UNIVERSITÀ
degli STUDI
di CATANIA

DIPARTIMENTO DI INGEGNERIA
ELETTRICA ELETTRONICA E INFORMATICA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Mario Anthony Guerrera

Progetto di un Sistema IoT utile al parcheggio con interfaccia OLED IIC e Sensore di Prossimità ad Ultrasuoni

---

Relatore:
Chiar.mo Prof. Davide Patti

Anno Accademico 2023/2024

INDICE

Introduzione

L' Internet of things, spesso abbreviato in IoT, è la nuova estensione dell'internet che collega insieme oggetti in grado di interagire con l'ambiente circostante allo scopo di migliorare la qualità della vita.

L'obbiettivo del progetto e quindi di questa tesi, è realizzare un sistema IoT che permetta ad un veicolo di interagire con l'ambiente esterno; nel nostro caso, permetterà di conoscere la distanza fra il dispositivo e l'ambiente circostante.

Il Sistema è realizzato con una scheda Elegoo Uno R3 equivalente alla famosa scheda Arduino Uno R3 basate entrambe sul microcontrollore ATmega328P[1].

L'esposizione è così strutturata:

- Nel primo capitolo si parla delle tecnologie di base del Progetto con particolare approfondimento ai sensori utilizzati.
- Nel secondo capitolo si discute dei fondamenti dell'iot e della sua importanza nel settore automobilistico.
- Nel terzo capitolo si affronta il Progetto di tesi in tutte le sue versioni, digitale e fisico.
- Nel quarto capitolo vengono illustrati i possibili miglioramenti del Progetto e come può integrarsi in un contesto IoT.

[1] https://www.microchip.com/en-us/product/atmega328p#document-table

1. Tecnologie di base

In questo capitolo tratteremo tutte le tecnologie utilizzate nel progetto, come i sensori utilizzati per la distanza e la temperatura, i dispositivi di output e i microcontrollori.

1.1. Panoramica dell'Arduino Uno R3

1.1.1. Generalità

I microcontrollori (MCU) sono circuiti integrati che combinano nello stesso chip un microprocessore con una serie di componenti periferici come: memoria, timer, convertitori analogici-digitali e soprattutto pin di I/O.

A differenza di un microprocessore (CPU) i microcontrollori non sono pensati per svolgere onerose attività di calcolo ma per interagire il più possibile con l'ambiente quindi gestire numerosi scambi fra le periferiche di Input/Output.

I microcontrollori sono divisi in famiglie a seconda del loro microprocessore. Ciò che può differire fra le varie schede della stessa famiglia sono i componenti hardware. Per esempio, Arduino Uno R3, Arduino Nano, Arduino Pro Mini, Arduino Micro, Elegoo Uno R3 utilizzano tutti lo stesso microprocessore ma hanno componenti diversi (Arduino, 2024) (Gandolfo, Scheda Elegoo Uno R3, 2022).

La scelta del microcontrollore fra le varie famiglie o fra le varianti di una stessa famiglia deve essere basata non sulle performance migliori, bensì sulla scelta ottimale per il progetto.

Ogni microcontrollore, come detto in precedenza, possiede all' interno dello stesso chip non solo la CPU, ma anche le memorie e tutti i vari moduli e componenti hardware (figura 1.1) per avere maggiore velocità e costi ridotti.

I componenti, integrati nello stesso chip, garantiscono un minor tempo di accesso ai dati e quindi una maggiore velocità.

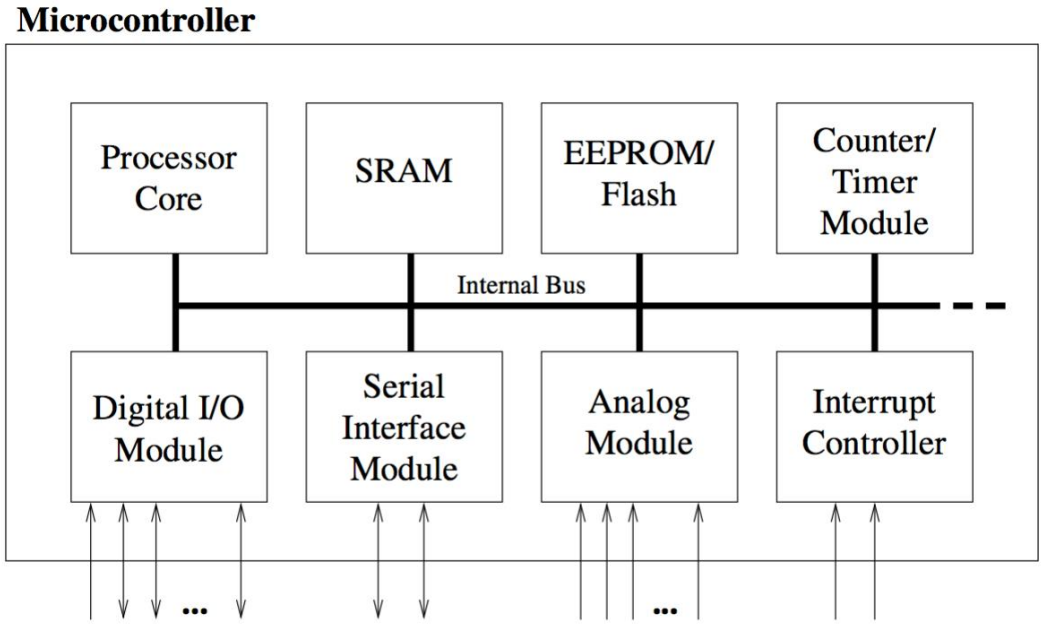

Figura 1.1. Componenti harware di un microcontrollore.

1.1.2. Memorie

Le memorie di un microcontrollore possono essere diverse a seconda del loro utilizzo e possono differire in volatili e non volatili.

Fra le volatili vi sono: SRAM e DRAM; tra le non-volatili: ROM, PROM, EPROM, EEPROM, flash EEPROM ed NVRAM.

L'Arduino Uno R3 comprende una memoria da 32KB flash EEPROM utilizzata per memorizzare il codice da eseguire, 2KB SRAM per le variabili utilizzate dal programma e 1KB EEPROM per le costanti (Arduino, 2024).

1.2. Sensori e Interfacce

1.2.1. HC-SR04

Il primo sensore di cui ci occupiamo è quello ad ultrasuoni HC-SR04. Questi tipi di sensori non misurano direttamente la distanza ma forniscono il tempo impiegato da un segnale per andare e tornare dal sensore.

La scheda si compone di due cilindri, trig ed echo, con integrati dei componenti per convertire il segnale. Il cilindro nominato Trig genera un'onda meccanica nella frequenza degli ultrasuoni a 40 KHz, l'Echo invece lo riceve.

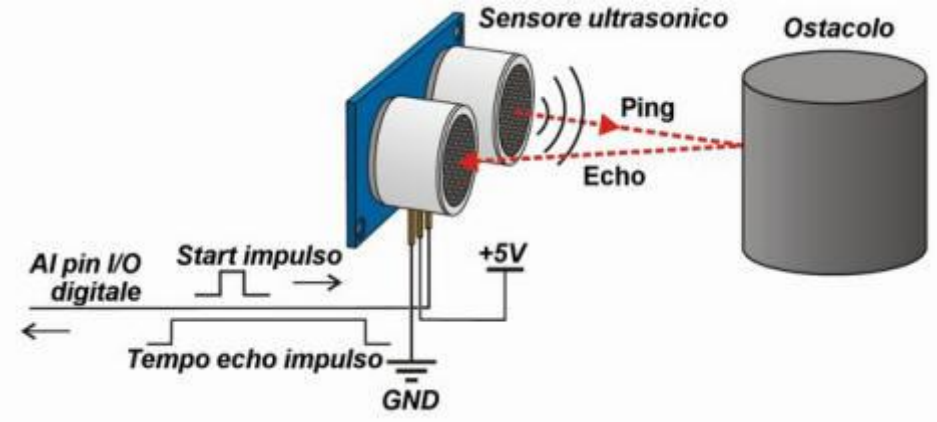

Figura 1.2. Sensore ad ultrasuoni

Il microcontrollore riceve la distanza ad ogni ciclo di funzionamento impostando il pin Trig su alto per $10\,\mu s$, il che attiva il sensore che emette otto impulsi ad ultrasuoni.

Quando l'eco degli impulsi, riflesso da un ostacolo, viene rilevato il sensore setta il pin Echo alto.

Gli 8 impulsi verranno ritrasmessi appena l'Echo si attenua, definendo il periodo di ciclo, vedi figura 1.3.

