



**Università  
degli Studi  
di Palermo**

# **Progetto Sistemi Embedded**

**Automatic Firedoor Alarm  
A.F.A.**

***Prof. Daniele Peri***

***Corso di Laurea Magistrale Ingegneria Informatica (LM-32)***

***A cura di  
Luca La Barbera  
Salvatore Drago***

# Sommario

<b>Introduzione .....</b>	<b>2</b>
<b>Elenco componenti.....</b>	<b>2</b>
<b>Configurazione.....</b>	<b>2</b>
<b>Schema del sistema .....</b>	<b>4</b>
<b>Descrizione Componenti .....</b>	<b>4</b>
<b>Descrizione funzionamento generale .....</b>	<b>22</b>
<b>11_MAIN.forth .....</b>	<b>25</b>
<b>Vantaggi .....</b>	<b>26</b>
<b>Limiti e Miglioramenti.....</b>	<b>27</b>
<b>Conclusioni .....</b>	<b>27</b>

## Introduzione

L'obiettivo del progetto è quello di realizzare un sistema di allarme e monitoraggio pensato per le sale server. Il sistema monitora e mostra i valori di temperatura e umidità per ogni stanza, e permette la rilevazione di incendi, consentendo la chiusura immediata delle porte tagliafuoco, garantendo l'isolamento delle aree e preservando le altre sale macchine adiacenti dall'avanzare delle fiamme.

## Elenco componenti

Questo è l'elenco dei componenti elettronici utilizzati per la realizzazione del progetto:

- x 1 Raspberry Pi 3B+
- x 1 Servo motore sg90
- x 1 Sensore di fiamma A-Z Delivery KY-026
- x 1 BreadBoard
- x 1 FT232RL USB Interfaccia seriale UART
- x 1 Mini USB Cable
- x 1 Sensore di umidità e temperatura DHT-11
- x 1 Active Buzzer
- x 1 RGB LED
- x 1 LCD Display 16x02
- Resistenze 10kΩ 1kΩ 220kΩ

## Configurazione

Si procede con la descrizione della configurazione iniziale e del corretto avvio del dispositivo AFA.

Il progetto è realizzato in FORTH su ambiente PijForthOS.

PijForthOS è un interprete FORTH bare-metal per Raspberry Pi (modello B) che utilizza la console seriale RPi. Installando pijForthOS su una scheda SD nell'RPi, è possibile collegare la Raspberry a un'altra macchina utilizzando un cavo USB-seriale.

Quando l'RPi è acceso, un programma terminale sulla macchina host consente l'accesso alla console FORTH. In questo caso si utilizza il software ZOC8 Terminal.

Tramite terminale si procede ad importare le librerie a disposizione, in seguito descritte.

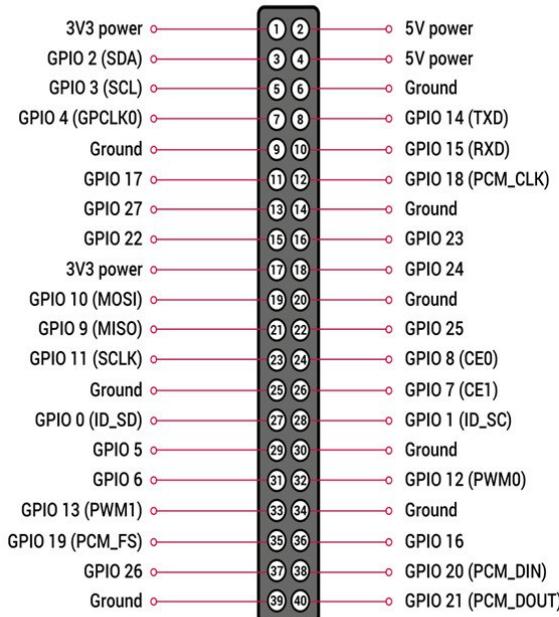
### *Import Librerie*

Descrizione delle librerie importate per l'attivazione e l'utilizzo del dispositivo AFA.

Ordine esecuzione	File.forth	Descrizione breve
1°	0-SE-ANS	File di configurazione iniziale del sistema
2°	1_AUTOCONFIG	File di configurazione che inizializza i registri riconoscendo il tipo di dispositivo Raspberry Pi tra 3b+ e 4
3°	2_UTILS	File di configurazione che istanzia alcune funzioni di utilità, che serviranno dopo per l'implementazione delle funzionalità del dispositivo AFA
4°	3_CONFSENS	File di configurazione dei sensori, abilitazione dei Pin GPIO e creazione delle variabili utili e degli array per la gestione dei sensori.
5°	4_RGB_LED	File che si occupa dell'implementazione dei comandi per RGB LED
6°	5_THERMO_SENSOR	File che si occupa dell'implementazione dei comandi per sensore di temperatura e umidità
7°	6_BUZZER	File che si occupa dell'implementazione dei comandi per Active Buzzer
8°	7_SERVO	File che si occupa dell'implementazione dei comandi per il Servo motore
9°	8_LCD	File che si occupa dell'implementazione dei comandi per accendere e pilotare LCD
10°	10_FLAMESENS	File che gestisce il sensore di fiamma
11°	11_MAIN	File principale per il funzionamento del sistema AFA.

## Schema del sistema

Si procede alla stesura dei dettagli del sistema dal punto di vista progettuale



GPIO	FUNCTION	COMPONENTS
2	ALT 0: SDA1	SDA I2C SLAVE – DISPLAY LCD1602
3	ALT 0: SCL1	SCL I2C SLAVE – DISPLAY LCD1602
5	INPUT	DATA - DHT11
6	INPUT	DATA – KY026
12	ALT 0: PWM0	SG90
13	OUTPUT	ACTIVE BUZZER
17	OUTPUT	RED RGB LED
18	OUTPUT	GREEN RGB LED
27	OUTPUT	BLUE RGB LED

## Descrizione

Il collegamento delle componenti è stato effettuato seguendo lo schema precedentemente descritto.

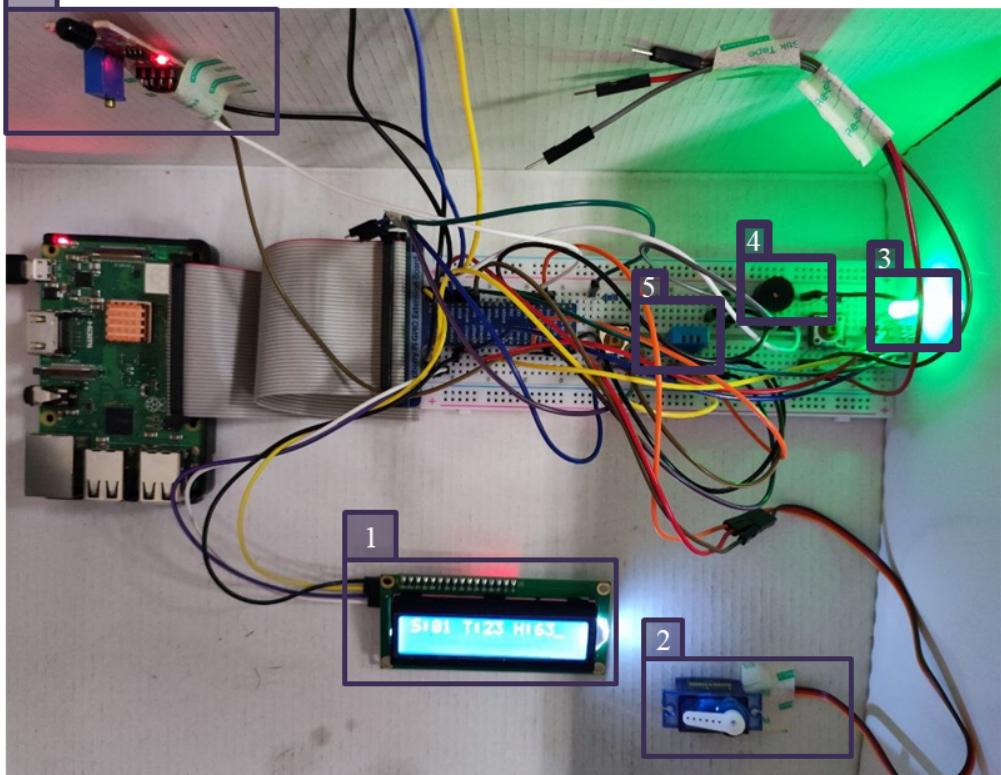
In particolare, le componenti interagiscono fra di loro nel seguente modo:

- Il Raspberry Pi coordina l'intero sistema e contiene la logica del software
- La Breadboard costituisce il ponte di collegamento per la restante componentistica
- L'alimentazione dell'intero sistema è fornita dal Raspberry Pi, che a sua volta è alimentato tramite il proprio alimentatore da una presa di corrente

## Descrizione Componenti

In questa sezione descriviamo le componenti utilizzate, il loro scopo all'interno del nostro progetto e il software da noi scritto per il loro funzionamento.

6



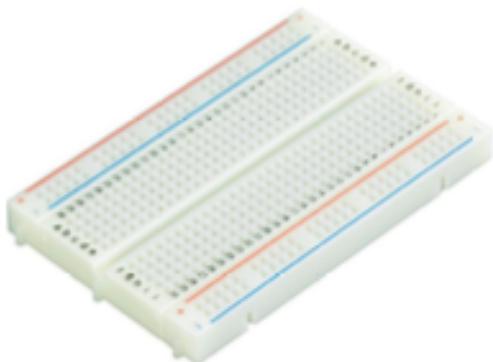
1. Display: LCD1602
2. Servomotore: SG90
3. Led RGB
4. Active Buzzer
5. Sensore Temperatura-Umidità: DHT11
6. Sensore di fiamma: KY026

## Raspberry Pi 3b+



Il Raspberry Pi è un computer a scheda singola (single-board computer) progettato dalla Raspberry Pi Foundation. Si basa sul system-on-chip (SoC) BCM2837B0, che include un processore ARMv8 quad-core a 64 bit da 1,4 GHz, RAM di circa 1 GB e una potente GPU VideoCore IV. Raspberry Pi con i suoi 40 pin GPIO funziona anche come controller programmabile in un'ampia varietà di applicazioni di robotica ed elettronica.

