Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

IE-0624 Laboratorio de Microprocesadores

Laboratorio 2

José Antonio Ramos Pereira, B86485

Grupo 01

Prof. MSc. Marco Villalta Fallas.

https://github.com/JAR1224/Laboratorio_2_Jose_Ramos

Índice de contenidos

Índice de diagramas

1. Introducción

1.1. Resumen

La elaboración del proyecto consistió en plantear una solución utilizando un microcontrolador Attiny4313 y otros componentes electrónicos para simular una lavadora. La aplicación debía de poder capturar entradas provenientes de botones por medio de interrupciones, poder decrementar el tiempo restante de los estados de lavado mostrado en los LEDS de 7 segementos por medio del periférico de temporización y implementar el funcionamiento de la lavadora por medio de una FSM.

Los retos principales para la elaboración de la solución fueron los siguientes:

1. Extender las capacidades IO del microcontrolador
2. Uso del periférico de temporización

Para poder superar los retos, se realizaron los siguientes pasos:

1. Usar un registro desplazante para extender el IO del microcontrolador
2. Configurar el módulo timer con un periodo de forma que con el uso de conteros, se pueda generar las bases de tiempo necesarias para la aplicación.

1.2. Conclusiones

Se concluyó que por medio de las metodologías mencionadas anteriormente, se pudo implementar una solución al problema planteado que cumple con todos los requisitos de diseño requeridos por el enunciado.

2. Nota Teórica

2.1. Microcontrolador

Características generales

Las características generales son las siguientes [1]:

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 120 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
- Data and Non-volatile Program and Data Memories
  - 2/4K Bytes of In-System Self Programmable Flash
    - Endurance 10,000 Write/Erase Cycles
  - 128/256 Bytes In-System Programmable EEPROM
    - Endurance: 100,000 Write/Erase Cycles
  - 128/256 Bytes Internal SRAM
  - Programming Lock for Flash Program and EEPROM Data Security
- Peripheral Features
  - One 8-bit Timer/Counter with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare and Capture Modes
  - Four PWM Channels
  - On-chip Analog Comparator
  - Programmable Watchdog Timer with On-chip Oscillator
  - USI – Universal Serial Interface
  - Full Duplex USART
- Special Microcontroller Features
  - debugWIRE On-chip Debugging
  - In-System Programmable via SPI Port
  - External and Internal Interrupt Sources
  - Low-power Idle, Power-down, and Standby Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Brown-out Detection Circuit
  - Internal Calibrated Oscillator
- I/O and Packages
  - 18 Programmable I/O Lines
  - 20-pin PDIP, 20-pin SOIC, 20-pad MLF/VQFN

Características Generales 2.1: Obtenido de [1]

Diagrama de bloques Attiny4313

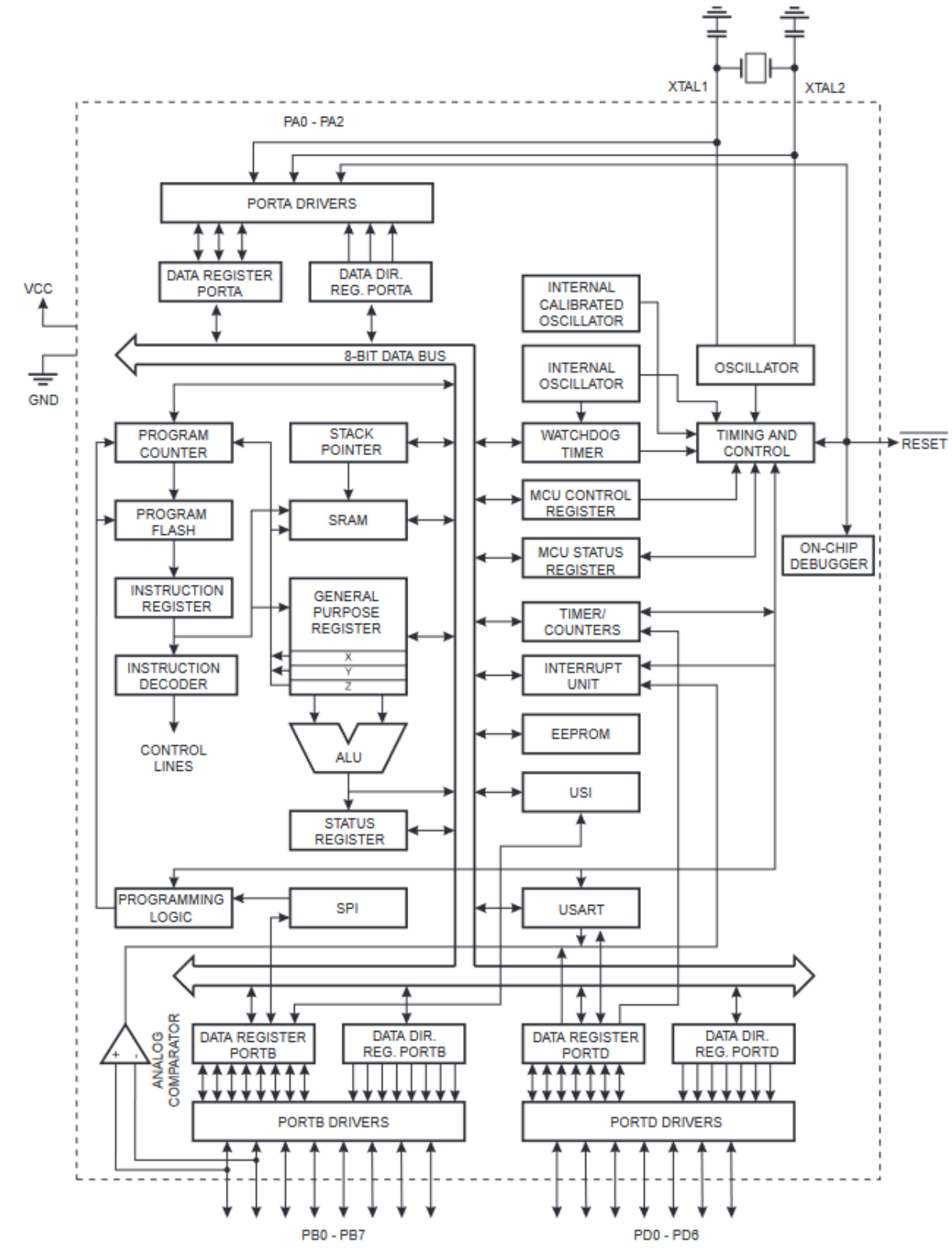


Diagrama de bloques 2.2: Obtenido de [1]

Diagrama de pines

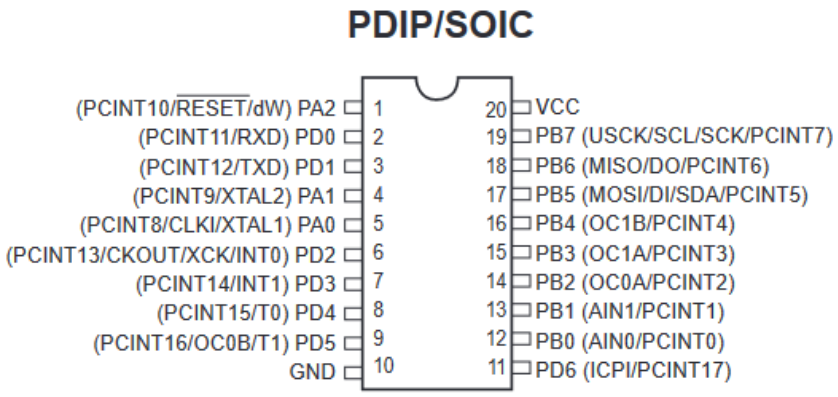


Diagrama de Pines 2.3: Obtenido de [1]

Características eléctricas

22.1 Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

Características eléctricas 2.4: Obtenido de [1]

Registros

Los registros que se utilizaron para la configuración de las interrupciones de temporización y interrupciones de señales externas fueron los siguientes:

TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Registro 2.5: Obtenido de [1]

Los bits que se configuraron para TCCR1B fueron WGM13...WGM12 y CS12...CS10. Un valor de 0b10 en WGM13...WGM12 ponen a operar le interrupción de timer1 en modo ctc. Este modo permite el borrado automático del contero del módulo timer cuando ocurre una igualdad entre el registro TCNT y el registro de 'output compare A' OCR1A. Un valor de 0b011 en CS12...CS10 elije un valor de preescalador de reloj de 64.

