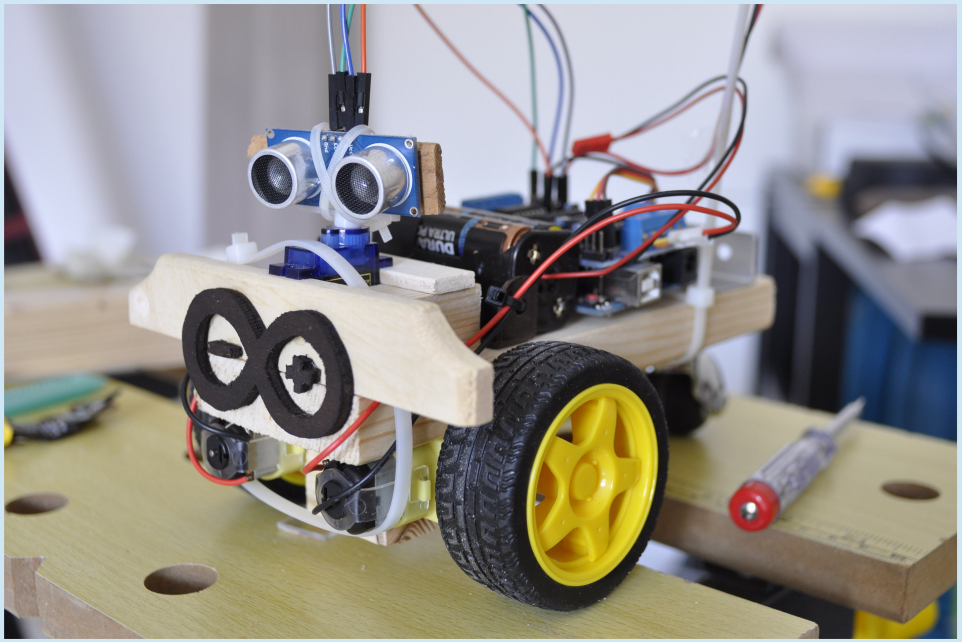
Eliot 1.0

Caso di studio di sviluppo software
Eliot 1.0



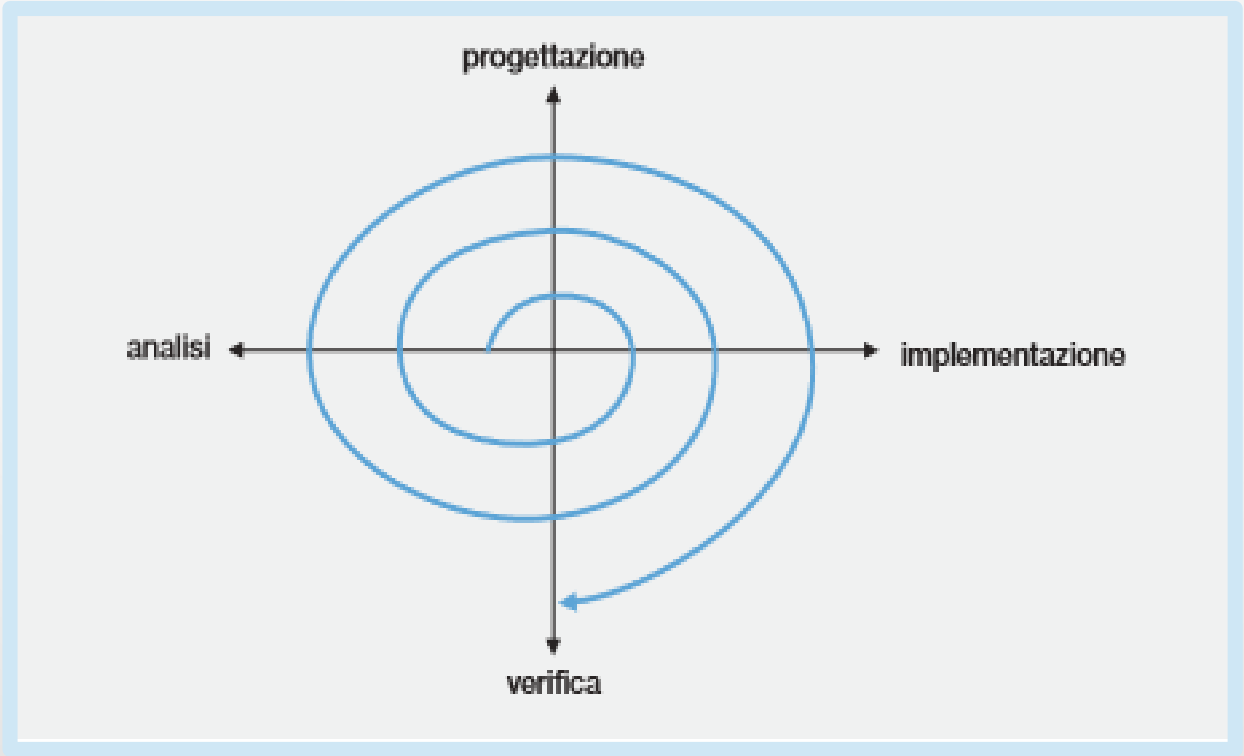
4 ITI A

Argomenti trattati

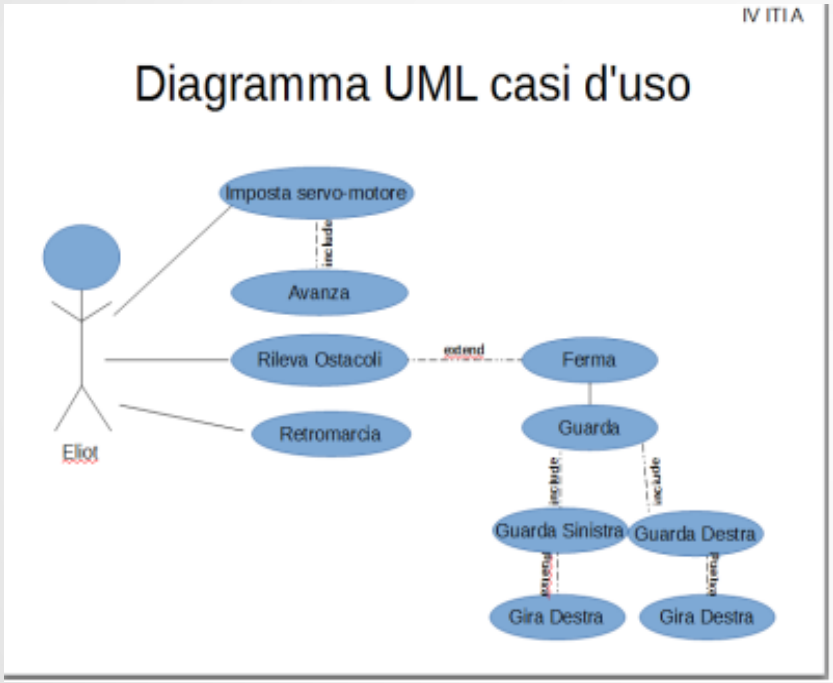
- Ciclo di sviluppo di un software
  - Analisi dei requisiti
  - Progettazione
  - Codifica
  - Test
- UML
  - Diagramma dei casi d'uso