## Breadboard



La Breadboard rappresenta un mezzo per realizzare montaggi di circuiti elettronici senza saldature. È costituita da una basetta provvista di una serie di fori disposti secondo righe e colonne e distanziati del passo standard di 2,54 mm, tipico dei pin dei circuiti integrati. I fori di una colonna, generalmente 5, sono internamente collegati fra loro mediante una barretta metallica a molla, ma non con i fori delle colonne adiacenti. Lungo i due lati maggiori della basetta sono disposte due file di fori (dette binari di alimentazione) per i collegamenti di alimentazione e di terra.

## FT232RL USB Interfaccia Seriale UART



È utilizzato per collegare la Raspberry con interfaccia seriale (UART) a un'interfaccia USB del proprio PC, garantendo robustezza in trasmissioni ad alta velocità. È dotata di 6 pin:

- DTR: Data Terminal Ready, un'uscita utilizzata per il controllo di flusso
- RX: Serial Data Receive Pin
- TX: Serial data Transmit Pin
- VCC: Uscita in tensione positiva
- CTS: Clear To Send, un ingresso utilizzato per il controllo di flusso
- GND: Messa a terra o 0V

Per la maggior parte dei casi, basta connettere i pin RX, TX, GND ai rispettivi pin della macchina target.

## SG90



SG90 è un servomotore piccolo e leggero con un'elevata potenza di uscita. Il servo può ruotare di circa 180 gradi (90 in ciascuna direzione).

Il codice che gestisce questo componente è contenuto nel file 7\_SERVO.forth, qui viene implementato un metodo FIREDOOR che prende in input:

1. Il numero della Stanza
2. una costante (APRI/CHIUDI), dichiarate all'inizio del file

Questo metodo quando invocato consente di attivare il servomotore e ruota di 90° la piccola asta montatagli sopra. Questo simula l'attivazione di attuatori che in caso di emergenza servono a chiudere le porte tagliafuoco.

### Funzionamento SG90

I servomotori sono una particolare tipologia di motore, che grazie alle loro caratteristiche e semplicità d'uso, vengono spesso utilizzati con schede come Raspberry o Arduino.

Un servomotore ha la peculiarità di impostare il perno di trasmissione della rotazione su angoli esatti compresi in un certo intervallo, che spesso vanno da 0 a 180°, ma ve ne sono anche di modelli che coprono intervalli maggiori.

### Componenti di un servomotore

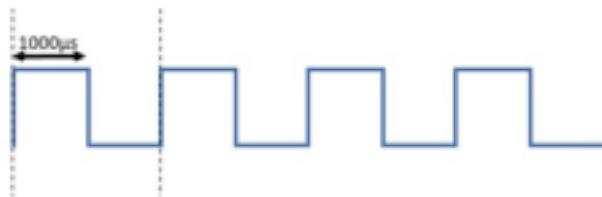
Per ottenere questo particolare movimento, si deve analizzare in dettaglio come è composto all'interno un servomotore.

La rotazione del perno in uscita si ottiene attraverso un motore DC (a corrente continua) collegato ad un meccanismo di demoltiplica che consente di aumentare la coppia in fase di rotazione, riducendo la velocità e aumentando la forza di torsione.

### Funzionamento servomotori

Un servomotore controlla la rotazione di un motore DC attraverso un circuito di controllo che ne regola l'angolo. Il controllo viene ottenuto regolando la lunghezza di un impulso di onda quadra inviato al servo motore. La lunghezza dell'impulso in un treno di segnali si definisce tramite il **PWM (Pulse Width Modulation)**.

Questo treno di impulsi si caratterizza dal **duty cycle**, cioè dal tempo occupato dall'impulso rispetto all'intero periodo destinato ad un singolo segnale. Per esempio, la classica onda quadra ha un duty cycle del 50%, dato che la durata dell'impulso (parte del segnale con stato 1) è pari alla metà del periodo dell'onda (cioè pari a quella parte del segnale con stato 0).



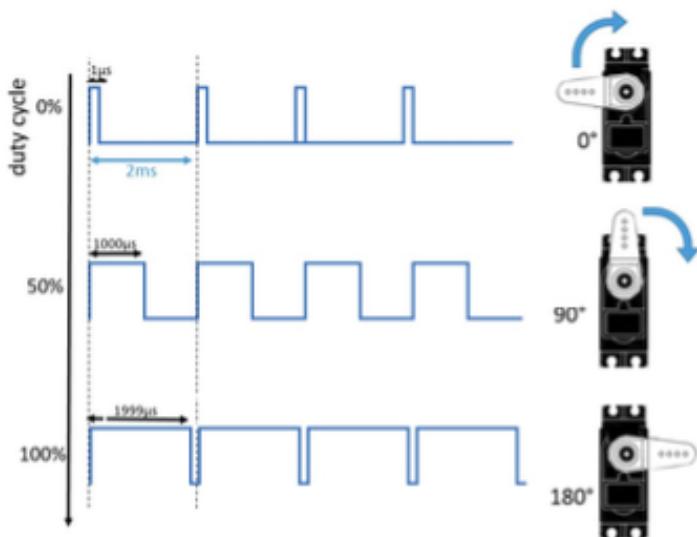
Ma si possono avere altri treni di impulsi con duty cycle diversi. Se infatti riduciamo la durata dello stato 1 del segnale ridurremo il duty cycle.



Oppure si può fare l'opposto, incrementando il periodo, per ottenere duty cycle maggiori.



Modulando quindi queste durante l'invio di un treno di impulsi si possono inviare delle informazioni al sistema di controllo del servomotore. A seconda del duty cycle, varierà l'angolo di rotazione. Partendo da un duty cycle del 1%, corrispondente all'angolo di 0°, via via incrementando il duty cycle, crescerà anche l'angolo di rotazione del servomotore. Con un duty cycle del 50%, il perno del motore si posizionerà su 90° (che è proprio la metà dell'intervallo di rotazione possibile). Con un duty cycle del 99%, avremo una rotazione di 180°.



Quindi, attraverso questo sistema di modulazione PWM, saremo in grado di inviare dei comandi dall'esterno al servo motore per fargli assumere l'angolo di rotazione desiderato.

## Funzionamento PWM

Raspberry Pi presenta un PWM controller che incorpora due bit-stream indipendenti, con clock a frequenza fissa.

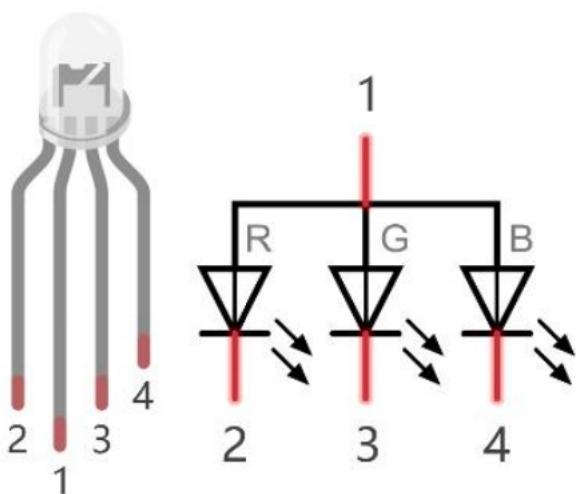
Il servomotore (SG90) ha una frequenza di funzionamento di 100khz. Poiché il clock del PWM è a 19,2 Mhz, si procede a dividerlo per 192 usando il CM\_PWMMDIV; grazie al registro CM\_PWMCTL abilitiamo il clock generator.

Per l'abilitazione del servo bisogna specificare il range (nel nostro caso 2000 che indica uno sfasamento da 0° a 180°) nel registro RNG1 e settare, nel registro CTL del PWM, il PWEN1 e il MSEEN1 a 1.

Il movimento del servo avviene modificando il valore all'interno del registro DAT1 del PWM.

Valore	Identificatore	Descrizione
0	CTL	registro PWM control
4	STA	registro PWMSTA
10	RNG1	registro PWM Channel 1 Range
14	DAT1	registro PWM Channel 1 Data
20	RNG2	registro PWM Channel 2 Range
24	DAT2	registro PWM Channel 2 Data

### RGB LED



Un LED RGB ha 3 LED integrati in un unico componente. Può emettere rispettivamente luce rossa, verde e blu. Per fare ciò, ha bisogno di 4 pin (è anche il modo in cui viene identificato). Il pin più lungo (1) è il comune, ovvero l'anodo (+) o cavo positivo, mentre gli altri 3 sono i catodi (-) o cavi negativi. Di seguito è riportata una rappresentazione di un LED RGB e del suo simbolo elettronico. Possiamo far sì che il LED RGB emetta vari colori di luce e luminosità controllando i 3 catodi (2, 3 e 4) del LED RGB.

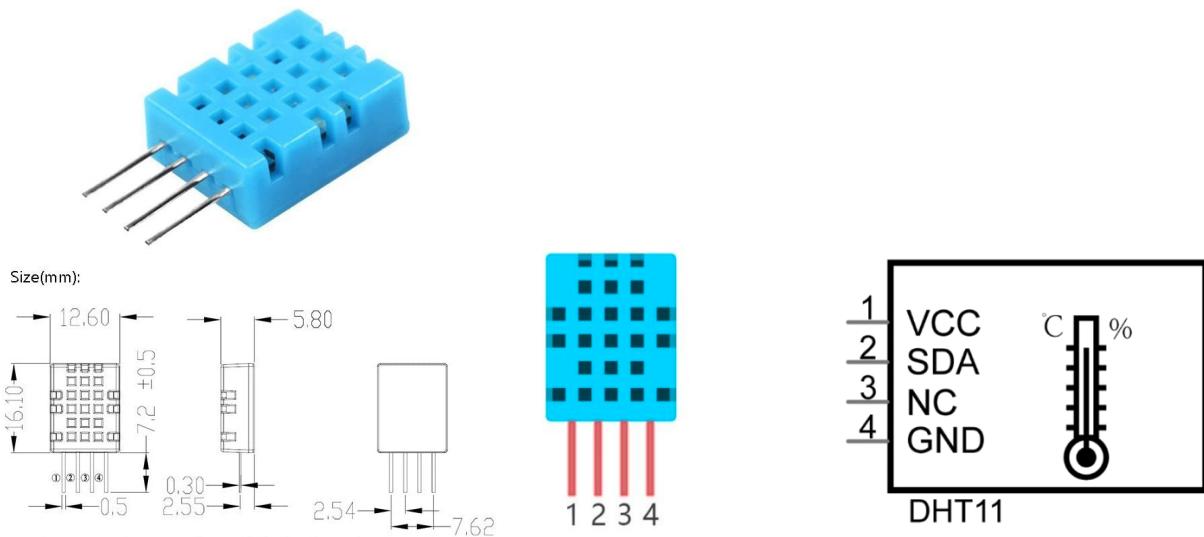
Nel nostro sistema questo componente viene utilizzato per evidenziare lo stato del sistema.

