UNIVERSIDAD NACIONAL DE LOJA

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

DISEÑO DIGITAL.

Tema: “Human Reaction Time ”

DOCENTE:

Ing. Freddy Ganazhapa.

AUTOR:

-Deiby Patricio Calva.

Periodo Académico

Octubre 2019-Marzo 2020

1. Introducción

En el presente documento se hace una descripción sobre el funcionamiento de un pequeño programa realizado en Atmel Studio que mide el tiempo de reacción de una persona al pulsar un botón. En este proceso se utiliza la placa **atmega** 2560, que a través de la configuración de sus puertos se lo va a implementar en un protoboard con otros componentes para hacer funcionar el ejercicio del tiempo de reacción.

2. Objetivos

- Desarrollar un dispositivo que permita medir el tiempo de reacción de una persona al pulsar un botón, utilizado la placa atmega 2560 y regístralo a través del monitor serial, empleando un Protoboard y componentes electrónicos adecuados para diseñar el dispositivo que permita medir el tiempo de reacción.
- Utilizar Atmel Studio para programar las funciones necesarias, los pines como entrada y salida, led y buzzer para el desarrollo y funcionamiento de la actividad de tiempo de reacción.
- Comparar la velocidad del tiempo de reacción en el que un usuario pulsa un botón guiándose solo por el sonido frente a la velocidad de reacción cuando el mismo usuario pulsa el botón guiándose por la luz de un led.

3. Materiales usados

Materiales y Reactivos	Equipos y herramientas
- 1 diodo emisor de luz de 5 mm (led) - 3 resistores de 330 Ω, ¼ de W - 1 pushbutton - Cables dupont de 20cm - 1 Transistor 2N3906 - 1 buzzer - 1 pulsador - 1 Transistor 2N3904	- 1 Arduino Atmega 2560 - 1 Protoboard

4. Actividades por desarrollar

Producir un dispositivo de medición del tiempo de reacción con la placa atmega2560 que le indica al usuario que encienda una luz y luego registre el tiempo que le toma presionar un botón. El tiempo de reacción del usuario se mostrará en una computadora usando el monitor en serie y el tiempo mínimo medido registrado en eeprom y así mismo mostrar la simulación realizada en proteus**.**

5. Marco Teórico

3.1 Conceptos

Atmel Sudio: es la plataforma de desarrollo integrado (IDP) para desarrollar y depurar todas las aplicaciones de microcontroladores AVR® y SAM. El IDP de Atmel Studio brinda un entorno transparente y fácil de usar para escribir, compilar y depurar sus aplicaciones escritas en C / C ++ o código de ensamblaje. También se conecta sin problemas a los depuradores, programadores y kits de desarrollo que admiten dispositivos AVR ® y SAM. (MicroChip, s.f.)

ARDUINO MEGA 2560: es una placa de desarrollo basada en el microcontrolador ATmega2560. Tiene 54 entradas/salidas digitales (de las cuales 15 pueden ser usadas como salidas PWM), 16 entradas analógicas, 4 UARTs, un cristal de 16Mhz, conexión USB, jack para alimentación DC, conector ICSP, y un botón de reseteo. La placa Mega 2560 es compatible con la mayoría de shields compatibles para Arduino UNO. (cl, s.f.)

Directiva #define: La directiva define se usa para definir un identificador y una cadena que el compilador sustituirá por el identificador cada vez que se encuentre en el archivo fuente. (Enrique Vicente Bonet Esteban)

Directiva #include: La directiva #include fuerza al compilador a incluir otro archivo fuente en el archivo que tiene la directiva #include, y a compilarlo. El nombre del archivo fuente a incluir se colocará entre comillas dobles o entre paréntesis de ángulo.

(Enrique Vicente Bonet Esteban)

#include: contiene los prototipos de las funciones, macros, y tipos para manipular datos de entrada y salida. (José Norbey Sánchez F.)

#include <avr/io.h>: Este archivo de encabezado incluye las definiciones de E / S apropiadas para el dispositivo que ha sido especificado por el -mmcu=interruptor de línea de comandos del compilador. (MicroChip, s.f.)

#include <inttypes.h>: proporciona características que mejoran la funcionalidad de los tipos definidos en <stdint.h>encabezado. Se incluyen las macros que definen la cadena de formato printf y los especificadores de cadena de formato scanf correspondientes a los <stdint.h>tipos y varias funciones para trabajar con intmax_ty uintmax_ttipos.

