UNIVERSIDAD de VALLADOLID

ESCUELA de INGENIERÍAS INDUSTRIALES

INGENIERO TÉCNICO EN TELECOMUNICACIÓN, ESPECIALIDAD EN SISTEMAS ELECTRÓNICOS

PROYECTO FIN DE CARRERA

CONSTRUCCIÓN ÍNTEGRA DE UN SEGUIDOR SOLAR DE DOS EJES Y CONTROL DE SUS RESPECTIVOS MOTORES

Autores:

Cervero Hernández, Víctor

López Quindós, David

Tutor:

García Ruíz, Francisco Javier — Ingeniería de Sistemas y Automática

JUNIO – 2013

INTRODUCCIÓN

DESCRIPCIÓN Y ANÁLISIS DE SOFTWARE Y HARDWARE EMPLEADO

PROCESO DE FABRICACIÓN (CONSTRUCCIÓN MECÁNICA)

MODELO Y DESCRIPCIÓN MATEMÁTICA

FUNCIONAMIENTO DEL SISTEMA DE CONTROL

RESULTADOS Y CONCLUSIONES

ANEXOS

INTRODUCCIÓN

1.1.- Introducción, resumen y características descriptivas.

En este proyecto fin de carrera, lo que se ha realizado es la construcción íntegra de un seguidor solar de dos ejes.

Un seguidor solar es un dispositivo mecánico, que mediante una estructura soporta una o varias placas solares, a la vez es capaz de orientar las placas solares de forma que estén lo más perpendicularmente posible a los rayos de Sol. Las placas solares se orientan al moverse la estructura que las soporta, esta estructura puede moverse sobre uno o sobre dos ejes.

Si se trata de dos ejes, este hace un movimiento de Este-Oeste y Norte-Sur, si es de un eje, generalmente hace un movimiento de Este a Oeste. Este tipo de mecanismo sirve para mejorar la captación de energía solar. Para elegir entre uno u otro habrá que tener en cuenta las características que se quieran conseguir con la instalación.

En este trabajo se pretende explicar la implementación de un control de la velocidad del motor que en este caso es de corriente directa, regulando su velocidad, regulamos también la posición de la placa con respecto de una intensidad luminosa.

1.2.- Objetivos principales.

- Obtención del modelo matemático aproximado del sistema usando leyes físicas como la segunda ley de Newton y transformadas de *Laplace*.
- Conseguir un sistema lo más estable posible durante su evolución temporal.
- Construcción mecánica del sistema intentando que haya el menor rozamiento para que el movimiento sea lo más limpio posible, tanto en el eje superior como en el inferior.
- Implementar el algoritmo correcto utilizando la herramienta de *Matlab*, *Simulink*, y el microprocesador Arduino UNO.
- Realizar una buena regulación de la velocidad de los motores en sus respectivos ejes para lograr la posición deseada y más óptima.
- Realizar diferentes simulaciones para adquirir una buena respuesta del sistema físico real.

1.3.- Funcionamiento y control.

La forma de guiar el seguidor es mediante sensores, en este caso fotorresistencias, por los que detectan si una intensidad luminosa inciden perpendicularmente sobre la placa, o no. Mediante estos sensores, el microprocesador Arduino, recoge la información, la envía a un PC y será procesada en Simulink, y partir de ahí, dará la orden a los motores para que se muevan, mediante la utilización de un circuito integrado (L293D).

DESCRIPCIÓN Y ANÁLISIS DE SOFTWARE Y HARDWARE EMPLEADO

2.1.- Sensor de luz (Fotorresistencia o LDR).

La '*LDR*', también conocida como resistencia o fotoconductor, es un sensor cuya resistencia eléctrica varía en función de la intensidad luminosa que recibe.

El funcionamiento de este semiconductor se basa en que al incidir fotones sobre el dispositivo, entonces el semiconductor los absorbe en forma de energía, de manera que los electrones de la banda de valencia saltan a la de conducción, siempre que la luz incidente tenga la suficiente frecuencia, o en otras palabras, la suficiente energía.

El resultado es, por lo tanto, la disminución de la resistencia eléctrica del dispositivo, dado que el electrón libre (y el hueco asociado) se genera en la banda de conducción.

Podemos dividir las fotorresistencias en dos tipos, que son los dispositivos intrínsecos, y los extrínsecos.

En el caso de los intrínsecos, los únicos electrones que tienen la capacidad de saltar a la banda de conducción están situados en la banda de valencia, y necesitan una elevada energía para pasar a la banda de conducción.

Los extrínsecos se dopan con impurezas, por lo que los electrones adquieren una energía inicial mayor que en el caso intrínseco, y por lo tanto, es necesaria una energía menor para saltar a la banda de conducción.

La resistencia de la '*LDR*' se caracteriza como:

$$R = AE^{-\alpha}$$

R: Resistencia de la LDR

A, α: Coeficientes que dependen del semiconductor utilizado

E: Densidad superficial de energía recibida

En nuestro caso hemos utilizado la *LDR NORPS – 12 de SILONEX*, que a continuación ilustramos y mostramos sus principales rangos de funcionamiento.

Fig.2.1.Fotorresistencia NORPS-12

Temperatura de funcionamiento: -60 a +75 ºC
Voltaje máximo: 250 Voltios
Potencia disipación a 30 ºC: 250 mW

2.2.- Micro-controlador Arduino UNO.

Arduino es una plataforma de prototipos de electrónica de código abierto basada en hardware y software flexibles y fáciles de usar. Arduino puede "*sentir*" el entorno mediante la recepción de entradas desde una variedad de sensores y puede afectar a su alrededor mediante el control de luces, motores, y otros artefactos. El lenguaje de programación que se usa es *C/C++.*

Arduino tiene muchas ventajas; es barato comparadas con otras plataformas micro-controladoras; es multiplataforma ejecutándose en diferentes sistemas operativos; el entorno de programación es simple y claro pero suficiente flexible para temática avanzada, y muchas otras ventajas que no cabe mencionar para este trabajo.

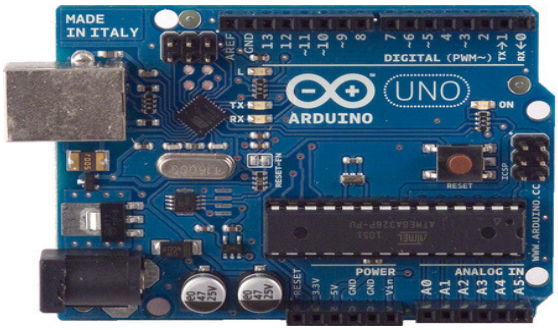


Fig.2.2.Arduino UNO

A continuación, pasaremos a explicar la estructura de un sketch y las funciones que se usarán en el entorno de programación de Arduino en este proyecto:

- Estructura de un sketch:

Función Setup(): Se invoca una sola vez cuando el programa empieza. Se utiliza para inicializar los modos de trabajo de los 'pins', o el puerto serie. Debe ser incluido en un programa aunque no haya declaración que ejecutar. Así mismo, se puede utilizar para establecer el estado inicial de las salidas de la placa.

Función Loop(): Después de llamar a *setup(),* la función *loop()* hace precisamente lo que sugiere su nombre, se ejecuta de forma cíclica, lo que posibilita que el programa este respondiendo continuamente ante los eventos que se produzca en la placa. Esta función es el núcleo de todos los programas de Arduino y la que realiza la mayoría del trabajo.

Declaración de variables: Una variable es una manera de nombrar y almacenar un valor numérico para uso posterior en el programa. Las variables son números que se pueden variar continuamente, tomando nombres descriptivos, para hacer el código más legible y entendible y ayudar al programador y a cualquier otra persona que representa la variable. Todas las variables tienen que declararse antes de que puedan ser utilizadas. Una variable puede ser declarada al inicio del programa antes de la parte de configuración *setup(),* a nivel dentro de las funciones, y, a veces, dentro de un bloque, como para los bucles del tipo *if…, for*…, etc.

Nosotros utilizaremos variables globales que son aquellas que pueden ser vistas y utilizadas por cualquier función y estamento de un programa. Esta variable se declara al comienzo del programa, antes del *setup().*

- pinMode(pin, OUTPUT):

Esta instrucción es utilizada en la parte de configuración *setup()* y sirve para configurar el modo de trabajo de un 'pin' pudiendo ser '*INPUT*' (entrada) u '*OUTPUT*' (salida). En nuestro caso lo configuramos como 'pin' de salida.

Los 'pins' configurado como '*OUTPUT*' (salida) se dice que están en un estado de baja impedancia y pueden proporcionar 40 mA de corriente a otros dispositivos y circuitos, corriente suficiente para alimentar un diodo LED, pero no suficiente para alimentar cargar de mayor consumo como relés, solenoides, o motores.

- analogRead(pin):

Lee el valor de un determinado 'pin' definido como entrada analógica con una resolución de 10 'bits'. Esta instrucción sólo funciona en los pines 0 – 5. El rango de valor que podemos leer oscila de 0 a 1023.

- analogWrite:

Esta instrucción sirve para escribir un pseudo - valor analógico utilizando el procedimiento de modulación por ancho de pulso (PWM) a uno de los 'pins' de Arduino marcados como pin *PWM*. El más reciente Arduino, que implementa el chip ATmega168, permite habilitar como salidas analógicas tipo PWM los pines 3, 5, 6, 9, 10 y 11. Los modelos de Arduino más antiguos que implementan el chip ATmega8, solo tiene habilidades para esta función los pines 9, 10, y 11. El valor que se puede enviar a estos pines de salida analógica puede darse en forma de variable o constante, pero siempre con un margen de 0-255.

Si enviamos el valor 0 genera una salida de 0 voltios en el pin especificado; un valor de 255 genera una salida de 5 voltios de salida en el pin especificado. Para valores de entre 0 y 255, el pin saca tensiones entre 0 y 5 voltios. Teniendo en cuenta el concepto de señal *PWM*, por ejemplo, un valor de 64 equivaldrá a mantener 0 voltios de tres cuartas partes del tiempo y 5 voltios una cuarta parte del tiempo; un valor de 128 equivaldrá a mantener la salida en 0 la mitad del tiempo y 5 voltios la otra mitad del tiempo, y un valor de 192 equivaldrá a mantener en la salida 0 voltios una cuarta parte del tiempo y de 5 voltios de tres cuartas partes del tiempo restante.

- HIGH/LOW:

Estas constantes definen los niveles de salida altos o bajos y se utilizan para la lectura o la escritura digital para las patillas. Alto se define con en la lógica de nivel 1, '*ON*', o 5 voltios, mientras que bajo es lógica nivel 0, '*OFF*', o 0 voltios.

- digitalWrite(pin, value):

Envía al 'pin' definido previamente como '*OUTPUT*' el valor '*HIGH*' o '*LOW*' (poniendo en 1 o 0 la salida). El 'pin' se puede especificar ya sea como variable o como una constante.

- Map(*value, fromLow, fromHigh, toLow, toHigh*)

Remapea un número desde un rango hacia otro. Esto significa que, un valor (valor) será mapeado del rango '*fromLow-fromHigh*' al rango '*toLow-toHigh'*.

Ten en cuenta que los límites "inferiores" de algún rango pueden ser mayores o menores que el límite "superior" por lo que *map()* puede utilizarse para revertir una serie de números.

La función maneja correctamente también los números negativos.

Esta función usa matemática de números enteros por lo que no generará fracciones, aunque fuere el resultado correcto. Los resultados en fracciones se truncan, y no son redondeados o promediados.

Resumiendo, los parámetros usados son:

'Value': El número o valor a mapear.

'fromLow': El límite inferior del rango actual del valor.

'fromHigh': el límite superior del rango actual del valor.

'toLow': límite inferior del rango deseado.

'toHigh': límite superior del rango deseado.

Nos devuelve el valor mapeado.

- abs(x):

Calcula el valor absoluto de un número.

'X': El número cuyo valor absoluto deseamos calcular.

Devuelve 'x' si el número 'x' es mayor o igual que 0.

Devuelve '-x' si el número 'x' es menor que 0.

- delay(valor):

Detiene la ejecución del programa la cantidad de tiempo en milisegundos que se indica en la propia instrucción. De tal manera que 1000 equivale a 1 segundo.

- Estamento if (si condicional):

If es un estamento que se utiliza para probar si una determinada condición se ha alcanzado, como por ejemplo, averiguar si un valor analógico está por encima de cierto número, y ejecutar una serie de declaraciones que se escriben dentro de llaves, si es verdad. Si es falso el programa salta y no ejecuta las operaciones que están dentro de las llaves.

- Serial.begin(*rate*):

Abre el puerto serie y asigna la tasa de baudios para la transmisión de datos serie. La típica tasa de baudios para comunicarse con el ordenador es 9600 aunque a otras velocidades están soportadas.

- Serial.write(*valor*):

Escribe datos binarios en el puerto serie. Estos datos se envían como un byte o una serie de bytes; para enviar los caracteres que representan los dígitos de un número se usa la función '*print()'* en su lugar. El valor se envía como un solo byte.

- Serial.read(valor):

Lee los datos entrantes del puerto serie. Devuelve el primer byte disponible recibido por el puerto serie (devuelve -1 si no hay datos disponibles).

2.2.1.- Señal PWM.

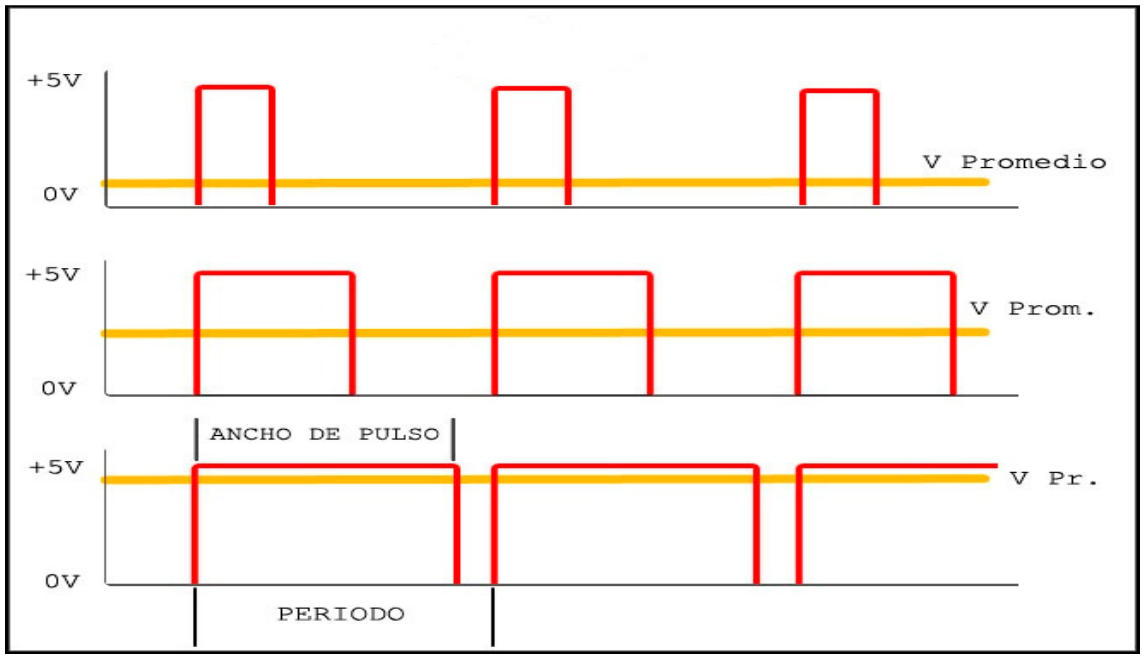


Fig.2.3.Señal PWM

A menudo necesitaremos algo más que una señal de 0 o 1 en nuestros proyectos, como ocurre en el nuestro, que para variar la velocidad de giro de un motor o para transmitir los grados de giro de un servo.

Para todo esto, y mucho más, nos servirá el *PWM*, que emula una señal analógica a partir de una señal digital.

Las siglas *PWM* vienen de '*Pulse Width Modulation*' o *Modulación de Ancho de Pulso*, a la duración del tiempo de encendido '*ON*' se le llama Ancho de Pulso y el tiempo entre pulso y pulso Periodo.

Lo que hace este tipo de señal es emitir, en lugar de una señal continua en nuestra salida, emite una serie de pulsos que podremos variar su duración pero con una frecuencia constante de aproximadamente 490 Hz, de manera que la tensión promedio resultante es directamente proporcional a la duración de estos dentro del rango de nuestro periodo, es decir, cuanto más juntos estén esos pulsos de 5V, mayor será la tensión promedio de nuestra salida, y cuanto más distantes sean estos pulsos, menor será dicha tensión.

La llamada a la función '*analogWrite()*' debe ser en la escala desde 0 a 255, siendo 255 el 100% de ciclo (siempre encendido, '*ON*'), el valor 127 será el 50% del ciclo (la mitad del tiempo de encendido), etc.

2.2.2.- Comunicación USB.

En la comunicación con el ordenador Arduino emplea la comunicación asíncrona. Además de realizar las conexiones físicas entre el micro-controlador y el ordenador, para que pueda establecerse la comunicación serial debe existir un acuerdo previo en la manera de cómo van a ser enviados los datos. Arduino facilita este proceso para que sólo sea necesario especificar la velocidad de envío de los datos. Esta velocidad es conocida como rata de pulsos por segundo y su velocidad frecuente es de 9.600 baudios.

Actualmente en la mayoría de los periféricos serie, la interfaz USB ha reemplazado al puerto serie por ser más rápida. La mayor parte de los ordenadores están conectados a dispositivos externos a través de USB y, a menudo, ni siquiera llegan a tener un puerto serie.

El bus universal en serie USB es un estándar industrial que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre ordenadores y periféricos y dispositivos electrónicos.

El USB, en este caso servirá para enviar y recibir datos desde y hacia el micro-controlador, además de proveer al mismo de alimentación.

