Programmazione di sistemi multicore

Michele Martinelli
Michele.martinelli@uniroma1.it

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

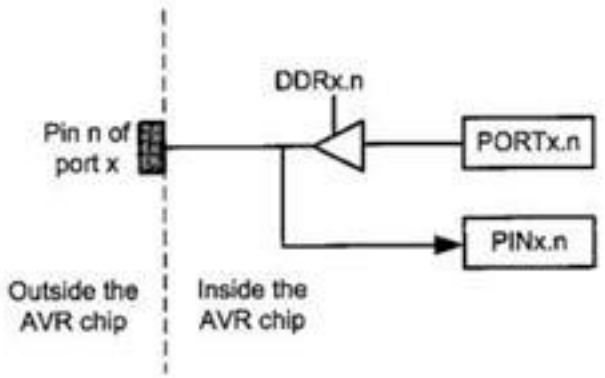
Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DDRD – The Port D Data Direction Register

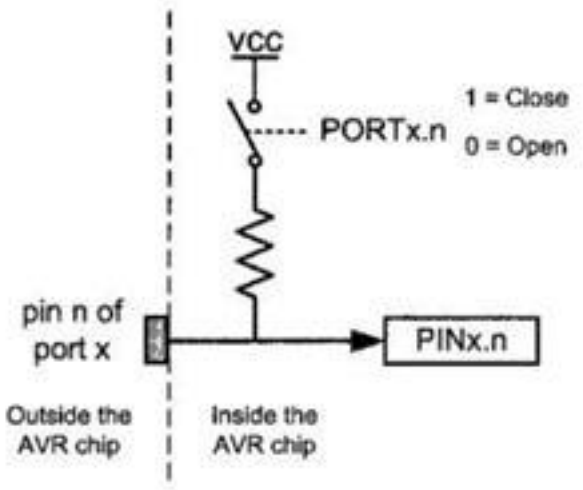
Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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/OPortin 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

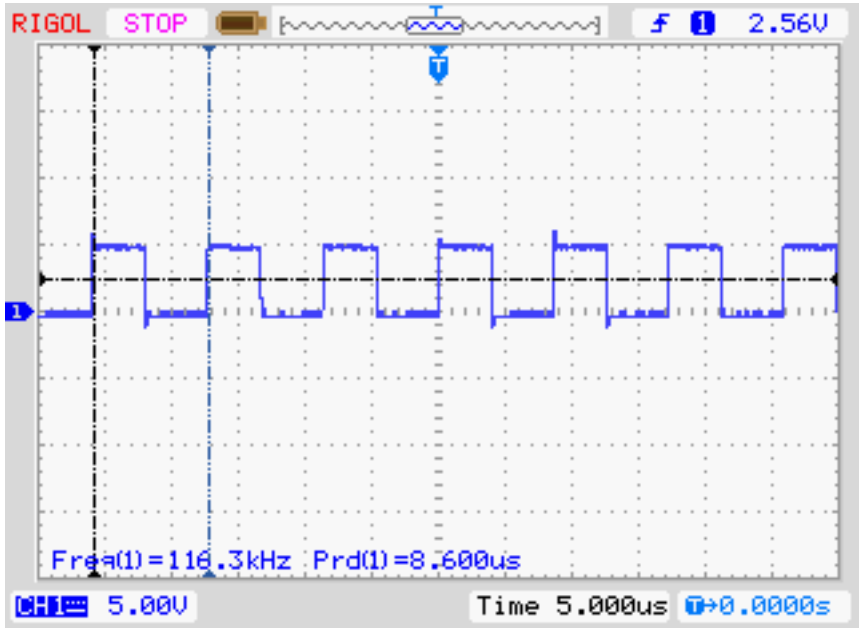
Ma anche dei vantaggi

- 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;
}
```

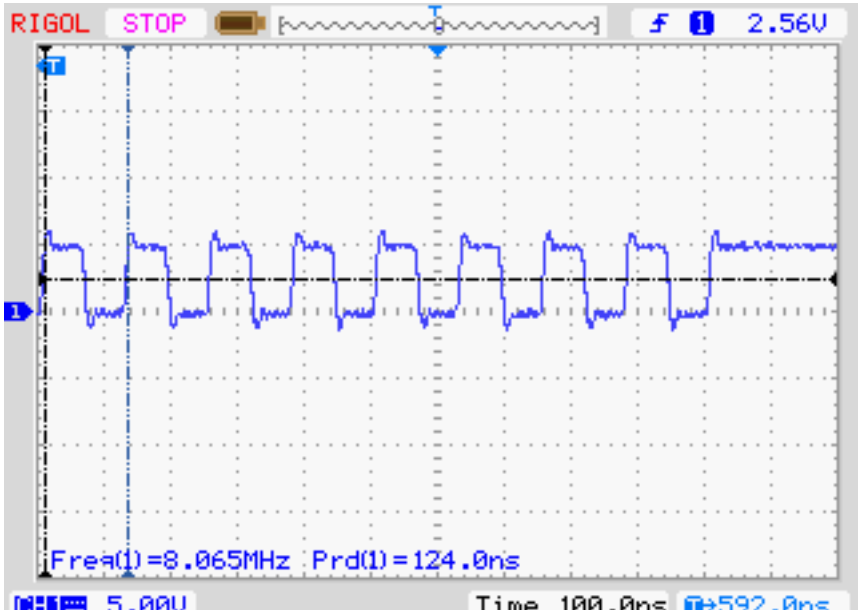
Risultato



...e port manipulation

