Sistemas Embebidos

Proyecto Final
Manual técnico

Integrantes del grupo
Lorena Ferrando
Valentina Palladino
Julio Suaya

Docentes
Bruno Bellini
Felipe Estévez

Fecha de entrega: 24/07/2020

Indice

Introducción 2

Herramientas y requisitos 2

Arquitectura del software, módulos creados y funciones. 3

- Tareas implementadas 3
- Log de datos 4
- Send_sms 4
- Módulos provistos por la cátedra 4
- Main 5

Introducción

A continuación presentaremos un documento dirigido a desarrolladores. Aquí se detallará con lenguaje técnico aquellas ideas y aspectos necesarios para en un futuro continuar desarrollando, corregir o trabajar con el sistema diseñado.

Se hablará sobre los requisitos, tanto para utilizar el sistema como para continuar con su desarrollo, y las herramientas utilizadas.

Está destinado a aquellos desarrolladores con conocimientos de Sistemas Embebidos y de programación en C.

Herramientas y requisitos

Para poder utilizar el sistema es necesario contar con un microprocesador PIC32MM. También se debe contar con una tarjeta SIM (que no tenga activado el pin de bloqueo), poseer una antena GPS y realizar las conexiones necesarias.

El entorno de desarrollo utilizado fue MPLAB X IDE. Es distribuido gratuitamente por Microchip y se puede descargar desde www.microchip.com. La versión utilizada para el desarrollo fue v5.40.

El compilador utilizado fue XC32 v2.41. Al igual que el entorno de desarrollo se puede descargar de la página de Microchip, siguiendo este enlace www.microchip.com/mplab/compilers.

Se utilizó Hércules, un software terminal para poder enviar y recibir mensajes por medio de la conexión USB. La versión utilizada de Hércules fue v3.2.8.

El lenguaje de programación utilizado fue C.

El sistema operativo en tiempo real utilizado fue el freeRTOS.

Arquitectura del software, módulos creados y funciones.

La arquitectura de software es el diseño de más alto nivel de la estructura de un sistema.

La arquitectura de este sistema, se basa en la implementación de ciertas tareas fundamentales, que trabajan en un multitasking preemptivo. Se utiliza el Sistema Operativo en tiempo real **FreeRTOS**.

Tareas implementadas

- **vTaskUSB:** Esta tarea se encarga de gestionar dos semáforos, los cuales son utilizados para la sincronización de enviar y recibir datos por el puerto serial. La prioridad de esta tarea es baja (prioridad 1), debido a que se encuentra continuamente en ejecución y esperando, por lo que si su prioridad fuera muy alta, sería muy bloqueante.

- **vTaskMenu:** Esta tarea se encarga de gestionar la UI, es decir, administrar que es lo que se envía y se recibe mediante el puerto serial. Es de prioridad alta en comparación a las otras tareas, ya que es de gran importancia que la interacción entre el usuario y el sistema sea rápida, sin tener grandes esperas. La mayor parte del tiempo está esperando por semáforos.

- **vTaskBoton:** Esta tarea de prioridad 1 es la encargada de decidir qué hacer si se presiona el botón. La tarea está atenta a una flag, la cual se levanta cuando hay una interrupción generada al apretar el botón, y en base a eso, entrega el semaforo “semaforo_btn”. Este semáforo es el que permite que se comience a tomar la temperatura en la tarea vTaskTemperature. Si se vuelve a presionar el botón mientras se toma la temperatura (se levanta el flag nuevamente), se elimina la tarea en ejecución vTaskTemperature y se vuelve a iniciar.

- **vTaskTemperature:** Esta tarea es la encargada de tomar las mediciones de temperatura. La tarea inicialmente está esperando un semáforo para así empezar a tomar la temperatura. Luego de tomado el semáforo, se toman 10 medidas, se tratan los datos para así llevarlos a un rango de entre 320 y 420(que corresponden a 32 y 42 grados centígrados, pero se tratan asi para ahorrar memoria, ya que un float ocupa el doble que un uint16_t). Esta tarea tiene prioridad 4 ya que, es de suma importancia medir la temperatura en el momento necesario.

Además, gracias a que tiene varios Delays y esperas por semáforos, nos aseguramos que no evite que se ejecuten otras tareas fundamentales.

- **vTaskGPS:** Esta tarea de baja prioridad, se encarga de constantemente actualizar la trama del gps, utilizando funciones brindadas por el módulo SIM808. Tiene prioridad 1.

Log de datos

Las **medidas** definidas en el log cuentan con id de registro, lectura (valor de temperatura), latitud, longitud y fecha. Estos datos están almacenadas en variables del tipo cuyo tamaño (espacio que ocupan en memoria) sea menor.
Se cuenta con métodos para agregar medidas, descargarlas (visualizarlas) y borrarlas a todas.
Utilizamos un semáforo que es requerido para iniciar cualquiera de las tres tareas mencionadas, en orden de no realizarlas simultáneamente para evitar fallos.

En el log se almacenan todas las medidas con trama válida, con un máximo de 200.

Las medidas se agregan desde la tarea **vTaskTemperature**. Para descargarlas o borrarlas, se hace desde la tarea **vTaskMenu.**

Send_sms

Provee las funcionalidades para, a partir de una medida recibida, armar el mensaje a enviar. Obtiene los datos de la fecha, hora y ubicación gracias a los módulos provistos del GPS.

La funcionalidad de enviar mensaje es llamada cuando en la tarea **vTaskTemperature** se detecta una medida que sobrepasa la temperatura de umbral (hay fiebre).

Módulos provistos por la cátedra

Los siguientes módulos fueron utilizados pero no desarrollados por nosotros mismos. Los detallamos a grandes rasgos:

- SIM808: provee funcionalidades para el envío de SMS.
- WS2812: provee las funcionalidades para manipular los LEDS RGB de la placa.
- GPS: provee las funcionalidades para obtener la ubicación de la placa.

Main

En el main sucede:

1. Se inicializa el sistema.
2. Se inicializa el USB. Esto quiere decir que se crean semáforos binarios para el envío y recepción de datos del USB.
3. Se inicializa el Botón, esto significa que se crea un semáforo binario y se setean en 0 las variables que se utilizan para corroborar si el botón fue presionado o vuelto a presionar.
4. Se inicializa Temp, lo único que hace es setear el valor por defecto de temperatura umbral.
5. Se inicializa el Log, por lo que crea el semáforo mutex del log.
6. Se crean las tareas: vTaskUSB, vTaskMenu, vTaskBoton, vTaskTemperature, vTaskGPS, SIM808_taskCheck, SIM808_initModule.
7. Por último se inicializa el Scheduler.