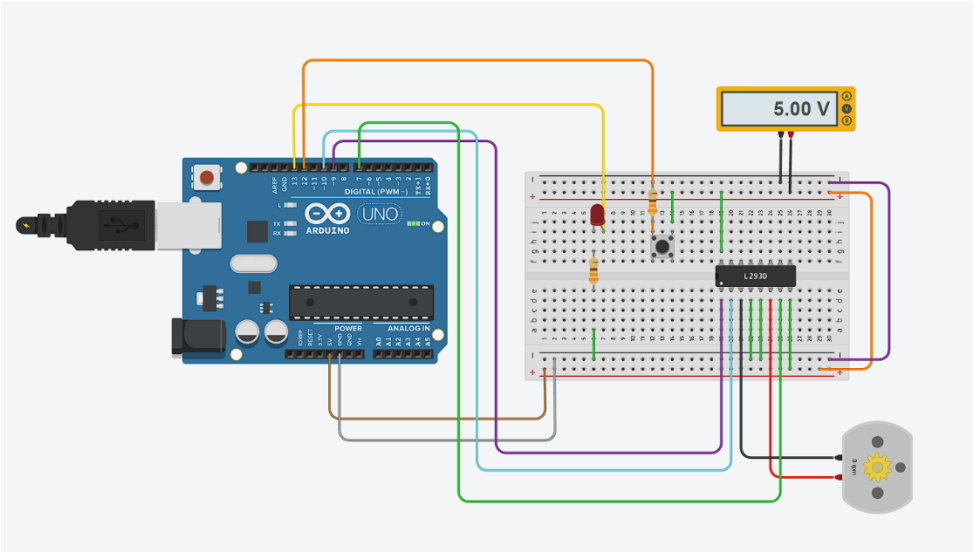
Brayan Camilo Cristancho Franco

Último Taller 2023-1 microprocesadores

1. **Hacer un montaje libre de un Arduino/Raspberry en un emulador, que muestre su funcionamiento, por ejemplo: encender un motor a voluntad, o varios LED´s o la interfaz con otro dispositivo.**

   Para este primer punto se tiene como objetivo realizar un montaje en Arduino donde se tenga un circuito que realice lo siguiente:

   El objetivo es controlar un motor de corriente continua (DC) mediante un botón. Cuando el motor esté apagado, se encenderá un LED rojo que servirá como indicador visual para indicar que el motor está detenido.



   Para este circuito se usó los siguientes materiales:

   - **Arduino UNO:** Es el microcontrolador que va a realizar las tareas con base al código programado.
   - **Protoboard:** Donde se implementará el circuito planteado.
   - **Led:** Es el componente que se busca encender al detener el motor.
   - **Resistencias:** Limitaran la corriente tanto del botón como del led y son de 330 ohmios.
   - **Circuito integrado L293D:** Es el que controla el motor.
   - **Motor DC:** Es el componente que se busca encender o detener al presionar el botón.
   - **Botón:** Es el que permite el encendido tanto del led como el motor.

DESCRIPCIONES DEL CÓDIGO

En este apartado se declaran las variables que se utilizarán en el programa. Aquí se definen los pines utilizados para el botón, los pines de control del motor y el pin del LED. También se inicializan las variables de estado del motor, estado del LED, velocidad del motor y variables auxiliares para el botón.

```
int buttonPin = 12;
int motorPin1 = 9;   // Conectado al pin IN1 del puente H
int motorPin2 = 7;   // Conectado al pin IN2 del puente H
int ledPin = 13;
int motorState = LOW;  // Estado inicial del motor: detenido
int ledState = HIGH;   // Estado inicial del LED: encendido
int buttonState;
int lastButtonState = HIGH;
int motorSpeed = 255;  // Velocidad del motor (valores entre 0 y 255)
```

En este apartado se declara una función setup que se ejecuta una vez al inicio del programa. En este bloque de código se configuran los modos de los pines utilizados. El pin del botón se configura como entrada (INPUT), mientras que los pines del motor y el LED se configuran como salidas (OUTPUT).

```
void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(ledPin, OUTPUT);
}
```

En este ultimo apartado se declara una función loop que como su nombre lo dice, es un ciclo que se ejecuta indefinidamente, en este bloque, se lee el estado actual del botón utilizando la función digitalRead() ya que se usaron los pines digitales del arduino. Si el estado del botón es diferente al estado anterior (lastButtonState), se entra en un bloque condicional.

