ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE TELE OPERACIÓN PARA UN ROBOT MÓVIL MEDIANTE RECONOCIMIENTO DE MOVIMIENTOS DE LA CABEZA.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y CONTROL

LENIN FERNANDO BAUTISTA MORETA

lenin.bautista@hotmail.com

DIRECTOR: Dr. GEOVANNY DANILO CHÁVEZ GARCÍA

danilo.chavez@epn.edu.ec

Quito, Mayo 2014

DECLARACIÓN

Yo, Lenin Fernando Bautista Moreta declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Lenin Fernando Bautista Moreta

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Lenin Fernando Bautista Moreta bajo mi supervisión.

_____

Dr. Danilo Chávez

DIRECTOR DEL PROYECTO

AGRADECIMIENTO

Agradezco a mi Madre Leticia, a mi Tío Mario y mis Abuelitos Juan y Rosa, que han estado con migo siendo un apoyo incondicional en todo momento a lo largo de toda mi vida. Además la asesoría del Dr. Danilo Chávez para desarrollar con éxito el presente proyecto de titulación.

-Lenin Fernando Bautista Moreta-

DEDICATORIA

Dedico a quienes amo, en especial a la memoria de mi Padre Jaime que ha sido una motivación más a lo largo de mi vida universitaria.

-Lenin Fernando Bautista Moreta-

CONTENIDO

RESUMEN

En el presente proyecto se realiza el diseño e implementación de un sistema Hombre-Máquina cuyo campo de aplicación es el manejo asistido, ya que la forma de efectuar el control se basa en detección e interpretación de movimientos de la cabeza del usuario.

Se desarrolla un software para cumplir con el objetivo planteado, dicho desarrollo se lo efectúa en Microsoft Visual Studio 2010 con la ayuda de las librerías de OpenCV a las que se accede a través de EmguCV, para trabajar en una aplicación con una interfaz de usuario de Windows en lenguaje Visual Basic.

Para simular el manejo de un vehículo se implementa un mini robot móvil con sistema de dirección tipo Ackerman, este robot está provisto de una cámara que transmite el video de su entorno en forma inalámbrica hacia la interfaz con el usuario. En dicha interfaz se crea un ambiente de manejo virtual que involucra variables como velocidad, temperatura y estado de baterías. Se realiza una comunicación serial robot-interfaz a través de bluetooth, haciendo posible el control del robot en forma inalámbrica y que el usuario conozca el estado entorno al robot desde la interfaz tornándose en un sistema tele-operado. El control del sistema lo realiza directamente el usuario, donde puede maniobrar con la velocidad y dirección del robot móvil. Este robot cuenta con sensores que posibilitan la ejecución de un algoritmo de seguridad.

El sistema está desarrollado de tal forma que distintos usuarios puedan hacer uso del mismo, ya que tiene un entorno amigable y es sencillo de operar gracias a que la interfaz incluye mensajes que ayudan al entendimiento del funcionamiento de la aplicación. Esta aplicación puede ser como un simulador básico de conducción parecido a un video juego simple, con la ventaja que no se requiere el uso de las manos en el funcionamiento de manejo, así personas que no pueden operar manualmente pueden ayudarse por esta aplicación, además se deja como premisa el uso de detección y reconocimiento facial para aplicaciones de control ya que es un campo en desarrollo que se constituye como el pilar fundamental de este proyecto.

PRESENTACIÓN

En el presente documento, realizado como parte de proyecto de titulación, se describe el desarrollo del software y hardware de un sistema de control, así como el fundamento teórico para la implementación de dicho sistema. Se divide en cinco capítulos que se describen a continuación.

El primer capítulo hace una introducción teórica basándose en las necesidades que derivan de los objetivos planteados para este proyecto. Se comienza por analizar el procesamiento de imágenes y video familiarizándose con conceptos que serán usados a lo largo del presente documento; principalmente en la detección de objetos, en particular de un rostro humano. También se hace una breve introducción a la tele-operación y sistemas hombre-máquina, conceptos relacionados directamente con el título de este proyecto. Por último se analiza brevemente la robótica móvil orientada al tipo car-like, modelo que se construye como parte del sistema.

En el segundo capítulo se detalla el desarrollo del software para la detección de un rostro humano, medio por el cual se realiza el control del sistema que se implementa. Se analizan las distintas posibilidades para cumplir con los requerimientos planteados que incluyen comunicación e interfaz gráfica de usuario. Se describe las librerías de OpenCV que son la base en la detección facial de igual manera que EmguCV. Como resultado del análisis de alternativas se elige una vía para desarrollar el software. Por último se describe el desarrollo de la aplicación incluyendo las herramientas y algoritmo utilizados.

En el tercer capítulo se describe la arquitectura general del sistema, incluyendo el hardware usado, su configuración y control implementado para el funcionamiento del robot móvil. Se hace referencia al diseño y selección del hardware, enfatizando en el robot móvil y la forma de efectuar el control del mismo integrándolo con la aplicación desarrollada para la detección de rostro e interpretación de movimientos. Además se diseña la instrumentación que posibilite tener el ambiente de manejo en la interfaz con el usuario.

En el cuarto capítulo se presentan los resultados tras poner en marcha la aplicación desarrollada se exponen lo obtenido al utilizar el sistema. Se analiza el comportamiento del robot bajo el control efectuado por el usuario tomando en cuenta la interacción que se tiene con el sistema, se comprueba la funcionalidad del sistema con distintos usuarios. Por último se detalla los costos involucrados en el desarrollo del presente proyecto.

En el Quinto y último capítulo se dan a conocer las conclusiones obtenidas tras la implementación del sistema en función de los objetivos planteados. De la misma manera se proponen recomendaciones enfocadas al sistema implementado así como para futuras aplicaciones similares.

CAPÍTULO 1

MARCO TEÓRICO.

1.1 INTRODUCCIÓN

Pensar en un sistema donde el humano pueda ejercer control en tiempo real sin la necesidad de contacto físico, ya no es pensar en ciencia ficción, gracias a los avances que se han dado en procesamiento de video; hecho, que ha servido de base para el desarrollo del presente proyecto, teniendo como motivo principal que una persona con alguna discapacidad motriz pueda formar parte de un sistema de control.

Por otra parte, la conducción vehicular como tal se ve rodeada de un gran número de problemas, sin embargo hoy en día es casi imposible prescindir de la misma; por ello el presente trabajo se enfoca en la robótica móvil tomando al manejo asistido como campo de aplicación. Es importante exteriorizar la ayuda que significaría conocer el comportamiento de un conductor en su cabina de manejo, ello permitiría que en base a este conocimiento se puedan efectuar gestiones de seguridad y control sobre el vehículo, por ello el presente trabajo busca figurar esta idea sobre un robot móvil, siguiendo la vía óptima para dar solución a un problema específico de control.

1.1.1 DESCRIPCIÓN DEL PROBLEMA

La gran mayoría de actividades cotidianas del humano promedio implican el uso de extremidades sea para levantarse de la cama, acicalarse, comer, movilizarse; esta última en particular hace que sea necesario el uso de un vehículo, en una civilización donde se requiere recorrer grandes distancias en un solo día.

Este hecho realmente no es un problema, dejando de lado el tema de la contaminación, tráfico y accidentes que el parque automotriz lleva de la mano, lo que hace que se requiera vehículos mucho más eficientes y seguros, pero ello no es un punto de análisis de esta tesis; sino más bien en cómo podría conducir una persona con discapacidad motriz.

La simulación de manejo apoyándose en robótica móvil, es solo una de las muchas aplicaciones que se puede imaginar, pues si se quiere simular un ambiente de manejo, en la robótica y tele-operación está la respuesta; pero el problema surge al preguntarse ¿cómo controlar la dirección de un vehículo sin necesidad de usar las manos?, esto nos conduce a plantear los objetivos generales y específicos de este proyecto que serán el lineamiento para la ejecución del mismo y se detallan a continuación.

Fig.1.1 Detección de rostro y mano en un ambiente de manejo [1].

1.1.2 OBJETIVOS

1.1.2.1 Objetivo General

Diseñar e implementar un sistema hombre – máquina para el manejo de un mini robot, donde las acciones de control se basan en el reconocimiento de movimientos de la cabeza.

1.1.2.2 Objetivos Específicos

- Desarrollar un software que permita hacer el reconocimiento de movimientos de la cabeza mediante procesamiento de video en tiempo real.
- Implementar los algoritmos de control necesarios, que permitan la tele operación de un mini robot móvil, garantizando la seguridad del mismo.
- Desarrollar una interfaz gráfica en donde el usuario pueda conocer la ubicación dentro del ambiente, en el que se encuentre un mini robot que emule un vehículo.
- Seleccionar y configurar los módulos necesarios para la comunicación tanto, en la transmisión inalámbrica del video captado por el robot hacia la interfaz del conductor, como las señales necesarias para controlar el robot.

1.2 PROCESAMIENTO DE IMÁGENES Y VIDEO

El procesamiento de imágenes en la actualidad está enfocado a un gran número de aplicaciones; tales como, cine, fotografía, reconocimiento, control inteligente y un sinfín de aplicaciones específicas, una de ellas que será tratada para este proyecto es el procesamiento de imágenes para reconocimiento facial.

Se realizará este proyecto basado en procesamiento de video debido a que el mismo es un método no invasivo, que como se verá al analizar los sistemas Hombre-Máquina, el confort del usuario deberá ser tomado en cuenta en pro de obtener un sistema confiable; antes de analizar la detección de rostro que es el pilar de la aplicación a desarrollar se hará una breve introducción a el procesamiento digital de imagen y video y algunos conceptos necesarios para una mejor comprensión.

1.2.1 CONCEPTOS FUNDAMENTALES.

Se dan a conocer algunos conceptos importantes, que se mencionan frecuentemente al trabajar con procesamiento de imágenes y video, al considerarlos trascendentales aunque pudiesen ser de conocimiento común.

1.2.1.1 Pixel

Se lo define como un punto físico en una imagen digitalizada, sea este el elemento más pequeño al que se puede acceder en una fotografía representada en una pantalla [2].

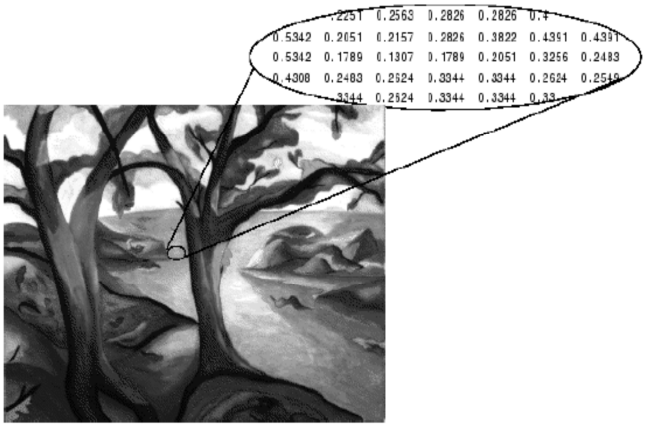


Fig. 1.2 Valores de pixel en escala de grises [4].

1.2.1.2 Bit por pixel (bpp)

Cada pixel puede ser representado por uno o varios bits, así se logra la representación en color; al tener un bit por pixel (1bpp) se tendrá la opción blanco o negro (monocromático) o a su vez al tener 24bpp $=2^{24}$ colores, se puede representar una amplia gama de colores (Truecolor) [2] [3].

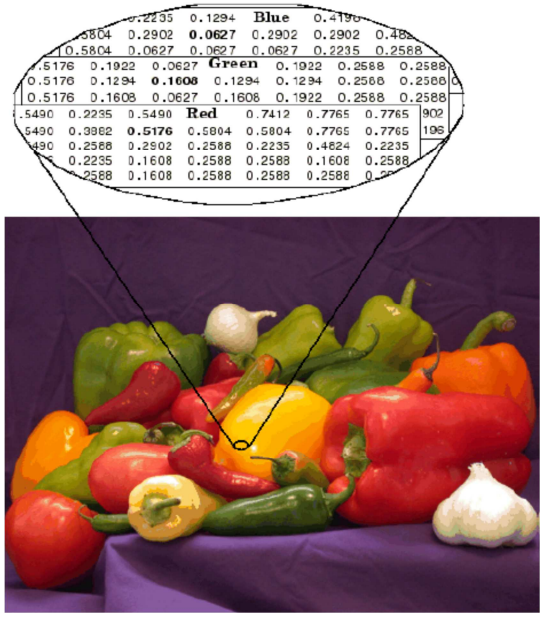


Fig.1.3 Los planos de color de una imagen Truecolor [4].

1.2.1.3 Red Green Blue (RGB)

Se puede tener un espacio de colores, al tener usualmente 16 bpp se puede dividir 5 bits para representar el rojo, 5 bits para representar el azul y seis bits para representar al verde [1], así se logra una combinación específica representada en RGB(red, green, blue) [3] [4].

Fig. 1.4 Representación de RGB [3].

1.2.1.4 Hue Saturation Value (HSV)

Es otro modelo de color cuyas siglas provienen del inglés (Hue, Saturation, Value /matiz, saturación, valor), se trata de una transformación no lineal del espacio de colores RGB. Es posible representar al matiz como una región circular, una región triangular exterior será la saturación y el valor del color así se puede seleccionar un color esta transformación se usa para hacer tracking por color y otras aplicaciones en el procesamiento de imágenes [5] [6].

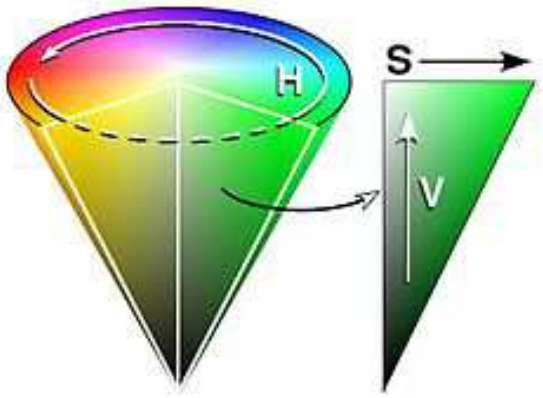


Fig. 1.5 Cono de colores del espacio HSV [5].

1.2.2 PROCESAMIENTO DIGITAL DE IMAGEN Y VIDEO

Es un conjunto de procesos que se ejecutan sobre una imagen digital con el objetivo de modificar la imagen o encontrar alguna información en ella, por lo mismo este es un campo de investigación abierto, en donde las investigaciones continúan a fin de mejorar calidad e innovar en aplicaciones.

1.2.2.1 Formatos De Imágenes De Mapa De Bit

Se conoce como imagen de mapa de bit a la representación de una imagen por puntos conocidos como pixeles, según el tipo de formato en el que se almacene una imagen se tendrá peor o mejor calidad y menor o mayor espacio de memoria requerida respectivamente.

- *Formato BMP*

Especificado de Microsoft, se usa para imágenes de alta resolución el tamaño ocupado en memoria es grande.

- *Formato GIF*

Permite imágenes de hasta 256 colores, usada en la generación de animaciones.

- *Formato TIF*

Almacena imágenes de alta calidad en formato comprimido, se tienen archivos generados que ocupan gran tamaño.

- *Formato JPG*

Muy utilizado como formato de almacenamiento, se puede comprimir en distintos niveles con lo que se define la calidad de la imagen.

- *Formato PNG*

Es una alternativa al GIF con mayor profundidad en el color.

- *Formato RAW*

Guarda los pixeles tal y como lo captura la cámara y sin pérdidas, son archivos muy grandes.

1.2.2.2 Imágenes Vectoriales

Las imágenes vectoriales permiten crear gráficos sin que su ampliación produzca pérdida de la calidad o distorsión de la misma. Se sustenta sobre cálculos matemáticos, de forma que los gráficos representados tienen unas ecuaciones asociadas. Una ampliación se realiza recalculando valores de las ecuaciones asociadas.

1.2.2.3 Representación de una Imagen

Para representar una imagen se lo hace mediante una función de intensidad de luz bidimensional $f(x, y)$, al representarla en forma digital se requiere discretizar esta función. Así, una imagen digital puede ser considerada como una matriz cuyos elementos indican el valor de un punto de la imagen (pixels), el tratamiento de las imágenes se lo divide el pasos como son; adquisición de la imagen, procesamiento de la imagen y presentación al usuario del resultado final [7].

Es importante entender la física de la adquisición de la imagen por los sensores (cámara, tomógrafo, etc.) incluso considerar la percepción del ojo humano; cabe destacar que otro consideración importante es la cantidad de luz, para poder describir cuantitativamente una imagen.

Una imagen puede ser representada en forma matemática, donde dicha función es definida considerando propiedades de cada punto de la imagen a esta se la conoce como Representación Determinística de una Imagen; también se puede representar por una función considerando las propiedades promedio de la imagen a esta representación se la conoce como Representación Estadística de una Imagen [8].

Sea $C(x, y, t, \lambda)$, la representación de la distribución de energía en el espacio de la fuente de imágenes de energía radiante cuyas coordenadas son (x, y) a un tiempo t y con una longitud de onda λ , esta función se la define como real positiva finita y se considera dentro de un rango A debido a las restricciones que puede tener el sistema de adquisición de datos.

$$0 < C(x, y, t, \lambda) < A \qquad (1.1)$$

Una imagen se la define en un rango acotado por ello se tiene,

$$-L_x \leq x \leq L_x \qquad (1.2)$$

$$-L_y \leq y \leq L_y \qquad (1.3)$$

Se visualiza una imagen en un intervalo de tiempo finito,

$$-T \leq t \leq T \qquad (1.4)$$

La respuesta de un observador humano ante una imagen se puede definir de la siguiente manera.

$$Y(x, y, t) = \int_0^{\infty} C(x, y, t, \lambda) V(\lambda) d\lambda \qquad (1.5)$$

Donde $V(\lambda)$ representa la función de eficiencia de luminosidad relativa, la cual es, la respuesta espectral de la visión humana, de la misma forma la respuesta de color se mide en base a una terna de valores que varían en forma lineal con respecto a la cantidad de color rojo verde y azul (RGB) teniéndose las siguientes expresiones:

$$R(x,y,t)=\int_0^{\infty} C(x,y,t,\lambda)R_s(\lambda)d\lambda \qquad (1.6)$$

$$G(x,y,t)=\int_0^{\infty} C(x,y,t,\lambda)G_s(\lambda)d\lambda \qquad (1.7)$$

$$B(x,y,t)=\int_0^{\infty} C(x,y,t,\lambda)B_s(\lambda)d\lambda \qquad (1.8)$$

Donde $R_s(\lambda), G_s(\lambda), B_s(\lambda)$ son respectivamente los valores de rojo, verde y azul [8].

Esta representación matemática puede complicarse según se tomen diferentes consideraciones, para el desarrollo de este proyecto solo se da una breve introducción al tratamiento matemático en el procesamiento digital de señales, puesto que profundizar en el tema está fuera del alcance del presente proyecto.

Por otra parte no se debe olvidar el hecho que toda la tecnología desarrollada en computadora está ligada directamente al mundo digital, por ello se menciona algunos tópicos importantes sobre el tratamiento digital de imágenes mediante matrices, donde las muestras con las que se trabajan son los pixeles.

Sea F la matriz que representa la imagen donde sus elementos se disponen como indica la Fig. 1.6 matemáticamente se la puede expresar como:

$$F=[F(n_1,n_2)] \qquad (1.9)$$

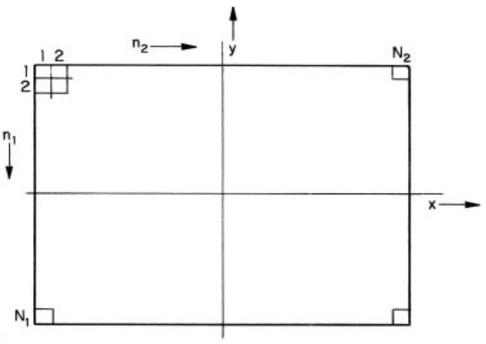


Fig. 1.6 Relación entre una imagen continua y su arreglo de muestras (pixels) [8].

Para propósito de análisis, algunas veces es conveniente trabajar en lugar de matrices con vectores sea por fila o columna barriendo los elementos de la matriz F y agrupando los elementos en un vector, este proceso se puede expresar en forma cuantitativa usando las matrices definidas como:

$$V_n = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} 1 \\ \vdots \\ n-1 \\ n \\ n+1 \\ \vdots \\ N_2 \end{matrix} \qquad N_n = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} 1 \\ \vdots \\ n-1 \\ n \\ n+1 \\ \vdots \\ N_2 \end{matrix} \qquad (1.9)$$

Entonces, la representación en vector de F viene expresada como;

$$f = \sum_{n=1}^{N_2} N_n F V_n \qquad (1.10)$$

En esencia el vector V_n extrae la enésima columna de F y la matriz N_n ubica esta columna en el enésimo segmento del vector f. La relación inversa para obtener la matriz F viene dada por la ecuación 1.11 expresada a continuación.

$$F = \sum_{n=1}^{N_2} N_n{}^T f V_n{}^T \qquad (1.11)$$

Con estas expresiones es posible trabajar tanto en forma vectorial como matricial representado por un arreglo bidimensional [8].

1.2.2.4 Procesamiento de Video

Al tratar video lo que se hace es procesar una trama de imágenes con el fin de simular una escena animada, la visión por computadora tiene como objeto de análisis, las imágenes, además constituye una de las más importantes tendencias dentro de la inteligencia artificial; es así como el procesamiento de una trama de imágenes por computador (Video) forman parte de la visión artificial [9].

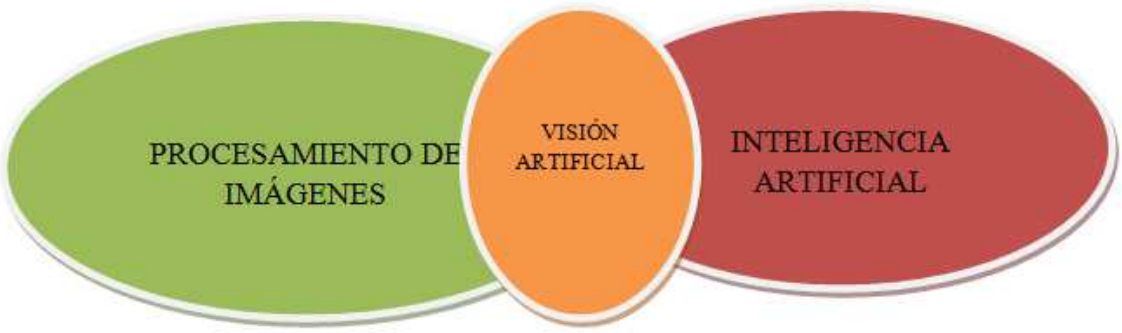


Fig.1.7 Interrelaciones de la Visión Artificial [9]

Es por ello que al tratar con video se siguen los mismos procedimientos que en el análisis de imágenes, siempre tratando de hacerlo de la manera más rápida en virtud de que no afecte la percepción del ojo humano; no se profundizará en lo que es procesamiento de video ya que se tiene una muy amplia cantidad de aplicaciones, técnicas y por ende conceptos no relevantes para la aplicación que se desarrolla; más bien se hará un enfoque de cómo se realiza la detección y reconocimiento de objetos, específicamente de un rostro humano, lo cual será una pieza clave para el correcto desarrollo del presente proyecto.

1.2.3 DETECCIÓN Y RECONOCIMIENTO DE UN ROSTRO HUMANO

Al hablar de reconocimiento facial se debe responder a las preguntas donde está el rostro y de quien es ese rostro cada una de estas preguntas implica plantear una solución individual, y ello ha sido tema de investigación por los últimos años lo que ha permitido tener aplicaciones cada vez más novedosas y sofisticadas basándose en las técnicas de detección y reconocimiento facial.

Detectar si dentro del panorama de una cámara se encuentra un rostro humano es el paso previo a reconocerlo o identificarlo, esta tarea se basa en comparaciones con patrones específicos de un rostro humano tales como su morfología, ubicación relativa de ojos nariz y boca entre sí, color característico de la piel y combinaciones de tales características que hacen algoritmos más sofisticados; se analiza brevemente los métodos más conocidos para detección de un rostro humano y que sirven como base en el desarrollo de la aplicación.
Para el año de 1991 Turk & Pentland publican el método de Eigenface, hacia 1997 Etemad & Che sacan a la luz el método de Fisherface, luego Viola-Jones en el año 2001 revolucionan el campo del reconocimiento facial al introducir los conceptos del Haar-Cascade y Ada-Boost que han servido de basa para el desarrollo de interesantes proyectos y es además este es uno de los métodos en que se basan las librerías de OpenCV que servirán para el desarrollo de la presente aplicación; todos estos son modelos basados en apariencia.

1.2.3.1 Eigenfaces

Se basa en el uso de algebra lineal donde cada imagen se almacena en un vector propio, busca reducir la dimensión de un espacio vectorial donde se almacena la imagen haciendo una transformación lineal, la idea de usar eigenfaces viene de la técnica desarrollada por Sirovich & Kirby (1987) y Kirby & Sirovich (1990) para una representación eficiente de imágenes de rostros usando Análisis de los Componentes principales (Principal Components Analysis PCA) [10] [11].

Se plantea en principio que cualquier imagen de un rostro puede ser reconstruida aproximadamente al almacenar una pequeña colección de pesos o ponderaciones para cada rostro y una base estándar de otras imágenes en las que existan rostros a las que se llama eigenpictures; en resumen se puede reconstruir una multitud de imágenes de rostros por la suma ponderada de una colección de eigenpictures ya que toda imagen de un rostro se puede representar como un vector; los pesos que se le asignan a cada imagen se adquieren en base a un entrenamiento.

Por otra parte todos los rostros lucen de la misma manera, dos ojos, nariz, boca ubicados bajo el mismo patrón, debido a esto los vectores de una imagen con un rostro constituyen un conjunto muy estrecho en el campo de las imágenes y tienen las mismas tendencias entre sí; este hecho posibilita que se use un espacio vectorial menor al de todas las imágenes, uno que solo tome a rostros como conjuntos, esta es la idea del PCA [11] [12].

Turk describe según [11] las operaciones para inicialización del proceso.

- Adquirir un conjunto de imágenes para entrenamiento.

- Calcular las eigenfaces del conjunto de entrenamiento, manteniendo solo las M imágenes que corresponden a los valores propios más altos, estas imágenes definen el espacio vectorial de los rostros; si se experimenta con nuevas imágenes los valores propios deben actualizarse o recalcularse.

- Calcular la ponderación para cada individuo conocido dentro del espacio de dimensión M, proyectando la imagen en el espacio de rostros.

Este proceso se lo puede ejecutar cada vez y cuando se tenga disponible tiempo de computo para reinicializar el proceso

Cuando el proceso se ha inicializado los siguientes pasos se realizan para reconocer los nuevos rostros de las siguientes imágenes.

- Calcular un set de pesos basados en la imagen de entrada y las M eigenfaces, proyectando la imagen de entrada en cada una de las eigenfaces.

- Determine si la imagen de entrada es de un rostro (detección) sea este conocido o no es importante saber si es un rostro, este es el paso más importante en el proyecto a realizarse, se determina si pertenece a un rostro o no al verificar si la imagen se aproxima o no al espacio de rostros o face-space determinado en la inicialización de proceso.

- Si la imagen es de un rostro se clasifica con los pesos como conocida o no conocida.

- Se actualiza las eigen-faces (opcional).

- Si la imagen es ingresada reiteradamente se la añade a la base de datos o conjunto de entrenamiento (opcional).

Debido a que se usa probabilidad y la dispersión se hace no solo entre clases sino dentro de las mismas esto hace que el método sea menos preciso además es sensible ante variaciones de luz [10].

En resumen Eigenfaces es un método estadístico que se basa en crear un espacio menor para tener una base de datos de imágenes con las cuales se pueda comparar la imagen o trama de imágenes que se desee procesar.

1.2.3.2 Fisherfaces

El discriminante lineal de Fisher es una técnica clásica en reconocimiento de patrones desarrollada por Robert Fisher en 1936, esta técnica ha sido aplicada en reconocimiento facial [10], del método anterior se debe resaltar el hecho de usar método lineal para la reducción de la dimensión del espacio cuando se trata de reconocimiento de un rostro, es posible usar un método lineal de clases específicas, este es el discriminante lineal de Fisher, el mismo trata de dar forma a la varianzas o dispersiones debidas al método estadístico con el fin de hacer un método más confiable cuando se clasifican las imágenes.

Fisherfaces es un método más preciso que Eigenfaces además usa menor cantidad de memoria y como se demuestra en [10] presenta mejores resultados ante variaciones en la iluminación, sin embargo el método aun no es tan confiable como para una aplicación en tiempo real donde existe movimientos de la cabeza, lo que implica imágenes de un rostro desde diferentes ángulos además de una velocidad relativamente alta y gran precisión.

1.2.3.3 Haar-Cascade y AdaBoost

Un "CASCADE" es una serie de "Haar-Like Feature" que se combinan para hacer un selector, y a su vez un "Haar Like Feature" se refiere a una característica que sigue un patrón determinado sea longitud, ángulos, bordes; lo que hace que en una imagen se denote las diferencias entre cada cuadrado, sin embargo analizar toda la imagen y más aún imágenes consecutivas (video) requiere una alta carga computacional, por ello el Ada Boost (Adaptative- Boosted) lo que hace es seccionar la imagen para optimizar el tiempo de cómputo.

Analizar una característica a la vez no brinda una buena confiabilidad por ello se analiza varias características en forma consecutiva (cascada) con ello se logra tener un algoritmo sólido y confiable lo que permitirá el reconocimiento de un rostro humano [13] [14].

El AdaBoost es un algoritmo de aprendizaje de máquina que combina varios clasificadores débiles pero efectivos para obtener uno poderoso y confiable, esto acompañado de que al usar Haar-Like Features se trabaja con rectángulos más grandes que los pixeles hace que el método sea además rápido haciendo el algoritmo ideal para usar en aplicaciones de tiempo real, con un procesador de 2Ghz es posible detectar rostros humanos a una taza de por lo menos 5 tramas por segundo, además se puede usar este algoritmo para detectar partes de un rostros como ojos nariz y boca entrenando al sistema [16], este algoritmo resultaría ideal para cumplir con los alcances propuestos en este trabajo.

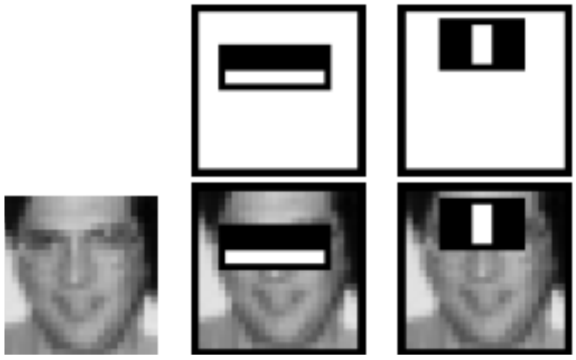


Fig. 1.8 Las primeras "features" seleccionadas por el AdaBoost [13].

1.3 TELE-OPERACIÓN

Uno de los objetivos específicos en este proyecto plantea implementar los algoritmos necesarios para la tele-operación de un robot, en otras palabras se desea operar un robot estando a cierta distancia; es importante por ello hacer una breve introducción sobre lo que es la tele-operación y su importancia en el desarrollo de la tecnología haciendo énfasis en su aplicación sobre sistemas robotizados.

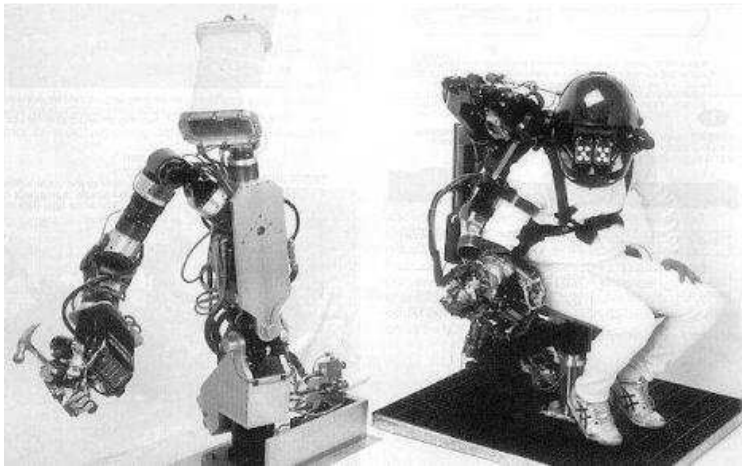


Fig. 1.9 Sistema Comercial Tele-operado [18].

Se puede definir a la tele-operación como un conjunto de tecnologías que comprenden la operación o gobierno a distancia de un dispositivo por un ser humano por ello un sistema de tele-operación será aquel que permita tele-operar un dispositivo tele-operado [17], en el caso puntual de este proyecto será un mini robot móvil que es que será operado desde un puesto remoto.

1.3.1 EVOLUCIÓN DE LA TELE-OPERACIÓN

La necesidad de efectuar tareas peligrosas pero necesarias que contribuyan al desarrollo social, impulso desde mucho tiempo atrás al hombre a buscar alternativas que permitan extender los alcances naturales de una persona; como trabajar con fuego u otras sustancias peligrosas para un humano. Es precisamente el peligro que representaba trabajar directamente con químicos nocivos dentro de la industria nuclear que lleva a Raymond Goertz a buscar desarrollar algún tipo de manipulador de fácil manejo a distancia haciendo una serie de prototipos (E1 – E4).

Los primeros tele-manipuladores fueron pasivos, es decir se transmitía la fuerza del operador por medios mecánicos esto hacia necesario uso de gran cantidad de energía del operador dependiendo de la aplicación, más tarde en pro de quitar carga física al operador se incorporó al motor eléctrico al sistema lo que hace que la cantidad energía que gasta el operador en realizar el control sea mínima.

Cada vez se incorporaba más tecnología en los sistemas de tele-operación, se introdujo a las aplicaciones marítimas y luego a espaciales con ayuda de cámara y sensores sofisticados, pero mientras más sofisticados los sistemas se querían iban apareciendo problemas, siendo de especial relevancia la presencia de retrasos en la comunicación; sin embargo la evolución en sistemas de cómputo, la gran cantidad de técnicas de control nuevas y la constante investigación en este campo ha permitido tener sistemas de tele-operación que mejoran día a día [17] [19].

Un sistema tele-operado implica la presencia de un operador humano aunque hoy en día se buscan sistemas autónomos insertando inteligencia al sistema a tal punto que se puedan tomar decisiones dentro de un computador, el campo de la tele-operación todavía tiene páginas por ser escritas donde los alcances son inimaginables que van de la mano con la robótica y automatización.

Fig. 1.10 Línea de tiempo en la tele-operación [19].

1.3.2 CONCEPTOS RELACIONADOS A TELE-OPERACIÓN

Para tener una vista amplia en este tema se dará a conocer algunos términos y su concepto que están ligados a la tele-operación, definiciones tomadas de [17].

1.3.2.1 Tele-manipulación

Conjunto de tecnologías que comprenden la operación o gobierno a distancia por un ser humano de un manipulador. Por tanto, tele-manipular no es más que hacer la tele-operación de un manipulador al que se denomina manipulador tele-operado.

1.3.2.2 Tele-robótica

Conjunto de tecnologías que comprenden la monitorización y reprogramación a distancia de un robot por un ser humano. Se hablará entonces de la tele-operación de un robot, que se denominará tele-robot o robot tele-operado.

1.3.2.3 Tele-presencia

Situación o circunstancia que se da cuando un ser humano tiene la sensación de encontrarse físicamente en el lugar remoto. La tele-presencia se consigue realimentando coherentemente al ser humano con suficiente cantidad de información sobre el entorno remoto.

1.3.2.4 Realidad Virtual

Situación o circunstancia que se da cuando un ser humano tiene la sensación de encontrarse en un lugar distinto de donde está físicamente, gracias a la información generada exclusivamente por un computador. El entorno que se genera, y en el que el operador se encuentra inmerso se denomina entorno virtual, y la situación de estar en él, también se conoce como presencia virtual.

1.3.2.5 Realidad Aumentada

Situación o circunstancia que percibe un operador cuando la información sensorial que le es realimentada de un entorno es modificada previamente por un computador con el objetivo de añadirle nueva información creada artificialmente, y que es no accesible directamente de la realidad por los sentidos del operador, aunque éste se encontrase en una zona remota.

1.3.2.6 Realimentación Táctil

Realimentación de la sensación de contacto aplicada a la piel. Es percibida por los receptores colocados cerca de la piel. Receptores que poseen un gran ancho

de banda (50 350 Hz) y que permiten detectar el primer contacto con el entorno, conocer la geometría de la superficie, su rugosidad y su temperatura.

1.3.2.7 Realimentación de Fuerzas

Realimentación de la sensación de una resistencia al avance o un peso que hace referencia a la excitación de los sensores colocados en los músculos y tendones, unidos a huesos y articulaciones, y que transmiten a la espina dorsal y al cerebro las tensiones y las fuerzas que se producen durante el movimiento (inerciales o de contacto). Se trata de receptores con poco ancho de banda y que proporcionan información sobre la fuerza total de contacto, así como el peso y deformabilidad de un objeto.

1.3.2.8 Realimentación Háptica

Realimentación de la sensación de contacto, ya sea de tipo táctil o de fuerzas. Hapteshai es un término griego clásico que significa tocar.

1.3.3 ELEMENTOS DE UN SISTEMA DE TELE-OPERACIÓN

Diseñar e implementar un sistema de tele-operación es la tarea principal de este proyecto por lo tanto se debe estar consciente de las partes que comprenden dicho sistema, para en los siguientes capítulos centrarse en el diseño de cada parte; se toma de [17] una definición básica de cada elemento que constituye el sistema de tele-operación añadiendo un aporte del autor.

1.3.3.1 Operador o tele-operador

Es un ser humano que ejecuta a distancia el control de las operaciones. Su acción puede ir desde un control continuo hasta una intervención por intervalos de tiempo donde sea necesaria una acción de control, con la que únicamente se ocupa de monitorizar y plantear objetivos y planes cada cierto tiempo.

Para el caso que compete el usuario será el operador en este caso ejecutará un control continuo por lo que forma parte permanente del sistema tele-operado.

1.3.3.2 Dispositivo tele-operado

Podrá ser un manipulador, un robot, un vehículo o dispositivo similar. Es la maquina que trabaja en la zona remota y que está siendo controlada por el operador.

Para el caso que compete será un robot tipo car-like que emula un vehículo, el mismo será operado por el usuario desde algún ordenador donde se ejecute la aplicación principal.

1.3.3.3 Interfaz

Conjunto de elementos que permiten la interacción del operador con el sistema de tele-operación. Se considera al manipulador maestro como parte del interfaz, así como a los monitores de vídeo, o cualquier otro dispositivo que permita al operador mandar información al sistema y recibir información del mismo.

Para el caso que compete se realiza una interfaz gráfica donde se visualiza el entorno del robot, esta interfaz se la realizará en pro de conseguir un ambiente visual parecido al de una cabina de conducción virtual.

1.3.3.4 Control y canales de comunicación

Conjunto de dispositivos que modulan, transmiten y adaptan el conjunto de señales que se transmiten entre la zona remota y la local. Generalmente se contará con uno o varias unidades de procesamiento.

Para el caso que compete se requiere transmitir hacia y desde el robot variables de control y señales del ambiente del mismo respectivamente; para ello se selecciona un módulo de comunicación inalámbrica del tipo bluetooth que se detalla posteriormente, además de uso de una cámara inalámbrica para la transmisión de video hacia la interfaz, el programa principal de control será implementado en una aplicación que se pueda correr en cualquier computador que cumpla los requerimientos para la ejecución de la misma; mayores detalles (CAPÍTULOS 2 y 3).

1.3.3.5 Sensores

Conjunto de dispositivos que recogen la información, tanto de la zona local como de la zona remota, para ser utilizada por el interfaz y el control.

Para el caso que compete los sensores principales son las cámaras, una que se instala en el robot para conocer su ubicación y otra que sirve para detectar los movimientos de la cabeza del usuario que serán usados como señales para ejecutar el control; mayores detalles (CAPÍTULO 3).

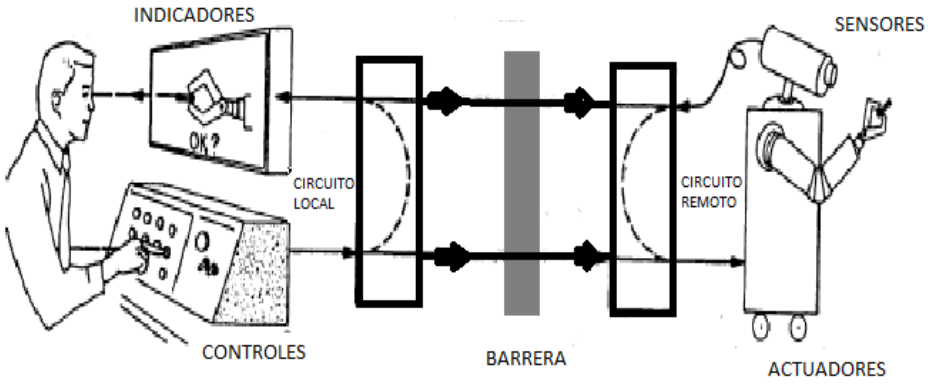


Fig. 1.11 Elementos de un sistema de tele-operación [17].

