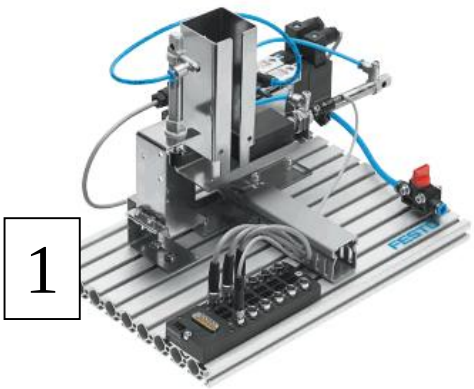
ISTITUTO PROF.LE DI STATO PER L'INDUSTRIA E L'ARTIGIANATO
"Antonio Guastaferro"

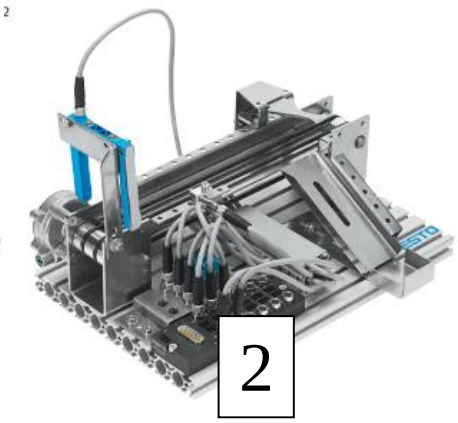
Viale dello Sport, 60 – 63074 San Benedetto del Tronto (AP)

UNIONE EUROPEA
FONDI STRUTTURALI EUROPEI
PON 2014-2020
PER LA SCUOLA - COMPETENZE E AMBIENTI PER L'APPRENDIMENTO (FSE-FESR)
MIUR
Ministero dell'Istruzione, dell'Università e della Ricerca
Dipartimento per la Programmazione
Direzione Generale per interventi in materia di edilizia scolastica, per la gestione dei fondi strutturali per l'istruzione e per l'innovazione digitale
Ufficio IV

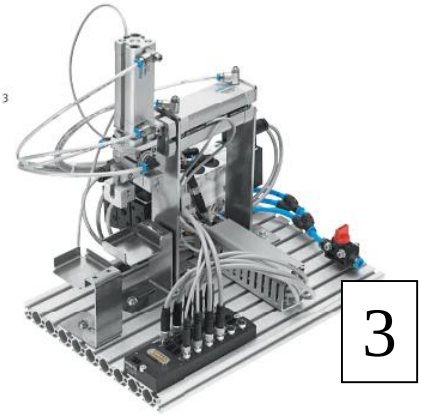
Shield Arduino-MecLab - Vers.2.2



Modulo magazzino a gravità con serbatoio
Stacking magazine station



Modulo nastro trasportatore
Conveyor station



Modulo manipolatore -
Handling station

MecLab® - aspetti generali

Tecnologie dell'Automazione a scuola

L'Automazione è uno dei settori a più rapida crescita nel mondo. Troviamo sistemi automatizzati praticamente in ogni ambito industriale e civile.
Festo è leader mondiale nel campo grazie alle proprie innovative soluzioni per l'industria di produzione e di processo.
L'azienda impiega questa esperienza per realizzare e commercializzare, attraverso il proprio marchio Festo Didactic, i migliori sistemi di apprendimento per lo sviluppo di competenze tecnico-scientifiche.
MecLab® è oggi il miglior sistema per le scuole superiori i cui curricula comprendano discipline scientifiche quali fisica, IT, sistemi ed automazione, scienza delle misure, e loro applicazioni. MecLab® consta di tre stazioni che riproducono i tipici processi che si ritrovano virtualmente in ogni sistema di produzione.

Contenuti didattici:

MecLab® è concepito per affrontare agevolmente in classe argomenti e modelli usualmente complessi, ma affascinanti quali :

1. Produzione industriale
2. Pianificazione, sviluppo e messa in funzione di sistemi automatizzati
3. Saper interpretare e sviluppare la documentazione tecnica (diagrammi di grandezze, schemi circuitali, parts lists, disegni tecnici)
4. Creare modelli di simulazione Comprendere il funzionamento di sistemi in retroazione
5. Analizzare le macchine con approccio sistemico.
6. Progettazione e realizzazione di circuiti elettronici e pneumatici.
7. Conoscere e saper impiegare attuatori pneumatici ed elettrici, sensori e controllori.
8. Applicazioni del computer per la programmazione e simulazione.

Le unità MecLab® possono essere utilizzate da sole o in sequenza.
Ognuna presenta propria autonomia sia funzionale che didattica, e presenta una serie completa di sperimentazioni su argomenti ed aspetti della disciplina.
In più, il collegamento delle stazioni in linea espande il campo di applicazione consentendo di affrontare sistemi automatizzati ad elevata complessità.

Hardware e programmazione

Il sistema comprende:

1. **Modulo magazzino a gravità con serbatoio - Stacking magazine station**
2. **Modulo nastro trasportatore - Conveyor station**
3. **Modulo manipolatore - Handling station**

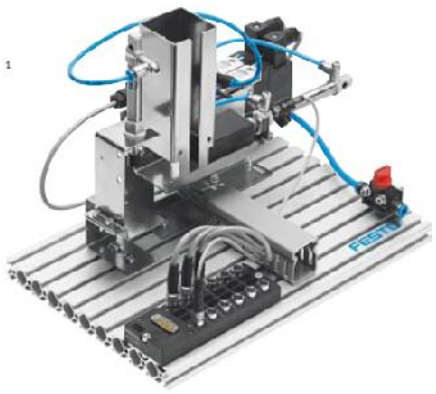
I moduli possono essere programmati con:

a) Mini PLC con software Ladder
b) FluidSim con l'interfaccia EasyPort (Programmazione e simulazione)
c) Arduino Mega con lo Shield Arduino-MecLab (Software IDE)

Descrizione Moduli

1) Modulo magazzino a gravità con serbatoio - *Stacking magazine station*

In qualsiasi linea automatizzata di produzione, i semilavorati necessitano sia di essere immagazzinati, che alimentati al processo in maniera ordinata.



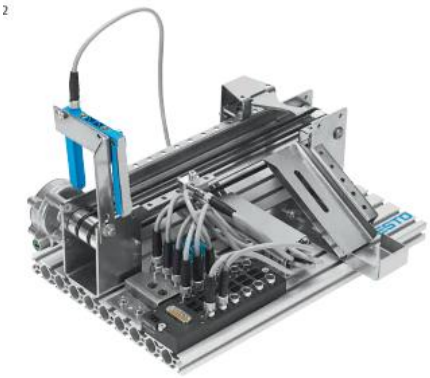
Questa funzione in MecLab® è svolta dall'Unità Magazzino. Questa immagazzina entrambe i tipi di componenti (scatola e coperchio)
nella sequenza desiderata, ed è in grado di separarle al momento dell'estrazione. I pezzi sono impilati nel magazzino a stack, ed espulsi mediante un cilindro orizzontale.
Il cilindro verticale serve invece a riprodurre un sistema di montaggio a pressione meccanica (es.: coperchio su scatola). Tutti i processi sono controllati da dispositivi elettro-pneumatici. Un finecorsa magnetico si può introdurre per tenere traccia della posizione del cilindro.

Programmabile tramite:
a) Mini PLC con software Ladder
b) FluidSim con l'interfaccia EasyPort (Programmazione e simulazione)
c) Arduino Mega con lo Shield Arduino-MecLab (Software IDE)
L'unità comprende:

- Modulo magazzino, Modulo Press-fit
- Connettore elettrico multi-pin
- 2 elettrovalvole, 2 cilindri. 1 finecorsa
- Piastra in alluminio profilato
- Set attrezzi, pezzi semilavorati, valigetta contenitore Systainer
- CD con sw FluidSIM® e documentazione

2) Modulo nastro trasportatore - *Conveyor station*

Il trasporto dei componenti tra due unità della linea è chiaramente funzione necessaria. Nel mondo reale vengono realizzati attraverso dispositivi non intelligenti, soprattutto i nastri trasportatori.



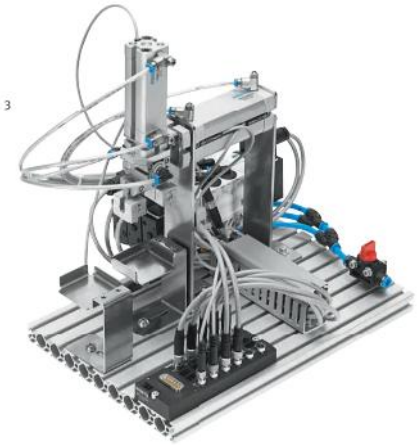
Questa unità di MecLab® fornisce quindi una simulazione realistica di un sistema di trasporto industriale. Il motore di azionamento del nastro è bidirezionale; in questo modo i pezzi vengono rilevati e selezionati da sensori secondo le caratteristiche. Pezzi non conformi alle specifiche vengono espulsi attraverso la slitta. Un finecorsa magnetico si può introdurre per tenere traccia della posizione del cilindro.