Dentro del bloque condicional, Básicamente lo que hace es que si el botón está en estado alto (HIGH), se invierte el estado del motor y el estado del LED utilizando el operador de negación (!). Esto significa que si el motor estaba detenido, se activará, y si el motor estaba activo, se detendrá. El estado del LED también se invertirá, lo que significa que se apagará si estaba encendido y se encenderá si estaba apagado.

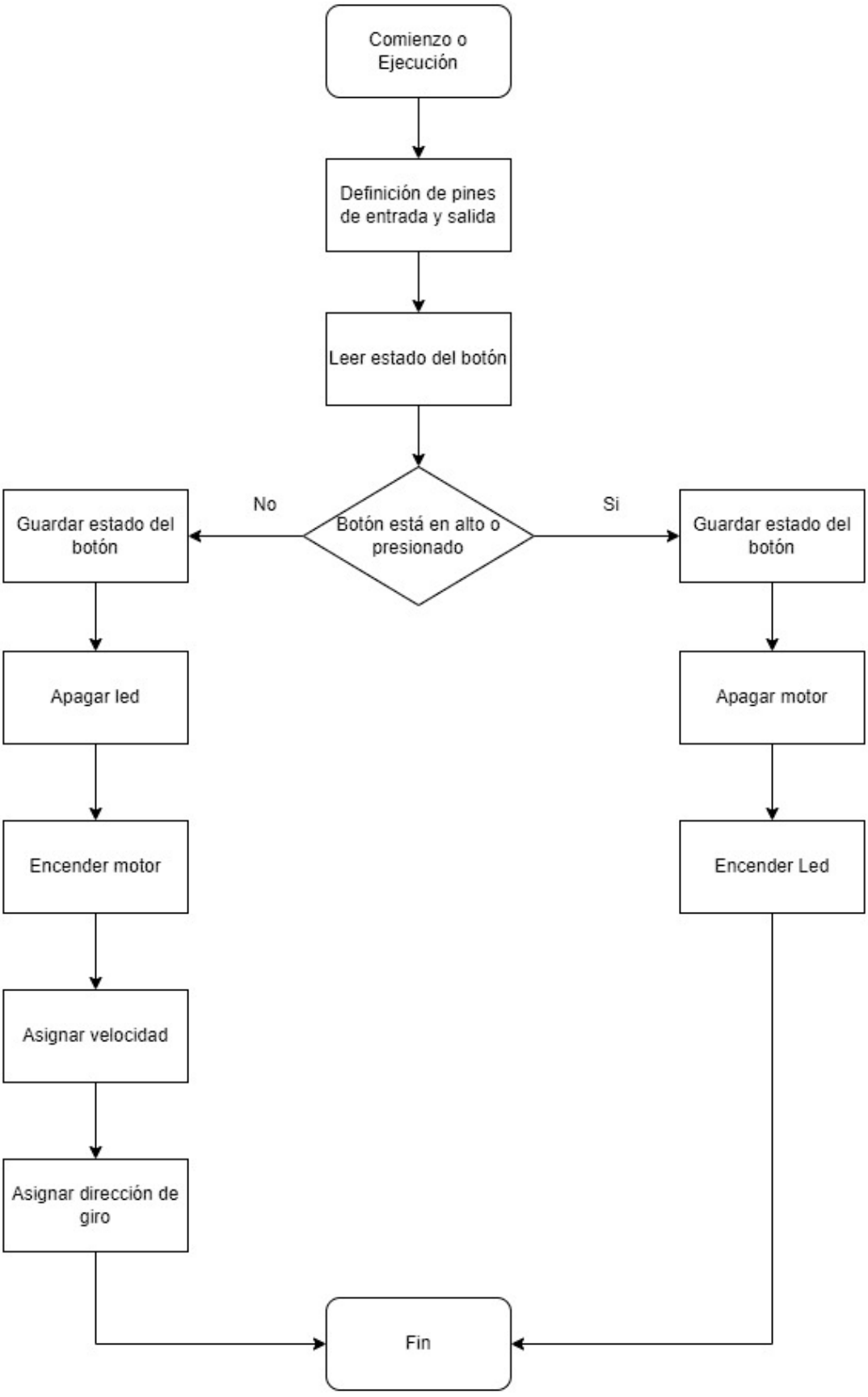
Luego se agrega una pequeña pausa (delay(50)) para evitar fluctuaciones en la lectura del botón (debounce).

