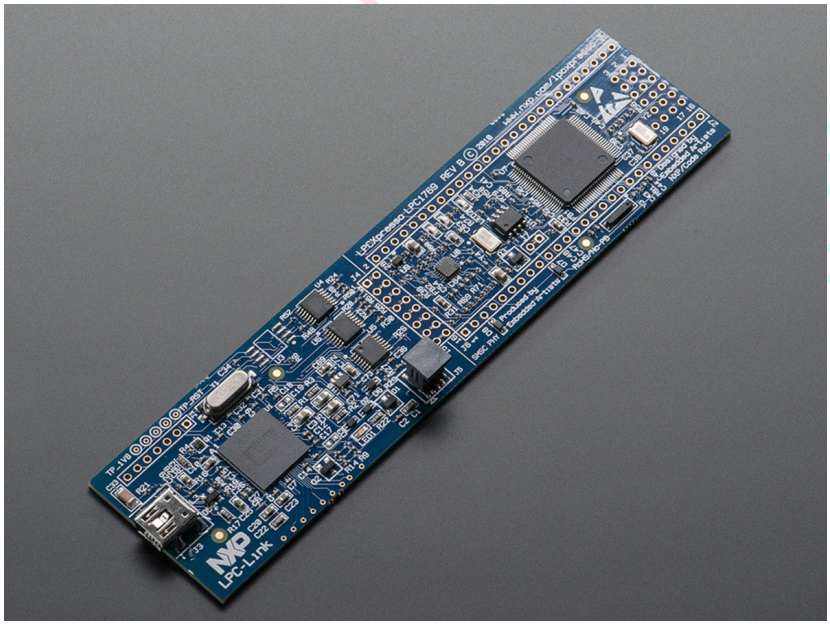
TENET TECHNETRONICS

2016

Interfacing RGB LED with NXP LPC1769 using LPCXpresso



Author: Gurudatta Palankar

Reviewers:

Version: 1.0

9/3, 2nd floor, SreeLaksmi Complex, opp, to Vivekananda Park, Giri

Email: info@tenettech.com, Phone: 080 - 26722726

Introduction:

LPCXpresso™ is a new, low-cost development platform available from NXP supporting NXP's ARM-based microcontrollers. The platform is comprised of a simplified Eclipse-based IDE and low-cost target boards which include an attached JTAG debugger. LPCXpresso™ is an end-to-end solution enabling engineers to develop their applications from initial evaluation to final production.

Step 1: Open LPCXpresso IDE

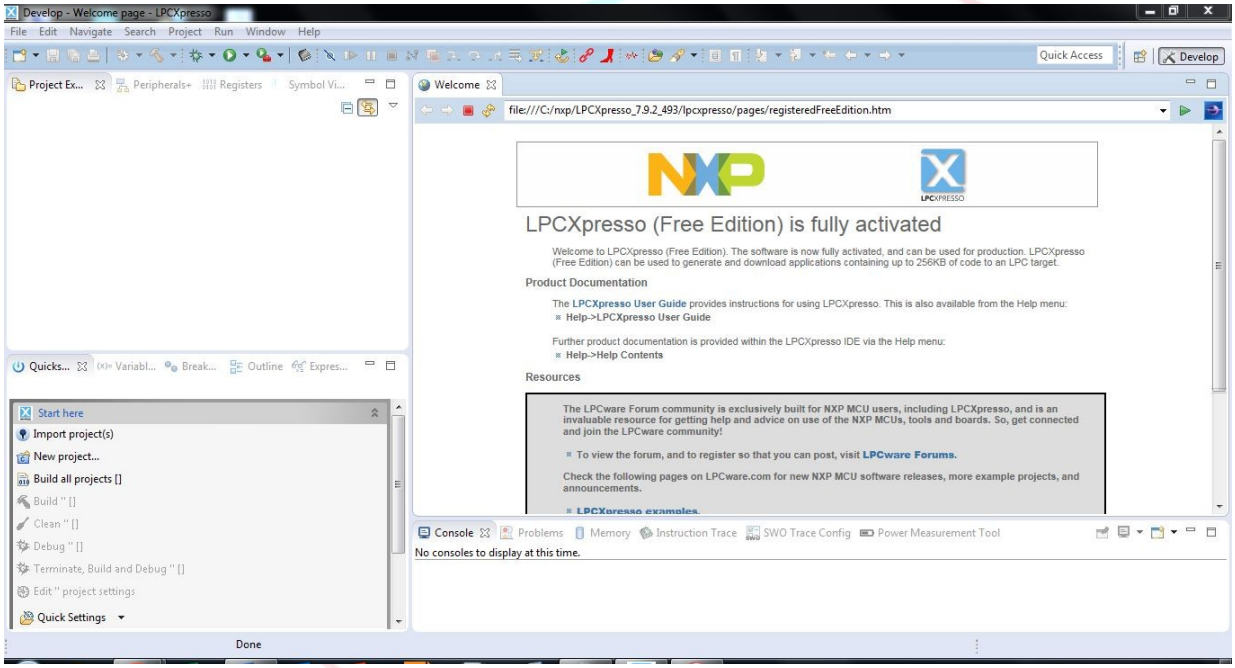


Figure 1

Step 2: Before writing a code, we have to Import some Library Files to the Workspace. Click on Import projects on Quickstart Panel on the bottom left of the window.

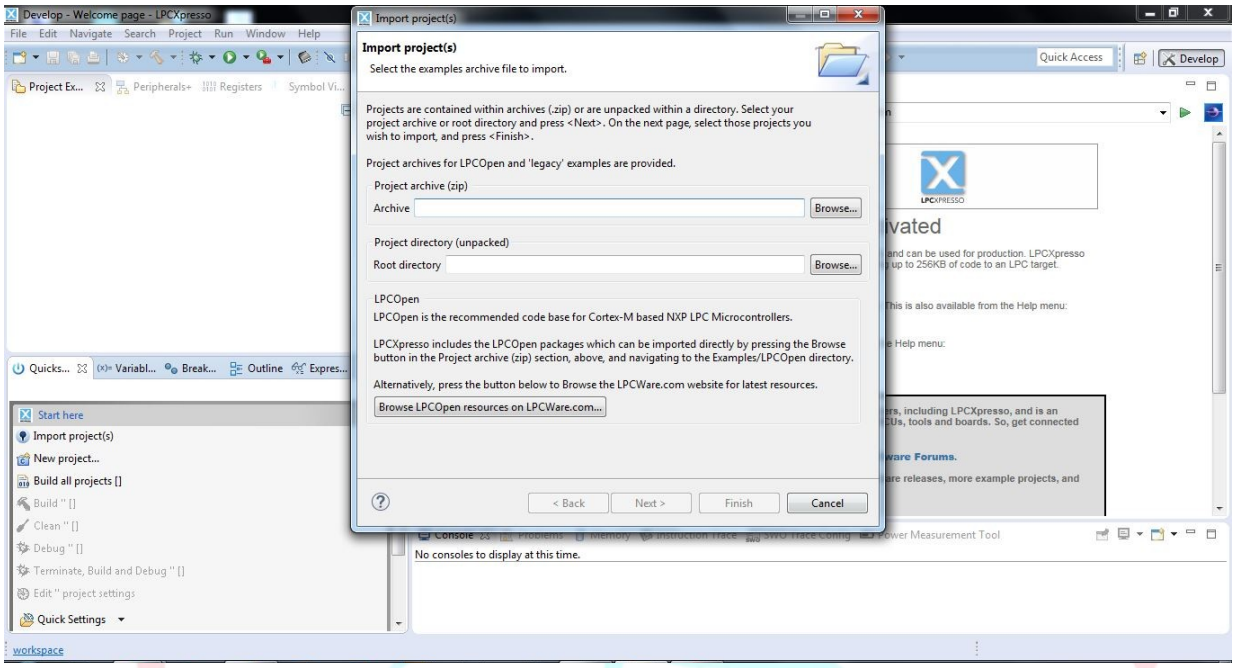


Figure 2

Step 3: Browse file, open the LPC1000 folder.

