POLITECNICO DI TORINO

A.A. 2021/22

Corso di Progettazione di Dispositivi Biomedici Programmabili

Titolo Progetto: Misuratore di temperatura a contatto

GRUPPO 23

Bennardo William 300947

Bonato Rebecca 300962

Sommario

1.Analisi delle specifiche

Il progetto da noi scelto prevede la realizzazione di un misuratore di temperatura a contatto mediante l'utilizzo del sensore LM35. Analizziamo più nel dettaglio le funzionalità di tale dispositivo.

Il sensore LM35 funziona in modo adeguato nel momento in cui la tensione di alimentazione applicata sia compresa tra i 4V e i 30V. Per tale ragione, il termometro da noi realizzato si occuperà di misurare la tensione di alimentazione circa ogni 1000ms e, mediante una doppia soglia, si preoccuperà di segnalare (per mezzo dell'accensione di un LED) quando la tensione di alimentazione potrebbe compromettere la misura di temperatura. In particolare, tale doppia soglia è così strutturata:

- Se la tensione di alimentazione si trova nel range compreso tra 4,2V e 4V, allora il LED segnalerà tale stato lampeggiando con una frequenza di 5 Hz e duty cycle del 50% (100ms acceso e 100 ms spento).
- Se la tensione di alimentazione scende al di sotto dei 4V allora il LED si accenderà in modo continuo.

La temperatura viene rilevata ogni 500ms e conseguentemente visualizzata su 3 display a 7 segmenti. L'obiettivo è quello di mostrare a schermo una temperatura compresa tra i 30 e i 45°C in modo tale da ottenere una risoluzione di 0,1°C. Nel momento in cui il valore di temperatura è al di fuori del range prestabilito, i tre display si spengono.

Inoltre, il dispositivo consente all'utente di cambiare la visualizzazione di temperatura da gradi Celsius a gradi Fahrenheit e viceversa mediante un pulsante. Disponendo esclusivamente della possibilità di montare su breadboard 3 display a 7 segmenti per questioni di spazio, e volendo mostrare una temperatura adeguata fino a fondo scala, la risoluzione che il termometro garantisce in °F è di 1°F. Il resistore che consente l'attivazione del punto decimale necessario per la visualizzazione in °C deve essere rimosso manualmente una volta che si visualizza il risultato in °F, e successivamente reinserito se intenzionati a ritornare alla situazione iniziale. Per visualizzare il risultato in °F non è necessaria una continua pressione sul pulsante.

Volendo soddisfare le specifiche assegnateci sulla risoluzione in Fahrenheit (0.1°F) e volendo garantire misure di temperatura fino a fondo scala, sarebbero necessarie due azioni aggiuntive:

1. Montaggio su breadboard di un quarto display e di un quarto decodificatore;
2. Amplificazione del segnale di ingresso: disponendo esclusivamente del riferimento interno dell'Arduino (al minimo 1,1V) e sfruttando i 10 bit dell'ADC, non è possibile raggiungere la risoluzione richiesta. Si osservi la seguente tabella implementata con la relazione sotto riportata:

$$Vout\ [mV] = 10 * [(°F - 32) * 5/9]$$

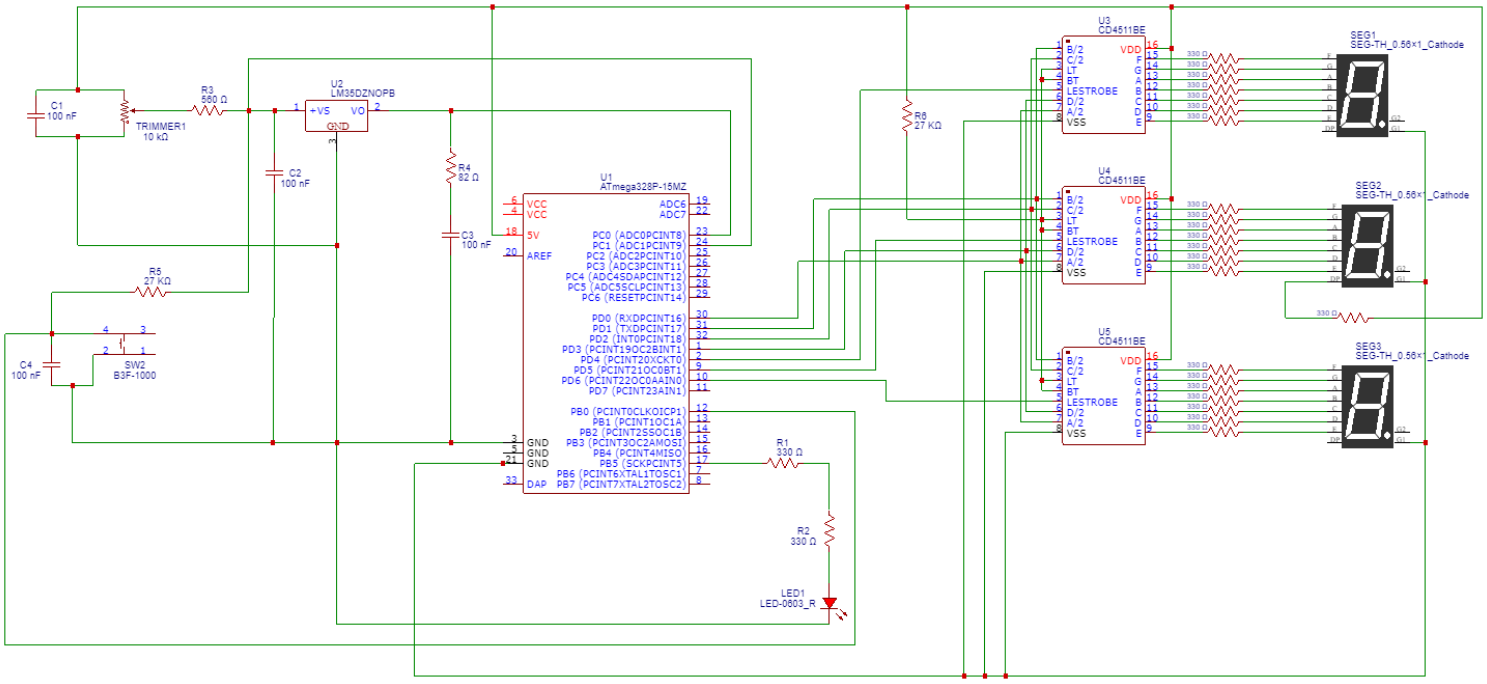
Gradi Fahrenheit [°F]	Gradi Celsius [°C]	Tensione associata [mV]
86	30	300
86,1	30,056	300,56
86,2	30,111	301,11

È necessario che l'ADC apprezzi una variazione di tensione almeno di 0,5 mV. Questo sarebbe possibile introducendo un guadagno di almeno 6dB.

2. Analisi dell'Hardware

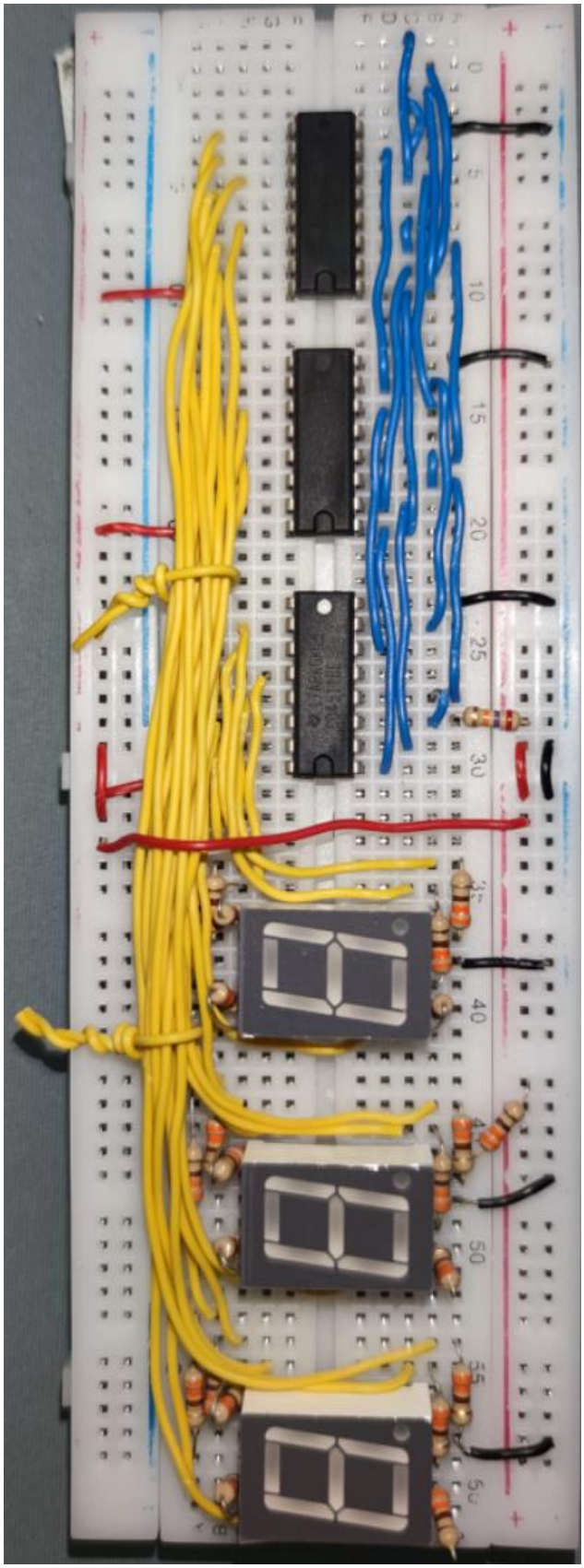
2.1 Schema Elettrico

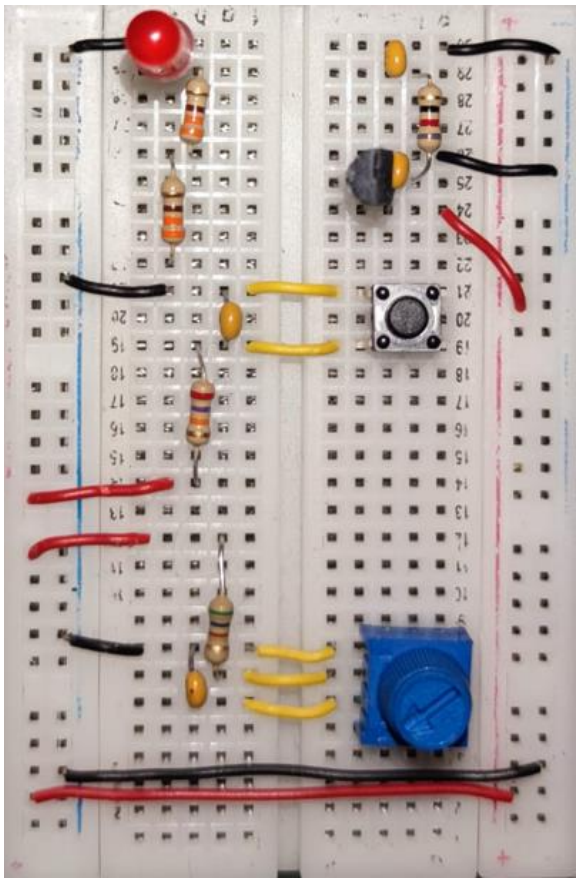
L'utilizzo del trimmer ha lo scopo di modulare la tensione di alimentazione, cosa altrimenti impossibile senza un generatore di tensione. Di conseguenza, per simulare meglio il comportamento del circuito in laboratorio, abbiamo scelto di collegare alla tensione di alimentazione modulata (all'uscita del trimmer) il sensore e il pulsante. Abbiamo preferito, invece, portare ai display a sette segmenti una tensione di alimentazione fissa a 5V: mediante i display è possibile analizzare il comportamento del sensore con diverse tensioni di alimentazione, è quindi opportuno che funzionino correttamente. Viceversa, non saremmo stati in grado di distinguere una lettura scorretta per il mal funzionamento del display o del sensore di temperatura.

