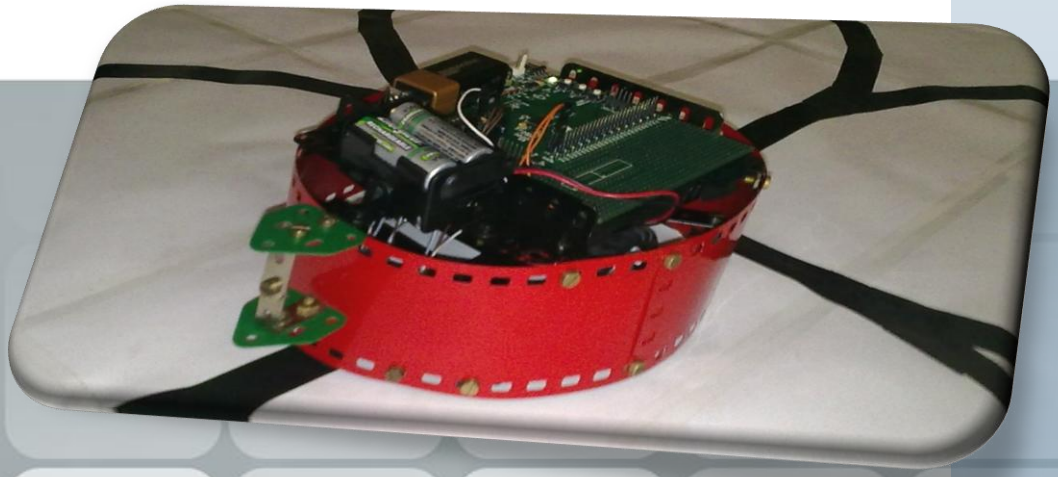
Line Follower Robot

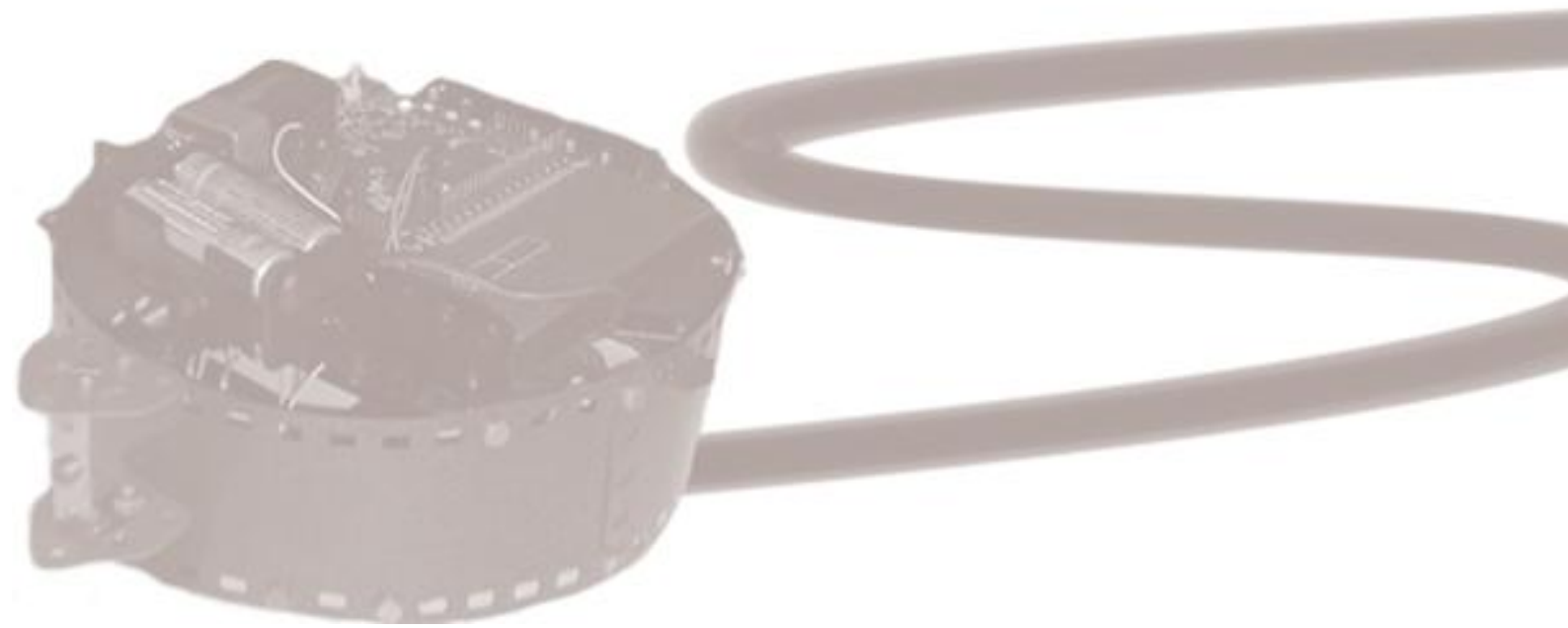


Titolo:
- Progetto di Informatica Industriale

Corso:
- Corso di Laurea Specialistica in Ingegneria Informatica

Autori:
- Giovanni Accongiagioco
- Simone Brienza
- Daniele Giannetti

SOMMARIO



1 INTRODUZIONE

L'applicazione è un classico esercizio di robotica che può essere risolto in molteplici modi; molta documentazione è presente in rete al riguardo. L'obiettivo è quello di realizzare un Line Follower Robot (LFR), ovverosia un robot in grado di seguire una linea, realizzata ad esempio con del nastro isolante nero, su una superficie bianca. Percorsi tipici scelti per questi robot possono essere semplici oppure contenere bivi e intersezioni.

Nonostante un robot di questo tipo sia puramente un "proof of concept", le tecnologie utilizzate (sensori riflessivi ad infrarossi, motori DC, ...) sono comunemente presenti nel mondo industriale e le problematiche affrontate si possono ritrovare qualora ad esempio si vogliano costruire sistemi di lavorazione automatici nelle catene di montaggio industriale.

Nel nostro caso abbiamo scelto componenti molto economici per riuscire a rientrare in un budget di circa 50€, questo riduce le configurazioni possibili per il robot: la maggior parte dei modelli che si trovano in rete lavorano con una moltitudine di sensori per sapere in ogni istante dove si trova la linea rispetto alle ruote del robot, mentre nel nostro caso il numero di sensori è limitato a 3.

Nel seguito presenteremo il funzionamento del robot descrivendo prima la struttura hardware e poi dando una breve panoramica del funzionamento del software che è stato scritto per il **microcontrollore ADuC836** della **Analog Devices**. Il documento è così strutturato:

- Sezione 2
  Struttura hardware con considerazioni sul pilotaggio dei vari componenti e sulla scelta dei piedini della evaluation board per realizzare le funzionalità desiderate.
    - 2.1 Forma dello chassis
    - 2.2 Pilotaggio dei motori DC
    - 2.3 Sensori riflessivi a infrarossi
    - 2.4 Periferiche usate nel microcontrollore e piedini della evaluation board
    - 2.5 Schema riassuntivo
- Sezione 3
  Architettura software del programma per il funzionamento del robot
    - 3.1 Scheduler
    - 3.2 Task

2 STRUTTURA HARDWARE

In questa sezione è descritta la struttura fisica dei componenti che formano il robot, con considerazioni di carattere elettronico. Occasionalmente si fa riferimento anche a informazioni prelevate dai datasheet disponibili online.