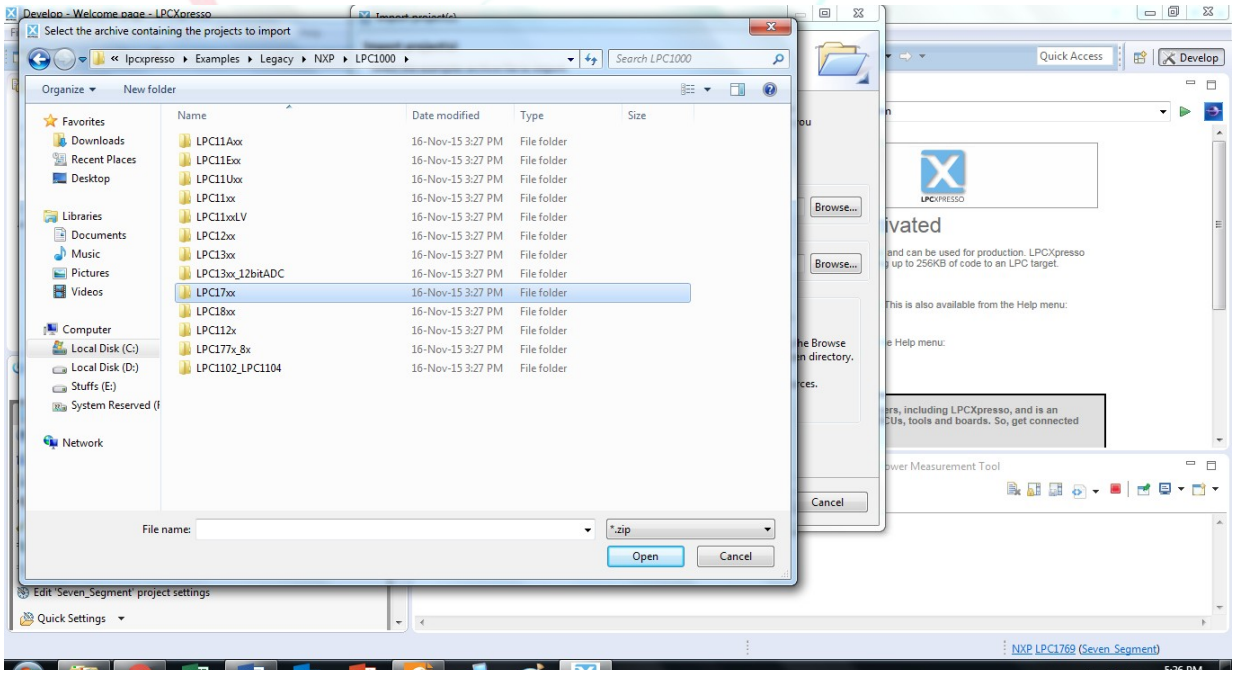


Figure 3

Step 4: Select the appropriate archive file. Let us select LPCXpresso176x_cmsis2. We can select CMSIS CORE library that include LPC17xx.h header file.

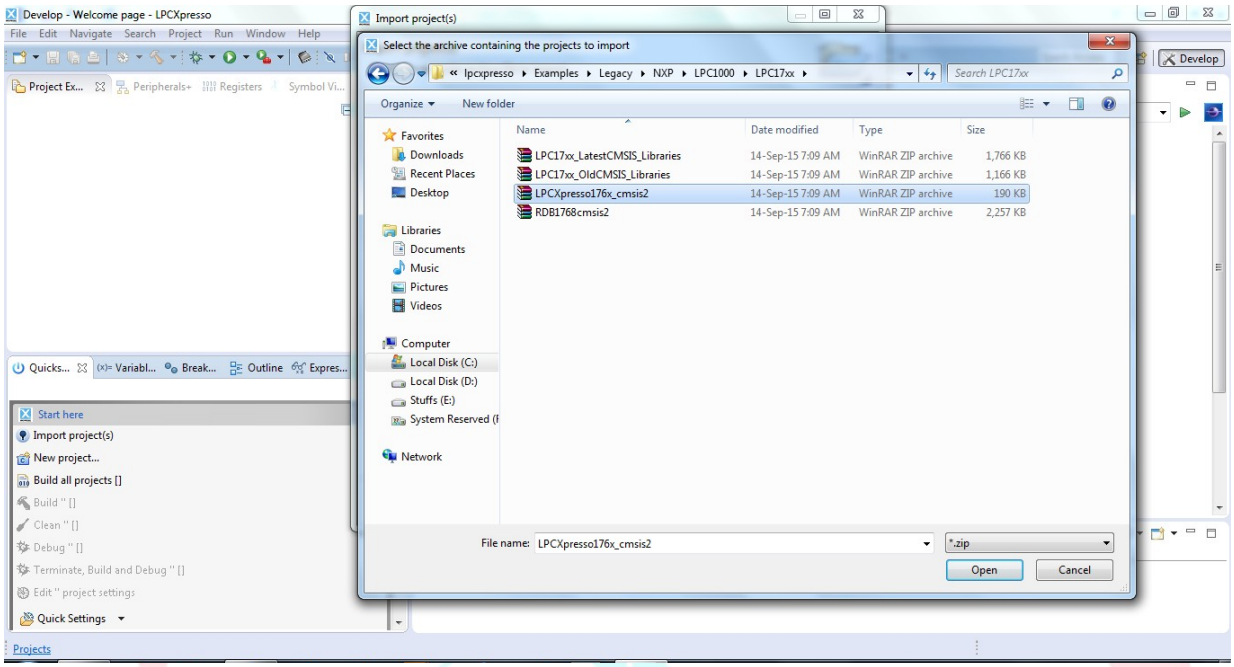


Figure 4

Step 5: After selecting you will be able to see the following libraries files. Let us select specific one.

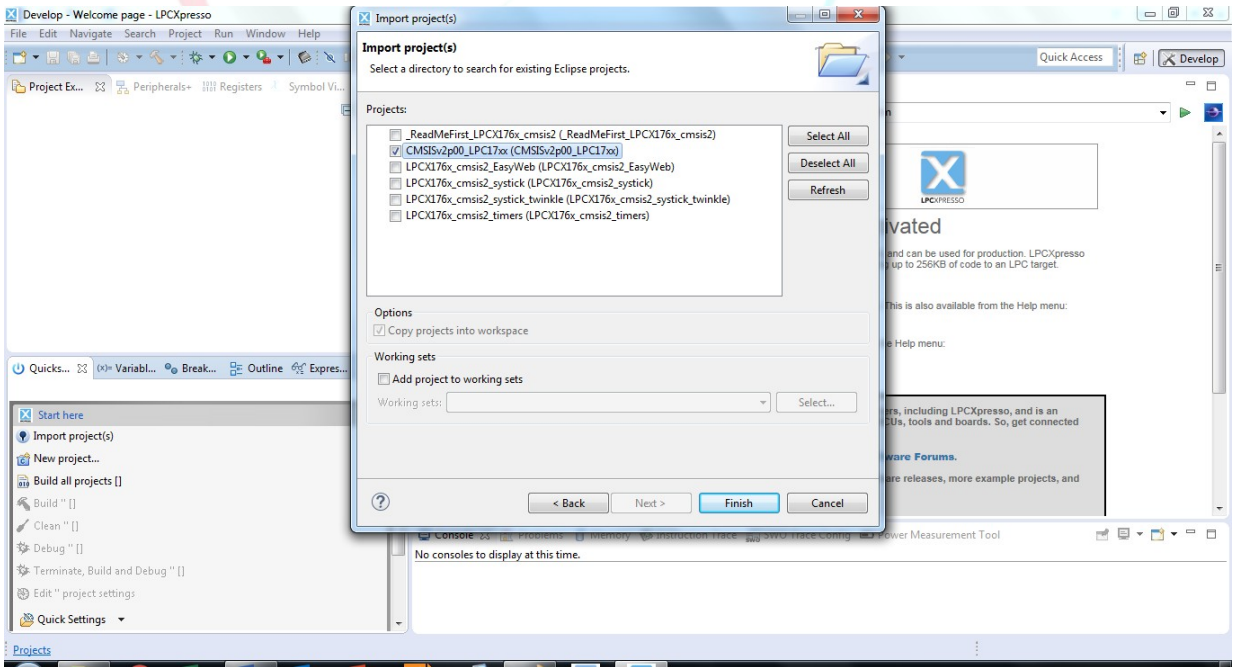


Figure 5

Step 6: Now we will be able to see those libraries in the workspace.

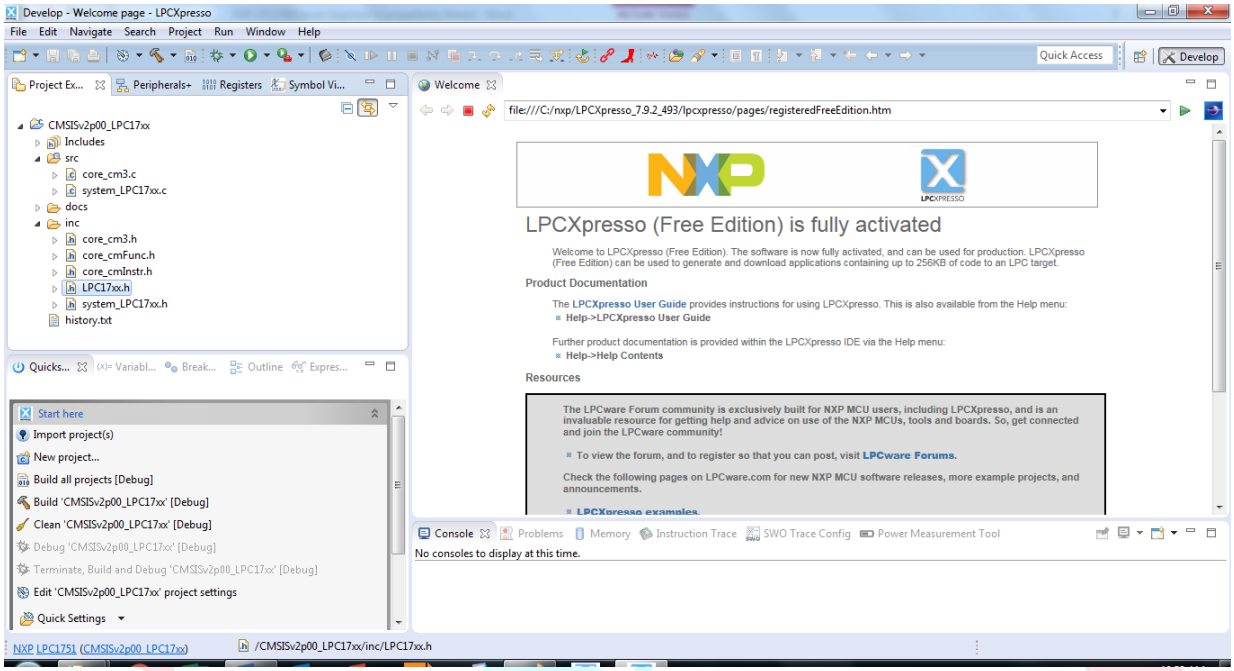


Figure 6

Step 7: Now we can start creating our new project. Goto File >> New >> Project. Select LPCXpresso C project.

