Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Ingeniería electrónica

Control Digital

Informe de Proyecto Final

“Aplicación Control digital en LA de velocidad angular PID de un motor DC en microcontrolador: Seguidor Solar de un eje”

Equipo No. 3

Guzmán Gudiño Said Raul

Cruz Bautista Dante Danilo

Muñiz Hernández Oscar Javier -358851

Grupo 363

Docente: Dr. Miguel Angel Murillo Escobar

Ensenada B.C. a 08 de Junio de 2022

ÍNDICE

INTRODUCCIÓN	**3**
OBJETIVO	**3**
MATERIALES UTILIZADOS	**4**
DESARROLLO	**4**
RESULTADOS	**8**
CONCLUSIÓN	**10**
BIBLIOGRAFÍA	**11**
ANEXOS	**11**

INTRODUCCIÓN

En este informe, se verá el diseño y la implementación de un circuito electrónico digital el cual está basado en el microcontrolador Arduino Uno (Atmega328P) para un sistema seguidor solar de un solo eje para la mejor de eficiencia de paneles solares utilizando un control PID controlando la velocidad angular en un motor de DC, y en base a esto, se hará uso de las LDR para controlar la dirección de giro del motor.

En resumen, se controlará tanto la velocidad angular como la dirección de giro de un motor, el cual a su vez controla la posición de una base en la cual existen dos LDR, y dependiendo de los valores de resistencia, el motor girará hacia un lado u otro, todo esto con ayuda de una base de un solo eje.

Un seguidor de un solo eje es cuando la rotación de la superficie de captación se hace sobre un solo eje, ya sea horizontal, vertical u oblicuo y cuenta con un costo menor en comparación a otros tipos de seguidores debido a que requiere de menos materiales para fabricarlo.

Para la programación de este seguidor se implementará por luminosidad. Los seguidores con este tipo basan su funcionamiento en la señal transmitida por uno o varios sensores, los cuales envían señales de control a distintos periféricos del sistema para llevar a cabo el posicionamiento en el punto más adecuado de luminosidad.

OBJETIVO

- Utilizar el controlador de velocidad o de posición angular para emplearlo en un uso práctico.

MATERIALES UTILIZADOS

- Microcontrolador Arduino Uno
- 1 Motorreductor 12 Vdc
- 1 Potenciometro 5K ohms
- 2 Fotoresistencias
- 2 Resistencias 10K ohms
- 1 Protoboard
- Driver TB6612FNG para motor DC

DESARROLLO

Para realizar este proyecto fue necesario haber realizado la práctica No.5 y No.7 de esta materia, ya que en ella se realiza un controlador digital de control de velocidad angular y la dirección del giro, tomando como referencia esas prácticas, se parte del circuito de la figura 1.

