Laboratorio 2

Neo Barrios
UTEC
Fraybentos, Uruguay
neo.barrios@estudiantes.utec.edu.uy

Resumen

Este informe describe el desarrollo y análisis de un conjunto de sistemas embebidos programados en C utilizando el microcontrolador ATmega328P, abarcando diferentes tareas de control y automatización. Para el laboratorio 2 se implementaron múltiples algoritmos para ejecutar funciones como el control preciso de un plotter para dibujar patrones usando el algoritmo de Bresenham, la detección de colores utilizando una fotocelda, un controlador de posición para un motor DC, y la regulación de temperatura en una planta. Se realizaron simulaciones de los sistemas al igual que se construyeron en físico la mayoria de los sistemas propuestos.

Introducción

Un plotter es un dispositivo de dibujo automatizado que mueve un cabezal de pluma o marcador sobre una superficie plana para trazar líneas y formas. La primera parte consiste en el desarrollo del firmware para controlar un plotter capaz de dibujar 5 figuras predefinidas. El movimiento del cabezal es controlado por motores DC que permiten desplazar el marcador en los ejes X e Y.

El PT100 es un sensor de temperatura de alta precisión. El microcontrolador ATmega328P recibe las señales del PT100 a través de un puente de Wheatstone con un circuito amplificador de instrumentación, y controla un foco y un ventilador para mantener la temperatura de la planta dentro de un rango.

Por último, se implementó un sistema para la detección de colores utilizando una fotocelda. El sistema fue diseñado para distinguir entre diferentes colores. Las fotoceldas capturan la intensidad de la luz filtrada por una hoja con colores. El microcontrolador ATmega328P, a través de un proceso de calibración y filtrado de señales, identifica el color presente.

Objetivo general

LLevar a cabo el diseño de diferentes algoritmos de control usando lenguaje C con el proposito de entender el funcionamiento de diferentes sistemas y componentes eletrónicos.

Objetivos especificos

- Programar el firmware de un plotter para que mediante una terminal se puedan elejir dibujar diferentes patrones programados.
- Diseñar un circuito para transformar el valor de resistencia de un PT100 en valores analogicos de voltaje que pueda leer un conversor adc del ATmega328p.
- Programar la lógica de un sistema de regulación de temperatura en base al valor del termometro PT100.
- Programar el algoritmo de control de posición de un motor DC en base al valor adc de un potenciometro de referencia y un potenciometro acoplado al eje del motor DC.
- Haciendo uso de python realizar una grafica de los valores acd de los potenciometros y pwm de los pines que controlan el motor DC.
- Implementar un detector de colores, teniendo en cuenta 3 colores registrados y un circulo cromatico disponibles en una hoja, programar un algoritmo y comprobar en la practica si el sistema es capaz de detectar los colores en el circulo cromatico.

Metodología

Marco conceptual

1) Plotter: Un plotter es un dispositivo de dibujo controlado electrónicamente que utiliza uno o más motores para mover un cabezal que puede contener un lápiz, marcador o herramienta de corte. Se usa para trazar gráficos, diseños y dibujos de alta precisión sobre superficies planas, como papel o materiales plásticos. A diferencia de una impresora tradicional, un plotter puede dibujar líneas continuas en lugar de solo puntos, lo que lo hace ideal para aplicaciones como planos arquitectónicos, circuitos impresos o dibujos artisticos.

El plotter utilizado para llevar a cabo el laboratorio es un plotter con precisión finita, esto se refiere a un plotter cuyo movimiento está limitado a incrementos discretos en los ejes X e Y. Esto se debe a la precisión del sistema de motores que controlan el cabezal de dibujo. En lugar de moverse continuamente a lo largo de los ejes, el cabezal se desplaza en pequeños pasos finitos, lo que puede limitar la resolución o exactitud de los trazos. Esta precisión finita depende de la capacidad del sistema de control para ajustar el posicionamiento de los motores y está influenciada por factores como la velocidad de los motores, la velocidad denmutación de los reles que contolan los motores y el algoritmo de control utilizado para calcular los movimientos.

2) Algoritmo de Bresenham: El algoritmo de Bresenham es un método que permite dibujar líneas rectas en una cuadrícula de píxeles utilizando solo cálculos con números enteros, lo que lo hace eficiente para sistemas con limitados recursos de procesamiento, como gráficos en plotters o pantallas de baja resolución.

