



**Università degli Studi di Bergamo**

---

FACOLTÀ DI INGEGNERIA  
Corso di Laurea Magistrale in Ingegneria Informatica

## **Relazione di Robotica**

Laureandi:

**Pitturelli Leandro**

Matricola 1043991

**Sardi Dario**

Matricola 1043991

# Indice

<b>1</b>	<b>ROS per NVIDIA Jetson TX2</b>	<b>2</b>
1.1	Setup tramite tool Jetpack . . . . .	2
1.2	Backup e ripristino . . . . .	2
1.2.1	Backup . . . . .	2
1.2.2	Ripristino . . . . .	2
1.3	Possibile errore . . . . .	3
1.4	Setup di Ubuntu . . . . .	3
1.4.1	Setup tastiera . . . . .	3
1.4.2	Installazione ros . . . . .	3
<b>2</b>	<b>ROS: COMUNICAZIONE SERIALE TRAMITE USB</b>	<b>4</b>
<b>3</b>	<b>ZED</b>	<b>8</b>
3.1	ZED SDK . . . . .	8
3.2	ZED-ROS WRAPPER . . . . .	8

# 1 ROS per NVIDIA Jetson TX2

Questa parte del progetto ha richiesto l'installazione e la configurazione dell'ambiente ROS sulla scheda Jetson TX2. Tra le operazioni preliminari è stata effettuata l'installazione di L4T (linux for tegra) sulla scheda per fornire kernel, driver, librerie e un sistema operativo correttamente configurato per l'architettura specifica della Jetson.

## 1.1 Setup tramite tool Jetpack

La scheda è dotata di pulsanti per il controllo diretto, per l'accensione della scheda è necessario premere il tasto contrassegnato sulla pcb con il label 'POWERBTN'.

La scheda inizialmente necessita di un flash tramite un tool chiamato 'JetPack' scaricabile dal sito nvidia da eseguire su un pc dotato di porta usb.

Come prima cosa collegare la Scheda Jetson a internet attraverso un cavo ethernet connesso allo stesso router del pc su cui eseguiamo Jetpack mantenendo la scheda spenta. Sulla scheda sono presenti 4 tasti rossi evidenziati da un riquadro rosso nella 1 che, dall'alto al basso, sono: RESET, VOL-, RECOVERY, POWER ON/OFF.

Avviare la scheda jetson in 'recovery mode' seguendo questa sequenza di operazioni:

1. accendere la scheda premendo il pulsante di POWER ON/OFF .
2. tendo premuto il pulsante RECOVERY premere e rilasciare il pulsante RESET.
3. rilasciare dopo 2 secondi dalla pressione del pulsante RESET il pulsante RECOVERY.
4. Collegare il cavo micro usb B alla scheda e l'altra estremità ad una porta usb sul pc su cui si eseguirà jetpack.

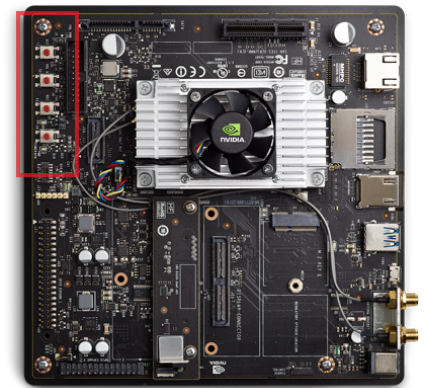


Figura 1: Jetson TX2.

Una volta scaricato l'archivio dal sito ufficiale e estratto in una cartella su un pc (non quindi sulla jetson) avviare da terminale (le istruzioni sono per S.O. linux) il file eseguibile 'Jetpack-L4T-versione-linux-x64.run' tramite il comando './Jetpack-L4T-versione-linux-x64.run', seguire le istruzioni dell'interfaccia che si presenterà a video per selezionare cartella di installazione, modello della scheda e selezione dei pacchetti che si desidera installare sulla jetson quali openCV e il toolkit CUDA. Accettare eventualmente i termini e le condizioni d'uso una volta premuto 'Next'. Nella scheda 'Network layout' selezionare la scelta coerente con la configurazione di rete attuale della nostra scheda ovvero la prima ('Device access internet via router/switch'). Selezionare la propria interfaccia di rete connessa a internet (nel dubbio utilizzare il comando da terminale 'ifconfig' e cercare a quale interfaccia di rete è assegnato un indirizzo IPv4). Dopo la schermata riassuntiva, alla pressione del tasto next si presenterà una finestra del terminale per visualizzare i progressi dell'installazione. In caso di errori relativi al collegamento della jetson ripetere la sequenza di operazioni per porla in recovery mode.

