UNIVERSITÀ DEGLI STUDI DI CAMERINO

SCUOLA DI ATENEO IN SCIENZE E TECNOLOGIE

Corso di Laurea in Informatica

Classe L-31

DEFINIZIONE DI UN MODULO DIDATTICO PER L'INSEGNAMENTO DI ARDUINO CON CENNI DI PROGRAMMAZIONE VISUALE E RELATIVE APPLICAZIONI PROGETTUALI

Laureando

Gioele Giachè

Relatore

Prof. Fausto Marcantoni

Correlatore

Prof. Fausto Curzi

ANNO ACCADEMICO 2018/2019

In Memoria di Gianluca...

"La vita è così breve che non c'è tempo per litigi, per il rancore e per la guerra.
C'è solamente il tempo per amare e dura solamente un istante."
(Mark Twain)

RINGRAZIAMENTI

Raggiungere questo obbiettivo non è stato affatto facile, ci sono stati numerosi incidenti di percorso, personali e non, che hanno tardato il tutto. Il primo ostacolo che purtroppo non solo io ho dovuto affrontare è stato il terremoto del 2016. Ne abbiamo parlato davvero tanto, ma un evento così disastroso non segna semplicemente le strutture, credo che in un modo o nell'altro segni le menti e i cuori di tutti noi. Purtroppo non è semplicemente parte del passato, purtroppo è realtà, e come realtà vivida e arrogante, per quanto impossibile, dobbiamo cercare di gestirla nel migliore dei modi. E così rivolgo il mio primo ringraziamento alla parte amministrativa di Camerino e alla sua Università, in particolar modo all'ex Rettore Flavio Corradini e all'attuale Magnifico Rettore Claudio Pettinari, che, come numerose volte ho ripetuto, hanno saputo davvero prendere la situazione di petto, davvero alla perfezione.
In seguito ringrazio la Segreteria e i vari Professori che mi hanno spesso sostenuto e guidato in questi anni, grandi dispensatori di innumerevoli consigli. Ho inviato oltre 600 e-mail solamente dalla mia casella di posta elettronica universitaria, in sostanza 1 e-mail ogni due giorni da quando mi sono immatricolato, che dire, Unicam si è dimostrata davvero parte integrante della mia vita. Poi vorrei ringraziare la mia scuola superiore, il Liceo Scientifico "Galileo Galilei" di Macerata, che oltre ad avermi conferito un'importantissima preparazione di base per affrontare questo Corso di Laurea, mi ha permesso di effettuare lo stage e realizzare anche questa Tesi, supervisionata dal Relatore Fausto Marcantoni e dal Correlatore Fausto Curzi, professori di notevole affabilità per i quali nutro davvero grande stima. Inoltre colgo l'occasione per ringraziare di cuore tutti coloro che sono stati di vitale importanza per la realizzazione di questo progetto, dagli svariati tecnici e professori con cui mi sono confrontato, sia del mio Liceo che dell' IPSIA di Corridonia, ai familiari, agli amici, agli operatori online che con le loro idee e soluzioni mi hanno aiutato a concretizzare il tutto, tra i tanti Alessio, un grande amico e un grande artigiano che mi ha guidato per quanto riguarda il settore decorativo.

Ora vorrei ricordare coloro che a livello personale si sono dimostrati davvero importanti, dai compagni universitari con i quali ho spesso condiviso gioie e delusioni, affrontato disavventure, cenato insieme, agli amici, quelli di una vita, quelli che nonostante tutto sono rimasti, quelli incontrati da poco, e quelli che purtroppo mi hanno lasciato, fra tutti vorrei ricordare Gianluca Alessandrini, la cui simpatia e generosità difficilmente potrò dimenticare. Con l'occasione vorrei salutare Tina, sua madre, per me emblema di forza e ultimamente mio grande punto di riferimento.

Vorrei ringraziare ovviamente la mia famiglia, le mie due colonne portanti: i genitori, ma anche i fratelli, le nonne, i nonni che durante il mio percorso se ne sono andati, e le ultime due arrivate, le mie nipotine, che ci hanno portato tanto Amore, tanti sorrisi e hanno donato un po' di leggerezza a questa vita davvero caotica. Senza tutti loro sicuramente non avrei raggiunto questo traguardo. Ringrazio chiunque mi abbia sempre sostenuto, chiunque sia stato di ispirazione nella mia vita, chiunque mi abbia tradito ma si è riscattato, chiunque abbia optato per il dialogo anziché per il distacco, chiunque abbia perdonato i miei errori, chiunque mi abbia semplicemente dedicato del tempo. Vi assicuro che, citati o non citati, non vi dimenticherò mai.
Ringrazio Dio per avermi sempre illuminato il cammino.
Infine ringrazio il Gioele di ieri, il Gioele di oggi, due persone magari diverse, ma che comunque non hanno mai gettato la spugna, non hanno mai smesso di credere e di lottare.

La vita è incredibilmente breve e assurdamente imprevedibile, viviamola a pieno, sono abbastanza sicuro che sia meglio scegliere sbagliando che non scegliere per nulla!

Inoltre aggiungo, a mo' di postilla, la mia grande ammirazione per come Unicam si è prodigata per sostenere tutti gli studenti in questo periodo particolare, in cui il COVID-19 ha preso il sopravvento sulle nostre vite.

Vi saluto tutti, con la speranza di vederci ed abbracciarci presto.

DEFINIZIONE DI UN MODULO DIDATTICO PER L'INSEGNAMENTO DI ARDUINO CON CENNI DI PROGRAMMAZIONE VISUALE E RELATIVE APPLICAZIONI PROGETTUALI

Gioele Giachè

INDICE

INDICE DELLE FIGURE

PREMESSA

“In prima istanza, l’insegnante è un facilitatore che, grazie alla propria capacità empatica, sa costruire rapporti interpersonali utili e creare contesti di collaborazione che favoriscono lo sviluppo armonico della persona e un apprendimento sereno.” [a]

* * *

Nel mio percorso di studi, ho valutato numerose occupazioni future, rispondere alla domanda: “Cosa vuoi fare nella vita?”, per me è stato da sempre un tabù. Ultimamente però, probabilmente per mezzo delle varie esperienze che ho avuto la fortuna di acquisire, mi ha molto affascinato il mondo dell’insegnamento.
Durante lo stage al Liceo Scientifico “Galileo Galilei” di Macerata ho avuto modo di realizzare un modulo didattico per l’insegnamento di Arduino in relazione alla programmazione visuale. È da questa esperienza che io ho redatto questa tesi di Laurea. In ordine viene discusso il ruolo del professore e la teoria costruzionista, punto di lancio per discutere della programmazione visuale, in particolare vengono presentati ambienti di sviluppo come Scratch e MIT App Inventor. Successivamente viene introdotta la piattaforma hardware Arduino e dopo aver trattato sia il concetto della programmazione visuale sia Arduino, viene proposto il binomio S4A, ovvero Scratch for Arduino, il cui scopo è quello di programmare questa piattaforma hardware attraverso Scratch. In conclusione, viene realizzato il progetto CARA (“Car Arduino”), ovvero l’implementazione di una RC Car Arduino gestita attraverso un’applicazione realizzata con MIT App Inventor.

Non so se insegnare sarà davvero il mio futuro, sarà il tempo a dirlo, ma devo ammettere che l’idea non mi dispiace!

1. INTRODUZIONE

L'informatica nei licei è una materia che è stata introdotta relativamente di recente. Infatti in seguito alla Riforma Gelmini del 2008 per quanto riguarda il liceo scientifico, inizia lentamente ad esserci la possibilità di scegliere l'opzione "Scienze Applicate", eliminando ufficialmente il PNI, "Piano Nazionale Informatica", facendo così diventare l'informatica una materia di studio a tutti gli effetti [b]. Viste le poche ore settimanali e visto il quantitativo di materiale che l'informatica ci propone, non è semplice definire un programma quinquennale che vada anche solamente ad accennare ogni aspetto della disciplina. Nella mia esperienza scolastica l'apprendimento di Arduino, una piattaforma hardware programmabile che permette di "concretizzare" righe di codice, non è mai stato messo ufficialmente in relazione con questa materia, motivo per il quale ho deciso di realizzare questa tesi al fine di integrare il programma di studi informatico.

1.1 CONTESTO E MOTIVAZIONI DI STUDIO

L'obiettivo di questo studio è quello di apportare un contributo all'insegnamento dell'informatica all'interno delle scuole secondarie di secondo grado, in particolare all'interno dei licei. Questa tesi si preoccupa di strutturare un percorso didattico per l'apprendimento più o meno complesso di Arduino, utilizzando anche il concetto della programmazione visuale. Si parte da un'introduzione che analizza il ruolo del professore e la teoria di apprendimento sulla quale si basa l'approccio grafico per la risoluzione di problemi, fino a giungere alla realizzazione concreta di un progetto funzionante, con dettagli anche burocratici. La strategia utilizzata è quella di alternare esempi pratici di crescente difficoltà alla spiegazione teorica del concetto.

1.1 OBIETTIVI DELLA TESI

Come già preannunciato lo scopo è quello di portare il lettore ad una consapevolezza abbastanza dettagliata e sempre crescente degli argomenti trattati, mettendo in relazione diversi aspetti che spesso si tende a dividere: teoria e pratica, software e hardware.

1.3 I CONTRIBUTI PIÙ SIGNIFICATIVI

I due contributi più importanti di questa tesi sono da una parte la realizzazione di un documento sintetico e ben strutturato sullo studio base di argomenti quali la programmazione visuale e Arduino, dall'altra l'applicazione pratica degli stessi, quindi sono presenti diversi esempi illustrativi che culmineranno con la costruzione di un progetto molto più complesso.

1.4 I PROBLEMI AFFRONTATI

Durante la realizzazione del *Progetto CARA*, la grande maggioranza dei problemi che si sono verificati erano inerenti alla materia di microelettronica, per lo più legati a malfunzionamenti hardware della piattaforma Arduino, che hanno necessitato interventi di microsaldature, sostituzione di componenti quali condensatori, transistor e altri sensori, l'utilizzo di strumenti quali multimetro, generatori di corrente, etc. Inoltre non sono mancati anche problemi di natura burocratica riguardanti la diffusione web dell'applicazione realizzata per controllare la *RC Car*.

1.5 STRUTTURA DEL DOCUMENTO

La tesi è strutturata in 7 capitoli. Ogni capitolo, esclusi quelli di mera teoria, è diviso in due parti, prima vi è una spiegazione teorica per poi procedere con l'applicazione pratica di quanto presentato.

Il primo capitolo si occupa di introdurre e spiegare in linea generale i dettagli principali della tesi, il secondo capitolo analizza la figura del professore con particolare attenzione alla teoria costruzionista dell'apprendimento, alla quale si ispira la programmazione visuale, trattata nel terzo capitolo.

Nel quarto capitolo viene introdotto Arduino, mentre nel quinto viene presentato il binomio *S4A*, *Scratch for Arduino*, ovvero l'utilizzo di quest'ultimo tramite l'ambiente di programmazione *Scratch*.

Nel sesto capitolo viene presentato il *Progetto CARA*, la realizzazione di una Robot Car wireless control, sulla base di quanto appreso in precedenza.

L'ultimo capitolo, il settimo, concerne la conclusione della tesi, ovvero i risultati ottenuti e un resoconto finale dell'esperienza.

2 LA DIDATTICA MODERNA

Per definizione la didattica è l'attività, il modo di insegnare, più nello specifico è quella parte della pedagogia che studia i metodi dell'insegnamento [1]. È più che normale pensare che la realizzazione di qualsiasi modulo didattico potrebbe essere inutile se un soggetto esterno, il docente, non ci fornisse la chiave di lettura necessaria per comprendere l'argomento; ciò avviene durante quello che siamo soliti chiamare "lezione" o anche comunicazione. Questo è il punto focale dove fattori soggettivi, le persone, e fattori oggettivi, l'ambito e i materiali di studio, si incontrano. La lezione, come suggerisce l'autore Guido Giugni è composta da numerosi soggetti:

-l'emittente, colui che ha il compito di trasmettere l'informazione, ovvero l'insegnante;

-il ricettore, colui che riceve, decodifica e interiorizza ciò che l'emittente gli ha trasmesso, cioè il discente;

-il canale, ovvero il mezzo con cui si effettua la trasmissione; un'informazione può trasmettersi in diversi modi: tramite la parola, la mimica, la scrittura, la musica, etc.

-Il codice, un sistema convenzionale di significati intelligentemente scelto da ambedue le parti, emittente e ricettore, che attribuisce uno specifico significato ai segni usati per la comunicazione [2].

L'insieme di tutto ciò permette la nascita di un rapporto costruttivo tra docente e studente.

2.1 IL DOCENTE

Molto spesso ci si chiede come un professore possa essere un ottimo professore. Lo stesso Giugni suggerisce che un buon docente dovrebbe preoccuparsi non solo di che cosa insegnare, ma anche di cosa lo studente dovrebbe apprendere. L'autore propone anche un'affascinante similitudine che mette a confronto il mondo della didattica con quello dello sport. Il docente come l'allenatore deve proporre esercizi disinteressati, inutili singolarmente ma che non sono assolutamente fine a sé stessi, infatti contribuiscono senz'altro alla formazione ben strutturata di un fisico adatto ad affrontare le varie discipline, più complesse ed importanti. Il docente quindi si deve preoccupare di fornire quella ginnastica mentale di base per poter permettere ai suoi alunni di affrontare con sempre maggiore autonomia i problemi didattici e non, nei quali durante la loro vita potrebbero incorrere.

2.1.1 Il progetto educativo

Ovviamente quello tra studente e professore è un rapporto la cui durata solitamente non è quella di una sola lezione, bensì di un percorso, anche di diversi anni. Questo percorso è spesso definito progetto educativo, dove l'insegnante deve "**sapere**", "**saper fare**" e "**saper essere**".
Per "sapere" si intende la conoscenza oggettiva degli argomenti che il docente si troverà a trattare, con le dovute nozioni di psicologia e pedagogia;
per "saper fare" si intende l'organizzazione del programma di studi, delle lezioni e del percorso didattico;
infine per "saper essere" si intende tutto ciò che riguarda la sfera morale, l'empatia, l'accettazione e la consapevolezza del ruolo che l'insegnante sta rivestendo.

Il mio pensiero è esattamente in linea con quello della direttrice Giuliana Musarra, che nonostante siano passati diversi anni evidenzia degli aspetti importanti che un docente dovrebbe preoccuparsi di seguire, inoltre aggiunge:
“Il docente deve:
-facilitare i processi di apprendimento e programmare stimoli culturali;
-saper comprendere l’alunno come persona;
-essere disponibile all’autocritica e all’ascolto;
-sapersi liberare da pregiudizi e modelli stereotipi di apprendimento e di cultura;
-saper acquisire modalità operative (lavorare in gruppo, individuare obiettivi educativi e didattici, utilizzare realisticamente i programmi)” [3].

2.2 L’APPRENDIMENTO

A sostegno di Giugni interviene l’importante accademico Joseph Novak, il quale definisce un buon insegnante colui che aiuterà l’alunno ad andare oltre l’apprendimento meccanico, analizzando insieme a lui i significati di quanto è stato appreso [4]. Entrambi mettono su due piani diversi i concetti di insegnamento, o educazione, e apprendimento.
Il primo è imposto da un soggetto esterno, che può essere l’insegnante, come la famiglia o la società in cui si vive, il secondo invece è un processo soggettivo libero, una responsabilità personale basata sull’esperienza che lo studente stesso deve compiere per raggiungere la consapevolezza di quanto appreso, in alcuni casi modificando interiormente conoscenze, abilità e valori, che potrebbero essere utili per la risoluzione di problemi analoghi futuri.
A sua volta l’apprendimento si suddivide in apprendimento meccanico e apprendimento significativo, ma come si può notare nella Figura 2.1 David Jonassen, un noto pedagogista statunitense, sostiene che l'approccio migliore consiste nell’impiego, a seconda dei contesti, di entrambi i paradigmi.

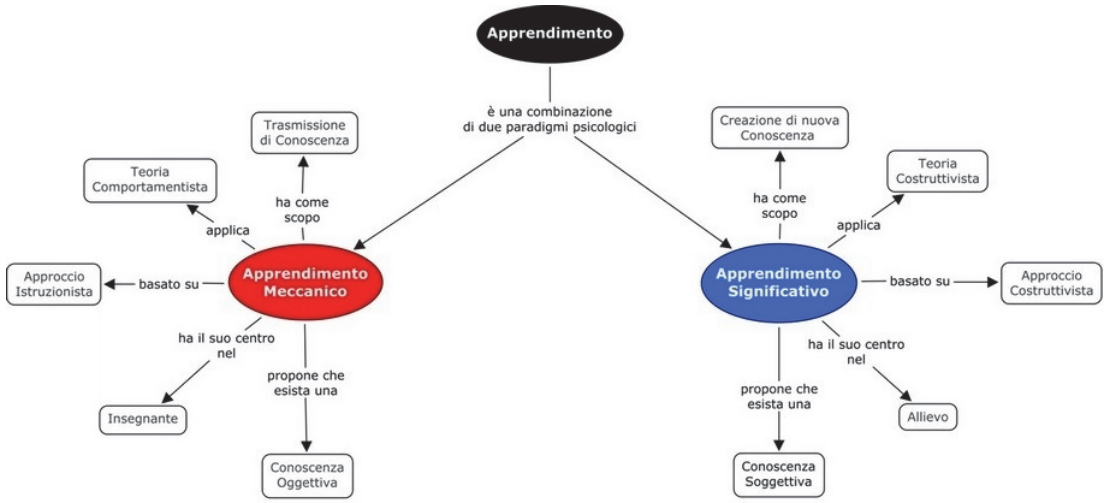

Figura 2.1

2.2.1 L' apprendimento meccanico

L'apprendimento meccanico riguarda quell'esperienza in cui vengono apprese nuove informazioni, comprensibili senza l'utilizzo di conoscenze personali, ma al limite di altre conoscenze oggettive. È sempre presente un lavoro soggettivo dell'alunno ma l'attenzione è ancora posta in larga misura sull'insegnante. Questo tipo di apprendimento è molto utile in contesti in cui occorre insegnare qualcosa che non implichi la presenza di un processo ragionativo, come ad esempio potrebbe essere imparare a memoria una poesia.

Base dell'apprendimento meccanico è la teoria comportamentista la quale trova fondamento sullo studio di variabili oggettive, in particolare l'impatto dell'ambiente circostante sull'individuo e la risposta di quest'ultimo, appunto il suo comportamento. Seguendo il pensiero aristotelico il soggetto è inteso come "tabula rasa", il quale si modifica in seguito agli stimoli di ciò che lo circonda.

