Muletto robotico, il software

Cosa ci serve

- Far muovere il robot
- Inseguire la linea
- Leggere il colore

Controllo di un motore

Servono componenti esterni. Arduino da solo non è abbastanza potente per pilotare un motore

- Arduino Motor Shield

Dobbiamo controllare direzione e velocità

- Direzione: un'uscita digitale -> HIGH: un verso di rotazione, LOW: il verso opposto
- Velocità: un'uscita PWM -> 0 = fermo, 255 = velocità massima. come luminosità di un LED

```
void setup() {
  int pinVel = 2; // pin a cui è collegato l'ingresso PWM
  int pinDir = 3; //pin a cui è collegato l'ingresso DIR
  pinMode(pinDir, OUTPUT);
}

void loop() {

  digitalWrite(pinDir, HIGH); //imposto una direzione
  analogWrite(pinVel, 125); //velocità al 50%
}
```

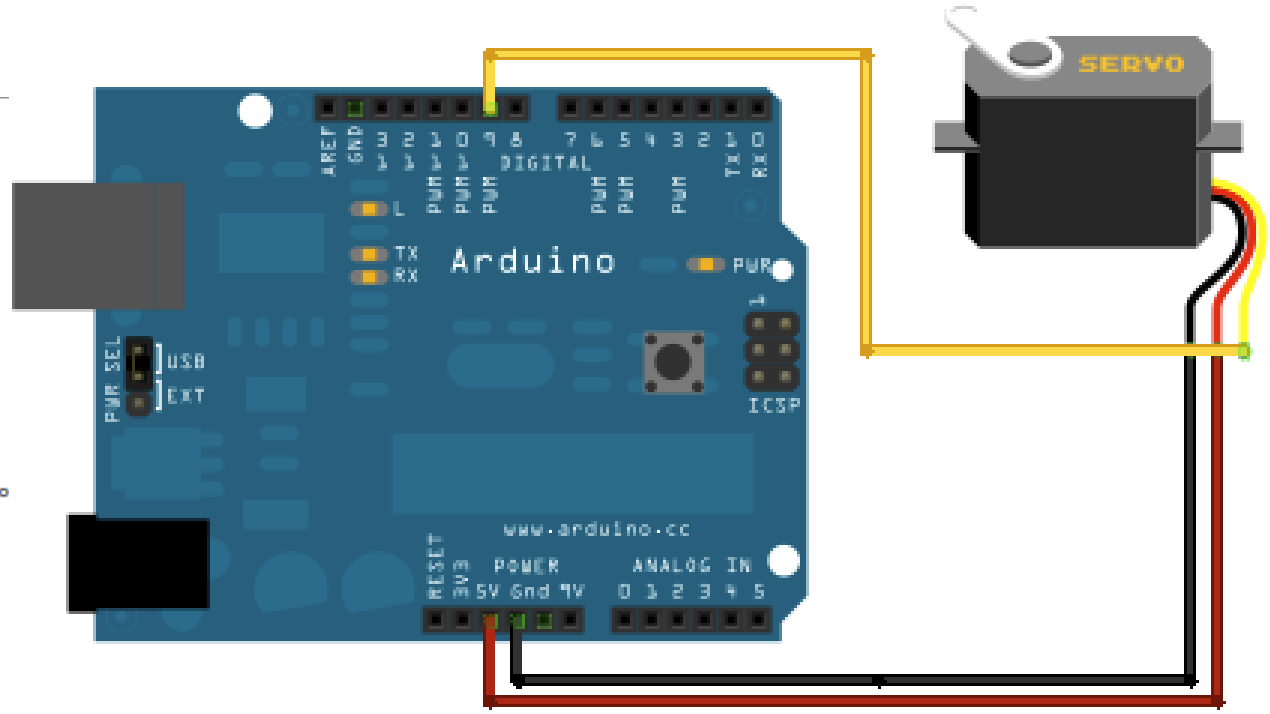
Il Servomotore

- Ha una sua elettronica interna di controllo
- Si usa un solo PIN
- Ha una sua libreria su arduino

```
#include <Servo.h> //includo la libreria
Servo mioServo;  //creo oggetto servomotore
int pos = 30;    // posizione, in gradi, tra 0 e 180°

void setup() {
  mioServo.attach(9);  // attacco il servo al pin 9
}

void loop() {
  mioServo.write(pos); //scrivo la posizione
}
```



Il Sensore di colore

```
#include <COLORPAL.h> //Includo la libreria
int rosso;  //CREO LE 3 VARIABILI IN CUI MEMORIZZARE LE COMPONENTI RGB
int blu;
int verde;
Colorpal colorp(10);  //CREO LA COLORPAL CON IL PIN sig COLLEGATO AL PIN 2 di arduino

void setup() {
  colorp.init(); //Inizializzo il sensore
}
void loop() {
  colorp.readRGB(rosso, verde, blu); //leggo le componenti di colore
  //saranno memorizzate dentro "rosso", "verde", e "blu"
}
```