Funciona calculando el punto inicial de la línea y, en cada paso, determina qué píxel cercano debe activarse para aproximar lo más posible la línea ideal. En lugar de calcular la pendiente de la línea con precisión decimal, el algoritmo evalúa un ”error” que representa la desviación de la línea ideal y ajusta los píxeles activados para corregir esa desviación, manteniéndose lo más cerca posible de la trayectoria deseada.

Su eficiencia proviene del uso exclusivo de sumas, restas y comparaciones enteras, sin necesidad de operaciones con puntos flotantes. Es particularmente útil en dispositivos de dibujo que dependen de motores discretos para mover cabezales, ya que permite definir claramente los pasos que se deben dar en cada dirección (x o y) de manera precisa y sin cálculos complejos.

Aunque este algoritmo fué diseñado con el proposito de trazar lineas en un espacio dominado por los pizeles los cuales son figuras bidimencionales, tambien se puede adaptar a una logica unidimencional para trazar lineas con un plotter.
3) PT100 y Puente de Wheatstone: El puente de Wheatstone es un circuito utilizado para medir resistencias con alta precisión. Consiste en cuatro resistencias conectadas en forma de un cuadrado, donde se mide la diferencia de voltaje entre dos puntos intermedios del circuito.

Para medir la temperatura con una PT100, que es un sensor cuya resistencia varía con la temperatura, el puente de Wheatstone detecta los pequeños cambios de resistencia de la PT100. Al comparar esta resistencia con resistencias conocidas en el puente, se puede calcular la temperatura con precisión, ya que la resistencia de la PT100 está directamente relacionada con la temperatura.

Pero solo con un puente de Wheastone no es suficiente ya que si quisieramos convertir esta variacion de resistencia en una variacion de voltaje en un rango de 0 a 5v para poder medir temperatura usando un conversor adc del ATmega328p se deberia de usar un circuito amplificador de instrumentación ya que la diferencia de voltaje entre los puntos a y b del puente en un rango de unas pocas decenas de grados es de milivoltios.
4) Servomotor y microservomotores: Los servomotres son motores parecidos a los DC pero con la cualidad de poder colocar su eje en un angulo especifico y mantenerlo en el. Hay diferentes tipos de servomotores pero centremosnos en los mas pequeños, los microservomotores. Su funcionamiento consiste en un motor con una caja de engranajes que reduce la velocidad, pero aumenta el torque, con su eje acoplado a un potenciómetro. Un controlador se encarga de regular la dirección y velocidad de giro del pequeño motor DC que contiene para mantenerlo en una posición específica en base a la señal de control que reciba el controlador. El controlador recive una señal PWM de 20ms de periodo, el grado se ajusta entre un rango de duty cicle del PWM siendo por lo general entre 0.5ms (0 grados) y 2.5ms (180 grados). En base al PWM recivido, el controlador gira el motor mediante un puente H. El controlador utiliza la señal de un potenciometro para saber en todo momento en que angulo se encuentra el eje y mediante un algoritmo es capaz de transformar el rango PWM a un rango de grados teniendo en cuenta la posicion usando el potenciometro.

El algoritmo llevado a cabo replica muy bien este funcionamiento pero con algunas diferencias como que el margen posible de giro abarca los 270 grados debido a la construcción del potenciometro de perilla o que el PWM del micro no controla la posicion sino la direccion y velocidad del giro o que el potenciometro que ”mide” el angulo motor esta acoplado directamente al eje, impidiendole a este acoplarse a cualquier otra cosa.
5) LDR y Colores: La luz se divide en colores debido a las diferentes longitudes de onda que la componen. Cada color tiene una longitud de onda específica dentro del espectro visible: el violeta tiene longitudes de onda cortas, mientras que el rojo tiene longitudes de onda largas.

Un LDR (resistor dependiente de luz) detecta la intensidad de luz en lugar de los colores directamente. Sin embargo, al iluminarlo con luz de diferentes colores, el LDR puede mostrar variaciones en la resistencia, ya que cada color tiene una intensidad diferente dependiendo de su longitud de onda y la sensibilidad del material fotosensible del LDR. Esto permite que, mediante calibración, se puedan distinguir diferentes colores basados en los valores de resistencia del LDR.

