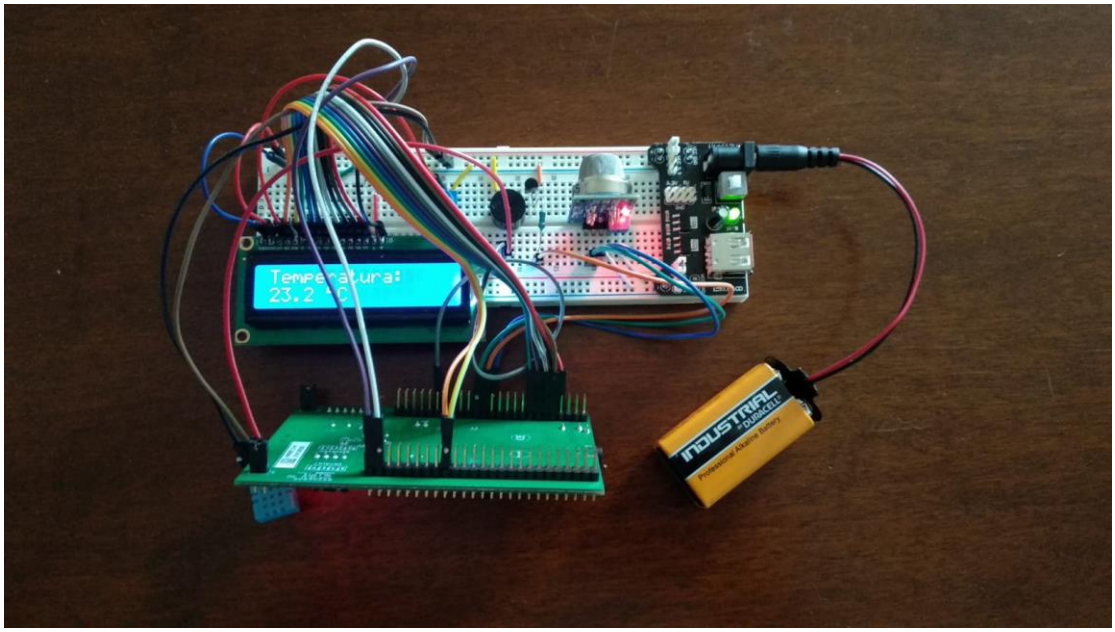
Università degli studi di Modena e Reggio Emilia
Dipartimento di Ingegneria “Enzo Ferrari”
Corso di Laurea Magistrale in Ingegneria Informatica



Progetto di un sensore di rilevamento fuga di gas naturale, temperatura ed umidità basato su STM32F4

PROGETTO DI CORSO

Werther Berselli, Fabrizio Di Blasi | Progettazione Elettronica Digitale | a.a 2018/2019

Sommario:

Introduzione:

In questa relazione viene esposto il processo di progettazione e prototipazione di una centralina casalinga per il rilevamento di fughe di gas naturale e misurazione di temperatura e umidità.
Il sistema ha lo scopo di allertare l'utente tramite un segnale acustico nel momento in cui il valore di ppm del gas nella stanza supera quello critico.
Il valore critico è stato fissato pari a 500 ppm, in modo da non superare mai gli 800 ppm (valore limite per legge).
Oltre al segnale acustico si è scelto di utilizzare un display LCD 16X2 in cui mostrare, in modo alternato, le misurazioni di temperatura e umidità. Il livello di gas nell'aria viene mostrato solamente quando rilevato.

Studio di fattibilità:

Requisiti e ambito di applicazione:

Il progetto presentato descrive un prodotto da considerarsi esclusivamente ad uso casalingo.
Il requisito di maggiore importanza è la rilevazione istantanea di un'eventuale fuga di gas e la durata maggiore possibile della batteria di alimentazione del sistema.
Essendo un'oggetto critico, è consigliato apporre un packaging ignifugo e resistente ad agenti esterni come polvere o sporcizia in modo da non compromettere il rilevamento del gas.

Vincoli:

Il prodotto finale per il corretto funzionamento dovrà essere disposto lontano da fonti di calore, in luoghi chiusi e non eccessivamente umidi.