Es un sistema de bus serie que puede conectarse con varios dispositivos. Hoy en día es muy utilizado para aplicaciones, tal como la conexión de dispositivos de forma sencilla al ordenador. El USB ha sustituido al RS – 232 porque es más rápido, utiliza baja tensión para el transporte de datos (3.3V) y es fácil de conectar. Este puerto es half – dúplex, lo que significa que solo puede enviar o recibir datos en un mismo instante de tiempo.

El USB 2.0 tiene cuatro hilos: '*VCC', 'GND'*, datos de entrada y salida, con una velocidad de transferencia de hasta 480Mbps pero por lo general de hasta 125Mbps. Está presente casi en el 99% de los ordenadores actuales.

2.3.- Simulink.

Simulink es un entorno de programación visual, que funciona sobre el entorno de programación de *Matlab*.

Es un entorno de programación de más alto nivel de abstracción que el lenguaje interpretado *Matlab*. *Simulink* genera archivos con extensión '*.mdl'.*

Simulink viene a ser una herramienta de simulación de modelos o sistemas, con cierto grado de abstracción de los fenómenos físicos involucrados en los mismos. Se hace hincapié en el análisis de sucesos, a través de la concepción de sistemas. Se emplea arduamente en Ingeniería Electrónica en temas relacionados con el procesamiento digital de señales (DSP), involucrando temas específicos de ingeniería biomédica, telecomunicaciones, entre otros. También es muy utilizado en Ingeniería de Control y Robótica.

A continuación pasaremos a explicar los diferentes bloques utilizados con esta herramienta, que por una parte tendremos, los bloques utilizados para la comunicación entre *Simulink* y nuestro micro-controlador Arduino, y por otra parte, tendremos aquellos bloques usados en el algoritmo de control de nuestra planta.

Para el envío y recepción de datos, hemos de tener en cuenta que tenemos, como ya hemos explicado en apartados anteriores, que nuestro puerto USB es '*half – dúplex*', lo que significa que en un mismo instante de tiempo, no podremos recibir o enviar datos a la vez, por lo que el puerto debe estar libre de tráfico de datos.

- Bloque Serial Configuration:

En este bloque se configuran los parámetros de nuestro puerto serie que utilizaremos para el envío y recepción de datos. Antes debemos configurar una serie de parámetros que son los siguientes:

Communication port: Específica el puerto serie a configurar. Ha de configurarse un puerto serie que aparezca en la lista desplegable. Por defecto, no hay ninguno marcado, seleccionaremos el que corresponda con nuestra tarjeta Arduino. El puerto que hayamos seleccionado aquí deberá ser el mismo para posteriores bloques de comunicación que detallaremos.

Baud rate: Nos índica la velocidad de transmisión de los datos en baudios, en nuestro caso será de 9600 baudios.

Data bits: Especifica el número de bits que se van a enviar por el puerto serie.

Parity: Indica como chequear los bits de paridad en los datos que transmitimos.

Stop bits: Indica la cantidad de bits que determinan el final de un byte.

Timeout: Especifica el tiempo que el modelo va a esperar a los datos durante cada paso de tiempo de simulación. Por defecto, su valor es de 10 segundos.

- Serial Receive:

Este bloque configura y abre una interfaz a una dirección remota dónde se va produciendo una adquisición de datos durante el tiempo de ejecución del modelo. La configuración ocurre una vez que se da al comienzo de la simulación.

Communication port: Especifica el puerto a través del cual se van a recibir los datos.

Data size: Se indica el número de bits que se reciben.

Data type: Tipo de dato que recibimos.

Sample time: Tiempo de muestreo del bloque, es decir, la frecuencia con la se va a leer un dato.

- Serial Send:

En este bloque enviamos los datos generados por nuestro algoritmo, a través del puerto serie, en este caso, hacia la tarjeta Arduino.

- Data Type Conversion:

La función de este bloque es convertir un tipo de dato en otro tipo de dato. En nuestro caso queremos convertir datos a formato "*uint8*" y a formato "*double*". En primer lugar convertimos los datos formato "*double*" debido a que *Simulink*, las operaciones las hace en este formato y también para los elementos gráficos del entorno, y posteriormente se convierte de nuevo en "*uint8*", ya que la tarjeta Arduino trabaja en su interior en formato de 8 'bits', y también se transmite por el puerto serie en este formato. Estas conversiones son necesarias en la comunicación Arduino – PC y viceversa.

- Bloque PID(z):

Bloque que se usa para implementar un controlador continuo o en tiempo discreto. Las ganancias del control '*PID'* son sintonizables de forma manual o automática.

La salida del bloque del controlador *'PID'* es una suma ponderada de la señal de entrada, la integral de la señal de entrada, y la derivada de la señal de entrada. Tenemos para ajustar los parámetros de ganancia, integral, derivativa y proporcional.

La entrada es típicamente una señal de error, que es la diferencia entre una señal de referencia y la salida del sistema.

Las opciones configurables dentro de este bloque son, el tipo de regulador (PID, PI, PD, P o I), forma de controlador (paralelo o ideal), el tiempo del dominio (continuo o discreto), las condiciones iniciales y de activación de reinicio, límites de saturación de salida y seguimiento de señales de control de transferencia sin perturbaciones y control multi-lazo.

En nuestro caso, tendremos un controlador '*PID'* en tiempo discreto, ya que la tarjeta Arduino en su interior digitaliza los datos recibidos. La función de transferencia de un '*PID'* en tiempo discreto es la siguiente:

$$C(z) = P + Ia(z) + D\left[\frac{N}{1 + Nb(z)}\right]$$

- Bloque Gain:

El bloque '*Gain'* multiplica la entrada por un valor constante (ganancia). La entrada y la ganancia pueden ser un escalar, un vector o una matriz. El valor de la ganancia se especifica a través del parámetro '*Gain'*. El parámetro '*Multiplication'* determina si la multiplicación es matricial o elemento a elemento. El orden de las multiplicaciones en las operaciones matriciales es configurado a través de este parámetro.

- Bloque Scope:

El bloque '*Scope'* representa gráficamente la entrada conectada a este bloque con respecto al tiempo de simulación. Este bloque permite representar varias variables a la vez para el mismo periodo de tiempo. El '*Scope'* permite ajustar el tiempo y el rango de los valores de entrada representados. Se puede mover y redefinir el tamaño de la ventana y se puede modificar los valores de sus parámetros durante la simulación.

Si la señal de entrada al bloque '*Scope'* está formada por varias variables, éste asigna colores a cada elemento de la señal en el siguiente orden: amarillo, magenta, cían, rojo, verde y azul oscuro. Cuando la señal posee más de seis elementos se repite el orden de los colores.

2.4.- Circuito integrado L293D.

El L293D de Texas Instruments es sin lugar a dudas un circuito integrado de un gran valor cuando necesitamos controlar motores de corriente continua o motores de paso a paso.

Es cierto que se trata de un puente en H (o medios puentes), en este caso cuádruple, que sin bien podríamos crearlo con transistores, el hecho de que se encuentre integrado en un único chip es de agradecer.

Capaz de conducir corrientes bidireccionales de hasta 1 amperio en el modelo L293 y hasta 600 mA en el modelo L293D y con tensiones que van desde los 4.5V hasta los 36V en ambos modelos.

Por supuesto podemos utilizarlo en otras aplicaciones o para controlar otros componentes: motores de corriente continua, relés, motores de paso bipolares, solenoides en general y cualquier carga que requiera una alta corriente y tensión.

Las entradas son de tipo *TTL* y se activan por parejas, es decir, desde la pata '*Enable*' 1,2EN, activamos las entradas 1 y 2 y desde la pata '*Enable'* 3,4EN activamos la 3 y la 4. Cada par de entradas forma un puente en 'H' completo (*Full-H bridge*). Para entender esto mejor podéis buscar un puente en 'H' creado con transistores.

El integrado permite formar, dos puentes 'H' completos, con los que se puede realizar el manejo de dos motores. En este caso el manejo será bidireccional, con frenado rápido y con posibilidad de implementar fácilmente el control de velocidad.

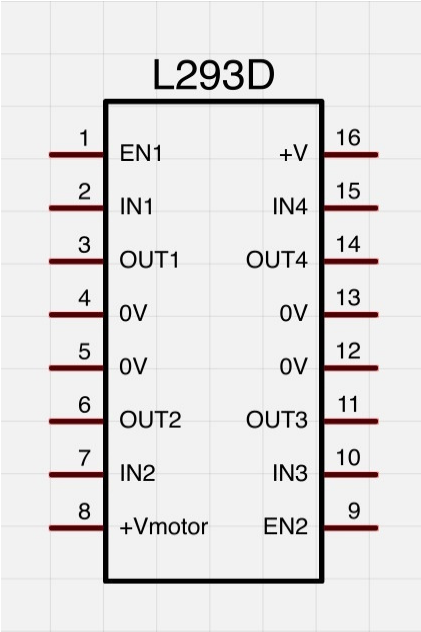


Fig.2.4.Pines integrado L293D

Las entradas de habilitación permiten controlar con facilidad el circuito, lo que facilita la regulación de la velocidad de los motores por medio de una modulación de ancho de pulso. En ese caso, las señales de habilitación en lugar de ser estáticas se controlarían por medio de pulsos de ancho variable.

Las salidas actúan cuando su correspondiente señal de habilitación está en alto. En estas condiciones, las salidas están activas y su nivel varía en relación con las entradas. Cuando la señal de habilitación del par de circuitos de manejo está en bajo, las salidas están desconectadas y en un estado de alta impedancia.

Las patas centrales de la cápsula del chip están pensadas para proveer el contacto térmico con un disipador que permitirá lograr la potencia máxima en el manejo del integrado.

Descripción de sus pines:

1: Entrada '*enable*' dónde se conectará nuestra señal PWM que controlará la velocidad del motor para aumentarla o disminuirla.

2: Una de las entradas del puente en H, para indicar el sentido del motor.

3: Una de las salidas hacia el motor.

4: Conexión a tierra.

5: Conexión a tierra.

6: Una de las salidas hacia el motor.

7: Una de las entradas del puente en 'H', para indicar el sentido del motor.

8: Alimentación de nuestro motor DC.

16: Alimentación del puente en 'H'.

A continuación ilustramos un claro ejemplo de sus 'pines' y conexiones con motores que nos ayudará a entender su funcionamiento de manera clara y sencilla:

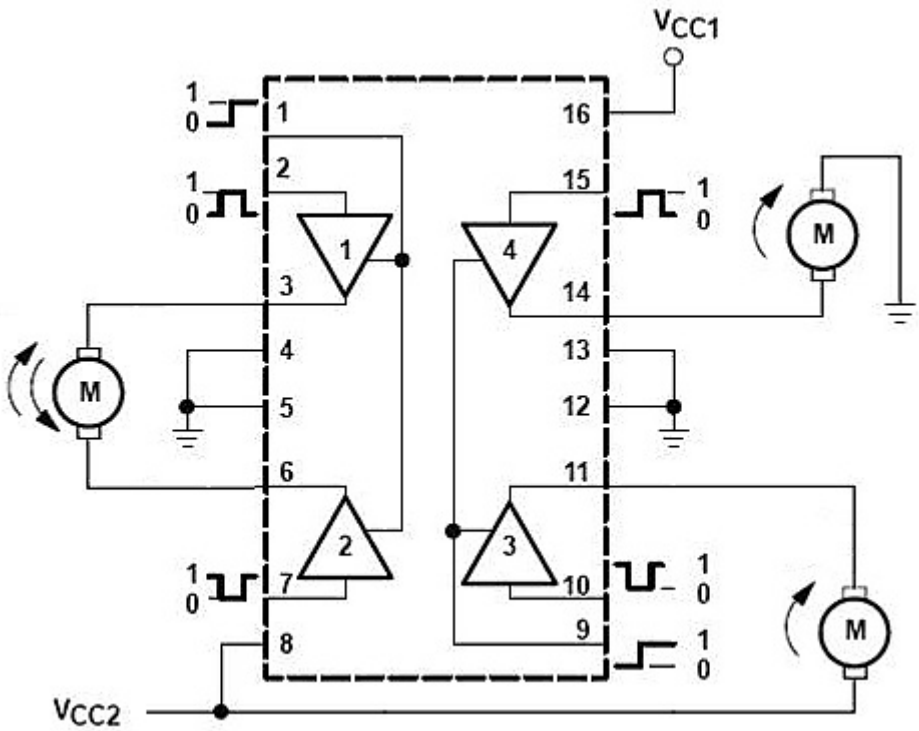


Fig.2.5.Funcionamiento integrado

PROCESO DE FABRICACIÓN (Construcción mecánica)

3.1.- Fabricación de la estructura

Necesitamos planchas de aluminio de 3mm de grosor. La estructura estará formada por dos piezas básicas.

-Una en forma de U, la superior, que sujetara la placa solar y girará sobre la inferior

-La inferior, circular, que hará de base.

EJE INFERIOR:

Unirá ambas piezas. Esta unión va a ser fundamental, pues va a soportar todo el peso del conjunto. Ha de girar con total libertad para evitar rozamiento, por ello instalamos un eje con un rodamiento cónico. El eje lo encargamos a un taller de mecanizado.

Para sujetar la parte externa del rodamiento a la parte inferior del conjunto hacemos un cajeado con unas arandelas del tamaño adecuado y 4 tornillos roscados de 4mm. De esta forma esa pieza queda totalmente sujeta.

Al eje se le suelda una pequeña plataforma para poder atornillarla a la pieza superior (a la U) con otros 4 tornillos roscados de 4 mm. Además en estos 4 tornillos vamos a poder anclar el engranaje después.

Ilustración 1: eje con el rodamiento

La parte interna del rodamiento es solidaria al eje. El eje se embute en esta parte a presión y se fija con una grupilla para que no se salga.

Ahora podemos introducir la parte superior con el eje en la parte inferior donde está fijada la parte exterior del rodamiento. Este gira perfectamente y sin rozamiento apenas.

El problema es que al ser el rodamiento cónico la estructura se tambalea hacia los lados ya que el rodamiento cónico se apoya en su carcasa por el peso del conjunto. Para solucionarlo hacemos una pieza circular de baquelita que es bastante resistente y el metal “patina” bien sobre ella para meter la punta del eje. De esta forma todo el peso recae sobre el rodamiento cónico y hace que gire sin rozamiento mientras que la otra pieza impide que el conjunto se tambalee hacia los lados. Esta pieza también nos sirve para unir ambas piezas definitivamente evitando que se salgan hacia arriba mediante un tornillo, una arandela y dos tuercas que se enroscan en el eje.

En la siguiente imagen de la parte de abajo se aprecia una de las arandelas que prensan la parte externa del rodamiento, junto con la pieza de madera en la que entra justo el eje para no baile, y la arandela con las 2 tuercas que evitan que este se salga.

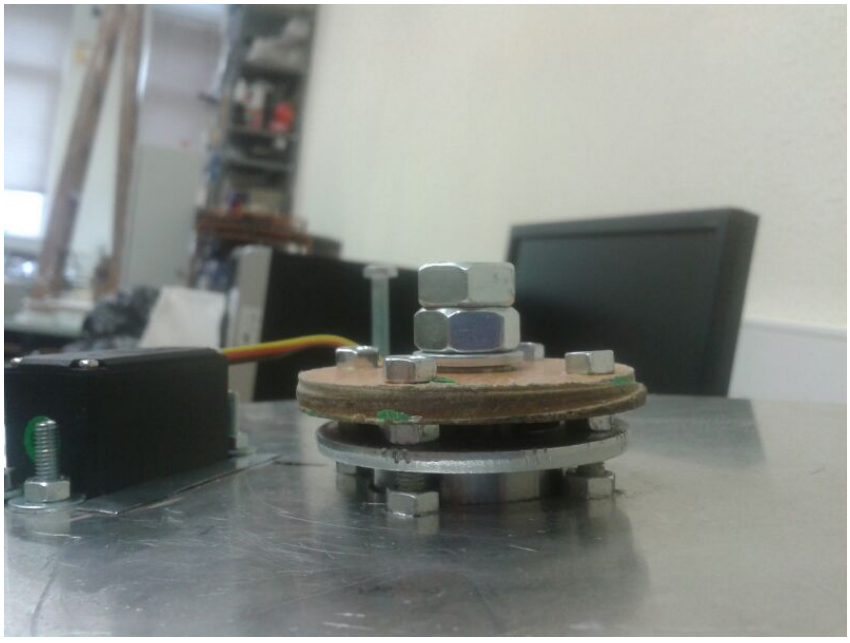


Ilustración 2: Detalle de la parte de abajo del eje inferior

En la imagen siguiente se ve la parte de arriba del eje inferior. Se aprecia la otra arandela que embute a la camisa del rodamiento y el detalle de la pieza circular soldada al eje con sus 4 tornillos que sujetan la parte superior de la estructura. Con esos 4 tornillos hemos aprovechado para sujetar el engranaje.

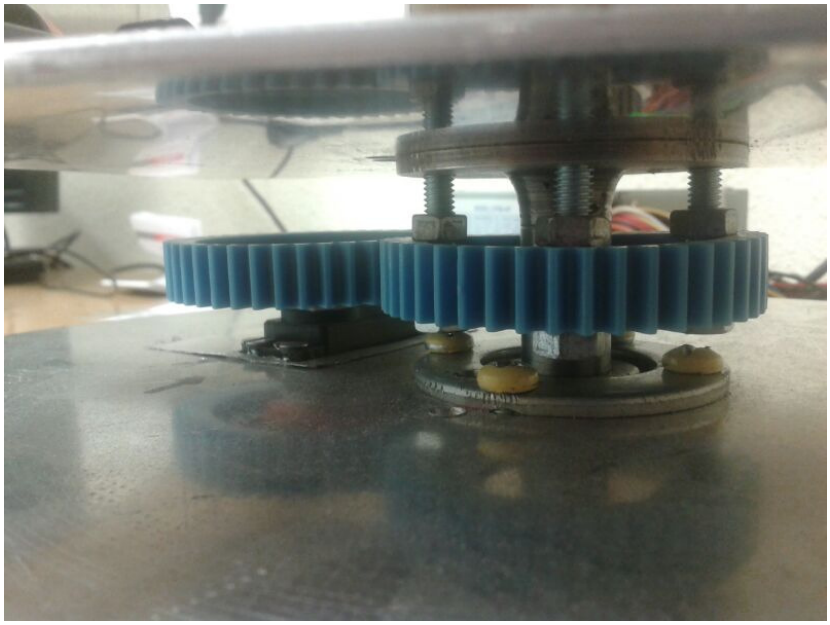


Ilustración 3: Detalle de la parte de arriba del eje inferior

EJE SUPERIOR

Una vez fabricado el eje inferior pasamos al eje superior, el que unirá a la placa solar con la pieza superior (la U).

