

2020F_ESE_3014_1

SEMESTER: 3rd SEM

INSTRUCTOR: Prof. Linchen Wang

LAB 8

SUBMISSION DATE: 26 Nov, 2020

NAME AND ID:

VISHAL HASRAJANI (C0761544)

Goutham Reddy Alugubelly (C0747981)

PARTH PATEL (C0764929)

INTRODUCTION :

USB is a high-performance serial bus that is capable of transmitting with the transmission speed of hundreds or even thousands of megabits per second. It is also called an asynchronous system as it does not have the exact clock and it can work with up to 127 devices on a single bus.

In this lab we will show how we sent the string from beaglebone black (using the code recommended by derek molloy) and received the same string on our linux host machine(Checked it using minicom).Here we used TDI TTL-232R-3V3 USB to TTL serial cable.

DESCRIPTION:

To begin with, we enabled the UART1 of Beaglebone black as follows :

```
--> sudo nano /boot/uEnv.txt
```

Then here we made the changes in the uEnv.txt file to enable the UART1 of beaglebone.

```
debian@beaglebone: ~
File Edit View Search Terminal Help
GNU nano 3.2                                         /boot/uEnv.txt

#Docs: http://elinux.org/Beagleboard:U-boot_partitioning_layout_2.0

uname_r=4.19.94-ti-r42
#uuid=
#dtb=

###U-Boot Overlays###
###Documentation: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian#U-Boot_Overlays
###Master Enable
enable_uboot_overlays=1
###

###Override capes with eeprom