```
PORTB =0;
PORTB =B100000;
```

Risultato



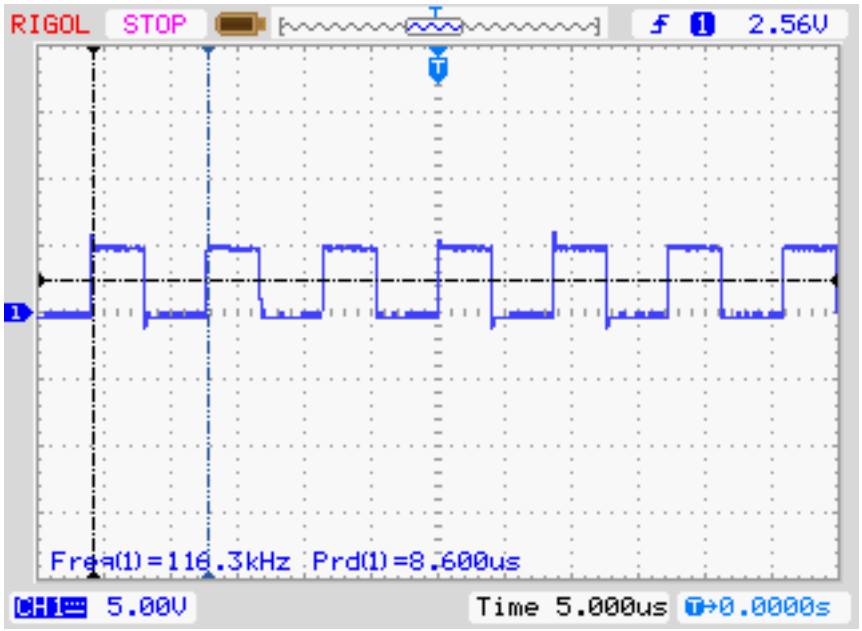
Importante: inizializzazione delle variabili

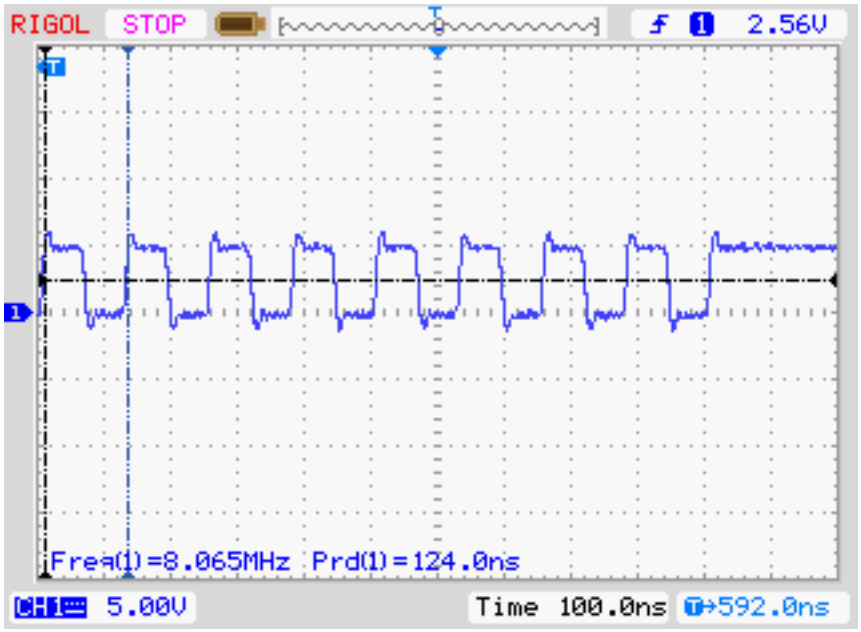
A = B00000101 -> B binario

A = 0x05 -> 0x esadecimale

A = 5 -> decimale

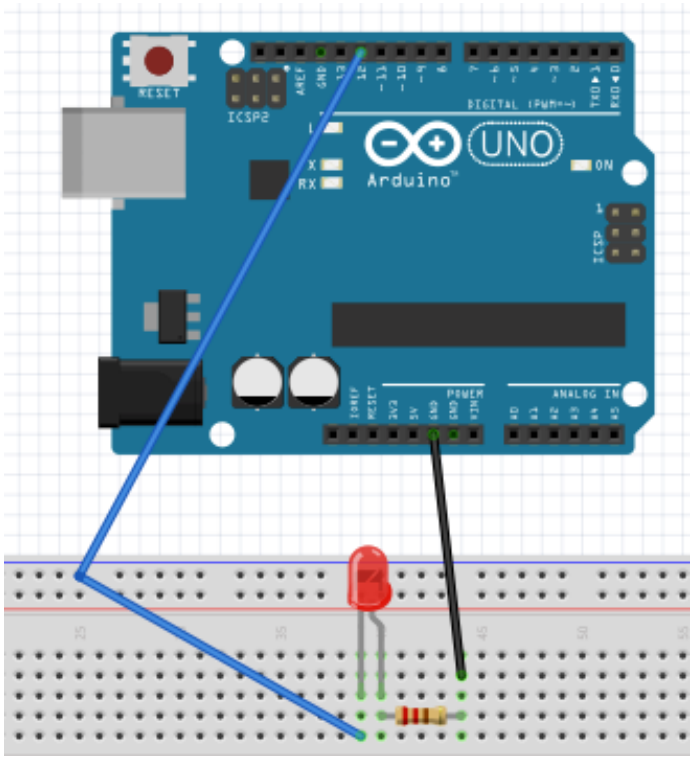
Confrontando...





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 xxxxxx11 (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

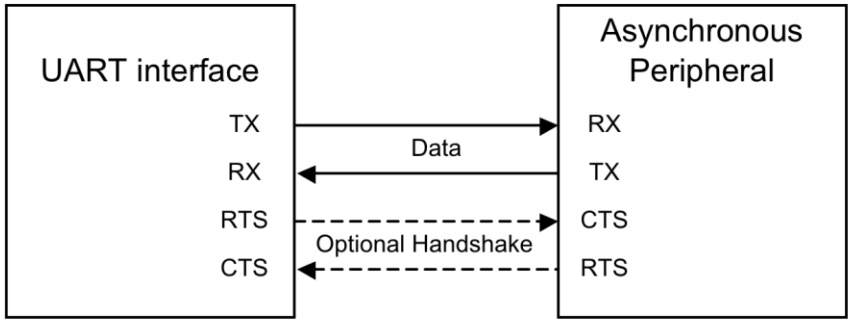
00000011 & 11111100 -> 00000000

Stringa: 01010111

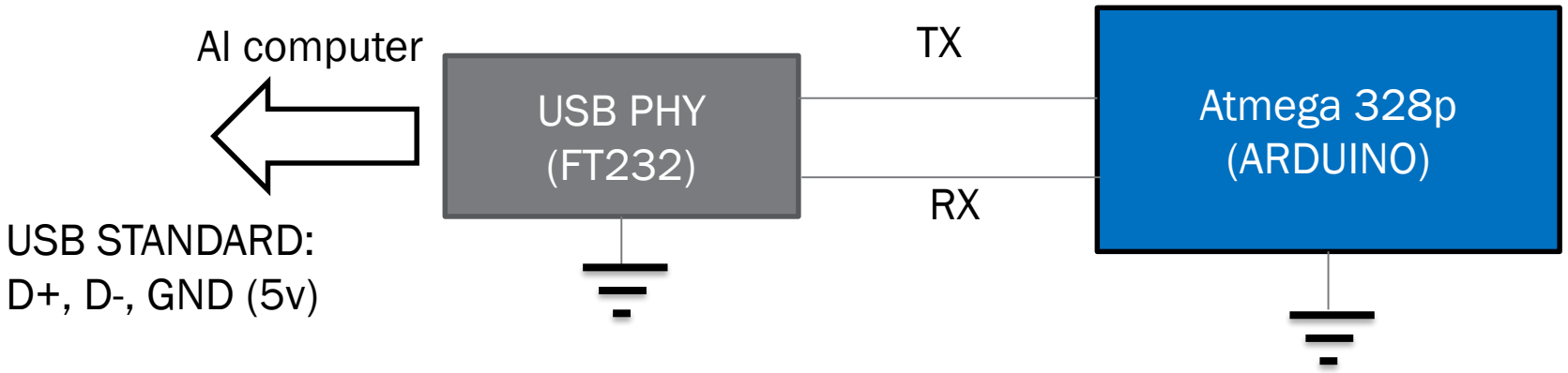
01010111 & 11111100 -> 01010100

COMUNICAZIONE SERIALE
ASYNCHRONOUS SERIAL INTERFACE (UART)

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



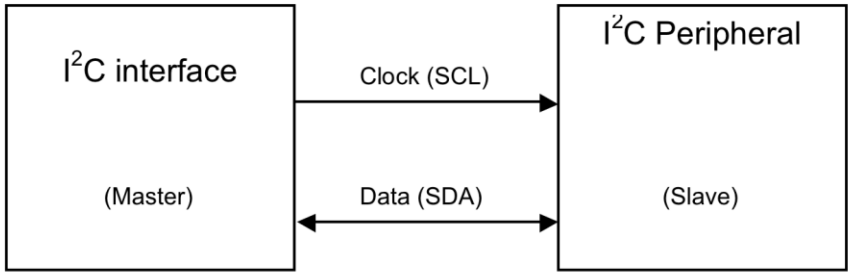
Molto lento e poco robusto, ma molto semplice da implementare e utilizzare



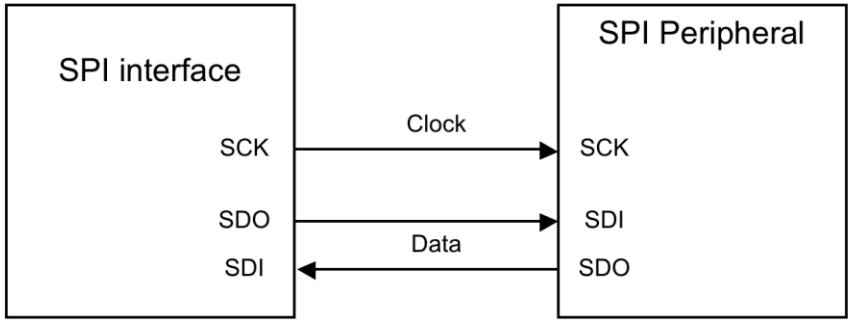
SYNCHRONOUS SERIAL INTERFACES

Utilizzati per parlare con periferiche esterne

I²C



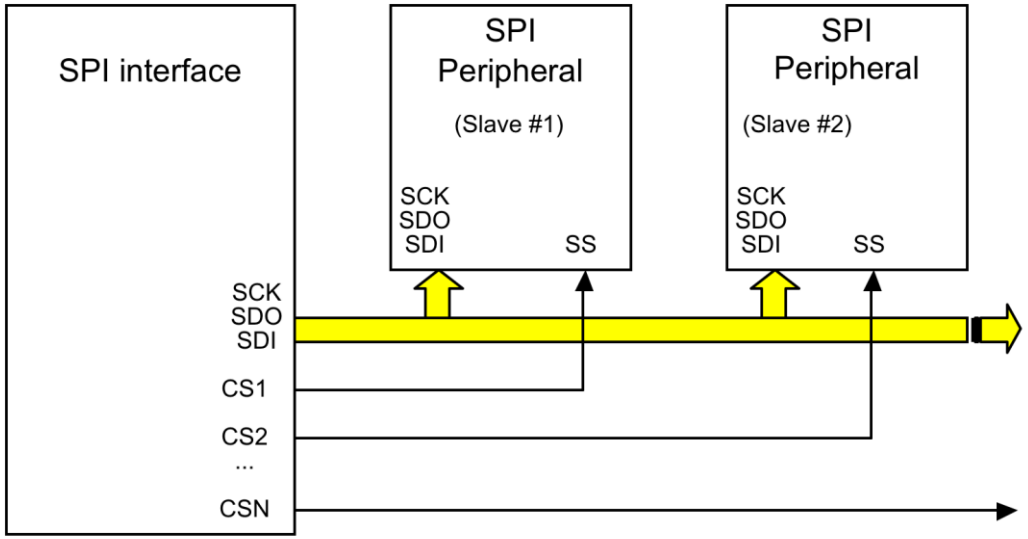
SPI



SPI BUS

Composto da:

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



Vantaggi

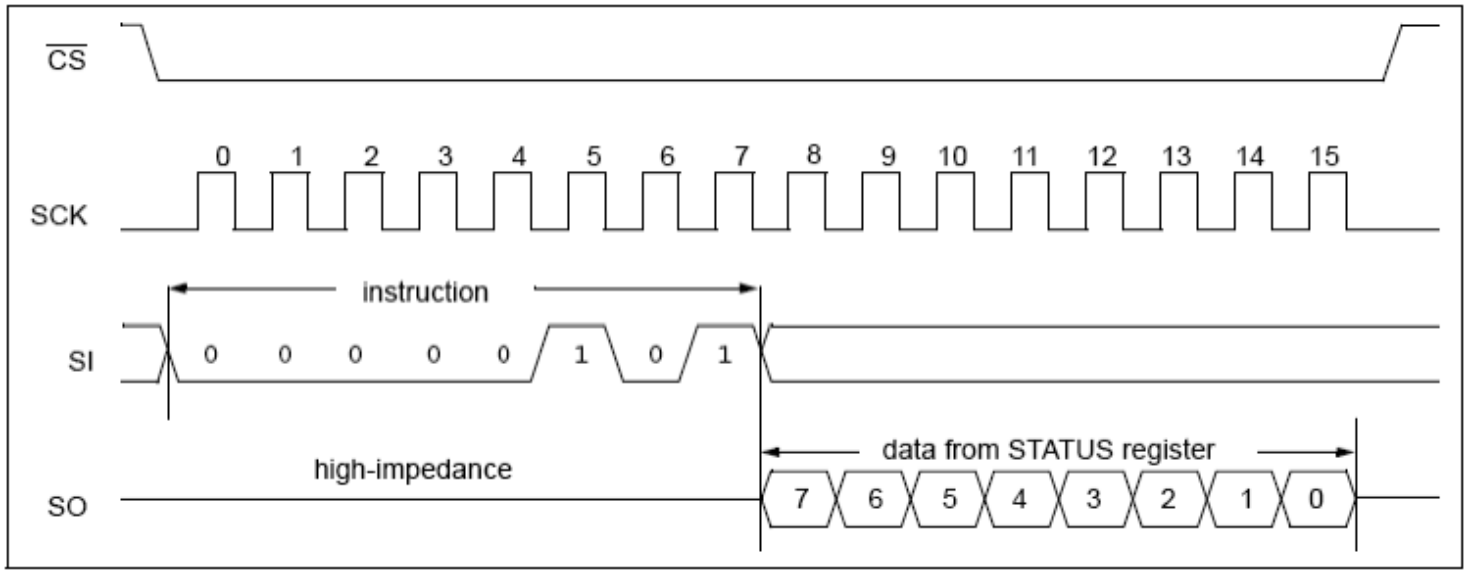
- Bidirezionale
- Solitamente non richiedono resistenze di pullup

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

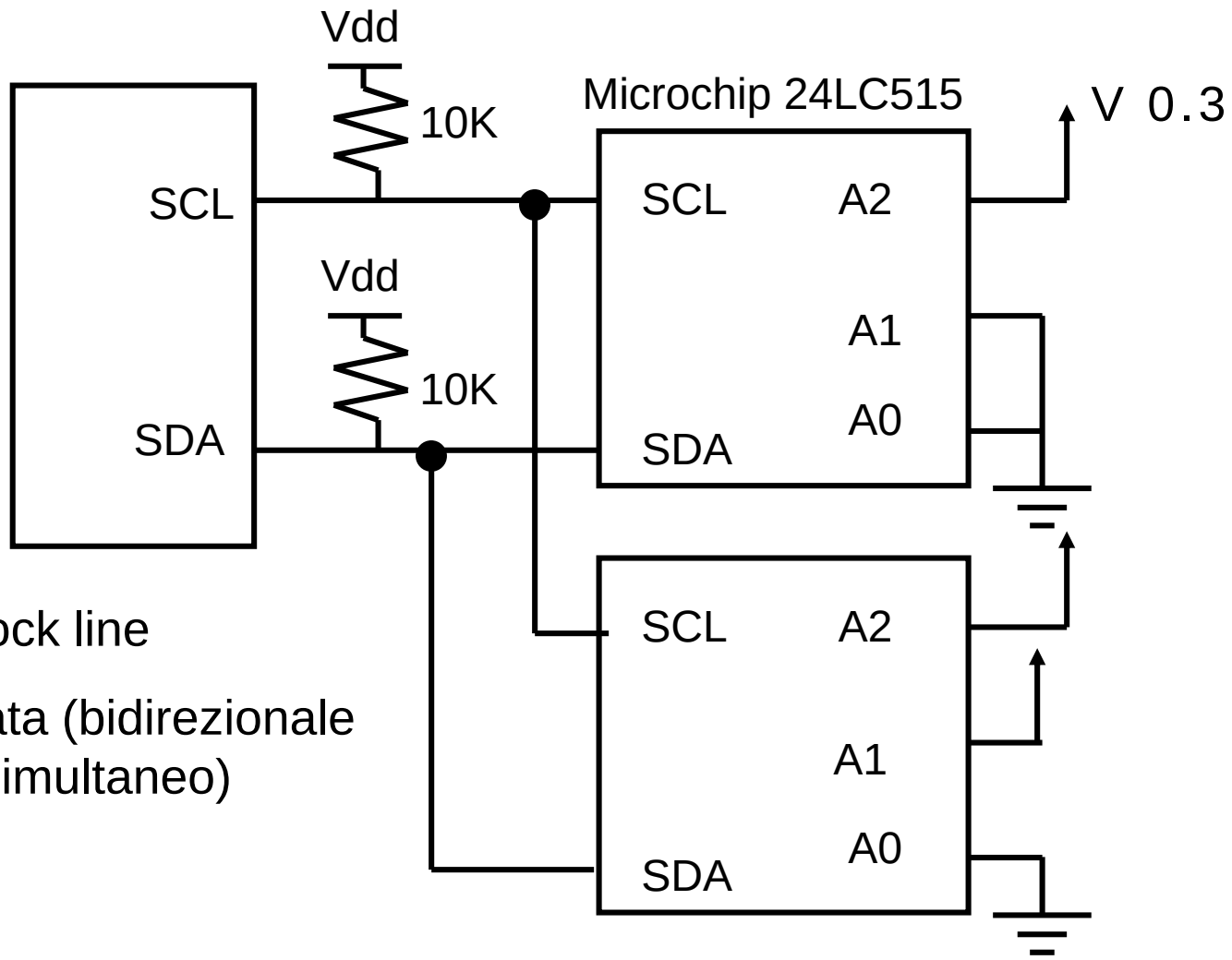
```
// send a Write command
    CSEE = 0;                   // select the Serial EEPROM
    WriteSPI2( CMD_WRITE);      // write command
    WriteSPI2( addr_MSB);       // address MSB first
    WriteSPI2( addr_LSB);       // address LSB (word aligned)
    WriteSPI2( data);           // send 8-bit of 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.



SCL – clock line

SDA – data (bidirezionale ma non simultaneo)

I^2C FEATURES

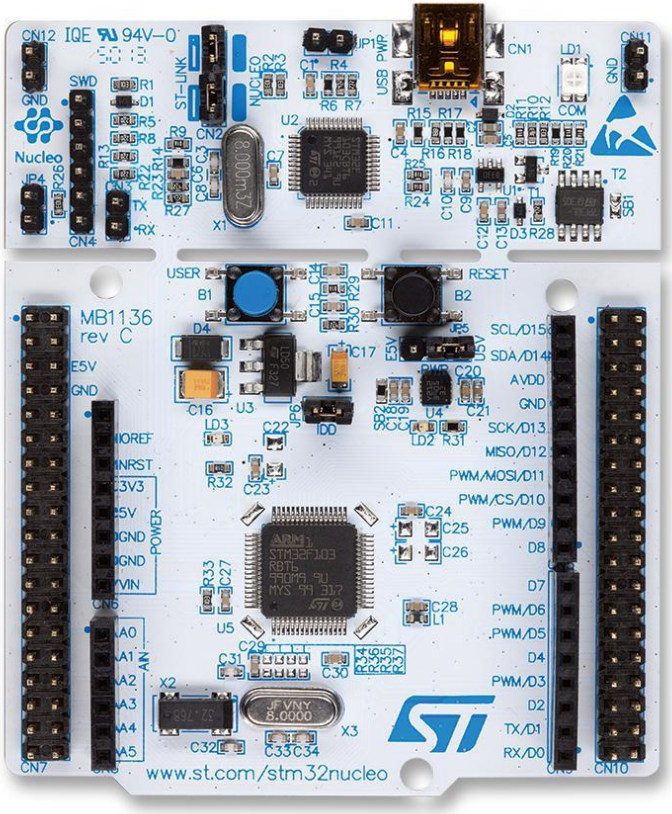
Una singola linea richiesta anche in presenza di molti slave

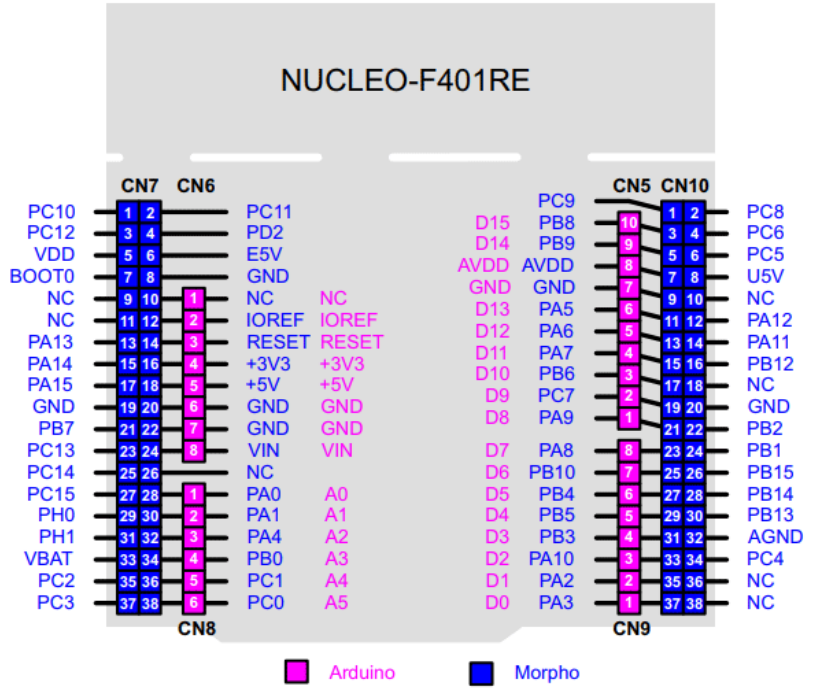
- Ogni transazione inizia con l'indirizzo del ricevente
- Tutti I ricevitori ascoltano e confrontano con il proprio indirizzo

Vengono utilizzate due line:

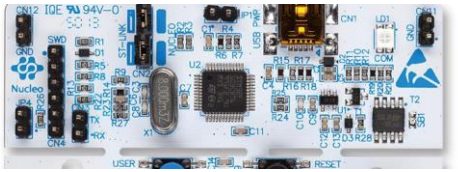
- SDA (data)
- SCL (clock)
- Bidirezionale ma non full-duplex (o trasmetti o ricevi)