  - Diagramma delle classi
- Ambiente di sviluppo integrato
- Software di sviluppo collaborativo (GIT)
- Documentazione software
- Licenze d'uso

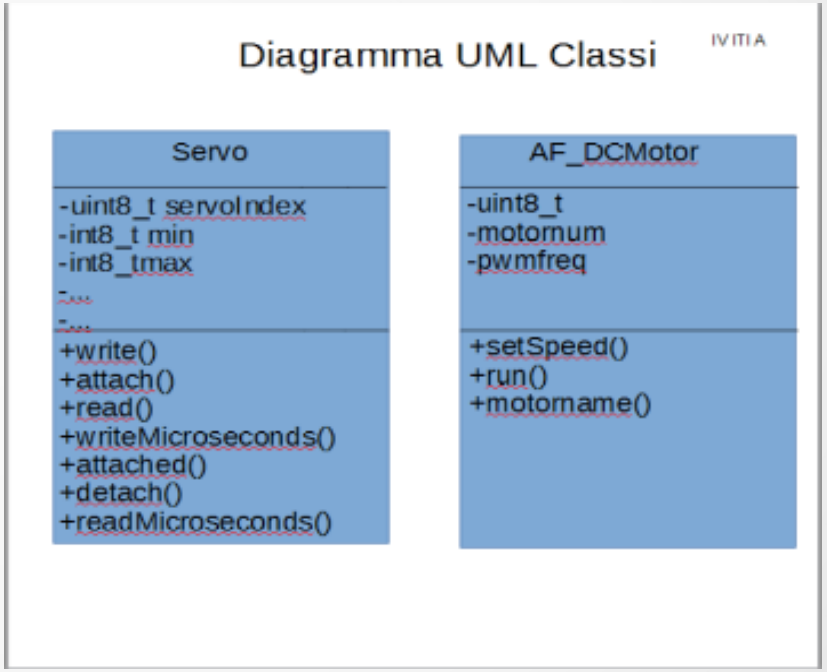
Ciclo di sviluppo

- Il ciclo di sviluppo del software è l'insieme della attività e delle azioni da intraprendere per realizzare un progetto software



