

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)

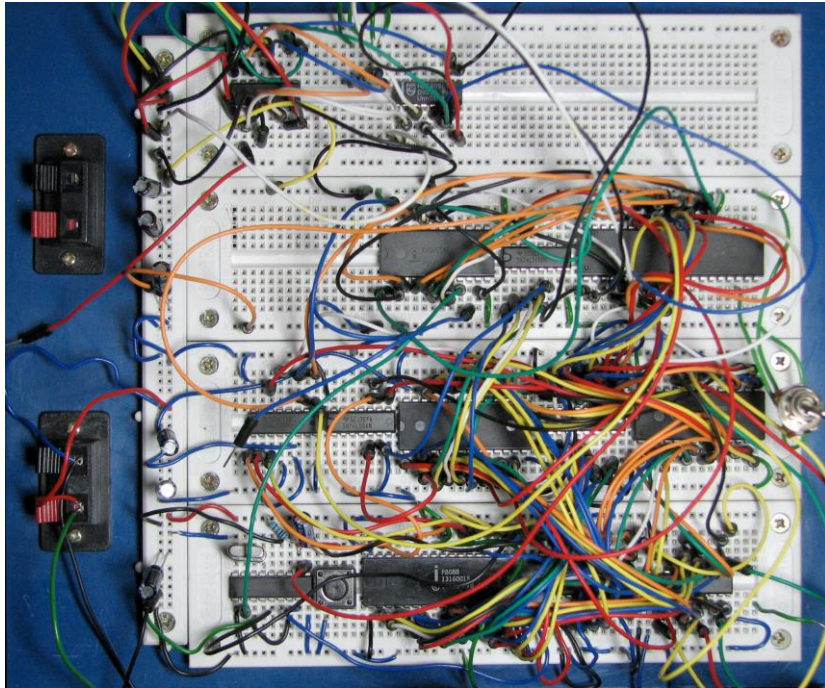
Programma di martedì prossimo

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

PER CHI VOLESSE APPROFONDIRE LA PARTE DI ARM

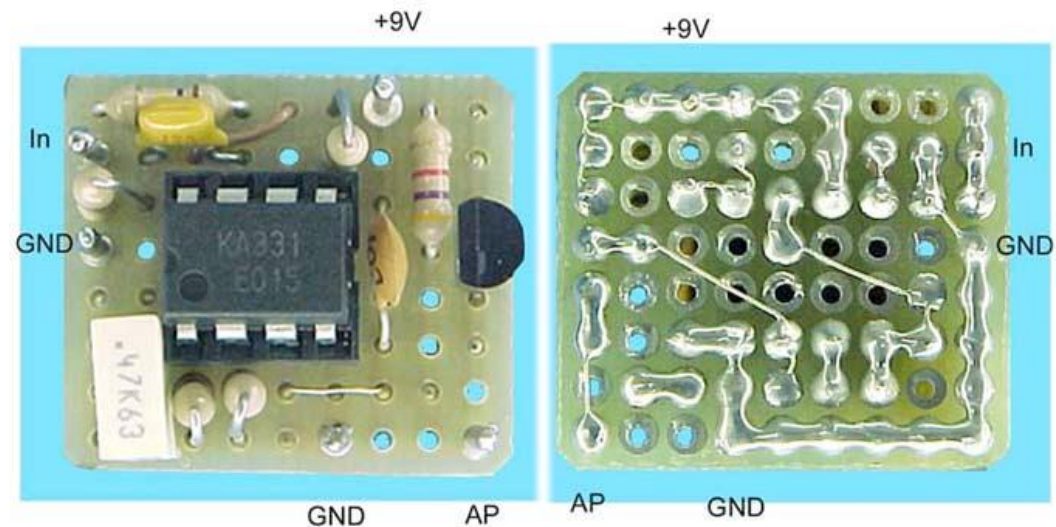


Realizzare PCB – fase 1



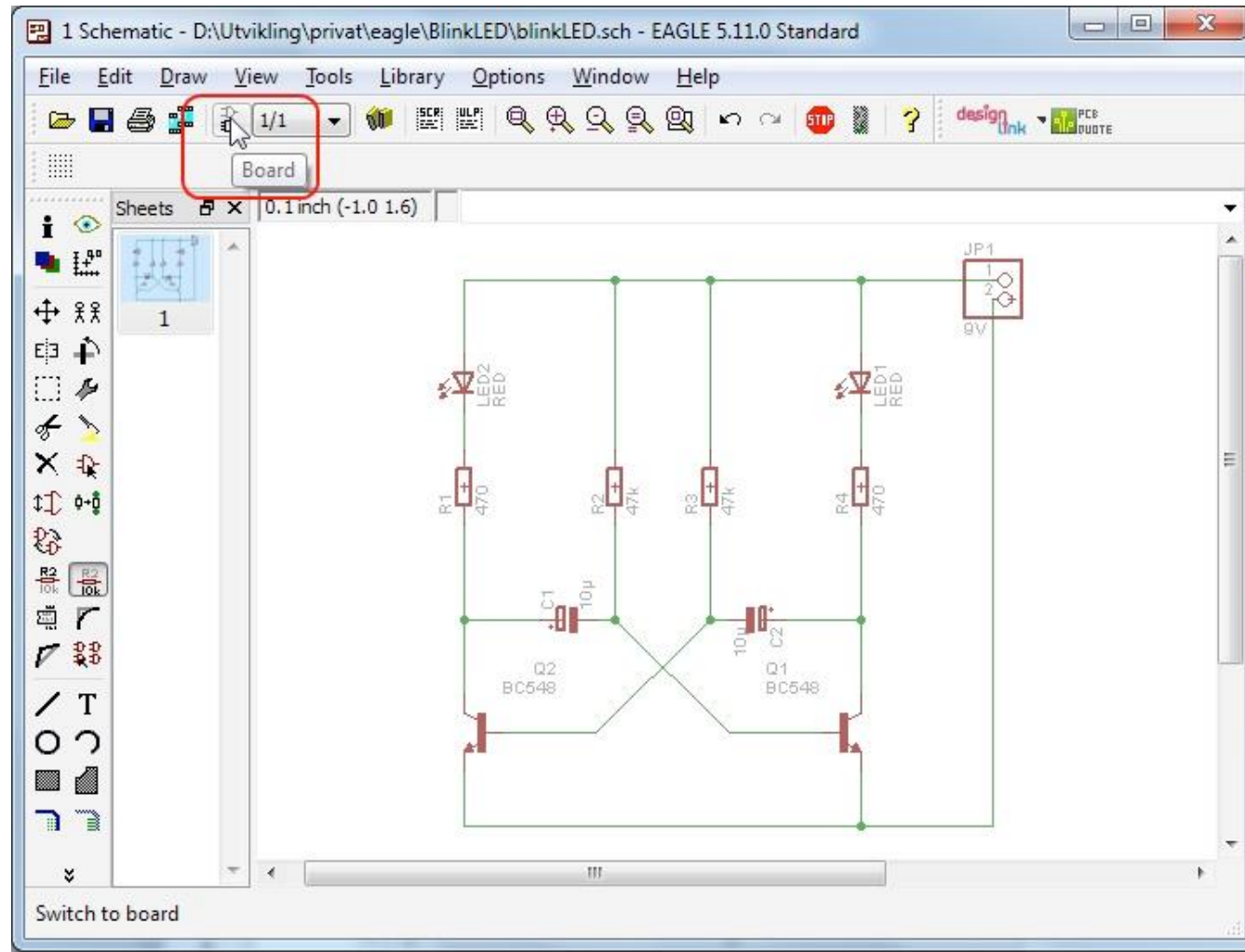
Progettare e testare il circuito su breadboard o millefori