Finalmente, se actualiza el estado del LED utilizando digitalWrite() y se guarda el estado actual del botón en la variable lastButtonState para su comparación en la siguiente iteración del bucle.

```
void loop() {
  buttonState = digitalRead(buttonPin);

  if (buttonState != lastButtonState) {
    if (buttonState == HIGH) {
      motorState = !motorState; // Cambia el estado del motor
      ledState = !ledState;     // Cambia el estado del LED
    }
    delay(50);  // Debounce - espera para evitar fluctuaciones
  }

  if (motorState == HIGH) {
    // Gira el motor a la derecha
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, LOW);
    analogWrite(motorPin2, motorSpeed);  // Establece la velocidad del motor
  } else {
    // Detiene el motor
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, LOW);
    analogWrite(motorPin2, 0);  // Establece la velocidad del motor en 0
  }

  digitalWrite(ledPin, ledState);

  lastButtonState = buttonState;
}
```

DIAGRAMA DE FLUJO

Este diagrama muestra cómo fluye el programa desde la configuración inicial de los pines hasta el bucle principal. En el bucle principal, se leen el estado del botón, se verifica si ha habido un cambio, se controla el estado del motor y el LED en función de ese cambio y se actualizan los pines correspondientes. Este proceso se repite continuamente en el bucle principal hasta que se detiene el programa.


Comienzo o Ejecución
Definición de pines de entrada y salida
Leer estado del botón
Botón está en alto o presionado
No
Si
Guardar estado del botón
Guardar estado del botón
Apagar led
Apagar motor
Encender motor
Encender Led
Asignar velocidad
Asignar dirección de giro
Fin

2. **Descargar emu8086 y ejecutar sobre éste alguno de los códigos finales del laboratorio del LMC.**

DESCRIPCIÓN DEL CÓDIGO

Para este punto se va a realizar el código para calcular el factorial de un número.

De manera general el código funciona así:

El programa comienza estableciendo el número inicial en 8 y lo guarda en los registros AX y CX. Luego, utiliza un bucle para calcular el factorial del número, multiplicando repetidamente el acumulador (AX) por el contador (CX).

Después de calcular el factorial, se mueve el resultado al registro BX y se utiliza un bucle para extraer los dígitos del número factorial y almacenarlos en la pila.