## 1.2 Backup e ripristino

Una volta eseguito jetpack e scaricate le dipendenze verranno creati in specifiche cartelle script per eseguire operazioni manualmente sulla jetson quali backup e ripristino ad esempio.

Per fare ciò navighiamo nella cartella contenente L4T (nel mio calcolatore la directory era '64\_TX2/Linux\_for\_Tegra') dove sarà presente il file 'flash.sh'.

Apriamo dunque un terminale in questa posizione e colleghiamo la scheda al pc in modalità recovery.

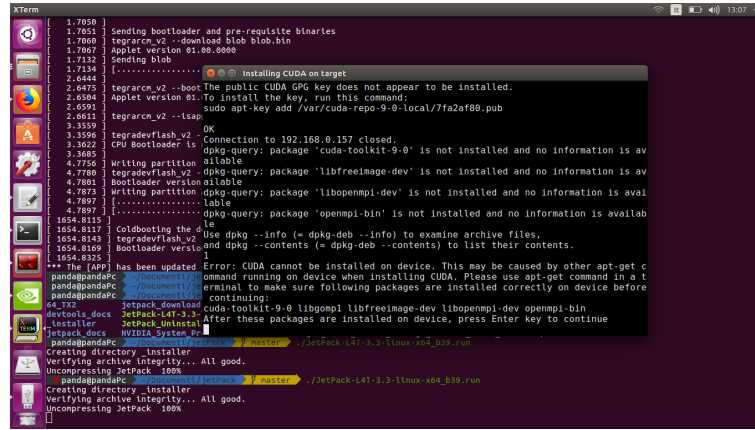
### 1.2.1 Backup

Utilizzare il comando "sudo ./flash.sh -r -k APP -G backup.img jetson-tx2 mmcblk0p1" che eseguirà il backup di tutta la memoria dell'emmc da 32Gb della scheda generando due files 'backup.img' e 'backup.img.raw' entrambi da conservare.

### 1.2.2 Ripristino

Per ripristinare un backup posizionare il "backup.img" nella directory contenente flash.sh e copiare il file backup.img.raw nella cartella bootloader rinominandolo system.img. Aprire ora un terminale nella cartella contenente flash.sh e digitare "sudo ./flash.sh -r -k APP jetson-tx2 mmcblk0p1" , questo comando si occuperà di eseguire il flash della partizione p1 del disco mmcblk0 (l'emmc della scheda.)

## 1.3 Possibile errore



```
1.7850 ] Sending bootloader and pre-requisite binaries
1.7860 ] tegra v2 --download blob blob.bin
1.7867 ] Applet version 01.00.0000
1.7872 ] Sending blob
1.7873 ] [.....] Installing CUDA on target
2.6444 ] tegra v2 --boot The public CUDA GPG key does not appear to be installed.
2.6475 ] To install the key, run this command:
2.6504 ] sudo apt-key add /var/cuda-repo-9-0-local/7fa2af80.pub
2.6509 ] tegra v2 --isap OK
3.3559 ] tegradevflash v2 Connection to 192.168.0.157 closed.
3.3596 ] CPU Bootloader is dpg-query: package 'cuda-toolkit-9-0' is not installed and no information is av
3.3627 ] Writing partition aillable
3.3685 ] tegradevflash v2 dpg-query: package 'libfreeimage-dev' is not installed and no information is av
4.7756 ] Bootloader version aillable
4.7780 ] dpg-query: package 'libopenmpi-dev' is not installed and no information is avai
4.7801 ] Writing partition aillable
4.7823 ] dpg-query: package 'openmpi-bin' is not installed and no information is avail
4.7897 ] [.....] table
4.7897 ] Coldbooting the d
1654.8112 ] tegradevflash v2 Use dpg --info (= dpg-deb --info) to examine archive files.
1654.8142 ] tegradevflash v2 and dpg --contents (= dpg-deb --contents) to list their contents.
1654.8169 ] Bootloader version 1
1654.8325 ] The [489] has been updated. Error: CUDA cannot be installed on device. This may be caused by other apt-get c
pandagpandac ommand running on device when installing CUDA. Please use apt-get command in a t
pandagpandac ermal to make sure following packages are installed correctly on device before
pandagpandac continuing:
64_T42 Jetpack_download cuda-toolkit-9-0 libgomp1 libfreeimage-dev libopenmpi-dev openmpi-bin
pandagpandac JetPack-447-3-3 JetPack-447-3-3-1 Linux-ARM-64_339.run
pandagpandac JetPack_Uninstall After these packages are installed on device, press Enter key to continue
pandagpandac NVIDIA System 0
pandagpandac pandagpandac /JetPack-447-3-3-Linux-ARM-64_339.run
Creating directory _installer All good.
Verifying archive integrity... All good.
Uncompressing JetPack 100%
pandagpandac /JetPack-447-3-3-Linux-ARM-64_339.run
Creating directory _installer All good.
Verifying archive integrity... All good.
Uncompressing JetPack 100%
```

