

# Programmazione di sistemi multicore

**Michele Martinelli**

*Michele.martinelli@uniroma1.it*



# Correzione esonero

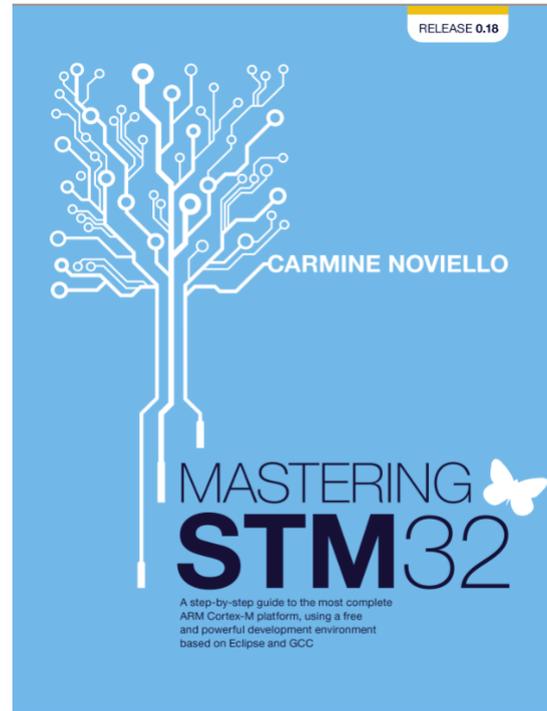
## Programma di oggi:

- Correzione esonero (i risultati saranno pubblicati a breve)
- Come si realizzano PCB (perché facciamo prototipi su breadboard? E perché serve Arduino?)
- Protocolli seriali di comunicazione (UART, SPI, I2C)
- Porte e registri
- Esempi e esercizi semplici su NUCLEO board (PWM con LED + FreeRTOS)

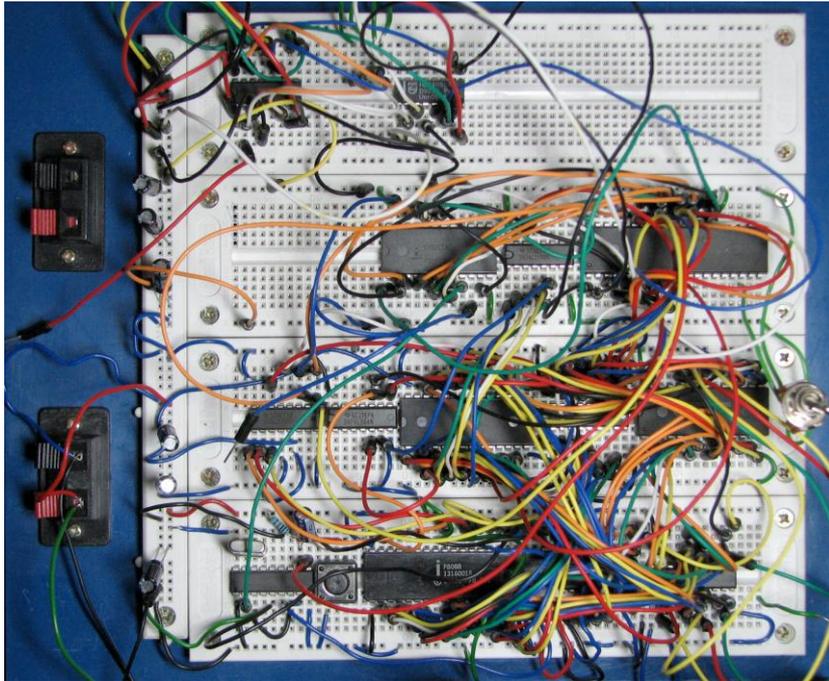
## Programma di martedì prossimo

- Teoria FreeRTOS (timer, interrupt)
- Esercizi su NUCLEO board (FreeRTOS + motori DC, servomotori)
- Revisione compiti

# PER CHI VOLESSE APPROFONDIRE LA PARTE DI ARM

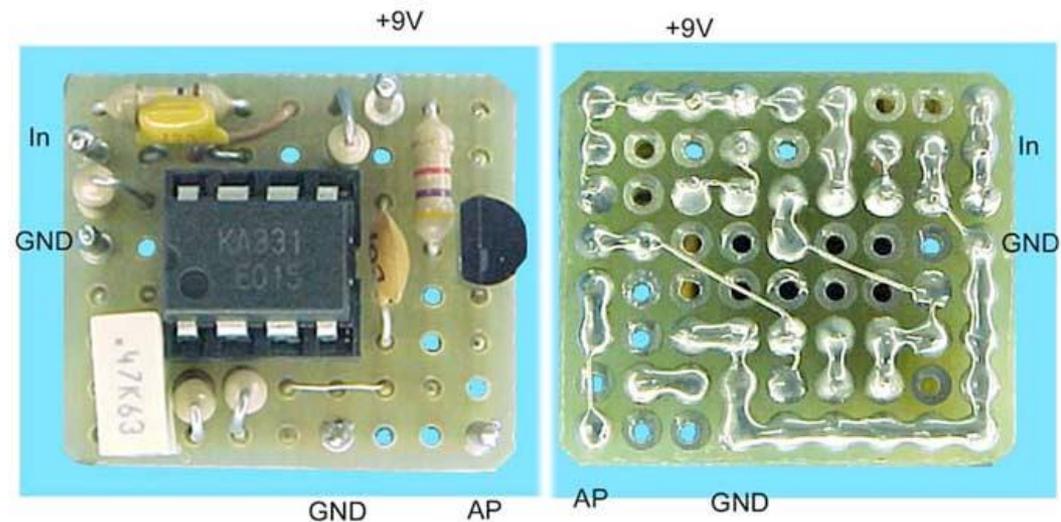


# Realizzare PCB – fase 1



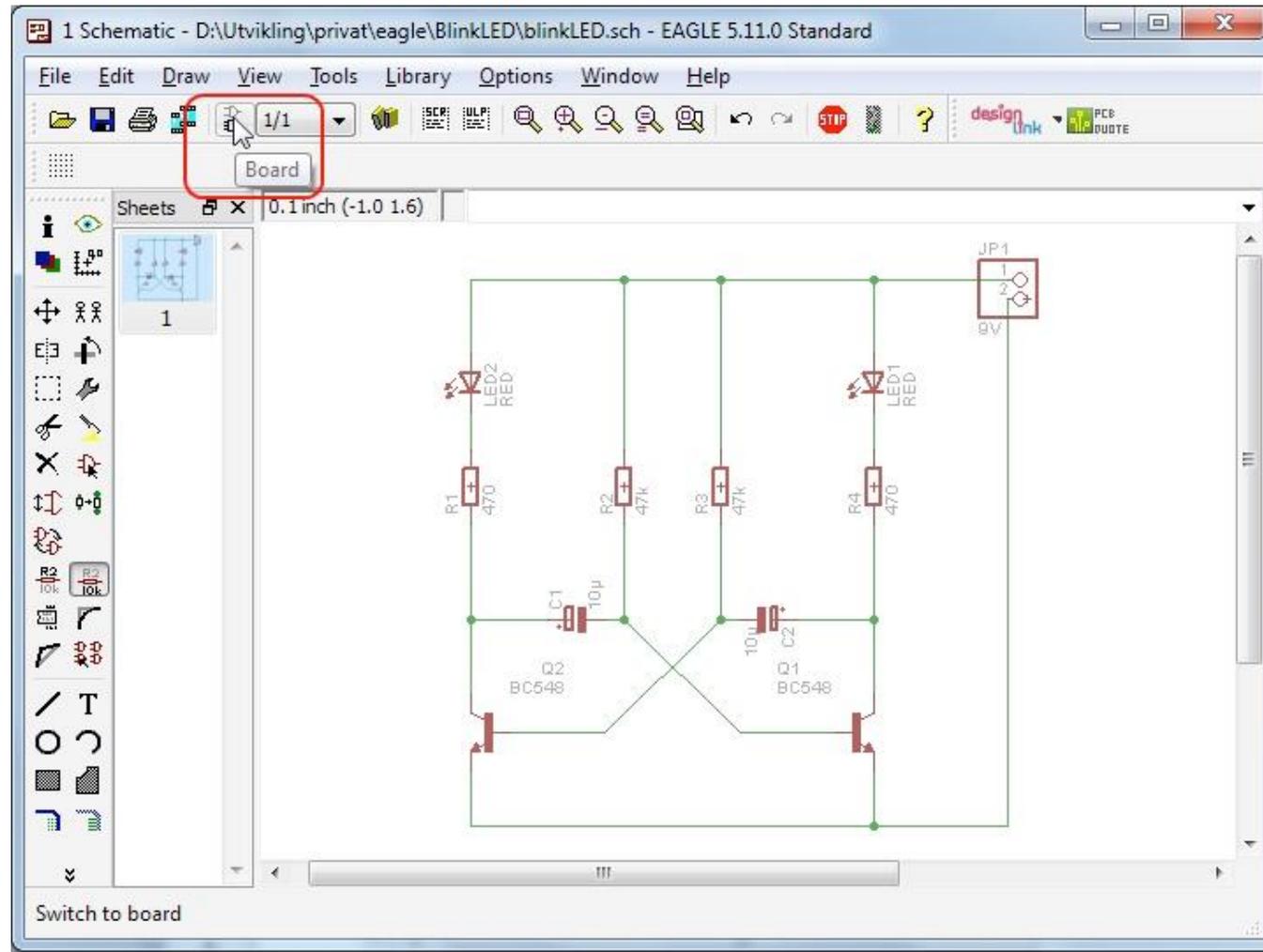
Progettare e testare il circuito su breadboard o millefori

