Practica 1: Práctica de control del motor a CD.

Unidad I: Motores a CD Tema 1.1
Sistemas Embebidos II 18MPEDS0729
Ago-Dic 2025
Centro de Enseñanza Tecnica Industrial Plantel Colomos
Tgo. en Desarrollo de Software
Academia: Sistemas Digitales
Profesor: Antonio Lozano Gonzáles

Emmanuel Buenrostro 22300891 7F1
Emiliano Arzate 22300929 7F1

27 de Agosto de 2025

§1 Objetivo

Variar la velocidad de un motor a CD, utilizando PWM y en cualquier tiempo programado.

§2 Desarrollo de la Práctica

§2.1 Condiciones de la Práctica

Realizar el control de un motor a CD. Deberá acelerar desde cero hasta la velocidad máxima en un tiempo estipulado. acompañado del reporte respectivo. Utilizar el teclado para pedir el tiempo y, la pantalla para mostrar las preguntas necesarias y, el porcentaje de aceleración del motor.
También deberá tener un instrumento o sensor que muestre las revoluciones por minuto del motor. Que se verán en la pantalla.

§2.2 Algoritmo o Diagrama de Flujo

- En la inicialización se configura la pantalla LCD, el teclado matricial y los pines del motor y sensor. Muestra una pantalla inicial pidiendo el tiempo de aceleración.
- Ingreso de tiempo:

  El usuario ingresa el tiempo (en segundos) usando el teclado. Si presiona '*', inicia el proceso de aceleración.
- Aceleración del motor:

  El motor aumenta su velocidad gradualmente desde 0 hasta el máximo durante el tiempo ingresado. En cada paso, actualiza la potencia del motor (con PWM) y calcula las RPM usando un sensor y una rutina de interrupción. Muestra el porcentaje de aceleración y las RPM en la pantalla LCD.
- Velocidad máxima:

  Mantiene el motor a velocidad máxima por 5 segundos, mostrando las RPM.
- Finalización:

  Apaga el motor, muestra un mensaje de finalización y vuelve a la pantalla inicial.

