro abitato.

Con la versione 4.x sono anche state aggiunte una serie di ulteriori funzioni a livello di gestione e manutenzione anche remota dei dispositivi.

Una prima funzione è il supporto del protocollo syslog verso un server remoto allo scopo di raccogliere, principalmente in fase di testing e di diagnostica, una serie di informazioni in modo permanente in modo da poter agevolmente tracciare il comportamento di un dispositivo; questa funzione richiede ovviamente che il dispositivo sotto test sia in grado di raggiungere direttamente internet per es. tramite una rete wifi locale (ad es. tramite un telefonino operante in modalità hotspot...).

Una ulteriore funzione è il supporto del protocollo MQTT verso un server (broker) remoto sempre tramite connettività internet.

Questa funzione può essere utilizzata con due modalità: in una prima modalità il sistema consente di effettuare della operatività remota su un dispositivo LoRa Sarimesh senza necessità di effettuare alcuna operazione locale sul dispositivo che fornisce la connettività internet: ad es. è possibile accedere ad alcune funzioni da remoto o anche effettuare delle operazioni sul dispositivo (ad. es. far ripartire il dispositivo o triggerare altre funzioni ad hoc).

Una seconda modalità di utilizzo del protocollo MQTT è quella classica sfruttata nelle applicazioni IoT per consentire il riporto verso server remoti di dati locali quali ad es. misurazione di temperatura, umidità o altri tipi di parametri.

Dal punto di vista poi della usabilità è stata notevolmente migliorata la gestione delle fasi di inizializzazione di un dispositivo e di setup di alcuni modi operativi senza ausilio di alcun dispositivo ausiliario (ad. es. un PC o un telefonino).

La strategia implementata si basa sull’utilizzo del tastino e dei due led di colore rosso e verde presenti sui dispositivi HW sarimesh (o equivalenti in caso di dispositivi HW diversi), per realizzare una semplice e primitiva forma di interfaccia di controllo che appunto sfrutta l’accensione dei led o alcuni semplici messaggi presentati sul display per consentire di attivare prefissate funzioni.

La logica base è che tendo premuto il tastino si attiva una funzione di scanning di varie opzioni presentate sul display e che csi suggeriscono a intervalli di tempo di qualche secondo... rilasciando il tastino quando una certa funzione è presentata sul display equivale a selezionare quella funzione...

Un banale esempio è per es. il voler inviare manualmente un beacon.... premendo il tastino viene presentata come prima opzione proprio quella di invio di un beacon... rilasciando il tastino viene inviato il beacon.

Altre funzioni attivabili tramite questo meccanismo sono l'accensione e lo spegnimento della connessione Upstream WiFi e la possibilità di far ripartire (reboot) un dispositivo. Volendo si possono inserire ulteriori funzioni agendo sul codice del SW.

Un caso molto particolare è l'operazione di restore della configurazione di fabbrica di un certo dispositivo.

Ogni dispositivo appena caricato il SW per la prima volta si viene a trovare in una precisa situazione che chiameremo di "factory default": in pratica il dispositivo si troverà impostato per operare nella funzione di tracker e con dei parametri di personalizzazione di esempio: tali parametri sono ovviamente da cambiare immediatamente per un concreto utilizzo, ma permettono subito di avere un dispositivo in grado di funzionare dando dei segni di vita e mettendo in grado l'utilizzatore di impostare i propri parametri personali direttamente dalla interfaccia GUI accessibile per esempio tramite un cellulare o un PC.

Il tema classico è ovviamente come fare a ritornare ai valori di default di fabbrica a valle della prima attivazione del SW.... infatti è facile che si creino delle situazioni tali che il dispositivo risulti non raggiungibile o che il dispositivo per qualche motivo non riesca a partire correttamente.

La soluzione a questo problema è stata implementata sfruttando l'accensione simultanea dei due led rosso e verde subito a valle dell'accensione del dispositivo: in pratica quando un dispositivo viene acceso dopo qualche secondo viene attivata una "finestra temporale" di alcuni secondi, segnalata dall'accensione permanente dei due led, durante la quale una pressione del tastino presente sul dispositivo viene interpretata come richiesta di reset ai defaults di fabbrica... in questo modo sarà possibile sempre forzare la ripartenza del dispositivo in condizioni prefissate e da cui procedere alla personalizzazione della configurazione del dispositivo stesso.

Qualora il SW in questione venga installato su dispositivi HW diversi da quello Sarimesh ovviamente si dovrà tener conto della presenza o meno di un tastino e di qualche led....

Nel caso di schedini tipo TTGO è facile in quanto esiste un tastino e un led di colore blue che viene mappato per operare come entrambi i due led rosso e verde dell'HW Sarimesh.

Nel caso di schedini HELTEC che sono privi di un tastino funzionalmente destinato ad un uso diverso dal semplice reset purtroppo sarà necessario per ripristinare i default di fabbrica procedere al ricaricamento da scratch di una immagine SW completa.

2 Note di montaggio HW

Allo stato esistono due varianti HW del circuito LoRa_Beacon: le due varianti si differenziano per il tipo di moduli fisici/funzionali supportati e per le dimensioni fisiche del prodotto finale, fermo restando il supporto delle funzioni essenziali per consentire l'utilizzo di entrambi i tipi di PCB per realizzare le principali funzioni (Tracker ed iGate APRS LoRa).

2.1 Versione LoRa_Beacon_2020_vr4_1 (alias mini-Tracker)

Le figure a seguire rappresentano lo schema elettrico e il PCB di questa versione, che si caratterizza per essere di **dimensioni molto contenute e destinata ad un uso mobile**.

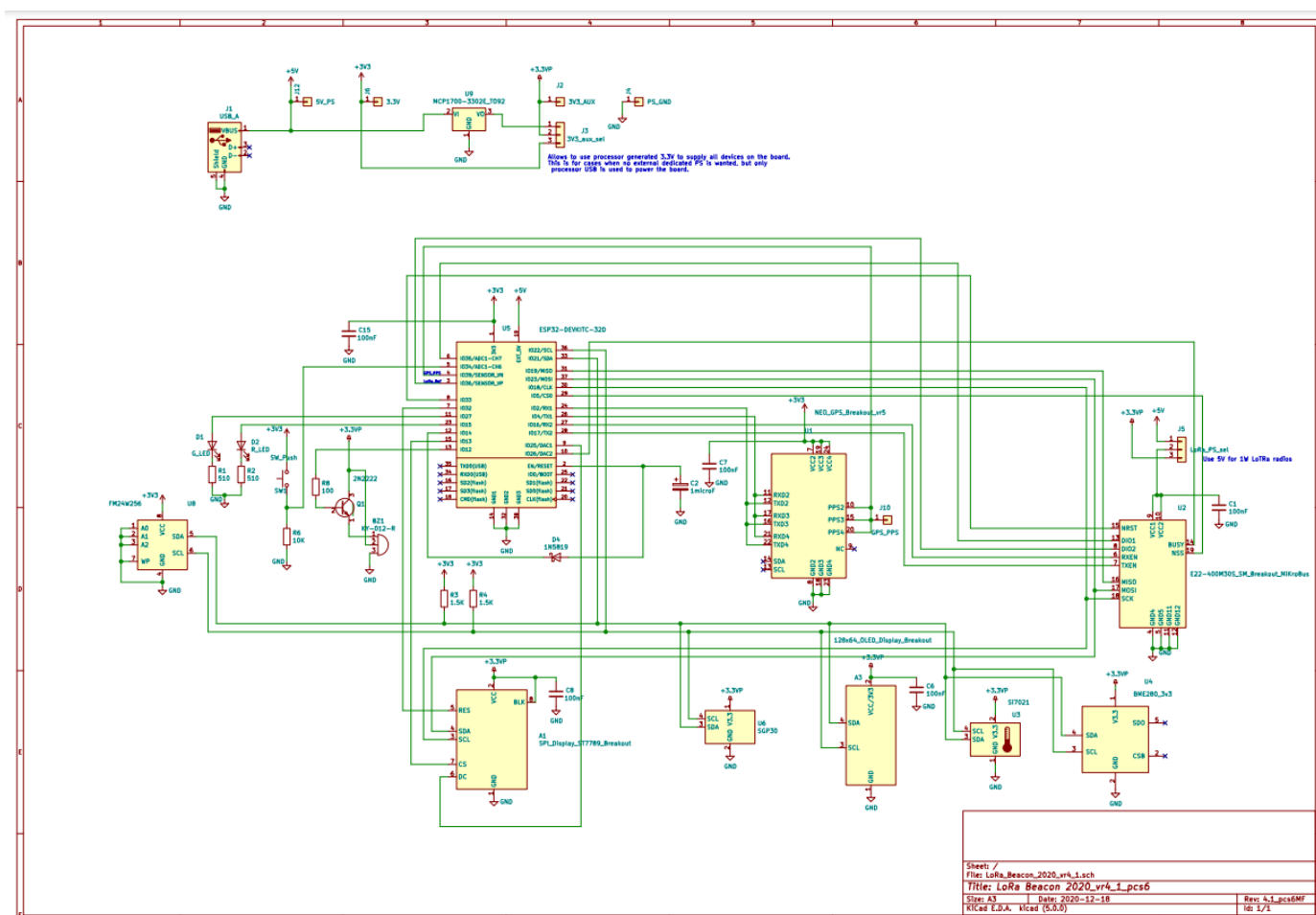


Figura 1 Schema elettrico Versione LoRa_Beacon_2020_vr4.1_pcs6

Questa versione è prevista per essere alimentata tramite un cavo USB (quindi a 5V) utilizzando una qualsiasi fonte tipo alimentatore di telefono cellulare, presa USB di auto o un qualsiasi powerbank previsto per alimentare un classico telefono cellulare.

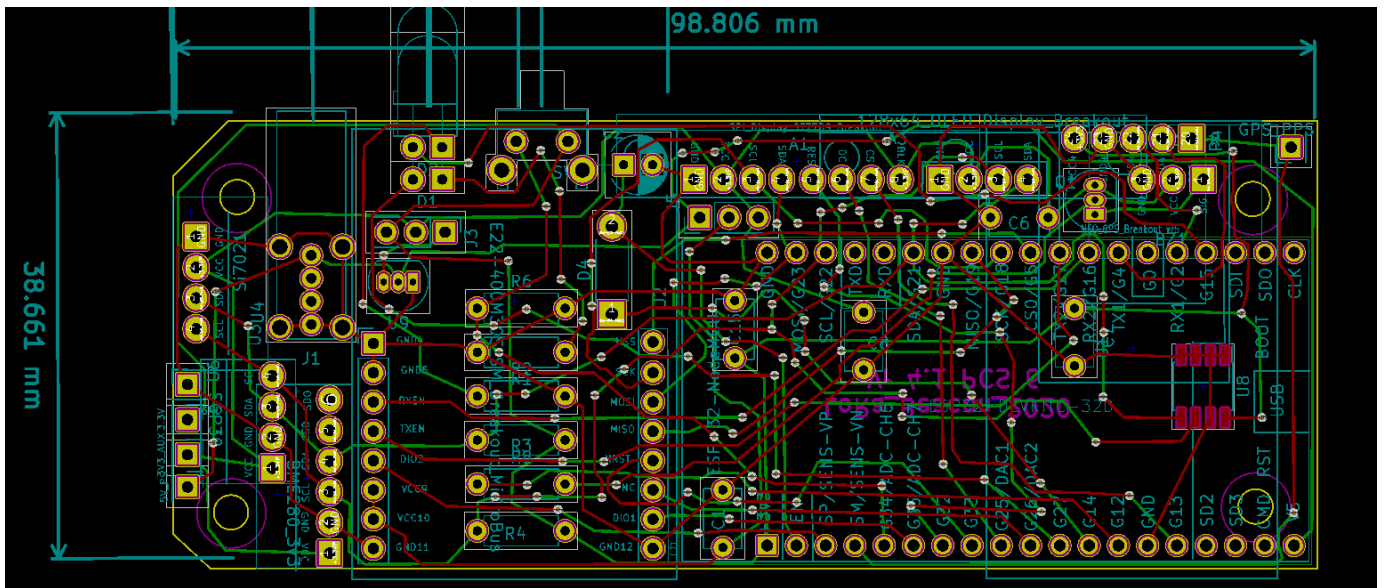


Figura 2 Layout LoRa_beacon_2020_Vr_4.1_pcs6

A seguire la part list del circuito

1	A1 - SPI_Display_ST7789_Breakout : APRS_Mini_Tracker:SPI_Display_ST7789_breakout
2	A3 - 128x64_OLED_Display_Breakout : APRS_Mini_Tracker:128x64_OLED_Vert_Display_breakout
3	BZ1 - KY-012-R : APRS_Mini_Tracker:KY-012-R_breakout
4	C1 - 100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
5	C2 - 1microF : Capacitors_THT:CP_Radial_D5.0mm_P2.50mm
6	C6 - 100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
7	C7 - 100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
8	C8 - 100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
9	C15 - 100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
10	D1 - G_LED : APRS_Mini_Tracker:LED_D5.0mm_Horizontal_O3.81mm_Z5.0mm
11	D2 - R_LED : APRS_Mini_Tracker:LED_D5.0mm_Horizontal_O3.81mm_Z5.0mm
12	D4 - 1N5819 : Diodes_THT:D_DO-41_SOD81_P7.62mm_Horizontal
13	J1 - USB_A : USB_A:USB_A_Vertical
14	J2 - 3V3_AUX : APRS_Mini_Tracker:Pin_Header_Straight_1x01_Pitch2.54mm_local
15	J3 - 3V3_aux_sel : Pin_Headers:Pin_Header_Straight_1x03_Pitch2.54mm
16	J4 - PS_GND : APRS_Mini_Tracker:Pin_Header_Straight_1x01_Pitch2.54mm_local
17	J5 - LoRa_PS_sel : Pin_Headers:Pin_Header_Straight_1x03_Pitch2.54mm
18	J6 - 3.3V : APRS_Mini_Tracker:Pin_Header_Straight_1x01_Pitch2.54mm_local
19	J10 - GPS_PPS : APRS_Mini_Tracker:Pin_Header_Straight_1x01_Pitch2.54mm_local
20	J12 - 5V_PS : APRS_Mini_Tracker:Pin_Header_Straight_1x01_Pitch2.54mm_local
21	Q1 - 2N2222 : TO_SOT_Packages_THT:TO-92_Inline_Narrow_Oval
22	R1 - 510 : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
23	R2 - 510 : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
24	R3 - 1.5K : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
25	R4 - 1.5K : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
26	R6 - 10K : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
27	R8 - 100 : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
28	SW1 - SW_Push : APRS_Mini_Tracker:Button_Angled
29	U1 - NEO_GPS_Breakout_vr5 : APRS_Mini_Tracker:NEO_GPS_Breakout_vr7
30	U2 - E22-400M30S_SM_Breakout_MiKroBus : APRS_Mini_Tracker:E22-M40030S_SM3_MiKroBus_Breakout_STACKED
31	U3 - Si7021 : APRS_Mini_Tracker:SI-7021_breakout
32	U4 - BME280_3v3 : APRS_Mini_Tracker:BME-280_breakout
33	U5 - ESP32-DEVKITC-32D : APRS_Mini_Tracker:MODULE_ESP32-DEVKITC-32D_STACKED
34	U6 - SGP30 : APRS_Mini_Tracker:SGP30_breakout
35	U8 - FM24W256 : Housings_SOIC:SOIC-8_3.9x4.9mm_Pitch1.27mm
36	U9 - MCP1700-3302E_TO92 : TO_SOT_Packages_THT:TO-92_Inline_Narrow_Oval

Figura 3 Part list LoRa_Beacon_2020_vr_4.1_pcs6

La figura seguente fornisce una vista forse più leggibile del posizionamento dei vari componenti sul PCB.

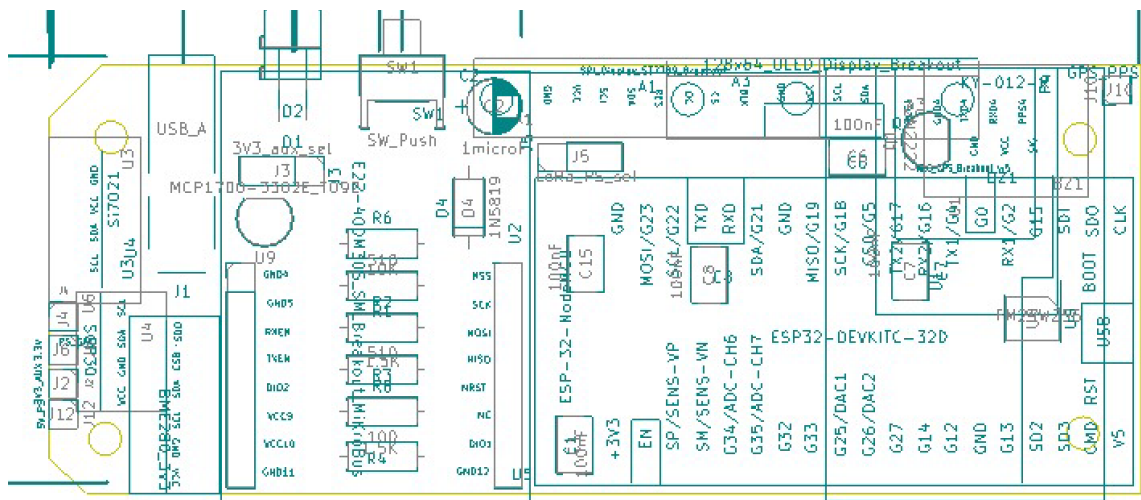


Figura 4 LoRa_Beacon Vr. 4.1-pcs6 : posizionamento componenti principali

La figura seguente fornisce un dettaglio del montaggio dell'unico chip SMD presente sul circuito; il pin 1 corrisponde ad un piccolo forellino presente sul corpo di plastica del dispositivo U8.

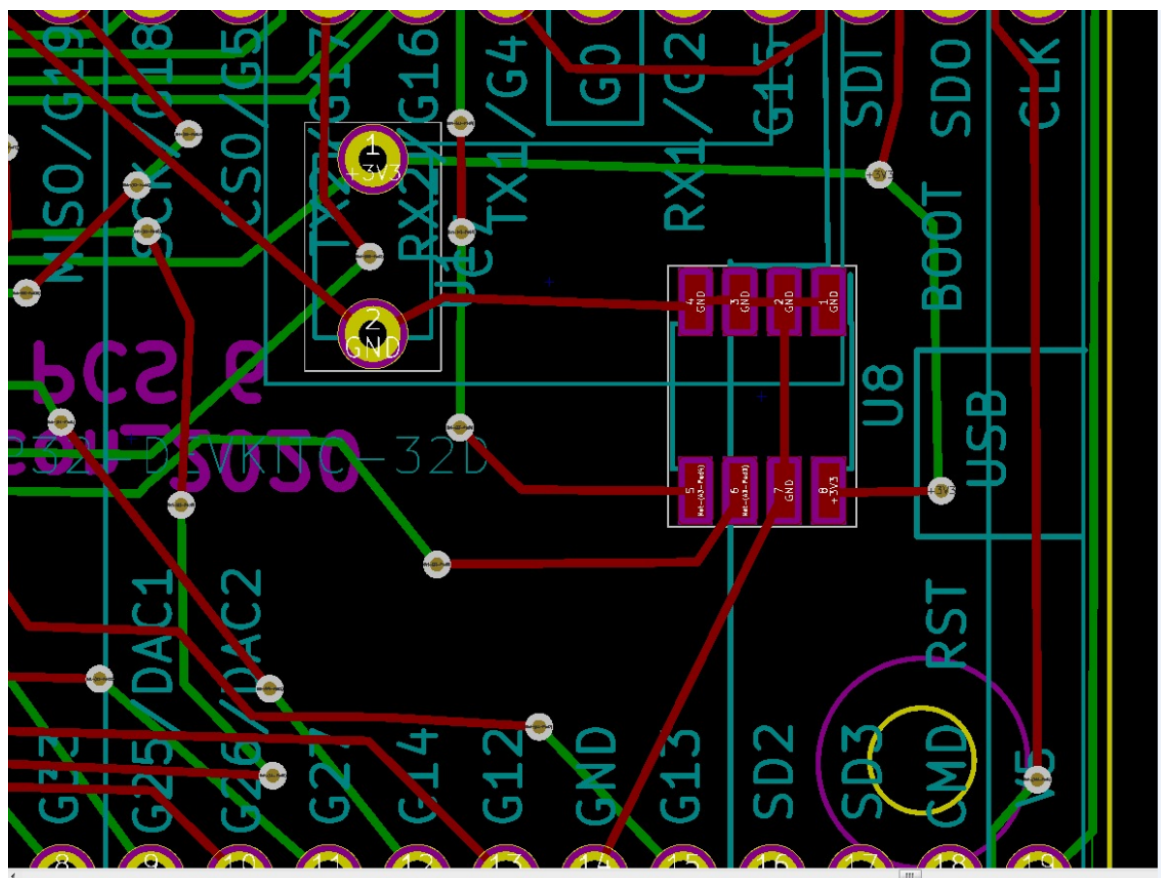


Figura 5 LoRa_Beacon Vr. 4.1_pcs6 : dettaglio montaggio FRAM U8 SMD

La foto seguente riporta una vista del lato componenti del PCB di questa versione

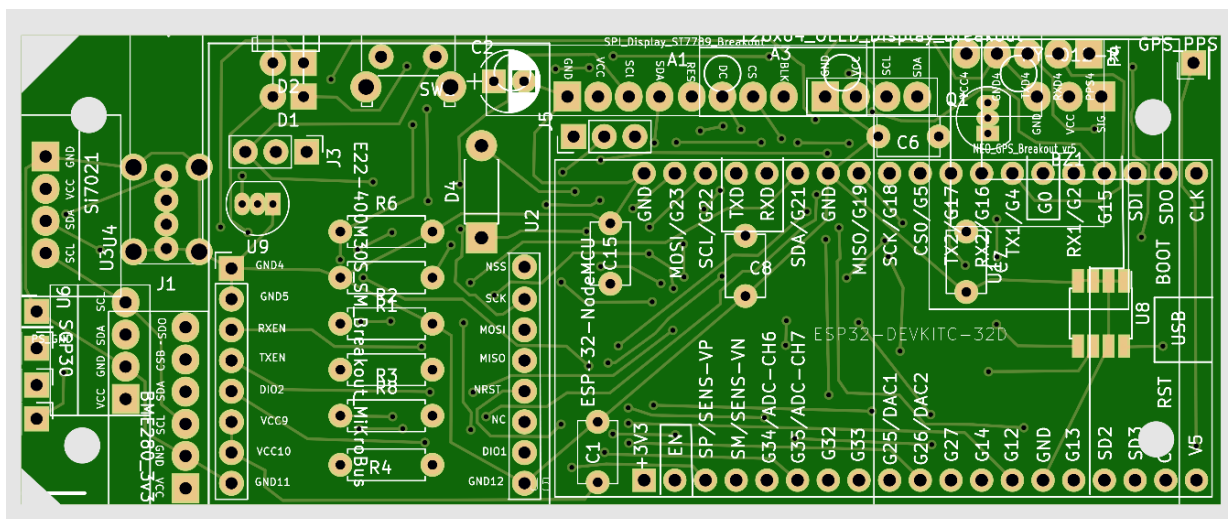


Figura 6 : LoRa_Beacon Vr 4.1_pcs6 foto lato superiore del circuito stampato

2.2 Versione LoRa_Beacon_2020_vr3_pcs4 (alias iGate)

Le figure a seguire rappresentano lo schema elettrico e il PCB di questa versione, che si caratterizza per essere di dimensioni maggiori della versione precedente e che presenta la possibilità di montare una serie di moduli aggiuntivi per consentire una sperimentazione più avanzata rispetto alla versione precedente.

In particolare questa versione si presta in maniera ottimale per essere usata in una installazione fissa ed in particolare per svolgere le funzionalità di iGate; infatti è prevista l'alimentazione a 12V, presenta la possibilità di connettersi ad internet sia in modalità WiFi che tramite interfaccia LAN, e monta un dispositivo RTC (Real Time Clock) per garantire la possibilità di tenuta del tempo locale anche in assenza di connessione ad un dispositivo GPS o alla presenza di connettività internet.

Le dimensioni del PCB sono maggiori della versione precedente in ragione dei moduli aggiuntivi installabili, ma per le parti presenti anche nella versione precedente esiste completa intercambiabilità dei moduli comuni; a livello di schema elettrico le similitudini sono ovviamente elevatissime anche se la numerazione e denominazione dei vari componenti sono diverse.

A seguire i dettagli di questa versione.

