

9

Introduction to MBED environment

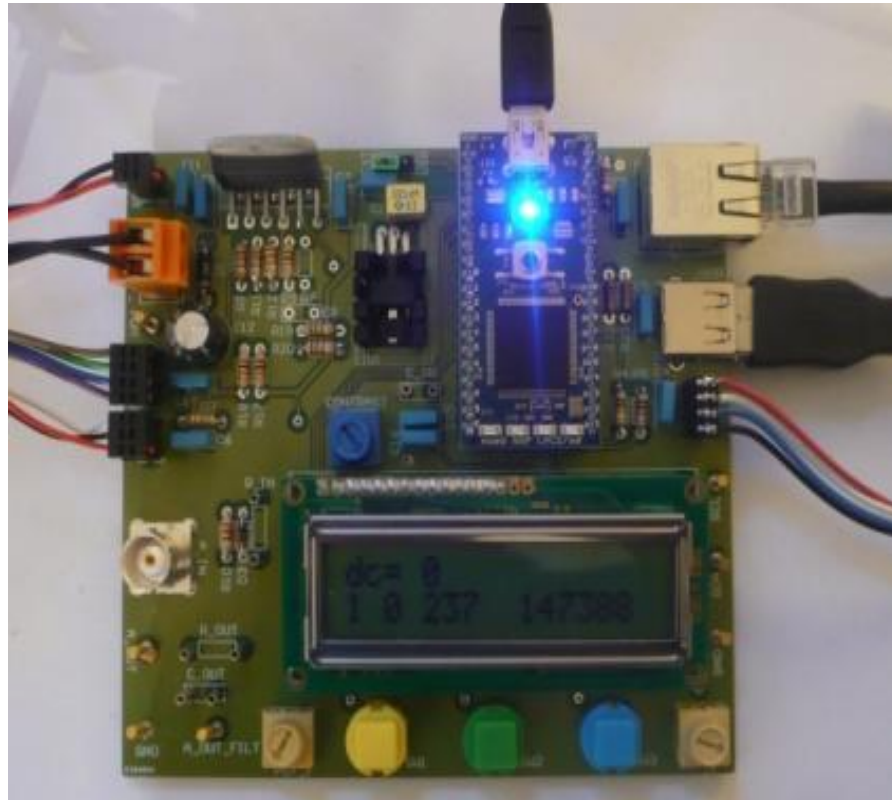


Figure 1 didactic board

1.Objective

To be familiar with MBED software libraries, hardware designs and online or offline tools for professional rapid prototyping of products based on ARM microcontrollers.

2.Overview

The mbed platform provides free software libraries, hardware designs and online tools for professional rapid prototyping of products based on ARM microcontrollers.

The platform includes a standards-based C/C++ SDK, a microcontroller HDK and supported development boards, an online compiler and online developer collaboration tools.

2.1 Rapid prototyping

- Easy to transfer the object code : USB link, no need of a specific programmer
- Easy to wire on a protoboard : 40 pins DIP package
- Easy to write a program :
 - Efficient C++ libraries
 - No need to install any software : online development tool
 - There is offline development tool
 - Community forum and wiki



