Interfacing laboratory

Report 3

MBED

Yara Al-Shafei

1120068

Teacher Assistant: Mohammed mudlal

Instructor: Dr Wasel Ghanem

Abstract

Programmable hardware has been used widely in the recent years due it is great benefits and uses. This report is going to give an introduction about MBED device, application to use it, software libraries and discussion for the procedure was followed to do them and a conclusion about the experiments.

Table of content

1. Introduction

1.1 What is MBED?

MBED is a board contains ARM microcontroller. This board can do many things such as communication and storing data, also, it connected to many components such as LEDs and potentiometer. MBED has it is software which allow the user to write a code to control these components. So it gives you the ability to build your project from the scratch.

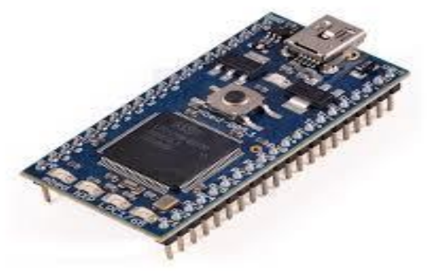
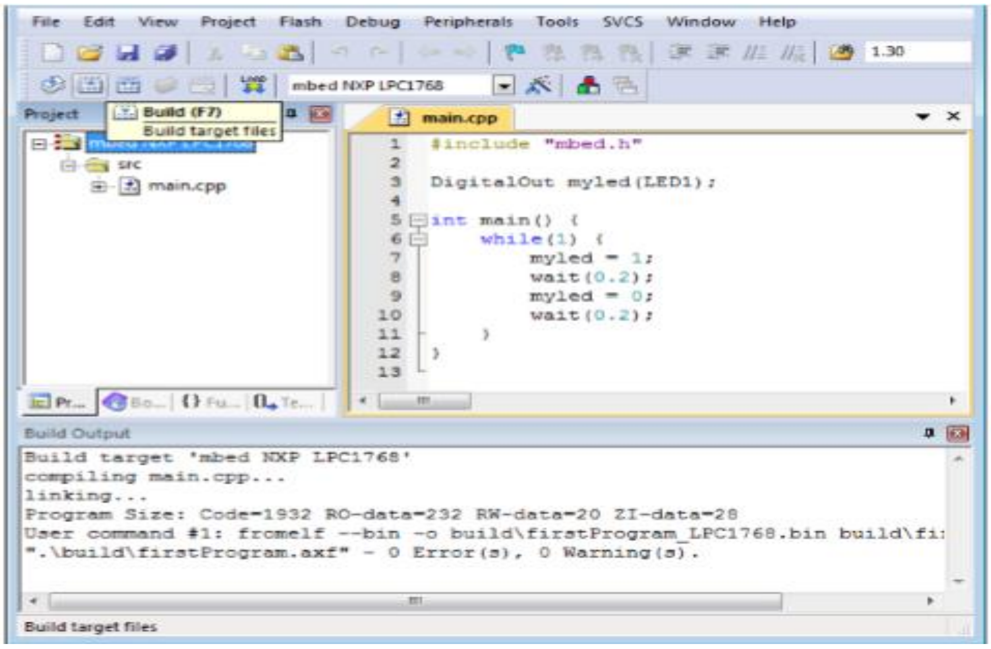


Figure 1.1.1 MBED

1.2 Prototyping MBED

MBED allows you to write a code on top of MBED OS very quickly and this code can work on any other board. The MBED software looks like the next picture:



```
#include "mbed.h"

DigitalOut myled(LED1);

int main() {
    while(1) {
        myled = 1;
        wait(0.2);
        myled = 0;
        wait(0.2);
    }
}
```

```
Build target 'mbed NXP LPC1768'
compiling main.cpp...
linking...
Program Size: Code=1932 RO-data=232 RW-data=20 ZI-data=28
User command #1: fromelf --bin -o build\firstProgram_LPC1768.bin build\fi:
".\build\firstProgram.axf" - 0 Error(s), 0 Warning(s).
```

Figure 1.2.1 MBED software

1.3 MBED pins

MBED has analog pins to read external voltage, and digital pins for the digital unputs and outputs.

The following figure shows the mbed microcontroller pins:

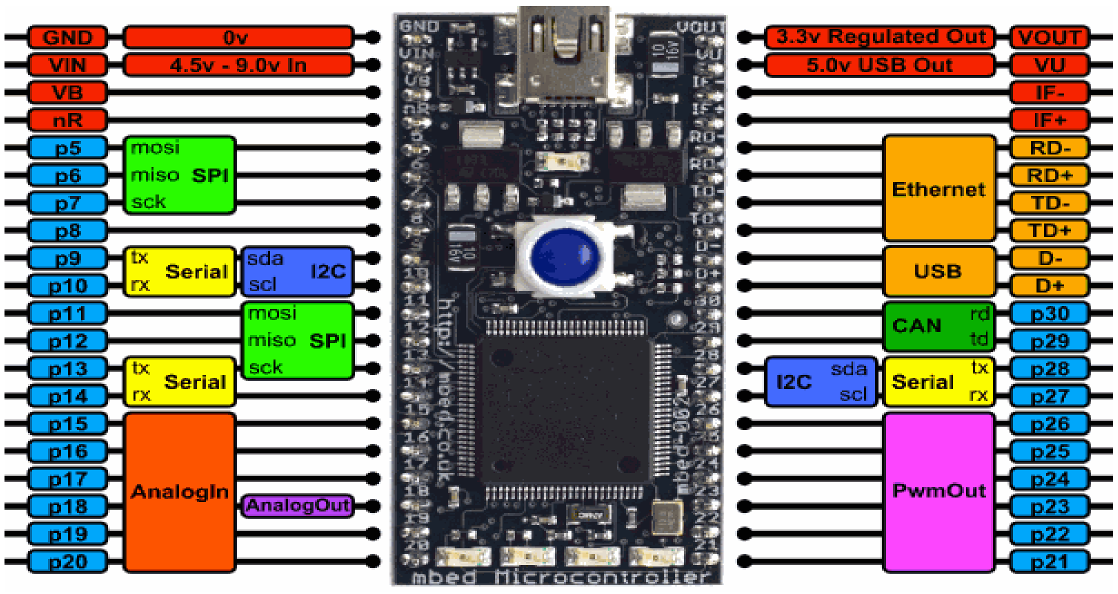


Figure 1.3.1 MBED pins

1.4 Combinational and sequential logic

Both sequential and combinational logic are types of digital circuits. Combinational logic is used for Boolean algebra and have no memory, while the sequential logic used for building finite state machines and have memory to help in building them. Combinational circuits output depends only on the present input but sequential circuits output depends on the present input and past input.

1.5 ADC

Analog to digital converter are systems used very widely because of they are helpful in many applications. These systems take analog signal input and converted to digital output by passing it in several stages. The stages are sampling which means converting the input signal into numbers based on the voltage value. The sampling will follow the nyqest theorem. The second stage is quntitization to assign a digital value for the samples.

1.6 DAC

Digital to analog converter is the oppose of ADC, it takes a digital input then converted to analog output. The stages differ from DAC. Many applications DAC can be used in like music players and military radar systems.

1.7 Timers, Tickers, Timeout and Interrupts

The Timer interface is used to create, start, stop and read a timer for measuring small times (between microseconds and seconds). Any number of Timer objects can be created, and can be started and stopped independently.

The InterruptIn interface is used to trigger an event when a digital input pin changes.

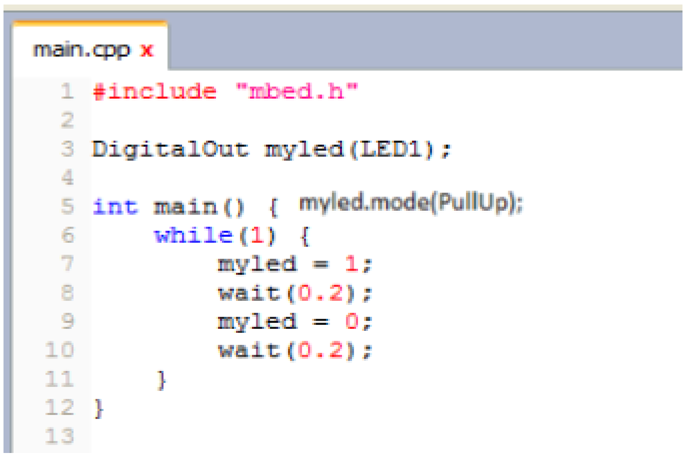
The Ticker interface is used to setup a recurring interrupt to repeatedly call a function at a specified rate. Any number of Ticker objects can be created, allowing multiple outstanding interrupts at the same time. The function can be a static function, or a member function of a particular object.

