



UNIVERSITÀ  
DEGLI STUDI  
DI PALERMO

EMBEDDED SYSTEMS

# TERMOMETRO DIGITALE

CON RASPBERRY PI 4 MODEL B E  
SENSORE DI TEMPERATURA ED  
UMIDITÀ DHT11

GIUSEPPE CACCIAFEDA

UNIVERSITÀ DEGLI STUDI DI PALERMO

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

16/09/2023

# Sommario

<i>Introduzione</i> .....	3
DESCRIZIONE DEL PROGETTO .....	3
<i>Componenti</i> .....	4
COMPONENTI HARDWARE .....	4
COMPONENTI SOFTWARE .....	4
<i>Hardware</i> .....	5
RASPBERRY PI 4 MODEL B .....	5
SENSORE DI TEMPERATURA ED UMIDITA' DHT11.....	6
CONNETTORE UART-USB CP2102 .....	7
GPIO 40 EXTENSION BOARD .....	8
BREADBOARD.....	9
LED .....	10
RESISTENZE .....	10
DISPLAY LCD 16X2 .....	10
<i>Interfaccia I2C</i> .....	12
FUNZIONAMENTO .....	12
<i>Schema Progettuale</i> .....	13
SCHEMA HEADER GPIO RASPBERRY PI 4 MODEL B E CONNESSIONI .....	13
<i>Funzionamento del Sistema</i> .....	14
CONNESSIONE DEI COMPONENTI .....	14
SCHEMA GPIO .....	14
SENSORE DHT11 .....	14
ANALISI DEGLI INDIRIZZI (BSC, TIMER, GPIO) .....	18
BSC .....	19
TIMER .....	20
GPIO .....	20
CALIBRAZIONE DEL SENSORE DHT11.....	21
<i>Flusso degli Eventi</i> .....	22
FLUSSO DEGLI EVENTI .....	22
MODALITA' OPERATIVE .....	22
<i>Analisi del codice</i> .....	23
JONESFORTH.F.....	23
UTILS.F .....	23
GPIO.F (GESTIONE DEI PIN GPIO) .....	23
CONNECTIONS.F .....	23

TIME.F .....	23
DHT11.F (ACQUISIZIONE E MANIPOLAZIONE DEI DATI TRAMITE DHT11).....	24
LCD.F (VISUALIZZAZIONE DEI DATI SUL DISPLAY LCD) .....	24
SYSTEM.F (GESTIONE DELLE FUNZIONALITA') .....	25
MAIN.F.....	25
<i>Codice</i> .....	26
JONESFORTH.F.....	26
UTILS.F .....	29
GPIO.F.....	29
CONNECTIONS.F .....	31
TIME.F .....	32
DHT.F.....	32
LCD.F.....	36
SYSTEM.F.....	38
MAIN.F.....	40
<i>Setting del Sistema</i> .....	41
SETTING DELL'AMBIENTE DI SVILUPPO.....	41
SETTING DEL RASPBERRY PI .....	41
SETTING DEL PC .....	41
PROCEDURA DI AVVIO TRAMITE MacOS E PICOCOM-3.1 .....	42
<i>Foto Test</i> .....	45
PANORAMICA DI SISTEMA.....	45
AVVIAMENTO DEL SISTEMA.....	45
MISURAZIONE GRADI CELSIUS.....	46
MISURAZIONE GRADI KELVIN .....	46
CLEANING DEL DISPLAY .....	46
RUN DI PROVA .....	47
<i>Misurazione in gradi Celsius</i> .....	47
<i>Misurazione in gradi Kelvin</i> .....	48
<i>Cleaning del Display LCD</i> .....	49

# Introduzione

## DESCRIZIONE DEL PROGETTO

Il seguente progetto consiste nella realizzazione di un termometro digitale che fa un utilizzo combinato di una scheda Raspberry Pi 4 Model B e di un sensore di temperatura ed umidità modello DHT11.

Attraverso l'elaborazione di un codice appositamente sviluppato in ambiente piжForthOs, è stato possibile far funzionare il sensore in modo accurato, permettendo di ottenere una serie di misurazioni coerenti di temperatura e umidità dell'ambiente circostante.

I risultati delle misurazioni saranno mostrati su un display LCD collegato al sistema, che offre più di una semplice visualizzazione dei dati.

Grazie all'aggiunta di una funzionalità extra, è possibile osservare le misurazioni direttamente tramite l'illuminazione di LED dedicati.

Ciascun LED è stato configurato per rispondere ad uno specifico intervallo di temperatura, permettendo una valutazione rapida delle condizioni ambientali.

Il dispositivo offre anche una scelta flessibile nella visualizzazione delle temperature, consentendo di selezionare tra una prima modalità che misura in gradi Celsius ed una seconda che misura in gradi Kelvin.

Tale caratteristica aggiunge versatilità all'esperienza dell'utente.

Una volta che il sistema è stato opportunamente configurato, sarà possibile inizializzarlo immettendo da tastiera il comando di avvio.

Segue la fase di “**calibrazione**” del sensore, che verrà analizzata più dettagliatamente in seguito, la quale garantirà delle misurazioni molto più precise ed accurate.

In sintesi, una volta effettuata una misurazione sarà possibile visualizzare i valori di temperatura ed umidità individuati direttamente sul display LCD e si assisterà all'accensione di uno dei tre LED connessi in base al range di temperatura entro cui cade tale rilevazione.

## AMBITI APPLICATIVI FUTURI

Il seguente progetto rappresenta un perfetto esempio di come sia possibile scegliere componenti con un costo accessibile che allo stesso tempo permettano di realizzare soluzioni pratiche e funzionali, senza compromettere l'integrità e l'efficacia dell'intero sistema.

Questo approccio dimostra che con un'attenta progettazione sia hardware che software è possibile ottenere risultati soddisfacenti senza dover investire cifre consistenti.

Sebbene il progetto sia stato realizzato su piccola scala, la sua natura modulare e la scelta di componenti di base affidabili aprono la strada a potenziali sviluppi futuri su larga scala.

In un ambiente lavorativo, ad esempio, una possibile implementazione potrebbe comportare l'utilizzo di componenti di maggior precisione e costo.

L'esperienza acquisita attraverso questa versione iniziale del progetto pone le basi per futuri miglioramenti, consentendo di esplorare opportunità di integrazione con tecnologie avanzate e componenti di alta qualità.

Un esempio concreto potrebbe essere quello di una futura integrazione all'interno di una stazione metereologica.

# Componenti

## COMPONENTI HARDWARE

I componenti elettronici e computazioni che sono stati impiegati per la realizzazione del progetto sono i seguenti:

- **Raspberry Pi 4 Model B 4GB**
- **Sensore di Temperatura ed umidità DHT11**
- **Connettore UART-USB**
- **GPIO 40 Extension Board**
- **Breadboard**
- **3 LED (Blu, Rosso, Verde)**
- **3 Resistenze da 220 Ohm**
- **Display LCD 16x2**
- **Cavi Jumper M/M e M/F**
- **Scheda MicroSD 64GB**
- **Alimentatore con cavo USB-C**
- **PC (Mac OS X)**

## COMPONENTI SOFTWARE

Per lo sviluppo software del progetto sono stati utilizzati contemporaneamente due componenti software per garantire una **programmazione interattiva** direttamente sul Target designato attraverso l'utilizzo di un dispositivo Host (PC).

Installato sul Target:

- **pijFORTHos**: sistema operativo con interprete FORTH basato su un linguaggio con grammatica Post-Fissa, il quale permette un'alta interazione con la componente Hardware sottostante per soluzioni Bare-Metal e programmazione di basso livello.

A seconda del sistema operativo installato sull' Host è necessario utilizzare uno tra i due seguenti componenti software per interagire con il target mediante l'interfaccia seriale UART.

Windows e MacOS:

- **ZOC8 Terminal**

MacOS e Linux:

- **Picocom-3.1**

È possibile utilizzare un qualsiasi software di tipo open-source il quale permetta la comunicazione seriale mediante l'utilizzo di interfaccia UART.

# Hardware

## RASPBERRY PI 4 MODEL B

Il Raspberry Pi 4 Model B è un **Single Board Computer** (SBC) ad alte prestazioni prodotto dalla Raspberry Pi Foundation. Il cuore della scheda è costituito dal SoC BCM2711 Broadcom, il quale integra un processore quad-core ARM Cortex-A72 da 1,5 GHz (a seconda della variante), unitamente a diverse periferiche integrate come GPIO, USB, I2C, SPI, UART, PWM, offrendo di conseguenza un'ampia potenza di calcolo per una grande varietà di applicazioni. Dispone di una memoria RAM da 2 fino ad 8GB, porte USB 3.0 e 2.0, connettività Gigabit Ethernet, Wi-Fi 802.11ac integrato e Bluetooth 5.0 per una connessione flessibile.

Supporta la visualizzazione su due schermi 4K a 60 Hz tramite porte micro-HDMI e può essere alimentato tramite una porta USB-C.

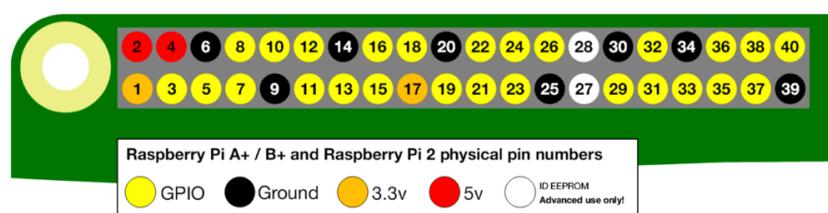
È in grado di supportare ed eseguire sistemi operativi come Linux, fornendo una piattaforma flessibile per lo sviluppo di progetti che richiedono calcoli computazionali, controllo di periferiche e comunicazione dati. Con il suo ecosistema di sviluppo, il Raspberry Pi 4 Model B è ampiamente utilizzato per progetti di apprendimento, domotica, sviluppo software e altro ancora.



Dispone inoltre di un **Header General Purpose Input/Output** a 40 pin che consente di connettere la scheda ad una ampia varietà di dispositivi esterni, quali sensori, attuatori e display.

Quest'ultimo comprende anche dei pin per la comunicazione seriale, **I2C** e **SPI**, che possono essere utilizzati per comunicare con dispositivi esterni, quali per esempio display LCD.

I pin GPIO possono essere programmati per divenire ingressi o uscite digitali, consentendo al Raspberry Pi 4 Model B di acquisire dati da sensori esterni.

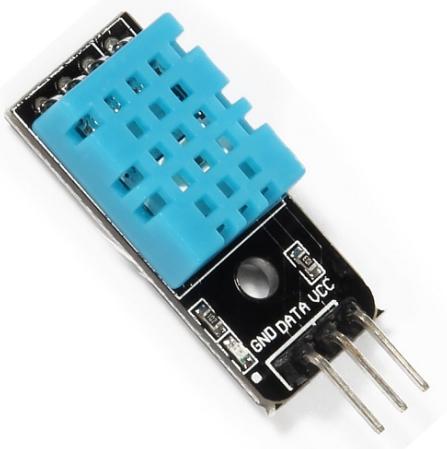


Caratteristiche:

- **Processore:** Quad-core ARM Cortex-A72 con clock a 1.5 GHz;
- **RAM:** 4 GB;
- **Connettività:**
  - due porte USB 3.0 e due porte USB 2.0 per dispositivi periferici;
  - porta Ethernet Gigabit per la connettività di rete cablata;
  - Wi-Fi 802.11ac integrato e Bluetooth 5.0 per la connettività wireless;
- **Video:** supporta la visualizzazione su due schermi tramite porte micro-HDMI con risoluzioni fino a 4K a 60 Hz;
- **Audio:** uscita audio tramite jack da 3,5 mm o HDMI, supporta l'audio ad alta definizione;
- **Alimentazione:** richiede un'alimentazione da 5V 3A tramite connettore USB-C;
- **Archiviazione:** slot per schede microSD per il sistema operativo e l'archiviazione dei dati;
- **GPIO:** 40 pin GPIO per il collegamento di sensori, attuatori e altre periferiche;
- **Porte:** porta di alimentazione USB-C, pin GPIO aggiuntivi;
- **Sistema Operativo:** supporta vari sistemi operativi, inclusi diversi tipi di distribuzioni Linux come Raspberry Pi OS (precedentemente chiamato Raspbian), Ubuntu e altri ancora;
- **USB Boot:** può avviarsi da un dispositivo di archiviazione USB esterno, oltre alla scheda microSD;

#### SENSORE DI TEMPERATURA ED UMIDITA' DHT11

Il sensore di temperatura DHT11 è un dispositivo di rilevamento di temperatura e umidità a basso costo. Esso fornisce dati affidabili di temperatura nell'intervallo da 0°C a 50°C e umidità dal 20% al 90% RH. Dotato di una risoluzione di 1°C per la temperatura e 1% RH per l'umidità, il DHT11 utilizza un protocollo di comunicazione a singolo filo per trasmettere i dati in formato digitale. Funziona con tensioni di alimentazione di solito di 3.3V o 5V DC, e la sua interfaccia si collega facilmente a microcontrollori e microprocessori attraverso pin di input/output digitali. Il sensore ha un tempo di risposta compreso entro i 2 secondi sia per la rilevazione della temperatura che dell'umidità. Grazie alle sue caratteristiche, è spesso utilizzato per monitorare l'ambiente in progetti come stazioni meteorologiche, monitoraggio dell'umidità in abitazioni e sistemi di automazione domestica.

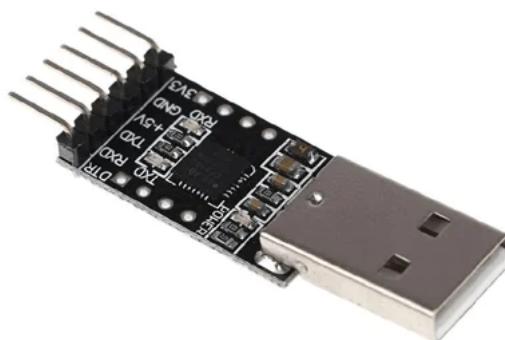


Caratteristiche:

- **Range di misurazione:**
  - Temperatura: da 0°C a 50°C (da 32°F a 122°F);
  - Umidità: dal 20% al 90% RH;
- **Accuratezza:**
  - Temperatura:  $\pm 2^\circ\text{C}$  ( $\pm 3.6^\circ\text{F}$ );
  - Umidità:  $\pm 5\%$  RH;
- **Risoluzione:**
  - Temperatura:  $1^\circ\text{C}$  ( $1.8^\circ\text{F}$ );
  - Umidità: 1% RH;
- **Output:**
  - Uscita digitale tramite un protocollo di comunicazione a singolo filo;
  - Dati di temperatura e umidità inviati insieme;
- **Alimentazione:**
  - Tipicamente funziona a 3,3V o 5V;
  - Basso consumo energetico;
- **Interfaccia:**
  - Protocollo di comunicazione a bus singolo;
  - Si collega a microcontrollori e microprocessori tramite pin di input/output digitali;
- **Response Time:**
  - Temperatura: entro 2 secondi;
  - Umidità: entro 2 secondi;

#### CONNETTORE UART-USB CP2102

Il connettore UART CP2102, Universal Asynchronous Receiver-Transmitter, è un'interfaccia di comunicazione seriale ampiamente utilizzata in elettronica. Essa consente la trasmissione e la ricezione di dati tra dispositivi digitali sfruttando una linea trasmittente ed una ricevente.



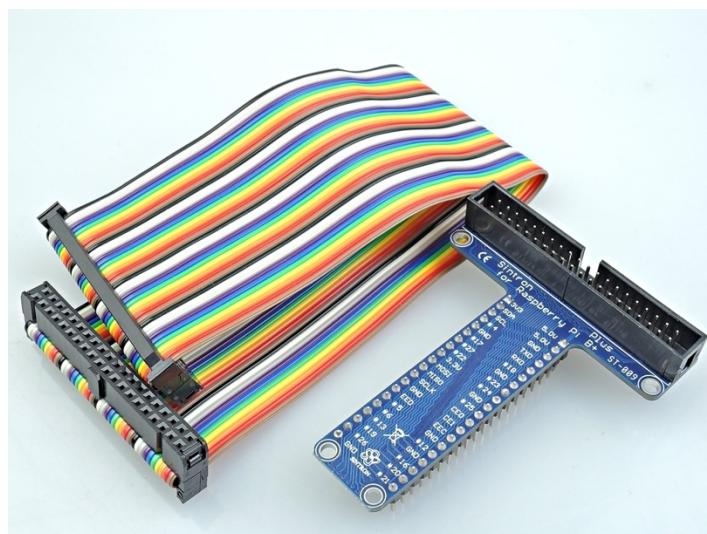
Caratteristiche:

- **Interfaccia USB:** connettore USB per il collegamento al computer o ad altri dispositivi tramite porta USB;
- **Interfaccia TTL:** connettori o pin per il collegamento ai dispositivi TTL (Transistor-Transistor Logic) come microcontrollori o microprocessori;

- **Tensione di alimentazione:** solitamente si opera a 5V o 3.3V, a seconda delle specifiche del dispositivo TTL;
- **Segnali TTL:** supporta segnali digitali a livelli TTL, come TX (trasmissione) e RX (ricezione), tipicamente utilizzati per la comunicazione seriale;
- **Velocità di comunicazione:** supporta diverse velocità di comunicazione seriale come 9600 bps, 115200 bps, ecc;

#### GPIO 40 EXTENSION BOARD

La scheda di estensione GPIO a 40 pin è un accessorio progettato per ampliare le capacità di input/output generico (GPIO) di dispositivi come il Raspberry Pi, che dispongono di un connettore a 40 pin. Queste schede offrono una maggiore flessibilità e possibilità di connessione per progetti elettronici.