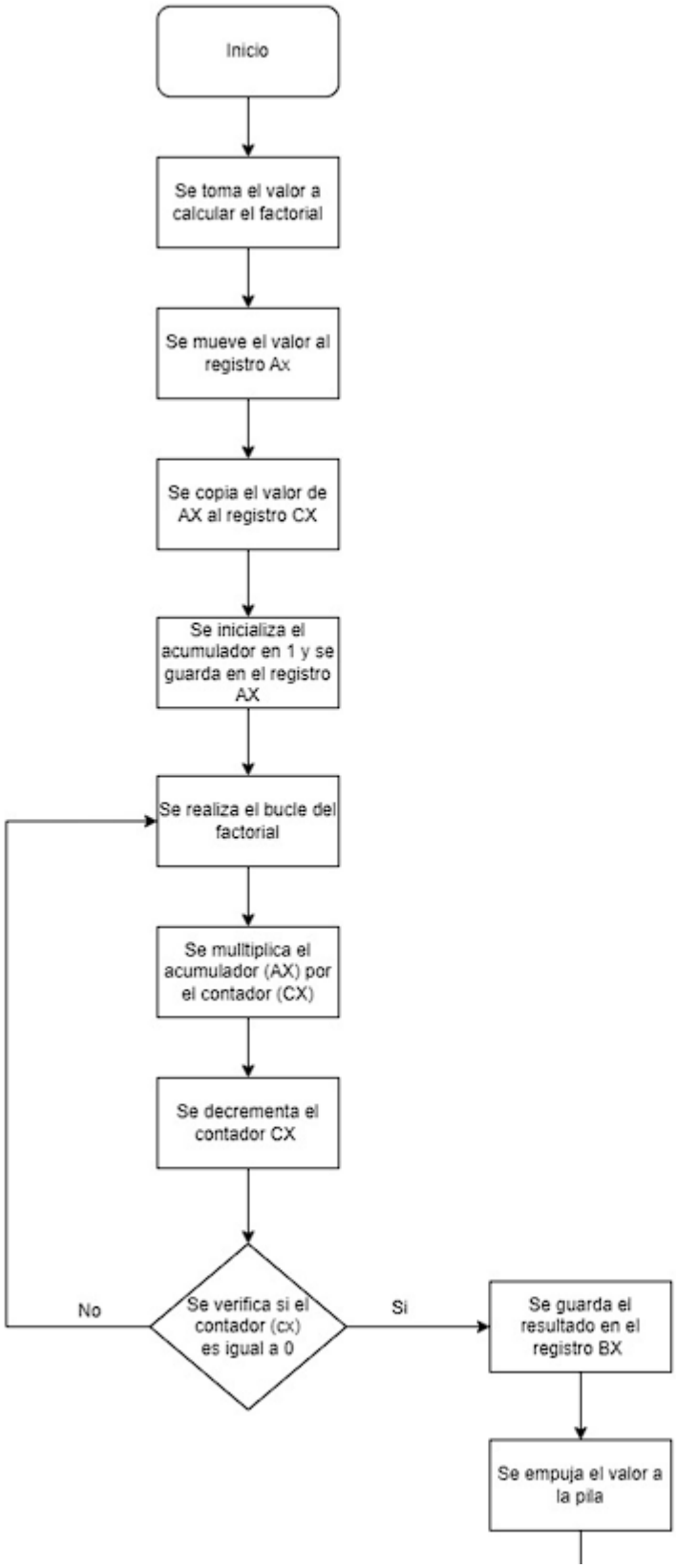
A continuación, otro bucle extrae los dígitos de la pila y los imprime en la pantalla utilizando la interrupción del DOS y traduciendo su valor al respectivo código Ascii.

