Treball de Fi de Grau

Grau en Enginyeria Electrònica Industrial i Automàtica

CONTROL D'UNA PLATAFORMA ESTABILITZADORA AMB UN MICROCONTROLADOR I UNA CÀMERA WEB

Autor: **JORDI BUSTOS CUNILL**

Tutor: **ANTONI VILA MARTA**

Convocatòria: **SETEMBRE 2014**

AGRAÏMENTS:

A tothom qui ha fet possible aquest projecte, i de forma molt especial al tutor Antoni Vila Marta per la informació aportada, per trobar sempre un forat en la seva agenda i per la seva supervisió i guia del projecte.

Molt agraït també a Ramon Magem per la gran ajuda rebuda en el muntatge mecànic de la estructura i per posar a la meva disposició tots els medis.

Finalment a Roger Roig Pagès i Joan Rota Cunill per la seva ajuda, i a la meva família per tot el recolzament.

Títol: Control d’una plataforma estabilitzadora amb un microcontrolador i una càmera web

Autor: Jordi Bustos Cunill
Tutor: Antoni Vila Marta

RESUM

Aquest projecte és una aplicació en temps real utilitzant el software *MATLAB*. Consisteix en un pla que es pot inclinar en dos eixos, i un objecte que llisca sobre ell.

El projecte consta de dues parts diferenciades: una relacionada amb la visió i el processament d'imatges, i l'altra amb sistemes automàtics de control. Aquest projecte es desenvolupa a la part que fa referència a l’algorisme de control.

La finalitat del sistema és controlar la posició de l’objecte al pla inclinat, per al que (aquí entra la part de visió) es calcula el centre del mateix a partir d’imatges capturades per una càmera web. Obtingut el centre, i en funció de l’estat del sistema (posició origen, posició objectiu, inclinació de la plataforma però no de la posició anterior, velocitat, etc.), es calcula la nova inclinació de la plataforma perquè l’objecte llisqui fins a la posició desitjada (mitjançant la part del control). La part de visió utilitza un driver específic que captura les imatges i permet controlar el seu format, les seves dimensions, etc.

L'algorisme de control implementat, basat en la teoria clàssica de control, és un controlador de tipus proporcional.

A més, s'ha dissenyat un algorisme per determinar la posició objectiu de la peça en funció de la trajectòria i de la posició que tingui en cada instant.

El moviment de la plataforma es realitza mitjançant dos servomotors de corrent contínu, dels empleats en sistemes de radiocontrol, que funcionen amb el

microcontrolador *Arduino Mega 2560*. Els servomotors estan alimentats per una font externa de 5V.

Tota la part de visió (tant la captura com el processament de la imatge), el control, el codi de les tasques, etc., estan escrits en llenguatge M (*MATLAB*). Aquest és un programa de càlcul numèric dissenyat per treballar amb matrius.

Título: Control de una plataforma estabilizadora con un microcontrolador y una cámara web

Autor: Jordi Bustos Cunill
Tutor: Antoni Vila Marta

RESUMEN

Este proyecto es una aplicación en tiempo real utilizando el software *MATLAB*. Consiste en un plano que se puede inclinar en dos ejes, y un objeto que se desliza sobre él.

El proyecto consta de dos partes diferenciadas: una relacionada con la visión y el procesamiento de imágenes, y la otra con los sistemas automáticos de control. Este proyecto se desarrolla en la parte que hace referencia al algoritmo de control.

La finalidad del sistema es controlar la posición del objeto en el plano inclinado, por lo que (aquí entra la parte de visión) se calcula el centro del mismo a partir de imágenes capturadas por una cámara web. Obtenido el centro, y en función del estado del sistema (posición origen, posición objetivo, inclinación de la plataforma pero no de la posición anterior, velocidad, etc.), se calcula la nueva inclinación de la plataforma para que el objeto se deslice hasta la posición deseada (mediante la parte del control). La parte de visión utiliza un driver específico que captura las imágenes y permite controlar su formato, sus dimensiones, etc.

El algoritmo de control implementado, basado en la teoría clásica de control, es un controlador del tipo proporcional.

Además, se ha diseñado un algoritmo para determinar la posición objetivo de la pieza en función de la trayectoria y de la posición que tenga en cada instante.

El movimiento de la plataforma se realiza mediante dos servomotores de corriente continua, de los empleados en sistemas de radiocontrol, que funcionan con el

microcontrolador *Arduino Mega 2560*. Los servomotores están alimentados por una fuente externa de *5*V.

Toda la parte de visión (tanto la captura como el procesamiento de la imagen), el control, el código de las tareas, etc., están escritos en lenguaje M (*MATLAB*). Éste es un programa de cálculo numérico diseñado para trabajar con matrices.

.

Title: Control of a stabilizing platform with a microcontroller and a webcam

Author: Jordi Bustos Cunill

Tutor: Antoni Vila Marta

ABSTRACT

This project is a real-time application using *MATLAB* software. It consists of a plan that can be tilted in two directions, and an object that slides on it.

The project consists of two parts: one related to vision and image processing and the other one with automatic control systems. This project develops the part that refers to the control algorithm.

The purpose of the system is to control the position of the object on the inclined plane, for which (here comes the vision) its own center is calculated from the images captured by an analogy camera. Retrieved the center, and depending on the state of the system (source position, target position, tilting platform, etc.), the new slope of the platform is calculated so the object slides to the desired position (using the control part). The vision part uses a specific driver that captures images and allows to control its format, its size, etc.

The control algorithm implemented, based on a classical control theory, is a P controller.

Furthermore an algorithm has been designed in order to determinate the target position of the piece in function of its current position and trajectory.

The movement of the platform is carried out using two DC servomotors which are used in radio control systems that work with *Arduino Mega 2560* microcontroller. The servomotors are powered by a 5V source.

All the vision part (both capture and image processing), the control, the code assignments, etc., are written in M. (*MATLAB*) language. This is a program that uses numerical calculations and is designed to work with numerical arrays.

ÍNDEX GENERAL

Documents bàsics:

Pàg.

MEMÒRIA

ÍNDEX MEMÒRIA

Pàg.

1 INTRODUCCIÓ

Aquest projecte es centrarà en el desenvolupament d'un controlador per al que s'executarà en el software *MATLAB.* La finalitat del sistema és controlar la posició d'una peça, que es troba sobre un pla que hem d’inclinar perquè es desplaci fins la posició desitjada. El problema consisteix en moure-la d'una posició a una altra controlant els angles d’inclinació del pla. És per tant necessari fer una breu introducció als sistemes de control, així com als sistemes operatius i, especialment aquells que, posseeixen certes característiques que, a diferència dels sistemes operatius de propòsit general, els permet l'execució d'aplicacions en temps real.

1.1 Objectius del projecte

El propòsit general és fer un programa o aplicació automatitzada que permeti posar a punt tècniques de control per processament d'imatges utilitzant el *MATLAB* i poder controlar una peça per mitjà d’una plataforma estabilitzadora amb un microcontrolador i una càmera web. El microcontrolador utilitzat és una placa d’A*rduino Mega 2560*.

L’objectiu és controlar la posició d'una peça dins d'un pla quadrat. S'actua sobre la inclinació d'aquesta plataforma mitjançant dos comandaments, un per a cada eix.

Es pretén substituir la vista de la persona per visió artificial, i el raonament i actuació per un controlador, que funcionin sense intervenció externa. Llavors, és realitzarà una aplicació automatitzada que corri en el sistema operatiu en temps real, i que controli la posició de la peça a la superfície del pla. Aquesta aplicació es compon de dues parts fonamentals que es tractaran de forma independent, ambdues imprescindibles:

- Captura d'imatges i càlcul del centre de masses de la peça.
- Algorisme per controlar la posició de la peça en el pla.

Per fer la captura d’imatges s’utilitzarà el *MATLAB*. Per tal d'utilitzar una càmera web com a sensor d'entrada per al programa de control, dues caixes d'eines addicionals seran necessàries, la imatge Acquisition Toolbox i la Imatge Processing Toolbox. En quan l'adquisició de la imatge hi ha dos possibles mètodes d'acaparament de dades d'imatge de la càmera. Una manera és utilitzar la funció *getsnapshot (vid)*, on *vid* és una càmera de vídeo i la funció retorna una imatge. Si bé aquest mètode és fàcil d'usar i implementar, el problema és el temps requerit. Sempre que *gentsnapshot* és cridat, la càmera primer es posa en marxa, a continuació fa una foto, i finalment es desactiva, el que resulta una pèrdua de temps. Un altre mètode és utilitzar un disparador, que és el mètode de comptar fotogrames que *MATLAB* fa amb les imatges adquirides. L'avantatge d'utilitzar un disparador és ara una funció que pot ser cridada després d'un cert nombre de trames que han estat adquirides per la càmera sense tenir el retard entre l’inici i detenció de la càmera.

Per tal de prendre avantatge d'això, certes propietats necessàries seran establertes. Per tenir un nombre il·limitat d'imatges per tret, la càmera prendrà un continu d'imatges fins que s’aturarà. El nombre de quadres adquirits abans de cridar una funció s'estableix en un, encara que aquest s’ajustarà més tard.

Es pot suposar que sempre coneixerem la posició en què es troba la peça i la posició a la qual volem dirigir-la. Això no és exactament cert, ja que només tindrem coneixement d'aquestes dades a partir de la càmera, i la càmera en si podem tractar-la com un convertidor Analògic / Digital, ja que a partir d'ella obtenim mostres periòdiques d'un senyal continu (la posició de la peça). De manera que només coneixem la posició en els instants de mostreig (suposant menyspreable qualsevol retard intern o de processat) i també menystenint la posició anterior i per tant la velocitat i l’acceleració.

A partir de la hipotètica i recentment calculada posició d'origen, hem d'actuar perquè la plataforma s'inclini en el sentit i magnitud angular apropiats, fent que la peça es dirigeixi en la direcció desitjada.

S'haurà de controlar l'angle d'inclinació del pla mitjançant dos servomotors. Cada servomotor està connectat als costats de la plataforma, de manera que un controla la inclinació en l'eix *x*, i l'altre en l’eix *y*. Perquè funcioni bé, el moviment haurà de ser independent en els dos eixos. Els eixos seran perpendiculars.

2 SISTEMA DE CONTROL

En els últims anys, els sistemes de control automàtics han adoptat un paper de creixent importància en el desenvolupament i avenç de la civilització i tecnologia modernes. A nivell domèstic, els controls automàtics en els sistemes de calefacció i condicionament d'aire regulen la temperatura i la humitat de les llars modernes per aconseguir ambients confortables. Rentadores, vitroceràmiques, frigorífics, rentavaixelles, planxes, màquines fotogràfiques, etc. són altres exemples de sistemes de control que podem trobar a la llar. En la indústria, els sistemes de control automàtics es troben en nombroses aplicacions, com ara el control de qualitat de productes manufacturats, l'automatització, control de màquines, eines, sistemes moderns de tecnologia espacial i d'armes (guiat de projectils), sistemes de computadors, sistemes de transport (pilotatge d'avions) i la robòtica. Fins i tot problemes com ara el control de magatzems, control de sistemes socials i econòmics, i control de sistemes ambientals i hidrològics, poden enfocar també des del punt de vista de la teoria del control automàtic.

Un sistema de control és un conjunt de components (no tenen perquè ser físics) connectats de manera que ell mateix pugui comandar, dirigir o regular, a si mateix o a un altre sistema.

En quant als tipus de sistemes de control (llaç obert i llaç tancat), s’ha utilitzat el sistema de control de llaç tancat.

2.1 Sistemes de control de llaç tancat

La següent figura representa de forma esquemàtica un sistema de control realimentat:

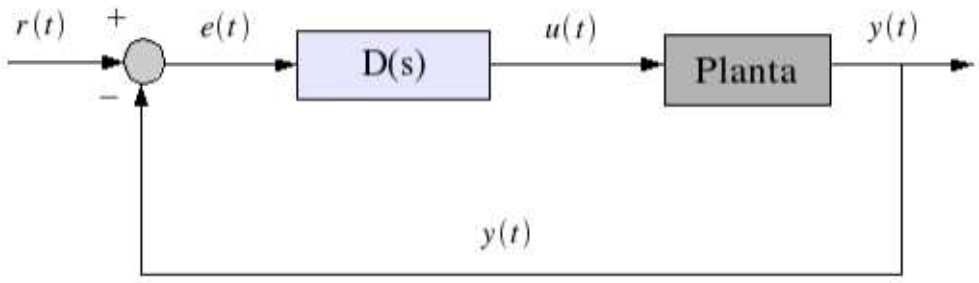


Figura 2.1: Sistema de control de llaç tancat

En aquests sistemes el senyal de sortida, *y (t)* té un efecte directe sobre l'acció del control. També se'ls coneix com a sistemes realimentats. El senyal d'error *e (t)*, que és la diferència entre el senyal d'entrada, *r (t)* i el senyal realimentat, y (t) (que a la vegada ha de ser el senyal de sortida, o una funció del senyal de sortida), realimenta el controlador, (representat en la figura per la seva funció de transferència D (s)) de forma que aquest intenta reduir l'error i portar la sortida del sistema a un valor desitjat.

Definició de Sistema:

Un sistema és un conjunt de components que s'interrelacionen i treballen junts per realitzar un objectiu determinat.

Definició de planta:

La planta normalment és un conjunt de parts que treballen juntes amb l'objectiu de realitzar una operació en particular. Se l'anomena planta a qualsevol sistema físic que es desitgi controlar.

3 DISSENY DE L’ALGORISME DE CONTROL

3.1 Funció de l’Algorisme

S'ha d'encarregar de prendre les dades calculades a la part del processat de la imatge i convertir-lo en una resposta del sistema adequada. Com hem dit, és desitjable que el sistema de control sigui realimentat per fer el sistema insensible (en la mesura del possible) tant a les pertorbacions externes com a les variacions dels paràmetres del sistema. No obstant això, el sistema ha de ser estable. Idealment, hauria de calcular un parell d’angles tals que, movent els motors a aquesta posició, la peça es mogui cap a on nosaltres vulguem.

Conèixer la posició de la peça no és suficient per poder controlar-la, sinó que necessitarem més informació.

Hi ha molts tipus d'algorismes de control, des de la teoria de control clàssica, passant pel control òptim i adaptatiu, fins als nous sistemes de control implementats mitjançant xarxes neuronals o lògica difusa. L'algorisme de control triat ha de ser estable, a més de complir certes especificacions com poden ser el tant per cent de sobredispar, error en estat estacionari o el temps de establiment. A més, el sistema de control ha de poder rebutjar o reduir l'efecte de qualsevol interferència a la sortida de la planta.

Una altra característica d'un sistema de control és la insensibilitat als errors en el model de la planta. Aquest model pot contenir paràmetres pel qual el valor no sigui conegut amb exactitud (de fet, és molt probable que així sigui). Si l'algorisme és prou robust, el sistema de control ha de funcionar correctament malgrat aquests valors incorrectes.

3.2 Plantejament del problema

Abans d'entrar pròpiament amb la part matemàtica i física del model, veiem una petita introducció amb els elements que considerarem, així com de les suposicions i aproximacions que s'han fet per simplificar el treball de modelatge.

Tenim una peça en un pla. Per facilitar la descripció, suposem que només tenim un grau de llibertat, és a dir, que la peça només es pot desplaçar al llarg de l' eix *x* del plànol, mantenint fixa la seva posició en l'eix *y*. Després a la pràctica n’hi haurà prou en passar el problema a dues dimensions utilitzant càlculs vectorials.

Tenim un servomotor unit a la superfície mitjançant el braç del servo i una vareta. Quan el servo gira un angle Θ, la palanca fa que el pla s'inclini un angle α. Quan l'angle és diferent de zero, la gravetat fa que la peça es desplaci al llarg del pla. Recordem que volem dissenyar un algorisme de control que ens permeti gestionar la posició de la peça.

