# Sistema operativo Linux della scheda Brain&Vision (Robot I-Droid01)

## 1 LOGIN

Per accedere al sistema linux occorre collegare tramite il cavo seriale la schedina di comunicazione della Brain&Vision con una porta COM del PC. A livello software si può utilizzare il programma brain_debug fornito col secondo CD oppure un terminale seriale come l'Hyperterminal che nei sistemi Windows si trova sotto Programmi -> Accessori -> Comunicazioni. L'Hyperterminal è molto più potente del brain_debug in quanto offre funzioni come il copia incolla di testo, la possibilità di cancellare (Canc) una parola, la possibilità di scorrere i comandi precedentemente digitati usando le freccine alto/basso, la possibilità di interrompere i processi in esecuzione usando combinazioni di tasti (solitamente CTRL + C), e tanti altri.

### 1.1 Login con il brain_debug

- copiare i file brain_debug.exe e brain_config.ini in una cartella del PC
- aprire il file brain_config.ini appena copiato su PC con Blocco note e modificare la riga
- "PORT = 0" sostituendo al numero 0 il numero della porta seriale alla quale è stato collegato il robot (ad es. 10); salvare il file brain_config.ini
- avviare il programma brain_debug.exe; se tutto funziona correttamente vi comparirà una schermata nera;
- se I-D01 non è acceso, accendetelo ora e aspettate qualche secondo fino a quando compare "IC - Voice recognition started" o "BT - Bluetooth Started"
- premere invio: vi comparirà la scritta "(none) login:" scrivete "root" e poi invio; a questo punto vi compare il simbolo cancelletto "#" (che è il prompt della shell) ad indicare che Linux è pronto a ricevere comandi.

### 1.2 Login con l'Hyperterminal

- Come prima cosa il programma vi chiede di inventare un mome: inventatene uno molto originale ad esempio I-DROID01
- Nella seconda schermata tralasciate le voci "Paese", "Indicativo località", "Numero di telefono". Nella voce "Connetti" scegliete la porta COM che state utilizzando per collegarvi con I-D01.
- Nella terza schermata impostate:
- Bit per secondo (Baud): 115200
- Bit di dati: 8
- Parità: nessuno
- Bit di stop: nessuno
- Controllo di flusso (handshake): nessuno
- se I-D01 non è acceso, accendetelo ora e aspettate qualche secondo fino a quando compare "IC - Voice recognition started" o "BT - Bluetooth Started"
- premere invio: vi comparirà la scritta "(none) login:" scrivete "root" e poi invio; a questo punto vi compare il simbolo cancelletto "#" (che è il prompt della shell) ad indicare che Linux è pronto a ricevere comandi.

Quando chiudete Hyperterminal potete salvare le impostazioni; in questo modo vi comparirà nel suo menù una nuova voce col nome che avete scelto prima e che vi permetterà di collegarvi automaticamente senza dover reimpostare tutto da capo. Per le successive connessioni, se I-D01 è già acceso vi basterà avviare il programma e premere invio; se invece avete avviato prima il programma, andate sul menù Chiama -> Disconnetti -> Attendi Chiamata e quindi accendete il robot.

## 2 Caricamento del sistema operativo

Il kernel linux in I-droid è un kernel 2.4.26 standard compilato per CPU ARM con moduli aggiuntivi per la gestione delle periferiche (la gestione flash e IIC è già native nel kernel 2.4) - però la shell è ridotta (BusyBox) ed il file system è compresso (CRAMFS). La shell quindi non implementa tutti i comandi che si possono trovare in una distribuzione linux standard, perchè in realtà tutta la shell è un oggetto solo (unico eseguibile) che implementa diverse chiamate in C. Digitando HELP si possono vedere quali sono implementate. Tutto questo per ridurre le dimensioni (l'impronta) del sistema operativo e massimizzare lo spazio per i servizi.