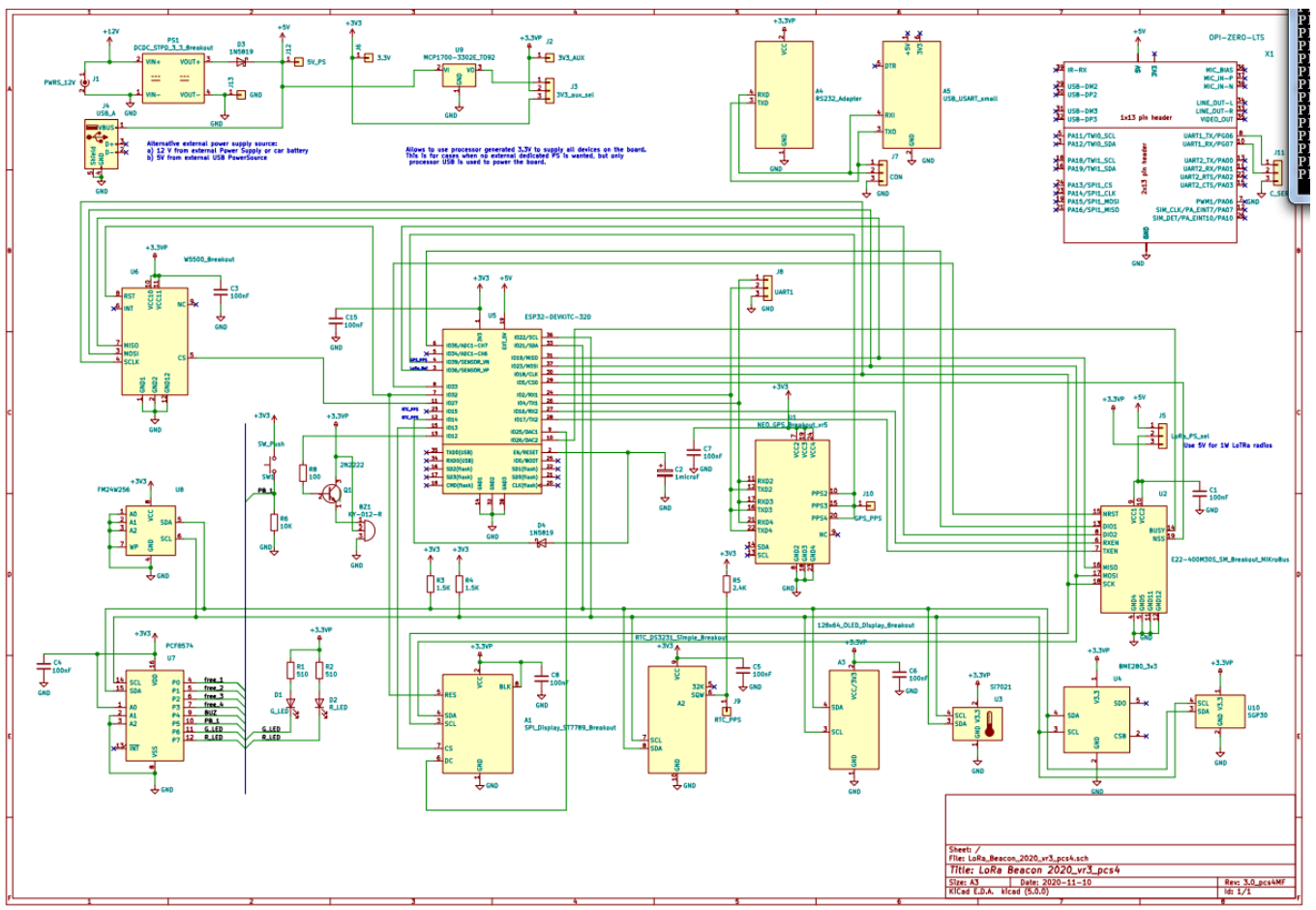


Figura 7 LoRa_Beacon_2020_vr3_pcs4 : schema elettrico



LoRa_Tracker – Vr. 4.1.x

1	A1 -	SPI_Display_ST7789_Breakout : APRS_Mini_Tracker:SPI_Display_ST7789_breakout
2	A2 -	RTC_DS3231_Simple_Breakout : APRS_Mini_Tracker:RTC_DS3231_breakout_simpl
3	A3 -	128x64_OLED_Display_Breakout : APRS_Mini_Tracker:128x64_OLED_Vert_Display_breakout
4	A4 -	RS232_Adapter : APRS_Mini_Tracker:RS232_adapter
5	A5 -	USB_USART_small : APRS_Mini_Tracker:USB_USART_small
6	BZ1 -	KY-012-R : APRS_Mini_Tracker:KY-012-R_breakout
7	C1 -	100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
8	C2 -	1microF : Capacitors_THT:CP_Radial_D5.0mm_P2.50mm
9	C3 -	100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
10	C4 -	100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
11	C5 -	100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
12	C6 -	100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
13	C7 -	100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
14	C8 -	100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
15	C15 -	100nF : Capacitors_THT:C_Disc_D5.1mm_W3.2mm_P5.00mm
16	D1 -	G_LED : APRS_Mini_Tracker:LED_D5.0mm_Horizontal_O3.81mm_Z5.0mm
17	D2 -	R_LED : APRS_Mini_Tracker:LED_D5.0mm_Horizontal_O3.81mm_Z5.0mm
18	D3 -	1N5819 : Diodes_THT:D_DO-41_SOD81_P2.54mm_Vertical_KathodeUp
19	D4 -	1N5819 : Diodes_THT:D_DO-41_SOD81_P2.54mm_Vertical_KathodeUp
20	J1 -	PWRS_12V : Connectors:JACK_ALIM
21	J2 -	3V3_AUX : APRS_Mini_Tracker:Pin_Header_Straight_1x01_Pitch2.54mm_local
22	J3 -	3V3_aux_sel : Pin_Headers:Pin_Header_Straight_1x03_Pitch2.54mm
23	J4 -	USB_A : USB_A:USB_A_Vertical
24	J5 -	LoRa_PS_sel : Pin_Headers:Pin_Header_Straight_1x03_Pitch2.54mm
25	J6 -	3.3V : APRS_Mini_Tracker:Pin_Header_Straight_1x01_Pitch2.54mm_local
26	J7 -	CON : Pin_Headers:Pin_Header_Straight_1x03_Pitch2.54mm
27	J8 -	UART1 : Pin_Headers:Pin_Header_Straight_1x03_Pitch2.54mm
28	J9 -	RTC_PPS : APRS_Mini_Tracker:Pin_Header_Straight_1x01_Pitch2.54mm_local
29	J10 -	GPS_PPS : APRS_Mini_Tracker:Pin_Header_Straight_1x01_Pitch2.54mm_local
30	J11 -	C_SER : Pin_Headers:Pin_Header_Straight_1x03_Pitch2.54mm
31	J12 -	5V_PS : APRS_Mini_Tracker:Pin_Header_Straight_1x01_Pitch2.54mm_local
32	J13 -	GND : APRS_Mini_Tracker:Pin_Header_Straight_1x01_Pitch2.54mm_local
33	PS1 -	DCDC_STPD_3_3_Breakout : APRS_Mini_Tracker:DCDC_STPD_3_3_TOP_Breakout
34	Q1 -	2N2222 : TO_SOT_Packages_THT:TO-92_Inline_Narrow_Oval
35	R1 -	510 : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
36	R2 -	510 : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
37	R3 -	1.5K : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
38	R4 -	1.5K : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
39	R5 -	2.4K : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
40	R6 -	10K : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
41	R8 -	100 : Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal
42	SW1 -	SW_Push : APRS_Mini_Tracker:Button_Angled
43	U1 -	NEO_GPS_Breakout_vr5 : APRS_Mini_Tracker:NEO_GPS_Breakout_vr5
44	U2 -	E22-400M30S_SM_Breakout_MiKroBus : APRS_Mini_Tracker:E22-M40030S_SM3_MiKroBus_Breakout
45	U3 -	Si7021 : APRS_Mini_Tracker:SI-7021_breakout
46	U4 -	BME280_3v3 : APRS_Mini_Tracker:BME-280_breakout
47	U5 -	ESP32-DEVKITC-32D : APRS_Mini_Tracker:MODULE_ESP32-DEVKITC-32D
48	U6 -	W5500_Breakout : APRS_Mini_Tracker:W5500_Breakout
49	U7 -	PCF8574 : APRS_Mini_Tracker:PCF8574_DIP
50	U8 -	FM24W256 : Housings_SOIC:SOIC-8_3.9x4.9mm_Pitch1.27mm
51	U9 -	MCP1700-3302E_TO92 : TO_SOT_Packages_THT:TO-92_Inline_Narrow_Oval
52	U10 -	SGP30 : APRS_Mini_Tracker:SGP30_breakout
53	X1 -	OPI-ZERO-LTS : APRS_Mini_Tracker:Controller_Vr1

Figura 9 LoRa_Beacon_2020_vr3_pcs4 : part list

A seguire una vista del posizionamento dei componenti:

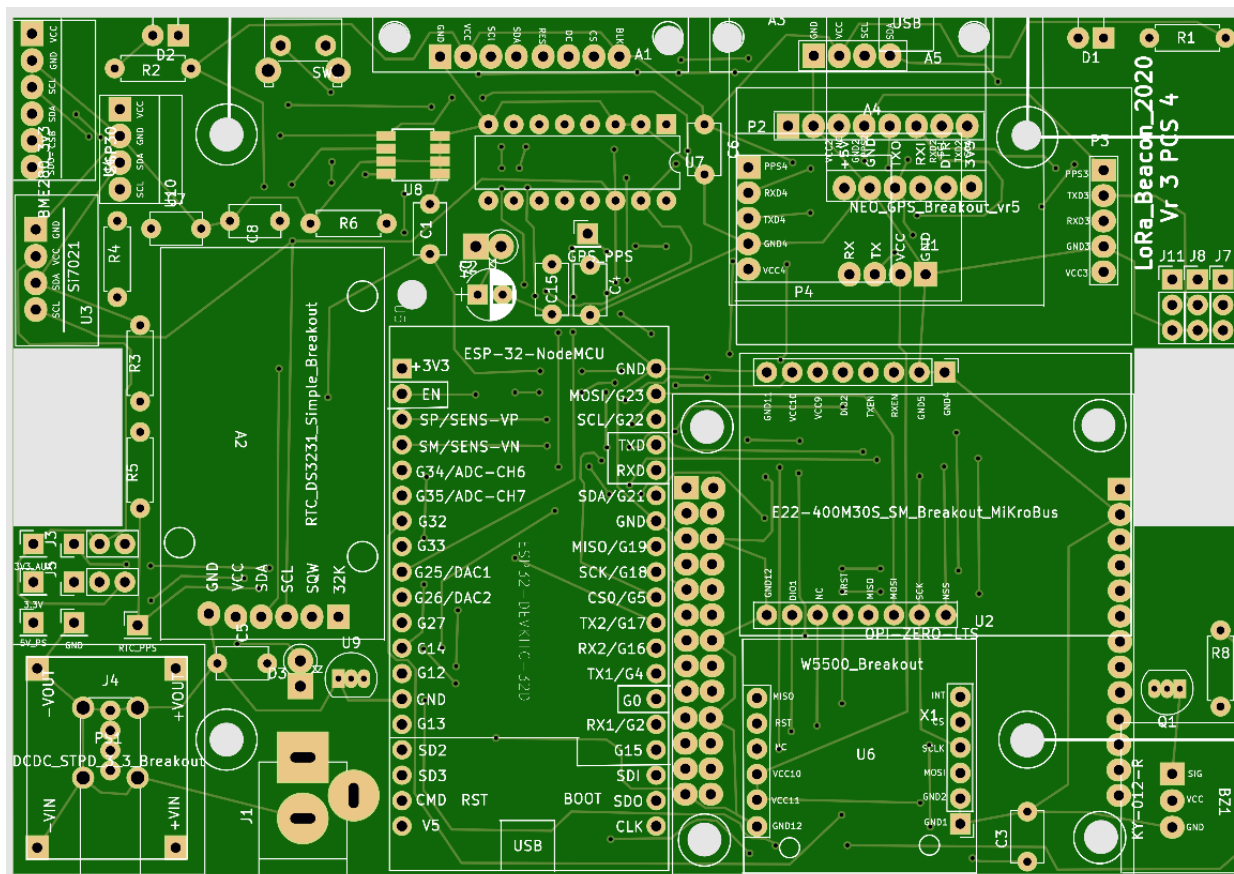


Figura 12 . LoRa_Beacon_Vr 3 pcs 4 : foto lato superiore PCB

2.3 Carrier per moduli radio LoRa (LoRa carrier)

I moduli radio LoRa disponibili sul mercato hanno purtroppo una notevole diversità di pin layout e formato fisico, in genere quasi sempre non compatibile con una pinnatura del tipo 2.54mm che caratterizza la maggioranza degli altri moduli richiesti per la nostra implementazione.

Da questa evidenza è nata l'esigenza di progettare un carrier ad hoc con cui adattare il pinout dei vari moduli LoRa presenti sul mercato ad un unico pinout a spaziatura 2.54mm da usare per montare tali moduli sul carrier principale.

Allo stato esistono diversi di questi carrier; a seguire si documenta quello che più verosimilmente potrà tornare utile almeno inizialmente.

Le figure a seguire rappresentano lo schema elettrico e il PCB di questa versione di carrier LoRa:

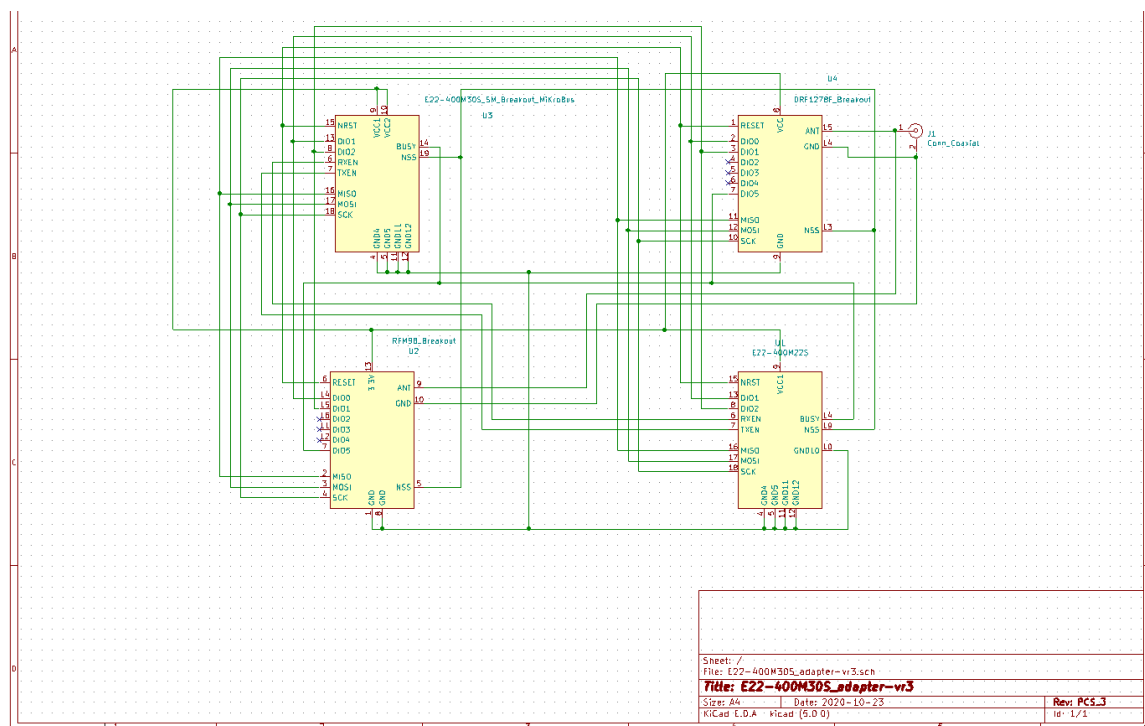


Figura 13 LoRa carrier : schema elettrico

Come si potrà notare la funzione di questo piccolo PCB è quella di consentire di adattare fisicamente i diversi moduli per essere montati sui PCB principali condividendo una piedinatura a 2.54 mm identica per i diversi tipi di moduli LoRa. Questo consentirà verosimilmente di intercambiare diversi moduli LoRa sugli stessi PCB principali per poter fare eventuali confronti.

Il PCB come si potrà notare presenta numerose impronte diverse tra loro: ovviamente una sola delle impronte andrà usata per volta; per montare chips LoRa diversi si costruiranno quindi diversi esemplari di carrier + chip LoRa.

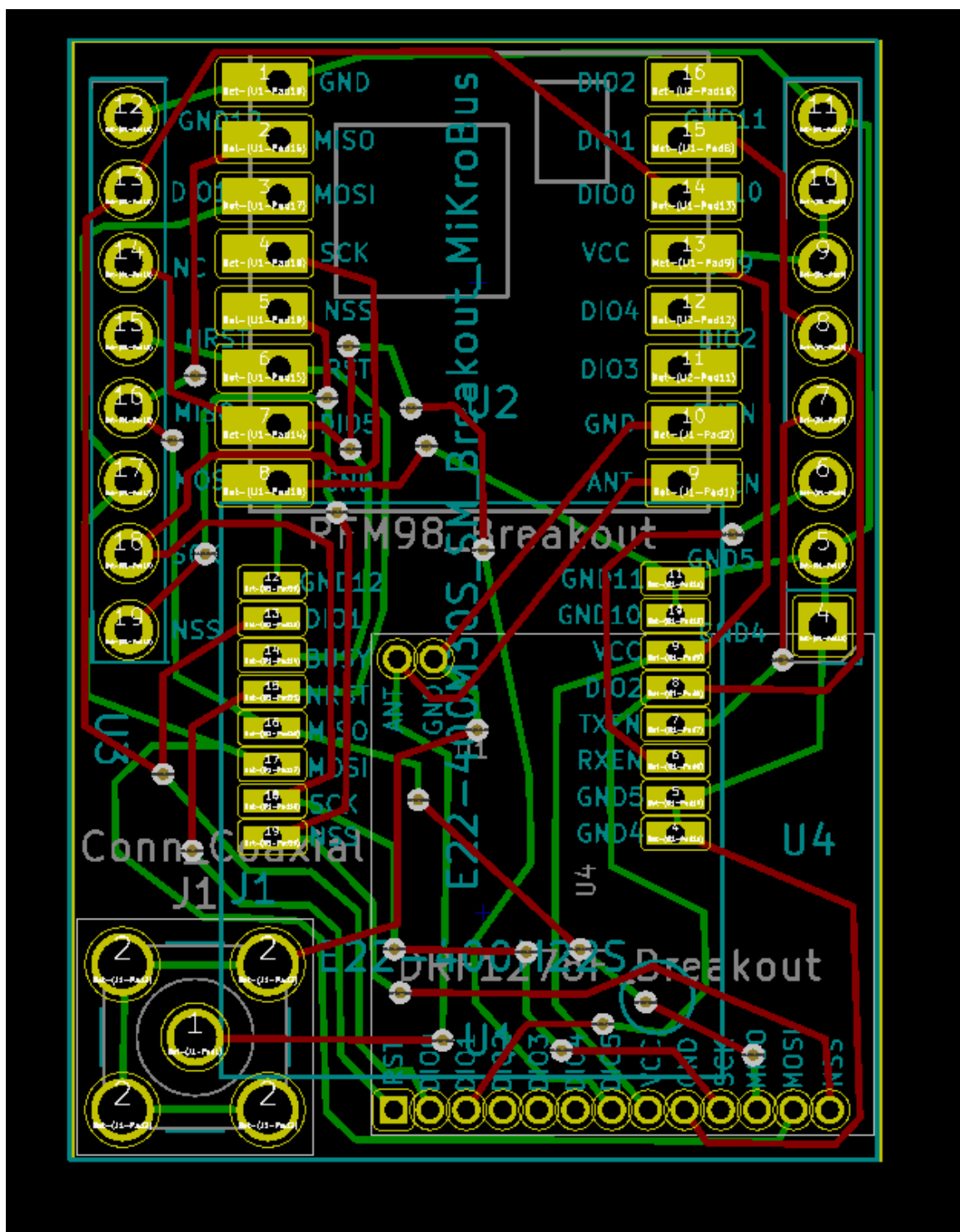


Figura 14 LoRa carrier : PCB layout

La figura seguente è una foto del lato superiore del PCB

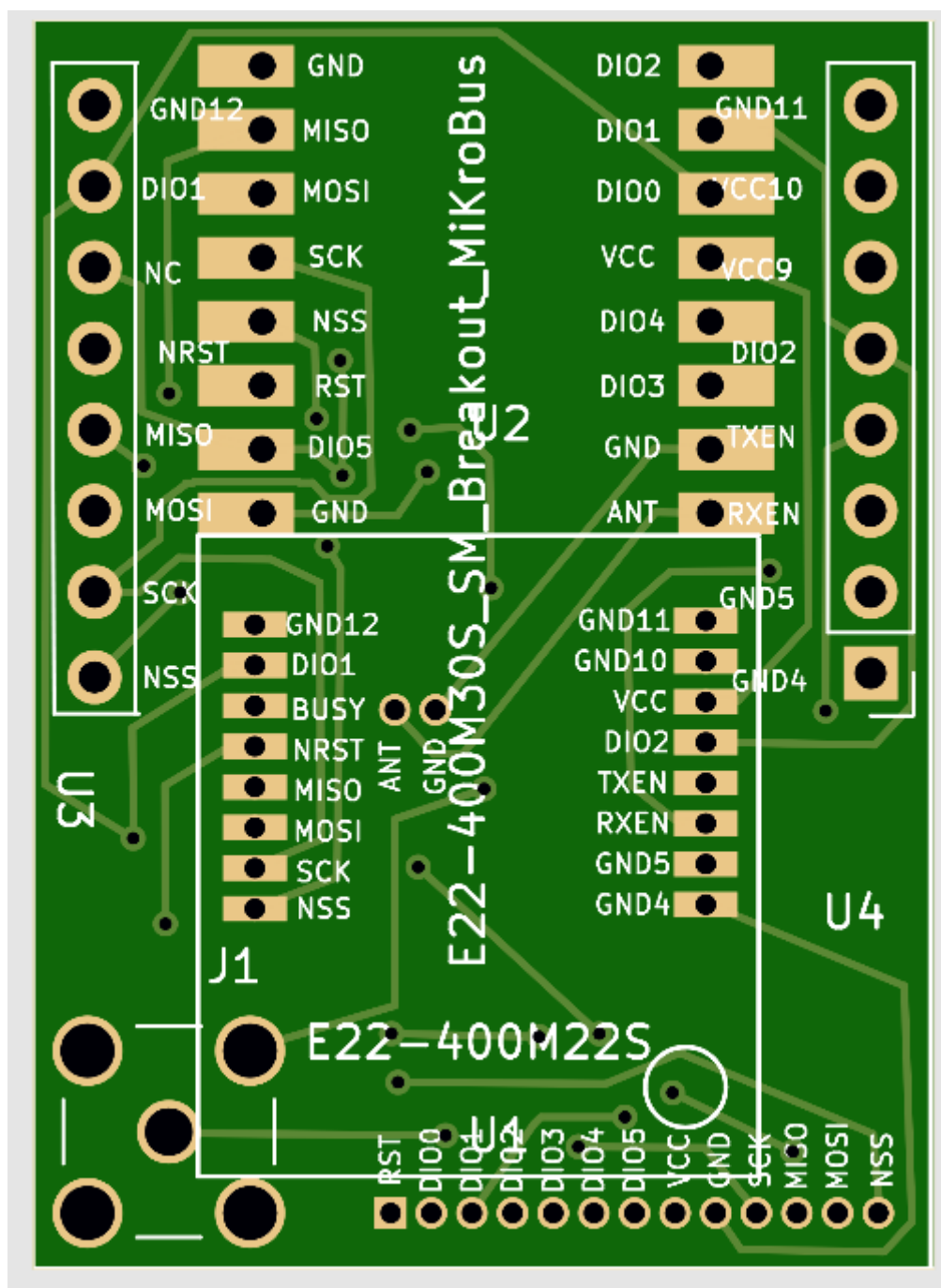


Figura 15 LoRa_Carrier : vista superiore del PCB

Il caso verosimilmente più frequente è quello in cui il modulo radio LoRa sia il tipo E22-400M30S che rappresenta il modulo radio attualmente a maggiori caratteristiche tecniche.

In questo specifico caso le dimensioni fisiche del modulo radio sono tali che già il modulo presenta una spaziatura dei pin di 2.54 mm anche se con un layout per montaggio SMD. In questo caso per il montaggio si sfrutterà il carrier indicato che consentirà di dotare il modulo di una piedinatura standard a 2.54 mm simile a tutti gli altri moduli usati nel progetto, consentendo di montare in maniera ottimale il modulo radio. Le figure seguenti illustrano la modalità di montaggio di questo modulo.



Figura 16 LoRa carrier : LoRa module montato su relativo PCB carrier



Figura 17 LoRa carrier : LoRa module montato su relativo PCB carrier

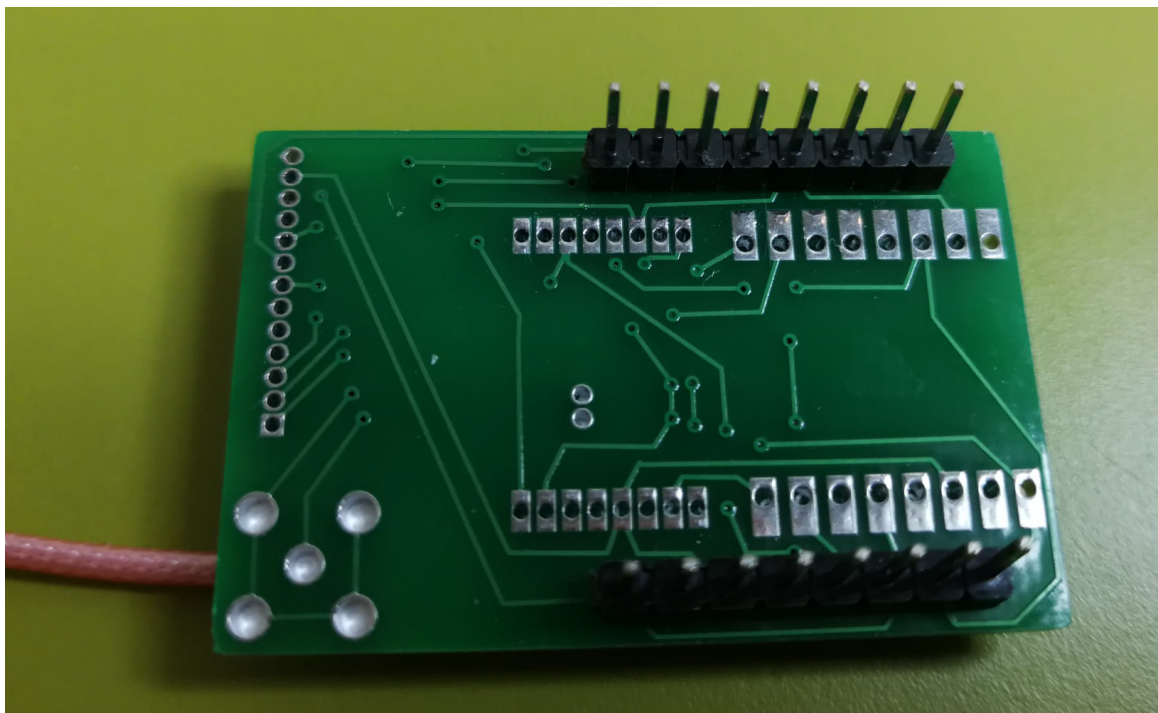


Figura 18 LoRa carrier : LoRa module montato su relativo PCB carrier lato inferiore

2.4 Note di montaggio relative ad entrambe le versioni di PCB carrier.

Per entrambe le versioni di PCB carrier principale esistono una serie di opzioni di montaggio inserite a solo scopo di flessibilità e per far fronte ad eventuali specifiche applicazioni.

La prima nota riguarda i moduli di display: entrambe le versioni presentano le impronte ed i collegamenti per poter montare sia un display a colori che un display monocromatico; i due display hanno interfacce diverse e possono essere utilizzati in alternativa o anche simultaneamente tramite una opportuna eventuale customizzazione del SW; nella versione standard del SW attualmente disponibile vengono supportati simultaneamente un modulo TFT a colori con interfaccia SPI ed un modulo OLED con interfaccia I2C, mostrando però lo stesso tipo di contenuti (questo consente di utilizzare alternativamente uno dei due moduli indifferentermente, senza nessun tipo di modifica SW).

Una ulteriore nota riguarda sempre il modulo display a colori che a seconda della versione acquistata può avere o meno i pin già saldati sul relativo PCB: per un montaggio ottimale si rende necessario usare dei pin con piegatura a 90°; qualora già sul modulo siano saldati dei pin diritti si rende necessario rimuoverli e sostituirli con dei pin piegati a 90°; per semplificare il lavoro di dissaldatura dei pin originali conviene con una tronchesina tagliare il supporto di plastica che lega i pin originali in modo che sia poi più agevole rimuoverli dissaldandoli e scuotendo gentilmente il modulo per farli cadere....

Per tutti i moduli da montare si consiglia di montare sul PCB carrier una fila di connettori a passo 2.54mm di tipo femmina in cui inserire poi il generico modulo dotato di pinnatura maschio sempre a 2.54mm di spaziatura.

Questo consente in futuro di poter agevolmente sostituire o intercambiare i moduli senza complicate operazioni di dissaldatura che inevitabilmente finirebbero per rovinare i circuiti.

Per entrambe le versioni di PCB i moduli sensore sono opzionali e attualmente non supportati nel SW base.

Per la versione iGate il modulo LAN è richiesto unicamente in caso si voglia utilizzare come modalità di connessione internet una connessione di tipo LAN anziché wifi; di default questa modalità non è abilitata.

Con la versione 4.x del SW viene rimosso il supporto della modalità LAN per il semplice motivo che non se ne è vista in realtà nessuna pratica utilità nella sperimentazione finora svolta e soprattutto per consentire l'introduzione di altre funzioni quali ad es. il supporto di uno storage basato su scheda compact flash nel prossimo futuro.

Il modulo GPS è mandatorio per la versione tracker mentre è opzionale per la versione iGate.

Il modulo RTC per la versione iGate è supportato da SW ma è comunque opzionale.

Il modulo GPS è stato previsto su entrambi i PCB carrier con una impronta che si può adattare a diversi tipi di moduli reperibili sui soliti portali di acquisto internet; per ulteriore dettagli sui moduli compatibili contattare lo scrivente per e-mail

Il PCB iGate presenta numerose ulteriori possibilità di customizzazione in quanto a moduli equipaggiabili che in questa fase non vengono ancora documentati in quanto non ancora sufficientemente testati a livello di supporto SW.

La versione iGate presenta due opzioni di alimentazione: o tramite connessione a 5V tramite cavo USB o tramite connessione a 12V e connettore standard 12V DC; la versione di default prevede di utilizzare l'alimentazione a 12V e quindi richiede il montaggio del modulo DC/DC PS1 ed il relativo connettore di alimentazione.

Esistono infine alcune predisposizioni effettuabili tramite opportuni jumpers con formato 2.54mm che si vanno di seguito a documentare:

Versione PCB mini-tracker: jumpers e loro significato

J3: 3V3_aux_sel default connettere pins 1-2

J5: LoRa_PS_sel utilizzare la posizione 1-2 per il modulo LoRa E22-400M30S che opera a 5V; posizione 2-3 per i moduli LoRa che lavorano a 3.3V

Versione PCB iGate: jumpers e loro significato

J3: 3V3_aux_sel default connettere pins 1-2

J5: LoRa_PS_sel utilizzare la posizione 1-2 per il modulo LoRa E22-400M30S che opera a 5V; posizione 2-3 per i moduli LoRa che lavorano a 3.3V

J7: per future espansioni nessun jumper da collegare

J8: per future espansioni nessun jumper da collegare

J11: per future espansioni nessun jumper da collegare

Come nota generale si consiglia di testare il circuito collegando gradualmente i vari moduli, iniziando dal processore ESP32 e continuando con il display, il GPS e infine il modulo LoRa.

Allo scopo di consentire eventualmente delle azioni di troubleshooting sono disponibili dei semplici “test segments” ovvero dei piccoli programmini che caricati opportunamente, tramite l’ambiente di sviluppo SW Arduino o, in versione 4.x, PlatformIO sul processore ESP32 possono consentire di testare i vari blocchi funzionali singolarmente o a piccoli gruppi.

