

Progetto di Ingegneria Informatica  
*GPSDataLogger and Parser* (Relazione)

Autore: Luca Padalino  
Matricola: 935033  
Codice Persona: 10695959

a.a. 2021/2022

Tutor: prof. Mirko Reguzzoni

# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Obiettivo . . . . .	1
1.2	Destinazione d'Uso . . . . .	1
<b>2</b>	<b>Specifica</b>	<b>1</b>
<b>3</b>	<b>Implementazione</b>	<b>2</b>
3.1	Sviluppo del Software . . . . .	2
3.1.1	Ricerca dei Dispositivi u-blox . . . . .	3
3.1.2	Acquisizione dello Stream . . . . .	3
3.1.3	Elaborazione dei files RINEX . . . . .	7
3.1.4	Funzionalità Aggiuntive . . . . .	8
3.2	Fasi dello Sviluppo . . . . .	8
3.3	Test e Difficoltà Riscontrate . . . . .	8
<b>4</b>	<b>Conclusione</b>	<b>9</b>

# 1 Introduzione

## 1.1 Obiettivo

L'obiettivo del progetto è realizzare uno strumento in grado di raccogliere, in tempo reale, stream di dati in formato binario provenienti da uno o più ricevitori u-blox EVK-M8T. Ognuno di questi connesso ad una porta COM. Lo strumento deve essere in grado di registrare il flusso, decriptarlo e convertirlo in formati dati standard come NMEA, RINEX e costruire files di sincronizzazione tra il tempo del computer e il tempo GNSS (sistema satellitare globale di navigazione).

## 1.2 Destinazione d'Uso

Il progetto ha come scopo la realizzazione di un software che verrà poi utilizzato dall'Università degli Studi di Ferrara, precisamente a bordo di un autogiro, impiegato in attività di rilievo geodetico e fotogrammetrico di supporto a misure di radioattività ambientale.

# 2 Specifica

Dopo la fase di analisi dei requisiti, condotta con il supporto dei tutor che, oltre a supervisionare il lavoro sono stati il tramite del committente (Università di Ferrara), sono state definite le funzionalità base del software, di seguito riassunte:

- acquisizione degli stream dai ricevitori
- estrazione delle stringhe NMEA e salvataggio dell'intero flusso
- generazione dei files RINEX
- estrazione di informazioni temporali dai messaggi provenienti, al fine di generare files di sincronizzazione tra i time-stamp del ricevitore e il computer locale

A completamento della specifica sono state aggiunte funzionalità come:

1. indicare un percorso in cui salvare i files acquisiti e generati dal software;
2. scegliere da quali sistemi satellitari globali di navigazione (GNSS) ricevere i dati;
3. caricare registrazioni nel formato `.ubx` già effettuate con software di terze parti (es. u-center);
4. scegliere la versione e la tipologia del file RINEX da generare;

5. possibilità di includere il numero della settimana nel file relativo alle sincronizzazioni dei tempi tra GNSS e computer locale (in caso di possibili variazioni dovuto a rilevazioni notturne)

### 3 Implementazione

La soluzione proposta consiste in un software sviluppato in Python, comprensivo di una GUI.

La scelta del linguaggio Python deriva dalla volontà di realizzare un applicativo cross-platform, con strutture dati già implementate (così da evitare puntatori e quindi la creazione di strutture dati personalizzate, difficili da espandere in futuro) orientato agli oggetti e che supportasse la programmazione concorrente al fine di parallelizzare le operazioni di raccolta dati.

La GUI è stata sviluppata per rendere il programma più semplice e intuitivo possibile e compatibile con qualsiasi destinazione d'uso: dal mouse e la tastiera al solo display touch-screen.