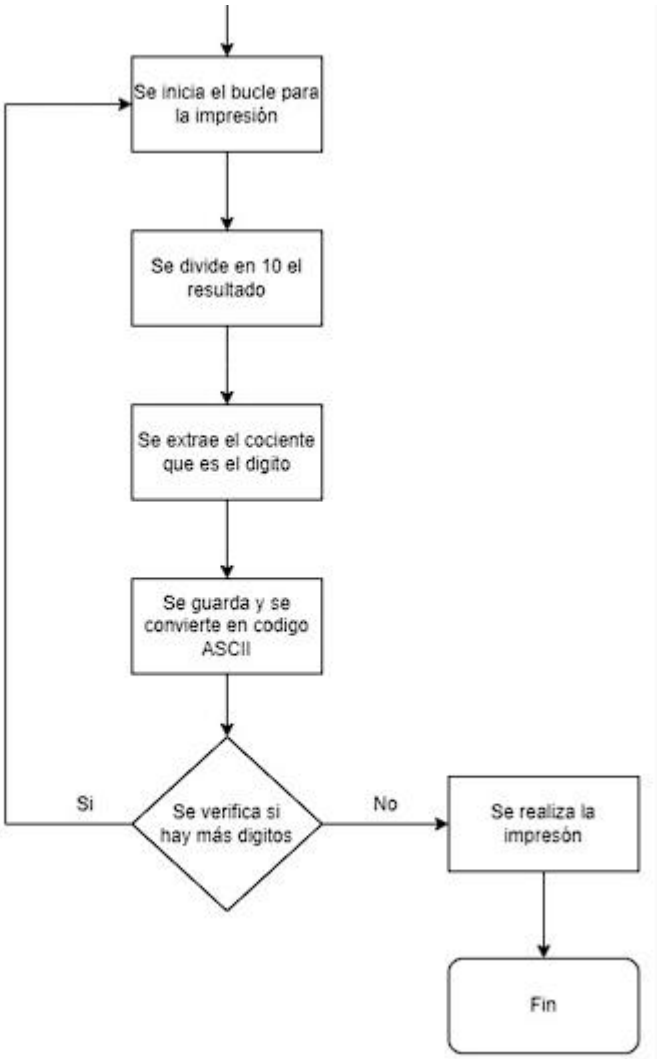
Finalmente, el programa espera la entrada de una tecla antes de finalizar.

```
org 100h  ; Direcci¾n de origen del programa

mov ax, 5  ; Mueve el valor 8 al registro AX
mov cx, ax  ; Copia el valor de AX al registro CX (contador)

mov ax, 1  ; Inicializa el acumulador en 1

factorial_loop:
    mul cx  ; Multiplica el acumulador por el contador
    loop factorial_loop  ; Repite el bucle hasta que CX llegue a cero (decrementa CX automßticamente)

mov bx, ax  ; Guarda el resultado del cßlculo del factorial en el registro BX (para imprimirlo)

push ax  ; Empuja el valor de AX a la pila
mov cx, 0  ; Inicializa el contador CX en 0

print_loop:
    mov dx, 0  ; Reinicia DX a 0
    mov bx, 10  ; Establece BX en 10 (para la divisi¾n)

    div bx  ; Divide AX por BX, el cociente se guarda en AH y el resto en AL
    add dl, '0'  ; Convierte el resto en un dÝgito ASCII y lo agrega a DL
    push dx  ; Empuja el dÝgito convertido a la pila
    inc cx  ; Incrementa el contador CX

    cmp ax, 0  ; Compara AX con 0
    jne print_loop  ; Salta a print_loop si AX no es igual a 0

print_digits:
    pop dx  ; Saca un dÝgito de la pila a DX
    mov ah, 02h  ; Establece la funci¾n 02h de la interrupci¾n 21h (imprimir carßcter)
    int 21h  ; Llama a la interrupci¾n del DOS para imprimir el carßcter en DX

    loop print_digits  ; Repite el bucle hasta que CX llegue a cero (decrementa CX automßticamente)

mov ah, 0  ; Establece AH en 0
int 16h  ; Espera la entrada de una tecla antes de finalizar el programa
ret
```

DIAGRAMA DE FLUJO

En este diagrama de flujo se describe brevemente el paso a paso del código para realizar el factorial en el emulador emu8086.


Inicio
Se toma el valor a calcular el factorial
Se mueve el valor al registro Ax
Se copia el valor de AX al registro CX
Se inicializa el acumulador en 1 y se guarda en el registro AX
Se realiza el bucle del factorial
Se multiplica el acumulador (AX) por el contador (CX)
Se decrementa el contador CX
Se verifica si el contador (cx) es igual a 0
No
Si
Se guarda el resultado en el registro BX
Se empuja el valor a la pila



ENLACES VIDEOS

Video Primer punto: https://youtu.be/z7LrdrETjxI

Video Segundo punto: https://youtu.be/Zbho1EF46AY