UNIVERSIDAD TECNOLOGICA NACIONAL
FACULTAD REGIONAL SAN NICOLAS

INGENIERIA EN ELECTRONICA

PROBLEMA DE INGENIERÍA

TECNICAS DIGITALES III

SEGUIMIENTO DE TRAYECTORIA MEDIANTE CONTROL VISUAL

Integrantes**:**

- Burini, Arnaldo
- Luzi, Martín
- Rolón, German

Docentes**:**

- Profesor: Poblete Felipe
- Auxiliar: Gonzalez Mariano

AÑO 2008

INDICE

OBJETIVOS DEL TRABAJO

El problema consiste en realizar y controlar un carro, para que el mismo sea capaz de seguir una trayectoria establecida por el usuario. Dicho trayecto podrá estar delineado en una superficie plana mediante una cinta o línea de un determinado color.

MATERIAS INTEGRADAS

Las siguientes son algunas de las materias que integra puntualmente el desarrollo de este proyecto. También, se indican en cada una los posibles contenidos curriculares que estarán relacionados.

• Técnicas Digitales III: Procesamiento digital de señales y Programación.
• Sistemas de Control: Acciones básicas de control.
• Medidas electrónicas II: Conversor Digital Analógico
• Electrónica Aplicada: Drivers para Motores.

POSIBLES APLICACIONES

Las posibles aplicaciones comerciales y/o didácticas que se le encuentran a este trabajo son:

• Utilización para el traslado de una carga desde un punto a otro, siguiendo una trayectoria fija, mediante un vehículo no tripulado.

• Implementación en un símil de Pantógrafo.
Básicamente este dispositivo es capaz de seguir el contorno de una pieza, por ejemplo de chapa o de madera mientras otro carro copia el movimiento realizado por el elemento sensor desplazando un elemento de corte o de propósito similar para en definitiva copiar dicha pieza.

Luego de discutir las diferencias entre estas dos aplicaciones, el grupo se decidió por realizar el desarrollo del proyecto teniendo en cuenta exclusivamente la primera aplicación. Esta decisión era muy importante a la hora de calcular las dimensiones del carro y los elementos necesarios para su operación.

PROFESORES ENTREVISTADOS

Durante el cursado de la materia se han realizado charlas con los profesores de la cátedra:

• Poblete, Felipe
• González, Mariano

Como así también, en conjunto con estas charlas se llevaron a cabo debates junto con los demás estudiantes, mediante los cuales se presentaba la idea del problema y se discutían las posibles soluciones.

BIBLIOGRAFÍA

• *Apuntes de cátedra.*
- Capitulo 3 Programación en ambiente de PC – Poblete, Felipe
- Detección y seguimiento de objetos – Bernardo Calla, Gabriel Malespina, Enzo Varela y Cristian Palomeque

• *Sitios de Internet.*
- http://dis.um.es/~ginesgm

Dr. Ginés Garcia Mateos - Departamento de Informática y Sistemas, Universidad de Murcia.

•http://www.latindevelopers.com/forum/viewtopic.1371.html;sid=fa18a27224e46677168243522fbffd60

Sitio para la descarga gratuita de OpenCV 5

- http://www.cuantosprogramas.com/descargar/1087/windows/Dev-C-5.0-Beta-9.2-4.9.9.2.html

Sitio para la descarga gratuita de DEV-Cpp 5.0

• *Libros, etc.*
- Procesamiento Audiovisual Sesión 2. Instalación y uso de IPL y OpenCV - Dr. Ginés Garcia Mateos.

DESARROLLO

INTRODUCCIÓN

La solución al problema se plantea como un carro en el cual irán montados una cámara Web junto con los dispositivos de control, los motores de dirección y avance y una notebook. Esta última será la encargada de correr el software que hará de intermediador entre las imágenes obtenidas por la cámara Web y la electrónica que controla los motores.
Se eligió el montaje de una notebook sobre el carro en lugar de utilizar una PC de escritorio, simplemente para darle mayor libertad de movimiento al mismo independizándolo de conexiones a la red eléctrica.
En la **Figura 1** se puede observar el montaje básico al cual se hace referencia y como la cámara esta apuntada a la línea de trayectoria. De esta manera a medida que el carro continúe avanzando, la cámara instalada sobre el mismo será capaz de detectar cuando este se deba desviar, de tal manera que según la variación de la línea de trayectoria dibujada, se controlará la dirección del carro para que se vuelva a alinear con la misma.
En la figura del montaje se observa también, que el carro podrá estar compuesto por tres ruedas, las dos traseras le darán la velocidad de avance y la delantera la dirección. Esta última estará direccionada por un motor de continua, del mismo modo que el correspondiente para el avance.
El software ejecutado en la notebook se encargará de controlar el accionamiento del motor de avance y corregir el desvió del carro a fin de que circule por la trayectoria indicada.

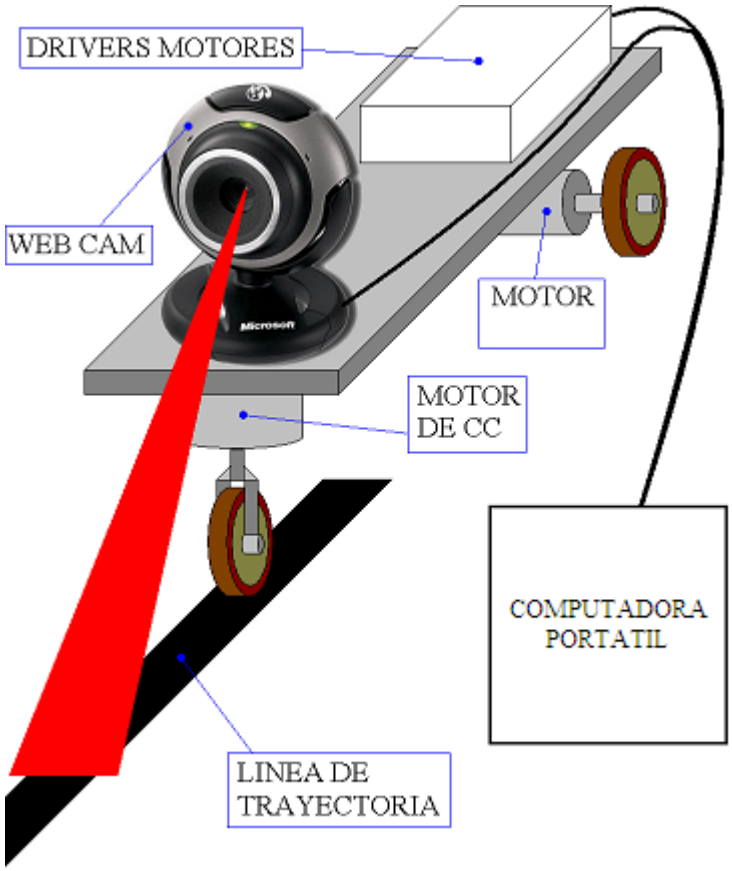


Figura 1. Montaje básico.

TEORÍA DE FUNCIONAMIENTO

Como ya se mencionó, el control de dirección y el avance serán llevados a cabo por un programa el cual estará desarrollado en Dev-Cpp y hará uso de las librerías OpenCV.
Básicamente este programa deberá ser capaz de detectar la línea de trayectoria e indicar su posición. Para esto se utiliza la función CvCamshift de OpenCV, en base a la cual se desarrolla todo el programa.
El seguimiento de la línea estará dado dentro de un área de detección, la cual podrá ser ajustada tanto en tamaño como en posición. De esta manera colocando el área de detección más cerca del extremo superior de la imagen se podrán llevar a cabo acciones predictivas, es decir doblar la rueda antes de que la curva llegue a la misma. De manera análoga se obtendrá una respuesta atrasada si se coloca el área de detección en la parte baja de la imagen.
Por otro lado, el ancho horizontal de esta área determina la máxima excursión de la curva, por lo tanto será lo más ancha posible para que la línea pueda ser detectada en la totalidad de la imagen que toma la cámara.
De modo contrario, el ancho vertical del área de detección será lo más pequeño posible para darle mayor sensibilidad ante las variaciones.
Se observa a modo de ejemplo una línea recta (**Figura 2**) y una curva (**Figura 3**), en las cuales la trayectoria es indicada por color rojo, el área de detección de color gris y el objeto detectado con color celeste. En ambos casos el área de detección tiene el máximo ancho horizontal y un mínimo vertical.

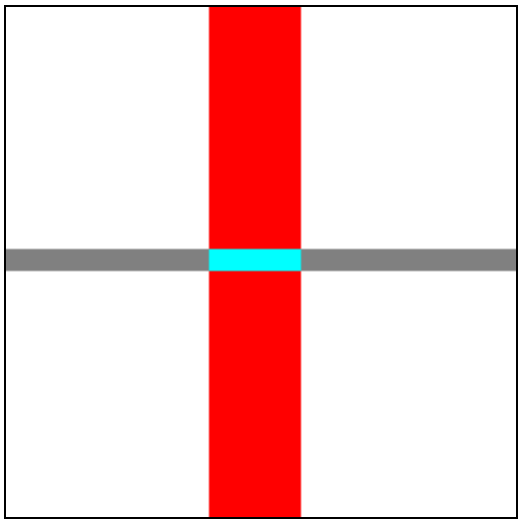


Figura 2. Imagen de una recta.

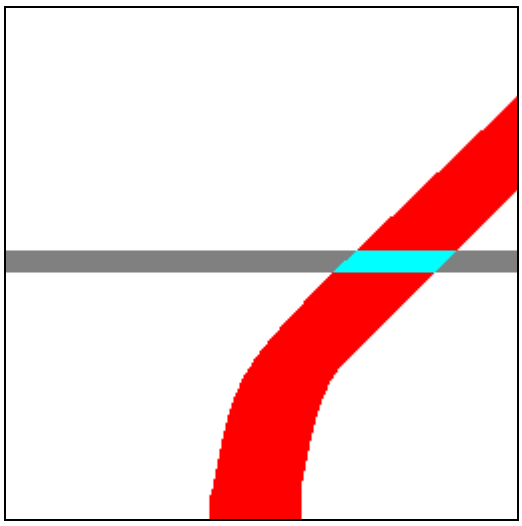


Figura 3. Imagen de una curva.

