



Universidad Veracruzana

UNIVERSIDAD VERACRUZANA

FACULTAD DE INSTRUMENTACIÓN
ELECTRÓNICA



FACULTAD
DE
INSTRUMENTACIÓN
ELECTRÓNICA

25-11-2020

TÓPICOS AVANZADOS DE INSTRUMENTACIÓN ELECTRÓNICA I (SISTEMAS EMBEBIDOS)

Tarea 9.- RTOs

Alumnos:

Rangel Pulido Julio

Martínez González Sergio David

Mendoza Ramírez Jesús Emiliano

Catedrático:

Hernández Reyes Sergio Machuca

ÍNDICE

1. ACTIVIDAD	3
2. DESCRIPCIÓN	3
3. LISTA DE COMPONENTES USADOS.....	9
4. CONCLUSIONES	9
5. BIBLIOGRAFÍA	10

1. ACTIVIDAD

Para la siguiente actividad se deberá ejercitar el uso de FreeRTOS ya sea en una placa Arduino (Mega2560) o de preferencia en una placa Esp323, los puntos que será necesario realizar son los siguientes:

Programar 8 tareas las cuales se verifique la secuencia de ejecución a través de la notación ya especificada (/ Tarea 1: --- 01, Tarea 2: ----- 02, etc.).

Mostrar si la tarea recibió o envió datos a otra tarea (esto implica verificar cuáles son las funciones o directivas que se pueden emplear para que una tarea comunique algo a otra).

Dar distintas prioridades, distintos valores de frecuencia a las tareas, así como distintos tiempos de conclusión

Ejercitar otras características más que crea importantes de FreeRTOS

2. DESCRIPCIÓN

Sistema en tiempo real con FreeRTOS:

En esta actividad se va a desarrollar un programa el cual realizará distintas tareas usando un sistema operativo de tiempo real (FreeRTOS) en este caso se utilizará una placa Esp32 por la facilidad de que no necesita de ninguna librería para ser compatible con el sistema operativo ya mencionado.

Las tareas mencionadas serán un total de 8 estas contarán con distintas conclusiones, también tendrán interrupciones a distintos momentos, algunas tareas se comunicarán con otras y todas las acciones que lleven a cabo serán mostradas al usuario a través del puerto serial

A continuación, se muestra el programa, el cual fue desarrollado en la misma ide de Arduino

Líneas del código.

```
//OCURRENCIA DE TAREAS
//Tareax: "---x"
//Tareay: "-----y"
//Tareaz: "-----z"

//ENVÍO DE DATOS ENTRE TAREAS
//Tarea x envió a y: "---x [-> y]"

//RECEPCIÓN DE DATOS ENTRE TAREAS
//Tarea x recibió de y: "---x [<- y]"

//SUSPENSIÓN DE TAREAS
// "---x [!]"

//AUMENTO DE PRIORIDAD EN TAREAS
// "---x [+]"
```

```

//=====
==
#include <Arduino.h>           //Librearía de arduino para ESP32

#define INCLUDE_vTaskSuspend 1 //Para activar la suspensión de tareas

QueueHandle_t coladetareas;    //Declaración de una cola para comunicación
entre                          //tareas
int estadoled = 0;             //Variable que muestra el estado del LED
int a = 0;                     //Variable para funcionamiento de impresiones

//-----
//-----[ AJUSTES ]-----
//-----
void setup() {

//Creación de una cola de comunicación entre tareas
coladetareas = xQueueCreate(10, // Número de elementos de la cola
                             sizeof(int) // Tamaño de cada dato de la cola
                             );

    if (coladetareas != NULL) { //Comprobación de que se ha creado la cola
        Serial.begin(115200);   //Se inicializa el puerto serie a 115200
baudios
        Serial.println("-- INICIO --"); //Se imprime una señal de inicio
        delay(1000);             //Delay de inicio de programa

// Creación de las tareas:
        xTaskCreate(
            Tarea1,                // Función que ejecuta la tarea
            "Tarea1",              // Cadena con el nombre de la tarea
            1000,                  // Tamaño del "Stack"
            NULL,                  // Parámetro pasado como entrada
            7,                     // Prioridad d la tarea
            NULL);                // Manejador (Handle) de la tarea
        xTaskCreate(Tarea2, "Tarea2", 1000, NULL, 6, NULL);
        xTaskCreate(Tarea3, "Tarea3", 1000, NULL, 5, NULL);
        xTaskCreate(Tarea4, "Tarea4", 1000, NULL, 4, NULL);
        xTaskCreate(Tarea5, "Tarea5", 1000, NULL, 3, NULL);
        xTaskCreate(Tarea6, "Tarea6", 1000, NULL, 2, NULL);
        xTaskCreate(Tarea7, "Tarea7", 1000, NULL, 1, NULL);
        xTaskCreate(Tarea8, "Tarea8", 1000, NULL, 0, NULL);
    }
}

//-----
//-----[ PRINCIPAL ]-----
//-----
void loop() {}

//=====
==

//-----
//-----[ TAREA 1 ]-----