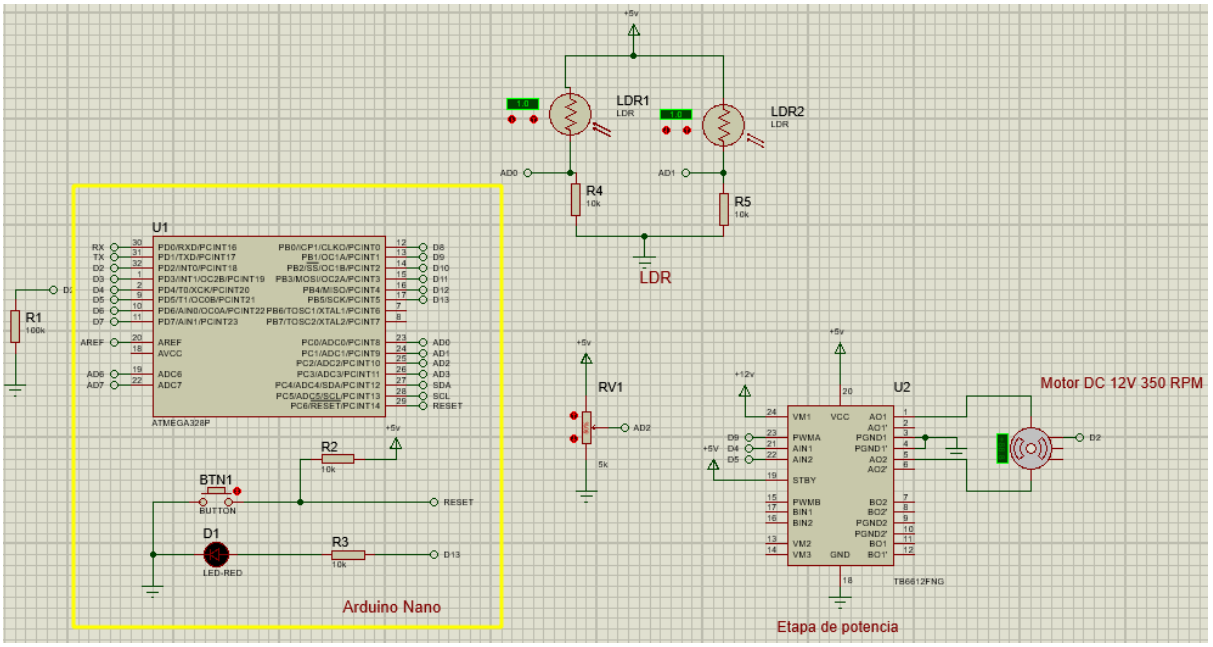

Figura 1. Circuito eléctrico digital para el control de velocidad angular y giro con motor DC.

Para realizar un seguidor de luz de un eje empleamos dos LDR (Light Dependent Resistor) o fotorresistencias las cuales nos permiten medir la incidencia de luz.

La tarea de este dispositivo es encontrar un punto de luz dentro de un ángulo de detección de las fotorresistencias y dirigirse lo más rápido posible hacia la fuente.

El funcionamiento del seguidor tiene una base simple, tenemos dos fotorresistencias

en los extremos de nuestra base, estas fotorresistencias miden el valor analógico que disminuye mientras más luz incida en las LDR. Al comparar estos valores podemos determinar de qué parte de la base incide mayor luz y con ello usar el motor DC junto con el controlador TB6612FNG para rotar la base hacia la fuente de luz incidente para que los dos sensores capten el mismo nivel de luz.
De esta manera logramos tener la mayor eficiencia en sistemas de generación de energía con paneles solares ya que tendremos la mayor incidencia de luz posible dentro de un eje de rotación.

Mientras tanto la velocidad de respuesta la definimos por una señal de modulación por ancho de pulso (PWM) que va de valores de 0 a 255, y que podemos modificar con un potenciómetro, una consideración importante es que si esta señal de control es muy pequeña el motor no puede girar completamente por la carga que está llevando, mientras que si la señal de control es muy grande el sistema responde con mucha velocidad dando lugar a movimientos bruscos y erráticos que podrían dañar el seguidor solar.

Construcción:

Partiendo de la simulación y del código realizado armamos el circuito en un protoboard para poder verificar su funcionamiento y realizar los ajustes necesarios.

