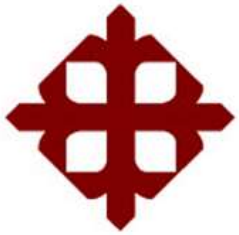


UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO
CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TEMA:

Implementación de algoritmos eficientes en la programación de un robot seguidor de línea utilizando métodos de control PID

AUTOR:
Vásquez Rojas, Pamela Michelle

Componente práctico del examen complexivo previo a la obtención del grado de **INGENIERA EN TELECOMUNICACIONES**

REVISOR:
M. Sc. Pacheco Bohórquez, Héctor Ignacio

Guayaquil, Ecuador

6 de marzo del 2020

UNIVERSIDAD CATÓLICA

DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

CERTIFICACIÓN

Certificamos que el presente **componente práctico del examen complexivo**, fue realizado en su totalidad por **Vásquez Rojas, Pamela Michelle** como requerimiento para la obtención del título de **INGENIERA EN TELECOMUNICACIONES**.

REVISOR

_____

M. Sc. Pacheco Bohórquez, Héctor Ignacio

DIRECTOR DE CARRERA

_____

M. Sc. Heras Sánchez, Miguel Armando

Guayaquil, 6 de marzo del 2020

UNIVERSIDAD CATÓLICA

DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Vásquez Rojas, Pamela Michelle**

DECLARÓ QUE:

El **componente práctico del examen complexivo: Implementación de algoritmos eficientes en la programación de un robot seguidor de línea utilizando métodos de control PID**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Guayaquil, 6 de marzo del 2020

EL AUTOR

_____

VÁSQUEZ ROJAS, PAMELA MICHELLE

UNIVERSIDAD CATÓLICA

DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Yo, **Vásquez Rojas, Pamela Michelle**

Autorizó a la Universidad Católica de Santiago de Guayaquil a la **publicación** en la biblioteca de la institución del **componente práctico del examen complexivo: Implementación de algoritmos eficientes en la programación de un robot seguidor de línea utilizando métodos de control PID** cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

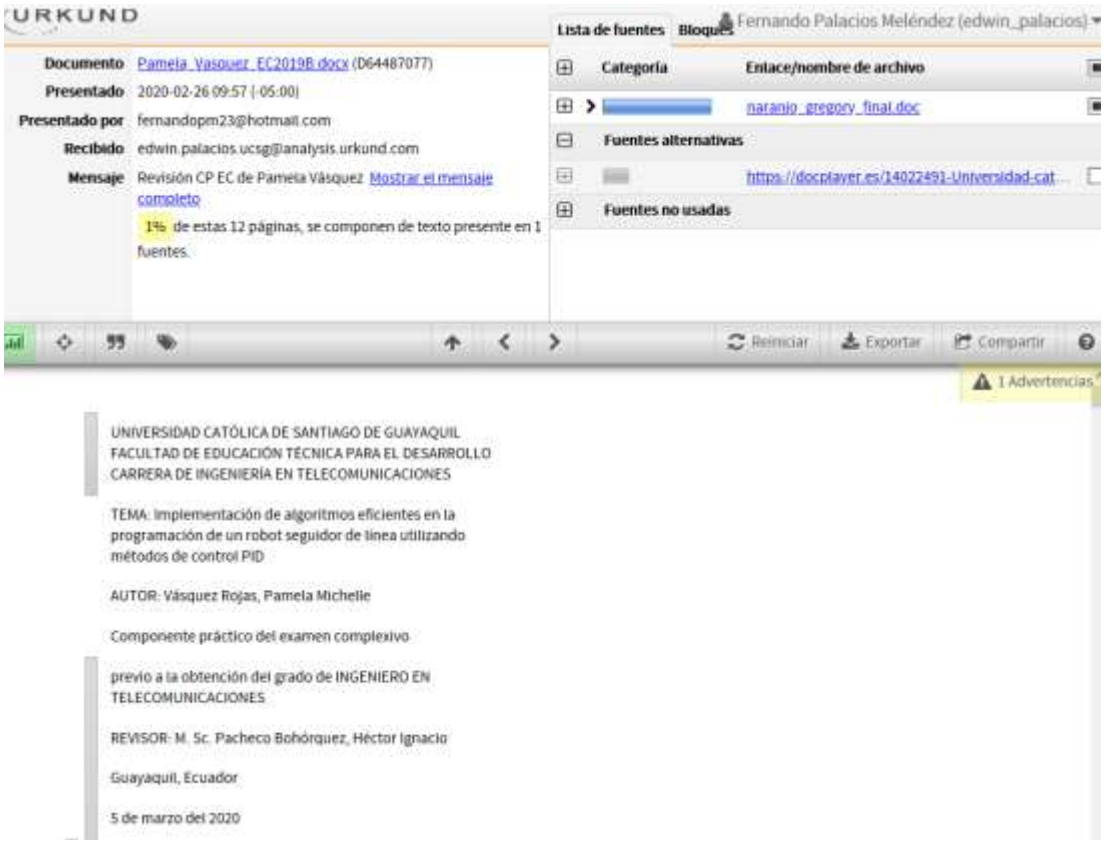
Guayaquil, 6 de marzo del 2020

EL AUTOR

_____

VÁSQUEZ ROJAS, PAMELA MICHELLE

REPORTE DE URKUND



DEDICATORIA

Este trabajo de titulación y todo el esfuerzo realizado se lo dedico con amor principalmente a los pilares de mi vida, mis papás, quienes siempre depositaron su confianza en mí y me motivaron a ser mejor persona con sus palabras de aliento y perseverancia.

A toda mi familia por el cariño incondicional de todos los días.

A mis amigos más cercanos quienes compartieron sus conocimientos y sentimientos a lo largo de esta trayectoria.

EL AUTOR

VÁSQUEZ ROJAS, PAMELA MICHELLE

AGRADECIMIENTO

A mis padres quienes me dieron la vida y cuyo apoyo fue una constante durante toda mi carrera profesional.

Al M. Sc. Edwin Palacios Meléndez quien supo guiarme en la trayectoria de este trabajo de titulación.

A mis amigos quienes me alegraron y me brindaron las mejores vibras.

Finalmente, a todas aquellas personas, colegas y amigos que me brindaron su apoyo, tiempo e información para el logro de mis objetivos.

EL AUTOR

VÁSQUEZ ROJAS, PAMELA MICHELLE

UNIVERSIDAD CATÓLICA

DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TRIBUNAL DE SUSTENTACIÓN

f._____

M. Sc. ROMERO PAZ, MANUEL DE JESÚS

DECANO

f._____

M. Sc. HERAS SÁNCHEZ, MIGUEL ARMANDO

DIRECTOR DE CARRERA

f._____

M. Sc. PALACIOS MELÉNDEZ, EDWIN FERNANDO

OPONENTE

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

Capítulo 2:

Capítulo 3:

Resumen

Este trabajo es el desarrollo del componente práctico del examen complexivo, y consistió en realizar una aplicación práctica en robots autónomos como es el caso del seguidor de línea, pero empleando métodos de control proporcional-integral-derivado llamado generalmente PID. El seguimiento de línea es uno de los aspectos más importantes de la robótica. Un robot seguidor de línea es un robot autónomo que puede seguir una línea negra o blanca dibujada en la superficie que consiste en un color contrastante. Este seguidor de línea está diseñado para moverse automáticamente y seguir la línea de trazado realizada. El control PID del seguidor de línea es un método que consiste en mejorar el movimiento del robot. El robot utiliza varios sensores para identificar la línea blanca, ayudando así al robot a permanecer en la pista. El seguidor de línea es impulsado por motores DC para controlar el movimiento de las ruedas. El microcontrolador Atmega se utilizó para realizar e implementar algoritmos PID para controlar la velocidad de los motores que conducen al robot a recorrer las líneas con suavidad. mediante el ajuste adecuado de los parámetros de control y así lograr un mejor rendimiento.

Palabras claves: ROBÓTICA, PID, MÉTODOS, CONTROLADOR, MICROCONTROLADORES, PARÁMETROS.

CAPÍTULO 1: DESCRIPCIÓN DEL COMPONENTE PRÁCTICO

1.1. Introducción.

Según Ortiz Martínez, (2016) el seguidor de línea es una máquina que puede seguir un camino. El trazado puede ser visible en una línea negra sobre una superficie blanca. Para Álvarez García, (2016) la sensación de una línea y la maniobra del robot para mantenerse en curso, mientras que constantemente corrige los movimientos incorrectos utilizando la retroalimentación del sensor forma un sistema simple pero eficaz. Se puede utilizar en automóviles, automatizaciones industriales, orientación, entre otras aplicaciones.

Para Pakdaman & Sanaatiyan, (2009) un robot seguidor de línea es una máquina móvil que puede detectar y seguir la línea dibujada en el suelo. En general, el camino está predefinido y puede ser visible como una línea negra sobre una superficie blanca con un color de alto contraste o puede ser invisible como un campo magnético.

Para muchos participantes que diseñan seguidores de línea, el seguidor de línea es un robot autónomo que detecta y sigue una línea dibujada en el suelo. Según Carrillo Romero et al., (2017) la mayoría de los robots seguidores de línea emplean controladores proporcional-integral-derivado (PID), que en si es un sistema auto operativo que detecta y sigue la pista dibujada en el piso.

