

Visión artificial aplicada a un brazo robot

Ibarra Alexis - Fernández Diego

Proyecto final de Ingeniería en Electrónica
UNNE - FaCENA
2013

Contenidos:

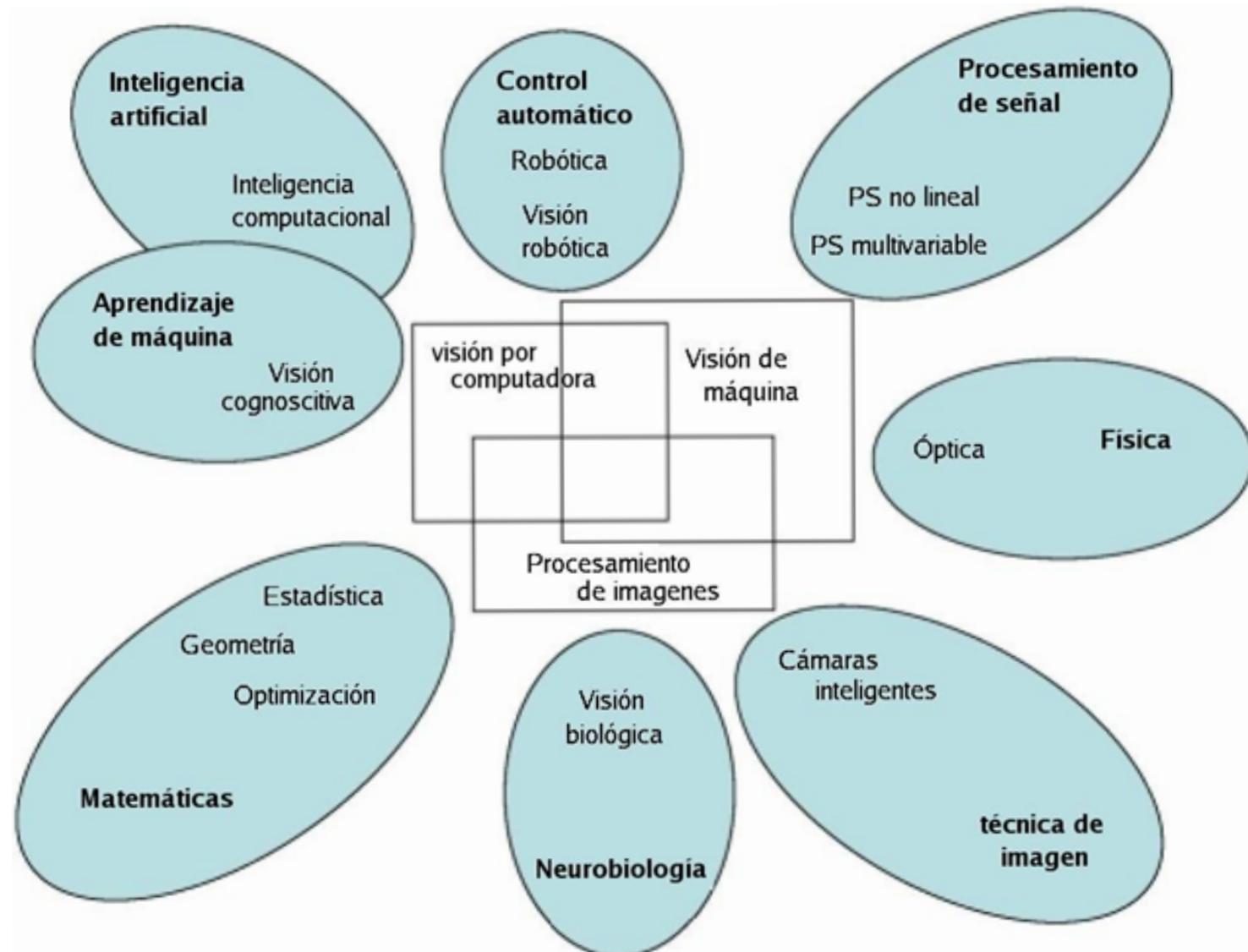
- **1. Fundamentos de visión artificial**
 - Procesamiento de imágenes y video.
 - OpenCV .
- **2. Hardware: arquitectura del proyecto**
 - Brazo robot
 - Cámara
- **3. Software: Diseño y desarrollo.**
 - Resultados (algoritmos finales)
- **4. Implementación : Ensayos y resultados**
 - Aplicación 1
 - Aplicacion 2
- **5. Conclusiones:**
 - Mejoras y trabajos futuros



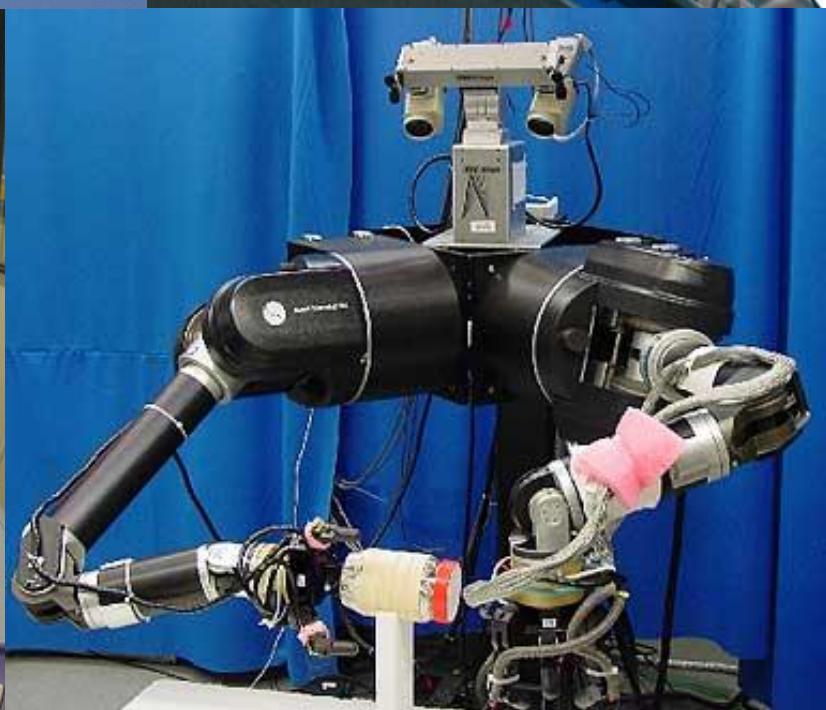
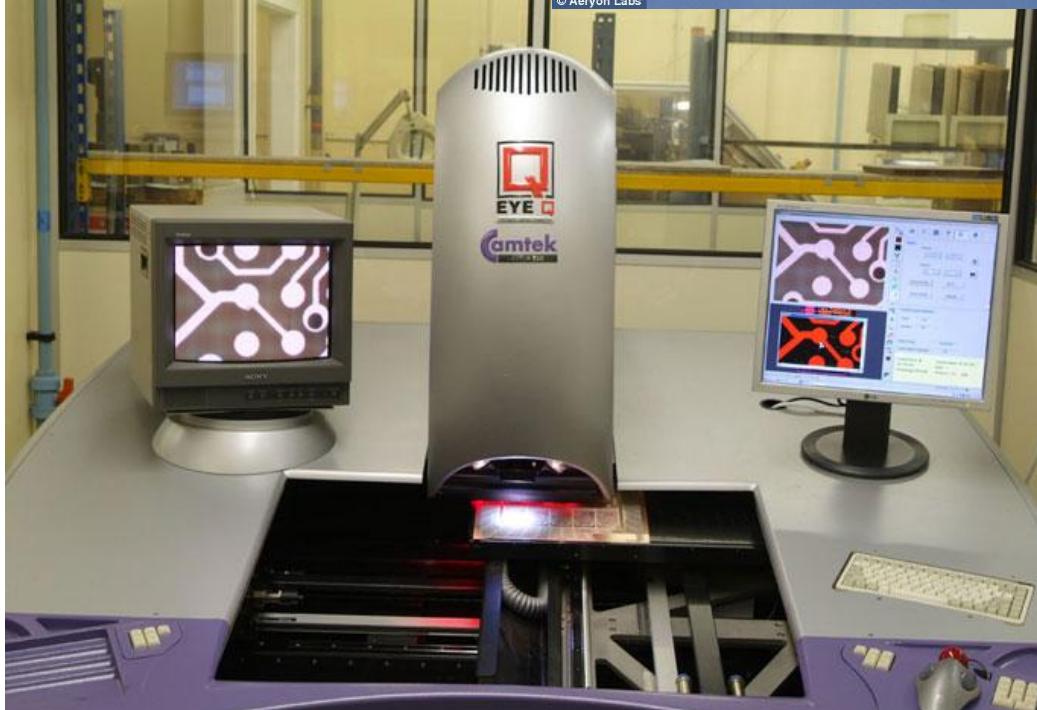
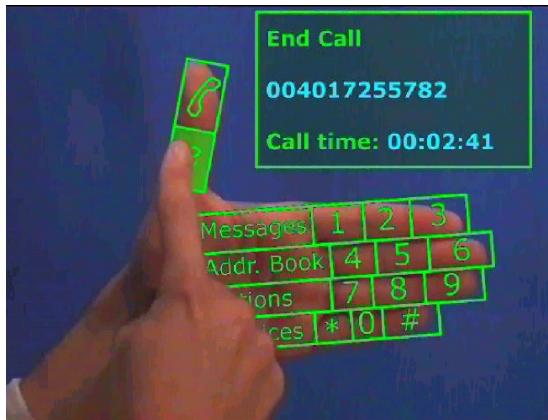
1. Fundamentos de Visión artificial