Programmabile tramite:
a) Mini PLC con software Ladder
b) FluidSim con l'interfaccia EasyPort (Programmazione e simulazione)
c) Arduino Mega con lo Shield Arduino-MecLab (Software IDE)
L'unità comprende:

- Modulo nastro trasportatore a trascinamento con motore DC
- 1 elettrobobina (stop), Connettore elettrico multi-pin
- 1 Sensore induttivo, 1 barriera a sensore ottico
- Piastra in alluminio profilato
- Set attrezzi, pezzi semilavorati, valigetta contenitore Systainer
- CD con sw FluidSIM® e documentazione

3) Modulo manipolatore - *Handling station*

Il ciclo di manipolazione è sempre presente in ogni processo, sia che si tratti di semplici operazioni di deposito, sia che riguardi complesse sequenze di assemblaggio.



I dispositivi che svolgono tali operazioni comprendono macchine diverse, dai semplici manipolatori cartesiani a 2 assi, sino ai complessi robot industriali a 6 assi.

Il manipolatore in MecLab® è realizzato con 2 cilindri pneumatici ed ha quindi 2 gradi di libertà.

Il pezzo viene bloccato da pinza ad azionamento ugualmente pneumatico.

.Il manipolatore viene impiegato per trasportare i pezzi da una stazione ad un altra; o anche per assemblare due parti dello stesso pezzo.

Programmabile tramite:

a) Mini PLC con software Ladder
b) FluidSim con l'interfaccia EasyPort (Programmazione e simulazione)
c) Arduino Mega con lo Shield Arduino-MecLab (Software IDE)

L'unità comprende:

- Modulo manipolatore
- 3 elettrovalvole, 4 finecorsa magnetici, 2 cilindri pneumatici a guida piana
- 1 pinza pneumatica,
- connettore multi-pin
- Piastra in alluminio profilato,
- Set attrezzi, pezzi semilavorati Valigetta contenitore, vaschette dei componenti
- CD con sw FluidSIM e documentazione

Shield Arduino-MecLab

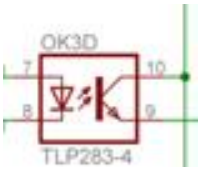
Lo Shield abbinato ad Arduino Mega permette il collegamento e la programmazione dei moduli MecLab tramite il sistema Arduino.

Gli elementi presenti nei moduli MecLab (sensori e attuatori) funzionano a 24V mentre Arduino gestisce segnali a 5V. Lo Shield effettua al traslazione dei livelli, in questo modo il segnali di Arduino sono compatibili con il segnali del MecLab.

In Fig.1 è riportato lo schema elettrico, principalmente è composto

- 32 fotoaccoppiatori, 16 di Input e 16 di Output.
- 2 connettori Centronics per il collegamento de moduli (è possibile collegare due moduli)

I fotoaccoppiatori si usano principalmente per trasferire un segnale, sia esso digitale o analogico, da un apparato ad un altro, tenendoli elettricamente isolati l'uno dall'altro.



Normalmente, un fotoaccoppiatore si presenta come un integrato plastico. Nell'interno di questo contenitore sono racchiusi

- un diodo emettitore all'infrarosso
- un fototransistor ricevente, anch'esso all'infrarosso

I due componenti, l'uno emittente e l'altro ricevente, sono separati tra loro tramite da un dielettrico trasparente e questo fa sì che l'accoppiamento tra di essi sia esclusivamente ottico.. Il fototransitor funziona come interruttore, la BASE viene pilotata dalla luce emessa da diodo emettitore

- Diodo emette luce - Transistor in saturazione (Interruttore ON)
- Diodo no emette luce - Transistor Interdetto (Interruttore OFF)

