

**2020F\_ESE\_3014\_1**

**SEMESTER: 3<sup>rd</sup> SEM**

**INSTRUCTOR: Prof. Linchen Wang**

**LAB 6**

**SUBMISSION DATE: 12 Nov, 2020**

**NAME AND ID:**

**VISHAL HASRAJANI (C0761544)**

**Goutham Reddy Alugubelly (C0747981)**

**PARTH PATEL (C0764929)**

## **INTRODUCTION :**

UART is also known as Universal Asynchronous Receiver/Transmitter. It is used to transfer the data serially , one bit at a time, between two electronic devices. UARTs were originally standalone ICs, but now we majorly integrate it with the host microprocessor/microcontroller.

It is called as asynchronous as the sender does not have to send a clock signal to synchronize the transmission. Here in this communication protocol start and stop bits are used to synchronize the transmission of data.

## **DESCRIPTION :**

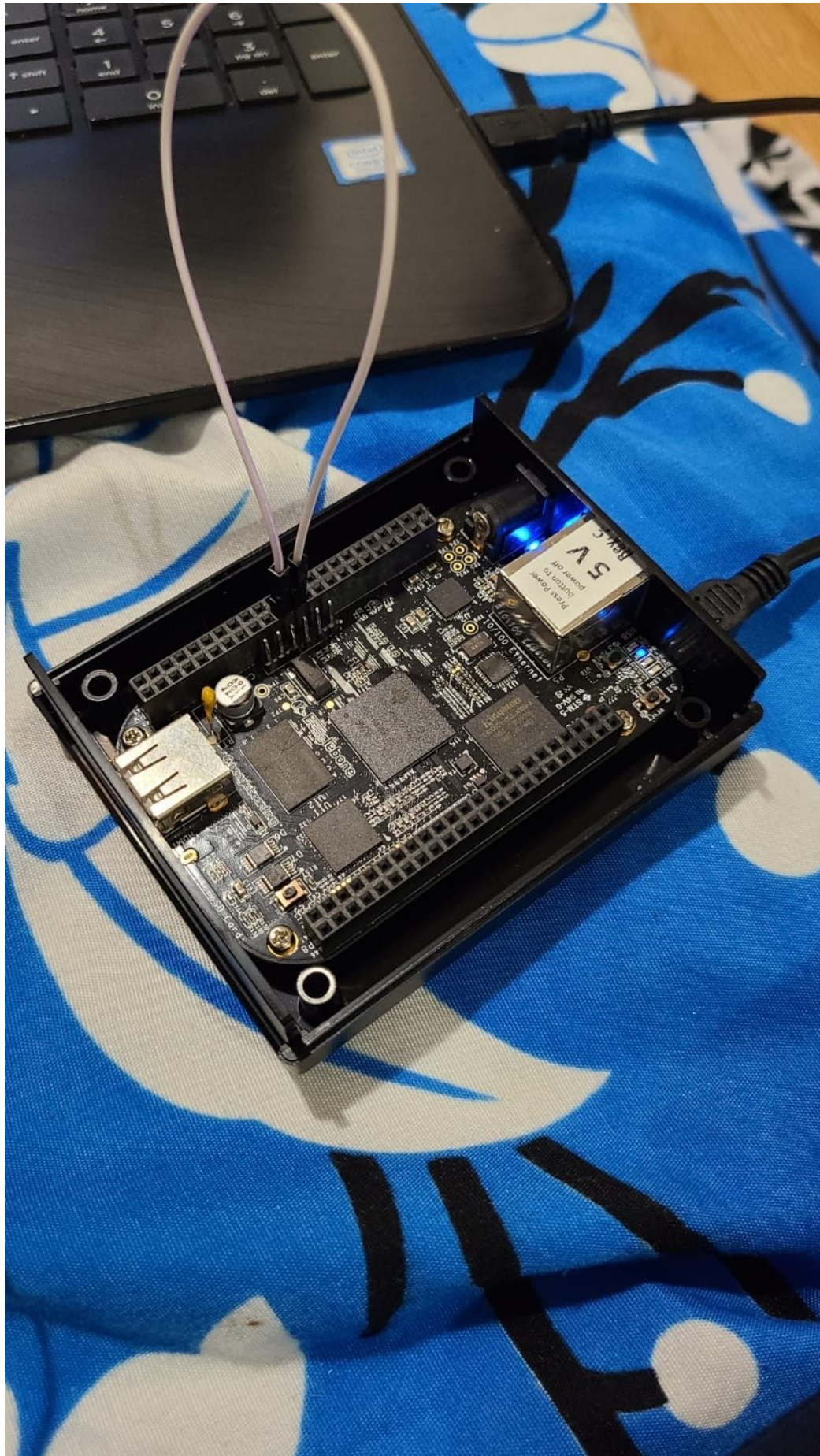
To begin with, we enabled the UART1 of Beaglebone black as follows :

```
--> 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/adx)  
#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).

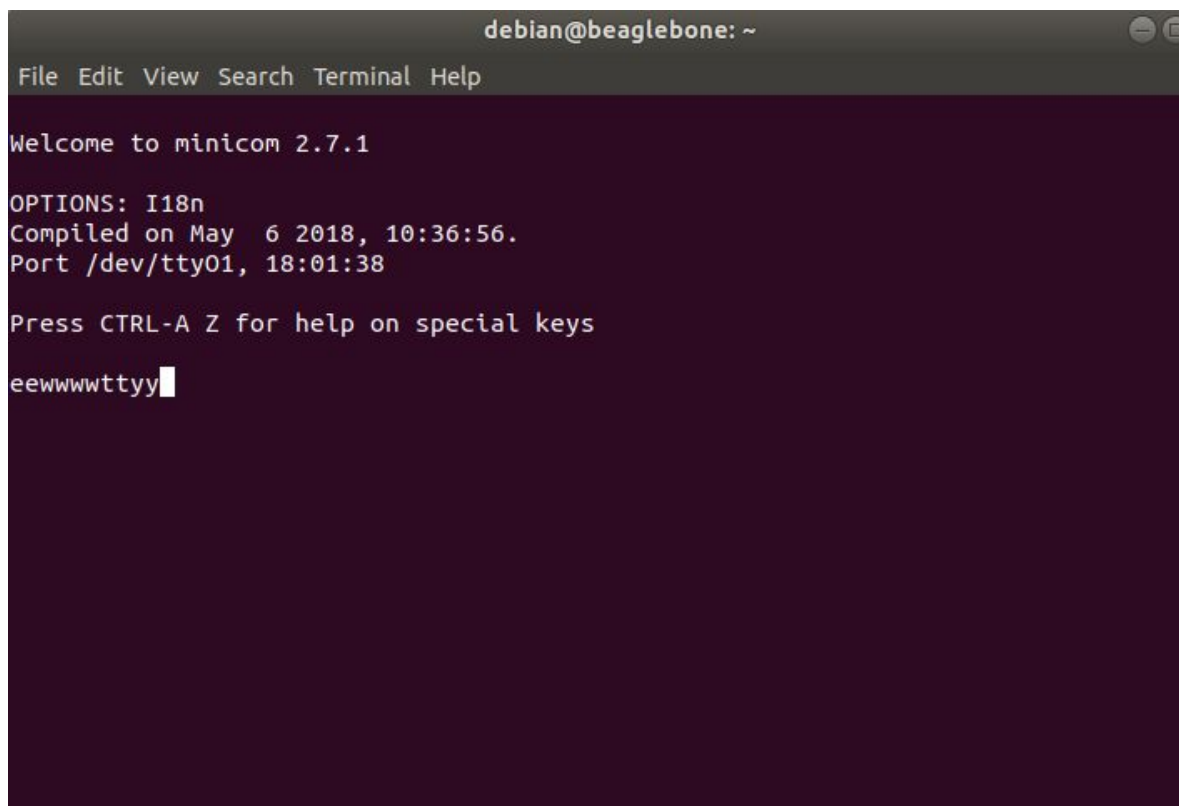




Further using the minicom, we tested whether the UART1 is working on beaglebone or not with the help of following command :

