 **Universidad Autónoma de Nuevo León**

**Facultad de Ingeniería Mecánica y Eléctrica**

CONTROLADORES Y MICROCONTROLADORES PROGRAMABLES

Producto Integrador de aprendizaje

PIA

Nombre o nombres de los integrantes junto con su matrícula:

Nahaliel Gamaliel Ríos Martínez 1884244

Ing. Jesus Daniel Garza Camarena

Semestre Febrero 2021 – Junio 2021

MN1N2

San Nicolás de los Garza, N.L. 3.06.2021

# Objetivo

Aplicar el conocimiento adquirido durante el semestre y aplicarlo en un proyecto relacionado a la asignatura

# Redacción del problema

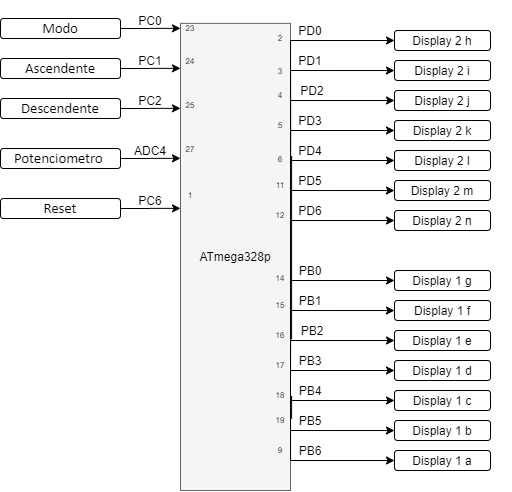
Diseñe, efectué la simulación y construya un prototipo de un sistema electrónico que muestre varias funciones mediante 2 displays 7 segmentos, el funcionamiento debe de estar basado en una máquina de estados cada vez que se dé clic a un botón estos cambiaran de modalidad.

El sistema cuenta con 4 modos que deben de trabajar bajo las siguientes condiciones:

1. Cuando se conecte el sistema este debe de mostrar un contador automático ascendente de 0 a 99, su incremento debe de ser por TIMER y no por delay.
2. Si se presiona el botón "modo" el sistema deberá de cambiar de modalidad y el contador 0 a 99 deberá de aumentar de forma manual desde un botón "ascendente".
3. Si se presiona de nuevo el botón "modo" el sistema deberá de cambiar de modalidad y el contador 0 a 99 deberá de descender de forma manual desde un botón "descendente".
4. Si se presiona de nuevo el botón "modo" el sistema deberá de cambiar de modalidad y los display deberán demostrar la lectura de un potenciómetro ADC (0V → 00, 5V → 99)

Se pueden utilizar una técnica de barrido para mostrar los displays, decodificadores o pueden estar conectados directamente a los puertos del MCU

**Diagrama de bloques**



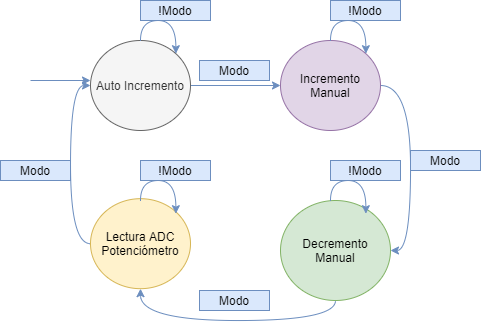
# Tabla del funcionamiento en alto nivel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tabla del funcionamiento | | | | |
| Modo de operación | Entradas | | | |
| Modo | Incremento | Decremento | Potenciómetro |
| Auto Incremento | 1 | 0 | 0 | 0 |
| Incremento Manual | 1 | 1 | 0 | 0 |
| Decremento Manual | 1 | 0 | 1 | 0 |
| Lectura ADC Potenciómetro | 1 | 0 | 0 | 1 |