In alternativa si può usare

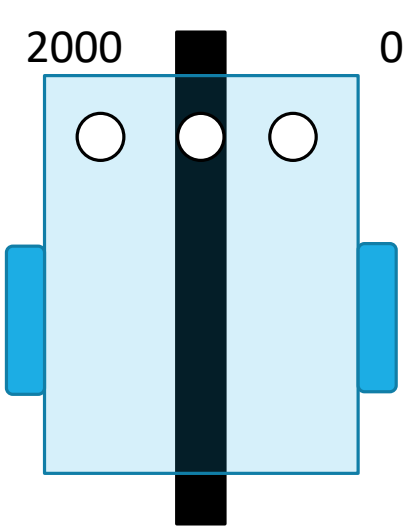
```
void loop() {
  rosso = colorp.readR();
  verde = colorp.readG();
  blu = colorp.readB();
}
```

Muovere un motore	digitalWrite(DIRpin, direzione); //direzione
	analogWrite(PWMpin, vel); //velocità tra 0 e 255

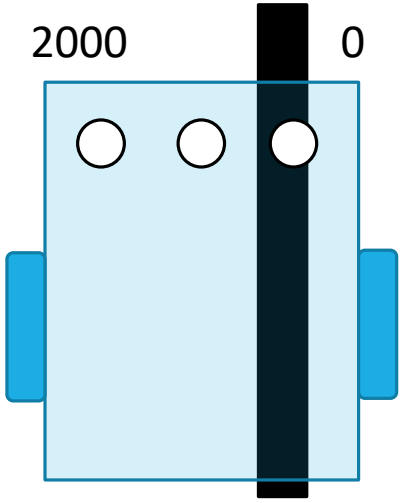
Muovere il servomotore	Servo myServo
	myServo.attach(PIN);
	myServo.write(pos) //tra 0 e 180, 90 = fermo

Leggere il sensore di linea	position = qtr.readLineBlack(sensorValues);
Leggere il sensore di colore	Colorpal color(PIN) color.init() Rosso = color.readR(); Verde = color.readG(); Blu = color.readB();

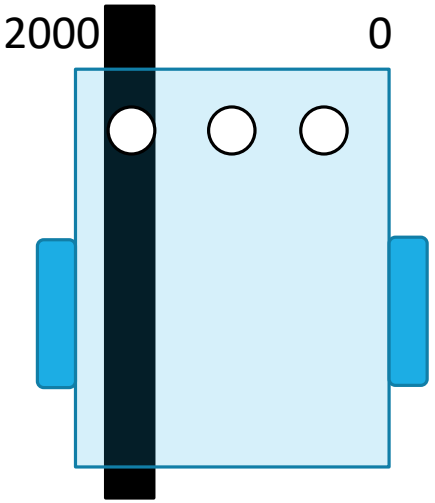
La logica del line following



pos = 1000



pos ≈ 0



pos ≈ 2000

Line following a soglie

Agisco su velocità di Dx e Sx.

- Se **pos** = 1000 -> non cambio nulla
- Se **pos** è minore di 500 -> SX +25 DX -25
- Se **pos** è maggiore di 1500 -> SX -25 DX +25
- Se **pos** = 0 -> SX +50 DX -50
- Se **pos** = 2000 -> SX -50 DX +50

Line following a soglie

Agisco su velocità di Dx e Sx. Calcolo **err = 1000 – pos ;**

- Se **err** = 0 -> non cambio nulla
- Se **err** è minore di -500 -> SX +25 DX -25
- Se **err** è maggiore di +500 -> SX -25 DX +25
- Se **err** = -1000 -> SX +50 DX -50
- Se **err** = 1000 -> SX +50 DX -50

ERR	Vel DX
-1000	-50
-500	-25
0	0
500	25
1000	50

Un altro approccio

err	Vel DX
-1000	-50
-500	-25
0	0
500	25
1000	50

err	Vel DX	
-1000	-50	= 0,05 · (-1000)
-500	-25	= 0,05 · (500)
-100	-5	= 0,05 · (100)
0	0	= 0,05 · (0)
100	5	= 0,05 · (100)
500	25	= 0,05 · (500)
1000	50	= 0,05 · (1000)

Non abbiamo bisogno di tutti quegli if
Possiamo sapere quanto correggere la velocità con una semplice moltiplicazione

Controllore proporzionale

```
void loop() {
  //linefollow
  position = qtr.readLineBlack(sensorValues);
  error = 1000 - position;
  vel = K * error;
  analogWrite(PWMA, (basevel - vel));
  analogWrite(PWMB, (basevel + vel));
```

Il resto del software

Aspetto che venga premuto un pulsante

Quando viene premuto un colore inizio a seguire la linea

- Seguo linea

Quando trovo una segno di arresto mi fermo e controllo colore

Se il colore è sbagliato allora torno a seguo linea

Se il colore è giusto :

- Ruoto a destra
- Mi avvicino e sollevo il carrello
- Torno indietro e ruoto di nuovo a destra
- Torno a seguilinea