Estación de medición para análisis y control de parámetros ambientales

Juan Ignacio Sala[1], Omar E. Rodriguez[1], Hugo Mazzeo[1], José Rapallini[1]

[1] Proyecto de Investigación y Desarrollo - CODAPLI (Codiseño Hardware Software para Aplicaciones en Tiempo Real) – Departamento de Ingeniería en Sistemas de Información. Universidad Tecnológica Nacional. Facultad Regional La Plata, Calle 60 y 124, La Plata 1900, Argentina.
nachosala89@gmail.com {orodriguez, hugo.maz, rapalini}
@frlp.utn.edu.ar
http://codapli.frlp.utn.edu.ar

Resumen. Se presenta la implementación de un prototipo de Estación Meteorológica Automática (EMA) bajo la plataforma de Computadora Industrial Abierta Argentina en su versión educativa (EDU-CIAA). La EMA realiza la adquisición de datos meteorológicos y los transmite en tiempo real a una computadora mediante el puerto serie, para su visualización, análisis y almacenamiento.
Para el desarrollo del prototipo se utilizó la metodología de Codiseño Hardware/Software y el entorno de programación Micropython para la programación de la EDU-CIAA

Palabras clave: Codiseño Hardware/Software, Sistemas embebidos, estación meteorológica, EDU-CIAA, Micropython, OpenOCD

1 Introducción

El objetivo de este proyecto es el diseño e implementación de un prototipo de Estación Meteorológica Autónoma (EMA) que, a diferencia de una estación meteorológica tradicional, permite su instalación en zonas poco accesibles o inhóspitas, minimizando el trabajo humano requerido para su atención. Los datos meteorológicos recolectados por los distintos sensores que poseen pueden ser reportados en tiempo real a través de distintas redes de comunicaciones o almacenados para requerimiento posterior.

Una Estación Meteorológica Automática (EMA) posee un número muy amplio de aplicaciones, que van desde la simple observación de variables climáticas hasta la detección y prevención de incendios forestales.

En general, poseen un conjunto de sensores para obtener las distintas variables meteorológicas, un datalogger y posibilidades de telemetría. Además, en casos de esta-

ciones situadas en locaciones remotas, las mismas cuentan con fuentes de energía como paneles solares y baterías recargables.

Por otro lado, existen diversas opciones para que una EMA reporte los datos que recolecta, ya sea por una conexión local a una computadora, haciendo uso de enlaces inalámbricos o incluso satelitales.

Para el desarrollo del proyecto se ha utilizado la Metodología de Codiseño Hardware/Software (HW/SW).

Si bien existen EMA comerciales, sus costos suelen ser altos y en casos de fallas no siempre cuentan con un buen servicio técnico de atención al cliente. En general suelen tener placas electrónicas y software propietarios que sólo pueden ser atendidos por el fabricante. En este caso se propone un prototipo inicial básico, utilizando una placa diseñada por ingenieros e investigadores en nuestro país, equipado con sensores básicos pero que puede ampliarse tanto en el número de variables a medir, capacidades de comunicación y alimentación de energía.

1.1 EDU-CIAA NXP

El proyecto CIAA [1] cuenta hoy en día con más de 6 plataformas de Hardware. En el presente trabajo será utilizada la EDU-CIAA-NXP, que es una versión de bajo costo de la CIAA-NXP pensada para la enseñanza universitaria, terciaria y secundaria. La EDU-CIAA está basada en la CIAA-NXP, por ser la primera versión de la CIAA que se encuentra disponible. Por lo tanto, su microcontrolador es también el LPC4337 (dual core ARM Cortex-M4F y Cortex-M0).

Sin embargo, con el objetivo de abaratar costos y reducir su complejidad la EDU-CIAA incorpora sólo algunas de las funcionalidades de la CIAA. A su vez, con el fin de permitir el desarrollo de algunas prácticas sencillas sin que sea necesario recurrir a hardware adicional, incluye además algunos recursos que no están presentes en la CIAA.

