Sommario

1 3

2 4

3 5

Arianna e i suoi fratelli 5

chi guida? 5

La guida locale, valutazione e misura 6

Sensori 6

Il guidatore ovvero il sistema di controllo 7

Prossimi passi 8

4 9

Il ritorno di Arianna. 9

Arianna oggi 9

Come sterza Arianna 11

5 15

Un sistema per correre parallelo alla libreria 15

Come guidiamo la macchina? 16

Controllo in anello chiuso 16

il codice del controllo 17

6 19

Il trasduttore meccanico 19

Sensore elettrico 19

Conteggio hw o sw debounce 21

L’interrupt 21

Traduzione in spazio 23

7 25

Pan e Tilt 25

il software per comandare i servo 26

La meccanica 26

Arianna indica il libro 27

La posizione di Arianna 28

La formula 29

Codice 30

Risultato e Problemi 33

9 35

10 IoT + Robotics = Cloud Robotics 37

# 1 come nasce Arianna

*Arrivo in biblioteca e chiedo “Robinia Blues” di Valter Binaghi, il bibliotecario consulta il computer e mi  informa che c’è e lo posso trovare tra gli autori italiani. Un attimo di imbarazzo, non frequento questo luogo abbastanza da conoscere la disposizione degli scaffali, faccio per domandare ma il mio interlocutore mi anticipa dicendomi di seguire Arianna. Arianna ? Un sibilo mi fa voltare, vedo una macchinina che parte lungo il corridoio, porta la scritta Arianna. Al mio sguardo interrogativo il mio interlocutore mi fa cenno di andare, segua Arianna mi dice. Tutto dura pochi secondi nei quali valuto anche la possibilità di essere vittima di “Scherzi a parte”, Arianna si ferma, si gira un poco e con un altro fruscio un puntatore luminoso si muove, si accende e quando decido di guardare dove Arianna sembra “indicare” vedo una luce sui libri e li vicino il mio “Robina Blues”. Lo prendo, lo sfoglio e me ne vado un poco stupito, saluto il bibliotecario e nel frattempo risento il sibilo. Vedo Arianna che torna da dove era venuta, la vedo accoccolarsi in una piccola casetta e io lascio la biblioteca pronto a rileggere il libro di un bravo insegnante che ha attraversato i nostri luoghi*.

Arianna è un robot che è in grado di muoversi sul pavimento e andare dove gli viene richiesto. E’ il progetto che abbiamo avviato all’interno di StuffCube. La prima parte è stato definire cosa deve fare, lo abbiamo descritto nell’incipit. Successivamente abbiamo scomposto il progetto nelle diverse funzioni di considerando le caratteristiche della biblioteca che ci ospita:

* Realizzare oggetto che si muove (*rover* cioè vagabondo)
* sensori sul rover
* ricevimento comandi di destinazione
* determinare dove deve andare
* determinare dove è
* determinare il moto verso la destinazione
* indicare il libro
* ritorno a casa
* ricarica
* la nostra bilbioteca ha degli scaffali su ruote, questo aggounge un nuovo compito, mappare gli scaffali quando vengono spostati.

Altri vincoli di progetto che ci siamo dati sono di essere il meno intrusivi possibile, questo fa si che dobbiamo evitare soluzioni che richiedano sorgenti di alimentazione, oppure il porre strisce guida sul pavimento e cosi via.

Inizieremo partendo dagli scaffali più semplici, quelli che affiancano la parete. L’idea è seguire lo scaffale viaggiandoci radente e identificando dove siamo misurando lo spostamento e usando dei segni già presenti sulla struttura. Guardare cosa hanno fatto altri è utile e istruttivo. Siamao partiti da: [Un robot che segue le pareti](http://www.allaboutcircuits.com/projects/how-to-build-a-robot-follow-walls/)

# 2 il punto di vista del robot

*Arianna, l’aiuto bibliotecario dove lavora? La biblioteca dove lavora Arianna è un ampio spazio aperto su un solo piano e ha un bel pavimento in parquet laminato. I libri sono ospitati da un lungo scaffale disposto lungo e da alcuni scaffali montati su rotelle. Ci sono sia tavoli che ospitano dei pc fissi che tavoli per lo studio. Alcune poltroncine e un pianoforte completano l’arredo. I vari elementi sono facilmente spostabili per adattare lo spazio a varie esigenze, studio, conferenze, laboratori. Questo almeno è quello che vediamo noi, lo vediamo coi nostri occhi e la  consuetudine che abbiamo con l’ambiente. Arianna è diversa, la sua percezione del mondo è dovuta ai suoi sensori e le sue scelte sono frutto di quello che scriveremo nel software.*

Il punto di vista è importante, già abbassandoci al livello di Arianna vediamo le cose in maniera diversa. Iniziamo dal movimento. Il prototipo zero ha due ruote motrici posteriori ognuna pilotata da un proprio motore, davanti una sfera funge da terzo appoggio. Controllando opportunamente i due motori siamo in grado di regolare sia la velocità che la direzione. Arianna zero usa due motori stepper derivati da dei lettori CD. Gli stepper sono motori nei quali possiamo imporre rotazioni in maniera controllata. Questo ci permette di conoscere anche l’entità dello spostamento, questa informazione ci permetterà di sapere dove siamo sul pavimento.

[](https://stuffcube.files.wordpress.com/2016/11/20161116_213145.jpg)

il punto di vsita di Arianna

Le ruote vengono da una scatola del Meccano. I supporti sono lamiera di allumino taglia e piegata. Mollette, viti e distanziali permettono di montare il tutto su delle piastre in PVC.

Abbiamo pronta la parte meccanica. Il prossimo passo è verificare il movimento, come curva e stimare la capacità di percorrere ripetutamente uno stesso percorso senza l’ausilio di sensori.

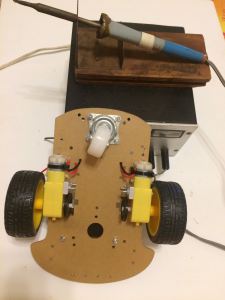
[](https://stuffcube.files.wordpress.com/2016/11/20161116_213259.jpg)

Arianna di fronte agli scaffali

# 3 vari prototipi. Arianna e i suoi fratelli

Tre sono i prototipi in corso di costruzione, servono a provare in campo diverse soluzioni di motorizzazione e guida.

**Arianna 00** ha come base una macchinina giocattolo, ha un motore DC posteriore per il movimento e il classico sterzo anteriore a due ruote come le automobili. Lo sterzo è stato modificato sostituendo l’esistente che ammetteva solo tre possibili direzioni dritto, tutto sinistra, tutta destra con un servomotore che regola a piacere la direzione.  
**Arianna 01, detta anche Ambros**, è invece un tre ruote. Due motrici comandate da due separati motori DC e una terza piroettante anteriore. In questo caso la direzione viene cambiata regolando le velocità dei due motori. Con questa soluzione è possibile ottenere qualunque angolo di sterzata, anche ruotando su se stessa.  
**Ultima è 02**, questa ha la stessa struttura di 01 eccetto che usa motori stepper.

[](https://stuffcube.files.wordpress.com/2016/12/ari-01-20161202-wa0004.jpg)

Arianna 01 detta “Ambros”

Con gli stepper siamo in grado di imporre una rotazione nota delle ruote, possiamo quindi imporre la traslazione e rotazione del robot senza usare sensori sui motori. Con il motore dc per conoscere lo spostamento delle ruote abbiamo bisogno di un sensore di rotazione posto sull’albero del motore.

## chi guida?

Fatta la macchina occorre il guidatore che a sua volta necessità di occhi bene aperti. Il nostro comune guidare l’auto è a ben vedere composto da diverse guide sovrapposte:  
una prima è quella che ci fa seguire la strada evitando muri, marciapiedi e altri ostacoli, un controllo sugli elementi locali, la chiameremo **guida locale**.  
Un livello superiore è quello che ci fa capire dove siamo e quale percorso fare per giungere a destinazione, diamogli il nome di **navigazione** e **posizionamento**.

## La guida locale, valutazione e misura

I questo momento stiamo lavorando sulla guida locale. Il primo test di guida  in corso  è quello di percorre l’intero scaffale diritto con una velocità ragionevolmente elevata e un andamento elegante e sicuro, **il test è detto “della libreria”**.  
Questi due parametri sono il criterio di misura e valutazione del risultato. La velocità deve essere paragonabile a una persona che cammina in un locale, l’abbiamo stimata ragionevole tra i 0.25-0.5 m/s pari a circa 1-2 km/h. L’incedere deve essere lineare senza bruschi cambi di direzione e neppure oscillate a mo’ di ubriaco. Un moto di questo tipo è prevedibile e ben accetto, è conforme alle attese di chi ci segue.

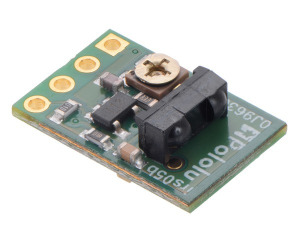
Nei video vediamo una demo di Arianna 00 con risultati promettenti.