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

Output :

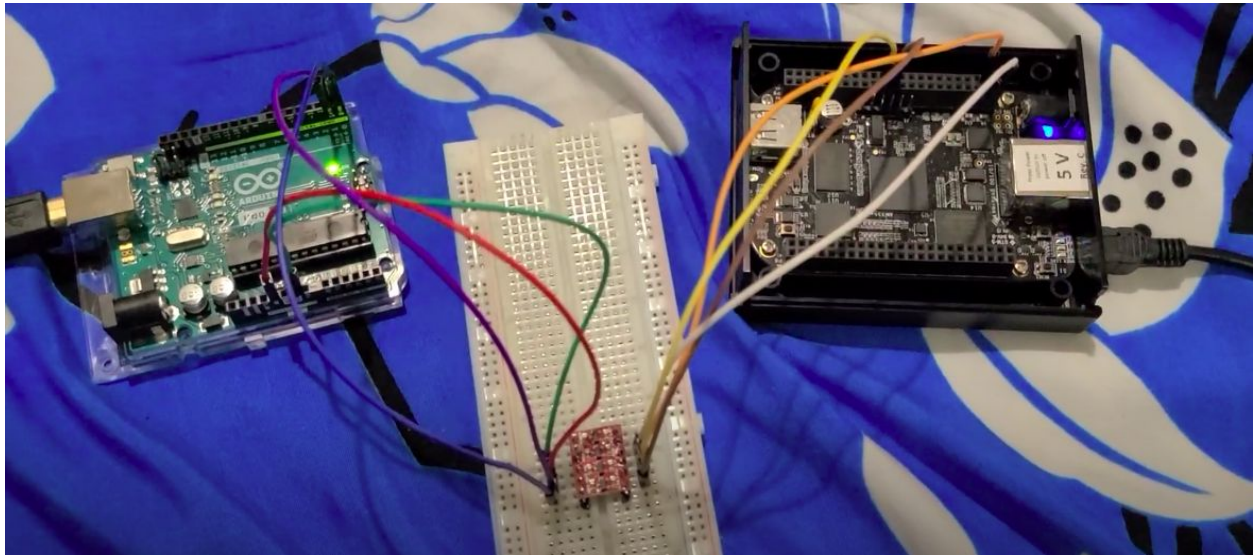
A screenshot of a terminal window titled 'debian@beaglebone: ~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal output shows: 'Welcome to minicom 2.7.1', 'OPTIONS: I18n', 'Compiled on May 6 2018, 10:36:56.', 'Port /dev/tty01, 18:01:38', and 'Press CTRL-A Z for help on special keys'. At the bottom, the text 'eeewwwtty' is displayed with a cursor at the end, indicating the characters typed during the test.