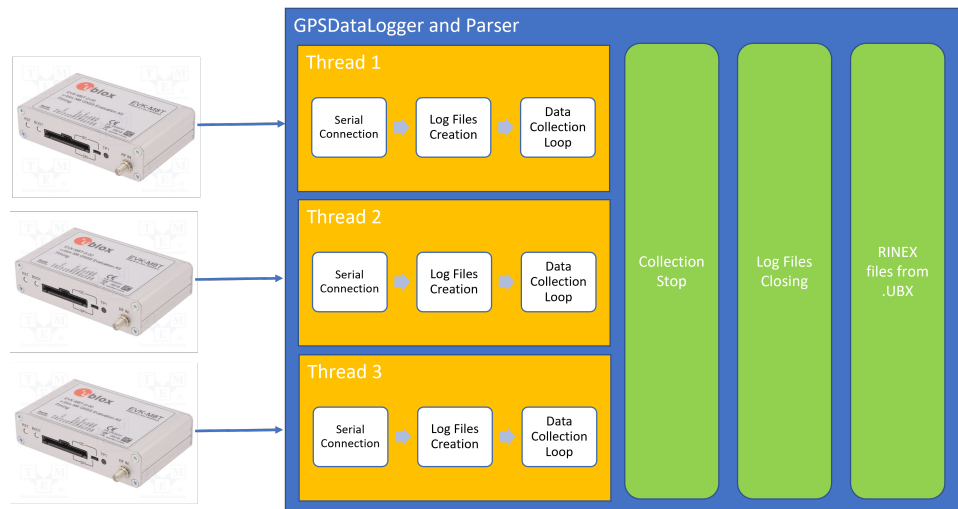


Figura 1: Sintesi del funzionamento del programma

#### 3.1 Sviluppo del Software

Nell'ottica del riuso del codice e della scrittura di un software versatile e facilmente personalizzabile, lo sviluppo è stato suddiviso in moduli, ciascuno dei quali associato ad una classe di codice.

Dopo la fase di studio dei requisiti e di discussione della soluzione proposta con i tutor del progetto, il primo oggetto realizzato è stato un prototipo della GUI, successivamente implementato in Python.

`main.py` è il punto di partenza dello sviluppo del codice. Esso si occupa di caricare le risorse relative alla GUI e contiene tutte le procedure che esulano dall'acquisizione ed elaborazione delle informazioni come la scelta del percorso di lavoro, la scelta dei GNSS da cui ricevere informazioni e la ricerca dei dispositivi u-blox connessi.

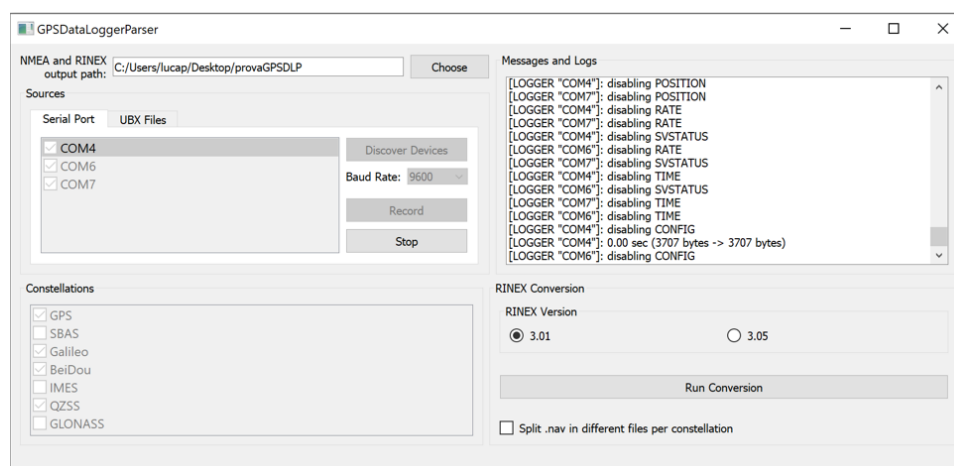


Figura 2: GUI di *GPSData Logger and Parser*

### 3.1.1 Ricerca dei Dispositivi u-blox

Questa operazione consiste nella scansione di tutte le periferiche connesse alle porte COM del computer. A ciascuna periferica viene inviato un messaggio `UBX-RXM-RAWX` a cui il ricevitore, se effettivamente è un dispositivo u-blox, risponde con un messaggio `UBX-ACK-ACK`. Le periferiche da cui si riceve un `ACK` sono quindi ricevitori, le altre sono escluse.

