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.

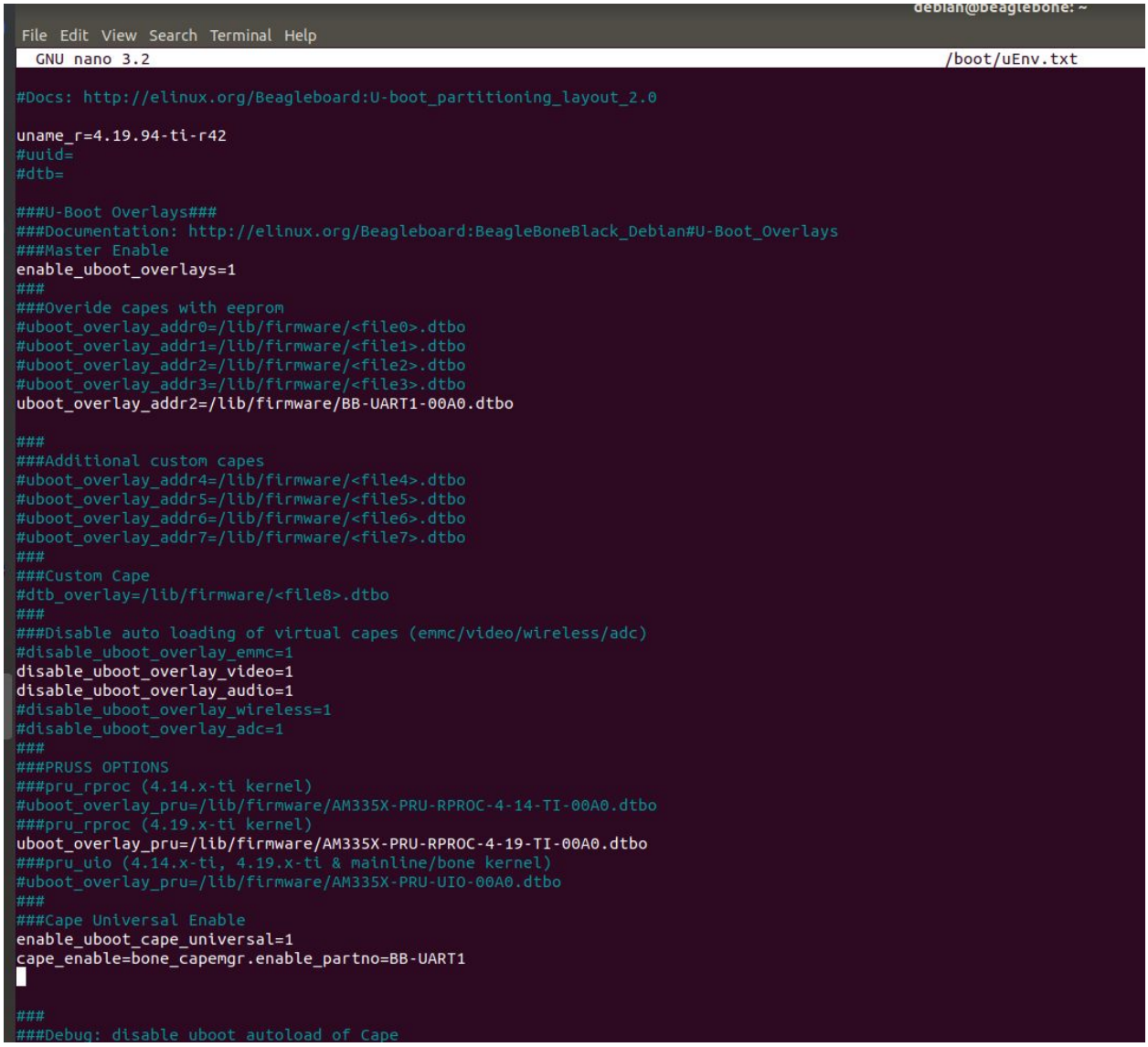
In this lab we will show how we sent the string from beaglebone black (using the code recommended by derek molloy) and received the same string on our linux host machine(Checked it using minicom).Here we used TDI TTL-232R-3V3 USB to TTL serial cable.