<https://youtu.be/69APaa1ccjU>

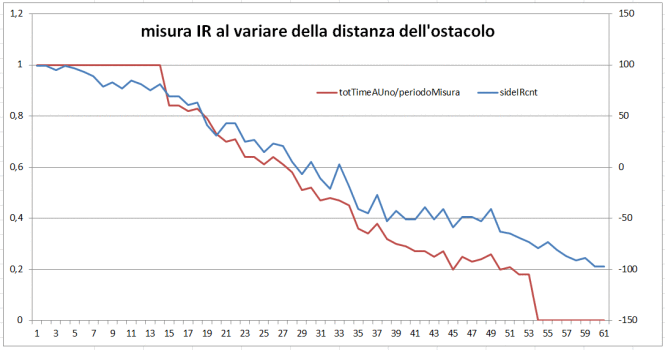
<https://youtu.be/A6ZCeFUlZlU>

## Sensori

Per correre lungo la libreria dobbiamo sapere dove questa si trova, tramite dei sensori posti sul lato del veicolo otteniamo l’informazione. In questo modello abbiamo usato sensori a infrarossi della Pololu 2579 – Pololu 38 kHz IR Proximity Sensor.

[](https://stuffcube.files.wordpress.com/2016/12/ir38khz.jpg)

Questi sono previsti per indicare la presenza o assenza di un ostacolo in un range tra i 30-40 cm. Hanno una banda di incertezza di 10-20 cm dove la loro uscita è instabile.  
**Abbiamo voluto sperimentare se fosse possibile sfruttare questa banda di incertezza per avere una informazione continua sulla distanza**, lo facciamo misurando il tempo in cui il segnale rimane a uno sul tempo di lettura totale.  
Usando sia il rapporto Ton/Tmisura che il numero di transizioni abbiamo ottenuto una misura utile nel passaggio tra presenza e assenza.  
**La funzione ha dato buoni risultati**. Il grafico mostra una misura fatta con un oggetto che si avvicina al sensore. La linea rossa con asse a sinistra mostra la variazione tra ostacolo assente, valore uno, a ostacolo presente, valore zero. Nella regione di transizione viene restituita una informazione significativa. In blu riferito all’asse destro c’è il contatore dello stato del sensore.

[](https://stuffcube.files.wordpress.com/2016/12/misurair_excel1.png)

misurazione ricavata nel campo di incertezza del sensore

Il contatore valore viene incrementato a tempo secondo il suo stato e serve a capire se siamo vicini alla zona di lettura stabile. I queste zone non possiamo estrarre ulteriori informazioni.

|  |  |
| --- | --- |
| 1  2 | **if** (digitalRead(SIDE\_IR)) sideIRcnt++;  **else** sideIRcnt--; |

disponiamo quindi di un’informazione continua che ci dice che siamo tra i 30-50 cm dalla libreria, e che siamo oltre i 50 o al disotto dei 30 cm. Questa è la nostra visione del mondo. sottraendo uno 0.5 abbiamo normalizzato questo numero tra +/- 0.5.

## **Il guidatore ovvero il sistema di controllo**

Noi vogliamo che Arianna viaggi a una distanza dalla libreria che vale 0.5. Per fare questo utlizziamo un **regolatore proporzionale**. La distanza misurata viene sottrata a 0.5, il valore ottenuto è l’errore, usiamo questo numero per pilotare l’angolo dello sterzo.  
Con errore 0 andiamo diritti.  
Quando ci allontaniamo curviamo verso destra per avvicinarci, quando ci avviciniamo andiamo a sinistra.  
Questi continui aggiustamenti ci mantengono sulla retta via. Così come quando impariamo a guidare dobbiamo imparare a dosare la curva per non avere una traiettoria ondeggiante da ubriaco. Più veloci andiamo più dobbiamo essere dolci. **Il guadagno proporzionale del nostro anello è quello che ci serve tarare**.

Attualmente il sistema in assenza di ostacolo si muove verso destra cercando la libreria.

## **Prossimi passi**

Le varie attività proseguono in parallelo a seconda delle difficoltà e disponibilità di tempo delle persone. Non c’è quindi un cronoprogamma rigoroso, c’è invece un lavoro flessibile e guidato dalle opportunità. Possiamo però indicare le previsioni a breve.

Appena pronti anche gli altri prototipo saranno sottoposti al test della libreria. Abbiamo visto che il sensore frontale deve avere un raggio di azione di almeno un paio di metri e lo sostituiremo con uno a ultrasuoni. Possiamo iniziare a definire una strategia di utilizzo dei sensori per navigare in assenza di informazioni, in altre parole se i miei sensori non vedano nulla dove vado? Il tema si chiama **posizionamento all’interno di locali (indoor positioning).**

# 4 struttura scelta

L’articolo precedente su Arianna, il robot bibliotecario, risale a dicembre. Anche se apparentemente i cambiamenti sono pochi per il nostro robot letterato la frase “tutto cambia affinché nulla cambi”  non vale.  Un grosso lavoro è stato fatto, molte le cose imparate e che condivideremo con voi. La cadenza degli articoli sarà settimanale e manterremo la continuità.StuffCube è un luogo di incontro reale e non virtuale, siamo il mercoledì e giovedì sera alla biblioteca di Canegrate e l’ingresso è libero.Arianna serve per imparare come si fa un robot, non per usarne uno già fatto!

A dicembre Arianna era fatto con una meccanica basata su sterzo di [**Ackerman**](http://www.modellismo.net/articoli/assetto.html), il classico sistema di curvatura delle automobili, e aveva la trazione posteriore, era capace di seguire lo scaffale della biblioteca grazie ad un sensore ad infrarossi, misurava la distanza percorsa tramite un sensore sull’asse posteriore, partiva comandato da un bottone. Aveva anche dei fratelli e delle sorelle che avevano scelto motorizzazioni diverse.

## Arianna oggi

**La struttura:** Viste le caratteristiche dei fratelli, Arianna oggi è basata su un kit con due ruote motorizzate e una ruota castor anteriore. Il testo [“Autonomous Mobile Robot Mechanical Design”](http://mech.vub.ac.be/multibody/final_works/ThesisKristofGoris.pdf)  
contiene una chiara presentazione dei vari tipi di scelte fattibili sul movimento di macchine, partendo da un semplice robottino sino a una ruspa passando attraverso l’Ape car. Leggendolo abbiamo scoperto le ruote ominidirezionali, mostrate nel video seguente

<https://youtu.be/IlmKcohyXG0>

Tornando a noi la meccanica è basata su un [kit](https://www.amazon.it/gp/product/B01F3OHY1U/ref=oh_aui_detailpage_o00_s00?ie=UTF8&amp;psc=1) reperibile su Amazon, kit che era stato usato da “Ambros”. È un sistema a tre ruote con due ruote fisse, motorizzate e indipendenti, la terza è una ruota piroettante castor.

[](https://stuffcube.files.wordpress.com/2017/02/51eazpp-s2l.jpg)  
È stato scelto per diversi motivi. Il primo era dare la possibilita di avere un oggetto che chiunque potesse facilmente ed economicamente riprodurre. L’arianna ottenuta modificando una macchinina giocattolo è un esercizio istruttivo ma richiede tempo e manualità che non tutti hanno. Questo permette di fare esperienza e sviluppare software in poco tempo.  
Inoltre la struttura a due motori separati permette un mobilità molto superiore allo sterzo classico.  Il kit contiene anche i dischi encoder che servono per rilevare lo spostamento delle ruote e da questo identificare dove va Arianna (**odometria**).

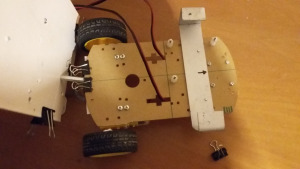
Arrivato il kit è stato adattato per ospitare l’elettronica precedente. La scelta fatta di usare una piastra aggiuntiva che raccogliesse tutta l’elettronica ha permesso un “trapianto abbastanza semplice. Una nuova struttura è stata fatta sul kit per accogliere la piastra elettronica e sono stati collegati i motori, uno destro e uno sinistro, e in sensori degli encoder. La scheda Arduino è passata dal modello “Uno” alla “Mega 2560”. Vedremo nei prossimi articoli che il cambio si è reso necessario per gestire sotto interrupt anche gli encoder delle ruote.

Nelle foto la meccanica attuale. Per le connessioni meccaniche in questo modello si fa largo uso di connessioni usando delle mollette che si trovano in cartoleria.

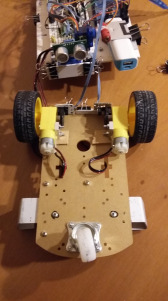
Tramite opportune fessure fatte col seghetto è possibile unire pezzi meccanici in maniera robusta, veloce e facilmente smontabile.

[](https://stuffcube.files.wordpress.com/2017/02/20170213_204817.jpg)

piastra controllo montata su base. Da sinistra, batteria. ATmega 2560, pan e tilt con sonar montato

[](https://stuffcube.files.wordpress.com/2017/02/20170213_204426.jpg)

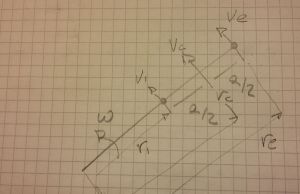
base: vista superiore

[](https://stuffcube.files.wordpress.com/2017/02/20170213_204531.jpg)

base: vista inferiore

## Come sterza Arianna

Avendo due ruote motrici indipendenti è facile intuire che la direzione viene stabilita dalla velocità delle ruote, se vanno alla stessa velocità Arianna procederà diritta, con velocità diverse curverà più o meno. Dall’intuizione passiamo alla pratica che in questo caso è matematica.

