

Programmazione di sistemi multicore

Michele Martinelli

Michele.martinelli@uniroma1.it



W • S E N S E

AVVISI

19 FREERTOS teoria e esercizi

21 FREERTOS teoria

26 MPI

28 MPI

3 CUDA

5 CUDA

10 CUDA

12 CUDA esercizi

17 applicazioni BC e esercizi

19 applicazioni droni / secondo esonero?

PORTE E REGISTRI

PORTD – The Port D Data Register

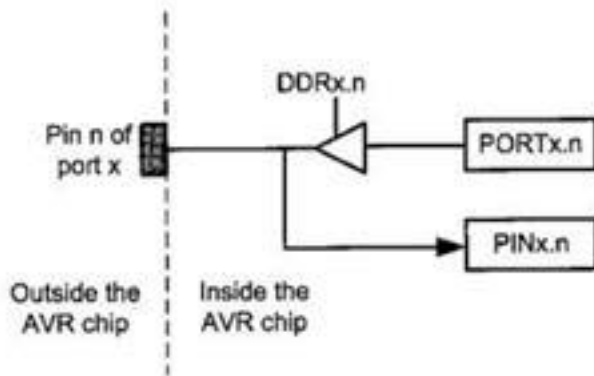
Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DDRD – The Port D Data Direction Register

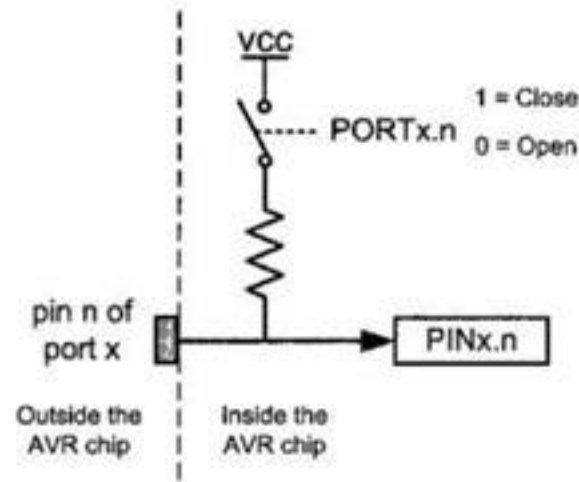
Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PIND – The Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	



I/O Port in AVR microcontrollers



Pull-up Resistor

DETTAGLI SUI REGISTRI

- DDRx
 - Data Direction bit in DDRx register (read/write)
- PORTxn
 - PORTxn bit in PORTx data register (read/write)
- PINxn
 - PINxn bit in PINx register (read only)

ACCESSO DIRETTO AI REGISTRI

Accedere direttamente ai registri del microcontrollore ha degli svantaggi

- difficoltà di manutenzione del codice
- perdita di portabilità
- errori

Ma anche dei vantaggi

- facilità/velocità di accesso
- Alcune volte serve di impostare diversi pin nello stesso momento
PORTB |= B1100;
(digitalRead() and digitalWrite() sono composte a molte righe di codice)

ESEMPIO: DIGITALWRITE

```
void digitalWrite(uint8_t pin, uint8_t val) {
    uint8_t timer, bit, port, oldSREG;
    volatile uint8_t *out;

    //timer = digitalPinToTimer(pin);
    timer = pgm_read_byte(digital_pin_to_timer_PGM + pin );
    //bit = digitalPinToBitMask(pin);
    bit = pgm_read_byte( digital_pin_to_bit_mask_PGM + pin );
    //port = digitalPinToPort(pin);
    port = pgm_read_byte( digital_pin_to_port_PGM + pin );

    if (port == NOT_A_PIN)
        return;

    //If the pin that support PWM output, we need to turn it off
    //before doing a digital write.
    if (timer != NOT_ON_TIMER)
        turnOffPWM(timer);

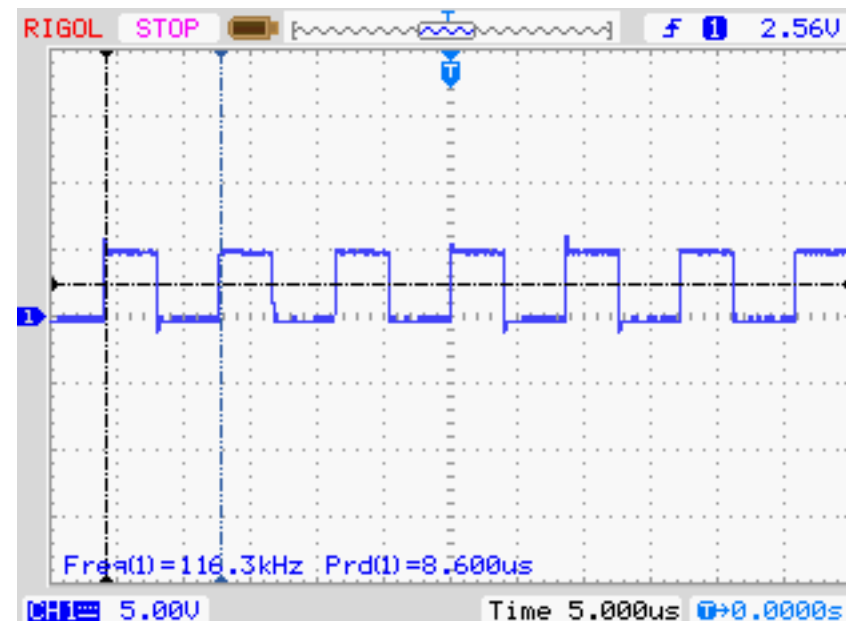
    //out = portOutputRegister(port);
    out =(volatile uint8_t *) (pgm_read_word( port_to_output_PGM + pin ));

    oldSREG = SREG;
    cli();

    if (val == LOW)
        *out &= bit; //clear bit
    else
        *out |= bit; //set bit

    SREG = oldSREG;
}
```

Risultato



...e port manipulation

```
PORTB =0;
```

```
PORTB =B100000;
```

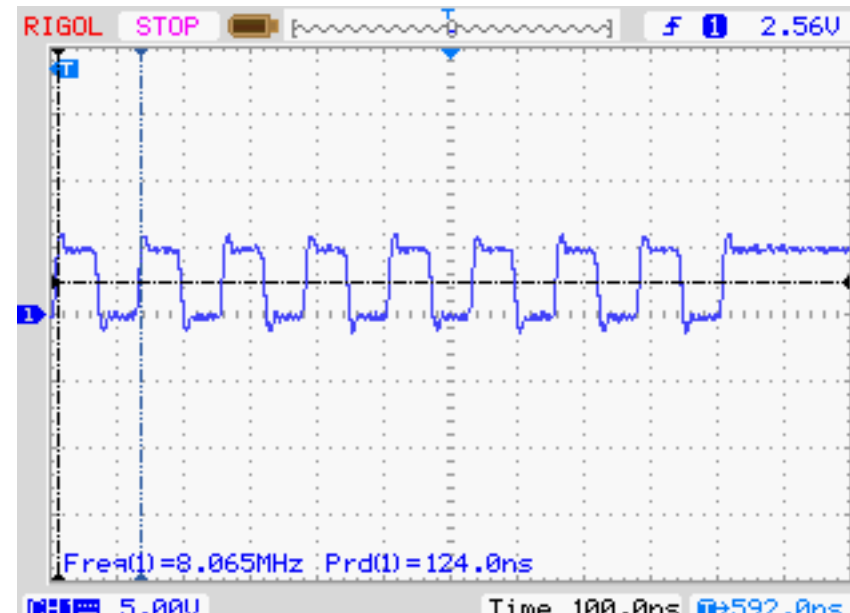
Risultato

Importante: inizializzazione delle variabili

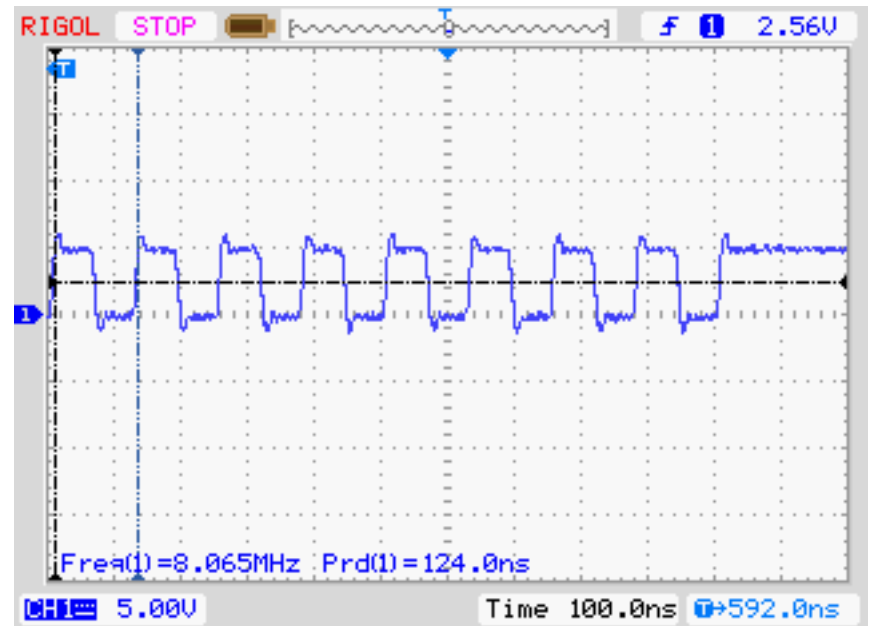
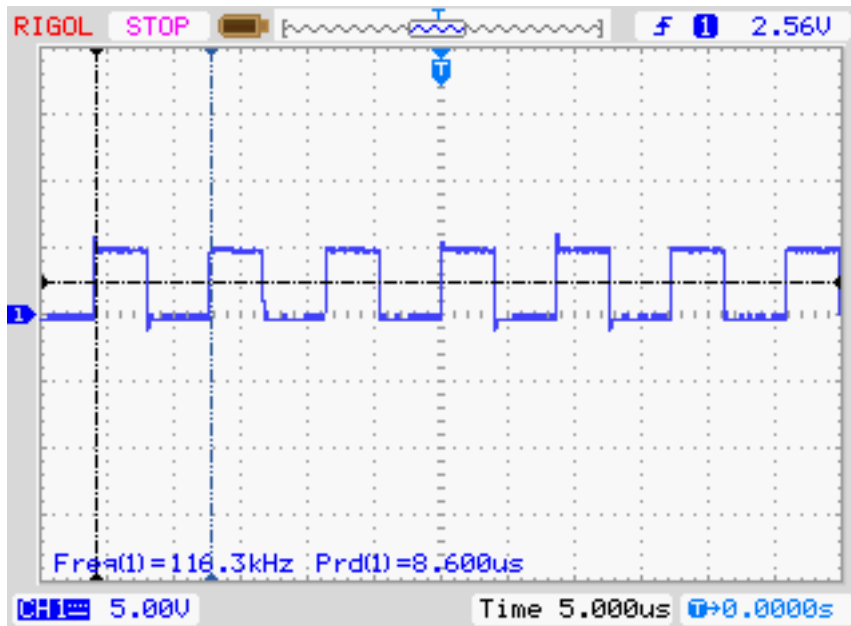
A = B00000101 -> B binario

A = 0x05 -> 0x esadecimale

A = 5 -> decimale

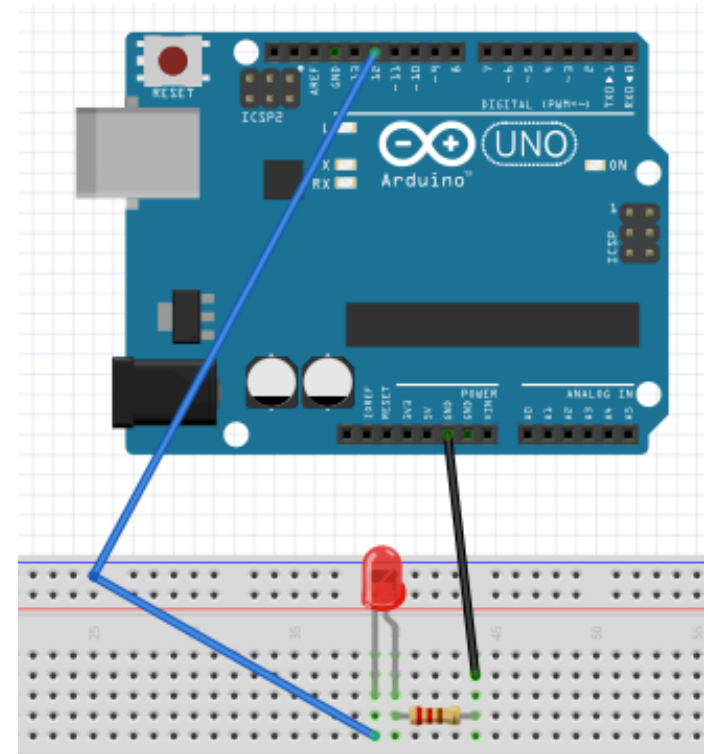


Confrontando...



