Design and Implementation of an AVR Microcontroller-Based Electronic Control System with Sensor Integration and Sequential State Machine

Demetrio Manuel Roa Perdomo

Notas del autor

Demetrio Manuel Roa Perdomo, Facultad de Ingeniería Mecánica y Eléctrica, Universidad Autónoma de Nuevo León

Esta investigación ha sido financiada por el propio alumno

La correspondencia relacionada con esta investigación debe ser dirigida a Demetrio Roa Universidad Autónoma de Nuevo León, Pedro de Alba S/N, Niños Héroes, Ciudad Universitaria, San Nicolás de los Garza, N.L.

Contacto: demetrio.roap@uanl.edu.mx

Design and Implementation of an AVR Microcontroller-Based Electronic Control System with Sensor Integration and Sequential State Machine

Introducción

¿En dónde se utilizan las máquinas de estado?

Las máquinas de estado, también conocidas como autómatas finitos, son una forma de modelar sistemas que tienen un número limitado de estados y que pueden cambiar de estado en respuesta a ciertos eventos. Se utilizan en una amplia variedad de aplicaciones, tanto en software como en hardware.

En sistemas de control, las máquinas de estado se utilizan para modelar y controlar sistemas como semáforos, sistemas de control de ascensores y sistemas de control de procesos industriales. En estos casos, los estados representan las diferentes condiciones o modos de operación del sistema, y las transiciones entre estados se producen en respuesta a ciertos eventos o condiciones. Por ejemplo, un semáforo puede tener estados como “verde”, “amarillo” y “rojo”, y las transiciones entre estos estados se producen en función de un temporizador.

En el diseño de circuitos digitales, las máquinas de estado son fundamentales. Los circuitos digitales, como los procesadores de computadoras, los controladores de dispositivos y otros sistemas embebidos, a menudo se diseñan como máquinas de estado. Los estados pueden representar las diferentes etapas de una operación, como la lectura de datos, el procesamiento de datos y la escritura de datos, y las transiciones entre estados se producen en respuesta a las señales de reloj y otras señales de control.

Los protocolos de red, como TCP, HTTP y FTP, también se pueden modelar como máquinas de estado. Los estados pueden representar las diferentes etapas de una conexión de red, como la escucha de conexiones entrantes, el establecimiento de una conexión, la transferencia de datos y el cierre de una conexión, y las transiciones entre estados se producen en respuesta a los paquetes de red recibidos y otros eventos.

En el desarrollo de videojuegos, las máquinas de estado se utilizan para controlar la lógica del juego y el comportamiento de los personajes del juego. Los estados pueden representar las diferentes acciones o actividades de un personaje, como estar parado, correr, saltar y atacar, y las transiciones entre estados se producen en respuesta a las entradas del jugador y otros eventos del juego.

En inteligencia artificial, las máquinas de estado se utilizan para modelar y simular comportamientos complejos. Por ejemplo, en un sistema de inteligencia artificial para un juego de ajedrez, se podría utilizar una máquina de estado para representar y controlar las diferentes fases del juego, como la apertura, el medio juego y el final.

Estos son solo algunos ejemplos de cómo se utilizan las máquinas de estado. Son una herramienta poderosa y versátil que se utiliza en muchas áreas de la informática, la ingeniería

y la tecnología. Su uso permite a los diseñadores y desarrolladores modelar de manera efectiva sistemas complejos y manejar una amplia variedad de condiciones y eventos.

Marco Teórico

Arduino UNO: Es una plataforma de creación de electrónica de código abierto, basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso.

Dip Switch: Es un pequeño interruptor perteneciente a un grupo de interruptores conectados a placas de sonido antiguas, placas base, impresoras, módems, y otros dispositivos electrónicos y de computación. Los interruptores DIP fueron muy comunes en antiguas tarjetas de expansión tipo ISA y han sido utilizados comúnmente para seleccionar el IRQ y configurar los otros recursos del sistema para la tarjeta.