Materiales

- 1 Protoboard
- 1 Plotter
- Resistencias de 1k y 10k
- 1 resistencia de 100ohm
- 2 potenciometros de 5k
- 1 Fuente de alimentación DC
- 1 Modulo puente H L298N
- 3 amplificadores operacionales
- 1 PT100
- 1 servomotor Sg90
- 1 LDR
- 1 Cable USB A-B
- 1 Arduino UNO
- Software ATMEL STUDIO 7
- Software Arduino IDE
- Software Proteus 8
- Software Visual studio code

Procedimiento

Parte A-Plotter: El codigo para establecer las formas se basa en una función llamada moveTo que como su nombre lo indica sirve para que el plotter se mueva, este moviemiento se basara en los puntos iniciales y los puntos finales que seran pasados como parametros a la función. La función contiene el algoritmo de Bresenham adaptado al plotter y realiza pasos en base a dos valores, uno llamado ’STEP DELAY’ y otro llamado ’CM TO DELAY RATIO’, estos valores establecen los pasos en ms y la cantidad de pasos que equivalen a un cm. Estos valores se basan en que 1mm equivale aproximadamente

a 100ms y siempre para mantener la dimencion de 1cm el valor del ratio tiene que ser 10 veces mayor al step delay.

```
void moveTo(float x0, float y0, float x1, float y1) {

        int stepsX = abs((int)((x1 - x0) * CM_TO_DELAY_RATIO));
        int stepsY = abs((int)((y1 - y0) * CM_TO_DELAY_RATIO));

        int directionX = (x0 < x1) ? 1 : -1;
        int directionY = (y0 < y1) ? 1 : -1;

        int deltaX = stepsX;
        int deltaY = stepsY;

        int err = deltaX - deltaY;
        int e2;

        while (stepsX > 0 || stepsY > 0) {
                e2 = 2 * err;

                if (e2 > -deltaY && stepsX > 0) {
                        err -= deltaY;
                        x0 += directionX * 0.1;
                        stepsX--;
                        digitalWrite((directionX > 0) ? x_right : x_left, 1);
                        _delay_ms(STEP_DELAY);
                        digitalWrite((directionX > 0) ? x_right : x_left, 0);
                }

                if (e2 < deltaX && stepsY > 0) {
                        err += deltaX;
                        y0 += directionY * 0.1;
                        stepsY--;
                        digitalWrite((directionY > 0) ? y_up : y_down, 1);
                        _delay_ms(STEP_DELAY);
                        digitalWrite((directionY > 0) ? y_up : y_down, 0);
                }
        }
}
```

Fig. 1: Función moveTo

Aunque el algoritmo esta hecho con el fin de usar explusivamente numeros enteros la funcion recibe como parametros numeros flotantes porque la logica esta estructurada para que los movimientos no se hagan en un plano milimetrado y no en un plano basado en pasos ya que es mas simple de intuir cuanto son 2.3cm que 230 pasos. Aunque dentro de la función se vuelven a convertir en pasos los valores pasados en cm. Esta conversión de medida temporal a metrica no es exacta, son valores extraidos con observación y aproximación mas no con una medición exacta. Trabajar con un sistema cuadriculado tiene la limitante de que si el paso es de 100ms, osea 1mm, no se puede pedir que el plotter se mueva de una posición (0,0.1) a una posición (0,0.25). Como mucho haría solo un paso.

Parte B-Planta: La figura 2 muestra el esquema del cirucito acondicionador para PT100 haciendo uso de un puente Wheatstone y un amplificador de instrumentación. La resistencia elegida para el puente es de 100 ohm debido a que el valor del pt100 es de 100 ohm a 0 grados centigrados, y el valor de Rg del amplificador de instrumentación es de 10 ohm para amplificar 200 veces la variacion de voltaje, esto se debe a que el voltaje presente entre los dos puntos da en el rango de menos de 0.2v y así se obtien un mapeo de 0-50 grados a 0-5 voltios. Esto en base a los valores experimentales del pt100 a 50 grados que ronda los 120 ohms.