[](https://stuffcube.files.wordpress.com/2017/02/formulacurva.jpg)

moto circolare uniforme intorno a un punto. w è la velocità angolare, Vi,Vc, Ve le velocità tangenziali alla distanza ri, rc, re

Ve = re\*w            velocità ruota esterna, re raggio curva ruota e(sterna)  
Vi  = ri\*w            velocità ruota interna, ri raggio curva ruota i(nterna)  
Vc  = rc\*w           velocità punto centrale, rc raggio curva punto centrale

w = Vc/rc            velocità angolare

Ve = re\*Vc/rc  
Vi = ri\*Vc/rc

a = carreggiata, distanza tra le due ruote motrici.

Re = rc + a/2  
Ri = rc – a/2

Ve = (rc + a/s)\*Vc/rc = Vc + Vc\*a/(2\*rc) = Vc\*(1 + a/(2\*rc))  
Vi  = (rc – a/s)\*Vc/rc  = Vc – Vc\*a/(2\*rc) = Vc\*(1 – a/(2\*rc))

Detto:  
S = a/(2\*rc)        scorrimento tra le ruote  
V = Vc                 velocità di Arianna

possiamo scrivere:  
Ve= V\*(1+S)  
Vi= V\*(1-S)

Le formule sopra ci mostrano come imporre la velocità di Arianna (V) su una curva di raggio rc imponendo il parametro S.

* Con S=0 abbiamo le ruote che viaggiano alla stesso velocità e quindi andiamo diritti, notiamo che S diventa zero se il raggio di curvatura rc tende a infinito.
* Se S vale 1 abbiamo che Vi vale zero, Arianna girerà su se stessa con una ruota ferma, la formula ci mostra che S vale uno quando rc = a/2, cioè il raggio di curvatura è pari a mezza carreggiata.
* Quando S supera l’uno Vi girerà in senso negativo portando Arianna a ruotare sempre più su se stessa, sempre la formula mostra che con rc tendente a zero S tende a infinito.

<https://youtu.be/tLTO42dV7H8>

<https://youtu.be/ULWIDnszzB4>

Il codice che realizza questa funzione è riportato sotto, S è limitato tra +/- 1.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | /\* sterzo con differenziale  raggiorSterzo indica lo scorrimento che applichiamo alle ruote  VA = motorSpeed\*(1-s)  VB = motorSpeed\*(1+s)    S\_NEUTRO è il valore per andare diritto  si ottiene per taratura (prova)  \*/  void differenziale(float motorSpeed){  static float rs;       if (direzione)    rs = raggiorSterzo + S\_NEUTRO\_FWD;     else            rs = raggiorSterzo - S\_NEUTRO\_REV;       if (rs &gt;  1.0) rs =  1.0;     if (rs &lt; -1.0) rs = -1.0;       VA = motorSpeed\*(1.0+rs);     VB = motorSpeed\*(1.0-rs);      if (VA &gt; 255) VA = 255;    if (VA &lt;   0) VA =   0;      if (VB &gt; 255) VB = 255;    if (VB &lt;   0) VB =   0;      if (direzione){       driver.motorAReverse(VA);       driver.motorBReverse(VB);    }    else{       driver.motorAForward(VA);       driver.motorBForward(VB);    }  } |

Abbiamo così lo strumento per fare sterzare Arianna.  
La misura della distanza dal bordo l’abbiamo visto in [arianna-e-i-suoi-fratelli-sperimentare-soluzioni/](https://stuffcube.wordpress.com/2016/12/08/arianna-e-i-suoi-fratelli-sperimentare-soluzioni/)

# 5 il primo anello di controllo. Un sistema per correre parallelo alla libreria

il sistema di pilotaggio di Arianna dei motori delle ruote ci permette di fare curve più o meno larghe e abbiamo visto che una curva a raggio infinito equivale ad andare diritto. Perché correre diritto e paralleli alla libreria dovrebbe essere un problema? Approfondiamo la teoria e vediamone che la soluzione sta nell’introdurre un sistema di controllo retroazionato. Vedremo l’elemento di base dei sistemi di controllo che sono onnipresenti intorno a noi, dalla caldaia di casa ai jet militari a reazione che non possono volare senza questi sistemi attivi.

**Per percorrere una retta le due ruote devono percorre lo stesso spazio.**  
Ci sono tre effetti che fanno si che il risultato sia vicino a quello atteso ma non abbastanza per garantire il risultato. Noi governiamo la velocità di rotazione di ogni singolo motore e quindi della ruota.  
**Supponiamo che le velocità siano uguali,** devono essere identiche anche le ruote. Identiche significa perfettamente uguali anche nell’appoggio al terreno. Se la loro circonferenza differisse di un millimetro, che significa una differenza tra i diametri diverso di 3 decimi di mm, avremmo che per ogni giro ruota perderemmo un mm, nel nostro caso una differenza di circa lo 0.7 %. Un piccolo errore che sommato nel tempo, integrato (dalla funzione matematica di integrale) è il termine corretto, conduce a delle differente importanti.  
**Successivamente abbiamo lo slittamento delle ruote sul terreno**, per piccola che sia Arianna, soffre anche lei di perdite di aderenza che fanno si che a un giro ruota non corrisponda esattamente un pari avanzamento sul pavimento.  
**Per ultimo l’effetto più importante,** la differenza tra la velocità richiesta e quella ottenuta. Noi stiamo usando un motore in continua con rotore alimentato a spazzole e statore a magneti permanenti.  Il rotore costituisce il circuito di armatura.

|  |  |
| --- | --- |
| [schemamotoredc](https://stuffcube.files.wordpress.com/2017/02/schemamotoredc.gif) | [eqmotoredc](https://stuffcube.files.wordpress.com/2017/02/eqmotoredc.png) |

* Va è la tensione di armatura. Nel nostro caso la tensione generata dal PWM.
* Ia è la corrente di armatura.
* Ra è la resistenza di armatura
* La è l’induttanza di armatura
* Ke e Kt son due costanti, proprie del motore, dette costante elettrica e di coppia
* Cm la coppia motrice
* ω è la velocità di rotazione
* Eg è la tensione controelettromotrice generata dalla rotazione del motore. Questa dipende dalla velocità di rotazione.

A questa aggiungiamo l’equazione della meccanica che lega momento di inerzia, ‘J’, a Coppia motrice e Coppia resistente.

J\*dω/dt = Cm – Cr

**Il tutto per dire che la velocità del motore, ω, dipende da una serie di fattori. Il considerarla proporzionale alla tensione di alimentazione è una prima approssimazione.**

E quindi??

Quindi se vogliamo governare il moto di Arianna non ci basta imporre le scorrimento tra le velocità dei motori, dobbiamo fare anche altro.

## Come guidiamo la macchina?

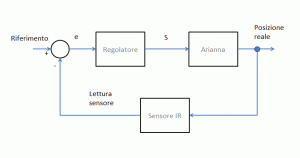
Quando guidiamo la macchina non lo facciamo ad occhi chiusi, sappiamo anzi che distogliere gli occhi dalla strada ci porta fuori via e dobbiamo correggere prontamente. Se scomponiamo le nostre azioni fatte inconsciamente queste sono nell’ordine:

* valutare nostra posizione rispetto ai bordi della strada,
* ruotare di conseguenza il volante. Questa azione modifica la posizione dell’auto.
* ritornare all’azione di valutazione. Punto 1

Il ripetere questo processo ci porta a seguire il percorso voluto. Questo algoritmo realizza un **sistema di controllo ad anello chiuso**.

## Controllo in anello chiuso

Vediamolo con uno schema a blocchi, una rappresentazione grafica dove le parti vengono scomposte e vengono indicate i collegamenti.  
Nel nostro caso il robot Arianna viene rappresentato come un elemento, blocco, che riceve un valore di scorrimento “S” e in conseguenza a questo si porta a una certa distanza dalla libreria, la posizione reale. All’interno del blocco ci sarà una funzione matematica che rappresenta la fisica del sistema. Al momento non ci interessa adesso sapere come si ricava la funzione.  
Vediamo invece come questa distanza fisica entri nel blocco “sensore IR” e da questo esca il numero “lettura sensore”. Questo è il blocco che rappresenta la misura fatta dal sensore IR. La misura viene condotta al nodo di differenza dove viene sottratto al valore “Riferimento”.  Questo nodo o blocco si chiama di confronto. È un blocco fondamentale in un sistema di controllo, qui viene fatta la differenza tra il valore desiderato “Riferimento” e dove è realmente il “sistema”. La sua uscita è l’errore “e” che viene portato al “regolatore” altro blocco fondamentale del sistema. Il regolatore in funzione dell’errore produce l’azione di sterzata su Arianna. Siamo così tornati all’inizio dove il ciclo si ripete nuovamente. Nel nostro caso ogni 100 ms ripetiamo tutta l’operazione.