Figure 3 mbed controller

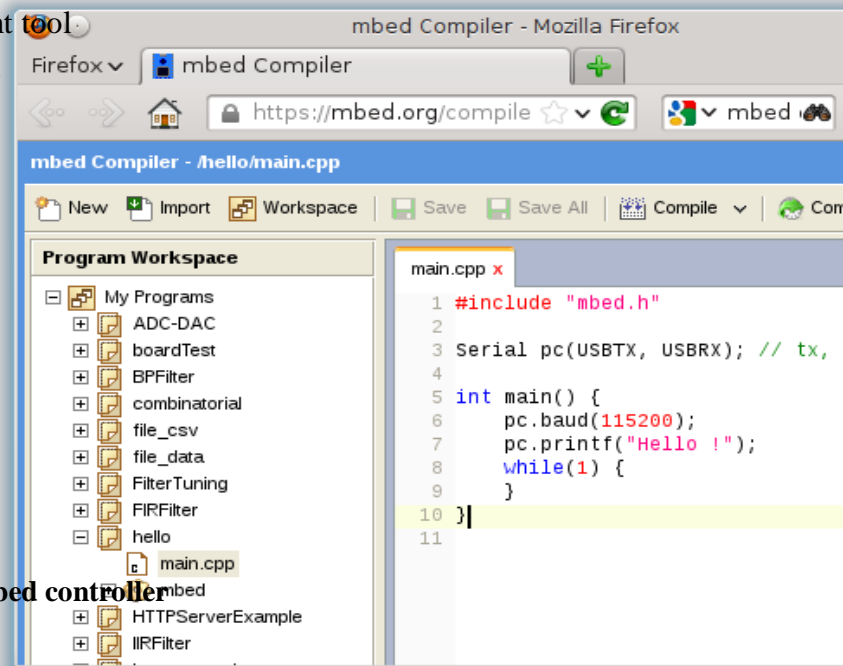


Figure 2 online compiler

2.2 MBED module

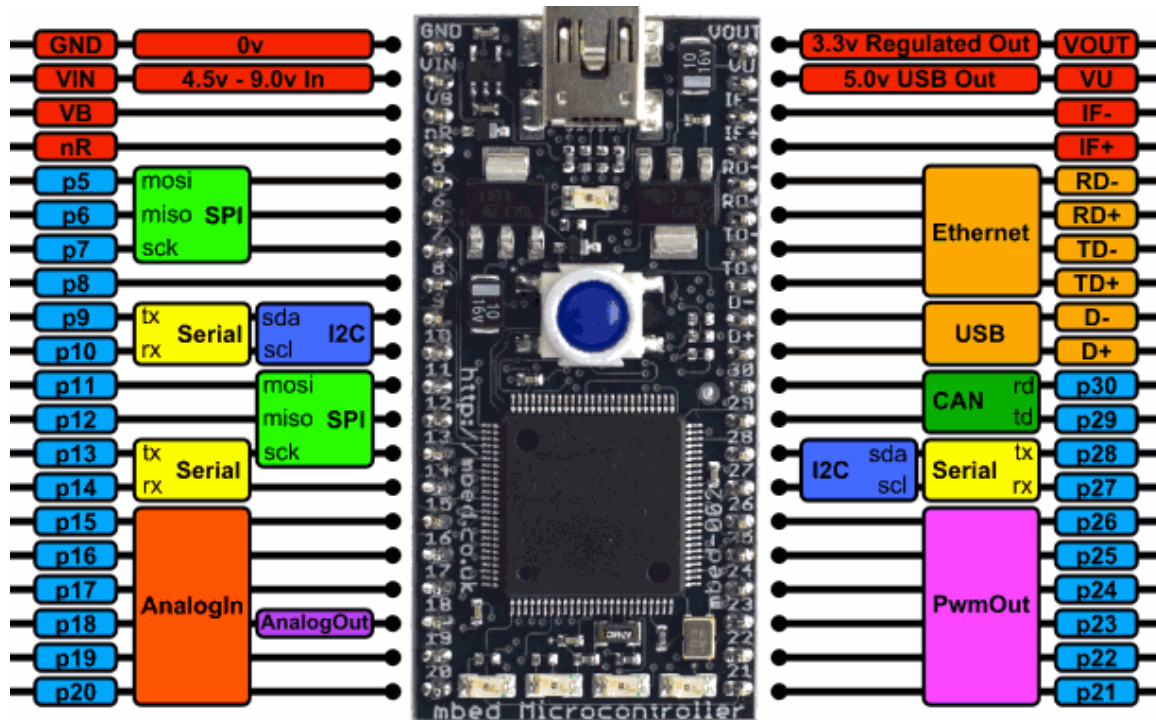


Figure 4 Schematic :<http://mbed.org/media/uploads/chris/mbed-005.1.pdf>

2.3 C++ basics

```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() {
6     while(1) {
7         myled = 1;
8         wait(0.25);
9         myled = 0;
10        wait(0.25);
11    }
12 }
```