Primero para poder sujetar la placa fabricamos 2 piezas de aluminio de 3 mm que irán fijadas a la placa con 2 tornillos roscados.

Ilustración 4: Pieza de sujeción de la placa

Mediante el eje vamos a unir estas dos últimas piezas con la U principal. Necesitaremos un eje de 4 mm de grosor y algo más del ancho de la placa de longitud. Este ira fijo en la pieza principal (la U). Usamos dos soportes que tienen una especie de rodamiento en el interior a través del cual entrara el eje.

Estos soportes que se pueden ver en la figura inferior, van atornillados a las 2 piezas que sujetan la placa. Hacemos un agujero en esta pieza para que pase el eje muy holgado, de tal forma que este solo descanse sobre estos soportes con rodamiento incorporado y el rozamiento sea mínimo.

Ilustración 5: Detalle eje superior

En la imagen siguiente se aprecia el eje al completo. Como en el eje inferior, los tornillos roscados empleados para fijar los soportes los aprovechamos para fijar el engranaje que moverá el sistema.

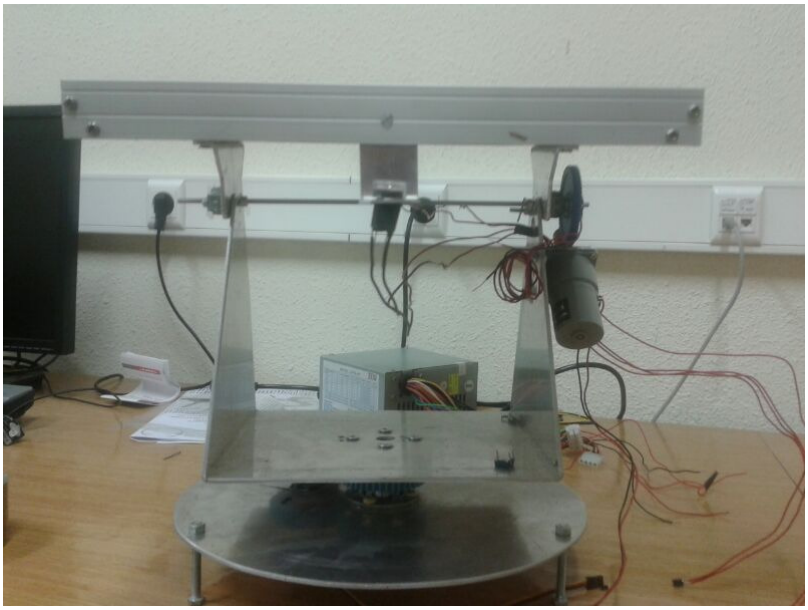


Ilustración 6: eje superior

El eje está fijado con pasadores y dos abrazaderas de tornillo a la U, por lo tanto solo giran los soportes con los rodamientos sobre este.

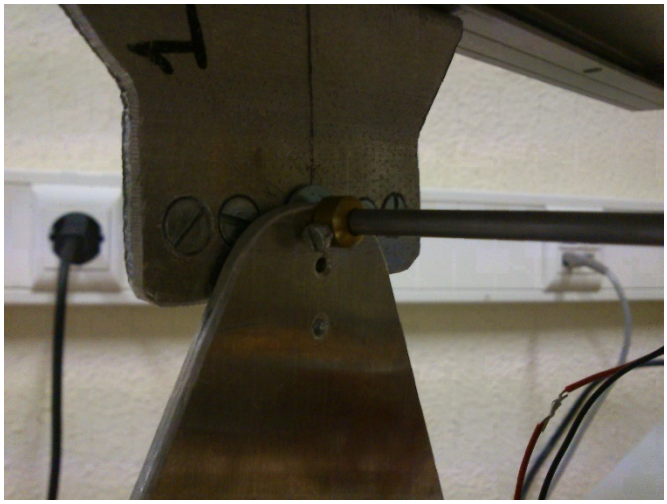


Ilustración 7: Detalle fijación del eje

3.2.- Instalación de los motores.

MOTOR INFERIOR:

Para el motor inferior hemos aprovechado la reductora y carenado de un servo, el cual se puede fijar tan solo con 4 tornillos. Solo hemos tenido que cortar el hueco necesario en la base con una sierra de calar para insertarlo. Hemos dejado algo de holgura para luego ajustar bien el acoplo de engranajes como se puede apreciar en las siguientes imágenes.

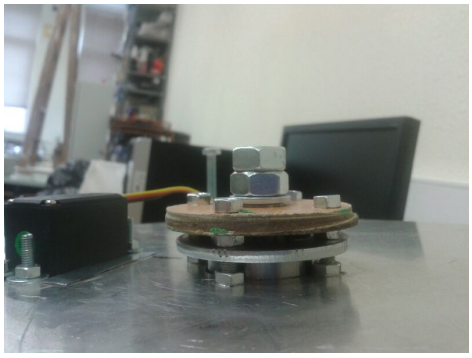
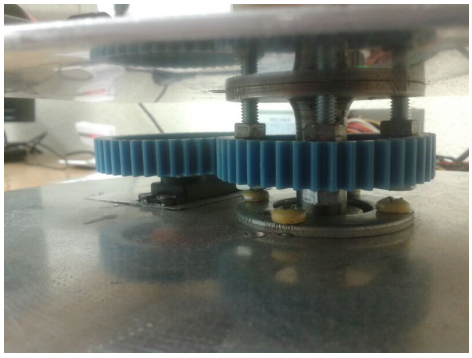


Ilustración 8: Motor inferior



MOTOR SUPERIOR:

En este caso empleamos un motor con reductora incorporada y con tornillo sinfín. Fabricamos en aluminio una especie de L para acoplarlo a la U con dos tornillos roscados. El motor se une a esta pieza con 4 tornillos roscados (habiendo hecho rosca a la pieza de aluminio previamente con unos machos).

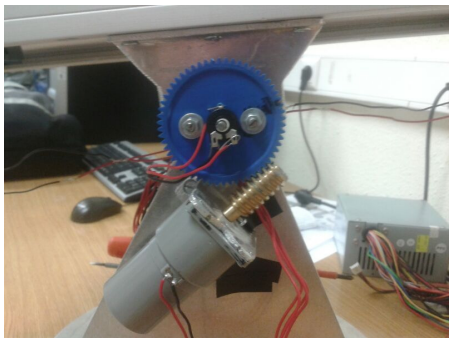


Ilustración 9: Motor superior

Como se ve en la imagen hemos incorporado un potenciómetro para tener datos sobre donde se encuentra el eje en cada momento.

3.3.- Instalación de las fotorresistencias.

Se colocan 4 LDRs, emparejadas dos a dos para que cada pareja controle un eje de movimiento.

Es necesario separar cada LDR de cada par por una barrera opaca para que esta cree una sombra si la fuente de luz no está perpendicular a ambas, así una LDR recibirá más luz que la otra, lo que se enviara al sistema para que este tome decisiones.

Las LDR las hemos colocado con una L de aluminio y un tornillo roscado.

Ilustración 10: LDR que controla eje superior

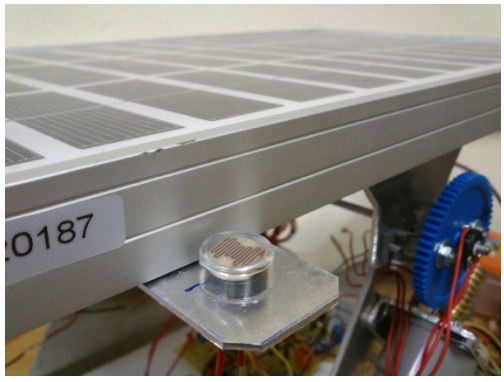


Ilustración 11: LDR que controla eje inferior

3.4.- Cálculo de las transmisiones.

EJE INFERIOR:

Para mover el eje inferior hemos aprovechado la reductora de un servomotor. Como el motor no es muy potente hemos adaptado la estructura para cambiar el motor original por uno más potente de 6v. Así conseguimos ya las rpm necesarias. La velocidad de giro conseguida es de 0.56 sec/60° con el nuevo motor.

Usamos dos engranajes de gran paso para llevar el movimiento al eje. Tienen el mismo número de dientes por lo que en este paso no se produce reducción alguna.

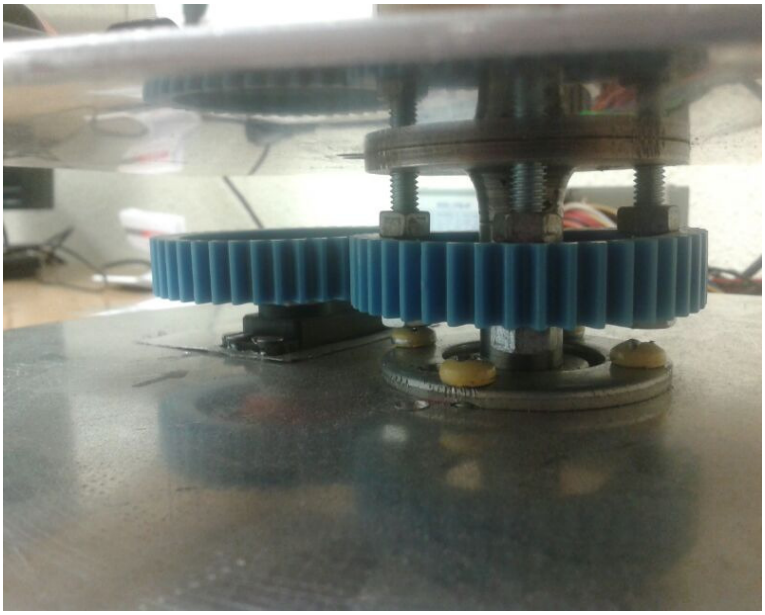


Ilustración 12: detalle transmisión eje inferior

En la siguiente imagen se aprecia la adaptación del carenado para el nuevo motor y la sujeción de este a la base con 2 tornillos y dos tuercas.

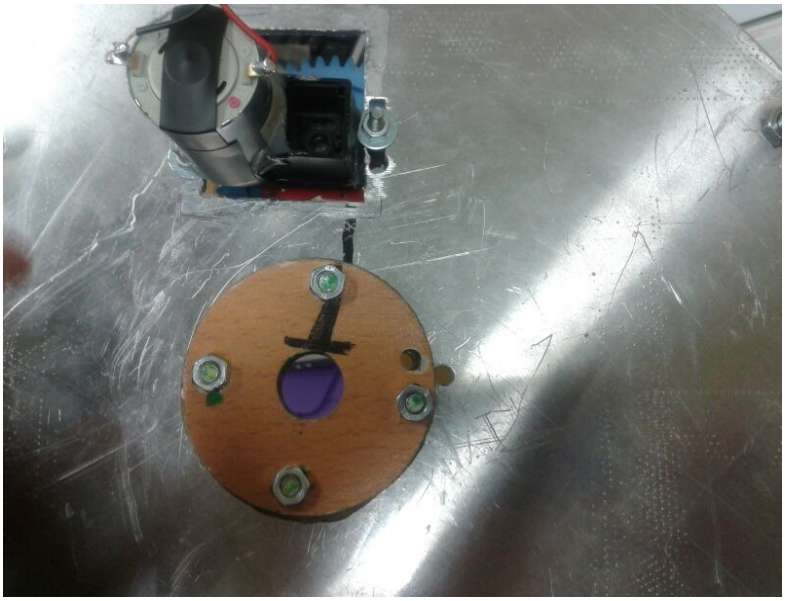


Ilustración 13: Sujeción motor inferior

EJE SUPERIOR:

Este eje deseamos que se mueva un poco más despacio, para ello fijamos una velocidad del 2 seg/ 45º.

Usamos un motor con reductora de 12v con par más que de sobra para mover toda la estructura. Este motor nos proporciona 240 rpm.

Para llegar al objetivo de 4 rpm que necesitamos tenemos que reducir ese valor unas 60 veces. Usamos un tornillo sinfín y un engranaje de 60 dientes, de tal modo que por cada 60 vueltas que dé el sinfín, y en este caso el motor, toda la estructura dará 1 sola vuelta llegando así a las 4 rpm deseadas.

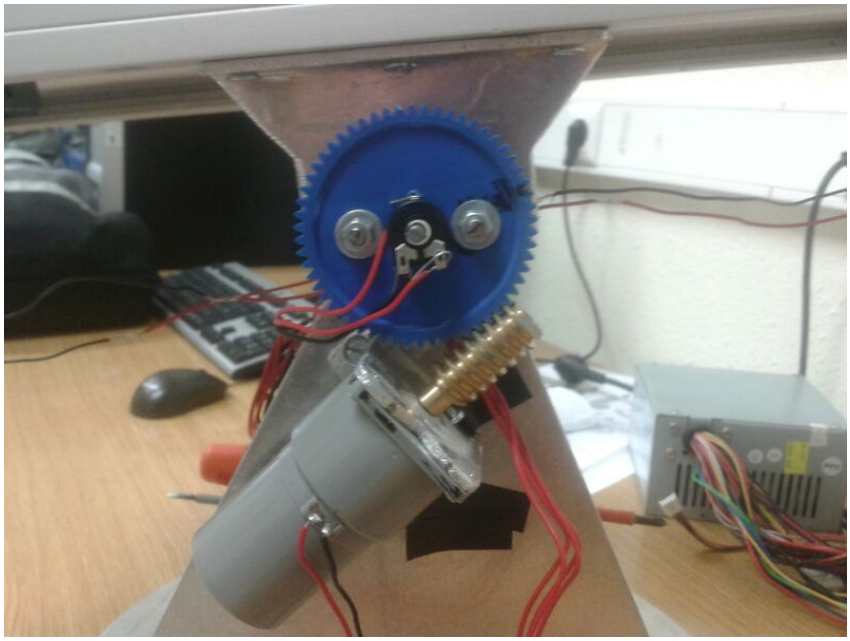


Ilustración 14: Reductora superior

Complemento:

Hemos colocado un tope para el eje de abajo, para evitar que sobrepase de vueltas, es un micro-interruptor. Cuando no está pulsado se encuentra abierto (0 voltios), y cuando se pulsa se encuentra cerrado (5 voltios). También cuando las fotocélulas no les llega nada de luz el eje se inicializa a la posición donde se encuentra el interruptor.

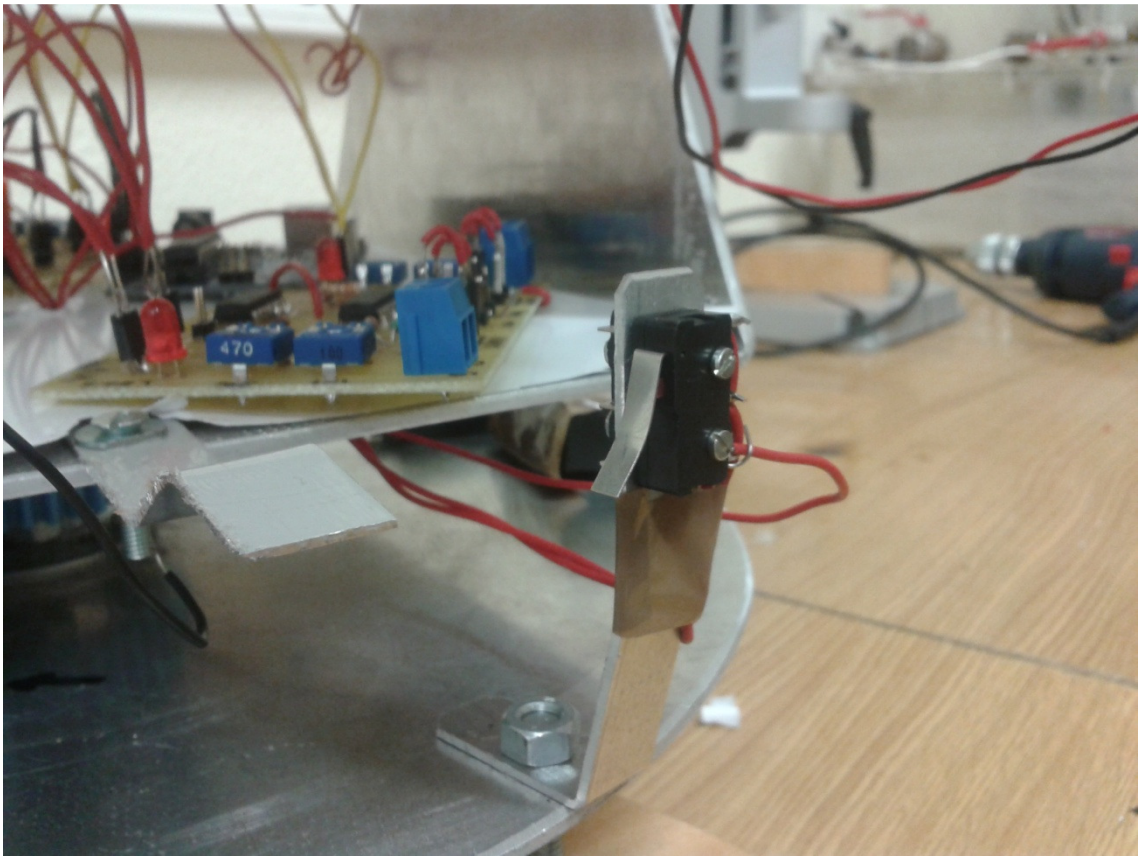


Ilustración 15: Micro-interruptor

MODELO Y DESCRIPCIÓN MATEMÁTICA

4.1.- Función de transferencia.

