Quick start guide to mBed and STM NUCLEO Boards

Introduction

The purpose of this manual is to give you a fast introduction to the use of mBed tool and STM NUCLEO boards.

For more info see the:

- **mBed API documentation that is here**
- **Fast and Effective Embedded Systems Design: Applying the ARM mbed** is **here**.

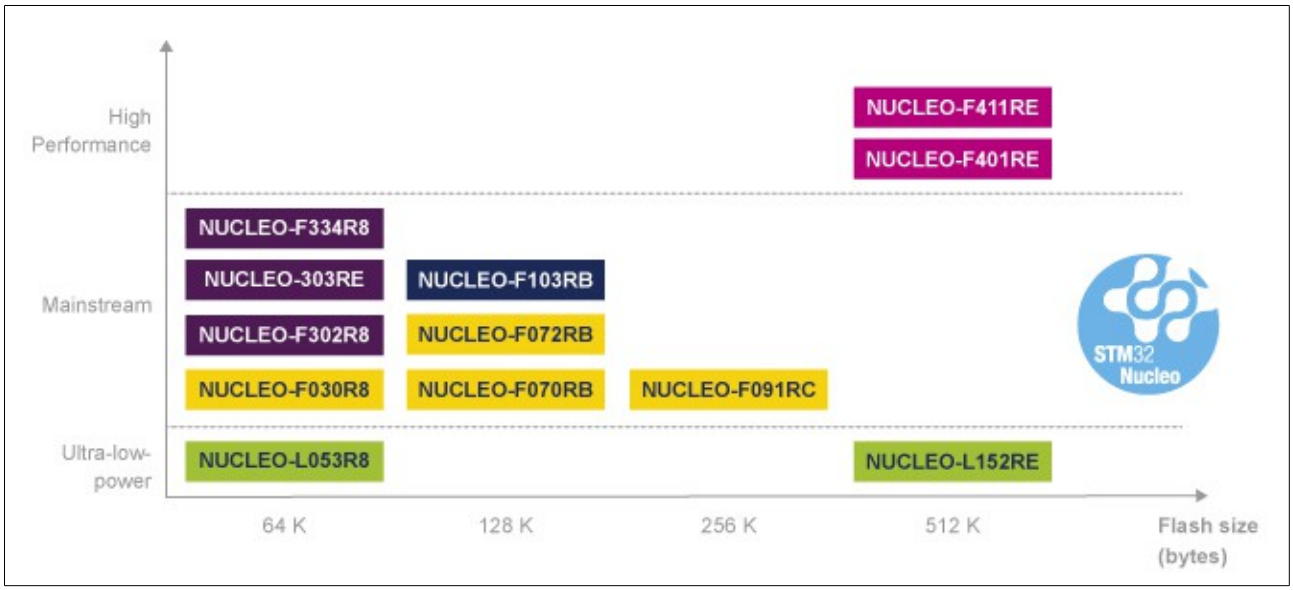
Also see the links below:

- Mbed home page
- General sw
- Library, provides the C/C++ software platform and libraries to build your applications
- Mbed compiler
- C++ Basics
- My mBed and NUCLEO tutorials

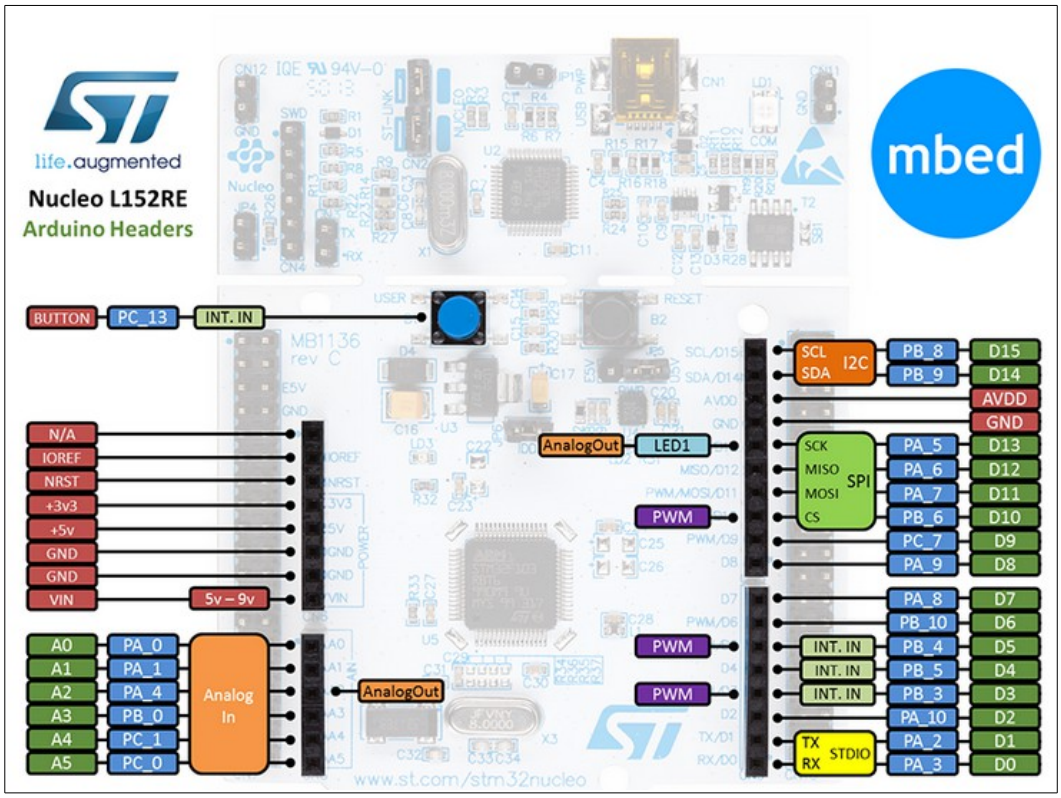
STM32 Nucleo – open development hardware supporting **Arduino**™ connectivity and **mbed**
STMicroelectronics is following a new path to support engineers in evaluation of MCUs and prototyping their applications.
The STM official link for NUCLEO board is **here** .
Nucleo boards includes an **ST-LINK/V2 embedded debug tool interface**.
After having shortlisted a microcontroller, the engineer starts an iterative process of prototyping, which may necessitate exchanging the microcontroller with a device of different characteristics.
STM32 Nucleo, ideally addresses this point, it is an open development tool which, at an RRP in the range of $10...$15, offers all you need to prototype an application.

The Nucleo boards available up to now are below.

Stay tuned to know the new release of Nucleo boards.



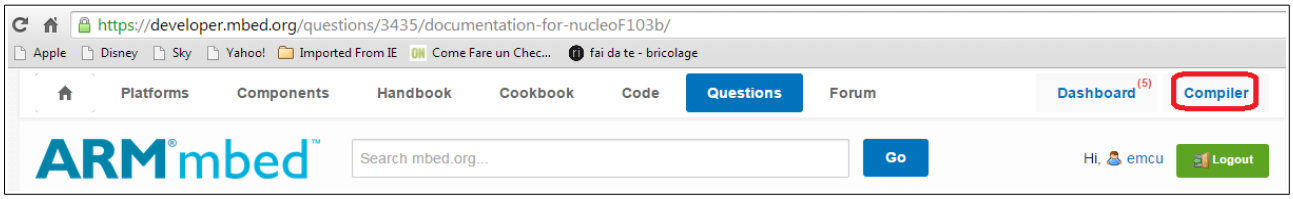
NUCLEO_L152RE and Arduino compatible headers – Fig.1



What is mBed

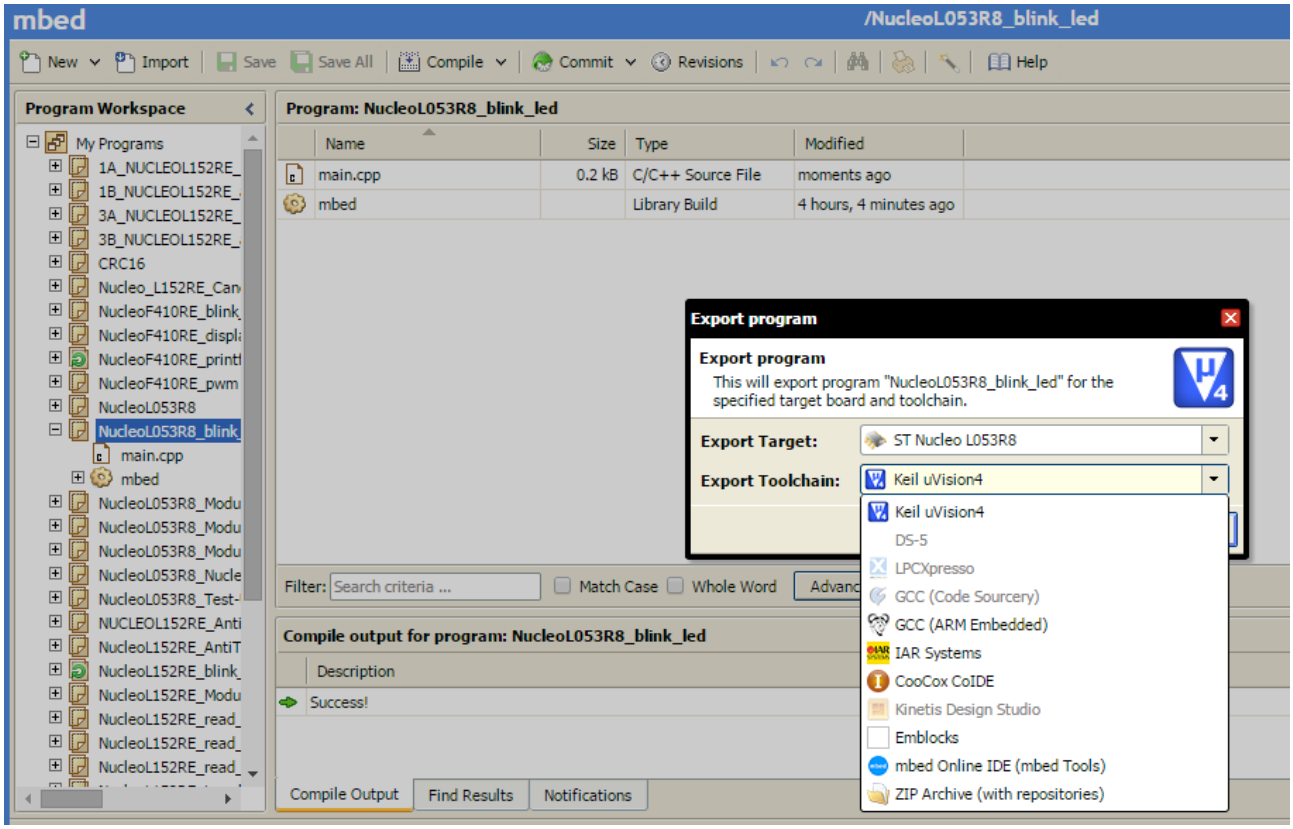
mBed is a free compiler of ARM that is **here**.
At the moment, mBed request an Internet connection because is a on line compiler.
At the moment, the only way to do the **debug** (using NUCLEO Boards) is to use **printf** and see the results on the PC. For this reason I suggest to use **TeraTerm** on PC.

The first thing that you must do is to register you on mBed and after the registration you have the possibility to create your project using the on line compiler.



mBed allows you to export the program to external IDE, that at the moment are:

- KEIL
- IAR
- CooCox
- GCC

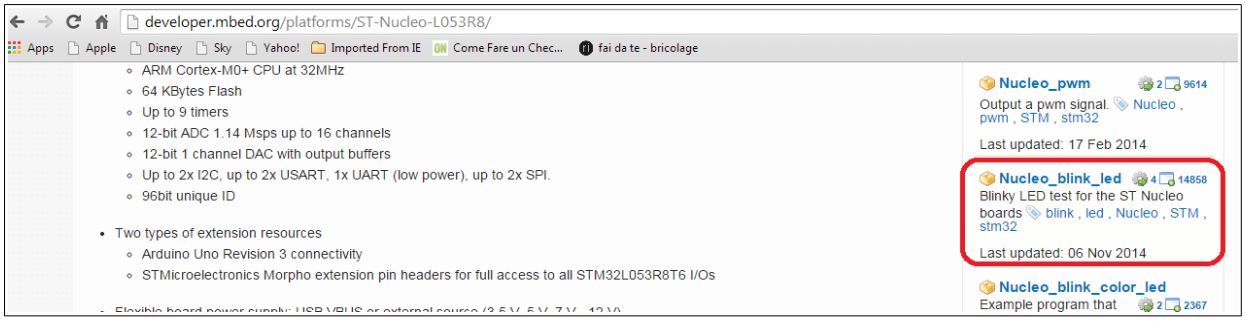


My first project in ten steps

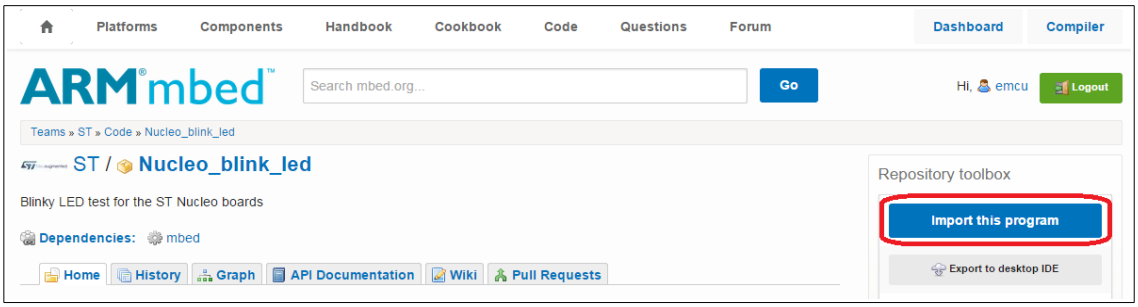
1. Choose the **NUCLEO board** that you need to use from the mBed PLATFORM page.
   Here there is the list of the NUCLEO boards.

2. I decided to use the **NUCLEO-L053R8** but you can choose what you want.
   From the page that appears you have all the informations regarding your NUCLEO board.

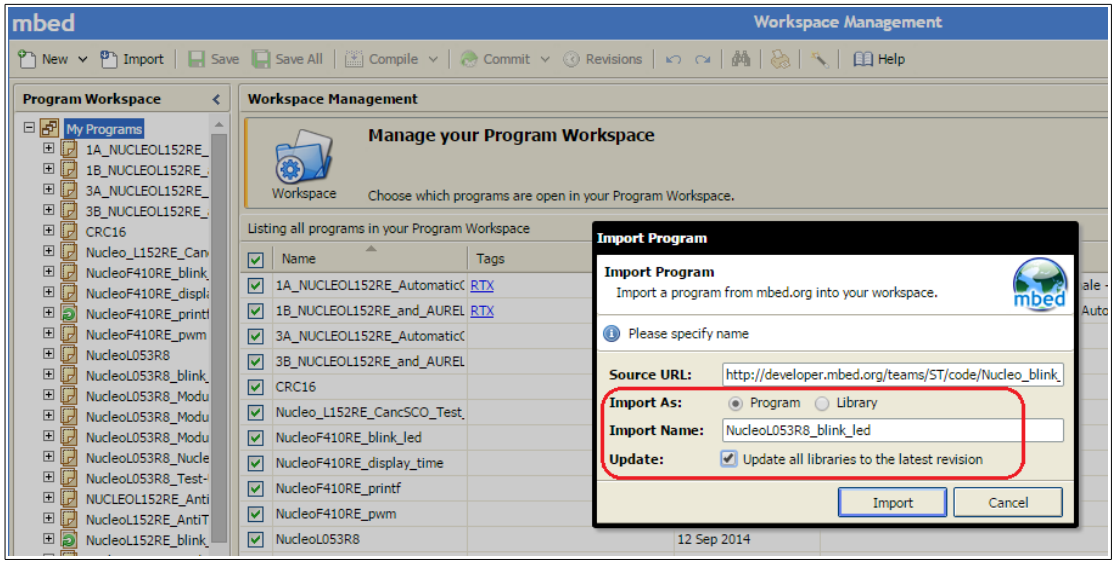
3. On the right of the page there are the Example programs, please choose the: **Nucleo_blink_led**



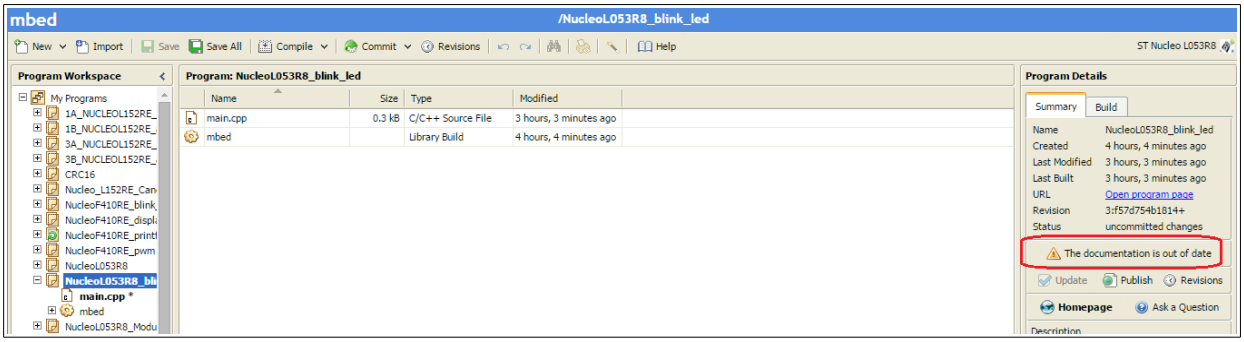
4. A new page will be opened and from this page choose: **Import this program**



5. At this point the compiler will start and you must see something like below.
   NOTE: I changed the original project name in: **NucleoL053R8_blink_led**
   Please select also: **Update all libraries to latest revision**
   At the end select: **Import**
   See below.



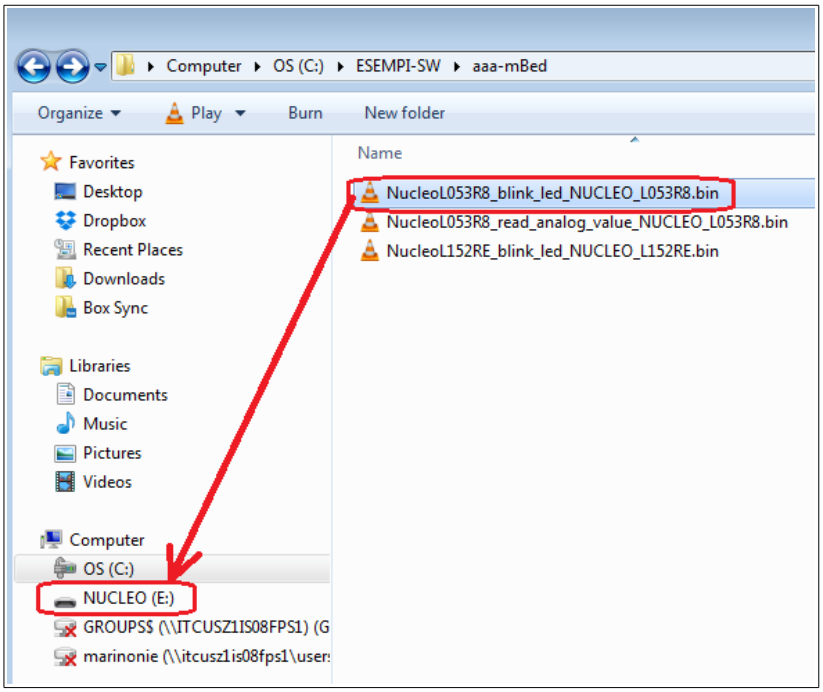
6. At the end of the import procedure you must see something like below.
   To UpDate the libraries click on: **The documentation is out of date**
   After this the flag will disappears.



7. Now click on the **COMPILE** icon.

8. At the end of the compilation, mBed asks us where save the **bin** file. Choose a directory and save it.

9. Now to program your NUCLEO board is only necessary to **drag and drop the bin file on the NUCLEO board** icon, see below.
   In other words: select the .bin file, drag it on the NUCLEO icon and release it.



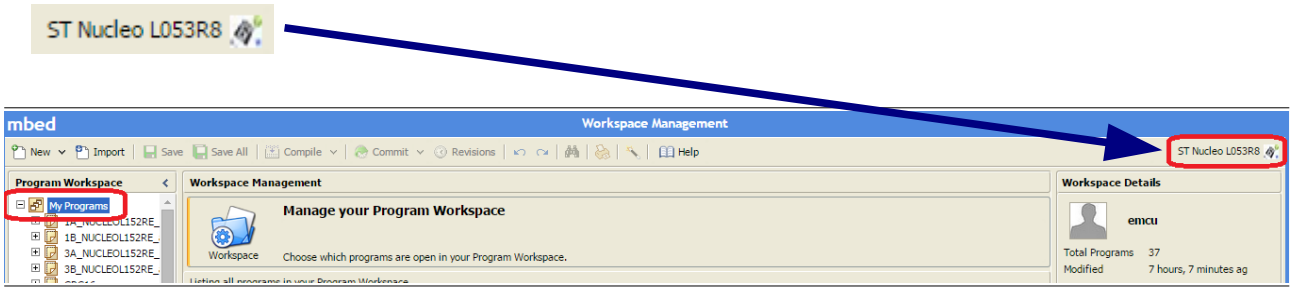
10. Now you must see the green LED that blinking.

Congratulation your first program is running.

Create a project from scratch

Now you are ready to create a project from scratch.

For this example we need to use the **Nucleo-F401RE**, to check if this platform is present click on this icon:



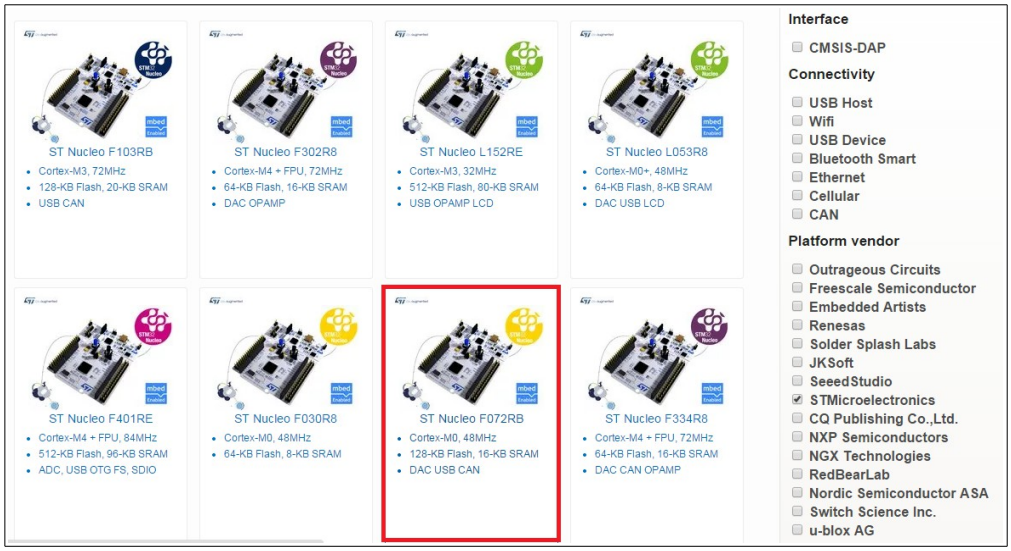
You must see something like below.
If NucleoL401RE is not present add it.

For insert a new platform click on the: **Add Platform** icon.

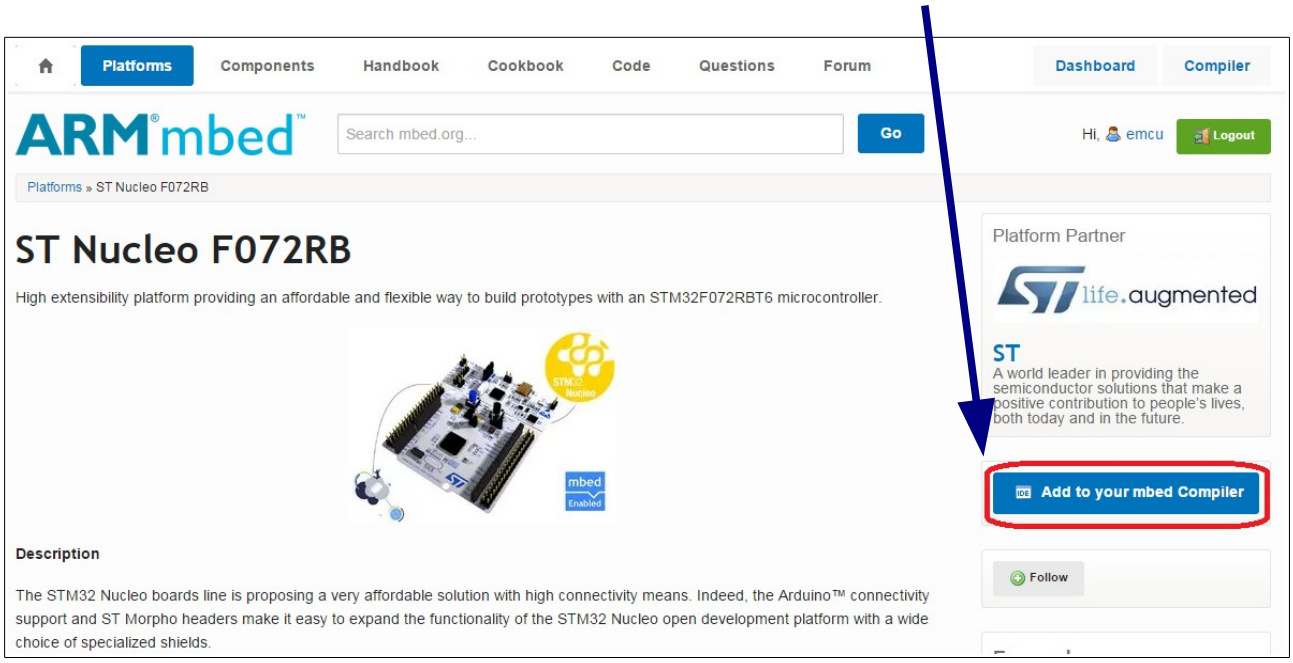
For example for add the **NUCLEO-F072RB** do this:

Press on the Add Platform icon.

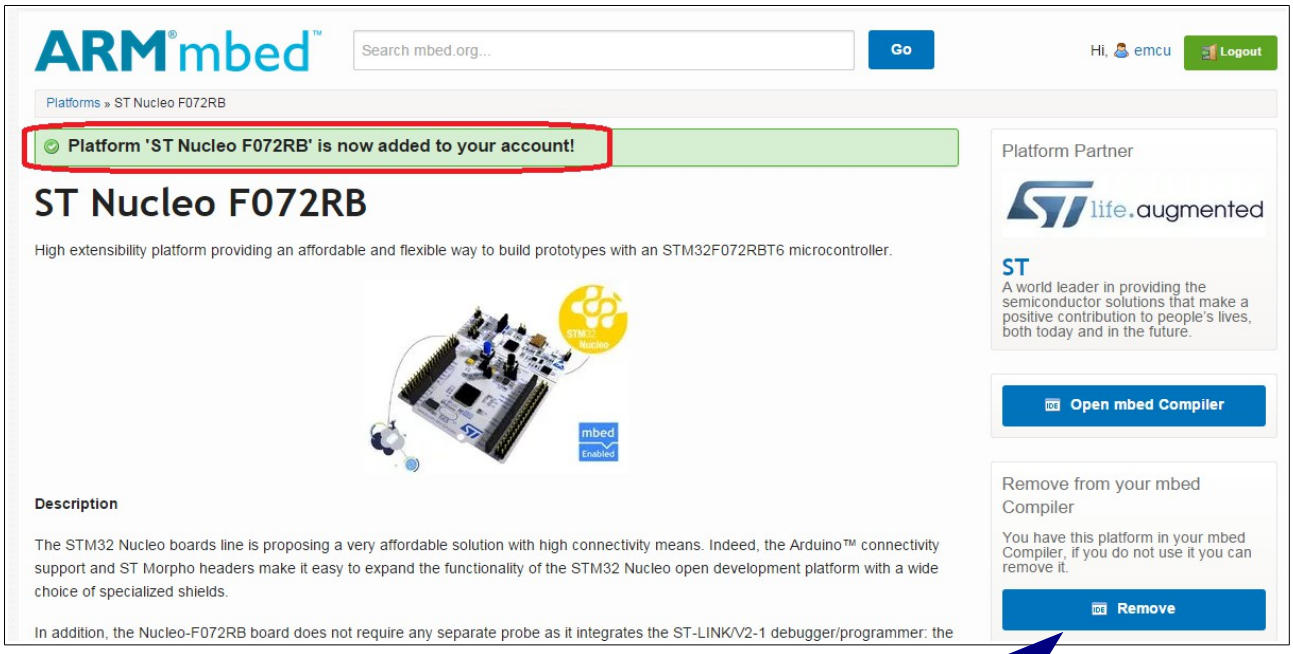
From the window that appears select the NUCLEO-F072RB



From the new window that appears select: **Add to your mbed Compiler**



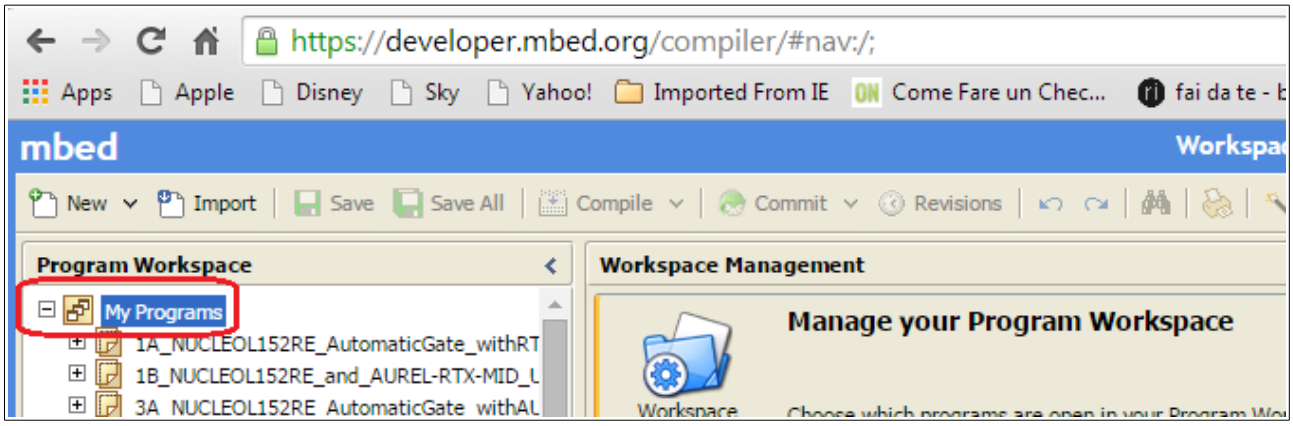
You must see a page like below.



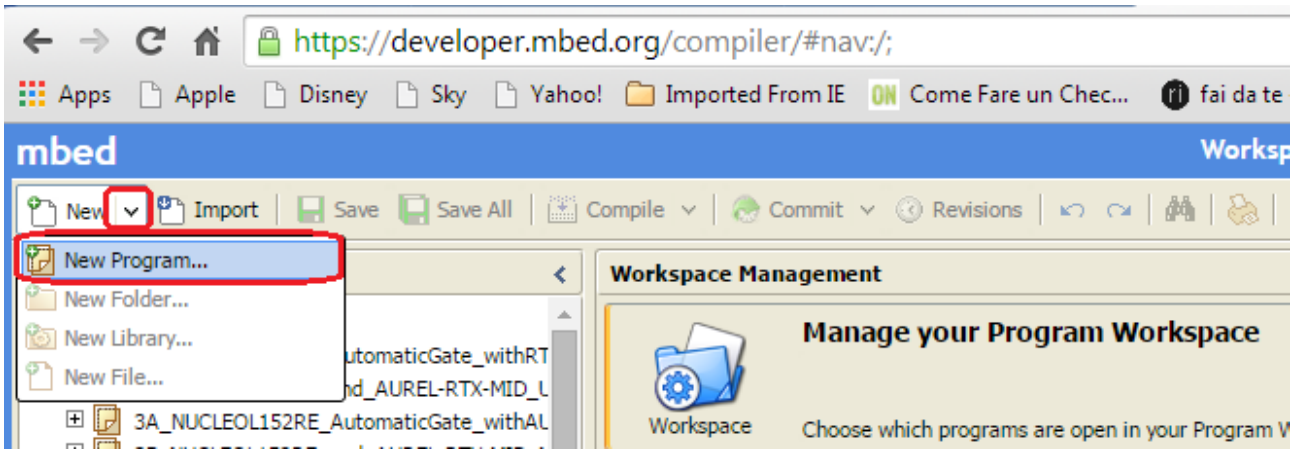
Note that it is also possible remove a platform from your compiler.

Ok, now we start to do a program from scratch using NUCLEO-F401RE.
For do this follow the steps below.

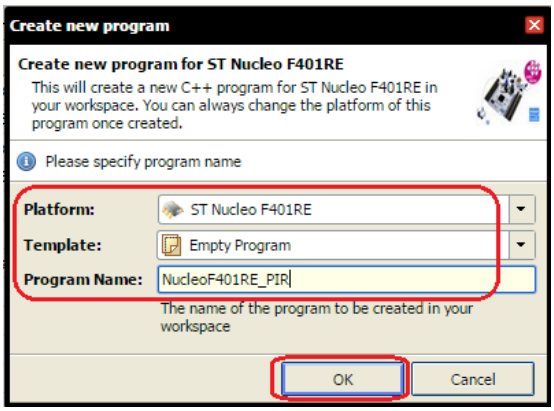
Select: **My Programs** (see below).



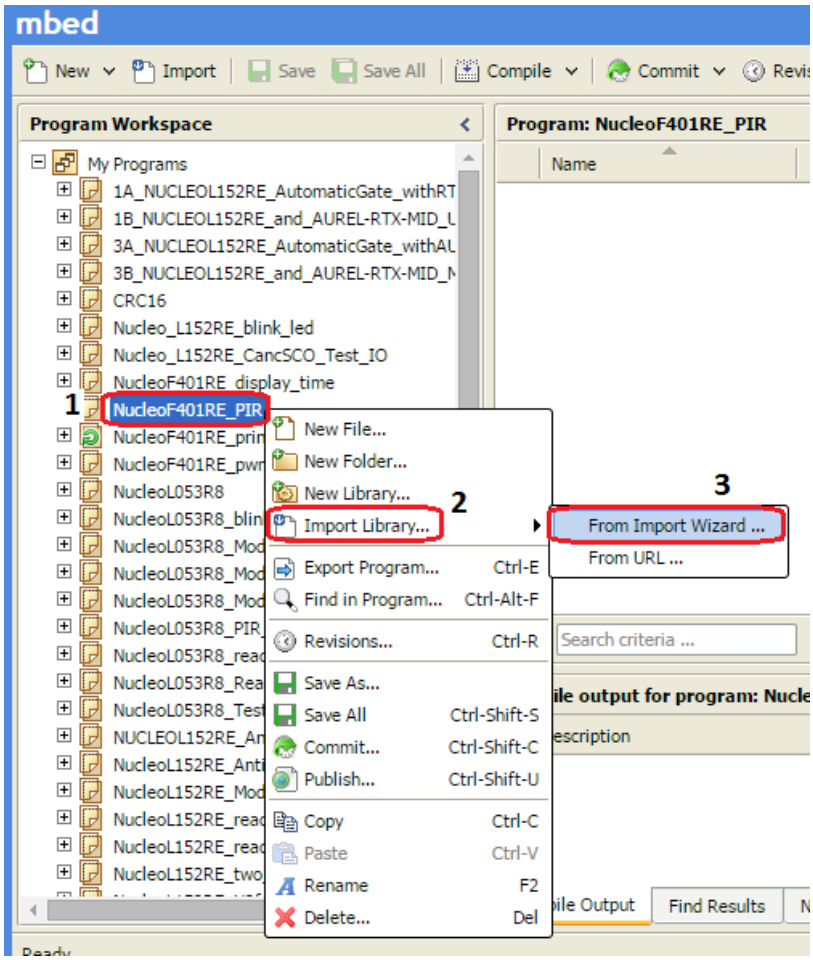
Select: **New Program...** (see the red boxes below).



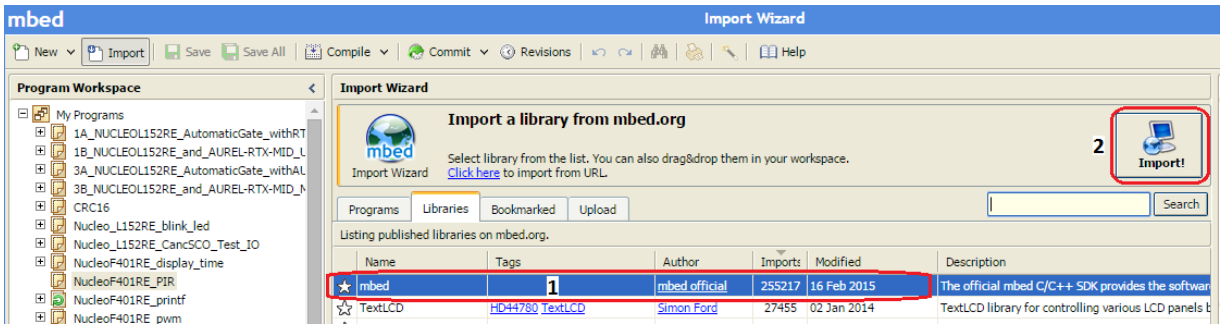
Now, from the new window that appears, select your **platform** (ST Nucleo F401RE), an **empty template** and a **name** for your project (NucleoF401RE_PIR), see below.
At the end of the configurations press **OK**.



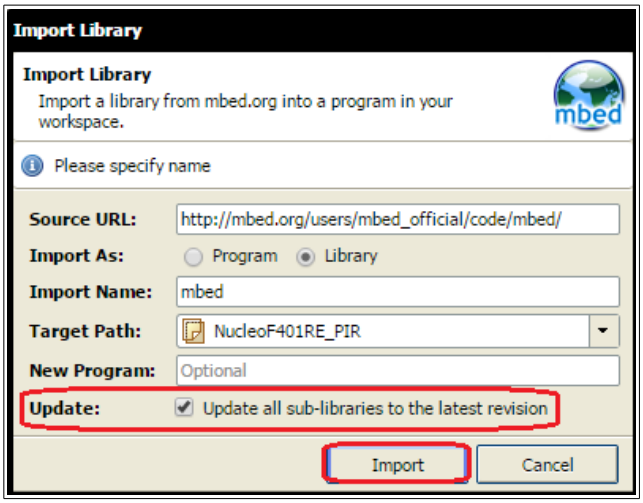
Now select your **project** (1), click on it with the right mouse button and choose: **Import Libraries** (2) and next select: **From Import Wizzard...**(3), see below.



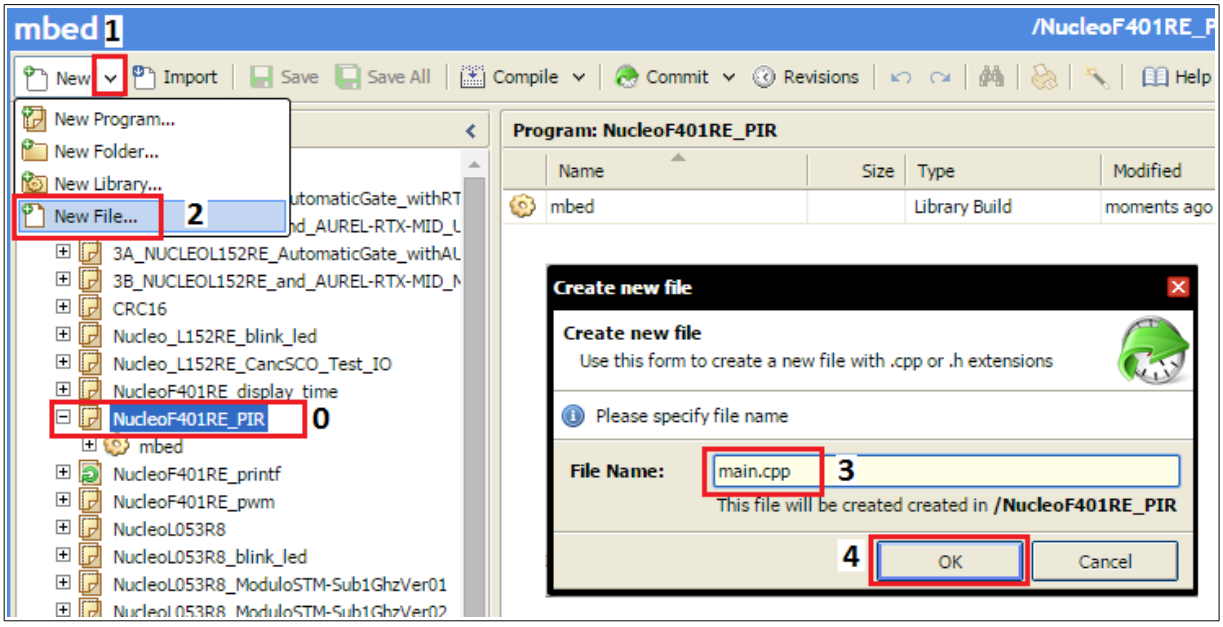
From the new window that appears select: **mbed** (1) and **Import** (2), see below..



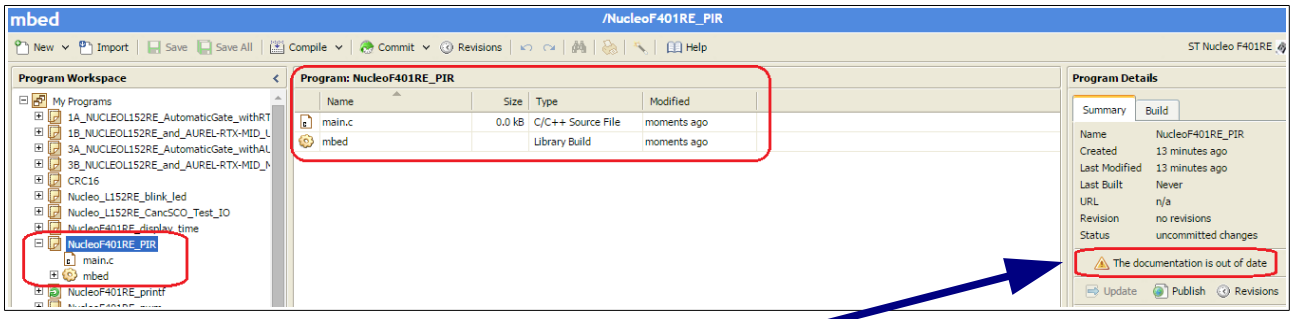
From the new window that appears select: **Update** and press **Import**, see below.



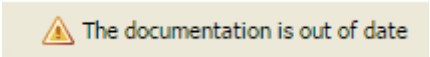
Now **select your project** (0) and **New File** (1 and 2) and write (in the box File Name) **main.cpp**, see below the steps 0, 1, 2, 3, 4.



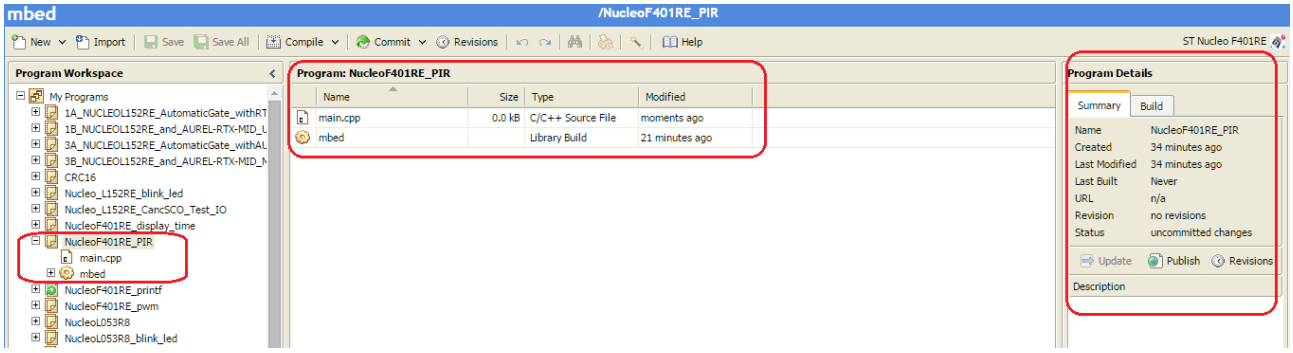
Now you must have something like below.



If there is the sentence:



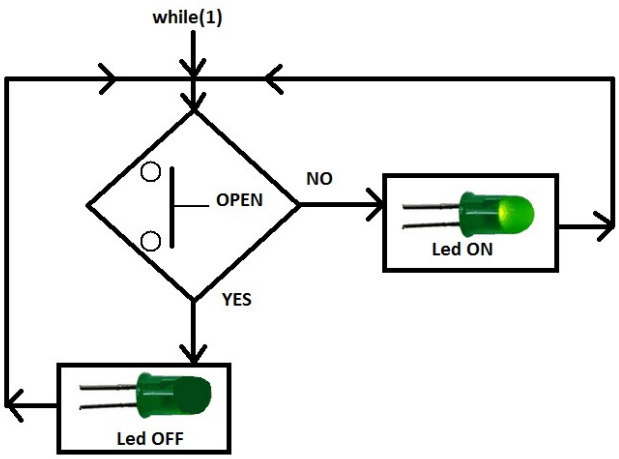
Please click on it to update all.
At the end you must see something like below.



To summarize:
We are created a new project for NUCLEO_F401RE named NucleoF401RE_PIR.

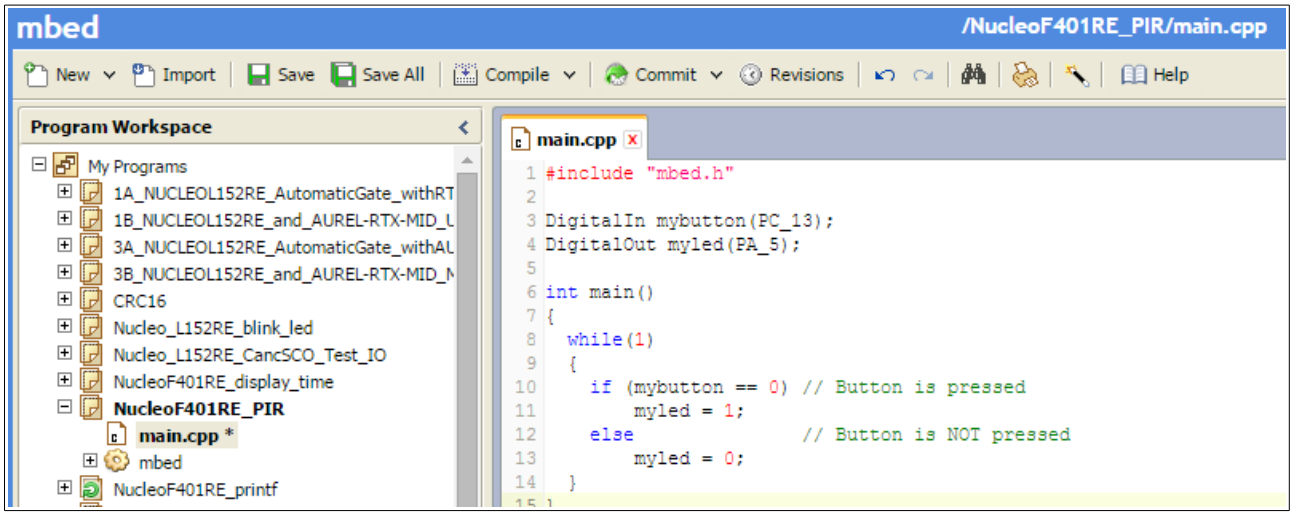
Now we want to use the Blue Push Button to Turn ON and Turn OFF the Green LED presents on the NUCLEO_F401RE board.

Flow diagram



For do this write, in the main.cpp, the below lines.

```
#include "mbed.h"

DigitalIn mybutton(PC_13);
DigitalOut myled(PA_5);

int main()
{
  while(1)
  {
    if (mybutton == 0) // Button is pressed
        myled = 1;
    else               // Button is NOT pressed
        myled = 0;
  }
}
```



Compile and **download** to NUCLEO-F401RE the **.bin** file generated.
See here how to download the .bin file to the NUCLEO boards.

Now if you press the Blue Button the Green LED goes ON, otherwise the Green LED is OFF.

Memory

mBed link

There are three type of memory that are:

FLASH
This is the non-volatile memory that primarily stores the program's instructions, and also any "constant" data values.
In general, you only read from this memory, and it is only written when you download new code to the mbed.

RAM
This is the volatile memory that is the working data space for storing all variables whilst the program is running. In C, this is static variables, the heap, and the stack.

EEPROM
Some microcontrollers have special non-volatile memory that can be erased and written byte for byte rather than in blocks or sectors. This memory is typically used by the application to store special data or configuration.

Variables

Variables are normally stored in **RAM** area and are **global** or **local**.

Global variables are defined before the main loop, see below.

```
int x = 5;
int main()
     {
     …
     …
     …
     }
```

Local variables are defined in a specific function, see below.

```
int Tst()
     {
     int x = 5;
     …
     …
     }
```

Variable CONST

If for some reason you have data that is fixed (such as a lookup table), you'd be much better off making sure the compiler can **allocate it in FLASH** to save valuable RAM space.
For do this there is the keyword: **const**
to tell the compiler that the variable will never be changed and in this way the variable is stored in Flash area.

```
const int x = 5;
int main()
     {
     …
     …
     …
     }
```

Debug using the printf via Virtual Com port (USB)

mBed Serial Com link

Up to now on NUCLEO boards, for debug your programs you must use **printf**.
To do this is very easy, follow the steps below.

First define the Virtual Com port to redirect the printf.
On NUCLEO boards I suggest to use the declaration shown below.

```
Serial pc(SERIAL_TX, SERIAL_RX);
```

By default the Virtual Comm configuration is:

9600-8-N-1 FlowControl None

Now you can use the **printf** to send to your PC the data you want. On PC I suggest to use TeraTerm. Below there is an example.

```
#include "mbed.h"
// Initialize a pins to perform Serial Communication for receive
// the result of the printf on PC (USB Virtual Com)
// I suggest to use TeraTerm on PC.
// TeraTerm configuration must be: 9600-8-N-1 FlowControl None
Serial pc(SERIAL_TX, SERIAL_RX);

DigitalOut myled(LED1);

int main() {

    pc.printf("RTC example\n");
    // Set RTC time to 16 December 2013 10:05:23 UTC
    set_time(1387188323);
    pc.printf("Date and time are set.\n");

    while(1) {

        time_t seconds = time(NULL);

        pc.printf("Time as a basic string = %s", ctime(&seconds));

        myled = !myled;
        wait(1);
    }
}
```

My examples that use USARTs

- How to use USART2 on NUCLEO-L152RE and Mbed
- How to use USART2 and USART1 on NUCLEO-L152RE and Mbed
- My mBed examples are **here**

Printf %c, %d, %x, %f, %e, \n, \r, etc

Reference are **here**.

specifier	Output	Example
d *or* i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
a	Hexadecimal floating point, lowercase	-0xc.90fep-2
A	Hexadecimal floating point, uppercase	-0XC.90FEP-2
c	Character	a
s	String of characters	sample
p	Pointer address	b8000000
n	Nothing printed. The corresponding argument must be a pointer to a signed int. The number of characters written so far is stored in the pointed location.	
%	A % followed by another % character will write a single % to the stream.	%

The reference are **here**.

- \n (newline)
- \t (tab)
- \v (vertical tab)
- \f (new page)
- \b (backspace)
- \r (carriage return)

USART functions

mBed Serial Com link

Public Member Functions

	Serial (PinName tx, PinName rx, const char *name=NULL) 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, Parity parity=SerialBase::None, 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.
void	**attach** (void(*fptr)(void), IrqType type=RxIrq) Attach a function to call whenever a serial interrupt is generated.
template<typename T > void	**attach** (T *tptr, void(T::*mptr)(void), IrqType type=RxIrq) Attach a member function to call whenever a serial interrupt is generated.
void	**send_break** () Generate a break condition on the serial line.
void	**set_flow_control** (Flow type, PinName flow1=NC, PinName flow2=NC) Set the flow control type on the serial port.

Write a message to a device (for example PC) **at a 19200 baud**

```
#include "mbed.h"

Serial device(SERIAL_TX, SERIAL_RX);

int main() {
    device.baud(19200);
    device.printf("Hello World\n");
}
```

Provide a serial pass-through between the PC and an external UART

```
#include "mbed.h"

Serial pc(SERIAL_TX, SERIAL_RX);    // This is USART2 tx, rx
Serial device(PB_6, PA_10);         // This is USART1 tx, rx

int main() {
    while(1) {
        if(pc.readable()) {
            device.putc(pc.getc()); // Get from PC and
                                    // Send to USART1
        }
        if(device.readable()) {
            pc.putc(device.getc()); // Get from device (USART1)
                                    // and Send to PC
        }
    }
}
```

List of the mBed functions

mBed manual

Analog I/O
• AnalogIn - Read the voltage applied to an analog input pin
• AnalogOut - Set the voltage of an analog output pin

Digital I/O
• DigitalIn - Configure and control a digital input pin.
• DigitalOut - Configure and control a digital output pin.
• DigitalInOut - Bi-directional digital pins
• BusIn - Flexible way to read multiple DigitalIn pins as one value
• BusOut - Flexible way to write multiple DigitalOut pins as one value
• BusInOut - Flexible way to read/write multiple DigitalInOut pins as one value
• PortIn - Fast way to read multiple DigitalIn pins as one value
• PortOut - Fast way to write multiple DigitalOut pins as one value
• PortInOut - Fast way to read/write multiple DigitalInOut pins as one value
• PwmOut - Pulse-width modulated output
• InterruptIn - Trigger an event when a digital input pin changes.

Timers
• Timer - Create, start, stop and read a timer
• Timeout - Call a function after a specified delay
• Ticker - Repeatedly call a function
• wait - Wait for a specified time
• time - Get and set the realtime clock

Digital Interfaces
• Serial - Serial/UART bus
• SPI - SPI bus master
• SPISlave - SPI bus slave
• I2C - I²C bus master
• I2CSlave - I²C bus slave
• CAN - Controller-area network bus

Real-time Operating System
• mbed RTOS

File System
• LocalFileSystem - Using the mbed disk as storage from within a program
• SDFileSystem - Using the mbed disk as storage from within a program

USB

• **USBDevice** - Using mbed as a USB Device ◦USBMouse - Emulate a USB Mouse with absolute or relative positioning

◦ USBKeyboard - Emulate a USB Keyboard, sending normal and media control keys

◦ USBMouseKeyboard - Emulate a USB Keyboard and a USB mouse with absolute or relative positionning

◦ USBHID - Communicate over a raw USBHID interface, great for driverless communication with a custom PC program

◦ USBMIDI - Send and recieve MIDI messages to control and be controlled by PC music sequencers etc

◦ USBSerial - Create a virtual serial port over the USB port. Great to easily communicate with a computer.

◦ USBAudio - Create a USBAudio device able to receive audio stream from a computer over USB.

◦ USBMSD - Generic class which implements the Mass Storage Device protocol in order to access all kinds of block storage chips

• **USBHost** - Using mbed to act as USBHost ◦USBHostMouse - Receive events from a USB mouse

◦ USBHostKeyboard - Read keycode-modifier from a USB keyboard

◦ USBHostMSD - Read-write a USB flash disk

◦ USBHostSerial - Communicate with a virtual serial port

◦ USBHostHub - You can plug several USB devices to an mbed using a USB hub

Networking

• Ethernet - Ethernet network ◦Ethernet Interface

◦ TCP/UDP Socket API

◦ TCP/IP Protocols and APIs

Digital In

mBed Digital In manual

Public Member Functions	
	DigitalIn (PinName pin) Create a **DigitalIn** connected to the specified pin.
	DigitalIn (PinName pin, PinMode mode) Create a **DigitalIn** connected to the specified pin.
int	**read** () Read the input, represented as 0 or 1 (int)
void	**mode** (PinMode pull) Set the input pin mode.
int	**is_connected** () Return the output setting, represented as 0 or 1 (int)
	operator int () An operator shorthand for **read()**

Define an **Input Pin** connected to a specific Pin

Syntax:

```
DigitalIn Name(Pin);
```

The **Pin** must be the name shown in the Blue Labels in the fig.1
Name is as you want.

Example:
```
DigitalIn mybutton(USER_BUTTON);
```
The USER_BUTTON is a Blue Button present on NUCLEO boards.

Example:
```
DigitalIn IN1(PA_10);
```

To define the **PullUp**, **PullDown**, **PullNone**, the syntax is:

```
Name.mode(Pull_Mode);
```

Example
```
mybutton.mode(PullUp);
```

Complete example is below

```
#include "mbed.h"
// By www.emcu.it
//-------------------------------------
// Hyperterminal configuration
// 9600 bauds, 8-bit data, no parity
//-------------------------------------

Serial pc(SERIAL_TX, SERIAL_RX);

DigitalIn mybutton(PA_10);
DigitalOut myled(LED1);

int main() {
  mybutton.mode(PullUp);
  while(1) {
    if (mybutton == 0)
    { // Button is pressed
      myled = 1;
      pc.printf("Button is PRESSED\n");
      wait(0.2); // 200 ms
    }
    else
      myled = 0;
  }
}
```

Digital Out

mBed Digital Out manual

Public Member Functions	
	DigitalOut (PinName pin) Create a **DigitalOut** connected to the specified pin.
	DigitalOut (PinName pin, int value) Create a **DigitalOut** connected to the specified pin.
void	**write** (int value) Set the output, specified as 0 or 1 (int)
int	**read** () Return the output setting, represented as 0 or 1 (int)
int	**is_connected** () Return the output setting, represented as 0 or 1 (int)
DigitalOut &	**operator=** (int value) A shorthand for **write()**
	operator int () A shorthand for **read()**

Define an **Output Pin** connected to a specific Pin

Syntax:

```
DigitalOut Name(Pin);
```

The **Pin** must be the name shown in the Blue Labels in the fig.1
Name is as you want.

After the declaration of the output pin, is possible to set it to 1 or to 0.
See the example below.

Example:

```
DigitalOut Rele(PA_8);

Rele = 1; // The output pin named Rele is set to 1
```

NOTE:

To invert the status of the DigitalOut pin the syntax is:

```
Rele = !Rele
```

A complete Example

```
#include "mbed.h"

DigitalOut myled(LED1);

int main()
{
    while(1)
     {
        myled = 1;  // LED is ON
        wait(0.2);  // 200 ms
        myled = 0;  // LED is OFF
        wait(1.0);  // 1 sec
     }
}
```

ADC

mBed ADC manual

Public Member Functions	
	AnalogIn (PinName pin) Create an **AnalogIn** , connected to the specified pin.
float	**read** () Read the input voltage, represented as a float in the range [0.0, 1.0].
unsigned short	**read_u16** () Read the input voltage, represented as an unsigned short in the range [0x0, 0xFFFF].
	operator float () An operator shorthand for **read()**

Define an **Analog Input** (**ADC**) connected to a specific Pin.

Declaration:

```
AnalogIn Name(Pin);
```

The **Pin** must be the name shown in the Blue Labels in the fig.1
Name is as you want.

Example

```
AnalogIn Ana_In1(A0);
```

After the declaration of the analog pin you are able to read an analog value.

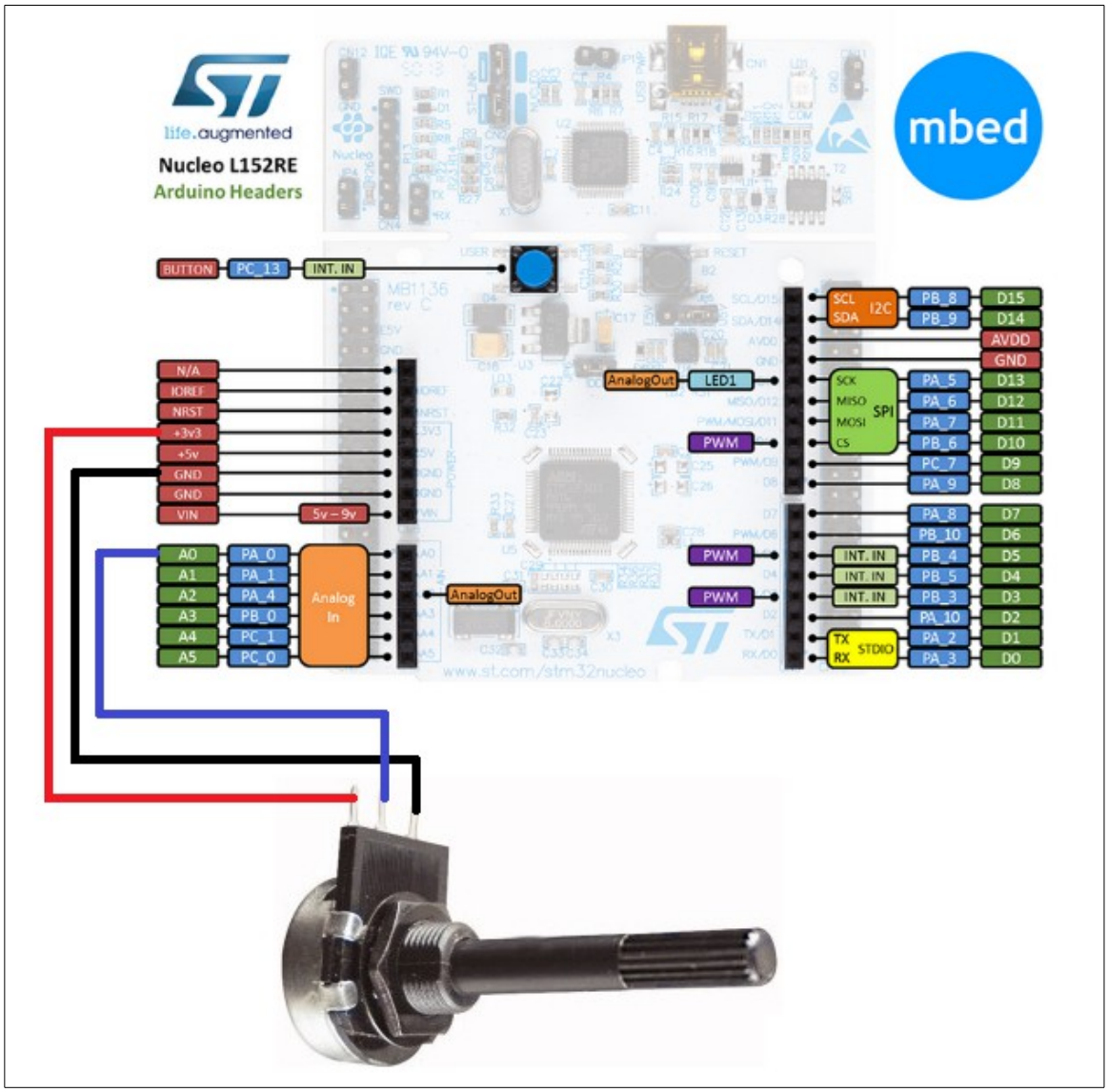
Example

```
unsigned long value  = 0;

value = Ana_In1.read_u16();
```

Complete example for NUCLEO_L152RE is below.