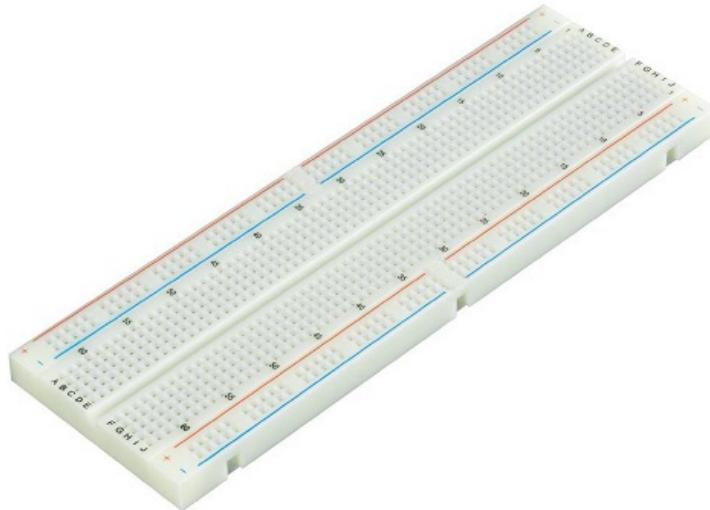


Caratteristiche:

- **Connettività:** consente di estendere le capacità di input/output general-purpose (GPIO) di un dispositivo, spesso utilizzando il layout a 40 pin;
- **Pin GPIO:** offre un ulteriore set di pin GPIO (Input/Output a uso generale) che possono essere utilizzati per collegare sensori, attuatori e altre periferiche;
- **Compatibilità:** progettato per funzionare con dispositivi che hanno un layout a 40 pin, come ad esempio alcune schede Raspberry Pi;
- **Interfaccia:** connettore per collegare la scheda di estensione alla scheda madre o al dispositivo principale;
- **Alimentazione:** potrebbe includere opzioni per l'alimentazione attraverso i pin GPIO, con i livelli di tensione supportati chiaramente specificati;
- **Utilizzo:** utilizzata per espandere le capacità di I/O dei dispositivi a 40 pin, consentendo una maggiore connettività e possibilità di interazione con il mondo esterno;

## BREADBOARD

Una breadboard è un pannello rettangolare con una superficie piatta e perforata, solitamente realizzata in plastica. La superficie è suddivisa in piccoli fori rettangolari, disposti in righe e colonne, che formano delle "strisce" di connessioni elettriche. Queste strisce sono utilizzate per collegare i componenti elettronici tra loro.



Caratteristiche:

- **Fori e Strisce:** i fori sulla breadboard sono ideali per inserire i pin dei componenti elettronici, come resistenze, condensatori, transistor, I2C e molti altri; i fori sono collegati tra loro in strisce lungo le colonne e le righe, permettendo di creare collegamenti elettrici senza saldature;
- **Strisce di Alimentazione:** le strisce orizzontali sui bordi della breadboard spesso rappresentano le strisce di alimentazione; queste sono collegate a tensioni specifiche, come positivo (+) e negativo (-), consentendo l'alimentazione dei componenti;
- **Zone Separabili:** una breadboard è generalmente divisa in sezioni separate, ma collegate; ciò aiuta a evitare interferenze elettroniche indesiderate;
- **Versatilità:** poiché i componenti possono essere inseriti e rimossi facilmente, le breadboard sono strumenti versatili per la prototipazione rapida e l'apprendimento dell'elettronica;

## LED

Il LED è un componente a semiconduttore costituito da materiali che conducono l'elettricità in una sola direzione. Quando una corrente elettrica attraversa il LED, gli elettroni all'interno del materiale si spostano da un livello di energia più alto a uno più basso, rilasciando energia sotto forma di fotoni. Questi fotoni sono la luce visibile emessa dal LED.



## RESISTENZE

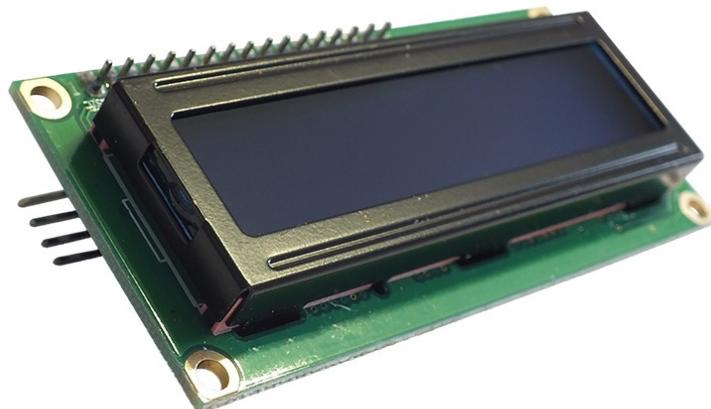
La resistenza è un componente realizzato generalmente da un materiale conduttivo, come il carbone o il metallo, che offre una determinata resistenza al passaggio della corrente elettrica. Essa è caratterizzata da un valore di resistenza elettrica misurato in ohm ( $\Omega$ ), che determina la quantità di corrente che scorre attraverso di essa quando una tensione è applicata.



## DISPLAY LCD 16X2

Un display LCD 16x2 è un tipo di schermo a cristalli liquidi utilizzato per visualizzare testo e informazioni in formato testuale.

Il display LCD 16x2 è composto da 16 caratteri per riga e 2 righe, per un totale di 32 caratteri visualizzabili. Ciascun carattere è composto da una matrice di pixel, in genere 5x8 oppure 5x10 pixel, che rappresenta ciascun carattere alfanumerico e simbolo. Questo tipo di schermo è ampiamente utilizzato per visualizzare informazioni in testo in progetti elettronici, dispositivi di controllo e interfacce utente.



Caratteristiche:

- **Dimensioni:** il termine "16x2" si riferisce alle dimensioni del display, con 16 caratteri per riga e 2 righe; ciò offre abbastanza spazio per visualizzare testo in modo chiaro e leggibile;
- **Retroilluminazione:** molti display LCD 16x2 sono dotati di retroilluminazione a LED, spesso blu o verde, che permette la visualizzazione in condizioni di scarsa illuminazione;
- **Controller:** i display LCD 16x2 sono controllati da un chip integrato, come il popolare controller HD44780, che semplifica l'interfacciamento con microcontrollori e microprocessori;
- **Interfaccia:** utilizzano spesso un'interfaccia parallela per i dati, che richiede più pin per il collegamento, ma ne semplifica il controllo e l'aggiornamento del contenuto;
- **Tensione di Alimentazione:** richiedono un'alimentazione tipica di 5V o 3.3V a seconda del modello;
- **Caratteri Personalizzati:** alcuni modelli consentono la creazione di caratteri personalizzati, estendendo le capacità di visualizzazione;

Segue la tabella con la corrispettiva codifica binaria di ciascun carattere:

	Upper 4 Bits	Lower 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		CG RAM (1)																
xxxx0000			Ø	ø	P	‘	F					-	‑	ℳ	ø	P		
xxxx0001	(2)		!	1	A	Q	a	q			.	ℳ	ℳ	ä	q			
xxxx0010	(3)		"	2	B	R	b	r			‘	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx0011	(4)		#	3	C	S	c	s			’	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx0100	(5)		\$	4	D	T	d	t			、	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx0101	(6)		%	5	E	U	e	u			・	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx0110	(7)		&	6	F	V	f	v			ヲ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx0111	(8)		'	7	G	W	g	w			ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx1000	(1)		(	8	H	X	h	x			イ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx1001	(2)		)	9	I	Y	i	y			カ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx1010	(3)		*	:	J	Z	j	z			エ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx1011	(4)		+	;	K	C	k	c			オ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx1100	(5)		,	<	L	¥	l	l			シ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx1101	(6)		-	=	M	]m	)				ヌ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx1110	(7)		.	>	N	^	n	+			ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
xxxx1111	(8)		/	?	O	_	o	+			ツ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	

# Interfaccia I2C

## FUNZIONAMENTO

L'interfaccia I2C (**Inter-Integrated Circuit**) rappresenta un sistema ottimizzato per l'interconnessione del display LCD 16x2 con il Raspberry Pi 4 Model B.

Questa tipologia di comunicazione di natura **seriale** agevola l'apertura di un canale di scambio dati tra il SBC ed il periferico di visualizzazione, ossia il display LCD.

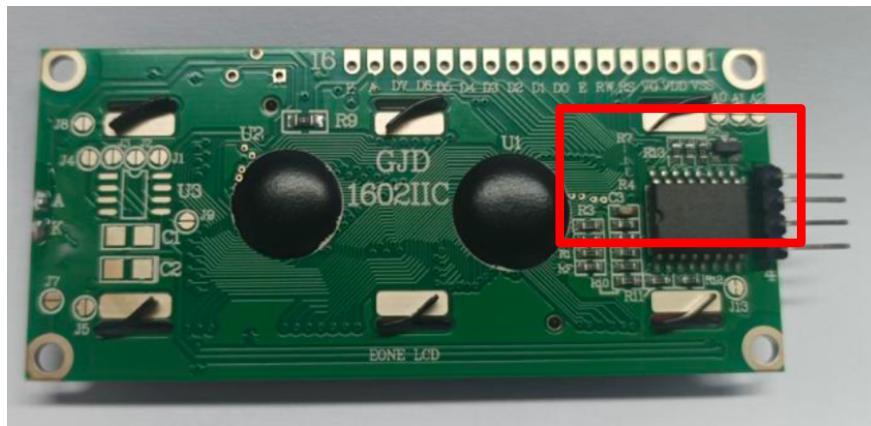
La peculiarità principale di tale interfaccia risiede nella sua capacità di operare con una quantità limitata di pin, un aspetto di notevole rilevanza per semplificare la disposizione fisica dei collegamenti e per liberare risorse GPIO da poter sfruttare in altre funzionalità.

Il Raspberry Pi 4 Model B dispone di porte GPIO (General Purpose Input/Output) predisposte per l'implementazione della comunicazione I2C.

Tale modalità di comunicazione, caratterizzata da un flusso sincrono e bidirezionale di informazioni, stabilisce un canale di dialogo strutturato tra i dispositivi.

In tale contesto, sono definiti due canali specifici, noti come SDA (**Serial Data Line**) e SCL (**Serial Clock Line**).

Il funzionamento del protocollo I2C prevede che il dispositivo "master" (Raspberry Pi) generi il segnale di clock, lo inoltri tramite la relativa linea dedicata (SCL) e si occupi della trasmissione o della ricezione dei dati attraverso la linea dati (SDA), realizzando un'architettura di tipo "multi-master" o "multi-slave". Per ciascun ciclo di trasmissione-ricezione può essere scelto un solo device master e lo si può fare mediante circuiteria specifica oppure direttamente pre-impostandolo.



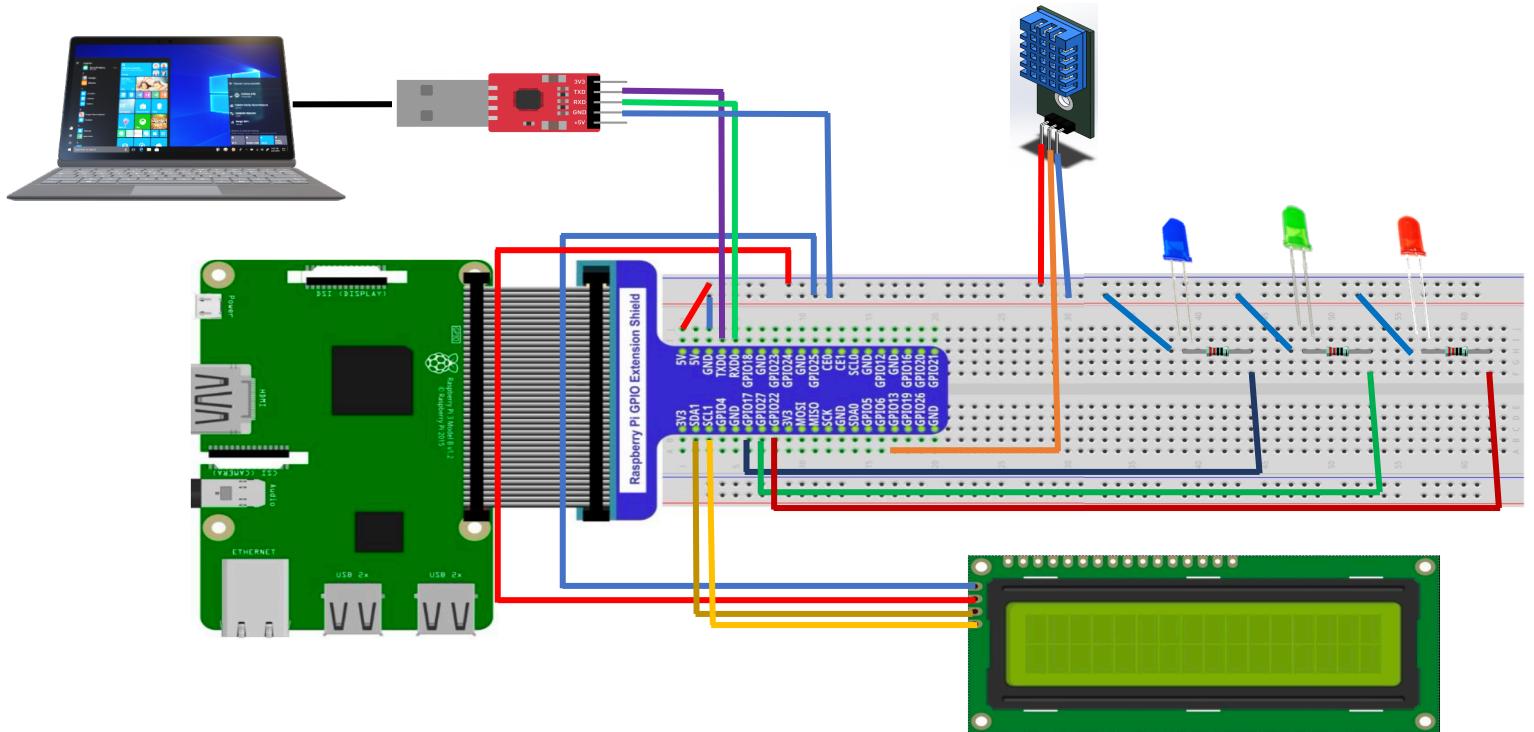
Il display LCD, a sua volta, risponde confermando la ricezione dei dati oppure fornendo informazioni di ritorno, come il proprio stato corrente.

Ogni dispositivo I2C è caratterizzato da un indirizzo univoco che consente al dispositivo "master" (Raspberry Pi) di discriminare tra le unità presenti nella rete e di stabilire quale sia l'obiettivo della comunicazione.

Nel caso del display LCD 16x2, il suo indirizzo I2C è predefinito e spesso si aggira intorno a valori come **0x27** o **0x3F**, sebbene possa variare in base al produttore o alle configurazioni specifiche.

Mediante il protocollo I2C, il Raspberry Pi orchestra l'invio di comandi e dati al display LCD. È possibile inviare sequenze di byte per posizionare il cursore su una specifica riga o colonna dello schermo, scrivere caratteri o stringhe di testo e regolare la retroilluminazione in maniera interattiva.