Procesamiento de imágenes

Áreas de uso de la visión artificial



Aplicaciones prácticas



Sensor Óptico (condiciones)

- **¿Qué es tiempo real en procesamiento de imágenes?**
 - 15 fps como minimo
- **¿Qué tasas de datos se maneja?**
 - bitrate = Tamaño Imagen x profundidad de color x fps
 - bitrate = (640x480)* 8bits * 15
 - **bitrate = 36,8 Mbits**

Transformaciones básicas en imágenes



Imagen original



Disminución de
resolución



Heuristica



umbral



Inversión

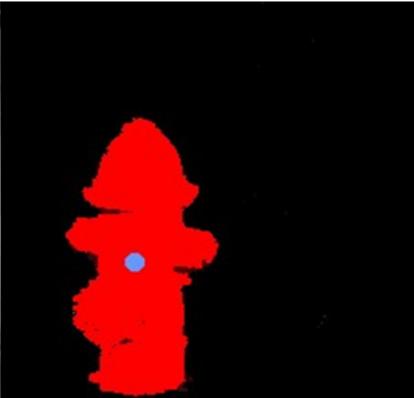


Brillo

Algoritmos básicos (generales)



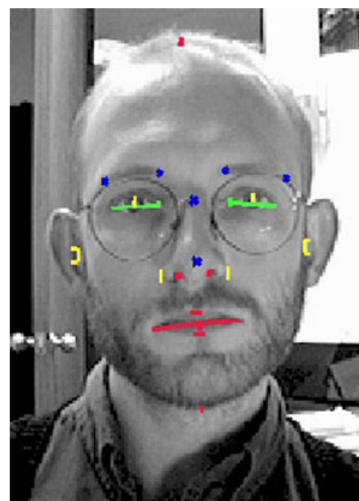
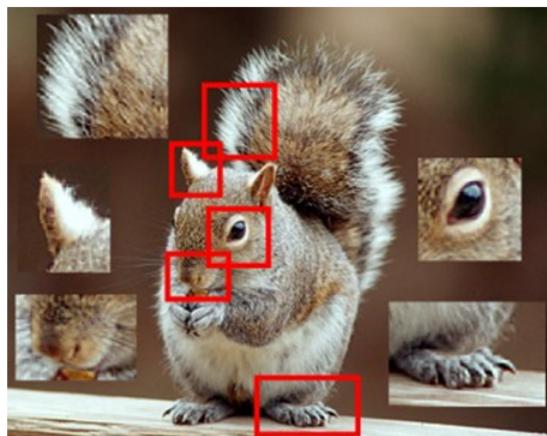
Detección de bordes
(filtros en espacio de color)



Detección de objetos y
centro de masa
(histograma,contornos)

Algoritmos complejos (específicos)

Haar-like features



Correlación de imágenes
y reconocimiento facial

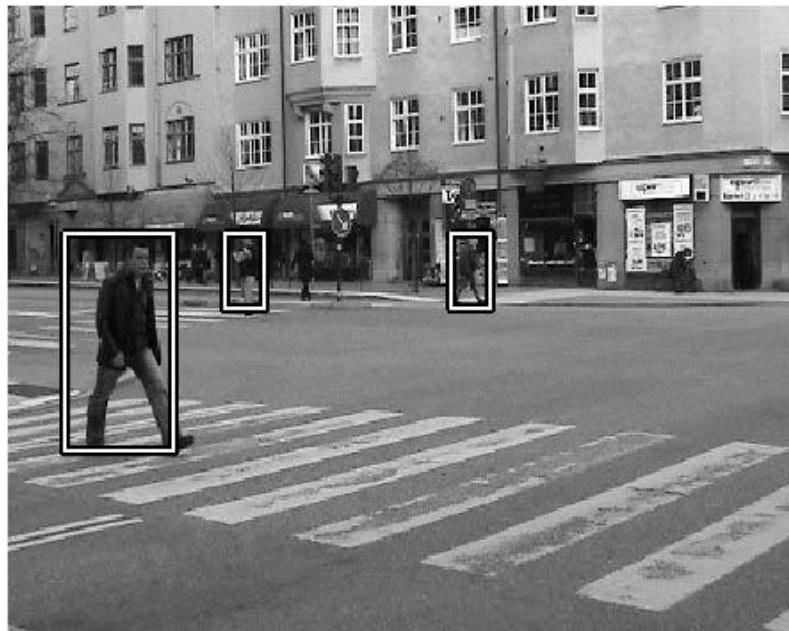




1.Fundamentos de Visión artificial

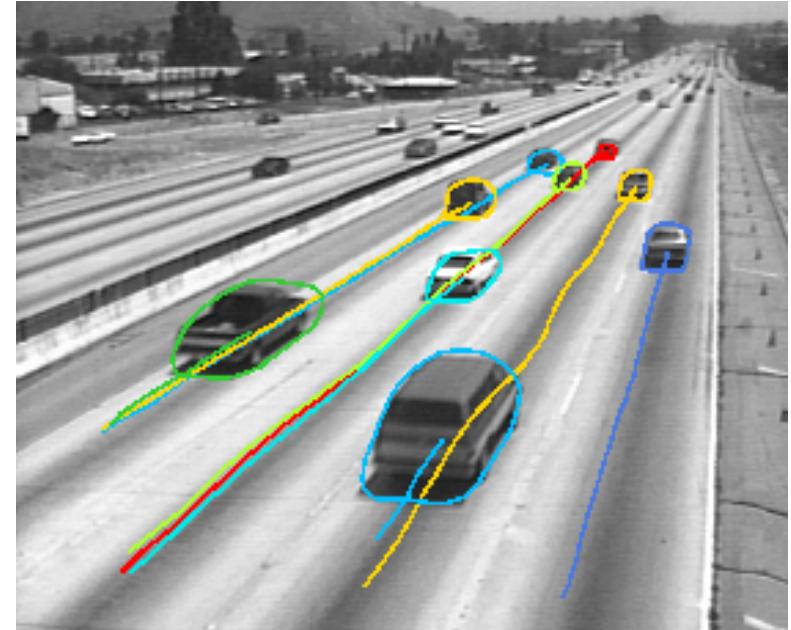
Procesamiento de video

Algoritmos básicos para procesamiento de video



Detección de movimiento
(Bulk motion)

Seguimiento y
predicción
(Tracking)



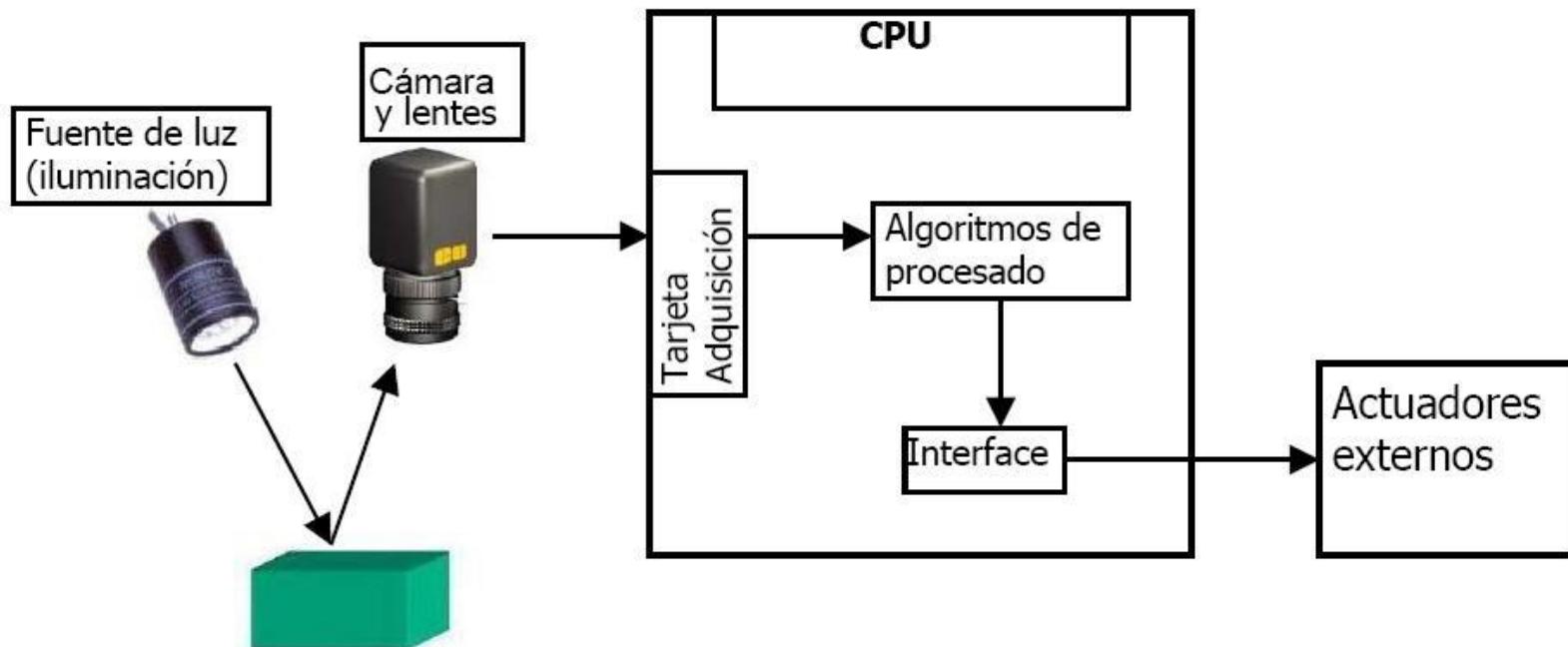
Algoritmos específicos

Flujo óptico
(Optical flow)

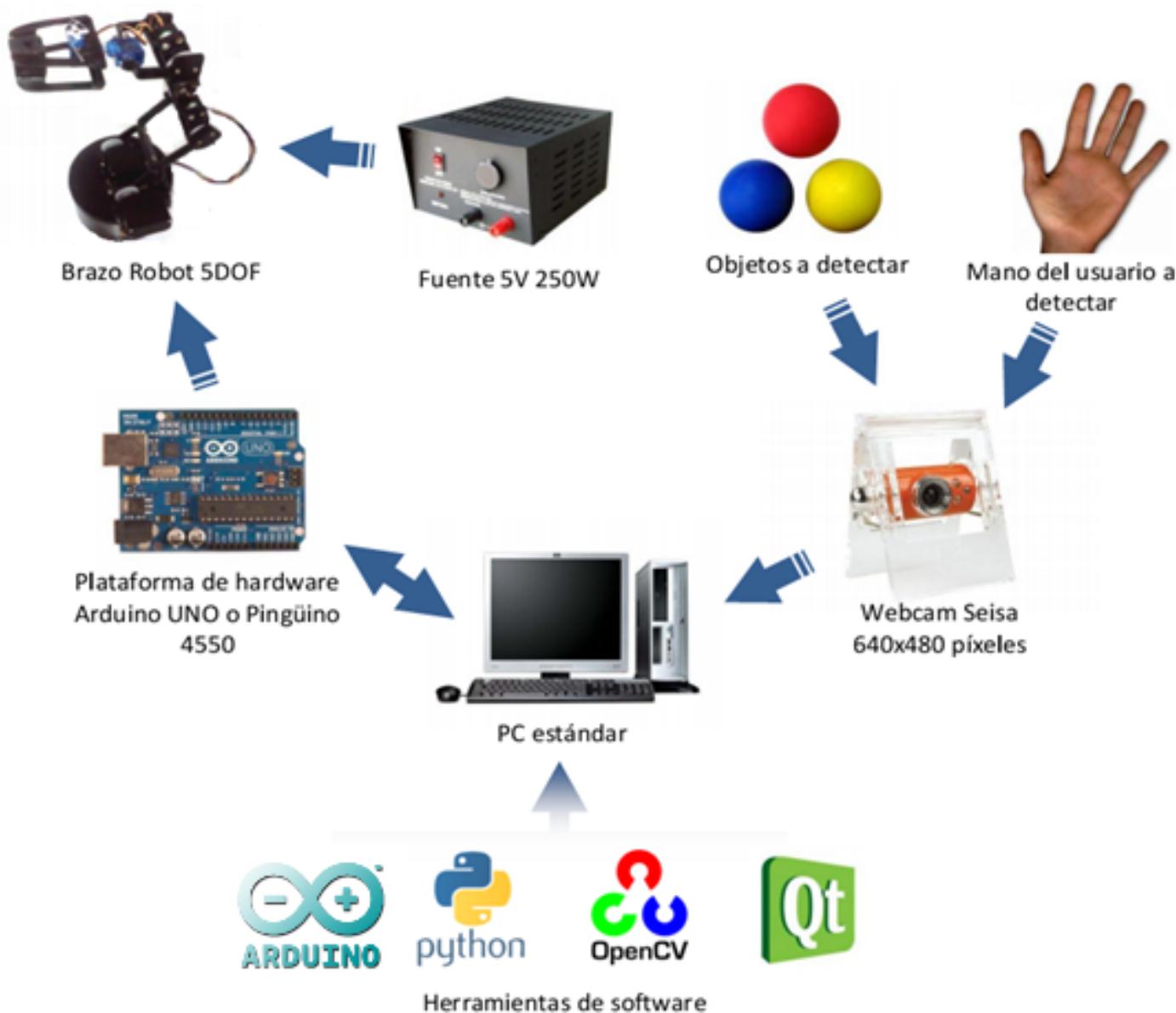


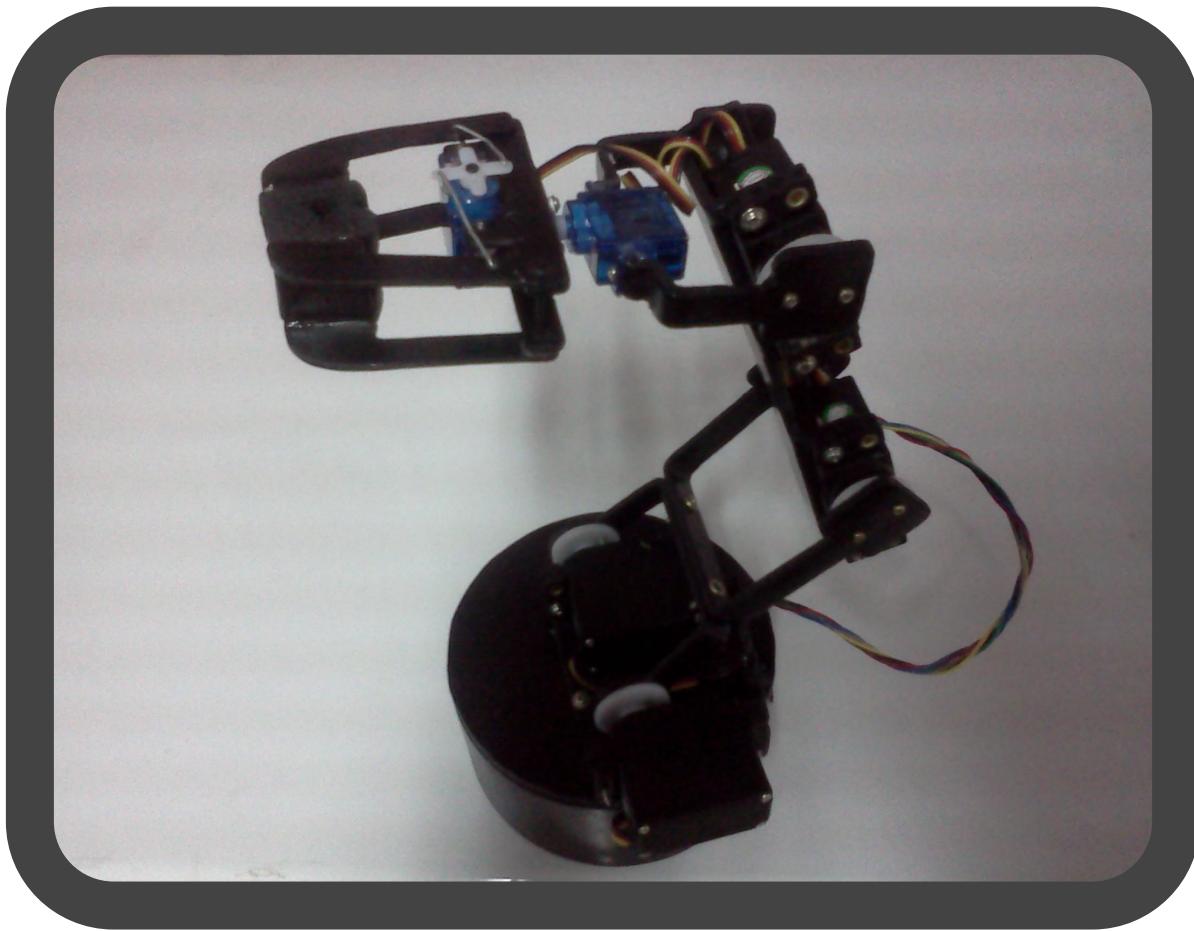
Sustracción de fondo
(análisis de profundidad)

2. Hardware: arquitectura del proyecto



Estructura general del proyecto





Brazo Robot

Construcción

- Piezas realizadas en PVC
- Servos utilizados:
 - 5 x Hitec HS-311
 - 2 x Hitec HS-55

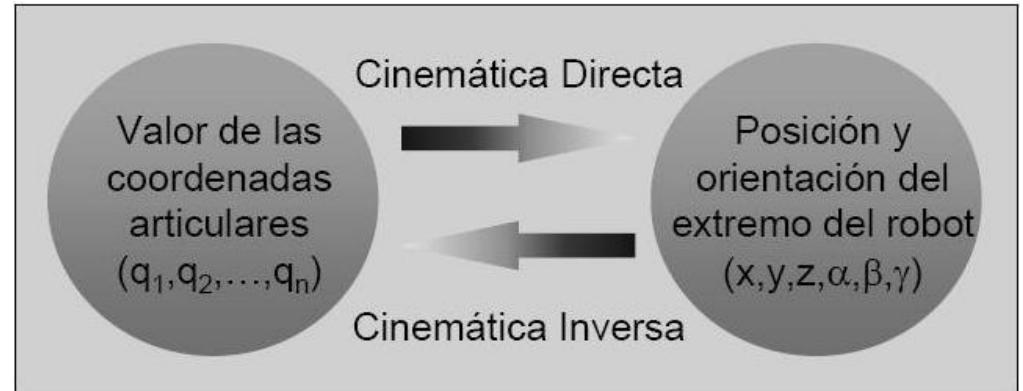


Control System: +Pulse Width Control 1500usec Neutral
Required Pulse: 3-5 Volt Peak to Peak Square Wave
Operating Voltage: 4.8-6.0 Volts
Operating Temperature Range: -20 to +60 Degree C
Operating Speed (4.8V): 0.19sec/60° at no load
Operating Speed (6.0V): 0.15sec/60° at no load
Stall Torque (4.8V): 42 oz/in (3.0 kg/cm)
Stall Torque (6.0V): 51 oz/in (3.7 kg/cm)
Current Drain (4.8V): 7.4mA/idle, 160mA no load operating
Current Drain (6.0V): 7.7mA/idle, 180mA no load operating
Dead Band Width: 5usec
Operating Angle: 45° one side pulse traveling 450usec
Direction: Multi-directional
Motor Type: Cored Metal Brush
Potentiometer Drive: 4 Slider/Direct Drive
Bearing Type: Top Resin Bushing
Gear Type: Nylon
360 Modifiable: Yes
Connector Wire Length: 11.81" (300mm)
Weight: 1.52oz (43g)



