¿Cómo se **configura** un puerto (todo el puerto) para que funcione de
entrada? DDRX=0x**00**
salida? DDRX=0x**FF**
entrada y salida? DDRX=0x**FF**
Un puerto
0 entrada
1 salida

Escriba la secuencia (código que pondrían dentro de la función de inicialización) para iniciar el UART1 en modo asíncrono a 1,000,000 bps, 9 bits de datos, 2 bits de paro, paridad par (even). La frecuencia del oscilador es de 8 MHz.

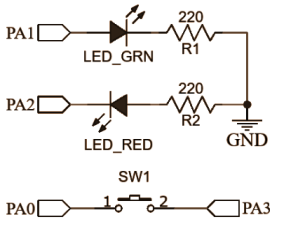
$$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$$

BR = 8000000 / (1000000 * 16) - 1 = 7

```
void UART_init()
{  UCSR1B= (3<<TXEN1) | (1<<UCS212);
   UCSR1A=(1<<02x1);
   UCSR1C=(3<<UCS210)|(1<<USBS1)|(2<<UPM10);
   UBR1=0;
}
```

¿Cuál es el rango de **voltaje** de **entrada** en un puerto digital del Microcontrolador atmega2560 alimentado por una fuente de 5V?
Entrada nivel lógico en bajo: Desde _**-0.5**_V hasta _**1.5**__V.
Entrada nivel lógico en alto: Desde _**3**_V hasta _**5.5**_V.

Tomando en cuenta el siguiente diagrama:



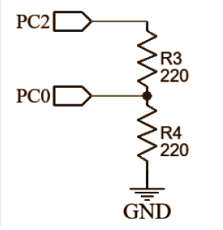

```
void PORT_INT(void)
{
DDRA!= <<PA1;
DDRA&=(1<<PA0);
PORTA != 1 <<PA0;
PORTA & = -(1<<3);
xxxx xxxx
1111 0111
xxxx 0xxx -> F7
}
Green_led funcion
{
PORTA |= 1<<PA1;
else
PORTA &= ~(1<<PA1);
}

Rojo_funcion
{
if(state ==OFF)
PORTA |= 1<<PA2;
else
PORTA &= ~(1<<PA2);
}

uint8_t sw_STATE(void)
{
if(PINA &(1<<PA0))
return 0;
else
return 1;
}

void UPDATE_STATUS(void)
{
if (sw_STATE())
{
leed_green(ON);
led_red(OFF);
}
else
{
led red(ON);
led_green(OFF);
}
}
```

Teniendo el siguiente diagrama:
Donde DDRC=1<<PC2: y PORTC=1<<PC2. ¿Qué valor hay en PINC en el bit 0?



Es **0**, tiene la mitad de vcc, si tuviera vcc conectada sería no se sabe.

Diseñe e implemente una función compacta en lenguaje C llamada GPIOA_isINPUT() que retorne si un GPIOn del Puerto A está configurado como entrada. Considere que el valor num_gpio recibido es válido.

Valor a retornar	Configuración del GPIO como entrada
0	No
1	Si

```
uint8_t GPIOA_isINPUT(uint8_t num_gpio)
{
return !ppra&(1<< num_gpio);
}
```

Describa la **funcionalidad de 3 registros del CPU** (No registros de propósito general) y que pueden ser operables por instrucciones.

- SP - Stack Pointer (Puntero de pila): El SP es un registro que indica la ubicación actual en la pila de memoria. La pila es utilizada para almacenar direcciones de retorno, valores de registro y otros datos temporales durante la ejecución del programa.
- Program Counter (PC - Contador de programa): El PC es un registro que mantiene la dirección de memoria de la próxima instrucción que se ejecutará. Después de ejecutar cada instrucción, el PC se incrementa para apuntar a la siguiente instrucción en secuencia.
- El registro SREG en los microcontroladores AVR almacena las banderas de estado que reflejan el resultado de operaciones aritméticas y lógicas.

En el **ATmega** ¿Porque existen instrucciones con **diferente cantidad de ciclos máquina** requeridos para su ejecución?
Requieren diferentes cantidades de ciclos de reloj porque algunas operaciones son más simples y rápidas, mientras que otras, como el acceso a memoria o cálculos complejos, necesitan más tiempo para completarse debido a la cantidad de trabajo u operaciones internas involucradas en su ejecución. Depende de la complejidad de la instrucción.

¿Cuál es la **dirección indirecta** del **apuntador de pila**? 0x5D

¿Cuáles son los **registros del CPU** del ATmega?

- Registros de propósito general (R0-R31): Son 32 registros de 8 bits que se utilizan para operaciones aritméticas, lógicas y de almacenamiento de datos.
- Program Counter (PC): El Program Counter es un registro especial que contiene la dirección de memoria de la próxima instrucción a ejecutar.
- Registro de estado (SREG): Este registro almacena las banderas de estado después de ejecutar operaciones aritméticas y lógicas. Las banderas reflejan resultados como cero, acarreo, desbordamiento, entre otros.
- Registro de dirección de memoria (MAR): Aunque no es un registro interno específico del CPU, es crucial para la interacción con la memoria, ya que almacena la dirección de memoria a la que se accederá para lectura o escritura.
- Registro de datos de memoria (MDR): Similar al MAR, el MDR no es un registro interno exclusivo del CPU, pero es relevante para almacenar temporalmente datos leídos o datos que se van a escribir en la memoria.

Seleccione las **respuestas** correctas
-Autónomos, Móviles, en Red, de Tiempo Real
→ Categorías de los sistemas embebidos
-Confiabilidad, Costo-Eficiencia, Bajo Consumo de Energía, Eficiente uso de poder de procesamiento, Eficiente uso de Memoria, Tiempo de Ejecución apropiados.
→ Requerimientos de los sistemas embebidos.
-Componentes que son de un denominador común en los sistemas empotrados.
→ Procesador, Memoria Flash, Memoria RAM y Puerto E/S
-Modular, Escalable, Configurable, Uso de poca memoria, Soporte para CPUs, Manejadores de dispositivos.
→ Características deseables de los SO Embebidos
-Funciones específicas, predefinidas, Recursos limitados (memoria, potencia), La aplicación se ejecuta desde ROM o FLASH.
→ Características de sistemas embebidos
-Sistemas computacionales con hardware estrechamente acoplado con la integración de software que están diseñados para realizar una función específica y son por lo general una parte integral de un sistemas más grande.
→ Sistemas embebidos
-La cuestión de elegir entre la implementación del hardware y software se conoce como un tema de esta índole.
→ CO-diseño
-Sistemas en los que restricciones en tiempo están presentes pero no son críticos.
→ Sistemas de tiempo real suaves

En el ATmega ¿Cuántos **ciclos de máquina requieren** la mayoría de **las instrucciones** para su ejecución y por qué?