# Schema Progettuale



SCHEMA HEADER GPIO RASPBERRY PI 4 MODEL B E CONNESSIONI

CONNECTION	HEADER	PIN	HEADER	CONNECTION
+3.3V POWER		1	2	+5V POWER DEVICES – VCC*
LCD - SDA	GPIO 2 (SDA1)	3	4	+5V POWER
LCD - SCL	GPIO 3 (SCL1)	5	6	GROUND DEVICES – GND*
	GPIO 4	7	8	GPIO 14 (TXD0) CP2102 – TX
	GROUND	9	10	GPIO 15 (RXD0) CP2102 – RX
LED BLUE+	GPIO 17	11	12	GPIO 18
LED GREEN+	GPIO 27	13	14	GROUND
LED RED+	GPIO 22	15	16	GPIO 23
+3.3V POWER		17	18	GPIO 24
	GPIO 19 (MOSI)	19	20	GROUND
	GPIO 9 (MISO)	21	22	GPIO 25
	GPIO 11 (SCLK)	23	24	GPIO 8 (CE0#)
	GROUND	25	26	GPIO 7 (CE1#)
	GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
	GPIO 5	29	30	GROUND
	GPIO 6	31	32	GPIO 12
DHT11 - DATA	GPIO 13	33	34	GROUND
	GPIO 19 (MISO)	35	36	GPIO 16 (CE2#)
	GPIO 26	37	38	GPIO 20 (MOSI)
	GROUND	39	40	GPIO 21 (SCLK)

\* Sfruttando la Breadboard è stato possibile utilizzare un unico canale di alimentazione per fornire una tensione di 5V al display LCD ed al sensore DHT11, ed un unico canale per fornire il GND al display LCD, al sensore DHT11, al connettore CP2102 (autoalimentato) ed ai LED.

# Funzionamento del Sistema

## CONNESSIONE DEI COMPONENTI

Una volta che tutti i componenti sono stati correttamente connessi tra di loro, rispettando le indicazioni riportate nello schema precedente, è possibile instaurare una connessione seriale **Universal Asynchronous Receiver-Transmitter (UART)** tra il Raspberry Pi e l'Host, la quale favorirà una **programmazione interattiva** direttamente sul Target sfruttando il terminale FORTH integrato nell'ambiente **pijFORTHos** installato sulla MicroSD appositamente “flashata” utilizzando il software **Raspberry Pi Imager**.

## SCHEMA GPIO

Di default le diverse porte GPIO del Raspberry Pi sono impostate in modalità **input** e sarà dunque necessario switchare le funzionalità delle porte interessate in base al compito designato.

Oltre alle funzioni **Input** ed **Output**, le GPIO posseggono delle cosiddette **Alternative Functions** che si differenziano a seconda della porta in esame.

Ai fini della realizzazione e comprensione del progetto, il relativo schema GPIO è il seguente:

GPIO	FUNCTION	UTILIZZO
2	ALT FUNCTION 0	SDA I2C LCD
3	ALT FUNCTION 0	SCL I2C LCD
13	INPUT/OUTPUT	DHT11
14	TX	UART TX
15	RX	UART RX
17	INPUT	BLUE LED
22	INPUT	RED LED
27	INPUT	GREEN LED

Si noti come:

- **GPIO 2, 3** siano necessarie per instaurare la comunicazione attraverso il bus **I2C** tra il Target ed il display LCD;
- **GPIO 13** cambia la propria funzionalità da **Input** ad **Output** per far sì che lo **start signal** possa essere inoltrato al sensore DHT11 e che successivamente si possano leggere i dati trasmessi da quest'ultimo al Target;
- **GPIO 14, 15** siano necessarie per instaurare il collegamento **UART** tra il Target ed il PC sfruttando le porte TX (transmitter) ed RX (receiver);
- **GPIO 17, 22, 23** siano necessari per pilotare i vari LED (Blu, Verde, Rosso);

## SENSORE DHT11

Il sensore DHT11 è un dispositivo che integra sensori capacitivi e termistori per misurare l'umidità relativa e la temperatura dell'ambiente circostante. Questo processo di misurazione è fondato su principi fisici di variazione elettrica in risposta alle variazioni dell'umidità e della temperatura.

Il rilevamento della temperatura si basa su un termistore, un componente elettronico con resistenza che varia in modo inversamente proporzionale alla temperatura. Questo termistore è inserito in un circuito di resistenza diviso, creando un ponte di resistenza. La differenza di resistenza tra le due parti del termistore varia in funzione della temperatura.

Il microcontrollore interno del DHT11 interpreta questa variazione di resistenza come una misura di temperatura.

All'interno del DHT11, il microcontrollore esegue operazioni di campionamento, conversione e calibrazione sui segnali provenienti dai sensori. Queste operazioni comprendono la conversione delle variazioni di capacità e resistenza in valori digitali che possono essere trasmessi per l'analisi.

La comunicazione dei dati avviene attraverso un protocollo seriale. Il DHT11 comunica con il mondo esterno inviando segnali seriali che rappresentano i dati di umidità e temperatura.

Questi segnali vengono interpretati e decodificati da un dispositivo esterno, come un microcontrollore o un computer, che traduce i dati grezzi in valori comprensibili di umidità relativa e temperatura.

Viene utilizzato un formato di dati su un singolo bus per la comunicazione e la sincronizzazione tra il BSC ed il sensore DHT11.

I dati sono composti da parti decimali e intere. Una trasmissione completa di dati è costituita da 40 bit, inviando per primi i dati più significativi.

Il formato dei dati inviati dal sensore è il seguente:

- 8 bit di dati relativi alla parte intera dell'umidità RH
- 8 bit di dati relativi alla parte decimale dell'umidità RH
- 8 bit di dati relativi alla parte intera della temperatura T
- 8 bit di dati relativi alla parte decimale della temperatura T
- 8 bit di checksum

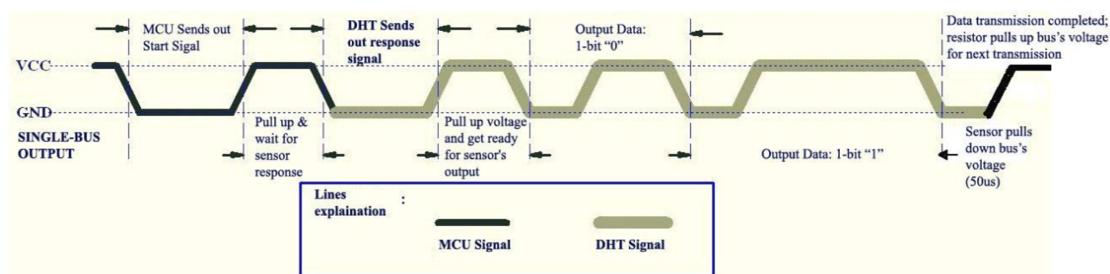
Se la trasmissione dei dati è corretta, la checksum dovrebbe coincidere con gli ultimi 8 bit dei primi 32 inoltrati ("8 bit di dati interi sull'umidità relativa + 8 bit di dati decimali sull'umidità relativa + 8 bit di dati interi sulla temperatura + 8 bit di dati decimali sulla temperatura").

Quando il microcontrollore (impropriamente il Raspberry) invia un segnale di avvio (**start signal**), il sensore DHT11 passa dalla modalità a basso consumo energetico alla modalità operativa, rimanendo in attesa che il MCU completi il segnale di avvio.

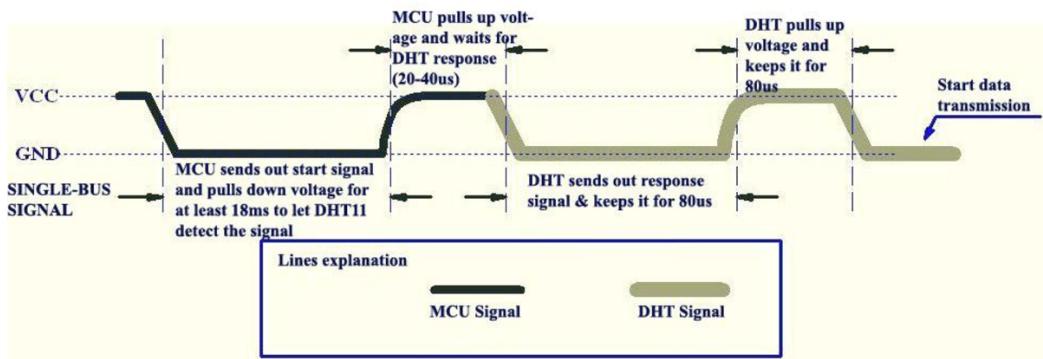
Una volta completato, il DHT11 invia un segnale di risposta composto da 40 bit che includono le informazioni sull'umidità e sulla temperatura relative all'ambiente circostante.

Senza il segnale di avvio dall'MCU, il DHT11 non invierà alcun segnale di risposta.

Una volta raccolti i dati, il DHT11 tornerà alla modalità a basso consumo energetico fino a quando non riceverà nuovamente un segnale di avvio dal MCU.



Per avviare la comunicazione, il microcontrollore (MCU) invia uno **start signal** settando il voltaggio del bus dati al livello **basso** per almeno 18 ms (millisecondi), quindi commuta il voltaggio del bus al livello **alto** ed attende dai 20 a 40  $\mu$ s (microsecondi) per ricevere la risposta dal sensore DHT11.

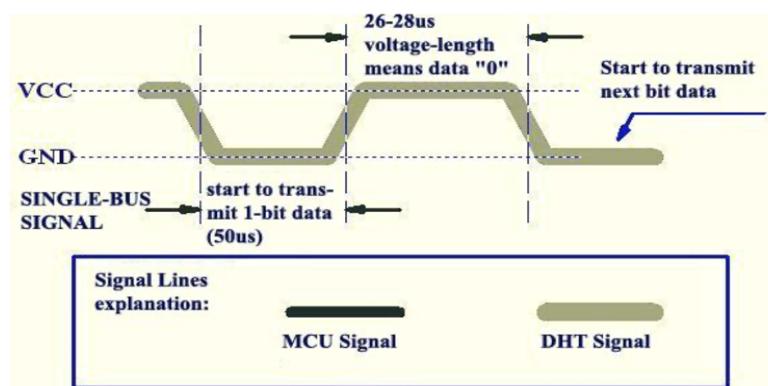


Una volta che il DHT ha rilevato il segnale di avvio, invierà un segnale di risposta a basso livello di tensione, che durerà 80  $\mu$ s.

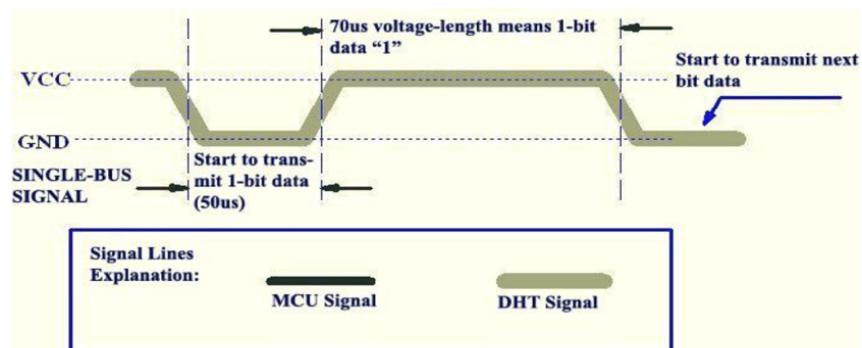
Successivamente, il programma del DHT imposta il livello di tensione del singolo bus dati da basso ad alto e lo mantiene per 80  $\mu$ s per prepararsi alla trasmissione dei dati.

Quando il DATA Single-Bus è a livello di tensione bassa, ciò significa che il DHT sta inviando il segnale di risposta.

Occorrono circa 50  $\mu$ s per inizializzare la trasmissione dei dati: in base alla durata ed al livello di tensione del segnale è possibile determinare se il bit di dati trasmesso avrà valore "0" (tensione mantenuta al livello alto per 26-28  $\mu$ s) o "1" (tensione mantenuta al livello alto per 70  $\mu$ s). Viene mostrato un esempio in cui il bit è "0":



Ed un altro in cui il bit è "1":



Se il segnale di risposta proveniente dal DHT è costantemente ad un livello di tensione elevato, ciò indica che il DHT non sta rispondendo correttamente e ciò potrebbe essere riconducibile ad un collegamento errato.

Quando l'ultimo bit di dati viene trasmesso, il DHT11 abbassa il livello di tensione e lo mantiene per 50 µs. Successivamente, la tensione sul Single-Bus verrà alzata dal resistore per riportarla allo stato libero.

Per la ricezione dei dati all'interno del progetto, è stato deciso di salvare i bit ricevuti in una variabile "**BITS**", della quale viene eseguito uno shift a sinistra ad ogni nuovo bit ricevuto al fine di conservarlo correttamente.

Si è deciso in ultimo luogo di eseguire un'operazione di filtraggio e manipolazione delle misurazioni di temperatura e di umidità rilevate al fine di "**calibrare**" il dispositivo.

## DISPLAY LCD 16X2

Un display LCD 16x2 è un dispositivo che utilizza cristalli liquidi per visualizzare testo e simboli su una griglia di 16 colonne e 2 righe.

Al suo interno vi sono dei segmenti che possono essere attivati o disattivati per formare dei caratteri. Questi segmenti sono formati da materiali a cristalli liquidi che possono cambiare la loro opacità in risposta ad una tensione elettrica. La retroilluminazione del display consente di visualizzare i caratteri anche in condizioni di scarsa illuminazione.

Il microcontrollore o un altro dispositivo di controllo invia comandi al display attraverso un'interfaccia standard, come I2C o SPI. Questi comandi includono istruzioni per posizionare il cursore, cancellare il display o scrivere testo.

I driver all'interno del display interpretano questi comandi e attivano i segmenti necessari per visualizzare i caratteri richiesti.

Per creare caratteri e simboli, i segmenti appropriati vengono attivati in modo da formare l'immagine desiderata. Ad esempio, una lettera "A" verrà visualizzata attivando i segmenti corrispondenti nella griglia.

L'aggiornamento del display è costante. Ogni carattere visualizzato richiede l'attivazione o la disattivazione dei segmenti appropriati, e il display deve essere costantemente controllato per garantire che l'informazione visualizzata sia accurata.

L'LCD può funzionare in due modalità diverse:

- modalità a 4 bit
- modalità a 8 bit

Nella modalità a 4 bit i dati vengono inviati in gruppi di 4 bit, detti **nibble**: prima viene inoltrato il nibble superiore e successivamente quello inferiore.

I quattro bit inferiori (D0-D3) di un byte costituiscono il nibble inferiore, mentre i quattro bit superiori (D4-D7) di un byte costituiscono il nibble superiore.

Invece, nella modalità a 8 bit è possibile inviare direttamente i dati a 8 bit in un'unica trasmissione, poiché si sfruttano tutte le 8 linee dati.

La modalità a 8 bit è più veloce e produce meno problemi rispetto alla modalità a 4 bit.

Tuttavia, il principale svantaggio è che richiede 8 linee dati collegate al microcontrollore.

Non vengono utilizzati pin di controllo per impostare queste modalità: varia esclusivamente il linguaggio di programmazione utilizzato per impostarli.

Nel progetto in esame viene sfruttata la comunicazione I2C collegando i pin SDA ed SCL del display direttamente ai pin SDA1 ed SCL1 del target.

La modalità scelta è stata la prima, ovvero la modalità a 4 bit.

#### ANALISI DEI COMPONENTI BCM2711 (BSC, TIMER, GPIO)

La scheda Raspberry Pi 4 Model B è dotata di una serie di periferiche integrate, rese possibili dall'integrazione del SoC **BCM2711**:

- Timer
- Controllore di interruzioni
- GPIO (Porte di In/Out Generiche)
- USB
- PCM/I2S (Modulazione a impulsi codificata / Interfaccia seriale audio)
- Controllore DMA (Accesso diretto alla memoria)
- Master I2C (Comunicazione seriale)
- Master SPI (Comunicazione seriale sincrona)
- PWM (Modulazione di larghezza d'impulso)
- UART (Trasmissione seriale asincrona)

Nel contesto del progetto è doveroso menzionare e descrivere le seguenti componenti per comprenderne maggiormente il funzionamento:

- **BSC (Broadcom Serial Control):** il BSC, acronimo di "**Broadcom Serial Control**", rappresenta un sottosistema essenziale nel Raspberry Pi 4 Model B che gestisce protocolli di comunicazione seriale come I2C (Inter-Integrated Circuit) e SPI (Serial Peripheral Interface);
- **Timer:** il Timer interno del Raspberry Pi 4 Model B è un componente temporale di rilevanza critica che offre la capacità di misurare intervalli di tempo, generare segnali periodici e sincronizzare operazioni; viene utilizzato in una vasta gamma di applicazioni, come la gestione di task programmati e il sincronismo di eventi all'interno del sistema;
- **GPIO (General Purpose Input/Output):** i pin GPIO, noti anche come "General Purpose Input/Output", costituiscono l'interfaccia tramite cui il Raspberry Pi 4 Model B può comunicare con il mondo esterno; questi pin possono essere configurati sia come ingressi che come uscite, consentendo al dispositivo di leggere segnali da sensori esterni o di controllare dispositivi attuatori;

#### ANALISI DEGLI INDIRIZZI (BSC, TIMER, GPIO)

La conoscenza degli indirizzi base relativi alle periferiche integrate all'interno del Raspberry Pi 4 Model B risulta di fondamentale importanza: sommando a questi ultimi uno specifico offset è possibile raggiungere i vari registri ad esse associati.