2.1 FORMA DELLO CHASSIS

Lo chassis per il LFR è stato realizzato con il Meccano (un sistema per la costruzione di modelli giocattolo molto utilizzato).

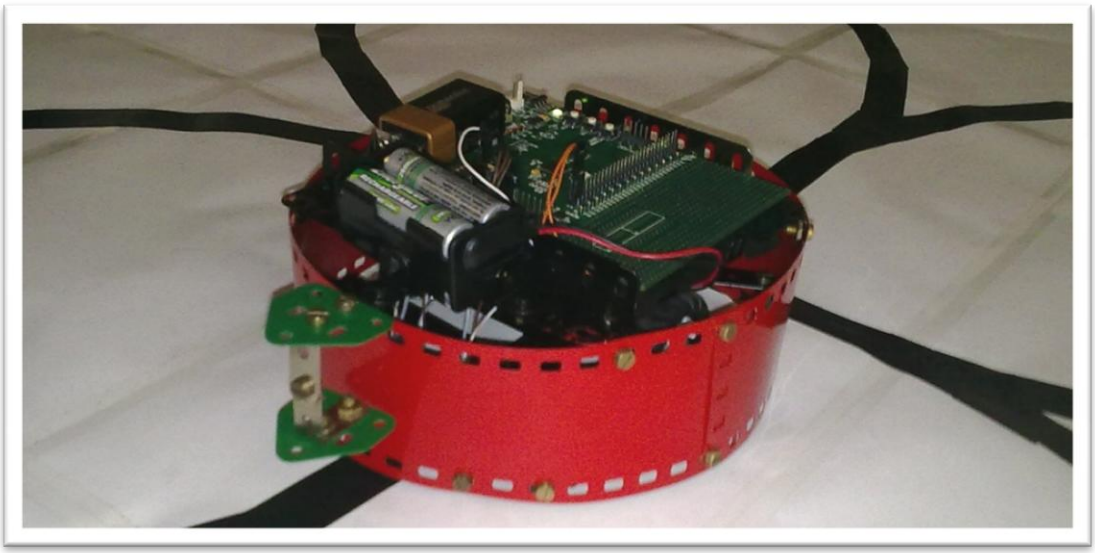


Figura 1: Line Follower Robot

Il corpo principale ha forma cilindrica; un componente addizionale funge da indicatore della direzione del robot.

All'interno del corpo cilindrico la struttura è organizzata su due livelli separati da due piani realizzati ancora una volta con il Meccano. Le due superfici fungono da appoggio per le schede elettroniche, pertanto i pezzi che le formano sono stati verniciati in modo tale che la vernice a base di alluminio (usata nel Meccano per alcuni pezzi) non creasse problemi con percorsi erronei a bassa impedenza; in aggiunta a questo le schede vengono tenute separate dai piani verniciati con del nastro isolante.
Sotto la superficie inferiore è agganciato un opportuno supporto per i tre sensori infrarossi a riflessione che il robot utilizza per individuare il percorso da seguire. Vi sono inoltre collocati i due motori, inseriti all'interno di un unico pezzo plastico con possibilità di impostare il rapporto di

riduzione. Collocata sopra al piano inferiore si trova una scheda millefori, dove è stata montata la circuiteria per il pilotaggio dei motori e dei sensori.
Sul piano superiore invece abbiamo la evaluation board con il microcontrollore ADuC836 e il pacchetto batterie (AA) per l'alimentazione di tutte le componenti esterne alla evaluation board (motori, sensori, e rispettiva circuiteria).

Il robot ha 3 punti di appoggio sul terreno: le 2 ruote direttamente connesse ai due motori e una piccola sfera metallica sotto la parte frontale dello chassis (ball caster). Le curve vengono fatte muovendo le ruote con velocità diverse; i sensori sono collocati tra il ball caster e le due ruote.

Nelle immagini seguenti il robot è mostrato da diversi punti di vista.

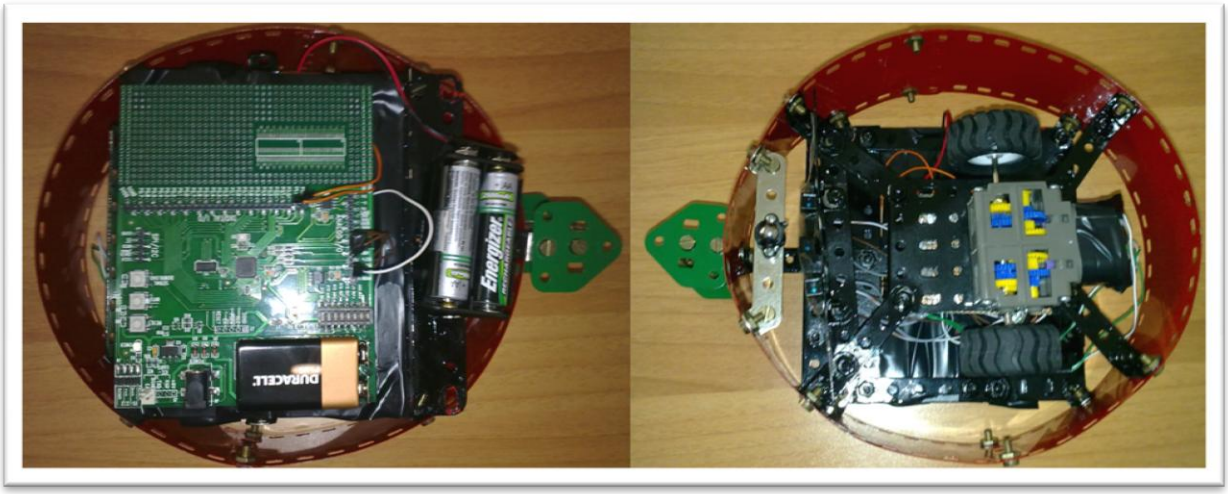


Figura 2: A sinistra vista dall'alto, a destra vista dal basso

Figura 3: Vista laterale

2.2 PILOTAGGIO DEI MOTORI DC

I due motori DC sono collocati all'interno di un pezzo plastico della Tamiya (**Tamiya 70168 double gearbox**) che nel seguito indichiamo semplicemente come double gearbox. I due motori sono pilotabili in modo completamente indipendente e i due rapporti di riduzione sono stati impostati in modo opportuno. I due motori sono motori **Mabushi FA-130RA** e possono essere alimentati con una tensione tra 3 e 6 V (DC), noi useremo alimentazione a 4.8 V corrispondente alle 4 pile ricaricabili collocate nel pacchetto batterie.