Abbiamo tre casi:

1. LED verde: il sistema rileva temperatura ed umidità rientranti nei valori soglia stabiliti, nessun sistema di allarme attivo.
2. LED giallo: il sistema rileva temperatura e/o umidità al di fuori dei valori soglia stabiliti, allerta pericolo dovuto ad innalzamento temperatura o umidità in sala server
3. LED rosso: il sistema rileva la presenza di fiamma e incendio e dunque, attiva allarme sonoro e chiude le porte tagliafuoco attivando il servomotore.

Il codice che gestisce lo switching delle colorazioni del LED RGB è scritto nel file 4\_RGB\_LED.forth. Nel codice è presente un solo metodo: SETCOLOR, che prende in input la stanza e il colore da attivare e, illumina il LED della stanza indicata del colore indicato.

## Thermo Sensor DHT11



Il sensore di temperatura e umidità DHT11 è un sensore composto di temperatura e umidità e il segnale digitale in uscita è stato calibrato dal produttore.

Dopo l'accensione, si inizializza in 1 secondo. La sua tensione di funzionamento è compresa nell'intervallo 3,3V-5,5V. Il pin SDA è un pin dati, utilizzato per comunicare con altri dispositivi. I pin NC (Not Connected Pin) sono un tipo di pin che si trovano in vari pacchetti di circuiti integrati. Questi pin non hanno alcuno scopo funzionale per il circuito esterno (ma possono avere una funzionalità sconosciuta durante la produzione e il test). Questi pin non devono essere collegati a nessuna delle connessioni del circuito.

Dettagli tecnici:

Parameters	Conditions	Minimum	Typical	Maximum
<b>Humidity</b>				
<b>Resolution</b>		1%RH	1%RH	1%RH
			8 Bit	
<b>Repeatability</b>				
<b>Accuracy</b>	25°C		± 1%RH	
		0-50°C	± 4%RH	± 5%RH
<b>Interchangeability</b>				
<b>Measurement Range</b>	0°C	30%RH		90%RH
	25°C	20%RH		90%RH
	50°C	20%RH		80%RH
<b>Response Time (Seconds)</b>	1/e(63%)25 °C, 1m/s Air	6 s	10 s	15 s
<b>Hysteresis</b>			± 1%RH	
<b>Long-Term Stability</b>	Typical		± 1%RH/year	
<b>Temperature</b>				
<b>Resolution</b>		1°C	1°C	1°C
		8 Bit	8 Bit	8 Bit
<b>Repeatability</b>			± 1°C	
<b>Accuracy</b>		± 1°C		± 2°C
<b>Measurement Range</b>		0°C		50°C
<b>Response Time (Seconds)</b>	1/e(63%)	6 s		30 s

Fasi principali per la lettura dei valori:

il sensore DHT11 trasmette in uscita un segnale di tipo analogico, è stato dunque necessario eseguire dei passi ben precisi per l'inizializzazione del sensore e la lettura dei valori:

### 1. Inizializzazione:

Quando il sensore viene alimentato, non bisogna inviare alcuna istruzione al sensore entro un secondo per evitare che lo stato di instabilità venga superato. È possibile aggiungere un condensatore da 100nF tra VDD e GND per filtrare l'alimentazione.

### 2. Tipo di comunicazione:

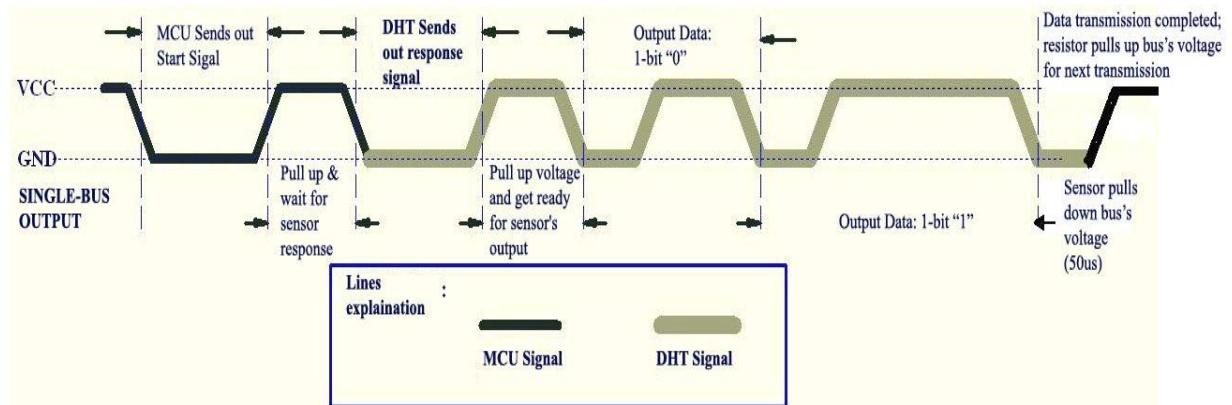
Per la comunicazione e la sincronizzazione tra MCU e sensore DHT11 viene utilizzato il formato dati a bus singolo. Un processo di comunicazione dura circa 4 ms.

I dati sono costituiti da parti decimali e integrali. La trasmissione completa dei dati è di 40 bit e il sensore invia per primo il bit di dati più alto.

Formato dei dati: Dati RH integrali a 8bit + dati RH decimali a 8bit + dati T integrali a 8bit + dati T decimali a 8bit + somma di controllo a 8bit. Se la trasmissione dei dati è corretta, la checksum dovrebbe essere l'ultimo 8bit di "8bit dati RH integrali + 8bit dati RH decimali + 8bit dati T integrali + 8bit dati T decimali".

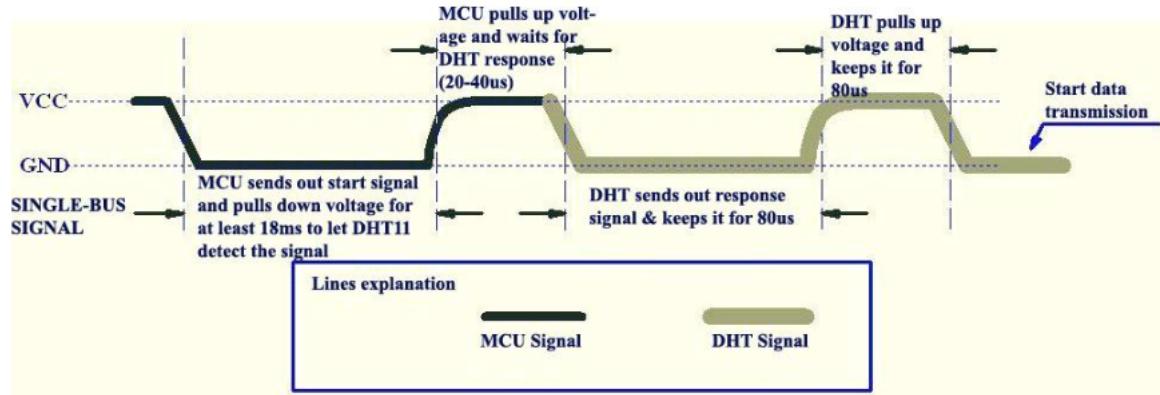
### 3. Processo di comunicazione:

Quando l'MCU invia un segnale di avvio, il DHT11 passa dalla modalità a basso consumo energetico alla modalità di funzionamento, in attesa che l'MCU completi il segnale di avvio. Una volta completato, il DHT11 invia all'MCU un segnale di risposta con dati a 40 bit che includono informazioni sull'umidità relativa e sulla temperatura. Gli utenti possono scegliere di raccogliere (leggere) alcuni dati. Senza il segnale di avvio da parte dell'MCU, il DHT11 non invierà il segnale di risposta all'MCU. Una volta raccolti i dati, il DHT11 passa alla modalità a basso consumo energetico fino a quando non riceve nuovamente un segnale di avvio dall'MCU.



#### 3.1 L'MCU invia il segnale di avvio al DHT

Lo stato libero del Data Single-bus è a livello di tensione alto. Quando inizia la comunicazione tra l'MCU e il DHT11, il programma dell'MCU imposta il livello di tensione del Data Single-bus da alto a basso; questo processo deve durare almeno 18 ms per garantire il rilevamento del segnale dell'MCU da parte del DHT; quindi, l'MCU tira la tensione verso l'alto e attende 20-40 ms per la risposta del DHT.