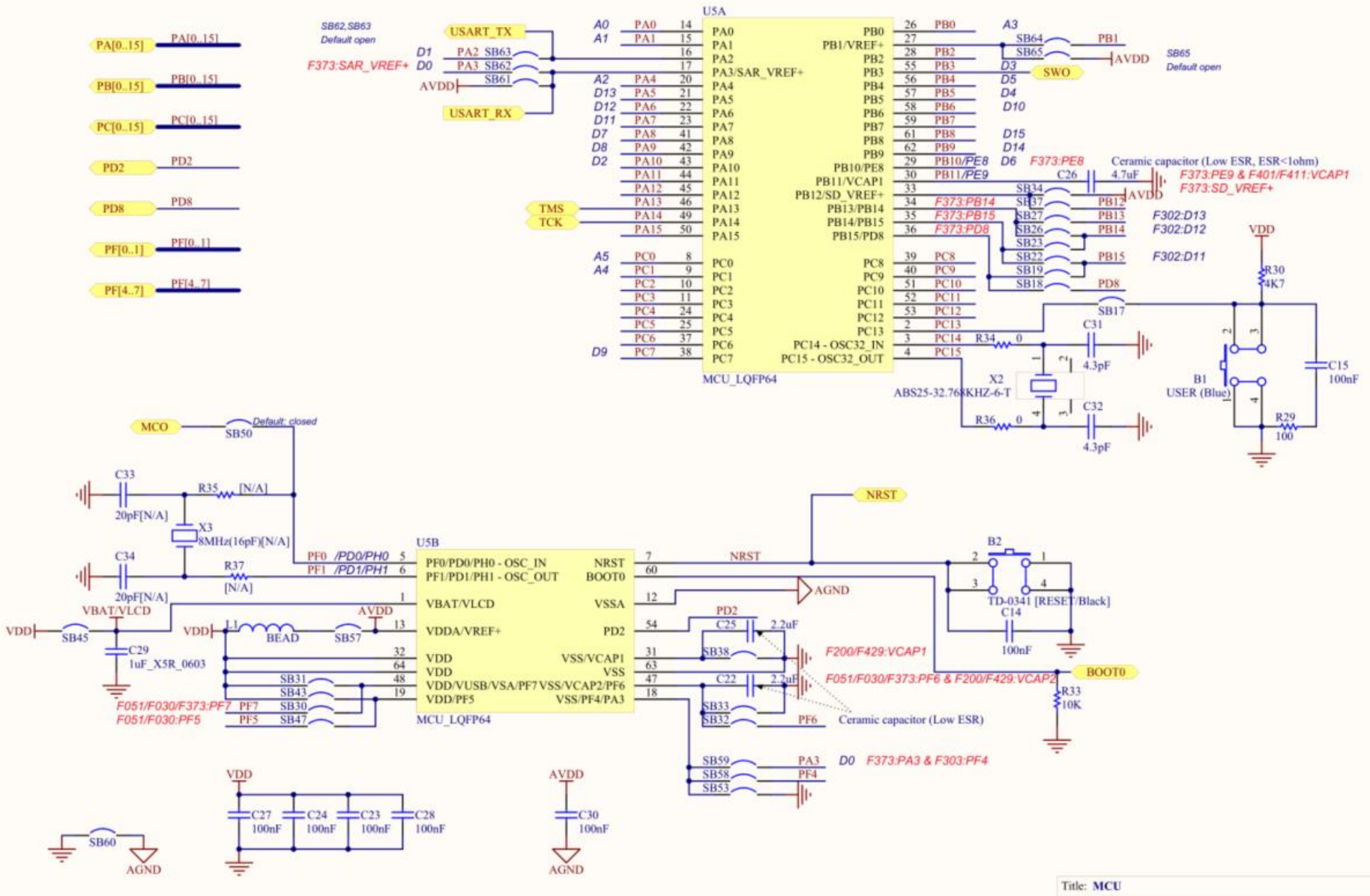
NUCLEO F401RE - HW



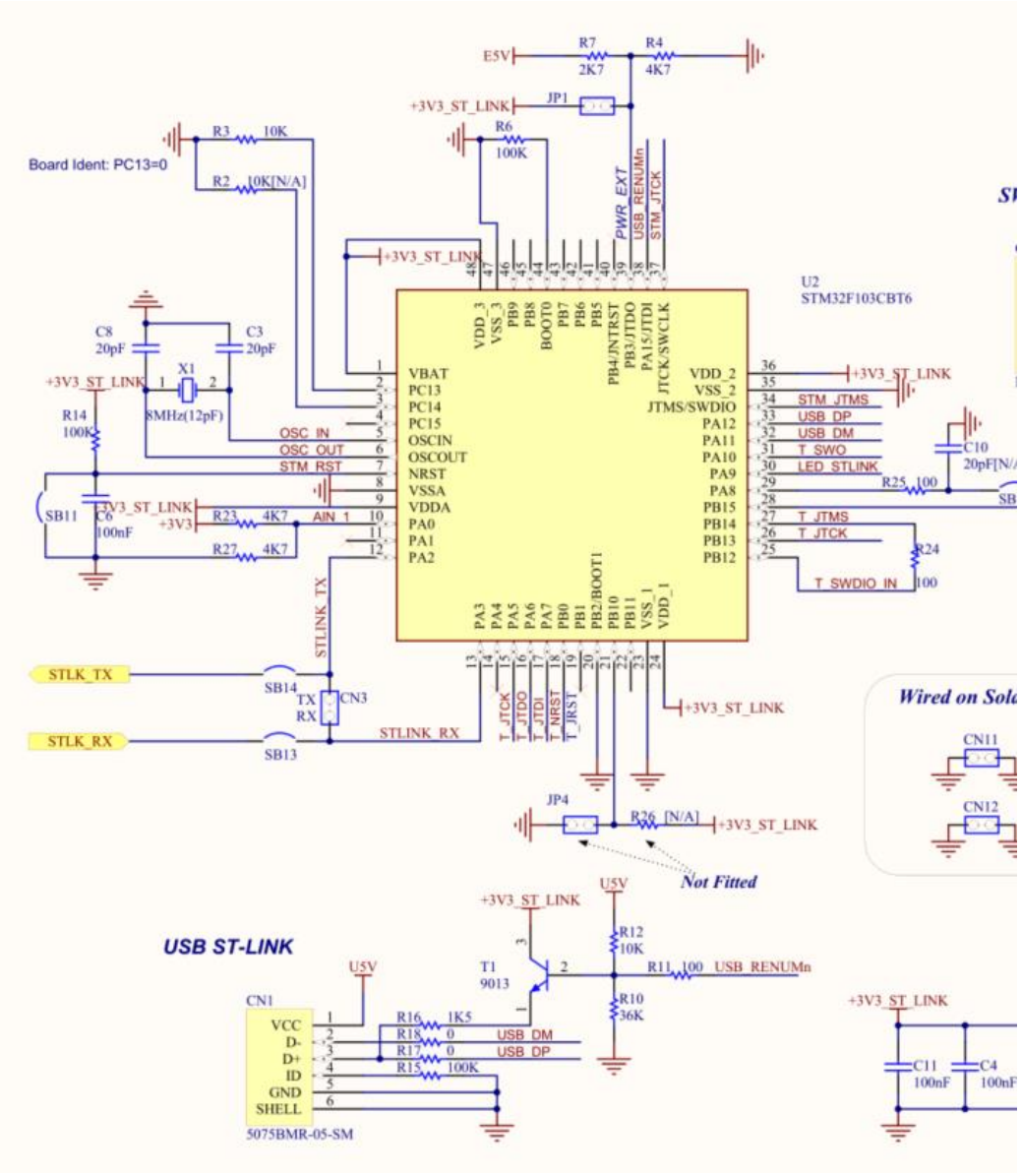


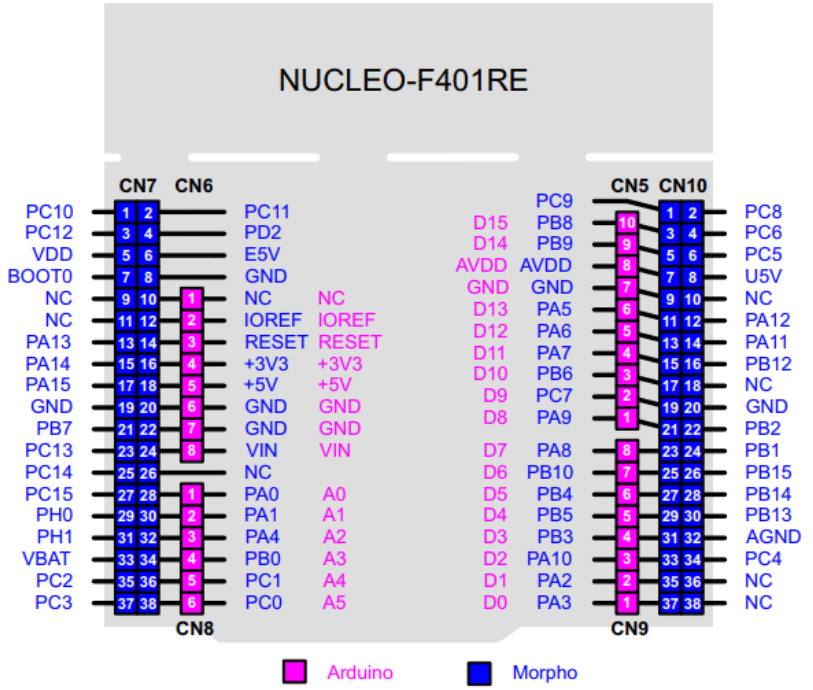
NUCLEO F401RE - SCHEMA



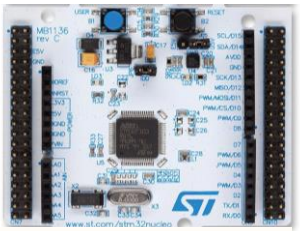


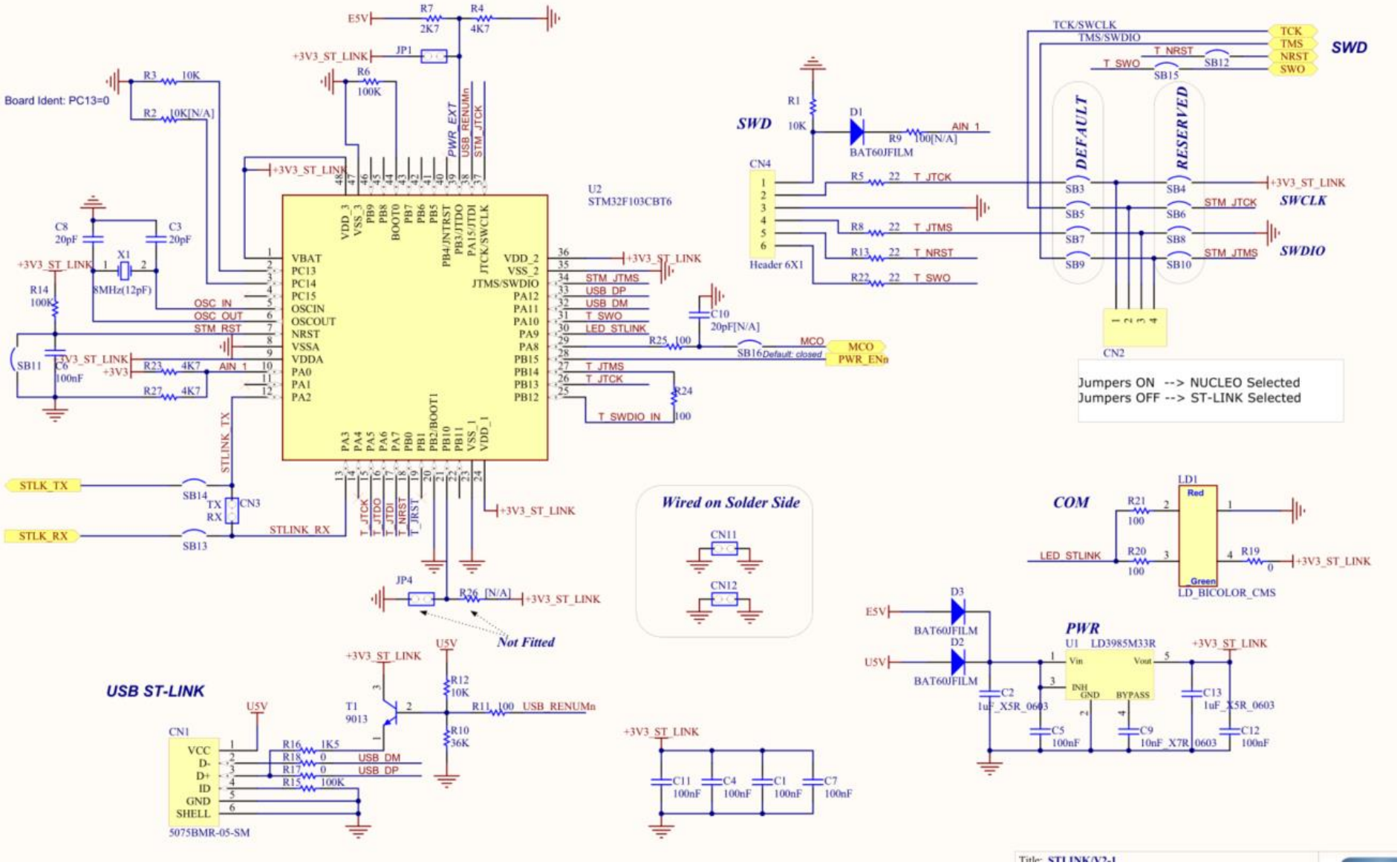
NUCLEO F401RE - SCHEMA





NUCLEO F401RE - SCHEMA

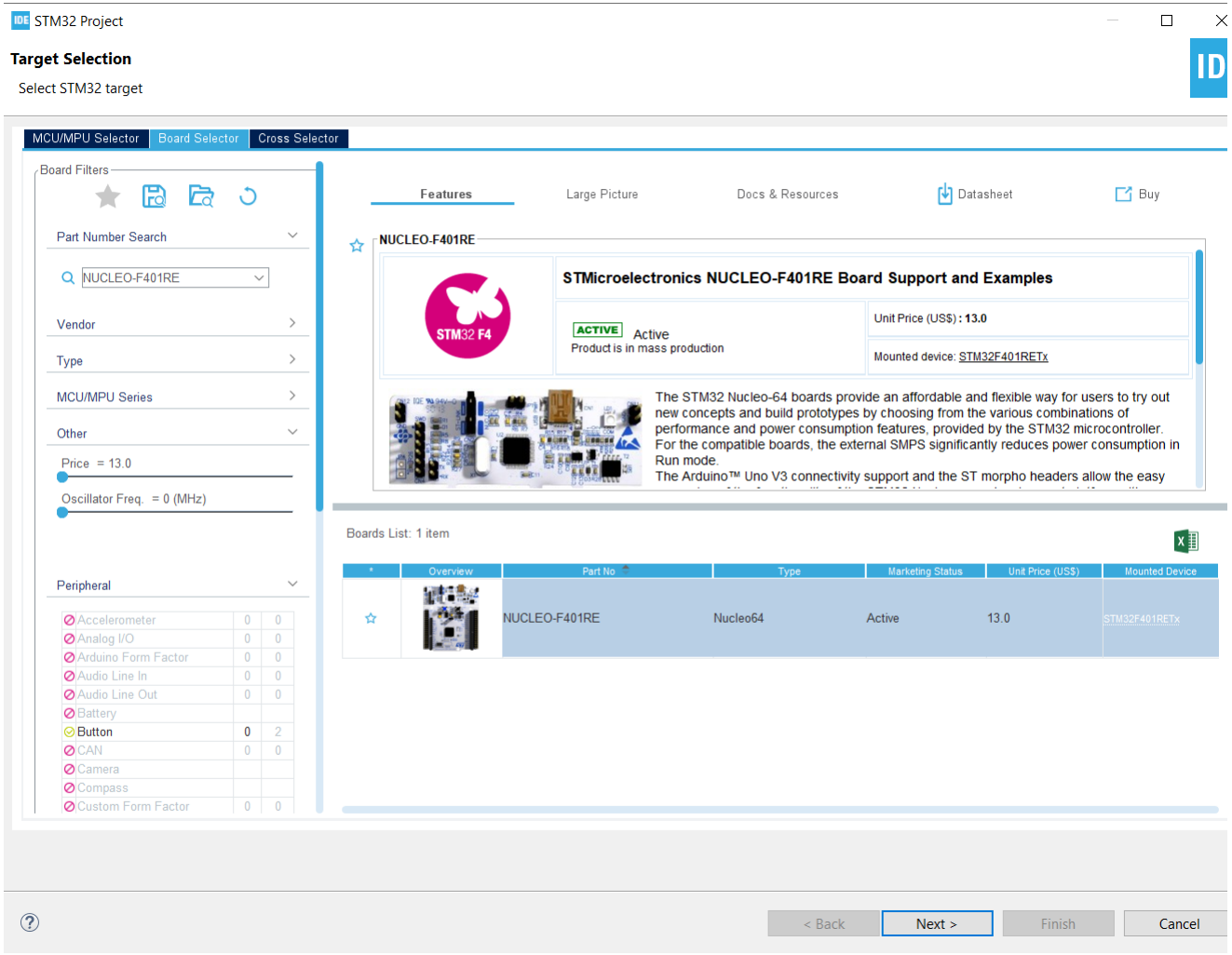




NUCLEO F401RE – ESEMPIO

BLINK LED INTEGRATO

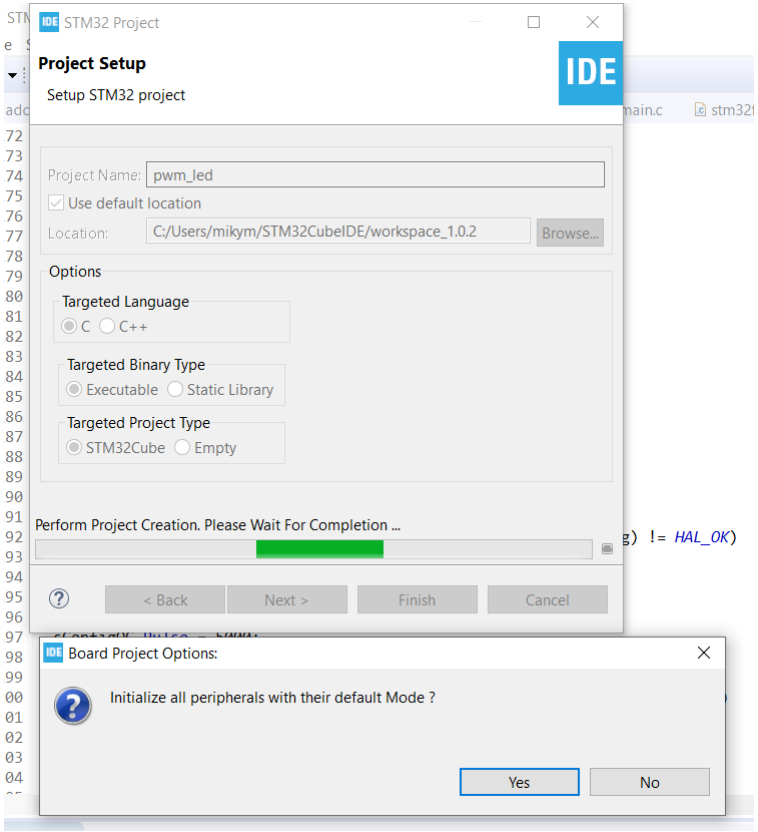
File -> new -> project -> STM32project -> board selector -> nucleo F401RE



NUCLEO F401RE – ESEMPIO