Control System: +Pulse Width Control 1500usec Neutral
Required Pulse: 3-5 Volt Peak to Peak Square Wave
Operating Voltage: 4.8-6.0 Volts
Operating Temperature Range: -20 to +60 Degree C
Operating Speed (4.8V): 0.17 sec/60° at no load
Operating Speed (6.0V): 0.14 sec/60° at no load
Stall Torque (4.8V): 15.27 oz-in. (1.1kg.cm)
Stall Torque (6.0V): 18.05 oz-in. (1.3kg.cm)
Operating Angle: 40° one side pulse traveling 400usec
360 Modifiable: Yes
Direction: Clockwise/Pulse Traveling 1500 to 1900usec
Current Drain (4.8V): 5.4mA/idle and 150mA no load operating
Current Drain (6.0V): 5.5mA/idle and 180mA no load operating
Dead Band Width: 8usec
Motor Type: Coreless
Potentiometer Drive: Direct Drive
Bearing Type: None, outer case serves as bearing
Gear Type: All Nylon
Connector Wire Length: 6.29" (160mm)
Dimensions: 0.89" x 0.45"x 0.94" (22.8 x 11.6 x 24mm)
Weight: 0.28oz (8g)

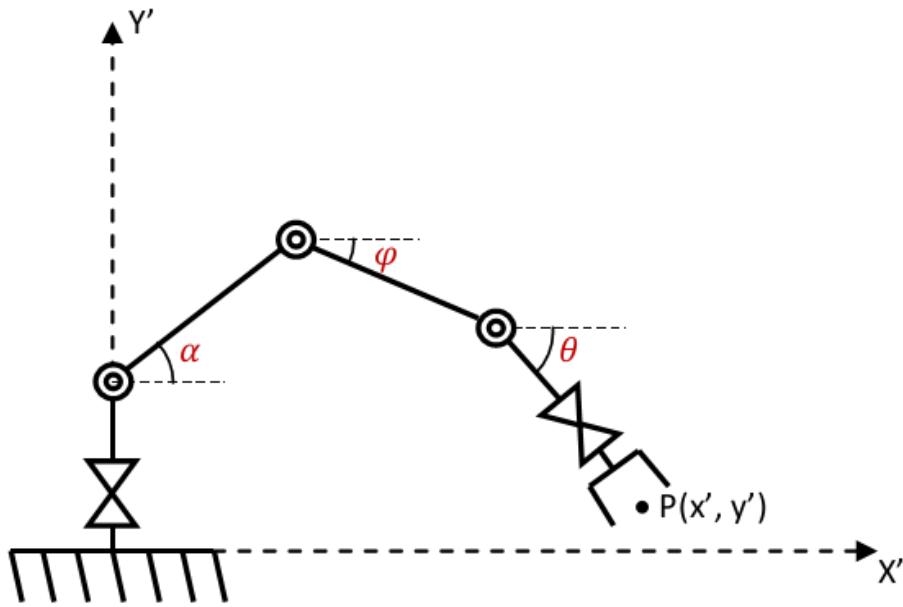
Cinemática



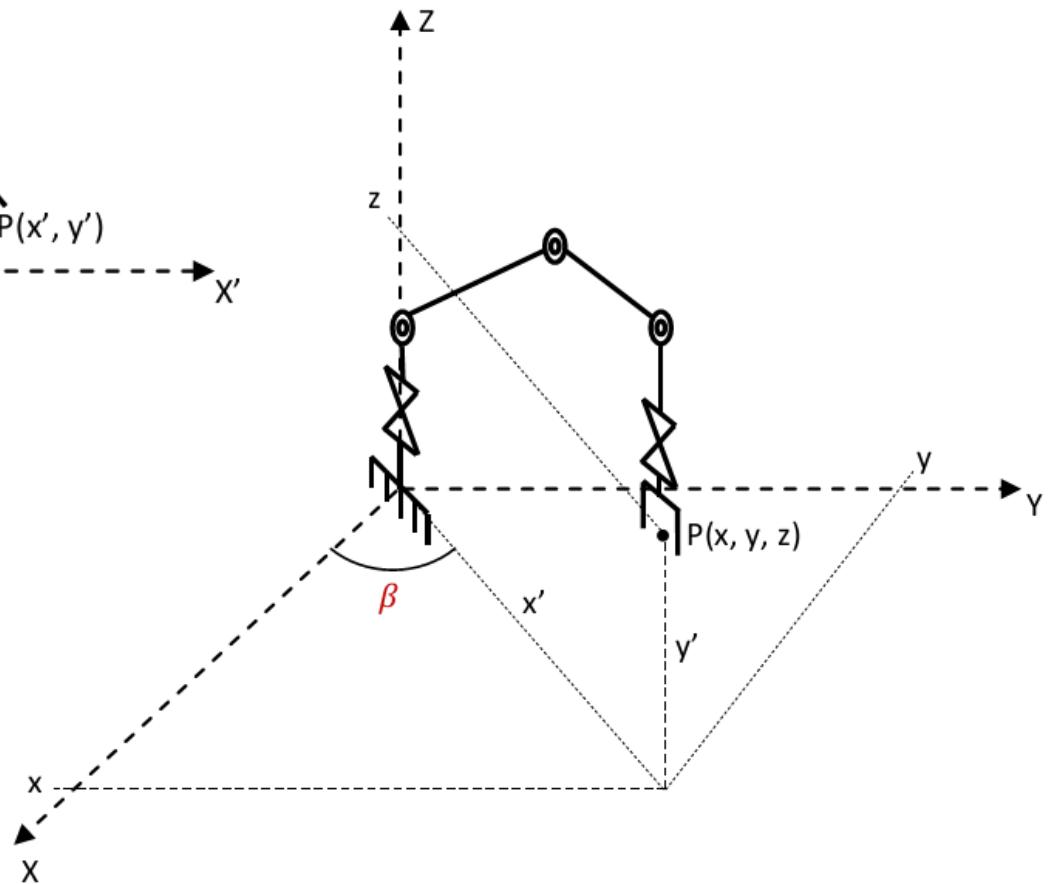
Cinemática directa: Conjunto de ecuaciones que relacionan el valor de cada una de las coordenadas articulares con la posición del extremo (efector) del robot en coordenadas espaciales y su orientación.

Cinemática inversa: Algoritmo ó conjunto de ecuaciones que representan los valores de las “N” coordenadas articulares en función de los valores de posición y orientación requeridos en el extremo del robot.

Método Geométrico



Representación gráfica del
robot en 2 y 3 dimensiones



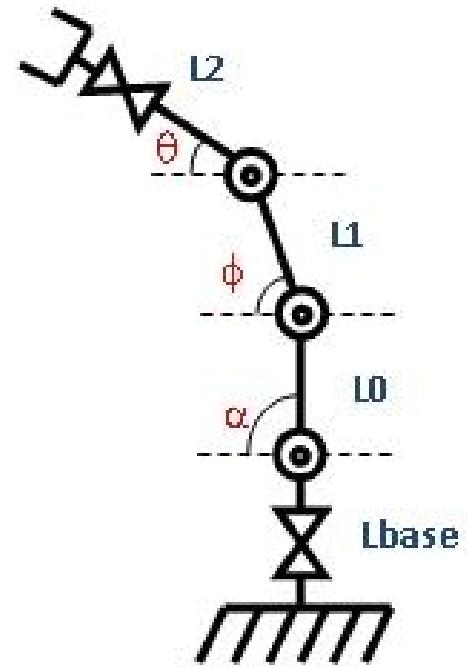
Cinemática Directa

Resolución por método geométrico:

$$\beta = \arctan\left(\frac{y}{x}\right)$$

$$x' = \sqrt{x^2 + y^2}$$

$$y' = z$$

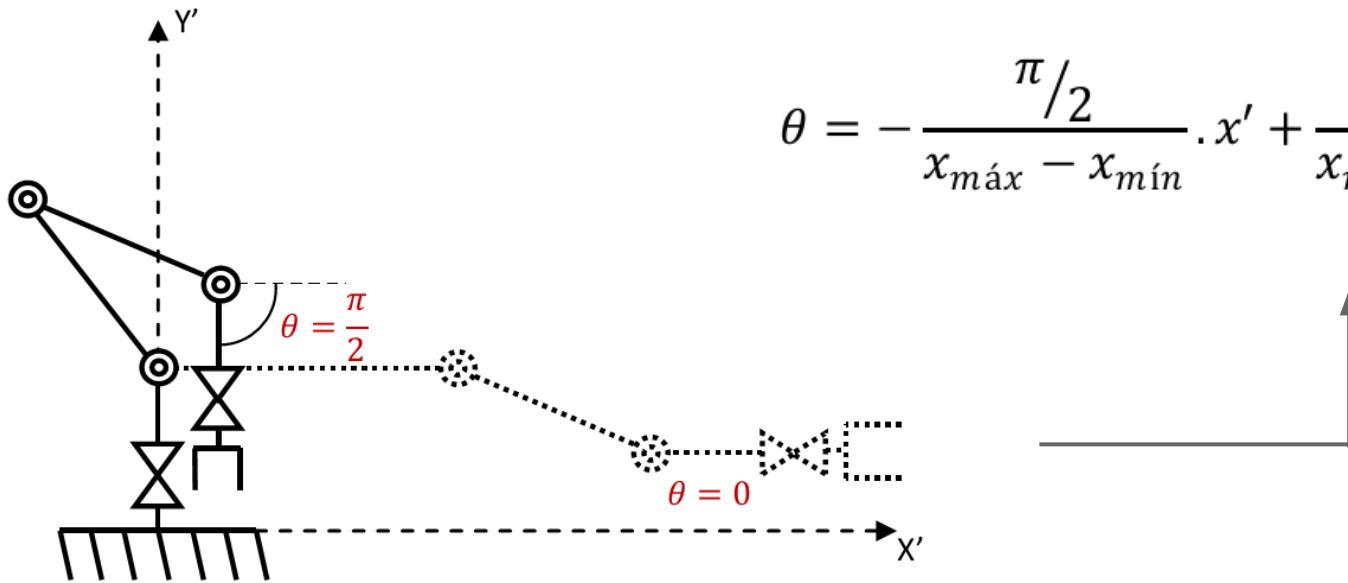


A partir de las ecuaciones de cinemática directa se obtiene:

$$x' = \cos(\alpha) \cdot L_0 + \cos(\varphi) \cdot L_1 + \cos(\theta) \cdot L_2$$

$$y' = \sin(\alpha) \cdot L_0 - \sin(\varphi) \cdot L_1 - \sin(\theta) \cdot L_2 + L_{base}$$

Cinemática Inversa



$$\theta = -\frac{\pi/2}{x_{máx} - x_{mín}} \cdot x' + \frac{\pi/2}{x_{máx} - x_{mín}} \cdot x_{máx}$$

$$\varphi = \arcsen \left(\frac{-y' + \sen(\alpha) \cdot L_0 - \sen(\theta) \cdot L_2 + L_{base}}{L_1} \right)$$

$$\begin{aligned} \alpha &= \arcsen \left(\frac{(x' - \cos(\theta) \cdot L_2)^2 + (y' + \sen(\theta) \cdot L_2 - L_{base})^2 + L_0^2 - L_1^2}{2L_0 \sqrt{(x' - \cos(\theta) \cdot L_2)^2 + (y' + \sen(\theta) \cdot L_2 - L_{base})^2}} \right) \\ &\quad - \arctan \left(\frac{x' - \cos(\theta) \cdot L_2}{y' + \sen(\theta) \cdot L_2 - L_{base}} \right) \end{aligned}$$



Hardware de control: Pinguino y Arduino

Tabla comparativa entre placas

Característica	Arduino UNO (Atmega328)	Pingüino (PIC 18F4550)
Voltaje operativo	5V	5V
Pines de I/O	14 (6 PWM)	20 (2 PWM)
Pines de entrada analógica	6	8
Corriente por cada pin I/O	40 mA	25 mA
Memoria Flash	32 KB (0.5 KB reservados)	32 KB (8 KB reservados)
SRAM	2 KB	2 KB
EEPROM	1 KB	256 Bytes
Frecuencia	16 MHz	20 MHz
USB Nativo	No	Si
Precio	26,39 U\$S (DigiKey)	23,27 U\$S (DigiKey)



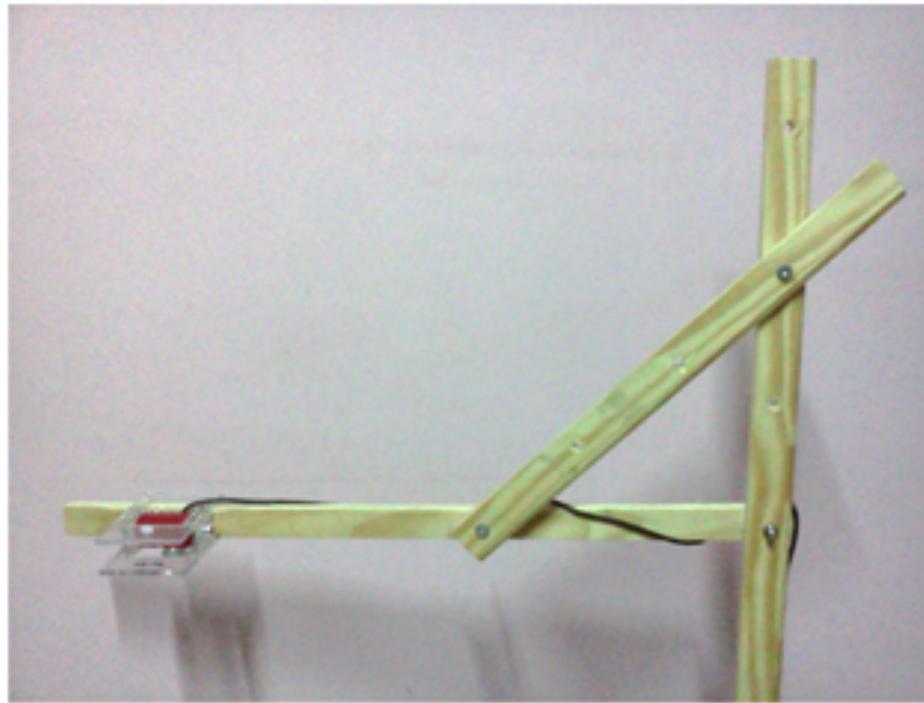
Cámara



Posee un sensor óptico CMOS, que permite trabajar en tiempo real con los siguientes parámetros:

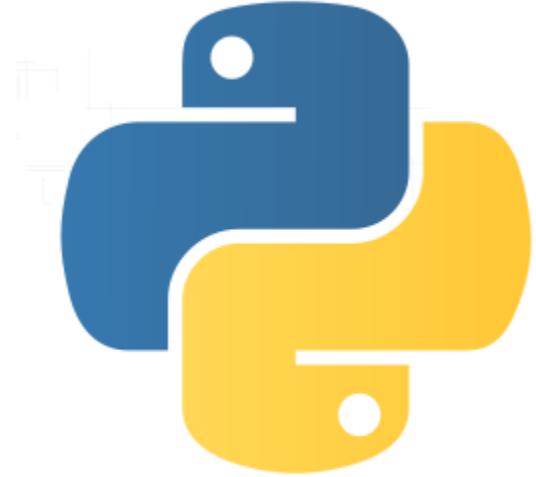
- Resolución: 640X480
- Fps: 15 a 30 fps
- Color: 8 bits
- Ajustes: automático o manual.

Soporte para cámara web





3. Software : Implementación y desarrollo



Python

¿Qué es Python?

Es un lenguaje de programación creado por Guido van Rossum a principios de los años 90, con la particularidad de que cuenta con una sintaxis muy limpia y que favorece un código legible.

```
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodeName()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print '%s [%label=%s]' % (nodename, label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s'; ' % ast[1]
        else:
            print ''
    else:
        print ']';
    children = []
    for n, child in enumerate(ast[1:]):
        children.append(dotwrite(child))
    print '%s -> {' % nodename
    for name in children:
        print '%s' % name,
```

Características

- Lenguaje interpretado o de script
- Tipado dinámico
- Fuertemente tipado
- Multiplataforma
- Orientado a objetos

¿Por qué Python?

Su sintaxis es simple, clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje entre otros, hacen que desarrollar una aplicación en Python sea sencillo y más rápido.



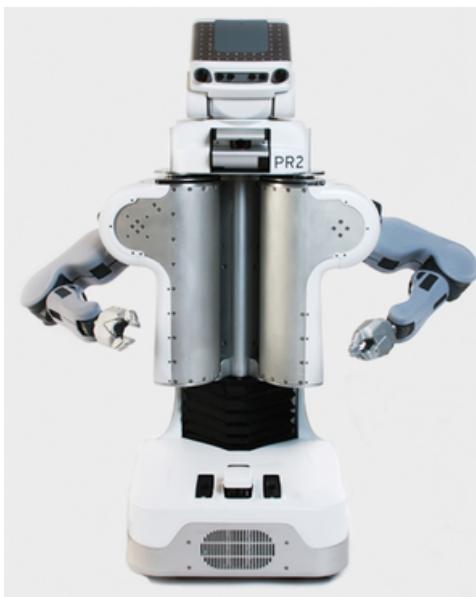
Open Source Computer Vision

¿Qué es OpenCV?

Conjunto de librerías desarrollado por Intel. Nació en 1999, se desarrolló principalmente en Rusia y se distribuye bajo licencia BSD (permite proyectos propietarios privados).

Se ha usado en infinidad de proyectos como por ejemplo robots para DARPA, el sistema ROS (robot operative systems) y robots de venta comercial como el **PR2** y el **texai**.

Engloba algoritmos y funciones de procesamiento de video, imágenes y visión artificial



OpenCV Overview: > 500 functions

opencv.willowgarage.com

Robot support

The image is a collage of various OpenCV applications and concepts, each with a green title bar:

- General Image Processing Functions**: Shows various image processing operations like erosion, dilation, and contrast adjustment.
- Image Pyramids**: Shows a diagram of a coarse-to-fine optical flow estimation process using image pyramids.
- Segmentation**: Shows foreground and background segmentation results for objects like people and cars.
- Geometric descriptors**: Shows geometric descriptors for shapes like triangles and hands.
- Features**: Shows feature extraction from a scene with red lines highlighting features.
- Tracking a calibration object**: Shows tracking of a chessboard pattern.
- Transforms**: Shows affine and perspective transformations.
- Camera calibration, Stereo, 3D**: Shows camera calibration and stereo vision examples.
- Machine Learning: Detection, Recognition**: Shows face detection and recognition results.
- Utilities and Data Structures**: Shows a circular diagram of OpenCV utilities and data structures.
- Tracking**: Shows tracking of a person's head.
- Optical Flow in 1D**: Shows optical flow vectors in 1D.
- Fitting**: Shows fitting of a circle to a set of points.
- Matrix Math**: Shows matrix operations like dot product and convolution.