La pista depende de los organizadores de torneos de robótica y puede ser una cinta negra sobre superficie blanca o viceversa. El sistema de control detecta la línea y maniobra el robot para mantenerse en curso mientras corrige constantemente los movimientos incorrectos utilizando el control PID, por lo que podría formar un sistema controlado efectivo. (Ferro Laspidea, 2014)

1.2. Planteamiento del problema:

La línea clásica que sigue al robot es una respuesta lenta al error que ocurrirá, dejará fácilmente su huella que se encuentra dibujada en el piso. Este

problema hará que el movimiento del robot no sea suave y, a veces, el robot tiende a moverse fuera de la pista. Aunque la línea que sigue al robot puede seguir la línea negra, su movimiento aún debe mejorarse, por lo que, para superar ese problema, se requiere un mejor controlador para que el robot siga la línea sin problemas y cometa menos errores. El movimiento del robot que sigue la línea se puede mejorar mediante el uso de un mecanismo de retroalimentación que forma un sistema eficaz de circuito cerrado. En este componente práctico del examen complexivo, se está empleando el controlador PID debido a su fácil implementación en robots autónomos.

1.3. Objetivo General.

Este proyecto se enfoca en diseñar algoritmos eficientes en la programación de un robot seguidor de línea utilizando métodos de control PID.

1.4. Objetivos Específicos.

a. Describir el estado del arte de publicaciones relacionadas a los robots seguidores de líneas.
b. Implementar algoritmos eficientes mediante métodos de control PID utilizando el compilador IDE Arduino.
c. Evaluar los resultados obtenidos empleando controladores PID y sin controlador PID inspeccionando el patrón de movimiento del robot mientras siguen la pista.

CAPÍTULO 2: Fundamentación teórica.

2.1. Visión general de robots seguidores de línea.

En estos últimos días, se han diseñado y utilizado muchos robots de seguimiento de línea. Esta línea que sigue a los robots normalmente consta de cuatro ruedas, dos ruedas o una sola rueda. La línea que sigue al robot fue diseñada para seguir una línea que puede ser un punto físico en el piso. El sensor se montó con el robot para comunicarse con el microcontrolador al detectar la línea dibujada por el trazador de línea en una superficie blanca o viceversa.

Varios trabajos relacionados a robots seguidores de líneas se encuentran disponibles en la nube, a través de repositorios y revistas arbitradas.

2.2. Funcionamiento básico de un seguidor de línea.

Las operaciones básicas de un seguidor de línea son las siguientes:

1. Capturar la posición de la línea con sensores ópticos montados en el extremo frontal del robot. Para esto, se ha utilizado una combinación de diodos emisores de luz (LED) infrarrojos (IR) y de fototransistores llamado acoplador óptico. Los requisitos para el proceso de detección de línea son alta resolución y robustez.
2. Guiar al robot requiere un mecanismo de dirección para el seguimiento. Se utilizan dos motores que gobiernan el movimiento de las ruedas para lograr esta tarea.

Este robot seguidor de línea dejará fácilmente su rastro de la línea negra dibujada en el piso porque es un sistema de circuito abierto. Este problema hará que el movimiento del robot se vuelva irregular. El robot que sigue la línea, aunque sigue la línea negra, su movimiento aún necesita ser mejorado. La aplicación del algoritmo PID digital puede suavizar el movimiento de seguimiento de línea negra o blanca. El controlador PID es un sistema de circuito cerrado, que proporcionará retroalimentación y corregirá el error que ocurre con una respuesta rápida.

2.3. Estado del arte de robots seguidores de líneas.

A continuación, se describen brevemente algunos proyectos relacionados a los robots seguidores de línea, dos sin emplear controladores PID y otros donde utilizan a PID para una mejoría en el rendimiento del robot.

Se encuentran tres trabajos publicados en revistas donde no utilizan controladores PID. Por ejemplo, citando a Galeano et al., (2014) donde se observa la implementación de un robot seguidor de línea básico utilizando Lego Mindstorms NXT 2.0. La aplicación es muy útil para abordar problemas antes de diseñar un prototipo más profesional. Igual manera, citando a Calderón et al., (2016) donde se observa la implementación de un seguidor de línea y también detección de obstáculos que los resultados mostrados son adecuados al momento de una competencia de velocidad. Finalmente, citando a Hasan et al., (2012) donde se observa que implementaron un robot seguidor de línea autónomo, el mismo que fue diseñado para seguir curvas muy cerradas, ya que los datos de los sensores son de naturaleza continua. Este robot es simple pero efectivo y tiene un diseño sencillo para realizar la tarea de seguimiento de línea.

En los siguientes dos trabajos se describen publicaciones relacionadas al empleo de controladores PID en robots seguidores de línea. Por ejemplo, citando a Gomes et al., (2016) se observa la implementación del controlador PID aplicado en un vehículo guiado automatizado *(Automated Guided Vehicle, AGV)* seguidor de línea usando una cámara RGB. Este artículo tiene como objetivo desarrollar un algoritmo de control para un AGV de tipo monociclo para seguir una trayectoria predefinida por una línea que evite la oscilación de sus movimientos. En esta aplicación, la cámara se utiliza en el circuito de control PID como un sensor para proporcionar información de retroalimentación, evaluando así la eficiencia del sistema de control aliado con la visión por computadora. Este tipo de aplicación simplifica el sistema de detección de un seguidor de línea AGV y hace posible utilizar la cámara para otros medios, como identificación de fichas y colores. Considerando un modelo simplificado y una técnica de control clásica, los resultados presentados han mostrado estabilidad en velocidades más bajas.

Ahora, citando a Proano & Gualsaqui, (2017) se observa que desarrollaron una simulación de un seguidor de línea mediante control difuso. Ellos desarrollaron un programa de Matlab para simular el robot seguidor de línea como una herramienta didáctica para aprender la teoría del control difuso. El entorno Simulink fue utilizado para modelar la dinámica del móvil y el controlador difuso, y se usa un Script para modelar la forma de la línea y la desviación del móvil de la línea. El simulador permite evaluar el rendimiento del control difuso proporcional-derivado (PD) ajustando los parámetros del controlador y la velocidad del móvil.

Se pudieran mencionar más trabajos del empleo de métodos de control PID aplicados a la robótica móvil en especial a los seguidores de línea.

2.4. Métodos de control en seguidores de línea.

En un sistema de control de procesos, la función del controlador es influir en el sistema de control a través de señales de control para que el valor de las variables controladas sea igual al valor de la referencia. El controlador puede ser conocido como el "cerebro" del sistema de control de procesos. El controlador genera una señal de control que se envía al elemento de control final dependiendo de la desviación entre el punto de ajuste y el valor medido de la variable controlada, tal como se muestra en la figura 2.1. El modo de controlador es la forma en que el controlador responde a la desviación. El sensor, el transmisor y las válvulas de control están ubicados cerca del proceso mismo, mientras el controlador está ubicado en el panel o reside como un programa dentro de la memoria de la computadora.

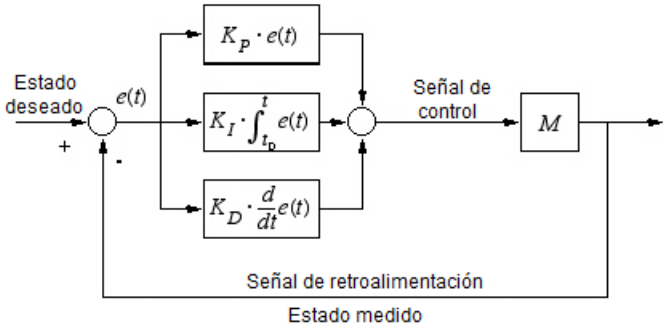


Figura 2. 1: Diagrama de bloques de la estructura general de un controlador.
Elaborado por: Autor

2.4.1. Teoría del controlador PID.

El control proporcional-integral-derivado (PID) es el algoritmo de control más común utilizado en la industria y ha sido aceptado universalmente. Los atributos de control de PID se deben en parte a su desempeño robusto en una amplia gama de condiciones de operación y en parte a su funcionalidad simple, permitiendo a los ingenieros operarlos de una manera simple. Como su nombre indica, un algoritmo PID consta de tres coeficientes básicos: proporcional, integral y derivado. Estas ganancias varían para lograr una respuesta óptima del sistema.

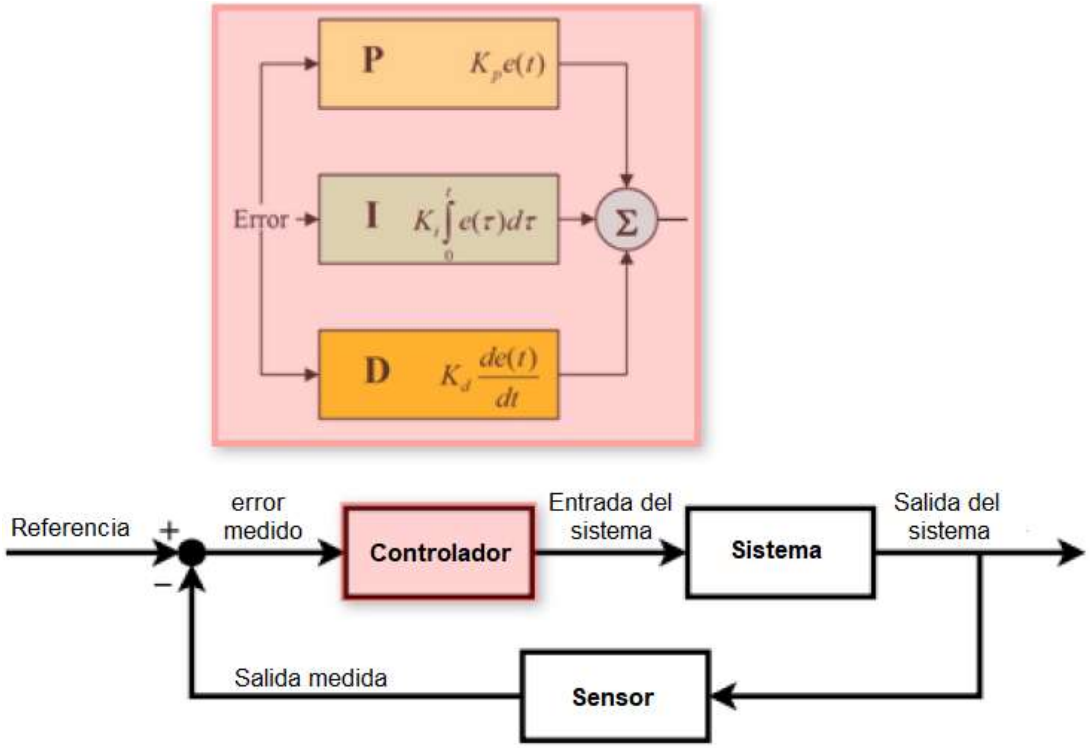


Figura 2. 2: Diagrama de bloques de la estructura general de un controlador PID.
Elaborado por: Autor