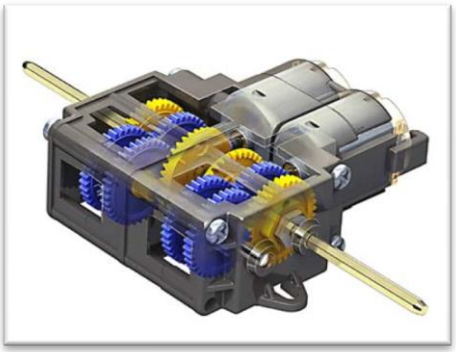


Figura 4: Double Gearbox Tamiya 70168

Ognuno dei due motori viene alimentato tramite una coppia di transistor di potenza, la quale fornisce la corrente di cui il motore ha bisogno per funzionare (impossibile da ottenere collegando direttamente il motore alla scheda del micro-controllore) quando dalla evaluation board (**AD MicroConverter ADuC8xx Sigma Delta Evaluation Board)** viene dato il segnale su un opportuno collegamento. I motori vengono pilotati in modalità PWM, con due segnali distinti, il cui duty cycle è regolabile singolarmente. Ciascuno di essi necessita al massimo (in condizioni di stallo) di qualche Ampere e dunque è necessario prevedere una struttura che possa funzionare anche sotto questi alti valori di corrente.

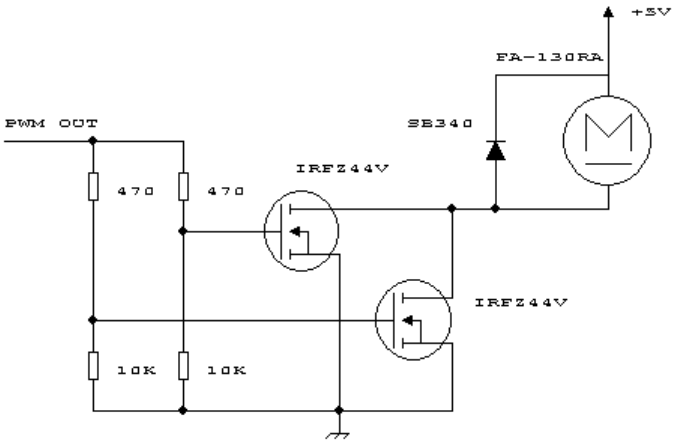


Figura 5: Driver per motore DC

Il circuito di pilotaggio è riportato a lato: come si evince, viene usata una struttura con due MOSFET in parallelo, che funge da transistor di potenza per il motore DC. Questo non sarebbe strettamente necessario nel nostro caso, considerando che la massima corrente richiesta potrebbe essere gestita anche con un singolo transistor, tuttavia qualora si volesse sostituire il motore con un modello più potente tale architettura ci garantisce la possibilità di pilotarlo correttamente anche in caso di maggiore amperaggio richiesto.

Va notata la presenza di un diodo vicino al motore. Per comprendere la sua ragion d'essere bisogna considerare che il motore si può rappresentare con un circuito equivalente dove compaiono delle componenti induttive. In condizioni normali, quando il motore è alimentato, il diodo risulta interdetto e il motore viene azionato. Quando però la coppia di transistor si interdice allora il motore non risulta più sottoposto alla differenza di potenziale della batteria e la brusca variazione di corrente genera, ai capi del motore, una differenza di potenziale indotta, ad opera dei componenti induttivi al suo interno. Tale tensione può causare danni ai transistor visto che può essere molto alta (un picco): dipende infatti dalla rapidità con cui la corrente varia nel tempo (———). Ne deriva che il diodo serve a consentire al motore di scaricarsi senza il rischio di bruciare i transistor.

I motori dovranno essere gestiti con due segnali PWM distinti, in modo tale che sia possibile impostare a piacere il duty cycle di ognuno dei due. I due segnali PWM causeranno la continua apertura e chiusura dei transistor di potenza, originando un segnale digitale ad alta frequenza diretto al motore. Poichè un motore DC è un componente meccanico (dunque lento), un segnale a frequenza sufficientemente elevata verrà di fatto interpretato come un segnale di alimentazione continuo con un valore proporzionale a quello del duty cycle del segnale PWM, fino ad un massimo valore dato dalla tensione continua di alimentazione (4.8 volt nel nostro caso).

Per la scelta della frequenza del segnale PWM bisogna fare alcune osservazioni:

- Una frequenza troppo elevata può avere un effetto indesiderato sulle componenti induttive nel motore che causano perdite di potenza e surriscaldamento.
- Una frequenza troppo bassa può causare un andamento "a scatti" del motore che ovviamente è da evitare.

Basandoci sulle frequenze PWM tipicamente utilizzate per pilotare questi piccoli motori della Tamiya, una frequenza tra 600 e 750 Hz risulta preferibile. La scelta della frequenza da utilizzare per il segnale PWM si ripercuote sul modo in cui il controllore PWM andrà impostato sul microcontrollore ADuC836.

2.3 SENSORI INFRAROSSI

Tipicamente questo tipo di robot sfrutta degli emettitori di luce per scandagliare il percorso e rileva l'intensità della luce riflessa per capire qual è il colore di alcuni punti sul percorso. Si possono usare diverse soluzioni come luce bianca o infrarossa, nel nostro caso abbiamo optato per la seconda soluzione per questioni di reperibilità dei pezzi. In particolare abbiamo utilizzato un sensore ottico riflessivo ad infrarossi della **Vishay**, modello ***CNY70***.

Tre di questi sensori sono montati in linea tra le ruote motrici ed il ball caster, ad una distanza dal suolo non superiore ai 3 millimetri. In caso di linea retta, il robot sta seguendo opportunamente il percorso quando solo il sensore centrale rileva la presenza della striscia nera.

Tali dispositivi hanno dei package di dimensione molto ridotta (inferiore al centimetro di lato) e hanno 4 piedini per il loro utilizzo, come mostrato nella figura sotto.

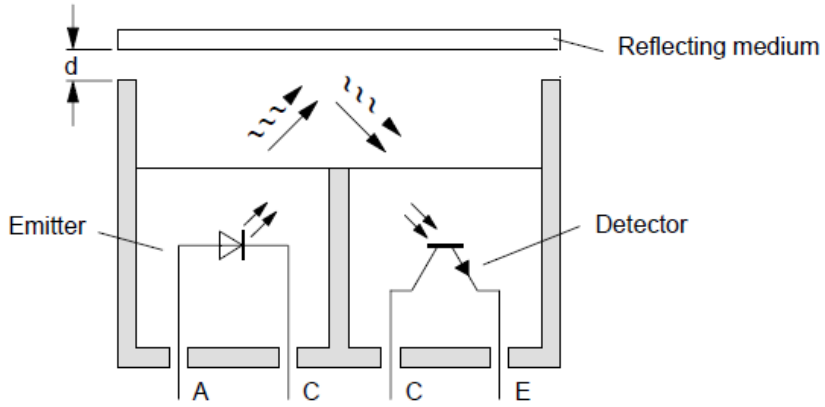


Figura 6: Sensore IR a riflessione

Da sinistra verso destra evidenziamo:

1. **A – Anodo**

   Questo va collegato alla batteria esterna in modo tale da fornire al diodo emettitore una corrente sufficiente (il valore effettivo si può regolare con una opportuna resistenza).

