Lab 3: DMA & PWR

Activity 1: DMA Configuration

Aim:

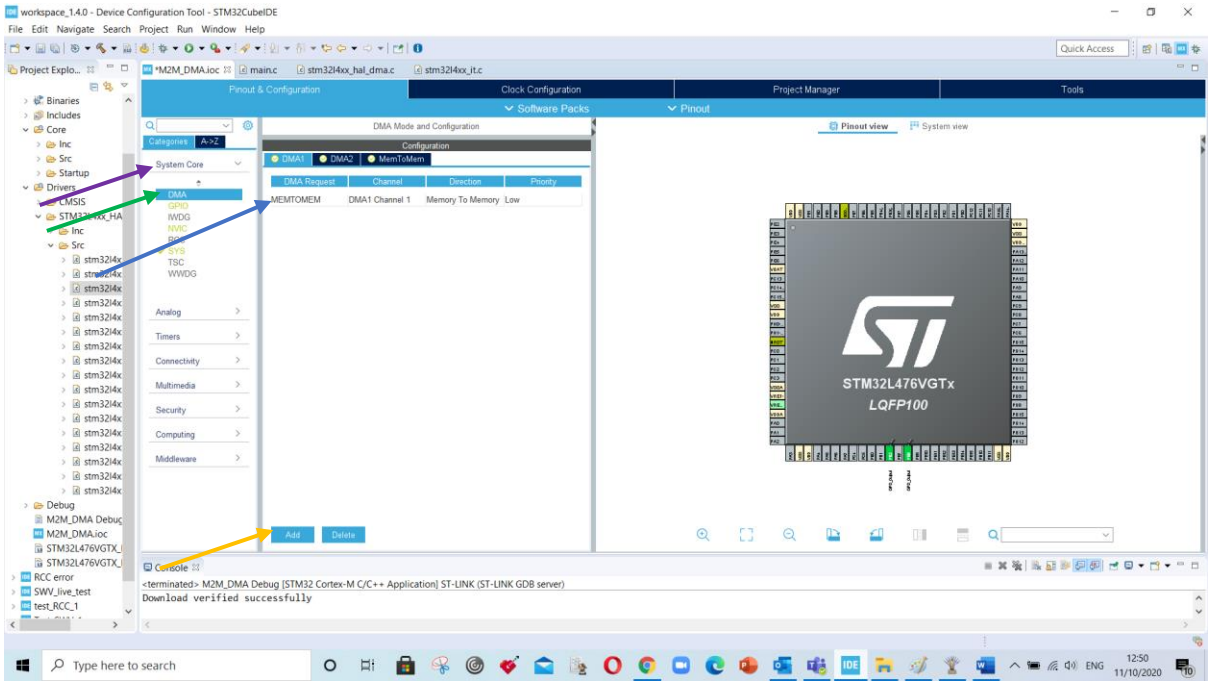
To configure a DMA channel for a memory-to-memory transfer.

Objectives:

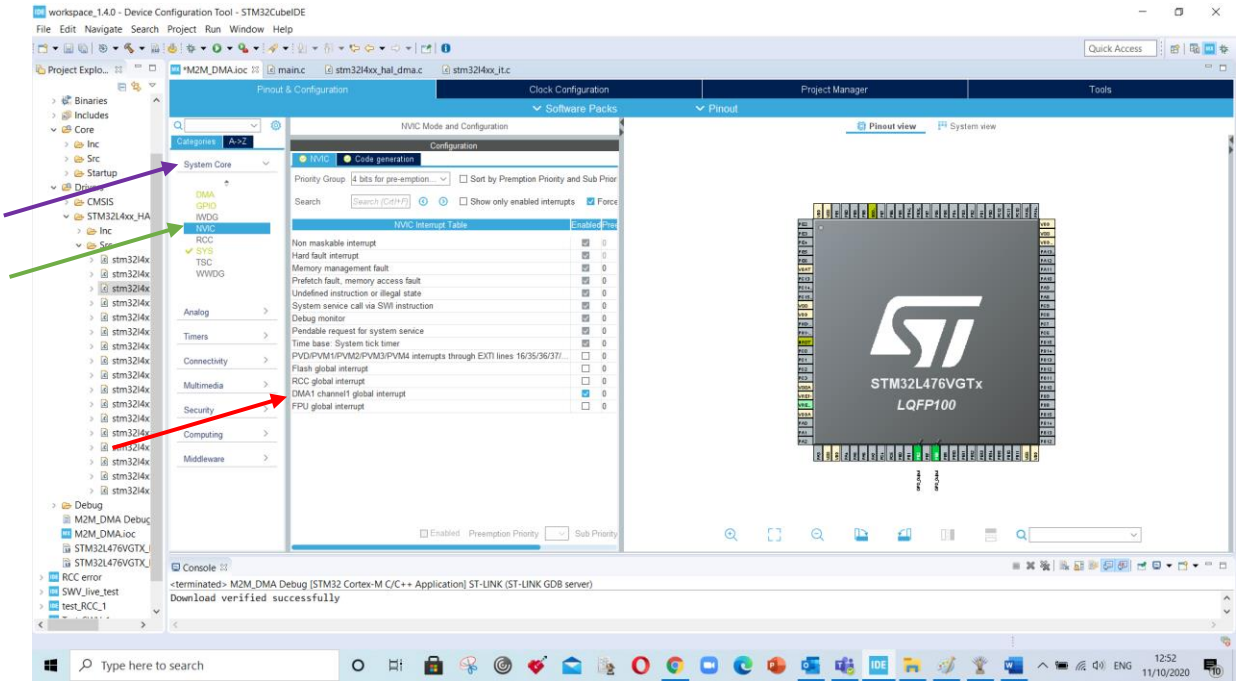
1- Learn how to configure DMA channel for a memory-to-memory transfer using NVIC interrupt.
2- Using logic analyzer to measure the CPU unloading and DMA transfer efficiency.
3- Developing understanding for various errors.

Step 1: Create a project in STM32CubeIDE and configure GPIO ‘PE8’ and ‘PB2’ as outputs.

Step 2: Go to ‘System Core’, select ‘DMA’, click on ‘Add’ and select ‘**MEMTOMEM**’ and use default DMA settings.



Step 3: From 'System Core' menu, select 'NVIC' and check the 'DMAv1 Channel 1 global interrupt'.