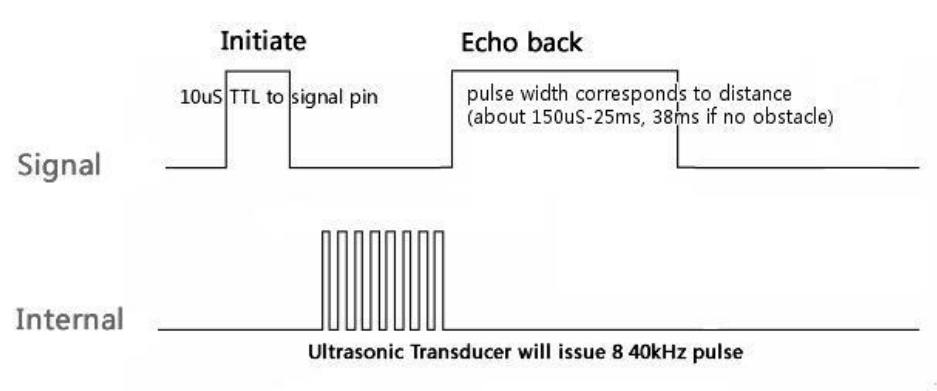

Figura 1.3. Ciclo di lettura del sensore ad ultrasuoni

La durata in cui il pin Echo rimane alto, corrisponde al tempo impiegato dal segnale a raggiungere l'ostacolo e tornare indietro.

Dunque, questo intervallo è due volte il tempo necessario a raggiungere l'ostacolo. Noto ciò, il microcontrollore può trarre la distanza fra il sensore e l'ostacolo tramite la formula che lega la velocità con lo spazio ed il tempo:

$$v = \frac{d}{t} \tag{1.1}$$

dove:

- *v* è la velocità del suono,
- *d* è la distanza,
- *t* è l'intervallo di tempo impiegato in μs .

La velocità del suono *v* viene calcolata in base alla temperatura T secondo la formula:

$$v = 331.45 + (0.62 \cdot \mathrm{T}) \quad \left[\frac{m}{s}\right] \tag{1.2}$$

Supponiamo che la temperatura sia T=20°C, quindi la velocità del suono è $v = 343.85 \ \left[\frac{m}{s}\right]$, che corrisponde a $0.0343 \ \left[\frac{m}{\mu s}\right]$.

Si ottiene quindi che $s = 0.0343 \cdot t \ \ [cm]$.

Essendo *t* il tempo impiegato dal segnale per andare verso l'ostacolo e tornare, è necessario considerare *t*/2 quindi

$$s = 0.0343 \cdot \frac{t}{2} = 0.0171 \cdot t \quad [cm] \tag{1.3}$$

Il sensore HC-SR04 si presenta come in foto:

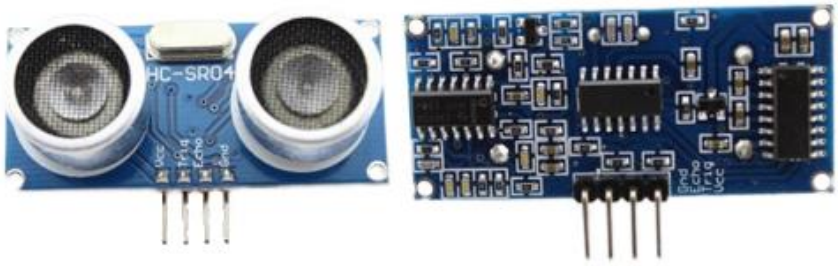

Figura 1.4. Sensore HC-SR04

(Coppola & Marchetta, 2016), (Gandolfo, Sensore ad Ultrasuoni HC-SR04, 2022)

1.2.2. DHT11

Il prossimo sensore di cui parliamo è il sensore di temperatura e umidità ambientale chiamato DHT11.

Il misuratore DHT11 come anche il DHT22[2], rilevano l'umidità relativa, cioè la presenza di vapore acqueo presente nell' aria rispetto al punto di saturazione, misurando la resistenza elettrica fra due elettrodi grazie ad un componente sensibile all'umidità.

Un substrato in grado di trattenere il vapore acqueo è posto sotto i due elettrodi, questo componente assorbe l'acqua rilasciando ioni che incrementano la conduttività fra i due elettrodi.

La variazione di resistenza è proporzionale all' umidità relativa, quindi alla quantità di vapore che comincia a condensarsi sulle superfici.

Per quanto riguarda la temperatura il DHT11 effettua la misurazione grazie ad un sensore NTC (termistore) montato su di esso.

Posto sul retro del sensore vi è un circuito integrato (IC) che converte le misure di resistenze in umidità relativa e controlla la trasmissione dati con l'esterno.

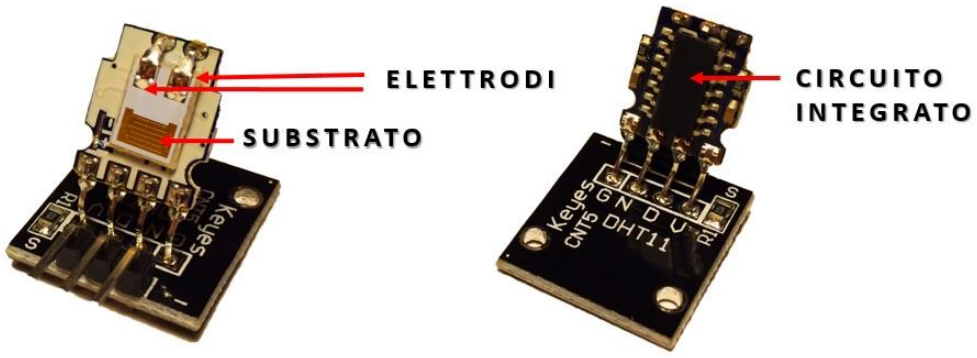

Figura 1.5. Visione frontale e posteriore del DHT11.

[2] https://www.adrirobot.it/dht22-am2302-sensore-temperatura-umidita/

Inoltre, sulla basetta del sensore di temperatura e umidità è posta una resistenza di pull-up di $10K\Omega$.

(Coppola & Marchetta, 2016), (Gandolfo, DHT11 - Sensore temperatura-Umidità, 2022)

1.2.3. Display OLED IIC

I display OLED (Organic Light Emitting Diode) sono schermi caratterizzati da un'architettura autoemissiva, in cui ogni pixel dello schermo non necessita di retroilluminazione. Sono formati da vari strati sovrapposti, un primo strato trasparente che ha funzioni protettive, un secondo strato a conduzione che funge da anodo seguito da tre strati di materiale organico, ad esempio un polimero conduttivo in grado di emettere luce bianca, rossa, verde e blu se opportunamente drogato ed infine uno strato riflettente con la funzione di catodo.

Il microcontrollore per interfacciare il display utilizza un'interfaccia seriale IIC che richiede due pin di dati. IIC è un'interfaccia di tipo sincrona, quindi il clock di ricezione è fisicamente collegato al clock di invio tramite un bus di tipo master-slave. I due pin necessari per la comunicazione sono:

- SDA (serial data): per i dati
- SCL (serial clock): per il clock

Ad essi sono collegati i resistori di pull-up.

Nel nostro progetto usiamo un display OLED da 0.96" ad una risoluzione di 128x64 pixel con interfaccia I2C e driver SSD1306.

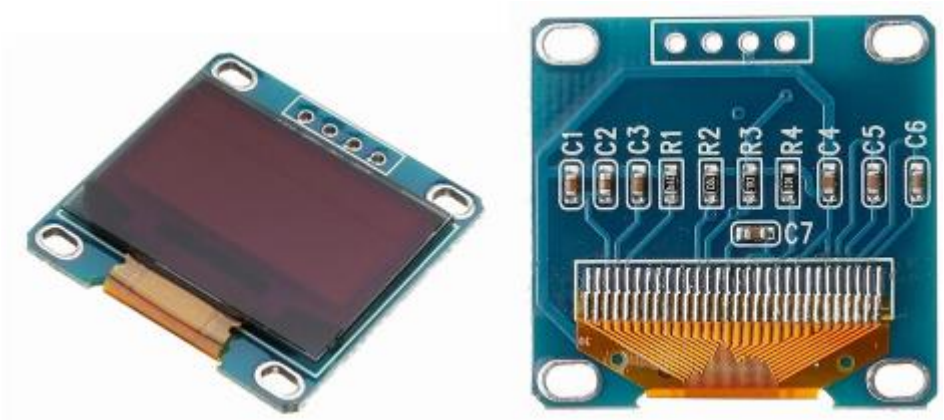

Figura 1.6. Visione frontale e posteriore di un display OLED 0.96".

(Gandolfo, Display OLED 0.96" 128x64, 2022), (Wikipedia, 2013), (Patti, 2024).