2. **C – Catodo**

   Collegato a massa.

3. **C – Collettore**

   Quando il transistor BJT fotorivelatore viene illuminato con luce infrarossa riflessa da un oggetto ad alto fattore di riflessione, entra in conduzione (interruttore chiuso) e il collettore viene portato alla tensione dell'emettitore.

Al collettore collegheremo pertanto una resistenza di pull-up e all'emettitore collegheremo la massa in modo tale da ricevere un 1 logico quando il transistor è perfettamente interdetto e uno 0 logico nel caso in cui il transistor sia in conduzione perché si sta ricevendo luce riflessa.

4. **E – Emettitore**

   Collegato a massa.

Lo schema di montaggio che si è usato per ognuno dei 3 sensori è allora il seguente.

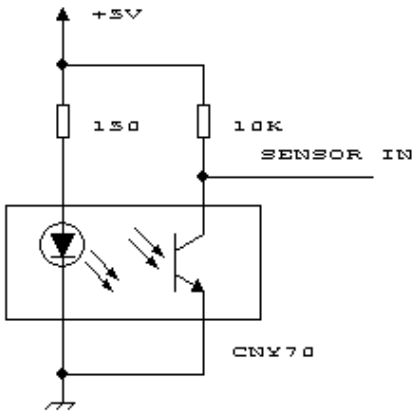


Figura 7: Montaggio di un sensore IR CNY70

La resistenza da 150 Ω per il pull-up dell'anodo del diodo garantisce una corrente di circa

---

che forza il diodo ad emettere una luce IR sufficiente ad essere rilevata dal transistor.

La resistenza da 10 kΩ per il pull-up sul collettore del fototransistor garantisce invece una corrente molto contenuta all'interno del transistor: questo comporta anche consumi di potenza molto ridotti per la parte relativa al fototransistor.

In generale, la tensione misurata sul piedino del microcontrollore sarà un valore intermedio tra 0 e 4.8 V a seconda della corrente che il transistor di rilevazione lascia scorrere (la quale influenza la caduta di potenziale sulla resistenza da 10 KΩ). In questo caso si potrebbe allora pensare che sia il caso di utilizzare uno dei convertitori A/D di cui il microcontrollore ADuC836 è equipaggiato. Tuttavia possiamo osservare che nel nostro caso quello che ci interessa è individuare la presenza di una linea fatta con nastro isolante nero (fattore di riflessione inferiore al 10%) su della carta

bianca (fattore di riflessione superiore al 90%) e dunque risulta possibile collegare i sensori direttamente a degli ingressi digitali sul microcontrollore e assumere il seguente comportamento:

- 1 logico → transistor interdetto → nero
- 0 logico → transistor in conduzione → bianco

Visto che il circuito per il pilotaggio dei sensori (visto sopra) ha una resistenza di pull-up apposita, non è necessario usarne una interna al microcontrollore e dunque si useranno dei pin di input digitale open-drain (che di base risultano flottanti).

2.4 PERIFERICHE USATE NEL MICROCONTROLLORE E PIEDINI DELLA EVALUATION BOARD

Il microcontrollore ADuC836 dispone di svariate periferiche che gli consentono di assolvere le funzioni più disparate. Il chip è montato sopra la evaluation board della Analog Devices e grazie a questa scheda alcuni ulteriori dispositivi vengono messi a disposizione del programmatore (LED, interfaccia seriale per la programmazione, pulsanti di interruzione esterna, ecc…).

Spiegato il modo in cui si pilotano i motori DC e i sensori infrarossi, bisogna adesso vedere quali dispositivi usare sul microcontrollore e in che modo.

2.4.1 CONTROLLORE PWM

Il microcontrollore ADuC836 è dotato di un controllore PWM (Pulse Width Modulator) capace di generare segnali PWM di svariato tipo. Abbiamo già detto che nel nostro caso vogliamo generare due diversi segnali PWM tra loro indipendenti (dove il duty cycle sia regolabile singolarmente) ad una frequenza tra 600 e 750 Hz.

La struttura del Pulse Width Modulator interno all'ADuC836 è la seguente:

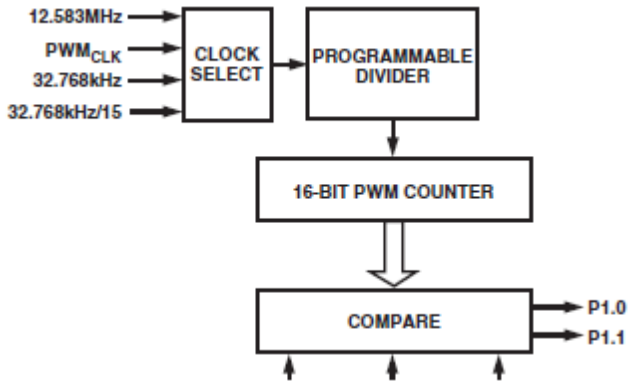


Figura 8: Struttura del PWM nell'ADuC836

Come si nota, degli opportuni bit di configurazione in particolari registri ci consentono di scegliere la frequenza di clock da utilizzare per la generazione del segnale PWM, in aggiunta è possibile impostare una divisione della frequenza nel caso in cui risultasse troppo elevata. Il clock così ricavato viene usato per incrementare un apposito contatore che in base a delle soglie configurabili scatena degli eventi sui piedini di uscita P1.0 e P1.1. Il tipo di segnale ottenuto sui due piedini dipende fortemente dalla modalità scelta per il funzionamento del PWM.

Nel nostro caso abbiamo bisogno di due segnali PWM con la stessa frequenza, i cui duty cycle siano impostabili individualmente, per cui le 3 modalità del controllore PWM che risultano accettabili considerando questo requisito sono:

- Mode 2: Twin 8-bit PWM
  Due segnali PWM vengono generati con una risoluzione sul duty cycle di al massimo 8 bit; la frequenza è la stessa e i segnali possono essere sfasati in modo arbitrario; il periodo dei due segnali è liberamente controllabile.
- Mode 3: Twin 16-bit PWM
  Due segnali PWM vengono generati con una risoluzione sul duty cycle di 16 bit; la frequenza è la stessa ma il periodo non è controllabile se non con la scelta del clock da utilizzare e con una configurazione opportuna del divisore.
- Mode 5: Dual 8-bit PWM
  Due segnali PWM vengono generati con una risoluzione sul duty cycle di al massimo 8 bit; la frequenza dei singoli segnali può essere diversa e il periodo dei segnali è liberamente controllabile.