Questo permette di sviluppare e testare software e hardware



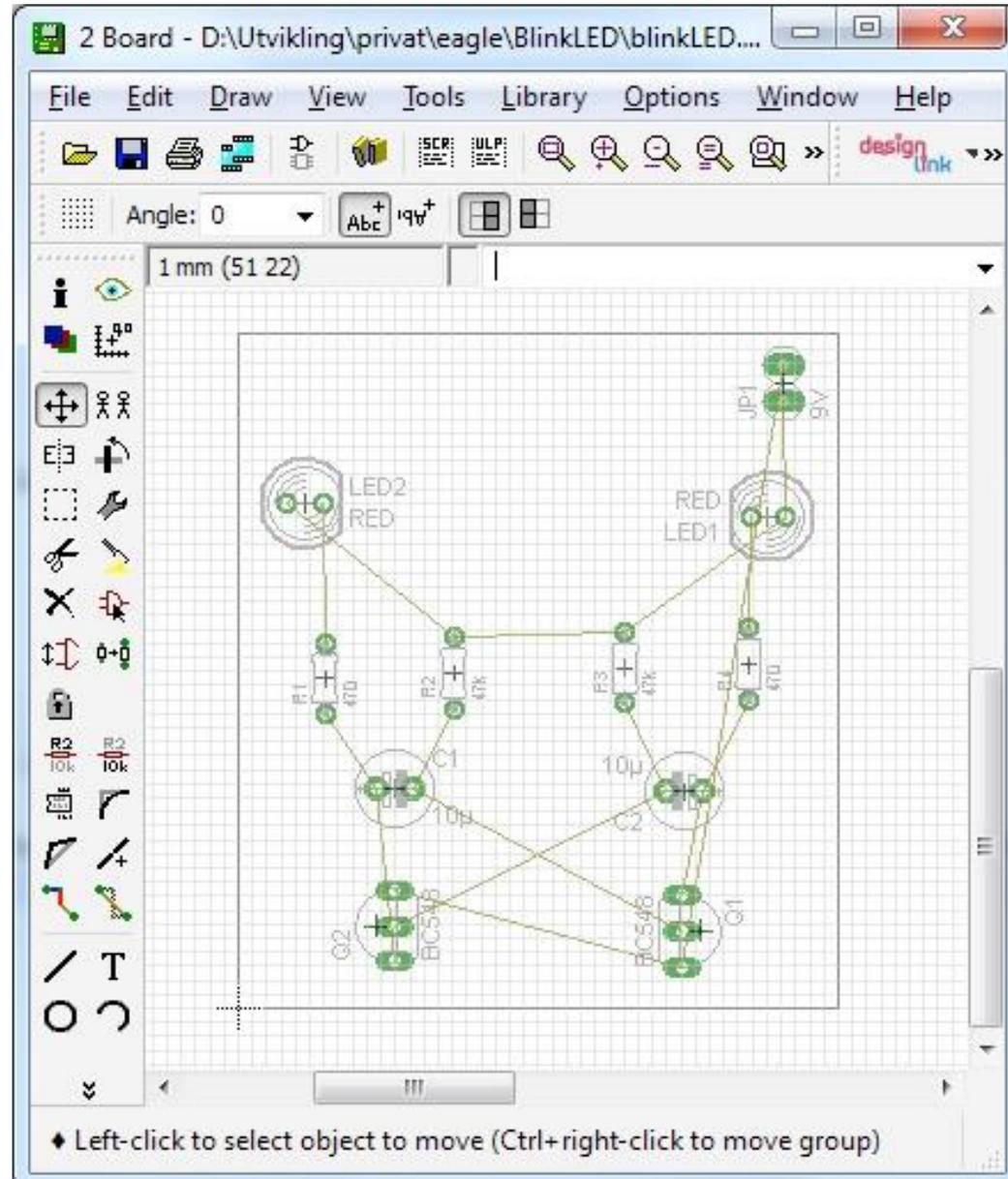
# Realizzare PCB – fase 2

Disegnare lo schematico al cad



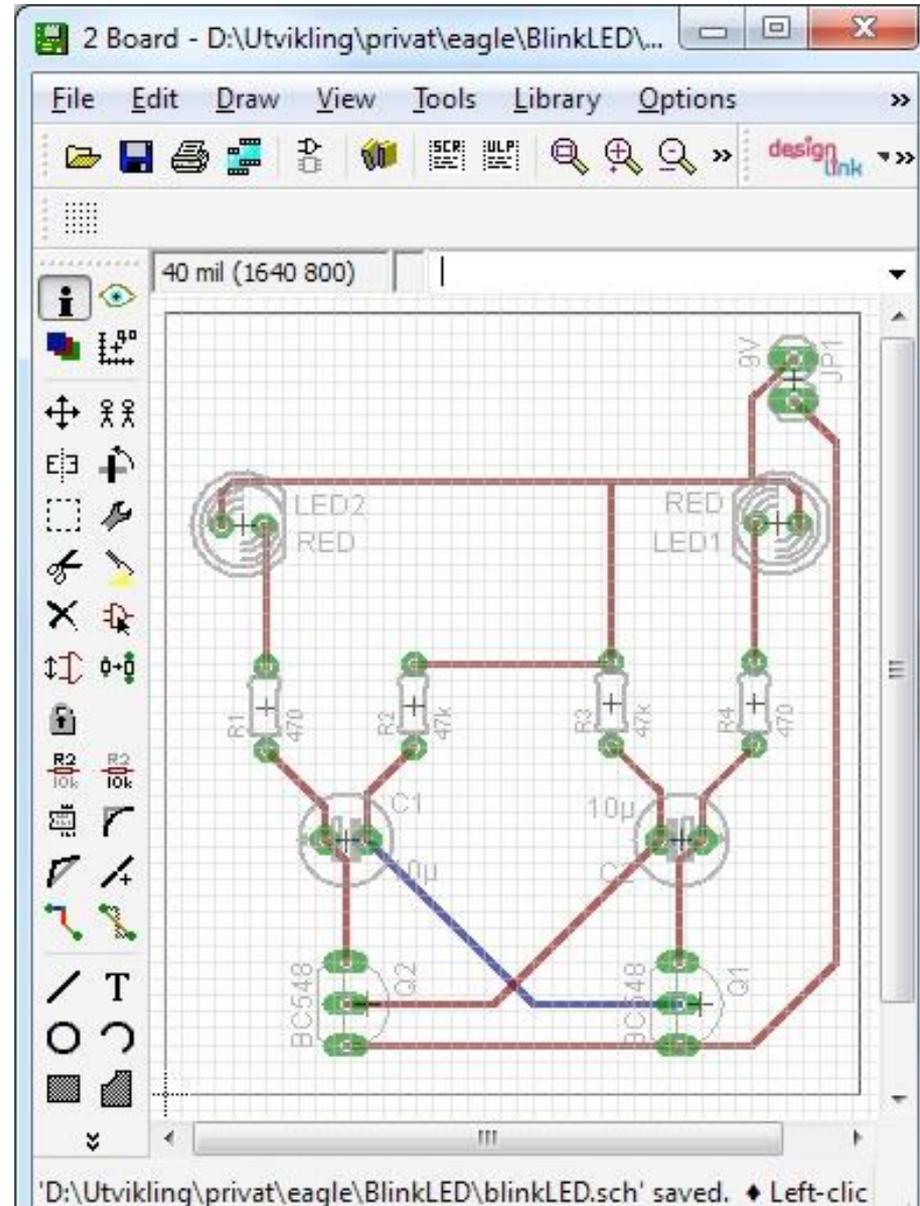
# Realizzare PCB – fase 3

Disegnare la board al cad



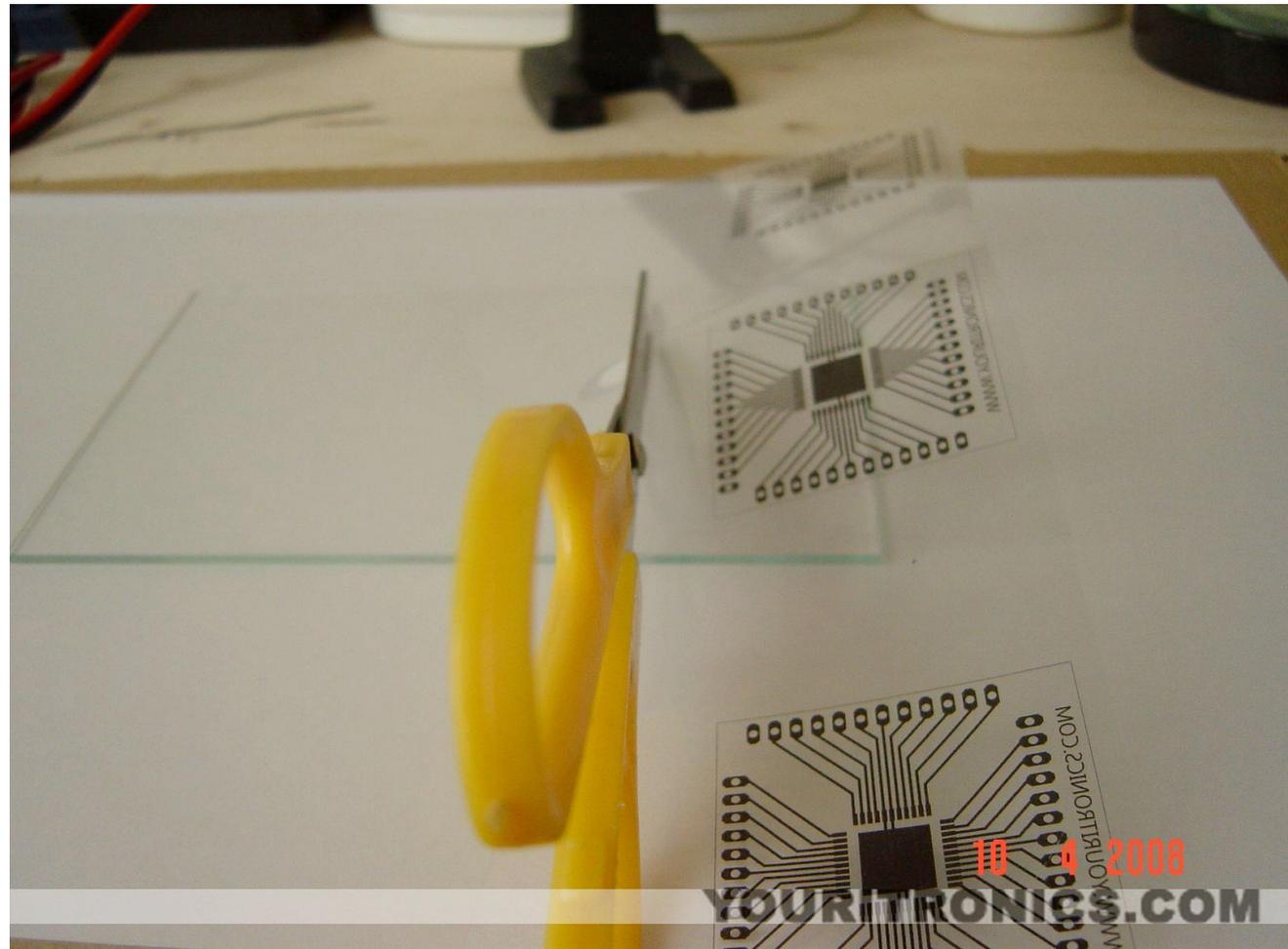
# Realizzare PCB – fase 4

Routare i componenti al cad



# Realizzare PCB – fase 5

Stampare su carta trasparente – MASTER (circuito specchiato)