ACCENDERE UN LED CON PORT MANIPULATION

```
void setup(){  
  pinMode(7, OUTPUT);  
  pinMode(6, OUTPUT);  
  pinMode(5, OUTPUT);  
}  
  
void loop(){  
  //digitalWrite(7, HIGH);  
  //digitalWrite(6, HIGH);  
  //digitalWrite(5, HIGH);  
  PORTD = PORTD | B11100000;  
  delay(1000);  
  |  
  //digitalWrite(7, LOW);  
  //digitalWrite(6, LOW);  
  //digitalWrite(5, LOW);  
  PORTD = PORTD & B00011111;  
  delay(1000);  
}
```



MASCHERE BITWISE – ACCENDERE UN BIT

Accendere uno specifico bit di una stringa (o il pin di una porta)

Voglio ottenere xxxxx11 (accendere gli ultimi due bit)

Esempi:

Stringa : 00000000

00000000 | 00000011 -> 00000011

Stringa: 11111111

11111111 | 00000011 -> 11111111

Stringa: 10101011

10101011 | 00000011 -> 10101011

MASCHERE BITWISE – SPEGNERE UN BIT

Accendere uno specifico bit di una stringa (o il pin di una porta)

Voglio ottenere xxxxxx00 (spegnere gli ultimi due bit)

Stringa : 11111111

11111111 & 11111100 -> 11111100

Stringa: 00000011

00000011 & 11111100 -> 00000000

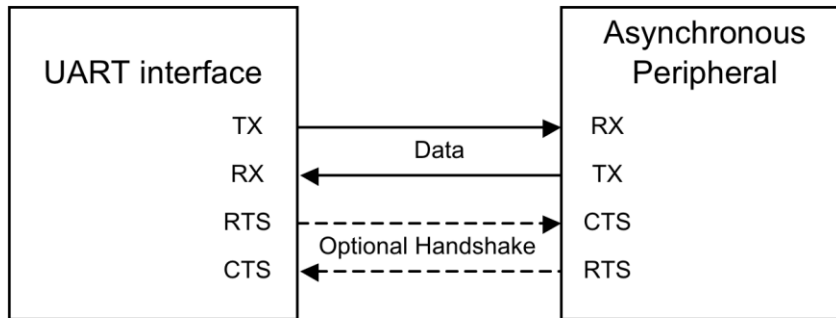
Stringa: 01010111

01010111 & 11111100 -> 01010100

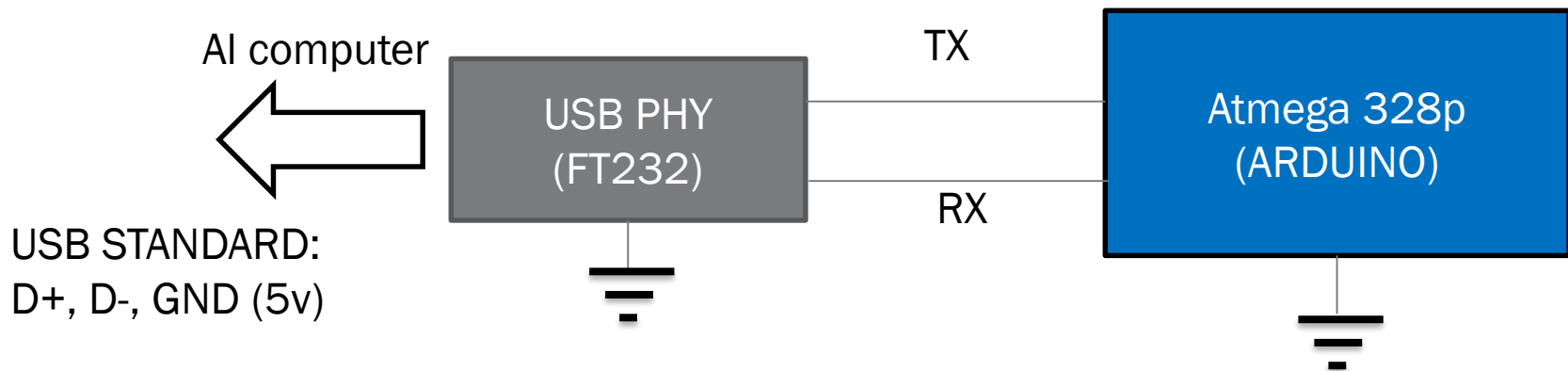
COMUNICAZIONE SERIALE

ASYNCHRONOUS SERIAL INTERFACE (UART)

Questo è quello che avete usato finora per «parlare» con Arduino



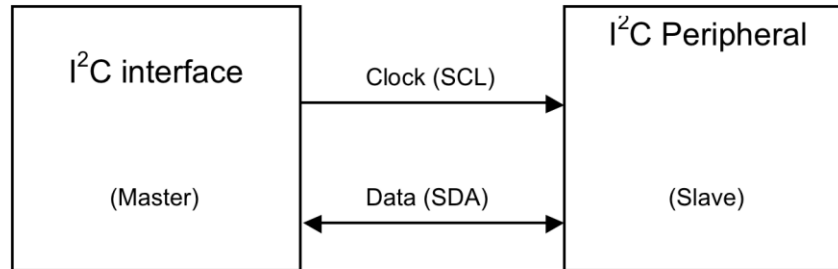
Molto lento e poco robusto, ma molto semplice da implementare e utilizzare



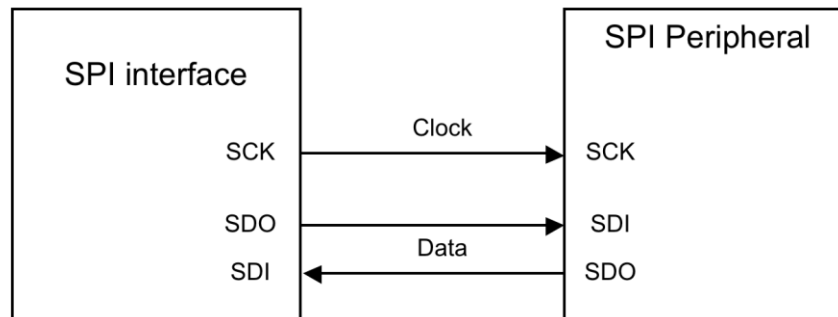
SYNCHRONOUS SERIAL INTERFACES

utilizzati per parlare con periferiche esterne

I²C



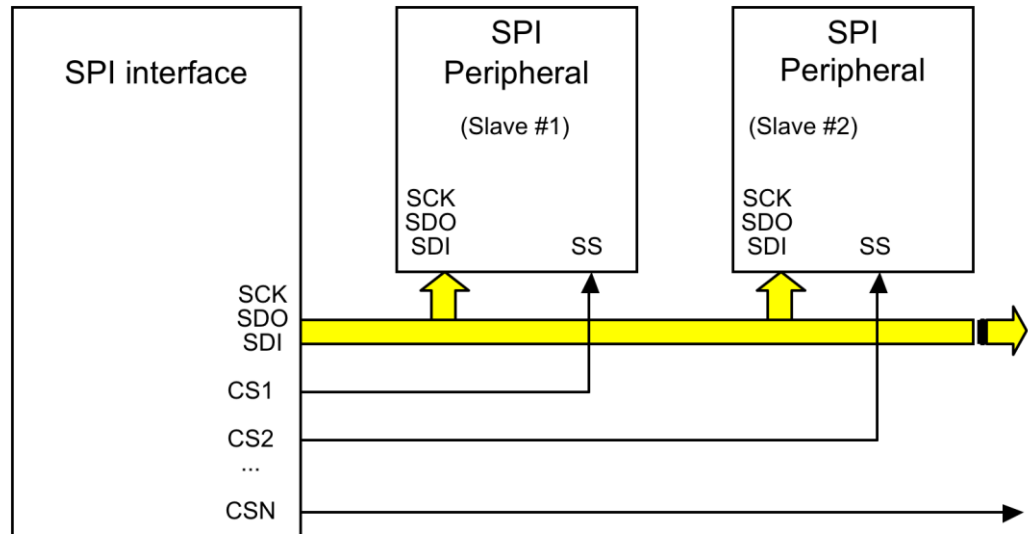
SPI



SPI BUS

Composto da:

- CLOCK
- SLAVE DATA IN
- SLAVE DATA OUT
- SLAVE SELECT



Vantaggi

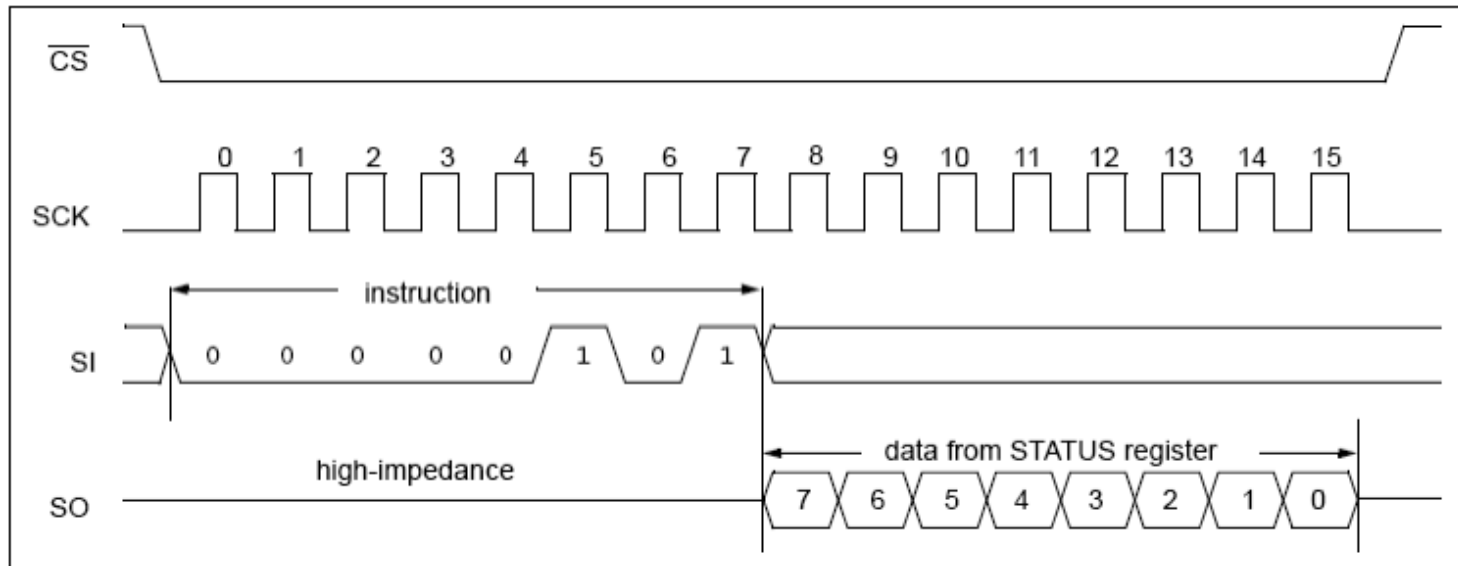
- Bidirezionale
- Solitamente non richiedono resistenze di pullup