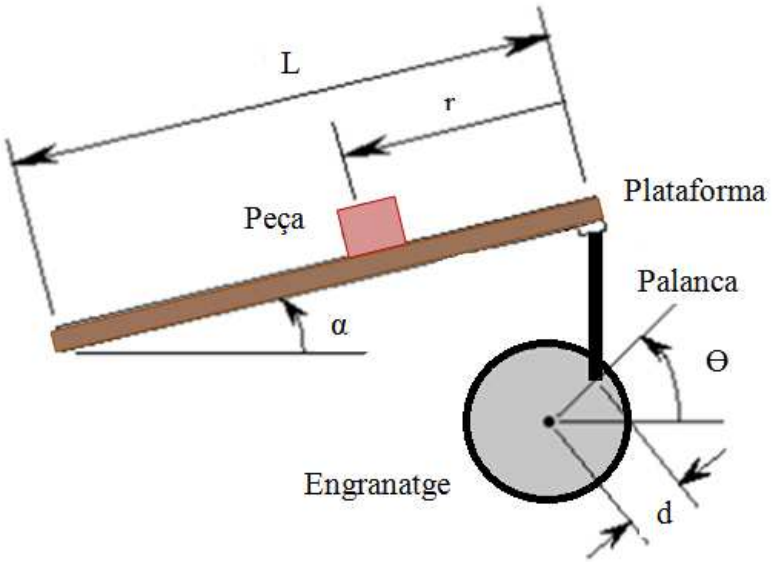


Figura 3.1: Planta del sistema

A la figura 3.1 apareix un dibuix esquemàtic del problema que hem de resoldre, i a continuació s'exposen les constants físiques i les variables sobre les que s'ha modelat matemàticament el sistema:

M : massa de la peça = 12 g
D : longitud de la palanca = 2.5 cm
g : acceleració de la gravetat = 9.8 m/s^2
L : longitud del plànol = 6 cm
r : posició de la peça
α : Angle que forma el pla amb l'horitzontal (º)
Θ : Angle del servo (º)

3.3 Modelat del sistema

Es descomponen les forces que actuen sobre la peça:

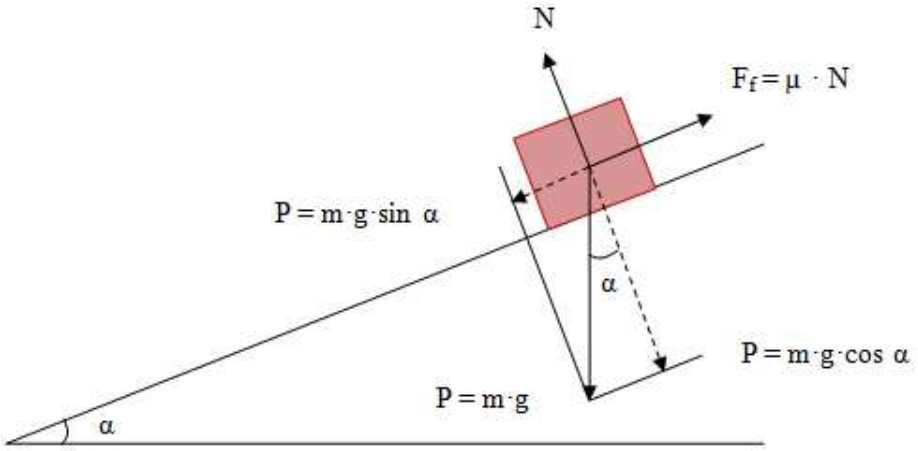


Figura 3.2: Components que actúen sobre la peça

A continuació es calculen les equacions que intervenen:

$$N = m \cdot g \cdot cos\alpha$$

$$F_f = \mu \cdot N$$

$$F_f = \mu \cdot m \cdot g \cdot cos\alpha$$

Apliquem la Segona Llei de Newton:

$$m \cdot \ddot{r} = -m \cdot g \cdot sin\alpha + \mu \cdot m \cdot g \cdot cos\alpha$$

Simplifiquem i obetenim l'equació del moviment de la peça:

$$\ddot{r} = -g \cdot sin\alpha + \mu \cdot cos\alpha$$
$$\ddot{r} = -g(sin\alpha + \mu cos\alpha)$$

Aproximant aquesta equació per a l'angle $\alpha \sim 0$:

$$\alpha \sim 0$$
$$sin\alpha \sim \alpha$$
$$cos\alpha \sim 1$$

$$\ddot{r} = -g(\alpha - \mu)$$

Podem aproximar l'equació que relaciona l'angle del pla amb el del servomotor, mitjançant la següent equació linial:

$$\alpha = \frac{d}{L} \cdot \theta$$

Substituint això a l'equació anterior, tenim:

$$\ddot{r} = -g(\frac{d}{L} \cdot \theta + \mu)$$

Tant el coeficient estàtic com dinàmic són molt petits i per tal de simplificar l'equació, anem a imposar que el fregament es 0:

$$\ddot{r} = -g(\frac{d}{L} \cdot \theta)$$

3.4 Funció de transferència

Agafant transformades de Laplace en l'equació anterior, i assumint que les condicions inicials son nul·les, obtenim:

$$s^2 \cdot R(s) = -g(\frac{d}{L} \cdot \theta(s))$$

Aïllant, obtenim la funció de transferència:

$$\frac{R(s)}{\theta(s)} = \frac{-g \cdot d}{L} \cdot \frac{1}{s^2}$$

3.5 Simulació en *MATLAB*

Abans de programar el procediment que ens permeti obtenir la inclinació que han de tenir els motors perquè la peça arribi a la posició desitjada en un temps T (interval entre imatges), simulem el comportament del sistema complet en MATLAB. La simulació ens permetrà saber si el sistema és estable o inestable.

Simulem la funció de transferència del sistema obtinguda anteriorment:

$$\frac{R(s)}{\theta(s)} = \frac{-g \cdot d}{L} \cdot \frac{1}{s^2}$$

```
g = -9.8;       % Gravetat (m/s^2)
L = 0.06;       % longitud del pla fins l'eix (m)
d = 0.025;      % offset servo (m)
```

```
s = tf('s');
P_obj = -(g*d/L)/s^2

step(P_obj)     % Gràfica

Transfer function:
4.083
-----
 s^2
```

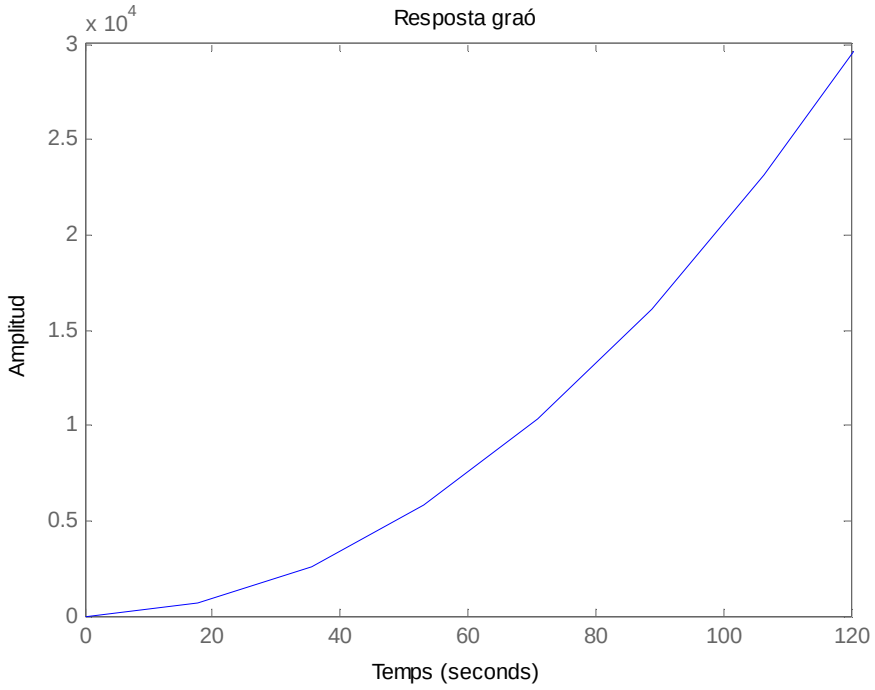


Figura 3.3 Resposta graó de la planta en llaç obert

En aquesta gràfica queda clar que el sistema és inestable en llaç obert, de manera que la peça es desplaçarà fins al final de la plataforma. Necessitarem algun mètode per controlar la posició de la peça dins del pla.

El mètode escollit ha estat un controlador proporcional (P).

3.6 Controlador proporcional

En el següent diagrama de blocs es mostra la representació del sistema juntament amb el controlador P. La sortida (posició d'objecte) serveix també per realimentar el sistema:

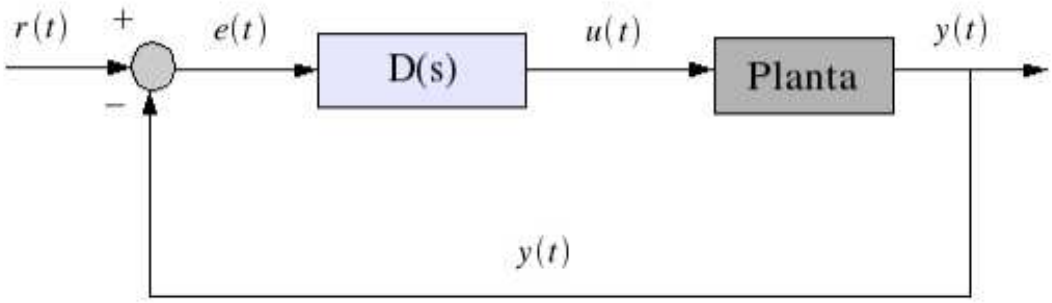


Figura 3.4: Conjunt Sistema + Controlador, amb realimentació unitat

A la figura 3.1, *r (t)* és l'entrada de referència. Si *r (t)* és zero tot el temps, llavors es coneix a *D (s)* com a regulador, i al disseny de *D (s)* és conegut com problema de regulació. En aquest cas, el propòsit de *D (s)* és mantenir la sortida *y (t)* a zero en el cas que hagués interferències externes en el sistema.

Una generalització del problema del regulador és el problema del punt fix. En aquest cas, *r (t)* és una constant, i el propòsit de *D (s)* és mantenir la sortida a un valor de referència constant, anomenat el punt fix, oposant-se a les interferències externes del sistema. Si *r (t)* no és constant durant tot el temps, llavors a *D (s)* se l'anomena compensador o controlador, i al disseny de *D (s)* se l'anomena problema de disseny de servo. El senyal *e (t)* és el senyal d'error entre l'entrada de referència i la sortida actual de la planta, i *u (t)* és el senyal d'entrada de la planta, generada pel compensador per fer que aquesta es comporti de la manera que desitgem.

La funció de transferència del controlador P ve donada per l'expressió:

$$u(t) = Kp \cdot e(t)$$

La realimentació té mòdul unitat i signe negatiu, de manera que l'entrada sobre la qual està treballant realment no és la posició de la peça, sinó que treballa sobre la diferència entre el valor d'entrada desitjat i el valor de sortida. A aquesta diferència se l'anomena error de seguiment

El senyal a la sortida del controlador és igual a K_P (guany proporcional) vegades la magnitud de l'error.

El senyal u (t) s'envia a la planta, obtenint-ne la nova sortida. Aquesta nova sortida es torna a enviar al sensor per calcular la nova senyal d'error.

3.7 Controlador proporcional digital

La següent figura mostra el típic sistema continu de controlador i planta realimentats, que van poder implementar-se en la majoria dels casos mitjançant electrònica analògica.

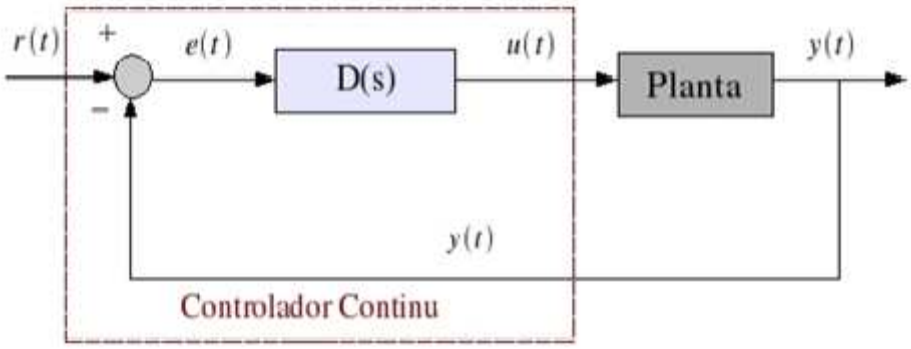


Figura 3.5: Controlador continu

El controlador continu (dins de la línia puntejada, en el dibuix anterior) pot ser substituït per un controlador discret que realitzi les mateixes tasques de control, tal com es mostra en el següent esquema:

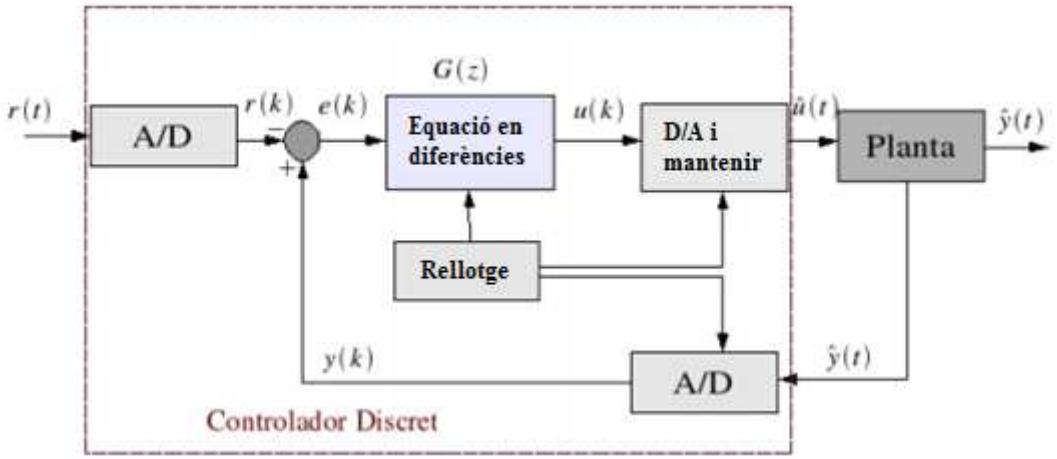


Figura 3.6: Controlador discret

La diferència bàsica entre aquests sistemes és que el controlador analògic treballa amb senyals continus, mentre que el sistema digital opera amb senyals discrets, obtingudes a partir de mostres del senyal continu.

Nosaltres no sabem en tot moment la posició de la peça, de manera que no coneixem *y (t*). No obstant això, vam rebre mostres d'aquest senyal cada vegada que es realitza l'adquisició d'una imatge i els programes encarregats del processament de la mateixa ens donen el centre de la peça en aquest instant. Això és equivalent a tenir la senya*l y(k),* formada a partir de mostres equiespaiades del senyal analògic *y (t),* que és un senyal continu en el temps, de la posició real de la peça.

El primer pas per dissenyar un controlador discret en el temps és obtenir un model de la planta també discret en el temps. Després hem de dissenyar un compensador discret per al model discret de la planta.

El sistema de control digital s'obté llavors connectant el compensador discret en el temps a la planta analògica mitjançant convertidors A / D i D / A.

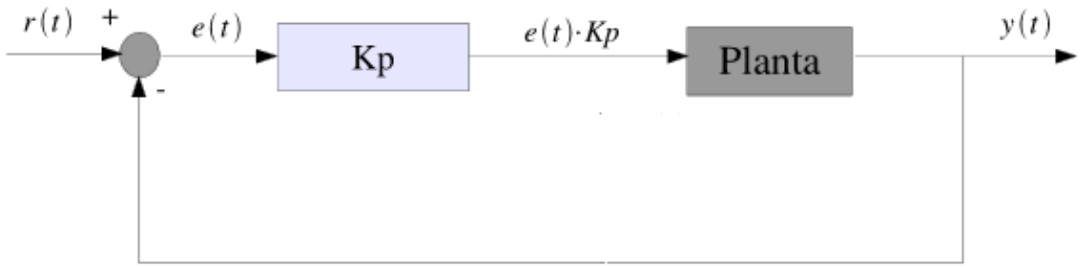


Figura 3.7: Conjunt Sistema + Controlador P, en llaç tancat

El senyal que arriba a la planta respon a la següent expressió:

$$u(t) = (r(t) - y(t)) \cdot Kp$$

$u(t)$ (°)
$r(t)$ (mm)
$y(t)$ (mm)
kp $\left(^{\circ}/_{mm}\right)$

I tenint en compte que el nostre sistema és digital, ja que treballem amb mostres del senyal d'entrada, per a cada instant discret de temps, tenim la següent equació per modelar el nostre algorisme de control:

$$u[k] = (r[k] - y[k]) \cdot Kp$$

$u[k]$ (°)
$r[k]$ (mm)
$y[k]$ (mm)
kp $\left(^{\circ}/_{mm}\right)$

Es pot observar que el primer i únic terme, multiplicat per la constant proporcional K_P, és l'error entre la posició desitjada i la posició actual de la peça (o més concretament, el centre que obtenim a partir de la càmera i el processat d'imatges, ja que pot ser que hi hagi un retard des de l'instant d'adquisició i el moment en què es realitza el processat).

La K_p utilitzada és $0.010^{\circ}/_{mm}$. S'han anat provant valors inferiors a 1 º/mm fins que la senyal s'ha mantingut marginalment estable.

Ara, podem modelar la resposta del sistema a una entrada esglaó de 0,25 m. Simulem el nou senyal que arriba a la planta per mitjà del següent codi:

```
g = -9.8;          % Gravetat (m/s^2)
L = 0.06;          % longitud del pla fins l'eix (m)
d = 0.025;         % offset servo (m)
s = tf('s');
P_obj = -(g*d/L)/s^2;
Kp = 0.010;        % (º/mm)
C = pid(Kp);
sys_cl=feedback(C*P_obj,1);
step(0.25*sys_cl)
axis([0 120 0 0.5])
```

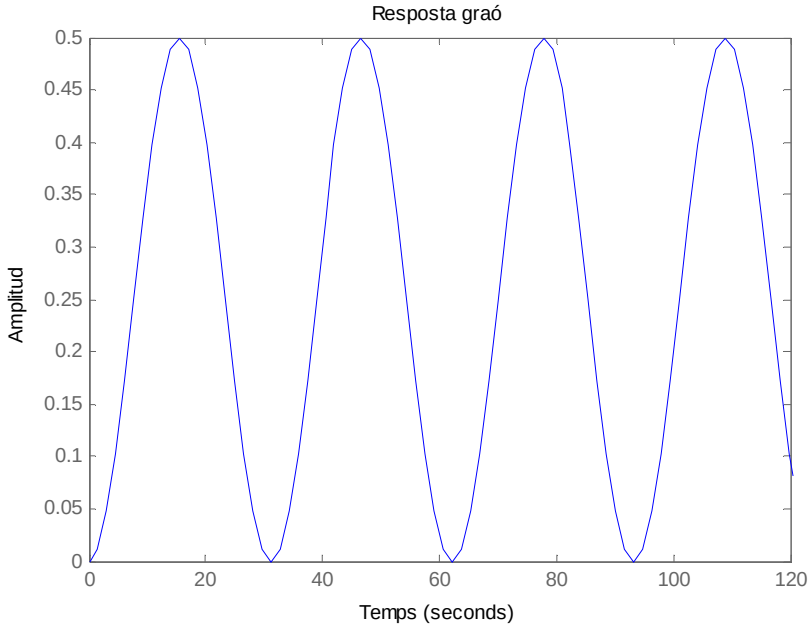


Figura 3.8:Resposta graó amb kp del sistema en llaç tancat

Com és pot observar en la figura anterior, el sistema es marginalment estable. A la realitat no seria axins, ja que nosaltres hem suposat que no hi ha fregament i que la peça llisca.

Com a característiques principals d’un control proporcional, tenim:

- Millora la dinàmica del sistema.
- Millora la precisió del sistema: Però no desapareix l’error estacionari.
- Augment de la inestabilitat relativa.
- Aparició de saturacions.

