Università degli studi di Bologna

Laboratorio di Making

A.A. 2017-2018

Relazione Progetto
Bassino

Autore
Cataldo Picciarelli

Introduzione

Bologna è una delle 14 città metropolitane d'Italia, 7° più grande d'Italia, con oltre 300 mila abitanti. Uno dei principali mezzi utilizzati per le strade del bolognese è la bici. Purtroppo la bicicletta, soprattutto perchè non richiede nessuna patente è un mezzo insicuro.
Nel 2017 gli incidenti con protagonisti i ciclisti sono aumentati del 15%, ed è un dato che tenderà ad aumentare. Questo ci fa intendere che la bicicletta è un mezzo molto datato che non ha ricevuto effettive migliorie tecniche che non la snaturassero (vedasi la bicicletta elettrica).
Il progetto Bassino (nome che tende a ricordare il ciclista Ivan Basso unito a quello di Arduino) propone una miglioria casalinga per aggiungere ulteriori feature senza stravolgerne il concetto, ovvero aggiungendo dei sistemi di sicurezza cercando di avere un approccio con la bicicletta tranquillo, tentando di far un minimo fronte ai mezzi inquinanti della quale la nostra società ne è ormai succube.
L'intero sistema verrà alimentato con un caricabatterie per cellulari a 5V.

Indice

Capitolo 1

I freni

Per poter implementare i freni sono stati utilizzati dei sensori ad ultrasioni, per la precisione degli HC-SR04. Sono i più utilizzati soprattutti per i progetti a basso costo, data la loro qualità prezzo. Di seguito le caratteristiche di questo sensore:

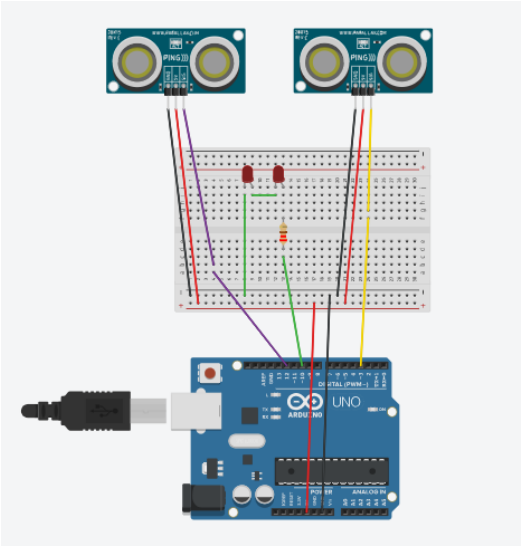
- Alimentazione: 5 V DC
- Corrente di operazione: 15mA
- Angolo effettivo: <15 °
- Range di distanza: 2 cm - 500 cm
- Risoluzione: 1 cm
- Frequenza ultrasonica: 40k Hz

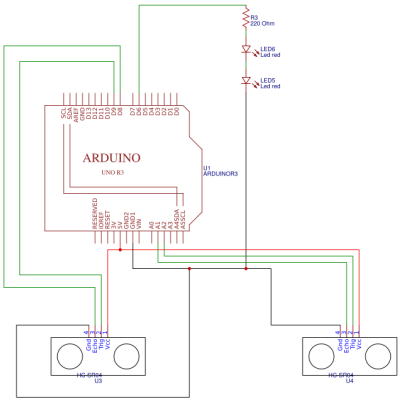
All'istante 0 il sensore manda un impulso. L'impulso verrà riflesso da un oggetto, facendo in modo che il segnale torni indietro.
Sfruttando questo principio del sensore, è stato optato il modo più semplice per poter simulare la frenata di una bici. Infatti, posizionando il sensore sensore nella parte interna del freno e impostando la distanza adatta, quando si effettuerà pressione sulla leva del freno, andando quindi ad attuare il sistema di frenaggio, il sensore riceverà una distanza minore ed accenderà dei led collegati in precedenza.
Per entrambi i freni sono stati utilizzati due sensori collegati in serie, in

modo da poter attivare gli stessi Led.
I led sono stati collegati in serie, in modo da evitare una variazione di corrente significativa sui led collegati in parallelo; ognuno dei quali ha una Vf di circa 1.8V (ma solo in questo caso, in quanto i led con colori diversi hanno anche una caduta di tensione diversa). La corrente che transita nei led sarà di 20mA.
Per la realizzazione sono stati utilizzati:

- 2 sensori ultrasuoni HC-SR04
- 2 led di colore rosso
- Una resistenza da 220 Ω

Figura 1.1: Schema elettrico dei freni





Problemi riscontrati:
I segnali inviati dal sensore variano a seconda della luce presente nell'ambiete. Ciò significa che, in un utilizzo notturno, potrebbe non funzionare correttamente.