Quando si accende I-Droid, vengono eseguite le seguenti operazioni nella B&V:

- reset vettori di interrupt e salto al bootloader
- caricamento bootloader dal suo spazio dedicato nella flash (si chiama BLOB!)
- il bootloader (equivalente diciamo a LILO "per metà" su un sistema ordinario) carica il kernel Linux e scrive "OS Loading" sul LCD
- una volta caricato il kernel con i relativi driver compilati per il support della flash (JFFS2_mtd = Journalled Flash File System 2 for Memory Technology Devices) viene eseguita la procedura usuale di avvio dei servizi da rc.d e viene lanciato lo script rc.modules che carica i relativi moduli (driver) per supportare le periferiche come la telecamera (digitando "lsmod" vengono elencati).
- Alla fine, viene lanciato lo script rc.local che lancia il file IDROID start. Questo fa partire la gestione dei sensori di Idroid ed il suo output viene inviato alla console seriale. Se toccate o interagite con I-droid in questo stato (cioè subito dopo il termine del suo avvio) vedrete sul terminale tutto ciò sente e fa (comanda) I-droid. Questa "routine", chiamata System Controller, viene lanciata in modalità non-daemon, cioè viene lanciata e non si "sgancia" dalla console (in gergo linux si dice "fork", che vuol dire forchetta o bivio, perchè il programma si separa dal chiamante e scende in background quindi lavora come servizio sganciato dalla console che lo ha chiamato). Tutto il suo output viene ridirezionato verso il cosìddetto "stdout" cioè standard output nonchè la console annessa. Ecco alcuni mesaggi di debug:

  quando viene percepito un suono
  SC - SOUND DETECTED

  quando viene toccato il capoccione
  SC - TOUCH DETECTED
  IC - Disabling Voice Recognition
  IC - Enabling Voice Recognition

  accensione Sound Follower
  SC - Start Sound Follower

  spegnimento Sound Follower
  SC - Stop Sound Follower

  per ogni movimento della testa compaiono sempre queste due stringhe
  IC - Disabling Voice Recognition
  IC - Enabling Voice Recognition
  ciò significa che quando i-droid ha compreso un comando, spegne il riconoscimento vocale e lo riaccende non appena il movimento è terminato

  quando va a dormire
  SC - Go To Sleep
  IC - End voice recognition
  IC - Disabling Voice Recognition
  IC - Voice recognition terminated
  Shutdown BT connection
  SC - Robot Sleeping

quando si risveglia
SC - Robot Wake up
IC - Start voice recognition
IC - Enabling Voice Recognition
IC - Voice recognition started
SC - SOUND DETECTED

Il System Controller termina non appena si digita un comando qualsiasi (invio) e chiude i vari collegamenti (BT, LCD etc) perché lavora in foreground e quindi "sente" i comandi dalla console. Per utilizzare la shell è necessario "uccidere" questo programma che comporta la disattivazione del modulo Bluetooth e delle funzioni avanzate della B&V; in pratica I-D01 riprende a funzionare come se non esistessero la BT e la B&V. Una volta loggati si può comunque riavviare il System Controller digitando da ovunque:

/idroid/IDROID start

# 3 ALCUNI COMANDI DELLA SHELL

Una parte dei comandi sono implementati dalla shell busybox che si trova in /bin; gli altri comandi sono programmi eseguibili che si trovano in /bin e /sbin. Le parole in rosso rappresentano i comandi da digitare; le parole in azzurro sono inventate a titolo esemplificativo.

# help
Per ottenere l'elenco dei principali comandi

# comando --help
per conoscere il significato e la sintassi di comando

# pwd
Per sapere in quale directory ci troviamo. Ad esempio, una volta loggati ci troviamo in /root

# cd ..
per tornare nella directory padre ( i due puntini devono essere staccati da 'cd')

# cd /
per tornare direttamente nella directory radice

# cd /ForumIdroid
per accedere alla directory /ForumIdroid

# cd ~
entra nella directory home dell'utente che essendo root corrisponde a /root

# cd
come il precedente; se non si specifica niente, il parametro passato per default a cd è il tilde (~)

# ls
elenca i file contenuti nella directory corrente

# ls -l
elenca i file descrivendoli dettagliatamente

# ls -a

elenca i file compresi quelli nascosti, cioè che iniziano col punto, ad esempio directory attuale e quella padre.