In particolare i vincoli del prodotto derivano direttamente dai vincoli dei singoli componenti utilizzati; rilevazioni corrette sono assicurate in ambienti compresi tra 0°C e 50°C, umidità comprese tra il 20% e il 90% e ambienti adeguatamente ossigenati (intorno al 21% per mantenere il normale regime di utilizzo).

Valutazione dei costi:

Dalle analisi di mercato effettuate, si è deciso che il costo del prodotto finale non debba superare i 20 euro; pertanto sono stati identificati i seguenti componenti per lo sviluppo dei prodotti finali da portare sul mercato:

1. **Microcontrollore STM32L051C8T3**
   Una delle versioni più low power della famiglia STM32, adatto ad essere alimentato tramite una batteria, riesce a svolgere perfettamente le funzioni di progetto.

Regolabile a una frequenza di 4 Mhz, dispone di 64 KB di memoria Flash e 37 porte di I/O; più che sufficienti.
Costo: 1,74 euro (x1000 pezzi) (3,27 euro costo unitario) (mouser.com)

2. **Buzzer attivo SD1614T5-B5ME**
   Costo: 0,34 euro (aliexpress.com)

3. **Sensore di Gas MQ5**
   Costo: 0,99 euro (aliexpress.com)

4. **Sensore di Temperatura DHT11**
   Costo: 0,75 euro (aliexpress.com)

5. **Display LCD1602**
   Costo: 1,52 euro (aliexpress.com)

6. **Led rosso 5mm**
   Costo: 0,014 euro (1,14 euro / lotto 100 pcs) (aliexpress.com)

7. **Resistenze e transistor (BC547 / 2N2222)**
   Costo: 0,03 euro (0,48 euro / lotto 100 resistenze, 0,76 euro / lotto 100 transistor) (aliexpress.com)

Supponendo di produrre più di 1000 pezzi, il costo totale del prodotto ammonta a 5,384 euro.

Producendo meno di 1000 pezzi, il costo totale del singolo prodotto sale a 6,914 euro.

Componenti Impiegati nel prototipo:

Per la realizzazione del prototipo sono stati usati i seguenti componenti:

Board STM32F407G-DISC1:

Core
* ARM Cortex M4F con frequenza di clock impostata a 4MHz
• Memoria
* SRAM da 192 KB
* Memoria flash da 1024 MB
• Periferiche
* 16 Timer a 16 bit e 2 timer a 32 bit
* 16 convertitori A/D a 12 bit
* 2 convertitori A/D a 12 bit
* 3 x I2C
* 3 x SPI
* 6 x UART/USART

Buzzer piezoelettrico attivo:

Caratteristiche:
Alimentazione a 5V DC - 30mA (MAX)
SPL a 10cm: ≥85dB
Frequenza: 2300 ± 300Hz

Sensore di Gas MQ5:

Caratteristiche:
Alimentazione a 5V DC
150mA di assorbimento
Uscita TTL (D0) e analogica (A0)

La misurazione di questo sensore è affidabile solamente se il sistema è ad una temperatura posta tra -20 °C e +70 °C ed al di sotto del 95% di umidità.
Inoltre un parametro da non sottovalutare è la concentrazione di ossigeno nell'ambiente, che in condizioni standard è del 21%. Un tasso maggiore o inferiore può causare degli errori di misura della concentrazione di gas.

Il sensore comunica col microcontrollore tramite un'uscita analogica e digitale (TTL).
L'uscita digitale assume valore logico