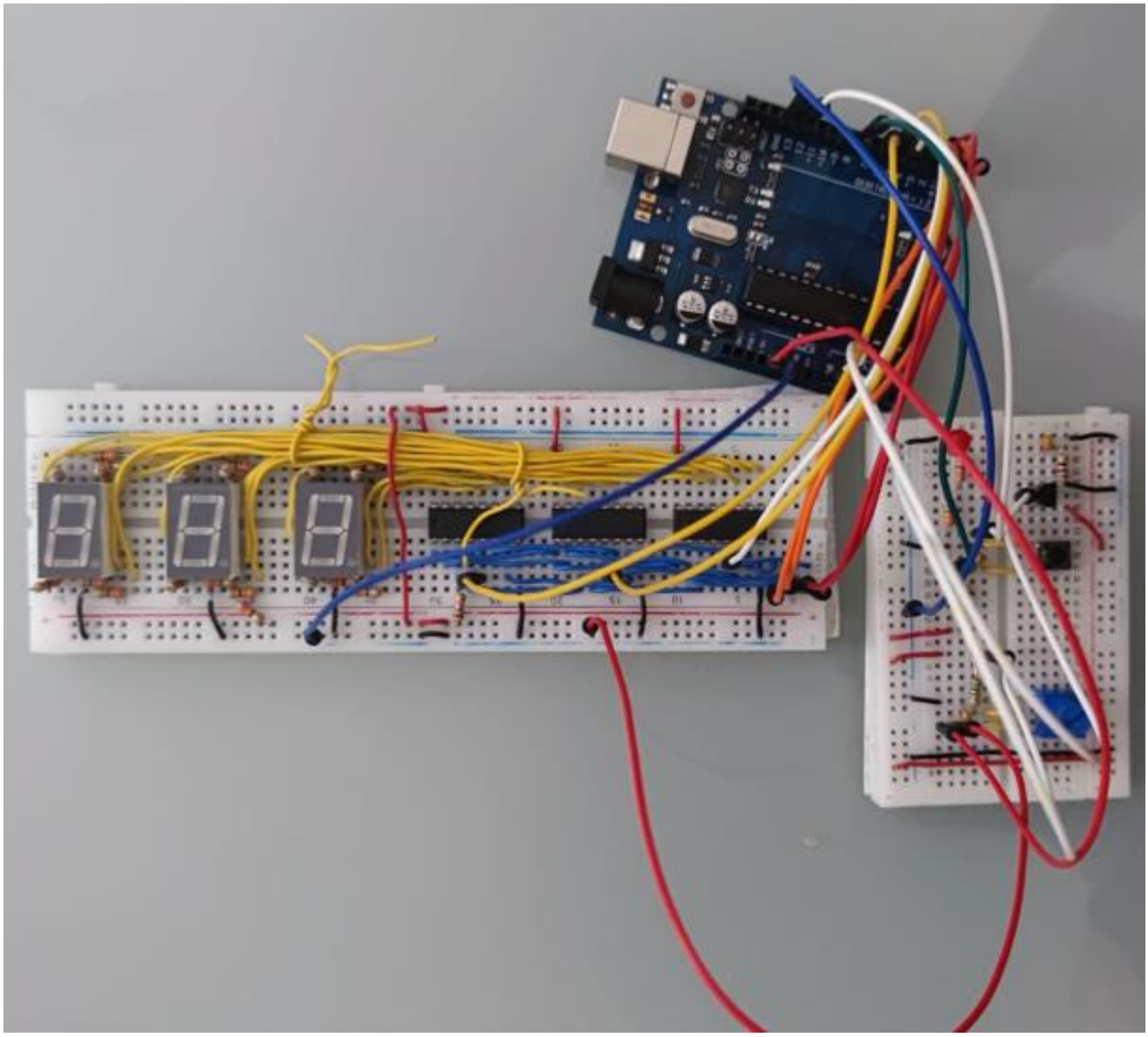


2.2 Immagini circuiti realizzati

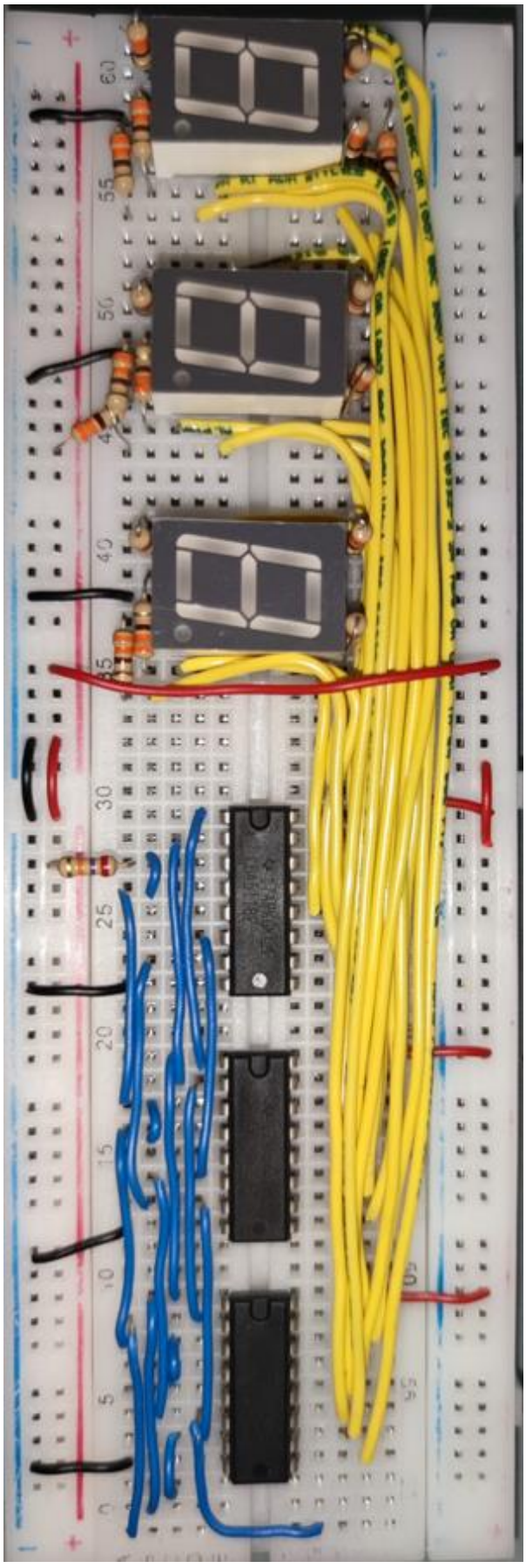
Circuiti di Bennardo William:

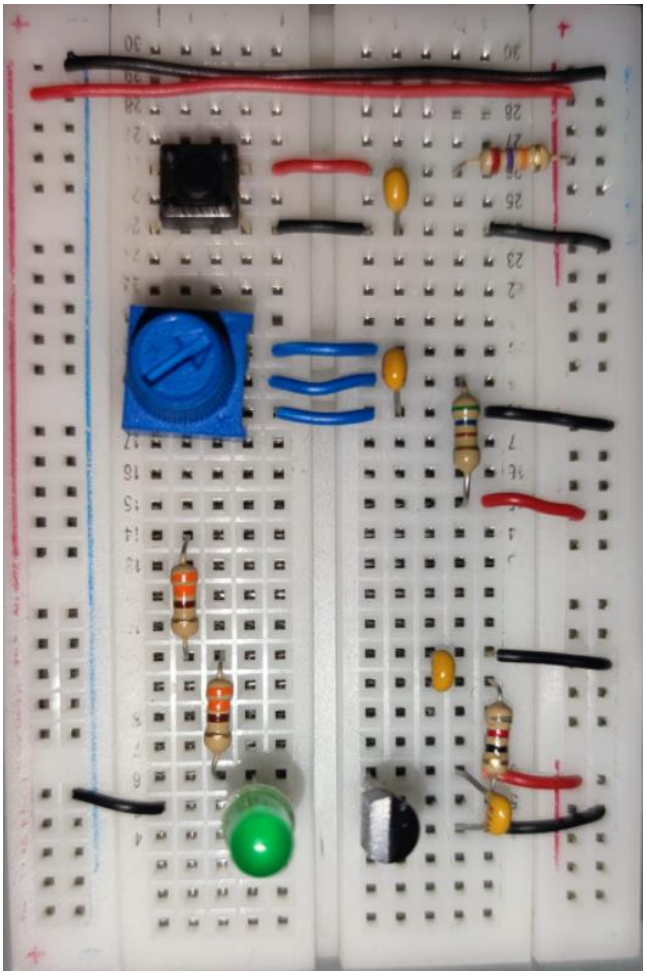


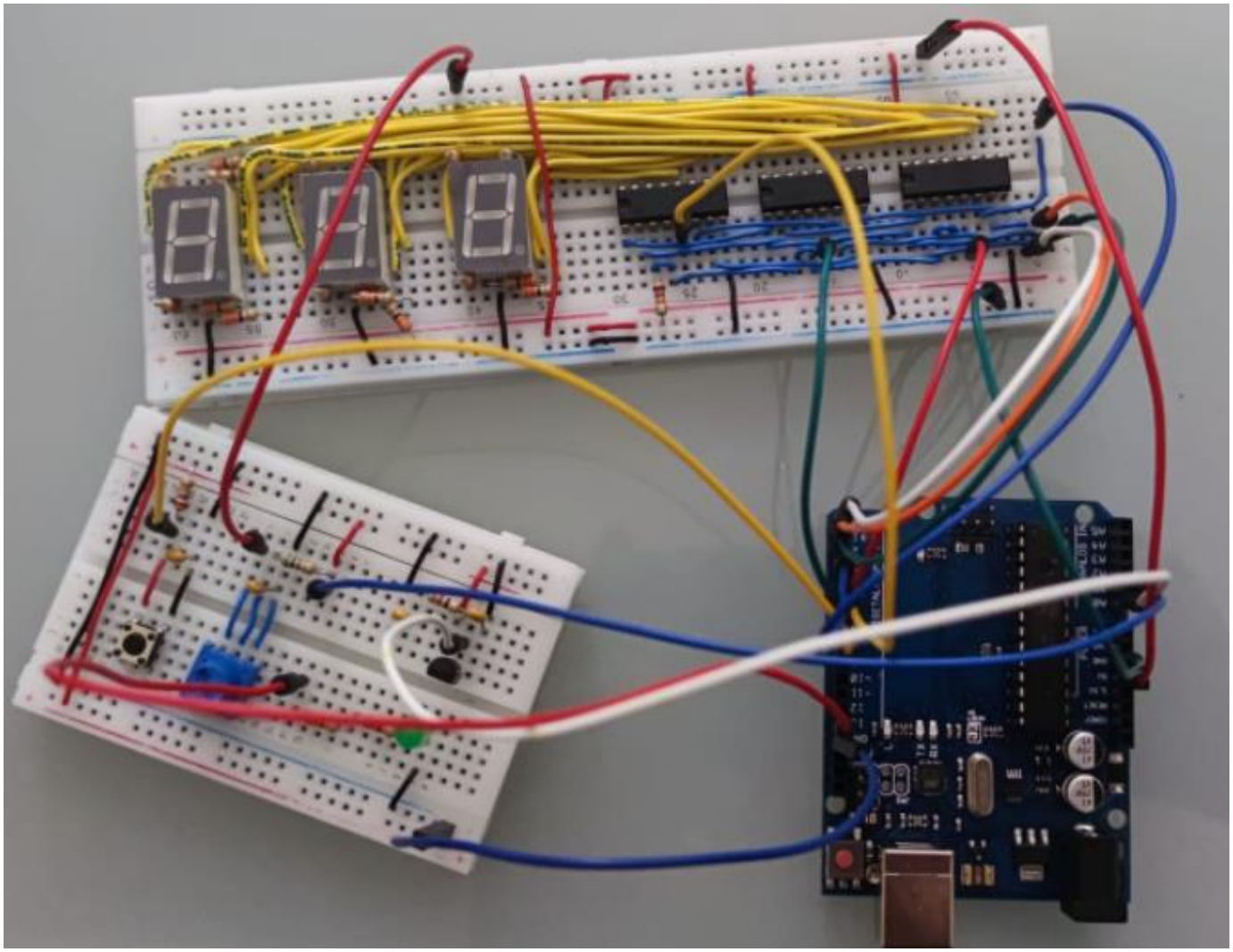




Circuiti di Bonato Rebecca







3. Analisi del software

3.1 Scelta delle temporizzazioni

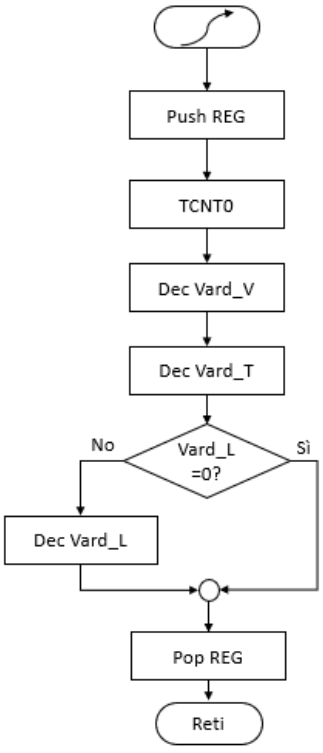
Per raggiungere gli obiettivi prefissati nelle specifiche, il software dovrà gestire delle temporizzazioni. Per implementare tali temporizzazioni abbiamo utilizzato il timer counter zero e il relativo interrupt in caso di overflow. Le scelte effettuate per raggiungere le temporizzazioni desiderate sono le seguenti:

• Frequenza di lavoro del sistema	2 MHz
• Frequenza di lavoro del timer Counter: prescaler passo 1024	2 KHz
• Valore iniziale di conteggio del contatore	246

Di conseguenza, il tempo che intercorre tra un interrupt e il successivo è pari a circa 10 ms. Sfruttando una variabile per ogni temporizzazione necessaria, abbiamo inizializzato le variabili come segue:

Azione da svolgere	Temporizzazione necessaria	Inizializzazione variabile
Campionamento tensione di alimentazione	1 s	Vard_V = 100
Campionamento tensione Vout LM35	500 ms	Vard_T = 50
Inversione dello stato del LED*	100 ms	Vard_L = 10

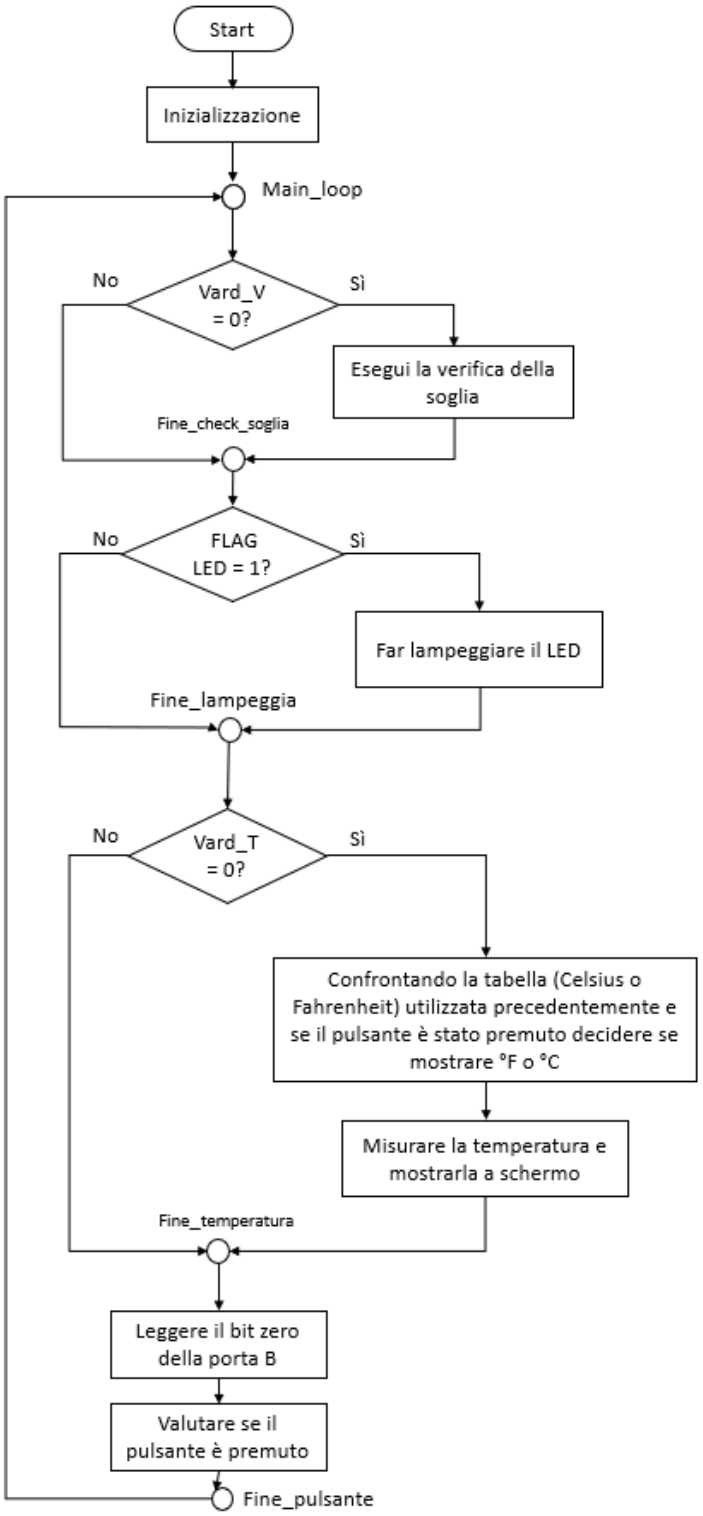
*La variabile viene decrementata solo se la tensione è compresa tra 4,2 e 4 V



SCHEMA A BLOCCHI SUBROUTINE
DI RISPOSTA AD INTERRUPT

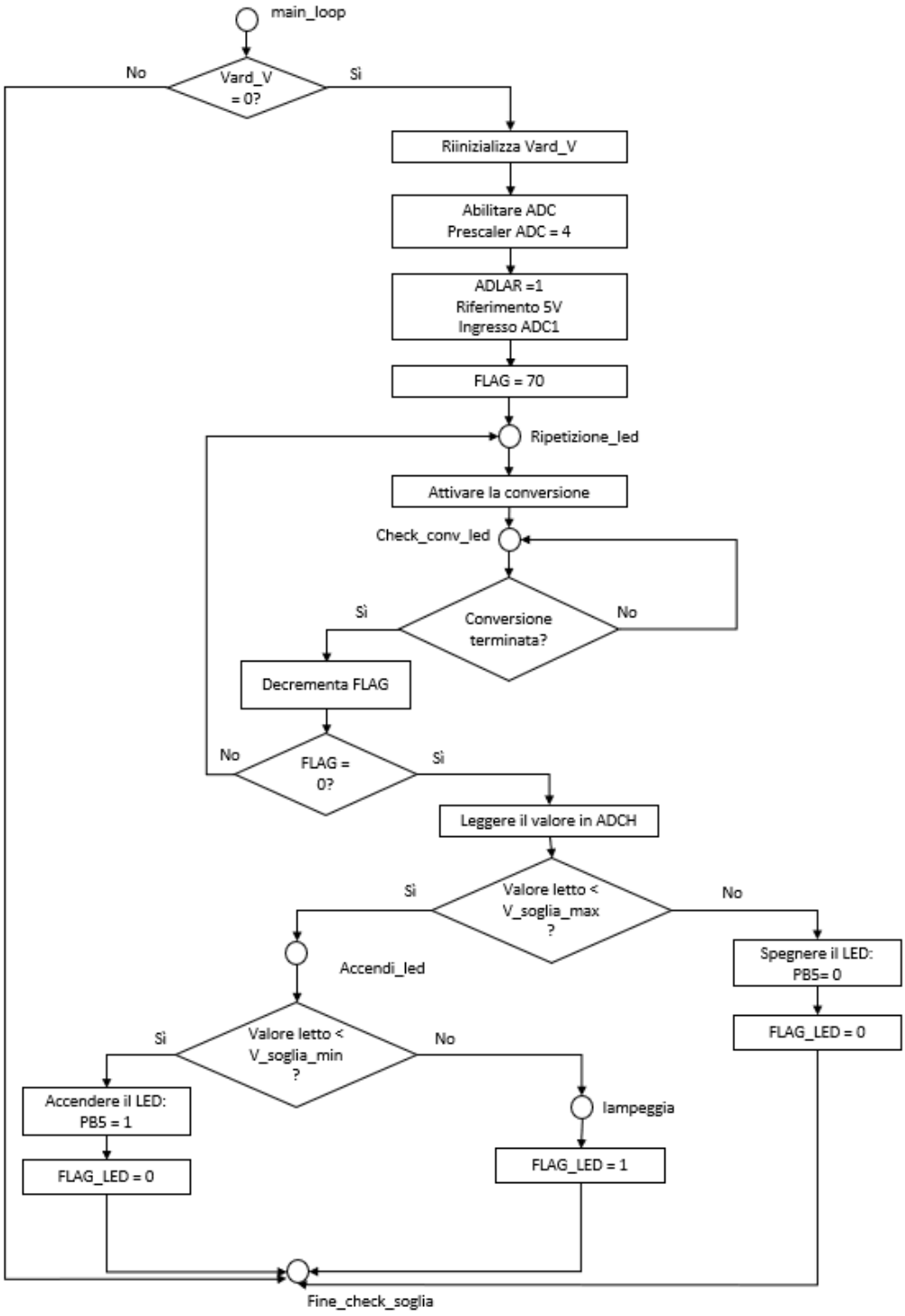
3.2 Analisi del main_loop

Nel main loop viene verificato ad ogni ciclo se le variabili che tengono conto della temporizzazione risultano pari a zero. Viceversa, l'analisi dello stato del pulsante viene svolta senza una temporizzazione da noi stabilita, ma ad ogni ciclo.



SCHEMA A BLOCCHI MAIN LOOP

3.3 Analisi della tensione di alimentazione e accensione del LED



SCHEMA A BLOCCHI CAMPIONAMENTO TENSIONE DI ALIMENTAZIONE E ACCENSIONE O SPEGNIMENTO DEL LED

La tensione di alimentazione è stata fatta variare mediante l'ausilio di un trimmer in grado di fornire in uscita tensioni nel range 0-5V. Conseguentemente, è stato scelto di utilizzare l'ADC al microcontrollore con Vref interna pari a 5V in modo tale da coprire tutta la dinamica. La risoluzione necessaria per raggiungere le specifiche è di 0,1V ed è ottenibile con un numero di bit pari ad 8: da qui la scelta di giustificare a sinistra.

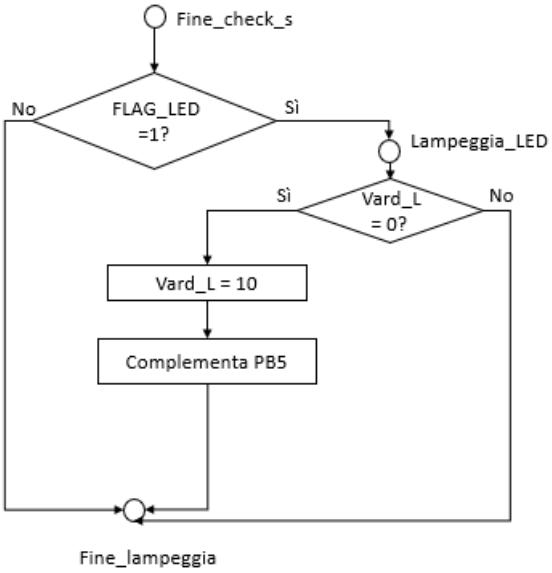
Pima di leggere il segnale campionato, vengono eseguiti dei cicli di campionamento a vuoto. Ciò è necessario per permettere la stabilizzazione dell'ADC in seguito ad una variazione del riferimento: in questo modo sono garantite misure più accurate.

I valori delle soglie con cui la tensione in ingresso viene confrontata vengono così determinati:

$$ADC = \frac{Vin*256}{5V}$$

Vin	**ADC**
V_soglia_min (4V)	205
V_soglia_max (4,2V)	230

L'accensione continua del LED (se sotto la soglia minima) o il suo spegnimento (se sopra la soglia massima) vengono gestiti all'interno del ramo Vard_V= 0. Ciò non può avvenire per il lampeggio del LED che richiede una temporizzazione e in quanto tale va gestito nel main loop. Nel main_loop l'inversione della linea che pilota lo stato del LED viene effettuata esclusivamente se nel ramo Vard_V=0 è stato impostato un apposito flag (FLAG_LED) che sancisce che il valore di tensione letto si trova tra le due soglie.