Modificando il contenuto di tali registri è possibile manipolarne la configurazione per controllare il comportamento delle relative periferiche, permettendone al contempo il corretto funzionamento.

## BSC

Il controller BSC dispone di otto registri mappati in memoria. Si presume che tutti gli accessi siano a 32 bit. Si noti che i master BSC2 e BSC7 sono dedicati all'uso delle interfacce HDMI e non dovrebbero essere accessibili dai programmi utente.

Ci sono otto master BSC all'interno di BCM2711. Gli indirizzi dei registri accessibili dall'utente iniziano da:

- BSC0: **0x7e205000**
- BSC1: **0x7e804000**
- BSC3: **0x7e205600**
- BSC4: **0x7e205800**
- BSC5: **0x7e205a80**
- BSC6: **0x7e205c00**

La tabella sottostante mostra gli indirizzi dei registri utilizzati, dove l'indirizzo rappresenta un offset da uno degli indirizzi di base elencati (BSC1) in precedenza:

OFFSET	NOME	DESCRIZIONE
0x08	DLEN	Data Length
0x0c	A	Slave Address
0x10	FIFO	Data FIFO

*Registro DLEN*

Il registro **DLEN** definisce il numero di byte di dati da trasmettere o ricevere nel trasferimento I2C. La lettura del registro restituisce il numero di byte rimanenti nel trasferimento corrente. Il campo DLEN specifica il numero di byte da trasmettere/ricevere.

La lettura del campo DLEN durante un trasferimento in corso (TA = 1) restituisce il numero di byte ancora da trasmettere o ricevere.

La lettura del campo DLEN quando il trasferimento è appena stato completato (DONE = 1) restituisce zero poiché non ci sono più byte da trasmettere o ricevere.

Infine, la lettura del campo DLEN quando TA = 0 e DONE = 0 restituisce l'ultimo valore scritto. Il campo DLEN può essere lasciato invariato durante trasferimenti multipli.

*Registro A*

Il registro **Slave Address** specifica l'indirizzo dello slave e il tipo di ciclo impiegati.

Il registro dell'indirizzo può essere lasciato invariato durante trasferimenti multipli.

Il campo ADDR specifica l'indirizzo slave del dispositivo I2C pilotato.

*FIFO*

Il registro **FIFO** viene utilizzato per accedere al FIFO. I cicli di scrittura a questo indirizzo inseriscono dati nel FIFO da 16 byte, pronti per la trasmissione sul bus BSC. I cicli di lettura accedono ai dati ricevuti dal bus.

Le scritture di dati in un FIFO pieno verranno ignorate e le letture di dati da un FIFO vuoto comporteranno dati non validi. Il FIFO può essere svuotato utilizzando il campo I2CC.CLEAR. Il campo DATA specifica i dati da trasmettere o ricevere.

## TIMER

L'indirizzo base fisico (hardware) per i timer di sistema è **0x7e003000**.

La tabella sottostante mostra gli indirizzi dei registri utilizzati, dove l'indirizzo rappresenta un offset dall'indirizzo di base elencato in precedenza:

OFFSET	NOME	DESCRIZIONE
0x04	CLO	System Timer Counter Lower 32 bits

*Descrizione del Registro CLO*

Il registro **CLO** del timer di sistema è un registro di sola lettura che restituisce il valore attuale dei 32 bit inferiori del contatore a corsa libera.

## GPIO

L'indirizzo di base del registro GPIO è **0x7e200000**.

La tabella sottostante mostra gli indirizzi dei registri utilizzati, dove l'indirizzo rappresenta un offset dall'indirizzo di base elencato in precedenza:

OFFSET	NOME	DESCRIZIONE
0x00	GPSEL0	GPIO Function Select 0
0x1C	GPSET0	GPIO Pin Output Set 0
0x34	GPLEV0	GPIO Pin Level 0

*Descrizione del Registro GPSET0*

I registri **GPSETN** vengono utilizzati per impostare la funzionalità di pin GPIO.

Il campo SETn definisce il pin GPIO corrispondente da impostare.

Se il pin GPIO viene utilizzato come ingresso (per impostazione predefinita), il valore nel campo SETn viene ignorato. Tuttavia, se il pin viene successivamente definito come uscita, il bit verrà impostato in base all'ultima operazione di impostazione/cancellazione.

La separazione delle funzioni di impostazione e cancellazione elimina la necessità di operazioni di lettura-modifica-scrittura.

*Descrizione del Registro GPCLR0*

I registri **GCLR** vengono utilizzati per cancellare le funzionalità di un pin GPIO.

Il campo CLRn definisce il pin GPIO corrispondente da cancellare.

Se il pin GPIO viene utilizzato come ingresso (per impostazione predefinita), il valore nel campo CLRn viene ignorato. Tuttavia, se il pin viene successivamente definito come uscita, il bit verrà impostato in base all'ultima operazione di impostazione/cancellazione. La separazione delle funzioni di impostazione e cancellazione elimina la necessità di operazioni di lettura-modifica-scrittura.

*Descrizione del Registro GPLEV0*

I **GPLEV** restituiscono il valore effettivo del pin.

Il campo LEVn restituisce il valore del pin GPIO corrispondente.

## CALIBRAZIONE DEL SENSORE DHT11

A causa delle limitazioni economiche del sensore DHT11 e delle misurazioni altamente inaccurate prodotte inizialmente, si è deciso di procedere con una **calibrazione** del sensore al fine di ottenere misurazioni più precise in output.

La procedura di **calibrazione** può essere così descritta:

1. Vengono raccolte 30 letture, le quali raccolgono ognuna la misurazione della temperatura e dell'umidità;
2. Dall'insieme di queste letture, sono state scartate tutte le letture di temperatura al di fuori del range compreso tra 0°C e 50°C, così come tutte le letture di umidità percentuale al di fuori del range compreso tra 0% e 100%;
3. Le letture filtrate vengono quindi sommate incrementalmente per categoria attraverso l'utilizzo di una relativa variabile d'accumulo;
4. Il valore risultante di ciascuna variabile viene successivamente estratto e diviso per il numero di letture valide, al fine di ottenere il valore medio delle misurazioni ottenute. Questo valore medio viene quindi conservato in memoria e visualizzato in output come risultato calibrato del sensore;

Il sistema, inoltre, stampa a video nel terminale una serie di informazioni riguardo ciascuna misurazione:

1. Numero di misurazione effettuata;
2. Misurazione effettuata / misurazione non effettuata;
3. Temperatura rilevata valida (entro il range);
4. Temperatura rilevata non valida (fuori dal range);
5. Umidità rilevata valida (entro il range);
6. Umidità rilevata non valida (fuori dal range);

Al termine delle 30 misurazioni, il sistema stampa a video nel terminale le seguenti informazioni:

1. Somma dei valori di temperatura rilevati;
2. Somma dei valori di umidità rilevati;
3. Numero di misurazioni di temperatura valide;
4. Numero di misurazioni di umidità valide;
5. Valore di temperatura misurato;
6. Valore di umidità percentuale misurato;

Quanto detto in precedenza permetterà all'utente finale di verificare facilmente che le stime, i calcoli e i risultati prodotti siano coerenti con le rilevazioni e le operazioni effettuate.

# Flusso degli Eventi

## FLUSSO DEGLI EVENTI

Descritti gli aspetti tecnici del sistema adesso è possibile descrivere il **Flusso degli Eventi** che definiscono tutte le fasi relative al funzionamento del Sistema:

1. **Configurazione del Sistema:** il sistema predispone la configurazione del Raspberry Pi attraverso le GPIO e la comunicazione **I2C** con il display LCD;
2. **Check Status:** viene effettuato un controllo per verificare che la configurazione di sistema sia stata effettuata correttamente;
3. **Inizializzazione del DHT11:** il sensore DHT11 viene inizializzato per effettuare le successive misurazioni;
4. **Calibrazione del DHT11:** si prosegue con la calibrazione del sensore DHT11, precedentemente descritta;
5. **Misurazione calibrata:** viene effettuata la misurazione calibrata relativa alla temperatura e all'umidità dell'ambiente circostante;
6. **Conversione:** il valore relativo alla temperatura precedentemente ottenuto viene memorizzato in una seconda variabile e convertito da gradi Celsius a gradi Kelvin;
7. **Visualizzazione dei dati rilevati:** i valori relativi a temperatura ed umidità vengono successivamente mostrati a schermo sul display LCD e stampati a video nel terminale;
8. **Accensione LED:** a seconda del range che il valore di temperatura rispetta, si assiste all'accensione del relativo LED presente sulla Breadboard tramite i pin GPIO;
9. **Switch della modalità:** ogni 15 secondi la modalità del sistema commuta ciclicamente da “Visualizzazione Gradi Celsius” a “Visualizzazione Gradi Kelvin” a “Clear LCD”;

Segue la tabella delle azioni specifiche di ciascun attuatore (in questo caso dei 3 LED):

<b>RILEVAZIONE DI TEMPERATURA</b>	<b>RANGE (°C)</b>			
	$0 \leq T < 19$	$19 \leq T < 29$	$29 \leq T < 50$	$0 > T \& T > 50$
<b>ATTUATORE</b>	<b>STATO</b>			
LED BLU	ON	OFF	OFF	OFF
LED VERDE	OFF	ON	OFF	OFF
LED ROSSO	OFF	OFF	ON	OFF

## MODALITA' OPERATIVE

Il Sistema presenta le seguenti modalità:

1. **Visualizzazione Gradi Celsius:** vengono mostrati sul display il valore di temperatura in gradi Celsius e il valore percentuale di umidità;
2. **Visualizzazione Gradi Kelvin:** vengono mostrati sul display il valore di temperatura in gradi Kelvin e il valore percentuale di umidità;
3. **Clear LCD:** il display LCD viene ripulito da tutti i dati precedentemente mostrati;

# Analisi del codice

## JONESFORTH.F

Il codice contiene definizioni di word aggiuntive che non sono state definite nativamente nell'ambiente pijFORTHos e che risultano utili ai fini del funzionamento del sistema.

## UTILS.F

Il codice contiene alcune definizioni di funzionalità utili che implementano conversioni ed istruzioni matematiche.

## GPIO.F (GESTIONE DEI PIN GPIO)

Il codice definisce una serie di procedure che consentono di interagire con i pin GPIO in vari modi.

All'inizio, vengono definite costanti che rappresentano gli indirizzi di base e gli offset dei registri GPIO sul Target. Questi indirizzi vengono utilizzati per accedere ai registri che controllano il comportamento dei pin GPIO.

Ad esempio, la procedura “**CHECK\_GPIO**” verifica se il numero di GPIO specificato è valido, ovvero se rientra nell'intervallo accettabile.

In caso contrario, viene generato un messaggio di errore.

Le procedure come “**SET\_INPUT**”, “**SET\_OUTPUT**” e “**SET\_ALTF0**” sono responsabili della configurazione delle funzionalità dei pin GPIO.

Queste procedure utilizzano i registri corrispondenti per impostare il comportamento del pin come input, output oppure assegnargli la rispettiva funzione alternativa.

La procedura “**GET\_INPUT**” permette di leggere lo stato di un pin GPIO specifico, restituendo un valore booleano che indica se il pin è attivo o meno.

Le procedure “**GPIO\_ON**” e “**GPIO\_OFF**” permettono di attivare o disattivare un pin GPIO specifico, impostando i bit corrispondenti nei registri appropriati. Questo consente di controllare lo stato dei pin in modo efficiente.

Inoltre, il codice contiene procedure per calcolare maschere di bit utilizzati nelle operazioni di controllo e configurazione dei pin GPIO.

## CONNECTIONS.F

Il codice contiene la definizione delle connessioni tra il Target e i vari devices e le rispettive funzioni di setup.

## TIME.F

Il codice contiene la definizione del Timer di sistema utilizzato per effettuare operazioni di temporizzazione.

## DHT11.F (ACQUISIZIONE E MANIPOLAZIONE DEI DATI TRAMITE DHT11)

Il codice è stato sviluppato per effettuare misurazioni di temperatura e umidità tramite il sensore DHT11, elaborare i dati raccolti e calcolare medie di questi valori.

Inoltre, il programma stampa i risultati delle misurazioni e delle elaborazioni sulla console.

Per realizzare queste funzionalità, il codice utilizza variabili per immagazzinare dati quali i bit letti dal sensore, gli indici correnti di temperatura e umidità, le medie temporanee di temperatura e umidità, oltre a contatori e costanti che definiscono il numero di misurazioni e cicli necessari.

Le procedure nel codice svolgono ruoli specifici. Alcune si occupano dell'interazione con il sensore DHT11: “**DHT\_SIGNAL**” instaura la comunicazione con il sensore impostando lo start signal, “**BIT\_0**” e “**BIT\_1**” leggono i bit trasmessi dal sensore.

La procedura “**READ\_DHT**” decodifica i dati trasmessi, e le procedure “**READ\_TEMP\_T**” e “**READ\_HUMIDITY\_T**” estrapolano i valori di temperatura e umidità temporanei per ciascuna iterazione compiuta.

Le fasi di filtraggio vengono gestite da “**HUM\_FILTERING**” e “**TEMP\_FILTERING**”, che verificano se i dati rientrano nei range previsti.

Le procedure “**CALCULATE\_AVERAGE\_HUMIDITY**” e “**CALCULATE\_AVERAGE**” calcolano le medie definitive per umidità e temperatura, memorizzando i risultati ottenuti.

L'esecuzione del programma segue la procedura “**START\_MEASUREMENT**”, che inizializza le variabili, esegue le misurazioni attraverso diversi cicli, calcola le medie dei valori di umidità e temperatura.

## LCD.F (VISUALIZZAZIONE DEI DATI SUL DISPLAY LCD)

Il codice permette di controllare un display LCD tramite comunicazione I2C.

Inizialmente, vengono definite alcune costanti che identificano gli indirizzi di base e gli offset dei registri BSC. Questi indirizzi saranno utilizzati per accedere ai registri necessari per gestire l'interfaccia I2C.

Successivamente, il codice definisce diverse procedure, che svolgono funzioni di utilità per la comunicazione con il display LCD.

Ad esempio, la procedura “**IS\_COMMAND**” verifica se un valore rappresenta un comando I2C.

La procedura “**BSC1\_STORE**” memorizza un valore nel registro FIFO dell'interfaccia BSC, che è utilizzato per inviare dati al display LCD.

“**BSC1\_1BYTE**” imposta la lunghezza dei dati da inviare a 1 byte.

La procedura “**I2C\_ENABLE**” abilita l'interfaccia I2C impostando un valore specifico nel registro di controllo. La procedura “**I2C\_BUS**” coordina l'invio di dati al display LCD tramite l'interfaccia I2C, utilizzando le procedure precedenti per memorizzare dati, impostare lunghezze di bit e abilitare l'interfaccia.

La procedura “**SEND\_NIBBLE**” è responsabile dell'invio di un nibble (4 bit) di dati al display, considerando anche i segnali di controllo necessari per l'interfaccia I2C.

La procedura “**PRINT\_CHAR**” invia un carattere al display LCD, coordinando l'invio dei suoi nibble superiore e inferiore tramite la procedura precedente.

Per gestire operazioni specifiche, vengono definite ulteriori procedure: “**LCD\_CLEAR**” cancella il contenuto del display inviando un comando specifico, “**PRINT\_STRING**” stampa una stringa sul display, inizializzando prima il display e poi inviando i caratteri della stringa.

Vi sono anche procedure per stampare i valori di umidità (“**PRINT\_HUMIDITY**”) e temperatura in gradi Celsius (“**PRINT\_TEMPERATURE\_CELSIUS**”) e Kelvin (“**PRINT\_TEMPERATURE\_KELVIN**”).

Ogni procedura converte il valore numerico in rappresentazione ASCII e lo invia al display.

Infine, la procedura “**LCD\_INIT**” configura il display LCD impostando l’indirizzo del dispositivo e inviando un comando di inizializzazione.

