Ejercicio 05:

Motor PaP Unipolar

Docente: M.I. Jesús Daniel Garza Camarena

Semestre: agosto - diciembre 2021

Datos del alumno:

Nombre	Matrícula	Carrera
Eduardo Vicente Reyna Villela	1868879	ITS

Frecuencia: Jueves

Hora: M4-M6

Grupo: 001

No. De Lista: 38

Código con formato de FSM

```
/*************************************************
 * Nombre: Eduardo Vicente Reyna Villela          *
 * Hora clase: M4-M6                              *
 * Día: Jueves                                   *
 * N° de lista: 38                                *
 * N° de Equipo: No aplica                       *
 * Dispositivo: Atmega328P                        *
 * Rev: 1.0                                       *
 * Propósito de la actividad:                     *
 * Utilizar los puertos del microcontrolador para  *
 * controlar un Motor DC por medio del Driver L293D,*
 * además del uso de un Display de 7 Segmentos     *
 *                              Fecha: 07/10/2021 *
 *************************************************/
#include <avr/io.h>
#define F_CPU 1000000UL
#include <util/delay.h>


// Definiciones
#define OUT_A PORTB0
#define OUT_B PORTB1
#define OUT_C PORTB2
#define OUT_D PORTB3

#define BTN PORTD0

// Constantes
const int TIME = 100;

// Macros - Inputs
#define OUT_A_ON  PORTB |=  _BV( OUT_A )
#define OUT_A_OFF PORTB &= ~_BV( OUT_A )

#define OUT_B_ON  PORTB |=  _BV( OUT_B )
#define OUT_B_OFF PORTB &= ~_BV( OUT_B )

#define OUT_C_ON  PORTB |=  _BV( OUT_C )
#define OUT_C_OFF PORTB &= ~_BV( OUT_C )

#define OUT_D_ON  PORTB |=  _BV( OUT_D )
#define OUT_D_OFF PORTB &= ~_BV( OUT_D )


// FSM
enum states
{
    s0,
    s1,
    s2,
    s3
}state;

uint8_t btn_state = 0;
```

```
//Funciones
void init_ports(void);
uint8_t debounce(uint8_t);


int main(void)
{
    init_ports();

    state = s0;

    while (1)
    {
        switch(state)
        {
            //-- State 0
            case s0:
                OUT_A_ON;
                OUT_B_OFF;
                OUT_C_OFF;
                OUT_D_ON;
                if ( debounce(BTN) )
                {
                    state = s1;
                }
                else
                {
                    state = s3;
                }
                _delay_ms(TIME);
            break;

            //-- State 1
            case s1:
                OUT_A_ON;
                OUT_B_ON;
                OUT_C_OFF;
                OUT_D_OFF;
                if ( debounce(BTN) )
                {
                    state = s2;
                }
                else
                {
                    state = s0;
                }
                _delay_ms(TIME);
            break;

            //-- State 2
            case s2:
                OUT_A_OFF;
                OUT_B_ON;
                OUT_C_ON;
                OUT_D_OFF;
                if ( debounce(BTN) )
```

```
                                                                    {
                        state = s3;
                    }
                    else
                    {
                        state = s1;
                    }
                    _delay_ms(TIME);
                break;

                //-- State 3
                case s3:
                    OUT_A_OFF;
                    OUT_B_OFF;
                    OUT_C_ON;
                    OUT_D_ON;
                    if ( debounce(BTN) )
                    {
                        state = s0;
                    }
                    else
                    {
                        state = s2;
                    }
                    _delay_ms(TIME);
                break;
            }// Fin switch
        }// Fin while
}// Fin init main

//init_ports : Inicializa los puertos
void init_ports(void)
{
    // Entradas
    DDRD  &= ~( _BV(BTN) );
    PORTD |=  ( _BV(BTN) );


    // Salidas
    DDRB |= ( _BV(OUT_A) | _BV(OUT_B)| _BV(OUT_C)| _BV(OUT_D) );

}

//debounce : detecta las transisciones positivas de los botones
uint8_t debounce(uint8_t btn)
{
    uint8_t valor_nuevo = bit_is_clear(PIND, btn);

    uint8_t result = ( (valor_nuevo) && btn_state );

    btn_state = valor_nuevo;

    if ( result )
    {
        return 1;
    }
```

```
    _delay_ms(25);

        return 0;
}
```

