INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA ENINGENIERÍA Y TECNOLOGÍAS AVANZADAS

UPIITA

SISTEMAS OPERATIVOS EN TIEMPO REAL

3MV9

INTEGRANTES:

- Reyes García Mauricio Itzae
- Téllez Pérez Luis José

Sistema Operativo en Tiempo Real – FreeRTOS

Es un sistema operativo de tiempo real kernel para dispositivos embebidos que ha sido portado a 35 plataformas de microcontrolador y está distribuido bajo el MIT Licencia. Proporcionando métodos para múltiples subprocesos o hilos, mutexes, semáforos y temporizadores de software, además de soportar prioridades de hilos.

FreeRTOS está diseñado para ser pequeño y simple. El núcleo en sí consta de sólo tres archivos implementados en C. Para hacer que el código sea legible, fácil de portar y mantener, está escrito principalmente en C, pero hay un par de funciones en assembler que están incluidas en donde se necesitan principalmente en rutinas de planificación de arquitectura específica.

No hay ninguna de las funciones más avanzadas que se encuentran normalmente en sistemas operativos como Linux o Microsoft Windows, como controladores de dispositivos, administración avanzada de memoria, cuentas de usuario y redes. El énfasis está en la compacidad y la velocidad de ejecución. FreeRTOS se puede considerar como una 'biblioteca de hilos' en lugar de un 'sistema operativo'. Aunque la interfaz de línea de comandos y los complementos de abstracción de E / S similares a POSIX-like están disponibles.

Sistema operativo popular de código abierto para microcontroladores, con bibliotecas de software que permiten conectar de forma sencilla y segura sus pequeños dispositivos de poca potencia con los servicios en la nube de AWS como, por ejemplo, AWS IoT Core, o con otros dispositivos.

Los microcontroladores (MCU) constan de un solo chip que contiene un procesador simple y están presentes en muchos dispositivos, incluidos electrodomésticos, sensores, monitores de actividad física, sistemas de automatización industriales y automóviles. Muchos de estos pequeños dispositivos podrían beneficiarse de su conexión con la nube o conexión local con otros dispositivos. Por ejemplo, los medidores de la luz inteligentes necesitan conectarse a la nube para notificar el consumo, y los sistemas de seguridad de los edificios necesitan comunicarse localmente para que una puerta se abra al pasar una tarjeta por el lector correspondiente.

Los microcontroladores poseen una potencia de cómputo y una capacidad de memoria limitadas, y suelen desempeñar tareas sencillas y funcionales. A menudo, los microcontroladores funcionan con sistemas operativos que no tienen integrada ninguna funcionalidad para conectarse a redes locales o a la nube, lo que convierte a las aplicaciones de IoT en un desafío. FreeRTOS ayuda a solucionar este problema proporcionando tanto el sistema operativo esencial (para ejecutar el dispositivo de borde) como las bibliotecas de software que facilitan la conexión segura a la nube (o a otros dispositivos de borde) con el

fin de que pueda recopilar datos de los dispositivos para las aplicaciones de IoT y para tomar las medidas pertinentes.os de borde más potentes en los que se ejecute AWS IoT Greengrass.

Funcionalidades

Este Sistema Operativo implementa múltiples subprocesos haciendo que el programa anfitrión llame a un método de marcación de subprocesos a intervalos cortos y regulares. El método de marca de hilo cambia las tareas en función de la prioridad y un esquema de programación de turno rotativo. El intervalo habitual es de 1/1000 de segundo a 1/100 de segundo, a través de una interrupción de un temporizador de hardware, pero este intervalo a menudo se cambia para adaptarse a una aplicación en particular.

Está diseñado para ser pequeño y simple, su núcleo en sí consta de sólo tres archivos implementados en C. También cuenta con funciones en ensamblador que están incluidas en donde se necesitan principalmente en rutinas de planificación de arquitectura específica.

Tiene un modo tickless para reducir el consumo energético. En FreeRTOS las aplicaciones pueden ser asignadas de manera completamente estáticas. Alternativamente en RTOS los objetos pueden ser asignados de manera dinámica con cinco esquemas previstos de asignación de memoria:

1. Solo Asignar.
2. Asigna y libera con un algoritmo muy simple y rápido.
3. Asignar con un algoritmo más complejo pero rápido de asignación y liberación con coalescencia de memoria.
4. Asignar con una alternativa al esquema más complejo que incluye la coalescencia de memoria que permite dividir un montón en múltiples áreas de memoria.
5. Biblioteca C asignan y liberan con alguna protección de exclusión mutua.

Características

- Cuenta con libros y los manuales de referencia abiertos.
- Huella de memoria pequeña, lo que implica gastos indirectos bajos y ejecución rápida.
- Como opción, existe Tickless para utilizar baja potencia en aplicaciones.
- Destinado tanto para aficionados como para desarrolladores profesionales que trabajan en productos comerciales.