Una función de transferencia es un modelo matemático que a través de un cociente relaciona la respuesta de un sistema (modelada) a una señal de entrada o excitación (también modelada). En la teoría de control, a menudo se usan las funciones de transferencia para caracterizar las relaciones de entrada y salida de componentes o de sistemas que se describen mediante ecuaciones diferenciales lineales e invariantes en el tiempo.

La función de transferencia de un sistema lineal e invariante en el tiempo (*LTI*), se define como el cociente entre la transformada de *Laplace* de la salida y la transformada de *Laplace* de la entrada, bajo la suposición de que las condiciones iniciales son nulas.

En nuestro caso, tomaríamos como entrada la fuerza par motor y como salida el ángulo formado en el movimiento, considerando siempre desplazamientos muy pequeños.

Por definición una función de transferencia se puede determinar según la expresión:

$$H(s) = \frac{Y(s)}{X(s)}$$

Donde '*H(s)*' es la función de transferencia, '*Y(s)'* es la transformada de *Laplace* de la respuesta y '*X(s)'* es la transformada de *Laplace* de la señal de entrada.

Para llegar a dicha función de transferencia debemos sacar la ecuación física de nuestro sistema, a través de la segunda ley de Newton (ecuación del movimiento de rotación de un sólido rígido), todo ello descrito en los siguientes apartados.

4.2.- Ecuación del movimiento de rotación de un sólido rígido.

La variación del estado de rotación de un sólido viene determinada por la variación de su velocidad angular, si queremos describir el movimiento tenemos que encontrar la ecuación que nos permita calcular la aceleración angular, es la siguiente:

$$\sum \vec{r}\, x\, \vec{F}(ext) = I\vec{\alpha}$$

Esta es dicha ecuación y será utilizada para plantear la ecuación que rige el movimiento de nuestro sistema. Como se observa en la ecuación, lo siguiente será hallar el momento de inercia total del sistema (I).

4.3.- Momento de inercia.

El momento de inercia es una medida de la inercia rotacional de un cuerpo. Cuando un cuerpo gira en torno a uno de los ejes principales de inercia, la inercia rotacional puede ser representada como una magnitud escalar llamada momento de inercia.

EJE SUPERIOR:

En nuestro sistema del eje superior tenemos tres elementos que aportan un cierto momento de inercia, cada uno de estos, que luego serán mencionados, aporta una inercia que se calcula de forma independiente y que finalmente se sumarán para hallar el momento de inercia del sistema en global.

Veamos los distintos elementos que intervienen en el eje superior:

- Placas de aluminio a ambos lados

En este caso se pueden aproximar a un rectángulo, el cual, su momento de inercia es el siguiente:

$I = \sum x^2\, m$; 'm' = masa de la partícula; 'x' = distancia al eje de rotación

Placa rectangular de masa 'M' y de lados 'a' y 'b', tomando un elemento de masa que dista 'x' del eje, longitud 'a' y anchura 'dx'.

$$dm = \frac{M}{a\,b}\, a\, dx = \frac{M}{b}\, dx$$

Su momento de inercia:

$$I_1 = \int_{\frac{-b}{2}}^{\frac{b}{2}} \frac{M}{b}\, x^2\, dx = \frac{1}{12}\, M\, b^2$$

Tenemos como incógnita la masa, asique la hallaremos relacionando densidad del aluminio y volumen de las figuras a través de la siguiente fórmula:

$$\rho = \frac{M}{V}$$

La densidad del aluminio es un dato conocido, 2.700 kg/m^3.

Ahora pasamos a hallar el volumen que es el producto del ancho por alto y por largo, que al realizar las medidas pertinentes son los siguiente valores respectivamente, 10 cm, 5 cm y 0,4 cm; el producto entre estos datos es 0,00002 m^3.

Ahora ya podemos hallar la masa:

M = ρ V = 2.700 kg/m^3 0,00002 m^3 = 0,054 kg

Ahora ya disponemos de todos los datos para hallar el momento de inercia:

$$I_1 = \frac{1}{12} M\, b^2 = \frac{1}{12}\, 0{,}054\, kg\, (0{,}1)^2 = 0{,}000054\, kg\, m^2$$

Como son dos: 0,00009 kg m^2.

- Placa solar

Para la placa solar tendremos que hallar el momento de inercia de un paralepípedo, su fórmula es la siguiente:

$$I_2 = \int_{\frac{-c}{2}}^{\frac{c}{2}} (\frac{1}{12}\, b^2 + x^2)\, \frac{M}{c}\, dx = \frac{M}{12}\, (b^2 + c^2)$$

Sabiendo el modelo de la placa, tenemos que su masa es 1,6 kg y sus medidas son 48 cm, 2,5 cm y 27,3 cm. Ahora simplemente sustituimos en la fórmula anterior:

$$I_2 = \frac{1{,}6\, kg}{12}\, (0{,}273^2 + 0{,}025^2) = 0{,}0102\, kg\, m^2.$$

- Varilla o cilindro

Vamos a calcular el momento de inercia de un cilindro de masa '*M*', radio '*R*' y longitud L respecto de su eje.

Tomamos un elemento de masa que dista '*x*' del eje de rotación. El elemento es una capa cilíndrica cuyo radio interior es '*x*' y longitud '*L*'. La masa dm que contiene esta capa es:

$$dm = \frac{M}{\pi\, R^2\, L} 2\pi x\, dx\, L = \frac{2M}{R^2}\, x\, dx$$

Tenemos que hallar la masa como en el caso anterior, a través de la misma fórmula que relaciona masa, volumen y densidad del hierro en este caso (7874 kg/m^3).

El volumen de un cilindro se halla con la siguiente ecuación:

$$V = \pi R^2 L = \pi\, 0{,}004^2\, 0{,}3 = 0{,}0000151\, m^3$$

Su masa es la siguiente:

$$M = \rho\, V = 7874\ \text{kg}/m^3\ 0{,}0000151\, m^3 = 0{,}1187\, kg$$

El momento de inercia es el siguiente:

$$I_3 = \int x^2\, dm = \int_0^R \frac{2M}{R^2}\, x^3\, dx = \frac{1}{2}\, M\, R^2 = \frac{1}{2}\, 0{,}1187\, kg\, 0{,}004^2 = 0{,}000001\, kg\, m^2$$

Ahora solo falta sumar los tres momentos de inercia, el aportado por la placa, la varilla y las placas de aluminio, para obtener el momento de inercia total, y así, emplear el dato en la obtención de la función de transferencia:

$$I_t = I_1 + I_2 + I_3 = 0{,}0102\, kg\, m^2 + 0{,}0001\, kg\, m^2 + 0{,}000001\, kg\, m^2 = 0{,}0103\, kg\, m^2$$

EJE INFERIOR:

En nuestro eje inferior tenemos dos elementos que aportan un cierto momento de inercia, cada uno de estos, aporta una inercia que se calcula de forma independiente y que finalmente se suman para hallar el momento de inercia del sistema en global.

- Placa solar:

Hallamos su momento de inercia como en el eje superior, con la diferencia que tomamos como referencia el eje inferior, que en este caso es perpendicular a la placa.

$$I_1 = \int_{\frac{-c}{2}}^{\frac{c}{2}} \left(\frac{1}{12}\, b^2 + x^2\right) \frac{M}{c}\, dx = \frac{M}{12}\, (b^2 + c^2)$$

Sustituimos en la ecuación con las medidas realizadas anteriormente:

$$I_1 = \frac{1{,}6\ kg}{12}\ (0{,}273^2 + \ 0{,}48^2) = \ 0{,}0406\ kg\ m^2.$$

- Soporte en forma de "U":

Para este caso, en particular, utilizamos la fórmula del paralepípedo para hallar el momento de inercia, ya que la base de este soporte es un paralepípedo, y los laterales son dos triángulos isósceles, que al ser dos, al estar simétricos en torno al eje de movimiento y tener las mismas medidas que la base, la inercia es la misma que la que hallamos en la base en forma de paralepípedo.

Sus medidas son 20 cm de largo y 23 cm de ancho (densidad: 2.700 kg/m^3).

Antes hallamos la masa del soporte, siguiendo el mismo procedimiento como en los casos anteriores, a través de la densidad y el volumen, por consiguiente, no es necesario mostrar el procedimiento, obteniendo como resultado final 0,99 kg, es decir, 1 kg.

$$I_2 = \frac{1\ kg}{12}\ (0{,}2^2 + \ 0{,}23^2) = \ 0{,}00774\ kg\ m^2.$$

El momento de inercia de la varilla del eje superior se puede considerar cero perfectamente.

Ahora sumamos los dos momentos de inercia para obtener el momento de inercia total que es el siguiente:

$$I_t = \ I_1 + \ I_2 = 0{,}0406\ kg\ m^2 + 0{,}00774\ kg\ m^2 = 0{,}04834\ kg\ m^2$$

En el siguiente apartado sacaremos la ecuación física de ambos ejes, y a partir de ahí, trabajar en frecuencia para poder sacar los modelos matemáticos.

4.4.- Desarrollo y obtención de la función de transferencia.

EJE SUPERIOR:

Para obtener la función de transferencia, lo primero que hay que tener claro, es que hay que relacionar la salida con la entrada. En nuestro caso el movimiento que realiza la placa solar con el eje central formando un ángulo θ con la fuerza par motor que tiene que realizar el motor empleado.

Lo siguiente será observar las fuerzas actuantes en el sistema del eje superior

$$Px = P \cdot \operatorname{sen} \alpha$$
$$Py = P \cdot \cos \alpha$$
$$P = m \cdot g$$

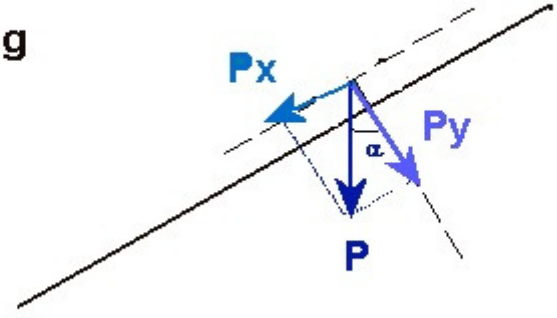


Fig.4.1.Descomposición de fuerzas eje superior

Para determinar nuestro modelo debemos emplear la ecuación del movimiento de rotación de un sólido hallada en apartados anteriores. Una vez aplicada esta ecuación obteniendo cada uno de sus parámetros habrá que aplicar la transformada de *Laplace* a dicha ecuación resultante, de esta manera, se podrá obtener la función de transferencia del modelo matemático.

Tendremos en cuenta que la placa solar está algo por encima de su eje, habiendo una pequeña distancia que habrá que medir, aún así, la tenemos colocada en mitad y demás fuerzas están compensadas a un lado y otro, por lo que sólo hacemos referencia a esta fuerza ($M_g \cos\theta \, d$), debida a la masa de la placa. La otra fuerza que tenemos y que se opone a la fuerza par motor es la fuerza de rozamiento debida a la fricción con el eje de giro, oponiendo resistencia al movimiento de la placa.

Una vez explicado ya podemos aplicar la ecuación resultante, que es la siguiente:

$$P_m - M_g \cos\theta \, d - F_{roz} = I\alpha$$

P_m: Fuerza del motor

M_g: Peso de la placa debido a la gravedad

θ : Ángulo de giro con respecto al eje

d : Distancia del centro de la placa al eje

F_{roz} : Fuerza de rozamiento

I : Momento de inercia total del sistema

α : Aceleración angular

Como tomamos desplazamientos de ángulos muy pequeños, tanto el seno de un ángulo, como en este caso, coseno, se pueden aproximar; el seno de un ángulo muy pequeño se considera 0, y en coseno, se considera 1, asique se puede aproximar al mismo ángulo ϑ.

La aceleración angular, por definición, en el movimiento plano del sólido rígido, al igual que la velocidad angular, tiene la dirección del eje de rotación y viene dada por la ecuación:

$$\alpha = \frac{d^2\theta}{dt^2}$$

Y, por último, la fuerza de rozamiento viene dada por:

$$F_{roz} = \beta \frac{d\theta}{dt}$$

Aclarado esto, la ecuación se puede escribir de la forma siguiente:

$$P_m - M_g \, \theta \, d - \beta \frac{d\theta}{dt} = I \frac{d^2\theta}{dt^2}$$

Dejamos el término de segundo grado con coeficiente 1, dividiendo toda la ecuación entre el momento de inercia y posteriormente aplicamos el teorema de diferenciación enunciado a continuación:

$$L\left[\frac{d^n f}{dt^n}\right] = s^n F(s) - \sum_{k=1}^{n} s^{n-k} f^{k-1}(0-)$$

Nos queda la siguiente ecuación:

$$\frac{1}{I} P_m(s) - \frac{M_g d}{I}\theta(s) - \frac{\beta}{I}[s\,\theta(s) - \theta(0)] = s^2\,\theta(s) - s\,\theta(0) - \theta(0)$$

Para obtener la función de transferencia hay que partir de condiciones iniciales nulas, entonces:

$$\frac{1}{I} P_m(s) - \frac{M_g d}{I}\theta(s) - \frac{\beta}{I}[s\,\theta(s)] = s^2\,\theta(s)$$

$$\frac{1}{I} P_m(s) = s^2\,\theta(s) + \frac{\beta}{I} s\theta(s) + \frac{M_g d}{I}\theta(s)$$

Ahora simplemente despejando, obtenemos la relación entre la entrada y la salida, obteniendo la siguiente función de transferencia:

$$\frac{\theta(s)}{P_m(s)} = \frac{{}^1/_I}{s^2 + \frac{\beta}{I} s + \frac{M_g d}{I}}$$

La función normalizada es la siguiente de un sistema de segundo orden:

$$F(s) = \frac{\theta(s)}{P_m(s)} = \frac{K}{s^2 + 2\xi w_n s + w_n^2}$$

Como incógnita tenemos el segundo y tercer miembro del denominador, asique igualando en las dos ecuaciones tenemos las siguientes relaciones:

$$2\xi w_n s = \frac{\beta}{I} s$$

$$w_n^2 = \frac{M_g d}{I}$$

En primer lugar, pasamos a hallar la frecuencia natural no amortiguada despejando de la segunda ecuación:

$$w_n = \sqrt{\frac{M_g d}{I}} = \sqrt{\frac{1.6Kg\ 10\frac{m}{s^2}0.07\ m}{0.0103}} = 10.428\frac{rad}{s}$$

En la ecuación del segundo término no tenemos el valor del coeficiente de rozamiento, tendremos que hallar el coeficiente de amortiguamiento ξ.

El coeficiente de amortiguamiento y la frecuencia natural no amortiguada están relacionados a través del tiempo de establecimiento t_s, que es el tiempo requerido para que la respuesta se mantenga alrededor de cierto rango alrededor del valor final (habitualmente el 5%).

$$t_s = \frac{4}{\xi w_n}$$

Una vez definido el tiempo de establecimiento, en nuestro sistema su valor se aproxima muy cercano a cero, siendo de 0.1 aproximadamente:

$$\xi = \frac{4}{t_s w_n} = \frac{4}{0.1s\ 10.428\frac{rad}{s}} = 3.83$$

Con este dato (ξ>1), sabemos que estamos ante un sistema sobre-amortiguado, y nos podemos hacer una idea de cómo será la respuesta ante una entrada escalón; deberíamos observar una respuesta no oscilatoria, que se va a parecer a la respuesta de un sistema de primer orden, con la diferencia de que la pendiente en el origen en este caso no va a ser cero, tampoco tendríamos sobre-impulso; y también nos podríamos hacer una idea de dónde estarán los polos acerca de su estabilidad; en este caso debería presentar dos polos reales.

Pasamos a escribir la ecuación completa con los datos calculados y a comprobar de manera fiable la respuesta del sistema ante una entrada escalón:

$$F(s) = \frac{\theta(s)}{P_m(s)} = \frac{97.09}{s^2 + 80s + 108.74}$$

Pasamos a usar Matlab, dónde usaremos los siguientes comandos:

Creamos la función de transferencia:

f=tf([97.09],[1 80 108.74])

f =

97.09

----

s^2 + 80 s + 108.7

Continuous-time transfer function.

Comando para observar la respuesta ante la entrada escalón:

Step(f)

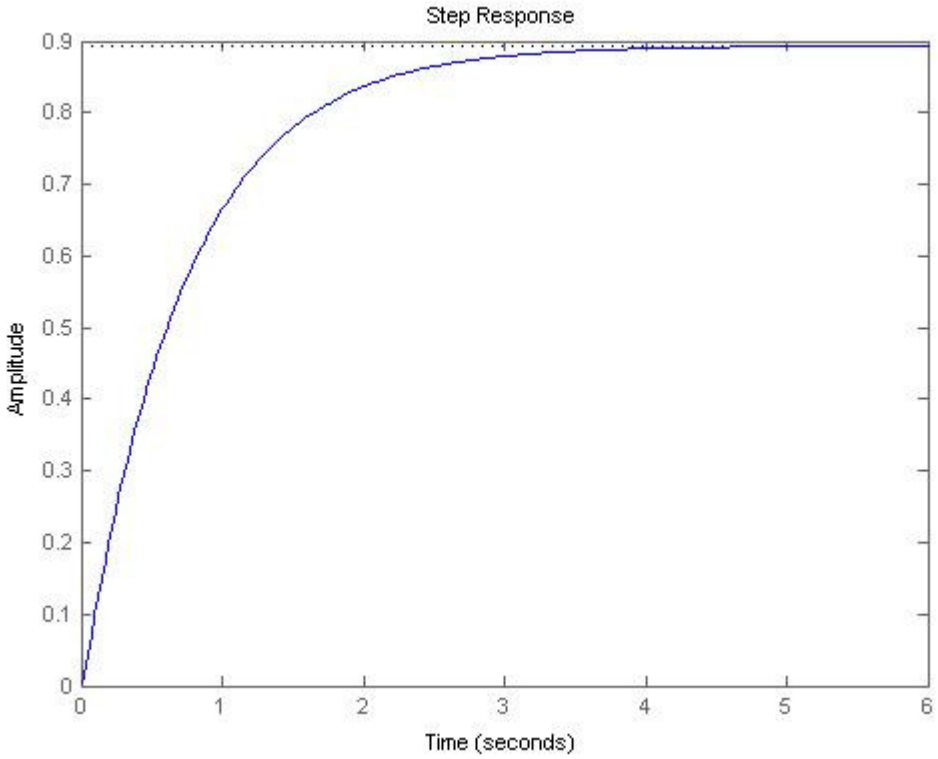


Fig.4.2.Respuesta a un escalón eje superior

Tenemos un sistema sobre-amortiguado, con respuesta no oscilatoria, se parece a un sistema de primer orden, apreciamos también que nos encontramos ante un sistema lento.