SCHEMA A BLOCCHI LAMPEGGIO DEL LED ALLA FREQUENZA DI 5HZ

3.4 Analisi del sensore LM35 e visualizzazione su display a 7 segmenti

La misura della temperatura e la conseguente visualizzazione a schermo viene gestita, sia nel caso di visualizzazione in Celsius che nel caso di visualizzazione in Fahrenheit, dalla subroutine measure. La risoluzione con cui si vuole visualizzare la temperatura è di 0,1°C e di 1°F, mentre la sensibilità del sensore LM35 è di 10mV/°C.
Per ottenere la risoluzione richiesta in Celsius è stato necessario sfruttare il riferimento interno dell'ADC a 1.1 V e tutti i 10 bit che l'ADC mette a disposizione per la rappresentazione del risultato (giustificare a destra): è infatti necessario apprezzare variazioni di 1mV della tensione in ingresso all'ADC.
La tensione in uscita dal sensore può essere relazionata alla corrispondente temperatura in °C nel seguente modo:

$$Vout = 10\frac{mV}{°C} * T_{°C} \qquad 3.4.1$$

Mentre una volta ottenuta la temperatura in °C, la corrispondente temperatura in °F si ricava come:

$$T_{°F} = \left(\frac{9}{5} * T_{°C}\right) + 32 \qquad 3.4.2$$

Effettuando un cambio di riferimento (da 5V a 1,1V) è opportuno anche in questo caso permettere all'ADC di stabilizzarsi eseguendo alcune letture a vuoto per ottenere misure attendibili.

Una volta che la misura di tensione in uscita dal sensore LM35 è stata campionata, è necessario mostrare a schermo il corretto valore di temperatura a cui la tensione corrisponde. È opportuno, al fine di giustificare quanto esplicitato nel diagramma a blocchi e nel codice, spendere qualche parola sulla realizzazione delle LUT.
Abbiamo scelto di realizzare due LUT per la visualizzazione in Celsius e in Fahrenheit, entrambe caratterizzate da righe da 4 byte così strutturate:

Valore numerico prodotto in uscita dall'ADC che funge da riferimento	Cifra da visualizzare sul display di sinistra	Cifra da visualizzare sul display centrale	Cifra da visualizzare sul display di destra

La prima colonna risulta indifferente nelle due tabelle e corrisponde al valore numerico prodotto in uscita dall'ADC:

$$ADC = \frac{Vin * 1024}{1{,}1V} \qquad 3.4.3$$

A seconda che si voglia visualizzare il risultato in °C o in °F, le rimanenti caselle della tabella saranno compilate sfruttando la relazione relativa alla sensibilità del sensore e alla conversione °C-°F precedentemente riportate.

In tabella riportiamo il valore minimo e il valore massimo della LUT che corrispondono alla prima e all'ultima riga:

Temperatura °C	Temperatura °F	Vin*	ADC
29,97 °C	85,95 °F	0,2997 V	279
45,01°C	113,02 °F	0,4501 V	419

*La tensione in ingresso all'ADC (Vin) corrisponde alla tensione in uscita dal sensore calcolata sfruttando la sensibilità

Una volta verificato che il valore fornito dall'ADC sia compreso tra 279 e 419, il programma deve fare in modo di puntare alla corretta riga della tabella.
Le due LUT sono state realizzate su un foglio Excel mediante il seguente modus operandi:

1. Vengono riportati i livelli da 0 a 1023
2. Invertendo la relazione 3.4.3 viene ricavata la Vin corrispondente a ciascun livello
3. Invertendo la relazione 3.4.1 viene ricavata la temperatura in °C e approssimata per difetto o per eccesso in modo tale da avere una risoluzione di 0,1 °C
4. Viene ricavata la temperatura in °F a partire dalla temperatura in °C (non ancora approssimata); conseguentemente la temperatura in °F viene approssimata in modo tale da ottenere una risoluzione di 1°F.

Quando la subroutine measure viene richiamata, gli viene fornito in ingresso l'indirizzo della prima cella della tabella (ZH e ZL) che deve sfruttare per la visualizzazione della temperatura (°C o °F). Sottraendo il valore fornito in uscita dall'ADC al valore minimo contenuto nella prima colonna della LUT (279), si trova il numero di righe che il puntatore deve scorrere per giungere alla riga della tabella corrispondente al valore da mostrare. Tenendo presente che ogni riga è composta da 4 colonne, il puntatore dovrà spostarsi di:

$$(Valore\ ADC - 279) * 4$$

La relazione complessiva che porta all'indirizzo corretto diventa:

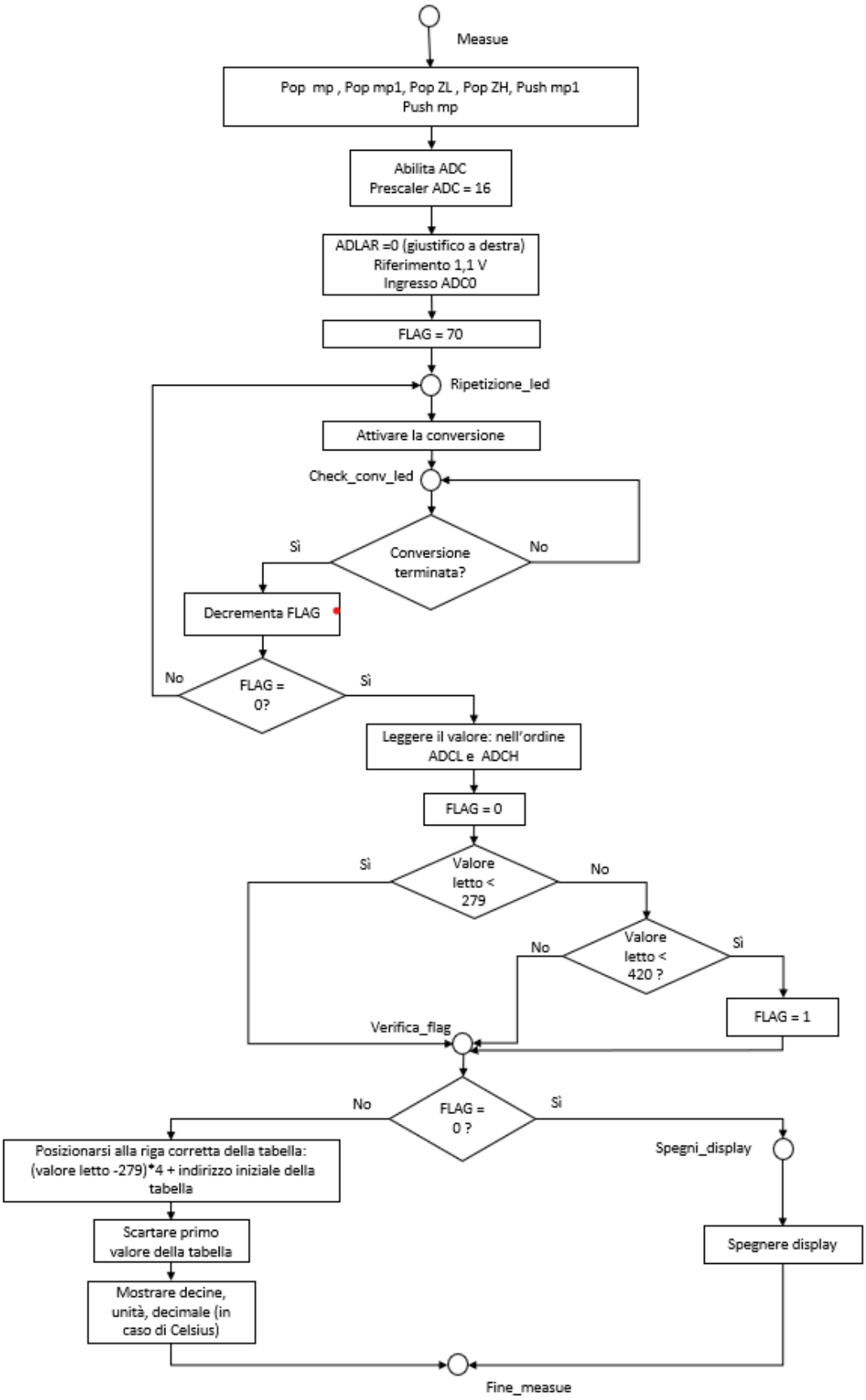
$$Indirizzo\ iniziale\ tabella + (Valore\ ADC - 279) * 4$$

Per quanto riguarda la sottrazione menzionata precedentemente è necessario fare un appunto: il registro che contiene i 2 bit più significativi del valore prodotto dall'ADC, nel range di tensioni scelte, sarà sempre definito come di seguito riportato:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	1

Di conseguenza non è stato necessario effettuare la sottrazione di tale valore con la parte alta di 279 poiché produce sempre 0.
A questo punto, seguono le istruzioni che permettono la visualizzazione sui 3 display a 7 segmenti.

Se invece il risultato della conversione fosse stato al di fuori del range prestabilito, il programma prevede lo spegnimento dei display a sette segmenti.



A SCHEMA A BLOCCHI DELLA SUBROUTINE MEASURE

3.5 Gestione del pulsante

Nel paragrafo precedente non è stata definita la scelta della LUT corretta tra °C e °F. La scelta viene effettuata in questa parte del programma. Il micropulsante è costituito da un interruttore che in condizione di riposo è aperto (1 logico), mentre necessita di una pressione continua per chiudersi (0 logico): si tratta infatti di un pulsante momentaneo. La nostra scelta è stata quella di determinare un cambiamento nella visualizzazione della temperatura (da °C a °F e viceversa) ogni volta che il pulsante viene premuto.
Possiamo dividere il ragionamento fatto in due parti:

1. Analisi dello stato del pulsante per definire se è necessario cambiare la visualizzazione della LUT precedente
2. Cambio della LUT se necessario

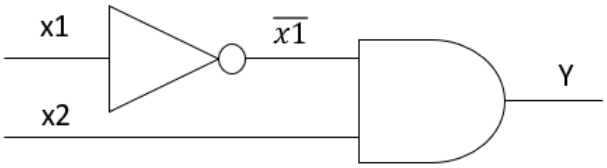
3.5.1 Analisi stato del pulsante

Nel main_loop viene costantemente monitorato lo stato del pulsante. In particolare, nel momento in cui il pulsante è a riposo viene letto in PB0 un 1 logico, mentre quando viene premuto la lettura è di uno 0 logico. Inizializzando una variabile che tenga conto dello stato del pulsante alla lettura precedente, voglio realizzare la tavola della verità sotto riportata. Per una lettura più immediata si tenga conto che:

- STATO_TABELLA = 1: deve essere cambiata la visualizzazione della LUT
- STATO_TABELLA = 0: non deve essere cambiata la visualizzazione della LUT

Stato pulsante precedente (x2)	Stato pulsante Attuale (x1)	STATO_TABELLA (y)	Descrizione
1	1	0	Il pulsante non era premuto e non è stato premuto: la LUT non deve essere cambiata
1	0	1	Il pulsante non era premuto ed è stato premuto: la LUT deve essere cambiata
0	1	0	Il pulsante era stato premuto e viene rilasciato: la LUT non deve essere cambiata
0	0	0	Il pulsate era premuto e si sta continuando a fare pressione: la LUT non deve essere cambiata

Tale tavola della verità viene implementata grazie al seguente circuito combinatorio.
Affianco ad esso la corrispondente tavola della verità in cui sono resi espliciti tutti i passaggi.



x1	x2	$\overline{x1}$	y
1	1	0	0
0	1	1	1
1	0	0	0
0	0	1	0

Il diagramma a blocchi che schematizza l'analisi dello stato del pulsante è il seguente