# ls -la
elenca i file compresi quelli nascosti descrivendoli dettagliatamente

# ls -laF
come ls -la con l'aggiunta degli asterischi per evidenziare i file eseguibili e delle slash per evidenziare le directory

# ll
(elle elle) corrisponde a ls -l

# ps
per avere l'elenco dei processi in esecuzione e i rispettivi PID

# kill [PID del processo]
per terminare un processo; il suo PID viene fornito dal comando ps.
Ad esempio si può usare per killare init (kill 1) che è il processo che gestisce tutti gli altri processi. La terminazione di init implica la chiusura e il riavvio automatico del sistema operativo.

# cp /mioFile /altroPercorso/CopiaFile
per copiare un file

# cp -R directory1 directory2
per copiare directory1 in directory2

# cp -R * nuovoPercorso
per copiare qualunque cosa (file e directory ricorsivamente) nel nuovoPercorso

# mkdir MiaDirectory
per creare una directory

# move /Percorso1/fileA /Percorso2/fileA
per spostare un file

# move fileA fileB
per rinominare il fileA in fileB

# rm file
per cancellare un file

# rmdir directory
per cancellare una directory solo se è vuota

# rm -R oggetto
per cancellare file o directory (anche non vuote)

# touch MioFile
per creare un file vuoto

# cat file
per visualizzare il contenuto di un file di testo; ma non è adatto per i file lunghi perché li fa scorrere fino alla fine senza dare il tempo di leggerli

# more file

per visualizzare il contenuto di un file di testo; permette di far scorrere la schermata un po' alla volta usando le freccine; per uscire basta premere q

# less file
come more, però lascia la schermata nera (usando Hyerterminal)

# alias MioComando='Comando Originale'
per dare un nuovo nome ad un comando; ad esempio se scriviamo alias percorso='pwd' la parola percorso equivale a pwd; se scriviamo alias dir='ls -laF' la parola dir equivale a ls -laF

# alias
per elencare tutti gli alias

# unalias MioComando
per eliminare l'alias MioComando

# unalias -a
per eliminare tutti gli alias

# ln fileA fileHardLink
per creare un hardLink a fileA; un hardlink è un file tale e quale a quello originale che cambia al cambiare dell'originale

# ln -s fileA fileSoftLink
per creare un softLink a fileA; un softlink è un file che punta all'originale, cioé è quello comunemente chiamato link

# which file
per sapere dove si trova il file specificato; questo comando cerca solo i programmi che si trovano in /bin e /sbin

# mount
per vedere come sono organizzate le partizioni. Digitando questo comando, l'output che si ottiene è:

rootfs on / type rootfs (rw)
/dev/root on / type cramfs (ro)
/proc on /proc type proc (rw)
tmpfs on /tmp type tmpfs (rw)
tmpfs on /var type tmpfs (rw)
/dev/mtdb3 on /mnt/disk type jffs2 (rw,sync)

L'ultima riga in particolare si riferisce al file system su flash sul quale si registrano alcuni dati utente. Infatti nella cartella /mnt/disk trovate la cartella programs, dove sono registrati i sorgenti in C trasmessi dal PC e gli eseguibili.

# free
per visualizzare la dimensione della memoria occupata e quella libera

# exit
per eseguire il logout

# login
per loggarsi, dopo aver fatto il logout

# whoami
per sapere con quale identità sto operando, alla fine dei conti è sempre root, dal momento che non si può cambiare identità.

# logname
per sapere con quale account mi sono loggato, alla fine dei conti è sempre root, dal momento che non si possono creare altri account.

