Xiaobin Zhuang - 300519184

ECEN301 Lab 5

11/09/2020

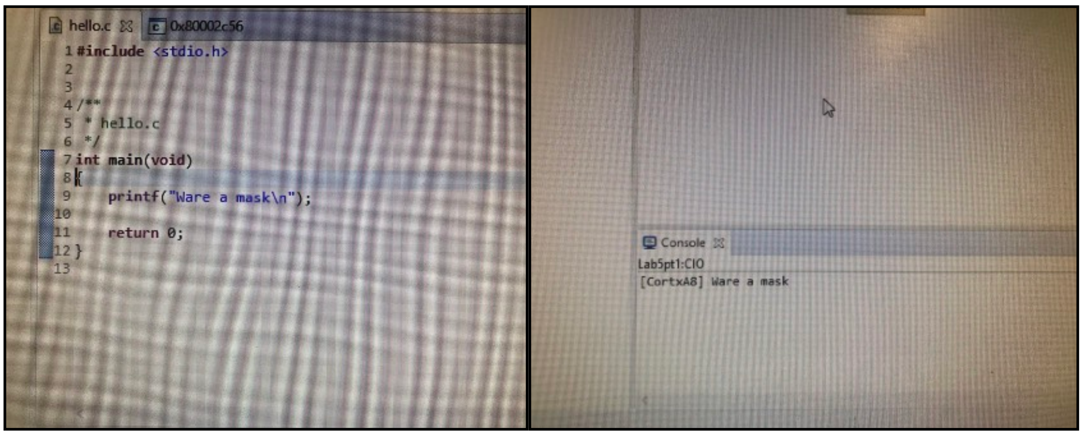
Objectives

The purpose of this lab session is to introduce a more powerful microprocessor (Texas Instruments Sitara AM3358 ARM Cortex-A8 32bit microprocessor) and development environment.

Methodology

Setting up development environment by following the lab manual from Step 1 to 5.

Step 6: changing the message from “Hello World!” To “Ware a mask”, then ran the code via debug. Shown in figure 1.



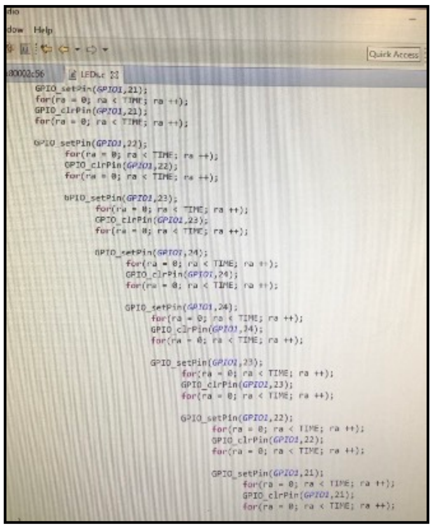
```c
#include <stdio.h>


/**
 * hello.c
 */
int main(void)
{
    printf("Ware a mask\n");

    return 0;
}
```

```
Lab5pt1:CIO
[CortxA8] Ware a mask
```

fig. 1

Step 7: changing flashing rate of LED by changing ‘TIME’ from 500000 to 50000. So the flashing speed would be slower.

The figure on the right showed the code of changing LED patterns. This was a waterfall light control. I lighted up LED one by one from left to right, and turn off one by one after lighting up. Then lighted up from right to left, and also turn off after lighting up. Keep repeating this procedure with flashing rate 50000.



```c
GPIO_setPin(GPIO1,21);
for(ra = 0; ra < TIME; ra ++);
GPIO_clrPin(GPIO1,21);
for(ra = 0; ra < TIME; ra ++);

GPIO_setPin(GPIO1,22);
    for(ra = 0; ra < TIME; ra ++);
    GPIO_clrPin(GPIO1,22);
    for(ra = 0; ra < TIME; ra ++);

    GPIO_setPin(GPIO1,23);
        for(ra = 0; ra < TIME; ra ++);
        GPIO_clrPin(GPIO1,23);
        for(ra = 0; ra < TIME; ra ++);

        GPIO_setPin(GPIO1,24);
            for(ra = 0; ra < TIME; ra ++);
            GPIO_clrPin(GPIO1,24);
            for(ra = 0; ra < TIME; ra ++);

            GPIO_setPin(GPIO1,24);
                for(ra = 0; ra < TIME; ra ++);
                GPIO_clrPin(GPIO1,24);
                for(ra = 0; ra < TIME; ra ++);

                GPIO_setPin(GPIO1,23);
                    for(ra = 0; ra < TIME; ra ++);
                    GPIO_clrPin(GPIO1,23);
                    for(ra = 0; ra < TIME; ra ++);

                    GPIO_setPin(GPIO1,22);
                        for(ra = 0; ra < TIME; ra ++);
                        GPIO_clrPin(GPIO1,22);
                        for(ra = 0; ra < TIME; ra ++);

                        GPIO_setPin(GPIO1,21);
                            for(ra = 0; ra < TIME; ra ++);
                            GPIO_clrPin(GPIO1,21);
                            for(ra = 0; ra < TIME; ra ++);
```

fig. 2

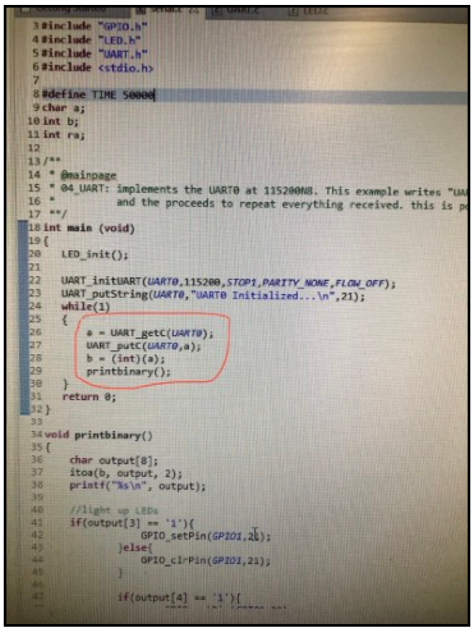
Step 8: Displays the lower 4 bits of ASCII character onto the 4 blue LEDs.

