Create a new STM32 Project to blink (on/off) with a period of 0.2 sec. for an on board LED

```
while (1)
{
  /* USER CODE END WHILE */
     HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5); //Toggle the state of pin PC9
     HAL_Delay(200); //delay 200ms
  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}
```

Create a new STM32 Project to toggle an LED (on/off) with pushing USER push-button. (Debouching is required)

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */

uint32_t init_time = 0; // No.3

while (1)
{
          if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == GPIO_PIN_RESET) {
              uint32_t temp = HAL_GetTick(); // get current time
              if (temp - init_time > 20) {
                      HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
              }
              init_time = temp;
          }

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

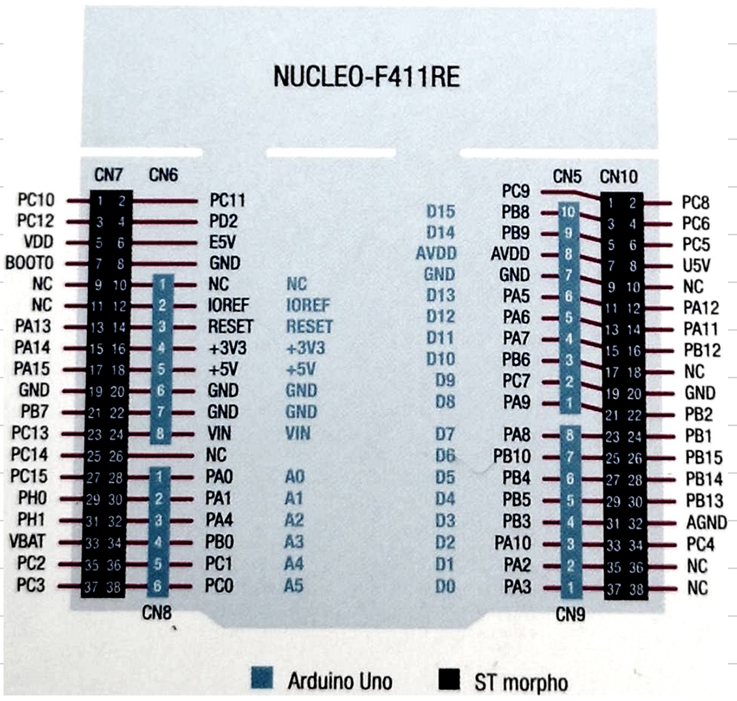
Implement the following:

1 The indicator LEDs blinks every one second (500ms on, 500ms off) after reset
2 Pushing the USER button the first time will change the state such that only one LED is blinking. (choose any)
3 Pushing the USER button the second time will change the state such that only two LEDs are blinking.
4 Pushing the USER button the third time will go back to the first state after reset

```
// Variable to track the state
uint8_t state = 0; // 0: Both LEDs blinking, 1: LED1 blinking, 2: LED2 blinking

int main(void) {
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();

    while (1) {
        switch (state) {
            case 0: // Both LEDs blinking
                HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
                HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
                HAL_Delay(500); // Delay for 500 milliseconds
                break;
            case 1: // LED1 blinking
                HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
                HAL_Delay(500); // Delay for 500 milliseconds
                break;
            case 2: // LED2 blinking
                HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
                HAL_Delay(500); // Delay for 500 milliseconds
                break;
            default:
                break;
        }
    }
}
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    if (GPIO_Pin == USER_BUTTON_Pin) {
        // Change state based on button presses
        state++;
        if (state > 2) // Reset state after reaching 3
            state = 0;
        // Wait for button release to avoid multiple state changes with one press
        while (HAL_GPIO_ReadPin(USER_BUTTON_GPIO_Port, USER_BUTTON_Pin) == GPIO_PIN_RE
        HAL_Delay(100); // Debounce delay
    }
}
```

Create a new STM32 Project to blink (on/off) with a period of 0.2 sec. for an external LED

```
HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_1);
HAL_Delay(200); //delay 200ms
```


NUCLEO-F411RE
CN7
CN6
CN5
CN10
PC10 PC12 VDD BOOT0 NC NC PA13 PA14 PA15 GND PB7 PC13 PC14 PC15 PH0 PH1 VBAT PC2 PC3
PC11 PD2 E5V GND NC IOREF RESET +3V3 +5V GND GND VIN NC PA0 PA1 PA4 PB0 PC1 PC0
NC IOREF RESET +3V3 +5V GND GND VIN
A0 A1 A2 A3 A4 A5
D15 D14 AVDD GND D13 D12 D11 D10 D9 D8
D7 D6 D5 D4 D3 D2 D1 D0
PC9 PB8 PB9 AVDD GND PA5 PA6 PA7 PB6 PC7 PA9 PA8 PB10 PB4 PB5 PB3 PA10 PA2 PA3
PC8 PC6 PC5 U5V NC PA12 PA11 PB12 NC GND PB2 PB1 PB15 PB14 PB13 AGND PC4 NC NC
CN8
CN9
Arduino Uno
ST morpho

```
ls /dev/tty.*
```

