PRÁCTICA DE ARDUINO - HILADORA AUTOMÁTICA

NIVEL: 2
DURACIÓN: 2 horas
OBJETIVO: Los estudiantes aprenderán a crear una hiladora automática
MATERIALES

- Arduino Uno (o similar)
- 2 Motores DC con encoder
- Teclado matricial 4x4 + PCF8574T
- LCD + PCF8574
- Driver L298N
- Cables y protoboard
- Computadora con el software Arduino IDE instalado
- Proteus

1. DESARROLLO DE LA PRÁCTICA

1.1 SIMULACIÓN EN PROTEUS

1.1.1 MONTAJE

Andrés Corredor – andfelcorredor@uniboyaca.edu.co

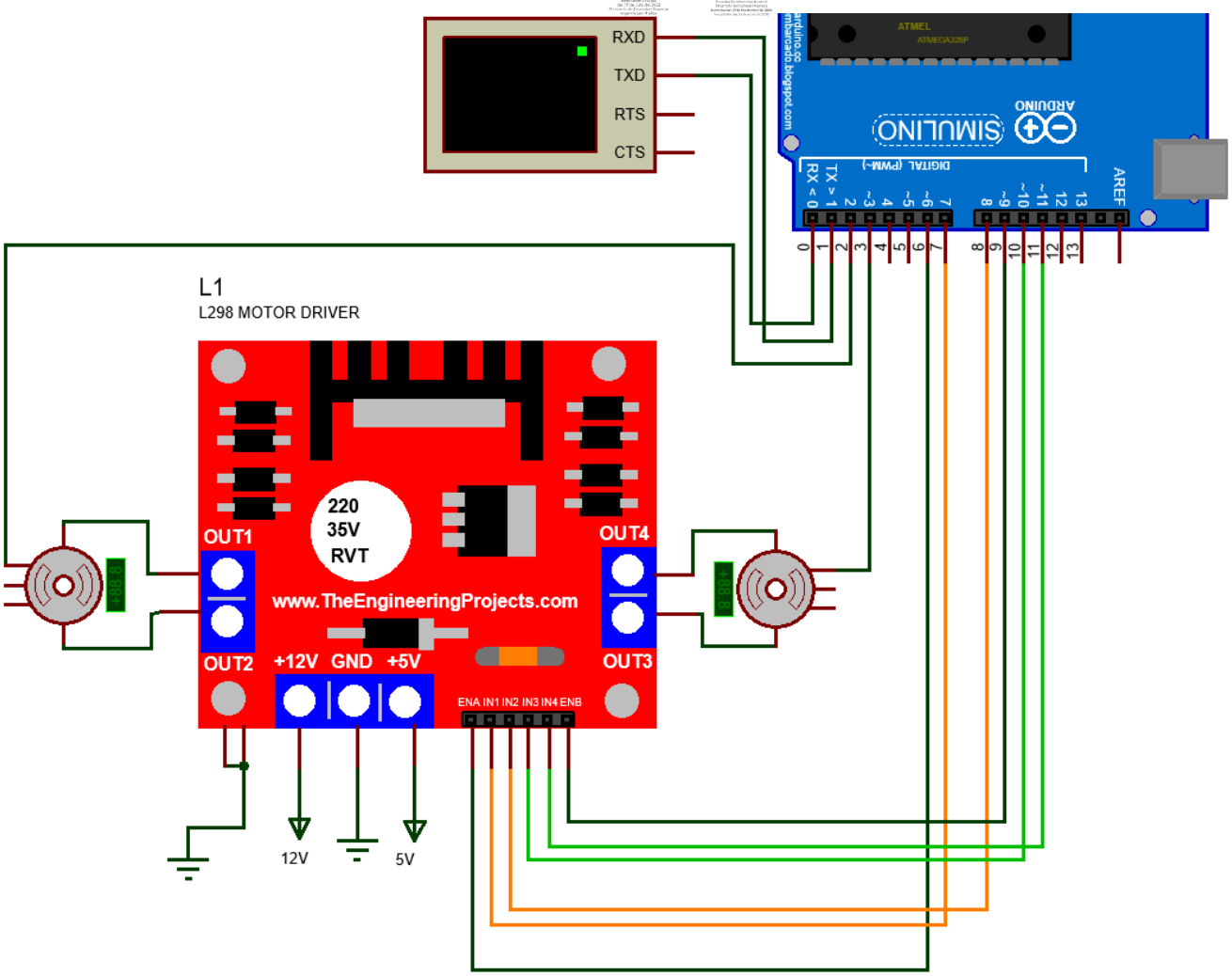

Figura 1. Montaje en pines Digitales

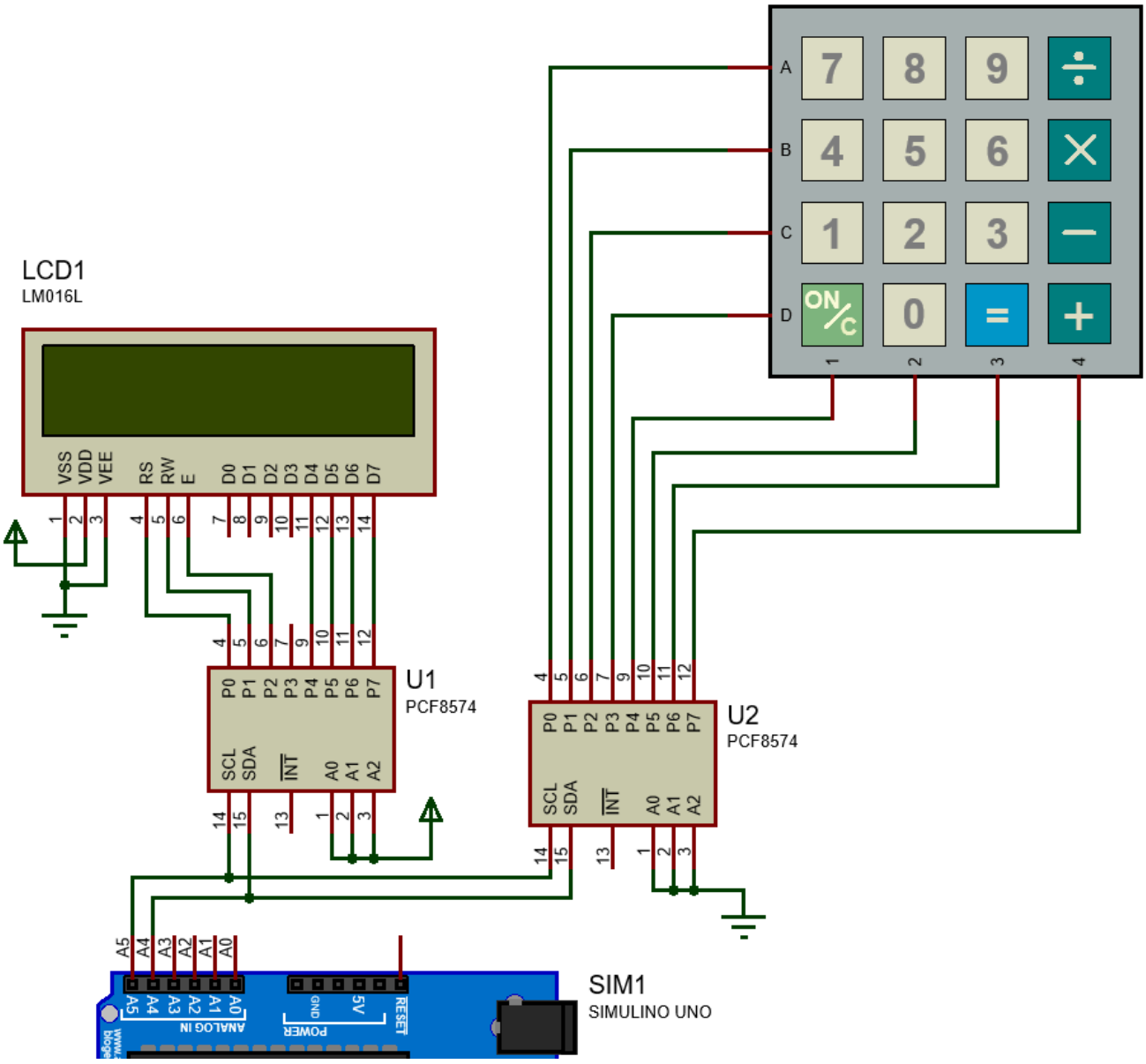

Figura 2. Montaje en pines Analogicos

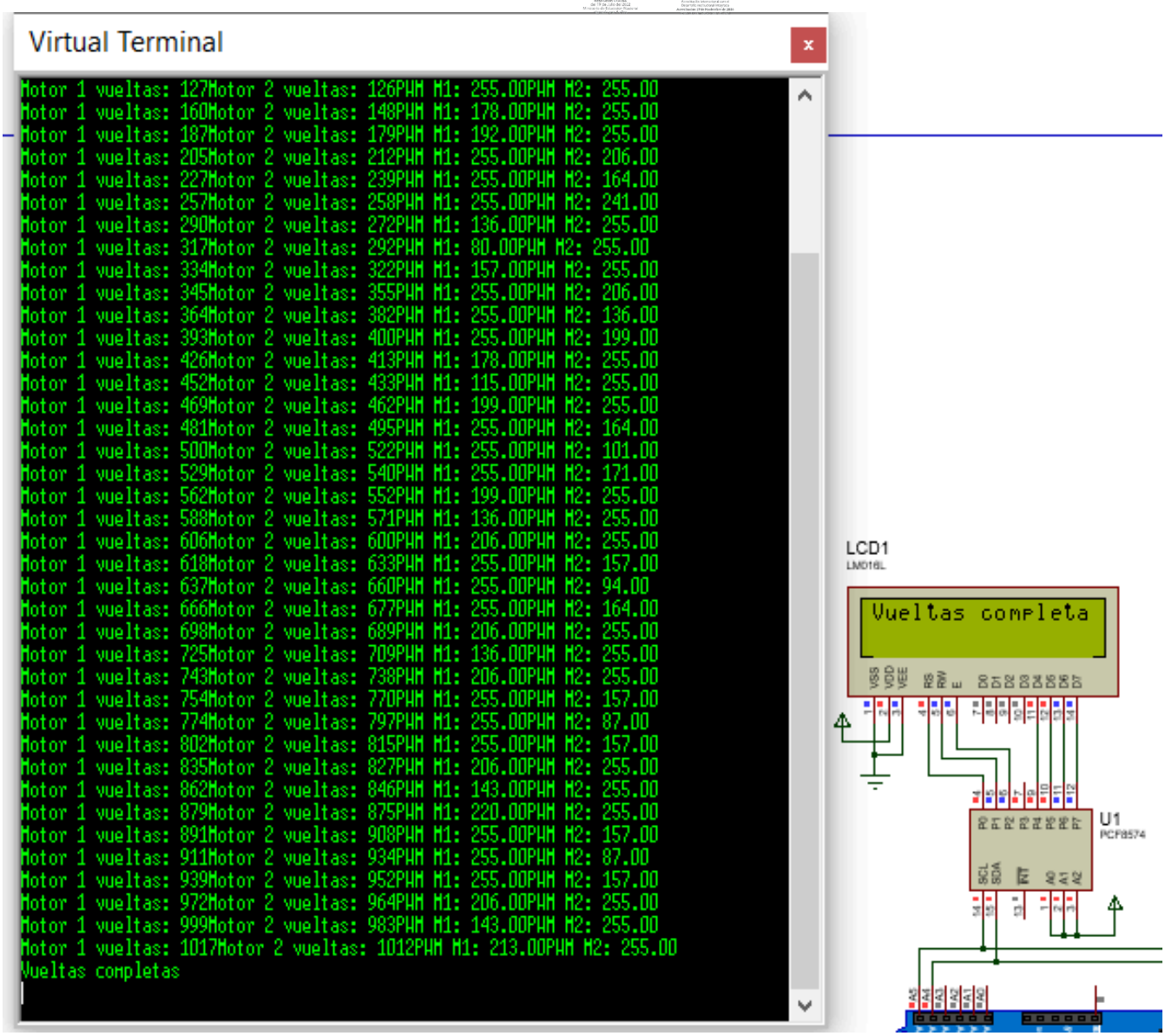

Figura 3. Resultado para 1020 vueltas

2.2 PROGRAMACIÓN EN ARDUINO IDE

```
#include <Keypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <PCF8574.h>

// Configuración del teclado con PCF8574
PCF8574 tecladoPCF(0x20); // Dirección del PCF8574 para el teclado (ajusta según tu configuración)
const int filas[4] = {0, 1, 2, 3}; // Pines de las filas en el PCF8574 (P0-P3)
const int columnas[4] = {4, 5, 6, 7}; // Pines de las columnas en el PCF8574 (P4-P7)
char keys[4][4] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

// Configuración del LCD con PCF8574
LiquidCrystal_I2C lcd(0x27, 16, 2); // Dirección del PCF8574 del LCD

// Pines de los encoders
int pinEncoderA1 = 2; // Encoder Motor 1
int pinEncoderA2 = 3; // Encoder Motor 2
volatile int contadorMotor1 = 0; // Contador de pulsos para Motor 1
volatile int contadorMotor2 = 0; // Contador de pulsos para Motor 2

// Pines del driver L298
int ENA = 6;  // Pin PWM para el motor 1
int ENB = 9;  // Pin PWM para el motor 2
int IN1 = 7;  // Pin de control del motor 1