La descrizione delle possibili azioni di test atte a fissare eventuali malfunzionamenti in fase di HW setup viene rimandata ad una successiva appendice di questo documento.

3 Installazione SW Iniziale

Il microcontrollore utilizzato nel progetto LoRa_Beacon è l'ESP32: si tratta di un dispositivo ad elevatissima scala di integrazione che contiene al suo interno una significativa mole di funzionalità che qui si evita di descrivere per brevità; su internet è possibile trovare moltissima documentazione in merito.

Il processore è acquistabile sotto forma di moduli con piedinatura standard a 2.54mm che contengono oltre al microcontrollore anche una serie di altri componenti tra cui una interfaccia USB tramite la quale è possibile interagire direttamente con il dispositivo sia per funzioni di sviluppo SW che per funzioni di monitoraggio diretto del funzionamento del dispositivo e di caricamento iniziale del SW.

La figura seguente riporta una foto del modulo processore e delle sue interfacce fisiche. Vale la pena osservare che in commercio esistono diverse varianti di questo modulo che differiscono per il numero di piedini e per il chip processore utilizzato; il nostro progetto richiede di usare la versione di modulo a 38 pins con antenna WiFi on board (stampata), come chiaramente mostrato nelle figure a seguire.

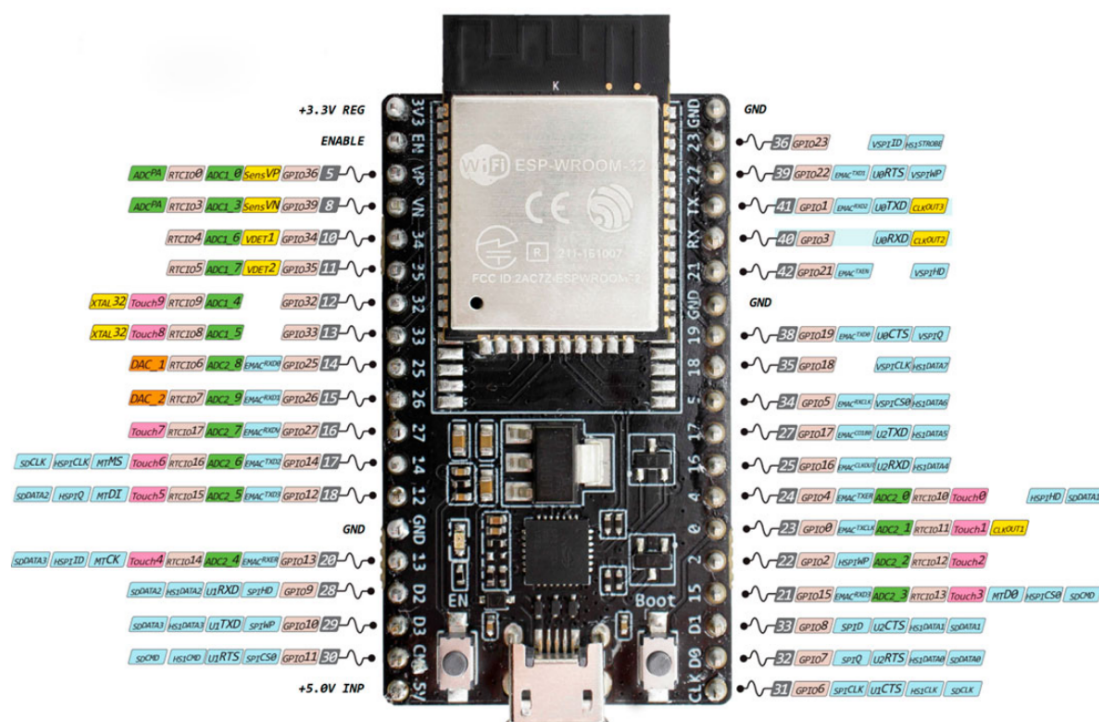


Figura 19 ESP32 modulo processore con relative interfacce

Il modulo utilizzato come processore contiene in sé tutto il necessario per poter funzionare autonomamente; in particolare contiene una memoria flash per contenere il programma SW, una memoria RAM da usare come memoria di lavoro, una interfaccia USB per il caricamento del SW e per il controllo del processore ed una interfaccia radio WiFi per poter interagire con il modulo via WiFi senza ulteriori componenti esterni richiesti.

Al momento dell'acquisto il processore arriva già dotato di un particolare SW che consente di caricare tramite l'interfaccia USB il SW utente necessario allo specifico progetto.

Questo SW consente sia di interagire con il processore direttamente tramite una semplice interfaccia grafica atta a caricare il SW richiesto, sia di interagire con il processore tramite un ambiente di sviluppo SW ad hoc.

Esistono vari ambienti di sviluppo possibili; quello utilizzato inizialmente nel corso del progetto LoRa_Beacon è basato sulla piattaforma SW Arduino IDE; con la versione SW 4.x l'ambiente di sviluppo di riferimento è stato migrato verso la piattaforma PlatformIO estremamente più performante e più agevole a conti fatti come setup e gestione; la familiarizzazione con questa nuova piattaforma comporta per gli utenti abituati ad Arduino una certa "learning curve" che però si dimostrerà abbastanza breve e consentirà di semplificare significativamente il lavoro per i successivi sviluppi.

Nel seguito viene illustrato il primo metodo di caricamento del SW, lasciando la descrizione della seconda modalità ad un altro documento: il grosso vantaggio di questa modalità è che consente di caricare il SW sul target HW senza dover necessariamente aver predisposto alcun ambiente di sviluppo SW (ad es. Arduino o PlatformIO) e senza necessità di alcuna esperienza di programmazione SW.

Ovviamente la seconda modalità, basata sull'uso di una piattaforma di sviluppo PlatformIO, è indispensabile qualora si vogliano fare delle modifiche al SW o impostare delle opzioni particolari non ancora gestibili direttamente tramite l'interfaccia GUI grafica del dispositivo.

3.1 Setup ambiente di caricamento SW e caricamento Immagine SW iniziale

L'operazione di caricamento del SW sul processore ESP32 richiede la predisposizione di un opportuno programma su un PC dotato di sistema operativo Windows o Linux.

E' richiesto di effettuare il collegamento tra il PC e il processore utilizzando un cavetto USB intestato con connettori appropriati.

All'atto del collegamento tramite il cavetto USB il processore verrà alimentato tramite il cavetto stesso e stimolerà il PC (Windows) ad installare automaticamente (in genere) i drivers che consentono di visualizzare il modulo come una interfaccia seriale per il computer.

Per scoprire l'identità della porta seriale con cui viene visto il modulo ESP32 è sufficiente esplorare la lista dei dispositivi del PC (tramite il pannello di controllo); su piattaforma Linux in genere il riconoscimento della nuova interfaccia è automatica.

In questa fase il modulo ESP32 può essere programmato anche non collegato al circuito su cui deve essere utilizzato.

Per scaricare da internet il tool di programmazione è possibile usare il seguente URL:
<https://www.espressif.com/en/support/download/other-tools>

La figura seguente mostra la pagina di download del tool e indica la versione da scaricare.

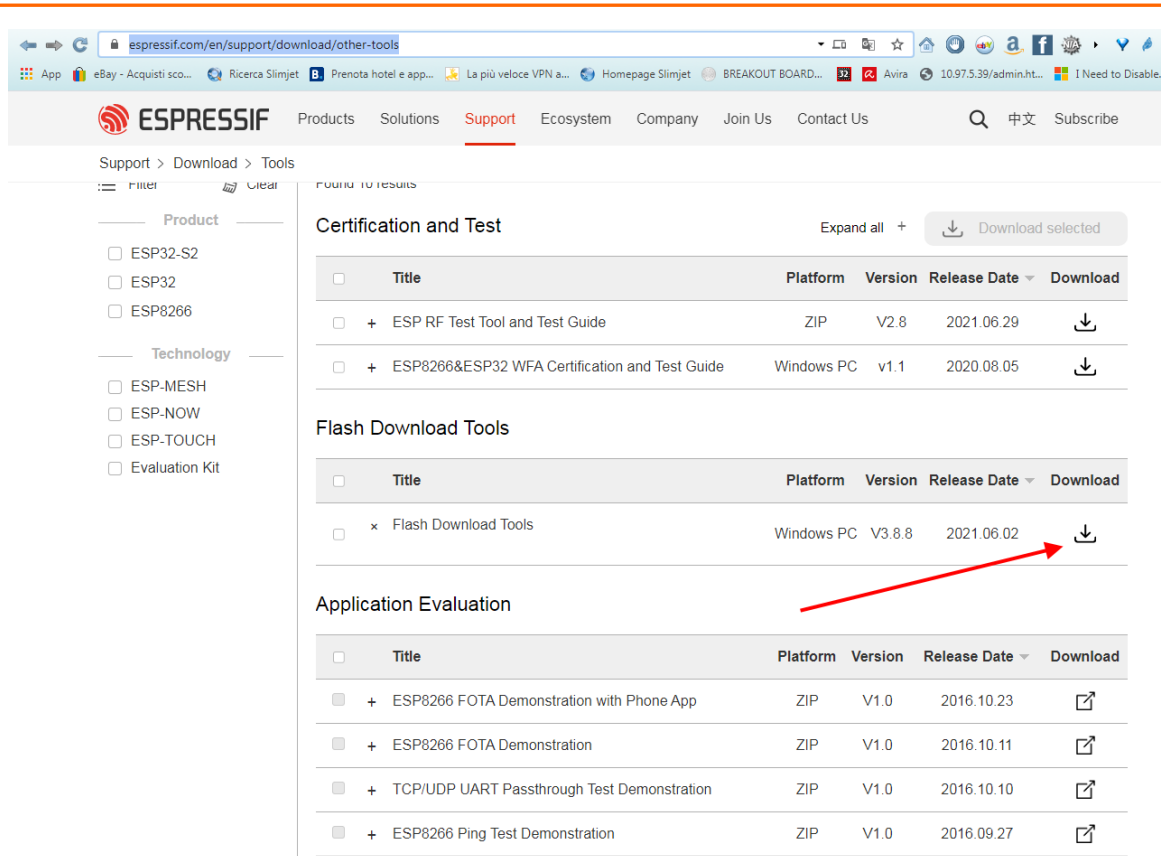


Figura 20 Pagina di download del tool di programmazione per il processore ESP32

Sul sito <http://iot-bits.com/esp32/esp32-flash-download-tool-tutorial/> è possibile anche trovare un piccolo tutorial sull'uso del tool; nel seguito descriviamo il solo caso di utilizzo del tool su ambiente Windows.

Una volta scaricato il tool “Flash Download Tool” sul PC espandere il relativo archivio e lanciare il file `flash_download_tool_3.8.8.exe`; quindi selezionare dal pannello che appare “DOWNLOAD TOOL MODE” i valori “**chip_Type = ESP32**” e “**WorkMode=develop**”; apparirà un pannello da settare come alla figura seguente:

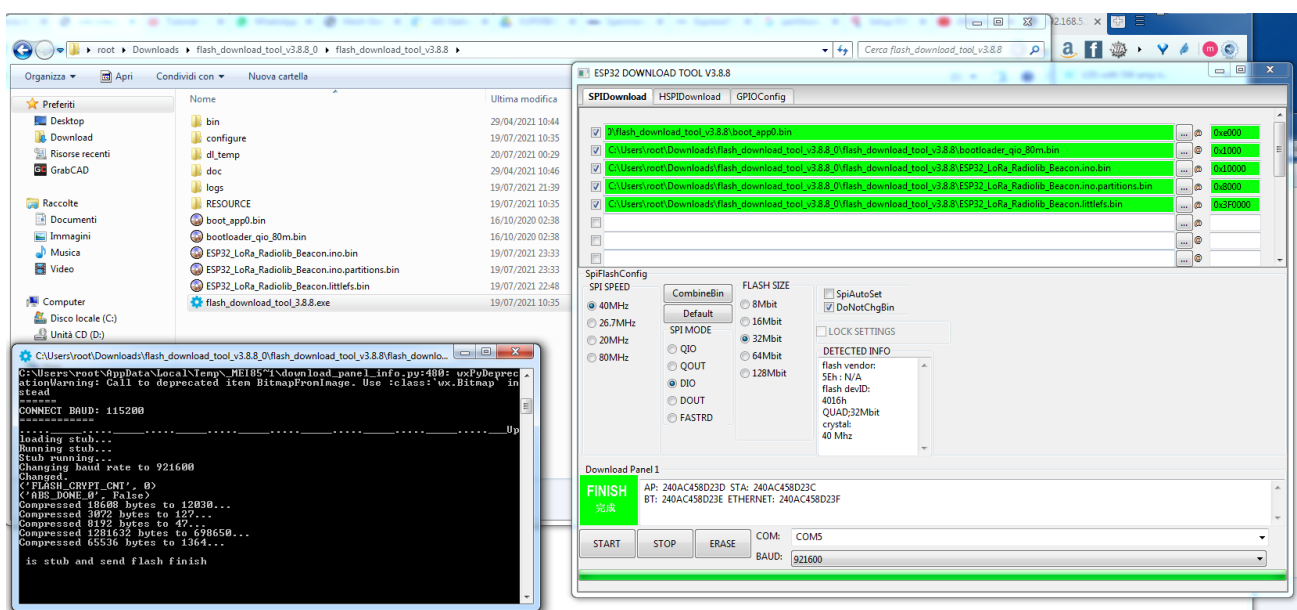


Figura 21 Utilizzo del tool di SW Download su processore ESP32

Le righe in verde sono i vari moduli SW che costituiscono l'immagine completa del SW: si tratta di 5 load modules che corrispondono ad altrettante sezioni della flash.

Il significato dei moduli SW che costituiscono l'immagine completa è il seguente:

- Boot_app di startup del processore
- Boot_loader per il caricamento del SW operativo
- Immagine SW applicativo
- Partition_table della flash
- Partizione LittleFS

La figura seguente è un ingrandimento ; per semplicità l'immagine viene fornita come un file .zip da espandere nel direttorio dove è stato espanso il tool di download del SW ; **per ogni sezione bisogna impostare manualmente nella parte destra della riga corrispondente l'indirizzo esadecimale dove quel pezzettino di SW deve essere caricato.** Questi indirizzi dipendono da come è stata configurata la flash del dispositivo in fase di sviluppo del SW e sono contenuti in un opportuno file readme.txt presente dell'archivio contenente i vari moduli. **Prestare molta attenzione al setup di questi valori, pena la non ripartenza corretta del processore.** In caso di errore è sufficiente ripetere l'operazione di flashing introducendo i valori corretti.

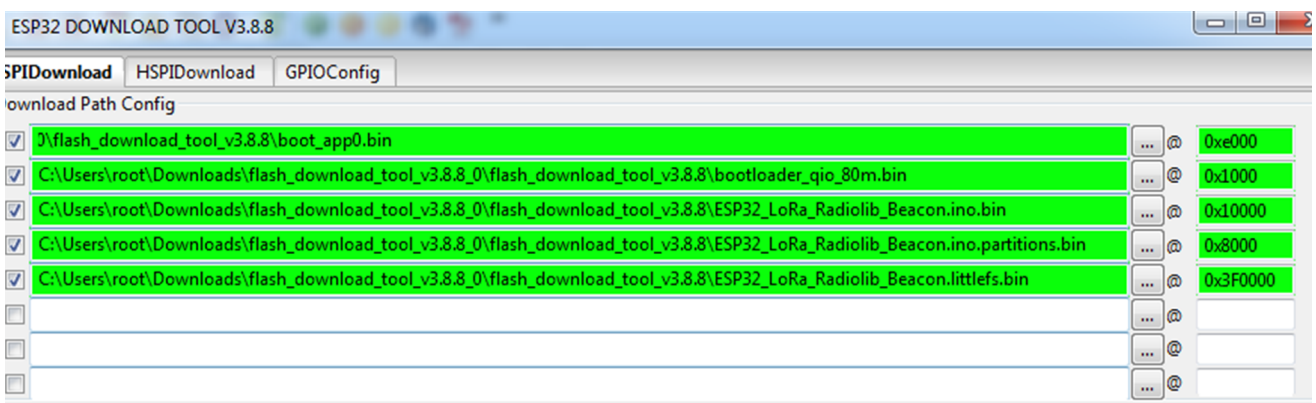


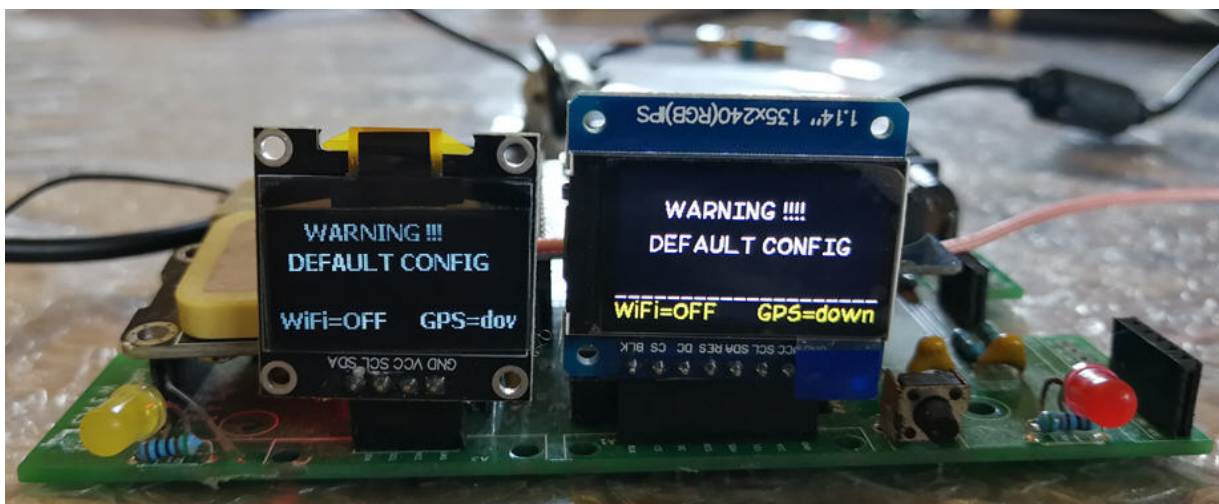
Figura 22 Impostazione load modules e relativi indirizzi in flash

Nella finestrella COM va inserita la porta seriale su cui il computer ha allocato l'interfaccia USB del modulo processore che si intende caricare; per gli altri parametri settarli come in figura. Settare il valore di BAUD a 921600 ; se non va, provare con 115200.

Verificare che in questa fase non sia aperta nessuna altra finestra relativa per es. al sistema di sviluppo Arduino o altra applicazione che utilizzi la seriale su cui è attestato il processore.

Premendo il tasto START il tool provvederà a caricare sul processore l'immagine SW indicata: nella finestra command che accompagna il tool è possibile seguire l'andamento del caricamento del SW; qualora non si noti la partenza della riga verde sul margine inferiore della finestra principale verificare i valori inseriti ed eventualmente provare a modificare il valore del BAUD impostato.

Una volta caricata l'immagine SW montare il modulo processore sul PCB principale e verificare che sul display si abbiano dei segni di vita :).



Con la versione SW 4.x a valle del primo caricamento del SW su un dispositivo completamente vergine viene caricata automaticamente una configurazione di default con una serie di parametri preimpostati per poter avere un dispositivo già in grado di operare come tracker anche se con dei parametri di personalizzazione del dispositivo di tipo default... ovviamente per un uso concreto sarà necessario modificare questa configurazione sostituendo i parametri di default con dei parametri ad hoc quali ad es. il callsign del dispositivo, la posizione di riferimento (latitudine/longitudine) ed i parametri di accesso alla rete esterna quale ad es. il WiFi per la connessione upstream verso internet (indispensabile per il modo di operation di tipo iGate) e le credenziali per l'accesso al sito <http://aprs.fi> per il riporto degli spot verso APRS-IS. Per la customizzazione fare riferimento alle sezioni seguenti di questo documento.

Qualora si vada invece a caricare il SW su un dispositivo già utilizzato in precedenza con lo stesso o una diversa versione di SW, verrà mantenuta la vecchia configurazione del dispositivo contenuta nella memoria FRAM o nella EEPROM del processore: questo implica che se la nuova versione del SW è compatibile con la precedente il dispositivo non avrà bisogno di alcun ritocco alla configurazione stessa, mentre se la nuova versione di SW non è compatibile con la precedente si potrà rendere necessaria una verifica puntuale ed un adattamento ovvero potrà essere effettuato un ripristino della configurazione ai parametri di default. La compatibilità tra versioni sarà evidenziata nelle release notes delle varie versioni di SW che si succederanno.

Con la versione 4.x vengono implementati dei controlli ad hoc per quanto riguarda la compatibilità e vengono emesse eventualmente delle indicazioni sul display per orientare il setup.

Sempre in SW vr. 4.x è stata semplificata la modalità di ripristino alla configurazione di default: ora **quando il dispositivo parte e prima che sul display compaia alcun nuovo messaggio ci sta una fase in cui i due led rosso e verde risultano accesi simultaneamente per qualche secondo... premendo il tastino presente sul dispositivo in questa fase si produce il ripristino delle condizioni di default** e si cancella completamente la onfigurazione vecchia del dispositivo.



Provare quindi a verificare con il PC che sia comparsa una nuova rete WiFi con identità del tipo ESP32-<mac address> : selezionare questa rete e avviare il collegamento ad essa (**il PC deve essere impostato per acquisire automaticamente l'indirizzo IP**). Verificando nelle connessioni di rete dovrà apparire una situazione simile alla figura seguente:

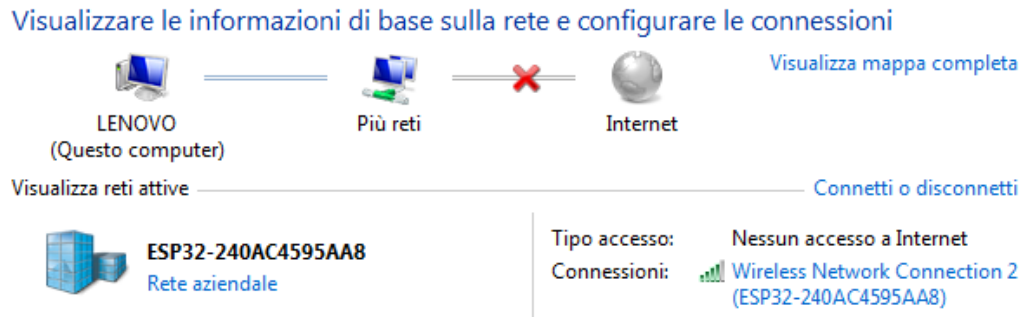
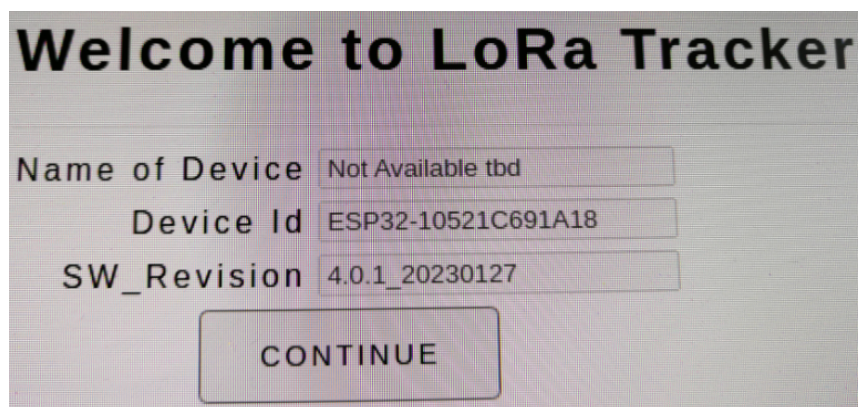


Figura 23 Visualizzazione rete WiFi di gestione del dispositivo

Accedere con un browser tipo chrome o firefox al seguente URL: <http://192.168.5.1> dovrà apparire una schermata come in figura sotto:



I Valori presenti nelle finestrelle ovviamente dipenderanno dal SW caricato e dal dispositivo specifico che si sta usando. Premendo il tasto “continue” apparirà la schermata principale della GUI, come in figura seguente:

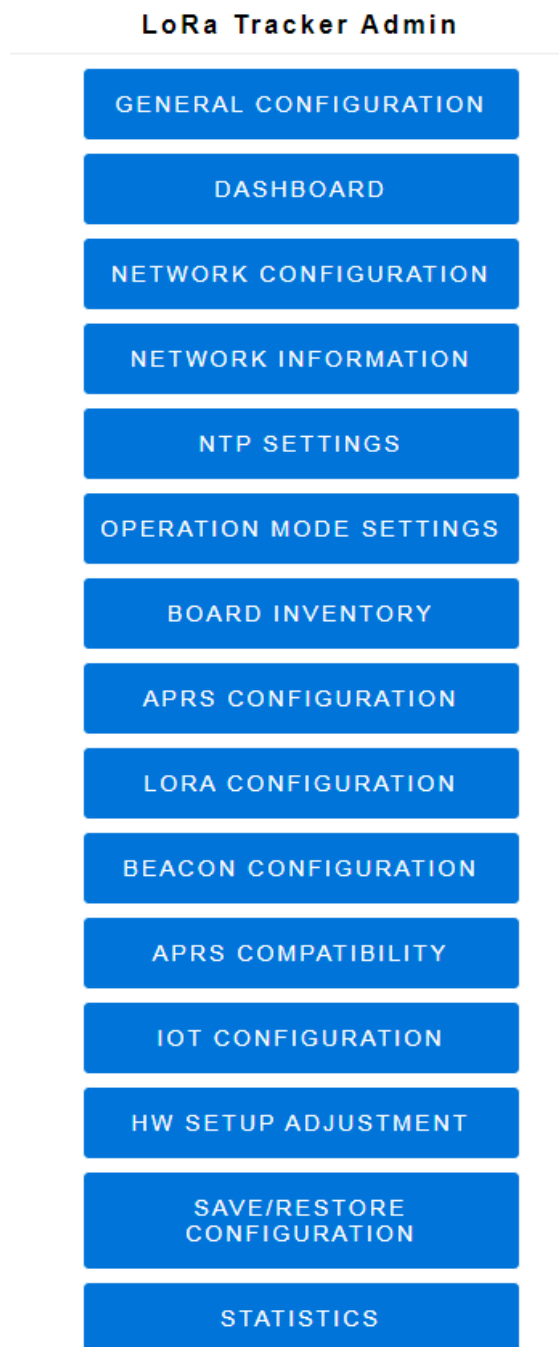


Figura 25 Pagina Principale interfaccia GUI

3.2 Setup iniziale del dispositivo LoRa_Beacon (qualsiasi versione)

L'operazione di caricamento del SW sul processore ESP32 come illustrata al paragrafo precedente si rende necessaria in linea di principio unicamente all'inizio dei tempi: la sua funzione è predisporre un ambiente SW in grado di essere agevolmente configurato tramite la sua interfaccia grafica senza dover ricorrere ad un collegamento diretto tra il PC utilizzato per la gestione ed il dispositivo fisico da gestire, e senza richiedere la predisposizione di un ambiente di sviluppo Arduino o PlatformIO necessariamente.

Nella sua configurazione di default il SW non è in grado di operare in maniera appropriata in quanto è **necessario specificare una serie di elementi strettamente dipendenti dallo specifico esemplare** di dispositivo che si sta gestendo, per cui è richiesta una fase di “configurazione del SW installato”.

Per illustrare la configurazione conviene innanzitutto illustrare grossolanamente come è impostata l’interfaccia di gestione e introdurre il concetto di “**modalità operative**” del dispositivo.

< Operation and Debug Functions

Debug Mode

gps_debug: ☐
LoRa_debug: ☒
APRS_debug: ☒
RTC_debug: ☐
ezTime_debug: ☐
pps_debug: ☐
PE_debug: ☐
WebConfig_debug: ☐
act_flag: ☒

Operation Mode

Admin_Mode: ☐
Beacon_Mode: ☐
iGate_Mode: ☒
TCP_KISS_Mode: ☐
Serial_KISS_Mode: ☐
Tracker_Mode: ☐
standalone: ☐
mqtt_ctrl_enable: ☒
syslog_enable: ☒
IoT_enable: ☐
no_gps: ☒

Maintenance

Load_Default_Conf: ☐
Reboot_Now: ☐

SAVE

Il SW è stato sviluppato per consentire di configurare il dispositivo fisico per poter comportarsi in diversi modi; questi diversi modi di funzionamento sono in generale tali che non è possibile passare da un modo ad un altro senza effettuare una ripartenza da fermo (reboot) del SW.