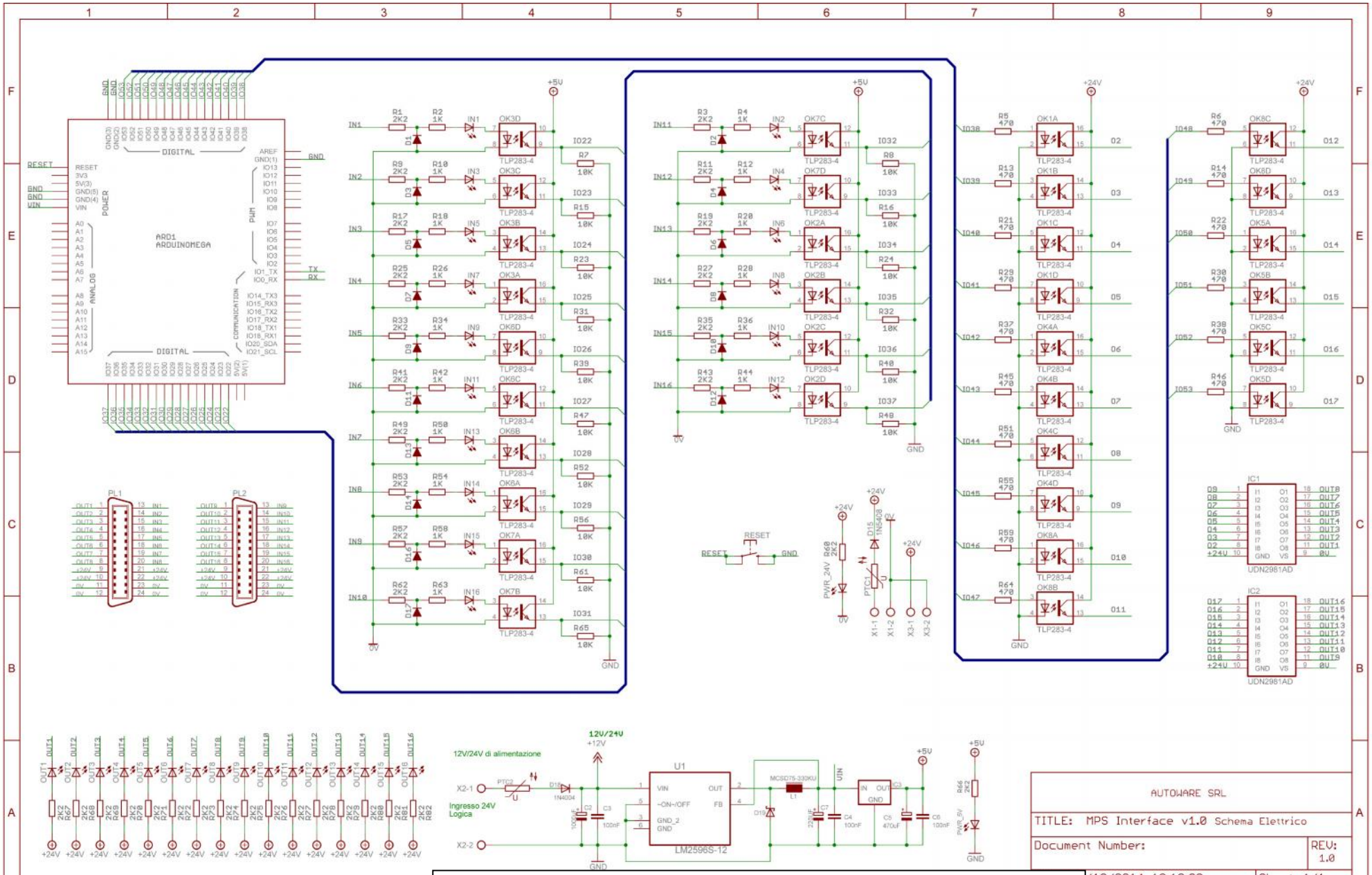


Fig. 1 - Schema elettrico - Shield Arduino MecLab

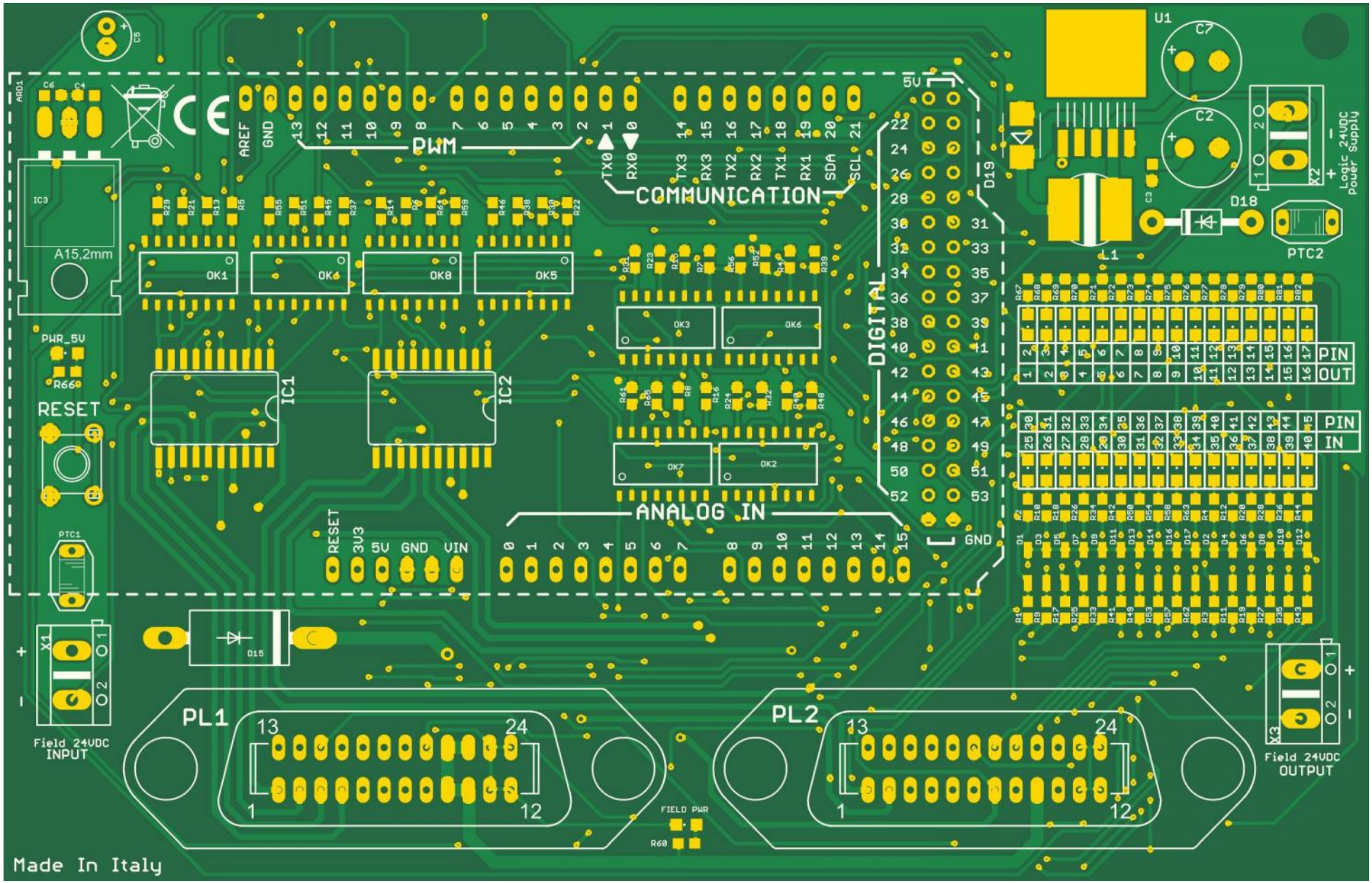
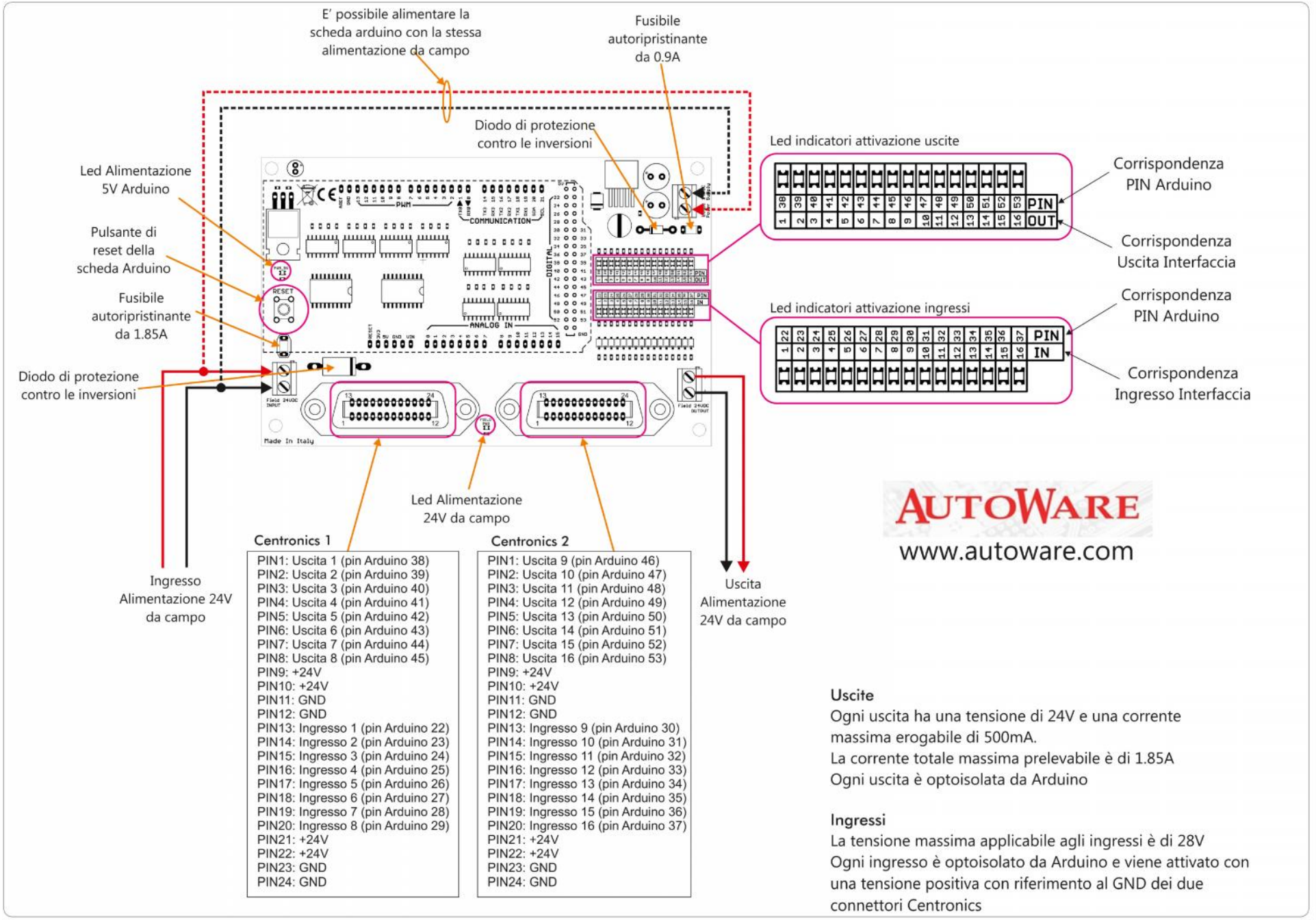


Fig. 2 - PCB - Shield Arduino MecLab



Centronics 1

PIN1: Uscita 1 (pin Arduino 38)
PIN2: Uscita 2 (pin Arduino 39)
PIN3: Uscita 3 (pin Arduino 40)
PIN4: Uscita 4 (pin Arduino 41)
PIN5: Uscita 5 (pin Arduino 42)
PIN6: Uscita 6 (pin Arduino 43)
PIN7: Uscita 7 (pin Arduino 44)
PIN8: Uscita 8 (pin Arduino 45)
PIN9: +24V
PIN10: +24V
PIN11: GND
PIN12: GND
PIN13: Ingresso 1 (pin Arduino 22)
PIN14: Ingresso 2 (pin Arduino 23)
PIN15: Ingresso 3 (pin Arduino 24)
PIN16: Ingresso 4 (pin Arduino 25)
PIN17: Ingresso 5 (pin Arduino 26)
PIN18: Ingresso 6 (pin Arduino 27)
PIN19: Ingresso 7 (pin Arduino 28)
PIN20: Ingresso 8 (pin Arduino 29)
PIN21: +24V
PIN22: +24V
PIN23: GND
PIN24: GND