2.2.2 L’ apprendimento significativo

Nell’ apprendimento significativo il focus è posto quasi esclusivamente sull'allievo, sulle sue conoscenze pregresse, apprese anche per mezzo dell’apprendimento meccanico, e sul suo desiderio di voler apprendere. È lo stesso Novak che dice: “L'apprendimento significativo si verifica quando chi apprende decide di mettere in relazione delle nuove informazioni con le conoscenze che già possiede.” A sostegno di quanto enunciato riporto la definizione di Jonassen: “Apprendere in modo significativo significa saper risolvere problemi nella realtà quotidiana. La risoluzione di problemi dà uno scopo all'apprendimento che può diventare significativo per la persona solo se essa ne comprende l'utilità per i suoi scopi.” Stiamo parlando di conoscenze soggettive che modificheranno il nostro modo di pensare e di risolvere problemi futuri. Questo genere di apprendimento si basa sulla teoria costruttivista che propone una concezione del sapere come costruzione dell'esperienza personale, e non come mera assunzione di informazioni. Alla luce di quanto detto, mi trovo in linea con il pensiero di Novak il quale afferma che per l’apprendimento meccanico potrebbe anche essere sufficiente un insegnante meccanico, ma per l’apprendimento significativo davvero costruttivo, è necessaria la presenza di un professore umano, poiché è di vitale importanza la condivisione di pensieri ed emozioni [c][4].

2.3 LA TEORIA COSTRUZIONISTA

Sulla base della teoria costruttivista, Seymour Papert noto informatico convertito pedagogista, elabora la teoria costruzionista. Questa teoria sostiene che l'apprendimento è più efficiente e proficuo se avviene mediante la creazione, da parte dell'interessato, di oggetti concreti e reali, chiamati artefatti cognitivi. L'idea di base è che la mente per apprendere e per generare un concetto potrebbe essere aiutata realizzando un qualcosa di concreto, e ovviamente scomponendo il problema in problemi di difficoltà decrescente di cui si è già a conoscenza della soluzione. Di grande importanza è la manipolazione di tali artefatti, modificandoli, unendoli, e magari talvolta generando anche errori, poiché in questo campo l'errore è indice di esplorazione. Papert è il primo che ha realizzato un linguaggio di programmazione utile per scopi educativi e per lo sviluppo cognitivo, chiamato *LOGO* basato fortemente sulla grafica e per questo spesso usato nelle scuole. Nella teoria del costruzionismo la scuola viene considerata come luogo di costruzione e non di trasmissione della conoscenza. Il ruolo dell'insegnante non è quello di essere un semplice docente, ma un mediatore, un facilitatore che allena gli studenti a raggiungere i propri obiettivi; inoltre il computer diventa uno strumento di apprendimento che permette agli studenti di formare le proprie conoscenze e idee in modo attivo e partecipe [d].

3 LA PROGRAMMAZIONE VISUALE

Come già suggerito in precedenza, Guido Giugni suddivide la lezione nei diversi aspetti che ho presentato: l'emittente, il ricettore, il canale e il codice. Per codice intendiamo il linguaggio che insegnante e alunno devono adoperare; esistono un'infinità di ambiti e diversi linguaggi utilizzabili per rappresentare un'informazione e per entrare in relazione con l'ambiente. Per quanto riguarda l'informatica si sta molto diffondendo il linguaggio di programmazione visuale. Questo genere di linguaggio consente la programmazione tramite la manipolazione grafica di elementi senza scrivere necessariamente righe di codice. Il principale punto a favore di tale programmazione è la facilità di apprendimento, al contrario viene penalizzata la libertà di concretizzare idee in totale assenza di vincoli. Inoltre, l'approccio grafico è il metodo migliore per permettere ai meno esperti del settore una prima esperienza informatica. Basti pensare a *Facebook* e *Instagram*, entrambi questi social network hanno dato la possibilità di realizzare autonomamente dei filtri personalizzati, nient'altro che delle maschere sviluppate in realtà aumentata che l'utente può applicare sui propri video e foto. Dai più semplici che modificano il gradiente di colore dei vari file multimediali, ai più complessi che possono generare dei veri e propri contesti, tramite l'inserimento di altri elementi come sfondi, personaggi, effetti sonori, o altro. Tutto ciò è realizzabile per mezzo del software gratuito *Spark AR Studio*, messo a disposizione dallo stesso *Facebook*.

3.1 SCRATCH

Logo di Papert è stato il primo linguaggio di programmazione a scopo didattico, con l'avanzare della tecnologia però sono stati realizzati numerosi altri linguaggi più potenti e con il medesimo scopo, orientandosi sempre di più verso una semplicità dal punto di vista visuale, tra i più importanti abbiamo *Scratch*. Questo è un ambiente di programmazione a blocchi di costruzione, ovvero prevede l'aggregazione di elementi per la realizzazione dei progetti, ovviamente ciò trova grande ispirazione nella teoria costruzionista precedentemente citata. Non a caso il nome deriva dalla omonima tecnica dello scratch usata dagli artisti dj, che consiste nel miscelare e modificare un prodotto musicale[d].

3.1.1 Installazione

Per procedere all'installazione, occorre semplicemente collegarsi al sito *https://scratch.mit.edu/download*, scegliere il proprio sistema operativo, scaricare il file exe ed eseguirlo. In alternativa Scratch offre anche la possibilità di lavorare online.

3.1.2 L'ambiente di lavoro

"Un programma *Scratch* è un insieme di script che vengono associati ai singoli *sprite* che si muovono all'interno di uno *stage*" [5].
All'avvio la finestra di Scratch, Figura 3.1, risulta essere suddivisa in 4 sezioni.
La sezione superiore è il menù, dove è possibile cambiare lingua, aprire e salvare file, consultare tutorial e molto altro.
La sezione di sinistra è suddivisa in tre parti, nell'estrema sinistra sono elencati i gruppi di istruzione e controlli, contrassegnati con diversi colori, che in base ai vari gruppi di azioni, elencati nella Figura 3.2, presentano nella parte destra tutti i comandi disponibili sottoforma di blocchi di codice.

Nella parte superiore invece sono presenti tre funzionalità che hanno il compito di modificare l'area centrale: "Script", "Costumi" e "Suoni".

Selezionando "Script" si attiva la finestra di lavoro dove è possibile programmare trascinando i blocchi di codice della sezione precedente; selezionando "Costumi" è possibile editare graficamente sia gli oggetti che verranno animati, chiamati *sprite*, sia lo *stage*, ovvero la finestra dove gli sprite interagiscono; selezionando infine "Suoni" verrà avviato un editor musicale con la possibilità di modificare materiali sonori associabili agli sprite.

L'ultima sezione è quella destra, anch'essa suddivisa in due parti, nella parte superiore si ottiene il risultato, ovvero avviene l'esecuzione del programma, invece nella parte inferiore si possono visualizzare tutti gli oggetti utilizzati nell'elaborazione con la possibilità di applicare semplici modifiche, come il loro ingrandimento, la loro rotazione o il loro spostamento sul piano.

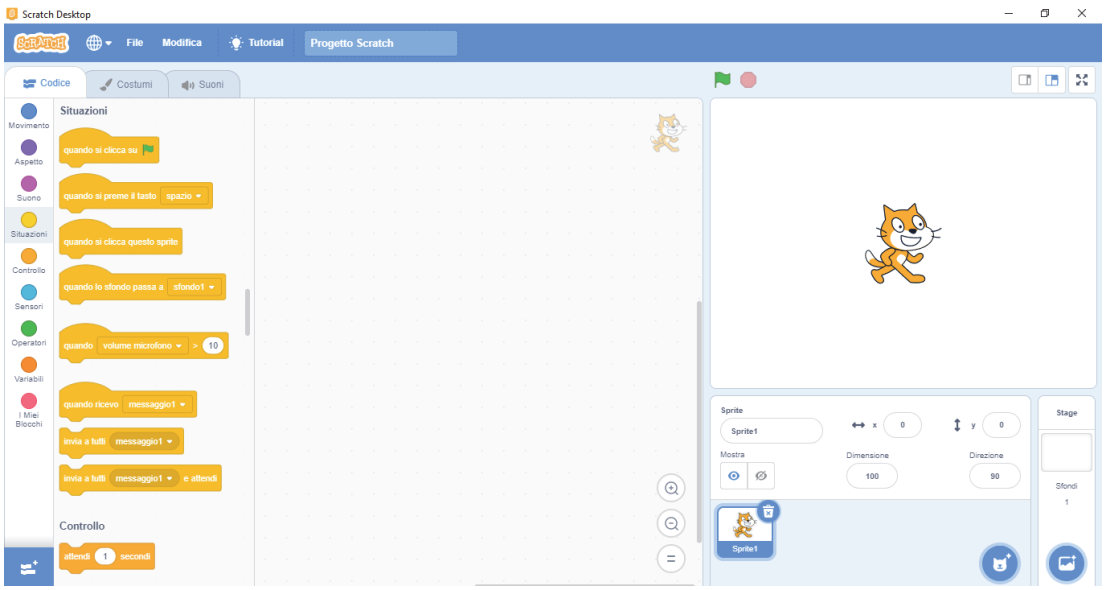

Figura 3.1

Colore	Categoria	Informazioni
	Movimento	Muove gli sprite in diverse direzioni.
	Aspetto	Controlla l'aspetto grafico dell'output.
	Suono	Esegue riproduzioni sonore.
	Variabili	Permette l'utilizzo, la modifica e la creazione di variabili.
	Situazioni	Gestisce diversi tipi di eventi come il click.
	Controllo	Istruzioni come cicli for, if, while,…
	Sensori	Propone valori sia di input dell'utente sia delle condizioni di uno sprite.
	Operatori	Operatori matematici e booleani (+,-,e,…)
	Aggiungi un'Estensione	Dà la possibilità di aggiungere altre tipologie di istruzioni, come l'utilizzo della penna, la gestione della musica, il traduttore,…
	I miei blocchi	Dà la possibilità di creare dei blocchi personalizzati.

Figura 3.2

3.1.3 Esempio

Il Problema

Si ha intenzione di realizzare un semplice programma in Scratch che simuli l'accensione di diversi led su specifiche richieste dell'utente inserite da tastiera:
-il pulsante "r" indica le operazioni effettuate sul Led Rosso;
-il pulsante "b" indica le operazioni effettuate sul Led Blu;
-il pulsante "g" indica le operazioni effettuate sul Led Giallo;
-il pulsante "v "indica le operazioni effettuate sul Led Verde;
-il pulsante "i" indica un'intermittenza, ovvero l'accensione e il successivo spegnimento di tutti i led.

Il Procedimento

Aperta la finestra di Scratch si procede con un nuovo progetto. Occorre creare 5 gruppi di blocchi, ognuno dei quali rappresenta un comando scelto dall'utente, tutti iniziano con il blocco di situazione "Quando si preme il tasto x" e terminano con il blocco di controllo "Ferma questo script". Quattro di questi gruppi sono molto simili, vedi Figura 3.3, infatti l'unica differenza è nel colore del led su cui vengono effettuate le istruzioni, l'ultimo invece ha un meccanismo diverso che riguarda l'accensione sequenziale e il successivo spegnimento di tutti i led, vedi Figura 3.4. In ogni gruppo è implementato un semplice concetto di *lock* per gestire la concorrenza di processi, con lo scopo di farne eseguire solamente uno alla volta, infatti per ogni istruzione posso procedere se la variabile è uguale a "0", ovvero se non è in esecuzione nessun altro blocco di codice, in tal caso il codice procede e la variabile *lock* viene impostata a "1", per evitare che l'applicazione inizi altri processi in contemporanea e solamente quando il blocco avrà completato tutte le istruzioni, lock verrà nuovamente settata a "0".
Nei blocchi riguardanti i singoli led, tramite i blocchi di aspetto viene fatto apparire sia lo sprite sia il messaggio "Si è acceso il led x", per mezzo di un blocco di controllo avviene la gestione del tempo da attendere per passare all'istruzione successiva, e infine viene fatto nuovamente apparire il messaggio "Si è spento il led x" con il successivo nascondimento dello sprite.

Per quanto riguarda il funzionamento dell'intermittenza inizialmente è utilizzato un blocco di controllo per permettere la ripetizione del procedimento un numero "x" di volte, il quale consiste nell'alternare blocchi di aspetto per la visibilità o l'accensione e blocchi di controllo per l'attesa.

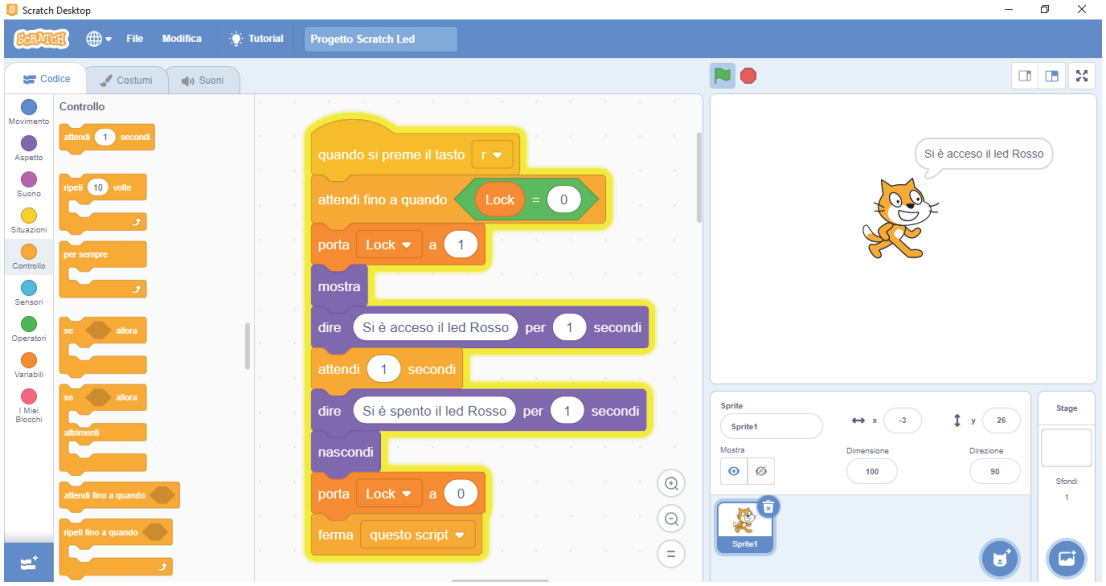

Figura 3.3

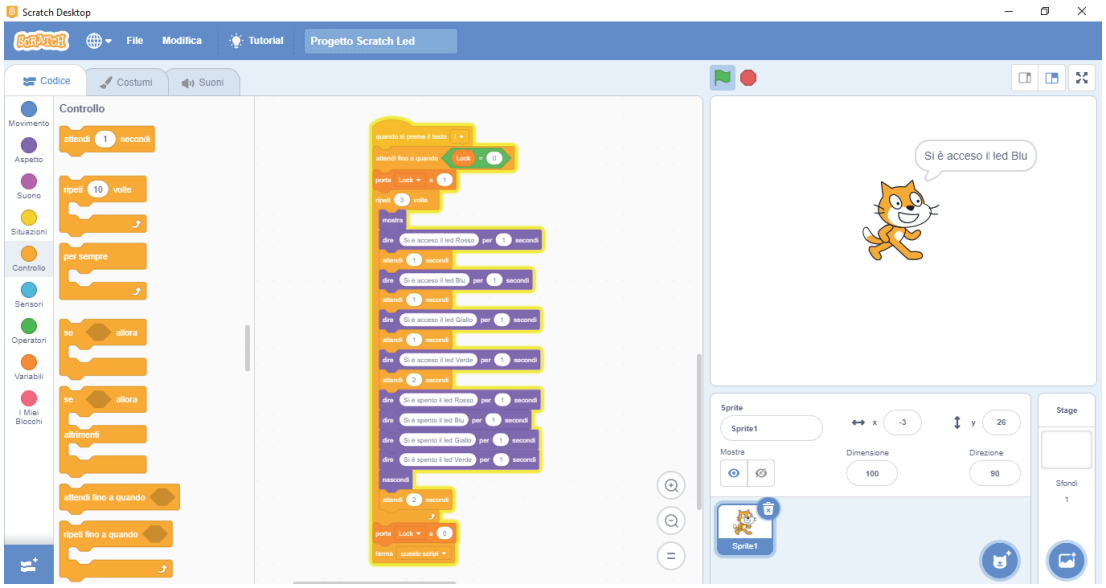

Figura 3.4

3.2 MIT APP INVENTOR

Il *MIT*, ovvero il *Massachussets Institute of Technology*, dopo aver realizzato l'ambiente di programmazione Scratch ha investito le sue risorse proponendo, sulle basi del prodotto App inventor di Google, un ambiente di sviluppo rivolto al sistema operativo Android, utilizzando sempre un linguaggio di programmazione di tipo grafico con la metodologia dei blocchi di costruzione: *MIT App Inventor*. Esistono ambienti di sviluppo molto simili, si è diffuso esponenzialmente *Thunkable*, il quale utilizza lo stesso concetto di *drag and drop* di MIT *App Inventor*, ma inoltre offre sia la possibilità di generare applicazioni per il sistema operativo *Ios*, sia di utilizzare tecnologie più avanzate, come l'inserimento di pannelli pubblicitari *Google Admob* o il componente di geolocalizzazione *Google Maps*.

Tramite questo genere di software difficilmente vengono realizzati progetti con un dominio applicativo molto complesso, infatti la logica ragionativa proposta è quella di creare intuitivamente un prototipo concreto della nostra idea, alternando lavori di *front-end* a lavori di *back-end*. Ovviamente di maggiore importanza sono le operazioni di grafica in quanto le azioni proposte dai blocchi di costruzione sono strettamente correlate agli oggetti creati per l'interfaccia utente. A sostegno di ciò basti notare che sia *MIT App Inventor* che *Thunkable* adoperano la sintassi *Dot Notation*, che impone all'istruzione una disposizione ordinata degli elementi che la compongono, come ad esempio "oggetto.proprietà" o "oggetto.azione".

3.2.1 Installazione

Mit App Inventor non necessita di alcuna installazione, in quanto è un servizio offerto online. Occorre collegarsi alla piattaforma web attraverso il seguente link *http://ai2.appinventor.mit.edu* ed accedere con il proprio account Google.

3.2.2 L’ambiente di lavoro

Dopo aver creato un nuovo progetto, nella finestra di MIT App Inventor sono presenti due sezioni: la progettazione grafica e i blocchi di costruzione, alternabili con un apposito pulsante.

Nella progettazione grafica, Figura 3.5, possiamo ulteriormente notare 4 porzioni: “Componenti disponibili”, “Visualizzatore”, “Componenti” e “Proprietà”.