# passwd NuovaPassword
per cambiare la propria password; questo comando è inutilizzabile perché il file che registra i dati /etc/passwd è in sola lettura

# adduser NomeUtente
per creare un nuovo account; in realtà questo comando è inutilizzabile perché il file che registra i dati /etc/passwd è in sola lettura

# su NomeUtente
per cambiare identità, da quella attuale a NomeUtente; in realtà questo comando è inutilizzabile perché non si possono creare nuovi account.

# hostname
nei normali sistemi linux fornisce il nome del computer; in questo caso restituisce (none)

# uname
fornisce il nome del sistema operativo, cioè Linux

# dmesg
fornisce l'output di caricamento del sistema

# halt
arresta il sistema; a questo punto per riavviarlo bisogna spegnere e riaccendere il robot.

# shutdown -h now
come halt

# reboot
arresta il sistema e lo riavvia

# shutdown -r now
come reboot

## 4 PROGRAMMA IDROID

Il programma IDROID, situato nella cartella /idroid/, permette di impartire comandi al robot, ad esempio per testare il suo funzionamento. Per farlo è semplice, una volta entrati dopo il login (dopo aver premuto invio ed al login prompt digitato "root" poi invio) basta digitare "cd /idroid". Da lì digitate "./IDROID" e vi verrà mostrato un listato dei comandi divisi per categorie.

Per dirgli di pronunciare il numero 0 per esempio basta digitare:
# ./IDROID play 1
e così via (i numeri sono uno in più di ciò che pronuncia fino ad un certo punto).

Se volete scrivere qualcosa sul LCD digitate:
# ./IDROID lcdwrt Hello World!

Notare che la prima parola viene sulla prima riga, e la seconda sulla seconda (lcdwrt prima_riga seconda_riga); il comando richiede sempre i 2 parametri. Ovviamente potete fargli fare altre cose utilizzando la sintassi indicata (il comando serve proprio per provare le funzionalità di idroid da "alto livello" di programmazione).

Per girare la testa a destra di 3 step:
# ./IDROID head right 3

Per girare la testa a sinistra di 2 step:
# ./IDROID head left 2

Per alzare la testa di 4 step:
# ./IDROID head up 4

Per abbassare la testa di 1 step:
# ./IDROID head down 1

Per fare ripartire la routine di controllo principale di idroid e vedere l'output degli stimoli, basta digitare (da ovunque):

# /idroid/IDROID start

e ripartira' il SYSTEM CONTROLLER. A questo punto, toccate o parlate e vedrete i messaggi di debug in console!

Esistono molti altri comandi del programma IDROID che possono essere appresi facilmente analizzando il listato come spiegato sopra.

## 5 EDITOR VI

Il VI è il classico editor del Linux che permette di editare file di testo. Per avviare il programma:

# vi MioFile.txt

Se il file esiste viene aperto, altrimenti viene creato. Qui trovate una guida a VI

Il VI è un editor particolare che era stato inventato quando ancora non esisteva il mouse e non si potevano usare tasti particolari come le freccine di scorrimento. Il VI funziona in due modalità:
1) modalità testo: permette di scrivere e basta; per entrare in modalità testo occorre premere il tasto i oppure a
2) modalità comandi: permette di modificare il testo e salvarlo; per entrare in modalità comandi occorre premere il tasto [Esc] seguito dal comando che si desidera fare.

