Project_3.0_Interrupt

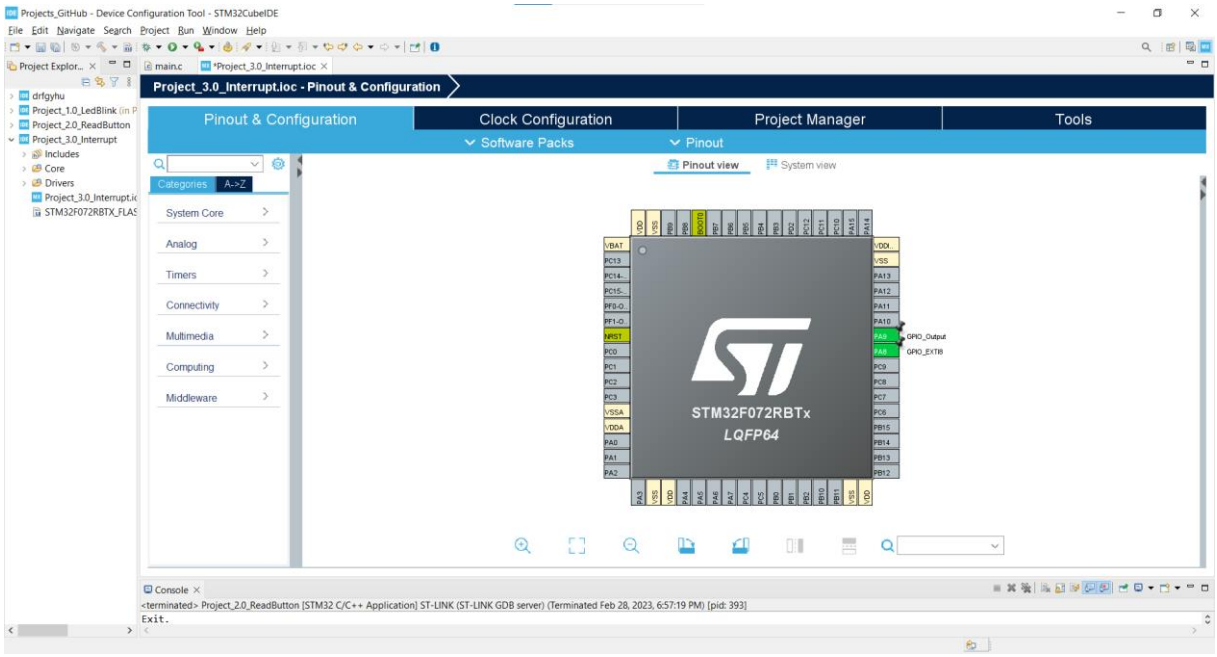
Bu projede, bir GPIO kesme pin'i ayarlayacağız. Ve bu kesmede bir çıkış pinini değiştireceğiz. GPIO kesme pinine bir buton bağlayacağız ve çıkış pinine ise yine bir led bağlayacağız. Kesme üretip ledi açma kapatma işlemi yapıcaz.

STEP 1 : Open STM32CubeIDE and Create New Project.