In “Componenti disponibili” è possibile scegliere tra i diversi oggetti che MIT App Inventor mette a disposizione, come pulsanti, caselle di testo e immagini, che con la filosofia *drag and drop* verranno rilasciati nel “Visualizzatore” dove possiamo avere un’anteprima dell’interfaccia utente del nostro prodotto.

In “Componenti” vengono visualizzati gli oggetti in uso della nostra applicazione;

In “Proprietà” è possibile visionare e modificare le caratteristiche di base di ogni componente selezionato.

Nello specifico la sezione “Blocchi di costruzione”, Figura 3.6, si divide in 2 sezioni: “Blocchi” e “Visualizzatore”. “Blocchi” a sua volta è composto dai blocchi “incorporati”, ovvero blocchi generici che devono essere uniti ad altri, come le funzioni logiche, i messaggi di testo e i colori, e dai blocchi che il sistema ha automaticamente associato ai componenti utilizzati nella nostra applicazione.

Questi blocchi vanno trascinati nel “Visualizzatore”, dove è possibile combinarli tra loro, facendo combaciare le forme, e realizzando così un programma.

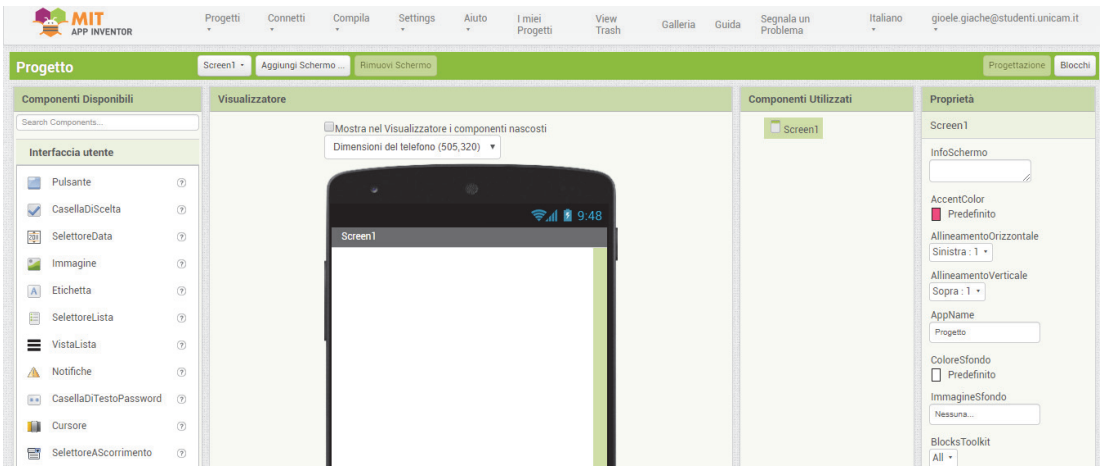

Figura 3.5

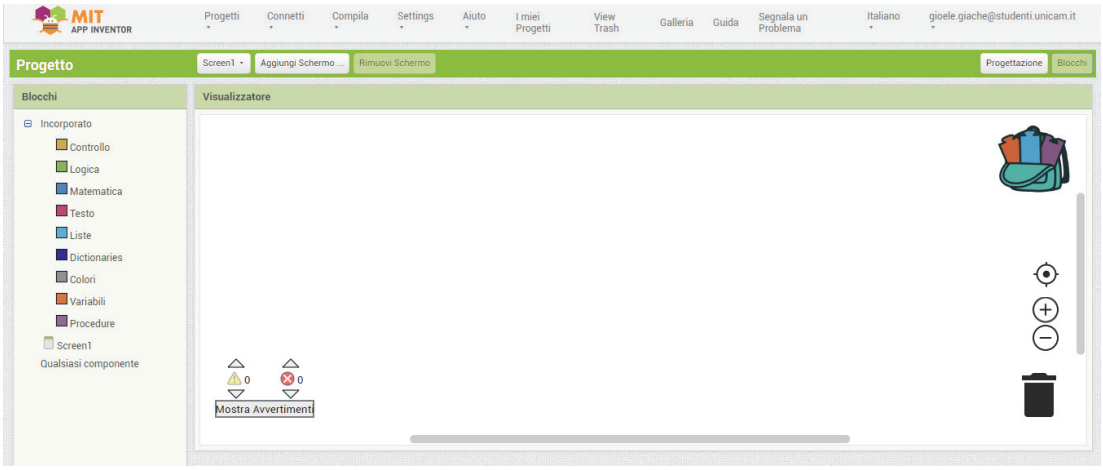

Figura 3.6

3.2.3 Test ed esportazione

Mit App Inventor offre numerose soluzioni per testare le applicazioni prodotte, consente sia di scaricare un emulatore, sconsigliato per i lunghi tempi di esecuzione, sia di eseguire il nostro progetto in tempo reale con lo smartphone. Nell'ultimo caso il nostro cellulare deve essere munito dell'applicazione *MIT AI2 Companion* che dopo aver scansionato il *QR code* o inserito il codice alfanumerico generati da *MIT App Inventor* nella sezione "Connetti", "AI Companion", del menù superiore, ci permetterà di eseguire la connessione. Terminate le operazioni di testing possiamo procedere con il download del file *apk* tramite la sezione "Compila" del menù, per poter utilizzare la nostra applicazione in totale libertà, o per distribuirla online sullo store di Google.

3.2.4 Inserimento di annunci pubblicitari

Il software MIT App Inventor non permette il caricamento di inserzioni pubblicitarie, ma rimanendo in tema "linguaggio grafico" un'ottima soluzione per ovviare a questo problema potrebbe essere quella di esportare il progetto nel formato *aia* e caricarlo in Thunkable, menzionato precedentemente.

Thunkable è molto simile a MIT App Inventor e come già affermato ha a disposizione qualche tecnologia più avanzata, tra queste la possibilità di aggiungere annunci offerti da *Google Admob*, un applicativo che tra i vari servizi permette la monetizzazione delle nostre app. Aperta la nostra applicazione in Thunkable, inseriamo per trascinamento la tecnologia desiderata, presente nella sezione “Componenti disponibili” e successivamente effettuiamo, nella sezione “Blocchi di costruzione”, la chiamata all’ annuncio pubblicitario nel punto e nel momento che desideriamo. Dopo aver dedicato uno spazio pubblicitario nella nostra applicazione, occorrerà creare l’unità pubblicitaria. Ovviamente ci sono diversi formati di pannelli pubblicitari, dai semplici *banner* che vengono visualizzati nella parte superiore o inferiore dello schermo del dispositivo, agli *Interstitial* che consistono in annunci a schermo intero, che supportano anche contenuti video. Indipendentemente dalla tipologia, per realizzarla dobbiamo registrarci in Google AdMob, il quale ha una grafica molto intuitiva che articola questo procedimento in pochi e semplici passaggi alla portata anche dei meno esperti, infatti occorre semplicemente collegare la nostra applicazione esportata in *apk* e creare l’unità pubblicitaria che preferiamo, vedi Figura 3.7.

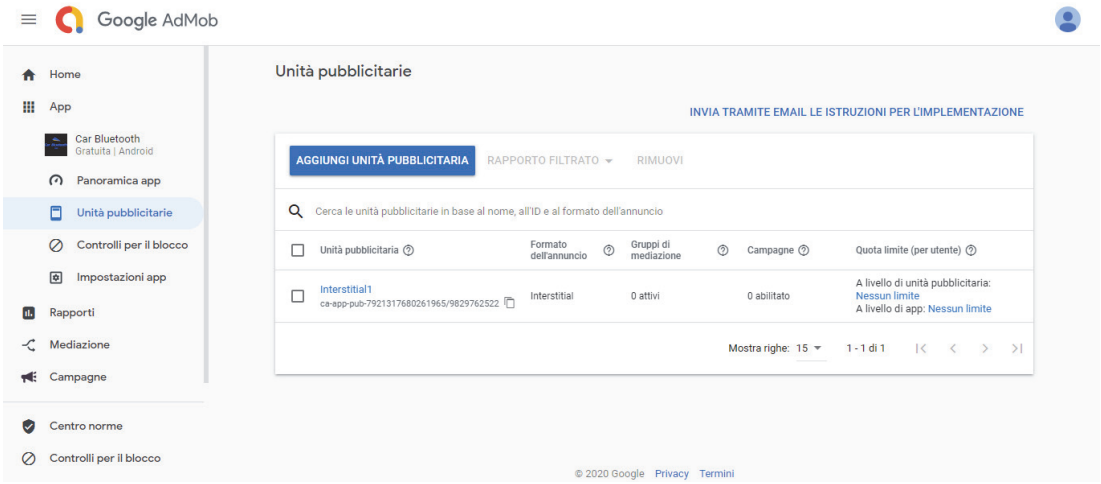

Figura 3.7

3.2.5 Pubblicazione su Google Play Store

Pubblicare un'applicazione nel Google *Play Store* significa rendere in linea di massima, escluse particolari restrizioni, quell'app disponibile a tutti i dispositivi Android.
Le procedure da seguire sono: collegarsi al seguente link: https://play.google.com/apps/publish/signup , accedere con il proprio account Google, completare la registrazione alla piattaforma e pagare la quota di iscrizione di 25 dollari, per essere abilitato al ruolo di sviluppatore Google.
Effettuati questi passaggi si avrà la possibilità di caricare l'applicazione nel *Play Store*, occorrerà ovviamente compilare una dettagliata scheda tecnica dell'app, in cui si dovranno caricare *screenshot*, certificare i contenuti trattati, definire il prezzo, la lingua, la categoria di appartenenza e molto altro. Ultimata la compilazione ed effettuato il caricamento si dovrà aspettare la verifica di idoneità da parte di Google, e nel caso di esito positivo dopo qualche giorno sarà possibile visionare la nostra app nello *Store*.

3.2.6 Esempio

Il Problema

L'esempio è molto simile a quello studiato nel paragrafo 3.1.3 infatti si ha intenzione di realizzare un'applicazione in MIT App Inventor che simuli l'accensione di diversi led su specifiche richieste dell'utente inserite tramite il *touch* del dispositivo.
-il bottone "Led Rosso" indica le operazioni effettuate sul Led Rosso;
-il bottone "Led Blu" indica le operazioni effettuate sul Led Blu;
-il bottone "Led Giallo" indica le operazioni effettuate sul Led Giallo;
-il bottone "Led Verde" indica le operazioni effettuate sul Led Verde;
-il bottone "Intermittenza" indica una successione di operazioni che consistono nell'accensione e nel successivo spegnimento di tutti i led.

Il Procedimento

Aperta la finestra web di MIT App Inventor si procede con un nuovo progetto.
Per la realizzazione dell'interfaccia grafica possiamo, a nostro piacimento, decidere come disporre gli elementi, Figura 3.8, in questa situazione quelli essenziali sono: i 5 bottoni per ogni led richiesti dal problema, un'etichetta su cui scrivere "Led Acceso" e una al suo fianco che andrà a rappresentare il colore del led acceso nel dato momento. Infine trasciniamo da i "Componenti disponibili" un sensore che ci sarà molto utile: l'orologio.
Per la parte di *back-end* occorre inizializzare le variabili che verranno utilizzate e creare 6 gruppi di blocchi, 5 dei quali rappresentano le istruzioni inerenti ai comandi dell'utente, infatti questi iniziano con il blocco di controllo "Per sempre quando bottonex.cliccato" associato all'oggetto preso in riferimento, quindi ogni volta che un bottone verrà premuto sarà eseguita una corrispondente azione. Anche in questo esempio viene implementato un semplice concetto di *lock*, con lo scopo di evitare la concorrenza di processi, infatti la logica seguita è quella di "*if-then*" se la variabile *lock* è settata a 0, allora eseguo il codice. Entrato nel codice setto la variabile *lock* a 1, rendendo di fatto inattivabili gli altri bottoni, imposto il colore dell'etichetta che rappresenta il led acceso a quello desiderato e soprattutto setto la variabile timer al valore che necessito, richiamando così un altro gruppo di blocchi di codice. Il fulcro di questo esempio infatti sono le istruzioni associate al sensore orologio, la variabile timer rappresenta i secondi di cui questo orologio è dotato che col passare del tempo diminuiscono; a seconda del valore che la variabile timer possiede, verranno eseguite operazioni differenti. Fino ad arrivare a "0" caso in cui il colore dell'etichetta viene reso bianco, ciò sta a significare che nessun led è acceso, inoltre la variabile lock viene settata nuovamente a 0, permettendo all'utente di effettuare un'altra operazione. Vedi Figura 3.9.

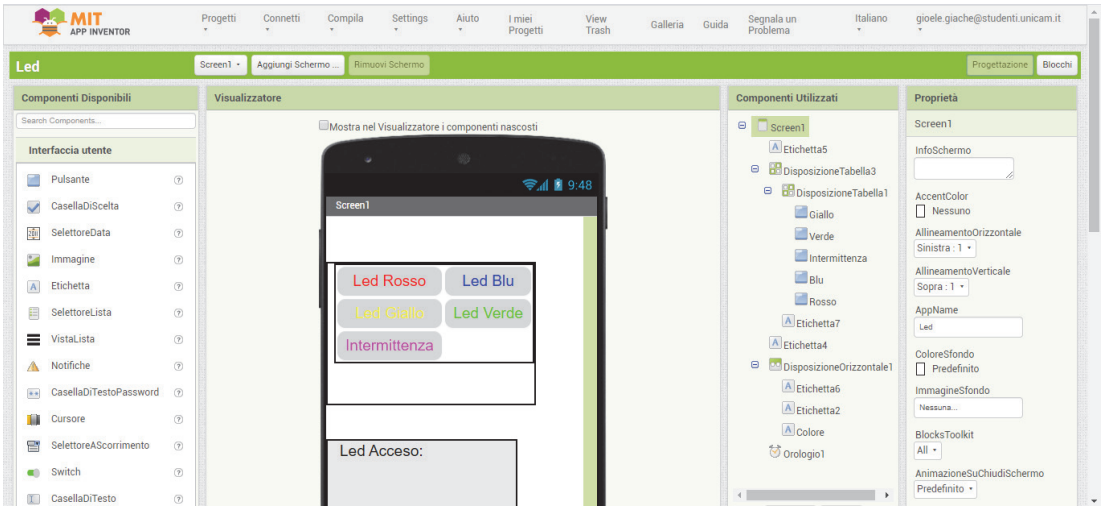

Figura 3.8

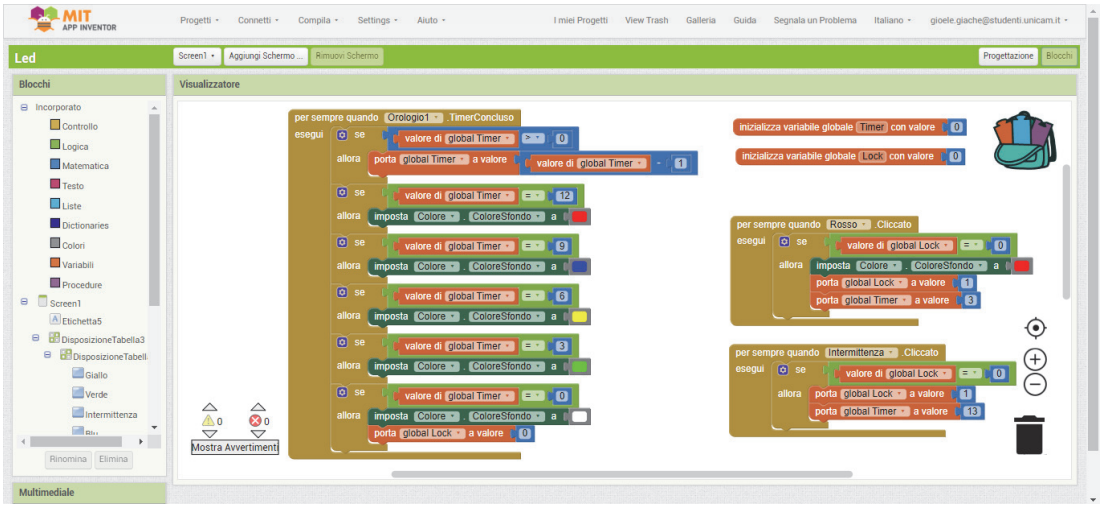

Figura 3.9

4 ARDUINO

“Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects” [e]. Questa è la definizione di Arduino che viene proposta nel sito dedicato. Come suggerisce Michele Maffucci, importante insegnante piemontese, vengono evidenziati 3 elementi inerenti ad Arduino: una parte hardware, una parte software e una comunità di sviluppo, per questo appunto lo definisce come una piattaforma senza vincoli di uso e modifica (Open Source), programmabile e in grado di interagire con altri dispositivi (Physical Computing) [f].
La storia di Arduino ha inizio quasi 20 anni fa, quando alcuni componenti dell’*IDII*, *Interaction Design Institute di Ivrea*, discutevano in un bar per la realizzazione di una scheda prestante e poco costosa capace di interfacciarsi con altri dispositivi. Il nome “Arduino” deriva da quel bar in cui è nata la discussione, che a sua volta si riferisce ad Arduino d'Ivrea, Re d'Italia incoronato nel 1002.
Arduino ha la capacità di “concretizzare” il nostro codice, nel gergo lo *sketch*, ci permette infatti di realizzare un’infinità di progetti, dagli apparati per la rivelazione di dati, come umidità, luce, distanza e temperatura, a droni e dispositivi controllabili wireless. Sono state realizzate numerose schede Arduino ma ora vedremo nel dettaglio Arduino Uno, probabilmente la scheda più diffusa e più nota.