Figura 2: errore durante il flash.

Nella mia personale esperienza si è verificato un errore durante il flash della scheda dovuto all'interruzione della connessione internet, probabilmente causato da filtri sulla rete universitaria o a una momentanea instabilità della connessione. Per risolvere il problema ho collegato la tastiera alla scheda jetson, ho installato manualmente i pacchetti mancanti tramite apt-get e infine premuto enter come indicato da jetpack nel terminale sul mio pc personale (n.b. eseguire "sudo apt-get update" sulla jetson prima di qualsiasi apt-get install per aggiornare le repositories)

## 1.4 Setup di Ubuntu

### 1.4.1 Setup tastiera

Avviato ubuntu il layout di tastiera potrebbe essere errato, per cambiarlo digitare in un terminale 'sudo dpkg-reconfigure keyboard-configuration':

1. accettare 'Generic 105-key (intl) PC'
2. selezionare Italian nel menù del paese di origine della tastiera
3. accettare le restanti impostazioni senza modificarle

### 1.4.2 Installazione ros

Seguire le indicazioni per l'installazione di ros-kinetic dal [sito ufficiale per ubuntu ARM](#).

## 2 ROS: COMUNICAZIONE SERIALE TRAMITE USB

Per interfacciarsi con board quali arduino tramite ros è risultato necessario un metodo per la comunicazione seriale tramite porta USB per poter ricevere e inviare dati in modo semplice e efficace. Tale metodo è stato sviluppato pensando ad un utilizzo con board arduino su macchina linux e sviluppato in C++ per essere compatibile con ROS. Tutte le board arduino sono dotate di un chip UART ( Universal asynchronous receiver-transmitter) incaricato della comunicazione seriale. Il metodo di comunicazione asincrono si basa sull'invio di dati a una velocità predefinita e con dati strutturati in modo ben preciso:

1. Bit contenenti parte del messaggio complessivo di grandezza predefinita (dai 5 ai 9 in base alle impostazioni)
2. Bit di sincronizzazione per comunicare l'inizio e la fine di ciascun pacchetto di dati.
3. Bit di parità per il controllo di errori
4. velocità di trasmissione dei dati per impostare su ogni singolo invio la velocità di invio e ricezione.