### 3.1.2 Acquisizione dello Stream

L'acquisizione dello stream è l'operazione più importante che il programma svolge. Non appena l'utente richiede di registrare il flusso dei ricevitori selezionati, per ciascun ricevitore viene creato un oggetto `BackendHandler`, sviluppato successivamente a `main.py`.

`BackendHandler` si occupa di istanziare tutti gli elementi necessari al raccoglimento e all'elaborazione delle informazioni. Utilizza infatti istanze dei seguenti oggetti:

- `SerialParser`, sviluppato allo scopo di gestire le connessioni, la raccolta e l'invio di dati dalla/alla periferica e la gestione di eventuali eccezioni;

- **Logger**, contenente tutte le funzioni per codificare e decodificare i messaggi, siano esse stringhe NMEA, messaggi UBX-RXM-RAWX o comandi per configurare il ricevitore e la funzione *logData*.

*logData* è una delle funzioni più importanti, in quanto è quella che si occupa di collezionare i dati dopo aver aperto la connessione e configurato il ricevitore. La configurazione del ricevitore consiste nelle seguenti operazioni, ripetute un massimo di 3 volte qualora il tentativo di eseguire il comando di configurazione non dovesse avere esito positivo nell'immediato:

1. salvataggio della configurazione precedente, al fine di ripristinarla per usi al di fuori di GPSTDataLogger and Parser;
2. impostazione del *rate* con cui devono viaggiare i messaggi (1Hz);
3. configurazione dei GNSS da cui ricevere messaggi;
4. configurazione dei messaggi UBX che si desidera ricevere: UBX-RXM-RAWX, indispensabili per generare il file RINEX relativo alle osservazioni; UBX-RXM-SFRBX, indispensabili per generare il file RINEX relativo alle navigazioni.
5. configurazione dei messaggi NMEA che si desidera ricevere: dai colloqui con i committenti è emerso che il software deve raccogliere soltanto messaggi di tipo GGA, quindi tutti gli altri in fase di configurazione sono stati disabilitati.

Dopo la configurazione, vengono creati nella directory indicata i files in cui andare a salvare i dati provenienti dal ricevitore, opportunamente elaborati:

- **PORTA\_rover.ubx**: contiene il flusso ricevuto dalla periferica. Può essere utilizzato come backup dei dati raccolti per elaborazioni future, ed è necessario per generare i files RINEX al termine della raccolta;
- **PORTA\_NMEA.txt**: contiene tutte le stringhe NMEA ricevute dalla periferica. Isolare le stringhe NMEA è una delle prime operazioni di parsing dello stream: tutte le stringhe NMEA sono introdotte da un "\$" e terminano con il carattere speciale di ritorno a capo "CRLF". Man mano che lo stream viene analizzato, vengono campionate le sentences NMEA e scritte in questo file;

NMEA messages sent by the GNSS receiver are based on NMEA 0183 Version 4.10. The following figure shows the structure of a NMEA protocol message.