La estructura básica de un sistema con control PID implementado se muestra en la figura 2.2. La salida del sistema (también llamada variable de proceso) con un sensor y se compara con la lectura con el valor de referencia (también llamado punto de ajuste). La referencia y la salida medida se comparan y el resultado es un valor de error que se utiliza para calcular respuestas proporcionales, integrales y derivadas. Sumando las tres respuestas para obtener la salida del controlador. La salida del controlador se utiliza como una entrada al sistema que desea controlar, cambiando algunos aspectos del sistema. Por ejemplo, si se controla el motor, el controlador

proporcionaría más o menos corriente. Si se controla el flujo de un fluido, el controlador provocará que una válvula se abra o se cierre. La salida del sistema se mide y el proceso continúa. La iteración del bucle de control puede conocerse como la finalización del proceso en un solo momento.

La ecuación en función del tiempo del controlador PID es:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$

Después, se aplica la transformada de Laplace a la ecuación anterior del controlador PID:

$$L[u(t)] = L\left[K_p e(t)\right] + L\left[K_i \int_0^t e(\tau)d\tau\right] + L\left[K_d \frac{d}{dt} e(t)\right]$$

$$U(s) = K_p E(s) + \frac{1}{t_I s} E(s) + T_D s E(s)$$

$$U(s) = \left(K_p + \frac{K_p}{t_I s} + K_p T_D s\right) E(s)$$

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{t_I s} + T_D s\right)$$

Donde, K_p es la ganancia proporcional, K_i es la ganancia integral, K_d es la ganancia derivada, e es el error (Error = Punto de ajuste – Valor de proceso), t es el tiempo instantáneo, t_I es el tiempo integral, y t_D es el tiempo derivado.

2.4.2. Controlador proporcional

Se produce el término proporcional, en el que un valor de salida es proporcional al valor de error actual. La respuesta proporcional se ajusta multiplicando el error con una Kp constante, llamada constante de ganancia proporcional. El término proporcional se da como:

$$u(t) = K_p e(t)$$

o

$$\frac{U(s)}{E(s)} = K_p$$

Una ganancia proporcional alta que resulta en un gran cambio en la salida para un cambio dado del error. Si la ganancia proporcional es muy alta, el sistema se vuelve inestable. Mientras que, una ganancia pequeña da como resultado una respuesta de salida pequeña a un error de entrada grande y un controlador de baja respuesta o menos sensible. Si la ganancia proporcional es mucho menor, la acción de control puede ser demasiado pequeña al responder a las perturbaciones del sistema. La teoría de ajuste y la práctica industrial implican que el término proporcional debe contribuir a la mayor parte de la producción.

2.4.3. Controlador integral.

La contribución del término integral es proporcional tanto a la magnitud del error como a la duración del error. En un controlador PID, la integral es la suma del error instantáneo a lo largo del tiempo y proporciona el desplazamiento acumulado que debería haberse corregido previamente. El error acumulado se multiplica por la ganancia integral (Ki) y se agrega a la salida del controlador. El término integral viene dado por:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau$$

o

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{t_I s}\right)$$

El término integral acelera el movimiento del proceso hacia el punto de ajuste y elimina el error residual de estado estacionario que ocurre con un controlador proporcional puro. El término integral responde a los errores acumulados del pasado, puede dar como resultado que el valor presente sobrepase el valor del punto de ajuste.

2.4.4. Controlador proporcional-derivado.

El controlador PD se utiliza en el proceso donde el valor de compensación es aceptable, donde se puede considerar cierta anticipación y debe tener poco ruido. La ecuación del controlador PD es:

$$m(t) = \bar{m} + K_c e(t) + K_c \tau_D \frac{d}{dt} e(t)$$

La función de transferencia ideal del controlador PD viene dada por:

$$G_c(s) = \frac{M(s)}{E(s)} = K_c(1 + \tau_D s)$$

Mientras que la función de transferencia real se da como:

$$G_c(s) = \frac{M(s)}{E(s)} = K_c \left[\frac{(1+\alpha)\tau_D s + 1}{\alpha \tau_D s + 1} \right]$$

Capítulo 3: Implementación de un control PID para el robot velocista seguidor de línea.

3.1. Reseña del capitulo

Este capítulo se redactará el proceso que se realizó para la configuración de un control P.I.D. para el robot seguidor de línea velocista, con esto el rendimiento del robot puede aumentar considerablemente en las participaciones del club de robótica de la Facultad Técnica para el Desarrollo de la Universidad Católica de Santiago de Guayaquil.

3.2. Elementos utilizados para la implementación de un sistema de control PID para el robot seguidor de línea.

En esta sección del capítulo se detalla los elementos a utilizar y una breve explicación de estos para la implementación de un control PID del robot seguidor de línea.

3.2.1. Microcontrolador Atmega 32u4

El microcontrolador Atmega 32u4 de la empresa Atmel es un dispositivo electrónico que permite la ejecución de ordenes programadas mediante distintos programas desde el ordenador, este micro cuenta con varias características descritas a continuación:

.

- Interfaz I2C, SPI, UART/USART, USB.
- Puertos I/O: 26
- Velocidad 16MHz
- 32KB de memoria flash para programación
- Temporizadores: 5.
- Canales ADC: 12 de 10 bits.
- Tamaño de datos RAM 2.5KB
- 6 salidas PWM
- Empaquetador TQFP-44
- Voltaje de operación: 2.7v a 5.5v dc

Figura 3. 1: Microcontrolador Atmel Mega32U4.
Elaborado por: Autor.

El diagrama de salidas de los pines de este microcontrolador se muestra en la figura 3.2, según el datasheet del fabricante, esta es de gran utilidad a la hora de asignar tareas programadas.

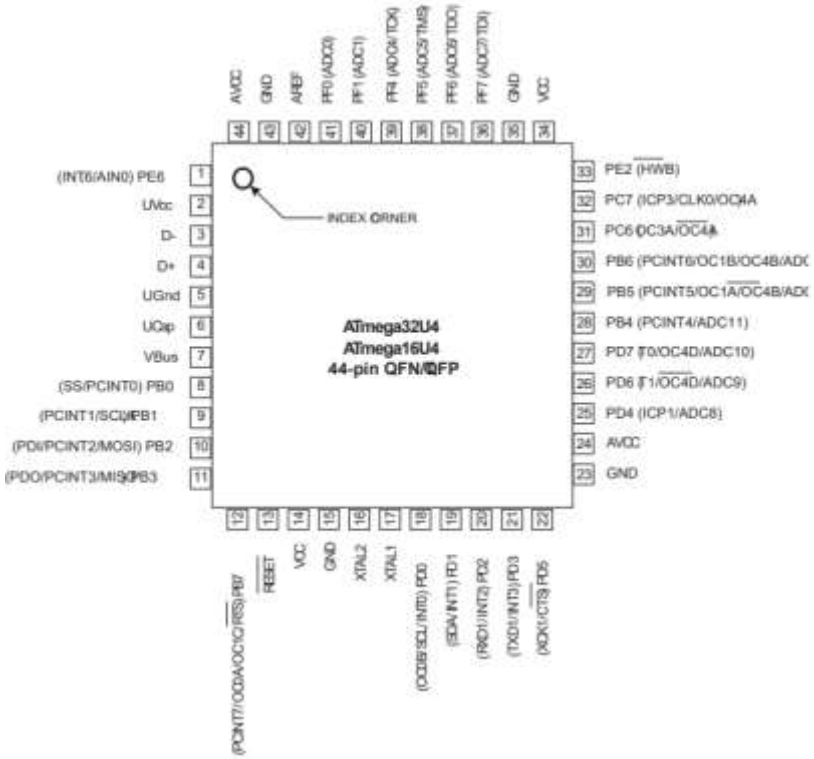


Figura 3. 2: Diagrama de pines de salida del microcontrolador Atmel Mega32U4.
Elaborado por: Autor.

La configuración de estas salidas puede varias según el programa que se utilizara para su programación, para el diseño del sistema PID de control del robot seguidor de línea se utilizara el ide de Arduino, este por su facilidad

de utilización es seleccionado para la implementación ya que facilita el aprendizaje y entendimiento a los estudiantes.

3.2.2. Motores force-up 4000RPM.

Los motores forcé-up de 4000 rpm, figura 3.3 son diseñados para un alto rendimiento alcanzando altas velocidades con un voltaje nominal de 12v dc, estos motores tienen un torque de 0.85kg-cm y con un peso de 28 gramos son prácticamente ideales para este tipo de prototipos. Entre sus características tenemos también:

- Tamaño 16x15mm.
- Corriente de operación de 990mA.
- Velocidad de 4000 RPM.
- Voltaje de operación de 12V DC.