Oltre a queste 3 vi sono anche modalità più complesse che potremmo utilizzare, non discusse in questa trattazione in quanto comunque non necessarie.

Un secondo requisito che dobbiamo rispettare è che vi deve essere la possibilità di ottenere delle frequenze del segnale PWM che siano idonee a pilotare i nostri motori. La mode 3 è stata allora scartata in quanto (con qualsiasi scelta possibile) non si riesce ad ottenere frequenze superiori a 192 Hz. Usando la mode 2 o la mode 5 si riesce invece ad ottenere frequenze nel range desiderato tra 600 e 750 Hz. Per ottenere maggiore flessibilità è stata scelta la modalità 5. In generale, in modalità 5, i due segnali PWM osservati sui piedini P1.0 e P1.1 evolvono come segue:

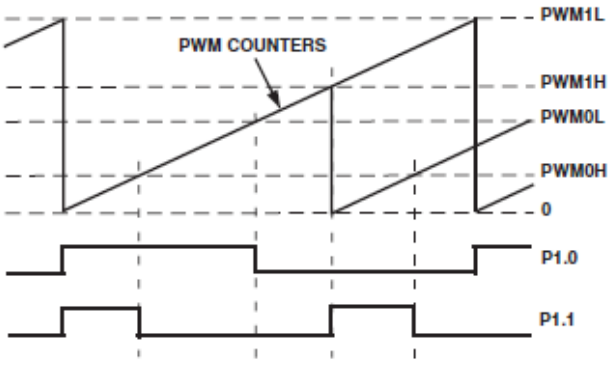


Figura 9: Segnali PWM in modalità 5

Nel nostro caso si sono configurati due segnali PWM alla stessa frequenza, pari a 655 Hz. Il duty cycle verrà configurato via software per ottenere la spinta voluta da ogni singolo motore.

2.4.2 PIN DI INPUT DIGITALE

Per ricevere correttamente i 3 segnali dai 3 sensori infrarossi a riflessione, bisogna considerare la struttura della circuiteria utilizzata per pilotare questi sensori.

Abbiamo già detto che, data la grande differenza di fattori di riflessione tra nastro adesivo nero e foglio bianco, possono essere semplicemente utilizzati degli ingressi digitali per leggere il segnale dai sensori. La circuiteria vista contiene una resistenza di pull-up ed un opportuno transistor per portare l'ingresso verso massa, e questo restringe la libertà di scelta per i pin utilizzati per l'I/O.

L'ADuC836 dispone di 4 porte per l'I/O digitale, ognuna dispone di 8 pin e ad ogni pin possono essere assegnate anche delle funzioni particolari (ad esempio segnale di interrupt proveniente dall'esterno). Volendo usare 3 di questi pin come ingressi digitali, l'unico requisito che si pone è che non abbiano una resistenza di pull-up interna per evitare che nel circuito del sensore si presentino delle correnti inattese.

I pin che meglio si configurano per questo scopo sono P1.2-P1.7, i quali sono semplici piedini, configurabili in modalità analogica oppure digitale, che possono essere utilizzati solo come ingressi. Nel caso in cui si volesse fare una analisi più fine del segnale ricevuto possono anche essere facilmente configurati per sfruttare i convertitori A/D interni al microcontrollore; pertanto risultano la scelta più ovvia.

La struttura di questi pin è la seguente:

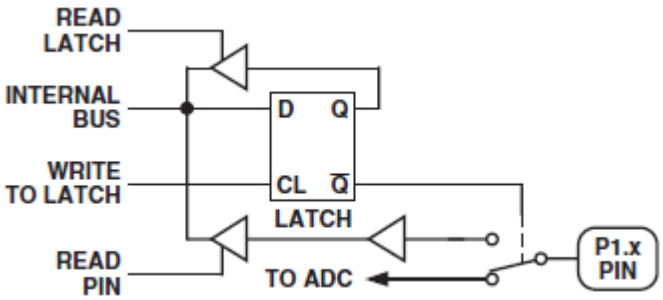


Figura 10: Pin di ingresso P1.2-P1.7

Come si vede nella figura, il pin si configura come digitale scrivendo nell'opportuno bit di registro il valore 0 in modo tale che lo switch mandi direttamente il valore del piedino desiderato sul bus interno.

Per la lettura dei nostri tre sensori sono stati scelti i pin P1.4, P1.5 e P1.6.

2.4.3 INTERFACCIA SERIALE UART

A scopo di debug, si è prevista la possibilità di utilizzare l'interfaccia seriale UART per comunicare alcuni messaggi di controllo ad un ascoltatore connesso con un PC. L'interfaccia UART ha bisogno di un timer per funzionare correttamente e nel nostro caso si è scelto di utilizzare il timer 3 che risulta particolarmente adatto per lo scopo (oltre che effettivamente inutilizzato nel resto del codice) e ci permette di generare dei baud rate molto accurati.

La configurazione e l'utilizzo dell'interfaccia UART avvengono solamente quando il codice viene compilato definendo la costante **DEBUG** che di base non viene definita. In tal caso la seriale viene configurata per funzionare ad un baud rate di 9600 bps con un singolo bit di start ed un singolo bit di stop.

2.4.4 ALTRI TIMER

Anche se il timer 3 viene dedicato alla gestione della seriale qualora si decida di utilizzare il software in modalità debug, il microcontrollore ci mette a disposizione altri 3 timer/counter.

Nel nostro caso il timer/counter 2 viene dedicato al funzionamento dello scheduler. L'utilizzo di uno scheduler ci consente di fornire dei vincoli real-time di tipo soft per i task che devono essere eseguiti nella nostra applicazione. Il suo funzionamento per lo scopo prefissato è assicurato però solo nel caso in cui non vi siano altre interruzioni da gestire nel codice.

Per ottenere delle garanzie effettive, non è concesso al programmatore l'utilizzo di interrupt esterni o interni. Qualunque operazione in questa architettura a task dovrà essere fatta in software in modalità polling.

A fronte di questa limitazione siamo però in grado di garantire dei vincoli temporali; inoltre il dispositivo entrerà automaticamente in modalità di risparmio energetico (idle mode) per essere svegliato qualora il timer/counter 2 generi una nuova interruzione ad indicare che un tick di scheduling è trascorso.

Il timer/counter 0 e il timer/counter 1 nel nostro caso non sono stati utilizzati per alcuna attività.

2.5 SCHEMA RIASSUNTIVO

Lo schema riportato di seguito riassume l'intera struttura dell'elettronica che è stata realizzata per il pilotaggio dei motori e dei sensori, con riferimento anche ai piedini della scheda (e del microcontrollore) a cui i collegamenti sono stati portati.

La batteria riportata in questa immagine simboleggia l'alimentazione esterna separata da quella usata per il microcontrollore. Come abbiamo già sottolineato, la scheda del microcontrollore infatti non è in grado di fornire l'amperaggio richiesto per il funzionamento dei motori.