Per scrivere (modalità testo)
i per entrare in modalità scrittura; il testo viene inserito a partire dalla posizione corrente
a per entrare in modalità scrittura; il testo viene inserito a partire dalla posizione successiva

Per modificare (modalità comandi)
[Esc] per entrare in modalità comandi
:x per cancellare il carattere segnato dal cursore
:dd per cancellare la riga nella quale ti trovi
H per spostarsi a sinistra di un carattere
J per spostarsi in basso di una riga
K per spostarsi in alto di una riga
L per spostarsi a destra di un carattere
:w per salvare
:q per uscire
:wq per salvare e uscire
:q! per uscire senza salvare

## 6 CREARE PROGRAMMI IN C

Facciamo un esempio di programmazione e compilazione sul droide partendo da zero. Andate alla cartella che contiene i programmi utente scrivendo:

```
# cd /mnt/disk/programs
```

Poi editate un nuovo programma con VI scrivendo:

```
# vi test.c
```

e scrivete questo programma:

```
#include <stdio.h>

int main(void)
{
    printf("Ciao a tutti\n");
}
```

Salvate il file e tornate al prompt della shell. Se guardate il contenuto della cartella corrente col comando:

```
# ll
```

dovreste vedere una cosa di questo tipo:

```
-rw-r--r--    1 root     root          71 Jan  1 00:10 test.c
```

A questo punto compiliamo scrivendo:

```
# tcc -otest test.c
```

Se compare il prompt senza nessun messaggio di errore allora il programma è stato compilato correttamente. Se invece ci sono messaggi di errore, editate ancora il programma di test col VI e controllate che il vostro sorgente sia esattamente come scritto sopra (verificate tutti segni di interpunzione e le maiuscole/minuscole). Corretti gli eventuali errori, ridate il comando di compilazione e vedete se va tutto bene.

Ora potete verificare che il programma è stato effettivamente compilato guardando ancora il contenuto della cartella:

```
# ll

-rwxr-xr-x    1 root     root        3008 Jan  1 00:29 test
-rw-r--r--    1 root     root          71 Jan  1 00:10 test.c
```

A questo punto eseguite il programma digitando:

```
# ./test
```

Se tutto funziona, dovreste vedere comparire la scritta "Ciao a tutti". Se siete arrivati a questo punto, avete scritto, compilato ed eseguito il vostro primo programma in C direttamente sul droide e quindi... COMPLIMENTI!!!

ESEMPIO DI PROGRAMMA
Facciamo un esempio di programma che consiste nell'accendere i led degli occhi di giallo per poi spegnerli dopo 2 secondi. Il listato è questo.

```
#include "c-like.h"
#include "robot.h"

define( behavior(Main))
{
   led_on(LED_YELLOW);
   msleep(2000);
   led_off(LED_YELLOW);
}
```

Questo listato può essere recuperato direttamente dall'ambiente Visual C-like creando lo stesso programma con i blocchi e poi andando sotto Program--> Show source; viene mostrato il sorgente in C che può essere selezionato con il mouse e poi copiato premendo [Ctrl+C]. Una volta copiato può essere incollato direttamente dentro un file della B&V aperto con l'editor VI e salvato nel percorso /mnt/disk/programs con l'estensione c (ad esempio MioProgramma.c). Prima di copiare il listato ricordatevi di entrare in modalità scrittura (tasto 'i') nel file.

Per compilare il programma:

```
tcc -I/idroid/include -lpthread -lc-like -oMioProgramma MioProgramma.c
```

Questa istruzione sta ad indicare che il file MioProgramma.c viene compilato linkandolo [-l] con le librerie pthread e c-like che si trovano [-I] nel percorso /idroid/include/.

Se la compilazione ha avuto esito positivo viene creato il file eseguibile MioProgramma; controllate digitando

```
# ls -laF
```

che vi permette di listare il contenuto della cartella evidenziando i file eseguibili con l'asterisco *. Quindi dovreste vedere il file MioProgramma*

A questo punto, se provate a mandare in esecuzione il programma

```
# ./MioProgramma
```

il programma non sarà eseguito e vi compariranno queste scritte:

```
[Runtime] User Process starting - pid: 109
[Runtime] Set up communication with system controller(0,0)
124;#[Runtime] Receiver Thread pid: 111

[Runtime] User process Started
[Main - 16384] started
```

Ciò è dovuto al fatto che quando è attiva la console di Linux, viene disattivato il System Controller di i-droid che gestisce le funzioni avanzate del robot.

