ISTITUTO TECNICO INDUSTRIALE STATALE
"ENRICO FERMI"

Andrea Bertolani

Simulazione di un impianto industriale per la manipolazione degli oggetti.

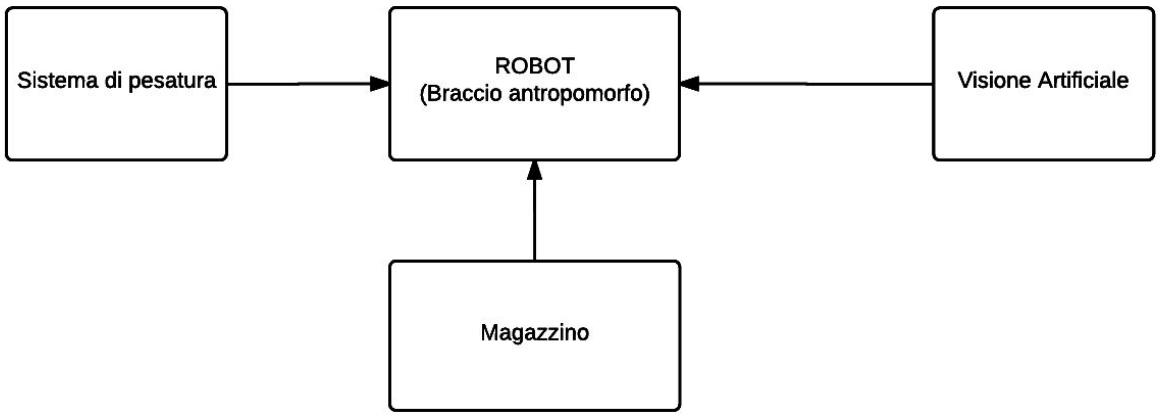
Anno Scolastico 2014/2015

INDICE:

1. Presentazione Progetto.
2. Cella di carico.
    2.1. Schema elettrico.
        2.1.1. Alimentazioni e tensioni di riferimento.
        2.1.2. Amplificazione segnale.
        2.1.3. Conversione segnale.
        2.1.4. Trasmissione dati acquisiti.
        2.1.5. Considerazioni generali.
    2.2. Circuito stampato.
3. Server di visione.
    3.1. Rilevamento oggetti.
    3.2. Parametri di rilevamento.
    3.3. Trasformazione coordinate.
    3.4. Trasmissione coordinate.
4. Robot.
    4.1. Caratteristiche.
    4.2. Linguaggio AS.
5. Programmi.
    5.1. Programma di visione.
    5.2. Programma di pesatura.
    5.3. Programma di movimento(AS)

1. Presentazione progetto

Il progetto prevede l'utilizzo di un robot industriale (braccio antropomorfo a 6 gradi di libertà) per la scelta di pezzi sulla base o del colore o del peso.
Il sistema di visione è stato programmato per riconoscere la posizione ed il colore di alcuni oggetti (cubi di acciaio colorati di varie dimensioni).
Dopo aver individuato la posizione, il robot preleva i pezzi e li pesa utilizzando una cella di carico, il cui circuito di condizionamento è parte del progetto stesso, connessa in una rete locale con il robot, la videocamera ed il server di elaborazione.
Il server di elaborazione è composto da un computer che riceve le immagini della videocamera e le elabora per determinare la posizione del pezzo e il suo colore. La comunicazione avviene tramite protocollo TCP/IP; i dati vengono inviati secondo un protocollo predefinito nel formato "posizionex, posizioney, colore;".
Il peso del pezzo può essere memorizzato in appositi file database, mentre la nuova disposizione del pezzo può avvenire sulla base o del colore o del peso.



In particolare, con il progetto, sono state gestite le seguenti parti:

1) Realizzazione della scheda per la pesatura dei pezzi, con comunicazione del dato al PC di controllo.
2) Software della telecamera per il riconoscimento degli oggetti e del colore.
3) Programmazione del robot per la movimentazione dei pezzi (linguaggio AS Kawasaki)

2 Cella di carico.

2.1 Schema elettrico del circuito di condizionamento.

Lo schema elettrico della cella di carico è allegato con il nome “cella_carico_sch.pdf”

2.1.1 Alimentazioni e tensioni di riferimento.

Il sistema fornisce solo un’alimentazione singola a 24V ma per alimentare gli integrati e per regolare in maniera ottimale la tara della bilancia si è resa necessaria un’alimentazione anche negativa, quindi per ottenere un'alimentazione duale ±12V si é scelto di usare due alimentatori switching integrati (IC5 e IC6) che hanno una buona efficienza (circa 80%) e che consentono di avere tensioni negative rispetto alla massa.
La cella di carico richiede un’alimentazione molto precisa a 10V per ottenere un buon segnale in uscita dal ponte resistivo: è stato quindi usato un alimentatore lineare LM317 (IC10) in quanto la corrente richiesta ed il drop-out sono molto bassi quindi la dissipazione dell’energia in eccesso in calore è minima; come riferimento di tensione per IC10 si é scelto un REF01 (IC9) per la sua elevata precisione e stabilità termica.
La conversione analogico-digitale necessaria per trasmettere il dato a distanza necessita di una tensione di riferimento precisa e stabile nel tempo per ottenere una conversione AD efficace, a tale scopo è stato inserito IC1 (REF02) che genera una tensione di 5,000V.

2.1.2 Amplificazione del segnale proveniente dalla cella di carico.

La cella di carico ha una sensibilità di 2mV/V, cioè fornisce in uscita un segnale differenziale di 2mV per ogni volt di alimentazione a pieno carico.
Alimentando il ponte a 10V la tensione differenziale massima è di 20mV, è quindi chiaro che il segnale deve essere amplificato per poter essere utilizzato.
La tensione massima è stata scelta sulla base della tensione di fondoscala del convertitore analogico/digitale:

$$Vomax = VFS = 5,00V$$

È stato usato l’amplificatore differenziale per strumentazione INA114 che ha diversi pregi:

- separa l’impedenza del ponte dal resto del circuito;
- amplifica il segnale differenziale;

- riferisce il segnale differenziale a massa;
- elimina i disturbi di modo comune;
- aggiunge un offset al segnale di uscita per gestire la tara.

Il guadagno introdotto dall'amplificatore è dato dalla resistenza fissa R5 e dal trimmer R6 poste in serie, il calcolo per il dimensionamento delle resistenze è stato fatto utilizzando le seguenti formule:

$$G = \frac{Vomax}{Vimax} = \frac{5[V]}{20 * 10^{-3}} = 250 \quad [1]$$

$$G = 1 + \frac{50 * 10^3}{Rg} \quad [2]$$

Unendo la [1] e la [2] si ottiene un'unica formula:

$$250 = 1 + \frac{50 * 10^3}{Rg} \quad [3]$$

Da cui si ricava Rg:

$$Rg = \frac{50 * 10^3}{249} = 200\Omega$$

Per la sua realizzazione è stato scelta una serie di 2 resistenze, una fissa da 100Ω ed una variabile da 200Ω multigiro.

2.1.3 Conversione segnale

Per la conversione del segnale è stato utilizzato l'ADC integrato nel microcontrollore ATMEGA328P della ATMEL che possiede le seguenti caratteristiche:
- Tipologia: ADC ad approssimazioni successive.
- Numero di bit: 10 bit.
- Tensione di ingresso: da 0V a Vcc (Vcc=5.00V).
- Risoluzione: $\frac{Vfs}{2^n} = \frac{5}{1024} = 4,88mV$
-Tempo di conversione: 3,3µs
- Rapporto S/N = 61dB.

2.1.4 Trasmissione segnale

Il segnale acquisito deve essere trasmesso quando richiesto ad atri dispositivi: la comunicazione avviene tramite protocollo TCP/IP.
La bilancia all'avvio crea un server che rimane in ascolto per le connessioni provenienti dai client, tutte le richieste vengono accettate se formulate nel modo corretto, che prevede la seguente stringa :"rPes:a;" dove rPes indica la richiesta del peso e "a" indica il primo canale dei sei canali di acquisizione disponibili (il microcontrollore può acquisire fino 6 segnali analogici contemporaneamente).