Displays de 7 Segmentos Cátodo Común: Es un dispositivo electrónico de visualización utilizado para representar números decimales. En su forma más común el display de 7 segmentos se presenta como un arreglo de diodos emisores de luz (LED) en el cual pueden controlarse individualmente los segmentos para representar el glifo deseado.

Resistencia 1/4W 330 Ω: Es una resistencia de 330 Ohm con tolerancia de +/- 5%, puede disipar una potencia máxima de 1/4 Watt. Ideal para limitar la corriente de Leds en sistemas de 5V y 3.3V.

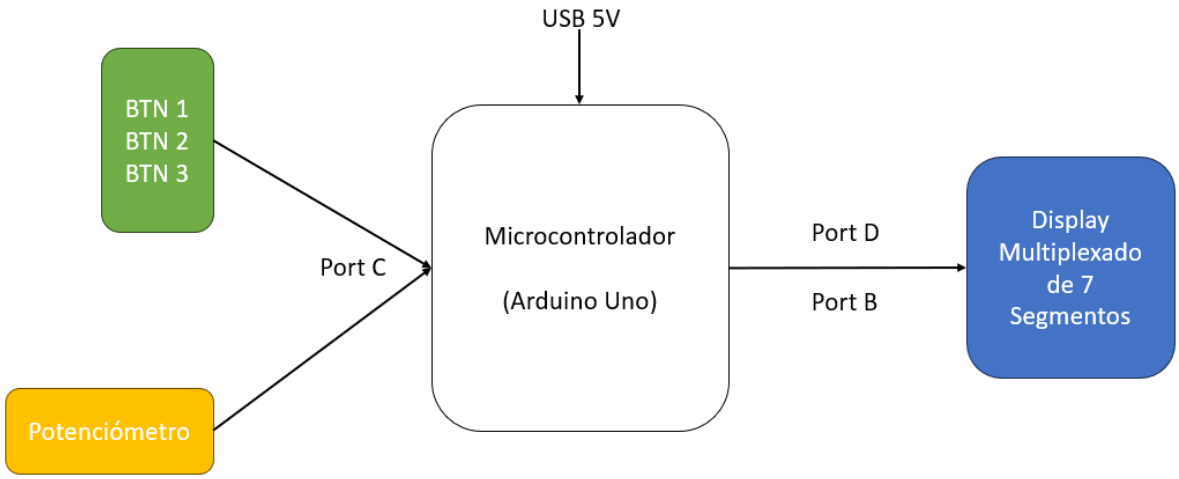
Resistencia 1/4W 1000 Ω: Es una resistencia de 1 kΩ y soporta una potencia máxima de ¼ W.

Potenciometro 250 kΩ: Un potenciómetro de 250 kΩ es un resistor eléctrico con un valor de resistencia variable y generalmente ajustable manualmente.

Transistores NPN, 2N2222: El 2N2222 es un famosísimo transistor bipolar NPN de uso general construido inicialmente por la Motorola en los años '60 y todavía muy usado por su gran versatilidad. Los 2N2222 tienen una capacidad de corriente de colector significativa, hasta 800 mA, característica que les permite de ser usados en muchas aplicaciones de control de media potencia como por ejemplo drivers para pequeños motores, relés y también tiras de leds cortas.

Alambre: Un alambre es un filamento flexible de metal. El alambre se suele formar trefilando el metal a través de un orificio en un troquel o placa de trefilado, de los diferentes metales de acuerdo con la propiedad de ductilidad que poseen los mismos.