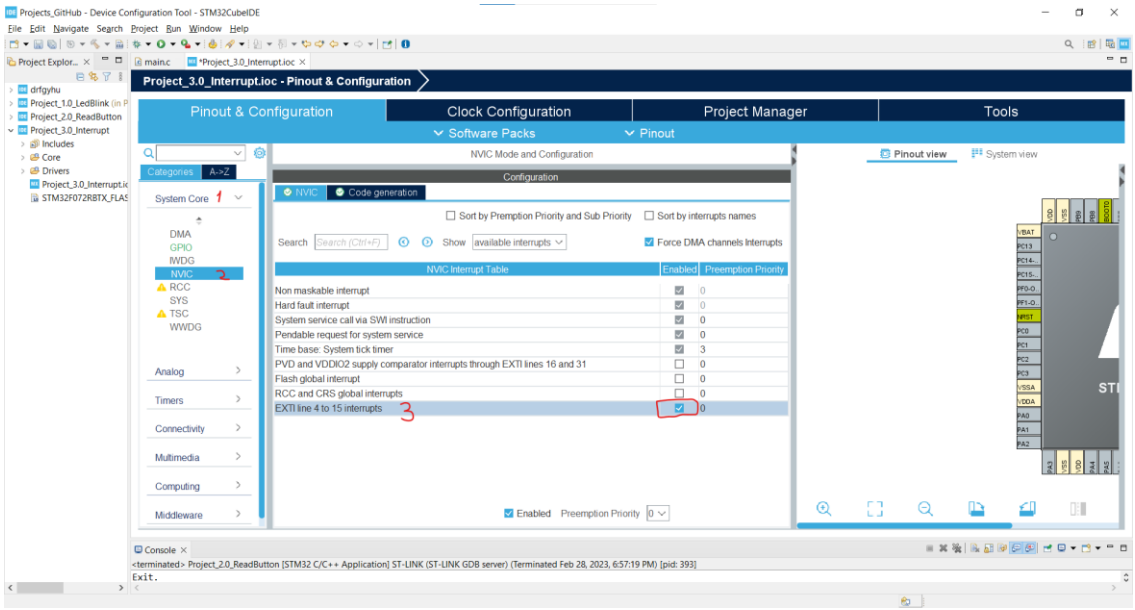
STEP 2 : Select Target microcontroller and double Click. My MCU is STM32F072RBT6.

STEP3 : Enter the project name and finish .

STEP 4 : Click the pin you want to set as input for interrupt and select GPIO_EXTIx/ GPIO_output. For example, in this project, I choose the A9 pin for output and A8 pin for input.