DESCRIPTION:

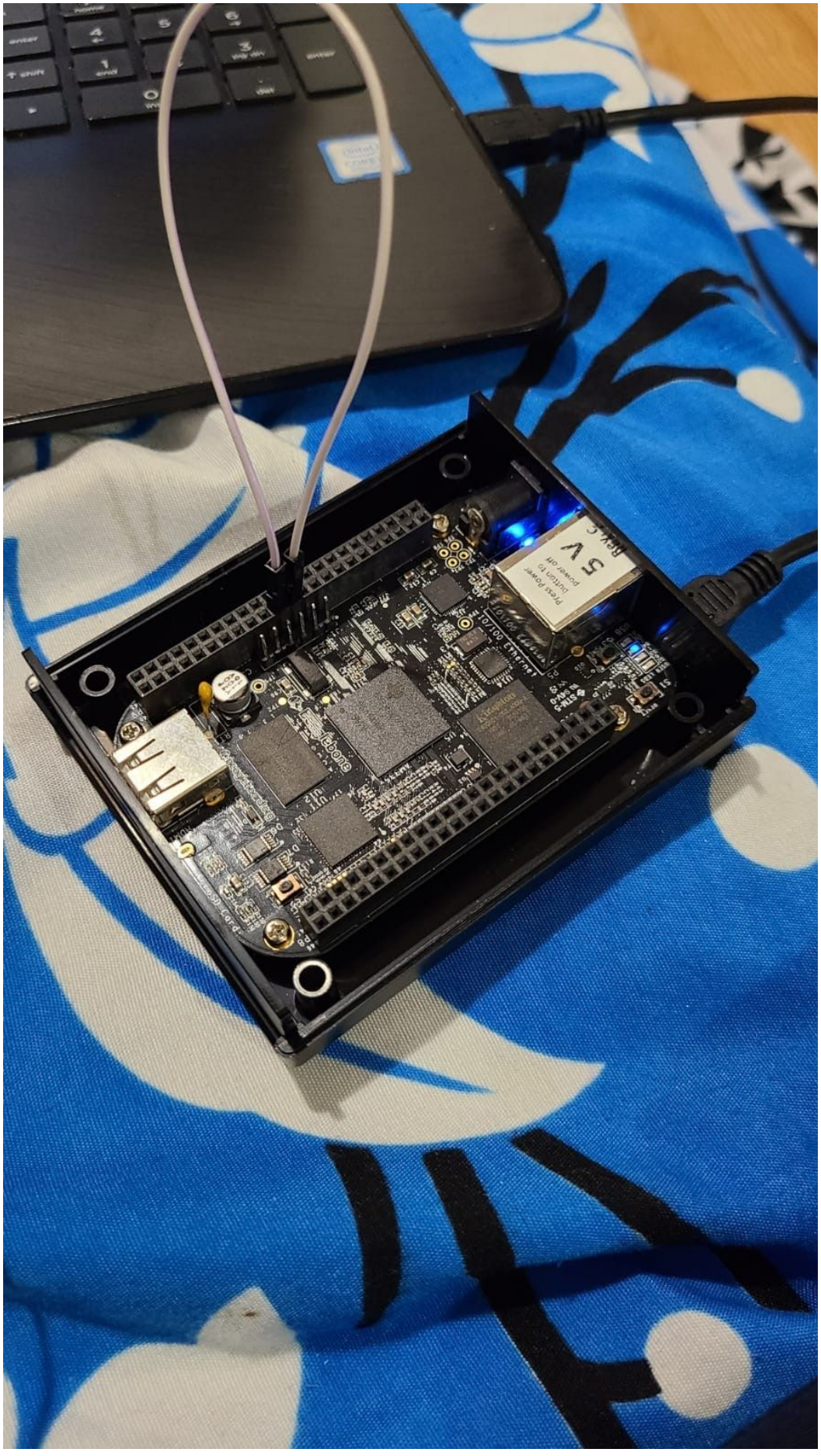
To begin with, we enabled the UART1 of Beaglebone black as follows :