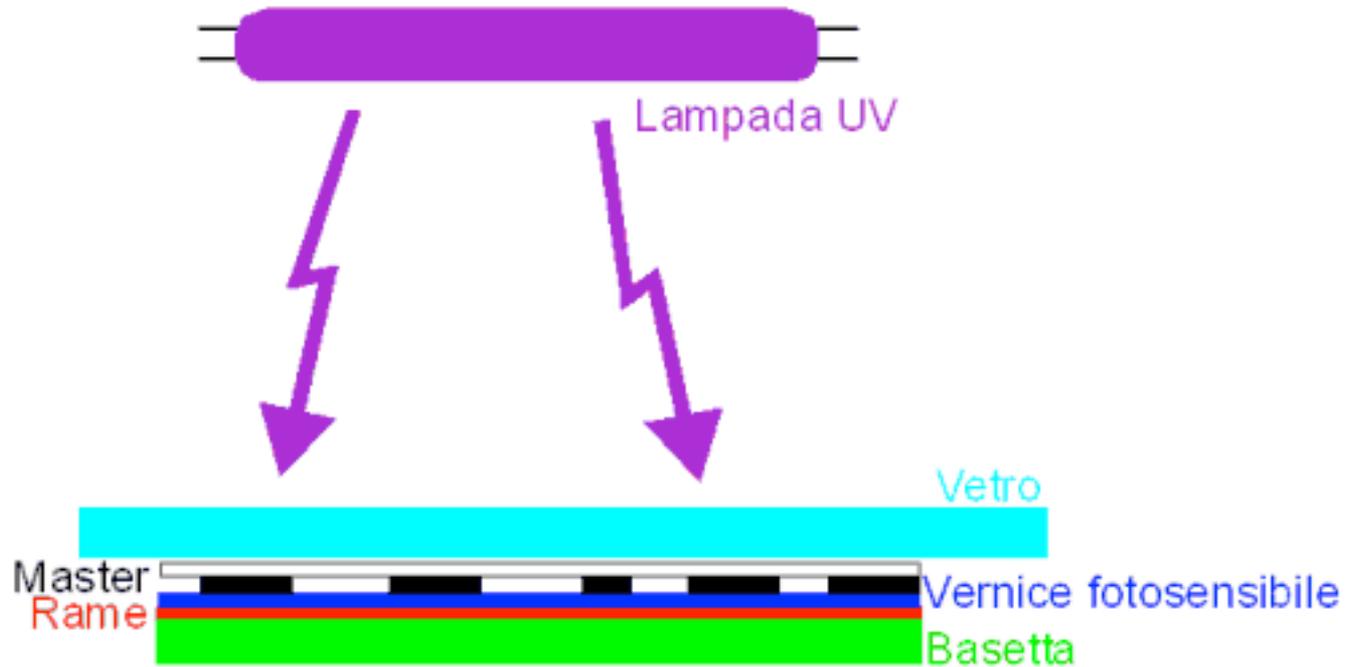
# Realizzare PCB – fase 6

Utilizzando delle basette fotosensibili



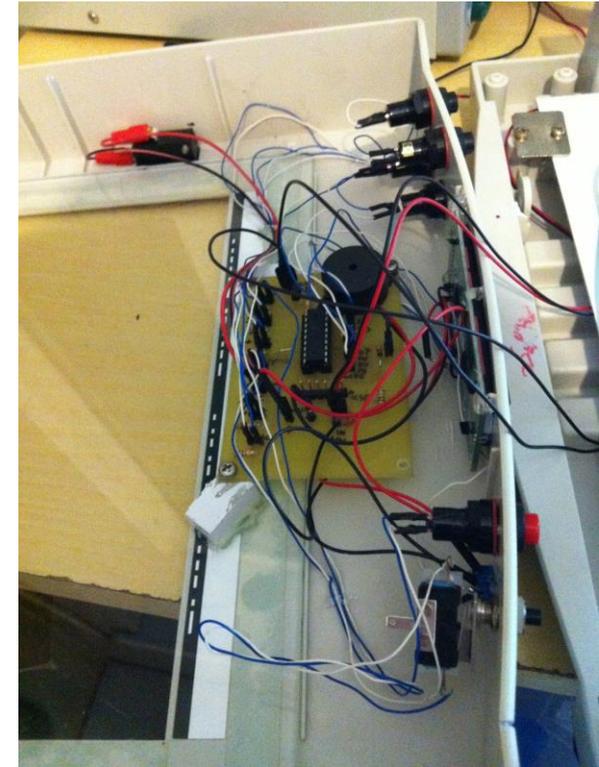
# Realizzare PCB – fase 7

Lo sviluppo con il bromografo



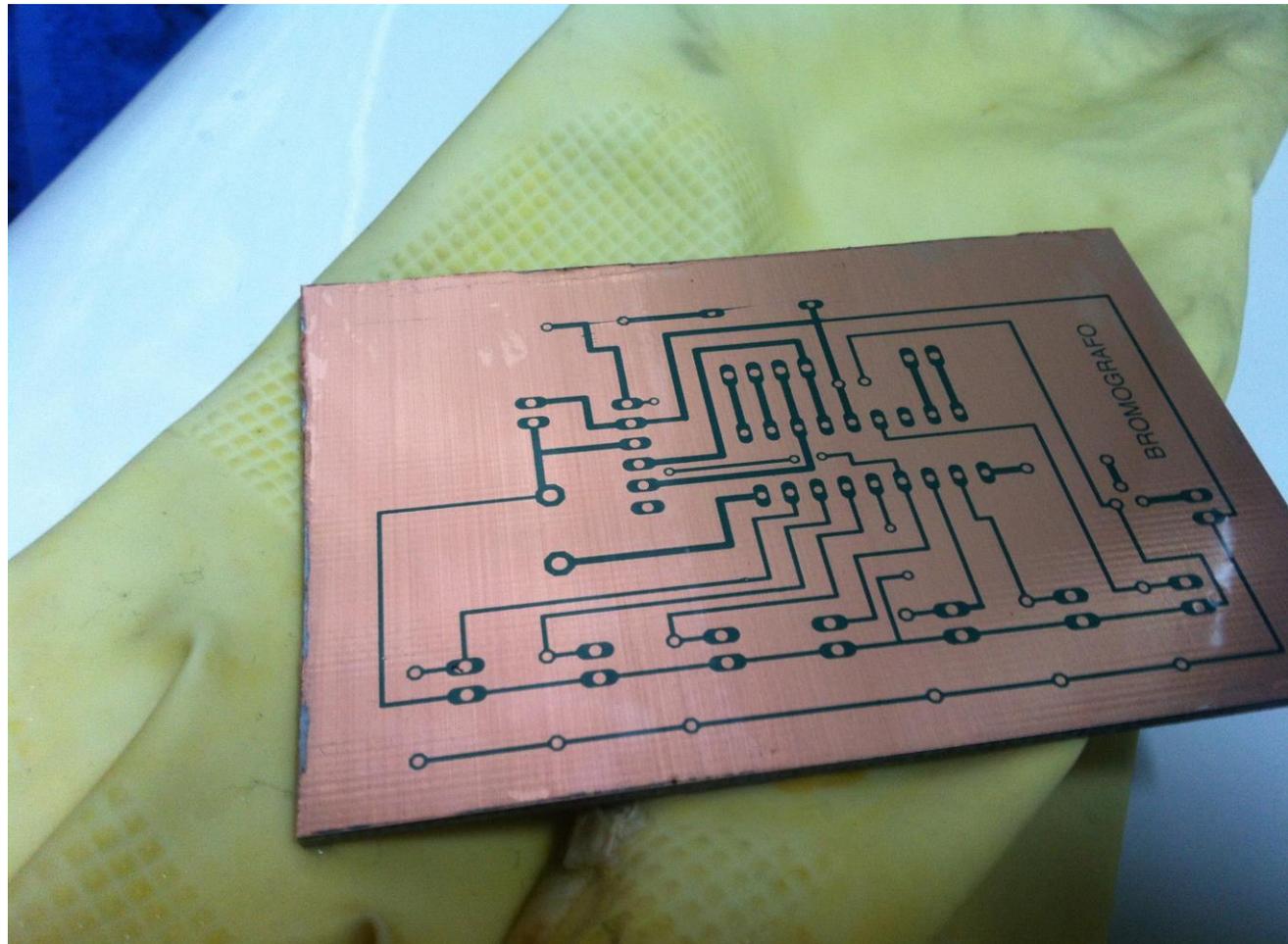
# Realizzare PCB – fase 8

Parentesi... il bromografo



# Realizzare PCB – fase 9

Dopo il lavaggio con la soda caustica



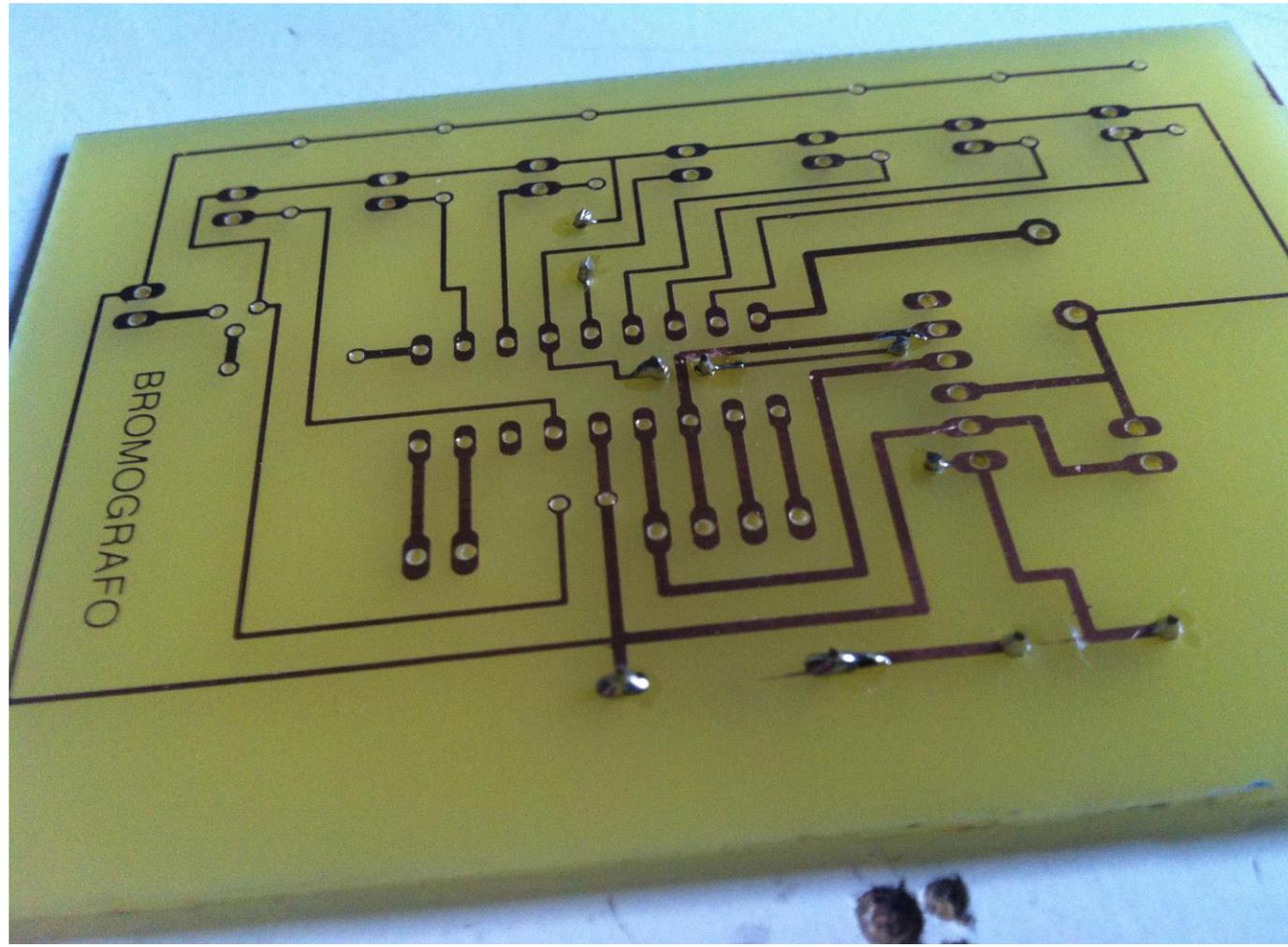
# Realizzare PCB – fase 10

Circuito nell'acido



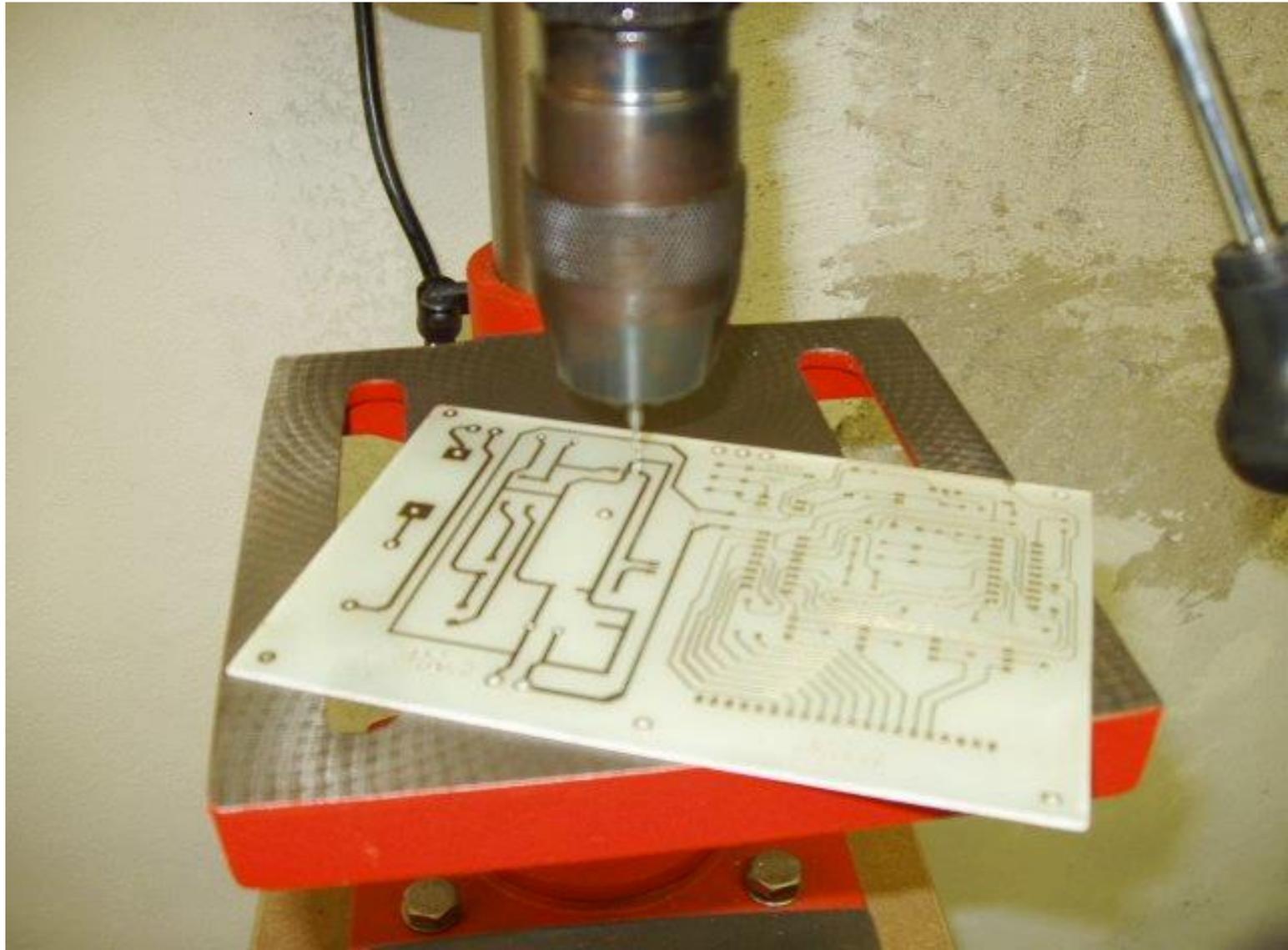