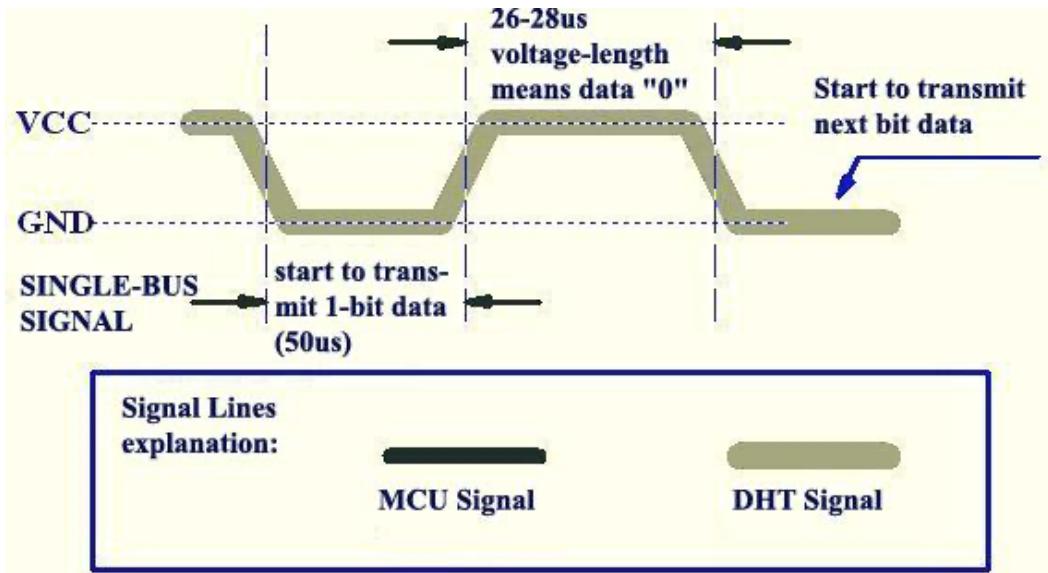


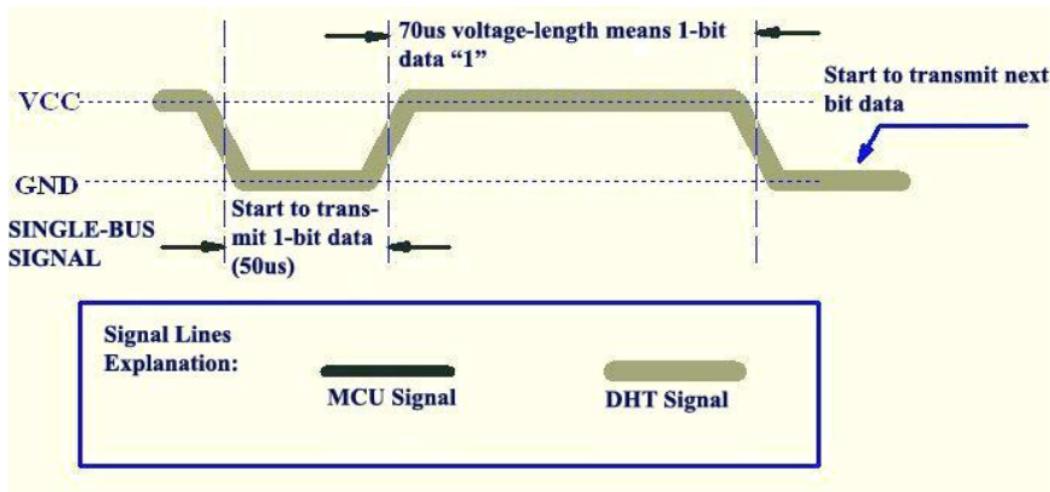
### 3.2 DHT risponde a MCU

Una volta rilevato il segnale di avvio, il DHT invia un segnale di risposta a basso livello di tensione, della durata di 80us. Quindi il programma del DHT imposta il livello di tensione di Data Single-Bus da basso ad alto e lo mantiene per 80us per la preparazione del DHT all'invio dei dati.

Quando DATA Single-Bus si trova al livello di tensione basso, significa che il DHT sta inviando il segnale di risposta. Una volta che il DHT ha inviato il segnale di risposta, tira su la tensione e la mantiene per 80us, preparandosi alla trasmissione dei dati.

Quando il DHT invia i dati all'MCU, ogni bit di dati inizia con il livello di bassa tensione di 50us e la lunghezza del successivo segnale di alta tensione determina se il bit di dati è "0" o "1".





Se il segnale di risposta del DHT è sempre ad alto livello di tensione, significa che il DHT non risponde correttamente e che è necessario controllare il collegamento. Quando viene trasmesso l'ultimo bit, il DHT11 abbassa il livello di tensione e lo mantiene per 50 secondi. Quindi la tensione del Single-Bus viene tirata su dal resistore per riportarla allo stato libero.

Tutto il codice riguardante il funzionamento del sensore DHT11 è stato scritto all'interno del file 5\_THERMO\_SENSOR.forth, all'interno del quale si trovano tutte le funzioni che realizzano le fasi per la lettura dei valori di temperatura e umidità sopradescritti.

Nello specifico, abbiamo scelto di effettuare le letture dei valori, scrivendo una Word che ci permettesse di valutare il segnale e decidere in base all'intervallo di tempo in cui questo rimane alto, se il bit in trasmissione fosse uno "0" o un "1".

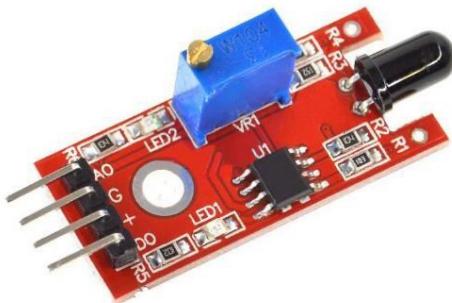
Questa funzione viene svolta da ZEROONEREAD. Una volta determinato il valore del bit trasmesso, questo viene memorizzato in un array di 40 celle, ognuna delle quali contiene in ordine i bit trasmessi durante una misurazione.

In una seconda fase viene calcolata la checksum tramite la funzione CHECKSUM che itera sull'array di 40 bit, questa ritorna un valore booleano "-1" o "0".

I valori letti infine vengono memorizzati in 4 variabili:

1. HHUMIDITY contenente il valore intero dell'umidità [bit da 0 a 7]
2. LHUMIDITY contenente il valore decimale dell'umidità [bit da 8 a 15]
3. HTEMPERATURE contenente il valore intero della temperatura [bit da 16 a 24]
4. LTEMPERATURE contenente il valore decimale della temperatura [bit da 24 a 31]

## Flame Sensor A-Z delivery KY-026



Il fotodiodo collegato è sensibile alla gamma spettrale di luce generata dalle fiamme libere.

Uscita digitale: Dopo aver rilevato una fiamma, viene emesso un segnale digitale 0 o 1.

Uscita analogica: Il sensore invia un segnale analogico continuo

1. LED1: Indica che il sensore è alimentato dalla tensione
2. LED2: Indica che il sensore rileva una fiamma

Il sensore ha 3 componenti principali sulla sua scheda elettronica. In primo luogo, l'unità sensore nella parte anteriore del modulo, che misura fisicamente l'area e invia un segnale analogico alla seconda unità, l'amplificatore.

Questo amplifica il segnale, in base al valore resistente del potenziometro, e lo invia all'uscita analogica del modulo.

Il terzo componente è un comparatore che commuta l'uscita digitale e il LED se il segnale scende sotto un determinato valore.

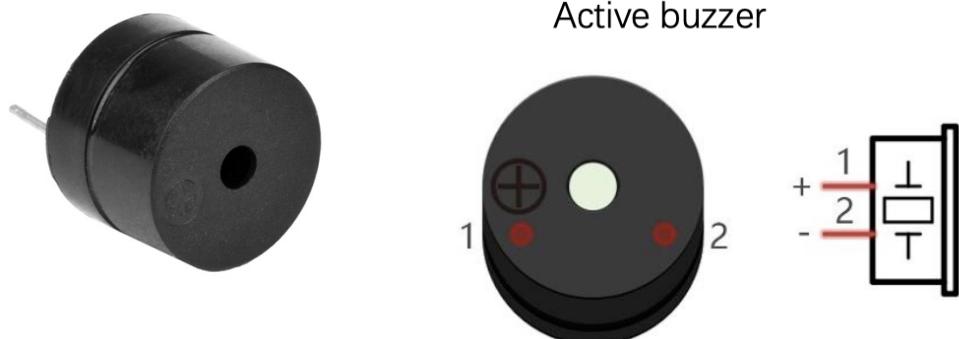
È possibile controllare la sensibilità regolando il potenziometro.

Attenzione: Il segnale digitale sarà invertito; ciò significa che se si misura un valore alto, questo viene visualizzato come un valore basso di tensione all'uscita analogica. Dunque, la lettura di un 1 nel GPIO a cui è collegato il sensore, indica la presenza di fiamma.

Nel nostro progetto, il sensore ci è utile per la rilevazione di fiamme e incendi.

Il codice che gestisce il sensore è scritto nel file 10\_FLAMESENSOR.forth, che possiede una funzione ISFIREEE che ritorna la lettura del valore del GPIO a cui è collegato l'output digitale del sensore.

## Active Buzzer



Il buzzer è un componente audio. È ampiamente utilizzato in dispositivi elettronici come calcolatrici, sveglie elettroniche, indicatori di guasto per automobili, ecc. Esistono buzzer di tipo attivo e passivo. I buzzer attivi sono dotati di un oscillatore interno e suonano finché viene fornita l'alimentazione. I cicalini passivi richiedono un segnale oscillatorio esterno (in genere utilizzando PWM con frequenze diverse) per emettere un suono.

I buzzer attivi sono più facili da usare. In genere emettono solo una frequenza sonora specifica. I cicalini passivi richiedono un circuito esterno per produrre suoni, ma possono essere controllati per produrre suoni di varie frequenze. La frequenza di risonanza del cicalino passivo in questo Kit è di 2kHz, il che significa che il cicalino passivo è più forte quando la sua frequenza di risonanza è di 2kHz.

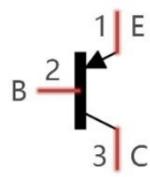
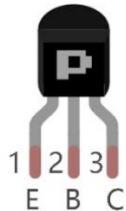
All'interno del nostro sistema il codice per il controllo del buzzer è scritto all'interno del file 6\_BUZZER.forth. Qui viene definita la funzione RING, questa prende in input la stanza e la costante ON/OFF. Il buzzer verrà attivato solo in caso di incendio. L'attivazione consiste nel settare a 1 l'uscita del GPIO a cui esso è collegato.