Svantaggi

- Una linea (filo, pista) necessaria per ogni slave

ESEMPIO: 25LC256 SERIAL EEPROM

```
// 25LC256 Serial EEPROM commands
#define CMD_WRSR    1    // write status register
#define CMD_WRITE  2    // write command
#define CMD_READ   3    // read command
#define CMD_WDI    4    // write disable
#define CMD_STAT   5    // read status register
#define CMD_WEN    6    // write enable
```



WRITING TO THE EEPROM

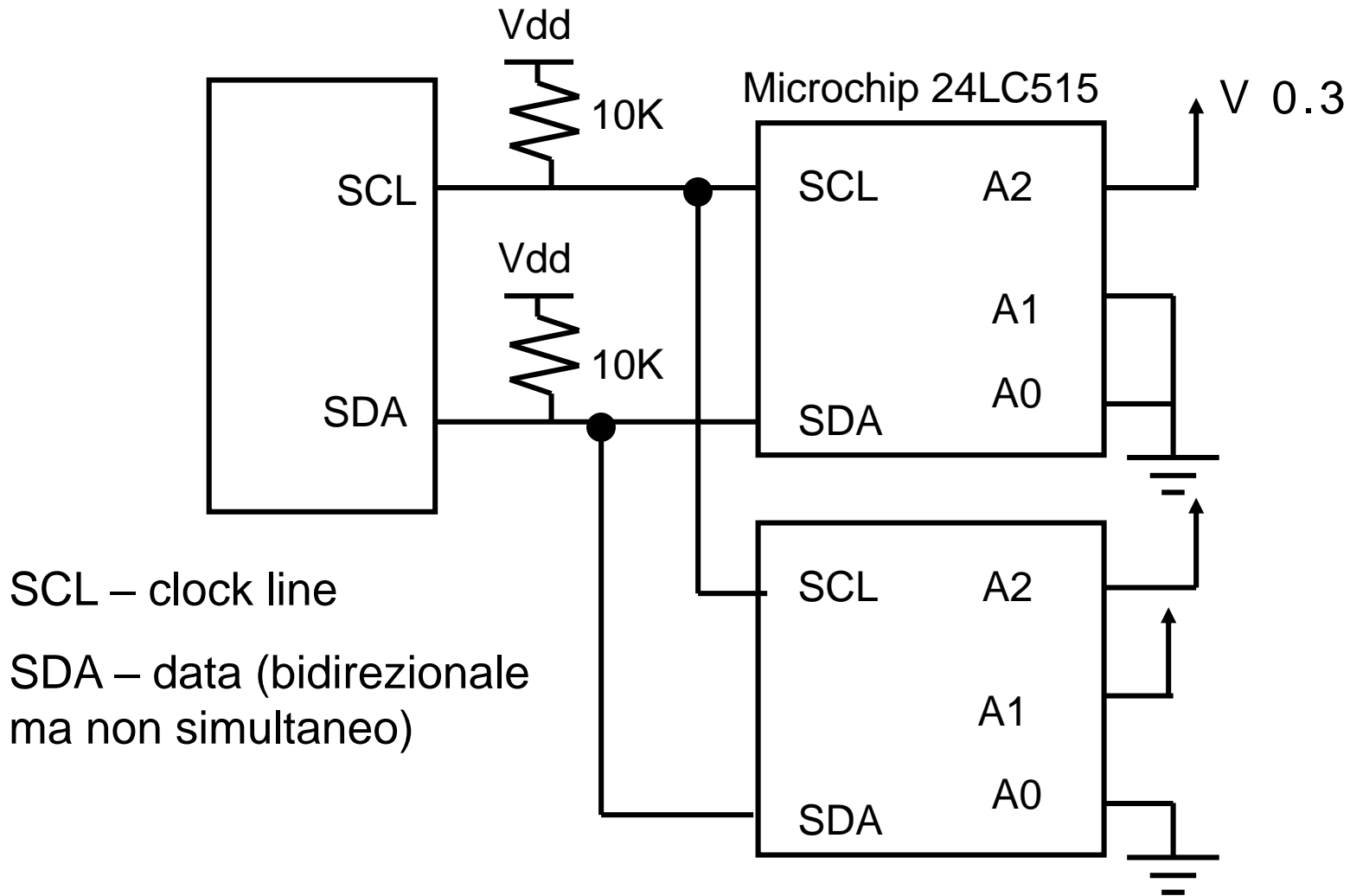
```
// send a write command
  CSEE = 0;           // select the Serial EEPROM
  WriteSPI2( CMD_WRITE); // write command
  WriteSPI2( addr_MSB); // address MSB first
  WriteSPI2( addr_LSB); // address LSB (word aligned)
  WriteSPI2( data);   // send 8-bit of data
  // continue writing more data..
  CSEE = 1;
// wait until any work in progress is completed
while ( ReadSR() & 0x1); // check the WIP flag
```

READING FROM THE EEPROM

```
// perform a read sequence
  CSEE = 0;           // select the Serial EEPROM
  WriteSPI2( CMD_READ); // read command
  WriteSPI2( addr_MSB); // address MSB first
  WriteSPI2( addr_LSB); // address LSB (word aligned)
  data = WriteSPI2( 0); // send dummy, read msb
  // continue reading a second byte, a third..
  CSEE = 1;
```


I²C (INTER-INTEGRATED-CIRCUIT) BUS

I2C is a two wire serial interface.



I²C FEATURES

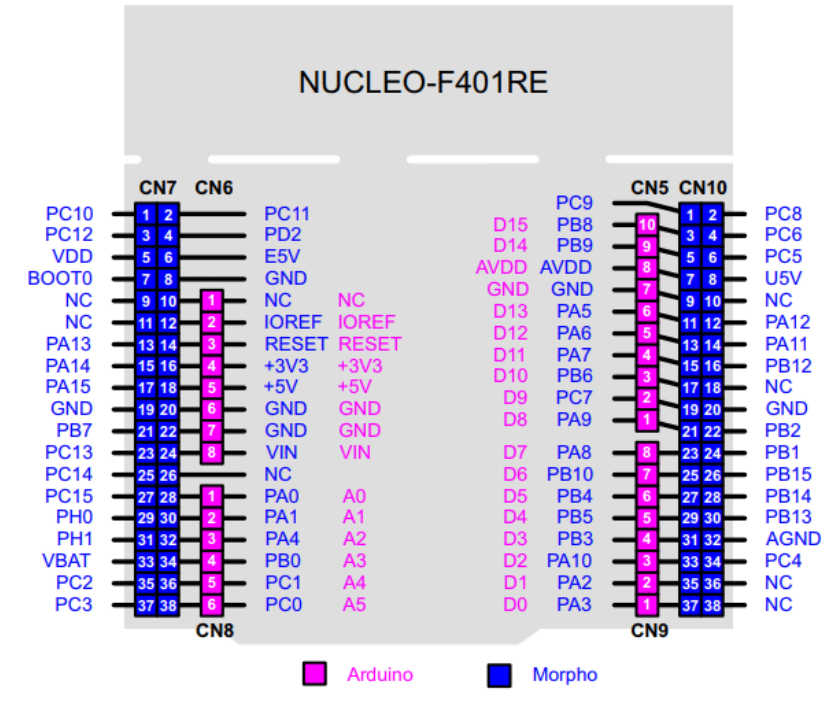
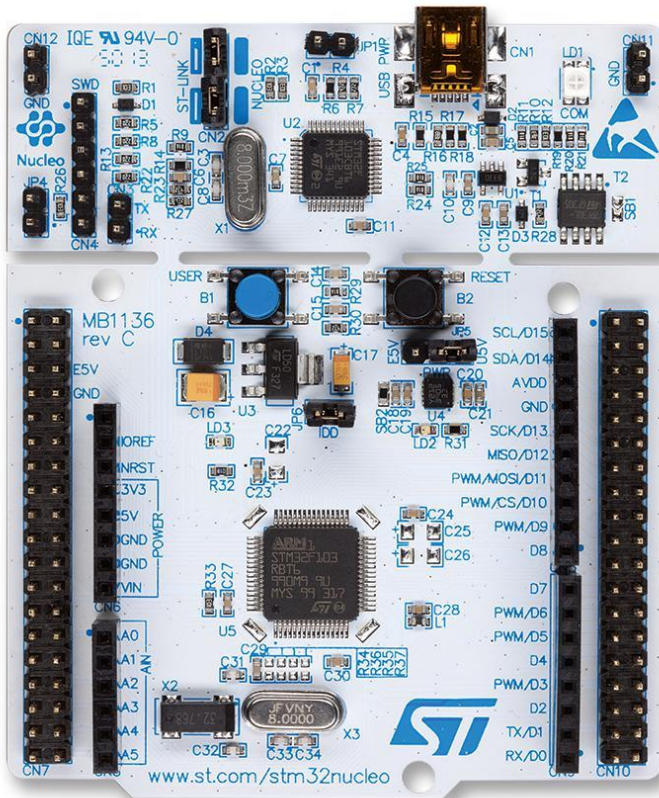
Una singola linea richiesta anche in presenza di molti slave

- Ogni transazione inizia con l'indirizzo del ricevente
- Tutti I ricevitori ascoltano e confrontano con il proprio indirizzo