OCR1AH and OCR1AL – Output Compare Register 1 A

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	OCR1A[15:8]								OCR1AH
0x2A (0x4A)	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Registro 2.6: Obtenido de [1]

Para configurar el registro de OCR1A, se tomaron en cuenta los siguientes requerimientos:

1. Para poder lograr un duty cycle alto de forma que el LED se vea brillante, es conveniente poder dividir los pulsos en por lo menos 10 partes de forma que los LEDS puedan lograr un duty cycle cercano al 100 %.

2. La frecuencia de los LEDS debe de ser de almenos 60Hz para que se vea de forma continua para el ojo humano.

Se sabe que [1]:

$$T_{pulse} = \frac{N(OCRA1 + 1)}{f_{clk}} \tag{1}$$

N es el valor de preescalador, que como se menciona anteriormente, es 64. El valor de la frecuencia del reloj para el Attiny4313 es de 8 MHz.

Se elije lograr que los LEDS operen a 127 Hz, para esto se tiene que cumplir lo siguiente:

$$f_{LEDS} = \frac{1}{10nT_{pulse}} \tag{2}$$

Donde n es la cantidad de LEDS de 7 segmentos que se estén utilizando (en este caso 2). Despejando y resolviendo para OCR1A, se obtiene un valor de 48. Entonces en el registro OCR1A se escribe un 48.

TIMSK – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Registro 2.7: Obtenido de [1]

En el registro TIMSK, se escribe un 1 en el bit OCIE1A para activar la interrupción por output compara del registro A

MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Registro 2.8: Obtenido de [1]

En el registro MCUCR se escribe un 1 en ISC01 y ISC00 para activar la interrupcion INT0 cuando se detecte un nivel alto de tensión en el pin de salida correspondiente, en este caso en PD2. De esta forma, cuando se conecte un botón a este pin, cuando se presione, solo se va a generar la interrupción 1 vez en vez de dos veces.

GIMSK – General Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x3B (0x5B)	INT1	INT0	PCIE0	PCIE2	PCIE1	–	–	–	GIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Registro 2.9: Obtenido de [1]

En el registro GIMSK se escribe un 1 en INT0 para habilitar la interrupción cuando se detecte un nivel alto de tensión en el pin PD2.

2.2. Componentes Electrónicos Complementarios

Registro desplazante CD4094

CHARACTERISTIC	V_{DD} (V)	LIMITS		UNITS
		MIN.	MAX.	
Supply-Voltage Range (For T_A=Full Package-Temperature Range)		3	18	V
Data Setup Time, t_S	5 10 15	125 55 35	– – –	ns
Clock Pulse Width, t_W	5 10 15	200 100 83	– – –	ns
Clock Input Frequency, f_{CL}	5 10 15	dc	1.25 2.5 3	MHz
Clock Input Rise or Fall time, t_rCL, t_fCL:*	5 10 15	–	15 5 5	μs
Strobe Pulse Width, t_W	5 10 15	200 80 70	– – –	ns

Características temporales 2.10: Características temporales CD4094 [**?**]

Se observa que para una tensión de operación de 5V, todas las condiciones temporales son, a lo sumo, de 200ns. Como el ciclo de instrucción del ATtiny4313 es de 200ns, entonces no es necesario implementar retardos adicionales dentro del código de la aplicación para satisfacer las condiciones temporales del cd4094.

Resistencias

- 4x 50 kΩ
- 4x 10 kΩ
- 15x 50Ω

Capacitores

- 4x 100 nF

Amplificador Operacional

- 2x LMC7101

LEDS

- 2x Led 7 segmentos LSHD-5601
- 8x diodos LED

Componentes

- Decodificador BCD a bin
- OR de 4 entradas

2.3. Precios

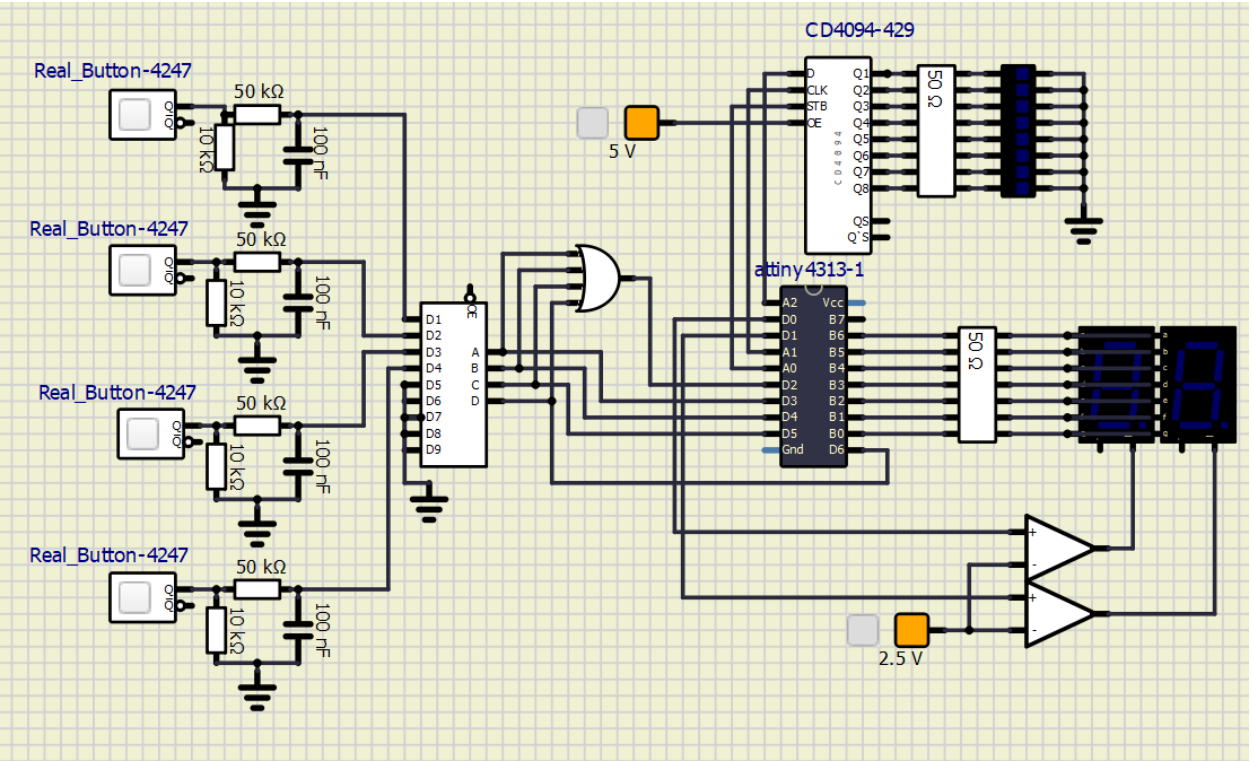
- 2x Op Amp LMC7101: $1.32
- 2x Led 7 segmentos LSHD-5601: $1.28
- Registro desplazante CD4094: $1.3
- 16x Resistencias: $0.00

- 4x Capacitor: $1
- ATTiny4313: $1.92
- Decodificador BCD a bin: $0.57
- OR de 4 entradas: $0.55
- 8x diodos LED: $1.92
- Total: **$9.86**

Obtenido de: [2]

2.4. Diseño del circuito

Esquemático



El circuito consta de 3 partes principales.