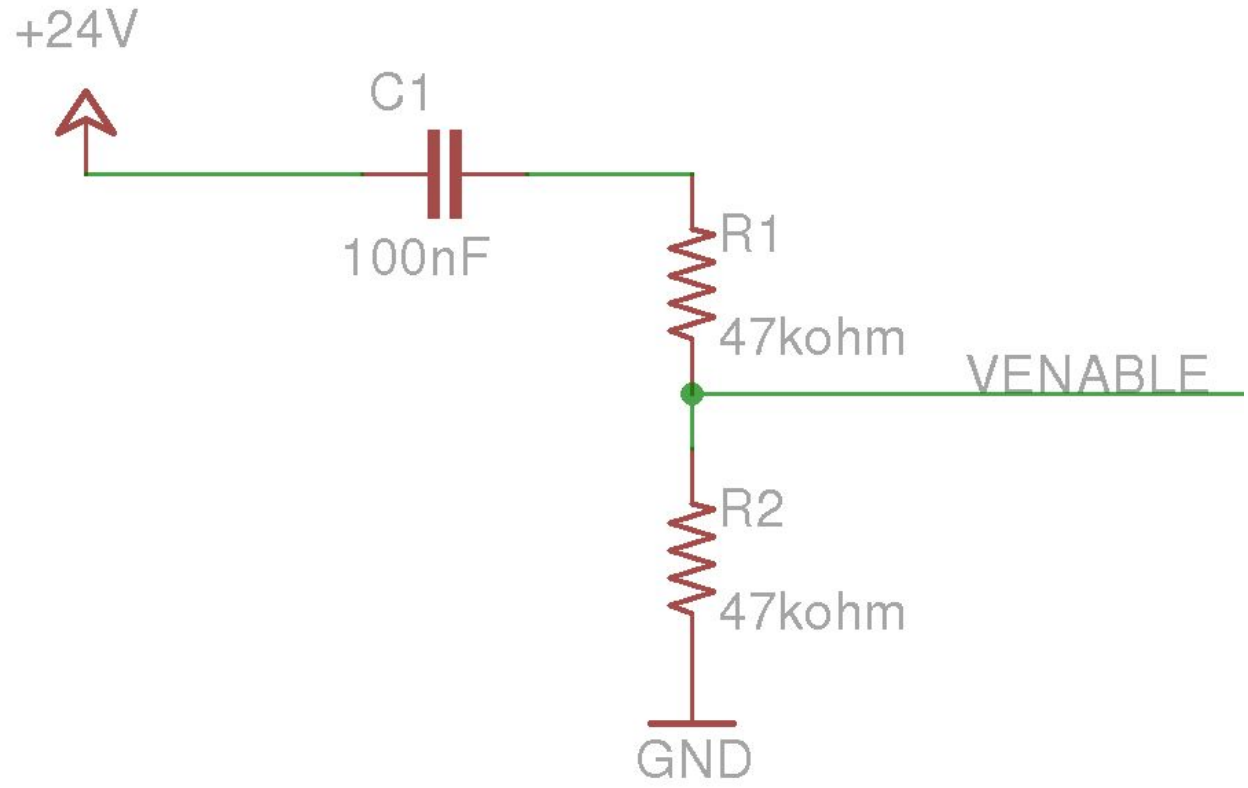
2.1.5 Considerazioni generali.

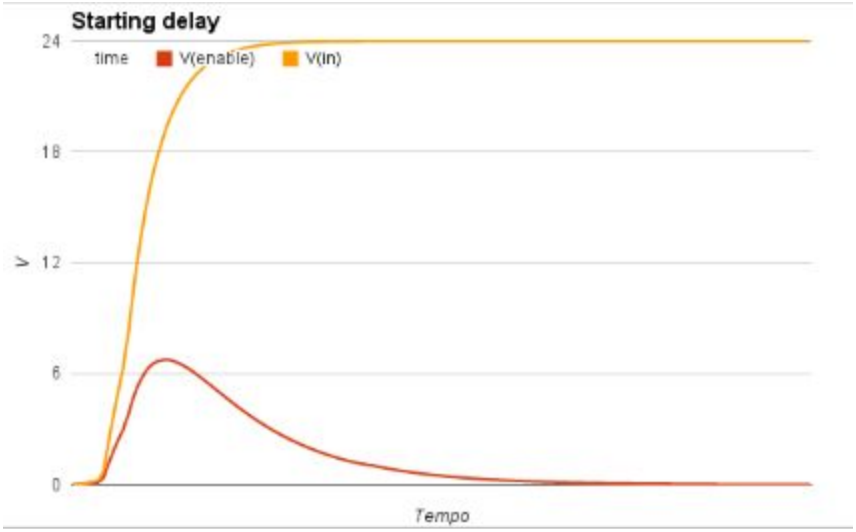
Per ottenere un buon risultato e limitare i disturbi nel circuito è stato necessario adottare alcuni accorgimenti:

1. Condensatori di disacoppiamento su tutte le alimentazioni dei circuiti integrati.
2. Condensatori di filtro su tutte le alimentazioni.
3. Circuito di ritardo per l'avvio degli alimentatori.
4. Fusibili autoripristinanti (termistori PTC).
5. LED che indicano la presenza delle alimentazioni.
6. Tensione di riferimento esterna per la conversione AD.

Nel dettaglio si possono fare le seguenti considerazioni:

1) Vicino ad ogni circuito integrato sono stati posti uno o più condensatori ceramici da 100nF-16V per eliminare i disturbi ad alta frequenza provenienti dall'esterno o generati internamente al circuito dagli alimentatori switching IC5 e IC6 e dal quarzo Q1.
2) Su tutte le fasi di alimentazione sono posti due condensatori in parallelo, uno elettrolitico da 680µF-20V per filtrare i disturbi a bassa frequenza ed uno ceramico da 100nF-16V per i disturbi ad alta frequenza.
3) Gli integrati che gestiscono le alimentazioni sono provvisti di un pin che abilita il funzionamento. Tramite un circuito composto da un condensatore e due resistenze è stato creato un ritardo all'accensione, necessario per permettere ai condensatori di caricarsi prima di abilitare il funzionamento. Il circuito è mostrato in figura.





Nel grafico si nota come la Venable scenda sotto i 3,5V necessari all' abilitazione dopo che la tensione di ingresso si stabilizza a 24V

4) Sono stati inseriti 3 termistori PTC utilizzati come fusibili autoripristinanti nel caso si verificassero corto circuiti o sovraccarichi, con le seguenti caratteristiche:

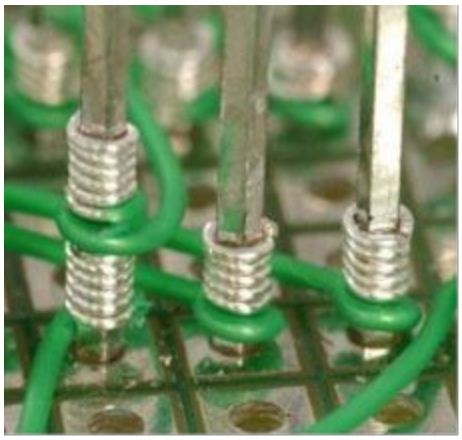
- Ih = 3.0A è la corrente massima sopportata dal fusibile prima di interrompere il circuito
- Vmax = 60V è la massima tensione applicabile al fusibile.
- Rmax = 0.285 Ω è la massima resistenza serie quando il dispositivo non è interdetto.
- Rs = 40kΩ è la resistenza che il PTC presenta quando è in stato di interdizione.

5) Sulla scheda sono stati inseriti 4 led per verificare la presenza delle alimentazioni, i led sono uno rosso per i +24V, due verdi per i ± 12V e uno giallo per i +5V.
6) Per effettuare una conversione analogico digitale nelle migliori condizioni possibili, è stato inserito un generatore di tensione di riferimento REF02 che genera 5,00V collegato al pin AREF del µC.

2.2 Circuito stampato.

Il circuito stampato è allegato con il nome “cella_carico_brd.pdf”

Una volta terminato lo schema elettrico è stato realizzato un primo prototipo del circuito con la tecnica del wire wrapping che permette di realizzare circuiti di prova facilmente modificabili.

