Laboratorio di making
Laurea magistrale in Informatica
Università di Bologna

Pull-up counter

Keran Jegasothy - 1007457

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Contents

1 Introduzione

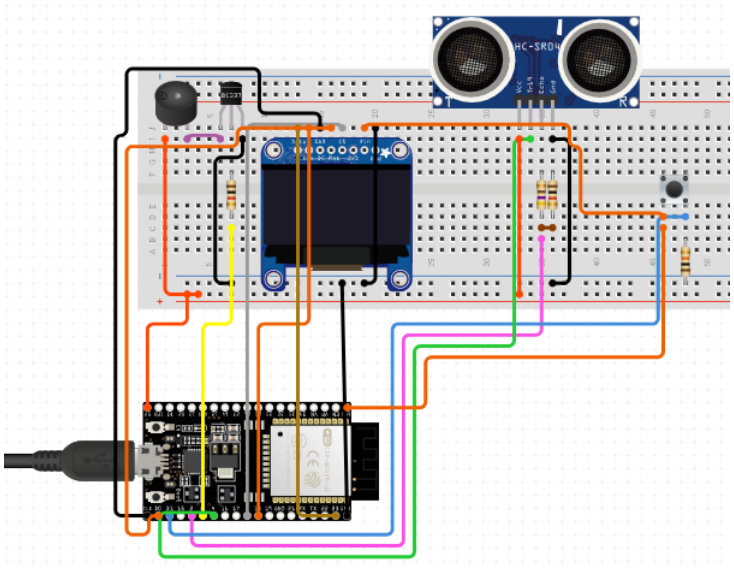
In questo progetto è stato realizzato un contatore di trazioni alla sbarra. Oltre a contare il numero di trazioni eseguiti dall'utente, avvia automaticamente la pausa tra una serie e l'altra ed avverte l'utente, tramite un segnale acustico, la fine della pausa . Inoltre, il numero di trazioni vengono inviati a un databese e a un bot telegram. Prima di iniziare le trazioni, oppure a seguito dell'invio dei dati, l'utente ha la possibilità di cambiare il tipo di trazione premendo il pulsante. Sono predefiniti 4 tipi di trazioni:

- **Normal**: le mani vengono posizionate a larghezza spalle o leggermente più larghe.
- **Chin Up**: si impugna la sbarra con la presa inversa a braccia distese, con la larghezza della presa che supera di poco l'ampiezza delle spalle.
- **Wide grip**: le mani sono posizionate almeno ad una larghezza di una volta e mezzo le spalle, fino ad arrivare a due volte.
- **Neutral**: si impugna la sbarra in modo da avere le mani parallele ed a larghezza delle spalle o leggermente più larghe.

Infine, l'utente ha la possibilità di gestire i dati salvati nel database tramite un'interfaccia web.

2 Circuito

In seguito è riportato il circuito del progetto:

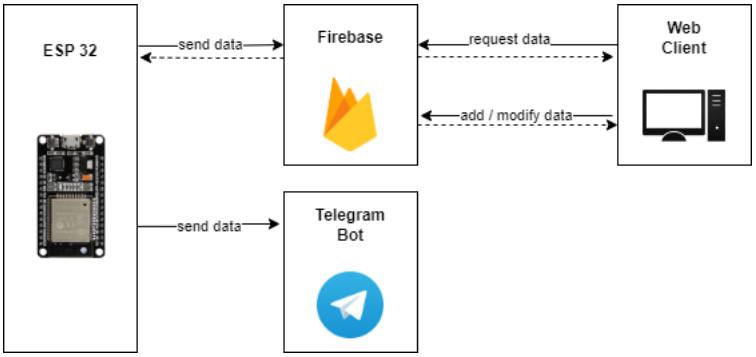


Quindi, le componenti utilizzati sono:

- **ESP32**: è un microcontrollore dotato di una CPU dual core a 32 bit.
- **HY-SRF05**: è un sensore ad ultrasuoni. Ha un campo di misura che si estende da 2cm a 4,5 metri con una precisione di 0,3cm.
- **Buzzer attivo**: è un sensore che usa un oscillatore interno che permette di emettere un tono a frequenza fissa se viene alimentato con una tensione continua.
- **Display OLED**: con risoluzione di 128x64 pixel. Utilizza un controller grafico SSD1306 e si collega tramite una interfaccia I2C.
- **Modulo con pulsante**: da 6x6mm normalmente aperto.