#uboot_overlay_addr0=/lib/firmware/<file0>.dtbo
#uboot_overlay_addr1=/lib/firmware/<file1>.dtbo
#uboot_overlay_addr2=/lib/firmware/<file2>.dtbo
#uboot_overlay_addr3=/lib/firmware/<file3>.dtbo
uboot_overlay_addr2=/lib/firmware/BB-UART1-00A0.dtbo

###

###Additional custom capes
#uboot_overlay_addr4=/lib/firmware/<file4>.dtbo
#uboot_overlay_addr5=/lib/firmware/<file5>.dtbo
#uboot_overlay_addr6=/lib/firmware/<file6>.dtbo
#uboot_overlay_addr7=/lib/firmware/<file7>.dtbo
###

###Custom Cape
#dtb_overlay=/lib/firmware/<file8>.dtbo
###

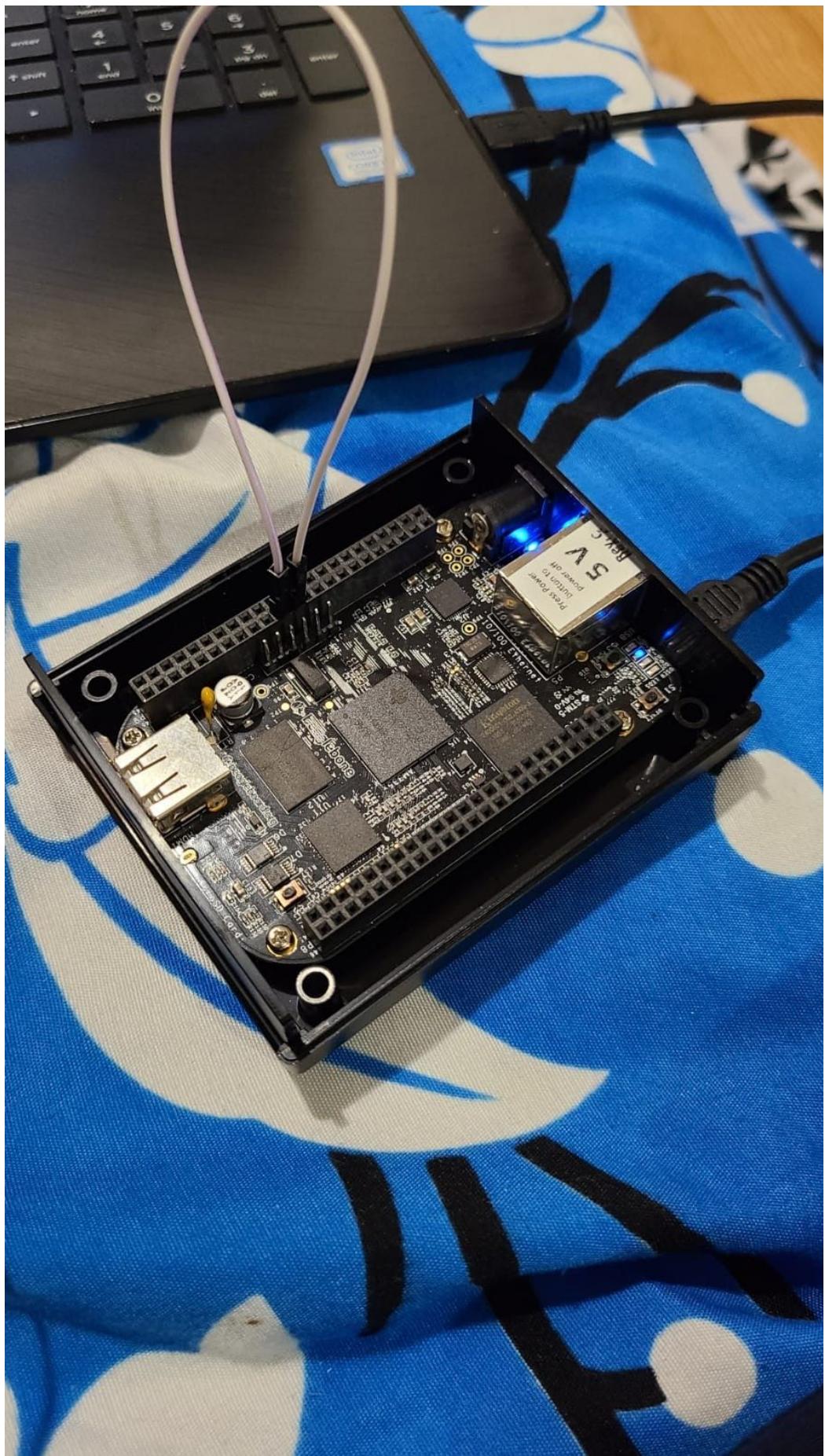
###Disable auto loading of virtual capes (emmc/video/wireless/adc)
#disable_uboot_overlay_emmc=1
disable_uboot_overlay_video=1
disable_uboot_overlay_audio=1
#disable_uboot_overlay_wireless=1
#disable_uboot_overlay_adc=1
###

###PRUSS OPTIONS
###pru_rproc (4.14.x-ti kernel)
#uboot_overlay_pru=/lib/firmware/AM335X-PRU-RPROC-4-14-TI-00A0.dtbo
###pru_rproc (4.19.x-ti kernel)
uboot_overlay_pru=/lib/firmware/AM335X-PRU-RPROC-4-19-TI-00A0.dtbo
###pru_uio (4.14.x-ti, 4.19.x-ti & mainline/bone kernel)
#uboot_overlay_pru=/lib/firmware/AM335X-PRU-UIO-00A0.dtbo
###

###Cape Universal Enable
enable_uboot_cape_universal=1
cape_enable=bone_capemgr.enable_partno=BB-UART1
| 

###Debug: disable uboot autoload of Cape
```

Then using the loop back(we connected 24 and 26 pins of beaglebone black).