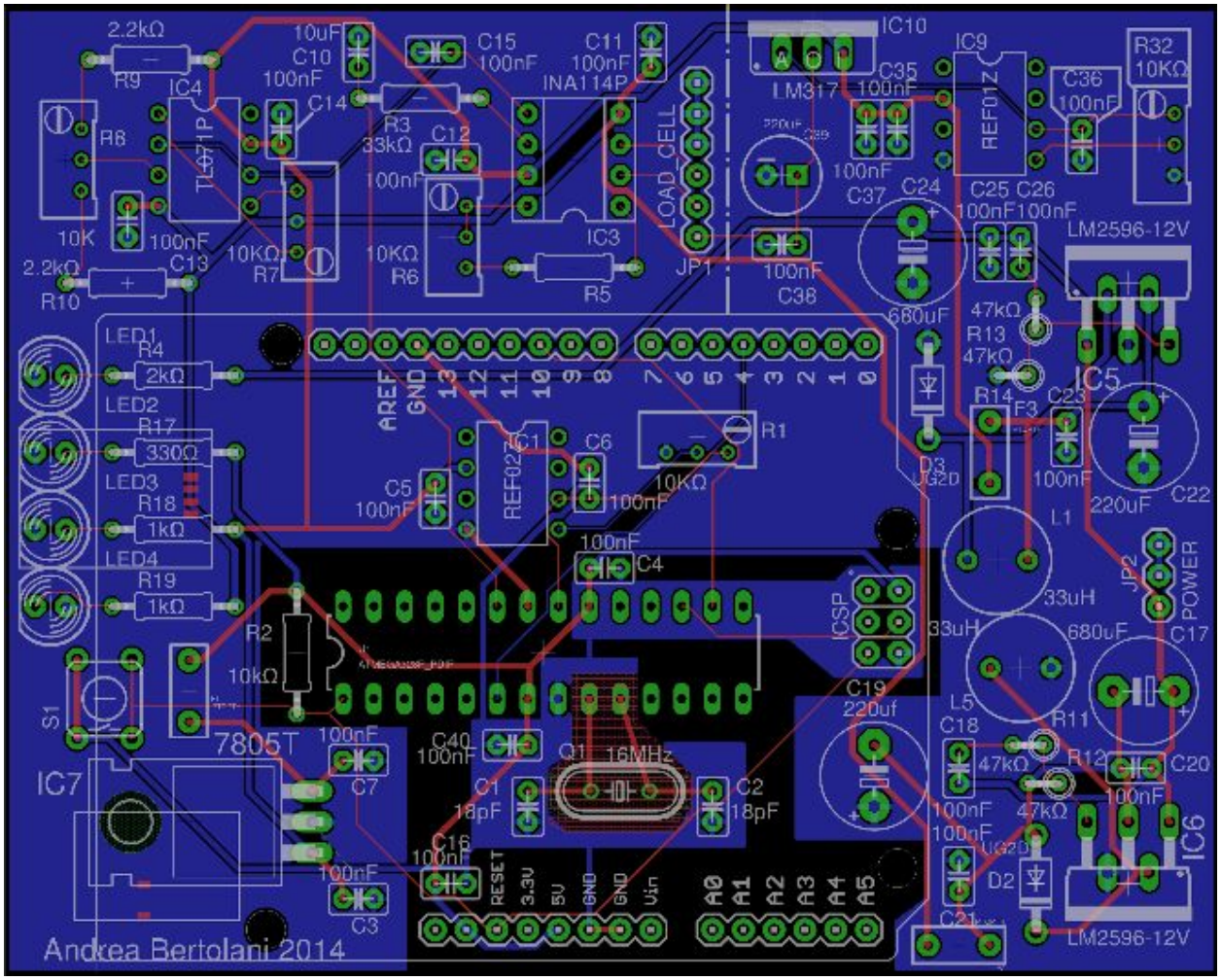


In figura si mette in evidenza come devono essere realizzati gli avvolgimenti attorno a ogni pin,con la tecnica del wire wrap: uno o più su ogni pin(con un massimo di 3/4), con 6 avvolgimenti per ogni collegamento.

Una volta corretti gli errori riscontrati e realizzato un nuovo prototipo è stato progettato il circuito stampato.
Per creare il PCB sono stati utilizzati alcuni accorgimenti indicati nelle application note delle grandi casi produttrici di circuiti integrati come Analog Devices e Texas Instruments.
Le soluzioni adottate sono state:

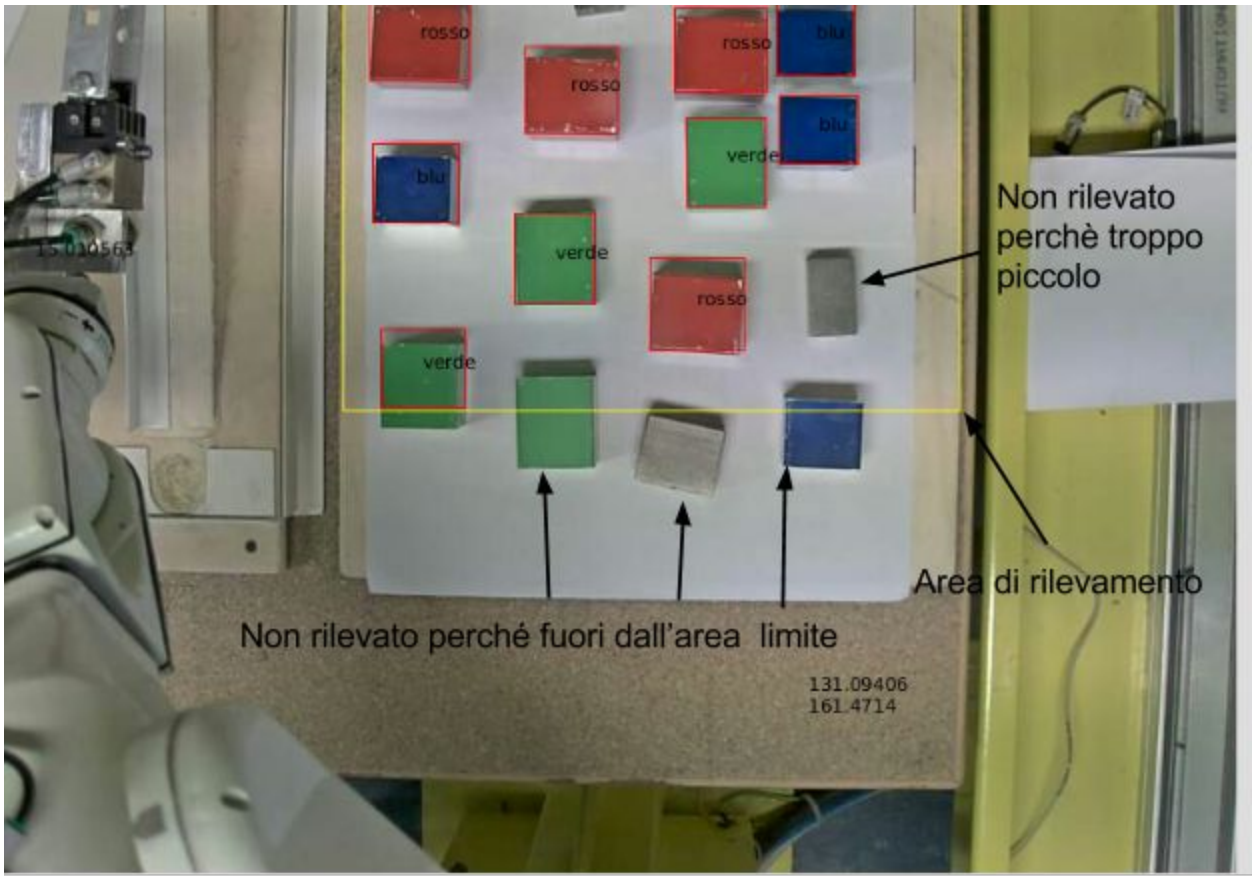
- Evitare che le piste formassero angoli di 90° per non introdurre induttanze parassite che avrebbero creato attenuazioni nei segnali.
- La larghezza di ogni pista è stata studiata in base alla corrente che deve scorrere su essa, ciò per evitare surriscaldamenti che avrebbero disturbato il sistema.
- Sono stati aggiunti tre piani di massa nella faccia inferiore del circuito: uno è stato utilizzato per limitare i disturbi dell'oscillatore al quarzo, un secondo piano è stato usato sotto le piste che collegano il microcontrollore alla scheda di rete, un ultimo piano di massa è stato inserito sotto la parte restante del circuito per limitare i disturbi residui.

Immagine del circuito stampato



3. Server di visione.

Il sistema per rilevare gli oggetti si affida ad un computer connesso in rete che acquisisce le immagini dalla videocamera e le elabora ottenendo le coordinate del centro ed il colore per poi trasmettere i dati al robot.



3.1a Immagine di prova del programma di visione artificiale

3.1. Il programma è stato realizzato in java con l'IDE Processing che mette a disposizione vare funzioni per gestire le immagini in modo semplificato, il programma crea inoltre un server TCP/IP che gestisce le richieste di posizione e, se necessario, recupera i dati relativi al peso degli oggetti e li salva in file esterni.

3.2. Parametri di rilevamento

Il programma di visione possiede alcuni parametri per la selezione degli oggetti, alcuni di essi sono facilmente visibili nell'immagine 3.1a

1) l_min, w_min
2) l_max, w_max
3) cut_x, cut_y
4) cut_w, cut_h

l_min e w_min indicano la dimensione minima che l'oggetto deve avere, vengono così esclusi oggetti troppo piccoli.
l_max e w_max servono per limitare la dimensione dell'oggetto, limitando la dimensione massima e minima, vengono così esclusi numerosi falsi rilevamenti che potrebbero compromettere la funzionalità del sistema.

cut_x, cut_y, cut_w e cut_h servono a circoscrivere un'area in cui il programma rileverà gli oggetti; nell'immagine 3.1a l'area di rilevamento è indicata in giallo.