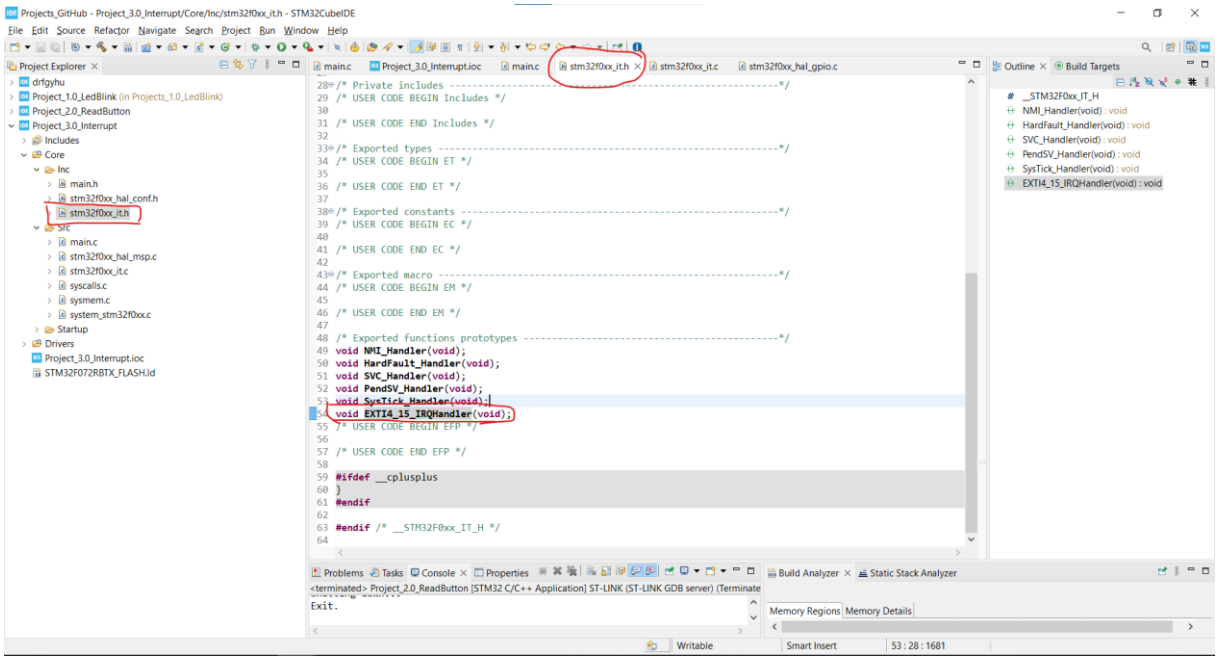


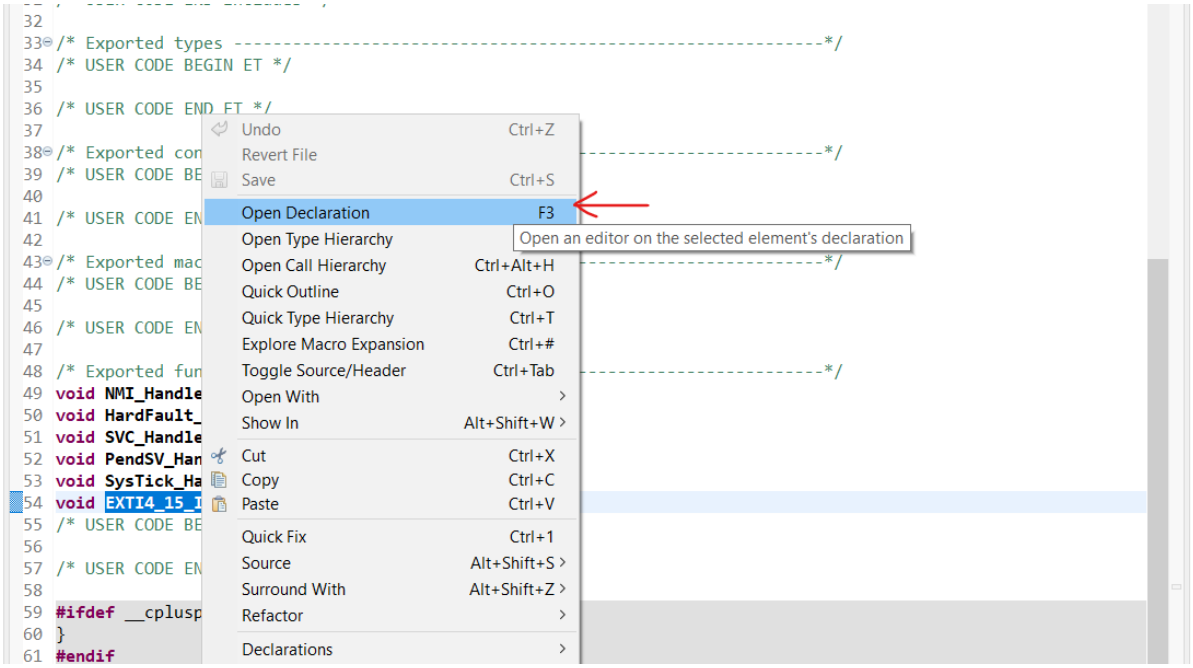
STEP 5: Open The NVIC Tab And Enable The EXTI line8 Interrupt.