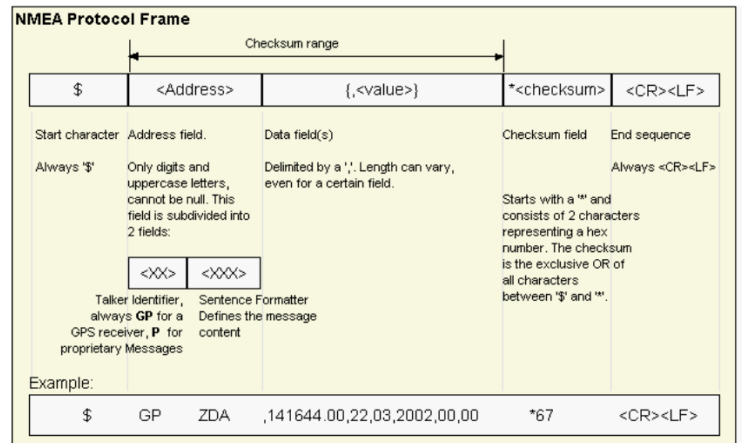


Figura 3: Struttura di una stringa NMEA

- **PORTA\_times.txt**: è il primo dei due files relativi alla sincronizzazione dei tempi tra GNSS e computer locale. Questo file contiene, per ogni messaggio **UBX-RXM-RAWX**, l'orario locale del computer e il campo **rcvTow** del messaggio in questione. La decodifica di questo messaggio è la seconda operazione di parsing che viene svolta. I messaggi **UBX** non sono codificati in ASCII come quelli dello standard NMEA, pertanto la decodifica ha richiesto un lavoro di analisi e studio del protocollo proprietario dei ricevitori che vede ogni messaggio **UBX** iniziare con due bytes (B5 62), seguiti dalla classe e dall'identificativo del messaggio (02 per **RXM** e 15 per **RAWX**), seguiti a loro volta dalla lunghezza del corpo del messaggio, espressa su 2 bytes e successivamente dal corpo, che contiene le informazioni ricercate, come **rcvTow** e **week**. Ogni messaggio **ubx** termina con due bytes di checksum, ossia una sequenza che esprime l'integrità del messaggio, da verificare secondo un algoritmo indicato nel manuale del protocollo (algoritmo di Fletcher a 8-bit, noto per essere utilizzato nel protocollo di rete TCP);

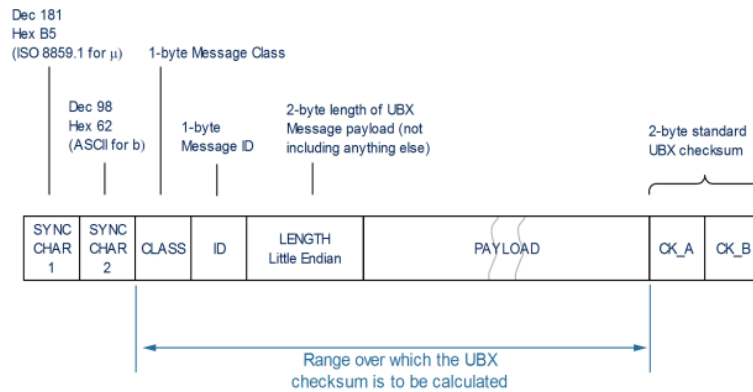


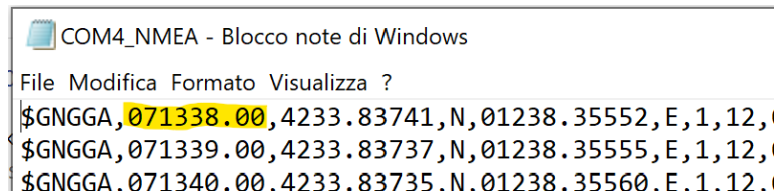
Figura 4: Frammento UBX

Message	<b>UBX-RXM-RAWX</b>					
Description	<b>Multi-GNSS raw measurements</b>					
Firmware	Supported on: • <a href="#">u-blox 8 / u-blox M8 protocol versions 18, 19, 19.1, 19.2, 20, 20.01, 20.1, 20.2, 20.3, 22, 22.01, 23 and 23.01 (only with ADR or High Precision GNSS or Time Sync products)</a>					
Type	Periodic/Polled					
Comment	This message contains the information needed to be able to generate a RINEX 3 multi-GNSS observation file (see <a href="ftp://ftp.igs.org/pub/data/format/">ftp://ftp.igs.org/pub/data/format/</a> ). This message contains pseudorange, Doppler, carrier phase, phase lock and signal quality information for GNSS satellites once signals have been synchronized. This message supports all active GNSS. The only difference between this version of the message and the previous version ( <b>UBX-RXM-RAWX-DATA0</b> ) is the addition of the version field.					
Message Structure	Header	Class	ID	Length [Bytes]	Payload	Checksum
	0xB5 0x62	0x02	0x15	16 + 32*numMeas	see below	CK_A CK_B
Payload Contents:						
Byte Offset	Number Format	Scaling	Name	Unit	Description	
0	R8	-	rcvTow	s	Measurement time of week in <a href="#">receiver local time</a> approximately aligned to the GPS time system. The receiver local time of week, week number and leap second information can be used to translate the time to other time systems. More information about the difference in time systems can be found in the RINEX 3 format documentation. For a receiver operating in GLONASS only mode, UTC time can be determined by subtracting the leapS field from GPS time regardless of whether the GPS leap seconds are valid.	
8	U2	-	week	weeks	GPS week number in <a href="#">receiver local time</a> .	
10	I1	-	leapS	s	GPS leap seconds (GPS-UTC). This field represents the receiver's best knowledge of the leap seconds offset. A flag is given in the recStat bitfield to indicate if the leap seconds are known.	
11	U1	-	numMeas	-	Number of measurements to follow	