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

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

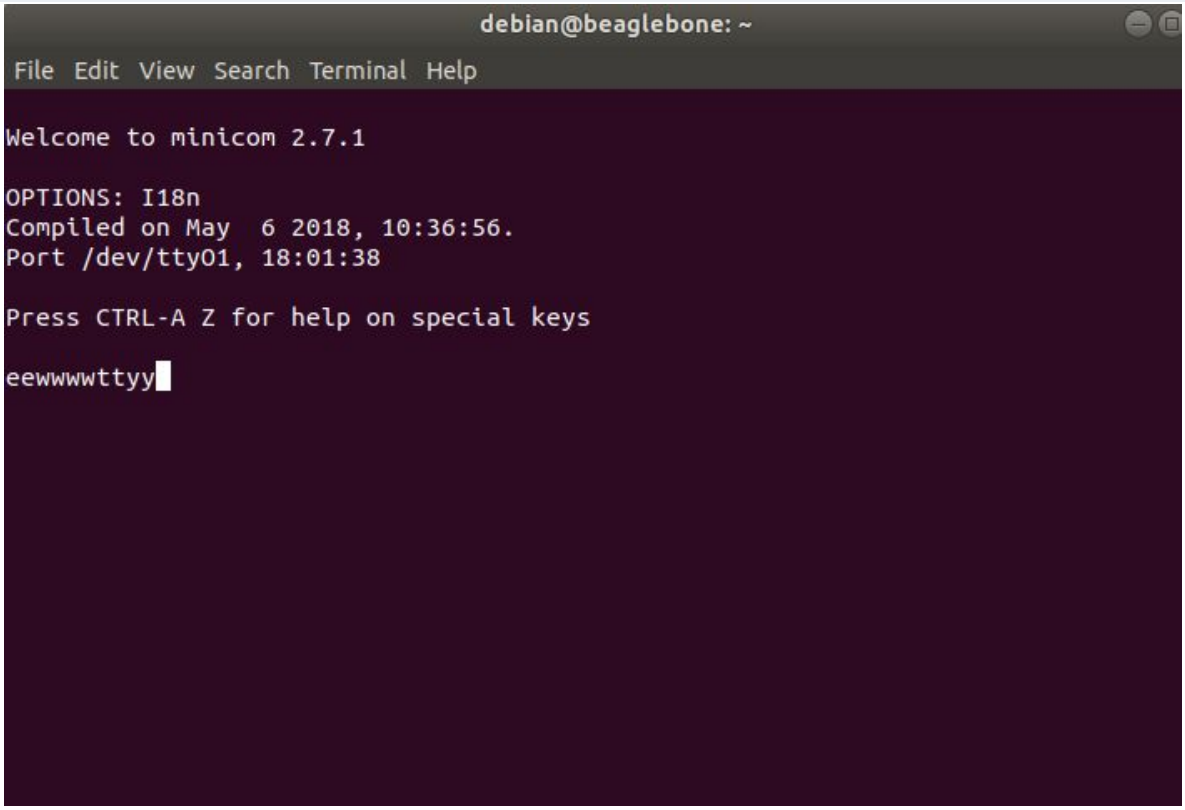


Then using the loop back(we connected 24 and 26 pins of beaglebone black).