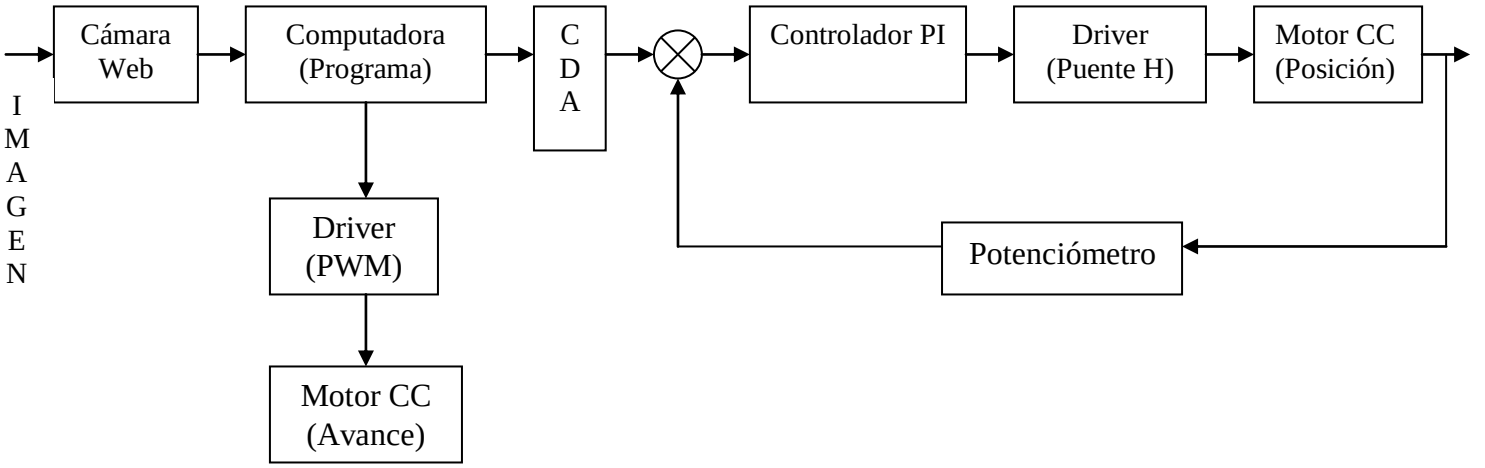
De esta manera, dependiendo de qué tan alejados esté el objeto detectado del centro de la imagen mayor será el ángulo giro que se deberá proporcionar el motor de dirección, ya sea hacia la izquierda como la derecha según corresponda.

Aparte de realizar el seguimiento de la línea e indicar su posición, el programa deberá indicar cuando la línea es detectada y cuando no. De esto último dependerá el accionamiento o no del motor de avance.

Otra característica que deberá tener el programa, será la utilización de una función de filtrado de colores para evitar que se realice el seguimiento de un objeto indeseado y solamente se tenga en cuenta la línea de trayectoria la cual tendrá un color predeterminado (rojo para el ejemplo dado).

DIAGRAMAS

El diagrama en bloques muestra básicamente como interactúan las distintas partes del sistema. Se debe tener en cuenta también la acción de control que se lleva a cabo internamente en la PC.
Diagrama del lazo de control:



La **Imagen**, es decir la trayectoria, es vista por la **Cámara Web** y enviada como información digital a la **Computadora** a través del puerto USB. La **computadora** recibe la información y mediante el software desarrollado toma dos acciones de control distintas que influyen sobre el motor de avance y el de dirección.
Por un lado, si el software interpreta que la línea esta presente, mediante uno de los pines del puerto paralelo la **computadora** habilita al **Driver PWM** que se encarga de energizar el **Motor de avance**.
Por otro lado, el software estará indicando en cada momento la posición de la línea dentro de la excursión de la imagen recibida a través de la **cámara Web**. Esta posición es transformada a un número hexadecimal que va de 00 a FF y sacada a través del puerto paralelo.
La información digital colocada en el puerto paralelo, correspondiente a la posición de la línea detectada, es convertida a un valor analógico mediante un **Conversor DA**. Este valor analógico se compara con una tensión de referencia que se obtiene de un **Potenciómetro** conectado al eje del **Motor de Posición**.
El error que se produce, entre el valor analógico a la salida del **CDA** y el correspondiente al **Potenciómetro**, es convertido mediante el **Controlador PI** en una acción correctiva.
Esta acción correctiva actúa sobre el **Driver de puente H** que manipula el movimiento del **Motor de Posición.**

DRIVERS DE LOS MOTORES

Los Drivers se referencian como los circuitos electrónicos que se encargarán de accionar los motores de CC. Para el caso del motor que da la dirección se utilizará un puente H, ya que el mismo deberá ser capaz de girar tanto para un lado como el otro. A su vez dicho puente H será comandado por un controlador tipo PI el cual comparara la posición del dicho motor a través de un potenciómetro montado en él, con un valor de referencia que será establecido a la salida de un conversor digital a analógico conectado en el puerto paralelo de la notebook.
Por otro lado el motor de avance será comandado por un circuito de PWM llevado a cabo con un NE555. De esta manera se obtiene un ahorro importante de potencia en el transistor que controla el motor y se puede regular la velocidad de avance del mismo.
Cabe destacar la salida del NE555 estará habilitada o no dependiendo de un cable conectado también al puerto paralelo de la notebook.
A continuación se detallan brevemente los pines que serán utilizados del puerto paralelo y cada uno de los circuitos descriptos anteriormente.

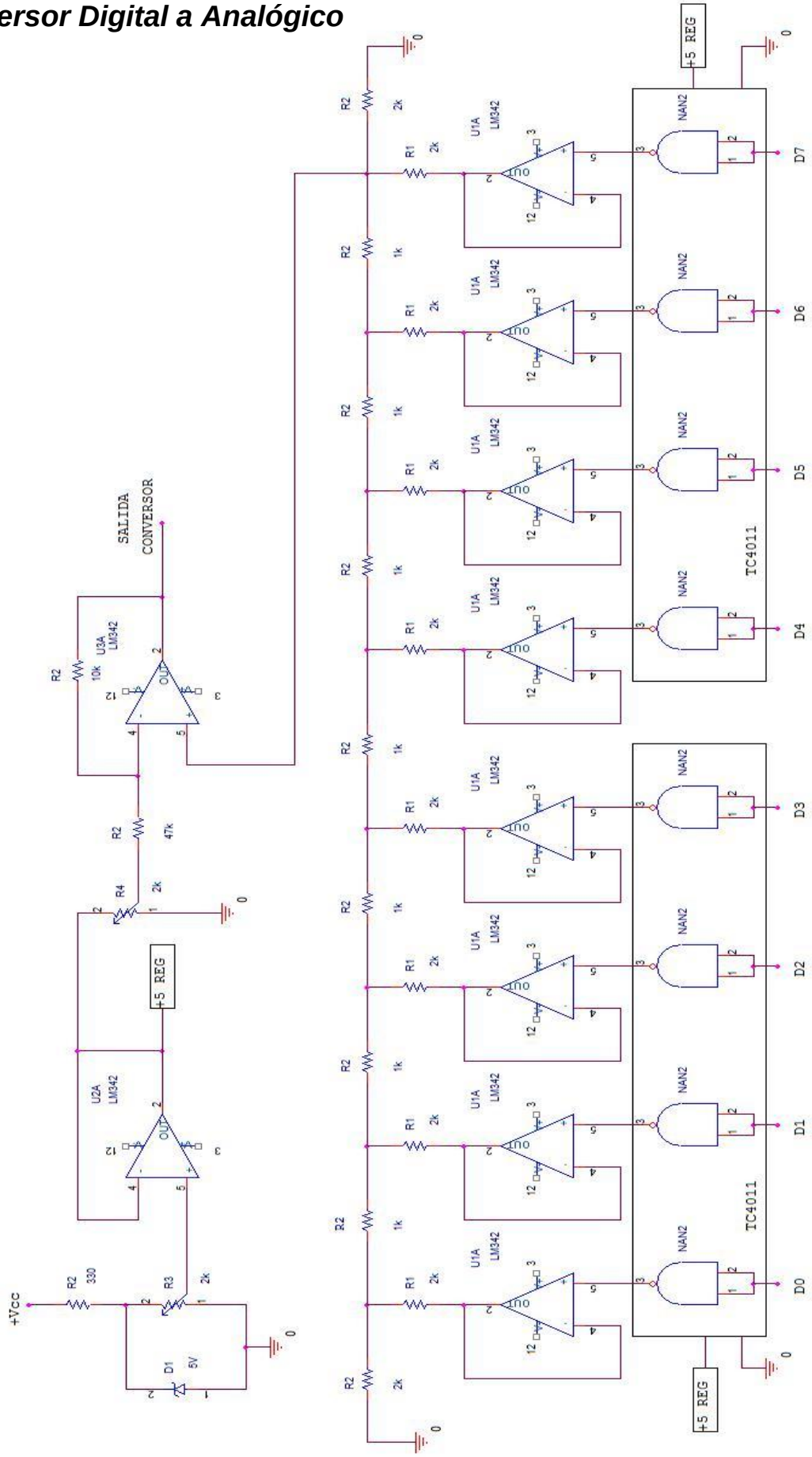
Puerto paralelo

La utilización del puerto paralelo fue uno de los mayores inconvenientes en el desarrollo de este problema de ingeniería. Esto no se debió a la programación o al manejo del mismo en Dev-Cpp, ya que como ya se explica en informes anteriores existe un bloqueo de parte de Windows Xp para que el usuario no acceda al mismo. Pero la solución a este problema se puede hallar fácilmente en internet o en la bibliografía de 'Detección y seguimiento de objetos', precisamente en la sección 9 se explica como usar la librería inpout32.dll.
El verdadero problema radica en que ninguna de las notebooks actuales incorpora un puerto paralelo.

La solución se logro al conseguir una placa PCMCIA de puerto paralelo compatible con IEEE 1284 SPP/EPP/ECP, es decir que cuando se la instala aparece un puerto LPT tal cual como en una PC de escritorio y no genera un puerto virtual como el caso de algunas otras.
Del puerto paralelo todos los bits del registro de Datos son usados y conectados como salida al conversor digital analógico. Por otro lado, solamente el bit de Strobe del registro de Control es usado como salida para habilitar/deshabilitar el sistema de PWM que maneja el motor de avance.

Una vez instalada la placa aparece un puerto LPT, donde en el Administrador de Dispositivos del Sistema se puede observar la dirección base del mismo. En este caso es FFE8.
Por lo tanto la dirección del registro de Datos es FFE8, mientras que para obtener la correspondiente al registro de Control hay que sumarle dos a la dirección base, encontrando la misma en FFEA.
Esta información es imprescindible a la hora de programar el puerto paralelo.

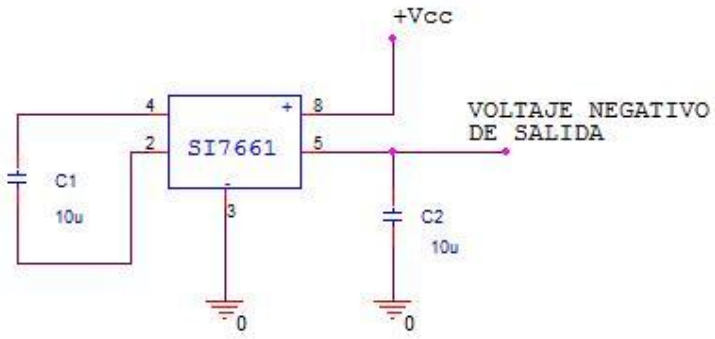
Conversor Digital a Analógico



Operación del conversor digital – analógico:

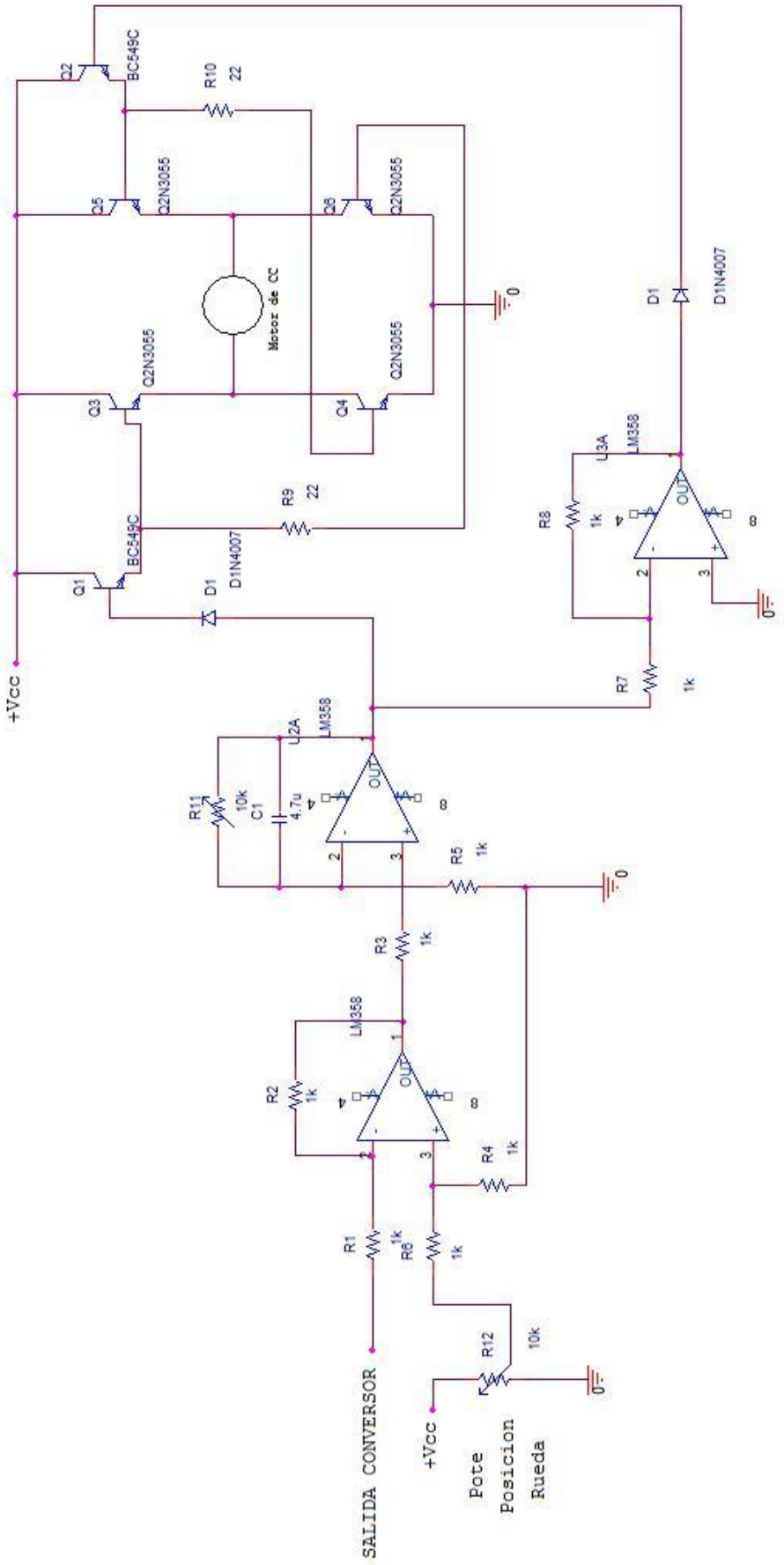
El convertidor R-2R basa su funcionamiento en una red resistiva conocida como red escalera o R-2R, note que la resistencia en serie es la mitad de la resistencia en paralelo. Fácilmente se puede obtener la resistencia equivalente y la corriente por cada resistencia 2R en paralelo. La resistencia equivalente vista por la fuente de alimentación es 2R. La corriente en la primera resistencia 2R (de izquierda a derecha) es I/2, siendo esta la resistencia de mayor peso, la corriente en la resistencia 2R que sigue es I/4, la corriente en la resistencia 2R que sigue es I/8 y en la resistencia 2R que sigue es I/16 siendo esta la de menor peso para este convertidor. Según el valor de la entrada digital estas corrientes son enviadas a tierra o son enviadas al amplificador operacional. Por ejemplo si el número Digital es 1000 0000b entonces I/2 es enviado al operacional mientras que el resto de las corrientes van a tierra, si el número digital es 0111 1111b entonces la corriente I/2 es enviado a tierra mientras que el resto de las corrientes son enviadas al operacional. Sólo las corrientes que son enviadas al operacional contribuyen al voltaje de salida del convertidor. Siendo que la contribución de cada corriente proporcional al peso de cada bit entonces es fácil concluir que existe una relación de linealidad entre el Voltaje y la entrada digital D por lo que la conversión lineal de Digital a Analógico es realizada por este circuito.

Fuente para los amplificadores

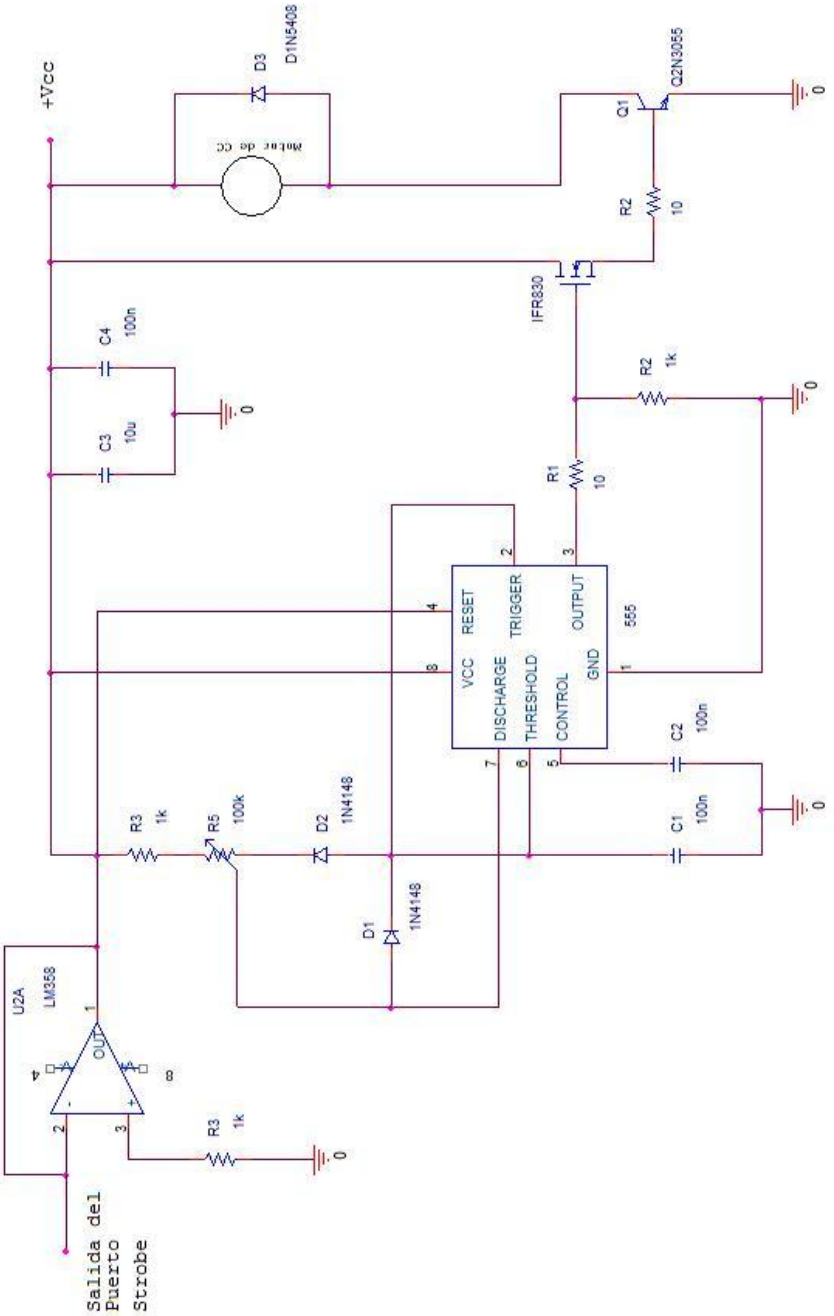


Este circuito es el que genera el voltaje negativo para alimentar a los amplificadores operacionales del controlador los cuales funcionan con –Vcc y +Vcc. Ya que los amplificadores del conversor digital – analógico se alimentan con +Vcc y Gnd solamente. El circuito integrado por lo tanto convierte el voltaje positivo de entrada en un voltaje negativo de salida pero del mismo valor en modulo.

Controlador PI y puente H



PWM



La modulación por anchura de pulsos (ó PWM, del ingles pulse-width modulation) es una técnica de modulación en la que se modifica el ciclo de trabajo de una señal periódica para, entre otras cosas, variar la velocidad de un motor. El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación al período. Cuanto mas tiempo pase la señal en estado alto, mayor será la velocidad del motor. Este tren de pulsos, en realidad, hace que el motor marche alimentado por la tensión máxima de la señal durante el tiempo en que esta se encuentra en estado alto, y que pare en los tiempos en que la señal esta en estado bajo. Se coloco un buffer a la salida del puerto para protegerlo en caso de que cortocircuito. El circuito que genera los pulsos es un integrado llamado 555 que opera en modo astable.

EL SOFTWARE

Como ya se mencionó, el programa se desarrolló en Dev-Cpp haciendo uso de las librerías de OpenCV y basándose en la función Camshift.
Debido a que ya existe una extensa documentación que abarca desde como instalar hasta explicaciones de cada una de las funciones de OpenCV, en este informe no se tratará esta información ya que inclusive ha sido incluida en posteriores informes publicados en la página web de la materia (http://www.frsn.utn.edu.ar/tecnicas3), como por ejemplo el de 'Detección y seguimiento de objetos'.
Por lo tanto, ante cualquier duda el lector podrá recurrir al material incluido en la bibliografía, evitando de esta manera material redundante.
El listado del programa se encuentra en el Anexo 1.

FUNCIONAMIENTO DEL PROTOTIPO

Cuando se ejecuta el programa 'Seguimiento.exe' aparecen tres ventanas que se pueden ver en la **Figura 4**.
La primera es una ventana de DOS que muestra el estado las variables de control, como son los puertos de Datos y de Control, a su vez da indicaciones al usuario de cómo utilizar el programa.
Otra ventana con el nombre 'Seguimiento' será encargada de mostrar la imagen capturada por la cámara Web en color pero con el agregado de que también se podrá observar sobre ella el área de detección y con una recta vertical la posición que está detectando.
Por último, la ventana de 'Filtrado' posee tres barras variables que permiten personalizar el color de la línea de trayectoria que se desea filtrar. En esta ventana se mostrara el resultado del filtrado, donde la imagen resultante aparecerá en blanco y negro e invertida verticalmente.
El filtrado puede regularse variando los valores mínimos y máximos (Vmin y Vmax) y la sensibilidad (Smin).