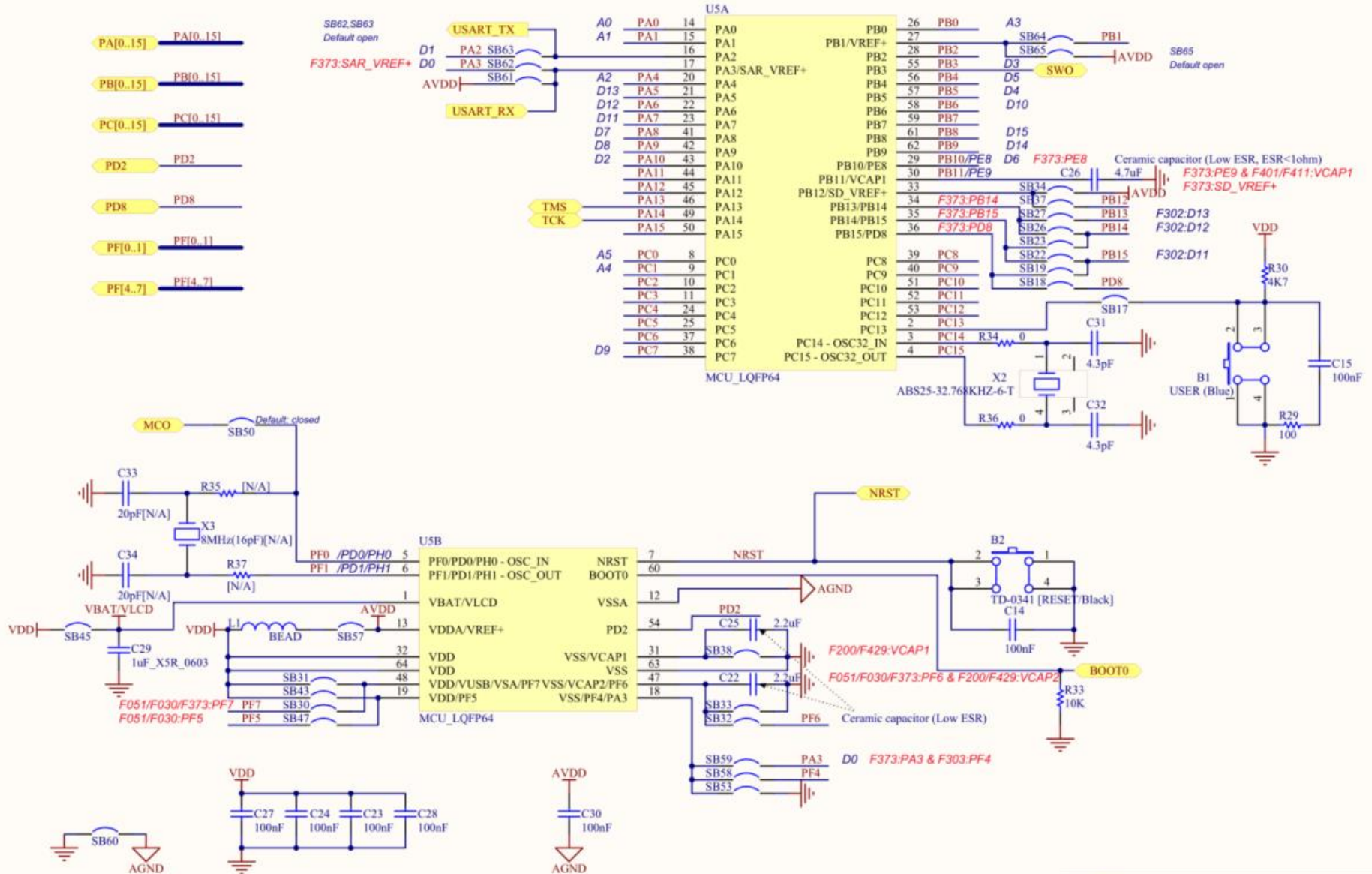
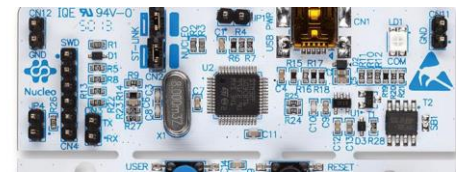
Vengono utilizzate due linee:

- SDA (data)
- SCL (clock)
- Bidirezionale ma non full-duplex (o trasmetti o ricevi)

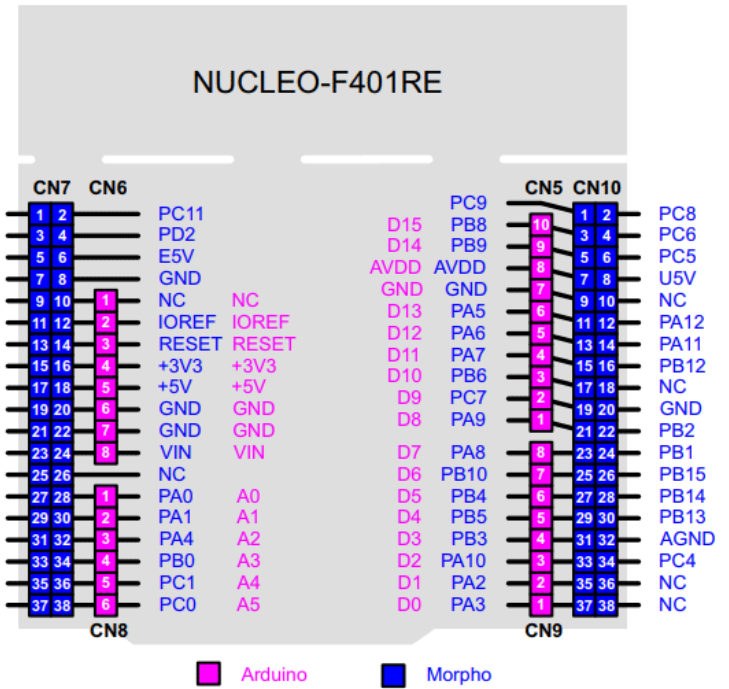
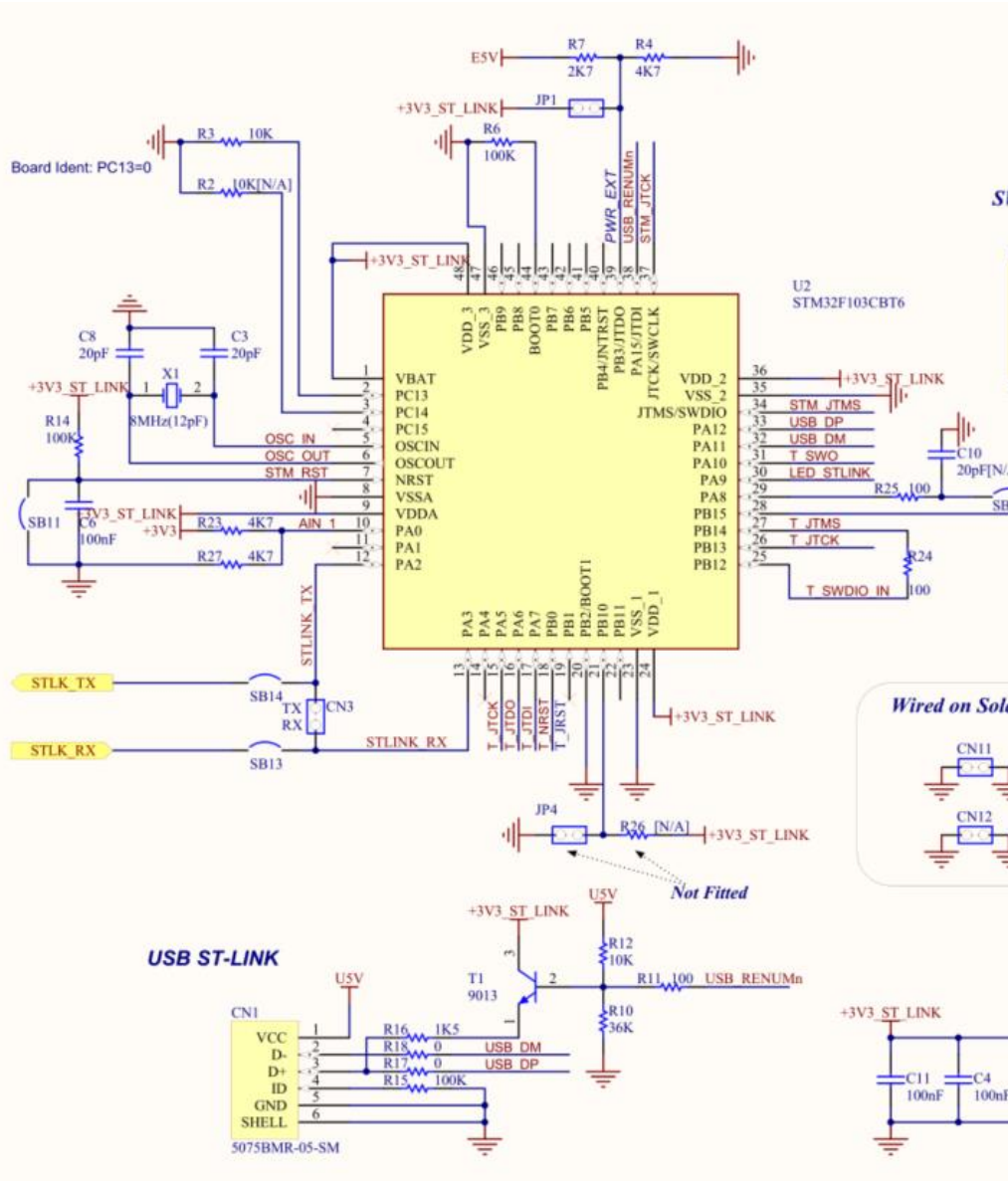
NUCLEO F401RE - HW



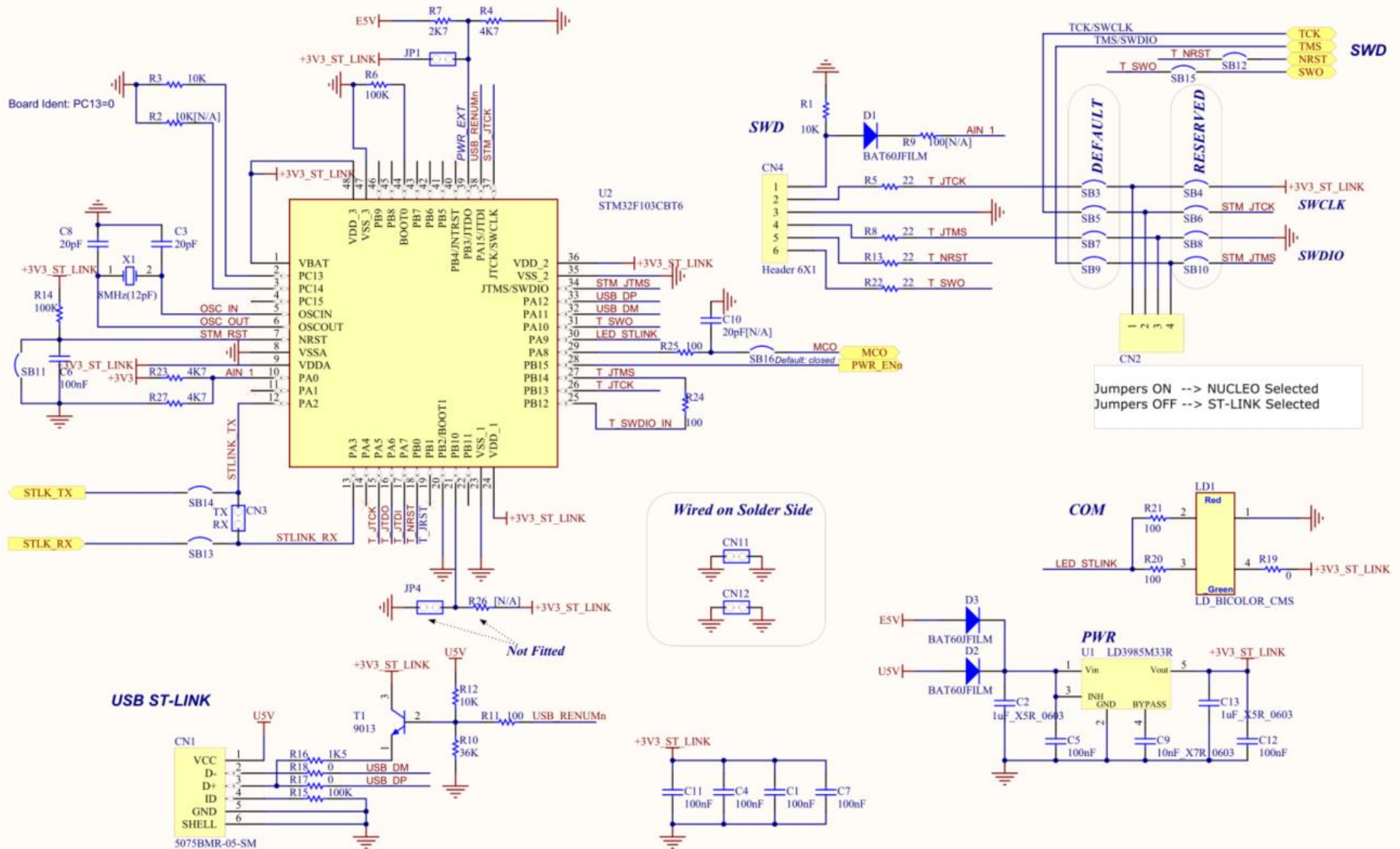
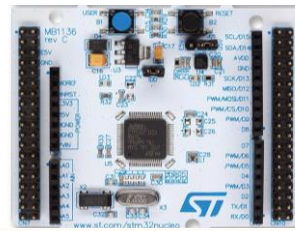
NUCLEO F401RE - SCHEMA