[](https://stuffcube.files.wordpress.com/2017/02/sistemacontrollo.gif)

Il risultato di questo ripetersi è che quando Arianna si allontana o avvicina alla libreria viene sterzata in maniera opportuna per mantenersi alla distanza voluta.

Questo modo di operare si dice regolazione ad anello chiuso o a retroazione, il significato di chiuso sta nel blocco di ritorno “sensore IR”. Questo sensore misura la realtà e permette al sistema di reagire a ciò che accade realmente.

Abbiamo introdotto il concetto di sistema di controllo ad anello chiuso perché avevamo visto che governando solo i valori dei PWM non sapevamo esattamente dove saremmo andati. Quel metodo di controllo si chiama in anello aperto.

Introducendo la misura della distanza dalla libreria ci siamo dati un occhio (il sensore) con il blocco comparazione e il blocco regolatore siamo passati alla guida in funzione della distanza letta (sistema in anello chiuso)

## il codice del controllo

Tutto questi concetti si traducono in poche righe di codice:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | // parte a 100 ms  if ((millis()-lastTime) &gt; 100){    lastTime = millis();    noInterrupts();  measureAvailable = 0;  errore = 0.5 - misuraSideIR(0); // nodo di confronto  interrupts();    // controllo sterzo da sensore laterale  if (statoRun == 1){   raggiorSterzo =   kp\*errore;        // regolatore proporzionale     if (raggiorSterzo < -MAX\_S) raggiorSterzo = -MAX\_S; if (raggiorSterzo >  MAX\_S) raggiorSterzo =  MAX\_S;   }     // raggiorSterzo va al blocco differenziale che gestisce i due motori   ..... |

ogni 100 ms il regolatore viene eseguito.  
\*per prima c’è una parte relativa alla gestione del sensore IR. Vengono bloccati gli interrupt per poter accedere ai dati in maniera coerente. Il flag measureAvailable viene azzerato per avviare la prossima misura. Sono dettagli che possiamo ora tarscurare.

\*la riga sotto è il nodo di confronto. Il nostro sensore restituisce, nel suo range di misura, un numero reale  0 e 1. Mettiamo il riferimento al valore fisso di 0.5 in maniera da poter rimanere all’interno della sua capacità di misura.

|  |  |
| --- | --- |
| 1 | errore = 0.5 - misuraSideIR(0); // nodo di confronto |

* il regolatore: quando Arianna è in movimento, statoRun == 1, viene eseguito la regolazione. L’errore viene moltiplicato per un numero kp detto guadagno proporzionale. Questo forma un regolatore Proporzionale. Il valore del guadagno verrà trovato in maniera empirica, con il sistema attuale numeri tra 4 e12 danno risultati disgnitosi.  
  L’uscita del regolatore viene limitata con un valore MAX\_S, questo evita che vengano applicate correzioni inutili (è inproduttivo sterzare a 90° se devo andare circa diritto, così come in autostrada facciamo correzioni dolci ). Il guadagno e il limite della correzzione possono essere cambiati in funzione della velocità e del tipo di movimento che facciamo adattandoli alle condizioni.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | // controllo sterzo da sensore laterale  if (statoRun == 1){   raggiorSterzo =   kp\*errore;        // regolatore proporzionale     if (raggiorSterzo < -MAX\_S) raggiorSterzo = -MAX\_S; if (raggiorSterzo >  MAX\_S) raggiorSterzo =  MAX\_S;   } |

# 6 gli encoder

Arianna parafrasando Totò e Peppino si chiede “per andare dove devo andare, quanto devo andare?”, a questa domanda gli encoder danno la risposta.  
Con encoder, nel caso di trasduttori di posizione, intendiamo dei dispositivi in grado di indicarci lo spostamento rotatorio di un albero o lineare di un carrello. Questo valore può essere assoluto o incrementale. I trasduttori possono essere di vari tipi: ai link [*https://en.wikipedia.org/wiki/Linear\_encoder*](https://en.wikipedia.org/wiki/Linear_encoder) e [*https://en.wikipedia.org/wiki/Rotary\_encoder*](https://en.wikipedia.org/wiki/Rotary_encoder), sono rappresentati esempi di encoder realizzati con diverse tecnologie.

Gli encoder sono composti di diverse parti: meccanica, elettronica, conteggio etc. In funzione dell’applicazione vengono acquistati già completi (es nell’automazione industriale) oppure costruiti con parti separate, per esempio in una stampante. Arianna ne ha uno realizzato a d hoc che ci permette di vedere le varie parti che lo costituiscono.

## Il trasduttore meccanico

[](https://stuffcube.files.wordpress.com/2017/03/20170306_113814.jpg)

Nel caso di Arianna usiamo un trasduttore incrementale rotatorio, detto anche angolare. Questo è un disco con dei fori lungo la periferia posto sull’asse della ruota, un sensore opportuno rileva il passaggio di ogni fessura, il numero di fessure sul disco è detto ppr cioè pulses per revolution. Quando contiamo ppr impulsi abbiamo fatto un giro. È possibile avere ppr che variano da 1 a decine di migliaia. Nel nostro caso ne abbiamo 20, con una ruota intorno ai 35 mm di diametro un impulso equivale a circa 5.25 mm.  
Gli encoder possono anche indicare il senso di rotazione, nel nostro caso non abbiamo questa informazione.

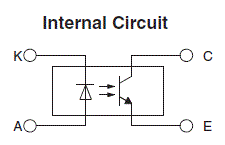
## Sensore elettrico

La rilevazione degli impulsi encoder viene fatta con sensori a infrarossi, questi sono composti da un emettitore e un ricevitore. L’interruzione del fascio luminoso viene rilevata dal ricevitore e costituisce l’informazione. I modelli a forcella (fork)  comprendono Tx e Rx meccanicamente allineati rendendoli facili da usare. Un modello usato è lo Omron ee\_sx4070

[](https://stuffcube.files.wordpress.com/2017/03/20170306_114329.jpg)

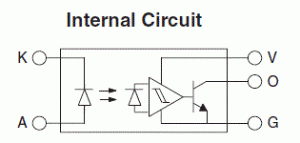
Tx Rx a infrarossi per la lettura degli impulsi

Il Tx è composto da un diodo led a infrarossi, si decide la corrente di lavoro ponendo una resistenza in serie. Il ricevitore è un fototransistor, un transistor la cui base è alimentata dalla luce.  
Per il fototransistor la radiazione luminosa ha lo stesso effetto di iniettare corrente in base. Facendola semplice lo fa commutare dandoci un livello alto o basso in uscita a seconda che il percorso Tx Rx sia libero o meno. Avendo un fototransistor va polarizzato per poter funzionare, può essere messo a collettore o emettitore comune e le correnti vanno viste dal data sheet.

[](https://stuffcube.files.wordpress.com/2017/03/omronfototransistor.gif)

fototransistor

Si consiglia di usare sensori con già uscita digitale. Si riconoscono dall’avere 5 terminali anziché quattro. Al loro interno hanno già la parte elettronica che squadra il segnale e gli applica una isteresi per evitare falsi conteggi. Per isteresi si intende che la tensione per dare un livello alto in uscita deve superare un valore Vh, per dare un livello basso non è sufficiente che scenda sotto Vh ma bensi sotto Vl, che è minore di Vh. Questa differenza di livelli, isteresi, fa si che in caso di segnali incerti nel commutare non si abbiano false commutazioni. Il simbolo nel triangolo è rappresenta l’isteresi.

[](https://stuffcube.files.wordpress.com/2017/03/schemaomron.gif)

uscita digitale

## Conteggio hw o sw debounce

Il segnale in uscita del foto sensore va ora collegato a un ingresso digitale e contato. Questa operazione può essere fatta via software o tramite un circuito hardware. In ambedue i casi lo scopo è contare il numero di impulsi che sono arrivati.  
Nel caso software possiamo mettere nel loop una lettura dell’ingresso, questo metodo è detto polling.  Ovviamente non possiamo rimanere fermi ad attendere il fronte, faremmo solo quello. Dobbiamo ad ogni ciclo verificare un eventuale cambio dell’ingresso e a fronte di un cambio incrementare un contatore. La prossima verifica avverrà al prossimo passaggio. Il sistema è adatto ove il tempo di cambio del segnale sia molto maggiore del tempo di ciclo. Nel nostro caso vogliamo avere una velocità intorno ai 0.5 m/s, con una ruota di sviluppo 0.15 m abbiamo un giro ogni 0.333 s. Il sensore da 20 impulsi a giro e quindi un impulso ogni 16 ms, poiché un impulso è un ciclo 50% alto e 50% basso abbiamo 8 ms per riconoscerlo.  
Voler utilizzare il sistema del polling costringe a scrivere un programma strozzato da questo vincolo, per esempio l’invio di dati con la seriale può causare perdita di impulsi encoder.