#### SYSTEM.F (GESTIONE DELLE FUNZIONALITA’)

Il codice gestisce ciclicamente le modalità del sistema: **Clear LCD, Visualizzazione Gradi Celsius e Visualizzazione Gradi Kelvin**. Questo gestisce le misurazioni, la visualizzazione sul display LCD e il controllo dei LED in base alla temperatura rilevata.

Lo fa definendo una variabile chiamata “**MODE**”, che rappresenta la modalità di funzionamento del sistema. I valori 0, 1 e 2 sono utilizzati rispettivamente per rappresentare le modalità precedentemente descritte.

Le procedure definite sono le seguenti:

- “**SWITCH\_MODE**”: cambia la modalità di funzionamento del sistema tra le diverse opzioni ciclicamente;
- “**BOOT**”: avvia il sistema impostando la modalità su 0 (Clear LCD), configurando i LED e il display LCD, e stampando un messaggio di completamento della configurazione;
- “**CELSIUS\_TEMPERATURE**”: esegue una misurazione e visualizza i valori di umidità e temperatura sul display in gradi Celsius;
- “**KELVIN\_TEMPERATURE**”: esegue una misurazione e visualizza i valori di umidità e temperatura sul display in gradi Kelvin;
- “**ALL\_LED\_ON**”: attiva tutti i LED;
- “**ALL\_LED\_OFF**”: disattiva tutti i LED;
- “**LED\_ON**”: attiva il LED corrispondente in base alla temperatura rilevata. Vengono definiti intervalli di temperatura per accendere i LED con diversi colori;
- “**CHECK\_MODE**”: verifica se la modalità è impostata su 0 prima di entrare nel ciclo principale; in caso contrario, segnala un problema;
- “**INIT**”: rappresenta il ciclo principale del programma. Attende un certo periodo di tempo e quindi controlla la modalità di funzionamento; se è in modalità “Celsius” o “Kelvin”, esegue le misurazioni appropriate e attiva i LED in base alla temperatura rilevata, mentre in caso contrario, pulisce il display LCD;

#### MAIN.F

Il codice contiene le operazioni di inizializzazione ed avviamento del sistema.

# Codice

JONESFORTH.F

```

: '\n' 10 ;
: BL 32 ;
: ':' [ CHAR : ] LITERAL ;
: ';' [ CHAR ; ] LITERAL ;
: '(' [ CHAR ( ] LITERAL ;
: ')' [ CHAR ) ] LITERAL ;
: """ [ CHAR " ] LITERAL ;
: 'A' [ CHAR A ] LITERAL ;
: '0' [ CHAR 0 ] LITERAL ;
: '-' [ CHAR - ] LITERAL ;
: '.' [ CHAR . ] LITERAL ;
: ( IMMEDIATE 1 BEGIN KEY DUP '(' = IF DROP 1+ ELSE ')' = IF 1- THEN THEN DUP 0=
UNTIL DROP ;
: SPACES ( n -- ) BEGIN DUP 0> WHILE SPACE 1- REPEAT DROP ;
: WITHIN -ROT OVER <= IF > IF TRUE ELSE FALSE THEN ELSE 2DROP FALSE THEN ;
: ALIGNED ( c-addr -- a-addr ) 3 + 3 INVERT AND ;
: ALIGN HERE @ ALIGNED HERE ! ;
: C, HERE @ C! 1 HERE +! ;
: S" IMMEDIATE ( -- addr len )
    STATE @ IF
        ' LITS , HERE @ 0 ,
        BEGIN KEY DUP """
        <> WHILE C, REPEAT
            DROP DUP HERE @ SWAP - 4- SWAP ! ALIGN
    ELSE
        HERE @
        BEGIN KEY DUP """
        <> WHILE OVER C! 1+ REPEAT
            DROP HERE @ - HERE @ SWAP
    THEN
;
: ." IMMEDIATE ( -- )
    STATE @ IF

```

```

[COMPILE] S" ' TELL ,
ELSE
    BEGIN KEY DUP """ = IF DROP EXIT THEN EMIT AGAIN
THEN
;
:DICT WORD FIND ;
:VALUE ( n -- ) WORD CREATE DOCOL , ' LIT , , ' EXIT , ;
:TO IMMEDIATE ( n -- )
    DICT >DFA 4+
        STATE @ IF ' LIT , , ' ! , ELSE ! THEN
;
:+TO IMMEDIATE
    DICT >DFA 4+
        STATE @ IF ' LIT , , ' +! , ELSE +! THEN
;
:ID. 4+ COUNT F_LENMASK AND BEGIN DUP 0> WHILE SWAP COUNT EMIT SWAP 1-
REPEAT 2DROP ;
:?HIDDEN 4+ C@ F_HIDDEN AND ;
:?IMMEDIATE 4+ C@ F_IMMED AND ;
:WORDS LATEST @ BEGIN ?DUP WHILE DUP ?HIDDEN NOT IF DUP ID. SPACE THEN @
REPEAT CR ;
:FORGET DICT DUP @ LATEST ! HERE !
:CFA> LATEST @ BEGIN ?DUP WHILE 2DUP SWAP < IF NIP EXIT THEN @ REPEAT DROP 0 ;
:SEE
    DICT HERE @ LATEST @
    BEGIN 2 PICK OVER <> WHILE NIP DUP @ REPEAT
        DROP SWAP ':' EMIT SPACE DUP ID. SPACE
        DUP ?IMMEDIATE IF ." IMMEDIATE " THEN
        >DFA BEGIN 2DUP
        > WHILE DUP @ CASE
            ' LIT OF 4 + DUP @ . ENDOF
            ' LITS OF [ CHAR S ] LITERAL EMIT """ EMIT SPACE
                4 + DUP @ SWAP 4 + SWAP 2DUP TELL """ EMIT SPACE + ALIGNED 4 -
            ENDOF
            ' OBRANCH OF ." OBRANCH ( " 4 + DUP @ .." ) " ENDOF
            ' BRANCH OF ." BRANCH ( " 4 + DUP @ .." ) " ENDOF
            ' ' OF [ CHAR ' ] LITERAL EMIT SPACE 4 + DUP @ CFA> ID. SPACE ENDOF

```

```

' EXIT OF 2DUP 4 + <> IF ." EXIT " THEN ENDOF
DUP CFA> ID. SPACE
ENDCASE 4 + REPEAT
';' EMIT CR 2DROP
;

: :NONAME 0 0 CREATE HERE @ DOCOL , ] ;
: ['] IMMEDIATE ' LIT , ;
: EXCEPTION-MARKER RDROP 0 ;
: CATCH ( xt -- exn? ) DSP@ 4+ >R ' EXCEPTION-MARKER 4+ >R EXECUTE ;
: THROW ( n -- ) ?DUP IF
    RSP@ BEGIN DUP R0 4-
    < WHILE DUP @ ' EXCEPTION-MARKER 4+
        = IF 4+ RSP! DUP DUP DUP R> 4- SWAP OVER ! DSP! EXIT THEN
        4+ REPEAT DROP
    CASE
        0 1- OF ." ABORTED" CR ENDOF
        ." UNCAUGHT THROW " DUP . CR
    ENDCASE QUIT THEN
;
: ABORT ( -- ) 0 1- THROW ;
: PRINT-STACK-TRACE
    RSP@ BEGIN DUP R0 4-
    < WHILE DUP @ CASE
        ' EXCEPTION-MARKER 4+ OF ." CATCH ( DSP=" 4+ DUP @ U. ." ) " ENDOF
        DUP CFA> ?DUP IF 2DUP ID. [ CHAR + ] LITERAL EMIT SWAP >DFA 4+ - .
THEN
    ENDCASE 4+ REPEAT DROP CR
;
: BINARY ( -- ) 2 BASE ! ;
: OCTAL ( -- ) 8 BASE ! ;
: 2# BASE @ 2 BASE ! WORD NUMBER DROP SWAP BASE ! ;
: 8# BASE @ 8 BASE ! WORD NUMBER DROP SWAP BASE ! ;
: # ( b -- n ) BASE @ SWAP BASE ! WORD NUMBER DROP SWAP BASE ! ;
: UNUSED ( -- n ) PAD HERE @ - 4/ ;
: WELCOME
    S" TEST-MODE" FIND NOT IF
        ." JONESFORTH VERSION " VERSION . CR

```

```

    UNUSED . ." CELLS REMAINING" CR
    ." OK "
THEN
;
WELCOME
HIDE WELCOME

UTILS.F

DECIMAL
\ From microseconds to milliseconds
: TO_MS ( time_us -- time_ms ) 1000 *;
\ From Celsius to Kelvin
: TO_KELVIN ( temp_celsius -- temp_kelvin ) 273 +;
\ Converts a char to his ASCII encoding
: TO_ASCII_2 ( var -- d_ascii u_ascii ) DUP 10 MOD 48 + SWAP 10 / 48 + SWAP ;
: TO_ASCII_3 ( var -- ascii ) DUP DUP 100 / 48 + ROT 10 / 10 MOD 48 + ROT 10 MOD 48 +;
: TO_ASCII @ DUP 100 / IF TO_ASCII_3 ELSE TO_ASCII_2 THEN ;
\ Computes the absolute value of the given number
: ABS ( n -- |n| ) DUP 0 < IF -1 * THEN ;

GPIO.F

HEX
\ RPPI4 GPIO base address
FE200000 CONSTANT RPI4_BASE_GPIO
\ Reach GPIO registers with an offset
RPI4_BASE_GPIO 1C + CONSTANT GPSET0
RPI4_BASE_GPIO 28 + CONSTANT GPCLR0
RPI4_BASE_GPIO 34 + CONSTANT GPLEV0
\ Checks if the GPIO is valid
: CHECK_GPIO ( n_gpio -- boolean ) DUP 0>= SWAP 39 <= AND ;
\ Drops invalid GPIOs
: INVALID_GPIO ( n_gpio -- ) DROP ." GPIO is not valid." CR ;
\ Finds Registers
: RPI4_BASE_GPIO_REGISTER ( n_gpio -- rpi4_base_register_addr ) DUP CHECK_GPIO

```

```

IF
  A / 4 * RPI4_BASE_GPIO +
ELSE
  INVALID_GPIO
THEN ;
: GPSET0_REGISTER ( n_gpio -- gpset0_register_addr ) DUP CHECK_GPIO
IF
  20 / 4 * GPSET0 +
ELSE
  INVALID_GPIO
THEN ;
: GPCLR0_REGISTER ( n_gpio -- gpclr0_register_addr ) DUP CHECK_GPIO
IF
  20 / 4 * GPCLR0 +
ELSE
  INVALID_GPIO
THEN ;
: GPLEV0_REGISTER ( n_gpio -- gplev0_register_addr ) DUP CHECK_GPIO
IF
  20 / 4 * GPLEV0 +
ELSE
  INVALID_GPIO
THEN ;
\ Calculates the mask for bit clearing
: MASK_CLEAN ( n_gpio -- mask_fsel ) A MOD 3 * 7 SWAP LSHIFT ;
\ Bit clear function
: BIT_CLEAR ( register mask -- cleaned_register ) INVERT AND ;
\ Sets GPIO input
: SET_INPUT ( n_gpio -- ) DUP RPI4_BASE_GPIO_REGISTER DUP @ ROT MASK_CLEAN BIT_CLEAR
SWAP !;
\ Sets GPIO output
: SET_OUTPUT ( n_gpio -- ) DUP DUP RPI4_BASE_GPIO_REGISTER DUP @ ROT MASK_CLEAN
BIT_CLEAR ROT A MOD 3 * 1 SWAP LSHIFT OR SWAP !;
\ Sets GPIO with its alternative function

```

```

: SET_ALTF0 ( n_gpio -- ) DUP DUP RPI4_BASE_GPIO_REGISTER DUP @ ROT MASK_CLEAN
BIT_CLEAR ROT A MOD 3 * 4 SWAP LSHIFT OR SWAP !;
\ Returns input from GPIO
: GET_INPUT ( n_gpio -- boolean )
    DUP GPLEV0_REGISTER @ SWAP 20 MOD 1 SWAP LSHIFT AND
    IF
        1
    ELSE
        0
    THEN ;
\ Sets GPIO bit to 1 in the GPSET register
: GPIO_ON ( n_gpio -- ) DUP GPSET0_REGISTER SWAP 20 MOD 1 SWAP LSHIFT SWAP !;
\ Sets GPIO bit to 1 in the GPCLR register
: GPIO_OFF ( n_gpio -- ) DUP GPCLR0_REGISTER SWAP 20 MOD 1 SWAP LSHIFT SWAP !;

```

## CONNECTIONS.F

## DECIMAL

```

\ GPIO connections
: LED_GREEN ( -- n_gpio ) 27 ;
: LED_RED ( -- n_gpio ) 22 ;
: LED_BLUE ( -- n_gpio ) 17 ;
: DHT11 ( -- n_gpio ) 13 ;
: SCL ( -- n_gpio ) 3 ;
: SDA ( -- n_gpio ) 2 ;
\ Setup for LED
: SETUP_LED_BLUE ( -- ) LED_BLUE SET_OUTPUT ;
: SETUP_LED_GREEN ( -- ) LED_GREEN SET_OUTPUT ;
: SETUP_LED_RED ( -- ) LED_RED SET_OUTPUT ;
: SETUP_LEDS SETUP_LED_BLUE SETUP_LED_GREEN SETUP_LED_RED ;
\ Sets the alternative function for display related GPIOs
: SETUP_DISPLAY SDA SET_ALTF0 SCL SET_ALTF0 ;

```

TIME.F

HEX

\ RPPI4 Timer base address

FE003000 CONSTANT RPI4\_BASE\_TIMER

\ Sets Timer registers with an offset

RPI4\_BASE\_TIMER 4 + CONSTANT CLO

\ Delays for the time given as input (in microseconds)

: DELAY ( us -- ) CLO @ BEGIN 2DUP CLO @ - ABS SWAP &gt; UNTIL 2DROP ;

DHT.F

DECIMAL

\ Contains the data received from the humidity-temperature sensor

VARIABLE DHT\_BITS

\ Contains the current temperature

VARIABLE CURRENT\_INDEX

\ Contains the current humidity index

VARIABLE CURRENT\_INDEX\_HUM

\ Contains the temperature average

VARIABLE MEDIA

\ Contains the humidity average

VARIABLE MEDIA\_HUMIDITY

\ Counter

VARIABLE COUNTER

\ Number of measurements

30 CONSTANT NUM\_MEASUREMENTS

\ Number DHT cicles

31 CONSTANT DHT\_CICLE

\ Contains each temperature value

VARIABLE CUMULATOR

\ Contains each humidity value

VARIABLE CUMULATOR\_HUMIDITY

\ Contains the integer value of temperature (in Celsius)

VARIABLE HUMIDITY\_INT

\ Contain temporaneus humidity values

```

VARIABLE HUMIDITY_T
\ Contains the integer value of temperature (in Celsius)
VARIABLE TEMP_INT
\ Contains temporaneous temperature values
VARIABLE TEMP_T
\ Contains the integer value of temperature (in Kelvin)
VARIABLE TEMP_INT_KELVIN
\ DHT11 Start Signal
:DHT_SIGNAL ( -- ) DHT11 SET_OUTPUT DHT11 GPIO_OFF 18000 DELAY DHT11 GPIO_ON
DHT11 SET_INPUT ;
:BIT_0 ( -- ) BEGIN DHT11 GET_INPUT 0 = WHILE REPEAT ;
:BIT_1 ( -- ) BEGIN DHT11 GET_INPUT 1 = WHILE REPEAT ;
\ Verify if the bit is 1 or 0
:READ_DHT ( -- )
    BIT_0 BIT_1
    DHT_CICLE BEGIN
        DHT_BITS DUP @ 1 LSHIFT SWAP ! BIT_0 CLO @ BIT_1 CLO @ SWAP - 50 >
        IF
            DHT_BITS DUP @ 1 OR SWAP !
        THEN
        1 - DUP 0 >
        WHILE REPEAT DROP ;
\ Do the measurements for temperature
:READ_TEMP_T ( -- ) DHT_BITS @ 8 RSHIFT 255 AND ;
\ Do the measurements for humidity
:READ_HUMIDITY_T ( -- ) DHT_BITS @ 24 RSHIFT ;
\ Humidity Filtering
:HUM_FILTERING ( -- )
    DUP DUP 0 > SWAP 100 < AND \verify if humidity is in range
    IF
        HUMIDITY_T !
        HUMIDITY_T @ CUMULATOR_HUMIDITY +!
        1 CURRENT_INDEX_HUM +!
        ." Umidità rilevata: " HUMIDITY_T @ .. "% " CR