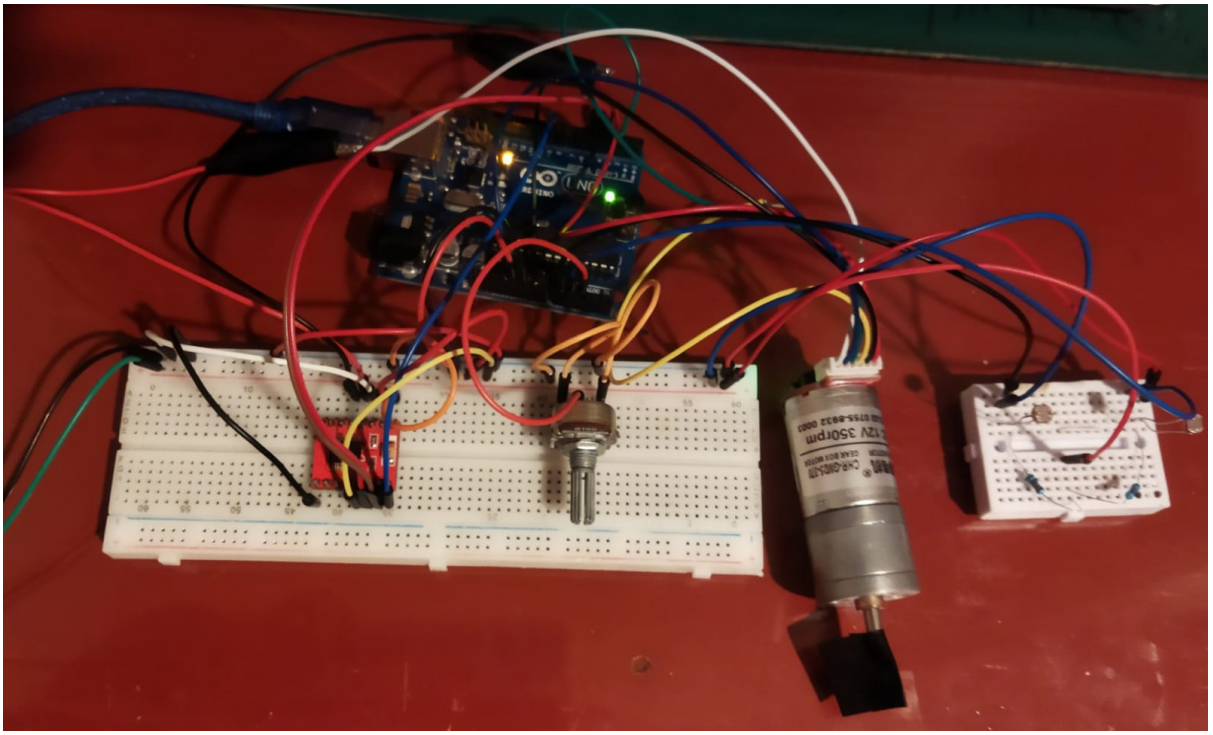

Figura 2. Circuito eléctrico fisico para el control de velocidad angular y giro con motor DC

Después de corregir errores tales como el rango de error de la diferencia entre los valores de las LDR y el giro erróneo del motor se pudo comprobar el buen funcionamiento de nuestro sistema.

Se construyó una base hecha de madera triplay para poder montar en ella nuestro microcontrolador, protoboard y en ella los componentes, además de posicionar nuestro motor junto con su base con los LDR.