Diagrama de máquina de estados

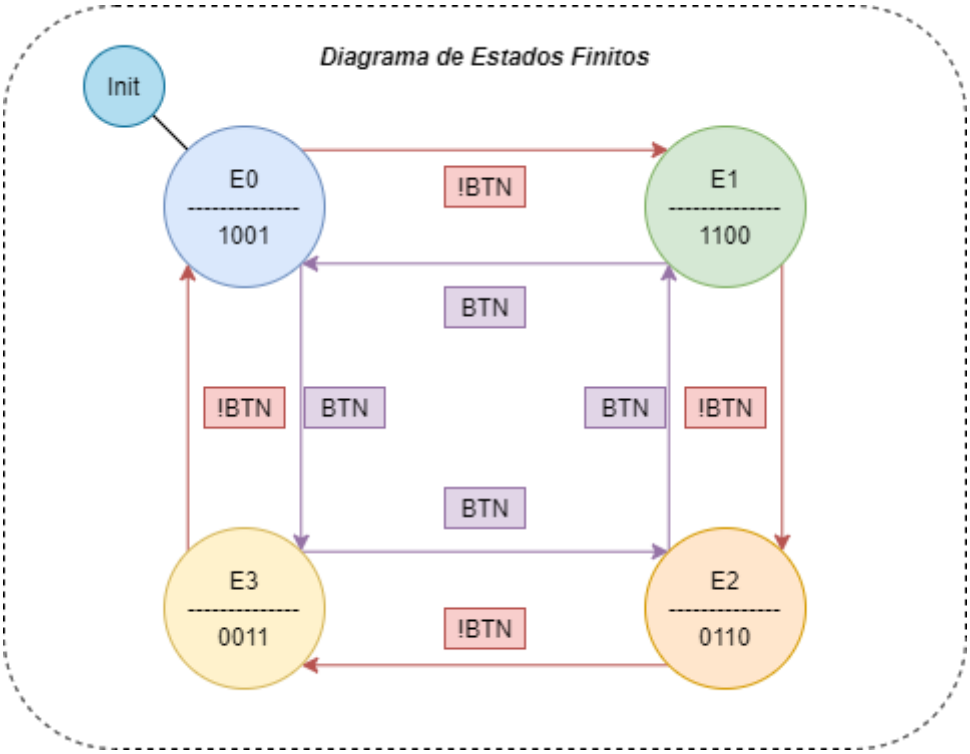

Ilustración 1 Diagrama de Estados elaborado en Draw.io

Imagen del esquemático

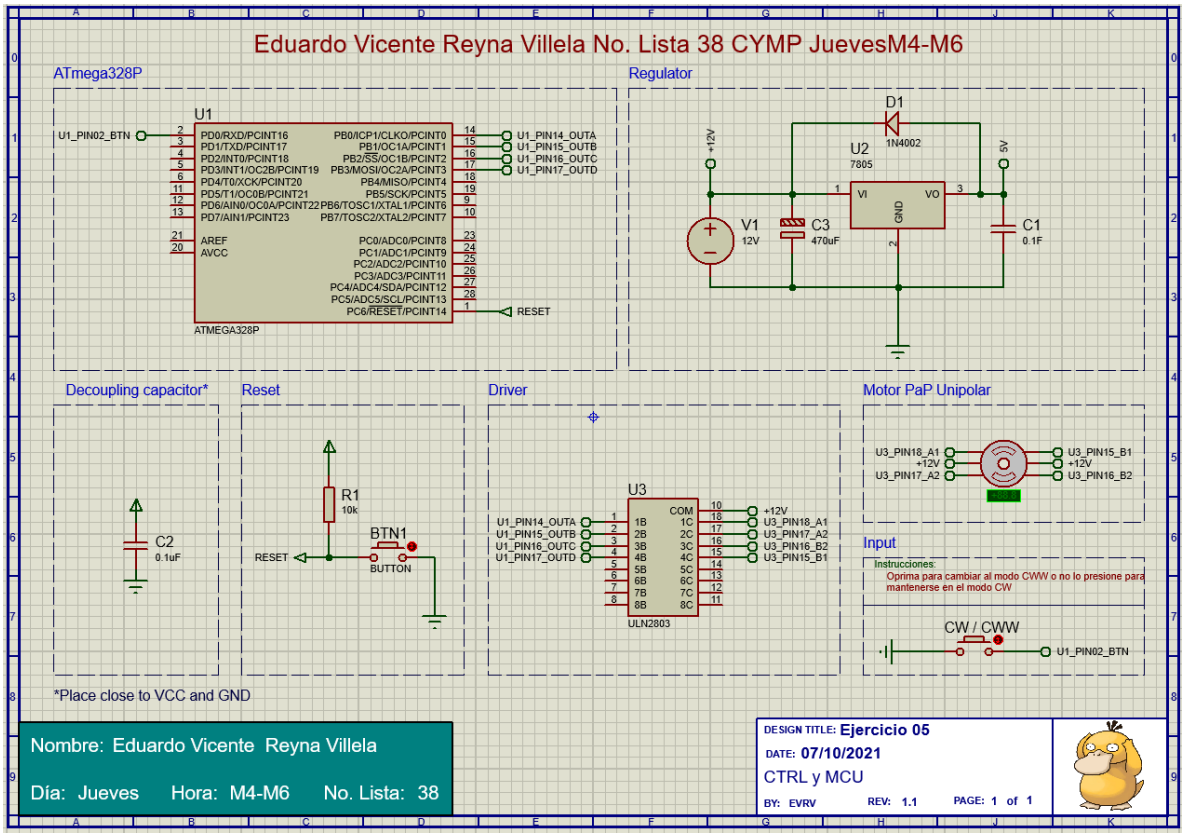

Ilustración 2 Diagrama Esquemático elaborado en Proteus

Agregar un eliminador de rebotes

```
//debounce : detecta las transisciones positivas de los botones
uint8_t debounce(uint8_t btn)
{
    uint8_t valor_nuevo = bit_is_clear(PIND, btn);

    uint8_t result = ( (valor_nuevo) && btn_state );

    btn_state = valor_nuevo;

    if ( result )
    {
        return 1;
    }

    _delay_ms(25);

    return 0;
}
```