Sono state sperimentate diverse librerie per la comunicazione seriale e metodi basati su programmazione seriale per sistemi POSIX per scegliere infine la libreria boost per la sua semplicità e per la documentazione online relativamente completa. Il codice scritto per la comunicazione è il seguente:

```
1 #include <unistd.h>
2 #include <iostream>
3 #include <boost/asio.hpp>
4 #include <boost/asio/serial_port.hpp>
5 #include <boost/system/error_code.hpp>
6 #include <boost/system/system_error.hpp>
7 #include <boost/bind.hpp>
8 #include <boost/thread.hpp>
9 #include <boost/date_time/posix_time/posix_time.hpp>
10 #include <boost/thread/thread.hpp>
11 #include <thread>
12
13 class SerialIO{
14 private:
15     boost::asio::io_service io_svc;
16     boost::asio::serial_port m_port;
17 public:
18     SerialIO(std::string portDir):m_port(io_svc,portDir){}
19
20     void write(std::string input){
21         const int BUFFSIZE=20;
22         boost::array<char,BUFFSIZE> buf;
23         if(sizeof(input)>0){
24             for(int i=0;i<sizeof(input)/8;i++){
25                 buf[i]=input[i];
26             }
27             buf[sizeof(input)/8]=0;
28             m_port.write_some(boost::asio::buffer(buf,BUFFER));
29         }
30     }
31
32     void read(){
33         const int BUFFSIZE=10;
34         char read_msg_[BUFFSIZE];
35         std::stringstream ss;
36         int read_size=0;
37         read_size=m_port.read_some(boost::asio::buffer(read_msg_,BUFFSIZE));
38         while(read_msg_[read_size-1]!='\n'){//ciclo fino al fine stringa
39             read_msg_[read_size]=0;//terminatore stringa inserito manualmente
40             ss<<read_msg_;
41             read_size=m_port.read_some(boost::asio::buffer(read_msg_,BUFFSIZE));
42         }
43         read_msg_[read_size]=0;
44         ss<<read_msg_;
45         std::cout << ss.str();
46     }
};
```

Le funzioni di read e write nella comunicazione asincrona stabilita sono bloccanti pertanto la coordinazione dei messaggi deve essere fatta a monte e non viene gestita dal programma. Questa libreria espone i due metodi di read e write e necessita di un'inizializzazione tramite il path del file che punta al dispositivo hardware nella cartella /dev/ (ad esempio un path valido può essere "/dev/tty1"). Il punto chiave della funzione read si trova alla riga 37 dove viene invocata la funzione di lettura dalla libreria boost:

```
37 read_size=m_port.read_some(boost::asio::buffer(read_msg_,BUFSIZE));
```

La funzione read\_some legge in modo asincrono parte del messaggio, lo copia nel buffer creato ad hoc dal nome read\_msg\_ e di dimensione BUFSIZE, ritornando la dimensione del messaggio letto. Il ciclo while successivo serve per giungere al finestringa e copiare in uno stringstream il contenuto dei vari pacchetti del messaggio asincrono.

```
38 while(read_msg_[read_size-1]!='\n')
```

Viene ripetuta la lettura fino a quando non viene trovato come ultimo carattere letto il terminatore. Si presuppone quindi che il codice inviato tramite arduino comunichi con una println sulla seriale o che sia inserito il terminatore di stringa manualmente al termine di un messaggio con una semplice print.

```
39 read_msg_[read_size]=0;  
40 ss<<read_msg_;
```

Poiché il buffer creato da asio risultava essere non vuoto la copia del buffer nello stringstream senza aggiungere il terminatore manualmente generava stringhe errate, la riga 39 ovvia a questo problema.

```
45 std::cout << ss.str();
```

Il messaggio viene stampato sullo standard output utilizzando la funzione di conversione da stringstream a stringa. Questa parte del codice può essere modificata per eventualmente effettuare un post in un topic ros. La funzione write è piuttosto simile alla read:

```
21 const int BUFSIZE=20;  
22 boost::array<char,BUFSIZE> buf;
```

Viene creato un buffer di dimensione massima 20 (buffer troppo grossi rallentano notevolmente il programma e causano blocchi, ho trovato 20 un numero adeguato ma il valore può essere modificato secondo necessità).

```
24 for(int i=0;i<sizeof(input)/8;i++){  
25     buf[i]=input[i];  
26 }
```

La stringa fornita come input viene copiata nel buffer, il ciclo for è stato usato per l'impossibilità di usare funzioni quali strcpy o alternative più efficienti.

```
27 buf[sizeof(input)/8]=0;
```

Viene aggiunto manualmente un terminatore nel buffer onde evitare errori.

```
28 m_port.write_some(boost::asio::buffer(buf,sizeof(input)));
```

Viene inviato il messaggio contenuto nel buffer specificando la dimensione del messaggio da inviare.