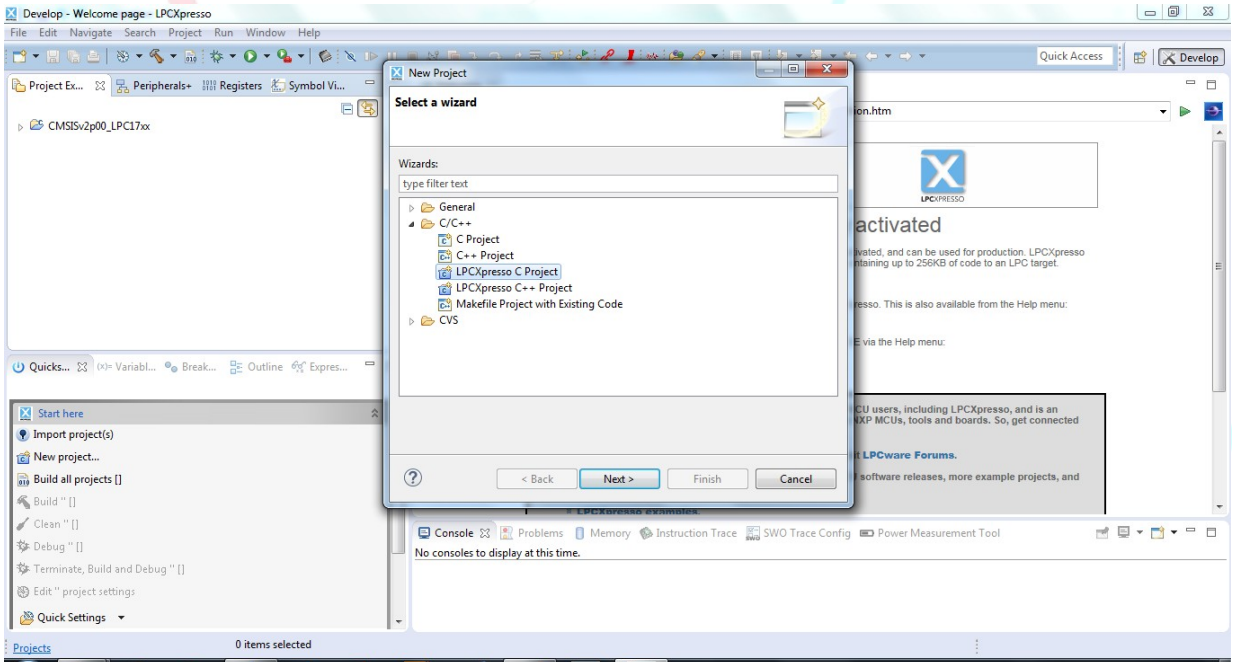


Figure 7

Step 8: Select LPC1769, C Project and give name to your project. Select target MCU as LPC1769.

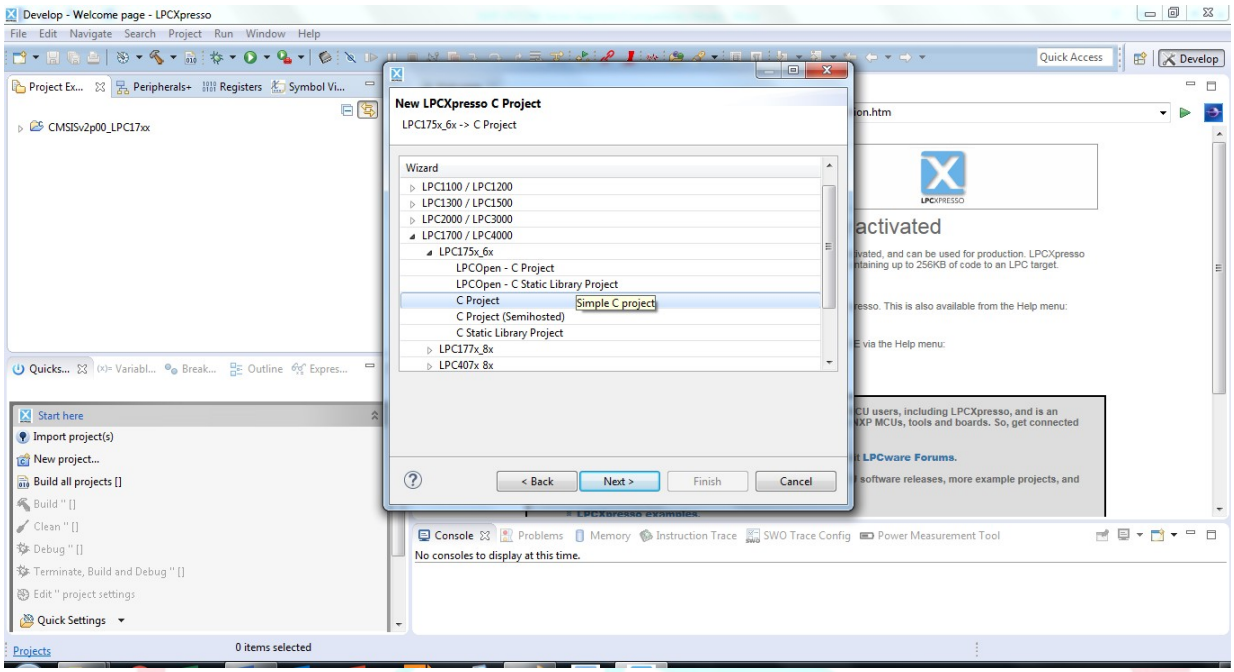


Figure 5

Step 9: Now select CMSIS Core library. Click on Next and keep all the other configurations as default and Finish.

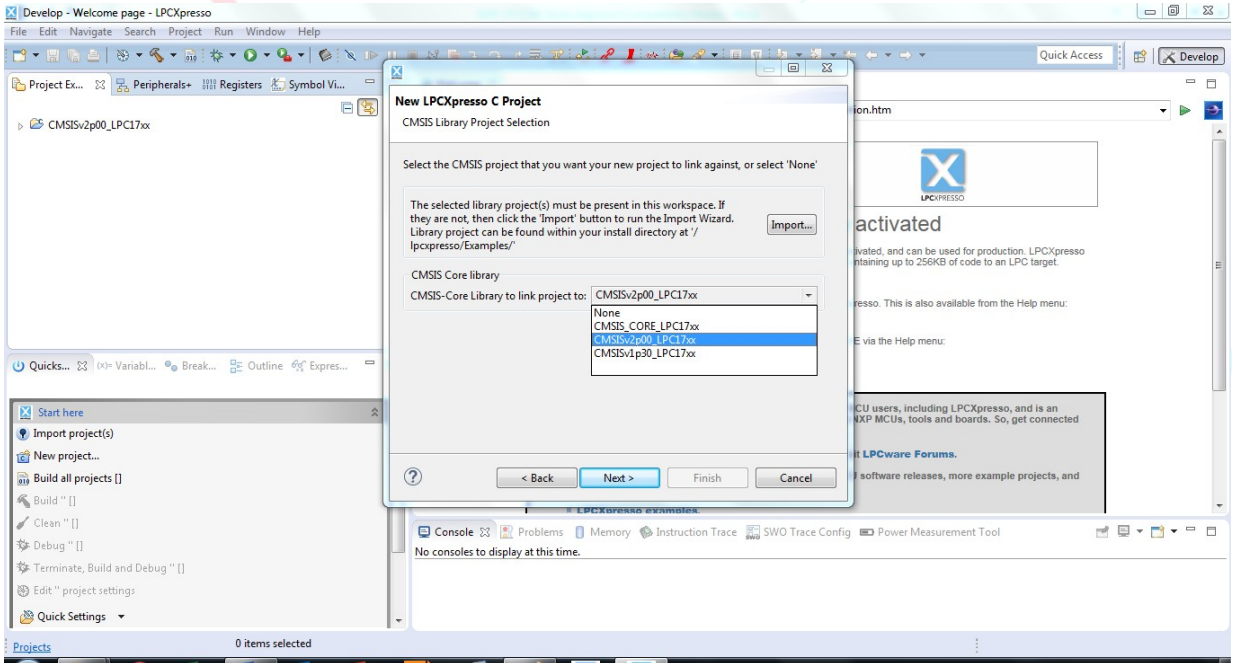


Figure 9

Step 10: Now we can see our project onto the workspace. Now by double clicking on RGB_LED.c file, we can start writing code.