1.4 SISTEMAS HOMBRE-MÁQUINA

Como se dijo en la sección anterior el operador en un sistema generalmente es un ser humano a no ser que sea un sistema autónomo completamente, por ello el humano entra a ser parte del sistema sea de forma directa o indirecta, continua o discreta, pero siempre aportando en el control del sistema. Para la aplicación a implementar; el humano, en este caso el usuario o conductor, será parte integral del proceso ya que del comportamiento del mismo dependerá completamente el funcionamiento del sistema.

Es por ello que se cree conveniente estudiar la interacción entre el ser humano y la máquina formando un sistema, este campo ha sido uno de los de mayor interés dentro de la investigación por las importantes aplicaciones que se tienen tanto en industria, exploración, medicina y mucho más pues hoy en día las máquinas conviven con el hombre en todas las actividades cotidianas.

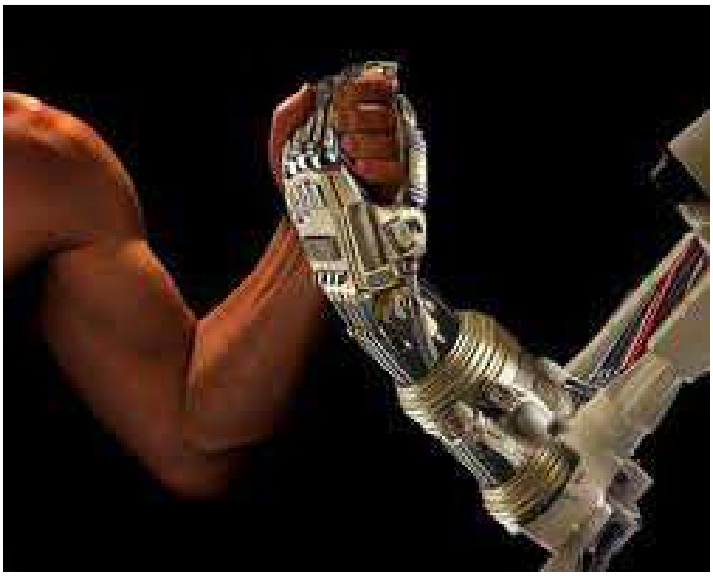


Fig. 1.12 Interacción Hombre-Máquina [20].

1.4.1 INTRODUCCIÓN A LOS SISTEMAS HOMBRE-MÁQUINA

Hoy en día los sistemas hombre máquina están directamente relacionados a los conceptos de robótica y tele-operación debido a que estos sistemas vienen casi siempre de la mano, es sí se podría definir como la cooperación entre personas y máquinas o componentes de físicos interactuando dentro de un entorno para alcanzar determinadas metas.

Por lo tanto una interacción entre el humano y la máquina se puede dar de forma manual, mecánica, o automática según la complejidad que la tarea a ejecutar amerite; siempre sacando provecho de las características positivas de cada uno de los entes que conforman el sistema así por ejemplo la capacidad de adaptarse al cambio que tiene el ser humano y la capacidad de hacer tareas que requieran grandes cantidades de energía que tiene la máquina.

El término “máquina” indica cualquier clase de sistema técnico y dinámico o una aplicación en tiempo real, lo cual incluye la automatización, el equipo de respaldo y software que permite la ejecución de diversas aplicaciones [22].

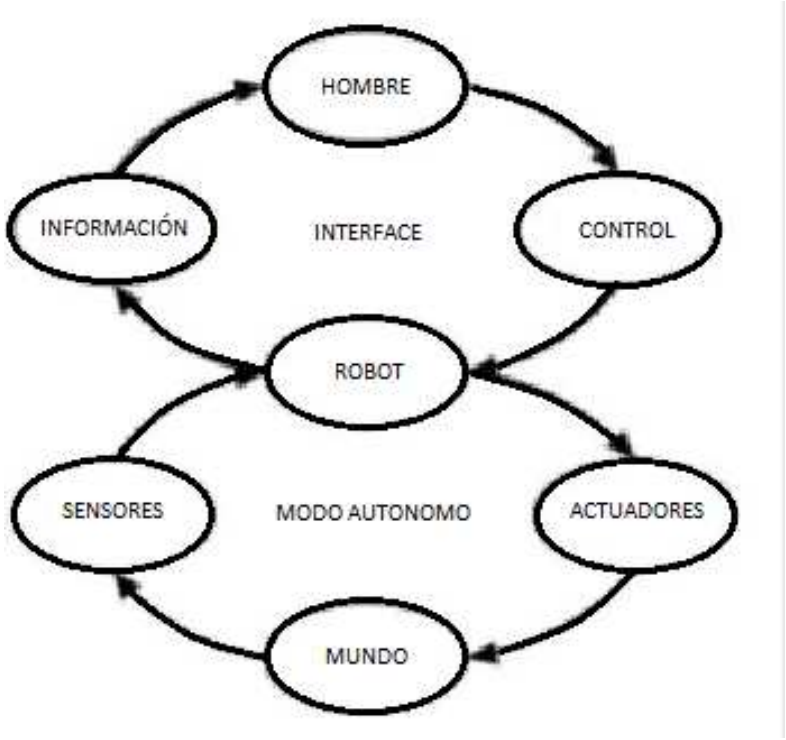


Fig. 1.13 Lazo de interacción Robot-Humano [21].

1.4.2 EFICIENCIA EN SISTEMAS HOMBRE-MÁQUINA

Un sistema hombre-máquina como todo sistema deberá ser eficiente en tal virtud según [21] para ejecutar un control óptimo es decir minimizar costos, se puede establecer una ley de control en base a la ecuación 1.12 bajo el supuesto establecido en la ecuación 1.13.

$$J(\pi) = E[\sum_{k} \Phi(s_{k+1}) + \Delta(\pi(s_k))] \qquad (1.12)$$

$$s_{k+1} = f(s_k, a_k) \qquad (1.13)$$

Donde $\Phi(s_k)$ es la función de costo del estado s_k y $\Delta(\pi(s_k))$ es la función de costo de la acción de control a_k, la suma indica la acción acumulada en el tiempo, en el modelo $f(s_k, a_k)$ se describe como la acción de a_k incide en un cambio de estado en un tiempo k, por último E indica la estimación lo que indica que es un modelo probabilístico.

Para implementar un sistema muy robusto se debería recurrir a métodos para modelar el comportamiento humano lo que hace que un sistema donde el humano está inmerso, sea más complicado de tratar, sin embargo es posible predecir el comportamiento del humano y el sistema cuando la aplicación lo amerite.

En la interacción robot-humano como se ve en la figura 1.13 el comportamiento del robot se produce en función de una ley de control que acepta a una acción humana como entrada, por ello se puede generalizar la noción de una ley de control que incluye en un lazo cerrado la interacción robot-humano, así que se puede remplazar el término conocido como "ley de control" por "plan de interacción" [22]

1.4.3 ERGONOMÍA Y CONFORT

Al ser un ser humano parte del sistema se debe tener muy en cuenta que este requiere un trato particular, pues el mismo sufre cansancio, aburrimiento, y otras emociones que inciden sean en forma positiva o negativa en el desenvolvimiento del sistema; por ello los conceptos de ergonomía y confort deben ser tomados en cuanta al diseñar un sistema hombre-máquina.

Según [24] se define ergonomía como una disciplina tecnológica que trata del diseño de lugares de trabajo, herramientas y tareas que coinciden con las características fisiológicas, anatómicas, psicológicas y las capacidades del operador, y etimológicamente significa ley o ciencia del "trabajo", todo ello para dar confort al usuario y por consiguiente mejorar el desempeño de todo el sistema; en el presente trabajo se deberá hacer la aplicación de tal forma que el usuario se sienta cómodo al usarla teniendo en cuenta estos factores.

Fig. 1.14 Sistema Hombre Máquina en ambiente de conducción [24].

1.5 ROBÓTICA MÓVIL

Parte importante de este proyecto está en incorporar al sistema un robot que emule un vehículo. Hablar de robótica sería un tema muy extenso por la gran cantidad de información que se puede encontrar y todo el trabajo de investigación que se hace en este campo alrededor del mundo, por ello solo se dará una breve introducción a lo que es la robótica móvil enfocándose en el robot tipo car-like que será parte del sistema a desarrollar.

1.5.1 INTRODUCCIÓN A LA ROBÓTICA MÓVIL

Desde el principio de los tiempos el hombre buscó movilizarse distancias cada vez más grandes en el menor tiempo posible ello condujo a la necesidad de buscar medios de transporte cada vez más novedosos, así surge los automóviles que sufriendo un proceso de evolución llegan a ser cada vez más seguros confortables y eficientes pero aún no lo suficiente, esto motiva a la necesidad de hacer investigaciones a escala donde son pequeños robots que cumplen el papel de un vehículo para simular los avances en esta materia.

El tipo de entorno donde el robot esté trabajando es muy importante para su diseño ya que puede ser hostil y requerir protecciones para ello se debe verificar si el robot es para uso en interior o exterior y luego si el entorno es estructurado, donde existe poco o ningún movimiento de otros agentes; o por el contrario, no estructurado, donde existe movimiento de agentes externos que pueden afectar el comportamiento.

Dentro de la robótica móvil existen diversos sistemas que permiten la locomoción sea por aire, tierra o agua, y para cada uno los sistemas de locomoción son varios y su selección depende de la aplicación a desarrollar. En un robot tipo car-like como es de esperar el sistema de locomoción serán ruedas por ello de su nombre. Es el más parecido a un auto móvil real en cuanto a tracción y dirección, por ello se analiza el modelo matemático de este tipo de robot, que será implementado para emular un vehículo a ser controlado.

1.5.2 MODELO CINEMÁTICO DE UN ROBOT MÓVIL.

Tomando como referencia [25], [26] y [27] se puede se puede detallar las ecuaciones que gobiernan el comportamiento de un robot móvil bajo el supuesto que sus ruedas no derrapen con respecto a la superficie en la que se mueve, se llega a la ecuación 1.14 donde x e y son las coordenadas en el plano sobre el que se mueven las ruedas y θ es la orientación de la rueda.

$$\begin{bmatrix}\sin\theta & \cos\theta & 0\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix}=0 \quad (1.14)$$

Como consecuencia de todas las generalizaciones posibles se obtiene la ecuación 1.15 donde v_1 y v_2 son respectivamente la velocidad lineal de la rueda y su velocidad angular alrededor del eje vertical; a esta ecuación se la conoce como el modelo cinemático de un uniciclo.

$$q=\begin{bmatrix}\cos\theta\\ \sin\theta\\ 0\end{bmatrix}v_1+\begin{bmatrix}0\\ 0\\ 1\end{bmatrix}v_2 \quad (1.15)$$

Para extender este modelo al robot tipo car-like nos remitimos a la Fig. 1.15, donde se llega a las restricciones para cada rueda expresadas en las ecuaciones 1.16 y 1.17, donde x_f y y_f denotan las coordenadas cartesianas de la rueda delantera; y las ecuaciones 1.18 y 1.19 describen las restricciones al trabajar como cuerpo rígido, donde l es la distancia entre las ruedas de ello se tiene la ecuación 1.20 como la primera restricción cinemática.

$$\dot{x}_f\sin(\theta+\phi)-\dot{y}_f\cos(\theta+\phi)=0 \quad (1.16)$$

$$\dot{x}\sin\theta-\dot{y}\cos\theta=0 \quad (1.17)$$

$$x_f=x+l\cos(\theta) \quad (1.18)$$

$$y_f=y+l\sin(\theta) \quad (1.19)$$

$$\dot{x}\sin(\theta+\phi)-\dot{y}\cos(\theta+\phi)-\dot{\theta}l\cos\phi=0 \quad (1.20)$$

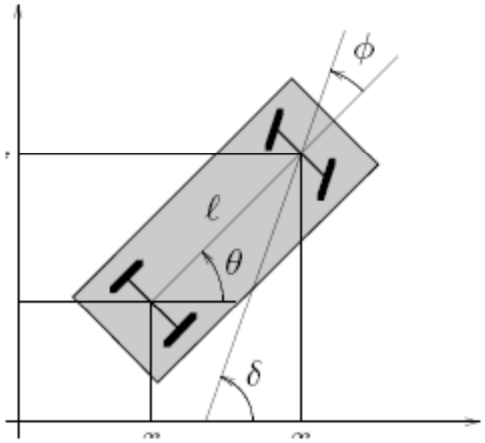


Fig. 1.15 Coordenadas de un robot car-like [27].

Por último se presenta la matriz de restricciones (1.21), modelo cinemático si el trabajan las ruedas traseras (1.22) donde v_1 y v_2 son las entradas de velocidad de tracción y dirección respectivamente, y el modelo cuando funciona las ruedas delanteras (1.23) donde v_1 y v_2 se refieren a las ruedas delanteras.

$$C(q)=\begin{bmatrix}\sin(\theta+\phi) & -\cos(\theta+\phi) & -l\cos\phi & 0\\ \sin\theta & -\cos\theta & 0 & 0\end{bmatrix} \qquad (1.21)$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{\phi}\end{bmatrix}=\begin{bmatrix}\cos\theta\\ \sin\theta\\ \tan\phi/l\\ 0\end{bmatrix}v_1+\begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix}v_2 \qquad (1.22)$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{\phi}\end{bmatrix}=\begin{bmatrix}\cos\theta\cos\phi\\ \sin\theta\cos\phi\\ \sin\phi/l\\ 0\end{bmatrix}v_1+\begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix}v_2 \qquad (1.23)$$

Así se describe un modelo que da una buena aproximación a la realidad cuando se trabaja con robots de tamaño pequeño, donde no es necesario analizar la dinámica del sistema ya que ello conlleva a tener mayor dificultad. En los posteriores capítulos se usaran conceptos mencionados para la implementación de todo el sistema para cumplir los objetivos planteados.

CAPÍTULO 2

DISEÑO DEL SOFTWARE PARA PROCESAR VIDEO E INTERFAZ GRÁFICA.

Un objetivo del presente trabajo es tener una aplicación cuyo código sea abierto, es decir se tenga acceso al mismo para futuros cambios, mejoras o adaptación a otras aplicaciones; por ello es importante elegir correctamente la plataforma sobre la cual se realizará el reconocimiento de movimientos de la cabeza; así como el lenguaje en el que será escrita la aplicación principal.

Como se mencionó en el capítulo anterior cada algoritmo para detección de un rostro humano mientras más preciso este sea, demanda de igual manera mayor trabajo el procesamiento de la información lo que concluye en la necesidad de mayor velocidad, este hecho es vital al momento de escoger el algoritmo y lenguaje a utilizar pues se requiere que el proceso se ejecute en tiempo real, sin malgastar recurso alguno.

Otro requerimiento es tener un entorno gráfico que permita la interpretación del usuario, como ya se ha manifestado el humano es parte del sistema, por ello se requiere que la plataforma con la que se trabaje en el proyecto tenga un entorno gráfico o sea compatible con otra aplicación que permita la programación con herramientas visuales, además es necesario que sea viable la comunicación en tiempo real con los distintos periféricos que conforman el sistema.

Bajo estos supuestos se analizan varias alternativas poniendo como premisa el uso de las librerías de OpenCV, pues serán tomadas como base para la detección del rostro del usuario e interpretación de movimientos, debido a la versatilidad con la que cuentan, la amplia documentación entorno a la misma y al ser librerías de código abierto.

2.1 SOFTWARE PARA PROCESAR VIDEO

De antemano se analizan los distintos programas que pueden ser usados para desarrollar la detección del rostro del usuario e interpretar los movimientos de su cabeza, dicho análisis se lo realiza de forma general para los distintos compiladores y se efectúa la selección en función de las bondades que presente cada uno.

2.1.1 MATLAB R2013a

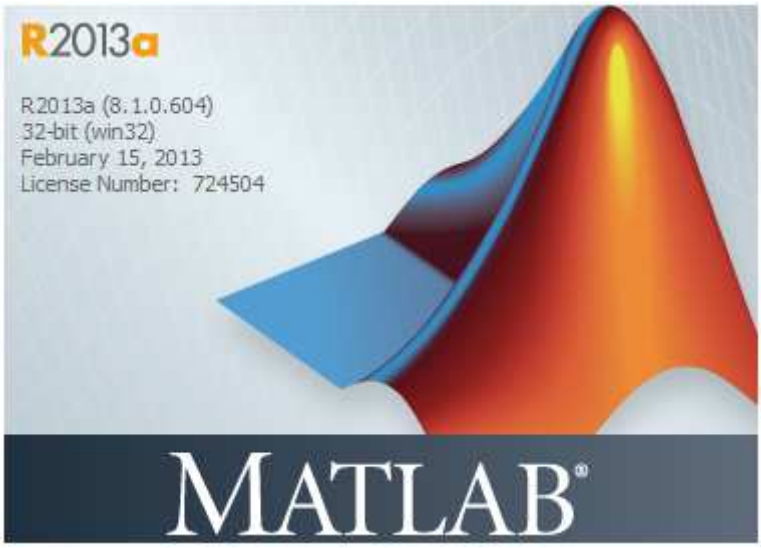


Fig. 2.1 Logotipo de Matlab R2013a

Esta por demás mencionar el amplio panorama que ofrece Matlab para aplicaciones de control, apoyado en las herramientas Toolbox y Simulink; la alternativa que presenta este software para la aplicación es usar las librerías de OpenCV trabajando en base a MexFiles o ejecutables, además se puede apoyar en los Toolbox Computer Vision System Toolbox [28] e Image Processing Toolbox [29] que se han mejorado para las versiones más recientes de MATLAB, en las referencias para usuario y página oficial de Mathworks se puede encontrar gran información para trabajar con estos Toolbox, sin embargo Matlab no procesa rápidamente como para llevar una aplicación en tiempo real conjuntamente con la adquisición de imágenes por lo que sería necesario convertir el código generado en Matlab a otra aplicación para ganar en velocidad de procesamiento de la información pero para una aplicación compleja pudiere tornarse muy laborioso.

La ventaja del Matlab es su fácil manejo en cuanto a librerías además de un entorno gráfico amigable, además de la posibilidad de comunicación por distintos protocolos como RS-232 o TCP/IP; además se cuenta con un Toolbox de realidad Virtual que sería ideal para la aplicación en cuestión, haciendo posible el desarrollo de una aplicación realista.

2.1.2 MICROSOFT VISUAL STUDIO 2010

Fig. 2.2 Logotipo de Microsoft Visual Studio 2010 Ultimate

Visual estudio 2010 es una herramienta para programación que soporta lenguaje C, C++, C# (C Sharp) y basic lo que permite un procesamiento rápido del código, ideal para la aplicación planteada, funciona muy bien para cualquier versión de OpenCV y posibilita comunicación por el puerto serial.

Este IDE crea como proyecto, una solución, la misma puede ser una aplicación de distinto tipo sea; una aplicación de Consola (Líneas de Comandos), una aplicación Windows Forms (Interfaz Grafica de Usuario), entre otras. Sin embargo el inconveniente que se presenta con este software, es que por sí solo no tiene entorno gráfico para C++, que es el lenguaje en el que se trabaja con OpenCV; es decir, en un principio se está limitado a usar OpenCV solo con aplicaciones de consola.

Pero es posible hacerlo con la ayuda de librerías (Qt) o trabajar en lenguaje compatible con interfaz gráfica de Windows como C# o basic en una aplicación Windows Forms, llamando a las funciones y clases de OpenCV con la ayuda de la herramienta EmguCV.

Se puede usar programación estructurada y dividir el proyecto en partes o trabajar con programación orientada a objetos usando las clases para procesamiento de video que usa OpenCV o EmguCV que se detallan más adelante. Finalmente se cuenta con vasta documentación en proyectos similares lo que sería de gran ayuda para la ejecución del proyecto.

2.1.3 PYTHON

Fig. 2.3 Imagen de Portada de [30]

El concepto de software libre esta intrínsecamente ligado al hecho de tener una aplicación donde el código sea susceptible a modificaciones por cualquier persona, por ello Linux es un sistema operativo donde impera este concepto, de aquí nace la idea de realizar la aplicación sobre una plataforma como Python, que es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 , cuyo nombre se inspira en el grupo de cómicos ingleses “Monty Python”; al trabajar en este entorno se tiene una sintaxis muy limpia que favorece un código legible.

Se trata de un lenguaje interpretado o de script (se ejecuta utilizando un programa intermedio llamado intérprete), con tipado dinámico (no es necesario declarar el tipo de dato que va a contener una determinada variable), fuertemente tipado (no se permite tratar a una variable como si fuera de un tipo distinto al que tiene), multiplataforma (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS,) y orientado a objetos [30].

Una ventaja es que existe la aplicación disponible también para Windows lo que elude el hecho de trabajar en el entorno de Linux, además se tiene disponible vasta documentación, el problema de usar esta plataforma esta en el entorno no amigable y necesidad de aprender programación orientada a objetos que es la que predomina al usar este código, otro inconveniente es la necesidad de usar un intérprete para la ejecución del código que en un principio resultaría laborioso.

2.1.4 SELECCIÓN DEL IDE

La decisión trascendental en el desarrollo del proyecto está en elegir el lenguaje y la plataforma sobre la cual se desarrollará el procesamiento de video, se han analizado las alternativas más promisorias llegando a la conclusión que se podría desarrollar la aplicación sobre cualquiera de los distintos IDE (Integrated Development Enviroment) planteados aunque con sendos conflictos.

Como ya se han desarrollado proyectos en procesamiento de video en la EPN utilizando como base Matlab, se propone hacer un producto diferenciado lo que deja como alternativa Visual Studio o Python, apoyándose en el hecho que Python en principio no es muy amigable y el poco conocimiento de este entorno conduce a descartar esta opción. Sin embargo podría dejarse abierta la opción de implementar este proyecto a futuro ya sea en Matlab o Python ya que ambas herramientas tienen gran potencial en materia de programación, por otra parte no se descarta la opción de usar estas herramientas como auxiliares en alguna parte durante la realización de este proyecto.

Se elige trabajar con Microsoft Visual Studio 2010 Ultimate ya que es compatible para versiones de Windows Vista, 7 y posteriores que son las versiones que predominan en uso a la fecha de desarrollo del presente proyecto, por ello será el entorno donde se ejecute el proyecto puesto que una versión actualizada de Visual Studio es compatible solo para la última versión de Windows Microsoft (Windows 8) dificultando el desarrollo en cuestión de compatibilidad en ordenadores.

2.2 DESCRIPCIÓN DE LIBRERÍAS

Para detección de objetos y tracking se cuenta con las librerías de OpenCV que son de código abierto, se disponen varias versiones que difieren un poco unas de otras, un hecho importante es que las versiones más actuales poseen librerías ya compiladas por lo que no es necesario usar un compilador auxiliar como C-Make sin embargo existe amplia documentación sobre cómo llevar a cabo la creación de nuestras propias librerías. Se analiza la versión más actual a la fecha de inicio del presente proyecto sea esta OpenCV 2.4.5.

Como se ha mencionado las versiones más actualizadas de OpenCV usan tipos de variables y funciones propias, algunas difieren de las versiones anteriores, por ello se hará un breve resumen tanto de las variables como funciones principales que se requieren para poder trabajar con las librerías de OpenCV dentro del entorno de Microsoft Visual Studio 2010.

Por otra parte la aplicación requiere un entorno visual, para lograr este acometido se podría recurrir a la biblioteca de Qt que cuenta con una API (Application Programming Interface) independiente y se usa principalmente para aplicaciones que requiere Interfaz de Usuario, o usar EmguCV que permite el uso de las funciones y clases de OpenCV para trabajar en el ambiente gráfico de Visual Basic C#/.NET o Basic.Net como una aplicación de Windows Forms, en donde además la comunicación serial es relativamente sencilla.

Será necesario hacer un resumen del modo de empleo y bondades de estas librerías sobre Microsoft Visual Studio ya que la aplicación será ejecutada sobre este IDE, por otra parte se hace un enfoque en el lenguaje de programación a usar, en virtud de cumplir a cabalidad los objetivos de la aplicación en cuanto a detección de movimientos de la cabeza, interfaz y comunicación, requerimientos ineludibles para este proyecto.

2.2.1 OPEN CV

Fig. 2.4 Logotipo de OpenCV

La versión más actual de OpenCV a la fecha de implementación del proyecto es OpenCV 2.4.5 y por ser una librería en código abierto, está disponible para descarga gratuita en su página oficial [31]; la misma que describe que OpenCV está liberada bajo la licencia BSD (Berkeley Software Distribution) y por lo tanto es libre tanto para uso comercial como para académico.

Esta librería tiene interface para C++, C, Python y Java además soporta Windows, Linux, Mac OS, iOS y Android. Open CV fue diseñado para eficiencia computacional y con un profundo enfoque en las aplicaciones en tiempo real. Escrito en lenguaje C, C++ optimizado, la librería puede tomar ventaja del procesamiento multi-core [31].

Acogida alrededor de todo el mundo, OpenCV tiene más de 47 mil personas como usuarios y se estima que el número de descargas excede los 6 millones. Usada en arte interactivo, inspección de minas, mapas en la web o robótica avanzada [31] como es el caso del presente proyecto, donde OpenCV será usado como herramienta base para la detección de la cabeza (rostro) del usuario ya que esta herramienta cuenta con funciones y clases propicias para efectuar este acometido; por ello se detalla el uso de esta librería enfocada en la aplicación a efectuar.

2.2.1.1 Instalación de OpenCV.

Como se ha mencionado es posible descargar de forma gratuita esta herramienta en [13], luego de elegir la pestaña de downloads se elige la versión y el sistema operativo en donde será usada, para el caso particular de este proyecto se trabajará en Windows al ser el más familiar, una vez descargada esta librería se procede a la instalación.

Para usar este paquete se tienen dos opciones; instalarlo usando las librerías pre-compiladas o realizando nuestras propias librerías, siendo la primera mucho más sencilla y rápida sin embargo, al recurrir a esta vía rápida, las librerías compiladas solo funcionarán con una versión actual de Microsoft Visual Studio y además no se tendrá ventaja sobre los avances que se han implementado en las últimas versiones como el uso de multi-core que permite ejecutar procesos en paralelo con el fin de optimizar una aplicación en tiempo real.

Aunque la complejidad de nuestra aplicación no es tan elevada para requerir ejecutar varios procesos en paralelo, sería conveniente compilar nuestras propias librerías personalizadas en el caso de requerir trabajar con Qt, para lo cual este requisito es ineludible; además de tener una idea de cuál es la diferencia entre las dos opciones sea para el presente o futuro proyecto.

2.2.1.1.1 Instalación Usando Librerías Pre-compiladas

Una vez descargada la librería se la debe descomprimir en un directorio conocido, pues será necesario conocer la dirección donde se instala la carpeta de OpenCV para futuras acciones, se recomienda instalar en una carpeta titulada OpenCV en un disco local del ordenador por ejemplo C: \OpenCV, allí se guardaran todas las subcarpetas que pertenecen a las librerías, incluidas las pre-compiladas.

Como ya se tiene las librerías, resta establecer la variable de entorno que permite trabajar de forma más sencilla para ello se acude a: panel de control / sistema y mantenimiento/ sistema; luego se accede a “configuración avanzada del sistema” y en el cuadro de diálogo que aparece se selecciona variable de entorno y en variables de sistema se crea una nueva variable donde su nombre sería

“OPENCV_BUILD” y el valor “C:\opencv245\build, por otra parte en la variable del sistema “Path” se añade el camino de la librería ya compilada %OPENCV_BUILD% \OPENCV245\build\x86\vc10\lib; con ello se tiene instalado OpenCV para ser usado en IDE requerido.

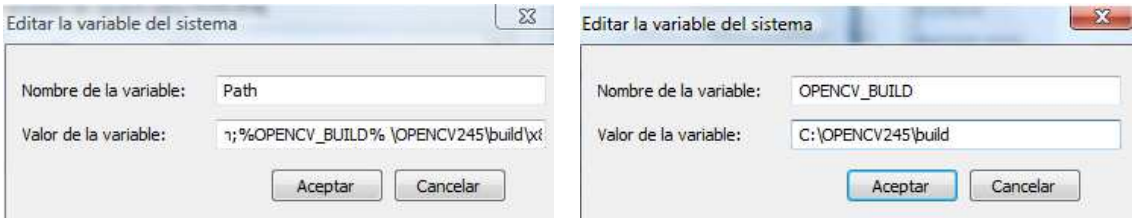


Fig. 2.5 Edición de las variables del sistema para OpenCV

2.2.1.1.2 Instalación Compilando Librerías Personalizadas.

El IDE usado para crear las librerías puede ser cualquiera que trabaje con lenguaje C++, por ejemplo Microsoft Visual Studio 2010, con el que se desarrolla el presente proyecto, además será necesario descargar e instalar CMake, herramienta requerida para personalizar las librerías y hacer los archivos de las mismas, además se deben descargar e instalar las herramientas necesarias para tomar ventaja de las bondades de las librerías para desarrollar aplicaciones específicas.

Según [32] detalla, las herramientas que pueden ser usadas al personalizar nuestras librerías, pueden ser:

-Python Libraries: Para paralelizar segmentos de código y asegurar que se tome ventaja de todos los núcleos del procesador.

-Numpy: Es un paquete computacional de carácter científico, usado en la interfaz de python.

-Intel Threading Building Blocks (TBB): También se usa para correr segmentos de código en paralelo hacienda uso de todos los núcleos del CPU.

-Intel Integrated Performance Primitives (IPP): Se usa para mejorar la conversión de color y en Haar Training, sin embargo esta herramienta no se puede usar en forma gratuita.

- Qt framework: Es una herramienta gráfica que se profundiza más adelante en la sección 2.2.3 de este documento.
-Eigen: es un template C++ para aplicaciones que requieran del uso de algebra lineal.
-CUDA Toolkit: Mejora la funcionalidad de los algoritmos de OpenCV usado en la GPU (Graphics Processing Unit) lo que ayuda a optimizar el proceso.
-OpenEXR: Se requieren para el trabajo con formato de imagen HDR (High Dinamic Range)
-OpenNI Framework: Contiene un conjunto de aplicaciones en código abierto en donde se tiene una interacción natural con dispositivos mediante métodos como reconocimiento de comandos de voz, señas o seguimiento de movimiento corporal.
-Miktex: Para implementar la documentación de OpenCV en formato TEX.
-Sphinx: genera la documentación de Python y OpenCV.

De las herramientas seleccionas, las más importantes se consideran Threading Building Blocks y Qt Framework, necesarias para aprovechar al máximo el procesador a usarse y desarrollar la interfaz gráfica para el usuario, además se toma ventaja del hecho de que estas aplicaciones son libres.

Como primer paso se abre CMake y selecciona los directorios donde se extrajo los archivos de OpenCV 2.4.5 que se descarga previamente de [31] y se configura según el compilador a usar para este propósito VS2010 como indica la Fig. 2.6.

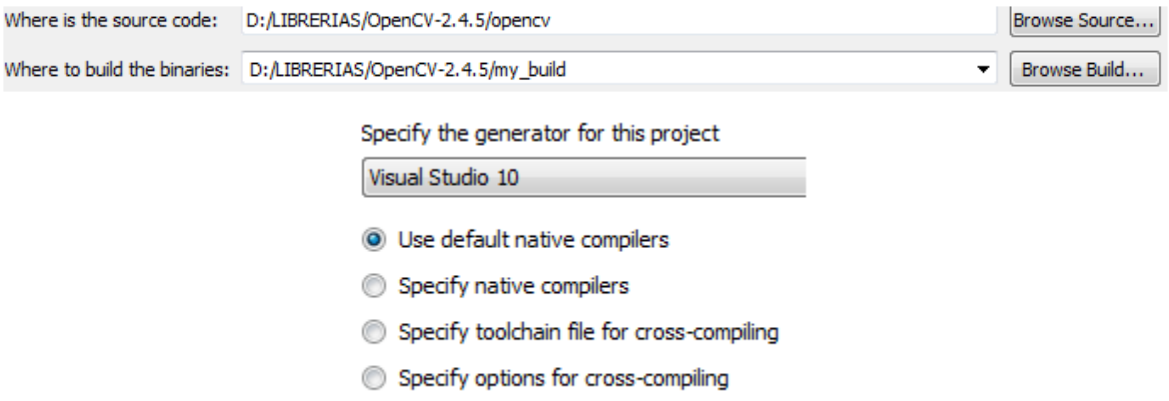


Fig. 2.6 Configuración Inicial en CMake

En el menú que aparece se elige la opción WHIT_TBB y WITH_QT, para ello previamente se debe descargar las librerías de TBB de la página de Intel, y extraerlas en una dirección conocida; a su vez se debe tener instalado una versión de Qt4, con ello se configura y se genera la solución en CMake. Si es requerido se deberá establecer las direcciones donde se encuentran instaladas las herramientas que se usan para la personalización de OpenCV, luego se genera nuevamente y se verifica si se ha generado los archivos requeridos.

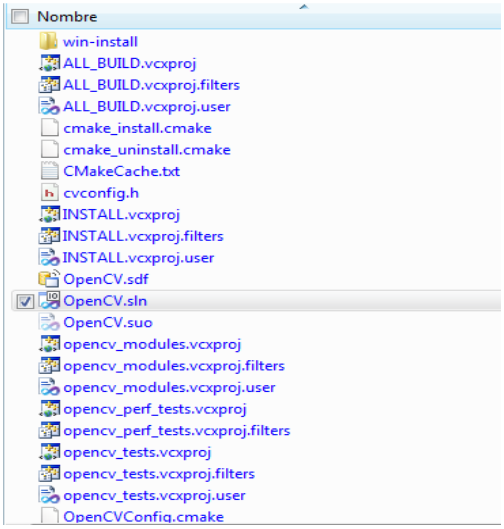


Fig. 2.7 Archivos de OpenCV generados con CMake.

En el directorio escogido se genera una solución para VS2010, se debe compilar las librerías personalizadas con el mismo IDE, para ello se abre la solución generada y se inicia la depuración. Resta establecer la variable de entorno con las nuevas librerías tal como se lo cita en la sección anterior, todo este proceso toma mucho tiempo pero es necesario.

2.2.1.1.3 Uso de OpenCV con Microsoft Visual Studio

Para poder realizar una aplicación usando OpenCV se debe generar una hoja de propiedades para el proyecto, en la misma se establece las direcciones de los archivos que serán incluidos a la solución cuando se trabaje con OpenCV, sea con las librerías pre-compiladas o con las librerías personalizadas, para ello se debe tener muy en cuenta en donde se han almacenado los archivos de OpenCV.

No se debe olvidar establecer la variable de entorno antes de nada. En VS2010 OpenCV trabaja para aplicaciones de Consola, por ello se debe crear un nuevo proyecto vacio de este tipo para visual C++. En el administrador de propiedades, se elige Debug y se añade una nueva hoja de propiedades que servirá para todos los proyectos que requieran el uso de las librerías de OpenCV en una aplicación de este tipo.

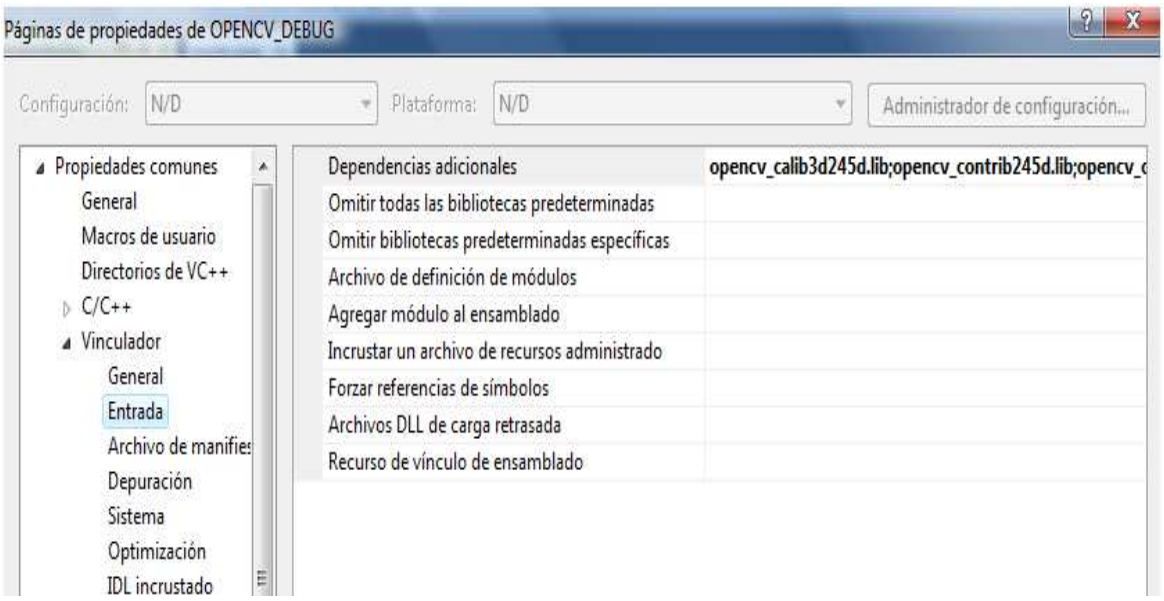


Fig. 2.8 Página de propiedades de OpenCV.

Una vez creada la nueva hoja de propiedades del proyecto se la accede a la ventana de configuración al dar click derecho sobre la misma y se elije la pestaña propiedades. Aparece una ventana similar a la de la figura 2.8 donde se debe configurar de la siguiente manera:

-Se accede a la pestaña: Propiedades comunes, C/C++, General, Directorios de inclusión adicionales; se procede a editar y se añade la dirección “$(OPENCV_BUILD)\include”, donde OPENCV_BUILD es la variable de entorno establecida según en el apartado 2.2.1.1.1.

-Luego en la pestaña: Vinculador, General, directorio de bibliotecas adicionales se edita y se añade la dirección “$(OPENCV_BUILD)\x86\vc10\lib”.

-Por último en la pestaña: Vinculador, Entrada, Dependencias adicionales se edita y asigna las dependencias:

opencv_calib3d245d.lib
opencv_contrib245d.lib
opencv_core245d.lib
opencv_features2d245d.lib
opencv_flann245d.lib
opencv_gpu245d.lib
opencv_haartraining_engined.lib
opencv_highgui245d.lib
opencv_imgproc245d.lib
opencv_legacy245d.lib
opencv_ml245d.lib
opencv_nonfree245d.lib
opencv_objdetect245d.lib
opencv_photo245d.lib
opencv_stitching245d.lib
opencv_superres245d.lib
opencv_ts245d.lib
opencv_video245d.lib
opencv_videostab245d.lib

Estos son los archivos contenidos en el directorio OPENCV_BUILD \build\x86\vc10\lib aunque no todos son necesarios para la aplicación en desarrollo.

La configuración de OpenCV para VS2010 esta lista y se puede probar ejecutando el código de una aplicación sencilla como la siguiente la cual accede al video de una cámara web y lo muestra en una ventana. El código y comentarios se muestra a continuación en donde las primerias líneas son siempre requeridas para acceder a las librerías de OpenCV.

```
#include "opencv2/opencv.hpp"
using namespace cv; //directiva para el uso de clases y funciones de OpenCV

int main(){

Mat imagen; //Se crea un objeto donde se almacena cada imagen de la trama
VideoCapture video; //Se crea un objeto para captura de video
video.open(0); //Se accede al video de la cámara
namedWindow("WEBCAM",1); //Se crea una ventana para mostrar la imagen
while(1)
{
video>>imagen; //Se asigna una trama la webcam a una imagen
imshow("WEBCAM",imagen); //Se muestra la imagen en la pantalla
waitKey(16); //retardo para tener aproximadamente 60 fps
}
return 0;}
```

Fig. 2.9 Ventana donde se muestra el video de una webcam usando OpenCV.

Para probar el funcionamiento de OpenCV en VS2010 ya se han usado algunas funciones y clases propias de OpenCV, todas ellas localizadas en el namespace cv; para una mejor comprensión de los argumentos que estas involucran, se hace un breve resumen de las mismas.

2.2.1.2 Principales estructuras de OpenCV.

OpenCV usa tipos de datos propios de las librerías, así que se exponen los más importantes para la aplicación y se toma [33] sus definiciones. Los expuestos son estructuras para trabajo en C y se puede aplicar en programación con vs2010.

2.2.1.2.1 cvMat

Pueden almacenar imágenes para procesar, es el análogo a IplImage que se usa en versiones anteriores de Open cv, para evitar inconvenientes de compatibilidad se usa una estructura Mat, la cual es mucho más extensa abarcando varios recursos de OpenCV

2.2.1.2.2 cvPoint

Tipo de dato para almacenar coordenadas en dos dimensiones. Para declarar un dato de este tipo en C++ su estructura es

Point punto (0,0);

2.2.1.2.3 cvRect

Almacena las coordenadas de un rectángulo, importante porque en método detectMultiScale devuelve sus datos en este tipo de estructuras.

2.2.1.3 Principales Funciones de OpenCV.

A continuación se analiza la utilidad de las funciones que se pudiesen emplear en el presente proyecto según la descripción propuesta en [33].

2.2.1.3.1 cvtColor

Convierte una imagen de un espacio de color a otro, como por ejemplo de RGB a HSV o escala de grises. Su estructura para C++ es la siguiente:

void cvtColor(InputArray src, OutputArray dst, int code, int dstCn=0)

-src: objeto donde esta almacenada la imagen a convertir.