¿Para qué sirve OpenCV en el proyecto?

Procesamiento de imágenes

- . Binarización de imágenes.
- . Cálculo de histogramas.
- . Mapeo de espacios de color (RGB y HSV).
- . Detección de formas,tamaños, etc.

Procesamiento de video

- . Desplazamiento medio de imágenes recursivo (Camshift98).
- . Filtrado de movimiento recursivo.
- . Interpolación de movimiento (estimador).



**Code less.
Create more.
Deploy everywhere.**

QT : Librerías para entorno gráfico

¿Qué es QT?



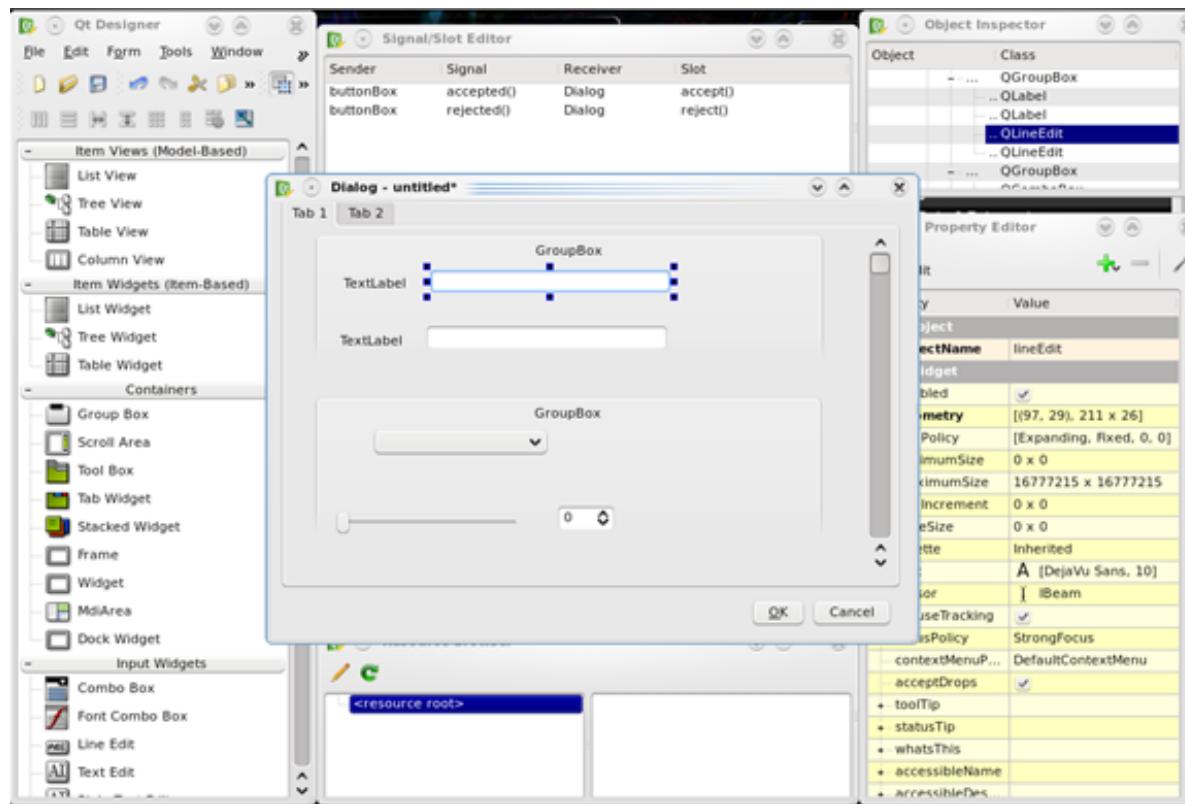
Es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con **interfaz gráfica de usuario**, así como también para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos y consolas para servidores.

Qt utiliza el lenguaje de programación **C++** de forma nativa, adicionalmente puede ser utilizado en varios otros lenguajes de programación a través de *bindings*.

¿Qué hace QT en el proyecto?

En el proyecto utilizamos **PyQt**, un binding de Qt para el lenguaje de programación Python.

Con estos componentes se logró una interfaz gráfica rápida, intuitiva, estable y ampliable, para modificaciones futuras.





Arduino : Entorno de programación

¿Qué es Arduino y para qué se lo utilizó?

El entorno de desarrollo integrado Arduino es una aplicación multiplataforma escrita en Java y derivada del IDE del lenguaje de programación Processing.

Esta aplicación se utiliza para la programación en lenguaje C y C++ de la plataforma de hardware Arduino.

Cuenta con un editor de textos y es capaz de compilar el código, para luego grabarlo en el microcontrolador con un simple clic.

Pinguino IDE x.3 rev. 303

Fichier Editer Preferences Aide

bt Terminal DS2432 Blink firstimpl RealTimeClock

(top)

```
1  /* -----
2   FILE:      RealTimeClock.pde
3   PROJECT:   Pinguino
4   PURPOSE:   Real Time Clock and Calendar functions Demo
5   PROGRAMER: regis blanchot <rblanchot@gmail.com>
6   BOARD:     PIC32-PINGUINO (OLIMEX)
7   or any other PIC32MX board with external 32.768 kHz clock crystal
8
9   FIRST RELEASE: 11 apr. 2011
10  LAST RELEASE: 05 jul. 2011
-----*/
11
12  char Day[7][5] = {"Sat.", "Sun.", "Mon.", "Tue.", "Wed.", "Thu.", "Fri."};
13  char Month[13][5] = {"", "Jan.", "Feb.", "Mar.", "Apr.", "May.", "Jun.", "Jul.", "Aug.", "Sep.", "Oct.", "Nov.", "Dec."};
14
15  void blink1()
16  {
17      toggle(LED1);
18  }
19
20  void setup()
21  {
22      u32 Tm = 0x23595500; // 23hr, 59 min, 55 sec
23      u32 Dt = 0x11123100; // Saturday (day 0 of the week), 31 Dec. 2011
24      u32 aTm = 0x00000500; // 00hr, 00 min, 05 sec
25      u32 adt = 0x12010101; // Sunday (day 1 of the week), 1st Jan. 2012
26      u16 drift = 200; // add 200 pulse every minute to adjust time
  
```

Output

```
Board: PIC32 Pinguino OTG
Proc: 32MX440F256H
compilation done
code size: 24548 / 262144 bytes (9% used)
3.42659282684 seconds process time
```

Rev. 303 | PIC32 Pinguino OTG

Arduino IDE

Arduino - 0011 Alpha

File Edit Sketch Tools Help

Blink

```
/*
 * Blink
 *
 * The basic Arduino example. Turns on an LED on for one second,
 * then off for one second, and so on... We use pin 13 because,
 * depending on your Arduino board, it has either a built-in LED
 * or a built-in resistor so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */

int ledPin = 13; // LED connected to digital pin 13

void setup() // run once, when the sketch starts
{
    pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

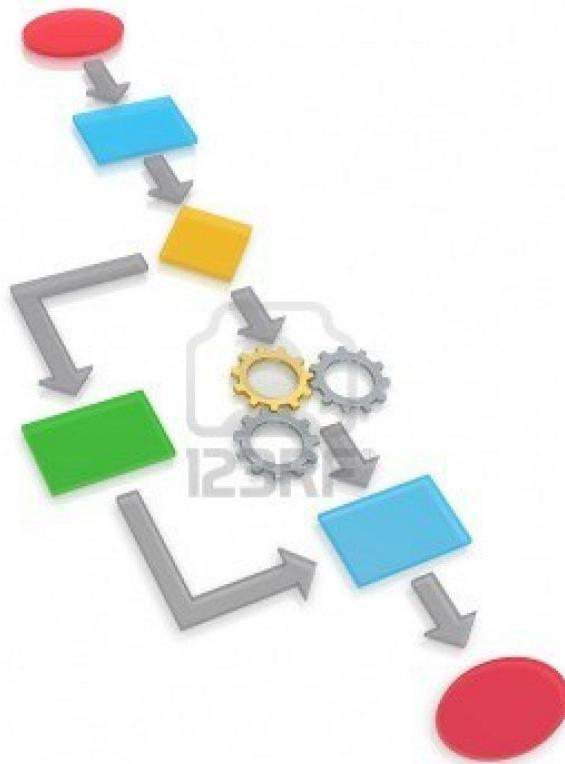
void loop() // run over and over again
{
    digitalWrite(ledPin, HIGH); // sets the LED on
    delay(1000); // waits for a second
    digitalWrite(ledPin, LOW); // sets the LED off
    delay(1000); // waits for a second
}
```

Done compiling.