- El programador se puede configurar para operaciones tanto preventivas como cooperativas.
- Asistencia de Corutina (las CorRutines en FreeRTOS son tareas simple y ligeras con un uso limitado de la pila de llamadas)

Arquitecturas Soportadas

- Altera Nios II
- ARM architecture
  - ARM7
  - ARM9
  - ARM Cortex-M
  - ARM Cortex-A
- Atmel
  - Atmel AVR
  - AVR32
  - SAM3 / SAM4
  - SAM7 / SAM9
  - SAMD20 / SAML21
- Cortus
  - APS1
  - APS3
  - APS3R
  - APS5
  - FPS6
  - FPS8
- Cypress
  - PSoC
- Energy Micro
  - EFM32
- Espressif
  - ESP8266ex
  - ESP32
- Fujitsu
  - FM3
  - MB91460
  - MB96340
- Freescale
  - Coldfire V1 / V2
  - HCS12
  - Kinetis
- IBM
  - PPC404 / PPC405
- Infineon
  - TriCore
  - Infineon XMC4000
- Intel
  - x86
  - 8052
- Microchip Technology
  - PIC18 / PIC24 / dsPIC
  - PIC32
- Microsemi
  - SmartFusion
- Multiclet
  - Multiclet P1
- NXP
  - LPC1000
  - LPC2000
  - LPC4300
- Renesas
  - 78K0R
  - RL78
  - H8/S
  - RX600
  - RX200
  - SuperH
  - V850
- STMicroelectronics
  - STM32
  - STR7
- Texas Instruments
  - MSP430
  - Stellaris
  - Hercules (TMS570LS04 & RM42)
- Xilinx
  - MicroBlaze
  - Zynq-7000

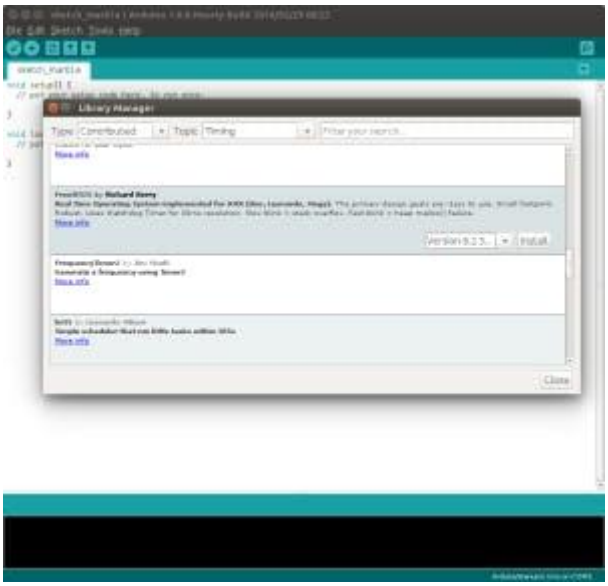
Estructura en lenguaje C de una tarea en FreeRTOS

```
void ATaskFunction( void *pvParameters )
{
/* Declara variables locales que utiliza la tarea*/
int iVariableExample = 0;

/* La tarea es definida dentro de un ciclo infinito. */
for( ;; )
{
/* El código que ejecuta la tarea está dentro del ciclo infinito. */
}
/* Si la terea sale del ciclo infinito, se entiende que finaliza las operaciones de la misma, por ello
esta debe borrarse
con la funcion de FreeRTOS vTaskDelete().
Esto libera todos los recursos que utilizaba dicha tarea. */
vTaskDelete( NULL );
}
```

Instalar FreeRTOS en Arduino

Primeramente en Arduino IDE Library manager, correspondiente a la Versión 1.6.8 o superior, buscar FreeRTOS Library bajo la secuencia: “Contributed” y tema: “Timing”.