Centronics 2

PIN1: Uscita 9 (pin Arduino 46)
PIN2: Uscita 10 (pin Arduino 47)
PIN3: Uscita 11 (pin Arduino 48)
PIN4: Uscita 12 (pin Arduino 49)
PIN5: Uscita 13 (pin Arduino 50)
PIN6: Uscita 14 (pin Arduino 51)
PIN7: Uscita 15 (pin Arduino 52)
PIN8: Uscita 16 (pin Arduino 53)
PIN9: +24V
PIN10: +24V
PIN11: GND
PIN12: GND
PIN13: Ingresso 9 (pin Arduino 30)
PIN14: Ingresso 10 (pin Arduino 31)
PIN15: Ingresso 11 (pin Arduino 32)
PIN16: Ingresso 12 (pin Arduino 33)
PIN17: Ingresso 13 (pin Arduino 34)
PIN18: Ingresso 14 (pin Arduino 35)
PIN19: Ingresso 15 (pin Arduino 36)
PIN20: Ingresso 16 (pin Arduino 37)
PIN21: +24V
PIN22: +24V
PIN23: GND
PIN24: GND

AUTOWARE
www.autoware.com

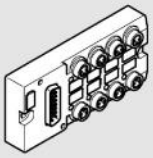
Uscite
Ogni uscita ha una tensione di 24V e una corrente massima erogabile di 500mA.
La corrente totale massima prelevabile è di 1.85A
Ogni uscita è optoisolata da Arduino

Ingressi
La tensione massima applicabile agli ingressi è di 28V
Ogni ingresso è optoisolato da Arduino e viene attivato con una tensione positiva con riferimento al GND dei due connettori Centronics

Fig. 3 -Shield Arduino MecLab - Lato connessioni: Centronics - Pin Arduino

In Fig.2 è riportato il PCB mentre in Fig.3 il lato connessioni (Centronics - Pin Arduino).
Il collegamento viene effettuato tramite un cavo (Centronics lato Shield Arduino, connettore Sub-D lato modulo MecLab) . I
In Fig.4 è riportato il Data Sheet del Distributore multipolare presente sui moduli

Distributore multipolare per connettore M8 con 8/12 attacchi



FESTO

Festo AG & Co. KG

Postfach
D-73726 Esslingen
++49/711/347-0
www.festo.com

(it) Descrizione breve

691 696
0812a

Originale: de

1 Indicazioni per l'utilizzatore

Leggere completamente questa descrizione breve.
Poi iniziare le operazioni di montaggio.
Prima della messa in servizio:
Vedi ISO 4414-1982 o DIN 24558 riguardo alle norme di sicurezza per installazione e impiego dei componenti pneumatici.

Avvertenza

- Le operazioni di montaggio e messa in servizio devono essere eseguite solo da personale qualificato e autorizzato.
- Disinserire l'alimentazione elettrica prima di iniziare le operazioni di installazione e manutenzione.
- Utilizzare esclusivamente alimentazioni elettriche in grado di garantire un sezionamento elettrico sicuro della tensione d'esercizio secondo IEC/DIN EN60204-1. Inoltre tenere presente le caratteristiche generali richieste ai circuiti elettrici PELV secondo IEC/DIN EN 60204-1.

2 Impiego ammesso

Distributore multipolare compatto per ingressi e uscite. Per il collegamento di sensori PNP e attuatori a 2 poli.
- Collegamento tramite connettore M8x1 a 3 poli (8 o 12 attacchi)
- Collegamento collettivo tramite connettore Sub-D a 15 poli
- Indicazione dello stato di commutazione mediante LED giallo

Il distributore multipolare compatto qui descritto per ingressi e uscite tipo MPV-E/A08/12-M8 è destinato al montaggio in una macchina o in un impianto di automazione. È assolutamente indispensabile osservare le avvertenze per la sicurezza riportate nella presente descrizione nonché l'impiego conforme alle prescrizioni del distributore multipolare.
Utilizzare il distributore solo nel modo seguente:
- in ambito industriale, nei limiti di impiego previsti
- nello stato originale, senza apportare modifiche non autorizzate
- in uno stato tecnicamente perfetto.

Quando si collegano componenti supplementari di tipo commerciale, ad sensori e attuatori, osservare i valori limite specificati per pressioni, temperature, dati elettrici, momenti, etc. Osservare le prescrizioni delle associazioni di categoria, dell'ente di sorveglianza tecnica, le normative VDE (associazione elettrotecnica tedesca) o le disposizioni nazionali in vigore.

3 Montaggio

Per evitare deformazioni e osservare il grado di protezione IP:
- Posizionare il distributore multipolare su una superficie completamente piana.
- Stringere le viti di fissaggio (vedi Fig. 1) applicando solo la coppia di serraggio indicata nei "Dati tecnici".

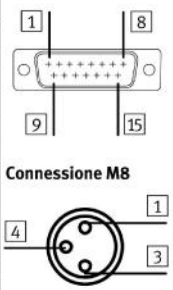
Nota

- Segnali del sensore 24 VCC a commutazione positiva
- Non è possibile alcuna connessione PE
- Nessuna protezione contro l'inversione di polarità e nessuna protezione da cortocircuito nel modulo.

4 Occupazione dei pin

Occupazione dei contatti connettore femmina M8 (secondo IEC 947-5-2) e connettore Sub-D (a 15 poli)

Connessione M8 Posto	Connettore femmina	Occupazione dei contatti	PIN Sub-D a 15 poli	Posizione di pin e connettori femmina
0	4	Linea di segnale	1	Connettore Sub-D
1	4	Linea di segnale	2	
2	4	Linea di segnale	3	
3	4	Linea di segnale	4	
4	4	Linea di segnale	5	
5	4	Linea di segnale	6	
6	4	Linea di segnale	7	
7	4	Linea di segnale	8	
8	4	Linea di segnale	9	
9	4	Linea di segnale	10	
10	4	Linea di segnale	11	
11	4	Linea di segnale	12	
0 ... 7 o 0 ... 11	1	24 VCC	13	
0 ... 7 o 0 ... 11	3	0 V	14 e 15	



5 Elementi di identificazione e di attacco

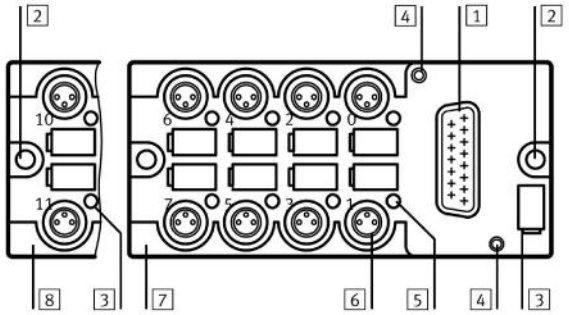


Fig. 1

1 Connettore Sub-D
2 Foro di fissaggio M4
3 Targhetta di identificazione
4 Fissaggio M3 per calotta connettore Sub
5 LED indicazione dello stato di commutazione
6 Attacco M8
7 MPV con 8 attacchi
8 MPV con 12 attacchi

6 Accessori

Nota

Gli accessori Festo sono riportati all'indirizzo:
➔ www.festo.com/catalogue
Per garantire il grado di protezione IP65 osservare quanto segue:
1. Per collegare il distributore, utilizzare cavo e connettore femmina multipolare dagli accessori Festo.
2. Chiudere gli attacchi non utilizzati con calotte valvola (M8).

7 Dati tecnici

Tipo	MPV-E/A-08-M8	MPV-E/A-12-M8
Dimensioni	98x 45x 20 mm	126x 45x 20 mm
Peso	ca. 92 g	ca. 125 g
Tensione d'esercizio	10 ... 30 VCC (non protetto contro l'inversione di polarità)	
Carico elettrico ammissibile	max. 1 A per attacco max. 6 A per corrente cumulativa	
Intervallo di temperatura	-20 ... +80 °C	
Grado di protezione secondo DIN EN 60529 Attacchi innestati e avvitati o dotati di calotte protettive	IP65	
Coppie di serraggio: – vite di fissaggio M4 – vite di fissaggio M3 – connettore M8	 0,8 ... 1,2 Nm 0,4 ... 0,8 Nm 0,25 ... 0,5 Nm	
Protezione contro le scosse elettriche (protezione contro contatto diretto e indiretto secondo IEC/DIN EN 60204-1)	mediante alimentatore PELV (Protected Extra Low Voltage)	
Materiali	Corpo: PA6.6 Inserti filettati: CuZn Colata: resina epossidica	

Fig.4 - Distributore multipolare

Esercizi con Modulo manipolatore - Handling station

Introduzione

In Fig. 5 è riportato il collegamento Arduino - Manipolatore

Gli elementi presenti nei moduli MecLab (sensori e attuatori) funzionano a 24V mentre Arduino gestisce segnali a 5V. Lo Shield effettua al traslazione dei livelli, in questo modo il segnali di Arduino sono compatibili con il segnali del MecLab.