int IN2 = 8;
int IN3 = 10; // Pin de control del motor 2
int IN4 = 11;

int vueltasObjetivo = 0; // Vueltas objetivo ingresadas por el usuario

// Variables para el control PID
float kp = 2.0;  // Ganancia proporcional
float ki = 6.0;  // Ganancia integral
float kd = 0.01; // Ganancia derivativa
float Tm = 0.1;  // Tiempo de muestreo

// Variables PID para ambos motores
float error1, error2;
float cv1, cv2;
float errorPrev1 = 0, errorPrev2 = 0;
float integral1 = 0, integral2 = 0;

void setup() {
  // Configuración del LCD
  Wire.begin();
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Ingresa vueltas:");

  // Configuración del teclado
  tecladoPCF.begin();
```

```
  // Configuración de los encoders
  pinMode(pinEncoderA1, INPUT);
  pinMode(pinEncoderA2, INPUT);
  attachInterrupt(digitalPinToInterrupt(pinEncoderA1), contarPulsosMotor1, RISING);
  attachInterrupt(digitalPinToInterrupt(pinEncoderA2), contarPulsosMotor2, RISING);

  // Configuración de los pines del driver L298
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  // Inicialización del Monitor Serial
  Serial.begin(115200);
  Serial.println("Sistema iniciado. Ingrese el número de vueltas:");
}

void loop() {
  char tecla = leerTeclado(); // Leer la tecla presionada
  if (tecla) {
    if (tecla >= '0' && tecla <= '9') { // Si se presionó un número
      vueltasObjetivo = vueltasObjetivo * 10 + (tecla - '0'); // Construir el número de vueltas
      lcd.setCursor(0, 1);
      lcd.print("Vueltas: ");
      lcd.print(vueltasObjetivo); // Mostrar en la LCD
    } else if (tecla == '#') { // Confirmar vueltas con '#'
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Comenzando...");
      Serial.print("Vueltas objetivo: ");