Further using the minicom,we tested whether the UART1 is working on beaglebone or not with the help of following command :

```
minicom -b 9600 -o -D /dev/tty01
```

```
debian@beaglebone: ~
File Edit View Search Terminal Help

Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on May  6 2018, 10:36:56.
Port /dev/tty01, 18:01:38

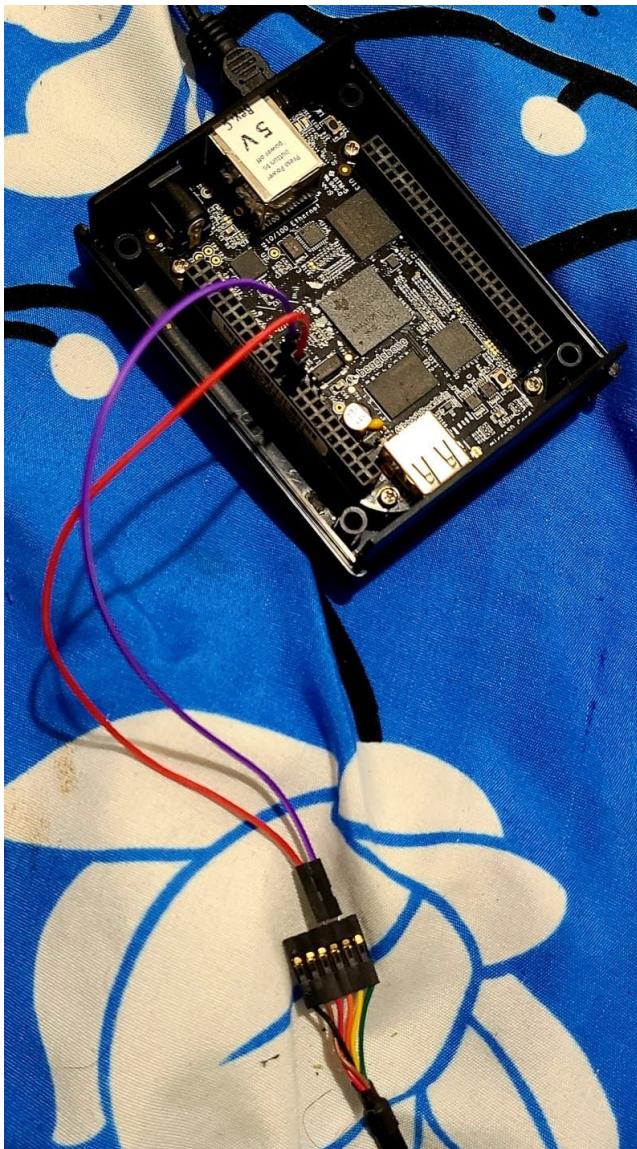
Press CTRL-A Z for help on special keys

eewwwwwttyy
```

So here you can see that as we typed the letter e,w,w,t and y , we got back the same result in return .

So Output confirms that the UART1 is enabled properly and its working.

Connections on Beaglebone Black(UART1 port) with FTDI cable



Here only RX and TX connections are used at both the ends.

After doing this we checked whether our FTDI cable is detected or not using the command :

```
dmesg | tail
```

```
star@vishal-hp-laptop-15-bs1xx:~
```

File Edit View Search Terminal Help

```
star@vishal-hp-laptop-15-bs1xx:~$ dmesg | tail
[ 3986.680314] usbcore: registered new interface driver usbserial_generic
[ 3986.680343] usbserial: USB Serial support registered for generic
[ 3986.689387] usbcore: registered new interface driver ftdi_sio
[ 3986.689431] usbserial: USB Serial support registered for FTDI USB Serial Device
[ 3986.689598] ftdi_sio 1-1:1.0: FTDI USB Serial Device converter detected
[ 3986.689729] usb 1-1: Detected FT232RL
[ 3986.690374] usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB0
[ 3995.756283] usb 1-1: USB disconnect, device number 4
[ 3995.758046] ftdi_sio ttyUSB0: FTDI USB Serial Device converter now disconnected from ttyUSB0
[ 3995.758154] ftdi_sio 1-1:1.0: device disconnected
star@vishal-hp-laptop-15-bs1xx:~$ dmesg | tail
[ 3995.758154] ftdi_sio 1-1:1.0: device disconnected
[ 4022.135029] usb 1-1: new full-speed USB device number 5 using xhci_hcd
[ 4022.288072] usb 1-1: New USB device found, idVendor=0403, idProduct=6001, bcdDevice= 6.00
[ 4022.288080] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 4022.288084] usb 1-1: Product: TTL232R-3V3
[ 4022.288087] usb 1-1: Manufacturer: FTDI
[ 4022.288090] usb 1-1: SerialNumber: FTBI4CBL
[ 4022.291198] ftdi_sio 1-1:1.0: FTDI USB Serial Device converter detected
[ 4022.291338] usb 1-1: Detected FT232RL
[ 4022.292170] usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB0
star@vishal-hp-laptop-15-bs1xx:~$
```