§2.3 Código C

```
#include <LiquidCrystal.h>
#include <Keypad.h>

#define MAX_RPM 60
#define PULSOS_POR_VUELTA 1
#define TIMEOUT_US 2000000UL
#define ALPHA_NUM 3
#define ALPHA_DEN 10

const int LCD_RS = 22;
const int LCD_E = 23;
const int LCD_D4 = 24;
const int LCD_D5 = 25;
const int LCD_D6 = 26;
const int LCD_D7 = 27;
```

```
16 LiquidCrystal lcd(LCD_RS, LCD_E, LCD_D4, LCD_D5, LCD_D6, LCD_D7);
17
18 const byte ROWS = 4;
19 const byte COLS = 4;
20 char keys[ROWS][COLS] = {
21   {'1','2','3','/'},
22   {'4','5','6','-'},
23   {'7','8','9','+'},
24   {'C','0','=','*'}
25 };
26 byte rowPins[ROWS] = {31, 33, 35, 37};
27 byte colPins[COLS] = {30, 32, 34, 36};
28 Keypad customKeypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS,
       COLS);
29
30 const int ENA = 9;
31 const int IN1 = 8;
32 const int IN2 = 7;
33 const int VELOCIDAD_MINIMA_ARRANQUE = 70;
34
35 const int SENSOR_PIN = 2;
36 volatile unsigned long ultimoPulso_us = 0;
37 volatile unsigned long intervalo_us = 0;
38
39 int rpm_disp = 0;
40 long rpm_f = 0;
41 String tiempoString = "";
42 bool valorIngresado = false;
43
44 void calcularIntervalo() {
45   unsigned long t = micros();
46   unsigned long dt = t - ultimoPulso_us;
47   ultimoPulso_us = t;
48   if (dt == 0) return;
49   unsigned long rpmCalc = (60000000UL / dt) / PULSOS_POR_VUELTA;
50   if (rpmCalc <= MAX_RPM) intervalo_us = dt;
51 }
52
53 void setup() {
54   lcd.begin(16, 2);
55   pinMode(ENA, OUTPUT);
56   pinMode(IN1, OUTPUT);
57   pinMode(IN2, OUTPUT);
58   pinMode(SENSOR_PIN, INPUT_PULLUP);
59   attachInterrupt(digitalPinToInterrupt(SENSOR_PIN), calcularIntervalo,
       RISING);
60   digitalWrite(IN1, LOW);
61   digitalWrite(IN2, LOW);
62   analogWrite(ENA, 0);
63   mostrarPantallaInicial();
64 }
65
66 void loop() {
67   if (!valorIngresado) {
68     char key = customKeypad.getKey();
69     if (key) {
70       if (key >= '0' && key <= '9') {
```

```
71          tiempoString += key;
72          lcd.print(key);
73        } else if (key == 'C') {
74          mostrarPantallaInicial();
75        } else if (key == '*') {
76          if (tiempoString.length() > 0) {
77            valorIngresado = true;
78            iniciarAceleracion();
79          }
80        }
81      }
82    }
83  }
84
85  void mostrarPantallaInicial() {
86    lcd.clear();
87    lcd.print("Tiempo (seg):");
88    lcd.setCursor(0, 1);
89    lcd.print("RPM: 0 ");
90    lcd.setCursor(14, 0);
91    tiempoString = "";
92  }
93
94  static inline void printLineaEstado(int porcentaje, int rpmVal) {
95    if (rpmVal > MAX_RPM) rpmVal = MAX_RPM;
96    lcd.setCursor(0, 1);
97    lcd.print("%:");
98    if (porcentaje < 10) lcd.print(' ');
99    lcd.print(porcentaje);
100   lcd.print(" RPM:");
101   char buf[6];
102   snprintf(buf, sizeof(buf), "%4d", rpmVal);
103   lcd.print(buf);
104   lcd.print(" ");
105 }
106
107 static inline void actualizarRPM() {
108   unsigned long intervaloCopia, ultimoPulsoCopia;
109   noInterrupts();
110   intervaloCopia = intervalo_us;
111   ultimoPulsoCopia = ultimoPulso_us;
112   interrupts();
113
114   if (intervaloCopia > 0) {
115     unsigned long rpmCalc = (60000000UL / intervaloCopia) /
          PULSOS_POR_VUELTA;
116     if (rpmCalc > MAX_RPM) rpmCalc = MAX_RPM;
117     rpm_f = (ALPHA_NUM * (long)rpmCalc + (ALPHA_DEN - ALPHA_NUM) * rpm_f)
          / ALPHA_DEN;
118     rpm_disp = (int)rpm_f;
119   }
120
121   if ((micros() - ultimoPulsoCopia) > TIMEOUT_US) {
122     rpm_f = 0;
123     rpm_disp = 0;
124     noInterrupts();
125     intervalo_us = 0;
```

```
126     interrupts();
127   }
128 }
129
130 void iniciarAceleracion() {
131   long tiempoSegundos = tiempoString.toInt();
132   if (tiempoSegundos > 300) tiempoSegundos = 300;
133   if (tiempoSegundos <= 0) { valorIngresado = false;
          mostrarPantallaInicial(); return; }
134
135   lcd.clear();
136   lcd.print("Acelerando...");
137   digitalWrite(IN1, HIGH);
138   digitalWrite(IN2, LOW);
139
140   long tiempoMilisegundos = tiempoSegundos * 1000L;
141   int delayPorPaso = tiempoMilisegundos / 255;
142   if (delayPorPaso <= 0) delayPorPaso = 1;
143
144   noInterrupts();
145   ultimoPulso_us = micros();
146   intervalo_us = 0;
147   interrupts();
148   rpm_f = 0;
149   rpm_disp = 0;
150
151   for (int velocidad = 0; velocidad <= 255; velocidad++) {
152     int potenciaReal = map(velocidad, 0, 255, 0, 255);
153     if (potenciaReal > 0 && potenciaReal < VELOCIDAD_MINIMA_ARRANQUE) {
154       potenciaReal = VELOCIDAD_MINIMA_ARRANQUE;
155     }
156     analogWrite(ENA, potenciaReal);
157
158     actualizarRPM();
159
160     int porcentaje = (velocidad * 100) / 255;
161     printLineaEstado(porcentaje, rpm_disp);
162
163     delay(delayPorPaso);
164   }
165
166   lcd.setCursor(0, 0);
167   lcd.print("Vel. Maxima! ");
168   unsigned long t0 = millis();
169   while (millis() - t0 < 5000UL) {
170     actualizarRPM();
171     printLineaEstado(100, rpm_disp);
172     delay(100);
173   }
174
175   analogWrite(ENA, 0);
176   lcd.clear();
177   lcd.print("Proceso");
178   lcd.setCursor(0, 1);
179   lcd.print("Finalizado ");
180   delay(2000);
181
```

```
182   valorIngresado = false;
183   mostrarPantallaInicial();
184 }
```

§3 Observaciones y Conclusiones

- Nuestro motor al ser más de potenica no tiene tantas RPM, lo cuál creo facilito la práctica.
- Está complicado mantener que el motor no se pare con el sensor que mide RPM.
- Está practica a sido de las más complicadas, ya que implica unir el puente H, el motor, y el sensor que mide RPM, todo esto además de un algoritmo algo más complejo debido al cambio gradual de velocidad, pero al ya haber usado el motor y puente H en nuestro proyecto de Embebidos 1 se nos facilito bastante.