4 IMPLEMENTACIÓ DEL PROGRAMA DE CONTROL

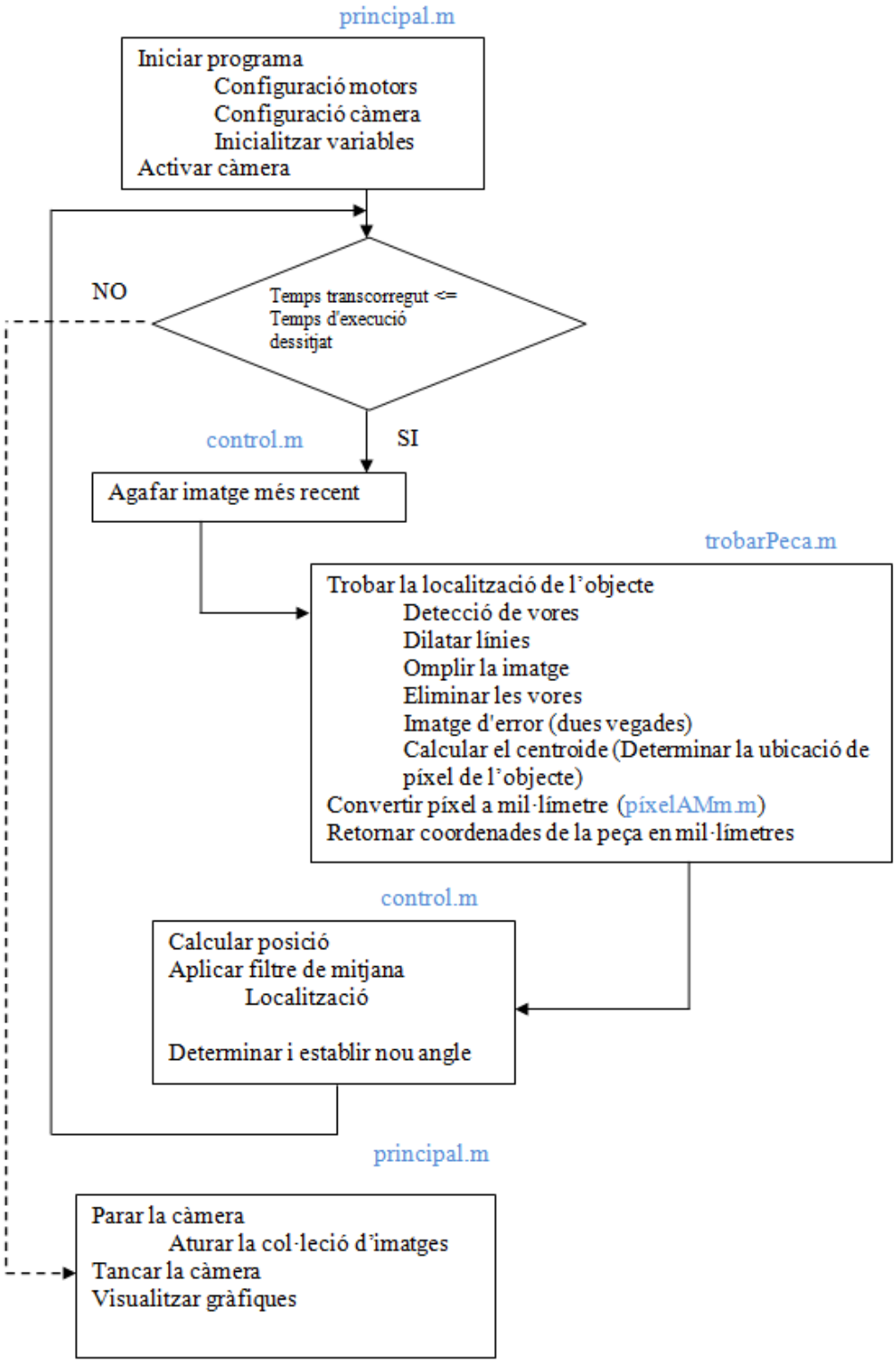
4.1 Estructura del programa

El llenguatge de programació del controlador ha estat implementat en *MATLAB*, ja que permet una major velocitat, i pel que fa a la programació serà més fàcil. La sortida gràfica i de processament d'imatges també serà més simple.

L’estructura del programa ve diferenciada per un programa principal i tres funcions dintre d’aquest, que són:

- Programa *principal*
    - Funció *control*
    - Funció *trobaPeca*
    - Funció *pixelAMm*

Per tal de veure aquestes funcions quines tasques realitzaran i quan es duran a terme, a continuació es pot visualitzar de forma esquemàtica tot el procés de control:


principal.m
Iniciar programa
Configuració motors
Configuració càmera
Inicialitzar variables
Activar càmera
NO
Temps transcorregut <= Temps d'execució dessitjat
SI
control.m
Agafar imatge més recent
trobarPeca.m
Trobar la localització de l'objecte
Detecció de vores
Dilatar línies
Omplir la imatge
Eliminar les vores
Imatge d'error (dues vegades)
Calcular el centroide (Determinar la ubicació de píxel de l'objecte)
Convertir píxel a mil·límetre (píxelAMm.m)
Retornar coordenades de la peça en mil·límetres
control.m
Calcular posició
Aplicar filtre de mitjana
Localització
Determinar i establir nou angle
principal.m
Parar la càmera
Aturar la col·lecció d'imatges
Tancar la càmera
Visualitzar gràfiques

4.2 Descripció del programa principal i les funcions

4.2.1 Programa *principal*

És l'algorisme principal del programa de control, on a partir d'aquest s'executen totes les funcions.

El primer que es fa és obrir el port on tenim connectada la placa d' *Arduino Mega 2560* i establir connexió amb el *MATLAB.* Aquest procés dura uns segons.

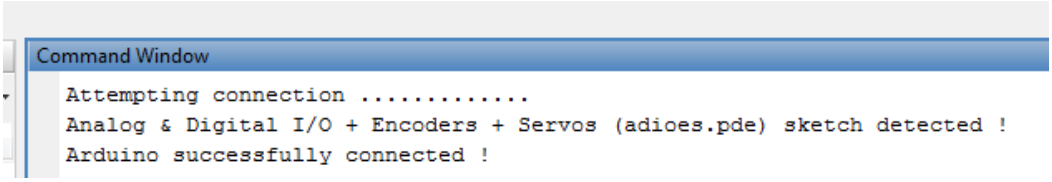


Figura 4.1 Connexió Arduino Mega 2560 amb MATLAB

Cal tenir present que si s'executa el programa des d'un ordinador diferent, caldrà prèviament establir quin port estem utilitzant i canviar-lo en el programa, ja que sinó, no serà possible la connexió i apareixerà un error.

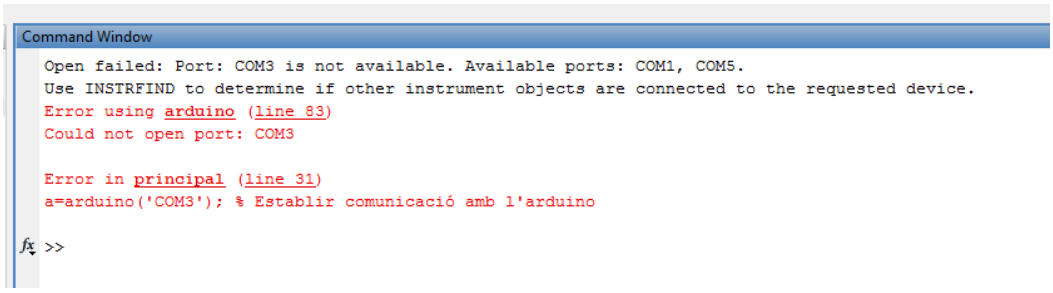


Figura 4.2 Error de connexió

Un cop establerta la connexió, es configuraran els servomotors per tal que es posicionin segons en nostre criteri, s'activarà la càmera web i s'inicialitzaran totes les variables. Sabrem que el programa comença a funcionar quan aparegui en pantalla: "Inici".

Per altre banda s'activarà una pantalla, on es mostrarà sobre un eix de coordenades de la mida de la plataforma la ubicació de l'objecte en tot moment.

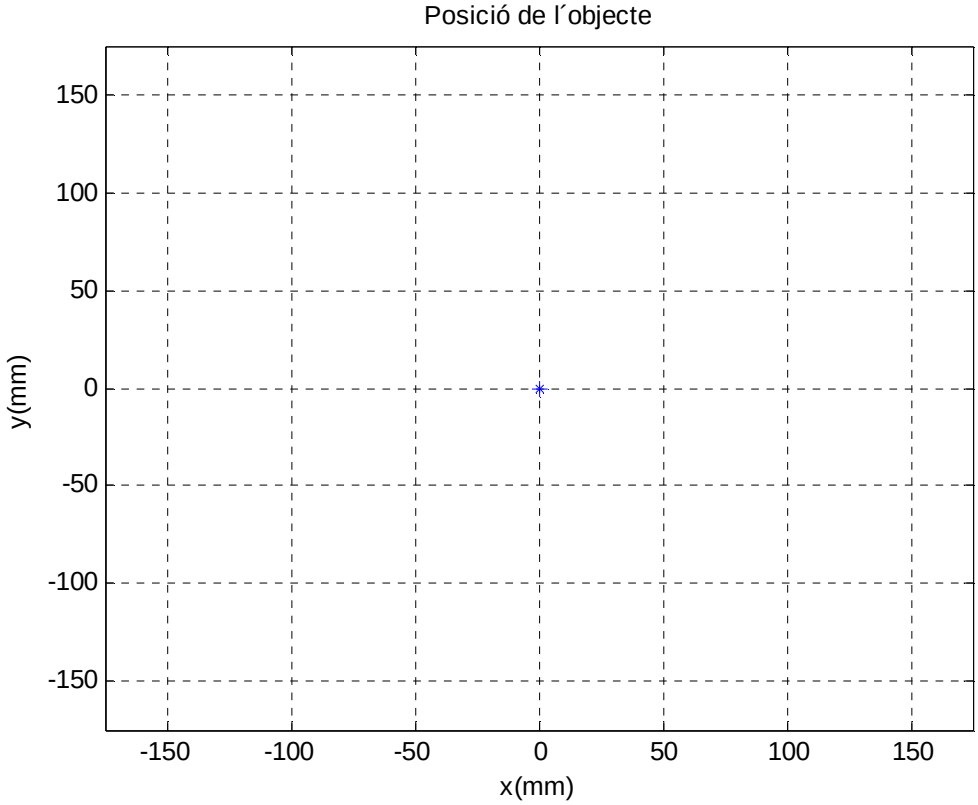


Figura 4.3 Posició de l'objecte en la plataforma

Sempre que no hi hagi objecte la plataforma estarà completament horitzontal i es representarà com si aquest es trobés al centre, tal i com es mostra a la figura anterior. A l'hora de representar gràficament la posició de la peça, el programa no distingeix de quan l'objecte està al centre de l'eix de coordenades de quan no hi ha peça, és a dir, ho interpreta tot de la mateixa manera.

Un cop hagi transcorregut el temps d'execució que haguem fixat, el programa s'aturarà, juntament amb la càmera web. Apareixerà un text a pantalla que dirà: "Fet", i a continuació es mostrarà una gràfica on es veuran tots els moviments realitzats per l'objecte al llarg del programa en funció del temps.

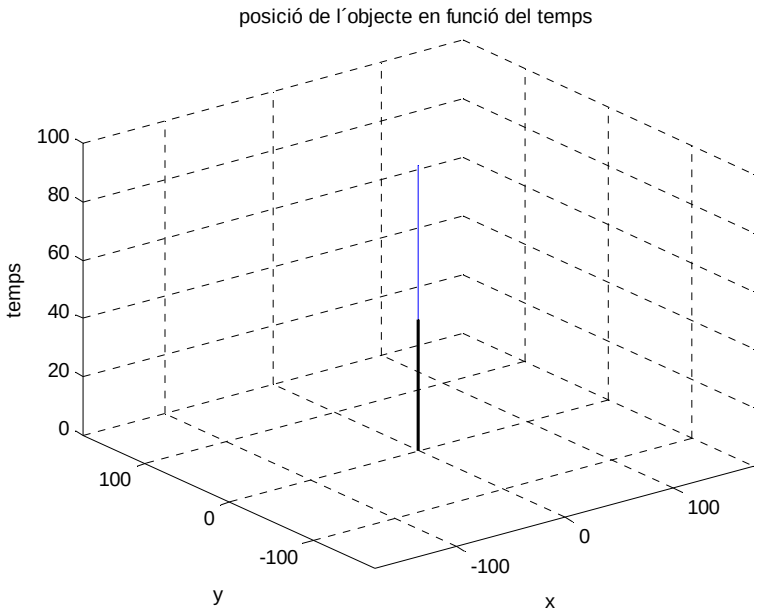


Figura 4.4 Posició de l'objecte en funció del temps

Com es pot observar a la figura 4.4, la línia negra representa l'eix central mentre que la línia blava representa per allà on s'ha desplaçat l'objecte. Al no tenir res i tal com hem dit anteriorment, el programa detecta com si l'objecte estigués situat al centre.

A partir de la funció del programa principal s'aniran cridant les altres funcions. Per tant, executant només aquest, es posarà en funcionament tot l'algorisme de control.

4.2.2 Funció *control* *(vídeo = [angle servo eix x, angle servo eix y])*

En aquesta part, l'algorisme ens dóna constantment la posició de l'objecte a través de la funció *trobarPeca* per mitjà de la càmera web. Depenent d'aquesta posició els servos giraran uns certs graus per tal d'inclinar la plataforma.

Per tal de poder determinar cada cop un nou angle s'utilitzarà la següent expressió:

```
nouAngle=((posDesitjada(:,1)-mitjLoc(:,1)).*kp+pi/2)*360/(2*pi)
```

- *posDesitjada* → Fa referència a la posició que hem establert des del principi. En el nostre cas serà el centre de la plataforma, el punt (0,0).
- *mitjLoc*→ És la posició actual de l'objecte. Aquesta posició anirà variant segons el recorregut que adopti l'objecte.
- *Kp* → És la constant de proporcionalitat que hem utilitzat. Té un valor de 0,010. Les unitats són $^{\circ}/_{mm}$.

La part de l'expressió on es multiplica *360/(2*pi)* és deguda a que el *MATLAB* opera en radiants, mentre que la placa d'arduino opera en graus. El que es fa és passar de radiants a graus.

Pel que fa la part on es suma *pi/2*, és conseqüència a la relació de l'angle del servo amb la placa, ja que perquè la placa estigui completament horitzontal al servo se li ha d'entrar 90º.

Un cop tinguem l'angle calculat s'envia per mitjà d'una comunicació sèrie cap a la placa d'arduino i calcula la ocupació corresponent del PWM per posicionar els servos en l'angle considerat.

4.2.3 Funció ***trobarPeca*** *(Imatge = [posició píxels eix x, posició píxels eix y])*

És la funció encarregada de localitzar la posició de la peça de manera contínua i enviar-la a la part de control. En aquest cas el sensor serà la càmera web.

Per tal de dur a terme aquest algorisme, s'ha hagut de fer un tractament d'imatges a partir d'unes funcions establertes pel propi *MATLAB* en el qual es treballa en escala de grisos en lloc de treballar en escala RGB (Red Green Blue).

A continuació podem observar al detall tot el procés del tractat de la imatge.

En primer lloc, cal dir que des de la funció *principal* ja activem la càmera web a partir de la següent funció:

```
vid=videoinput('winvideo')
```

Simplement el que fa aquesta funció és crear un objecte d’ entrada de vídeo.

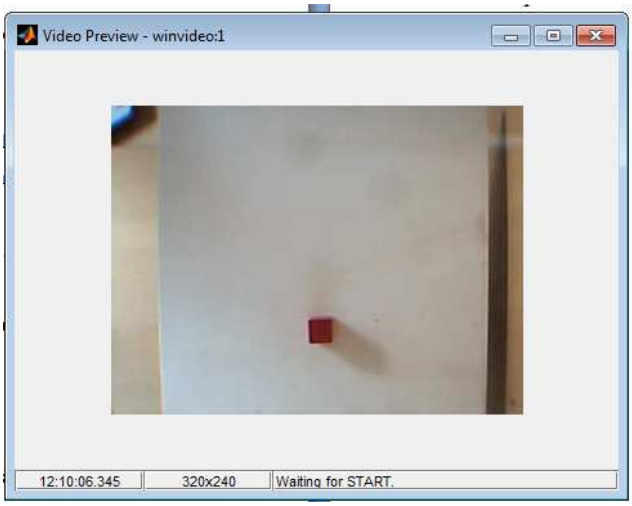


Figura 4.5 Visionat càmera web

Després és passa la imatge a escala de grisos:

```
vid.ReturnedColorSpace='grayscale';
```

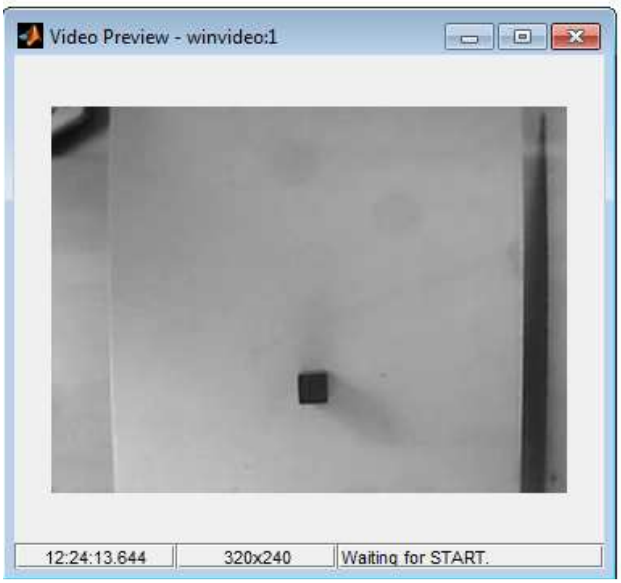


Figura 4.6 Imatge a escala de grisos

A partir d’aquí la funció *trobarPeca* fa tot un procés de tractat d’imatge per tal de trobar la posició exacte de l’objecte.