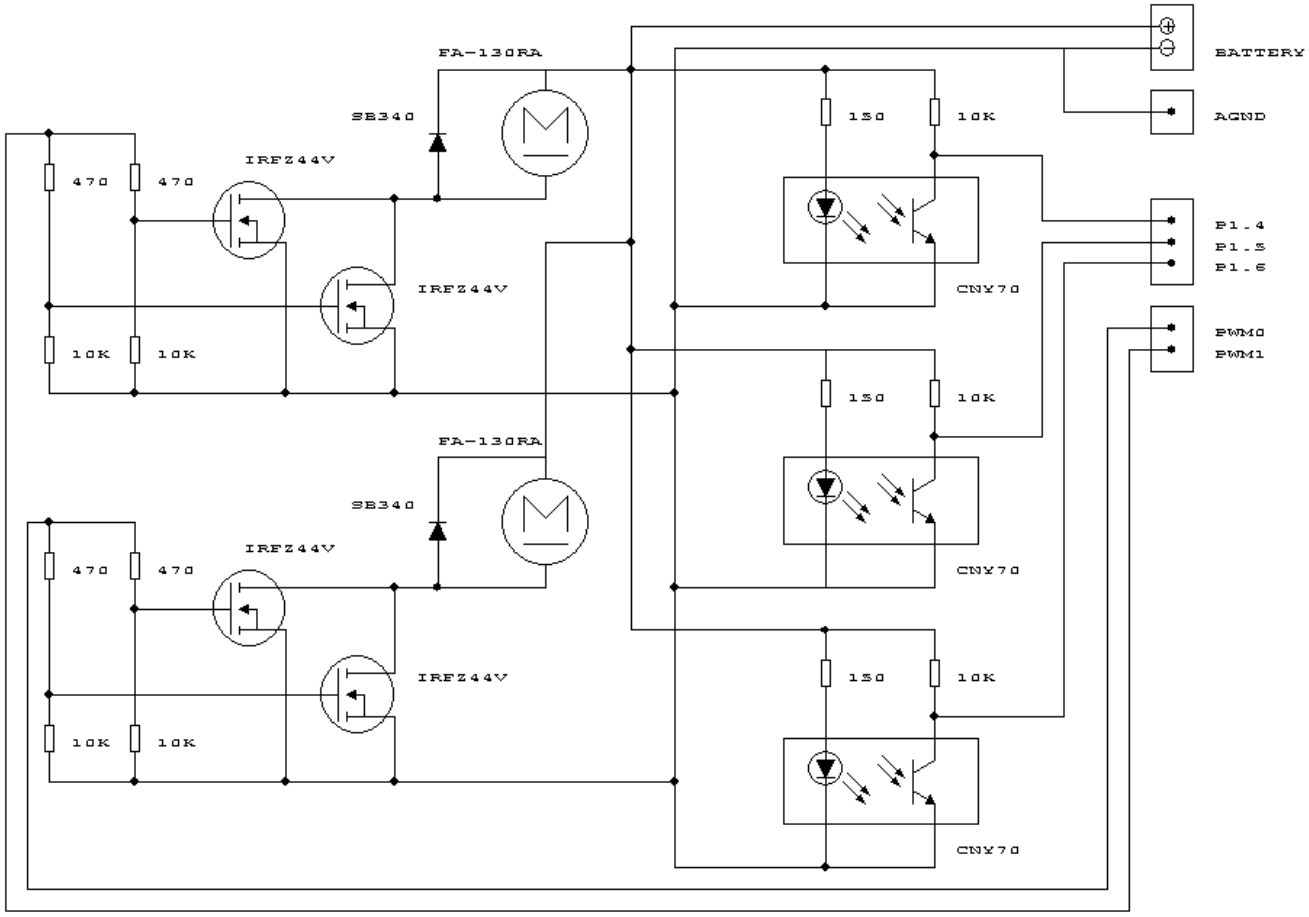


Figura 11: Elettronica del Line Follower Robot

3 ARCHITETTURA SOFTWARE

In questa sezione vediamo come è strutturata l'applicazione da un punto di vista software, ossia come è progettato il programma che è eseguito sul microcontrollore ADuC836.

3.1 SCHEDULER

Il software deve svolgere due compiti (quando non impostato in modalità di debug):

- lettura dei valori dai tre sensori e elaborazione di una informazione di attuazione;
- modifica del pilotaggio PWM dei motori allo scopo di seguire la linea sulla base di quanto rilevato dai sensori.

Queste due operazioni avvengono sempre nell'ordine di cui sopra e devono essere eseguite con regolarità in modo da essere sicuri che il percorso venga correttamente rilevato.

Per realizzare un software di questo tipo abbiamo utilizzato un semplice modulo di scheduling tratto dal libro "*Patterns for Time-Triggered Embedded Systems*" di Michael J. Pont. Il modulo in questione consente di sfruttare il timer/counter 2 allo scopo di dividere l'asse temporale in tick di scheduling della lunghezza di 1 ms. Ad ogni task che si introduce nel sistema viene assegnato un periodo in termini di tick di scheduling e un ritardo di attivazione (quanti tick devono passare prima che si attivi per la prima volta). Ogni tick comincia con l'arrivo dell'interruzione dal timer 2: la ISR svolge delle attività di manutenzione e aggiornamento della coda dei task, dopodiché si eseguono i task che in quel particolare ciclo devono essere eseguiti, uno dopo l'altro. Se i task terminano prima del tick successivo allora si imposta il microcontrollore in stato idle, risparmiando energia. Lo stato idle persiste fino alla prossima interruzione, che risveglia il processore e provoca (dopo l'aggiornamento ad opera della ISR) l'esecuzione di eventuali task.

Nel nostro caso abbiamo ipotizzato che la velocità massima per il robot segui-linea sia di circa 0.5 m/s e che la linea nera sia realizzata con del nastro adesivo di larghezza pari a circa 2.5 cm. Volendo ottenere almeno 2 campioni anche nel caso in cui si arrivasse in modo completamente ortogonale alla linea, abbiamo scelto un periodo di campionamento che ci garantisse almeno un campione ogni centimetro:

$$\frac{}{}$$

Il che significa che sia il task per la lettura ed elaborazione dei valori dai sensori che il task per la modifica dei valori di duty cycle dei segnali PWM, devono lavorare con un periodo di circa 20 ms.

Nella versione originale dello scheduler, il tick ha una durata di 1 ms, ma questa condizione operativa nel nostro caso comporta un forte overhead senza nessun vantaggio pratico.

Allo scopo di ridurre l'overhead, lo scheduler è stato modificato per ottenere un periodo di tick di scheduling pari a 20 ms. In questo modo abbiamo che i due task di lettura e attuazione dovranno essere eseguiti in ordine ad ogni tick, e l'overhead viene ridotto drasticamente.

In caso di funzionamento in modo debug, un terzo task sarà presente nel sistema al solo scopo di comunicare tramite seriale degli opportuni messaggi di debug. Questo task, pur utilizzando la seriale, non scatena nuove interruzioni in quanto gli interrupt provenienti dalla UART non vengono gestiti a software.