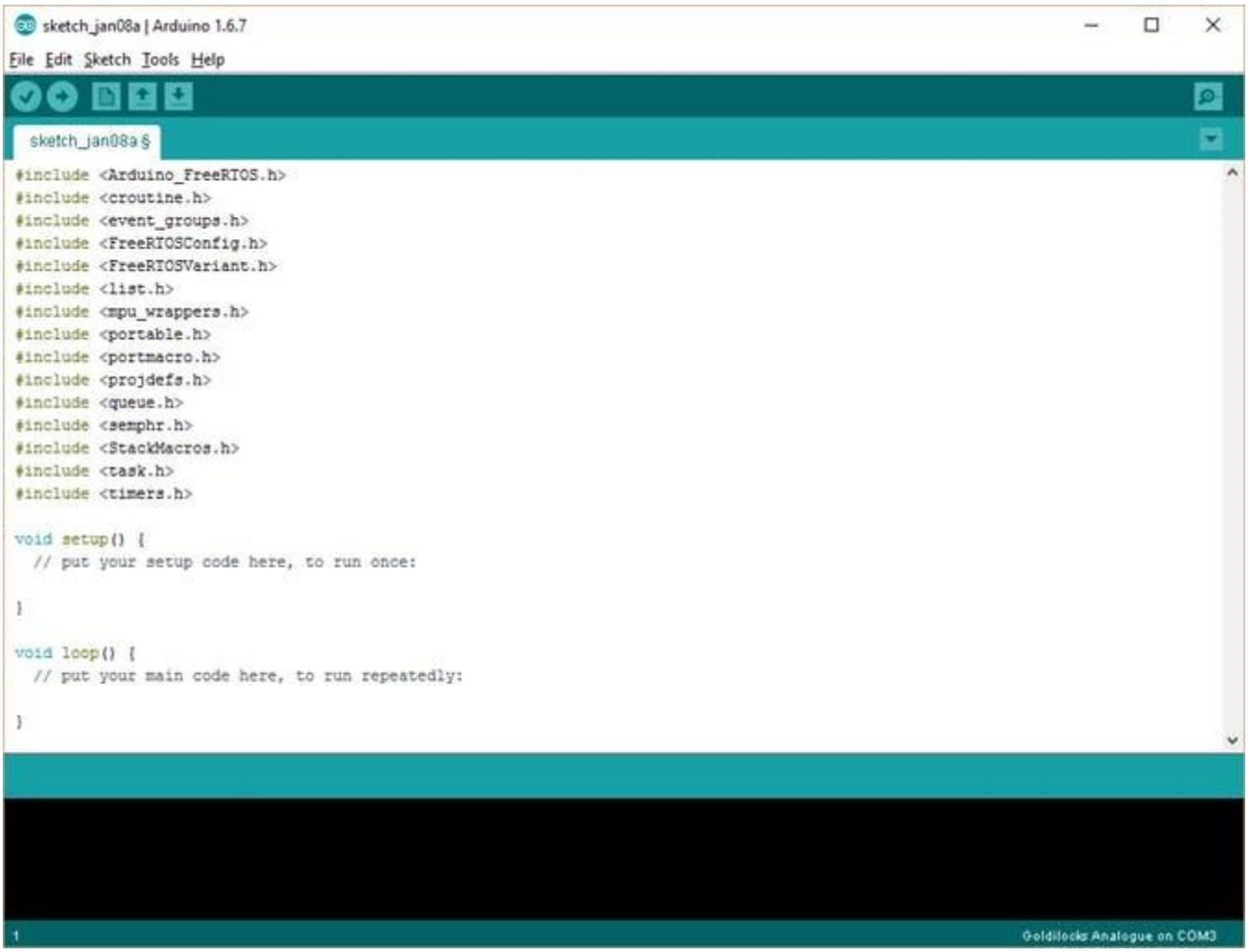


Buscando en Arduino Library Manager

Asegurarse de que la versión más reciente de la Librería FreeRTOS disponible está instalada.

FreeRTOS v8.2.3 Release 6 installed.

Después en el Sketch->Include Library menu, asegurarse de que la librería FreeRTOS está incluida en el sketch. Un Nuevo Proyecto debe lucir como en la siguiente imagen.



Nuevo Sketch con FreeRTOS Incluido.

Compilar y cargar este nuevo sketch en tu Arduino Uno/Yun/Leonardo/Mega o Goldilocks 1284p. Esto le mostrará la cantidad de flash que consume el programador de FreeRTOS. Como guía, la siguiente información se compiló usando Arduino v1.6.9 en Windows 10.

```
// Device:   loop() -> FreeRTOS | Additional Program Storage
// Uno:        444 ->   7018   |     20%
// Goldilocks:  502 ->   7086   |      5%
// Leonardo:   3618 ->  10166   |     23%
// Yun:        3612 ->  10160   |     23%
// Mega:        656 ->   7086   |      2%
```

En esta etapa, FreeRTOS ya se está ejecutando en su dispositivo.

Casos de uso

Aplicaciones industriales

Los clientes industriales utilizan dispositivos basados en microcontroladores que generan datos sobre las cargas de trabajo esenciales del negocio. Los sensores industriales, los accionadores, las bombas y los componentes de automatización utilizan microcontroladores por su bajo costo, su poca potencia y porque pueden desempeñar acciones en tiempo real. Por ejemplo, cada bomba de una plataforma petrolífera está controlada mediante un microcontrolador y es capaz de paralizar completamente la producción si se produce algún error.