1 – se non c'è fuga di gas
0 – se viene rilevata la fuga di gas

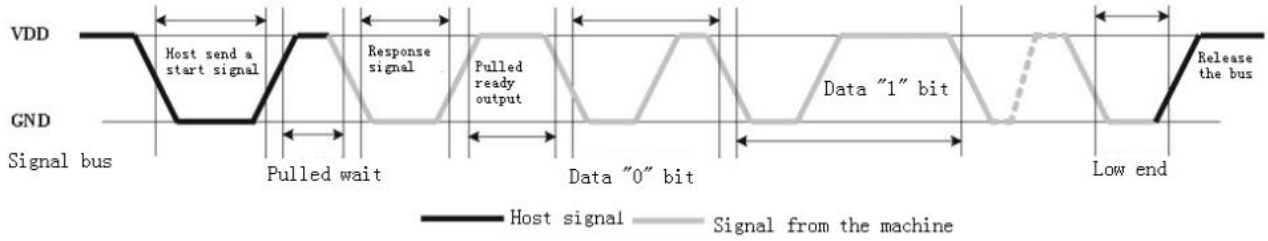
Tale segnale è utilizzato per la gestione dell'interrupt. Infatti appena avviene il rilevamento, viene mostrata sul display esclusivamente la misura del gas.
L'uscita analogica invece è quella che fornisce la misurazione in ppm della concentrazione di gas. Tale output viene fornito al convertitore ADC del microcontrollore.

Sensore di temperatura e umidità DHT 11:

Caratteristiche:
Alimentazione da 3,3V a 5,5V
Gamma di misura umidità: da 20% a 90% RH
Gamma di misura temperatura: da 0°C a 50°C
Precisione per la misura dell'umidità: ± 1% RH
Precisione per la misura della temperatura: ± 2 °C

Questo sensore usa un protocollo di comunicazione chiamato "Single Wire bidirectional".
È un paradigma comunicativo di tipo master-slave; infatti il microcontrollore (host) invia un segnale di start, mantenendo a livello logico 0 il segnale per almeno 18ms e portandolo ad 1 per un massimo di 40µs.
Ciò permette al DHT di rilevare la richiesta dando un feedback al microcontrollore, portando a 0 il livello per 80µs e rispostandolo ad 1 per 80µs.

Finita la fase di check il sensore comunicherà un bit alla volta.
Per trasferire un bit ad 1 si ha 50 µs a livello logico 0 e 70µs a livello logico 1, per trasferire uno “0” si ha 50µs a livello logico 0 ed 26/28µs a livello logico 1.



Data Timing Diagram

Il DHT invia 5 pacchetti da 8 bit:
Parte intera umidità | Parte decimale umidità | Parte intera Temperatura | Parte decimale Temperatura
Infine il checksum, che sarà la somma di tutti i pacchetti inviati; ciò è molto utile per rilevare errori comunicativi.

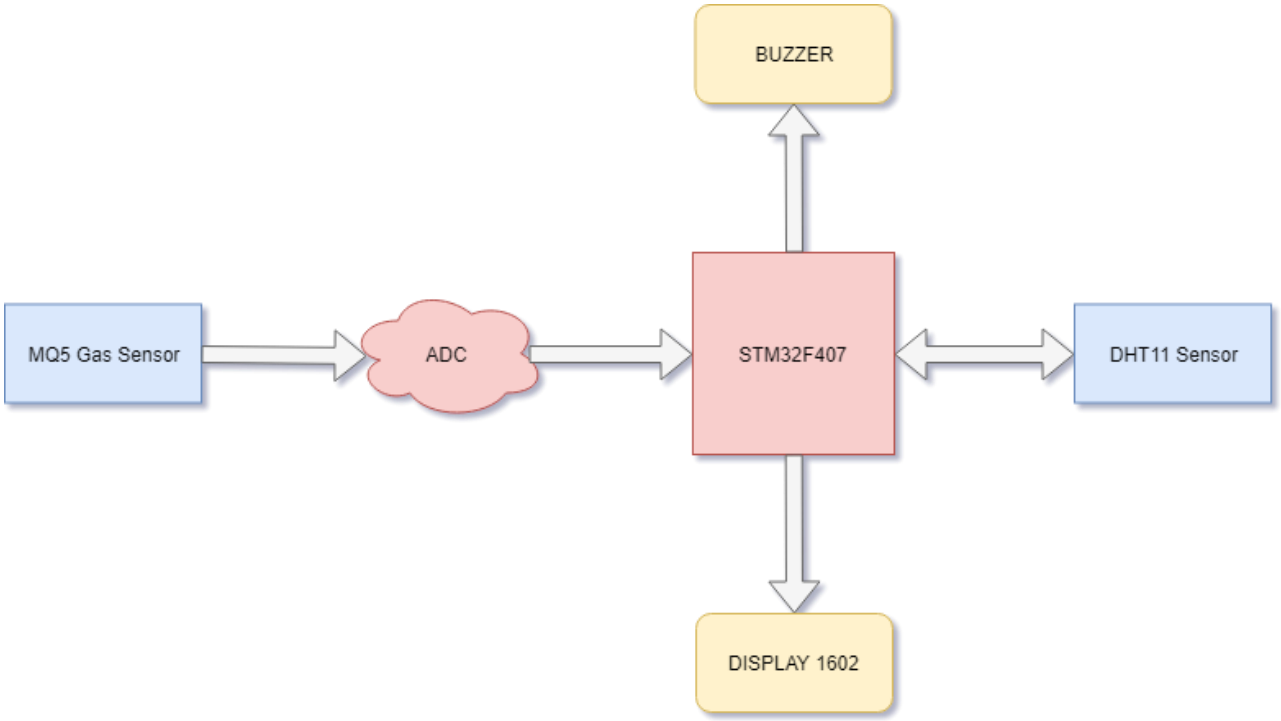
Display LCD1602:

Caratteristiche:
Alimentazione a 5V
Interfaccia parallela
Retroilluminazione a led
Controller SPLC780 compatibile con HD44780

Il controller del display possiede 2 registri a 8-bit, un Istruction Register (IR) e un Data Register (DR). L’IR memorizza i codici operativi (modalità di visualizzazione, cursore di testo..), il DR invece memorizza temporaneamente le informazioni da leggere o scrivere da Data RAM (DRAM) o Character Gerneator RAM (CGRAM). La scelta del registro al quale accedere viene effettuata tramite un segnale di Register Selector (RS).
Nel prototipo sviluppato è stata utilizzata la modalità ad 8 bit, quindi, vengono impiegate tutte le linee da $DB0$ a $DB7$
Il microcontrollore fornirà i comandi al display solamente tramite i registri IR e DR in questo modo:
• Write signal - R/W: Un segnale alto (1) attiva la modalità di lettura, un segnale basso (0) attiva la modalità di scrittura. In questo caso utilizzando il display esclusivamente come periferica di output è stato collegato al GND.
• Read signal - RS: (1) permette di accedere all’IR (in scrittura), (0) permette di accedere al DR (in lettura o scrittura).
• Enable - E: Quando il segnale passa a livello logico “0” verranno letti i dati dal bus.

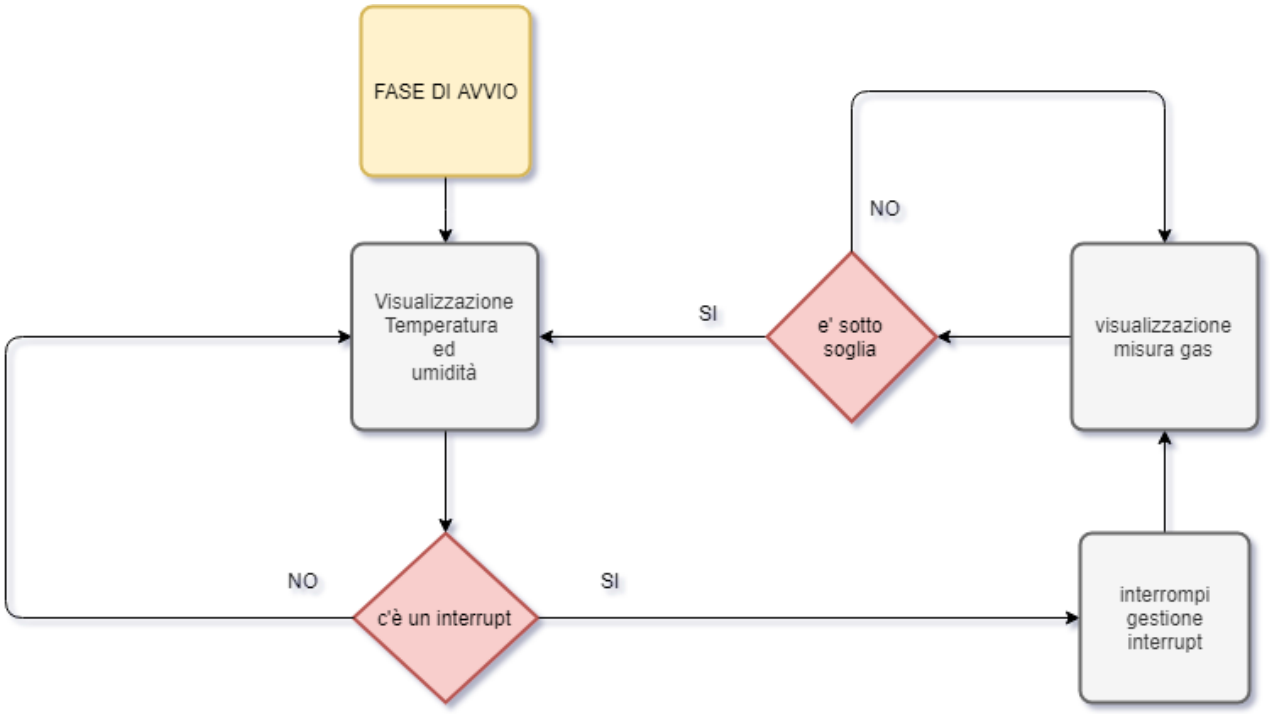
Implementazione:

Schema a blocchi:

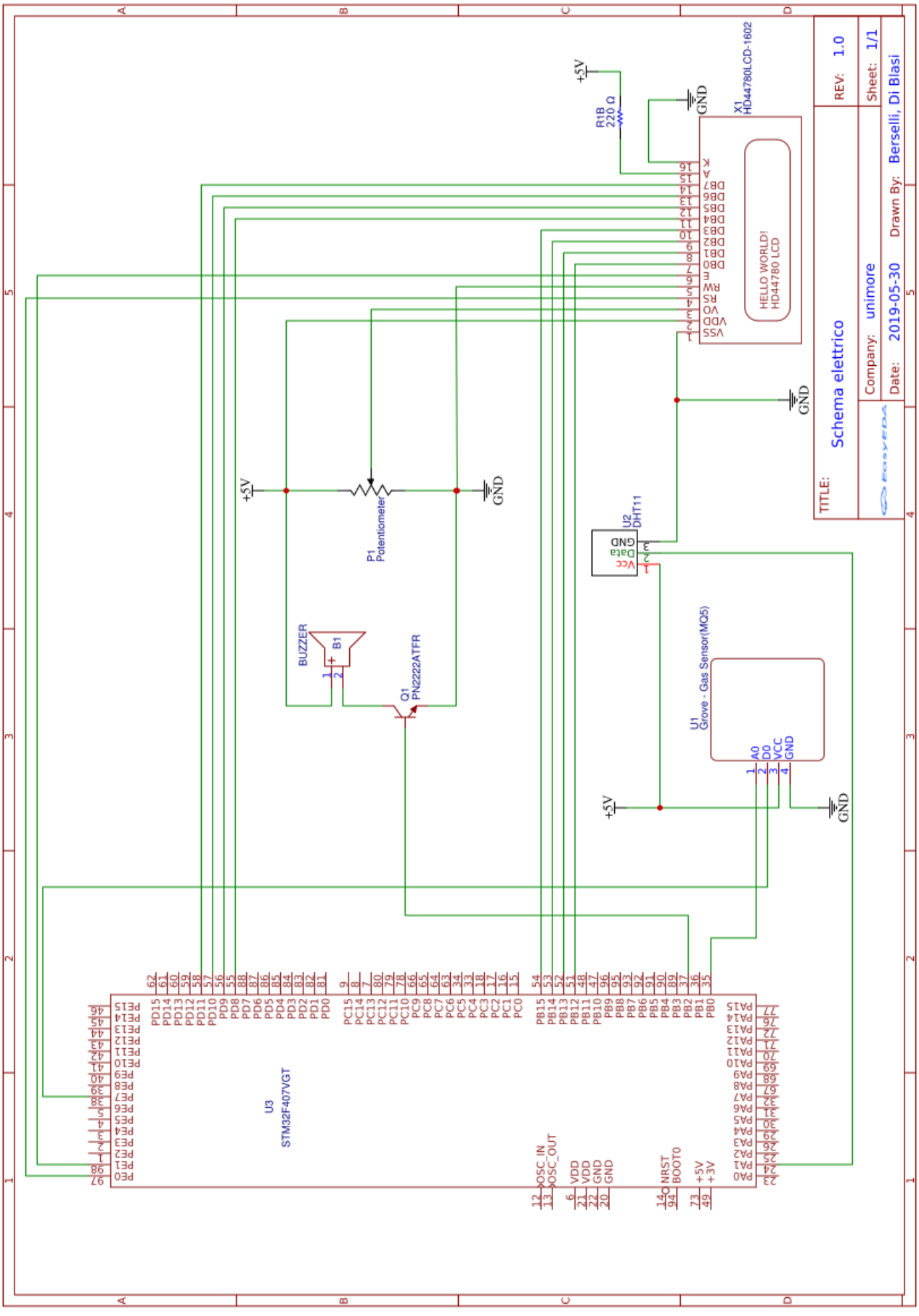


Nello schema a blocchi si può capire, tramite l'orientamento delle frecce, il verso della comunicazione tra i vari dispositivi. Si può notare come il sensore MQ5, avendo uscita analogica debba prima interagire con un ADC, nel nostro caso interno al microcontrollore, e poi tali dati potranno essere elaborati dal MCU.

Le funzionalità svolte dal microcontrollore possono essere schematizzate in questo modo:



Schema elettrico:



Software:

il software è stato organizzato in più file sorgenti, inoltre, è stato inserito un watchdog per avere la certezza del corretto funzionamento.

Sensore DHT11*:*

tra le tante funzioni utili per la gestione di questo sensore vi è la seguente:

```
1.  uint8_t dht11_read(void)
2.  {
3.      uint8_t i,j;
4.      for (j=0;j<8;j++)
5.      {
6.          while (!(HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_1)));   // wait for the pin to go high

7.          DWT_Delay_us (40);   // wait for 40 us
8.          //HAL_Delay(40);
9.          if ((HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_1)) == 0)   // if the pin is low
10.         {
11.             i&= ~(1<<(7-j));   // write 0
12.         }
13.         else i|= (1<<(7-j));  // if the pin is high, write 1
14.         while ((HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_1)));  // wait for the pin to go low
15.     }
16.     return i;
17. }
```