1.2.4. Diodo LED RGB

I led (light emitting diodes) sono diodi in grado di produrre fotoni attraverso un fenomeno di emissione spontanea se attraversati da corrente.

Il diodo è un componente elettronico passivo a due terminali, chiamati anodo e catodo, formato da una giunzione pn.

La giunzione pn è una fetta di materiale di tipo-n, solitamente silicio con una certa concentrazione di atomi donatori N_D, la cui metà viene successivamente drogata con una certa concentrazione di atomi di tipo accettori N_A tale che $N_A > N_D$.

In questo modo avremo una regione di tipo-p (ricca di lacune) ed una regione di tipo-n (ricca di elettroni), separati da una giunzione metallurgica ai quali colleghiamo rispettivamente anodo e catodo.

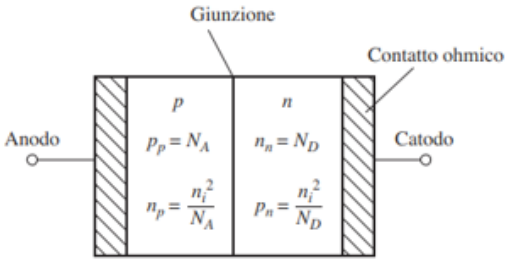

Figura 1.7. Struttura di un diodo a giunzione pn

Ricordando il fenomeno della diffusione, sappiamo che le lacune andranno dalla regione a maggior concentrazione, cioè la regione p, verso quella a minor concentrazione, la n. Dualmente, gli elettroni andranno dalla zona a maggior concentrazione, la n, verso la zona a minor densità di elettroni, quindi la p. Se la corrente di diffusione agisse indisturbata avremmo una distribuzione di carica uniforma; ma, la corrente di diffusione è controbilanciata dalla corrente di deriva. Quando nel processo di diffusione gli elettroni dalla zona n vanno verso la zona p lasciano degli accettori fissi vicino alla giunzione metallurgica, sulla zona n; allo stesso modo, le lacune lasciano delle cariche negative fisse nella zona p, vicino alla giunzione metallurgia, a causa del processo di diffusione. Abbiamo quindi due correnti: la prima generata per costruzione, la seconda generata come conseguenza

della prima. Si crea, dunque, una regione intorno alla giunzione metallurgica svuotata da cariche mobili, chiamata **regione di carica spaziale**. Vedi figura 1.8.

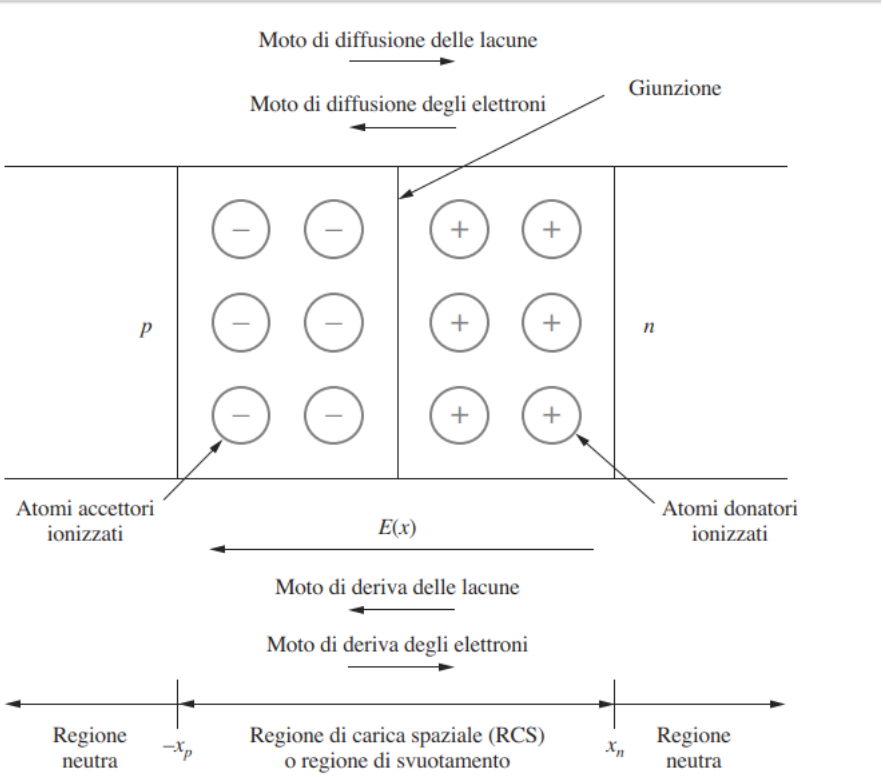

Figura 1.8. Illustrazione delle regioni del diodo

Nel momento in cui viene applicato un potenziale in polarizzazione diretta quindi alla zona p, viene applicato un potenziale positivo e alla zona n un potenziale negativo, gli elettroni della regione n del diodo respinti dal potenziale negativo entrano nella regione di svuotamento e trasformano le cariche fisse positive della regione di svuotamento in cariche libere negative.

Allo stesso modo le lacune della regione p, entrando nella regione di svuotamento, trasformano le cariche fisse negative in cariche libere negative.

Di conseguenza, la regione di carica spaziale si restringe fino a scomparire del tutto; in quel momento gli elettroni della banda di conduzione e le lacune della banda di valenza si ricombinano rilasciando fotoni con energia pari all' Energia di Gap che caratterizza la sua lunghezza d' onda quindi il colore dei fotoni liberati.

I led RGB usati nel progetto, sono costituiti da tre diodi, quindi da tre giunzioni pn, controllabili singolarmente per poter permettere di regolare la luminosità delle tre componenti e creare quindi ogni combinazione di colore. (Jaeger & Blalock, 2018).

2. Fondamenti dell'Internet of Things

L' Internet of Things ha trasformato qualsiasi dispositivo in un dispositivo in grado di connettersi ad una rete e condividere informazioni; vedremo in questo capitolo cos'è e come è strutturato l'IoT.

2.1. Definizione e importanza

L' internet of things, anche chiamato IoT, possiede due termini chiave: **internet** e **things**. Il concetto di internet, quindi di connessione alla rete, viene implementato e reso accessibile alle *cose*; cioè a tutti quegli oggetti che ogni giorno utilizziamo e che possono essere resi **smart**.

Questi oggetti possono raccogliere dati dall' ambiente tramite sensori, svolgere funzioni meccaniche grazie ad attuatori o trasmettere informazioni tramite ricetrasmettitori; solitamente possiedono memoria limitata con energia e potenza di calcolo esigua.

Il termine Internet of Things si sviluppò gradualmente dall' inizio di internet con le reti ArpaNet negli anni '80 fino al 1999 quando Kevin Ashton, un ingegnere inglese co-fondatore degli "Auto-ID laboratory" dell'MIT, durante una conferenza per la Procter&Gamble[3], usò per la prima volta il termine Internet of Things.

Nel 2010 si diffonde grazie all' avvento della domotica e dei sistemi multimediali e digitali delle automobili.

[3] https://it.pg.com/

Ad oggi l'IoT rappresenta l'innovazione tecnologica più significativa ed importante degli ultimi anni grazie al suo utilizzo in vari ambiti come:

- Automazione: grazie ai sistemi IoT i processi industriali, ad esempio, possono essere resi autonomi diminuendo e addirittura tal volta eliminare l'intervento umano.
- Monitoraggio in tempo reale: grazie a sensori ed attuatori è possibile rilevare errori e addirittura correggerli quando possibile.
- Miglioramento della qualità della vita: l'IoT integrata all'assistenza sanitaria o alla domotica porta un grande miglioramento alla vita di ogni giorno
- Sicurezza: sensori di rilevamento di fumo e gas, sistemi di videosorveglianza e dispositivi di allarme rappresentano una componente significativa dell'IoT.

2.2. Architettura

L'Internet of Things può essere modellizzato da diversi livelli tecnologici:

1. Livello fisico: si compone di tutti i dispositivi che astraggono le informazioni dall' ambiente; ad esempio, sensori di temperatura, umidità, pressione, luce o gas.
   Per essere efficienti ed utili i dispositivi devono rispettare delle condizioni: una fra queste è il basso consumo energetico in quanto spesso alimentati con una batteria.
2. Livello di comunicazione: comprende tutti i protocolli di comunicazione e le infrastrutture di rete che permettono la trasmissione dei dati da e per il dispositivo IoT; ad esempio, verso l'edge computing.
3. Livello di elaborazione: questo livello si occupa del preprocessing e dello storage temporaneo dei dati.