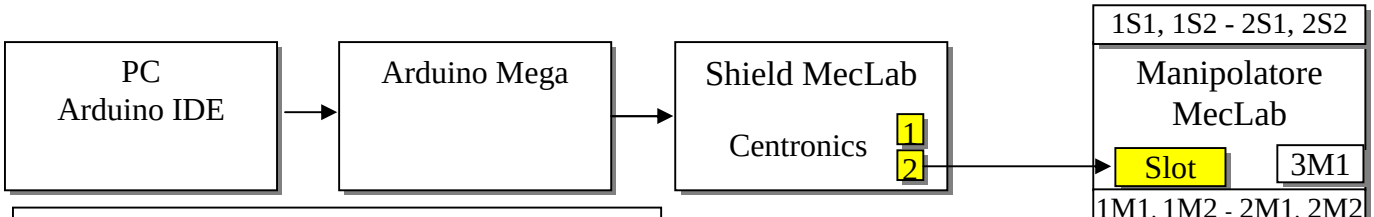


Fig.5 - Collegamento: Arduino -Manipolatore

In Fig. 6 è riportata la struttura del manipolatore in cui sono indicati gli elementi fondamentali

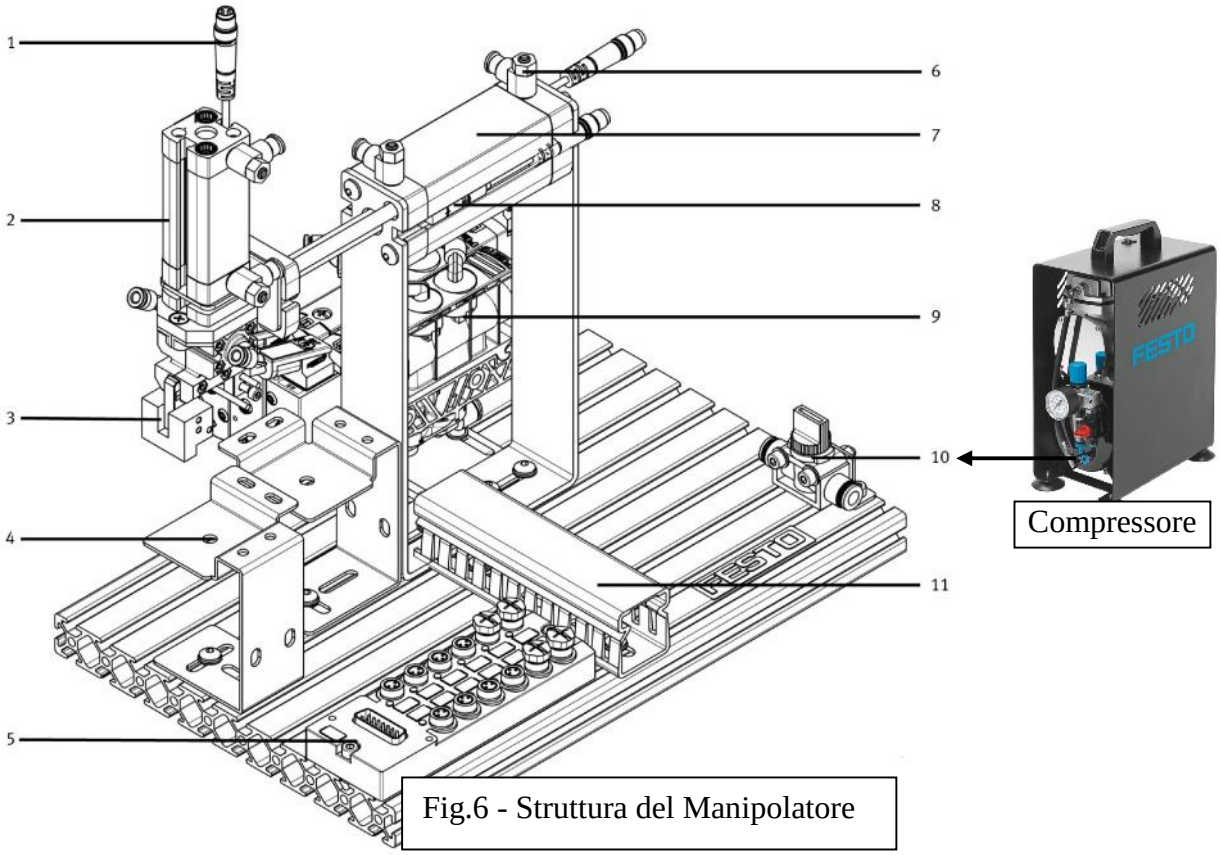
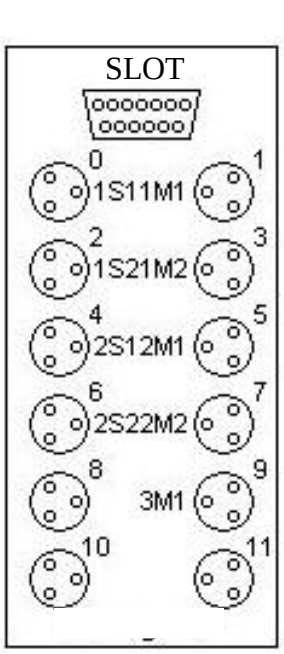


Fig.6 - Struttura del Manipolatore

Per il funzionamento occorre un compressore che va collegato al punto 10

Nella tabella seguente è riportata la spiegazione degli elementi fondamentali.

No.	Designation	Function within the station
1	Magnetic proximity sensor	Outputs a signal when a piston passes the sensor
2	Double-acting cylinder (guided)	Drive for movement in the z direction
3	Gripper	Grips the workpiece
4	Storage plate	Stores workpieces
5	Multi-pin plug distributor	Electrical connection for the sensors and actuators
6	One-way flow control valve	Regulates the speed of the cylinders
7	Double-acting cylinder (guided)	Drive for movement in the x direction
8	Magnetic proximity sensor	Outputs a signal when a piston passes the sensor
9	Solenoid valves	Control the movement of the cylinders
10	3/2-way hand valve	Shuts off the compressed air supply
11	Cable duct	Routes the cables



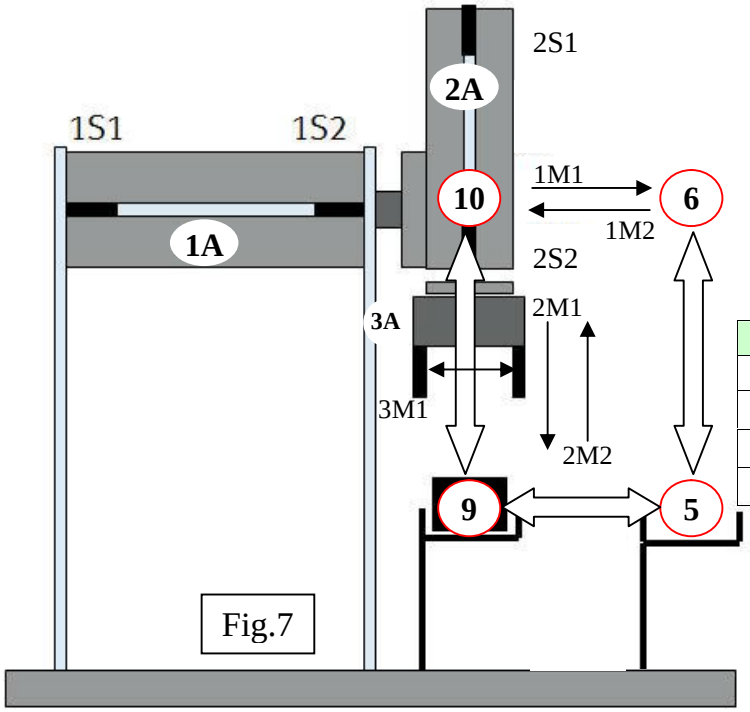


Fig.7

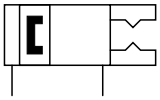
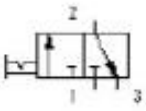
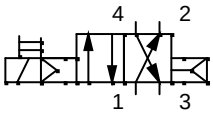
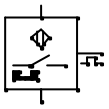
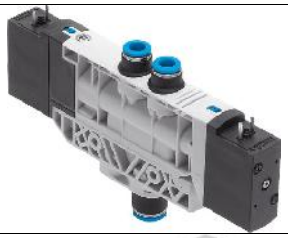
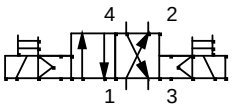
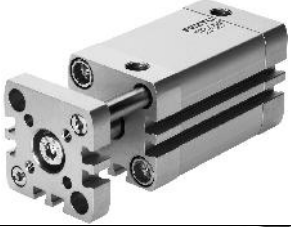
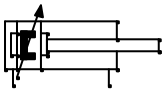
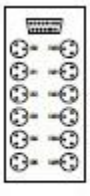
Cod. posizione	
10	6
9	5

1S1	1S2	2S1	2S2	Cod
1	0	1	0	**10**
0	1	1	0	**6**
0	1	0	1	**5**
1	0	0	1	**9**

In Fig. 7 sono riportati:

- Distributore multipolare
- Disegno della struttura dove sono evidenziate le posizioni possibili con i rispettivi codici
- Tabella di verità dei sensori (1S1, 1S2, 2S1, 2S2)

In tabella sono riportati i componenti del manipolatore con i rispettivi simboli.