Questo permette di sviluppare e testare software e hardware



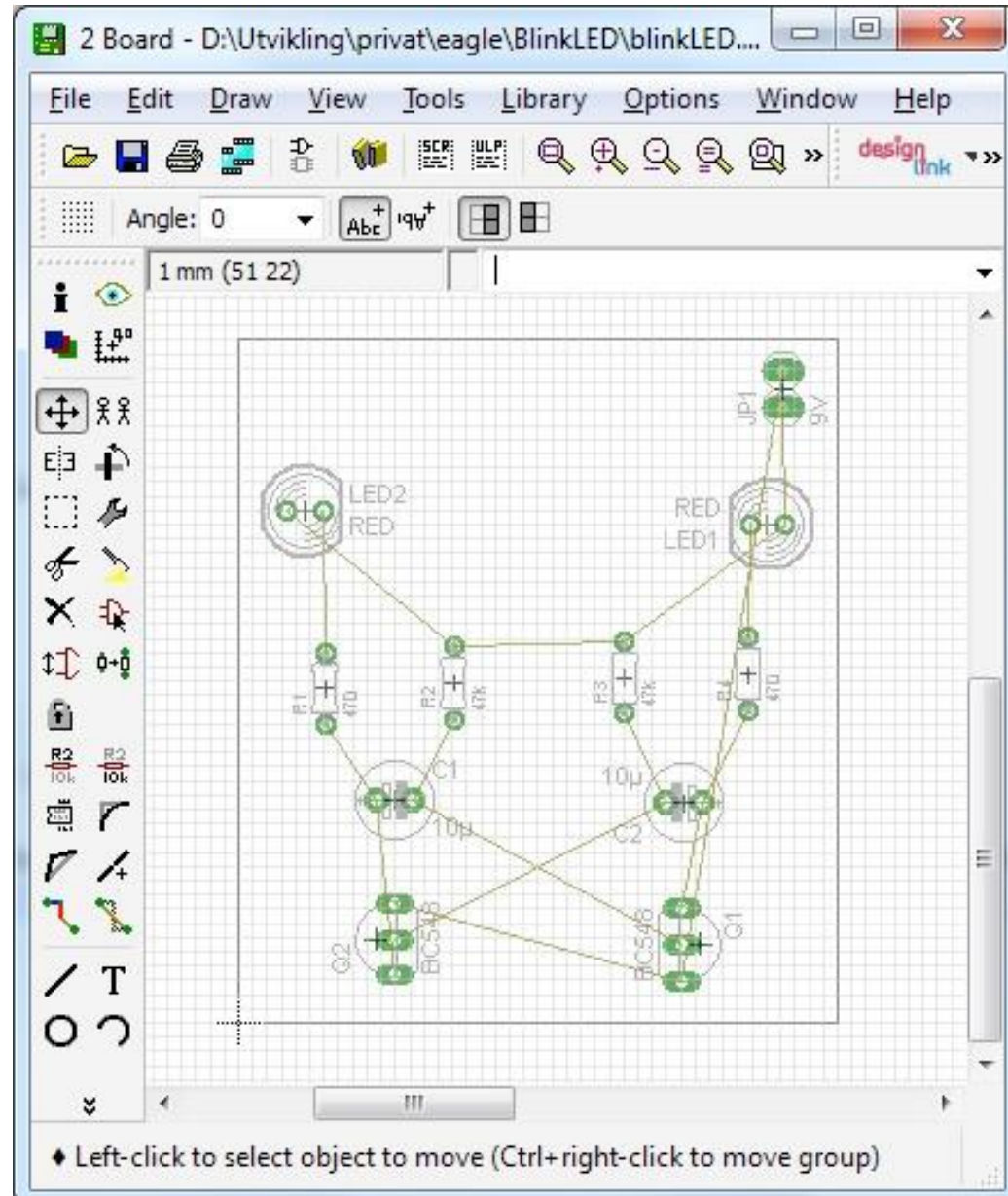
Realizzare PCB – fase 2

Disegnare lo schematico al cad



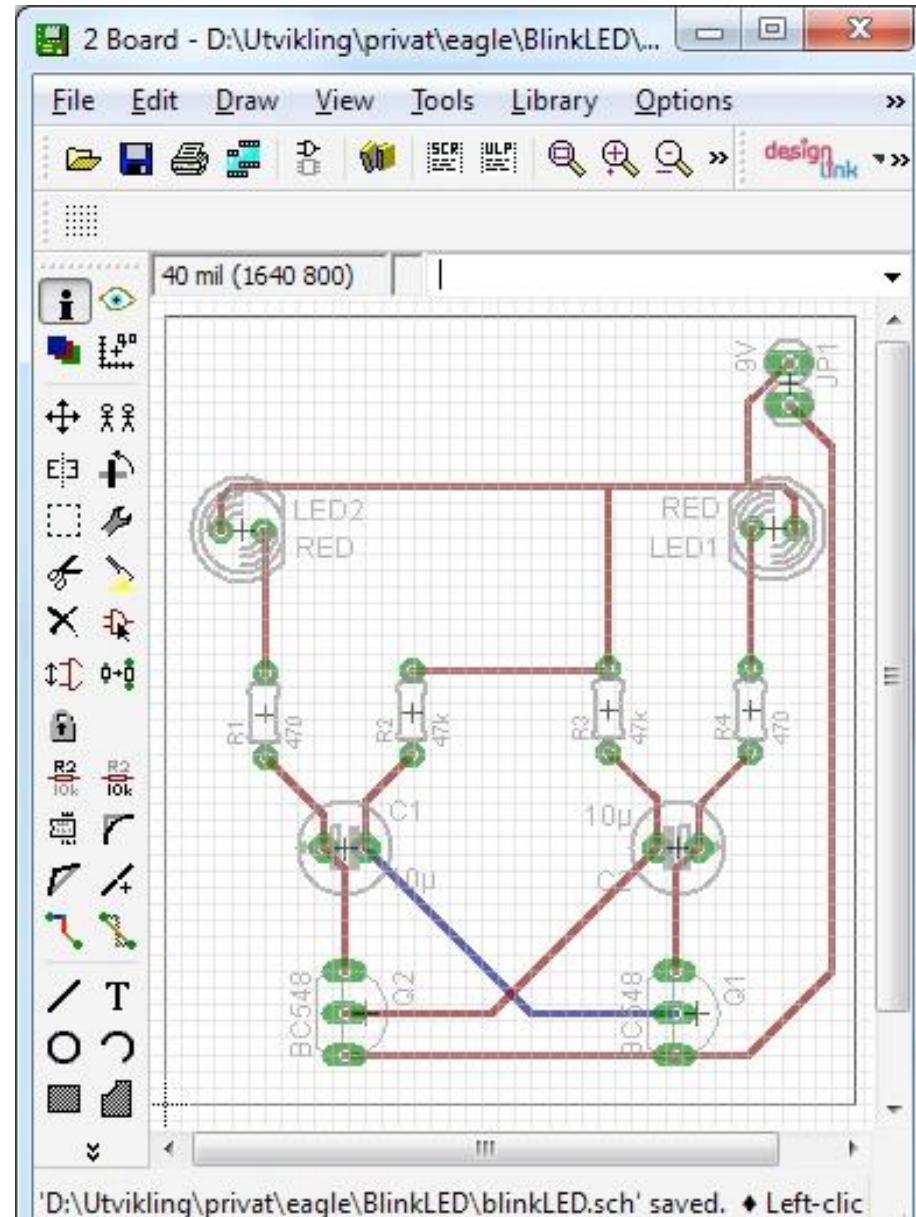
Realizzare PCB – fase 3

Disegnare la board al cad



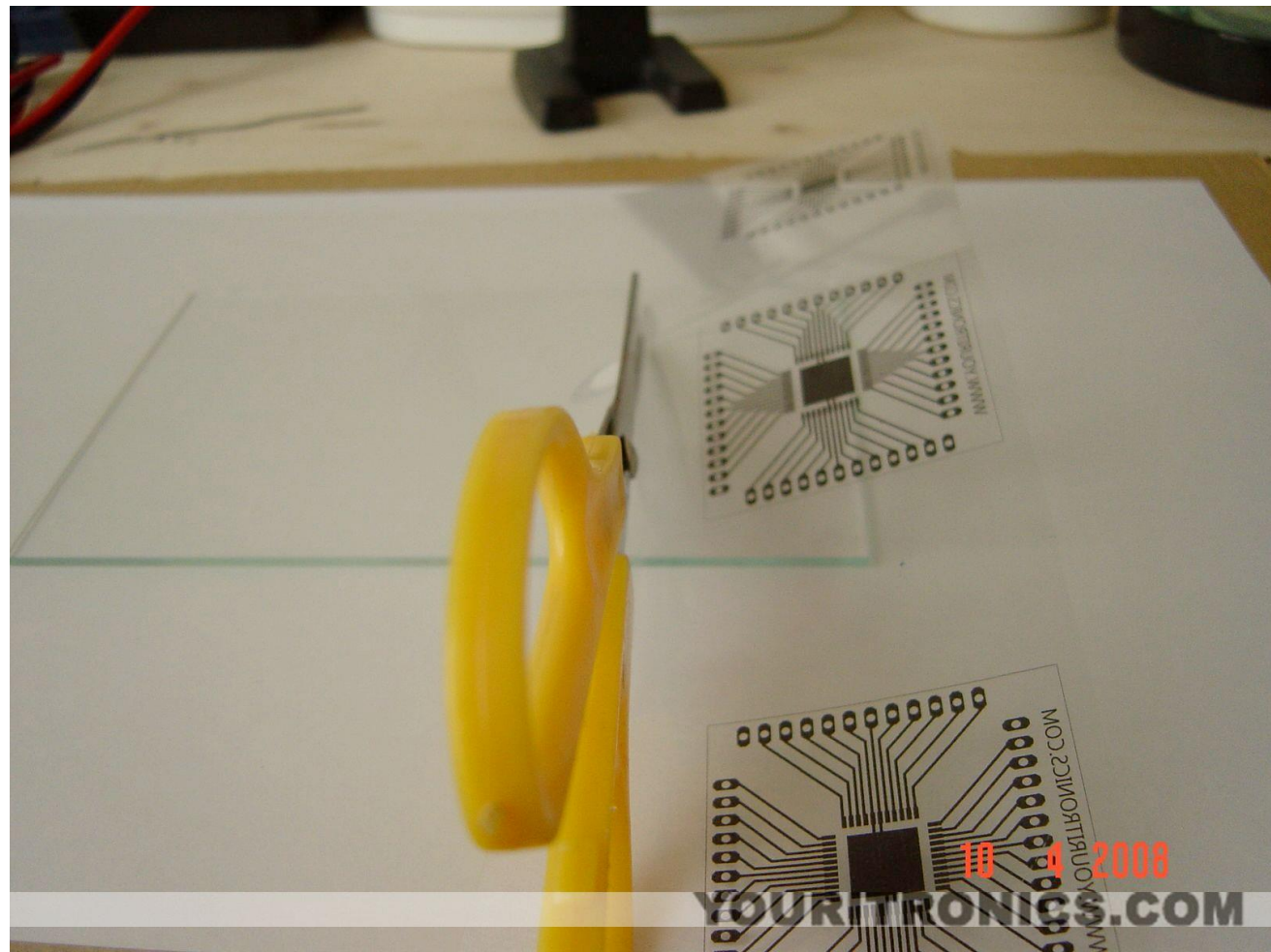
Realizzare PCB – fase 4

Routare i componenti al cad



Realizzare PCB – fase 5

Stampare su carta trasparente – MASTER (circuito specchiato)