Ahora vamos a comprobar la estabilidad del sistema mediante el lugar geométrico de las raíces.

El lugar de las raíces es el lugar geométrico de los polos y ceros de una función de transferencia a medida que se varía la ganancia del sistema en un determinado intervalo.

El método del lugar de las raíces permite determinar la posición de los polos de la función de transferencia a lazo cerrado para una determinada ganancia a partir de la función de transferencia a lazo abierto.

Recordemos que un sistema es estable si todos sus polos se encuentran en el semiplano izquierdo del plano 's'.

Usando el comando 'rlocus(f)' en la consola de Matlab, obtenemos la siguiente gráfica:

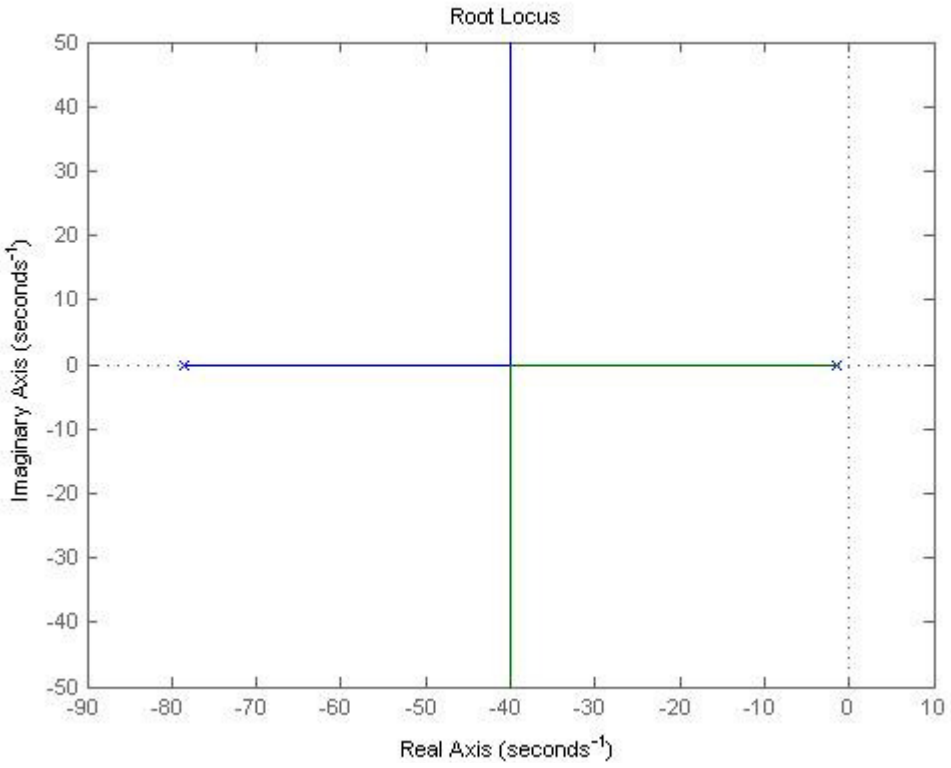


Fig.4.3.Lugar de las raíces eje superior

Como debe ser, presenta dos polos reales a la izquierda del eje imaginario, por consiguiente podemos determinar que tenemos un sistema estable.

EJE INFERIOR:

Como explicamos en el eje superior, para obtener la función de transferencia debemos relacionar la salida entre la entrada, en este caso como el anterior, es relacionar el ángulo ϑ realizado en el movimiento con la fuerza que realiza nuestro motor. Para este caso se presenta un movimiento circular.

Para determinar el modelo del eje inferior debemos emplear otra vez la ecuación de movimiento de rotación de un sólido. Una vez aplicada esta ecuación con sus parámetros aplicamos la transformada de *Laplace* a dicha ecuación, y así, obtendremos la función de transferencia para este sistema con dicho movimiento.

En primer lugar, debemos saber que estamos trabajando sobre el eje inferior, en este caso, tendremos en cuenta el eje superior ya que intervendrá en el desarrollo de la ecuación física, en este caso, el eje superior lo fijaremos a un ángulo constante, y en el peor de los casos, que sería aproximadamente un ángulo α=45º (π/4 rad). Misma descomposición de fuerzas que antes, pero con un ángulo de π/4 rad, todo ello relacionado con el ángulo de movimiento θ para este eje. Otra fuerza vinculante es la debida a la masa por la fuerza gravitatoria del soporte en forma de "U" cuyo movimiento es circular y descomposición de fuerzas describimos en la siguiente figura:

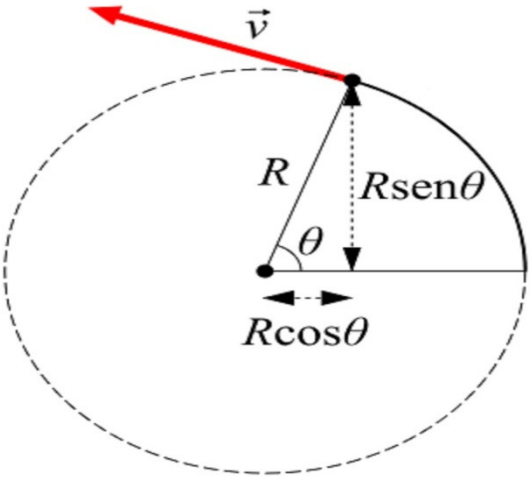


Fig.4.4.Descomposición de fuerzas eje inferior

La última fuerza que hay es la fuerza de rozamiento debida a la fricción con el eje de giro, oponiendo resistencia al movimiento realizado por el soporte.

La ecuación física resultante es la siguiente:

$$P_m - \left(\left[M_1 \left(sen\frac{\pi}{4}\right) d_1\right] \cos\theta + M_g \cos\theta \ d_2\right) - F_{roz} = I\alpha$$

P_m: Fuerza del motor
M_1: Masa placa solar
d_1: Distancia al eje
θ: Ángulo de giro
M_g: Masa del soporte
F_{roz}: Fuerza rozamiento
I : Momento de inercia total del sistema
α : Aceleración angular

Consideramos lo mismo que en el eje anterior, tomamos desplazamientos de ángulos muy pequeños, y sustituimos por las ecuaciones que definen la aceleración angular y fuerza de rozamiento descritas anteriormente, quedándonos como ecuación la siguiente:

$$P_m - \left(0{,}79\theta + M_g\theta \ d_2\right) - \beta\frac{d\theta}{dt} = I\frac{d^2\theta}{dt^2}$$

Dejamos término de segundo grado con coeficiente 1 y aplicamos el teorema de diferenciación:

$$L\left[\frac{d^n f}{dt^n}\right] = s^n F(s) - \sum_{k=1}^{n} s^{n-k} f^{k-1}(0-)$$

Ecuación resultante:

$$\frac{1}{I}P_m(s) - \frac{(M_g d) + 0{,}79}{I}\theta(s) - \frac{\beta}{I}[s\,\theta(s) - \theta(0)] = s^2\,\theta(s) - s\,\theta(0) - \theta(0)$$

Consideramos condiciones iniciales nulas, entonces:

$$\frac{1}{I}P_m(s) - \frac{0{,}79 + (M_g d)}{I}\theta(s) - \frac{\beta}{I}[s\,\theta(s)] = s^2\,\theta(s)$$

$$\frac{1}{I}P_m(s) = s^2\,\theta(s) + \frac{\beta}{I}s\,\theta(s) + \frac{0{,}79 + (M_g d)}{I}\theta(s)$$

Como hemos ido explicando a lo largo del eje inferior y superior, despejamos relacionando salida entre la entrada:

$$\frac{\theta(s)}{P_m(s)} = \frac{{}^{1}/_{I}}{s^2 + \frac{\beta}{I}s + \frac{0{,}79 + (M_g d)}{I}}$$

Como en el eje anterior, relacionamos la función normalizada con la función obtenida anteriormente para hallar los valores que no faltan:

$$F(s) = \frac{\theta(s)}{P_m(s)} = \frac{K}{s^2 + 2\xi w_n s + w_n^2}$$

$$2\xi w_n s = \frac{\beta}{I}s \qquad\qquad w_n^2 = \frac{(M_g d) + 0{,}79}{I}$$

Pasamos a hallar la frecuencia natural no amortiguada ya que disponemos de todos los valores para hallarla:

$$w_n = \sqrt{\frac{(M_g d) + 0{,}79}{I}} = \sqrt{\frac{\left(1Kg\ 10\frac{m}{s^2}0.15\ m\right) + 0{,}79}{0.04834}} = 6.883\frac{rad}{s}$$

Como ya hicimos en el eje anterior, el coeficiente de amortiguamiento y la frecuencia natural no amortiguada están relacionados a través del tiempo de establecimiento t_s, que es el tiempo requerido para que la respuesta se mantenga alrededor de cierto rango alrededor del valor final (habitualmente el 5%).

El tiempo de establecimiento de establecimiento como en el eje descrito anteriormente se aproxima a cero siendo su valor de 0.1 segundos

$$\xi = \frac{4}{t_s w_n} = \frac{4}{0.1s\ 6{,}883\frac{rad}{s}} = 5.81$$

Lógicamente, y como era de suponer, nos encontramos con un sistema similar al que describimos con anterioridad con el eje superior, ya que no dista mucho el sistema del eje superior con el del eje inferior, diferentes movimientos pero sistemas similares; mismas conclusiones sacados anteriormente con respuesta ante una entrada escalón y estabilidad similar. Aún así, obtendremos sus debidas gráficas para cerciorarnos de ello.

Antes pasamos a escribir la función de transferencia completa:

$$F(s) = \frac{\theta(s)}{P_m(s)} = \frac{20{,}687}{s^2 + 80s + 47{,}37}$$

Comandos y gráfica en Matlab:

Crear función de transferencia:

```
>> f=tf([20.687],[1 80 47.37])


f =


      20.69

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

  s^2 +80 s + 47.37


Continuous-time transfer function.
```

Comando para mostrar gráfica ante un escalón:

```
Step(f)
```

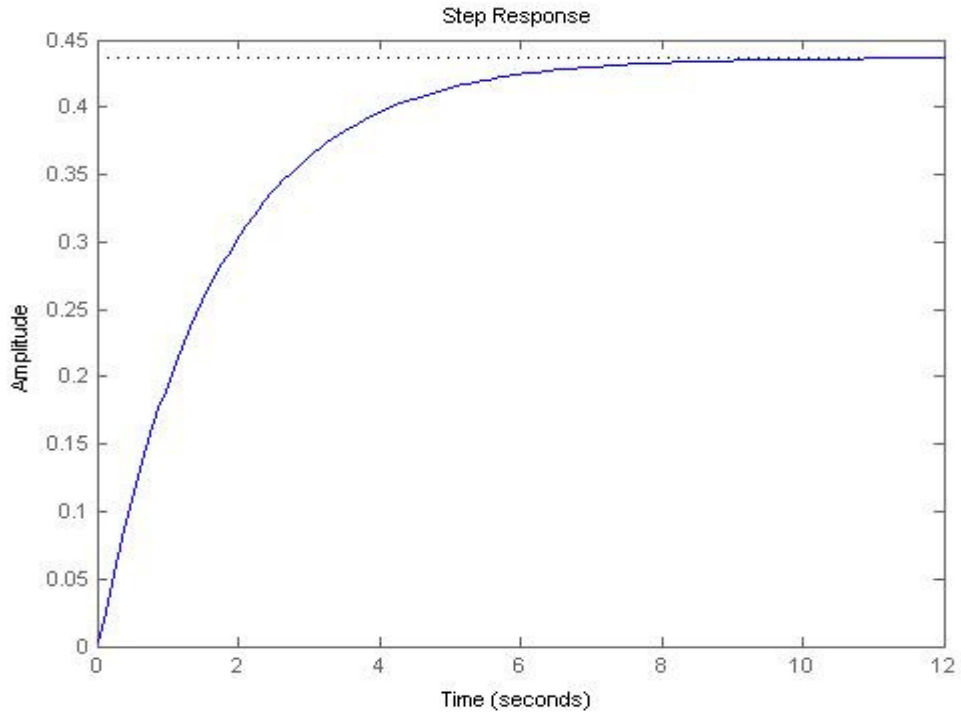


Fig.4.5.Respuesta a un escalón eje inferior

Como vimos en el eje superior, tenemos un sistema sobre-amortiguado, con respuesta no oscilatoria, se parece a un sistema de primer orden, apreciamos también que nos encontramos ante un sistema lento.

Ahora comprobamos su estabilidad, mediante el método del lugar de las raíces usando el comando *'rlocus(f)'*.

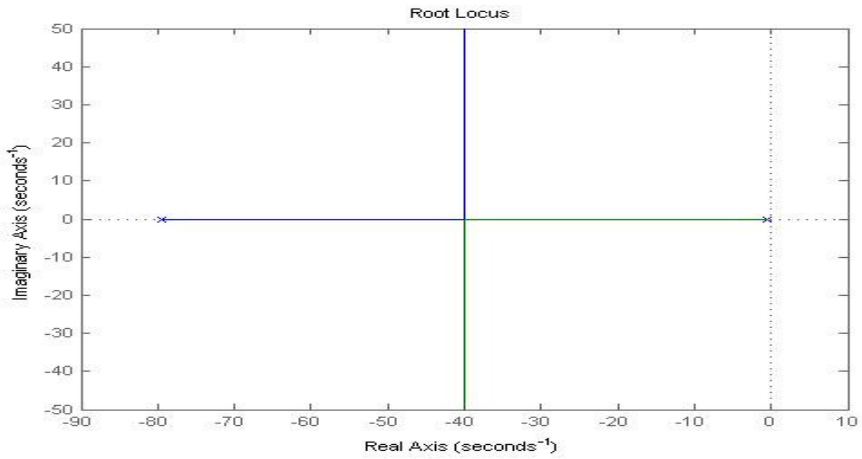


Fig.4.6.Lugar de las raíces eje inferior

Tenemos dos polos reales colocados en la parte izquierda del eje imaginario, asique tenemos un sistema estable.

FUNCIONAMIENTO DEL SISTEMA DE CONTROL

5.1.- Sistema de control.

Un sistema de control es el conjunto de equipos y componentes, que van a permitir llevar a cabo las operaciones de control, estas operaciones de control es el conjunto de acciones que buscan mantener una variable dentro de unos patrones de funcionamiento deseados. El control es la acción ejercida con el fin de poder mantener una variable dentro de un rango de valores predeterminados.

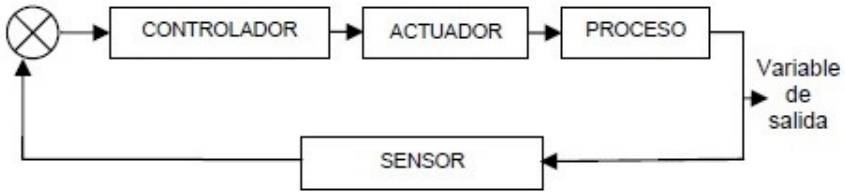


Fig.5.1.Sistema de control

Con este pequeño esquema nos hacemos una idea del sistema de control, pasamos a definir cada una de las partes para que así nos quede más claro:

Controlador: Es aquel instrumento que compara el valor medido con el valor deseado, en base a esta comparación calcula un error, para luego actuar a fin de corregir este error. Tiene por objetivo elaborar la señal de control.

Actuador: Es aquel equipo que sirve para regular la variable de control y ejecutar la acción de control, es conocido como elemento final de control, hay varios tipos, en nuestro caso es un actuador eléctrico (motor DC).

Proceso: Esta referido al equipo que va a ser automatizado, en nuestro caso, nuestro montaje de seguidor solar.

Sensor: Es un elemento de medición de parámetros o variables del proceso. Los sensores pueden ser usados también como indicadores, para transformar la señal medida en señal eléctrica. En nuestro caso es un sensor de luz (fotorresistencia).

Dentro de este apartado pasamos a explicar los controladores y el circuito linealizador del sensor.

5.1.1.- Controladores PID.

El controlador '*PID*' (Proporcional, Integral, Derivativo) es un controlador realimentado cuyo propósito es hacer que el error en estado estacionario, entre la señal de referencia y la señal de salida de la planta, sea cero de manera asintótica en el tiempo, lo que se logra mediante el uso de la acción integral. Además el controlador tiene la capacidad de anticipar el futuro a través de la acción derivativa que tiene un efecto predictivo sobre la salida del proceso.

Los controladores *'PID'* son suficientes para resolver el problema de control de aplicaciones en la industria, particularmente cuando la dinámica del proceso lo permite, y los requerimientos de desempeño son modestos.

- Acción proporcional(P):

Es un control que se basa en la ganancia aplicada al sistema, se basa en el principio de que la respuesta del controlador debe ser proporcional a la magnitud del error. No corrige ni elimina perturbaciones, puede aumentar o atenuar la señal de error. Se representa a través del parámetro '*Kp*' y define la fuerza o potencia con que el controlador reacciona frente a un error.