```

```

//-----
//-----
--
void Tarea1( void * parameter ){
    for( int i = 0;i<6;i++ ){           // Número de veces que ocurre esta tarea
        Serial.println("--- 01");        // Evidencia de ocurrencia
        vTaskDelay(2000);                // Espera...
    }
    Serial.println("--> 01");             // Conclusión de la tarea
    vTaskDelete( NULL );                 // Se "Borra"
}
//-----
--
//-----
//-----[ TAREA 2 ]-----
//-----
//-----
--
void Tarea2( void * parameter){
    for( int i = 0;i<10;i++ ){           // Número de veces que ocurre esta tarea
        Serial.println("----- 02");    // Evidencia de ocurrencia
        vTaskDelay(2000);                // Espera...
    }
    Serial.println("--> 02");             // Conclusión de la tarea
    vTaskDelete( NULL );                 // Se "Borra"
}
//-----
--
//-----
//-----[ TAREA 3 ]-----
//-----
//-----
--
void Tarea3( void * parameter){
    for( int i = 0;i<10;i++ ){           // Número de veces que ocurre esta tarea
        if(i == 4){                      //Suceso a la 5ta ejecución
            estadoled = 1;                //Ajusta el LED en ON

            //Envío de un dato a la cola de comunicación para su recepción
            xQueueSend(coladetareas, &estadoled, portMAX_DELAY);
            //Confirmación al usuario de que se ha enviado el dato
            Serial.println ("----- 03 [-> 4]");//Tarea 3 envía a 4

        }
        else{
            Serial.println("----- 03 "); // Evidencia de ocurrencia
        }

        vTaskDelay(2000);                // Espera...
    }
    Serial.println("--> 03");             // Conclusión de la tarea
    vTaskDelete( NULL );                 // Se "Borra"
}
//-----
--
//-----
//-----[ TAREA 4 ]-----

```

```

//-----
//-----
--
void Tarea4( void * parameter){

    for( int i = 0;i<12;i++ ){          // Número de veces que ocurre esta tarea

        int valor = 0;                  //Contiene el estado del LED

        //Recepción de un dato de la cola de comunicación solo si hay
datos
        if(xQueueReceive(coladetareas, &valor, 0) == pdPASS){
            //Confirmación al usuario de que se ha recibido el dato
            Serial.println ("----- 04 [<- 3]"); //Tarea 4 recibe de 3

        }
        else{
            Serial.println("----- 04");// Evidencia de ocurrencia
        }

        vTaskDelay(2000);                // Espera...
    }
    Serial.println("--> 04");              // Conclusión de la tarea
    vTaskDelete( NULL );                 // Se "Borra"
}
//-----
--
//-----
//-----[ TAREA 5 ]-----
//-----
//-----
--
void Tarea5( void * parameter){
    for( int i = 0;i<13;i++ ){          // Número de veces que ocurre esta tarea
        Serial.println("----- 05");// Evidencia de ocurrencia
        vTaskDelay(2000);                // Espera...
    }
    Serial.println("--> 05");              // Conclusión de la tarea
    vTaskDelete( NULL );                 // Se "Borra"
}
//-----
--
//-----
//-----[ TAREA 6 ]-----
//-----
//-----
--
void Tarea6( void * parameter){
    for( int i = 0;i<10;i++ ){          // Número de veces que ocurre esta tarea

        if( i > 5){                      //Ocurrencia a la 6ta ejecución
            vTaskPrioritySet( NULL, 7 );//Cambio de prioridad a la tarea
            a=a+1;
            if(a > 1){                    //Espera al cambio de prioridad ya que
                                            //tarda una vuelta más
                Serial.println("----- 06"); //Se muestra al usuario la
nueva prioridad
            }
        }
    }
}