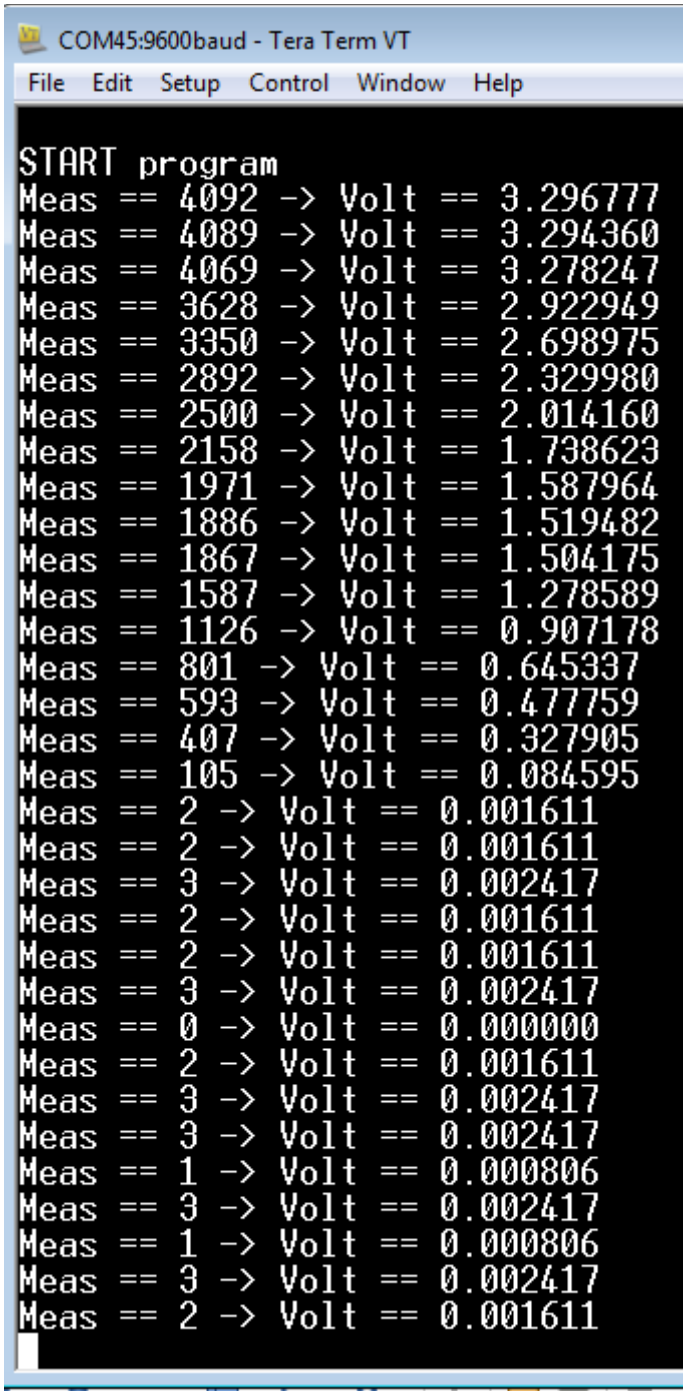
```
//
// By www.emcu.it
// Tested on NUCLEO_L152RE
//

#include "mbed.h"

AnalogIn analog_value(A0);          // Analog InPut
DigitalOut myled(LED1);             // Digital OutPut
Serial pc(SERIAL_TX, SERIAL_RX);    // Default USART is: 8N1 NO FlowControl

// Variables ------------------------------------------------------------
uint16_t meas = 0;
float_t  Result_V = 0;
float_t  MinStepRes = 0;

// ATTENTION
// The two value below, must be changed in according to the Vcc and ADC
// resolution
float_t  Valim  = 3300; // This is the supplay voltage of ADC (or MCU)
float_t  ADCres = 4096; // This is the ADC resolution that in this case si 12Bit
                        // All NUCLEO boards use an ADC with 12bit resolution

// Define ---------------------------------------------------------------
// Calculate the corresponding acquisition measure for a given value in mV
#define MV(x) ((0xFFF*x)/3300)

int main()
{
    pc.printf("\n\r\n\rSTART program\n\r");
    while(1)
    {
        // Read the analog input value
        meas = analog_value.read_u16();

        if (meas > MV(1000)) // If the value is greater than 1000 toggle the LED
        {
          myled = !myled;
        }
        else
          myled = 0;

        // Convert meas in Volt and put it in Result_V
        MinStepRes = (Valim / ADCres);
        Result_V = ((MinStepRes * meas))/1000;
        // Display the result via Virtual COM
        pc.printf("Meas == %d -> Volt == %f\n\r", meas, Result_V);

        wait(0.8); // 800 ms
    }
}
```

Electrical connections of the potentiometer to NUCLEO-L152RE



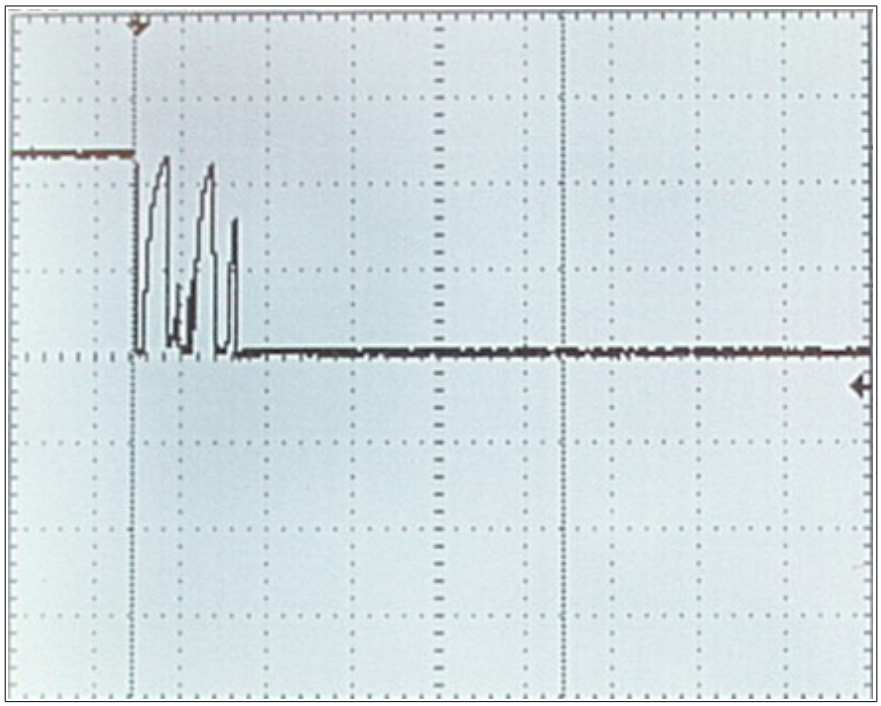
If you run on PC the TeraTerm you must see something like below, during the rotation of the potentiometer.

TeraTerm configuration must be:
9600-8-N-1
FlowControl None



DEBOUNCE

When a switch is pressed, there is a period of time in which the electrical connection "bounces" between open and closed.
It is important to choose good Debounce Time for filtering the digital input noise.
Normaly the Deboundance Time must be from 5 to 30mS.



Typically there are two methods employed to debounce a switch that are:

- using timer and interrupt to test the state of the switch
- using polling method to test the state of the switch

In the example below the use the polling method.

```
#include "mbed.h"
// By www.emcu.it
// see: http://www.emcu.it/NUCLEOevaBoards/NUCLEOevaBoards.html

//-----------------------------------------------
// Hyperterminal configuration
// 9600 bauds, 8-bit data, no parity, 1 stop bit
//-----------------------------------------------

Serial pc(SERIAL_TX, SERIAL_RX);

DigitalIn mybutton(PA_10);
DigitalOut myled(LED1);

int main()
{
  mybutton.mode(PullUp);
  while(1)
     {
     if (mybutton == 0)
          { // Button is pressed, we do the delay for Debounce
           wait(0.2); // Wait 200 ms for debounce
           if (mybutton == 0)
               {
               myled = 1;
               pc.printf("Button is PRESSED\n");
               }
          }
    else
          myled = 0;
     }
}
```

INTERRUPT

mBed Interrupt link

I started from this **link** to develop my Interrupt custom functions that are below.
The functions below are tested on NUCLEO-L152RE.

Function for flashing a led.

```
#include "mbed.h"

Timeout to1;
DigitalOut myled(LED1);     // This LED is on NUCLEO-L153RE

#define DLYFlash 0.5

void IntFlash(void);

int main()
{
    to1.attach(&IntFlash, DLYFlash);

    while(1)
    {
    }
}


void IntFlash(void) {
    myled = !myled;
    to1.detach();
    to1.attach(&IntFlash, DLYFlash);
}
```

Sometimes it is necessary flashing a led to highlight an operation and stop the flashing at the end of the operation.
The function below does this.

```
void IntFlash(void) {
    if (ONOFF_Flashing == ON)
        myled = !myled;
    else
        myled = 0;
    to1.detach();
    to1.attach(&IntFlash, DLYFlash); // this line reload Interrupt
}
```

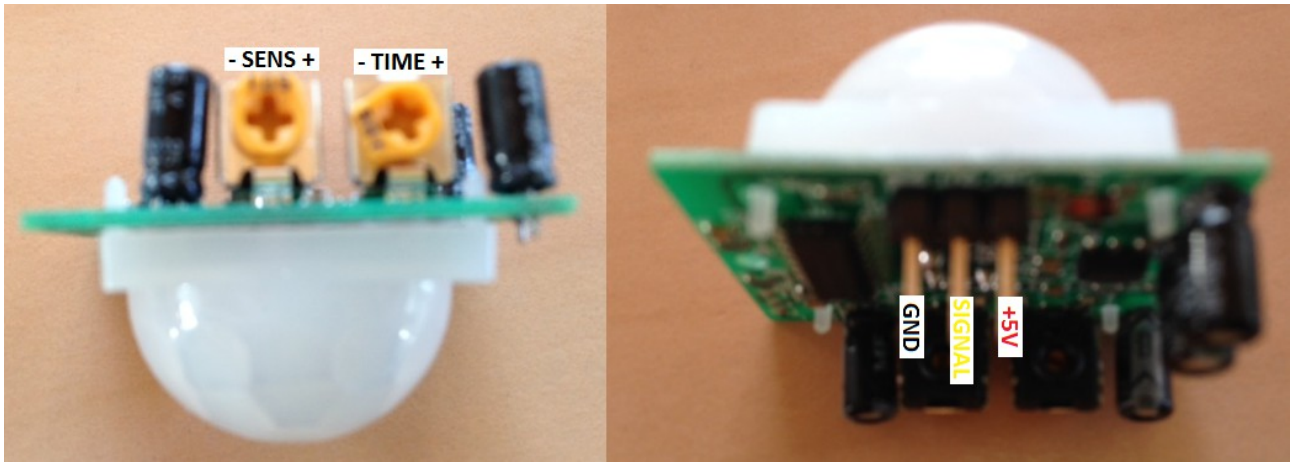
The **ONOFF_Flashing**, enables or disables the OutPut **`myled`**.
A complete example is below.

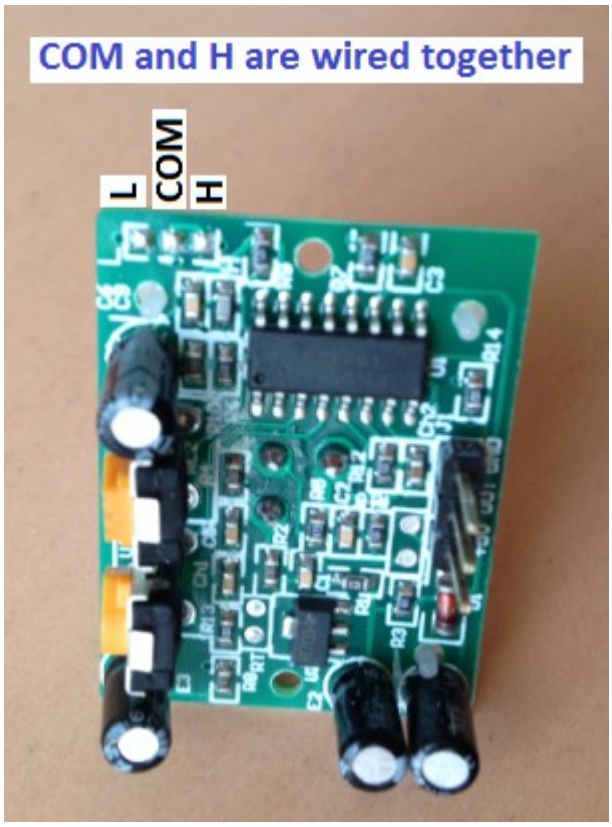
```
#include "mbed.h"

Timeout to1;
// This LED is on NUCLEO-L153RE
DigitalOut myled(LED1);
// This is Blue-Button and is on NUCLEO-L153RE
DigitalIn BlueButton(USER_BUTTON);


#define DLYFlash 0.2
#define OFF 0
#define ON 1

int ONOFF_Flashing = OFF;

void IntFlash(void);

int main()
{
    to1.attach(&IntFlash, DLYFlash);

    while(1)
    {
        if (BlueButton == 0)
            ONOFF_Flashing = ON;
        else
            ONOFF_Flashing = OFF;
    }
}

void IntFlash(void) {
    if (ONOFF_Flashing == ON)
        myled = !myled;
    else
        myled = 0;
    to1.detach();
    to1.attach(&IntFlash, DLYFlash); // this line reload Interrupt
}
```

How to use PIR sensor (Digital Infrared Motion Sensor Board) **and NUCLEO** (F401re)

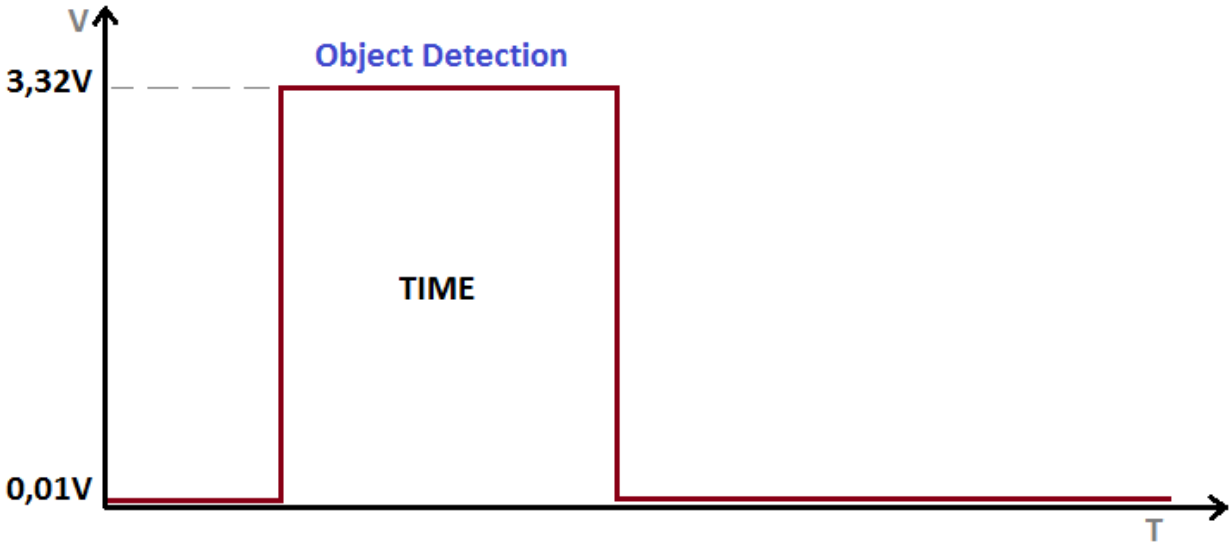
This is a simple example that show the way to use the low cost PIR board (Digital Infrared Motion Sensor) and NUCLEO board.
The SW below is tested on **NUCLEO-F401RE**.

