Diseño de un Sistema de Monitoreo Meteorológico Utilizando la Metodología de Codiseño Hardware/Software

Matías Martínez[1], Juan Ignacio Sala[2], José Rapallini[3], Hugo Mazzeo[4].

{martinezamatias, nachosala89, josrap}@gmail.com, hhmgvm@yahoo.com.

[1 2] Alumnos de la cátedra Aplicaciones en Tiempo Real.
[3 4] Docentes, CODAPLI (Proyecto de Investigación en Codiseño Hardware Software para Aplicaciones en Tiempo Real).
[1 2 3 4] Universidad Tecnológica Nacional, Facultad Regional La Plata., La Plata, Argentina.

Resumen. En este artículo se presenta un proyecto realizado para la cátedra Aplicaciones En Tiempo Real que consiste en el desarrollo de un prototipo de un Sistema de Monitoreo Meteorológico.

Mediante la Metodología de Codiseño Hardware/Software se llegó a la implementación del mismo. El sistema posee una arquitectura distribuida que cumple con los requerimientos propuestos en las fases iniciales del desarrollo, y que consisten en la adquisición, transporte y consumo de los datos.

El prototipo implementado consta de 2 subsistemas principales, el dispositivo de adquisición de datos, que está basado en una placa Arduino MEGA y un servidor de aplicación que permite el almacenamiento y el acceso a los datos a distintos clientes, tanto en tiempo real como de forma histórica.

Keywords: Codiseño Hardware/Software, Sistemas Embebidos, Sensores, HTTP, Arduino.

1 Introducción

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 estaciones 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 HW/SW. Ésta 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. A continuación se realiza una breve descripción de la misma.

2 Metodología De Codiseño Hardware-Software

Según [1] 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".

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 y humedad
- Anemómetro
- Sensor de lluvia
- Panel para alimentar la estación
- Sensores de radiación solar

4 Desarrollo del Sistema

4.1 Especificación 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, velocidad y dirección del viento.
- 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 a nivel sistema, donde establecemos a alto nivel como estará constituido el mismo[2]. El sistema estará formado por dos partes, las estaciones y un servidor que almacenará y distribuirá la información.

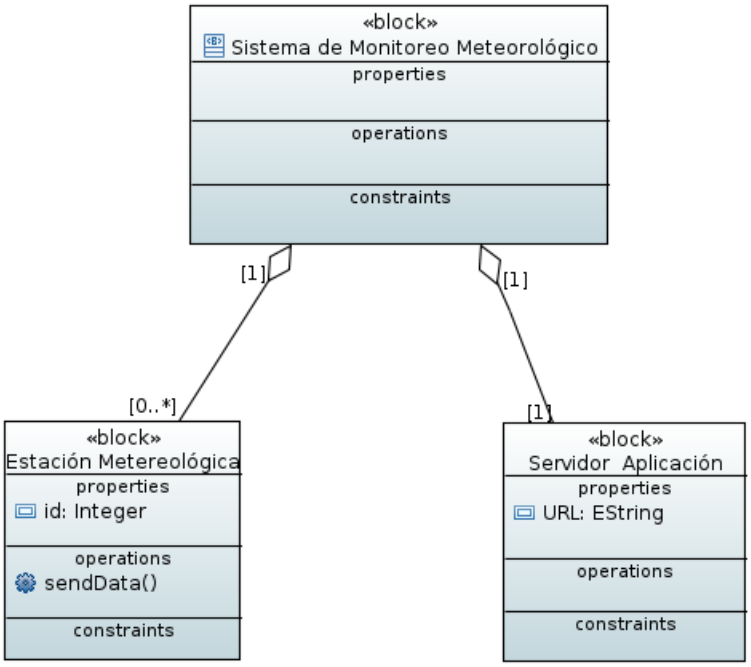

Fig. 1. Diagrama de Definición de Bloques

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

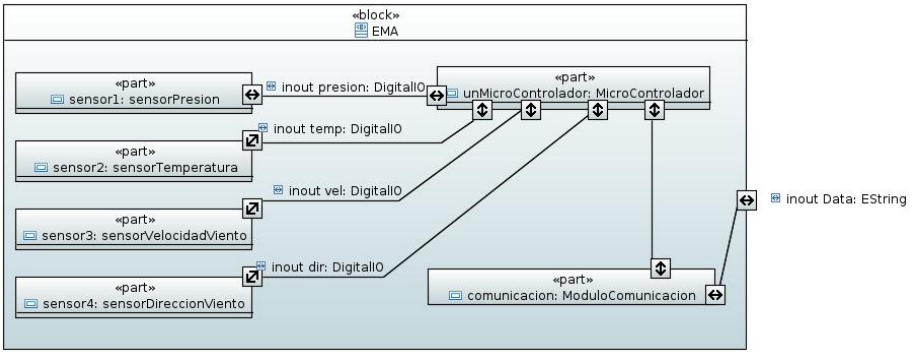

Fig. 2. Diagrama de Bloques Interno de la Estación.