Per una piu facile inizializzazione della comunicazione seriale è stato scritta una libreria sempre in C++ per trovare un Arduino e ritornare il path del suo riferimento necessario per l'inizializzazione.

#### deviceFinder.h

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <errno.h>
4 #include <malloc.h>
5 #include <iostream>
6 #include <string.h>
7 #define BUF_SIZE 1024
8 char addr[30]="";
9 int search(void)
10 {
11     FILE *f;
12     char* buf;
13     f=popen("./finder.o | grep Arduino | awk '{print $1;}'", "r");
14     if (f==NULL) {
15         perror("1 - Error");
16         return errno;
17     }
18     buf=(char*) malloc (BUF_SIZE);
19     if (buf==NULL) {
20         perror("2 - Error");
21         pclose(f);
22         return errno;
23     }
24     while (fgets(buf,BUF_SIZE,f)!=NULL) {
25         //printf("arduino found at: %s, len %d\n",buf,strlen(buf));
26
27         for(int i=0;i<strlen(buf);i++){
28             if (buf[i]==' ' | buf[i]=='\n') {
29                 //printf("find it\n");
30                 buf[i]=0;
31             }
32             else {
33                 //printf("%c\n-",buf[i]);
34             }
35         }
36         strcpy(addr,buf); //copia l'ultimo risultato
37     }
38     pclose(f);
39     free(buf);
40     return 0;
41 }
42
43 char* getArduino(){
44     search();
45     //printf("%s-+-+-",addr);
46     return addr;
47 }
```

#### finder.o

```
1 #!/bin/bash
2 for sysdevpath in $(find /sys/bus/usb/devices/usb* -name dev); do
3     (
4         syspath="${sysdevpath%/dev}"
5         devname="$(udevadm info -q name -p $syspath)"
6         [[ "$devname" == "bus/*" ]] && continue
7         eval "$(udevadm info -q property --export -p $syspath)"
8         [[ -z "$ID_SERIAL" ]] && continue
9         echo "/dev/$devname - $ID_SERIAL - $ID_VENDOR - $ID_TYPE"
10    ) done
```

La libreria deviceFinder.h sfrutta lo script per la bash finder.o per ricavare la stringa desiderata. È importante notare che il programma finder.o deve avere le autorizzazioni per essere eseguito (chmod +x finder.o) e deve essere eseguito su un sistema Linux. Lo script finder.o elenca semplicemente path dei device connessi e relativi id caratteristici:

```
8 echo "/dev/$devname - $ID_SERIAL - $ID_VENDOR - $ID_TYPE"
```



### 3 ZED

Lo ZED è una fotocamera che riproduce il modo in cui funziona la visione umana. Usando due sensori RGB da 4MP e attraverso la triangolazione, la ZED fornisce una comprensione tridimensionale della scena che osserva, permettendo al robot di diventare consapevole dello spazio e del movimento. Cattura video 3D ad alta definizione, fino a 2K, con un ampio campo visivo (110°) e apertura f/2.0. La camera genera due flussi video destro e sinistro sincronizzati in formato side-by-side su USB 3.0. È disponibile la retrocompatibilità con USB 2.0.

Sono disponibili diverse modalità video:

Modalità video	Risoluzione di uscita (affiancata)	Frame Rate (fps)	Campo visivo
2.2K	4416x1242	15	Largo
1080p	3840x1080	30, 15	Largo
720p	2560x720	60, 30, 15	Extra largo
WVGA	1344x376	100, 60, 30, 15	Extra largo

Lo ZED emette le immagini in diversi formati. È possibile selezionare tra immagini rettificate, non rettificate e in scala di grigi.

**Profondità** Lo ZED riproduce il funzionamento della visione binoculare umana. Gli occhi umani sono separati orizzontalmente di circa 65 mm in media. Quindi, ogni occhio ha una visione leggermente diversa del mondo circostante. Confrontando queste due viste, il nostro cervello può inferire non solo la profondità ma anche il movimento 3D nello spazio. Allo stesso modo, lo ZED ha due occhi separati da 12 cm che catturano un video 3D ad alta risoluzione della scena e stimano profondità e movimento confrontando lo spostamento dei pixel tra le immagini sinistra e destra.

**Mappa di profondità** Lo ZED memorizza un valore di distanza (Z) per ciascun pixel (X, Y) nell'immagine. La distanza è espressa in metri e calcolata dal retro dell'occhio sinistro della telecamera all'oggetto della scena.

Le mappe di profondità catturate dallo ZED non possono essere visualizzate direttamente poiché sono codificate su 32 bit. Per visualizzare la mappa di profondità, è necessaria una rappresentazione monocromatica (in scala di grigi) a 8 bit con valori compresi tra [0, 255], dove 255 rappresenta il valore di profondità più vicino possibile e 0 il valore di profondità più distante possibile.

**Mappatura 3D** La ZED scansiona continuamente i suoi dintorni e crea una mappa 3D di ciò che vede. Questa mappa viene aggiornata quando il dispositivo viene spostato acquisendo nuovi elementi nella scena. Poiché la fotocamera percepisce le distanze oltre la gamma dei tradizionali sensori RGB-D, è in grado di ricostruire rapidamente mappe 3D di grandi aree interne ed esterne.

#### 3.1 ZED SDK

Prima di tutto bisogna scaricare e installare la SDK relativa. Essa contiene tutte le librerie e strumenti che consentono di testare le caratteristiche e modificare le impostazioni della ZED. La jetson-TX2 utilizza una versione diversa di linux, chiamata L4T (linux for tegra). Dal sito bisogna cercare nella pagina del download ZED SDK per la versione per TX2. Si noti che l'SDK ZED è collegato con la versione CUDA utilizzata da JetPack. È possibile verificare il corretto funzionamento della ZED utilizzando i tool presenti nell'SDK.