-dst: objeto donde esta almacenada la imagen convertida

-code: tipo de conversión.

2.2.1.3.2equalizeHist

Ecualiza el histograma de una imagen en escala de grises, permitiendo normalizar el brillo y aumentar el contraste de la imagen. Su estructura para C++ es la siguiente:

void equalizeHist(InputArray src, OutputArray dst)

-src: objeto donde esta almacenada la imagen a ecualizar.

-dst: objeto donde esta almacenada la imagen ecualizada.

-code: tipo de conversión.

2.2.1.3.3namedWindow

Crea una ventana que puede ser usada como un muestrario de imágenes y barras. Su estructura para C++ es la siguiente:

void namedWindow(const string& winname, int flags=WINDOW_AUTOSIZE)
-winname: nombre de la ventana que se crea.

2.2.1.3.4imshow

Despliega una imagen en una ventana específica. Su estructura para C++ es la siguiente:

void imshow(const string& winname, InputArray mat)
-winname: nombre de la ventana que se despliega
-InputArray: Imagen a mostrar en la ventana.

2.2.1.3.5waitKey

Espesa que se presione una tecla o un retardo en milisegundos, solo funciona cuando una ventana HighGUI está activa. Su estructura para C++ es la siguiente:

int waitKey (int delay=0)
-delay: retardo en milisegundos, sin argumento significa por siempre.

2.2.1.4 Principales Clases de OpenCV

En la programación orientada a objetos, se ejecutan métodos asociados a un objeto que pertenece a una clase. A continuación se pone de manifiesto algunas clases con las que se pudiese ejecutar el proyecto y algunos métodos asociados a las mismas.

2.2.1.4.1CascadeClassifier

Es una clase usada para la detección de objetos a la cual se asocian varios métodos, para los más importantes a continuación se describe su aplicación y parámetros.

- CascadeClassifier

Su estructura para C++ es: CascadeClassifier::CascadeClassifier(const string& filename)

Al ejecutar este método a un objeto de la clase CascadeClassifier, se carga en el mismo un classifier cuyo nombre se especifica en su argumento “filename”

-load

Su estructura para C++ es: bool CascadeClassifier::load(const string& filename)

Parecido al método anterior, pero se puede cargar un obsoleto “HAAR” entrenado con la aplicación “haartraining” o un nuevo “cascade classifier” entrenado en la aplicación “traincascade”. OpenCV viene con varios classifier ya entrenados que se encuentran en la carpeta “data” sean estos; haarcascades, hogcascades o lbpcascades que se eligen según el objeto a detectar.

-detectMultiScale

Su estructura para C++ es: void CascadeClassifier::detectMultiScale(const Mat& image, vector<Rect>& objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0,Size minSize=Size(), Size maxSize=Size())

Detecta objetos de distintos tamaños en la imagen deseada y me devuelve una lista de rectángulos asociados a los objetos, esta función se puede paralelizar con TBB para detectar varios objetos en simultáneo. Sus argumentos se detallan a continuación.

cascade.- el cascade usado para la detección.

image.- Matriz del tipo CV_8U (escala de grises) contiene la imagen en donde se va a detectar el objeto.

objects.- Vector de datos tipo Rectangle donde cada uno está asociado al objeto detectado.

scaleFactor.- Especifica que tanto es reducida la imagen en la búsqueda del objeto.

minNeighbors.- Se especifica cuantos elementos adyacentes debe tener un rectángulo para retener el objeto.

flags – no se usa con CascadeClassifier soso con los Haar.

minSize – Mínimo tamaño del objeto.
minSize –Máximo tamaño del objeto.

2.2.1.4.2VideoCapture

Es una clase para la captura de video, sea desde archivos o desde cámaras. Esta clase provee una API (Application Programming Interface). Los métodos más importantes son:

- VideoCapture

Su estructura para C++ es: VideoCapture::VideoCapture(int device) o C++: VideoCapture::VideoCapture(int device)
Se usa como constructor para acceder a un archivo o dispositivo.

-release

Su estructura para C++ es: void VideoCapture::release()
Cierra el archivo o dispositivo de captura.

2.2.1.5 Uso de OpenCV en la Aplicación.

OpenCV es una herramienta para Vision Computacional, por ello trabaja con imágenes y video facilitando la consecución de un sinfín de aplicaciones, entre ellas la detección de objetos. Para la aplicación que se desarrolla en el proyecto OpenCV está la respuesta en la tarea de detección de un rostro, pues el método “detectMultiScale” asociado a la clase “CascadeClassifier” permite la detección de un rostro en una imagen, basándose un algoritmo HaarCascade (detallado en el capítulo 1 sección 1.2.3.3) más sofisticado y actualizado.

Por otra parte estas librerías incluyen haarcascades y lbpcascades ya entrenados e implementados como archivo, que puede ser usado para detección de rostro sea de lado o de frente, ojos, nariz boca, entre otros y se los utiliza según el tipo de aplicación, para la presente se puede usar un “frontalface” sea Haar o lbp (Local binary patterns), con el que se realiza la detección del rostro basándose en patrones característicos del mismo, como ubicación de ojos nariz y boca ya que para la aplicación no es necesario la detección individual de cada parte del rostro.

2.2.2 EMGU CV

Fig. 2.10 Logotipo de EmguCV

Es una herramienta tipo envoltura que permite el llamado de funciones de OpenCV desde soluciones .NET, con lo que se puede hacer uso de distintos lenguajes como C#, VB, VC++, IronPython etc. Además esta herramienta puede ser usada para diferentes sistemas operativos y dispositivos sean estos según Windows, Linux, Mac OS X, iPhone, iPad and Android la necesidad.

En otras palabras EmguCV es OpenCV para entorno .NET en donde se puede trabajar con entorno gráfico ideal para trabajar con aplicaciones “Windows Forms” de Microsoft Visual Studio. Las librerías están escritas en lenguaje C# y tienen la misma funcionalidad de las versiones de OpenCV.

2.2.2.1 Instalación de EmguCV con Microsoft Visual Studio.

La instalación de esta herramienta es muy sencilla basta con extraer los archivos de las librerías en un directorio conocido. Estas librerías se las obtiene de forma gratuita en [34] en distintas versiones para desarrollo de aplicaciones en código abierto.

Al igual que OpenCV se requiere establecer la variable de entorno para facilitar la configuración con VS2010, se debe asignar la dirección de las librerías como indica la figura 2.11.

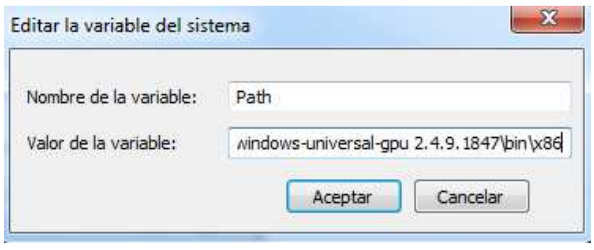


Fig. 2.11 Logotipo de EmguCV

Que la variable de entorno sea establecida, será necesario para que la aplicación a desarrollar pueda ser ejecutada, por lo que este acción será realizada en cualquier ordenador en el que se desee probar la aplicación posteriormente.

Para poder desarrollar la aplicación con VS2010 se debe agregar las referencias de EmguCV en el proyecto. En la pestaña “Proyecto” se escoge la opción “agregar referencia”. En la ventaja que se abre se escoge Browser y se agrega los archivos con extensión .dll contenidos dentro de la carpeta “bin” de los archivos de EmguCV.

Para utilizar los recursos de OpenCV se debe agregar al proyecto elementos existentes, de igual forma en la pestana “Proyecto” se selecciona la opción “Agregar elemento existente” y se selecciona los archivos cuyo nombre comienza con OpenCV que están localizados en el directorio de EmguCV bin/x86. Se verifica en el explorador de soluciones de VS2010 que los archivos seleccionados anteriormente se hayan añadido correctamente.

EmguCV cuenta con componentes de .NET framework para trabajar en una aplicación Windows Forms de Visual Studio, que se debe añadir al cuadro de herramientas de VS 2010. Para ello se da clic derecho sobre el Cuadro de Herramientas y se escoge la opción “elegir elementos”. En la ventana que aparece se acciona la pestaña de “examinar” y dentro de la carpeta “bin” contenida en los archivos de EmguCV escogemos el ítem “Emgu.CV.UI.dll”; con ello se habilitan las herramientas gráficas de EmguCV especialmente Image BOX que se usa para mostrar imágenes o video que será necesario para desarrolla la interfaz de proyecto en ejecución.

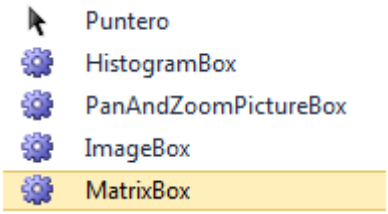


Fig. 2.12 Herramientas gráficas de EmguCV

2.2.2.2 Codificación en EmguCV

Las funciones y clases son análogas a las usadas en OpenCV, simplemente usando los argumentos con las clases de EmguCV acorde al lenguaje utilizado. La principal clase a ser Usada es CvInvoque que provee una vía directa para llamar a funciones dentro de OpenCV hacia lenguajes .NET, donde cada método de la clase CvInvoque corresponde a una función de OpenCV con el mismo nombre.

Como referencia para usar el EmguCV se tienen según [34] la tabla 2.1 las forma de usar estructuras de OpenCV

Estructura .Net	Estructura OpenCV
System.Drawing.Point	CvPoint
System.Drawing.PointF	CvPoint2D32f
System.Drawing.Size	CvSize
System.Drawing.Rectangle	CvRect

Tabla 2.1 Estructuras Equivalentes en .NET a OpenCV

Para trabajar con EmguCV se debe definir primero el lenguaje a usar, sin embargo su arquitectura en general es muy parecida al OpenCV por ello se debe simplemente hacer una migración de código encontrando la analogía con C++ hacia el lenguaje en el que se trabaje con EmguCV. Esta herramienta es muy amigable y fácil de usar una vez conocida la forma de trabajo con OpenCV.

2.2.2.3 Uso de EmguCV en la Aplicación.

Esta herramienta como tal, presenta una gran facilidad en cuanto uso, por ello desarrollar una aplicación con EmguCV no resulta del todo complicado. Para el presente proyecto se puede usar EmguCV, como medio para tener los algoritmos de detección de objetos con los que cuenta OpenCV, en un entorno visual específicamente aplicaciones “Windows Forms” que dentro de Microsoft Visual Studio permiten un desarrollo para interfaz muy amigable además de comunicación serial indispensable para la interfaz con el robot del sistema.

2.2.3 QT

Fig. 2.13 Logotipo de Qt

Qt 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 es desarrollada como un software libre y de código abierto a través de Qt Project. Qt utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios otros lenguajes de programación a través de bindings. También es usada en sistemas informáticos empotrados para automoción, aeronavegación y aparatos domésticos como frigoríficos [35].

2.2.3.1 Uso de Qt en la Aplicación.

La instalación de este paquete no es muy complicada, sin embargo para usarlo junto con OpenCV se requiere tener las librerías personalizadas para uso con Qt como se lo detalla en 2.2.1.1.2.

La ventaja que tiene trabajar con Qt es el entorno gráfico y que hay gran cantidad de aplicaciones al ser un software libre. De optar por desarrollar la aplicación con la ayuda de Qt se puede realizar toda la aplicación trabajando con la API propia de Qt y obviando el uso de VS2010 o tratar de emplear las librerías desde VS2010; si selecciona la primera opción se requiere implementar una comunicación serial para Qt, que no es tan sencilla en principio; por otra parte Qt trabaja con programación muy orientada a objetos incluso para el entorno gráfico por ello este entorno no resulta muy amigable a primera vista.

2.3 SELECCIÓN DEL LENGUAJE

En el IDE Microsoft Visual Studio se puede elegir el tipo de lenguaje con el que se desarrollará la aplicación sean estos C, C++, C# o Basic; así como otro tipo de ambientes de desarrollo que se usan para aplicaciones web que quedan lejos del alcance de este proyecto.

Trabajar con lenguaje C++ en un principio sería ideal para el uso de las librerías de OpenCV, con lo que la detección del rostro se facilita, no obstante es necesario implementar un entorno gráfico; para ello se requieren las librerías de Qt ya que en una aplicación Windows Forms en lenguaje C++ no es posible usar directamente OpenCV lo que conduce a la exigencia de crear librerías propias personalizadas, y ello amerita un coste mayor en cuanto a tiempo; por otro lado el entorno de Qt en un principio no es muy amigable lo que pudiese dificultar el desarrollo de la aplicación.

Por otra parte se cuenta con EmguCV que permite usar las funciones y clases de OpenCV para trabajar en un entorno gráfico .NET, donde se tiene la opción de usar lenguaje C# o Basic que resulta mucho más tentador que la opción anterior; así que el proyecto se basará en usar OpenCV de forma indirecta amparándose en la herramienta EmguCV.

Para optar por una de las distintas opciones en cuanto a lenguaje se elaboró un algoritmo básico para detectar los movimientos de la cabeza en los 3 lenguajes C++, C# y basic, teniendo resultados muy similares por lo que se concluye que con un código optimizado, el lenguaje no se constituye en una limitación para cumplir con los requerimientos del proyecto en ejecución.

Por ello se opta por desarrollar la aplicación completa con el lenguaje más sencillo y amigable "Basic", así la aplicación completa se la desarrolla con un template "Windows Forms Application" para Visual Basic, con ello es posible tener una Interfaz Gráfica de Usuario y Comunicación Serial cumpliendo con los objetivos del proyecto.

2.4 DESARROLLO DEL ALGORITMO PARA DETECCIÓN ROSTRO E INTERPRETACIÓN DE MOVIMIENTOS EN LA APLICACIÓN

2.4.1 DETECCIÓN DE ROSTRO

Como ya se ha mencionado trabajar con EmguCV en lenguaje VisualBasic es relativamente sencillo ya que sus clases, funciones y argumentos son análogos a los usados en OpenCV para C++. Para saber cómo usar correctamente EmguCV con tratamiento de imágenes nos remitimos a lo publicado en [34] y [36] donde se detallan las funciones para trabajar con video en EmguCV.

Como primer paso se requiere adquirir correctamente el video del entorno en el que se encuentra el usuario para lo cual se utiliza un objeto de la clase "Image" de EmguCV para almacenar cada imagen del video captado por la cámara. Para la captura de la trama de video se ocupa el método QueryFrame sobre una clase de tipo "Capture" (análoga a VideoCapture de OpenCV).

El clase principal del programa comienza con la adquisición de la imagen en la que se requiere buscar un rostro humano, para ello se usa un objeto de la clase "CascadeClassifier" donde se cargara un CascadeClassifier ya entrenado, disponible dentro la carpeta “data” de OpenCV, este CascadeClassifier puede ser "lbpcascade_frontalface.xml" (classifier que usa el modelo local binary patterns), sobre esta objeto se aplicará un método DetectMultiScale, trascendental en la detección facial. El método DetectMultiScale devuelve una estructura de tipo Rectangle en la que se almacena él o los rostros detectados en la imagen; para el caso particular, el rostro del usuario.

Como la detección está sujeta al uso del método DetectMultiScale es menester tener todos los argumentos en la forma adecuada para hacer uso de la misma, por ello previamente se deberá procesar la imagen original, convertirla a escala de grises y filtrarla; la imagen procesada se almacenará en otra clase tipo "Image" sobre la cual se efectúa la detección facial. Se usa el método “convert” para pasar la imagen a escala de grises y el método SmoothGaussian para suavizar la imagen.

Se comienza con la detección de rostro trabajando con el objeto cargado anteriormente con el CascadeClassifier "lbpcascade_frontalface.xml", sobre este objeto se aplica el método DetectMultiScale con los siguientes argumentos:
-image: La imagen pasada a escala de grises y filtrada anteriormente.
-scaleFactor: Se elige el valor 1.2 recomendado para trabajar en aplicaciones de video en tiempo real.
-minNeighbors: Se elige el valor 3 tras diferentes pruebas buscando mayor precisión y velocidad.
-minSize: Un rostro detectado en un área mayor a 100*100 pixels.
-minSize: Un rostro detectado en un área menor a 250*250 pixels.

Cada imagen captada por la cámara tiene una Resolución de 640*480 pixels, esta comprende el área total de trabajo para la detección y control, por lo que los valores antes mencionados son las limitantes del algoritmo a desarrollar.

Si dentro de la imagen procesada se ha detectado uno o varios rostros, esta información se almacena en una estructura Rectangle. En particular, la información requerida para la aplicación en cuestión es si se detecta o no un rostro, de ser así, en que ubicación dentro de la imagen captada.

Basándose en CascadeClassifier, en particular ("lbpcascade_frontalface.xml") se logra detectar el rostro del usuario en cada imagen del frame del video; toda la información en torno se almacena en un objeto al que se define como rostro, mismo que servirá para el desarrollo del algoritmo de interpretación de movimientos que se detalla en la siguiente sección.

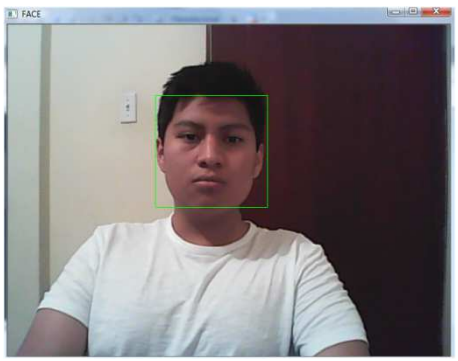


Fig. 2.14 Detección del rostro

2.4.2 INTERPRETACIÓN DE MOVIMIENTOS DE CABEZA

Para tomar las acciones de control es necesario saber los movimientos que realiza el usuario con su cabeza. Para poder interpretar dichos movimientos se desarrolla un algoritmo que sea lo suficientemente robusto, para que el sistema se comporte correctamente cuando operen distintos usuarios. Un hecho que ayuda de sobremanera a este objetivo es que OpenCV viene con classifiers muy bien entrenados con diferentes usuarios, por ello la problemática con la detección de rostro se minimiza, haciendo posible enfocarnos netamente en la interpretación de los datos del rostro ubicado en la pantalla.

Primeramente se debe definir en qué parte se encuentra el rostro inicialmente; con la información obtenida del objeto “rostro” explicado anteriormente se puede conocer la ubicación del rostro en cada imagen. Sin embargo, no se debe olvidar que la aplicación trabaja con video en tiempo real por ello se implementa el siguiente algoritmo.

Se determina el centroide del área donde se ha detectado el rostro en cada imagen, se almacena este dato.

Se requiere encontrar un punto de referencia promedio para la primera trama de imágenes, para ello se toman las primeras 30 muestras y se opera sobre las mismas para obtener el punto de referencia. Como se opera aproximadamente a 30fps este proceso dura aproximadamente un segundo más el tiempo que se tome en procesar los datos que es poco.

Una vez determinado el punto de referencia, se puede interpretar los movimientos de la cabeza con respecto a este punto, para ello se requiere conocer el centroide que define la posición actual del rostro en cada imagen del video captado.

Para cada frame se realiza la comparación entre la referencia y posición actual con el fin de determinar el error que servirá para saber si existió movimiento relativo o no y en qué dirección.

El dato de error obtenido en la comparación anterior es el parámetro para ejecutar las órdenes, junto de las dimensiones actuales (ancho y alto) del objeto rostro. El dato de error tiene parte tanto para coordenada “x” como para “y”, cada uno se compara con alto o ancho del rostro detectado respectivamente. En base a estas comparaciones se puede determinar la posición relativa actual del rostro del usuario, sean estas: izquierda, derecha, arriba, abajo o alguna combinación de las mismas. Sabiendo ya los movimientos del usuario se pueden tomar las acciones de control respectivas. Todo el algoritmo de comparación y acciones pertinentes se sintetiza en el diagrama de flujo expuesto en la sección 2.7 del presente documento.

2.5 COMUNICACIONES

Para efectuar acciones de control y tener un ambiente amigable en la interfaz de usuario, se requiere implementar una correcta comunicación desde el robot a implementar hacia la interfaz y viceversa. Para cumplir con este cometido, se requiere definir en la aplicación el protocolo a seguir para transmisión y recepción de datos.

2.5.1 COMUNICACIÓN CON LA CÁMARA DEL ROBOT.

El robot móvil a implementar será dotado de una cámara que transmita video hacia la interfaz. Debido a las bondades que brinda Microsoft Visual Studio, se puede usar distintos protocolos para comunicación entre los cuales TCP/IP susceptible a comunicación en una red inalámbrica. El protocolo antes mencionado podría ser usado para la comunicación con la cámara, hecho que será vital al momento de elegir el hardware en el siguiente capítulo, donde se detalla la configuración fuera de la aplicación.

Se puede acceder a un browser asociado a la dirección IP de la cámara por ello se requiere asignar una IP estática al momento de configurar equipos. Por facilidad se usa la aplicación incluida con la cámara para visualizar el video del robot, con ello la recepción de video se vuelve simple e independiente de los otros procesos optimizando los recursos de la unidad de procesamiento gráfico en la que se cargue la aplicación.

2.5.2 TRANSMISIÓN Y RECEPCIÓN DE DATOS ENTRE PC Y ROBOT

Puesto que el robot a implementar seguramente será gobernado por un micro-controlador, se implementará una comunicación serial bidireccional para el intercambio de datos, interfaz-robot.

Microsoft Visual Studio en aplicaciones “Windows Forms” para lenguaje basic, entorno de desarrollo de la aplicación, cuenta con una herramienta que permite acceder a los puertos seriales del computador para ello se debe usar la referencia IO.Ports del sistema.

La transmisión de los datos hacia el robot se limita al envío de comandos tipo string que serán interpretados en el sistema micro-procesado, los comandos a enviar se vinculan estrictamente a la detección de movimientos, con excepción del comando de habilitación y des habilitación del sistema que se controlará desde la interfaz. Mediante la lógica usada para la interpretación de movimientos se determina que comando enviar, mismos que cumplan la función de decir al sistema que se dirija izquierda, derecha, centro, acelere o frene. Estos 5 comandos serán la base para el control de la movilidad del robot.

Se requiere implementar una comunicación inalámbrica, lo que dificulta tener una comunicación sincrónica, por ello la recepción de datos se la hace en una subrutina que permite almacenar los datos recibidos hasta que sean usados sin afectar los procesos que se ejecutan en el momento de la recepción.

Como se requiere la visualización de variables es menester que se reciban datos analógicos, los mismos que deben ser interpretados en la aplicación. Para facilitar el trabajo se interpretan comandos tipo enteros que se encuentren fuera del rango de las variables que se trabaja. Cada variable recibida se lo vincula con un comando, de esta manera se pude saber que dato ha sido recibido y que acciones tomar. Por último se reciben comandos que anuncian una alerta de obstáculo. Todas las variables que se involucran se las detalla en el siguiente apartado donde se describe la interfaz de usuario.

2.6 INTERFAZ GRÁFICA DE USUARIO

Uno de los objetivos del presente proyecto es desarrollar una interfaz gráfica lo más amigable posible donde se simule un ambiente de manejo, para ello se deben conocer las herramientas gráficas de Visual Studio, que pudieran servir para el desarrollo en cuestión.

2.6.1 HERRAMIENTAS PARA INTERFAZ GRÁFICA EN VISUAL STUDIO

A continuación se detalla el funcionamiento de las herramientas que sirven para la visualización de imagen y video sea de cámara web o de una red TCP/IP, visualización de variables, etiquetas, indicadores y controles; elementos que forman parte de una interfaz gráfica amigable.

-ImageBox: Muestra una imagen, usada cuando se desarrolla con EmguCV.
-Button: Desencadena un evento cuando el usuario hace click sobre él.
-ComboBox: Muestra un cuadro de texto editable con una lista despegable de los valores permitidos.
-Label: Proporciona información en tiempo de ejecución o texto descriptivo para un control.
-ProgressBar: Muestra una barra que se va completando para indicar al usuario el progreso de una operación.
-RichTextBox: Proporciona una entrada de texto y características de edición avanzadas, como el formato del párrafo y caracteres.
-TrackBar: Permite al usuario elegir entre un rango de valores que puede elegir desplegándose por una pequeña barra situada junto a otra barra.
-LineShape: Control gráfico que muestra una línea horizontal, vertical o diagonal.
-OvalShape: Control gráfico que muestra una elipse.
-RectangleShape: Control gráfico que muestra un rectángulo.

Todas las herramientas anteriormente descritas según la misma API de Visual Studio 2010 serán de utilidad para el diseño de la interfaz de usuario, con el fin que la misma tenga un ambiente de conducción amigable.

2.6.2 AMBIENTE DE CONDUCCIÓN

Garantizar el confort del usuario es uno de los objetivos planteados, por ello se busca tener un ambiente virtual que emule una cabina de manejo. Para cumplir este requerimiento se debe incorporar el video del ambiente alrededor de un robot móvil, el mismo servirá como realimentación visual para el sistema.

Dentro de una cabina de manejo aparte de tener la visión del entorno se cuenta con indicadores de estado de energía y velocidad, los mismos serán implementados en la pantalla principal lo más accesible a la vista del usuario pero sin distraerlo del video del robot.

Se elige poner en el ambiente el video del robot, el video del usuario como fondo, texto de ayuda para el usuario, indicador de velocidad, estados de baterías tanto de la cámara como del robot, temperatura, e indicadores de ayuda. La forma de implementar los anteriores en la interfaz se detalla en el siguiente apartado.

2.6.3 VISUALIZACIÓN DE VARIABLES, INDICADORES Y VIDEO.

Aparte del ambiente de conducción en la interfaz se debe incluir los controles para la puesta en marcha de la aplicación, con todo lo detallado se describe a continuación cada elemento incorporado en la interfaz gráfica según las variables usadas para la aplicación.

-ibCamara: Se usa una ImageBox en donde se visualiza al usuario y se verifica que se detecta y sigue el rostro de mismo.

-btnAbrirCerrar: Un Button que actúa en forma intermitente, con el fin de abrir o cerrar el puerto de comunicación serial seleccionado.

-BtnInicioReset: Un Button que actúa en forma intermitente, con el fin de iniciar la aplicación de control o resetear el sistema para cambiar de usuario o reubicarse.

-rsEstado: Indicador del estado del sistema sea; habilitado, deshabilitado o en adquisición de referencia.

-txt: Se usa un RichTextBox en donde se mostrará ayudas de texto para el usuario.

-rtbRobot: RichTextBox usado como indicador digital del porcentaje restante de la batería del robot.

-pbRobot: ProgressBar usada como indicador analógico del porcentaje restante de la batería del robot.

-rtbCamara: RichTextBox usado como indicador digital del porcentaje restante de la batería de la cámara.

-pbCamara: ProgressBar usada como indicador analógico del porcentaje restante de la batería de la cámara.

-rtbTemperatura: RichTextBox usado como indicador digital de temperatura.

-tbtemperatura: TrackBar usado como indicador analógico de temperatura.

-rtbVelocidad: RichTextBox usado como indicador digital de la velocidad del robot.

-lsVelocidad: LineShape usado como indicador analógico de velocidad.

-Os_derecha: OvalShape usado como indicador de giro a la derecha.

-Os_izquierda: OvalShape usado como indicador de giro a la izquierda.

A parte de las herramientas mencionadas anteriormente que se añaden a la interfaz de usuario, se hace uso de etiquetas “label” que ayudan a la familiarizar al usuario con la interfaz.

Como resultado la ubicación de los elementos agregados a la interfaz y su respectiva disposición se muestra en la Fig. 2.15 donde se detalla la descripción final de la interfaz de la aplicación.

La puesta en marcha de la aplicación se muestra en el capítulo 4 donde se detalla los resultados del funcionamiento de la interfaz y su uso se lo describe a detalle en el anexo1 “Manual de Usuario de la aplicación”.

Fig. 2.15 Disposición de elementos en la Interfaz.

2.7 DIAGRAMAS DE FLUJO DE LA APLICACIÓN

Microsoft Visual Studio para aplicaciones con lenguaje “Basic” trabaja a manera de subrutinas, estas últimas se encuentran asociadas a eventos, el evento principal es el que permite la presentación del GUI que se inicializa al cargar el Form principal. Al suceder un evento se corre su respectiva subrutina, todas estas se encuentran dentro de la clase principal.

Se detalla en los siguientes diagramas de flujo la lógica de las subrutinas asociadas a los eventos que suceden en la ejecución de la aplicación, en las mismas se pone de manifiesto la lógica que se usa para la interpretación de movimientos para el control del sistema, así como la interpretación de los datos recibidos desde el robot.

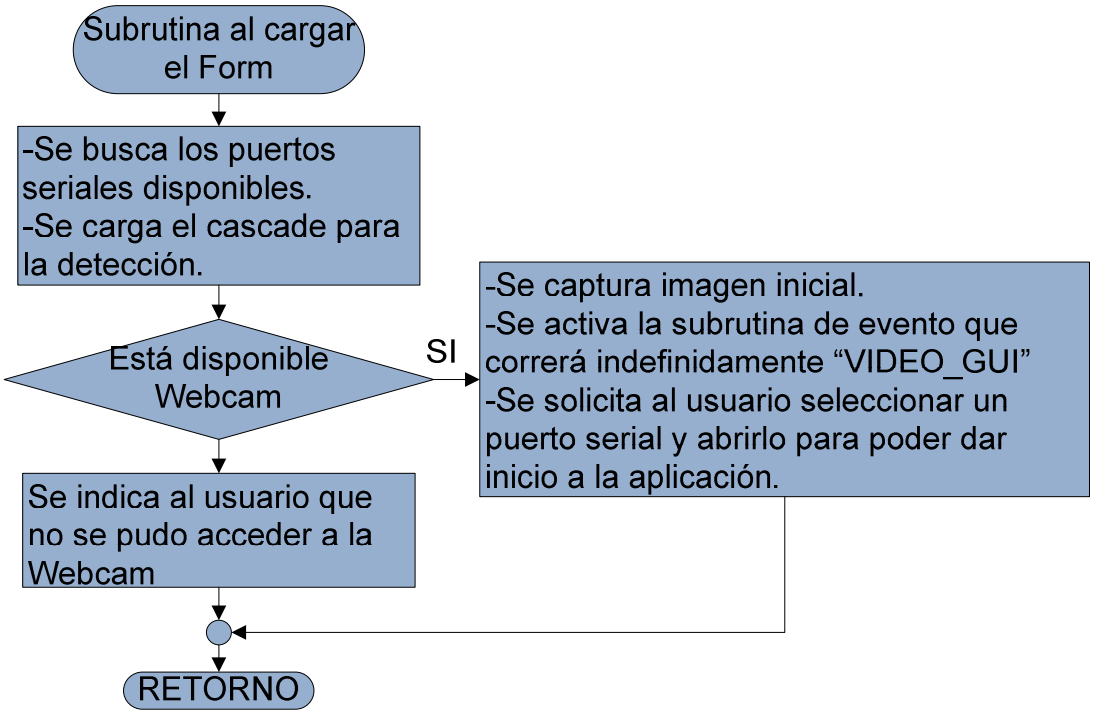


Fig. 2.16 Diagrama de flujo de la subrutina que se ejecuta al abrir la aplicación.

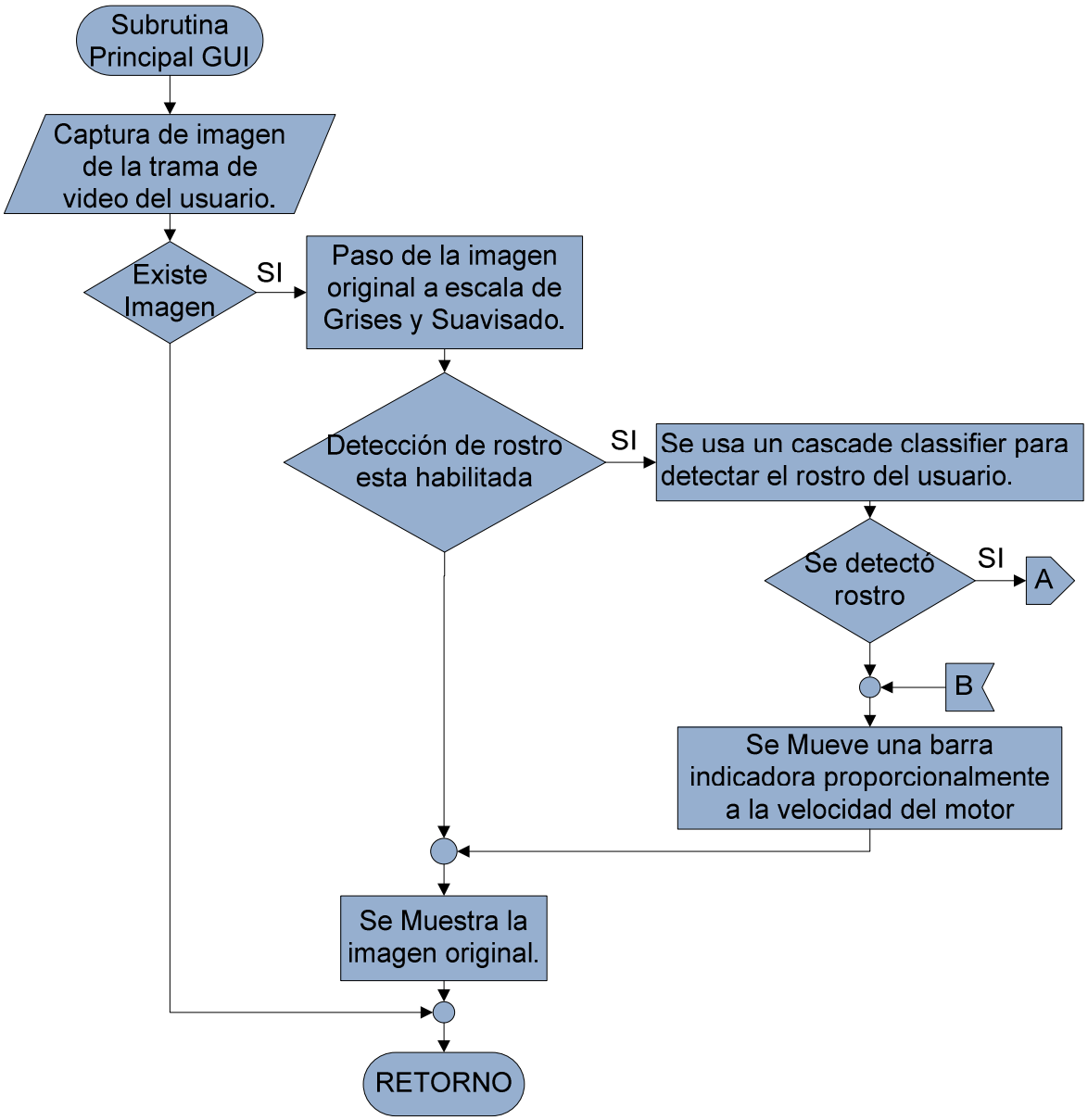


Fig. 2.17 Diagrama de flujo de la subrutina principal del GUI (a).

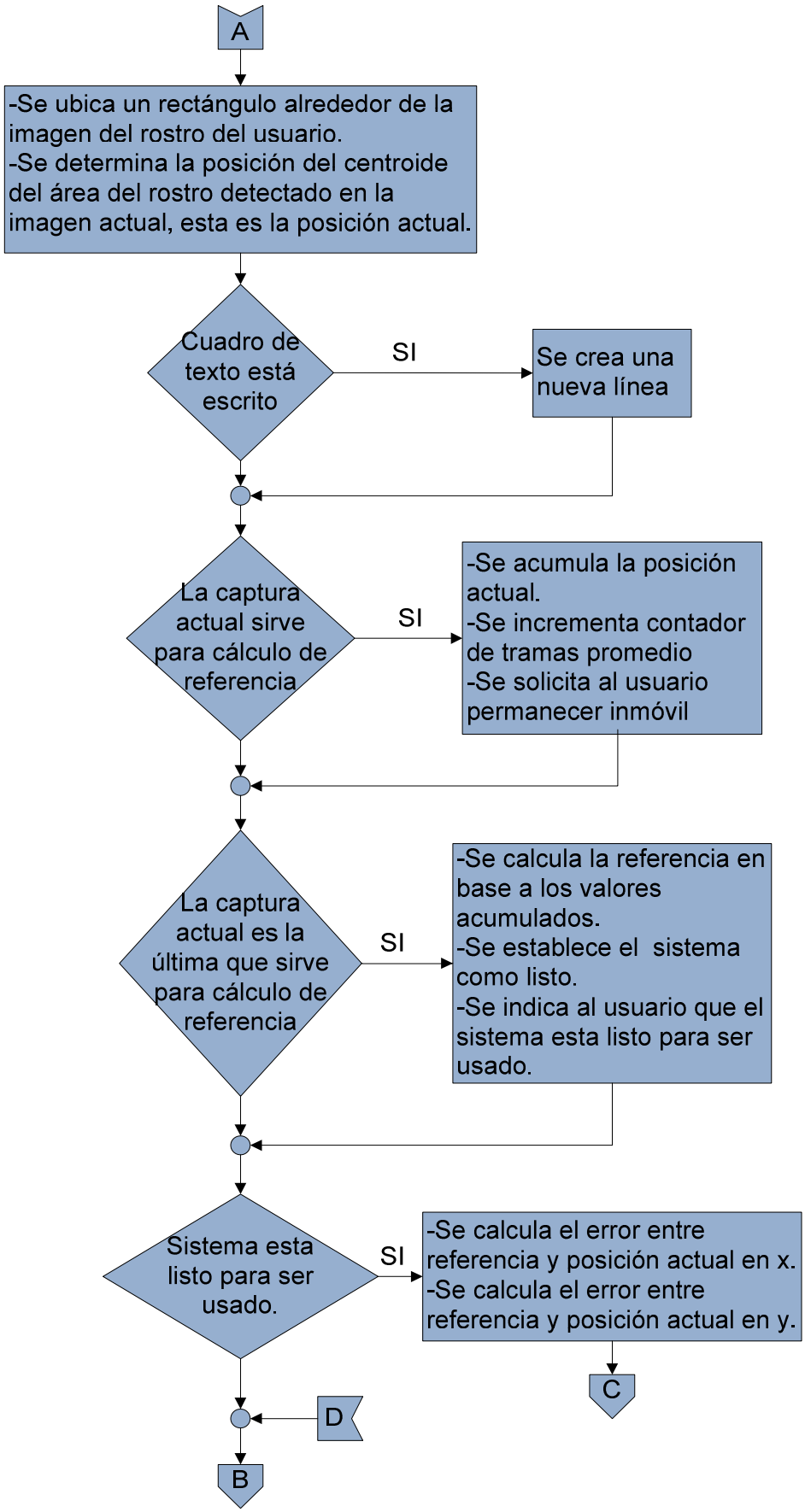


Fig. 2.18 Diagrama de flujo de la subrutina principal del GUI (b).

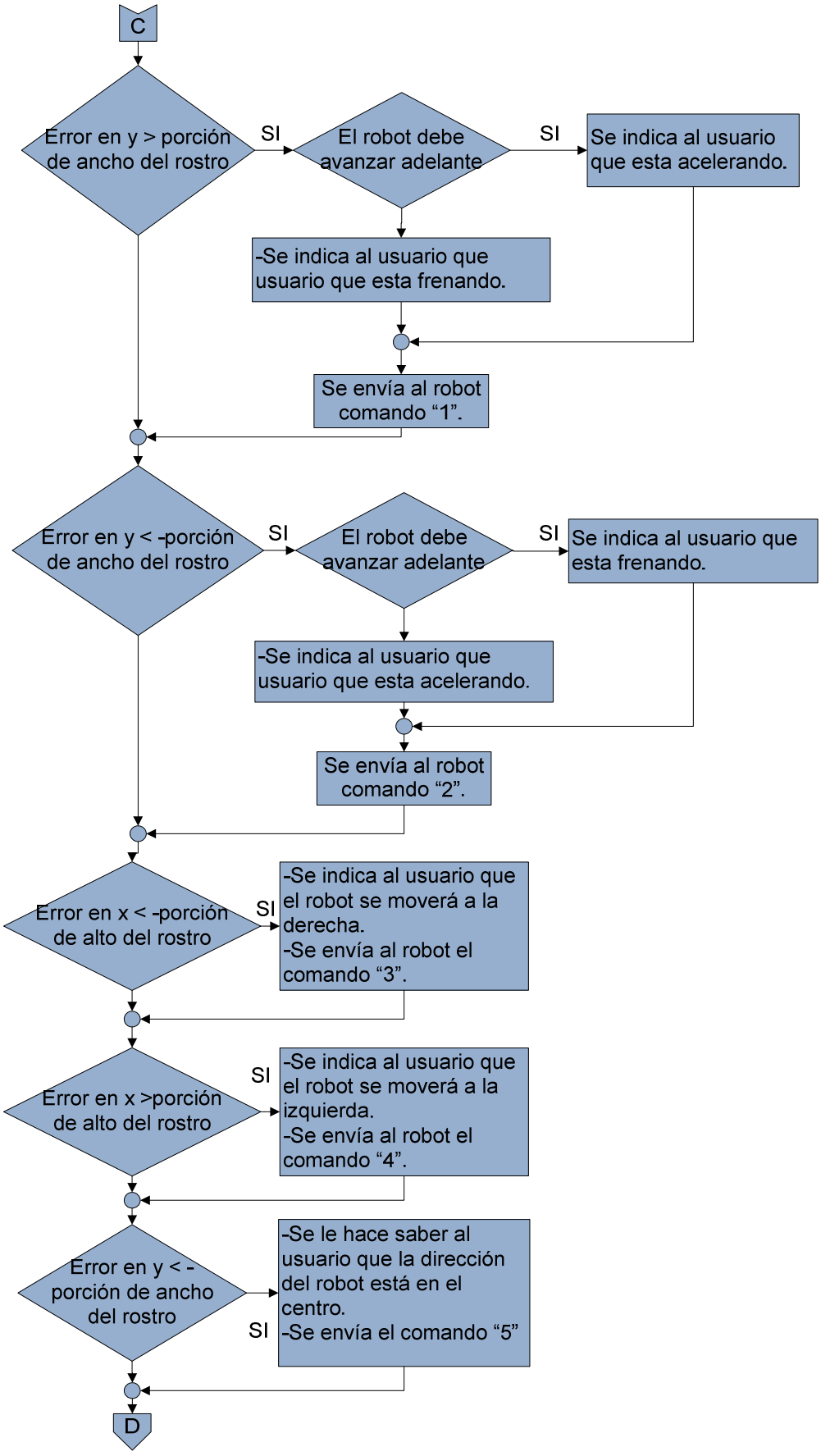


Fig. 2.19 Diagrama de flujo de la subrutina principal del GUI (c).