BLINK LED INTEGRATO

Assegnare nome – e poi yes



NUCLEO F401RE – ESEMPIO

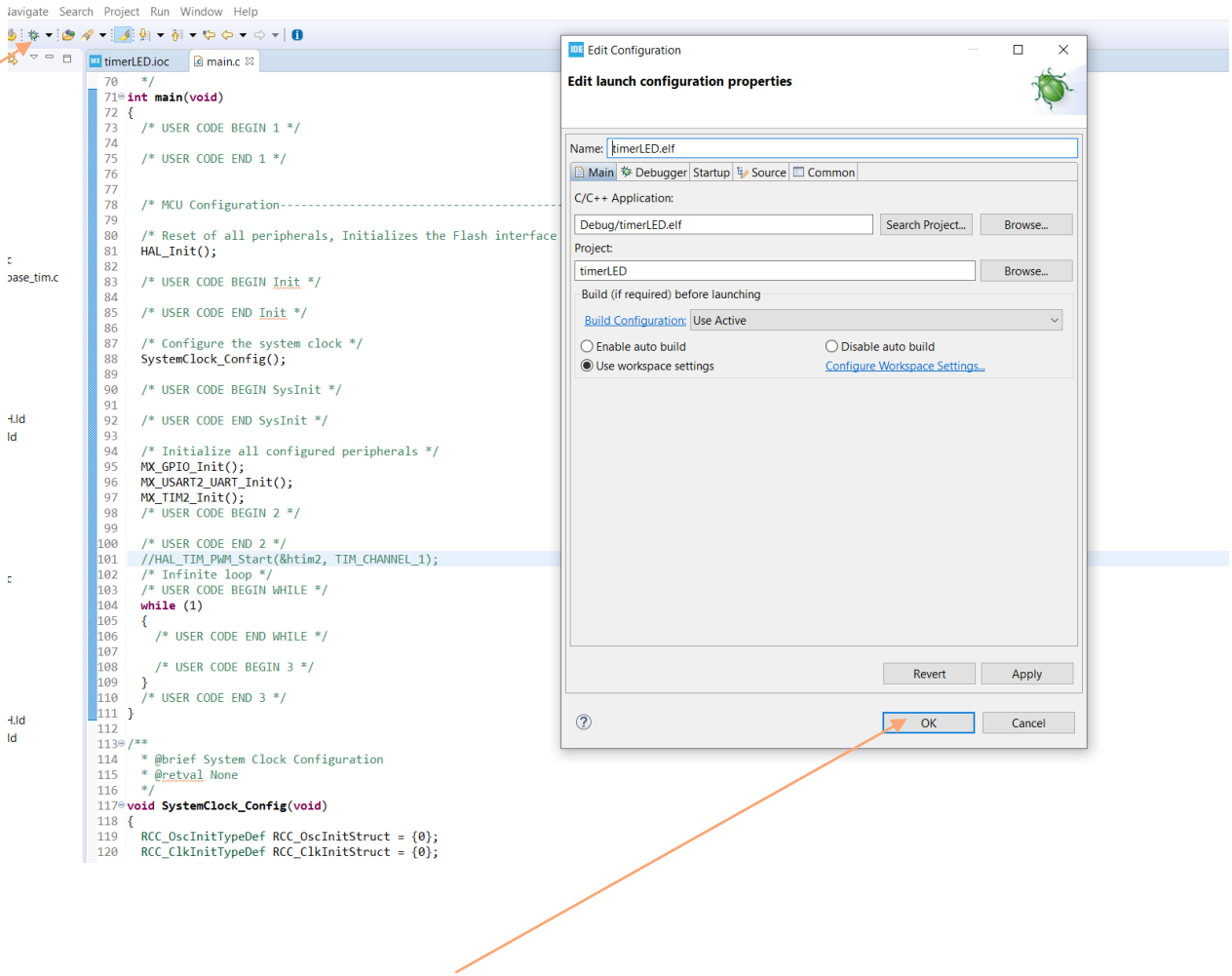
BLINK LED INTEGRATO

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

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

NUCLEO F401RE – ESEMPIO

BLINK LED INTEGRATO

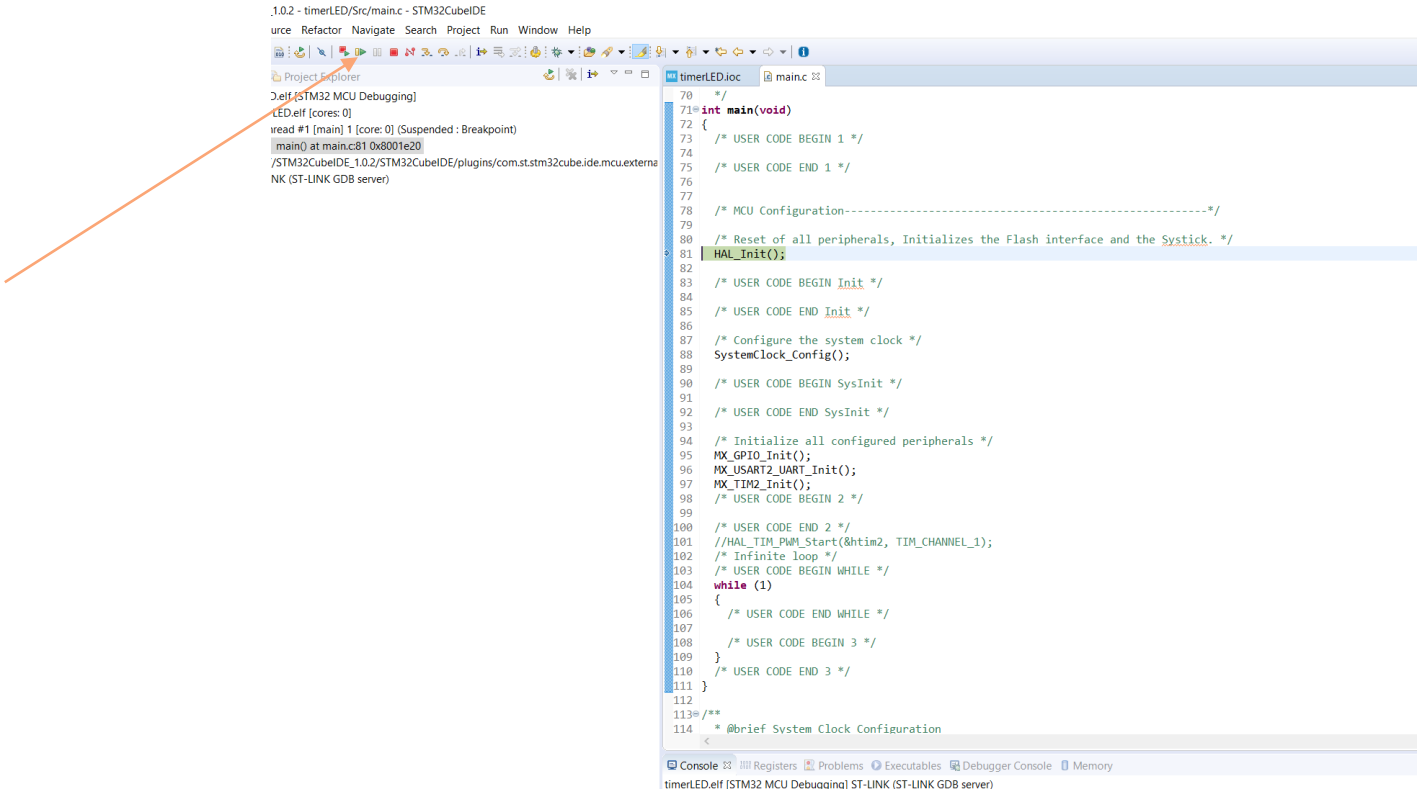
Programmare il target



NUCLEO F401RE – ESEMPIO

BLINK LED INTEGRATO

Avviare il target



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 lest 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