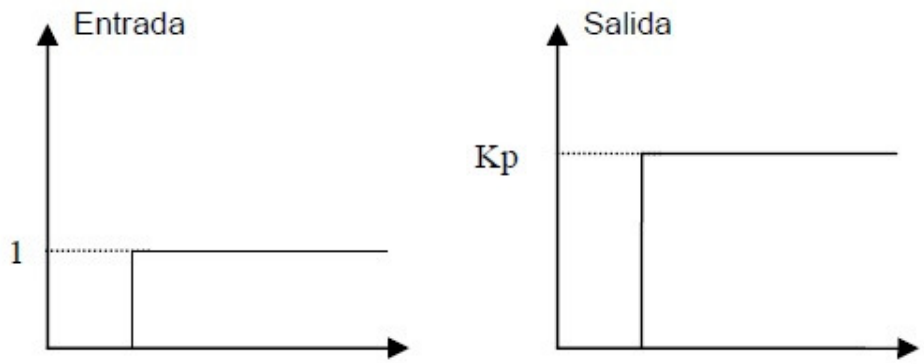


Fig.5.2.Controlador proporcional

Un controlador proporcional responde rápidamente ante el error del sistema pero no es capaz de eliminar completamente las perturbaciones, o eliminar completamente el error. Puede producir inestabilidad.

La ley de control es simplemente proporcional al error de control, se reduce a:

$$u(t) = Ke(t)$$

- Acción integral(I):

Este tipo de controlador anula errores y corrige perturbaciones, mediante la búsqueda de la señal de referencia, necesita de un tiempo '*Ti*' para localizar dicha señal. Se representa mediante el término '*Ki*' que es el coeficiente de acción integral y es igual a '*1/Ti*'.

Su forma matemática sería la siguiente:

$$u(t) = T_i \int_0^t e(t)dt$$

La función principal de la acción integral es asegurar que la salida del proceso concuerde con la referencia en estado estacionario. Con el controlador proporcional, normalmente existiría un error en estado estacionario. Con la acción integral, un pequeño error positivo siempre producirá un incremento en la señal de control y, un error negativo siempre dará una señal decreciente sin importar cuánto de pequeño sea el error.

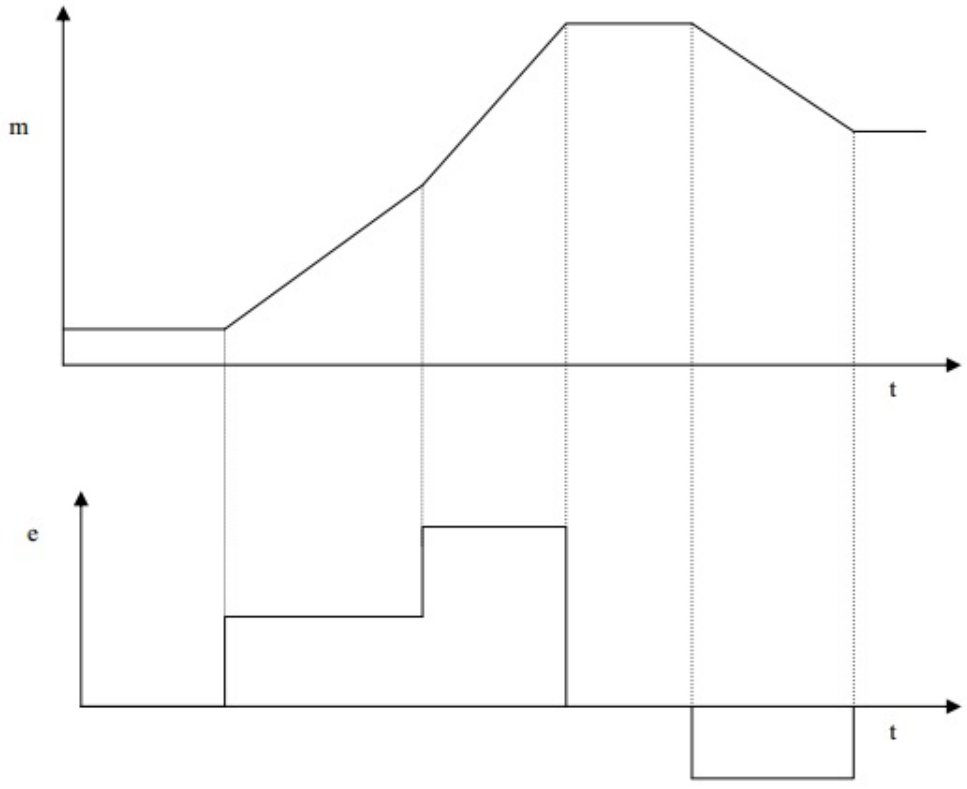


Fig.5.3.Controlador integral

- Acción derivativa(D):

Este controlador por sí solo no es utilizado, necesita estar junto al proporcional y al integral. Sirve para darle rapidez o aceleración a la acción de control. Necesita de una diferencial de tiempo '*Td*' para alcanzar la señal de referencia, se representa mediante el término '*Kd*' que es el coeficiente de acción derivativa y es igual a '*1/Td*'.

Su representación es la siguiente:

$$u(t) = T_d \frac{de(t)}{dt}$$

La acción derivativa tiene la desventaja de que amplifica las señales de ruido de alta frecuencia y puede producir saturación en el actuador.

La acción de un controlador con acción proporcional y derivativa puede ser interpretada como si el control proporcional fuese hecho para predecir la salida del proceso.

- Controlador proporcional-derivativo(PD):

Está compuesto por una acción proporcional más una acción derivativa. La acción derivativa tiene efecto cuando se producen cambios en el error.

$$m(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt}$$

El tiempo derivativo *Td* es el intervalo de tiempo en el que la acción de velocidad de variación del error se adelanta al efecto de acción proporcional. De alguna forma intenta anticiparse a los cambios en el error, mediante las lecturas de los errores antiguos. Incrementa la estabilidad del sistema.

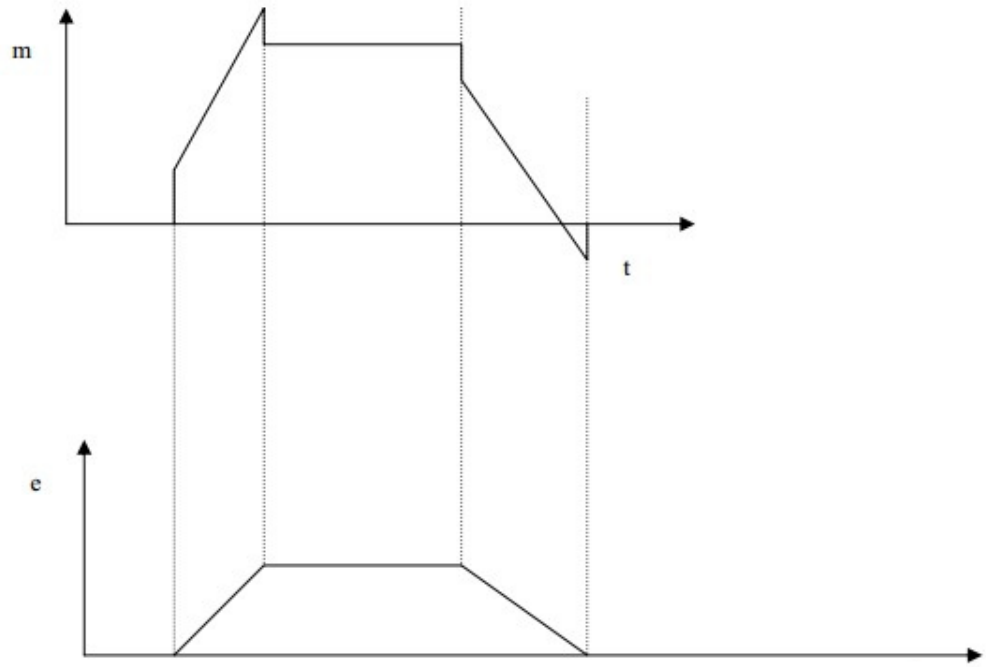


Fig.5.3.Controlador PD

El control proporcional-derivativo se usa en procesos con cambios de carga repentinos y en los que el control proporcional no es capaz de mantener el error dentro de unos niveles aceptables.

Aunque no afecta directamente al error estacionario, añade amortiguamiento al sistema y permite usar valores de ganancia más elevados, lo cual produce una mejoría en aquel.

Debido a que el control derivativo actúa sobre la variación del error y no sobre el error en sí, nunca se utiliza solo, siempre en combinación con la acción proporcional, o la proporcional-integral.

- Controlador proporcional-integral-derivativo(PID):

En este controlador se combinan las tres acciones para intentar aprovechar las ventajas que ofrecen cada una de ellas.

$$m(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt} + \frac{K_p}{T_i}\int_0^t e(t)dt$$

La acción proporcional aumenta la dinámica del controlador, la acción diferencial se anticipa a la acción integral en función de la variación del error, la acción integral aumenta según se acumula el error en el tiempo.

Generalmente la acción integral se utiliza para eliminar el error estacionario, mientras que la acción derivativa tiende a estabilizar el sistema y suaviza el error producido por cambios bruscos de la señal de entrada.

5.1.2.- Circuito linealizador-regulador de los sensores fotosensibles.

La curva característica de una fotorresistencia no es lineal (Recordemos: $R = AE^{-\alpha}$), la resistencia en función de la luz recibida no varía de una forma lineal, por consiguiente, la tensión que vamos a leer en el Arduino, no variará de una manera lineal en función de la resistencia de nuestro sensor.

Para ello vamos a diseñar un circuito linealizador, donde se vea como varía la tensión de manera proporcional a los Lux recibidos en la '*LDR*'. El circuito es el siguiente:

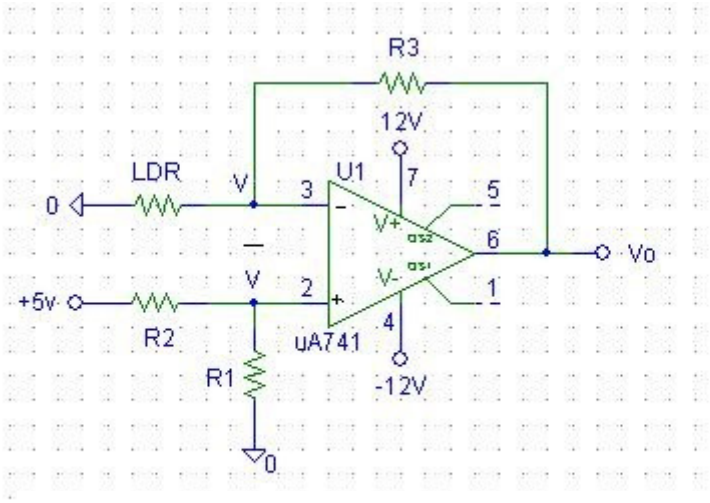


Fig.5.4.Esquema circuito linealizador

En la entrada positiva del amplificador operacional tenemos un divisor de tensión donde ajustamos la tensión '*V*', al ajustar esta tensión, también la estamos ajustando en el punto que entra por el terminal negativo del amplificador. Entonces, al variar la resistencia del sensor a partir de la intensidad de luz recibida, tendremos menos resistencia a más Lux, por consiguiente menos tensión *'V'*, y así, variamos la tensión $'V_o'$ en función de la '*LDR*', que será mayor de manera lineal, que lo veremos a continuación al analizar el circuito. (Nota: A más Lux, menos resistencia, más $'V_o'$ y viceversa; también deducimos que a menos R_1 , habrá menor tensión *'V'*, conviene

saber este concepto porque más adelante sustituiremos la resistencia por un potenciómetro, con su debida explicación).

Analizamos el circuito:

$$\frac{V_o - V}{R_3} = \frac{V}{R_{LDR}}$$

$$R_{LDR}(V_o - V) = R_3 - V$$

En este punto vamos a realizar una aproximación de la fotorresistencia para que sea más fácil de realizar nuestro diseño. Una aproximación inicial será la siguiente:

$$\frac{X}{L}(V_o - V) = R_3 V$$

Haremos que: 'X' = R_3, nos queda:

$$V_o - V = LV$$

$$V_o = LV + V$$

Con esta fórmula final se ve claramente como la tensión de salida leída por el Arduino varía de forma lineal con los Lux recibidos (L). Ahora debemos aproximar la variable *'X/L'* realizando una modelización de nuestra fotorresistencia utilizada.

- Modelización del sensor (LDR).

Como vimos en apartados anteriores, la LDR viene definida por la siguiente función:

$$R = A\,L^{-a}$$

Sacaremos dos puntos con la hoja del fabricante mostrada a continuación y obtendremos las constantes 'A' y 'a'.

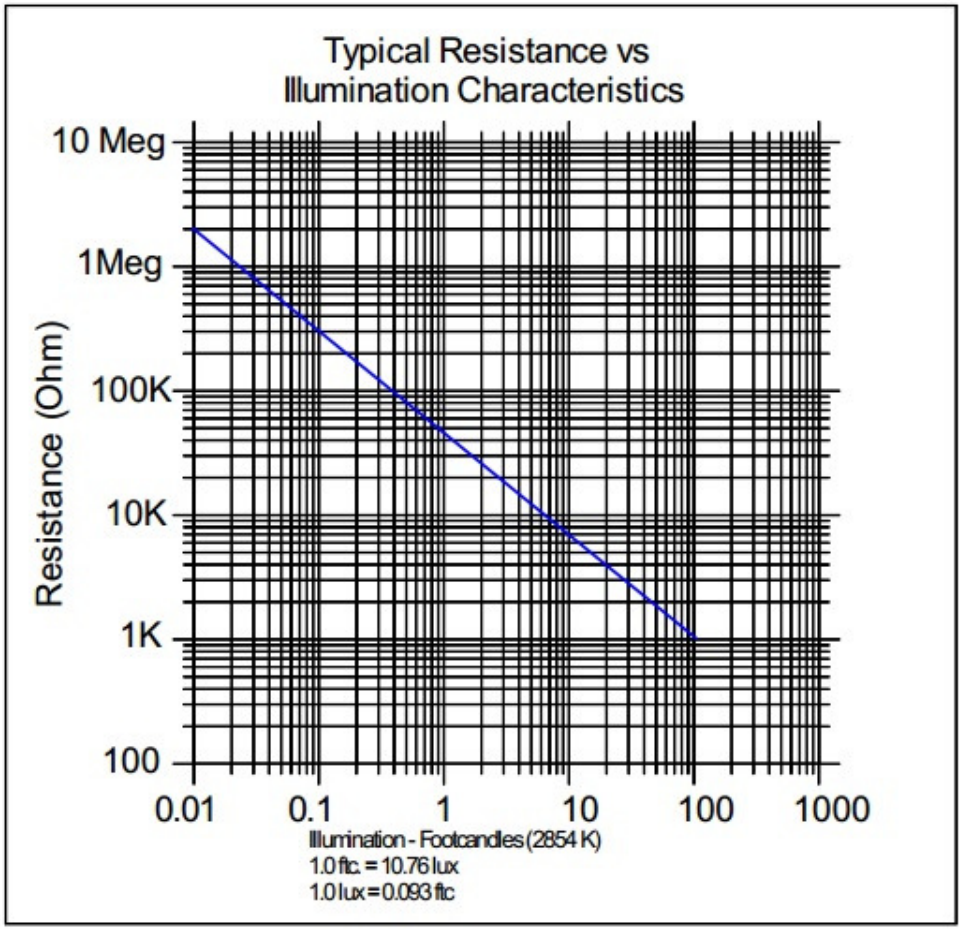


Fig.5.5.Gráfica proporcionada por el fabricante de nuestra LDR

Punto B: B (45000, 10.76) $\quad 45000 = A\ 10{,}76^{-a}$

Con este sistema de dos ecuaciones y dos incógnitas, primero hallamos 'a':

$$\frac{1000}{45000} = \frac{1076^{-a}}{10.76^{-a}}$$

$$0.022 = 100^{-a}$$

$$\ln 0.022 = -a \ln 100$$

$$3.81 = a\ 4.60$$

$$a = \frac{3.81}{4.60} = 0.83$$

Hallamos 'A':

$$1000 = A\ 1076^{-0.83}$$

$$A = \frac{1000}{1076^{-0.83}} = 333.333$$

La ecuación que modela nuestra fotorresistencia es la siguiente:

$$R_{LDR} = 333.333\ L^{-0.83}$$

Vamos a realizar dos tramos, uno para luz artificial de 1 a 1000 Lux y otro de luz solar de 1000 a 100000 Lux. Para simplificar operaciones y que el diseño sea más sencillo, hallamos anteriormente que la aproximación debe ser '*X/L*'.

Para el primer tramo, después de hacer varias aproximaciones, su valor final es *450.000/L*. A través de *Matlab*, vemos la gráfica nuestra ecuación real y la de nuestra aproximación y vemos como se aproximan perfectamente.