El primer pas és espessir les línies. Per això utilitzem la funció:

```
BWs = edge(Im,'sobel', threshold * fudgeFactor)
```

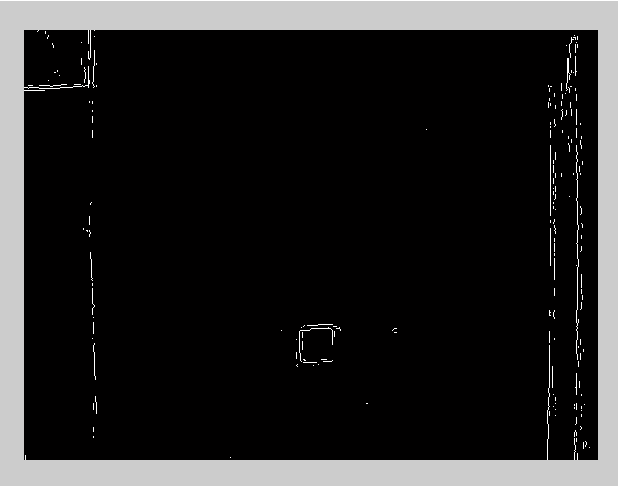


Figura 4.7 Visionat espessir les línies

A continuació anem a donar més gruix a les línies. Això es fa d’aquesta manera per tal poder definir amb més claredat el contorn de l’objecte a controlar. Utilitzem:

```
BWsdil = imdilate(BWs, [se90 se0]);
```

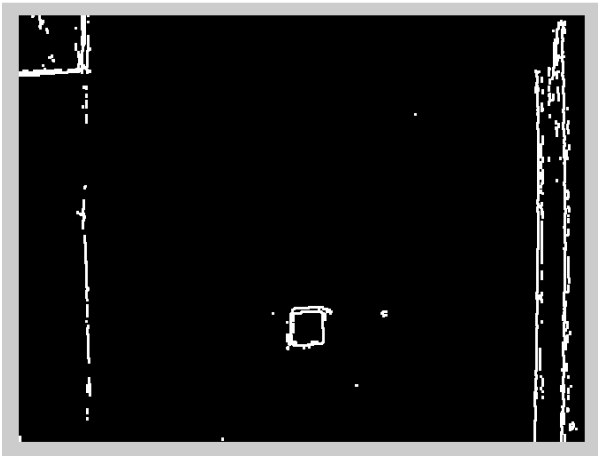


Figura 4.8 Visionat gruix de línia

Un cop hi ha tots els contorns definits, es procedeix a omplir els forats.

```
BWdfill = imfill(BWsdil, 'holes');
```

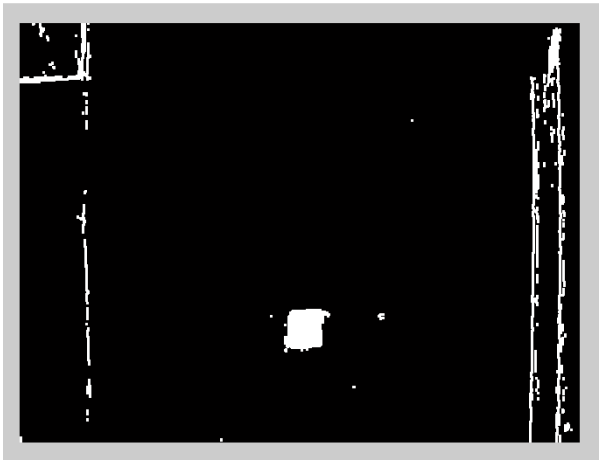


Figura 4.9 Visionat omplir forats

El següent pas és eliminar tots els objectes que estan tocant a la frontera i llavors els dels voltants. En aquest cas, qualsevol coloració blanca que toqui els límits. Això es fa perquè a la frontera solen haver-hi ombres i per tal de no confondre el programa el que es fa és eliminar-les.

```
BWnoborder=imclearborder(BWdfill, 4);
```

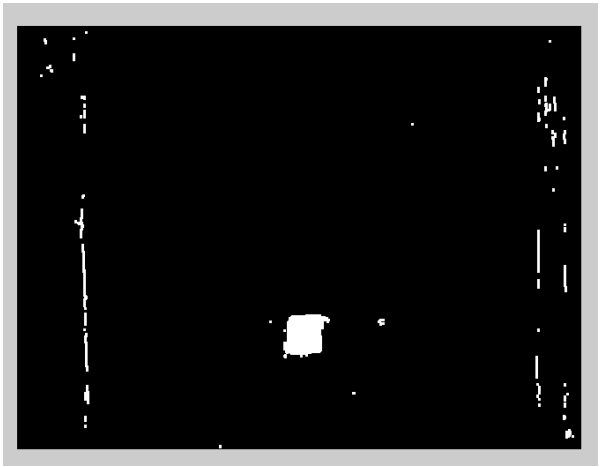


Figura 4.10 Visionat eliminació d’objectes a la frontera

Finalment, només queda erosionar la imatge perquè es vegi amb més claredat la peça i amb el mínim de punts blancs possibles. El resultat final és el següent:

```
BWseg = imerode(BWnoborder,seD);

BWseg = imerode(BWseg,seD);
```

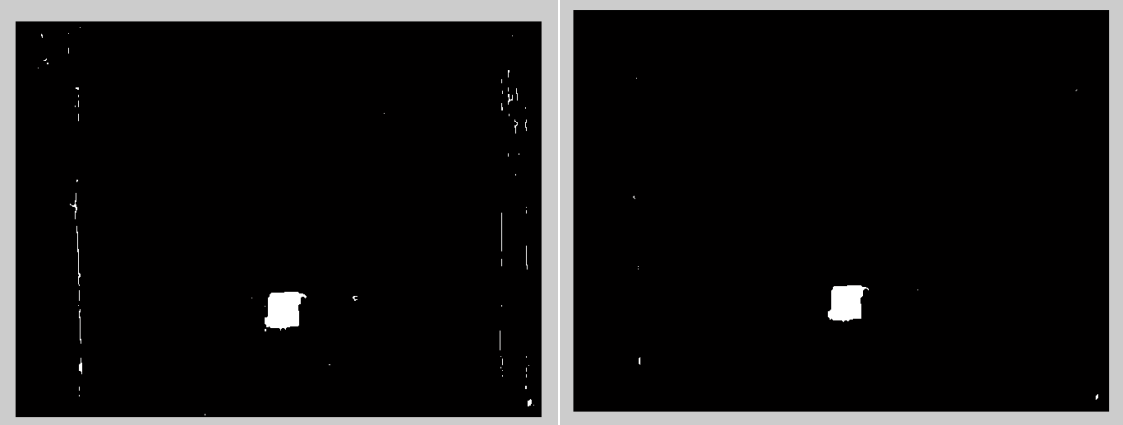


Figura 4.11 Visionat erosionat imatge final

A partir d'aquí es troba el centroide de la peça per saber-ne la posició i es determina un interval d'àrees per els possibles punts blancs que puguin sorgir que no siguin interpretats com a altres objectes. Es realitza de la següent manera (*veure funció trobarPeca*):

```
L = bwlabel(BWseg);
regio = regionprops(L, 'Centroid');
area=regionprops(L,'Area');
max=0;
referencia=0;

for i=1:numel(area)
    if area(i).Area>175 && area(i).Area<5000 && area(i).Area>max
        max=area(i).Area;
        referencia=i;
    end
end
```

Per trobar el centroide (posició del píxel) de la peça utilitzem:

```
regio = regionprops(L, 'Centroid');
```

Per trobar l'àrea (en píxels) de la peça utilitzem:

```
area=regionprops(L,'Area');
```

Tal i com s’ha dit anteriorment per tal de diferenciar l’àrea de la peça de les possibles ombres s’ha establert un condicional:

```
area(i).Area>175 && area(i).Area<5000 && area(i).Area>max
        max=area(i).Area;
```

La peça té una dimensió de 400 píxels. Aquesta pot variar molt poquet degut a petites ombres que puguin aparèixer al seu voltant. És per aquest motiu que les condicions que s’han establert per la peça és que sigui més gran de 175 píxels i més petita de 5000 píxels. I una última condició perquè l’àrea que es trobi, sempre sigui la màxima. Un cop fet el tractament d’imatge explicat anteriorment, es pot veure com queden petits punts blancs. L’algorisme calcularà totes les àrees possibles i sempre és quedarà amb la més gran, que en aquest cas sempre serà la de la peça.

Definició de centroide:

Centre de massa d'un objecte amb densitat uniforme. Per a un objecte unidimensional uniforme de longitud L, el centroide és el punt mitjà del segment de línia.

Per a un triangle, el centroide és el punt d'intersecció de les seves tres mitjanes.

El centroide d'una figura geomètrica és el centre de simetria. Per a qualsevol altre objecte de forma irregular de dues dimensions, el centroide és el punt on un suport simple pot equilibrar aquest objecte. En general, el centroide d'un objecte bidimensional o tridimensional es troba utilitzant integrals dobles o triples.

4.2.4 Funció *pixelAMm* *[(posició píxels eix x, posició píxels eix y] = [posició mm eix x, posició mm eix y])*

Finalment, aquesta funció és l'encarregada de passar les coordenades de la posició de l'objecte de píxels a mil·límetres.

Per tal que aquesta conversió sigui possible ha d'estar ben escalat. Per aquest motiu fa falta que hagi passat per un procés de calibratge, que s'explicarà amb més detall en el següent apartat.

4.3 Calibratge

El calibratge és un tema a tenir present, ja que un mal ajustament pot influir en una mala visualització de la càmera i per tant una mala execució del programa, i lògicament una representació totalment errònia.

El primer calibratge a tenir en compte és el que fa referència a la relació dels dos servomotors amb la plataforma, ja que aquesta tant en l'inici de l'execució del programa com en el final, s'ha de mantenir completament horitzontal. Això s'aconsegueix, en el nostres cas, col·locant els servos a la posició de 90º. Com que les ròtules tenen un petit joc, s'ha d'ajustar manualment, situant els servos, fins a assolir la posició idònia en la plataforma.

El segon calibratge és el que fa referència a la funció *pixelAMm.* Tal com hem dit anteriorment la funció de vídeo utilitzada ens dóna una resposta en píxels i el que volem saber és el seu posicionament en mil·límetres, per tant, hem d'establir una relació entre aquestes dues unitats de mesura. Primer de tot cal tenir present les mesures de la nostra plataforma i les dimensions de les fotografies realitzades per la càmera.

Aquestes dimensions són:

Plataforma

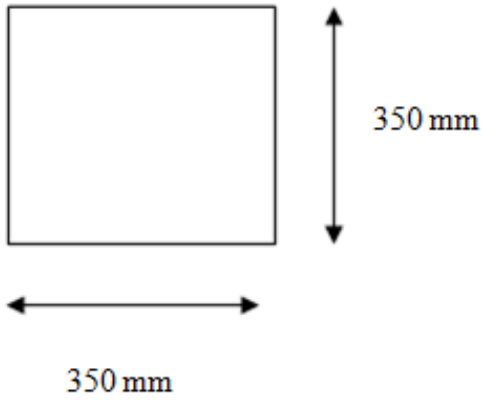


Figura 4.12 Dimensió de la plataforma

Dimensió fotografia

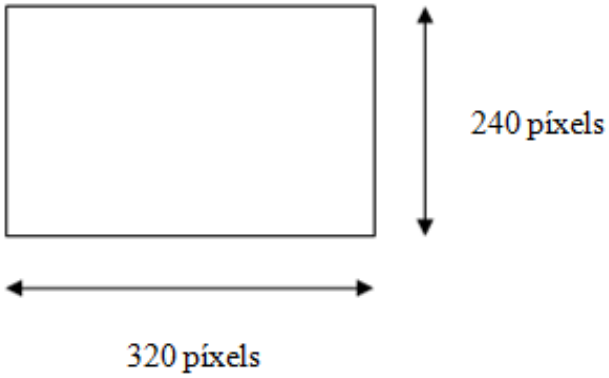


Figura 4.13 Dimensió de la fotografia

Tal i com es pot veure en la *figura 4.5*, la plataforma queda completament ajustada amb la càmera web per l'eix de les *y*, per tant, la relació serà a partir d'aquest eix. Llavors ens quedaria de la següent manera:

Relació = 350/240 = 1.45

Però, tot i això,aquesta escala no estarà del tot ajustada ja que la càmera es menja 10 mm per banda de l'eix de les *y*. El que s'ha de fer es restar-li aquests 20mm totals i tornar a fer la operació anterior.

Relació = (350-20)/240 = 1.375

Un cop tenim ben ajustada aquesta relació, ens queda determinar on es trobarà el centre de la nostre plataforma. Això es podrà fer de dues maneres: o bé col·locant l'objecte al mig o bé deixant marcat un punt negre en ell.

A partir d'aquí, a l'algorisme *trobarPeça* hi ha una funció on ens determina el centroide d'aquesta regió en píxels.

```
regio = regionprops(L, 'Centroid');
```

Finalment, tenint el centroide i l'escalat correctament, ens quedarà realitzar aquests càlculs finals (*veure funció pixelAMm*):

```
(1)   x=-(x'-xMig);
      y=(y'-yMig);

(2)   xloc=x*escala;
      yloc=y*escala;
```

La primera expressió (1) és perquè la imatge té el píxel (0,0) a la part superior esquerra, i nosaltres volem que el punt (0,0) sigui al centre de la nostra plataforma. Per tant utilitzem aquesta expressió per desplaçar el centre. Les coordenades *xMig* i *yMig*, són els píxels que fan referència al centre de la plataforma, per tant, seran constants. En el nostre cas són:

```
xMig=173.3854;
yMig=114.7417;
```

Tal i com es pot veure, el centre buscat no coincideix amb el centre de les dimensions de la *figura 4.14* , el qual seria [160 , 120]. Això és degut perquè l'enfocament manual de la càmera web sobre la plataforma no és precís.

Les variables *x'* i *y'* són les coordenades en píxels per on es mourà la peça al llarg de la seva trajectòria. Pertanyen al sistema de referència de la càmera web.

En quan a les variables *x* i *y* són el resultat de tenir l'origen de coordenades en el centre de la plataforma i pertanyen al sistema de referència de la persona. També estan en píxels.

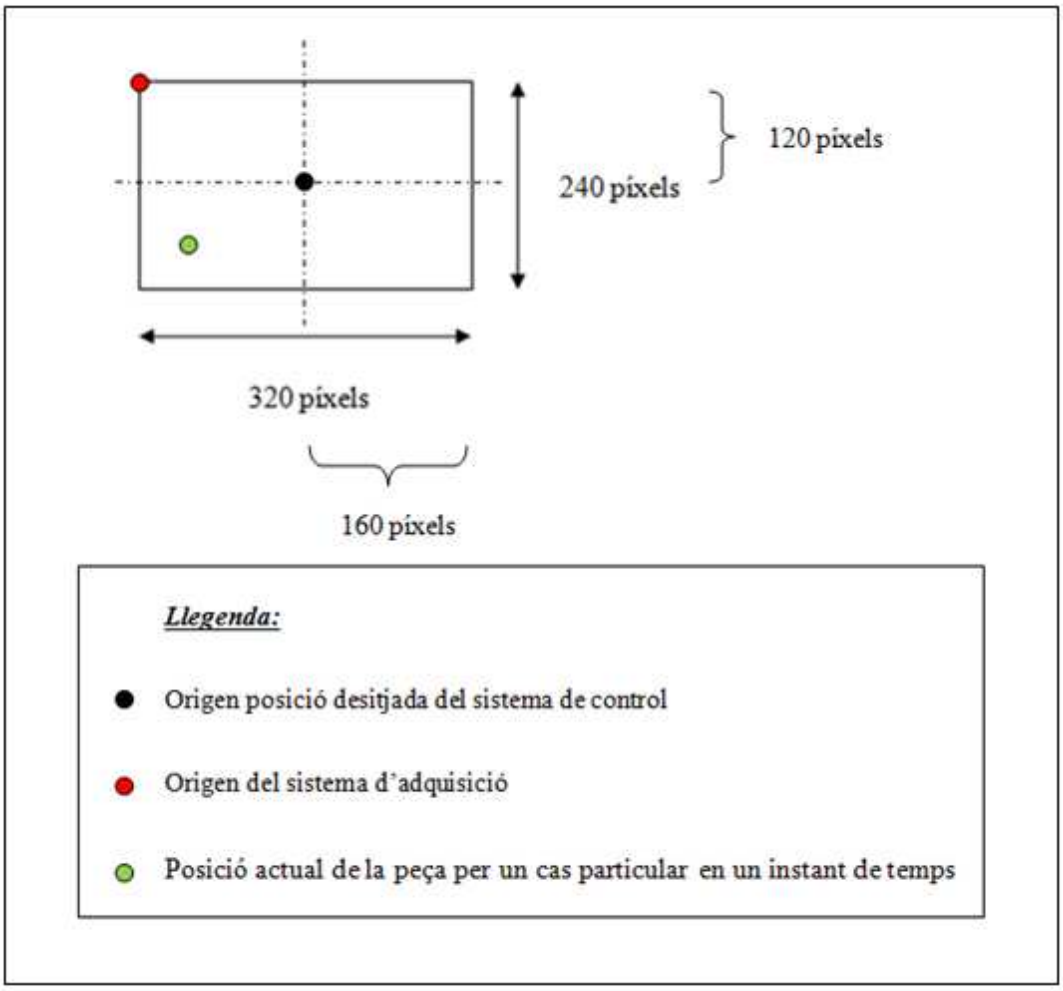


Figura 4.14 Dimensió de la fotografia + posicions

Pel que fa al signe negatiu que hi ha davant del parèntesis de l'expressió *x=-(x'-xMig)*, és perquè el nostre sistema de referència en l'eix de les *x* és en la direcció oposada del sistema de referència de la càmera web. L'eix *y* és manté igual.

A continuació es representa més detalladament:

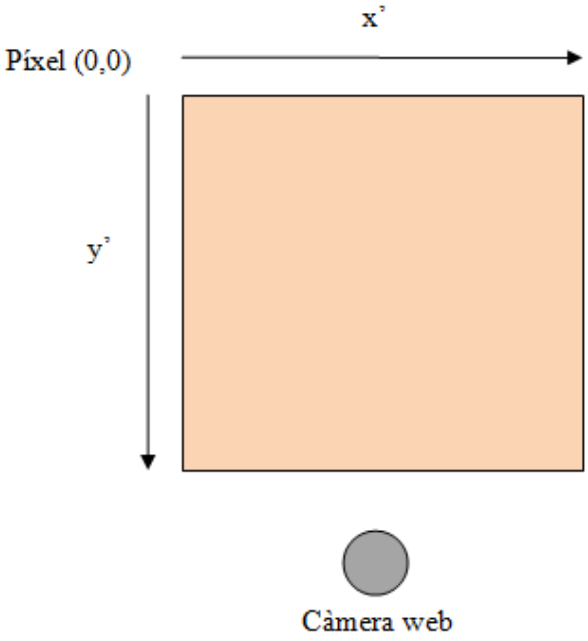


Figura 4.15 Sistema de referència de la càmera web

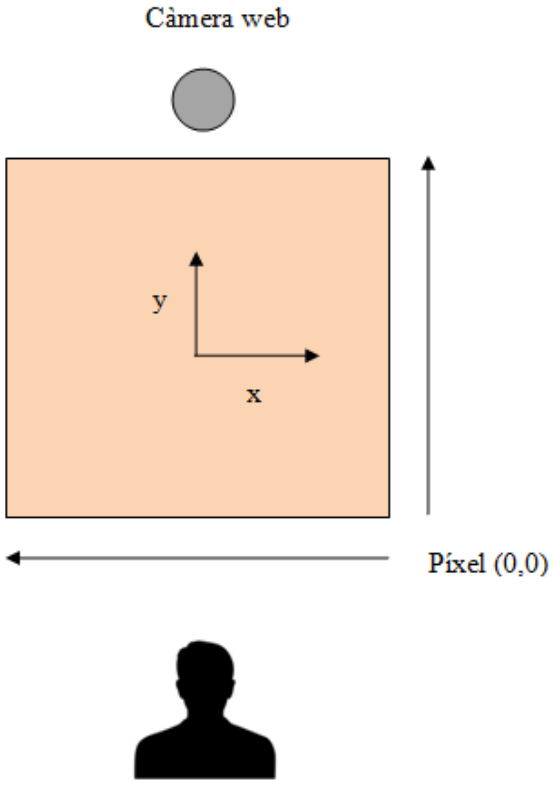


Figura 4.16 Sistema de referència de la persona

Tal i com es pot apreciar a la *figura 4.16* els eixos de les *y* coincideixen, mentre que els eixos de les *x* estan oposats.