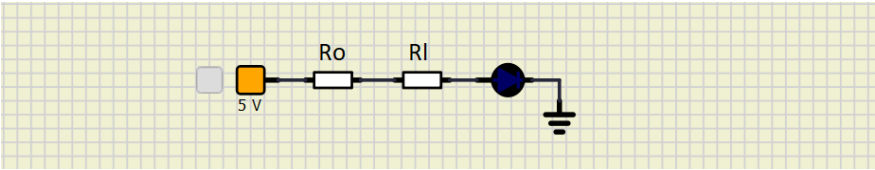
1) Decodificador

El decodificador se encarga de transformar las señales provenientes de los botones a un número en BCD. Como la interrupciones externas del ATtiny no vienen con la capacidad de poder identificar cual pin en específico causó la interrupción, entonces este decodificador se encarga de arreglar este problema. La salida de decodificador va a dar a los pines D3...D6 y a una OR de 4 entradas. La salida de la OR va a dar al pin D2 que es el que activa la interrupción. De esta forma, cuando se presiona un botón y el decodificar calcula so valor en BCD, se activa la interrupción y por medio de software se leen los pines D3...D6 para saber cual boton causó la interrupción.

2) LEDS 7 segmentos

Los pines B0...B6 se utilizan para activar cada uno de los segmentos de los LEDS de 7 segmentos. Los pines D0...D1 se utilizan para activar y desactivar las salidas de cada uno de los displays de 7 segmentos.

En la salida de los registros desplazantes se le conectan resistencias de 100Ω para restringir la cantidad de corriente que pasa por los LEDS. Para poder elegir el valor ideal de estas resistencias, se tiene que cumplir que la tensión de alimentación de 5V dividido entre la resistencia total (R_l + R_o), de una corriente menor a 20mA para evitar quemar los LEDS; donde R_l son las resistencias conectadas a la salida de los registros y R_o es la impedancia de salida de los pines del registro.



Circuito eq. LEDS 2.11: Circuito eq. LEDS

Sabiendo que en la hoja del fabricante se incluye que la impedancia de salida de los pines es de 170 Ω. Y sabiendo que:

$$\begin{aligned} \frac{5V}{R_l + R_o} &< 20mA \\ 250\Omega &< R_l + R_o \end{aligned} \tag{3}$$

Entonces, eligiendo R_l=100Ω, se obtiene una impedancia total de 270Ω > 250Ω.

3) Paralelización de salidas IO

Como no quedan suficientes pines IO para poder controlar individualmente 8 diodos leds, se utiliza un registro desplazante para poder extender las capacidades IO del microcontrolador. Este registro se conecta a los pines A0...A2 y se utilizan para mandar la información de salida de forma serial al registro desplazante y convertirlos en información paralela.

Para elegir los valores de las resistencias que restringen la cantidad de corriente que pasa por los LEDS, se aplica la misma metodología que se proporciona en la sección anterior para los LEDS de 7 segmentos.

2.5. Temas de laboratorio

FSM

Una máquina de estado (FSM) es un modelo matemático que describe el comportamiento de un sistema en términos de una serie de estados y transiciones. Los aspectos más importantes de una máquina de estado son los siguientes: los estados que representan las condiciones o modos del sistema, las transiciones que ocurren entre los estados en respuesta a eventos externos, las funciones de transición que definen las condiciones para el cambio de estado, y la capacidad de modelar y controlar sistemas complejos. Las máquinas de estado son ampliamente utilizadas en la programación de controladores lógicos programables, la implementación de protocolos de comunicación y la simulación de sistemas complejos. [3]

Interrupción Timer

La interrupción Timer se puede usar para generar interrupciones periódicas en función del valor del temporizador. El temporizador puede funcionar en modo de cuenta ascendente o descendente, y se puede configurar para generar una interrupción cuando el temporizador alcanza un valor determinado. El timer puede utilizarse para realizar tareas periódicas, como actualizar una pantalla LCD, leer sensores o enviar datos seriales a través de un puerto. [3]

Interrupción externa

Una interrupción externa permite que el microcontrolador sea interrumpido por una señal externa en algún pin IO del microcontrolador. Esta señal externa puede ser una señal digital, como un pulso de un sensor o un botón, que causa que el microcontrolador detenga la ejecución del programa principal y ejecute una función. [3]

3. Desarrollo y análisis

3.1. Programa

Código del programa

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define OCR1A_VALUE 48
#define ESPERAR_USUARIO 0
#define LAVAR 1
#define Hz60_FLAG 0
#define START_STOP_FLAG 1
#define TIMER_FLAG 2
#define duty_cycle 9
```

```
#define CARGA_1 0
#define CARGA_2 1
#define CARGA_3 2
#define LAVADO_1 3
#define LAVADO_2 4
#define LAVADO_3 5
#define LAVADO_4 6
#define START_STOP 7



uint8_t counter_60Hz = 0;
uint8_t counter_1Hz = 0;
uint8_t flag = 0;
uint8_t LED1 = 0;
uint8_t LED2 = 0;
uint8_t nivel_de_carga = 0;
uint8_t estado_de_lavado = 0;
uint8_t estado = 0;
uint8_t botones_LED = 0;



void send_byte(uint8_t cod) {

        for (unsigned int c = 0 ; c < 8 ; c++) {
                PORTA = (PORTA & 0xFB) | ((cod&0x01) << 2);
```

```
                PORTA ^= (1 << PA1);
                PORTA ^= (1 << PA1);
                cod>>=1;
        }
        PORTA ^= (1 << PA0);
        PORTA ^= (1 << PA0);
}

void add_time() {
        if (nivel_de_carga == 1) {
                if (estado_de_lavado == 1) {
                        LED2=1;
                        botones_LED = (botones_LED & 0x87) | (0x08);
                } else if (estado_de_lavado == 2) {
                        LED2=3;
                        botones_LED = (botones_LED & 0x87) | (0x10);
                } else if (estado_de_lavado == 3) {
                        LED2=2;
                        botones_LED = (botones_LED & 0x87) | (0x20);
                } else if (estado_de_lavado == 4) {
                        LED2=3;
                        botones_LED = (botones_LED & 0x87) | (0x40);
                } else {
                        botones_LED = (botones_LED & 0x87) | (0x00);
                }
        } else if (nivel_de_carga == 2) {
                if (estado_de_lavado == 1) {
                        LED2=2;
```

```
                botones_LED = (botones_LED & 0x87) | (0x08);
        } else if (estado_de_lavado == 2) {
                LED2=7;
                botones_LED = (botones_LED & 0x87) | (0x10);
        } else if (estado_de_lavado == 3) {
                LED2=4;
                botones_LED = (botones_LED & 0x87) | (0x20);
        } else if (estado_de_lavado == 4) {
                LED2=6;
                botones_LED = (botones_LED & 0x87) | (0x40);
        } else {
                botones_LED = (botones_LED & 0x87) | (0x00);
        }
} else if (nivel_de_carga == 3) {
        if (estado_de_lavado == 1) {
                LED2=3;
                botones_LED = (botones_LED & 0x87) | (0x08);
        } else if (estado_de_lavado == 2) {
                LED1=1;
                botones_LED = (botones_LED & 0x87) | (0x10);
        } else if (estado_de_lavado == 3) {
                LED2=5;
                botones_LED = (botones_LED & 0x87) | (0x20);
        } else if (estado_de_lavado == 4) {
                LED2=9;
                botones_LED = (botones_LED & 0x87) | (0x40);
        } else {
                botones_LED = (botones_LED & 0x87) | (0x00);
```

```
                }
        }
}

void dec_value() {
        if ( ((flag&(1<<START_STOP_FLAG))>0) ) {
                if (LED2 == 0) {
                        if (LED1 == 0) {
                                flag |= (1<<TIMER_FLAG);
                                return;
                        } else {
                                LED1--;
                                LED2=9;
                        }
                } else {
                        LED2--;
                }
        }
}

void display_LED() {

        if ((flag&(1<<Hz60_FLAG)) == 0x00) {
                PORTD &= ~(1 << PD0);
                if (LED1 == 0x00) {
                        PORTB = (0b01111110);
                } else if (LED1 == 0x01) {
                        PORTB = (0b00110000);
```