La mayoría de las instrucciones en los microcontroladores ATmega ejecutan en un solo ciclo de máquina debido a la arquitectura RISC (Conjunto de Instrucciones de Computadora Reducido) que prioriza operaciones simples y eficientes, permitiendo realizar tareas básicas como operaciones aritméticas y lógicas en un único ciclo, aunque algunas instrucciones más complejas, como acceso a memoria o operaciones condicionales, podrían requerir múltiples ciclos para completarse.

El microcontrolador se tarda **casi siempre** un ciclo de reloj para ejecutar una instrucción, esto se debe a su ejecución **encausada**

¿Cuál es la **dirección indirecta** del **contador** del programa?
No existe

Liste 3 **ventajas y** 3 **desventajas** de los **microcontroladores** (uC)
Ventajas
Bajo consumo de energía
Programables por lo que permite adaptar constantemente
Compacto y de bajo costo
Desventajas
No tiene tanta potencia para tareas más complejas por sus recursos limitados
Memoria limitada, por lo que no puede permitir mucho almacenamiento
Dificultad en su desarrollo por su bajo nivel, como el usar lenguaje ensamblador y considerar las limitaciones por los recursos

---

¿Qué es un **microcontrolador**?
Un microcontrolador es un dispositivo electrónico integrado en un solo chip, diseñado para controlar funciones específicas en sistemas embebidos. Tiene CPU, memoria y periféricos. Tienen la capacidad de leer entradas del mundo exterior (sensores, botones, etc.), procesar esa información y tomar decisiones o realizar acciones

---

Indique cuales son las **características de un microcontrolador**:

a) Contiene un microprocesador adentro: Falso
b) Tiene RAM interna: verdadero
c) La mayoría manejan un conjunto de instrucciones complejas: Falso
d) Normalmente se implementan para aplicaciones específicas: verdadero
e) Están contenidos en un solo IC: verdadero

Elementos SE
CPU: Unidad Central de Procesamiento encargada de ejecutar instrucciones y controlar las operaciones.
Memoria: Incluye memoria de programa (ROM o flash) para almacenar el código a ejecutar y memoria de datos (RAM) para almacenar información temporal.
Periféricos: Componentes como puertos de entrada/salida (GPIO), convertidores analógico-digitales (ADC), temporizadores, interfaces de comunicación (UART, SPI, I2C) y otros que permiten la interacción con el mundo exterior.

Qué es un **Sistema embebido**, sistema empotrado
Un sistema embebido, también conocido como sistema empotrado, es un sistema informático diseñado para realizar funciones específicas dentro de un dispositivo más grande o un sistema más complejo. Está dedicado a tareas particulares y generalmente está integrado en un dispositivo que no es una computadora convencional.

Describa la organización de un **sistema empotrado Alexa**
Componentes
-Micrófonos: para capturar comandos de voz para su procesamiento.
-Altavoces: para proporcionar respuestas
-Memoria, Almacena software, datos y configuraciones, incluyendo el firmware del dispositivo.
-Microcontrolador, gestionan las operaciones del dispositivo, posiblemente encargados de tareas específicas como la gestión de energía o el control de periféricos.
Categoría:
Autónomo (Stand Alone) -este no
Sistemas Empotrados de Tiempo Real (Real Time)
Dispositivos en Red (Networking Appliances)
Dispositivos Móviles (Mobile Devices)

El Microcontrolador ATmega1280/2560 **opera sobre** el **rango de voltaje** desde:
1.8V hasta 5.5V

¿Cuál es el **rango de direcciones** de la **RAM** del Microcontrolador atmega1280?
Desde la dirección 0x0200 hasta la dirección 0x21FF

¿A qué categoría(s) de S.E. pertenece un horno de microondas?
Tiempo real, autónomos
¿A qué categoria(s) de S.E. pertenece una teléfono celular?
Todos
¿A qué categoría(s) de S.E. pertenece un dispositivo GPS?
Autónomos, tiempo real, móviles

¿**Dónde** pueden ser encontrados los **sistemas embebidos**?
Están en el 90% de los dispositivos electrónicos y de computo en todo el mundo. En cualquier sistema construido en una tarjeta basada en procesador o up con programas almacenados en forma permanente. por ejemplo: laptops, celulares, camaras, microondas, etc.

Defina lo que es un **Sistema de Tiempo Real**.
Sistema de Tiempo real es que está sucediendo en ese momento, no significa que sea lo más rápido posible o que sea exacto, sino que sea significativo de acuerdo a la velocidad de
cambio en el sistema, ya que puede existir un pequeño desfase de tiempo. Por ejemplo en un celular durante una llamada telefónica las 2 personas se comunican en tiempo real. Existen duros (el tiempo es crítico) y blandos (el tiempo no es crítico).

---

a) El propósito de la ALU es: Realizar operaciones aritméticas y lógicas
b) Ocurre un conflicto en el ducto cuando: Más de un dispositivo está escribiendo en el ducto de datos
c) También ocurre un conflicto en el ducto cuando: Un dispositivo de entrada es habilitado por una señal de escritura
d) El ducto de datos es: Bidireccional y de tres estados
e) El microprocesador tipo RISC sabe cuales bytes interpretar como códigos de operados porque: Cada dirección contiene una instrucción
f) El microprocesador tipo CISC sabe cuales bytes interpretar como códigos de operando porque: Cada código de operación implica el número de bytes de información que le sigue
g) Los sistemas basados en microprocesadores se comunican con los periféricos mediante: Los puertos de entrada y salida
h) En un sistema basado en microprocesador que utiliza dispositivos de memoria de 64K bytes, ¿Qué líneas de direcciones se conectará al IC de memoria?: A0-A15
i) Determina la dirección(es) a la(s) que responde el siguiente decodificador: _3000_H

¿Qué tiene en **común** un **uC y** un **uP**?
Ambos contienen registros, banderas, una ALU y son los encargados de funciona como el “cerebro” del sistema.

¿En qué **difiere** un **microcontrolador** de un **microprocesador**?
El uP no trabaja solo, requiere de dispositivos externos como dispositivos de memoria y E/S. En cambio del uC esta formado por run uP y el conjunto de subsistemas que este mismo requiere (memoria y E/S) por lo que es capaz de realizar tareas básicas solo

---

Escriba una secuencia (ASM) para incrementar el registro Z. La secuencia deberá ser lo más optimizada como sea posible.