4. Comprende le tecniche di elaborazione distribuita come l'edge computing, che si occupa di elaborare i dati vicino al dispositivo che fornisce i dati, ed il fog computing, che estende l'elaborazione ai nodi intermedi; in questo livello vi si trova anche il middleware che fornisce un'interfaccia fra i diversi livelli o componenti della rete.
5. Livello di archiviazione: responsabile della conservazione dei dati
6. Livello di astrazione: il livello gestisce l'aggregazione dei dati per semplificare l'interazione con essi.
7. Livello di servizio: fornisce servizi basati sui dati raccolti e analizzati
8. Strato applicativo: comprende strumenti per il controllo e la gestione dei dispositivi IoT.
9. Livello di collaborazione e processi: implementa processi aziendali nei settori che utilizzano dati generati dall'IoT; ad esempio, un'azienda che tiene sotto controllo i macchinari tramite sensori può effettuare manutenzione predittiva.

(Catania, 2024)

2.3. L'IoT nel settore automobilistico

In quest'epoca di innovazioni l'Internet of Things gioca un ruolo fondamentale nel settore automobilistico; l'IIoT, Industrial Internet of Things, offre molteplici potenzialità sia per il reparto produttivo che per il consumatore.

I vantaggi dell'IIoT nell' automotive sono:

- Monitoraggio in tempo reale: durante il processo di produzione dei veicoli dei sensori consentono di monitorare ogni passaggio del processo produttivo e prevenire guasti di fabbrica e inefficienze del veicolo.
- Manutenzione predittiva: attraverso la raccolta di dati tramite sensori e utilizzando metodi di analisi avanzate e modelli predittivi è possibile prevedere guasti e di conseguenza, effettuare manutenzione predittiva.

- Sicurezza negli impianti produttivi: grazie all'analisi dei macchinari e al costante monitoraggio di essi vengono previsti guasti o incidenti e migliorare quindi la sicurezza delle macchine e dei lavoratori.
- Assistenza alla guida: dati forniti da sensori di prossimità e telecamere forniscono un'assistenza alla guida ed in alcuni casi perfino raggiungere la guida autonoma dei veicoli.
- Monitoraggio del conducente: sensori all' interno dell'abitacolo permettono di riconoscere i segni di affaticamento del conducente.
- Integrazione con smart home: ad esempio, riconoscere quando si è nelle prossimità della propria abitazione ed effettuare azioni personalizzate come accendere l'aria climatizzata.
  (TaskSRL, 2023)

3. Implementazione del sistema

Come detto nell'introduzione di questa tesi, l'obbiettivo è quello di costruire un sensore di parcheggio basato su Arduino.

Vedremo in questo capitolo come implementarlo, sia in formato fisico che in formato digitale.

3.1. Specifiche del progetto

Il sistema è composto da un sensore di prossimità ad ultrasuoni che comunica con il microcontrollore, un Elegoo Uno R3, fornendo ogni 500ms la distanza fra il dispositivo e l'ostacolo.

Per interfacciarsi con l'utente il progetto prevede uno schermo OLED da 0.96" con una risoluzione di 128x64 pixel ed un led RGB per fornire un'indicazione visiva immediata della distanza.

All' accensione del sistema e ogni 500ms il sensore di parcheggio deve fornire, tramite il dispositivo di output scelto, la distanza fra il veicolo e l'ostacolo posto in prossimità del sensore ad ultrasuoni; contemporaneamente, quando il veicolo raggiunge una distanza minore di 20cm il led RGB assumerà il colore rosso; ad una distanza compresa fra 20cm e 30cm il led sarà di colore giallo; infine, ad una distanza maggiore di 30cm si colorerà di verde.

Per ricavare la distanza fra il dispositivo e l'ostacolo, oltre al sensore a ultrasuoni, ho utilizzato un sensore di temperatura per ricavare la T utile per la velocità del suono nell' aria.

3.2. Versione digitale

Per implementare il sistema in formato digitale è stato utilizzato il software Tinkercad, programma di simulazione circuitale prodotto da Autodesk.

Il software ci mette a disposizione vari componenti, come l'Arduino e molteplici sensori, utilizzabili nei progetti per simulare il circuito prima che venga fatto con l'hardware fisico.

I componenti utilizzati sono: un sensore di distanza ad ultrasuoni HC-SR04, un led RGB, tre resistori da 220 Ω, un sensore di temperatura TMP36 ed infine, un display con interfaccia I2C.

Preso un Arduino Uno R3, dal catalogo, insieme ad una breadboard ho collegato le righe dell'alimentazione della board ai pin 5V e GND dell'Arduino.

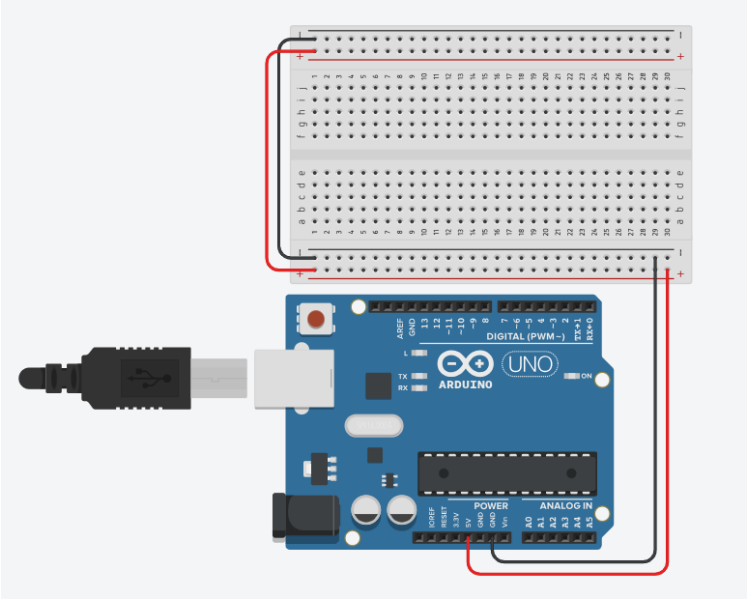

Figura 3.1. Arduino Uno R3 con breadboard

Una volta alimentata la breadboard, ho collegato i pin VCC e GND del sensore di prossimità a ultrasuoni HC-SR04 rispettivamente alla riga + e alla riga – dell'alimentazione; fatto ciò, ho connesso il pin TRIG del sensore al pin digitale numero 2 dell'Arduino e il pin ECHO al pin digitale numero 3.

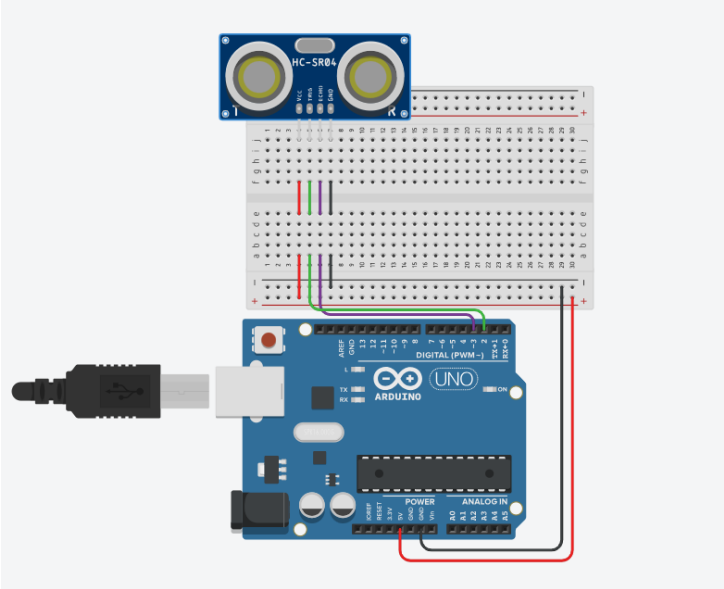

Figura 3.2. Sensore di prossimità ad ultrasuoni integrato alla breadboard

Provveduto al corretto collegamento del sensore di prossimità, ho connesso alla breadboard il sensore di temperatura TMP36.

Il dispositivo, l'unica opzione disponibile offerta dal catalogo di Tinkercad, è stato collegato con la seguente configurazione: il pin di alimentazione al + della breadboard, il pin GND alla riga – della breadboard ed il pin Vout al pin analogico A0 di Arduino.