To achieve this program, in the main while loop:

First, got the characters that type by keyboard by UART0, then save them into an array ‘a’.

Then, converted data type *char* into *int* and saved in b array.

Last, execute *printbinary()* program. Shown in figure 3.



```
#include "GPIO.h"
#include "LED.h"
#include "UART.h"
#include <stdio.h>

#define TIME 50000
char a;
int b;
int ra;

/**
 * @mainpage
 * 04_UART: implements the UART0 at 115200N8. This example writes "UA
 *          and the proceeds to repeat everything received. this is po
 **/
int main (void)
{
    LED_init();

    UART_initUART(UART0,115200,STOP1,PARITY_NONE,FLOW_OFF);
    UART_putString(UART0,"UART0 Initialized...\n",21);
    while(1)
    {
        a = UART_getC(UART0);
        UART_putC(UART0,a);
        b = (int)(a);
        printbinary();
    }
    return 0;
}

void printbinary()
{
    char output[8];
    itoa(b, output, 2);
    printf("%s\n", output);

    //light up LEDs
    if(output[3] == '1'){
            GPIO_setPin(GPIO1,21);
        }else{
            GPIO_clrPin(GPIO1,21);
        }

        if(output[4] == '1'){
```

fig. 3

Inside of *printbinary()* program, first, I created a char array with size 8 called **output.**

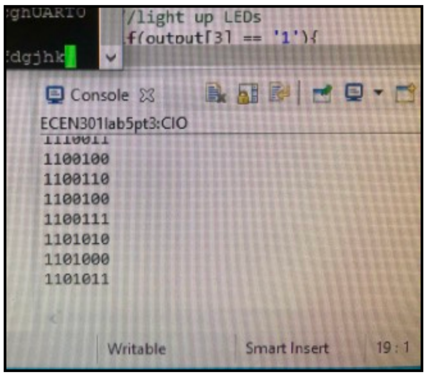
Then use itoa() function to convert the integer of b array into binary numbers, and saved in **output.** Using printf() to observe outcome.

ASCII are 8 bits numbers, however first bit is usually 0. So only 7 bits could be saved into array.

Last, I needed lower 4 bits, so started to judge from 4th element of the array. If the element equal to 1, light up LED; if not, keep off.

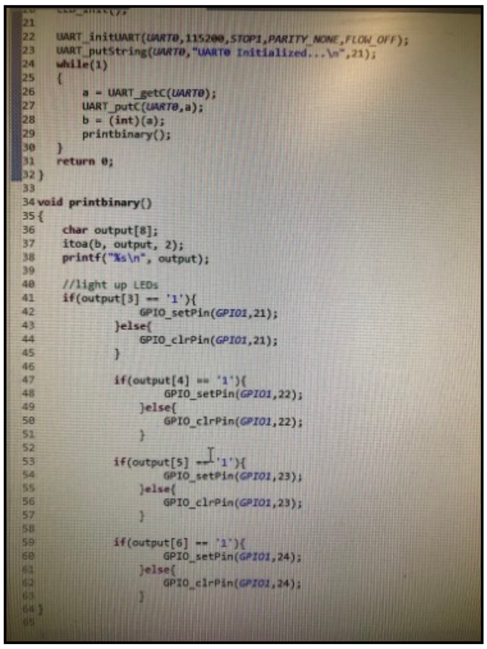
Code shown in figure 5.

Figure 4 showed the printed binary numbers which were converted from keyboard characters. There were only 7 bits in the array.



```
1110011
1100100
1100110
1100100
1100111
1101010
1101000
1101011
```

fig. 4



```
    UART_initUART(UART0,115200,STOP1,PARITY_NONE,FLOW_OFF);
    UART_putString(UART0,"UART0 Initialized...\n",21);
    while(1)
    {
        a = UART_getC(UART0);
        UART_putC(UART0,a);
        b = (int)(a);
        printbinary();
    }
    return 0;
}

void printbinary()
{
    char output[8];
    itoa(b, output, 2);
    printf("%s\n", output);

    //light up LEDs
    if(output[3] == '1'){
            GPIO_setPin(GPIO1,21);
        }else{
            GPIO_clrPin(GPIO1,21);
        }

        if(output[4] == '1'){
            GPIO_setPin(GPIO1,22);
        }else{
            GPIO_clrPin(GPIO1,22);
        }

        if(output[5] == '1'){
            GPIO_setPin(GPIO1,23);
        }else{
            GPIO_clrPin(GPIO1,23);
        }

        if(output[6] == '1'){
            GPIO_setPin(GPIO1,24);
        }else{
            GPIO_clrPin(GPIO1,24);
        }
}
```

fig. 5