FreeRTOS permite a estos clientes recopilar datos sobre el rendimiento y el desgaste del sistema mediante una conexión directa con la nube, así como tomar medidas importantes y locales en tiempo real con AWS IoT Greengrass, para impedir interrupciones semejantes a las comentadas anteriormente.

Productos de consumo

FreeRTOS puede ayudar a las empresas fabricantes de productos de consumo (como, por ejemplo, electrodomésticos, tecnologías vestibles o sistemas de iluminación inteligentes) a estandarizar el desarrollo, la comercialización y el mantenimiento de dispositivos basados en microcontroladores en una amplia gama de productos y modelos. FreeRTOS proporciona un solo sistema operativo para microcontroladores compatible con una gran variedad de hardware de microcontroladores, con una potencia y una capacidad variables. De este modo, las empresas pueden centrarse en innovar los productos, en lugar de administrar la complejidad del desarrollo de software en varias líneas de productos. Además, FreeRTOS ofrece la posibilidad de actualizar de forma segura las características de los productos de consumo actuales en el campo con la característica de actualización OTA.

Soluciones B2B (entre negocios)

Los microcontroladores se usan a menudo en los dispositivos comerciales B2B debido a sus requisitos de poca potencia y a su bajo costo. Por ejemplo, los fabricantes de equipos de seguridad incorporan cada vez más funciones de conectividad a dispositivos basados en microcontroladores como los sistemas de cierre y de sensores de las puertas comerciales. Un ejemplo de este Sistema Operativo es Amazon FreeRTOS permite a estas empresas acelerar el lanzamiento de nuevos productos conectados al simplificar el proceso de diseño y desarrollo. El desarrollo se simplifica porque el AWS Partner Device Catalog le permite encontrar y adquirir rápidamente microcontroladores calificados de Amazone FreeRTOS. Los fabricantes también pueden aplicar parches de manera segura en cerraduras comerciales mediante la característica de actualización OTA.

Beneficios con el uso de FreeRTOS

- Proporciona una solución única e independiente para muchas arquitecturas y herramientas de desarrollo diferentes.
- Es conocido por ser confiable. La confianza está asegurada por las actividades emprendidas por el proyecto hermano SafeRTOS.
- Es rico en características y todavía está en continuo desarrollo activo.
- Tiene una ROM mínima, RAM y sobrecarga de procesamiento. Normalmente, una imagen binaria del kernel RTOS estará en la región de 6K a 12K bytes.
- Es muy simple: el núcleo del kernel RTOS está contenido en solo 3 archivos C. La mayoría de los muchos archivos incluidos en la descarga del archivo .zip se relacionan solo con las numerosas aplicaciones de demostración.
- Es realmente gratuito para su uso en aplicaciones comerciales (consulte las condiciones de la licencia para obtener más información).
- Cuenta con licencias comerciales, soporte profesional y servicios de portabilidad disponibles en forma de OPENRTOS de nuestro socio WITTENSTEIN sistemas de alta integridad.
- Tiene una ruta de migración a SafeRTOS, que incluye certificaciones para los sectores médico, automotriz e industrial.
- Está bien establecido con una base de usuarios grande y cada vez mayor.
- Contiene un ejemplo preconfigurado para cada puerto. No es necesario averiguar cómo configurar un proyecto, ¡solo descargue y compile!
- Tiene un excelente, monitoreado y activo foro de soporte gratuito.
- Tiene la seguridad de que el soporte comercial está disponible en caso de ser necesario.
- Proporciona amplia documentación.
- Es muy escalable, simple y fácil de usar.
- FreeRTOS ofrece una alternativa de procesamiento en tiempo real más pequeña y fácil para aplicaciones donde eCOS, Linux incorporado (o Real Time Linux) e incluso uCLinux no caben, no son apropiados o no están disponibles.

Socios de FreeRTOS

Bibliografía

https://www.freertos.org/

https://aws.amazon.com/es/freertos/?sc_channel=PS&sc_campaign=acquisition_LATAM&sc_publisher=google&sc_medium=ACQ-P%7CPS-GO%7CNon-Brand%7CDesktop%7CSU%7CIoT%7CFreeRTOS%7CLATAM%7CEN%7CText&sc_content=freertos_e&sc_detail=freertos&sc_category=IoT&sc_segment=293624790156&sc_matchtype=e&sc_country=LATAM&s_kwcid=AL!4422!3!293624790156!e!!g!!freertos&ef_id=EAIaIQobChMIt-aWrOOn4QIV3bfACh3-Dwd5EAAYASAAEgLYlfD_BwE:G:s

https://es.wikipedia.org/wiki/FreeRTOS

https://aprendiendoarduino.wordpress.com/tag/rtos/

http://www.coffeebrain.org/wiki/index.php?title=Arduino-UNO:_Pasos_para_implementar_FreeRTOS_en_Arduino