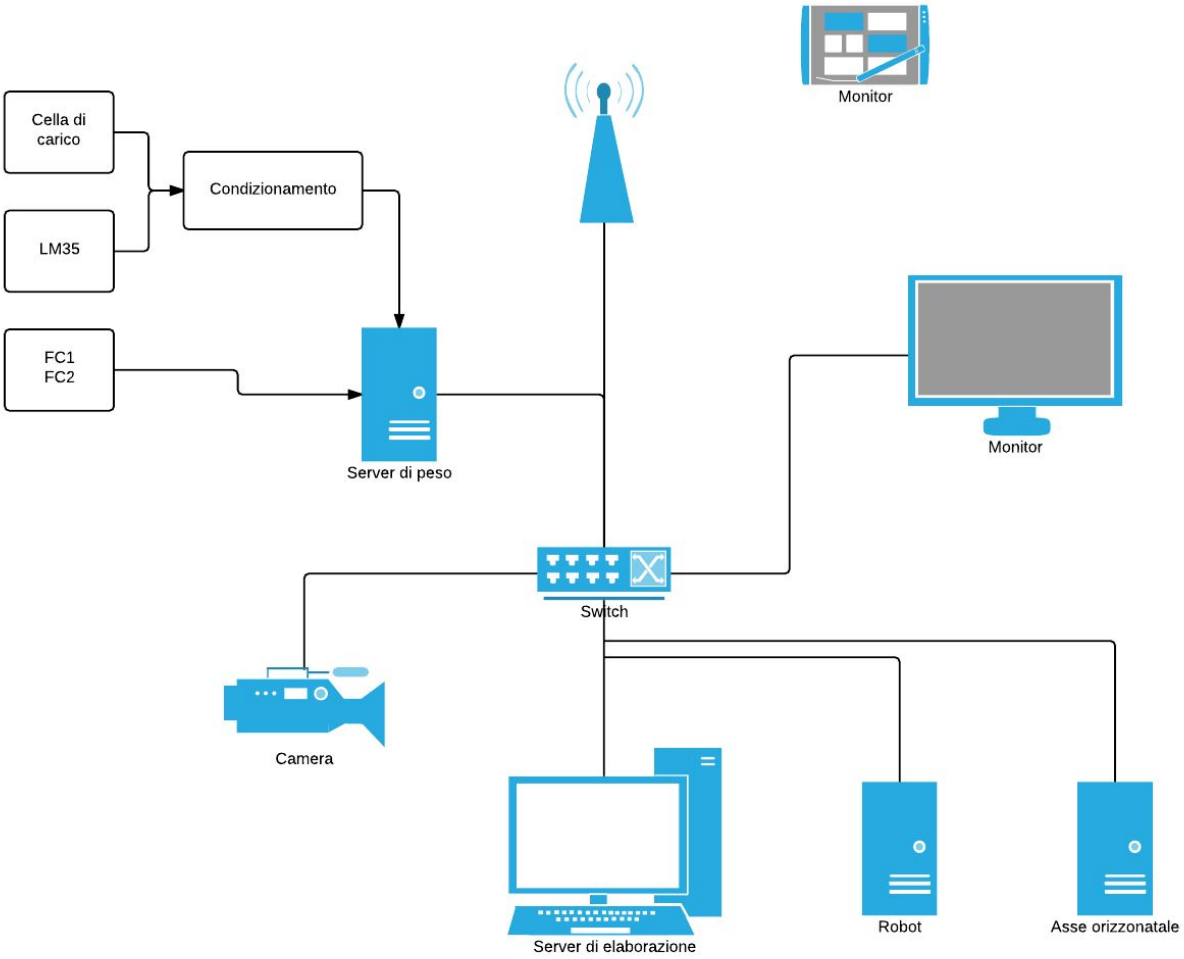
3.3. Trasformazione coordinate.

Tramite il programma di visione si ottiene la posizione dell'oggetto riferita alla videocamera, avente come unità di misura il pixel; è necessario quindi trasformare le coordinate in mm e riferirlo al centro del robot.
Per fare ciò è stata utilizzata una funzione matematica integrata in Processing chiamata map.

```
long map(long x, long in_min, long in_max, long out_min, long out_max){
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

3.4. Trasmissione coordinate.

Come già detto in precedenza il programma crea un server TCP/IP che risponde alle richieste ricevute tramite una stringa contenente la posizione e il colore del pezzo, un esempio di risposta è il seguente: "127.215,214.731,b;" la stringa indica un oggetto in posizione x di 127,215mm, posizione y di 214,731mm e di colore blu.

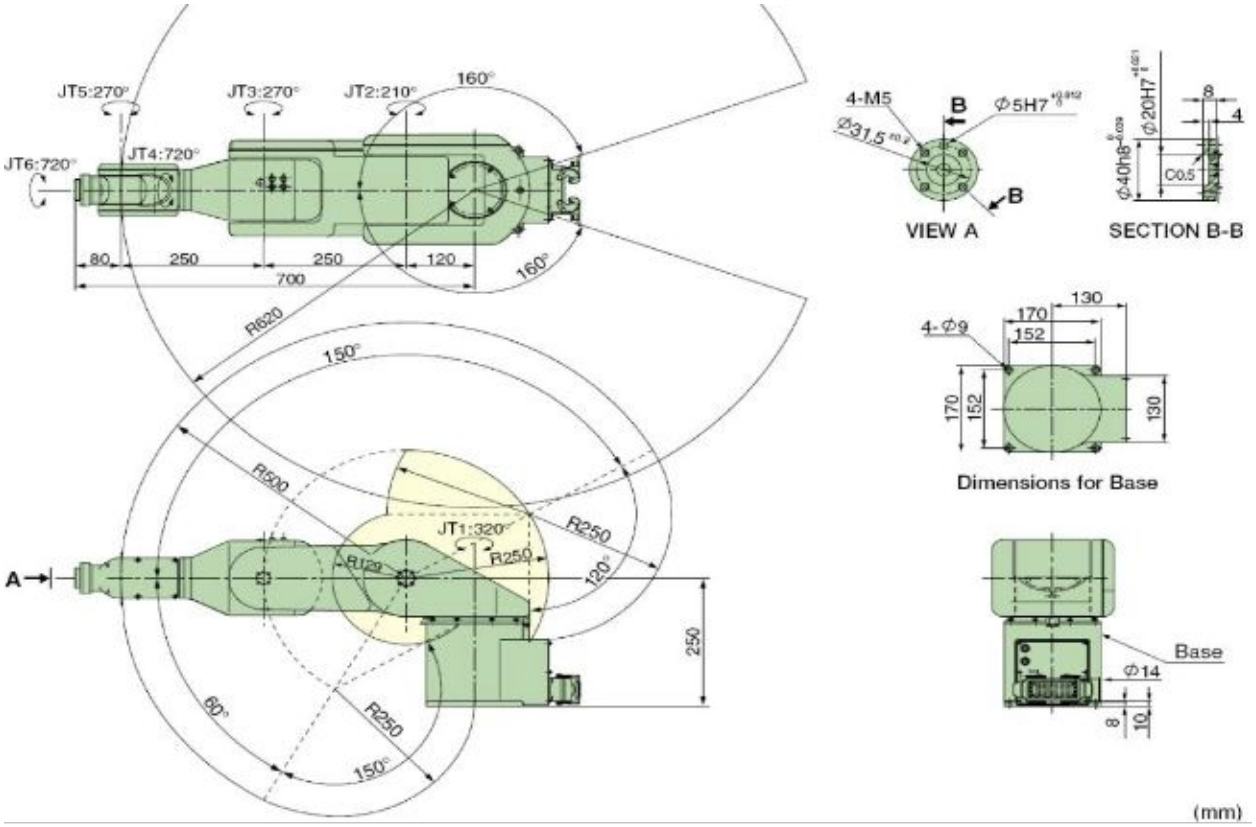


Dall’immagine si nota come tutti i dispositivi siano connessi allo stesso switch di rete e quindi possano scambiarsi le informazioni tra essi.

4. Robot

4.1 Caratteristiche

Il kawasaki FS03N è un robot antropomorfo da 6 gradi libertà, ciò comporta la possibilità di raggiungere qualsiasi oggetto in qualsiasi configurazione nel raggio di azione, le sue caratteristiche sono le seguenti:
- Alimentazione: 230V.
- Potenza assorbita: 2kW.
- Carico utile: 3 kg.
- Raggio d'azione: 620 mm.
- Motori: brushless.
- Coppia meccanica: JT4 5,8 Nm; JT5 5,8 Nm; JT6 2,9 Nm.
- Numero di I/O: 24
- Tensione degli I/O: 24V



L'immagine mostra il raggio d'azione del robot, cioè le posizione che il polso può raggiungere.

L'immagine mostra i 6 assi del robot con la corrispondenza al corpo umano

Immagine della postazione di lavoro con il robot.

Immagine del Teach Pendant, è l'interfaccia fisica tra il robot e l'operatore, e permette di controllare manualmente il movimento, gli I/O e controllare la posizione dei vari giunti.

4.2 Linguaggio AS

Il robot è programmato tramite il linguaggio proprietario della Kawasaki, chiamato AS. Il linguaggio è molto simile al BASIC ma possiede istruzioni specifiche per il controllo dei robot quali:
- JMOVE posizione : muove il robot in una posizione salvata in memoria (acquisita in precedenza o calcolata) tramite un movimento dalla traiettoria ad arco.
- LMOVE posizione : molto simile al JMOVE ma utilizza movimenti di tipo lineare.
- JAPPRO e LAPPRO posizione, distanza : muovono il robot vicino alla posizione di arrivo mantenendosi ad una distanza indicata.
- SIGNAL valore : modifica lo stato di un I/O, se il valore è positivo il pin viene settato a livello alto, se il valore è negativo viene settato a livello basso.

5. Programmi

Il sistema è composto da vari dispositivi sui quali vengono eseguiti diversi programmi in diversi linguaggi:

- Sulla cella di carico è presente un programma in C++ che converte il segnale e lo trasmette ai dispositivi che lo richiedono.
- Sul server di visione viene eseguito un programma in Java che controlla la posizione degli oggetti e la trasmette.
- Sul controller del robot viene eseguito un programma in AS (linguaggio proprietario Kawasaki)
che gestisce il movimento del robot e recupera le varie informazioni necessarie.

Programma di visione:

```
import controlP5.*;

//Andrea Bertolani
//server di visione
//2015

import processing.net.*; //([{<**>}])//
import blobDetection.*;

Server cubi;
Client peso;

int cubi_port = 9752; //porta dove vengono ricevute le richieste
int peso_port = 9789;//client che richiede e riceve il peso

final int letture_max = 20;

int[] numbers = new int[3];

public class punto { //classe che definisce un punto nello spazio bidimensionale(x,y)
  public float x;
  public float y;
}

public class rettangolo {//classe che definisce un rettangolo e ne conserva i parametri più utili cone il punto centrale
  public float w;
  public float h;
```

```
  public punto min = new punto();
  public punto max = new punto();
  public punto centro = new punto();
}

public class posa { //classe che definisce una punto nello spazio tridimensionale
  public float x;
  public float y;
  public float z;
  public float rox;
  public float roy;
  public float roz;
}

BlobDetection rilevatoreCubi;

PImage webImg;//oggetto che contiene l'immagine  completa
PImage webImg_e;//oggetto che contiene l'immagine tagliata

//webImg.pixels[][] è una matrice che contiene i pixel dell'immagine

//la stringa url contiene il percorso dell'immagine da elaborare, può essere locale o remoto(tramite connessione di rete)

//String url = "http://192.168.1.6:8090/photo.jpg";
//String url = "/home/andrea/esame/data/test8.jpg";
String url = "http://192.168.0.90/jpg/1/image.jpg";//percorso dell'immagine della videocamera di rete
//String url = "/home/andrea/Scaricati/image (14).jpg";//percorso dell'immagine locale(file salvato su disco)
//http://192.168.0.90/view/snapshot.shtml?picturepath=/jpg/1/image.jpg&streamprofile=Balanced&timestamp=1418472493542&width=1280

final int l_min = 40, w_min = 40;//dimensione minima del cubo per essere riconosciuto
final int l_max = 100, w_max = 100;//dimensione massima del cubo

int scalefact = 800;//ridimensionamento dell'immagine acquisita (non più utilizzato)

final int cut_x = 220, cut_y = 0;//area di riconoscimento degli oggetti
final int cut_w = 400, cut_h = 300;//come sopra

rettangolo posizione;

punto pos_pix, pos_mm;
punto pos_mem = new punto();

punto reale1, processing1;//punti necessari ad effettuare la traslazione delle coordinate da pixel riferiti alla videocamera a millimetri riferiti al robot
punto reale2, processing2;//come sopra
```

```
boolean first = true;//variabile che indica se è stato rilevato almeno un oggetto
Blob first_b;//primo oggetto rilevato dal programma

final int cont_col = 50;//variabile che indica quanti punti vengono utilizzati per effettuare la media e ottenere un colore da un punto(vedi funzione get_color)

ControlP5 cp5;
void setup() {
  //inizializzazione variabili
  posizione = new rettangolo();

  //smooth();
  cp5 = new ControlP5(this);

  pos_mm = new punto();
  pos_pix = new punto();

  reale1 = new punto();
  processing1 = new punto();

  reale2 = new punto();
  processing2 = new punto();

  //18 MARZO t:9/10
  reale1.x = -41.37f;
  reale1.y = 699.66f;

  reale2.x = 147.05f;
  reale2.y = 493.99f;

  processing1.x = 234.0f;
  processing1.y = 6.0f;

  processing2.x = 461.0f;
  processing2.y = 240.0f;

  //FINE 18 MARZO

  upimage();

  rilevatoreCubi = new BlobDetection(webImg_e.width, webImg_e.height);
  //rilevatoreCubi = new BlobDetection(400, 400);
  rilevatoreCubi.setPosDiscrimination(false);
   rilevatoreCubi.setThreshold(0.48f);//0.48f//soglia di riconoscimento, è un numero float che può variare da 0 a 1
  rilevatoreCubi.setConstants(1000, 4000, 1000);
  rilevatoreCubi.computeBlobs(webImg_e.pixels);
```

```
  cubi = new Server(this, cubi_port);//avvio il server tcp

  cp5.addToggle("toggle")
    .setPosition(80, 640)
      .setSize(50, 20)
        .setValue(true)
          .setMode(ControlP5.SWITCH)
            ;


  size(webImg.width, webImg.height+100);//imposto le dimensioni della finestra come quelle dell'immagine
}

int min = 0, max = 400, maxd = 360;

long t1 = millis();
int tcc = 100;

String data_in;

boolean canReq = false;

void draw() {//ciclo infinito
  background(255);//pulisco lo schermo
  upimage();//aggiorno l'immagine

    rilevatoreCubi.computeBlobs(webImg_e.pixels);//eseguo l'algoritmo di rilevamento oggetti
  first = true;

  draw_GUI_debug();


  if (canReq) {
    if ((millis()-t1) > tcc) {
      //println("aggiornamento peso");
      t1 = millis();
      peso.write("a");
    }

    if (peso.available()>0) {

      data_in = peso.readStringUntil(';');
      println(data_in);
    }
  }

  Client cubi_client = cubi.available();//controllo se un client è connesso
```

```
  if (cubi_client != null) {//se il client esiste
    //println("nuovo client"+cubi_client.ip());//scrivo l'ip del client
    String req = cubi_client.readString();//leggo ciò che il client scrive
    //println(req);//scrivo la richiesta
    if (req.indexOf("posizione") >= 0) {//se è presente la stringa posizione

      if (first) {//se non è rilevato alcun cubo trasmetto la stringa null
        cubi_client.write("0,0,0");
      } else {//altrimenti trasmetto le coordinate del primo cubo
        punto send_cord = new punto();
                                  //println("x    =    "+convert_cord_mm(convert_b_pos_ret(first_b).centro).x+"    y    =
"+convert_cord_mm(convert_b_pos_ret(first_b).centro).y);
          send_cord.x = convert_cord_mm(convert_b_pos_ret(first_b,false).centro).x;//converto la x da pixel a
millimetri
         send_cord.y = convert_cord_mm(convert_b_pos_ret(first_b,false).centro).y;//converto la y da pixel da
millimetri
        cubi_client.write(send_cord.x+","+send_cord.y+","+get_color(posizione.centro)+";\n");//scrivo al client le
coordinate del primo cubo e il suo colore separati da una virgola e terminati da un punto e virgola
         println(send_cord.x+","+send_cord.y+","+get_color(posizione.centro)+";\n");//per debug scrivo ciò che
ho appena trasmesso
      }
    }
  }
}


float correct_x(float x) {
    return 0.116618100279861 + 0.993602458001236*x + 3.41691186534614*cos(2.78254562094655 +
1.99892112017856*x) - 0.590970384639418*cos(x);
   //return 0.994326275489225*x + 3.07090547904805*cos(2.65057423566345 + 1.99963013196625*x) +
0.839662594687821*sin(2.00002145514397*x*x);
}

float correct_y(float x) {
   return 19.4379317559359 + 0.964779039143149*x + 0.0153368308935755*cos(1.00026176894146*x) +
1.80316818252241*cos(0.0124993075837112 + 6.15074354183296*x);
}


int col = color(255);

boolean toggleValue = false;

void toggle(boolean theFlag) {
  if (theFlag==false) {//on
    connect_to_cc();
  } else {//off
```