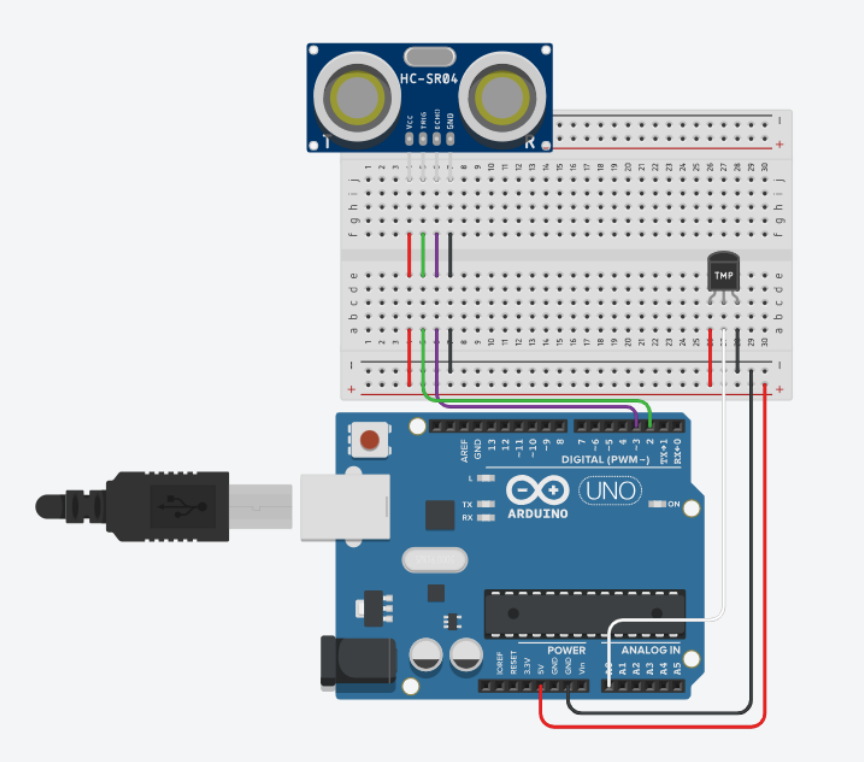

Figura 3.3. Sensore di temperatura integrato alla breadboard

Collegati alla board il sensore di prossimità e il sensore di temperatura connetto il dispositivo di output, un LCD con interfaccia I2C, alla breadboard.

Collego l'alimentazione VCC e GND del display al + e – della board; il display utilizza un'interfaccia I2C, dunque servono due pin per il collegamento SDA ed SCL da collegare rispettivamente ai pin SDA ed SCL dell'Arduino.

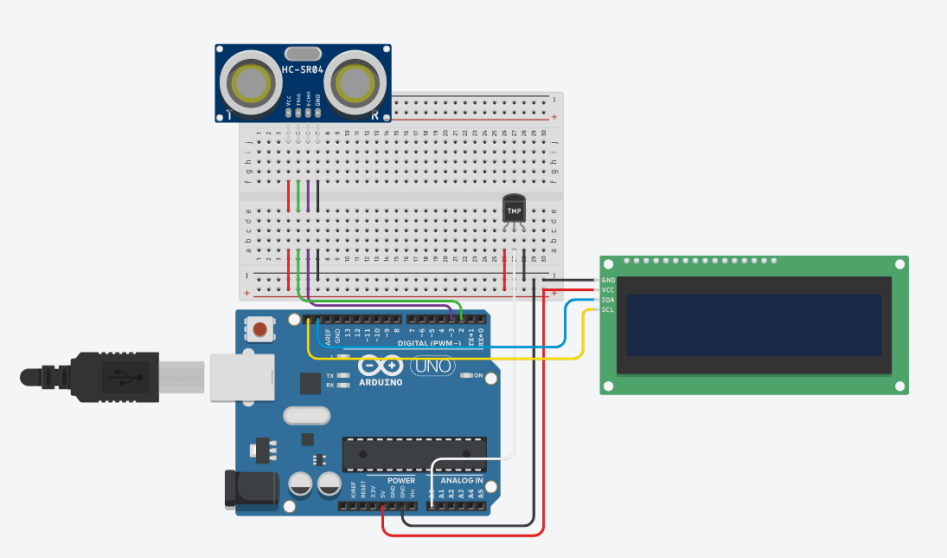

Figura 3.4. Schermo LCD (I2C) connesso alla board

Infine, per completare l'hardware del sistema, ho collegato i pin del led RGB alla board; trattandosi di tre diodi separati, ogni pin (tranne il GND) deve essere collegato ad una resistenza di 220 Ω in modo da limitare la corrente e non far bruciare il led. Quindi ho connesso il pin rosso, blu e verde, rispettivamente, ai pin digitali 5, 6 e 7 di Arduino.

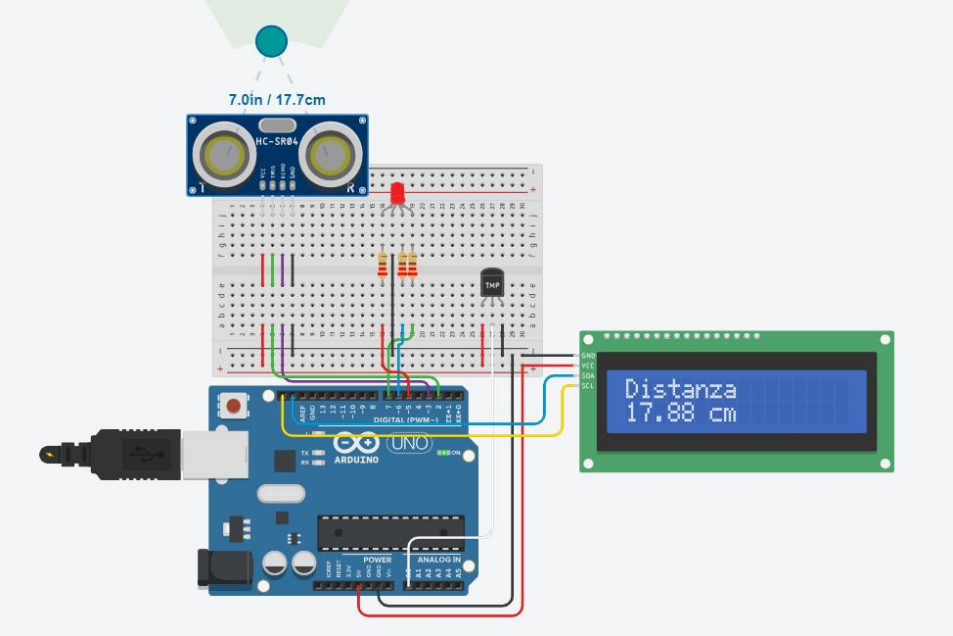

Figura 3.5. Versione digitale di un sensore di parcheggio

Aver costruito, prima del progetto fisico, una versione digitale di esso permette di poter risolvere errori evitando guasti ai dispositivi.

3.3. Versione fisica

Dopo aver implementato e testato il progetto in formato digitale grazie all' ambiente di simulazione Tinkercad, esamineremo i passaggi dell'implementazione fisica con i componenti hardware necessari alla realizzazione del sensore di parcheggio.

Presa una breadboard ed un Elegoo Uno R3, come detto precedentemente, equivalente all' Arduino Uno R3 si procede a collegare i cavi dell'alimentazione alla board; quindi, 5V alla riga + e GND alla riga -.

Alimentata la board si collega il sensore di prossimità ad ultrasuoni HC-SR04, con la stessa configurazione del progetto digitale e quindi il pin TRIG al pin digitale numero 2 di Arduino e il pin ECHO al pin numero 3.

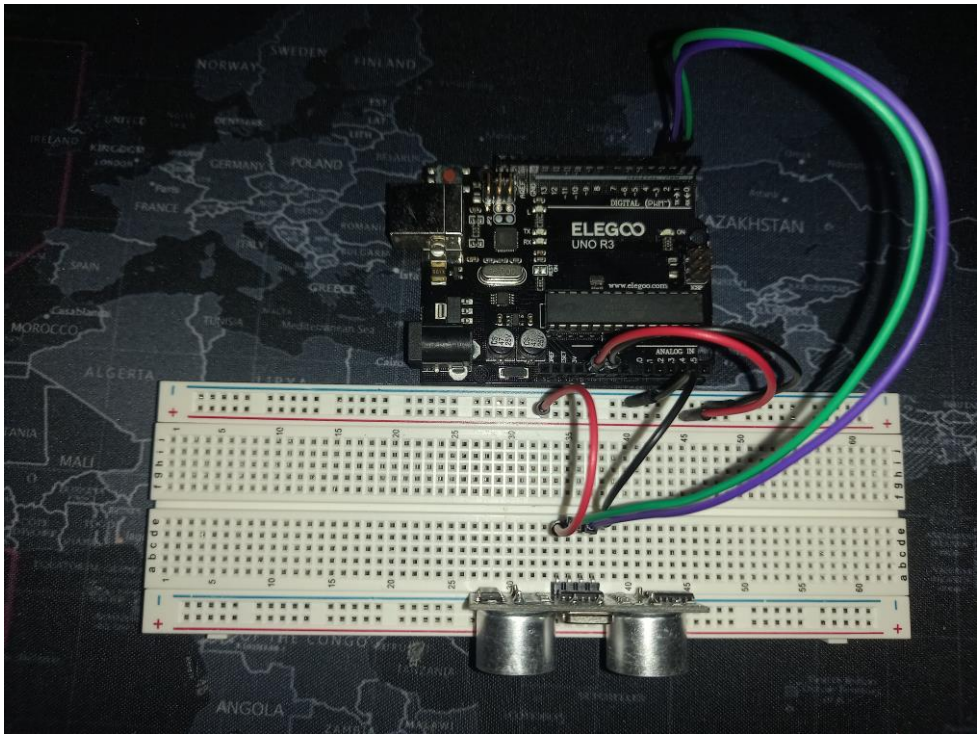

Figura 3.6. Board con sensore HC-SR04

Collegato il sensore di prossimità, connetto al sistema il sensore di temperatura DHT11; leggermente diverso dal sensore usato nel prototipo digitale, il DHT11 utilizza due librerie: DHT.h e DHT_U.h.