## L’interrupt

Questo problema si pose si dall’inizio della programmazione. Trovò una soluzione con l’interrupt o interruzione. Questo consiste nell’interrompere il flusso di istruzioni corrente per saltare a una parte di codice appunto di interrupt, al termine di questa sezione il controllo ritorna all’istruzione dove era stato interrotto. Così facendo il nostro processore sembra fare più cose insieme, in realtà ne fa sempre una per volta. Dal punto di vista logico l’interrupt ci permette di ragionare separando le cose dal punto di vista logico, alcune operazioni vengono eseguite in polling nel loop principale, quando serve un codice in interrupt gestisce immediatamente l’encoder.

dichiarazione dell’interrupt

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | /\* odometro dx     pin 21 (atMega) int 2         la funzione attachInterrupt richiede:          il numero dell'interrupt da collegare. La funzione digitalPinToInterrupt() passandogli il pin lo          trova per noi;          la funzione o callBack da chiamare quando accade l'evento;          l'evento sul quale attivarsi, questo può essere fronte di salita (RISING), discesa (FALLING),          qualunque transizione (CHANGE).      \*/  attachInterrupt(digitalPinToInterrupt(GIRO\_DX\_PIN), odometroDxMisuraHW, CHANGE);  // pin 20 (atMega) int 3  attachInterrupt(digitalPinToInterrupt(GIRO\_SX\_PIN), odometroSxMisuraHW, CHANGE); |

routine di interrupt lato Destro

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | /\*      pin 21 (atMega) int 2      sotto interrupt incremmento contatore interi      solo se R != 0  \*/  void odometroDxMisuraHW(void){  unsigned long pulseTime;          /\*            questo test verifica il tempo tra due impulsi            se sono troppo vicini significa che c'è un errore nella lettura          \*/      if ((millis() - pulseTime) < MIN\_TIME\_TRA\_PULSE) return;      pulseTime = millis();        if (statoRun == 0) return;      //Serial.println("dx");     // debug        digitalWrite(ledPin, !digitalRead(ledPin));    // segnalazione visiva      if (direzione == AVANTI)  odometroDxCnt ++;      else                      odometroDxCnt --;    } |

Il codice in interrupt deve spesso essere veloce, cioè durare poco. Evitiamo quindi ove possibile di utilizzare conti “difficili” e fatti in floating point.  Nella tabella sotto abbiamo una idea dei tempi di esecuzione con diversi formati. Vediamo che la divisione è un conto difficile, questo perché la moltiplicazione gode di una parte hardware dedicata alle moltiplicazioni 8  bit per 8 bit = 16 bit mentre la divisione no.

**Nanosecond execution times on Arduino Mega2560 (16 MHz)**  
Operation  uint8          int16          int32          int64          float  
+                  63               884           1763           8428          10943  
\*                  125            1449          4592          57038        10422  
/                 15859         15969       41866       274809      31951

<http://forum.arduino.cc/index.php?topic=196522.0>

## Traduzione in spazio

Dal numero degli impulsi alla misura fisica il passo è breve, ci basta una moltiplicazione per la costante meccanica. La costante meccanica è il numero di mm che Arianna percorre per impulso encoder arrivato. Questa traduzione in unita fisiche, i mm, viene fatta solo quando ci serve. In questo caso nel task di controllo che gira ogni 100 ms. Giunta prossima alla posizione richiesta Arianna rallenta per poi fermarsi arrivata nell’intorno della destinazione fissato.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | if ( distanza > odometro )   direzione = AVANTI;  else                        direzione = INDIETRO;  /\* gestione velocità     se sono ancora fuori dalla zona di approccio     tengo la velocita' normale  \*/      if ( abs(distanza - odometro) > E\_APPROCCIO){          motorSpeedRef = motorSpeedValue;      }      else          // altrimenti attuo la velocita' di approccio          if ( abs(distanza - odometro) > E\_POSIZIONAMENTO){              motorSpeedRef = APPROCCIO;          }          else{              // arrivato all'interno dell'errore di posizionamento              // mi fermo              motorSpeedRef = FERMO;              lastPosition  = odometro;              statoRun      = 0;          }  }// fine parte temporizzata TEMPO\_CONTROLLO ms    // faccio le variazioni di velocita' con una rampa  if ((millis()-lastTimeFast) > 10){      lastTimeFast = millis();        if (statoRun == 0) motorSpeedRef = 0;        // rampa sulla velocita'      if (motorSpeedRef > motorSpeed)  motorSpeed += 5;      if (motorSpeedRef < motorSpeed) motorSpeed -= 15; if (motorSpeed > 250) motorSpeed = 250;      if (motorSpeed <   1) motorSpeed = 0;  }// fine temporizzata veloce |

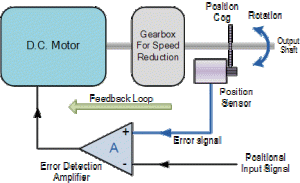
# 7 Servomotori per Pan e Tilt, come muovo la testa.

Arianna si muove sul pavimento, giace su spazio bidimensionale definito con un piano cartesiano x, y. Il libro è posto nello spazio tridimensionale x, y, z. Una volta giunti di fronte allo scaffale cercato, cioè alle coordinate x, y ci manca la dimensione z, nel nostro caso l’altezza, per indicare dove sta il libro. La fantasia può immaginare tante idee ognuna coi suoi pro e contro. Arianna usa un sistema di pan e tilt con un puntatore a diodo. A fianco di questo c’è un progetto di puntatore più simpatico “innovativo” che presenteremo in futuro.

## Pan e Tilt

Pan e tilt sono i movimenti angolari di due assi. Il secondo asse e montato sul primo e quindi si muove con lui. Nel video il progetto per il supporto di un sensore a ultrasuoni fatto da Luca. Il cad permette di verificare movimenti e eventuali inetrferenze.

per muovere i due assi vengono utilizzati i servomotori utilizzati in auto radiocomandati. I servomotori per radiocomandi contengono al loro interno un sistema che permette, variando opportunamente duty cycle applicato, di imporre la posizione dell’albero. Lo schema a blocchi ci mostra le parti costituenti:

[](https://stuffcube.files.wordpress.com/2017/03/servomotorblockdiagram.gif)

schema a blocchi del servomotore

* Il motore dc comanda il movimento meccanico
* Un riduttore a ingranaggi, gearbox, ne diminuisce la velocità di rotazione. L’albero lento è quello di uscita, output shaft, che viene usato per muovere gli oggetti.
* Su questo albero è montato un ingranaggio che si collega a un sensore di posizione. Il sensore, un piccolo potenziometro, ruota con l’albero di uscita, quindi la tensione sul cursore dipende dalla posizione.
* error signal è la differenza tra l’uscita del sensore di posizione e il riferimento, positional input signal.
* L’errore viene passato a un amplificatore capace di pilotare in maniera lineare il motore dc chiudendo l’anello. La parte sensore, nodo di confronto è l’anello di retroazione, feedback loop.

La struttura è la stessa usata per controllare che Arianna viaggiasse parallela alla libreria. La troveremo usata per altre funzioni e che impariamo a conoscere con diversi usi ma sempre lo stesso principio. **Azione, misura dell’effetto, comparazione con il valore richiesto, errore e nuova azione.**  
La differenza tra i due controlli è che Arianna per correre parallela realizzava questo controllo nel software, in questo caso è realizzato con componenti analogici. Il primo è detto **controllo digitale,** il secondo **controllo analogico.**

## il software per comandare i servo

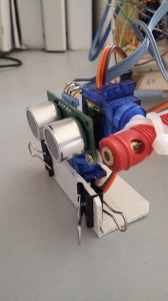
Per comandare i due servo usiamo le librerie presenti su Arduino. È sufficiente dichiarare l’oggetto servo e usare il metodo write per scrivere l’angolo desiderato.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // dichiarazione degli oggetti  Servo servoPan; // create servo object to control a servo  Servo servoTilt; // create servo object to control a servo      // loop vengono scritti gli angoli ai due servomotori  servoPan.write ( panAngle);  servoTilt.write(tiltAngle); |

## La meccanica

[](https://stuffcube.files.wordpress.com/2017/03/pantilt_01.jpg)

montaggio dei servo

[](https://stuffcube.files.wordpress.com/2017/03/pantilt_00.jpg)

montaggio dei servo

la foto mostra come montare i servo uno sull’altro usando angolari, lamiera e mollette. Il pan e tilt serve a muovere un sensore di distanza a ultrasuoni e il diodo è usato per puntare il libro.

## Arianna indica il libro

<https://youtu.be/BjnJZCtSfhg>

La sequenza è dunque questa: Arianna arriva all’altezza dello scaffale che contiene il libro, ruota, si mette perpendicolare alla libreria e si allontana un poco.  
Una volta ferma, tramite il sensore a ultrasuoni, misura la distanza dalla libreria. Conoscendo l’altezza dello scaffale da puntare calcola l’angolo di inclinazione del tilt, lo orienta e accende il puntatore.  
Il servo del pan serve per correggere disallineamenti rispetto alla libreria e per muovere il puntatore per indicare la zona dove si trova il libro.  
Il video mostra con ottimismo il libro esattamente puntato, nella realtà, si indica la zona dove sta.