Imagen del diagrama de bloques



Lista de Materiales

Qty	Descripción	Costo/Pieza	Costo
1	Arduino UNO	$490.00	$490.00
1	Dip Switch	$5.00	$5.00
4	Displays de 7 Segmentos Cátodo Común	$20.00	$80.00
8	Resistencia 1/4W 330 Ω	$0.50	$4.00
4	Resistencia 1/4W 1000 Ω	$0.50	$2.00
1	Potenciometro 250 kΩ	$20.00	$20.00
4	Transistores NPN, 2N2222	$20.00	$80.00
-	Alambre	$20.00	$20.00

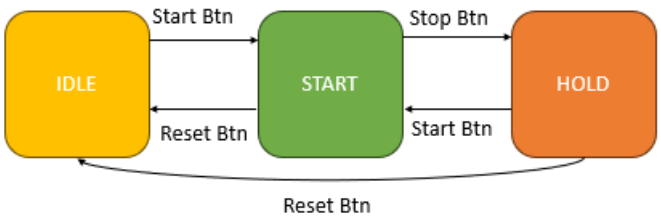
Tabla de Estado Siguiente

Estado Actual	Botón de Inicio	Botón de Retención	Botón de Reset	Estado Siguiente
IDLE	No presionado	X	X	IDLE
IDLE	Presionado	X	X	START
START	X	No presionado	X	START
START	X	Presionado	X	HOLD
HOLD	No presionado	X	X	HOLD
HOLD	Presionado	X	X	START
X	X	X	Presionado	IDLE

Donde:

- “X” significa “No importa” o “Cualquier valor”.
- “No presionado” significa que el botón no está siendo presionado.
- “Presionado” significa que el botón está siendo presionado.

Diagrama de Transición de Estados Finitos (FSM)



Donde:

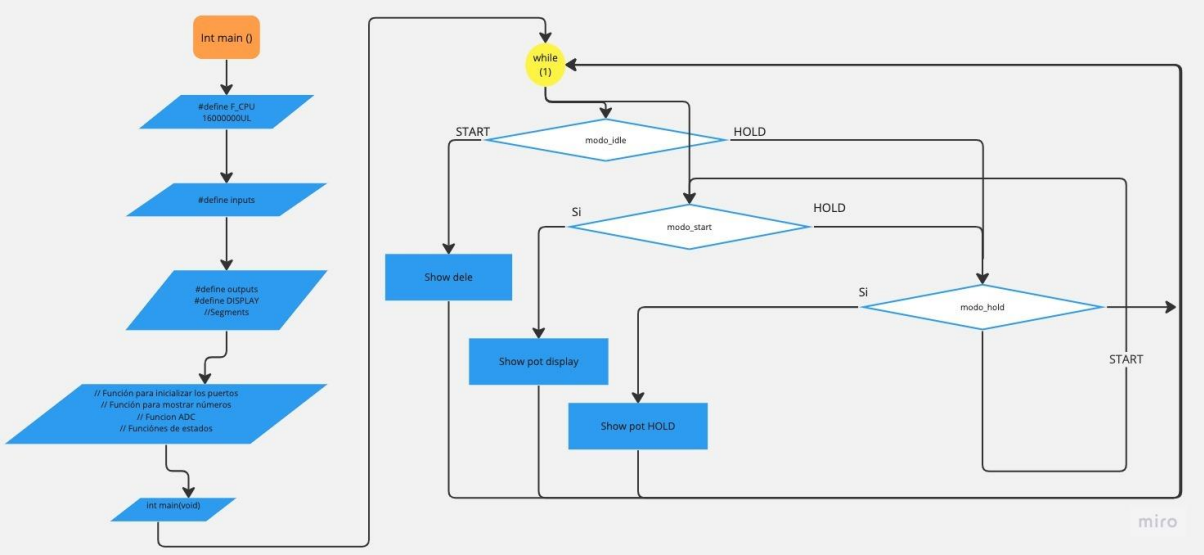
- **IDLE** es el estado inicial en el que el sistema se encuentra en reposo.
- **START** es el estado en el que el sistema ha comenzado a funcionar después de que se presiona el botón de inicio.
- **HOLD** es el estado en el que el sistema se mantiene después de que se presiona el botón de retención/stop.

Las transiciones entre los estados son las siguientes:

- **IDLE a START**: Cuando se presiona el botón de inicio, el sistema cambia del estado IDLE al estado START.