Necessary to use MBED libraries

myled is a DigitalOut object
Call the constructor of DigitalOut class

Figure 5 example code

2.3 Offline compiler (Kel uVision 4)

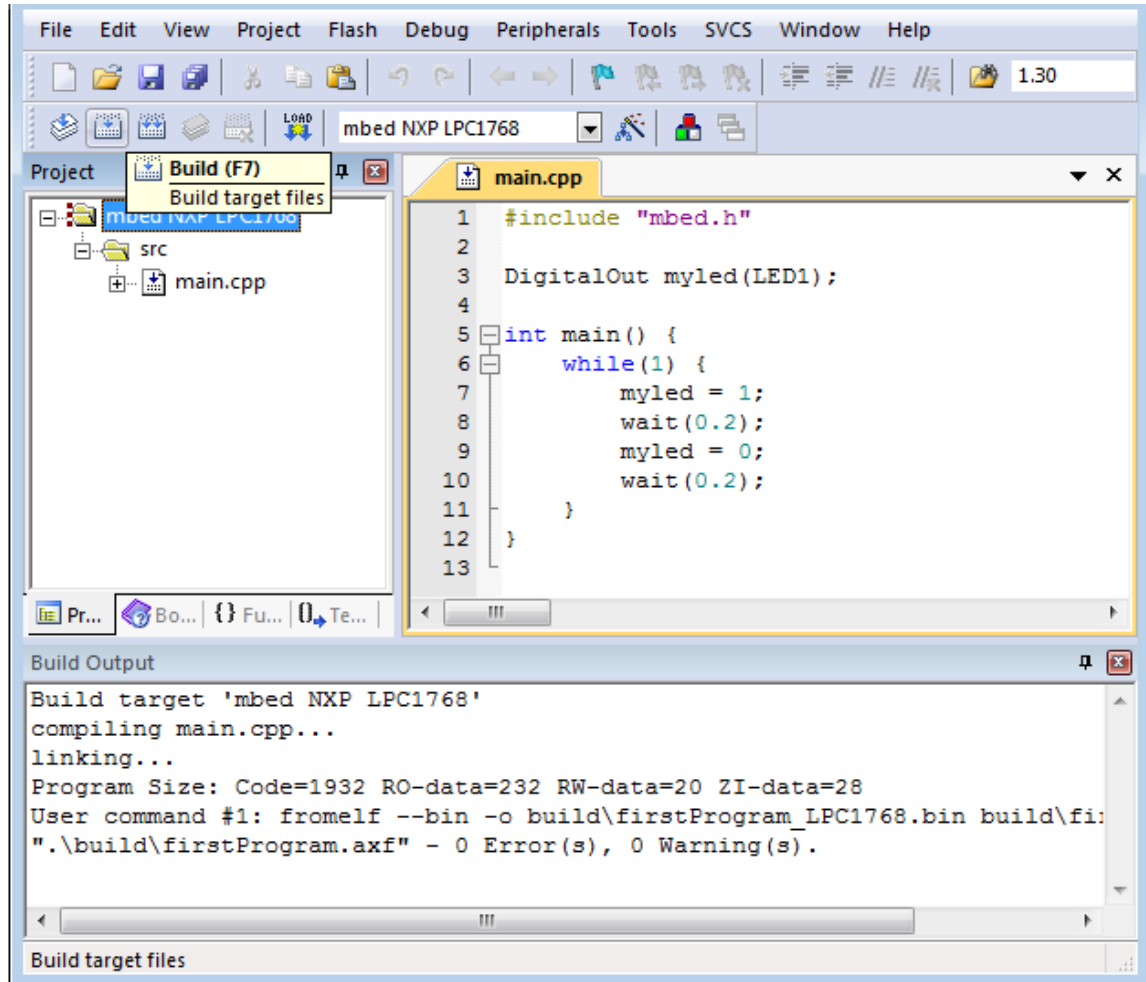


Figure 6 Offline compiler (Kel uVision 4)

2.4 The didactic board (shows in figure 1) has:

External power supply

5V regulator

Selector 5V from USB or from regulator

Motor control

1 chopper (LMD18200)