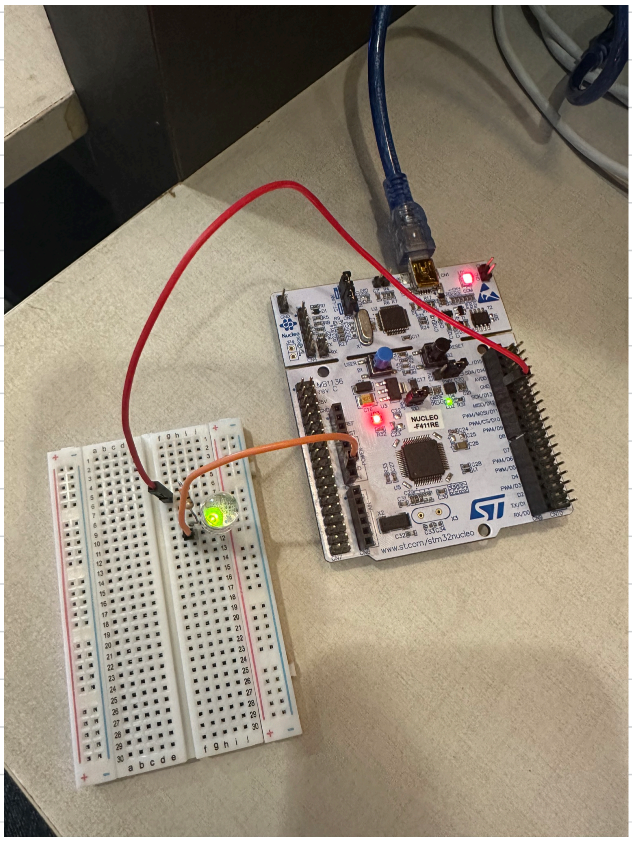
then you can read that serial port using the screen command, like this

```
screen /dev/tty.[yourSerialPortName] [yourBaudRate]
```

for example:

```
screen /dev/tty.usbserial-A6004byf 9600
```

115200



```
/* USER CODE END WHILE */
  int time = HAL_GetTick();

  // LED change
  if (state == 0) {
    if (isOn) {
      if (time - lastChange > 175) {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
        isOn = 0;
        lastChange = time;
      }
    } else {
      if (time - lastChange > 75) {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
        isOn = 1;
        lastChange = time;
      }
    }
  } else if (state == 1) {
    if (isOn) {
      if (time - lastChange > 50) {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
        isOn = 0;
        lastChange = time;
      }
    } else {
      if (time - lastChange > 50) {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
        isOn = 1;
        lastChange = time;
      }
    }
  }
```

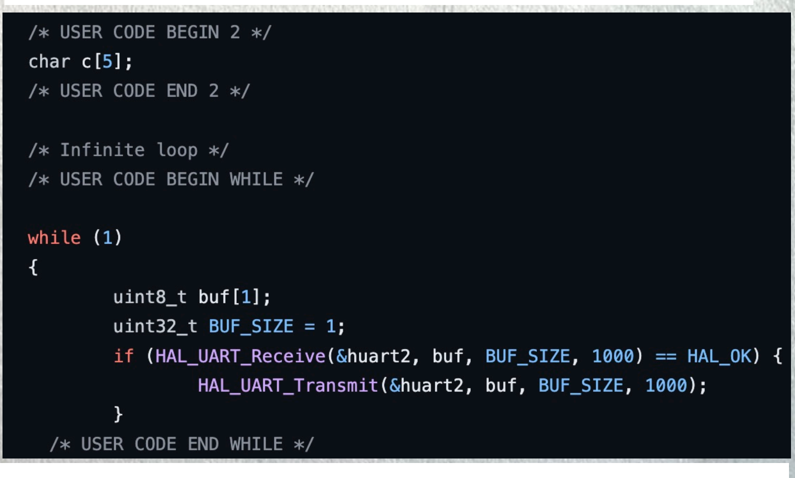
```
if (isPush == 1) {
        isPush = !HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
        if (isPush == 1) {
            // hold  1->1
            }
        } else {
            lastReleaseTime = time;
            lastStateChangeTime = time;

            // release  1->0
        }
    } else {
        isPush = !HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
        if (isPush == 1) {
            // push  0->1
            lastPushTime = time;
            lastStateChangeTime = time;
        } else {
            // idle 0->0
        }
    }
```

```
/* USER CODE BEGIN 2 */
  int state = 0;
  int isOn = 0;
  int lastChange = 0;

  int DEBOUNCE_TIME = 50;
  int lastPushTime = -1;
  int lastReleaseTime = 0;
  int lastStateChangeTime = 0;
  int isPush = 0;
  /* USER CODE END 2 */
```

```
  // button change
  if (time - lastStateChangeTime > DEBOUNCE_TIME) {
    if (isPush == 1) {
      isPush = !HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
      if (isPush == 1) {
        if (time - lastPushTime > 2000) {
          if (state != 2) {
            HAL_UART_Transmit(&huart2, "power down\r\n", strlen("power down\r\n"), HAL_MAX_DELAY);
            state = 2;
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
          }
        }
      } else {
        lastReleaseTime = time;
        lastStateChangeTime = time;
      }
    } else {
      isPush = !HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
      if (isPush == 1) {
        if (state != 2) state = (state + 1) % 2;
        lastPushTime = time;
        lastStateChangeTime = time;
      } else {
      }
    }
  }


  int main(void) {
        /* USER CODE BEGIN 1 */
        int state = 0;
        char buf[1];
        /* USER CODE END 1 */

   GIN WHILE */
   1) {
    if (HAL_UART_Receive(&huart2, buf, 1, 1000) == HAL_OK) {
          if (state == 0) {
                  if (buf[0] == 'o')
                          state = 1;
          } else if (state == 1) {
                  if (buf[0] == 'n')
                          state = 2;
                  else if (buf[0] == 'f')
                          state = 3;
                  else
                          state = 0;
          } else if (state == 2) {
                  if (buf[0] == 13 || buf[0] == 10)
                          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
                  state = 0;
          } else if (state == 3) {
                  if (buf[0] == 'f')
                          state = 4;
                  else
                          state = 0;
          } else if (state == 4) {
                  if (buf[0] == 13 || buf[0] == 10)
                          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
                  state = 0;
          }
          HAL_UART_Transmit(&huart2, buf, 1, 1000);
    }

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
```