Possibili migliorie:
Un possibile miglioria sarebbe quella di collegare dei cavi ai freni,evitando l'uso ingombrante del sensore ed i problemi che derivano da essi.

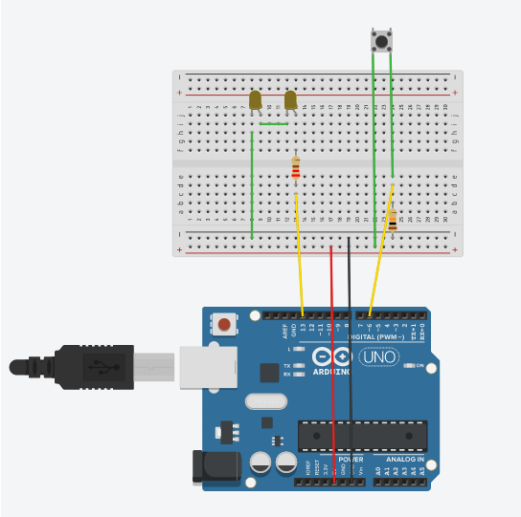
Capitolo 2

Gli indicatori di direzione

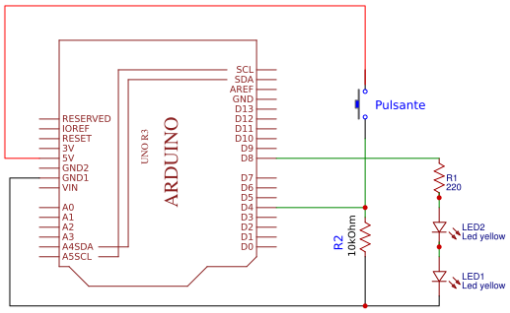
Per le frecce sono stati utilizzati 6 led di colore giallo, suddivisi 3 per la freccia sinistra e 3 per la destra, posizionati sulla breadboard vicino ai led dei freni.
Alla pressione dei pulsanti presenti sul manubrio si illuminano a tempo alternato, con un delay di 120 per i led accesi e 70 spenti. È stato inserito un controllo per il problema delle "4 frecce": nel caso in cui si dovessero premere entrambi i pulsanti, le frecce resteranno spente.
I pulsanti sono collegati su due pin differenti, lo stesso per i led. Inoltre i led solo collegati in serie, utilizzando una resistenza sulla primo led che riceve l'impulso.
Come nel caso analizzato in precedenza, i led sono stati collegati in serie, con una caduta di tensione di 1.9V. Anche in questo caso la corrente sarà di 20mA.
Per la realizzazione sono stati utilizzati:

- 4 led di colore giallo
- 2 resistenze da 220 Ω
- 2 resistenze da 10 KΩ
- 2 pulsanti

Figura 2.1: Schema elettrico delle frecce



Per semplicità, lo schema rappresenta il funzionamento di una freccia.



Problemi riscontrati:

Non sono saltati fuori problemi riguardante il funzionamento delle frecce. Unica pecca i pulsanti utilizzati, forse troppo piccoli e rigidi.

Capitolo 3

La temperatura

La temperatura verrà mostrata a grazie ad uno schermo LCD 16X2, collegato ad un sensore di temperatura, il DHT-11.
Questo sensore calcola sia la temperatura che l'umidità nell'aria.
Per il calcolo nello sketch di Arduino si è dovuto utilizzare la libreria 'DHT', disponibile gratuitamente e distribuita da Adfruit [1]. La libreria è stata creata per poter calcolare umidità e temperatura dei sensori Dht-11,Dht-12,Dht-21. Nel sketch, dopo aver richiamato la libreria, bisognerà definire una variabile assegnandone il tipo di DHT utilizzato (oltre che il pin al quale verrà collegato).
Scheda tecnica DHT-11:

- Tensione 5 V
- Corrente : 0.5mA (max 2.5mA)
- Corrente in stand-by:100uA (max 150uA)
- Temperatura di funzionamento: 0 / +50°C ±2°C
- Range di umidità:20-90 % RH ±5 % RH

Per la realizzazione sono stati utilizzati:

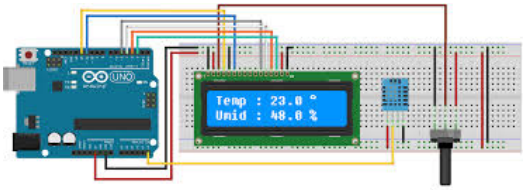
- Schermo lcd 16x2

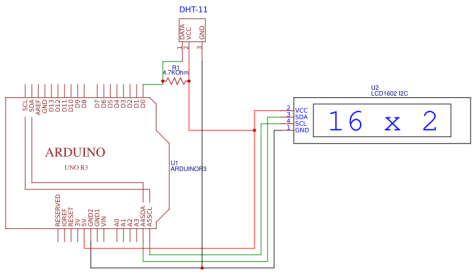
[1] Copyright (c) 2012 Adafruit Industries. Tutti i diritti sono riservati.

- Potenziometro 10 KΩ
- una resistenza da 4,7 KΩ
- Sensore di temperatura DHT-11

Inizialmente il sensore sarà in stand-by, in attesa del segnale di start. Quando il sensore riceve lo start, inizierà a mettere la linea dati a livelli bassi ed alti per circa 80uS per livello.Successivamente inizierà a trasmettere dati. Ogni bit inizia da un livello basso (50uS in questo caso), per poi passare a quello alto (il tempo dipenderà dal bit emesso, 27uS per 1 e 70uS per 0). Dopo aver emesso l'ultimo bit, il DHT-11 torna in stand-by. In totale vengono inviati 40 bit.

Figura 3.1: Schema elettrico del DHT-11





Problemi riscontrati:

Alucne operazioni dovute al calcolo dei km/h con il sensore capacitivo, entrano in contrasto con il sensore DHT-11, il quale restituisce valori errati in caso di velocità elevata.

Capitolo 4

Calcolo della distanza e km/h

Per il calcolo dei Km/h, non avendo a disposizione un reed switch, è stato utilizzato un metodo "fai da te".
In primo luogo sono state effettuate delle prove con dei sensori ad ultrasuoni, che hanno portato ad un esito negativo in quanto troppo instabile nel leggere il valore al passaggio di un oggetto ad alta velocità, ed inoltre restituiva valori non corretti.
Si è optato per un sensore capacitivo. Il sensore capacitivo riesce a rilevare la presenza di carica su un corpo nelle sue vicinanze e ne calcola la sua capacità complessiva. È stato effettuato un test con della carta stagnola per mostrare gi eventuali cambiamenti di stato del valore del sensore. Avendo riscontrato cambiamenti, si è supposta l'idea di poter attaccare un pezzo di carta stagnola sui raggi della bici e, a contatto col sensore dopo aver effettuato una rotazione , restituirne la velocità semplicemente dividendo la circonferenza della ruota per il tempo intercosso tra un "tocco" e l'altro, moltiplicando il tutto per 3.6 per ottenere i km/h. Il tempo su Arduino è dato in millisecondi. Dopo aver catturato il tempo si divide per mille per ottenere i secondi e dividere i metri.
In breve la formula per ottenere i Km/h è:

m/s*3.6

Nel nostro caso sarà:

((1.8)/(time))*3.6

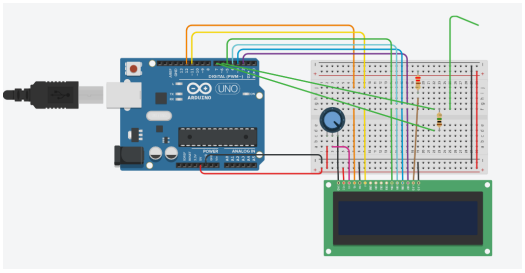
I risultati sembrerebbero fedeli ma sicuramente non affidabili.
Per i km percorsi è stata utilizzata una varibiale nella quale, ad ogni tocco del sensore, somma la circonferenza della ruota. Essendo nel nostro caso una circonferenza di 1 metro e 80, si sommerà 1 metro e 80 ogni volta che la ruota abbia compiuto un giro completo. Il valore mostrato a video sarà in Km, quindi il nostro metro e 80 equivale a 0.0018 Km.
Perciò per ottenere i Km tot percorsi si ha:

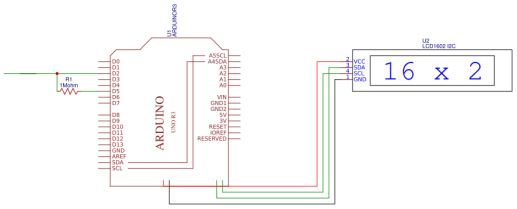
$$s=s+0.0018$$

Per la realizzazione sono stati utilizzati:

- Schermo lcd 16x2
- Potenziometro 10 KΩ
- Una resistenza da 1 MΩ
- Cavo scoperto

Figura 4.1: Schema elettrico del sensore capacitivo



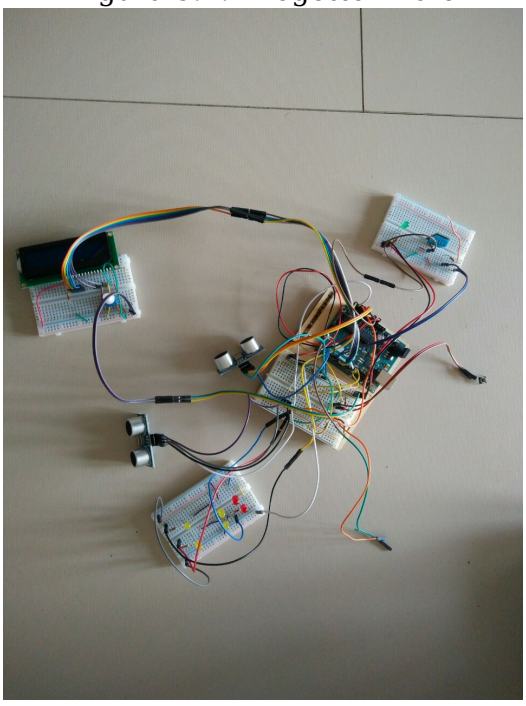


Problemi riscontrati:
Per quanto riguarda la divisione per il calcolo dei km/h, abbiamo bisogno di azzerare il tempo utilizzato una funzione chiamata ResetMills. Senza di essa il calcolo approssimativo non sarebbe stato possibile. I Km/h a video vengono mostrati con la virgola. I Km percorsi percorsi invece vengono approssimati: ad es. 0.0054 Km verranno mostrati come 0.01 Km.
Possibili migliorie:
Un reed switch risolverebbe il problema, migliorandone la precisione e tempi di risposta.

Conclusioni

Come già discusso in precedenza, Bassino è un progetto molto a basso costo , fatto con i materiali a disposizione. I risultati ottenuti non sono quelli che si desideravano, ma si prestano ad essere base per i miglioramenti futuri ed eventuali aggiunte.
La seguente foto mostra il progetto nella sua fase conclusiva, assemblando tutti i punti visti in precedenza.

Figura 5.1: Progetto finale



Sviluppi futuri:

Uno dei principali problemi è stato quello del "feedback" da parte di Arduino. Infatti a volte le risposte erano lente a causa dei vari processi in esecuzione. Sicuramente un Arduino con un processore più potente potrebbe migliorarne la situazione.

L'aggiunta di un GPS in caso di furto della bici, avendo quindi la possibilità di poterla localizzare.

Poter utilizzare un modo alternativo per alimentarla, evitando il carica batteria portabile.

Elementi scartati:

Non sono stati presi in considerazione delle eventuali luci di posizione: per le bici si utilizzano già dei catarifrangenti, acquistabili a basso costo.

Luci anabbaglianti:oltre al non aver a disposizione una lampadina adatta, si è ritenuto inutile in quanto una lampadina attaccata ad una dinamo risparmierebbe di costi e prestazioni.

Applicazioni web utilizzati:

Tinkercad (© 2018 Autodesk, Inc. All Rights Reserved.) e EasyEDA per gli schemi. ShareLaTex per la documentazione.