I tre pin del DHT11 che sono partendo da sinistra: il segnale, l'alimentazione ed il ground, vengono collegati partendo dal primo al pin digitale numero 4 ed i due rimanenti alla riga positiva e negativa dell'alimentazione.

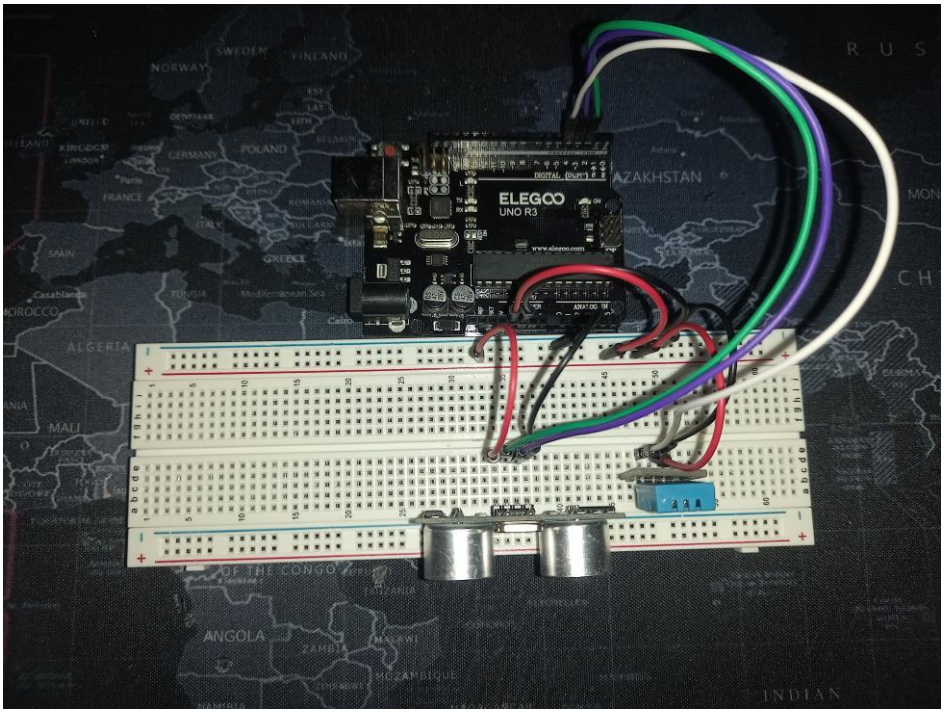

Figura 3.7. Board con HC-SR04 e DHT11

Integrati al sistema, il sensore di prossimità e quello di temperatura e verificato il corretto funzionamento dei dispositivi, viene collegato il display OLED con interfaccia I2C.

Alimentato il dispositivo connettendo Vcc e GND alle righe + e – della board; si collegano SCL ed SDA, i due pin necessari alla comunicazione I2C, ai pin SCL ed SDA di Elegoo.

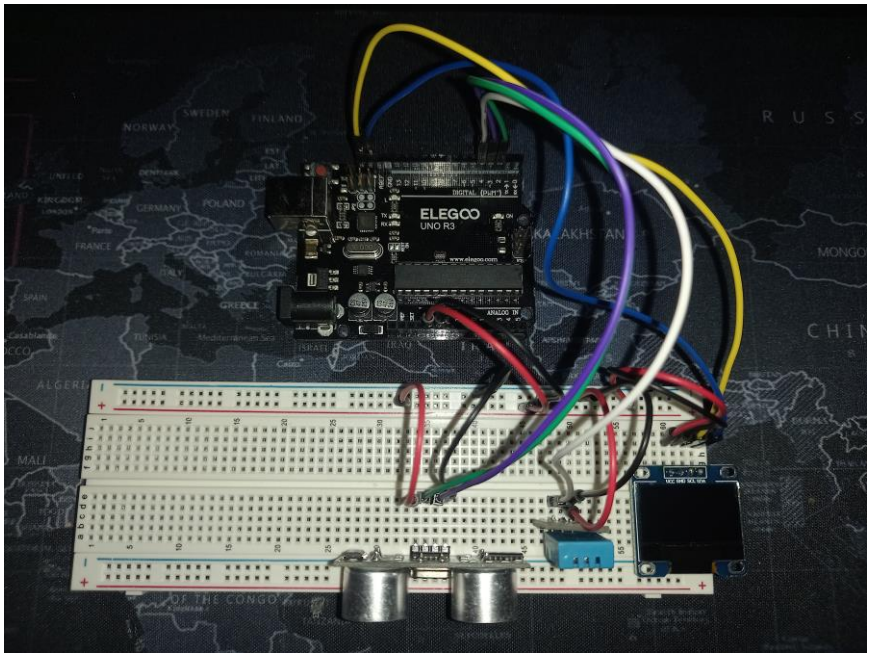

Figura 3.8. Collegamento alla board del display OLED

Infine, viene collegato il led RGB con il microcontrollore attraverso i suoi 4 terminali: Rosso, GND, Blu e Verde; connessi rispettivamente ai pin digitali 5, 6 e 7 per i colori e al ground per il GND.

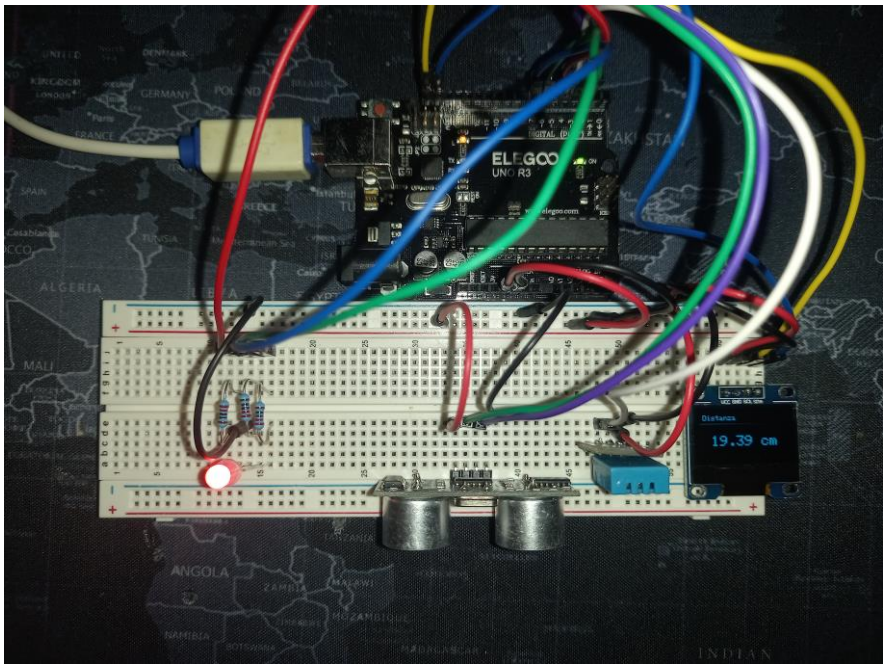

Figura 3.9. Sensore di parcheggio in formato fisico

3.4. Codice sorgente e funzionamento

In questo paragrafo vedremo come è implementato il codice analizzandone ogni sezione e verificandone, infine, il corretto funzionamento del sistema completo di hardware e software.