4.1 HARDWARE

Arduino Uno si basa su un circuito stampato, il cui elemento fondamentale è il microcontrollore, il cuore della scheda, un chip che andremo a programmare in base alle nostre esigenze. Altri elementi fondamentali di questa scheda sono:

-**Il Connettore di alimentazione**, uno tra i diversi punti di ingresso per fornire energia ad Arduino. Nello specifico occorre un alimentatore che generi tensione tra 7 e 12 volt, dotato di un connettore cilindrico.
-**La Porta USB**, è un'altra opzione per alimentare Arduino ma inoltre consente di collegare la scheda al pc per poterla programmare.
-**Il Pulsante di reset**, permette di reinizializzare la scheda, disattivando tutte le uscite e riavviando lo sketch caricato, il risultato è analogo al privare Arduino di energia elettrica.
-**I Led TX e RX**, indicano la trasmissione e la ricezione di dati tra Arduino e altri dispositivi.
-**Il Power Led**, ha lo scopo di segnalare se Arduino è collegato e riceve perciò alimentazione.
-**I Pin Digitali**, ne sono presenti 14, da 0 a 13, che mediante il codice saranno utilizzati per l'input o per l'output di dati; i valori che possono essere assunti sono 1(High, tensione a 5 volt) o 0(Low, tensione a 0 volt) quindi sono adatti a situazioni in cui un componente può essere complessivamente identificato in due situazioni. La scheda però è stata equipaggiata anche di speciali pin digitali capaci di assumere un range di valori maggiore, i *PWM*, *Pulse Width Modulation* (modulazione di larghezza di impulso), basati sul concetto di durata dello stato 0 o 1 che il componente ha assunto. Particolarità del pin 13 è che possiede un led associato, se il pin ha un valore elevato il led è acceso, altrimenti è spento.
-**I Pin Analogici**, ne sono presenti 6 e sono riservati all'ingresso di segnali analogici che ricevono valori letti da sensori esterni. Per mezzo del codice ho la possibilità di conferire a questi 6 pin le stesse funzioni dei pin digitali.
-**I Pin GND**, sono i pin di terra della scheda, dove il potenziale elettrico è a 0.
-**Il Pin 5V** questo pin emette un potenziale elettrico di 5 Volt gestito dal transistor, regolatore di tensione della scheda. [6]
Vedi Figura 4.1

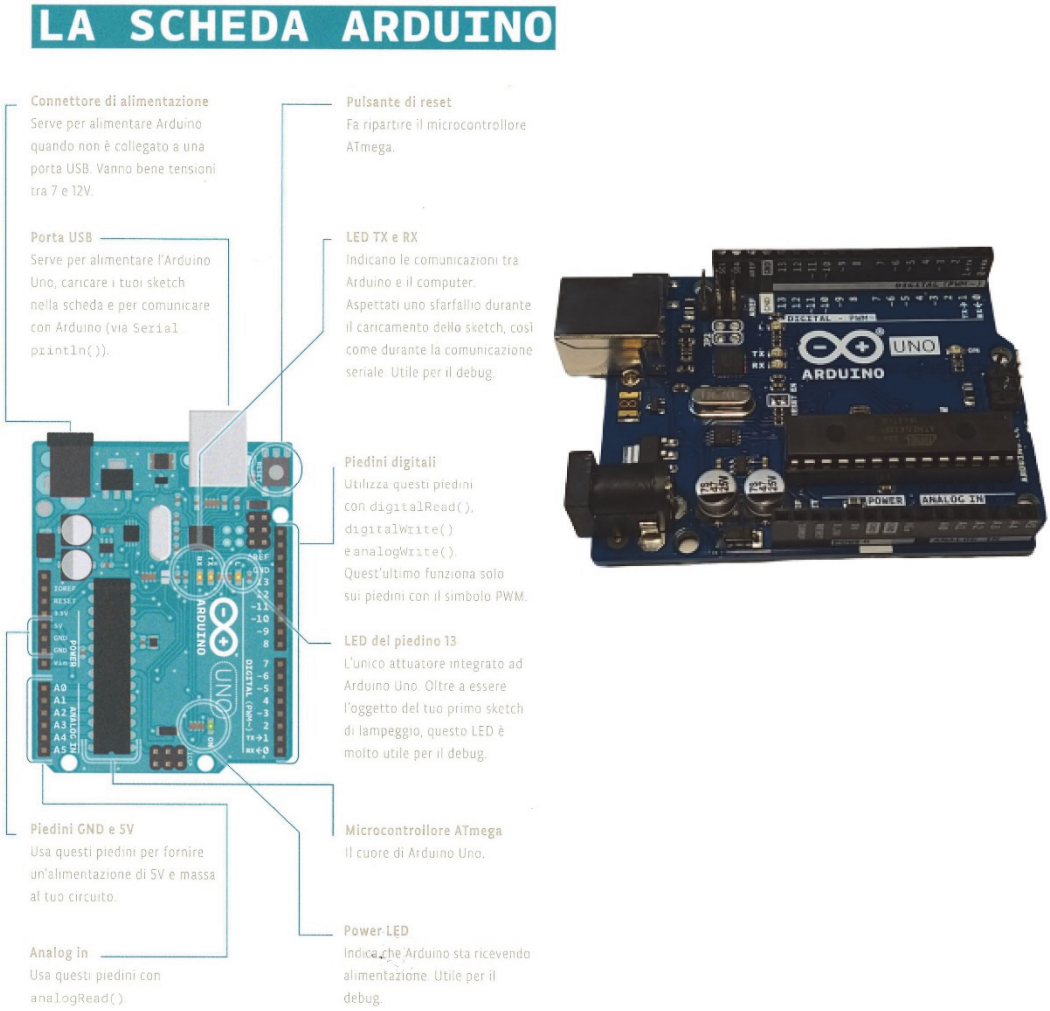

Figura 4.1 [6]

4.1.1 Componenti esterni

Arduino è in grado di interagire con un'infinità di dispositivi, qui propongo i principali componenti che incontreremo nel corso di questa trattazione.

Il Cavo USB, il quale ci permette di programmare la scheda utilizzando il nostro computer inoltre, come già affermato, è anche un modo per fornire l'alimentazione necessaria per il corretto funzionamento di Arduino.

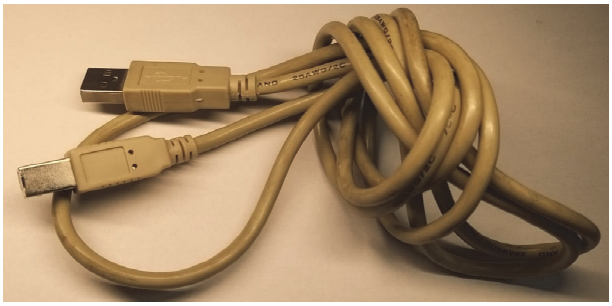

Figura 4.2

La Breadboard, ovvero il luogo fisico dove verranno costruiti i vari circuiti. Ne esistono diverse tipologie, quella in figura è la classica breadboard che non necessita interventi di saldature. Le due file esterne sia a sinistra che a destra sono dedicate all'alimentazione, la parte interna invece è l'area di prototipazione del nostro progetto.

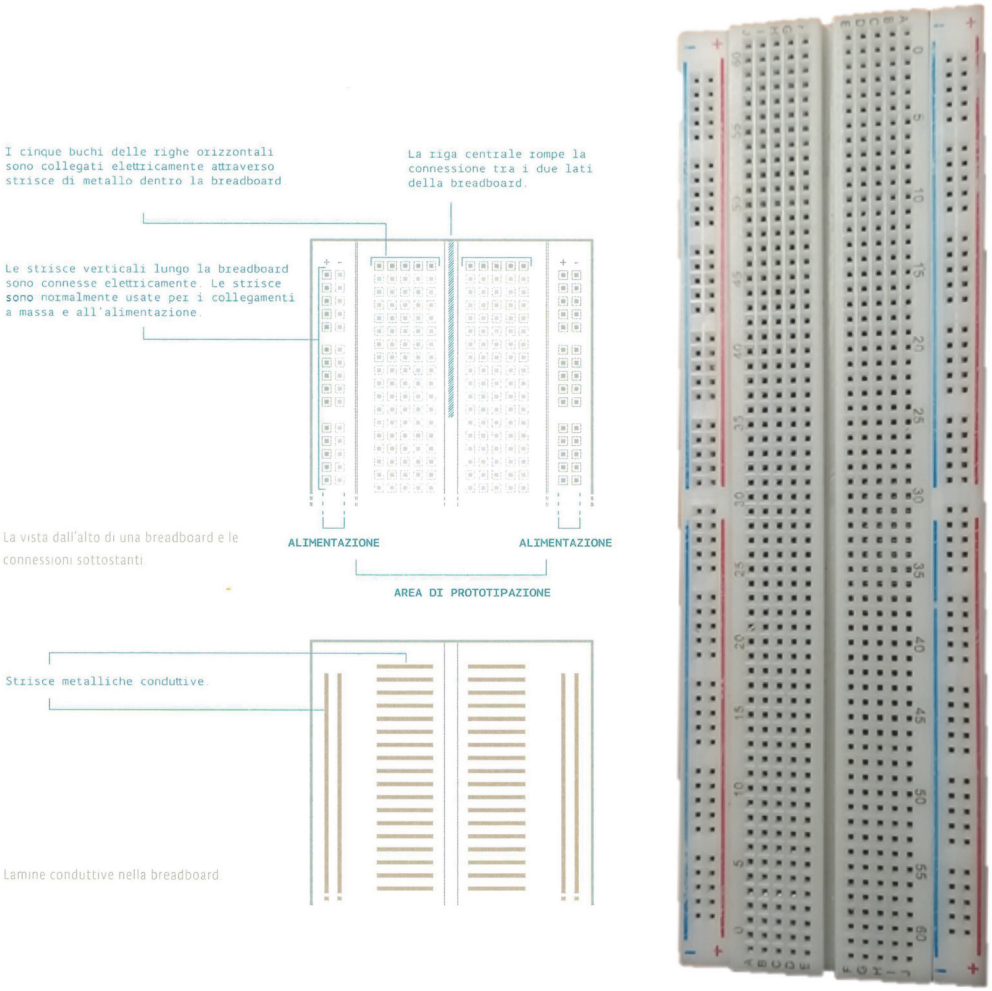

Figura 4.3 [6]

I ponticelli, sono dei veri e propri ponti metallici con lo scopo di collegare e alimentare i vari componenti adoperati. Ne esistono di diverse lunghezze, inoltre ogni estremo può essere maschio, dotato di spina, o femmina, dotato di presa.

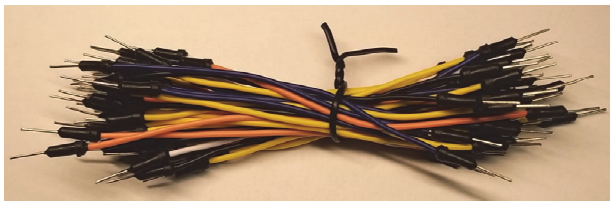

Figura 4.4

I led emettitori di luce, sono dei diodi che si illuminano se percorsi da corrente elettrica. Il polo positivo solitamente è il terminale più lungo, quello più corto è il polo negativo.

Figura 4.5

I resistori, componenti in grado di consumare parte dell'energia elettrica al fine di abbassare la tensione in ingresso di un componente, evitando sovraccarichi e malfunzionamenti elettrici. Ne esistono molti tipi, in base alla loro capacità di resistenza.

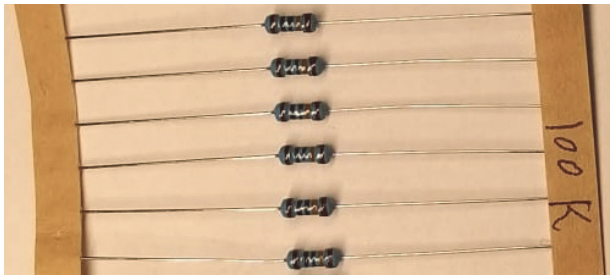

Figura 4.6

I transistor, sono dei regolatori di tensione il cui funzionamento è simile a quello delle resistenze, ma hanno un comportamento più complesso. Infatti possiedono diverse modalità di funzionamento a seconda della corrente che li attraversa, rendendo il circuito progettato chiuso, aperto o con un flusso di tensione controllato. La stessa scheda Arduino Uno ha un transistor integrato, situato vicino al connettore di alimentazione.

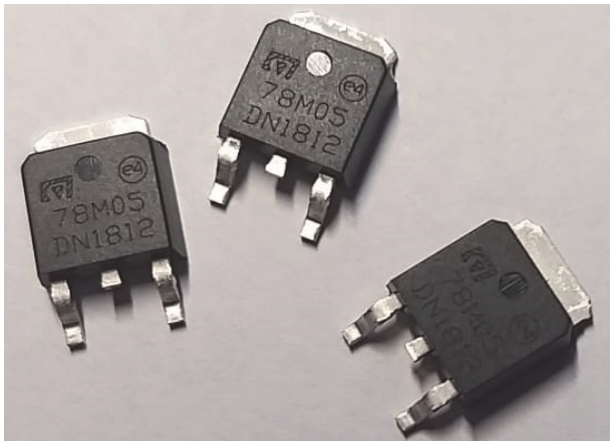

Figura 4.7

I pulsanti, ovvero degli interruttori che interrompono il flusso di corrente, chiudendo così il circuito.

Figura 4.8

I condensatori, componenti con lo scopo di rendere omogeneo il flusso di tensione. Se la tensione fornita è superiore a quella immagazzinata dal condensatore, la corrente fluisce, altrimenti se la tensione è inferiore il condensatore provvederà a rilasciare corrente.

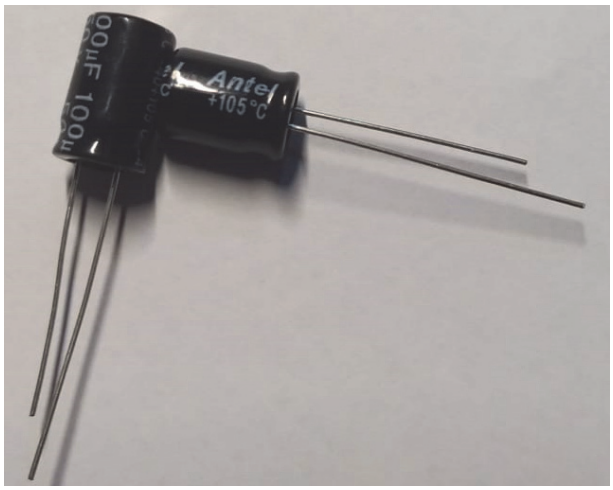

Figura 4.9

I motori a corrente continua, i quali convertono l'energia elettrica fornita in ingresso in energia meccanica, facendo così girare l'albero. Possono anche funzionare inversamente come una sorta di generatori, trasformando l'energia meccanica in energia elettrica. La tensione fornita da Arduino non è sufficiente, è necessario quindi l'utilizzo di una sorgente esterna come un generatore o delle batterie.

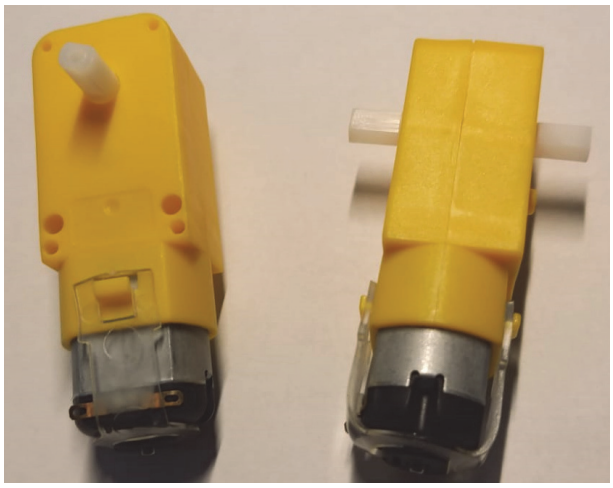

Figura 4.10

I sensori bluetooth HC-05, che conferiscono ad Arduino il potere di scambiare dati tramite altri dispositivi per mezzo della connessione bluetooth.

Figura 4.11

I sensori di posizione a ultrasuoni HC-SR04, i quali permettono di misurare la distanza da un ostacolo, tramite la riflessione del segnale a ultrasuoni generato dal componente sul primo oggetto incontrato.

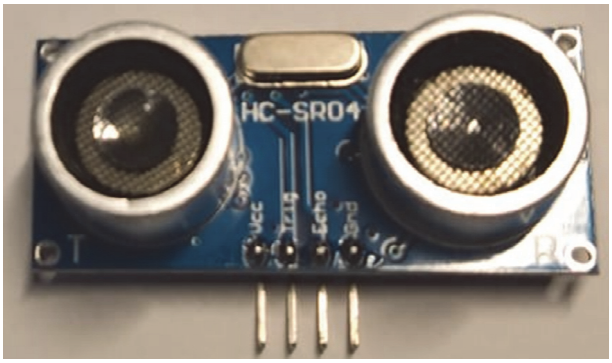

Figura 4.12

4.2 SOFTWARE

Come abbiamo già appreso la parte hardware di Arduino è affiancata da una parte software. Infatti è presente un ambiente di sviluppo integrato, *IDE*, che permette a chiunque di interfacciarsi semplicemente con la scheda Arduino, sfruttando la segnalazione di errori di sintassi e di compilazione del codice, inoltre permette di caricare con semplici passaggi il nostro programma sulla scheda. Il linguaggio adoperato è un derivato dal *C++*: il linguaggio *Wiring*.

4.2.1 Installazione

Per l'installazione del software innanzitutto bisogna collegarsi al seguente sito web: *https://www.arduino.cc/en/Main/Software*, scaricare l'Arduino IDE in coerenza con il proprio sistema operativo ed eseguirlo. Fatto ciò è possibile scrivere codice utilizzando questo ambiente di programmazione, ovviamente per poterlo caricare nella scheda dovremmo preoccuparci di collegarla al nostro computer attraverso la porta usb, inoltre nella finestra del nostro Arduino IDE nella sezione "Strumenti" va settata la tipologia di scheda che andremo a programmare, nel nostro caso Arduino Uno. Ora è possibile caricare lo *sketch* tramite l'apposito pulsante di upload.

4.2.2 Ambiente di lavoro

Potremmo dividere la finestra del software di Arduino in 3 settori. Un menù superiore che fornisce le operazioni di base, come il salvataggio del progetto, l'aggiunta di librerie specifiche per poter interagire con particolari sensori, l'apertura del monitor seriale nel caso di interazioni della scheda con il pc (come ad esempio la stampa a video di dati), però dobbiamo porre particolare attenzione ai bottoni in primo piano che rappresentano le attività più frequenti come la verifica e il caricamento dello sketch.
Immediatamente sotto troviamo l'ambiente di lavoro vero e proprio dove andremo a scrivere il nostro codice. Possiamo notare come sono presenti due importanti funzioni: `void setup()` e `void loop()`.
`void setup()` è la prima funzione ad essere eseguita quando viene avviato uno sketch, quindi qui vengono inizializzati i valori iniziali come le variabili o lo stato dei pin; inoltre saranno specificati se i pin utilizzati sono di input o output.
`void loop()` è la parte principale dello sketch che viene eseguita ciclicamente fino a quando non ci sarà un intervento esterno sulla scheda, come la rimozione di alimentazione o il reset di Arduino.

All’inizio dello sketch, esternamente alle due funzioni di cui sopra, è buona norma dichiarare variabili, costanti e librerie che verranno utilizzate nel corso del nostro programma.

Infine nella parte inferiore vi è l’area di status dove viene comunicato se la compilazione e il caricamento sono andati a buon fine o meno. Vedi Figura 4.13.