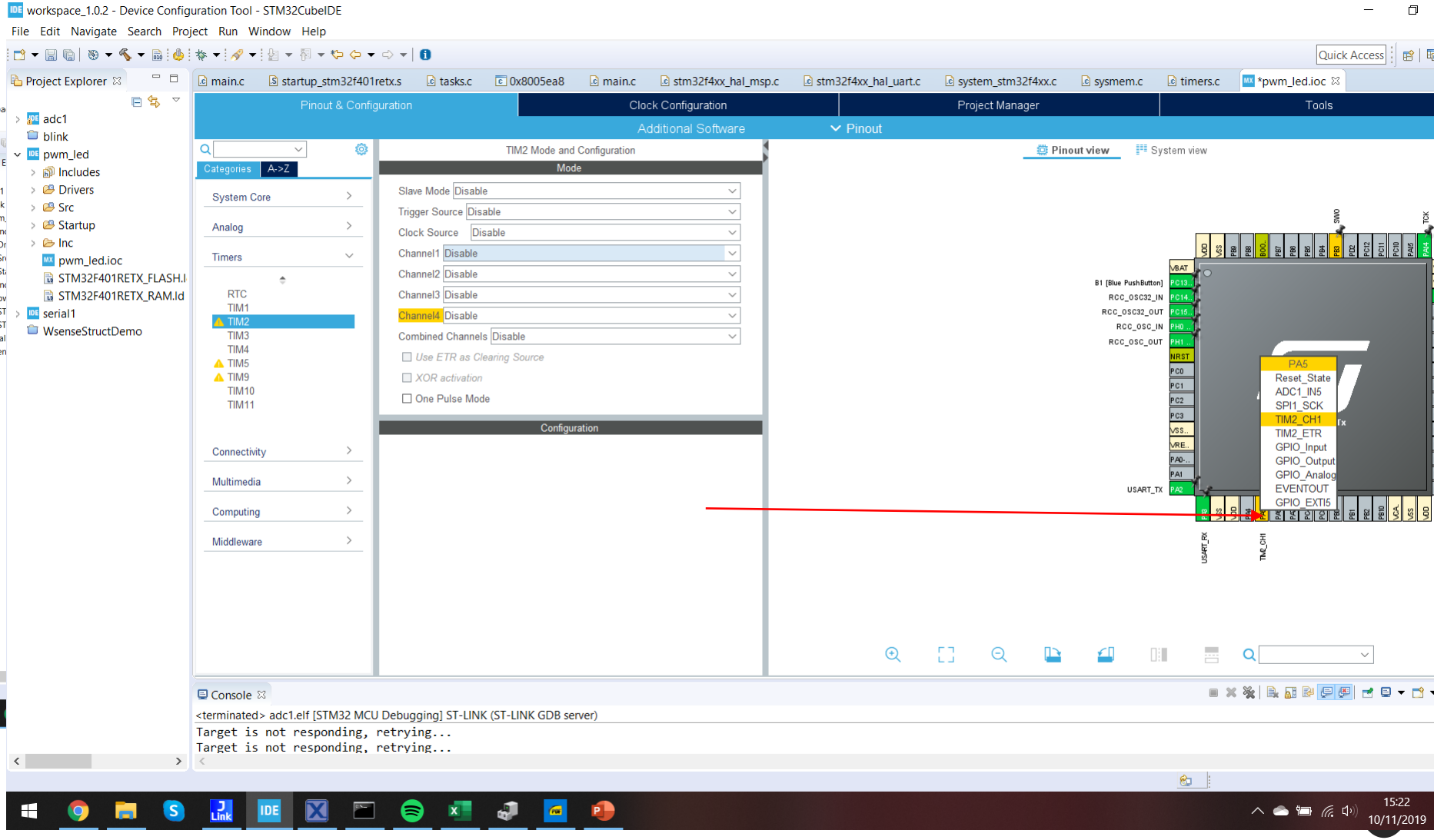
RICEVERE DALLA SERIALE

```
printMessage:

	printWelcomeMessage();

	while (1)  {
		opt = readUserInput();
		processUserInput(opt);
		if(opt == 3)
			goto printMessage;
	}
}

void printWelcomeMessage(void) {
	HAL_UART_Transmit(&huart2, (uint8_t*)"\033[0;0H", strlen("\033[0;0H"), HAL_MAX_DELAY);
	HAL_UART_Transmit(&huart2, (uint8_t*)"\033[2J", strlen("\033[2J"), HAL_MAX_DELAY);
	HAL_UART_Transmit(&huart2, (uint8_t*)WELCOME_MSG, strlen(WELCOME_MSG), HAL_MAX_DELAY);
	HAL_UART_Transmit(&huart2, (uint8_t*)MAIN_MENU, strlen(MAIN_MENU), HAL_MAX_DELAY);
}

uint8_t readUserInput(void) {
	char readBuf[1];

	HAL_UART_Transmit(&huart2, (uint8_t*)PROMPT, strlen(PROMPT), HAL_MAX_DELAY);
	HAL_UART_Receive(&huart2, (uint8_t*)readBuf, 1, HAL_MAX_DELAY);
	return atoi(readBuf);
}


uint8_t processUserInput(uint8_t opt) {
	char msg[30];

	if(!opt || opt > 3)
		return 0;

	sprintf(msg, "%d", opt);
	HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);

	switch(opt) {
	case 1:
		HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
		break;
	case 2:
		sprintf(msg, "\r\nUSER BUTTON status: %s", HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == GPIO_PIN_RESET ? "PRESSED" : "RELEASED");
		HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
		break;
	case 3:
		return 2;
	};

	return 1;
}
```

https://github.com/cnoviello/mastering-stm32/blob/master/nucleo-f091RC/src/ch8/main-ex1.c

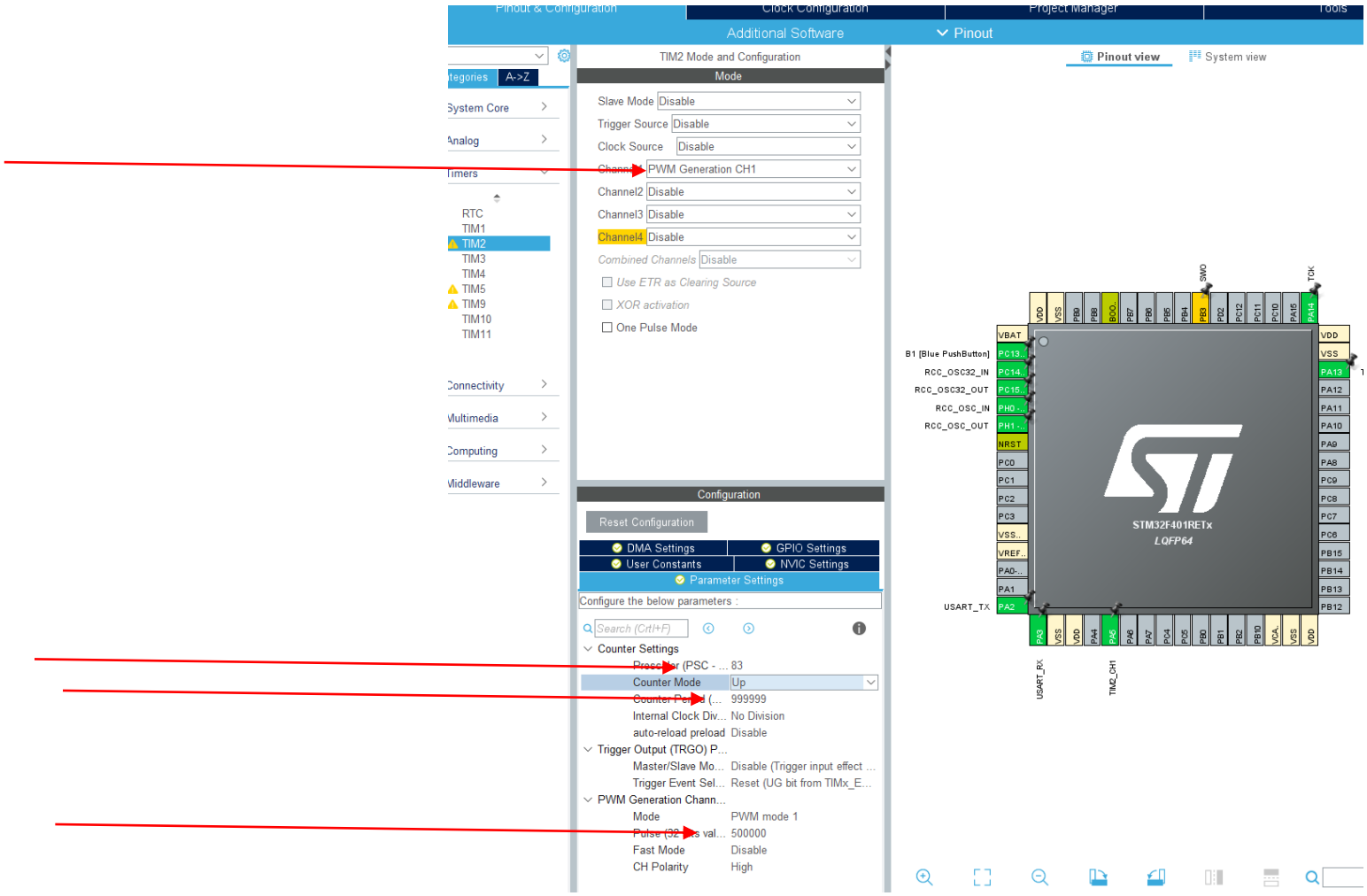
NUCLEO F401RE – ESEMPIO

USIAMO IL TIMER PER FAR BLINKARE IL LED INTEGRATO



NUCLEO F401RE – ESEMPIO

USIAMO IL TIMER PER FAR BLINKARE IL LED INTEGRATO



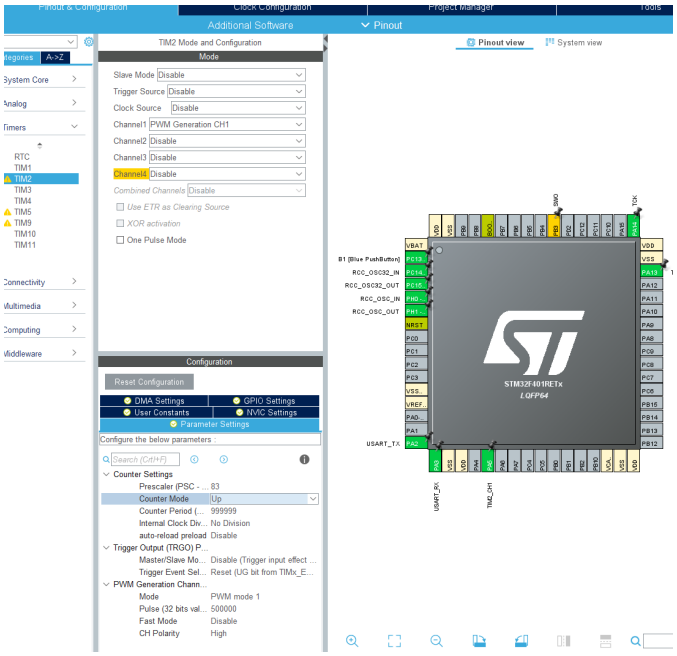
NUCLEO F401RE – ESEMPIO

USIAMO IL TIMER PER FAR BLINKARE IL LED INTEGRATO

Nel main.c inserire:

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

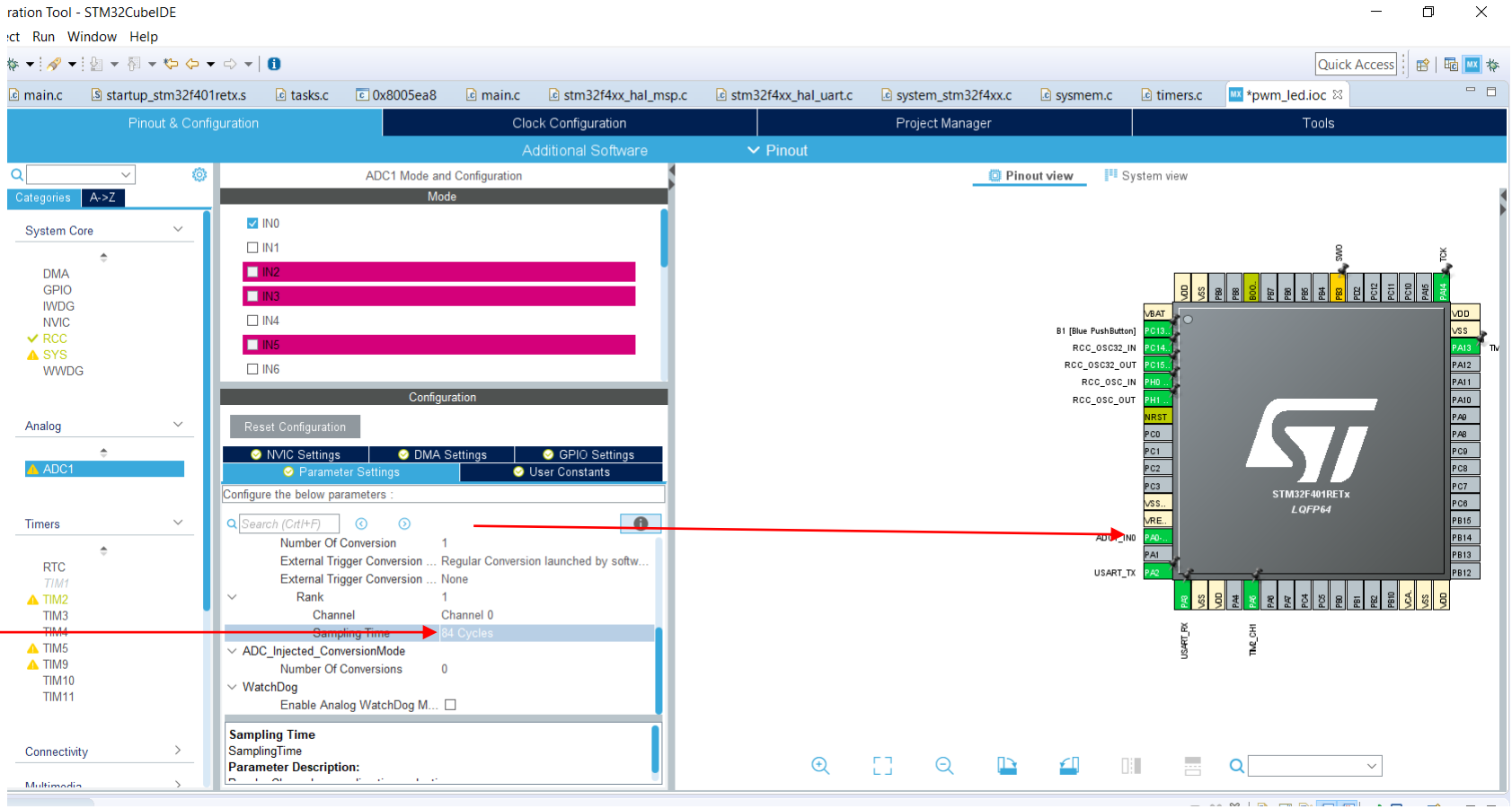
Private a cambiare period (e pulse in accordo)