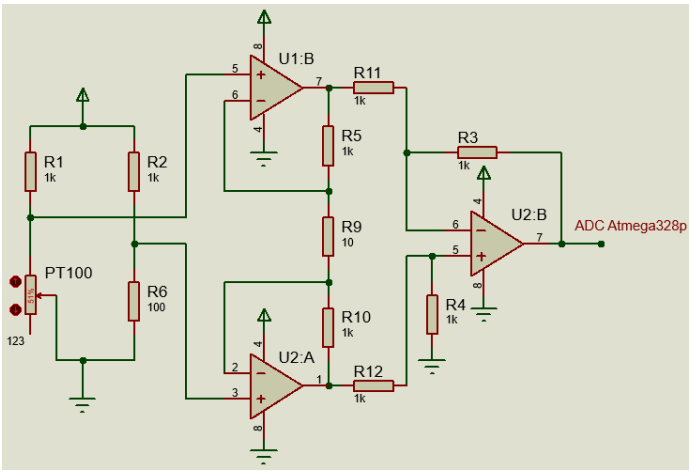

Fig. 2: Esquema de conexiónes para PT100 a ATmega328p

En terminos de logica, la figura 3 muestra un pseudocodigo adaptable facilmente a C, consiste en un sistema de degulación de temperatura basado en la PT100, un foco calentador y un ventilador mostrando la temperatura en tiempo real y el estado de los actuadores del sistema.

```
LEER temperatura desde el sensor PT100
   IMPRIMIR temperatura a través del puerto serie

   SI temperatura < 22:
       ENCENDER calefactor
       APAGAR ventilador
       IMPRIMIR "Calefactor encendido, ventilador apagado"

   SINO SI 23 <= temperatura <= 30:
       APAGAR calefactor
       APAGAR ventilador
       IMPRIMIR "Calefactor y ventilador apagados"

   SINO SI 31 <= temperatura <= 40:
       APAGAR calefactor
       ENCENDER ventilador a baja velocidad
       IMPRIMIR "Calefactor apagado, ventilador baja velocidad"

   SINO SI 41 <= temperatura <= 50:
       APAGAR calefactor
       ENCENDER ventilador a velocidad media
       IMPRIMIR "Calefactor apagado, ventilador velocidad media"

   SINO SI temperatura > 51:
       APAGAR calefactor
       ENCENDER ventilador a alta velocidad
       IMPRIMIR "Calefactor apagado, ventilador alta velocidad"

   ESPERAR 5 segundos
```

Fig. 3: Pseudocodigo

Parte C-Servomotor: Los potenciometros iran conectados a los pines adc 0 y adc 1 del microcontrolador y seran polarizados a 5v, por otro lado el motor será conectado a un driver puente H para controlar si sentido e intensidad de giro mediante PWM de los pines PB1 y PB2. Luego mediante la logica en C que se puede ver en la figura 4, se intentará contolar la posicion del motor para que coincida con la del otro potenciometro.

```
void control_pwm(int16_t adc_motor, int16_t adc_referencia) {
        int16_t error = adc_referencia - adc_motor;

        if (error > 0) {
                OCR1A = error;
                OCR1B = 0;
                } else if (error < 0) {
                OCR1B = -error;
                OCR1A = 0;
                } else {
                OCR1A = 0;
                OCR1B = 0;
        }
        if (OCR1A > 1024) OCR1A = 1024;
        if (OCR1B > 1024) OCR1B = 1024;
}
```

Fig. 4: Función de control

Dicho algoritmo está basado en el diagrama de la figura 5

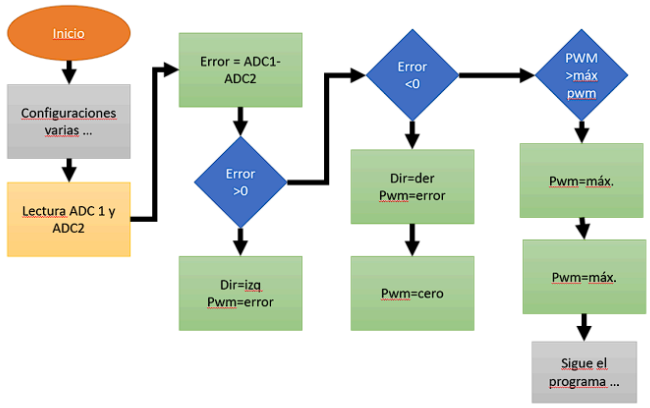

Fig. 5: Diagrama de control

Parte D-Detector de color: La LDR tiene que ser adondicionada para poder leer una variación de voltaje segun el color en un ADC del microcontrolador. Para ello se dispone de la figura 6 para mostrar su configuración haciendo uso de un divisor de tensión.