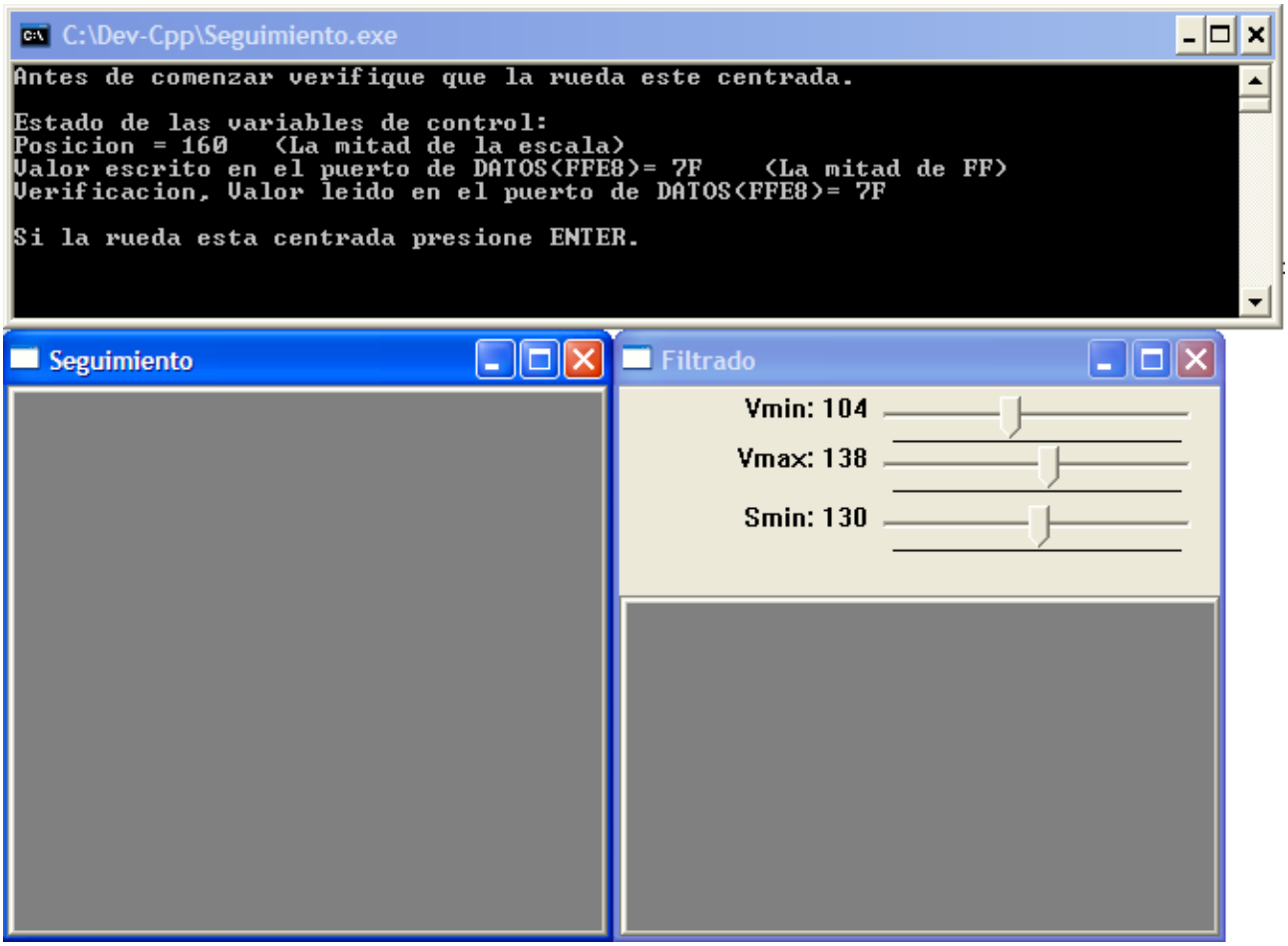


Figura 4. Inicio del programa.

Cuando apenas se ejecuta el programa, lo primero que hace es decirnos que verifiquemos que la rueda de dirección este centrada. Esto se debe a que cuando el programa inicia coloca en el puerto de Datos el valor 7F que posiciona a la rueda en el medio.
Si la rueda no está centrada, esta se puede centrar mecánicamente aflojando la tuerca que la sostiene y modificando su posición o re posicionando el potenciómetro que mide la posición de la misma.

Esto último se deberá hacer solo si se observa que el mismo se ha desplazado se su posición correcta.
Una vez que se verifica que la rueda de dirección está centrada, se presiona la tecla Enter para continuar con la ejecución del programa, visualizando lo que se observa en la **Figura 5**.

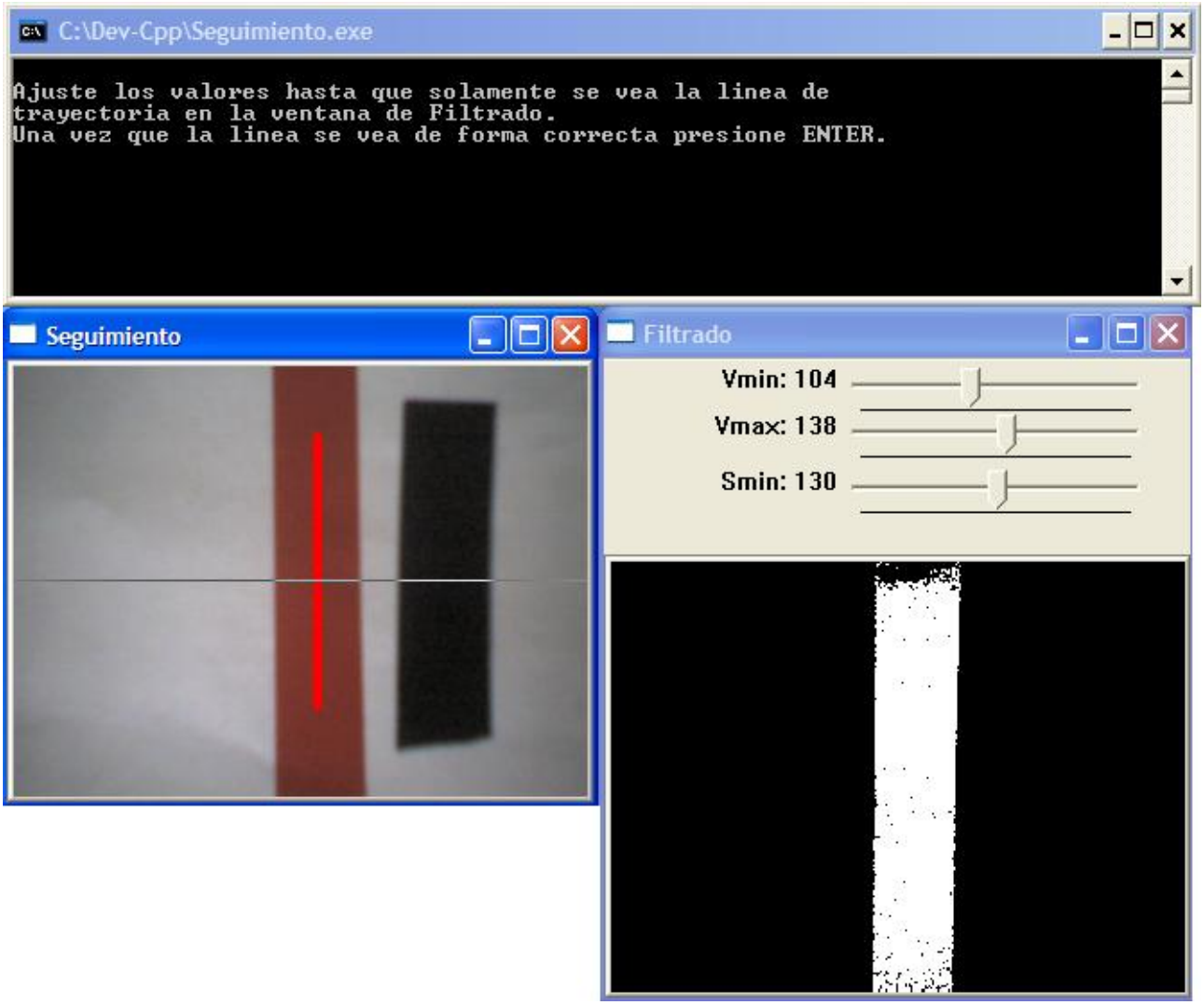


Figura 5. Ajuste del filtrado.

En esta etapa del programa, en la ventana de DOS se nos pide que ajustemos el filtrado hasta que solamente se vea la línea de trayectoria.
Se observa en la ventana de 'Seguimiento' que en este caso la línea de trayectoria es de color roja, también se observa la presencia de un objeto indeseado (línea negra) el cual se desea filtrar.
En dicha ventana, como ya se menciono, se observa el área de detección de color gris. Para el caso esta área es de un solo pixel de ancho vertical, tiene el ancho máximo horizontal y está centrada verticalmente en el medio de la imagen.
Por último, en dicha ventana también se observa una línea recta vertical de color roja que nos indica la posición del objeto que se está detectando. Esta posición es traducida a un valor que luego se coloca en el puerto de datos.
En la ventana de 'Filtrado' se han ajustado los valores para eliminar la línea negra.

Cabe destacar que los valores de filtrado podrían haber sido ajustados para eliminar la línea roja en lugar de la negra, como se observa en la **Figura 6**.

En esta figura se puede ver también como la línea roja de detección se posiciona sobre la línea negra.

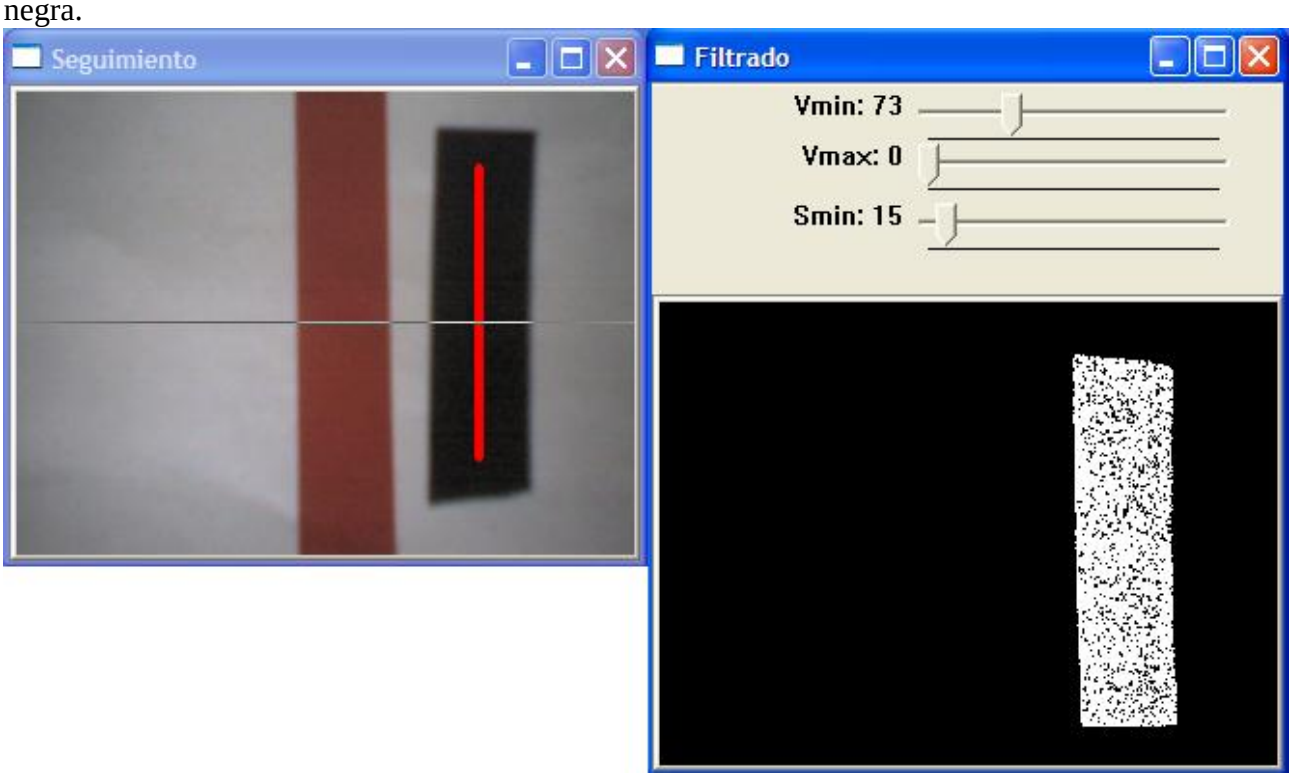


Figura 6. Detección de la línea negra.