# 8 Odometria

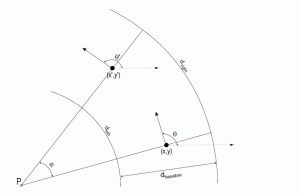
Nella prima puntata ci siamo detti che dobbiamo vedere col il punto di vista di Arianna,  l’odometria è una tecnica che usa sensori per stimare dove ci si trova nello spazio. Se pensiamo alla nostra esperienza di vita è la vista il senso che ci fornisce il grosso delle informazioni per stabilire dove siamo. Stabilire dove siamo significa posizionarci nello spazio in relazione ad altri oggetti, quindi in maniera relativa. Messo in un spazio senza oggetti mi sento perso!La visione ci permette di vedere a grandi distanze, ci risulta difficile provare questa senza sensazione a meno di entrare in camere appositamente realizzate in maniera da annullare la percezione di forme e ombre, Villa Panza a Varese ne ha ospitato sino a qualche anno fa un esempio [*http://www.aisthesis-fai.it/Ganzfeld.htm*](http://www.aisthesis-fai.it/Ganzfeld.htm) .Possiamo provare una situazione di perdita di posizione se chiudiamo gli occhi, ci mettiamo al centro di una stanza senza toccare nulla e in un luogo senza sorgenti sonore, oppure all’aperto su un sentiero in un parco. Dopo qualche tempo con gli occhi chiusi iniziamo a perdere i riferimenti.  Se ci muoviamo il nostro cervello inizia a stimare il nostro movimento e noi costantemente immaginiamo la nostra posizione salvo poi, aprendo gli occhi trovarci da tutt’altra parte. Quello che fa il nostro cervello è fondere tutte le informazioni provenienti dai nostri sensi e ricostruire il nostro movimento, mancando la vista che gli permette una verifica continua della posizione, deve aspettare di trovare un punto di riferimento e sperare di riconoscerlo.

L’odometria in base ai sensori di movimento posti sulle ruote, gli encoder, tenta di ricostruire la posizione di Arianna da quando inizia il suo percorso.

Usiamo come riferimento per questo post l’introduzione che troviamo all’indirizzo: [odomtutorial.pdf](http://web.mit.edu/6.186/2007/tutorials/odomtutorial/odomtutorial.pdf).

## La posizione di Arianna

La posizione di Arianna è rappresentata in un piano cartesiano dalle sue coordinate x e y. Questo piano è il pavimento su cui si muove. L’origine è il punto da cui parte per mostrarci il libro, la x è allineata con la libreria, la y rappresenta la distanza da questa. Ci accorgiamo però che queste coordinate non bastano, manca dove è orientata, dove sta guardando. Questa terza variabile è teta, l’angolo che forma l’asse longitudinale del robot, centrato tra le ruote che va dal dietro al davanti, e l’asse delle x. Questo angolo vale quindi 0 se Arianna si muove lungo la libreria allontanandosi dalla base, vale 180 se torna verso la base.

[](https://stuffcube.files.wordpress.com/2017/04/odometria_1.gif)

Geometria. Su un periodo sufficientemente piccolo il moto del robot si approssima a un arco. Il problema della odometria è calcolare (x’, y’, teta’) a partire a (x, y, teta). Nella figura il robot si muove in senso antiorario. d\_baseline è la distanza tra le due ruote (carreggiata).

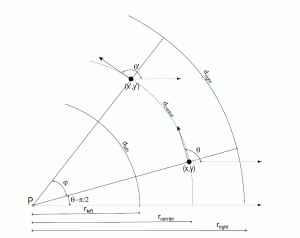
La figura illustra il movimento delle due ruote del robot, d\_left la distanza percorsa dalla ruota sinistra e d\_right dalla destra. L’arco di cerchio percorso ha centro nel punto P. Essendo un moto circolare con centro P in ogni punto  sul raggio r la distanza percorsa d vale:

d1 = φ\*r1

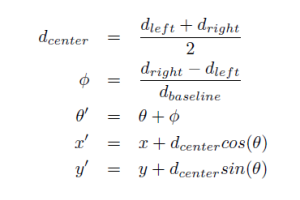
Dagli encoder, ammesso di non avere slittamento, abbiamo la misura della distanza percorsa dalle singole ruote. Ci serve una formula che a partire dalla posizione attuale ci dia la prossima posizione.  
Ripetendo il calcolo continuamente potremo stimare dove siamo.  
Arianna calcolerà questo nel loop di controlloche  viene eseguito ogni 100 ms, all’interno di questo periodo possiamo supporre che nel velocità dei due motori siano costanti anche se diverse. Vale quindi il disegno della figura 1. Da evidenziare che viaggiare in linea retta significa avere le due velocità uguali.

## La formula

Lo sviluppo della formula è riportato nell’introduzione indicata sopra. Concentriamoci sull’algoritmo che ne deriva.

[](https://stuffcube.files.wordpress.com/2017/04/odometria_2.gif)

Con riferimento alla figura sopra le formule da applicare a ogni giro sono le seguenti:

[](https://stuffcube.files.wordpress.com/2017/04/odometria_4.png)

Gli ingressi sono dati dalle coordinate attuali (x, y, θ) e dallo spostamento delle due ruote d\_left e d\_right. La d\_baseline è una costante meccanica, la carreggiata. Applicando le formule ne deriva il nuovo set di coordinate (x’, y’, θ’) che diventano a loro volta le nuove coordinate attuali. Il ciclo va dunque a ripetersi.

## Codice

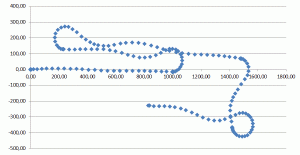
Il codice raggruppa quanto visto nei post precedenti. Un sensore encoder per ogni ruota e quindi due routine di interrupt che agiscono su due contatori diversi.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107 | #define GIRO\_RUOTA          2.625   // mm per impulso\*0.5 = sviluppo ruota[mm]/ppr (pulsi per rivoluzione)  #define GIRO\_RUOTA\_DX       2.625  #define GIRO\_RUOTA\_SX       2.615          // odometro      // pin 21 (atMega) int 2      attachInterrupt(digitalPinToInterrupt(GIRO\_DX\_PIN), odometroDxMisuraHW, CHANGE);      // pin 20 (atMega) int 3      attachInterrupt(digitalPinToInterrupt(GIRO\_SX\_PIN), odometroSxMisuraHW, CHANGE);      /\*      pin 21 (atMega) int 2      sotto interrupt incremmento contatore interi      solo se R != 0. cioè se in movimento  \*/  void odometroDxMisuraHW(void){  unsigned long pulseTime;        // se gli impulsi sono troppo vicini è dovuto a un disturbo      if ((millis() - pulseTime) < MIN\_TIME\_TRA\_PULSE) return;      pulseTime = millis();        if (statoRun == 0) return;        if (direzione == AVANTI)  odometroDxCnt ++;      else                      odometroDxCnt --;    }    /\*      pin 20 (atMega) int 3      sotto interrupt incremmento contatore interi      solo se R != 0. cioè se in movimento  \*/  void odometroSxMisuraHW(void){  unsigned long pulseTime;        // se gli impulsi sono troppo vicini è dovuto a un disturbo      if ((millis() - pulseTime) < MIN\_TIME\_TRA\_PULSE) return;      pulseTime = millis();        if (statoRun == 0) return;        if (direzione == AVANTI)  odometroSxCnt ++;      else                      odometroSxCnt --;    }          void updatePosition(void){    static long SxCnt\_k\_1 = 0;      // valore cnt a k-1  static long DxCnt\_k\_1 = 0;      // valore cnt a k-1  static long dDxCnt, dSxCnt;     // delta cnt  static long letturaDx;          // congelo cnt  static long letturaSx;          // congelo cnt  static float deltaC;            // delta cnt        // valore complessivo: usato temporaneamente      odometro = (odometroDxCnt + odometroSxCnt)\*GIRO\_RUOTA;        // calcolo evoluzione nel periodo        // blocco gli interrupt per fare le letture coerenti      noInterrupts();          letturaDx= odometroDxCnt;          letturaSx= odometroSxCnt;      interrupts();        dDxCnt   = letturaDx - DxCnt\_k\_1;               // delta sx e dx in count      dSxCnt   = letturaSx - SxCnt\_k\_1;        deltaC   = (dDxCnt + dSxCnt)\*GIRO\_RUOTA;        // avanzamento del centro nel periodo in mm        DxCnt\_k\_1= letturaDx;                           // memoria per prossimo ciclo      SxCnt\_k\_1= letturaSx;        // integro teta      teta    += ((float)dDxCnt\*GIRO\_RUOTA\_DX - (float)dSxCnt\*GIRO\_RUOTA\_SX)\*2.0/BASELINE;        // integro posizioni      xpos    +=  deltaC\*cos(teta);      ypos    +=  deltaC\*sin(teta);        // monitor dati      if (monitorDati){          Serial1.print(dDxCnt);          Serial1.print(", ");          Serial1.print(dSxCnt);          Serial1.print(", ");          Serial1.print(deltaC);          Serial1.print(", ");          Serial1.print(teta);          Serial1.print(", ");          Serial1.print(xpos);          Serial1.print(", ");          Serial1.print(ypos);          Serial1.print(", ");          Serial1.print(errore);          Serial1.println();      }    } |

Allo scattare del tempo viene chiamata la updatePosition. Qui viene eseguito l’algoritmo descritto, per risparmiare tempo si usano gli interi per i contatori sotto interrupt, le trasformazioni in float vengono fatte solo quando necessario per i conti difficili. Le costanti meccaniche per ogni impulso sono definite separatamente  per ogni ruota.  Durante la fase di taratura una differenza di 1 centesimo di mm per ogni impulso rendeva il sistema più preciso. Il valore equivale a 4 decimi di mm di differenza sullo sviluppo tra le due ruote.