#include <avr / eeprom.h>: Este archivo de encabezado declara la interfaz a algunas rutinas de biblioteca simples adecuadas para manejar los datos EEPROM contenidos en los microcontroladores AVR. La implementación utiliza una interfaz de modo de sondeo simple. Las aplicaciones que requieran acceso EEPROM controlado por interrupción para garantizar que no se pierda tiempo en spinloops tendrán que implementar su propia implementación.

La macro FDEV_SETUP_STREAM: se puede usar para configurar un búfer que es válido para las operaciones de stdio. El buffer inicializado será de tipo FILE. Puede definir buffers separados para entrada y salida. Alternativamente, puede definir solo un búfer que funcione tanto para la entrada como para la salida. El primer y segundo parámetro son los nombres de las funciones que se llamarán cuando los datos se lean o se escriban en el búfer. (https://appelsiini.net, 2011)

DDRD: es el registro de dirección para el Puerto D (pines digitales Arduino 0-7). Los bits en este registro controlan si los pines en PORTD están configurados como entradas o salidas. (Arduino, 2019)

DDRC: El registro de dirección de datos del puerto C - lectura / escritura.

El temporizador / contador OCR0A: comienza en cero con el pin OC0A en un estado. Cuando el temporizador cuenta hasta el punto donde el temporizador tiene el mismo valor que OCR0A, el hardware cambia automáticamente el estado del pin OC0A. El temporizador continúa contando sin cambiar el pin OC0A. Cuando el temporizador se desplaza desde su valor máximo a 0x00, entonces el pin OC0A se restablece a su valor original. (Bill Earl, s.f.)

TIMSK0: este registro, se utiliza para activar / desactivar las interrupciones relacionadas con los temporizadores, controla las interrupciones de los tres temporizadores. BIT0 (primer bit desde la derecha) controla las interrupciones de desbordamiento de TIMER0. Hay que tener en cuenta que TIMER0 tiene una interrupción, y el resto de los bits son para otros contadores.

TCNT0: Este es el contador real en el contador TIMER0. El reloj del temporizador cuenta este registro como 1, es decir, el reloj del temporizador aumenta el valor de este registro de 8 bits en 1 con cada pulso de reloj del temporizador. El reloj temporizador puede ser definido por el registro TCCRO.

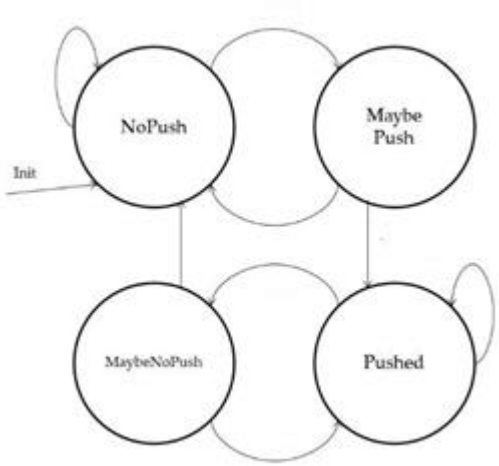
TCCR0: En este registro (utilizado para con figurar el temporizador), hay 8 bits, pero solo se utilizan los últimos 3 bits CS02, CS01 y CS00. Estos son los bits CLOCK SELECT que se usan para configurar el prescaler.

ISR (Rutina de servicio de interrupción): es una función que la CPU debe ejecutar cada vez que se produce una interrupción. (Aspencore, 2019)

PORTC: El puerto C se inicializa utilizando el registro DDRx. Los bits a 1 corresponden a los pines de salida es decir se pone en ALTO (1) o HIGH, lo que da un voltaje de salida de VCC en ese pin.

WGM (Modos de generación de forma de onda): en arduino, hay dos tipos principales de modos de generación de forma de onda, el PWM rápido y el PWM de fase correcta. La fase correcta PWM crea el pulso en el punto medio del período. Con el modo de fase correcta, el temporizador Arduino cuenta hacia arriba y luego cuenta hacia atrás. Al comienzo del período (cuando el conteo del temporizador es 0), el tiempo cuenta hasta un número establecido que almacena en el registro de Arduino (ICR1).

Máquina de estado: Las máquinas de estado son una parte integral de la programación de software. Las máquinas de estado hacen al código más eficiente, más fácil de depurar y ayudan a organizar el flujo del programa. La primera ventaja de utilizar las máquinas de estado, es que promueve buenas técnicas de diseño de firmware. En este caso usamos: Una Máquina de Estado Finita (FSM = Finite State Machine) está basada en la idea de que hay un número finito de estados para un sistema determinado.