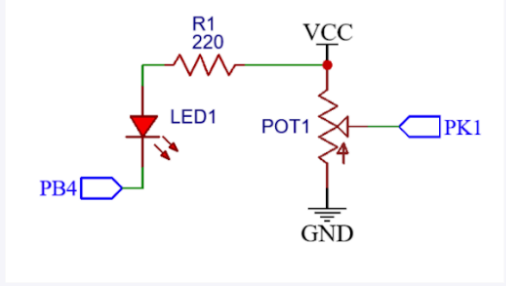
```
ldi z, 0
INC Z
LDI R16, 1   ; Carga el valor 1 en el registro temporal R16
ADD Z, R16   ; Suma el valor de R16 al registro Z
```

Escriba una instrucción o secuencia de instrucciones que cargue el valor del R31 Y R30 (reg Z) en PC.

```
MOVW R30, R31  ; Mueve el contenido de R31 y R30 a los registros de trabajo R31:R30
IJMP           ; Salto indirecto al contenido de la dirección R31:R30
```

Primer microcontrolador
El Intel 4004 es un microprocesador de 4 bits con 2300 transistores, lanzado por Intel Corporation en 1971.

Implementar un programa que lea el potenciómetro y controle directamente el brillo del LED en 5 niveles de intensidad: 0%, 25%, 50%, 75% y 100%. En base al siguiente diagrama:



Donde la correspondencia es 0% de brillo en el LED a 0V del ADC, y 100% de brillo en el LED a 5V en el ADC.
La frecuencia del PWM deberá ser ~488Hz

Criterios de evaluacion:

- Configuracion correcta del Timer (20 pts)
- Configuracion correcta del ADC (20 pts)
- Funcionabilidad del programa (60 pts)

https://wokwi.com/projects/382943603522784257

He visto comentarios que dicen que no se puede o que no funciona wokwi, aquí les comparto una posible solución para que lo revisen y comparen con lo que implementaron https://wokwi.com/projects/383438789006287873 Les recuerdo que el esquemático no es una sugerencia.

```
#include <avr/io.h>
#define HYSTERISIS  16

void delay(void)
{
    volatile uint16_t counter = 0xFFFF;
    while(--counter)
    ;
}
void Adc_Init()
{
  ADMUX = (1 << REFS0) | (1 << MUX0);
  ADCSRB = 1 << MUX5; // ADC 9
                ADCSRA = (1 << ADEN) | (7 << ADPS0);
}

uint16_t Adc_Read(void)
{
                uint16_t result = 0;
                ADCSRA |= (1 << ADSC);

                while ((ADCSRA >> ADSC) & 1)
                                ;
                result = ADCL;
                result |= ADCH << 8;
                return result;
}

void Pwm_Init(void)
{
  DDRB |= (1<<PB4); //OC2A as output
  PORTB |= (1<<PB4); //So that LED is off at the start
  TCCR2A = (3 << COM2A0) | (3 << WGM20); // Inverted Fast PWM
  TCCR2B = (5 << CS00); // PS = 128, FPWM = ~488
  OCR2A = 0; // 0%
}

void Pwm_update(uint16_t adc_value)
{
  uint8_t msb = adc_value >> 2; // use only the 8 msb
  if(msb < HYSTERISIS)
  {
    // Stop PWM to generate "0%"
    TCCR2B &= ~(7 << CS00);   // Stop Timer2
    PORTB |= (1<<PB4); // Turn off the LED
  }
  else if (msb < (0x7F - HYSTERISIS) && msb > (0x3F - HYSTERISIS))
  {
    TCCR2B = (5 << CS00);   // PS = 128, FPWM = ~488
    OCR2A = 0x3F; // 25%
  }
    else if (msb < (0xBF - HYSTERISIS) && msb > (0x7F - HYSTERISIS))
  {
    TCCR2B = (5 << CS00);   // PS = 128, FPWM = ~488
    OCR2A = 0x7F; // 50%
  }
    else if (msb < (0xFF - HYSTERISIS) && msb > (0xBF - HYSTERISIS))
  {
    TCCR2B = (5 << CS00);   // PS = 128, FPWM = ~488
    OCR2A = 0xBF; // 75%
  }
  else if (msb > (0xFF - HYSTERISIS))
  {
    TCCR2B = (5 << CS00);   // PS = 128, FPWM = ~488
    OCR2A = 0xFF; // 100%
  }
  OCR2A = msb;
  PORTB = (PORTB & 0xF0) | (OCR2A >> 4);
}

int main(void)
{
  //Inicializar perifericos
  Adc_Init();
  Pwm_Init();
  //debugging LEDs
  DDRB |= 0xF;
  uint16_t value;
  while(1)
  {
    // Leer potenciometro y actualizar brillo del LED
    value = Adc_Read();
    //PORTB = (PORTB & 0xF0) | (value >> 6);
    Pwm_update(value);
    delay();
  }
}
```

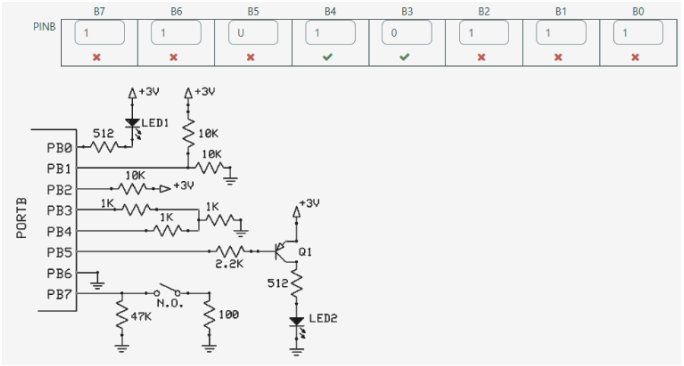
Configure el puerto B según la tabla indicando los correspondientes valores de los bits para configurar DDRB y PORTB. *AVR_*

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink) bajo
1	1	X	Output	No	Output High (Source) alto

Aquí lo que se busca es DDxn Y PORTxn
Nos basamos en el I/O y el Pull-up
En DDRB, si es entrada lleva 0 y si es salida lleva 1
En valor 0 es bajo y 1 alto

No. bit	Dirección	valor		B7	B6	B5	B4	B3	B2	B1	B0
Bit0	entrada	en bajo	DDRB	0 ✔	0 ✔	1 ✔	1 ✔	1 ✔	1 ✔	0 ✔	0 ✔
Bit1	entrada	pull-up off									
Bit2	salida	en bajo	PORTB	1 ✔	1 ✔	0 ✔	1 ✔	0 ✔	0 ✔	1 ✖	0 ✔
Bit3	salida	en bajo									
Bit4	salida	en alto									
Bit5	salida	en bajo									
Bit6	entrada	pull-up on									
Bit7	entrada	pull-up on									