# Realizzare PCB – fase 11

Dopo il lavaggio



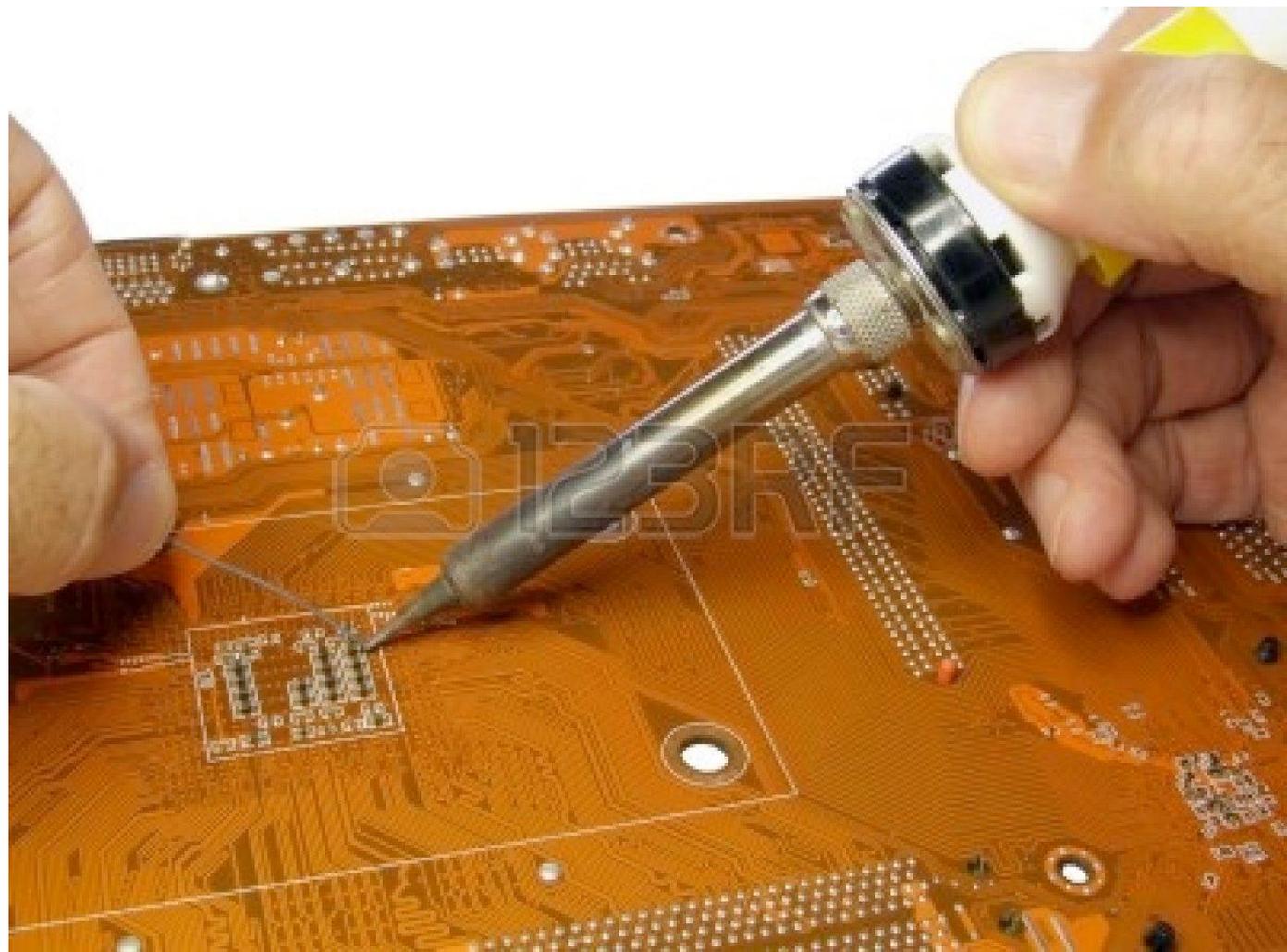
# Realizzare PCB – fase 12

foratura

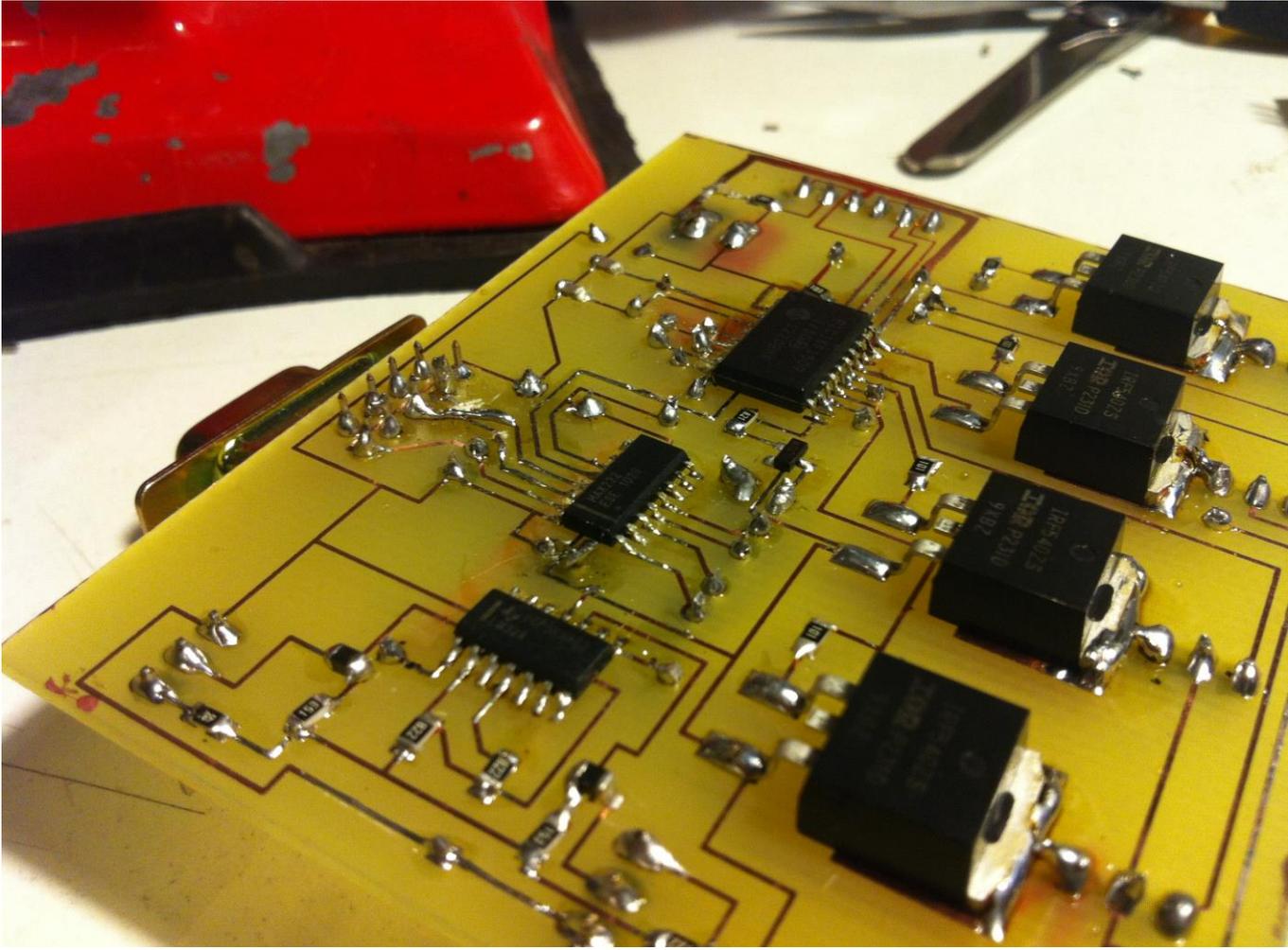


# Realizzare PCB – fase 13

saldatura



Alla fine



# 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);

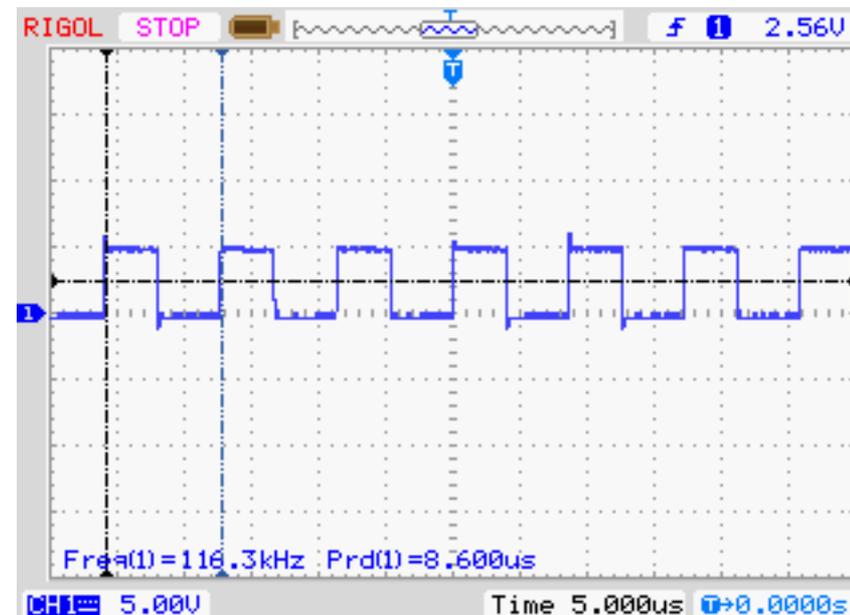
    //out = portOutputRegister(port);
    out =(volatile uint8_t *) (pgm_read_word( port_to_output_PGM + pin ));

    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;
```