Las máquinas de estado requieren una Variable de Estado (State Variable - SV). La variable de estado es un apuntador que mantiene un control del estado en que se encuentra el microcontrolador y dirige el flujo del programa al módulo de software correspondiente. La variable de estado puede modificarse en los módulos (o estados) de software por si misma o por una función externa.

3.1 Desarrollo de la aplicación

Para el desarrollo de la aplicación se utiliza el IDP **Atmel Studio 7.** En esta plataforma de desarrollo integrado es donde se creará el código que permite manipular los pines de la placa atmega2560. Se utiliza librerías que permite imprimir mensajes, en este caso se quiere conocer el tiempo que tarda una persona en pulsar el botón cuando se enciendo un led.

Para empezar a programar lo primero que se debe hacer es definir las variables que se va a emplear. También se debe activar los registros del contador timer 0 el cual permite establecerlo como contador. Para la actividad se empleara una frecuencia master de 16000000 Hz con un tiempo retarda de 1000 ms.

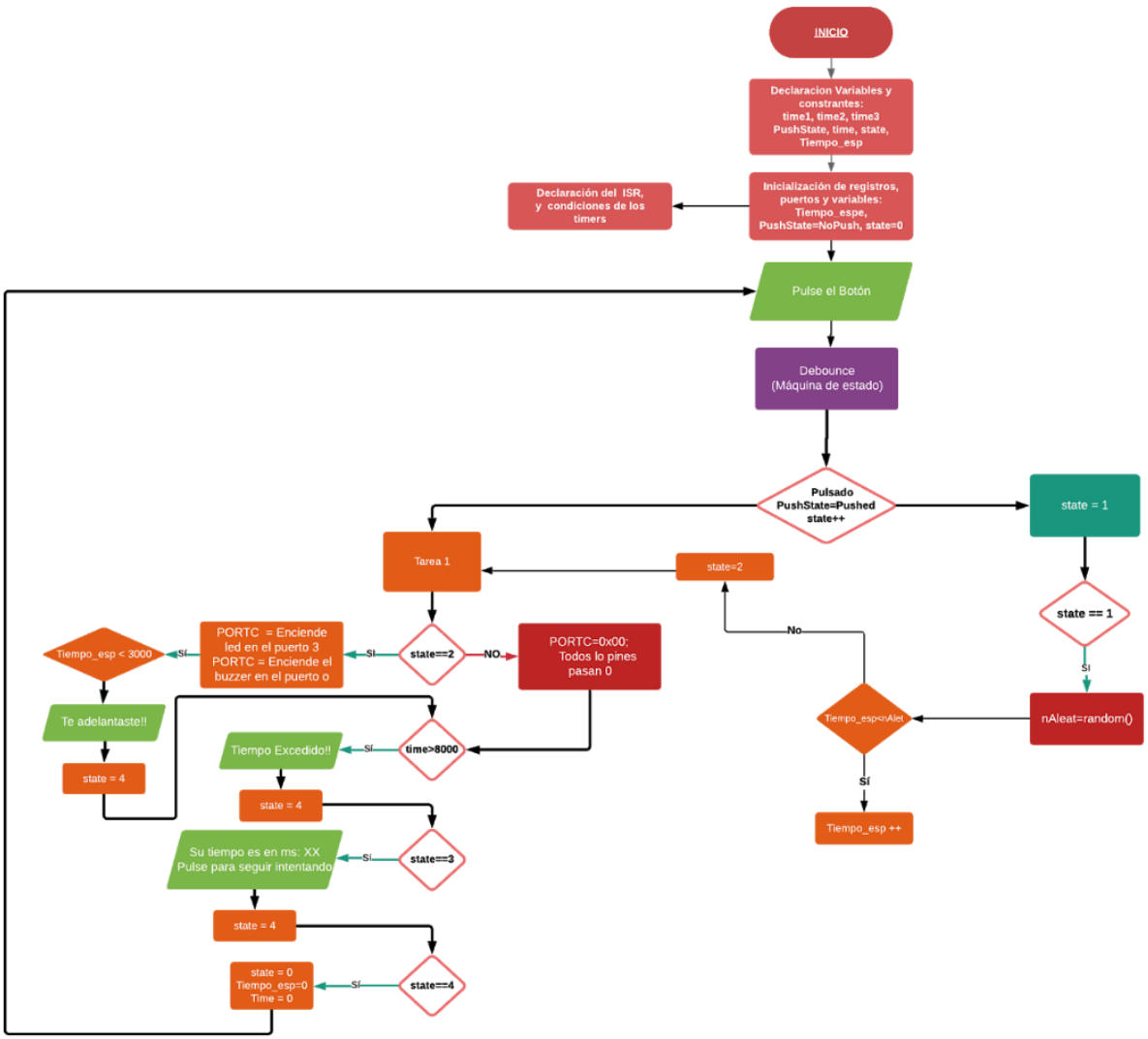
Se crearan en total tres tareas, las cuales cada una de ellas realizará funciones únicas. En la primera tarea se establecerán los parámetro para que el led se encienda y del mismo modo para que el buzzer empiece a sonar por ello se van a manipular los pines del puerto C como salida (0xff), de esta forma cuando un usuario pulse el botón, la señal será enviada a través del puerto D los cuales actúan (0x00) como entrada para que el buzzer y el led se enciendan. En esta tarea se incluirán también condiciones. Estas condiciones permitirán imprimir el resultado que queremos a través del monitor serial de la placa atmega 2560, en este caso el tiempo que tarda una persona en pulsar el botón cuando un led se enciende. También arrojaran mensaje de errores que provocarán que todos los contadores declarado se reseten a 0.