- **START a HOLD**: Cuando se presiona el botón de retención, el sistema cambia del estado START al estado HOLD.
- **HOLD a START**: Cuando se presiona el botón de inicio, el sistema cambia del estado HOLD al estado START.
- **START a IDLE**: Cuando se presiona el botón de reset, el sistema cambia del estado START al estado IDLE.
- **HOLD a IDLE**: Cuando se presiona el botón de reset, el sistema cambia del estado HOLD al estado IDLE.

Diagrama de Flujo en Lenguaje de Alto Nivel



Lista de Pines Utilizados con su Función y Nombre

Pines del display:

Segmento A: Pin conectado al segmento A del display. (PORTD2)

Segmento B: Pin conectado al segmento B del display. (PORTD3)

Segmento C: Pin conectado al segmento C del display. (PORTD4)

Segmento D: Pin conectado al segmento D del display. (PORTD5)

Segmento E: Pin conectado al segmento E del display. (PORTD6)

Segmento F: Pin conectado al segmento F del display. (PORTD7)

Segmento G: Pin conectado al segmento G del display. (PORTB0)

Pines comunes del display:

Común 1: Pin conectado al pin común del primer dígito del display. (PORTC0)

Común 2: Pin conectado al pin común del segundo dígito del display. (PORTC1)

Común 3: Pin conectado al pin común del tercer dígito del display. (PORTC2)

Común 4: Pin conectado al pin común del cuarto dígito del display. (PORTC3)

Pines de los botones:

Start Button Pin: Pin conectado al botón de inicio. (A3)

Hold Button Pin: Pin conectado al botón de retención. (A4)

Reset Button Pin: Pin conectado al botón de reset. (A5)

Pin del potenciómetro:

Potenciómetro Pin: Pin conectado al potenciómetro. (A0)

Código en lenguaje C

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <avr/interrupt.h>
#include <util/delay.h>
#define TIEMPO 0.05
//ENTRADAS
#define botonesDDR DDRD
#define botonesPORT PORTD
#define botonesPIN PIND
#define boton0 PIND0
#define boton1 PIND1
#define boton2 PIND2
//Macro
#define boton0Read (!(botonesPIN & (1 << boton0))) //START
#define boton1Read (!(botonesPIN & (1 << boton1))) //RESET
#define boton2Read (!(botonesPIN & (1 << boton2))) //HOLD
//Variables
char letras[4];
char numbers[4];
volatile int contador = 0;
volatile int contador1s = 0;
volatile int valueADC = 0;
volatile int lastScaledValue = 0;
#define DISPLAY_DDRX_SEG_AF DDRB
#define DISPLAY_PORTX_SEG_AF PORTB
#define DISPLAY_DDRX_SEG_G DDRC
#define DISPLAY_PORTX_SEG_G PORTC
// SEG_A
#define DISPLAY_SEG_A PORTB0
#define DISPLAY_SEG_A_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_A)
#define DISPLAY_SEG_A_OFF DISPLAY_PORTX_SEG_AF &= ~(1 << DISPLAY_SEG_A)
// SEG_B
#define DISPLAY_SEG_B PORTB1
#define DISPLAY_SEG_B_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_B)
#define DISPLAY_SEG_B_OFF DISPLAY_PORTX_SEG_AF &= ~(1 << DISPLAY_SEG_B)
// SEG_C
#define DISPLAY_SEG_C PORTB2
#define DISPLAY_SEG_C_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_C)
#define DISPLAY_SEG_C_OFF DISPLAY_PORTX_SEG_AF &= ~(1 << DISPLAY_SEG_C)
// SEG_D
#define DISPLAY_SEG_D PORTB3
#define DISPLAY_SEG_D_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_D)
#define DISPLAY_SEG_D_OFF DISPLAY_PORTX_SEG_AF &= ~(1 << DISPLAY_SEG_D)
// SEG_E
#define DISPLAY_SEG_E PORTB4
#define DISPLAY_SEG_E_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_E)
#define DISPLAY_SEG_E_OFF DISPLAY_PORTX_SEG_AF &= ~(1 << DISPLAY_SEG_E)
// SEG_F
#define DISPLAY_SEG_F PORTB5
#define DISPLAY_SEG_F_ON DISPLAY_PORTX_SEG_AF |= (1 << DISPLAY_SEG_F)
#define DISPLAY_SEG_F_OFF DISPLAY_PORTX_SEG_AF &= ~(1 << DISPLAY_SEG_F)
```