3 Architettura

In seguito è riportato un'immagine con l'architettura del progetto.

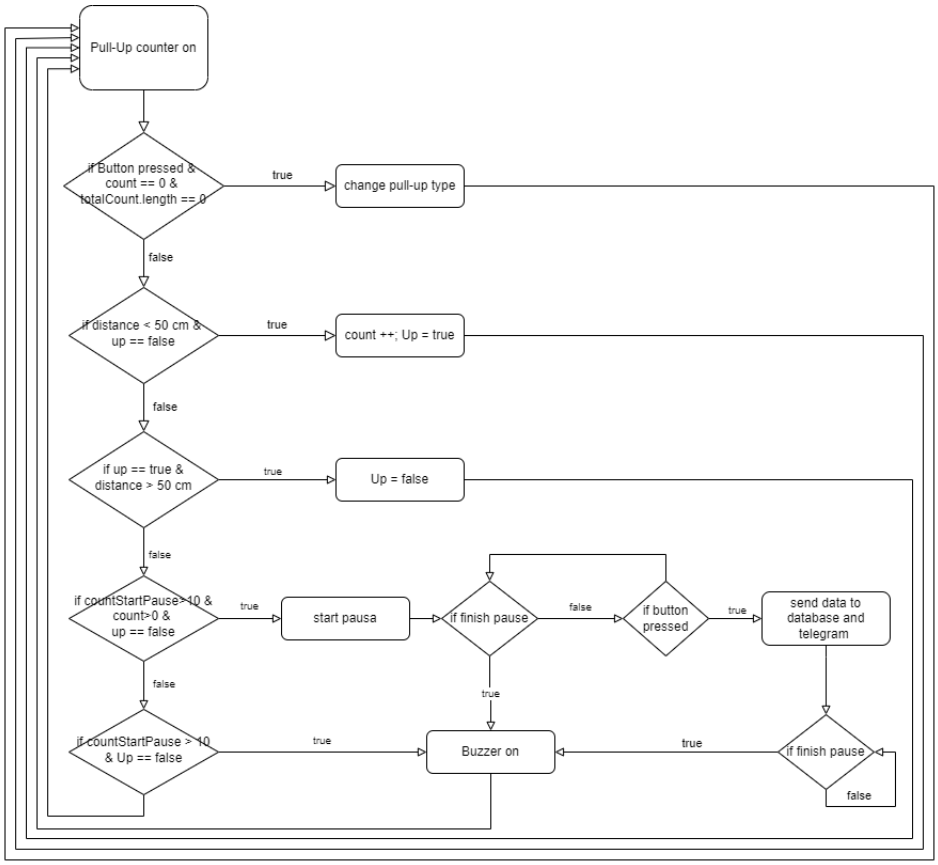


Quindi, gli elementi principali sono:

- **ESP32 + tutti i sensori** (vedi sez.2)
- **Firebase**: il database realtime di Firebase è un database ospitato su cloud. I dati vengono archiviati con il formato JSON e sincronizzati in tempo reale con ogni client connesso.
- **Bot telegram**: riceve i dati inivati dall'ESP32.
- **Web client**: l'utente ha la possibilita di interagire con il database tramite la pagina web.

4 ESP32

In seguito è riportato un diagramma di flusso che riporta tutte le operazioni che il *contatore di trazioni* può eseguito.



Riassumento, le principali operazioni sono:

- contare il numero di trazioni
- cambiare tipo di trazione
- gestire la pausa tra le serie (notificare l'utente dell'inizio e della fine della pausa)
- inviare i dati al database e al bot telegram

4.1 Cambiare tipo di trazione

L'utente ha la possibilità di cambiare il tipo di trazione prima di eseguire la prima trazione oppure a seguito dell'invio dei dati. Come detto nell'introduzione,

sono presenti 4 tipologie di trazione: normal, chin-up, wide grip, e neutral. In seguito è riportato il codice che permette di eseguito tale operazione.

```
if (last_button_state == HIGH && button_state == LOW && count == 0
    && totalCount.length() == 0) {
    selectPullUpType++;
    if (selectPullUpType > 3) {
      selectPullUpType = 0;
    }
    ...
}
```

4.2 Conteggio del numero di trazioni

```
if ((distance_cm < distanceUp)&&(Up == false)) {
    Up = true;
    count ++;
    countStartPause=0;
  } else if ((Up == true) && (distance_cm > distanceUp)){
    Up = false;
  }
```

Il codice soprariportato permette di incremenare il valore della variabile *count* ogni volta che l'utente esegue una trazione.