Figura 3. 3: Motor Force-up.
Elaborado por: Autor.

3.2.3. Sensores reflectivos

Los sensores reflectivos son dispositivos activos que requieren de un emisor infrarrojo y un fototransistor de silicio NPN montado a la par montado en una estructura plástica de color negro como se aprecia en la figura 3.4. Entre sus características están:

- Voltaje de operación de 5v
- Corriente de operación 17mA
- Voltaje de salida de 0 a 5v, salida análoga.
- Distancia optima de censado 3mm
- Distancia máxima de censado 6mm

Figura 3. 4: Sensor QTR.
Elaborado por: Autor.

En la figura 3.5 se puede apreciar el esquemático que debe poseer este tipo de sensor para su perfecto funcionamiento, esta cuenta con dos resistencias para limitar la corriente en el diodo emisor y el fototransistor para proporcionar una salida analógica con valores de 0VDC a 5VDC.

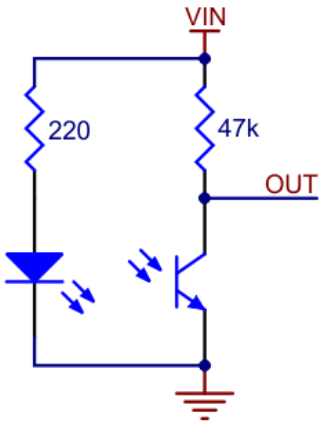


Figura 3. 5: Diagrama de sensor QTR.
Elaborado por: Autor.

3.2.4. Pantalla OLED 128x64

Esta pantalla será la que proporcione información al usuario para la correcta configuración del robot seguidor de línea, el estado de los sensores, entre otras cosas. Esta cuenta con una resolución de 128x64 puntos, su distancia en diagonal es de 1,3 pulgadas, para ser muy pequeña cuenta con una visualización muy buena gracias a su alto contraste como se muestra en la figura 3.6.

Figura 3. 6: Pantalla OLED.
Elaborado por: Autor.

Este tipo de pantalla posee un controlador (driver) interno, el SSD1306 que permite la comunicación por SPI o I2C al microcontrolador permitiendo la impresión de figuras o letras, internamente funciona con 3,3VDC y cuenta con un consumo promedio de 40mA. Entre las características que presenta este tipo de pantallas están:

- Dimensión de 35mmx35mm
- Grosor de 5mm
- Peso de 8,5g
- Área activa 29,420x14,70 mm
- Brillo (cd/m2): 100(Typ)

3.2.5. Sensor industrial fotoeléctrico M18.

El sensor industrial fotoeléctrico M18, figura 3.7, es de muy alta calidad y permite la detección de obstáculos en su área de trabajo, este permitirá al robot determinar si existe un impedimento para avanzar hacia adelante. Su señal de salida es digital de 5VDC cuando detecta un objeto a una distancia de hasta 80cm.

Los cables para alimentar y obtener su estado lógico son tres, para la conectividad a tierra o GND se representa por el color VERDE, con el color ROJO para su alimentación de hasta 5VDC y la salida de su estado está representada por el color AMARILLO. De acuerdo con su fabricante, la forma

de posicionar este sensor varía mucho los resultados obtenidos como se muestra en la figura 3.8, donde la distancia del resultado del sensor se ve afectada por su ángulo de posición.

Figura 3. 7: Sensor Fotoeléctrico 18M.
Elaborado por: Autor.

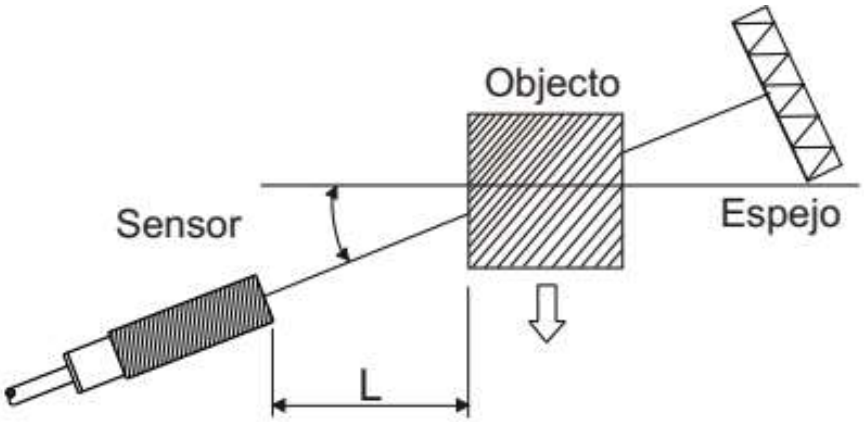


Figura 3. 8: Posición de sensor.
Elaborado por: Autor.

3.3. Diseño e implementación del control PID para un robot velocista seguidor de línea.

En esta sección del capítulo se describirá la implementación de control PID para un robot velocista utilizando el microcontrolador atmega 32U4 con el bootloader del Ide de Arduino.

3.3.1. Diseño de funcionamiento del sistema

El diagrama de bloque que se muestra en la figura 3,9 indica las entradas y salidas que tendrá el microcontrolador atmega 32u4, este contara con la

señal de los 8 sensores de piso, uno de obstáculos y el botón que permitirá el inicio de la secuencia programada para que el robot pueda trasladarse en la pista siguiendo la línea establecida

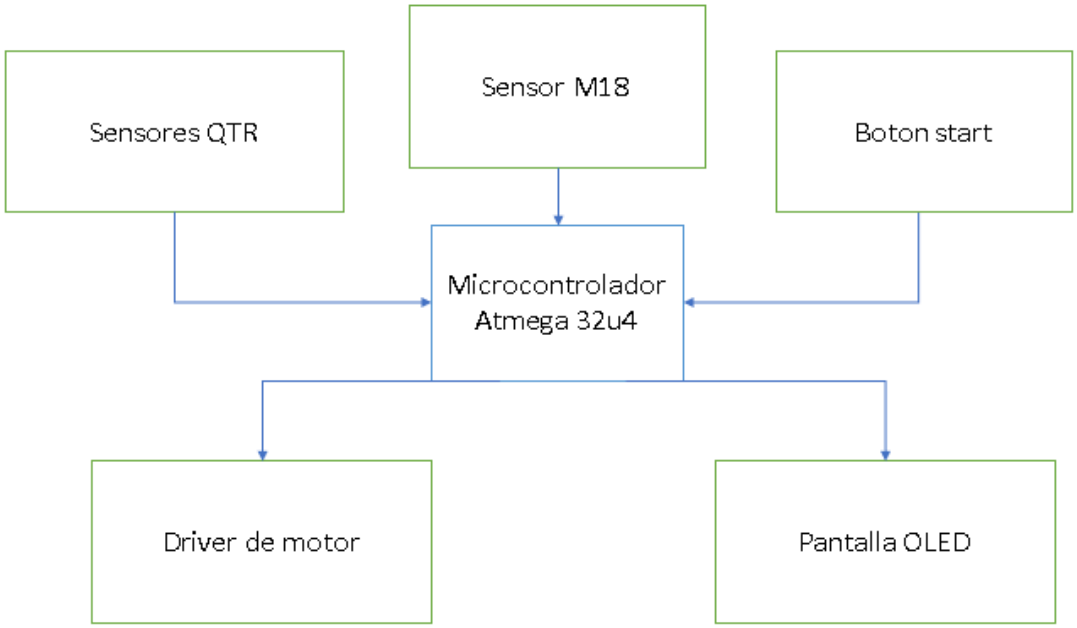


Figura 3. 9: Diagrama de bloque del sistema.
Elaborado por: Autor.

Los sensores QTR proporcionan una señal al microcontrolador atmega 32u4 tipo analógica para calibrar el contraste dependiendo de las condiciones de luz que se encuentre en el sitio donde deba realizar la secuencia, estos tendrán un valor de entre 0 a 5VDC. El sensor M18 y el botón de inicio (Start) proporcionara al microcontrolador una señal digital, correspondiente a 0VDC como un valor digital de LOW y de 5 VDC con un valor digital de HIGH.

En las salidas del microcontrolador están dadas por dos tipos, una para el control de los drivers que permitirán al robot conseguir la movilidad de acuerdo al control PID, este permitirá que el robot pueda corregir su error de acuerdo al resultado de los sensores utilizando una ecuación el diagrama de bloques que se muestra en la figura 3.10 y la segunda está definida para la interacción con el usuario en su pantalla de visualización donde se imprimirá las condiciones en las que se encuentra el robot.

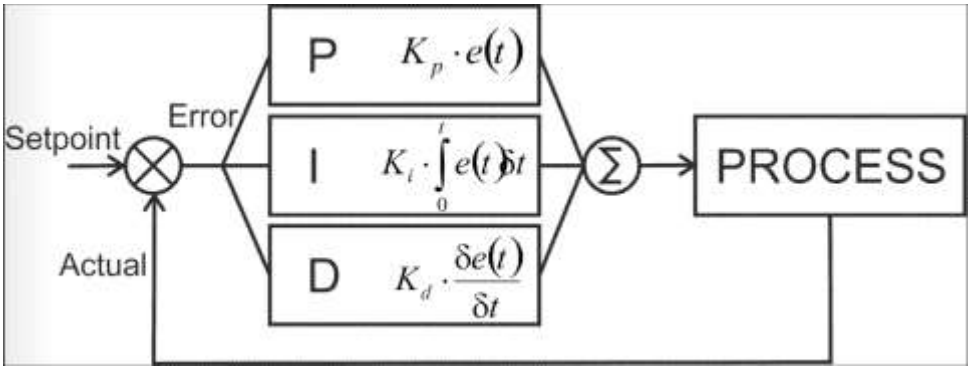


Figura 3. 10: Cuadro de control PID.
Elaborado por: Autor.