```
// SEG_G
#define DISPLAY_SEG_G PORTC0
#define DISPLAY_SEG_G_ON DISPLAY_PORTX_SEG_G |= (1 << DISPLAY_SEG_G)
#define DISPLAY_SEG_G_OFF DISPLAY_PORTX_SEG_G &= ~(1 << DISPLAY_SEG_G)
//barrido
#define BARRIDO_DDR DDRD
#define BARRIDO_PORT PORTD
#define BARRIDO_Millares PORTD4
#define BARRIDO_Centenas PORTD5
#define BARRIDO_Decenas PORTD6
#define BARRIDO_Unidades PORTD7
//Declaracion
void init_display(void);
void display_show_number(uint8_t numero);
void display_show_letter(char letra);
void init_barrido(void);
void barrido_show_word(char letra3, char letra2, char letra1, char letra0);
void barrido_show_numbers(int number);
void init_timer0(void);
void init_ports(void);
void init_ADC(void);
void init_INT0(void);

int map(int x, int in_min, int in_max, int out_min, int out_max) {
   return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

//Estados
enum State {IDLE, START, HOLD};
State state = IDLE;

void IDLE(void){
  // Lógica para el estado IDLE
          barrido_show_word('C', 'I', 'A', 'O');
          _delay_ms(3000);  // Esperar 3 segundos
          barrido_show_word('D', 'E', 'L', 'E');

}

void START(void){
  // Lógica para el estado START
        int scaledValue = map(valueADC, 0, 255, 0, 100);
         lastScaledValue = scaledValue;  // Store the current scaled value

         // Mostrar el valor escalado en el display
         barrido_show_numbers(lastScaledValue);

}

void HOLD(void){
  // Lógica para el estado IDLE
          barrido_show_word('S', 'T', 'O', 'P');
          barrido_show_numbers(lastScaledValue);  // Mostrar el último valor escalado en el display


             // Esperar un breve periodo para evitar cambios rápidos
             _delay_ms(50);
}

void init_ports(void) {
botonesDDR &= ~(1 << boton0);
botonesDDR &= ~(1 << boton1);
botonesDDR &= ~(1 << boton2);