N	Componente	Simbolo	Descrizione
1			Gripper
2			One-way flow control valve
3			**T-distributor for distributing the compressed air.**
4			**3/2-way stop cock for shutting off the compressed air and exhausting.**
5			4/2-way single solenoid valve
6			Inductive proximity sensor
7			4/2-way double solenoid valve
8			Double-acting cylinder
9			**Multi-pin plug distributor, interface for connecting all actuators and sensors of the conveyor station to the control PC.**

Il Manipolare dispone di:

- 4 **Sensori** induttivi di prossimità (1S1, 1S2, 2S1, 2S2) i sensori forniscono un livello alto ("1") oppure un livello basso ("0").
  La lettura dello stato viene effettuata con il comando *Var=digitalRead(pinsensore)*
- 2 **attuatori** (elettrovalvole) per il comando dei due cilindri a doppio effetto (1M1, 1M2, 2M1, 2M2).
  Il comando viene effettuato con il comando *digitalWrite (pinattuatore, HIGH/LOW)*
- 1 **attuatore** (elettrovalvola) per il comando della pinza (3M1)
  Il comando viene effettuato con il comando *digitalWrite (pinpinza,HIGH/LOW).*

Di seguito è riportata la tabella connessioni

Slot distributore - Arduino Mega con Shield MecLab (Centronics 2)

Tabella connessioni			
Slot - MultiPin	**Designation**	**Pin Arduino**	**Description**
0	1S1	30 - Input	Magnetic proximity sensor at cylinder 1A, rear
2	1S2	31 - Input	Magnetic proximity sensor at cylinder 1A, front
4	2S1	32 - Input	Magnetic proximity sensor at cylinder 2A, top
6	2S2	33 - Input	Magnetic proximity sensor at cylinder 2A, bottom
1	1M1	46 - Output	Advance cylinder 1
3	1M2	47 - Output	Retract cylinder 1
5	2M1	48 - Output	Advance cylinder 2
7	2M2	49 - Output	Retract cylinder 2
9	3M1	50 - Output	Close gripper

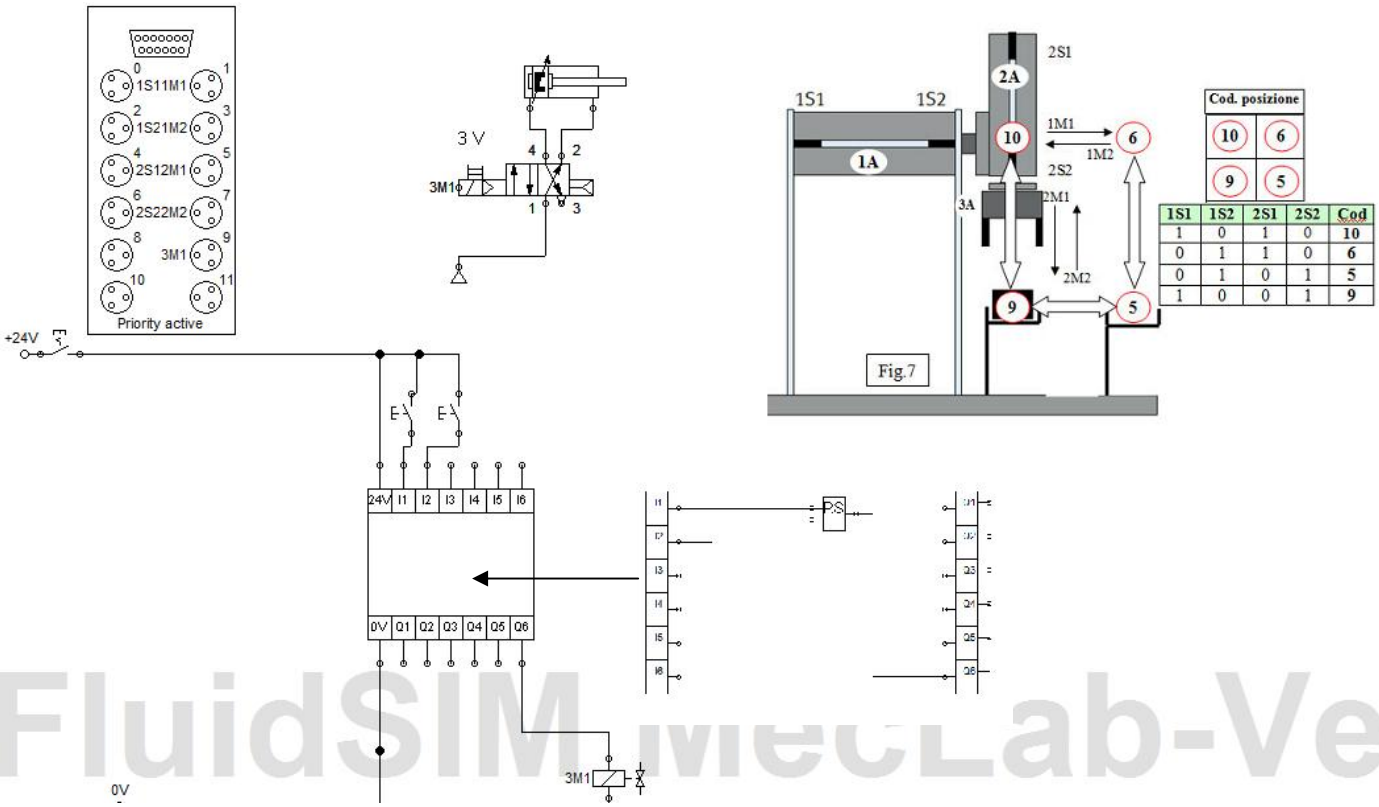
Esercizio 01 - Manipolatore MecLab - Comando Pinza

```
/*
  Es. 01 - Manipolatore MecLab  - Comando Pinza
           Arduino pin: 19 = Pulsante apertura pinza
                        14 = Pulsante chiusura pinza
                        50 = Elettrovalvola comando pinza
*/
byte p1=19, p2=14,pinza=50; // Variabili Pin
void setup()
{
pinMode(p1,INPUT);pinMode(p2,INPUT); pinMode(pinza,OUTPUT);

}

void loop()
{
 if(digitalRead(p1)==LOW)
 {
 digitalWrite(pinza,HIGH);
 }

 if(digitalRead(p2)==LOW)
 {
 digitalWrite(pinza,LOW);
 }
}
```



Esercizio 02 - Manipolatore MecLab - Comando Pinza Security

```
/*
  Es. 02 - Manipolatore MecLab  - Comando Pinza Security
           Esclude la pressione simultanea dei pulsanti
           Arduino pin: 19 = Pulsante apertura pinza
                        14 = Pulsante chiusura pinza
                        50 = Elettrovalvola comando pinza
*/
byte p1=19, p2=14,pinza=50; // Variabili Pin
void setup()
{
pinMode(p1,INPUT);pinMode(p2,INPUT);
}

void loop()
{
  if(digitalRead(p1)==LOW && digitalRead(p2)==HIGH )
 {
  digitalWrite(pinza,HIGH);
 }

  if(digitalRead(p2)==LOW && digitalRead(p1)==HIGH)
 {
  digitalWrite(pinza,LOW);
 }
}
```

Esercizio 03 - Manipolatore MecLab - Comando Cilindro Orizzontale

```
/*
  Es. 03 - Manipolatore MecLab  - Comando Cilindro Orizzontale
            Arduino pin: 18 = Pulsante comando cilindro - Avanti   (Advance)
                         17 = Pulsante comando cilindro - Indietro (Retract)
                         46 = Elettrovalvola comando cilindro - Avanti   (Advance)
                         47 = Elettrovalvola comando cilindro - Indietro (Retract)
*/
byte p1=18, p2=17;       // Pin Pulsanti
byte Pin1M1=46,Pin1M2=47; // Pin elettrovalvola orizzontale
void setup()
{
pinMode(p1,INPUT);     pinMode(p2,INPUT);
pinMode(Pin1M1,OUTPUT);pinMode(Pin1M2,OUTPUT);
digitalWrite(Pin1M1,LOW);digitalWrite(Pin1M2,LOW); // Disattivazione elettrovalvola cilindro Ori.
}
void loop()
{
  if(digitalRead(p1)==LOW)
 {
  digitalWrite(Pin1M1,HIGH);digitalWrite(Pin1M2,LOW);
 }
  if(digitalRead(p2)==LOW)
 {
  digitalWrite(Pin1M1,LOW); digitalWrite(Pin1M2,HIGH);
 }
}
```