```
        } else if (LED1 == 0x02) {
                PORTB = (0b01101101);
        } else if (LED1 == 0x03) {
                PORTB = (0b01111001);
        } else if (LED1 == 0x04) {
                PORTB = (0b00110011);
        } else if (LED1 == 0x05) {
                PORTB = (0b01011011);
        } else if (LED1 == 0x06) {
                PORTB = (0b01011111);
        } else if (LED1 == 0x07) {
                PORTB = (0b01110000);
        } else if (LED1 == 0x08) {
                PORTB = (0b01111111);
        } else if (LED1 == 0x09) {
                PORTB = (0b01110011);
        }
} else if ((flag&(1<<Hz60_FLAG)) > 0) {
        PORTD &= ~(1 << PD1);
        if (LED2 == 0x00) {
                PORTB = (0b01111110);
        } else if (LED2 == 0x01) {
                PORTB = (0b00110000);
        } else if (LED2 == 0x02) {
                PORTB = (0b01101101);
        } else if (LED2 == 0x03) {
                PORTB = (0b01111001);
        } else if (LED2 == 0x04) {
```

```
                PORTB = (0b00110011);
            } else if (LED2 == 0x05) {
                PORTB = (0b01011011);
            } else if (LED2 == 0x06) {
                PORTB = (0b01011111);
            } else if (LED2 == 0x07) {
                PORTB = (0b01110000);
            } else if (LED2 == 0x08) {
                PORTB = (0b01111111);
            } else if (LED2 == 0x09) {
                PORTB = (0b01110011);
            }
        }

}

int main(void)
{
    MCUCR = 0x03;                                   // External Interrupt INT0
    GIMSK = 0x40;

    DDRA |= (1 << PA2) | (1 << PA1) | (1 << PA0);
    DDRB |= 0xFF;                                   // Set PB0...PB6 as output
    DDRD |= (1 << PD0) | (1 << PD1);                // Set PD0,PD1 as output
    //PORTB |= (1 << PB1);

    TCCR1B |= (1 << WGM12);                         // Set Timer1 to CTC mode
    TCCR1B |= (1 << CS11) | (1 << CS10);            // Set prescaler to 64
```

```
OCR1A = OCR1A_VALUE;                    // Set OCR1A value for 1 seco
TIMSK |= (1 << OCIE1A);                 // Enable Timer1 compare matc

sei();                                  // Enable global interrupts

while(1)
{
                                        // Main loop

    if (estado == ESPERAR_USUARIO) {
            if ( ((flag&(1<<START_STOP_FLAG))>0) && nivel_de_carga != 0)
                estado = LAVAR;
            }

    } else if (estado == LAVAR) {
            if ( (estado_de_lavado == 5) ) {
                flag &= ~(1<<START_STOP_FLAG);
                estado_de_lavado=0;
                LED1=0;
                LED2=0;
                botones_LED = (botones_LED & 0x07) | (0x00);
                estado = ESPERAR_USUARIO;
            } else if (((flag&(1<<TIMER_FLAG))>0)){
                estado_de_lavado++;
                counter_1Hz=0;
                add_time();
                flag &= ~(1<<TIMER_FLAG);
            }
```

```
        }
    }
}


                                                    // INTERRUPCIONES
ISR(TIMER1_COMPA_vect)
{
    counter_60Hz++;
    if (counter_60Hz == duty_cycle) {
        PORTB = 0x00;
        PORTD |= (1 << PD0) | (1 << PD1);
    } else if (counter_60Hz == 10) {
        counter_1Hz++;
        if (counter_1Hz == 255) {
            dec_value();
        }
        display_LED();
        send_byte(botones_LED);
        counter_60Hz = 1;
        flag ^= (1<<Hz60_FLAG);
    }
}

ISR( INT0_vect )
{
        if ( ((PIND >> 3) & 0x03) != 0 ) {
                if (estado == ESPERAR_USUARIO) {
```

