# 2020F_ESE 3014_1

**SEMESTER:** 3rd

**INSTRUCTOR:** Linchen Wang

**LAB:** 6

**DUE DATE:** 15 Nov 2020 - 23:55

**SUBMITTED DATE:** 13 NOV 2020

**STUDENTS:** Amandeep Singh (C0765434)

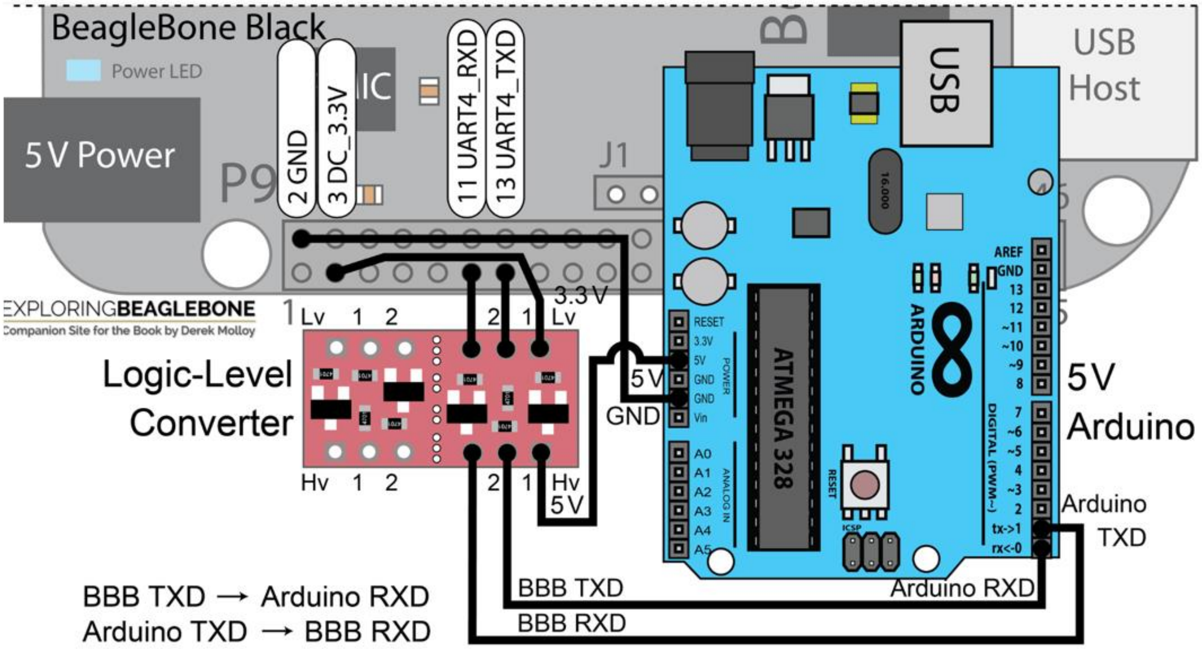
**Lab-6**

## INTRODUCTION

In this Lab we are going to setup interfacing, with the BBB as master and Arduino as slave using UART.

## DESCRIPTION

- Pin Configuration:
  We use the following pin configuration for setting up communication between Arduino and BBB.

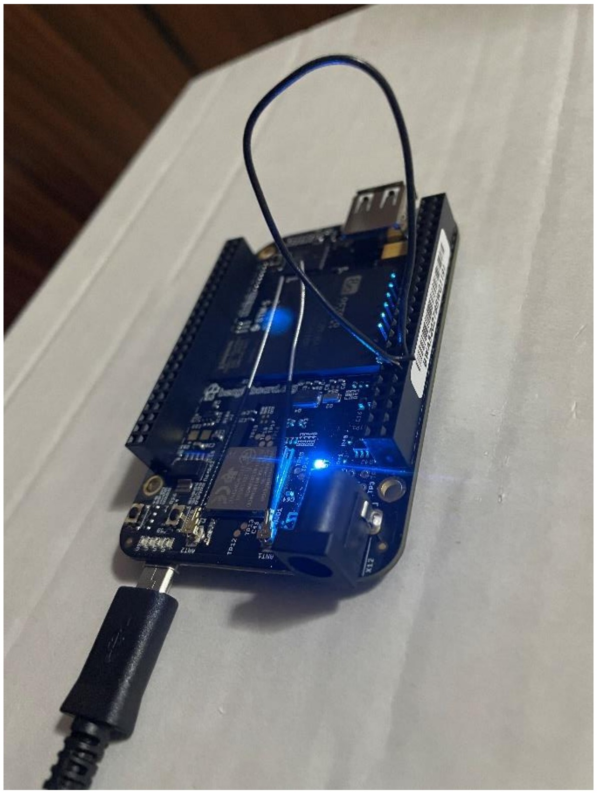


1. **Test your UART on BBB by loopback TXD and RXD as our slides**

- In order to enable ttyO4 we put the following commands in the uEnv.txt file which can be found inside the boot folder.