Considerando el circuito de la figura 1 y en inciso anterior. Indique el valor de las terminales después una lectura llenando cada posición del bit correspondiente al registro PINB. Los posibles valores son 0 (cero lógico), 1 (1 lógico) y U (desconocido).
00 010000 LED1 OFF LED 2 ON para que sea ON debe estar como salida 00010000



Indique cuales son las características de un microcontrolador:

a) Contiene un microprocesador adentro: Falso
b) Tiene RAM interna: verdadero
c) La mayoría manejan un conjunto de instrucciones complejas: Falso
d) Normalmente se implementan para aplicaciones específicas: verdadero
e) Están contenidos en un solo IC: verdadero

Considerando que el uC ATmega1280 opera a una frecuencia de 16MHz escriba los valores para configurar el UART3 como 14.3K,8,O,2. baud rate, bits, paridad (No, E-paridad, O-impar), bits por parada
¿Cuál es el error que presentaría esta configuración?
UBRR3=0x45 (16000000/(16*14300))-1 =69.93-1 69 a hexa es 45
UCSR3A=0x00 no se activa doble velocidad U2X
UCSR3B=0x1C
UCSR3C=0x3E
¿Cuál es el porcentaje de error en la velocidad? (16000000/(16*70))=14285.714 ((14285.714/14300)-1)*100 = 0.1%

Considerando que el uC ATmega1280 opera a una frecuencia de 4MHz escriba los valores para configurar el UART0 como 43K,8,N,2 y sin hacer uso de la funcionalidad de doblar velocidad con un bit.
baud rate frecuencia, bits, paridad (No, E-paridad, O-impar), bits por parada

¿Cuál es el error que presentaría esta configuración?
UBRR3=0x05 (4000000/(16*43000))-1 =5.81-1 5 a hexa es 05
UCSR3A=0x00 No doblar velocidad
UCSR3B=0x18
UCSR3C=0x00
¿Cuál es el porcentaje de error en la velocidad? 3.1%

Haciendo uso de los recursos de temporización del microcontrolador escriba una función para generar una onda cuadrada con frecuencia lo más cercano a 8KHz en la terminal PG5

1. El microcontrolador es un Atmega1280/2560
2. El microcontrolador que opera a 16MHz
3. La frecuencia de la onda cuadrada simétrica es 8KHz aprox
4. La onda es generada en la terminal PG5

Respuesta:

```
void Set_8Kz_Out( void )
{
    TCCR0A=      ;   /* configurar modo de operación y acciones sobre salidas */
    TCCR0B=      ;   /* configurar prescalador */
                 ;/* inicializar valor, OCRx = */
    DDRG |=      ;   /* configurar terminal como salida */
}
```

```
/* Ftimer = Fcpu/PS = 16MHz/64 = 250000
Tt = 1 / 8kHz
Ticks = Tt * (16MHz/64) = 31.25 - 1 = 30
Salida PG5*/

void Set_8Kz_Out(void)
{
  TCCR0A = (1 << COM0A0) | (2 << WGM0);
  TCCR0B = (3 << CS00);
  OCR0A = 30;
  DDRG |= 1 << PG5;

/*
  TCCR0A = 0100 0010 -> 0x42
  TCCR0B = 0000 0011 -> 0x3
  OCR0A = 30 -> 0x1E
  DDRG |= 0010 0000 -> 20

*/
}
```

Realizar un programa que despliegue el tiempo de respuesta que una persona tarda en responder con una acción a un evento según la funcionalidad descrita a continuación en base al siguiente esquemático:

1. El eventos corresponde a encender el LED verde (LED_GRN) en un tiempo aleatorio mayor a 1 segundo y menor a 10 segundos.
2. El usuario tiene que presionar el botón (SW1) una vez encendido el LED verde.
3. El tiempo de respuesta se mide iniciando al momento que se enciende el LED y hasta que el usuario presiona el botón.
4. Mediciones mayores a un segundo se consideran una respuesta fallida y deberá encender el LED rojo (LED_RED) durante 2 segundos e imprimir en la terminal el tiempo de respuesta.
5. El formato para desplegar los valores es **XXX.X** ms por que la resolución es de una décima de milisegundo.
2. El usuario tiene que presionar el botón (SW1) una vez encendido el LED verde.
3. El tiempo de respuesta se mide iniciando al momento que se enciende el LED y hasta que el usuario presiona el botón.
4. Mediciones mayores a un segundo se consideran una respuesta fallida y deberá encender el LED rojo (LED_RED) durante 2 segundos e imprimir en la terminal el tiempo de respuesta.
5. El formato para desplegar los valores es **XXX.X** ms por que la resolución es de una décima de milisegundo.
6. Repetir una vez que se apague el LED rojo.

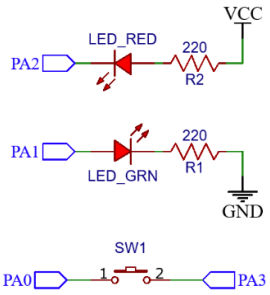
El programa opera en general en lo siguiente manera:
1. Desplegar "Presiona una tecla para iniciar."
2. Se presenta un mensaje de advertencia del inicio.
3. Se imprimen los tiempos de respuesta de cada intento indefinidamente.

b) Subir un archivo correspondiente a la carpeta del proyecto completo (solo código fuente)
Nota: el archivo debe ser tipo ZIP de lo contrario se descartará.

----- código para generar números pseudo aleatorios --
Uso: defina una semilla inicial (por ejemplo **srand**(100)) y luego ya puede usar **rand**()

```
static long holdrand = 1L;
void srand( unsigned int seed ) {   holdrand = ( long ) seed; }
int rand( void )  { return ( ( ( holdrand = holdrand * 214013L + 2531011L ) >> 16 ) & 0x7fff ); }
```

Se pueden apoyar en el siguente link de wokwi: https://wokwi.com/projects/383598589437185025



https://wokwi.com/projects/383598589437185025
https://www.ocr2edit.com/es/convertir-a-txt