3.3.2. Diagrama de flujo de programación para el control PID del robot seguidor de línea.

En esta sección del documento se pretende hacer conocer al lector el diagrama de flujo establecido para el control PID del robot seguidor de línea, este deberá conocer los estados de los sensores y poder calcular el error en cada momento corrigiéndola, permitiendo al prototipo desplegar movimientos precisos.

En la primera parte del diagrama de flujo diseñado, figura 3.11, se muestra el inicio de las rutinas donde se debe establecer las librerías y definir los pines del microcontrolador utilizados que serán usados como entradas o salidas. Posteriormente se pretende mediante una sentencia de control FOR imprimir en la pantalla las advertencias en caso de tener un error al iniciar todo el sistema.

Una vez realizada la advertencia y establecer que el robot se encuentra en perfecto funcionamiento se deberá salir de esta sentencia para llegar a un WHILE que lo que permitirá tener un ciclo sin fin y donde se iniciará toda la se cuenta de control.

A continuación, al estar dentro del ciclo WHILE se pretende mostrar al usuario mediante la pantalla led la condición de los sensores de piso, estos si no tienen ningún tipo de anomalía el usuario establecerá el valor de la variable Okval como un TRUE como se puede ver en la primera parte del diagrama de la figura 3.12. Posteriormente se debe definir otro ciclo WHILE que permitirá

la espera del usuario para el arranque del prototipo en la pista definida por la competencia.

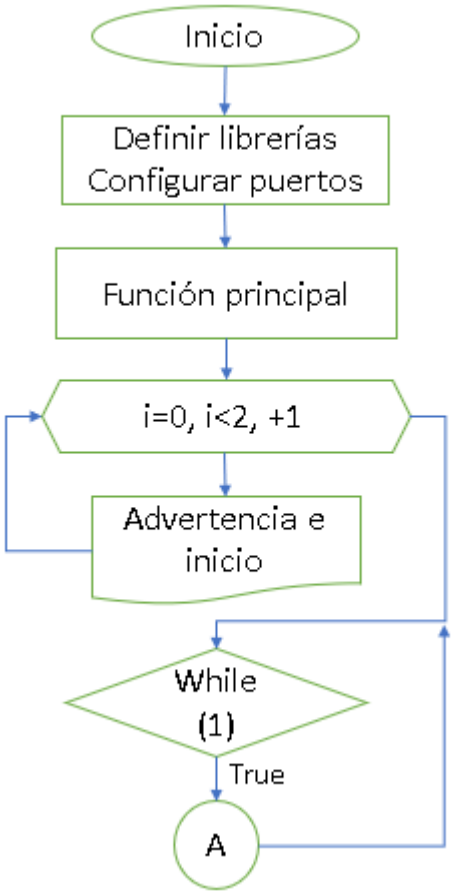


Figura 3. 11: Diagrama de flujo del inicio de rutinas.
Elaborado por: Autor

Si el usuario final presiona el botón Start, el flujo del programa permitirá ingresar a un tercer ciclo WHILE donde el robot testea los 8 sensores analógicos de piso y da el resultado para ser procesado en la función Start que permitirá el cálculo del control PID y corrección de errores.

En la figura 3.13 se puede apreciar la continuación del diagrama de flujo de programación, donde inicia con la lectura del botón Start en el que si se encuentra en un estado lógico digital LOW el robot estará inmóvil, caso contrario se da inicio al proceso del control PID y por consecuencia al movimiento del robot siguiendo la línea corrigiendo el error para un desplazamiento suave y preciso. Posteriormente finaliza la función Start y el

flujo del programa vuelve a iniciar con el primer WHILE establecido en la figura 3.11 anteriormente explicada.

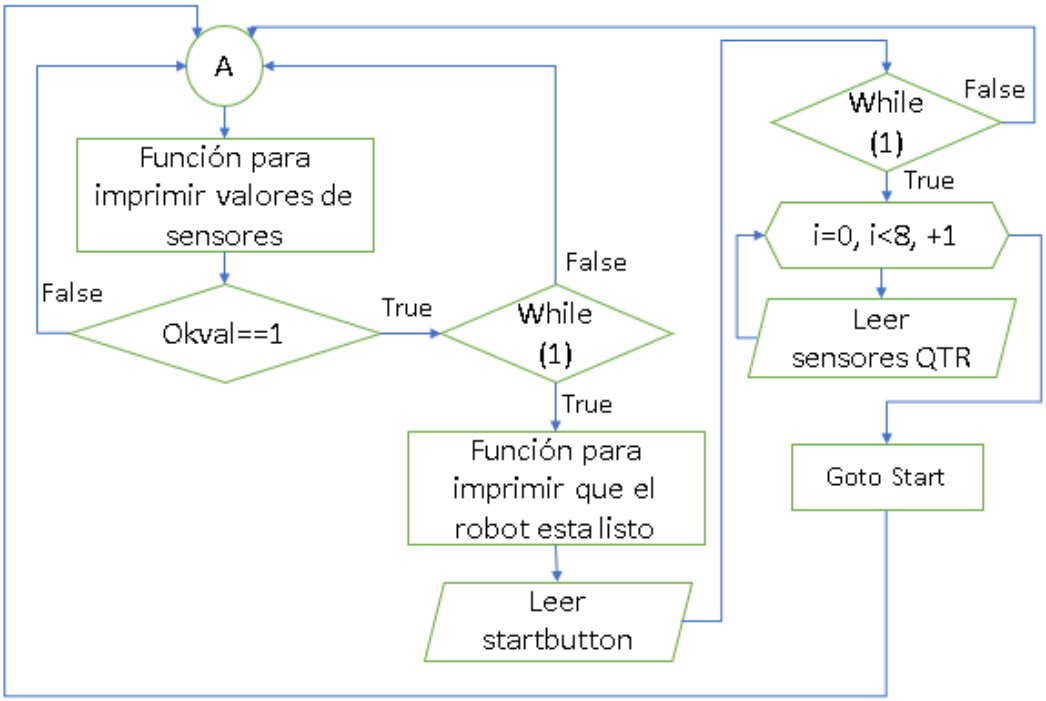


Figura 3. 12: Diagrama de bloques de función principal
Elaborado por: Autor

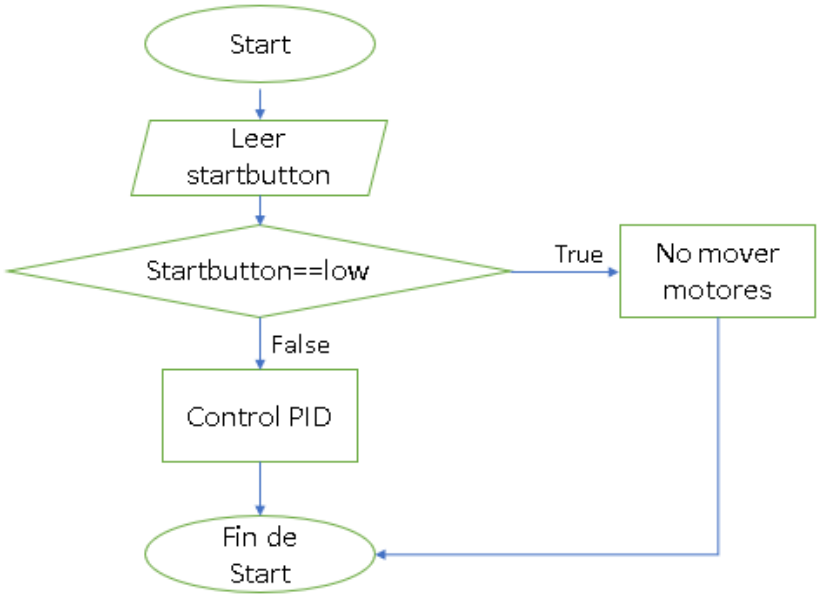


Figura 3. 13: Función Start
Elaborado por: Autor

3.4. Implementación del diseño del diagrama de flujo utilizando Ide de Arduino.

Posteriormente al tener claro el diagrama de flujo que se presentó en la sección anterior es preciso poder transformarlo a código de programación, para ello es necesario utilizar el ide de Arduino el cual permite poder programar el microcontrolador atmega 32u4 con muchas facilidades ya que este cuenta con varias librerías al alcance de cualquier persona.