Questa è la funzione che si occupa di leggere il bit che viene comunicato dal sensore. Infatti a riga 6, tramite un'attesa attiva, si aspetta che il sensore lasci libero il pin e poi dopo un ritardo di 40µs (effettuati tramite la funzione *DWT_Delay_us( )*) si va a controllare il livello logico imposto dal sensore.
Se è a livello logico 0 andiamo a scrivere un bit a 0 in caso contrario un bit ad 1, tutto effettuando dovuti shift a sinistra, poiché il sensore comunicherà prima il MSB del dato. Essendo un protocollo "one wire" è possibile trovare all'interno due funzioni:

```
1.  void set_gpio_input(void){...}
```

e

```
2.  void set_gpio_output(void){...}
```

che servono a cambiare la modalità di funzionamento del pin del MCU, la prima come input, la seconda come output, sfruttando le API messe a disposizione dal compilatore.

Modalità di misurazione della temperatura e umidità:

Per mitigare possibili errori di misurazione del sensore, la misurazione che verrà mostrata all'utente è frutto della media di 10 misurazioni. Ciò viene effettuato tramite l'utilizzo di un buffer circolare in cui la misura più vecchia in ordine cronologico è sempre quella ad uscire dal vettore.
La funzione che si occupa di ciò è la seguente:

```
1.  int dht11_measurements_mean(struct dht11_measurements* u)
2.  {
3.      if(u->size == MSDHT11)
4.      {
5.          for(int i = 0; i < MSDHT11; i++)
6.          {
7.              u->Rh_mean += u->Rh[i];
8.              u->Temp_mean += u->Temp[i];
9.          }
10.
11.         u->Rh_mean /= MSDHT11;
12.         u->Temp_mean /= MSDHT11;
13.     }
14.     else
15.     {
16.         return -1;
17.     }
18.
19.     return 0;
20. }
```