The Timeout interface is used to setup an interrupt to call a function after a specified delay. Any number of Timeout objects can be created, allowing multiple outstanding interrupts at the same time.

2. Procedure and discussion

2.1 LED lighting

We started with this simple program to get familiar with MBED device and its software. We wrote the given program in the experiment, as follow:



main.cpp x

```
#include "mbed.h"

DigitalOut myled(LED1);

int main() {  myled.mode(PullUp);
    while(1) {
        myled = 1;
        wait(0.2);
        myled = 0;
        wait(0.2);
    }
}

```

Figure 2.1.1 Code

When we ran the programm the LED1 on the MBED lighted, as in the following picture:

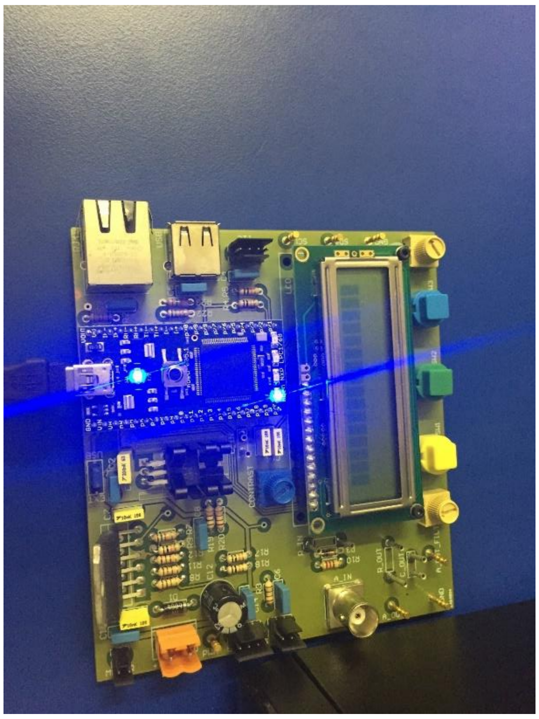
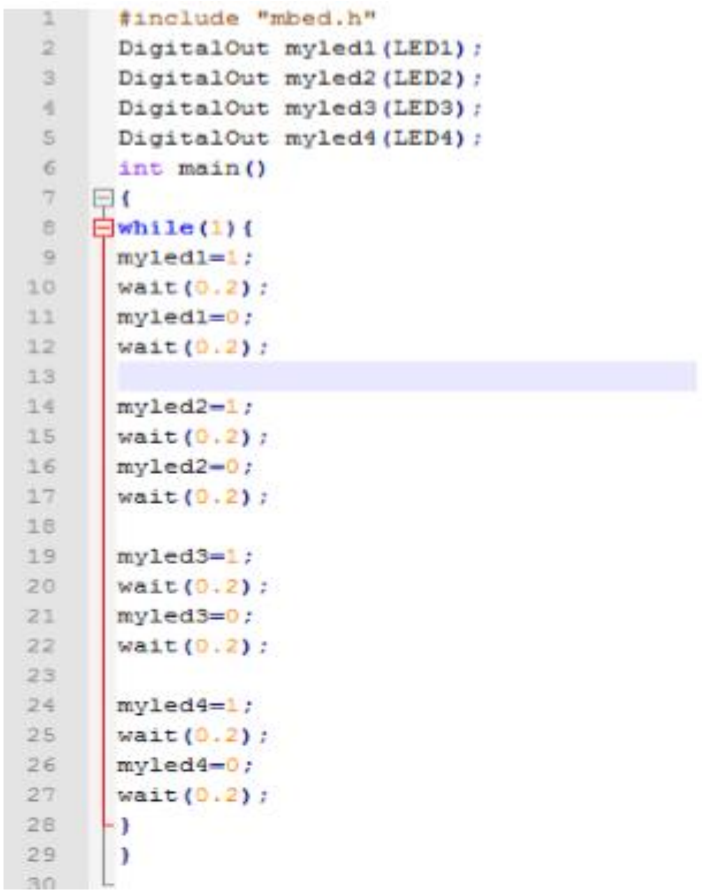


Figure 2.1.2 LED1 light

After that, we wrote a code to light the four LEDs. The code as following:

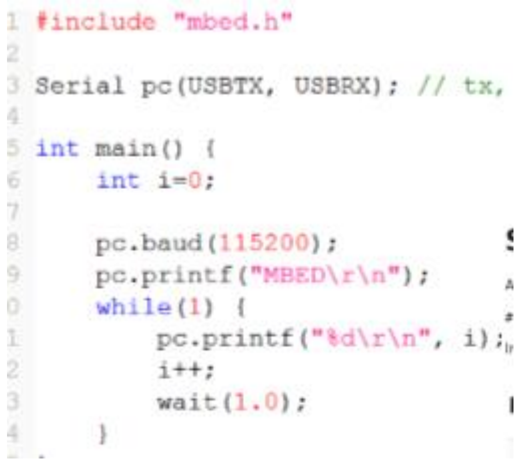


```
#include "mbed.h"
DigitalOut myled1(LED1);
DigitalOut myled2(LED2);
DigitalOut myled3(LED3);
DigitalOut myled4(LED4);
int main()
{
while(1){
myled1=1;
wait(0.2);
myled1=0;
wait(0.2);

myled2=1;
wait(0.2);
myled2=0;
wait(0.2);

myled3=1;
wait(0.2);
myled3=0;
wait(0.2);

myled4=1;
wait(0.2);
myled4=0;
wait(0.2);
}
}
```

Figure 2.1.3 Code

2.2 Serial communication via USB

In this part we used the TeraTerm program to make serial communication with the computer. We wrote the given code into MBED software and opened TeraTerm on the computer to see the communication. The following photos show the code, MBED connection and TeraTerm screen in order:



```
#include "mbed.h"

Serial pc(USBTX, USBRX); // tx,

int main() {
    int i=0;

    pc.baud(115200);
    pc.printf("MBED\r\n");
    while(1) {
        pc.printf("%d\r\n", i);
        i++;
        wait(1.0);
    }
```

Figure 2.2.1 Code

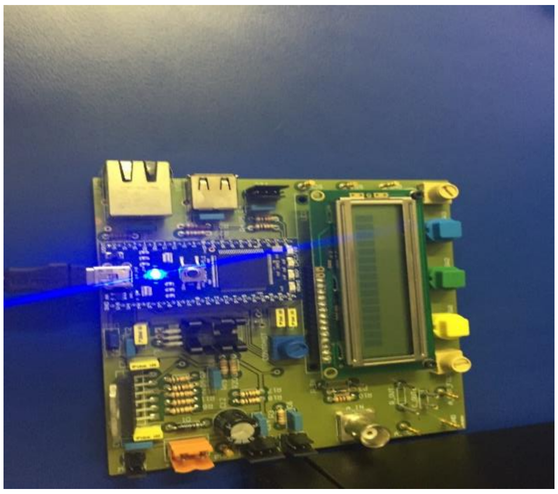
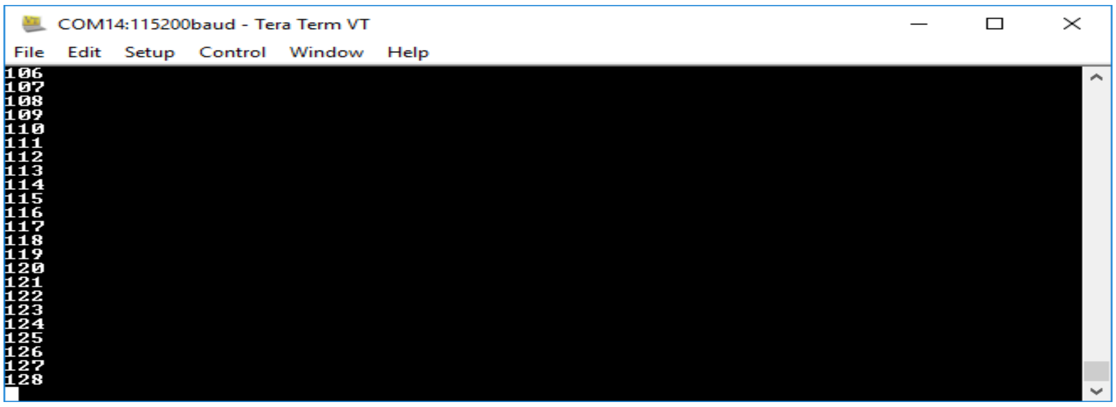