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h> // Para srand y rand

#define LED_RED PA24
#define LED_GREEN PA23
#define BUTTON_SW1 PA22
#define RESPONSE_THRESHOLD 1000 // Umbral de respuesta en milisegundos

void delay_ms(uint16_t milliseconds) {
    while (milliseconds > 0) {
        _delay_ms(1);
        milliseconds--;
    }
}

void init() {
    // Configurar pines de salida y entrada
    DDRA &= ~(1 << BUTTON_SW1);
    DDRA |= (1 << LED_RED) | (1 << LED_GREEN);
}

void print_response_time(uint16_t time) {
    printf("%u.%u ms\n", time / 10, time % 10);
}

uint16_t get_random_time() {
    srand(100); // Semilla inicial para la generación de números pseudoaleatorios
    return (rand() % 9000) + 1000; // Tiempo aleatorio entre 1 y 10 segundos
}

int main(void) {
    init();
    uint16_t response_time;

    while (1) {
        printf("Presiona una tecla para iniciar.\n");
        // Esperar a que se presione una tecla para comenzar
        while (!(PINA & (1 << BUTTON_SW1)));

        // Evento: Encender LED verde (LED_GREEN) en un tiempo aleatorio entre 1 y 10 segundos
        response_time = get_random_time();
        PORTA |= (1 << LED_GREEN);
        delay_ms(response_time);
        PORTA &= ~(1 << LED_GREEN);

        // Medir el tiempo de respuesta
        uint16_t start_time = 0;
        uint16_t end_time = 0;
        while (!(PINA & (1 << BUTTON_SW1))) {
            if (start_time == 0) {
                start_time++;
                start_time = TCNT0; // Iniciar el contador al encender el LED verde
            }
        }
        end_time = TCNT0; // Registrar el tiempo al presionar el botón

        uint16_t time_diff = end_time - start_time;

        // Si el tiempo de respuesta es mayor que el umbral, se considera una respuesta fallida
        if (time_diff > RESPONSE_THRESHOLD) {
            printf("Respuesta fallida: ");
            print_response_time(time_diff);
            PORTA |= (1 << LED_RED);
            _delay_ms(2000); // Encender el LED rojo (LED_RED) durante 2 segundos
            PORTA &= ~(1 << LED_RED);
        } else {
            printf("Tiempo de respuesta: ");
            print_response_time(time_diff);
        }
    }
```

```
    return 0;
}
```

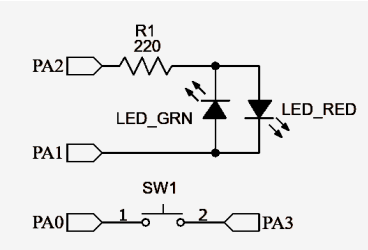
json

```
{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-arduino-mega", "id": "mega", "top": 0, "left": 0,
"attrs": {} },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": -61.2,
      "left": 340.2,
      "attrs": { "color": "red", "flip": "1" }
    },
    {
      "type": "wokwi-led",
      "id": "led2",
      "top": -61.2,
      "left": 311.4,
      "attrs": { "color": "green", "flip": "1" }
    },
    {
      "type": "wokwi-pushbutton",
      "id": "btn1",
      "top": 25.4,
      "left": 460.8,
      "attrs": { "color": "green" }
    }
  ],
  "connections": [
    [ "led2:A", "mega:23", "#8f4814", [ "v28.8", "h44.6" ] ],
    [ "btn1:1.l", "mega:22", "yellow", [ "h-48", "v-19.2", "h-51.8" ] ],
    [ "btn1:2.l", "mega:25", "cyan", [ "h-76.8", "v-28.6" ] ],
    [ "led1:A", "mega:24", "red", [ "v0" ] ],
    [ "led1:C", "mega:23", "orange", [ "v0" ] ],
    [ "led2:C", "mega:24", "black", [ "v0" ] ]
  ],
  "dependencies": {}
}
```

Realizar un programa que despliegue el tiempo de respuesta que una persona tarda en responder con una acción a un evento según la funcionalidad descrita a continuación en base al siguiente esquemático:
El programa opera en general en lo siguiente manera:
Desplegar "Presiona una tecla para iniciar." (Inicializar UART0 a 115,200,8,N,1)
Se presenta un mensaje de advertencia del inicio.
El programa deberá esperar un tiempo aleatorio, mayor a 1 segundo y menor a 10 segundos, después del cual el LED verde (LED_GRN) se encenderá.
El usuario tiene que presionar el botón (SW1) una vez encendido el LED verde.
El tiempo de respuesta se mide iniciando al momento que se enciende el LED y hasta que el usuario presiona el botón.
Mediciones mayores a un segundo se consideran una respuesta fallida y deberá encender el LED rojo (LED_RED) durante 2 segundos e imprimir en la terminal el tiempo de respuesta.
El formato para desplegar los valores es XXX.X ms por que la resolución es de una décima de milisegundo. (Inicializar Timer2 para tener resolución de cada 100 ns).
Regresar al paso 2 una vez que se apague el LED rojo.

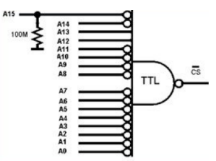
----- código para generar números pseudo aleatorios --
Uso: defina una semilla inicial (por ejemplo srand(100)) y luego ya puede usar randegg