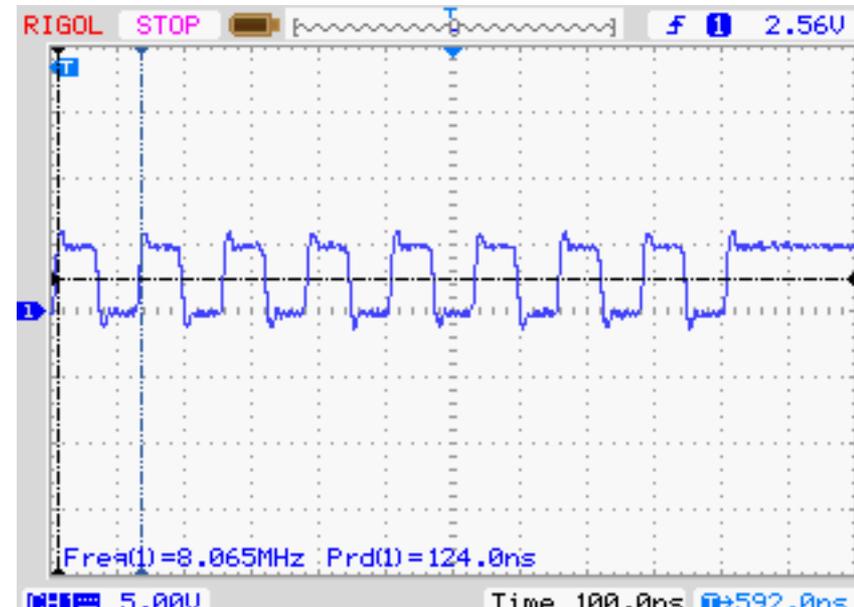
Importante: inizializzazione delle variabili

A = B00000101 -> B binario

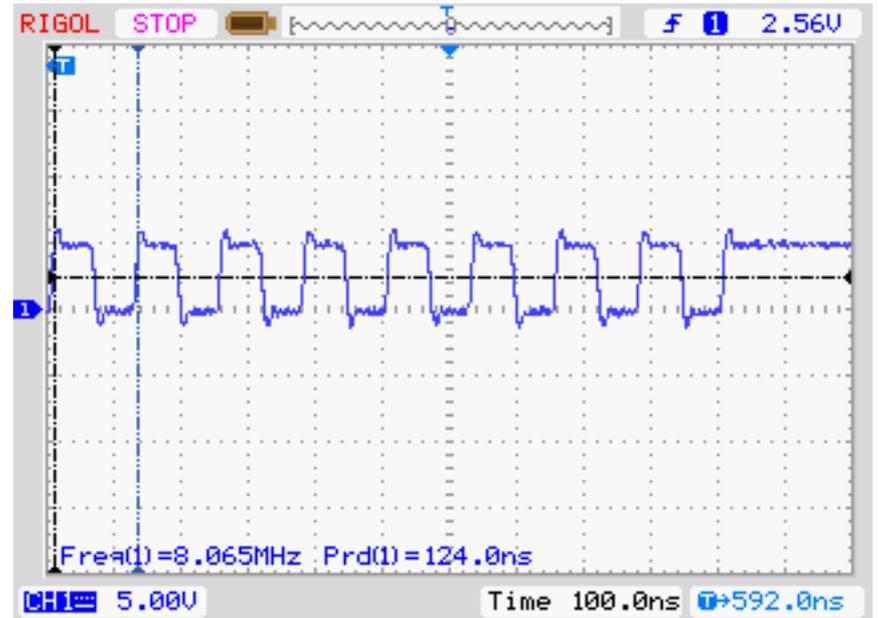
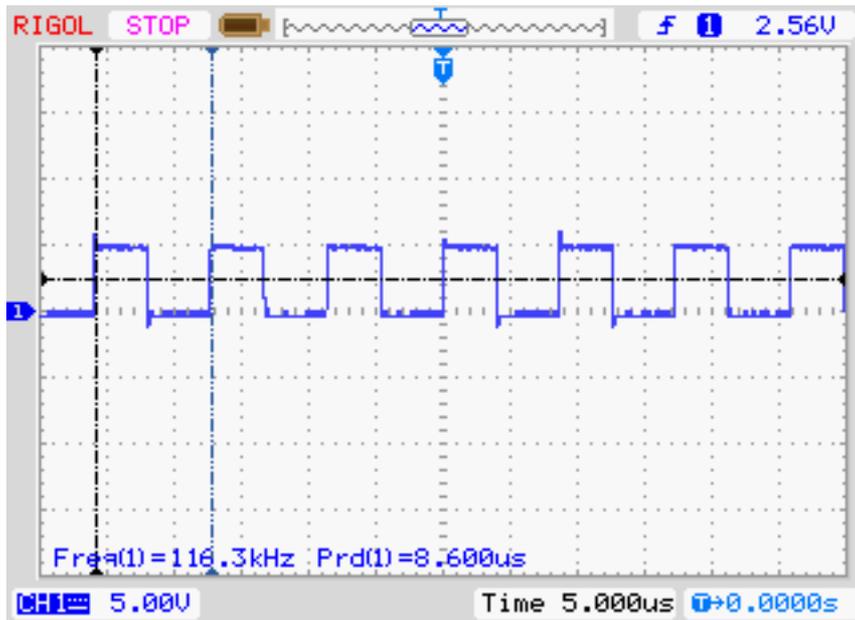
A = 0x05 -> 0x esadecimale

A = 5 -> decimale

Risultato



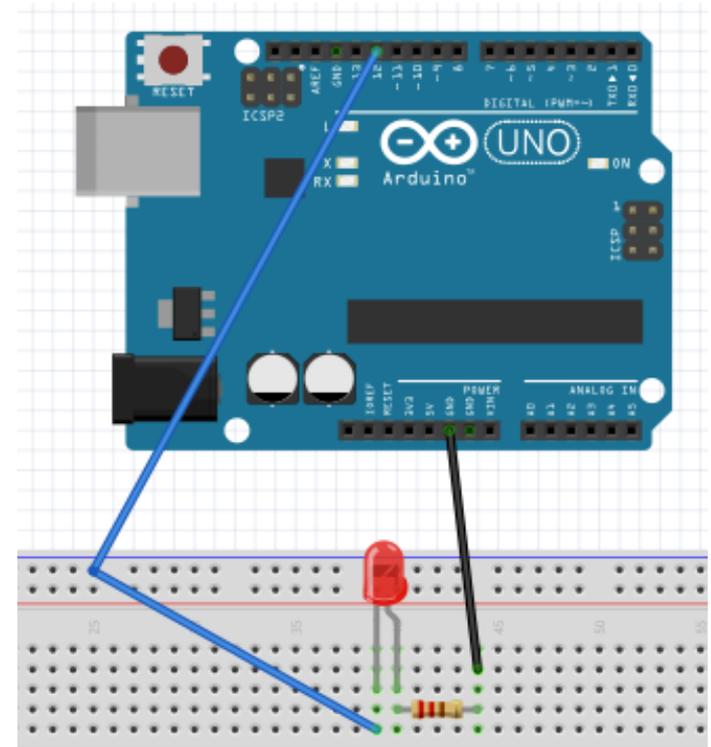
# Confrontando...



# ACCENDERE UN LED CON PORT MANIPULATION

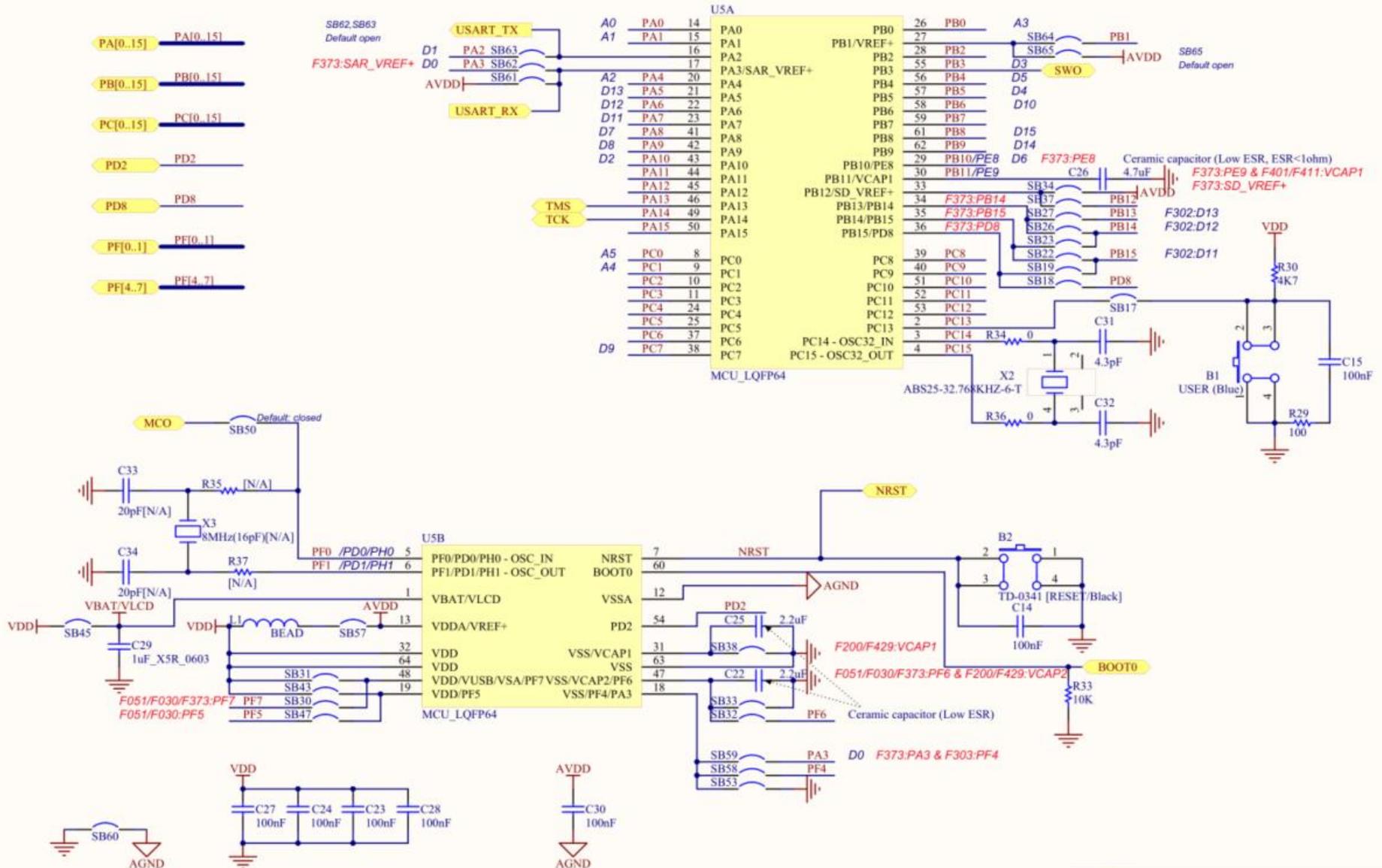
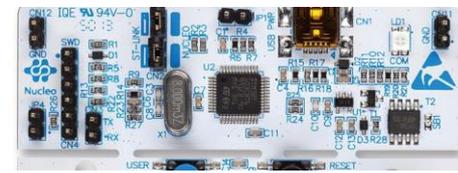
```
void setup(){
  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(5, OUTPUT);
}