Protocol used :UART

Maximum data rate : 115200

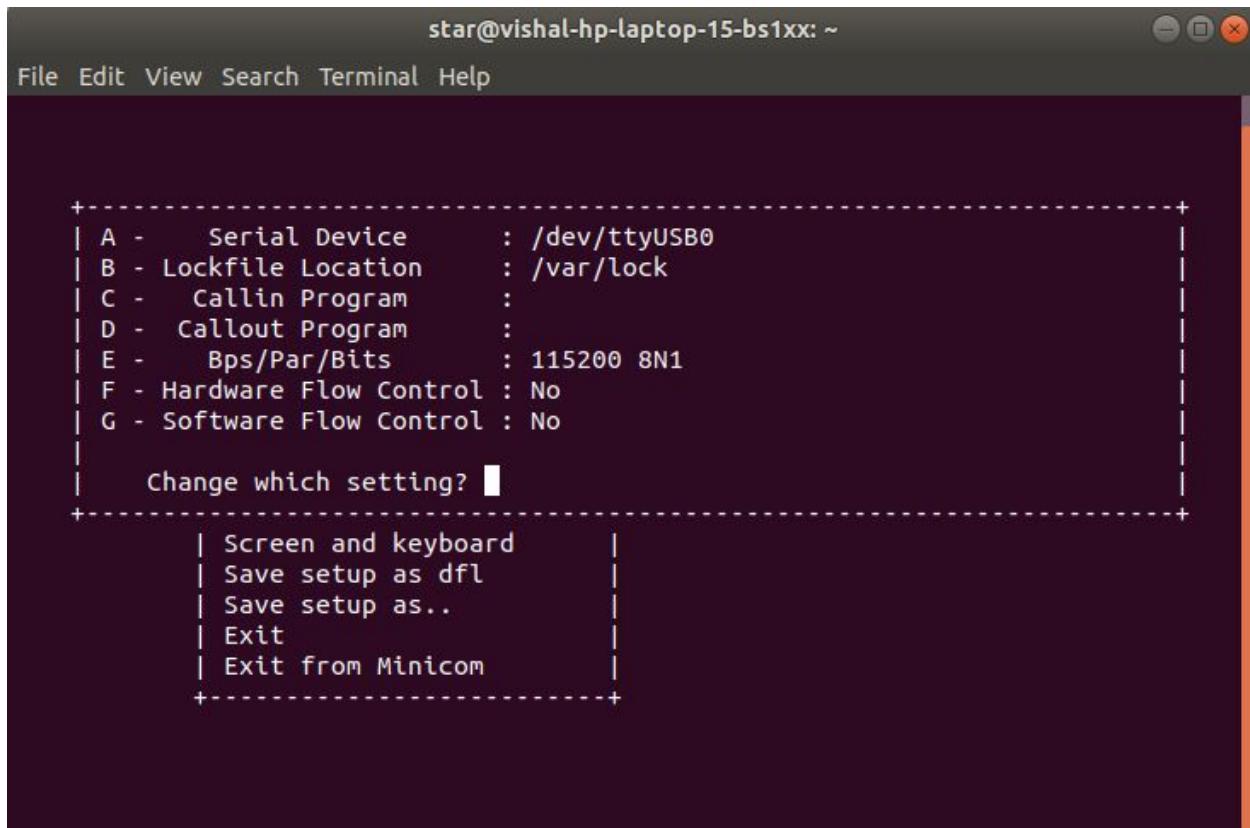
Then after we are done with the previous steps we setup the device on the minicom setup using terminal code:

```
minicom -s
```

The new device created here is ttyUSB0.