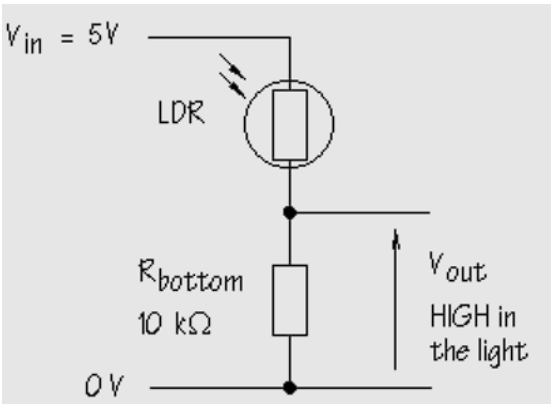

Fig. 6: Circuito de acondicionamiento para LDR

La logica del algoritmo se basa en determinar primero el valor analogico mapeado de un valor de voltaje de 0-5V a un valor de 10 bits de resolución. Este valor será leido mediante el puerto serial y cuando se tengan los valores medidos de los colores se procedera a sustituir esos valores en las definiciones globales (valorRojo) (valorAmarillo) (valorRojo), estos valores serán usados posteriormente en la función 'decoder' que se ve en la figura 7.

```
int decorder(int valorADC) {
        if (valorADC >= (valorRojo - margenError) && valorADC <= (valorRojo + margenError)) {
                diferencia=valorADC-valorRojo;
                return pwmRojo;
                }
        else if (valorADC >= (valorAmarillo - margenError) && valorADC <= (valorAmarillo + marenError)) {
                diferencia=valorADC-valorAmarillo;
                return pwmAmarillo;
                }
        else if (valorADC >= (valorVerde - margenError) && valorADC <= (valorVerde + margenError)) {
                diferencia=valorADC-valorVerde;
                return pwmVerde;
                }
        else {
                diferencia=0;
                return 0;
        }
}
```

Fig. 7: Función decoder

La logica de esta función rige por la determinación del color en base a un margen de error, este valor de margen es una definición globlar, este está puesto para ser de de 10 debido a que la diferencia analogica entre el verde y el rojo es poca. Por esto tambien pueden haber mal entendidos a la hora de medir el color en el circulo cromatico debido a que algunos tonos de rojo que tienen al amarillo tienen valores iguales a algunos tonos de verde que tienden a amarillo.

RESULTADOS

Los videos donde se muestra el funcionamiento de la parte A,C y D al igual que los codigos comentados se encuentran en la carpeta llamada "Lab2" en el repositorio de github citado en Anexos.