Step 4: Generate the code and for no DMA transfer write the codes on location shown by comments, as below:

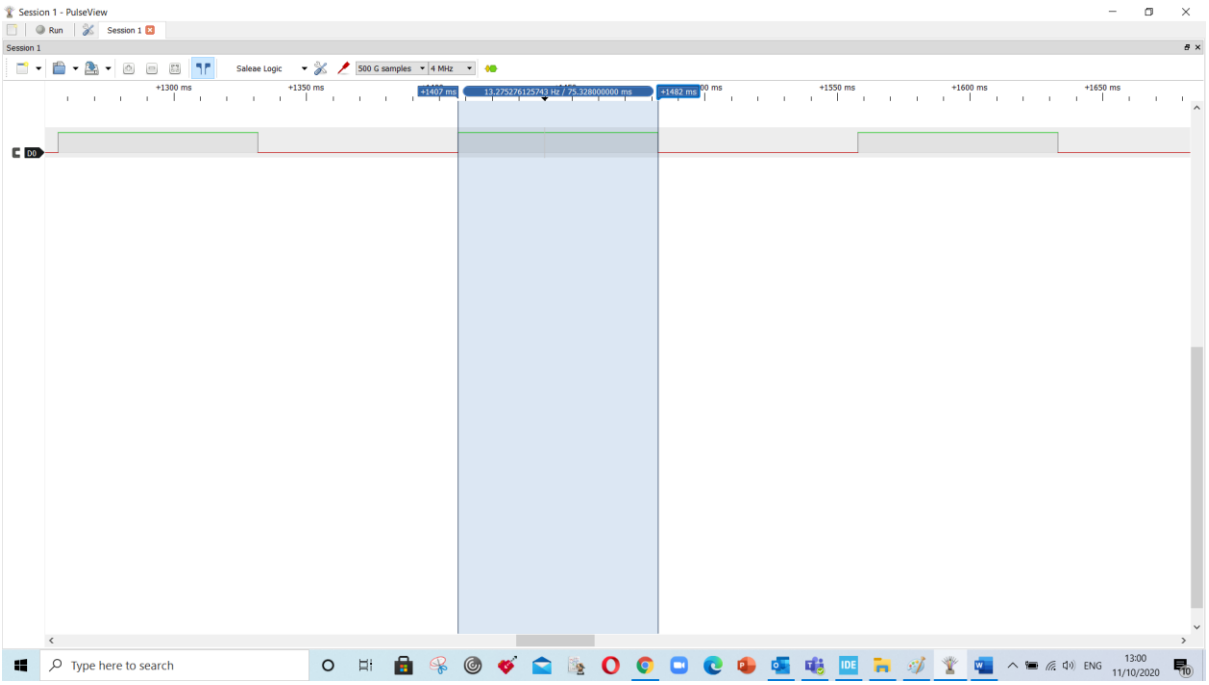
```
/* USER CODE BEGIN PTD */
#define M2M_DMA_size 8000
```

```
  /* USER CODE BEGIN 1 */
       uint32_t source[M2M_DMA_size];
       uint32_t destination[M2M_DMA_size];
       uint32_t i;
```

```
  /* USER CODE BEGIN 2 */
  for (i=0;i<=M2M_DMA_size; i++)
  {
         source[i]=i;
  }
```

```
    /* USER CODE BEGIN 3 */
         HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_8);

         for (i=0;i<=M2M_DMA_size; i++)
         {
                destination[i] = source[i];
         }
```

Step 5: Run ‘  ’ and build the code. You will not find any error. Now test your code execution by capturing the trace of ‘PE8’ in logic analyzer and measure the time period for data transfer (13Hz approx). Note the Sampling time is 4MHz.

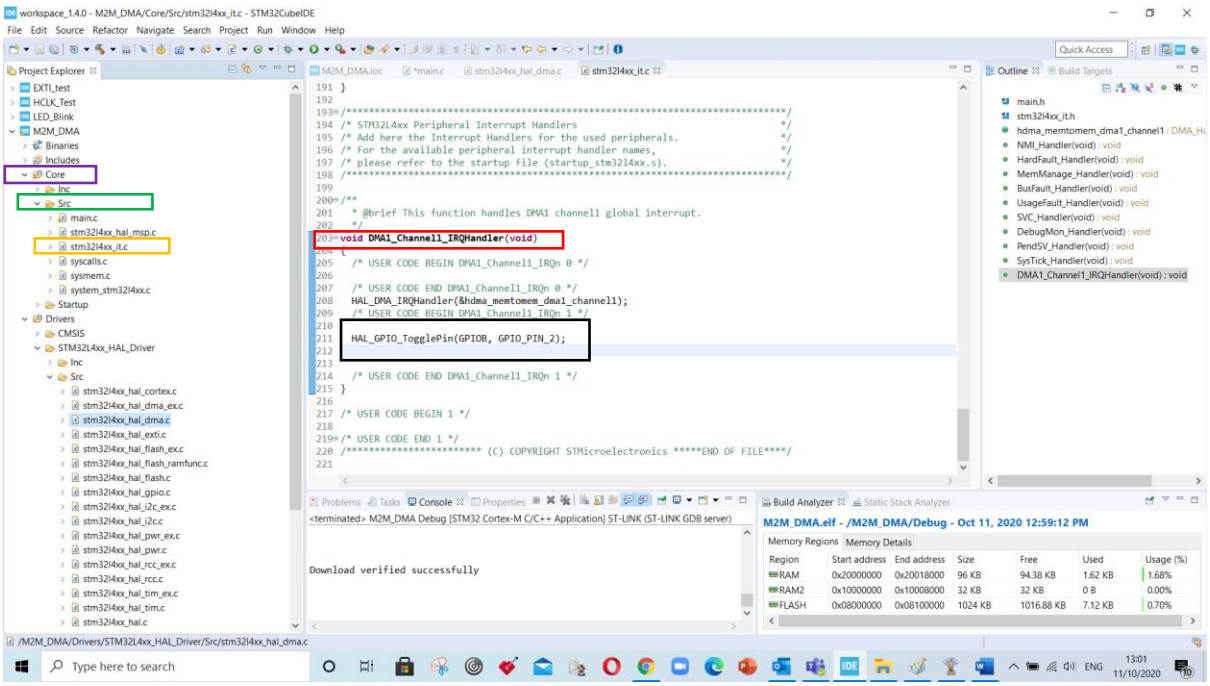


Step 6 (DMA Transfer): In generated code files, select folders ‘Core’, then select folder ‘Src’, then select ‘STM32L4xx_it.c’’ and in function ‘DMA1_channel1_IRQHandler’ add toggle led code as below:

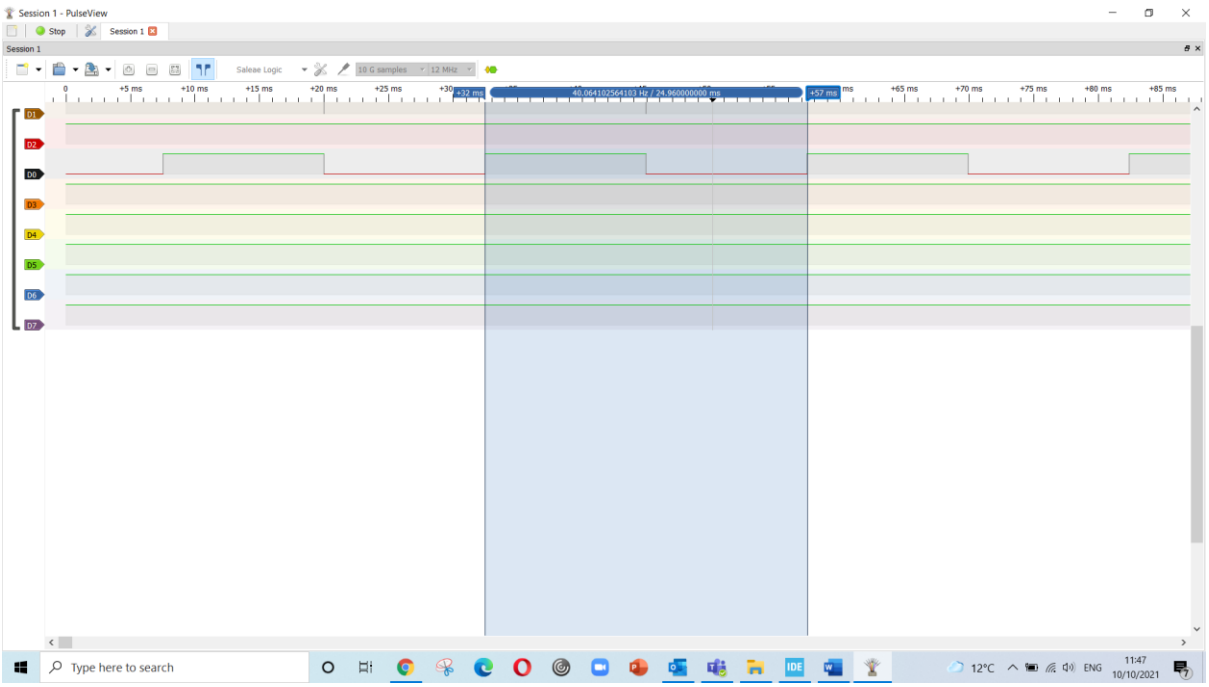
```
/* USER CODE BEGIN DMA1_Channel1_IRQn 1 */
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_2);
```

Start the DMA Transfer by enabling the DMA global interrupt. Comment the for-loop transfer code as below:

```
    /* USER CODE BEGIN 3 */

HAL_DMA_Start_IT(&hdma_memtomem_dma1_channel1, source, destination, M2M_DMA_size);

/*      HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_8);

        for (i=0;i<=M2M_DMA_size; i++)
        {
              destination[i] = source[i];
        }*/
```



Step 7: Run ‘  ’ and build the code. You will not find any error. Now test your code execution speed by Capturing the trace of ‘PB2’ in logic analyzer and measure the time period of waveform (40 Hz approx.)



To do task:

Try to introduce error and note their responses:

1- Define the higher sized of typedef variable 65k

```
/* USER CODE BEGIN PTD */
#define M2M_DMA_size 65000
```

2- Declare ‘i’ of 8-bit

```
Uint8_t i;
```

3- Use both for loop and DMA transfer together and observe the waveform
4- Optional: Add another DMA channel for memory-to-memory transfer and execute both at same time and observer the execution speed.

Activity 2: Low Power mode configuration

Aims:

Learn how to setup Low power mode on STM32L476VG MCU.

Objectives:

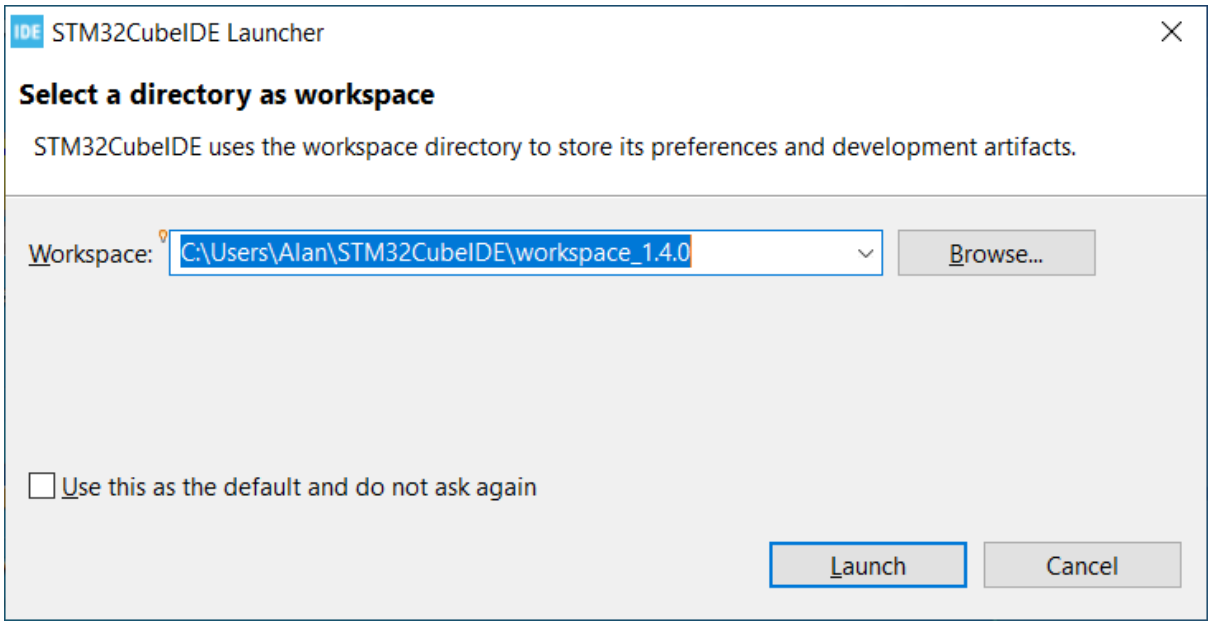
1. Configure GPIO & Generate the Code.
2. Activate the low power mode.
3. Debug the code and investigate Special function register (SFR) for low power setting bits.
4. Configure RCC and activate MCO.

Step 1:

Double click the 'STM32CubeIDE' icon on the desktop.