```
static long holdrand = 1L;
void srand( unsigned int seed ) {   holdrand = ( long ) seed; }
int rand( void )  { return ( ( ( holdrand = holdrand * 214013L + 2531011L ) >> 16 ) & 0x7fff ); }
```



Revisar su funcionamiento con el siguente link de wokwi: https://wokwi.com/projects/383598589437185025

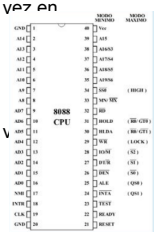
a) El propósito de la ALU es: Realizar operaciones aritméticas y lógicas
b) Ocurre un conflicto en el ducto cuando: Más de un dispositivo está escribiendo en el ducto de datos
c) También ocurre un conflicto en el ducto cuando: Un dispositivo de entrada es habilitado por una señal de escritura
d) El ducto de datos es: Bidireccional y de tres estados
e) El microprocesador tipo RISC sabe cuales bytes interpretar como códigos de operados porque: Cada dirección contiene una instrucción
f) El microprocesador tipo CISC sabe cuales bytes interpretar como códigos de operando porque: Cada código de operación implica el número de bytes de información que le sigue
g) Los sistemas basados en microprocesadores se comunican con los periféricos mediante: Los puertos de entrada y salida
h) En un sistema basado en microprocesador que utiliza dispositivos de memoria de 64K bytes, ¿Qué líneas de direcciones se conectará al IC de memoria?: A0-A15



i) Determina la dirección(es) a la(s) que responde el siguiente decodificador: _3000_H

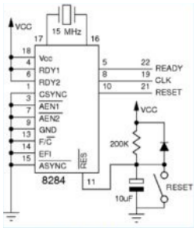
j) Los dispositivos de tres estados se utilizan porque: Estos permiten que varios dispositivos puedan conectarse fácilmente
k) Los dispositivos E/S en un sistema basado en microprocesador: Todas las respuestas
l) Los sistemas basados en uP y uC son ms flexibles que los diseños lógicos debido a: Su operación es controlada por software
m) Las memorias RAM no se utilizan para almacenar datos por tiempo prolongado porque: su contenido se pierde cuando se dejan de energiza
n) Las memorias FLASH y ROM se utilizan principalmente para: almacenar permanentemente programas y datos
o) Los periféricos son: los dispositivos de entrada y salida
p) Los sistemas basados en uP y uC manejan datos en grupos de 4 bits llamado: nibble
q) En un sistema basado en microprocesador con un ducto de direcciones de 20bits, ¿Cuál es el número máximo de dispositivos de memoria de 1K byte que puede contener?: 1024

Seleccione el lenguaje acorde a la descripción:
Lenguaje que será convertido: Lenguaje fuente
Lenguaje que solamente es una representación simbólica de un lenguaje numérico: Lenguaje ensamblador
Lenguaje numérico baso en 1's y 0's entendible y ejecutable por uP y uC: Lenguaje máquina
Lenguaje al que se convertirá: Lenguaje objeto
Lenguaje que utiliza una notación especial orientada al problema a resolver y no se requiere conocimiento del código máquina: Lenguaje de alto nivel
Selecciona la opción acorde a la descripción:
Grupo de instrucciones que desarrollan una tarea, puede usarse dentro del programa varias veces pero se almacena solamente una vez en memoria, y generalmente regresa valor(es).
- función
Permite asignar un nombre a una porción de texto y reutilizarla en el programa las veces que sea necesario.
-macro
Grupo de instrucciones que desarrollan una tarea, puede usarse dentro del programa varias veces pero se almacena solamente una v memoria
-procedimiento
Selecciona la respuesta correcta:



-El procesador usa ésta terminal para indicar que el ducto de direcciones contiene una dirección de memoria válida o una dirección de un puerto válido. ALE
-Una terminal utilizada para controlar la dirección del flujo del dato mediante una conexión externa a los reforzadores del bus de datos. DT/R'
-Es una terminal de entrada que al ser activada causa una interrupción tipo 2 y que es entendida al final de la ejecución de la actual instrucción. NM
-Siempre que esta terminal es cero lógico, el procesador prepara su ducto de datos para recibir un dato de la memoria o de los dispositivos de E/S conectados al sistema. RD'
-Es una terminal en la interfaz con dispositivos que requieren que el ciclo ducto sea mayor en tiempo al ciclo ducto típico. READY
-El procesador la usa para indicar que el ducto de datos contiene un dato válido para ser almacenado en la memoria o enviado a E/S. WR'

Considerando las condiciones en las que se encuentra el generador de reloj 8284 del siguiente diagrama determine lo solicitado.



a) ¿Que frecuencia se presenta en las terminales CLK y PCLK?
CLK = 5 MHz PCL= 2.5 MHz
b) ¿Que frecuencia se presenta en las terminales CLK Y PCLK si la terminal F/_C se conecta a Vcc?

CLK = 0 MHz PCL = 0 MHz

Si uP 8088 opera con una frecuencia de 5MHz ¿Cuánto tiempo requiere para completar un ciclo ducto estándar? 800 ns

Decodifique los siguientes opcode para un 8088

Dirección	Contenido
072A:100	31
072A:101	D2

Instrucción: XOR DX, DX
Registro modificadi: DX, valor: 0h

Decodifique los siguientes opcode para un 8088

Dirección lógica	Contenido
1740:1000	E9
1740:1001	FD
1740:1001	FF

¿Qué instrucción de salto es? NEAR JMP
Según el contenido de memoria
¿A qué dirección lógica salta? 1740: 1000
Ensamble la siguiente secuencia según los mnemónicos y códigos máquina del uP 8088 correspondiente

Instrucción	Código máquina (hex)
MOV AL,34h	h
MOV DX,40h	h
OUT 42h,AL	h
NEXT: IN AL,DX	h
JMP NEXT	h

B034 BA4000 E642 E4 EBFE

Decodifique los siguientes opcode para un 8088 y describa utilizando los mnemónicos

Dirección	Contenido
072A:200	B9
072A:201	10
072A:202	00
072A:203	31
072A:204	FF
072A:205	8A
072A:206	05
072A:207	E6
072A:208	42
072A:209	47
072A:20A	E2
072A:20B	F9

Y que valor tiene BX?

mov cx,10h
xor di,di
@label: mov al,[di]
out 42h,al
inc di
loop @label
El valor de BX es desconocido

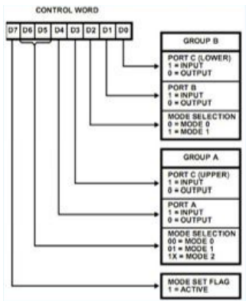
Nota: Las respuestas a las preguntas abiertas es de forma numérica (ej.: 1,2,3...) respetando las unidades especificadas o base numérica.

1. ¿En que modo está trabajando el uP? MIN
2. ¿A que frecuencia esta trabajando el uP? 4MHZ
3. ¿Cuántos ciclos de espera (Tw) se insertan al trabajar con la EEPROM? 0 ciclope
4. ¿Cuánto tiempo se requiere para completar un ciclo ducto? 1000 ns
5. ¿En qué memoria existen direcciones espejo? en EEPROM
6. ¿Se está utilizando completamente la RAM en el sistema?- si
7. ¿Cuál es el rango de direcciones de la RAM en el sistema? Desde 0h h
   hasta la dirección- 7FFFh
8. ¿Qué tamaño de memoria tiene la EEPROM? 32 KByte
9. ¿Se está utilizando completamente la EPROM en el sistema? no
10. ¿Cuál es el rango de direcciones de EPROM en el sistema? Desde 0h h
    hasta la dirección 3FFFh

Determine las ecuaciones para los decodificador de direcciones para una memoria para la sección de RAM y EPROM de un sistema basado en el procesador 80C88 según los siguientes requerimientos

1. La RAM es **512KB** y debe iniciar en la dirección **00000H** con direcciones espejo al inicio de la segunda mitad del espacio total de memoria del sistema.
2. La EPROM es de **256KB** se encuentra finaliza en la dirección **FFFFFH** y no tiene direcciones espejo.

Este diseño no es posible debido a que se sobreponen las memorias
Determine las ecuaciones de un decodificador para la sección de E/S con un dispositivo PPI (8255) para que éste tenga una dirección base 0378h.
R: A15' * A14' * A13' * A12' * A11' * A10' * A9 * A8 * A7' * A6 * A5 * A4 * A3 * A2' * (IO/M')
Dado un 8255 con dirección base 80H, escriba un procedimiento lo más compacto posible para programar en modo 0 al puerto A y puerto C (bajo) de entrada, puerto B y puerto C (alto) de salida.