Allora si procede in questo modo:
1) rinominate file eseguibile MioProgramma in program1 (scritto in minuscolo) e assicuratevi che si trovi in /mnt/disk/programs. Per cambiare nome: mv MioProgramma program1

2) avviate il System Controller digitando /idroid/IDROID start
3) terminato il riavvio del System Controller, avviate il programma da voi creato utilizzando il tastierino oppure il PC-Control.

In pratica col Linux della Brain&Vision si ha questo comportamento: i programmi che non invocano le funzioni del robot (le librerie) come ad esempio test.c scritto in precedenza, possono essere compilati ed eseguiti immediatamente; quando si invocano le librerie del robot è necessario riavviare il System Controller.

NOTA. I programmi che gestiscono il robot devono necessariamente chiamarsi program1 perché questo è il nome del programma che i-droid cerca per mandarlo in esecuzione.

SEMPLIFICARE LA COMPILAZIONE

Che noia usare l'istruzione tcc -I/idroid/include -lpthread -lc-like -oMioProgramma MioProgramma.c; è troppo lunga e poi basta commettere un errore di ortografia che non funziona niente. Per semplificare la compilazione si ricorre allo script compile_up, che si trova sotto /idroid. Questo file non fa altro che richiamare automaticamente le librerie; necessita solo di due parametri che sono il nome del file da compilare (passato senza estensione) e il nome di un file (ci pensa lui a crearlo) nel quale inserire eventuali messaggi di debug.

Quindi basta che ci mettiamo nella directory /mnt/disk/programs e scriviamo:

```
# /idroid/compile_up  program1  build.log
```

build.log è un file di testo dove verranno inseriti eventuali errori di compilazione. Il nome di questo file può essere scelto a piacimento (con o senza l'estensione). Questa è la soluzione utilizzata dall'ambiente Visual C-like per compilare il programma.

E' interessante analizzare il contenuto del file compile_up:

```
#!/bin/sh
#echo S0 $0
#echo S1 $1
#echo S2 $2
echo compiling $1.c
echo redirecting output to $2
rm -f $1
tcc -I/idroid/include -lpthread -lc-like -o$1 $1.c 1>$2 2>$2
```

Le prime istruzioni che iniziando con il simobolo del cancelletto stanno ad indicare che questo file si avvale dei comandi della shell e che prenderà fino a tre parametri: $0 (il nome di se stesso, cioé compile_up), $1 (il primo parametro passato, cioé il nome del programma), $2 (il secondo parametro passato (cioé il build.log). L'istruzione echo compiling $1.c produce l'output testuale compiling program1.c; serve per avvisarci che ha trovato il file sorgente. L'istruzione echo redirecting output to $2 produce l'output testuale redirecting output to build.log; serve per avvisarci che inserirà le informazioni di debug nel file segnalato. L'istruzione rm -f $1 elimina il file eseguibile program1 (se esiste, è quello precedente). L'ultima istruzione effettua la compilazione ed è simile a quella vista in precedenza salvo il fatto che aggiunge alla fine il pezzo 1>$2 2>$2 che sta ad indicare che l'eventuale output testuale o eventuali errori saranno salvati nel file build.log.

Volendo semplificare ulteriormente la compilazione possiamo crearci uno script di questo tipo:

```
#!/bin/sh
#echo $1
/idroid/compile_up $1 $1.log
```

che non fa altro che invocare il file compile_up e creare il file di log con lo stesso nome del programma. In questo modo possiamo invocare il nostro script passandogli solo un parametro che è program1.

Se ad esempio lo script si chiama compila, basterà scrivere:

```
# ./compila program1
```

che è molto più semplice di:

```
# tcc -I/idroid/include -lpthread -lc-like -oMioProgramma MioProgramma.c
```

NOTA
Una volta creato un file script, occorre renderlo eseguibile; ad esempio per il file compila si scrive:
```
# chmod +x compila
```