```

```

ELSE
DROP
." Umidità fuori dal range previsto " CR
THEN ;
\ Temperature Filtering
: TEMP_FILTERING ( -- )
DUP DUP 0 > SWAP 50 < AND \ Verify if temperature is in range
IF
TEMP_T !
TEMP_T @ CUMULATOR +
1 CURRENT_INDEX +
." Temperatura rilevata: " TEMP_T @ .. °C " CR
ELSE
DROP
." Temperatura fuori dal range previsto " CR
THEN ;
\ Calculate humidity average
: CALCULATE_AVERAGE_HUMIDITY ( -- )
CUMULATOR_HUMIDITY @ 0 >
IF
CUMULATOR_HUMIDITY @ CURRENT_INDEX_HUM @ / MEDIA_HUMIDITY !
ELSE
0 MEDIA_HUMIDITY !
THEN
;
\ Calculate temperature average
: CALCULATE_AVERAGE ( -- )
CUMULATOR @ 0 >
IF
CUMULATOR @ CURRENT_INDEX @ / MEDIA !
ELSE
0 MEDIA !
THEN
;

```

```

\ Start measurements
: MEASURE ( -- )
  30000 DELAY
  DHT_SIGNAL READ_DHT
  ." Misurazione Eseguita " CR
  READ_TEMP_T
  TEMP_FILTERING
  READ_HUMIDITY_T
  HUM_FILTERING
;

\ Start cycling
: CICLE ( -- )
  BEGIN
    ." Misurazione N° " COUNTER @ . CR
    MEASURE
    1 COUNTER +!
    COUNTER @ NUM_MEASUREMENTS <
    WHILE
      TRUE
      REPEAT DROP ;
\ Do all the measurements
: MEASURE_ALL ( -- )
  CICLE
  CALCULATE_AVERAGE
  CALCULATE_AVERAGE_HUMIDITY
  ." Somma delle temperature rilevate: " CUMULATOR @ . CR
  ." Somma delle umidità rilevate: " CUMULATOR_HUMIDITY @ . CR
  ." Numero di misurazioni di temperatura valide: " CURRENT_INDEX @ . CR
  ." Numero di misurazioni di umidità valide: " CURRENT_INDEX_HUM @ . CR
  MEDIA @ TEMP_INT !
  MEDIA_HUMIDITY @ HUMIDITY_INT !
;

\ Print Humidity in the console
: PRINT_HUM ( -- ) ." Humidity: " HUMIDITY_INT ? ." % " CR ;

```

```

\ Print Temperature in the console
: PRINT_TEMPERATURE ( -- ) ." Temp: " TEMP_INT ? ." °C" CR ;
\ Initialize variables
: INITIALIZE ( -- ) 0 COUNTER ! 0 CURRENT_INDEX ! 0 CURRENT_INDEX_HUM ! 0 CUMULATOR !
0 CUMULATOR_HUMIDITY ! 0 DHT_BITS ! 0 TEMP_INT_KELVIN !;
\ Converts from Celsius to Kelvin
: CONVERT ( -- ) TEMP_INT @ TO_KELVIN TEMP_INT_KELVIN !;
\ Start measuring
: START_MEASUREMENT ( -- ) INITIALIZE MEASURE_ALL CONVERT PRINT_HUM
PRINT_TEMPERATURE ;

```

LCD.F

HEX

```

\ RPPI4 BSC (Broadcom Serial Control) base address
FE804000 CONSTANT RPI4_BASE_BSC1
\ Sets BSC registers with an offset
RPI4_BASE_BSC1 8 + CONSTANT BSC1_DLEN
RPI4_BASE_BSC1 10 + CONSTANT BSC1_FIFO
RPI4_BASE_BSC1 C + CONSTANT BSC1_A
\ Returns true if the value is a command
: IS_COMMAND ( value -- boolean ) 100 AND ;
\ Stores the value in the data BSC1 FIFO register
: BSC1_STORE ( value -- )
    BSC1_FIFO !;
\ Sets as 1 the dimension of the value (in bytes)
: BSC1_1BYTE ( -- )
    1 BSC1_DLEN !;
\ Sets as 0 the "Read Transfer bit" and as 1 the "Start Transfer" bit and the "I2C Enable" bit in
the control register
: I2C_ENABLE ( -- )
    8080 RPI4_BASE_BSC1 !;
\ Starts I2C Bus
: I2C_BUS ( value -- )

```

```

BSC1_STORE
BSC1_1BYTE
I2C_ENABLE ;
\ Sends 4 bits to the display
: 4BIT_CHAR ( is_command, value -- )
DUP ROT
IF
  C +
ELSE
  D +
THEN
I2C_BUS 1 TO_MS DELAY
8 + I2C_BUS 2 TO_MS DELAY ;
\ Prints a character on the display
: PRINT_CHAR ( value -- ) DUP IS_COMMAND SWAP 2DUP F0 AND 4BIT_CHAR F AND 4 LSHIFT
4BIT_CHAR 5 TO_MS DELAY ;
\ Cleans the display
: LCD_CLEAR ( -- )
  101 PRINT_CHAR ;
\ Prints a string on the display
: PRINT_STRING ( a_0, a_1, ..., a_n -- )
  LCD_CLEAR DEPTH DUP
BEGIN
  DUP 0 >
  WHILE
    ROT >R 1 -
    REPEAT
    DROP
  BEGIN
    DUP 0 >
  WHILE
    R> PRINT_CHAR 1 -
    REPEAT
    DROP ;

```

\ Prints the humidity value on the display  
`:PRINT_HUMIDITY ( -- ) 48 75 6D 69 64 69 74 79 3A 20 HUMIDITY_INT TO_ASCII 25 1C0 ;`

\ Prints the temperature value on the display (in Celsius)  
`:PRINT_TEMPERATURE_CELSIUS ( -- ) 54 65 6D 70 3A 20 TEMP_INT TO_ASCII DF 43 ;`

\ Prints the temperature value on the display (in Kelvin)  
`:PRINT_TEMPERATURE_KELVIN ( -- ) 54 65 6D 70 3A 20 TEMP_INT_KELVIN TO_ASCII DF 4B ;`

\ Configures the display by setting the value 3F in the slave address and sending the command 0x02 ("set 4-bit mode")  
`:LCD_INIT ( -- ) 3F BSC1_A ! 102 PRINT_CHAR ;`

## SYSTEM.F

\ System MODE:

\ 0 = OFF

\ 1 = Celsius

\ 2 = Kelvin

## DECIMAL

## VARIABLE MODE

\ Switch mode

`:SWITCH_MODE ( -- ) MODE DUP @ 1 + 3 MOD SWAP ! ;`

\ "Boot" the system

`:BOOT ( -- ) 0 MODE ! SETUP_LEDS SETUP_DISPLAY LCD_INIT ." Setup Completed " CR ;`

\ Makes a measurement and shows the measured values on the display (Celsius)

`:CELSIUS_TEMPERATURE ( -- ) LCD_CLEAR START_MEASUREMENT PRINT_HUMIDITY PRINT_TEMPERATURE_CELSIUS PRINT_STRING ;`

\ Makes a measurement and shows the measured values on the display (Kelvin)

`:KELVIN_TEMPERATURE ( -- ) LCD_CLEAR START_MEASUREMENT PRINT_HUMIDITY PRINT_TEMPERATURE_KELVIN PRINT_STRING ;`

\ Turn all LEDs ON

`:ALL_LED_ON ( -- ) LED_RED GPIO_ON LED_BLUE GPIO_ON LED_GREEN GPIO_ON ;`

\ Turn all LEDs OFF

`:ALL_LED_OFF ( -- ) LED_RED GPIO_OFF LED_BLUE GPIO_OFF LED_GREEN GPIO_OFF ;`

\ Turn the LED ON according to temperature range

`:LED_ON ( -- )`

```

\ Cool Temperature
TEMP_INT @ 0 > TEMP_INT @ 19 < AND
IF
  LED_BLUE GPIO_ON
  LED_GREEN GPIO_OFF
  LED_RED GPIO_OFF
  ." BLUE LED ON! " CR CR
ELSE
  \ Medium Temperature
  TEMP_INT @ 19 >= TEMP_INT @ 29 < AND
  IF
    LED_GREEN GPIO_ON
    LED_BLUE GPIO_OFF
    LED_RED GPIO_OFF
    ." GREEN LED ON! " CR CR
  ELSE
    \ Hot Temperature
    TEMP_INT @ 29 >= TEMP_INT @ 50 < AND
    IF
      LED_RED GPIO_ON
      LED_BLUE GPIO_OFF
      LED_GREEN GPIO_OFF
      ." RED LED ON! " CR CR
    ELSE
      ALL_LED_OFF
      ." LEDs OFF! " CR CR
    THEN
  THEN ;
\ Check if MODE is zero before entering the loop
: CHECK_MODE ( -- )
  MODE @ 0 =
  IF
    ALL_LED_ON ." System OK. Starting INIT LOOP." CR

```

```

ELSE
." System IS NOT OK. Aborting INIT LOOP." CR
THEN ;
\ Makes the system work
: INIT( -- )
BEGIN
15000000 DELAY
IF
SWITCH_MODE
THEN
MODE @ 1 =
IF
." Loading Mesuraments... " CR
CELSIUS_TEMPERATURE
LED_ON
ELSE
MODE @ 2 =
IF
." Loading Mesuraments... " CR
KELVIN_TEMPERATURE
LED_ON
ELSE
." Cleaning LCD Display... " CR CR
LCD_CLEAR
THEN
THEN
DROP
AGAIN;

MAIN.F
: MAIN( -- )
BOOT
CHECK_MODE
INIT;

```

# Setting del Sistema

## SETTING DELL'AMBIENTE DI SVILUPPO

Affinché il sistema funzioni correttamente, è fondamentale seguire una serie di passaggi per preparare l'ambiente di sviluppo sia sul Target che sul PC. Questa preparazione consentirà di realizzare in modo corretto la programmazione interattiva utilizzando il protocollo di comunicazione UART sfruttando il connettore CP2102.

## SETTING DEL RASPBERRY PI

Il Raspberry Pi 4 Model B utilizza una MicroSD come memoria di archiviazione principale, che deve essere formattata correttamente per poter funzionare (formato FAT32).

Per eseguire tale formattazione, è necessario utilizzare il software **Raspberry Pi Imager**, scaricabile gratuitamente dal sito <https://www.raspberrypi.com/software/>.

Di default, questo software installerà il Raspberry Pi OS, il sistema operativo proprietario della piattaforma, sulla MicroSD.

Tra tutti i file presenti nell'installazione, solo alcuni risultano rilevanti per il funzionamento del sistema. Una volta completata la procedura di formattazione e avviato l'accesso alla directory principale della MicroSD, bisogna seguire i prossimi passaggi:

1. Eliminare tutti i file denominati "**kernelN.img**" nella directory principale e sostituirli con l'unico file "**kernel7.img**" di pijFORTHos, fornito su gentile concessione dal Professore Daniele Peri. Questo file "kernel7.img" dovrà essere l'unico kernel presente sulla MicroSD;
2. Successivamente, modificare il file "**config.txt**" eliminando il commento all'inizio della stringa "**dtparam=i2c\_arm=on**" ed aggiungendo la stringa "**enable\_uart=1**" alla fine del file di testo;
3. Assicurarsi di salvare la modifica e chiudere il file;

Questi passaggi garantiranno una corretta preparazione della MicroSD e la configurazione necessaria per l'utilizzo di pijFORTHos sul Target, il quale sarà adesso pronto per essere utilizzato mediante qualsiasi software che permetta la comunicazione tramite protocollo di comunicazione UART.

## SETTING DEL PC

In seguito, è necessario procedere con la configurazione dell'ambiente di sviluppo sul PC destinato alla programmazione del Target attraverso il terminale FORTH.

Tuttavia, a seconda del sistema operativo presente sul PC, sarà necessario utilizzare software diversi. Verrà focalizzata l'attenzione sulla procedura da seguire sulla piattaforma MacOS.

## PROCEDURA DI AVVIO TRAMITE MacOS E PICOCOM-3.1

Collegare adesso il Target al PC utilizzando il connettore UART-USB **CP2102** e digitare successivamente il seguente comando nel terminale per avviare il software **picocom**:

1. Da terminale eseguire: **exec bash**
2. Eseguire: **sudo picocom --b 115200 /dev/tty.usbserial-0001 --send "ascii-xfr -sv -l100 -c10" --imap delbs**

```
Last login: Thu Sep 14 23:06:07 on ttys000
[giuseppecacciafeda@MacBook-Pro-di-Giuseppe-4 ~ % exec bash

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[bash-3.2$ sudo picocom --b 115200 /dev/tty.usbserial-0001 --send "ascii-xfr -sv -l100 -c10" --imap delbs
Password:
[picocom v3.1

port is      : /dev/tty.usbserial-0001
flowcontrol  : none
baudrate is   : 115200
parity is    : none
databits are  : 8
stopbits are  : 1
escape is    : C-a
local echo is: no
noinit is    : no
noreset is   : no
hangup is   : no
nolock is   : no
send_cmd is  : ascii-xfr -sv -l100 -c10
receive_cmd is: rz -vv -E
imap is      : delbs,
omap is      :
emap is      : crcrlf,logfile is none
initstring is: none
exit_after is: not set
exit is      : no

Type [C-a] [C-h] to see available commands
Terminal ready
```

Una volta che la schermata contenente tutti i dettagli dei parametri di comunicazione è visualizzata, si provvederà ad alimentare il dispositivo Target. Fatto ciò si sarà pronti per utilizzare il terminale FORTH e per avviare l'interazione col device:

```
Type [C-a] [C-h] to see available commands
Terminal ready

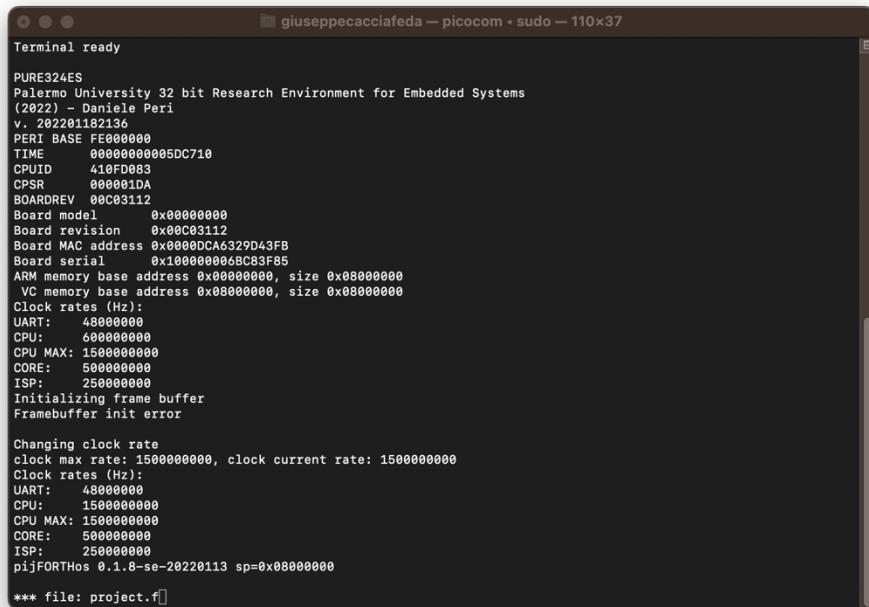
PURE324ES
Palermo University 32 bit Research Environment for Embedded Systems
(2022) - Daniele Peri
v. 202201182136
PERI BASE FE00000
TIME      0000000005DC710
CPUID     410FD083
CPSR     000001DA
BOARDREV  00C03112
Board model      0x00000000
Board revision   0x00C83112
Board MAC address 0x000000CA6329D43FB
Board serial     0x100000006BC83F85
ARM memory base address 0x00000000, size 0x08000000
VC memory base address 0x08000000, size 0x08000000
Clock rates (Hz):
UART:    48000000
CPU:     60000000
CPU MAX: 150000000
CORE:    50000000
ISP:     25000000
Initializing frame buffer
Framebuffer init error

Changing clock rate
clock max rate: 1500000000, clock current rate: 1500000000
Clock rates (Hz):
UART:    48000000
CPU:     1500000000
CPU MAX: 1500000000
CORE:    50000000
ISP:     25000000
pi:jFORTHos 0.1.8-se-20220113 sp=0x08000000
```

È necessario adesso procedere al caricamento del codice sorgente del progetto nel Target. Questa operazione richiede l'utilizzo di una sequenza specifica di tasti in **picocom**:

**[CTRL] + A [CTRL] + S**

Usando questa sequenza, **picocom** chiederà di inserire il nome del file che si desidera inviare al Target, il quale dovrà includere anche l'estensione del file. Si inserisce il path:



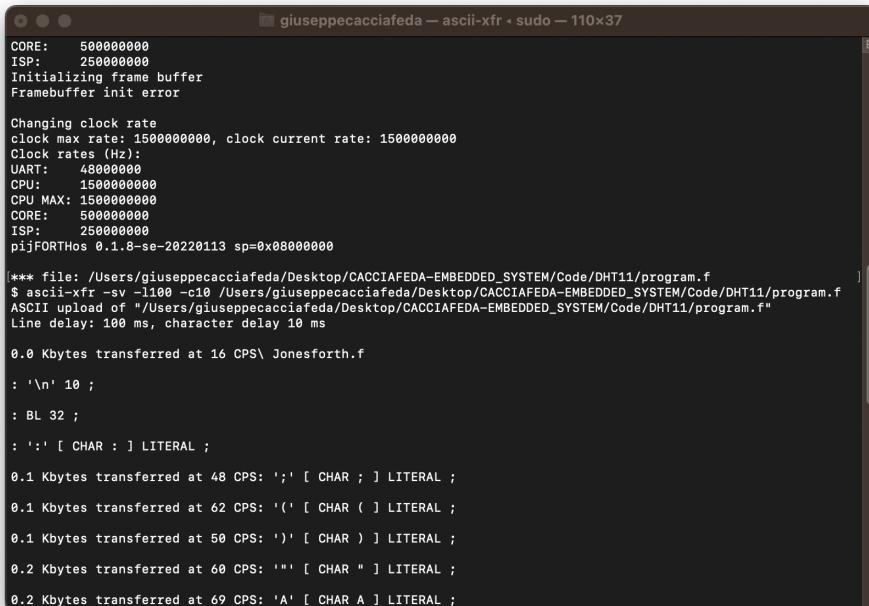
```

Terminal ready
PURE324ES
Palermo University 32 bit Research Environment for Embedded Systems
(2022) - Daniele Peri
v. 202201182136
PERI BASE FE000000
TIME 00000000005DC710
CPUID 410FD0B3
CPSR 000001DA
BOARDREV 00C03112
Board model 0x00000000
Board revision 0x00C03112
Board MAC address 0x0000DCA6329D43FB
Board serial 0x100000006BC83F85
ARM memory base address 0x00000000, size 0x08000000
VC memory base address 0x08000000, size 0x08000000
Clock rates (Hz):
UART: 4800000
CPU: 60000000
CPU MAX: 150000000
CORE: 50000000
ISP: 25000000
Initializing frame buffer
Framebuffer init error

Changing clock rate
clock max rate: 150000000, clock current rate: 150000000
Clock rates (Hz):
UART: 4800000
CPU: 150000000
CPU MAX: 150000000
CORE: 50000000
ISP: 25000000
pijFORTHos 0.1.8-se-20220113 sp=0x08000000
*** file: project.f

```

Una volta fatto ciò, verrà avviato automaticamente il caricamento:



```

CORE: 50000000
ISP: 25000000
Initializing frame buffer
Framebuffer init error

Changing clock rate
clock max rate: 150000000, clock current rate: 150000000
Clock rates (Hz):
UART: 4800000
CPU: 150000000
CPU MAX: 150000000
CORE: 50000000
ISP: 25000000
pijFORTHos 0.1.8-se-20220113 sp=0x08000000
[...]
*** file: /Users/giuseppecacciafeda/Desktop/CACCIAFEDA-EMBEDDED_SYSTEM/Code/DHT11/program.f
$ ascii-xfr -sv -l100 -c10 /Users/giuseppecacciafeda/Desktop/CACCIAFEDA-EMBEDDED_SYSTEM/Code/DHT11/program.f
ASCII upload of "/Users/giuseppecacciafeda/Desktop/CACCIAFEDA-EMBEDDED_SYSTEM/Code/DHT11/program.f"
Line delay: 100 ms, character delay 10 ms
0.0 Kbytes transferred at 16 CPS\ Jonesforth.f

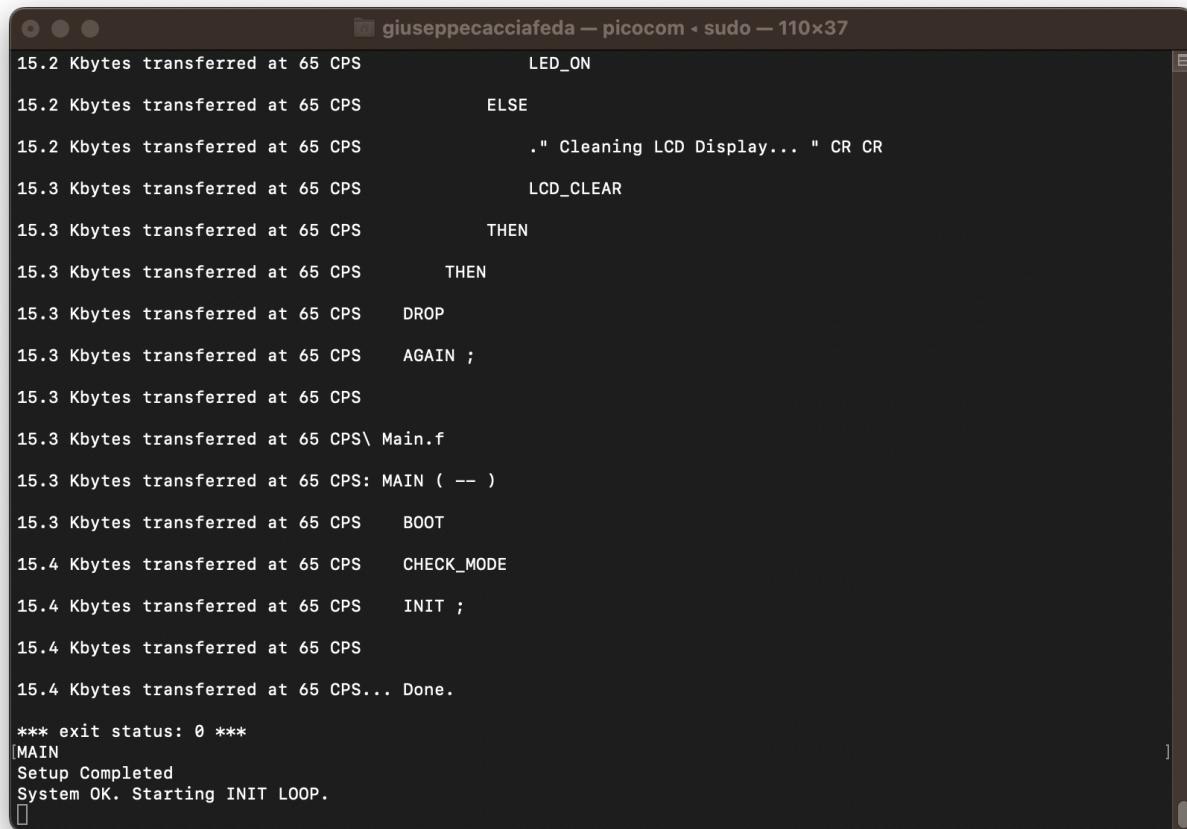
: '\n' 10 ;
: BL 32 ;
: ':' [ CHAR : ] LITERAL ;

0.1 Kbytes transferred at 48 CPS: ';' [ CHAR ; ] LITERAL ;
0.1 Kbytes transferred at 62 CPS: '(' [ CHAR ( ) ] LITERAL ;
0.1 Kbytes transferred at 50 CPS: ')' [ CHAR ) ] LITERAL ;
0.2 Kbytes transferred at 60 CPS: '"' [ CHAR " ] LITERAL ;
0.2 Kbytes transferred at 69 CPS: 'A' [ CHAR A ] LITERAL ;

```

Una volta che il programma sarà stato caricato con successo, bisognerà avviarlo eseguendo il comando **"MAIN"**.

Questa parola chiave è definita nel linguaggio FORTH di alto livello nel file “main.f”, la quale si occupa di eseguire tutte le operazioni necessarie per avviare il sistema in maniera corretta. Il programma verrà adesso eseguito e risulterà funzionante.



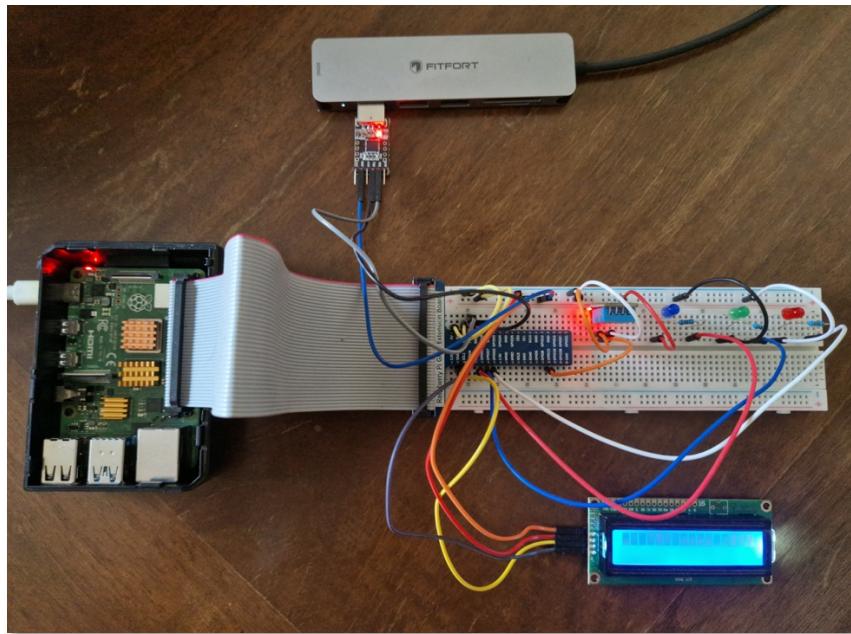
```
15.2 Kbytes transferred at 65 CPS          LED_ON
15.2 Kbytes transferred at 65 CPS          ELSE
15.2 Kbytes transferred at 65 CPS          ." Cleaning LCD Display... " CR CR
15.3 Kbytes transferred at 65 CPS          LCD_CLEAR
15.3 Kbytes transferred at 65 CPS          THEN
15.3 Kbytes transferred at 65 CPS          THEN
15.3 Kbytes transferred at 65 CPS          DROP
15.3 Kbytes transferred at 65 CPS          AGAIN ;
15.3 Kbytes transferred at 65 CPS
15.3 Kbytes transferred at 65 CPS\ Main.f
15.3 Kbytes transferred at 65 CPS: MAIN ( -- )
15.3 Kbytes transferred at 65 CPS          BOOT
15.4 Kbytes transferred at 65 CPS          CHECK_MODE
15.4 Kbytes transferred at 65 CPS          INIT ;
15.4 Kbytes transferred at 65 CPS
15.4 Kbytes transferred at 65 CPS... Done.

*** exit status: 0 ***
[MAIN
 Setup Completed
 System OK. Starting INIT LOOP.
]
```

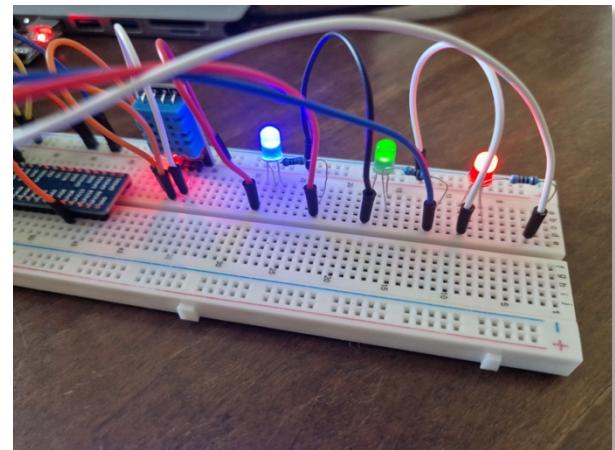
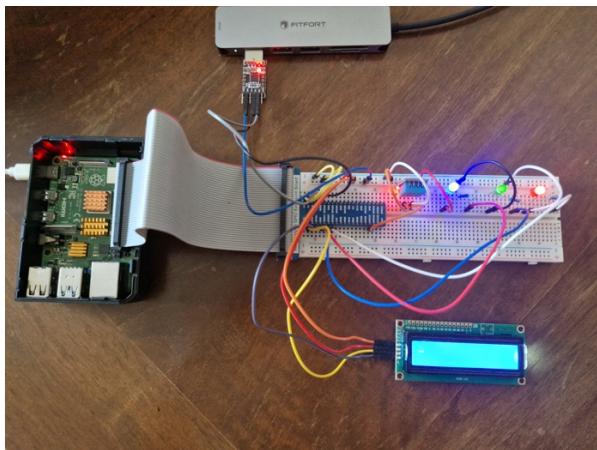
A questo punto sarà possibile rimuovere il connettore UART collegato al PC e visualizzare il risultato ottenuto.

# Foto Test

## PANORAMICA DI SISTEMA

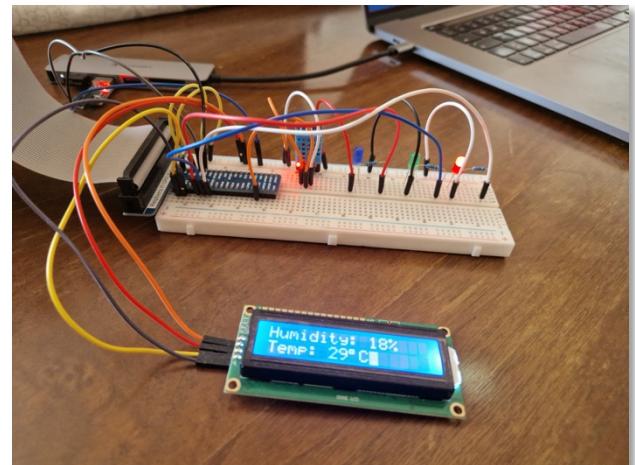
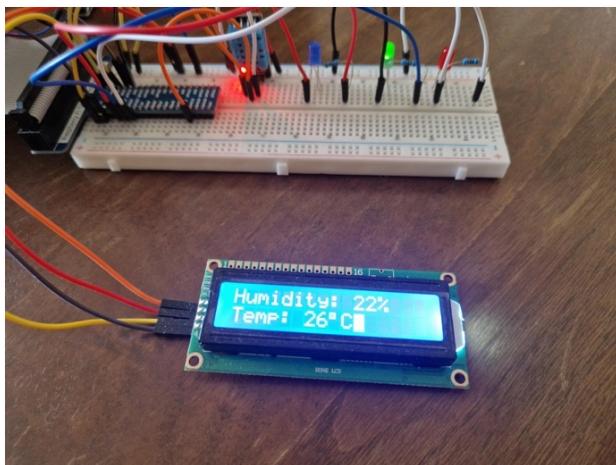


## AVVIAMENTO DEL SISTEMA

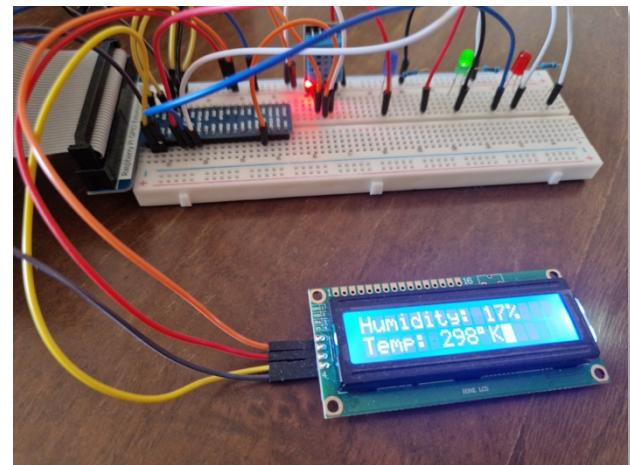
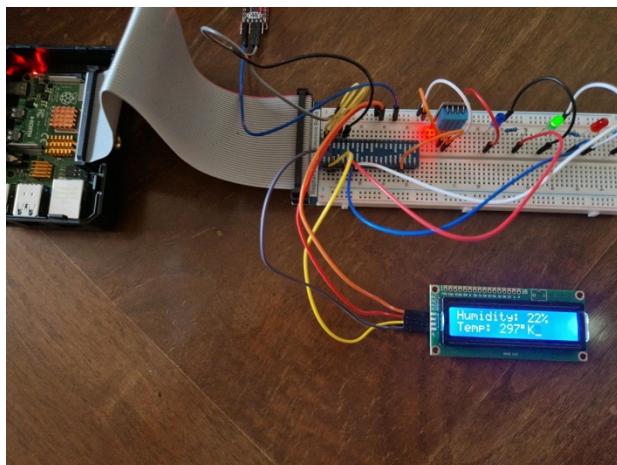


```
giuseppeacciafeda — picocom - sudo — 125x30
*** exit status: 0 ***
MAIN
Setup Completed
System OK. Starting INIT LOOP.
Loading Measurements...
Misurazione N° 0
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità rilevata: 22 %
Misurazione N° 1
Misurazione Eseguita
Temperatura rilevata: 14 °C
Umidità fuori dal range previsto
Misurazione N° 2
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità rilevata: 21 %
Misurazione N° 3
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità rilevata: 21 %
Misurazione N° 4
Misurazione Eseguita
Temperatura rilevata: 29 °C
Umidità rilevata: 21 %
Misurazione N° 5
Misurazione Eseguita
Temperatura rilevata: 14 °C
Umidità rilevata: 21 %
Misurazione N° 6
```