Figura 5: Struttura del Payload di un messaggio UBX-RXM-RAWX



- `PORTA_NMEA_times.txt`: è il secondo ed ultimo file generato dal software. Contiene l'ulteriore sincronizzazione tra i tempi GNSS e computer locale, estratti dalle stringhe NMEA. Come riportato nella specifica dei messaggi NMEA-GGA, l'orario in formato UTC è il secondo parametro da sinistra dopo il titolo del messaggio, separato da virgola.



```

File Modifica Formato Visualizza ?
$GNGGA,071338.00,4233.83741,N,01238.35552,E,1,12,
$GNGGA,071339.00,4233.83737,N,01238.35555,E,1,12,
$GNGGA,071340.00,4233.83735,N,01238.35560,E,1,12,

```

Figura 6: Orario UTC contenuto nel messaggio NMEA-GGA

Fino a che l'utente non ne richiede l'interruzione, si ha un loop che raccoglie i dati e li elabora nelle modalità sopra descritte per ciascun file.

In accordo con i requisiti precedentemente analizzati, al fine di garantire pieno supporto a più di un ricevitore, è stato fatto ricorso alla programmazione concorrente: nel processo di acquisizione dei dati dai ricevitori, viene creato un thread per ciascun ricevitore e, una volta terminata l'acquisizione, questo viene terminato.

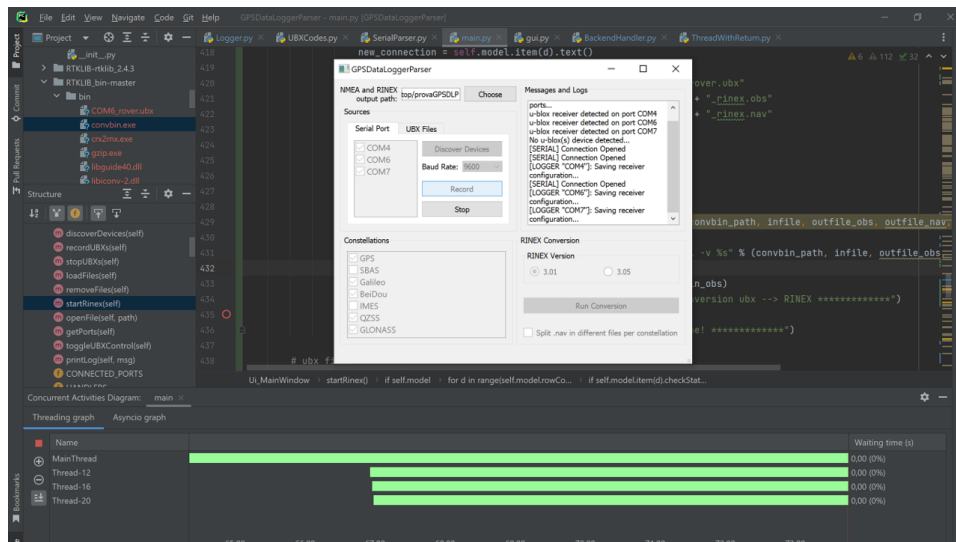


Figura 7: debug del software con monitoraggio dei thread in esecuzione

Prima della terminazione dei thread, vengono chiusi tutti i file creati all'inizio dell'acquisizione per ciascun ricevitore.