2 Metodología de codiseño hardware/software

Según [2] el codiseño es "*el proceso de diseño de un sistema que combina las perspectivas hardware y software desde los estados primarios, para aprovechar la flexibilidad del diseño y la localización eficiente de las funciones*".

Esta metodología apunta a satisfacer los objetivos a nivel sistema haciendo uso de las interacciones entre el hardware y el software a partir del diseño en paralelo de ambos subsistemas.

La Metodología utilizada como referencia define las etapas que se describen a continuación:

— *Definición del problema a solucionar*: Una diferencia importante de los sistemas embebidos y de tiempo real con los sistemas de información más tradicionales es que los problemas que resuelven son más específicos y acotados y por lo tanto su conceptualización es en muchos casos más simple.

— *Especificación*: Con la información de la etapa anterior se determinan los requerimientos, entradas, salidas, interfaces con el medio que requiere la solución. En esta etapa se produce un documento que explicita los requerimientos funcionales y no funcionales del sistema.

— *Descripción formal del sistema*: Tiene como objetivo describir, independientemente de cualquier implementación, al sistema mediante un modelo formal del mismo.

— *Simulación*: Con los modelos que describen al sistema se realizan una serie de simulaciones que permitirán verificar los requerimientos y refinar el diseño del sistema.

— *Partición HW/SW*: Consiste en seleccionar una arquitectura y plataforma de implementación y asignar las tareas en los recursos elegidos para la implementación.

— *Cosíntesis*: Adaptación y mapeo de los modelos realizados en los elementos seleccionados para formar parte de la implementación.

— *Cosimulación*: Validación de la transformación realizada comprobando el cumplimiento de requerimientos.

— *Implementación*: En cuanto al hardware, se traslada lo descripto a los dispositivos elegidos, o se realiza la fabricación de los circuitos integrados necesarios. En cuanto al software, se realiza el desarrollo del mismo.

— *Verificación*: Integración del hardware y software y generación de un prototipo.

3 Estado del arte

En la actualidad, entre los vendedores más destacados se encuentran Davis Instruments, Texas Weather Instruments y AcuRite.

Existen diversos tipos de productos, que varían en función de su aplicación. El modelo más básico consiste en:

− La estación propiamente dicha, la cual se encarga de la captura de distintas variables mediante los sensores que la componen.

− La consola de visualización, que recibe los datos enviados por la estación y se encarga de presentarlos.

La comunicación entre la consola y la estación depende de la gama, el tipo de producto y el público objetivo del producto. La misma puede ser tanto cableada como inalámbrica.

Este tipo de productos típicamente incluye los siguientes sensores y elementos auxiliares:

- Temperatura
- Humedad
- Presión atmosférica
- Anemómetro
- Sensor de lluvia
- Panel para alimentar la estación
- Sensores de radiación solar

4 Desarrollo del sistema

4.1 Especificaciones y requerimientos

Partiendo del problema inicial planteado, que es el desarrollo del sistema de monitoreo meteorológico, se determinaron los siguientes requerimientos:

Funcionales:

- Adquisición de variables meteorológicas: temperatura, humedad relativa, presión atmosférica.
- Debe permitirse contar con datos de más de una estación.
- Cálculo de sensación térmica.
- Consultas en soft real time.
- Consultas históricas.

No Funcionales:

- En lo temporal, definimos un intervalo de actualización de 1 minuto.
- En cuanto a una consulta en tiempo real, su tiempo de respuesta debe ser inferior a los 5 segundos.
- Bajo consumo de energía eléctrica para las estaciones.

4.2 Descripción formal del sistema

En base a los requerimientos definidos, comenzamos a modelar el sistema. En la Fig. 1 observamos el diagrama de definición de bloques de SysML [3] a nivel sistema, donde establecemos a alto nivel como estará constituido el mismo. El sistema estará formado por dos partes, las estaciones y un servidor que almacenará y distribuirá la información.