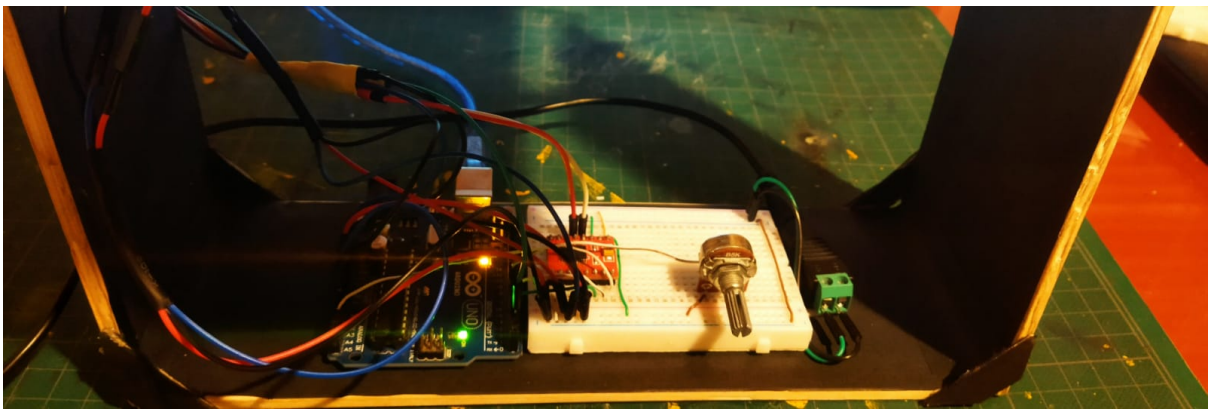

Figura 3. Circuito armado en base

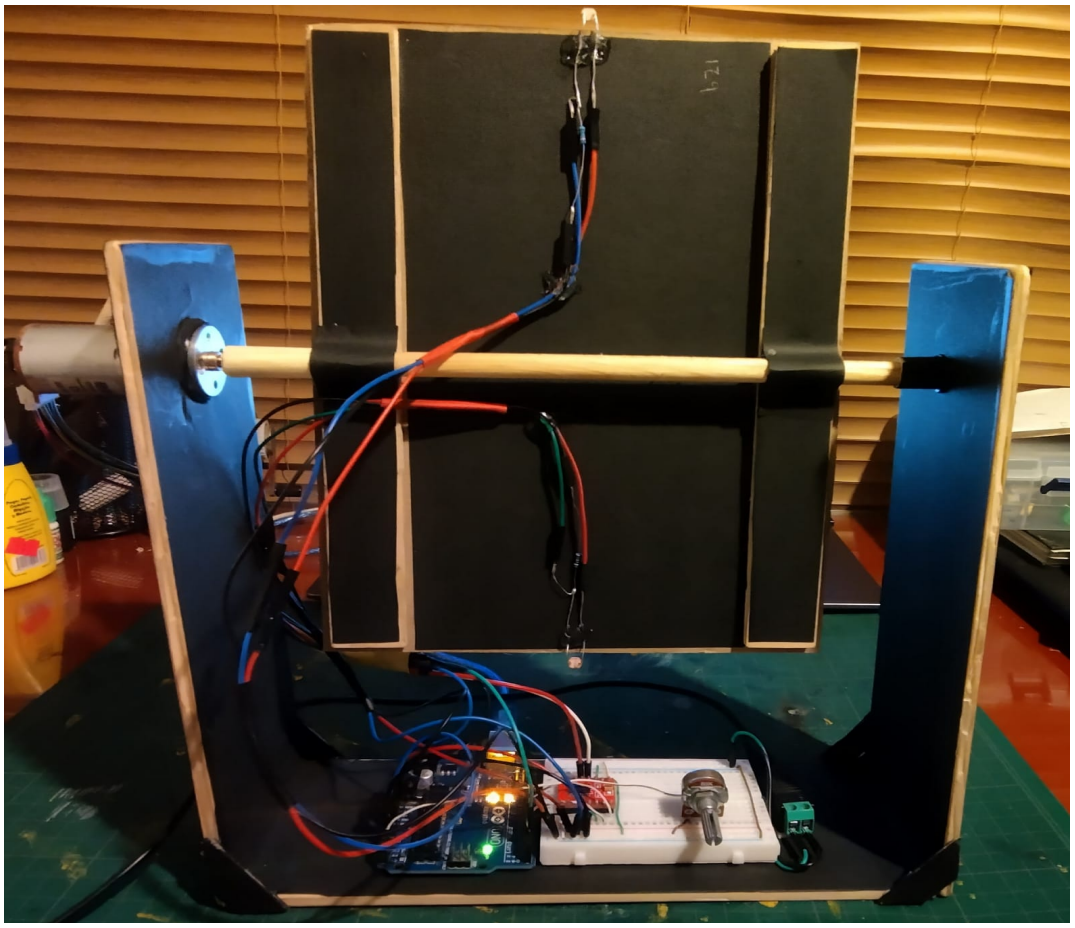

Figura 4. Circuito completo de seguidor solar de un eje

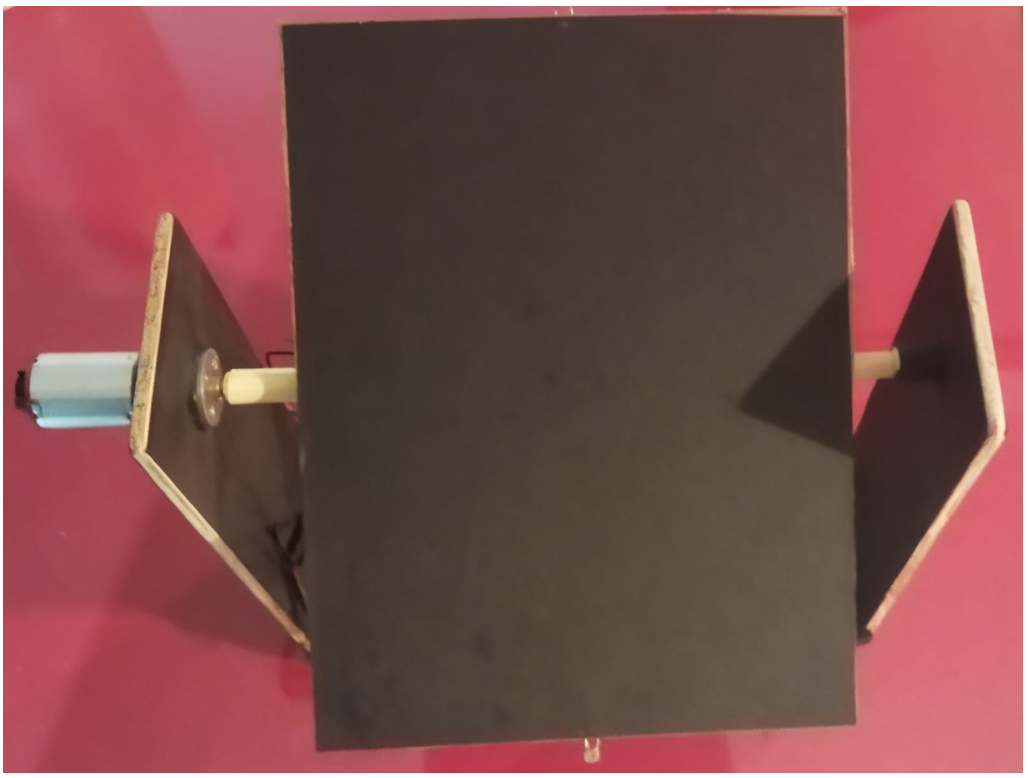

Figura 5. Base de Seguidor solar

Después de conectar todos nuestros componentes y volver a verificar el buen funcionamiento mejoramos la estética del sistema realizando un manejo de cables.