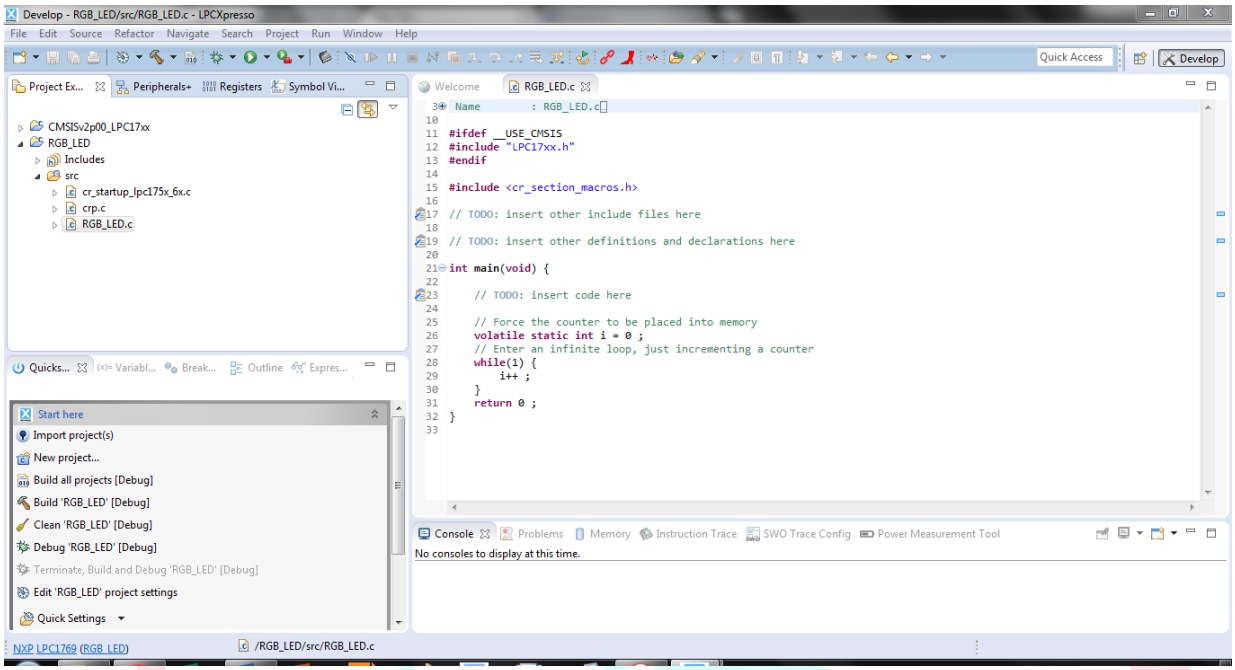


Figure 10

Step 11: Write a code as shown below.

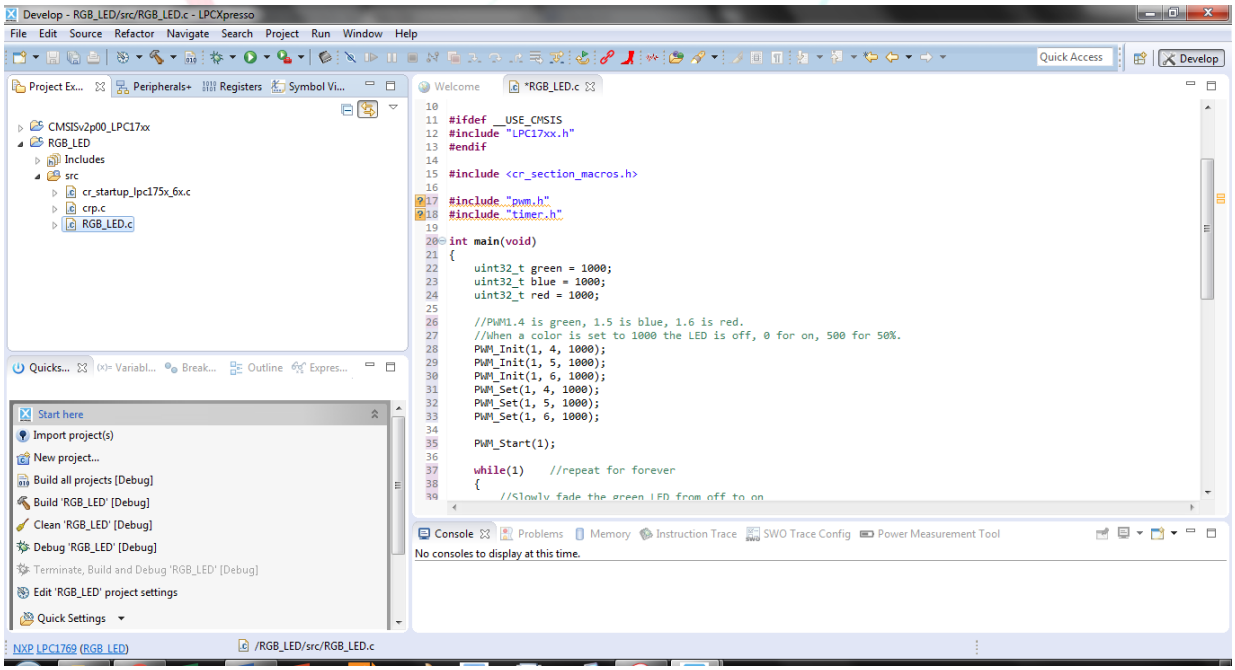


Figure 11

CODE:

```
#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

#include <cr_section_macros.h>

#include "pwm.h"
#include "timer.h"

int main(void)
{
    uint32_t green = 1000;
    uint32_t blue = 1000;
    uint32_t red = 1000;

    //PWM1.4 is green, PWM1.5 is blue, and PWM1.6 is red.
    //When a color is set to 1000 the LED is off, 0 for on, 500 for 50%.

    PWM_Init(1, 4, 1000);
    PWM_Init(1, 5, 1000);
    PWM_Init(1, 6, 1000);

    PWM_Set(1, 4, 1000);
    PWM_Set(1, 5, 1000);
    PWM_Set(1, 6, 1000);

    PWM_Start(1);

    while(1)
    {
        //Slowly fade the green LED from off to on
        for (green = 1000; green > 1; green--)
        {
            PWM_Set(1, 4, green);
            delayMs(0, 1);
        }

        //Slowly fade the blue LED from off to on
        for (blue = 1000; blue > 1; blue--)
        {
            PWM_Set(1, 5, blue);
            delayMs(0, 1);
        }

        //Slowly fade the red LED from off to on
        for (red = 1000; red > 1; red--)
        {
            PWM_Set(1, 6, red);
            delayMs(0, 1);
```

```
        }

        //Slowly fade the green LED from on to off
        for (green = 0; green < 1000; green++)
        {
            PWM_Set(1, 4, green);
            delayMs(0, 1);
        }

        //Slowly fade the blue LED from on to off
        for (blue = 0; blue < 1000; blue++)
        {
            PWM_Set(1, 5, blue);
            delayMs(0, 1);
        }

        //Slowly fade the red LED from on to off
        for (red = 0; red < 1000; red++)
        {
            PWM_Set(1, 6, red);
            delayMs(0, 1);
        }
    }
}
```

NOTE: The above code will not work until and unless we add pwm.h and timer.h library files.

Step 12: To create or add library files, right click on src file of your project file, then New >> Source File.

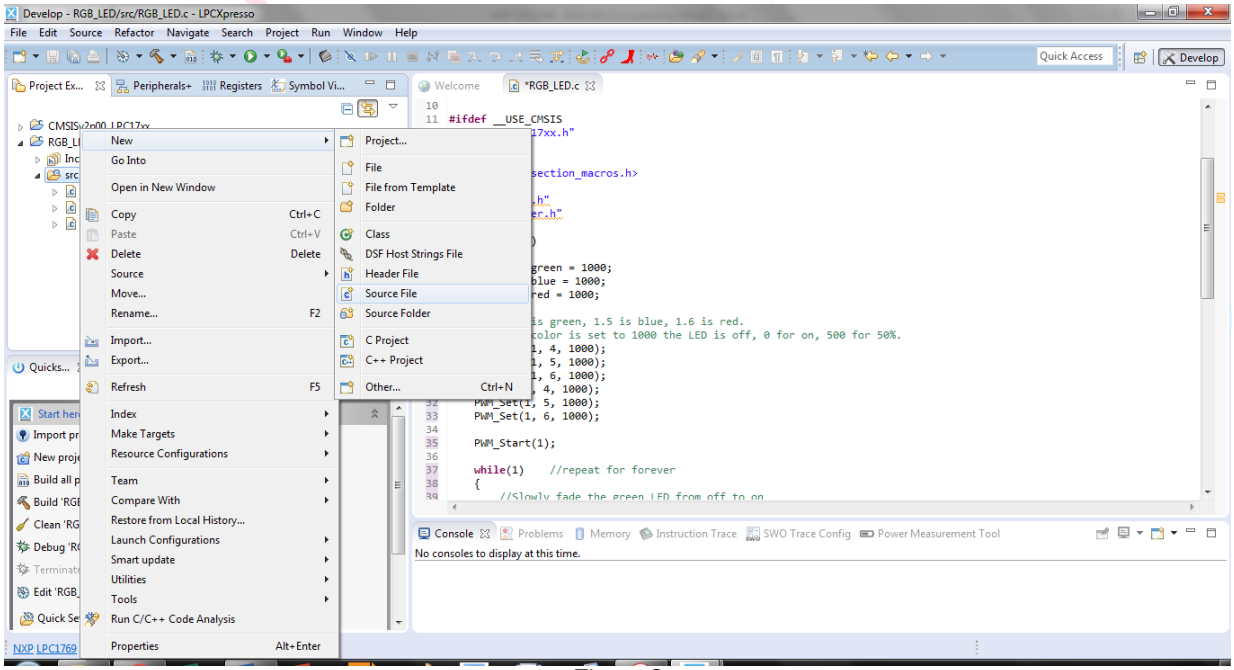


Figure 6

Step 13: Save the Source File name with .c file extension.