Esercizio 04 - Manipolatore MecLab - Comando Cilindro Orizzontale Security

```
/*
  Es. 04 - Manipolatore MecLab  - Comando Cilindro Orizzontale
          Esclude la pressione simultanea dei pulsanti
          Arduino pin: 18 = Pulsante comando cilindro - Avanti   (Advance)
                       17 = Pulsante comando cilindro - Indietro (Retract)
                       46 = Elettrovalvola comando cilindro - Avanti   (Advance)
                       47 = Elettrovalvola comando cilindro - Indietro (Retract)
*/
byte p1=18, p2=17;       // Pin Pulsanti
byte Pin1M1=46,Pin1M2=47; // Pin elettrovalvola orizzontale
void setup(){
pinMode(p1,INPUT);     pinMode(p2,INPUT);
pinMode(Pin1M1,OUTPUT);pinMode(Pin1M2,OUTPUT);
digitalWrite(Pin1M1,LOW);digitalWrite(Pin1M2,LOW); // Disattivazione elettrovalvola cilindro Ori.
}
void loop(){
  if(digitalRead(p1)==LOW && digitalRead(p2)==HIGH)
 {
  digitalWrite(Pin1M1,HIGH);digitalWrite(Pin1M2,LOW);
 }
  if(digitalRead(p2)==LOW && digitalRead(p1)==HIGH)
 {
  digitalWrite(Pin1M1,LOW); digitalWrite(Pin1M2,HIGH);
 }
}
```

Esercizio 05 - Manipolatore MecLab - Comando Cilindro Orizzontale Security

```
/*
  Es. 05 - Manipolatore MecLab  - Comando Cilindro Orizzontale Security
            Esclude la pressione simultanea dei pulsanti
           Arduino pin: 16 = Pulsante comando cilindro - Basso   (Advance)
                        15 = Pulsante comando cilindro - Alto    (Retract)
                        48 = Elettrovalvola comando cilindro - Basso (Advance)
                       49 = Elettrovalvola comando cilindro - Alto  (Retract)
*/
byte p1=16, p2=15;        // Pin Pulsanti
byte Pin2M1=48,Pin2M2=49; // Pin elettrovalvola verticale
void setup()
{
pinMode(p1,INPUT);     pinMode(p2,INPUT);
pinMode(Pin2M1,OUTPUT);pinMode(Pin2M2,OUTPUT);
digitalWrite(Pin2M1,LOW);digitalWrite(Pin2M2,LOW); // Disattivazione elettrovalvola cilindro Vert.
}
void loop()
{
  if(digitalRead(p1)==LOW && digitalRead(p2)==HIGH)
 {
  digitalWrite(Pin2M1,HIGH);digitalWrite(Pin2M2,LOW);
 }
  if(digitalRead(p2)==LOW && digitalRead(p1)==HIGH)
 {
  digitalWrite(Pin2M1,LOW); digitalWrite(Pin2M2,HIGH);
 }
}
```

Esercizio 06 - Manipolatore MecLab - Lettura sensori cilindro orizzontale

```
/*
  Es. 06 -Manipolatore Mec Lab  - Lettura  sensori cilindro orizzontale
            Arduino pin: 30= (1S1) sensore F.C. orizzontale 31= (1S2)sensore F.C. orizzontale
*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F,20,4);
boolean OS1,OS2; //variabili stato dei sensoni (1S1, 1S2)
byte pin1S1=30,pin1S2=31;  // Assegnazione Pin sensori

void setup(){
  lcd.init(); lcd.clear();lcd.backlight();
  lcd.setCursor(0,0); lcd.print("Manipolatore Lett.S.");
  lcd.setCursor(0,1); lcd.print("Sensori Orinzontali");
  lcd.setCursor(0,2); lcd.print("1S1 1S2");
  pinMode(pin1S1,INPUT);pinMode(pin1S2,INPUT); // Modalità (INPUT)
}
void loop()  {
    OS1=digitalRead(pin1S1); OS2=digitalRead(pin1S2); // Lettura stato sensori orizzontali

    // Visualizzazione sul display LCD 20x4
    lcd.setCursor(1,3);lcd.print(OS1);
    lcd.setCursor(5,3);lcd.print(OS2);
}
```

Esercizio 07 - Manipolatore MecLab - Lettura sensori cilindro verticale

```
/*
  Es. 07 -Manipolatore Mec Lab  - Lettura  sensori cilindro verticale
          Arduino pin: 32= (2S1) sensore F.C. orizzontale 33= (2S2)sensore F.C. orizzontale
*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F,20,4);
boolean VS1,VS2; //variabili stato dei sensoni (2S1, 2S2)
byte pin2S1=32,pin2S2=33;  // Assegnazione Pin sensori

void setup()
{
  lcd.init(); lcd.clear();lcd.backlight();
  lcd.setCursor(0,0); lcd.print("Manipolatore Lett.S.");
  lcd.setCursor(0,1); lcd.print("Sensori Verticali");
  lcd.setCursor(0,2); lcd.print("2S1 2S2");
  pinMode(pin2S1,INPUT);pinMode(pin2S2,INPUT); // Modalità (INPUT)
}

void loop()
  {
    VS1=digitalRead(pin2S1); VS2=digitalRead(pin2S2); // Lettura stato sensori verticali

    // Visualizzazione sul display LCD 20x4
    lcd.setCursor(1,3);lcd.print(VS1);
    lcd.setCursor(5,3);lcd.print(VS2);
}
```

Esercizio 08 - Manipolatore MecLab - Lab - Lettura posizione

```
/*
  Es. 08 - Manipolatore Mec Lab  - Lettura  posizione
           Arduino pin: 30= (1S1) sensore F.C. orizzontale     31= (1S2)sensore F.C. orizzontale
                        32= (2S1) sensore F.C. Verticale       33= (2S2) sensore F.C. Verticale
        1S1   1S2
       ------------
   2S1  | 10  |  6 |
      ------------
   2S2  | 9   |  5 |
       ------------
*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F,20,4);
boolean OS1,OS2,VS1,VS2; //variabili stato dei sensoni (1S1, 1S2, 2S1, 2S2)
byte pin1S1=30,pin1S2=31,pin2S1=32,pin2S2=33;  // Assegnazione Pin sensori
byte posizione=0; // Variabile posizione
void setup()
{
  lcd.init(); lcd.clear();lcd.backlight();
  lcd.setCursor(0,0); lcd.print("Manipolatore Lett.P.");
  pinMode(pin1S1,INPUT);pinMode(pin1S2,INPUT);
pinMode(pin2S1,INPUT);pinMode(pin2S2,INPUT); // Modalità (INPUT)
```

```
}

void loop()
  {
    OS1=digitalRead(pin1S1); OS2=digitalRead(pin1S2); // Lettura stato sensori orizzontali
    VS1=digitalRead(pin2S1); VS2=digitalRead(pin2S2); // Lettura stato sensori verticali
    posizione=8*OS1+4*OS2+2*VS1+1*VS2; // Calcolo posizione
    // Visualizzazione sul display LCD 20x4
    lcd.setCursor(0,1);  lcd.print("1S1  1S2  2S1  2S2");
    lcd.setCursor(1,2);  lcd.print(OS1);
    lcd.setCursor(6,2);  lcd.print(OS2);
    lcd.setCursor(11,2); lcd.print(VS1);
    lcd.setCursor(16,2); lcd.print(VS2);
    lcd.setCursor(0,3); lcd.print("Posizione=   ");lcd.setCursor(10,3);lcd.print(posizione);
}
```