```
/usr/local/zed/tools
```

Si possono avviare lo ZED Explorer che permette un'anteprima e la registrazione. Consente di modificare la risoluzione video, le proporzioni, i parametri della fotocamera e acquisire istantanee ad alta risoluzione e video 3D.

```
/usr/local/zed/tools/ZED Explorer
```

Oppure lo ZED Depth Viewer utilizza l'SDK per acquisire e visualizzare la mappa di profondità e la nuvola di punti 3D.

```
/usr/local/zed/tools/ZED Depth Viewer
```

#### 3.2 ZED-ROS WRAPPER

Il pacchetto *zed-ros-wrapper* consente di utilizzare le telecamere stereo ZED con ROS. Fornisce le immagini della fotocamera sinistra e destra, la mappa della profondità, la nuvola di punti, le informazioni sulla posa e supporta l'uso di più telecamere ZED.

Per poterlo utilizzare devono essere soddisfatti alcuni prerequisiti

- Ubuntu 16.04
- ZED SDK  $\geq 2.3$  e la sua dipendenza CUDA
- ROS Kinetic

Per installare `zed-ros-wrapper` , aprire un terminale bash, clonare il pacchetto da Github e crearlo:

```
cd ~/catkin_ws/src/ #use your current catkin folder
git clone https://github.com/stereolabs/zed-ros-wrapper.git
cd ..
catkin_make -DCMAKE_BUILD_TYPE=Release
echo source $(pwd)/devel/setup.bash >> ~/.bashrc
source ~/.bashrc
```

Aprire un terminale e utilizzare `roslaunch` per avviare il nodo ZED:

```
roslaunch zed_wrapper zed.launch
```

**webcam interna jetson-TX2** La jetson-TX2 è fornita di una telecamera interna che avrà come indice `/dev/video0` mentre la telecamera ZED, se non ci sono collegate altre camere, sarà `/dev/video1`. Il pacchetto `zed-ros-wrapper` avrà un launch file che chiamerà la camera di default come indice 0. Modificare il `zed.launch` dentro il percorso `ROS/src/zed-ros-wrapper/zed_wrapper/launch` inserendo nel argomento relativa alla camera il valore 1.

```
<arg name="camera_model"          default="1"/>
```

Per vedere se correttamente la ZED comunica con ROS avviare RVIZ.

RVIZ è un'interfaccia grafica di ROS che consente di visualizzare molte informazioni, utilizzando plugin per molti tipi di topic disponibili.

Per una documentazione più approfondita sulla ZED e i nodi utilizzati è consigliabile consultare [docs](#)