Parte A

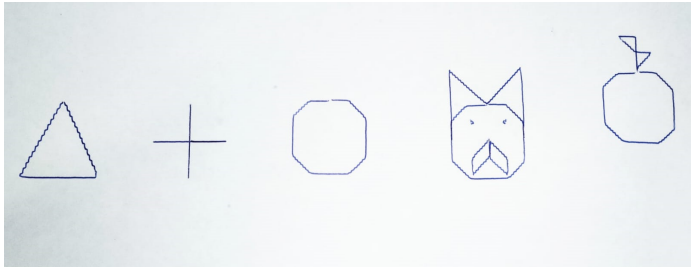

Fig. 8: Dibujos

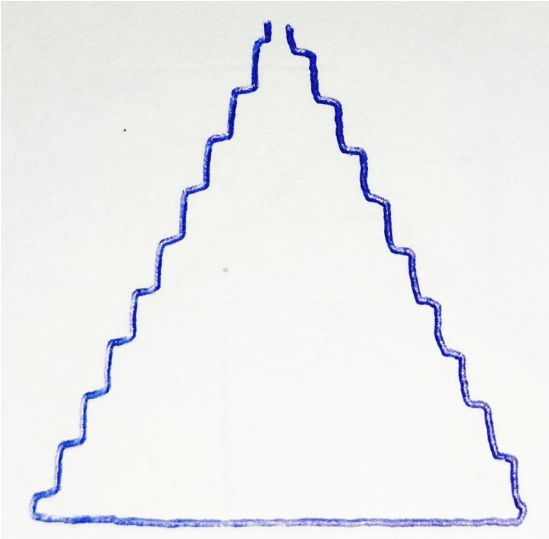

Fig. 9: Triángulo con step de 100ms

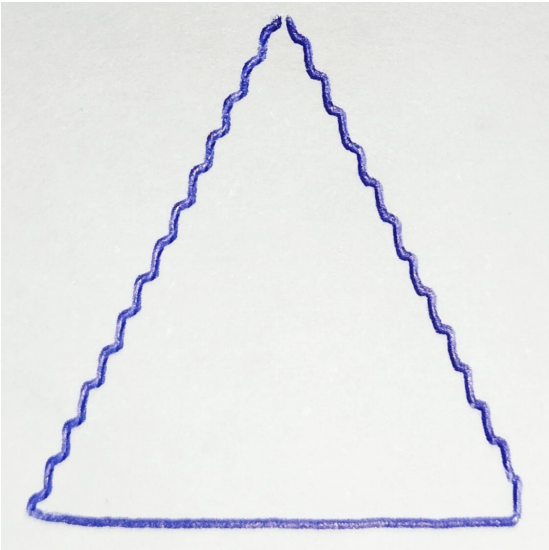

Fig. 10: Triángulo con step de 70ms

Se logra apreciar una pequeña diferencia en la calidad de ambas formas debido a una reducción en el tiempo de cada step, mientras la figura 9 tiene una apariencia mas escalonada, la figura 10 adquiere una apariencia mas lineal. Esta diferencia radica en que el step de la figura 9 es de 100ms, mientras que el tiempo de la figura 10 es de 70ms, aunque ambas figuras tengan steps diferentes la dimension sigue siendo la misma en ambos casos debido a un balance hecho con el valor de ratio de steps por cm, siendo que aunque el step disminuyó, su ratio de steps per cm aumentó. En este caso tuvo que haber sido aumentado a 14, teniendo un pequeño error de -0.2cm cada 14 pasos.

Parte C

En la figura 11 se muestran las graficas de valores ADC del potenciometro de referencia y el potenciometro conectado al eje del motor. Estos valores de adc tienden a ser los mismos, por otro lado los pines de pwm muestran la compensación que produce el pwm para que los valores de ADC se intenten igualar, tendiendo estos a ser igual a 0 cuando los valores ADC tienden a ser iguales.

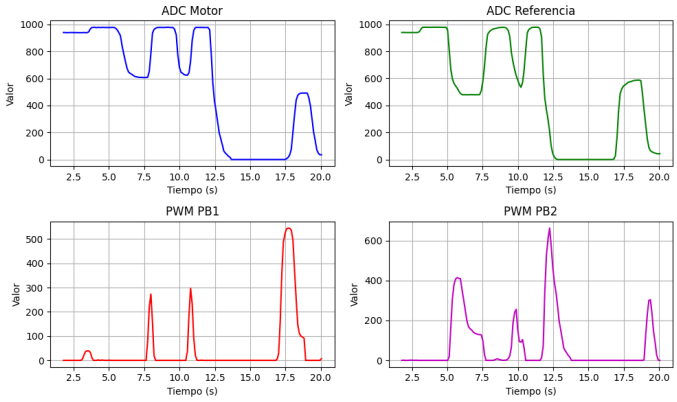

Fig. 11: Graficas de valores ADC y PWM

Conclusiones

Parte A

El algoritmo aunque funciónal presenta una falla al graficar puntos que solo tienen 1 step y cuando se intenta graficar un cuadrado de 2 steps por 2 steps el plotter grafica un cuadrado de 1x1 step lo que indicaria que la función no dibuja el ultimo paso, esto talvez se solucionaria implementando una logica para que cuando termine de dibujar la linea, que dibuje 1 step mas respetando la logica del algoritmo y siguiendo la trayectoria de la linea. Acerca de las figuras: Aunque si es posible crear una función que dibuje un circulo con diferentes radios, se puede prescindir de ello para este caso porque un octagono es una de las posibles aproximaciónes finitas de un circulo. El oso fué hecho basado en el oso rojo o tambien llamado panda rojo el cual tiene mas rasgos caninos que ursos.