4.3 Pausa

```
if ((countStartPause>10)&& (count>0) && (Up == false)) {
    Serial.println("Pausa");
    digitalWrite (BUZZER_PIN, HIGH); //turn buzzer on
    delay(1000);
    digitalWrite (BUZZER_PIN, LOW);  //turn buzzer off

    ...

    int whileCount = 0;
    while (whileCount<61) {
      whileCount++;

      last_button_state = button_state;
      button_state = digitalRead(BUTTON_PIN);

      if (last_button_state == HIGH && button_state == LOW &&
    dataSend == false) {
        // send data
      }
      ...
    }
}
```

Il codice soprariportato permette di avviare la pausa tra una serie e l'altra. La pausa si avvia se l'utente non esegue nessuna trazione per 10 secondi di fila dopo aver eseguito almeno una serie. Per notificare l'utente dell'inzio della pausa, verrà attivato un segnale acustico. Durante la pausa l'utente ha la possibiltà di

inviare i dati sia al database che al bot telegram. Inviando i dati verrà azzerato il conteggio delle trazioni.

```
if ((countStartPause > 10)&& (Up == false)) {
      Serial.println("Suona");
      digitalWrite (BUZZER_PIN, HIGH); //turn buzzer on
      delay(1000);
      digitalWrite (BUZZER_PIN, LOW);  //turn buzzer off
  }

```

Il codice soprariportato viene richiamato alla fine della pausa ed emetterà dei segnali acustici fino a quando l'utente non eseguirà una trazione.

4.4 Invio dei dati

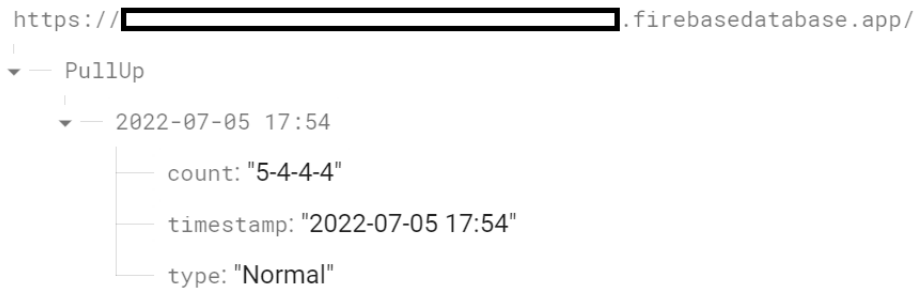
```
if (Firebase.ready() && signupOK){
    String topic1 = "PullUp/" + currentDate + "/count";
    if (Firebase.RTDB.setString(&fbdo, topic1, totalCount)){
        Serial.println("PASSED");
        Serial.println("PATH: " + fbdo.dataPath());
        Serial.println("TYPE: " + fbdo.dataType());
        dataSend = true;
    } else {
        Serial.println("FAILED");
        Serial.println("REASON: " + fbdo.errorReason());
    }
    ...
}
```

Il codice soprariportato permette di inviare i dati al realtime database (Firebase); mentre il codice sottoriportato permette di inivare un messaggio tramite il bot telegram.

```
bot.sendMessage(CHAT_ID,currentDate + " \n Serie: " + totalCount +
    " \n Type: " + typePullUp[selectPullUpType], "");
```

5 Database

Tutti i dati prodotti dal contatore di trazioni vengono salvati in un realtime database di Firebase.



I dati vengono salvati con il seguente formato JSON:

```
{
    "PullUp" : {
            timestamp_value : {
                    "count" : value,
                    "timestamp" : value,
                    "type" : value,
            },
            timestamp_value : {
                    "count" : value,
                    "timestamp" : value,
                    "type" : value,
            },
            ...
     }
}
```

6 Bot Telegram

É stato creato un bot telegram tramite *BotFather* e successivamente avviato con il comando */start*.
Ogni volta che il contatore di trazioni invia i dati, viene inviato un messaggio all'utente tramite il bot. Nel messaggio vengono riportati i seguenti dati:

- timestamp
- numero totale di trazioni (es. 5-4-4-3)
- tipo di trazione

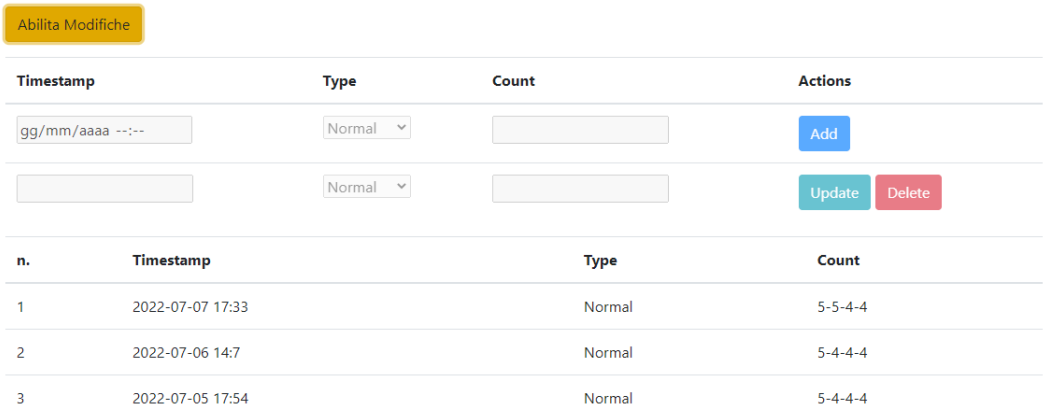
7 Web site

L'utente ha la possibilità di interagire con il database tramite un'interfaccia web. In particolare, può eseguire le seguenti operazioni:

- visualizzare tutti i dati salvati in tempo reale
- aggiungere nuovi dati

- modificare dati
- eliminare dati

Per poter modificare o eliminare un dato, l'utente deve inserire in input il timestamp relativo a tale dato.



Le tecnologie utilizzate per la realizzazione della pagina sono HTML e Javascript.

```
function GetAllDataRealtime() {
      const dbRef = ref(db, "PullUp");

      onValue(dbRef, (snapshot) => {
        var PullUps = [];
        snapshot.forEach(childSnapshot => {
          PullUps.push(childSnapshot.val());
        });

        AddAllItemsToTable(PullUps)
      });
    }
    window.onload = GetAllDataRealtime;
```

Il codice sopraripportato permette di estrarre i dati in tempo reale dal database.

```
function InsertData() {
     if (addTimestamp.value == "" || addType.value == "" ||
    addCount.value == "") {
        alert("Error: complete all fields")
    } else {
        var timestampWithoutT = addTimestamp.value.replace("T", " "
    );
        set(ref(db, "PullUp/" + timestampWithoutT), {
          timestamp: timestampWithoutT,
          type: addType.value,
          count: addCount.value,
        })
        .then(() => {
            alert("data stored succeccfully");
```

```
        })
        .catch((error) => {
            alert("unsuccessful, error " + error);
        });
    }
}
```

Il codice soprariportato permette di inserire nuovi dati nel databse.

```
function UpdateData() {
    if (modifyType.value == "") {
        update(ref(db, "PullUp/" + modifyTimestamp.value), {
          count: modifyCount.value
        })
        .then(() => {
            alert("data updated succeccfully");
         })
         .catch((error) => {
            alert("unsuccessful, error " + error);
         });
    } else if (modifyCount.value == "") {
        update(ref(db, "PullUp/" + modifyTimestamp.value), {
          type: modifyType.value,
        })
        .then(() => {
            alert("data updated succeccfully");
         })
        .catch((error) => {
            alert("unsuccessful, error " + error);
         });
    } else {
        update(ref(db, "PullUp/" + modifyTimestamp.value), {
          type: modifyType.value,
          count: modifyCount.value
        })
        .then(() => {
            alert("data updated succeccfully");
         })
         .catch((error) => {
            alert("unsuccessful, error " + error);
         });
    }
}
```

Il codice soprariportato permette di modificare un dato all'interno del database.

```
function deleteData() {
     remove(ref(db, "PullUp/" + modifyTimestamp.value))
    .then(() => {
         alert("data removed succeccfully");
    })
    .catch((error) => {
         alert("unsuccessful, error " + error);
    });
}
```

Il codice soprariportato permette di elimanare dei dati presenti nel database.

8 Risultato - immagini