5V
Press Power
button to
power on

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

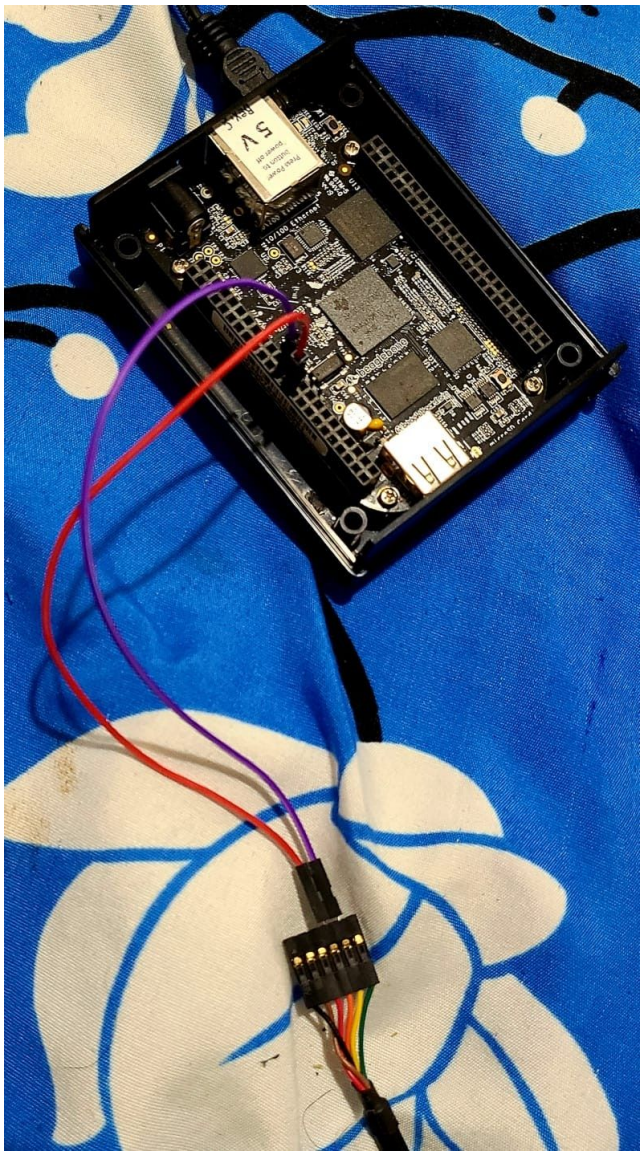
```
minicom -b 9600 -o -D /dev/ttyO1
```



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 .

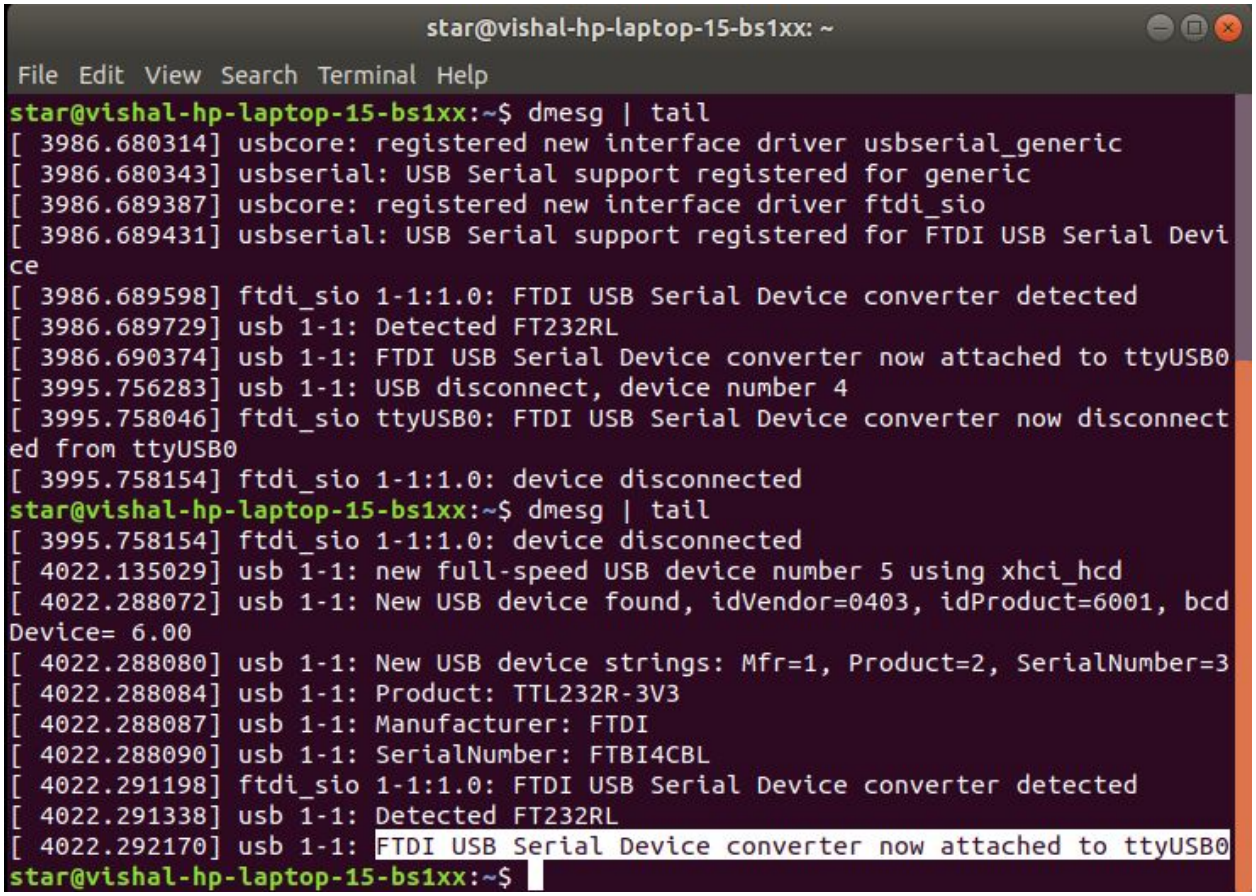
So Output confirms that the UART1 is enabled properly and its working.