Estos casos ocurren cuando se pulsa el boton antes de que se encienda o después de que se apague el led. Así de esta forma si ocurren estos caso los contadores se vuelven a 0.Estos contadores se resetean cuando el contador state es igual a 4.

Sin embargo cuando se pulsa el botón mientras el led esta encendido se presentara el tiempo que se ha tardado en reaccionar al encenderse el led a través del monitor serial, y esto ocurre cuando el contado state es igual 3.

En la segunda tarea se desarrolla una máquina de estado la cual cuanta con un contador que se va incrementado desde que se pulsa el botón. Este contador llamado state ese inicia en 2, así que cuando se cumple el primer estado de la maquina este se incrementara para hacer cumplir todas la condiciones en la tare 1.

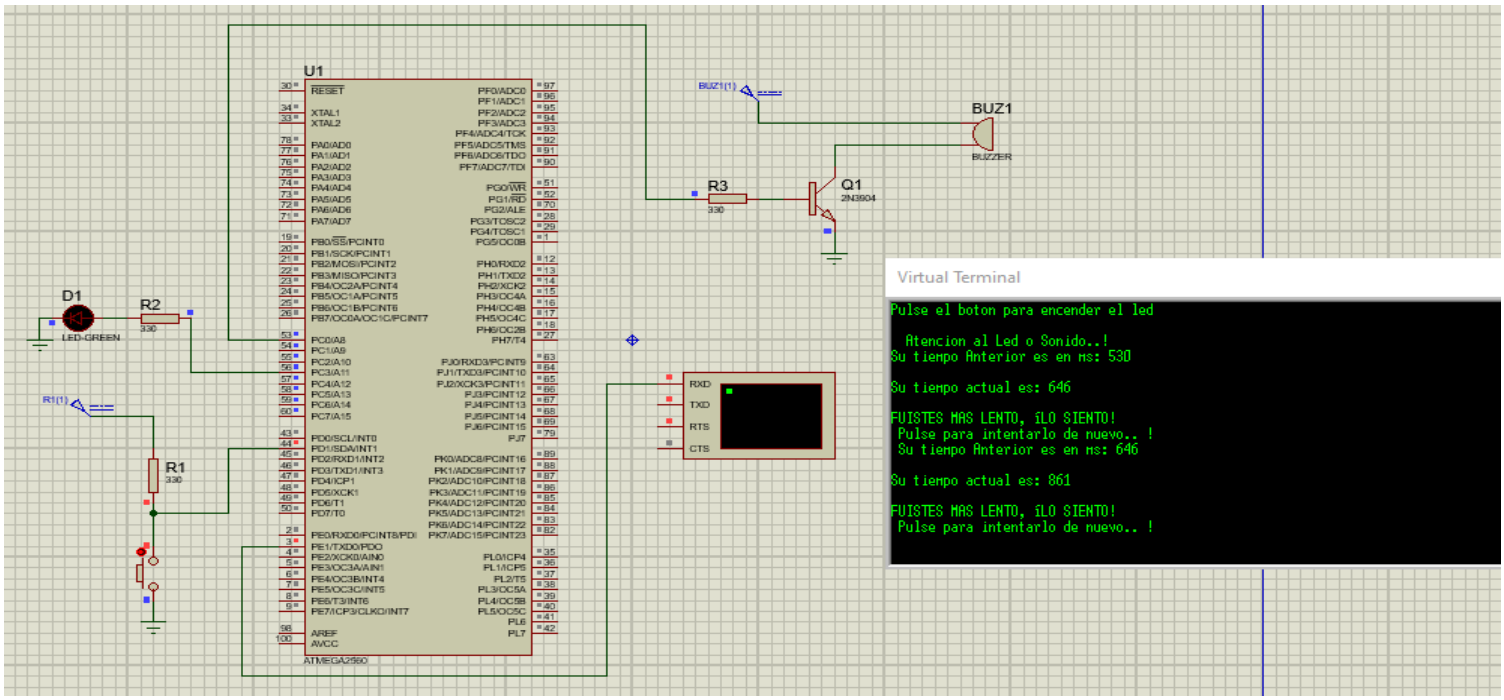
En la tarea tres, llamada task_delay se establece un tiempo de espera, es decir se establece el tiempo que debe tardar en encenderse el led después de pulsar el botón por primera vez. En esta tarea el tiempo se lo genera de forma aleatoria entre dos valores, básicamente entre 4000ms y 8000ms. Para establecer esto se usa la función radón(). Para que esta tarea se ejecute el state tiene que ser igual a 1, y esto se produce cuando se ejecuta una de las condiciones de la tarea 1, que todos los contadores a 0. De esta forma el contador state comienza a incrementarse en la máquina de estado hasta llegar al valor de 1. Cuando llega a este valor se va a generar una comparación mediante un if donde, si el tiempo de espera es menor al valor generado aleatoriamente, entonces el tiempo de espera se ira incrementado hasta ser igual al valor generado aleatoriamente, una vez cumplido esto el contador state se incrementara a 2 y ejecutara la tarea 1