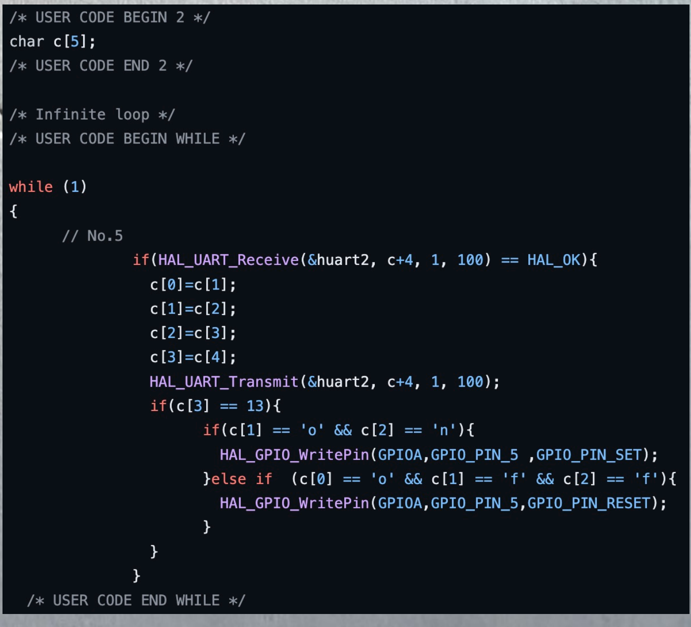
4 Create a new STM32 Project to echo back (transmit the receive data) the communication data from UART peripheral (USART2) interface using blocking mode APIs on STM32CUBE library (Look at stm32f4xx_hal_uart.c).

For those of you who are using STM32F4Discovery, you will have to connect to a USB-Serial such as ET-MINI USB-TTL as mention above.



```
/* USER CODE BEGIN 2 */
char c[5];
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */

while (1)
{
        uint8_t buf[1];
        uint32_t BUF_SIZE = 1;
        if (HAL_UART_Receive(&huart2, buf, BUF_SIZE, 1000) == HAL_OK) {
                HAL_UART_Transmit(&huart2, buf, BUF_SIZE, 1000);
        }
   /* USER CODE END WHILE */
```

5 Create a new STM32 Project to toggle an LED status (on/off) with commands via serial console. (Type "on" or "off" then press Enter to on or off the LED



```
/* USER CODE BEGIN 2 */
char c[5];
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */

while (1)
{
      // No.5
            if(HAL_UART_Receive(&huart2, c+4, 1, 100) == HAL_OK){
              c[0]=c[1];
              c[1]=c[2];
              c[2]=c[3];
              c[3]=c[4];
              HAL_UART_Transmit(&huart2, c+4, 1, 100);
              if(c[3] == 13){
                   if(c[1] == 'o' && c[2] == 'n'){
                     HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5 ,GPIO_PIN_SET);
                   }else if  (c[0] == 'o' && c[1] == 'f' && c[2] == 'f'){
                     HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,GPIO_PIN_RESET);
                   }
              }
            }
  /* USER CODE END WHILE */
```