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
```

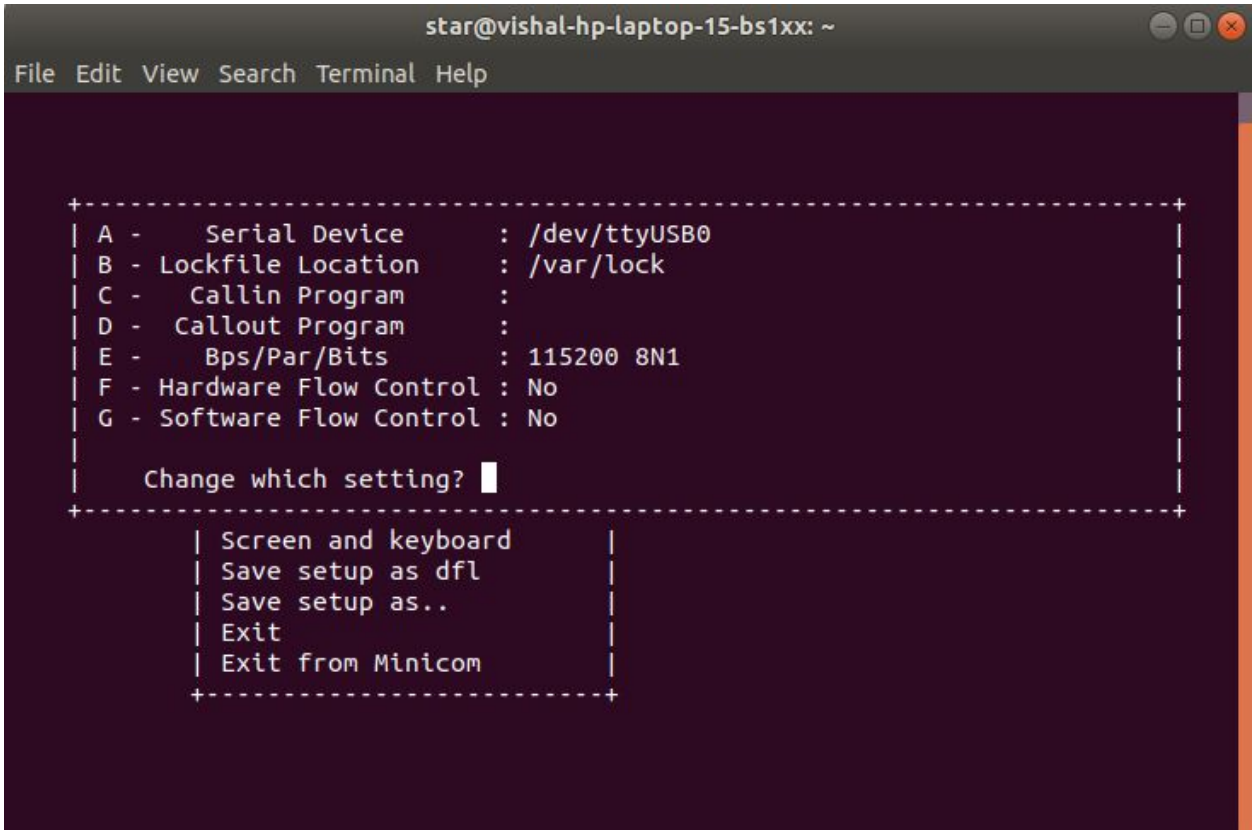


Protocol used :UART
Maximum data rate : 115200

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 .

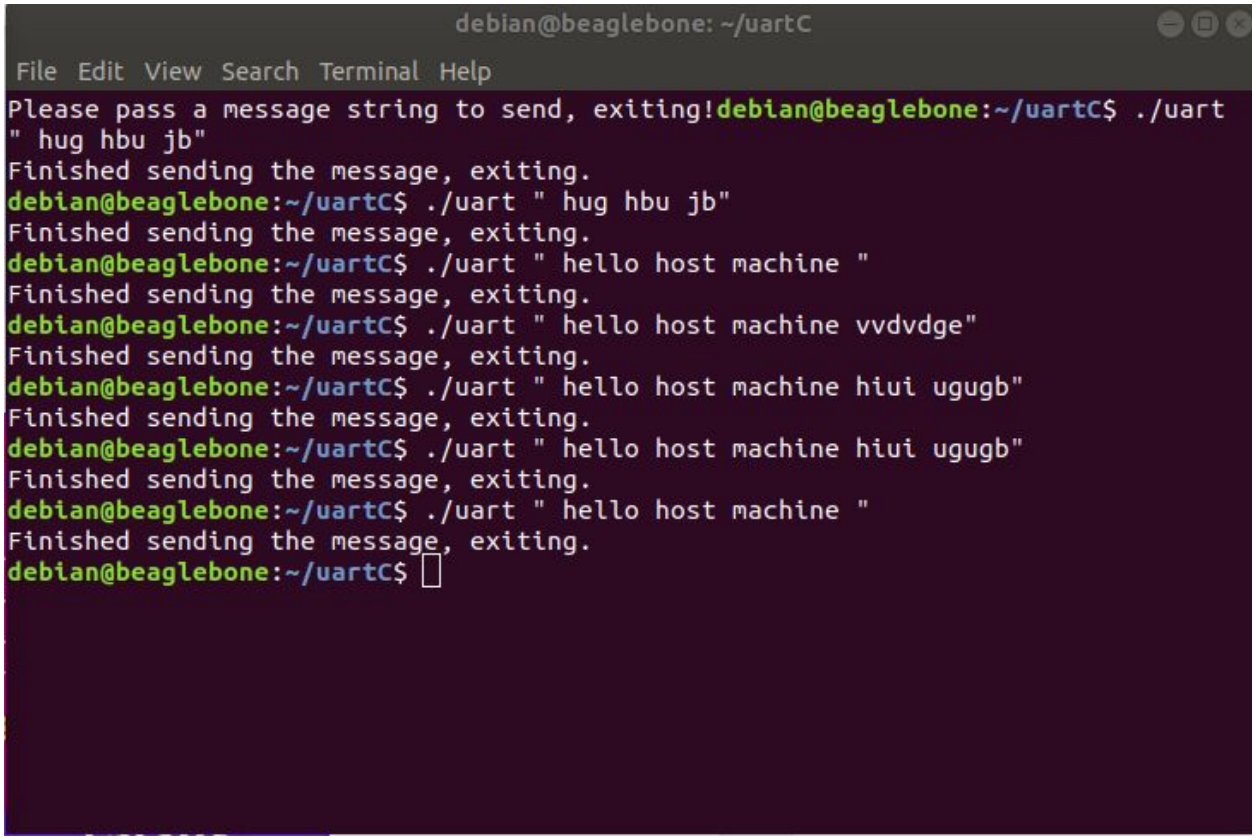