La segona expressió (2), és per passar de píxels a mil·límetres fen servir l’escalat trobat anteriorment.

4.4 Resultats

A continuació es fa una representació amb l’objecte:

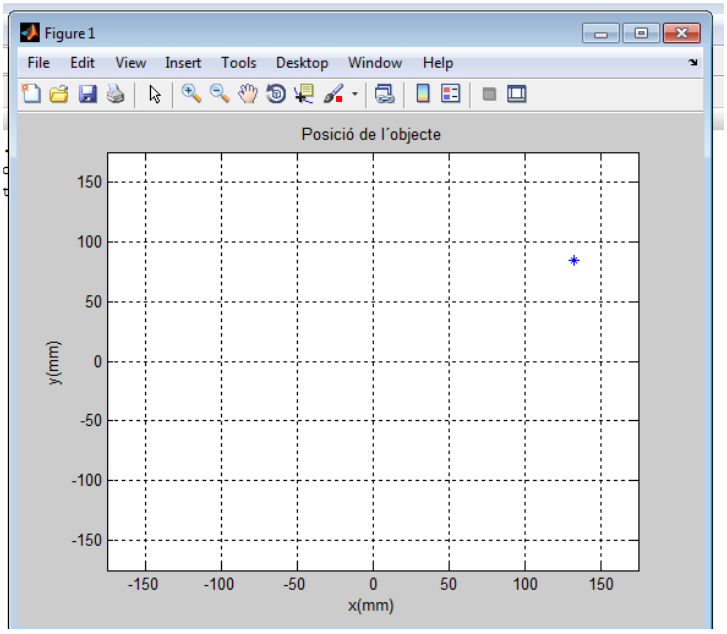


Figura 4.17 Exemple Posició de l’objecte

Tal i com es pot veure a la figura anterior, veiem que la peça es troba situada en un extrem de la plataforma en un instant de temps, ja que aquesta s’anirà movent fins estabilitzar-se al centre.

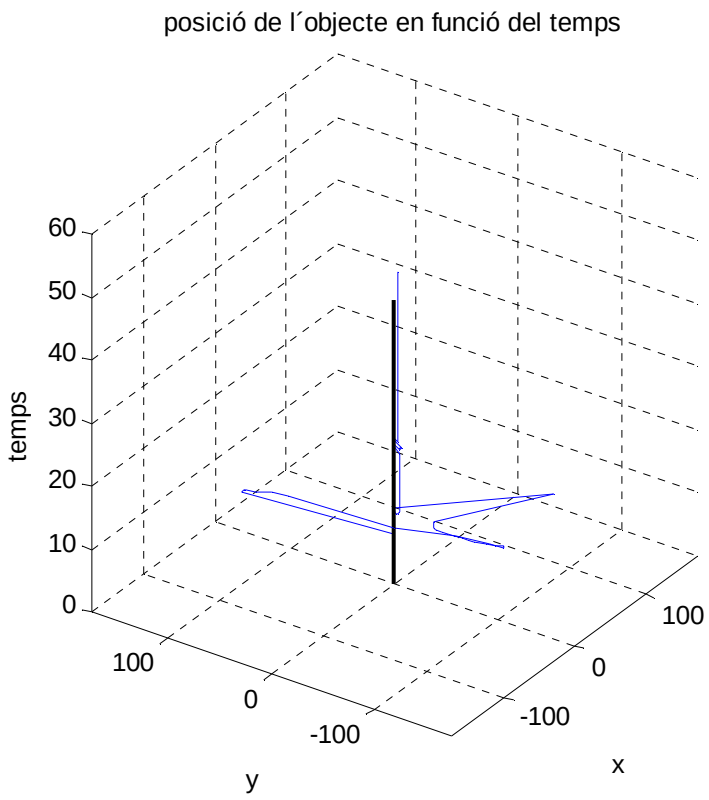


Figura 4.18 Exemple posició de l'objecte en funció del temps

A la *figura 4.18* es pot observar la trajectòria que ha patit l'objecte depenent de la posició inicial on s'ha situat. Sempre es torna a l'origen, és a dir el punt on s'ha establert per defecte.

4.5 Errors

En el nostre cas poden sorgir quatre possibles causes d'error:

1.La primera causa d'error pot venir donada per la poca il·luminació sobre la plataforma, el que pot originar moltes ombres de dimensions pràcticament iguals a la peça i crear confusió a l'algorisme de control.

Si la llum entra molt horitzontalment a la plataforma farà que la peça creï una ombra força considerable i el programa de tractament d'imatges ho interpreti com un únic objecte de dimensions més grans. El problema d'aquest fet vindrà donat a la hora

de trobar el centroide de la peça, el que influirà notablement en el moment de determinar el posicionament d’aquesta.

2. La segona causa d’error és a l’hora de desplaçar l’aparell al complert. Al desmuntar el suport que aguanta la càmera web, pot ser que aquesta es mogui un mica i quan es torni a muntar, el camp de visió amb la plataforma es vegi notablement desplaçat. Això implica ajustar la càmera manualment. Un desavantatge és que la càmera web no porta un programa de zoom instal·lat.

3. Una altre causa d’error vindrà donada quan l’objecte estigui situat molt als límits de la plataforma. Per tal d’evitar possibles ombres amb els límits s’ha utilitzat una funció que fa que els elimini. En cas que passi això, el programa detectarà com si no hi hagués peça i deixarà la plataforma en posició totalment horitzontal.

4. Finalment l’última causa d’error és la diferència de posició de la peça respecte a la càmera web amb l’horitzontal quan la plataforma varia els seus angles. A continuació es pot veure amb més detall:

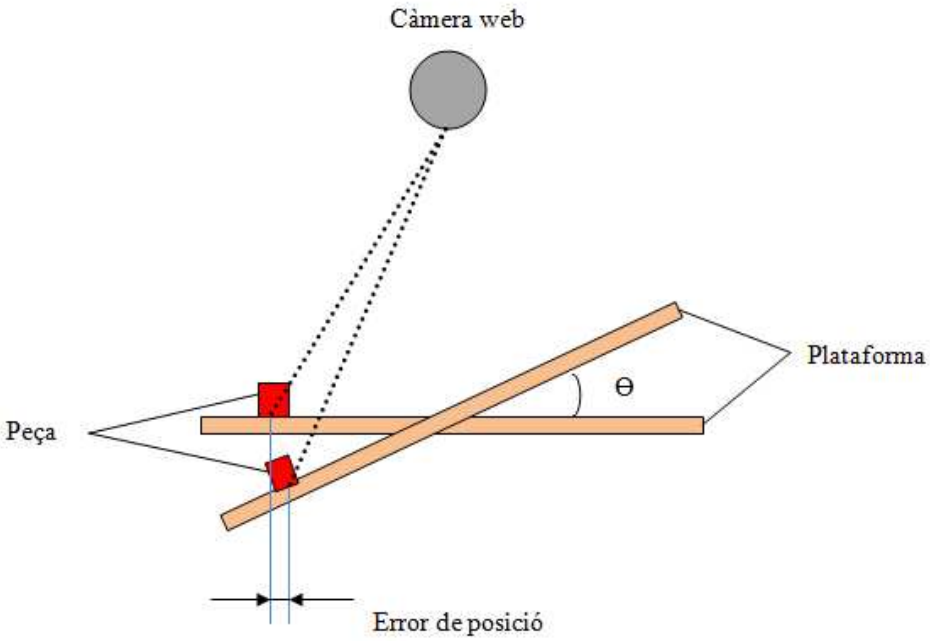


Figura 4.19 Error de posició de la peça respecte la càmera web

Degut a les petites dimensions de la plataforma juntament amb la poca inclinació que pateix, l’error de posició és tan sols d’uns mil·límetres que en aquest cas es consideren menyspreables.

5 PRESSUPOST

PRESSUPOST			
Pressupost part electrònica			
MATERIAL	***QUANTITAT***	***P. UNITAT***	***P. TOTAL***
Placa *Arduino Mega 2560*	1	38,30 €	38,30 €
Càmera web *Logitech V-UBC40*	1	13,50 €	13,50 €
Cable electrònic (90 cm)	1	1,35 €	1,35 €
SUBTOTAL 1			**51,80 €**
Pressupost part mecànica			
MATERIAL	***QUANTITAT***	***P. UNITAT***	***P. TOTAL***
Suport central	1	27,00 €	27,00 €
Ròtules Servos	2	22,00 €	44,00 €
Suports flexors	4	4,70 €	18,80 €
Servomotor *HS-322HD*	2	17,25 €	34,50 €
Junta tòrica (2 cm diàmetre)	2	0,20 €	0,40 €
Plataforma inferior (39x52x1,8 cm)	1	10,00 €	10,00 €
Plataforma superior (35x35x1 cm)	1	6,00 €	6,00 €
Suport càmera web	1	22,00 €	22,00 €
Estructura càmera web	1	16,00 €	16,00 €
Suports plataforma superior	3	1,20 €	3,60 €
Suports fusta servomotors	2	2,00 €	4,00 €
Suport alumini servomotors	2	2,30 €	4,60 €
Brides	5	0,20 €	1,00 €
Perfil alumini en U (182 cm)	1	4,00 €	4,00 €
Visos(3x30mm)	33	0,07 €	2,31 €
Cargols (3x16 mm)	8	0,08 €	0,64 €
Femelles (M3)	8	0,02 €	0,16 €
Peça quadrada	1	1,50 €	1,50 €
Tela verda aironfix	1	4,00 €	4,00 €
Suport goma - plataforma inferior	4	0,16 €	0,64 €
SUBTOTAL 2			**205,15 €**

MATERIAL	*QUANTITAT*	*P. UNITAT*	*P. TOTAL*
Ma d'obra - muntatge	30	12	360
Ma d'obra - programació	6	18	108
SUBTOTAL 3			**468,00 €**

TOTAL ***(SENSE IVA)***	***724,95 €***
IVA (21%)	***152,24 €***
TOTAL	***877,19 €***

6 CONCLUSIONS

El sistema és capaç de trobar la posició de la peça en qualsevol punt de la plataforma sempre i quan no estigui tocant els extrems, i a més, controlar-la. Per trobar-ne la posició, un gran inconvenient ha estat la forta dependència amb la il·luminació, que fa que el calibratge hagi de ser molt precís i, per tant, complicat. Per al control de la posició, l’ impediment fonamental ha estat la baixa resolució de la càmera web, ja que petits errors en la posició de l’objecte es tradueixen en grans correccions per part de l'algorisme de control, el que fa que el sistema perdi l’estabilitat.

L'aplicació construïda, treballa de forma totalment automatitzada. No necessita paràmetres d'entrada (només estar calibrada), i segueix funcionant correctament fins que es para. Si actuem sobre la peça (introduint un moviment imprevisible per al sistema), es recupera sense problema. Encara que aquesta desaparegui (portant al sistema al seu estat inicial, en què la plataforma es col·loca en posició horitzontal) en el moment en què es recupera la visió de la peça, es comença a controlar de nou la seva posició.

7 BIBLIOGRAFIA

Adreces d'Internet

Arduino [en línia]. [Consulta Febrer 2014]. Disponible a: http://www.arduino.cc/es/

Arduino IO package [en línia]. [Consulta Març 2014]. Disponible a: http://www.mathworks.com/matlabcentral/fileexchange/27843-arduino-io-package--slides-and-examples

Arduino:Drivers [en línia]. [Consulta Febrer 2014]. Disponible a: http://arduino.cc/en/Main/Software

QuickCam® Messenger [en línia]. [Consulta Febrer 2014]. Disponible a**:** http://www.logitech.com/es-es/support/3780?crid=405&osid=14&bit=64

HS-322HD Standard Deluxe [en línia]. [Consulta Febrer 2014]. Disponible a: http://www.servocity.com/html/hs-322hd_standard_deluxe.html#.U75sx_l_u4k

El servomotor [en línia]. [Consulta Febrer 2014]. Disponible a: http://www.info-ab.uclm.es/labelec/solar/electronica/elementos/servomotor.htm

Servomotor de modelismo [en línia]. [Consulta Febrer 2014]. Disponible a: http://es.wikipedia.org/wiki/Servomotor_de_modelismo

Sistemas automáticos y de control [en línia]. [Consulta Març 2014]. Disponible a: http://e-ducativa.catedu.es/44700165/aula/archivos/repositorio/4750/4925/html/index.html

Controlador de acción Proporcional(P) [en línia]. [Consulta Març 2014]. Disponible a:

http://e-ducativa.catedu.es/44700165/aula/archivos/repositorio//4750/4926/html/11_controlador_de_accin_proporcional_p.html

José Mª Uriarte Anoro. *Sistemas automáticos y de control* [en línia] IES Sta. Maria de Alarcos, 2010-2011. [Consulta Febrer 2014]. Disponible a: http://ieshuelin.com/huelinwp/download/Tecnologia/Tecnologia%20industrial/3-SISTEMAS-DE-CONTROL-AUTOMaTICO.pdf

Control Automàtic. Tema 4 CIA_2013_2014. 3 de Març 2014. EPSEM. PDF [Consulta Març 2014].

MATLAB [en línia]. [Consulta Gener 2014]. Disponible a: http://es.wikipedia.org/wiki/MATLAB

MathWorks [en línia]. [Consulta Gener 201]. Disponible a: http://www.mathworks.es/products/?s_tid=gn_ps

ANNEXES

ÍNDEX ANNEXES

Pàg.

A ESTRUCTURA DEL MUNTATGE, COMPONENTS I CONNEXIONAT

A.1 Descripció del sistema mecànic

A.1.1 Pla inclinat i peça

Durant el disseny de l'aplicació, s’han utilitzat materials que no fossin molt pesats. Així els servomotors es poden moure més fàcilment, ja que han de suportar el pes de la plataforma, les bieles i les ròtules. Però tot i així, han de ser el suficientment robustos per no doblegar-se ni bombar-se, ja que l'aplicació estarà sotmesa a moviments (possiblement) bruscos. A més, si la superfície no és llisa o està bombada, l'algorisme de control funcionarà pitjor: no controlarem la inclinació de la plataforma de manera uniforme en tots els punts de la seva superfície, sinó que serà depenent de la forma en què estigui deformada en cada punt.

A més, el pla no ha de brillar massa. Això és així perquè, en cas de que la peça vagi directament sobre la superfície, la part encarregada de fer les captures i el processat de les imatges ha de ser capaç de discernir el que és peça del que no, incloent els possibles brillantors dels focus sobre la superfície.

Veiem un detall dels diferents components:

Figura A.1: Plataforma i peça (vista des de la càmera web)

El material escollit per a la fabricació física del pla és un quadrat de fusta(pollancre) d'1 cm de gruix i 35 cm de costat amb un pes de 0,490 Kg. Utilitzem

ròtules i bieles d’acer i alumini, una per cada servomotor, juntament amb un eix central amb ròtula inclosa. L’eix central té una longitud de 29 cm amb un pes de 0,270 Kg, mentre que els eixos dels servos tenen una longitud 24 cm amb un pes de 0,022 Kg i 0,092 Kg respectivament (la diferència de pes és deguda a la fabricació artesanal que es va fer ja que una superfície és més gruixuda que l’altre).

Figura A.2: Eixos dels servomotors

Figura A.3: Eix central

Per la unió de la plataforma amb els eixos s’han utilitzat uns suports de fusta fets a mida.

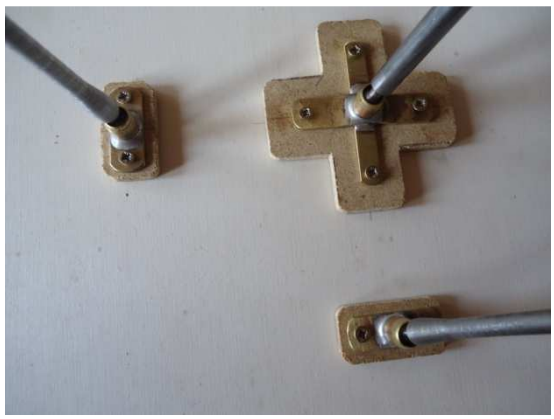


Figura A.4: Suports

Per tal de mantenir els eixos dels servomotors perpendiculars a l’eix central, s’han utilitzat uns suports flexors.

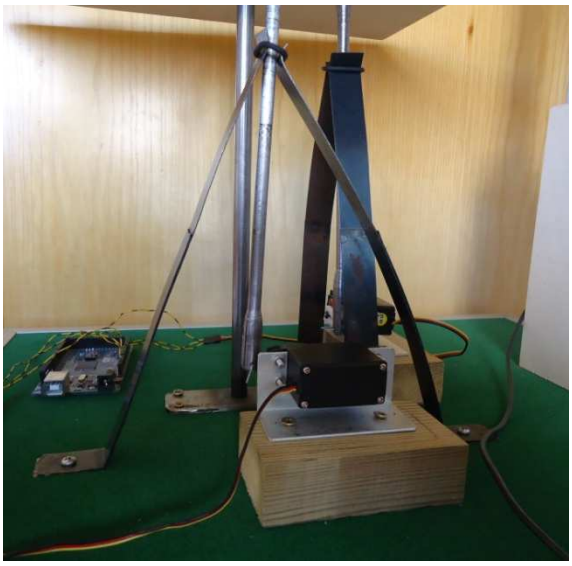


Figura A.5: Suports flexors

En quant als dos servomotors, estan col·locats sobre un suport metàl·lic d’alumini i aquest sobre una fusta per tal de que la biela del servomotor pugui realitzar tot el moviment sense que hi hagi cap objecte que l’obstrueixi (en aquest cas la plataforma inferior formada també per fusta que serveix de base dels mecanismes).