Figure 2.2.2 MBED connection

TeraTerm output:

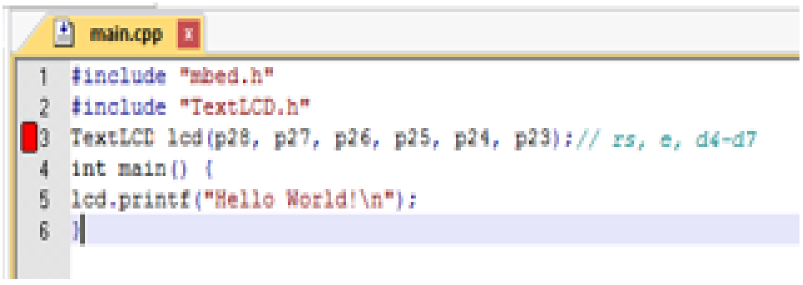

COM14:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
```

Figure 2.2.3 TeraTerm screen

2.3 Text LCD – Hello World !

In this part we showed "Hello World!" on the LCD connected with the MDEB, using the given code and pins number to connect. The following photos show the code and the output in order:



```
#include "mbed.h"
#include "TextLCD.h"
TextLCD lcd(p28, p27, p26, p25, p24, p23);// rs, e, d4-d7
int main() {
lcd.printf("Hello World!\n");
}
```

Figure 2.3.1 Code

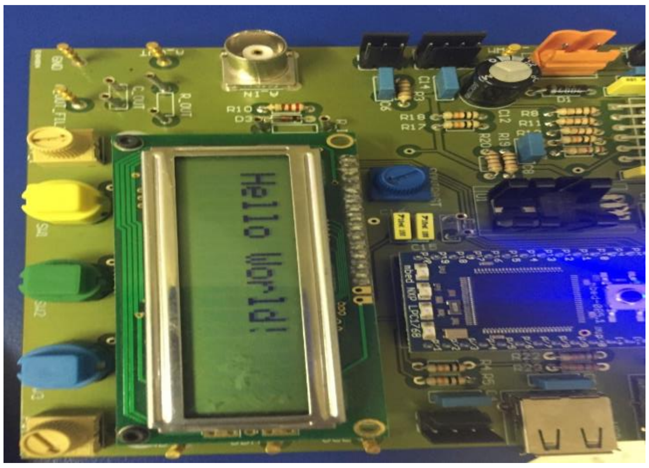
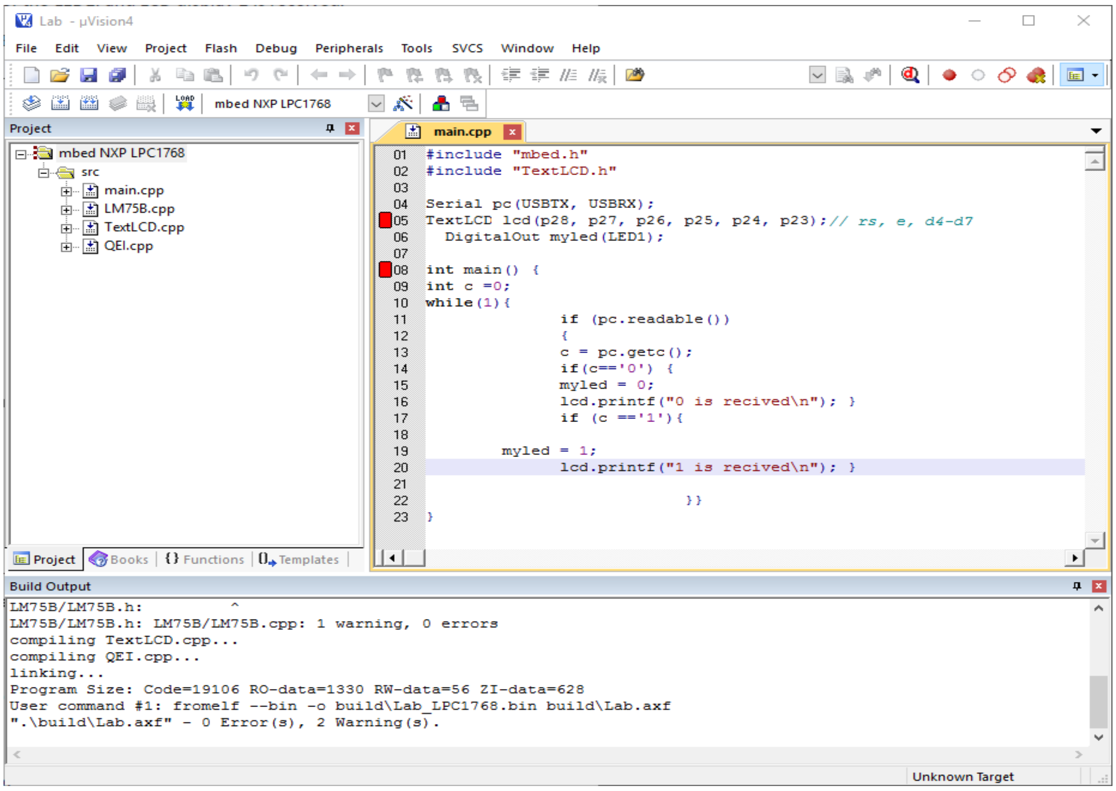


Figure 2.3.2 Output

2.4 TODO

In this TODO we want to display 1 on the LCD which will turn the LED1 on and 0 which will turn the LED1 off. We wrote the following code:



```
#include "mbed.h"
#include "TextLCD.h"

Serial pc(USBTX, USBRX);
TextLCD lcd(p28, p27, p26, p25, p24, p23);// rs, e, d4-d7
  DigitalOut myled(LED1);

int main() {
int c =0;
while(1){
              if (pc.readable())
              {
              c = pc.getc();
              if(c=='0') {
              myled = 0;
              lcd.printf("0 is recived\n"); }
              if (c =='1'){

        myled = 1;
              lcd.printf("1 is recived\n"); }

                          }}
}
```

```
LM75B/LM75B.h:                ^
LM75B/LM75B.h: LM75B/LM75B.cpp: 1 warning, 0 errors
compiling TextLCD.cpp...
compiling QEI.cpp...
linking...
Program Size: Code=19106 RO-data=1330 RW-data=56 ZI-data=628
User command #1: fromelf --bin -o build\Lab_LPC1768.bin build\Lab.axf
".\build\Lab.axf" - 0 Error(s), 2 Warning(s).
```

Figure 2.4.1 Code

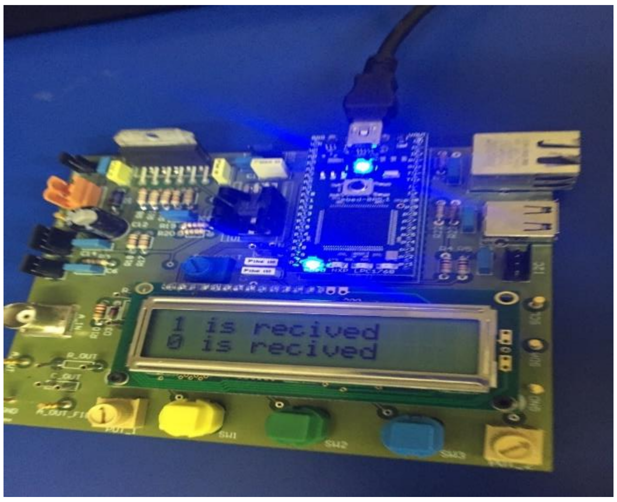


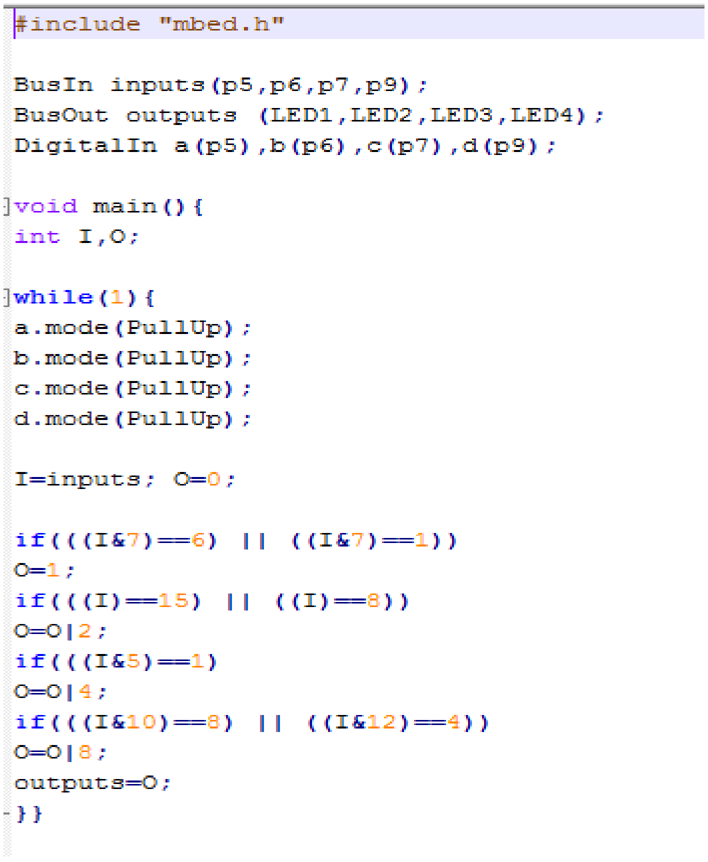
Figure 2.4.2 Output

2.5 Combinatorial equation

In this part we wrote a code for the combinational equation:

X = /a.b.c + a./b./c.d + a.c.d

The code:



```
#include "mbed.h"

BusIn inputs(p5,p6,p7,p9);
BusOut outputs (LED1,LED2,LED3,LED4);
DigitalIn a(p5),b(p6),c(p7),d(p9);

void main(){
int I,O;

while(1){
a.mode(PullUp);
b.mode(PullUp);
c.mode(PullUp);
d.mode(PullUp);

I=inputs; O=0;

if(((I&7)==6) || ((I&7)==1))
O=1;
if(((I)==15) || ((I)==8))
O=O|2;
if(((I&5)==1)
O=O|4;
if(((I&10)==8) || ((I&12)==4))
O=O|8;
outputs=O;
}}
```

2.5.1 Code

We pressed different buttons to make the equation true and these are different ouputs:

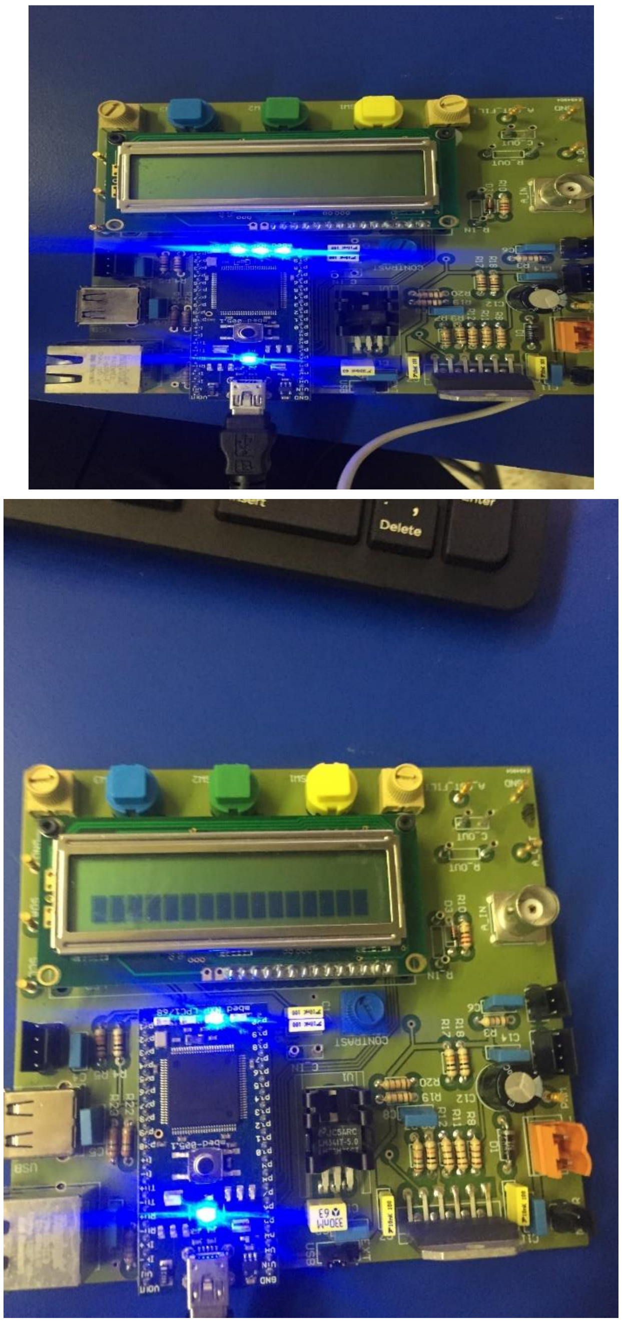


Figure 2.5.2 Output

2.6 Combintional logic

Using BusIn and BusOut classes, and the successive evaluation method, we wrote a code to solve the following equations:

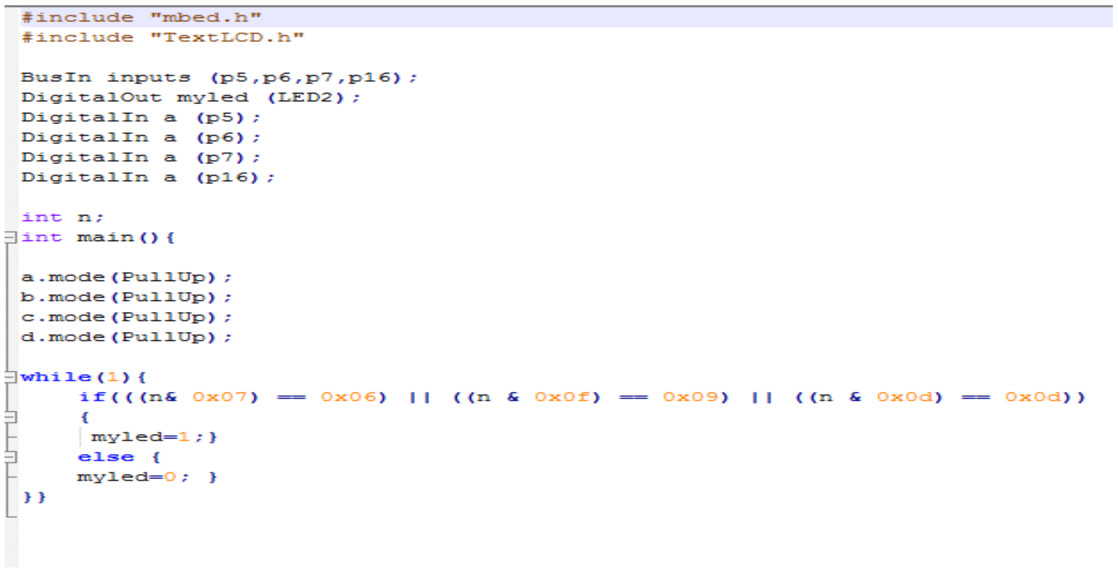
X = /a.b.c + a./b./c

Y = a.b.c.d + /a./b./c.d

Z = a./c

T = /b.d + c./d

The code:



```
#include "mbed.h"
#include "TextLCD.h"

