Smart Plug

FIRMWARE

Arquitectura

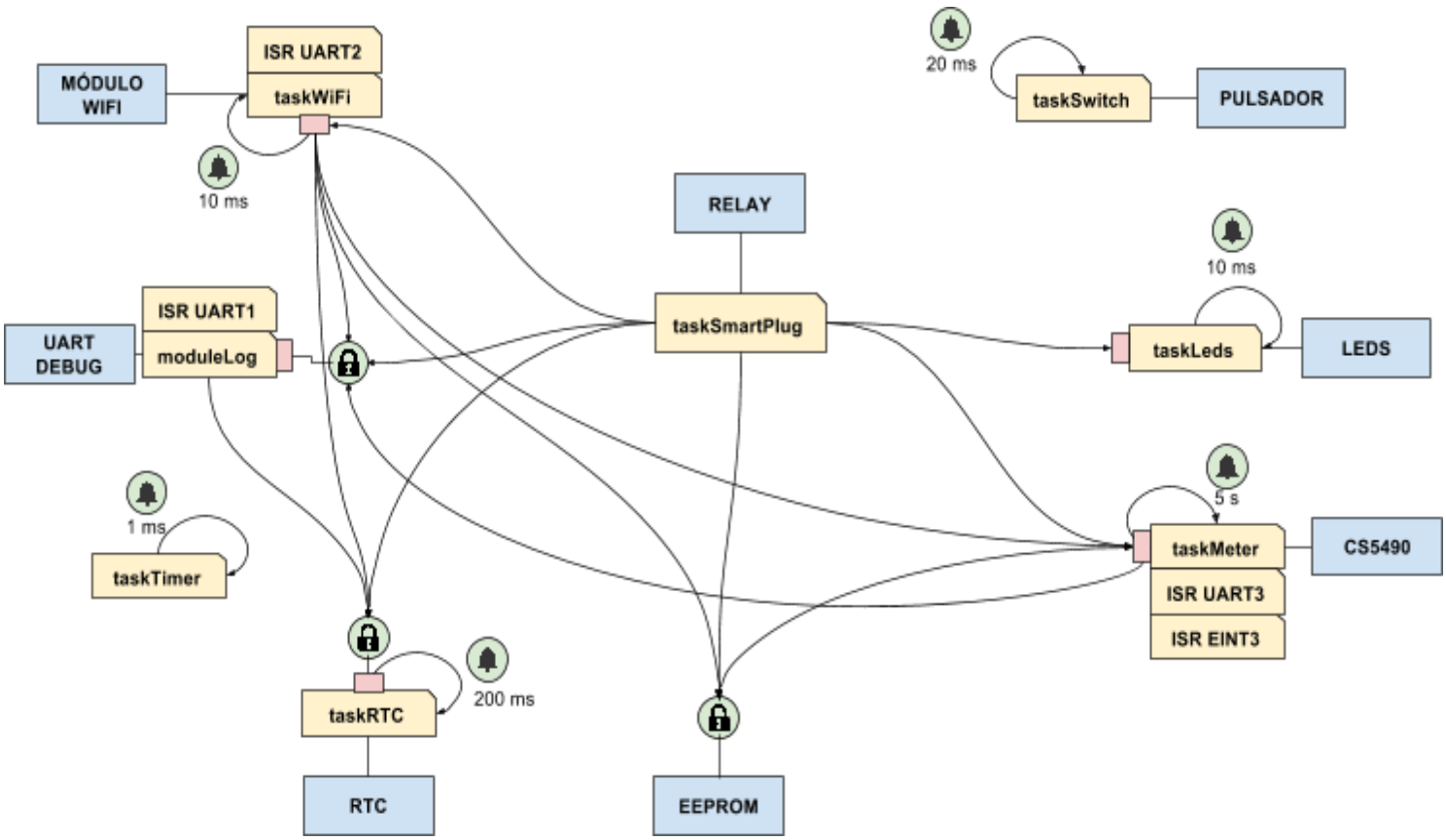
Autor
Ing. Mariano Mondani

Tabla de contenidos

Registro de cambios

Revisión	Cambios realizados	Fecha
1.0	Creación del documento	13/08/2016

Arquitectura del firmware



Descripción de las tareas

taskSmartPlug

Descripción

Es la tarea principal del Smart Plug. Se encarga de implementar la lógica del dispositivo. Se encarga de realizar las señalizaciones con los leds verde y rojo de acuerdo a la siguiente tabla:

	Modo Soft-AP	Modo WPS	Autenticado	RTC Sincronizado
Verde destella a 2Hz	1	X	X	X

Verde destella a 1Hz	0	1	X	X
Rojo destella a 1Hz	0	0	0	X
Verde y rojo destellan	0	0	1	0
Verde encendido	0	0	1	1

Es una tarea que se bloquea esperando eventos.

