# Homework #7 : ADC e temperatura + DWT

Michele Corrias

14 maggio 2021

## 1 Consegna

Scrivere un programma per STM32F4 che acquisisce i dati dal sensore di temperatura incorporato usando gli interrupt, e nel caso in cui la temperatura rilevata superi i 28° attivi il led rosso, in caso in cui la temperatura scenda sotto i 18° accenda il led blu. Per le temperature intermedie resta acceso il led verde.

Il programma deve inoltre calcolare il tempo impiegato dalla routine di risposta all'interrupt generata dal circuito ADC.

Descrivere anche come avete fatto a testare il programma.

## 2 Premessa

Per questo homework sono stati utilizzati: Analog to Digital Converter (ADC), il sensore di temperatura connesso internamente al canale ADC1_16 (che converte in digitale l'output del sensore di temperatura) ed il registro CYCCNT dell'unità di debugging Data Watchpoint and Tracing (DWT) per contare il numero di cicli della CPU (oltre ovviamente alla UART/USART e ai led verde, rosso e blu).

## 3 Soluzione

Nel main vengono eseguite, tra le solite operazioni, anche quelle di inizializzazione di ciò che utilizzeremo.

```
1  MX_GPIO_Init();
2  MX_ADC1_Init();
3  MX_USART2_UART_Init();
4  HAL_NVIC_SetPriority(ADC_IRQn, 0, 0);
5  HAL_NVIC_EnableIRQ(ADC_IRQn);
6  HAL_ADC_Start_IT(&hadc1);
7  RetargetInit(&huart2);
8  /* indirizzi di memoria dei registri di DWT */
9  volatile uint32_t *DWT_CONTROL = (uint32_t*) 0xE0001000;
10 volatile uint32_t *DWT_CYCCNT = (uint32_t*) 0xE0001004;
11 /* Reset e abilitazione del registro counter */
12 *DWT_CYCCNT = 0;
13 *DWT_CONTROL |= 0x1;
```

Il cuore del programma sta nella callback della Interrupt Service Routine che gestisce l'interrupt generato dal circuito ADC.

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
  char msg[30];
  uint16_t rawValue;
  float temp; // temperatura
  /* Ottengo il dato, poi da convertire in gradi celsius */
  rawValue = HAL_ADC_GetValue(&hadc1);
  temp = ((float)rawValue) / 4095 * 2900; // 2900    il VDD misurato sulla
   board, anzich  3.3V
  temp = ((temp - 760.0) / 2.5) + 25;
  /* Stampo via UART il dato acquisito da ADC */
  sprintf(msg, "Raw Value = %hu\r\n", rawValue);
  HAL_UART_Transmit(&huart2, (uint8_t*) msg, strlen(msg), HAL_MAX_DELAY);
  /* Stampo via UART la temperatura convertita */
  sprintf(msg, "Temperature = %.2f Celsius\r\n", temp);
  HAL_UART_Transmit(&huart2, (uint8_t*) msg, strlen(msg), HAL_MAX_DELAY);
  // LEDS
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12|GPIO_PIN_14|GPIO_PIN_15, 0);
  /* temperatura troppo calda: led rosso */
  if(temp>28)
  {
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, 1);
  }
  /* temperatura troppo fredda: led blu */
  else if(temp<18)
  {
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, 1);
  }
  /* temperatura regolare: led verde */
  else
  {
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, 1);
  }
  /* delay di attesa per rallentare l'output */
  for(int i = 0; i < 0xfffff; i++);
}
```

Per la conversione del valore della temperatura non è stato utilizzato un $V_{ref} = 3300mV$ come indicato sul libro: infatti utilizzando questo valore la temperatura risultava sui 60 °C. Allora misurando direttamente dal PIN sulla scheda la VVD, che dovrebbe essere la tensione di riferimento per ADC, questa era pari a 2.9V: ecco perché nel codice sono stati utilizzati 2900 mV anzichè i 3300 mV di Noviello. Questo procedimento probabilmente rientra nel normale processo di calibrazione che varia da scheda a scheda.

Per la temperatura è stata utilizzata la formula:

$$Temp(^{\circ}C) = \frac{V_{SENSE} - V_{25}}{AvgSlope} + 25^{\circ}C$$

con i valori di riferimento della STM32F401RE.

Poi vengono stampati i valori via UART/USART.
Sulla base della temperatura misurata si accende il LED corrispondente.
Infine si esegue un delay alla fine, tramite il ciclo for, semplicemente per avere una frequenza

minore dell'output della routine (in alternativa al ciclo si potrebbe utilizzare una `HAL_Delay` abbassando la priorità dell'interrupt di ADC).

Il calcolo del tempo impiegato dalla routine di risposta all'interrupt generato dal circuito ADC viene eseguito dall'interrupt handler per l'interrupt di ADC, che si trova in `stm32f4xx_it.c`.