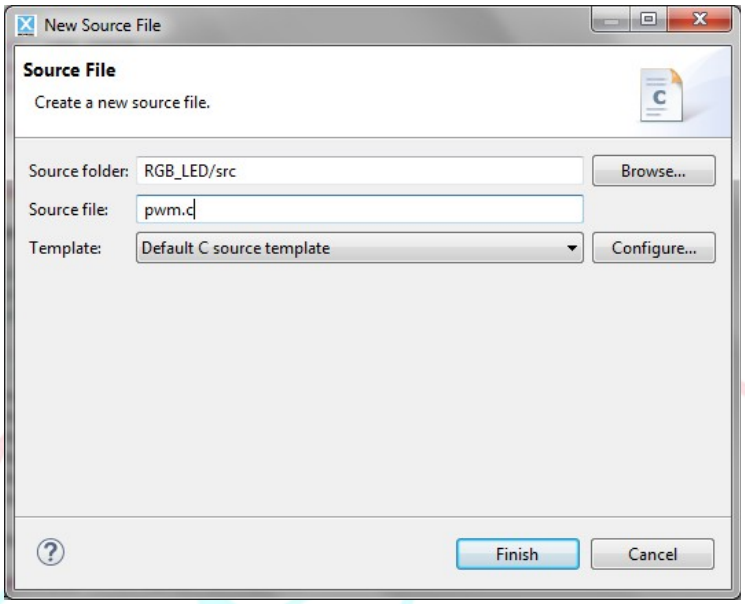


Figure 13

Step 14: Write or paste the Source File (Library).

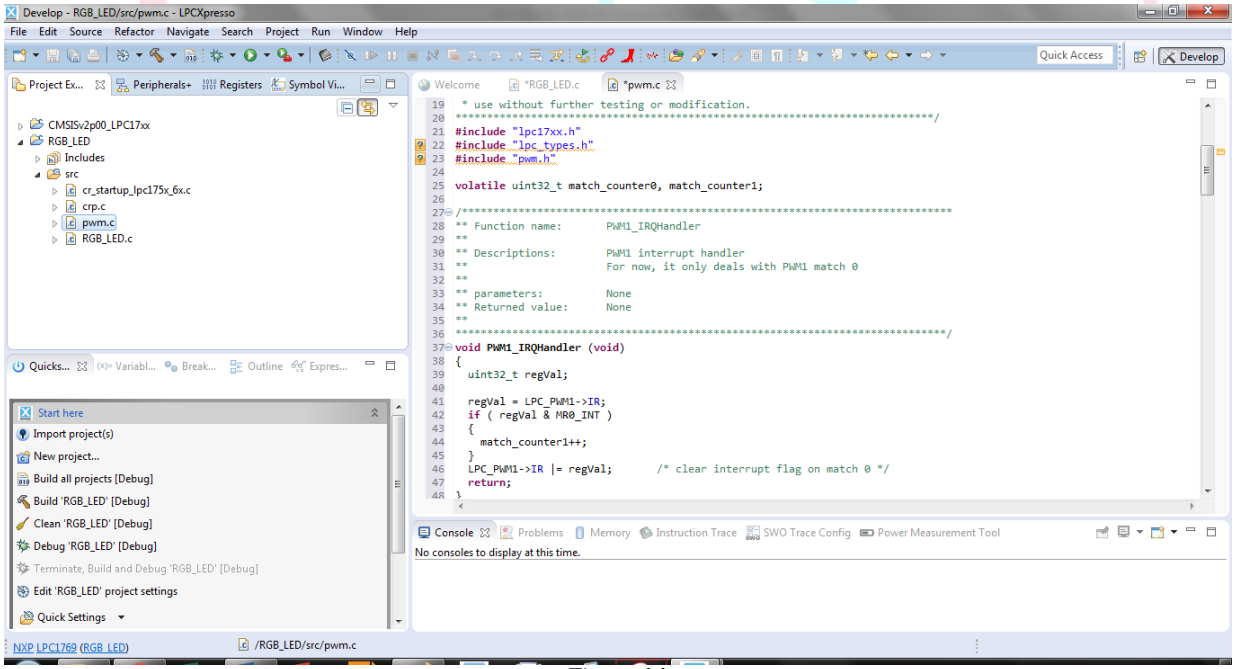


Figure 14

Step 15: Similarly add the Header File.

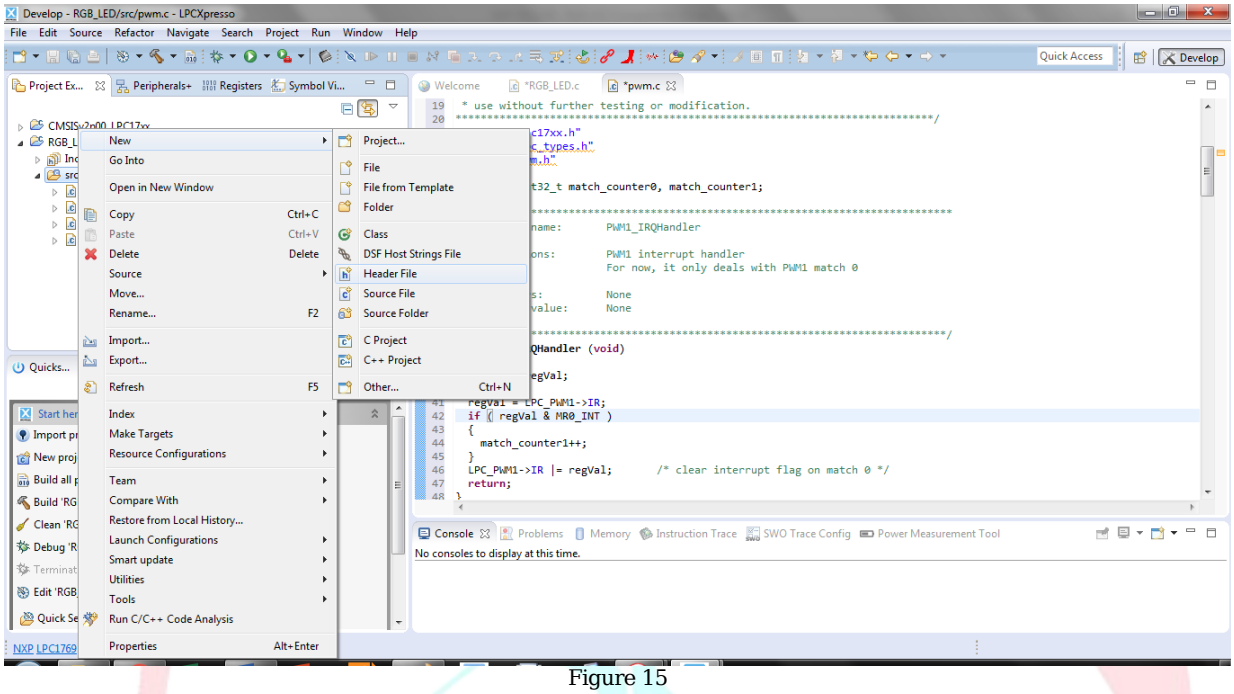


Figure 15

Step 16: Save the Header File name with .h file extension.

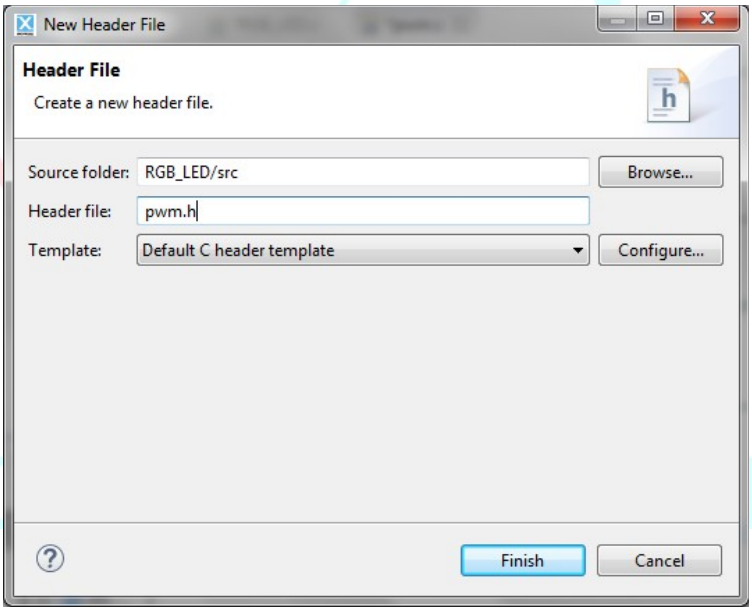


Figure 16

Step 17: Write or paste the Header File (Library).