```
                        nivel_de_carga = (PIND >> 3) & 0x03;
                        botones_LED = (botones_LED & 0xF8) | (1<<(nivel_de_ca
                }
        } else {
                flag ^= (1<<START_STOP_FLAG);
                botones_LED = (botones_LED & 0x7F) | ((flag&(1<<START_STOP_FL
        }
}
```

Descripción del programa

El programa está estructurado principalmente alrededor de una FSM implementado en el main() y la interrupción timer1.

Se usa la interrupción timer1 para generar una base de tiempos de 127 Hz (para uso de los LEDS de 7 segmentos) y otra de 1Hz (para la cuenta regresiva del tiempo restante de cada estado de lavado).

Cada estado de la FSM se encarga de activar ciertas salidas, y sus entradas son capturadas por la interrupción externa INT0 o por alguna condición alcanzada en la interrupción Timer1.

La máquina de estados consiste solamente de 2 estados:

1. ESPERAR_USUARIO
2. LAVAR

El estado inicial es ESPERAR_USUARIO. Este estado no tiene salidas, pero si espera a que 2 condiciones se cumplan de entradas para transicionar al segundo estado; Que el usuario seleccione uno de los niveles de carga y que presione el botón de inicio.

El estado LAVAR se encarga de esperar que todos los estados de lavados se completen. Este recibe como entrada el botón de inicio/pausa y se encarga de detener la cuenta regresiva del tiempo restante en caso de que llegue la señal de pausa. Si se completan todos los estados de lavado, el estado transiciona al estado inicial ESPERAR_USUARIO.

Adicionalmente se utilizan las siguientes funciones auxiliares:

- send_byte()
- add_time()
- dec_value()
- display_LED()

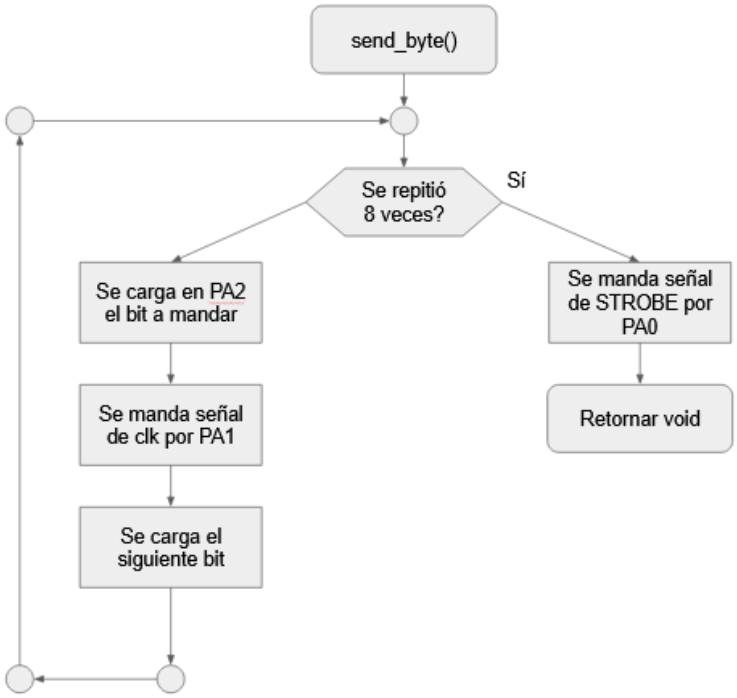
