24-4-2021

ADRIAN OROZCO ANZURES 18300328  
CARLOS DANIEL LOZANO VAZQUEZ 18300249

SiSTEMAS EMBEBIDOS I

CETI COLOMOS

**Interrupciones**

Actividad 17 – Práctica 10

Logotipo

Descripción generada automáticamente

**Objetivo**

El objetivo de esta práctica es aprender a utilizar las Interrupciones del Arduino para realizar una aplicación demostrativa.

**Descripción**

Utilizar Arduino MEGA y conectarle una LCD. Programar 3 interrupciones.

Programa principal: Mandar a la pantalla los números de 0 a 99 separados por un tiempo de 500 ms entre cada número.

Interrupción 1: Cuando se active una interrupción (por ejemplo, la externa) mostrar el nombre de uno de los integrantes del equipo o una frase cualquiera “Hola CETI”

Interrupción 2: Cuando se active otra interrupción (por ejemplo, la de un timer cada 30 segundos) mostrar el nombre de otro de los integrantes del equipo o una frase cualquiera “Buen día a todos”

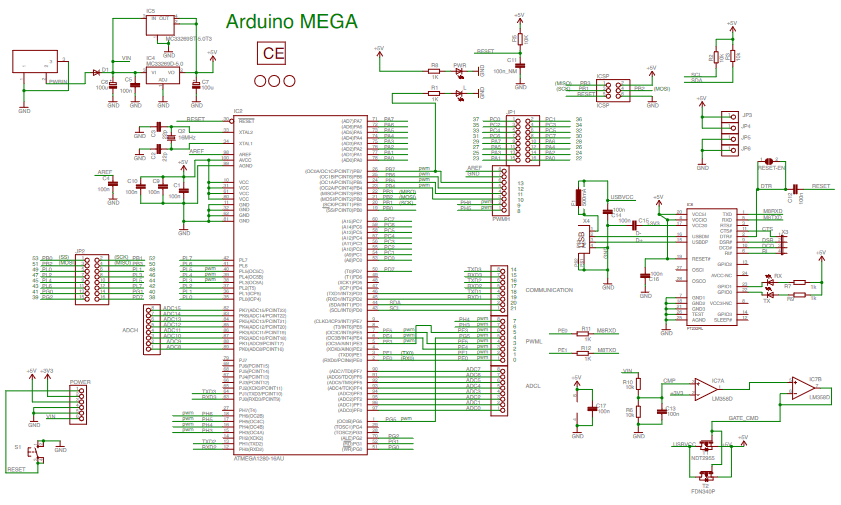
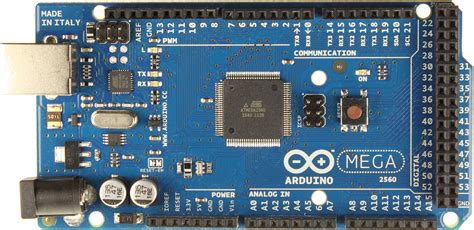
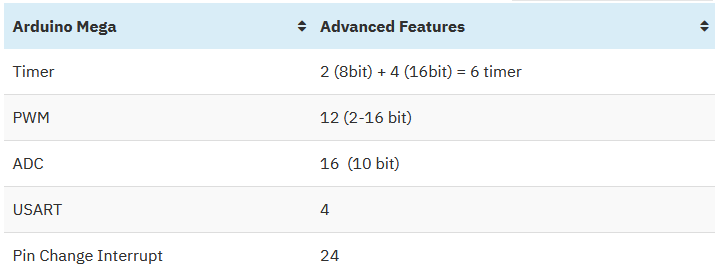
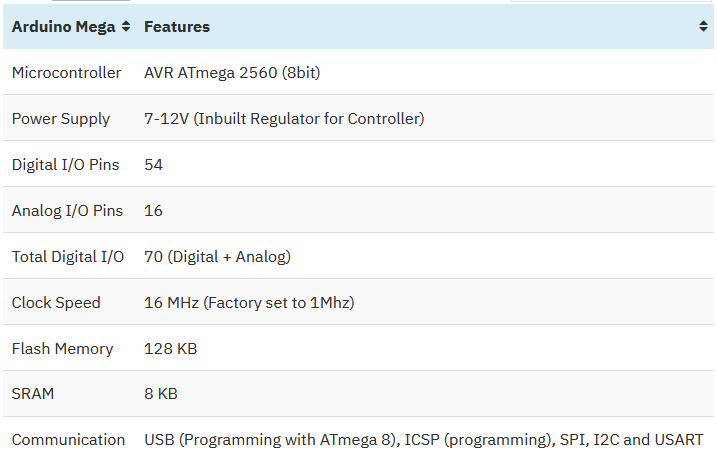
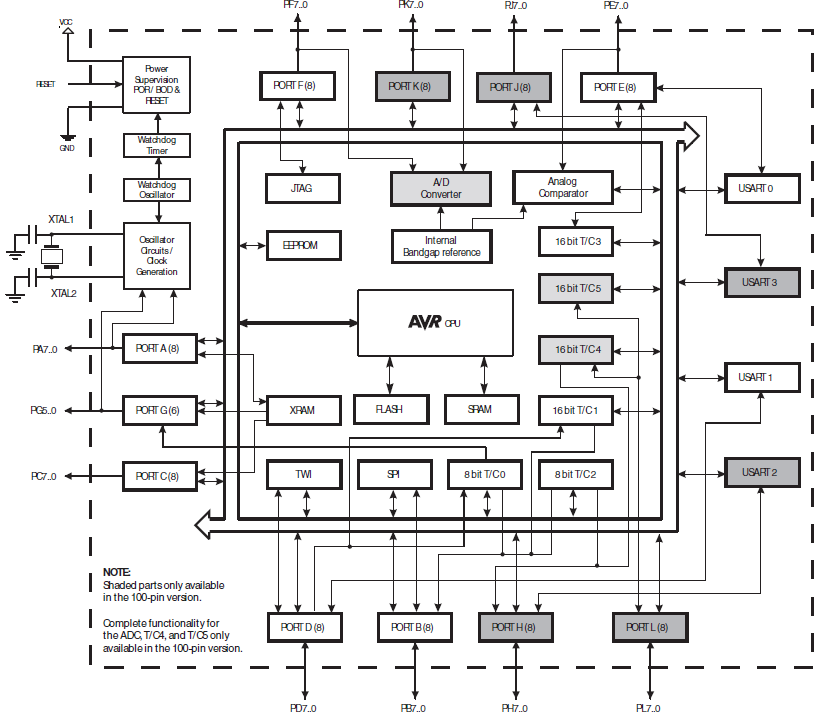
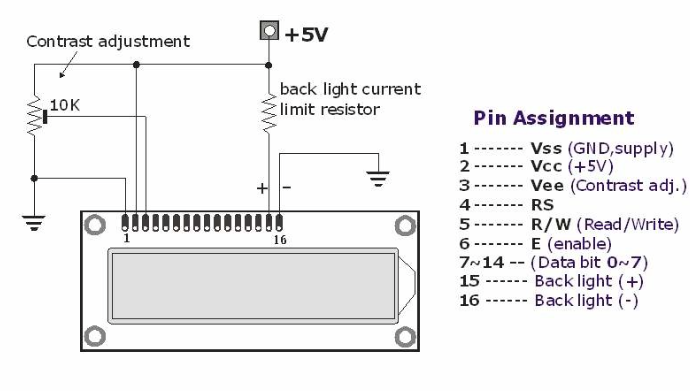
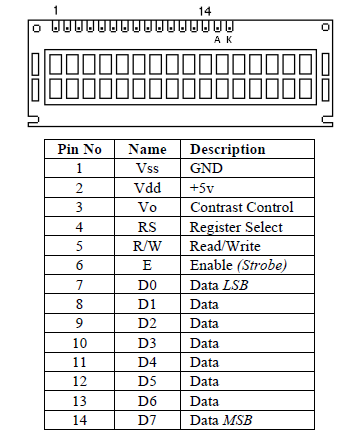
Interrupción 3: Cuando se active otra interrupción, la que tu elijas del Arduino, mostrar el nombre de otro de los integrantes del equipo o una frase cualquiera “Quédate en casa”.