1 connector for a quadratic encoder

1 connector for a reference position encoder

HMI

3 pushbuttons

2 potentiometers

1 LCD screen 2x16

Analog

1 analog input (BNC connector) filtered or not

1 analog output filtered or not

Communications

1 Ethernet RJ-45

1 USB host

1 I2C

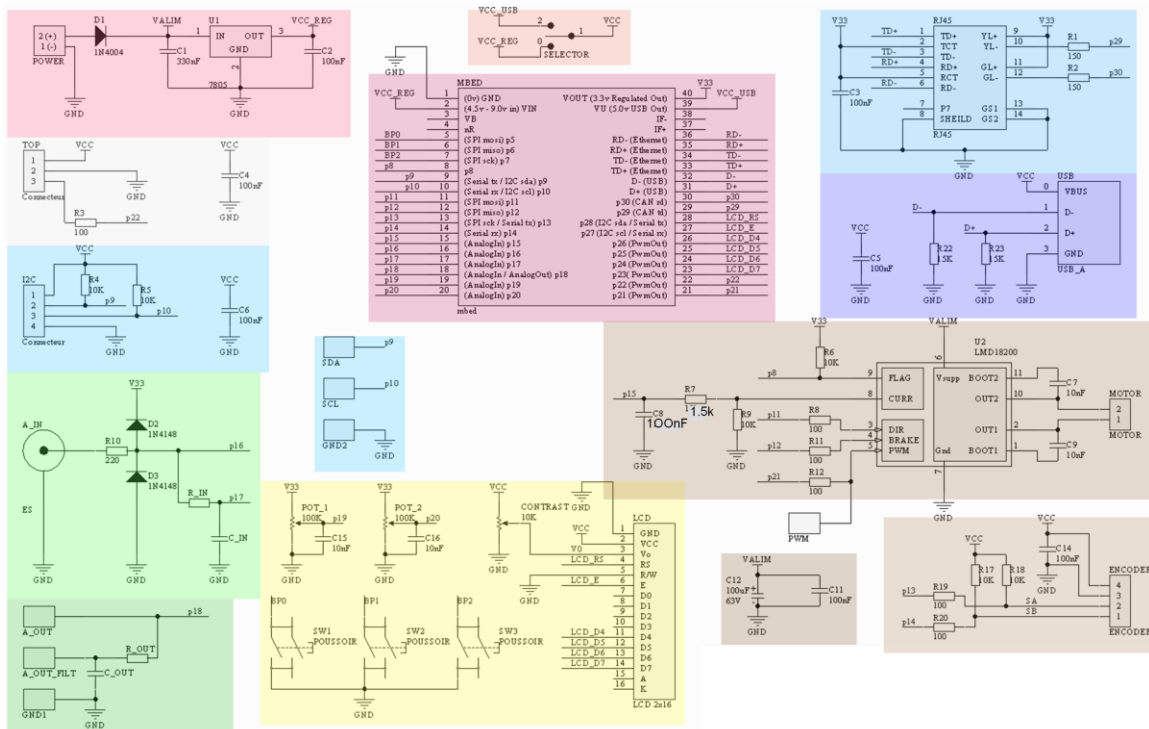


Figure 7 Schematic of didactic board

2.5 Benefits and Limitations of using MBED

☺ MBED (microcontroller, website and documentation) is user-friendly. The MBED module is really interesting to make practices and projects with students.

☺ Using well tested libraries is a good practice.

☺ Use USB link as serial communication with a PC.

☹ Only features provided by libraries avoid low level programming.

