PC3 Control inteligente de velocidad de motor

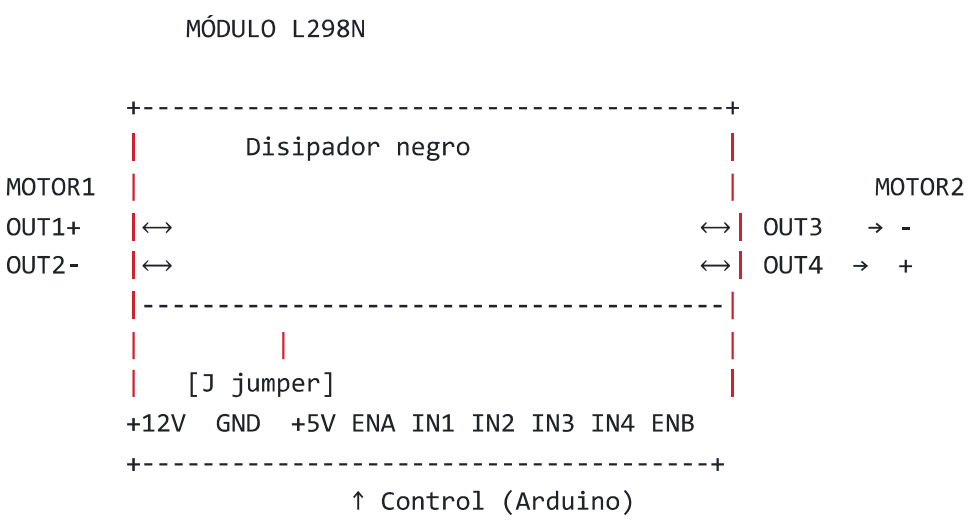
Grupo 1 Control inteligente de velocidad de un motor DC mediante un L298N conectado al Arduino UNO.

- Omar Romulo Quispe Santos
- Esau Flores Villar
- Milagros Cristina Ruiz Arica
- Brian Alberto Huaman Garcia
- David Fernando Reeves Goñi
- Jhan Carlos Ccanto Quispe

Se implementa una interfaz con Tkinter que:

- Permite ajustar la velocidad del motor mediante un Scale (PWM de 0–255).
- Muestra la velocidad actual en porcentaje.
- Incluye un botón de inicio y parada que detenga inmediatamente el motor.
- Mantiene el motor a más del 80% de velocidad durante más de 10 segundos, muestra una alerta visual (roja) en la interfaz.
- Integra un gráfico de evolución de velocidad vs. tiempo

L298n es un modulo para manejar motores , sera el puente entre el motor que requiere potencia y el arduino , quien trabaja con voltajes menores.



Se usara el MOTOR 1 luego se conectara estos bordes de potencia(CAJAS AZULES) al motor DC en cuestión OUT1 y OUT2.En este caso los cables de nuestro motor no tiene polaridad fija entonces los conectamos de forma indistinta. Seguidamente la entrada EN2 se conecta al pin 7 del arduino, para luego hacer lo propio entre la entrada EN1 y el pin 8 del arduino. De modo que :

```
PIN 8 =1 →→ IN1 = 1 →→ OUT1 =6V
PIN 7 =0 →→ IN2 = 0 →→ OUT2=GND=0
#obtenemos el giro en un sentido
PIN 8=0 →→  IN1=0  →→  OUT1=GND
PIN 7 =1 →→ IN2=1  →→ OUT2=6V
#obtenemos el giro contrario
```

Ademas si se quiere variar la velocidad conectamos el pin 9(pwm) del arudino al jumper ENA de activacion para el MOTOR1

La bateria por su parte se conecta a los bordes de potencia (CAJAS AZULES) 12V(+) y GND(- referencia electrica "tierra") respectivamente. Ahora bien como se va a medir potencias tanto arduino como L289n deben tener la misma referencia desde el cual medir dicho voltaje, luego se conecta tanto el negativo de la bateria como el GND del arduino al GND del l289n.

Pasos previos

Creamos una variable 'arduino' usamos serial.Serial() para la vinculacion con arduino transmitiendo 9600 bit por segundo, usamos modulacion por ancho de banda, para simular un voltaje variable, usaremos para ese fin un pin digiral HIGH-LOW.

Actualizar_valor() define una funcion interna para la modificacion del valor que sera enviado al arduino via **arduino.write()**

Palanca_motor() define una variable global motor_encendido si este este tiene valor 1 apagamos motor Caso contrario encendemos Luego se envia a arduino {valor},{1 ó 0} codificados

seguidamente se crea la **ventana = tk.Tk()**, asignamos un titulo, definimos el tamaño de la ventana.

y el slider dentro de la ventana con los argumentos recomendados, con el **command' = _on_escale** , de modo que manipulando el handle modificamos el valor enviado al arduino. El slider se posiciona via place() quien usa el sistema x y y hacia abajo ,lo posicionamos el el centro **anchor='center'**

Tambien se crea un **stop_motor** dentro de ventana y el **command** usado por este es palanca_motor , que justamente enciende o apaga el motor

La ventana se muestra y al finalizar cerramos el arduino

Para el testeo usamos un mock que simula la conexion serial con arudino UNO.

Implementacion real

Sin embargo la implementacion real es un tanto más complejo. El codigo es reformulado y repotenciado

importamos los modulos de utilidad:
tkinter, serial , matpltlib, threading y time, ademas **collections.deque** una estructura para almacenar datos. Con esto tendremos un dashboard en python que controlara un motor al cual tendremos acceso via conexion serial con arduino.