It is a serial device

The one limitation in USB devices over VCP is that we cannot see the bootup process using a USB cable.



Boot up process using VCP :

```
debian@beaglebone:~$ sudo reboot
[sudo] password for debian:
Sorry, try again.
[sudo] password for debian:
[ 147.630152] watchdog: watchdog0: watchdog did not stop!
[ 147.865387] reboot: Restarting system

U-Boot SPL 2018.09-00002-g0b54a51eee (Sep 10 2018 - 19:41:39 -0500)
Trying to boot from MMC2
Loading Environment from EXT4... Card did not respond to voltage select!

U-Boot 2018.09-00002-g0b54a51eee (Sep 10 2018 - 19:41:39 -0500), Build: jenkins-github_Bootloader-Builder-65

CPU   : AM335X-GP rev 2.1
I2C:   ready
DRAM:  512 MiB
No match for driver 'omap_hsmmc'
No match for driver 'omap_hsmmc'
Some drivers were not found
Reset Source: Global warm SW reset has occurred.
Reset Source: Power-on reset has occurred.
RTC 32KCLK Source: External.
MMC:  OMAP SD/MMC: 0, OMAP SD/MMC: 1
Loading Environment from EXT4... Card did not respond to voltage select!
Board: BeagleBone Black
<ethaddr> not set. Validating first E-fuse MAC
BeagleBone Black:
BeagleBone: cape eeprom: i2c_probe: 0x54:
BeagleBone: cape eeprom: i2c_probe: 0x55:
BeagleBone: cape eeprom: i2c_probe: 0x56:
BeagleBone: cape eeprom: i2c_probe: 0x57:
Net:   eth0: MII MODE
Could not get PHY for cpsw: addr 0
cpsw, usb_ether
Press SPACE to abort autoboot in 2 seconds
```

Then we used the code that is recommended by derek molloy that is **uart.c** on our beaglebone to send the strings to our linux host machine using the UART1 port .

Code :

uart.c

```
/* Simple send message example for communicating with the
 * UART that is connected to a desktop PC. */

#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>
#include<termios.h>
#include<string.h>

int main(int argc, char *argv[]){
    int file, count;
    if(argc!=2){
        printf("Please pass a message string to send,
exiting!\n");
        return -2;
    }
    if ((file = open("/dev/tty01", O_RDWR | O_NOCTTY |
O_NDELAY))<0){
        perror("UART: Failed to open the device.\n");
        return -1;
    }
    struct termios options;
    tcgetattr(file, &options);
    options.c_cflag = B115200 | CS8 | CREAD | CLOCAL;
    options.c_iflag = IGNPAR | ICRNL;
```