BusIn inputs (p5,p6,p7,p16);
DigitalOut myled (LED2);
DigitalIn a (p5);
DigitalIn a (p6);
DigitalIn a (p7);
DigitalIn a (p16);

int n;
int main(){

a.mode(PullUp);
b.mode(PullUp);
c.mode(PullUp);
d.mode(PullUp);

while(1){
    if(((n& 0x07) == 0x06) || ((n & 0x0f) == 0x09) || ((n & 0x0d) == 0x0d))
    {
     myled=1;}
    else {
    myled=0; }
}}
```

A photo for the output:

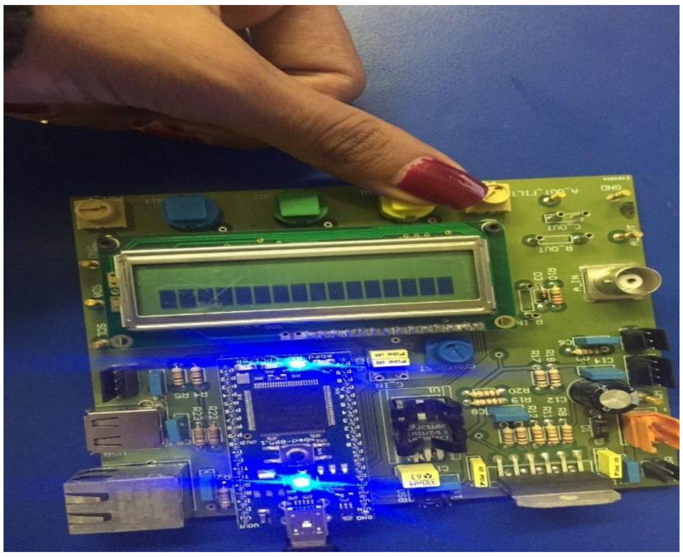


Figure 2.6.1 output

2.7 Sequential system : state machine

Here we wrote a program to describe the following state machine, the output on the LCD is the state we are in. The following photos show the state machine digram, the code and the output:

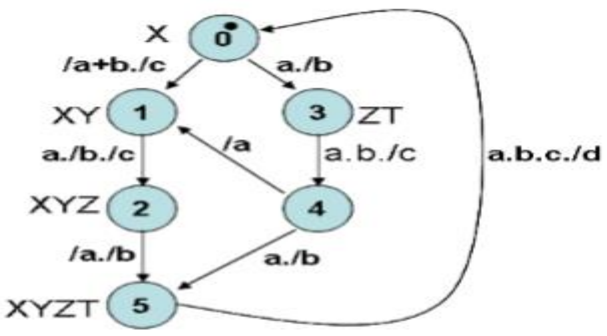
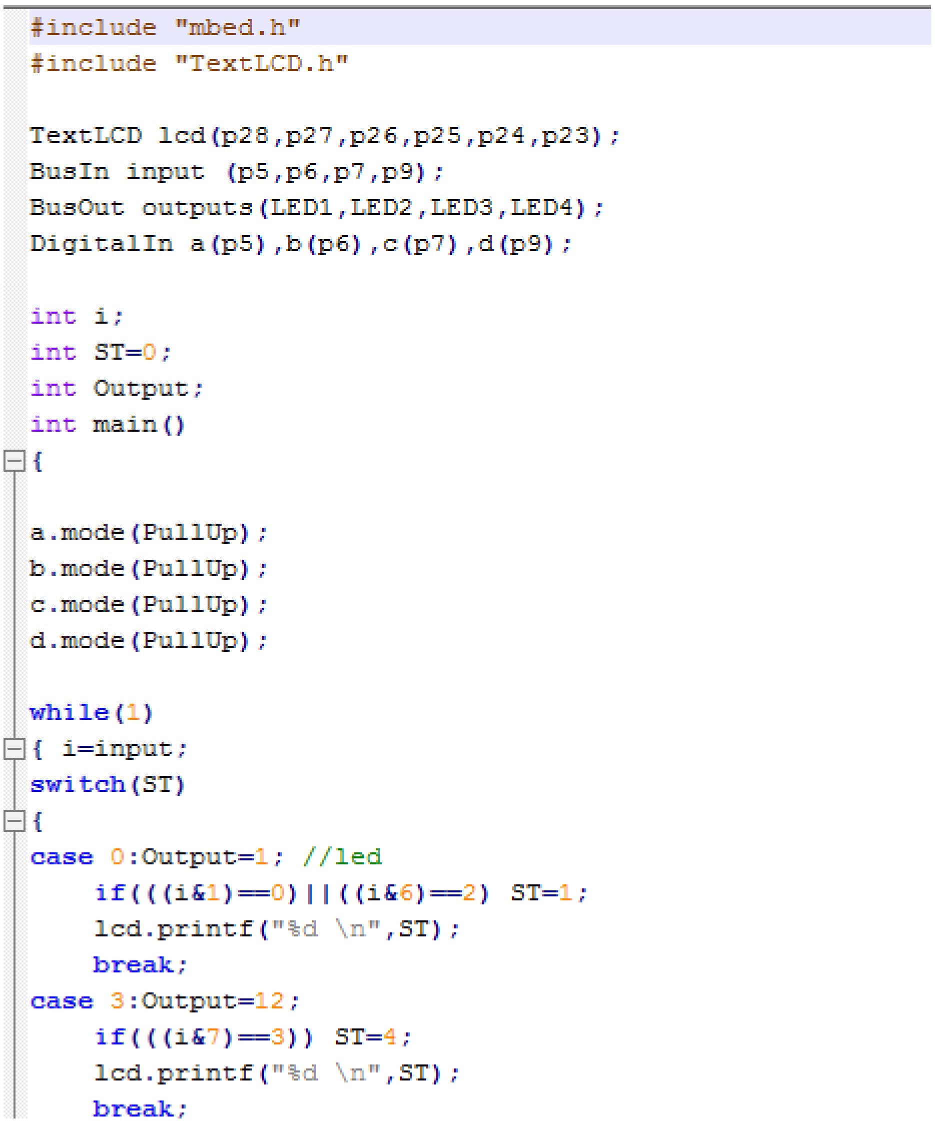
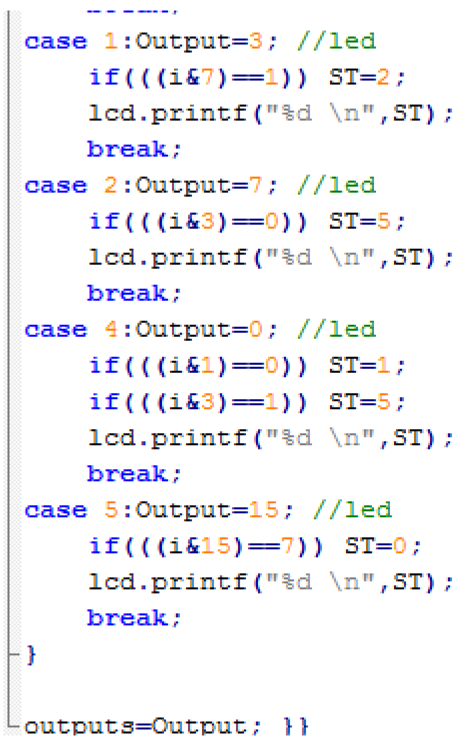


Figure 2.7.1 state machine



```
#include "mbed.h"
#include "TextLCD.h"

TextLCD lcd(p28,p27,p26,p25,p24,p23);
BusIn input (p5,p6,p7,p9);
BusOut outputs(LED1,LED2,LED3,LED4);
DigitalIn a(p5),b(p6),c(p7),d(p9);

int i;
int ST=0;
int Output;
int main()
{

a.mode(PullUp);
b.mode(PullUp);
c.mode(PullUp);
d.mode(PullUp);

while(1)
{ i=input;
switch(ST)
{
case 0:Output=1; //led
    if(((i&1)==0)||((i&6)==2)) ST=1;
    lcd.printf("%d \n",ST);
    break;
case 3:Output=12;
    if(((i&7)==3)) ST=4;
    lcd.printf("%d \n",ST);
    break;
```



