## meshtastic_meshmonitor

E' il progetto at https://github.com/vinloren/meshtastic_meshmonitor che si prefigge lo scopo di controllare la rete Lora Meshtastic visibile dal nostro punto di osservazione attraverso un nodo Tlora d'interfaccia radio collegato alla porta seriale del computer Linux o Windows che lo ospita.

Per 'controllo' si intende visualizzazione della mappa geografica sulla quale appariranno i riferimenti ai nodi coi quali siamo in contatto, lo scambio di messaggi testuali, la tracciatura dei percorsi fatti dai nodi mobili, la gestione dell'abilitazione dei nodi ad apparire in mappa oltre ad altre funzioni di manutenzione del sistema. Il tutto gira su un server Python Flask locale con al centro un Db Sqlite3 per il salvataggio e il controllo dei dati. Il Db è l'elemento chiave che permette l'interattività all'interno della rete cui si aggiunge questa nuova funzione di ricezione via mail di testi da inviare in rete sul canale primario ch0.

## meshtastic_meshcontroller

E' l'applicazione python all'interno del progetto meshtastic_meshmonitor che si occupa di controllare tutti i messaggi di protocollo e testuali ricevuti memorizzandone i contenuti sul Db Sqlite3 per caratterizzare ciascun nodo in vista con data/ora e posizione geografica oltre ad eventuali dettagli di telemetria presenti. Questa applicazione si occupa anche di gestire i messaggi testuali ricevuti fornendo risposte immediate in caso di richiesta qsl?. o anche inviare su ch0 messaggi che attraverso il server Flask abbiamo inserito nella relativa tabella per l'invio ad opera di mesh_controller.

## Ricezione Mail

Sfruttando il meccanismo della trasmissione dei messaggi testuali su accennato, diventa possibile fare in modo che un utente non presente presso il proprio sistema meshmonitor sia in grado di inviare messaggi in rete come se fosse davanti alla console. Il meccanismo di trasmissione messaggi è il seguente:

1. ad ogni fine ciclo di scansione di un qualunque messaggio di protocollo da gestire, mesh_controller va a leggere in Db la tabella 'messaggi' per vedere se l'ultimo presente inizia col carattere ^ che è l'identificatore di messaggio da trasmettere.
2. la presenza di un simile messaggio può essere dovuta al fatto di averlo ricevuto da un nodo in rete che ce lo ha trasmesso oppure può essere stato invece inserito da server Flask su sollecitazione dell'operatore.
3. nel primo caso mesh_controller invia il messaggio privato del carattere ^ sul ch0 come risposta tipo pappagallo, nel secondo come messaggio intenzionalmente inviato dall'operatore. In poche parole, indifferentemente da chi ha innescato il processo, la presenza in tabella 'messaggi' di un testo che inizia con ^ provoca la sua trasmissione in rete su ch0.

Stante questa premessa diventa facile capire come si possa ottenere la trasmissione in rete di un messaggio che il sistema ha ricevuto dall'esterno via Email.

## Alice_ReadMail.py

E' l'applicazione python che si occupa di ascoltare la propria casella mail ogni 5 minuti. Il provider di mail prescelto per questa applicazione è alice.mail perché è uno di quelli sui quali sono registrato e

sono riuscito a far funzionare. Altro provider su cui posso fare affidamento è gmail ma questo richiede una configurazione su Google di una password specifica (token) per accesso da programma e, sebbene l'abbia provata e fatta funzionare, è una procedura piuttosto complicata da spiegare in un documento: lascio quindi il compito a ciascuno che voglia cimentarsi nell'impresa.

Il problema che si pone è che più o meno ogni provider ha le sue specifiche regole che sono da rispettare per veder garantito accesso al mail, e ciò induce un lavoro di adattamento specifico che per alice.mail è stato eseguito. Credo perciò che la soluzione più pratica per chi volesse introdurre questa funzione in meshtastic_meshmonitor sia quella di registrare un account su alice.mail.