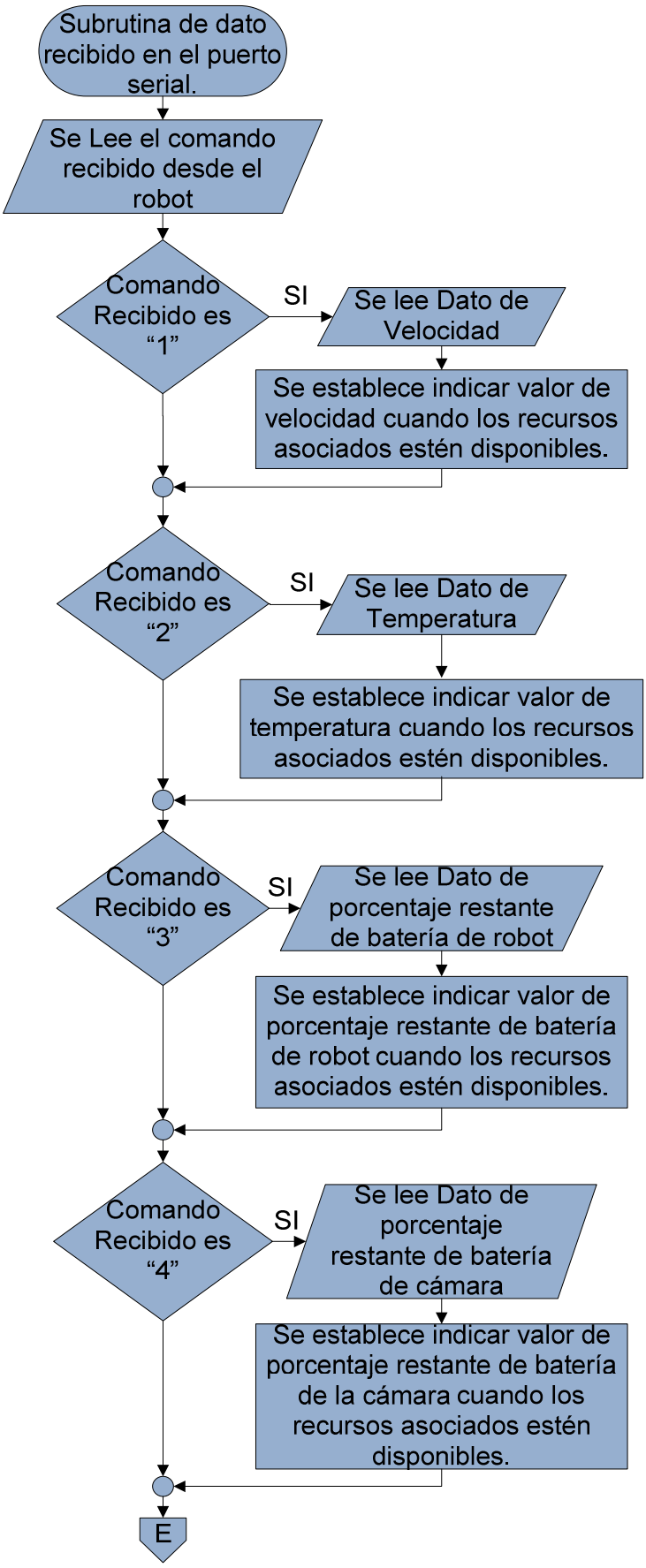


Fig. 2.20 Diagrama de flujo de la subrutina que se ejecuta al recibir un dato (a).

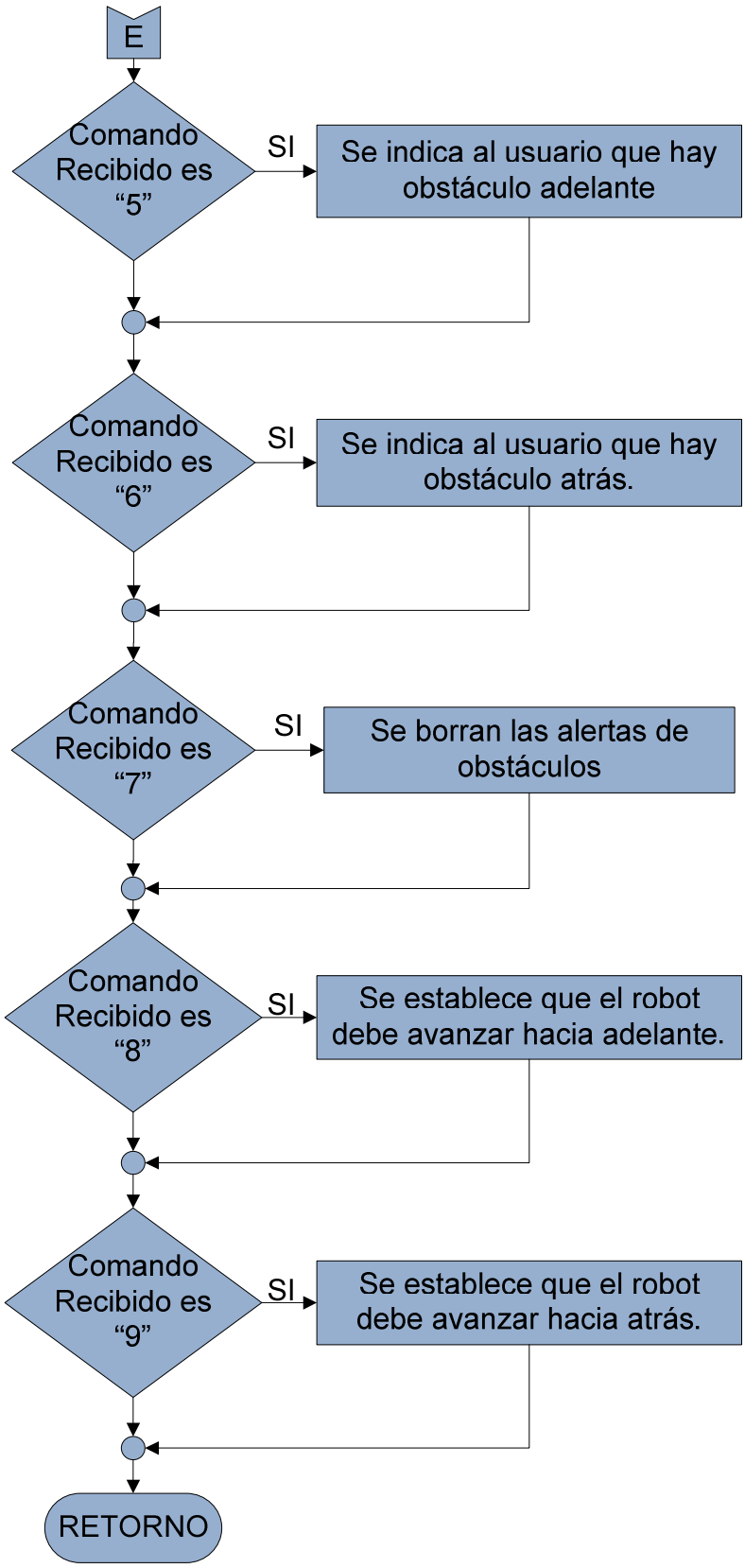


Fig. 2.21 Diagrama de flujo de la subrutina que se ejecuta al recibir un dato (b).

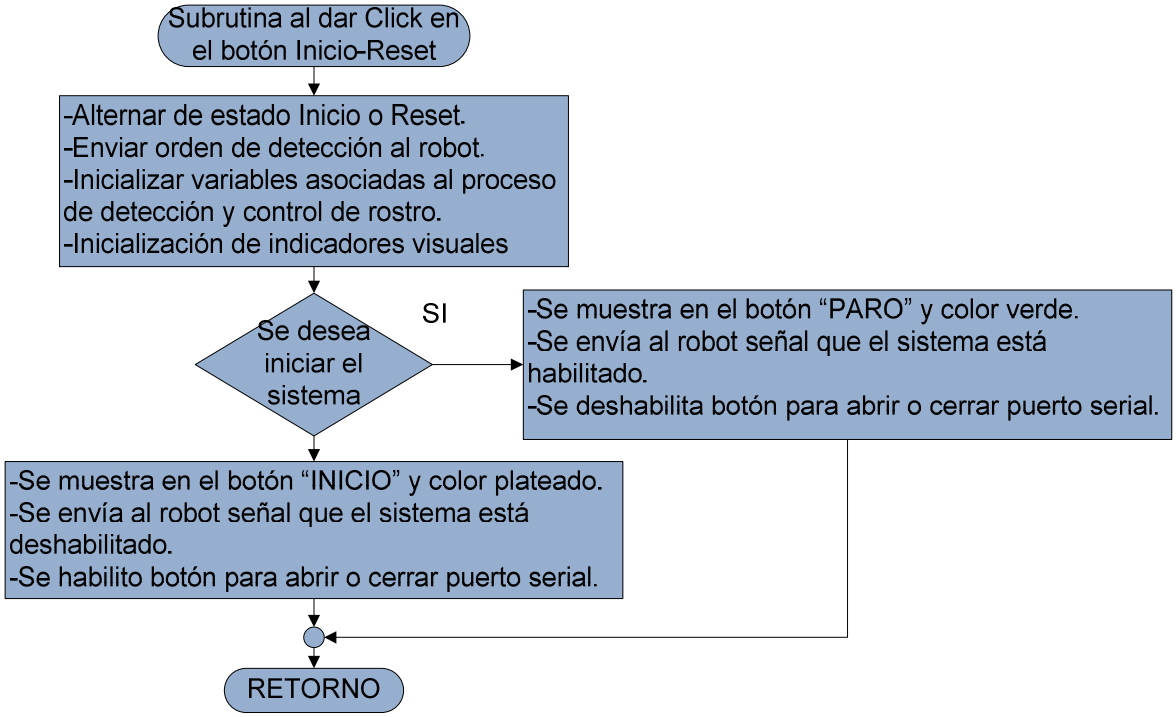


Fig. 2.22 Diagrama de flujo de la subrutina Inicio-Reset.

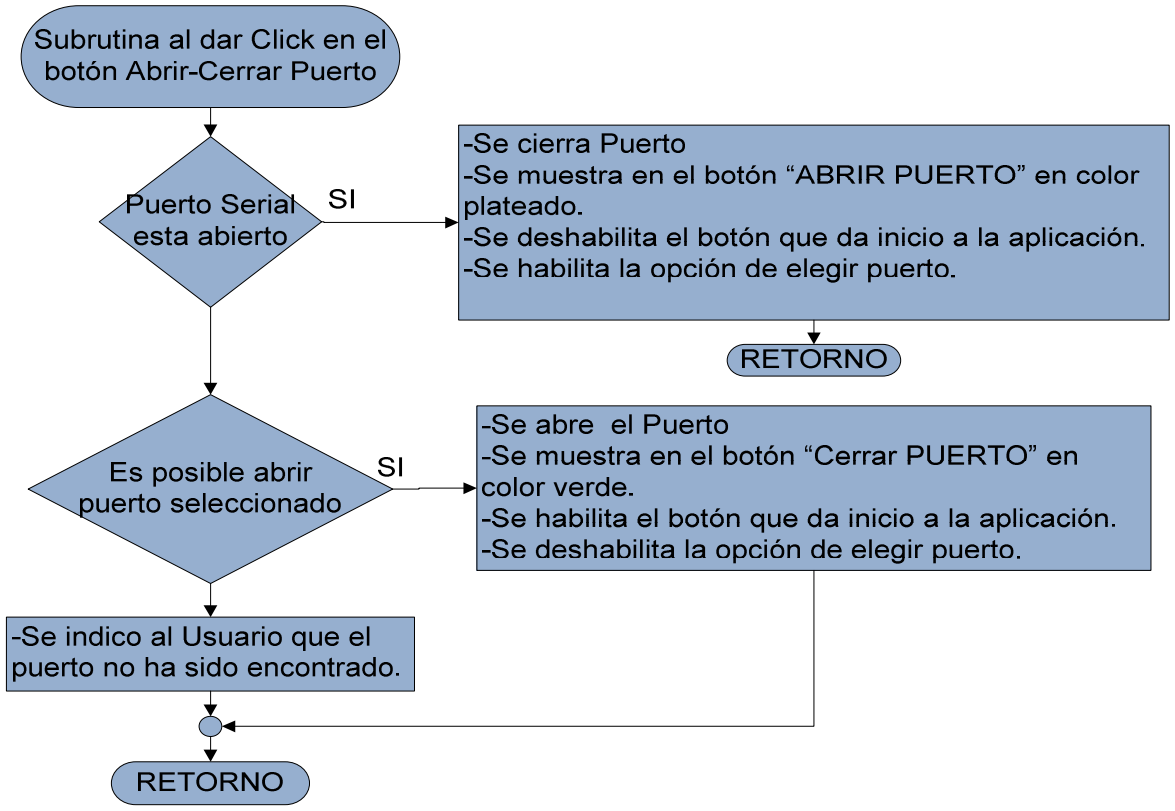


Fig. 2.23 Diagrama de flujo de la subrutina Abrir-Cerrar puerto.

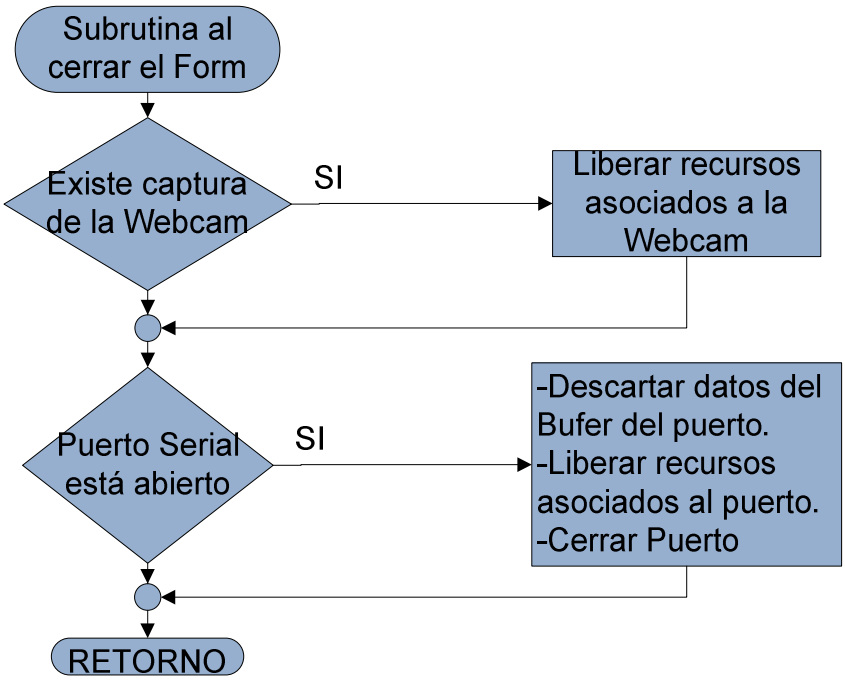


Fig. 2.24 Diagrama de flujo de la subrutina que se ejecuta al cerrar la aplicación.

CAPÍTULO 3

DISEÑO Y CONFIGURACIÓN DE LA ELECTRÓNICA DEL SISTEMA

En los alcances del presente proyecto se establece implementar el control de un robot móvil, dejando a consideración el diseño o selección del mismo según se estime lo más viable para el proyecto. Se ha considerado conveniente diseñar íntegramente un mini robot móvil que pueda cumplir la función de emular un vehículo a controlar. El robot que se diseña debe ser estructurado de tal forma que sus características se asimilen a un vehículo común por ello se toma como modelo el tipo car-like. Por tratarse de un mini-robot y dado que el fin del presente capítulo es describir la electrónica del sistema no se profundiza en un diseño mecánico. Sin embargo, es necesario conocer superficialmente la estructura mecánica del robot, como punto de partida para delinear las directrices del control del mismo, dicha estructura se describe en el siguiente apartado.

3.1 ROBOT MÓVIL

En el capítulo 1 de este documento se dio una breve descripción del robot tipo car-like, que haciéndolo simple se trata de un móvil de 4 ruedas (dos adelante y dos atrás) con tracción en las ruedas traseras y dirección en las delanteras, sistemas que actúan en forma independiente.

El diseño del robot del presente proyecto, toma como base una estructura que cumpla lo establecido anteriormente. En donde el eje de las ruedas traseras esté acoplado a una caja reductora a la que se deberá articular un motor, pudiendo así tener un mayor torque en la carga. Además las ruedas delanteras se encuentran provistas de un mecanismo que permite dar dirección al móvil en funcionamiento, para controlar dicho mecanismo se le deberá acoplar otro motor. La estructura usada en el desarrollo del mini robot requerido ha sido tomada de un juguete en desuso adaptado a las necesidades del sistema a implementar. Hecho que ha permitido reducir los costos del prototipo.

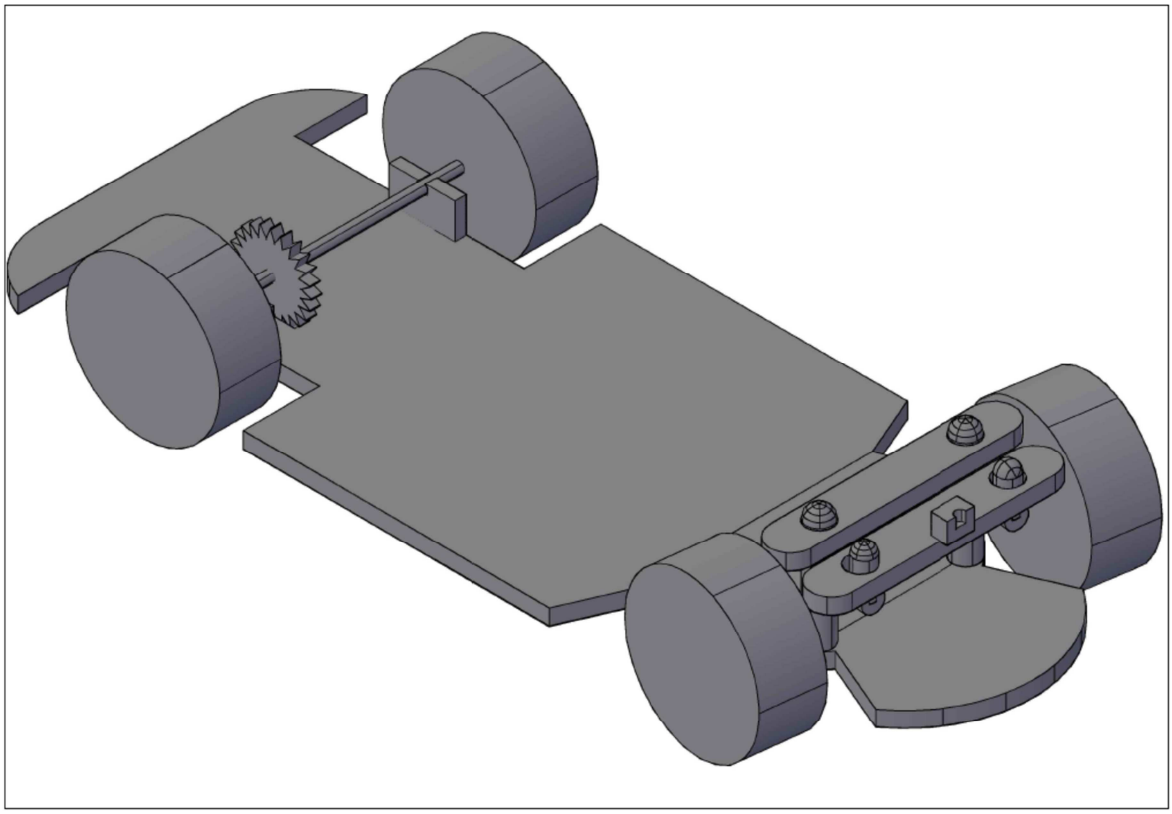


Fig. 3.1 Estructura base del robot móvil.

3.2 SELECCIÓN DE EQUIPOS Y ARQUITECTURA

A continuación se describen los equipos seleccionados para el desarrollo del sistema a implementar.

3.2.1 UNIDAD DE PROCESAMIENTO GRÁFICO

La tecnología actual brinda un mundo de posibilidades en cuanto a equipos usados para procesamiento de imágenes desde mini computadoras, teléfonos inteligente hasta sofisticados ordenadores, teniendo así una amplia gama de posibilidades para realizar procesamiento digital de video.

Al tener una aplicación abierta como la del presente proyecto, es menester que se quiera usarla en distintos entornos, por ello software desarrollado puede ser ejecutado en cualquier ordenador que cumpla con los requerimientos básicos detallados en el anexo 1, dicho computador será un complemento al sistema, desempeñando como función principal el procesamiento digital de imagen video sin dejar de lado la interfaz con el usuario.

3.2.2 SENSORES UTILIZADOS

Para tener un ambiente de manejo así como cumplir con requisitos de seguridad del robot, será necesario dotar al prototipo la debida instrumentación para la adquisición de los datos demandados en el desarrollo de un sistema amigable y seguro.

3.2.2.1 Temperatura

Debido a que la temperatura no es una variable que interviene directamente con el proceso, la precisión en la medida de dicha variable no es de gran trascendencia. Se desea que el usuario tenga una noción de la temperatura ambiente en el entorno en el que se mueve el robot. Por ello no es necesario un sensor digital de gran precisión, así que se usa el LM35 que es un sensor económico y simple. La salida de este sensor es de 10mV/ºC y puede ser polarizado con 5V ideal para trabajar con niveles de voltaje TTL [37].

Fig. 3.2 Sensor de temperatura LM35.

3.2.2.2 Velocidad

Para tener una medida velocidad se acopla al eje de tracción, un encoder tipo herradura, que funciona en base a un foto sensor. Al interponerse un elemento que impida el paso del haz entre emisor y receptor, se conmuta una salida transistorizada. Se genera una señal pulsante proporcional a la frecuencia con que un elemento acoplado directamente a la rueda corta el haz entre emisor y receptor del sensor.

Fig. 3.3 Encoder usado para medir velocidad.

3.2.2.3 Distancia

Tener seguridad en cuanto a choques es primordial para garantizar la integridad del robot, debido a ello el robot estará dotado de un par de sensores ultrasónicos que den una señal de alerta en presencia de obstáculos tanto adelante como atrás. Los sensores envían una señal de ultrasonido de 40 [KHz] que al encontrar un obstáculo rebota, así que en base a una medida de tiempo se logra tener indirectamente una medida de distancia, esta medida servirá para conocer la distancia con respecto a los obstáculos del robot y tomar acciones de control.

Para la aplicación se elige el sensor HC-SR04 por ser de bajo costo, ideal para el prototipo de prueba implementado que mide de 2 [cm] a 4 [m] y funciona con tecnología TTL.

Fig. 3.4 Sensor ultrasónico HC-SR04 [38]

3.2.2.4 Video Robot.

El robot implementado estará provisto de una cámara inalámbrica. Se ha seleccionado una cámara SkyIPCam, la cual cumple con las características detalladas en la siguiente tabla.

CARACTERÍSTICA	Especificación
Voltaje	5 [V_{Dc}]
Corriente	2.5 [A]
Potencia	5 [W]
Resolución	640 x 480 (VGA) 10fps
Peso	235 [g]
Comunicación	IEEE 802.11g

Tabla 3.1 Características de la cámara IP [39].

La razón de usar esta cámara es que al ser Wi-Fi se la puede integrar a la red del usuario o configurar una red propia para el sistema, además se alimenta con 5 V, hecho que ayuda en cuestión de alimentación y autonomía. Se debe considerar el peso de la misma al seleccionar los motores del robot.

Fig. 3.5 Cámara IP Dericam M501W [39].

3.2.2.5 Video Usuario

Para detectar el rostro del usuario se requiere una Webcam tal y como se ha desarrollado la aplicación. Para que el sistema funcione correctamente la webcam debe tener un comportamiento adecuado. Por ello una correcta selección de esta cámara es trascendental para el éxito del proyecto. Aunque para proyectos en procesamientos de imágenes se usan cámaras sofisticadas tipo Estéreo, su elevado costo ha hecho desistir de esta opción y se ha elegido una webcam de bajo costo pero buenas características, Logitech c170 cuyas características se describen en la siguiente tabla.

CARACTERÍSTICA	Especificación
Voltaje	5 [V_{Dc}]
Corriente	500 [mA]
Potencia	5 [W]
Calidad de Video	VGA
Comunicación	Hi-Speed USB 2.0
Resolución en video	640*480 [Pixels]
Calidad de fotografías	5[Mp]

Tabla 3.2 Características de la Webcam [40].

Como la aplicación se desarrolla para tiempo real se utilizó una resolución de 640*480 para establecer las referencias en el control, con esta cámara se logra un buen comportamiento en la detección de rostro y respectivo movimiento y forma parte del sistema, y se la conectará al computador donde se ejecute la aplicación.

Fig. 3.6 Webcam Logitech c170.

3.2.3 ACTUADORES

3.2.3.1 Motor Tracción

Debido al peso que implica la cámara y la fuente de alimentación para el robot mismo, se requiere un motor con un buen torque de arranque y fácil de controlar. Otro aspecto a considerar es que los alcances del proyecto plantean hacer un control simple de un robot móvil, por lo que la velocidad del mismo no es una variable crítica, ya que el fin es controlar el movimiento del robot y no una variable específica.

Tomando en cuenta las causales anteriores además que se requiere un motor que no emita gran cantidad de ruido, ya que pudiera afectar la adquisición de datos, se buscó un motor DC pequeño que se ajuste a las necesidades. El motor usado para la tracción del robot trabaja a 12V y se lo acopla a una caja reductora de relación de engranes 32/8 con lo que se logra la movilidad del robot.

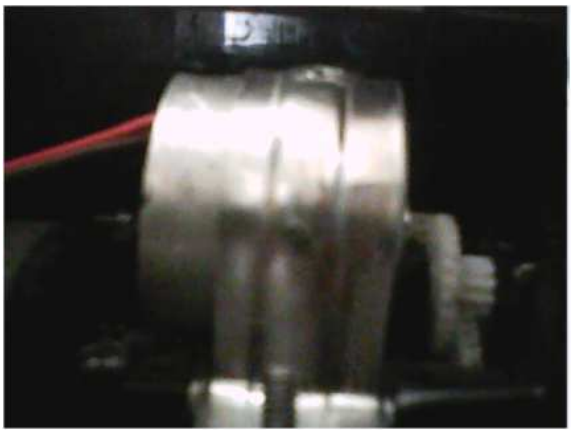


Fig. 3.7 Motor DC usado para tracción del robot.

3.2.3.2 Motor Dirección

Se requiere controlar la dirección del robot sea derecha o izquierda mediante un mecanismo acoplado a las llantas delanteras. Para controlar este mecanismo se utilizará un servo-motor Tower Pro MG996R que desarrolla un torque máximo al trabajar con 6[V] y se lo controla fácilmente con una señal PWM de frecuencia aproximada de 60 [HZ]

Fig. 3.8 Motor DC usado para dirección del robot.

3.2.4 CONTROL

Debido a la importancia en la detección de obstáculos para seguridad y a la complejidad del control de los sensores ultrasónicos, se independiza el control del robot y el control de los sensores ultrasónicos. Sin embargo se utilizarán las interrupciones del micro-controlador principal para interpretar a los micro-controladores asociados al control de los sensores ultrasónicos.

3.2.4.1 Control del Robot

Debido a que se requieren controlar 2 motores de forma independiente, comunicación USART, medición análoga de varias variables en indicadores de luces se opta por realizar el control con un micro ATMEGA164P que cumple con los requerimientos de la aplicación. Su funcionamiento en la aplicación se detalla en la sección 3.5 (Diseño del software para control del Robot) [41].

Fig. 3.9 Atmega164P usado como micro-controlador principal.

3.2.4.2 Controlador para Sensores Ultrasónicos.

Los sensores ultrasónicos usados tienen como salida una señal que está ligada al tiempo que demora en retornar el eco tras rebotar en el obstáculo. Se debe medir este tiempo, interpretar si existe un obstáculo peligroso y enviar al micro-controlador principal una señal que indique en caso de existir, presencia de obstáculo. Como esta tarea no es muy complicada y no se requiere de muchos recursos en el micro-controlador se elige un par de ATTINY13 para controlar e interpretar a cada sensor ultrasónico (un micro por sensor) ya que este es el micro-controlador más económico y simple encontrado en el mercado nacional [42].

Fig. 3.10 Attiny13.

3.2.5 DISPOSITIVOS PARA COMUNICACIÓNES

3.2.5.1 Transmisión de Datos y Comandos

La comunicación inalámbrica bidireccional entre robot-interfaz es esencial para la tele-operación en el sistema hombre-máquina implementado. Para efectuar dicha comunicación se usa un módulo Bluetooth HC 05 que se enlaza al micro-controlador principal. Además se usa un adaptador genérico USB-Bluetooth para conectarlo al lado del computador donde se desee ejecutar la aplicación realizada.

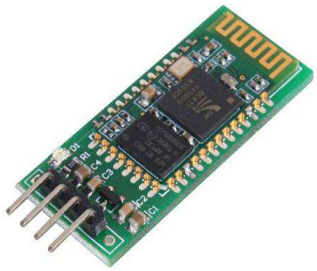


Fig. 3.11 Modulo Bluetooth para la comunicación.

3.2.5.2 Transmisión de Video

Para que el Usuario tenga el video del entorno del robot, se debe transmitirlo en forma inalámbrica. Puesto que la cámara seleccionada es una cámara IP inalámbrica y en la interfaz se ha configurado para acceder a la misma mediante un Web Browser; se debe acceder a la misma red en la que se encuentre la computadora en donde se ejecuta la interfaz para ello se requiere que el Router de la red TCP/IP a la que se enlaza la cámara del robot sea inalámbrico.

Fig. 3.12 Router inalámbrico.

3.2.6 FUENTE DE ALIMENTACIÓN

Por razones de autonomía y que el consumo de la cámara es elevado, se independizan las fuentes de alimentación sean una para el robot en sí y otra únicamente para la cámara, sin embargo deberán tener la misma referencia ya que se debe sondear el porcentaje de energía que tiene cada batería para saber cuándo se las debe recargar.

3.2.6.1 Batería para Robot

Para el robot se usa una batería seca de 12V y 1.2 A/h/20Horas de la marca First Power, la misma abastecerá de energía a los motores de tracción y dirección así como al circuito de control con todos sus periféricos.

Fig. 3.13 Batería 12[V].

3.2.6.2 Batería para Cámara de Robot

Para la cámara del robot se usa una batería seca de 6V y 1.2 A/h/20Horas de la marca First Power, abastecerá únicamente a la cámara IP inalámbrica.

Fig. 3.14 Batería 6[V].

3.2.7 ARQUITECTURA DEL SISTEMA

En el siguiente diagrama se muestra la arquitectura general del sistema implementado incluyendo todos los equipos y elementos que intervienen en el diseño del mismo. Con este antecedente se diseña el hardware para que todos los elementos interactúen cumpliendo los requerimientos del sistema.

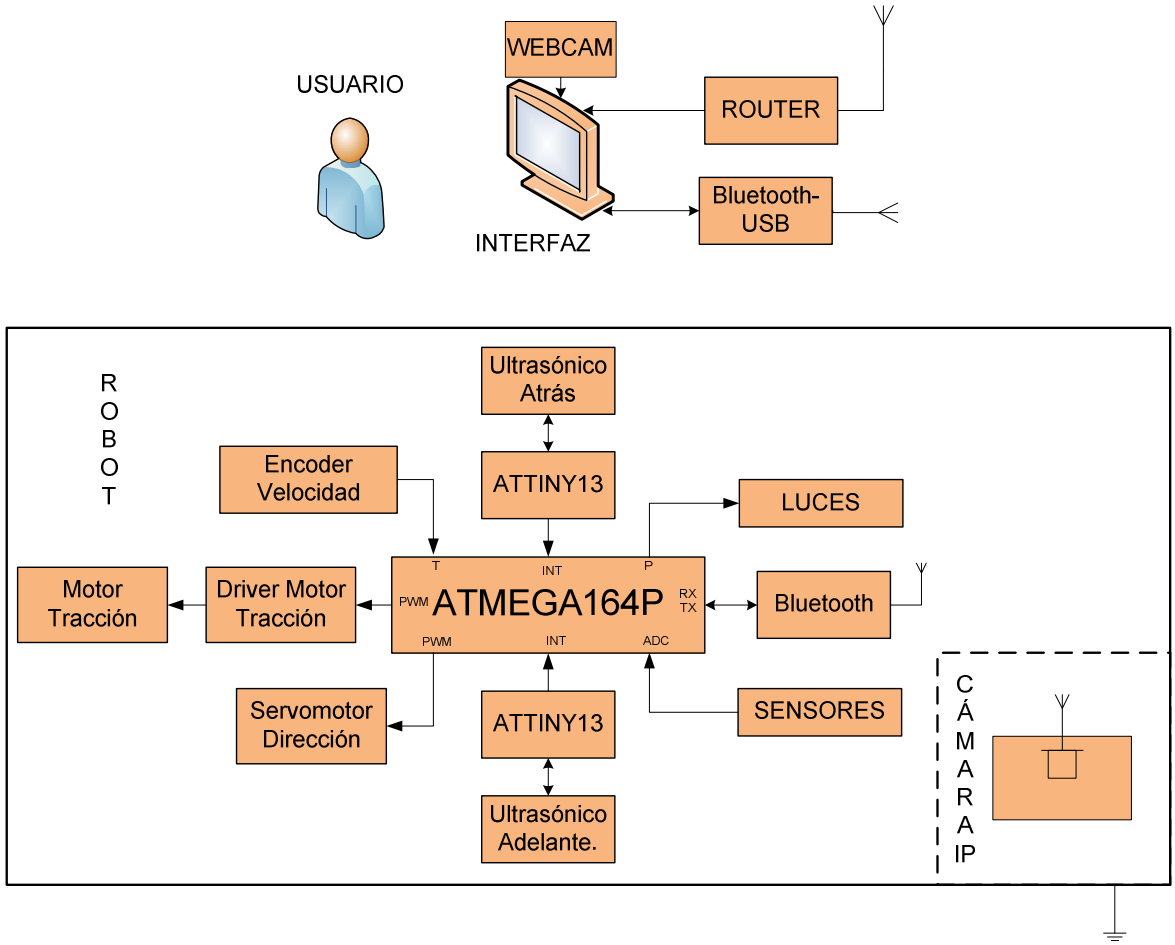


Fig. 3.15 Arquitectura del sistema implementado.

3.3 CONFIGURACIÓN DE EQUIPOS

En esta sección se describe la configuración requerida para los equipos que pertenecen al sistema.

3.3.1 CONFIGURACIÓN DE LA CÁMARA IP

La cámara IP se la debe configurar de modo tal que se comunique con la interfaz de forma inalámbrica, previamente se debe asignar una IP estática con la herramienta “AirLink101 IP Camera Setup Wizard” incluida para poder acceder a la interfaz de la cámara desde un Web browser o la aplicación incluida “IPViewPro” con la cámara. Para sincronizar con la red inalámbrica se deberá configurar con las mismas características Wireless del router al que se ligue la cámara. Para la aplicación se accede al video del robot a través de la aplicación incluida en donde se detecta automáticamente la dirección de la cámara y se

agrega la misma. El video se lo verá con pantalla completa para tener una ambiente de manejo más real. Más detalles en el manual de Usuario, Anexo 1.

3.3.2 CONFIGURACIÓN DEL MÓDULO BLUETOOTH

El módulo bluetooth seleccionado viene por default a 9600 baudios, sin embargo en la aplicación desarrollada se trabajará con 4800 baudios para tener un error mínimo en la comunicación. Para cambiar este valor se comunica el módulo bluetooth con los valores dados por default y se lo configura mediante comandos AT detallados en la documentación del módulo por ejemplo AT+UART=4800 para configurar con baudrate de 4800 [43]. Para tener comunicación con el computador se usa un adaptador bluetooth-USB así se accede al módulo y se lo configura como un puerto serial que será el que se use para la comunicación robot-interfaz. Más detalles en el manual de Usuario, Anexo 1.

3.4 DISEÑO DEL HARDWARE PARA CONTROL DEL ROBOT.

Puesto que el control se realizará con micro-controladores Atmel en encapsulados PDIP se requiere trabajar con alimentación de 5V, por otra parte la batería seleccionada para alimentar al robot es de 12 [V]. Debido a lo anterior, es requerido usar un regulado LM7805 para alimentar la placa de control del robot, incluyendo el hardware de control del robot, con excepción del servomotor que se alimenta con 6 [V] a través de un LM 7806, y del módulo Bluetooth que se alimenta con 3.3 [V] a través de un regulador LM1117. La cámara se alimenta con un arreglo de 3 LM7805 dispuestos en paralelo para garantizar tener 5 [V] durante todo el tiempo de autonomía de la batería seleccionada específicamente para la cámara.

3.4.1 CONTROL DE MOTOR DE TRACCIÓN

Para controlar este motor se usará un Driver L298 en encapsulado multiwatt15 que con una configuración en paralelo soporta una corriente de hasta 3.5A que permite el control del motor seleccionado, los valores máximos que soporta este componente se resumen en la siguiente tabla

Parámetro	Valor	Unidad
Voltaje Potencia	50	[V_{Dc}]
Voltaje Control	7	[V_{Dc}]
Corriente no Repetitiva (100µs)	3	[A]
Potencia	25	[W]

Tabla 3.3 Características del L298 [44].

Para la aplicación se usa el L298 conectado en paralelo, es decir usando los dos puentes que posee, además se usan los diodos fast recovery “FR104” con un trr 150µs que cumple las especificaciones dadas para la configuración del driver con carga inductiva. Para controlar este el motor con este driver se requieren de dos pines que establezcan la dirección y otro que habilite en driver con lo que se logra tener control de velocidad y dirección.

Por otra parte el voltaje para el motor se lo toma directamente de la batería de 12 [V] seleccionada para el robot. Para controlar este este motor se deberá usar una señal PWM variable de aproximadamente 8 [KHz] enlazada al pin de habilitación del driver.

3.4.2 CONTROL DE MOTOR DE DIRECCIÓN

Se ha seleccionado un servo-motor cuyo torque máximo se logra al aplicar 7V al mismo, por encontrar en el mercado el regulador más cercano a este valor, el LM7806, se lo elije para alimentar el servo-motor, es decir reducirá los 12 [V] de la batería a 6[V] para alimentar el servo. El control del servomotor es simple, se requiere una señal PWM de una frecuencia de 50[Hz] cuyo pulso en nivel alto tenga una duración de entre 0.6 a 2.4 [ms] con lo que se logra un control continua del eje del motor, para la aplicación se debe calibra los valores de tiempo en alto de tal forma que se tenga la dirección del robot completamente a la derecha, a la izquierda o al centro para efectos de control del mismo. La señal de control se la enlaza directamente del micro-controlador al servo.

3.4.3 LUCES INDICADORAS

El robot se lo diseña con luces que indiquen el estado del mismo para efectos de realizar pruebas y para tener indicadores en funcionamiento. Para ello se utilizan diodos led, los mismos que indicaran; giro derecho, giro izquierdo, retro. Se reserva un puerto del micro-controlador para salidas de luces en donde se usan resistencias limitadoras de corriente de 330 [Ω] a ¼ [W].

3.4.4 USART-BLUETOOTH

El módulo bluetooth usado trabaja con 3.3 voltios, por lo que es necesario que se acople a los niveles TTL que utilizan el micro-controlador principal y usar un regulador de voltaje LM1117 para energizar el módulo. Para acoplar el Atmega146P y el módulo bluetooth se usa un arreglo de resistencia y zener para el receptor del módulo. El transmisor del módulo se conecta directamente al micro, pues 3.3V se reconoce como nivel alto en TTL. Se muestra a continuación el circuito usado para el efecto.