```
    disconnect_from_cc();
  }
  println("a toggle event.");
}


void disconnect_from_cc() {
  if (canReq) {
    peso.stop();
    canReq = false;
  }
}



void connect_to_cc() {
  if (!canReq) {
    println("prima del misfatto");
    peso = new Client(this, "192.168.0.51", 9860);
    canReq = true;
    println("connessione alla bilancia");
  }
}

void keyPressed() {
  pos_mem.x = mouseX;
  pos_mem.y = mouseY;
  println(pos_mem.x+"\t"+pos_mem.y);
}


void draw_GUI_standard() {
  drawBlobsAndEdges(true, false,true);//disegno gli oggetti trovati
  background(255);

  textSize(40);
                                                text(""+convert_cord_mm(convert_b_pos_ret(first_b,true).centro).x+",
"+convert_cord_mm(convert_b_pos_ret(first_b,true).centro).y, 220, 650);
  textSize(12);


  textSize(60);
  if (data_in != null)
    text(convertGr(Integer.parseInt(data_in.replaceFirst(";", ""))), 50, height -25);
  textSize(12);

  text("Abilitazione lettura peso", 40, 625);
```

```
  punto mcord = new punto();//punto che contiene le coordinate del mouse espresse in pixel

  mcord = get_cord(mouseX, mouseY);

  drawCords(mcord);//convert_pix_mm(mcord));//scrivo vicino al puntatore del mouse le sue coordinate
  //drawCords(convert_cord_mm(mcord));//convert_pix_mm(mcord));

  stroke(0);
  line(70, 0, 70, height-200);
  line(140, 0, 140, height-200);
  line(210, 0, 210, height-200);
  line(280, 0, 280, height-200);

  fill(color(0, 0, 255));
  text("BLU", 95, 20);

  fill(color(255, 0, 0));
  text("ROSSO", 150, 20);

  fill(color(0, 255, 0));
  text("VERDE", 220, 20);
}


void draw_GUI_debug() {
  image(webImg, 0, 0, width, height-100);//disegno l'immagine sullo schermo

  drawBlobsAndEdges(true, false, true);//disegno gli oggetti trovati

  fill(0, 0, 0, 0);
  stroke(255, 255, 0);
  rect(cut_x, cut_y, cut_w, cut_h);//disegno un rettangolo giallo in corrispondenza dell'area di rilevamento
  fill(255, 255, 0);

  if (!first)//se è stato trovato almeno un oggetto allora lo disegno con il contorno verde
    draw_b_stroke(first_b, color(0, 255, 0));//

  fill(0);//colore nero
  text(""+frameRate, 20, 200);//scrivo a schermo il framerate(utile per il debug)
  punto mcord = new punto();//punto che contiene le coordinate del mouse espresse in pixel

  mcord = get_cord(mouseX, mouseY);

  drawCords(mcord);//convert_pix_mm(mcord));//scrivo vicino al puntatore del mouse le sue coordinate
  //drawCords(convert_cord_mm(mcord));//convert_pix_mm(mcord));

  //println(get_color(mcord));//scrive il colore su cui si trova il mouse
```

```
  textSize(60);
  if (data_in != null)
    text(convertGr(Integer.parseInt(data_in.replaceFirst(";", ""))), 150, height -25);
  textSize(12);

  text("Abilitazione lettura peso", 40, 625);

  textSize(40);
   if(!first)
                                                    text(""+convert_cord_mm(convert_b_pos_ret(first_b,true).centro).x+",
"+convert_cord_mm(convert_b_pos_ret(first_b,true).centro).y,220,650);
   textSize(12);
}

float convertGr(int data) {
  return data*2000/1024;
}

char get_color(punto pos) {//la funzione restituisce il colore(r,g,b) del punto passato come argomento tramite
una media di più punti
  long r=0, g=0, b=0;
  char ret = 'n';

  for (int i = 0; i < cont_col; i++) {
    int x = int(random(-10, 10));
    int y = int(random(-10, 10));

    color get_c = get(int(pos.x + x), int(pos.y + y));

    r += red(get_c);
    g += green(get_c);
    b += blue(get_c);
  }

  r /= cont_col;
  g /= cont_col;
  b /= cont_col;

  if (max(r, g, b) == r)
    ret = 'r';
  else if (max(r, g, b) == g)
    ret = 'g';
  else if (max(r, g, b) == b)
    ret = 'b';

  return ret;
}
```

```
void upimage() {//aggiorno l'immagine recuperandola da url

  webImg = loadImage(url, "jpg");

  //webImg.resize(scalefact,0); //ridimensionameto immagine
  //webImg = webImg.get(cut_x, cut_y, cut_w, cut_h);
  //webImg.filter(GRAY); //filtro in scala di grigi per l'immagine
   webImg_e = webImg.get(cut_x, cut_y, cut_w, cut_h);//porzione di immagine dove deve essere effettuato il
riconoscimento
}

void drawCords(punto cord) {//scrivo le coordinate passate come argomento
  text(cord.x+"\n"+cord.y, mouseX+10, mouseY+10);
}

punto convert_pix_mm(punto pos2) {//converto le coordinate in millimetri(non più utilizzata)

  punto convert = new punto();

  convert.x = map(pos2.x, 225.0f, 354.0f, -89.085f, 8.893f);
  convert.y = map(pos2.y, 1.0f, 131.0f, 604.703f, 504.460f);
  //convert.x = map(pos.x,225.0f,354.0f,-89.085f,8.893f);

  return convert;
}

punto convert_cord_mm(punto pos2) {//converte le dimensioni indicate da pixel a millimetri

  punto convert = new punto();

  convert.x = map(pos2.x, processing1.x, processing2.x, reale1.x, reale2.x);
  convert.y = map(pos2.y, processing1.y, processing2.y, reale1.y, reale2.y);

  //funzione map:
  /*long map(long x, long in_min, long in_max, long out_min, long out_max)
   {
   return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
   }*/

  return convert;
}

// ================================================
// drawBlobsAndEdges()
// ================================================
void drawBlobsAndEdges(boolean drawBlobs, boolean drawEdges, boolean correct)
{
```

```
noFill();
Blob b;
EdgeVertex eA, eB;
for (int n=0; n<rilevatoreCubi.getBlobNb (); n++)
{
  b=rilevatoreCubi.getBlob(n);

  if (b!=null)
  {

    // Edges
    if (drawEdges)
    {
      strokeWeight(2);
      stroke(0, 255, 0);
      for (int m=0; m<b.getEdgeNb (); m++)
      {
        eA = b.getEdgeVertexA(m);
        eB = b.getEdgeVertexB(m);
        if (eA !=null && eB !=null)
          line(
          eA.x*width, eA.y*height,
          eB.x*width, eB.y*height
            );
      }
    }

    // Blobs
    if (drawBlobs) {
      if ((b.w*webImg_e.width < w_max) && (b.h*webImg_e.height < l_max)) {
        if ((b.w*webImg_e.width > w_min) && (b.h*webImg_e.height > l_min)) {

          b.xMin = b.xMin + (float(cut_x)/webImg_e.width);
          //draw_b(resize_b(shift_b(b, p_shift),p_s_1,p_s_2,p_s_3,p_s_4));
          draw_b(b);

          //shift
          //resize

          posizione.h = b.h*webImg_e.height;
          posizione.w = b.w*webImg_e.width;
          posizione.min.x = b.xMin*webImg_e.width;
          posizione.min.y = b.yMin*webImg_e.height;
          posizione.max.x =  posizione.min.x + posizione.w;
          posizione.max.y =  posizione.min.y + posizione.h;
          posizione.centro.x =  posizione.min.x + (posizione.w / 2);
          posizione.centro.y =  posizione.min.y + (posizione.h / 2);
```

```
        fill(0, 0, 0);
        String text_t = null;
        //switch(get_color(get_cord(int(posizione.centro.x),int(posizione.centro.y)))){
        switch(get_color(posizione.centro)) {
        case 'r':
          text_t = "rosso";
          break;
        case 'g':
          text_t = "verde";
          break;
        case 'b':
          text_t = "blu";
          break;
        }
        text(text_t, posizione.centro.x, posizione.centro.y);

        //line(posizione.centro.x, posizione.centro.y, 0, 0);