**Resumen**

Usando una tarjeta de Arduino MEGA y un LCD de 16x2, se debe crear un programa que utilice interrupciones de la tarjeta. Para esto se usarán 3 interrupciones, uno que se muestra cada 30 segundos, otro cuando se presiona un push button, y otro mientras se presione un push button.

El programa es creado en un editor de texto, ya sea Visual Studio Code, Sublime Text, Atom, o hasta el mismo Arduino IDE; pero para utilizar el código este debe ser verificado en el IDE de Arduino.

**Diagrama Eléctrico**

********

**Circuito**

**Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja**

**Código**

#include <LiquidCrystal.h>

#include <TimerOne.h>

#define BTN\_1 2

#define BTN\_2 3

LiquidCrystal lcd(27, 26, 25, 24, 23, 22);*// Creación de la LCD*

volatile int cont = 0;

unsigned long timeA = 0, timeC = 0;*// Tiempo para Adrian y Carlos*

auto render = [] (const char\* *str*) {

    lcd.setCursor(0, 0);

    lcd.print(*str*);

    return;

};

void setup () {

    pinMode(BTN\_1, INPUT\_PULLUP);

    pinMode(BTN\_2, INPUT\_PULLUP);

*// Interrupción de falling*

    attachInterrupt(digitalPinToInterrupt(BTN\_2), [] {

        if (millis() - timeA > 150) {

            render(" \*Adrian Orozco\*");

            timeA = millis();

        }

        return;

    }, FALLING);

*// Interrupción de rising*

    attachInterrupt(digitalPinToInterrupt(BTN\_1), [] {

        if (millis() - timeC > 150) {

            render(" \*Carlos Lozano\*");

            timeA = millis();

        }

    }, RISING);

    Timer1.initialize(1000000);*// Inicializado en 1 segundo*

*// Interrupción interna por tiempo*

    Timer1.attachInterrupt([] {

        cont++;

        if (!(cont % 30))

            render("Buen dia a todos");

        return;

    });

    lcd.begin(16, 2);

    render("Hola");

}

void loop () {

    return;

}

**Explicación**

El programa comienza en setup(), donde se inicializa las interrupciones. Estas toman un modo de interrupción, la función a ejecutar en la interrupción y el modo de interrupción.

En este caso, la interrupción de falling se ejecuta mientras el botón este presionado, y esta misma imprime “ \*Adrián Orozco\*”, la interrupción de rising se ejecuta cuando el botón se presiona y se suelta, si el botón no se suelta no se ejecutará la interrupción; está interrupción imprimirá “ \*Carlos Lozano\*”.

El programa por predeterminado tiene una interrupción integrada llamada Timer1, esta es modificada para imprimir “Buen día a todos” cada 30 segundos.

**Observaciones**

Una práctica muy sencilla de realizar, tuve un poco de problemas configurando los botones porque algunos puertos no los podía usar como input.

**Conclusión**

Una práctica muy potente para los timers, saber manejar timers e interrupciones por igual nos permite crear aplicaciones muy potentes, pues podemos realizar funciones sin alterar el estado del programa, simplemente poniéndolo en “pausa”.