Il linguaggio utilizzato è il linguaggio Wiring, un’evoluzione del C++ con l'aggiunta di elementi per la gestione dell'hardware del microcontrollore. Come qualsiasi altro linguaggio prevede rigide regole di sintassi e di semantica che avremo modo di analizzare nel corso della trattazione[g][7].

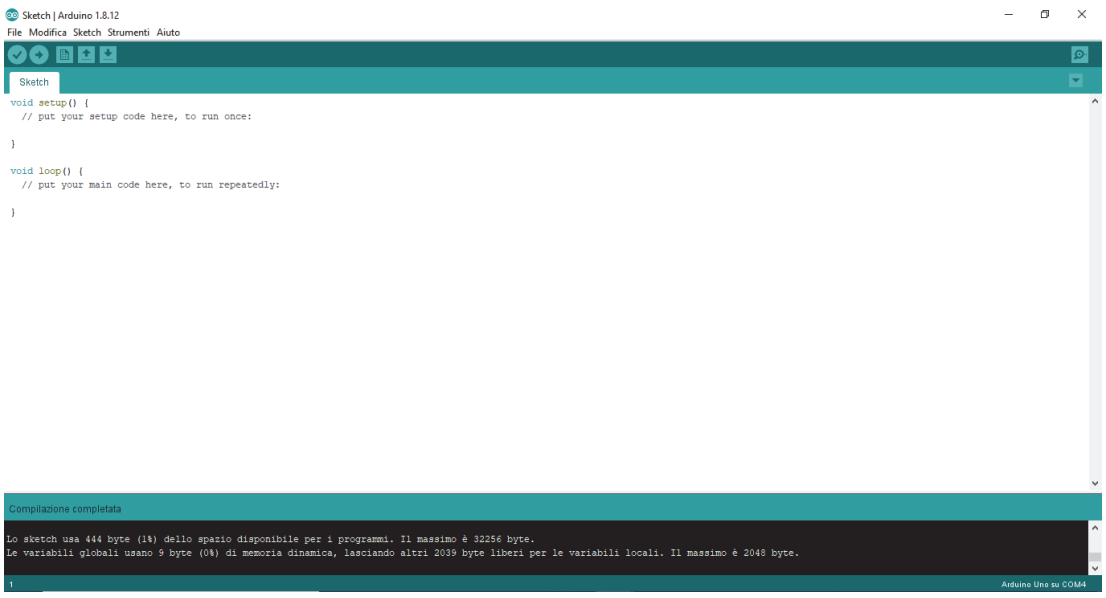

Figura 4.13

4.3 ESEMPIO

Il Problema

Si ha intenzione di realizzare uno sketch in Arduino che controlli l'accensione di diversi led.

Il Materiale

Oltre alle conoscenze di base dell'argomento, il materiale necessario per la creazione di questo progetto è il seguente:

-Un computer munito del software Arduino collegato tramite un cavo usb alla scheda Arduino Uno;

-Una breadboard;

-4 led, preferibilmente di colore diverso;

-4 resistori da 220 Ohm;

-6 ponticelli.

Il Procedimento Hardware

Solitamente prima di generare del codice si provvede a creare il circuito e affinché tutto funzioni dobbiamo realizzare un percorso completo che va dalla fonte, il punto di massima energia, sino alla massa, quello di minor energia.

La breadboard utilizzata è quella nella Figura 4.3, e su di essa andremo a disporre i led, nell'esempio corrente si trovano sulla colonna "F". Nell'area di prototipazione della breadboard le colonne non sono in comunicazione tra loro, ma lo sono le righe. Essendo a conoscenza di ciò per ogni led collegheremo un estremo di un ponticello (del colore del led) sulla riga corrispondente all'anodo del diodo, l'altro estremo verrà collegato su uno dei pin digitali offerti da Arduino. In questo caso: l'anodo del led rosso è collegato al pin 12, quello del led blu al pin 9, quello del led giallo al pin 6 e quello del led verde al pin 3. Ora abbiamo fornito l'energia ai veri led ma il circuito ancora non è terminato poiché non abbiamo collegato la massa.

Nella breadboard la massa è una delle due colonne laterali, contrassegnata con il segno “-“, andremo quindi a collegare questa colonna ad uno dei pin GND della scheda per mezzo di un altro ponticello(nero) e in seguito dovremo collegare il catodo di ogni led alla colonna GND della breadboard. Per far ciò utilizziamo i resistori, in modo tale da preservare i led da eventuali bruciature. Per scegliere il valore di resistenza adatto si deve utilizzare la seguente formula:

$R = (\Delta V)/i$,

dove “R” è la resistenza che stiamo cercando, ΔV (“Delta V”) è la differenza di potenziale tra la fonte di energia e il diodo ed “i” è la corrente di esercizio dello stesso. Nel nostro caso “Delta V” equivale alla caduta di tensione del led sottratta al potenziale elettrico offerto da Arduino, 5 Volt. Ogni led ha una caduta di tensione differente, prendiamo in considerazione quella minima che appartiene a quella del led rosso, ovvero 1,8 Volt. Quindi il nostro “Delta V” ammonterà a circa 3,2 V. La corrente di esercizio dei led si aggira sui 20mA (0,02 A, l’Ampere è l’unità di misura della corrente elettrica). In conclusione la nostra R equivarrà a circa 160 Ohm (l’Ohm è l’unità di misura della resistenza di corrente), solitamente il resistore con il valore di resistenza più basso utilizzato e reperibile è di 220 Ohm. Scegliere resistori più potenti è ugualmente sicuro ma ne risentirebbe la luminosità emessa dal diodo.

Questi resistori faranno da ponte, come già appurato, tra la colonna di massa della breadboard e la riga del catodo di ogni led. Il modello di breadboard adoperato è diviso in due sezioni quindi la corrente e la massa non sono continue, ultima accortezza sarà appunto quella di fare un ponte tramite un ponticello(bianco) tra le due parti. Nella Figura 4.14 vi è sia la foto del circuito, sia una simulazione online realizzata nel sito *https://www.tinkercad.com*. Questo sito offre anche la possibilità di generare frammenti di codice per mezzo del linguaggio grafico, con una tecnologia molto simile a quella proposta da S4A, Scratch For Arduino, di cui tratterremo in seguito.

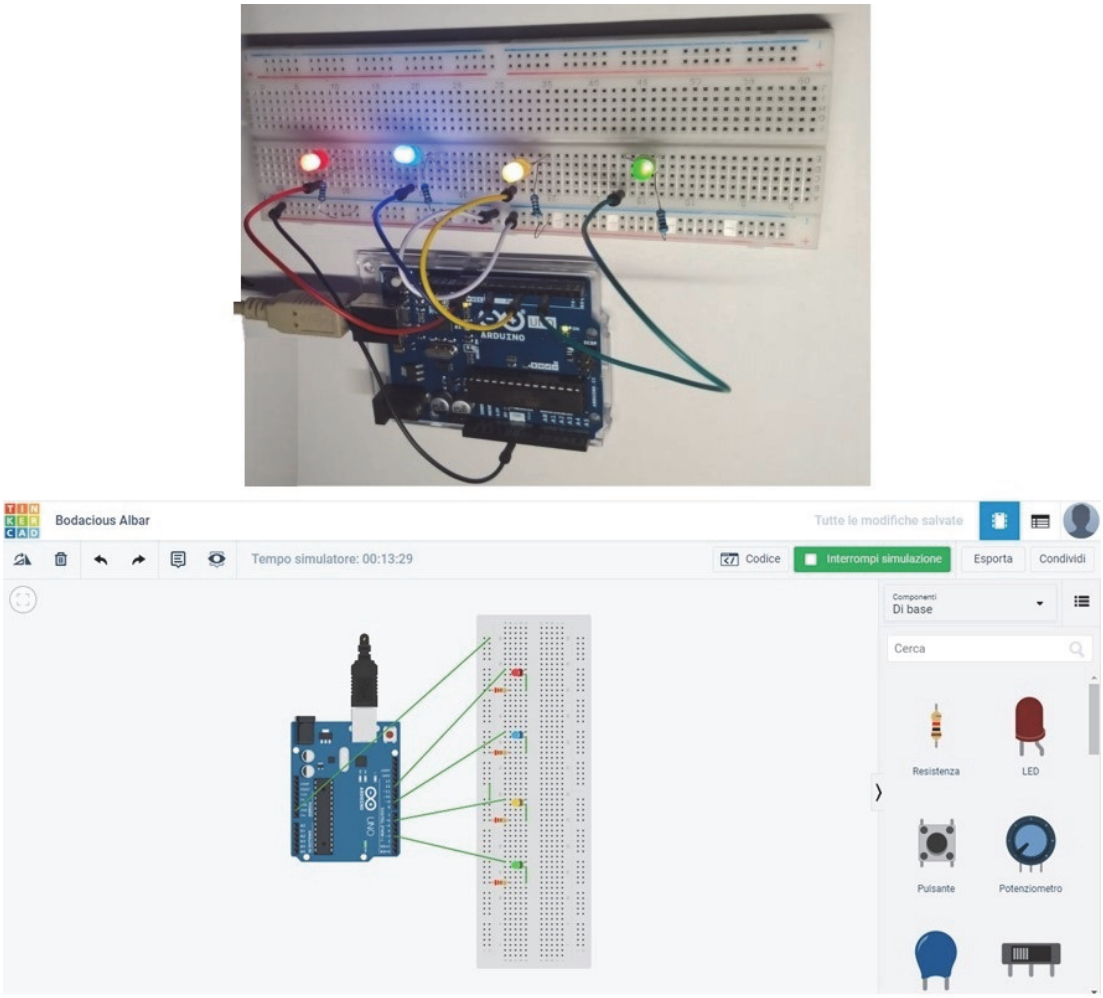

Figura 4.14

Il Procedimento Software

Conclusa la realizzazione del circuito, possiamo iniziare a trattare la parte software del nostro progetto. Vista la brevità del codice ve lo propongo nella sua interezza:

```
void setup() {
pinMode(12,OUTPUT);     // Il pin 12 è in modalità di output
pinMode(9,OUTPUT);
pinMode(6,OUTPUT);
pinMode(3,OUTPUT);
}
```

```
void loop() {
digitalWrite(12,LOW);
digitalWrite(9,LOW);    //Il pin 9 viene disattivato
digitalWrite(6,LOW);
digitalWrite(3,LOW);
delay(2000);            //Attendo 2000 millisecondi
digitalWrite(12,HIGH);
delay(1000);
digitalWrite(9,HIGH);
delay(1000);
digitalWrite(6,HIGH);   //Il pin 6 viene attivato
delay(1000);
digitalWrite(3,HIGH);
delay(2000);
digitalWrite(12,LOW);
digitalWrite(9,LOW);
digitalWrite(6,LOW);
digitalWrite(3,LOW);
delay(2000);
digitalWrite(12,HIGH);
digitalWrite(9,HIGH);
digitalWrite(6,HIGH);
digitalWrite(3,HIGH);
delay(2000);}
```

All'inizio del nostro sketch, nel `void setup()` definisco, tramite il comando `pinMode()` se i pin utilizzati sono in modalità input oppure output. Nel nostro caso non dobbiamo acquisire dati ma attivare i nostri led, perciò siamo in modalità di output. Il `void loop()` rappresenta una successione di istruzioni che verranno ripetute sino a quando non interverremo fisicamente, qui gestisco le funzioni principali del programma. Tramite `digitalWrite()` posso cambiare lo stato dei miei pin, attivandoli o disattivandoli e per mezzo di `delay()` posso far attendere il programma prima di passare all'istruzione successiva un numero di millisecondi pari al valore presente tra le parentesi tonde.

Nel complesso questa funzione si occupa di spegnere tutti i led, attivarli sequenzialmente, spegnerli nuovamente allo stesso istante e infine riaccenderli tutti in contemporanea, per poi ripetere nuovamente il ciclo.

Collegata la nostra scheda Arduino Uno al computer, questo sketch andrà caricato tramite l'apposito comando di upload, e terminato il caricamento inizierà automaticamente l'esecuzione del programma.

Procedimento aggiuntivo

Potremmo rendere il progetto più complesso cercando di gestire i led per mezzo di un sensore bluetooth HC-05. Per quanto riguarda il circuito è sufficiente aggiungere quattro ponticelli a quello precedente.

Infatti utilizzeremo 4 dei 6 pin di cui è munito il sensore: il pin VCC è riservato alla sua alimentazione e verrà collegato(ponticello bianco) con il pin 5V della scheda mentre il pin GND, la massa, verrà collegato(ponticello nero) al pin GND di Arduino , il pin RX di ricezione dovrà comunicare(ponticello giallo) con il pin digitale 1 TX di trasmissione della scheda mentre il pin TX di trasmissione del sensore dovrà comunicare(ponticello marrone) col il pin digitale 0 RX di ricezione della scheda. Questo perché ciò che viene trasmesso dal modulo bluetooth viene ricevuto dalla scheda Arduino e viceversa. Vedi Figura 4.15.

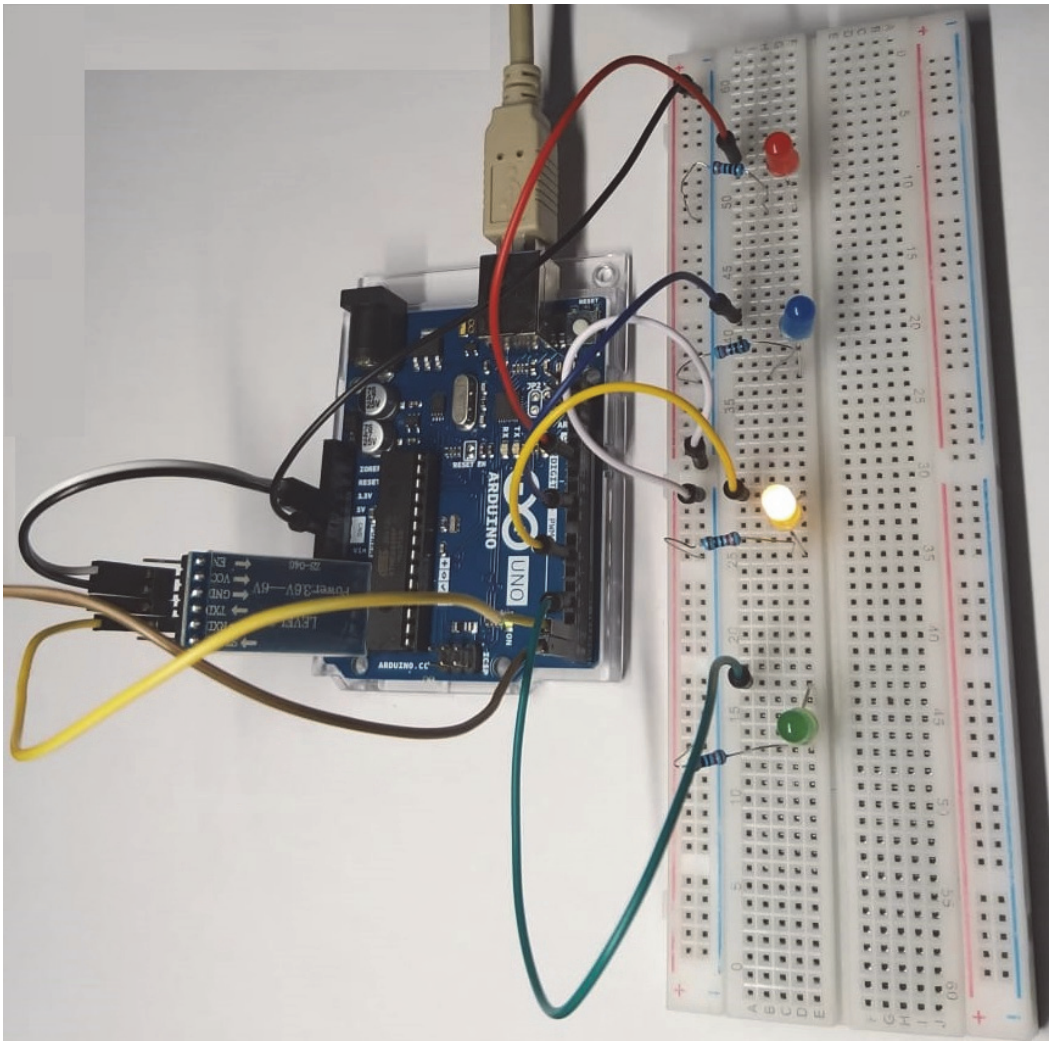

Figura 4.15

Per quanto riguarda il codice invece, è leggermente differente:

```
char command;        //Dichiarazione variabile char
void setup() {
pinMode(12, OUTPUT);
pinMode(9, OUTPUT);
pinMode(6, OUTPUT);
pinMode(3, OUTPUT);
Serial.begin(9600); //Inizializza la comunicazione seriale
                    //con il dispositivo Bluetooth
}

void loop() {
if (Serial.available() > 0) //Controlla la presenza di dati
{
command = Serial.read();    //Acquisisce il dato ricevuto
switch (command) {
case 'R':
rosso();
break;
case 'B':                   //equivale a If command=='B'
blu();
break;
case 'G':
giallo();                   //Esegue la funzione giallo()
break;
case 'V':
verde();
break;                      //Esce dallo switch
case 'I':
intermittenza();
break;
}
}
}

void rosso() {
digitalWrite(12, HIGH);
delay(3000);
digitalWrite(12, LOW);
}

void blu() {
digitalWrite(9, HIGH);
delay(3000);
digitalWrite(9, LOW);
}
```

```
void giallo() {
digitalWrite(6, HIGH);
delay(3000);
digitalWrite(6, LOW);
}

void verde() {
digitalWrite(3, HIGH);
delay(3000);
digitalWrite(3, LOW);
}
void intermittenza() {
rosso();
blu();
giallo();
verde();

}
```

In aggiunta alle istruzioni che abbiamo esaminato nello sketch precedente, ne sono presenti altre. Esternamente alle funzioni viene introdotto il concetto di variabile, infatti per mezzo del `char command` dichiaro una variabile di tipo char e di nome "command". Nel `void setup()` è presente il comando `Serial.begin(9600)` il cui scopo è quello di inizializzare una porta seriale con velocità di trasmissione di 9600 bit al secondo (la velocità di trasmissione è detta *baud rate*) per permettere la comunicazione tra Arduino e il dispositivo bluetooth. Successivamente è presente il comando `if (Serial.available() > 0)`, per verificare se effettivamente la nostra scheda Arduino ha ricevuto dei dati. Se la condizione dovesse essere vera leggiamo il dato ricevuto tramite il comando `Serial.read()` e lo copiamo all'interno della variabile "command". Successivamente è presente un costrutto del tipo "*switch-case*", il quale a seconda del valore che assume "command", eseguirà delle istruzioni differenti, ma tutte termineranno con il comando `break`, il cui scopo è quello di uscire dallo switch senza entrare in nessun altro case. Nel seguente sketch ogni case rimanda ad una funzione, ovvero un blocco di codice che sarà eseguito quando verrà richiamato, le funzioni sopra proposte sono: `rosso()`, `blu()`, `giallo()`, `verde()` e `intermittenza()`.