```
prog proc
mov ax, 91h
 out 83h, ax
ret
endp
```

¿Qué es lo sucede en el procesador (8088) cuando se ejecuta la INT 40h?

Se guardan los registros cs, ip y las banderas. Brinca al vector de interrupciones y busca la sección correspondiente a la interrupción 40h donde recupera el contenido y lo coloca en cs e ip para brincar al espacio correspondiente de memoria donde se encuentra el código de la interrupción.

Diseñe e implemente un procedimiento en lenguaje ensamblador (x86 -16bits) para leer escribir un byte a un puerto. El procedimiento será invocado desde programas en lenguaje C considerando la siguiente función prototipo:

```
_outportb   PROC          ; comentarios
            push bp       ; protocolo de entrada a función
            mov bp,sp
            mov al,[bp+6] ; copiar parámetro dato
            mov dx,[bp+4] ; copiar parámetro puerto
            out dx,al     ; sacar dato por puerto
            pop bp        ; protocolo de salida de función
            ret
            ENDP
```

Diseñe e implemente un procedimiento en lenguaje ensamblador (x86 -16bits) para leer un byte de un lugar de memoria dado el segmento y desplazamiento correspondiente (dirección lógica). El procedimiento será invocado desde programas en lenguaje C

```
_peekb  PROC        ; comentarios
        push bp     ; protocolo de entrada a función
        mov bp,sp
        mov di,[Bp+6 ]   ; copiar parámetro desp al registro correspondiente
        push ds     ; salvar segmento original
        mov ds,[Bp+4]    ; copiar parámetro segm al registro correspondiente
        mov al, [di]     ; leer valor de memoria al registro de retorno
        pop ds      ; recuperar segmento original
        pop bp      ; protocolo de salida de función
        ret
        ENDP
```

Considere:

```
typedef unsigned char uint8_t;
/*
* Prototipo:
* void WriteBit ( uint8_t *dato, uint8_t num_bit, uint8_t valor_bit );
*
* Ejemplo de uso:
* Si existe uint8_t valorA=0x07
* Escribir 1 al bit 3 de valorA
*/
WriteBit( &valorA, 3, 1 ); /* Resultado: valorA = 0x0F */
```

(para después ser llamada desde lenguaje C) para escribir un bit de un determinado dato el cual será pasado como argumento por referencia, así como el número del bit a operar y el valor del bit a escribir:

```
_WriteBit proc
push bp
mov bp,sp
mov bx,[bp+4] ;dirección del dato
mov cl,[bp+6] ;numbit // bit a modificar
mov ch,[bp+7] ;valor bit
mov dx,1 ;mascara
shl dx,cl //
cmp ch,0
je @@cero
uno: or [bx],dl
jmp @@fin
cero: not dx
and [bx],dl
fin: pop bp
ret
endp
```

Diseñe una función en lenguaje C para desactivar un determinado bit de un determinado puerto el cual será pasado como parámetro así como el número del bit a operar.

Notas:

- Debe hace uso de la técnica leer-modificar-escribir (read-modify-write)
- Deberá hacer uso de la función **WriteBit()** -- ver parte a)
- Puede hace uso de las funciones :

```
uint8_t inportb( uint16_t puerto );
        outportb ( uint16_t puerto , uint8_dato );

typedef unsigned char uint8_t;
typedef unsigned int uint16_t;
/* Prototipo:
* void ClrBitPort ( uint16_t puerto, uint8_t nbit );
*
*/
ClrBitPort( 0x80, 1 );
```

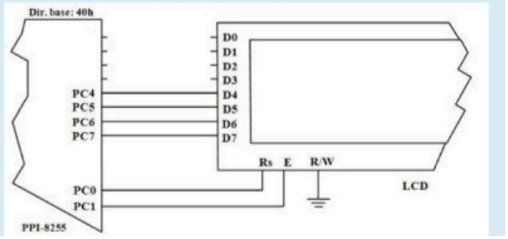
Respuesta:

```
void    ClrBitPort ( uint16_t puerto , uint8_t nbit ) {
        uint8_t x;
        x =inportb( puerto );          /* leer      */
        WriteBit(&x,nbit,0);           /* modificar */
        outportb( puerto , x );        /* escribir  */
}
```

Considerando la figura, diseñe una función en lenguaje C (WriteLCD)

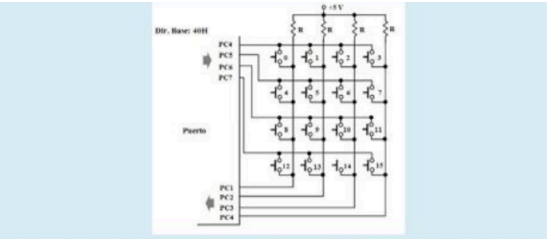
Considerando la figura, diseñe una función en lenguaje C (WriteLCD) para escribir un dato o instru
Dato, Rs=0: Instrucción) de 8 bits al LCD. El dato a introducir al LCD se debe hacer por nibble (prim
significativo) y **se graba en el flanco de bajada de E**.
Nota1: asumir que el 8255 fue previamente programado con PC como salidas.
Nota2: Considere la existencia de las funciones inportb, outportb, SetBitPort, ClrBitPort, NotBitPort.

```
/* Prototipo: void WriteLCD( uint8_t tipo, uint8_t dato);
Si tipo=0 : Instrucción de otra manera dato */
```




```
void WriteLCD(uint8_t tipo, uint8_t dato)
{
uint8_t aux = dato;
if(tipo == 0) ClrBitPort(PC,0);
if(tipo == 1) SetBitPort(PC,0);
aux = aux<<4;
outportb(PC,aux);
SetBitPort(PC,1);
ClrBitPort(PC,1);
outportb(PC,dato);
SetBitPort(PC,1);
ClrBitPort(PC,1);
}
```

Según la figura diseñe una función en lenguaje C que rastrea y determina que tecla ha sido oprimida. Si no se detecta tecla presionada la función retorna -1, de lo contrario retorna el número de la tecla presionada.



Nota1: Considere que la disponibilidad de las funciones **inportb, outputb, SetBitPort, ClrBitPort, NotBitPort** y **GetBitPort**
Nota2: Considere el **PPI-8255** ya configurado e iniciado de la manera correcta para esta aplicación específica.