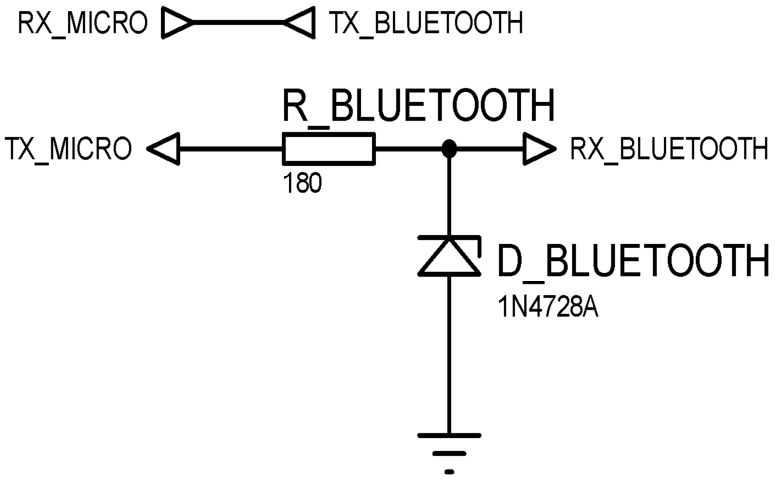


Fig. 3.16 Conexión entre micro-controlador y módulo bluetooth.

3.4.5 MEDICIONES

Para medición análoga se usará la referencia interna de 2.56 [V] y 10 bits de resolución leídos como ADC, los mismos que serán filtrados para tener una muestra con el menor valor posible para enviar hacia la interfaz.

3.4.5.1 Medida de Temperatura

Para medir temperatura con el LM35 se requiere una medición análoga que interprete 10[mV/ºC], por ello usando como referencia para la conversión 2.56 [V] interna en el micro y una resolución de 10 bits se tendría teóricamente 256[ºC] para un valor de conversión de 1024, así que se tiene como ecuación.

$$T[^{o}C] = ADC_T/4 \qquad (3.1)$$

3.4.5.2 Medida de Velocidad

En la medición de velocidad se dará lectura al registro de un timer a intervalos de tiempo continuos, al mismo que se le asignará como señal de reloj una señal externa dada por el encoder acoplado al eje de tracción. El valor leído corresponde a 8 pulsos por vuelta y una lectura a los 20 [ms] se le acondiciona con la siguiente ecuación.

$$v[m/min] = \frac{15}{4} \cdot TCNT_x \qquad (3.2)$$

El hardware usado para el encoder corresponde al de un opto-acoplador normal que se polariza con los mismos 5 [V] que alimenta la placa de control, el haz de luz se interrumpe al girar la rueda y se genera pulsos que son medidos en el micro-controlador, por software se limita este valor hasta 15 m/min.

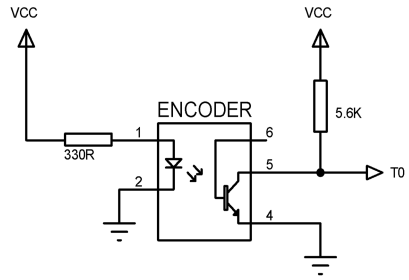


Fig. 3.17 Hardware para el encoder.

3.4.5.3 Medida de Porcentaje de la Batería del Robot

Para sondear el estado de las baterías se debe conocer el voltaje en sus terminales con carga, para ello se usa un divisor de voltaje en el que se medirá el voltaje directamente desde un canal análogo del micro-controlador con referencia 2.56 [V], para el diseño del divisor nos basamos en la Fig. 3.18 donde V_2 es el voltaje de la batería que se mide y V_1 es el voltaje que se le aplica al canal análogo encargado de la medición.

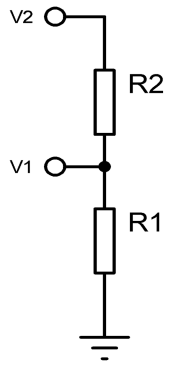


Fig. 3.18 Divisor de voltaje para sondear estado de baterías.

Se realiza el análisis del circuito para obtener el valor de R2 en función de las otras variables.

$$V_2 - V_1 = V_2\left(\frac{R_2}{R_1 + R_2}\right) \qquad (3.3)$$

$$(V_2 - V_1)\cdot(R_1 + R_2) = V_2 \cdot R_2 \qquad (3.4)$$

$$(V_2 - V_1)\cdot R_1 + (V_2 - V_1) R_2 = V_2 \cdot R_2 \qquad (3.5)$$

$$(V_2 - V_1)\cdot R_1 + (V_2 - V_1 - V_2)\cdot R_2 = 0 \qquad (3.6)$$

$$R_2 = \frac{(V_2 - V_1)\cdot R_1}{V_1} \qquad (3.7)$$

Para el caso de la batería del robot se asume R_1=1[K], V_2=12[V], V_1=2.56 [V], con lo que calculando y eligiendo un valor estándar R_2=4.7 [K]. Los valores de voltaje a los terminales de la batería deberán fluctuar entre -5% y +10 % del voltaje nominal siendo para el caso entre 11.4[V] y 13.2[V] correspondientes al 0 y 100% de autonomía de la batería, así con el divisor de voltaje diseñado al canal analógico correspondiente le llegará un voltaje entre 2.0[V] y 2.31 que con una resolución del conversor de 10 bits corresponde a valores entre 800 y 924 que serán los límites para acondicionar esta variable en el micro-controlador. Tomando en cuenta estos valores se llega a la ecuación de porcentaje de batería del motor (BAT_MOT) en función del valor análogo leído en el divisor de voltaje.

$$BAT_{MOT[\%]} = \frac{25}{31}\cdot ADC_{BAT_{MOT}} - 645 \qquad (3.8)$$

3.4.5.4 Medida de Porcentaje de la Batería de la Cámara del Robot

Con el sondeo de la batería de la cámara del robot se lleva a cabo el mismo procedimiento del literal anterior variando el valor V_2=6.0 [V] resultando R_2=1.8 [K], así los valores correspondientes al 0 y 100% de autonomía de la batería medidos sobre divisor de voltaje diseñado fluctuarán entre 2.03 y 2.35 que con una resolución del conversor de 10 bits corresponde a valores entre 812 y 940. La ecuación correspondiente (BAT_CAM) en función del valor análogo leído en el divisor de voltaje será (3.9).

$$BAT_{CAM[\%]} = \frac{25}{32} \cdot ADC_{BAT_{CAM}} - 634 \qquad (3.9)$$

3.4.5.5 Medida de Distancia a Obstáculos.

Se han seleccionado dos microcontroladores independientes que informen al micro principal la presencia o no de obstáculo sea adelante o atrás del robot, para ello se realiza un programa que cumpla la tarea de disparar un sensor ultrasónico e interpretar su señal de retorno. Para ello se envía una pulso de 10 [us] a una frecuencia de aproximadamente 16 [Hz], se lee con un timer el tiempo que la señal ECHO del sensor permanece en nivel alto, este tiempo es directamente proporcional a la distancia. Se compara este tiempo para que cuando la distancia a un obstáculo aproximadamente sea menor a 10 [cm] se active la alerta y cuando sea mayor a 15 dicha alerta desaparezca.

$$[cm] = [\mu s]/58 \qquad (3.9)$$

Usando la expresión (3.9) tomada de [2] se tienen los tiempos aproximados entre los cuales efectuar el control por histéresis para activar o desactivar un pin que le informa al micro-controlador principal la presencia o ausencia de obstáculos que pongan en peligro la integridad del robot. Se esboza a continuación los diagramas de flujo que resumen el programa implementado para cada ATTINY13.

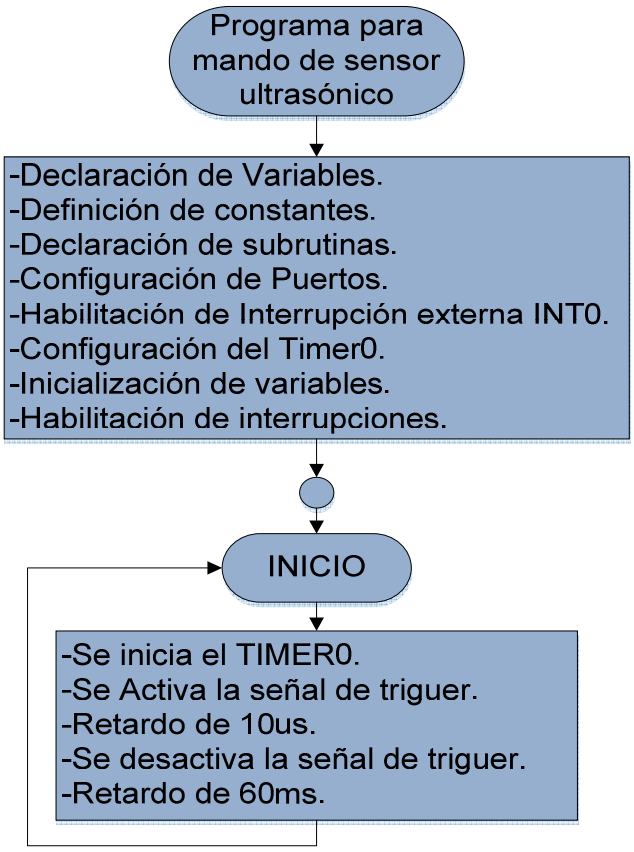


Fig. 3.19 Diagrama de flujo para el control de sensores ultrasónicos.

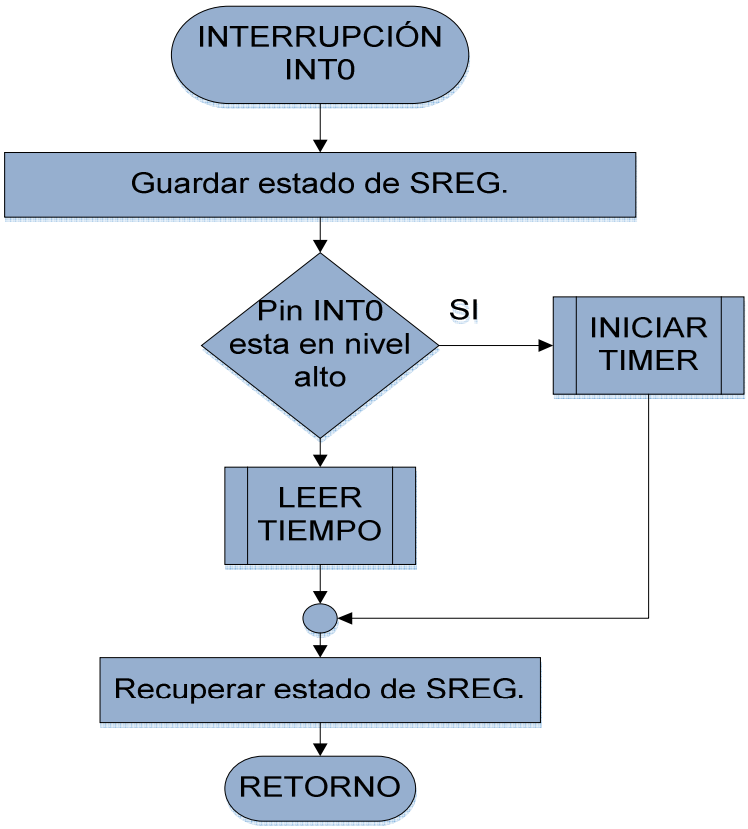


Fig. 3.20 Diagrama de de interrupción externa conectada al pin ECHO.

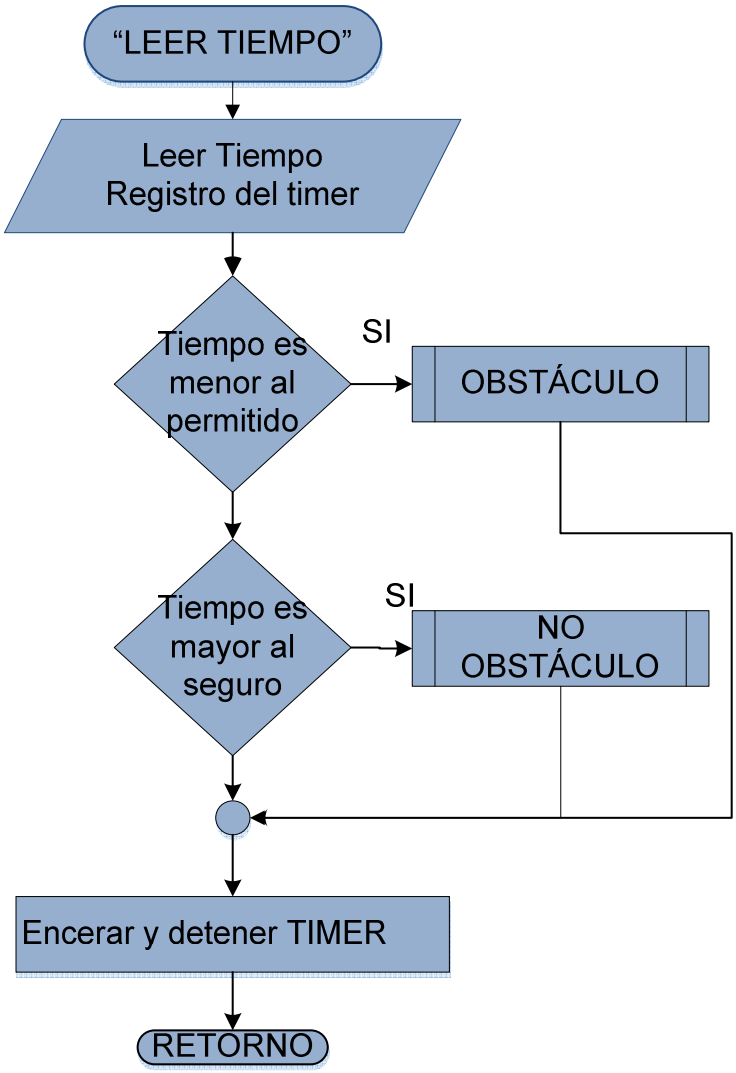


Fig. 3.21 Diagrama de flujo Subrutina para leer el tiempo.

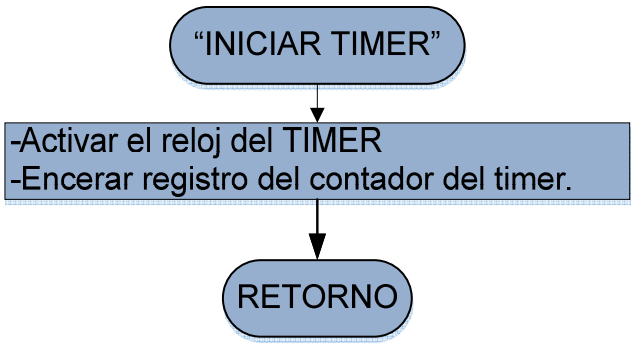


Fig. 3.22 Diagrama de flujo Subrutina para iniciar TIMER0.

Fig. 3.23 Diagrama de flujo Subrutina si existe obstáculo.

Fig. 3.24 Diagrama de flujo Subrutina no existe obstáculo.

3.4.6 Asignación de pines del Micro-controlador principal.

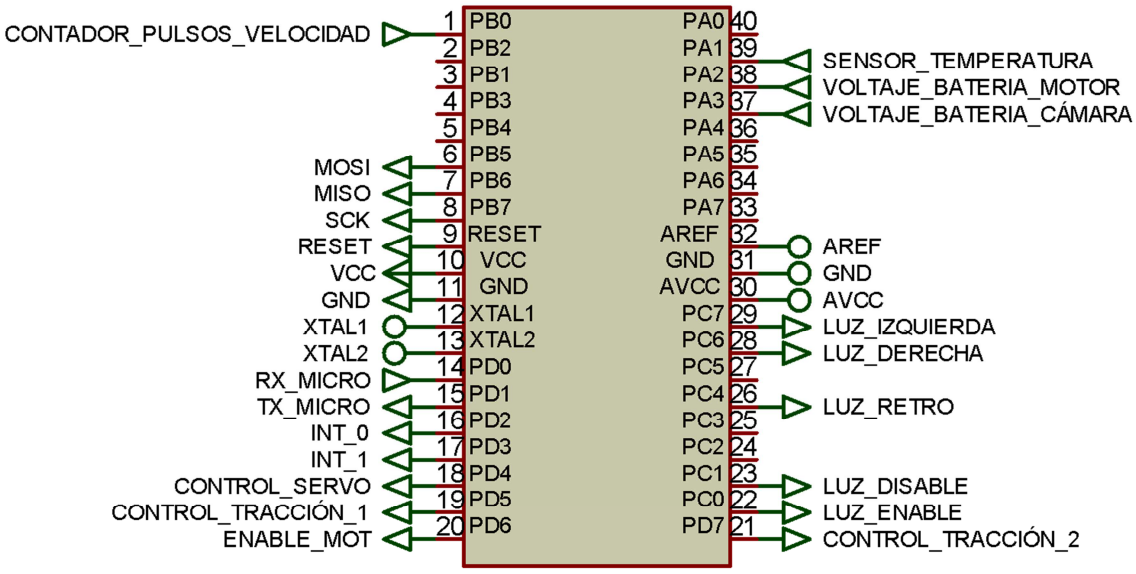


Fig. 3.25 Asignación de pines del controlador principal.

3.5 DISEÑO DEL SOFTWARE PARA CONTROL DEL ROBOT.

El control del robot se lo programa sobre un ATMEGA164 P, el cual trabaja a 20 MHz para este proyecto. En base a este hecho se efectúan las configuraciones del micro-controlador para cumplir los requerimientos del robot interactuando con el hardware descrito anteriormente.

Se programa en forma modular a manera de subrutinas que se efectúan al ocurrir las interrupciones pertinentes, por lo que en el programa principal solo se efectúan las configuraciones respectivas y se ejecuta un bucle a la espera del requerimiento solicitado por alguna interrupción.

Se efectúa la declaración de variables, subrutinas y constantes necesarias para la programación. Se configura los puertos acorde a las necesidades del hardware según la asignación de pines expuesta en la sección 3.4.5 de este documento. Se configuran las interrupciones, la primera para que se ejecute una subrutina al recibir un dato en el USART0_RX, otra subrutina se ejecuta al detectar flanco de bajada en INT0 que indica que se ha detectado un obstáculo adelante y por último la interrupción en INT1 como flanco de bajada que indica que se ha detectado un obstáculo atrás.

Se requiere configurar los 3 timers con los que cuenta el ATMEGA164P, el timer 0 se usa como contador de pulsos que servirá para la medición de velocidad en T0 para ello se lo configura en modo “NORMAL” y el reloj en el pin T0. El timer 1 se usará para generar la señal PWM de frecuencia 50 Hz constante para controlar el servomotor; se genera la señal en el pin OC1B y se configura en modo Fast PWM con desborde en OCR1A, además se habilita una interrupción al existir un desborde del timer1 para medir las variable y enviarlas hacia la interfaz con ello se logra un periodo de muestreo de 20ms usando un pre-escalador que divide la señal de reloj para 8 y asignando al desborde un valor de 50000. El timer 2 se usa para generar la señal PWM para controlar el motor de tracción, se genera la señal por el pin OC2B, la configuración inicial se la hace para que el pin este deshabilitado con el fin de no forzar al motor cuando el robot esté detenido, este funciona en modo FAST PWM con desborde en OCR2A al que se asigna un

valor de 78 y usando un pre-escalador que divida la señal de reloj para 256 se logra una frecuencia de la PWM aproximadamente de 1KHZ la activación y configuración total de este timer se la hace al momento de acelerar, de la misma manera al detener el robot se desactiva el pin por el que sale la PWM y se lo deja de modo que no intente activar el motor de tracción.

Para configurar el ADC es necesario usar un factor de escala de 128 cuando se usa un cristal de 20MHz, por facilidad se lo configura en modo free running. Debido a los requerimientos del hardware diseñado para el robot se requiere tener como referencia para la conversión 2.56 que es una referencia interna del ATMEGA164P. Por último en la configuración se inicializan las variables requeridas para la programación y se inicia la conversión del ADC se entra en un lazo infinito y en adelante todo el control se ejecuta por interrupciones.

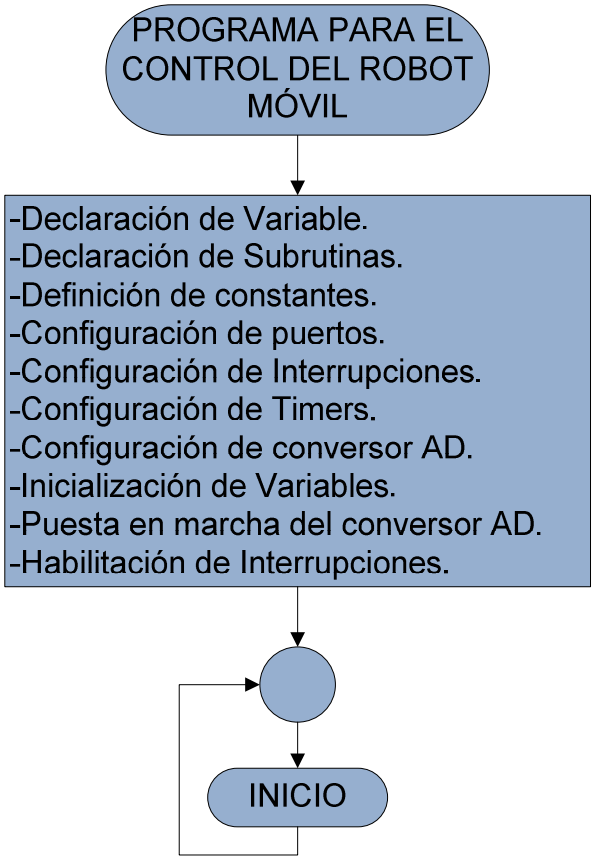


Fig. 3.26 Diagrama de flujo de configuración inicial (Robot Móvil).

Al recibir un dato en el USART0 se ejecuta una subrutina para interpretar los comandos que se envían desde la interface, la lógica de esta subrutina se detalla en el siguiente diagrama de flujo.

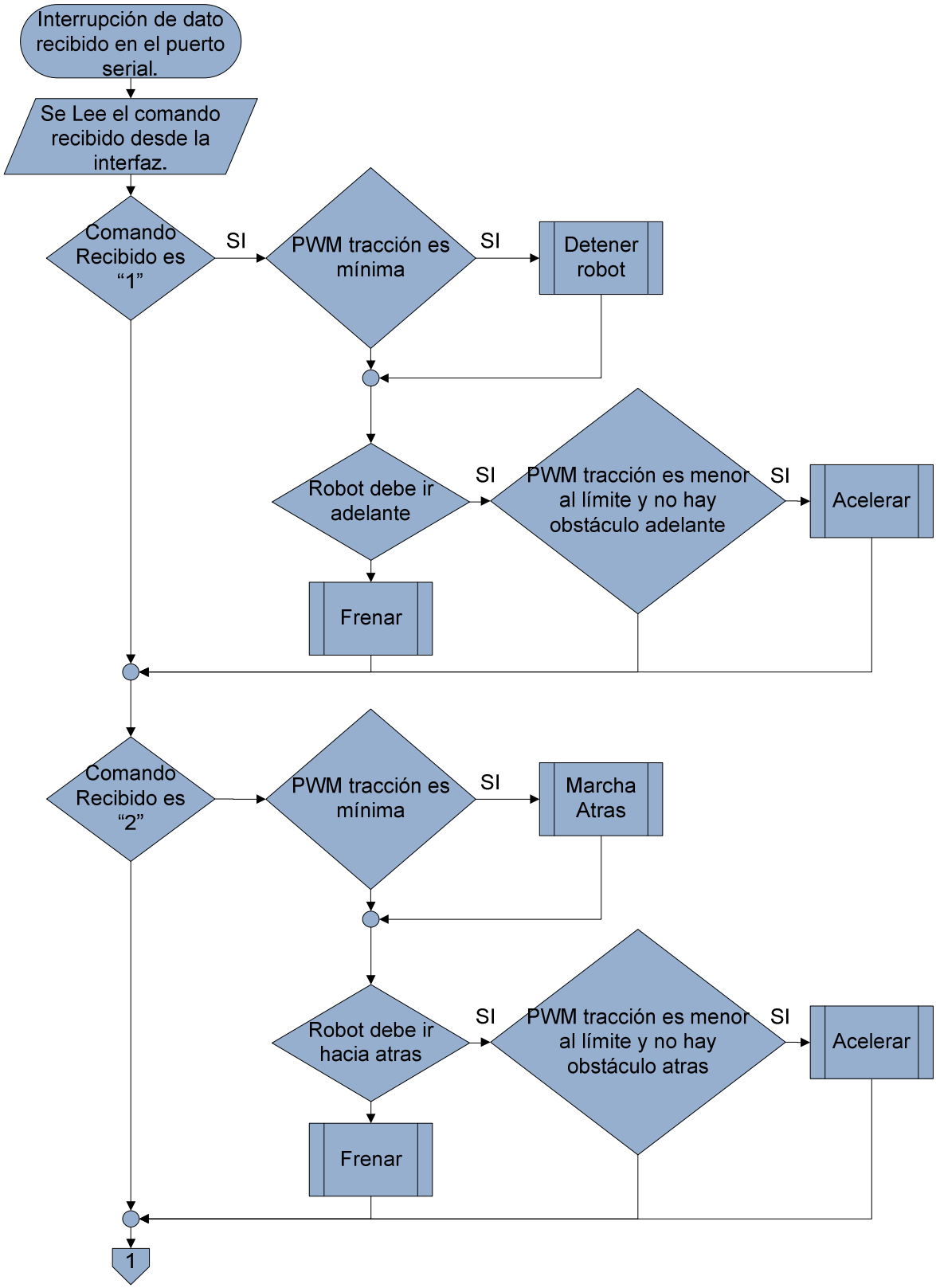


Fig. 3. 27 Diagrama de flujo de interrupción por dato recibido (a).

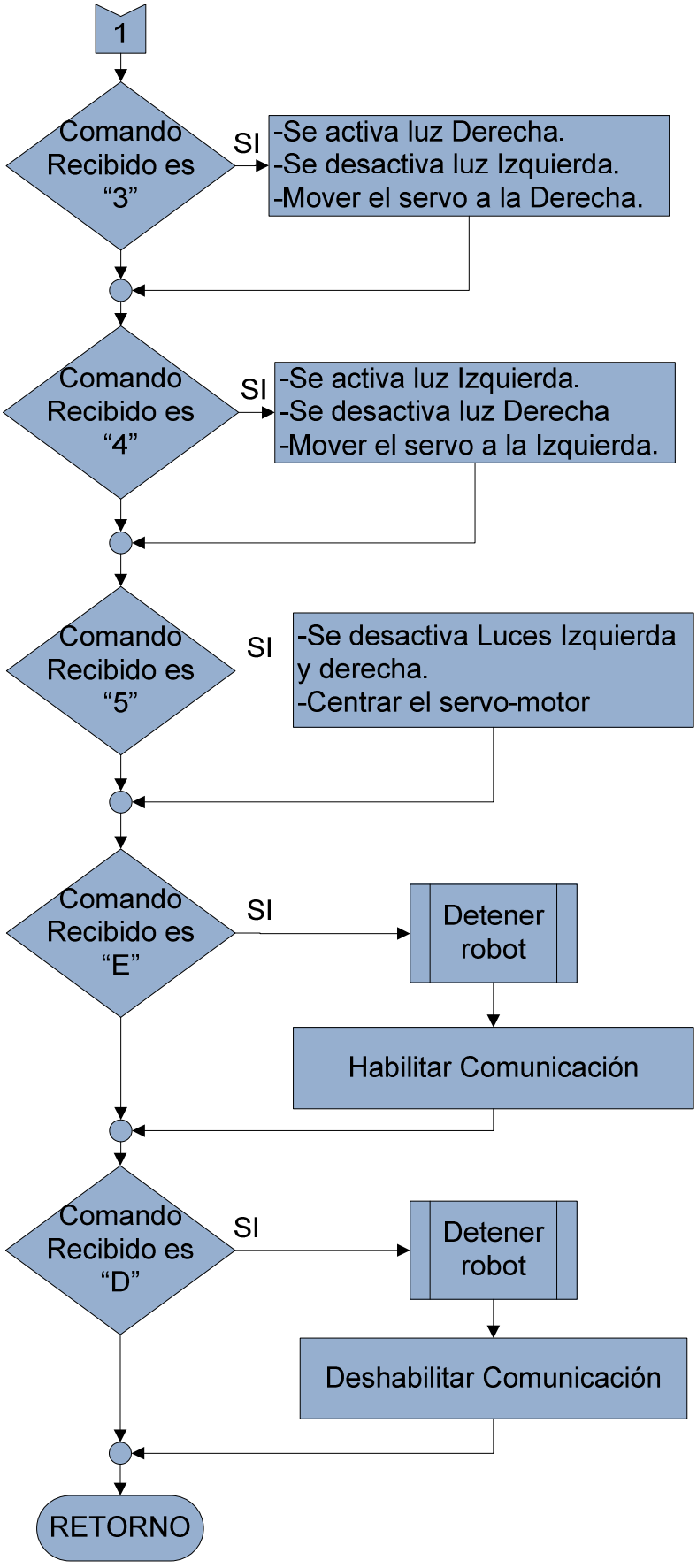


Fig. 3. 28 Diagrama de flujo de interrupción por dato recibido (b).

El timer1 se ha configurado para que en un tiempo de 20ms se desborde produciendo una interrupción, en la cual se efectúa la medición de canales analógicos y de pulsos asociados a la velocidad del robot. Para disminuir el error en la medida se toman 15 muestras; con ello cada 300ms se hace el procesamiento de las muestras para enviar los valores de las variables requeridas hacia la interfaz. El procesamiento y envío de las variables medidas se hace siempre y cuando la comunicación haya sido habilitada desde la interfaz. Para efectuar dentro del micro-controlador los requerimientos planteados, se sigue la lógica expuesta en el siguiente diagrama de flujo.

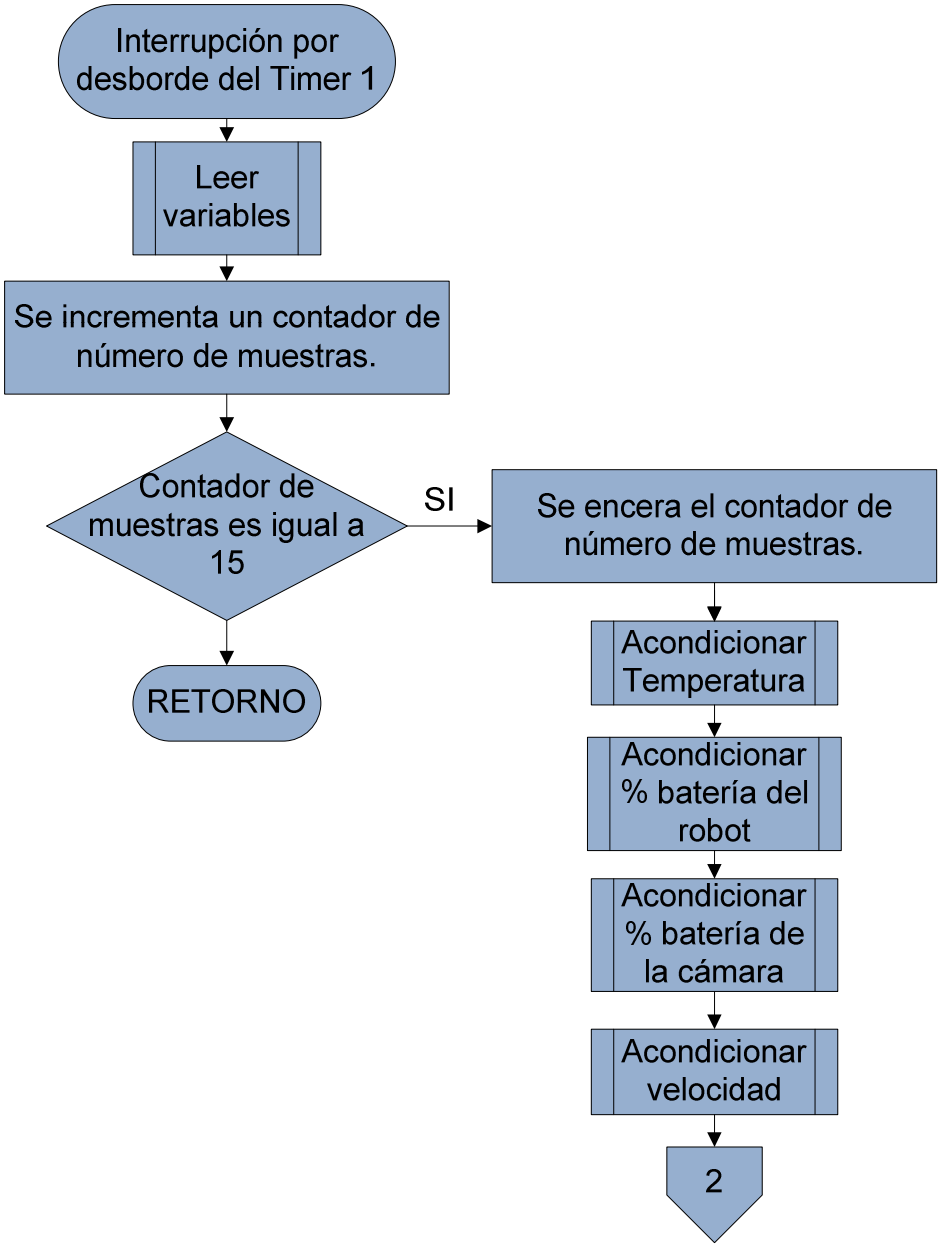


Fig. 3. 29 Diagrama de flujo de interrupción por desborde del TIMER1 (a).

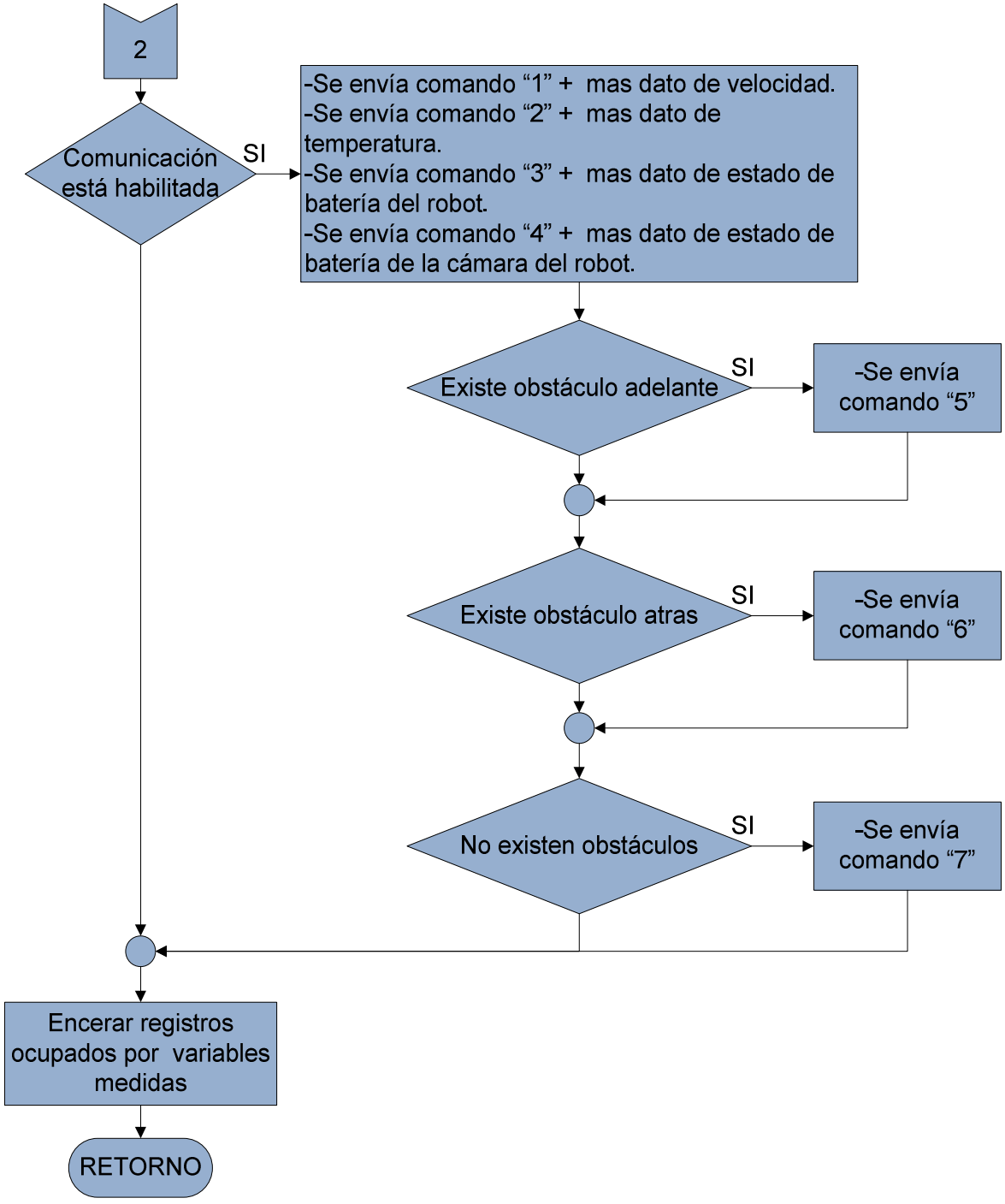


Fig. 3. 30 Diagrama de flujo de interrupción por desborde del TIMER1 (b).

Para obtener una mayor seguridad en el robot en cuestión de choques, se implementa la parada instantánea al detectar un obstáculo que ponga en peligro la integridad del robot. Para ello, se usan las interrupciones externas del micro-controlador principal configuradas para detectar un cambio a cero lógico.

Se presentan los diagramas de flujo asociados a las interrupciones externas para detección de obstáculos, realizada por los ATTINY13 que controlan los sensores ultrasónicos ubicados adelante y atrás del robot. Estos últimos envían una señal que será interpretada por el micro-controlador principal a manera de interrupción externa.

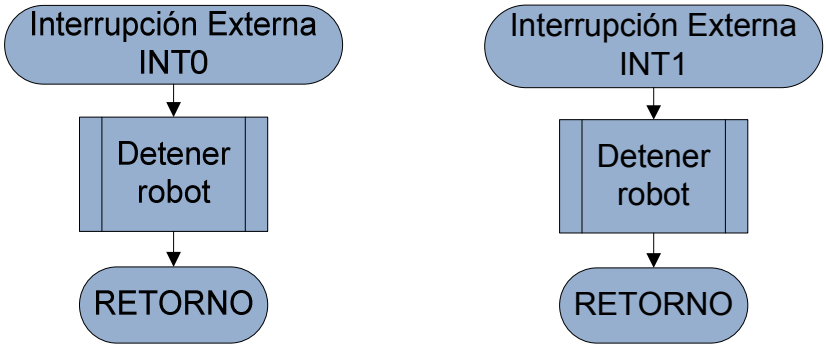


Fig. 3. 31 Diagrama de flujo de interrupciones externas existir obstáculo.

A continuación se describen las subrutinas usadas para realizar una programación eficiente. La primer subrutina que se describe es “Detener robot” en donde se cumple la función de apagar el pin que habilita al driver del motor de tracción y centrar el servo-motor, además se configuran las variables del robot para que se regrese a condiciones iniciales, donde la marcha se establece seguir hacia adelante.

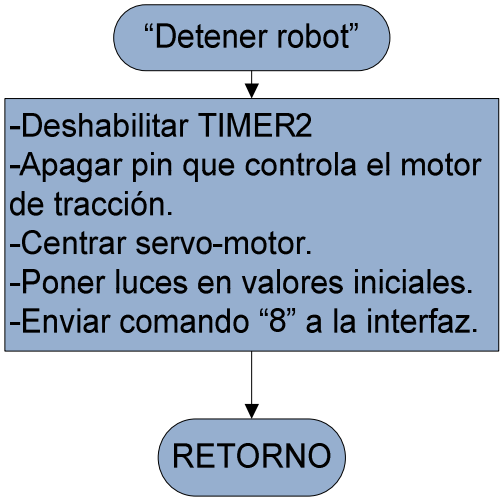


Fig. 3. 32 Diagrama de flujo de Subrutina para detener el robot móvil.

En la siguiente subrutina se establece el cambio de marcha hacia atrás con lo que el robot cambia su dirección, además informa al interfaz dicho evento.

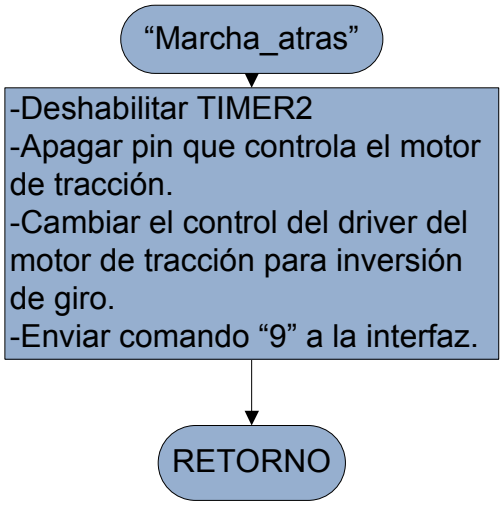


Fig. 3. 33 Diagrama de flujo de Subrutina para cambiar la dirección del robot hacia atrás.

En la subrutina “Acelerar” se habilita la señal PWM que controla el driver del motor de tracción y se incrementa la relación de trabajo, esto actúa para acelerar hacia adelante o hacia atrás según la dirección establecida.