Realizzare PCB – fase 6

Utilizzando delle basette fotosensibili

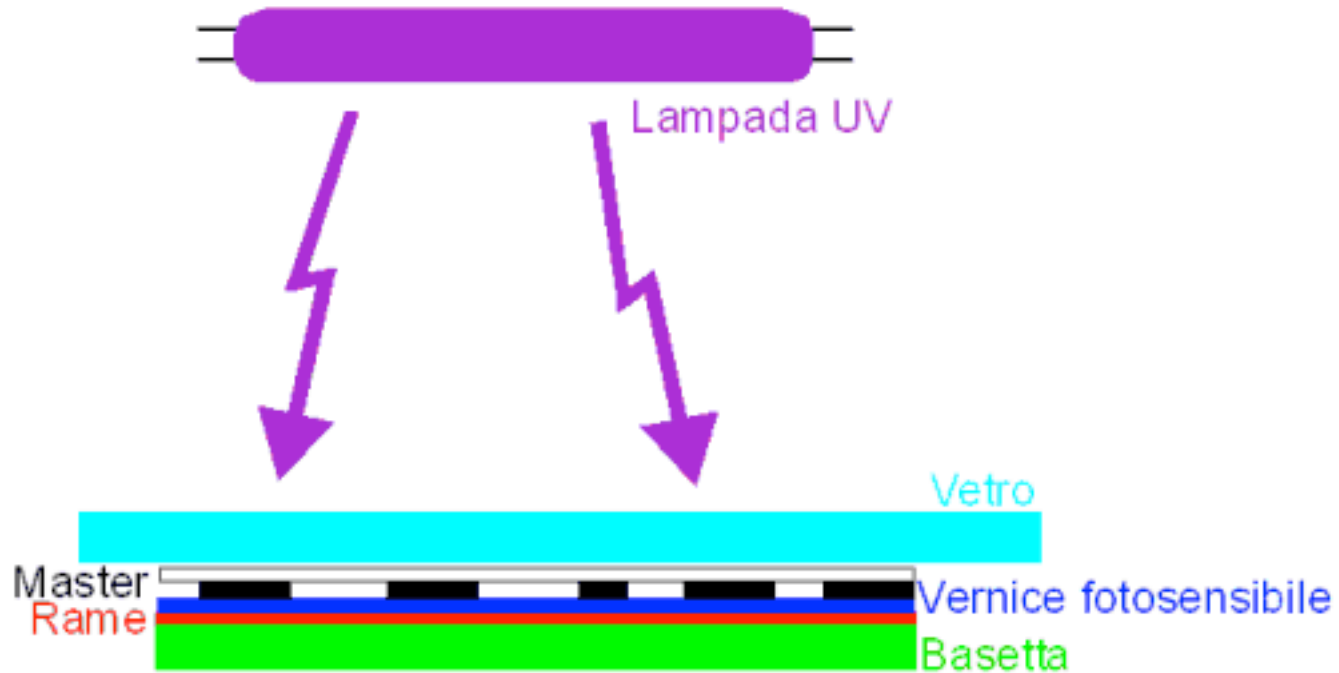


10 4 2008

YOURITRONICS.COM

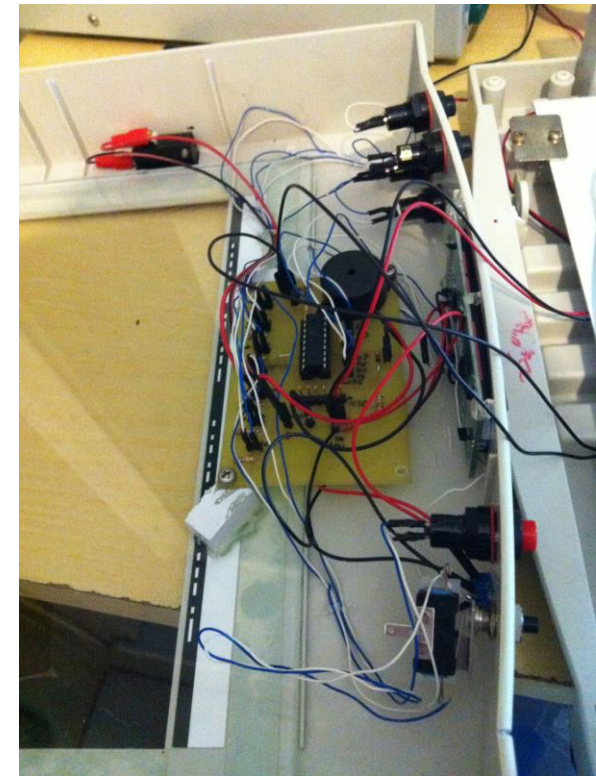
Realizzare PCB – fase 7

Lo sviluppo con il bromografo



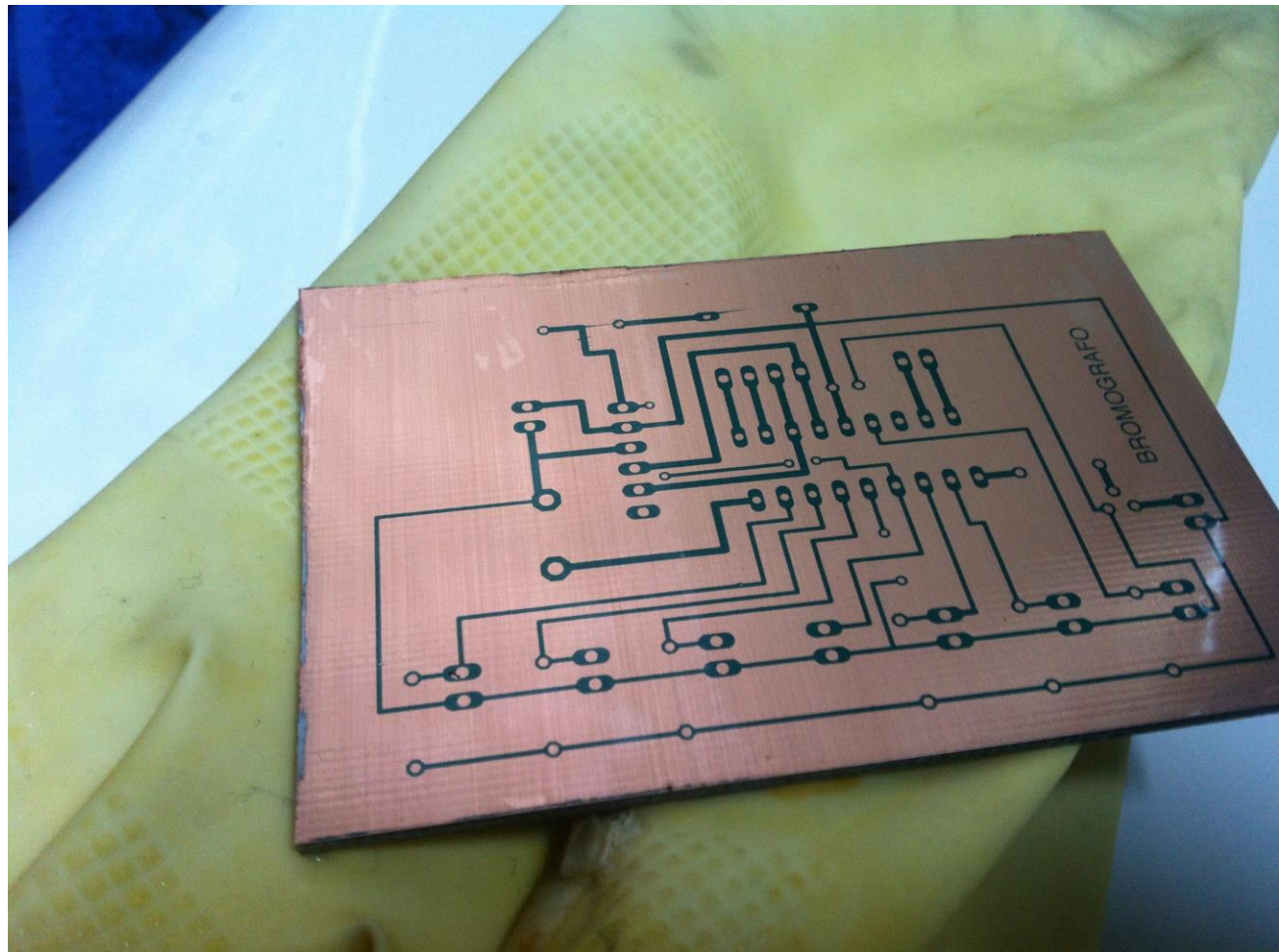
Realizzare PCB – fase 8

Parentesi... il bromografo



Realizzare PCB – fase 9

Dopo il lavaggio con la soda caustica