In questo progetto è necessario un transistor perché la corrente del buzzer è così elevata che la capacità di uscita del GPIO dell'RPi non è in grado di soddisfare il requisito di potenza necessario per il funzionamento. Un transistor NPN è necessario per amplificare la corrente.

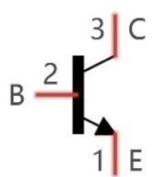
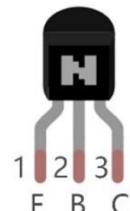
Il transistor è un dispositivo a semiconduttore che controlla la corrente (si pensi al transistor come a un "dispositivo di amplificazione o commutazione" elettronico). I transistor possono essere utilizzati per amplificare segnali deboli o per funzionare come interruttori. I transistor hanno tre elettrodi (PIN): base (b), collettore (c) ed emettitore (e). Quando c'è corrente che passa tra "be" e "ce" la corrente aumenta di diverse volte (ingrandimento del transistor); in questa configurazione il transistor agisce come un amplificatore. Quando la corrente prodotta da "be" supera un certo valore, "ce" limita la corrente in uscita. A questo punto il transistor lavora nella sua regione di saturazione e agisce come un interruttore. I transistor sono disponibili in due tipi, come illustrato

di seguito: PNP e NPN

PNP transistor



NPN transistor

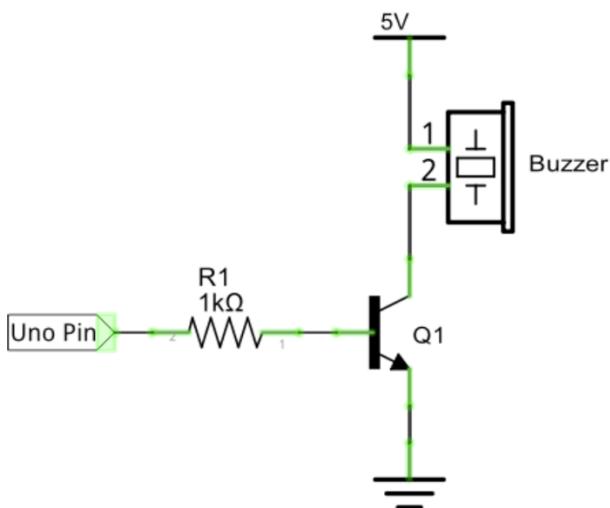


Grazie alle loro caratteristiche, i transistor sono spesso utilizzati come interruttori nei circuiti digitali. Poiché la capacità di corrente di uscita dei microcontrollori è molto debole, utilizzeremo un transistor per amplificare la sua corrente al fine di pilotare i componenti che richiedono una corrente maggiore.

Quando si utilizza un transistor NPN per pilotare un cicalino, spesso si utilizza il metodo seguente. Se il GPIO emette un livello alto, la corrente passa attraverso R1 (resistenza 1), il transistor conduce la corrente e il cicalino emette un suono. Se il GPIO emette un livello basso, non scorre corrente attraverso R1, il transistor non conduce corrente e il cicalino rimane silenzioso (nessun suono).

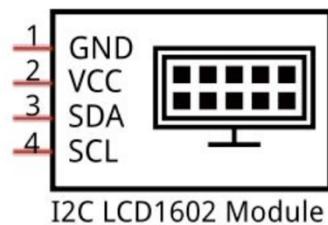
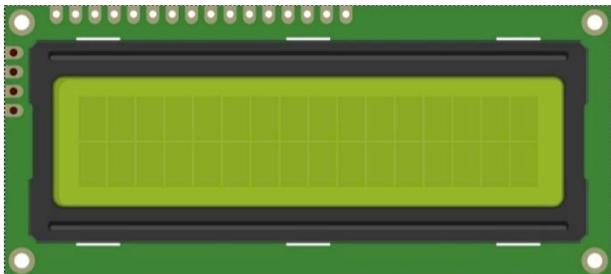
Nel nostro progetto utilizziamo un transistor NPN con la seguente configurazione:

NPN transistor to drive buzzer



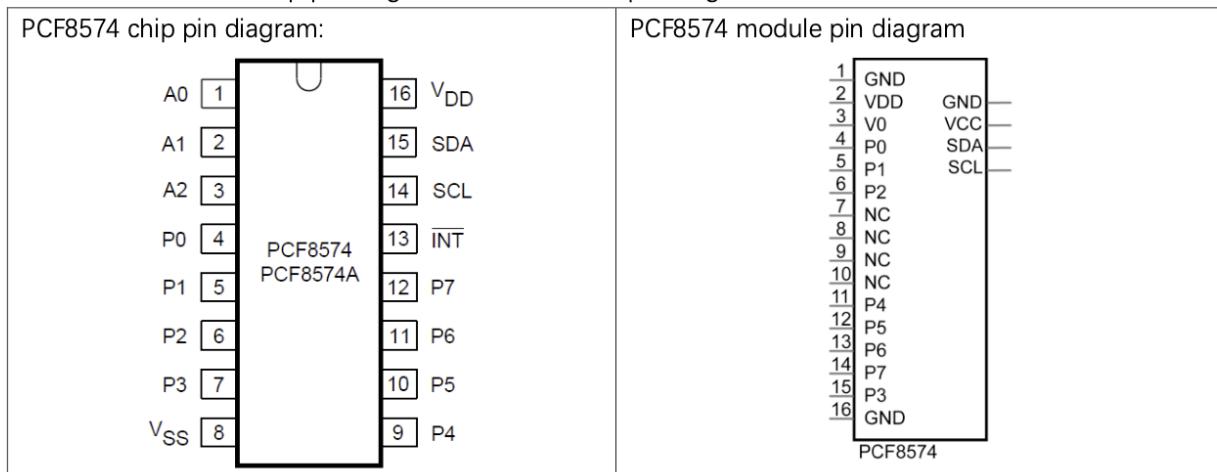
## I2C LCD 1602

Lo schermo LCD1602 può visualizzare 2 righe di caratteri in 16 colonne. È in grado di visualizzare numeri, lettere, simboli, codici ASCII e così via. Lo schermo LCD1602 I2C integra un'interfaccia I2C che collega il modulo di ingresso seriale e uscita parallela allo schermo LCD1602. Ciò consente di utilizzare solo 4 linee per il funzionamento del display LCD1602.

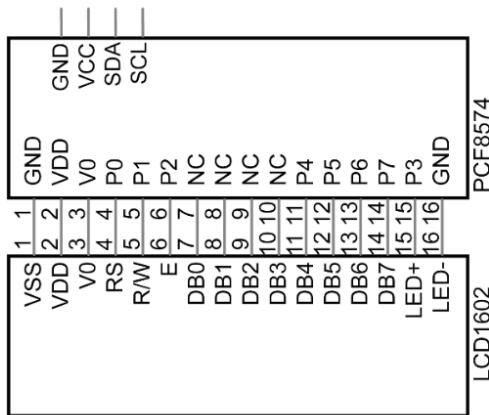


Il chip IC seriale-parallelo utilizzato in questo modulo è PCF8574T (PCF8574AT) e il suo indirizzo I2C predefinito è 0x27(0x3F).

Below is the PCF8574 chip pin diagram and its module pin diagram:



PCF8574 module pins and LCD1602 pins correspond to each other and connected to each other:



Per questo motivo, come detto in precedenza, abbiamo bisogno solo di 4 pin per controllare i 16 pin dello schermo LCD1602 attraverso l'interfaccia I2C.

In questo progetto, utilizzeremo lo schermo LCD1602 I2C per visualizzare alcuni caratteri statici e variabili dinamiche.

## I<sup>2</sup>C BUS

Bus informatico seriale sincrono, multi-master, multi-slave, a commutazione di pacchetto, single-ended, inventato da Philips Semiconductor (ora NXP Semiconductors).

Utilizzato per collegare circuiti integrati periferici a bassa velocità a processori e microcontrollori in comunicazioni intra-scheda a breve distanza.

Ogni dispositivo collegato al bus è indirizzabile via software con un indirizzo univoco.

Dotato di arbitrato e rilevamento delle collisioni

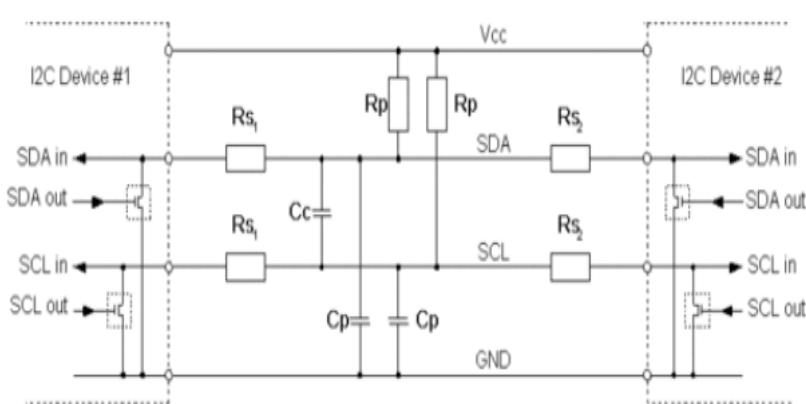
Nessun requisito rigoroso di velocità di trasmissione, il clock del bus è generato dal master  
La velocità di comunicazione originale è stata definita con un massimo di 100 kbit al secondo  
**(modalità Standard).**

**Modalità successive:**

- Modalità veloce (400 kbit/s)
- Modalità ad alta velocità (3,4 Mbit/s) con logica aggiuntiva
- Modalità veloce plus (fino a 1 MHz di frequenza massima, 1 Mbit/s) per velocità di trasferimento intermedie senza logica aggiuntiva.
- Modalità ultraveloce UFM (fino a 5 Mbit/s) ottenuta con modifiche importanti (master singolo, nessun riconoscimento, nessun arbitraggio multimaster, connessioni push-pull).