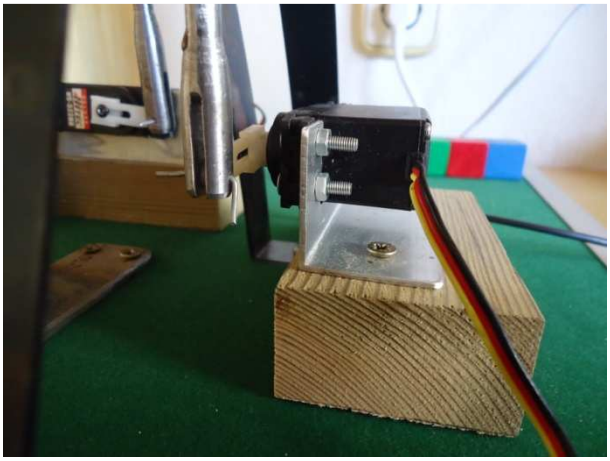


Figura A.6:Suports servomotors

La peça utilitzada és de fusta d’uns 12 grams de pes amb unes dimensions de 25x25mm.

Figura A.7: Peça

A.1.2 Estructura suport de la càmera web

L'estructura que ens permet penjar la càmera web s'ha d'aixecar al costat del pla. Ha de ser el més senzilla possible, perquè es pugui accedir fàcilment al pla i als servomotors, però ha de tenir la suficient rigidesa com per aguantar el pes de la càmera. Per mantenir amb més seguretat l’estructura s’ha aplicat un suport de fusta.

Figura A.8: Suport estructura de la càmera web

La càmera web està situada a 105 cm respecte la base del mecanisme.

Figura A.9: Estructura suport de la càmera web

A.1.3 Estructura completa

L’estructura final, amb unes dimensions de base de 51.5x37.5cm (suport de tots els materials i components que formen part del mecanisme) i una alçada de 110cm, queda representada en la següent figura:

Figura A.10: Estructura completa de la plataforma estabilitzadora

Angle de visió de la càmera web:

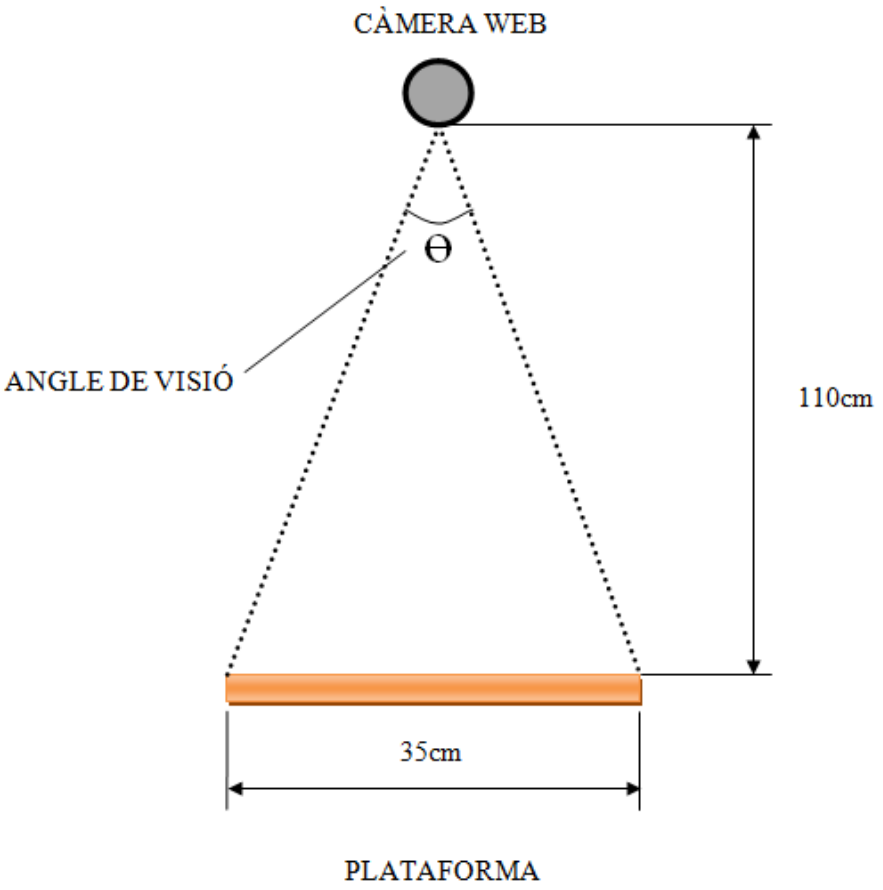


Figura A.11 Angle de visió de la càmera web

A partir de les dades esmentades anteriorment de la mida de la plataforma i la distància d'aquesta respecte a la càmera web, podem determinar l'angle de visió aplicant trigonometria:

$$\theta = \left(arctg \frac{\frac{x}{2}}{L}\right) \cdot 2$$

$$\theta = (arctg \frac{\frac{35}{2}}{110}) \cdot 2$$

$$\theta = 18°$$

Per tant la nostre càmera web té un angle de visió de 18°.

A.1.4 Servomotors

El servomotor utilitzat ha estat el *hitec hs-322hd*. La força que mourà la nostra plataforma la proporcionen dos servomotors iguals als que s'utilitzen en modelisme i ràdio control per fer girar la direcció dels cotxes i per moure el timó dels avions. Gràcies al modelisme podem disposar d'aquests motors a un preu relativament assequible.

Figura A.11: Servomotor hitec hs-322hd

Els servos de radiocontrol són un tipus especial de motor que es caracteritzen per la seva capacitat per posicionar-se ràpidament en qualsevol posició dins del seu rang d'operació. Per a això, el servo espera un tren de polsos que corresponen amb el moviment a realitzar. Estan generalment formats per un amplificador, un motor, la reducció d'engranatge i la realimentació, tot en una mateixa caixa de petites dimensions. El resultat és un servo de posició amb un marge d'operació de 180 ° aproximadament.

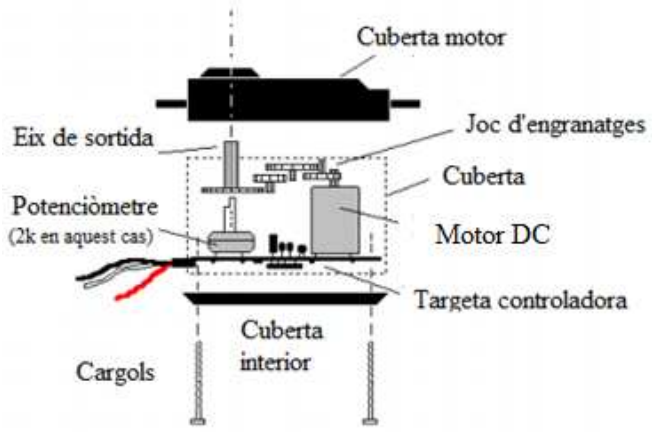


Figura A.12: Detall de l’estructura interna d’un servo

- El *potenciòmetre* és una resistència variable que està directament connectada a l'eix central del servo, per tant permet saber la posició de gir actual al circuit controlador.

- El *motor DC* és l'encarregat del gir dels engranatges i ho fa a la velocitat màxima que li permeti el corrent subministrat. El sentit de gir ho determina el circuit controlador invertint la polaritat de la tensió aplicada.

- Els *engranatges reductors* s'encarreguen de reduir la velocitat i proporcionar més força o parell de gir.

- El *circuit controlador* s'encarrega del control de la posició del servomotor. Rep un senyal de control que representa la posició desitjada i aplica corrent al motor DC fins que el servo arriba a certa posició. Aquesta posició és coneguda a través del potenciòmetre.

El motor elèctric en miniatura genera la magnitud que s'ha de controlar: el gir i posicionament de l'eix del motor. A la vegada, el moviment de rotació angular del motor modifica la posició d'un potenciòmetre intern que controla l’amplada dels polsos d'un monoestable també integrat en el servo.

L'eix del motor pot ser girat fins a una posició angular s’especifica mitjançant el senyal de control. Mentre es mantingui aquest senyal de control, la posició de l'eix del motor es mantindrà. Si el senyal de control canvia, també canvia la posició angular de l'eix.

El senyal de control és un pols modulat PWM que per a la gran majoria de fabricants de servomotor ha de tenir una freqüència de 50 Hz, el que equival a enviar un pols de control cada 20 ms.

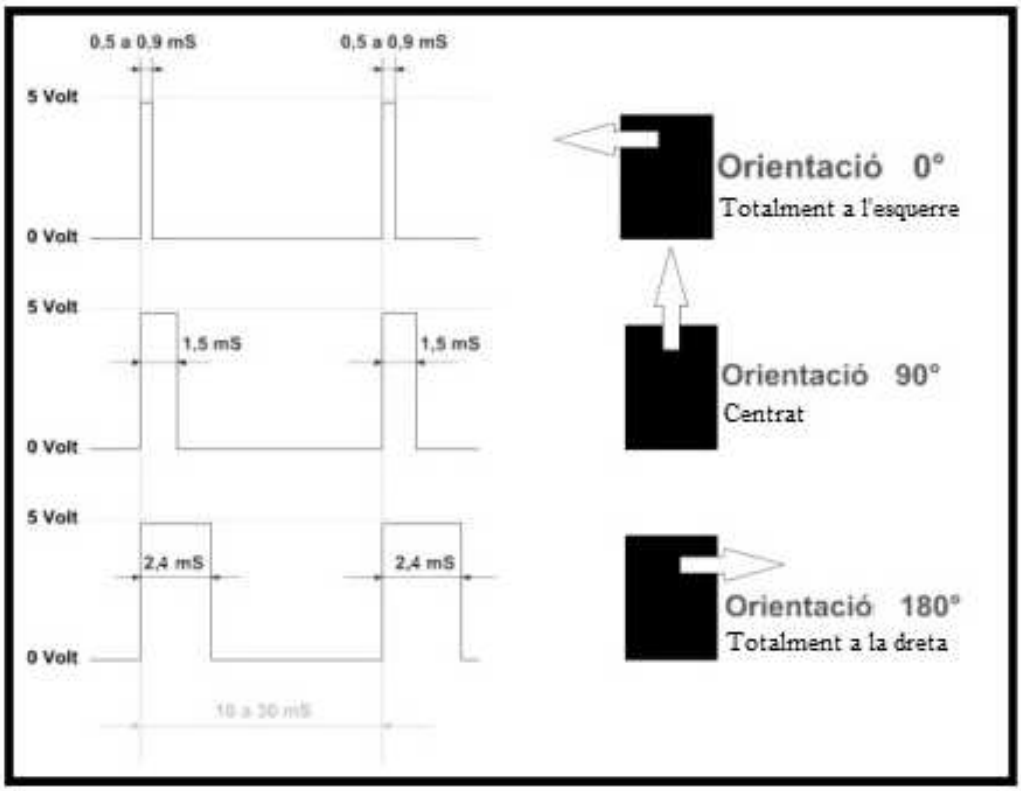


Figura A.13: Modulació PWM de la senyal de control

Durant els 20ms que dura cada repetició s'emet un pols d'amplada variable que depèn directament de la posició angular desitjada. L'ample del pols va de 600µs a 2400µs amb el punt central al 1500µs, encara que cada motor té les seves limitacions dins d'aquest rang. Per a 600µs es configuren 0 º i per 2400µs els 180 º que arriba al servo, per tant treballa utilitzant tot el rang possible.

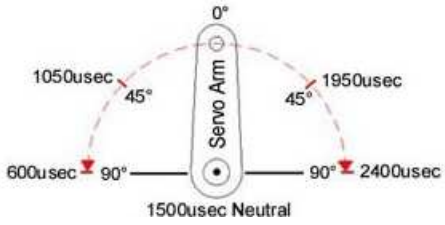


Figura A.14: Ample de pols

Respecte a les limitacions de gir, solen venir donades pels límits del potenciòmetre o per límits mecànics construïts al servo. Quan arriba als seus límits normalment sona un brunzit que indica que s'està forçant.

El servomotor treballa comparant l'amplada del pols d'entrada amb l’amplada del pols generat pel rellotge o timer intern, que és controlada pel potenciòmetre connectat a l'eix del servo. La lògica del servo s'encarrega de determinar la direcció en què ha de girar el motor per minimitzar l'error entre les amplades d'ambdós polsos, el d'entrada i l'intern.

Quan arriba el següent pols es torna a fer la comparació, comprovant de forma contínua la posició de l'eix, i realitzant les correccions necessàries en la posició d'aquest.

En la següent figura podem veure esquemàticament la realimentació de la posició del motor, comparació amb la posició d'entrada i correcció de l'error mitjançant un filtre PID:

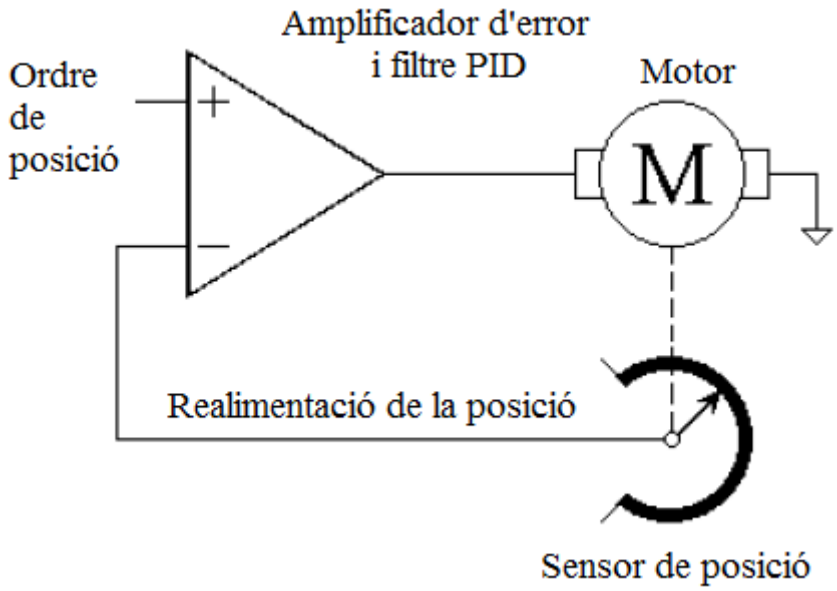


Figura A.15:Operació de posicionament del servo

La precisió en posicionar depèn tant de la precisió del potenciòmetre com de la precisió de l'amplada dels polsos que arriben al motor. La majoria dels servomotors aconsegueixen una resolució de 0.5 graus.

Quan el senyal d'error cau per sota d'un determinat llindar (aproximadament uns 5 microsegons), l'eix del servo es troba a la posició correcta. En aquest moment el

corrent del motor s'apaga. És a dir, quan el servo està en aquest rang conegut com a zona morta o banda de guarda (guard band), el servo apaga els drivers del motor.

Si el senyal d'error no està per sota del llindar, l'electrònica interna seguirà intentant cancel·lar el minúscul error, fent girar el motor enrere o endavant en un moviment anomenat "hunting". L'electrònica interna té com a objectiu fer l’amplada dels polsos del monoestable igual a l'amplada dels polsos d'entrada.

Com que hi ha una relació fixa entre l'angle de rotació del potenciòmetre i l'amplada del pols intern, podem controlar directament l'angle de rotació del servo mitjançant l'amplada dels polsos aplicats.

A continuació podem veure les característiques dels servomotors en concret que hem fet servir en la fabricació del sistema:

- *Sistema de control*: + ample de pols de control 1500usec Neutral
- *Pols requerit:* 3-5 volts pic a pic d'ona quadrada
- *Tensió de funcionament:* 4,8-6,0 volts
- *Rang de temperatura de funcionament:* -20 a +60 graus C
- *Velocitat de funcionament (4.8V):* 0.19sec/60 ° sense càrrega
- *Velocitat de funcionament (6.0V):* 0.15sec/60 ° sense càrrega
- *Lloc de Torque (4.8V):* 42 oz / in (3,0 kg / cm)
- *Parell màxim (6.0V):* 51 oz / in (3,7 kg / cm)
- *Consum de corrent (4.8V):* 7.4mA/idle i 160mA cap operació de càrrega
- *Consum de corrent (6.0V):* 7.7mA/idle i 180mA cap operació de càrrega
- *Amplada de banda morta:* 5usec
- *Angle de funcionament:* 40 Deg. un pols banda viatja 400usec
- *Direcció:* A la dreta / Pols Viatjant 1500 a 1900usec
- *Tipus de motor:* Tubular metall del raspall
- *Potenciòmetre Drive:* 4 Control lliscant / Direct Drive
- *Tipus de coixinet:* Top / Resina boixa
- *Tipus d'art:* Karbonite
- *Modificable contínua Rotació:* Sí

- *Connector Longitud del cable:* 300 mm)
- *Pes:* 43 g

A.2 Descripció del sistema electrònic

A.2.1 Càmera web

La càmera web utilitzada és de la casa *Logitech* el model *V-UBC40*

Figura A.15: Càmera web Logitech V-UBC40

A.2.2 Microcontrolador: *Arduino Mega 2560*

La placa controladora Arduino està formada per una plataforma electrònica de maquinari lliure basada en programari i maquinari flexible fàcil d'utilitzar. Disposa de múltiples entrades i sortides analògiques i digitals configurables a través del llenguatge de programació Processing / Wiring.

Arduino es pot utilitzar per desenvolupar objectes interactius autònoms o pot ser connectat a programari de l'ordinador com ara *MATLAB*. Les plaques es poden muntar a mà o adquirir, i l'entorn de desenvolupament integrat lliure es descarrega gratuïtament.

Hi ha diversos models d'Arduino però les plaques més noves i econòmiques són la *Arduino Uno* i la *Arduino Mega*. Ambdues s'utilitzen de forma similar però la *Mega* és més gran i potent, permetent configurar més perifèrics d'entrada i sortida.