Esistono inoltre delle attività, quali ad es. quelle di sostituzione del SW e di salvataggio o restore della configurazione operativa, che richiedono parimenti che il processore venga posto in una opportuna modalità di funzionamento caratterizzata dal congelamento di alcune funzioni (ad es. la parte LoRa e la parte GPS) che definiamo “**Admin Mode**”.

Esiste un pannello nella interfaccia grafica , accessibile tramite la voce “**OPERATION MODE SETTINGS**” che consente di gestire queste diverse modalità operative e che quindi rappresenta la **prima sezione da configurare** dopo il caricamento iniziale del SW.

La figura 26 illustra il contenuto della pagina di cui sopra.

Una prima sezione della schermata consente di settare una serie di modalità di debug utilizzabili per tracciare e risolvere eventuali anomalie di funzionamento o di settare specifiche condizioni di equipaggiamento del dispositivo. Tale sezione inizialmente può essere lasciata nella sua condizione di default.

Figura 26 Operation Mode Settings

La seconda sezione della schermata consente di impostare la modalità operativa per il dispositivo; sono possibili varie modalità operative che nel seguito verranno individualmente illustrate.

La terza sezione consente di effettuare il reboot manuale del dispositivo senza bisogno di togliere e rimettere l’alimentazione elettrica al dispositivo, ovvero anche il restore della configurazione di default per il dispositivo.

Particolarmente importante è la modalità “Admin_Mode” come sopra descritto. Tale modalità andrà impostata ogni qualvolta sia richiesta a livello di interfaccia grafica.

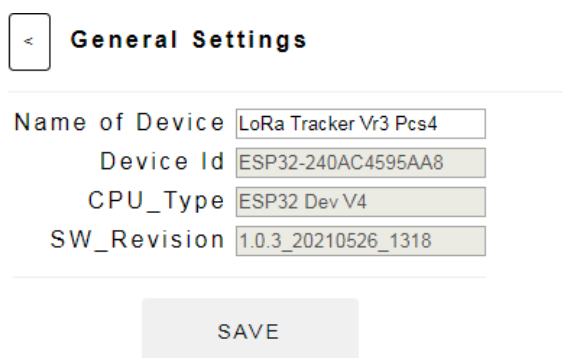
La modalità “iGate_Mode” predispone il dispositivo per operare in modalità iGate APRS LoRa e richiede la configurazione di una serie di parametri nella schermata “APRS CONFIGURATION”.

Qualora questa voce non venga selezionata, viene automaticamente selezionata la modalità “Tracker APRS”, anch’essa personalizzabile tramite la stessa schermata di APRS Configuration.

Le modalità “BT_KISS_Mode”, “Serial_KISS_Mode” sono per il momento non documentate ma consentono di operare il dispositivo come un classico TNC KISS per l’interfacciamento verso SW tipo APRX o direwolf.

La voce “syslog Enable” consente di abilitare la funzione di trasferimento automatico dei messaggi di log verso un server esterno di tipo syslog. La funzione verrà documentata in seguito.

La voce “mqtt_ctrl_enable” abilita le funzioni di controllo remoto del dispositivo tramite protocollo mqtt. La funzione verrà documentata in seguito.



< **General Settings**

Name of Device

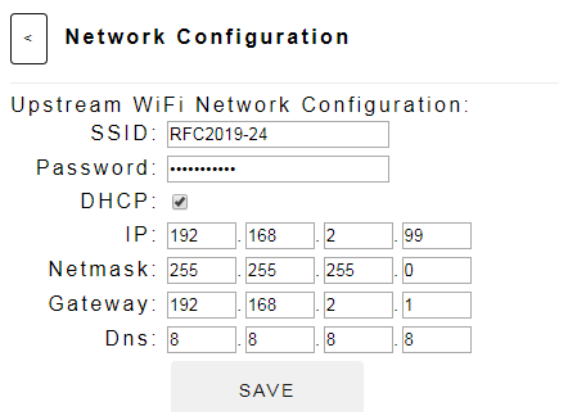
Device Id

CPU_Type

SW_Revision

SAVE

Figura 27 General settings



< **Network Configuration**

Upstream WiFi Network Configuration:

SSID:

Password:

DHCP: ☒

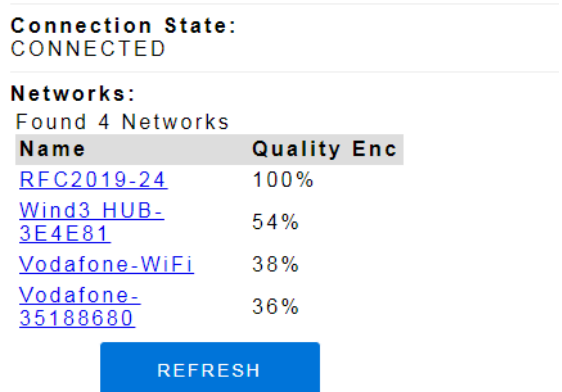
IP:

Netmask:

Gateway:

Dns:

SAVE



Connection State:
CONNECTED

Networks:
Found 4 Networks

Name	Quality	Enc
RFC2019-24	100%	
Wind3 HUB-3E4E81	54%	
Vodafone-WiFi	38%	
Vodafone-35188680	36%	

REFRESH

La voce “IoT_enable” serve ad abilitare una serie di funzionalità di tipo IoT collegate all’utilizzo della sensoristica presente sul dispositivo ed è allo stato ancora da documentare.

La voce “no_gps” serve per dichiarare che il dispositivo è privo di GPS o non deve assumere la presenza di GPS on board: serve per utilizzare per es. un dispositivo come iGate con posizione geo predefinita da GUI, in modo da non richiedere la presenza di un dispositivo GPS operativo (overo con vista del cielo).

Una ulteriore sezione da configurare inizialmente è rappresentata dalla voce “GENERAL CONFIGURATION” della schermata principale della GUI.

Questa sezione riporta una serie di dati relativi alla versione SW, al tipo di processore e alla identità del dispositivo, il tipo di CPU di cui è dotato, e la revisione SW installata e consente di settare manualmente alla prima riga un nome di fantasia per il dispositivo.

Questo nome verrà unicamente utilizzato per individuare il dispositivo ed è quindi liberamente settabile dall’utilizzatore.

Una ulteriore sezione da settare inizialmente è la “**NETWORK CONFIGURATION**”: questa sezione consente di settare la modalità di connessione a livello WiFi che si intende utilizzare per il dispositivo.

A tale proposito bisogna illustrare brevemente quali funzionalità WiFi sono implementate sul dispositivo ed il loro utilizzo.

Il dispositivo implementa due tipi di funzionalità WiFi , utilizzabili per esigenze diverse:

- Modalità di AP Access Point WiFi
- Modalità di tipo “Station WiFi”

La prima modalità consente di accedere al dispositivo da un qualsiasi client WiFi come se fosse un normale punto di accesso WiFi: effettuando con il PC per es. una scansione delle reti WiFi disponibili si noterà la presenza di una rete WiFi con SSID del tipo “ESP32-<mac_address>” : collegandosi a tale rete WiFi, avendo impostato il PC per acquisizione dell’indirizzo IP tramite DHCP, il dispositivo fornirà al PC una configurazione IP tale che sarà possibile collegarsi al dispositivo utilizzando l’indirizzo standard 192.168.5.1.

Il sistema è parimenti utilizzabile usando un qualsiasi dispositivo mobile dotato di Browser tipo Chrome o Firefox. Le schermate della GUI sono ottimizzate per l’uso con dispositivi mobili.

In questo modo si potrà configurare il dispositivo anche senza la presenza di un access point esterno o di una qualsiasi copertura WiFi di accesso ad internet.

Nella modalità “Station WiFi” invece il dispositivo effettuerà autonomamente una scansione delle reti WiFi presenti in zona e presenterà una lista delle SSID ascoltate consentendo di impostare il collegamento del dispositivo ad una di queste reti; la funzionalità riguarda solo le reti a 2.4 Ghz.

La schermata di cui si accennava sopra serve proprio a selezionare la rete WiFi a cui collegare il dispositivo in modalità Station, e in pratica consente di collegare il dispositivo alla propria rete locale e ad internet.

E’ possibile sia specificare per questa connessione l’acquisizione dell’indirizzo IP da parte del dispositivo tramite protocollo DHCP, sia specificare un indirizzo statico manualmente.

Se si accede a questa schermata la prima volta la sezione “Network” potrà apparire vuota...

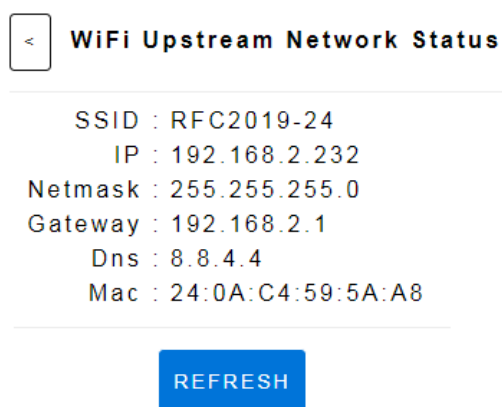


Figura 29 Network Informations

aspettando alcuni secondi verrà automaticamente popolata con la lista delle reti WiFi a 2.4Ghz disponibili: selezionando una delle reti indicate si potrà impostare nella prima sezione della schermata l’eventuale password.

Selezionando la casella DHCP il dispositivo acquisirà automaticamente la configurazione IP dalla rete selezionata.

In assenza della selezione della casella DHCP si potrà specificare manualmente un indirizzamento per il dispositivo. Finito il setup delle varie caselle sarà necessario salvare la configurazione con il tasto “SAVE”.

Il tasto “REFRESH” serve a riacquisire la lista delle

reti WiFi disponibili.

Per conoscere lo stato della connessione di rete WiFi si può consultare la sezione “NETWORK INFORMATION” della pagina principale della GUI: tale pagina fornirà i valori attualmente in uso da parte del dispositivo per la sua connessione alla rete locale e ad internet.

Va osservato che le due funzionalità indicate non possono essere utilizzate simultaneamente a

causa delle limitazioni SW presenti nel SW di base del processore ESP32; quindi per accedere al dispositivo via WiFi, si dovrà utilizzare o la modalità AP (e quindi

Figura 28 Network Configuration

l'indirizzo fisso 192.168.5.1) se il dispositivo è impostato per NON utilizzare l'upstream WiFi verso internet tramite un AP esterno, ovvero, se la connessione upstream è attiva, tramite l'indirizzo che il dispositivo acquisirà dalla rete WiFi a cui si sarà collegato.

Per attivare o disattivare l'upstream WiFi, una volta configurata la funzione dalla schermata prima indicata, è possibile utilizzare un meccanismo abbastanza semplice che sfrutta il tastino presente sul dispositivo e quindi senza richiedere alcun dispositivo di supporto esterno: il meccanismo prevede di tenere premuto il tastino per un certo tempo... appariranno sul display dei messaggi che indicheranno la funzione da selezionare... rilasciando il tastino mentre è indicata una certa funzione verrà predisposta l'attivazione della relativa operatività... per rendere operativa la nuova modalità sarà necessario riavviare il dispositivo.

La funzione di riavvio del dispositivo è realizzabile sia spegnendo e riaccendendo il dispositivo, sia tenendo il tastino premuto fino a far apparire la scritta "Rebot Now".

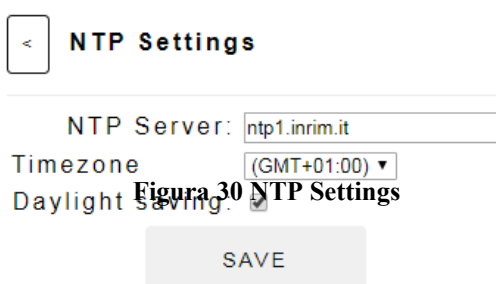


Figura 30 NTP Settings

Per completare la configurazione dei parametri di rete del dispositivo è necessario specificare un "Network Time Server" da eventualmente utilizzare per la sincronizzazione dell'orario locale del dispositivo in modo da consentire di marcare temporalmente una serie di dati acquisiti durante il funzionamento, quali ad es. gli spots LoRa o i messaggi di evento/errore prodotti dal dispositivo.

Allo scopo si utilizzerà la sezione "NTP SETTINGS" della pagina principale riportata a fianco.

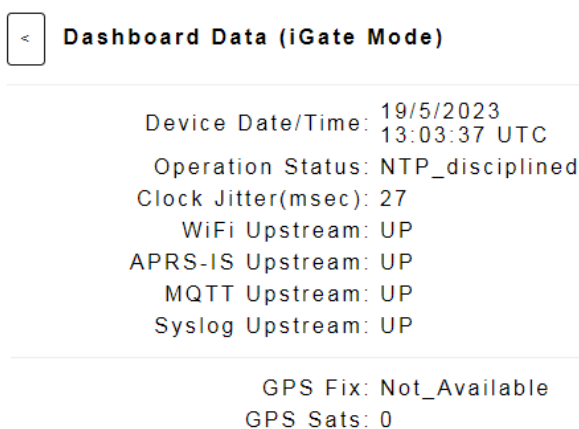


Figura 31 Dashboard: modalità di sincronizzazione

Il valore del server NTP dovrà essere impostato con il nome DNS del server che si intende utilizzare mentre la casella Timezone va selezionata in base al fuso orario geografico.

La casella Daylight Setting serve ad indicare se è in uso l'orario legale.

Il settaggio di un server NTP è importante in quanto consente di avere correttamente impostato l'orario del dispositivo in particolare in assenza di un modulo GPS come tipicamente avverrà nell'utilizzo come iGate.

Infatti il dispositivo è progettato per asservire il suo orologio locale ad un modulo GPS, se presente, o in sua assenza ad un Server NTP raggiungibile via internet.

Per conoscere lo stato di connessione del dispositivo e le sue principali caratteristiche operative è possibile utilizzare la schermata "DASHBOARD" presente sulla pagina principale della GUI.

Questa schermata, che verrà descritta nel prosieguo in dettaglio, presenta alle prime righe proprio lo stato di funzionamento del dispositivo dal punto di vista del settaggio del tempo locale.

Una sezione particolarmente importante da consultare è la pagina "**BOARD INVENTORY**" accessibile dalla pagina principale della GUI.

Come conseguenza della sua modularità il dispositivo potrà avere equipaggiati una serie di moduli opzionali: la maggior parte di questi moduli verrà riconosciuta automaticamente alla

<

Inventory Configuration

Inventory Configuration:

has_PCF8574: ☒

has_DS3231: ☒

has_DS3231_eeprom: ☒

has_SI5351: ☐

has_SSD1306: ☐

has_ST77XX: ☒

has_BME280: ☐

has_SI7021: ☐

has_FM24W256: ☒

LoRaDevice:

LoraDeviceType:

has_GPS: ☐

has_LittleFS: ☒

has_SD_Card:

SD_Card_Size:

REFRESH

Figura 32 Inventory configuration

debug settabili in tale pagina.

Il dispositivo essendo previsto per fare della sperimentazione potrà richiedere eventualmente di avere accesso ad una serie di funzioni di test e debug non necessariamente richieste durante la normale operatività del dispositivo.

La pagina indicata consente di settare una serie di “flags” che attiveranno selettivamente le funzioni di debug presenti per “sezioni” del progetto.

Questo consente di avere accesso tramite le vie di monitoraggio che verranno descritte in seguito, ad una notevole mole di messaggi di debug atti ad individuare eventuali problemi o caratteristiche operative in tempo reale, senza dover necessariamente accedere ad un ambiente di debug specifico aggiuntivo.

Un cenno particolare al flag “act_flag”, ovvero activity flag: è un flag da lasciare sempre attivato.

<

LoRa APRS Configuration

LoRa APRS Main Configuration:

LoraFreq:

LoraBw:

LoraSf:

LoraCodingRate:

LoraPreambleLen:

LoraSync:

LoraPower:

LoraFreqCorr(ppm * 10):

LoRa_FreqJitter(ppm)now: 0

partenza del dispositivo nella sua fase di configurazione dinamica: il risultato di questa fase viene sinteticamente riportata nella schermata di cui sopra.

In particolare consultando questa pagina si potrà scoprire se tutti i moduli teoricamente inseriti sul PCB principale siano stati riconosciuti.

Una feature importantissima è quella indicata come “has_FM24W256”: trattasi della memoria FRAM destinata a contenere i dati di configurazione del dispositivo; l’assenza del riconoscimento di tale dispositivo impedisce il salvataggio dei dati di configurazione in FRAM (comunque in questo caso verrà utilizzata la flash on board del processore ESP32 per salvare i dati di configurazione).

A conclusione di questa sezione vale la pena ritornare un attimo sulla pagina “OPERATION MODE SETTINGS” per accennare alle funzioni di

3.3 Setup sottosistema LoRa (qualsiasi versione)

Selezionando la sezione “**LORA CONFIGURATION**” dalla pagina principale della GUI si accede alla configurazione del sottosistema LoRa.

Come noto sul dispositivo possono essere equipaggiati diversi tipi di moduli radio LoRa: per conoscere il modulo LoRa attualmente configurato e riconosciuto è possibile consultare la pagina di “Inventory

configuration”: le righe presenti nella pagina di configurazione LoRa dipenderanno dal tipo di modulo equipaggiato/configurato.

Nella schermata sono presenti solo un sottinsieme di parametri configurabili su un generico modulo LoRa; i parametri non presenti sono volutamente non riportati in quanto il loro settaggio è tipicamente legato a particolari discorsi di natura HW/SW che richiedono modifiche a livello di codice sorgente del SW.

Molti dei parametri sono intuitivi e il loro settaggio dipende proprio dal tipo di sperimentazione che si intende fare.

In linea di principio affinché due dispositivi possano comunicare tra loro i parametri LoRa dei due dispositivi devono coincidere.

Gli unici parametri che possono differire sono il valore di potenza in trasmissione e il valore di correzione di frequenza.

Quest’ultimo parametro potrà essere tipicamente lasciato a zero per i dispositivi che montano moduli LoRa ad alta precisione di frequenza (ad es. il modello E22_400M30S o altri moduli di seconda generazione dotati di TCXO) mentre andrà sperimentalmente impostato per i moduli LoRa di prima generazione che non hanno una buona precisione di frequenza o per quelli di seconda generazione sprovvisti di TCXO. **Allo scopo si potrà utilizzare come suggerimento il valore di FreqJitter misurato in tempo reale per i pacchetti ricevuti e riportato nell’ultima riga della schermata.**

Come verifica del corretto allineamento di frequenza si potrà osservare su un RX SDR il segnale trasmesso per giudicare il valore di offset in parti-per-milione da impostare.

The screenshot shows the 'Aprs Configuration' screen with a back button in the top left. The settings are organized into sections:

- Location Coordinates**
 - Latitude: 4038.67N
 - Longitude: 01424.55E
- APRS/IS iGate**
 - APRS_Host: rotate.aprs2.net
 - APRS_Port: 14580
 - APRS_Login: I8FUC-10
 - APRS_Pass:
 - APRS_Filter: max 2 km
 - iGate_Beacon: I8FUC-10>APZMDM,WIDE1
 - iGate_Beacon_Int: 5 min.
- APRS Tracker**
 - Tracker_Beacon: I8FUC-8>APZMDM,WIDE1
 - Tracker_Beacon_Int: 30 secs.
- APRS Logger**
 - APRS Logger Host: 192.168.2.150
 - APRS Logger Port: 44445

A 'SAVE' button is located at the bottom center of the configuration area.

Figura 33 LoRa Configuration

3.4 Setup sottosistema APRS (qualsiasi versione)

Selezionando la sezione “APRS CONFIGURATION” dalla pagina principale della GUI si accede alla pagina che consente di configurare i parametri per il servizio APRS.

Come ormai è chiaro il dispositivo può operare in due modalità APRS diverse:

- come iGate/repeater
- come tracker

I parametri da configurare per le due modalità dono diversi anche se simili. la figura a seguire copre entrambe le modalità.

3.4.1 Setup sottosistema APRS per modalità iGate e Tracker

Affinchè la modalità iGate possa svolgersi è necessario innanzitutto che il dispositivo sia in grado di collegarsi ad internet in modo da potersi collegare ad un opportuno server della rete ARPS-IS che dovrà risultare raggiungibile a livello IP.

E' poi necessario che il gateway internet in uso sia configurato in modo da consentire il traffico uscente sulla porta IP utilizzata dal server APRS-IS e che dipenderà dal sever APRS-IS scelto.

Infine è necessario che sia disponibile un account sulla rete APRS-IS con cui consentire l'autenticazione del dispositivo da parte del server APRS-IS scelto; in particolare serviranno una Login identity ed una Password.

Tutti i parametri ora indicati vanno popolati nelle rispettive caselle presenti nella sezione "APRS/IS iGate" della schermata indicata.

Per quanto riguarda la posizione che l'iGate riporterà verso la rete come propria posizione è possibile dedurla automaticamente, se è presente un modulo GPS in grado di acquisire un fix 3D; in caso contrario è possibile inserire manualmente la posizione da riportare .

Per la determinazione del valore di coordinate da riportare bisognerà utilizzare il formato nativo APRS; ovvero:

- latitudine: 2 digit per i gradi + 4 digit con punto dopo i primi due digit per i primi e secondi come previsto da APRS seguiti dalle lettere N/S per nord o sud
- longitudine: 3 digit per i gradi + 4 digit con punto dopo i primi due digit per i primi e secondi come previsto da APRS seguiti dalle lettere E/W per nord o sud

Sempre per la funzionalità iGate è necessario inserire un parametro che consenta di filtrare le posizioni da replicare sulla rete LoRa e provenienti da APRS-IS: l'unica modalità supportata dalla GUI è la modalità "distanza massima dalla posizione del iGate" ovvero la distanza in Km entro cui replicare sul lato LoRa gli spots APRS ricevuti da APRS/IS.

E' poi possibile inserire i parametri richiesti per creare la stringa da utilizzare come beacon dell'iGate/tracker verso la rete LoRa e dall'iGate verso la rete APRS/IS.

Tale stringa verrà automaticamente composta in base ai parametri inseriti in questo campo in modo da produrre come risultato una stringa simile all'esempio di seguito illustrato:

parametri inseriti nel campo iGate_Beacon o Tracker_Beacon:

11XYZ-10, WIDE1-1, LoRa, &, L

string beacon risultante(campo location in chiaro):

11XYZ-10>APLS01,WIDE1-1:!GPS_LATGPS_LON&LoRa -32B125S12C8P10

string beacon risultante(campo location compresso):

11XYZ-10>APLS01,WIDE1-1:!L9w?#R-GG&!!GLoRa -32B125S12C8P10

Le stringhe "GPS_LAT" e "GPS_LON" indicano i parametri latitudine e longitudine raccolti dal modulo GPS locale al momento di generare uno spot se il GPS è equipaggiato, ovvero i valori immessi nei campi latitudine/longitudine della schermata in oggetto in caso di GPS assente .

Il formato della stringa che rappresenta il contenuto informativo del beacon, come definita nelle specifiche APRS, si compone come si può vedere di diverse parti alcune delle quali possono essere settate in maniera ad hoc dal gestore del dispositivo; nel caso specifico gli unici campi che si lasciano settabili sono quelli indicati in rosso allo scopo di poter agevolmente identificare la sorgente di pacchetti come un dispositivo in tecnica HW/SW Sarimesh (*APLS01*) ; il campo che segue il simbolo "!" rappresenta la location del dispositivo: se tale campo inizia con un carattere numerico si

Figura 34 APRS Configuration

assume che la location sia in formato “in chiaro”, altrimenti si assume che i 13 caratteri che seguono rappresentino in maniera compressa la location stessa; l’algoritmo di compressione è descritto sempre nella specifica del protocollo APRS.

Con la versione SW 4.x il software supporta entrambe le modalità di indicazione della location in ricezione, mentre in trasmissione è possibile indicare, tramite una successiva schermata, quale modalità utilizzare; l’utilizzo della modalità compressa consente di ridurre la lunghezza del campo location a vantaggio di una maggiore velocità di trasmissione del singolo pacchetto.

I caratteri L ed & presenti nell’esempio di cui sopra possono essere sostituiti con altri caratteri allo scopo di modificare il simbolo con cui lo spot APRS apparirà sui portali relativi; per una descrizione in merito è possibile consultare il seguente URL: <https://www.iz3mez.it/aprs-server/simboli-aprs/> (grazie a Giovanni IZ0CZW per la segnalazione). Si suggerisce di non modificare il carattere L in quanto per uso consolidato indica un dispositivo in tecnologia LoRa.

Il campo che segue l’ultimo campo in colore rosso negli esempi sopra indicati viene automaticamente generato ed inserito dal SW e riporta in maniera sintetica una serie di indicatori delle condizioni di lavoro del dispositivo; in particolare il significato è il seguente:

- working conditions (esempio):

-32B125S12C8P10

- significato dei campi se presenti:

- indica che il beacon è stato generato in una area blacklisted (solo per test)
- 32** Sequence Number modulo 100 per il pacchetto (viene incrementato ad ogni pacchetto beacon generato)
- B125** Bandwidth utilizzata per la trasmissione in Khz (senza decimali)
- S12** Spreading Factor utilizzato per la trasmissione
- C8** Coding Rate utilizzata per la trasmissione
- P10** Preamble Length usata per la trasmissione in numero di simboli

Nelle stringhe beacon generate possono essere anche presenti altri due campi a seguire caratterizzati dall’essere delle parentesi contenenti uno o più sottocampi: la funzione di questi campi , che possono essere inseriti o meno a seconda dei settaggi di una schermata successiva che si andrà a presentare, e quella di indicare il primo e l’ultimo nodo attraversato dal pacchetto APRS ricevuto da un certo dispositivo; questo consente agevolmente di avere una indicazione , in particolare per il secondo caso) delle condizioni di ricezione del pacchetto da parte del nodo indicato nel primo sottocampo della parentesi: il significato è deducibile dal seguente esempio:

- campo dispositivo attraversato:

(IQ8SO-10 -120 -16 186)

- decodifica dei sottocampi:

- IQ8SO-10** callsign del nodo attraversato
- 120** signal strength dello spot ricevuto dal nodo attraversato in db senza decimali
- 16** SNR dello spot ricevuto dal nodo attraversato in db senza decimali
- 186** frequency shift misurato dal ricevitore del nodo attraversato in Hz

Tutti i campi che seguono la parte standard del pacchetto di beacon sono chiaramente NON conformi allo standard APRS come documentato nella specifica APRS, per contenere al minimo la lunghezza di pacchetto da trasmettere; tutti i campi opzionali possono essere omessi in parte o in toto allo scopo di avere dei pacchetti emessi conformi allo standard APRS. In particolare per le condizioni di lavoro si assume di trasmettere l’intero campo solo ogni quinto pacchetto di una

sequenza di cento pacchetti. Per tutti gli altri pacchetti di una sequenza di cento verrà trasmesso esclusivamente il Sequence Number. La funzione del sequence number è quella di poter evidenziare sia situazioni di tipo perdita di pacchetto sia situazioni di alterazione della sequenza per cause diversificate; risulta ovviamente utile esclusivamente in fase di sperimentazione per approfondire l'analisi del comportamento del sistema LoRa.

E' infine possibile specificare il tempo da far intercorrere tra due invii del beacon dell'iGate o del tracker qualora non venga utilizzata la funzione di "Agile beaconing" come descritta a seguire.

Per la modalità tracker il setup di questa schermata è più snella come configurazione; gli unici parametri richiesti sono la stringa da inviare come beacon (secondo il formato spiegato per la il caso iGate) e il periodo del beacon in sec.

Per il tracker è evidente che si assume che nell'applicazione sia sempre disponibile un modulo GPS da cui vengano dedotti sia la posizione che il tempo reale per il dispositivo.

3.4.2 Setup sottosistema APRS per connessione ad un server di servizio

Il dispositivo è in grado di collegarsi ad un server specializzato a raccogliere una serie di dati relativi agli spots APRS ricevuti dalla rete LoRa; tale interfaccia è basata su un semplice protocollo UDP e consente di raccogliere da parte di questo server ad hoc, in genere da locare sulla rete locale o sulla rete mesh a cui eventualmente il dispositivo, funzionante come iGate, risulta collegato, gli spots LoRa ricevuti in modo da poterli rappresentare su di una mappa geolocalizzata.

La documentazione di questa interfaccia è lasciata a successive attività di documentazione.