Binary sketch size: 1098 bytes (of a 14336 byte maximum)

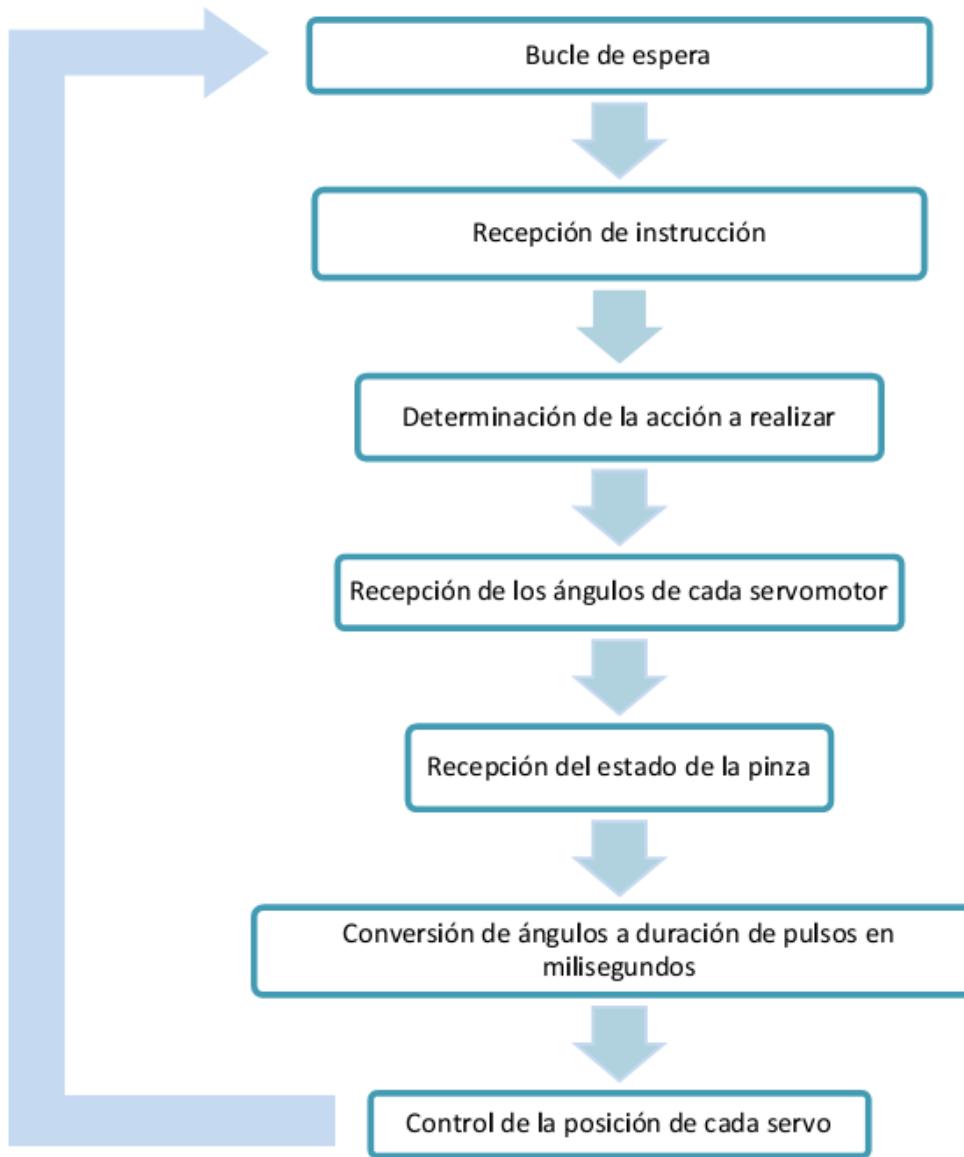
22

Pinguino IDE

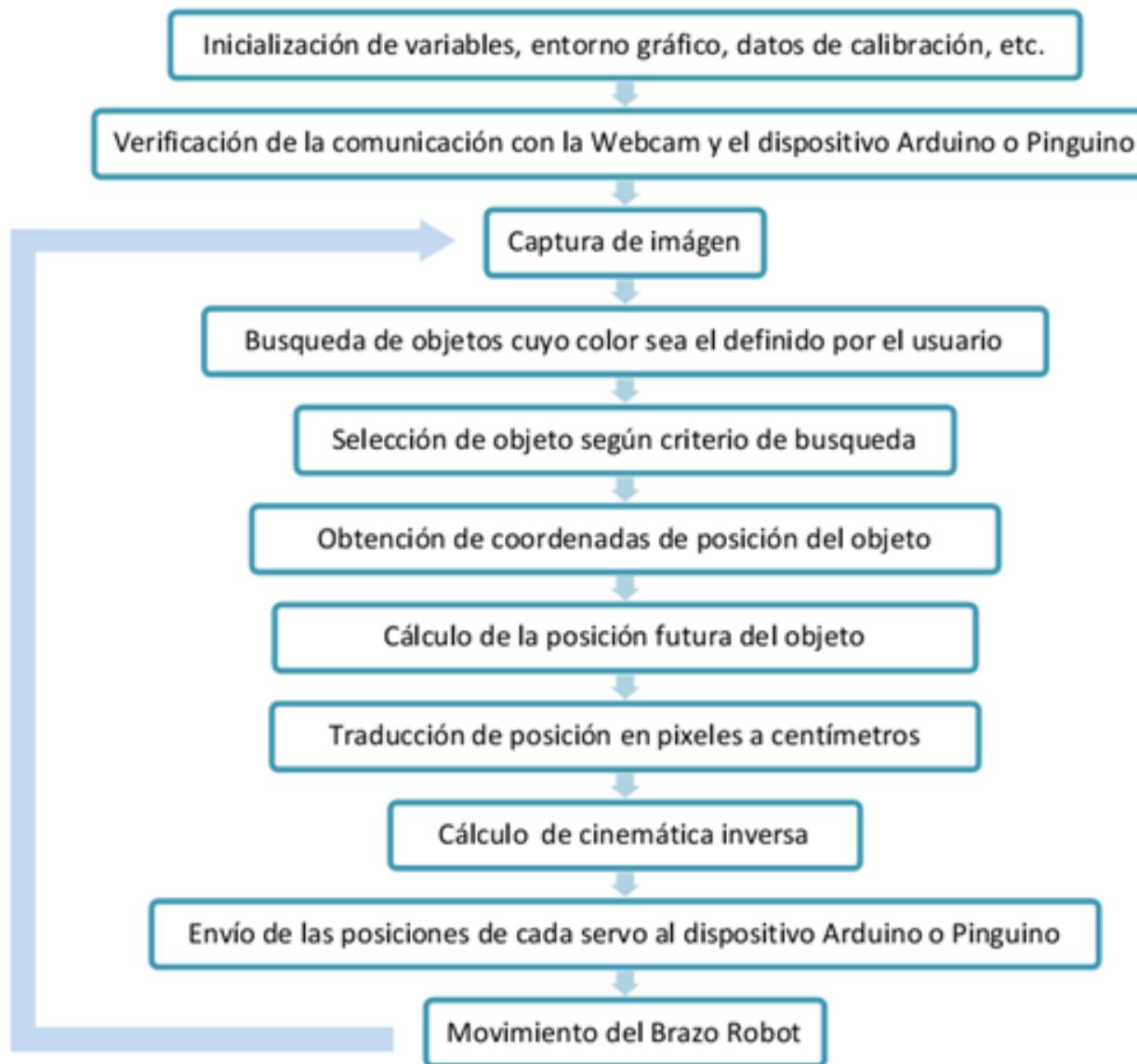


Resultados : algoritmos finales del sistema

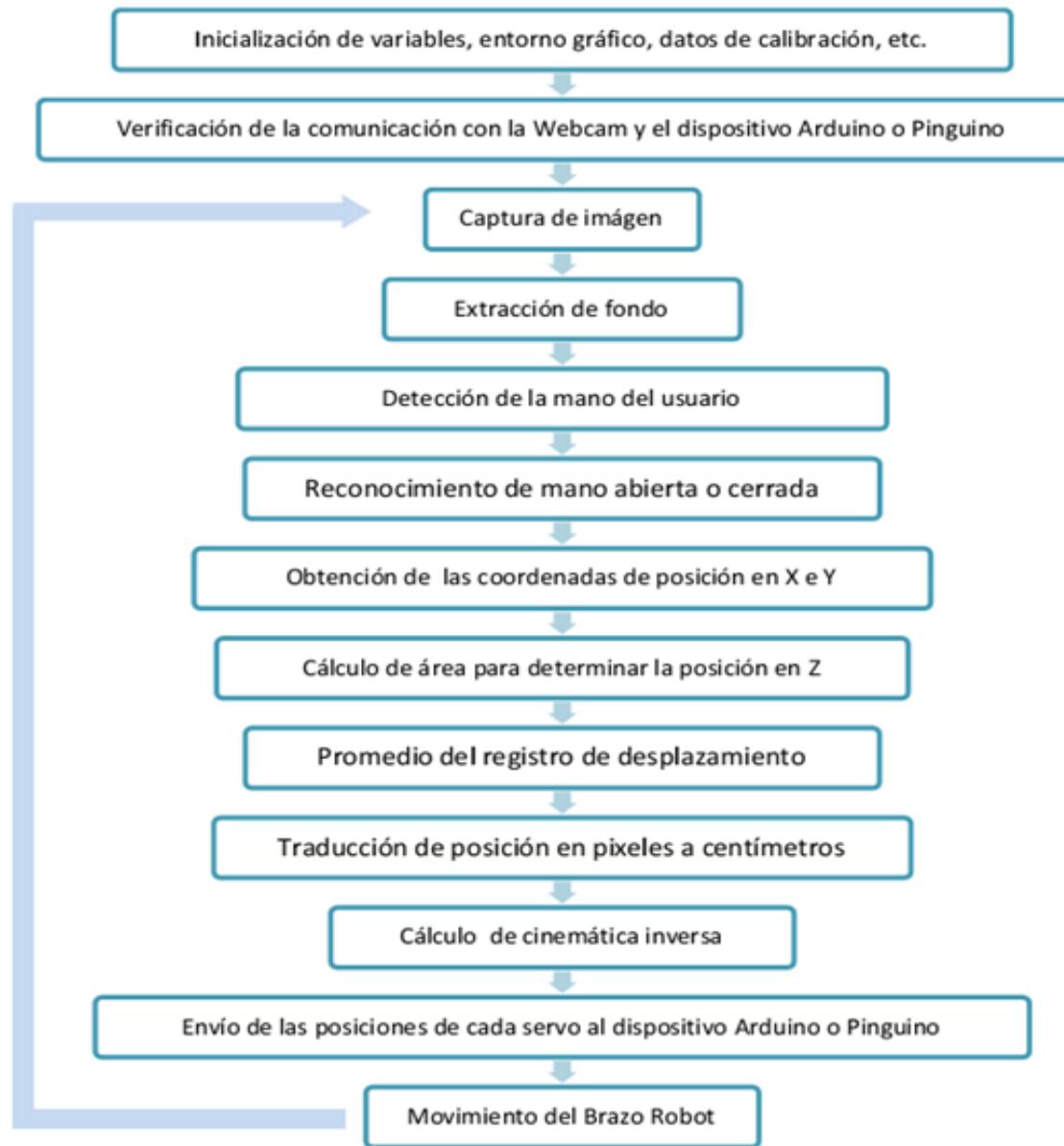
Software del microcontrolador

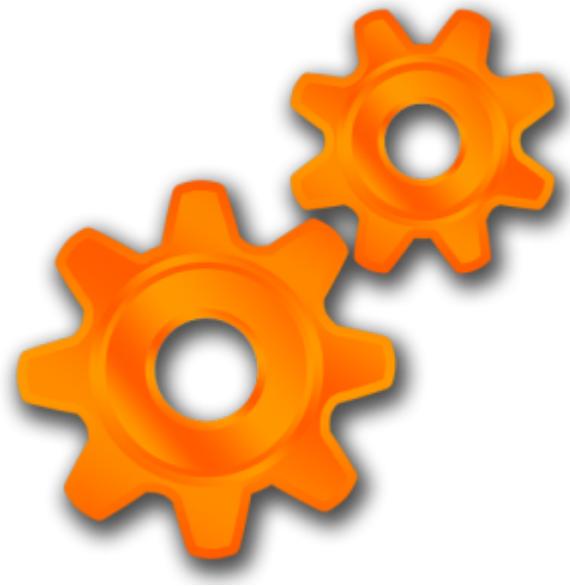


Aplicación 1 : "Reconocimiento de objetos"



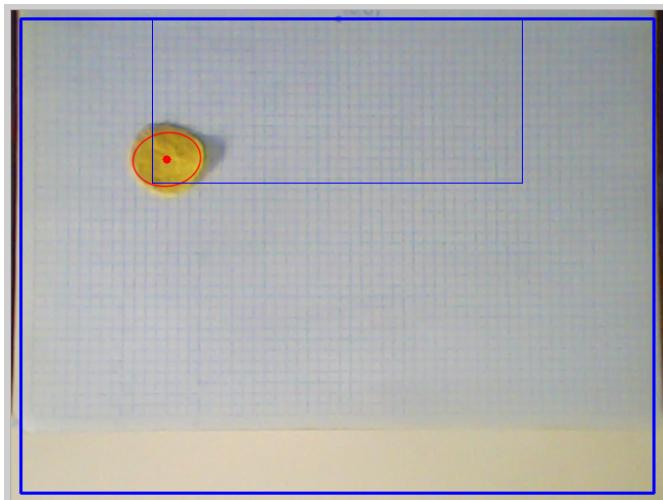
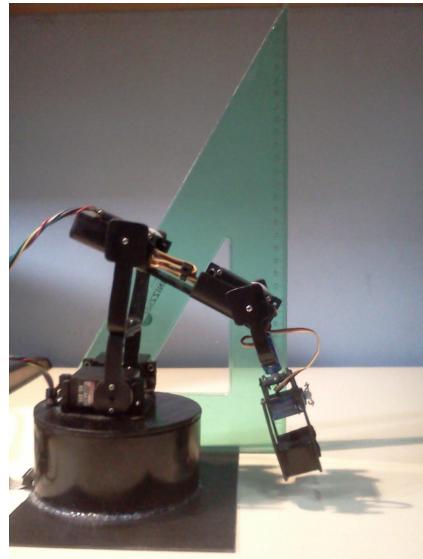
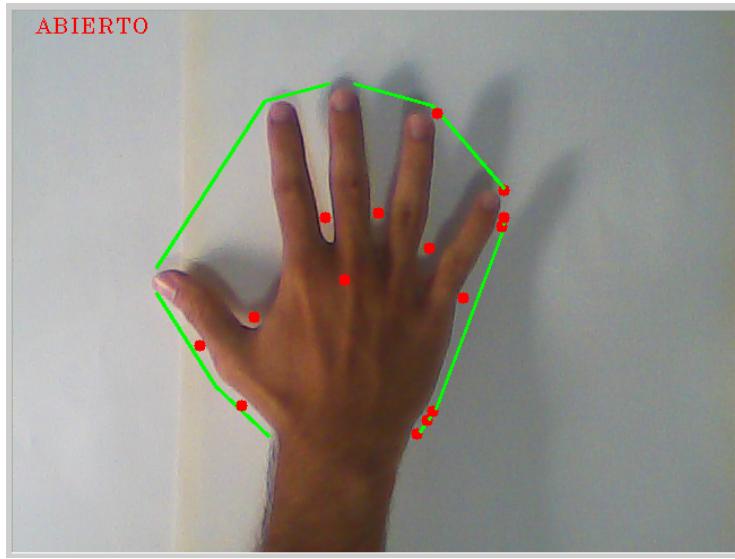
Aplicación 2 : "Manipulación del brazo robot"

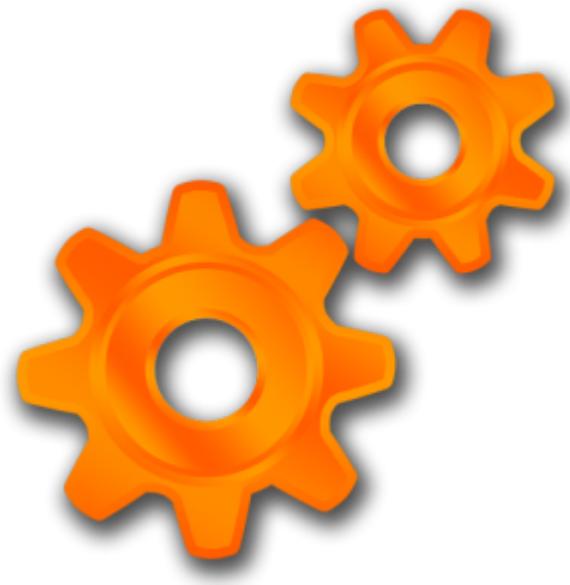




4. Implementación: Ensayos y resultados

Ensayos realizados





Demostración



5. Conclusiones

Conclusiones

- Se logró desarrollar un sistema clasificador de objetos, utilizando la visión artificial para la detección de las posiciones y las características de los objetos.
- Se elaboraron algoritmos de visión por computadora capaces de extraer la información visual necesaria.
- Se diseñó y construyó un brazo robot de 5 grados de libertad (DOF) con materiales de bajo costo.
- Se realizó el estudio de cinemática inversa permitiendo que el autómata manipule y clasifique los objetos conociendo solamente sus posiciones.
- El método geométrico utilizado resultó sencillo y se logró muy buena precisión.
- Se debe proporcionar al sistema de condiciones de iluminación uniformes.
- Se pudo realizar un sistema de visión computarizada portable y flexible.