La **Figura 7** introduce una tercera linea con contenido de color azul. Se observa como variando los valores de filtrado se consigue que sea este el objeto detectado en lugar de los anteriores mencionados.

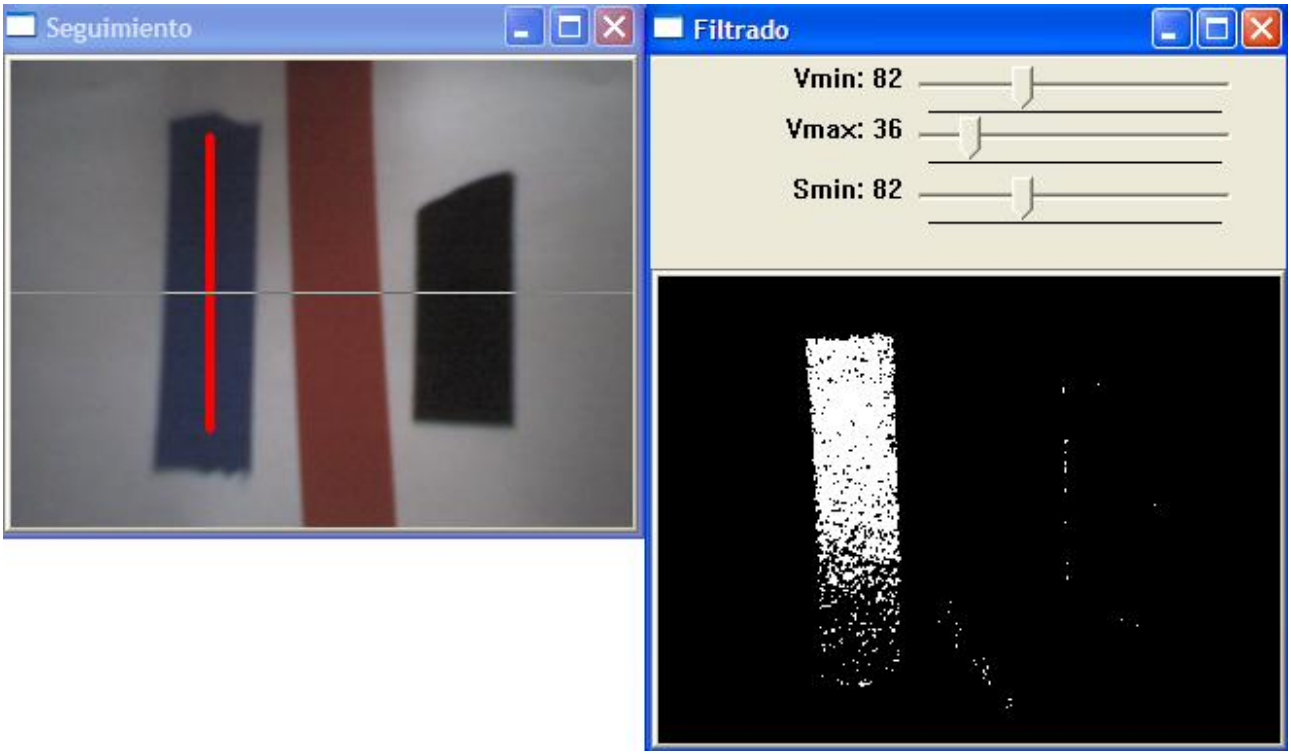


Figura 7. Detección de un objeto con contenido de color azul.

Para nuestra aplicación es necesario que cualquier objeto extraño sea filtrado y solo se detecte la línea de color roja establecida como trayectoria. Es por esto que los valores de filtrado deben ser posicionados nuevamente para obtener este resultado observado en la **Figura 8**.

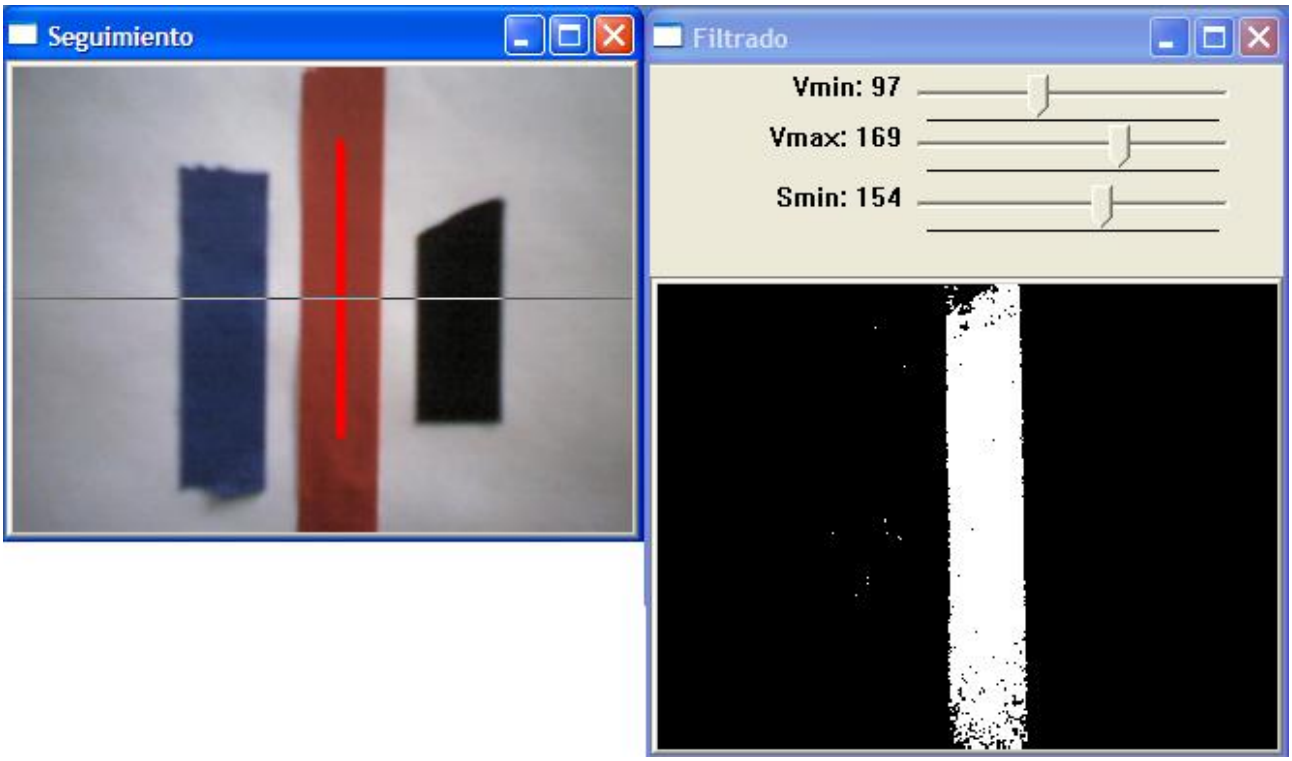


Figura 8. Detección de la línea roja solamente.

Logrado este último resultado debemos presionar Enter para continuar con la ejecución del programa. Una vez presionada dicha tecla el motor de avance será puesto en marcha y el carro iniciará su recorrido, visualizando en la ventana de DOS lo que se muestra en la **Figura 9**.

En dicha ventana se muestran los valores presentes en los puertos de Datos y de Control.
El puerto de Datos posee ocho pines por lo tanto su valor puede ir de 00 a FF. Se encontrará con un 00 cuando la línea sea detectada posicionada en el extremo izquierdo y viceversa, abarcando todos los valores del rango con las posibles posiciones de la misma.
Para el ejemplo, la **Figura 9** muestra que la línea está siendo detectada desplazada hacia la izquierda por lo que el valor en el puerto de datos es 1F.

Por otro lado, también se indica el valor que se encuentra en el puerto de Control. La figura muestra que el valor en el mismo es 0. Aquí cabe aclarar que, debido a los circuitos electrónicos que se conectan al pin de Strobe del puerto de Control, cuando dicho puerto esta en 0 el sistema de PWM se habilita, mientras que si esta en 1 es sistema queda deshabilitado y el motor se detiene.
Por lo dicho hasta aquí, en la figura de ejemplo, es lógico que el mismo se encuentre en 0 ya que la línea está siendo detectada y por lo tanto el motor debe estar encendido para continuar avanzando.

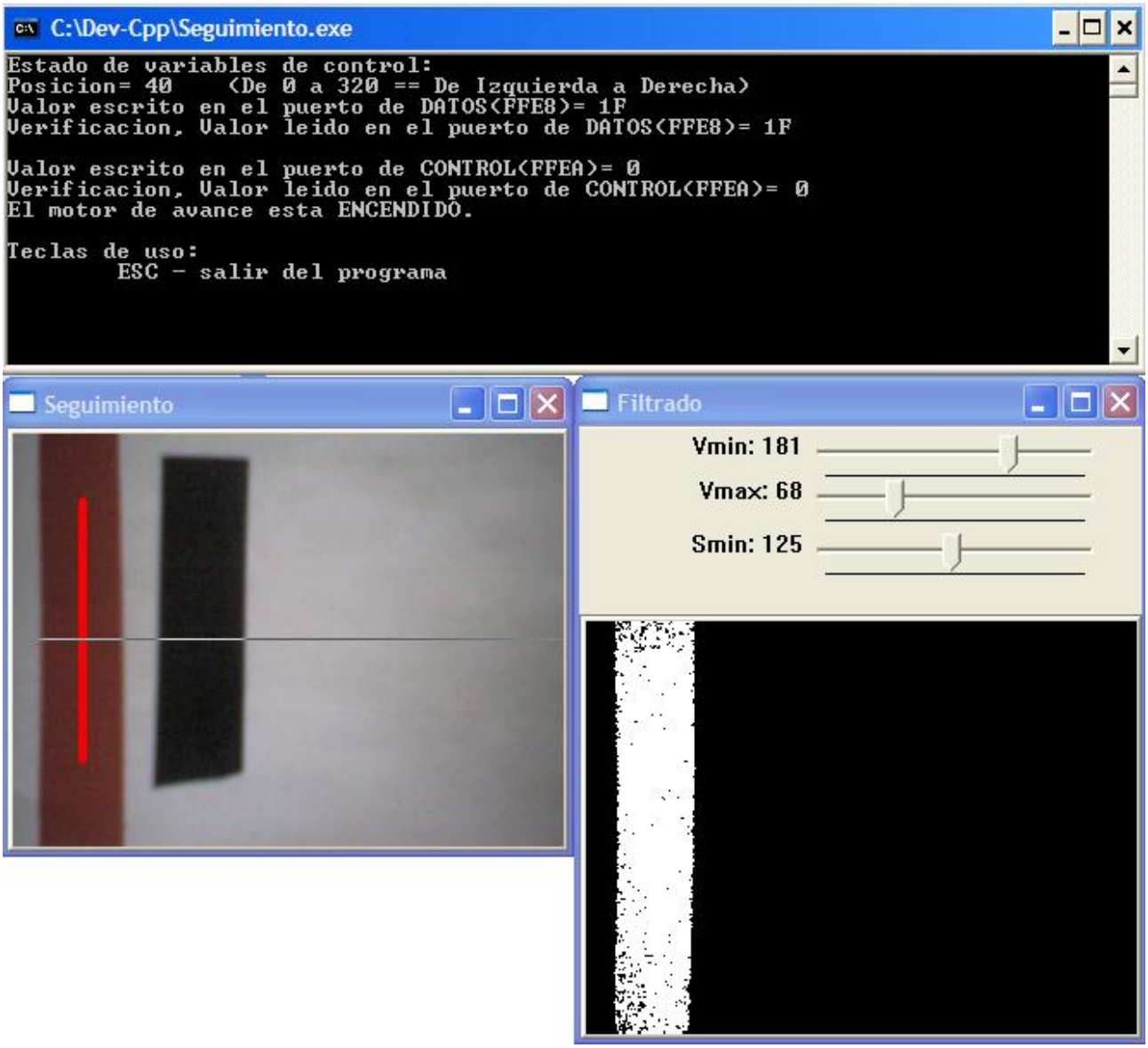


Figura 9. Línea detectada sobre la izquierda.