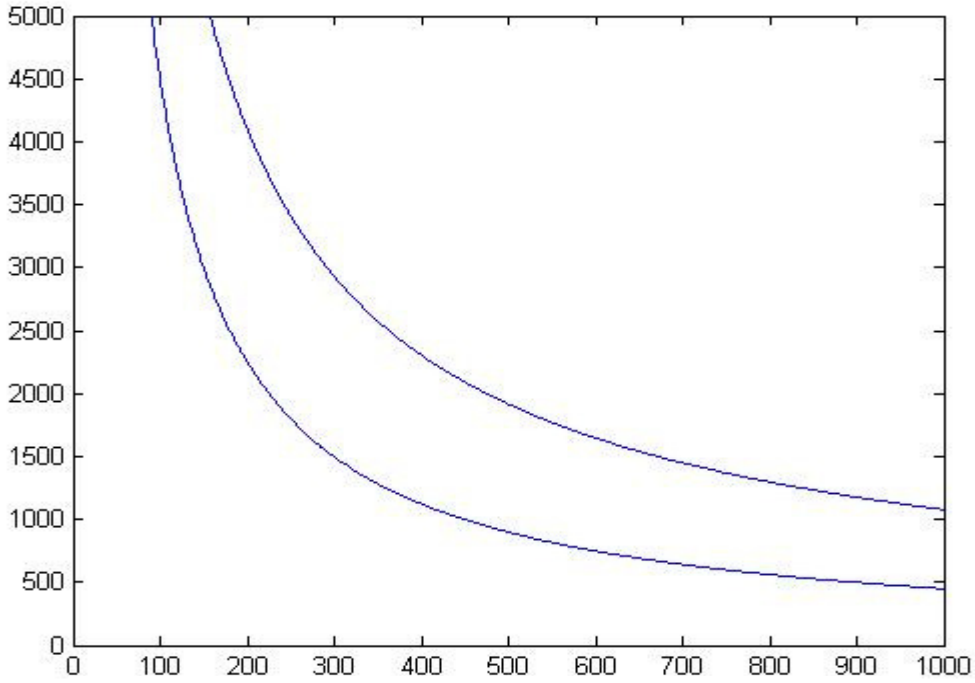


Fig.5.6.Aproximación del tramo 1

Para el segundo tramo, como de 1000 a 100000 Lux hay mucho rango de valores, realizamos la raíz de la variable '*L*', la ecuación aproximada sería la siguiente:

$$10000/\sqrt{L}$$

Ahora mostramos las dos ecuaciones mediante *Matlab* y veremos cómo son muy aproximadas:

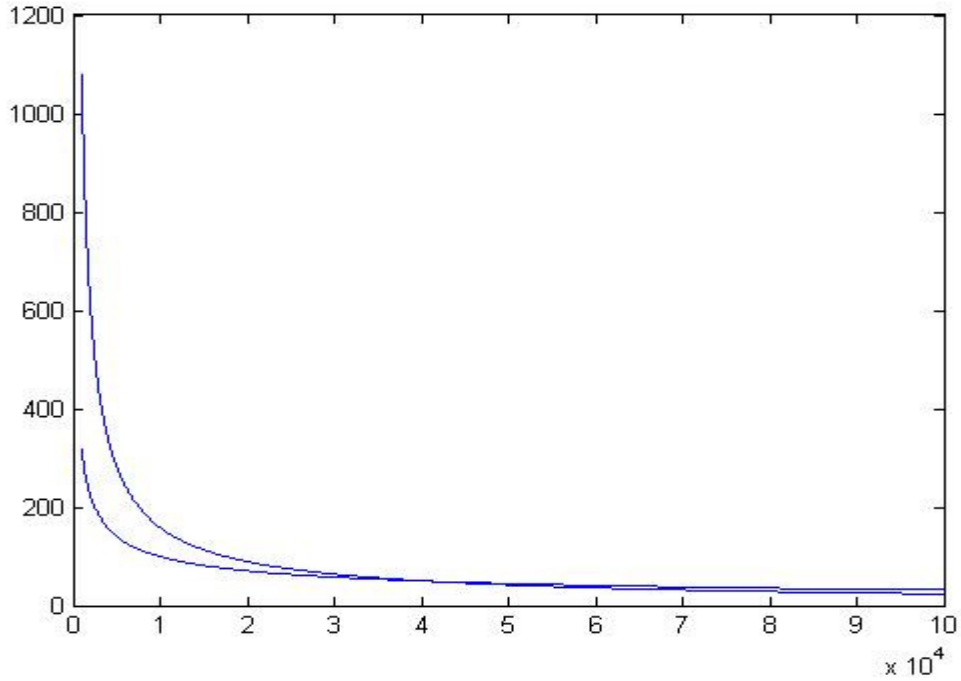


Fig.5.7.Aproximación del tramo 2

- Tramo 1:

En el primer tramo tendremos una variación de 1:1000 Lux, rango para luz artificial, queremos que la tensión $'V_o'$ sea de 5 voltios cuando los Lux de la sala artificial proporcionan intensidad máxima. La tensión $'V_o'$ debe estar entre 0 y 5 voltios ya que es el rango de lectura analógica que posee el Arduino. Fijamos $'R_3'$ a 450KΩ como ya explicamos anteriormente.

Hallamos el divisor de tensión de la entrada positiva del amplificador operacional para hallar las diferentes relaciones entre resistencias.

$$\frac{V_{ref} - V}{R_2} = \frac{V}{R_1}$$

$$(V_{ref} - V)R_1 = VR_2$$

$$V = \frac{V_{ref}R_1}{R_1 + R_2}$$

Fijando R_2 a 10KΩ, vamos rellenando la tabla con los diferentes valores según los Lux:

Lux	$V_o = LV + V$	$V = V_{ref} R_1 / R_1 + R_2$	R_1
100	5=101*V	V=0.05	100
200	5=201*V	V=0.025	50
500	5=501*V	V=0.01	20
1000	5=1001*V	V=0.005	10

Vamos obteniendo los diferentes valores de 'R1':

$$0.05 = \frac{5R_1}{R_1 + 10000}$$

$$0.05R_1 + 500 = 5R_1$$

$$R_1 = 100\Omega$$

$$0.025 = \frac{5R_1}{R_1 + 10000}$$

$$R_1 = 50\Omega$$

$$0.01 = \frac{5R_1}{R_1 + 10000}$$

$$R_1 = 20\Omega$$

$$0.005 = \frac{5R_1}{R_1 + 10000}$$

$$R_1 = 10\Omega$$

Vemos que la resistencia varía entre unos 10 y 100 ohmios, asique pondremos un potenciómetro de 100Ω.

Valores finales:

$R_1 =$ Potenciómetro de 100Ω

$R_2 = 10k\Omega$

$R_3 = 450k\Omega$

- Tramo 2:

En el segundo tramo tendremos una variación de 1000:100000 Lux, rango para luz solar. Seguimos el mismo razonamiento que para el tramo 1; queremos que la tensión de salida sea de 5 voltios cuando incido el pleno sol, pero hay diversos tipos de pleno sol que dan lugar a una amplia gama de Lux, luego asique pondremos un potenciómetro como en el tramo 1 dependiendo de si el día está muy claro o nuboso. La tensión $'V_o'$ debe estar entre 0 y 5 voltios ya que es el rango de lectura analógica que posee el Arduino. Fijamos $'R_3'$ a 10KΩ como ya explicamos anteriormente.

Fijamos $R_2 = 10K\Omega$ como en el anterior tramo, y rellenamos la tabla con diferentes valores de Lux máximos.

Lux	$V_o = \sqrt{L}V + V$	$V = V_{ref}R_1/R_1 + R_2$	R_1
1000	5=32.6*V	0.15	300
5000	5=71.71*V	0.07	140
10000	5=101*V	0.05	100
20000	5=142*V	0.035	70
50000	5=224.6*V	0.022	44
100000	5=317*V	0.015	30

$$0.15 = \frac{5R_1}{R_1 + 10000}$$

$$0.15R_1 + 1500 = 5R_1$$

$$R_1 = 300\Omega$$

$$0.07R_1 + 700 = 5R_1$$

$$R_1 = 140\Omega$$

$$0.05R_1 + 500 = 5R_1$$

$$R_1 = 100\Omega$$

$$0.035R_1 + 350 = 5R_1$$

$$R_1 = 70\Omega$$

$$0.022R_1 + 220 = 5R_1$$

$$R_1 = 44\Omega$$

$$0.015R_1 + 150 = 5R_1$$

$$R_1 = 30\Omega$$

Los valores de 'R1' en este caso varía entre 30 y 300 ohmios, asique nos quedamos con un potenciómetro de 470Ω.

Los valores finales para este tramo son los siguientes:

$R_1 =$ Potenciómetro de 470Ω

$R_2 = 10k\Omega$

$R_3 = 10k\Omega$

Para fijar la tensión de salida a los 5 voltios aproximadamente, antes mencionados, a nuestra salida del linealizador que hemos modelado, debemos colocar un regulador *zener*, con el cual nunca vamos a sobrepasar los 5.3 voltios que puedan dañar nuestro Arduino, su funcionamiento es muy sencillo.

Si a un diodo *zener* se le aplica una corriente eléctrica del ánodo al cátodo (polarización directa) toma las características de un diodo rectificador básico, pero si se le suministra corriente eléctrica de cátodo a ánodo (polarización inversa), el diodo solo dejará pasar un voltaje constante, en nuestro caso 5.1 voltios, que es el diodo colocado.

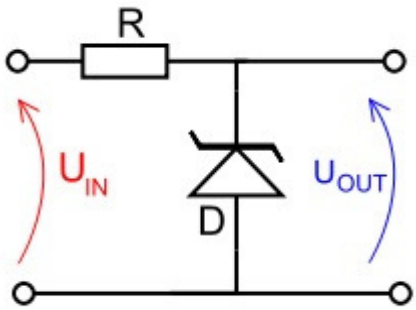


Fig.5.8.Esquema circuito regulador de tensión

La tensión de entrada nos llega a unos 12 voltios con muchos Lux, y la intensidad que pasa por nuestra *zener* cuando empieza a conducir en directa viene en la hoja de especificaciones en los anexos.

En nuestro caso nuestra resistencia '*R*' deberá de ser de un valor aproximado, el cual nos sale al resolver la malla:

$$R = \frac{U_{in} - U_{out}}{I_z} = \frac{12 - 5.1}{0.049} = 140$$

Colocamos una resistencia de valor normalizado de 220Ω. El punto de tensión entre la resistencia y el cátodo del zener, es el punto de tensión que leerá el Arduino con sus pines de entrada analógicos.

Mostramos una foto de la placa de nuestro circuito impreso de nuestra circuitería explicada anteriormente.

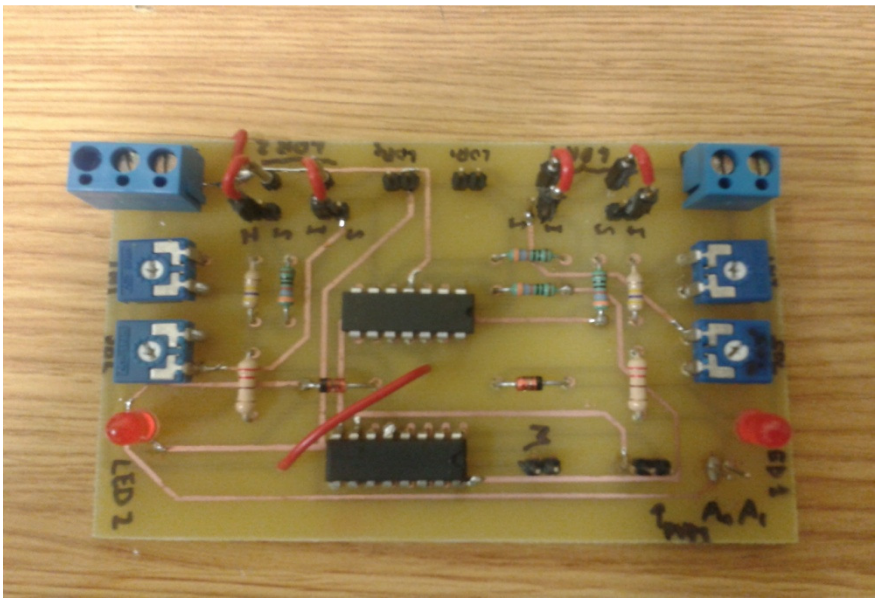


Fig.5.5. Placa circuito impreso

5.1.3.- Actuador (motor DC).

El motor de corriente continua es una máquina que convierte la energía eléctrica continua en mecánica, provocando un movimiento rotatorio. En la actualidad existen nuevas aplicaciones con motores eléctricos que no producen movimiento rotatorio, sino que con algunas modificaciones, ejercen tracción sobre un riel. Se conocen como motores lineales.

Esta máquina de corriente continua es una de las más versátiles en la industria. Su fácil control de posición, par, velocidad la han convertido en una de las mejores opciones en aplicaciones de control y automatización de procesos.

La principal característica del motor de corriente continua es la posibilidad de regular la velocidad desde vacío a plena carga.

El sentido de giro de un motor de corriente continua depende del sentido relativo de las corrientes circulantes por los devanados inductor e inducido.

La inversión del sentido de giro del motor de corriente continua se consigue invirtiendo el sentido del campo magnético o de la corriente del inducido.

5.2.- Código y programación de Arduino.

En este apartado pasamos a explicar el código de programación para Arduino empleado, será fácil comprenderlo después de la explicación teórica explicada en apartados anteriores.

El programa empieza con la declaración de las variables, que usaremos a lo largo del programa, y asignación de pines de salida, que serán los pines '2', '3' y '9'.

Posteriormente, entramos en la función *void 'setup()'*, dónde inicializamos el puerto serie a 9.600 baudios; las siguientes tres líneas inicializamos los pines de salida nombrados antes en la declaración de variables que serán las señales que llegan al driver L293D, las señales digitales que marcan el sentido del motor y la señal *PWM* que le llega al circuito integrado.

Por último, entramos en la función cíclica *void 'loop()'* dónde leemos la señal analógica de uno de los sensores en el pin '*A0'* de entrada y luego la mapeamos de 10 bits a 7 bits para poder transmitir el dato a través del puerto serie, lo mismo hacemos con el otro sensor leído en el pin '*A1'* de lectura de entrada de datos. Realizamos la diferencia entre los dos pines que será la señal de error a controlar en *Simulink*.

El siguiente paso el escribir el dato en el puerto serie, al cual le preguntamos si está disponible, si está disponible, pasamos a procesar el dato en *Simulink* y leemos el dato procesado en la variable '*valorPID'*.

El dato que recibimos a través del puerto tendrá valores entre 0 y 255 y nosotros queremos una señal igual pero con valores entre -127 y 127 para saber el sentido que debe tener nuestro motor indicado por el signo del dato, por lo que mapeamos el dato recibido.

Si el dato leído por uno de los pines de entrada analógica es mayor que cero (positivo) el motor deberá moverse en un sentido, habilitando el puente 'H' del driver hacia un sentido con una entrada a 5 voltios y la otra a 0 voltios y escribiendo el valor procesado en el pin *PWM*, hasta hacer la diferencia error lo más pequeña posible llegando a cero. El resto de código restante es idéntico a lo explicado en este párrafo, con la diferencia de que cambiamos los valores de las salidas digitales del Arduino (de 5 a 0 y de 0 a 5, respectivamente) que leerá el driver para invertir el sentido de la corriente hacia el motor si el dato leído es menor que cero (negativo). No perdemos potencia en la señal que recibe el motor al mapear, debido que realizamos el valor absoluto del dato y le sumamos 127, por lo que en el 'pin' *PWM* le llegan valores entre 0 y 255. Tenemos dos estamentos '*if*' que nos indica que si el error es insignificante, que no nos llegue tensión al motor, escribiendo un cero en el pin '*PWM*'.

En el primer *'if'* del código controlamos el micro-interruptor instalado, cuando está por debajo de un rango bajo la lectura de tensión de las fotorresistencias, le metemos tensión en el pin '9' mientras el interruptor está abierto (LOW), es decir, sin estar pulsado.

Veamos el código implementado:

```
//Declaración de variables

int Ldr1=0;
int Ldr2=0;
int error=0;
int valorPID=0;
int valorPIDabs=0;
int entradamotor1=2;
int entradamotor2=3;
int micro1=7;
int pwm=9;

void setup(){
Serial.begin(9600);
pinMode(entradamotor1,OUTPUT);
pinMode(entradamotor2,OUTPUT);
pinMode(pwm,OUTPUT);
pinMode(micro1,INPUT);
}

void loop(){
Ldr1=analogRead(A0);
Ldr1=map(Ldr1, 0, 1023, 0, 127);
Ldr2=analogRead(A1);
Ldr2=map(Ldr2, 0, 1023, 0, 127);
error=Ldr1-Ldr2;
```

```
if(Ldr1<10 && Ldr2<10){
while(digitalRead(micro1)==LOW){
digitalWrite(entradamotor1,HIGH);
digitalWrite(entradamotor2,LOW);
analogWrite(pwm,200);
}
}

Serial.write(error);
delay(20);
if(Serial.available()>0){
valorPID=Serial.read();

valorPID=map(valorPID, 0, 255, -127, 127);
valorPIDabs=abs(valorPID);
valorPIDabs=valorPIDabs+127;
}

if(valorPID>0){
digitalWrite(entradamotor1,HIGH);
digitalWrite(entradamotor2,LOW);
if(error<2){
analogWrite(pwm,0);
}
else{
analogWrite(pwm,valorPIDabs);
}
}

if(valorPID<0){
digitalWrite(entradamotor1,LOW);
digitalWrite(entradamotor2,HIGH);
if(error>-2){
analogWrite(pwm,0);
}
else{
analogWrite(pwm,valorPIDabs);
}
}
}
```

Este sería el código necesario para tener programado correctamente nuestro micro-controlador y así procesar los datos con *Simulink*.

5.3.- Modelo del lazo control implementado en *Simulink*.

En este apartado explicaremos el diagrama de bloques realizado en *Simulink* donde se envían los datos leídos en Arduino para que sean procesados y en el controlador.

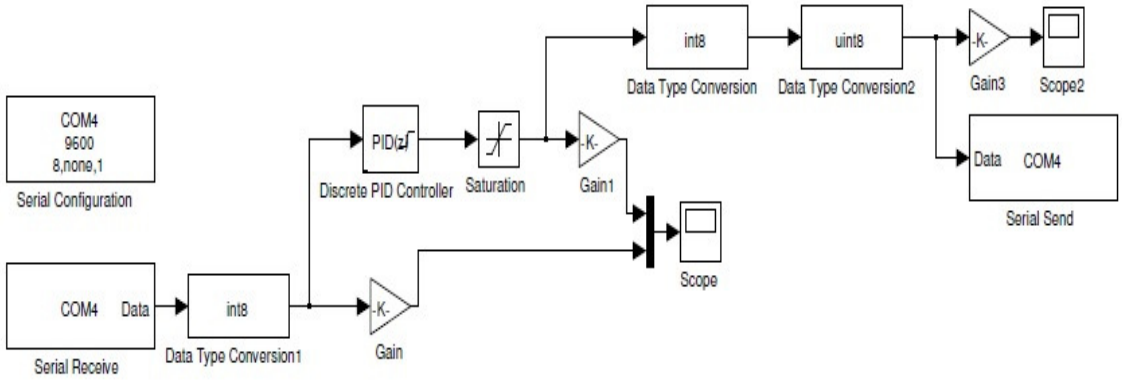


Fig.5.9.Lazo de control en Simulink

El diagrama de bloques comienza con la lectura de datos que se reciben del Arduino, que será realizado por el bloque '*Serial Receive*', y convertimos el dato en un entero de 8 bits. Posteriormente, pasamos el dato al bloque '*PID*' para realizar el algoritmo y lo visualizamos en un '*Scope'* con la respectiva resolución para poder visualizarlo en voltios reales, en este caso:

$$\frac{Rango}{2^{nºbits}} = \frac{5}{2^7} = \frac{5}{127}$$

El error producido entre ambos sensores es enviado al bloque '*PID*', que creará la señal de control correspondiente para corregir el error mencionado. A la salida del bloque hemos colocado un bloque de saturación entre -127 y 127 para que no sobrepase estos valores, visualizamos el valor como antes hemos dicho y convertimos el valor entero de 8 bits en un valor entero de 8 bits también pero sin signo para que así pueda transmitir los datos a través del puerto serie el bloque '*Serial Send*', siendo el dato entre 0 y 255. Realizamos otra visualización del dato enviado al Arduino (5/255).