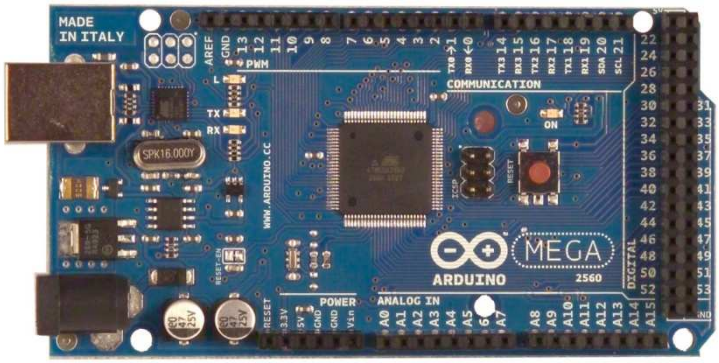


Figura A.16: Arduino Mega 2560

El Arduino Mega 2560 és una placa electrònica basada en el microprocessador Atmega2560. Compta amb 54 pins digitals d'entrada / sortida (dels quals 15 es poden utilitzar com a sortides PWM), 16 entrades analògiques, 4 UARTS (portes serials), un oscil·lador de 16MHz, una connexió USB, un connector d'alimentació, 1 header ICSP, i un botó de reinici.

Característiques de la placa:

- *Microcontrolador ATmega1280*
- *Voltatge de funcionament:* 5V
- *Voltatge d'entrada (recomanat):* 7-12V
- *Voltatge d'entrada (límit):* 6-20V
- *Pins I / S digitals:* 54 (14 proporcionen sortida PWM)
- *Pins d'entrada analògica:* 16
- *Intensitat per pin:* 40mA
- *Intensitat en pin 3.3V:* 50mA
- *Memòria Flash:* 128 KB de les quals 4 KB les utilitza el gestor d'arrencada (bootloader)
- *SRAM:* 8kB
- *EEPROM:* 4kB
- *Velocitat de rellotge:* 16MHz

Com a detall a tenir en compte es pot observar que la placa d’arduino treballa amb un voltatge de 5V (des de connector USB) i cada pin pot rebre o proporcionar una

intensitat de 40mA, mentre que els servos treballant a 4,8V tenen un consum de corrent de 160mA.

Alimentació

L’arduino pot ser alimentat a través de la connexió USB o amb una font d'alimentació externa. La targeta pot funcionar amb un subministrament extern de 6 a 20 volts. Si es subministra amb menys de 7 V, no obstant, el pin de 5V pot subministrar menys de cinc volts i el tauler pot ser inestable. Si s'utilitza més de 12V, el regulador de voltatge es pot sobreescalfar i danyar la placa. El rang recomanat és de 7 a 12 volts.

Els pins d'alimentació són:

- **VIN:** El voltatge d'entrada a la placa Arduino quan es tracta d'utilitzar una font d'alimentació externa (per oposició a 5 volts de la connexió USB o d'una altra font d'energia regulada).
- **5V:** Aquest pin de sortida regula uns 5V des del regulador en el tauler. El tauler pot ser alimentat ja sigui des de la presa d'alimentació de CC (7 - 12), el connector USB (5V), o el pin VIN del tauler (7-12V). El subministrament de tensió a través dels pins de 5V o 3.3V no passa pel regulador, i pot malmetre el tauler. No és aconsellable
- **3V3:** Un subministrament de 3,3 volts generada pel regulador de bord. El drenatge actual màxim és de 50 mA.
- **GND:** Pins de terra.
- **Instrucció IOREF:** Aquest pin de la placa Arduino proporciona la referència de tensió amb la qual opera el microcontrolador. Un escut configurat pot llegir el voltatge pin instrucció IOREF i seleccionar la font d'alimentació adequada o habilitar traductors de voltatge en les sortides per treballar amb el 5V o 3.3V.

Memòria

El Atmega2560 té 256 KB de memòria flash per emmagatzemar codi (dels quals 8 KB s'utilitza per el carregador), 8 KB de SRAM i 4 KB de EEPROM.

Entrades i sortides

Cadascun dels 54 pins digitals en el *Mega* es pot utilitzar com una entrada o sortida. Operen a 5 volts. Cada pin pot proporcionar o rebre un màxim de 40 mA i té una resistència de pull-up (desconnectada per defecte) de 20-50 kOhms.

- **De Sèrie: 0 (RX) i 1 (TX); Sèrie 1: 19 (RX) i 18 (TX); Sèrie 2: 17 (RX) i 16 (TX); Sèrie 3: 15 (RX) i 14 (TX).** S'utilitza per rebre (RX) i transmetre dades en sèrie (TX) TTL. Pins 0 i 1 estan també connectats als pins corresponents del ATmega16U2 USB-to-TTL xips Serial.
- **Interrupcions externes: 2 (interrompre 0), 3 (alarma 1), 18 (interrupció 5), 19 (interrupció 4), 20 (interrompre 3), i 21 (2) d'interrupció**: Aquests pins poden ser configurats per activar una interrupció en un valor baix , un flanc ascendent o descendent, o un canvi en el valor.
- **PWM:** 2 a 13 i 44 a 46 per a sortides PWM de 8 bits .
- **SPI:** 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).
- **LED:** Hi ha un LED predefinit connectat al pin digital 13. Quan el pin és d'alt valor, el LED està encès, quan el passador és baix, està apagat.
- **TWI:** 20 (SDA) i 21 (SCL). Donar suport a la comunicació TWI.

Comunicació

El *Arduino Mega2560* té una sèrie d'instal·lacions per a la comunicació amb un ordinador, un altre Arduino, o altres microcontroladors. La comunicació es per el port COM. Els RX i TX LED al tauler parpellegen quan s'estan transmetent dades a través de la ATmega8U2 / ATmega16U2 xip a l'ordinador per connexió USB (però no per a la comunicació en sèrie en els pins 0 i 1.

Protecció de sobrecorrent USB

El Arduino Mega2560 té un polifusible resetejable que protegeix els ports USB de l’ordinador de curtcircuits i sobrecorrent. Encara que la majoria dels ordinadors proporcionen la seva pròpia protecció interna, el fusible proporciona una capa addicional de protecció. Si hi ha més de 500 mA al port USB, el fusible trenca automàticament la connexió fins que es talla o s'elimina la sobrecàrrega.

A.3 Connexionat

Tal i com es pot veure en la figura, l’esquema del connexionat per al control de l’estructura és el següent:

Figura A.17: Connexionat control plataforma

A.3.1 Connexionat del PC a la càmera web

Per establir la connexió entre ells és fa mitjançant un port sèrie (*USB*).

A.3.2 Connexionat del PC a la placa d’Arduino Mega 2560

En aquesta connexió també utilitzem el port sèrie (en aquest cas el *‘COM3’*), per mitjà d’un cable *USB* amb connectors del tipus *A* i *B.*

Figura A.18: Connectors tipus A i B

A.3.3 Connexionat de la placa d'*Arduino Mega 2560* als servomotors *HS-322HD*

Primer de tot cal definir les connexions de cada component per separat:

Placa Arduino Mega 2560

A continuació les parts ven diferenciades de les placa:

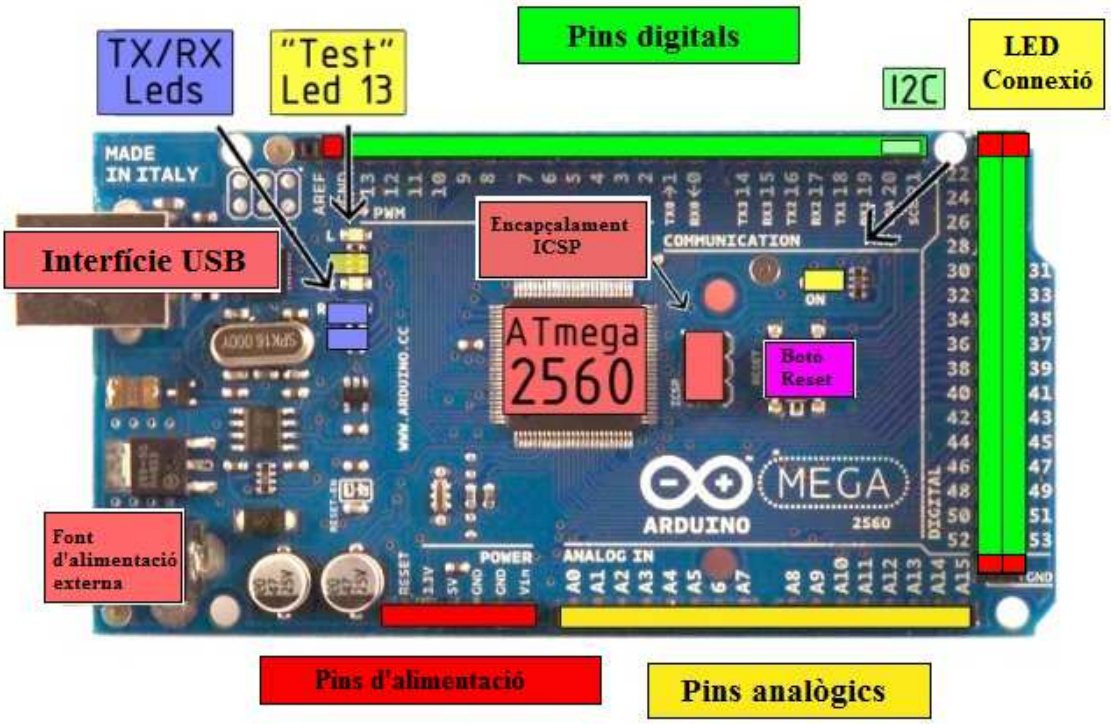
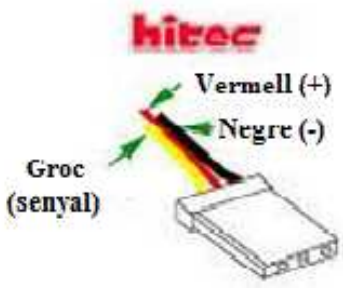


Figura A.19: Parts placa Arduino Mega 2560

Servomotor HS-322HD

Disposen de tres connexions elèctriques: Vcc (vermella), GND (negra) i entrada de control (groga). Aquests colors d'identificació i l'ordre de les connexions depenen del fabricant del servo. És important identificar les connexions ja que un voltatge de polaritat contrària podria danyar el servo.



FiguraA.20: Cables connexió servomotor HS-322HD

Un cop definides les connexions dels dos components ja podem mostrar a continuació com quedarien connectats entre ells en el nostre cas concret:

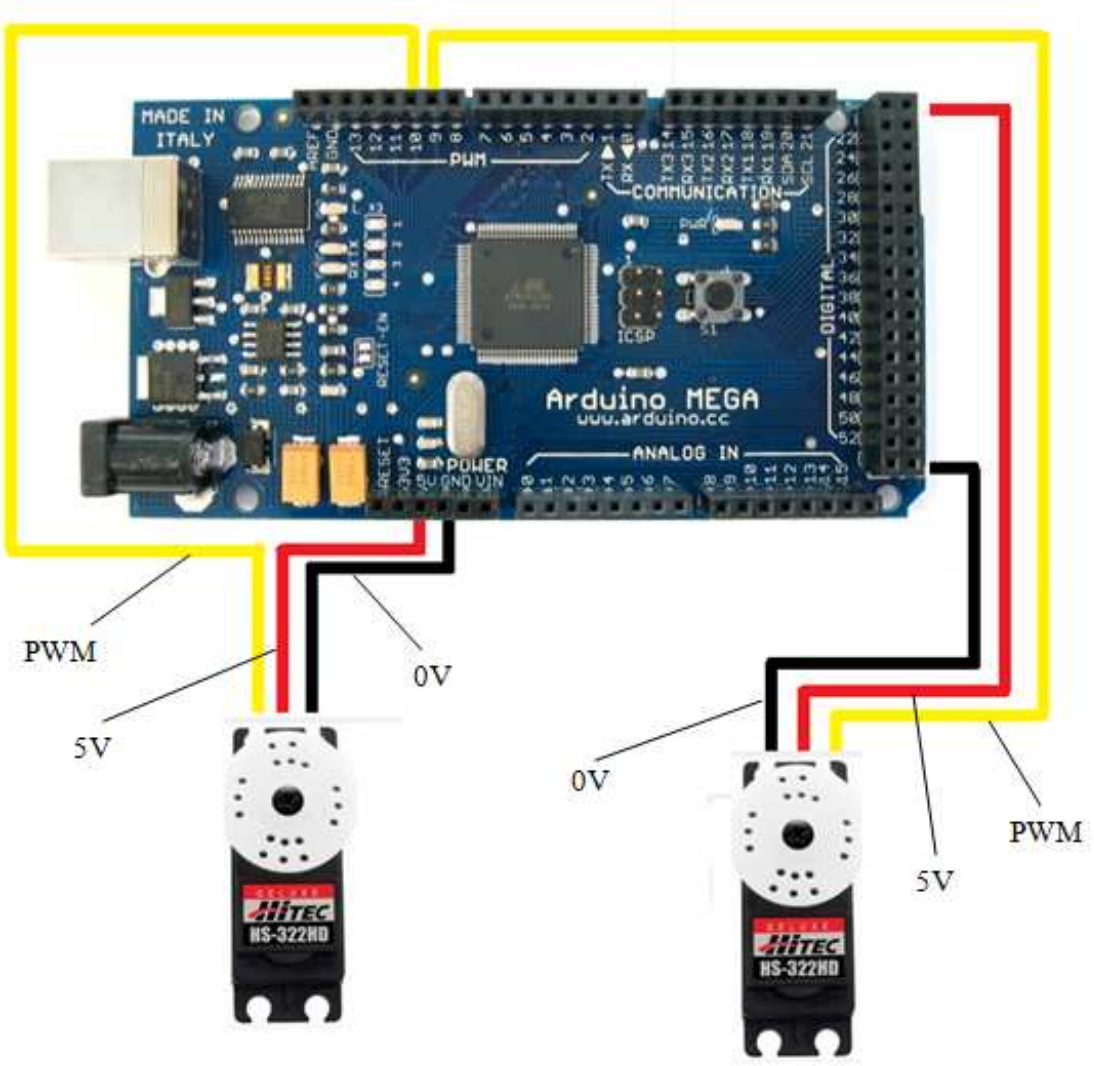


Figura A.21: Connexionat entre Servos HS-322HD i placa Arduino Mega 2560

B ALGORISME DE CONTROL: CODI MATLAB

B.1 Codi *principal*

```
clear all   % Borrar tot
close all   % Tancar tot
clc         % Netejar pantalla

%variables
global t0; % Temps d'arrencada inicial de rellotge
global mitjanaLoc;
global loc;
global contador;
global posDesitjada;
global radius;
global Kp;
global a;


Kp=0.010;    % Constant proporcional (º/mm)
totalTime=40;    % Temps d'execució


% Posició final desitjada de la peça (en mil·límetres)
posDesitjada(1,1)=0; % x posició desitjada
posDesitjada(2,1)=0; % y posició desitjada

%%
%**********************

a=arduino('COM5'); % Establir comunicació amb l'arduino
a.servoAttach(31);  % Activar els servos
a.servoAttach(33);

% Càmera
vid=videoinput('winvideo'); % Activar la càmera
vid.ReturnedColorSpace='grayscale'; % Passar el vídeo a escala de
grisos

% La càmera va realitzant frames
set(vid,'FramesPerTrigger', Inf);
set(vid,'FramesAcquiredFcnCount', 1);
set(vid,'FramesAcquiredFcn', {'control'});

disp('Inici');
%% Inicialitza variables
% Variables per emmagatzemar dades

loc=[];  % Matrius de zeros
mitjanaLoc=[];

contador=0; % S'utilitza per comptar el nombre de vegades que es diu
el control
```

```
%% Comença el programa
start(vid);
t0=clock;
pause(totalTime);
stop(vid);
disp('Fet');

% Gràfica de la posició de l'objecte en funció del temps

figure(2)
plot3(loc(1,:),loc(2,:),loc(3,:))
hold on
plot3([0 0],[0 0],[0 45],'k','LineWidth',2)
title('posició de l´objecte en funció del temps')
xlabel('x')
ylabel('y')
zlabel('temps')
axis([-175 175,-175 175])
grid on



delete (vid)
clear vid
```

B.2 Codi *control*

```
function control(obj, event)
% Funciona sempre que un nou marc s'adquireix des de la webcam
% S'agafa a una imatge i s'obté l'hora
% Amb aquesta informació es decideix quant s'inclina la placa

% Variables
global loc;
global contador;
global t0;
global Kp;
global posDesitjada;
global a;


contador=contador+1; % Nombre d'increments dels temps de control
im=peekdata(obj,1);
[loc(1, contador) loc(2,contador)]=trobarPeca(im);
loc(3,contador)=etime(clock,t0);

% Gràfica de la posició de la peça

figure(1)
plot(loc(1,contador),loc(2,contador),'*')
axis([-175 175,-175 175])
title('Posició de l´objecte')
xlabel('x(mm)')
ylabel('y(mm)')
grid on

if contador==1
    mitjLoc=loc(1:2,contador);

elseif contador==2
    mitjLoc=median(loc(1:2,contador-1:contador),2);

else %contador>=3
    mitjLoc=median(loc(1:2,contador-2:contador),2);

end

% posDesitjada és el vector [0,0]
% mitjLoc és la posició actual de la peça

Kp=0.010;  % constant de proporcionalitat
nouAngle=((posDesitjada(:,1)-mitjLoc(:,1)).*Kp+pi/2)*360/(2*pi);

nouAngle=abs(round(nouAngle-180*fix(nouAngle./180)));
a.servoWrite(31,nouAngle(1));   % S'envia l'angle als servos
a.servoWrite(33,nouAngle(2));


pause(0.05)% La càmera web té un límit de fotogrames per segon,
llavors sense la pausa utilitzariem el mateix fotograma(imatge)
```