Code :

uart.c

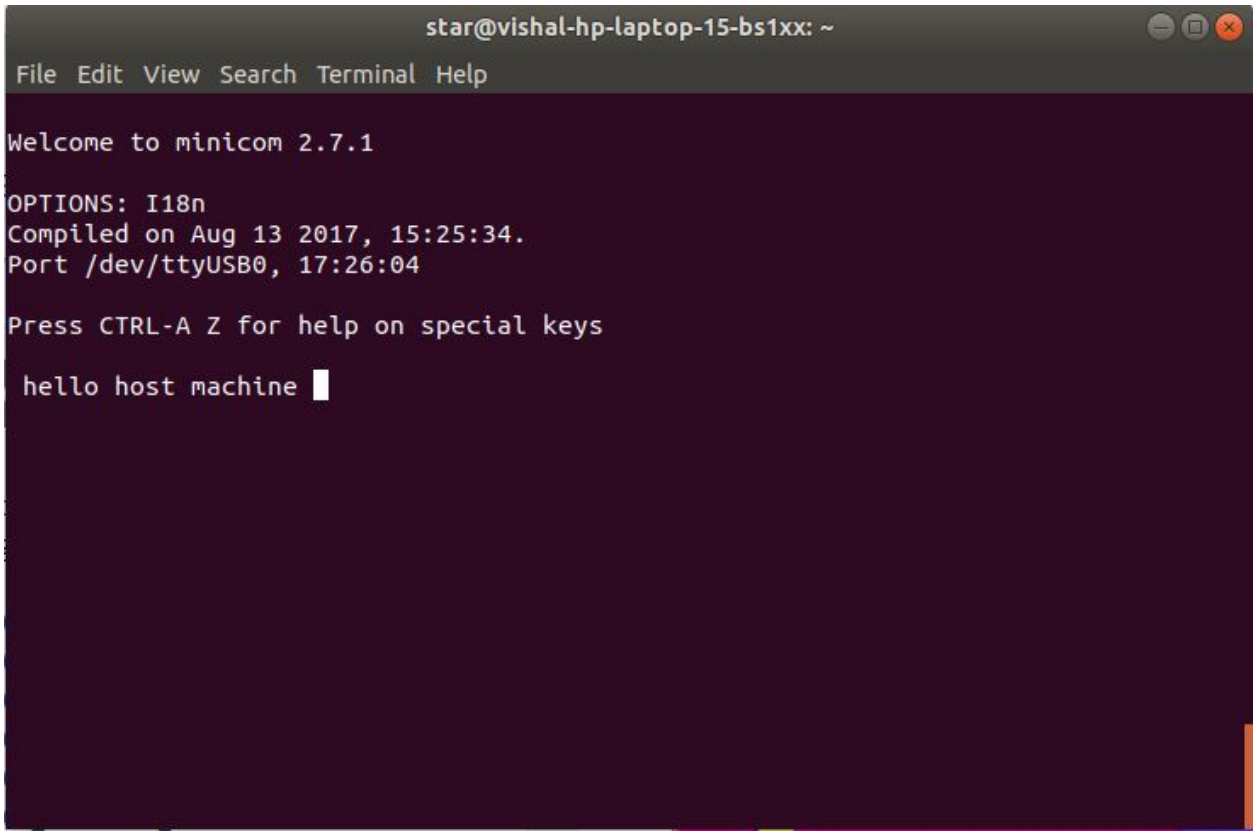
```
/* Simple send message example for communicating with the
*  UART that is connected to a desktop PC.  */

#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/ttyO1", 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 “



Received string on linux host machine:(using minicom)



Youtube Link :

https://www.youtube.com/watch?v=RTp7T6Ks7eI&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.