### 3.1.3 Elaborazione dei files RINEX

Grazie alla libreria RTKLib e all'integrazione con le API del sistema operativo fornita dalla libreria *os*, su pressione del pulsante relativo all'avvio della conversione, viene eseguita una riga di comando che a partire dal file `.ubx` acquisito o caricato dall'utente, genera i files RINEX `.obs` o `.nav`, oppure files suddivisi per costellazione (a seconda della selezione dell'opzione da parte dell'utente) nella versione scelta dall'utente.

### 3.1.4 Funzionalità Aggiuntive

Oltre ai requisiti indicati nella specifica, il software permette di caricare stream già acquisiti in formato `ubx`, per poter estrarre da questi le stringhe NMEA ed effettuare la generazione dei RINEX nelle modalità sopra indicate.

## 3.2 Fasi dello Sviluppo

Lo sviluppo del progetto è stato suddiviso nelle seguenti fasi:

- colloquio di presentazione con i tutor relativo alla scelta del progetto;
- invio della proposta e valutazione di quest'ultima da parte dei tutor e dei colleghi dell'Università di Ferrara;
- inizio dei lavori seguendo un approccio *bottom-up* modulare:
  - prototipo della GUI;
  - sviluppo in codice della GUI;
  - sviluppo dell'oggetto **SerialParser** e test con un ricevitore;
  - sviluppo dell'oggetto **Logger** e test con un ricevitore;
  - sviluppo dell'oggetto **ThreadWithReturn** e **BackendHandler** e test con uno e successivamente più ricevitori;
  - integrazione di tutte le funzionalità con la GUI
  - testing e misurazione delle performance
- presentazione del progetto ai tutor
- stesura della relazione (per i tutor), presentazione (per la docente referente del progetto) e documentazione (descrizione e commento del codice);
- consegna del progetto

### 3.3 Test e Difficoltà Riscontrate

Al termine dello sviluppo di ciascun modulo sono stati condotti dei test sperimentali (utilizzando prima uno, poi tutti i ricevitori forniti per il test). Dopo aver verificato la correttezza dell'apertura, chiusura, raccolta dati e gestione di eccezioni improvvise della periferica, si è passati alla verifica dell'elaborazione dei messaggi da ricevere e dei comandi da inviare costruendo i vari messaggi di tipo `UBX-CFG-*`, e all'elaborazione dei messaggi inviati. Successivamente si è passati alla generazione dei file di log e all'integrazione con più dispositivi utilizzando il multithreading. La fase di sviluppo è terminata con l'integrazione di tutte le funzionalità con la GUI.

È stato qui che si sono verificate alcune difficoltà dovute al fenomeno delle corse critiche tra thread. In particolare, si verificavano conflitti tra il thread che si occupa dell'aggiornamento della GUI e quelli relativi all'acquisizione degli stream dei ricevitori. Per stampare a video il *log* dell'attività all'interno del box contenuto nella GUI, i thread dei ricevitori violavano l'accesso esclusivo alle risorse condivise (GUI). Il tutto è stato risolto grazie all'implementazione di un oggetto intermedio `GUIUpdater` guadagnando soltanto lui un accesso esclusivo.

## 4 Conclusione

A conclusione di questo progetto, ringrazio i proff. Mirko Reguzzoni e Lorenzo Rossi per il supporto, la disponibilità e la pazienza che sempre hanno avuto nello spiegarmi concetti un po' lontani dalle mie conoscenze.

Il progetto è stato utile per apprendere e sperimentare sul campo abilità relative alla gestione delle dinamiche relazionali con i committenti, alle capacità di progettare una soluzione e realizzarla mediante codice.

"GPSTDataLogger and Parser" è quindi un software open-source, modulare e quindi scalabile. La modularità apre le porte a future implementazioni di possibili funzionalità come il plot real-time della posizione in opportuni grafici o la comunicazione con dispositivi in rete, per arrivare all'implementazione dell'intero software o porzioni di esso in altri progetti.