//text(posizione.w+"\n"+posizione.h+"\n"+posizione.min.x+"\n"+posizione.min.y,posizione.min.x,posizione.mi
n.y);

        if (first) {
          first_b = b;
          //draw_b(shift_b(resize_b(first_b,p_s_1,p_s_2,p_s_3,p_s_4), p_shift));
          //draw_b(resize_b(first_b,p_s_1,p_s_2,p_s_3,p_s_4));

          first = false;
        }
        /*punto send_cord = new punto();
                              //println("x  =  "+convert_cord_mm(convert_b_pos_ret(first_b).centro).x+"  y  =
"+convert_cord_mm(convert_b_pos_ret(first_b).centro).y);
             send_cord.x = convert_cord_mm(convert_b_pos_ret(first_b).centro).x;//converto la x da pixel a
millimetri
            send_cord.y = convert_cord_mm(convert_b_pos_ret(first_b).centro).y;//converto la y da pixel da
millimetri

        println(send_cord.x+","+correct_x(send_cord.x)+","+get_color(posizione.centro)+";\n");*/


       }
      }
     }
    }
  }
}
```

```
rettangolo convert_b_pos_ret(Blob b,boolean correct) {//trasforma da variabile blob a rettangolo
  rettangolo ret = new rettangolo();

  ret.h = b.h*webImg_e.height;
  ret.w = b.w*webImg_e.width;
  ret.min.x = b.xMin*webImg_e.width;
  ret.min.y = b.yMin*webImg_e.height;
  ret.max.x =  ret.min.x + ret.w;
  ret.max.y =  ret.min.y + ret.h;
  ret.centro.x =  ret.min.x + (ret.w / 2);
  ret.centro.y =  ret.min.y + (ret.h / 2);

  if(correct){
    ret.centro.x = correct_x(ret.centro.x);
    ret.centro.y = correct_y(ret.centro.y);
  }

  return ret;
}

void draw_b(Blob b) {//disegna l'oggetto passato come funzione
  strokeWeight(1);
  stroke(255, 0, 0);
  fill(0, 0, 0, 0);
  rect(b.xMin*webImg_e.width, b.yMin*webImg_e.height, b.w*webImg_e.width, b.h*webImg_e.height);
}

void draw_b_stroke(Blob b, color c) { //disegna l'oggetto passato come argomento con un colore definito da
c
  strokeWeight(1);
  stroke(c);
  fill(0, 0, 0, 0);
  rect(b.xMin*webImg_e.width, b.yMin*webImg_e.height, b.w*webImg_e.width, b.h*webImg_e.height);
}

void draw_b_full(Blob b, color c) {//disegna l'oggetto con un colore pieno
  strokeWeight(1);
  stroke(255, 0, 0);
  fill(c);
  rect(b.xMin*webImg_e.width, b.yMin*webImg_e.height, b.w*webImg_e.width, b.h*webImg_e.height);
}

Blob shift_b(Blob b, punto p_s) {//trasla la posizione dell'oggetto di p_s.x e p_s.y
  Blob b_ret;

  b_ret = b;
  b_ret.xMin = b.xMin*width + p_s.x;
```

```
  b_ret.xMin /= width;
  b_ret.yMin = b.yMin*height + p_s.y;
  b_ret.yMin /= height;

  return b_ret;
}

punto get_cord(int x, int y) {//trasforma due variabili intere x e y in una variabile coordinata
  punto ret = new punto();
  ret.x = x;
  ret.y = y;
  return ret;
}
```

Programma di gestione della cella di carico:

```
#include <Ethernet.h>
#include <SPI.h>

#include "definizioni.h"

#ifdef scuola
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0D, 0x15, 0x61 };
byte ip[] = { 192, 168, 0, 51 };
byte gateway[] = { 192, 168, 0, 50 };
byte subnet[] = { 255, 255, 255, 0 };
#endif

#ifdef casa
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0D, 0x15, 0x61 };
byte ip[] = { 192, 168, 1, 120 };
byte gateway[] = { 192, 168, 1, 1 };
byte subnet[] = { 255, 255, 255, 0 };
#endif

EthernetServer server = EthernetServer(9860);
EthernetClient client_cubi;

IPAddress ip_cubi(192, 168, 0, 50);

int cubi_port = 9752;

int selezione(char);

char response[BUFFER_SIZE];
String return_s;

long time_last_req = 0;

void setup()
{
  Serial.begin(115200);

  Ethernet.begin(mac, ip, gateway, subnet);
  server.begin();

  Serial.println("inizio");

  Serial.println(client_cubi.connect(ip_cubi, cubi_port));

}

void loop(){
```

```
  if(Serial.available()){
    Serial.println(return_s);
    client_cubi.write(Serial.read());//richiesta di dati dal server di visione(processing)
  }

  if((millis()-time_last_req) > delay_req_ms){
    client_cubi.write(abracadabra);
    //Serial.print("req");
    //Serial.print(return_s);
  }

  //int i = 0;
  if(client_cubi.available()) {
    return_s = "";
    while(client_cubi.available()){
      char c = client_cubi.read();
      return_s += c;
      //Serial.println(return_s);
      return_s.toCharArray(response,return_s.length());
    }
  }

  EthernetClient client = server.available();//richieste arrivate da client remoti(robot)
  if (client) {
    char sel = client.read();
    Serial.println(sel);
    if(selezione(sel) > -1){
      String ciao =  String(analogRead(selezione(sel)))+";";
      Serial.print(ciao.c_str());
      Serial.print("\t");
      Serial.println(selezione(sel));
      server.write(ciao.c_str());
    }
    else if(sel == pos_rq){
      Serial.print("richiesta posizione, risposta = ");
      Serial.println(response);

      server.write(response);
    }
  }
}

int selezione(char sl){
  int rt = -1;
  switch(sl){

  case 'a':
    rt = A0;
  break;
```

```
  case 'b':
    rt = A1;
  break;

  case 'c':
    rt = A2;
  break;

  case 'd':
    rt = A3;
  break;

  case 'e':
    rt = A4;
  break;

  case 'f':
    rt = A5;
  break;

  }
  return rt;
}

int analogReadMedia(char pin){
  int valMin = 0;
  int valMax = 0;

  long int readings = 0;

  for(int i = 0; i < num_readings; i++){
    int tmp = analogRead(pin);
    readings += tmp;

    if(tmp > valMax)
      valMax = tmp;
    if(tmp < valMin)
      valMin = tmp;
  }

  readings -= valMax - valMin;
  return readings/(num_readings-2);
}
```

File “definizioni.h”

```
#define scuola
#define num_readings 600
#define pos_rq 'p'
#define risposta_posizione "109.163,529.479,-245.002,139.995,179.401,-37.777"
//#define risposta_posizione "-64.590,615.989,-242.997,-160.123,0,153.925,-22.356"
```

```
#define BUFFER_SIZE 64

#define delay_req_ms 100
#define abracadabra "posizione"

#define peso_max 2000 //peso in grammi
#define vfs 5000 //mV
#define nbit 8 //numero bit