      Serial.println(vueltasObjetivo);
      iniciarMotores(); // Función que inicia los motores con las vueltas ingresadas
    }
```

```
  }

  delay(500); // Actualización cada medio segundo
}

// Función para leer el teclado conectado al PCF8574
char leerTeclado() {
  for (int fila = 0; fila < 4; fila++) {
    tecladoPCF.write(filas[fila], LOW); // Configurar la fila como salida baja
    for (int columna = 0; columna < 4; columna++) {
      if (tecladoPCF.read(columnas[columna]) == LOW) {
        delay(50); // Esperar para evitar rebotes
        while (tecladoPCF.read(columnas[columna]) == LOW);
        return keys[fila][columna]; // Devolver la tecla presionada
      }
    }
    tecladoPCF.write(filas[fila], HIGH); // Restaurar la fila a estado alto
  }
  return 0; // Si no se presionó ninguna tecla
}

// Función para contar los pulsos del encoder del Motor 1
void contarPulsosMotor1() {
  contadorMotor1++;
}

// Función para contar los pulsos del encoder del Motor 2
void contarPulsosMotor2() {
  contadorMotor2++;
}

// Función para iniciar los motores y controlar la velocidad con PID para sincronizar las vueltas
void iniciarMotores() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);

  while (contadorMotor1 < vueltasObjetivo || contadorMotor2 < vueltasObjetivo) {
    // Control PID para Motor 1
    error1 = vueltasObjetivo - contadorMotor1;
    integral1 += error1 * Tm;
    cv1 = kp * error1 + ki * integral1 + kd * (error1 - errorPrev1) / Tm;
    errorPrev1 = error1;

    if (cv1 > 255.0) cv1 = 255.0;
    if (cv1 < 0) cv1 = 0;

    // Control PID para Motor 2
    error2 = vueltasObjetivo - contadorMotor2;
    integral2 += error2 * Tm;