Le prime quattro accendono e successivamente spengono il led con il colore associato, l'ultima è una funzione che richiama le altre 4, facendo così accendere non uno ma tutti i led.

Per entrare in ogni case dello switch, la variabile "command" deve assumere determinati valori, i quali sono dati che Arduino ha ricevuto dal sensore bluetooth. Il quale a sua volta dovrà riceverli da un altro dispositivo: il nostro smartphone.

Per permettere questa comunicazione potremmo infatti utilizzare l'applicazione precedentemente realizzata come esempio per la programmazione in MIT App Inventor, ma devono essere apportate alcune modifiche. Nella sezione progettazione dobbiamo aggiungere, trascinando dai componenti disponibili, un selettore lista che se cliccato ci offre la possibilità di scegliere tra diverse opzioni ed inoltre il componente di connessione "*Client Bluetooth"* necessario per permettere al dispositivo che eseguirà l'applicazione di collegarsi ad un server bluetooth, il nostro sensore HC-05. Vedi Figura 4.16

Fatto ciò entriamo nella sezione "Blocchi". In ogni blocco riferito ai bottoni già presenti dobbiamo aggiungere un'istruzione associata al Client Bluetooth che permette l'invio di un carattere al sensore, che coinciderà appunto con un valore associabile allo switch-case. Inoltre dobbiamo creare altri due blocchi associati al selettore lista precedentemente inserito: uno, prima della selezione, che ci mostra i vari dispositivi bluetooth con i quali è possibile stabilire una connessione (già associati con il nostro smartphone), un altro, dopo la selezione, che associa il nostro smartphone al dispositivo scelto, nel nostro caso il sensore HC-05. Vedi Figura 4.17. Dopo aver installato l'applicazione creata nel nostro smartphone e caricato lo sketch nella scheda Arduino, è possibile scegliere quale led accendere del circuito realizzato semplicemente attraverso il touch.

Figura 4.16

Figura 4.17

5 S4A: SCRATCH FOR ARDUINO

Arduino è di per sé uno strumento che ha reso più accessibile e semplificato l'approccio al mondo tecnologico, ma comunque presenta una parte software basata su un linguaggio di programmazione più o meno complesso che potrebbe smorzare il desiderio di intraprendere la strada dell'informatica. In soccorso a questo problema vengono proposti numerosi metodi alternativi per interfacciarsi con Arduino, il più noto è quello di poterlo gestire attraverso un ambiente di programmazione di tipo grafico. Fra tutti è molto diffuso *S4A*, in cui il semplice l'ambiente di programmazione Scratch, analizzato precedentemente, si affianca a questa potentissima scheda. Questo è un progetto educativo nato nel 2010 dalla fusione di diverse idee che trovano capo nel team *Citilab*, un laboratorio digitale interattivo situato in Spagna il cui obiettivo è far incontrare hardware e software garantendo un'esperienza interattiva semplice e formativa.

5.1 INSTALLAZIONE

L'installazione di S4A si articola in diversi passaggi. Diamo per scontato che il nostro computer sia dotato del software Arduino di cui abbiamo parlato nel sottoparagrafo 4.2.1. Innanzitutto occorre collegarsi al seguente link *http://s4a.cat/* e scaricare il file di installazione in relazione con il proprio sistema operativo ed eseguirlo. Successivamente scarichiamo il firmware di S4A dal seguente indirizzo *http://s4a.cat/downloads/S4AFirmware16.ino*, lo apriamo e lo carichiamo come un normale sketch nella nostra scheda Arduino Uno, di fatto sono una sequenza di istruzioni in Wiring che permettono la comunicazione tra Arduino e Scratch.

5.2 AMBIENTE DI LAVORO

La finestra di lavoro di S4A è molto simile a quella di Scratch, sono presenti le stesse quattro sezioni di cui abbiamo parlato nel sottoparagrafo 3.1.2. Ovviamente i blocchi di costruzione disponibili sono largamente inferiori rispetto a quelli dell'ambiente di programmazione Scratch, in quanto operano in contesti differenti; S4A propone un incontro agevolato tra hardware e software e perciò non necessita di tutte le numerose funzionalità astratte disponibili in Scratch. Inoltre visto il dominio applicativo chiaramente più complesso e visto l'obiettivo comunque di mantenere un ambiente di programmazione semplificato, è comprensibile che le possibilità creative di *S4A* siano più limitate. È risaputo infatti che mantenere una semplicità comunicativa in un contesto in cui è presente un aumento di complessità, genera una diminuzione delle possibilità inventive, favorendo così la praticità piuttosto che la qualità. Gran parte delle funzionalità offerte dalle istruzioni di *S4A* sono rivolte alla programmazione del microcontrollore e alle operazioni sui dati dei pin analogici e digitali. Per quanto riguarda il resto delle sezioni sono analoghe, nonostante la grafica meno estrosa posso gestire gli sprite, lo stage, i suoni e quant'altro allo stesso modo. Una particolarità è che in questo ambiente di programmazione lo sprite predefinito è proprio una immagine della nostra scheda Arduino. Vedi Figura 5.1

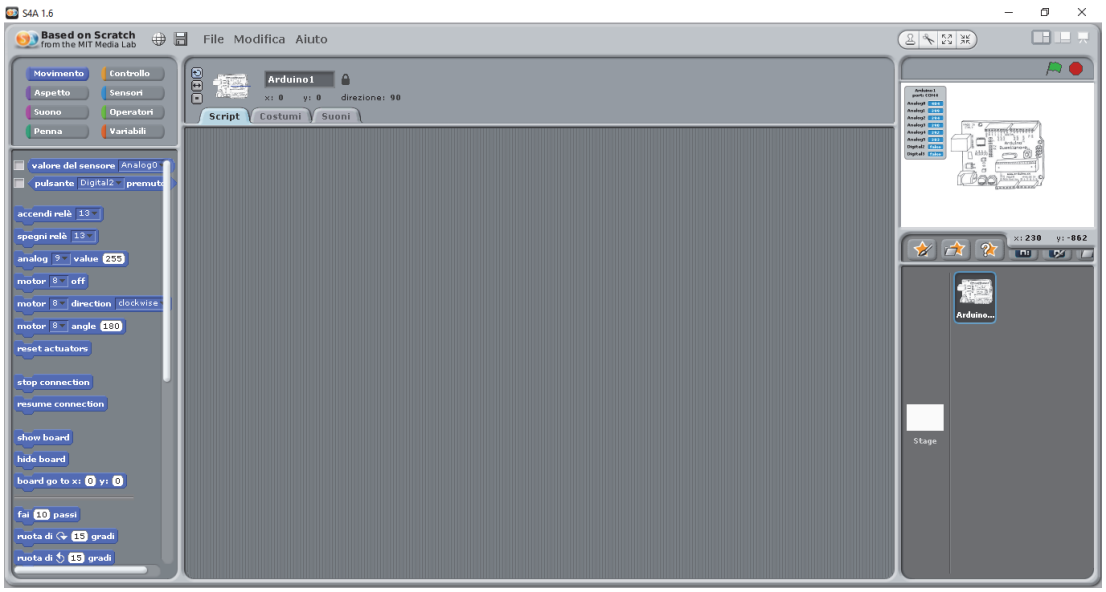

Figura 5.1

5.3 ESEMPIO

Il Problema

Si ha intenzione di realizzare uno sketch in Arduino che controlli l'accensione di diversi led su specifiche richieste dell'utente inserite da tastiera:

-il pulsante "r" indica le operazioni effettuate sul Led Rosso;

-il pulsante "b" indica le operazioni effettuate sul Led Blu;

-il pulsante "g" indica le operazioni effettuate sul Led Giallo;

-il pulsante "v "indica le operazioni effettuate sul Led Verde;

-il pulsante "i" indica un'intermittenza, ovvero l'accensione e il successivo spegnimento di tutti i led.

Il Materiale

Oltre alle conoscenze di base dell'argomento, il materiale necessario per la creazione di questo progetto è il seguente:

-Un computer munito del software Arduino collegato tramite un cavo usb alla scheda Arduino Uno;

-Una breadboard;

-4 led, preferibilmente di colore diverso;

-4 resistori da 220 Ohm;

-6 ponticelli.

Il Procedimento Hardware

Il Procedimento Hardware è esattamente identico a quello proposto nel paragrafo 4.3. L' unica differenza è a quali pin della scheda i nostri led sono collegati. Nel precedente esempio infatti i pin utilizzati erano il 12, il 9, il 6, e il 3, in questo contesto per semplicità lavorative proponiamo il seguente schema: l'anodo del led rosso è collegato al pin 13, quello del led blu al pin 12, quello del led giallo al pin 11 e quello del led verde al pin 10. Questo perché S4A predispone i pin dal 13,12,11,10 per la gestione dei relè, ovvero tutti quei dispositivi, come i led, che possono essere gestiti in due stati, aperto(acceso) o chiuso(spento).

Ovviamente è possibile utilizzare anche gli altri pin per la gestione dei led, questa è una scelta prettamente contestuale. Vedi Figura 5.2.

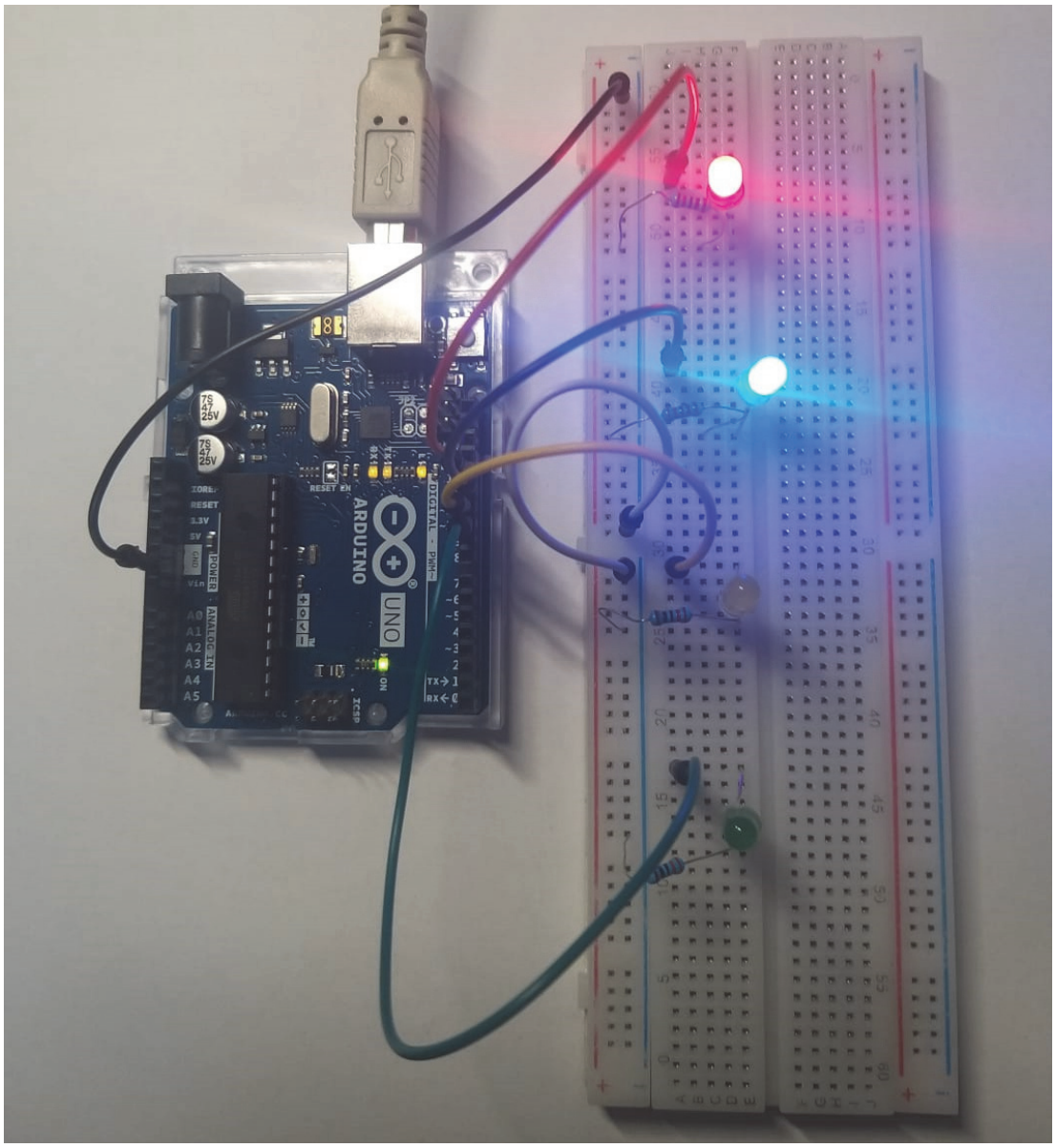

Figura 5.2

Il Procedimento Software

Il Procedimento Software invece è esattamente identico a quello proposto nel paragrafo 3.1.3. È presente lo stesso concetto di lock per la gestione della concorrenza, e inoltre sono rimasti invariati sia il tipo che la posizione dei blocchi di istruzione utilizzati. L'unica differenza è che anziché far apparire i messaggi del tipo "Si è acceso il led x" o "Si è spento il led x", per mezzo dei blocchi di aspetto, questi sono sostituiti dai blocchi di movimento che accendono o spengono il relè al pin associato. Vedi Figura 5.3.

Figura 5.3

6 IL PROGETTO CARA

Questo progetto presenta molte delle conoscenze trattate nel corso di questa tesi, infatti, come suggerisce il nome CARA, “Car Arduino”, prevede di realizzare una robot car wireless gestita attraverso lo smartphone.

6.1 MATERIALE

Oltre alle conoscenze di base dell’argomento, il materiale necessario per la creazione di questo progetto è il seguente:

-Un computer munito del software Arduino collegato tramite un cavo usb alla scheda Arduino Uno;

-Una Robot Car baseboard, al contrario di una normale breadboard, questa possiede altri componenti integrate, come condensatori, transistor, led, buzzer per emettere effetti sonori, moduli inseguitori di riga KY-033, pin per collegare dispositivi aggiuntivi, punti di ancoraggio per componenti esterni, come il box batteria o la scheda Arduino Uno stessa. Inoltre sono presenti due Motor Driver i quali ci permettono di controllare la velocità e la direzione in contemporanea di due motori a corrente continua. Sono integrati molti altri componenti, ma mi sono limitato ad elencare quelli che effettivamente utilizzeremo. Con le dovute conoscenze elettroniche, riprodurre un circuito del genere su una breadboard è possibile, probabilmente la difficoltà maggiore è nel costruire una struttura compatta e fisicamente idonea al movimento. Vedi Figura 6.1;

-Un box batteria per inserire un minimo di due batterie da 3,7 Volt;

-4 Motori a corrente continua;

-4 ruote;

-Viti, dadi, prolunghe di viti, cerniere di supporto, giraviti e quant’altro concerne l’assemblamento meccanico del dispositivo;

-3 sensori di posizione a ultrasuoni HC-SR04;

-Un sensore HC-05 bluetooth;

-Svariati ponticelli.

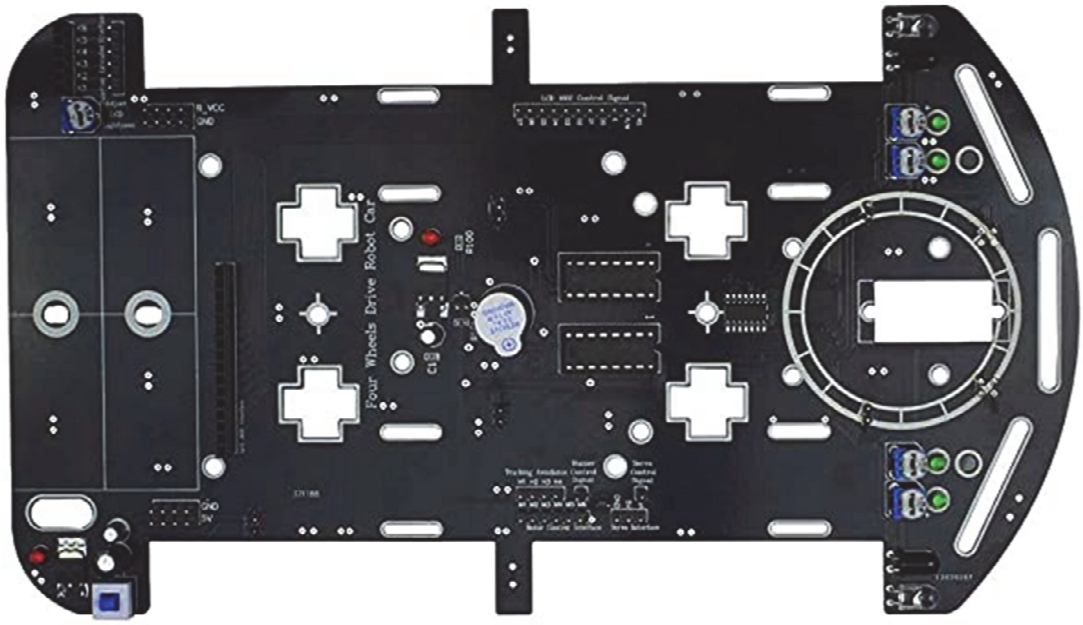

Figura 6.1

6.2 PROCEDIMENTO HARDWARE

In prima istanza bisogna procedere con l'assemblamento del dispositivo, quindi il fissaggio dei motori e delle ruote associate, le viti di sostegno per posizionare Arduino, il box batteria e infine i 3 sensori di posizione posti anteriormente alla Robot car, a destra, sinistra e centro. Fatto ciò si procede al collegamento dei vari pin. Inizialmente ci preoccupiamo di fornire energia alla nostra scheda, collegando l'alimentazione di 5 Volt e il GND della baseboard con il pin 5V e il pin GND della scheda.

In seguito colleghiamo i motori, in questo contesto la baseboard li gestisce attraverso i motor driver, quindi a coppia, proponendo un'interfaccia a cui ad ogni pin è associata un'opzione che consiste nel muovere i motori di destra o di sinistra in avanti o indietro.