3.4.1. Inclusione delle librerie

La parte iniziale del codice, essenziale per il funzionamento delle funzioni utilizzate nel programma, è l'inclusione delle librerie, che sono:

- *Adafruit_SSD1306.h*: utilizzata per interfacciarsi con il display OLED che utilizza un driver SSD1306.
- *DHT_U.h*: questa libreria supporta le funzioni utilizzate per gestire la temperatura e l'umidità date dal sensore DHT11.

```
#include <Adafruit_SSD1306.h>
#include <DHT_U.h>
```

Figura 3.10. Inclusione delle librerie utilizzate nel software

3.4.2. Definizione delle costanti

Successivamente, per migliorare la leggibilità e l'ordine, troviamo le costanti:

- *OLED_I2C_ADDRESS, OLED_WIDTH, OLED_HEIGHT*: costanti definite rispettivamente per indicare l'indirizzo (0x3C in esadecimale), la larghezza (128) e l'altezza (64), del display OLED.
- *DHTPIN, DHTTYPE*: specificano la porta a cui è collegato il sensore DHT ed il tipo, nel nostro caso il pin è il pin digitale numero 4 ed il tipo è il DHT11.
- *pinTrigger, pinEcho*: i pin utilizzati dal sensore ad ultrasuoni, 2 per il pinTrig e 3 per il pinEcho.
- *pinRed, pinGreen, pinBlue*: indicano i pin a cui sono collegati i 3 diodi indipendenti del led RGB, 5 per il rosso, 6 per il verde e 7 per il blu.

```
#define OLED_I2C_ADDRESS 0x3C
#define OLED_WIDTH 128
#define OLED_HEIGHT 64
#define DHTPIN 4
#define DHTTYPE DHT11

const int pinTrigger=2;
const int pinEcho=3;
const int pinRed=5;
const int pinGreen=6;
const int pinBlue=7;
```

Figura 3.11. Definizione delle costanti utilizzate nel software

3.4.3. Definizione degli oggetti

In questa sezione vengono definiti due oggetti:

- *dht*: oggetto della classe DHT_Unified, rappresenta il sensore di temperatura e umidità DHT11. Il metodo della classe DHT_Unified prende come parametri il pin di Arduino a cui è collegato il sensore ed il tipo.
- *oled*: oggetto della classe *Adafruit_SSD1306*, rappresenta il display OLED e richiede come parametri della funzione la larghezza e l'altezza del dispositivo.

```
DHT_Unified dht(DHTPIN, DHTTYPE);
Adafruit_SSD1306 oled(OLED_WIDTH, OLED_HEIGHT);
```

Figura 3.12. Definizione degli oggetti dht ed oled utilizzati nel software

3.4.4. Funzione setup

Ogni programma Arduino è composto da due funzioni, la funzione setup e la funzione loop; la prima eseguita una sola volta all' esecuzione del programma, la seconda viene svolta ciclicamente all' infinito. La funzione setup è composta da:

- Inizializzazione del display OLED: tramite la funzione begin, definita in *Adafruit_SSD1306*, richiamata sull'oggetto *oled* creato precedentemente, inizializzo il display.

Dopodiché azzero il suo contenuto e tramite la funzione showHeader, definita da me, definisco l'intestazione che deve avere il display per poi visualizzarlo con la funzione display, anch'essa definita nella libreria *Adafruit_SSD1306*.

- Configurazione dei pin di Arduino: in questa fase vengono settate le modalità di interazione di ogni pin utilizzato dal programma.
    - pinTrigger: output, è il pin con in quale produciamo l'onda meccanica.
    - pinEcho: input, pin con cui captiamo gli ultrasuoni.
    - pinRed, pinGreen, pinBlue: vengono settati tutti ad output perché sono i pin con il quale comandiamo i tre diodi del led RGB.
- Inizializzazione del monitor Seriale: utilizzato per il debug e la verifica dei dati, impostato a 9600 baud[4].
- Inizializzazione del sensore di temperatura: tramite la funzione begin, definita nella libreria *DHT_U*, inizializzo il sensore di temperatura DHT11.

```
void setup() {
  if (!oled.begin(SSD1306_SWITCHCAPVCC, OLED_I2C_ADDRESS)) {
    while (true);
  }
  oled.clearDisplay();
  showHeader();
  oled.display();
  pinMode(pinTrigger, OUTPUT);
  pinMode(pinEcho, INPUT);
  pinMode(pinRed, OUTPUT);
  pinMode(pinGreen, OUTPUT);
  pinMode(pinBlue, OUTPUT);
  Serial.begin(9600);
  dht.begin();
}
```

Figura 3.13. Funzione setup del codice

[4] https://it.wikipedia.org/wiki/Baud

3.4.5. Funzione Loop

La funzione loop, il cuore del programma, inizia con la misurazione della distanza; il pinTrigger viene impostato ad un livello basso (LOW), per accertarsi che non ci siano ultrasuoni prodotti da cicli precedenti, per poi subito dopo impostare un livello alto (HIGH) per produrre un'onda meccanica.

Dopo $10\ \mu s$, il pinTrigger viene abbassato, spegnendo così la sorgente dell'onda. Successivamente tramite la funzione pulseIn ricaviamo il tempo impiegato dall'eco per ritornare. Noto il tempo di andata e ritorno, ricaviamo tramite la funzione dht.temperature().getEvent(), la temperatura dell' ambiente per poi utilizzarla nell'equazione 1.2.

Ricavata la velocità del suono, utilizzando le considerazioni già commentate al sottoparagrafo *§1.2.1*, determino la distanza dell'oggetto tramite l'equazione 1.3.

Il codice gestisce anche l'evenienza in cui il sensore di temperatura sia guasto e la lettura non vada a buon fine, in questo caso viene supposta una temperatura generica di 20 °C. La distanza viene poi visualizzata sul display OLED e, insieme alla temperatura e alla velocità del suono, anche sul monitor seriale.

Successivamente, in base alla distanza con l'ostacolo, viene attivato il led RGB con la funzione setLed, infine, viene inserito un ritardo di $500\ ms$ per il prossimo ciclo.

```
void loop() {
  digitalWrite(pinTrigger, LOW);
  digitalWrite(pinTrigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(pinTrigger, LOW);
  float durata = pulseIn(pinEcho, HIGH);
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  float x = 1 / 58.31;
  if (isnan(event.temperature)) {
    Serial.println(F("Error reading temperature!"));
  } else {
    float vs = (331.45 + (0.62 * event.temperature)) * 0.0001;
    x = vs / 2;
    Serial.println("Velocita' Suono: " + String(vs) + " T: " +
String(event.temperature) + " °C");
  }
  float distanza = durata * x;
  if (durata > 38000) {
    Serial.println("> 600 cm");
    showValue("> 600 cm");
    oled.display();
  } else {
    Serial.print("Distanza: ");
    Serial.print(distanza);
    Serial.println(" cm");
    showValue(String(distanza) + " cm");
    oled.display();
  }
  setLed(distanza);
  delay(500);
}
```

Figura 3.14. Funzione loop del codice

3.4.6. Funzioni varie

Per una maggiore leggibilità del codice ho definito le funzioni:

- showHeader(): configura l'header del display OLED.
- showValue(String value): permette di visualizzare il valore passato come parametro sul display subito sotto l'intestazione.
- printCenteredText(String text): utilizzata nella funzione showValue che calcola la dimensione del testo e lo posiziona al centro dello schermo.

- setLedColor(bool red, bool green, bool blue): imposta i pin del led RGB secondo i valori booleani passati come parametro.
- setLed(float distanza): imposta la configurazione di valori da dare ai valori booleani red, blue e green in funzione della distanza passata come parametro.

```
void showHeader() {
  oled.setTextSize(1);
  oled.setTextColor(WHITE);
  oled.setCursor(4, 4);
  oled.print("Distanza");
  oled.setCursor(112, 4);
  oled.drawLine(0, 15, 128, 15, WHITE);
}

void showValue(String value) {
  oled.fillRect(0, 16, 128, 48, BLACK);
  oled.setTextSize(2);
  oled.setCursor(OLED_WIDTH / 2, OLED_HEIGHT / 2 + 8);
  printCenteredText(value);
}

void printCenteredText(String text) {
  int16_t x = 0, y = 0;
  uint16_t w = 0, h = 0;
  int16_t cursorX = oled.getCursorX();
  int16_t cursorY = oled.getCursorY();
  oled.getTextBounds(text, 0, 0, &x, &y, &w, &h);
  oled.setCursor(cursorX - x - w / 2, cursorY - y - h / 2);
  oled.print(text);
}

void setLedColor(bool red, bool green, bool blue) {
  digitalWrite(pinRed, red);
  digitalWrite(pinGreen, green);
  digitalWrite(pinBlue, blue);
}

void setLed(float distanza) {
  if (distanza < 20) {
    setLedColor(HIGH, LOW, LOW);
  } else if (distanza >= 20 && distanza <= 30) {
    setLedColor(HIGH, HIGH, LOW);
  } else {
    setLedColor(LOW, HIGH, LOW);
  }
}
```

Figura 3.15. Funzioni personali utilizzati nel codice

Dopo aver implementato e caricato il codice nell'IDE di Arduino, effettuato l'upload del software sul microcontrollore e configurato l'hardware come descritto nel paragrafo §3.3, è stata eseguita la verifica per un corretto funzionamento del dispositivo, ottenendo un esito positivo.