Due fili trasportano i segnali di dati (SDA) e di clock (SCL). Le connessioni open drain consentono di:

- il funzionamento simultaneo di più di un master I2C (se sono in grado di gestire più master)
- l'allungamento del clock (gli slave possono rallentare la comunicazione tenendo premuto SCL)

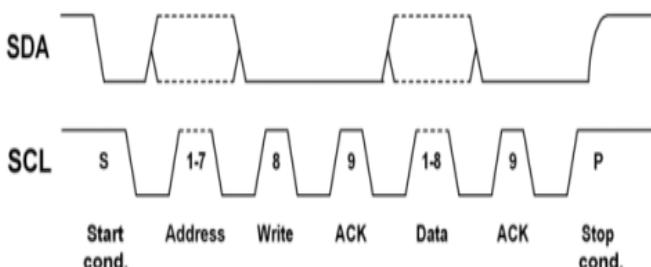


VCC	I2C supply voltage (1.2 V - 5.5 V)
GND	Common ground
SDA	Serial data (I2C data line)
SCL	Serial clock (I2C clock line)
Rp	Pull-up resistance (I2C termination)
Rs	Serial resistance
Cp	Wire capacitance
Cc	Cross channel capacitance

### Indirizzamento di più slave sullo stesso bus

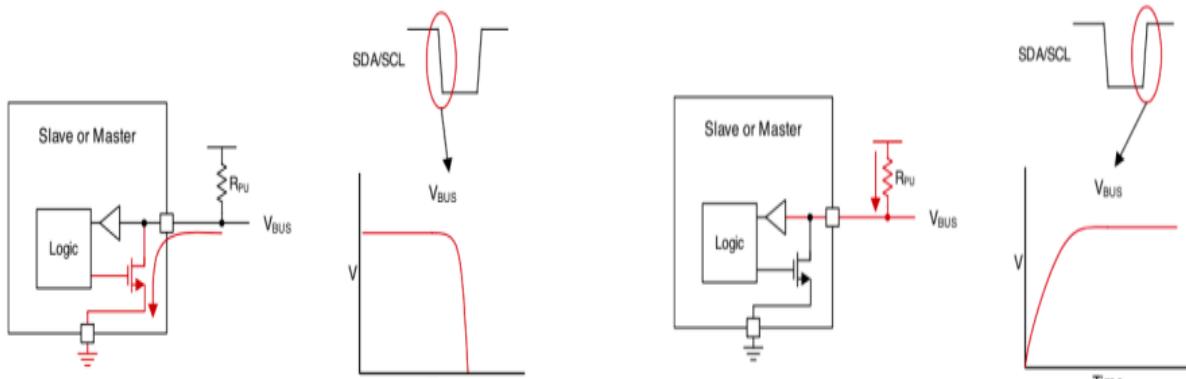
Le connessioni a scarico aperto consentono:

- il funzionamento simultaneo di più master I2C (se sono multi-master). multi-master)
  - l'allungamento del clock (gli slave possono rallentare la comunicazione tenendo premuto SCL)
- Tutti, utilizzando una singola linea dati bidirezionale.



Address	Purpose
0000000 0	General Call
0000000 1	Start Byte
0000001 X	CBUS addresses
0000010 X	Reserved for Different Bus Formats
0000011 X	Reserved for future purposes
00001XX X	High-Speed Master Code
11110XXX	10-bit Slave Addressing
11111XXX	Reserved for future purposes

Passa-parola aperta per la comunicazione bidirezionale. Per trasmettere un valore basso la logica attiva il FET di pull-down. La linea è cortocircuitata a terra (tirata al minimo). Per trasmettere un alto la logica può rilasciare il bus solo disattivando il FET di pull-down. La linea viene lasciata fluttuante e il resistore di pull-up tira la tensione fino alla barra di tensione.



### Interfaccia bidirezionale standard che utilizza un controllore (master) per comunicare con i dispositivi slave

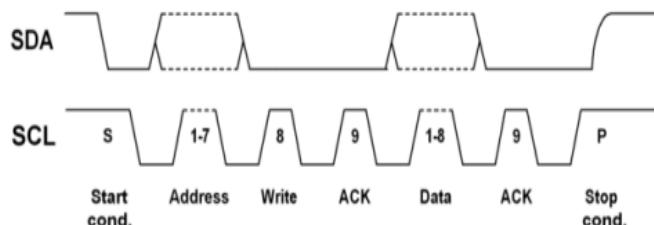
Uno slave non può trasmettere dati se non è stato indirizzato dal master.

Ogni dispositivo sul bus I2C ha un indirizzo specifico per distinguere gli altri dispositivi che si trovano sullo stesso bus I2C.

Molti dispositivi slave richiedono una configurazione all'avvio per impostare il comportamento del dispositivo.

Questo avviene in genere quando il master accede alle mappe dei registri interni dello slave, che hanno indirizzi di registro univoci.

Un dispositivo può avere uno o più registri in cui i dati vengono memorizzati, scritti o letti.



TI - SIVA704- linea 2015 n. 3 (siva704.pdf)

Address	Purpose
0000000 0	General Call
0000000 1	Start Byte
0000001 X	CBUS addresses
0000010 X	Reserved for Different Bus Formats
0000011 X	Reserved for future purposes
00001XX X	High-Speed Master Code
11110XX X	10-bit Slave Addressing
11111XX X	Reserved for future purposes

### Procedura generale per l'invio di dati da parte di un master a un dispositivo slave

Il master-trasmettitore invia una condizione di START e indirizza lo slave-receiver

Il trasmettitore master invia i dati al ricevitore slave.

Il trasmettitore master termina il trasferimento con una condizione di STOP.

### Procedura generale per la ricezione/lettura di dati da parte di un master da parte di uno slave

Il master-ricevitore invia una condizione di START e indirizza lo slave-trasmettitore

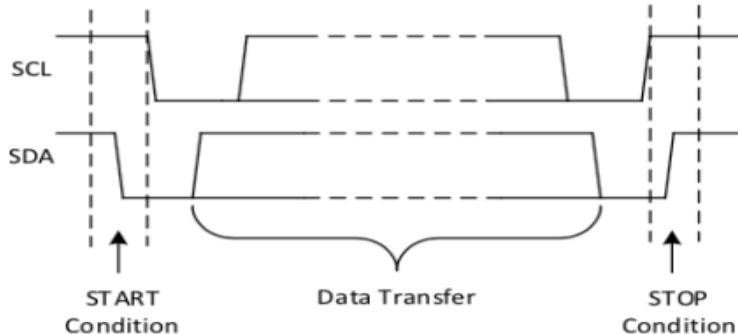
Il master-ricevitore invia allo slave-trasmettitore il registro richiesto da leggere.

Il master-ricevitore riceve i dati dallo slave-transmitter

Il master-ricevitore termina il trasferimento con una condizione di STOP.

## In generale:

La comunicazione viene avviata dal master inviando una condizione di START e termina con l'invio da parte del master una condizione di STOP. Una transizione da alto a basso sulla linea SDA mentre SCL è alto definisce una condizione di START. Una transizione da basso ad alto sulla linea SDA mentre la linea SCL è alta definisce una condizione di STOP.



Una condizione di AVVIO ripetuto è simile a una condizione di START e viene utilizzata al posto di una condizione di condizione di STOP e poi di AVVIO ripetuti. Sembra identica a una condizione di START, ma differisce dalla condizione di START perché si verifica prima di una condizione di STOP (quando il bus non è inattivo). Utile quando il master desidera avviare una nuova comunicazione, ma non vuole lasciare il bus inattivo con la condizione di STOP, che potrebbe far perdere al master il controllo del bus a favore di un altro master (in ambienti multi-master).

Durante ogni impulso di clock di SCL viene trasferito un bit di dati. Un byte è composto da otto bit sulla linea SDA. Un byte può essere un indirizzo di dispositivo, un indirizzo di registro o un dato scritto o letto da uno slavo. I dati vengono trasferiti prima con il bit più significativo (MSB). È possibile trasferire un numero qualsiasi di byte di dati dal master allo slave tra le condizioni di START e di STOP. I dati sulla linea SDA devono rimanere stabili durante la fase alta del periodo di clock. in quanto i cambiamenti nella linea di dati quando SCL è alto vengono interpretate come comandi di controllo (START o STOP).

## Funzionamento

All'interno del sistema AFA, l'LCD viene utilizzato per mostrare i valori di temperatura ed umidità delle varie stanze server. Nel file 3\_CONFSENS.forth vengono settati i parametri dei registri per I2C come segue:

```
\SEZIONE GESTIONE I2C
HEX

OFFSETPI @ 00205000 + CONSTANT I2C0
OFFSETPI @ 00804000 + CONSTANT I2C1
OFFSETPI @ 00805000 + CONSTANT I2C2 \ dedicato all'interfaccia HDMI: NON UTILIZZARE

0 CONSTANT _C          \ CONTROL 1000 0000 1000 0000
4 CONSTANT _S          \ STATUS
8 CONSTANT _DLEN        \ DATA LENGTH 0:15 BIT ESPRIMONO IL NUMERO DI BYTE DA INVIARE
0C CONSTANT _A          \ SLAVE ADDRESS REGISTER 0X27 100111 (PCF8574T)
10 CONSTANT _FIFO        \ DATA FIFO AGGIUNGI UN BYTE ALLA CODA FIFO DA 16 BYTE, USARE BIT 0:7
14 CONSTANT _DIV          \ CORE_CLOCK/_DIV ----> 150 MHZ/ 1500 = 100 KHZ DI DEFAULT, BIT 0:15
18 CONSTANT _DEL          \ data delay register DEFAULT
1C CONSTANT _CLKT         \ clock stretch timeout DEFAULT

\(
  I2Cn REG_NAME -- REG
: SELI2C + ;

\(
  I2C --
: I2CCLEAR _C SELI2C DUP @ 1 5 LSHIFT OR SWAP ! ; \ PULIRE LA FIFO
```

```

\(\ I2C -- )
: I2CDATALEN 1 SWAP _DLEN SELI2C ! ; \ STABILIRE IL NUMERO DI BYTE DA INVIARE

\(\ I2C DATA -- )
: I2CDATA SWAP _FIFO SELI2C ! ; \ SETTARE I DATI DA INVIARE

\(\ I2C -- )
: I2CRESET _S SELI2C 302 SWAP ! ; \RESETTARE IL FLAG DI DONE 1100000010

\(\ I2C ADDR_SLAVE -- )
: I2CSLAVE SWAP _A SELI2C ! ; \ SETTA L'INDIRIZZO DELLO SLAVE

\(\ I2C -- )
: I2CSTART _C SELI2C 8080 SWAP ! ; \INIZIARE LA COMUNICAZIONE

\(\ I2C -- )
: I2CDONE BEGIN DUP _S SELI2C @ 1 1 LSHIFT AND 0 = WHILE REPEAT I2CRESET ; \ASPETTARE CHE L'INVIO TERMINI

\(\DATA I2C -- )
: I2CSEND DUP I2CRESET DUP I2CCLEAR DUP I2CDATALEN DUP ROT I2CDATA DUP I2CSTART I2CDONE ; \ INVIAO SEMPRE
1 BYTE ALLA VOLTA

```

Nel file `8_LCD.forth` vengono creati i comandi per l'inizializzazione dell'LCD e l'invio dei caratteri.