A SCHEMA A BLOCCHI DELLA LETTURA DEL PULSANTE

3.5.2 Scelta della LUT

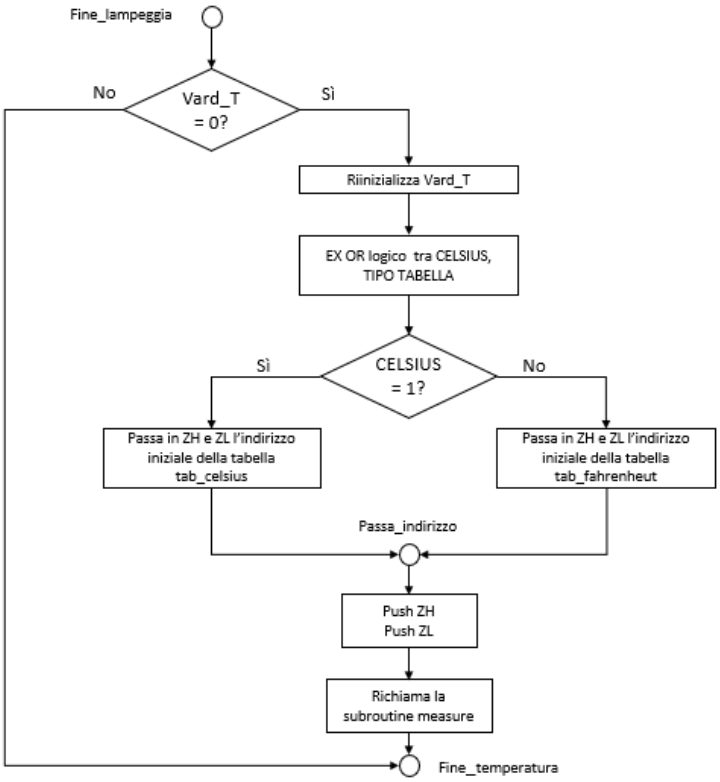
Una volta definito se la LUT deve essere cambiata rispetto allo stato precedente, è necessario determinare l'indirizzo iniziale di quale LUT deve essere fornito alla subroutine measure. Per tale scopo si deve inizializzare una variabile che tenga conto del fatto che la temperatura precedente è stata visualizzata in °C o in °F. Tale variabile è stata denominata Celsius e ha il seguente significato:

- CELSIUS =1: temperatura visualizzata in Celsius
- CELSIUS = 0: temperatura visualizzata in Fahrenheit

Abbiamo implementato la funzione logica ex-or per raggiungere l'obiettivo:

STATO_TABELLA (x1)	CELSIUS (x2)	CELSIUS (y)
1	1	0
1	0	1
0	1	1
0	0	0

Questa operazione viene effettuata nel momento in cui la variabile di temporizzazione Vard_T è pari a zero cioè quando si deve effettuare la misura di temperatura e visualizzarla a schermo.