```
int cnt1 = 0, cnt2 = 0, res=-1;
while(cnt1<4){
if(getbitport(PC,cnt1)==0) break;
cnt1++;
}
if(cnt1 != 4){
while(cnt2<4){
setbitport(PC,cnt2+4);
if(getbitport(PC,cnt1) == 1) break;
clrbitport(PC,cnt2+4);
cnt2++;
}
res = cnt2*4+cnt1;
}
return res
```

¿Qué tiene en común un uC y un uP?

Ambos contienen registros, banderas, una ALU y son los encargados de funciona como el “cerebro” del sistema.
¿En qué difiere un microcontrolador de un microprocesador?
El uP no trabaja solo, requiere de dispositivos externos como dispositivos de memoria y
E/S. En cambio del uC esta formado por run uP y el conjunto de subsistemas que este
mismo requiere (memoia y E/S) por lo que es capaz de realizar tareas basicas solo
El microcontrolador se tarda **casi siempre** un ciclo de reloj para ejecutar una instrucción, esto se debe a su ejecución **encausada**
Considerando que el procesador no cuenta con las instrucciones de rotación (sin acarreo) y solo tiene corrimientos a la izquierda y a la derecha.
Escriba una secuencia para hacer la rotación (no través del acarreo) sobre el registro z.

```
ldi r30,0x080; valores de prueba para el atmel
ldi r31,0x80
mov r29,r31 ;ZH
mov r28,r30 ;ZL
lsl r31
lsl r30; recorre z
andi r29,0x80 ;1000 0000
andi r28,0x80 ;1000 0000 mascara
```

```
cpi r29,0x00
breq siguiente
ori r30,0x01
siguiente:
cpi r28,0x00
breq final
ori r31,0x01
final:
```

Diseñe e implemente una función en ensamblador (para ser llamada desde lenguaje C) que active un determinado bit de un determinado puerto del Atmega1280/2560. El punto y el número de bit a operar son pasados como parámetros

```
typedef unsigned char uint8_t;
typedef unsigned int uint16_t;
/* Prototipo:
void SetBitPort ( uint16_t puerto, uint8_t nbit );
*/
SetBitPort( 0x80, 1 );
```

```
ldi r16,0x01
mov r30,r24
mov r31,r25
cpi r22,0
..breq fin
mov r18,r22
```

```
nxt: lsl r16
dec r18
cpi r18,0
brne nxt
fin: ld r22, Z
or r16,r22
st Z,r16
```

Considere la siguiente secuencia

```
        ;**** inicio ****; num. de ciclos
        ldi r25,N         ; 1
next:   nop               ; 1
        dec r25           ; 1
        brne next         ; 2 y 1
        ;**** fin ****
```

a. ¿Qué valor debe ser N para lograr que la secuencia consima 768 ciclos que en tiempo equivale a 48us si el uC ATmega2560 opera a 16MHz? 192
b. ¿Qué valor debe ser N para lograr que la secuencia consuma 1024 ciclos que en tiempo equivale a 64us si el uC ATmega2560 opera a 26MHz? 0 (256, pero 0)

Implemente una secuencia (repetitiva) en lenguaje ensamblador para el microcontrolador ATmega2650 que consuma 4096 ciclos

```
ldi R25, 32
nxt1: ldi R26,25
nxt2: nop
nop
```

```
dec R26
brne nxt2
dec R25
brne nxt1
```

Implemente una función de retardo llamada Delay750uS_int() la cual se basa en el temporizador 0 (Temer0) para un retardo de 750uS considerando los siguientes requerimientos

1. La función es para un microcontrolador **ATmega1280/2560.**
2. La función deberá utilizar la(s) interrupción(es) del **timer0** correspondiente
3. El uso de **timer0** es exclusivo para la función -- no existe otro código que que hace uso del timer
4. El microcontrolador opera tiene como reloj principal el oscilador externo de **16 MHz.**
5. La forma de uso es:
   Delay750uS_int(); /* va a la función y regresa logrando un retardo de 750us */
6. La función deberá hacer uso de rutina(s) de servicio de interrupción (**RSI**) lo más compacta posible
7. El código de la función realiza la configuración y uso de los recursos requeridos para lograr la funcionalidad

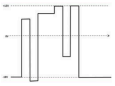
```
static volatile uint8_t flag;
void Delay750us_init(){
flag=0; ←--- inicializa bandera en 0
TCCROA = 2<<WGM0 //MODO ctc
OCR0A= 186;
TCNT=0
TCCROB= 3<<CS00; //PRESCALADOR DE 64 (750US*(16MHZ/64))=187.5;
while(!flag);<--- mientras bandera sea 0.
TCCR0B=0;
}
ISR(TIMER0_COMPA_vect){
flag=1;
```

}

Considerando una comunicación UART 9600,5.N,1 y la captura del Frame de un dato que se muestra diga que dato ha sido transmitido

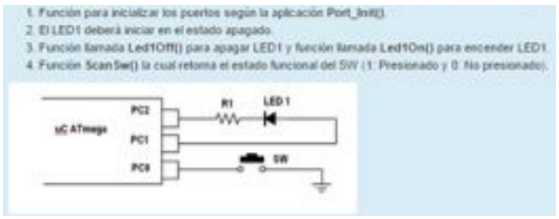
a) Dato= 29

b) Dato=D1



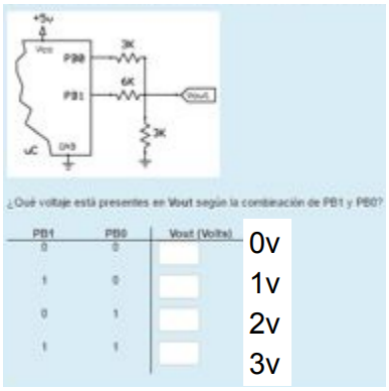
¿Qué registros funcionan para configurar la velocidad de operación del puerto serie 0?
UBRR0 y UCSR0A(bit u2x0)

Diseñe e implemente las siguientes funciones en lenguaje C según la configuración mostrada en la figura:




```
void Port_Init( void ){
DDRC = (1<<PC1) | (1<<PC2) configurar puertos */
PORTC = /* inicializar puertos */
(1<<PC0)
}
void Led1On( void ){
PORTC ^=
4>>PC1
}
void Led1Off( void ){
PORTC = 1
<<PC1); /* apaga led 1 */
}
uint_8_t ScanSW( void ){
return (
==0) 1 ; /* retorna estado de SW*/
(PINC & (1<<PC0)
}
```

Considere el siguiente diagrama donde el sistema es alimentado con 5.0 volts y las terminales PB0 y PB1 están configuradas como salida



¿Qué voltaje está presentes en Vout según la combinación de PB1 y PB0?

PB1	PB0	Vout (Volts)	
0	0		0v
1	0		1v
0	1		2v
1	1		3v