### come funziona questa applicazione

una volta lanciata, l'applicazione viene rilanciata automaticamente ogni 5 minuti dal time scheduler incluso. Viene fatto login al mailer leggendo le credenziali dal file '.env' (senza apici) residente nella directory ./source insieme col resto dei programmi python. .env è un file di testo che contiene 2 righe:

1. EMAIL_USER='tuo_username@'alice.it'
2. EMAIL_PASS='tuapassword'

quindi vengono scandite le ultime 5 mail ricevute cercando al loro interno il Subject = /^ che identifica mail contenente testo da inviare in rete su ch0.

Se c'è un riscontro positivo l'applicazione ne preleva il testo (troncando se necessario a max 188 chars) e lo va a scrivere nel Db in tabella 'messaggi' apponendovi in testa il carattere ^ per informare mesh_controller che questo è un messaggio da inviare in rete su ch0.

L'applicazione si appoggia al log ../logs/alice_readmail.log dove trovare indicazioni sui messaggi trasmessi e su eventuali errori occorsi.

## ReadMail_alice.py o ReadMail_gmail.py

Chi avesse un account google gmail può usare l'applicazione alternativa presente in ./source che funziona allo stesso modo di Alice_ReadMail.py. Il problema da superare in questo caso risiede nel fatto che differentemente da Alice mail Google gmail non permette l'uso della stessa password usata sotto Thunderbird o Outlook o Posta etc. ma richiede uno specifico token se l'accesso non è operato da questi. Per ottenere questo token, che sarà la password inserita dall'applicazione python, occorre andare sul proprio account google per generarlo. Questa cosa la feci un anno fa e il token lo sto usando senza problemi ma non ricordo più la procedura per ottenerlo; credo tuttavia che oggi questo non sia più un problema grazie a ChatGpt.

### .env da configurare

Come già visto per Alice_ReadMail.py occorre aggiungere le credenziali necessarie all'accesso:

1. GMAIL_USER='iltuouser@gmail.com'
2. GMAIL_PASS='iltoken_ottenuto_su_google'

Questa applicazione funziona in modo simile all'altra ma usa imap_tools in aggiunta a imaps il che rende più snella la programmazione. La peculiarità rispetto a Alice_ReadMail.py sta nel fatto che si possono ricevere le mail non ancora lette invece che le ultime 5 e poi scandendone l'elenco trovare la

presenza o meno di mail con subj = /^. e interrompere la ricerca alla prima trovata. In questo modo potremmo anche, avendo inviato 10 messaggi con subject = /^, trovarli spediti uno alla volta ogni 5 minuti dato che ad ogni passaggio il mail/messaggio precedente non viene più presentato perchè già letto. Questo sistema può essere un sistema per inviare messaggi a ripetizione come avveniva con l'applicazione broadcast_msg_pyqt5.py sotto 'invio messaggi periodici'.

## Trasmissione messaggi da ch0 a Mail (Gmail o Alice)

Può risultare utile ricevere sulla propria casella Email e sul repository ho previsto questa opzione per chi ha account gmail e ha la app password per accesso a gmail da app oppure per chi ha account Alice mail. La cosa funziona così:

1. in derectory ./source abbiamo due file: alice_mailout.py e gmail_mailout.py. Dovendo scegliere se usare gmail o alice mail per inviare messaggi di Alert al nostro account di posta, facciamo copia su mailout.py della versione scelta (cp se linux o copy se windows alice_mailout.py (o gmail_mailout.py) mailout.py).
2. ogni volta che mesh_controller intercetta un messaggio testuale contenente la parola Alert allora provvede a inviare il testo ricevuto, arricchito da indicazione del longname di chi lo ha inviato e da rssi/snr, verso il destinatario indicato nel file .env e come mittente il proprio account di posta.
3. a supporto di questa opzione è necessario aggiungere nel file .env che già conteneva le credenziali di lettura mail: MITTENTE='miomailaccount' , DESTINATARIO='miomailaccountl' di destinazione (che può coincidere col mittente), MAILOUT_PASS='password_di_accesso' ovvero token app password se usciamo su gmail.

Con questo sistema, in aggiunta alla pubblicazione su ch0 di messaggi inviati via mail, si può ottenere uno scambio di messaggi fra utenti del mesh anche non essendo presenti davanti alla propria installazione.