El algoritmo tampoco fué totalmente probado por lo que no se tiene total serteza de su buen funcionamineto, la unica evidencia rescatada de que el algoritmo funcióna es el escalonamiento del triangulo pero como las debas figuras usan lineas rectas o a 45 grados no quedó en evidencia el poder real del algoritmo para dibujar.

Para terminar, el codigo precente en github sobre la parte A no es el cargado al atmega, esto debido a un error a la hora de ser modificado en el proceso de testeo, el codigo usado para el video de evidencia no fue guardado, aunque la unica diferencia entre el de github y el modificado tiene una cordenada mal puesta para el patron del circulo debido a un signo negativo y que el piston siempre estaba abajo y nunca subia debido a la ausencia de setear en 1 el pin para levantar el piston.

Parte B

El valor de Rg fué supuesto en base a la tabla de conversion de resistencia a grados más no fué calculado pero se estima

que sería mas conveniente usar resistencias de 10k para el amplificador operacional para poder colocar una resistencia de 100ohm en el lugar de Rg.

Parte C

El principal inconveniente en esta parte ha sido la falta de presición a la hora de que el motor replicará la posición del potenciometro de referencia, esto debido principalmente a la caja reductora que presenta este debido a que no es un motor puramente DC, sino que es un servomotor. Con la ausencia de esta caja reductora se esperaría que el motor lograse ajustarse con mayor presición a la posición deseada. Una solución propuesta, y la cual se puede ver en el video de evidencia a modo de comparación es implementar un offset. Esto debido a que cuando la diferencia entre los valores de los potenciometros es menor a 100, el PWM tambien es menor a 100, y sucede que cuando es menor a 100 el motor no tiene la potencia suficiente como para moverse, pero esto se solucionaría incrementandole 100 al valor variable del error, aunque de esta forma los pasos son mas bruscos y menos suaves. Otra solución que podría ser aún mejor habria sido mapear el dato del error para que cuando el error sea 0 el valor del PWM sea 100 y cuando el valor error sea 1024 que el valor del PWM sea 1024.

Parte D

Aunque considerar hacer un acondicionamiento de la LDR usando un puente de Wheatstone parecia ser una idea correcta, la verdad es que usandso un divisor de tensión se puede lograr el resultado esperado. En este caso como la hoja estaba pegada a la ventana, los valores de cada colores en el ADC cambiaban en todo momento por la hora del dia, el grado del sol con respecto a la tierra, las nuebes, los pajaros, etc. Por lo que una idea pensada que no se llevo a cabo fué colocar una LDR por cada color a registrar, y de esta forma el sistema podria detectar en todo momento con mas exactitud que color se estaba leyendo del circulo cromatico. La detección del color en el puerto serial dió problemas y por eso no aparece en el video de evidencia. Esto se debe a un problema a la hora de mandar el resutlado de la detección de colores. Si bien el sistema puede seguir siendo capaz de detectar el color, la comunicación se corta abruptamente una vez se detecta un color, aunque en algunos casos es capaz de mostrar que color esta leyendo pero al cabo de 1 o 2 lecturas más se cierra el puerto serial.

BIBLIOGRAFÍA

- (S/f). Naylampmechatronics.com. Recuperado el 28 de septiembre de 2024, de https://naylampmechatronics.com/blog/11_tutorial-de-uso-del-modulo-l298n.html
- Features, D. (s/f). 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash. Microchip.com. Recuperado el 28 de septiembre de 2024, de https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
- (S/f). Sterlingsensors.co.uk. Recuperado el 28 de septiembre de 2024, de https://www.sterlingsensors.co.uk/pt100-resistance-table
- (S/f). Sterling Sensors. Recuperado el 28 de septiembre de 2024, de https://www.sterlingsensors.co.uk/pt100-resistance-table
- (S/f-b). Digital Bunker. Recuperado el 28 de septiembre de 2024, de https://digitalbunker.dev/bresenhams-line-algorithm/

ANEXOS

Repositorio de github