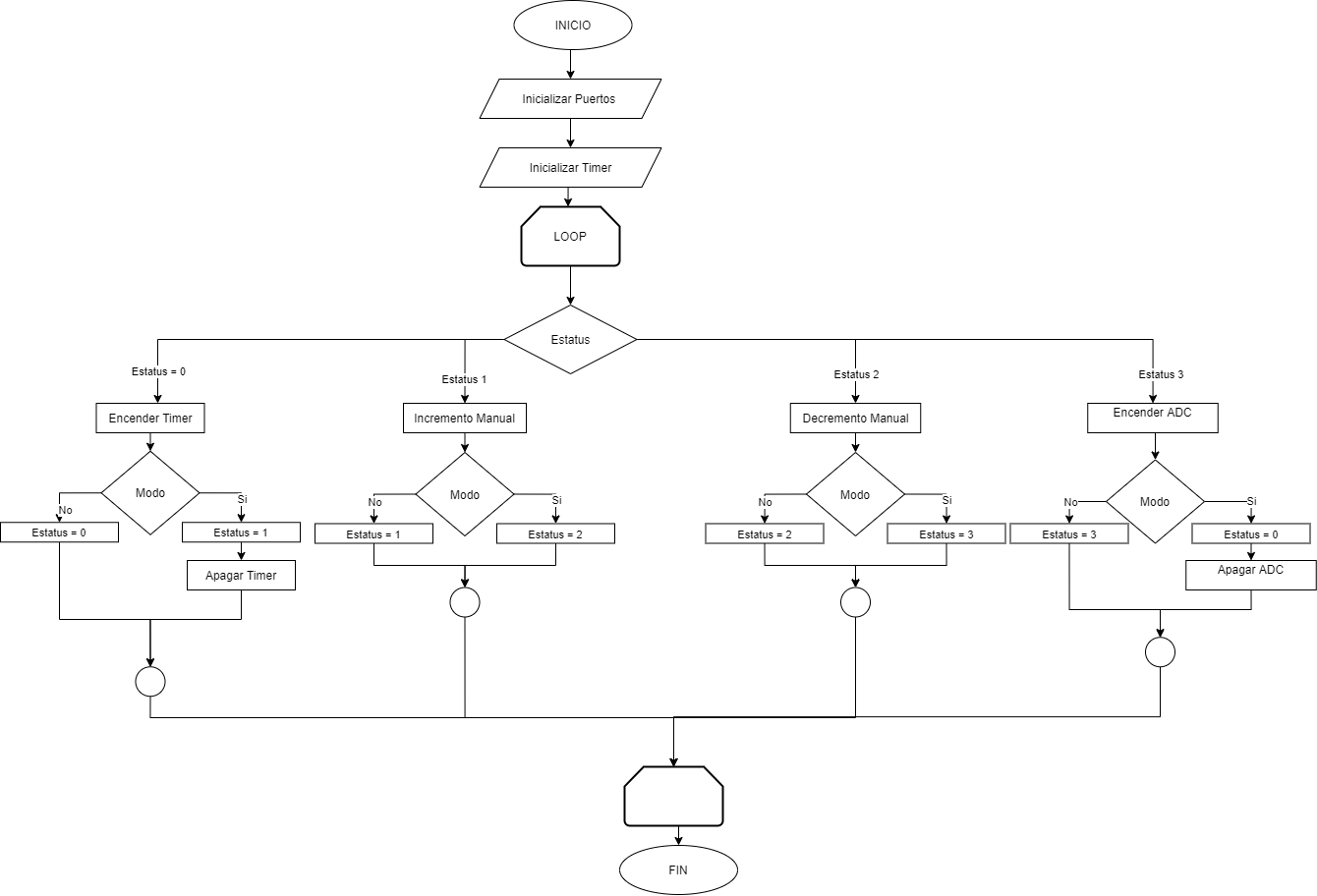
# Tabla de estado siguiente

|  |  |  |  |
| --- | --- | --- | --- |
| Tabla estado siguiente | | | |
| Inputs | | | Outputs |
| Modo | 0 | 1 |
| Estado Actual | Estado Siguiente | | Display |
| E0 | E0 | E1 | Auto Incremento |
| E1 | E1 | E2 | Incremento Manual |
| E2 | E2 | E3 | Decremento Manual |
| E3 | E3 | E2 | Lectura ADC Potenciómetro |

# Diagrama de transición de estados finitos



# Diagrama de flujo del código en alto nivel



# Lista de materiales utilizados

1 ATMEGA328p

14 resistencias 330

1 resistencia 10K

1 capacitor 100nF

1 capacitor 1nF

1 capacitor 0.1uF

2 display 7seg cc

4 botones

1 potenciómetro

# Código en lenguaje C.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* LLENAR ESTE ESPACIO CON LOS SIGUIENTES DATOS: \*

\* Nombre: Nahaliel Gamaliel Rios Martinez \*

\* Hora clase: N4 \*

\* Día: LMV \*

\* N° de lista: 33 \*

\* Dispositivo: ATMEGA328P \*

\* Rev: 1.0 \*

\* Propósito de la actividad: \*

\* Diseñe, efectué la simulación y construya un \*

\* prototipo de un sistema electrónico que muestre \*

\* varias funciones mediante 2 displays 7 segmentos,\*

\* el funcionamiento debe de estar basado en una \*

\* maquina de estados cada vez que se de clic a un \*

\* botón estos cambiaran de modalidad. \*

\* El sistema cuenta con 4 modos que deben de \*

\* trabajar bajo las siguientes condiciones: \*

\* \*

\* 1. Cuando se conecte el sistema este debe de \*

\* mostrar un contador automático ascendente de \*

\* 0 a 99, su incremento debe de ser por TIMER \*

\* y no por delay. \*

\* \*

\* 2. Si se presiona el botón "modo" el sistema \*

\* deberá de cambiar de modalidad y el contador 0 a \*

\* 99 deberá de aumentar de forma manual desde un \*

\* botón "ascendente". \*

\* \*

\* 3. Si se presiona de nuevo el botón "modo" el \*

\* sistema deberá de cambiar de modalidad y el \*

\* contador 0 a 99 deberá de descender de forma \*

\* manual desde un botón "descendente". \*

\* \*

\* 4. Si se presiona de nuevo el botón "modo" el \*

\* sistema deberá de cambiar de modalidad y los \*

\* display deberán de mostrar la la lectura de un \*

\* potenciómetro ADC (0V ? 00, 5V ? 99) \*

\* \*

\* Fecha: 3.06.2021 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*atmega328P PIN - OUT\*/

/\* PIN - OUT

atmega328P

-------

PC6 |1 28| PC5

PD0 |2 27| PC4

PD1 |3 26| PC3

PD2 |4 25| PC2

PD3 |5 24| PC1

PD4 |6 23| PC0

VCC |7 22| GND

GND |8 21| AREF

PB6 |9 20| AVCC

PB7 |10 19| PB5

PD5 |11 18| PB4

PD6 |12 17| PB3

PD7 |13 16| PB2

PB0 |14 15| PB1

--------

\*/

/\*atmega328P PIN FUNCTIONS\*/

/\*

atmega328P PIN FUNCTIONS

pin function name pin function name

1 !RESET/PCINT14 PC6 15 PCINT1/OC1A PB1

2 RxD/PCINT16 PD0 16 PCINT2/OC1B/SS PB2

3 TxD/PCINT17 PD1 17 PCINT3/OC2A/MOSI PB3

4 INT0/PCINT18 PD2 18 PCINT4/MISO PB4

5 INT1/PCINT19/OC2B PD3 19 PCINT5/SCK PB5

6 PCINT20 PD4 20 ANALOG VCC AVCC

7 +5v VCC 21 ANALOG REFERENCE AREF

8 GND GND 22 GND GND

9 XTAL1/PCINT6 PB6 23 PCINT8/ADC0 PC0

10 XTAL2/PCINT7 PB7 24 PCINT9/ADC1 PC1

11 PCINT21/OC0B PD5 25 PCINT10/ADC2 PC2

12 PCINT22/OC0A/AIN0 PD6 26 PCINT11/ADC3 PC3

13 PCINT23/AIN1 PD7 27 PCINT12/ADC4/SDA PC4

14 PCINT0/AIN1 PB0 28 PCINT13/ADC5/SCL PC5

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Bibliotecas\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <avr/io.h>

#include <avr/interrupt.h>// librería de interrupciones

#include <avr/delay.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Macros y constantes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define *F\_CPU* 1000000UL //1 Mhz

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Variables globales\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//--Espacio para declarar variables globales

#define a PINB0

#define b PINB1

#define c PINB2

#define d PINB3

#define e PINB4

#define f PINB5

#define g PINB6

#define h PIND0

#define i PIND1

#define j PIND2

#define k PIND3

#define l PIND4

#define m PIND5

#define n PIND6

#define MODO PINC0

#define ASCENDENTE PINC1

#define DESCENDENTE PINC2

#define Mod\_Press bit\_is\_set(PINC,MODO)

#define Btn\_Ascendente bit\_is\_clear(PINC,ASCENDENTE)

#define Btn\_Descendente bit\_is\_clear(PINC,DESCENDENTE)

*uint8\_t* numeros[10] = {

//gfedcba

0b0111111, //0

0b0000110, //1

0b1011011, //2

0b1001111, //3

0b1100110, //4

0b1101101, //5

0b1111101, //6

0b1000111, //7

0b1111111, //8

0b1100111, //9

};

enum states

{

state\_0, //0 - auto increíble

state\_1, //1 - encender y apagar cada 1 seg

state\_2, //2 - encender y apagar 1 a 1 cada 1 seg

state\_3 //3 - Efecto libre

} state;

//volatile int num;

//volatile int dis1;

//volatile int dis2;

volatile *uint8\_t* display1;

volatile *uint8\_t* display2;

unsigned on\_off;

unsigned on\_off\_adc;

volatile char timer = 0; //Contador para el timer

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Funciones\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//--Espacio para Establecer funciones

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Declaración de Funciones\*\*\*\*\*\*\*\*\*\*\*\*\*/

//--Espacio para declarar funciones

void initialize\_ports(void); // Inicializar puertos

void initialize\_timer(void);// Función para inicializar Timer\_0

void timer\_on(void);// Función para encender Timer\_0

void timer\_off(void);// Función para apagar Timer\_0

void ADC\_init(void);

void ADC\_on(void);

void ADC\_off(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Programa principal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)

{

//--Inicialización

cli();

initialize\_ports(); // va hacía la inicialización

initialize\_timer();// va hacía la inicialización del TIMER para controlar Led

ADC\_init();

sei();

//inicialización de variables globales necesarias para control de switch

state = state\_0;

on\_off = 0;

on\_off\_adc = 0;

//--Ejecución

while (1) //loop infinito

{

//Maquina de estados

switch (state)

{

//1. Cuando se conecte el sistema este debe de mostrar un contador automático ascendente de 0 a 99, su incremento debe de ser por TIMER y no por delay.

case state\_0:

if (Mod\_Press)

{

if (on\_off == 0){

timer\_on(); //Encendemos Timer0

on\_off = 1;

}

//

}else{

state = state\_1;

timer\_off(); //Encendemos Timer0

on\_off = 0;

*\_delay\_ms*(200);

break;

}

PORTB = numeros[display1];

PORTD = numeros[display2];

break;

//2. Si se presiona el botón "modo" el sistema deberá de cambiar de modalidad y el contador 0 a 99 deberá de aumentar de forma manual desde un botón "ascendente".

case state\_1:

if (Mod\_Press)

{

if (Btn\_Ascendente){

display2++;

if (display2 == 10){

display1++;

display2 = 0;

}

if (display1 == 10){

display1 = 0;

}

*\_delay\_ms*(200);

}

}else{

state = state\_2;

*\_delay\_ms*(200);

break;

}

PORTB = numeros[display1];

PORTD = numeros[display2];

break;

//3. Si se presiona de nuevo el botón "modo" el sistema deberá de cambiar de modalidad y el contador 0 a 99 deberá de descender de forma manual desde un botón "descendente".

case state\_2:

if (Mod\_Press)

{

if (Btn\_Descendente){

display2--;

if (display2 < 1) {

display1--;

display2 = 9;

}else if (display2 > 10){

display1--;

display2 = 9;

}

if (display1 < 1){

display1 = 9;

}else if (display1 > 10){

display1 = 9;

}

*\_delay\_ms*(200);

}

}else{

state = state\_3;

*\_delay\_ms*(200);

break;

}

PORTB = numeros[display1];

PORTD = numeros[display2];

break;

//4.- Si se presiona de nuevo el botón "modo" el sistema deberá de cambiar de modalidad y los display deberán de mostrar la la lectura de un potenciómetro ADC (0V ? 00, 5V ? 99)

case state\_3:

if (Mod\_Press)

{

//state = state\_0;

if (on\_off\_adc == 0){

ADC\_on(); //

on\_off\_adc = 1;

}

//

}else{

state = state\_0;

ADC\_off(); //

on\_off\_adc = 0;

*\_delay\_ms*(200);

break;

}

//PORTB = numeros[display1];

//PORTD = numeros[display2];

break;

}

}// END loop infinito

}// END MAIN

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Definición de funciones\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Descripcion de lo que hace la funcion: \*

//initialize\_ports : inicializa los puertos de entrada y \*

//salida \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void initialize\_ports(void)

{

//--Entradas

DDRC &= ~\_BV(MODO);

PORTC |= \_BV(MODO); //Activamos PULL Up

DDRC &= ~\_BV(ASCENDENTE);

PORTC |= \_BV(ASCENDENTE); //Activamos PULL Up

DDRC &= ~\_BV(DESCENDENTE);

PORTC |= \_BV(DESCENDENTE); //Activamos PULL Up

//--Salidas

DDRB |=\_BV(a);

DDRB |=\_BV(b);

DDRB |=\_BV(c);

DDRB |=\_BV(d);

DDRB |=\_BV(e);

DDRB |=\_BV(f);

DDRB |=\_BV(g);

DDRD |=\_BV(h);

DDRD |=\_BV(i);

DDRD |=\_BV(j);

DDRD |=\_BV(k);

DDRD |=\_BV(l);

DDRD |=\_BV(m);

DDRD |=\_BV(n);

PORTD = 0x00; //-Por seguridad iniciamos en 0

PORTB = 0x00; //-Por seguridad iniciamos en 0

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//initialize\_timer\_led : inicializa el timer para controlar Display

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void initialize\_timer(void)

{

//Modo de operacíon configurado como CTC

TCCR0A &=~ (1<<WGM00);// 0 en el bit WGM00

TCCR0A |= (1<<WGM01);// 1 en el bit WGM01

TCCR0B &=~ (1<<WGM02);// 0 en el bit WGM02

OCR0A = 97; //Registro de 8 bits donde se pone el numero a comparar

TIMSK0 |= (1<<OCIE0A);//Se pone un 1 en el bit OCIE0A del registro

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//timer\_on: Enciende el timer para controlar Display

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void timer\_on(void)

{

TCNT0 = 0; // Registro de 8 bits que lleva el conteo del timer\_0

//Prescaler configurado en 1024

TCCR0B |= (1<<CS00);// 1 en el bit CS00

TCCR0B &=~ (1<<CS01);// 0 en el bit CS01

TCCR0B |= (1<<CS02);// 1 en el bit CS02

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//timer\_off: Apaga el timer 0

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void timer\_off(void)

{

//Modo Timer detenido

TCCR0B &=~ (1<<CS00);// 0 en el bit CS00

TCCR0B &=~ (1<<CS01);// 0 en el bit CS01

TCCR0B &=~ (1<<CS02);// 0 en el bit CS02

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Descripcion de lo que hace la funcion: \*

//ADC\_init : Habilitamos la interrupción y configuramos \*

// el ADC \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void ADC\_init(void)

{

//Avcc como pin de referencia

ADMUX &=~ (1<<REFS1);

ADMUX |= (1<<REFS0);

//8 bits

ADMUX |= (1<<ADLAR);

//PIN ADC4

ADMUX &=~ (1<<MUX3);

ADMUX |= (1<<MUX2);

ADMUX &=~ (1<<MUX1);

ADMUX &=~ (1<<MUX0);

//Freeruning

ADCSRA |= (1<<ADATE);

//Habilitar interrupción

ADCSRA |= (1<<ADIE);

//velocidad de muestreo

// 1 MHz clock / 8 = 125 kHz ADC clock debe de estar entre 50 - 200Khz

ADCSRA &=~ (1<<ADPS0);

ADCSRA |= (1<<ADPS1);

ADCSRA |= (1<<ADPS2);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Descripcion de lo que hace la funcion: \*

//ADC\_init : Leer y convertir señal análoga \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void ADC\_on(void)

{

//Encendemos el ADC

ADCSRA |= (1<<ADEN);

*\_delay\_ms*(10);

// Iniciar la conversión

ADCSRA |= (1 << ADSC);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Descripcion de lo que hace la funcion: \*

//ADC\_init : Apagar ADC \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void ADC\_off(void)

{

//Apaga el ADC

ADCSRA &=~ (1<<ADEN);

*\_delay\_ms*(10);

// Apaga la conversión

ADCSRA &=~ (1 << ADSC);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Vectores de interrupción, Interrupt service routine (ISR)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//TIMER\_0: Modificar display cada 0.2 Segundos

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ISR (TIMER0\_COMPA\_vect)// Vector de interrupción para el Timer0 (0.1s)

{

timer++;

if (timer == 2)

{

display2++;

timer = 0;

if (display2 == 10){

display1++;

display2 = 0;

}

if (display1 == 10){

display1 = 0;

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//ISR : Leer Potenciometro y modificar display

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ISR(ADC\_vect)

{

//0 a 5V -> 0 a 255bits

if ((ADCH >= 0) && (ADCH <= 2)) {PORTB = numeros[0];PORTD = numeros[0];}

else if((ADCH >= 3 ) && (ADCH <= 4)){PORTB = numeros[0];PORTD = numeros[1];}

else if((ADCH >= 5 ) && (ADCH <= 7)){PORTB = numeros[0];PORTD = numeros[2];}

else if((ADCH >= 8 ) && (ADCH <= 9 )){PORTB = numeros[0];PORTD = numeros[3];}

else if((ADCH >= 10) && (ADCH <= 12)){PORTB = numeros[0];PORTD = numeros[4];}

else if((ADCH >= 13) && (ADCH <= 14)){PORTB = numeros[0];PORTD = numeros[5];}

else if((ADCH >= 15) && (ADCH <= 17)){PORTB = numeros[0];PORTD = numeros[6];}

else if((ADCH >= 18) && (ADCH <= 20)){PORTB = numeros[0];PORTD = numeros[7];}

else if((ADCH >= 21) && (ADCH <= 22)){PORTB = numeros[0];PORTD = numeros[8];}

else if((ADCH >= 23) && (ADCH <= 25)){PORTB = numeros[0];PORTD = numeros[9];}

else if((ADCH >= 26) && (ADCH <= 27)){PORTB = numeros[1];PORTD = numeros[0];}

else if((ADCH >= 28) && (ADCH <= 30)){PORTB = numeros[1];PORTD = numeros[1];}

else if((ADCH >= 31) && (ADCH <= 32)){PORTB = numeros[1];PORTD = numeros[2];}

else if((ADCH >= 33) && (ADCH <= 35)){PORTB = numeros[1];PORTD = numeros[3];}

else if((ADCH >= 36) && (ADCH <= 38)){PORTB = numeros[1];PORTD = numeros[4];}

else if((ADCH >= 39) && (ADCH <= 40)){PORTB = numeros[1];PORTD = numeros[5];}

else if((ADCH >= 41) && (ADCH <= 43)){PORTB = numeros[1];PORTD = numeros[6];}

else if((ADCH >= 44) && (ADCH <= 45)){PORTB = numeros[1];PORTD = numeros[7];}

else if((ADCH >= 46) && (ADCH <= 48)){PORTB = numeros[1];PORTD = numeros[8];}

else if((ADCH >= 49) && (ADCH <= 51)){PORTB = numeros[1];PORTD = numeros[9];}

else if((ADCH >= 52) && (ADCH <= 53)){PORTB = numeros[2];PORTD = numeros[0];}

else if((ADCH >= 54) && (ADCH <= 56)){PORTB = numeros[2];PORTD = numeros[1];}

else if((ADCH >= 57) && (ADCH <= 58)){PORTB = numeros[2];PORTD = numeros[2];}

else if((ADCH >= 59) && (ADCH <= 61)){PORTB = numeros[2];PORTD = numeros[3];}

else if((ADCH >= 61) && (ADCH <= 63)){PORTB = numeros[2];PORTD = numeros[4];}

else if((ADCH >= 64) && (ADCH <= 66)){PORTB = numeros[2];PORTD = numeros[5];}

else if((ADCH >= 67) && (ADCH <= 69)){PORTB = numeros[2];PORTD = numeros[6];}

else if((ADCH >= 70) && (ADCH <= 71)){PORTB = numeros[2];PORTD = numeros[7];}

else if((ADCH >= 72) && (ADCH <= 74)){PORTB = numeros[2];PORTD = numeros[8];}

else if((ADCH >= 75) && (ADCH <= 76)){PORTB = numeros[2];PORTD = numeros[9];}

else if((ADCH >= 77) && (ADCH <= 79)){PORTB = numeros[3];PORTD = numeros[0];}

else if((ADCH >= 80) && (ADCH <= 81)){PORTB = numeros[3];PORTD = numeros[1];}

else if((ADCH >= 82) && (ADCH <= 84)){PORTB = numeros[3];PORTD = numeros[2];}

else if((ADCH >= 85) && (ADCH <= 87)){PORTB = numeros[3];PORTD = numeros[3];}

else if((ADCH >= 88) && (ADCH <= 89)){PORTB = numeros[3];PORTD = numeros[4];}

else if((ADCH >= 90) && (ADCH <= 92)){PORTB = numeros[3];PORTD = numeros[5];}

else if((ADCH >= 93) && (ADCH <= 94)){PORTB = numeros[3];PORTD = numeros[6];}

else if((ADCH >= 95) && (ADCH <= 97)){PORTB = numeros[3];PORTD = numeros[7];}

else if((ADCH >= 98) && (ADCH <= 99)){PORTB = numeros[3];PORTD = numeros[8];}

else if((ADCH >= 100) && (ADCH <= 102)){PORTB = numeros[3];PORTD = numeros[9];}

else if((ADCH >= 103) && (ADCH <= 105)){PORTB = numeros[4];PORTD = numeros[0];}

else if((ADCH >= 106) && (ADCH <= 107)){PORTB = numeros[4];PORTD = numeros[1];}

else if((ADCH >= 108) && (ADCH <= 110)){PORTB = numeros[4];PORTD = numeros[2];}

else if((ADCH >= 111) && (ADCH <= 112)){PORTB = numeros[4];PORTD = numeros[3];}

else if((ADCH >= 113) && (ADCH <= 115)){PORTB = numeros[4];PORTD = numeros[4];}

else if((ADCH >= 116) && (ADCH <= 117)){PORTB = numeros[4];PORTD = numeros[5];}

else if((ADCH >= 118) && (ADCH <= 120)){PORTB = numeros[4];PORTD = numeros[6];}

else if((ADCH >= 121) && (ADCH <= 123)){PORTB = numeros[4];PORTD = numeros[7];}

else if((ADCH >= 124) && (ADCH <= 125)){PORTB = numeros[4];PORTD = numeros[8];}

else if((ADCH >= 126) && (ADCH <= 128)){PORTB = numeros[4];PORTD = numeros[9];}

else if((ADCH >= 129) && (ADCH <= 130)){PORTB = numeros[5];PORTD = numeros[0];}

else if((ADCH >= 131) && (ADCH <= 133)){PORTB = numeros[5];PORTD = numeros[1];}

else if((ADCH >= 134) && (ADCH <= 136)){PORTB = numeros[5];PORTD = numeros[2];}

else if((ADCH >= 137) && (ADCH <= 138)){PORTB = numeros[5];PORTD = numeros[3];}

else if((ADCH >= 139) && (ADCH <= 141)){PORTB = numeros[5];PORTD = numeros[4];}

else if((ADCH >= 142) && (ADCH <= 143)){PORTB = numeros[5];PORTD = numeros[5];}

else if((ADCH >= 144) && (ADCH <= 146)){PORTB = numeros[5];PORTD = numeros[6];}

else if((ADCH >= 147) && (ADCH <= 148)){PORTB = numeros[5];PORTD = numeros[7];}

else if((ADCH >= 149) && (ADCH <= 151)){PORTB = numeros[5];PORTD = numeros[8];}

else if((ADCH >= 152) && (ADCH <= 154)){PORTB = numeros[5];PORTD = numeros[9];}

else if((ADCH >= 155) && (ADCH <= 156)){PORTB = numeros[6];PORTD = numeros[0];}

else if((ADCH >= 157) && (ADCH <= 159)){PORTB = numeros[6];PORTD = numeros[1];}

else if((ADCH >= 160) && (ADCH <= 161)){PORTB = numeros[6];PORTD = numeros[2];}

else if((ADCH >= 162) && (ADCH <= 164)){PORTB = numeros[6];PORTD = numeros[3];}

else if((ADCH >= 165) && (ADCH <= 166)){PORTB = numeros[6];PORTD = numeros[4];}

else if((ADCH >= 167) && (ADCH <= 169)){PORTB = numeros[6];PORTD = numeros[5];}

else if((ADCH >= 170) && (ADCH <= 172)){PORTB = numeros[6];PORTD = numeros[6];}

else if((ADCH >= 173) && (ADCH <= 174)){PORTB = numeros[6];PORTD = numeros[7];}

else if((ADCH >= 175) && (ADCH <= 177)){PORTB = numeros[6];PORTD = numeros[8];}

else if((ADCH >= 178) && (ADCH <= 179)){PORTB = numeros[6];PORTD = numeros[9];}

else if((ADCH >= 180) && (ADCH <= 182)){PORTB = numeros[7];PORTD = numeros[0];}

else if((ADCH >= 183) && (ADCH <= 184)){PORTB = numeros[7];PORTD = numeros[1];}

else if((ADCH >= 185) && (ADCH <= 187)){PORTB = numeros[7];PORTD = numeros[2];}

else if((ADCH >= 188) && (ADCH <= 190)){PORTB = numeros[7];PORTD = numeros[3];}

else if((ADCH >= 191) && (ADCH <= 192)){PORTB = numeros[7];PORTD = numeros[4];}

else if((ADCH >= 193) && (ADCH <= 195)){PORTB = numeros[7];PORTD = numeros[5];}

else if((ADCH >= 196) && (ADCH <= 197)){PORTB = numeros[7];PORTD = numeros[6];}

else if((ADCH >= 198) && (ADCH <= 200)){PORTB = numeros[7];PORTD = numeros[7];}

else if((ADCH >= 201) && (ADCH <= 202)){PORTB = numeros[7];PORTD = numeros[8];}

else if((ADCH >= 203) && (ADCH <= 205)){PORTB = numeros[7];PORTD = numeros[9];}

else if((ADCH >= 206) && (ADCH <= 208)){PORTB = numeros[8];PORTD = numeros[0];}

else if((ADCH >= 209) && (ADCH <= 210)){PORTB = numeros[8];PORTD = numeros[1];}

else if((ADCH >= 211) && (ADCH <= 213)){PORTB = numeros[8];PORTD = numeros[2];}

else if((ADCH >= 214) && (ADCH <= 215)){PORTB = numeros[8];PORTD = numeros[3];}

else if((ADCH >= 216) && (ADCH <= 218)){PORTB = numeros[8];PORTD = numeros[4];}

else if((ADCH >= 219) && (ADCH <= 221)){PORTB = numeros[8];PORTD = numeros[5];}

else if((ADCH >= 222) && (ADCH <= 223)){PORTB = numeros[8];PORTD = numeros[6];}

else if((ADCH >= 224) && (ADCH <= 226)){PORTB = numeros[8];PORTD = numeros[7];}

else if((ADCH >= 227) && (ADCH <= 228)){PORTB = numeros[8];PORTD = numeros[8];}

else if((ADCH >= 229) && (ADCH <= 231)){PORTB = numeros[8];PORTD = numeros[9];}

else if((ADCH >= 232) && (ADCH <= 233)){PORTB = numeros[9];PORTD = numeros[0];}

else if((ADCH >= 234) && (ADCH <= 236)){PORTB = numeros[9];PORTD = numeros[1];}

else if((ADCH >= 237) && (ADCH <= 239)){PORTB = numeros[9];PORTD = numeros[2];}

else if((ADCH >= 240) && (ADCH <= 241)){PORTB = numeros[9];PORTD = numeros[3];}

else if((ADCH >= 242) && (ADCH <= 244)){PORTB = numeros[9];PORTD = numeros[4];}

else if((ADCH >= 245) && (ADCH <= 246)){PORTB = numeros[9];PORTD = numeros[5];}

else if((ADCH >= 247) && (ADCH <= 249)){PORTB = numeros[9];PORTD = numeros[6];}

else if((ADCH >= 250) && (ADCH <= 251)){PORTB = numeros[9];PORTD = numeros[7];}

else if((ADCH >= 252) && (ADCH <= 254)){PORTB = numeros[9];PORTD = numeros[8];}

else if((ADCH >= 255)){PORTB = numeros[9];PORTD = numeros[9];}

}

# Imagen del Diagrama Esquemático

Diagrama

Descripción generada automáticamente con confianza baja

# Imagen del diseño del circuito impreso

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza media

# Cotización del PCB

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

# Conclusión

En este proyecto puse en práctica muchos de los conocimientos adquiridos durante el curso de microcontroladores, desde la creación del esquemático, programación del microcontrolador, hasta el diseño del circuito impreso, aprendí a utilizar herramientas que están integradas dentro del propio microcontrolador y desconocía que existían, como el uso de timers, interrupciones, lectores de ADC, etc.

Aunque ya anteriormente había tenido la oportunidad de programar en Arduino, no entendía o comprendía muchas cosas que utilizaba, por ejemplo, a la hora de declarar un pin de salida o entrada, no sabía la lógica que había detrás para que el pin se comportara de una manera u otra, este tipo de cosas creo que son de gran valor para entender el funcionamiento del microcontrolador a detalle, y que después a la hora de encontrarnos con algún problema sea más fácil de resolver.

# Bibliografía

ATmega328P. 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash. DATASHEET. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P\_Datasheet.pdf