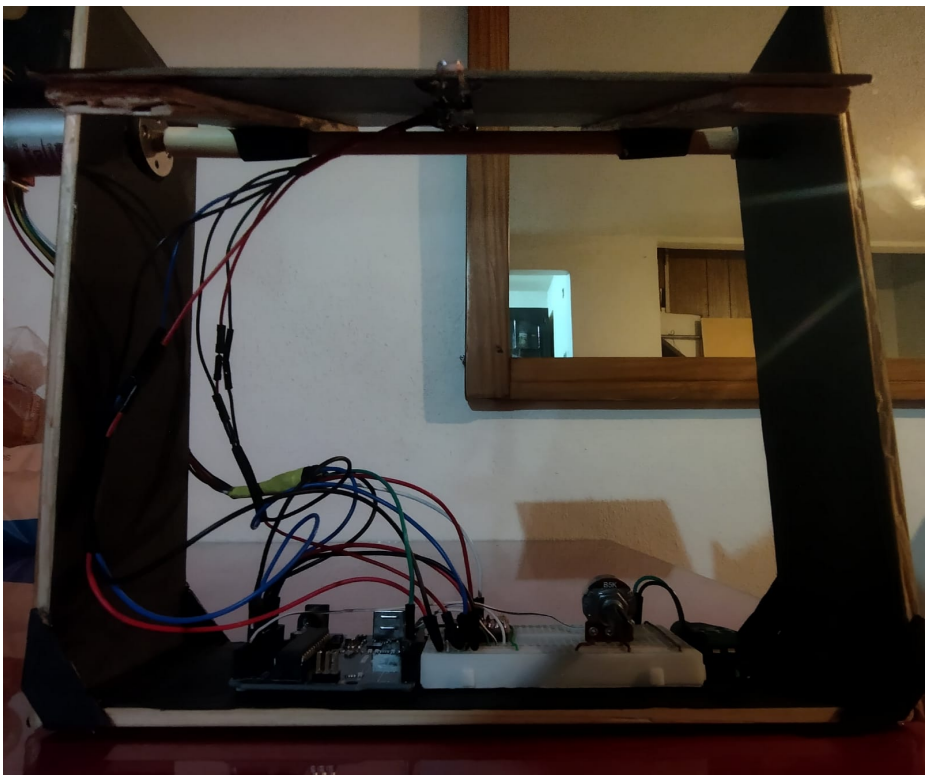

Figura 6. Seguidor solar de un eje

RESULTADOS

El seguidor solar funciona de manera óptima después de varios ajustes en el código y en la base del proyecto.
Sin embargo hay que tener en cuenta diversas consideraciones y cambios que tuvimos que realizar para poder hacer funcionar el seguidor solar de un solo eje.
El primer cambio que tuvimos que hacer fue que nuestro seguidor solar es de un solo eje y no de dos ejes, esto se debe a que solo contamos con un solo motor para este proyecto, lo que si bien es suficiente para aprovechar de mejor manera la luz solar no es la mejor opción posible ya que un seguidor de dos ejes permite tener un mejor control del panel solar lo que nos da la mayor eficiencia posible de un sistema fotovoltaico.

Otro punto importante a considerar en el funcionamiento es que la señal PWM que decidimos a través del potenciómetro debe mantenerse a menos de 40, ya que si se superan el sistema responde de manera muy rápida, situación que mayormente da a lugar a la desestabilización y funcionamiento errático. Por otro lado si la señal de modulación por ancho de pulso bajan de 20 el motor no responde de manera correcta ya que por la carga que tiene el motor tiene que realizar una mayor fuerza de torque para girar lo que significa que necesita mayores RPM para girar, si no, este se queda estático.