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);
}
```

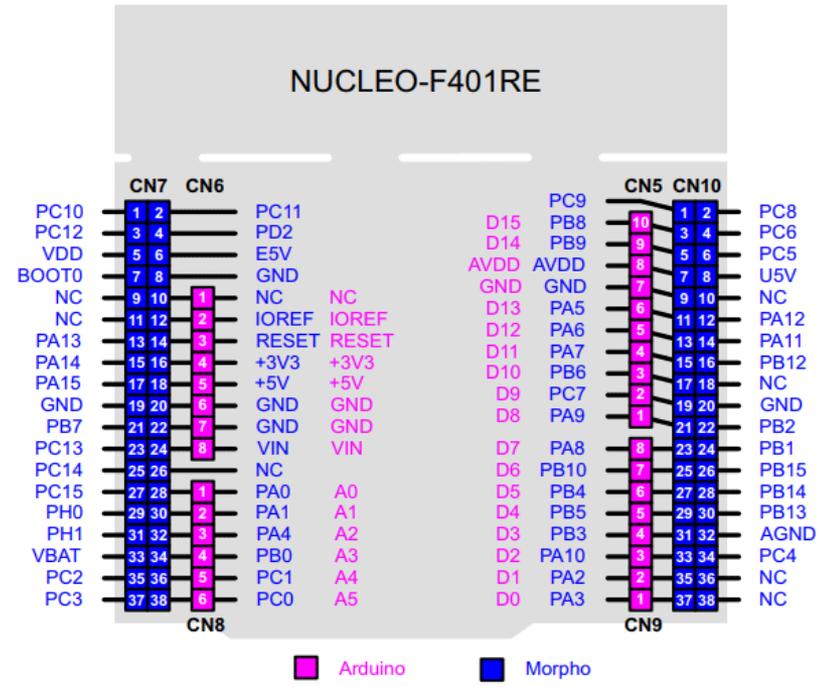
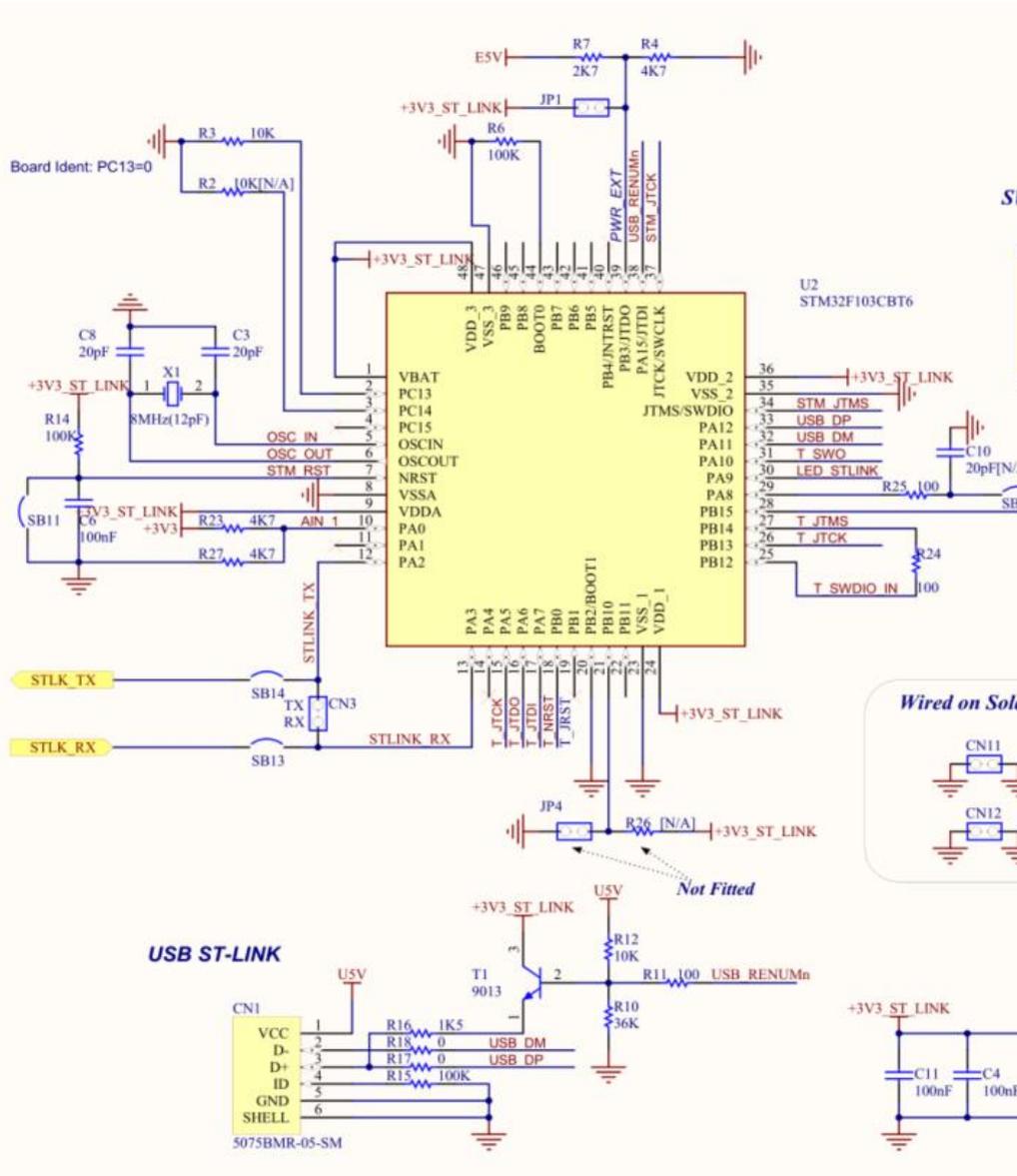
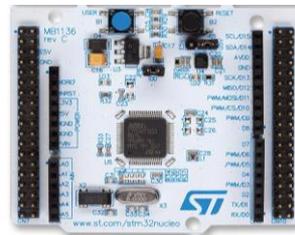




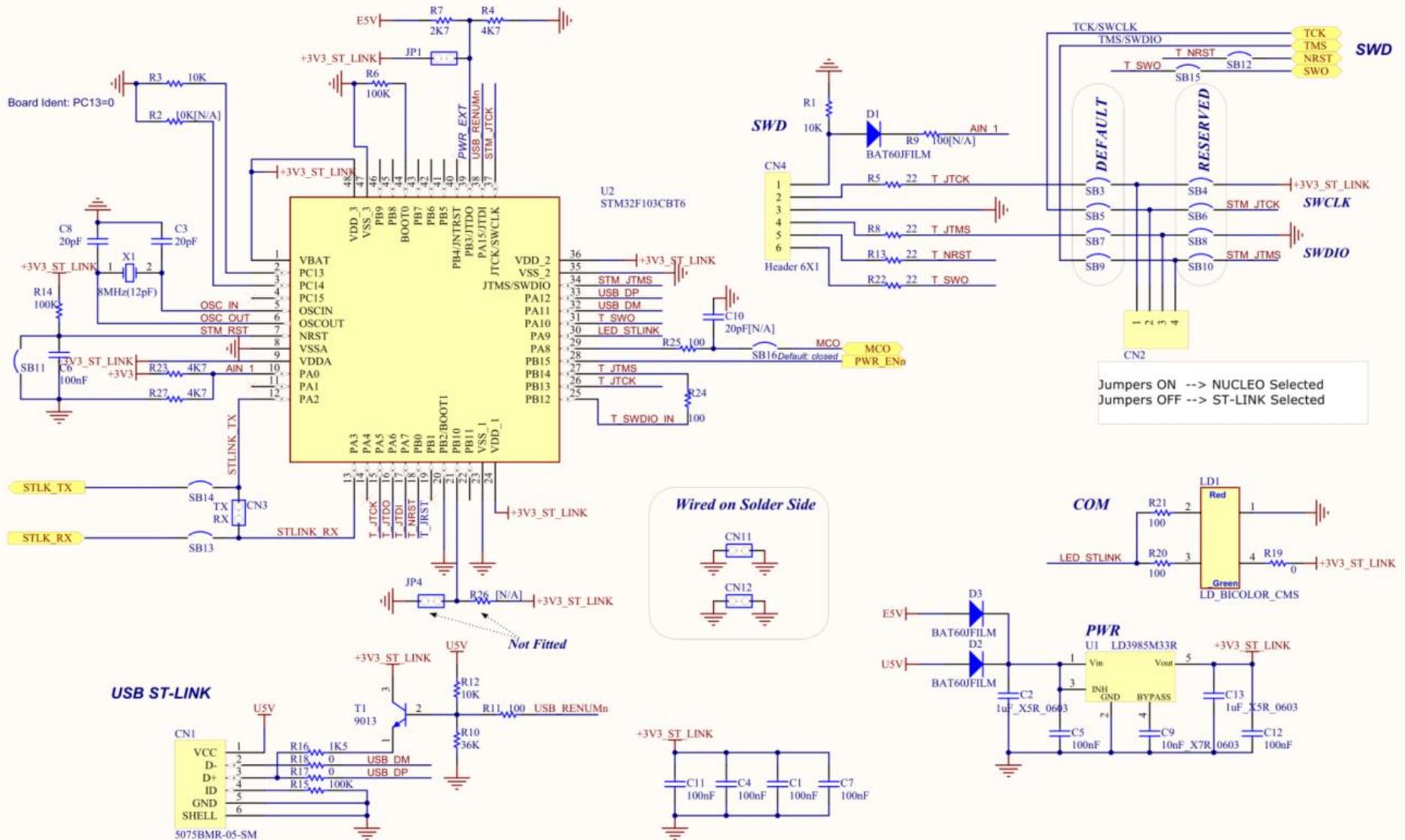
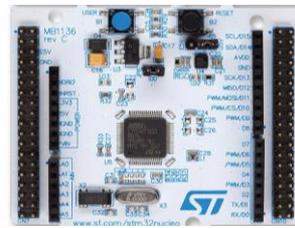
# NUCLEO F401RE - SCHEMA



# NUCLEO F401RE - SCHEMA



# NUCLEO F401RE - SCHEMA



# NUCLEO F401RE – ESEMPIO

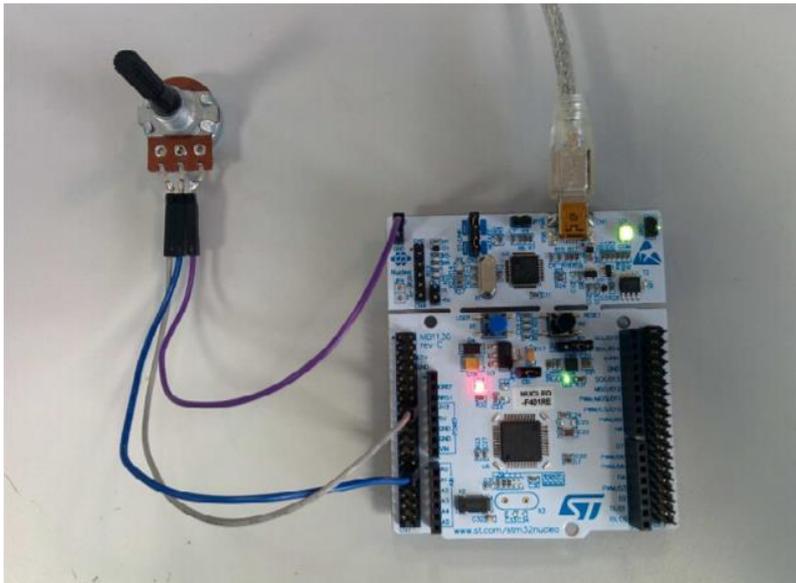
USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

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

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 "Part Number Search" field with "NUCLEO-F401RE" entered, and various filter categories 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 Selection:** The "Board Selector" tab is active, showing a search result for "NUCLEO-F401RE". The result includes a product image, a description: "STMicroelectronics NUCLEO-F401RE Board Support and Examples", and a status of "ACTIVE" (Product is in mass production). The unit price is listed as 13.0 and the mounted device is "STM32F401RETx".
- Boards List:** A table showing the selected board:

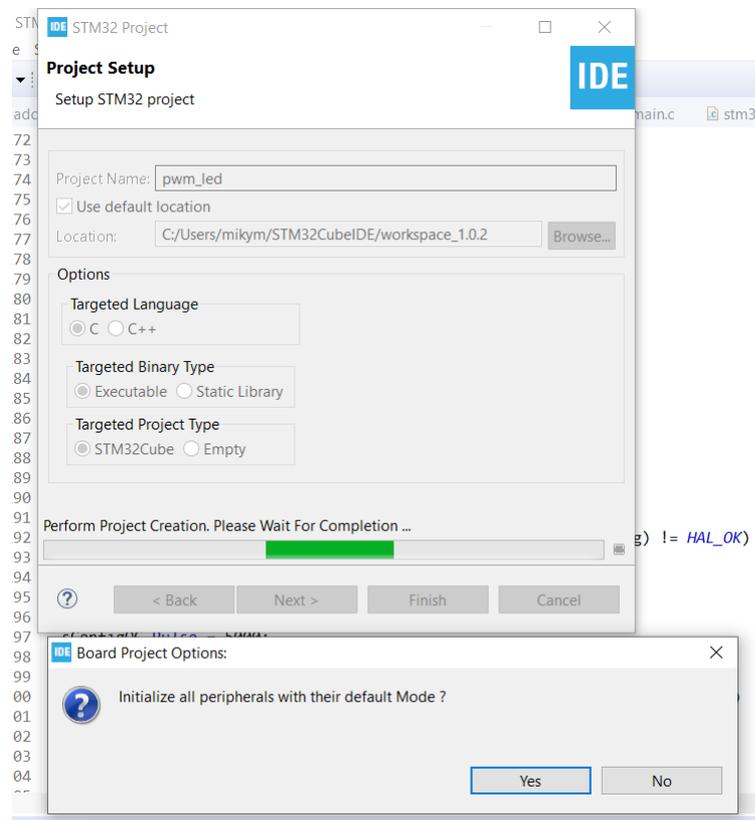
| * | Overview | Part No       | Type     | Marketing Status | Unit Price (US\$) | Mounted Device |
|---|----------|---------------|----------|------------------|-------------------|----------------|
| ☆ |          | NUCLEO-F401RE | Nucleo64 | Active           | 13.0              | STM32F401RETx  |

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

# NUCLEO F401RE – ESEMPIO

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

Assegnare nome – e poi yes



# NUCLEO F401RE – ESEMPIO

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

The screenshot displays the STM32CubeIDE interface for configuring a TIM2 timer. The main window is titled "workspace\_1.0.2 - Device Configuration Tool - STM32CubeIDE". The "Project Explorer" on the left shows a project named "pwm\_led" with files like "pwm\_led.ioc" and "STM32F401RET\_FLASH.I". The central panel is titled "TIM2 Mode and Configuration" and shows the following settings:

- Mode: TIM2 Mode and Configuration