Interacción

- Hardware: con el relay conectado en el pin P2[7].
- Software:
  - EEPROM: guarda periódicamente las mediciones de energía.
  - taskWiFi: le indica que comience el servicio de WPS o que se configure como Soft-AP.
  - taskLeds: a partir de los eventos que recibe, le indica a la tarea que realice las señalizaciones.
  - taskCS5490: obtiene las mediciones actuales para guardarla en la memoria EEPROM.
  - moduleLog

Eventos generados

-

Eventos recibidos

- evAuthenticated: señaliza de acuerdo a la tabla.
- evDeAuthenticated: señaliza de acuerdo a la tabla.
- evRTCNoSynch: señaliza mediante el led verde y rojo destellantes.
- evRTCSynch: señaliza de acuerdo a la tabla.
- evSwitch: iniciar el proceso de WPS. Señaliza de acuerdo a la tabla.
- evSwitch_5sec: iniciar el módulo WiFi en modo Soft-AP. Señaliza de acuerdo a la tabla.
- evRelayOn: encender la carga manipulando el relay.
- evRelayOff: apagar la carga manipulando el relay.
- evRTC_1min: chequea si hay que apagar o encender la carga comparando la hora actual con la configurada para el día de la semana. Toma la medición de potencia activa y la va promediando para sacar un promedio por hora.
- evRTC_1hour: guarda las mediciones de energía de la hora, la acumula en el día y la acumula en el acumulador general; la medición promedio de la potencia activa.
- evRTC_1day: actualiza los punteros de la EEPROM: de potencia activa por hora, de energía por hora y de energía por día.
- evNewConn: no se hace ninguna acción con este evento.
- evCloseConn: no se hace ninguna acción con este evento.
- evResetEnergy: le indica al taskMeter que resetee los pulsos de energía.
- evChangeOnOffTime: vuelve a leer de la EEPROM los horarios de encendido y apagado de la carga para cada día.
- evFactoryReset: indica que debe volver a los valores de fábrica para todos los parámetros.

Recursos

- resLog
- resRTC
- resEEPROM

Interfaz

- **void** *taskSmartPlug_init* (**void*** **_**eeprom)
- uint32_t getLoadState (**void**)

taskSwitch
Descripción Gestiona el pulsador de tipo tact switch que se encuentra en el PCB. Realiza el anti-rebote y genera los eventos cuando es presionado. Va a generar un evento cuando se suelte el pulsador (evSwitch) y otro cuando se mantenga presionado por más de 5 segundos (evSwitch_5sec). Es una tarea de ejecución periódica.
Interacción ● Hardware: un pulsador de tipo tact switch conectado en el pin P0[3]. Es activo alto. ● Software: no interacciona con otras tareas.
Eventos generados ● evSwitch: al soltar el pulsador. ● evSwitch_5sec: al mantener presionado el pulsador por más de 5 segundos.
Eventos recibidos -
Recursos ● alarmRunTaskSwitch: cada 20 milisegundos.
Interfaz ● **void** *taskSwitch_init* (**void**)

taskLeds
Descripción Se encarga de realizar las señalizaciones indicadas por otras tareas. Las señalizaciones se forman a partir de un led bicolor (rojo y verde), tanto encendiéndolos, apagándolos o haciéndolos destellar a distintas frecuencias. Es una tarea periódica para actualizar el valor de los leds.
Interacción ● Hardware: led bicolor conectado a los pines: P0[21] (verde) y P2[13] (rojo). Se debe poner un nivel alto en el pin para encender el led.

- Software: no interacciona con otras tareas.

Eventos generados

-

Eventos recibidos

-

Recursos

- alarmRunTaskLeds: cada 10 milisegundos.

Interfaz

- **void** *taskLed_init* **(void)**
- **void** *blinkLed* (uint32_t ledID, uint32_t tOn, uint32_t tOff)

  ledID: constante que indica qué led se está configurando.
  tOn: tiempo de encendido del led (en milisegundos).
  tOff: tiempo de apagado del led (en milisegundos)

  Nota: para encender un led se debe configurar tOn != 0 y tOff == 0. Para apagarlo, tOn == 0 y tOff != 0.

taskRTC

Descripción

Se encarga de configurar y acceder al RTC interno del microcontrolador.
Es una tarea periódica

Interacción

- Hardware: módulo RTC interno del microcontrolador.
- Software: no interacciona con otras tareas.

Eventos generados

- evRTC_1min: se genera cada vez que pasa 1 minuto en el RTC.
- evRTC_1hour: se genera cada vez que pasa 1 hora en el RTC.
- evRTC_1day: se genera cada vez que pasa 1 día en el RTC.

Eventos recibidos

-

Recursos

- alarmRunTaskRTC: cada 200 ms.
- resRTC

Interfaz

- **void** *taskRTC_init* **(void)**
- **void** *setTime* (rtc_type_t* fullTime)

  fullTime: estructura que contiene la hora y la fecha para configurar en el RTC del microcontrolador.
- **void** *getTime* (rtc_type_t* fullTime)

fullTime: estructura donde volcar la hora y la fecha leída del RTC.