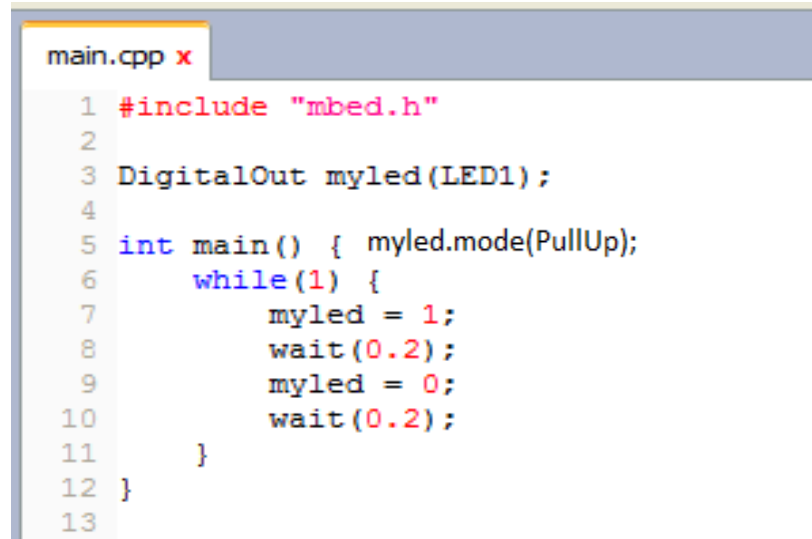
⊖ MBED libraries are not open source not easy to extend low-level implementation of existing libraries and it's difficult to understand the low-level behavior.

⊖ Not yet debugging functionality.

3.Procedure

1. Create your first program.

Write down the following code in the "main.cpp" file



```
main.cpp x
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() { myled.mode(PullUp);
6     while(1) {
7         myled = 1;
8         wait(0.2);
9         myled = 0;
10        wait(0.2);
11    }
12 }
13
```

Figure 8

This program defines a digital output connected to the LED1. Four blue leds are present on the MEBD module. They are associated with pins names LED1, LED2, LED3, LED4.

Click compile button. The program is compiled and a binary file is downloaded. Save this file on the MBED drive. Rest the MBED module. A led is blinking.

Task: Write a code to make the four leds switch on one after the other. First LED1 is on for 0.2s, then LED2 is on for 0.2s, then LED3 is on for 0.2s ...

2. Serial communication via USB.

- Enter this code to the main.cpp file.
- Compile this file and copy the binary file on the MBED driver.
- Execute TeraTerm program and configure the serial link.
- Connect to the MBED virtual serial port and verify the communication.

```

1 #include "mbed.h"
2
3 Serial pc(USBTX, USBRX); // tx, rx
4
5 int main() {
6     int i=0;
7
8     pc.baud(115200);
9     pc.printf("MBED\r\n");
10    while(1) {
11        pc.printf("%d\r\n", i);
12        i++;
13        wait(1.0);
14    }
15 }

```

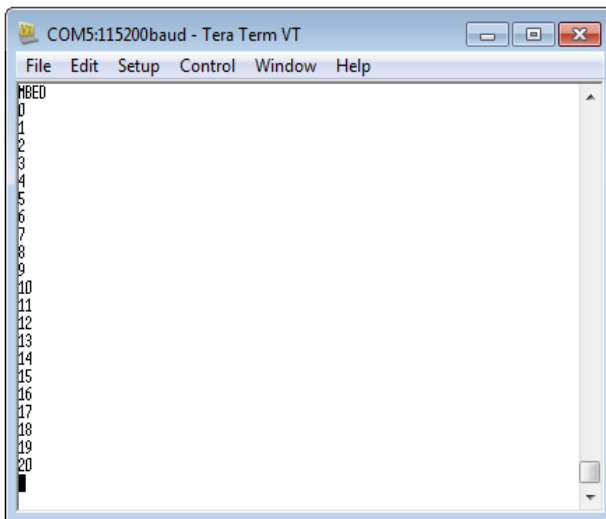
Serial Class Reference

A serial port (UART) for communication with other serial devices. [More...](#)

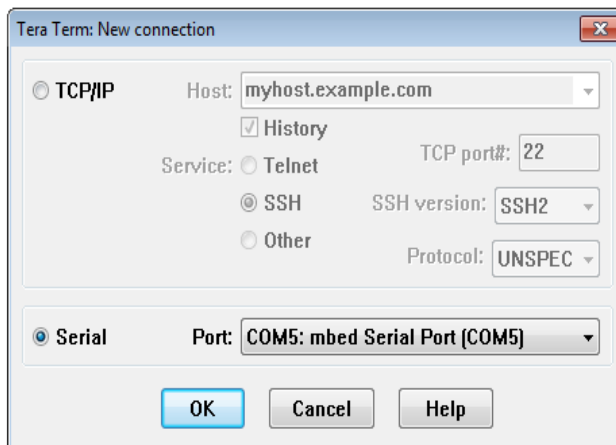
```
#include <Serial.h>
Inherits mbed::Stream.
```