I'm used a low cost PIR sensor board (< 3$ - see on Internet to find it).





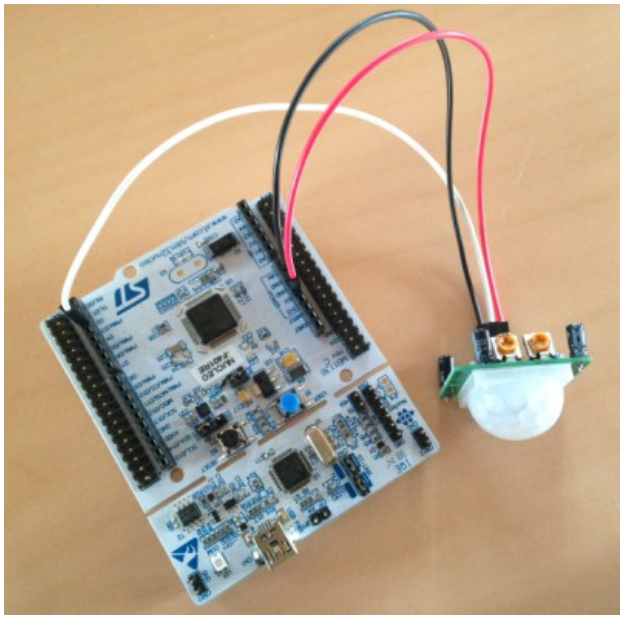
Pin	Status	Mode	Comments
1	L）	Unrepeatable trigger mode	
1	H）	Repeatable trigger mode	duration adjustment by potentiometer

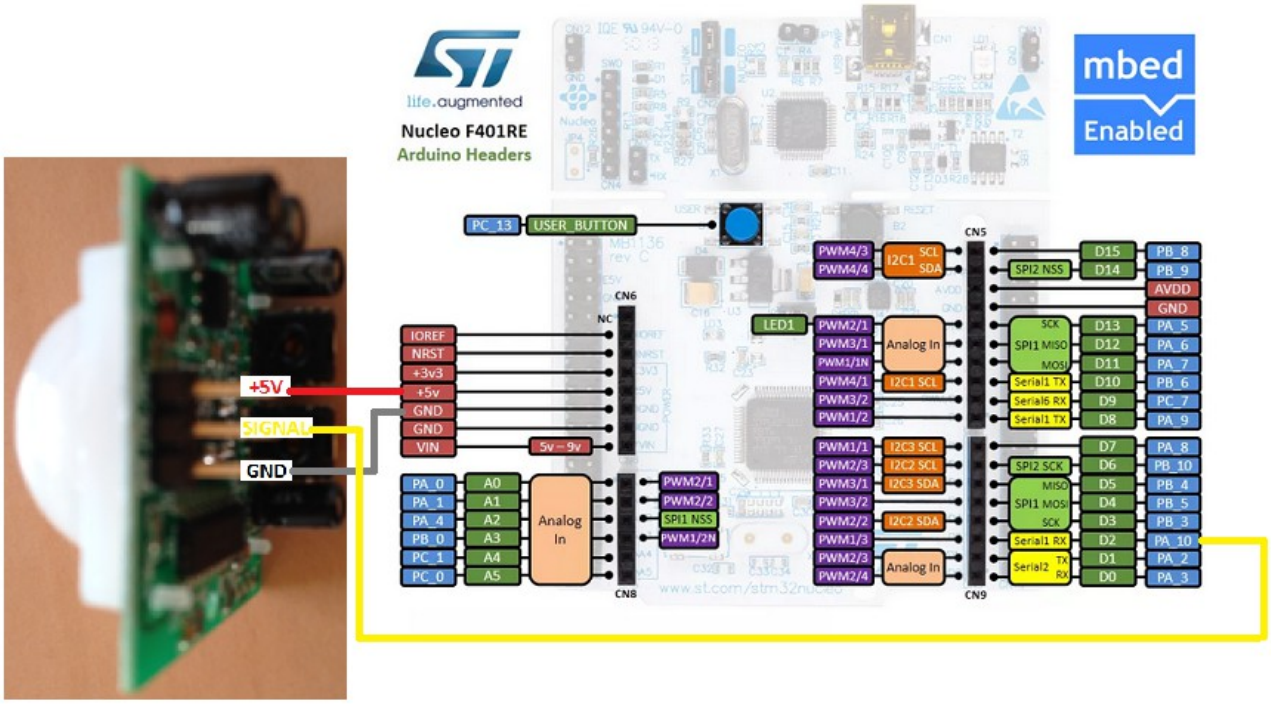


TIME (duration) depend of the value of the TIME trimmer.

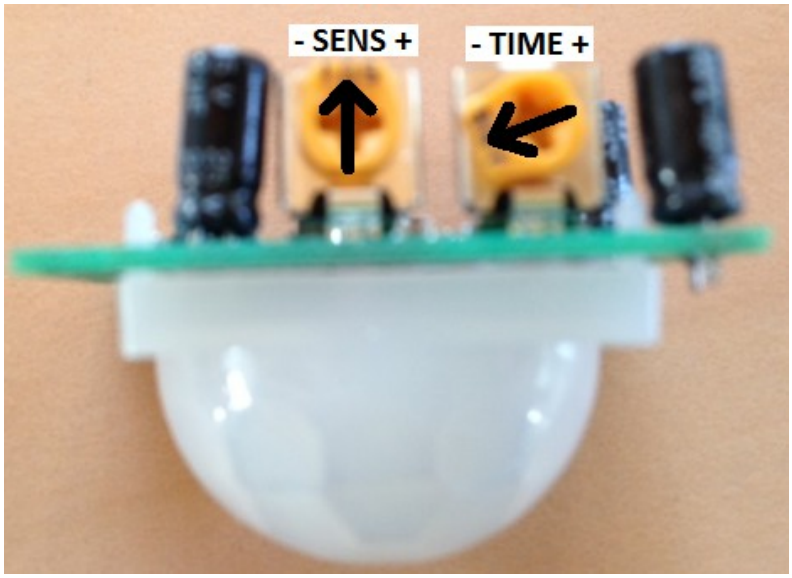
Power supply of the sensor is **5V**.

The PIR board and NUCLEO board connections are below.





Configure the trimmer as show below.
SENS medium
TIME minimum



The SW for mBed is below.

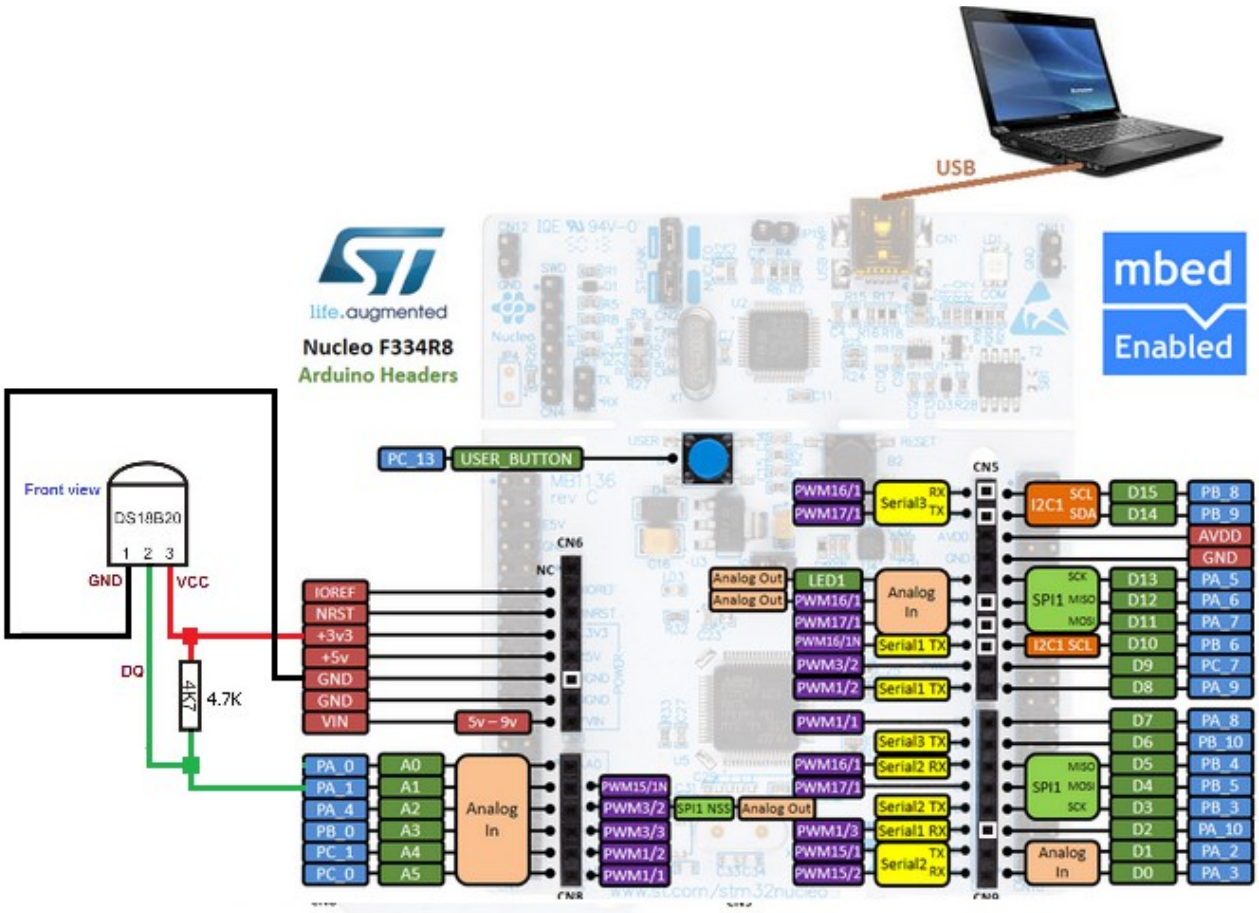
```
//
// By www.emcu.it
// Feb.2015
// Tested on NUCLEO-F410RE
//

#include "mbed.h"

DigitalOut myled(LED1);
DigitalIn Sensore(PA_10);

int n=0;

int main()
    {
    // Delay for waiting the PIR stabilization (30 sec.)
    for (n=0; n<30; n++)
        {
        wait_ms(1000);       // 1sec delay
        myled = !myled;      // Blinking LED1
        }
    myled = 0;               // LED is OFF

    while(1)
    {
        if (Sensore == 1)
        {
            myled = 1;       // LED1 is ON
            wait_ms(20000);  // LED1 stay ON for 20 sec
        }
        else
            myled = 0; // LED is OFF
    }
}
```

How to use the DS18B20 on the NUCLEO-F334R8 and see the results on the PC

Temperature measurement made using an **NUCLEO-F334R8** and **DS18B20**

The source code of this example is available **here**.

The connections are shown below

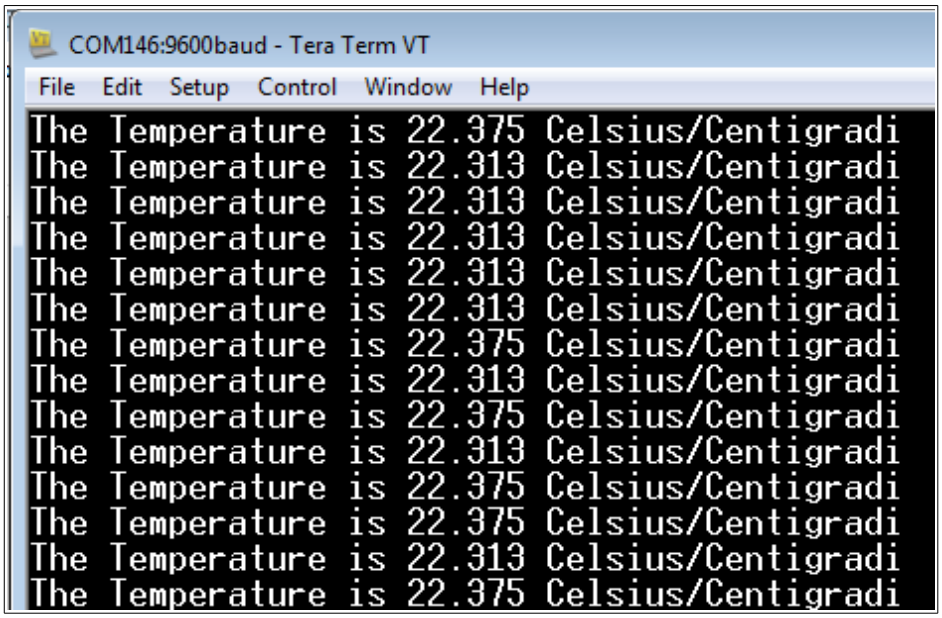


The USART parameters are:

Baud Rate: **9600**
Data: **8**
Parity: **NONE**
Stop: **1**
Flow Control: **NONE**

How to use this example

Connect the DS18B20 to the NUCLEO-F334R8 board.
Connect the NUCLEO-F334R8 to the PC.
On the PC, run the TeraTerm and see the result that must be similar to my result shown below.