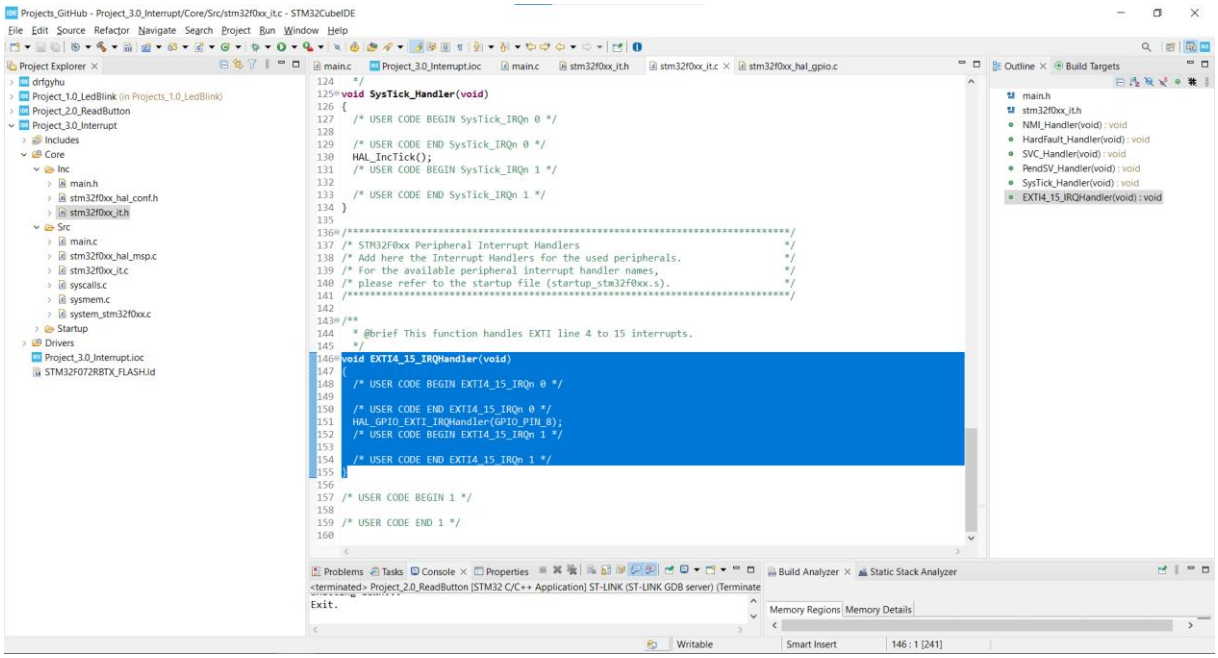


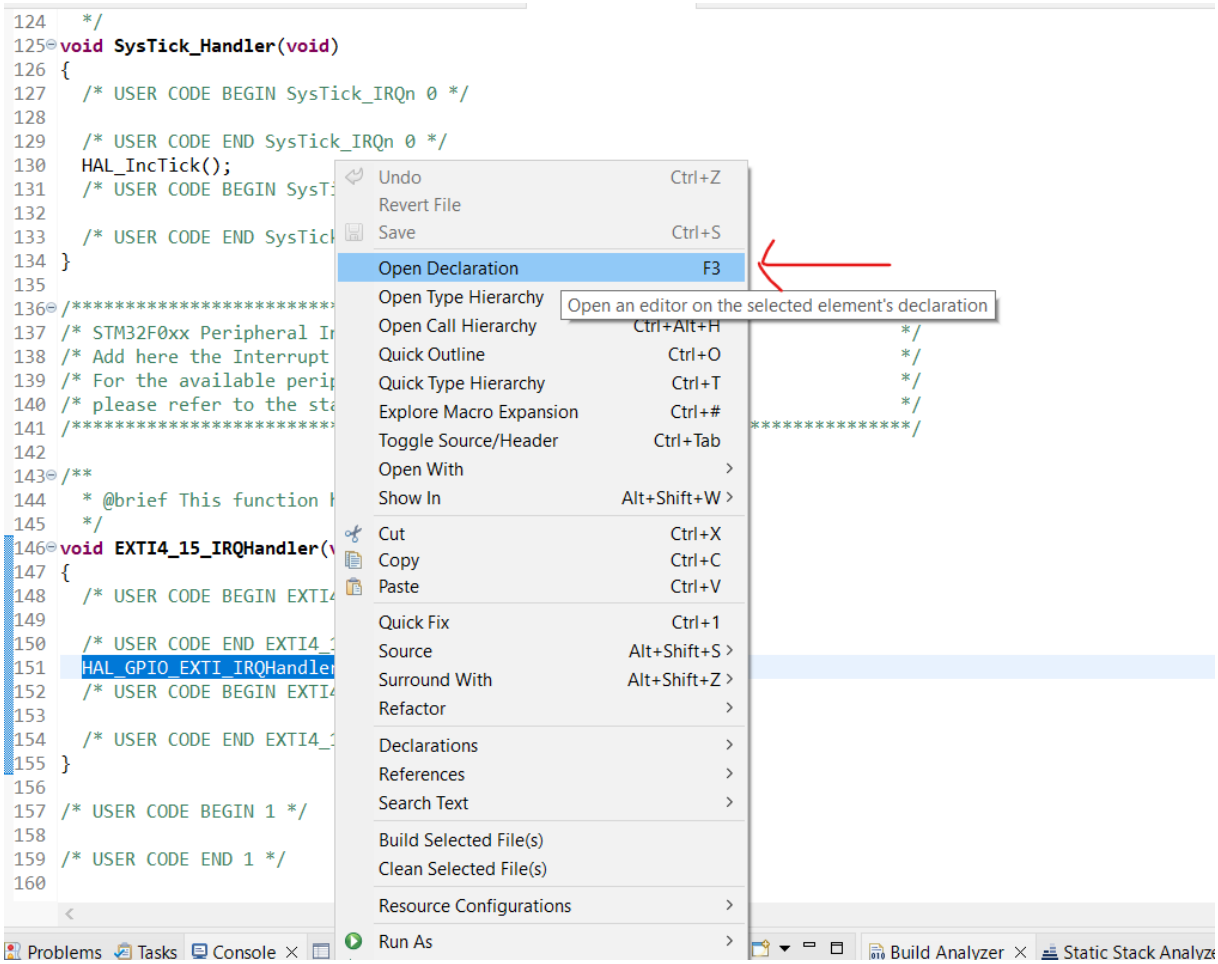
STEP 6 : Set the RCC External Clock Source and then CTRL + S to generate the project code. And we open our main.c file in the project files.