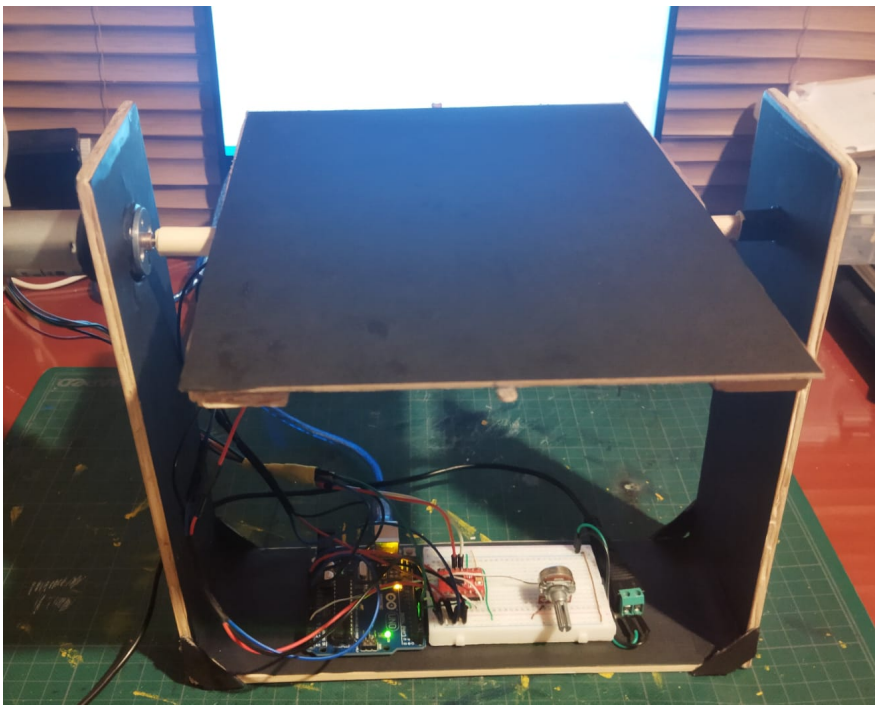

Figura 7. Estabilidad de eje del seguidor solar

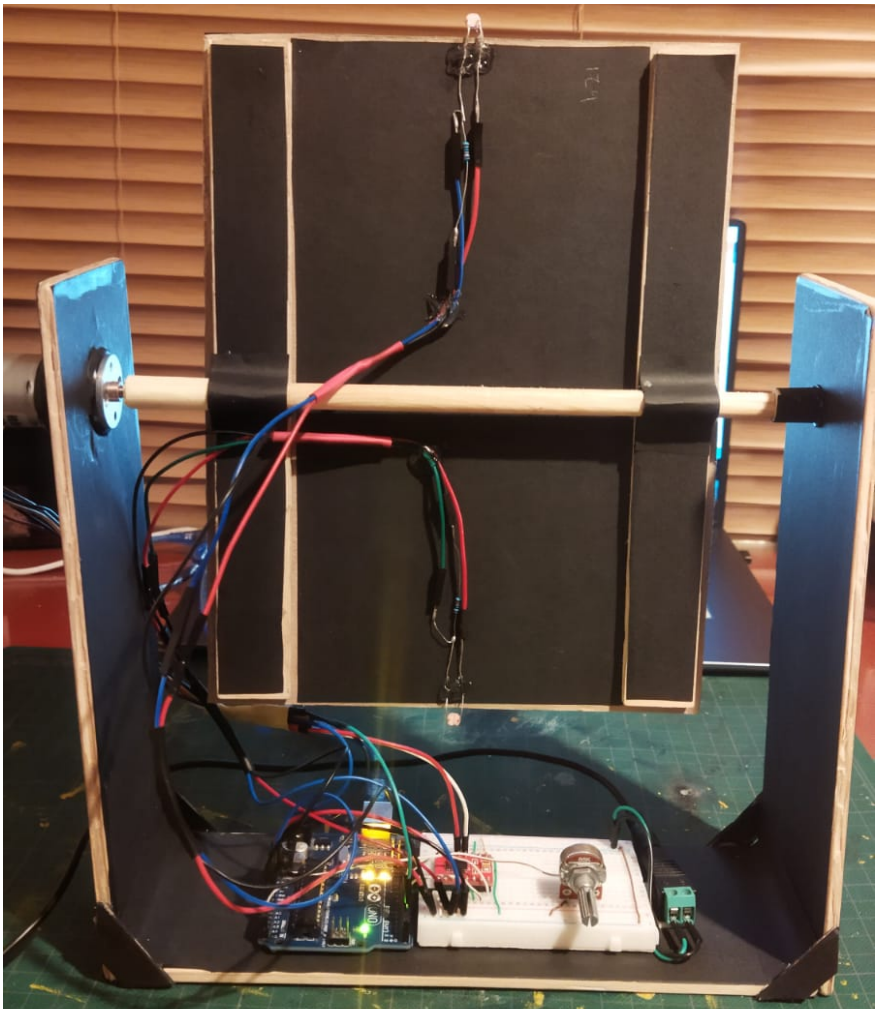

Figura 8. Movimiento del eje de seguidor solar

CONCLUSIÓN

En base a los resultados obtenidos al finalizar el proyecto, se concluye que el funcionamiento del circuito, tanto la parte física, como es el diagrama montado en el protoboard junto con los materiales de apoyo como lo es la base, y la parte digital, así de igual forma el código programado en Arduino para el seguidor de luz y para el ajuste de los sensores, fueron los esperados ya que cumplieron el objetivo principal de este proyecto que fue el darle un uso al controlador de velocidad al darle movimiento a un sistema de seguimiento solar maximizando la producción de electricidad de la instalación fotovoltaica, ya que optimiza el ángulo con el que los paneles reciben la radiación solar.

Para la realización de este proyecto fue necesario tener como base el circuito de la práctica No.5, No.6 y No.7 de esta materia, además de conocimientos de la carrera de electrónica, ya que en este proyecto fue necesario programar y entender cómo funciona cada componente para lograr los resultados finales, así que para la realización de este proyecto fueron necesarios los conocimientos de varias materias de la carrera, principalmente la materia de control digital y electronica analogica.

Algunas consideraciones a tomar en cuenta son:

- La precisión del movimiento de orientación está limitada a la cantidad de luz que recibe y la orientación.
- Ante un entorno sobresaturado de luz el dispositivo puede no reaccionar correctamente.

BIBLIOGRAFÍA

Bajo Sánchez C. (2017) *SEGUIDOR FUENTE DE LUZ. [TRABAJO FIN DE MÁSTER: INGENIERÍA INDUSTRIAL].* Escuela Politécnica Superior, Universidad Carlos III de Madrid.

Boylestad, R., "Fundamentos de Electrónica", Cuarta Edición. Pearson Education, 1997.