Usando le funzioni complesse della libreria di I/O standard (come ad esempio la printf()) abbiamo notato che la memoria interna al microcontrollore veniva saturata e alcune variabili allocate in tale memoria venivano sovrascritte col risultato che il funzionamento diventava impredicibile.

Abbiamo allora provveduto (nel modulo di debug fornito con il codice del LFR) a implementare una funzione apposita per la stampa delle stringhe di debug che risultasse *lightweight,* sfruttando, fra tutti i servizi forniti dalla libreria di I/O standard, solo la *putchar()*, la funzione di base per scrivere sulla seriale un singolo carattere. Questa funzione sfrutta una stringa statica di lunghezza fissa in memoria dati per immagazzinare i dati da emettere, la stringa da stampare viene perciò memorizzata in questa struttura dati e viene emessa carattere per carattere.

Il modulo di debug introduce un forte overhead nel software e dunque occorre evitare di utilizzarlo nella versione finale in quanto può impedire il rispetto dei vincoli temporali. Per contenere l'impredicibilità del software quando viene usato in modo debug, il task che sfrutta la seriale verrà eseguito molto di rado. In particolare abbiamo impostato un periodo di 25 tick di scheduling che con tick di 20 ms significa che un messaggio verrà stampato ogni mezzo secondo.

Nonostante la presenza dello scheduler impedisca l'utilizzo di interrupt interni o esterni durante la normale esecuzione, è comunque possibile utilizzare degli interrupt per bloccare l'esecuzione. Nel programma del LFR è stata infatti inserita una ISR ad alta priorità in risposta all'interrupt esterno *INT0* (mandata in esecuzione mediante la pressione del tasto apposito sulla Evaluation Board) che si occupa di bloccare il robot e spegnere il microcontrollore.

3.2 TASK

Come visto in precedenza, abbiamo due task che periodicamente vengono eseguiti:

1. il primo, costituito dalla funzione **Sensor_Read**, legge i valori della piedinatura connessa ai sensori ottici e, dopo un'opportuna elaborazione, stabilisce il movimento che il robot deve compiere, configurando precise variabili;
2. il secondo, costituito dalla funzione **PWM_Update**, legge i valori delle suddette variabili e in base ad esse regola la rotazione dei motori, andando a scrivere sui registri che determinano il duty cicle del segnale PWM.

3.2.1 ALGORITMO DI LINE FOLLOWING (SENSOR_READ)

Sfruttando i valori restituiti dai 3 sensori ottici riflessivi, il robot deve essere in grado di individuare l'andamento del tracciato e conseguentemente modificare la propria direzione per mantenersi sulla linea nera. Per farlo, è stato implementato un semplice algoritmo rappresentabile quasi interamente mediante una rete combinatoria: questo significa che conoscendo gli ingressi del sistema (ossia le rilevazioni dei sensori), escludendo alcuni casi particolari, è possibile determinare il movimento del robot.

Il LFR è in grado di curvare con due diverse velocità, è cioè in grado di affrontare curve più o meno secche: direzione e intensità della curva sono ricavate dai valori letti dal sensore. Questi in particolare sono considerati ingressi digitali e, all'interno del task di lettura, vengono negati, in modo da ottenere un bit pari a '1' se il sensore è posto su una superficie chiara, '0' se la superficie è scura. Ci riferiremo ai bit ottenuti dalla lettura (negata) dei piedini collegati ai sensori, con i nomi S2 (sensore di sinistra), S1 (sensore centrale) e S0 (sensore di destra).
È necessario specificare che i sensori distano circa 2 cm l'uno dall'altro e che la linea usata per comporre il tracciato ha una larghezza fissata a 2.5 cm, pertanto quando la linea dei sensori è perpendicolare al nastro nero, possiamo avere fino a due sensori che rilevano la superficie scura.

Il comportamento è il seguente:

- se solo il sensore centrale vede nero, il robot va dritto (S2.S1.S0 == 101);
- se due sensori adiacenti vedono nero (S2.S1.S0 == 001 ovvero 100) il robot curva lentamente nella direzione indicata (rispettivamente a sinistra e a destra);
- se solo un sensore laterale vede nero, il robot curva velocemente nella direzione del sensore (S2.S1.S0 == 011 ovvero 110).

In questo modo rispondiamo in maniera più marcata qualora il LFR si rendesse conto di essere prossimo a uscire dal percorso.

Consideriamo ad esempio il caso di una curva verso destra, partendo dalla condizione iniziale in cui il robot è centrato sulla linea nera:

1. nella fase precedente la curva il robot procede dritto, in quanto rileva che solo il sensore centrale è posto sulla linea (S2.S1.S0 == 101);
2. entrando nella curva, ad un certo punto anche il sensore laterale destro si trova sopra il nastro, pertanto il robot capisce di trovarsi in una curva e comincia a sterzare lentamente verso destra (S2.S1.S0 == 100);
3. qualora la curva sia particolarmente decisa, la modifica alla traiettoria del robot non sarà sufficiente a mantenerlo nel circuito, il LFR dunque devierà verso l'esterno della curva fintantoché solo il sensore destro restituirà valore 0 (S2.S1.S0 == 110); a questo punto l'algoritmo prevede un'azione ancora più accentuata sui motori, in modo da diminuire il raggio di curvatura e quindi permettere al dispositivo di rientrare in pista.

I passi appena esposti sono rappresentati nelle seguenti immagini. Se la sterzata del robot risultasse sufficiente, dopo il passo 3 seguiranno nuovamente, in ordine, il passo 2 e 1, per tornare così alla condizione iniziale.

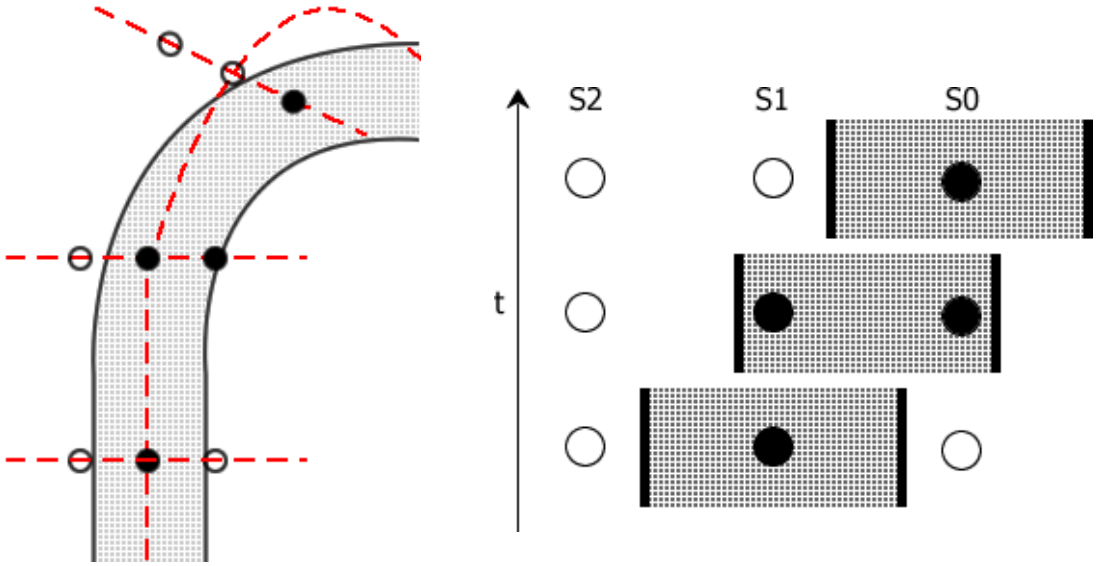


Figura 12: Esempio di tracciato, traiettoria percorsa dal LFR e rilevazioni dei sensori