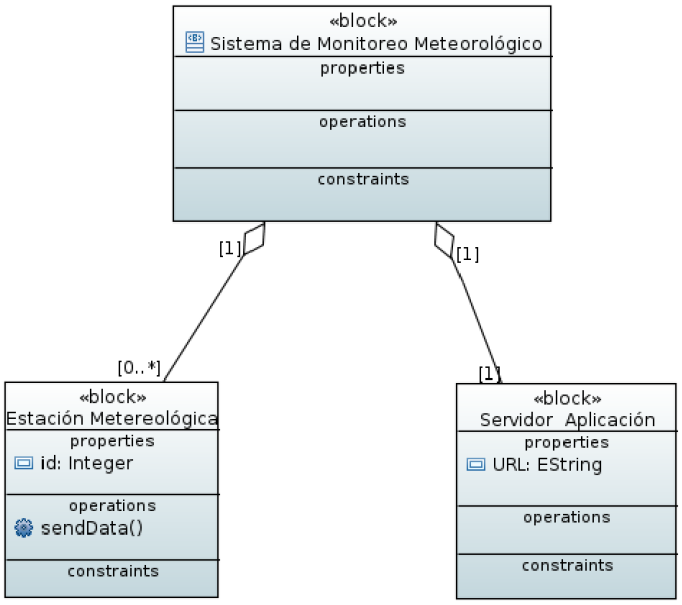

Fig. 1. Diagrama de Definición de Bloques

A su vez, como vemos en la Fig. 2 definimos el Diagrama de Bloques Interno de la Estación.

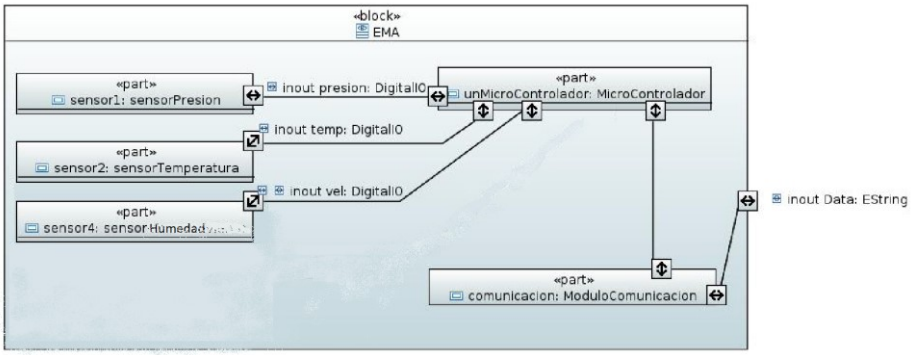

Fig. 2. Diagrama de definición de bloques internos

4.3 Simulación

Como parte del desarrollo se realizaron diversas simulaciones con el fin de analizar los resultados y obtener conclusiones a partir de ellos, para validar los objetivos y alcance definidos en el proyecto. De esta manera se realizaron aproximaciones sucesivas a la definición del alcance del prototipo.

Las herramientas utilizadas para las mismas fueron: Red de Petri, ADA y C. Las simulaciones realizadas con ADA y C incluyeron la utilización de una placa conectada al puerto paralelo de una PC.

Utilizando la Red de Petri (Fig. 3) modelamos el comportamiento del sistema en su conjunto, para el caso de una consulta en tiempo real, que consiste en una operación de consulta sincrónica que comienza con una solicitud de un cliente, lo que genera una serie de pasos en los que el servidor envía la consulta a la estación, la misma adquiere los datos de los sensores y los devuelve al mismo, que se encarga de realizar la respuesta para el cliente.