SCHEMA A BLOCCHI RAMO VARD_T = 0

4. Codice

```
.DEF mp = R16            ; registro di lavoro generico
.DEF mp1 = R17           ; registro di lavoro generico
.DEF Vard_V = R18        ; registro di conteggio
.DEF val_prec = R19      ; registro che tiene conto del valore precedentemente letto del
pulsante
.DEF Vard_T = R20        ; registro di conteggio
.DEF FLAG = R21
.DEF Vard_L = R22        ; registro di conteggio
.DEF FLAG_LED = R23
.DEF CELSIUS = R24
.DEF TIPO_TABELLA = R25
;
;
.EQU T_MIN = 279         ; 30,0 gradi Celsius
.EQU T_MAX = 420         ; 45,1 gradi Celsius
.EQU V_SOGLIA_max = 230      ; tale valore corrisponde alla tensione 4,2V convertita in 8
bit su una tensione del convertitore analogico digitale di 5V
.EQU V_SOGLIA_min = 205  ; tale valore corrisponde a 4V, soglia sotto la quale il sensore
non è opportunamente alimentato.
;
;
.CSEG
.ORG   0x1FFF               ; definisce l'inizio della tabella che sarà scritta in flash
;
tab_celsius:
.db    279,   3,0,0
.db    280,   3,0,1
.db    281,   3,0,2
.db    282,   3,0,3
.db    283,   3,0,4
.db    284,   3,0,5
.db    285,   3,0,6
.db    286,   3,0,7
.db    287,   3,0,8
.db    288,   3,0,9
.db    289,   3,1,0
.db    290,   3,1,2
.db    291,   3,1,3
.db    292,   3,1,4
.db    293,   3,1,5
.db    294,   3,1,6
.db    295,   3,1,7
.db    296,   3,1,8
.db    297,   3,1,9
.db    298,   3,2,0
.db    299,   3,2,1
.db    300,   3,2,2
.db    301,   3,2,3
.db    302,   3,2,4
.db    303,   3,2,5
.db    304,   3,2,7
.db    305,   3,2,8
.db    306,   3,2,9
.db    307,   3,3,0
.db    308,   3,3,1
.db    309,   3,3,2
.db    310,   3,3,3
.db    311,   3,3,4
.db    312,   3,3,5
```

```
.db    313,   3,3,6
.db    314,   3,3,7
.db    315,   3,3,8
.db    316,   3,3,9
.db    317,   3,4,1
.db    318,   3,4,2
.db    319,   3,4,3
.db    320,   3,4,4
.db    321,   3,4,5
.db    322,   3,4,6
.db    323,   3,4,7
.db    324,   3,4,8
.db    325,   3,4,9
.db    326,   3,5,0
.db    327,   3,5,1
.db    328,   3,5,2
.db    329,   3,5,3
.db    330,   3,5,4
.db    331,   3,5,6
.db    332,   3,5,7
.db    333,   3,5,8
.db    334,   3,5,9
.db    335,   3,6,0
.db    336,   3,6,1
.db    337,   3,6,2
.db    338,   3,6,3
.db    339,   3,6,4
.db    340,   3,6,5
.db    341,   3,6,6
.db    342,   3,6,7
.db    343,   3,6,8
.db    344,   3,7,0
.db    345,   3,7,1
.db    346,   3,7,2
.db    347,   3,7,3
.db    348,   3,7,4
.db    349,   3,7,5
.db    350,   3,7,6
.db    351,   3,7,7
.db    352,   3,7,8
.db    353,   3,7,9
.db    354,   3,8,0
.db    355,   3,8,1
.db    356,   3,8,2
.db    357,   3,8,3
.db    358,   3,8,5
.db    359,   3,8,6
.db    360,   3,8,7
.db    361,   3,8,8
.db    362,   3,8,9
.db    363,   3,9,0
.db    364,   3,9,1
.db    365,   3,9,2
.db    366,   3,9,3
.db    367,   3,9,4
.db    368,   3,9,5
.db    369,   3,9,6
.db    370,   3,9,7
.db    371,   3,9,9
.db    372,   4,0,0
.db    373,   4,0,1
.db    374,   4,0,2
.db    375,   4,0,3
.db    376,   4,0,4
```

```
.db    377,   4,0,5
.db    378,   4,0,6
.db    379,   4,0,7
.db    380,   4,0,8
.db    381,   4,0,9
.db    382,   4,1,0
.db    383,   4,1,1
.db    384,   4,1,3
.db    385,   4,1,4
.db    386,   4,1,5
.db    387,   4,1,6
.db    388,   4,1,7
.db    389,   4,1,8
.db    390,   4,1,9
.db    391,   4,2,0
.db    392,   4,2,1
.db    393,   4,2,2
.db    394,   4,2,3
.db    395,   4,2,4
.db    396,   4,2,5
.db    397,   4,2,6
.db    398,   4,2,8
.db    399,   4,2,9
.db    400,   4,3,0
.db    401,   4,3,1
.db    402,   4,3,2
.db    403,   4,3,3
.db    404,   4,3,4
.db    405,   4,3,5
.db    406,   4,3,6
.db    407,   4,3,7
.db    408,   4,3,8
.db    409,   4,3,9
.db    410,   4,4,0
.db    411,   4,4,2
.db    412,   4,4,3
.db    413,   4,4,4
.db    414,   4,4,5
.db    415,   4,4,6
.db    416,   4,4,7
.db    417,   4,4,8
.db    418,   4,4,9
.db    419,   4,5,0
;
;
tab_fahrenheit:
;
       .db    279,   0,     8,     6
       .db    280,   0,     8,     6
       .db    281,   0,     8,     6
       .db    282,   0,     8,     7
       .db    283,   0,     8,     7
       .db    284,   0,     8,     7
       .db    285,   0,     8,     7
       .db    286,   0,     8,     7
       .db    287,   0,     8,     7
       .db    288,   0,     8,     8
       .db    289,   0,     8,     8
       .db    290,   0,     8,     8
       .db    291,   0,     8,     8
       .db    292,   0,     8,     8
       .db    293,   0,     8,     9
       .db    294,   0,     8,     9
       .db    295,   0,     8,     9
```

```
.db    296,   0,     8,     9
.db    297,   0,     8,     9
.db    298,   0,     9,     0
.db    299,   0,     9,     0
.db    300,   0,     9,     0
.db    301,   0,     9,     0
.db    302,   0,     9,     0
.db    303,   0,     9,     1
.db    304,   0,     9,     1
.db    305,   0,     9,     1
.db    306,   0,     9,     1
.db    307,   0,     9,     1
.db    308,   0,     9,     2
.db    309,   0,     9,     2
.db    310,   0,     9,     2
.db    311,   0,     9,     2
.db    312,   0,     9,     2
.db    313,   0,     9,     3
.db    314,   0,     9,     3
.db    315,   0,     9,     3
.db    316,   0,     9,     3
.db    317,   0,     9,     3
.db    318,   0,     9,     3
.db    319,   0,     9,     4
.db    320,   0,     9,     4
.db    321,   0,     9,     4
.db    322,   0,     9,     4
.db    323,   0,     9,     4
.db    324,   0,     9,     5
.db    325,   0,     9,     5
.db    326,   0,     9,     5
.db    327,   0,     9,     5
.db    328,   0,     9,     5
.db    329,   0,     9,     6
.db    330,   0,     9,     6
.db    331,   0,     9,     6
.db    332,   0,     9,     6
.db    333,   0,     9,     6
.db    334,   0,     9,     7
.db    335,   0,     9,     7
.db    336,   0,     9,     7
.db    337,   0,     9,     7
.db    338,   0,     9,     7
.db    339,   0,     9,     8
.db    340,   0,     9,     8
.db    341,   0,     9,     8
.db    342,   0,     9,     8
.db    343,   0,     9,     8
.db    344,   0,     9,     9
.db    345,   0,     9,     9
.db    346,   0,     9,     9
.db    347,   0,     9,     9
.db    348,   0,     9,     9
.db    349,   0,     9,     9
.db    350,   1,     0,     0
.db    351,   1,     0,     0
.db    352,   1,     0,     0
.db    353,   1,     0,     0
.db    354,   1,     0,     0
.db    355,   1,     0,     1
.db    356,   1,     0,     1
.db    357,   1,     0,     1
.db    358,   1,     0,     1
.db    359,   1,     0,     1
```

```
        .db    360,   1,     0,     2
        .db    361,   1,     0,     2
        .db    362,   1,     0,     2
        .db    363,   1,     0,     2
        .db    364,   1,     0,     2
        .db    365,   1,     0,     3
        .db    366,   1,     0,     3
        .db    367,   1,     0,     3
        .db    368,   1,     0,     3
        .db    369,   1,     0,     3
        .db    370,   1,     0,     4
        .db    371,   1,     0,     4
        .db    372,   1,     0,     4
        .db    373,   1,     0,     4
        .db    374,   1,     0,     4
        .db    375,   1,     0,     5
        .db    376,   1,     0,     5
        .db    377,   1,     0,     5
        .db    378,   1,     0,     5
        .db    379,   1,     0,     5
        .db    380,   1,     0,     5
        .db    381,   1,     0,     6
        .db    382,   1,     0,     6
        .db    383,   1,     0,     6
        .db    384,   1,     0,     6
        .db    385,   1,     0,     6
        .db    386,   1,     0,     7
        .db    387,   1,     0,     7
        .db    388,   1,     0,     7
        .db    389,   1,     0,     7
        .db    390,   1,     0,     7
        .db    391,   1,     0,     8
        .db    392,   1,     0,     8
        .db    393,   1,     0,     8
        .db    394,   1,     0,     8
        .db    395,   1,     0,     8
        .db    396,   1,     0,     9
        .db    397,   1,     0,     9
        .db    398,   1,     0,     9
        .db    399,   1,     0,     9
        .db    400,   1,     0,     9
        .db    401,   1,     1,     0
        .db    402,   1,     1,     0
        .db    403,   1,     1,     0
        .db    404,   1,     1,     0
        .db    405,   1,     1,     0
        .db    406,   1,     1,     1
        .db    407,   1,     1,     1
        .db    408,   1,     1,     1
        .db    409,   1,     1,     1
        .db    410,   1,     1,     1
        .db    411,   1,     1,     1
        .db    412,   1,     1,     2
        .db    413,   1,     1,     2
        .db    414,   1,     1,     2
        .db    415,   1,     1,     2
        .db    416,   1,     1,     2
        .db    417,   1,     1,     3
        .db    418,   1,     1,     3
        .db    419,   1,     1,     3
;
;
.CSEG
```

```
.ORG 0x0000                                          ; definisce l'inizio del codice
all'indirizzo 0x0000 (obbligatorio)
;
;
; INTERRUPT VECTORS FOLLOW
;
       jmp RESET                          ; vector 1:          Reset Handler
       jmp EXT_INT0                       ; vector 2:          IRQ0 Handler
       jmp EXT_INT1                       ; vector 3:          IRQ1 Handler
       jmp PCINTR0                        ; vector 4:          PCINT0 Handler
       jmp PCINTR1                        ; vector 5:          PCINT1 Handler
       jmp PCINTR2                        ; vector 6:          PCINT2 Handler
       jmp WDT                            ; vector 7:          Watchdog timer handler
       jmp TIM2_COMPA                     ; vector 8:          Timer2 Compare A handler
       jmp TIM2_COMPB                     ; vector 9:          Timer2 compare B handler
       jmp TIM2_OVF                       ; vector 10:  Timer2 Overflow Handler
       jmp TIM1_CAPT                      ; vector 11:  Timer1 Capture Handler
       jmp TIM1_COMPA                     ; vector 12:  Timer1 CompareA Handler
       jmp TIM1_COMPB                     ; vector 13:  Timer1 CompareB Handler
       jmp TIM1_OVF                       ; vector 14:  Timer1 Overflow Handler
       jmp TIM0_COMPA                     ; vector 15:  Timer 0 CompareA handler
       jmp TIM0_COMPB                     ; vector 16:  Timer 0 CompareB handler
       jmp TIM0_OVF                       ; vector 17:  Timer0 Overflow Handler
       jmp SPI_STC                        ; vector 18:  SPI Transfer Complete Handler
       jmp USART_RXC                      ; vector 19:  USART RX Complete Handler
       jmp USART_UDRE                     ; vector 20:  USART UDR Empty Handler
       jmp USART_TXC                      ; vector 21:  USART TX Complete Handler
       jmp ADC_conv                       ; vector 22:  ADC Conversion Complete Handler
       jmp EE_RDY                         ; vector 23:  EEPROM Ready Handler
       jmp ANA_COMP                       ; vector 24:  Analog Comparator Handler
       jmp TWSI                           ; vector 25:  Two-wire Serial Interface Handler
       jmp SPM_RDY                        ; vector 26:  Store Program Memory Ready Handler
;
; END OF INTERRUPT VECTORS
;
;
RESET:
; In questo punto inizia la routine principale del programma
;
; Inizializzazione dello stack pointer
;
   ldi mp, high(RAMEND)
   out SPH, mp
   ldi mp, low(RAMEND)
   out SPL, mp
;
; Programmazione porte necessarie
;
; Programmo la linea 5 della porta B in uscita
;
   ldi mp1, 0b0010_0000
   out DDRB, mp1
;
; Inizio con il LED spento e imposto il pull-up sulle altre linee
;
   in mp, DDRB
   ldi mp1, 0b1111_1111
   eor mp, mp1
   andi mp, 0b1101_1111
   out PORTB, mp
;
; Inizializzazione in uscita dei primi 7 bit della porta D. PD0 - PD3 = ABCD; PD4 = LE
decine; PD5 = LE unità; PD6 = LE cifra decimale.
```

```
   ldi mp,0b0111_1111
   out DDRD,mp
;
   ldi mp, 0b0111_0000                  ; mette a 1 i LE dei 4511 (condizione di memoria)
   out PORTD, mp
;
;
; Divido la frequenza di clock (16 MHz) per 8
;
   ldi mp, 0b1000_0000
   sts CLKPR, mp
   ldi mp, 0b0000_0100
   sts CLKPR, mp
;
; Seleziono un prescaler passo 1024
;
   ldi mp, 0b0000_0101
   out TCCR0B, mp
;
; Seleziono tempo tra interrupt pari a 10 ms
   ldi mp, 246
   out TCNT0, mp
;
; Abilito l'interrupt in caso di overflow
   ldi mp,0b0000_0001
   sts TIMSK0, mp
;
; Inizializzo le variabili che verranno decrementate nella subroutine di risposta ad
interrupt
   ldi Vard_V, 100
   ldi Vard_T, 50
   ldi Vard_L, 10
;
; Inizializzo altre variabili
   ldi val_prec, 1        ; inizio con l'interruttore aperto (ossia connesso
all'alimentazione)
   ldi CELSIUS, 1   ; con l'interruttore aperto voglio la conversione in celsius
;
; Programmazione dell'ADC in modo tale da abilitarlo senza abilitare gli interrupt
; deve essere sempre fatto prima della selezione dell'ingresso
;
   ldi mp, 0b1000_0000
   sts ADCSRA, mp
;
; Abilitazione degli interrupt a livello di SREG
   sei
;
;
main_loop:
;
;check_soglia
;
   cpi vard_V, 0
;
   brne fine_check_s
;
; Reinizializzo la variabile di conteggio
   ldi vard_V, 100
;
; Abilito l'ADC e seleziono un prescaler pari a 4
   ldi mp, 0b1000_0010
   sts ADCSRA, mp
;
```

```
; Seleziono gli ingressi dell'ADC (canale 1), il riferimento e giustifico il risultato a
sinistra
   ldi mp,0b0110_0001
   sts ADMUX,mp
;
; Inizio il campionamento. Per ottenere un valore corretto scarto le prime 70 letture
(necessario a causa del cambio di riferimento dell'ADC)
   ldi FLAG, 70
ripetizione_led:
   lds mp, ADCSRA
   ldi mp1, 0b0100_0000
   or mp, mp1
   sts ADCSRA,mp
;
; Attende che la conversione sia terminata
check_conv_led:
   lds mp, ADCSRA
   ldi mp1,0b0100_0000
   and mp, mp1
   brne check_conv_led
   dec FLAG
   cpi FLAG, 0
   brne ripetizione_led
;
; Se la conversione è terminata legge il risultato ottenuto in ADCH (8 bit più
significativi)
   lds XH, ADCH
;
; SOLUZIONE A SINGOLA SOGLIA
   cpi XH, V_SOGLIA_max
;
   brlo accendi_led
;
; se mp maggiore della soglia allora il led deve essere spento
   ldi mp, 0b1101_1111
   in mp1, PORTB
   and mp, mp1 ;voglio che il bit 5 sia a zero mentre voglio che gli altri rimangano
invariati
   out PORTB, mp
   ldi FLAG_LED,0
   jmp fine_check_s
;
;
accendi_led:
   cpi XH,V_SOGLIA_min
;
   brsh lampeggia
;
   ldi mp, 0b0010_0000
   in mp1, PORTB
   or mp, mp1 ; voglio che il bit 5 sia a 1 mentre gli altri devono rimanere invariati
   out PORTB, mp
   ldi FLAG_LED,0
   jmp fine_check_s
;
lampeggia:
   ldi FLAG_LED,1
;
fine_check_s:
;
; verifico quanto vale il flag: solo se FLAG_LED è 1 il sistema deve lampeggiare e
proseguirà con le istruzioni
   cpi FLAG_LED,1
   brne fine_lampeggia
```

```
;
; il led deve lampeggiare con la giusta temporizzazione, analizzo Vard_L che viene
decrementato nella subroutine di risposta ad interrupt
;
lampeggia_led:
   cpi vard_L,0
   brne fine_lampeggia
   ldi vard_L,10
   in mp, PORTB
   ldi mp1, 0b0010_0000
   eor mp, mp1
   out PORTB, mp
fine_lampeggia:
;
;
; Inizia qui la seconda richiesta
   cpi Vard_T,0
   brne fine_temperatura
;
; Rinizializzo la variabile di conteggio
   ldi Vard_T, 50
;
; Definisco se voglio il valore in Celsius o in Fahrenheit
   eor CELSIUS, TIPO_TABELLA
   cpi CELSIUS, 1
   brne fahrenheit
;
; se CELSIUS = 1
   ldi ZH, high(2*tab_celsius)
   ldi ZL, low(2*tab_celsius)
   jmp passa_indirizzo
;
fahrenheit:
   ldi ZH, high(2*tab_fahrenheit)
   ldi ZL, low(2*tab_fahrenheit)
;
passa_indirizzo:
;
;
       push   ZH                          ; mette ZH nello stack per passarlo alla subroutine
che lo utilizzerà come parametro
       push   ZL                          ; mette ZL nello stack per passarlo alla subroutine
che lo utilizzerà come parametro
;
; richiamo della subroutine measure che campiona la tensione e visualizza il risultato
(attenzione, la call memorizza nello stack i due byte dell'indirizzo di rientro)
;
       call measure
;
fine_temperatura:
;
       nop                                    ; istruzione di comodo aggiunta per il
debugging (nop = no operation)
;
; pulsante
;
; Leggo il valore del bit 0 di PINB
   in mp, PINB
   ldi mp1, 0b0000_0001
   and mp, mp1
;
; Inverto il bit 0 del valore letto
   ldi mp1, 0b0000_0001
   eor mp, mp1
```

```
;
; Funzione logica che viene giustificata nella relazione
   and val_prec, mp
   mov TIPO_TABELLA, val_prec ; se TIPO_TABELLA è pari a 1 allora significa che il nome
della tabella va cambiato. Viceversa se TIPO_TABELLA = 0 allora il nome della tabella non va
cambiato.
   mov val_prec, mp ; sovrascrivo il valore
;
fine_pulsante:
;
;
       rjmp main_loop
;
measure:
;
       pop          mp           ; estrae temporaneamente dallo stack l'ultimo byte
dell'indirizzo di rientro messo nello stack dalla call
       pop          mp1          ; estrare temporaneamente dallo stack il primo byte
dell'indirizzo di rientro messo nello stack dalla call
       pop          ZL           ; recupera il byte basso dell'indirizzo  della tabella che
è stato passato dal programma chiamante nello stack
       pop          ZH           ; recupera il byte alto dell'indirizzo della tabella che è
stato passato dal programma chiamante nello stack
       push         mp1                 ; ripristina nello stack il primo byte
dell'indirizzo di rientro messo nello stack dalla call
       push         mp                  ; ripristina nello stack l'ultimo byte
dell'indirizzo di rientro messo nello stack dalla call
;
; Abilito ADC, seleziono il rimerimento e il canale ADC0
;
   ldi mp, 0b1000_0011
   sts ADCSRA, mp
   ldi mp,0b1100_0000
   sts ADMUX,mp
;
   ldi FLAG, 70
ripetizione:
; Inizia la conversione. Scarto i primi 70 valori
;
       lds          mp,ADCSRA
       ldi          mp1,0b0100_0000
       or           mp,mp1
       sts          ADCSRA,mp
;
; Aspetta che la conversione sia pronta testando ADSC in ADCSRA: a conversione terminata
ADSC torna a 0
;
check_conv_temp1:
       lds          mp,ADCSRA
       ldi          mp1,0b01000000
       and          mp,mp1
       brne         check_conv_temp1
       dec FLAG
       cpi FLAG,0
       brne ripetizione
;
; Leggo il risultato distribuito su 10 bit: devo leggere prima ADCL e poi ADCH
;
   lds XL, ADCL
   lds XH, ADCH
;
; Inizializzo FLAG a 0
   ldi FLAG, 0b0000_0000
;
```

```
; Confronto il valore letto con 279: se è minore vado a verifica_flag
;
   ldi mp, high (T_MIN)
   ldi mp1, low (T_MIN)
;
   cp XL, mp1
   cpc XH, mp
;
   brlo verifica_flag
;
; Se invece sono maggiori o uguali devo fare un ulteriore verifica: lo confronto con 420 e
se maggiore o uguale salto a verifica flag
;
   ldi mp, high (T_MAX)
   ldi mp1, low (T_MAX)
;
   cp XL, mp1
   cpc XH, mp
;
   brsh verifica_flag
;
; Se supera entrambe le verifiche (ossia se è compreso nel range 279-419) allora impongo il
FLAG a 1
;
   ldi FLAG, 0b0000_0001
;
verifica_flag:
;
; Se il flag è uguale a 1 vado alla riga corretta della tabella e mostro a schermo il valore
;
   ldi mp,1
   cp FLAG,mp
;
   brne spegni_display
;
; Vado alla riga corretta della tabella
; Il valore letto sta nel range tra 279 e 419. Quindi sottrendo i valori a 10 bit trovo che
i due bit meno significativi saranno sempre nulli
; di conseguenza posso effettuare la sottrazione esclusivamente tra gli 8 bit meno
significativi
;
   ldi mp, low (T_MIN)
   mov mp1, xL
   sub mp1, mp
   ldi mp, 4
   mul mp1, mp
   add ZL,R0
   adc ZH,R1
;
; A questo punto mi trovo all'indirizzo giusto e devo visualizzare sul display
;
; Prepara visulaizzazione su display
;
      lpm          mp, Z+       ; legge primo valore della riga della tabella che contiene
il valore completo intero su 8 bit ed incrementa Z
;
      lpm          mp, Z+       ; legge il secondo valore della riga della tabella che
contiene la cifra delle centinaia  ed incrementa Z
      ori          mp,0b0111_0000      ; lascia a uno PD4 - PD7   e non modifica PD0 - PD3

      out          PORTD, mp           ; scrive in uscita la cifra delle centinaia
      andi         mp, 0b0110_1111     ; prepara PD4 a livello basso (LE centinaia) senza
modificare la cifra delle centinaia
      out          PORTD, mp           ; abilita LE centinaia
```

```
	nop				; istruzioni di attesa necessarie per garantire
l'efficacia dello strobe
	nop
	nop
	nop
	nop
	ori		mp,0b0111_0000	; mette a uno PD4 - PD7 (alza LE)
	out		PORTD, mp
;
;
	lpm		mp, Z+		; legge il terzo valore della riga della tabella
che contiene la cifra delle decine  ed incrementa Z
;
	ori		mp,0b0111_0000	; lascia a 1  PD4 - PD7		e non modifica
PD0 - PD3
	out		PORTD, mp		; scrive in uscita la cifra delle decine
	andi		mp, 0b0101_1111		; prepara PD5 a livello basso (LE decine)
senza modificare la cifra delle decine
	out		PORTD, mp		; abilita LE decine
	nop				; istruzioni di attesa necessarie per garantire
l'efficacia dello strobe
	nop
	nop
	nop
	nop
	ori		mp,0b0111_0000	; mette a uno PD4 - PD7 (alza LE)
	out		PORTD, mp
;
	lpm		mp, Z+		; legge il quarto valore della riga della tabella
che contiene la cifra delle unità  ed incrementa Z
;
	ori		mp,0b0111_0000	; lascia a 1  PD4 - PD7		e non modifica
PD0 - PD3
	out		PORTD, mp		; scrive in uscita la cifra delle unità
	andi		mp, 0b0011_1111	; prepara PD6 a livello basso (LE unità) senza
modificare la cifra delle unità
	out		PORTD, mp		; abilita LE unità
	nop				; istruzioni di attesa necessarie per garantire
l'efficacia dello strobe
	nop
	nop
	nop
	nop
	ori		mp,0b0111_0000	; mette a uno PD4 - PD7 (alza LE)
	out		PORTD, mp
;
;	Scrittura display terminata
;
; L'operazione di spegnimento del display deve essere fatta solo se FLAG è pari a zero
;
spegni_display:
;
   ldi mp,0
   cp FLAG,mp
   brne fine_measure
;
;
   ldi mp, 0b0111_1111
   out PORTD, mp
   andi mp, 0b0000_1111 ; Fuori dalla condizione di memoria: dal datasheet lo schermo deve
spegnersi. Lo faccio insieme per decine, unità e decimale
   out PORTD, mp
   nop				; istruzioni di attesa necessarie per garantire
l'efficacia dello strobe
```

```
   nop
   nop
   nop
   nop
   ori        mp,0b0111_0000      ; riporto in condizione di memoria i tre display
   out        PORTD, mp
;
;
fine_measure:
       ret                                                    ; ritorna al programma
chiamante
;
;
nop
;
;
;
;
;  INTERRUPT HANDLERS FOLLOW
;
;
EXT_INT0:
       reti                                        ; vector 2:          IRQ0 Handler
;
EXT_INT1:
       reti                                        ; vector 3:          IRQ1 Handler
;
PCINTR0:
       reti                                        ; vector 4:          PCINT0 Handler
;
PCINTR1:
       reti                                        ; vector 5:          PCINT1 Handler
;
PCINTR2:
       reti                                        ; vector 6:          PCINT2 Handler
;
WDT:
       reti                                        ; vector 7:          Watchdog timer
handler
;
TIM2_COMPA:
       reti                                        ; vector 8:          Timer2 compare A
handler
;
TIM2_COMPB:
       reti                                        ; vector 9:          Timer2 compare B
handler
;
TIM2_OVF:
       reti                                        ; vector 10:         Timer2 Overflow
Handler
;
TIM1_CAPT:
       reti                                        ; vector 11:         Timer1 Capture
Handler
;
TIM1_COMPA:
       reti                                        ; vector 12:         Timer1 CompareA
Handler
;
TIM1_COMPB:
       reti                                        ; vector 13:         Timer1 CompareB
Handler
;
```