NUCLEO F401RE – ESEMPIO

LEGGIAMO IL VALORE DI TENSIONE DA UN POTENZIOMETRO

Adc: canale 0 e 84 cicli a conversione



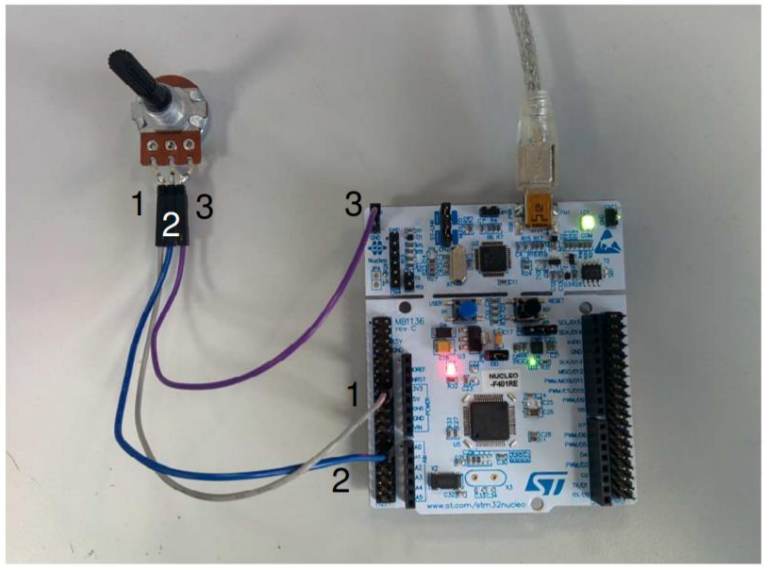
NUCLEO F401RE – ESEMPIO

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   }
```



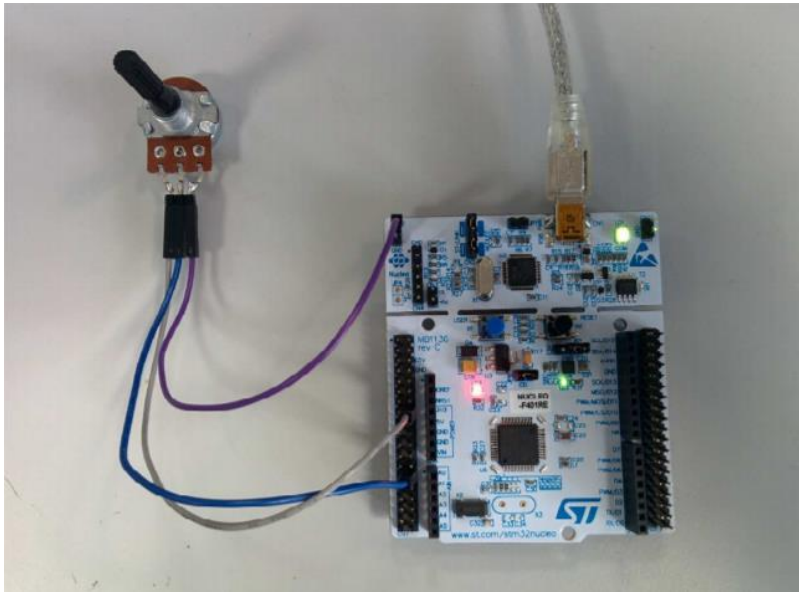
1. 3.3V => Pin 16 CN7
2. AIN0 => Pin 28 CN7
3. Ground

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 •

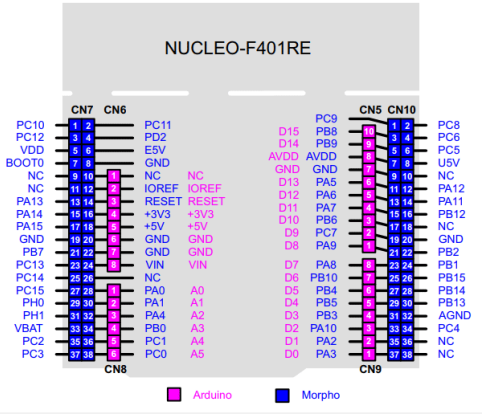
ADC => per variare la luminosità del LED variando proporzionalmente al valore analogico acquisito il Duty Cycle del PWM
• ADC1 IN0 – Single and regular conversion



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

NUCLEO F401RE – ESEMPIO

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



Schermata pin

ADC

Timer

onfiguration Tool - STM32CubeIDE

Project Run Window Help

main.c | startup_stm32f401retx.s | tasks.c | 0x8005ea8 | main.c | stm32f4xx_hal_msp.c | stm32f4xx_hal_uart.c | system_stm32f4xx.c | sysmem.c | timers.c | pwm_led.ioc

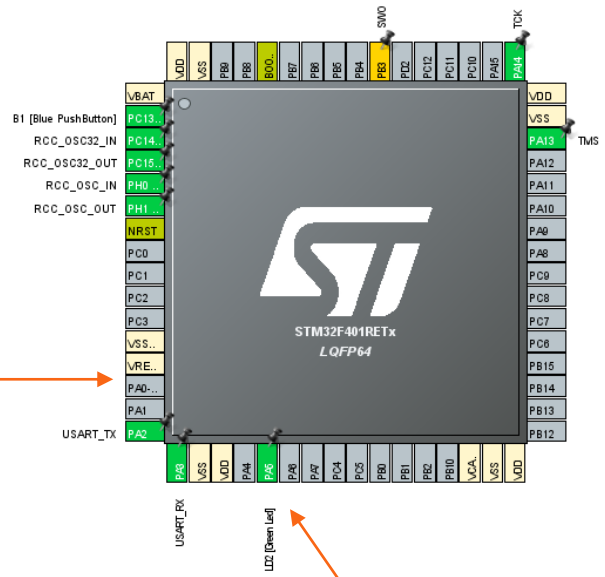
Pinout & Configuration | Clock Configuration | Project Manager | Tools

Additional Software | Pinout

Pinout view | System view

Categories | A->Z

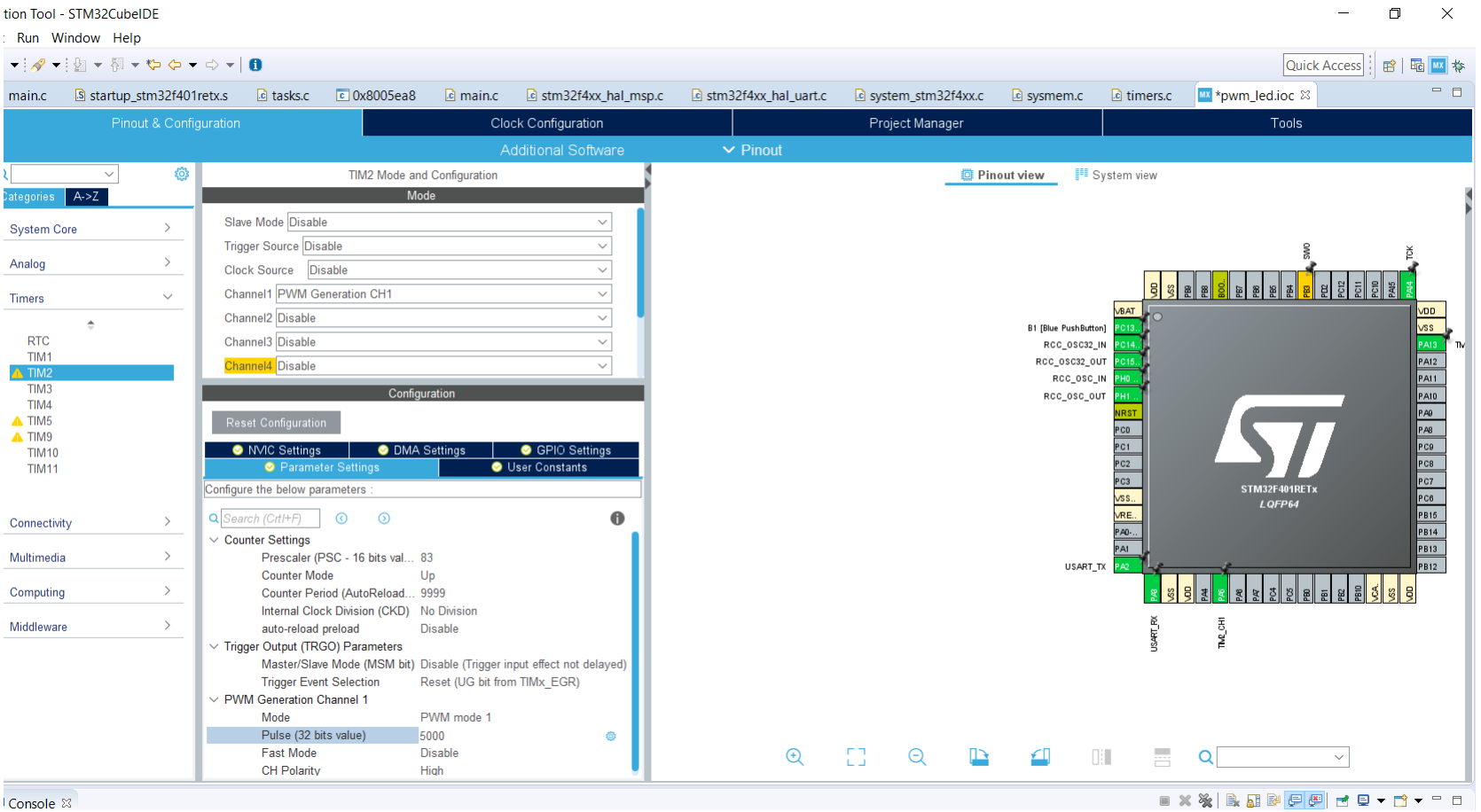
- System Core
- Analog
- Timers
- Connectivity
- Multimedia
- Computing
- Middleware



NUCLEO F401RE – ESEMPIO

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

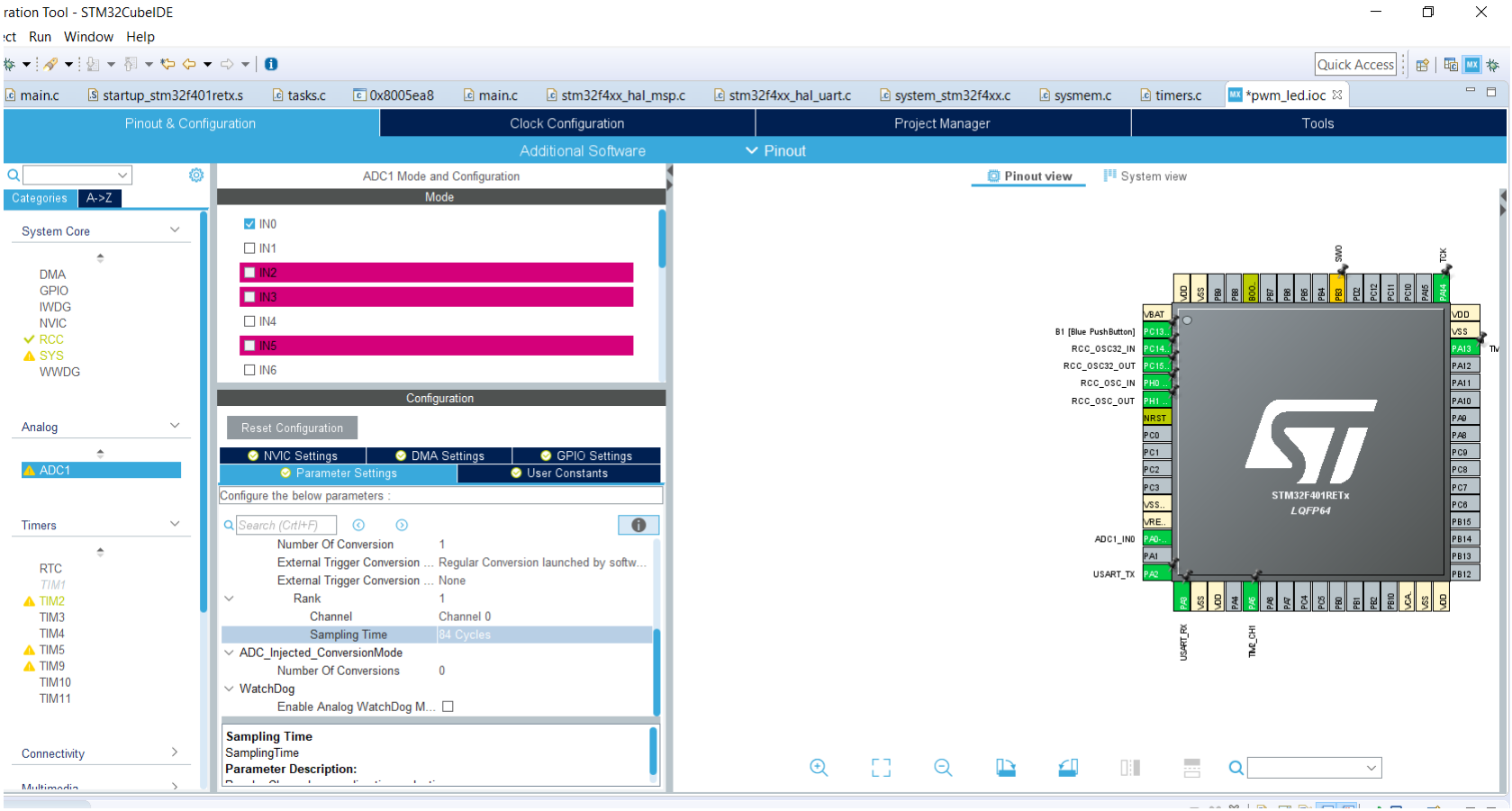