- Sintonización del controlador '*PID'.*

En cuanto a la sintonización de nuestro bloque '*PID'* se ha escogido un controlador con las tres acciones, proporcional, integral y derivativa.

Para sintonizar nuestro '*PID'* que se ajuste de la mejor manera a nuestro sistema, hemos utilizado el método de prueba y error logrando una respuesta lo más rápida posible y conseguir que el error sea lo más próximo a cero y sin sobrepicos. Después de varias pruebas los valores de las acciones proporcional, integral y derivativa deben ser 1.2, 0.05 y 0.01 respectivamente.

RESULTADOS Y CONCLUSIONES

6.1.- Resultados.

Al realizar varias pruebas sobre el sistema obtuvimos los siguientes resultados, los mostrados en las figuras de este capítulo. En nuestra primera figura, en color lila mostramos la señal error producida entre los dos sensores y la señal amarilla es la producida por el regulador '*PID'.*

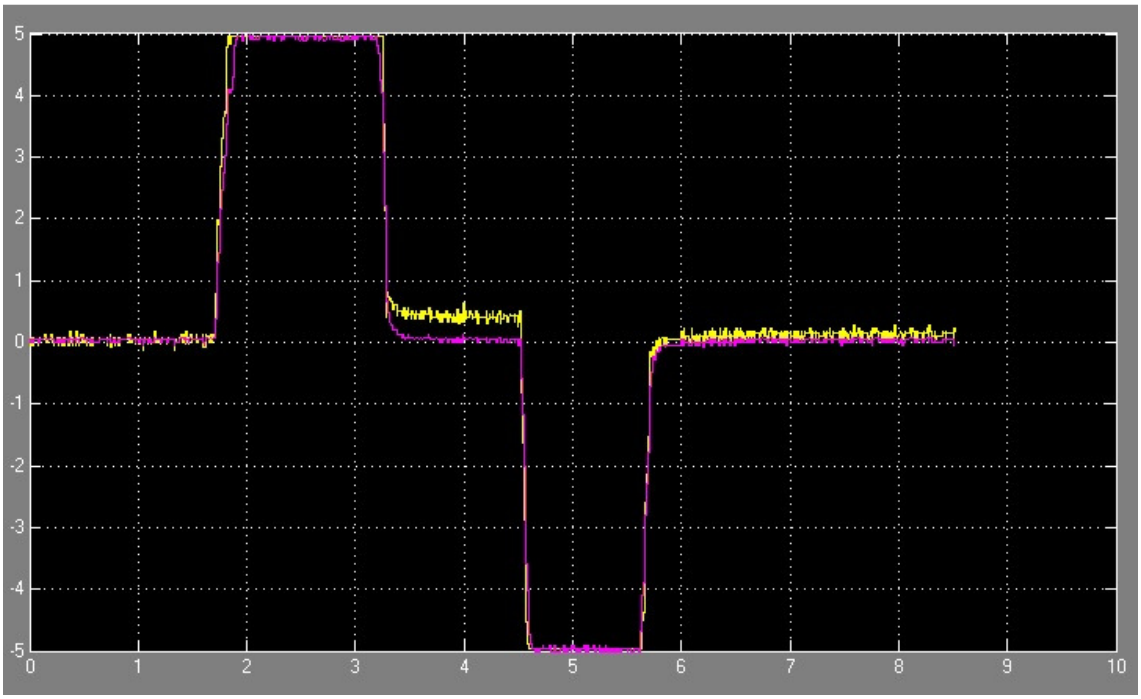


Fig.6.1.Respuesta del sistema (1)

En esta primera figura lo que hemos realizado es dejar los dos sensores ante una misma intensidad viendo como el error es cero, posteriormente, hemos tapado uno de los sensores (sensor A) y el otro se sometió a una intensidad de luz (sensor B) de manera rápida, viendo como asciende el error rápidamente a 5 voltios y se mantiene en ese valor, volvemos a poner los sensores ante la misma intensidad y la señal error vuelve a cero. Vemos perfectamente como la señal producida por nuestro controlador es capaz de seguir nuestra señal error y así transmitírselo a nuestro actuador, en nuestro caso un motor de continua, sin retardo ni sobrepicos. Por último, realizamos lo mismo pero al revés, sometemos a oscuridad sensor B y a intensidad alta el sensor A, observamos el mismo resultado alcanzando los -5 voltios, mantenemos, las dejamos ante la misma intensidad de luz y rápidamente vuelve a cero la señal error y la señal generada por nuestro controlador la sigue perfectamente sin retardo ni sobrepicos.

En la siguiente figura seguimos el mismo patrón de colores, color lila para la señal error producida entre los dos sensores y la señal amarilla producida por el controlador *'PID'*.

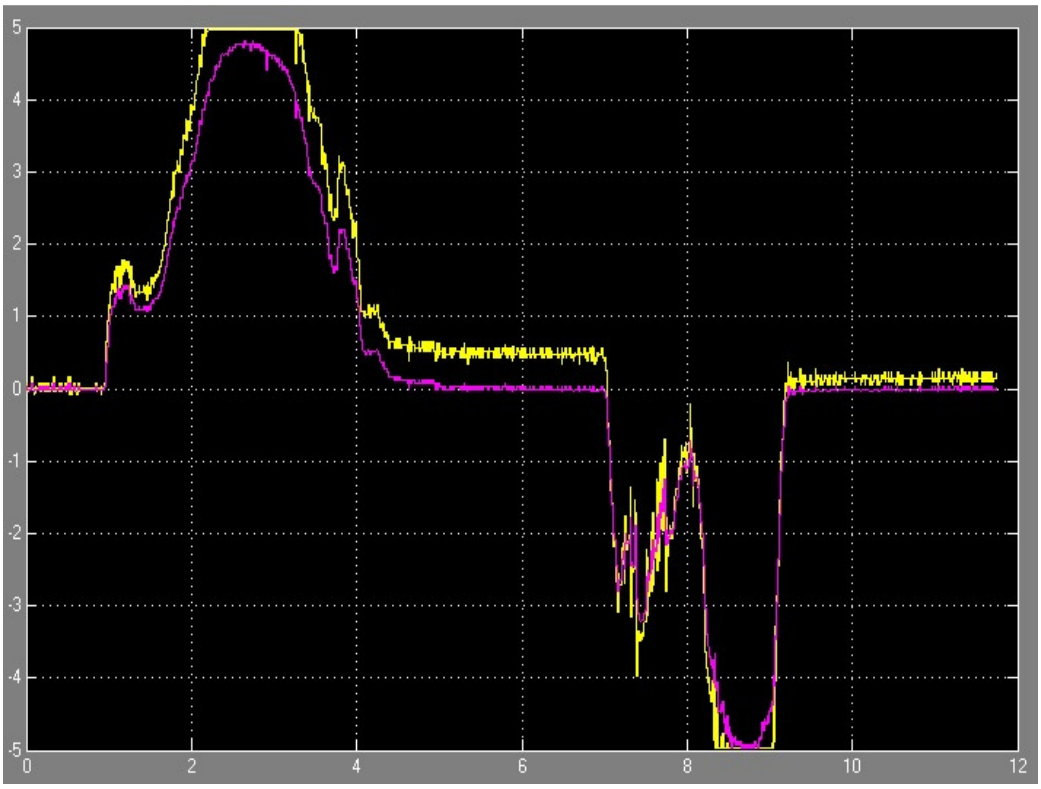


Fig.6.2.Respuesta del sistema (2)

En esta imagen ya no sometemos a los sensores a diferencias bruscas (sensor A plena oscuridad y sensor B a alta intensidad), si no que el seguidor irá siguiendo una intensidad luminosa de una manera progresiva.

Vemos en color lila la señal error entre los dos sensores como aumenta progresivamente, la señal de control en color amarillo generada la sigue perfectamente, siendo la tensión que le llega al motor, luego va disminuyendo poco a poco hasta llegar a cero donde el error es cero y los sensores se mantienen a la misma intensidad de luz. Por último, le damos más luz al otro sensor, aumenta el error entre ellos y la señal de control salta rápidamente para seguir al error, luego disminuye rápidamente el error y la señal de control igualmente, llegando a cero, rápidamente, que es nuestro objetivo principal.

6.2.- Conclusiones.

Las conclusiones sacadas son buenas y satisfactorias, ya que hemos logrado crear un sistema que tiene buen comportamiento y se cumple con los objetivos planteados inicialmente. El sistema funciona bien y es capaz de seguir cualquier rango de luz y variar la velocidad del motor de continua para así seguir de manera satisfactoria el error entre los dos sensores.

Se han utilizado componentes de bajo coste unidos a un bajo consumo de energía para controlar el dispositivo.

Personalmente, las conclusiones son positivas, ya que hemos puesto en práctica buena parte de los conceptos y conocimientos que hemos ido adquiriendo durante nuestra formación en la ingeniería y hemos sido capaces de solventar los problemas que nos han surgido de una manera real.

Líneas futuras:

Incorporar una batería y un sistema de carga de ésta a través del módulo fotovoltaico.

Comprobar otros algoritmos de seguimiento.

Comparar experimentalmente las prestaciones de este seguidor con una estación solar fija.

Diseñar un sistema de transmisión que conecte el seguidor diseñado con un teléfono móvil.

ANEXOS

Circuito impreso: definición y ventajas.

• DEFINICIÓN:

– CIRCUITO IMPRESO: Soporte de material **aislante** (rígido o flexible) sobre el que se encuentran adheridas **pistas conductoras** que conectan entre sí puntos de un circuito eléctrico. Normalmente el circuito impreso sirve de **soporte** físico para la colocación y **soldadura** de los componentes (placa de circuito impreso).

– Los circuitos electrónicos están formados por **componentes activos** y **pasivos** unidos entre sí por conductores. También forman parte del circuito elementos que no tienen propiedades eléctricas como son los **conectores**, **bornas**, **terminales**, etc.

• VENTAJAS :

– Permite la producción automática.
– Reduce la mano de obra y por tanto el coste.
– Permite un montaje más compacto, reduciendo considerablemente el espacio y el peso.
– Su exacta reproductibilidad evita errores de cableado.
– Reduce los efectos parásitos inductivos o capacitivos del cableado.
– Es fácilmente verificable.
– Permite una sustitución rápida de componentes.

Proceso de fabricación en el laboratorio.

• Placas positivas.

• Insolado

–Colocar la placa, con el fotolito perfectamente alineado, en la insoladora, debajo del cristal, el tiempo que indique el fabricante de la placa (normalmente 4 o 5 minutos).

• Revelado de la placa.

–Introducir la placa en una cubeta con 75 ml de revelador positivo.

• Lavado con agua.

• Atacado de la placa.

–Preparar en una cubeta el atacador, mezclando:

•25 ml de agua.
•25 ml de ácido clorhídrico 35%.
•25 ml de agua oxigenada (Hidrogeno peróxido 110 vol).

• Lavado con agua.

• Limpieza de la placa con alcohol para eliminar la fotorresina.

Hoja de especificaciones diodo zener

Features

- Silicon Planar Power Zener Diodes
- For use in stabilizing and clipping circuits with high power rating.

- Standard Zener voltage tolerance suffix "A" for ± 5 % tolerance. Other Zener voltages and tolerances are available upon request.

Applications

Voltage stabilization

Mechanical Data

Case: DO-41 Glass Case
Weight: approx. 350 mg
Packaging Codes/Options:
TR / 5k per 13 " reel , 25k/box
TAP / 5k per Ammo mag. (52 mm tape), 25k/box

Absolute Maximum Ratings

T_{amb} = 25 °C, unless otherwise specified

Parameter	Test condition	Symbol	Value	Unit
Power dissipation	$T_{amb} \leq 50$ °C	P_{Diss}	1	W
Z-current		I_Z	P_V/V_Z	mA
Junction temperature		T_j	200	°C
Storage temperature range		T_{stg}	- 65 to + 200	°C
Junction ambient	l = 9.5 mm (3/8 "), T_L = constant	R_{thJA}	100	K/W

Electrical Characteristics

T_{amb} = 25 °C, unless otherwise specified

Parameter	Test condition	Symbol	Min	Typ.	Max	Unit
Forward voltage	I_F = 200 mA	V_F			1.2	V

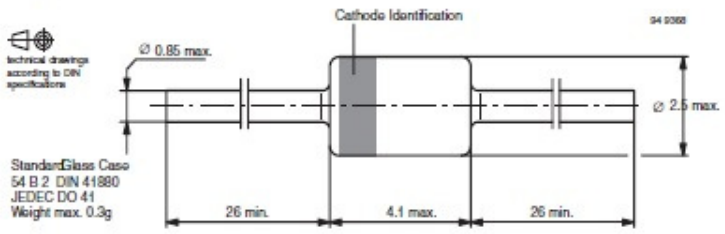
Electrical Characteristics

1N4728A...1N4764A

Partnumber	Nominal Zener Voltage[1]	Test Current	Maximum Dynamic Impedance			Maximum Reverse Leakage Current		Surge curent	Maximum Regulator Current[2]
	V_Z @ I_{ZT}	I_{ZT}	Z_{ZT} @ I_{ZT}	Z_{ZK} @ I_{ZK}	I_{ZK}	I_R	Test Voltage V_R	I_R @ T_{amb} = 25 °C	I_{ZM} @ T_{amb} = 50 °C
	V	mA	Ω	Ω	mA	µA	V	mA	mA
1N4728A	3.3	76	10	400	1	100	1	1380	276
1N4729A	3.6	69	10	400	1	100	1	1260	252
1N4730A	3.9	64	9	400	1	50	1	1190	234
1N4731A	4.3	58	9	400	1	10	1	1070	217
1N4732A	4.7	53	8	500	1	10	1	970	193
1N4733A	5.1	49	7	550	1	10	1	890	179
1N4734A	5.6	45	5	600	1	10	2	810	162
1N4735A	6.2	41	2	700	1	10	3	730	146
1N4736A	6.8	37	0.5	700	1	10	4	660	133
1N4737A	7.5	34	0	700	0.5	10	5	605	121
1N4738A	8.2	31	0.5	700	0.5	10	6	550	110
1N4739A *	9.1	28	0	700	0.5	10	7	500	100
1N4740A *	10	25	7	700	0.25	10	7.6	454	91
1N4741A *	11	23	8	700	0.25	5	8.4	414	83
1N4742A *	12	21	9	700	0.25	5	9.1	380	76
1N4743A *	13	19	10	100	0.25	5	9.9	344	69
1N4744A *	15	17	14	700	0.25	5	11.4	304	61
1N4745A *	16	15.5	16	700	0.25	5	12.2	285	57
1N4746A *	18	14	20	750	0.25	5	13.7	250	50
1N4747A *	20	12.5	22	750	0.25	5	15.2	225	45
1N4748A *	22	11.5	23	750	0.25	5	16.7	205	41
1N4749A *	24	10.5	25	750	0.25	5	18.2	190	38
1N4750A *	27	9.5	35	750	0.25	5	20.6	170	34
1N4751A *	30	8.5	40	1000	0.25	5	22.8	150	30
1N4752A *	33	7.5	45	1000	0.25	5	25.1	135	27
1N4753A *	36	7	50	1000	0.25	5	27.4	125	25
1N4754A *	39	6.5	60	1000	0.25	5	29.7	115	23
1N4755A *	43	6	70	1500	0.25	5	32.7	110	22
1N4756A *	47	5.5	80	1500	0.25	5	35.8	95	19
1N4757A *	51	5	95	1500	0.25	5	38.8	90	18
1N4758A *	56	4.5	110	2000	0.25	5	42.6	80	16
1N4759A *	62	4	125	2000	0.25	5	47.1	70	14
1N4760A *	68	3.7	150	2000	0.25	5	51.7	65	13
1N4761A *	75	3.3	175	2000	0.25	5	56	60	12
1N4762A *	82	3.0	200	3000	0.25	5	62.2	55	11
1N4763A *	91	2.8	250	3000	0.25	5	69.2	50	10
1N4764A *	100	2.5	350	3000	0.25	5	76.0	45	9

[1] Based on dc measurement at thermal equilibrium while maintaining the lead temperature (T_L) at 30 °C + 1 °C, 9.5 mm (3/8") from the diode

Package Dimensions in mm



REFERENCIAS

http://arduino.cc/es/Reference/HomePage

http://wechoosethemoon.es/2011/07/21/arduino-matlab-simulink-controlador-pid/

http://www.sc.ehu.es/sbweb/fisica/solido/din_rotacion/inercia/inercia.htm

http://es.wikipedia.org/wiki/Funci%C3%B3n_de_transferencia

http://robots-argentina.com.ar/MotorCC_L293D.htm

http://es.wikipedia.org/wiki/Amplificador_operacional

Apuntes de la asignatura de Sistemas Electrónicos de Control de 3ºI.T.T. Sistemas Electrónicos de Universidad de Valladolid.