Public Member Functions

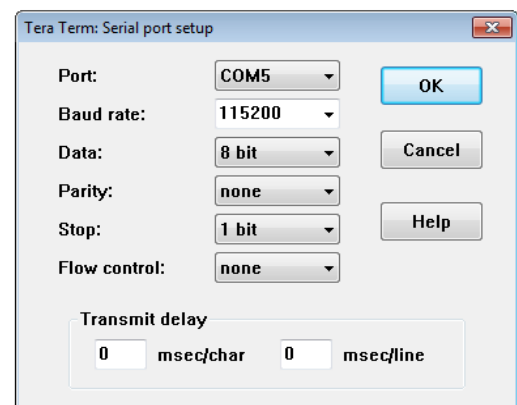
| | |
|--|--|
| Serial (PinName tx, PinName rx) | Create a Serial port, connected to the specified transmit and receive pins. |
| void baud (int baudrate) | Set the baud rate of the serial port. |
| void format (int bits=8, SerialParity parity=ParityNone, int stop_bits=1) | Set the transmission format used by the Serial port. |
| int readable () | Determine if there is a character available to read. |
| int writeable () | Determine if there is space available to write a character. |



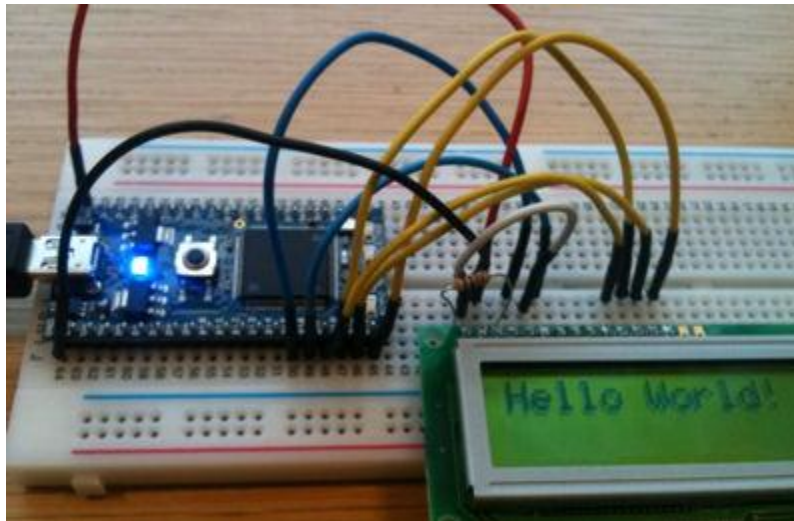
TeraTerm serial communication configuration.



Menu Setup → Serial port...



3. TextLCD>HelloWorld! – main.cpp



```
// Hello World! for the TextLCD

#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");
}
```

We have sent data from the MBED to the PC. Now we will send data from the PC to the MBED. The member function **readable()** of **Serial** class return 1 if there is a character available to read, 0 otherwise. The member function **getc()** reads the character.

TODO:

We will write the code to do the following:

- Character '1' send by the PC switch on the LED1, and LCD display 1 is received.
- Character '0' send by the PC switch of the LED1, and LCD display 1 is received.

Define myled as a DigitalOut connected to LED1.

Define c as an int and insert and complete this code inside the infinite loop:

```
    if ( pc.readable() )      {
        C = pc.getc();
        if (c == '0') {
            ...
        } else if ( c == '1')
            ....
    }
```


Test your code.

Now we will use the interrupt mechanism of Serial class instead of polling the reception. The method **attach(myFunction)** of **Serial** class attach a function to call whenever character is received.

Create a function **receive ()** which reads and analyzes a received character. Attach the function **receive ()** to the **Serial** object **pc** before the infinite loop:

```
pc.attach ( receive( ) );
```

Remove any code inside the infinite loop.

Test your code.