    cv2 = kp * error2 + ki * integral2 + kd * (error2 - errorPrev2) / Tm;
    errorPrev2 = error2;

    if (cv2 > 255.0) cv2 = 255.0;
    if (cv2 < 0) cv2 = 0;

    // Sincronización de motores
    // Si un motor ha completado más vueltas que el otro, reducir su velocidad proporcionalmente a la diferencia
    int diferenciaVueltas = abs(contadorMotor1 - contadorMotor2);
```

```
    if (contadorMotor1 > contadorMotor2) {
      cv1 -= diferenciaVueltas * 7;  // Reducción más fuerte proporcional al exceso de vueltas
      if (cv1 < 50) cv1 = 50;  // Límite mínimo más bajo para el motor 1
    } else if (contadorMotor2 > contadorMotor1) {
      cv2 -= diferenciaVueltas * 7;  // Reducción más fuerte proporcional al exceso de vueltas
      if (cv2 < 50) cv2 = 50;  // Límite mínimo más bajo para el motor 2
    }

    analogWrite(ENA, cv1); // Ajustar PWM del motor 1
    analogWrite(ENB, cv2); // Ajustar PWM del motor 2

    // Mostrar las vueltas en la pantalla LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Vueltas:");
    lcd.setCursor(0, 1);
    lcd.print("M1:");
    lcd.print(contadorMotor1);
    lcd.print(" M2:");
    lcd.print(contadorMotor2);

    // Mostrar información en el Monitor Serial
    Serial.print("Motor 1 vueltas: ");
    Serial.print(contadorMotor1);
    Serial.print("\tMotor 2 vueltas: ");
    Serial.print(contadorMotor2);
    Serial.print("\tPWM M1: ");
    Serial.print(cv1);
    Serial.print("\tPWM M2: ");
    Serial.println(cv2);

    delay(500);  // Actualización cada 500ms
  }

  // Detener los motores una vez alcanzado el objetivo
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Vueltas completas");

  // Mostrar mensaje en el Monitor Serial
  Serial.println("Vueltas completas");
}
```