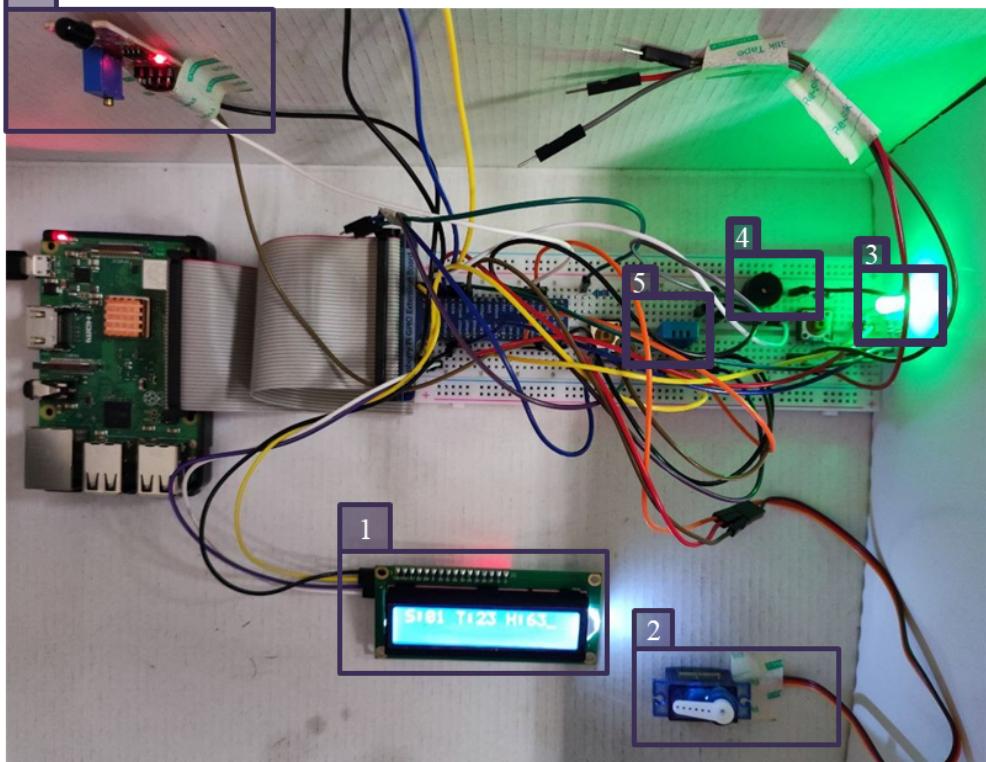
## Descrizione funzionamento generale

In questa sezione descriviamo il funzionamento generale del nostro sistema, mostrando quali sono le principali fasi di funzionamento, illustrando il comportamento delle componenti che interagiscono tra di loro.

### Funzionamento del sistema:

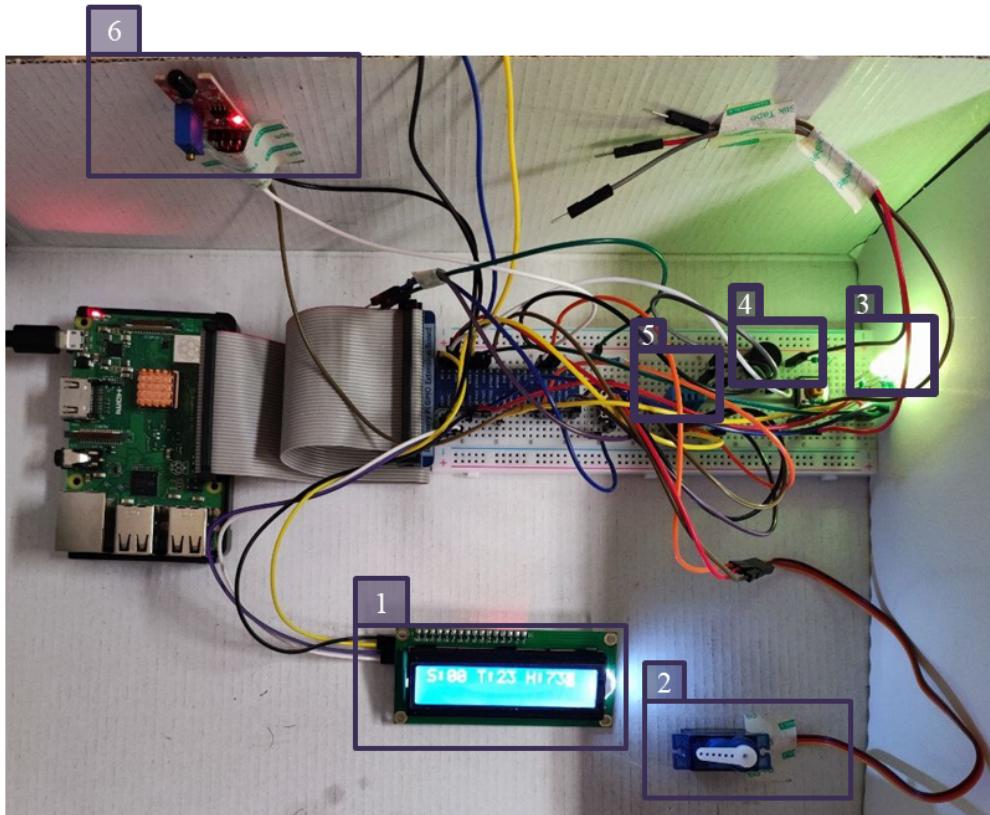
1. Accensione sistema AFA
2. Il sistema, una volta attivato, svolge i seguenti compiti:
  - a. Monitoraggio delle variabili ambientali attraverso l'acquisizione di dati dai sensori
  - b. Mostra sul display led temperatura e umidità delle sale server, cambiando il numero di stanza e i relativi dati visualizzati ogni 3 secondi.
  - c. Attiva i led del semaforo in base alle condizioni rilevate al punto "a" come descritto durante l'analisi del componente RGB LED
  - d. Nel caso in cui si attivi il sensore di fiamma: il sistema setta a rosso, il colore del led della stanza in cui vi è stata la rilevazione dell'incendio, attiva l'allarme acustico e fa scattare la chiusura delle porte tagliafuoco (ruota il servo di 90°), per isolare la sala e ridurre al minimo i danni.
3. Al rientrare dei parametri, il sistema riapre le porte tagliafuoco (ruota di -90° il servo), il led ritorna ad illuminarsi del colore corrispondente ai valori rilevati.

6



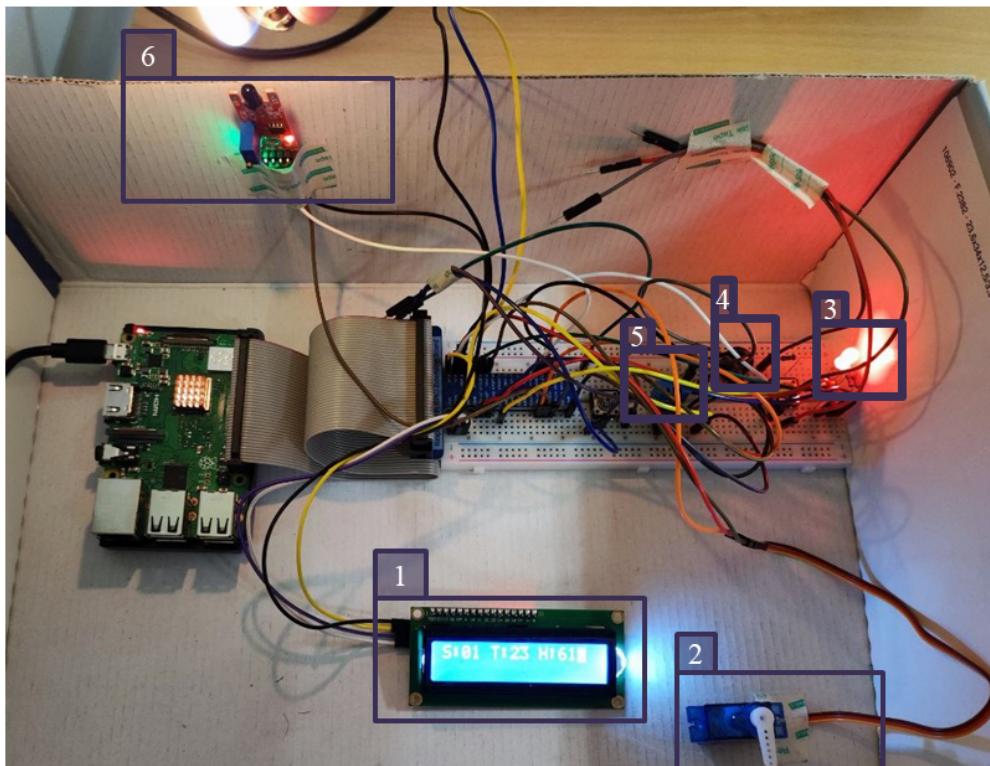
## Valori nella norma

1. S:01 T:23 H:63
2. Porta tagliafuoco aperta
3. Led Verde
4. Nessun allarme sonoro
5. Valori rilevati nella norma
6. Nessuna fiamma rilevata