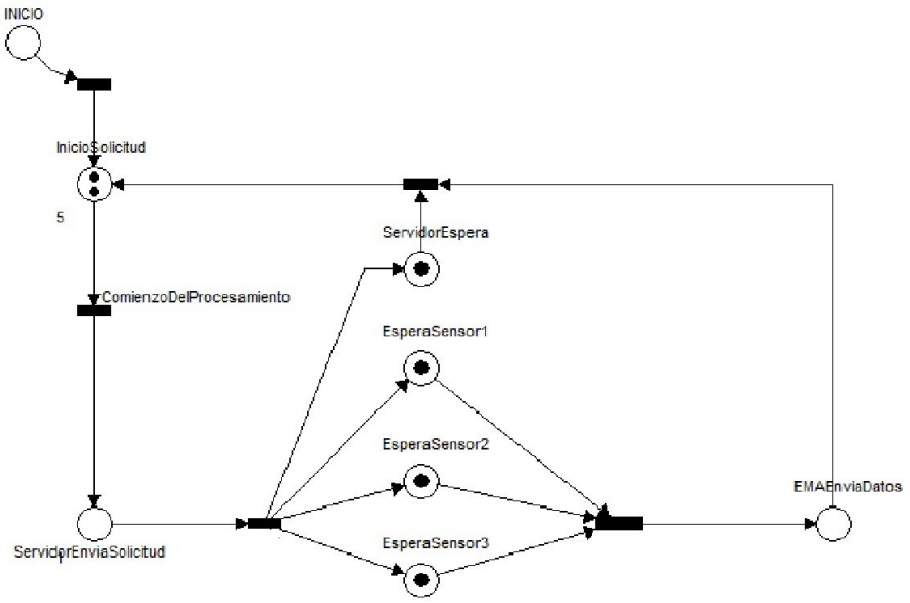

Fig. 3. Red de Petri

4.4 Particionamiento hardware/software

Teniendo claros los requerimientos y una especificación formal del sistema, procedemos a definir la plataforma sobre la cual se realizará la implementación del prototipo.

Hardware:

1. EDU-CIAA-NXP
2. Sensor de presión barométrica BMP180
3. Sensor de temperatura y humedad DHT22
4. Servidor. Procesador IntelCore i3 + 4GB de RAM

Software:

1. Ubuntu 16.04: Sistema operativo para el Servidor
2. Python 2.7.x
3. Firmware e IDE de Micropython para CIAA [4]
4. OpenOCD [5]
5. Framework Web Python Flask
6. MongoDB
7. Librerías Javascript
    (a) D3
    (b) socketIO
    (c) epoch

Para la construcción de este prototipo por razón de costos se optó por incluir sólo la medición de temperatura, presión barométrica y humedad. Pero el prototipo es fácilmente expandible a otro tipo de sensores.

Otra decisión de diseño tuvo que ver con la comunicación entre la estación y el servidor. En este primer prototipo se optó por utilizar el puerto serie.

4.5 Implementación

Por un lado, se realizó la implementación de la estación, que consistió en la conexión de los sensores y la programación del código necesario para la captura y envío de los datos al servidor.

Fig. 4. EDU-CIAA-NXP y conexión a sensores.

Se implementó un esquema bajo el cual la estación está en comunicación constante, enviando los datos en función de un intervalo de tiempo que según los requerimientos es de 1 minuto.

El sensor de presión barométrica BMP180 se comunica utilizando el bus I2C, mientras que el sensor de temperatura y humedad DHT22 usa un protocolo no estándar y envía 40 bits, utilizando 16 bits para la temperatura, otros 16 bits para la humedad y los 8 bits restantes para un checksum.

Las librerías que se implementaron para controlar los sensores han sido adaptadas de librerías existentes en la Web para Micropython, ya que la implementación de Micropython de la CIAA no es completa y algunas funciones no pueden utilizarse.

El servidor se había implementado previamente para una aplicación similar para Arduino, utilizando Python y el framework Flask [6], y se reutilizó para este proyecto con el objetivo principal de visualizar los datos. Este framework se define como minimalista, provee un servidor web de desarrollo integrado y está basado en Werkzeug. La integración con una base de datos se realiza a través de extensiones y utiliza Jinja2 como sistema de templates.

La estructura de la aplicación web es la siguiente:

- Home
    - Consulta en Tiempo Real
    - Ingreso de límites temporales
    - Consulta Histórica

En el caso de las consultas en tiempo real (Fig.5), cuando el servidor recibe la solicitud HTTP correspondiente, además de responder con el contenido estático de la página, crea un thread que se encarga de leer el puerto serie al cual está conectada la estación, interpretar los datos y emitirlos a través de socketIO.