- Se puede dar mayor funcionalidad al sistema e implementar nuevas aplicaciones mediante simples modificaciones.

Trabajos futuros y mejoras

- Inclusión de filtros predictivos Kalman.
- Implementación de visión estéreo.
- Utilizar cámaras digitales de mayor calidad.
- Agregar funciones al sistema (más aplicaciones).
- Incorporar a la plataforma un autómata de mayor tamaño y capacidad (brazo industrial).
- Implementar realimentación activa del brazo robot.
- Desarrollar un sistema embebido capaz de realizar el procesamiento de datos.
- Agregar sensores al efecto final del brazo robot.
- Aumentar grados de libertad del autómata.
- Integración del sistema desarrollado al entorno ROS.
- Implementar un control a distancia del sistema vía Internet (servidor SSL).

Agradecimientos

A nuestras familias, en especial a Sequi, Axel, Ailen, Olga, Gabriel, Erica, Tony, Ceci, Merce y Javi.

A Carolina.

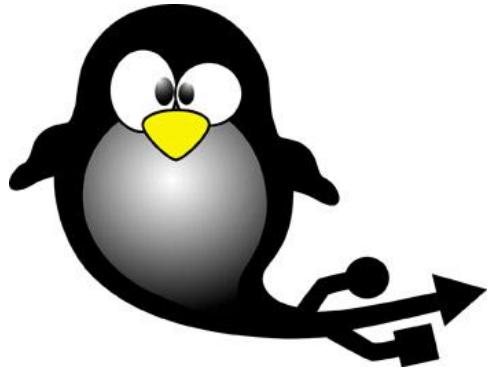
A Jimena.

A Luciano Zini, Juanjo Cochia, Aquino Carlos, Julian Veglia, y todos nuestros amigos, profesores, compañeros y conocidos.

Gracias a estas personas que estuvieron siempre, que nos brindaron apoyo y fuerzas en el transcurso de nuestra carrera y que contribuyeron con opiniones, objeciones, ideas, cuestiones, preguntas, y todo tipo de aportes indispensables para haber logrado nuestro objetivo. Se agradece profundamente el tiempo que nos dedicaron a nosotros y al proyecto.

¿Preguntas?





Muchas Gracias

Alexis Ibarra - Fernández Diego