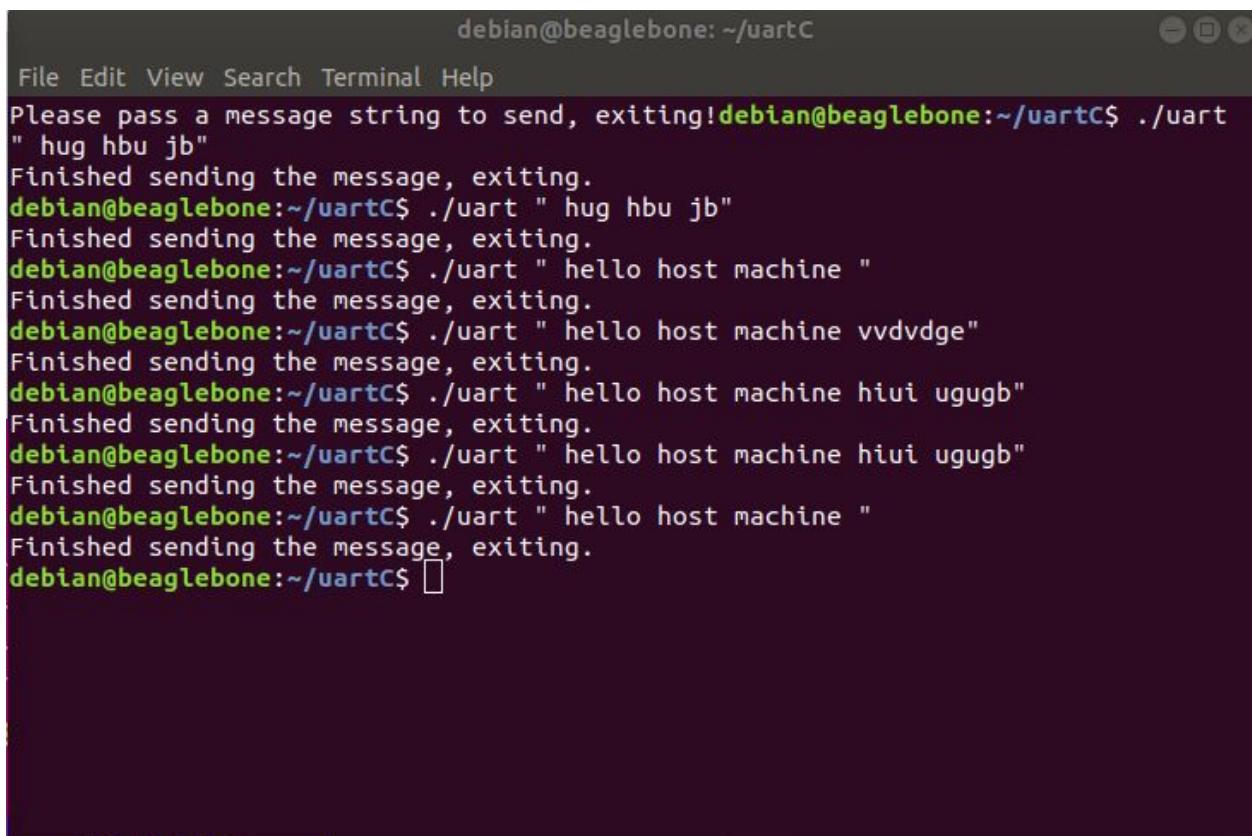
```

tcflush(file, TCIFLUSH);
tcsetattr(file, TCSANOW, &options);

// send the string plus the null character
if ((count = write(file, argv[1],
strlen(argv[1])+1))<0){
    perror("UART: Failed to write to the output.\n");
    return -1;
}
close(file);
printf("Finished sending the message, exiting.\n");
return 0;
}

```

Output (Sending the string) : "Hello host machine "



```

debian@beaglebone: ~/uartC
File Edit View Search Terminal Help
Please pass a message string to send, exiting!debian@beaglebone:~/uartC$ ./uart
" hug hbu jb"
Finished sending the message, exiting.
debian@beaglebone:~/uartC$ ./uart " hug hbu jb"
Finished sending the message, exiting.
debian@beaglebone:~/uartC$ ./uart " hello host machine "
Finished sending the message, exiting.
debian@beaglebone:~/uartC$ ./uart " hello host machine vvdvdge"
Finished sending the message, exiting.
debian@beaglebone:~/uartC$ ./uart " hello host machine hiui ugugb"
Finished sending the message, exiting.
debian@beaglebone:~/uartC$ ./uart " hello host machine hiui ugugb"
Finished sending the message, exiting.
debian@beaglebone:~/uartC$ ./uart " hello host machine "
Finished sending the message, exiting.
debian@beaglebone:~/uartC$ 

```

Received string on linux host machine:(using minicom)

```
star@vishal-hp-laptop-15-bs1xx: ~
File Edit View Search Terminal Help

Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyUSB0, 17:26:04

Press CTRL-A Z for help on special keys

hello host machine █
```

Youtube Link :

<https://www.youtube.com/watch?v=RTp7T6Ks7el&feature=youtu.be>

CONCLUSION :

Overall we learned USB to TTL Serial cable can be used to see whether communication between BBB and Host machine is proper or not, by sending a series of strings using UART protocol.