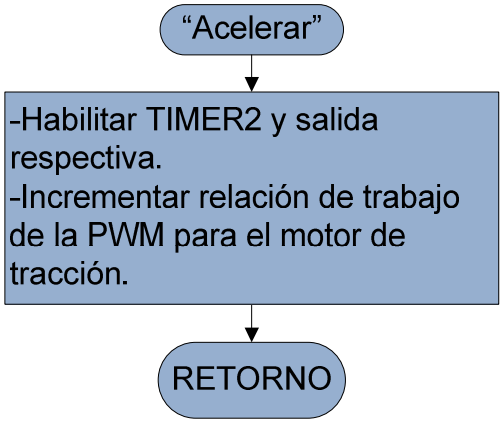


Fig. 3.34 Diagrama de flujo de Subrutina para acelerar el robot móvil.

En la subrutina “Frenar” solo se disminuye la relación de trabajo de la señal PWM que controla al motor de tracción no se está por debajo del límite establecido para mantener al robot en movimiento.

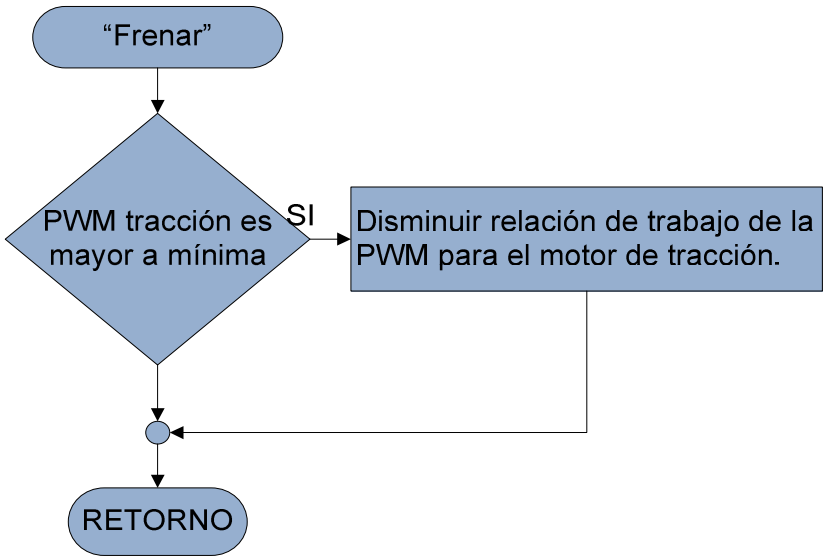


Fig. 3.35 Diagrama de flujo de Subrutina para frenar el robot móvil.

Se compacta en una solo subrutina la lectura de las variables analógicas y digitales que intervienen en el ambiente de conducción para realizar promedio de varias muestras y obtener una medida más confiable.

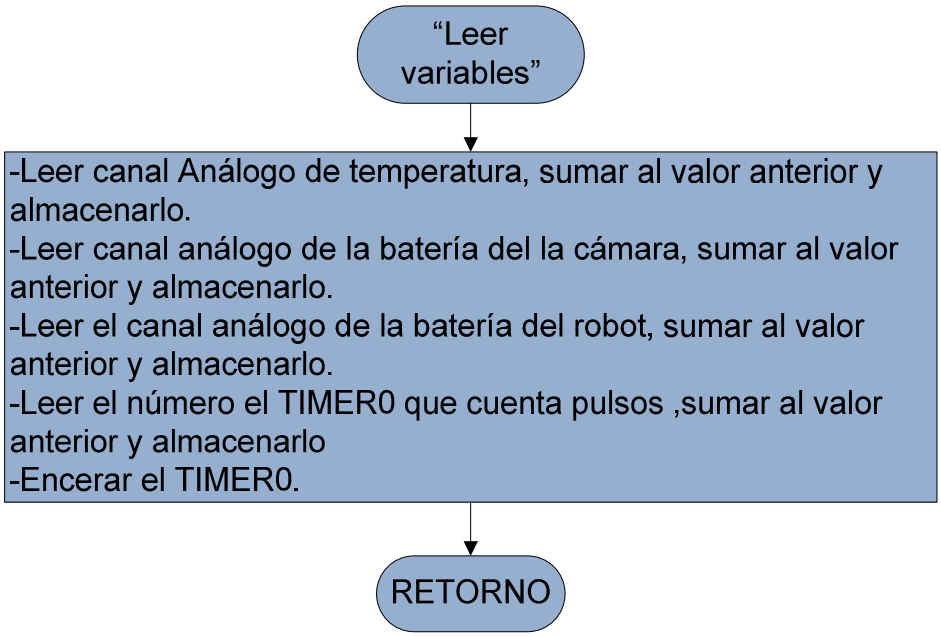


Fig. 3. 36 Diagrama de flujo de Subrutina para leer variables.

En el apartado 3.4.5 (MEDICIONES) se establecen las ecuaciones para acondicionar las respectivas variables que intervienen en el ambiente de conducción, este proceso se lo efectúa dentro del micro-controlador.

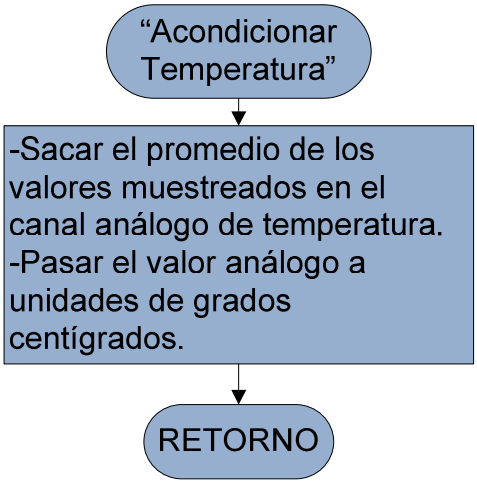


Fig. 3. 37 Diagrama de flujo de Subrutina para acondicionar temperatura.

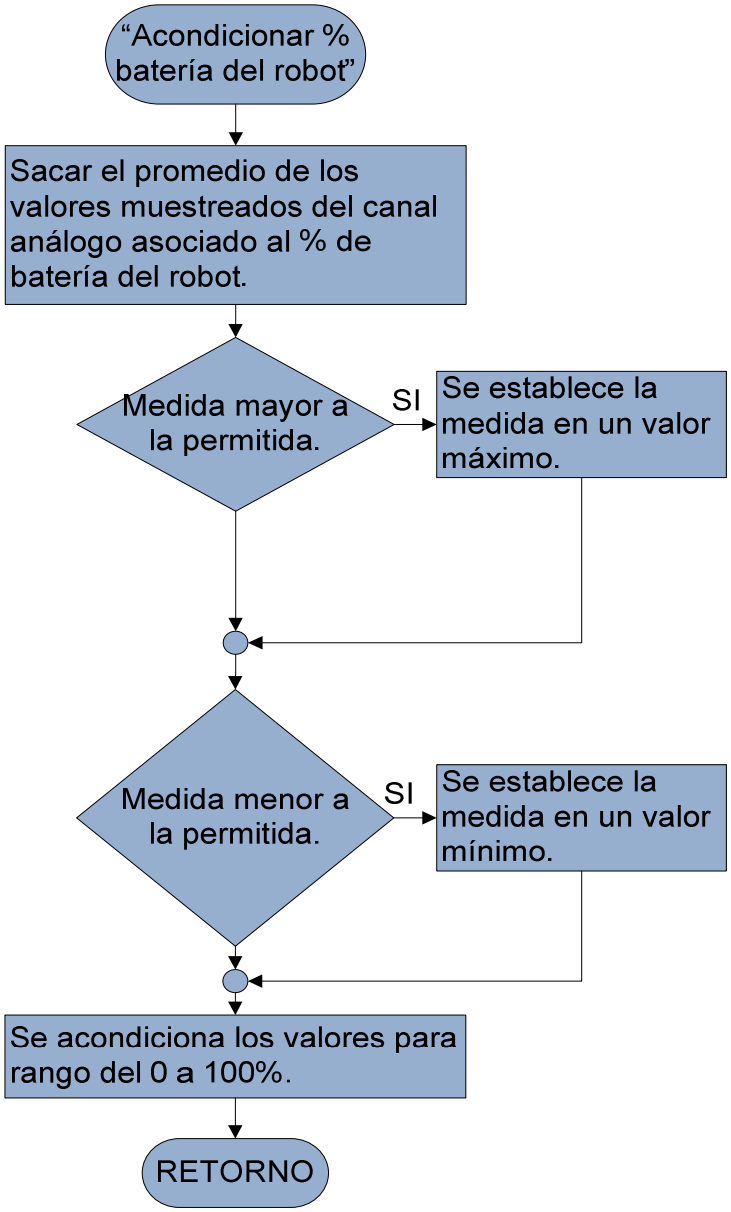


Fig. 3. 38 Diagrama de flujo de Subrutina para acondicionar % batería de robot.

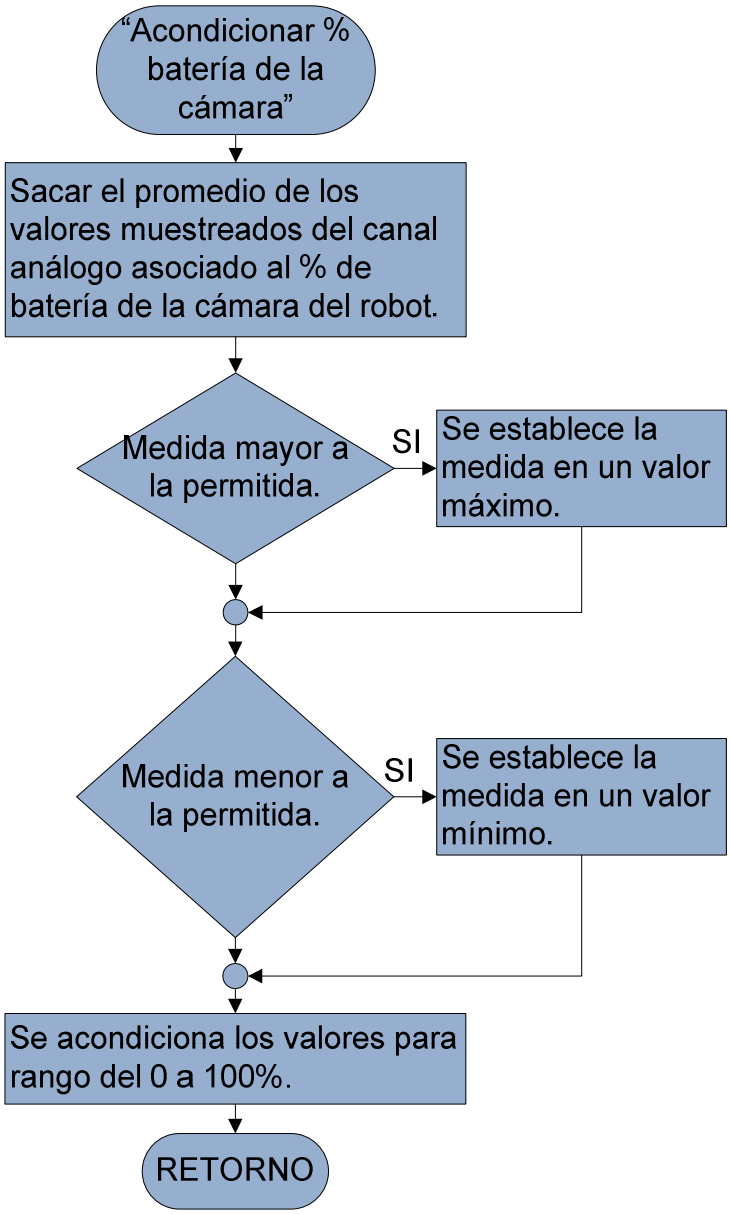


Fig. 3. 39 Diagrama de flujo de Subrutina para acondicionar % batería de cámara.

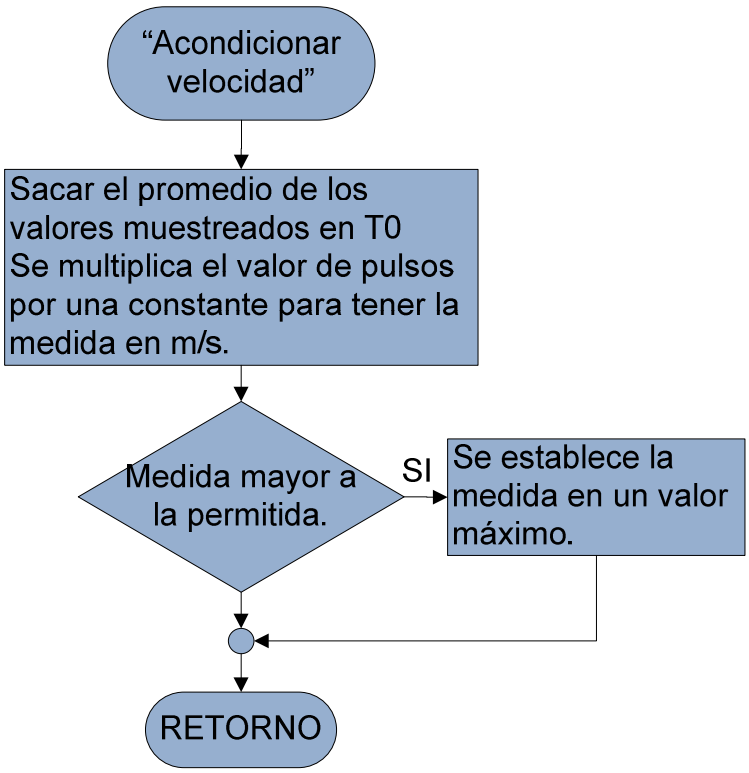


Fig. 3. 40 Diagrama de flujo de Subrutina para acondicionar velocidad.

CAPÍTULO 4

PRUEBAS Y RESULTADOS.

4.1 PUESTA EN MARCHA DE LA APLICACIÓN.

Para usar la aplicación desarrollada se deberá instalar y configurar todos los requerimientos de la aplicación según como detalla el ANEXO 1 “Manual de Usuario”. El control del sistema una vez iniciada la aplicación se efectúa solamente con movimientos de la cabeza. La interfaz de usuario indica el comportamiento del sistema, para acceder a la misma se ejecuta el acceso directo de menú inicio o escritorio creado tras instalar la aplicación.

Fig.4.1 Acceso Directo de la aplicación.

Se le solicita al usuario seleccionar el puerto serial correspondiente al robot móvil, luego que defina su posición neutral y que inicie el sistema. Una vez iniciado el sistema, la primera acción de la aplicación será reconocer y filtrar la posición inicial del usuario y tomarla como referencia para la interpretación de los movimientos.

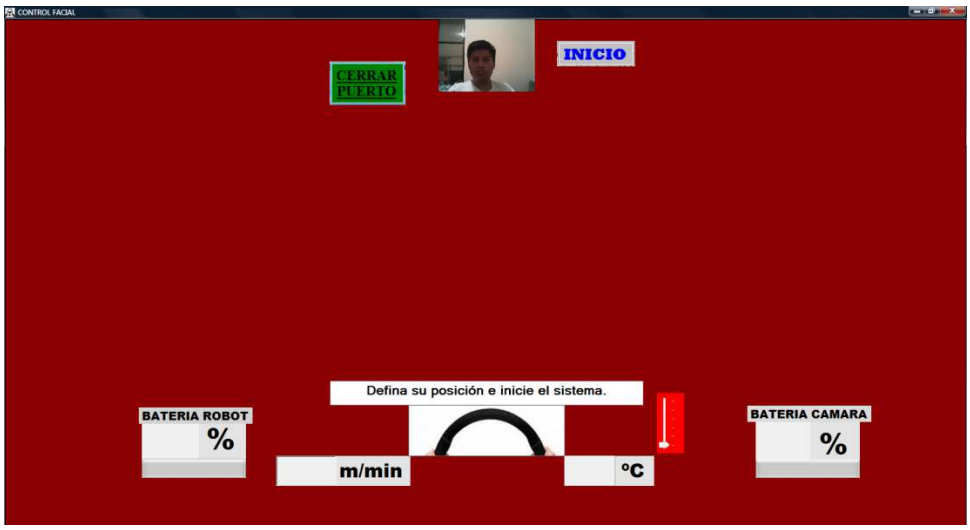


Fig.4.2 Fijando la posición inicial del usuario.

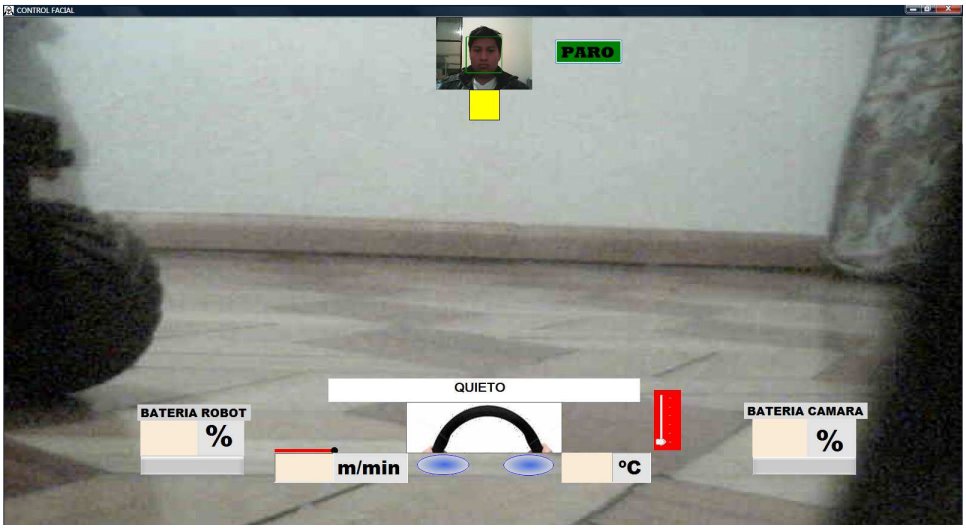


Fig.4.3 Filtrado de la posición inicial del usuario.

En la opción antes mencionada se le solicita al usuario permanecer quieto hasta que se le dé luz verde indicando que el sistema está listo para ser usado e interpretar los movimientos de la cabeza. Al ponerse en marcha el sistema se activa la interfaz de usuario en donde se tiene el ambiente de conducción. A partir de este momento los movimientos de la cabeza hacia la izquierda, derecha, arriba, abajo serán interpretados para poder conducir al robot móvil cuyo video, como puede observarse en la interfaz, se lo tiene como fondo de pantalla. En la siguiente sección se detalla el comportamiento del sistema ante los diferentes movimientos antes mencionados.

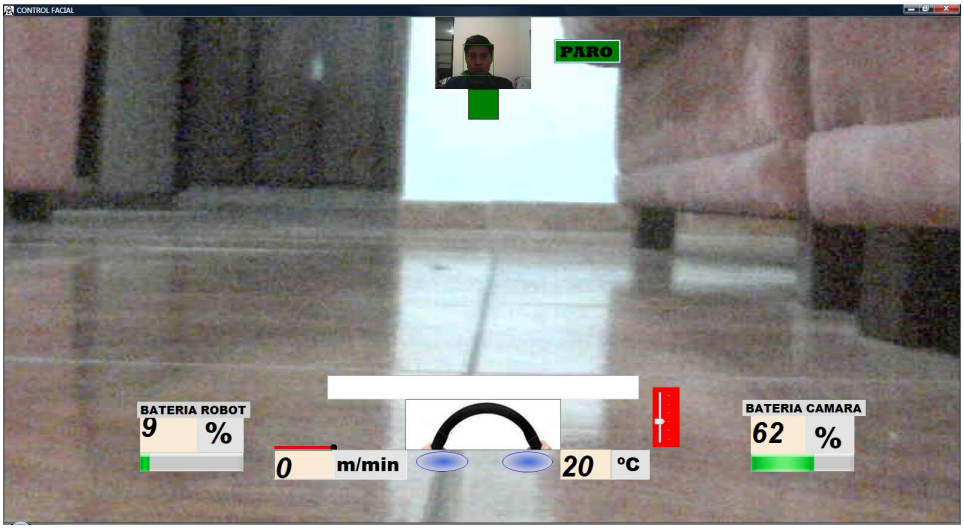


Fig.4.4 Sistema Listo.

4.2COMPORTAMIENTO DEL ROBOT.

El robot reacciona a las órdenes dadas desde la interfaz acorde a los movimientos de la cabeza del usuario. El método de manejo es en parte intuitivito, la forma de acelerar se la hace con un ligero movimiento de la cabeza sea abajo (acelerar hacia adelante) o arriba (acelerar hacia atrás). La interpretación de este movimiento dependerá de la dirección que tenga el robot. Cuando el robot está detenido o se inicia el sistema se tiene predefinido el movimiento del robot hacia adelante. Al regresar a la posición neutral el robot conserva la última velocidad fijada.

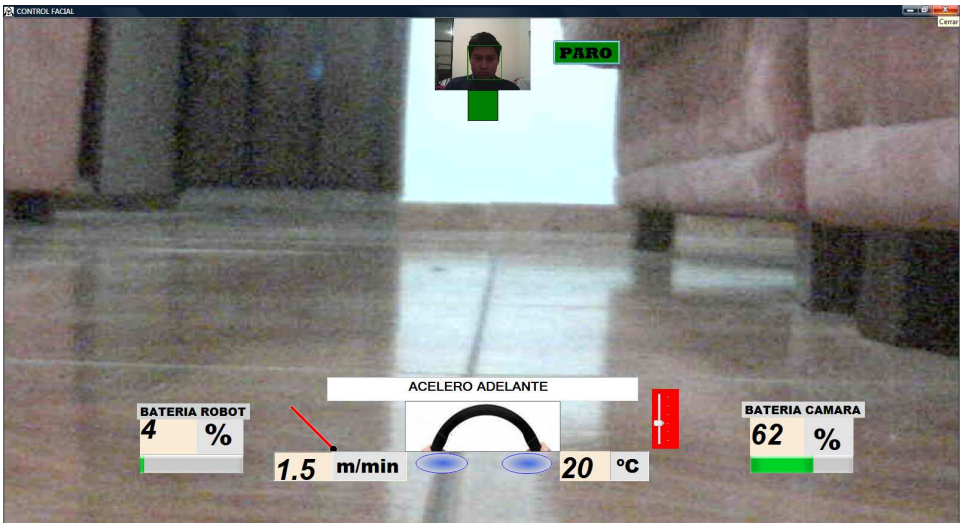


Fig.4.5 Robot Acelerando hacia Adelante.

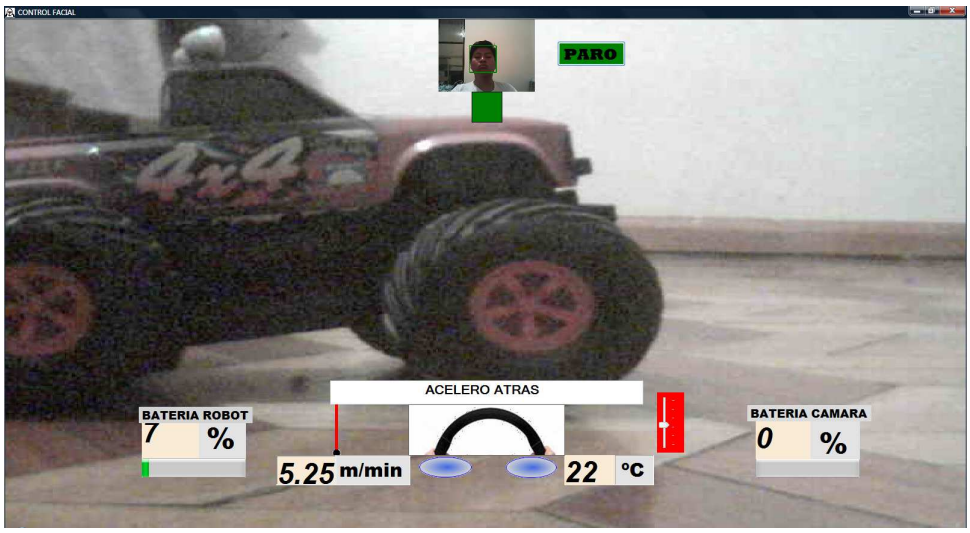


Fig.4.6 Robot Acelerando hacia atrás.

Dependiendo del sentido en el que circule el robot móvil, el mismo puede ser frenado en base a los mismos movimientos de cabeza arriba o abajo contrarios respectivamente a los realizados para acelerar. En las figuras Fig.4.5 a Fig.4.8 se muestra los movimientos de cabeza que detecta el sistema y la ayuda de texto de la interfaz que le indica al usuario la acción de control realizada.

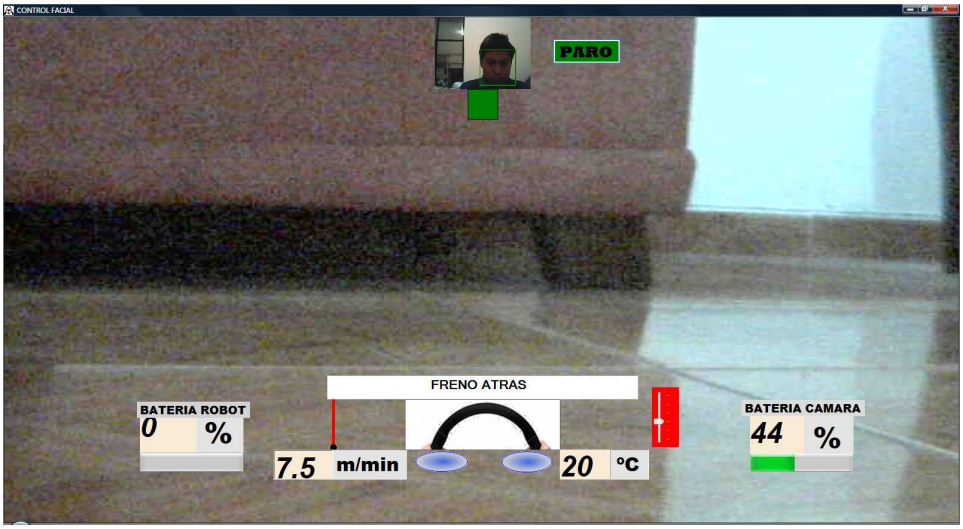


Fig.4.7 Robot Frenando cuando se mueve hacia atrás.

Fig.4.8 Robot Frenando cuando se mueve hacia adelante.

El giro del robot tanto a la izquierda como a la derecha se lo gobierna directamente con movimientos de la cabeza a la izquierda o derecha

respectivamente. La interfaz de usuario ante dichos eventos se comporta como detallan las Figuras Fig.4.9 y Fig.4.10.

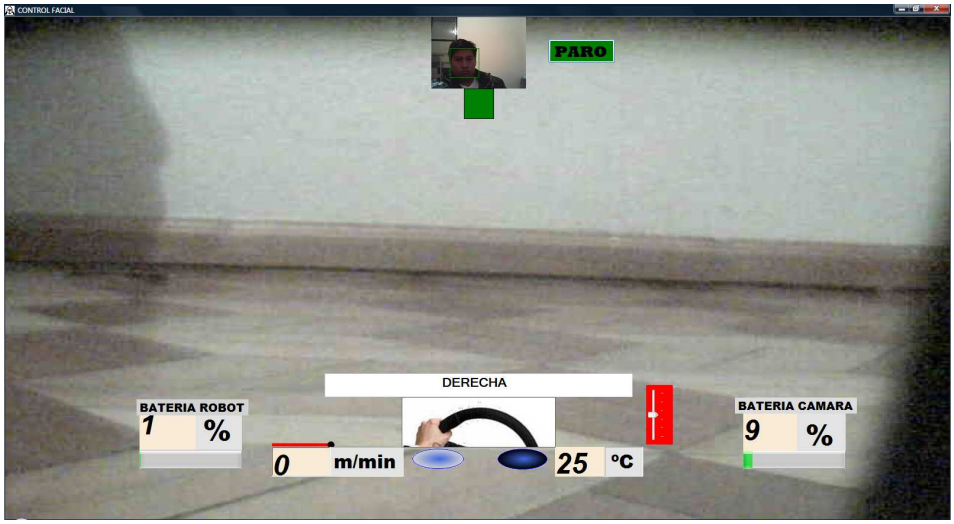


Fig.4.9 Movimiento a la derecha.

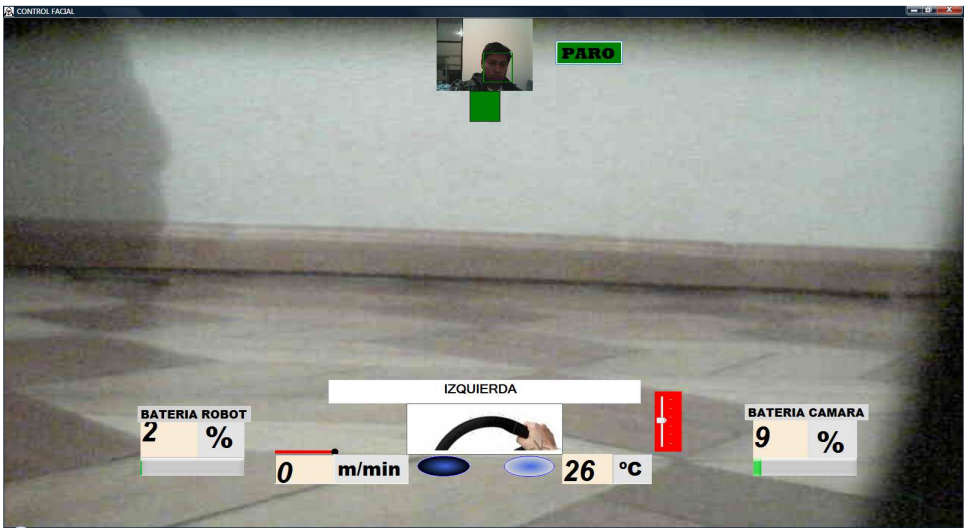


Fig.4.10 Movimiento a la izquierda

El alcance de este proyecto detalla interpretar los movimientos de la cabeza antes mencionados, así como combinados entre sí, es decir que se pueda interpretar un movimiento de la cabeza hacia arriba y a la izquierda como se muestra en la figura Fig.4.11 en donde se quiere acelerar y girar a la izquierda para esquivar un obstáculo, constituyendo esta una “orden combinada”.

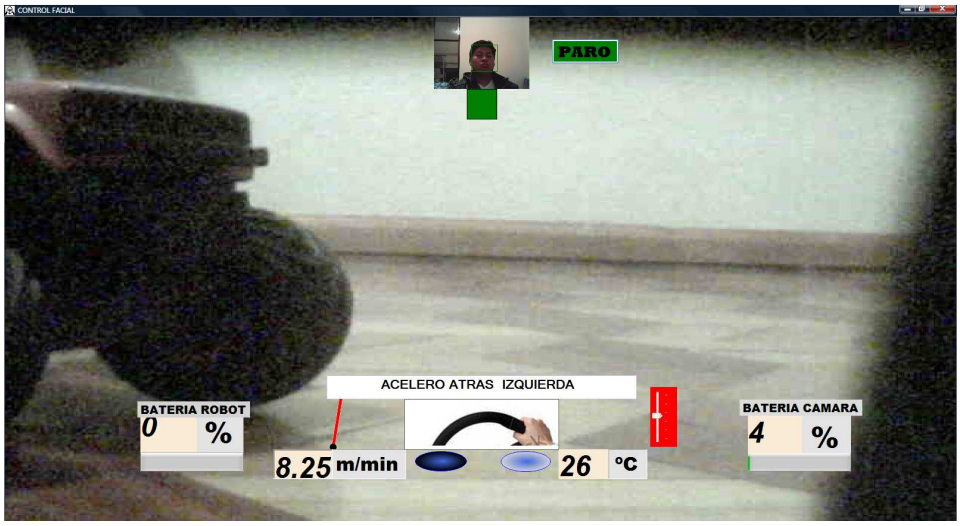


Fig.4.11 Orden combinada.

Otro requerimiento del sistema es tener alertas visuales que le permitan al usuario conocer de peligros que pudieren poner al sistema en riesgo. El robot está dotado de un par de sensores ultrasónicos con el fin de detectar obstáculos adelante o atrás del robot. El caso de que se detecte un obstáculo en el trayecto del robot, el mismo se detendrá automáticamente por seguridad, además no le permitirá acelerar hacia el obstáculo hasta que este desaparezca o se encuentre lo suficientemente lejos de modo que el robot se encuentre fuera de peligro. En la figuras Fig.4.12 y Fig.4.13 se pude observar la interfaz cuando el robot ha encontrado un obstáculo y se ha detenido.

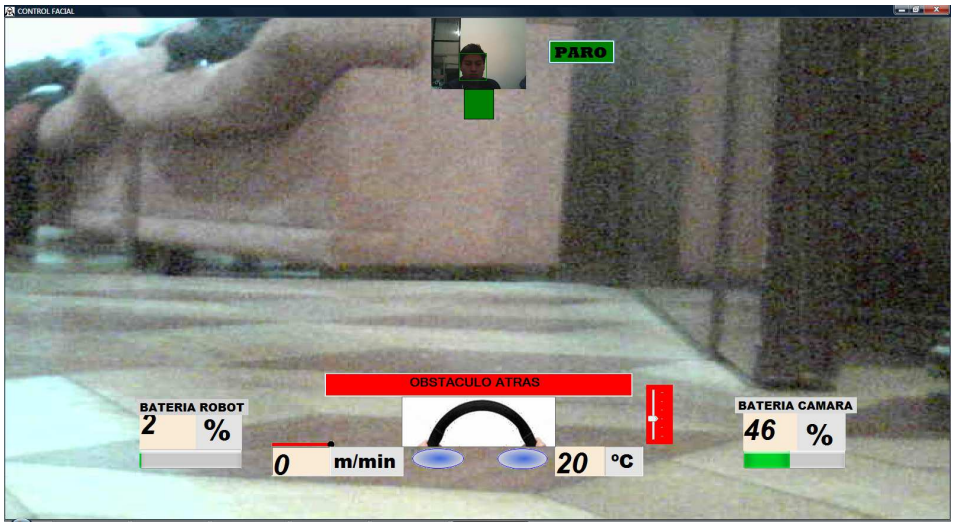


Fig.4.12 Detección de Obstáculo Atrás del robot.

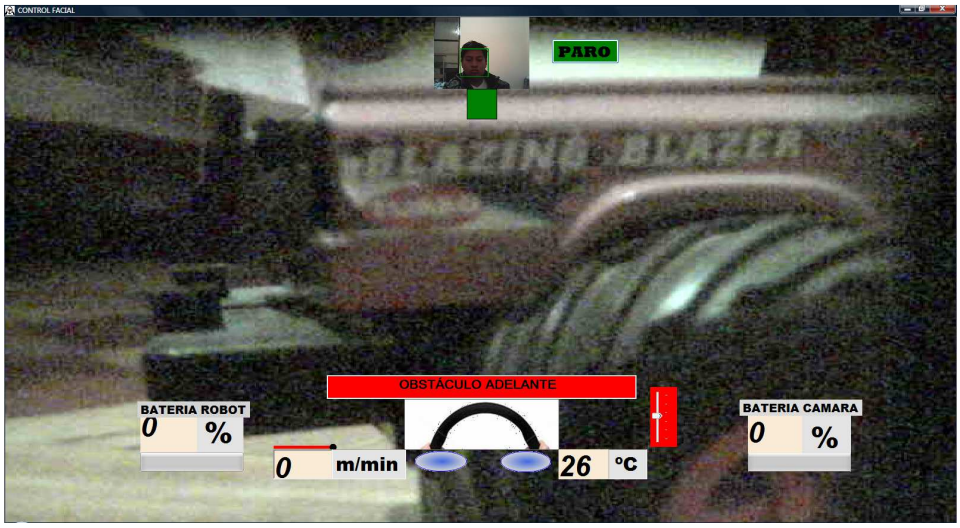


Fig.4.13 Detección de Obstáculo Adelante del robot.

Otro evento que se puede dar es la pérdida de la conexión entre el robot y la interfaz, en este caso se libera todos los recursos usados en la aplicación para la comunicación y se cierra el puerto serial por seguridad. Además el robot se detendrá de ser necesario. La interfaz pasa a modo inicial y le indica que se ha perdido la conexión por lo que el usuario debería verificar el porqué de la perdida de conexión, si el robot se ha apagado o alejado mucho, para restablecer la conexión, abrir el puerto de comunicaciones nuevamente e iniciar el sistema. En la Fig.4.14 se muestra el comportamiento de la interfaz al perderse la conexión.

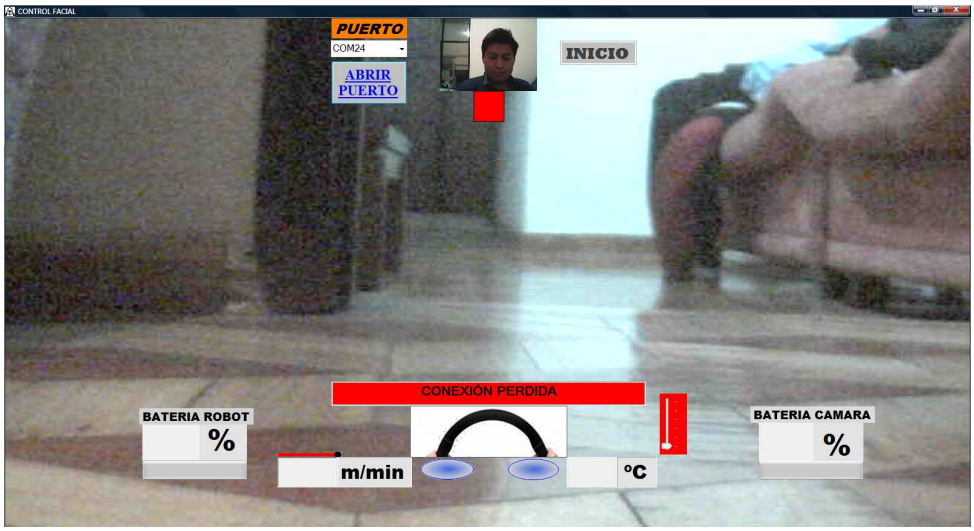


Fig.4.14 Interfaz cuando se ha perdido conexión con el robot.

Para intercambiar de usuario de la aplicación será necesario detener el sistema, en donde el mismo vuelve a sus condiciones iniciales. La apariencia de la interfaz cuando el sistema ha sido detenido y se puede cambiar de usuario o cerrar la aplicación se la expone en la figura Fig.4.15.

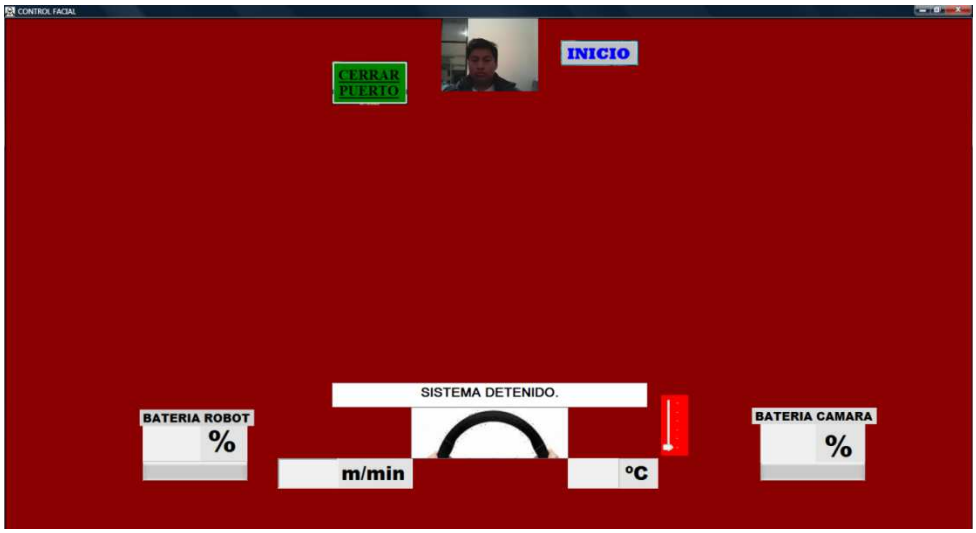


Fig.4.15 Paro del Sistema.

4.3 RESULTADOS DE ENCUESTA A DISTINTOS USUARIOS.

Con el fin de evaluar el sistema al ser usado por distintos usuarios se ha realizado una encuesta a un total de 30 personas en donde 50% son hombre y un 50% mujeres comprendidos en edades de 18 a 26 años. Los resultados a continuación.

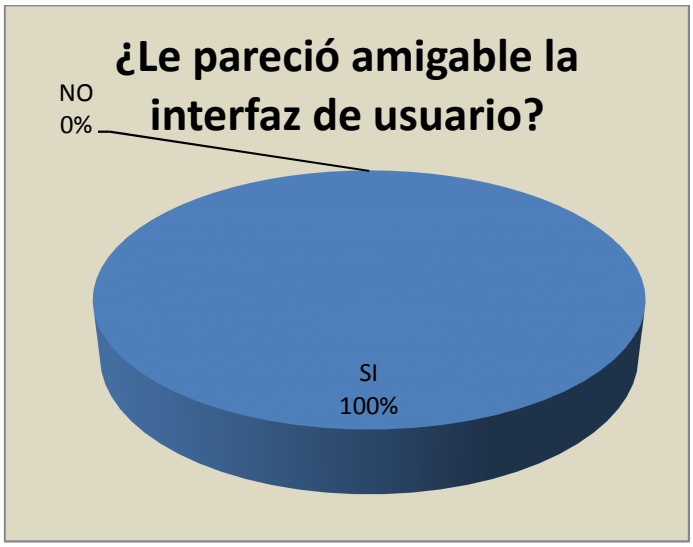


Fig.4.16 Diagrama de respuestas a Pregunta 1.