```
case 1:Output=3; //led
    if(((i&7)==1)) ST=2;
    lcd.printf("%d \n",ST);
    break;
case 2:Output=7; //led
    if(((i&3)==0)) ST=5;
    lcd.printf("%d \n",ST);
    break;
case 4:Output=0; //led
    if(((i&1)==0)) ST=1;
    if(((i&3)==1)) ST=5;
    lcd.printf("%d \n",ST);
    break;
case 5:Output=15; //led
    if(((i&15)==7)) ST=0;
    lcd.printf("%d \n",ST);
    break;
}

outputs=Output; }}
```

Figure 2.7.2 Code

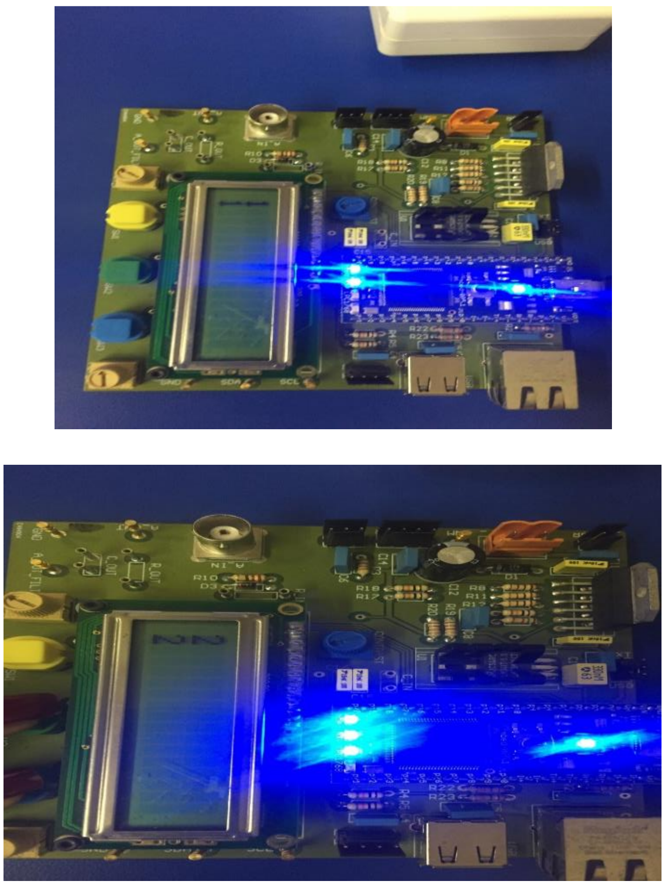
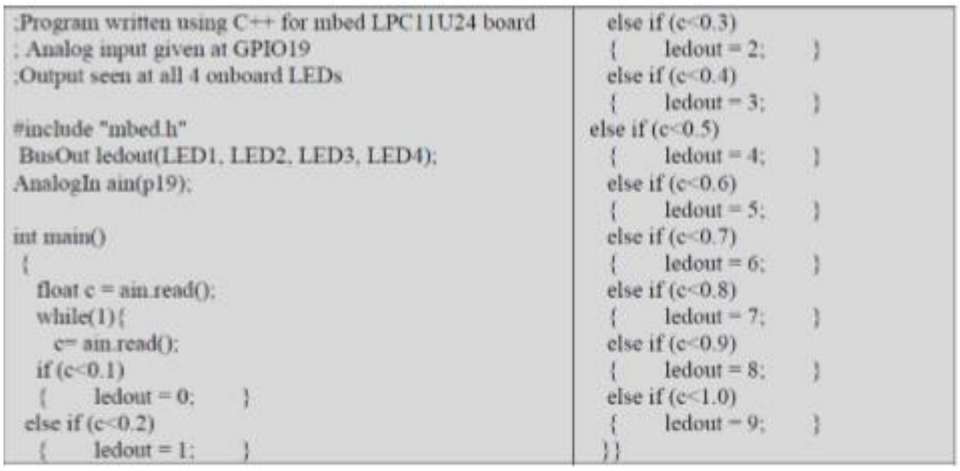


Figure 2.7.3 Output

2.8 Reading an Analog Input

In this part we will connect MBED with a potentiometer, we just needed to know which pin the MBED is connected with the potentiometer to fill the given code:



```
;Program written using C++ for mbed LPC11U24 board
; Analog input given at GPIO19
;Output seen at all 4 onboard LEDs

#include "mbed.h"
 BusOut ledout(LED1, LED2, LED3, LED4);
AnalogIn ain(p19);

int main()
 {
   float c = ain.read();
   while(1){
     c= ain.read();
   if (c<0.1)
    {       ledout = 0;       }
  else if (c<0.2)
    {       ledout = 1;       }
```

```
  else if (c<0.3)
  {       ledout = 2;       }
  else if (c<0.4)
  {       ledout = 3;       }
else if (c<0.5)
  {       ledout = 4;       }
  else if (c<0.6)
  {       ledout = 5;       }
  else if (c<0.7)
  {       ledout = 6;       }
  else if (c<0.8)
  {       ledout = 7;       }
  else if (c<0.9)
  {       ledout = 8;       }
  else if (c<1.0)
  {       ledout = 9;       }
  }}
```

Figure 2.8.1 Code

The output was as the following:

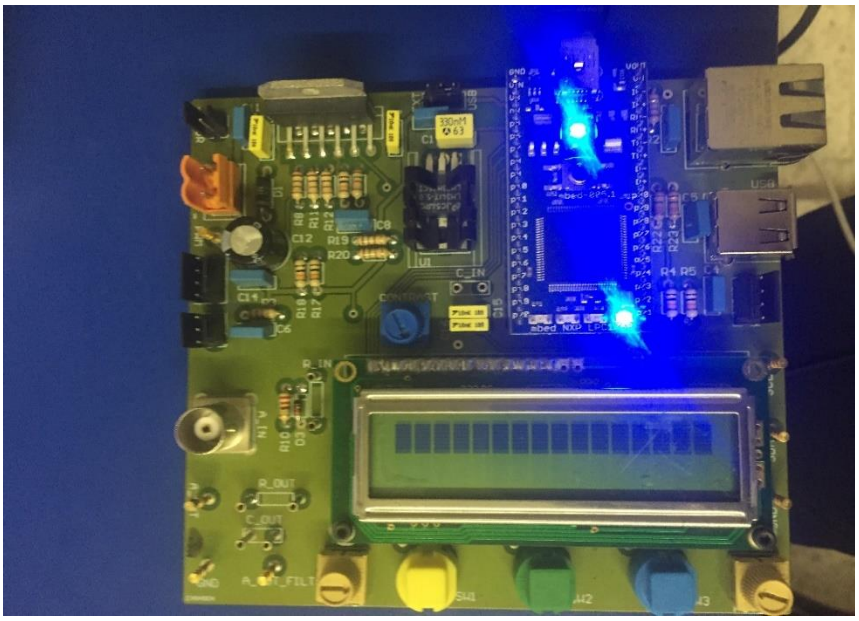
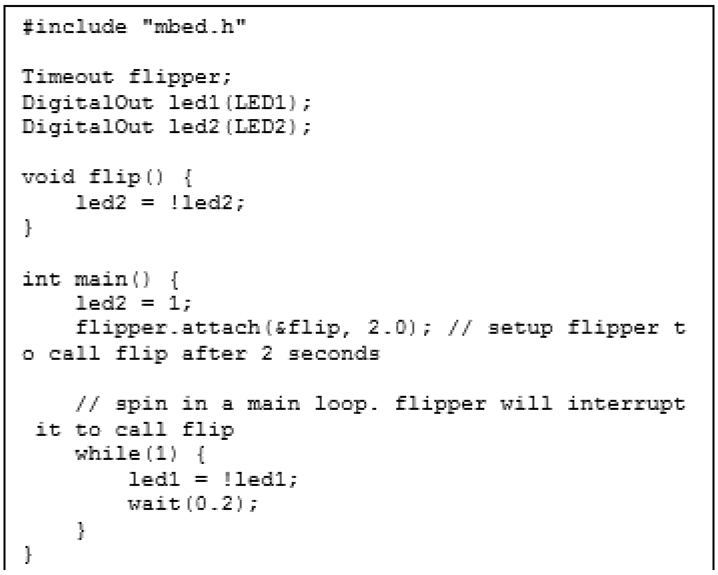


Figure 2.8.2 Output

2.9 Using Timeout interface

In this part we loaded the given code in the experiment which make an interrupt and this interrupt will switch the LED sate after a delay. The following is the code and the output:



```
#include "mbed.h"

Timeout flipper;
DigitalOut led1(LED1);
DigitalOut led2(LED2);

void flip() {
    led2 = !led2;
}

int main() {
    led2 = 1;
    flipper.attach(&flip, 2.0); // setup flipper t
o call flip after 2 seconds