- 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 "Pinout view" on the right shows the pinout of the Nucleo F401RE. The PA5 pin is highlighted in yellow, and a tooltip shows its configuration: PA5, Reset\_State, ADC1\_IN5, SPI1\_SCK, TIM2\_CH1, TIM2\_ETR, GPIO\_Input, GPIO\_Output, GPIO\_Analog, EVENTOUT, and GPIO\_EXTI5. 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 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 for configuring the TIM2 timer. The 'Parameter Settings' tab is selected, showing the following configuration:

- Mode: PWM Generation CH1
- Channel1: PWM Generation CH1
- Channel2: Disable
- Channel3: Disable
- Channel4: Disable

Under the 'Configuration' section, the 'Parameter Settings' sub-tab is active, showing the following parameters:

- 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 input effect not delayed)
  - Trigger Event Selection: Reset (UG bit from TIMx\_EGR)
- PWM Generation Channel 1:
  - Mode: PWM mode 1
  - Pulse (32 bits value): 5000
  - Fast Mode: Disable
  - CH Polarity: High

The Pinout view on the right shows the STM32F401REx LQFP64 package with pins PA9 and TIM2\_CH1 highlighted, indicating the connection for the timer output.



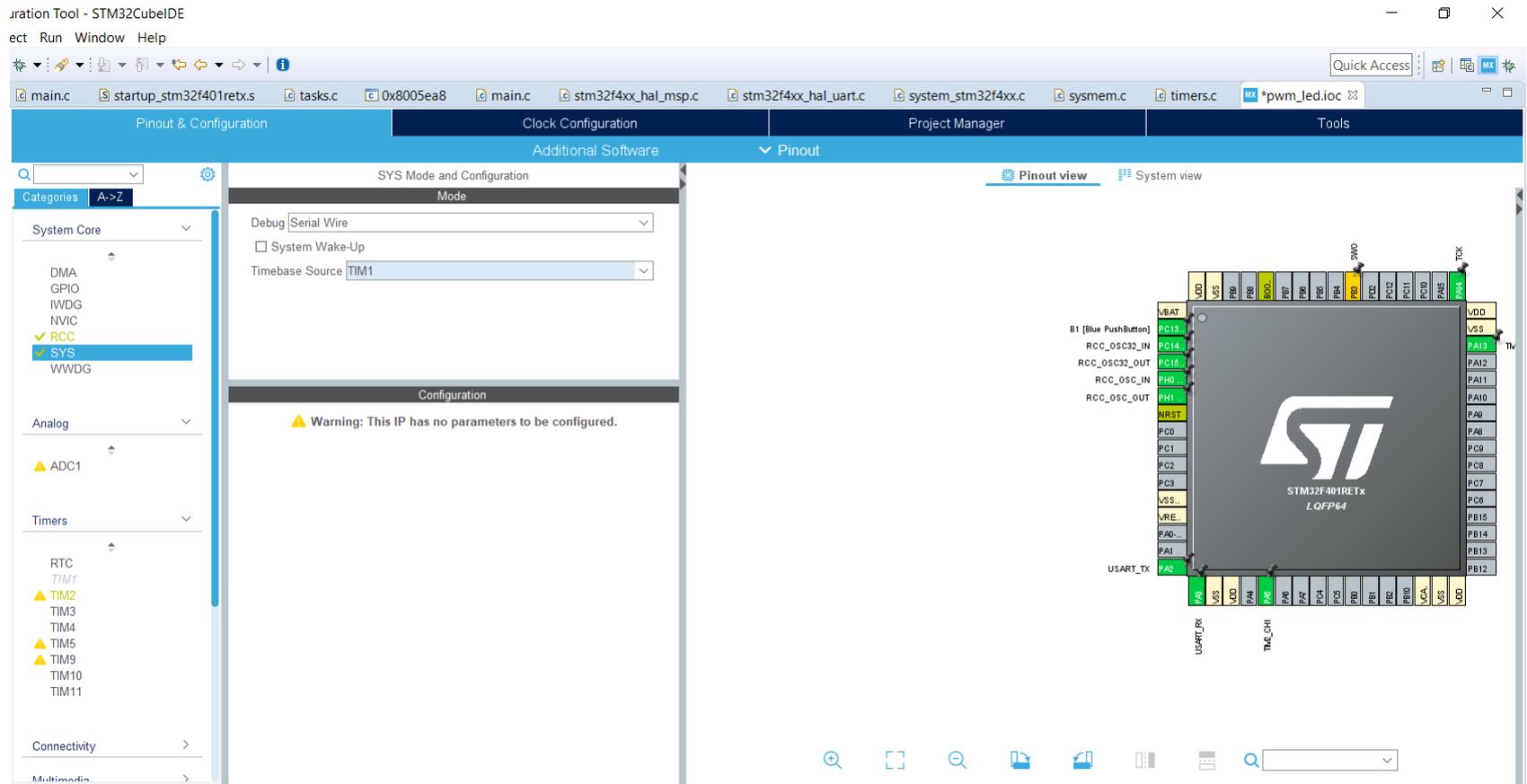




# NUCLEO F401RE – ESEMPIO

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

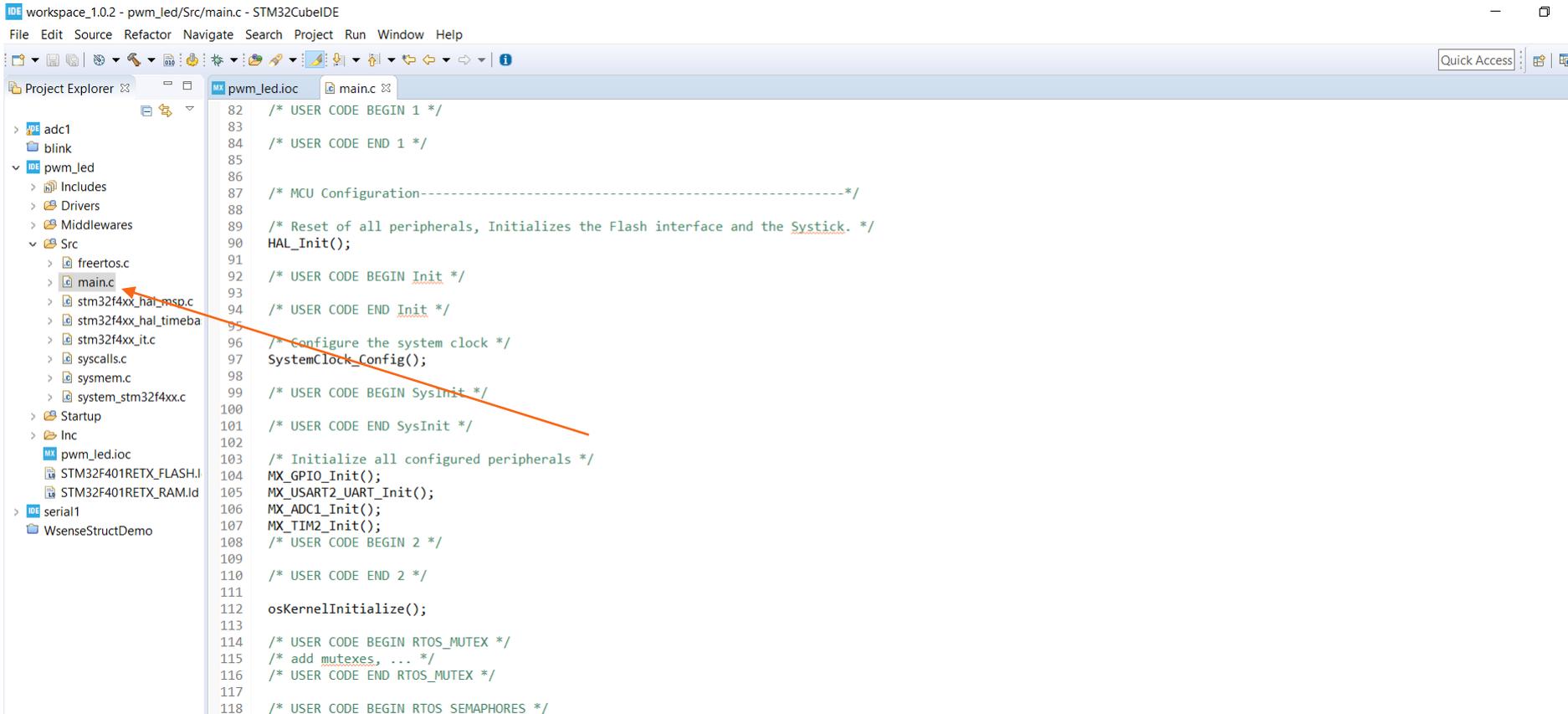
Generate project

The screenshot displays the STM32CubeIDE interface. The main window shows the 'TIM2 Mode and Configuration' settings. The 'Channel4' is highlighted in yellow. The 'Configuration' section includes tabs for 'NVIC Settings', 'DMA Settings', 'GPIO Settings', 'Parameter Settings', and 'User Constants'. A search bar for constants is visible. On the right, a pinout diagram shows the connection of a potentiometer to pins PA1, PA2, and PA0. The potentiometer is connected to PA1 (wiper), PA2 (bottom), and PA0 (top). The diagram also shows other pins like VBAT, VSS, VDD, and USART\_TX.

# NUCLEO F401RE – ESEMPIO

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

## Aprire main.c



The screenshot shows an IDE window titled "workspace\_1.0.2 - pwm\_led/Src/main.c - STM32CubeIDE". The Project Explorer on the left shows a project structure with a "pwm\_led" folder containing a "Src" subfolder. The "main.c" file is selected in the "Src" folder. The main code window displays the following C code:

```
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

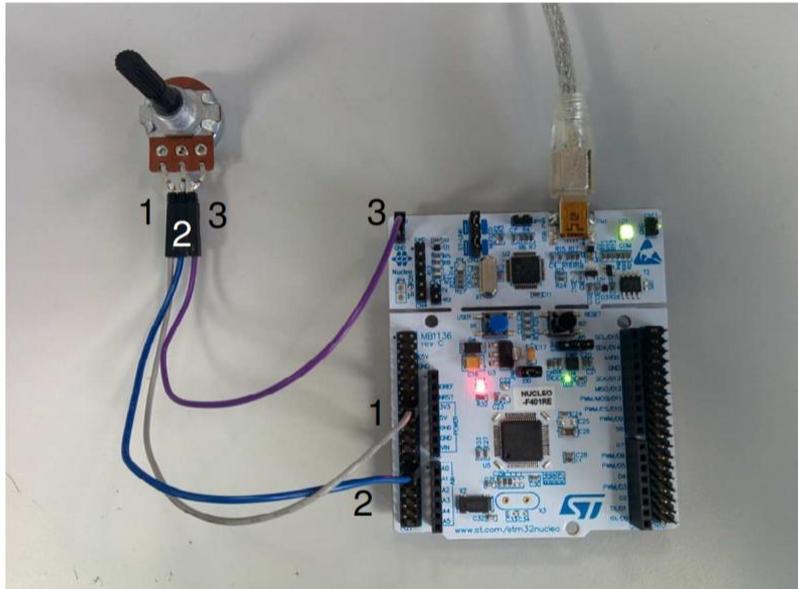
Scrive le informazioni sulla seriale

```
.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

collegamenti



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