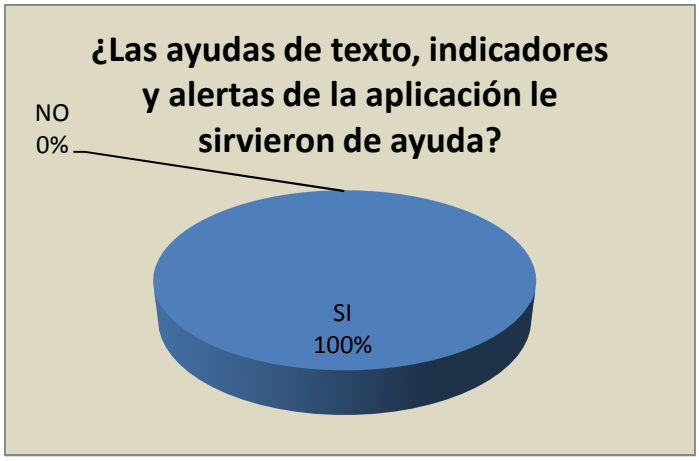


Fig.4.17 Diagrama de respuestas a Pregunta 2.

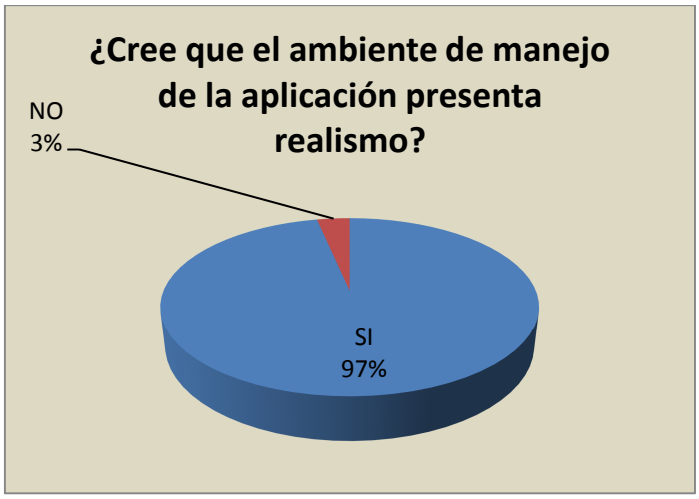


Fig.4.18 Diagrama de respuestas a Pregunta 3.

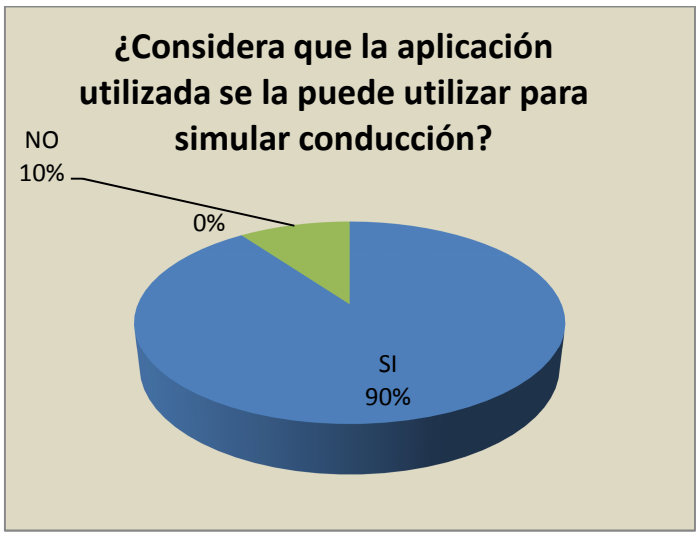


Fig.4.19 Diagrama de respuestas a Pregunta 4.

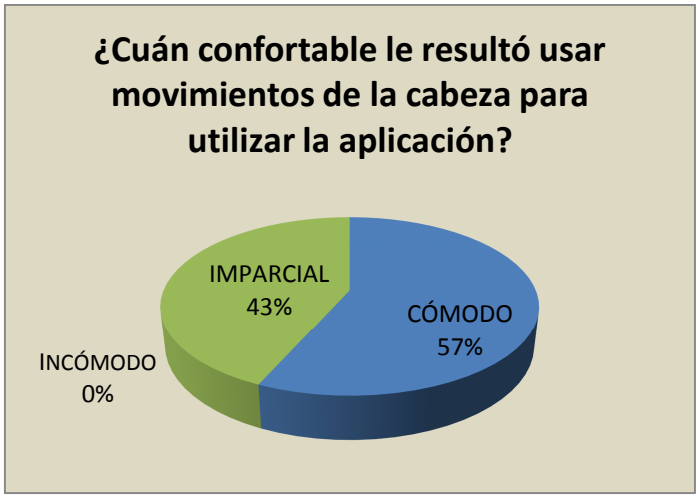


Fig.4.20 Diagrama de respuestas a Pregunta 5.

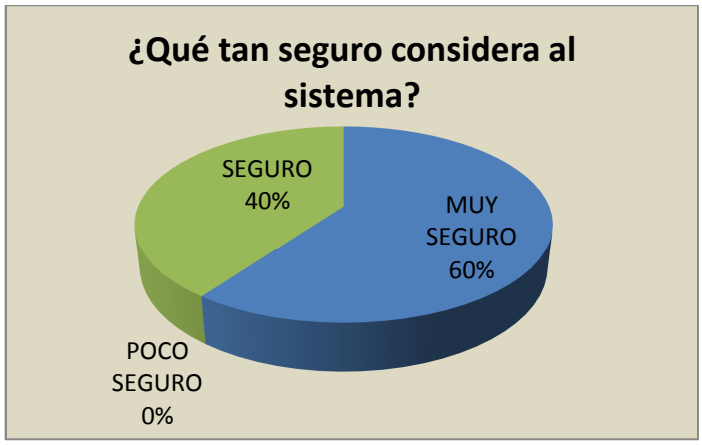


Fig.4.21 Diagrama de respuestas a Pregunta 6.

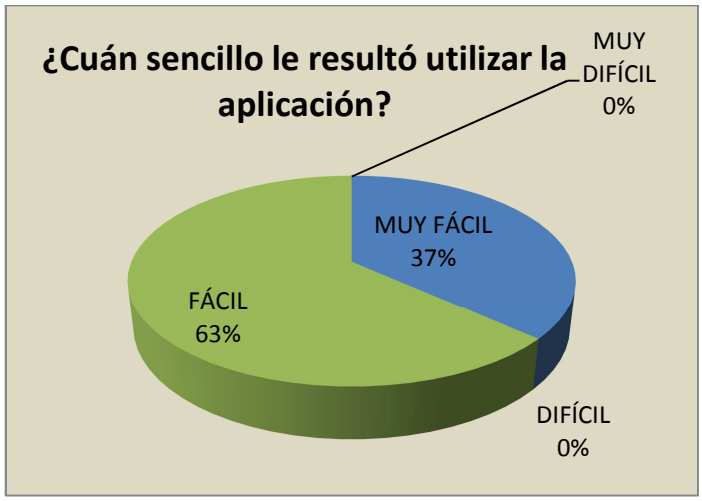


Fig.4.22 Diagrama de respuestas a Pregunta 7.

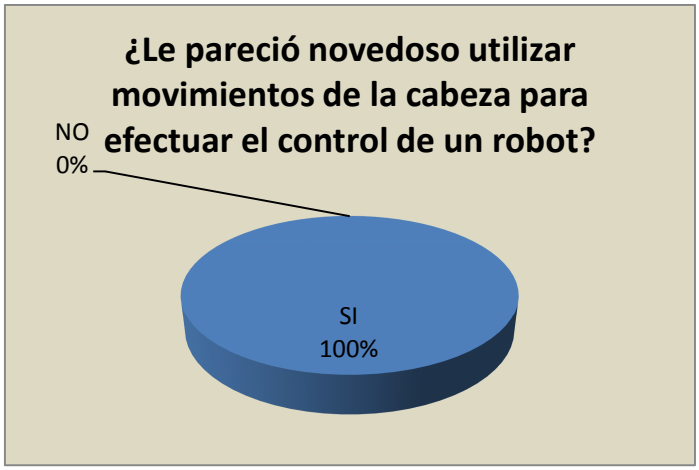


Fig.4.123Diagrama de respuestas a Pregunta 8.

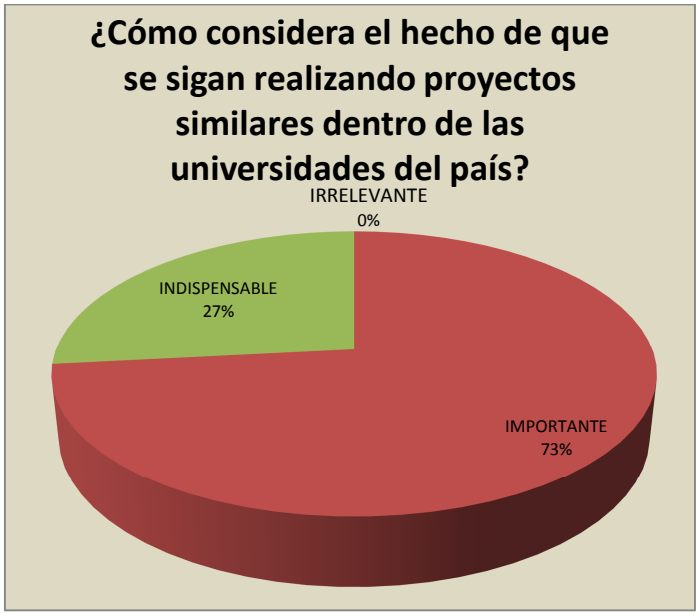


Fig.4.24Diagrama de respuestas a Pregunta 9.

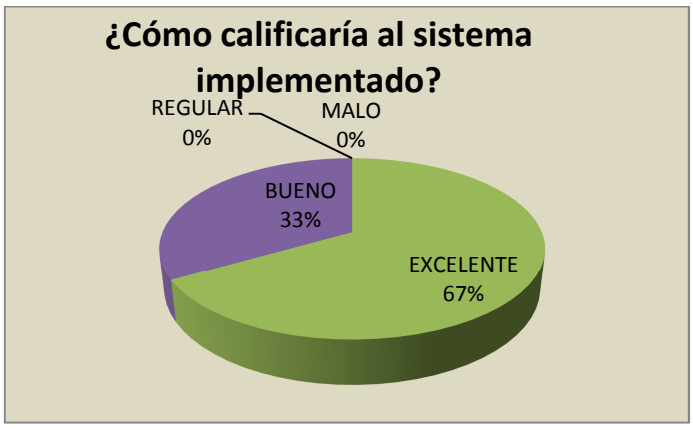


Fig.4.25 Diagrama de respuestas a Pregunta 10.

En virtud de las diversas pruebas realizadas en el sistema se tiene como resultado general un robot controlado mediante movimientos de la cabeza del usuario arriba, abajo, izquierda, derecha y combinados, el mismo dotado de seguridades para prevención de choques y pérdida de conexión. Además el control del robot en cuanto a velocidad se lo hace mediante una rampa de aceleración con lo que se evita aceleraciones bruscas. Todo el control y captura de variables y video del entorno del robot se transmiten de forma inalámbrica desde y hacia una interfaz con la que interactúa el usuario.

Dicha interfaz emula un ambiente de manejo indicándole al usuario la velocidad y temperatura del robot además del estado de sus baterías, y le da la impresión de efectuar el manejo como en un video juego. Al usar la aplicación se identifica automáticamente el movimiento de cabeza del usuario en forma rápida y precisa con la ayuda de una cámara web, la detección del rostro se la puede también visualizar en la pantalla principal de la aplicación dándole al usuario una realimentación visual de lo que se encuentra haciendo.

Los indicadores visuales, texto y gráficos le ayudan al usuario a usar la aplicación en forma más intuitiva y fácil, lo que los mismos lo corroboran en las encuestas realizadas.

En definitiva se puede decir que se ha realizado un proyecto interesante que ha llamado la atención del usuario y puede aplicarse en diversas áreas como en el manejo mismo o hasta en la industria, además con ligeras adecuaciones puede ayudar a personas discapacitadas a realizar actividades o ejecutar tareas en forma automática a parte del manejo del robot del sistema.

4.4 COSTO DEL PROYECTO

El presente proyecto ha sido desarrollada de modo que se cumpla con los objetivos planteados y haciéndolo lo más económico posible. A continuación se detalla los costos totales del proyecto en dólares americanos.

DESCRIPCIÓN	CANTIDAD	COSTO UNITARIO	COSTO TOTAL
Cámara IP	1	80.00	80.00
Webcam	1	30.00	30.00
Router Wireless	1	30.00	30.00
Módulo Bluetooth	1	20.00	20.00
Servo Motor	1	17.00	17.00
Sensor Ultrasónico	2	8.00	16.00
Adaptador USB-Bluetooth	1	8.00	8.00
Batería 12 [V]	1	15.00	15.00
Batería 6[V]	1	9.00	9.00
Atmega 164P	1	6.50	6.50
Attiny 13	2	2.50	5.00
LM35	1	2.00	2.00
LM7805	4	0.5	2.00
L298	1	4.00	4.00
Porta fusible	1	0.5	0.50
Disipador Pequeño	1	0.5	0.50
Disipador Grande	1	1.00	1.00
Fibra para placas 20*40	1	4.00	4.00
Elementos Electrónicos varios	-	10.00	10.00
Suelda, Conectores, Cableado y Chasis	-	30.00	30.00
Motor DC	1	5.00	5.00
Base Robot	1	5.00	5.00
Hora de ingeniería	300	20.00	6000.00
COSTO TOTAL DEL PROYECTO			6300.50

CAPÍTULO 5.

CONCLUSIONES Y RECOMENDACIONES.

5.1CONCLUSIONES

En el sistema hombre-máquina implementado, los movimientos de la cabeza del usuario hacen posible el control del mismo en tiempo real, por lo cual la optimización del código es parte vital para que el software desarrollado cumpla las acciones pertinentes en el momento adecuado, sin malgastar recursos ni afectar el comportamiento del sistema.

Al desarrollar aplicaciones en código abierto la principal ventaja está en que, al tener acceso al código fuente se puede interactuar entre distintos desarrolladores de software que trabajan en proyectos similares y desarrollar aplicaciones cada vez más interesantes; hecho que permite una irreductible evolución en el campo de la tecnología.

En un sistema donde el ser humano forma parte del lazo de control, el comportamiento de dicho sistema puede verse alterado entre distintos usuarios; razón por la cual en un sistema como tal, se debe garantizar el confort del usuario en general, haciendo imperceptibles las variaciones en el resultado final, cuando se intercambia usuarios. La similitud en el patrón morfológico que rigen a los rostros humanos hace imperceptible dicha variación de usuarios en el sistema implementado.

Al implementar un sistema tele-operado, el usuario por lo general no se encuentra en la cercanía del o los dispositivos a controlar como es el caso del robot implementado. La seguridad con la que se dote al sistema a controlar debe ser tal que requiera lo mínimo del operador en caso de peligro, tomándose automáticamente las acciones preventivas pertinentes.

La transmisión inalámbrica de video genera retardos al conocer la ubicación del entorno capturado por la cámara; para el caso específico, el entorno del robot. La

selección correcta del equipamiento para captura y transmisión de video, ha permitido minimizar el retardo en conocer el entorno de tal forma que el usuario perciba el comportamiento del robot en tiempo real.

5.2RECOMENDACIONES

Se recomienda extender el uso de procesamiento de imagen y video en tiempo real al campo del control de procesos, pues esta tecnología se torna cada vez más útil y eficaz con la mejora en unidades de procesamiento de imágenes.

En la implementación de todo sistema hombre-máquina es necesario enfocarse en el posible comportamiento del usuario, por excéntrico que pudiere parecer, pues el comportamiento que puede adoptar el ser humano es impredecible, es por ello que se debe dotar al sistema de seguridades a tal punto que se minimice el impacto debido a comportamiento inusual del usuario.

Se recomienda utilizar una cámara igual o similar a la seleccionada en caso de desarrollar una aplicación similar, o usar un procesador con cámara integrada, pues el retardo entre captura y procesamiento de información es vital y puede verse afectada por la interfaz usada en comunicación. Por lo que en el presente proyecto se ha utilizado una cámara que se comunica a través un puerto USB de alta velocidad.

Se recomienda en la construcción de un prototipo, hacerlo de forma modular, pues efectuarlo de tal modo ayuda tanto en implementación, pruebas, como en mantenimiento.

En el desarrollo de un software se recomienda hacerlo en forma legible y ordenada, separando funciones métodos o clase según sea el caso, acorde al papel que juegue dentro de la aplicación. De esta manera se facilitará futuros cambios, extensiones y mejoras en el futuro.

Para un prototipo en desarrollo se recomienda dotarlo de un buen tiempo de autonomía de energía y del que se pueda tener conocimiento de su estado. Sea el caso del sistema implementado cuenta con realimentación visual del estado de las baterías. Este particular ha ayudado a efectuar un considerable número de pruebas para mejorar el sistema en general.

Se recomienda el uso de las librerías de OpenCV y EmguCV, para aplicaciones no solo similares, sino a distintas áreas dentro de la visión computarizada, pues dichas librerías cuentan con herramientas tan poderosas que pudieren permitir la consecución de un gran número de aplicaciones y ejecutar ideas que en un principio parecerían inalcanzables.

REFERENCIAS BIBLIOGRÁFICAS

[1] VIT, “Driver monitoring tool detects driver workload and stress levels”, disponible en: http://www.vtt.fi/references/driver_monitoring_tool.jsp?lang=en

[2] Wikipedia “Pixel”, The free encyclopedia, disponible en: http://en.wikipedia.org/w/index.php?title=Pixel.

[3] A. Aceves, “Segmentación de colores con Matlab” Seminario del Proyecto de Investigación en Robótica Humanoide.

[4] The MATHWORKS Inc., Image *Processing Toolbox™ User’s Guide,* R2013a.

[5] Wikipedia “Modelo de color HSV”, La enciclopedia libre, disponible en: http://es.wikipedia.org/w/index.php?title=Modelo_de_color_HSV.

[6] K. Hounslow “Tutorial Real Time Tracking Using OpenCV”, video tutorial, 2013.

[7] B. Escalante “Procesamiento Digital de Imágenes”, Apuntes del curso impartido, 2006.

[8] W. Pratt, “Digital Image Procesing”, A Wiley-Interscience Publication JONH WILEY & SONS, INC. New York 2001.

[9] “Introducción al Procesamiento Digital de Imágenes” Disponible en; www.vinuesa.com.

[10] P. Belhumeur, J.Hespanha, and D.Kriegman “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no.7, pp. 711-720. Julio 1997.

[11] M. Turk, A. Pentland “Eigenfaces for Recognition” Vision and Modeling Group, the Media Laboratory, Massachusetts Institute of Technology, 1991.

[12] S. Kaymak “Face Detection, Recognition and Reconstruction using Eigenfaces”, EE574 Detection & Estimation Theory Class Presentation, Department of Electrical and Electronic Engineering, 2003.

[13] P. Viola, M. Jones, “Rapid Object Detection Using a Boosted Cascade of Simple Features”, accepted Conference on Computer Vision and Pattern Recognition, 2001.

[14] The OpenCV Reference Manual Release 2.4.5.0, Abril 05, 2013.

[15] M. Yang “Face detection” University of California.

[16] P. Wilson, J. Fernandez “Facial Feature detection Using Haar Classifiers”, 2006.

[17] E Nuño, L. Basañez “Tele operación: técnicas, aplicaciones, entorno sensorial y tele-operación inteligente”, Universidad Politécnica de Catalunya, Abril 2004

[18] “La Teleoperación”, disponible en: http://www.tecnun.es/asignaturas/control1/proyectos/teleop2D/teleoperacion.htm.

[19] R. Chellali, “Tele-operation and Human Robots Interactions”, Human Robots Medited Interactions Lab. Tele-robotics and Applications Dept, Italian Institute of Technology.

[20] Imagen disponible en: http://t0.gstatic.com/images?q=tbn:ANd9GcQP2mKuU_M7gnl7Y1w2L1R0gFr3HuzRDV-p1X6Kgav11vNTTeTC.

[21] J. Crandall, M. Goodrich “Characterizing Efficiency of Human Robot Interaction: A Case Study of Shared-Control Teleoperation”, Computer Science Department, Brigham Young University.

[22] G. Johannsen, “Human-machine Interaction”, Control Systems, Robotics, and Automation –Vol. XXI.

[23] Imagen disponible en: http://www.esafetysupport.org/en/esafety_activities/esafety_working_groups/human-machine_interaction_hmi_.htm.

[24] Wikipedia “Ergonomía”, la enciclopedia libre, disponible en: http://es.wikipedia.org/wiki/Ergonom%C3%ADa.

[25] E. Moret, “Dynamic Modeling and Control of a Car-Like Robot”, Blacksburg, Virginia, February 5, 2003.

[26] M. Egerstedt, X Hu, A Stotsky, “Control of a car-like Using a Virtual Vehicle Approach”, Royal Institute of Tecnology, Sweden.

[27] A. De Luca, G. Oriolo , C. Samson, ” Feedback Control of a Nonholonomic Car-Like Robot”, Universita di Roma “La Sapienza”.

[28] The MATHWORKS Inc., Computer Vision System Toolbox™ User’s Guide, R2013a.

[29] The MATHWORKS Inc., Image Processing Toolbox™ User's Guide, R2013a.

[30] GONZÁLEZ, Raúl Duque. Python Para Todos, Creative Commons, España.

[31] "OpenCV" página oficial; http://opencv.org

[32] The OpenCV Tutorials, Release 2.4.5.0, Abril 05, 2013.

[33] The OpenCV Reference Manual, Release 2.4.5.0, Abril 05, 2013.

[34] "EmguCV" página oficial; http://www.emgu.com

[35] Wikipedia "Qt", la enciclopedia libre, disponible en: http://en.wikipedia.org/wiki/Qt_(framework)

[36] "OpenCV tutorial 5: Emgu CV with C#" video disponible en: http://www.youtube.com/watch?v=vdjoutNR2DQ

[37] Lm35 datasheet, Precision Centigrade Temperature Sensors, National Semiconductor.

[38] Ultrasonic Ranging Module HC - SR04, Elec Freaks.

[39] http://airlink101.com/products/aic250w.php

[40] http://www.logitech.com/es-es/product/webcam-c170#section=specs

[41] ATMEGA164P, 8-BIT AVR Microcontroler "Manual" Atmel.

[42] ATTINY13, 8-BIT AVR Microcontroler "Manual" Atmel.

[43] HC Serial Bluetooth Products, User Instructional Manual.

[44] L298, Dual Full Bridge. ST "datasheet".

ANEXOS

ANEXO1

MANUAL DE USUARIO

INDICE

MANUAL DE USUARIO

Antes de utilizar el sistema de tele-operación para el robot móvil implementado se deberá seguir los pasos y sugerencias detalladas en este documento. Primero se deberá configurar el ordenador donde se instalará la aplicación desarrollada y las herramientas necesarias. Para poder usar el sistema se deberá tener en el ordenador los siguientes requerimientos.

-Instaladas las librerías de “EmguCV”
-Acceso a la cámara del robot.
-Sincronización de la PC con el Modulo Bluetooth.
-Instalado la aplicación “Control Facial”
Los instaladores respectivos se entregan como anexo del proyecto. Para facilidad del usuario de detalla la manera de instalar el software requerido.

Instalación de EMGUCV.

Se debe instalar EmguCV, se accede al ejecutable “libemgucv-windows-universal-gpu-2.4.9.1847” y se siguen los pasos detallados en los gráficos del Fig. A1.1 - Fig. A1.7.

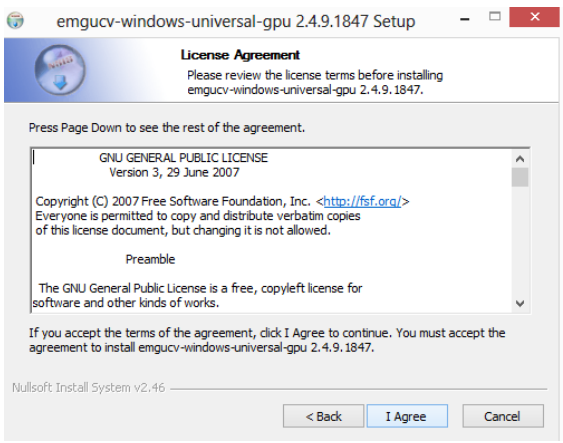


Fig. A1.1

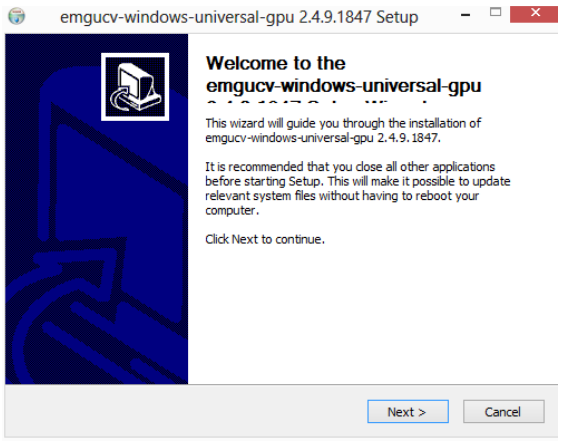


Fig. A1.2

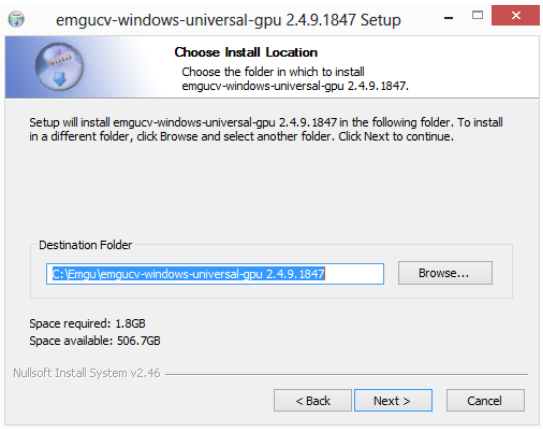


Fig. A1.3

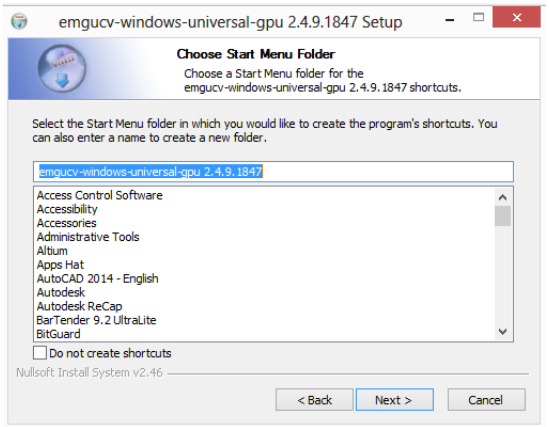


Fig. A1.4

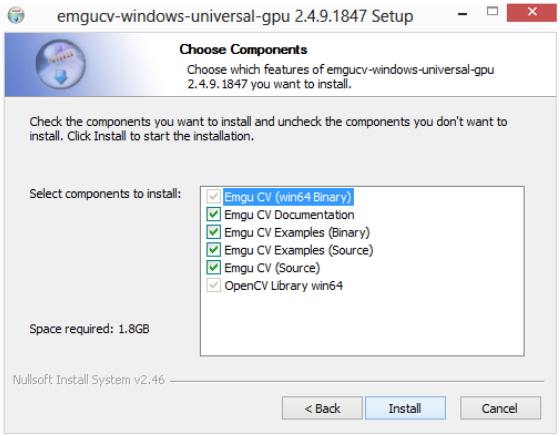


Fig. A1.5

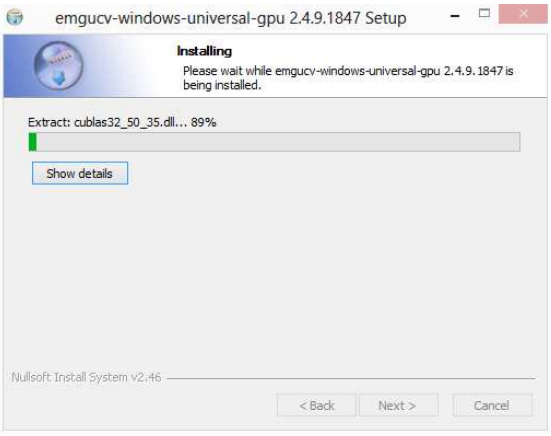


Fig. A1.6

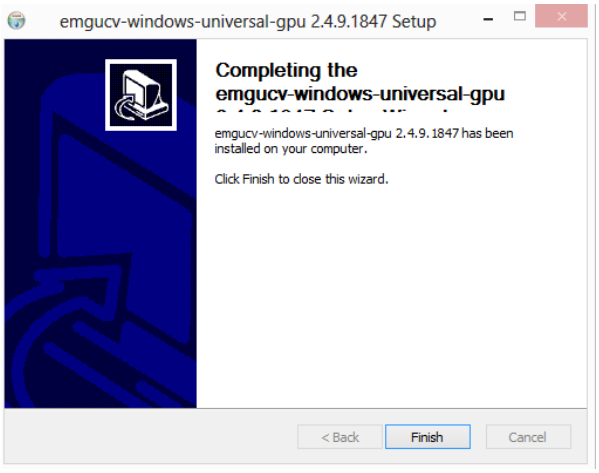


Fig. A1.7

Una vez instalado EmguCV en el Sistema, se debe fijar la variable de entorno para ello se debe acceder al directorio de instalación detallado en la Fig. A1.3 y copiar la dirección de la ruta de instalación.

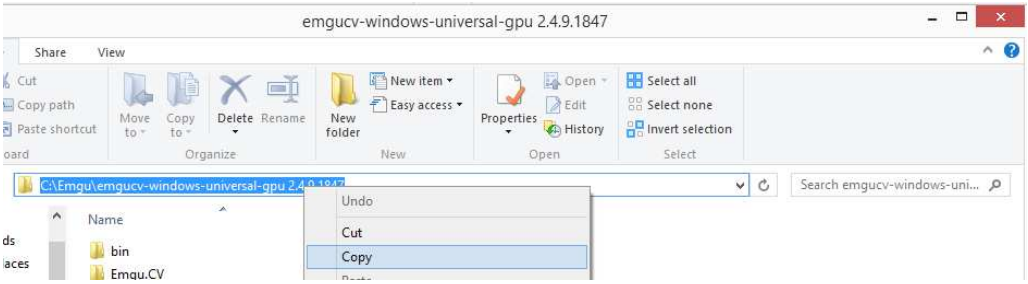


Fig. A1.8

Se ingresa al panel de control y accede al Editor de Variables de sistema siguiendo los pasos detallados en las figuras Fig.A1.9 - Fig.A1.14 para acceder a la configuración avanzada del sistema para fijar la variable de entorno.

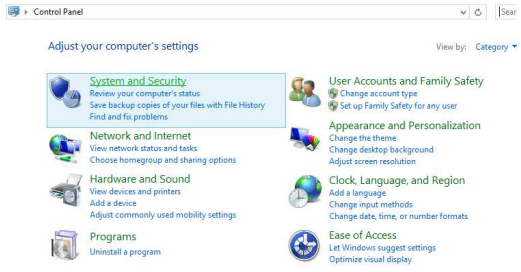


Fig.A1.9

Fig.A1.10

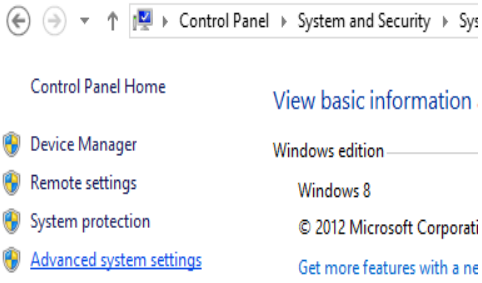


Fig.A1.11

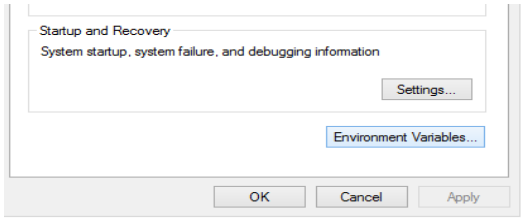


Fig.A1.12

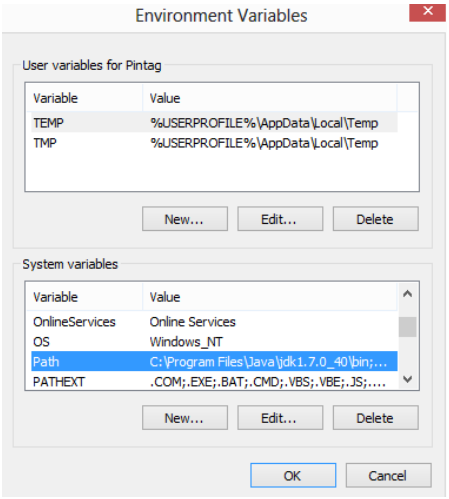


Fig. A1.13

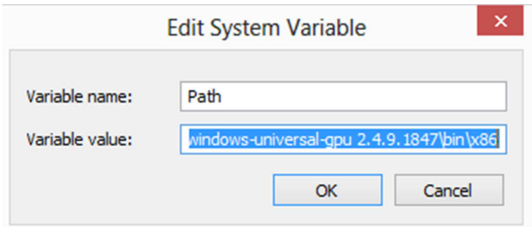


Fig. A1.14

Acceso a la Cámara Del Robot.

Sea para la configuración inicial de la cámara o para el acceso a través de un Router inalámbrico se deberá configurar conexión de área local a la que se vincule el Router que sirve de canal de comunicación entre cámara e Interfaz. Para ello desde el Panel de Control se acede según los pasos indicados en las figuras Fig. A1.15- Fig. A1.18.

Fig. A1.15

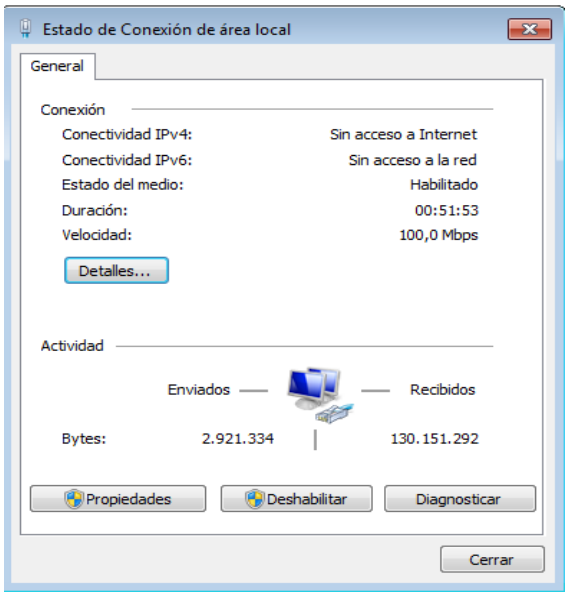


Fig. A1.16

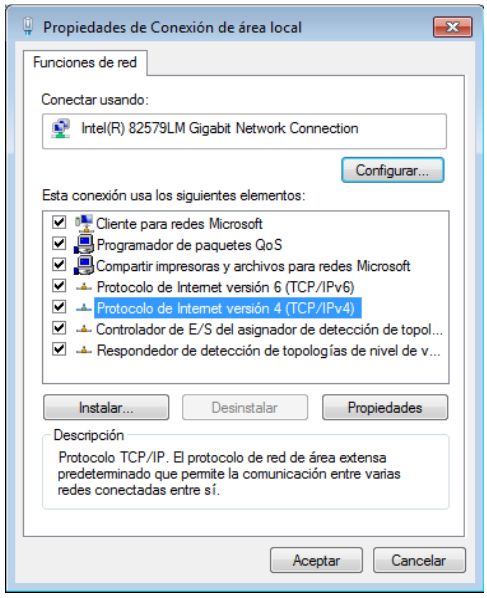


Fig. A1.17

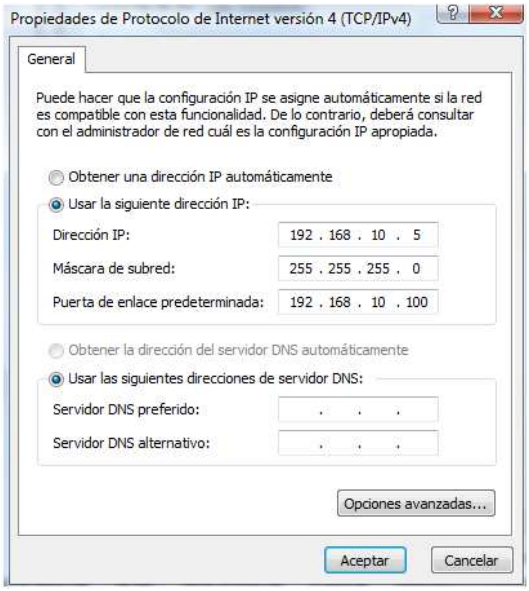


Fig. A1.18

En esta última ventana se asigna una dirección IP estática deseada al computador donde se conecte la cámara o si se lo hace directamente con un Router se puede usar dirección IP dinámica. En caso de requerir utilizar un Router distinto al seleccionado para el sistema implementado se deberá reconfigurar la cámara según la red a la que se integre.

El acceso a la cámara se efectúa a través de IE o la aplicación “IPViewPro” incluida donde se detecta la IP de todas las cámaras IP conectadas a la red en uso, se debe añadir la cámara del ROBOT configurada con el ID “ROBOT” y dirección IP “192.168.10.10”.

En caso de no poder acceder al video desde IE se debe instalar el Active X de la cámara IP para ello a través de Internet Explorer se debe configurar en Herramientas/ Opciones de Internet siguiendo los pasos detallados en las figuras Fig. A1.19 - Fig. A1.21.

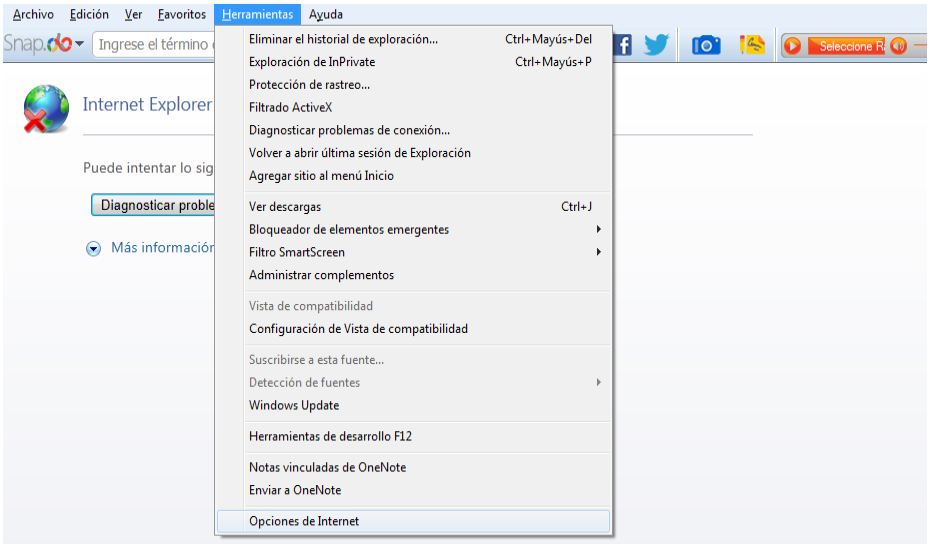


Fig. A1.19

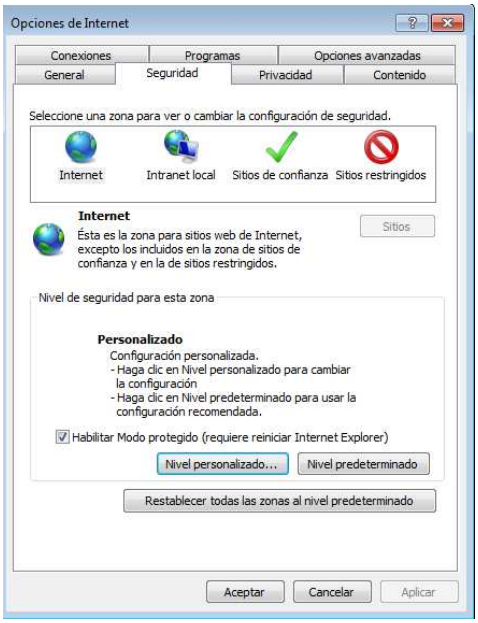


Fig. A1.20

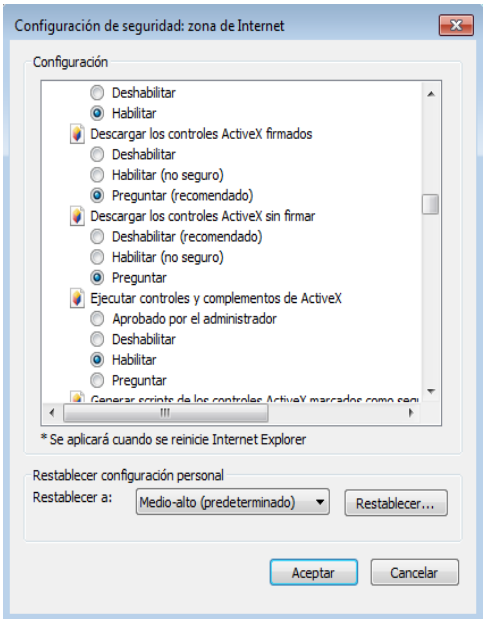


Fig. A1.21

Para usar la aplicación se debe tener encendida la cámara y acceder a la misma a través de “IPViewPro”, propia de la cámara Air link Se debe agregar la cámara del robot ya configurada con un dirección fija, para lo cual abrimos la aplicación mencionada.