send_byte() se encarga de mandar serialmente la información de los LEDS relacionados a los botones al registro desplazante por medio de los pines A0...A2.

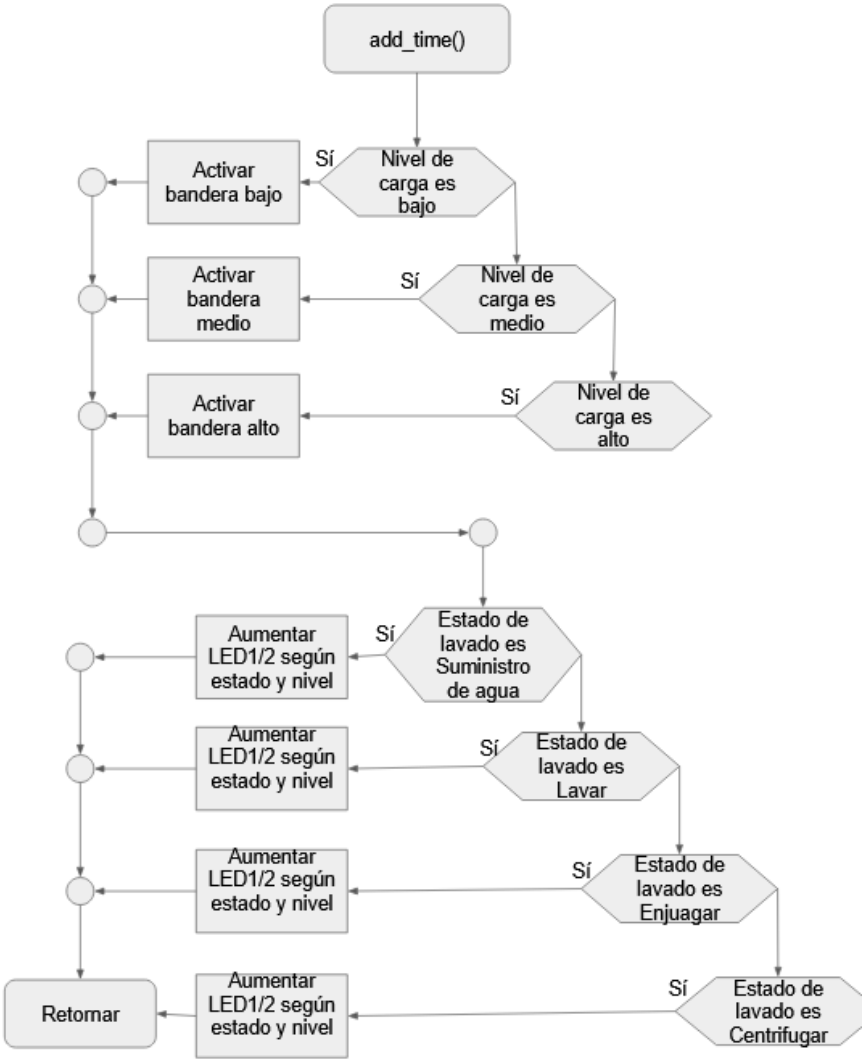
add_time() se encarga de determinar, en base al nivel de carga ingresado por el usuario y el estado de lavado actual, cuánto tiempo hay que agregarle a los timers para el siguiente estado de lavado.

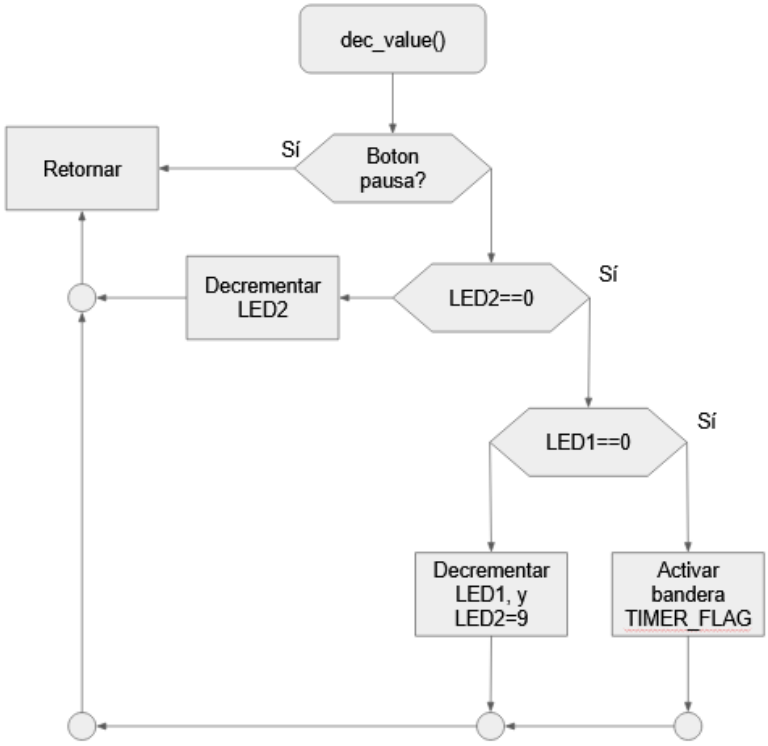
dec_time() se encarga de realizar una resta en BCD a las variables que contienen el tiempo restante del estado actual de lavado.

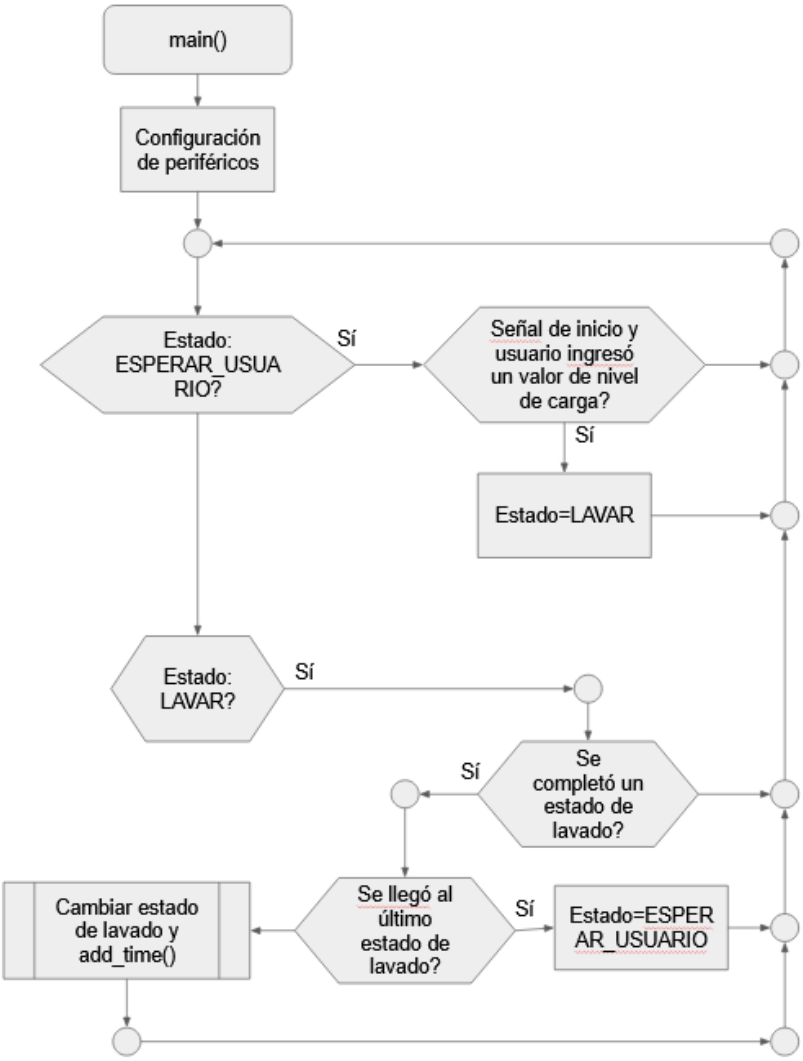
display_LED() se encarga de mandar la codificación en 7 segmentos correcta a los displays según los valores almacenados en las variables que contienen el tiempo restante en BCD.

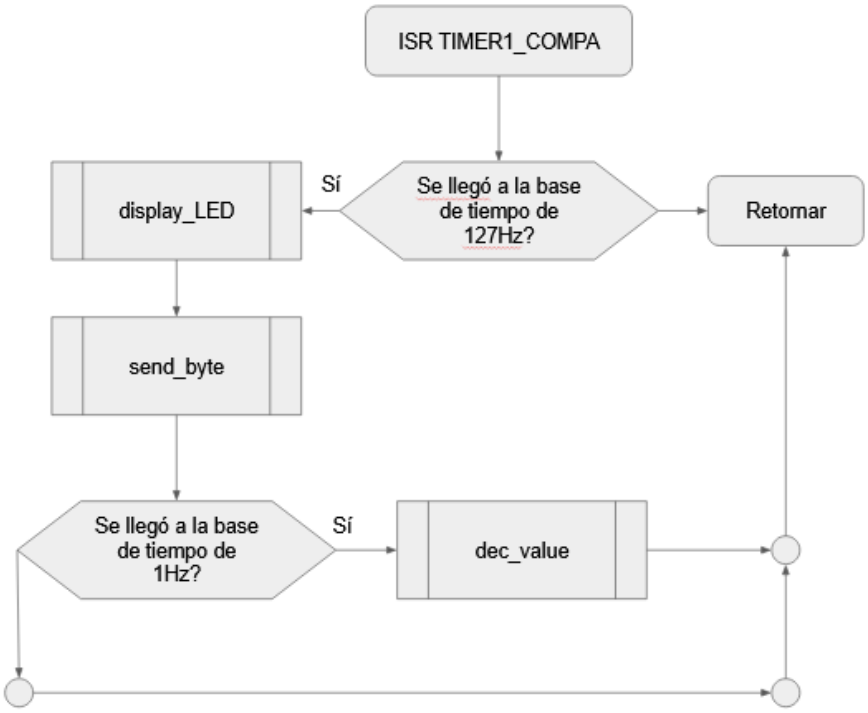
Diagrama de bloques

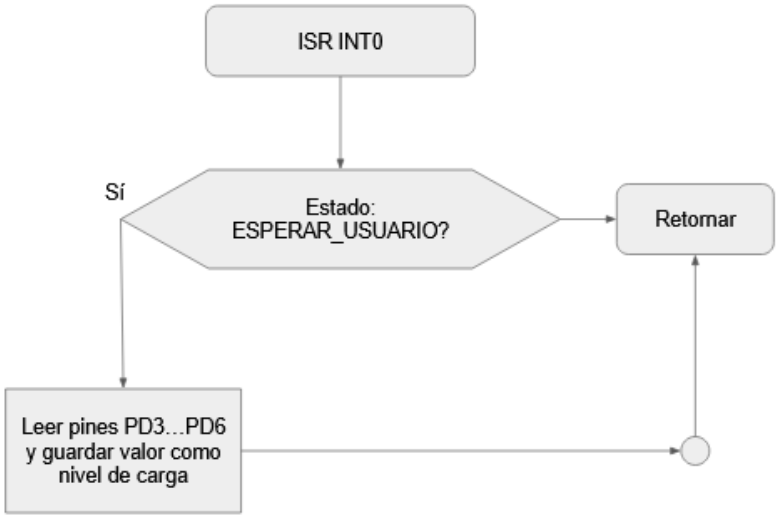



add_time()
Nivel de carga es bajo
Sí
Activar bandera bajo
Nivel de carga es medio
Sí
Activar bandera medio
Nivel de carga es alto
Sí
Activar bandera alto
Estado de lavado es Suministro de agua
Sí
Aumentar LED1/2 según estado y nivel
Estado de lavado es Lavar
Sí
Aumentar LED1/2 según estado y nivel
Estado de lavado es Enjuagar
Sí
Aumentar LED1/2 según estado y nivel
Estado de lavado es Centrifugar
Sí
Aumentar LED1/2 según estado y nivel
Retornar


dec_value()
Boton pausa?
Sí
Retornar
LED2==0
Sí
Decrementar LED2
LED1==0
Sí
Decrementar LED1, y LED2=9
Activar bandera TIMER_FLAG


main()
Configuración de periféricos
Estado: ESPERAR_USUARIO?
Sí
Señal de inicio y usuario ingresó un valor de nivel de carga?
Sí
Estado=LAVAR
Estado: LAVAR?
Sí
Se completó un estado de lavado?