}
void init_display(void)
{
DISPLAY_DDRX_SEG_AF |= (1<<DISPLAY_SEG_A);
DISPLAY_DDRX_SEG_AF |= (1<<DISPLAY_SEG_B);
DISPLAY_DDRX_SEG_AF |= (1<<DISPLAY_SEG_C);
DISPLAY_DDRX_SEG_AF |= (1<<DISPLAY_SEG_D);
DISPLAY_DDRX_SEG_AF |= (1<<DISPLAY_SEG_E);
DISPLAY_DDRX_SEG_AF |= (1<<DISPLAY_SEG_F);
DISPLAY_DDRX_SEG_G |= (1<<DISPLAY_SEG_G);
}
```

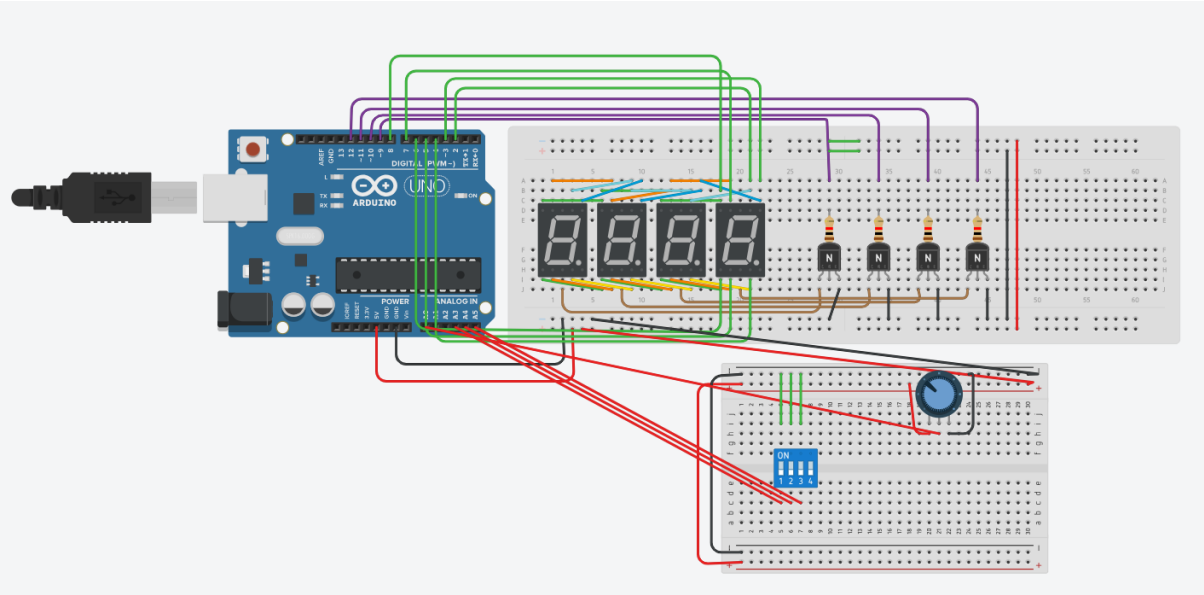
```
void display_show_number(uint8_t numero)
{
switch (numero)
{
case 0:
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_OFF;
break;
case 1:
DISPLAY_SEG_A_OFF;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_OFF;
DISPLAY_SEG_G_OFF;
break;
case 2:
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_OFF;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_OFF;
DISPLAY_SEG_G_ON;
break;
case 3:
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_OFF;
DISPLAY_SEG_G_ON;
break;
case 4:
DISPLAY_SEG_A_OFF;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 5:
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 6:
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 7:
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_OFF;
DISPLAY_SEG_G_OFF;
```