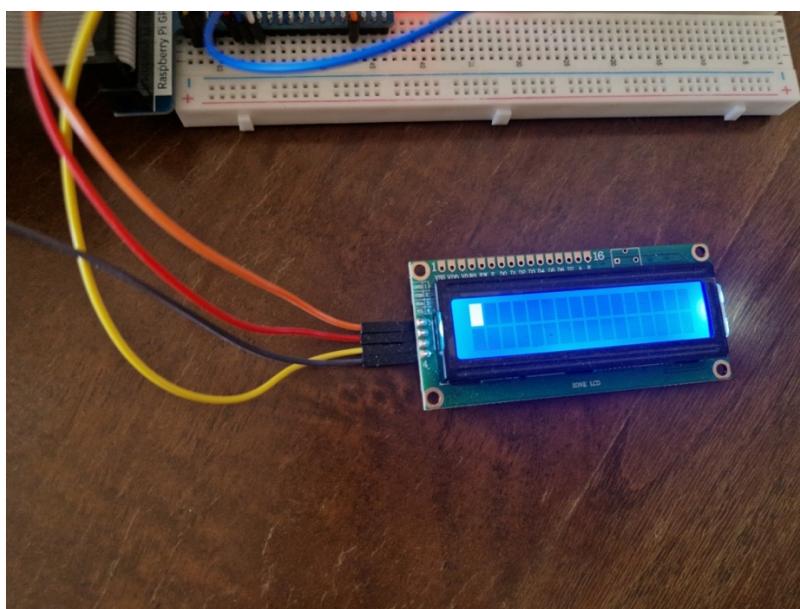
MISURAZIONE GRADI CELSIUS



MISURAZIONE GRADI KELVIN

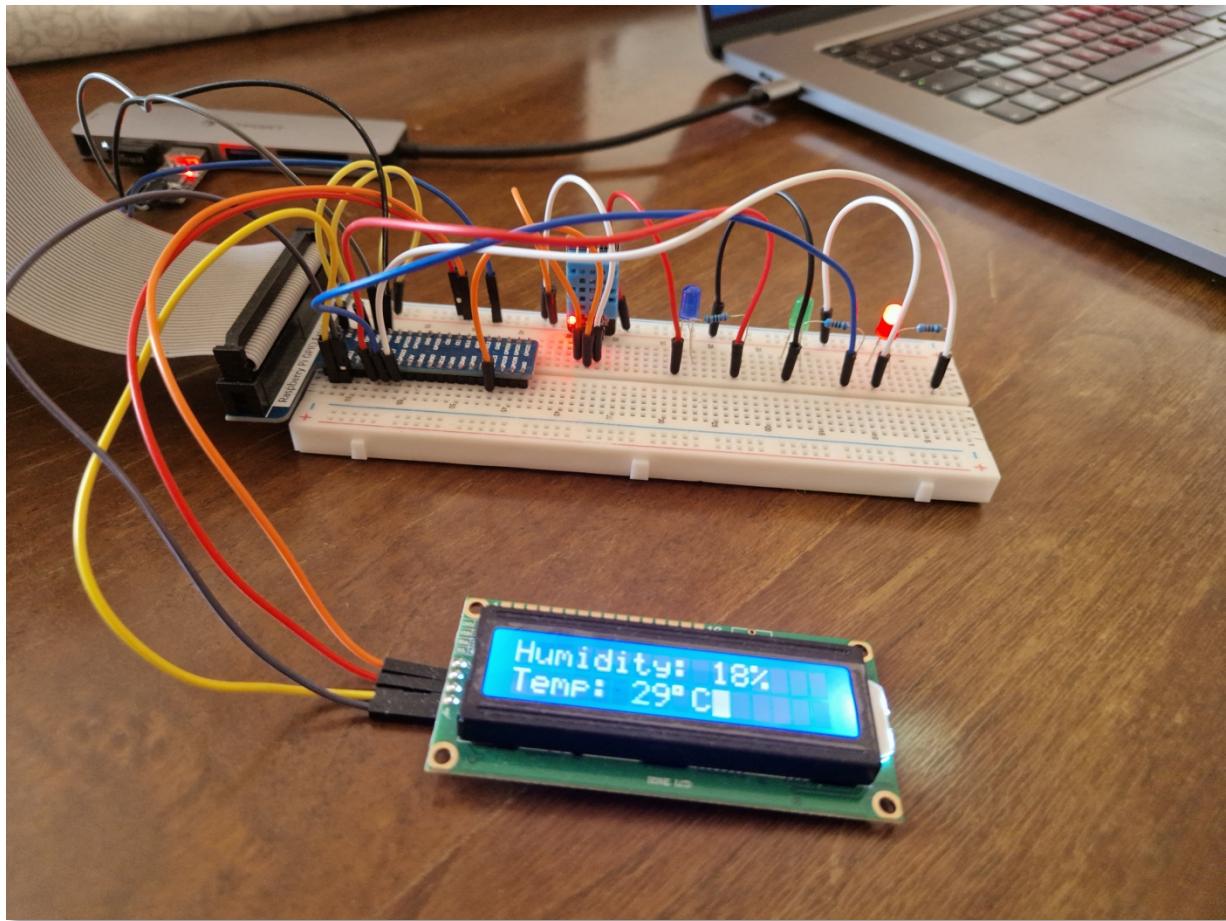


CLEANING DEL DISPLAY



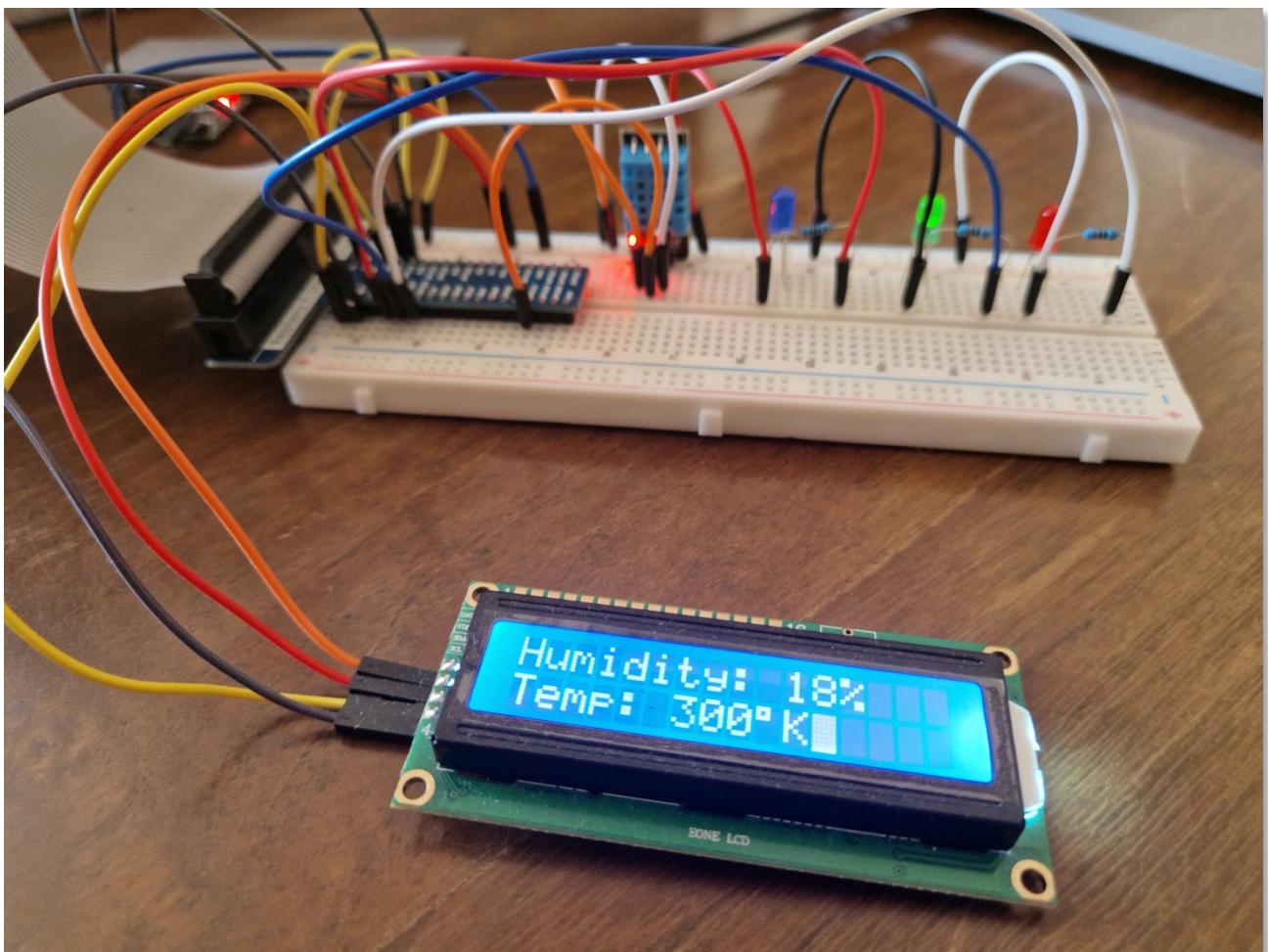
## RUN DI PROVA

*Misurazione in gradi Celsius*

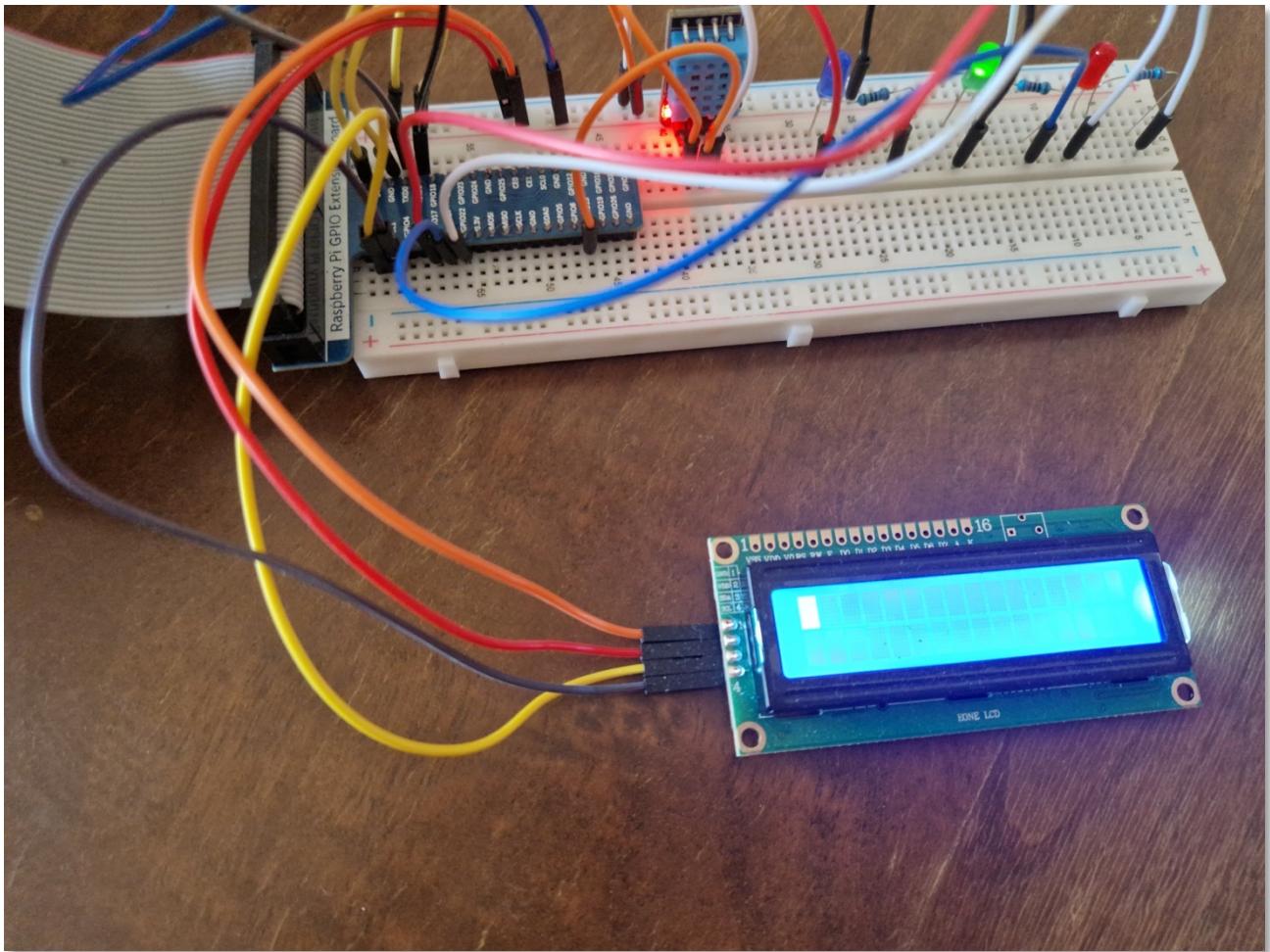


```
giuseppecacciafeda — picocom < sudo — 112x33
Unidità fuori dal range previsto
Misurazione N° 24
Misurazione Eseguita
Temperatura rilevata: 30 °C
Umidità rilevata: 13 %
Misurazione N° 25
Misurazione Eseguita
Temperatura rilevata: 30 °C
Umidità fuori dal range previsto
Misurazione N° 26
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità rilevata: 13 %
Misurazione N° 27
Misurazione Eseguita
Temperatura rilevata: 30 °C
Umidità fuori dal range previsto
Misurazione N° 28
Misurazione Eseguita
Temperatura rilevata: 30 °C
Umidità rilevata: 27 %
Misurazione N° 29
Misurazione Eseguita
Temperatura rilevata: 15 °C
Umidità rilevata: 13 %
Somma delle temperature rilevate: 525
Somma delle umidità rilevate: 412
Numero di misurazioni di temperatura valide: 18
Numero di misurazioni di umidità valide: 22
Humidity: 18 %
Temp: 29 °C
RED LED ON!
```

Misurazione in gradi Kelvin



```
Temperatura rilevata: 30 °C
Umidità rilevata: 14 %
Misurazione N° 24
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità fuori dal range previsto
Misurazione N° 25
Misurazione Eseguita
Temperatura rilevata: 30 °C
Umidità fuori dal range previsto
Misurazione N° 26
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità rilevata: 14 %
Misurazione N° 27
Misurazione Eseguita
Temperatura rilevata: 30 °C
Umidità fuori dal range previsto
Misurazione N° 28
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità fuori dal range previsto
Misurazione N° 29
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità rilevata: 28 %
Somma delle temperature rilevate: 435
Somma delle umidità rilevate: 224
Numero di misurazioni di temperatura valide: 16
Numero di misurazioni di umidità valide: 12
Humidity: 18 %
Temp: 27 ° C
GREEN LED ON!
```

*Cleaning del Display LCD*

```
giuseppecacciafeda — bash — 112x33
Umidità fuori dal range previsto
Misurazione N° 29
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità rilevata: 28 %
Somma delle temperature rilevate: 435
Somma delle umidità rilevate: 224
Numero di misurazioni di temperatura valide: 16
Numero di misurazioni di umidità valide: 12
Humidity: 18 %
Temp: 27 ° C
GREEN LED ON!

Cleaning LCD Display...

Loading Mesuraments...
Misurazione N° 0
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità rilevata: 28 %
Misurazione N° 1
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità rilevata: 28 %
Misurazione N° 2
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità fuori dal range previsto
Misurazione N° 3
Misurazione Eseguita
Temperatura fuori dal range previsto
Umidità fuori dal range previsto
Misurazione N° 4
```