Se definen constantes globales:

- PWM_MAX = 255 (maxima velocidad del motor)
- ALERT_SECONDS = 10.0 (para la alerta pedida)
- PLOT_WINDOWS_SECONDS = 60 (graficamos los 60 sgs ultimos)

Seguidamente definimos nuestra clase principal **MotorControllerApp**

El constructor de esta clase, definen atributos utiles como :

- self.root la pantalla Tk()
- self.motor_running para controlar el estado del motor

- self.times = collections.deque() para tiempos
- self.speeds = collections.deque() para velocidades

Y Algunas funciones:

- _build_ui(): Construye la interfaz grafica con:
  - Combobox's (similar al del IDE audino) para eligir 'COM'
  - con un boton "conectar"
  - El slider self.Scale para elegir PWM
  - Botones Inicio Parar
  - El lienzo Figure() para el grafico de Matplotlib
- _list_serial_ports() : muestra los COM disponibles
- _refresh_ports() : actualiza los Combobox, si hay nuevos
- connect_serial(): Abre una conexion serial con el puerto seleccionado en el Combobox
- send_serial(msg) : envia el mensaje al arduino
- _on_scale(val) :actuliza el valor de PWM, lo procesa para ser enviado en last_send_value
- start_motor(): envia la velocidad a arduino ..
- _update_loop():
  - Actualiza el tiempo
  - Guarda los valores para la grafica
  - Revisa alertas
  - Programa la siguiente ejecucion
- _start_periodic_update() arranca el ciclo que analiza cada 500ms las metricas deseadas

Mientras que para el IDE Arduino se define los siguiente:

```
const int pwma = 9; # PWM pin (PWMA)
const int ain1 = 8; #INT1
const int ain2 = 7; #INT2
const int stby = 6;  #activa o desactiva el driver
```

En void setup() : activamos los pines de salida y abrimos el puerto serial

```
Serial.begin(9600);
pinMode(pwma, OUTPUT);
pinMode(ain1, OUTPUT);
pinMode(ain2, OUTPUT);
pinMode(stby, OUTPUT);
```

Seguidamente activamos el driver **digitalWrite(stby, HIGH)** , se establecen las direcciones por defecto , las indicadas al inicio

```
digitalWrite(ain1, HIGH);
digitalWrite(ain2, LOW);
```

En el loop(): recibimos los valores el texto encodeado, y obtenemos las directivas, **S:120 ,STOP, EMERGENCY_STOP** si tenemos un S , ajustamos la velocidad via el siguiente bloque:

```
if (line.startsWith("S:")) {
  int val = line.substring(2).toInt();
  val = constrain(val, 0, 255);
  analogWrite(pwma, val);
  Serial.print("OK S:");
  Serial.println(val);
}
```

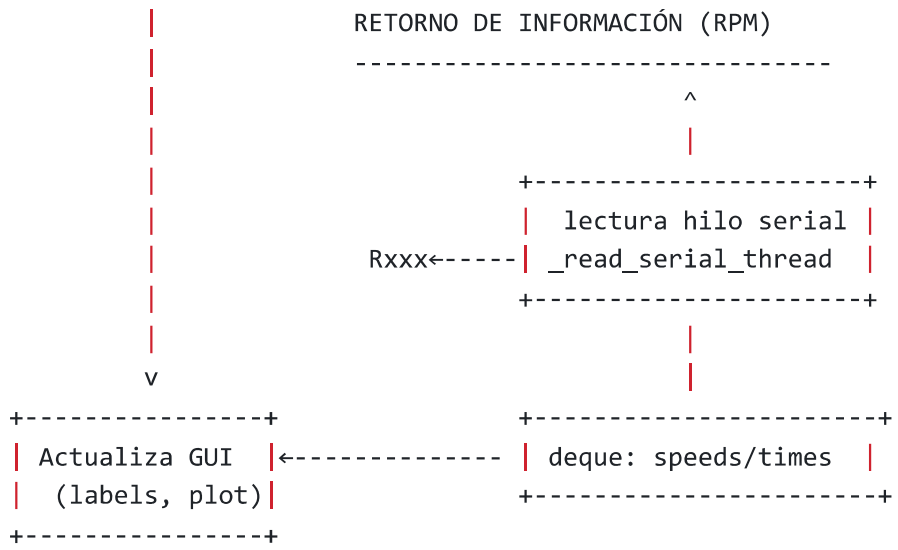
Donde se extrae el valor para la velocidad, y lo aplicamos a PWMA

Finalmente se muestra un esquema del funcionamiento del sistema

```
                    CONTROL DE VELOCIDAD — FLUJO COMPLETO
                    =====================================

                GUI (Tkinter)                    PC (Python)
                --------------                    ------------

                +-----------------+              +-----------------+        +
  Slider mueve → |   Slider PWM    | --evento-->  | handle_speed    | --Sxxx→|
                +-----------------+              +-----------------+        +
                        |                                 |
                        |                                 |
                        |                                 v
                        |                       +----------------------+
                        |                       |  serial.write('Sxx') |---
                        |                       +----------------------+
                        |
                        |
                        |
```


RETORNO DE INFORMACIÓN (RPM)
lectura hilo serial
Rxxx
_read_serial_thread
deque: speeds/times
Actualiza GUI
(labels, plot)