Sí
Se llegó al último estado de lavado?
Sí
Estado=ESPERAR_USUARIO
Cambiar estado de lavado y add_time()

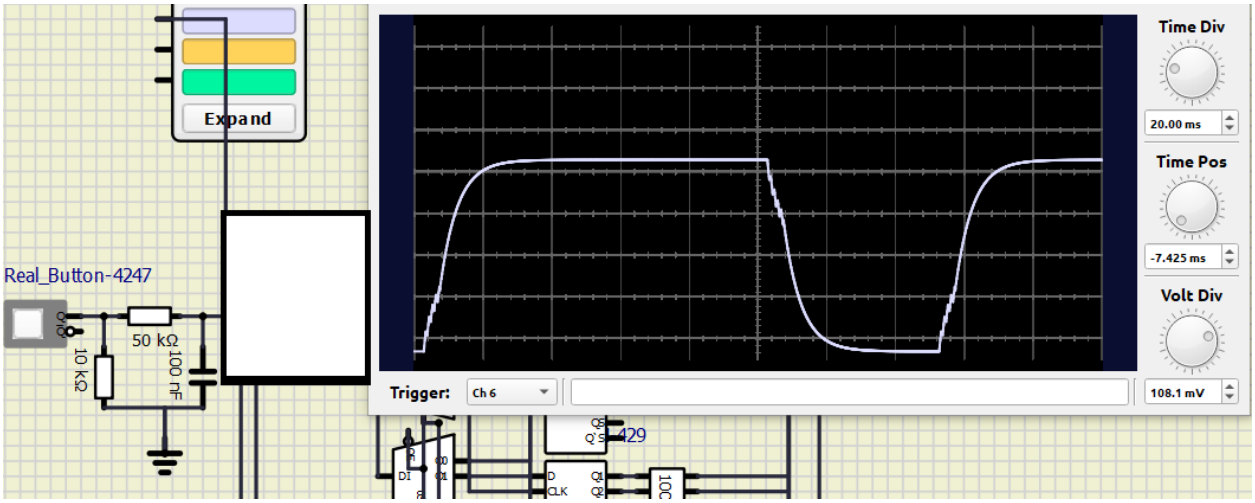

ISR TIMER1_COMPA
Se llegó a la base de tiempo de 127Hz?
Sí
display_LED
Retornar
send_byte
Se llegó a la base de tiempo de 1Hz?
Sí
dec_value


ISR INT0
Estado: ESPERAR_USUARIO?
Sí
Retornar
Leer pines PD3...PD6 y guardar valor como nivel de carga

3.2. Componentes

Supresor de rebotes

Según los valores escogidos de resistencia y capacitancia y según lo mencionado en la nota teórica del diseño del circuito, el supresor de rebotes debería de causar un tiempo de activación y desactivación de aproximadamente 24 ms.

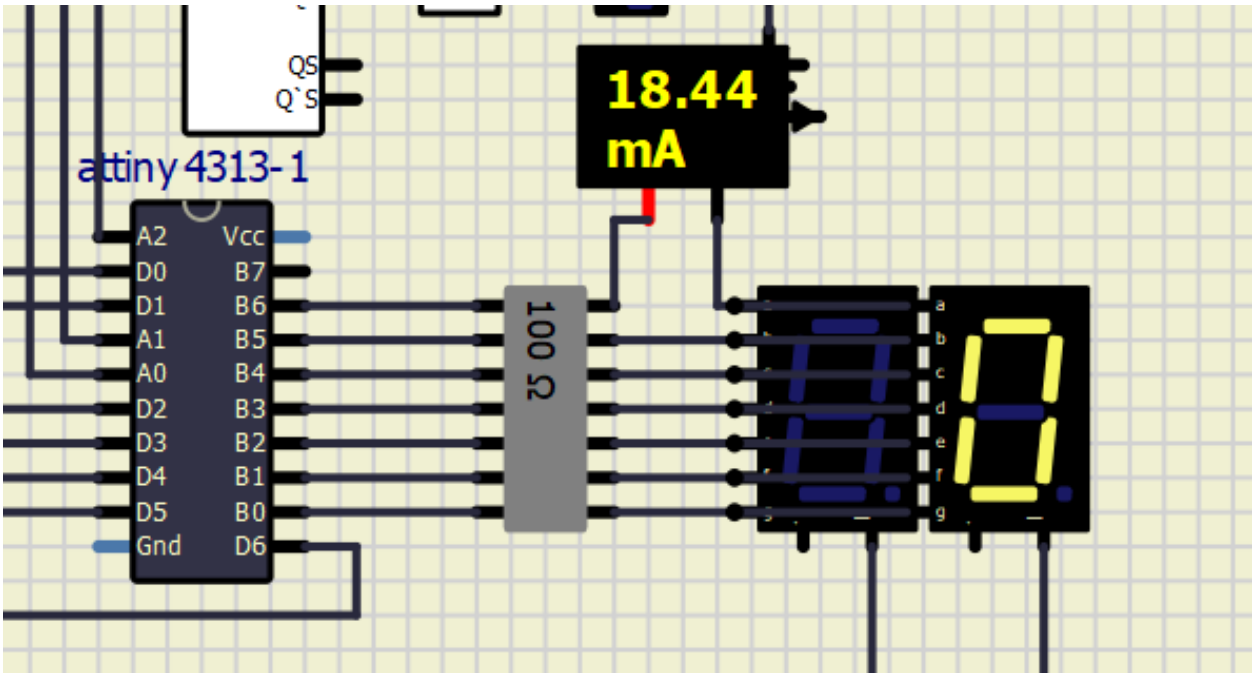


Respuesta supresor de botones 3.1: Obtenido del simulador

Como se observa de la imágen 3.1, en el analizador digital, a una resolución 20ms por división del gráfico, el tiempo de carga y descarga del capacitor es aproximadamente 20 ms. Este valor es el esperado.

LEDS 7 segmentos

Según los valores escogidos de resistencias de acuerdo con lo mencionado en la nota teórica del diseño del circuito, la corriente pasando por los LEDS debería ser marginalmente menor a 20 mA.

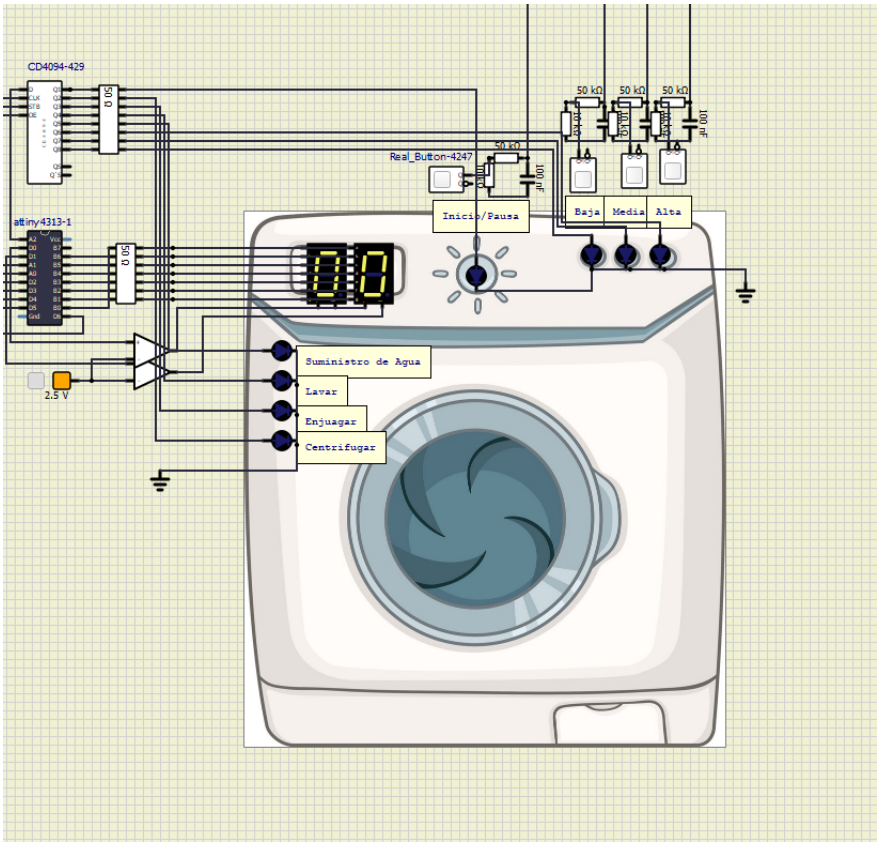


Corriente LEDS 3.2: Obtenido del simulador

Como se observa de la imágen 3.2, la corriente es de 18.44 mA, un valor esperado.

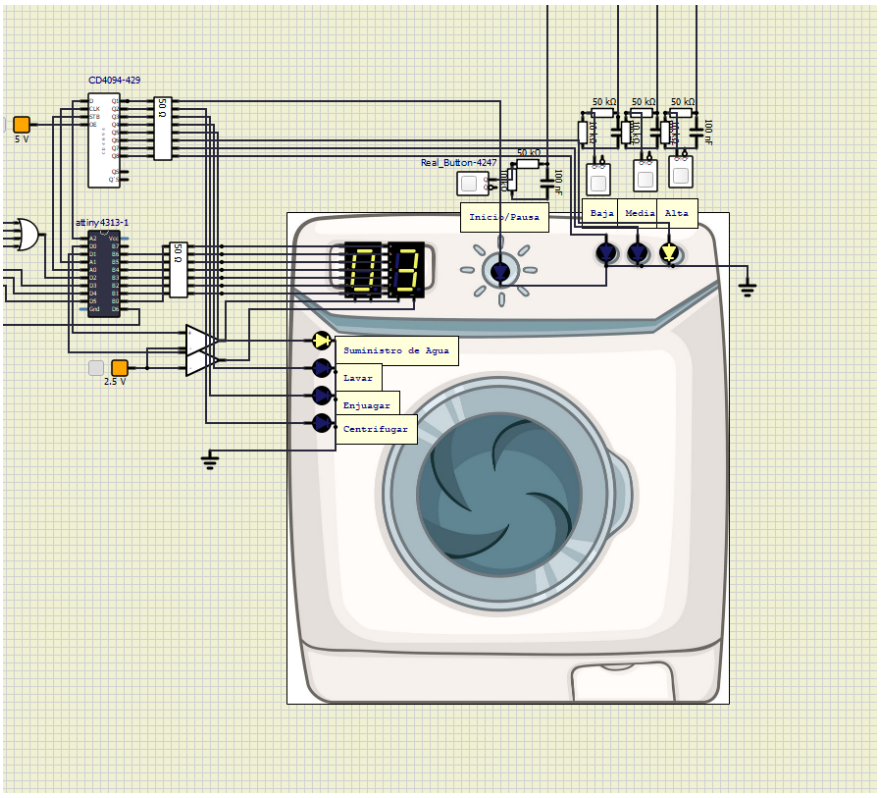
3.3. Funcionamiento

Esperar usuario



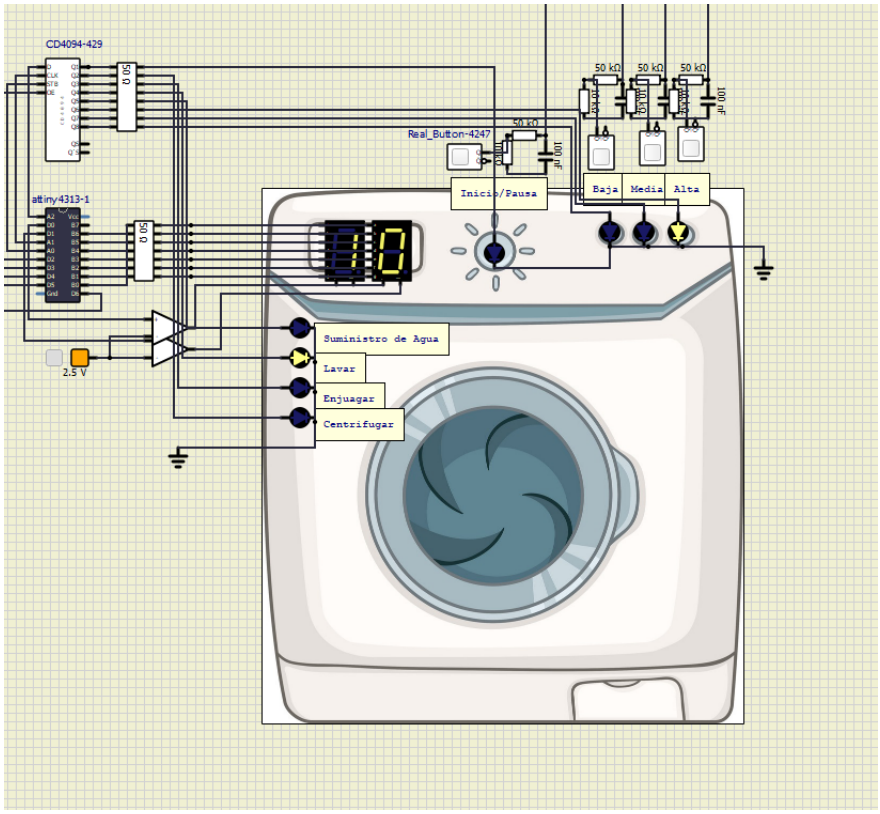
Esperar Usuario 3.3: Obtenido de simulador

Suministro de agua



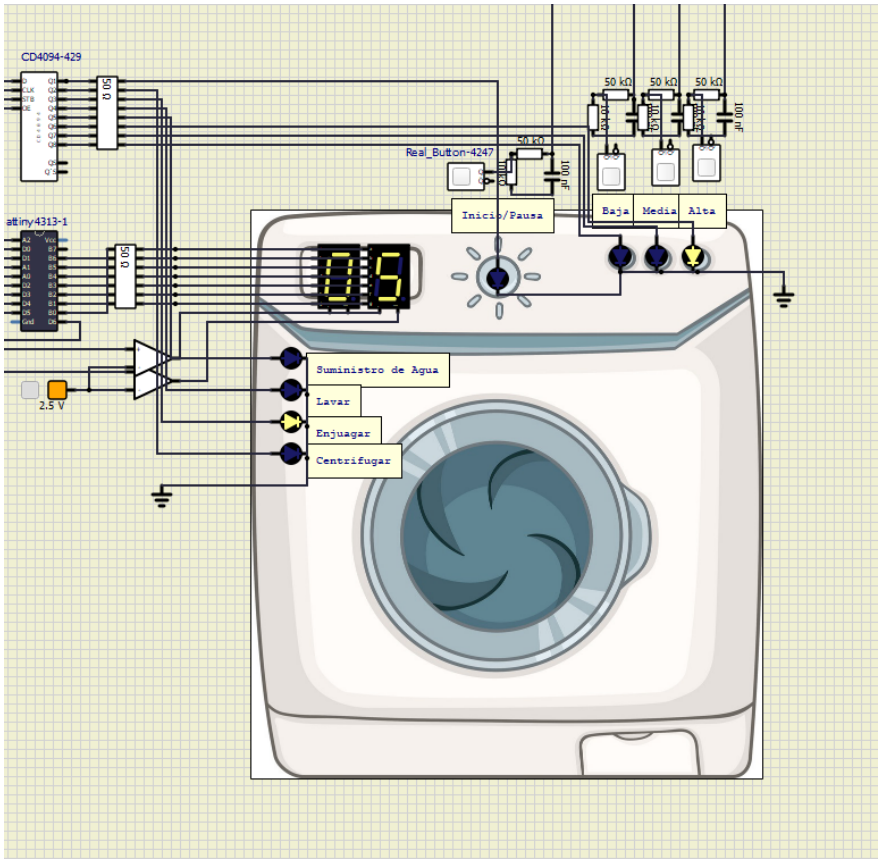
Suministro de agua 3.4: Obtenido de simulador

Lavar



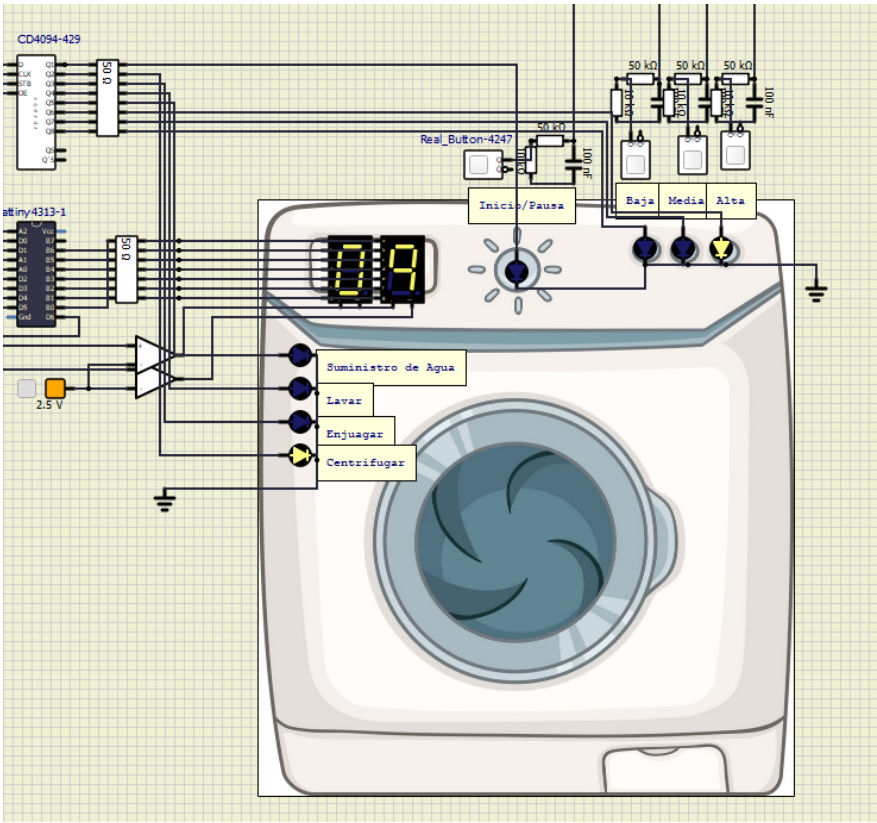