```
break;
case 8:
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 9:
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
default:
DISPLAY_SEG_A_OFF;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_OFF;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_OFF;
DISPLAY_SEG_G_OFF;
break;
}
}
void display_show_letter(char letra)
{
switch (letra)
{
case 'C':
DISPLAY_SEG_A_OFF;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 'I':
DISPLAY_SEG_A_OFF;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_OFF;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_OFF;
break;
case 'A':
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 'D':
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_OFF;
break;
case 'E':
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_OFF;
DISPLAY_SEG_D_ON;
```

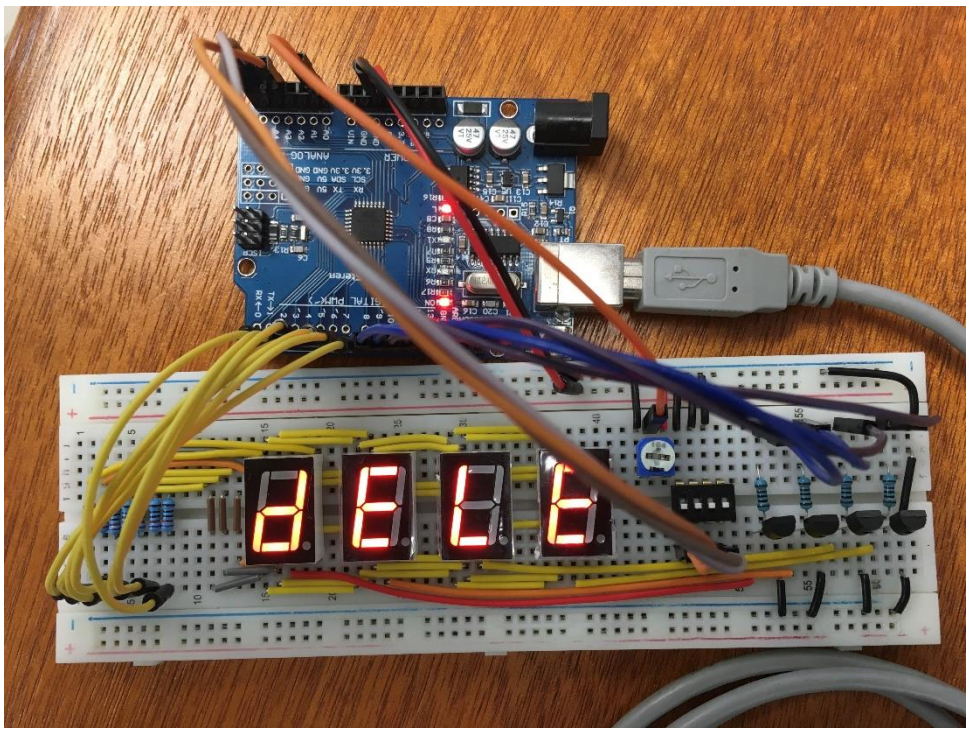
```
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 'L':
DISPLAY_SEG_A_OFF;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_OFF;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_OFF;
DISPLAY_SEG_G_ON;
break;
case 'S':
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_OFF;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 'T':
DISPLAY_SEG_A_OFF;
DISPLAY_SEG_B_OFF;
DISPLAY_SEG_C_OFF;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 'P':
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_OFF;
DISPLAY_SEG_D_OFF;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_ON;
break;
case 'O':
DISPLAY_SEG_A_ON;
DISPLAY_SEG_B_ON;
DISPLAY_SEG_C_ON;
DISPLAY_SEG_D_ON;
DISPLAY_SEG_E_ON;
DISPLAY_SEG_F_ON;
DISPLAY_SEG_G_OFF;
break;
}
}
void init_barrido(void)
{
BARRIDO_DDR |= (1<<BARRIDO_Millares);
BARRIDO_DDR |= (1<<BARRIDO_Centenas);
BARRIDO_DDR |= (1<<BARRIDO_Decenas);
BARRIDO_DDR |= (1<<BARRIDO_Unidades);
}

void barrido_show_word(char letra3, char letra2, char letra1, char letra0)
{
letras[3] = letra3;
letras[2] = letra2;
letras[1] = letra1;
letras[0] = letra0;
for(int i=4, k=3 ;i<8;i++)
{
BARRIDO_PORT=~(1<<i);
display_show_letter(letras[k]);
_delay_ms(TIEMPO);
k--;
}
}
void barrido_show_numbers(int number)
{
```

```
numbers[3] = number/1000;
numbers[2] = (number%1000)/100;
numbers[1] = (number%100)/10;
numbers[0] = number%10;
for(int i=4, e=3 ;i<8;i++)
{
BARRIDO_PORT=~(1<<i);
display_show_number(numbers[e]);
_delay_ms(TIEMPO);
e--;
}
}
void init_INT0(void)
{
//Rising edge
EICRA |= (1<<ISC00);
EICRA |= (1<<ISC01);
//ENABLE
EIMSK |= (1<<INT0);
}
ISR(INT0_vect)
{
contador++;
}
//Timer init
void init_timer0(void){
//CTC
TCCR0A &= ~(1<<WGM00);
TCCR0A |= (1<<WGM01);
TCCR0B &= ~(1<<WGM02);
//Clock select -> Prescaler /8
TCCR0B |= (1<<CS02);
TCCR0B &= ~ (1<<CS01);
TCCR0B |= (1<<CS00);
//TOP
OCR0A = 156;
//Interrupt enable
TIMSK0 |= (1<<OCIE0A);
}
ISR(TIMER0_COMPA_vect) {
   contador++;