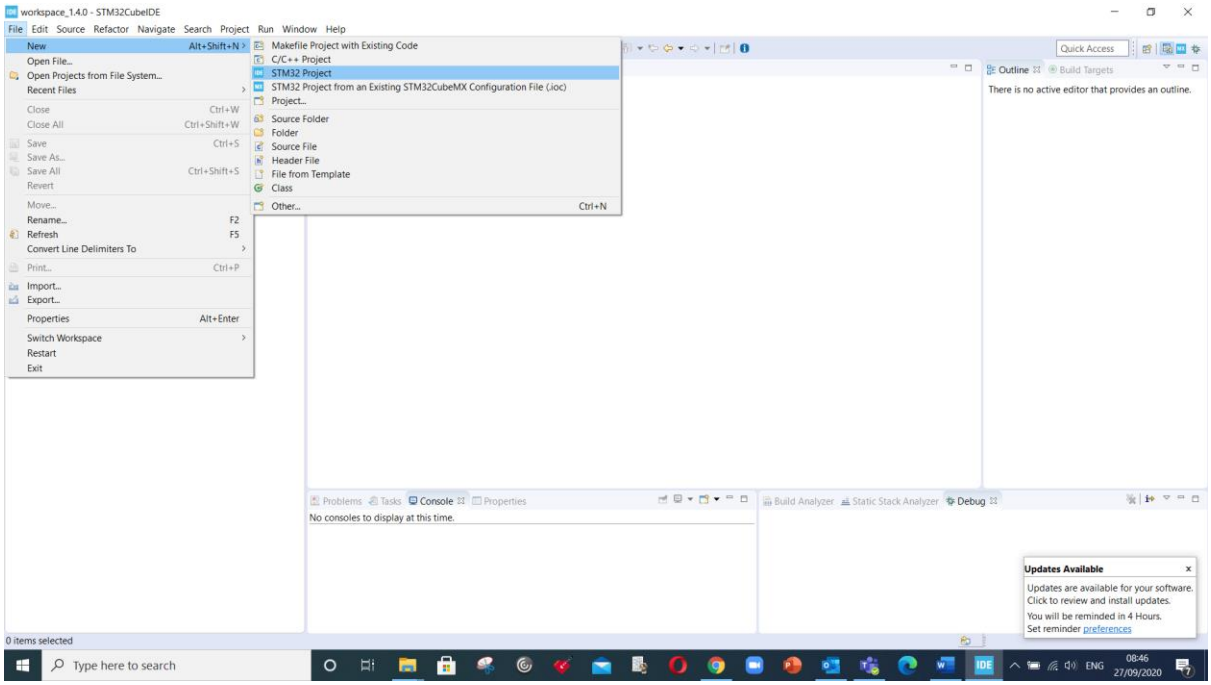
Step 2:

Select the workspace location, Default location is prefered



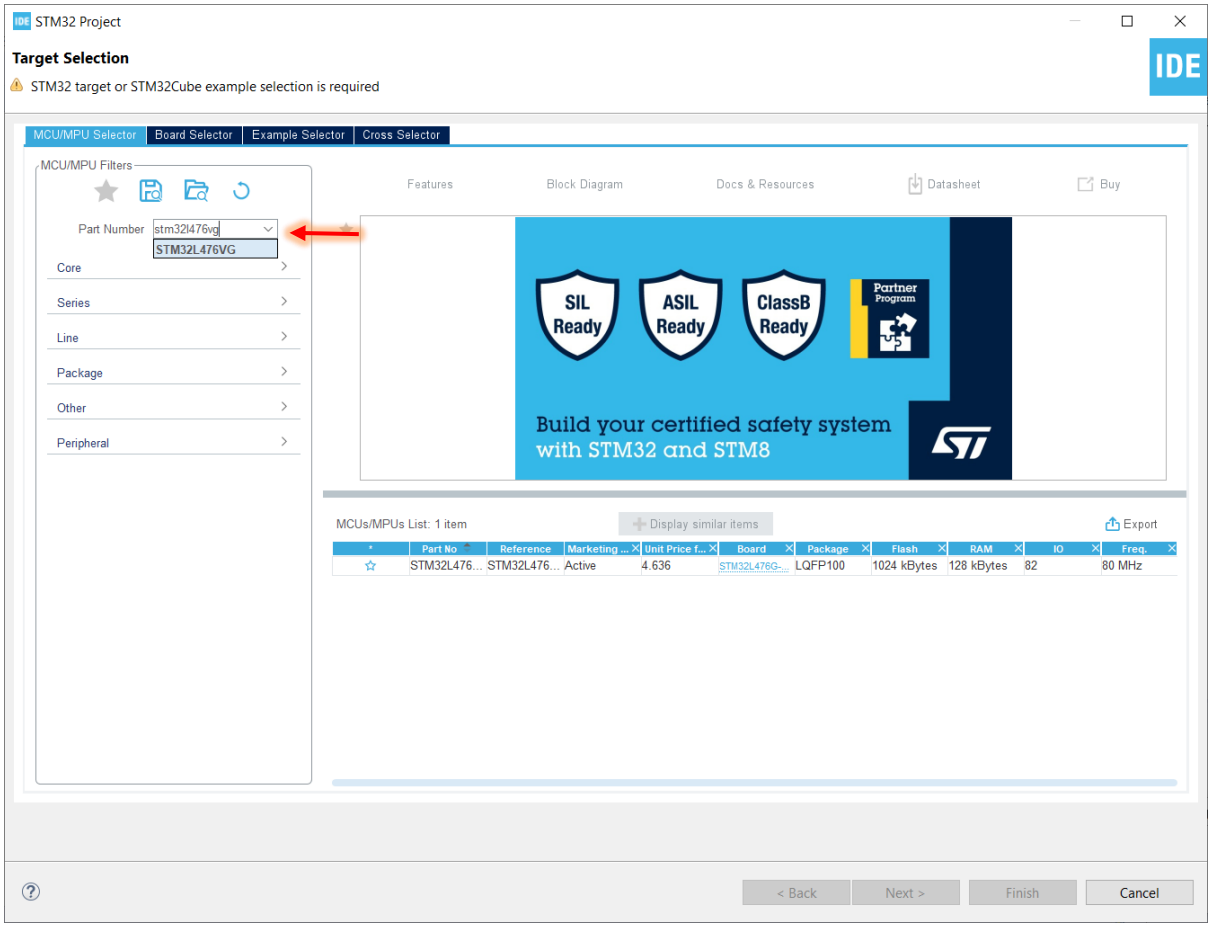
Step 3:

Open new project: File-> New-> STM32 Project



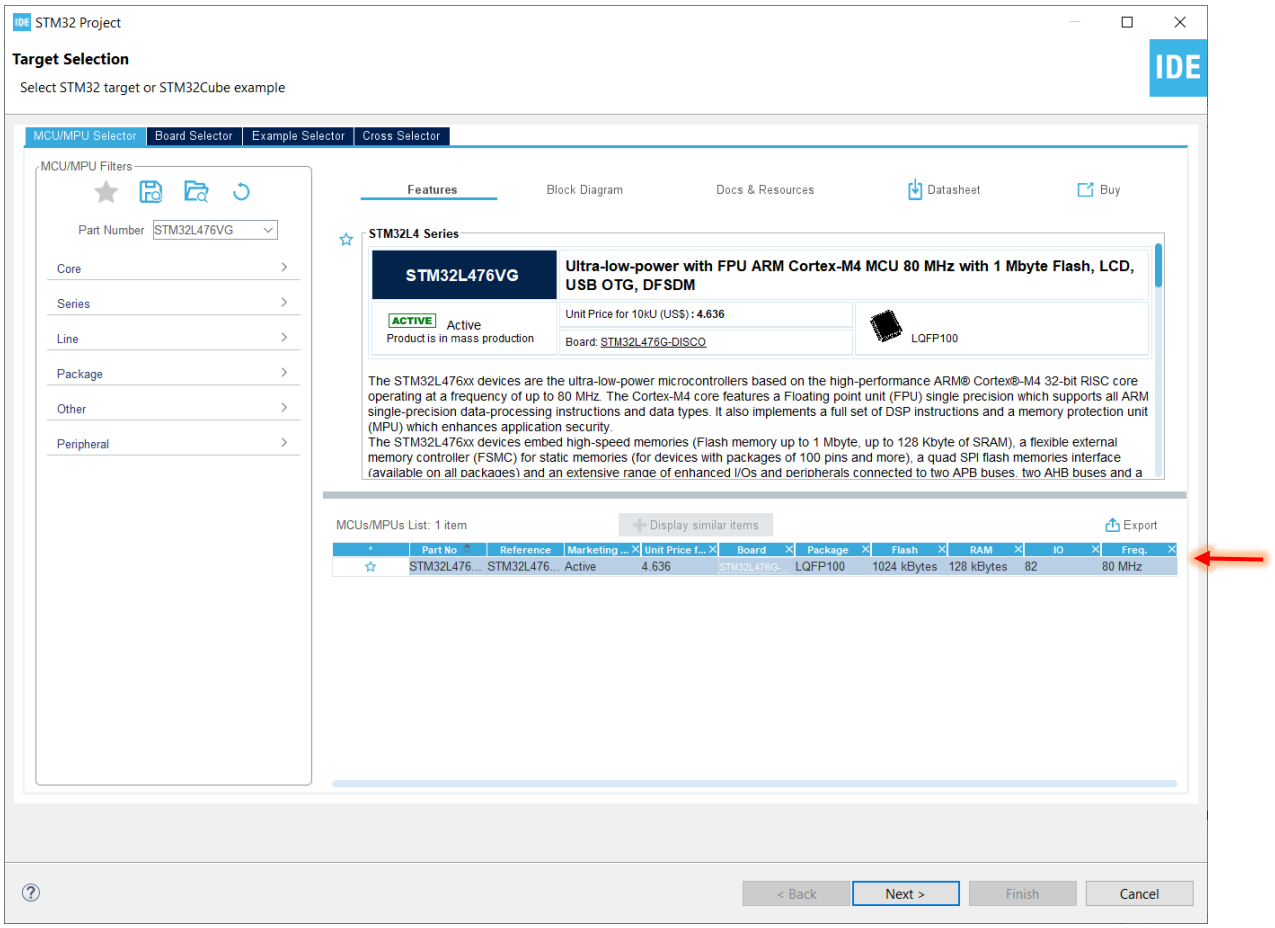
Step 4:

Type in ‘STM32L476VG’ in part number area.



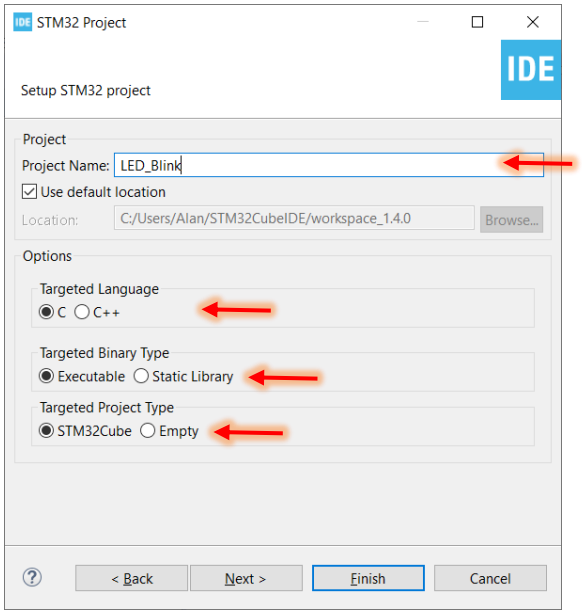
Step 5:

Select the MCU and Press ‘Next’



Step 6:

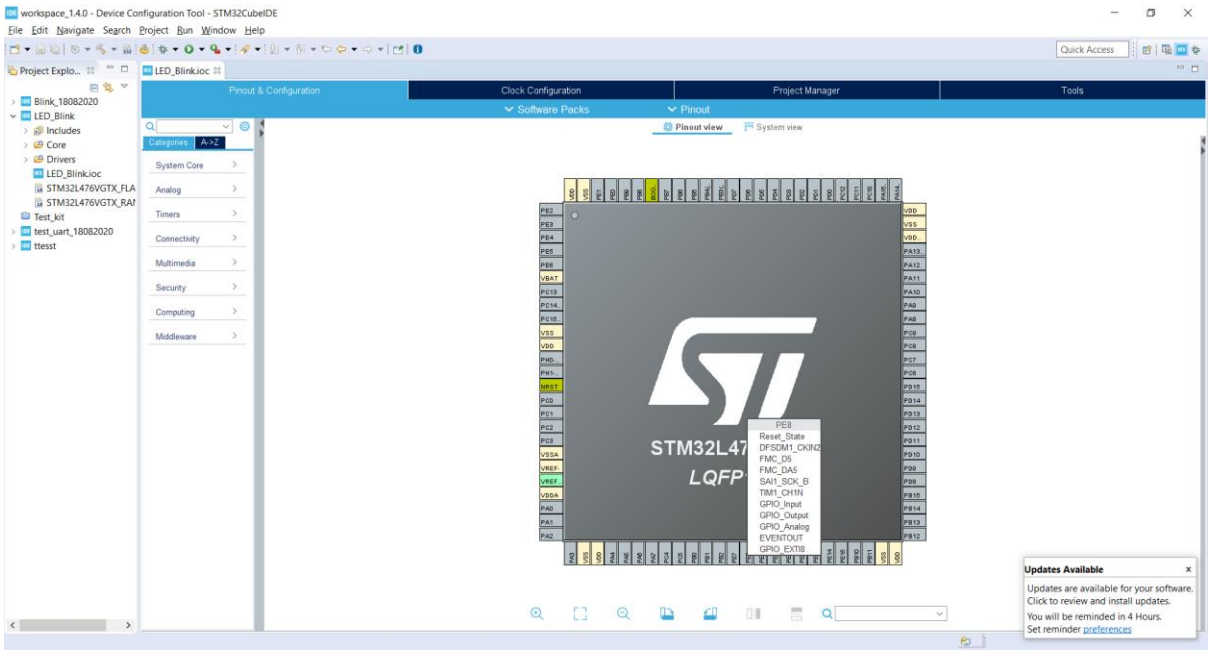
Name you project and observer the default options.