Questi pin verranno collegati nel seguente modo:
con il pin 3 vengono gestiti i motori di sinistra che girano indietro;
con il pin 9 vengono gestiti i motori di sinistra che girano avanti;
con il pin 10 vengono gestiti i motori di destra che girano avanti;
con il pin 11 vengono gestiti i motori di destra che girano indietro.
I pin utilizzati sono i pin PWM che, come abbiamo già studiato, possono gestire un range di valori più ampio rispetto al semplice dualismo proposto dai pin digitali, infatti ho modo di utilizzare valori compresi tra 0, punto di minima tensione (i motori saranno fermi) a 255, punto di massima tensione (i motori gireranno alla massima velocità).
Come già affermato la baseboard integra anche i moduli inseguitori di riga KY-033, questi saranno utili per permettere alla nostra Robot Car di seguire un percorso tracciato con una riga nera sul pavimento emettendo dei raggi infrarossi che verranno assorbiti esclusivamente dal colore nero. Facciamo in modo di collegare il pin che gestisce il sensore KY-033 di sinistra con il pin 6, mentre quello di destra con il pin 7.
Per quanto riguarda il beep sonoro, la breadboard predispone un apposito pin che verrà associato al pin 13 della scheda.
Per attivare il sensore bluetooth HC-05 bisogna seguire lo schema proposto nel procedimento aggiuntivo dell'esempio al paragrafo 4.3, infatti occorre collegare il pin RXD di ricezione del sensore al pin TX 1 di trasmissione della scheda, il pin TXD di trasmissione del sensore al pin RX 0 di ricezione della scheda, il GND e la VCC(5V) del sensore rispettivamente ad un pin GND e ad un pin 5V della scheda se disponibili o della baseboard.
Infine restano da mettere in funzione i 3 sensori di posizione a ultrasuoni HC-SR04. Questo sensore è formato da 4 pin :
Il pin VCC che deve essere collegato alla tensione di alimentazione di 5V della scheda o della baseboard (è utile sapere che anche il pin IOREF di Arduino Uno è capace di fornire una tensione di 5V);
Il pin GND, il ground, che va collegato con il pin GND della scheda o della baseboard;

Il pin Trigger che deve essere attivato per poter inviare il segnale ad ultrasuoni, per il sensore di sinistra lo colleghiamo al pin analogico A0, per quello di destra al pin analogico A2 e per quello di centro al pin analogico A4;
Il pin Echo, il quale genera un impulso che si interrompe quando viene ricevuto il segnale riflesso dell'ostacolo, per il sensore di sinistra lo colleghiamo al pin analogico A1, per quello di destra al pin analogico A3 e per quello di centro al pin analogico A5.

Il circuito della nostra RC CAR ora è completo. Vedi Figura 6.2

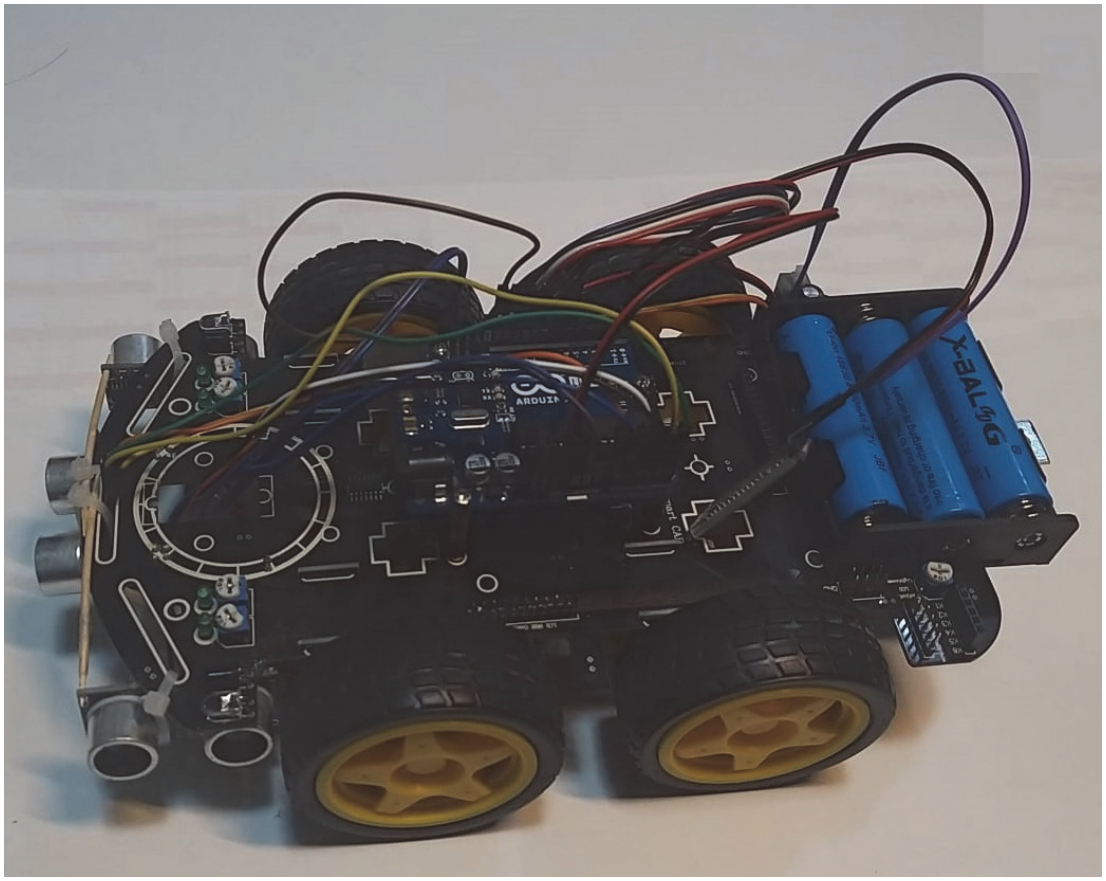

Figura 6.2

6.2.1 Quotatura e decorazione

Si potrebbe pensare di creare una sorta di carrozzeria per il nostro dispositivo avente un duplice scopo, sia per proteggere ulteriormente i ponticelli e i componenti utilizzati, sia per un fine decorativo.
L'ispirazione è tratta dalla *Tesla Cybertruck*, particolare prototipo di automobile elettrica dalla curiosa forma squadrata. Per far ciò ho realizzato una quotatura, Figura 6.3, un disegno tecnico del prodotto da costruire provvisto delle sue dimensioni effettive, in modo da poter comunicare l'immagine desiderata all'artigiano che si preoccuperà di concretizzare l'artefatto. Tramite una piastrina metallica fissata con delle viti alla Robot Car baseboard costui ha realizzato il supporto per la carrozzeria, la quale è costituita da legno fenolico multistrato di abete, e plexiglas, utilizzato per simulare i vetri, inoltre con dei chiodini interni a scomparsa la carrozzeria è munita sia di portellone che di cofano. Vedi Figura 6.4.

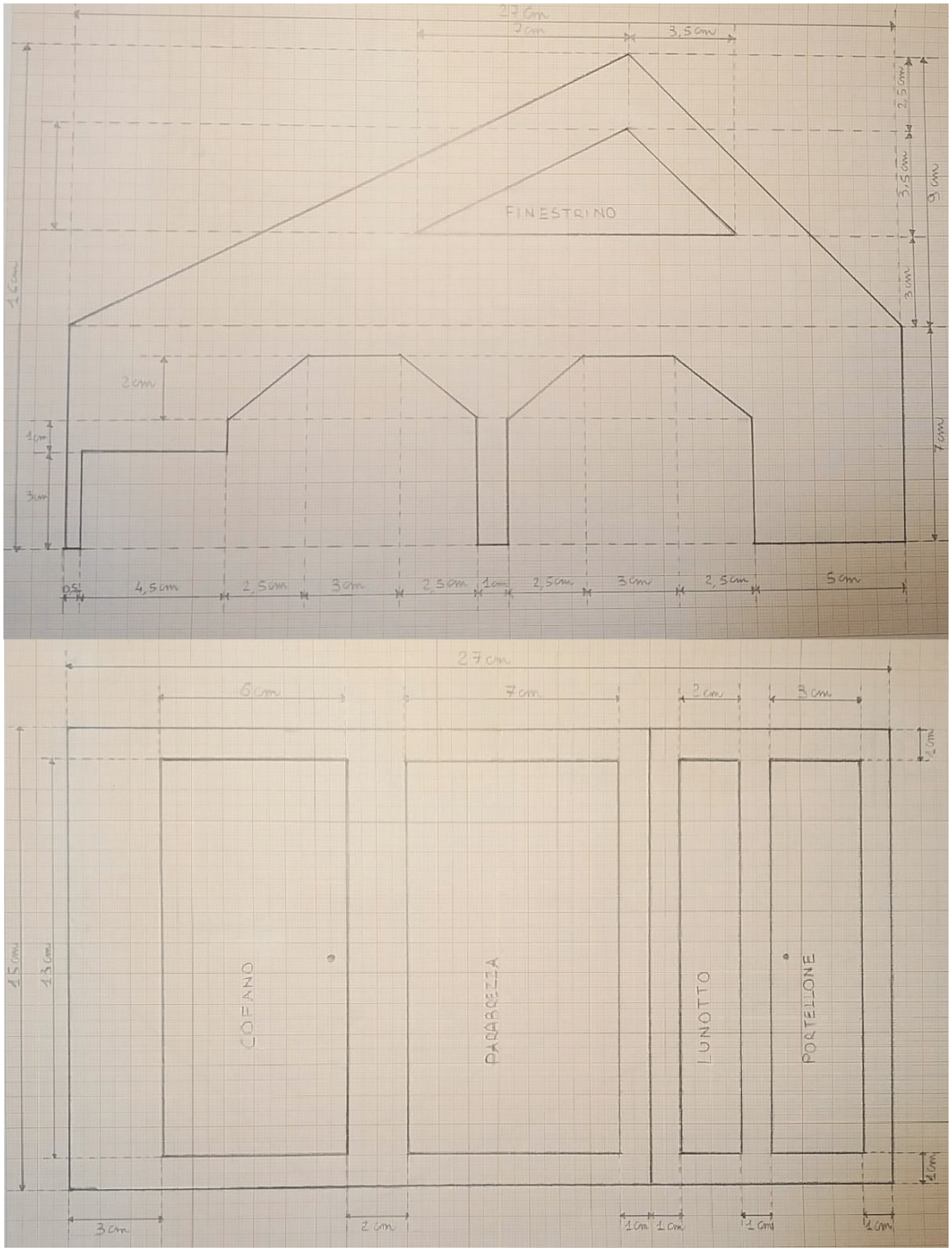
FINESTRINO
27 cm
6 cm
7 cm
2 cm
3 cm
COFANO
PARABREZZA
LUNOTTO
PORTELLONE

Figura 6.3

Figura 6.4

6.3 PROCEDIMENTO SOFTWARE

Per il procedimento software la struttura è analoga a quella presentata nel procedimento aggiuntivo per l'esempio presentato nel paragrafo 4.3. Infatti dobbiamo preoccuparci di realizzare due apparati: lo sketch da caricare sulla nostra scheda Arduino, e un'applicazione in grado di interagire con la stessa via bluetooth.

6.3.1 Sketch in Arduino

Prima di realizzare le funzioni vere e proprie è opportuno definire, per mezzo del comando `#define`, un nome consono per ogni pin della nostra scheda Arduino Uno e dichiarare le variabili che dovremo utilizzare:

```
#define left_F 9      //Il pin 10 si chiamerà Left_F
#define left_B 3
#define right_F 10
#define right_B 11
#define TRIG_L A0
#define ECHO_L A1
#define TRIG_R A2
#define ECHO_R A3
#define TRIG_C A4
#define ECHO_C A5
#define beep 13
char command;
int Speed = 185;      //Velocità con valori tra 0 e 255
double Distance_L;
double Distance_R;
double Distance_C;
```

Fatto ciò nel `void setup()` inizializzo le variabili e inoltre configuro i pin in input o in output attraverso il comando `pinMode()`.

Il `void loop()` presenta molte analogie a quello visto precedentemente, proponendo uno switch della variabile "command" con una serie di case per ognuno dei quali verranno eseguite istruzioni diverse.

È da notare che i pin utilizzati dai motori sono i pin PWM 3,9,10 e 11 che per mezzo del comando `analogWrite()` possono assumere valori di un range compreso tra 0 e 255 per gestire così la velocità di rotazione.

```
void Stop() {
analogWrite(left_F, 0); //Il pin Left_F assume il valore 0
analogWrite(left_B, 0);
analogWrite(right_F, 0);
analogWrite(right_B, 0);
}
```

Quando questa funzione viene richiamata, impone ai motori di fermarsi.

```
void forward() {
analogWrite(left_F, Speed);
analogWrite(right_F, Speed);
}
```

In questa funzione la Robot car si muove in avanti, infatti vengono attivati i pin associati a motori sia di destra che di sinistra che girano in avanti, alla velocità "Speed", variabile che può assumere sostanzialmente qualsiasi valore desiderato del range specificato.

```
void distance_L() {
digitalWrite(TRIG_L, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_L, LOW);
unsigned long tempo1 = pulseIn(ECHO_L, HIGH);
Distance_L = 0.03438 * tempo1 / 2;
}
```

Questa funzione si preoccupa di gestire il sensore ultrasuoni di distanza HC-SR04. Tramite il comando `digitalWrite()` attivo l'impulso, attendo almeno 10 millisecondi affinché ci sia l'elaborazione dei dati, e poi spengo il trigger. All'interno della variabile "tempo1", per mezzo del comando `pulseIn()`, inserisco il tempo che ha impiegato il pin Echo per passare dallo stato "Low" allo stato "High", ovvero i millisecondi che ha impiegato per ricevere il segnale riflesso. Per calcolare la distanza dobbiamo moltiplicare la velocità delle onde emesse (velocità del suono), ovvero 0.03438 centimetri/microsecondo (nient'altro che la conversione di 343,8 m/s nell' unità di misura desiderata) per il tempo impiegato. Tutto questo fratto due poiché il tempo include sia il tempo per effettuare il percorso di andata sia quello per effettuare il percorso di ritorno, che per semplicità di calcolo consideriamo identici. Nello specifico questa funzione riguarda il sensore di sinistra, ma ovviamente esistono delle funzioni identiche per il sensore di destra e per quello di centro: `distance_R()` e `distance_C()`.

```
void avoidObstacles() {

while (command=='W'){
distance_L();
distance_R();
distance_C();

if(Distance_C<25 && Distance_C>10)
{

if(Distance_L<25 && Distance_R<25)
{spin();
}
else {

if(Distance_L<Distance_R)
{ currentMillis = millis(); //Millisecondi trascorsi
while ((millis()-currentMillis<Spin/3)){
forwardRight();
}
}

if(Distance_L>Distance_R)
{ currentMillis = millis();
while ((millis()-currentMillis<Spin/3)){
forwardLeft();
} }
}
}


else if (Distance_C>25)
{

if(Distance_L<20 && Distance_R<20)
{spin();}

else if (Distance_L<20)
{currentMillis = millis();
while ((millis()-currentMillis<Spin/5)){
forwardRight();  }
}

else if (Distance_R<20)
{currentMillis = millis();
while ((millis()-currentMillis<Spin/5)){
forwardLeft();  }
}
```

```
else if (Distance_L>20 && Distance_R>20)
{forward();}
}

else if (Distance_C<10)
{currentMillis = millis();
while ((millis()-currentMillis<1000)){
back();  }

if(Distance_L>Distance_R)
{currentMillis = millis();
while ((millis()-currentMillis<100)){
forwardLeft();  }
}

else
{currentMillis = millis();
while ((millis()-currentMillis<100)){
forwardRight();  }
}
Stop();
}
loop();
}
Stop();}
```

Questa funzione ha lo scopo di far muovere la nostra Robot Car in totale autonomia evitando gli ostacoli che incontra nel percorso. E' associata al case `command=="W"` e sfrutta i valori ricavati dalle funzioni che manipolano i sensori di distanza. A seconda delle 3 distanze vengono proposte diverse combinazioni gestite da numerosi "if" e "else" che prevedono comportamenti differenti: nel caso in cui tutti i sensori rilevassero delle misure sufficientemente grandi (maggiore di 25 centimetri per il sensore di centro e maggiore di 20 per gli altri due), la vettura si sposterà in avanti (`forward()`), oppure nel caso in cui le distanze dovessero iniziare a ridursi potrebbe essere richiamata la funzione `spin()` che ha l'obiettivo di far effettuare un giro del dispositivo di 180°, o ,in altre determinate condizioni, potrebbero attivarsi le funzioni `forwardLeft()` e `forwardRight()`. La prima attiva in avanti solamente i motori di destra, la seconda solamente quelli di sinistra, ma la particolarità principale è che in questo contesto e per queste funzioni viene introdotto il concetto del tempo.

A seconda della situazioni queste continuano ad essere eseguite per un lasso di tempo predefinito, e per far ciò, ci avvaliamo del comando `millis()` il quale restituisce in millisecondi il tempo trascorso da quando lo sketch è stato avviato, il tutto veicolato dal costrutto *while*.
Data la lunghezza del codice ne è stato proposto una minima parte, ovviamente per ogni case dello switch del loop è associata una diversa funzione che genera comportamenti differenti, che analizzeremo brevemente durante la descrizione dell'applicativo realizzato in *MIT App Inventor*.
In progetti come questi, dotati di grande soggettività, è molto spesso il software che si adatta all'hardware. Determinati valori settati, le rilevazioni dei sensori stessi, la loro sensibilità potrebbero avere un impatto differente a seconda delle caratteristiche fisiche del dispositivo costruito (peso, forma, dimensioni della carrozzeria e quant'altro), è quindi opportuno che ogni creatore studi il proprio progetto e trovi i settaggi più adeguati.

6.3.2 Applicazione MIT App Inventor

Il ruolo principale della nostra applicazione in *MIT App Inventor* è quello di gestire i movimenti della RC Car, inviando specifici valori al sensore bluetooth che attivano a seconda del "case" le funzioni richieste. Gran parte della progettazione è guidata dalla soggettività, infatti abbiamo la libertà di modificare numerose caratteristiche dei componenti, come il font, le immagini, lo stile, i colori e quant'altro, vedi figura 6.5.