< **APRS Compatibility**

Payload Encapsulation Configuration:
Payload Encapsulation:

Payload Style Configuration:
Payload Style:

Repeater operation Configuration:
Repeater operation:

TX Location Compression Configuration:
LocationCompression:

TX Agile Beaconing Configuration:
AgileBeaconing:

Beacon Blacklist Configuration:
Black list:

LoraSync word Configuration:
LoraSync_word:

SAVE

Figura 35 - Setup APRS Compatibility

3.5 Setup APRS Compatibility (qualsiasi versione)

Nella applicazione APRS i pacchetti trasmessi devono essere "incapsulati" come in una busta per essere riconoscibili come delle entità correttamente trasmesse e ricevute; allo scopo sono possibili diverse modalità.

Nella implementazione SARIMESH l'incapsulamento preferito è quello cosiddetto AX.25 che trae la sua origine proprio da tale protocollo e presenta i migliori attributi di compatibilità con altre applicazioni di trasmissione dati in uso (es. modalità TNC KISS).

Esistono altre implementazioni di LoRa APRS in cui si utilizza un diverso tipo di incapsulemento: in particolare viene supportata anche la modalità diffusa nelle implementazioni austriaca e tedesca che indichiamo sinteticamente come modalità "OE_Style.

Il SW in ricezione consente di ricevere pacchetti che utilizzano entrambe le modalità di incapsulamento, realizzando le eventuali azioni di

adattamento necessarie alla compatibilità dei dati trasportati; in trasmissione è necessario invece specificare quale tipo di incapsulamento utilizzare; a questo provvede la schermata a fianco che quindi consente di impostare il tipo di incapsulamento da usare in trasmissione.

Data la natura sperimentale di questo progetto e dell'uso della tecnologia LoRa in quanto tale, si sono definite delle modalità di lavoro che in svariati punti divergono dallo standard APRS; un esempio molto diffuso è quello ora illustrato della modalità di incapsulamento dei dati che costituiscono il pacchetto utile da trasmettere, ampiamente utilizzato nella stragrande maggioranza dei dispositivi (in tecnologia LoRa) funzionanti.

Nella schermata di compatibilità è possibile specificare quali opzioni non standard si intende utilizzare e quali no: in questo modo è possibile profilare un generico dispositivo per operare in maniera completamente conforme agli standard o che si adatti al meglio alle esigenze di sperimentazione.

Sono presenti in particolare le seguenti **opzioni (che influenzano solo il comportamento in trasmissione dei dispositivi)**:

- **payload encapsulation**: già descritta in precedenza
- **payload style configuration**: abilita o meno i campi aggiuntivi descritti al paragrafo precedente (working conditions e riporto parametri dei nodi attraversati)
- **repeater operation configuration**: abilita la funzione digipeater o meno (solo in caso di funzionamento come iGate)
- **location compression configuration**: abilita la compressione dei campi di location
- **TX Agile Beaconing configuration**: abilita la funzione di beaconing adattivo (vedi seguito)
- **Beacon BlackList configuration**: abilita la black list per le aree protette
- **LoRa sync word configuration**: setta il valore della sync word da usare nei pacchetti LoRa

Una nota particolare vale la pena di riportare relativamente alla funzione di beaconing, ovvero di modalità di generazione dei beacon da parte di un dispositivo.

La funzione dei beacon è ovviamente quella di riportare dei dati relativi allo stato del dispositivo che emette il beacon: la natura di questi dati può essere molto diversificata e in genere include come dato principale una "location" ovvero una posizione individuabile o tramite delle coordinate geo o tramite un QRA locator; in alcuni casi è anche possibile evitare di trasmettere questo dato di base in quanto per esempio non soggetto a variazione (come per es. per una stazione meteo o per un iGate fisso); in questi casi in genere il dato viene trasmesso opzionalmente ogni certo tempo in modo da accorciare la lunghezza dei pacchetti evitando di trasmettere dati che non mutano nel tempo.

Nel caso delle sperimentazioni delle applicazioni APRS per la tecnologia LoRa diventano interessanti una serie di dati relativi ovviamente agli aspetti target della sperimentazione quali ad es. i valori di intensità di segnale ricevuto, di rapporto segnale rumore ed eventualmente di perdite di pacchetti o altri dati simili.

Un secondo aspetto riguarda la cadenza di emissione dei beacon: infatti mentre nel classico uso di APRS come modo per evidenziare il movimento sul territorio di un certo dispositivo ha senso emettere dei pacchetti solo quando ci sta qualcosa di per es. statico... ad es. una sosta in un viaggio o una nuova sede raggiunta, nel caso delle sperimentazioni LoRa diventa interessante ed importante l'aspetto "mappatura radio" del territorio allo scopo di valutare il livello di copertura e le caratteristiche di copertura radio che il protocollo LoRa consente di avere.

In questa ultima situazione diventa quindi interessante poter emettere dei beacon con una logica legata appunto all'utilizzo dei dati raccolti... per es. usare come base di riferimento per l'emissione degli spot il cambio di posizione o il percorso seguito dal dispositivo mobile: un

esempio è per rilevare la mappa di copertura radio di una certa zona oppure il livello di copertura di un certo percorso (es. un percorso di montagna o una strada particolare).

La funzione di “Agile Beaconing” si prefigge l’obiettivo di dosare i tempi di emissione dei beacon secondo diversi criteri in modo da ottenere l’obiettivo di cui sopra.

Nel caso specifico si utilizzano i dati acquisiti in tempo reale dal GPS (in particolare posizione, direzione di marcia, tempo, velocità e altitudine) per dosare appropriatamente i tempi di emissione dei beacons.

Esistono due principali sotto-modalità: modalità basate semplicemente sulla distanza percorsa o modalità basata su “eventi” intendendosi con tale termine degli insiemi di condizioni predefinite e significative del percorso che si sta seguendo.

E’ poi possibile disabilitare completamente la funzione Agile ricadendo in questo caso nella modalità base di emissione dei beacons , ovvero la modalità a tempo costante; in ogni caso anche se è attiva la modalità “Agile”, il parametro di tempo indicato nella schermata di setup della funzione APRS resta attiva in modo da poter inviare dei beacon al peggio ogni certo tempo se non ci sono eventi.

Va osservato che data la natura dei dati di posizionamento derivati dal GPS è normale che tali dati siano affetti da un errore in particolare per la posizione che è valutabile in alcuni metri e che dipende dalle condizioni di ricezione del segnale GPS: per evitare un invio eccessivo di beacons a causa di tali errori viene comunque evitato di trasmettere dati di posizione che differiscono di pochi metri.

Una simile limitazione esiste per l’intervallo di tempo tra beacons: in questo caso , per evitare situazioni di congestione vengono evitate trasmissioni di beacons che siano temporalmente troppo vicine a beacons precedenti;.

In ogni caso è possibile sempre inviare immediatamente un beacon manualmente premendo il tastino presente sui dispositivi.

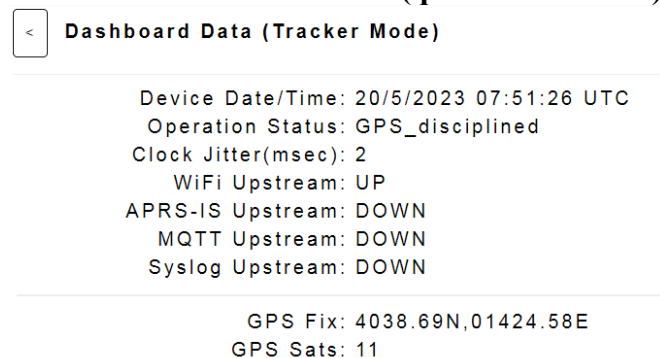
Una ulteriore funzione legata al beaconing è quella di evitare di emettere beacons quando il mobile si trovi in particolari aree da cui si vuole evitare di trasmettere dati per motivi o di privacy o di tipo tecnico legati per es. a problemi di congestione radio.

Allo scopo è possibile definire un insieme di aree, individuabili tramite un centro ed un raggio, tali che se ci si trova all’interno di una di tali aree il beacon viene soppresso.

Tale funzione è abilitabile tramite la schermata di compatibilità.

Tutti i parametri caratteristici della funzione di beaconing sono allo stato attuale NON modificabili da GUI ma richiedono di modificare dei parametri presenti in opportune tabelle nel codice sorgente; in un prossimo futuro se la funzione si dimostra di reale interesse si potrà implementare anche per questa funzione una opportuna schermata per consentire la modifica dei diversi parametri da GUI.

3.6 Schermata Dashboard (qualsiasi versione)



Esiste una pagina dell’interfaccia di gestione grafica (GUI) che consente di avere un colpo d’occhio generale del funzionamento di un dispositivo: la DashBoard.

Questa pagina è organizzata in varie sezioni e consente di avere sotto controllo i principali aspetti del funzionamento di un dispositivo sia dal punto di vista operativo che dal punto di vista della configurazione e dello stato dei principali aspetti del dispositivo.

A seguire vengono illustrate le varie sezioni e il significato dei dati ivi riportati.

La prima sezione riporta i principali parametri del modo operativo del dispositivo dal punto di vista dei principali sottosistemi; in particolare vengono riportati data e tempo del dispositivo e il modo in cui tali parametri sono dedotti ovvero se il dispositivo è agganciato ad un GPS ovvero al protocollo NTP o se opera in maniera standalone per es.

Viene anche riportato il valore di Clock Jitter in msec ovvero una indicazione di quanto il tempo locale varia nel tempo in relazione al tipo di sincronizzazione in atto.

Viene poi riportato lo stato della connettività WiFi upstream, lo stato della connessione al sistema APRS-IS e lo stato dei sottosistemi MQTT e syslog presenti sul dispositivo; tutte queste

<	Dashboard Data (iGate Mode)
Device Date/Time: 20/5/2023 08:09:49 UTC	
Operation Status: NTP_disciplined	
Clock Jitter(msec): 37	
WiFi Upstream: UP	
APRS-IS Upstream: UP	
MQTT Upstream: UP	
Syslog Upstream: UP	
GPS Fix: Not_Available	
GPS Sats: 0	

funzionalità sono in pratica dipendenti dalla presenza di una connessione upstream verso internet e servono per un uso complesso dei dispositivi; per un uso base dei dispositivi in modalità tracker non sono funzioni richieste attive, mentre per l'uso come iGate l'unica funzione che deve essere necessariamente attiva è la connessione WiFi e la connessione ad APRS-IS.

Questa sezione riporta anche lo stato del sottosistema GPS ed in particolare il valore di FIX attuale ed il numero di satelliti su cui è

basato.

Questa sezione consente anche di sapere in quali modalità operative macroscopiche il dispositivo sta operando; dall'esempio sopra si legge per esempio che opera per esempio in modalità tracker, dalla figura a fianco invece si evidenzia un esempio di dispositivo che opera in modalità

iGate. In questo caso per es. si vede che il GPS non è attivo e quindi il tempo locale è derivato dal protocollo NTP internet; come risultato si evidenzia che il clock locale presenta un jitter di alcune decine di msec come è normale per il tempo derivato da internet.

Il valore di Jitter è importante per alcune funzionalità, descritte nel seguito, che richiedono una base dei tempo pseudo-sincrona per il loro funzionamento.

Una seconda sezione della DashBoard riporta sinteticamente lo stato della parte di ricezione del dispositivo per quanto riguarda l'aspetto dei messaggi ricevuti.

In particolare questa sezione riporta solo l'ultimo pacchetto ricevuto e per tale pacchetto presenta innanzitutto il "report di ricezione" ovvero i dati che il nodo riporterà eventualmente ad un nodo a valle qualora il pacchetto venga "ripetuto" ovvero riporterà verso APRS-IS: questo record contiene l'identità del nodo che sta ricevendo il pacchetto ed i valori di livello di segnale, rapporto segnale/rumore e delta di frequenza con cui il pacchetto è stato ricevuto. I valori di intensità di segnale sono in dbm, quelli di SNR in db e quelli di frequenza in Hz.

Nella successiva parte viene riportato il payload contenuto nel pacchetto ricevuto: se il pacchetto è ricevuto come "CRC Errored" significa che il pacchetto pur essendo stato recuperato risulta affetto da errori non correggibili per cui può contenere caratteri errati o simboli non stampabili; la strategia è quella di riportare comunque in contenuto recuperato per eventuali successive azioni.

La terza parte di questa sezione riporta il Path con cui l'ultimo pacchetto è stato ricevuto: vengono solo riportati le componenti del path identificabili ovvero in genere solo l'originale

sorgente del pacchetto ricevuto (indipendentemente da eventuali ripetizioni che il pacchetto ha subito) e l'ultimo nodo da cui il pacchetto è transitato se si riesce ad identificarlo.

```
LoRa_rx_packets: 117
LoRa_tx_packets: 42
LoRa_rx_AX25_packets: 0
LoRa_tx_AX25_packets: 0
LoRa_rx_OEStyle_packets: 117
LoRa_tx_OEStyle_packets: 42
LoRa_rx_native_packets: 0
LoRa_tx_native_packets: 0
LoRa_lost_packets: 0
LoRa_CRC_errored_packets: 0 / 0
LoRa_UMN_errored_packets: 0
LoRa_CAD_errors: 3
LoRa_ReSched_packets: 0

AprsIS_rx_packets: 0
AprsIS_tx_packets: 158
AprsIS_dropped_packets: 0
AprsIS_relayed_packets: 0

IPC_lost_msgs: 0
```

pacchetto rischedulati in fase di trasmissione sempre a causa di problemi di occupazione del canale radio .

Lato APRS-IS (ovvero connessione APRS da e verso internet) vengono riportati i pacchetti ricevuti e trasmessi nonché quelli ripetuti verso la rete LoRa o soppressi in quanto non conformi alle policy di ripetizione verso il lato rete LoRa.

```
LoRa_OnAirTime(msec): 3973
LoRa_DutyCycle(%): 0.22
LoRa_ChanCong(%): 0.40
LoRa_ENL(dbm): 0.00
LoRa_FreqJitter(ppm): -0.78
CPU Temperature(C°): 36.11
CPU_UpTime(secs): 74371
```

REFRESH

nella schermata di setup della GUI APRS.

L'ultima sezione della Dashboard riporta una serie di dati relativi al nodo nella sua interezza; in particolare il primo parametro riporta la media mobile dei valori di tempo di trasmissione (OnAirTime in msec) dei pacchetti trasmessi , il DutyCycle in % di occupazione del canale ovvero quanto del tempo di trasmissione del nodo è stato impiegato in realtà dal nodo , ed un ulteriore parametro (euristico) che riporta una indicazione approssimativa del livello di congestione del canale radio come percepito dal nodo).

La successiva sezione della Dashboard riporta una serie di contatori che il nodo mantiene per il traffico di pacchetti che lo attraversa, classificato in modo da evidenziare alcuni aspetti specifici dei vari tipi di traffico; il significato dei vari contatori è abbastanza intuitivo basandosi sul suo nome.

In particolare vengono elencati i pacchetti in base al loro incapsulamento (AX_25 o OE_Style), in base alla direzione in cui sono stati visti (TX o RX dal punto di vista del nodo), in base al tipo di payload riconosciuto (APRS o formato nativo); vengono poi riportati i pacchetti persi (per indisponibilità del canale in trasmissione) ed i pacchetti ricevuti ma rivelatisi CRC_Errored, ovvero affetti da errori di trasmissione non correggibili.

Vengono poi riportati i pacchetti che non si sono riusciti a trasmettere a causa dell'occupazione del canale radio rivelato dal meccanismo di CAD (Channel Activity Detection) ; un ulteriore parametro è il numero di

L'ultimo parametro riportato è il numero di messaggi persi nel sistema di interprocess-communication (IPC) utilizzato internamente dal SW per far interagire tra loro le varie componenti del SW: questo parametro è un indicatore di congestione del SW.

E' da osservare che il contatore dei messaggi ricevuti dal lato APRS-IS sconta il filtraggio applicato in tale direzione dal nodo: in particolare l'unico filtro applicato è quello basato sulla distanza della sorgente come riportato dal sistema APRS-IS, ed impostato

Seguono due dati relativi al livello LoRa: il valore di “ExstimatedNoiseLevel” (ENL in dbm) valutato per il sito in cui è installato il nodo e la media mobile dei valori di Differenza di Frequenza misurata sui pacchetti ricevuti.

Il valore di ENL è una stima euristica del livello di rumore in dbm del sito calcolato considerando solo i pacchetti ricevuti con un rapporto segnale/rumore (SNR) negativo: quindi tale parametro sarà presente solo se il nodo ha ricevuto pacchetti con SNR negativo. Esso rappresenta il valori di dbm (valutato dal chip lora nel suo funzionamento interno) dell’intensità di segnale ai morsetti del chipset LoRa a prescindere dai valori di “guadagno di processo” che il protocollo LoRa consente di ottenere.

Il valore del Jitter indicato (in parti-per-milione) è la media mobile dei valori di differenza di frequenza diviso per il valore della frequenza di trasmissione che il nodo ha avuto modo di misurare sui pacchetti ricevuti indipendentemente da dove questi pacchetti siano stati trasmessi: quindi rappresenta una forma di stima di quanto il nodo corrente potrebbe essere “risintonizzato come

< HW Setup Configuration

HW Setup Configuration:

I2C Bus Pins:

i2c_sda: 21

i2c_scl: 22

SPI Bus Pins:

spi_sck: 5

spi_miso: 19

spi_mosi: 27

OLED Pins/Orient:

oled_addr: 0x3C ▼

oled_rst: 16

oled_orient: up ▼

LoRa Pins:

lora_cs: 18

lora_rst: 14

lora_dio: 26

GPS Pins:

gps_rx: 34

gps_tx: 12

SAVE

frequenza” per operare al meglio in rete LoRa nella situazione contingente di questo sito; questo valore viene presentato nella schermata di setup LoRa del nodo per essere in effetti usato come valore di correzione da applicare al setup della frequenza LoRa.

Gli ultimi valori presentati sono una stima del valore di temperatura della CPU del nodo e il valore del tempo di funzionamento (in sec.) dall’ultimo reboot del nodo stesso.

3.7 HW Setup Configuration

Con la versione 4.x del SW non viene più ufficialmente supportato l’utilizzo del SW su piattaforme HW diverse da quello Sarimesh sostanzialmente non tanto per un problema di natura funzionale quanto principalmente perchè il numero di piattaforme disponibili su internet e potenzialmente in grado di far girare il SW è aumentato notevolmente , per cui si pone il problema economico di avere a disposizione dei prototipi fisici su cui testare la compatibilità del SW , oltre al manpower necessario per effettuare i necessari non regression tests e gli eventuali adattamenti di codice o di configurazione.

In ogni caso essendo il SW molto modulare e basato su componenti HW decisamente standard, è agevole adattare il codice a girare su quasi tutte le schede basate sul processore ESP32 e che abbiano le componenti HW necessarie (es. un modulo GPS e un chipset LoRa).

Qualora il SW venga utilizzato su piattaforme HW diverse da quelle SARIMESH, in generale si può rendere necessario settare in maniera appropriata alcune features del SW in modo da adattarsi alle diverse caratteristiche dell’HW; in particolare quello che può differire, nell’ambito di

Figura 36 HW Setup Configuration

una certa tipologia di schedino HW, è il valore di alcuni pins del processore a cui si collegano le periferiche presenti sullo schedino.

Questa schermata consente di settare i pins del processore ESP32 da utilizzare per le seguenti funzioni:

- Bus I2C
- Bus SPI
- OLED pins, addr and orientation
- LoRa specific pins
- GPS specific pins

I valori da impostare vanno dedotti dalla documentazione HW dei dispositivi usati.

Per l'HW SARIMESH questa pagina riporta i valori utilizzati ma non modificabili in quanto invariati.

3.8 Setup syslog logging (qualsiasi versione)

Questa funzionalità è stata introdotta già nel SW Vr. 3.x anche se non è stata finora documentata.

La funzione consiste nella possibilità di inviare dei messaggi di tipo evento o errore verso un server esterno in cloud sfruttando il protocollo syslog ampiamente diffuso in applicazioni internet.

La funzione non è ancora dotata di una interfaccia grafica ed è stata implementata come “proof of concept” allo scopo di acquisire dati di funzionamento o condizioni di errore di un certo numero di nodi che facciano parte di un certo esperimento.

La funzione richiede che i nodi siano ovviamente in grado di raggiungere il cloud e quindi si assume che tipicamente si tratti di nodi collocati con una postazione dotata di connettività internet sia essa fissa o mobile.

Un uso tipico è quello per esempio di coordinare e controllare l'operation di nodi impegnati in una fase di sperimentazione quale quella descritta in uno dei paragrafi successivi di questo documento.

Qualora ci sia qualcuno interessato all'uso di questa funzionalità è possibile consultare il codice sorgente del SW o contattare l'autore del SW.

3.9 Setup sottosistema MQTT (qualsiasi versione)

Questa funzionalità è stata introdotta già nel SW Vr. 3.x anche se non è stata finora documentata .

La funzione consiste nella possibilità di sfruttare il protocollo MQTT, molto diffuso in ambito IoT, per interagire con un opportuno server proxy locato su cloud internet.

L'utilizzazione di questa funzionalità è quindi strettamente legata alle proprietà del protocollo MQTT e alle funzioni che si decidono di implementare a livello di cloud grazie alla presenza del server (definito broker) a cui l'istanza di MQTT si appoggia.

L'uso tipico di MQTT è quello di supportare funzioni di IoT quali ad es. la raccolta ed il display di dati quali temperatura, pressione, etc. da uno o più nodi remoti consentendo una agevole e flessibile organizzazione e presentazione dei dati.

Un secondo uso, meno noto ma estremamente interessante, è quello di operare come proxy per effettuare operazioni di manutenzione remota su un certo nodo senza assolutamente richiedere alcun intervento di riconfigurazione del gateway internet del nodo remoto che si intende controllare.

Nella “proof of concept” realizzata nel SW Sarimesh si implementa e si utilizza proprio questa seconda modalità allo scopo di acquisire dati di stato o l’intera dashboard di un generico nodo remoto e allo scopo di poter effettuare azioni remoto su un certo nodo in particolare per forzare il reboot del nodo.

La funzione MQTT essendo molto sperimentale è solo supportata a livello di codice sorgente del SW e non è stata ancora creata alcuna interfaccia grafica per la sua gestione; qualora ci sia interesse nello scoprire e provare ad utilizzare questa funzione è possibile consultare il codice sorgente e/o contattare l’autore del SW per supporto.

3.10 Setup sottosistema Beaconsing Nativo (qualsiasi versione)

Il SW Sarimesh contiene già dalla versione 3.x una funzionalità finora non documentata che definiamo “Beaconsing Nativo” : è una funzione sperimentale tendente a consentire di utilizzare un certo nodo per operare in modalità diverse con un approccio di tipo “time slicing” statico.

Tale funzione assume innanzitutto che un nodo operi in una modalità di tipo GPS_Disciplined o anche NTP_disciplined in modo tale che i valori di tempo locale siano sufficientemente “precisi” ovvero abbiano delle tolleranze sul loro valore contenuto entro massimo poche decine di msec.

Assunta questa ipotesi di lavoro, attivando la funzione di “Beaconsing Nativo” il nodo entrerà in una modalità di operazione particolare che andiamo a descrivere di seguito.

In pratica il tempo di funzionamento del nodo viene gestito in maniera dinamica sulla base di un certo “programma di utilizzo” costituito da una certa base temporale (per . es. 3 minuti) allineata in un certo modo (per es. su base tempo di orologio UTC) : tale programma specifica come operare nei sottointervalli di questo tempo di durata del programma (slice time) .

In ogni slice temporale il nodo potrà operare in maniera completamente diversa per ogni slice, ovvero con valori di frequenza, modo di emissione e potenza nonché di tipologia di incapsulamento dei pacchetti e relativo payload diverse.

La successione degli slice viene gestita automaticamente dal SW e assume che ci siano nella definizione dell’utilizzo di ogni singolo slice degli opportuni meccanismi di protezione in modo da assicurare che i vari slice siano in pratica “disgiunti”.

Un esempio che è stato utilizzato per circa un anno in una particolare sperimentazione è consistito nell’ implementare una modalità mista APRS/Beacon Nativo nel quale una coppia di nodi operava in modalità Nativa su una certa frequenza per 100 sec con un certo SF e potenza per stimolare condizioni molto marginali (SF12 / BW 10.4 Khz) del collegamento e per il rimanente tempo (80 secondi) in modalità APRS con i parametri standard attuali (SF12 / BW 125 Khz) su una differente frequenza.

L’obiettivo era acquisire e confrontare i parametri tipici del collegamento nelle due diverse modalità operative.

L'esperimento è stato realizzato su una tratta radio di 120 Km ed è documentato sul sito Sarimesh.net a cui si può accedere per i dettagli (<http://www.sarimesh.net/lora-beacon-propagation-test/> e <http://www.sarimesh.net/lora-propagation-test-bed-2/>).

Per il settaggio della modalità operativa mista è stata predisposta una pagina ad hoc nella GUI che di seguito viene illustrata.

Beacon Configuration

Beacon Configuration:

Payload content:

BeaconId: \$1 2 bytes

BeaconSeqNbr: enable (+1 byte) ▼

BeaconUnixTime: disable ▼

BeaconLocation: disable ▼

BeaconFreq: disable ▼

BeaconPower: disable ▼

BeaconWorkConditions: enable (+2 byte) ▼

Beacon Operation:

BeaconEngineType: txEnable ▼

BeaconRun: singlePhase_100 ▼

singlePhase_Config

BCN_LoRa_Vector: 0, 433.800, 10.4, 12, 8, 0x34, 22, -2, 10

(format) sqnbr, loraFreq, bw, sf, cr, sync, pwr, ppm, prlen

(example) 0, 433.725, 7.8, 11, 8, 0x34, 10, 0, 15

BCN_TimeSlotOffset(secs): 0

BCN_TimeSlotSync: OnMinuteModulo_4 ▼

BeaconStatsCollector

StatsCollector_IP: 192 . 168 . 2 . 150

StatsCollector_Port: 33330

SAVE

Per attivare la funzionalità di operation mista è sufficiente attivare la funzione di Beacon Nativo sulla pagina GUI del Modo di Operation.

Innanzitutto è stato definito un tipo di payload “nativo” in cui di fatto vengono, in pochissimi bytes, sintetizzati una serie di dati rilevanti per un collegamento e selezionabili nella prima sezione della schermata; per individuare un nodo si usa semplicemente un byte con un codice numerico; il massimo numero di nodi attivi in un certo esperimento dipende dal programma di slicing definito a seguire.

La codifica dei vari parametri è definita nel SW e per gli interessati basta andare a consultare i sorgenti del SW.

Segue la sezione di “Beacon Operation” che specifica il tipo di programma e se le varie fasi sono di sola ricezione o di tipo bidirezionale.

La voce “BeaconRun” definisce i vari tipi di utilizzo del tempo di programma e può assumere i valori SinglePhase (di varia durata

da 18 sec a 100 sec) o tuning mode nel quale il beacon segue automaticamente una sequenza di fasi di trasmissione/ricezione con diversi valori di frequenza per scoprire eventuali shift di frequenza).

Segue poi una sezione che specifica come la fase Beacon Nativo si svolge: in pratica specifica sinteticamente le condizioni di lavoro a livello LoRa da utilizzare e l’offset temporale rispetto al tempo di programma da cui partire con la fase di beacon nativo.

La fase di tipo APRS viene definita per complemento al tempo di durata del programma e utilizza i parametri LoRa / APRS del nodo come descritti nelle relative schermate.

Per garantire la coesistenza di più nodi in una fase di esperimento si sfrutta il parametro “Identità del nodo” con la semplice logica che in un certo intervallo temporale solo un certo nodo possa trasmettere; in particolare si sfrutta il parametro NodeId definito sopra come indice per attivare la fase di trasmissione di uno solo dei nodi in una frazione del tempo assegnato alla fase “beacon nativo” del programma; questa modalità operativa è solo disponibile selezionando una durata della fase beacon nativo di 100 sec.; per tutti gli altri valori si assume sempre una singola coppia di nodi come target dell’esperimento.

La funzione ora descritta è come si vede ancora molto sperimentale e rappresenta esclusivamente una “proof of concept” da eventualmente meglio definire e sfruttare in futuro.

4 Interfaccia di Debug Remoto

I dispositivi della famiglia LoRa Beacon sono dotati di una interfaccia di debug remoto che consente di accedere ad una serie di funzionalità pensate per un uso avanzato dei dispositivi ovvero per accedere a delle funzioni di monitoraggio del funzionamento in tempo reale senza richiedere il collegamento ad un computer via USB o seriale e quindi sfruttabile anche da remoto per dispositivi per es. locati in un punto collegato via rete internet.