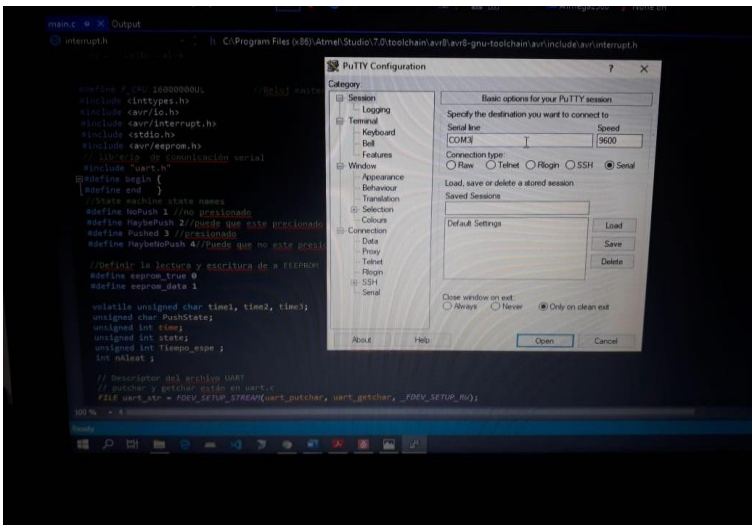


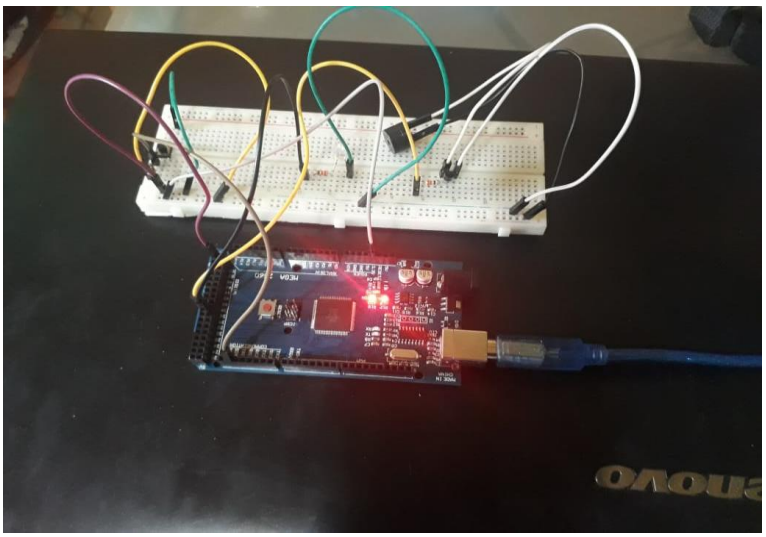
6. Resultados obtenidos

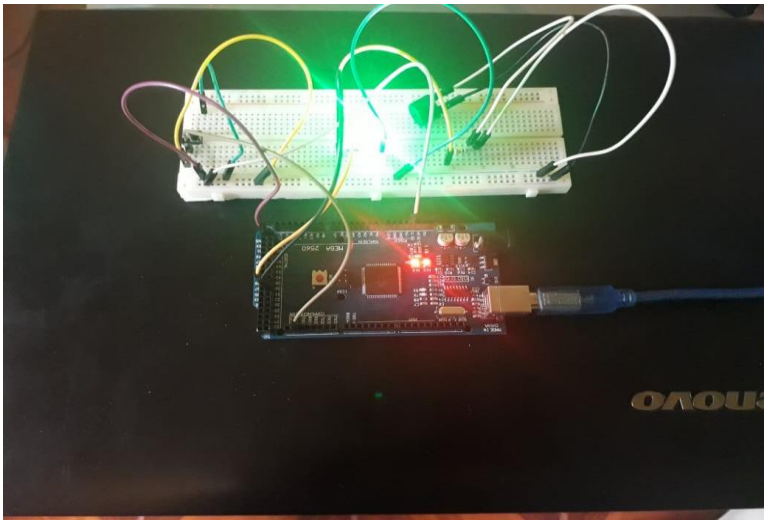
Tiempo de reacción con led	Tiempo de reacción con buzzer
350	224
375	202
295	162
194	193
219	202

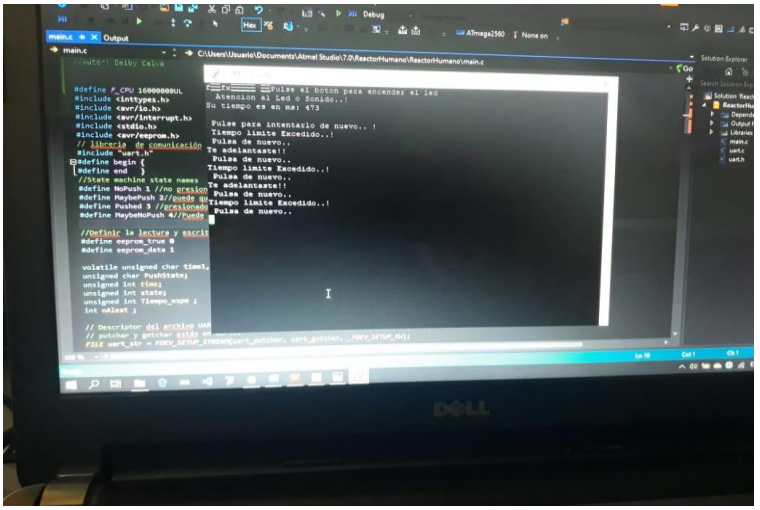
7. Imágenes











Link Archivo: https://github.com/DeibyCalva/HumanReactionTime/tree/master

Conclusiones

- En la presente actividad denominada tiempo de reacción, se pudo determinar que una persona pulsa el botón con mayor rapidez al escuchar el sonido producido por el buzzer, que al observar la luz que emite el LED.
- Atmel Studio es una plataforma que permite configurar o programar los pines de la placa atmega 2560 de forma sencilla sin utilizar largas líneas de código.

Recomendaciones

- Se recomienda tener en claro el funcionamiento del contador timer0, de todos los registro que lo componen y de cómo se empleen cada uno de estos para poder activar el temporizador.
- Es necesario revisar las hojas de especificaciones y el diagrama de pines del atmega2560 para evitar confusiones al momento de utilizar las diferentes entradas y/o salidas.

Preguntas de control

- **¿Cuál es su tiempo de reacción promedio?**

  Registros del tiempo de reacción en ms: 183; 207; 188; 247; 217; 246; 474; 250; 307; 236;

  $$\overline{x} = \frac{\sum fiXi}{n}$$

  $\sum fiXi$ =183+207+188+247+217+146+474+250+307+236=2455

  $\overline{x}$=$\frac{2455}{10}$=245,5

- **¿Su tiempo de reacción es más rápido viendo el LED sin sonido, o el sonido sin LED, o con ambos?**