Unified Modeling Language

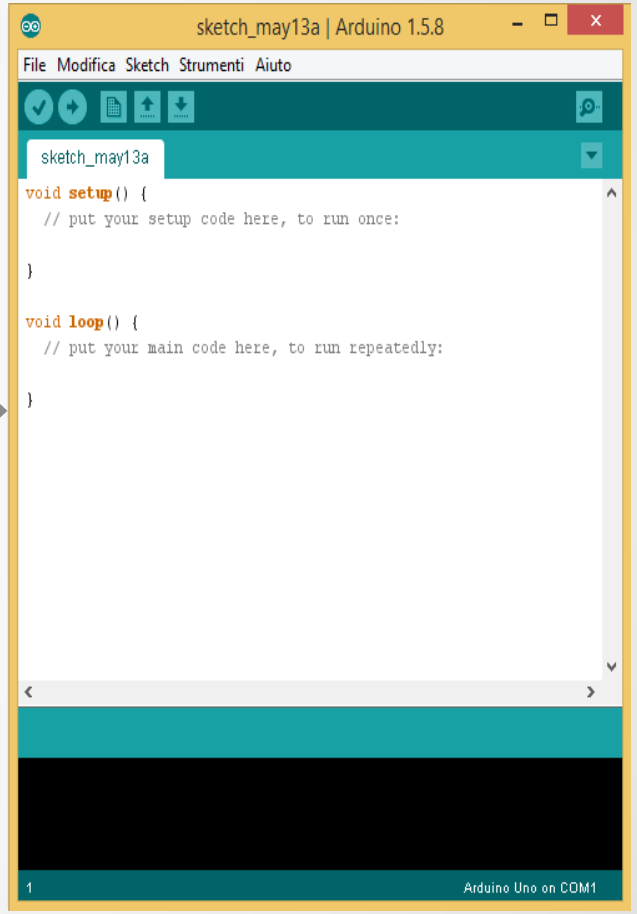




Ambiente di sviluppo integrato

Un ambiente di sviluppo integrato comprende

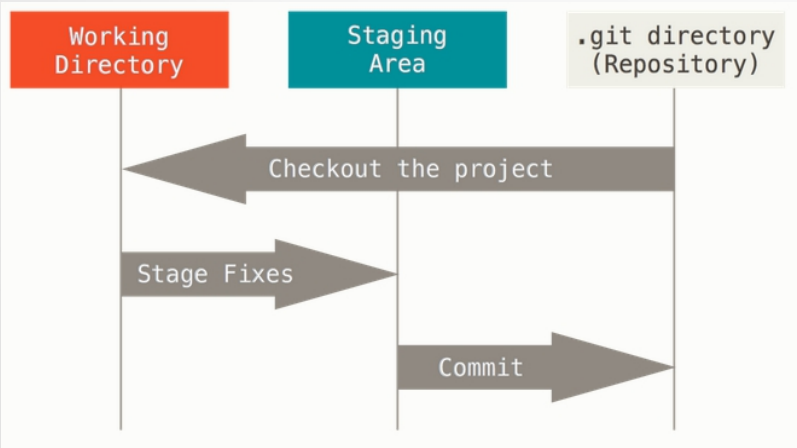
- un editor
- un compilatore
- un debugger
- un performance-profiler
- un analizzatore
- un editore visuale
- un editor UML
- un sistema di controllo delle versioni



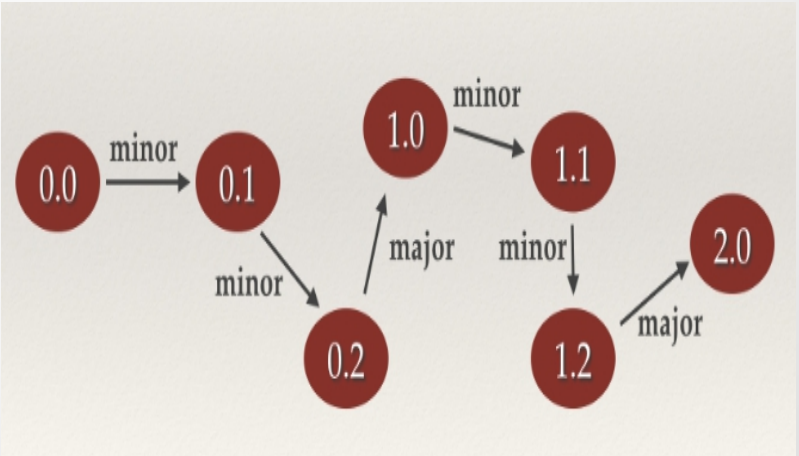
GIT

GIT è un software open source di controllo di versione dove ogni programmatore può lavorare alla propria porzione di codice e poi fondere il proprio lavoro con i contributi degli altri

Funzionamento salvataggio dei dati



Aggiornamento delle versioni



Documentazione software

Tutti i progetti software adottano convenzioni di stile per la codifica finalizzata a massimizzare la leggibilità e la comprensibilità del codice sorgente

- Regole di denominazione generali
- Nomi dei file
- Nomi dei tipi
- Nomi delle variabili
- Nomi di costanti
- Nomi di funzioni
- …..