L'accesso alla interfaccia di Debug Remoto utilizza il protocollo telnet dotato di un semplice sistema di sicurezza basato su password.

L'accesso a tale funzionalità può avvenire sia utilizzando un classico client Telnet disponibile per tutte le piattaforme di computer esistenti, sia via una Applicazione web sfruttando un qualsiasi browser internet.

Indipendentemente dalla modalità di accesso alla interfaccia di Debug Remoto, sono disponibili una serie di comandi che corrispondono ad altrettante funzioni di monitoraggio o di debug.

Di seguito vengono descritte le due modalità di accesso e successivamente i comandi di debug disponibili.

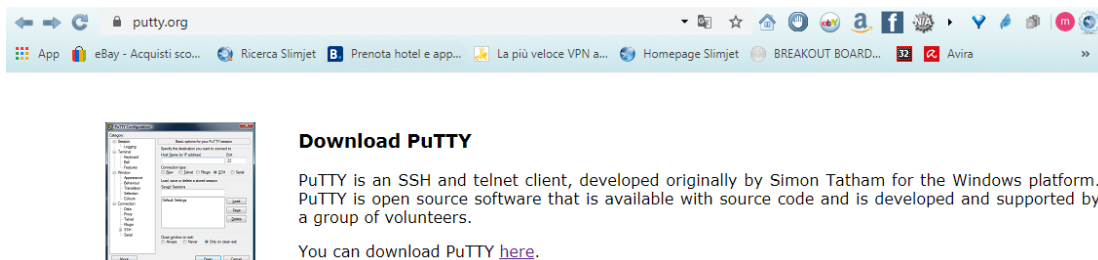
4.1 Accesso alla IF di Remote Debug tramite Telnet Client

Per utilizzare questa modalità di accesso è sufficiente procurarsi un qualsiasi client Telnet quale ad esempio il classico “putty” scaricabile dal seguente URL: <http://putty.org>

Scaricato il file putty.exe corrispondente alla propria piattaforma PC che si intende utilizzare è sufficiente avviare il file senza necessità di installazione.

Selezionare dalla schermata che viene proposta le seguenti opzioni in aggiunta a quelle di default:

- Terminal: Implicit CR in every LF
- Session: Other --> Telnet
- Host Name: indirizzo IP in rete LAN del dispositivo



Apparirà una schermata che richiede l'immissione di una password: default è esp32

Immettendo la password apparirà l'interfaccia a comandi della console di debug del dispositivo.

Tale interfaccia è basata sulla modalità “comando” ovvero sono resi disponibili una serie di “comandi” che permettono di accedere alle funzioni disponibili.

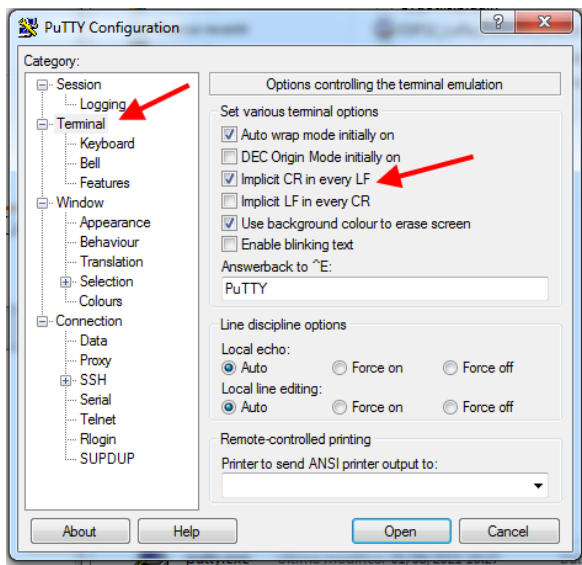


Figura 38 Putty setup - 1

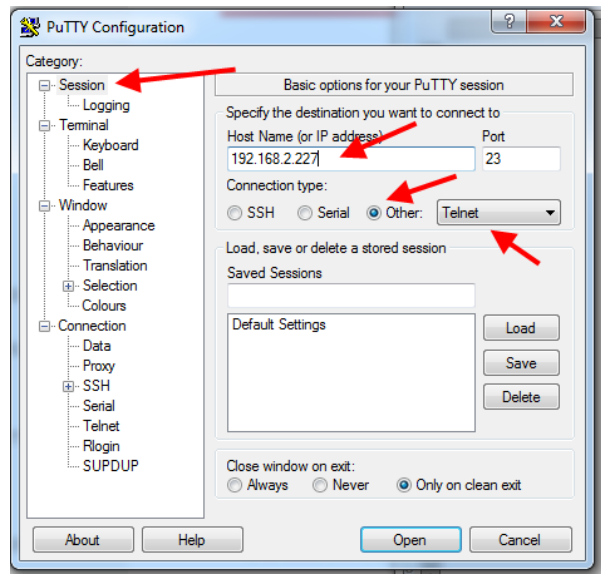


Figura 39 Putty setup - 2

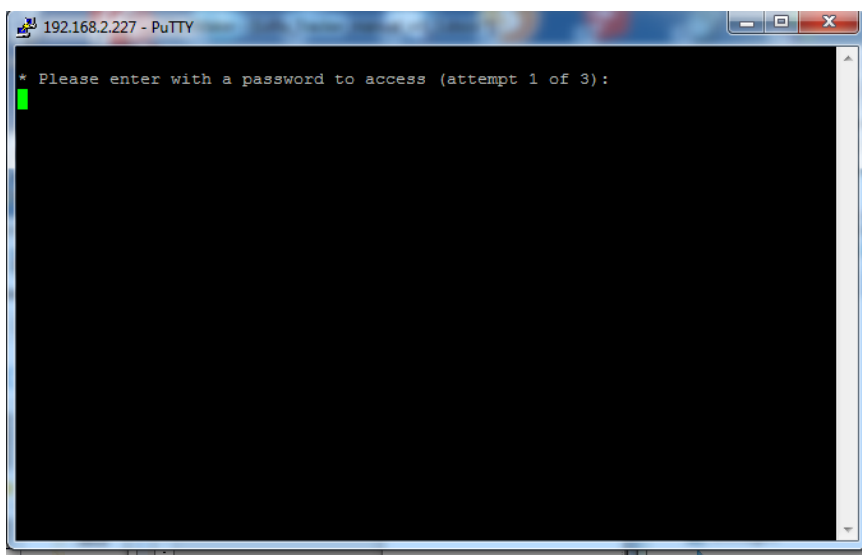


Figura 40 Schermata richiesta password console di debug

Inoltre di default la console riporta una serie di informazioni diagnostiche prodotte dal SW del dispositivo, sulla base delle funzioni di debug abilitate allo stato.

La schermata iniziale, dopo il login, elenca i comandi disponibili ed il loro significato sinteticamente; alcuni comandi sono definiti “Application Commands” e servono ad attivare funzioni specifiche del SW.

Alcuni dei comandi disponibili rappresentano l’esatto equivalente di quanto visualizzabile sulla console seriale via USB accessibile tipicamente dall’interno dell’ambiente arduino.

Altri comandi rappresentano funzioni aggiuntive disponibili esclusivamente tramite questa interfaccia di Remote Debug e vengono descritti di seguito in dettaglio.

4.1.1 Comando “gps_status”

```

192.168.2.227 - PuTTY
* Please enter with a password to access (attempt 1 of 3):
esp32
* Password ok, allowing access now...
*** Remote debug - over telnet - for ESP32 - version 3.0.5
* Host name: ESP32-10521C691A18 IP:192.168.2.227 Mac address:10:52:1C:69:1A:18
* Free Heap RAM: 149844
* ESP SDK version: v3.3.1-61-g367c3c09c
*****
* Commands:
  ? or help -> display these help of commands
  q -> quit (close this connection)
  m -> display memory available
  v -> set debug level to verbose
  d -> set debug level to debug
  i -> set debug level to info
  w -> set debug level to warning
  e -> set debug level to errors
  s -> set debug silence on/off
  l -> show debug level
  t -> show time (millis)
  profiler:
    p -> show time between actual and last message (in millis)
    p min -> show only if time is this minimal
    P time -> set debug level to profiler
    c -> show colors
  filter:
    filter <string> -> show only debugs with this
    nofilter -> disable the filter

* Application commands:

reboot
gps_status
temperature
wifi_scan
display_config
i2c_scan
selftest_start
selftest_stop
show_stats
show_events
log_display
fram_dump
fram_log_set
fram_log_reset

* Please type the command and press enter to execute.(? or h for this help)
***

```

Figura 41 Schermata iniziale console di Debug Remoto

Il comando “gps_status” consente di avere una serie di dettagliate informazioni sullo stato del GPS: in particolare fornisce i seguenti parametri:

- Elenco dei satelliti attualmente utilizzati per il fix: numero di satelliti utilizzati, valori di PRN, elevazione, di azimuth e di rapporto SNR per ogni satellite utilizzato per il fix
- parametri relativi al fix : latitudine, longitudine, età del fix, data, tempo, altezza, velocità, parametri di qualità del fix

```

gps_status
(D) (gps_status_header) (C1)
(D) (gps_status_header) (C1)
(D) (gps_status_header) (C1)
(D) (gps_status) (C1) Sats=12 Nums=13 14 20 30 Elevation=18 19 57 77 Azimuth=281 167 291 227 SNR=31 0 30 18
(g) loop-extended (C1) GPS status -> 12 1.2 40.644817 14.409652 273 08/01/2021 19:39:37 385 75.80 23.41 0.26 NNE 1642 322.20 NW 62376 179 4

```

GPS Satellites used

GPS Fix data (according to gps_status_header)

Figura 42 Uscita el comando “gps_status”

4.1.2 Comando “temperature”

Fornisce una stima del valore della temperatura in °C a livello del chip ESP32 interno al dispositivo; questo valore va utilizzato come una indicazione di massima del valore di temperatura essendo la sua determinazione basata su di una feature non documentata del chip ESP32.

4.1.3 Comando “wifi_scan”

```
wifi_scan
(do_wifi_scan)(C1) wifi_scan start...scan done: 11 networks found
1: RFC2019-24 (-55)*
2: MikroTik-66BABF-24 (-59)*
3: SARIMESH (-59)*
4: ESP32-240AC4595AA8 (-73)
5: TIM-24138604 (-83)*
6: casa Izzo (-89)*
7: Vodafone-35188680 (-90)*
8: Vodafone-WiFi (-91)
9: FASTWEB-SK5JE6 (-91)*
10: FASTWEB-VJX4Y7 (-92)*
11: WOW FI - FASTWEB (-93)*
```

Effetua una scansione della banda WiFi dei 2.4Ghz e riporta la lista delle reti trovate e dei relativi livelli di segnale e presenza o meno di crittografia (*).

Figura 43 Output comando “wifi_scan”

“display_config”

```
display_config
(DisplayConfig)(C1) Display active Configuration data
ESP_config.DeviceName = LoRa Tracker Vr4.1
ESP_config.DeviceId = Not Available
ESP_config.cpu_type = ESP32 Dev V4
ESP_config.dhcp = 1
ESP_config.daylight = 1
ESP_config.Update_Time = 0
ESP_config.timezone = 10
ESP_config.IP = 192.168.2.60
ESP_config.Netmask = 255.255.255.0
ESP_config.GatewayIP = 192.168.2.1
ESP_config.DnsIP = 8.8.8.8
ESP_config.ssid = RFC2019-24
ESP_config.password = 
ESP_config.ntpServer = ntp1.inrim.it
ESP_config.gps_debug = 0
ESP_config.LoRa_debug = 0
ESP_config.RTC_debug = 0
ESP_config.ezTime_debug = 0
ESP_config.pps_debug = 0
ESP_config.PE_debug = 0
ESP_config.BT_KISS_Mode = 0
ESP_config.Serial_KISS_Mode = 0
ESP_config.Tracker_Mode = 1
```

Figura 44 Output del comando “display_config”

4.1.4 Comando

Effetua un display in forma testuale della attuale configurazione del dispositivo in uso.

La configurazione è costituita da una lista di linee del tipo “chiave = valore” di cui l’immagine a fianco riporta un piccolo sottonsieme.

Il comando consente quindi di esplorare in dettaglio tutti i parametri di configurazione in uso anche senza disporre di una interfaccia grafica.

La nomenclatura usata per le chiavi dovrebbe rendere agevole individuare il significato dei vari parametri. elencati

4.1.5 Comando “show_stats”

```
show_stats
(D) (show_stats)(C1) ===== Stats start =====
(D) (show_stats)(C1)   SW_version           = 1.0.5_20210730_1611
(D)
(D) (show_stats)(C1)   LoRa_rx_packets       = 200
(D) (show_stats)(C1)   LoRa_tx_packets       = 66
(D) (show_stats)(C1)   LoRa_lost_packets     = 0
(D) (show_stats)(C1)   LoRa_CRC_errorred_packets = 0
(D)
(D) (show_stats)(C1)   LoRa_OnAirTime(msec)  = 602
(D) (show_stats)(C1)   CPU_Temperature(C°)   = 57.42
(D) (show_stats)(C1)   Processor_Uptime(secs) = 2083
(D) (show_stats)(C1) ===== Stats end =====
```

Riporta una lista di parametri sintetici relativi al funzionamento del dispositivo dettagliatamente per la parte LoRa.

Figura 45 Output comando “show_stats”

in uso sul dispositivo, il numero di pacchetti LoRa ricevuti e trasmessi, il numero di pacchetti LoRa persi (ovvero soppressi in trasmissione) a causa della congestione del canale radio, il numero di pacchetti LoRa ricevuti con valore di CRC a livello radio errato (in genere a causa di errori di trasmissione sul canale radio o di conflitto di accesso al canale radio), il tempo di attesa per la trasmissione dei pacchetti LoRa, la temperatura della CPU ESP32 ed il valore di tempo “uptime” della CPU dall’ultimo reboot.

In particolare riporta la versione SW

4.1.6 Comando “show_events”

Il dispositivo, nella versione basata su HW LoRa Beacon , presenta on board una memoria FRAM utilizzata per la tenuta oltre che dei dati di configurazione del dispositivo, anche di una serie di dati raccolti in tempo reale dal dispositivo e relativi sia ad eventi particolari che alla ricezione di spots a livello del sottosistema radio LoRa.

```
show_events
(D) (show_events)(C1) FRAM Log parameters: log_head=340 log_len=400 log_size=400 fram_base_log=2048
(D) (show_events)(C1) ===== Events Log Start =====
(D) ==> 20210801 09:33:41.000  cntr=308 [pntr=4352]==> [1627810421|EVENT|===== system reboot completed =====]
(D) ==> 20210801 08:34:28.000  cntr=303 [pntr=4712]==> [1627806868|EVENT|===== system reboot by GUI =====]
(D) ==> 20210801 21:38:01.000  cntr=297 [pntr=5144]==> [1627853881|EVENT|===== system reboot completed =====]
(D) (show_events)(C1) ===== Events Log End =====
```

Figura 46 Output comando “show_events”

Il comando “show_events” elenca la sola componente di “eventi” presenti nel log tenuto nella memoria FRAM; gli eventi riportati sono tipicamente legati ad azioni di management che implicano cambi di configurazione e ripartenze del processore.

Gli eventi sono marcati con il tempo reale nel quale sono stati registrati e quindi sopravvivono ai restart del dispositivo, per cui rappresentano uno strumento di “post mortem data analysys” per eventualmente diagnosticare situazioni anomale di funzionamento.

4.1.7 Comando “log_display”

Il dispositivo, nella versione basata su HW LoRa Beacon , presenta on board una memoria FRAM utilizzata per la tenuta oltre che dei dati di configurazione del dispositivo, anche di una serie di dati raccolti in tempo reale dal dispositivo e relativi sia ad eventi particolari che alla ricezione di spots a livello del sottosistema radio LoRa.

Il comando “log_display” riporta la lista completa della “coda circolare” in cui sono riportati tutti gli eventi e gli spots registrati in tempo reale nel corso del funzionamento del dispositivo.

L'utilizzo di una "coda circolare" per conservare questi dati consente di tenere traccia degli ultimi 400 eventi o spots registrati, scartando per ogni nuovo evento da registrare il più vecchio di quelli già registrati.

Il formato degli "spots" registrati è diverso per il caso di utilizzo come iGate o come Tracker:

- nel primo caso (uso come iGate) ogni spot riporta il nominativo della stazione sorgente come contenuto nello spot ricevuto via radio dall'iGate, la posizione contenuta nello spot ricevuto ed il livello di segnale e SNR con cui lo spot è stato ricevuto
- nel secondo caso (uso come tracker) lo spot riporta la posizione fisica in cui si trovava il tracker al momento della ricezione di uno spot, la frequenza cui è stato ricevuto lo spot ed i valori di livello del segnale e del SNR con cui lo spot è stato ricevuto

```
(D) (log_display)(C1) FRAM Log parameters: log_head=395 log_len=400 log_size=400 fram_base_log=2048
(D) (log_display)(C1) ===== Log Start =====
(D) ==> 19700101 00:15:10.093 cntr=399 [pntr=30560]====> [910|40.6435013|14.4095001|433.725|-88.00|10.75|0.00] d=3341105
(D) ==> 19700101 00:15:29.093 cntr=398 [pntr=30632]====> [929|40.6435013|14.4095001|433.725|-102.00|14.75|0.00] d=3341105
(D) ==> 19700101 00:15:30.093 cntr=397 [pntr=30704]====> [930|40.6435013|14.4095001|433.725|-41.00|11.50|0.00] d=3341105
(D) ==> 19700101 00:15:33.093 cntr=396 [pntr=30776]====> [933|40.6435013|14.4095001|433.725|-101.00|3.25|0.00] d=3341105
(D) ==> 19700101 00:15:41.093 cntr=395 [pntr=2048]====> [941|40.6445007|14.4090004|433.725|-96.00|8.00|0.00] d=3882434
(D) ==> 19700101 00:15:42.093 cntr=394 [pntr=2120]====> [942|40.6445007|14.4090004|433.725|-32.00|12.00|0.00] d=3882434
(D) ==> 19700101 00:15:43.093 cntr=393 [pntr=2192]====> [943|40.6445007|14.4090004|433.725|-96.00|8.50|0.00] d=3882434
(D) ==> 19700101 00:16:06.093 cntr=392 [pntr=2264]====> [966|40.6445007|14.4090004|433.725|-86.00|11.25|0.00] d=3882434
(D) ==> 19700101 00:16:11.093 cntr=391 [pntr=2336]====> [971|40.6446686|14.4095001|433.725|-39.00|10.50|0.00] d=8415512
(D) ==> 19700101 00:16:12.093 cntr=390 [pntr=2408]====> [972|40.6446686|14.4095001|433.725|-79.00|12.00|0.00] d=8415512
(D) ==> 19700101 00:16:13.093 cntr=389 [pntr=2480]====> [973|40.6446686|14.4095001|433.725|-42.00|12.00|0.00] d=8415512
(D) ==> 19700101 00:00:30.093 cntr=388 [pntr=2552]====> [30|EVENT]===== system reboot completed =====] d=4749550
(D) ==> 19700101 00:00:39.093 cntr=387 [pntr=2624]====> [39|0.0000000|0.0000000|433.725|-49.00|11.75|0.00] d=4749550
(D) ==> 19700101 00:00:39.093 cntr=386 [pntr=2696]====> [39|0.0000000|0.0000000|433.725|-86.00|11.75|0.00] d=4749550
(D) ==> 19700101 00:00:44.093 cntr=385 [pntr=2768]====> [44|0.0000000|0.0000000|433.725|-88.00|11.25|0.00] d=4749550
(D) ==> 19700101 00:01:10.093 cntr=384 [pntr=2840]====> [70|40.6448326|14.4096670|433.725|-49.00|11.25|0.00] d=5323454
(D) ==> 19700101 00:01:10.093 cntr=383 [pntr=2912]====> [70|40.6448326|14.4096670|433.725|-87.00|11.75|0.00] d=5323454
(D) ==> 19700101 00:01:41.093 cntr=382 [pntr=2984]====> [101|40.6448326|14.4096670|433.725|-47.00|11.25|0.00] d=10600898
(D) ==> 19700101 00:01:41.093 cntr=381 [pntr=3056]====> [101|40.6448326|14.4096670|433.725|-88.00|11.25|0.00] d=10600898
(D) ==> 19700101 00:01:43.093 cntr=380 [pntr=3128]====> [103|40.6448326|14.4096670|433.725|-49.00|12.00|0.00] d=10600898
```

Figura 47 Output comando "log_display": dati più vecchi

```
(D) ==> 20210801 20:34:19.003 cntr=16 [pntr=29336]====> [1627850059|40.6448326|14.4096670|433.725|-89.00|11.25|0.00] d=23
(D) ==> 20210801 20:34:49.003 cntr=15 [pntr=29408]====> [1627850089|40.6448326|14.4096670|433.725|-48.00|11.25|0.00] d=23
(D) ==> 20210801 20:34:50.003 cntr=14 [pntr=29480]====> [1627850090|40.6448326|14.4096670|433.725|-88.00|11.25|0.00] d=23
(D) ==> 20210801 20:35:20.003 cntr=13 [pntr=29552]====> [1627850120|40.6448326|14.4095001|433.725|-48.00|11.25|0.00] d=18
(D) ==> 20210801 20:35:21.003 cntr=12 [pntr=29624]====> [1627850121|40.6448326|14.4095001|433.725|-87.00|11.00|0.00] d=18
(D) ==> 20210801 20:35:52.003 cntr=11 [pntr=29696]====> [1627850152|40.6448326|14.4096670|433.725|-48.00|11.00|0.00] d=23
(D) ==> 20210801 20:35:52.003 cntr=10 [pntr=29768]====> [1627850152|40.6448326|14.4096670|433.725|-87.00|11.25|0.00] d=23
(D) ==> 20210801 20:36:23.003 cntr=9 [pntr=29840]====> [1627850183|40.6448326|14.4096670|433.725|-48.00|11.25|0.00] d=23
(D) ==> 20210801 20:36:23.003 cntr=8 [pntr=29912]====> [1627850183|40.6448326|14.4096670|433.725|-88.00|11.25|0.00] d=23
(D) ==> 20210801 20:36:54.003 cntr=7 [pntr=29984]====> [1627850214|40.6448326|14.4096670|433.725|-50.00|11.25|0.00] d=23
(D) ==> 20210801 20:36:54.003 cntr=6 [pntr=30056]====> [1627850214|40.6448326|14.4096670|433.725|-90.00|10.75|0.00] d=23
(D) ==> 20210801 20:37:21.003 cntr=5 [pntr=30128]====> [1627850241|40.6448326|14.4096670|433.725|-48.00|11.25|0.00] d=23
(D) ==> 20210801 20:37:22.003 cntr=4 [pntr=30200]====> [1627850242|40.6448326|14.4096670|433.725|-88.00|10.75|0.00] d=23
(D) ==> 20210801 20:37:25.003 cntr=3 [pntr=30272]====> [1627850245|40.6448326|14.4096670|433.725|-48.00|11.50|0.00] d=23
(D) ==> 20210801 20:37:26.003 cntr=2 [pntr=30344]====> [1627850246|40.6448326|14.4096670|433.725|-87.00|11.00|0.00] d=23
(D) ==> 20210801 20:37:41.003 cntr=1 [pntr=30416]====> [1627850261|40.6448326|14.4096670|433.725|-87.00|11.25|0.00] d=23
(D) (log_display)(C1) ===== Log End =====
```

Figura 48 Output comando "log_display": dati più recenti

```
(D) ==> 20210731 16:59:50.690 cntr=52 [pntr=9680]====> [1627750770|41.2881660|13.2609997|120CZW-9|-118.00|-8.00|0.00] d=121756
(D) ==> 20210731 17:00:01.690 cntr=31 [pntr=9752]====> [1627750801|41.2905006|13.2644997|120CZW-9|-114.50|-6.50|0.00] d=121684
(D) ==> 20210731 17:01:03.690 cntr=30 [pntr=9824]====> [1627750863|40.6445007|14.4091663|18FUC-10|-62.00|11.50|0.00] d=3682
(D) ==> 20210731 17:01:32.690 cntr=29 [pntr=9896]====> [1627750892|40.6119995|14.4008331|1Q8SO-10|-62.00|11.75|0.00] d=0
(D) ==> 20210731 17:03:03.690 cntr=28 [pntr=9968]====> [1627750983|40.6445007|14.4091663|18FUC-10|-62.00|11.75|0.00] d=3682
(D) ==> 20210731 17:03:08.690 cntr=27 [pntr=10040]====> [1627750988|41.3068352|13.2873335|120CZW-9|-110.00|-4.00|0.00] d=121325
(D) ==> 20210731 17:05:12.690 cntr=26 [pntr=10112]====> [1627751112|41.3193321|13.3013334|120CZW-9|-98.00|3.25|0.00] d=121311
(D) ==> 20210731 17:05:43.690 cntr=25 [pntr=10184]====> [1627751143|41.3224983|13.3056669|120CZW-9|-101.00|1.50|0.00] d=121262
(D) ==> 20210731 17:06:14.690 cntr=24 [pntr=10256]====> [1627751174|41.3268318|13.3078337|120CZW-9|-105.75|-1.75|0.00] d=121437
(D) ==> 20210731 17:06:34.690 cntr=23 [pntr=10328]====> [1627751194|40.6119995|14.4008331|1Q8SO-10|-62.00|11.75|0.00] d=0
(D) ==> 20210731 17:07:05.690 cntr=22 [pntr=10400]====> [1627751225|40.6445007|14.4091663|18FUC-10|-62.00|11.75|0.00] d=3682
(D) ==> 20210731 17:09:06.690 cntr=21 [pntr=10472]====> [1627751346|40.6445007|14.4091663|18FUC-10|-62.00|11.75|0.00] d=3682
(D) ==> 20210731 17:11:07.690 cntr=20 [pntr=10544]====> [1627751467|40.6445007|14.4091663|18FUC-10|-62.00|11.50|0.00] d=3682
(D) ==> 20210731 17:11:35.690 cntr=19 [pntr=10616]====> [1627751495|40.6119995|14.4008331|1Q8SO-10|-64.00|11.00|0.00] d=0
(D) ==> 20210731 17:13:09.690 cntr=18 [pntr=10688]====> [1627751589|40.6445007|14.4091663|18FUC-10|-62.00|11.50|0.00] d=3682
(D) ==> 20210731 17:15:10.690 cntr=17 [pntr=10760]====> [1627751710|40.6445007|14.4091663|18FUC-10|-62.00|11.50|0.00] d=3682
(D) ==> 20210731 17:16:36.690 cntr=16 [pntr=10832]====> [1627751796|40.6119995|14.4008331|1Q8SO-10|-62.00|12.00|0.00] d=0
(D) ==> 20210731 17:19:12.690 cntr=15 [pntr=10904]====> [1627751952|40.6445007|14.4091663|18FUC-10|-62.00|11.75|0.00] d=3682
(D) ==> 20210731 17:21:13.690 cntr=14 [pntr=10976]====> [1627752073|40.6445007|14.4091663|18FUC-10|-62.00|12.00|0.00] d=3682
(D) ==> 20210731 17:21:17.690 cntr=13 [pntr=11048]====> [1627752077|41.3328323|13.3161669|120CZW-9|-121.50|-9.50|0.00] d=121345
```

Figura 49 Esempio record di log relativi a spot ricevuti e riportati da un tracker

La figura precedente riporta un esempio di log per un iGate: il motivo del diverso contenuto informativo deriva dalla impossibilità nel caso tracker di identificare la sorgente del messaggio LoRa.

In ogni spot è anche contenuto un campo d (distanza) il cui significato è lasciato ad una futura fase di documentazione; allo stato è da considerare un campo non significativo.