4. Miglioramenti futuri del progetto

In questo capitolo attenzioneremo le possibili aree sottoponibili a miglioramenti, sia dal punto di vista hardware che software; al fine di migliorare le prestazioni e l'efficienza del dispositivo.

4.1. Miglioramenti hardware

Attualmente il sistema utilizza un sensore di temperatura DHT11 ed un sensore di prossimità ad ultrasuoni. I dati che ritornano i due dispositivi possono essere più accurati grazie al miglioramento dei sensori. Per esempio:

- DHT22 o BME280: entrambi sono rilevatori di temperatura e umidità, ma offrono una maggiore precisione rispetto al DHT11.
- VL53L0X: sensore di distanza laser, spesso impiegato in applicazioni impegnative come il volo dei droni o nella robotica.

Insieme all'upgrade dei sensori che sono già stati utilizzati nel sistema, si può procedere all' aggiunta di sensori aggiuntivi; come:

- Sensore di movimento: per attivare il sistema solo quando necessario, migliorando così l'efficienza energetica.
- Lidar: emette migliaia di impulsi laser al secondo per misurare le distanze e mappare lo spazio che lo circonda creando una rappresentazione 3D dell' ambiente.
- Telecamere: possono giocare un ruolo fondamentale nella guida assistita. In grado di rilevare ostacoli e leggere segnali stradali possono facilitare la guida ed evitare incidenti.
- GPS: combinato ad un'interfaccia che permetta sia l'input che l'output, trasformerebbe il sensore di parcheggio in un sistema satellitare completo con tracciamento della posizione ed indicazioni stradali.

- ESP8266: permetterebbe al dispositivo di integrarsi in una rete IoT e trasmettere i dati relativi al traffico o al veicolo.

4.2. Miglioramenti software

Il codice utilizzato nel progetto può essere migliorato implementando diverse modalità di esecuzione:

- Modalità risparmio energetico: aumentando il delay contenuto alla fine della funzione loop del programma possiamo modificare la frequenza con cui leggiamo la distanza quindi abbassare i consumi energetici.
- Modalità alte prestazioni: in modo analogo, abbassando il delay aumentiamo la frequenza di rilevazione della distanza ed avere un sistema molto reattivo, veloce.

Integrando al sistema l'hardware descritto nel paragrafo precedente, algoritmi di machine learning come il riconoscimento degli ostacoli o la predizione di essi, possono aumentare l'assistenza alla guida.

Gli algoritmi di riconoscimento degli ostacoli: come le *Reti neurali convoluzionali* (CNN[5]) spesso usate in bioinformatica, permettono di classificare gli oggetti in real-time e di prevederne il comportamento grazie a dei modelli matematici (MPC[6]).

Infine, integrando al sistema un modulo wi-fi (ESP8266[7]), è possibile connetterlo ad una rete IoT e abilitare diverse funzionalità, quali: condivisione dei dati col cloud, controllo remoto o perfino azioni automatizzate.

[5] https://it.wikipedia.org/wiki/Rete_neurale_convoluzionale
[6] https://en.wikipedia.org/wiki/Model_predictive_control
[7] https://it.wikipedia.org/wiki/ESP8266

Conclusioni

L’ Internet of Things è una tecnologia in continua espansione e sta raggiungendo quasi tutti gli oggetti di uso comune, dalla borraccia smart fino all’ automotive. Diventata famosa per la domotica, con Amazon Echo e Google Home, l’IoT sta acquisendo sempre più popolarità grazie a nuovi dispositivi e protocolli di comunicazione.

Lo scopo di questo articolo scientifico è stato quello di costruire con tecnologie accessibili a tutti un dispositivo integrabile in una rete IoT, in grado di rilevare ostacoli nelle vicinanze di un veicolo e di migliorare l’esperienza di guida.

Il sistema sviluppato, con un microcontrollore Elegoo Uno R3 e con tutti i vari sensori ed interfacce, è servito come base per progetti futuri applicabili all’automazione e alla sicurezza dei veicoli.

Nel primo capitolo abbiamo visto come gli aspetti tecnici dei sensori e dei loro protocolli siano di fondamentale importanza per un risultato efficiente e preciso.

Nel secondo capitolo si è discussi dell’Internet of Things e di come questo impatta nel settore automobilistico.

Il progetto sviluppato in questa tesi è un piccolo esempio di come l’ IoT impatta nell’ automotive mettendo in evidenza come un semplice dispositivo in grado di rilevare gli ostacoli possa essere implementato su qualsiasi tipo di vettura.

Nel terzo capitolo si è analizzato il sistema in ogni sua fase, dalla costruzione alla verifica del sistema compreso di hardware e software.

Infine, nel quarto capitolo abbiamo attenzionato come il dispositivo progettato possa essere migliorato sia dal punto di vista hardware, grazie a nuove tecnologie sempre più diffuse, che dal punto di vista software, grazie all’ avvento delle intelligenze artificiali.

I risultati ottenuti dimostrano come un semplice sistema, caratterizzato da costi esigui, possa migliorare l'interazione fra l'utente con il proprio veicolo e l' ambiente circostante; nel nostro caso come un sistema può essere utile e offrire supporto durante il parcheggio.

Ringraziamenti

Alla fine di questo elaborato mi sento in dovere di ringraziare tutti quelli che hanno contribuito in questi tre anni di carriera universitaria. In primis, desidero ringraziare la mia famiglia, che ha reso possibile tutto ciò. Grazie al vostro incoraggiamento incondizionato e alla vostra comprensione, ho potuto affrontare ogni sfida di questi tre anni. Vi sono grato per aver creduto in me anche quando io stesso non ci credevo.

Un ringraziamento speciale va a Giorgia, la cui pazienza e comprensione hanno avuto un impatto immenso durante questi anni di studio. Le tue parole di incoraggiamento e il tuo supporto sono stati fondamentali per superare ogni sfida di questo percorso. Grazie perché ci sei sempre stata, anche nei momenti più difficili.

Ai miei amici, che ci sono sempre stati nello sconforto e nella gioia, che mi hanno sempre appoggiato ed incoraggiato, grazie. Negli anni gli amici si incontrano e si perdono, funziona così, ma mai si dimenticano. Grazie per le serate di gioia e i pomeriggi di studio. Grazie a tutti.

Infine, i miei ringraziamenti sono rivolti a quei docenti che in questi tre anni hanno saputo trasmettere con passione, dedizione e preparazione l’ amore che essi stessi nutrono per le loro materie. Grazie al loro impegno e alla loro capacità di coinvolgere hanno creato ambienti stimolanti, capaci di lasciare a tutti gli studenti, me compreso, amore e passione per gli insegnamenti svolti.

Indice delle figure

Bibliografia

Arduino. (2024). *docs.arduino.cc*. Tratto da Arduino.cc: https://docs.arduino.cc/hardware/uno-rev3/

Catania, V. (2024). Smart Objects and the Internet of Things.

Coppola, A., & Marchetta, P. (2016). *Corso pratico di Arduino - Modulo Avanzato.* Bologna: area51.

Gandolfo, A. (2022). *DHT11 - Sensore temperatura-Umidità*. Tratto da Adrirobot.it: https://www.adrirobot.it/sensore_dht11

Gandolfo, A. (2022). *Display OLED 0.96" 128x64*. Tratto da Adrirobot.it: https://www.adrirobot.it/oled-096-128x64-pixel-interface-i2c/

Gandolfo, A. (2022). *Scheda Elegoo Uno R3.* Tratto da Adrirobot.it: https://www.adrirobot.it/scheda-uno-r3-smd-elegoo/

Gandolfo, A. (2022). *Sensore ad Ultrasuoni HC-SR04*. Tratto da Adrirobot.it: https://win.adrirobot.it/sonar/HC-SR04/Sensore_sonar_HC-SR04.htm

Jaeger, R., & Blalock, T. (2018). *Miicroelettronica* (5 ed.). McGraw-Hill Education.

Patti, D. (2024). Introduction to Microcontrollers - Communication Interfaces. Catania, Italia.

TaskSRL. (2023). *Le applicazioni dell'IoT nel settore automotive*. Tratto da TaskSRL: https://www.tasksrl.it/it/blog/le-applicazioni-delliot-nel-settore-automotive-n43#

Wikipedia. (2013). *OLED*. Tratto da Wikipedia.org: https://it.wikipedia.org/wiki/OLED