La stessa logica di misurazione è stata applicata alla misurazione del GAS

Sensore MQ5:

Il sensore MQ5 viene gestito tramite interrupt, grazie alla sua uscita TTL.
Infatti, il pin del MCU quando rileva un segnale logico “0” interrompe la visualizzazione della temperatura ed umidità e visualizza esclusivamente la rilevazione del GAS.
Questa funzione verrà eseguita finché il livello del gas non sarà sceso sotto 500 ppm. Inoltre si è reso necessario riavviare il timer del whatchdogs che, secondo una stima svolta in modalità debug, è settato ad 8 secondi (tempo massimo necessario a visualizzare una sola volta temperatura e umidità).
Nella funzione sottostante (riga 19), si può invece notare la API che ha il ruolo di interpellare il convertitore ADC impostato con 12bit di precisione per prelevare il valore convertito. In questa funzione viene svolta anche la media delle misurazioni da mostrare a video.

```
1.  void check_gas(ADC_HandleTypeDef hadc1, IWDG_HandleTypeDef hiwdg)
2.  {
3.      int val;
4.      int cond;
5.      cond = HAL_GPIO_ReadPin(GPIOB,GAS_FLAG_Pin);
6.      while(cond)
7.      {
8.          HAL_IWDG_Refresh(&hiwdg);
9.          hiwdg.Instance->PR = hiwdg.Init.Prescaler;
10.         hiwdg.Instance->RLR = hiwdg.Init.Reload;
11.
12.         val = 0;
13.         HAL_ADC_Start(&hadc1);
14.         HAL_GPIO_WritePin(GPIOB,BUZZ_Pin,GPIO_PIN_SET);
15.         LCD1602_1stLine();
16.         LCD1602_print("Gas:            ");
17.         LCD1602_2ndLine();
18.         HAL_ADC_PollForConversion(&hadc1,1000);
19.         for(int i = 0; i < MSMQ5; i++)
20.         {
21.             val += HAL_ADC_GetValue(&hadc1);
22.         }
23.         val /= MSMQ5;
24.         LCD1602_PrintInt(val);
25.         LCD1602_print(" ppm     ");
26.         DWT_Delay_us (2000000);
```

```
        if(val < THRESHOLD)
        {
            cond=0;
            HAL_GPIO_WritePin(GPIOB,BUZZ_Pin,GPIO_PIN_RESET);

            HAL_IWDG_Refresh(&hiwdg);
            hiwdg.Instance->PR = hiwdg.Init.Prescaler;
            hiwdg.Instance->RLR = hiwdg.Init.Reload;
        }
    }
}
```

Display LCD1602:

Il display ha un funzionamento molto rigoroso, infatti si devono fornire al controller degli specifici comandi. Per prima cosa quando si vuole visualizzare qualcosa a video, avendo due linee, si deve selezionare la riga sulla quale si vuole scrivere. Ciò è possibile chiamando una tra le seguenti funzioni:

```
// Set cursor to 1st line.
void LCD1602_1stLine(void)
{
    LCD1602_writeCommand(0x80);
}

// Set cursor to 2nd line.
void LCD1602_2ndLine(void)
{
    LCD1602_writeCommand(0xc0);
}
```