NUCLEO F401RE - SCHEMA



NUCLEO F401RE - SCHEMA



NUCLEO F401RE – ESEMPIO

BLINK LED INTEGRATO

File -> new -> project -> STM32project -> board selector -> nucleo F401RE

The screenshot shows the STM32 Project Board Selector interface. The window title is "STM32 Project". The main heading is "Target Selection" with the instruction "Select STM32 target". The interface is divided into several sections:

- Board Filters:** Includes a search bar with "NUCLEO-F401RE" entered, and various filter options like Vendor, Type, MCU/MPU Series, and Other. A price filter is set to 13.0 and an oscillator frequency filter is set to 0 MHz.
- Board Details:** Displays the selected board, "NUCLEO-F401RE", with a "Features" tab. It includes a "Large Picture" view, "Docs & Resources", "Datasheet", and "Buy" options. The board is marked as "ACTIVE" and "Product is in mass production". The unit price is listed as 13.0 (US\$) and the mounted device is "STM32F401RETx".
- Boards List:** A table showing the selected board as the only item in the list.

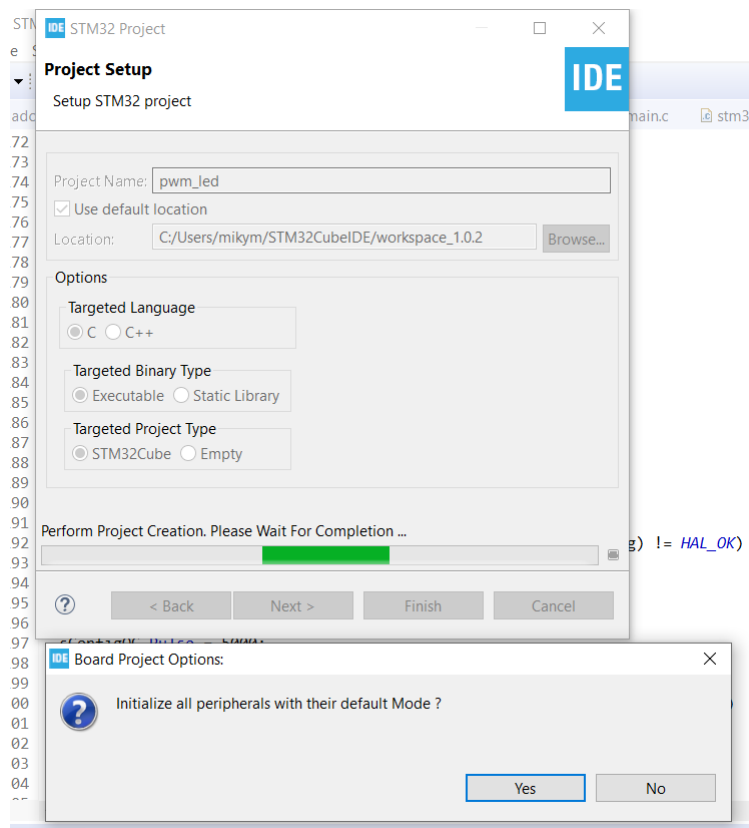
*	Overview	Part No	Type	Marketing Status	Unit Price (US\$)	Mounted Device
☆		NUCLEO-F401RE	Nucleo64	Active	13.0	STM32F401RETx

At the bottom of the window, there are navigation buttons: "< Back", "Next >", "Finish", and "Cancel".

NUCLEO F401RE – ESEMPIO

BLINK LED INTEGRATO

Assegnare nome – e poi yes



NUCLEO F401RE – ESEMPIO

BLINK LED INTEGRATO

```
HAL_GPIO_TogglePin(LD2_GPIO_Port,LD2_Pin); //Toggle LED
```

```
HAL_Delay(1000); //Delay 1 Seconds
```

NUCLEO F401RE – ESEMPIO

BLINK LED INTEGRATO

Programmare il target

The image shows a screenshot of an IDE with a C program for a Nucleo F401RE target and its configuration dialog. The code in the main window is as follows:

```
70 /*
71 int main(void)
72 {
73 /* USER CODE BEGIN 1 */
74
75 /* USER CODE END 1 */
76
77 /* MCU Configuration-----
78
79 /* Reset of all peripherals, Initializes the Flash interface
80 HAL_Init();
81
82 /* USER CODE BEGIN Init */
83
84 /* USER CODE END Init */
85
86 /* Configure the system clock */
87 SystemClock_Config();
88
89 /* USER CODE BEGIN SysInit */
90
91 /* USER CODE END SysInit */
92
93 /* Initialize all configured peripherals */
94 MX_GPIO_Init();
95 MX_USART2_UART_Init();
96 MX_TIM2_Init();
97 /* USER CODE BEGIN 2 */
98
99 /* USER CODE END 2 */
100 //HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
101 /* Infinite loop */
102 /* USER CODE BEGIN WHILE */
103 while (1)
104 {
105 /* USER CODE END WHILE */
106
107 /* USER CODE BEGIN 3 */
108
109 }
110 /* USER CODE END 3 */
111 }
112
113 /**
114 * @brief System Clock Configuration
115 * @retval None
116 */
117 void SystemClock_Config(void)
118 {
119 RCC_OscInitTypeDef RCC_OscInitStruct = {0};
120 RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
```

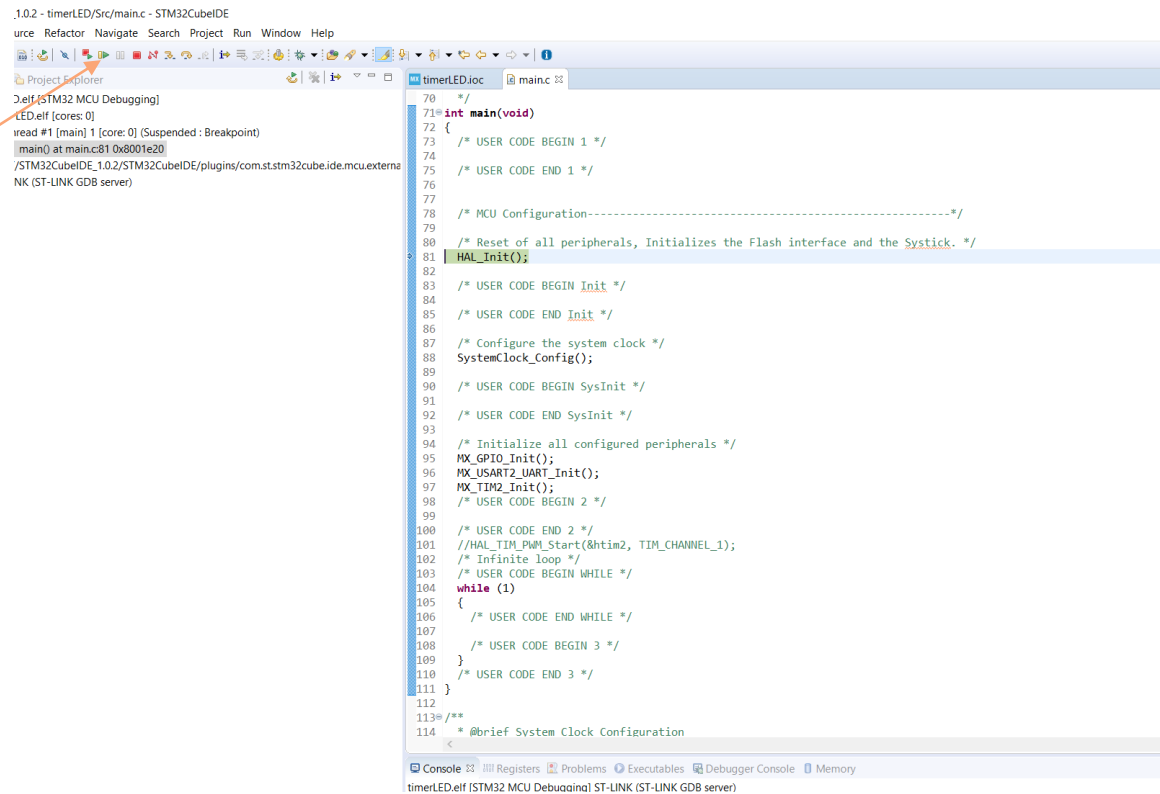
The configuration dialog, titled "Edit Configuration", shows the following settings:

- Name: timerLED.elf
- Debugger: Main
- C/C++ Application: Debug/timerLED.elf
- Project: timerLED
- Build Configuration: Use Active
- Build options: Use workspace settings
- Buttons: Revert, Apply, OK, Cancel

NUCLEO F401RE – ESEMPIO

BLINK LED INTEGRATO

Avviare il target



The screenshot shows the STM32CubeIDE interface. On the left, the Project Explorer displays the file structure, with 'main.c' selected under 'timerLED\Src'. The main editor window shows the code for 'main.c'. An orange arrow points from the 'Run' button in the top toolbar to the 'main.c' file in the Project Explorer. The code in the editor is as follows:

```
1.0.2 - timerLED\Src\main.c - STM32CubeIDE
urce Refactor Navigate Search Project Run Window Help
Project Explorer
D:\elf [STM32 MCU Debugging]
LED.elf [cores: 0]
read #1 [main] 1 [core: 0] (Suspended : Breakpoint)
main() at main.c:1 0x8001e20
/STM32CubeIDE_1.0.2/STM32CubeIDE/plugins/com.stm32cube.ide.mcu.external
NK (ST-LINK GDB server)

70 */
71 int main(void)
72 {
73 /* USER CODE BEGIN 1 */
74
75 /* USER CODE END 1 */
76
77
78 /* MCU Configuration-----*/
79
80 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
81 HAL_Init();
82
83 /* USER CODE BEGIN Init */
84
85 /* USER CODE END Init */
86
87 /* Configure the system clock */
88 SystemClock_Config();
89
90 /* USER CODE BEGIN SysInit */
91
92 /* USER CODE END SysInit */
93
94 /* Initialize all configured peripherals */
95 MX_GPIO_Init();
96 MX_USART2_UART_Init();
97 MX_TIM2_Init();
98 /* USER CODE BEGIN 2 */
99
100 /* USER CODE END 2 */
101 //HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
102 /* Infinite loop */
103 /* USER CODE BEGIN WHILE */
104 while (1)
105 {
106 /* USER CODE END WHILE */
107
108 /* USER CODE BEGIN 3 */
109 }
110 /* USER CODE END 3 */
111 }
112
113 /**
114 * @brief System Clock Configuration
```

NUCLEO F401RE – ESEMPIO

USIAMO IL PULSANTE PER ACCENDERE E SPEGNERE IL LED INTEGRATO

```
if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13)) {  
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);  
} else {  
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);  
}  
  
osDelay(100);
```

NUCLEO F401RE – ESEMPIO

AGGIUNGERE UN TASK A PARTE PER LA SERIALE

```
char *msg = "Hello from Nucleo!\n\r";
```

```
HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), 0xFFFF);  
osDelay(100);
```

NUCLEO F401RE – ESEMPIO

RICEVERE DALLA SERIALE

```
HAL_StatusTypeDef HAL_UART_Receive(UART_HandleTypeDef *huart, uint8_t *pData,  
                                   uint16_t Size, uint32_t Timeout);
```

where:

- `huart`: it is the pointer to an instance of the struct `UART_HandleTypeDef` seen before, which identifies and configures the UART peripheral;
- `pData`: is the pointer to an array, with a length at least equal to the `Size` parameter, containing the sequence of bytes we are going to receive. The function will block until all bytes specified by the `Size` parameter are received.
- `Timeout`: is the maximum time, expressed in milliseconds, we are going to wait for the receive completion. If the transmission does not complete in the specified timeout time, the function aborts and returns the `HAL_TIMEOUT` value; otherwise it returns the `HAL_OK` value if no other errors occur. Moreover, we can pass a timeout equal to `HAL_MAX_DELAY (0xFFFF FFFF)` to wait indefinitely for the receive completion.