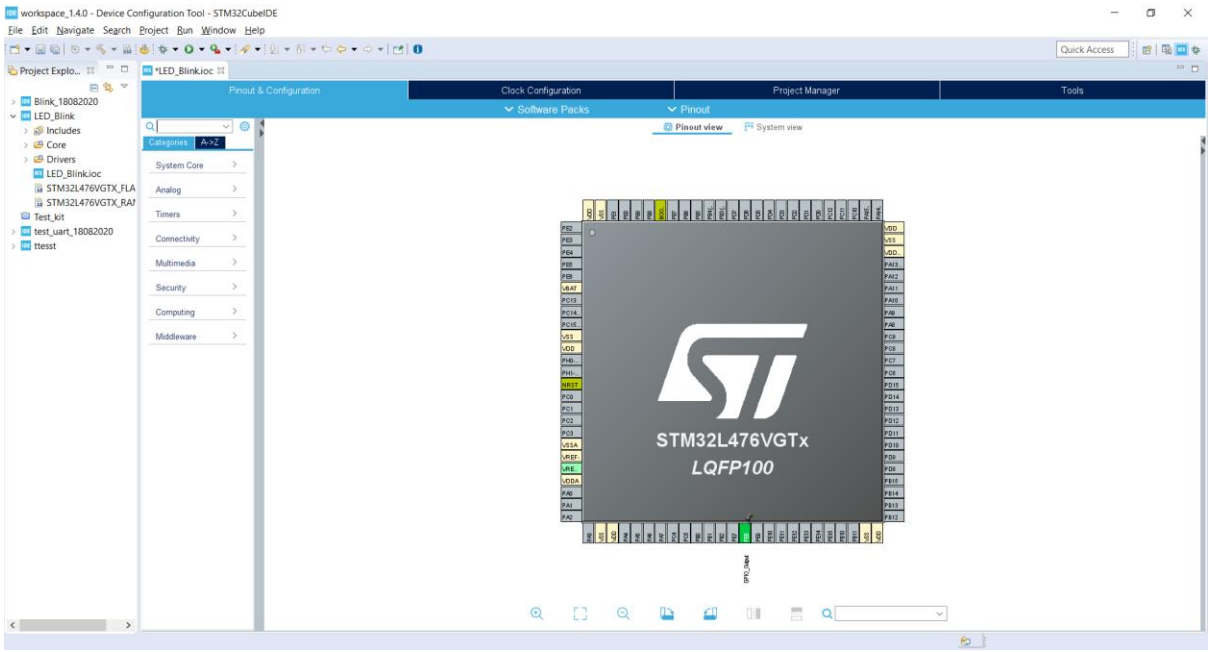


Step 7:

Left click on ‘PE8’ and select ‘GPIO_Output’.

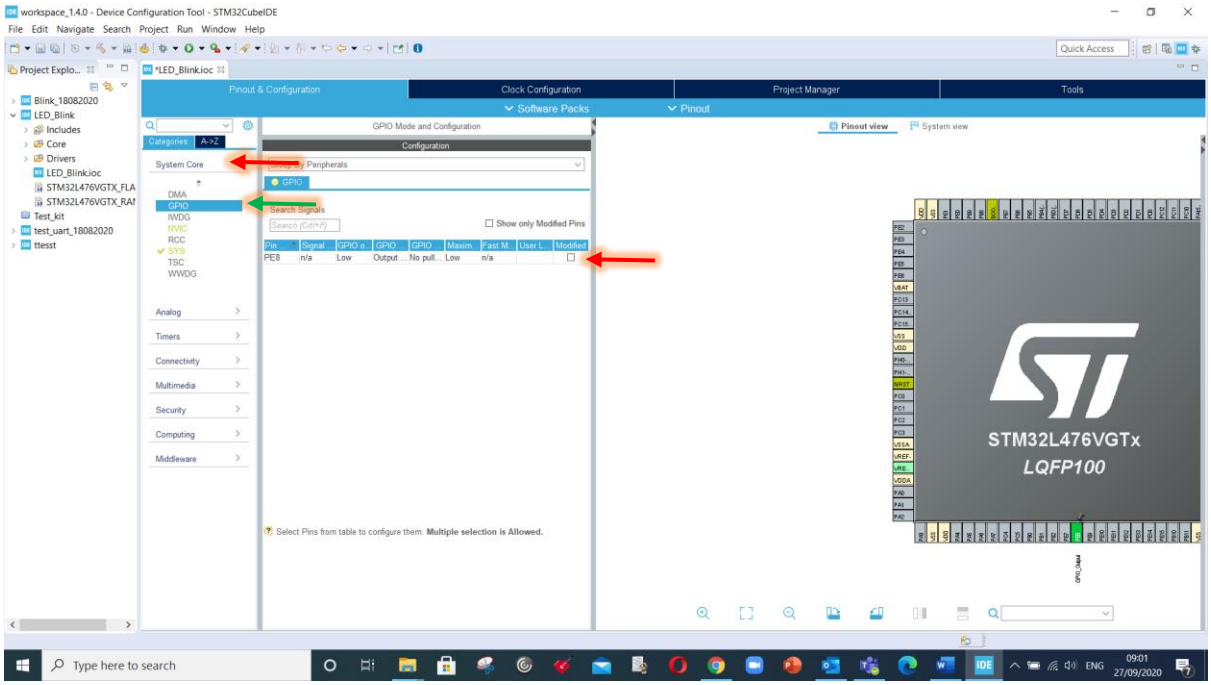
Note: The PIN PE8 is connected to RED LED on the discovery Kit.