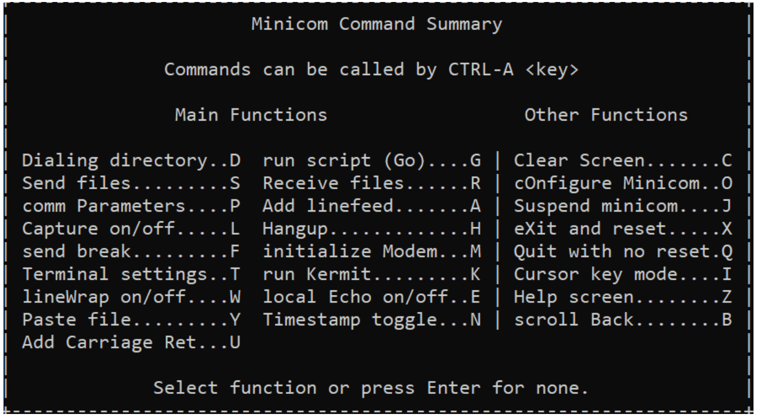
  ➔ Uboot_overlay_addr2=/lib/firmware/BB-UART4-00A0.dtbo
  ➔ Cape_enable=bone_capemgr.enable_partno=BB-UART4
  **NOTE:** after the uEnv.txt file is saved the BBB has to be restarted for the changes to take effect.

- Now to check whether our UART is working or not we connect a jumper wire between the pins 11 and 13, once the connections are made, we use the following command to operate the minicom.



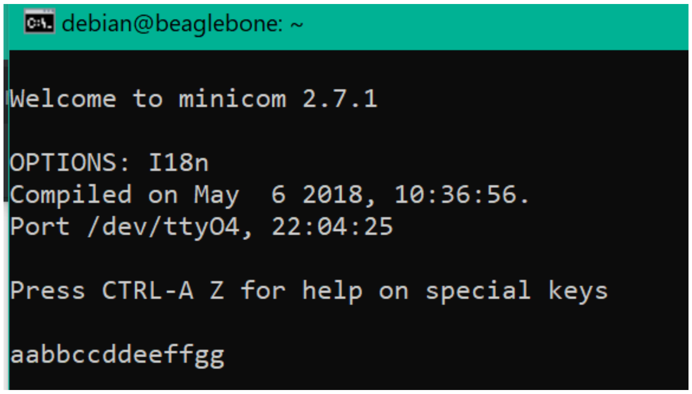
➔ $ minicom -b 115200  -o -D /dev/ttyO4

**Note:** this opens the minicom window, once the window is open, we need to turn on local Echo, which can be done by pressing “ctrl+a” and then “z”, a window will open and you just have to press ‘E’ to enable local Echo.



```
                        Minicom Command Summary

                  Commands can be called by CTRL-A <key>

               Main Functions                  Other Functions

 Dialing directory..D  run script (Go)....G | Clear Screen.......C
 Send files.........S  Receive files......R | cOnfigure Minicom..O
 comm Parameters....P  Add linefeed.......A | Suspend minicom....J
 Capture on/off.....L  Hangup.............H | eXit and reset.....X
 send break.........F  initialize Modem...M | Quit with no reset.Q
 Terminal settings..T  run Kermit.........K | Cursor key mode....I
 lineWrap on/off....W  local Echo on/off..E | Help screen........Z
 Paste file.........Y  Timestamp toggle...N | scroll Back........B
 Add Carriage Ret...U

                 Select function or press Enter for none.
```

- If ttyO4 turned on properly and the connections are snug we should see that the output is equal to the input, meaning whatever we type on the terminal is printed twice



```
Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on May  6 2018, 10:36:56.
Port /dev/ttyO4, 22:04:25

Press CTRL-A Z for help on special keys

aabbccddeeffgg
```

2. **Set up and achieve UART interfacing communication between BBB and Arduino. You can write a program to control the on-board LED of Arduino input and response the feedback as output.**

- To achieve the above objective we need to code the Arduino and beaglebone separately.