  Registro tiempo de reacción con sonido: 224; 202; 162; 193; 202;
  Registro tiempo de reacción con led: 350; 375; 295; 194; 219;

  $$\overline{x} = \frac{\sum fiXi}{n}$$

  $\sum fiXi$ = 224+202+162+193+202=983

  $\overline{x}bu$=$\frac{983}{5}$=196,6

  $\sum fiXi$ = 350+375+295+194+219=1433

  $\overline{x}led$=$\frac{1433}{5}$=286,6

  El tiempo de reacción es más rápido solo escuchado el sonido sin observar el led

Una lista muy comentada de su código.

```
/*
 * HumanReactionTime.c
 *
 * Created: 28/1/2020 17:47:04
 * Author : Deiby Calva
 */


#define F_CPU 16000000UL   //Reloj master de 16000000
#include <inttypes.h>     //utiliza si quieres usar nombres de tipos
independientes de cpu como char(u_int8 es lo mismo que char)
#include <avr/io.h>      //nombres lógicos de cada nombre de pin (OCR0A, OCIE0A)
#include <avr/interrupt.h> //sólo dos macros para configurar ISR
#include <stdio.h>        //archivos de cabecera para las interfaces en serie,
getchar putchar
#include <avr/eeprom.h>
// libreria  de comunicación serial
#include "uart.h"  // Transmisor receptor asincrono universal, controla los
puertos y dispositivos en seri
#define begin {
#define end   }
//nombres de los estados de las máquinas de estado
#define NoPush 1 //no presionado
#define MaybePush 2//puede que este precionado
#define Pushed 3 //presionado
#define MaybeNoPush 4//Puede que no este presionadao

//Definir la lectura y escritura de a EEEPROM
#define eeprom_true 0
#define eeprom_data 1

volatile unsigned char time1, time2, time3;           //contadores de tiempo
//estas variables deben ser compartidas
unsigned char PushState;                                    //estado de la
maquina
unsigned int time;                                                  //timepo de
lectuura del boton  // un valor de milisegundos desde el último reinicio
unsigned int state;                                                 //comprobar
la maquina de estado
unsigned int Tiempo_espe ;                                  ///tiempo de retardo
int nAleat ;

// Descriptor del archivo UART
//La función getchar recibe un carácter, mientras que la función putchar imprime
un carácter.
//  Permiten manipular de distintas maneras archivos y caracteres.
// putchar y getchar están en uart.c
FILE uart_str = FDEV_SETUP_STREAM(uart_putchar, uart_getchar, _FDEV_SETUP_RW);

void initialize(void){
       //configurar los puertos
       DDRD = 0x00;               // todos los puertos DDRD como entrada
       DDRC = 0xff;               // todos los puertos DDRC como salida
       //OCR0A=16000000/(64*1000)-1=249
       //OCR0A: carga el valor hasta el cual se quiere que llegue el registro
TCNT0 el el modo CTN
       OCR0A = 249;
       TIMSK0= (1<<OCIE0A); //turn on timer 0 cmp match ISR
       //Envia al prescaler el valor de 64
```

```
	//pone a CS02=0    CS01=1    CS00=1  activa  los dos primeros bits del
registro TCCR0B
	TCCR0B= 3;                          //elige el presaclar a utilizar para
obtener en cuanto tiempo se quiere que el registro TCNT0
	//sea igual al registro OCR0A
	// enciende on clear-on-match
	TCCR0A= (1<<WGM01) ;
	Tiempo_espe=0;
	PushState = NoPush;//Empujar el estado.
	state = 0; //progama inicia en 0


	uart_init();
	stdout = stdin = stderr = &uart_str; //envias un mensaje a la puerta
serial
	fprintf(stdout,"Pulse el boton para encender el led \n\r  Atencion al Led
o Sonido..! \n");
	//poner en marcha los ISR
	sei() ;

}
//*Interrupt Service Routine
//llamado controlador de interrupciones
ISR (TIMER0_COMPA_vect) {
	if (time1>0)  --time1;
	if (time2>0)  --time2;  // decrementan los time si son mayores a cero
	if (time3>0)  --time3;
}


void tarea1(void)
begin
time1=1;                                        //reset the task timer
if (state==2){                              //si  el estado es igual a 2 entoces
	PORTC |= (1<<3);                    //SE enciende el led en el puerto 3 Esto
activa el Bit y deja el resto a 0
	PORTC |= (1<<0);                    //SE enciende el buzeer en el puerto 0
	time++;                                 //el tiempo se incrementa
	if(Tiempo_espe<3000){      //se entra al if anidado donde si el tiempo de
espera es menor a 3000
		fprintf(stdout,"Te adelantaste!! \n Pulsa de nuevo.. \n\r");  //se