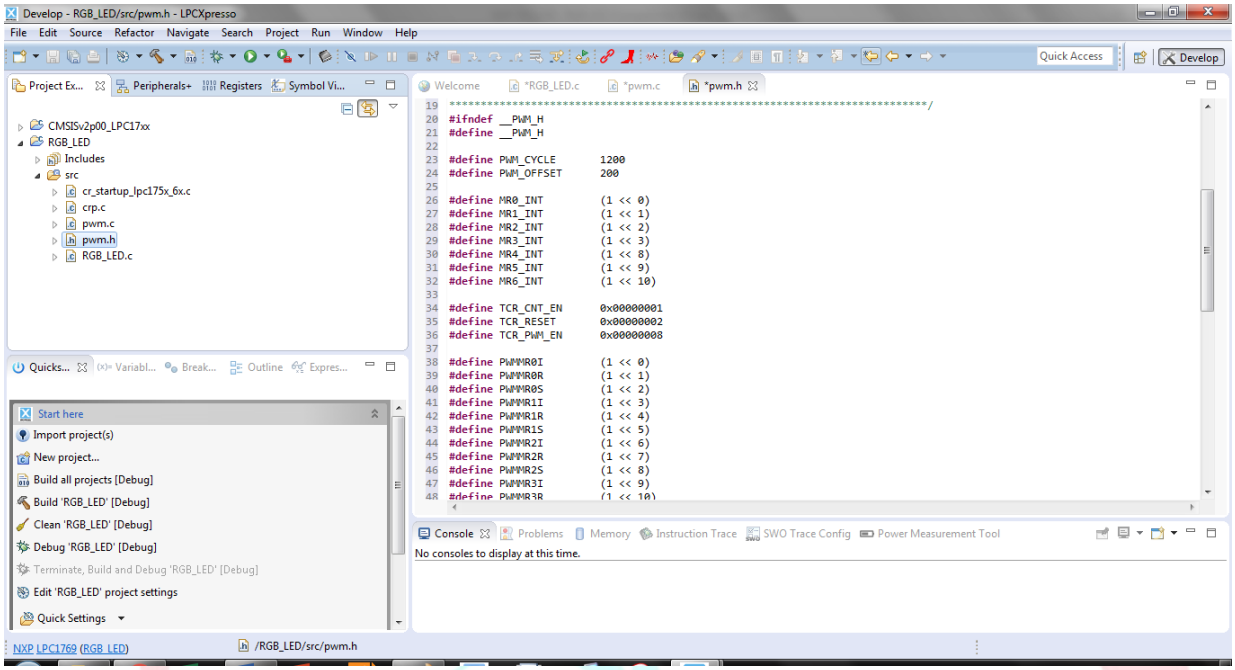


Figure 17

NOTE: Similarly, add libraries for timers (timer.c and timer.h). Here we have used Timers to generate delays.

Step 18: After writing code and adding libraries, Build the project by clicking on Build RGB_LED on the Quickstart Panel on the bottom left of the window.

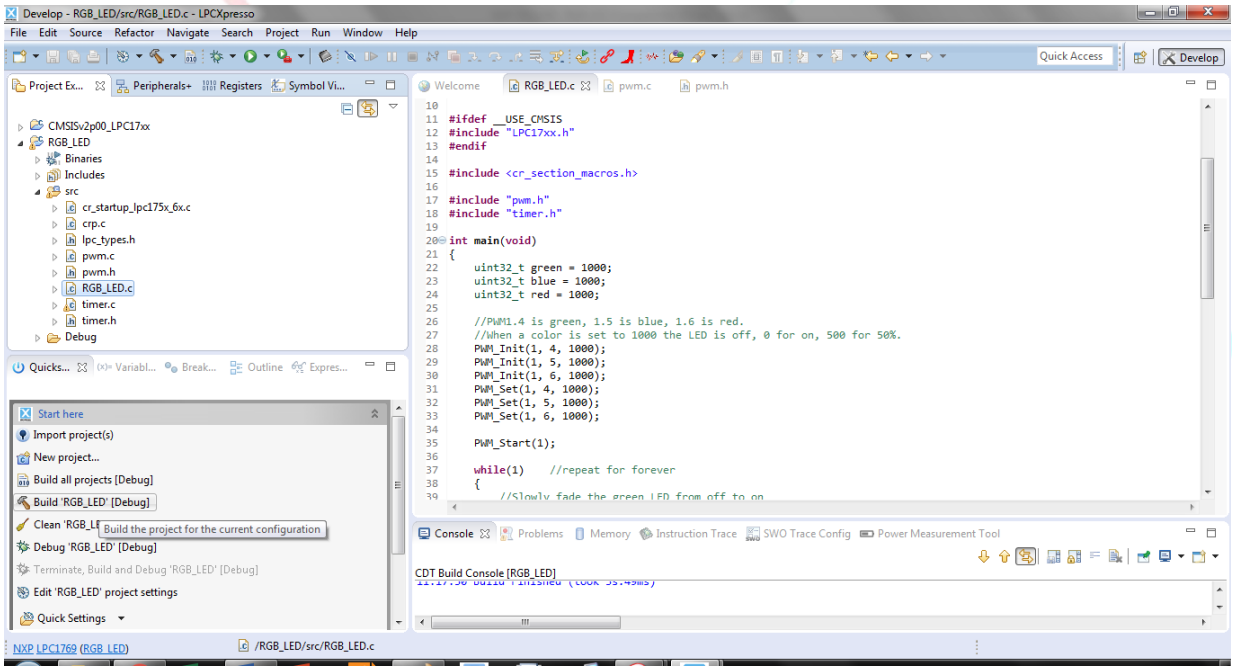


Figure 18

Step 19: Now, if all goes well connect the Micro B cable to LPC1769 and connect it to your computer. To upload the project file, click on the Program flash.

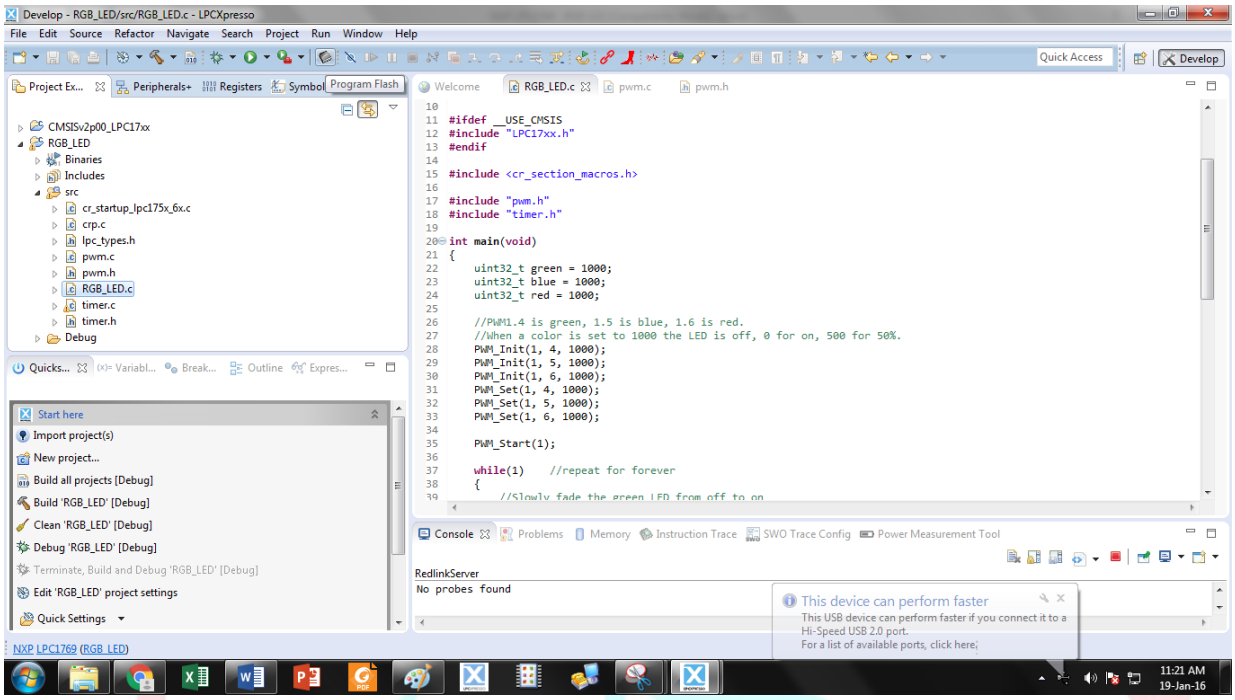


Figure 19

Step 20: Now select the Project file RGB_LED.axf. We can find it in our project folder.