Proprietà del timer: prescaler 83, period 9999 (100hz), pulse 5000 (50%)



NUCLEO F401RE – ESEMPIO

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

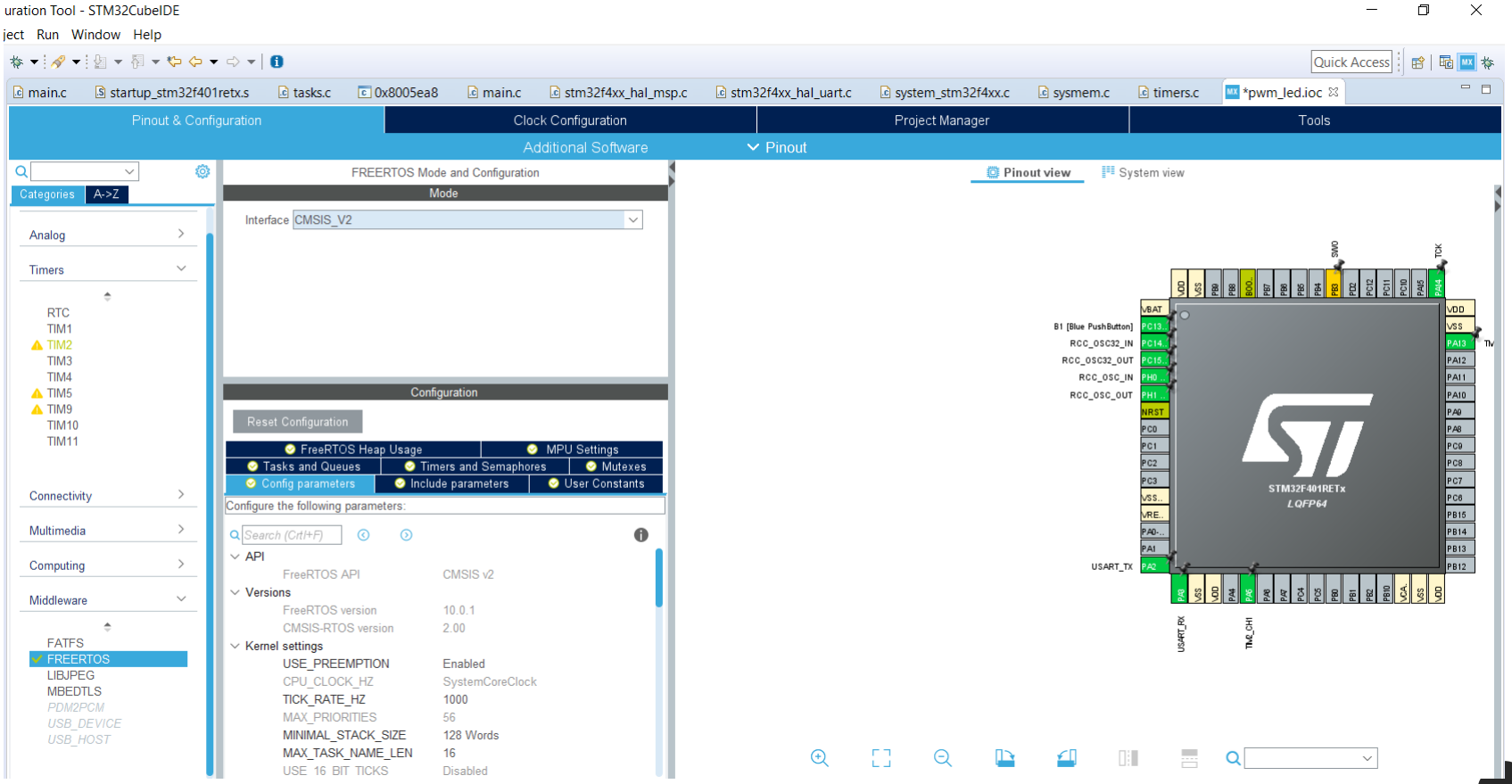
Adc: canale 0 e 84 cicli a conversione



NUCLEO F401RE – ESEMPIO

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

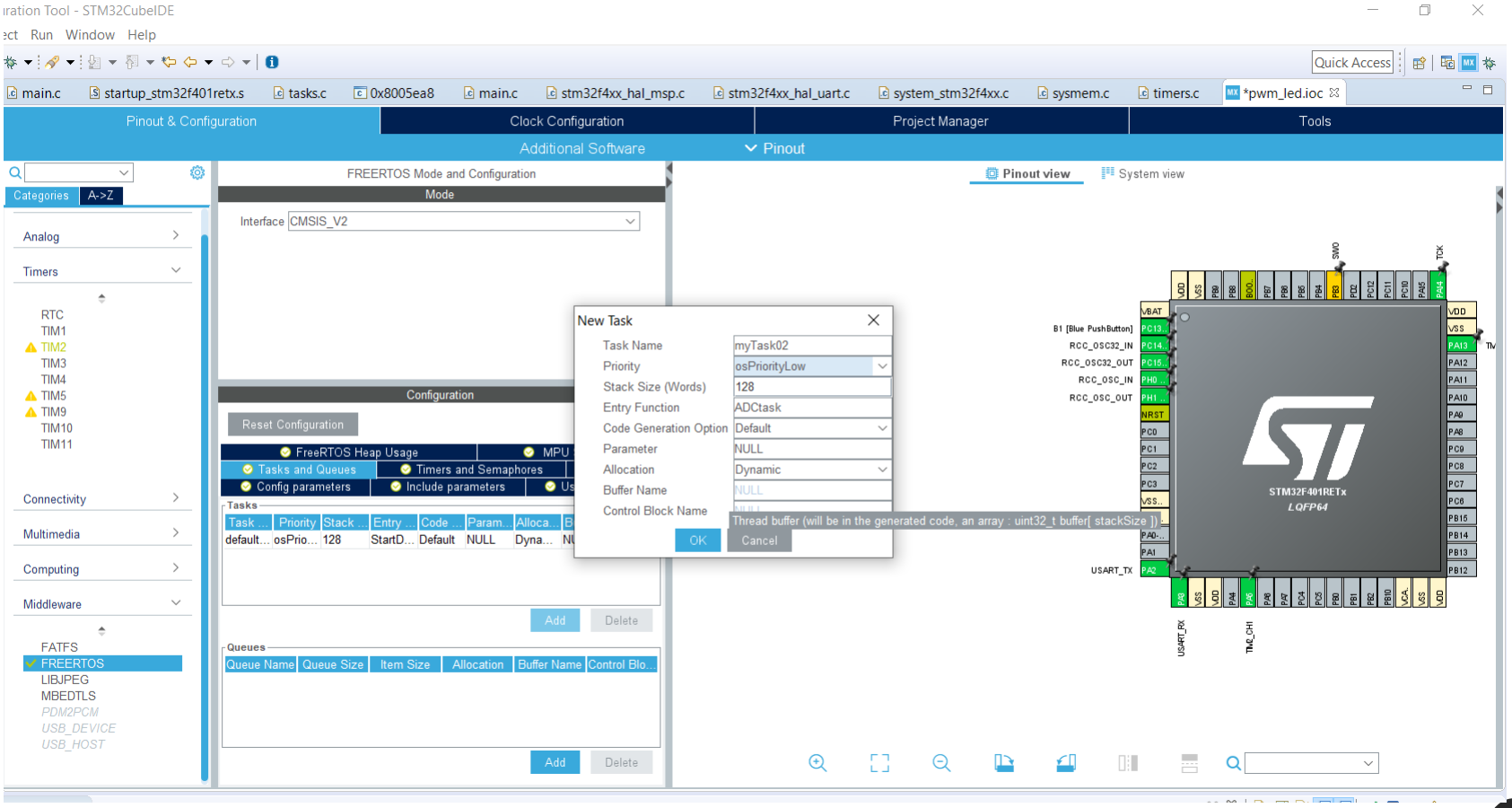
Middleware - Freertos



NUCLEO F401RE – ESEMPIO

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

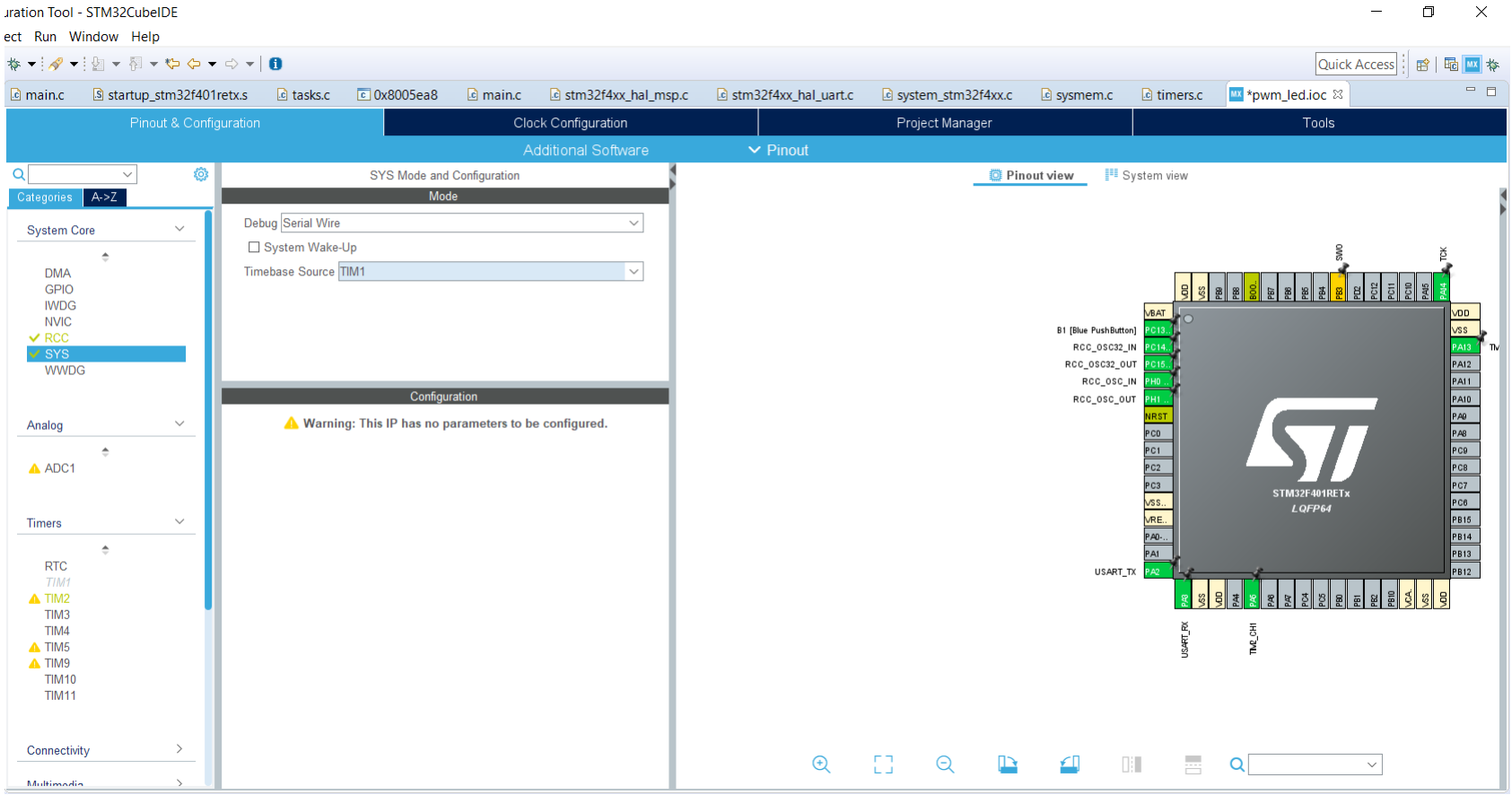
Freertos – creazione task



NUCLEO F401RE – ESEMPIO

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

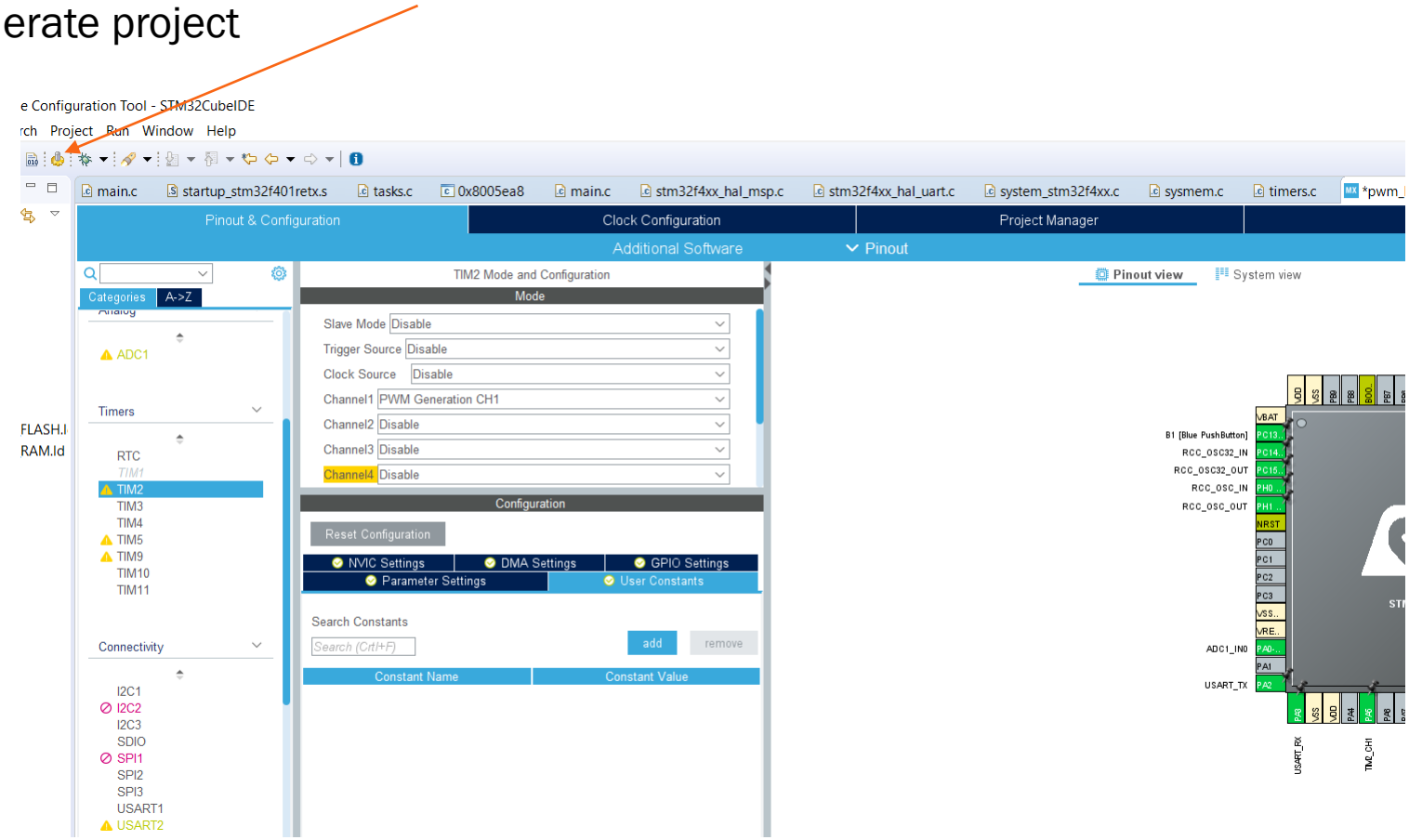
Freertos – systick



NUCLEO F401RE – ESEMPIO

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

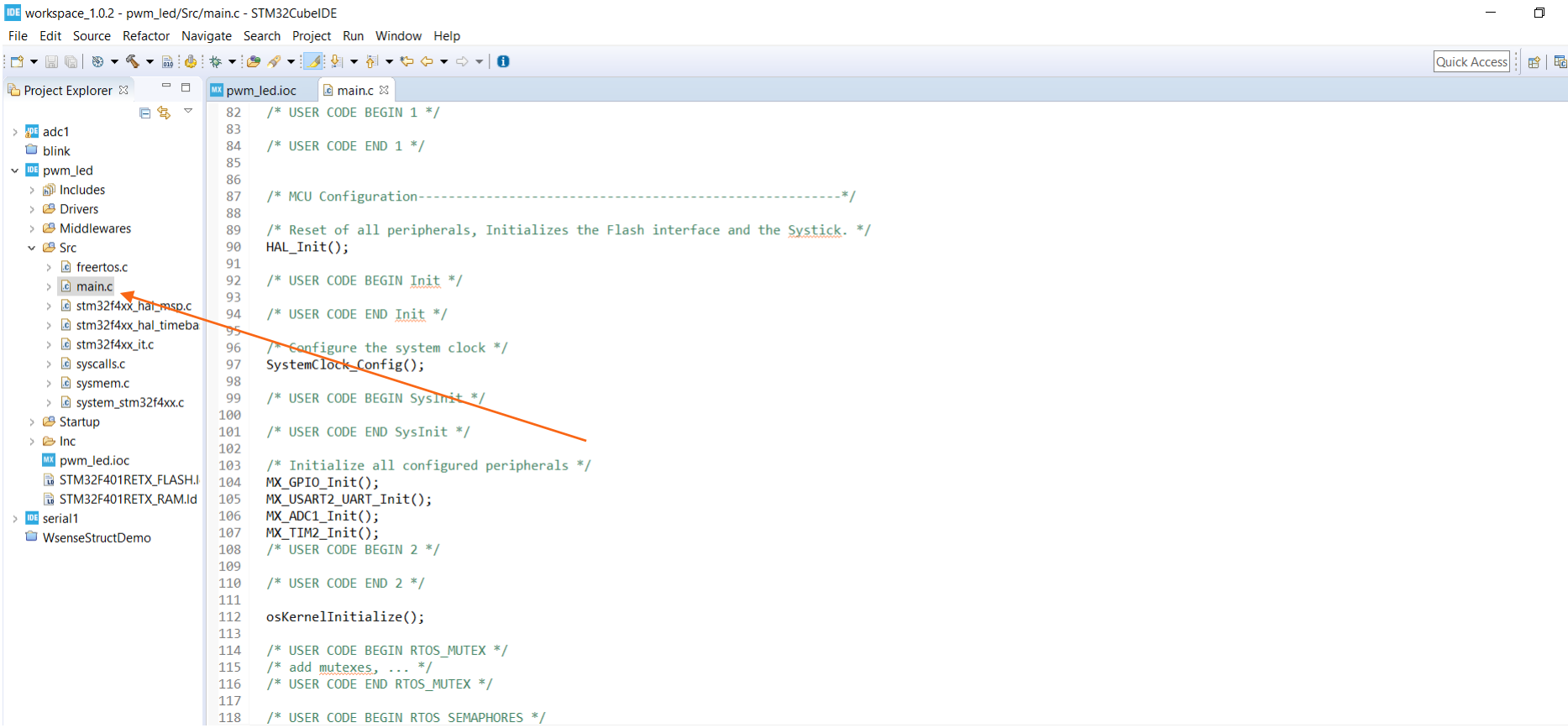
Generate project



NUCLEO F401RE – ESEMPIO

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

Aprire main.c



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

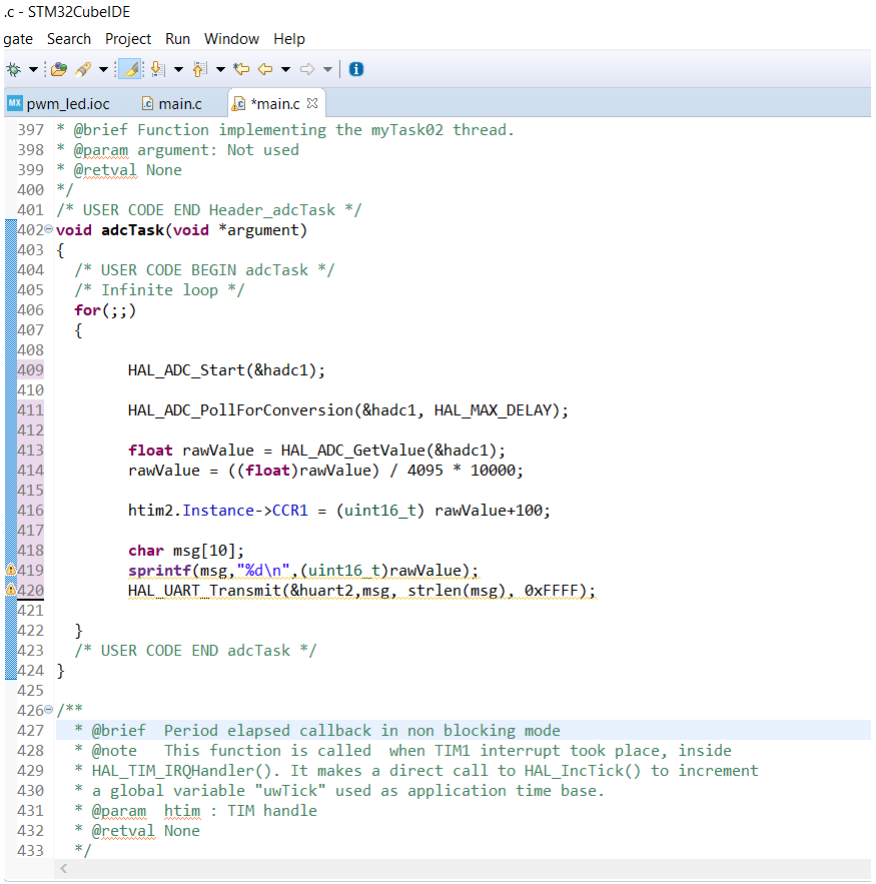
```
136   };
137   defaultTaskHandle = osThreadNew(StartDefaultTask, NULL, &defaultTask_attributes);
138
139   /* definition and creation of myTask02 */
140   const osThreadAttr_t myTask02_attributes = {
141     .name = "myTask02",
142     .priority = (osPriority_t) osPriorityLow,
143     .stack_size = 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
```

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

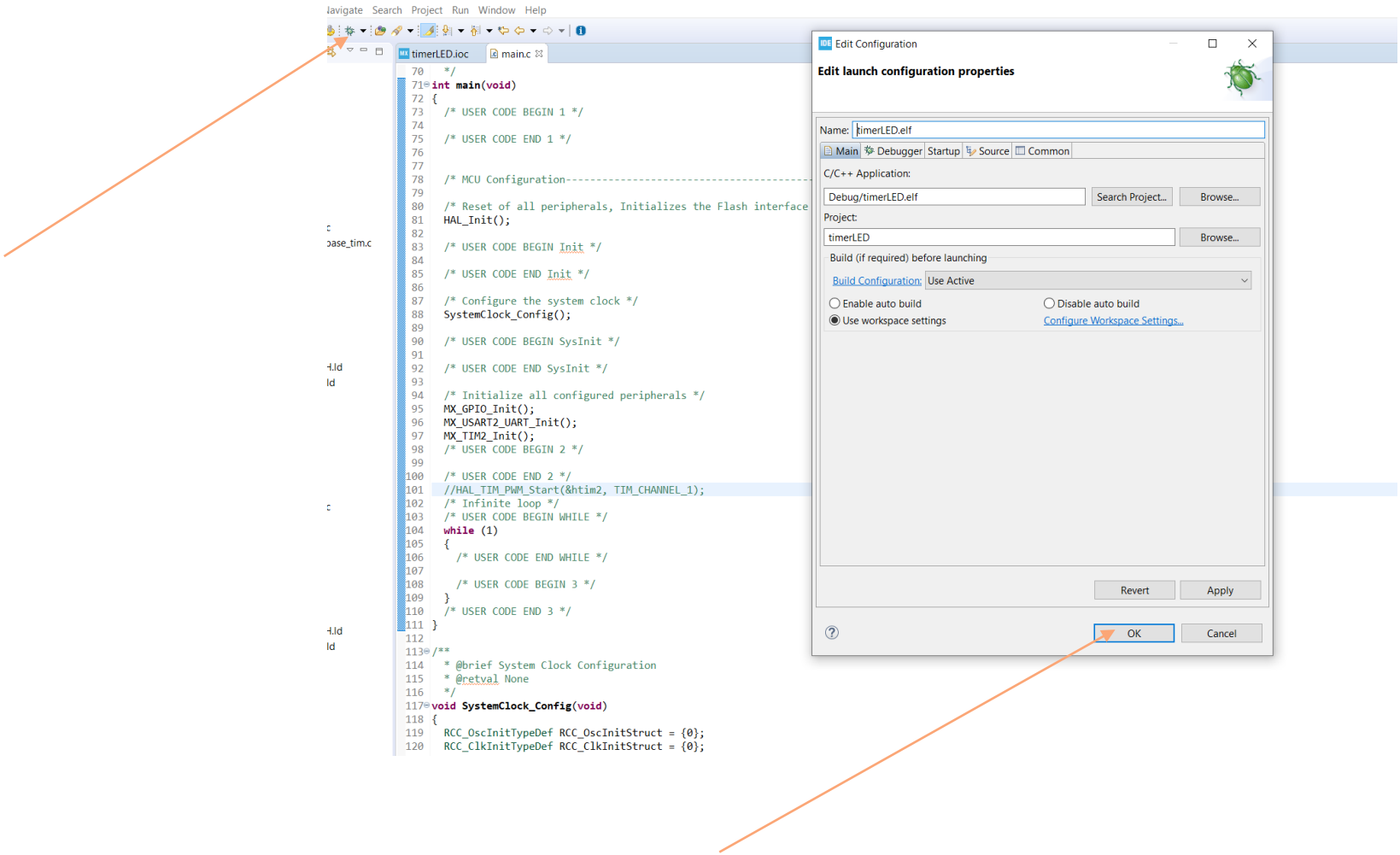