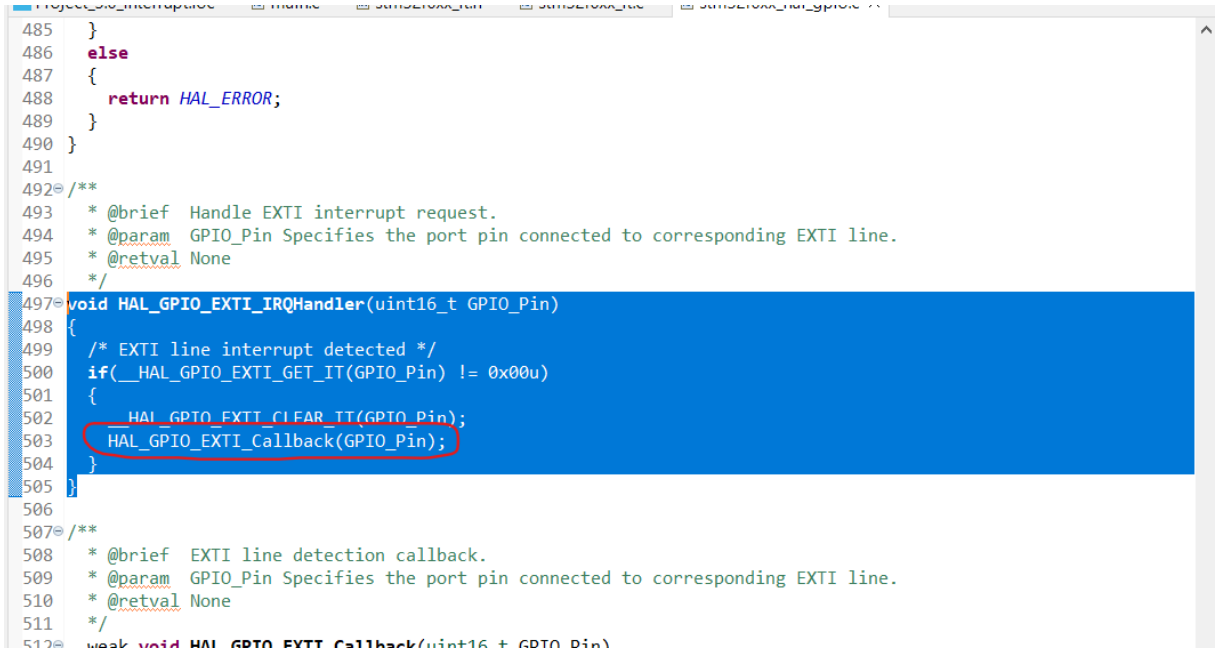
When this interruption occurs, the called ISR (Interrupt Service Routine) is located in the stm32f1xx_it.h file.