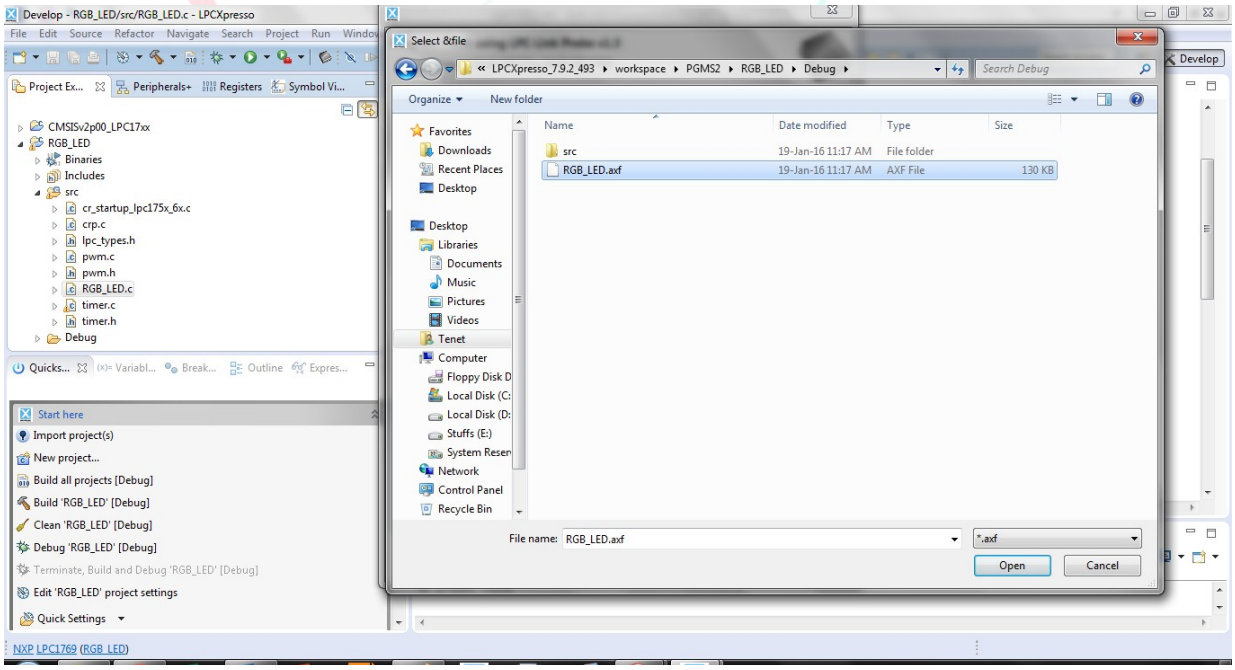


Figure 20

Step 21: Now this window shows we have finally dumped our project onto LPC1769.

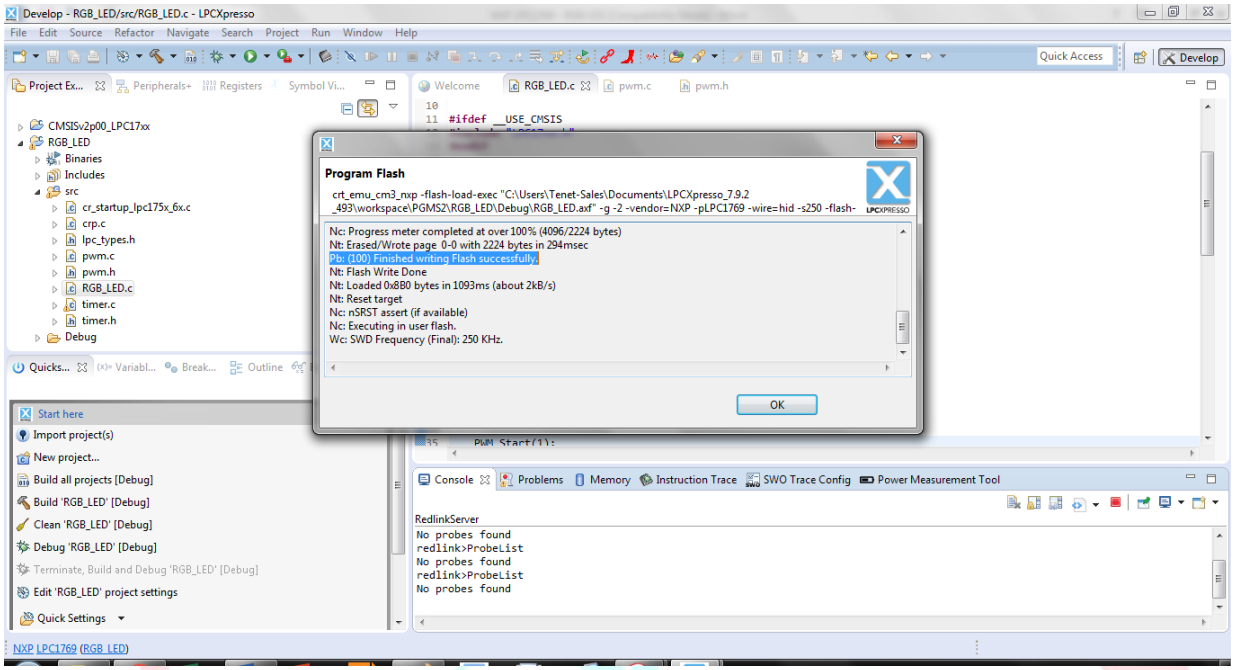


Figure 21

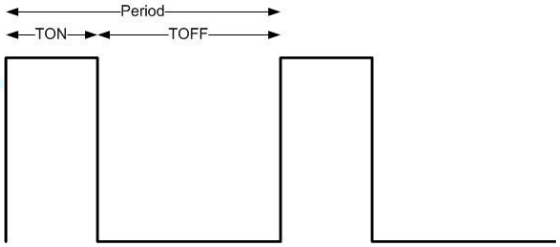
EXPLANATION:

Hardware:

- **LPC1769 Board**
- **RGB LED**
- **breadboard**
- **hook-up wire**

Basic Principle of PWM:

Pulse width modulation is basically a square wave with a varying high and low time. A basic PWM signal is shown in the figure below.



Period = TON + TOFF

Frequency = 1 / Period

Duty Cycle = $\frac{TON}{TON + TOFF}$ *** 100**

There are various terms associated with PWM:

- **On-Time: Duration of time signal is high**
- **Off-Time: Duration of time signal is low**
- **Period: It is represented as the sum of on-time and off-time of PWM signal**
- **Duty cycle: It is represented as percentage of time signal remains on during the period of the PWM signal**

Period:

As shown in the figure, T_{on} denotes the on-time and T_{off} denotes the off time of signal. Period is the sum of both on and off times and is calculated as shown in the equation below:

$$Period = T_{on} + T_{off}$$

Duty Cycle:

Duty cycle is calculated as on-time to the period of time. Using the period calculated above, duty cycle is calculated as:

$$Duty\ Cycle = T_{on} / (T_{on} + T_{off})$$
$$= T_{on} / Period$$

PWM: Voltage Regulation

PWM signal when used at a different duty cycles gives a varying voltage at the output.
This method is used in various areas of application like:

- **Switching regulators**
- **LED dimmers**
- **Audio**
- **Analog signal generation**
- **and many more...**

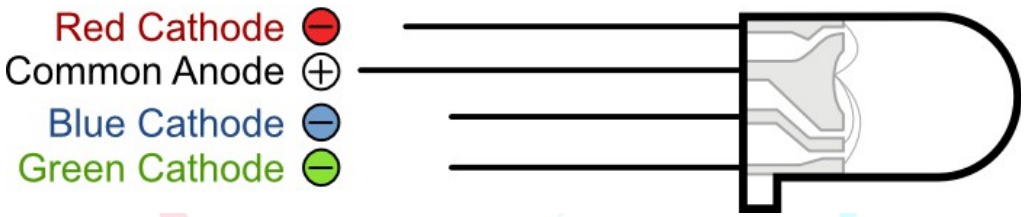
Voltage regulation is done by averaging the PWM signal. Output voltage is represented by the following equation:

$$V_{out} = Duty\ Cycle * V_{in}$$

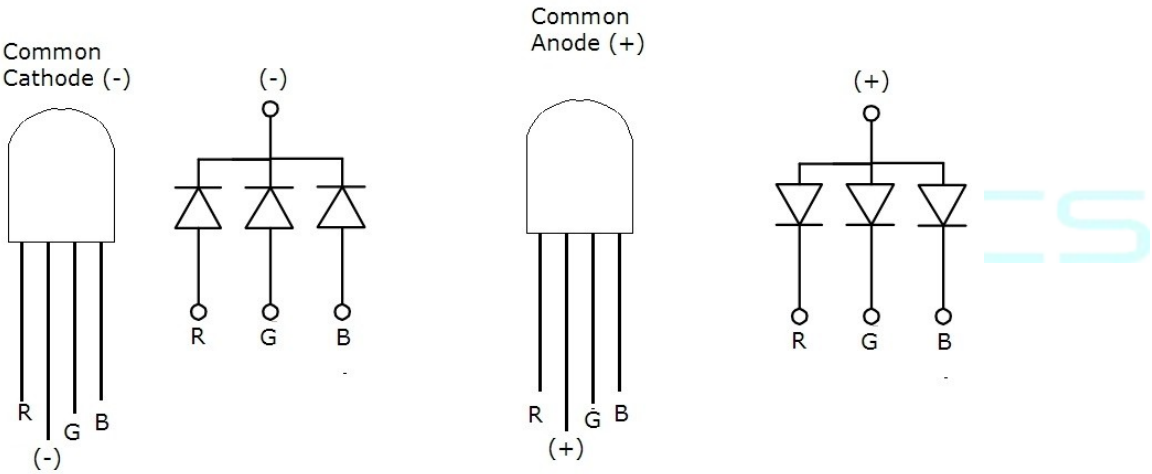
As you can see from the equation the output voltage can be directly varied by varying the T_{on} value. If T_{on} is 0, V_{out} is also 0. if T_{on} is T_{total} then V_{out} is V_{in} or say maximum.