```


```

        }else{
            Serial.println("----- 06 [+]");//Ha cambiado de prioridad
        }
    }else{
        Serial.println("----- 06");// Evidencia de ocurrencia
    }

    vTaskDelay(2000);          // Espera...
}
Serial.println("--> 06");      // Conclusión de la tarea
vTaskDelete( NULL );          // Se "Borra"
}
//-----
--
//-----
//-----[ TAREA 7 ]-----
//-----
//-----
--
void Tarea7( void * parameter){
    for( int i = 0;i<10;i++){    // Número de veces que ocurre esta tarea
        if(i > 4){
            Serial.println("----- 07 [!]" ); //Tarea suspendida
            vTaskSuspend( NULL ); //Suspensión anticipada de la tarea
        }else{
            Serial.println("----- 07");// Evidencia de ocurrencia
        }
        vTaskDelay(2000);        // Espera...
    }
    Serial.println("--> 07");      // Conclusión de la tarea
    vTaskDelete( NULL );          // Se "Borra"
}
//-----
--
//-----
//-----[ TAREA 8 ]-----
//-----
//-----
--
void Tarea8( void * parameter){
    for( int i = 0;i<13;i++){    // Número de veces que ocurre esta tarea
        Serial.println("----- 08");// Evidencia de ocurrencia
        Serial.println("");
        vTaskDelay(2000);        // Espera...
    }
    Serial.println("--> 08");      // Conclusión de la tarea
    vTaskDelete( NULL );          // Se "Borra"
}
//*****
**

```


3. LISTA DE COMPONENTES USADOS

COMPONENTE	IMAGEN	DESCRIPCIÓN
Esp32		ESP32 es la denominación de una familia de chips SoC de bajo costo y consumo de energía, con tecnología Wi-Fi y Bluetooth de modo dual integrada. El ESP32 emplea un microprocesador Tensilica Xtensa LX6 en sus variantes de simple y doble núcleo e incluye interruptores de antena, balun de radiofrecuencia, amplificador de potencia, amplificador receptor de bajo ruido, filtros, y módulos de administración de energía.

4. CONCLUSIONES

Para concluir esta práctica es bueno mencionar que al llevarla a cabo pude observar las facilidades que nos da el sistema FreeRTOS al momento de programar ya que con unas líneas de código no tan difíciles de recordar se puede crear un sistema de tiempo real siendo este muy ligero y permitiéndonos usar temporizadores, semáforos, banderas y mutex facilitando al programador el control de las tareas que necesita para controlar las tareas, cosa que con programación tradicional llevaría más tiempo implementar así como recursos del el sistema al cual se va a subir el programa.

Esto nos llevó a la conclusión de que Free RTOS es una herramienta muy útil cuando se desea implementar sistemas de tiempo real sin tantas complicaciones y que no consuman tantos recursos.

5. BIBLIOGRAFÍA

- [1] FreeRTOS. (2020, 8 enero). FreeRTOS API categorías. <https://www.freertos.org/a00106.html>
- [2] FreeRTOS. (2018). The FreeRTOS™ Reference Manual (v10.0.0 ed.) [Libro electrónico]. https://www.freertos.org/fr-content-src/uploads/2018/07/FreeRTOS_Reference_Manual_V10.0.0.pdf
- [3] FreeRTOS. (2018). *Mastering the FreeRTOS Real Time Kernel A Hands On Tutorial Guide* [Libro electrónico]. https://www.freertos.org/fr-content-src/uploads/2018/07/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf