**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

Imagen que contiene Gráfico

Descripción generada automáticamente

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**;**

**}**