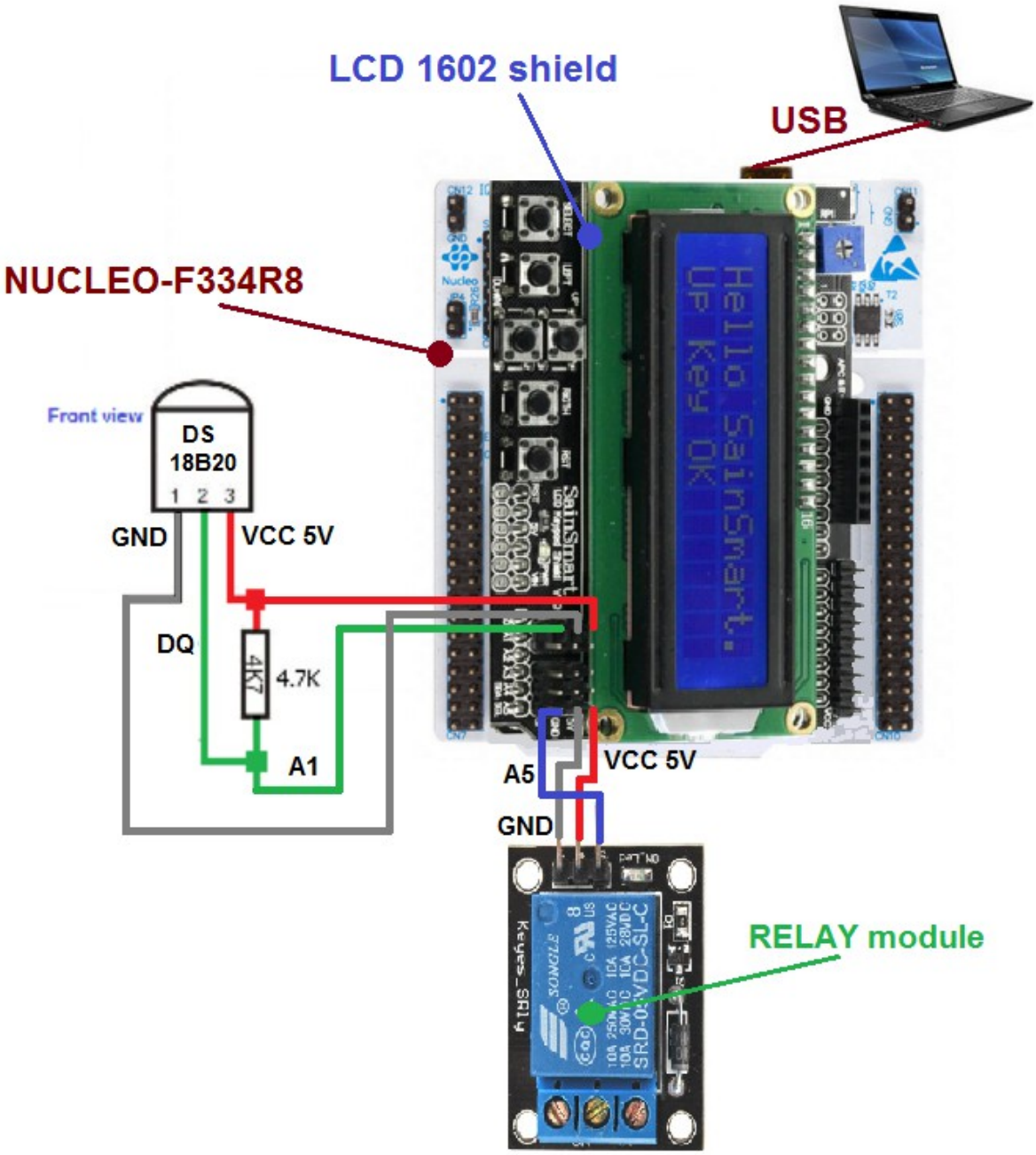


Temperature control based on:

NUCLEO-F334R8
DS18B20
RELAY module
LCD1602 shield (for more info see here)

The source code of this example is available **here**.

The connections are shown below



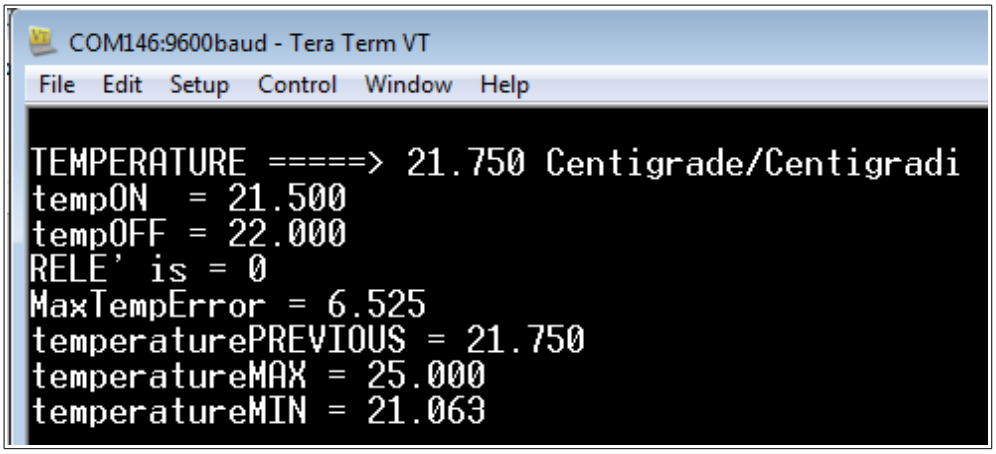
- **DS18B20 is connected to A1**
- **RELAY is connected to A5**

The USART parameters are:

Baud Rate: **9600**
Data: **8**
Parity: **NONE**
Stop: **1**
Flow Control: **NONE**

How to use this example

Connect the NUCLEO-F334R8 to the PC.
On the PC, run the TeraTerm and see the result that must be similar to my result shown below.



TEMPERATURE - is the actual ambient temperature, it is in Celsius degrees.
Celsius, historically known as centigrade.
tempON – if the ambient temperature is equal or minor (<=), the relay is turned on.
tempOFF – if the ambient temperature is equal or major (>=), the relay is turned off.
RELE' – is the status of the relay, 0 == OFF, 1 == ON.
MaxTempError - is a parameter calculated in this way:
`MaxTempError = (temperature * MaxError) / 100;`
If the new temperature measured exceed this value is not considered (is an error) and the actual temperature is set to the `temperaturePREVIOUS`.
temperaturePREVIOUS - is the previous temperature measured.
temperatureMAX - is the maximum temperature measured
temperatureMIN - is the minimum temperature measured

On the LCD1602 shield you must see something like below (**Principal Menu**).

T 21.750 - is the temperature in Celsius degree.
RELEon - means that the relay is ON
TM25.000 - means that the maximum temperature measured was 25.000 Celsius degree.
Tm21.063 - means that the minimum temperature measured was 21.063 Celsius degree.

For navigate in the menu there are two buttons available:
RIGHT →
LEFT ←

LCD backlight control from the **Principal Menu**:
Turn OFF by pressing the button DOWN
Turn ON by pressing the button UP

SetUp menù:
From the **Principal Menu**, for enter in the **SetUp** menù press the button **LEFT**.
You must see something like below.

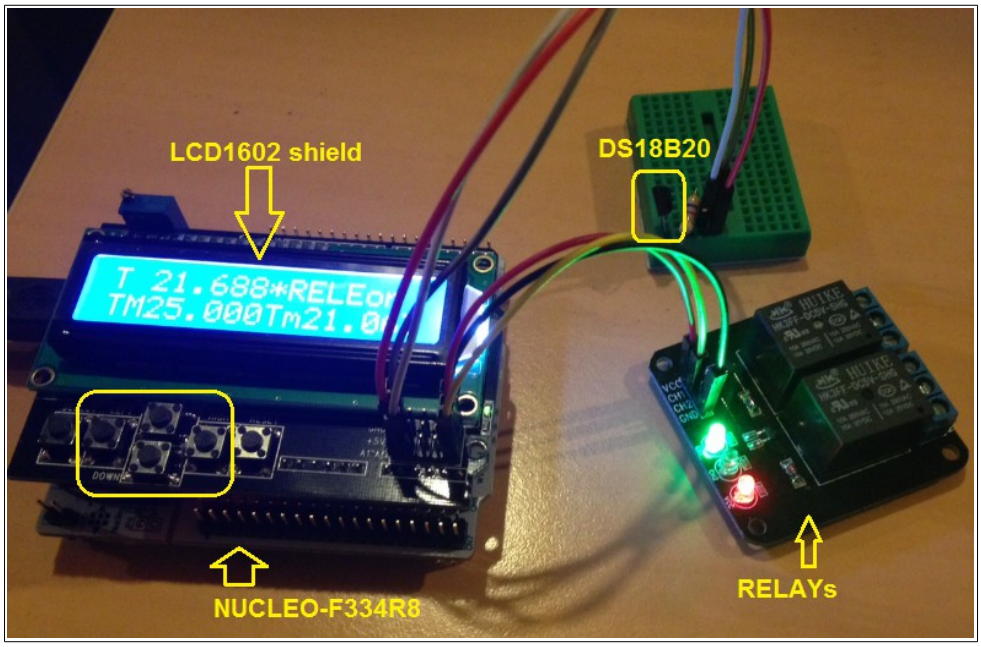
Tm20.00 - is the **tempON** – if the ambient temperature is equal or minor (<=), the relay is turned on.
TM20.50 - is the **tempOFF** – if the ambient temperature is equal or major (>=), the relay is turned off.

After the pressure of the LEFT button you are in the page for configure the **TempMin** (tempON).
In other words, now you have the possibility to configure the minimum temperature to control the turn on of the relay.
If the temperature measured is <= of the TempMin the relay is turned on.

UP – increase the TempMin of 0.1
DOWN – decrease the TempMin of 0.1
RIGHT – go to next menù

In the last menu, see below, you do the same for the **TempMax** (tempOFF).

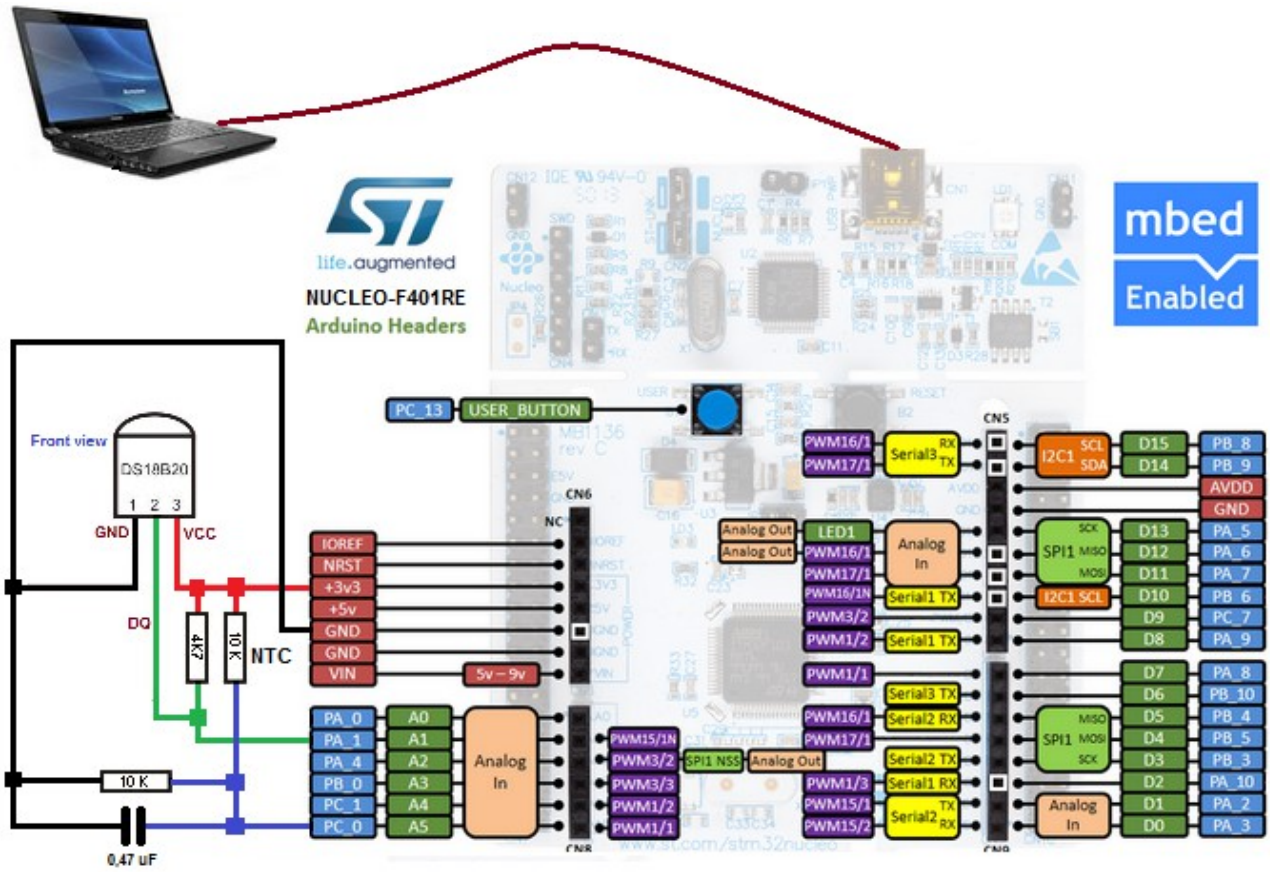
Try to navigate inside the various menus.



NUCLEO-F401RE + DS18B20 + Thermistor

The source code of this example is available **here**.

The connections are shown below

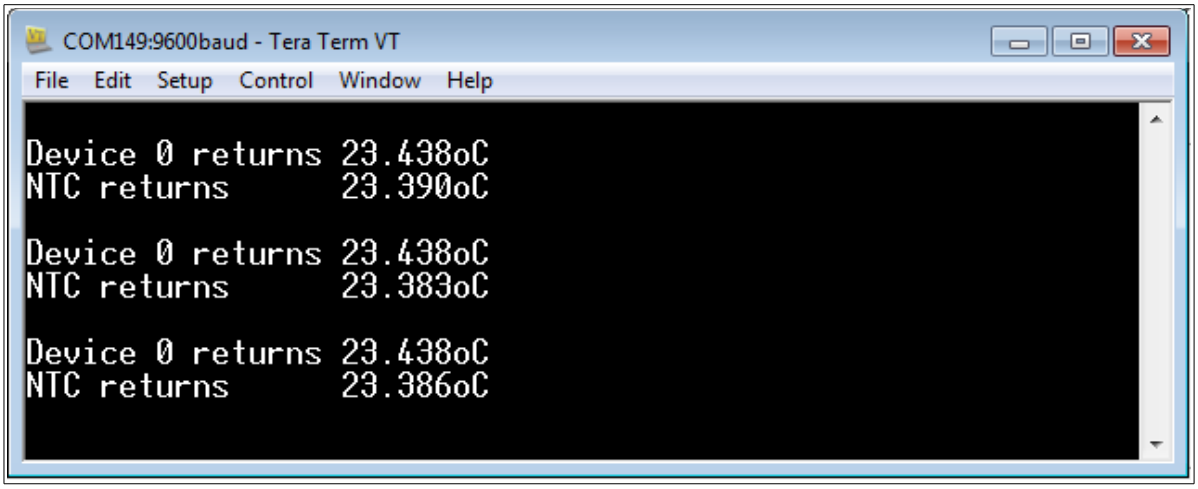


The USART parameters are:

Baud Rate: **9600**
Data: **8**
Parity: **NONE**
Stop: **1**
Flow Control: **NONE**

How to use this example

Connect the DS18B20 and the Thermistor to the NUCLEO-F401RE board.
Connect the NUCLEO-F334R8 to the PC.
On the PC, run the TeraTerm and see the result that must be similar to my result shown below.



Device 0 is DS18B20
NTC is the Thermistor

NOTE:
Thermistor (NTC) value is 10K.
For calculate the temperature we use the **Steinhart-Hart** equation, see **here**.