RGB LED:

RGB LED PINOUT:



RGB LED, Common Anode and Common Cathode



OUTPUT:

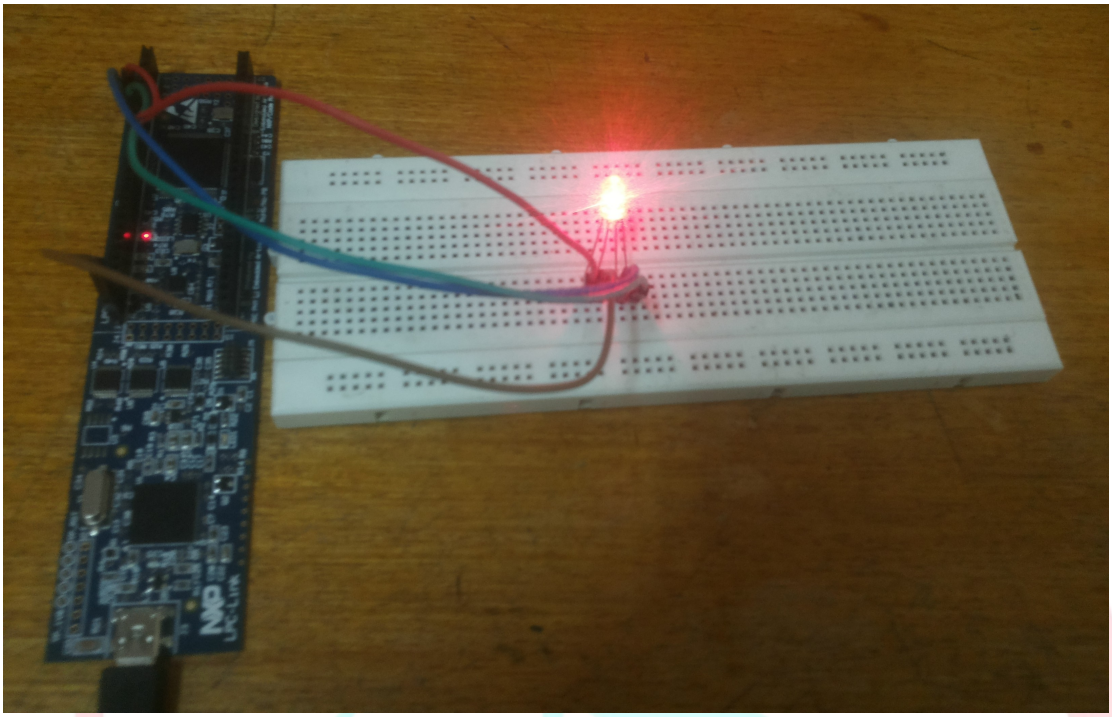


Figure 22

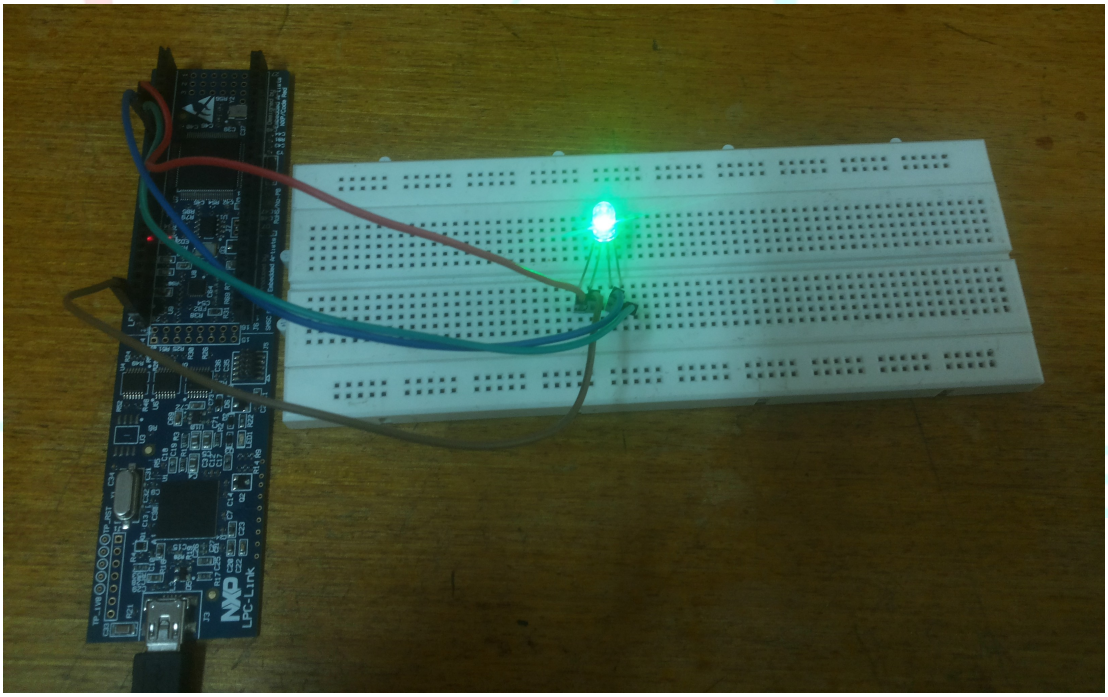


Figure 23

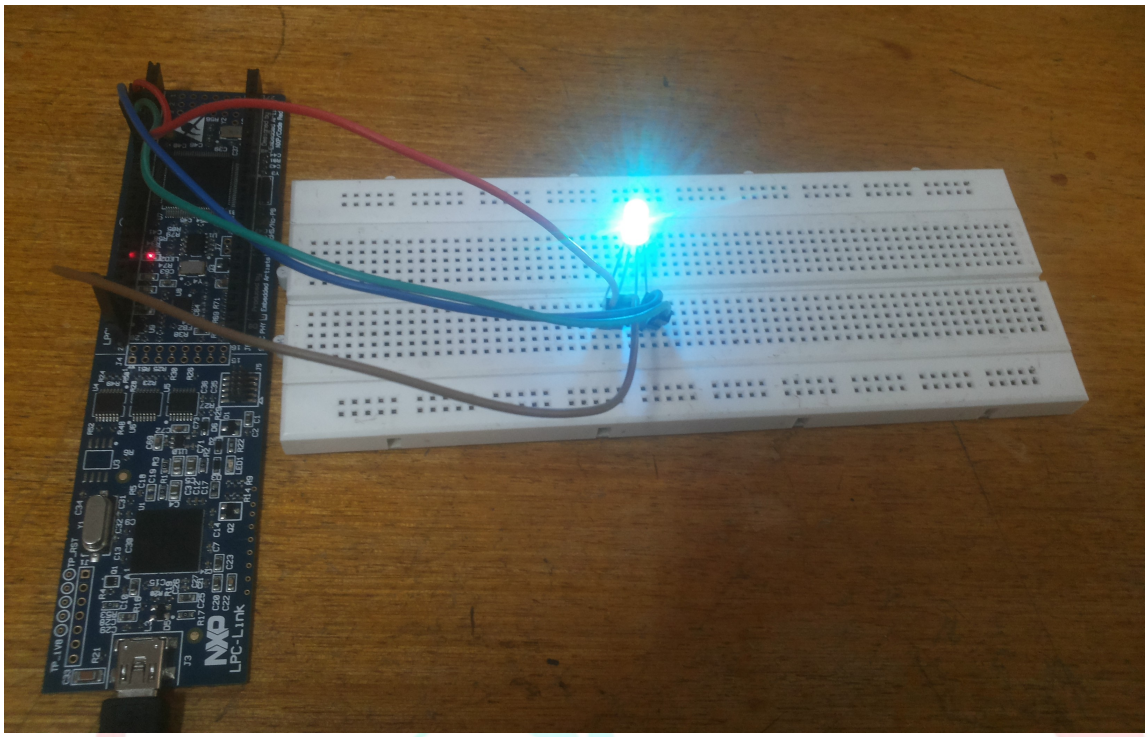


Figure 24

For product link:

1. **http://www.tenettech.com/product/1548/lpc1769-lpcxpresso-board**
2. **http://www.tenettech.com/product/6761/8mm-rgb-led-lamp-common-anode**

For more information please visit: www.tenettech.com

For technical query please send an e-mail: info@tenettech.com