Licenze d'uso

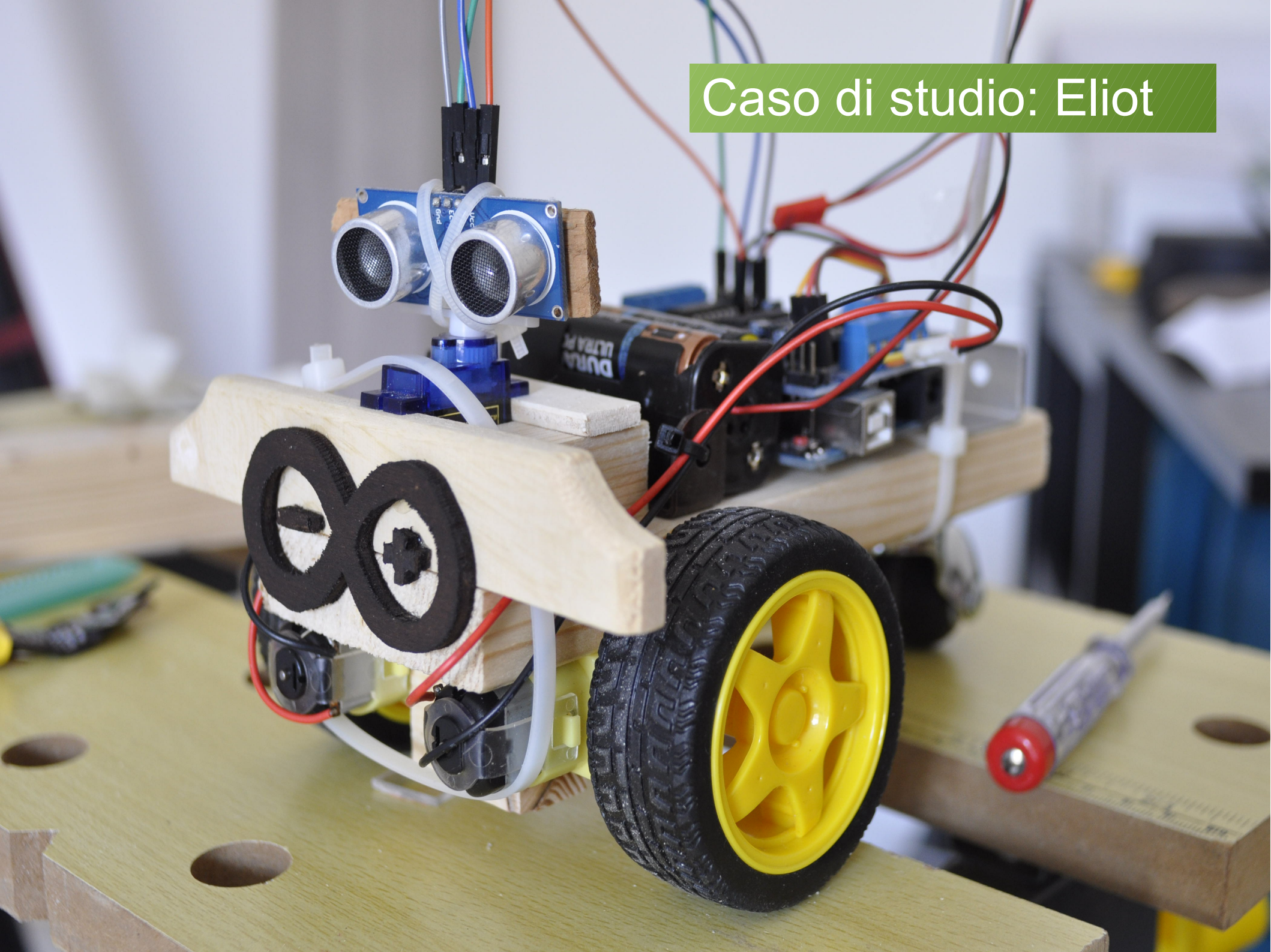
La licenza o contratto d'uso è Il contratto con il quale il titolare dei diritti di sfruttamento sul software definisce le limitazioni e l'utilizzo e nella cessione dell'opera

Le licenze possono essere di due tipi:

- open-source, (GPL,BSD, CC. Ecc)
- licenze proprietarie (copyright)