```
void ADC_IRQHandler(void)
{
  /* USER CODE BEGIN ADC_IRQn 0 */
  double cycle_time = 1.0/HAL_RCC_GetSysClockFreq();  // tempo di
    esecuzione di un ciclo di clock
  uint32_t start, end, elapsed;
  char msg[34];
  start = DWT->CYCCNT;
  /* USER CODE END ADC_IRQn 0 */
  HAL_ADC_IRQHandler(&hadc1);
  /* USER CODE BEGIN ADC_IRQn 1 */
  end = DWT->CYCCNT;
  elapsed = end - start;  // numero di cicli di clock necessari a gestire l
    'interrupt di ADC
  sprintf(msg, "Routine time = %f micros\r\n\n", elapsed*cycle_time);
  printf(msg);  // output della printf su UART/USART
  /* USER CODE END ADC_IRQn 1 */
}
```

In questo modo viene considerato il tempo complessivo impiegato dalla Interrupt Service Routine ed è un approccio preferibile rispetto a quello di calcolare solo il tempo di esecuzione della callback, perché in questo modo sarebbero andati persi alcuni cicli di clock.
Il codice fa uso del registro `CYCCNT` per calcolare il numero di cicli di clock necessario per eseguire la routine di risposta all'interrupt e moltiplicarlo per il tempo necessario per un singolo ciclo di clock principale (che sulla STM32F401RE Discovery è 84MHz).
In questo caso è stato preferibile utilizzare la "`printf()` modificata" che redireziona l'output su UART/USART, modificando il file `syscall.c`, data l'impossibilità di poter utilizzare `HAL_UART_Transmit()` sulla variabile `UART_HandleTypeDef huart2` (a meno di renderla `external` per aumentarne la visibilità).

# 4 Test

In fase di testing, per verificare il corretto funzionamento del programma, sono state introdotte nell'ambiente circostante la board delle *variazioni termiche* e contemporaneamente si sono temporaneamente modificati i valori limite (19 gradi anziché 18 e 23 gradi anziché 28), per poi ristabilirli.
Per alzare la temperatura è stato sufficiente posizionare un dito direttamente sul sensore termico, che si trova nell'MCU al centro, ed attendere qualche secondo.
Per abbassare la temperatura la board è stata posizionata qualche secondo in refrigeratore: i test sono stati superati con successo.

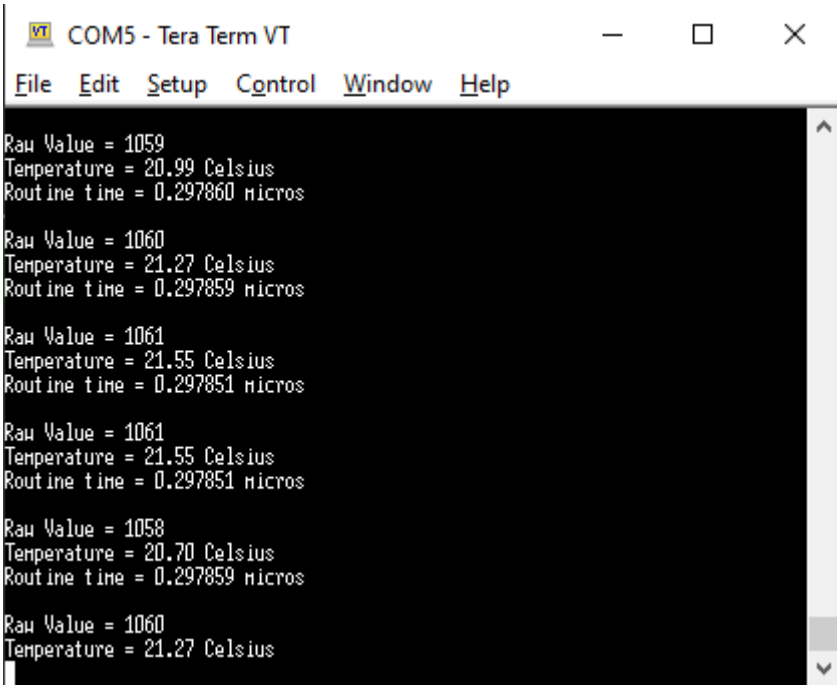


Figura 1: Output del programma