Fig. A1.22

Seleccionamos el botón “System Configuration”,  con el que acedemos a la pantalla de configuración mostrada en la figura Fig. A1.23 en donde se elije el botón “Search” hasta encontrar la cámara del robot, luego de encontrada se pulsa “Add Camera” para agregar a la lista de cámaras sincronizadas con la aplicación, en este caso será la cámara del robot la única cámara de la lista.

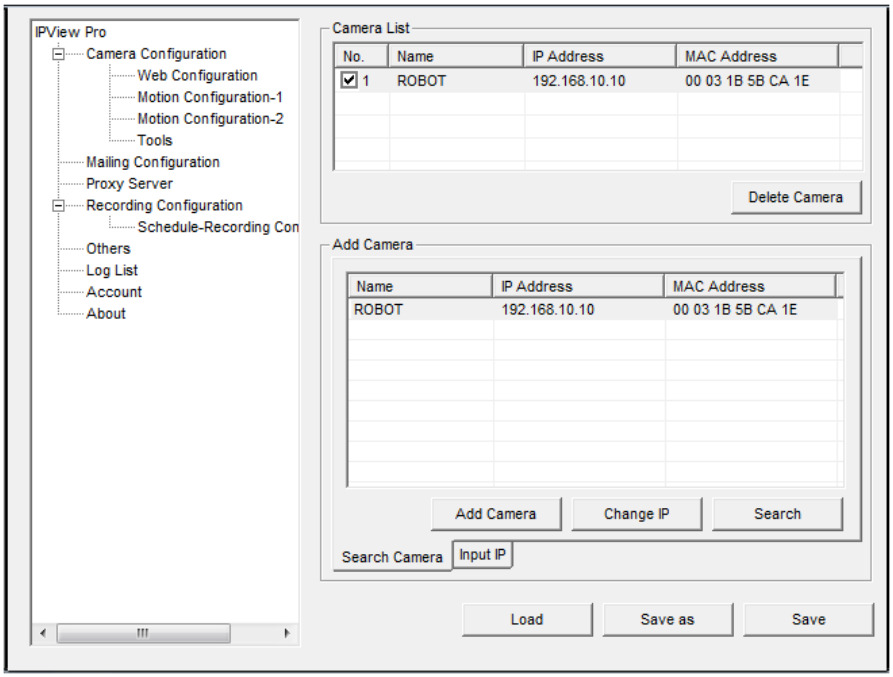


Fig. A1.23

Se presiona nuevamente el botón “System Configuration” y se retorna a la pantalla principal en donde ya se tiene acceso al video del robot, tal y como se muestra en la figura Fig. A1.24.

Se recomienda usar la función de vista en pantalla completa para tener una visualización sugestiva del entorno del robot. Para ello se debe pulsar en la pantalla principal el botón “Full-Screen Mode”  , con ello inmediatamente se visualizará el video en pantalla completa, sobre la misma se montara la interfaz de la aplicación principal desarrollada. Se guarda automáticamente la configuración de la cámara en la memoria, por ello la siguiente ocasión que se acceda a la cámara se tendrá el video del robot tras esperar al rededor de 20 [s] hasta que el sistema identifique a la cámara automáticamente.

Fig. A1.24

Sincronización Modulo Bluetooth.

Se debe sincronizar el módulo Bluetooth integrado al robot móvil a la computadora donde se ejecute la interfaz de usuario. En la barra de tareas de Windows se selecciona la pestaña de Bluetooth y se siguen los pasos detallados en las figuras Fig. A1.25 - Fig. A1.31.

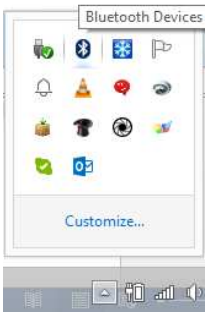


Fig.A1.25

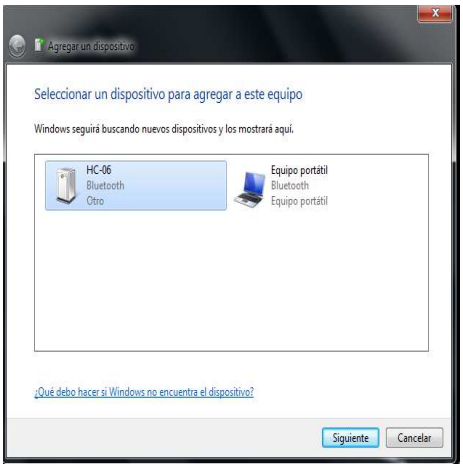


Fig.A1.26

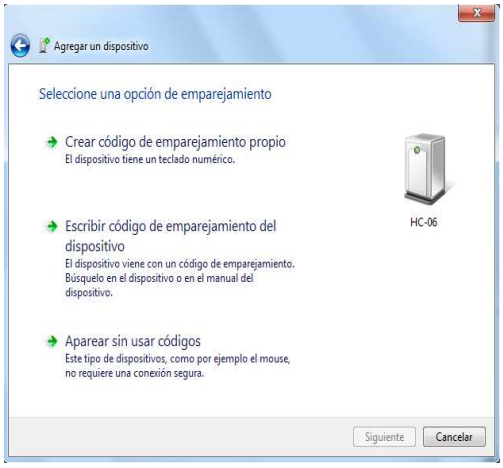


Fig.A1.27

Se selecciona “escribir código de emparejamiento del dispositivo” el mismo que por default es 1234.

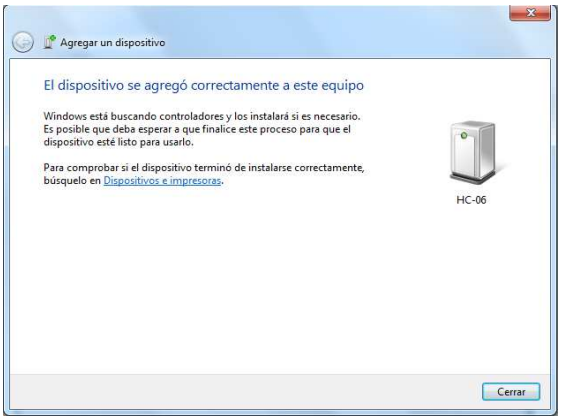


Fig.A1.28

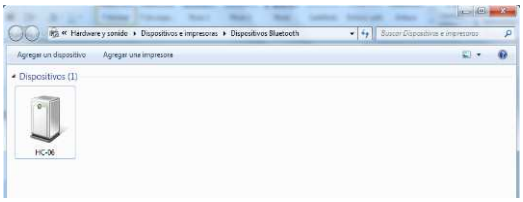


Fig.A1.29

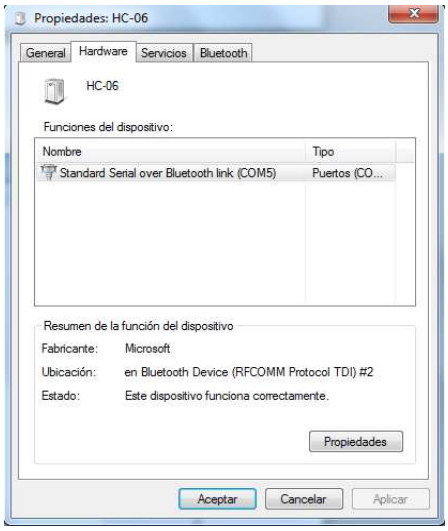


Fig.A1.30

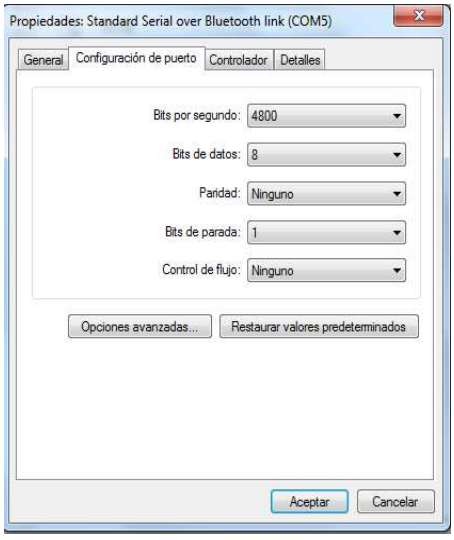


Fig.A1.31

Instalación de la aplicación “CONTROL FACIAL”.

Con el fin de facilitar la puesta en marcha de la aplicación se ha generado un instalador con las librerías y archivos necesarios para ejecutar la aplicación principal a la que se ha denominado “CONTROL FACIAL”. Tras ejecutar el Setup.exe ubicado en el Debug de la carpeta “CONTROL FACIAL” el asistente le guiará con la instalación. Es importante que en el proceso de instalación tenga conexión a internet en caso de que requiera algún complemento del sistema para concretar la instalación. Esta actualización se ejecutará de forma automática en la instalación. De las figuras Fig.A1.32 a la Fig.A1.36 se detalla el proceso de instalación de la aplicación principal.

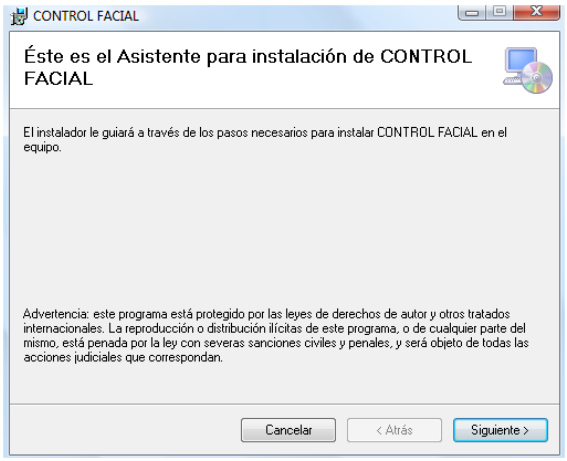


Fig.A1.32

Fig.A1.33

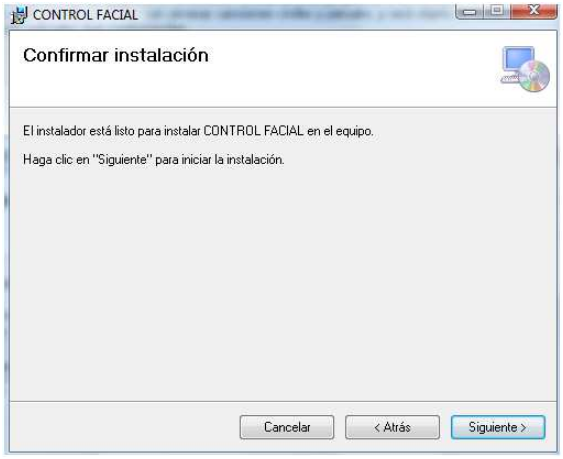


Fig.A1.34

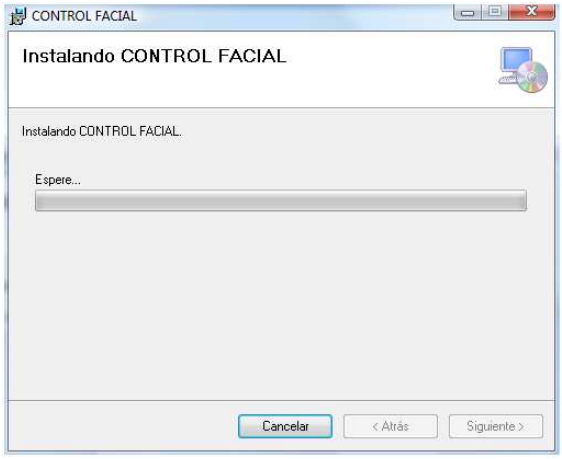


Fig.A1.35

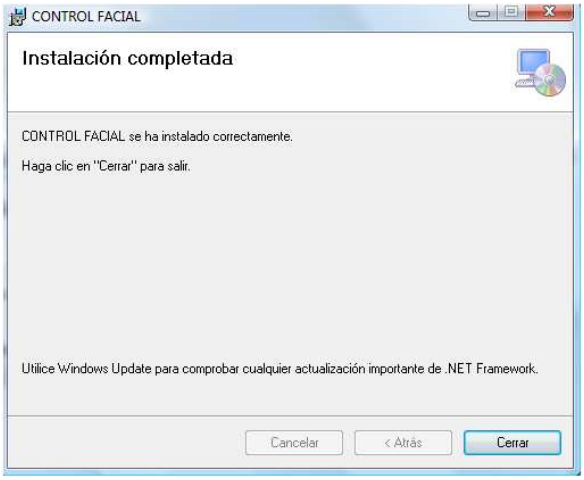


Fig.A1.36

Manipulación del Robot.

El robot móvil está provisto de un par de interruptores el de la Fig. A1.37 es para la activación del circuito de control y motores, mientras que el mostrado en la figura Fig. A1.38 tiene como objetivo la activación de la cámara integrada al mismo cuando la cámara se activa deberá esperarse a que el led rojo indique que está conectada a la red inalámbrica y lista para transmitir video. De igual manera un led del módulo Bluetooth titila indicando que está activado. Finalmente el robot se ubica en el ambiente de pruebas para usar la aplicación.

Fig. A1.37

Fig. A1.38

De igual forma se cuenta con un par de conectores tipo hembra para la carga de baterías se le vincula a una batería de 12 y 6[V] respectivamente. Para la carga de las baterías se tiene un conector tipo hembra para cada batería, es importante por seguridad tener los interruptores en posición de apagado en el momento de carga de las baterías. La carga de las baterías se la efectúa desde cualquier fuente con un voltaje equivalente al nominal de las baterías. Los conectores para la carga se los identifica en las figura Fig.A1.39 y Fig. A1.40

Fig. A1.39

Fig. A1.40

Uso de la aplicación.

Una vez instalada correctamente la aplicación, su manejo es muy sencillo e intuitivo gracias a las ayudas visuales con las que ha sido programada.

Al ejecutar la aplicación, se le solicitará que seleccione un puerto serial de entre los que la aplicación ha encontrado, se deberá seleccionar el puerto asignado al módulo bluetooth previamente sincronizado con la PC y pulsar el botón “ABRIR PUERTO”. Luego se pedirá que el usuario fije su posición; este deberá ubicarse de modo tal que la webcam tenga el enfoque del entorno del usuario y el mismo se sienta cómodo ya que será la posición definida como neutral es decir el sistema detectará esta posición como referencia.

Se selecciona el botón “INICIO” e inmediatamente la aplicación hace un sondeo de la posición del usuario por aproximadamente 3 segundos en los cuales se le solicita al conductor permanecer “QUIETO” antes de dar luz verde e indicar al usuario final que la aplicación está lista a ser usada en donde la interfaz se mostrará de forma similar a la indicada en la figura Fig. A1.41.

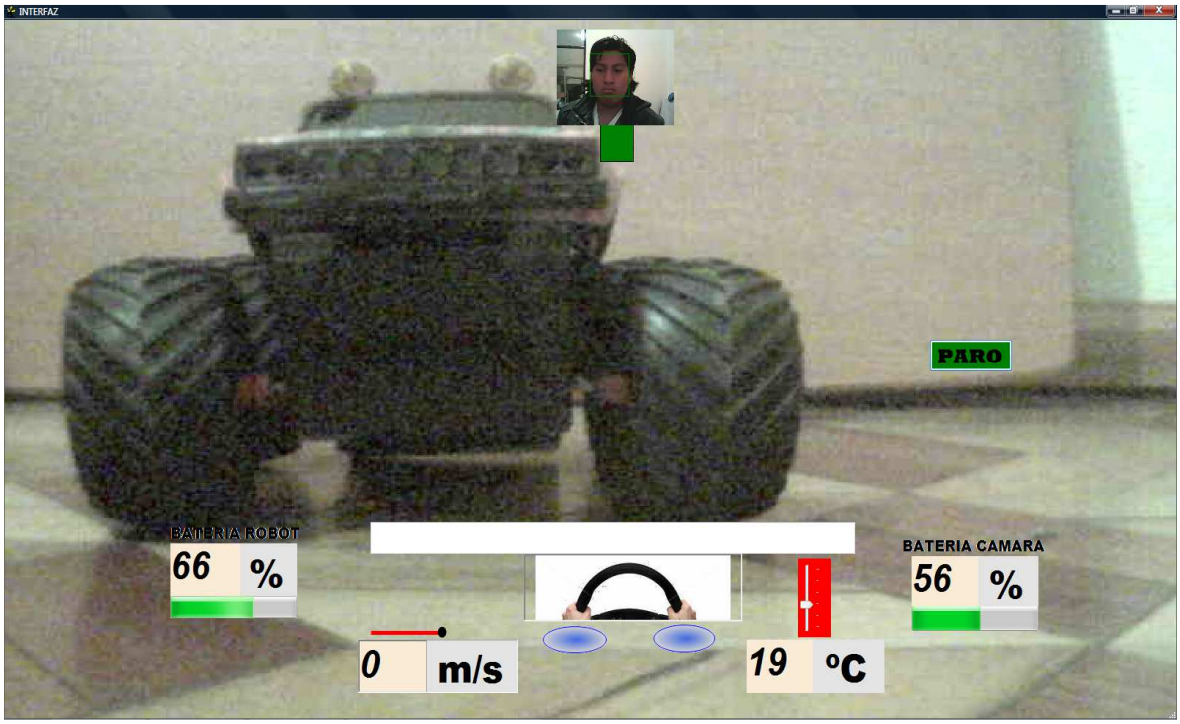


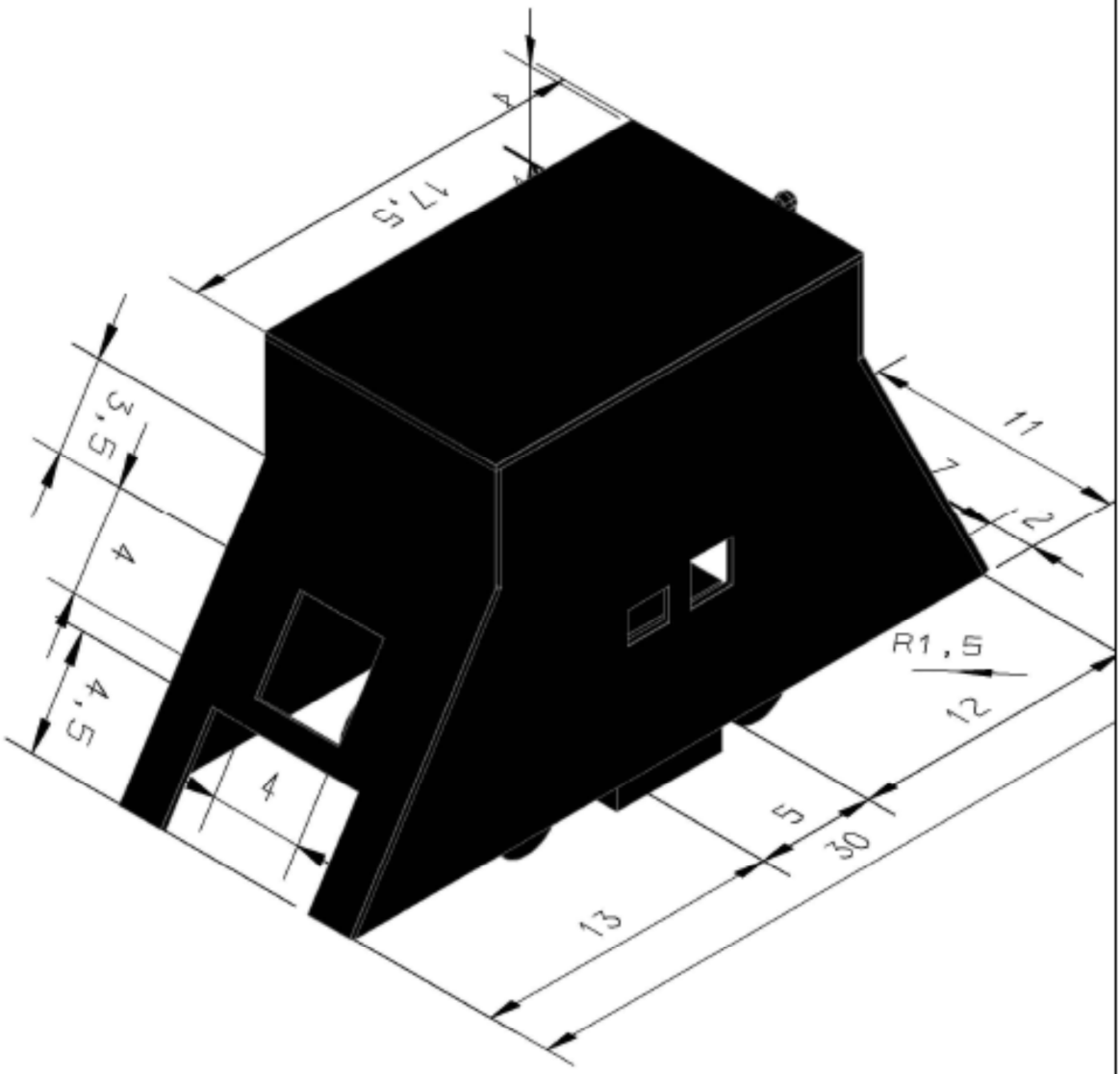
Fig. A1.29

Para controlar el mini robot móvil el usuario debe realizar los movimientos de cabeza correspondientes. Las condiciones iniciales del robot fijan como dirección inicial hacia adelante. Cuando se mueve la cabeza hacia abajo el robot acelera si su dirección esta hacia adelante y frena si el robot se dirige hacia atrás; al mover la cabeza hacia atrás el robot frena si se dirige hacia adelante y acelera si se dirige hacia atrás. Para cambiar la dirección basta con seguir frenando cuando el robot se ha detenido. La dirección del robot se la gobierna de tal forma que el movimiento de la cabeza hacia la derecha guía al robot a la derecha, de igual forma el movimiento de cabeza a la izquierda guía al robot a la izquierda. La presencia de obstáculos sea adelante o atrás detendrá al robot y lo pondrá en condiciones iniciales indicándole al usuario lo ocurrido.

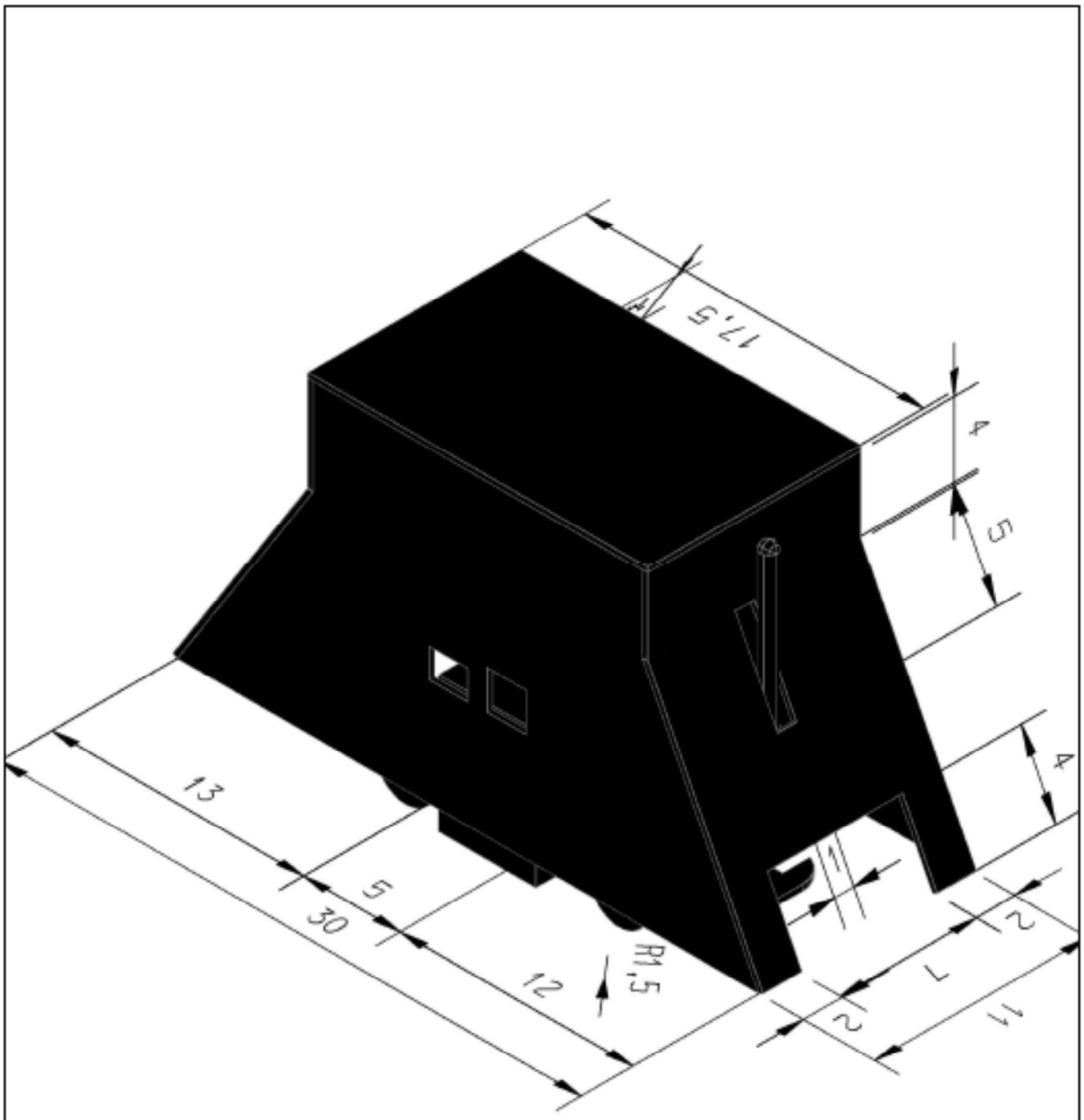
Para finalizar la aplicación o cambiar de usuario se hará uso del botón “PARO” (Habilitado solo cuando la aplicación está en uso), el robot vuelve a condiciones iniciales donde se puede iniciar de nuevo el uso de la aplicación con el mismo o diferente usuario, o bien finalizar el programa cerrando el puerto y accediendo a la pestaña del catálogo cerrar.

ANEXO2

PLANOS DEL ROBOT.



	NOMBRE:	FECHA:		INSTITUCION:
Disenado	Lenin Bautista	02-2014		ESCUELA POLITECNICA NACIONAL
Dibujado	Lenin Bautista	02-2014		
Revisado	Danilo Chavez	02-2014		
Aprobado	Danilo Chavez	02-2014		
Escala	1:5	PLANO No. 1		TITULO: ROBOT MOVIL
Unidad	cm			
VISTA	FRONTAL			
FACULTAD DE INGENIERIA ELECTRICA Y ELECTRONICA				



	NOMBRE:	FECHA:	INSTITUCION:
Diseñado	Lenin Bautista	02-2014	ESCUELA POLITECNICA NACIONAL
Dibujado	Lenin Bautista	02-2014	
Revisado	Danilo Chavez	02-2014	
Aprobado	Danilo Chavez	02-2014	
Escala	1:5	PLANO No. 2	TITULO: ROBOT MOVIL
Unidad	cm		
VISTA	POSTERIOR		
FACULTAD DE INGENIERIA ELECTRICA Y ELECTRONICA			

ANEXO3

DIAGRAMAS CIRCUITALES Y ESQUEMAS PCB

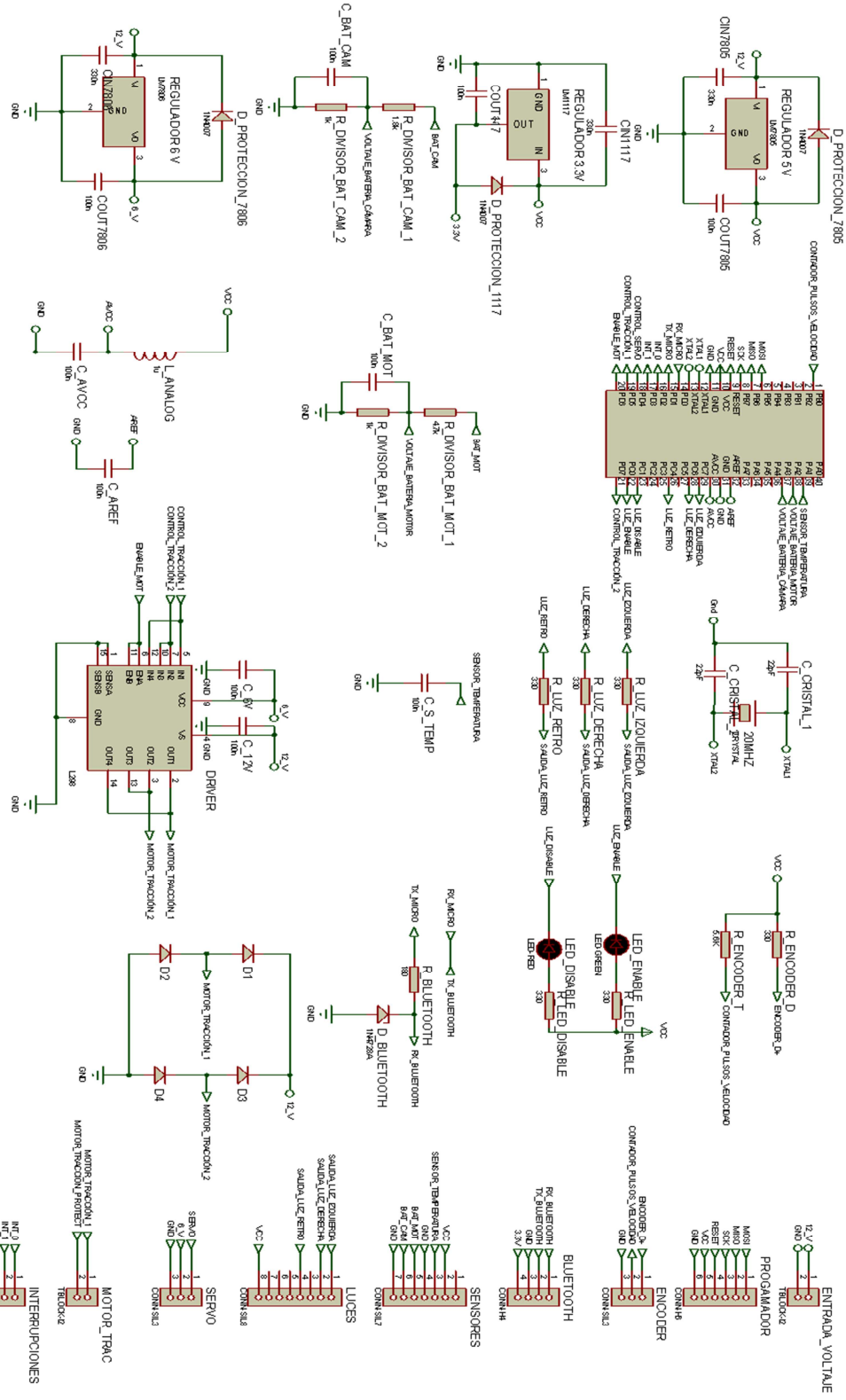


Fig. A3.1. Diagrama de placa principal de control

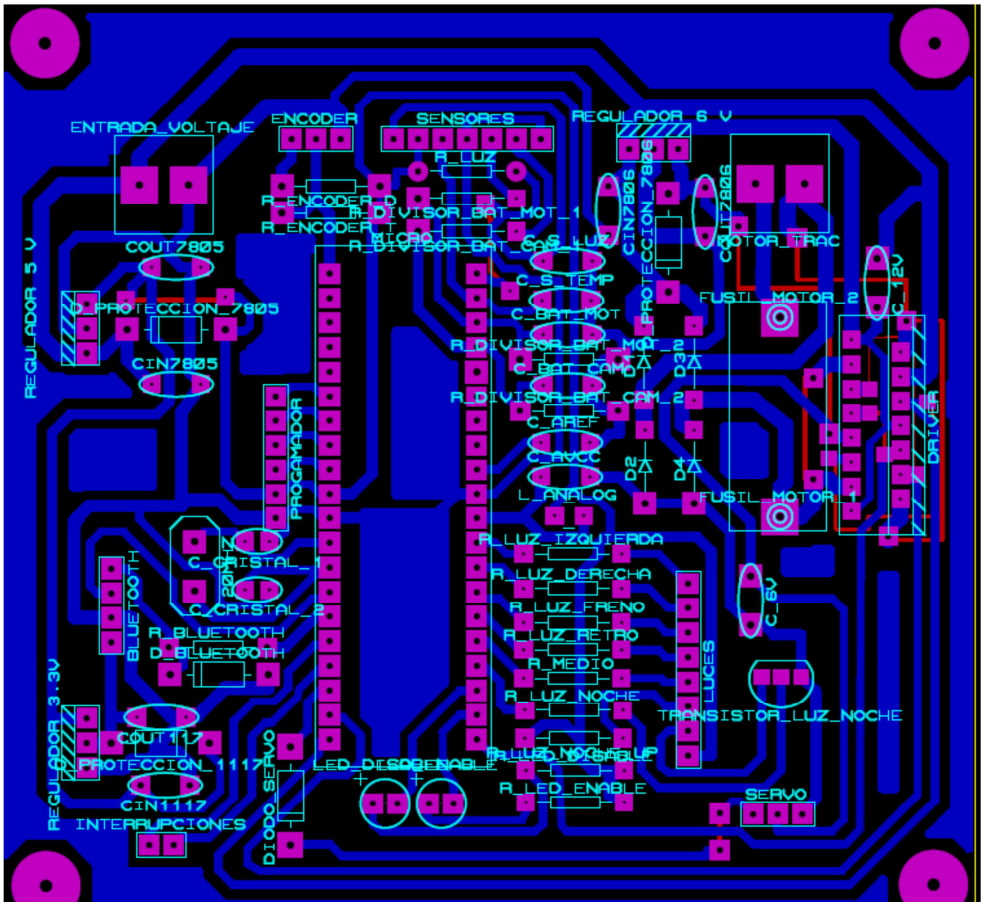


Fig. A3.2. PCB de placa principal de control

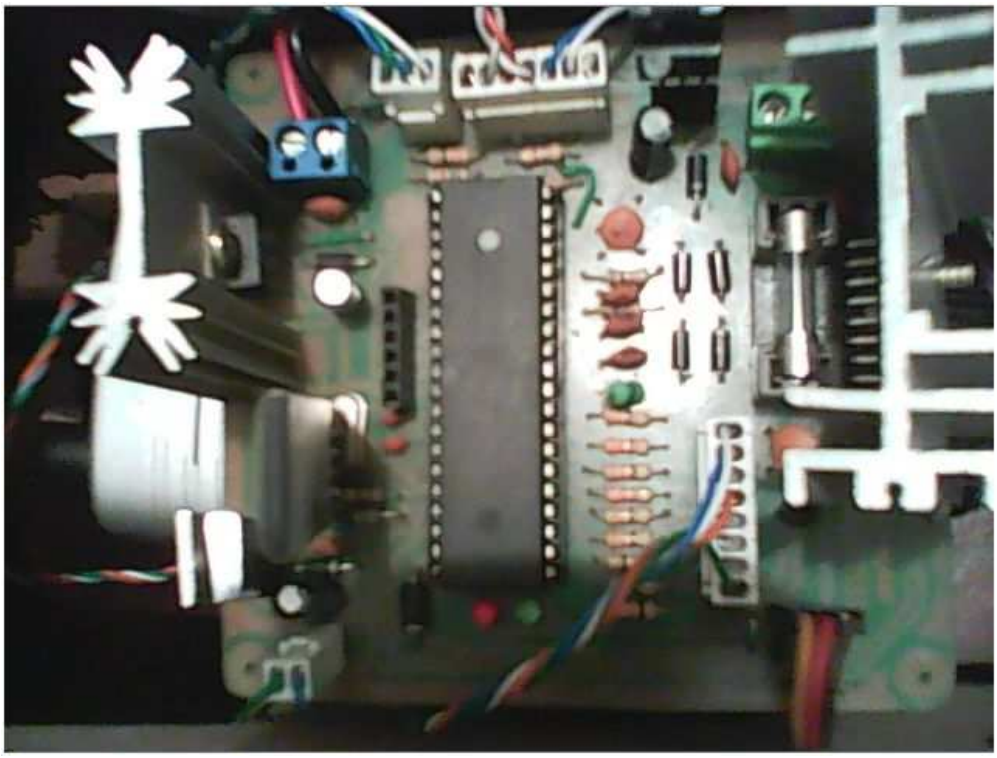


Fig. A3.3. Placa principal de control

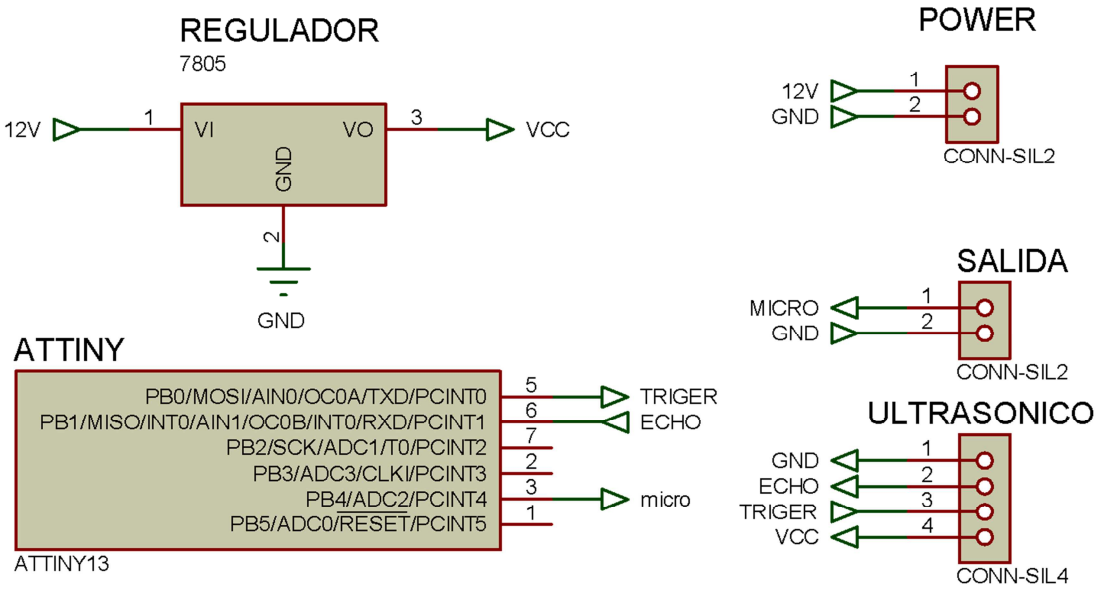


Fig. A3.4. Diagrama de placa para sensor ultrasónico.

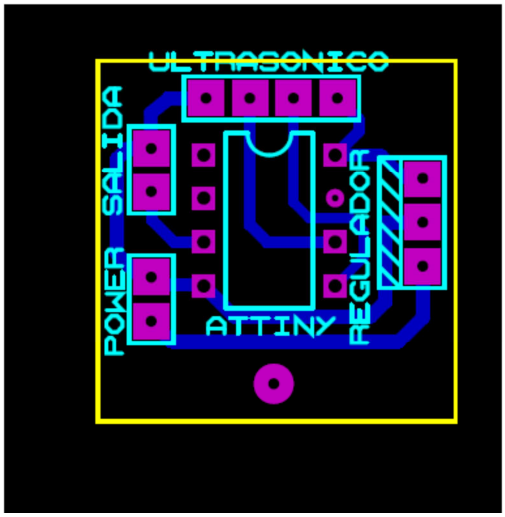


Fig. A3.5. PCB de placa para sensor ultrasónico.

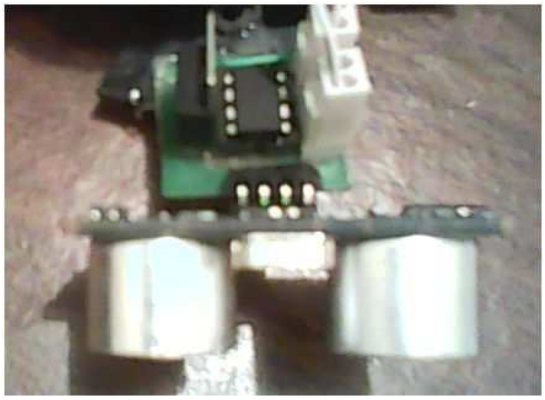


Fig. A3.6. Placa para sensor ultrasónico.