La licenza da noi utilizzata è la licenza GPL

Caso di studio: Eliot



File inclusi nel pacchetto git

- Readme
- Requisiti software del codice
- Diagramma UML del codice
- Algoritmo (parziale) di progettazione del codice
- Codice Arduino
- Licenza

https://github.com/doceo/Eliot

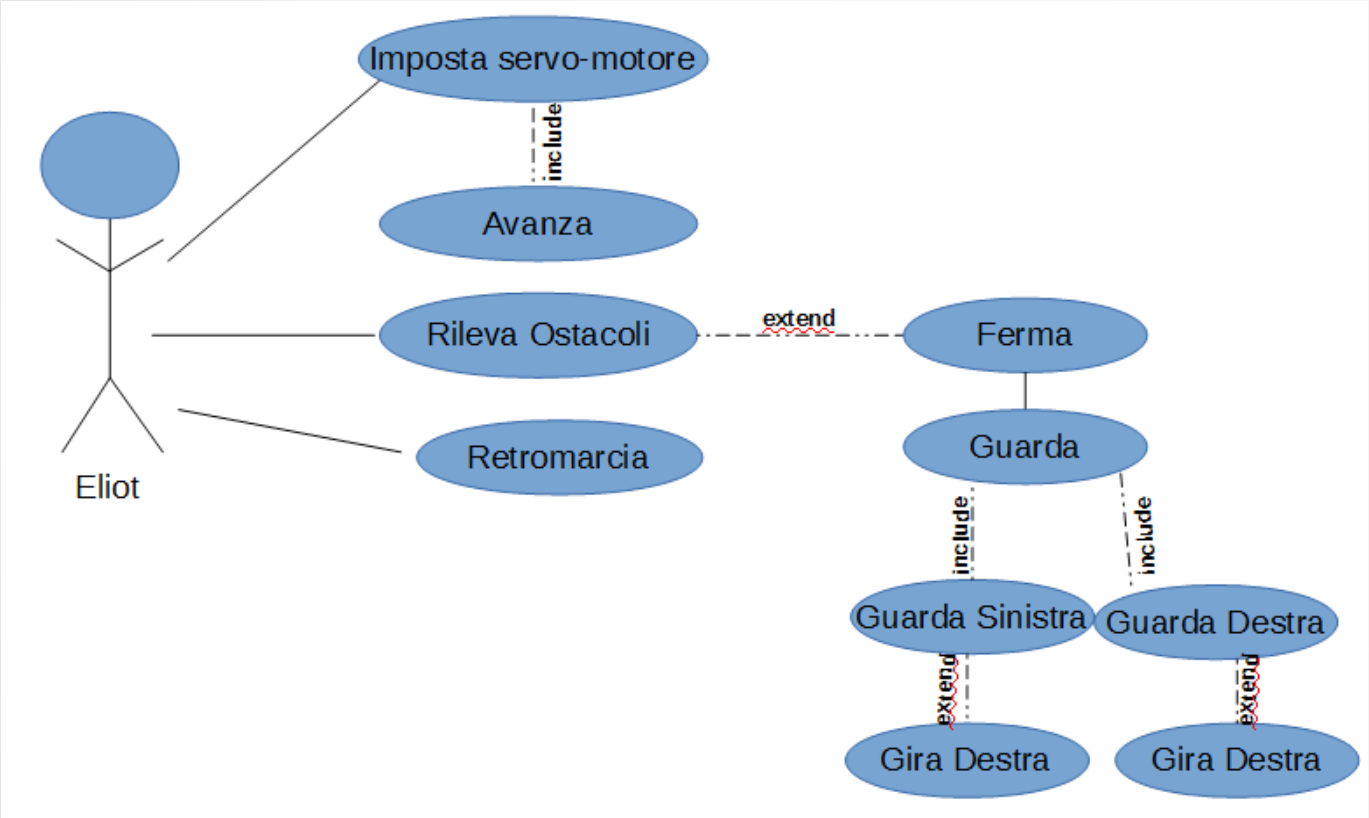
Readme

```
=== Eliot ===
Autori: Presenti nel codice di Arduino
Tag: Motori, Arduino, Macchina
Versione Stabile:1.0
Licenza:GPL 3.0

== Descrizione ==

Il codice è stato scritto per scopi didattici, da parte di una 4 ITI di Napoli, seguiti dai professori:
Mazzone Diomede;
Orecchio Alessandro.

La macchina ha:
La ricerca dell'ostacolo;
Emissione di un suono all'ostacolo trovato, con distanza uguale a 30cm;
La conversione dei valori in cm;
Le funzioni rispettive per comandare i motori, e il servo-motore;

Requisiti:
*       3 Ruote;
*       2 Motori DC;
*       1 Servo-Motore;
*       1 Sensore IR;
*       Cavetti di Collegamento;
*       Arduino 1 rev 3;
*       Motor Shield Adafruit.

== Istallazione ==

Per installare il seguente codice per una prorpia macchina, c'è bisogno dell'IDE Arduino,
```

Un file readme è generalmente caratterizzato da:

- ***Istruzioni per la configurazione***
- ***Guida all'installazione***
- ***Un elenco dei file contenuti nel pacchetto***
- ***Informazioni riguardo al tipo di licenza e il copyright***
- ***Contatti del distributore e del programmatore***
- ***Eventuali bug (conosciuti)***
- ***Risoluzione di problemi***
- ***Ringraziamenti***
- ***Un changelog***

Diagramma UML casi d'uso



Requisiti software del codice

Tabella dei requisiti versione 1.0 di Eliot

Requisito	Tipologia	Priorità	Definizione
1	Tecnologico	Must	L'accensione avviene alimentando i motori e la scheda Arduino.
2	Tecnologico	Must	Lo spegnimento dell'auto avviene staccando le batterie.
3	Funzionale	Must	L'auto all'avvio sincronizza velocità e rotazione delle ruote e orienta il servo-motore a 90°.
4	Funzionale	Must	Se : - incontra un ostacolo ad una distanza di 30 cm / - un'angolazione compresa tra 90° +/- alfa la macchina si ferma.
5	Funzionale	May	L'auto viene guidata utilizzando gli infrarossi.
6	Tecnologico	May	Controllo del dispositivo tramite Wi-Fi
7	Tecnologico	Must	L'auto deve avere due motori di tipo DC.
8	Tecnologico	Must	L'auto deve avere tre ruote.
9	Tecnologico	Must	L'auto deve avere due batterie: 1 da 6V(4*1.5V) per il motore; 1 da 9V per Arduino.
10	Tecnologico	Must	Il software dell'auto è presente su Arduino.
11	Funzionale	Must	Deve tenere sempre traccia se l'ostacolo più vicino è a destra o a sinistra.
12	Tecnologico	Must	La distanza va calcolata con un sensore ad infrarossi.

Licenza GPL 3.0

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

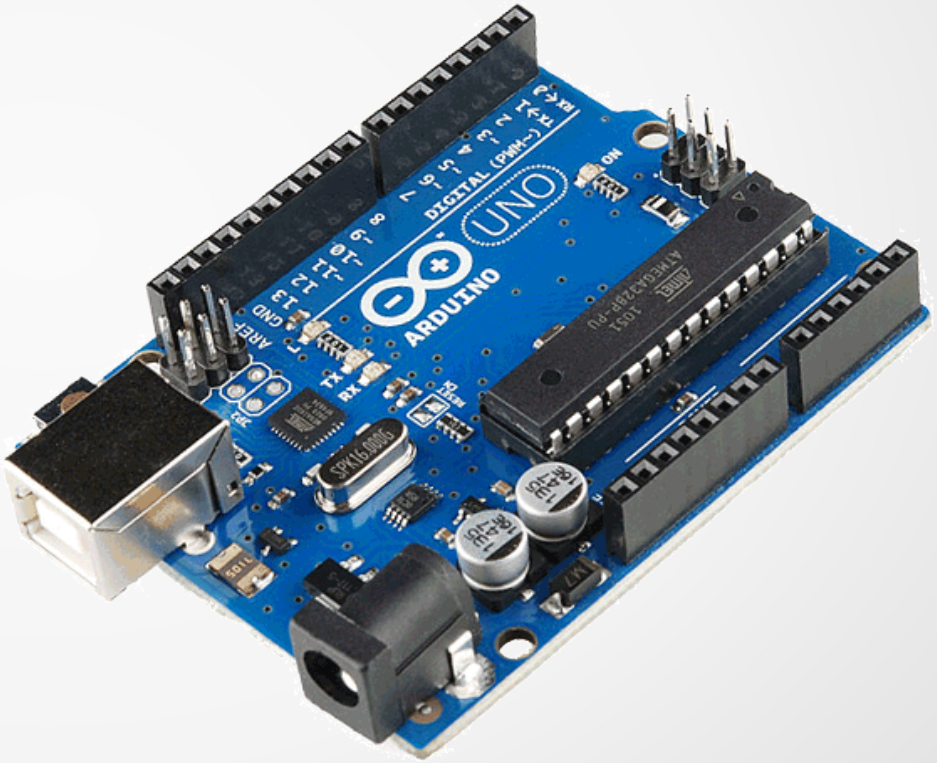
When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Codice Arduino

```
#include <AFMotor.h>
#include <Servo.h>

Servo myservo;                                  // Crea la classe servo per controllare l oggetto .
int triggerPort=A3;                             // Sensore che invia il segnale .
int echoPort = A5;                              // Sensore che riceve il segnale inviato dal trigger .
int cmconv = 59;                                // Calcolo distanza dell'onda .
int pos_in = 90;                                // Variabile che definisce la posizione iniziale a 90° del servo-motore .
int ang_mov=30;                                 // Variabile che definisce l'angolazione del servo in movimento .
int vel=200;                                    // Variabile che definisce la velocità .
int ost_S;                                      // Variabile che trova l'ostacolo a sinistra .
int ost_D;                                      // Variabile che trova l'ostacolo a destra .
int ang_fer=75;                                 // Variabile che indica i gradi dell'angolo fermo .
int dist_min=30;                                //Variabile che definisce la distanza minima dell'oggetto .
boolean trovato;                                //Variabile che trova l'oggetto .

AF_DCMotor motor1(1);                           //Dichiarazione oggetto motore destro .
AF_DCMotor motor2(2);                           //Dichiarazione oggetto motore sinistro .
   // Inizializzo la variabile per contenere il pin collegato al buzzer
 int buzzer = A0;

void setup() {
    myservo.write(pos_in);                      // Il servo ruota nella posizione iniziale .
    pinMode(triggerPort, OUTPUT);               // Il trigger viene inserito ad una porta input .
    pinMode(echoPort, INPUT);                   // L'echo viene inserito ad una porta input .
    digitalWrite(triggerPort, LOW);             // Porta a 0 l'input del trigger .
    myservo.attach(9);                          // Attaccare il servo nel pin 9 per l' oggetto.
```