Esercizio 9 - Manipolatore MecLab - Spostamento Manuale con 6 pulsanti

```
/*
  Es. 09 - Manipolatore MecLab  - Spostamento Manuale con 6 pulsanti
            Arduino pin: 30= (1S1) sensore F.C. Orizzontale        31= (1S2) sensore F.C. Orizzontale
                         32= (2S1) sensore F.C. Verticale          33= (2S2) sensore F.C. Verticale
                         46= (1M1) Output Advance cylinder 1   47= (1M2) Output Retract cylinder 1
                         48= (2M1) Output Advance cylinder 2   49= (2M2) Output Retract cylinder 2
                        50= (3M1) Output Close gripper (Pinza)

        1S1   1S2
       ------------
   2S1  | 10  |  6 |
      ------------
   2S2  | 9   |  5 |
       ------------
*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F,20,4);
boolean OS1,OS2; //variabili fine corsa cilindro orizzontale
boolean VS1,VS2; //variabili fine corsa cilindro verticale
// Assegnazione Pin di Arduino
byte pin1M1=46, pin1M2=47,pin1S1=30,pin1S2=31;  // Orizzontale (Sensori e cilindri)
byte pin2M1=48, pin2M2=49,pin2S1=32,pin2S2=33;  // Verticale   (Sensori e cilindri)
byte pinza=50; // Gripper (Pinza)
byte PO1=18,PO2=17,PV1=16,PV2=15,Pg1=19,Pg2=14;  // pin pulsanti Pg=Pinza
byte posizione=0; //  Variabile cod. posizione cilindri
int ritardo=1500;
void setup()
{
  lcd.init(); lcd.clear();lcd.backlight();
  lcd.setCursor(0,0); lcd.print("Manipolatore P.Step");
  lcd.setCursor(0,1); lcd.print("P1 P2   P3 P4  P5 P6");
  lcd.setCursor(0,2); lcd.print("C  A    SX DX  DW UP");
  pinMode (pin1M1,OUTPUT);pinMode (pin1M2,OUTPUT);pinMode (pin2M1,OUTPUT); pinMode
(pin2M2,OUTPUT);pinMode(pinza,OUTPUT);
  pinMode(PO1,INPUT);pinMode(PO2,INPUT); pinMode(PV1,INPUT);pinMode(PV2,INPUT);
```

```
  pinMode(Pg1,INPUT);pinMode(Pg2,INPUT); // Pulsanti comando pinza
  OS1=digitalRead(pin1S1); OS2=digitalRead(pin1S2);
  VS1=digitalRead(pin2S1); VS2=digitalRead(pin2S2);
  posizione=8*OS1+4*OS2+2*VS1+1*VS2;
  // Imposta i cilindri nella posizione iniziale (Cod.10)
  digitalWrite(pin2M1,LOW); delay(500);digitalWrite(pin2M2,HIGH); delay(500);
  digitalWrite(pin1M1,LOW); delay(500);digitalWrite(pin1M2,HIGH); delay(500);
  digitalWrite(pin2M2,LOW); digitalWrite(pin1M2,LOW); digitalWrite(pinza,LOW);
  lcd.setCursor(0,3); lcd.print("Aperta");
  delay(ritardo);
}
void loop()
  {
    lcd.setCursor(8,3); lcd.print(digitalRead(pin1S1));lcd.setCursor(11,3); lcd.print(digitalRead(pin1S2));
    lcd.setCursor(15,3); lcd.print(digitalRead(pin2S1));lcd.setCursor(18,3); lcd.print(digitalRead(pin2S2));
    if (digitalRead(PO1)==LOW & digitalRead(PO2)==HIGH)
    {
      digitalWrite(pin1M1,HIGH); digitalWrite(pin1M2,LOW);
    }
    if (digitalRead(PO1)==HIGH & digitalRead(PO2)==LOW)
    {
      digitalWrite(pin1M1,LOW); digitalWrite(pin1M2,HIGH);
    }
    if (digitalRead(PV1)==LOW & digitalRead(PV2)==HIGH)
    {
      digitalWrite(pin2M1,HIGH); digitalWrite(pin2M2,LOW);
    }
    if (digitalRead(PV1)==HIGH & digitalRead(PV2)==LOW)
    {
      digitalWrite(pin2M1,LOW); digitalWrite(pin2M2,HIGH);
    }
if (digitalRead(Pg1)==LOW & digitalRead(Pg2)==HIGH)
    {
      digitalWrite(pinza,HIGH);
      lcd.setCursor(0,3); lcd.print("Aperta");
      lcd.setCursor(0,3); lcd.print("Chiusa");
    }
if(digitalRead(Pg1)==HIGH & digitalRead(Pg2)==LOW)
    {
      digitalWrite(pinza,LOW);
      lcd.setCursor(0,3); lcd.print("Aperta");
    }
    }
```

Esercizio 10 - Manipolatore MecLab - Sequenza automatica- Avvio con Pulsante

```
/*
  Es. 10 - Manipolatore MecLab  - Sequenza automatica- Avvio con Pulsante
            Arduino pin: 30= (1S1) sensore F.C. Orizzontale        31= (1S2) sensore F.C. Orizzontale
                         32= (2S1) sensore F.C. Verticale          33= (2S2) sensore F.C. Verticale
                         46= (1M1) Output Advance cylinder 1   47= (1M2) Output Retract cylinder 1
                         48= (2M1) Output Advance cylinder 2   49= (2M2) Output Retract cylinder 2
                         50= (3M1) Output Close gripper (Pinza)

        1S1   1S2
       ------------
   2S1  | 10  |  6 |
      ------------
   2S2  | 9   |  5 |
       ------------
*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F,20,4);
boolean OS1,OS2; //Variabili fine corsa cilindro orizzontale
boolean VS1,VS2; //Variabili fine corsa cilindro verticale
byte pin1S1=30,pin1S2=31,pin2S1=32,pin2S2=33;  // Pin Sensori F.C.
byte pin1M1=46,pin1M2=47,pin2M1=48,pin2M2=49; //  Pin elettrovalvole
byte pinza=50;                          // pin pinza
byte posizione=0; // Posizione cilindri
byte Pstart=19;   // Pulsante Start sequenza
int ritardo=1500;
boolean Vstart=HIGH;  // Stato pulsante Start
void setup()
{
  lcd.init(); lcd.clear();lcd.backlight();
  lcd.setCursor(0,0); lcd.print("Manipolatore MecLab");
  pinMode (pin1M1,OUTPUT);pinMode (pin1M2,OUTPUT);pinMode (pin2M1,OUTPUT); pinMode
(pin2M2,OUTPUT);pinMode(pinza,OUTPUT);
  pinMode(pin1S1,INPUT);pinMode(pin1S2,INPUT);
pinMode(pin2S1,INPUT);pinMode(pin2S2,INPUT);
  pinMode(Pstart,INPUT);
  OS1=digitalRead(pin1S1); OS2=digitalRead(pin1S2);
  VS1=digitalRead(pin2S1); VS2=digitalRead(pin2S2);
  posizione=8*OS1+4*OS2+2*VS1+1*VS2;
  // Imposta i cilindri nella posizione iniziale (Cod.10)
  digitalWrite(pin2M1,LOW); delay(500);digitalWrite(pin2M2,HIGH); delay(500);
  digitalWrite(pin1M1,LOW); delay(500);digitalWrite(pin1M2,HIGH); delay(500);
  digitalWrite(pin2M2,LOW);digitalWrite(pin1M2,LOW); digitalWrite(pinza,LOW);
  delay(ritardo);
}

void loop()
  {
    lcd.setCursor(0,1); lcd.print("Premi Start  ");
 while(Vstart==HIGH)
  {
  Vstart=digitalRead(Pstart);
```

```
  }
  Vstart=HIGH;
  lcd.setCursor(0,1); lcd.print("Mov    10 - 9");
digitalWrite(pin2M1,HIGH); digitalWrite(pin2M2,LOW);
delay(ritardo);
 lcd.setCursor(0,1); lcd.print("Chiude  Pinza");digitalWrite(pinza,HIGH);  delay(ritardo);
 lcd.setCursor(0,1); lcd.print("Mov    9 - 10");
digitalWrite(pin2M1,LOW); digitalWrite(pin2M2,HIGH);
delay(ritardo);
 lcd.setCursor(0,1); lcd.print("Mov    10 - 6");
digitalWrite(pin1M1,HIGH); digitalWrite(pin1M2,LOW);
delay(ritardo);
lcd.setCursor(0,1); lcd.print("Mov     6 - 5");
digitalWrite(pin2M1,HIGH); digitalWrite(pin2M2,LOW);
delay(ritardo);
lcd.setCursor(0,1);  lcd.print("Apre Pinza   "); digitalWrite(pinza,LOW);delay(ritardo);
 lcd.setCursor(0,1); lcd.print("Mov    5 - 6"); digitalWrite(pin2M1,LOW);
digitalWrite(pin2M2,HIGH);delay(ritardo);
 lcd.setCursor(0,1); lcd.print("Mov    6 - 10");digitalWrite(pin1M1,LOW);
digitalWrite(pin1M2,HIGH);delay (ritardo);
 lcd.setCursor(0,1); lcd.print("Fine Ciclo   ");delay (ritardo);
}
```