En esta figura podemos ver que existe una gran diferencia entre los valores de ajuste filtrado con respecto a la **Figura 8.** En la **Figura 9**, la diferencia que más se destaca es que el valor de Vmin es mayor a Vmax, cosa que era al revés en la **Figura 8**.
Entonces, ¿por qué se han cambiado los valores de filtrado si se sigue detectando solo la línea roja?

Este hecho se debe a que la cámara es sensible a las variaciones de luz. Si la luz se incrementa la cámara experimenta una especie de encandilamiento y pierde de vista al objeto en seguimiento.
Por el lado contrario, si la luz disminuye por debajo de cierto límite, entonces la cámara no puede distinguir con claridad las diferencias de colores y también se pierde el objeto en seguimiento.

Para solucionar el problema de la variación de la luz, se monta en el frente del carro un arreglo de leds de luz blanca apuntando directamente hacia el suelo donde hace foco la cámara.
Esto elimina el problema de la deficiencia de luz pero trae el problema del encandilamiento.
Pues justamente la solución al encandilamiento es trabajar con valores de Vmin por encima de Vmax.

A continuación, en la **Figura 10**, se muestra a modo de ejemplo lo que sucede cuando la línea es detectada sobre la derecha.

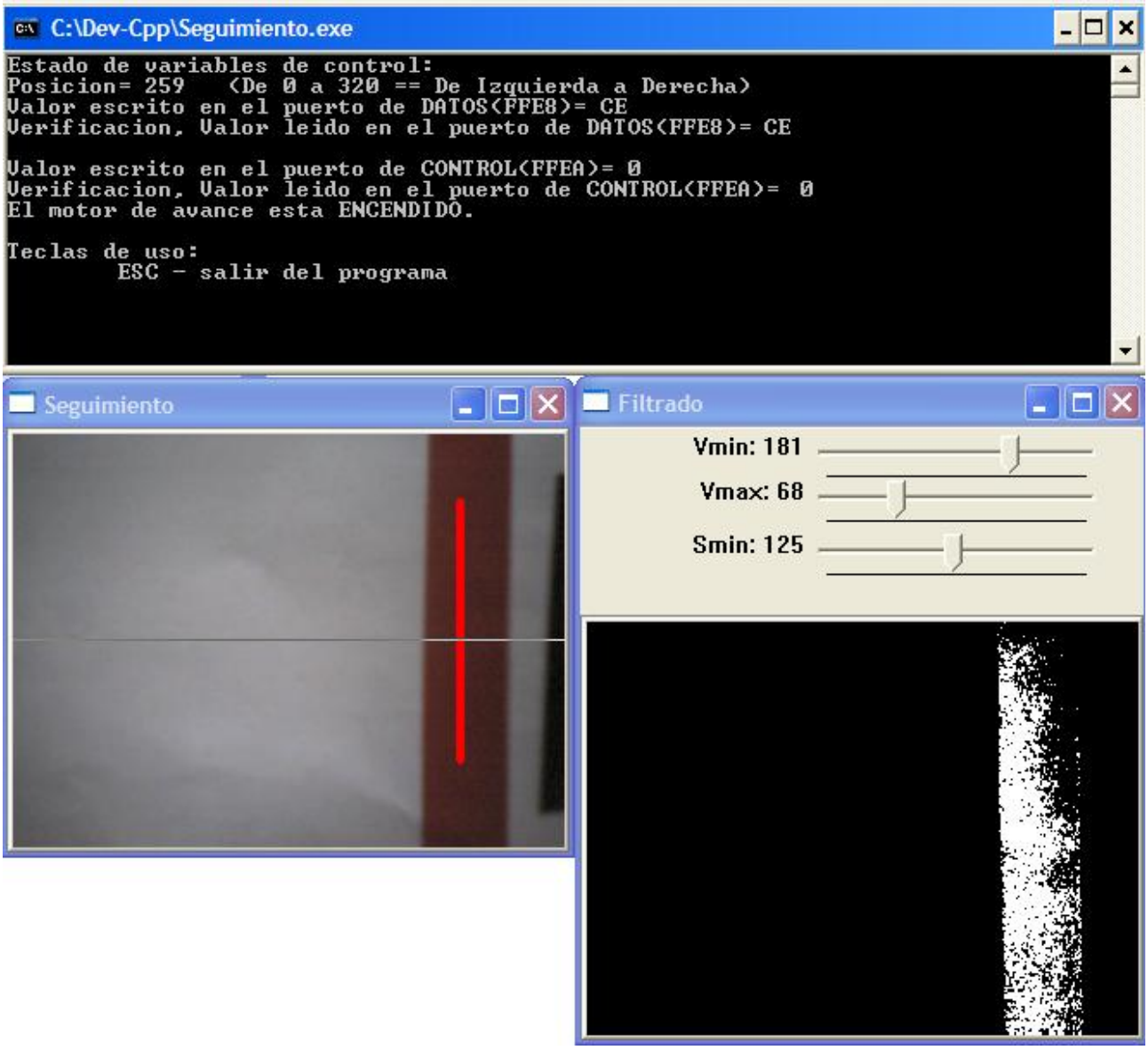


Figura 10. Línea detectada sobre la derecha.

Se observa que ahora la posición es 259 y el valor en el puerto de Datos es CE, lo que lógico ya que la línea se encuentra siendo detectada cerca del extremo derecho.

El valor en el puerto de Control sigue siendo 0, ya que la línea está siendo detectada.

A su vez esta figura sirve para demostrar cómo afecta el fenómeno descripto de la deficiencia de luz. Debido a como estaba posicionada la cámara con respecto a la fuente de luz, existía una sombra en el extremo derecho. Esto provoca que la densidad del objeto detectado disminuya poniendo en riesgo el correcto seguimiento del mismo.
Es por esto que el arreglo de leds es de vital importancia para evitar un mal funcionamiento frente a la posible aparición de sombras.

Por último, en la **Figura 11**, se observa el caso de cuando la línea no es detectada o un objeto extraño se posiciona sobre la misma.

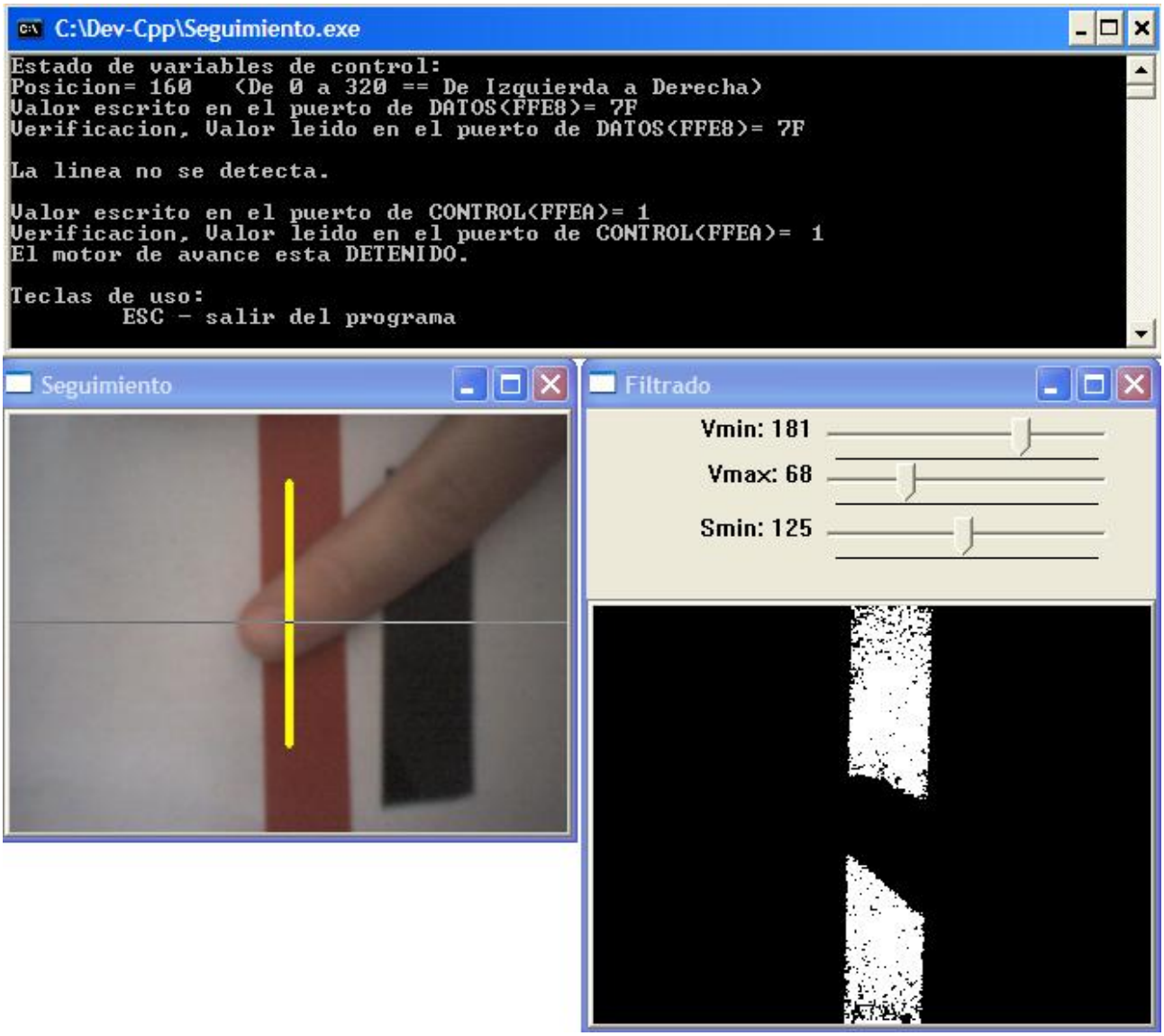


Figura 11. Línea no detectada.

En este caso el dedo de una persona está tapando la línea justamente sobre el área de detección.
Se puede ver en la ventana de 'Filtrado' que si los valores han sido ajustados de modo correcto, el dedo aparece como una sombra negra borrando prácticamente la línea de trayectoria. Recordar que la imagen de filtrado esta invertida verticalmente.
Si el dedo hubiera obstruído la línea en un sector fuera del área de detección, entonces la línea hubiera sido detectada.

Cuando la línea no es detectada, el color de la línea recta de detección cambia de color rojo a amarillo y queda centrada en el centro de la imagen. Además en la ventana de DOS, se puede ver que el valor en el puerto se fija en 7F, es decir la rueda vuelve a su posición central.
A su vez, se observa que se indica que la línea no está siendo detectada y que el motor de avance ha sido detenido. Esto último se puede corroborar ya que el valor en el puerto de Control ha cambiado a 1, deshabilitando es sistema de PWM.

Si el objeto que obstruye la línea es retirado, la línea será detectada nuevamente y el carro continuará con su recorrido inmediatamente.

Por último, la pantalla de DOS indica que para cerrar el programa se debe presionar la tecla Escape.

CONTROL PI POR SOFTWARE

En la sección anterior se menciono que en un principio cuando se ejecuta el programa se pide que la rueda este centrada y que para esto el potenciómetro que mide su posición debe estar igualmente centrado.
Luego de la primer demostración del carro en funcionamiento ante los titulares de la cátedra una de las criticas que surgieron fue que el funcionamiento del mismo debía admitir una desviación del potenciómetro por si el operador ejecutaba el programa sin darse cuenta que el potenciómetro estaba descentrado.
Como hasta aquí el valor que sale por el puerto paralelo es directamente proporcional a la posición detectada de la trayectoria, si se el potenciómetro se descentraba el funcionamiento del carro no era correcto. Esto hacia que siguiera la línea con un gran margen de error o incluso era incapaz de seguir la trayectoria perdiéndola de vista.
La solución a este problema se planteo mediante el agregado de una acción de control integral por software.
Para esto se definió la variable 'error' que mide la diferencia entre la posición de la línea detectada 'xpos' y el valor de referencia o 'bias', es decir el centro de la imagen que equivale al valor 320/2=160.
Por otro lado, para la acción integral se define la variable 'integral' cuya función es ir sumando de manera iterativa los errores acumulándolos en la misma.
La variable de salida se obtiene de la siguiente ecuación de control:

X = (bias+error+10*integral/ti)*255/320;

El 'bias' es la salida que tiene que dar el control cuando el error es cero.
Se observa que el bias, el error y el valor integral están multiplicados por la fracción 255/320, esta representa la constante proporcional.
Esta dada de esta manera ya que 255 = FF (hexa) es el máximo valor que se puede tener en el puerto paralelo, y dividida por 320 ya que este es el valor de la máxima excursión de detección.
A su vez, para poder variar el efecto de la acción integral, esta variable es dividida por el tiempo integral 'ti'. El valor del mismo puede ser fijado en una nueva barra deslizante que se agrega en la ventana de 'Seguimiento'. La barra muestra el valor de 'ti' multiplicado por 10 ya que solo permite mostrar números enteros. En la ecuación se observa que el 'ti' es dividido por 10 para obtener valores de 'ti' menores a la unidad y por lo tanto una acción integral más acentuada.

Error de división por cero

La barra de desplazamiento que permite fijar el valor del tiempo integral va de 0 a 100.
Se debe tener en cuenta que, como ya se menciono el valor mostrado esta multiplicado por 10, por lo tanto el verdadero valor del tiempo integral ira de 0.1 a 10.
Si en algún momento, durante la ejecución del programa, el operador coloca la barra en el extremo izquierdo, es decir ti=0, en la pantalla de DOS se observara el mensaje "Error de división por cero (se toma Ti=1)".
Se recuerda que la variable 'ti' está dividiendo al valor integral por lo tanto no puede ser cero.
Si el operario comete este error el programa automáticamente asignara al tiempo integral el valor unitario.

Reajuste excesivo (Wind up)

En control sucede un reajuste excesivo cuando el error es muy grande y la integral almacena un valor mayor al de saturación del controlador. Entonces cuando el error comienza a disminuir se tendrá un indeseado tiempo de retardo hasta que vuelva a compensarse.
Esto es debido a que la acción integral es muy fuerte y en definitiva es un tiempo durante el cual se pierde la acción de control.
Todos los controladores digitales poseen una función de ARW (anti reset wind up) para evitar este tiempo muerto, básicamente dejan de integrar cuando se llega a un valor máximo.
En nuestro caso el programa cuenta con una función de ARW, mediante la cual la variable 'integral' deja de integrar cuando alcanza un valor superior al modulo de 160*ti.
Este sale de tener en cuenta que si la acción integral alcanza un valor de ±160*ti indudablemente se producirá la saturación de la variable de salida.

Saturación del control digital

La saturación del control digital sucede cuando la variable de salida ya ha alcanzado los valores 0 o 255. Esto puede suceder ante la presencia de una curva muy pronunciada en donde el ángulo de giro de la dirección no es suficiente para disminuir el error, por lo que la acción integral aumenta indefinidamente llevando a saturación al control digital.
Cuando se produce la saturación ya sea porque se ha llegado a 0 o 255 aparece un mensaje en la pantalla de DOS indicando que esto ha sucedido, diciendo "Se ha llegado a la saturación del control digital".
Si aun con la saturación no se consigue disminuir el error y la línea se sigue desviando llegara un punto en que la misma se perderá por uno de los laterales.
Un instante previo a que esto suceda el programa detendrá el motor de avance e indicara en el DOS que "Aun con la saturación del control digital no es suficiente para seguir la trayectoria"

Margen de error de centrado del potenciómetro

Una vez incluido un control PI en el software se puede asegurar que el funcionamiento del carro será correcto siempre y cuando el potenciómetro no esté descentrado más de ±60º.
Esto se asegura teniendo en cuenta que un potenciómetro lineal tiene una libertad de movimiento de 270º. Considerando que el centro se encuentra en 135º, si se produce un error de 60º en el centrado tendrá aun 75º de libertad para girar en sentido contrario al que se produzco el descentrado.
Se considera que 75º son suficientes para lograr seguir una curva pronunciada.

En la **Figura 12** se puede observar la barra de desplazamiento que permite fijar el valor del tiempo integral como así también algunos de los mensajes de anteriormente mencionados.
Además se observa que, ente las variables de control que son impresas en pantalla ahora figuran también el Error, el valor Integral y la variable de salida X.

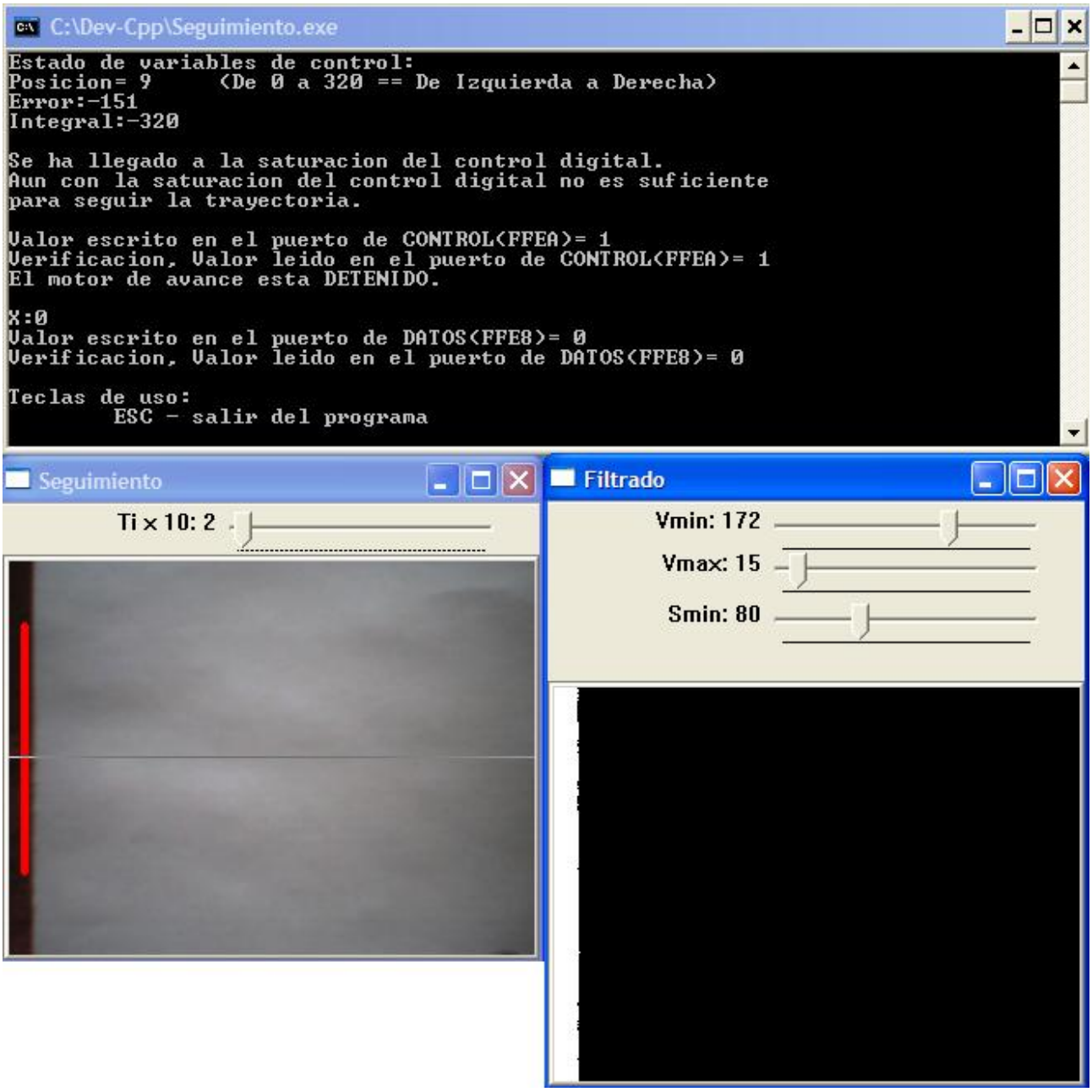


Figura 12. Variables y mensajes del agregado control PI por software.

COSTO DEL CARRO

Para tener una idea de la inversión que se llevó a cabo para solucionar este problema de ingeniería, a continuación se efectúa una lista con los materiales utilizados incluyendo sus costos para el cálculo del costo total.

Elemento	Costo [$ pesos]
Notebook HP 530	2400
Placa PCMCIA de puerto paralelo	150
Cámara Web Genius Messenger	45
2 x Motores de CC	100
Componentes electrónicos	100
Componentes de ferretería	50
Batería de auto	250
Costo total	3095

Cabe destacar que para la elección de la Notebook se tuvo en cuenta que la misma fuera de gama económica y que contara con puerto PCMCIA.
Los componentes electrónicos incluyen los transistores de potencia para el manejo de los motores, circuitos integrados, otros transistores, capacitores y resistencias.
Los componentes de ferretería incluyen madera, las rueditas, tuercas y tornillos.

MODELO FUNCIONANDO

La **Figura 13** es una foto del carro funcionando. También, si usted tiene acceso a internet, puede ver un video en YouTube.com del mismo en pleno funcionamiento.
El link es el siguiente: http://www.youtube.com/watch?v=ptbHn3seZ18

Figura 13. Modelo funcionando.

CONCLUSION

El presente trabajo nos ha permitido integrar una gran cantidad de conocimientos adquiridos durante nuestro cursado en la casa de estudio, desde los conocimientos más básicos hasta los más complejos e incluso desconocidos que nos vimos ante la necesidad de aprender para terminar este proyecto.
Fueron muchas las dificultades que surgieron a medida que el problema de ingeniería se fue resolviendo, donde para resolverlas tuvimos que hacer uso de nuestra capacidad de trabajo en grupo, aportando ideas, criticándolas y llegando así a las soluciones más adecuadas.

ANEXO 1 - Listado del programa

```
#ifdef _CH_
#pragma package <opencv>
#endif

#ifndef _EiC
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <ctype.h>
#include <conio.h>
#include <windows.h>
#endif

IplImage *image = 0, *hsv = 0, *hue = 0, *mask = 0, *backproject = 0;
CvHistogram *hist = 0;

int backproject_mode = 0;
int xizquierda, ancho, xpos, bias, error;
int integral = 0;
int select_object = 0;
int track_object = 0;
int show_hist = 1;
CvPoint origin, pt1, pt2;
CvRect selection;
CvRect track_window;
CvBox2D track_box;
CvConnectedComp track_comp;
int hdims = 16;
float hranges_arr[] = {0,180};
float* hranges = hranges_arr;
int vmin = 181, vmax = 68, smin = 125, ti = 2;
float kp = 255/320;
//para que detecte solo línea roja sin encandilamiento vmin = 181, vmax = 68, smin = 125;
//para que detecte solo elrojo vmin = 104, vmax = 126, smin = 114;
//para que detecte solo negro vmin = 0, vmax = 60, smin = 0;
typedef short _stdcall (*inpfuncPtr)(short portaddr);
typedef void _stdcall (*oupfuncPtr)(short portaddr, short datum);


CvScalar hsv2rgb( float hue )
{
    int rgb[3], p, sector;
    static const int sector_data[][3]=
        {{0,2,1}, {1,2,0}, {1,0,2}, {2,0,1}, {2,1,0}, {0,1,2}};
    hue *= 0.033333333333333333333333333333333f;
    sector = cvFloor(hue);
```

```
    p = cvRound(255*(hue - sector));
    p ^= sector & 1 ? 255 : 0;

    rgb[sector_data[sector][0]] = 255;
    rgb[sector_data[sector][1]] = 0;
    rgb[sector_data[sector][2]] = p;

    return cvScalar(rgb[2], rgb[1], rgb[0],0);
}

int main( int argc, char** argv )
{
    HINSTANCE hLib;
    inpfuncPtr inp32;
    oupfuncPtr oup32;

    short x,x2;
    int address, address2, c;
    int avance;

    //Carga la libreria
    hLib = LoadLibrary("inpout32.dll");

    if (hLib == NULL) {
        printf("LoadLibrary Failed.\n");
        return -1;
    }
//Obtiene la direccion de la funcion

    inp32 = (inpfuncPtr) GetProcAddress(hLib, "Inp32");

    if (inp32 == NULL) {
        printf("GetProcAddress for Inp32 Failed.\n");
        return -1;
    }


    oup32 = (oupfuncPtr) GetProcAddress(hLib, "Out32");

    if (oup32 == NULL) {
        printf("GetProcAddress for Oup32 Failed.\n");
        return -1;
    }

    CvCapture* capture = 0;
    capture = cvCaptureFromCAM(-1);
    cvNamedWindow( "Seguimiento", 1 );
    cvCreateTrackbar( "Ti x 10", "Seguimiento", &ti, 100, 0 );
    cvNamedWindow( "Filtrado", 1 );
    cvCreateTrackbar( "Vmin", "Filtrado", &vmin, 256, 0 );
```

```
cvCreateTrackbar( "Vmax", "Filtrado", &vmax, 256, 0 );
cvCreateTrackbar( "Smin", "Filtrado", &smin, 256, 0 );

system("cls");
printf("Antes de comenzar verifique que la rueda este centrada.\n");
printf("\nEstado de las variables de control:\n");
xpos = 160;
printf("Posicion = %i\t (La mitad de la escala)\n",xpos);
   address=0xffe8;
   x=xpos*255/320;
   (oup32)(address,x);
printf("Valor escrito en el puerto de DATOS(%X)= %X\t (La mitad de FF)\n" ,address,x);
   //Verificacion
x = (inp32)(address);
printf("Verificacion, Valor leido en el puerto de DATOS(%X)= %X\n",address,x);
printf("\nSi la rueda esta centrada presione ENTER.");
address2=0xffea;
x2=0x01;
(oup32)(address2,x2);
avance=-1;
for(;;)
{
c = cvWaitKey(10);
   if( c == 13 )
   break;
}

for(;;)
{
   IplImage* frame = 0;
   int i, bin_w, c, colorR, colorG, colorB;

   frame = cvQueryFrame( capture );
   if( !frame )
      break;

   if( !image )
   {

      image = cvCreateImage( cvGetSize(frame), 8, 3 );
      image->origin = frame->origin;
      hsv = cvCreateImage( cvGetSize(frame), 8, 3 );
      hue = cvCreateImage( cvGetSize(frame), 8, 1 );
      mask = cvCreateImage( cvGetSize(frame), 8, 1 );
      backproject = cvCreateImage( cvGetSize(frame), 8, 1 );
      hist = cvCreateHist( 1, &hdims, CV_HIST_ARRAY, &hranges, 1 );

   }

   cvCopy( frame, image, 0 );
```

```
cvCvtColor( image, hsv, CV_BGR2HSV );

if( track_object )
{
   int _vmin = vmin, _vmax = vmax;

   cvInRangeS( hsv, cvScalar(0,smin,MIN(_vmin,_vmax),0),
           cvScalar(180,256,MAX(_vmin,_vmax),0), mask );
   cvSplit( hsv, hue, 0, 0, 0 );

   if( track_object < 0 )
   {
      float max_val = 0.f;
      cvSetImageROI( hue, selection );
      cvSetImageROI( mask, selection );
      cvCalcHist( &hue, hist, 0, mask );
      cvGetMinMaxHistValue( hist, 0, &max_val, 0, 0 );
      cvConvertScale( hist->bins, hist->bins, max_val ? 255. / max_val : 0., 0 );
      cvResetImageROI( hue );
      cvResetImageROI( mask );
      track_window = selection;
      track_object = 1;
   }

   cvCalcBackProject( &hue, backproject, hist );
   cvAnd( backproject, mask, backproject, 0 );
   cvCamShift( backproject, track_window,
           cvTermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ),
           &track_comp, &track_box );

   track_window = track_comp.rect;
   xizquierda=track_window.x;
   ancho=track_window.width;
   xpos=xizquierda+(ancho)/2;
   pt1.x=xpos;
   pt1.y=50;
   pt2.x=xpos;
   pt2.y=200;


   if( backproject_mode )
      cvCvtColor( backproject, image, CV_GRAY2BGR );
   if( image->origin )
      track_box.angle = -track_box.angle;
   cvRectangle( image, pt1, pt2, CV_RGB(colorR,colorG,colorB), 3, CV_AA, 0 );

}

if( select_object && selection.width > 0 && selection.height > 0 )
{
```

```
        cvSetImageROI( image, selection );
        cvXorS( image, cvScalarAll(255), image, 0 );
        cvResetImageROI( image );
    }

    cvShowImage( "Seguimiento", image );
    cvShowImage( "Filtrado", mask );
    system("cls");


    select_object = 1;
    track_object= -1;
    selection.x = 0;
    selection.y = 120;
    selection.width = 320;
    selection.height =1;
    if ( ancho == 320 )
    //LA LINEA NO ES DETECTADA
    {
        colorR=255;
        colorG=255;
        colorB=0;
    }
    else
    //LA LINEA ES DETECTADA
    {
       colorR=255;
       colorG=0;
       colorB=0;
    }
    if (avance != 13)
    {
            avance = -1;
    }
    if( avance == -1 )
    {
    printf("\nAjuste los valores hasta que solamente se vea la linea de \ntrayectoria en la ventana
de Filtrado.\n");
    printf("Una vez que la linea se vea de forma correcta presione ENTER.\n");
    avance = cvWaitKey(10);
    }
    else
    {
    address=0xffe8;
    bias=160;
    error=xpos-bias;
    integral=integral+error;
    //ANTI RESET WIND UP. PARA EVITAR EL REAJUSTE EXCESIVO
    if (integral > 160*ti || integral < -160*ti)
    {
```

```
            if (integral > 160*ti)
            integral=160*ti;
            else
            integral=-160*ti;
      }
      //ERROR DE DIVISION POR CERO SI TI=0
      if (ti==0)
      {
      printf("Error de division por cero (se toma Ti=1)\n");
      x=(bias+error+integral)*255/320;
      }
      else
      {
      x=(bias+error+10*integral/ti)*255/320;
      }
      //IMPRESION DE LAS VARIABLES DE CONTROL
      printf( "Estado de variables de control:\n");
      printf("Posicion= %i\t(De 0 a 320 == De Izquierda a Derecha)\n",xpos);
      printf("Error:%i\n",error);
      printf("Integral:%i\n",integral);

      //SATURACION DEL CONTROL DIGITAL
      if (x>255 || x<0)
      {
            printf("\nSe ha llegado a la saturacion del control digital.\n");
            if (x>255)
            x=255;
            else
            x=0;
      if ( xpos!=320 && (xpos >= 310 || xpos <= 10))
      //SI LA LINEA SE FUE A UNO DE LOS EXTREMOS
      {
      printf("Aun con la saturacion del control digital no es suficiente\npara seguir la
trayectoria.\n");
      address2=0xffea;
      x2=0x01;
      (oup32)(address2,x2);
      printf("\nValor escrito en el puerto de CONTROL(%X)= %X\n" ,address2,x2);
      //Verificacion
      x2 = (inp32)(address2);
      printf("Verificacion, Valor leido en el puerto de CONTROL(%X)= %X\n",address2,x2);
      printf("El motor de avance esta DETENIDO.\n");
      }
      }

      printf("\nX:%i\n",x);
      (oup32)(address,x);
      printf("Valor escrito en el puerto de DATOS(%X)= %X\n" ,address,x);
      //Verificacion
      x = (inp32)(address);
```

```
        printf("Verificacion, Valor leido en el puerto de DATOS(%X)= %X\n",address,x);

        if ( ancho == 320 )
        //LA LINEA NO ES DETECTADA
        {
        printf("\nLa linea no se detecta.\n");
        address2=0xffea;
        x2=0x01;
        (oup32)(address2,x2);
        printf("\nValor escrito en el puerto de CONTROL(%X)= %X\n" ,address2,x2);
        //Verificacion
        x2 = (inp32)(address2);
        printf("Verificacion, Valor leido en el puerto de CONTROL(%X)= %X\n",address2,x2);
        printf("El motor de avance esta DETENIDO.\n");
        }
        else
        if (xpos>10&&xpos<310)
        //LA LINEA ES DETECTADA
        {
        address2=0xffea;
        x2=0x00;
        (oup32)(address2,x2);
        printf("\nValor escrito en el puerto de CONTROL(%X)= %X\n" ,address2,x2);
        //Verificacion
        x2 = (inp32)(address2);
        printf("Verificacion, Valor leido en el puerto de CONTROL(%X)= %X\n",address2,x2);
        printf("El motor de avance esta ENCENDIDO.\n");
        }
        printf( "\nTeclas de uso: \n"
        "\tESC - salir del programa\n");
}
        c = cvWaitKey(10);
        if( c == 27 )
        break;

}

    cvReleaseCapture( &capture );
    cvDestroyWindow("Seguimiento");
    FreeLibrary(hLib);
    return 0;
}
```