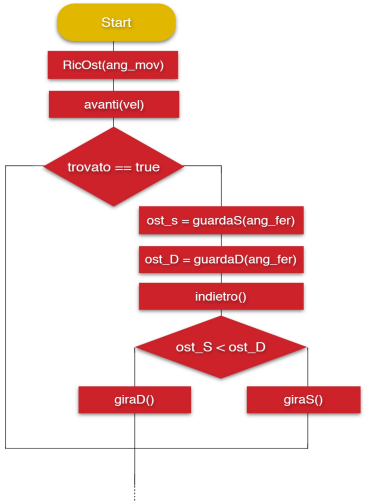
Strumenti utilizzati

- Arduino 1 rev. 3
- Motor Shield Adafruit
- 2 MOTORI di tipo “DC”
- Servo motore
- Sensore distanza HC SR04
- 2 BATTERIE
  - 6 v per i motori
  - 9 v per Arduino
- 2 ruote motrici
- 1 ruota d'appoggio
- Cavetti di collegamento

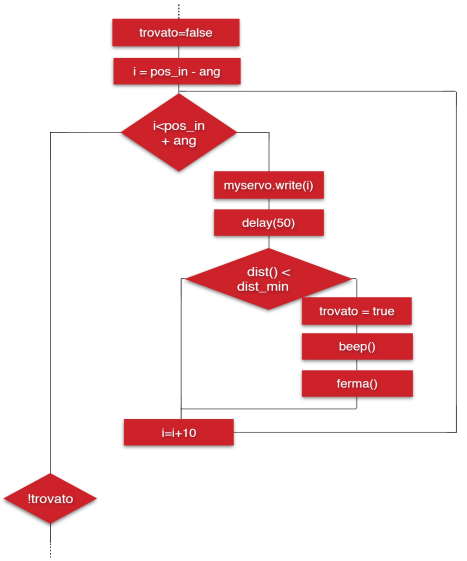


Algoritmo Eliot

loop()



RicOst(ang)



IDE Arduino

Struttura

Ordine

Funzioni

Il software è suddiviso in funzioni necessarie al corretto funzionamento di Eliot

- dist () ; (RILEVAZIONE DISTANZA)
- beep () ; (SUONO AL RILEVAMENTO DI UN OSTACOLO)
- RicOst () ; (RICERCA OSTACOLO)
- marcia_avanti () ; (FUNZIONE PER LA MARCIA IN AVANTI)
- marcia_dietro () ; (FUNZIONE PER LA MARCIA INDIETRO)
- gira_S () ; (FUNZIONE PER SVOLTARE A SINISTRA)
- guarda_S () ; (FUNZIONE PER GUARDARE A SINISTRA)
- gira_D () ; (FUNZIONE PER SVOLTARE A DESTRA)
- guarda_D () ; (FUNZIONE PER GUARDARE A DESTRA)
- frenata () ; (FUNZIONE PER FRENARE)

FUNZIONE DIST ()

Permette di rilevare la distanza di un oggetto dalla macchina in centimetri, dando un segnale di input al sensore ogni 50 microsecondi

```
int dist(){

int distMax = 30;
int trovato = false;                                    // imposta la variabile che restiuisce a false

digitalWrite (triggerPort, HIGH);                       // attraverso il trigger inizia a emettere onde
delayMicroseconds(10);                                  // per dieci secondi
digitalWrite(triggerPort,LOW);                          // e si ferma

long duration =pulseIn(echoPort, HIGH);                 //attraverso la funzione pulseIn acquisiamo il segnale tramite il sensore

long int distanza = 0.036 * duration /2;                //calcoliamo la distanza

Serial.print(" durata: ");                              //stampiamo sul monitor seriale la durata del segnale e la distanza ottenuta
Serial.println(duration);
Serial.print(" distanza: ");

if (duration >38000)     {
        Serial.println("fuori portata");                //segnaliamo se la distanza è fuori dalla portata dello strumento

}else {
         Serial.print(distanza);
         Serial.println ("cm");
         Serial.println (" ");
       }

//evitiamo una divisione per zero, sostituendo lo zero con 1000
if (duration == 0)
   duration == 1000;

   long int rval = microsecondsToCentimeters(duration);  //calcoliamo la distanza in centimetri


 return rval;
```