4.2 Accesso alla IF di Remote Debug tramite Web App

Per utilizzare questa modalità di accesso è necessario installare sul proprio PC un particolare applicativo di nome Remote Debug WEB App scaricabile dal seguente URL:
<https://github.com/JoaoLopesF/RemoteDebugApp#installing>

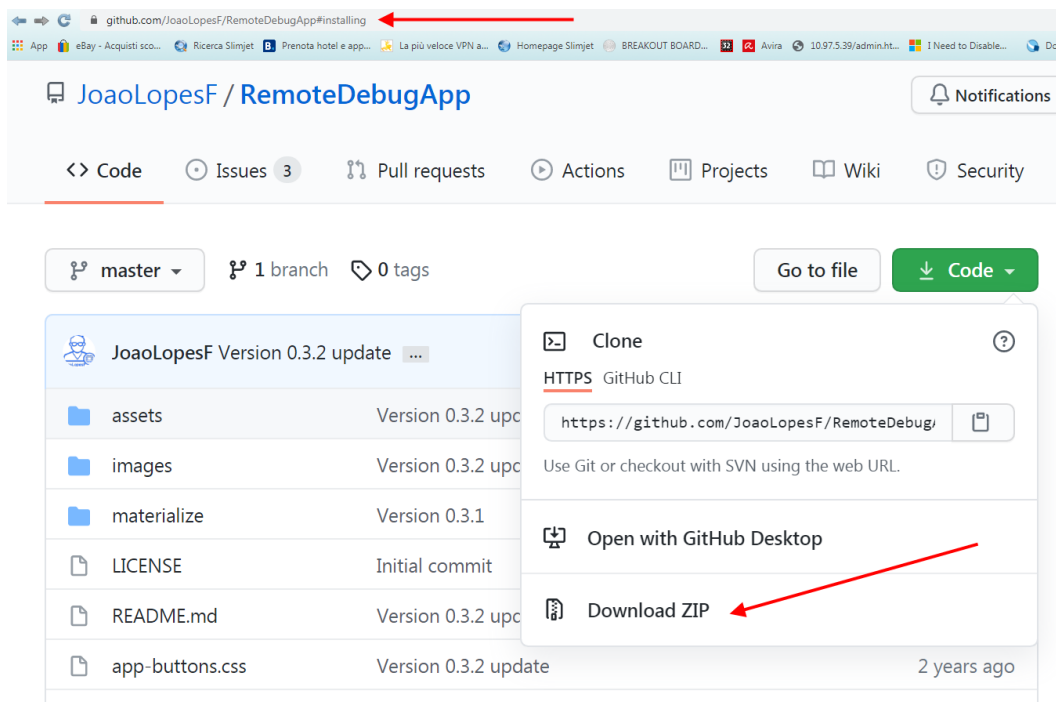


Figura 50 Pagina di download Remote Debug App

La figura sopra riporta una immagine della pagina di download; per scaricare la App è necessario selezionare la casella “Code” e dalla successiva finestra di selezione che viene presentata scegliere l’opzione “Download”.

L’archivio .zip così ottenuto va espanso sul PC che si sta utilizzando in una directory a piacere; quindi con il tool locale “esplora risorse” selezionare all’interno della directory in cui è stato espanso l’archivio il file index.html ed aprirlo con un browser tipo chrome o firefox.

La figura successiva riporta l’apparenza della finestra del browser chrome che si ottiene.

Come si potrà notare è una classica interfaccia web che richiede inizialmente di introdurre l'indirizzo IP del dispositivo da monitorare.

E' poi necessario inserire nella finestra di input la password di accesso (di default esp32).

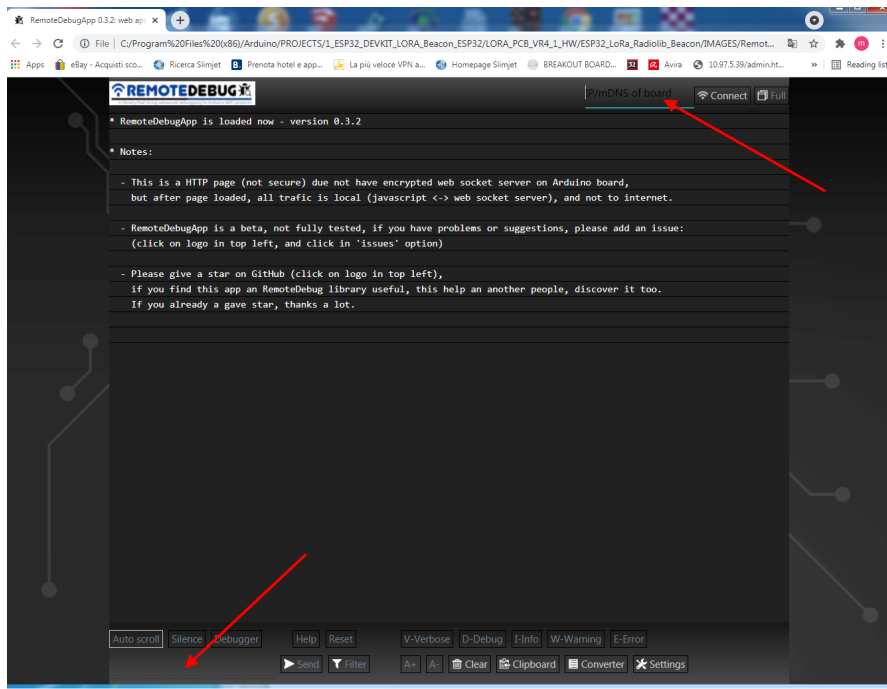


Figura 51 Schermata iniziale Remote Debug App

poter introdurre un “Filtro” sul contenuto delle informazioni presentate nella finestra di lavoro in modo per es. da displayare solo dei particolari messaggi; questa funzione quindi si rende molto utile in caso di attivazione delle funzionalità di debug presenti sul dispositivo.

A valle dell'introduzione della password ci si troverà una finestra del tutto simile a quella che si sarebbe ottenuta utilizzando l'applicazione putty di cui al paragrafo precedente.

Valgono a questo punto esattamente tutte le funzionalità e considerazioni svolte al paragrafo precedente.

Utilizzando questo tipo di interfaccia sono disponibili alcune funzioni aggiuntive tra cui forse la più interessante è quella di

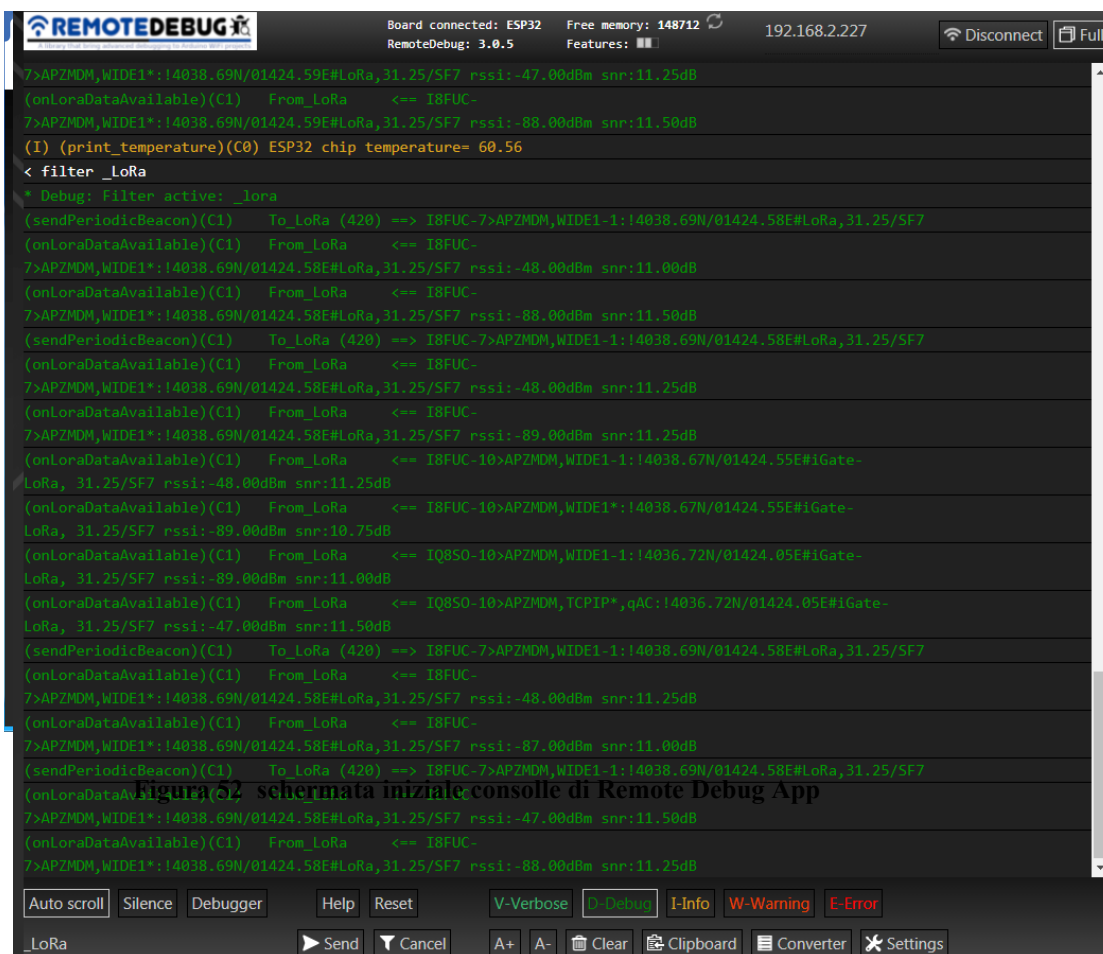


Figura 52 schermata iniziale console di Remote Debug App

Figura 53 Remote Debug App Filter function

5 Porting del SW LoRa Beacon su altre piattaforme HW

Il SW del progetto LoRa Beacon è stato pensato per poter essere utilizzato non solo sulla piattaforma HW LoRa Beacon descritta precedentemente, ma anche su dispositivi HW simili, che cioè utilizzino lo stesso tipo di microcontrollore (ESP32) e che comprendano un insieme di funzioni HW simili, quali ad es. un modulo GPS e/o un modulo LoRa.

Allo scopo di poter utilizzare le funzionalità SW presenti ovviamente è necessario che l'HW sia in grado di fornire le funzionalità HW sottese da tali funzionalità SW.

A seconda dello “schedino” che si vuole utilizzare è quindi necessario verificare questo aspetto e settare opportunamente alcuni parametri a livello della configurazione di “build” del SW in modo da attivare o disattivare dei pezzi di SW relativi.

Allo stato sono stati effettuati i porting del SW su due schedini particolari che rappresentano degli esempi significativi di dispositivi “all-in-one” reperibili sui soliti portali cinesi.

L'operazione di “porting” consiste nelle seguenti operazioni:

- configurare opportunamente l'ambiente Arduino ovvero PlatformIO di sviluppo del SW selezionando opportunamente la piattaforma HW da supportare
- analizzare la struttura HW del dispositivo e settare opportunamente il file “master_config.h” di configurazione del SW
- effettuare una “build” del SW collegandosi direttamente al target tramite USB
- effettuare i necessari test di “non regressione” per verificare il corretto funzionamento dell'insieme HW-SW, per le sole funzioni supportabili dall'HW in uso
- verificare che funzionalmente il dispositivo si comporti come atteso.

Una volta completato con successo il lavoro di porting è possibile generare una immagine completa del SW caricabile sullo schedino tramite il classico tool di download del microcontrollore ESP32, senza quindi richiedere necessariamente la presenza dell'ambiente di sviluppo SW Arduino ovvero PlatformIO per coloro che intendono utilizzare semplicemente il SW senza necessariamente apportare modifiche allo stesso, ma sfruttando l'interfaccia di configurazione fornita per l'impostazione del dispositivo e per effettuare eventuali sperimentazioni con lo stesso.

Nei paragrafi seguenti viene riportata la documentazione delle impostazioni da utilizzare per il tool di download in relazione ai diversi tipi di schedini da supportare.

5.1 Installazione SW su dispositivo TTGO T-beam-V1-2019

Questo schedino è la prima versione di una lunga serie di dispositivi resi disponibili dallo stesso Vendor e che pur condividendo lo stesso nome commerciale T-Beam hanno mostrato un equipaggiamento HW diverso anche parecchio dalla versione originale. Facendo una ricerca su internet è possibile approfondire questo argomento (<https://github.com/Xinyuan-LilyGo/TTGO-LoRa-Series>)

La figura seguente riporta i dettagli del dispositivo utilizzato per il test.

La procedura seguente fa riferimento ovviamente allo specifico dispositivo indicato per ovvi motivi di disponibilità dell'HW su cui effettuare il test.

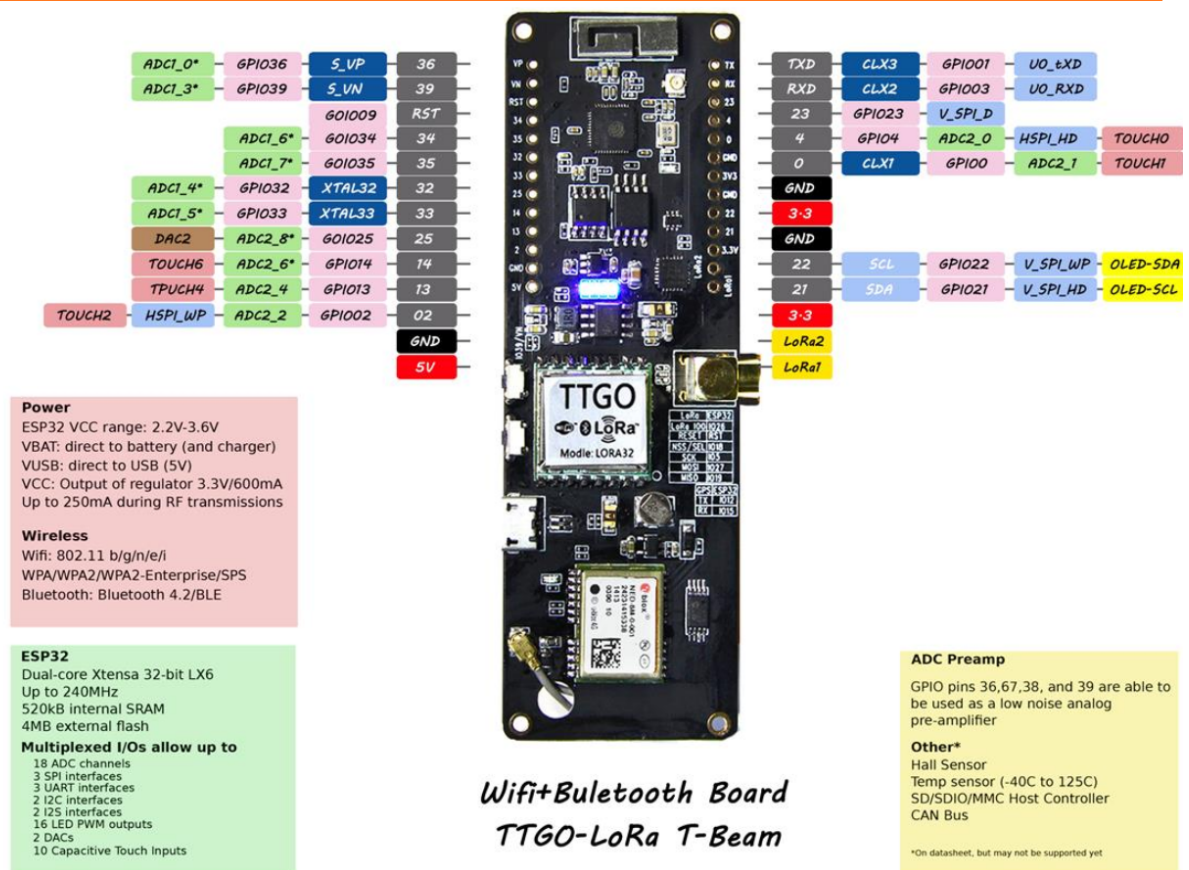


Figura 54 Dispositivo TTGO T-Beam V1 utilizzato per il test

Questo schedino non è in grado di supportare la memoria FRAM presente nel progetto LoRa Beacon, per cui non sono supportate tutte le funzioni SW che richiedono tale componente, con esclusione della sola parte di configurazione del dispositivo.

Per conservare la configurazione del dispositivo viene sfruttata, al posto delle FRAM, una porzione della memoria Flash; dal punto di vista funzionale ci sono solo minime differenze rispetto al caso di utilizzo della FRAM.

Come chipset LoRa questo schedino supporta solo (nella versione testata) chips di prima generazione LoRa quindi con potenza max di uscita di circa 100mWatt, con sensibilità della parte ricevente radio non migliorata e senza supporto di TCXO per la stabilizzazione della frequenza emessa.

Come display l'unico display supportato è quello monocromatico da 0.96" con interfaccia I2C.

La figura a fianco illustra i collegamenti del display allo schedino.

Non sono supportate le funzionalità dei leds rosso e verde ma è possibile mappare gli stessi sul led blu presente sui dispositivi.

A livello di funzionalità di debug non sono supportate tutte le funzioni legate alla FRAM, ad es. la tenuta dei log e quindi degli eventi di riconfigurazione e reboot e la registrazione degli spots.

Per l'installazione del SW tramite il download tool presentato nei capitoli precedenti si applicano esattamente le stesse considerazioni con l'unica differenza che è necessario impostare diversamente l'interfaccia seriale su cui si presenta il dispositivo una volta collegato al PC , e gli indirizzi a cui caricare i diversi moduli che costituiscono l'immagine SW.

La figura riporta le impostazioni da utilizzare:

L'immagine SW è distribuita sotto forma di archivio .zip che contiene i 5 moduli SW da caricare sul dispositivo. Tali moduli vanno caricati seguendo l'esempio della figura precedente.

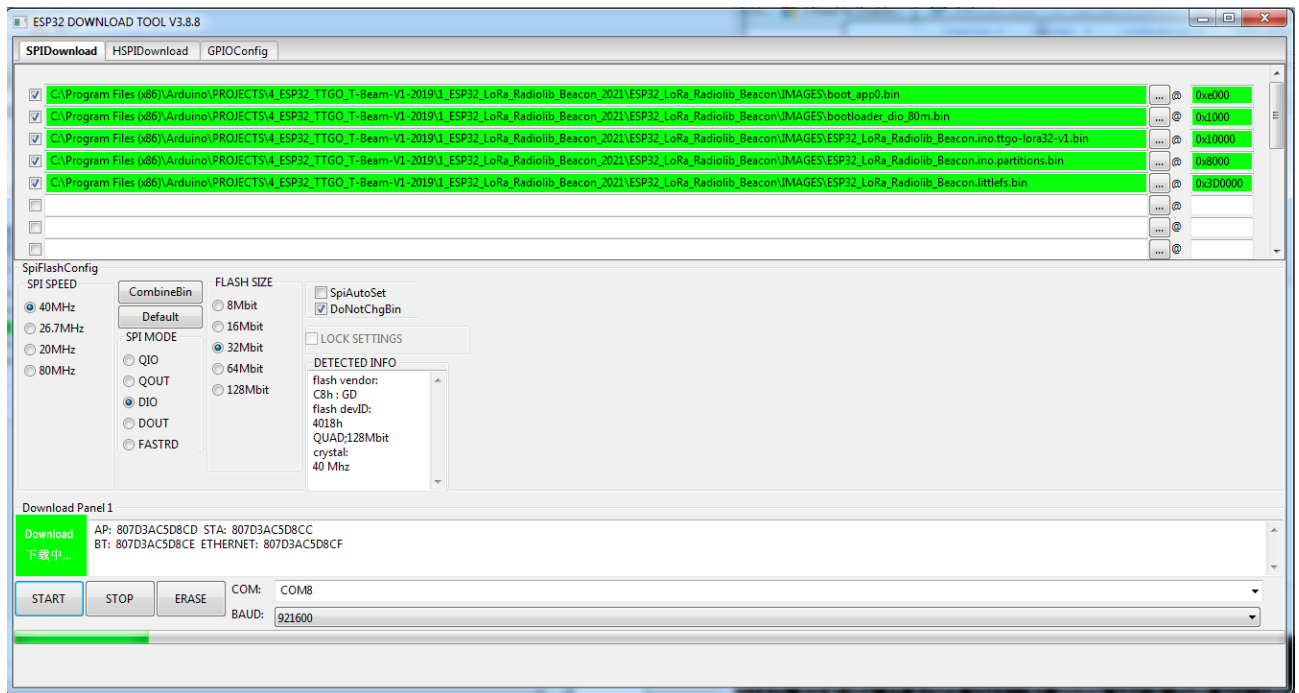


Figura 56 Settaggi da usare per il download del SW su TTGO T-Beam V1

La figura seguente riporta la schermata delle caratteristiche generali di questa versione come appaiono collegandosi alla interfaccia grafica del dispositivo a valle dell'installazione del SW LoRa Beacon.

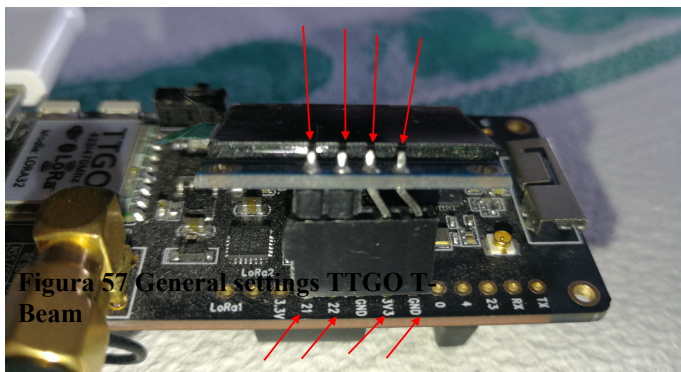


Figura 55 Collegamento Display I2C

La figura a seguire è un esempio del contenuto del display.

Si noti la presenza del campo “**ERR**” che riporta l'errore di allineamento di frequenza tra RT/TX dovuto alla mancanza di TCXO su questo tipo di HW; tale offset può essere corretto tramite il campo “ppm” presente nella schermata di setup della sezione LoRa.



Figura 58 Esempio Display TTGO T-Beam

Questa procedura di caricamento è richiesta esclusivamente per il primo caricamento del SW sullo schedino; per i successivi aggiornamenti SW è possibile utilizzare in alternativa il caricamento tramite OTA , quindi via radio, senza richiedere il collegamento USB con il dispositivo fisico.

5.2 Installazione SW su dispositivo Heltec_wifi_lora32

Questo schedino è molto simile al dispositivo di cui al paragrafo precedente, ma manca del modulino GPS; un possibile utilizzo è come iGate non essendo indispensabile in questa applicazione disporre della funzionalità di GPS.

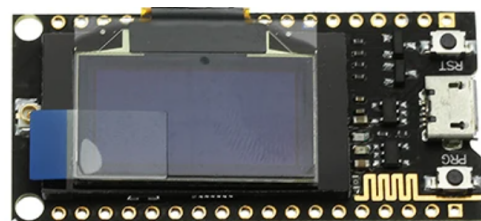


Figura 59 Heltec WiFi LoRa 32

Per il resto condivide quasi tutte le limitazioni di cui al paragrafo precedente.

La figura a seguire ritrae lo schedino in questione; esiste uno schedino molto simile reperibile con il nome commerciale TTGO WiFi Lora 32 che è praticamente identico a quello mostrato

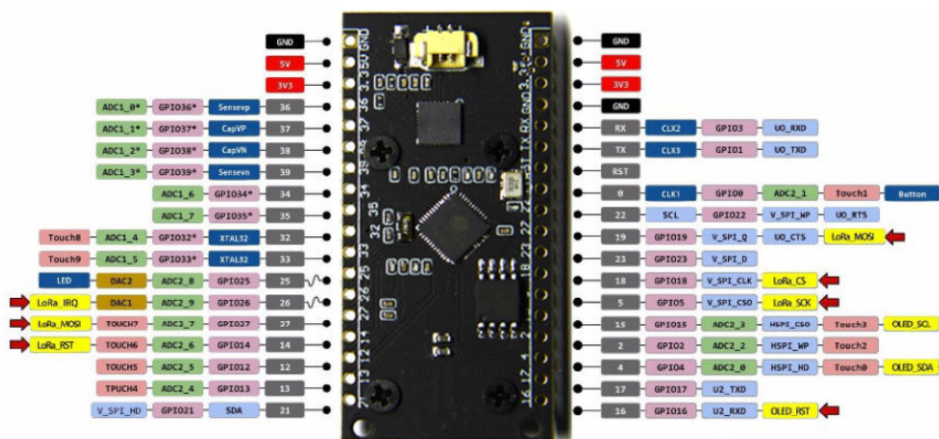


Figura 60 Pin Layout Heltec WiFi LoRa 32

La figura successiva rappresenta il pinout del dispositivo utilizzato per il test.

La figura seguente riporta le impostazioni da utilizzare per il tool di caricamento del Sw:

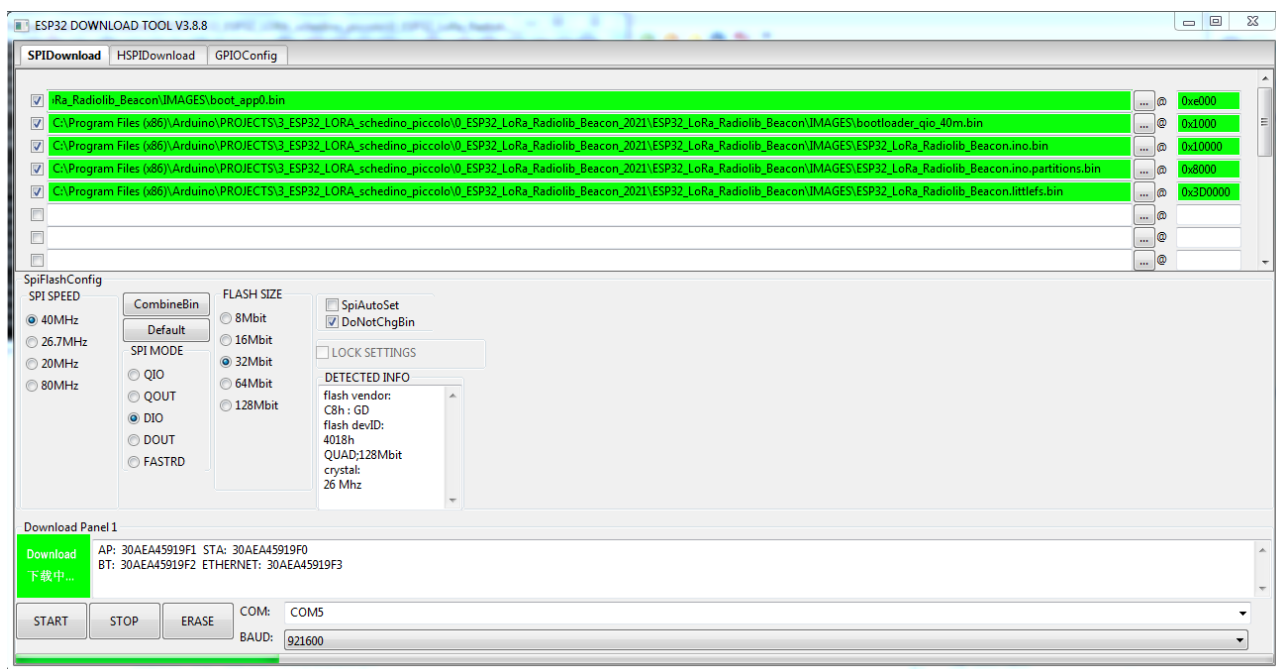


Figura 61 Settings Tool di download Sw per schedino Heltec WiFi LoRa 32

<
General Settings

Name of Device	Heltec LoRa Tracker
Device Id	ESP32-30AEA45919F0
CPU_Type	Heltec WiFi LoRa32
SW_Revision	1.0.5_20210730_1611

SAVE

La figura a fianco mostra la schermata delle proprietà generali di questa versione.

Figura 62 General Settings Heltec WiFi LoRa 32

6 Installazione SW via OTA (Over-The-Air)

Il SW LoRa Beacon fornisce una modalità per l'aggiornamento del SW senza richiedere alcun collegamento fisico al dispositivo target.

Questa modalità, chiamata OTA (Over-The-Air), sfrutta la funzionalità di WiFi Station di cui è dotato il SW per consentire, a meno del primissimo caricamento di una immagine sul dispositivo HW di destinazione, di aggiornare la versione SW presente sul dispositivo.

La disponibilità di questa funzione consente per esempio di aggiornare il SW di un dispositivo remoto senza bisogno di essere on-site.

L'aggiornamento del SW non impatta i dati di configurazione che quindi vengono preservati a cavallo di un aggiornamento SW. Resta comunque da verificare ad ogni aggiornamento eventuali

Per poter realizzare quindi l'aggiornamento del SW è necessario disporre di un unico modulo SW da scaricare su un computer dotato di un browser della famiglia chrome o firefox .

Il computer che si intende utilizzare per l'aggiornamento SW deve essere in grado di raggiungere il dispositivo da aggiornare tramite WiFi (ed eventualmente tramite una connessione internet): quindi il dispositivo da aggiornare deve essere collegato a sua volta ad un access point da cui sia possibile raggiungere (anche tramite internet) il PC che si intende utilizzare per l'aggiornamento SW.