## Risultato e Problemi

Grazie all’odometria è stato aggiunto un nuovo meccanismo di guida, è possibile definire l’angolo cioè la direzione del movimento. Il regolatore confronta l’angolo desiderato con il valore stimato dall’odometria. L’errore influisce sullo scorrimento tra le due ruote.  
Il meccanismo funziona! Sotto il grafico dei valori x e y restituiti da Arianna con il monitorDati attivo.

[](https://stuffcube.files.wordpress.com/2017/04/odometria_3.gif)

Il grafico x, y mostra i punti stimati in un viaggio con varie curve. Il tragitto partiva da 0, 0 con direzione 0°, poi 180°, di nuovo 0°, -90° e per finire 180°. Le distanze da percorrere erano diverse nei vari tratti. È curioso come per passare da -90 a 180° Arianna abbia fatto un intero giro su se stessa, in realtà ha fatto quanto richiesto, infatti per curvare ancora verso destra avremmo dovuto dirgli di andare a -180°!! Per capirlo occorre ragionare un poco sugli angoli!

Una nota sulla posizione y, come si vede durante le curve si sposta dallo zero. anche qui nessuno stupore, quella che viene controllata è la direzione non le posizioni x e y.

Gli ondeggiamenti che si vedono sono dovuti alla dinamica del sistema e del regolatore, possiamo paragonarlo a quando guidando l’auto dobbiamo correggere più del dovuto, ad esempio uscendo velocemente da una curva o essendo lenti nelle correzioni.

Altra fonte di errore sono gli scivolamenti delle ruote sul terreno. La base del principio usato è la misura dello spostamento se questa è falsata il risultato è falsato. Il pavimento della biblioteca ha reso necessari degli ulteriori accorgimenti per evitare questo fenomeno.

Altro errore è la bassa risoluzione degli encoder usati, questi fanno si che la risoluzione minima dell’angolo sia circa 3.5°, valore di per se non male per l’angolo. Purtroppo questo errore riportato sulle y ad esempio da luogo a deviazioni notevoli. Possiamo dire che mentre l’angolo e quindi l’orientamento è sufficiente lo stesso non vale per le coordinate cartesiane.

Per finire rammentiamo che il meccanismo di stima della posizione per integrazione dei piccoli passi a lungo andare accumula forzatamente errori, questo rende necessari dei riallineamenti quando si passa in prossimità di punti noti. Riprendendo la metafora iniziale abbiamo che  viaggiando ad occhi chiusi dobbiamo ogni tanto tastare le pareti alla ricerca di un punto di riferimento.

# Arianna ritorno alla docking station.

Per avere l’autonomia un robot mobile deve essere in grado di tornare da solo alla stazione di ricarica. Affidarsi alla sua capacità di sapere dove si trova grazie all’odometria non è sufficiente, sia per motivi sistematici, l’odometria è un processo di integrazione e quindi portato a fare errori, sia per cause accidentali come urti e ingombri sull terreno.

È necessario ricorrere a riferimenti esterni. Il metodo usato è basato su trasmettitori e ricevitori IR del tipo utilizzato nei telecomandi del televisore.

## Il metodo usato

Il principio è basato su una base fissa che monta due diversi trasmettitori. questi sovrappongono parzialmente i loro coni di emissione. Quando il robot è nel raggio di azione dei trasmettitori si muoverà per seguire la zona dove legge ambedue i fasci.

|  |
| --- |
| Figura 1: fascio emesso dai due diversi trasmettitori IR. I mezzo la zona centrale che guida al bersaglio. |

Sul robot andiamo a porre tre ricevitori. Questo ci permette di avere una copertura angolare molto ampia.   
Una soluzione alternative è usare un solo IR e orientarlo col PAN.

Figura 2: tre ricevitori a poco meno di 90° l'uno dall'altro allargano l'angolo di copertura

|  |
| --- |
|  |

La realizzazione

### La base di trasmissione

La prima necessità era quella di limitare l’angolo di emissione del led IR. Questi hanno un ampio angolo di emissione

|  |
| --- |
|  |

E’ utile sapere che gli infrarossi usati dai led (near IR) sono diversi da quelli termici (thermal IR) <https://edavies.me.uk/2014/08/ir-myth/>. Quelli dei nostri Led sono vicini di spettro alla luce visibile si comportano in maniera simile. Questo permette di limitare il cono di diffusione con del materiale che non riflette la luce. La verifica sperimentale ha confermato la funzionalità della schermatura.

Per identificare i trasmettitori si sono utilizzati due diversi codici, uno per il led destro e uno per il sinistro. Questi vengono trasmessi alternativamente per evitare sovrapposizioni. Il codice di trasmissione usato è quello dei telecomandi dei televisori implementato su una scheda Arduino.

Due transistor vengono utilizzati per fornire la corrente necessaria ad avere la portata di oltre tre metri.

### La parte ricevente

Questa parte è montata sul robot. Per sviluppare velocemente il test si è utilizzata una scheda arduino addizionale che comunica con la scheda principale tramite la seriale 4. Meccanicamente i tre ricevitori sono montati a 90 gradi l’uno dall’altro. I ricevitori sono circondati da separatori che restringono l’angolo di ricezione, facendolo più stretto per il ricevitore centrale il quale deve seguire la pista verso la docking station. Elettricamente si è usato un unico pin di ricezione, tramite delle porte logiche si abilita un ricevitore alla volta.

La stazione trasmittente emette alternativamente il due codici: CodeSx dal led sinistro e codice CodeDX dal quello destro.  
Il ricevitore, su comando, avvia la scansione dei ricevitori. Il ciclo di lettura consiste in 20 letture sui tre sensori sinistro (sx), centrale (cnt) e destro (dx). Se il segnale letto corrisponde a un codice valido un viene incrementato il contatore relativo a ricevitore ( sx, cnt, dx) e sul codice ricevuto (CodeSx, CodeDX).

L’informazione del codice lettore e del numero di letture per il sensore sinistro, centrale, destro ha la forma seguente:

#{'CodeSX': {'sx': 8, 'ct': 0, 'dx': 0, 'cn': 0},

# 'CodeDX': {'sx': 9, 'ct': 0, 'dx': 0, 'cn': 0}}

### La procedura di ricerca della base

Quando si ritiene che il robot sia nel raggio di azione della docking station si avvia la procedura di ricerca della base (seeking). È una procedura ripetitiva composta da tre passi fondamentali: misura, scelta e azione.

1. *Misura*: lettura dati
2. *scelta*: decisione del movimento. Se arrivato vai alla fine
3. *movimento*: azione. Avanza della distanza e secondo l’angolo scelto e li si arresta.
4. torna a 1
5. fine

l’andamento è quindi simile all’avanzare di un ubriaco. Osserva i luoghi, decide dove andare, fa un piccolo spostamento, si arresta e ripete il ciclo

**la misura** è l’avvio della scansione dei sensori che ritorna la frequenza dei codici sui sensori.

**La scelta** deve decidere che movimento fare. Il robot viene comandato in angolo (direzione) e spazio da percorrere. Lo spazio è adesso fisso e pari a 100 mm. La direzione viene calcolata sommando alla direzione corrente un valore dato dalla seguente formula:

eDX = IRcode['CodeSX']['dx'] + IRcode['CodeDX']['dx']

eSX = IRcode['CodeSX']['sx'] + IRcode['CodeDX']['sx']

eCT = IRcode['CodeSX']['ct'] - IRcode['CodeDX']['ct']

# decidi

e = 4\*eSX + eCT - 4\*eDX

alfa += e

la formula fornisce un numero che rappresenta come il robot è inclinato rispetto alla base. Questo numero indica quanto, rispetto alla direzione corrente, sterzare a destra i sinistra.  
partiamo da eCT, errore sui centrali, indica se il ricevitore frontale (centrale) riceve uniformemente i due codici. Lo sbilancio dei conteggi dei codici indica se sterzare a destra o sinistra.  
Per i sensori sx e dx i conteggi si sommano andando richiedere di sterzare maggiormente. Il segno – su eDx attribuisce la direzione.  
eSx ed eDx vengono moltiplicati per 4. Questo attribuisce un peso maggiore sull’angolo da parte di questi sensori.