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[1] 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.

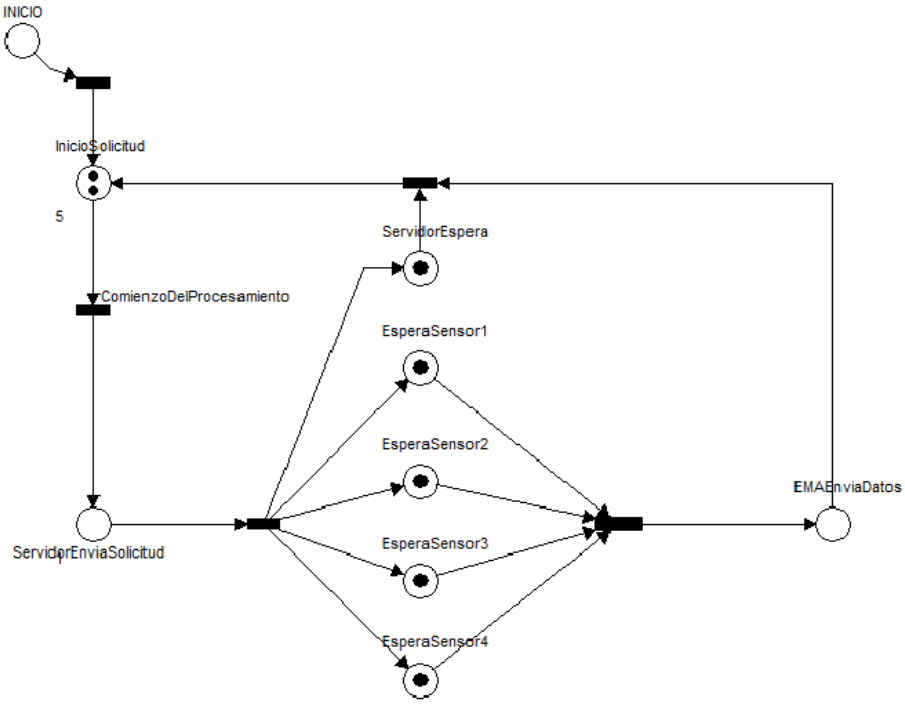

Fig. 3. Red de Petri.

4.4 Partición 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.

[1] Código fuente disponible en: https://github.com/matias-martinez/ada-tasks-lpt/commit/57c96ebfc10f65bc6c131363df17c316615f60a3 .

Hardware:

1. Arduino MEGA
2. Sensor de presión barométrica BMP180
3. Sensor de temperatura y humedad DHT22
4. Servidor. Procesador IntelCore i3 + 4GB de RAM

Software:

1. Arch Linux: Sistema operativo para el Servidor
2. Python 2.7.x
3. Framework Web Python Flask
4. MongoDB
5. Librerías Javascript
   (a) D3
   (b) socketIO
   (c) epoch

De la partición seleccionada puede notarse la ausencia de sensores de velocidad y dirección del viento. Se decidió limitar las variables medidas porque el objetivo es obtener un prototipo económico y en el menor tiempo posible.

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

Todo el código generado para la implementación, tanto de Arduino como del Servidor, se pueden encontrar en [3].

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).

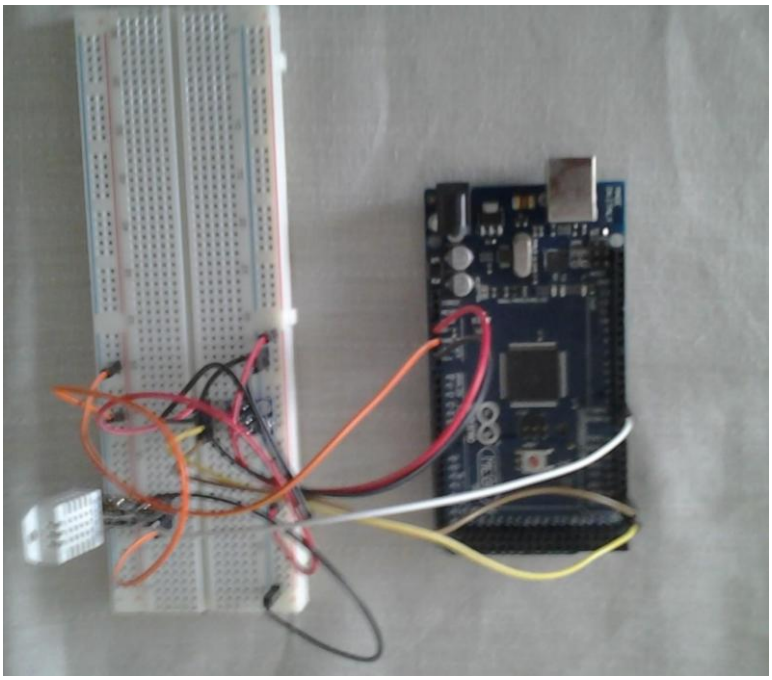

Fig. 4. Arduino y sensores.