#define version "3_mag"
```

Programma di gestione del robot:

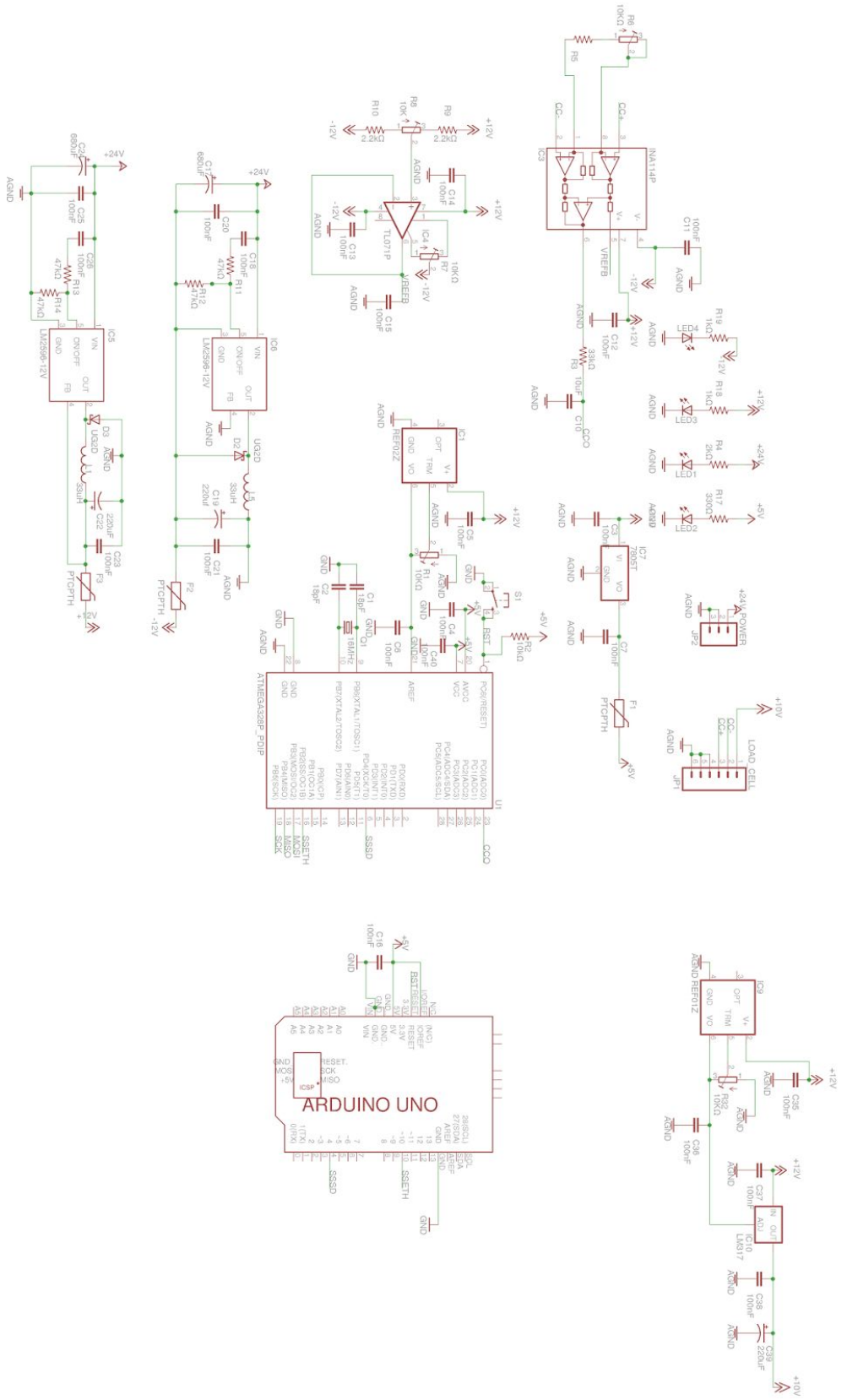
```
.PROGRAM bera_esame()
  CP ON
  TOOL toolventosa1
  SPEED 40 ALWAYS
  ABOVE
  RIGHTY
  DWRIST
  ACCURACY 70 ALWAYS
  ip[1] = 192
  ip[2] = 168
  ip[3] = 0
  ip[4] = 51
  port = 9860
  max_length = 255
  tout_open = 1
  tout_rec = 1
  eret = 0
  ret = 0
  h_min = 150
  h_min_2 = 50
  h_min_3 = 100
  POINT bera_depo_r_tem = bera_depo_r
  POINT bera_depo_g_tem = bera_depo_g
  POINT bera_depo_b_tem = bera_depo_b
  POINT bera_move_color = bera_depo_r_tem
  r_i = 0
  r_j = 0
  g_i = 0
  g_j = 0
  b_i = 0
  b_j = 0
  deltax = 65
  deltay = 65
  v3 = 70
  v2 = 20
  v1 = 10
  v_d = 1
  v_d_2 = 1
  JMOVE #bera_riposo
  CALL open_socket
  IF sock_id<0 THEN
          GOTO exit_end
  END
  text_id = 0
  tout = 3
```

```
  WHILE (1) DO
ricevi_dati:
        ret = 0
        CALL bera_rec_data ;ricezione dati da server TCP
        CALL decode_string ;provo a deodificare i dati ricevuti
        ;PRINT "************"
        ;PRINT flag
        ;PRINT "************"
        IF ret<0 THEN
          PRINT "Communication error code=",ret
          GOTO exit
        END
        TWAIT 0.2
        IF flag==1 THEN ;se non stato rilevato nessun oggetto aspetto 5 secondi e riprovo
          TWAIT 5
          GOTO ricevi_dati
          PRINT "ricevuto null"
        END
        IF flag==0 THEN ;se le informazioni sono state decodificate correttamente eseguo il ciclo
          SPEED v3
          LAPPRO bera_move,h_min ;avvicino la ventosa al pezzo a circa 10 cm di distanza
          SPEED v1
          LMOVE bera_move ;raggiungo il pezzo
          SIGNAL 17 ;eseguo il vuoto nella ventosa
          TWAIT 0.5 ;aspetto che la ventosa agganci il pezzo
          SPEED v3
          LDEPART h_min_3   ;allontano la ventosa di circa 10 cm
          SPEED v3
          JAPPRO bera_bilancia,h_min_2 ;avvicino la cella di carico
          SPEED v_d
          ACCURACY 1
          CP OFF
          LMOVE bera_bilancia ;raggiungo la cella di carico
          SIGNAL -17 ;sgancio la ventosa
          TWAIT 0.5 ; attendo che l'istruzione precedente abbia effetto
          ACCURACY 70 ALWAYS
          CP ON
          SPEED v3
          LDEPART 50 ;ritraggo la ventosa per effettuare la misurazione
          TWAIT 3 ;attendo 3 secondi per permettere al sistema di misurare il peso
          SPEED v3
          ACCURACY 1
          LMOVE bera_bilancia
          SIGNAL 17
          ACCURACY 70 ALWAYS
          CP ON
          SPEED v3
          LDEPART 150
```

```
; PRINT "."
;PRINT $colore
;PRINT "."
IF ($colore==",r") THEN
        POINT bera_move_color = bera_depo_r_tem
        IF (r_i<1) THEN
          r_i = r_i+1
        ELSE
          r_i = 0
          r_j = r_j+1
        END
        POINT bera_depo_r_tem = SHIFT(bera_depo_r BY -deltax*r_i,deltay*r_j)
END
IF ($colore==",g") THEN
        POINT bera_move_color = bera_depo_g_tem
        IF (g_i<1) THEN
          g_i = g_i+1
        ELSE
          g_i = 0
          g_j = g_j+1
        END
        POINT bera_depo_g_tem = SHIFT(bera_depo_g BY -deltax*g_i,deltay*g_j)
END
IF ($colore==",b") THEN
        POINT bera_move_color = bera_depo_b_tem
        IF (b_i<1) THEN
          b_i = b_i+1
        ELSE
          b_i = 0
          b_j = b_j+1
        END
        POINT bera_depo_b_tem = SHIFT(bera_depo_b BY -deltax*b_i,deltay*b_j)
END
SPEED v3
;JMOVE #bera_riposo
SPEED v3
JAPPRO bera_move_color,h_min_3
SPEED v_d
ACCURACY 1
CP OFF
LMOVE bera_move_color
SIGNAL -17
TWAIT 0.5
CP ON
ACCURACY 70 ALWAYS
SPEED v3
LDEPART 100
SPEED v3
```

```
            JMOVE #bera_riposo
          END
          IF eret<0 THEN
            GOTO exit
          END
  END
exit:
  CALL close_socket
exit_end:
  .END



.PROGRAM bera_rec_data()
  CALL com_test
  CALL recv
.END



.PROGRAM open_socket()
  .er_count = 0
connect:
  TCP_CONNECT sock_id,port,ip[1],tout_open
  IF sock_id<0 THEN
          IF .er_count>=5 THEN
            PRINT "Connection with PC failed. Program is stopped."
          ELSE
            .er_count = .er_count+1
            PRINT "TCP_CONNECT error id=",sock_id," error count=",.er_count
            GOTO connect
          END
  ELSE
          PRINT "TCP_CONNECT OK id=",sock_id
  END
.END



.PROGRAM decode_string()
  flag = 0
  $rx = $recv_buf[1]
  PRINT $rx
  IF ($recv_buf[1]=="niente") THEN
          GOTO fine_muovi
          flag = 1
          PRINT "nessun cubo trovato"
```

```
  END
  $temp = $DECODE($rx,",",0)
  x = VAL($temp)
  IF x<>0 THEN
          PRINT ("no")
          $temp = $DECODE($rx,",",1)
          $temp = $DECODE($rx,",",0)
          y = VAL($temp)
          $temp = $DECODE($rx,";",1)
          $temp = $DECODE($rx,";",0)
          $colore = $temp
          POINT bera_move = TRANS(x,y)
          POINT/OAT bera_move = bera_temp
          POINT/Z bera_move = -240
  ELSE
          flag = 1
          PRINT "nessun cubo trovato"
          GOTO fine_muovi
  END
fine_muovi:
.END



.PROGRAM recv()
  .num = 0
  .tout_rec = 2
  TCP_RECV ret,sock_id,$recv_buf[1],.num,tout_rec
  IF ret<0 THEN
          PRINT "TCP_RECV error in RECV",ret
          $recv_buf[1] = "000"
  ELSE
          IF .num>0 THEN
            PRINT "TCP_RECV OK"
            PRINT "8"
            PRINT $recv_buf[1]
            PRINT "8"
          ELSE
            $recv_buf[1] = "000"
          END
  END
.END
```


ATMEGA328P-PDIP
ARDUINO UNO
INA114P
TL071P
LM317
LOAD_CELL
POWER