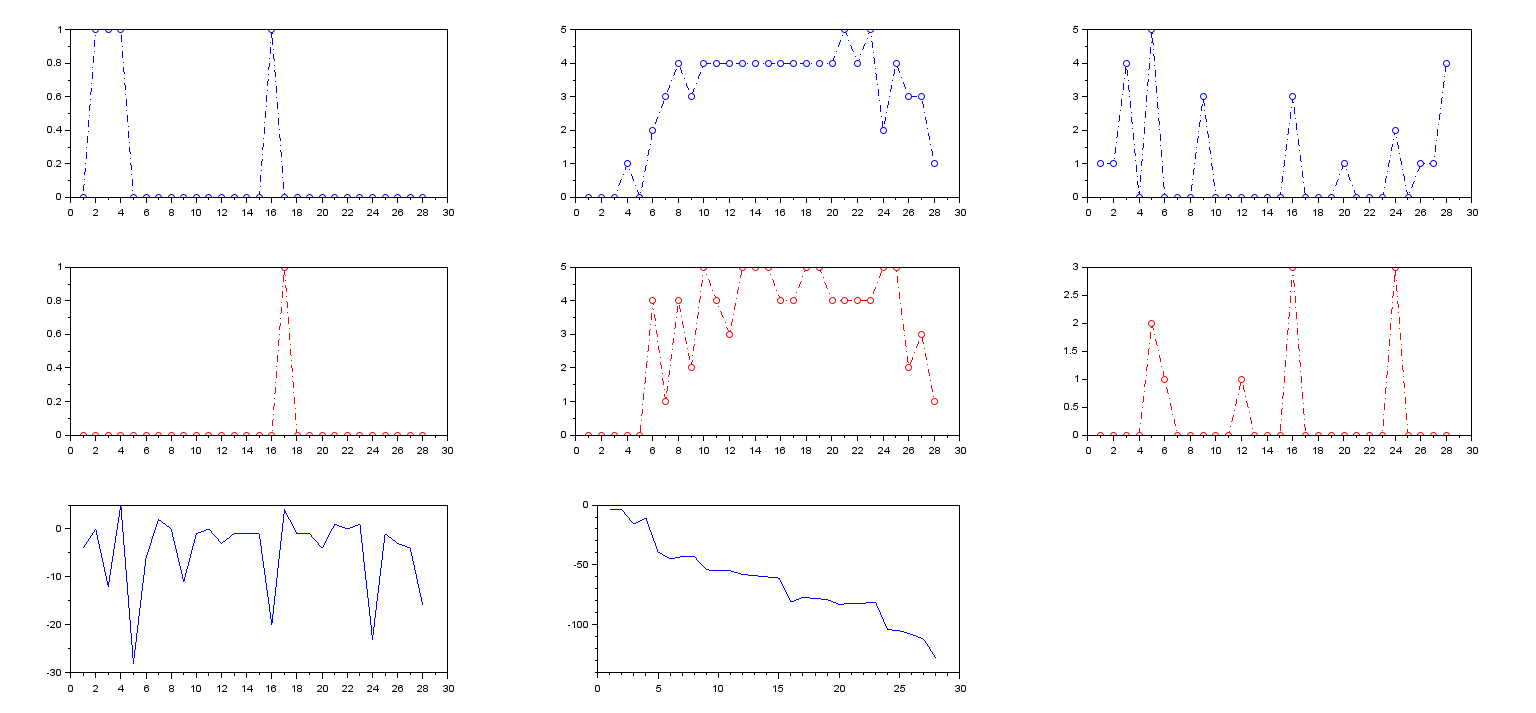


Figura 3: le due righe superiori mostrano il CodeSx e CodeDx ricevuti dal sensore sx, cnt, dx. La riga inferiore a sinistra da l'errore ‘e’ e a destra l'angolo ‘alfa’. Sull’asse x il ciclo di ricerca. Attenzione alle scale sulle y diverse.

I grafici sopra mostrano un avvicinamento. *Attenzione alle scale sulle y diverse*. Nei primi cicli non vi sono segnali sui sensori centrali e sinistro. L’angolo passa da zero a 50 gradi in pochi cicli. Iniziano ad apparire letture sui sensori centrali sino a portare il robot a destinazione. Alcuni segnali sui sensori laterali arrivano per riflessione dovuta alle pareti circostanti.

Rimane da definire come decretare che il robot è arrivato alla base. È necessario rilevare l’arrivo del robot e inviare l’informazione di robot arrivato alla procedura di ricerca. Per la prima parte è stata realizzata una semplice barriera luminosa. Un diodo laser del tipo utilizzato per i puntatori luminosi illumina una cellula fotoelettrica. Quando il robot arriva interrompe il raggio e la resistenza della fotocellula cambia. Il micro della docking station rileva questa variazione.  
Per trasferire l’informazione si è utilizzato il canale ad infrarossi. Quando il raggio è interrotto il codice trasmesso cambia e la procedura può agire di conseguenza.

## Miglioramenti

La procedura di ricerca può essere velocizzata misurando la distanza tramite il lidar dalla base e variando il passo di avanzamento del robot. Quando il sensore frontale è ben illuminato, ha cioè un numero elevato di codici ricevuti, è ragionevole ignorare i sensori laterali che possono ricevere riflessioni. La stazione di arrivo può modulare la potenza e il codice emesso, questo può permettere una valutazione ulteriore della posizione. Le stazione può dialogare col il robot tramite wifi o bluetooth per ulteriori strategie di sincronizzazione tra trasmissione e ricezione. Queste possono permettere la lettura al volo senza la necessità di fermata.

# 9 Tempo di libri, Technology Hub

StuffCube ha contribuito ad animare lo stand del consorzio delle biblioteche CSBNO alla fiera “Tempo di libri” che si svolge a Rho Fiera Milano. lo ha fatto con gli oggetti prodotti dai partecipanti, in questo caso le versioni di Arianna -il robot bibliotecario-, il braccio e mano meccanica Luka-Arm e un video su StuffCube.

Nel contempo ho visitato Technology Hub, una piccola fiera dedicata all’innovazione, che si teneva alla fiera di Milano city. I trend erano materiali compositi, la stampa 3d, la realtà aumentata e virtuale, i robot collaborativi.

Due fiere diverse, libri verso innovazione tecnologica. La domanda che mi sorge è : ”Sono veramente fiere diverse?”

[](https://stuffcube.files.wordpress.com/2017/04/2017-04-22-at-19-45-58.jpeg)

A  Technology Hub, in uno stand sulla robotica,  in un video ho visto queste parole “ **un robot lavora 24 ore al giorno, 365 giorni all’anno, non vuole aumenti di stipendio, non sciopera, non sbaglia**”.

I robot collaborativi hanno la novità di poter lavorare a fianco di un essere umano, non dovendo essere confinati per ragioni di sicurezza, e possono apprendere da lui.  
Nulla di nuovo, semplicemente la prosecuzione di quanto iniziato da Taylor all’inizio del secolo scorso. Taylor, citato spesso insieme a Ford, introdusse la scomposizione del lavoro in singole operazioni aumentando la produttività delle fabbriche.  
Oltre ad aumentare l’efficienza questo modo di lavorare apprendeva dagli artigiani la conoscenza della tecnologia di produzione. Prima della fabbrica moderna erano gli artigiani che sapevano fare le cose e se le tramandavano in bottega.  Nella fabbrica moderna con i concetti introdotti da Taylor e Ford  ogni operaio  sa fare solo la sua parte di lavoro e quindi è più produttivo e  facilmente sostituibile.

Queste informazioni si trovano scritte sui libri che vanno letti e ne vanno letti molti perché ogni episodio della vita umana ha molti aspetti. Volendo dare uno spunto di riflessione direi che la lettura della storia del inizio del ‘900 propone temi attuali anche oggi.  
Leggere i pensieri dei nostri predecessori ci permette di vedere lontano “**di vedere seduti sulle spalle di giganti”**.

Quello che voglio dire è che i libri, **conoscenza umanistica cioè dell’uomo, e tecnologia non possono vivere separatamente,** le grandi menti del Rinascimento, uno per tutti Leonardo, erano umanisti, tecnici e scienziati allo stesso tempo. Avere una tecnologia, con tutto il suo impatto sull’uomo, che si muove guidata da chi ignora cosa sia l’uomo non è bello e le conseguenze potrebbero essere spiacevoli.  
L’atto di semplificare le cose va fatto quando le cose si sono comprese a fondo, oggi un malinteso processo di specializzazione e semplificazione ci fa perdere il quadro d’insieme degli accadimenti e quindi ci porta a decisioni che danno effetti imprevisti ma non imprevedibili.

# 10 IoT + Robotics = Cloud Robotics

Sabato 13 maggio abbiamo fatto un corso su  IoT + Robotics = Cloud Robotics. Un progetto made in Torino per rendere facile la realizzazione di robot utilizzando il sistema ROS (Robot Operating System). Il lavoro è all’inizio, ambisce ad essere l’Arduino dei Robot connessi in rete.  
Il termine robot è ampio e non si limita a solo quelli che si movono, un robot può anche accendere luci o aprire porte o quello che volete.

Nelle otto ore di workshop tenutosi a WeMake di Milano abbiamo giocato con alcuni robot, questi sono basati su Raspberry PI, tramite l’iterfaccia su cloud abbiamo acceso un led, inviato comandi a i motori da un joystick da web, da una interfaccia vocale, intefacciato al Raspberry una scheda Arduino via seriale.

Abbiamo inoltre conosciuto tante persone dalle esperienze più diverse che vanno a formare una comunità di sviluppatori, utilizzatori etc etc….

una giornata intensa con prospettive tutte da scoprire.

Low cost Open Source Robots now connect to the power of Cloud Computing  
[hotBlackRobotics](http://www.hotblackrobotics.com/)