    // spin in a main loop. flipper will interrupt
 it to call flip
    while(1) {
        led1 = !led1;
        wait(0.2);
    }
}
```

Figure 2.9.1 Code

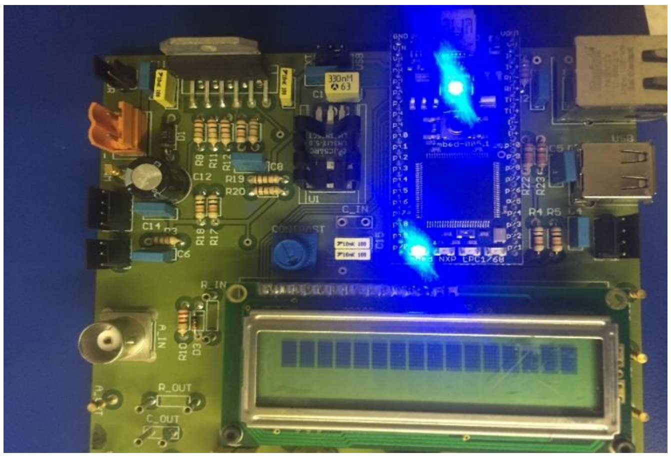
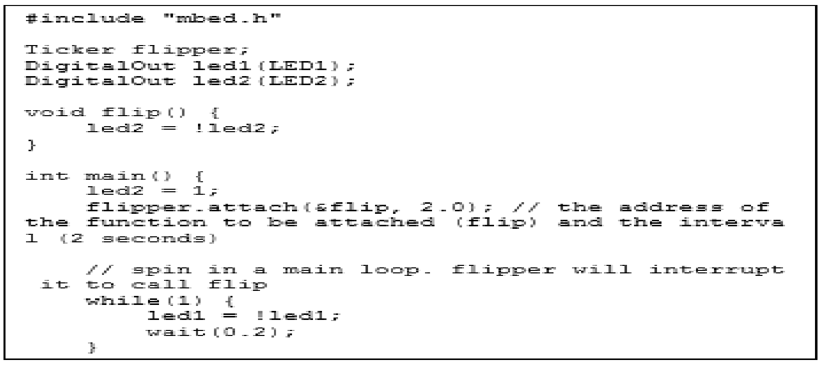


Figure 2.9.2 Output

2.10 Using Ticker interface

In this part we used ticker instead of timer to switch the LED state. The ticker calls the function repeatedly. The following is the code:



```
#include "mbed.h"

Ticker flipper;
DigitalOut led1(LED1);
DigitalOut led2(LED2);

void flip() {
    led2 = !led2;
}

int main() {
    led2 = 1;
    flipper.attach(&flip, 2.0); // the address of
the function to be attached (flip) and the interva
l (2 seconds)

    // spin in a main loop. flipper will interrupt
 it to call flip
    while(1) {
        led1 = !led1;
        wait(0.2);
    }
```

Figure 2.10.1 Code

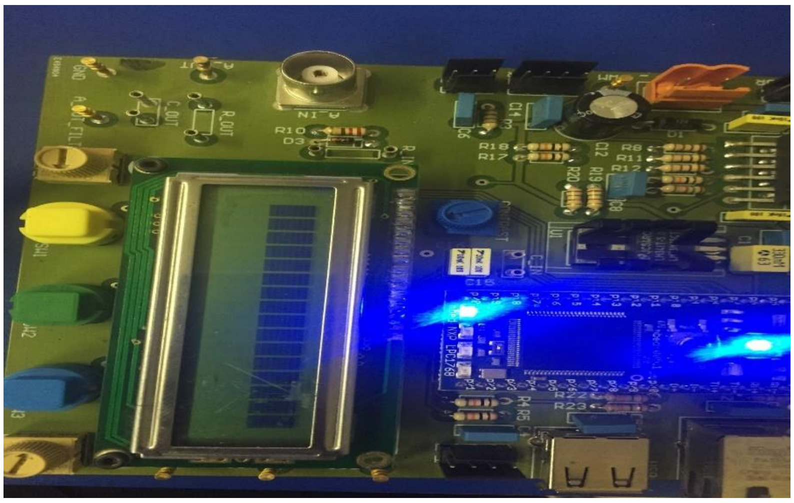
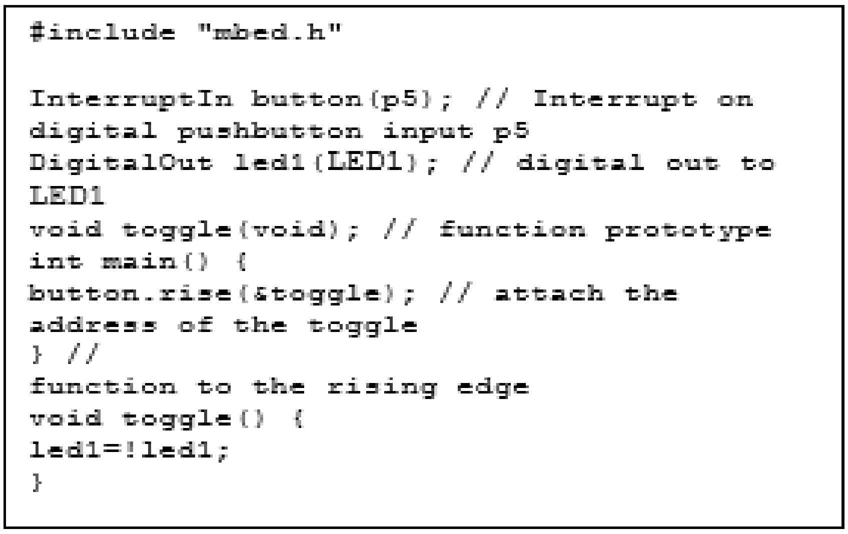


Figure 2.10.2 Output

2.11 External interrupts on the mbed

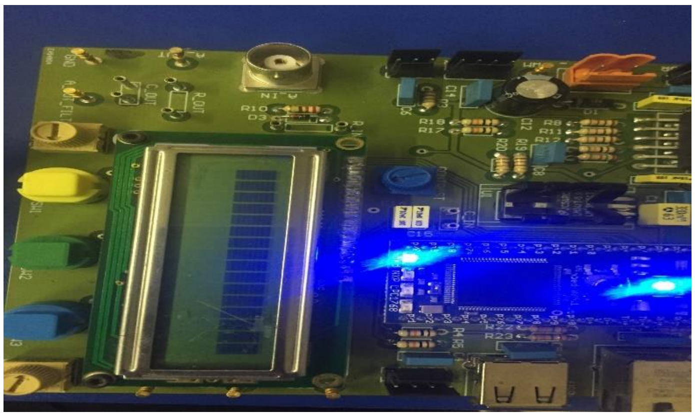
Here we will use mbed InterruptIn library to toggle an LED when the pushbutton is high. The following is the code and the output:



```
#include "mbed.h"

InterruptIn button(p5); // Interrupt on
digital pushbutton input p5
DigitalOut led1(LED1); // digital out to
LED1
void toggle(void); // function prototype
int main() {
button.rise(&toggle); // attach the
address of the toggle
} //
function to the rising edge
void toggle() {
led1=!led1;
}
```

Figure 2.11.1 Code

The output:



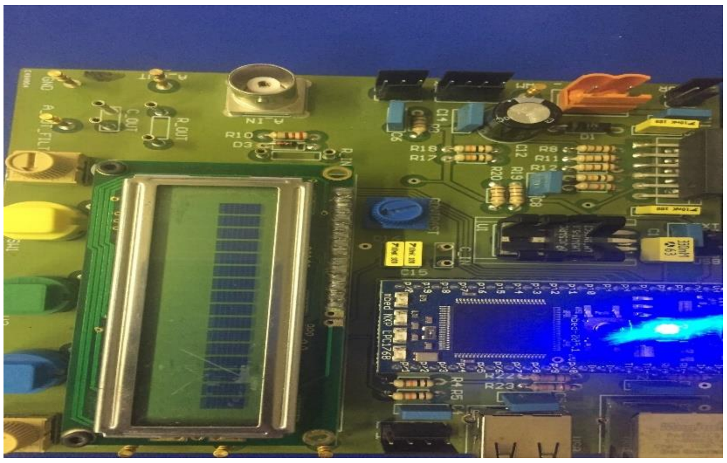
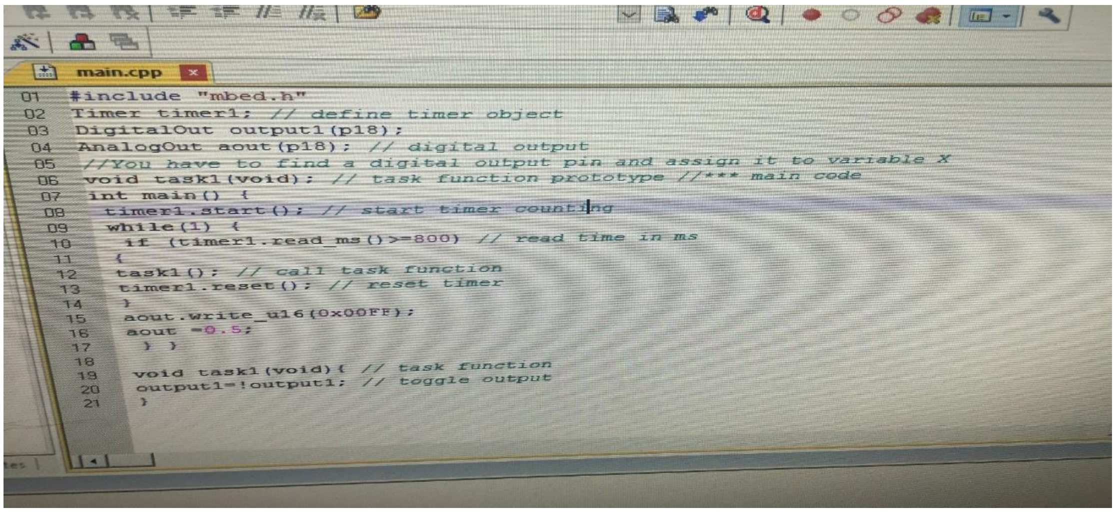


Figure 2.11.2 output

2.12 Using Timer Object

In this part we are going to generate square wave signal using MBED with duty cycle = 0.5.

The following photos will show the code, connection and the output in order:



```
#include "mbed.h"
Timer timer1; // define timer object
DigitalOut output1(p18);
AnalogOut aout(p18); // digital output
//You have to find a digital output pin and assign it to variable X
void task1(void); // task function prototype //*** main code
int main() {
 timer1.start(); // start timer counting
 while(1) {
  if (timer1.read_ms()>=800) // read time in ms
  {
  task1(); // call task function
  timer1.reset(); // reset timer
  }
  aout.write_u16(0x00FF);
  aout =0.5;
   } }

 void task1(void){ // task function
 output1=!output1; // toggle output
 }
```

Figure 2.12.1 Code

The connection:

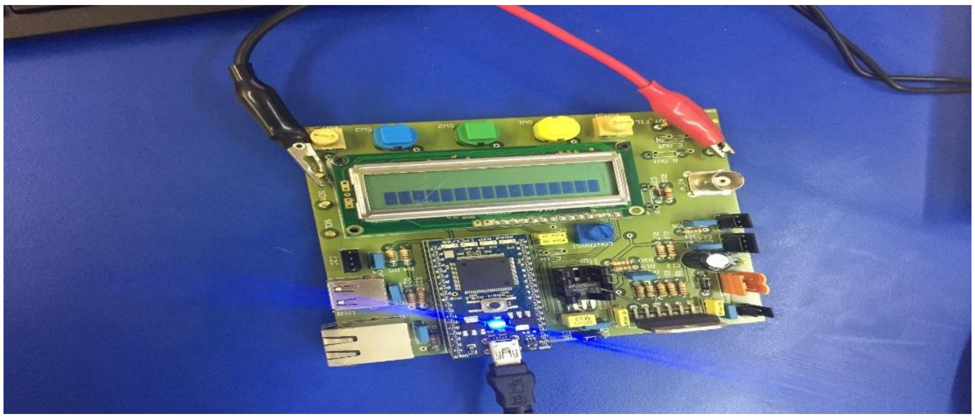


Figure 2.12.2 connection

The output signal:

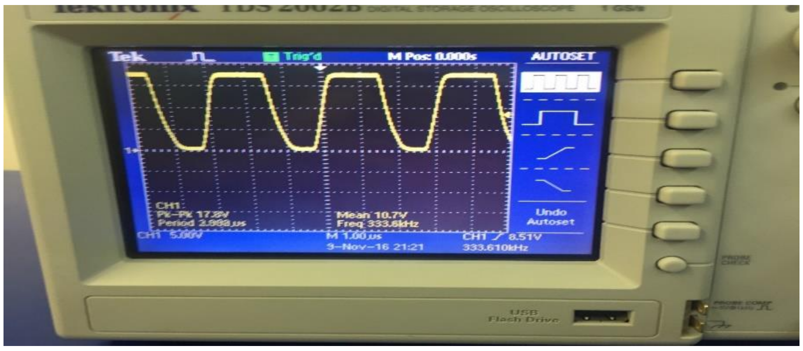


Figure 2.12.3 The output

2.13 TODO

This part is the same as the previous one but with different duty cycle. The new duty cycle is 0.3 so we changed aout=0.3. The output:

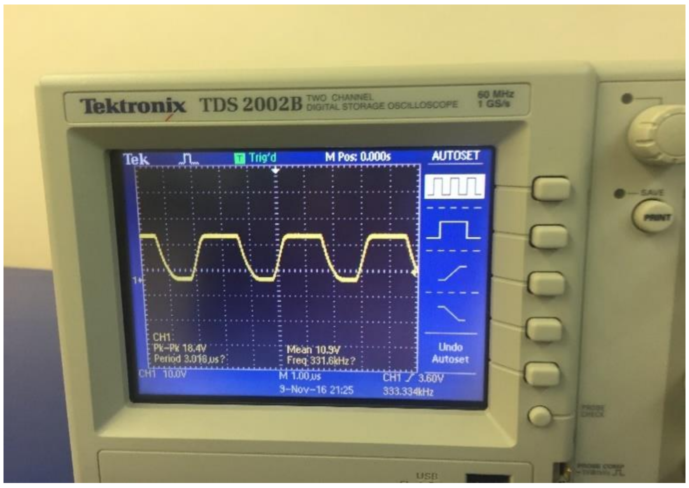


Figure 2.13.1 output

3. Conclusion

MBED can be used to build many useful applications because it is a programmable device, connected to many components and can be connected with many devices.

The experiments were easy and simple. No problems faced us in them.

4.References

[1] "Mbed." ARM. Accessed November 14, 2016. https://www.mbed.com/en/about-mbed/what-mbed/.
[2] "AnalogIn - Handbook | Mbed." AnalogIn - Handbook | Mbed. Accessed November 14, 2016. https://developer.mbed.org/handbook/AnalogIn.
[3] "DigitalIn - Handbook | Mbed." DigitalIn - Handbook | Mbed. Accessed November 14, 2016. https://developer.mbed.org/handbook/DigitalIn.
[4] "Mbed LPC1768." Development Platform for Devices. Accessed November 14, 2016. https://developer.mbed.org/platforms/mbed-LPC1768/.
[5] "Combinational Logic." Wikipedia. Accessed November 14, 2016. https://en.wikipedia.org/wiki/Combinational_logic.
[6] "Sequential Logic." Wikipedia. Accessed November 14, 2016. https://en.wikipedia.org/wiki/Sequential_logic.
[7] **"Analog-to-digital Converter." Wikipedia. Accessed November 16, 2016. https://en.wikipedia.org/wiki/Analog-to-digital_converter**
[8] @HardwareSecrets. "How Analog-to-Digital Converter (ADC) Works - Page 2 of 11 - Hardware Secrets." Hardware Secrets. 2015. Accessed November 16, 2016. http://www.hardwaresecrets.com/how-analog-to-digital-converter-adc-works/2/.
[9] "Digital-to-analog Converter." Wikipedia. Accessed November 16, 2016. https://en.wikipedia.org/wiki/Digital-to-analog_converter.
[10] "Timer - Handbook | Mbed." Timer - Handbook | Mbed. Accessed November 16, 2016. https://developer.mbed.org/handbook/Timer.
[11] "InterruptIn - Handbook | Mbed." InterruptIn - Handbook | Mbed. Accessed November 16, 2016. https://developer.mbed.org/handbook/InterruptIn.
[12] "Ticker - Handbook | Mbed." Ticker - Handbook | Mbed. Accessed November 16, 2016. https://developer.mbed.org/handbook/Ticker.
[13] "Timeout - Handbook | Mbed." Timeout - Handbook | Mbed. Accessed November 16, 2016. https://developer.mbed.org/handbook/Timeout.