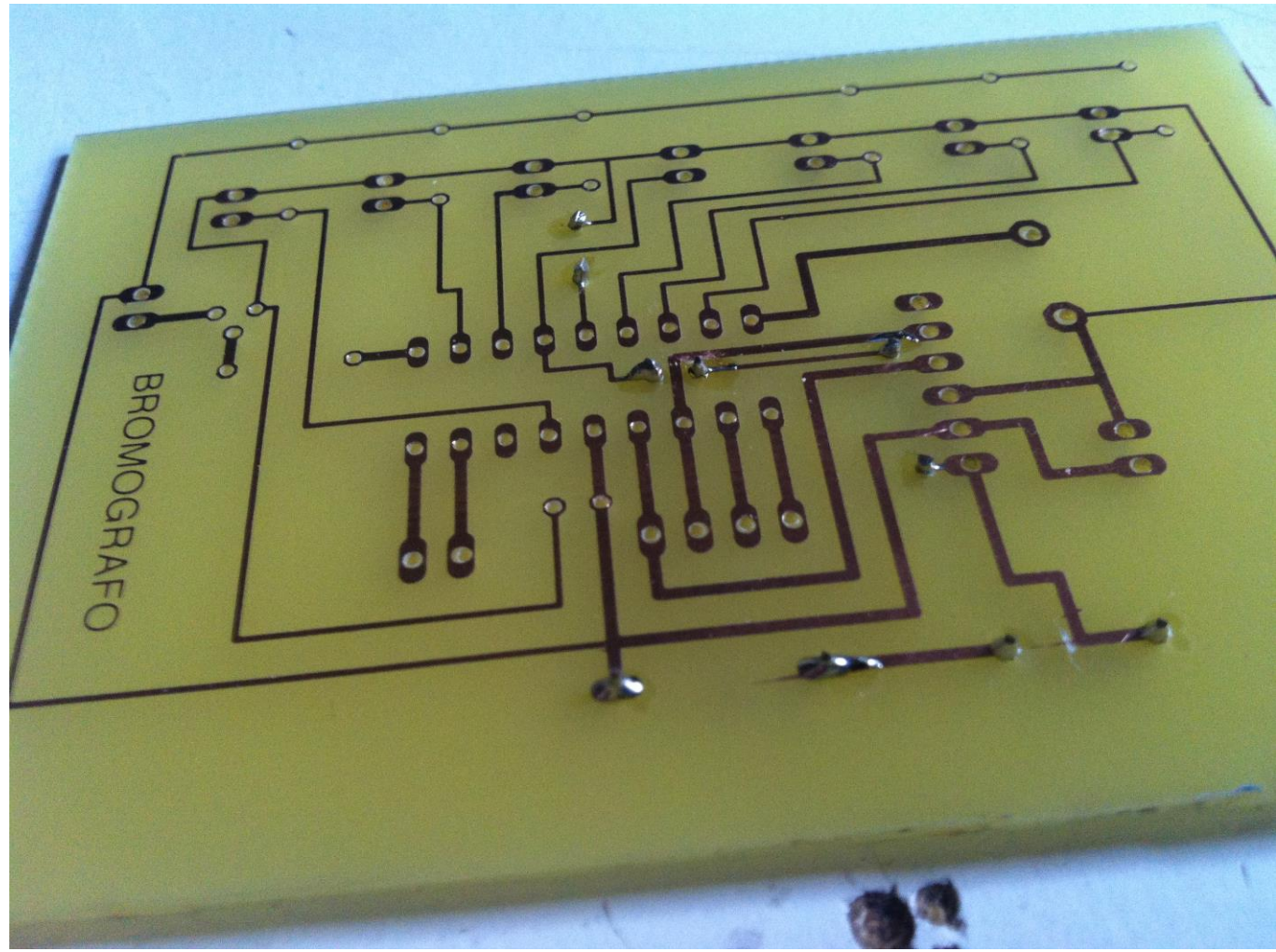
Realizzare PCB – fase 10

Circuito nell'acido



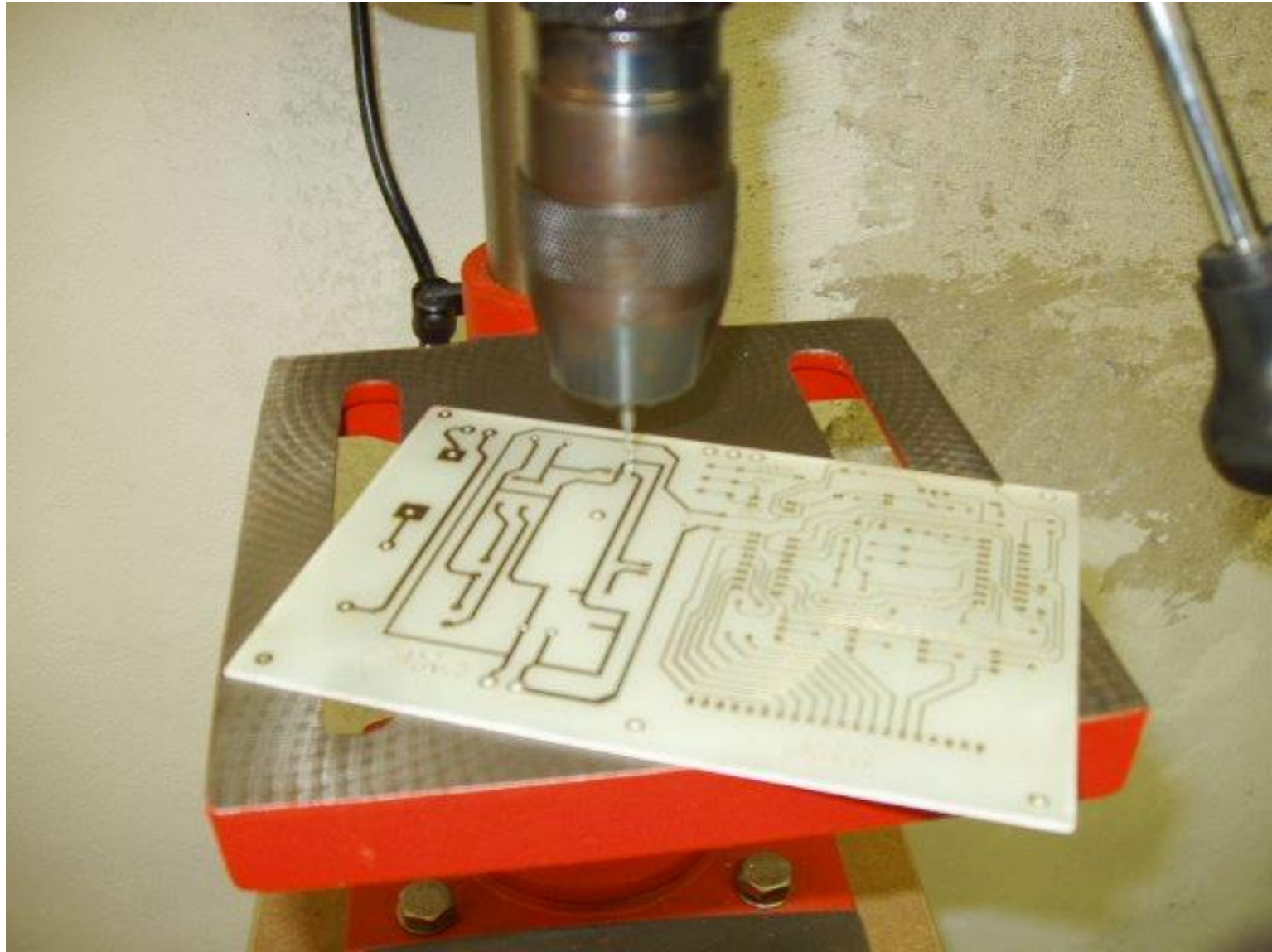
Realizzare PCB – fase 11

Dopo il lavaggio



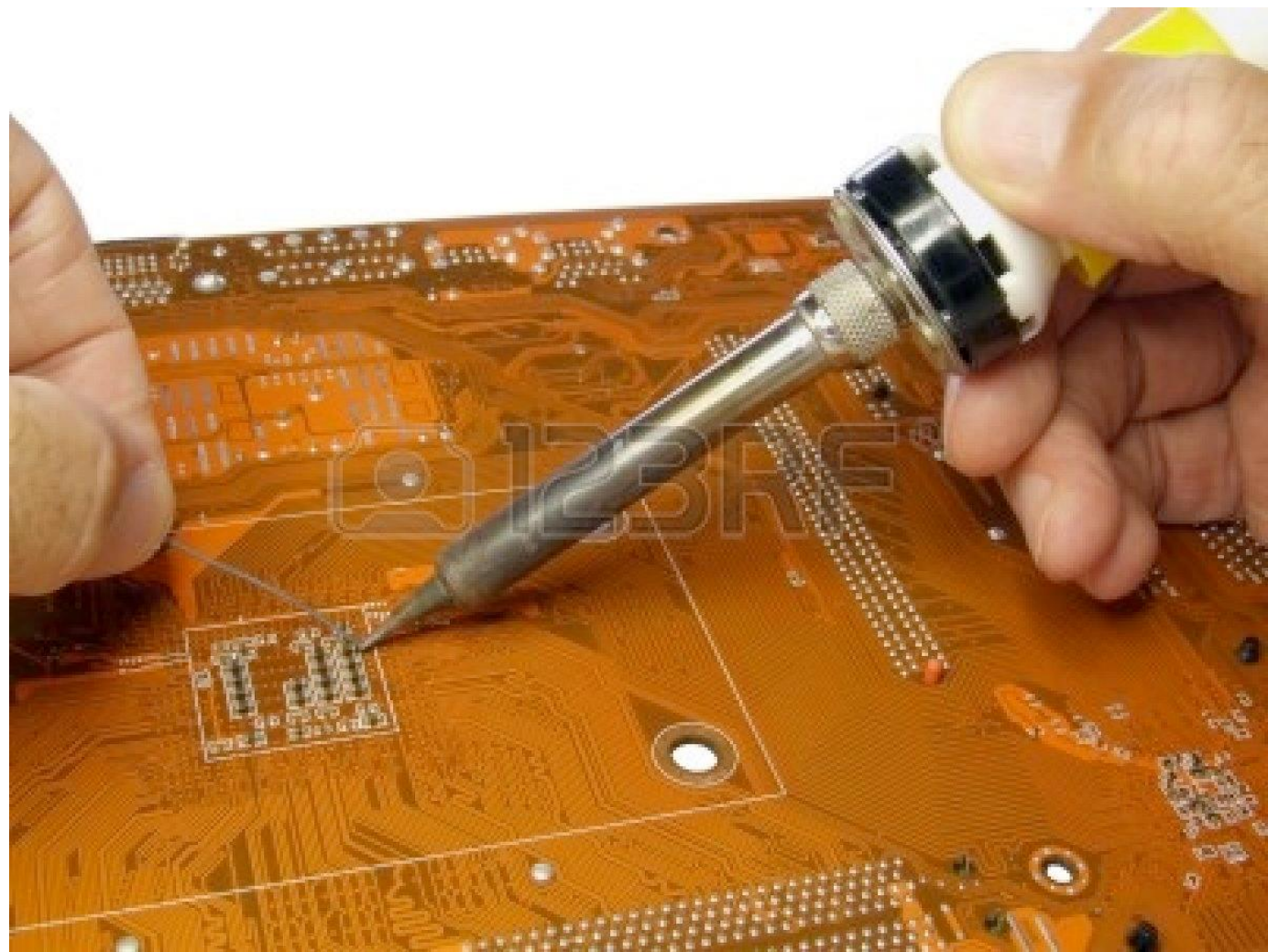
Realizzare PCB – fase 12

foratura

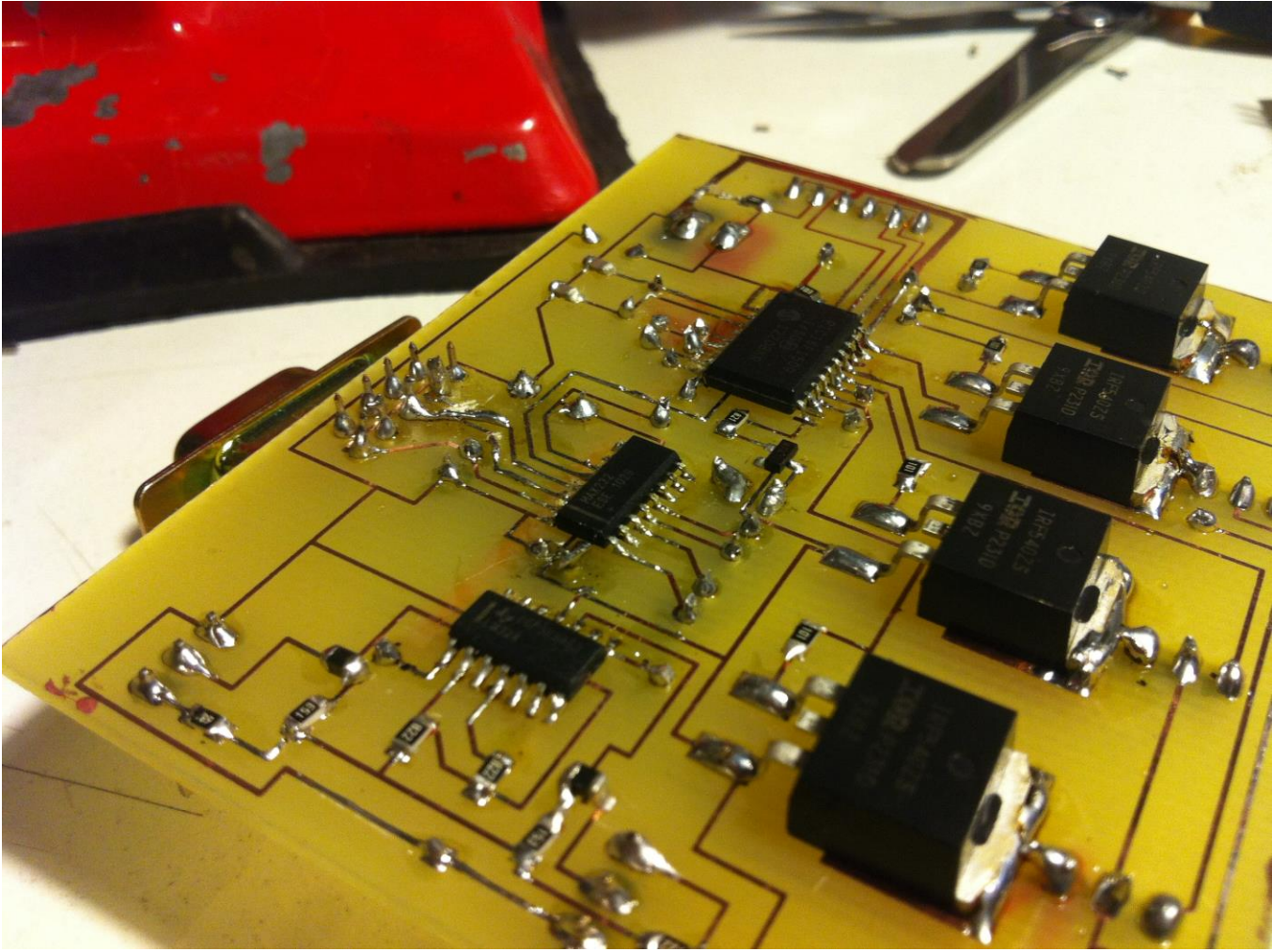


Realizzare PCB – fase 13

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);