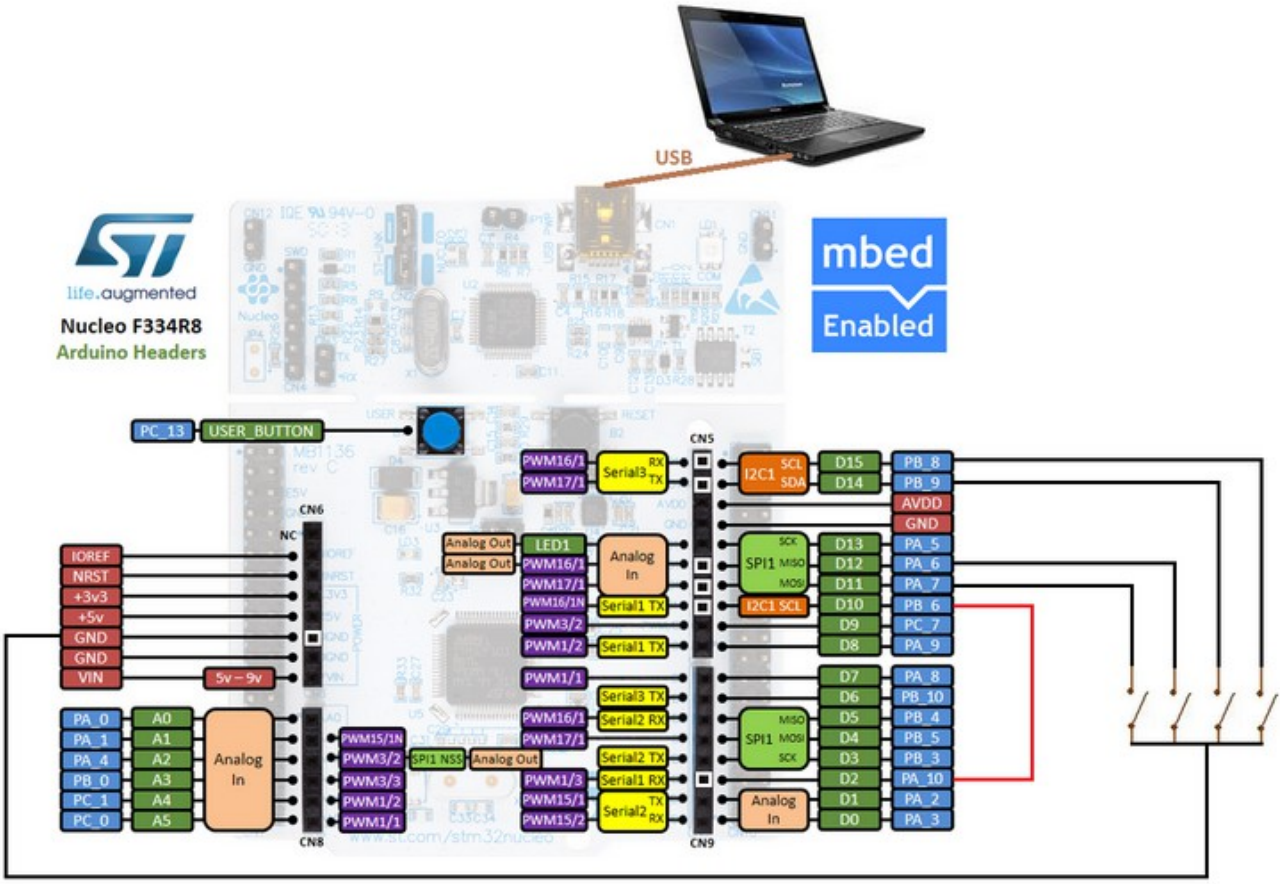
ATTENTION:
Thermistor response, is not fully tested.

How to use NUCLEO-F334R8 and..

Digital_IN
CRC calculation
Conversion from DECIMAL to BINARY
USART1 and USART2

The source code of this example is available **here**.

The connections are shown below



The USARTs parameters are:

Baud Rate: **1200**
Data: **8**
Parity: **NONE**
Stop: **1**
Flow Control: **NONE**

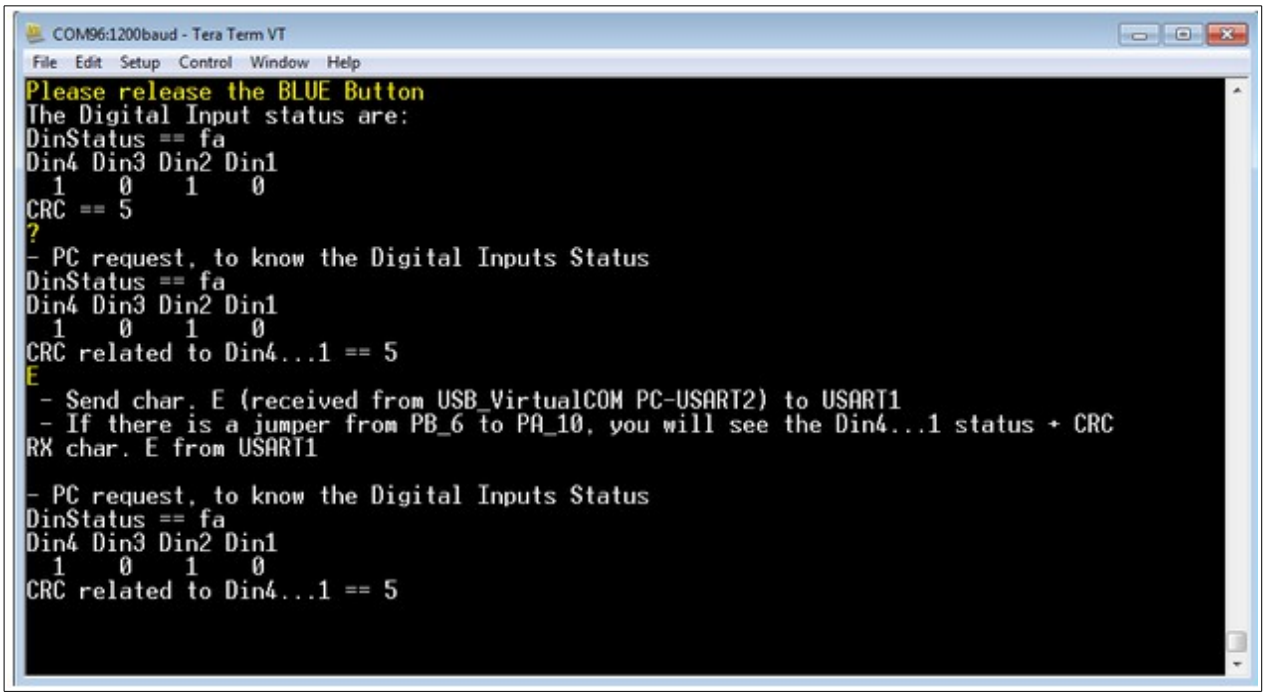
How to use this example

If you press the **Blue button** (it is on NUCLEO-F334R8 board) you must see the phrase:
Please release the Blue Button
After you release the button, you must see the status of Digital Inputs and the CRC value.

If you press the: **?** on your PC
you must see the status of Digital Inputs and the CRC value.

If you press the: **E** on your PC
you must see the status of Digital Inputs and the CRC value.

The results are shown below (we use TeraTerm).

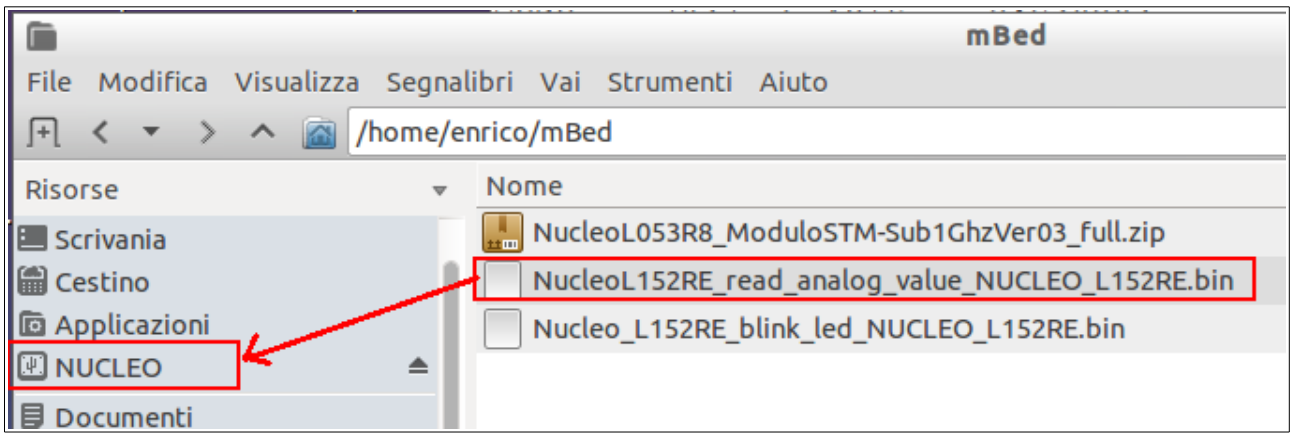


LINUX and mBed + Nucleo Boards

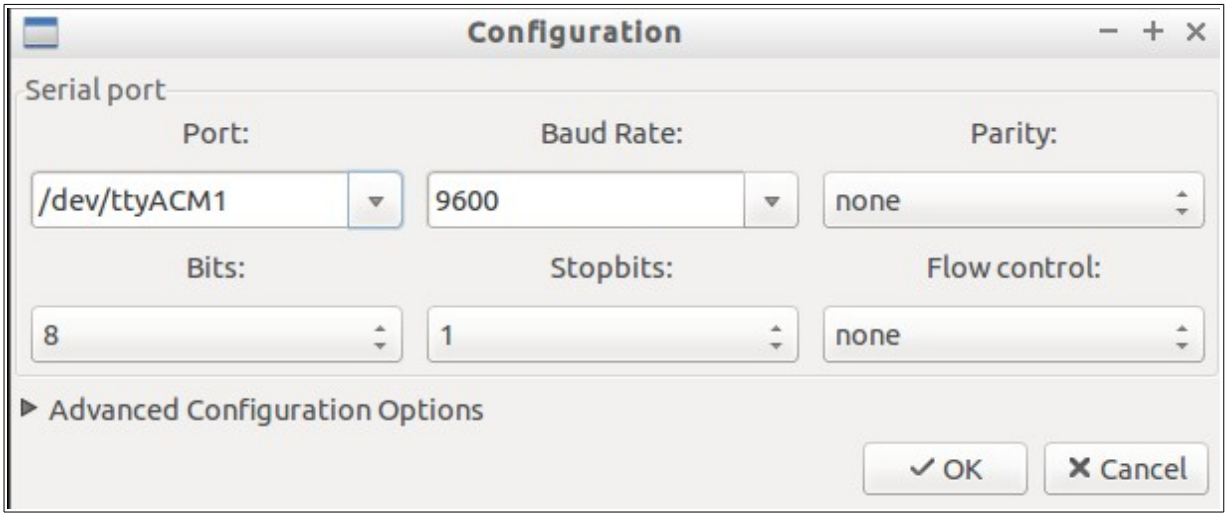
I'm using **LUBUNTU** that is a derivation of UBUNTU.
Lubuntu use the **LXDE** (Lightweight X11 Desktop Environment) that is a free desktop environment with comparatively low resource requirements.
This makes it especially suitable for resource-constrained personal computers such as old netbooks or system on a chip computers.

I'm using **Firefox Browser Web** for access to **mBed** without problems.

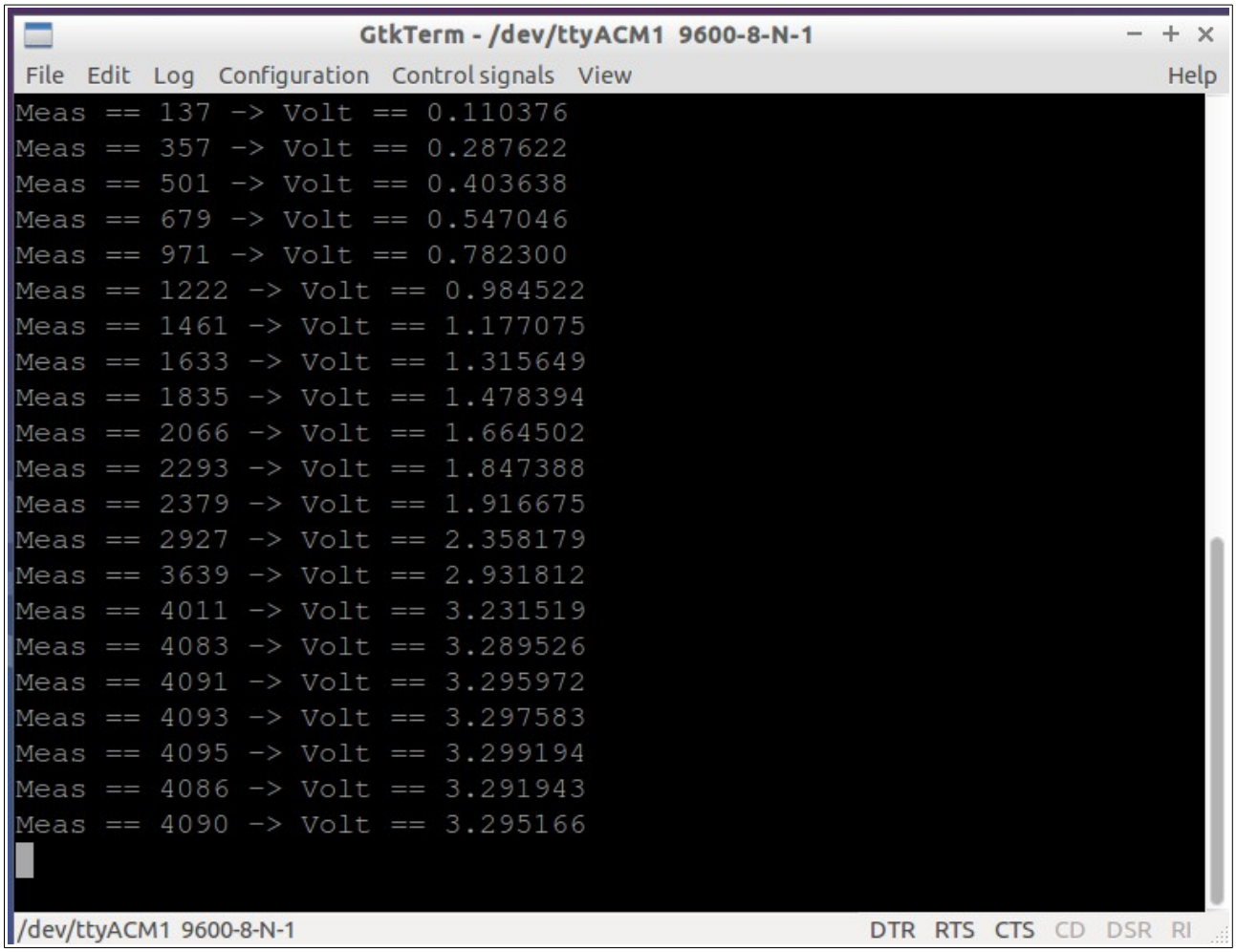
For **programming** the Nucleo board it is only necessary to **drag and drop** the **.bin file** on the **NUCLEO board icon**, see below.



For using **Virtual Com** for debugging (**printf**) there are a loot of free programs available under Linux i.e., PuTTY, moserial Trminal, GtkTerm, etc.
My prefered Terminal Emulation is **GtkTerminal**.
Below the setup page of GtkTerminal.



Below there is the output of the GtkTerminal used for monitoring the AnalogIn (ADC) example explained **here**.



LINK

- **mBed API**
- **Fast and Effective Embedded Systems Design: Applying the ARM mbed** is **here**.
- **Regarding interrupts, their use and blocking**
- mBed home page
- General sw
- Library, provides the C/C++ software platform and libraries to build your applications
- Mbed compiler
- **C++ Basics**
- **Extra tutorials concerning mBed and Nucleo Boards**