

**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

[Descrizione progetto 2](#_Toc120707333)

[Introduzione 2](#_Toc120707334)

[**Descrizione componenti** 2](#_Toc120707335)

[**Configurazione** 2](#_Toc120707336)

[**Schema del sistema** 3](#_Toc120707337)

## **Descrizione progetto**

## **Introduzione**

L’obiettivo finale del nostro progetto è quello di realizzare un sistema di monitoraggio di temperatura umidità, rilevazioni di anomalie e incendi che permettono la chiusura immediata delle porte tagliafuoco di una sala server, consentendo l’isolamento delle aree durante gli incendi e preservando le altre sale macchine adiacenti dall’avanzare dell’incendio.

### **Descrizione 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
* x1 LCD Display 16x02
* Resistenze 10 kΩ 1kΩ 220kΩ

### **Configurazione**

Si procede con la descrizione della configurazione iniziale e la descrizione 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° | 9\_BUTTONS | File che si occupa dell’implementazione dei comandi per utilizzare i bottoni |
| 11° | 10\_FLAMESENS | File che gestisce il sensore di fiamma |
| 12° | 11\_MAIN | File principale per il funzionamento del sistema AFA. |

#### Schema di collegamento

### **Schema del sistema**

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

\+++++++Schema collegamenti sistema +++++++++\

#### Descrizione

\+++++++Schema collegamenti +++++++++\

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 i servomotori ed il sensore IR
* 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

#### 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.

##### **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**



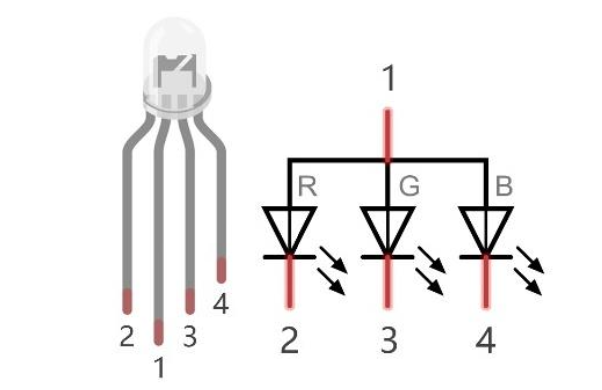
SG90 è un servomotore piccolo e leggero con un'elevata potenza di uscita. Il servo può ruotare di circa 180 gradi (90 in ciascuna direzione).

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.

##### **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.

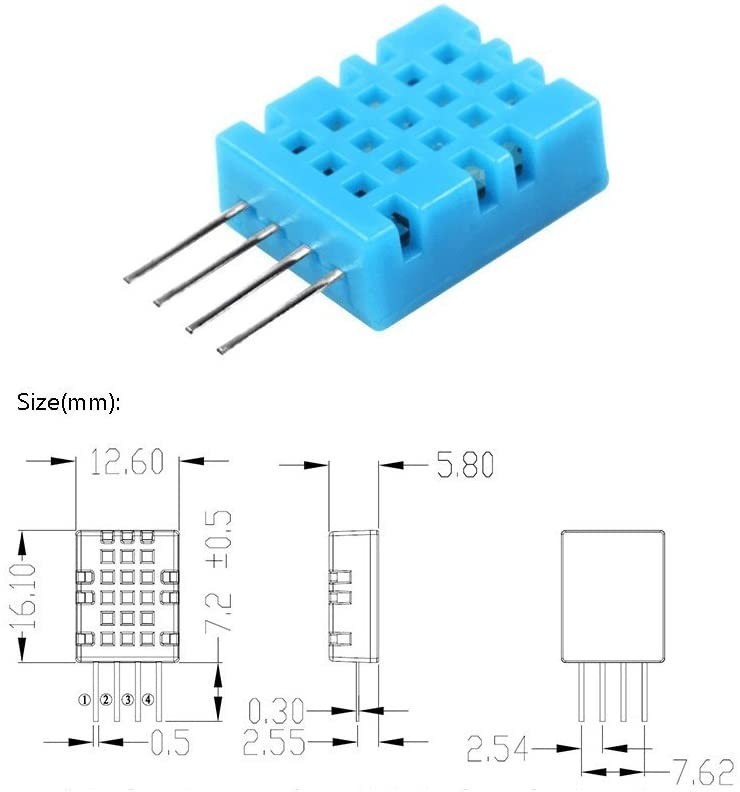
Nel nostro sistema questo componente viene utilizzato per evidenziare lo stato del sistema. Abbiamo tre casi:

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**

Immagine che contiene tavolo

Descrizione generata automaticamente

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:

Immagine che contiene tavolo

Descrizione generata automaticamente

###### 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.

1. ***Tipo di comunicazione:***

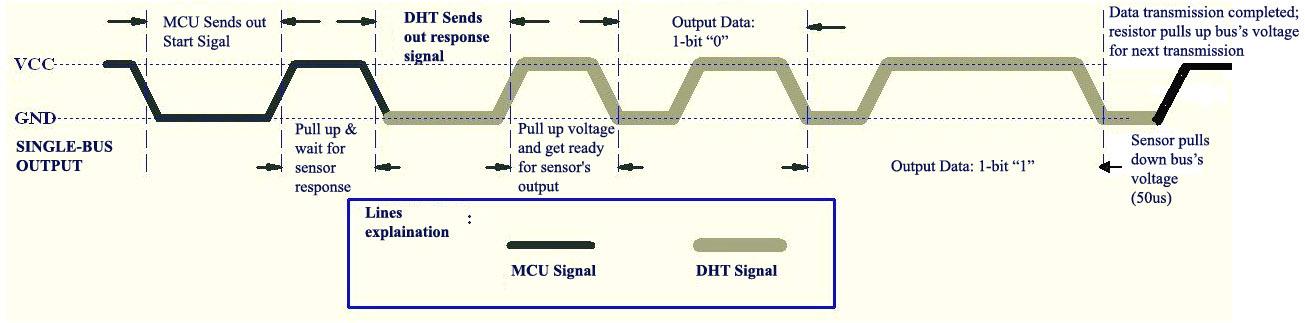
Per la comunicazione e la sincronizzazione tra MCU e sensore DHT11 viene utilizzato il formato dati a bus singolo. Un processo di comunicazione dura circa 4 ms.

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".

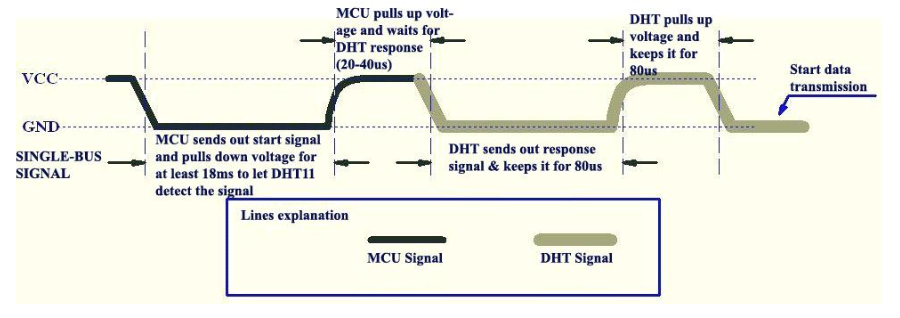
1. ***Processo di comunicazione:***

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.



* 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.

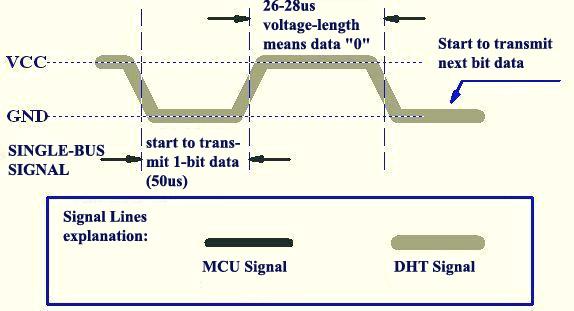
**

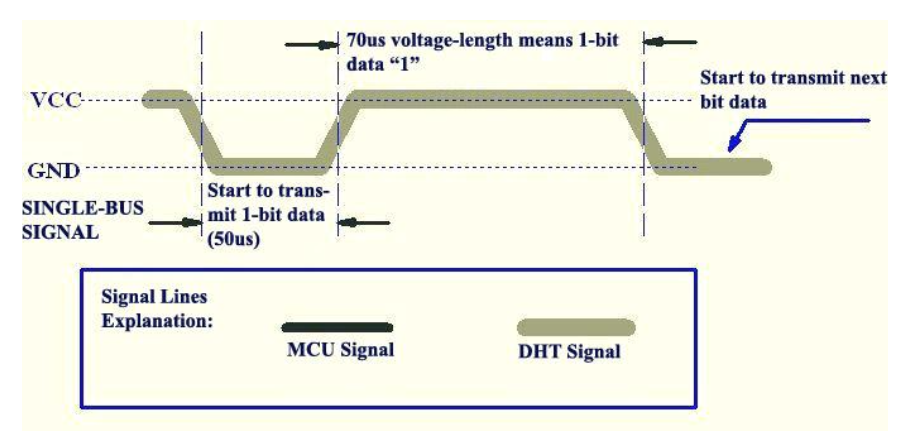
* 1. *DHT risponde a MCU*

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" (vedere le figure 4 e 5).



**

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]