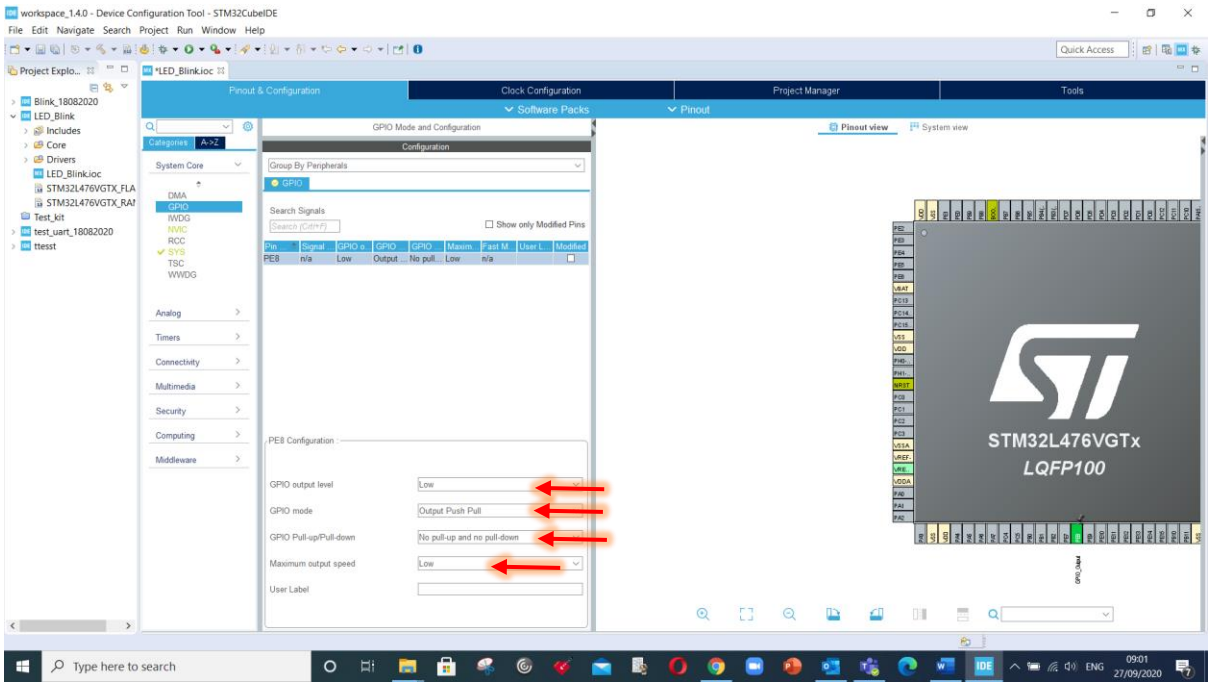


Step 8:

For GPIO configuration go to ‘System Core’ and click drop down menu and select ‘GPIO’.

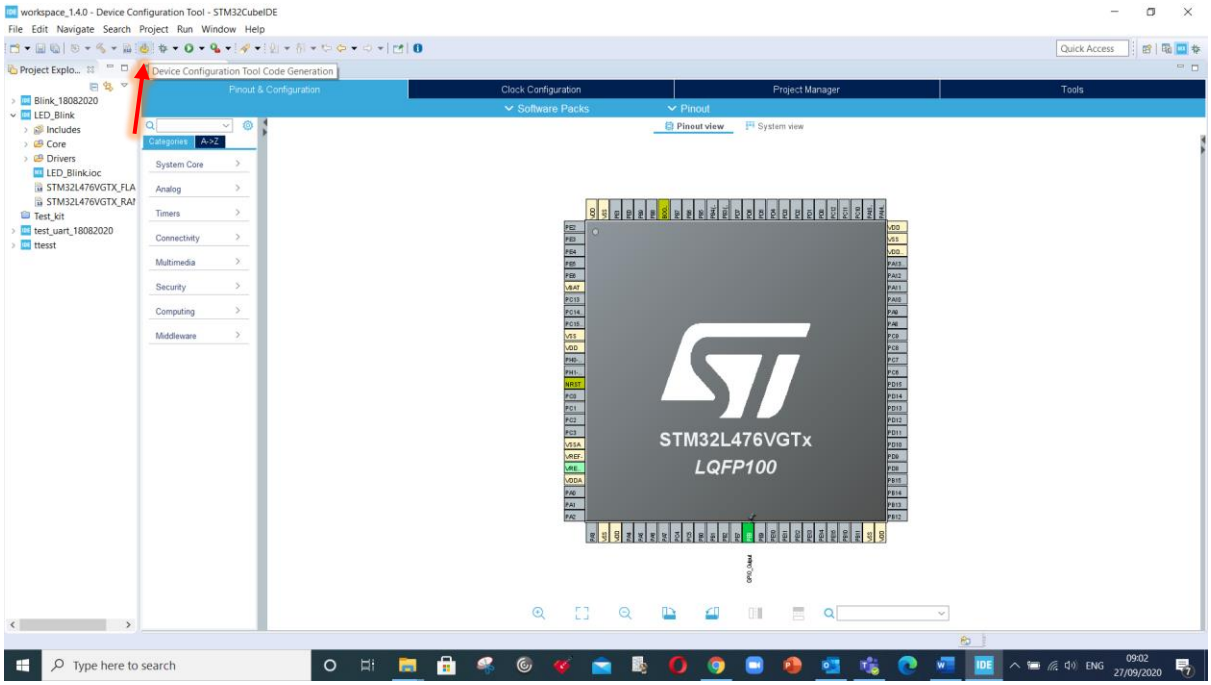


In GPIO menu select ‘PE8’ then it will the configuration make sure ‘GPIO output level’ is ‘low’, ‘GPIO mode’ is ‘Output push pull’, ‘GPIO Pull-Up/Pull-Down’ is ‘No pullup and pulldown’ and ‘maximum output speed’ is ‘Low’.



Step 9:

Click on ‘Project’ menu and select ‘Generate Code’ or use the icon as shown here

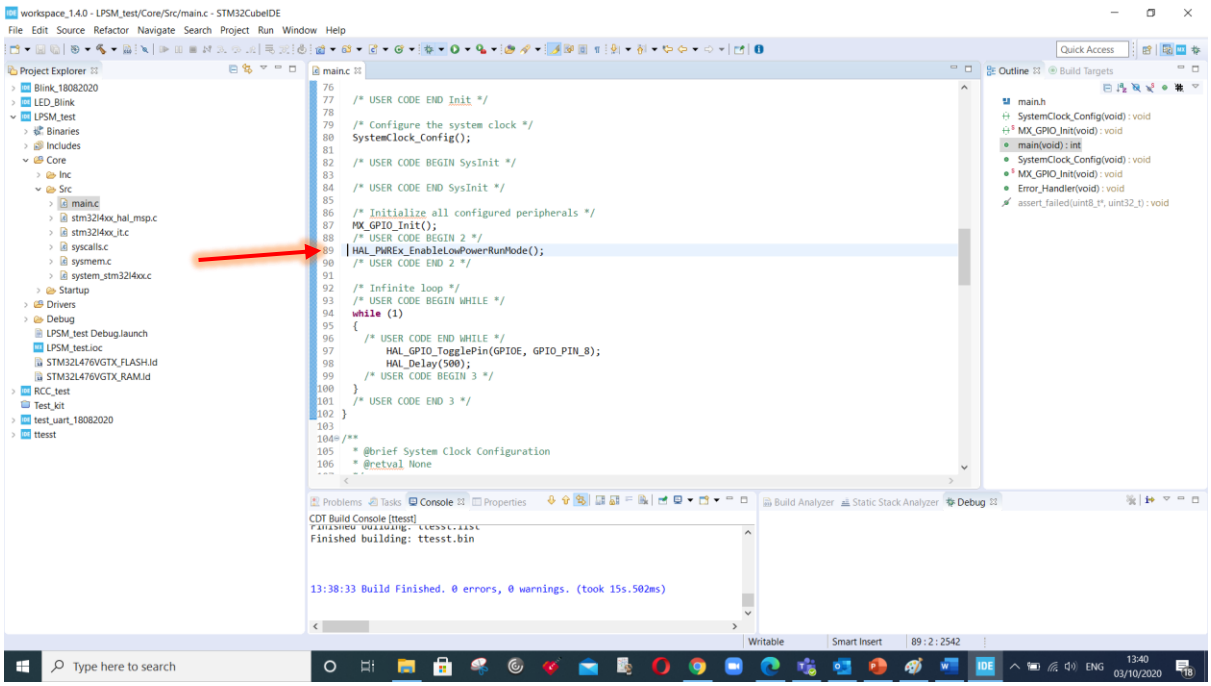


Step 10:

Enable the low power mode by using code below:

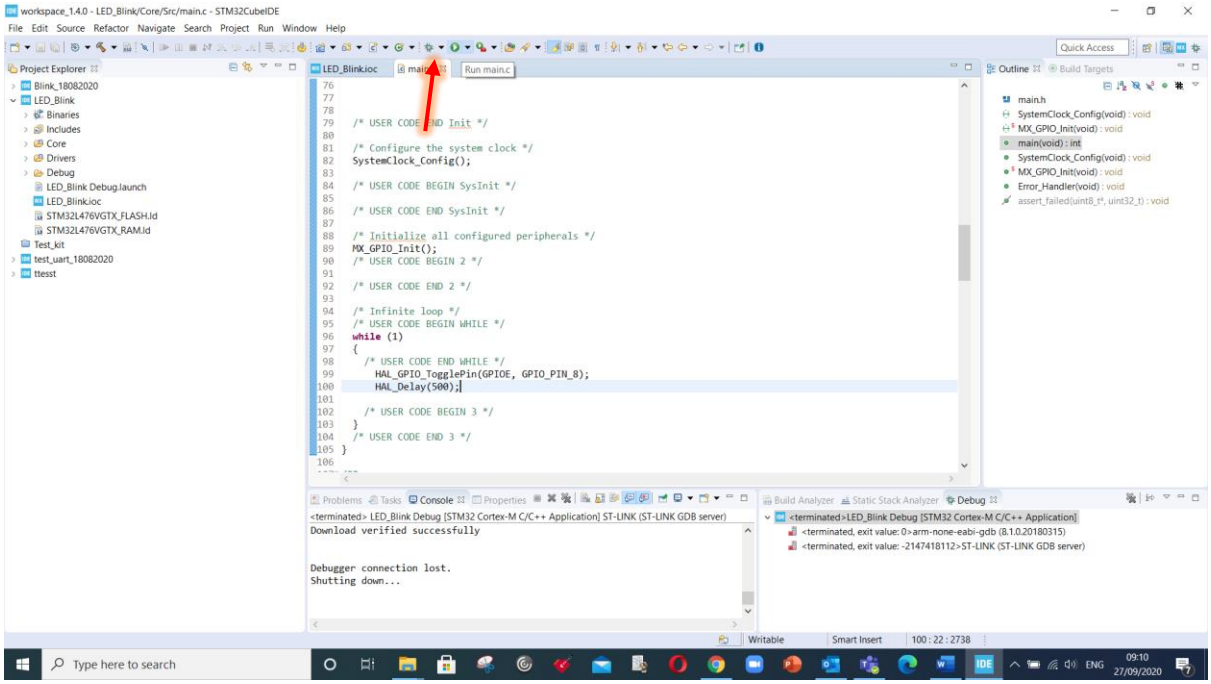
```
/* USER CODE BEGIN Init */
HAL_PWREx_EnableLowPowerRunMode();
```

Add the toggle LED Code HAL_GPIO_Toggle, use CTRL + SPACE to autofill the optins.



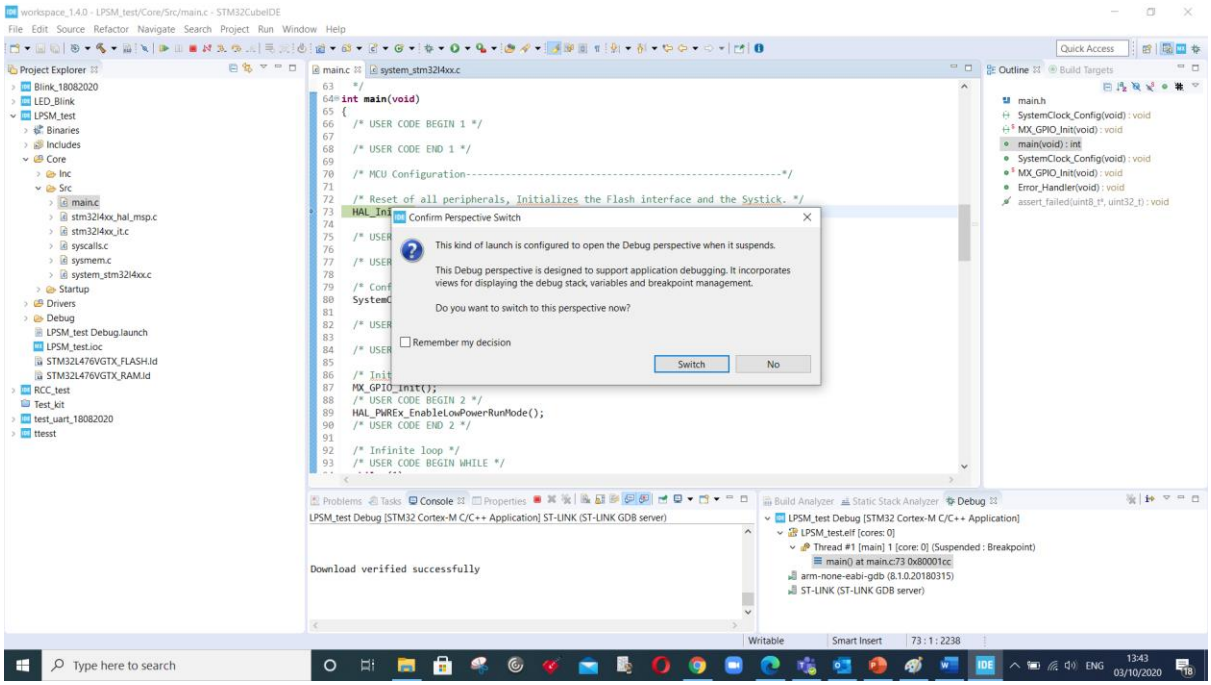
Step 11:

Press the Run button to build the project and then click on ‘Debug’ icon  .



Step 12:

Click ‘Switch’ on appearing menu.



Step 13:

Add a break point on toggle pin statement and then Click on ‘Resume’ icon to run/stop the execution. Observe LPR register-> Click on ‘SFRs’ and select ‘PWR’ and expand the list to view setting of ‘LPR’ bit which should change from ‘0’ to ‘1’.