**NOSH Robotics – Assignment**

-Deepak S

1)

**Q-1 Switch & LED Blinking with STM32G070RB:**

There is a tactile switch (**only one switch**) and LED connected to the microcontroller in a NUCLEO-G070RB board. The LED is off initially. Depending on switch press, the LED blinks in following way.

1st Switch press: LED blinks at frequency of 0.5 Hz.

2nd Switch press: LED blinks at frequency of 1 Hz.

3rd Switch press: LED blinks at frequency of 2 Hz.

4th Switch press: LED turns off.

5th Switch press: considered 1st switch press and the LED blinking cycle repeats.

Create circuit and code for target microcontroller. Assume relevant functions. Also, optimize for power consumption. Use micro-controllers low power modes to achieve it.

(After the assignment and 1st video call interview, in the in person interview at Nosh’s office this needs to be showcase on a NUCLEO board.)

The LED is present in PA5 in the board, and the button is present in PC13. We can either use this internal button and LED or we could use the same GPIO to connect an external switch and LED at the same position like this:

A circuit board with wires and numbers

AI-generated content may be incorrect.

Design was made using EasyEDA. The switch in this case is a pull up, so the logic will be HIGH (1) and becomes LOW (0) only when it is pressed.

Code (main.c):

**#include** "main.h"

**void** **SystemClock\_Config**(**void**);

**static** **void** **MX\_GPIO\_Init**(**void**);

**int** **main**(**void**)

{

**HAL\_Init**();

**SystemClock\_Config**();

**MX\_GPIO\_Init**();

**int** count = 0;

**int** last\_button\_state = 0;

**while** (1)

{

**int** button\_state = **HAL\_GPIO\_ReadPin**(GPIOC, GPIO\_PIN\_13);

**if** (button\_state == *GPIO\_PIN\_RESET*) //pull up, hence the logic becomes 0 when button is pressed

{

count++;

**HAL\_Delay**(200); // Debounce delay

}

last\_button\_state = button\_state;

**switch** (count % 4)

{

**case** 1: // 0.5 Hz => 1 sec ON, 1 sec OFF

**HAL\_GPIO\_WritePin**(GPIOA, GPIO\_PIN\_5, *GPIO\_PIN\_SET*);

**HAL\_Delay**(1000);

**HAL\_GPIO\_WritePin**(GPIOA, GPIO\_PIN\_5, *GPIO\_PIN\_RESET*);

**HAL\_Delay**(1000);

**break**;

**case** 2: // 1 Hz => 0.5 sec ON, 0.5 sec OFF

**HAL\_GPIO\_WritePin**(GPIOA, GPIO\_PIN\_5, *GPIO\_PIN\_SET*);

**HAL\_Delay**(500);

**HAL\_GPIO\_WritePin**(GPIOA, GPIO\_PIN\_5, *GPIO\_PIN\_RESET*);

**HAL\_Delay**(500);

**break**;

**case** 3: // 2 Hz => 0.25 sec ON, 0.25 sec OFF

**HAL\_GPIO\_WritePin**(GPIOA, GPIO\_PIN\_5, *GPIO\_PIN\_SET*);

**HAL\_Delay**(250);

**HAL\_GPIO\_WritePin**(GPIOA, GPIO\_PIN\_5, *GPIO\_PIN\_RESET*);

**HAL\_Delay**(250);

**break**;

**case** 0:

**default**:

**HAL\_GPIO\_WritePin**(GPIOA, GPIO\_PIN\_5, *GPIO\_PIN\_RESET*);

**HAL\_Delay**(100);

**break**;

}

**HAL\_PWR\_EnterSLEEPMode**(PWR\_MAINREGULATOR\_ON, PWR\_SLEEPENTRY\_WFI); // low power consumption mode

}

}

**void** **SystemClock\_Config**(**void**)

{

**RCC\_OscInitTypeDef** RCC\_OscInitStruct = {0};

**RCC\_ClkInitTypeDef** RCC\_ClkInitStruct = {0};

**HAL\_PWREx\_ControlVoltageScaling**(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSIDiv = RCC\_HSI\_DIV1;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;

**if** (**HAL\_RCC\_OscConfig**(&RCC\_OscInitStruct) != *HAL\_OK*)

{

**Error\_Handler**();

}

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

**if** (**HAL\_RCC\_ClockConfig**(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != *HAL\_OK*)

{

**Error\_Handler**();

}

}

**static** **void** **MX\_GPIO\_Init**(**void**)

{

**GPIO\_InitTypeDef** GPIO\_InitStruct = {0};

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

**HAL\_GPIO\_WritePin**(LED\_GREEN\_GPIO\_Port, LED\_GREEN\_Pin, *GPIO\_PIN\_RESET*);

GPIO\_InitStruct.Pin = GPIO\_PIN\_13;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

**HAL\_GPIO\_Init**(GPIOC, &GPIO\_InitStruct);

GPIO\_InitStruct.Pin = LED\_GREEN\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_HIGH;

**HAL\_GPIO\_Init**(LED\_GREEN\_GPIO\_Port, &GPIO\_InitStruct);

}

**void** **Error\_Handler**(**void**)

{

**\_\_disable\_irq**();

**while** (1)

{

}

}

**#ifdef** USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

**void** assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

**#endif** /\* USE\_FULL\_ASSERT \*/

Uploaded the project separately in github.