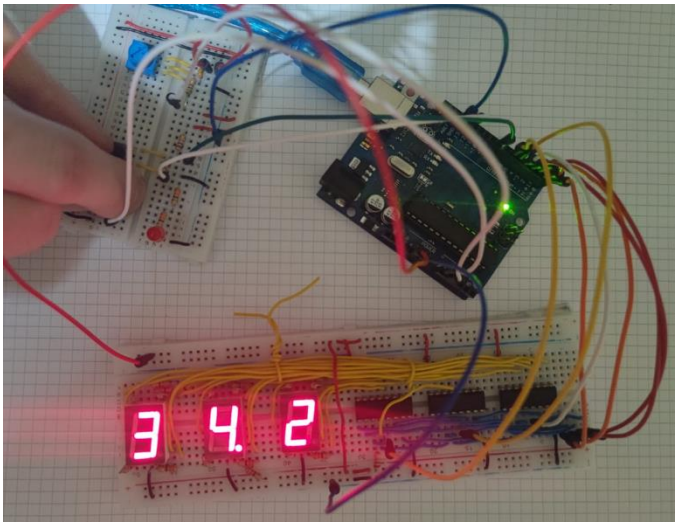
```
TIM1_OVF:
	reti                                            ; vector 14:        Timer1 Overflow
Handler
;
TIM0_COMPA:
	reti                                            ; vector 15:        Timer 0 CompareA
handler
;
TIM0_COMPB:
	reti                                            ; vector 16:        Timer 0 CompareB
handler
;
;
TIM0_OVF:
;
;	salva mp nello stack prima di utilizzarlo
	push   mp
;	salva SREG nello stack
	in     mp,SREG
	push   mp
;	inizializza nuovamente la variabile di conteggio del timer counter (10,24 ms)
	ldi    mp,246
	out    TCNT0,mp
;	decrementa le variabili di conteggio
	dec Vard_T
	dec Vard_V
	cpi Vard_L,1
	brlo avanti
	dec Vard_L
	avanti:
;	ripristina SREG
	pop    mp
	out    SREG,mp
;	ripristina mp
	pop    mp
;
	reti                                            ; vector 17:        Timer0 Overflow
Handler
;
;
;
SPI_STC:
	reti                                            ; vector 18:        SPI Transfer
Complete Handler
;
USART_RXC:
	reti                                            ; vector 19:        USART RX Complete
Handler
;
USART_UDRE:
	reti                                            ; vector 20:        USART UDR Empty
Handler
;
USART_TXC:
	reti                                            ; vector 21:        USART TX Complete
Handler
;
ADC_conv:
	reti                                            ; vector 22:        ADC Conversion
Complete Handler
;
EE_RDY:
	reti                                            ; vector 23:        EEPROM Ready
Handler
```

```
;
ANA_COMP:
       reti                                    ; vector 24:         Analog Comparator
Handler
;
TWSI:
       reti                                    ; vector 25:         Two-wire Serial
Interface Handler
;
SPM_RDY:
       reti                                    ; vector 26:         Store Program
Memory Ready Handler
```

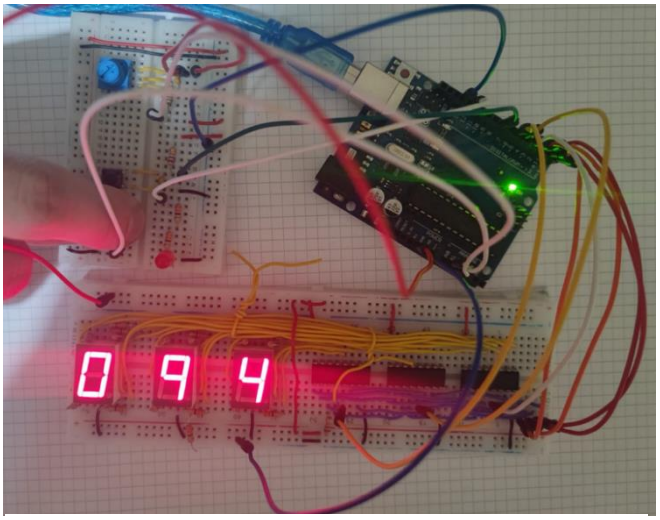
5. Verifica del funzionamento

5.1 Sensore LM35 a contatto con le dita della mano

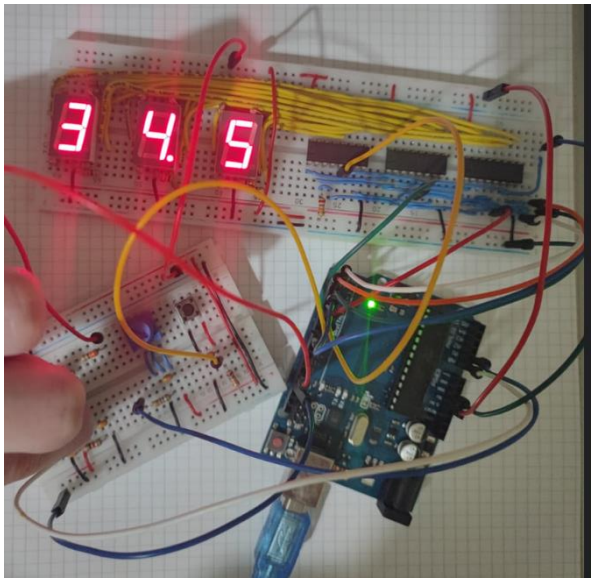
Stringendo con le dita della mano il sensore LM35 è possibile visualizzare la temperatura salire e stabilizzarsi. Facendo differenti prove, in un ambiente con temperatura prossima ai 22°C, sono state rilevate temperature tra i 32 e i 35,5 °C a seconda delle condizioni del soggetto (mani più o meno calde).
Di seguito le immagini del circuito funzionante nel momento in cui si applica una tensione di 5V mediante il PC. In tali condizioni il sensore è alimentato correttamente. Tale prova verifica anche il funzionamento del pulsante.