Lavar 3.5: Obtenido de simulador

Enjuagar



Enjuagar 3.6: Obtenido de simulador

Centrifugar



Centrifugar 3.7: Obtenido de simulador

4. Conclusiones

Se concluyé que la solución propuesta para este segundo laboratorio cumple con los requerimientos del enunciado y se comporta según el diseño teórico realizado.

4.1. Recomendaciones

Un cambio que se puede realizar a la solución propuesta es, en vez de utilizar una solo interrupción externa, usar las 4 interrupciones externas que proporciona el microcontrolador. El efecto de esto sería que se podría no usar un decodificador para saber cual de los botones fue presionado. Al usar las 4 interrupciones, se sabría cual botón fue presionado basado en

cual de las interrupciones fue activada. Esta opción tiene la siguiente invonveniencia: Solo las interrupciones externas INT0 y INT1 tienen la opción de ser configuradas para disparar cuando detecta un nivel alto, lo cual es conveniente cuando se desea que la señal de un botón presionandose solo cause una interrupción en vez de 2 interrupciones. Al usar las 4 interrupciones y por ende las interrupciones PCI, habría que diferenciar entre el botón siendo presionado y el botón siendo liberado por software.

Bibliografía

[1] "Attiny4313 data sheet." https://mv1.mediacionvirtual.ucr.ac.cr/pluginfile.php/2544747/mod_resource/content/1/ATtiny2313.pdf, 2007. (Accessed on 21/04/2023).

[2] "Componentes electrónicos." https://www.digikey.com/en/products/, 2005. (Accessed on 21/04/2023).

[3] "Microcontroladores." https://en.wikipedia.org/wiki/Microcontroller, 2006. (Accessed on 21/04/2023).

5. GIT

https://github.com/JAR1224/Laboratorio_2_Jose_Ramos