Para empezar, es necesario como se indicó en la parte del diagrama de flujo definir las librerías a utilizar como también los puertos y variables, por la longitud del programa esta parte se presentará en dos secciones, la primera en la figura 3.14 donde se puede apreciar la llamada a la librería a utilizar como son Adafruit_SSD1306 y la definición de puertos como el reset para la pantalla OLED que utilizará el puerto 4 y los puertos 6, 5, 10 y 11 que pertenecen a los PWM para el control del driver de motor de cada lado.

```
#include <Adafruit_SSD1306.h>
#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);
#define Rpwm1 6
#define Rpwm2 5
#define Lpwm1 10
#define Lpwm2 11
#define ObjectSens 2
#define LQ1 A5
#define LQ2 A4
#define LQ3 A3
#define LQ4 A2
#define LQ5 A1
#define LQ6 A0
#define LQ7 A11
#define LQ8 A9
#define LQ9 1
#define Led1 14
#define Led2 16
#define StartButton 7
#define OkButton 8
```

Figura 3. 14: Llamada a la librería y definición de pines a utilizar.
Elaborado por: Autor

También se puede apreciar la definición de las entradas para los sensores de piso desplegados en la parte frontal del prototipo, estos puertos son A5, A4, A3, A2, A1, A0, A11 y A9 que son entradas analógicas. El puerto utilizado para el sensor fotoeléctrico descrito anteriormente en este

documento utilizará el puerto 1 ya que su señal es de tipo digital y no es necesario utilización de puertos analógicos del microcontrolador. Posteriormente se definen las variables que se utilizaran durante el proceso de la adquisición de información por parte de los sensores y también los de control PID como se puede apreciar en la figura 3.15.

```
int S1 = 0;
int S2 = 0;
int S3 = 0;
int S4 = 0;
int S5 = 0;
int S6 = 0;
int S7 = 0;
int S8 = 0;

float veri = 0;
float LeftBaseSpeed  = 75 ;
float RightBaseSpeed = 75 ;
float Kp = 0.07;
float Kd = 2.5;
float Ki = 0.000001 ;
```

Figura 3. 15: Definición de variables.
Elaborado por: Autor

Las variables para los 8 sensores serán de tipo entero mientras que las de control PID se definen como flotantes y esto se debe a que estas tendrán que corregir valores con decimales. En la figura 3.16 se aprecia un arreglo de variables LQ1 hasta el LQ8, como también los arreglos para valores máximos, mínimos y actuales de los sensores, un error final, error, y velocidades para los motores.

```
int Q[8] = {LQ1, LQ2, LQ3, LQ4, LQ5, LQ6, LQ7, LQ8};
int SensMin[8];
int SensMax[8];
int SensVal[8];
int SensValX[8];
int FinalError = 0;
int Error = 0;
int ExtraSpeed = 0;
int RightSpeed = 0;
int LeftSpeed = 0;
int Integral = 0;
int Val = 0;
int OkVal = 0;
unsigned int Position = 3500;

byte LineOk = 1;
byte WhiteLine = 0;
```

Figura 3. 16: Definición de variables restantes.
Elaborado por: Autor.

A continuación, en la figura 3.17 se presenta la función de Blink, que es una secuencia de animación de los leds integrados en la placa de control del robot seguidor de línea permitiendo dar un aviso al usuario de ciertas acciones que está realizando el microcontrolador.

```
void Blink() {
  for (int i = 0; i < 10; i++) {
    digitalWrite(Led1, LOW);
    digitalWrite(Led2, LOW);
    delay(180);
    digitalWrite(Led1, HIGH);
    digitalWrite(Led2, HIGH);
    delay(180);
  }
}
```

Figura 3. 17: Función Blink.
Elaborado por: Autor.

Esta función blink se da por un lazo for contable de 0 hasta 10 donde parpadean los leds correspondientes con un delay de 180 ms, una vez finalizado se termina el ciclo de la instrucción y puede realizar las demás ordenes programadas según sea necesario.

```
void MotorsSpeed(int RightSpeed, int LeftSpeed) {
  if (RightSpeed <= 0) {
    RightSpeed = abs(RightSpeed);
    digitalWrite(Rpwm1, LOW);
    analogWrite(Rpwm2, RightSpeed);
  }
  else {
    digitalWrite(Rpwm2, LOW);
    analogWrite(Rpwm1, RightSpeed);
  }
  if (LeftSpeed <= 0) {
    LeftSpeed = abs(LeftSpeed);
    digitalWrite(Lpwm1, LOW);
    analogWrite(Lpwm2, LeftSpeed);
  }
  else {
    digitalWrite(Lpwm2, LOW);
    analogWrite(Lpwm1, LeftSpeed);
  }
}
```

Figura 3. 18: Función MotorsSpeed.
Elaborado por: Autor.

En la figura 3.18 se muestra la función que permitirá realizar los movimientos de los motores a distintas velocidades de acuerdo con los valores de las variables enteras RightSpeed y LeftSpeed, con esto se asegura a que los valores del PWM utilizados sean los adecuados y pueda el robot responder ante el control PID y sus correcciones en tiempo real.

A continuación, en la imagen 3.19 se muestra el conjunto de instrucción que forman parte de la función de adquisición de datos por parte de los sensores analógicos QTR ubicados en la parte frontal inferior del prototipo. Para ello es necesario dos lazos FOR que mediante el arreglo que se presentó anteriormente se puede obtener uno a uno los valores de los sensores y mediante las operaciones apreciadas es posible devolver un valor numérico para cálculos del PID.

```
int ReadSensor() {
  unsigned long Middle = 0;
  unsigned int Total = 0;
  LineOk = 0;
  for (int i = 0; i < 8; i++) {
    int Difference = SensMax[i] - SensMin[i];
    long int Calculation = analogRead(Q[i]) - SensMin[i];
    Calculation = Calculation * 1000;
    Calculation = Calculation / Difference;
    SensVal[i] = constrain(Calculation, 0, 1000);
    SensValX[i] = SensVal[i];
  }
  for (int i = 0; i < 8; i++) {
    if (WhiteLine == 1)  SensValX[i] = 1000 - SensVal[i];
    if (SensValX[i] > 250)  LineOk = 1;
    if (SensValX[i] > 50) {
      Middle += SensValX[i] * i ;
      Total += SensValX[i];
    }
  }
  if (LineOk == 0) {
    if (Val < 3000)  return 0;
    if (Val > 5000)  return 7000;
    if (Val > 3000 && Val < 5000) return 3500;
  }
  Middle = Middle * 1000;
  Val = Middle / Total;
  return Val;
}
```

Figura 3. 19: Función ReadSensor.
Elaborado por: Autor.

Después de haber definido estas funciones es necesario configurar los puertos utilizados por el microcontrolador, esto se realiza con la función void Setup el cual permite definir como entradas o salidas los pines del microcontrolador, figura 3.20, Además de asegurarse que los motores no tengan ningún tipo de movimiento llamando a la función de MotorSpeed explicado anteriormente, también dar un aviso en pantalla al usuario de bienvenida utilizando las sentencias de la librería Display establecida anteriormente en el encabezado del código.

```
void setup() {
  pinMode(Rpwm1, OUTPUT);
  pinMode(Rpwm2, OUTPUT);
  pinMode(Lpwm1, OUTPUT);
  pinMode(Lpwm2, OUTPUT);
  pinMode(LQ1, INPUT);
  pinMode(LQ2, INPUT);
  pinMode(LQ3, INPUT);
  pinMode(LQ4, INPUT);
  pinMode(LQ5, INPUT);
  pinMode(LQ6, INPUT);
  pinMode(LQ7, INPUT);
  pinMode(LQ8, INPUT);
  pinMode(LQ9, INPUT);
  pinMode(ObjectSens, INPUT_PULLUP);
  pinMode(StartButton, INPUT_PULLUP);
  pinMode(OkButton, INPUT);
  pinMode(Led1, OUTPUT);
  pinMode(Led2, OUTPUT);
  MotorsSpeed(0, 0);
  Serial.begin(9600);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.setTextColor(WHITE);
  display.clearDisplay();
  drawBitMap(44, 8);
  display.display();
  delay(1500);
  display.clearDisplay();
  display.setTextSize(2);
  display.setCursor(12, 15);
  display.println(F("LineCraft"));
  display.display();
  delay(1500);
}
```

Figura 3. 20: Configuración de pines utilizados en el microcontrolador.
Elaborado por: Autor.

Finalizada esta sección se define una función para la muestra del resultado del test en la pantalla OLED que debe hacer los sensores previos a

realizar las acciones que permiten el robot moverse a lo largo de la pista, esta función, figura 3.21, será de gran ayuda para el usuario el cual determinará que todos los sensores de piso estén en óptimas condiciones y no encontrará algún tipo de anomalía para la ejecución del código en el microcontrolador. Ya que esta función es muy extensa se la presenta en varias partes.

```
void SensScreen() {
 S1 = analogRead(A5);
  S2 = analogRead(A4);
  S3 = analogRead(A3);
  S4 = analogRead(A2);
  S5 = analogRead(A1);
  S6 = analogRead(A0);
  S7 = analogRead(A11);
  S8 = analogRead(A9);

  S1 = map(S1, 0, 1023, 1, 26);
  S2 = map(S2, 0, 1023, 1, 26);
  S3 = map(S3, 0, 1023, 1, 26);
  S4 = map(S4, 0, 1023, 1, 26);
  S5 = map(S5, 0, 1023, 1, 26);
  S6 = map(S6, 0, 1023, 1, 26);
  S7 = map(S7, 0, 1023, 1, 26);
  S8 = map(S8, 0, 1023, 1, 26);
```

Figura 3. 21: Función SensScreen.
Elaborado por: Autor.

La primera parte de esta función tiene como objetivo obtener los valores de los puertos analógicos recibidos de parte de los sensores, una vez finalizado se los convierte a valores de 1 al 26 con la instrucción map y se los asigna a las variables correspondientes.

Una vez finalizada la obtención de datos, se procede a establecer el marco donde se presentarán los valores obtenidos en barras de progreso en tiempo real. En la figura 3.22 se puede apreciar la primera parte del código con una instrucción de cleanDisplay el cual realizara la acción de limpieza de la pantalla para evitar mostrar valores no actualizados, posterior a ello se define el nombre de cada una de las barras de progreso que serán S1 al S8, y para finalizar se hace el marco que contendrá la barra con un color blanco.

```
display.clearDisplay();
display.setTextSize(1);
display.setCursor(2, 1 );
display.println(F("S1"));
display.setCursor(18, 1 );
display.println(F("S2"));
display.setCursor(34, 1 );
display.println(F("S3"));
display.setCursor(50, 1 );
display.println(F("S4"));
display.setCursor(66, 1 );
display.println(F("S5"));
display.setCursor(82, 1 );
display.println(F("S6"));
display.setCursor(98, 1 );
display.println(F("S7"));
display.setCursor(114, 1 );
display.println(F("S8"));
display.setTextSize(2);
display.drawRect(2, 8, 10, 26, WHITE);
display.drawRect(18, 8, 10, 26, WHITE);
display.drawRect(34, 8, 10, 26, WHITE);
display.drawRect(50, 8, 10, 26, WHITE);
display.drawRect(66, 8, 10, 26, WHITE);
display.drawRect(82, 8, 10, 26, WHITE);
display.drawRect(98, 8, 10, 26, WHITE);
display.drawRect(114, 8, 10, 26, WHITE);
```

Figura 3. 22: Función SensScreen dibujo de marco.
Elaborado por: Autor.