- uint32_t *isTimeSynchronized* (**void**)
  Retorna: 0 si no está sincronizado.
  1 si está sincronizado.

taskTimer
Descripción Tarea usada por las clases que necesitan generar una base de tiempo. Por ejemplo la clase cTimer. Es una tarea periódica.
Interacción -
Eventos generados -
Eventos recibidos -
Recursos ● alarmRunTaskTimer: cada 1 milisegundo.
Interfaz ● **void** *taskTimer_init* **(void)**

moduleLog
Descripción Se encarga de enviar por la UART1 los mensajes que son enviados por otros tasks para ayudar en el debug de la aplicación.
Interacción ● Hardware: con la UART1. ● Software: ○ taskRTC: cuando debe loguear un evento, solicita la fecha y la hora a la tarea taskRTC.
Eventos generados -
Eventos recibidos -
Recursos

- ISR UART1: usada por el driver de la UART (ioUART).

Interfaz

- **void** *moduleLog_init* **(void)**
- **void** log (uint8_t* str)
  str: puntero al string que se debe loguear.

taskWiFi

Descripción

Se encarga de recibir las conexiones desde la aplicación móvil. Va a interaccionar con otras tareas para enviar los datos pedidos por la app móvil o para configurar el Smart Plug de acuerdo a lo indicado por la app móvil.
La tarea es periódica para controlar lo que llega desde el módulo RN1723 de forma asincrónica.

Interacción

- Hardware: con el módulo RN1723 a través de:
  - Comunicación: UART2
  - Pin reset: P2[2]
  - Pin IP_ON: P2[4]
  - Pin TCP_ON: P2[5]
- Software:
  - EEPROM: accede a la memoria para escribir y leer parámetros de configuración del Smart Plug. También obtiene los valores históricos de las mediciones de los parámetros eléctricos.
  - taskRTC: el módulo WiFi obtiene la hora desde un servidor NTP y le indica al taskRTC que configure la fecha y la hora a partir del dato obtenido.
  - taskMeter: consulta los parámetros eléctricos actuales medidos por el CS5490.
  - moduleLog

Eventos generados

- evAuthenticated: cuando se logra unir a una red WiFi.
- evDeAuthenticated: cuando sale de una red WiFi.
- evRTCNoSynch: no se pudo sincronizar con el servidor NTP.
- evRTCSynch: se pudo sincronizar con el servidor NTP.
- evNewConn: cuando se inicia una nueva conexión con la aplicación móvil.
- evCloseConn: cuando se cierra la conexión con la aplicación móvil.
- evRelayOn: para encender la carga.
- evRelayOff: para apagar la carga.
- evResetEnergy: para resetear el acumulador de energía.
- evChangeOnOffTime: indica que hubo un cambio en algún valor de encendido o apagado. Este valor ya está actualizado en la EEPROM.
- evFactoryReset: indica que debe volver a los valores de fábrica para todos los parámetros.

Eventos recibidos

-

Recursos

- resEEPROM
- alarmRunTaskWiFi: cada 10 milisegundos.
- ISR UART2: interrupción de la UART2 usado por el driver de la UART (ioUART).
- resLog
- resRTC

Interfaz

- **void** *taskWiFi_init* **(void*** _eeprom**)**
- **void** *initWPS* (**void**)
- **void** *initWebServer* (**void**)
- uint32_t *isAuthenticated* (**void**)
  Retorna: 0 si no está unido a una red WiFi.
  1 si está unido.
- uint32_t *isTCPConnected* (**void**)
  Retorna: 0 si no hay una conexión TCP abierta
  1 si hay una conexión TCP abierta.
- **void** *configureID* (uint8_t* id)

taskMeter

Descripción

Se encarga de obtener periódicamente las mediciones del front-end analógico CS5490. Además acumula los pulsos generados por el mismo integrado para llevar un registro de la energía consumida por la carga.
Es una tarea periódica.

Interacción

- Hardware: con el front-end analógico CS5490 a través de:
  - Comunicación: UART3
  - Reset: P1[30]
  - Pulsos: P0[27]
- Software:
  - moduleLog
  - EEPROM: al iniciar lee los valores de calibración. TO-DO.

Eventos generados

-

Eventos recibidos

-

Recursos

- alarmRunTaskMeter: cada 5 segundos.
- ISR UART3: interrupción de la UART3 usado por el driver de la UART (ioUART).
- ISR EINT3: interrupción por cambios en un pin digital. Captura los flancos negativos

que se producen en el pin P0[27] para contar los pulsos generados por el CS5490.
- resLog
- resRTC
- resEEPROM

Interfaz

- **void** *taskMeter_init* **(void*** _eeprom**)**
- float *getMeterValue* (uint32_t variableID)
  variableID: constante que identifica la variable de la que se quiere obtener el valor (Vrms, Irms, Frecuencia, Energía, Factor de potencia, Potencia activa, etc).
- **void** *resetEnergyPulses* (**void**)
  Resetea el contador de pulsos de energía.
- **void** *initCalibrationProcess* (uint32_t processID)
  processID: constante que identifica al proceso de calibración: V DC offset, I DC offset, V AC gain, I AC gain.