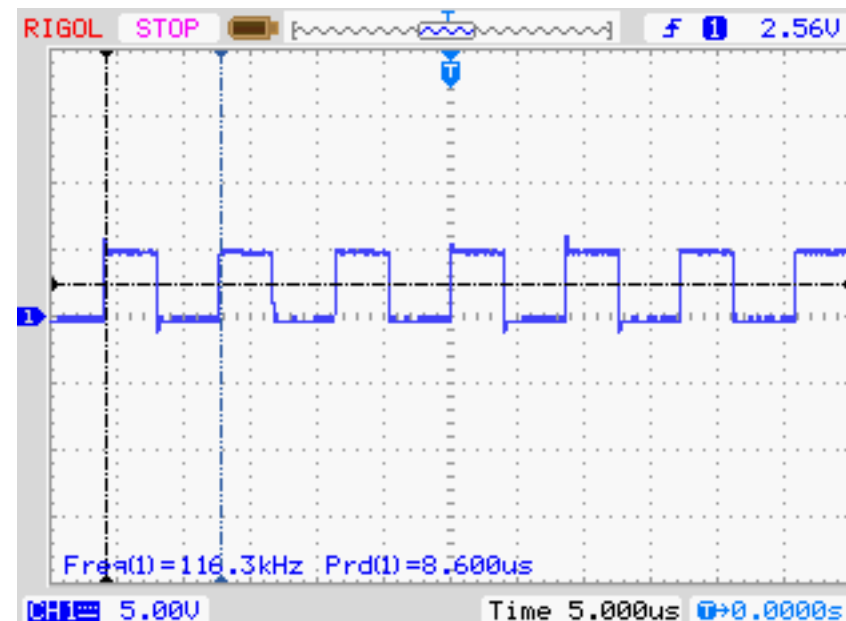
    //out = portOutputRegister(port);
    out =(volatile uint8_t *) (pgm_read_word( port_to_output_PGM + pin ));

    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;
```

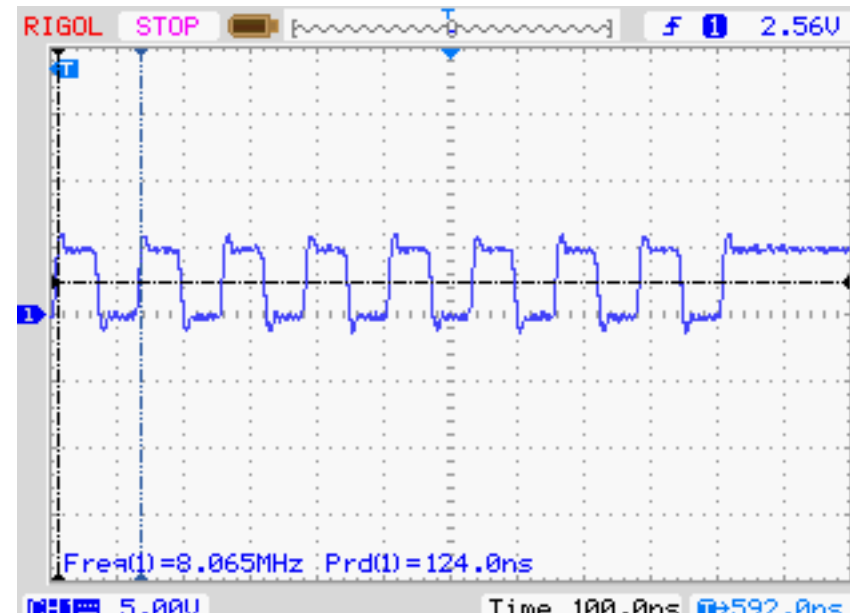
Importante: inizializzazione delle variabili

A = B00000101 -> B binario

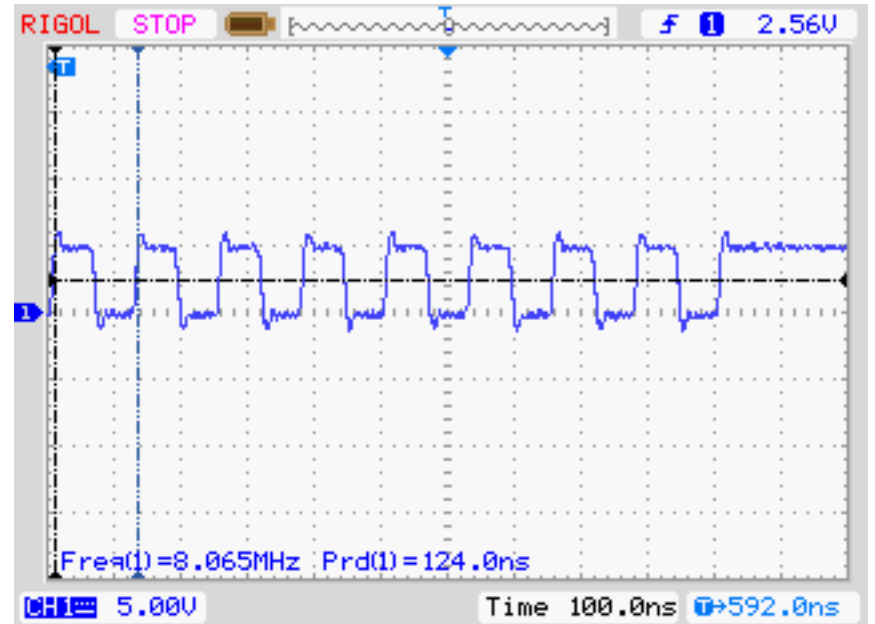
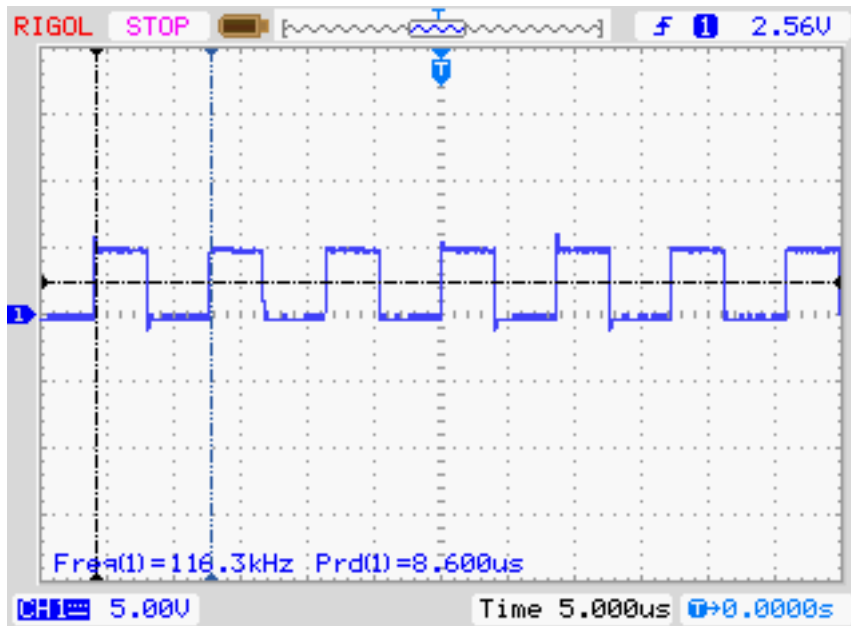
A = 0x05 -> 0x esadecimale

A = 5 -> decimale

Risultato



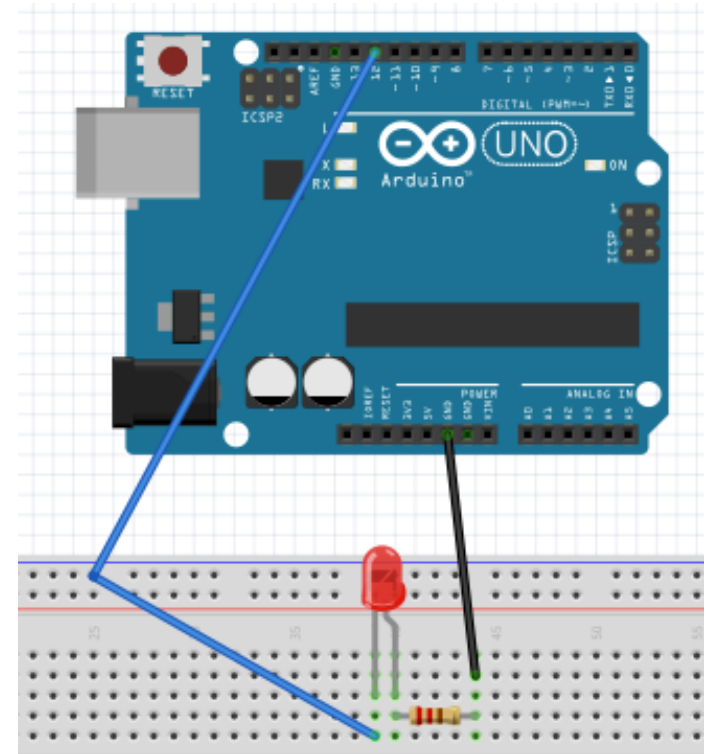
Confrontando...



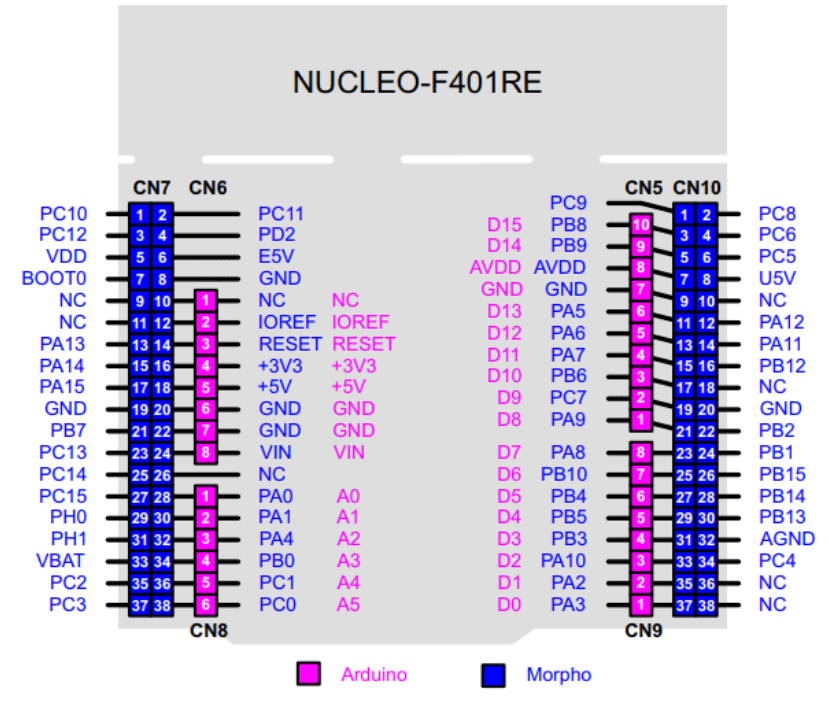
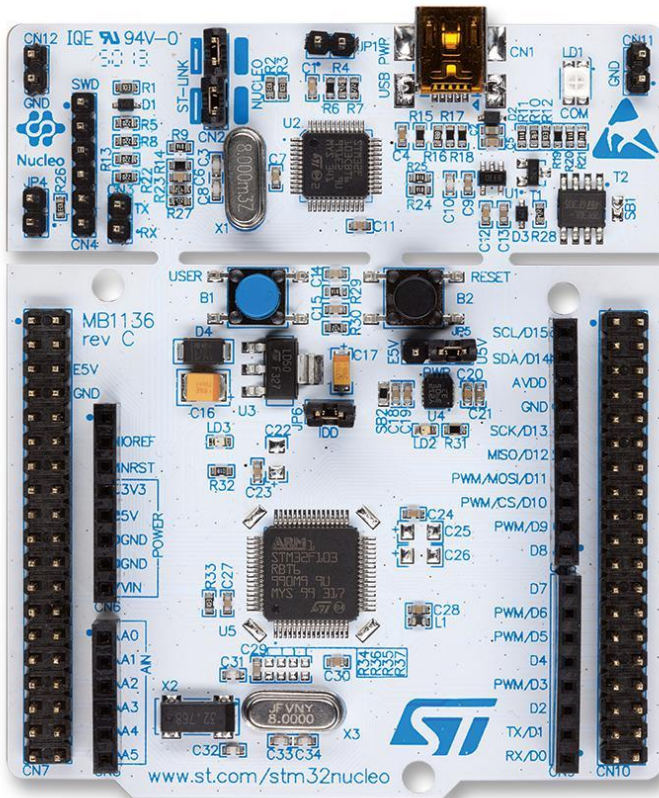
ACCENDERE UN LED CON PORT MANIPULATION

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

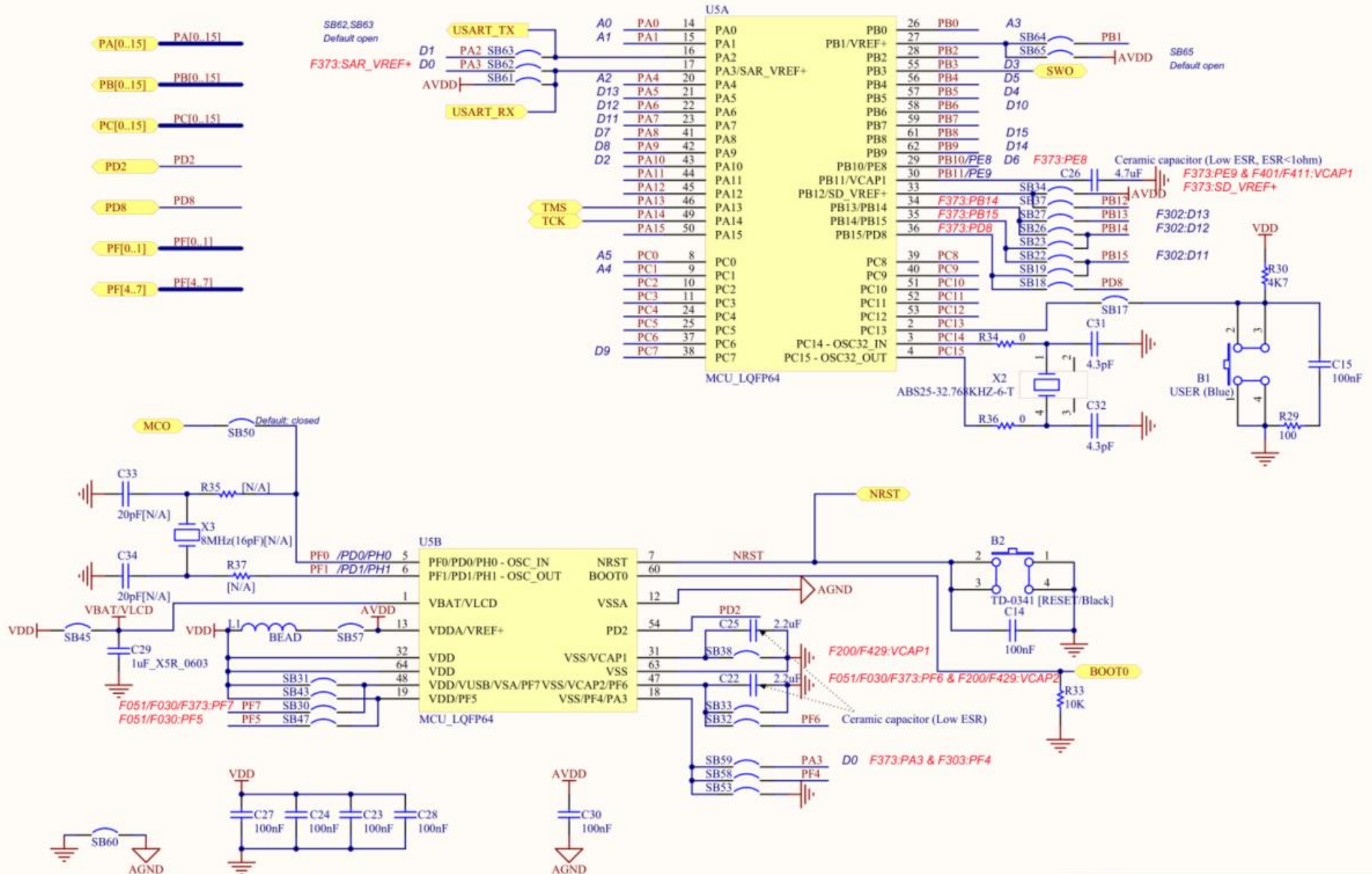
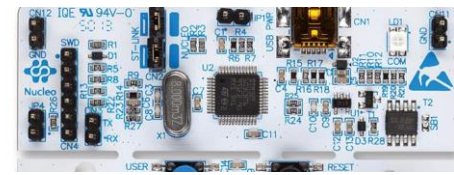
void loop(){
  //digitalWrite(7, HIGH);
  //digitalWrite(6, HIGH);
  //digitalWrite(5, HIGH);
  PORTD = PORTD | B11100000;
  delay(1000);
  |
  //digitalWrite(7, LOW);
  //digitalWrite(6, LOW);
  //digitalWrite(5, LOW);
  PORTD = PORTD & B00011111;
  delay(1000);
}
```