Quindi il primo passo per poter realizzare l'aggiornamento SW è verificare la connettività tra PC e dispositivo da aggiornare sfruttando per es. il classico strumento “ping” disponibile su qualsiasi PC.

Una volta verificata la connettività, è necessario porre il dispositivo da aggiornare in una condizione operativa particolare detta “Admin Mode” che è richiesta allo scopo di disabilitare temporaneamente alcune funzioni non necessarie nella fase di aggiornamento SW.

A tale scopo è possibile accedere alla schermata “OPERATION MODE SETTINGS” e selezionare la casella Admin_Mode salvando quindi la pagina.

Il dispositivo remoto effettuerà a questo punto un reboot e si ripresenterà con la modalità Admin_Mode selezionata.

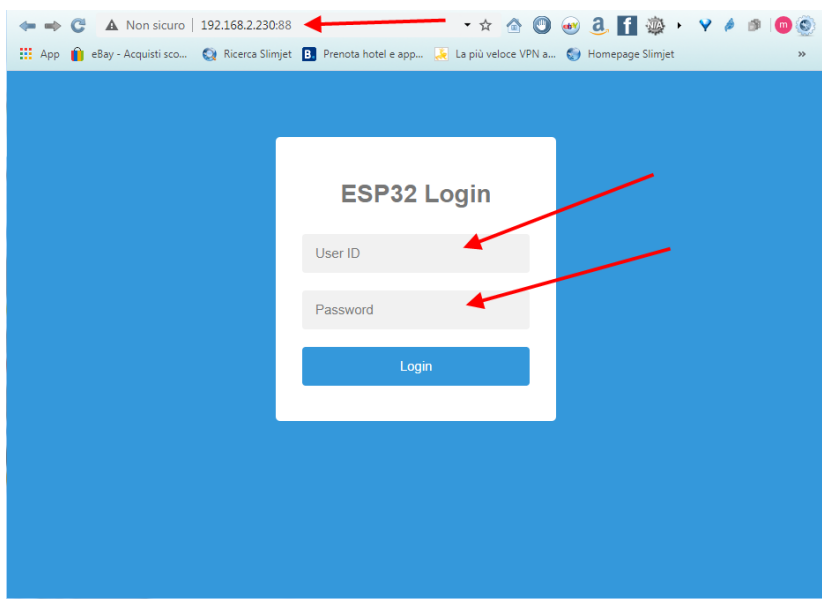
Va osservato che il reboot automatico è disponibile esclusivamente utilizzando dispositivi basati sulla piattaforma HW LoRa Beacon; in caso di piattaforme diverse sarà necessario far ripartire manualmente il dispositivo remoto (e quindi sarà necessaria una persona vicina al dispositivo o un altro meccanismo per far ripartire il dispositivo in maniera remota ad es. cliccando l'alimentazione).

In queste condizioni il dispositivo apparirà temporaneamente non funzionante a livello LoRa in quanto tale funzionalità sarà, come spiegato , temporaneamente non disponibile.

L'attivazione della modalità Admin_Mode farà apparire una nuova interfaccia grafica dedicata all'aggiornamento SW: per accedere a tale interfaccia bisognerà puntare con una altra finestra del browser all'indirizzo del dispositivo remoto, usando un numero di porta pari a 88 come nell'esempio della figura a fianco.

A questo punto andranno inserite come credenziali (di default) i seguenti valori:

- User Id: admin
- Password: adminota



Apparirà una nuova schermata che inviterà a selezionare dal proprio PC la nuova immagine SW di aggiornamento: selezionare il file ricevuto come aggiornamento, il cui nome terminerà tipicamente in .bin.

Avviare quindi il caricamento del pacchetto SW ed attendere che l'operazione si completi....

Figura 63 Interfaccia per caricamento SW via OTA

Terminata la fase di caricamento della nuova immagine

di aggiornamento il dispositivo remoto ripartirà e si ripresenterà con la sua interfaccia grafica, in accordo al nuovo SW caricato; accedendo alla pagina “General Configuration” si potrà verificare che la versione SW del dispositivo sia quella attesa come effetto del caricamento effettuato.

A questo punto il dispositivo è aggiornato per cui è possibile uscire dal “Admin_Mode” tramite la pagina di “Operation_Mode_settings” ; il dispositivo effettuerà un nuovo reboot e si presenterà nella modalità di funzionamento selezionata.

A valle di ogni aggiornamento SW è bene controllare che la configurazione del dispositivo sia corretta in quanto a seconda dei cambi introdotti nella nuova versione SW potrebbero essere necessari dei piccoli aggiustamenti della configurazione; tali aggiustamenti dovrebbero essere eventualmente documentati nella “Release Note” che dovrebbe accompagnare ogni nuovo rilascio SW.

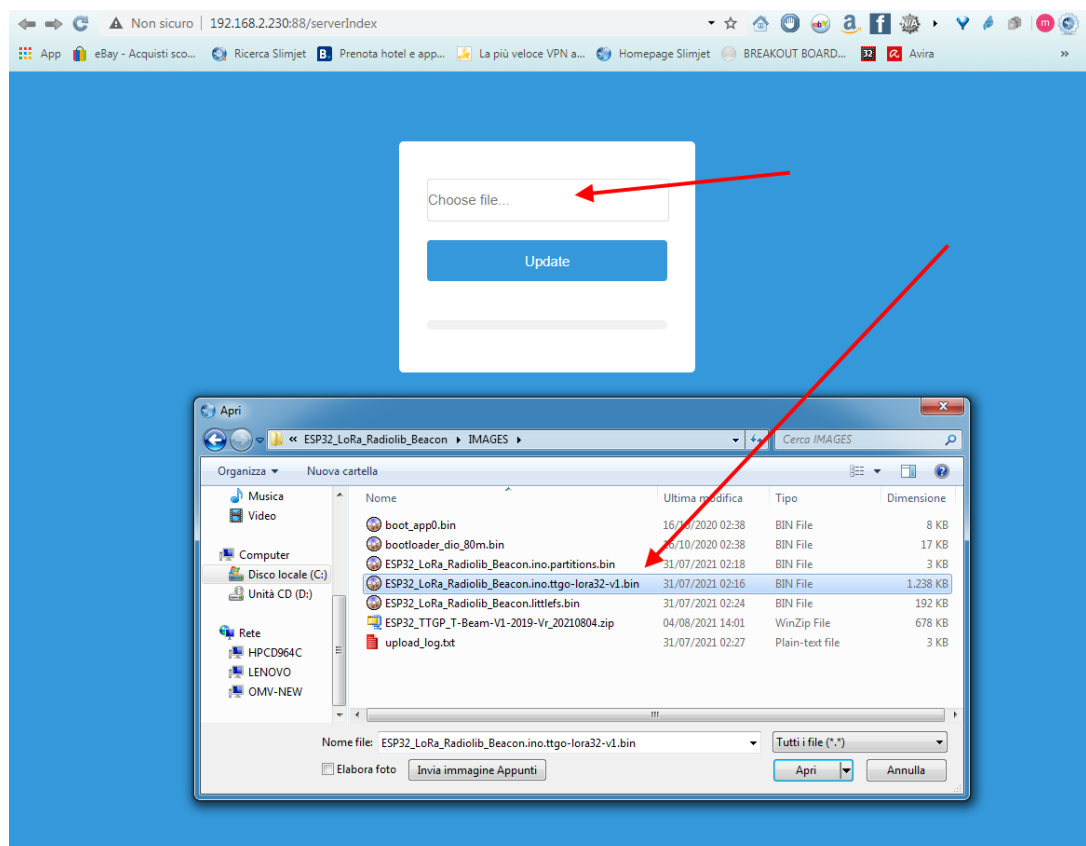


Figura 64 selezione immagine SW da caricare dal proprio PC

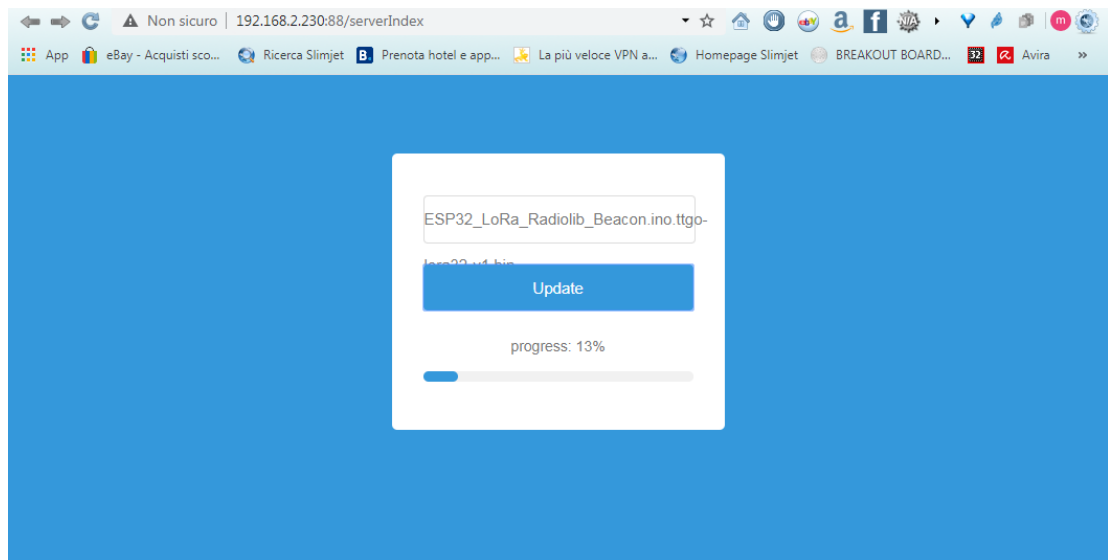


Figura 65 Display durante la fase di caricamento del nuovo SW

7 Salvataggio e Caricamento della configurazione via OTA

Il SW LoRa Beacon è stato concepito per essere utilizzato agevolmente per fare della sperimentazione basata sul protocollo radio LoRa per cui più che mirare ad un esercizio di sviluppo SW è mirato ad un utilizzo strumentale ai fini della valutazione della tecnologia LoRa in impegni non standard (secondo gli obiettivi per cui è nata), ovvero per servizi tipici del mondo dei radioamatori.



Figura 66 Schermata di Save/Restore della configurazine di un dispositivo LoRa Beacon

Quindi l’obiettivo è stato quello di fornire la possibilità di modificare agevolmente una serie di parametri di funzionamento senza dover necessariamente avere esperienza di sviluppo SW.

Il modo scelto è stato quindi di parametrizzare tutta una serie di elementi funzionali rendendoli modificabili tramite una opportuna interfaccia grafica; l’insieme dei parametri che caratterizzano una data impostazione di ogni dispositivo viene qui indicata con il termine “configurazione” ed è visualizzabile come un insieme di attributi e valori corrispondenti.

La configurazione di un certo dispositivo è mantenuta internamente al dispositivo in una opportuna memoria non volatile che può essere una FRAM per i dispositivi che chiamiamo LoRa Beacon HW oppure una frazione della memoria flash per quei dispositivi privi di FRAM.

Il modo canonico per gestire la configurazione di un dispositivo è tramite l’interfaccia grafica dello stesso, salvo a dare la possibilità di visualizzare, salvare su un PC esterno o caricare da un PC esterno una certa configurazione.

Il formato scelto per il file di configurazione è lo standard JSON che è agevolmente leggibile e gestibile sia con un semplice editor, che con uno dei tanti tools esistenti allo scopo.

La possibilità di salvare e ricaricare una certa configurazione consente di tenere agevolmente traccia delle condizioni di prova in cui un certo insieme di test viene effettuato; in mancanza di un tale strumento sarebbe certamente molto più complesso tenere traccia ordinata delle condizioni di prova allo scopo di poter fare dei confronti.

La scelta di un formato testuale per il file di configurazione consente di poter realizzare agevolmente il confronto di più files di configurazine per evidenziare eventuali differenze tra gli stessi.

Tutte le operazioni di save e restore della configurazione richiedono che il dispositivo sia messo nella modalità “Admin_Mode” come per le operazioni di aggiornamento SW.

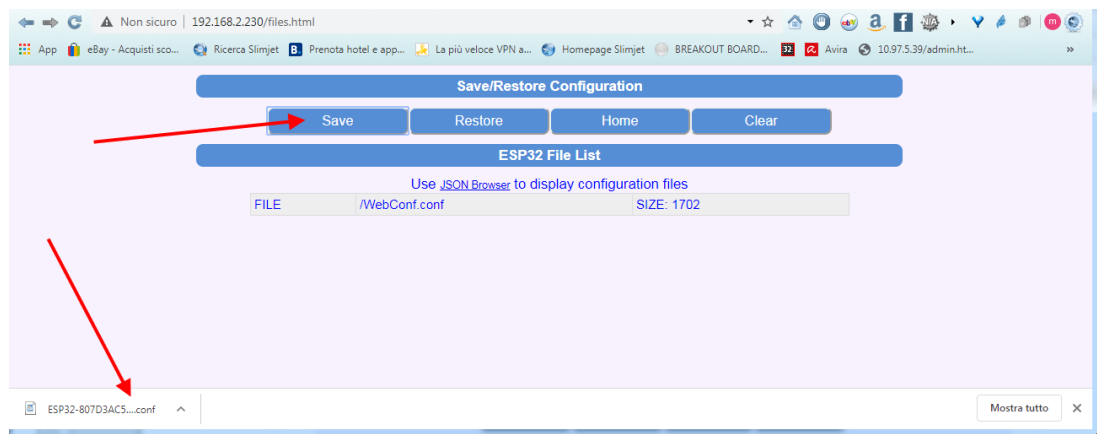
Per accedere alle funzioni di save/restore della configurazione è disponibile la pagina **“SAVE/RESTORE CONFIGURATION”**.

La figura seguente riporta il contenuto della schermata che appare: come si può notare è presente una lista di files che consente di conoscere i files di configurazione presenti nel dispositivo ed eventualmente caricati manualmente. Il File /WebConf.conf rappresenta sempre il file di configurazione attivo al momento.

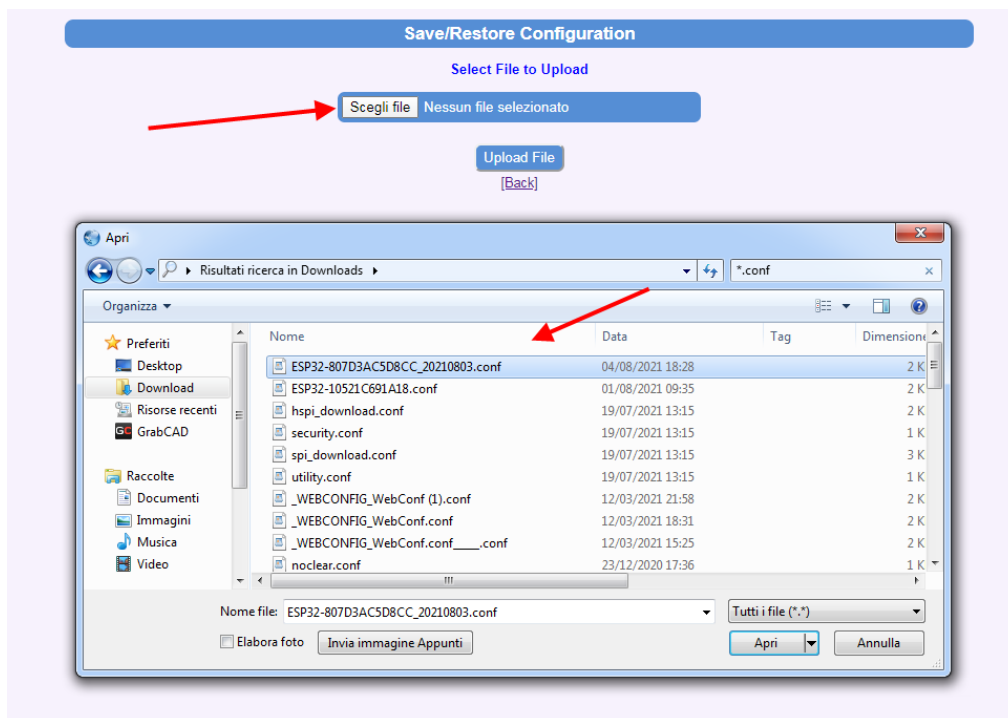
Figura 67 Save della configurazine di un dispositivo LoRa Beacon su PC

Per salvare la configurazione corrente del dispositivo è sufficiente selezionare il tasto Save: verrà creato e scaricato un file direttamente sul PC, come dalla figura seguente:

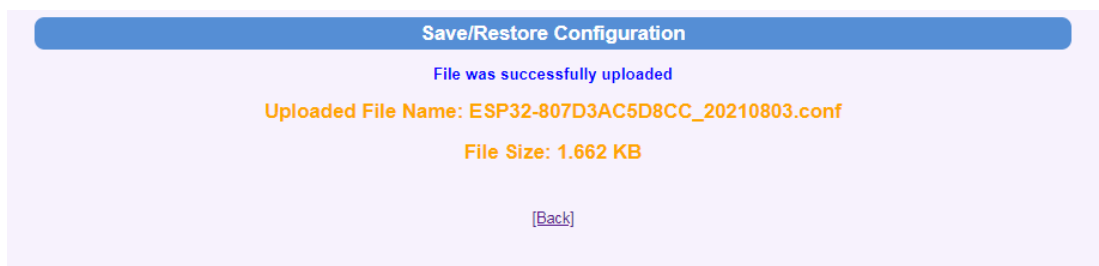
Il file scaricato avrà un nome del tipo “ESP32-<MAC address>.conf” e verrà salvato in una location dipendente dalla configurazione del browser in uso. La figura sopra è un esempio.



Per effettuare il Restore della configurazione da un certo file è sufficiente selezionare il tasto “Restore” e cercare e selezionare il file che interessa caricare; quindi selezionare il tasto Upload File.



Se il caricamento avviene con successo si avrà una schermata del tipo:



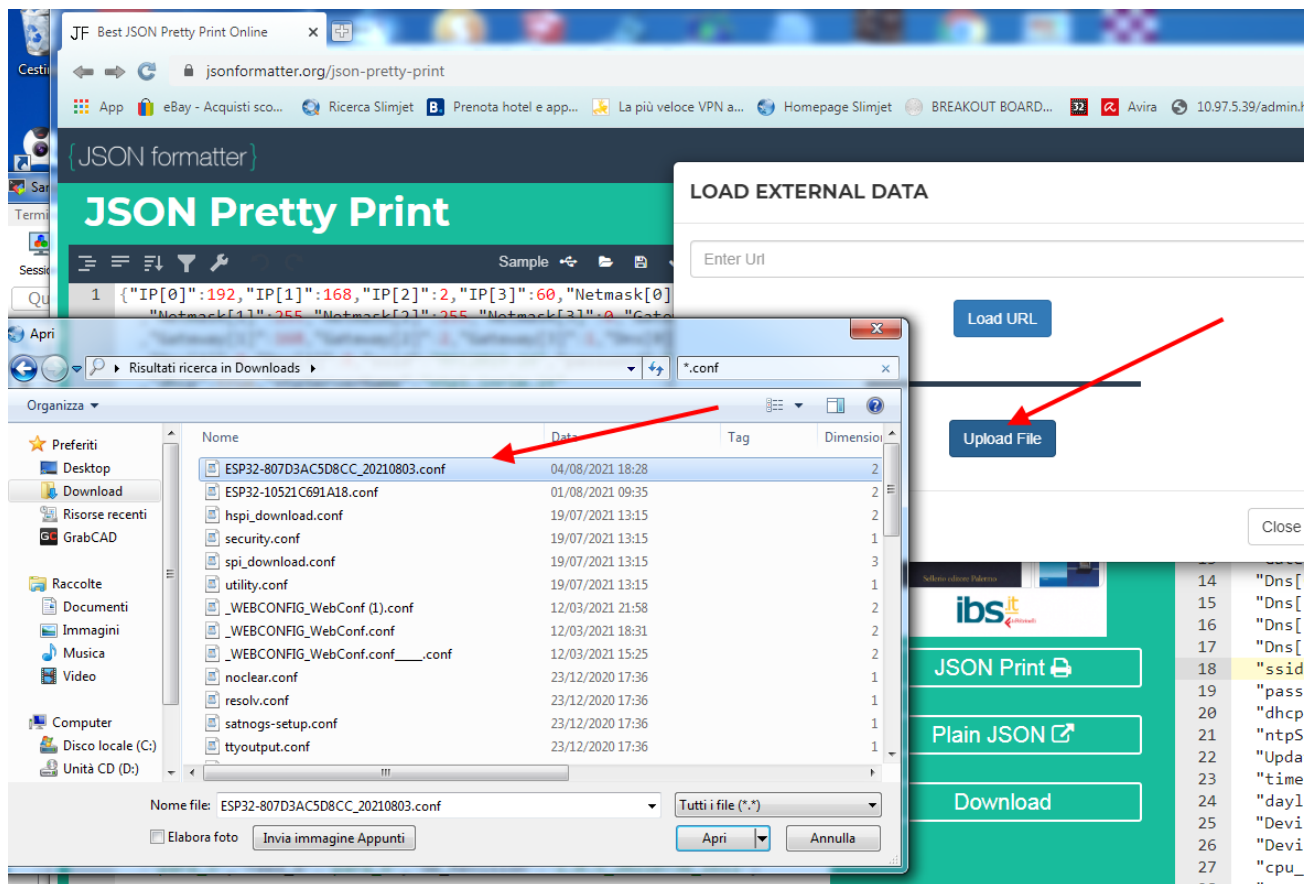
Selezionando il tasto “Back” apparirà la schermata seguente dalla quale si evidenzia la presenza del file caricato tra i files presenti sul dispositivo.



Per rendere effettiva la nuova configurazione si rende necessario uscire dalla modalità “admin_Mode” e riavviare il dispositivo.

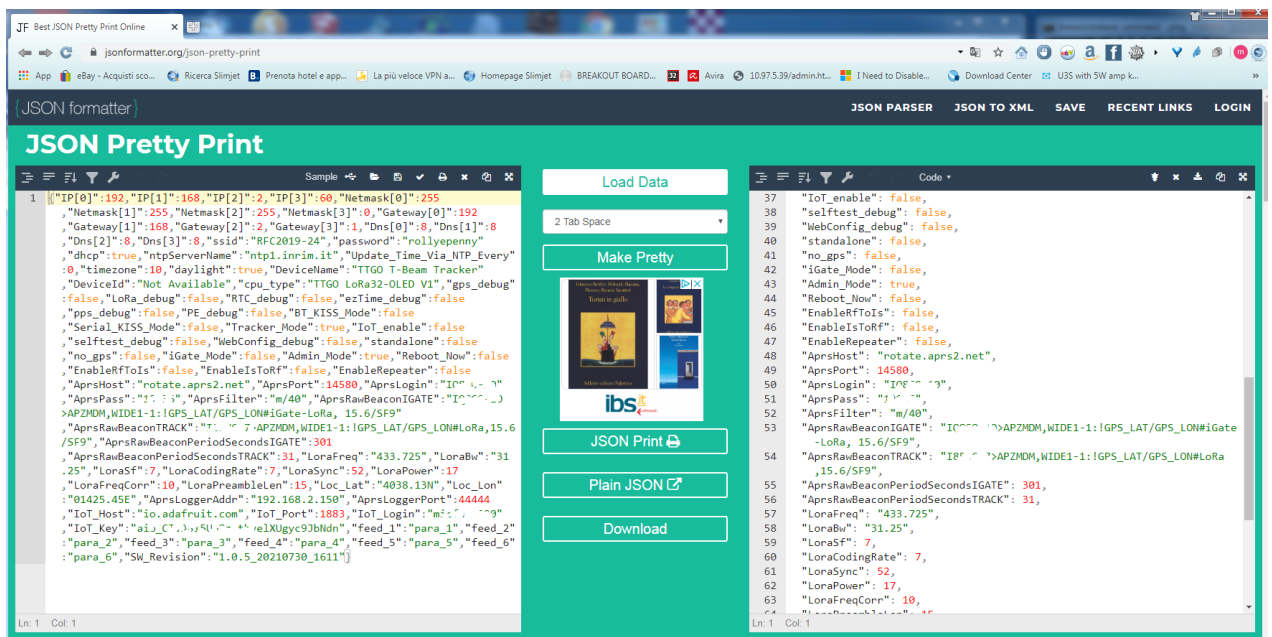
Volendo ispezionare e/o eventualmente modificare la configurazione di un dispositivo sulla base di un file di configurazione è possibile utilizzare per esempio il tool <https://jsonformatter.org/json-pretty-print> accessibile liberamente su internet tramite un qualsiasi browser tipo chrome o firefox.

La figura seguente è un esempio di utilizzo di questo tool per visualizzare in chiaro il contenuto di un file di configurazione di un certo dispositivo:



Una volta caricato il file da analizzare è possibile utilizzare il tool per modificare ed eventualmente salvare il file modificato.

La figura seguente riporta la schermata principale del tool di cui sopra.



8 Esempi di dove acquistare la componentistica

Tutta la componentistica necessaria per la costruzione di entrambe le versioni di LoRa Beacon è facilmente acquistabile sui classici portali asiatici; è importante tenere conto che per quanto generalmente è molto agevole trovare quanto necessario per le proprie esigenze, **non bisogna essere superficiali nell'acquistare componenti apparentemente identici a quelli consigliati, ma decisamente più economici**: è purtroppo una classica trappola in quanto come ben noto non esiste in pratica nessun reale tipo di garanzia che i componenti acquistati siano conformi a quanto dichiarato o anche mostrato in foto.

Il consiglio è di cercare si l'affare, ma badando molto bene a che sia realmente tale !

Come detto la totalità della componentistica è facilmente reperibile; gli unici componenti ovviamente non reperibili su tali portali sono i circuiti stampati che possono essere richiesti al seguente indirizzo info@sarimesh.net: il prezzo di tali PCB è comunque molto contenuto in quanto non gravato da particolari markup ma funzione delle sole spese di fabbricazione e spedizione.

A seguire si fornisce una lista puramente indicativa, alla data, dei componenti richiesti per il montaggio delle due versioni di LoRa Beacon: la lista è unica per le due versioni per cui sulla base del tipo di versione che si vuole montare sarà necessario eventualmente acquistare solo un sottinsieme di componenti. **Non si assume ovviamente nessuna responsabilità per eventuali errori contenuti in questa lista, nè per l'affidabilità dei venditori indicati.**

I tempi di spedizione dalla Cina sono ultimamente calati moltissimo grazie all'entrata in vigore della nuova regolamentazione sugli acquisti effettuati sui portali online; è però da considerare che tutti i prezzi che si vedono sui portali vanno poi maggiorati dal valore dell'IVA (22%) all'atto della conclusione degli ordini.

Lista di esempio per acquisto parti aggiornata al 20210805

- [Microcontrollore ESP32-WROOM-32D versione 38 pin con antenna stampata](#)
- [Display I2C 0.96"](#)
- [Display SPI IPS 1.14" 135x240 LCD Module](#)
- [FRAM FM24W256](#)
- [Moduli GPS NEO-6M NEO-7M NEO-8M](#)
- [Port Expander PCF8574](#)
- [Modulo LoRa 1 Watt SX1268 Ebyte E22-400M30S](#)
- [Modulo LoRa 100milliWatt SX1268 E22-400M22S](#)
- [Modulo LoRa 100 milliWatt SX1278 RFM98W](#)
- [Buzzer KY-012](#)
- [Connettore USB Verticale per PCB](#)
- [Assortimento di diodi](#)
- [transistor 2N2222](#)
- [Assortimento LEDs 5 mm rosso e verde](#)
- [MCP1700](#)
- [Tastino 6x6x12 verticale](#)

-
- [Strisce contatti 2.54 mm altezza 7.1 mm femmina](#)
 - [strisce pin 2.54 mm maschio angolo retto tipo R1](#)
 - [DS3231 RTC](#)
 - [Cavetto IPX-SMA 15 cm](#)
 - [Strisce pin 2.54 mm maschio diritto](#)
 - [Assortimento condensatori ceramici](#)
 - [Assortimento resistenze](#)
 - [Zoccoletto DIP 16 pin](#)
 - [Antennina con calamita e attacco SMA 433 Mhz](#)
 - [Cavo prolunga USB](#)
 - [Connettore alimentazione 12V](#)
 - [Antenna GPS esterna attiva](#)
 - [DC/DC Step down conv.](#)
 - [Assortimento condensatori elettrolitici](#)