Undo Ctrl+Z
Revert File
Save Ctrl+S
Open Declaration F3
Open Type Hierarchy
Open Call Hierarchy Ctrl+Alt+H
Quick Outline Ctrl+O
Quick Type Hierarchy Ctrl+T
Explore Macro Expansion Ctrl+#
Toggle Source/Header Ctrl+Tab
Open With
Show In Alt+Shift+W
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Quick Fix Ctrl+1
Source Alt+Shift+S
Surround With Alt+Shift+Z
Refactor
Declarations
Open an editor on the selected element's declaration
/* Exported types */
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* USER CODE BEGIN EXTI4_15_IRQHandler(void) */
/* USER CODE END EXTI4_15_IRQHandler(void) */


Projects_GitHub - Project_3.0_Interrupt/Core/Src/stm32f0xx_it.c - STM32CubeIDE
void SysTick_Handler(void)
void EXTI4_15_IRQHandler(void)
/* USER CODE BEGIN EXTI4_15_IRQn 0 */
HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_8);
/* USER CODE END EXTI4_15_IRQn 1 */





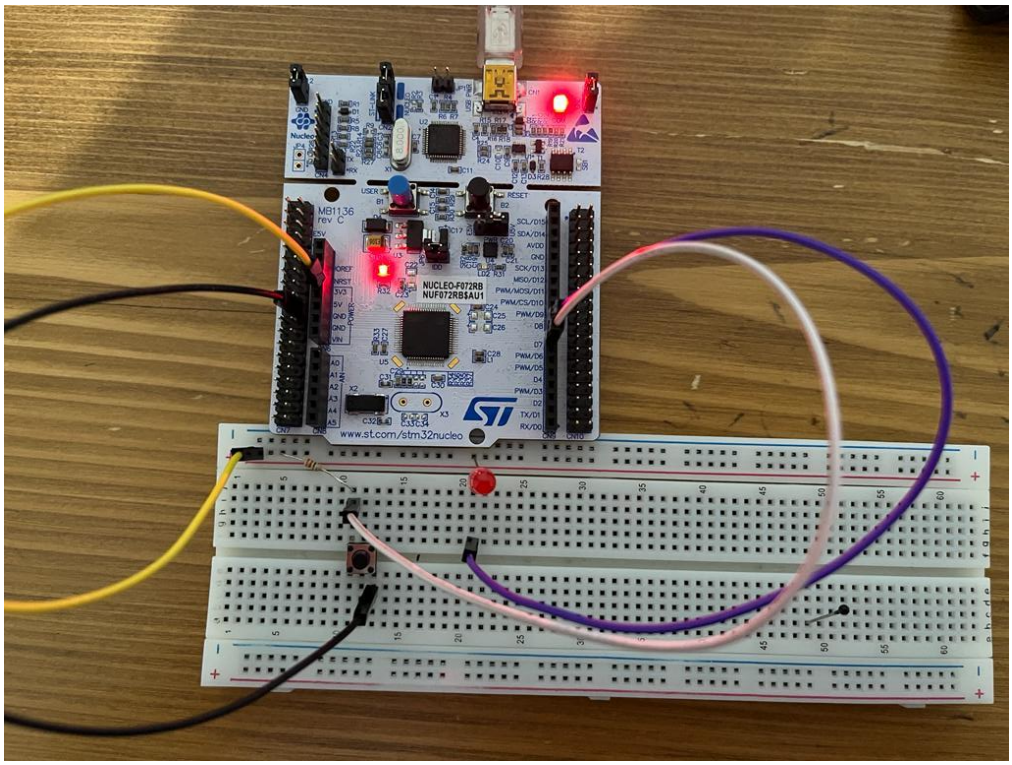
This function clears the interrupt source, then calls the ISR handler callback function. Now we will write code for this callback function (main.c file)

Project_3.0_Interrupt.ioc | main.c | stm32f0xx_it.h | stm32f0xx_it.c | stm32f0xx_hal_gpio.c

```
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)  // EXTI Line[4:15] External Interrupt  CallBackFunction
{
    if(GPIO_Pin == GPIO_PIN_8)                  // If The Interrupt source Is EXTI Line8 (A8 Pin)
    {
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_9);  // Toggle The Output Pin
    }

}
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

```

STEP 7 : We press RUN to compile the code and upload it to the board.

STEP 8 : Now it's time to connect the led to the board.



That's it.

When we press the button, an interrupt will occur and the interrupt function will toggle the output pin. If the led is off it will turn on, if it is on it will turn off.