B.3 Codi *trobarPeca*

```
function [xloc, yloc]= trobarPeca(Im)
% Pren l'arxiu d'imatge i genera una matriu de 2x1 amb x i y
% Localitza el centre de la peça
% La imatge que entra ja està en escala de grisos

%% Detecció de vores utilitzant el mètode Sobel
[junk threshold] = edge(Im, 'sobel');
fudgeFactor =1;
BWs = edge(Im,'sobel', threshold * fudgeFactor);  % Espessir les
línies

se90 = strel('line', 3, 90);
se0 = strel('line', 3, 0);
BWsdil = imdilate(BWs, [se90 se0]); % Fer les línies més gruixudes
BWdfill = imfill(BWsdil, 'holes'); % Omplir els forats
%%
% Eliminar els objectes que estan tocan a la frontera, llavors els
voltants
% En aquest cas, qualsevol coloració blanca tocant la frontera
% Un possible problema és la peça que es pot moure i tocar la frontera
BWnoborder=imclearborder(BWdfill, 4);
%%
% Erosionar la imatge
seD = strel('diamond',1);
BWseg = imerode(BWnoborder,seD);

BWseg = imerode(BWseg,seD);


%% Tobar el centre de la peça
% Trobar el centroide de les regions
% Troba l'àrea.
% Un problema és que el centre exacte de la peça no pot ser retornat,
% ja que és possible que part de la seva ombra estigui connectada al
% contorn de la peça

L = bwlabel(BWseg);
regio = regionprops(L, 'Centroid');
area=regionprops(L,'Area');
max=0;
referencia=0;
k=175; %límits de la taula en mm a partir de l'origen

for i=1:numel(area)
    if area(i).Area>175 && area(i).Area<5000 && area(i).Area>max
        max=area(i).Area;
        referencia=i;
    end
end
%% *******************************************************
if reference==0 % No troba la peça
    xloc=0;
    yloc=0;
else % Troba la posició de la peça
    [xloc1,yloc1]= ...
```

```
pixelAMm(regio(reference).Centroid(1),regio(reference).Centroid(2));
% Comprova que les coordenades retornades són raonables

    if abs(xloc1)>k
        xloc=0;
    else
        xloc=xloc1;
    end
    if abs(yloc1)>k
        yloc=0;
    else
        yloc=yloc1;
    end
end
```

B.4 Codi *pixelAMm*

```
function [xloc, yloc]= pixelAMm(x,y)
% Entrada: Les x i les y representades en pixels
% Sortides: Les x i les y representades en mil·límetres
% (centre de la placa és el 0,0)

xMig=173.3854; % Centroide de l'origen de la plataforma
yMig=114.7417;

relacio=330/240; % Mm/píxels

x=-(x-xMig); % La càmera està mirant en la posició oposada
y=(y-yMig);

xloc=x*relacio;    % La posició de la peça en mm
yloc=y*relacio;
```

C PAQUET DE SUPORT DE MATLAB PER ADUINO

```
ARDUINO IO (Also Known As: "TETHERED" MATLAB SUPPORT PACKAGE FOR
ARDUINO):

This package allows using an Arduino connected to the computer to
perform
Analog and Digital Input and Output, (and command motors) from MATLAB.

-------------------------------------------------------------------
-----

DETAILS ABOUT ARDUINO:

Arduino is a powerful and inexpensive open-source microcontroller
board,
with an associated development environment: http://arduino.cc/
An extensive Knowledge base can be found here:
http://www.freeduino.org/


BUYING AN ARDUINO BOARD:

An extensive list of sites where you can buy an Arduino is here:
http://www.arduino.cc/en/Main/Buy

In the US, adafruit industries (http://www.adafruit.com/)
provides a starter pack that includes pretty much everything that you
need to get started with the board.

While earlier version of this package were targeted at smaller boards
like
the Uno and Nano, since August 2012 (ver 3.3) the package also works
right
out of the box on the Mega boards, without requiring any Mega-related
tweaking.


GETTING STARTED GUIDES:

The official getting started guide is here :
http://arduino.cc/en/Guide/HomePage
and a very good tutorial is here:
http://www.ladyada.net/learn/arduino/
However note that for the purpose of using this package you only need
to have
the IDE environment installed but you won't need to use it, because
you can
do all the programming in MATLAB.


CHIP KIT 32 BOARDS:

Note that the the package works fine with the ChipKit32 boards (Uno32,
Max32):
http://www.digilentinc.com/Products/Catalog.cfm?NavPath=2,892&Cat=18
```

```
All of the analog and digital input and output functionality work
fine,
but unfortunately not all the interrupts functionalities work in the
same
way, so servos and encoders might not work exactly as on the Arduino
boards.
Since Sep 2012, when the AFMotor library was updated to support PIC32,
the Adafruit Motor Sield also works with these boards.

The Cerebot MX7ck also works fine with the package. Note that in order
to be
accessed with the MPIDE you need to short jumper JP11, set the switch
to ON,
and connect it to the PC through the UART (not the DEBUG) usb port.
If you don't have external power then make sure to also set the power
jumper to UART (as opposite to the default position DBG).


TI LAUNCHPAD (MSP430) BOARDS:

Using the Energia IDE (http://energia.nu/Guide_MSP430LaunchPad.html)
the adio.pde and adioe.pde can be compiled for the MSP430G2553 and
MSP430G2452 (the G2231 does not have enough memory). The adioes.pde
sketch
can be compiled for the FR5739. Note however that no real testing has
been
performed on these platforms though.

------------------------------------------------------------------------
-----

DOWNLOADING AND INSTALLING THE IDE (to be done only once):

A step by step driver installation can be found at:
http://arduino.cc/en/Guide/HomePage and there is no need to duplicate
it here.
It is a good idea to go trought all the 9 steps, although after you
have
installed the drivers, maybe the shield library, and verified that the
IDE
can communicate with the Arduino, you can already start using this
package.


INSTALLING THE ADAFRUIT MOTOR SHIELD LIBRARY (to be done only once):

If you want to use this package with the ADAFRUIT motor shield,
(either
version 1: http://learn.adafruit.com/adafruit-motor-shield or version
2:
http://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino ), you
need
to download and install the respective adafruit motor shield library
(follow instructions given in the above sites, which amount to
download
a zip file and unzip it into the arduino-1.X/libraries folder.

As better explained in the next step, this you will also need to
upload
```

```
eiter the motor_v1.pde or the motor_v2.pde sketch on the board.

NOTE that if you don't have the adafruit motor shield and you don't
plan
to use it, you might also SKIP THIS STEP.

Also note that the "official" Arduino (not Adafruit) motor shield:
(http://arduino.cc/it/Main/ArduinoMotorShieldR3)
does not require any additional library, works fine with the chipkit32
boards, and can be used with this package right away. Therefore if you
plan to use the official motor shield then you can safely skip this
step.

-----------------------------------------------------------------------
-----

UPLOAD ADIOES.PDE TO THE ARDUINO BOARD (to be done only once):

The adioes.pde sketch is the "server" program that will continuously
run
on the microcontroller board. It listens for MATLAB commands arriving
from
the serial port, executes the commands, and, if needed, returns a
result.

The following instructions are needed to upload the adioes.pde file
into
the controller's flash memory. As long as no other file is uploaded
later,
this step does not need to be repeated anymore, and the package can be
used as soon as the board is connected to the computer.

Note that if you want to use the adafruit shield library then you need
to
both install the appropriate library (see previous step) and upload
the
the appropriate sketch (either motor_v1.pde or motor_v2.pde) instead
of
the adioes.pde sketch.

From the Arduino IDE, go to File > Open, locate the file adioes.pde,
(in the ArduinoIO/pde/adioes folder) and open it. If a dialog appears
asking for the permission to create a sketch folder and move the file,
press OK (this will create an adioes folder and move the adioes.pde
file inside it).

Connect the Arduino, make sure that the right board and serial port
are
selected in the IDE, (Tools/Board and Tool/Serial Port) then select
File -> Upload to I/O Board and wait for the "Done Uploading" message.

At this point the adioes.pde file is uploaded and you can close the
IDE,
which is not needed anymore for the purpose of this package. Actually
closing the IDE is suggested, so you can be sure that the serial
connection
to the arduino board is not taken by the IDE when matlab needs to use
it.
```

```
-------------------------------------------------------------------------
-----

PACKAGE INSTALLATION (to be done only once):

When installing the Arduino IO package on your operating system it is
important that users have the right to access the serial port and
modify
the pathdef.m file.


ON LINUX:
---------

To make sure that the pathdef.m file is writable, issue a command like
this:
sudo chmod 777 usr/local/matlab/R2012a/toolbox/local/pathdef.m
(modify the path above according to where MATLAB is installed).

Also, still on Linux, create a symbolic link as follows:
sudo ln -s /dev/ttyACM0 /dev/ttyS101
and make sure it is accessible by any user:
sudo chmod 777 /dev/ttyS101
Forgetting this last step might lead to a serial port which is
unaccessible by a normal user (therefore being grayed-out in the IDE).

Then from MATLAB, launch the "install_arduino" command, this will
simply
add the relevant ArduinoIO folders to the matlab path and save the
path.


ON WINDOWS:
-----------

Run MATLAB as administrator (just one time for the purpose of
installing
the package) by right-clicking on the MATLAB icon and selecting
"Run as Administrator". This will allow the updated path to be saved.

Then from MATLAB, launch the "install_arduino" command, this will
simply
add the relevant ArduinoIO folders to the matlab path and save the
path.

-------------------------------------------------------------------------
-----

TYPICAL USAGE:

Make sure the board is connected to the computer via USB port, make
sure
you know which serial port the Arduino is connected to (this is the
same
port found at the end of the drivers installation step), and finally,
make sure that the port is not used by the IDE (in other words, the
IDE must
be closed or disconnected), so that MATLAB can use the serial
connection.
```

```
From MATLAB, launch the command a=arduino('port') where 'port' is the
COM port to which the Arduino is connected to, e.g. 'COM5' or 'COM8'
on Windows,
or '/dev/ttyS101' on Unix (use '/dev/ttyUSB0' for Arduino versions
prior to Uno)
or 'DEMO' (see below for the DEMO mode) and make sure the function
terminates
successfully.

Then use the commands a.pinMode, a.digitalRead, a.digitalWrite,
a.analogRead,
and a.analogWrite, to respectively change the mode (input/output) of
each pin,
perform digital input, digital output, analog input, and analog
output.
Consult the help of the files to get more information about their
usage.

If you also have a servo motor, then you can use commands such as
a.servoAttach, a.servoStatus, a.servoWrite, a.servoRead, and
a.servoDetach
to respectively attach a servo to a PWM pin, get its status
(attached/detached)
move it to a certain angle, and read its position.

NOTE that since August 2012 (ver 3.8) the servos are no longer
referred by
the numbers 1 and 2 but instead by the PWM pin number to which they
are
attached (e.g. a.servoRead(9) reads the servo attached to pin #9).

For encoders you can use the commands a.encoderAttach,
a.encoderStatus,
a.encoderRead, a.encoderReset and a.encoderDetach to respectively
attach an
encoder to 2 interrupt pins (2,3 on the Uno, 2,3,21,20,19,18 on the
Mega)
get its status (attached/detached), read and reset its position, and
detach it when you are done. The positive rotation directions for
encoders
is assumed to be clockwise, and the range goes from -32768 to +32767,
if you need a bigger range all you have to do is to go in the Encoder
structure typedef line of the sketch file and replace the third field,
"int pos;" with "long int pos;".

There is a couple of other functions such as a.serial (returning the
name
of the serial port), a.flush (flushing the input side of the PC's
serial
port) and a.roundTrip, which sends a value to the arduino and back.

Finally, use a.delete to delete the arduino object, (and free up the
serial port) when the session is over.

Have a look below for an example.

-------------------------------------------------------------------
-----
```

```
EXAMPLE:

% connect the board
a=arduino('COM5')

% specify pin mode for pins 4, 13 and 5
pinMode(a,4,'input');
pinMode(a,13,'output');
pinMode(a,5,'output');

% read digital input from pin 4
dv=digitalRead(a,4);

% output the digital value (0 or 1) to pin 13
digitalWrite(a,13,dv);

% read analog input from analog pin 5 (physically != from digital pin
5)
av=analogRead(a,5);

% normalize av from 0:1023 to 0:254
av=(av/1023)*254;

% ouptput value on digital (pwm) pin 5 (again, different from digital
pin 5)
analogWrite(a,5,round(av))

% change reference voltage for analog pins to external
analogReference(a,'external');

% change it back to default
analogReference(a,'default');

% gets the name of the serial port to which the arduino is connected
to
serial(a)

% flushes the PC's serial input buffer (just in case)
flush(a);

% sends number 42 to the arduino and back (to see if it's still there)
roundTrip(a,42)

% attach servo on pin #9
servoAttach(a,9);

% return the status of all servos
servoStatus(a);

% rotates servo on pin #9 to 100 degrees
servoWrite(a,9,100);

% reads angle from servo on pin #9
val=servoRead(a,9)

% detach servo from pin #9
servoDetach(a,9);
```

```
% attach encoder #0 on pins 3 (pin A) and 2 (pin B)
encoderAttach(a,0,3,2)

% read the encoder position
encoderRead(a,0)

% attach encoder #2 on pins 18 (pin A) and 21 (pin B)
encoderAttach(a,2,18,21)

% sets debouncing delay to 17 (~1.7ms) for encoder #2
encoderDebounce(a,2,17)

% read position or encoder #2
encoderRead(a,2)

% reset position of encoder #2
encoderReset(a,2)

% get status of all three encoders
encoderStatus(a);

% detach encoder #0
encoderDetach(a,0);

% close session
delete(a)

NOTE: Should for any reason the serial port not be relaesed after you
clear
the arduino object, you can use the following commands to force the
release
ot the serial connection:

% delete MATLAB serial connection on COM3
delete(instrfind({'Port'},{'COM3'}));

% delete all MATLAB serial connections
delete(instrfind('Type', 'serial'));

Also note that due to MATLAB OO naming conventions, the first argument
can be passed also before any function (method) name, therefore a
typical
function such as: function(a,arg2,arg3,...) can also be called like
this:
a.function(arg2,arg3,...)

-------------------------------------------------------------------
-----

PROVIDED SKETCHES:

The following sketches are provided with the package:

-) adio.pde     : analog and digital IO, plus basic serial commands
only
-) adioe.pde    : adio.pde + encoders support
-) adioes.pde   : adioe.pde + servo support
```

```
-) motor_v1.pde : adioes.pde + afmotor v1 shield
-) motor_v2.pde : adioes.pde + afmotor v2 shield

In most cases the adioes.pde will cover all you need. The last two
sketches
are needed for the Adafruit Motor Shield (version 1 and 2
respectively).
The first 2 sketches are mostly provided in case your specific
platform
does not support servos or encoders and as a better (simpler) starting
point for people that want to customize a sketch for their own
purposes.
In the latter case, the "roundtrip" function is probably the easiest
place to get started to introduce custom code.

-------------------------------------------------------------------
-----

DEMO MODE:

Whenever 'DEMO' is specified as argument of the arduino function, i.e.
a=arduino('DEMO') then a virtual connection is opened that does not
involve
any hardware. This allows trying the package without actually having
an
arduino board. In this mode, all the "read" functions return random
values
according to their output range. Also, delays are used internally so
that
the execution time of any function approximately matches the average
execution time that can be observed when the actual board is
connected.

Note that the same behavior (random output values) occurs when the
sketch
running on the board does not support the specific operation (e.g. a
servo
read is issued but the sketch running on the board, say adio.pde) does
not
support servos.

-------------------------------------------------------------------
-----

BLINK CHALLENGE:

The "blink_challenge" is an example application that switches the leds
on and
off with variable frequency and with mode changed by pressing a switch
button
Have a look at the m file for more info, (type help blink_challenge
from the
command line) and launch blink_challenge to execute the demo.

NOTE that running this applicaton only makes sense if the Arduino
board is
connected to an appropriate circuit where digital pins 9 to 13 are
configured
as outputs and connected to leds, digital pin #2 is configured as
input and
```

```
connected to a button that pulls the voltage down when pressed, and
analog
pin #2 is connected to a potentiometer. Have a look at the schematics
in the
blink_challenge_sch.mdl file, in the examples folder.

-----------------------------------------------------------------
-----

USING THE ADAFRUIT MOTOR SHIELD:

The adafruit motor shield is a shield (with an associate library)
to control dc and stepper motors. The Arduino IO package allows using
the
adafruit motor shield library primitives (in addition to the other
basic
IO and servo functions described above) directly from MATLAB.

Details on the shield are here, version 1:
http://learn.adafruit.com/adafruit-motor-shield
and version 2:
http://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino

Make sure you follow the instructions on installing the required
library
(also see the "Installing the motor shield library" section above).

If you were using the adioes.pde sketch then you need to upload either
the motor_v1.pde (for versions 1.x of the shield) or motor_v2.pde (for
the version 2 of the shield) sketch on the board, in order to use the
motor
shield related instructions. Both sketches NEED the related adafruit
motor
shield LIBRARY (see related installation step above) to work,
regardless
on whether the motor shield is mounted on top of the arduino or not.

Both motor sketches are the most comprehensive ones, that is you can
use
them also for basic I/O, servos, encoders and so on, even when the
motor
shield is not mounted on the arduino. However, note that the adafruit
motor shield V1 uses a lot of the pins, (see the manual) so if you use
both
analog and digital IO and motor instructions they will likely
INTERFERE
WITH EACH OTHER, (especially for Version 1 of the library and shield).
Therefore the best approach is using EITHER the IO instructions OR the
motor shield instructions and delete and reinstantiate the arduino
object
everytime one needs to switch from one set of instructions to the
other.

DO NOT HOT SWAP SHIELDS. Before mounting the motor shield delete the
arduino object from the workspace, disconnect, then mount the motor
shield
on top of the arduino, reconnect, upload motor.pde to the board, and
finally, from MATLAB, create a new arduino object.

ADAFRUIT MOTOR SHIELD EXAMPLE:
```

```
% connect the board
a=arduino('COM5')

% sets speed of motor 4 as 200/255
motorSpeed(a,4,200)

% prints the speed of all motors
motorSpeed(a);

% runs motor 1 forward
motorRun(a,4,'forward');

% runs motor 3 backward
motorRun(a,4,'backward');

% release motor 1
motorRun(a,4,'release');

% sets speed of stepper 1 as 10 rpm
stepperSpeed(a,1,10)

% prints the speed of stepper 1
stepperSpeed(a,1);

% rotates stepper 1 forward of 100 steps in interleave mode
stepperStep(a,1,'forward','double',100);

% rotates stepper 1 backward of 50 steps in single mode
stepperStep(a,1,'backward','single',50);

% releases stepper 1
stepperStep(a,1,'release');

% close session
delete(a)

Note that all the other functions related to pins analog and digital
IO
as well as to servos, can still be used as described in the previous
example above.

Also note that the stepper instructions are blocking, that is the
board
cannot start to execute a new instruction until the execution of the
previous stepper command has come to completion (which might take
several
seconds if the number of required steps is high).

-------------------------------------------------------------------
-----
```