ANEXOS

Código:

```
#define LDR1 A0
#define LDR2 A1
#define POT_PWM A2 // POTENCIOMENTO PWM 0-255 DUTY 0%-100%
#define GIRO_AIN1 4 // Salida digital control de giro en puente H
#define GIRO_AIN2 5 // Salida digital control de giro en puente H
#define PWM_OUT 9 // define PWM_OUT al pin 9 para 490 Hz; pin 6 para 980 Hz
#define encoder_pin 2 // define encoder_pin al pin 2 para interrupción externa por encoder

float tol_error = 50;
float left_sensor = 0;
float right_sensor = 0;
int power = 200;

int VAL_POT_PWM = 0; // variable para almacenar valor 10bit lectura del POT_PWM
float VAL_01=0;
float VAL_0350=0;

// PARA TEMPORIZADORES
unsigned long tpo=0;
unsigned long timeold = 0;

// CONTROL DE VELOCIDAD
float RPM = 0; // Revoluciones por minuto calculadas.
byte DUTY=0; // Variable para almacenar valor 8bit lectura del POT_PWM
volatile unsigned int pulses = 0; // Número de pulsos leidos por el Arduino en un segundo
static volatile unsigned long debounce = 0; // Tiempo del rebote.
unsigned int pulsesperturn = 389; // Número de pulsos por revolucion del motor DC 350 RPM 12Vdc teórico

// CONTROL PID LAZO CERRADO
float set_speed=0;
float kp=0.5; // Ganancia proporcional
float ki=1; // Ganancia integrativa
float kd=0.01; // Ganancia derivativa
float integral=0; // Almacena la integral del error
float derivada=0; // Almacena el valor de la derivada

// ERRORES DE CONTROL PID
float e_speed=0;
float e_speed_pre=0;
```

```
float e_speed_sum=0;

// Señal de control PWM
byte pwm_pulse=0; // señal de control para el actuador Motor DC
int pwm_pulse_int=0; // señal de control para calculo de PID.


// Configuracion inicial:
void setup() {
      attachInterrupt(digitalPinToInterrupt(encoder_pin), ext_ISR, RISING); // Habilitar interrupción externa en bajada.


  pinMode(POT_PWM, INPUT); // Define POT_PWM de tarjeta entrada.
  pinMode(LDR1, INPUT); // Define de tarjeta entrada.
  pinMode(LDR2, INPUT); // Define POT_PWM de tarjeta entrada.
  pinMode(PWM_OUT, OUTPUT); // Define PEM de tarjeta salida.
  pinMode(GIRO_AIN1, OUTPUT); // Define BOTON1 de tarjeta entrada ACTIVO ALTO.
  pinMode(GIRO_AIN2, OUTPUT); // Define BOTON1 de tarjeta entrada ACTIVO ALTO.

  Serial.begin(9600); // setup serial

}


// Ciclo infinito
void loop() {
  tpo=millis(); // Actualiza el tiempo en milisegundos continuamente.

  // Seleccionar RPM entre [60,220] con el potenciometro A0
  VAL_POT_PWM = analogRead(POT_PWM); // Leer DUTY 10 bit del POT
  VAL_01 = VAL_POT_PWM/1023.0; // Convertir 10 bits proporcional (0,1)
  VAL_0350 = round(VAL_01*350); // Convertir (0,1) a [0,350] RPM

  if(VAL_0350 < 29) // Minimo 60 RPM
  {
  set_speed = 30;
  }else
  {
  set_speed = VAL_0350;
  }

   left_sensor = analogRead(LDR1);
   right_sensor = analogRead(LDR2);

  if(left_sensor > right_sensor + tol_error){ //sentido derecho?
    digitalWrite(GIRO_AIN1,HIGH); // GIRO MOTOR CW
    digitalWrite(GIRO_AIN2,LOW);
  }
  else if (right_sensor >left_sensor + tol_error){//sentido izquierdo?
    digitalWrite(GIRO_AIN1,LOW); // GIRO MOTOR CCW
    digitalWrite(GIRO_AIN2,HIGH);
  }
  else {
    digitalWrite(GIRO_AIN1,LOW); //no giro
```

```
    digitalWrite(GIRO_AIN2,LOW);
  }
  delay(10);

 // Entra cada 0.2 segundos.
 // Durante los 0.2 segundos cuenta pulsos del encoder.
 if(tpo - timeold >= 200) // Entra cada xxx ms, limitado por el encoder.
 {
 detachInterrupt(0); // Desactiva interrupción para que no actué en esta parte del programa.
 RPM = (60.0 * 1000.0 / pulsesperturn ) / (tpo - timeold)* pulses; // Calcula revoluciones por
minuto RPM

  // Control PI de lazo cerrado
  e_speed = set_speed - RPM; // determina el error
  integral = integral + (e_speed*0.20); // calcula la integral del error, numéricamente.
  derivada = (e_speed - e_speed_pre)/0.20; // calcula la derivada del error, numéricamente.
  pwm_pulse_int = (kp*e_speed)+ (ki*integral) + (kd*derivada); // Proporcional + Integrativa +
Derivativa

  //pwm_pulse_int = (kp*e_speed) + (ki*integral); // Proporcional + Integrativa
  e_speed_pre = e_speed; //save last (previous) error
  e_speed_sum += e_speed; // suma de los errores para parte de contro Integrativa.

  if(pwm_pulse_int <= 255)
  {
  pwm_pulse=pwm_pulse_int;
  }else
  {
  pwm_pulse=255;
  }
  if(pwm_pulse >= 0 && pwm_pulse <= 255) // entra si el valor del PWM esta entre 0-255
  {
  analogWrite(PWM_OUT,round(pwm_pulse)); // envia el PWM al motor.
  }
  if(set_speed == 0) // entra si FR es de 0
  {
  analogWrite(PWM_OUT,0); // envia PWM de 0 al motor para detenerlo.
  }

 // Enviar datos a terminar virtual
 Serial.print(pulses); // debug value
 Serial.print("\t \t \t");
 Serial.print(pwm_pulse); // debug value
 Serial.print("\t \t \t");
 Serial.print(RPM); // debug value
 Serial.print("\t \t \t");
 Serial.print(left_sensor);
 Serial.print("\t \t \t");
 Serial.println(right_sensor);

 timeold = millis(); // Actializa el temporizador para RPM del motor.
 pulses = 0; // reinicia la cantidad de pulsos detectados.
    attachInterrupt(digitalPinToInterrupt(encoder_pin), ext_ISR, RISING); // reactiva la
interrupcion para cada pulso.
 }
```

```
}

 // *************************
 // **** INTERUPCION POR ENCODER DEL MOTOR DC *********
 // *************************
 void ext_ISR() // entra si se detecta un pulso en encoder por servicio de interrupción.
 {
 pulses++; // Suma el pulso bueno que detecta.
 }
```