Luego, el cliente recibe esos datos utilizando la versión cliente de socketIO, y almacena los mismos en una variable que es utilizada para mostrar los valores de temperatura, humedad y presión atmosférica en el browser.

Fig. 5. Visualización en Navegador Cliente.

SocketIO permite establecer una conexión bidireccional entre un servidor HTTP y un cliente utilizando TCP como transporte.

En el caso de las consultas históricas, una vez que el cliente determina los límites temporales, se obtienen los datos almacenados en la base de datos y se le responde con el recurso solicitado.

En cuanto al consumo eléctrico de la EMA estará dado fundamentalmente por la corriente consumida por la EDU-CIAA, encargada del procesamiento de las señales y la comunicación con el servidor. Si bien la hoja de datos no aporta información al respecto, algunas mediciones realizadas dieron consumos por debajo de los 50 mA. Considerando este valor como consumo máximo, utilizando un pack de pilas recargables de 2800 mAh para la alimentación, nos daría una duración aproximada de 2800 mAh/50 mA= 56 hs, es decir poco más de dos días. El consumo de energía diario, teniendo en cuenta una alimentación de 5 V, sería de 50 mA x 5V x 24 h = 6W/día, cantidad perfectamente recuperable con un panel solar de dimensiones reducidas.

Respecto a los tiempos de respuesta, el mayor retardo está dado por el sensor DHT22 cuando mide temperatura, ya que el manual especifica un valor típico menor a 10 segundos, que es lo que tarda en reflejar un cambio real de temperatura en el entorno. Este tiempo es bastante inferior al intervalo de actualización de un minuto definido en los requerimientos, por lo que, aun sumando los retardos introducidos por el procesamiento y la comunicación con el servidor, estaríamos por debajo de dicho intervalo.

En cuanto a la visualización de datos en pantalla, el acceso a la base de datos y el refresco de pantalla sus tiempos de respuesta han sido menores a los 3 segundos.

5 Conclusiones

Las distintas opciones de hardware libre de bajo costo que ofrece el mercado en la actualidad, combinado con la gran cantidad de herramientas software disponible, permiten desarrollar soluciones de codiseño hardware/software para una gran variedad de aplicaciones.

Este simple prototipo sólo se ocupa de obtener, almacenar y visualizar algunas variables climáticas. Pero con simples modificaciones se podría extender su funcionalidad a través de hardware (agregando sensores para obtener más variables, o dispositivos para proveer una conexión inalámbrica) o de software (proveer un sistema de pronósticos, alarmas, datos estadísticos, visualización de datos en dispositivos móviles mediante aplicaciones nativas).

También se probó la potencialidad de la EDU-CIAA para este tipo de aplicaciones, que combinada con el entorno de Micropython, provee una alternativa robusta y sencilla para la realización de proyectos de pequeña envergadura o para la realización de prototipos.

La EMA también puede ser adaptada físicamente para funcionar en donde se la requiera. Al ser utilizada necesariamente en espacios exteriores, dependiendo del entorno puede requerir protección contra fenómenos naturales o sociales (como hurto o vandalismo).

Por lo tanto, el sistema mostrado a través de este prototipo es flexible, adaptable y además escalable, ya que el servidor también podría obtener datos de varias estaciones concurrentemente si se lo deseara.

Referencias

1. Computadora Industrial Abierta Argentina http://proyecto-ciaa.com.ar/
2. Franke D.W., Purvis M.: Hardware/Software Codesign: A Perspective. In: Proceedings of the 13th Conference on Software Engineering, Austin-Texas-USA (1991).
3. Mischkalla, F., He, D., Müller, W.: Closing the gap between UML-based modeling, simulation and synthesis of combined HW/SW systems. In: Design, Automation & Test in Europe Conference & Exhibition (2010) 1201-1206
4. MicroPython, a lean and efficient implementation of the Python 3 https://micropython.org/
5. Free and Open On-Chip Debugging, In-System Programming and Boundary-Scan Testing http://openocd.org
6. Flask microframework for Python http://flask.pocoo.org