En la figura 3.23 se muestra la continuación del código de la función, para iniciar se requiere evaluar el estado del valor del botón de ok para la confirmación del usuario al ver el estado de los sensores, mediante las sentencias IF se evalúa el valor de cada una de las variables y dibuja en la pantalla una línea a continuación de la anterior para aumentar o disminuir según los resultados de la prueba. Cuando el usuario este de acuerdo con el presionar el botón de ok se da paso al cambio de la variable OkVal que permitirá la continuación del flujo del programa escrito.

En la figura 3.24 y 3.25 se muestra al lector la continuación del código de la función con la misma mecánica.

```
if (digitalRead(OkButton) == LOW) {
  OkVal = 1;
} else {
  OkVal = 0;
}
for (int a = 26; a >= S1; a--) {

  display.setCursor(2, a);
  display.println(F("-"));
  if (digitalRead(OkButton) == LOW) {
    OkVal = 1;
  } else {
    OkVal = 0;
  }
}
for (int b = 26; b >= S2; b--) {

  display.setCursor(18, b);
  display.println(F("-"));

  if (digitalRead(OkButton) == LOW) {
    OkVal = 1;
  } else {
    OkVal = 0;
  }
}
```

Figura 3. 23: Evaluación de valores de sensores para graficar en OLED.
Elaborado por: Autor.

```
for (int c = 26; c >= S3; c--) {

  display.setCursor(34, c);
  display.println(F("-"));
  if (digitalRead(OkButton) == LOW) {
    OkVal = 1;
  } else {
    OkVal = 0;
  }
}
for (int d = 26; d >= S4; d--) {

  display.setCursor(50, d);
  display.println(F("-"));
  if (digitalRead(OkButton) == LOW) {
    OkVal = 1;
  } else {
    OkVal = 0;
  }
}
for (int e = 26; e >= S5; e--) {

  display.setCursor(66, e);
  display.println(F("-"));
  if (digitalRead(OkButton) == LOW) {
    OkVal = 1;
  } else {
    OkVal = 0;
  }
```

Figura 3. 24: Evaluación de valores de sensores.
Elaborado por: Autor.

```
  for (int f = 26; f >= S6; f--) {

    display.setCursor(82, f);
    display.println(F("-"));
    if (digitalRead(OkButton) == LOW) {
      OkVal = 1;
    } else {
      OkVal = 0;
    }
  }
  for (int g = 26; g >= S7; g--) {

    display.setCursor(98, g);
    display.println(F("-"));
    if (digitalRead(OkButton) == LOW) {
      OkVal = 1;
    } else {
      OkVal = 0;
    }
  }
  for (int h = 26; h >= S8; h--) {

    display.setCursor(114, h);
    display.println(F("-"));
    if (digitalRead(OkButton) == LOW) {
      OkVal = 1;
    } else {
      OkVal = 0;
    }
```

Figura 3. 25: Finalización de la función SensScreen.
Elaborado por: Autor.

Para finalizar la descripción del código de programación descrito se procede a mostrar mediante la figura 3.26 el inicio del código de la función principal el cual se inicia la presentación de la configuración del robot.

```
void loop() {
  display.clearDisplay();
  for (int i = 0; i < 2; i++) {
    display.setTextSize(1);
    display.setCursor(40, 0);
    display.println(F(" Warning!!! "));
    display.setCursor(0, 9);
    display.println(F("press the (OK) button "));
    display.setCursor(15, 17);
    display.println(F("   to exit the"));
    display.setCursor(14, 25);
    display.println(F(" test screen "));
    display.display();

    delay(3000);

  }
   display.clearDisplay();
       display.display();
       delay(200);
```

Figura 3. 26: Función principal encabezado.
Elaborado por: Autor.

Una vez finalizada esta sección se da paso a la primera instrucción WHILE como se puede apreciar en la figura 3.27, esto cumpliendo con el diagrama de flujo presentado anteriormente en este documento.

```
while (1) {
  SensScreen();
  if (OkVal == 1) {
    while (1) {
      display.setTextSize(1);
      display.clearDisplay();
      display.display();
      display.setCursor(40, 0);
      display.println(F(" Warning!!! "));
      display.setCursor(8, 8);
      display.println(F(" Show your hand to "));
      display.setCursor(7, 16);
      display.println(F("   MZ80 Sensor "));
      display.setCursor(8, 24);
      display.println(F("    to Start!! "));
      display.display();
      for (int i = 0; i < 8; i++) {
        SensMin[i] = 1024;
        SensMax[i] = 0;
      }
      delay(200);
      if (digitalRead(StartButton) == LOW) {
        display.clearDisplay();
        display.display();
         display.setCursor(8, 17);
      display.println(F(" Need to calibrate!"));
                display.display();
                          Blink();
      delay(200);
```

Figura 3. 27: Función principal, evaluación de variable OkVal y primer While.
Elaborado por: Autor.

Al iniciar el primer WHILE se llama a la función de evaluación de los sensores en pantalla llamado SensScreen descrito anteriormente, cuando el valor de la variable globar OkVal es verdadero es posible seguir el flujo del programa introduciendo a otra sentencia WHILE que permite al microcontrolador escribir en pantalla lo necesario para empezar. Si el botón de Start está en bajo la pantalla mostrara el mensaje que es necesario calibrar al finalizar la asignación de valores en el arreglo existente SensMax y SensMin.

En la figura 3.28 se puede apreciar la continuación del código en el que después del despliegue de los mensajes en la pantalla se realiza la asignación de valores pertenecientes a los cambios de contraste en los arreglos

anteriormente descritos. Con ello se pretende dar aviso al programa cual es el centro del robot y que valores deben ser tomados en cuenta cuando se realice el cálculo del error en el control PID.

```
        while (1) {
  for (int i = 0; i < 8; i++) {
  SensMin[i] = 1024;
  SensMax[i] = 0;
  delay(1);
}
for (int i = 0; i < 150; i++)
{
  for (int i = 0; i < 8; i++) {
    if (SensMin[i] > analogRead(Q[i])) SensMin[i] = analogRead(Q[i]);
    if (SensMax[i] < analogRead(Q[i])) SensMax[i] = analogRead(Q[i]);
    delay(1);
  }
}
delay(2000);
      delay(400);
                Blink();
                display.clearDisplay();
        display.display();
        delay(200);
         display.setCursor(0, 10);
      display.println(F("Bring the robot to          Center "));
                display.display();
      delay(5000);
         Wire.endTransmission();
                          display.clearDisplay();
        display.display();
        delay(200);
         Wire.endTransmission();
```

Figura 3. 28: Función principal definición de valores en arreglos.
Elaborado por: Autor.

Para finalizar, la sección Start del código permite el cálculo del error de acuerdo con el control PID establecido, en la figura 3.29 se puede apreciar la evaluación del puerto StartButton donde su resultado llevara a las acciones correspondientes. El control PID esta dado el cálculo del error donde:

Error = Position - 3500

Y la verificación final será:

Verificación = kp * Error + kd * (Error - FinalError) + ki * Integral

Se demuestra que existe una retroalimentación en el control PID según el código propuesto para este prototipo, posterior a ello las velocidades de los motores están definidas de acuerdo con el resultado de estos cálculos.

```
Start:
while(1){

        if (digitalRead(StartButton) == LOW) {
            MotorsSpeed(0, 0);
            delay(15);
        }else{
            Position = ReadSensor();
            Error = Position - 3500;
            Integral += Error;

            int Verify = Kp * Error + Kd * (Error - FinalError) + Ki * Integral;
            FinalError = Error;

            RightSpeed = RightBaseSpeed + Verify + ExtraSpeed ;
            LeftSpeed = LeftBaseSpeed - Verify +  ExtraSpeed ;

            RightSpeed = constrain(RightSpeed, -100, 100);
            LeftSpeed = constrain(LeftSpeed, -100, 100);
            MotorsSpeed(LeftSpeed, RightSpeed);
        }

}
   goto Start;
          }
        }

      }
    }
```

Figura 3. 29: Función principal cálculo de control PID.
Elaborado por: Autor.

3.5. Resultados obtenidos del robot seguidor de línea velocista.

En esta última sección de este documento se presenta al lector los resultados del algoritmo construido para el sistema de control PID para el robot seguidor de línea velocista. En la figura 3.30 se muestra el robot seguidor de línea con sus partes electrónicas montadas y motores descritos anteriormente.

A continuación, se muestra en la figura 3.31 los sensores montados en la parte frontal inferior del prototipo seguidor de línea y en la figura 3.32 .se muestra el sensor fotoeléctrico M18 montado en la parte frontal del robot.

Posterior al respectivo montaje se procedió a la programación del microcontrolador con el código presentado anteriormente y se realizó las pruebas del prototipo para observar su eficiencia con resultados satisfactorios, en la figura 3.33 y 3.34 se muestra las visualizaciones en la pantalla OLED del robot.