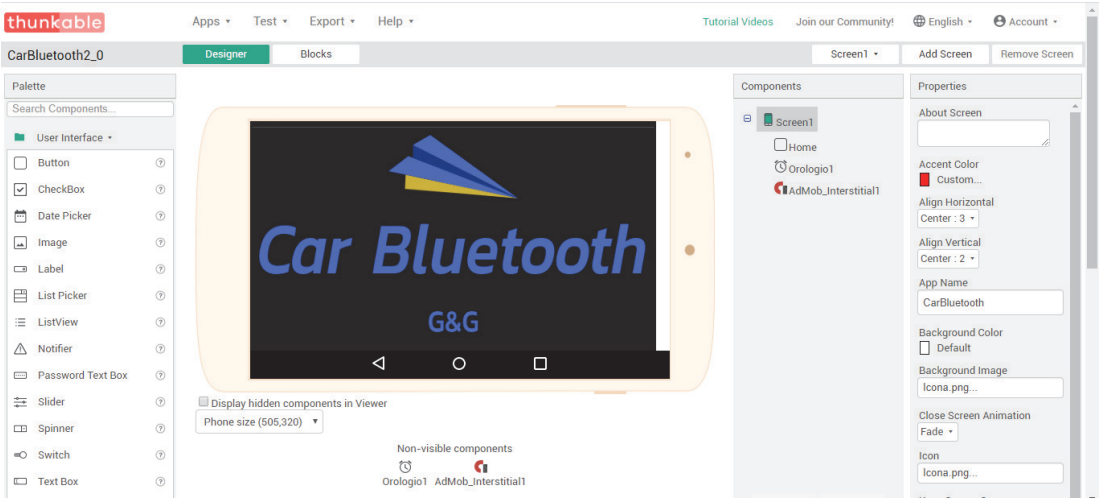

Figura 6.5

L'unico vincolo è quello di fornire i valori char utilizzati nello sketch di Arduino, per garantire una comunicazione efficace tramite il componente di connessione Client Bluetooth, analogamente a quanto presentato nel procedimento dell'esempio per la gestione dei led tramite smartphone. Nella Figura 6.6 vengono analizzati nel dettaglio 3 blocchi di codice, che riguardano il bottone "Stop" che esegue la funzione `stop()` dello sketch e il bottone "Su" che è associato alla funzione `forward()`. Possiamo notare che gran parte del codice è di natura prettamente grafica, come cambiare lo sfondo, aggiornare le immagini, variare i colori e al tempo stesso che le uniche azioni davvero necessarie sono inerenti all'invio di dati bluetooth.

Inoltre possiamo notare che questa applicazione è dotata di pannelli pubblicitari, in questo contesto vengono attivati ogni volta che il pulsante "Stop" viene premuto; come abbiamo già affermato sono stati inseriti per mezzo della piattaforma Thunkable, sottoparagrafo 3.2.4.

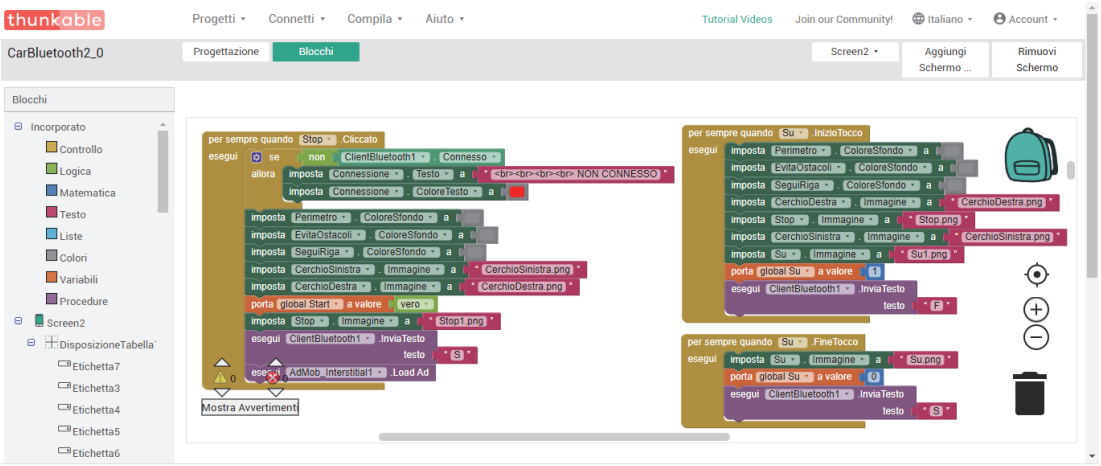

Figura 6.6

Seguendo i passaggi già presentati nel capitolo 3 questa applicazione è stata pubblicata nel Play Store di Google, ed è disponibile a questo indirizzo: *https://play.google.com/store/apps/details?id=com.gioele_giache.CarBluetooth2_0* , con il nome di CarBluetooth.

Nella Figura 6.7 è possibile visionare come si presenta l'applicazione da Smartphone.

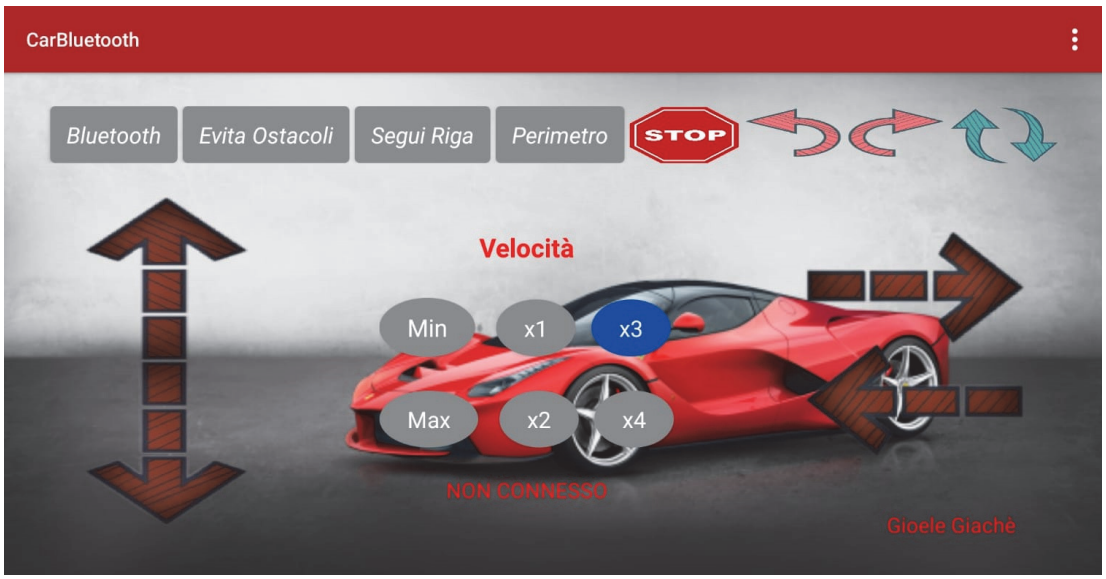

Figura 6.7

Al click di ogni bottone, eccetto per “Bluetooth” il quale è un selettore gestito esclusivamente dall’applicativo, viene inviato al sensore bluetooth un diverso valore char che attiverà così una specifica opzione dello switch:
Le quattro frecce di dimensione maggiore si occupano di far muovere la vettura nella direzione indicata, e intuitivamente le loro combinazioni generano altre direzioni.
I 6 bottoni al centro variano i valori da 0 a 255 dei pin dei motori modificando la velocità con cui la RC Car si muove, ci stiamo riferendo alla variabile “Speed”.
In alto sono presenti 8 bottoni, 4 bottoni testuali e 4 immagini.
I bottoni testuali sono:
-“Bluetooth”, il quale non è un vero e proprio bottone, bensì un selettore di lista che si occupa di stabilire la connessione con il sensore HC-05;
-“Evita Ostacoli”, è associato alla funzione `avoidObstacle()` di cui sopra e permette all’autovettura di muoversi autonomamente in totale sicurezza;
-“Segui Riga”, se il pavimento ne è provvisto, la RC Car seguirà la linea nera;
-“Perimetro”, per mezzo di questo bottone, il dispositivo si muove autonomamente mantenendo sempre una determinata distanza dalla parete o da qualsiasi altro ostacolo posto alla sua sinistra.
Invece le 4 immagini sono:
-“Stop”, associato alla funzione `Stop()`,la quale arresta il movimento dei motori;
-La freccia sinistra, fa percorrere un movimento circolare alla vettura in senso antiorario;
- La freccia destra, fa percorrere un movimento circolare alla vettura in senso orario;
-Le due frecce, associate alla funzione `spin()`,rappresentano il “testacoda”, effettuando così una rotazione di 180°.

7 CONCLUSIONI

Scrivendo questa tesi ho interiorizzato quanto sia valida la teoria costruzionista di Papert, e la grande adattabilità ad un contesto tecnologico come Arduino, allo stesso tempo però ho realizzato quanto sia importante la presenza di un insegnante, un educatore umano che guidi questo processo di crescita, per agevolare il raggiungimento, da parte dello studente, dell'apprendimento significativo.

Nello specifico questa trattazione cerca di proporre un percorso didattico volto all'acquisizione di conoscenze base per quanto riguarda sia la programmazione visuale, sia la progettazione hardware e software di Arduino. È un modulo articolato in termini di crescente difficoltà e potrebbe essere adatto per istituti scolastici di vario livello poiché non solo è un valido punto di inizio per interfacciarsi con il mondo tecnologico, ma fornisce anche numerosi input per realizzare progetti di qualità e con funzionalità sempre maggiori. Infatti la RC Car precedentemente realizzata potrebbe evolversi in un drone, o ancor meglio potrebbe entrare in uno scenario di domotica, con lo scopo di agevolare la qualità delle nostre vite per mezzo dell'informatica. Utilizzando specifici sensori potremmo automatizzare azioni quotidiane, magari ridurre il consumo di energia elettrica o di acqua, potremmo regolare gli antifurti e molto altro. Le possibilità sono davvero infinite, forse l'unico limite è imposto dalla nostra creatività.

Lo scrittore Rodotà guardava il domani con ammirazione e speranza: "Cosa ci riserva il futuro? È vicino il giorno in cui potremo vedere su di un monitor collegato al telefono l'immagine della persona con cui conversiamo?" [8]. Senza ombra di dubbio possiamo affermare di aver soddisfatto le sue aspettative, ma l'accademico Jay David Bolter aggiunge, agli occhi pieni di meraviglia di Rodotà, un cenno di angoscia. Si preoccupa del futuro in relazione alla natura circostante, si preoccupa della pericolosità della tecnologia, pericolosità che in un periodo incerto come questo, abbiamo iniziato a comprendere. "Viviamo in un'epoca spettacolare ma molto incerta e le prospettive per il futuro non sono mai state così eccitanti.[…]Molte epoche passate hanno riposto grandi speranze

nell’affrontare notevoli difficoltà, ma la nostra epoca è l’unica nella quale i problemi e le promesse scaturiscono da un’unica sorgente: dagli straordinari risultati raggiunti dalla scienza e dalla tecnologia.[…] L’umanità è insieme Bene e Male e ci si deve aspettare che a volte, la tecnologia umana danneggi la natura piuttosto che migliorarla” [9]. Pensieri del genere fanno vacillare, ci si domanda se davvero la strada dell’inarrestabile progresso che abbiamo intrapreso sia davvero quella giusta. L’unica certezza è che fin quando il progresso e la tecnologia sono veicolati dal vero buon senso, di certo non stiamo commettendo alcun errore. Ma qual è il vero buonsenso?

* * *

Sono felice di aver realizzato questa esperienza nel mio ex Liceo scientifico e per la mia formazione, spero vivamente di poterne fare altre analoghe. Questa scuola per me rimarrà sempre un luogo di grande importanza dove ho trascorso diversi anni della mia vita, un posto in cui ho imparato tanto, in cui ho avuto le prime delusioni, un posto dove sono nati i primi amori, un posto dove ho conosciuto grandi amici. Un posto dove c’eri anche tu amico mio, ed io ti ricorderò per sempre qui. Ancora Ciao Gianluca!

APPENDICE

GLOSSARIO DEI TERMINI

Acronimo/Termine	*Definizione*
aia	Estensione dei progetti utilizzato da piattaforme web come MIT App Inventor e Thunkable.
Ampere	Unità di misura della corrente elettrica.
Android	Sistema operativo per dispositivi mobili sviluppato da Google.
apk	Formato di file utilizzato per la distribuzione di applicazioni nei dispositivi mobili Android.
Back-end	Parte di un applicativo non visibile all'utente finale che ne permette l'effettivo funzionamento.
Banner	Pubblicità dalla forma rettangolare che appare nella parte superiore o inferiore dello schermo del dispositivo.
Baud Rate	È la velocità di trasmissione con cui I dati vengono trasferiti in un canale di comunicazione.
Bluetooth	Consiste nella trasmissione di dati senza l'utilizzo di fili per brevi distanze e per mezzo delle onde radio.
C++	È un linguaggio di programmazione, evoluzione del linguaggio C.
Client Bluetooth	Componente in MIT App Inventor utilizzato per connettere il dispositivo ad altri dispositivi tramite bluetooth.
Domotica	Disciplina che si occupa di studiare tecnologie per migliorare la qualità della vita negli edifici.
Dot Notation	È uno schema sintattico per riferirsi alle proprietà degli oggetti (<oggetto>.<proprietà>).

Drag and Drop	Successione di operazioni che consistono nel cliccare su un oggetto, trascinarlo in un'altra posizione e rilasciarlo.
exe	L'estensione di un file che contiene codice eseguibile, cioè un programma.
Front-end	Parte visibile all'utente di un programma e con cui egli può interagire.
Hardware	Si intende la parte fisica e i componenti concreti di un dispositivo.
IDE	Un ambiente di sviluppo integrato che supporta i programmatori nello sviluppo del codice di un programma.
IDII	Interaction Design Institute Ivrea, scuola che studia l'interazione tra uomo e macchina.
If-Then	Logica che dopo aver valutato una condizione, determina quali sono le istruzioni da eseguire.
Interstitial	Annunci pubblicitari che occupano tutto lo schermo del telefono.
Ios	Sistema operativo sviluppato da Apple.
Lock	Meccanismo per limitare l'accesso ad una risorsa e quindi la concorrenza.
Logo	Linguaggio di programmazione a scopo didattico.
Monitor Seriale	Permette ad Arduino di comunicare con il pc.
Ohm	Unità di misura della resistenza elettrica.
Open Source	Software non protetto da copyright e liberamente modificabile dagli utenti.

Physical Computing	Disciplina che ha per scopo la realizzazione di sistemi attraverso l'impiego combinato di software, hardware e altri dispositivi.
Pin Analogici	In Arduino Uno sono i pin da A0 a A5 ed hanno la capacità di leggere tensioni comprese tra 0 e +5V.
Pin Digitali	In Arduino Uno sono i pin da 0 a 13 e possono assumere due valori, LOW (0V) e HIGH (5V).
Progetto CARA	Realizzazione di un progetto hardware e software di una RC Car gestita tramite bluetooth.
PWM	Pulse width modulation, modulazione di larghezza di impulso. Questo tipo di segnale permette di ottenere un segnale analogico variando il periodo dell'impulso. In Arduino Uno sono i pin digitali 3,5,6,9,10 e 11.
QR Code	Un codice QR è un codice a barre bidimensionale che può essere letto dagli smartphone.
RC Car	Remote Controlled Car o Radio Controlled Car, veicolo controllato solitamente tramite onde radio.
S4A	Scratch for Arduino è una modifica di Scratch che permette di programmare Arduino.
ScreenShot	La fotografia dello schermo di un computer o di uno smartphone.
Server Bluetooth	Componente in MIT App Inventor utilizzato per trasformare il dispositivo in un server che riceve connessioni bluetooth.
Sketch	È il codice scritto nell’Arduino IDE.
Software	Sono l’insieme dei programmi appartenenti ad una specifica piattaforma.
Spark AR Studio	Piattaforma di realtà aumentata che consente di creare effetti AR, ovvero augmented reality, realtà aumentata.

Switch-case	Struttura di controllo che permette la verifica del valore di un'espressione, distinguendo più casi possibili (case).
Tabula rasa	Termine filosofico per esprimere l’assenza della conoscenza.
Touch	È un particolare dispositivo che permette all'utente di interagire con un'interfaccia grafica mediante il tocco dello stesso.
Ultrasuoni	Onde sonore con una frequenza elevata, perciò non udibili.
Volt	Unità di misura del potenziale elettrico.
While	Consente di far eseguire al programma dei cicli finché una determinata condizione è valida.
Wiring	È il linguaggio di programmazione, derivato dal C++, utilizzato per scrivere programmi nell’ Arduino IDE.

BIBLIOGRAFIA

[1] Dizionario di italiano Garzanti 2010.

[2] Guido Giugni, "*Ipotesi e strategie per la programmazione didattica nella scuola*" 1986.

[3] Giuliana Musarra, "*Progetto educativo, qualità dell'istruzione: criteri -metodi -sussidi operativi*" 1996.

[4] Joseph Novak, "*L'apprendimento significativo. Le mappe concettuali per creare e usare la conoscenza"* 2001.

[5] Paolo Camagni,Riccardo Nikolassy, *"InfoM@t Volume 1*" 2018.

[6] Scott Fitzgerald & Michael Shiloh with Tom Igoe *"Arduino, il libro dei progetti"* 2013.

[7] Tiziana Marsella & Romano Lombardi, "*Elementi base del linguaggio di programmazione di Arduino".*

[8] Carlo Rodotà, "*Un modo agevole per comprendere L'INFORMATICA e per farne un uso corretto"* 1986.

[9] Jay David Bolter, "*L'uomo di Turing. La cultura occidentale nell'età del computer"* 1985.

[10] Mario Gattullo, "*Didattica e Docimologia, misurazione e valutazione nella scuola*" 1985.

[11] Anthony Chandor, John Graham e Robin Williamson, "*Dizionario di Informatica"* 1974.

[12] Samuele Giachè, "Tesi di Laurea Sperimentale in Reti degli Elaboratori: *Modellazione e sviluppo di una ontologia a supporto dell'adattabilità dei servizi nel dominio di e-Government", 2006.*

[13] Samuele Giachè, "Tesi di Laurea Sperimentale: *Definizione di un modello logico di gestione dei ruoli e sua implementazione nei sistemi informativi del gruppo Ubi Banca", 2009.*

SITOGRAFIA

[a] https://www.mondadorieducation.it/media/contenuti/statici/didattica/flipped/assets/pdf/01_ruolo_docente.pdf

[b] https://www.dirittoscolastico.it/riforma-superiori-e-guerra-tra-scuole/

[c] http://www.pensierocritico.eu/apprendimento-significativo.html

[d] https://www.informarsi.net/seymour-papert-costruzionismo/

[e] https://www.arduino.cc/

[f] https://www.maffucci.it/

[g] http://www.angeloangeletti.it/

[h] http://www.treccani.it/

[i] https://it.wikipedia.org/

[j] https://scratch.mit.edu/

[k] https://appinventor.mit.edu/

[l] http://app.thunkable.com/

[m] https://www.tinkercad.com/

[n] https://www.html.it/

[o] http://s4a.cat/