FUNZIONE BEEP ()

Permette alla macchina di emanare un suono (beep) quando rileva un ostacolo ad una distanza di almeno 30 cm

```
void beep ()
{

  analogWrite(buzzer, 300);                    //indica al pin buzzer un valore analogico pari a 300 .
  delay(100);                                  //ritarda il suono di 100 millisecondi .
   analogWrite(buzzer, 0);                     //azzera l'output riferita al pin buzzer .

}
```

FUNZIONE RicOst ()

Permette alla macchina in movimento di ricercare l'ostacolo tramite sensori posti sul servomotore che ruota ad un'angolazione specifica

Per poter fare ciò abbiamo innestato all'interno della funzione appositi cicli

```
bool RicOst()
{
  for (int i = pos_in - ang_mov ; i <= pos_in + ang_mov; i = i + 15) // ciclo di for per la rotazione del servo .
  {
    myservo.write (i);    // comanda al servo di angolarsi in base alla misura stabilita nel ciclo di for .
    delay (50);           // ritarda la rotazione di 10 millisecondi .
      if ( dist()< dist_min )      // scelta per emissione del suono alla rilevazione dell'ostacolo .
       {
         trovato=true;
         beep();          // Funzione di avvertimento tramite un beep .
         frenata();
       }
   }
   if (trovato==false)
   {
     for (int i = pos_in + ang_mov ; i >= pos_in - ang_mov; i = i - 15)
     {
        myservo.write (i);
        delay (50);
        if ( dist()< dist_min )
        {
          trovato=true;
          beep();
          frenata();
        }
     }
   }
}
```

FUNZIONE marcia_avanti ()

Permette alla macchina di muoversi in avanti grazie ad appositi motori collegati alle ruote e specifici comandi

```
void marcia_avanti(int vel) {        //Funzione per far muovere la macchina verso avanti
motor1.run(FORWARD);
motor1.setSpeed(vel);
motor2.run(FORWARD);
motor2.setSpeed(vel);
}
```

FUNZIONE marcia_dietro ()

Permette alla macchina di muoversi indietro grazie ad appositi motori collegati alle ruote e comandi che permettono la retromarcia

```
void marcia_dietro() {                  //Funzione per far muovere la macchhina verso dietro
motor1.run(BACKWARD);
motor1.setSpeed(vel/2);
motor2.run(BACKWARD);
motor2.setSpeed(vel/2);
delay (500);
frenata();
}
```

FUNZIONI gira_S () e gira_D ()

Permettono alla macchina di poter girare a sinistra o a destra grazie ad appositi comandi

```
 void gira_S()
{
  motor2.run(FORWARD);
  motor2.setSpeed(vel);
  motor1.run(RELEASE);
  delay(500);
  frenata();

  }
```

```
 void gira_D()
{
  motor1.run(FORWARD);
  motor1.setSpeed(vel);
  motor2.run(RELEASE);
  delay(500);
  frenata();
  }
```

FUNZIONE guarda_S () e guarda_D ()

Sono state create per gestire lo spostamento della macchina in modo tale da scegliere la giusta direzione paragonando le distanze e scegliendo quella più opportuna

Quindi tramite sensore e un oscillazione di angolazione specifica e due appositi cicli guarda a destra e a sinistra

```
int guarda_S ()
{
  int locale=200;
  for(int i = pos_in - ang_fer; i<=pos_in; i = i + 10)
  {
    myservo.write(i);
    locale=dist();
    if (locale<dist())
    {
      locale=dist();
    }
    delay(500);
    return locale;
  }
}
```

```
int guarda_D ()
{
  int locale=200;
  for(int i = pos_in + ang_fer; i<=pos_in; i = i - 10)
  {
    myservo.write(i);
    locale=dist();
    if (locale<dist())
    {
      locale=dist();
    }
    delay(500);
    return locale;
  }
}
```

FUNZIONE FRENATA ()

Permette alla macchina di potersi arrestare quando necessario

```
void frenata() {                    //Funzione per far fermare la macchina
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  delay (500);
}
```

Test algoritmo

Bug riscontrati

- Non sceglie la direzione dove girare in modo corretto
- Non ha una convergenza di ruote corretta (risolto via software)