Per quanto riguarda le combinazioni non discusse (S2.S1.S0 == 000 ovvero 111 ovvero 010), queste rappresentano situazioni anomale o casi particolari, la cui gestione richiede la conoscenza del comportamento passato del robot. Per questo motivo la reale implementazione dell'algoritmo è più complessa rispetto ad una semplice rete combinatoria. Nello specifico:

- *S2.S1.S0 == 000*

  questa configurazione può essere ottenuta nel caso si incontri una intersezione lungo il tracciato, oppure, nel caso di una curva ad 'S' molto stretta, rientrando in modo quasi perpendicolare al nastro. Nel primo caso (presupponendo che ci stessimo muovendo di moto rettilineo) l'intersezione è ignorata e si prosegue a dritto; nel secondo invece si curva in direzione opposta alla precedente: così facendo si supera la curva ad 'S' e si prosegue correttamente;

- *S2.S1.S0 == 111*

  La reazione a questa combinazione varia a seconda del momento in cui è rilevata: al reset del microcontrollore infatti questa ci indica che il robot è stato posto in una zona bianca, pertanto il LFR si muoverà di moto rettilineo fino a trovare la linea del tracciato. Da questo momento l'interpretazione della combinazione 111 risulterà diversa, ossia significherà l'uscita dal tracciato in seguito ad una curva particolarmente stretta e ad essa dunque corrisponderà una maggiore velocità di sterzata del robot;

- *S2.S1.S0 == 010*

  una rilevazione del genere si ha nel caso di biforcazione: il comportamento del robot è difficilmente predicibile in prossimità di un bivio, in quanto altamente influenzato dalla posizione dei sensori rispetto alla linea. A seconda infatti che il robot rilevi prima l'una o l'altra diramazione, la sua direzione sarà diversa; semplicemente allora manteniamo la curvatura decisa dall'algoritmo nei passi precedenti. Nel caso in cui invece il bivio sopraggiunga mentre il robot si muove in linea retta, si decide arbitrariamente di svoltare a destra.

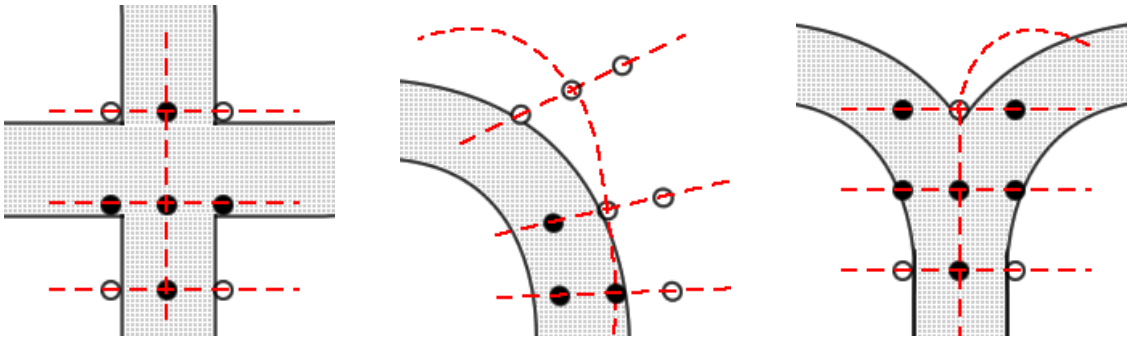


Figura 13: alcuni casi particolari: incrocio a 'T', fuoriuscita dalla curva, bivio

Riassumendo, risulta che gli elementi circuitali gestibili tramite l'algoritmo di Line Following sono complessivamente i seguenti:

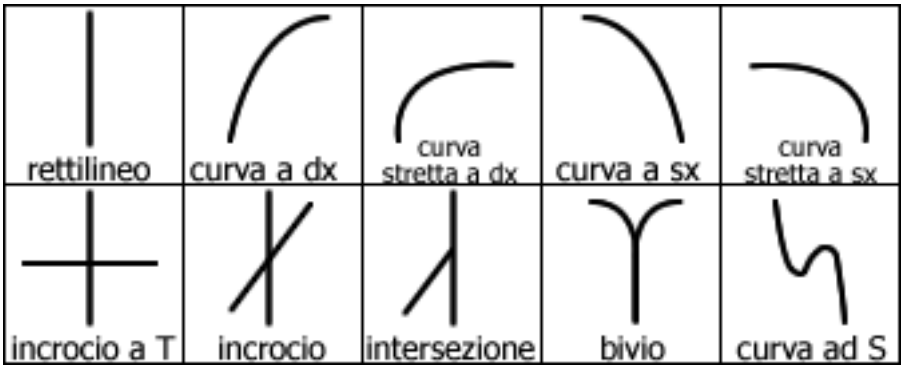


Figura 14: elementi circuitali gestibili dal LFR

3.2.2 PROCEDURA DI ATTUAZIONE (PWM_UPDATE)

Nella fase precedente, le conclusioni ricavate dall'algoritmo sono tradotte in opportuni valori di alcune variabili, accessibili anche dal task PWM_Update. Precisamente si impiegano 3 variabili bit, che contengono le informazioni sulla direzione e velocità del robot:

1. *Ahead/turn*: indica se il robot deve andare diritto o curvare (diritto == 0, curva == 1);
2. *Direction*: in caso di curva stabilisce la direzione (destra == 0, sinistra == 1);
3. *Speed*: indica la velocità di curva (lento == 0, veloce == 1).

Partendo dai valori riscontrati in queste variabili, si procede dunque all'aggiornamento dei registri PWM0L e PWM0H che regolano il duty cicle dei segnali PWM diretti rispettivamente al motore sinistro e destro.
In sostanza i due registri assumono lo stesso valore (PWM_MEAN) nel caso il robot si muova di moto rettilineo, mentre assumono valori diversi in caso di curva: il motore posto dalla parte della direzione di sterzata, deve ruotare più lentamente, e pertanto vede il proprio registro PWM diminuire di una quantità PWM_STEP rispetto al caso rettilineo; l'altro registro resta invece invariato. Se il bit di velocità è pari ad 1, il decremento, rispetto al caso precedente, è aumentato a 2 volte la quantità PWM_STEP.