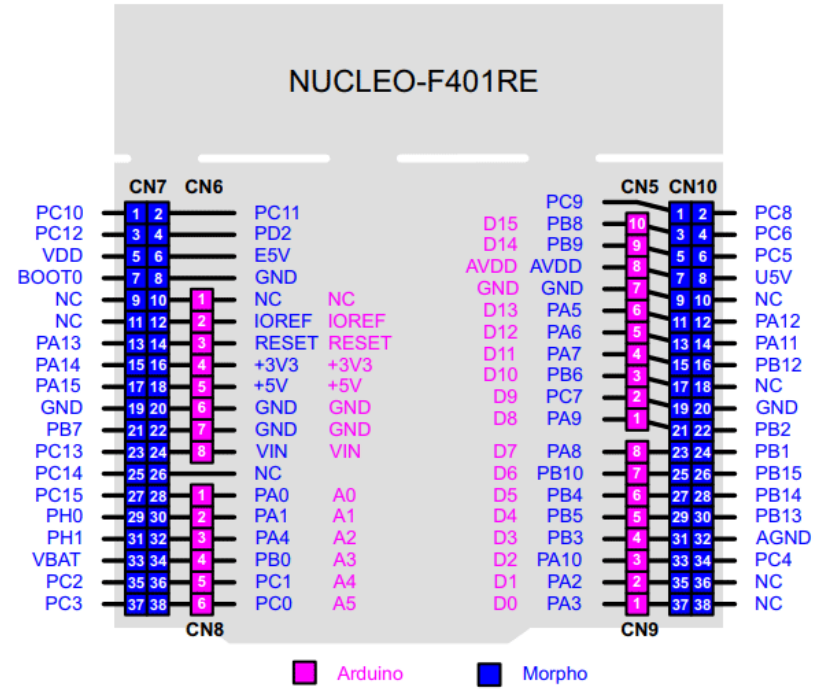
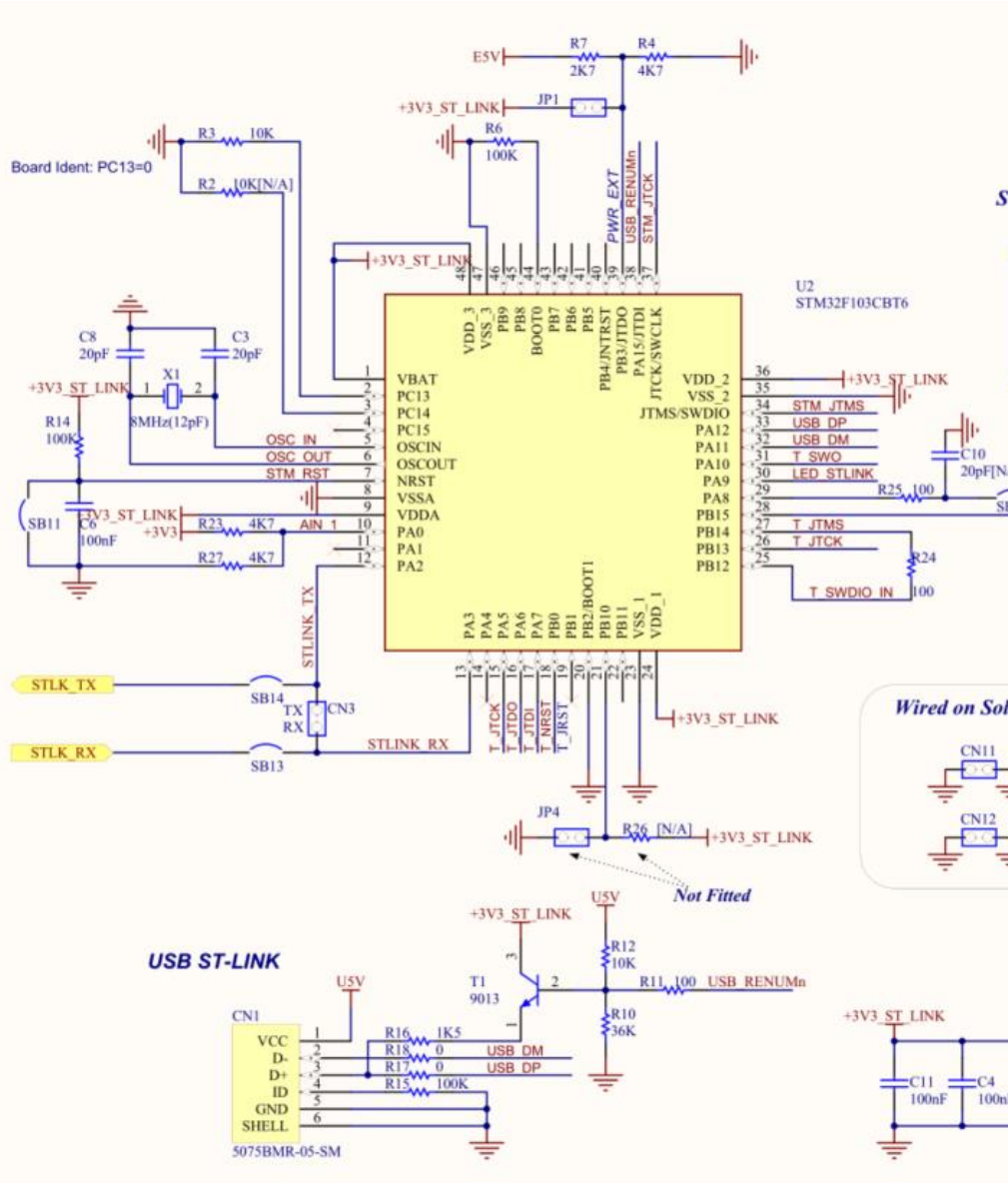
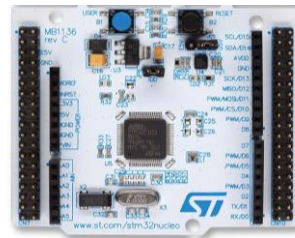
NUCLEO F401RE - HW