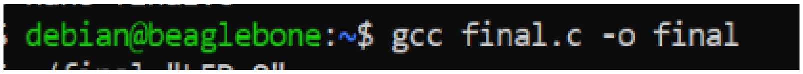
  ➔ **Starting with Arduino**

```
int ledPin = 13; // LED used for the brightness control
void setup()
    {
        Serial.begin(115200, SERIAL_8N1); //here 115200 is the baud rate.
        pinMode(ledPin, OUTPUT);
    }
void loop()
    {
        String command;
        char buffer[100];
        if ( Serial.available () > 0 )
            {
                command = Serial.readStringUntil('\0'); //reads until null
                                                        //character is reached.
                if( command.substring ( 0,4 ) == "LED " )
                {
                String intString = command.substring( 4, command.length () );
                int level = intString.toInt ();
                    if ( level >= 0 && level <= 255 ) //max Brightness = 255,
                                                      //min Brightness = 0
                    {
                    analogWrite ( ledPin, level ); //sets the brightness equal
                                                   // to the value of level
                    sprint ( buffer, " Set brightness to %d ", level );
                    }
                }
                Else
                {
                Sprint ( buffer, " Unknown command: %s ", command.c_str() );
                }
                Serial.print ( buffer );
            }
    }
```

- ➔ **Code for the BBB:**

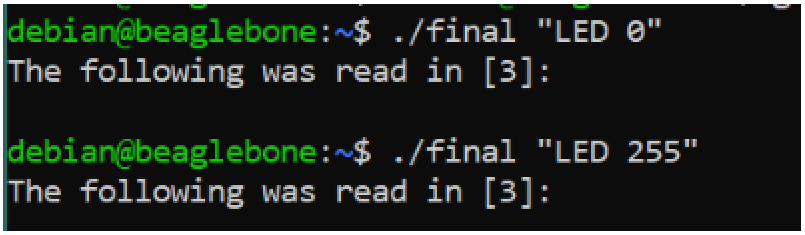
Note: please navigate to the appendix to see the code.

- **Creating the object file of final.c**



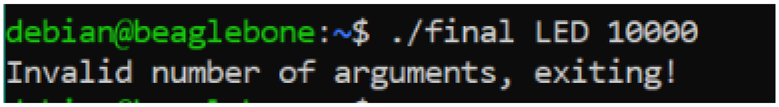
```
debian@beaglebone:~$ gcc final.c -o final
```

- **Executing the object file with the brightness count for the LED.**



```
debian@beaglebone:~$ ./final "LED 0"
The following was read in [3]:

debian@beaglebone:~$ ./final "LED 255"
The following was read in [3]:
```

- **When an invalid argument is used**



```
debian@beaglebone:~$ ./final LED 10000
Invalid number of arguments, exiting!
```

**CONCLUSION**

- Lab-6 was really important to understand how the UART interfacing of Arduino works with the beaglebone black and it can used in the future for our capstone project.
- Also the arduino IDE is really intuitive and easy to work with.
- There are a lot of pre-defined examples that can be used for a lot of sensors while working with arduino.

**APPENDIX**

Youtube video link (please watch it on a phone for better experience):

1. through Arduino IDE terminal
   https://www.youtube.com/watch?v=AlZST-iJ_Gg

2. throught beaglebone terminal
   https://www.youtube.com/watch?v=gM2rXrexXBE

- CODE:

```
//HEADERFILES
#include <stdio.h>                //printf and scanf
#include <fcntl.h>                //for sleep() function
#include <unistd.h>               //for read and write funtions
#include<termios.h>               //used for the terminal interface
#include<string.h>                //used for strings and inbuilt string functions
```

```
//MAINFUNCTION of final.c

int main(int argc, char *argv[])
{
    int file, count;
    if(argc!=2)
            {
                    printf("Invalid number of arguments, exiting!\n");
                    return -2;
            }
    if ((file = open("/dev/ttyO4", O_RDWR | O_NOCTTY | O_NDELAY))<0) //used to access
                                                                   //ttyO4
            {
                    perror("UART: Failed to open the file.\n");
                    return -1;
            }

struct termios options; //declaring an object for the struct termios
tcgetattr(file, &options);
options.c_cflag = B115200  | CS8 | CREAD | CLOCAL; //115200  is the baud rate
options.c_iflag = IGNPAR | ICRNL;
tcflush(file, TCIFLUSH);
tcsetattr(file, TCSANOW, &options);

if ((count = write(file, argv[1], strlen(argv[1])+1))<0) //+1 is because of the null character
    {
            perror("Failed to write to the output\n");
            return -1;
    }

usleep(100000);  //to provide a delay
unsigned char receive[100];
if ((count = read(file, (void*)receive, 100))<0) //reading from the file
    {
            perror("Failed to read from the input\n");
            return -1;
    }
```

```
 if (count==0)
      printf("There was no data available  to read!\n");

 else
      {
                  receive[count]=0; //There is no null character sent by the Arduino
                  printf("The following was read in [%d]: %s\n",count,receive);
      }
 close(file);
 return 0;
}
```