```
397 * @brief Function implementing the myTask02 thread.
398 * @param argument: Not used
399 * @retval None
400 */
401 /* USER CODE END Header_adcTask */
402 void adcTask(void *argument)
403 {
404   /* USER CODE BEGIN adcTask */
405   /* Infinite loop */
406   for(;;)
407   {
408
409         HAL_ADC_Start(&hadc1);
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

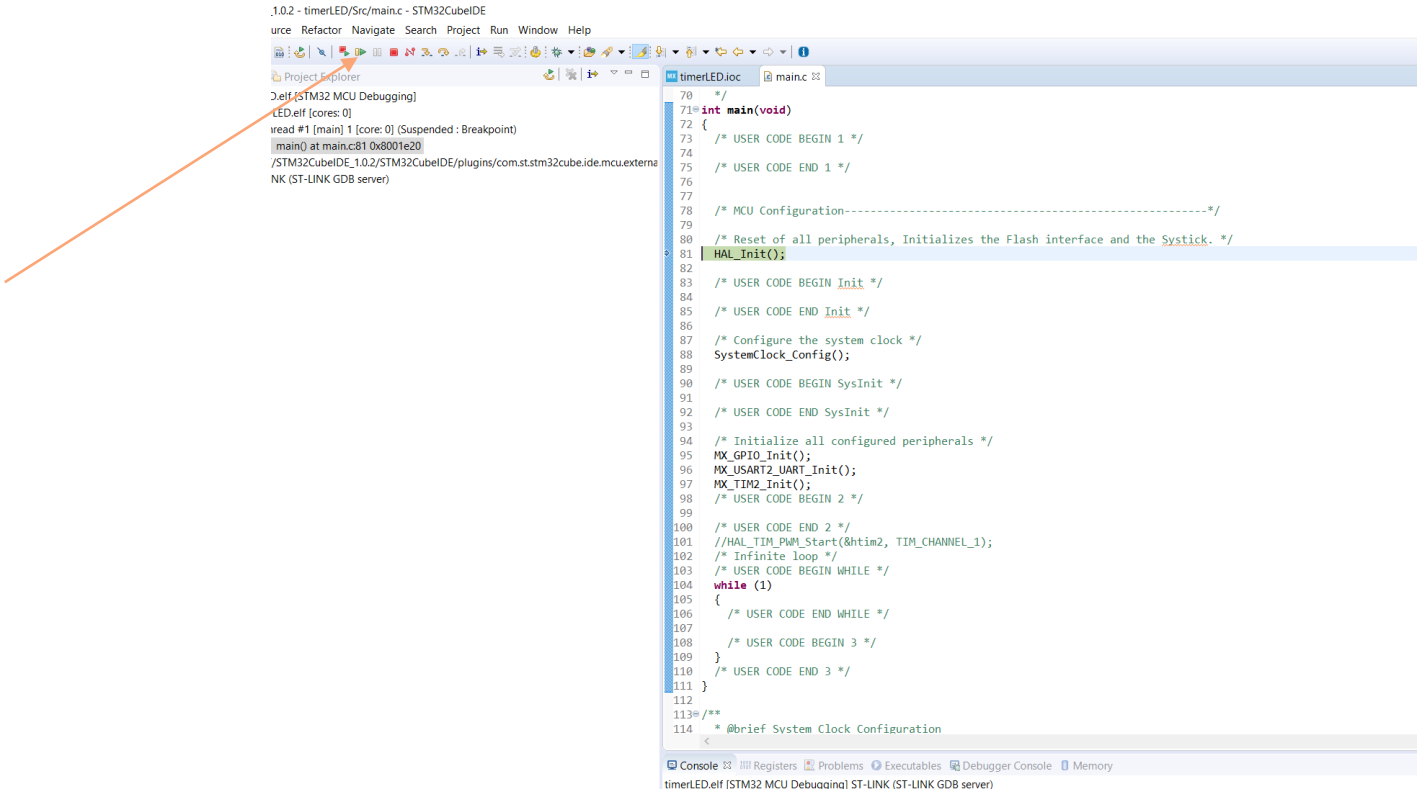
Programmare il target



NUCLEO F401RE – ESEMPIO

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

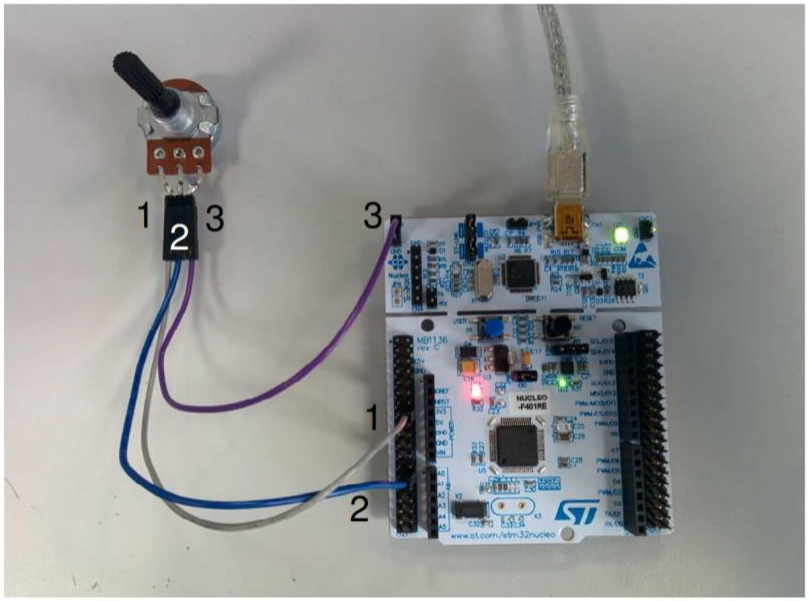
Avviare il target



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