GPIO INPUT: PULSANTE
Per l'input GPIO i parametri sono:

- pull-up, pull-down o floating (né pull-up né pull-down); le resistenze pull-up e pull-down sono usate nel caso di input ad alta impedenza (esempio: pulsanti);
- trigger di Schmitt attivo (ma non si capisce se e come si può disattivare; notare che disattivare un trigger di Schmitt su un input GPIO di cui non si sa nulla non è in genere una buona idea).

Il pulsante è il tipico input GPIO che è pull-up o pull-down quando è chiuso, ed è flottante (alta impedenza) quando è aperto, quindi il GPIO deve avere una resistenza di pull-down o pull-up per gestire la situazione di pulsante aperto. Quale resistenza dipende se il pulsante è collegato alla terra o all'alimentazione. Nel caso della scheda Nucleo il pulsante B1 (pin PC13) è collegato all'alimentazione, pertanto serve una resistenza di pull-down. Però se guardiamo lo schema della scheda (MB1137.pdf, pagina 2) vediamo che ha già una sua resistenza di pull-down sulla scheda (R58). Quindi non bisogna attivare le resistenze di pull-up / pull-down. Si può notare che questa è già la configurazione di default (selezionare nel configuratore System Core > GPIO, nella tabella il pin PC13, e quindi GPIO Pull-up / pull-down: No pull-up and no pull-down). Codice generato:

main.c:
```
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  ...

  /* Configure GPIO pin : USER_Btn_Pin */
  GPIO_InitStruct.Pin = USER_Btn_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING; /* interrupt! */
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(USER_Btn_GPIO_Port, &GPIO_InitStruct);
  ...
}
```

main.h:
```
...
#define USER_Btn_Pin GPIO_PIN_13
#define USER_Btn_GPIO_Port GPIOC
...
```

Il nostro codice che usa il polling:

main.c:
```
int main(void)
{
...
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    HAL_GPIO_WritePin(GPIOB, LD1_Pin|LD3_Pin|LD2_Pin,
        HAL_GPIO_ReadPin(USER_Btn_GPIO_Port, USER_Btn_Pin));
    /* USER CODE END 3 */
```

```
  }
```

Effetto: quando il pulsante è premuto le tre luci si accendono, quando è lasciato le tre luci si spengono. Notare che il pulsante è impostato in modalità interrupt (IT_RISING), perché non c'è modo di impostarlo in modalità non interrupt!

Adesso consideriamo il pulsante comandato in interrupt. Dal manuale dell'HAL (sez. 2.11.2 UM1905) occorre:

- impostare la modalità interrupt del GPIO del pulsante sia sul fronte di salita che sul fronte di discesa del segnale del pulsante (IT_RISING_FALLING);
- impostare la priorità ed attivare le interruzioni invocando le funzioni HAL_NVIC_SetPriority() e HAL_NVIC_EnableIRQ();
- editare la funzione HAL_GPIO_EXTI_Callback() ed inserirvi il codice custom da eseguire sull'evento di interrupt.

Le prime due configurazioni si possono impostare dal solito device configuration tool, il primo selezionando System Core > GPIO, nella tabella il pin PC13, e quindi GPIO Mode: External Interrupt Mode with Rising/Falling edge trigger detection, il secondo selezionando System Core > NVIC, e nella tabella EXTI line[15:10] interrupts e flaggare Enabled. Il codice generato è:

main.c:
```
static void MX_GPIO_Init(void)
{
  ...
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
  ...

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
}
```

L'aggancio del nostro codice custom alla routine di interrupt del pulsante si effettua facendo l'overriding della funzione weak HAL_GPIO_EXTI_Callback() come segue:

main.c:
```
...
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
      if (GPIO_Pin == USER_Btn_Pin) {
            HAL_GPIO_TogglePin(GPIOB, LD1_Pin | LD2_Pin | LD3_Pin);
      }
}
/* USER CODE END 4 */
...
```

Cosa succede se impostiamo la modalità interrupt del GPIO solo sul fronte di salita (IT_RISING)? Dal momento che il pulsante della scheda Nucleo è collegato all'alimentazione, quando il pulsante viene chiuso sull'ingresso GPIO viene impostata una tensione alta, mentre quando viene aperto la resistenza di pull-down porta sull'ingresso GPIO una tensione bassa. Pertanto, si ha un fronte di salita quando schiaccio il pulsante, e un fronte di discesa quando lo rilascio. Le luci dovrebbero commutare solo quando il pulsante viene schiacciato, e non dovrebbe succedere nulla quando il pulsante viene rilasciato. In realtà si verifica che questo succede quasi sempre, ma a volte le luci non commutano quando il pulsante viene schiacciato, oppure commutano anche quando il pulsante

viene rilasciato. Il motivo di questo comportamento è una non idealità del pulsante, che viene detta bouncing, e che verrà trattata in un laboratorio successivo.