   if (contador > 100) {
      // Cap contador at 100
      contador = 0;
   }
 }

ISR(ADC_vect) {
   valueADC = ADCH; // Obtener el valor del ADC (de 0 a 255)
   ADCSRA |= (1 << ADSC); // Iniciar la próxima conversión
}

void init_ADC(void) {
   // Seleccionar ADC2 como fuente de entrada

            ADMUX &= ~(1 << MUX0);
            ADMUX |= (1 << MUX1);
            ADMUX &= ~(1 << MUX2);
            ADMUX &= ~(1 << MUX3);

   // Establecer la referencia de voltaje a Vcc
   ADMUX &= ~(1 << REFS1);
   ADMUX |= (1 << REFS0);
   // Habilitar el ADC y la interrupción del ADC
   ADCSRA |= (1 << ADEN) | (1 << ADIE);
   // Establecer el prescaler del ADC a 128
   ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
   // Iniciar la conversión del ADC
   ADCSRA |= (1 << ADSC);
}
```

Simulación de la actividad en Tinkercad



Fotografía del prototipo armado



Conclusión

Las máquinas de estado son una herramienta esencial en la caja de herramientas de cualquier ingeniero o desarrollador. Su capacidad para modelar sistemas complejos de una manera que es fácil de entender y razonar hace que sean invaluables en una amplia variedad de aplicaciones, desde el diseño de circuitos digitales y sistemas de control hasta el desarrollo de videojuegos y la inteligencia artificial. Al proporcionar una representación visual de cómo un sistema responde a diferentes eventos y condiciones, las máquinas de estado pueden ayudar a los diseñadores y desarrolladores a identificar errores, optimizar el rendimiento y mejorar la robustez de sus sistemas.

Además, las máquinas de estado no solo son útiles para modelar sistemas existentes, sino también para diseñar nuevos sistemas. Al comenzar con una máquina de estado, los diseñadores pueden explorar diferentes diseños y tomar decisiones informadas sobre cómo debe comportarse su sistema bajo diferentes condiciones. Esto puede ser especialmente útil en las primeras etapas de un proyecto, cuando las decisiones de diseño pueden tener un impacto significativo en la complejidad, el rendimiento y la fiabilidad del sistema final. En resumen, las máquinas de estado son una herramienta poderosa y versátil que juega un papel crucial en la informática, la ingeniería y la tecnología.

El proyecto que estás realizando es un excelente ejemplo de cómo se pueden aplicar las máquinas de estado en la ingeniería biomédica. Al diseñar un sistema de control basado en un microcontrolador AVR, estás utilizando una máquina de estado para controlar el comportamiento del sistema en función de las entradas del usuario y las lecturas del sensor.

Esto te permite crear un sistema que responde de manera eficiente y predecible a las entradas del usuario, lo cual es fundamental en muchas aplicaciones de la ingeniería biomédica.

Referencias

colaboradores de Wikipedia. (2022, 1 abril). *Máquina de estados*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/M%C3%A1quina_de_estados

De Jess Agustn Flores Cuautle, J. (2020a). *Desarrollo y tendencias de la Ingenier a Biom Dica en M Xico*. https://www.redalyc.org/journal/5122/512267930008/

IBM documentation. (s. f.). https://www.ibm.com/docs/es/baw/19.x?topic=ss8jb4-19-x-com-ibm-wbpm-wid-main-doc-prodoverview-topics-cadaptivebo-html

Margit. (2022, 26 octubre). *Máquina de estados*. TechEdu. https://techlib.net/techedu/maquina-de-estados/

State machine (máquina de estados). (s. f.). MATLAB & Simulink. https://la.mathworks.com/discovery/state-machine.html

Wikiwand - máquina de estados finita. (s. f.). Wikiwand. https://www.wikiwand.com/pt/M%C3%A1quina_de_estados_finita