```
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  
eeewwwtty
```

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 between beaglebone black and Arduino Uno



Now coming to Arduino uno, we wrote a testing program to check whether the communication is proper or not .

We wrote the following program in arduino ide.

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  Serial.print("Test");  
  delay(1000);  
}
```

**Output on the beaglebone :**

Follow the youtube link below :

<https://www.youtube.com/watch?v=ul5-lzUsYyU>

After getting the clear idea that the communication is done properly, we are writing a program to flash the led at different frequencies (i.e having different delay values).

### arduino ide program :

```
char x;
void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600); //setting up baud rate to 9600
    pinMode(LED_BUILTIN,OUTPUT); //setting on board led of
    arduino as output

}

void loop() {

    //Serial.print("Test");

    if(Serial.available()) //checking whether the
    communication is there or not
    {

        x=Serial.read(); //reading the data from the
        beaglebone and saving that value in x.
        if(x=='1') //if x == 1 then the delay will be 100
        ms
        {

            digitalWrite(LED_BUILTIN,HIGH); // LED ON
```

```

    delay(100); //delay of 100ms(milliseconds)
    digitalWrite(LED_BUILTIN,LOW); //LED OFF
    delay(100);
    digitalWrite(LED_BUILTIN,HIGH);
    delay(100);
    digitalWrite(LED_BUILTIN,LOW);
    delay(100);
    digitalWrite(LED_BUILTIN,HIGH);
    delay(100);
    digitalWrite(LED_BUILTIN,LOW);
    delay(100);
    digitalWrite(LED_BUILTIN,HIGH);
    delay(100);
    digitalWrite(LED_BUILTIN,LOW);
    delay(100);

}

else if(x=='2') //if x == 2 then the delay will be
1000 ms
{

    digitalWrite(LED_BUILTIN,HIGH);
    delay(1000);

```



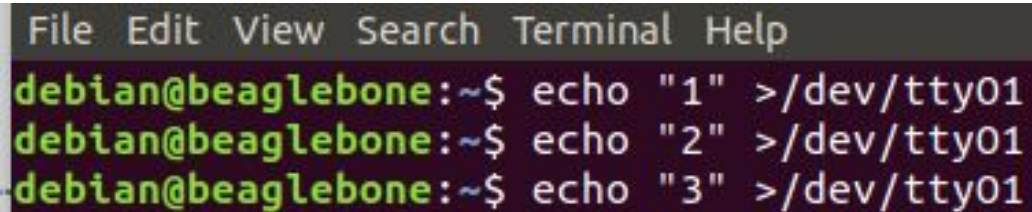
```
digitalWrite(LED_BUILTIN,LOW);
delay(1000);
digitalWrite(LED_BUILTIN,HIGH);
delay(1000);
digitalWrite(LED_BUILTIN,LOW);
delay(1000);
    digitalWrite(LED_BUILTIN,HIGH);
delay(1000);
digitalWrite(LED_BUILTIN,LOW);
delay(1000);
}
```

```
    else if(x=='3') //if x == 3 then the delay will
be 500 ms
{
```

```
digitalWrite(LED_BUILTIN,HIGH);
delay(500);
digitalWrite(LED_BUILTIN,LOW);
delay(500);
digitalWrite(LED_BUILTIN,HIGH);
delay(500);
digitalWrite(LED_BUILTIN,LOW);
delay(500);
    digitalWrite(LED_BUILTIN,HIGH);
delay(500);
digitalWrite(LED_BUILTIN,LOW);
delay(500);
digitalWrite(LED_BUILTIN,HIGH);
delay(500);
```

```
    digitalWrite(LED_BUILTIN, LOW);  
    delay(500);  
    digitalWrite(LED_BUILTIN, HIGH);  
    delay(500);  
    digitalWrite(LED_BUILTIN, LOW);  
    delay(500);  
  
}  
    Serial.println(x); //printing the value of x on the  
serial command window  
}  
}
```

### Beaglebone commands:

A terminal window with a dark background and light-colored text. The menu bar at the top shows 'File Edit View Search Terminal Help'. The prompt is 'debian@beaglebone:~\$'. Three lines of commands are shown: 'echo "1" >/dev/tty01', 'echo "2" >/dev/tty01', and 'echo "3" >/dev/tty01'.

```
File Edit View Search Terminal Help  
debian@beaglebone:~$ echo "1" >/dev/tty01  
debian@beaglebone:~$ echo "2" >/dev/tty01  
debian@beaglebone:~$ echo "3" >/dev/tty01
```

**OUTPUT:** FOLLOW THE YOUTUBE LINK BELOW---

[https://youtu.be/XF1\\_gUbcncM](https://youtu.be/XF1_gUbcncM)

## **CONCLUSION:**

To conclude , we can say that,in this lab we successfully controlled our Arduino on-board led frequency using beaglebone black.