

Sistema Multimodale di Sensoristica con scambio dati in formato Vrs

Marco Greco, Marco Scarabello e Cristian Tramarin
20 Marzo 2023



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

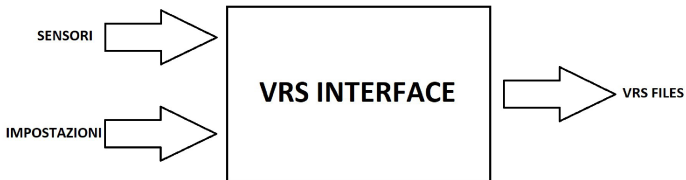


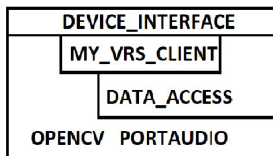
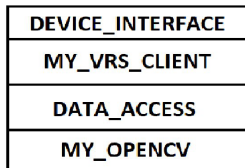
- 1 Softcoded vrs interface
- 2 La Crosscompilazione
- 3 Scambio dati protocollo Client Server

Per lo sviluppo del dispositivo Aria, gli sviluppatori di Meta hanno progettato un nuovo sistema di formattazione, al fine di gestire la sensoristica, chiamato VRS. Esso è in grado di processare in poco tempo anche quantità considerevoli di dati dai sensori, oltre a permettere di includere in un'unico file le registrazioni di tutti i dispositivi attivi, previa una configurazione iniziale, il cui stato viene poi anch'esso salvato nel file vrs. Per queste sue proprietà, abbiamo deciso di usare la libreria VRS per la progettazione di un'applicazione per dispositivi Android in grado, attraverso una connessione TCP IP, di inviare i dati di sensoristica, quali microfono, telecamera e giroscopio, ad un server.

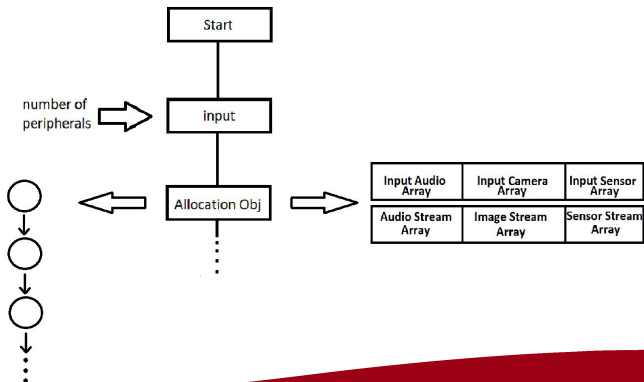
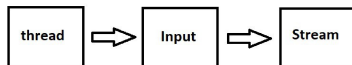
Il progetto quindi verte sullo sviluppo di un'applicazione acquisizione di dati da un sistema multimodale di sensoristica, tramite l'utilizzo di una formattazione file vrs e una connessione Client Server. Abbiamo utilizzato:

- Software Unity
- Codice C++
- Codice C#
- Vrs

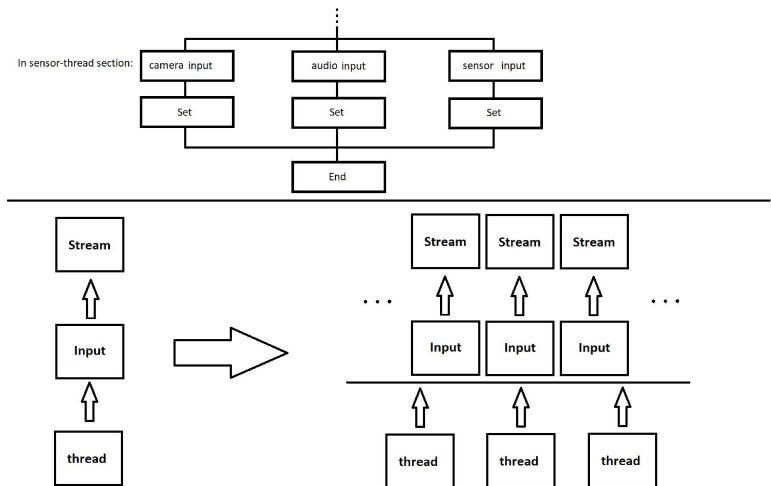




DEVICE_INTERFACE:



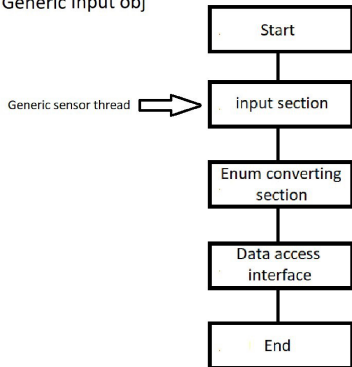
Softcoded vrs interface



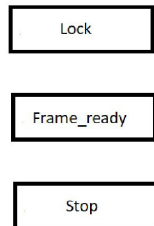
Softcoded vrs interface



Generic Input obj



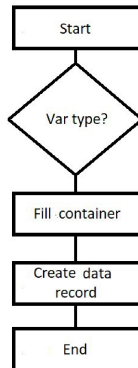
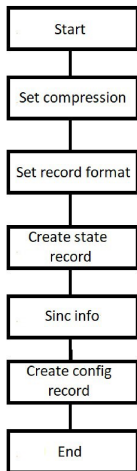
In-out flags



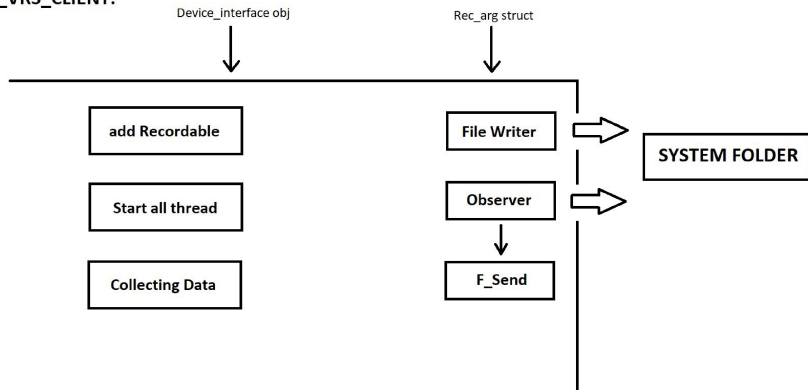
Softcoded vrs interface



Generic stream obj



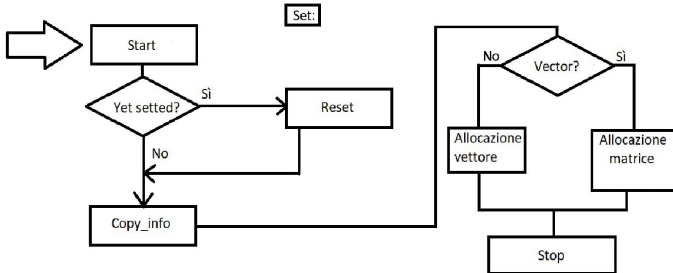
MY_VRS_CLIENT:

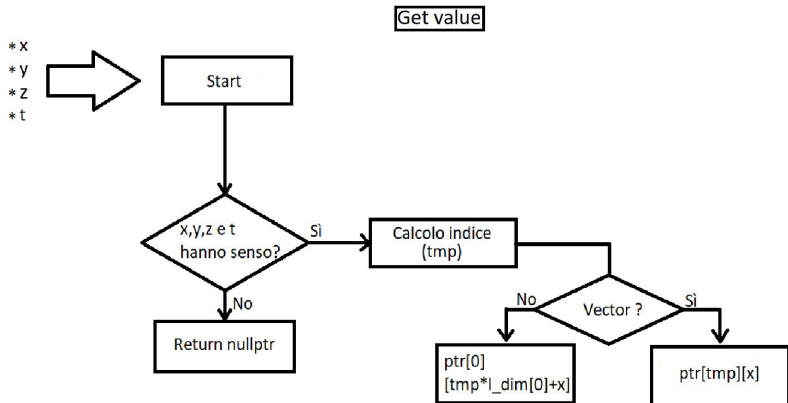


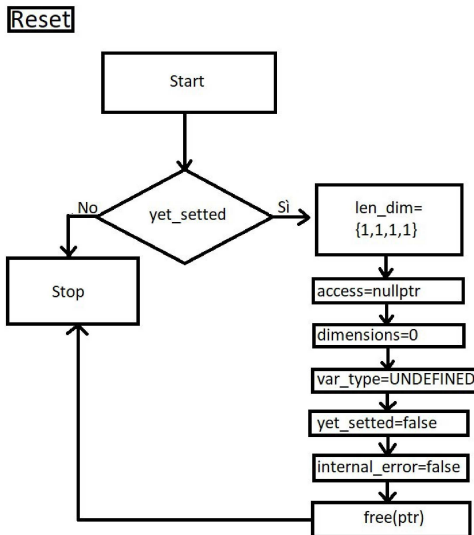
Data_access:



- * Data
- * Var_type
- * Dim
- * Dim_len
- * Vector







Android supporta applicazioni in C# e Java, per cui scelgo Unity. Posso usare codice C++ solo se precompilato e wrappato in una libreria dinamica. Necessaria quindi crosscompilazione per ottenere una libreria dinamica compatibile con il dispositivo bersaglio.