imprime un mensaje de error
		state=4;                        // se pasa al estado 4 dode todo
se pone a cero
	}
}
else{                                       //en caso de que el estado no sea igual
a 2
	PORTC=0x00;                         //todos lo pines va a estar en 0
}
if (time>10000){                            //si el tiempo  es mayor al tiempo
aleatorio que se genero en el la tarea tskdelay
	//genera un mesaje de que el tiempo  supero el limete para pulsar el boton
	fprintf(stdout,"Tiempo limite Excedido..! \n Pulsa de nuevo.. \n\r");
	state=4;                                // y pasa al estdo 4  donde todo
se pone a 0
}
if (state==3)                               // si el estado es igual a 3
{
	// escribe algunos datos en la bandera "written"
	int tiempoAnteriror = eeprom_read_word((uint16_t*)eeprom_data);
```

```
	fprintf(stdout,"Su tiempo Anterior es en ms: ");	// imprime un mesaje
del tiempo anterior  el ususario en pulsar el boton cuando el led esta encendido
	fprintf(stdout,"%d \n\r", tiempoAnteriror) ;


	eeprom_write_word((uint16_t*)eeprom_data,time);
	fprintf(stdout,"Su tiempo actual es: "); // imprime un mesaje del tiempo
que tarda el ususario en pulsar el boton cuando el led esta encendido
	fprintf(stdout,"%d \n\r", eeprom_read_word((uint16_t*)eeprom_data)) ;


	if (time<tiempoAnteriror)
	{
		fprintf(stdout,"FUISTES MAS RAPIDO, EXCELENTE");
	}
	else
	{
		fprintf(stdout,"FUISTES MAS LENTO, ¡LO SIENTO!");
	}

	fprintf(stdout,"\n Pulse para intentarlo de nuevo.. ! \n ");/// imprime un
mensaje para volver a intentar de nuevo medir el tiempo de reaccion

	state=4;				// despues de imprimir los mejase se va
al estado 4 donde todo se pone  0
}
if (state==4){				//si el estado es igual a 4
	state=0;				// el estado se resetea y se pone
a 0
	Tiempo_espe=0;				// el tiempo de espera se resetea
y se pone a 0
	time=0;					// y el time se recetea y
se pone a 0
}
end

void tarea2(void){
	time2=1;	//reset the task timer
	switch (PushState){//estado del boton
		case NoPush:			//1
		if (~PIND & 0x02){
			PushState=MaybePush;
			}else{
			PushState=NoPush ;
		}
		break;
		case MaybePush:			//2
		if (~PIND & 0x02){		//en esta caso PIND vale 00000000 pero
al negarlo vale  111111/00000 como esta afirmacion es falsa
			PushState=Pushed;	// pushState vale 3, es decir se va al
etado 3
			state++;			// estado incrementa
		}
		else{
			PushState=NoPush;
		}
		break;
		case Pushed:			//3
		if (~PIND & 0x02){
			PushState=Pushed;
			}else{
			PushState=MaybeNoPush;
```

```
                }
                break;
                case MaybeNoPush:                  //4
                if (~PIND & 0x02){
                        PushState=Pushed;
                        }else{
                        PushState=NoPush;
                }
                break;
        }
}

void tskdelay(void){
        time3=1;
        if (state==1){
        // si el estado es igual a 1
                nAleat=(9000-5000) + rand ()% 4000;       // se genera valores
aleatorios  entre 4 y 8 seg y se entra al segundo if
                if (Tiempo_espe< nAleat){                              // si el
tiempo de espera es menor al valor generado aleatoriamente
                        Tiempo_espe++;
        // el tiempo de espara se incrementa
                        }else{
        // si el tiempo de espera es igual o mayor al valor generado
aleatoriamente
                        state=2;
        // se va al estado que vale 2 de la maquina de estado que esta en la tarea
2
                }
        }
}

//programa principal
int main(void){
        initialize();
        while(1){
                if (time1==0){              //si time1 llega a 0
                        tarea1();           // se ejecuta la primera tarea
                }
                if (time2==0){              //si time2 llega a 0
                        tarea2();           //se ejecuta la segunda tarea
                }
                if(time3==0){       ////si time3 llega a 0
                        tskdelay();         //se ejecuta la ultima tarea
                }
        }
}
```

Bibliografía

cl, A. (s.f.). Obtenido de http://arduino.cl/arduino-mega-2560/

MicroChip. (s.f.). Obtenido de https://www.microchip.com/mplab/avr-support/atmel-studio-7