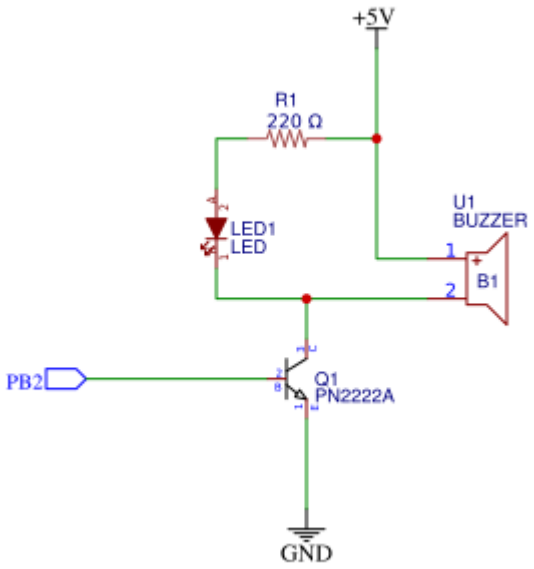
Invece per scrivere i caratteri sullo schermo viene chiamata la funzione seguente, incapsulata da altre per facilitare la stampa di tipi di dato interi o caratteri.

```
static void LCD1602_write(uint8_t byte)
{
    HAL_GPIO_WritePin(PORT_LSB, D0_PIN, (GPIO_PinState)(byte&0x1));
    HAL_GPIO_WritePin(PORT_LSB, D1_PIN, (GPIO_PinState)(byte&0x2));
    HAL_GPIO_WritePin(PORT_LSB, D2_PIN, (GPIO_PinState)(byte&0x4));
    HAL_GPIO_WritePin(PORT_LSB, D3_PIN, (GPIO_PinState)(byte&0x8));

    HAL_GPIO_WritePin(PORT_MSB, D4_PIN, (GPIO_PinState)((byte>>4)&0x1));
    HAL_GPIO_WritePin(PORT_MSB, D5_PIN, (GPIO_PinState)((byte>>4)&0x2));
    HAL_GPIO_WritePin(PORT_MSB, D6_PIN, (GPIO_PinState)((byte>>4)&0x4));
    HAL_GPIO_WritePin(PORT_MSB, D7_PIN, (GPIO_PinState)((byte>>4)&0x8));

    //Write the Enable pulse
    HAL_GPIO_WritePin(PORT_RS_and_E, PIN_E, GPIO_PIN_SET);
    TIM3->ARR = 10-1;
    TIM3->SR &= ~(0x0001);  // Clear UEV flag
    TIM3->CR1 |= 1UL;
    while((TIM3->SR&0x0001) != 1);
    HAL_GPIO_WritePin(PORT_RS_and_E, PIN_E, GPIO_PIN_RESET);
    TIM3->ARR = 60-1;
    TIM3->SR &= ~(0x0001);  // Clear UEV flag
    TIM3->CR1 |= 1UL;
    while((TIM3->SR&0x0001) != 1);
}
```

Variante per non udenti:



Questa versione può o non essere fornita in base alle richieste dell'utente.
Oltre al buzzer viene aggiunto un led rosso che si illuminerà quando il MCU rileva la fuga di GAS e contemporaneamente suonerà il buzzer.

Conclusioni:

Questo progetto ci ha consentito di acquisire più familiarità con il mondo dei microcontrollori; imparando a utilizzare componenti e sensori da zero basandoci sulle indicazioni dei datasheet per capire come interfacciarli con la nostra board di riferimento.
Il sistema nella sua versione finale si presenta come un oggetto utile da impiegare per tenere monitorata la qualità dell'aria in ambiente domestico, tuttavia per poterlo impiegare in ambiti industriali sarebbe necessaria la sostituzione di alcuni componenti in favore di altri qualitativamente migliori, più affidabili e costosi.

Valutazioni sui consumi:

Il microcontrollore è stato settato per lavorare alla frequenza di 4 MHz. Questa è la frequenza minima ottenibile in quanto il convertitore AD non permette di scendere oltre questo valore.
Il sistema raggiunge così un'ottima efficienza energetica, garantendo agli utenti finali un'elevata durata della batteria.

Ottimizzazioni e miglioramenti futuri:

I miglioramenti fondamentali potranno essere raggiunti sostituendo il sensore DHT11 con un altro sensore di qualità superiore (come il DHT22) e aggiungendo altri sensori di GAS per coprire uno spettro più ampio dei gas rilevabili dal sistema (come l'MQ7 per il monossido di carbonio).
Un altro miglioramento possibile è dato dalla sostituzione del display LCD1602 in favore di un display più grande e con risoluzione maggiore, in grado di indicare anche i livelli di carica della batteria.
Sarebbe poi una comodità in più per l'utente finale avere dei pulsanti a corredo del sistema che permettano di cambiare il dato visualizzato, con la possibilità di fissare a schermo la visualizzazione del/dei dato/i di maggior interesse dell'utente.