NUCLEO F401RE – ESEMPIO

RICEVERE DALLA SERIALE

```
34     printMessage:
35
36         printWelcomeMessage();
37
38     while (1) {
39         opt = readUserInput();
40         processUserInput(opt);
41         if(opt == 3)
42             goto printMessage;
43     }
44 }
45
46 void printWelcomeMessage(void) {
47     HAL_UART_Transmit(&huart2, (uint8_t*)"\033[0;0H", strlen("\033[0;0H"), HAL_MAX_DELAY);
48     HAL_UART_Transmit(&huart2, (uint8_t*)"\033[2J", strlen("\033[2J"), HAL_MAX_DELAY);
49     HAL_UART_Transmit(&huart2, (uint8_t*)WELCOME_MSG, strlen(WELCOME_MSG), HAL_MAX_DELAY);
50     HAL_UART_Transmit(&huart2, (uint8_t*)MAIN_MENU, strlen(MAIN_MENU), HAL_MAX_DELAY);
51 }
52
53 uint8_t readUserInput(void) {
54     char readBuf[1];
55
56     HAL_UART_Transmit(&huart2, (uint8_t*)PROMPT, strlen(PROMPT), HAL_MAX_DELAY);
57     HAL_UART_Receive(&huart2, (uint8_t*)readBuf, 1, HAL_MAX_DELAY);
58     return atoi(readBuf);
59 }
60
61
62 uint8_t processUserInput(uint8_t opt) {
63     char msg[30];
64
65     if(!opt || opt > 3)
66         return 0;
67
68     sprintf(msg, "%d", opt);
69     HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
70
71     switch(opt) {
72     case 1:
73         HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
74         break;
75     case 2:
76         sprintf(msg, "\r\nUSER BUTTON status: %s", HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == GPIO_PIN_RESET ? "PRESSED" : "RELEASED");
77         HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
78         break;
79     case 3:
80         return 2;
81     };
82
83     return 1;
84 }
85
86 --
```

NUCLEO F401RE – ESEMPIO

USIAMO IL TIMER PER FAR BLINKARE IL LED INTEGRATO

The screenshot displays the STM32CubeIDE interface for configuring a timer on a Nucleo F401RE. The main window is titled "workspace_1.0.2 - Device Configuration Tool - STM32CubeIDE". The "Project Explorer" on the left shows the project structure, including the "pwm_led" folder and the "pwm_led.ioc" file. The central panel is divided into "TIM2 Mode and Configuration" and "Configuration" sections. The "Mode" section shows the following settings:

- Slave Mode: Disable
- Trigger Source: Disable
- Clock Source: Disable
- Channel1: Disable
- Channel2: Disable
- Channel3: Disable
- Channel4: Disable
- Combined Channels: Disable
- Use ETR as Cleaning Source:
- XOR activation:
- One Pulse Mode:

The "Configuration" section is currently empty. The "Pinout view" on the right shows the microcontroller pinout, with a yellow box highlighting the PA5 pin. A red arrow points from the console to the PA5 pin. The console at the bottom shows the following output:

```
<terminated> adc1.elf [STM32 MCU Debugging] ST-LINK (ST-LINK GDB server)
Target is not responding, retrying...
Target is not responding, retrying...
```


NUCLEO F401RE – ESEMPIO

USIAMO IL TIMER PER FAR BLINKARE IL LED INTEGRATO

The screenshot displays the STM32CubeMX configuration interface. On the left, the 'Timers' category is expanded to show 'TIM2' selected. The main configuration area is titled 'TIM2 Mode and Configuration'. Under the 'Mode' section, 'Channel4' is selected and set to 'Pulse Generation CH1'. The 'Configuration' section below shows 'Parameter Settings' expanded, with 'Counter Settings' further detailed: 'Counter Mode' is set to 'Up', 'Counter Prescaler' is '999999', and 'Pulse (20-bit val...)' is '500000'. The 'PWM Generation Channel' settings are also visible, with 'Mode' set to 'PWM mode 1'. On the right, the 'Pinout view' shows the physical layout of the Nucleo F401RE board. Red arrows point from the software settings to the board: one from 'Channel4' to pin PA4, one from 'Counter Mode' to pin PA0, and one from 'Pulse (20-bit val...)' to pin PA4. The board diagram labels various pins including VBAT, VSS, VDD, VREF, PA0-PA15, PB0-PB15, PC0-PC7, and USART pins.

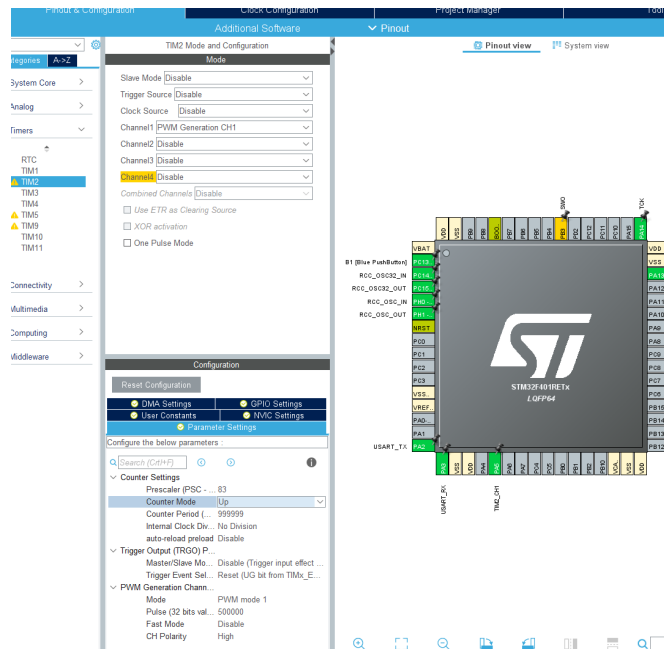
NUCLEO F401RE – ESEMPIO

USIAMO IL TIMER PER FAR BLINKARE IL LED INTEGRATO

Nel main.c inserire:

```
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
```

Private a cambiare period (e pulse in accordo)



NUCLEO F401RE – ESEMPIO

LEGGIAMO IL VALORE DI TENSIONE DA UN POTENZIOMETRO

Adc: canale 0 e 84 cicli a conversione

The screenshot displays the STM32CubeIDE interface for configuring the ADC1. The 'ADC Mode and Configuration' window is open, showing the 'Mode' section with IN0, IN2, IN3, and IN5 selected. The 'Configuration' section is expanded to 'Parameter Settings', where the 'Sampling Time' is set to 84 Cycles. A red arrow points from this setting to the 'ADC_IN0' pin on the STM32F401RE microcontroller pinout diagram on the right. The pinout diagram shows various pins including VBAT, VDD, VSS, PA0-PA13, PB0-PB15, PC0-PC15, PD0-PD15, PE0-PE7, PF0-PF7, PG0-PG7, PH0-PH7, PI0-PI7, and PD0-PD15. The microcontroller is labeled 'STM32F401RETx LQFP64'.

NUCLEO F401RE – ESEMPIO

LEGGIAMO IL VALORE DI TENSIONE DA UN POTENZIOMETRO

Adc: canale 0 e 84 cicli a conversione

Inizializza l'ADC

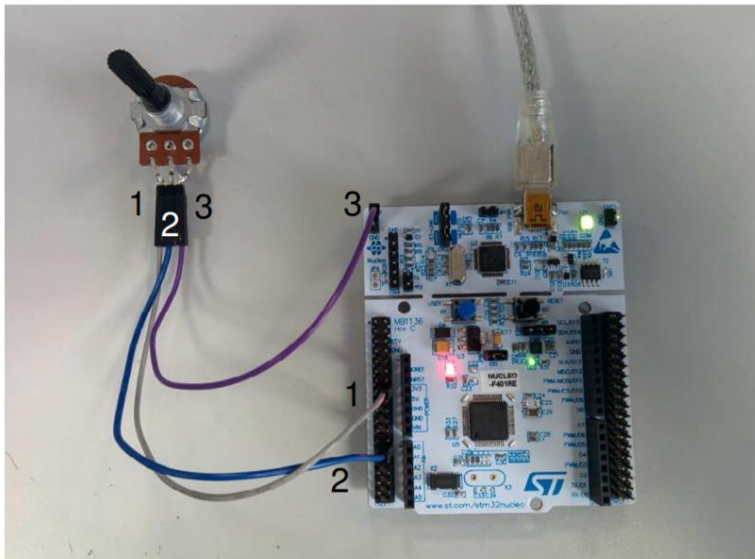
Aspetta la conversione del valore

Utilizza il valore per il PWM

Scrive le informazioni sulla seriale

```
406     for(;;)
407     {
408
409         HAL_ADC_Start(&hadc1);
410
411         HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
412
413         float rawValue = HAL_ADC_GetValue(&hadc1);
414         rawValue = ((float)rawValue) / 4095 * 10000;
415
416         htim2.Instance->CCR1 = (uint16_t) rawValue+100;
417
418         char msg[10];
419         sprintf(msg,"%d\n", (uint16_t)rawValue);
420         HAL_UART_Transmit(&huart2,msg, strlen(msg), 0xFFFF);
421     }
422 }
```

1. 3.3V => Pin 16 CN7
2. AIN0 => Pin 28 CN7
3. Ground



NUCLEO F401RE – ESEMPIO

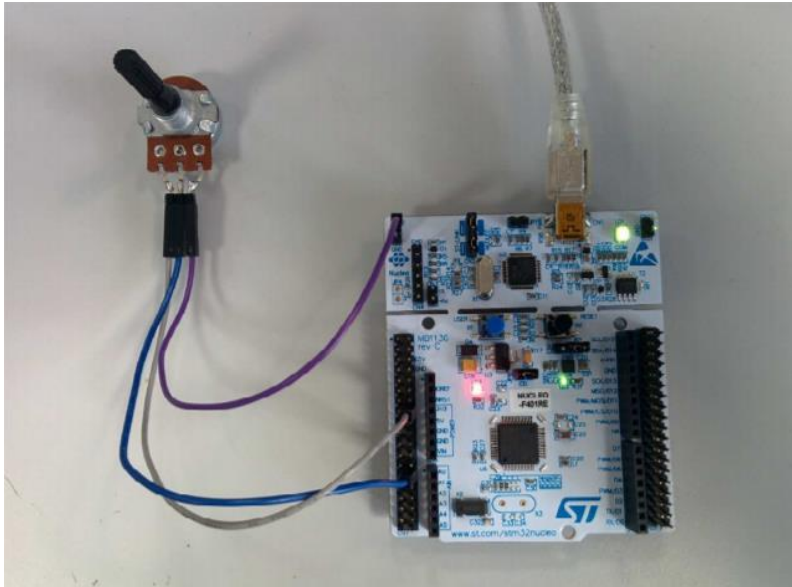
USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

PWM => per pilotare il LED

Timer 2 Channel 1 - PWM mode • 100Hz •

ADC => per variare la luminosità del LED variando proporzionalmente al valore analogico acquisito il Duty Cycle del PWM

- ADC1 IN0 – Single and regular conversion



AVVERTIMENTO:
Sarà PARECCHIO più complicato rispetto a scrivere
«analogWrite»

NUCLEO F401RE – ESEMPIO

USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

Proprietà del timer: prescaler 83, period 9999 (100hz), pulse 5000 (50%)

The screenshot displays the STM32CubeIDE interface. The left sidebar shows the 'Timers' category with TIM2 selected. The main window is split into two panes: 'TIM2 Mode and Configuration' and 'Pinout view'.

TIM2 Mode and Configuration:

- Mode:** Slave Mode (Disable), Trigger Source (Disable), Clock Source (Disable), Channel1 (PWM Generation CH1), Channel2 (Disable), Channel3 (Disable), Channel4 (Disable).
- Configuration:** NVIC Settings, DMA Settings, GPIO Settings, Parameter Settings, User Constants.
- Counter Settings:** Prescaler (PSC - 16 bits val... 83), Counter Mode (Up), Counter Period (AutoReload... 9999), Internal Clock Division (CKD) (No Division), auto-reload preload (Disable).
- Trigger Output (TRGO) Parameters:** Master/Slave Mode (MSM bit) (Disable), Trigger Event Selection (Reset).
- PWM Generation Channel 1:** Mode (PWM mode 1), Pulse (32 bits value) (5000), Fast Mode (Disable), CH Polarity (High).

Pinout view: Shows the STM32F401RETx LQFP64 package with pins labeled. The TIM2_CH1 pin is highlighted in green, corresponding to the selected timer configuration.

NUCLEO F401RE – ESEMPIO

USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

Adc: canale 0 e 84 cicli a conversione

The screenshot displays the STM32CubeIDE interface with the 'Pinout & Configuration' tab active. The 'ADC1 Mode and Configuration' window is open, showing the following settings:

- Mode:** IN0, IN1, IN2, IN3, IN4, IN5, IN6. IN2, IN3, and IN5 are selected.
- Configuration:** NVIC Settings, DMA Settings, GPIO Settings, Parameter Settings, User Constants.
- Configure the below parameters:**
 - Number Of Conversion: 1
 - External Trigger Conversion ... Regular Conversion launched by softw...
 - External Trigger Conversion ... None
 - Rank: 1
 - Channel: Channel 0
 - Sampling Time: 84 Cycles
 - ADC_Injected_ConversionMode
 - Number Of Conversions: 0
 - WatchDog
 - Enable Analog WatchDog M...:
- Sampling Time:** SamplingTime
- Parameter Description:**

The right side of the interface shows a pinout diagram of the STM32F401RETx LQFP64 package. The pins are color-coded and labeled with their functions, including VBAT, VDD, VSS, PA0-PA13, PC0-PC18, PB0-PB15, and USART_TX/RX.

NUCLEO F401RE – ESEMPIO

USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

Middleware - Freertos

uration Tool - STM32CubeIDE
ject Run Window Help

main.c startup_stm32f401retx.c tasks.c 0x8005ea8 main.c stm32f4xx_hal_msp.c stm32f4xx_hal_uart.c system_stm32f4xx.c system.c timers.c *pwm_led.ioc

Pinout & Configuration Clock Configuration Project Manager Tools

Additional Software Pinout

Categories A->Z

Analogs

Timers

- RTC
- TIM1
- ▲ TIM2
- TIM3
- TIM4
- ▲ TIM5
- ▲ TIM9
- TIM10
- TIM11

Connectivity

Multimedia

Computing

Middleware

- FATFS
- FREERTOS**
- LIBJPEG
- MBEDTLS
- PDM2PCM
- USB_DEVICE
- USB_HOST

FREERTOS Mode and Configuration

Mode

Interface CMSIS_V2

Configuration

Reset Configuration

<input checked="" type="checkbox"/> FreeRTOS Heap Usage	<input checked="" type="checkbox"/> MPU Settings
<input checked="" type="checkbox"/> Tasks and Queues	<input checked="" type="checkbox"/> Timers and Semaphores
<input checked="" type="checkbox"/> Config parameters	<input checked="" type="checkbox"/> Mutexes
<input checked="" type="checkbox"/> Include parameters	<input checked="" type="checkbox"/> User Constants

Configure the following parameters:

Search (Ctrl+F)

- API: FreeRTOS API CMSIS v2
- Versions: FreeRTOS version 10.0.1, CMSIS-RTOS version 2.00
- Kernel settings: USE_PREEMPTION Enabled, CPU_CLOCK_HZ SystemCoreClock, TICK_RATE_HZ 1000, MAX_PRIORITIES 56, MINIMAL_STACK_SIZE 128 Words, MAX_TASK_NAME_LEN 16, USE 16 BIT TICKS Disabled

Pinout view System view

NUCLEO F401RE – ESEMPIO

USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

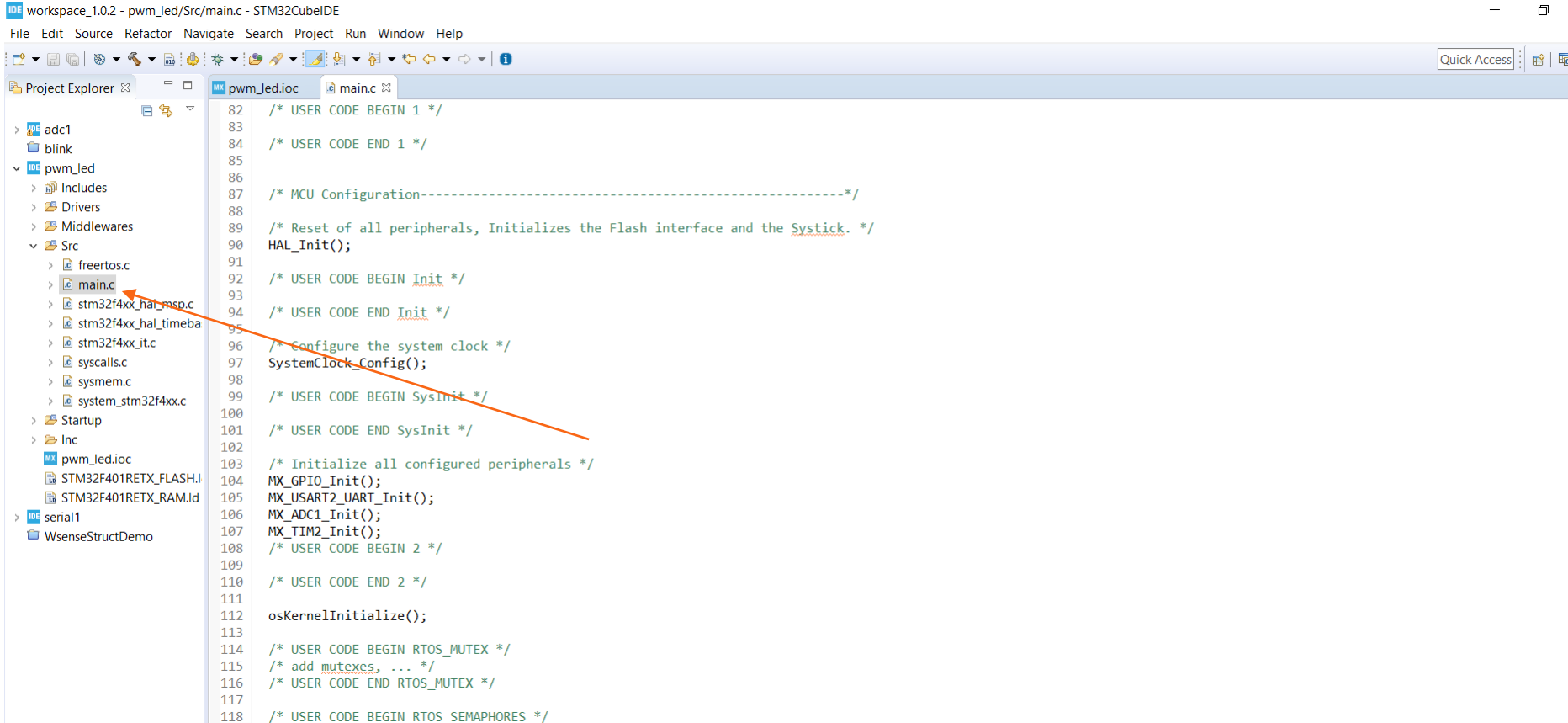
Generate project

The screenshot shows the STM32CubeIDE interface. The main window displays the 'TIM2 Mode and Configuration' settings. The 'Channel4' is highlighted in yellow. The 'Configuration' section includes 'NVIC Settings', 'DMA Settings', 'GPIO Settings', 'Parameter Settings', and 'User Constants'. A 'Search Constants' field is visible with 'add' and 'remove' buttons. On the right, a pinout diagram shows the connections for the Nucleo F401RE, including VBAT, VSS, VDD, NRST, and various peripheral pins like PC13, PC14, PH0, PH1, PA1, PA2, PA9, PA10, PA11, PA12, PA13, PA14, PA15, PA16, PA17, PA18, PA19, PA20, PA21, PA22, PA23, PA24, PA25, PA26, PA27, PA28, PA29, PA30, PA31, PA32, PA33, PA34, PA35, PA36, PA37, PA38, PA39, PA40, PA41, PA42, PA43, PA44, PA45, PA46, PA47, PA48, PA49, PA50, PA51, PA52, PA53, PA54, PA55, PA56, PA57, PA58, PA59, PA60, PA61, PA62, PA63, PA64, PA65, PA66, PA67, PA68, PA69, PA70, PA71, PA72, PA73, PA74, PA75, PA76, PA77, PA78, PA79, PA80, PA81, PA82, PA83, PA84, PA85, PA86, PA87, PA88, PA89, PA90, PA91, PA92, PA93, PA94, PA95, PA96, PA97, PA98, PA99, PA100.

NUCLEO F401RE – ESEMPIO

USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

Aprire main.c



```
workspace_1.0.2 - pwm_led/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
  adc1
  blink
  pwm_led
    Includes
    Drivers
    Middlewares
    Src
      freertos.c
      main.c
      stm32f4xx_hal_msp.c
      stm32f4xx_hal_timeba
      stm32f4xx_it.c
      syscalls.c
      systemem.c
      system_stm32f4xx.c
    Startup
    Inc
  pwm_led.ioc
  STM32F401RETX_FLASH
  STM32F401RETX_RAM.ld
  serial1
  WsenseStructDemo
main.c
82  /* USER CODE BEGIN 1 */
83
84  /* USER CODE END 1 */
85
86
87  /* MCU Configuration-----*/
88
89  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
90  HAL_Init();
91
92  /* USER CODE BEGIN Init */
93
94  /* USER CODE END Init */
95
96  /* Configure the system clock */
97  SystemClock_Config();
98
99  /* USER CODE BEGIN SysInit */
100
101  /* USER CODE END SysInit */
102
103  /* Initialize all configured peripherals */
104  MX_GPIO_Init();
105  MX_USART2_UART_Init();
106  MX_ADC1_Init();
107  MX_TIM2_Init();
108  /* USER CODE BEGIN 2 */
109
110  /* USER CODE END 2 */
111
112  osKernelInitialize();
113
114  /* USER CODE BEGIN RTOS_MUTEX */
115  /* add mutexes, ... */
116  /* USER CODE END RTOS_MUTEX */
117
118  /* USER CODE BEGIN RTOS SEMAPHORES */
```

NUCLEO F401RE – ESEMPIO

USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

Inizializzare i PWM (questo è un TIMER)

```
ain.c - STM32CubeIDE
avigate Search Project Run Window Help
Quick Access

pwm_led.ioc main.c main.c
136 };
137 defaultTaskHandle = osThreadNew(StartDefaultTask, NULL, &defaultTask_attributes);
138
139 /* definition and creation of myTask02 */
140 const osThreadAttr_t myTask02_attributes = {
141     .name = "myTask02",
142     .priority = (osPriority_t) osPriorityLow,
143     .stack_size = 128
144 };
145 myTask02Handle = osThreadNew(adcTask, NULL, &myTask02_attributes);
146
147 /* USER CODE BEGIN RTOS_THREADS */
148 /* add threads, ... */
149 /* USER CODE END RTOS_THREADS */
150
151 /* Start scheduler */
152 HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
153 osKernelStart();
154
155 /* We should never get here as control is now taken by the scheduler */
156
157 /* Infinite loop */
158
159 /* USER CODE BEGIN WHILE */
160 while (1)
161 {
162     /* USER CODE END WHILE */
163
164     /* USER CODE BEGIN 3 */
165 }
166 /* USER CODE END 3 */
167 }
168
169 /**
170  * @brief System Clock Configuration
171  * @retval None
172  */
```

NUCLEO F401RE – ESEMPIO

USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

Inserire il task

Inizializza l'ADC

Aspetta la conversione del valore

Utilizza il valore per il PWM

Scrive le informazioni sulla seriale

```
.c - STM32CubeIDE
gate Search Project Run Window Help
pwm_led.ioc main.c *main.c
397 * @brief Function implementing the myTask02 thread.
398 * @param argument: Not used
399 * @retval None
400 */
401 /* USER CODE END Header_adcTask */
402 void adcTask(void *argument)
403 {
404     /* USER CODE BEGIN adcTask */
405     /* Infinite loop */
406     for(;;)
407     {
408
409         HAL_ADC_Start(&hadc1);
410
411         HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
412
413         float rawValue = HAL_ADC_GetValue(&hadc1);
414         rawValue = ((float)rawValue) / 4095 * 10000;
415
416         htim2.Instance->CCR1 = (uint16_t) rawValue+100;
417
418         char msg[10];
419         sprintf(msg, "%d\n", (uint16_t)rawValue);
420         HAL_UART_Transmit(&huart2, msg, strlen(msg), 0xFFFF);
421
422     }
423     /* USER CODE END adcTask */
424 }
425
426 /**
427 * @brief Period elapsed callback in non blocking mode
428 * @note This function is called when TIM1 interrupt took place, inside
429 * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
430 * a global variable "uwTick" used as application time base.
431 * @param htim : TIM handle
432 * @retval None
433 */
```

NUCLEO F401RE – ESEMPIO

USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

Programmare il target

The image shows a screenshot of an IDE with a C program for Nucleo F401RE and its configuration dialog. The code in the background is as follows:

```
70 /*
71 int main(void)
72 {
73 /* USER CODE BEGIN 1 */
74
75 /* USER CODE END 1 */
76
77 /* MCU Configuration-----
78
79 /* Reset of all peripherals, Initializes the Flash interface
80 HAL_Init();
81
82 /* USER CODE BEGIN Init */
83
84 /* USER CODE END Init */
85
86 /* Configure the system clock */
87 SystemClock_Config();
88
89 /* USER CODE BEGIN SysInit */
90
91 /* USER CODE END SysInit */
92
93
94 /* Initialize all configured peripherals */
95 MX_GPIO_Init();
96 MX_USART2_UART_Init();
97 MX_TIM2_Init();
98 /* USER CODE BEGIN 2 */
99
100 /* USER CODE END 2 */
101 //HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
102 /* Infinite loop */
103 /* USER CODE BEGIN WHILE */
104 while (1)
105 {
106 /* USER CODE END WHILE */
107
108 /* USER CODE BEGIN 3 */
109 }
110 /* USER CODE END 3 */
111 }
112
113 /**
114 * @brief System Clock Configuration
115 * @retval None
116 */
117 void SystemClock_Config(void)
118 {
119 RCC_OscInitTypeDef RCC_OscInitStruct = {0};
120 RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
```

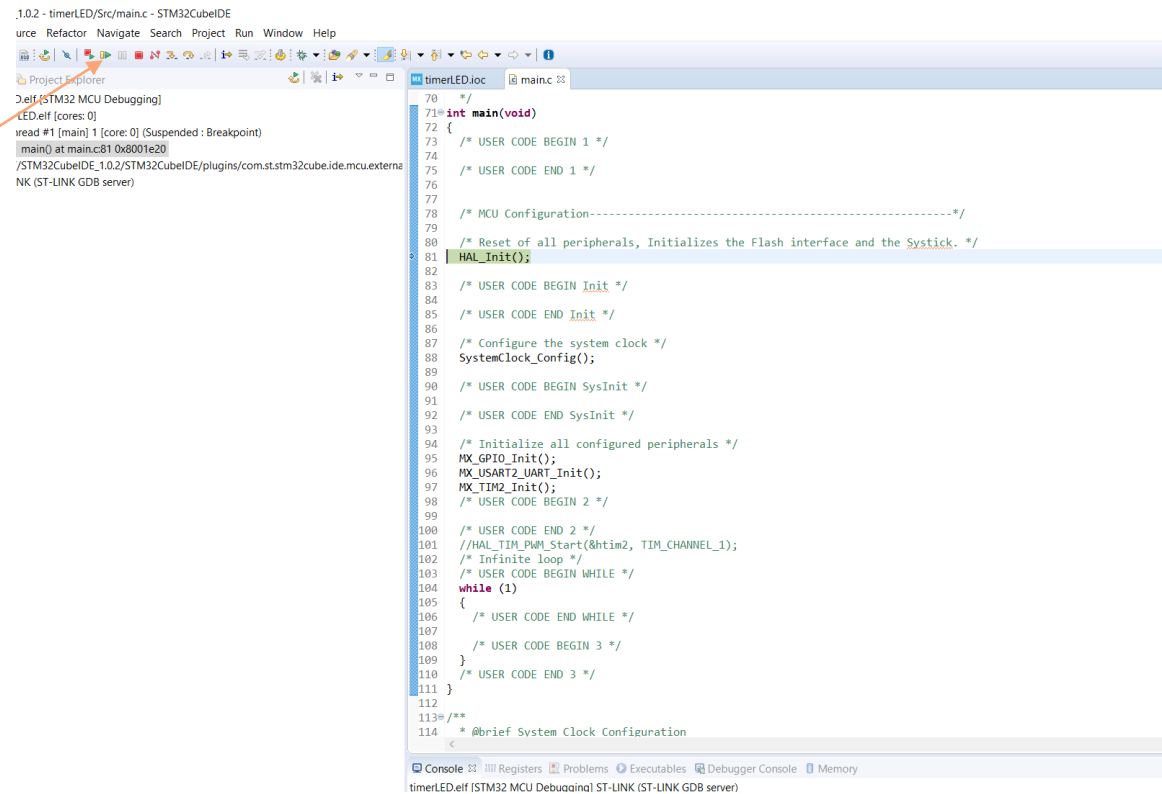
The configuration dialog, titled "Edit Configuration", shows the following settings:

- Name: timerLED.elf
- Debugger: Debug/timerLED.elf
- Project: timerLED
- Build Configuration: Use Active
- Build options: Use workspace settings
- Buttons: Revert, Apply, OK, Cancel

NUCLEO F401RE – ESEMPIO

USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

Avviare il target



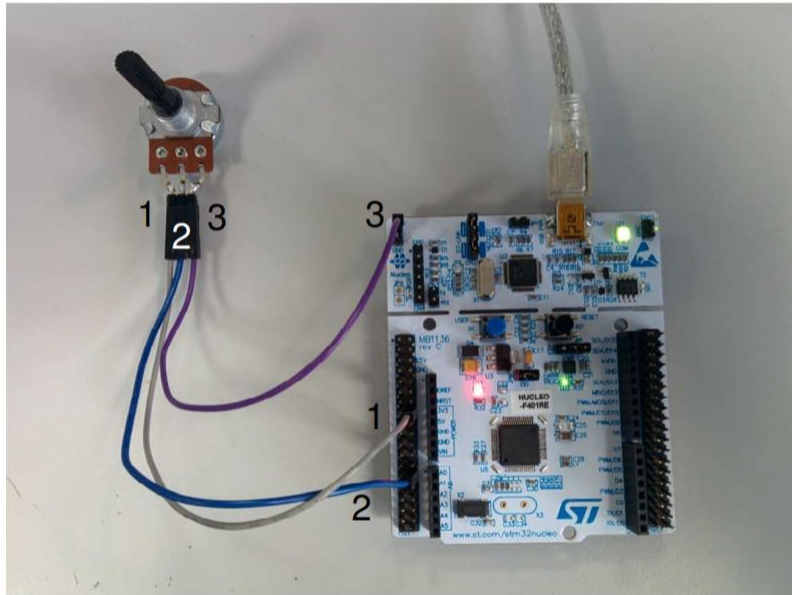
The screenshot shows the STM32CubeIDE interface. On the left, the Project Explorer shows the project structure. The main.c file is open in the editor. The Run button (a green play icon) in the top toolbar is highlighted with an orange arrow. The code in main.c is as follows:

```
70 /*
71 int main(void)
72 {
73 /* USER CODE BEGIN 1 */
74
75 /* USER CODE END 1 */
76
77
78 /* MCU Configuration-----*/
79
80 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
81 HAL_Init();
82
83 /* USER CODE BEGIN Init */
84
85 /* USER CODE END Init */
86
87 /* Configure the system clock */
88 SystemClock_Config();
89
90 /* USER CODE BEGIN SysInit */
91
92 /* USER CODE END SysInit */
93
94 /* Initialize all configured peripherals */
95 MX_GPIO_Init();
96 MX_USART2_UART_Init();
97 MX_TIM2_Init();
98 /* USER CODE BEGIN 2 */
99
100 /* USER CODE END 2 */
101 //HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
102 /* Infinite loop */
103 /* USER CODE BEGIN WHILE */
104 while (1)
105 {
106 /* USER CODE END WHILE */
107
108 /* USER CODE BEGIN 3 */
109 }
110 /* USER CODE END 3 */
111 }
112
113 /**
114 * @brief System Clock Configuration
```

NUCLEO F401RE – ESEMPIO

USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

collegamenti



- 1. 3.3V => Pin 16 CN7
- 2. AIN0 => Pin 28 CN7
- 3. Ground