Explicación

- Este código implementa el control de dos motores DC con encoders utilizando un driver L298 y un teclado matricial para ingresar el número de vueltas que los motores deben hacer. El sistema emplea control PID para sincronizar ambos motores y asegurar que completen el número de vueltas especificado.

- Inclusión de librerías:
  - Keypad.h: Para manejar el teclado matricial conectado a través del módulo PCF8574.
  - Wire.h: Para la comunicación I2C entre el microcontrolador y los módulos periféricos.
  - LiquidCrystal_I2C.h: Para controlar la pantalla LCD a través de I2C.
  - PCF8574.h: Para utilizar el expansor de pines PCF8574 con el teclado y el LCD.

- Se usa un teclado matricial de 4x4 conectado a un expansor PCF8574. Las filas y columnas del teclado están definidas en arrays para manejar las teclas de forma adecuada.

- Se utiliza una pantalla LCD conectada también a través del PCF8574, permitiendo mostrar mensajes como las vueltas ingresadas y el progreso de los motores.

- Se conectan dos encoders, uno para cada motor, cuyos pulsos se cuentan mediante interrupciones. Cada pulso se registra en las variables contadorMotor1 y contadorMotor2.

- Los pines ENA, ENB, IN1, IN2, IN3, IN4 están configurados para controlar los motores DC a través del driver L298.

- El control PID se utiliza para ajustar la velocidad de los motores basándose en el error (diferencia entre las vueltas objetivo y las vueltas actuales). Las ganancias del controlador PID (kp, ki, kd) están definidas para ajustar la salida de PWM que controla la velocidad de los motores.

- El sistema espera la entrada del usuario a través del teclado. Las teclas numéricas permiten ingresar el número de vueltas deseadas, y al presionar #, el sistema inicia el movimiento de los motores.