Ho riscontrato i seguenti problemi maggiori:

- Come crosscompilare.
- Come rappresentare i dati da fornire alle funzioni VRS.
- Come scambiare dati in maniera continua tra applicazione e libreria.

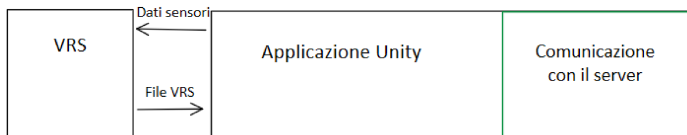
Processo di compilazione attraverso il quale si genera del codice eseguibile da una architettura differente da quella su cui si compila.

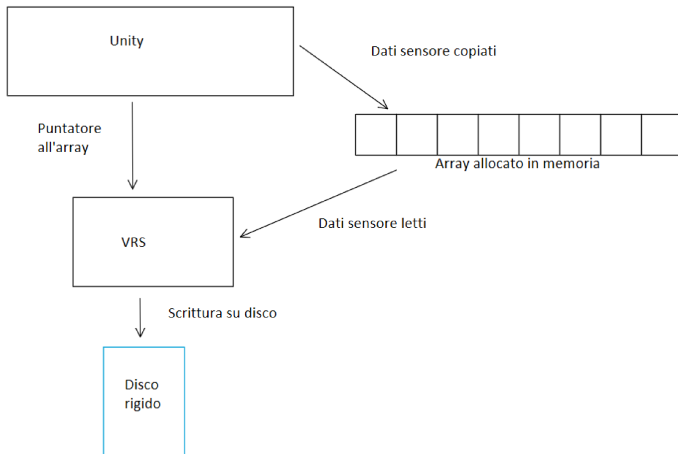
Tool necessari su Linux:

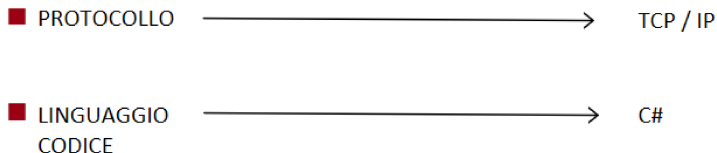
- Android ndk
- Cmake

Procedura:

- 1** Configurazione iniziale
- 2** Installazione delle dipendenze cross-compilate
- 3** Cross-compilazione della libreria

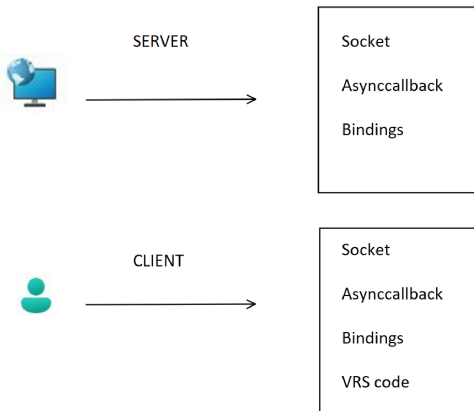




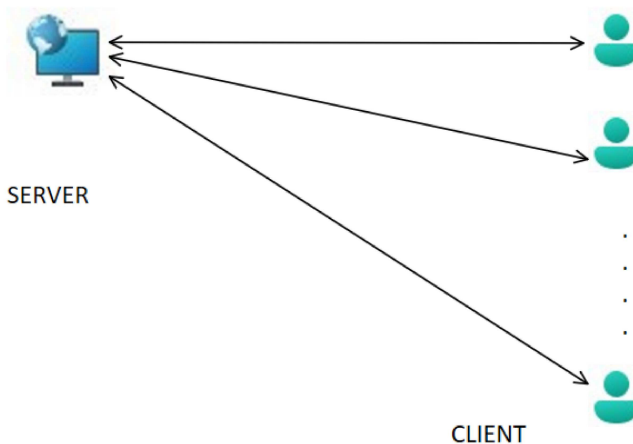


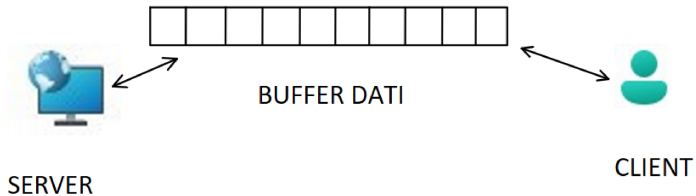
C# si interfaccia con il programma Unity e presenta delle librerie atte alla creazione di un codice Client Server.

Protocollo Client Server, con connessione AsyncCallback Così costituito:



Server Multi-client così costituito:





Le possibili implementazioni future:

- Introdurre un sistema di gestione tramite database file vrs lato server.
- Implementare un metodo tramite richiesta per l'invio file da server verso client.
- Migliorare la gestione di memoria della connessione Client Server.