### Valori fuori norma

1. S:01 T:23 H:73
2. Porta tagliafuoco aperta
3. Led Giallo
4. Nessun allarme sonoro
5. Valori rilevati fuori norma
6. Nessuna fiamma rilevata



### Fiamma rilevata

1. S:01 T:23 H:61
2. Porta tagliafuoco chiusa
3. Led Rosso
4. Allarme sonoro attivo
5. Valori rilevati (irrilevanti)
6. Fiamma rilevata

## 11\_MAIN.forth

DECIMAL

VARIABLE TI4 \ VARIABILE PER VERIFICARE QUANTO TEMPO E' TRASCORSO DALL'ULTIMO CAMBIAMENTO NEL DISPLAY LED  
: INITTI4 TIMESTAMP TI4 ! ;

VARIABLE ROOM \ STANZA SELEZIONATA

VARIABLE II \ CONTATORE CICLO INTERNO

40 CONSTANT MIN\_H \ VALORE MINIMO DI UMIDITA'

69 CONSTANT MAX\_H \ VALORE MASSIMO DI UMIDITA'

10 CONSTANT MIN\_T \ VALORE MINIMO DI TEMPERATURA

27 CONSTANT MAX\_T \ VALORE MASSIMO DI TEMPERATURA

-2 CONSTANT FLAGDIGIT \ VARIABILE DI UTILITA' PER TERMINARE L'INVIO DI DATI

CREATE HUMIDITY NROOMS 1 - ALLOT ALIGN \ ARRAY DOVE VENGONO SALVATI I VALORI DELLE ULTIME MISURAZIONI  
DELL'UMIDITA' PER OGNI STANZA

CREATE TEMPERATURE NROOMS 1 - ALLOT ALIGN \ ARRAY DOVE VENGONO SALVATI I VALORI DELLE ULTIME MISURAZIONI  
DELLA TEMPERATURA PER OGNI STANZA

\ PAROLA PER SALVARE UMIDITA' E TEMPERATURA LETTE NEGLI ARRAY

\( STANZA(II) --- )

: MSTORE CHECKSUM IF DUP HHUMIDITY @ HUMIDITY ROT ASTORE HTEMPERATURE @ TEMPERATURE ROT ASTORE ELSE DROP THEN  
;

\ PAROLA CHE DATO UN NUMERO LO DIVIDE NELLE SUE CIFRE E LE INVIA DALLA PIU' SIGNIFICATIVA ALLA MENO'  
SIGNIFICATIVA AL DISPLAY LCD

\(MISURA --- )

: PRINTNUMBER FLAGDIGIT SWAP BEGIN LASTDIGIT DUP 9 > WHILE REPEAT  
BEGIN DUP FLAGDIGIT <> WHILE DIGITPRINT REPEAT DROP ;

\ PAROLA CHE PERMETTE DI VISUALIZZARE NELL DISPLAY LCD LA SEQUENZA " S:<<NUMERO STANZA>> "  
\(---)

: ROOMDISPLAY STANZA CHASEND PUNTI CHASEND ROOM @ PRINTNUMBER SPACE CHASEND ;

\ PAROLA DI UTILITA' CHE RECUPERA LA TEMPERATURA DELLA STANZA CORRENTE

\(--- TEMPERATURA)

: TEMPFETCH TEMPERATURE II @ AREAD ;

\ PAROLA CHE PERMETTE DI VISUALIZZARE NELL DISPLAY LCD LA SEQUENZA " T:<<TEMPERATURA STANZA>> "

\(---)

: TEMPDISPLAY GRADI CHASEND PUNTI CHASEND TEMPERATURE ROOM @ AREAD PRINTNUMBER SPACE CHASEND ;

\ PAROLA DI UTILITA' CHE RECUPERA L'UMIDITA' DELLA STANZA CORRENTE

\(--- UMIDITA')

: HUMFETCH HUMIDITY II @ AREAD ;

\ PAROLA CHE PERMETTE DI VISUALIZZARE NELL DISPLAY LCD LA SEQUENZA " H:<<UMIDITA STANZA>> "

\(---)

: HUMDISPLAY HUM CHASEND PUNTI CHASEND HUMIDITY ROOM @ AREAD PRINTNUMBER ;

\ COMPOSTA DA

\(---)

: DISPLAY ROOMDISPLAY TEMPDISPLAY HUMDISPLAY ;

\ PAROLA CHE RIPULISCE LO SCHERMO E VISUALIZZA I NUOVI DATI NELL DISPLAY LCD

\(---)

```

: SHOW DISPLAYCLEAR 10000 DELAY DISPLAY ;

\PAROLA PER VERIFICARE CHE TEMPERATURA E UMIDITA' RIENTRANO NEI VALORI DI SOGLIA
\(-- BOOL)
: CHECK HUMFETCH DUP MIN_H > SWAP MAX_H < AND TEMPFETCH DUP MIN_T > SWAP MAX_T < AND AND ;

\PAROLA CHE MODIFICA IL VALORE DI ROOM ( LA STANZA DI CUI VERRANO MOSTRATE LE INFORMAZIONI SUL DISPLAY LCD)
OGNI 3 SEC
\( -- )
: CHANGEROOM TIMESTAMP TI4 @ - 3000000 >= IF ROOM @ 1 + NROOMS MOD ROOM ! -1 INITTI4 ELSE 0 THEN ;

\ PAROLA CHE INIZIALIZZA LE VARIABILI DEL SISTEMA E IL DISPLAY LCD
\(-- )
: RESET INITC LCDINIT BEGIN COUNTER @ WHILE 1 COUNTER DECC
    COUNTER @ WHITE SETCOLOR
    COUNTER @ OFF RING
    COUNTER @ APRI FIREDOOR
    REPEAT -1 ROOM ! 0 II ! INITTI4 ;

\(--)
: MAIN RESET BEGIN
    0 II ! BEGIN
        II @ NROOMS < WHILE
            II @ DUP RILEVAZIONE MSTORE CHECK IF
                II @ GREEN SETCOLOR
                ELSE
                    II @ YELLOW SETCOLOR
                    THEN
                II @ ISFIRE IF
                    II @ DUP DUP RED SETCOLOR ON RING CHIUDI
FIREDOOR
                    ELSE
                        II @ DUP OFF RING APRI FIREDOOR
                    THEN
                        1 II ADDC 1500000 DELAY CHANGEROOM IF SHOW
ELSE THEN
    REPEAT

    AGAIN ;

```

## Descrizione metodo “MAIN”

Il metodo MAIN inizializza le variabili per il funzionamento e contiene due cicli innestati:

1. Il ciclo più esterno BEGIN-AGAIN;
2. Il ciclo più interno, iterando sulle stanze, effettua i dovuti controlli sui valori di temperatura, umidità e presenza di fiamma e pilota RGB LED, servo motore e buzzer.

Al termine del ciclo più interno il comando “CHANGEROOM” verifica se è scaduto l’intervallo temporale stabilito, e in tal caso cambia la stanza da visualizzare sul display lcd.

## Vantaggi

Il software è stato progettato e implementato per essere quanto più scalabile possibile. La scalabilità è data dalla presenza di variabili e comandi adatti ad operare in maniera indipendente dal numero di stanze e componenti presenti.

Ecco alcuni esempi esplicativi:

- A. Aggiunta di una stanza:
  - a. Modificare il valore della variabile NROOMS
  - b. Salvare i valori dei GPIO necessari per il funzionamento delle componenti di una stanza con il comando ASTORE
- B. Aggiunta di un componente:
  - a. Creare l'array che conterrà i valori del GPIO a cui il nuovo componente è collegato in ogni stanza
  - b. Salvare nell'array precedentemente creato il GPIO necessario per il funzionamento del nuovo componente, usando il comando ASTORE.
  - c. Abilitare correttamente tutti i GPIO salvati nell'array usando il comando ABIL.
  - d. Creare eventuale file con il codice per la gestione del nuovo componente.

## Limiti e Miglioramenti

In questa sezione evidenziamo quelli che sono i limiti del sistema AFA, proponendo soluzioni per il miglioramento e il superamento degli stessi.

### **Limite Misurazione tramite sensore DHT11:**

Attraverso la funzione CHECKSUM si è riscontrato che riducendo l'intervallo di tempo tra una misurazione e l'altra sotto 1,5 sec, aumentava esponenzialmente la probabilità di ricevere una misurazione errata. Per questo motivo alla fine del ciclo più interno è stata aggiunto un delay di 1,5 sec. Questa soluzione risulta accettabile in quanto non sono presenti vincoli temporali stringenti per il corretto funzionamento del sistema. Dunque, al più, trascorreranno 1,5 sec dal cambiamento delle condizioni di temperatura, umidità e presenza di fiamma, al loro corretto rilevamento.

**N.B.:** nel codice del main sopra mostrato il delay viene eseguito ad ogni iterazione del ciclo più interno in quanto si possiede un solo sensore e con esso si effettuano le misurazioni come se vi fosse un sensore diverso per ogni stanza.

### **Aggiunta Bottoni Accensione/Spegnimento e scorrimento dati display:**

Per migliorare l'usabilità del sistema AFA, si ritiene opportuno l'introduzione di due bottoni che permettano di accendere e spegnere il sistema e di cambiare la stanza e i dati visualizzati del display a piacimento dell'utente.

**N.B.:** Per l'introduzione di nuovi componenti, il sistema necessita di fonti di alimentazioni esterne.

## Conclusioni

La realizzazione di questo progetto ha comportato l'uso di diversi componenti collegati ai GPIO Raspberry attraverso diverse modalità: GPIO input, GPIO output, PWM, I2C.

In particolare, il sensore DHT11 di temperatura e umidità ha richiesto la corretta fase di handshake tra master e slave e il successivo campionamento del segnale analogico [[Sez. DTH11](#)].

Il segnale dati del sensore è stato collegato direttamente ad un GPIO input, il tutto è stato gestito via software.