NUCLEO F401RE - SCHEMA

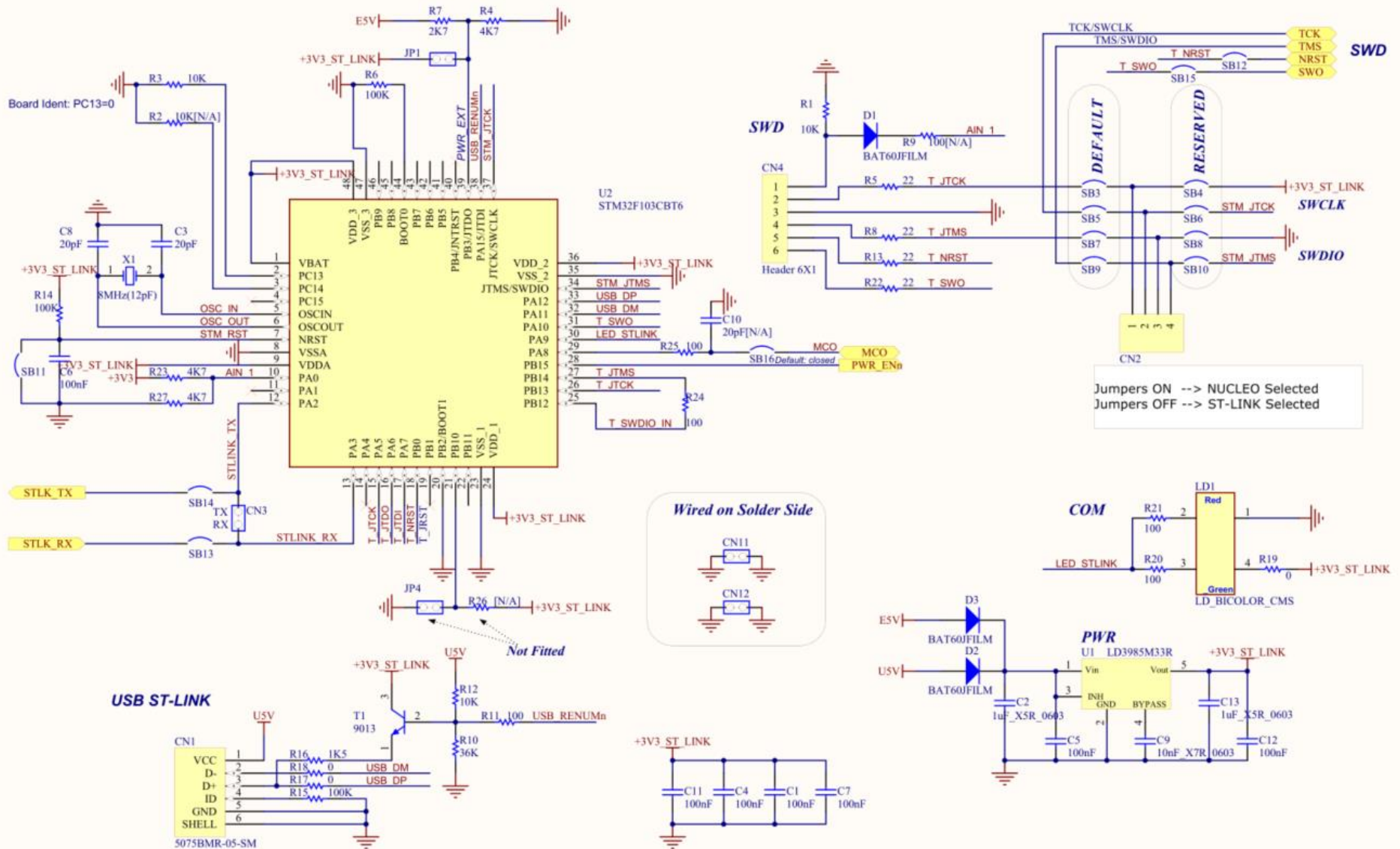


NUCLEO F401RE - SCHEMA



■ Arduino
 ■ Morpho

NUCLEO F401RE - SCHEMA



NUCLEO F401RE – ESEMPIO

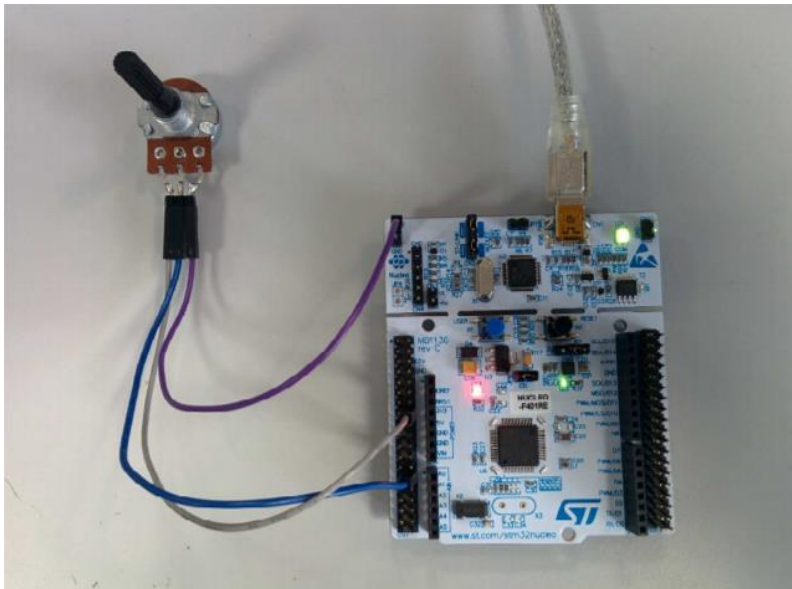
USIAMO UN POTENZIOMETRO PER REGOLARE LA LUMINOSITÀ DI UN LED IN PWM

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 "Target Selection" section indicates "Select STM32 target". The "Board Selector" tab is active, showing a search for "NUCLEO-F401RE". The "Board Filters" panel on the left includes a search bar, vendor, type, and MCU/MPU series filters, along with price and oscillator frequency sliders. The "Peripheral" section lists various features like Accelerometer, Analog I/O, and Button. The main content area displays the "NUCLEO-F401RE" board details, including a description, marketing status (Active), unit price (13.0), and mounted device (STM32F401RETx). A table below shows the board list with one item: NUCLEO-F401RE, Nucleo64, Active, 13.0, and STM32F401RETx. The bottom navigation bar includes buttons for Back, Next, Finish, and Cancel.

Board Filters

Part Number Search

Vendor

Type

MCU/MPU Series

Other

Price = 13.0

Oscillator Freq. = 0 (MHz)

Peripheral

- Accelerometer 0 0
- Analog I/O 0 0
- Arduino Form Factor 0 0
- Audio Line In 0 0
- Audio Line Out 0 0
- Battery
- Button 0 2
- CAN 0 0
- Camera
- Compass
- Custom Form Factor 0 0

Features

Large Picture

Docs & Resources

Datasheet

Buy

NUCLEO-F401RE

STMicroelectronics NUCLEO-F401RE Board Support and Examples

ACTIVE Active
Product is in mass production

Unit Price (US\$): 13.0

Mounted device: [STM32F401RETx](#)

The STM32 Nucleo-64 boards provide an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the external SMPS significantly reduces power consumption in Run mode. The Arduino™ Uno V3 connectivity support and the ST morpho headers allow the easy

Boards List: 1 item

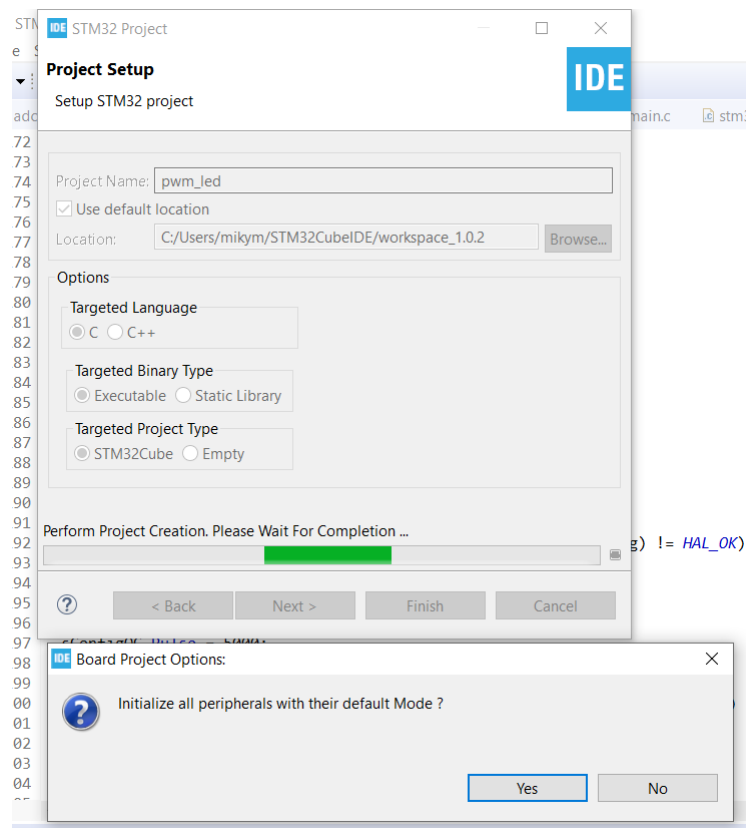
*	Overview	Part No	Type	Marketing Status	Unit Price (US\$)	Mounted Device
☆		NUCLEO-F401RE	Nucleo64	Active	13.0	STM32F401RETx

< Back Next > Finish 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 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" project and its source files. The "Pinout & Configuration" tab is active, showing the "TIM2 Mode and Configuration" settings. The "Mode" section is expanded, showing the following configuration:

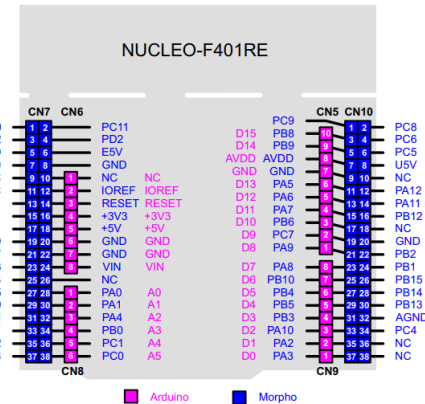
- Slave Mode: Disable
- Trigger Source: Disable
- Clock Source: Disable
- Channel1: Disable
- Channel2: Disable
- Channel3: Disable
- Channel4: Disable
- Combined Channels: Disable
- Use ETR as Cleaning Source:
- XOR activation:
- One Pulse Mode:

The "Configuration" section is currently empty. The "Pinout view" on the right shows the physical pinout of the Nucleo F401RE, with the PA5 pin highlighted in yellow. A tooltip for PA5 lists the following functions: 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 F



Schermata pin

ADC

Timer

onfiguration Tool - STM32CubeIDE

Project Run Window Help

Pinout & Configuration

Categories

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

STM32F401RETx LQFP64

Pinout view

System view

SH.I
M.Id

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 with the TIM2 configuration window open. The configuration is set for PWM mode with the following parameters:

- Mode:** Slave Mode: Disable, Trigger Source: Disable, Clock Source: Disable
- Channel1:** PWM Generation CH1
- Channel2:** Disable
- Channel3:** Disable
- Channel4:** Disable

Configuration:

- Reset Configuration
- Parameter Settings: NVIC Settings, DMA Settings, GPIO Settings, Parameter Settings, User Constants
- Configure the below 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 PA13 and PA14 highlighted in green, corresponding to the RCC_0SC32_OUT and RCC_0SC32_IN pins respectively.

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 'ADC1 Mode and Configuration' window open. The 'Mode' section shows the following settings:

- IN0
- IN1
- IN2
- IN3
- IN4
- IN5
- IN6

The 'Configuration' section includes the following parameters:

- Reset Configuration: [Reset]
- NVIC Settings:
- DMA Settings:
- GPIO Settings:
- Parameter Settings:
- User Constants:

Configure the below parameters:

- Search (Ctrl+F): []
- 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: []

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, and USART_TX/USART_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.s 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

Freertos – systick

The screenshot shows the STM32CubeIDE Pinout & Configuration window. The left sidebar displays the 'Categories' tree with 'SYS' selected under 'RCC'. The main area shows the 'SYS Mode and Configuration' section with the following settings:

- Mode: Debug (Serial Wire)
- System Wake-Up:
- Timebase Source: TIM1

The 'Configuration' section below shows a warning: "Warning: This IP has no parameters to be configured." The right side of the window displays a pinout diagram of the STM32F401RE LQFP64 package, with various pins labeled and color-coded.

NUCLEO F401RE – ESEMPIO

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

Generate project

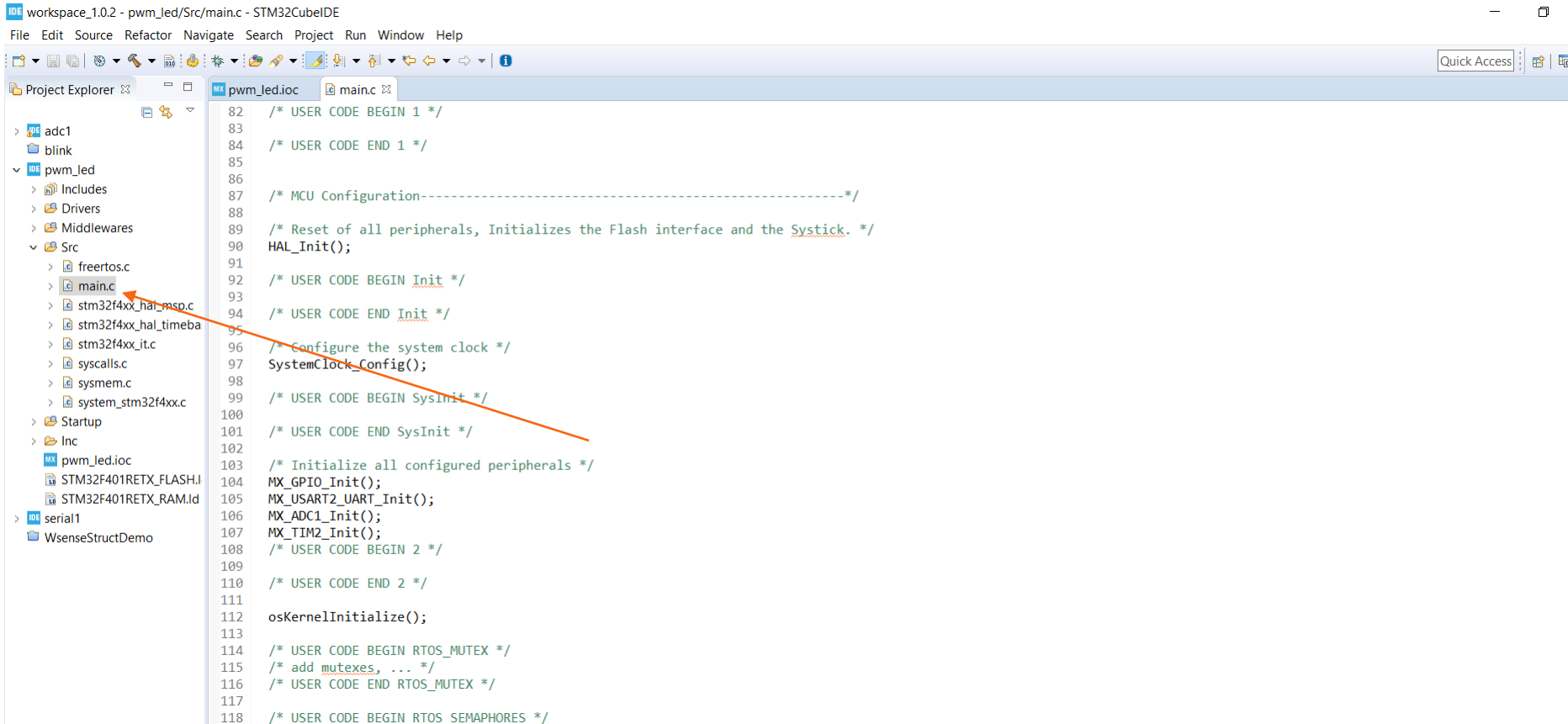
The screenshot displays the STM32CubeIDE interface. The top menu bar includes 'File', 'Project', 'Run', 'Window', and 'Help'. The toolbar contains various icons for file operations and execution. The main workspace is divided into several panes:

- Left Pane:** A tree view showing project components. Under 'Timers', 'TIM2' is selected and highlighted in blue. Other timers listed include RTC, TIM1, TIM3, TIM4, TIM5, TIM9, TIM10, and TIM11. Under 'Connectivity', various peripherals like I2C1, I2C2, SDIO, SPI1, SPI2, SPI3, USART1, and USART2 are listed.
- Center Pane:** The 'TIM2 Mode and Configuration' window. The 'Mode' section shows 'Slave Mode' set to 'Disable', 'Trigger Source' to 'Disable', and 'Clock Source' to 'Disable'. 'Channel1' is set to 'PWM Generation CH1', while 'Channel2', 'Channel3', and 'Channel4' are all set to 'Disable'. The 'Configuration' section includes a 'Reset Configuration' button and tabs for 'NVIC Settings', 'DMA Settings', 'GPIO Settings', 'Parameter Settings', and 'User Constants'. Below this is a 'Search Constants' field with 'add' and 'remove' buttons, and a table with columns for 'Constant Name' and 'Constant Value'.
- Right Pane:** A 'Pinout' view showing a diagram of the Nucleo F401RE board. The board is oriented vertically. Labels on the left side include: VBAT, B1 (Blue PushButton), RCC_OS32_IN (PC13), RCC_OS32_OUT (PC14), RCC_OS32_IN (PH0), RCC_OS32_OUT (PH1), NRST, PC0, PC1, PC2, PC3, VSS, VRE, ADC1_IN0 (PA0), PA1, and USART_TX (PA2). Labels on the right side include: U1D, VSS, PB0, PB1, PB2, B100, PB7, and PA0. At the bottom, labels include USART_RX (PA9), VSS, U1D, PA4, PA5, PA6, PA7, PA8, and TIM2_CH1 (PA9).

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 folders like "adc1", "blink", "pwm_led", "Includes", "Drivers", "Middlewares", "Src", "Startup", "Inc", "pwm_led.ioc", "STM32F401RETX_FLASH", "STM32F401RETX_RAM.ld", "serial1", and "WsenseStructDemo". The "main.c" file is selected in the "Src" folder. The main function code is displayed in the editor, starting with a comment "/* USER CODE BEGIN 1 */" and ending with "/* USER CODE BEGIN RTOS SEMAPHORES */". The code includes comments for MCU Configuration, HAL_Init(), SystemClock_Config(), SysInit(), and initialization of peripherals (MX_GPIO_Init(), MX_USART2_UART_Init(), MX_ADC1_Init(), MX_TIM2_Init()).

```
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

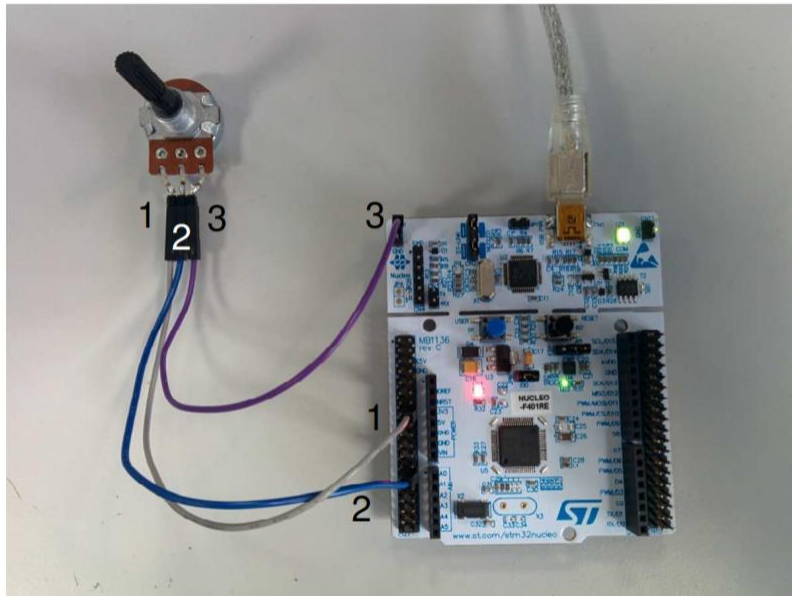
Scrive le informazioni sulla seriale

```
.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