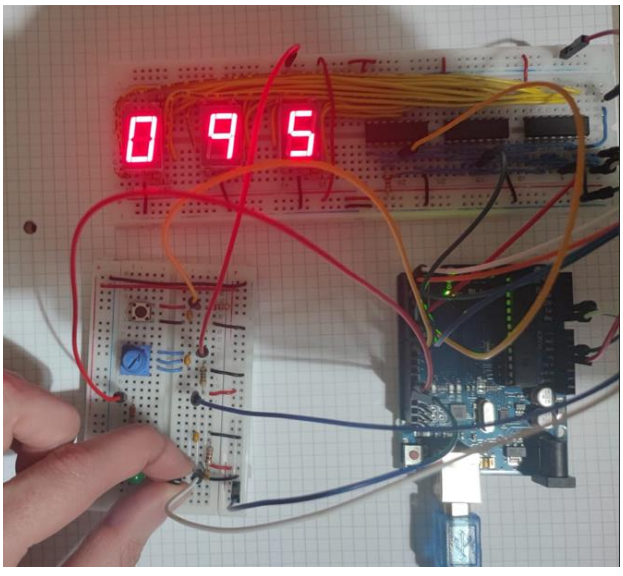
DISPOSITIVO DI BENNARDO WILLIAM: MISURA DELLA TEMPERATURA IN CELSIUS



DISPOSITIVO DI BENNARDO WILLIAM: MISURA DELLA TEMPERATURA IN FAHRENHEIT

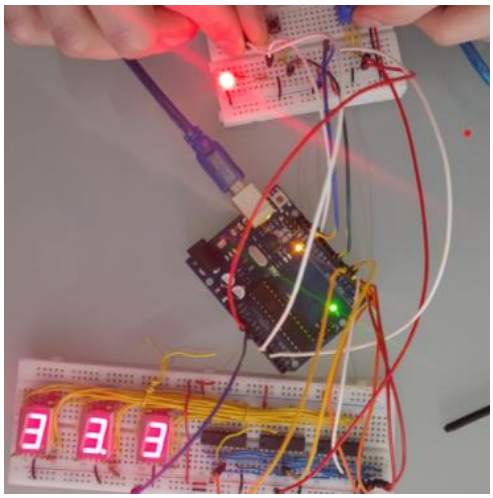


DISPOSITIVO DI BONATO REBECCA: MISURA DELLA TEMPERATURA IN CELSIUS

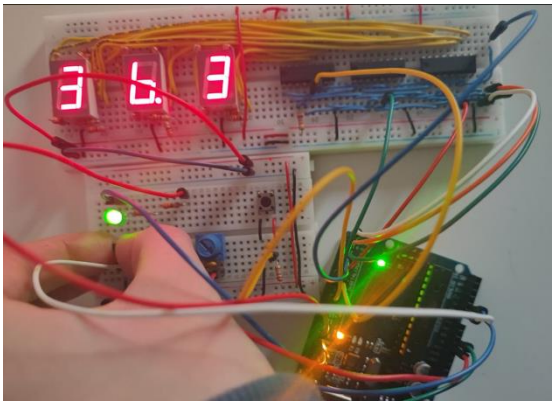


DISPOSITIVO DI BONATO REBECCA: MISURA DELLA TEMPERATURA IN FAHRENHEIT

Ruotando la manopola del trimmer, ossia diminuendo la tensione di alimentazione a circa 2,5V (valutazione qualitativa per l'impossibilità di misurare mediante un multimetro), il termometro non risulta più funzionante. I display si accendono e si spengono senza un ordine predefinito come se la misura di tensione restituisse risultati casuali. In questo caso il LED risulta acceso come nell'immagine successiva.



DISPOSITIVO DI BENNARDO WILLIAM: ERRORE NELLA MISURA DELLA TEMPERATURA E ACCENSIONE DEL LED

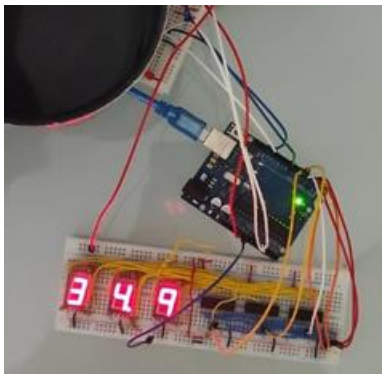


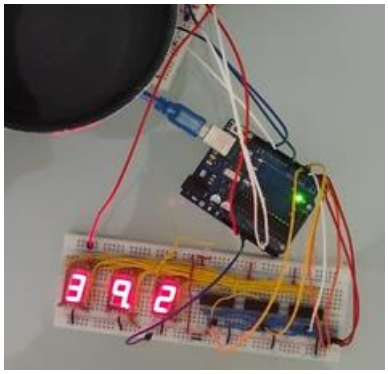
DISPOSITIVO DI BONATO REBECCA: ERRORE NELLA MISURA DELLA TEMPERATURA E ACCENSIONE DEL LED

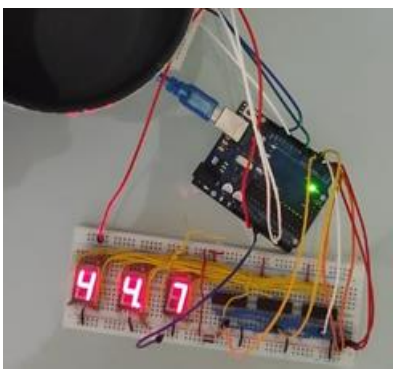
5.2 Sensore LM35 a contatto con un pentolino contenente acqua portata a ebollizione

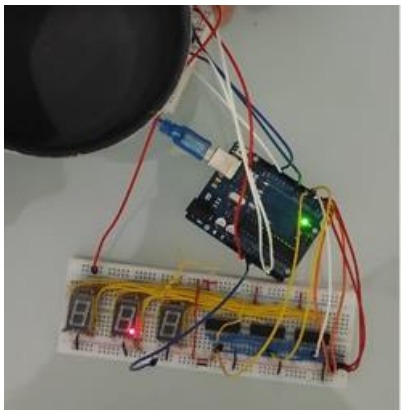
La seguente prova di verifica consiste nel far salire la temperatura fino a sopra soglia e successivamente farla diminuire mediante l'ausilio di un pentolino contenente acqua portata ad ebollizione. Tale pentolino viene posto a contatto con il sensore: in seguito sono riportati in sequenza delle immagini dei valori ottenuti per entrambi i dispositivi. le prime 4 foto fanno riferimento al sensore che viene scaldato; alla quarta foto la temperatura è sopra i 45°C quindi il display risulta spento; successivamente il pentolino viene tolto: i display si riaccendono e la temperatura scende.

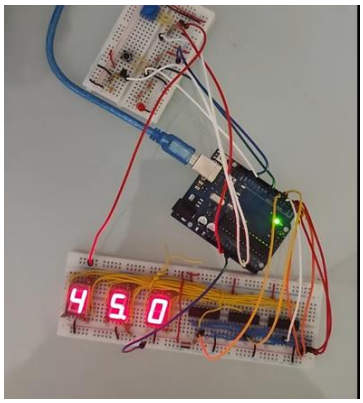
Bennardo William:

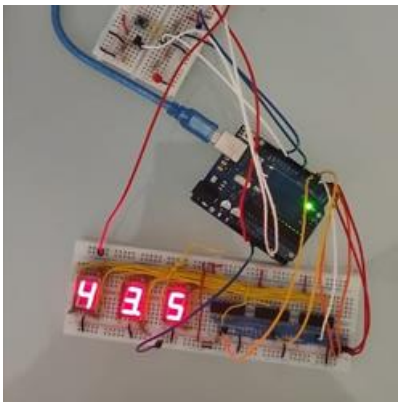




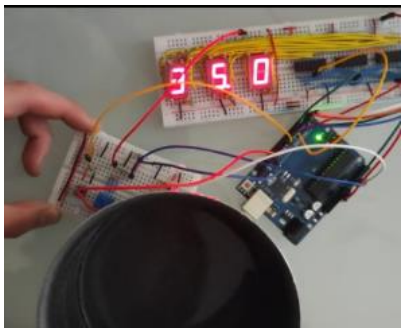


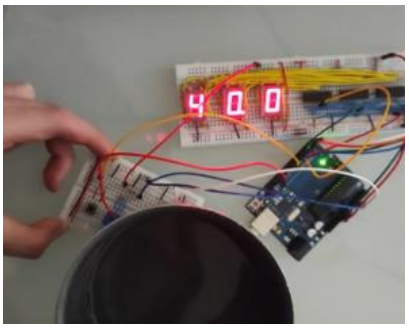


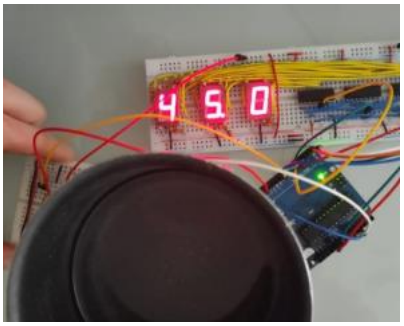


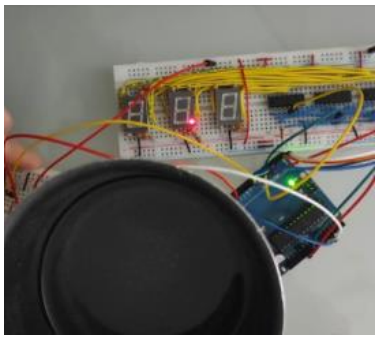


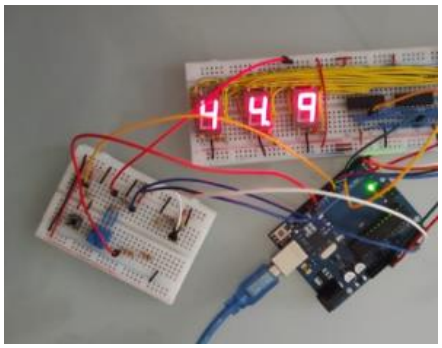
Bonato Rebecca:

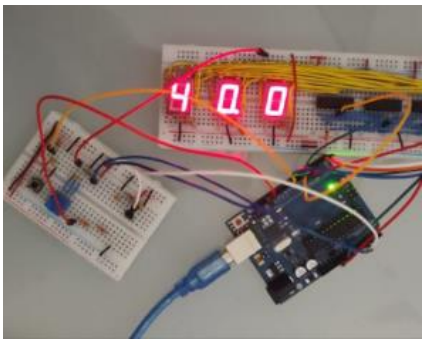












5.3 Verifica della soglia

Per quanto riguarda l'accensione e lo spegnimento del LED in seguito al campionamento della tensione di alimentazione, non disponendo di un multimetro non abbiamo potuto verificare che la tensione alla quale lo stato del LED cambia sia effettivamente quella corretta. Tuttavia, mediante un'analisi qualitativa possiamo osservare che:

- Il trimmer compie una rotazione di 270° coprendo una variazione della tensione tra 0 e 5V
- Partendo dal valore di tensione pari a 5V e facendo ruotare il trimmer, il LED inizia a lampeggiare dopo un angolo di circa 40°. 40° rappresenta circa il 15% di 270°. Diminuire una tensione di 5V del 15% significa avere una diminuzione di 0,75V. Ciò si dimostra

qualitativamente in linea con quanto richiesto: lampeggiare ad una diminuzione della tensione massima di 0,8V.

- Partendo dal valore di tensione pari a 5V e facendo ruotare il trimmer, il LED si accende in modo continuo dopo una rotazione poco più di 50°, circa il 19% di 270°C. Diminuire una tensione di 5V del 19% significa avere una diminuzione della tensione di alimentazione di 0,95V: anche in questo caso ci troviamo in linea con quanto dichiarato nelle specifiche, ossia l'accensione continua alla diminuzione della tensione di alimentazione di 1V.

6. Conclusioni

Per concludere possiamo affermare che il LED segnala adeguatamente il superamento delle soglie prefissate nelle specifiche, fungendo da segnale di allarme.

In merito alla misura di temperatura e alla sua relativa visualizzazione a schermo, viene rilevata una temperatura conforme a quello che ci aspettiamo.
La visualizzazione sul display viene aggiornata ogni 500ms come richiesto dalle specifiche.

Per quanto riguarda la gestione del pulsante, esso determina correttamente il passaggio da una visualizzazione in °C ad una visualizzazione in °F (o viceversa) se premuto. Talvolta, se il tasto viene premuto e rilasciato troppo velocemente o senza la sufficiente pressione, può capitare che la conversione non avvenga. Per ovviare al problema è sufficiente esercitare pressione per un tempo leggermente maggiore.