- Si un motor está avanzando más rápido que el otro, el código ajusta dinámicamente el valor del PWM del motor más rápido para igualar las vueltas de ambos motores.

2. ANÁLISIS DE RESULTADOS

- Cómo se ingresan las vueltas objetivo?

Las vueltas se ingresan mediante el teclado matricial. Al presionar números, se construye el valor de vueltas, y se confirma con #.

- ¿Qué ocurre después de ingresar las vueltas?

El sistema muestra el número de vueltas en la pantalla LCD y, al confirmar con #, los motores empiezan a girar.

- ¿Cómo se cuentan las vueltas de los motores?

Los encoders conectados a los motores generan pulsos. Estos pulsos se cuentan mediante interrupciones en los pines digitales del microcontrolador.

- ¿Cómo se ajusta la velocidad de los motores?

La velocidad de los motores se ajusta usando un control PID que regula el valor de PWM en función de la diferencia entre las vueltas actuales y las vueltas objetivo.

- ¿Por qué se usa un control PID?

El PID permite ajustar la velocidad de los motores de manera precisa, para que ambos motores completen el mismo número de vueltas de manera sincronizada, a pesar de las posibles diferencias en carga o rendimiento.

- ¿Qué ocurre si un motor avanza más rápido que el otro?

El sistema detecta la diferencia de vueltas y reduce el PWM del motor que está avanzando más rápido, ajustándolo para que se igualen.

- ¿Cómo se muestra el progreso en el LCD?

Durante el funcionamiento, el LCD muestra el número de vueltas actuales de ambos motores en tiempo real.

- ¿Cómo se muestra el progreso en el Monitor Serial?

Se imprime el número de vueltas de cada motor y los valores de PWM actuales para cada motor, lo que permite monitorear el funcionamiento en tiempo real desde el Monitor Serial.

- ¿Cómo se maneja la diferencia de vueltas?

Se calcula la diferencia entre los contadores de los dos motores, y se ajusta el PWM del motor más rápido para reducir esa diferencia.

- ¿Qué ocurre cuando ambos motores alcanzan el número de vueltas objetivo?

Cuando ambos motores alcanzan el número de vueltas objetivo, el sistema detiene ambos motores, muestra "Vueltas completas" en la pantalla LCD y lo informa en el Monitor Serial.

- ¿Por qué se limita el PWM mínimo a 50?

Para evitar que los motores se detengan completamente durante el ajuste del PID. Un valor muy bajo de PWM podría no ser suficiente para mover los motores.

- ¿Cómo se ajusta el tiempo de muestreo del PID (Tm)?

El tiempo de muestreo está definido como 0.1 segundos, lo que afecta la frecuencia con la que se actualizan los cálculos del PID.

- ¿Qué pasa si se ingresa un valor no numérico en el teclado?

Solo las teclas numéricas son válidas para ingresar el número de vueltas. El código ignora las otras teclas, excepto #, que confirma la entrada.

- ¿Qué tan precisa es la sincronización de los motores?

La precisión depende de los ajustes del PID y la respuesta de los motores. La reducción de PWM proporcional a la diferencia de vueltas ayuda a mejorar la sincronización.

- ¿Qué mejoras se pueden hacer al código?

Se podría optimizar el ajuste del PID para mejorar la sincronización en diferentes condiciones de carga. Además, se podría implementar un mecanismo de aceleración progresiva para evitar cambios bruscos en la velocidad.

3. CONCLUSIONES

a. El uso del control PID permite un control preciso de la velocidad de los motores, ajustando dinámicamente el PWM para sincronizar el número de vueltas.

b. Aunque los motores puedan tener diferencias en rendimiento, el sistema ajusta automáticamente las velocidades para asegurar que ambos motores completen el número de vueltas ingresadas.

c. Tanto el Monitor Serial como el LCD proporcionan información en tiempo real sobre el número de vueltas de cada motor y los valores de PWM, lo que facilita el monitoreo y el ajuste del sistema.

d. El sistema es fácil de usar gracias al teclado matricial, que permite ingresar el número de vueltas objetivo de manera directa.

e. El código está diseñado de manera modular, lo que permite hacer ajustes en el control PID o el sistema de entrada/salida para adaptarse a diferentes tipos de motores o necesidades.