Si bien se habían realizado simulaciones en las cuales la comunicación entre el servidor y la estación se realizaba sólo cuando se recibía una solicitud, 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 I^2C, 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.

El servidor, se implementó utilizando Python y el framework Flask [4]. Este framework se define como minimalista, provee un servidor web de desarrollo integrado y está basado en Werkzeug [5]. La integración con una base de datos se realiza a través de extensiones y utiliza Jinja2 [6] 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 cuál está conectada la estación, interpretar los datos y emitirlos a través de socketIO:

```
@app.route('/realtime')
def realtime():
  global thread
  if thread is None:
    thread = Thread(target=background_thread)
    thread.start()
  return render_template('realtime.html')

def background_thread():
  serial = Arduino('/dev/ttyACM0', 9600)
  while True:
    data = serial.readline().strip()
    tm = datetime.now()
    if (data and data != 'fail'):
      data = data.split(',')
      captura = {'time': tm, 'temperatura': data[0],
'humedad': data[1],
                 'presion': data[2]}
      socketio.emit('EMA data', captura, names-
pace='/test')
```

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 en el browser.

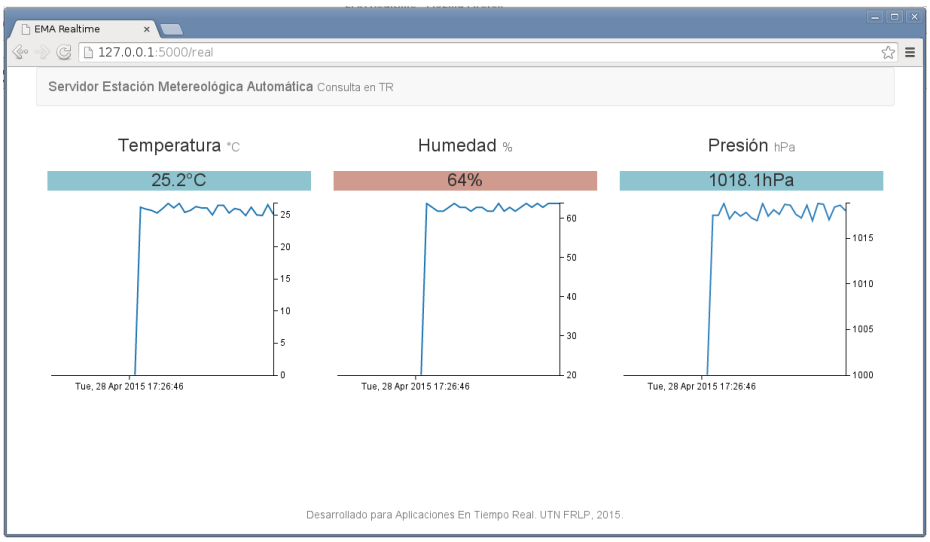

Fig. 5. Visualización en Navegador Cliente.

SocketIO [7] permite establecer una conexión bidireccional entre un servidor HTTP y un cliente utilizando TCP como transporte. Si bien está basado en el protocolo WebSocket [8] definido en la RFC 6455, agrega otras características útiles como broadcasting.
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.

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 disponibles, 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).

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. D.W. Franke ; M. Purvis, "Hardware/SoftwareCodesign: A Perspective." In Proceedings of the 13th Conference on Software Engineering, Austin-Texas(USA), Mayo de 1991.
2. Mischkalla, F.; Da He; Mueller, W., "Closing the gap between UML-based modeling, simulation and synthesis of combined HW/SW systems." Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010 , vol., no., pp.1201,1206, 8-12 Marzo de 2010.
3. https://github.com/RodrigoJacznik/arduino-flask-socketio/commit/ab3b81107802f12fe0c4fcf7fe7fbaa13716f135 .
4. Framework Flask: http://flask.pocoo.org .
5. Werkzeug: http://werkzeug.pocoo.org .
6. Jinja2: http://jinja.pocoo.org .
7. SocketIO: http://socket.io .
8. RFC6455 WebSocket: https://tools.ietf.org/html/rfc6455 .