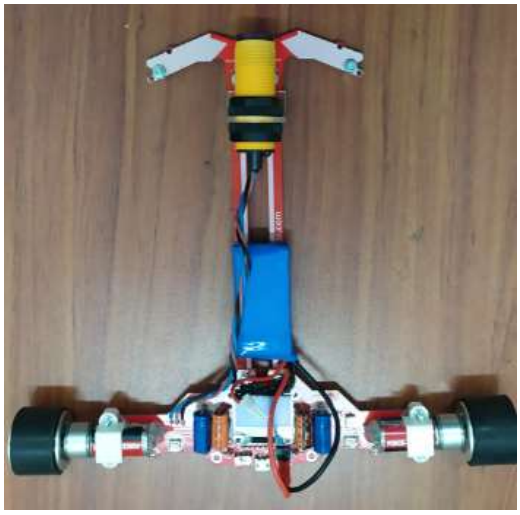


Figura 3. 30: Partes montadas en el robot seguidor de línea.
Elaborado por: Autor.

Figura 3. 31: Sensores QTR
Elaborado por: Autor.

Figura 3. 32: Sensor frontal fotoeléctrico M18.
Elaborado por: Autor

Figura 3. 33: Resultados de test de sensores de piso.
Elaborado por: Autor

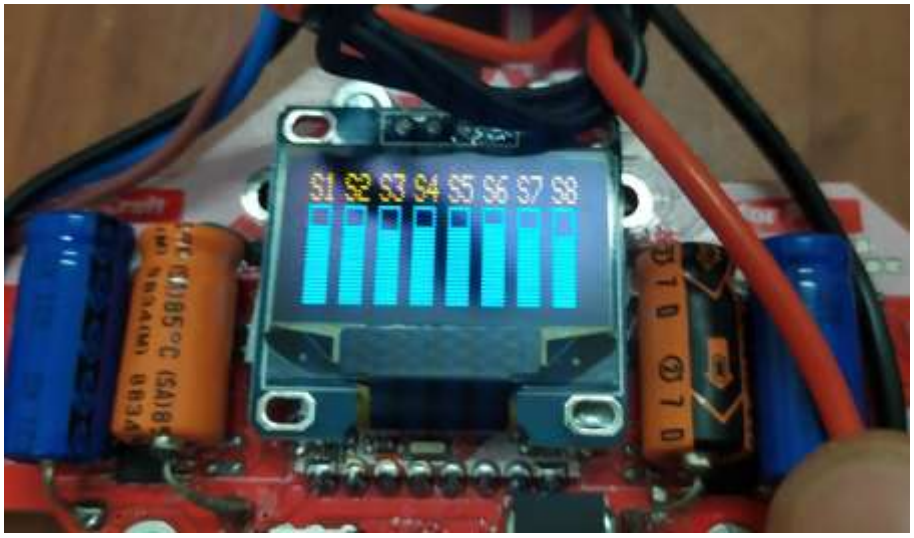


Figura 3. 34: Resultados de lectura de sensores de piso.
Elaborado por: Autor

Conclusiones

En el proyecto, se estudió el diseño el algoritmo de programación para el robot seguidor de línea y la implementación del controlador PID. Los parámetros del controlador se diseñaron utilizando el método de ajuste manual. Debido a los ocho sensores de piso en el hardware, se obtuvieron resultados eficientes en el control de velocidad de motores en especial durante las curvas complejas de la pista. La respuesta del sistema fue mejor en comparación con la de un controlador de circuito abierto simple. La cantidad de sobreimpulso también se redujo utilizando las técnicas mencionadas anteriormente.

Recomendaciones.

El alcance futuro de este proyecto implica el uso del método Ziegler Nichols para ajustar el controlador PID. Esto se puede hacer descubriendo la función de transferencia del proceso y ajustando el valor de la ganancia proporcional hasta el punto en que la salida comienza a oscilar. Al observar la disminución de la ganancia y el período de tiempo de las oscilaciones, el valor de la ganancia integral y derivada también se puede determinar.

Bibliografía.

Álvarez García, D. (2016). *Diseño e implementación de un robot pedagógico* [Proyecto Final de Carrera, Universidad Politécnica de Cataluña]. https://upcommons.upc.edu/handle/2117/86657

Calderón, H. A., Mejía, C., & Cobo, L. (2016). Implementación de un robot móvil seguidor de línea y detector de obstáculos con comunicación Bluetooth. *Revista Ontare*, *4*(2), Article 2. https://doi.org/10.21158/23823399.v4.n2.2016.1639

Carrillo Romero, M., Cardona Soto, J. A., Arvizo Gutiérrez, G. A., & Rodríguez Rico, F. (2017). Sistema de control y arquitectura de un robot seguidor de línea. *Cultura Científica y Tecnológica*, *2*(59), Article 59. http://148.210.132.19/ojs/index.php/culcyt/article/view/1570

Ferro Laspidea, P. (2014). *Diseño e implementación de un robot para la automatización de un almacén* [Trabajo Fin de Grado, Universidad Pública de Navarra]. https://academica-e.unavarra.es/xmlui/handle/2454/11744

Galeano, P. A., Torres, I. D., & Álvarez, J. F. (2014). Un Robot Móvil Autónomo Seguidor De línea. *Investigación e Innovación en Ingenierías*, *2*(1), Article 1. https://doi.org/10.17081/invinno.2.1.2058

Gomes, M. V., Bassora, L. A., Morandin, O., & Vivaldini, K. C. T. (2016). PID control applied on a line-follower AGV using a RGB camera. *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 194–198. https://doi.org/10.1109/ITSC.2016.7795553

Hasan, K. M., Abdullah-Al-Nahid, & Al Mamun, A. (2012). Implementation of autonomous line follower robot. *2012 International Conference on Informatics, Electronics & Vision (ICIEV)*, 865–869. https://doi.org/10.1109/ICIEV.2012.6317486

Ortiz Martínez, D. (2016). *Robótica para seguimiento de líneas* [Proyecto Final de Carrera, Universidad Politécnica de Cataluña]. https://upcommons.upc.edu/handle/2117/97322

Pakdaman, M., & Sanaatiyan, M. M. (2009). Design and Implementation of Line Follower Robot. *2009 Second International Conference on Computer and Electrical Engineering*, 585–590. https://doi.org/10.1109/ICCEE.2009.43

Proano, V., & Gualsaqui, M. (2017). Line follower simulator with fuzzy control. *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, 1–6. https://doi.org/10.1109/CHILECON.2017.8229638

DECLARACIÓN Y AUTORIZACIÓN

Yo, **Vásquez Rojas, Pamela Michelle** con C.C: # 094033295-0 autor del Trabajo de examen complexivo: **Implementación de algoritmos eficientes en la programación de un robot seguidor de línea utilizando métodos de control PID** previo a la obtención del título de **INGENIERA EN TELECOMUNICACIONES** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 6 de marzo del 2020

f. _____

Nombre: Vásquez Rojas, Pamela Michelle

C.C: 094033295-0

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA

FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN

TÍTULO Y SUBTÍTULO:	Implementación de algoritmos eficientes en la programación de un robot seguidor de línea utilizando métodos de control PID.		
AUTOR(ES)	Vásquez Rojas, Pamela Michelle		
REVISOR(ES)/TUTOR(ES)	M. Sc. PACHECO BOHÓRQUEZ, HÉCTOR IGNACIO		
INSTITUCIÓN:	Universidad Católica de Santiago de Guayaquil		
FACULTAD:	Facultad de Educación Técnica para el Desarrollo		
CARRERA:	Ingeniería en Telecomunicaciones		
TITULO OBTENIDO:	Ingeniera en Telecomunicaciones		
FECHA DE PUBLICACIÓN:	6 de marzo del 2020	**No. DE PÁGINAS:**	38
ÁREAS TEMÁTICAS:	Sistemas Digitales, Microcontroladores, Comunicaciones digitales		
PALABRAS CLAVES/ KEYWORDS:	Robótica, PID, Métodos, Controlador, Microcontroladores, Parámetros		

RESUMEN/ABSTRACT:

Este trabajo es el desarrollo del componente práctico del examen complexivo, y consistió en realizar una aplicación práctica en robots autónomos como es el caso del seguidor de línea, pero empleando métodos de control proporcional-integral-derivado llamado generalmente PID. El seguimiento de línea es uno de los aspectos más importantes de la robótica. Un robot seguidor de línea es un robot autónomo que puede seguir una línea negra o blanca dibujada en la superficie que consiste en un color contrastante. Este seguidor de línea está diseñado para moverse automáticamente y seguir la línea de trazado realizada. El control PID del seguidor de línea es un método que consiste en mejorar el movimiento del robot. El robot utiliza varios sensores para identificar la línea blanca, ayudando así al robot a permanecer en la pista. El seguidor de línea es impulsado por motores DC para controlar el movimiento de las ruedas. El microcontrolador Atmega se utilizó para realizar e implementar algoritmos PID para controlar la velocidad de los motores que conducen al robot a recorrer las líneas con suavidad. mediante el ajuste adecuado de los parámetros de control y así lograr un mejor rendimiento.

ADJUNTO PDF:	☒ SI	☐ NO
CONTACTO CON AUTOR/ES:	**Teléfono:** +593-9-84981780	E-mail: pamelamichelle.pv1@gmail.com
CONTACTO CON LA INSTITUCIÓN:	**Nombre:** Palacios Meléndez Edwin Fernando	
COORDINADOR DEL PROCESO DE UTE	**Teléfono:** +593-9-67608298	
	E-mail: edwin.palacios@cu.ucsg.edu.ec	

SECCIÓN PARA USO DE BIBLIOTECA

Nº. DE REGISTRO (en base a datos):	
Nº. DE CLASIFICACIÓN:	
DIRECCIÓN URL (tesis en la web):	