Aplicación de servidor para una arquitectura de monitoreo inteligente y de bajo costo para procesos y sistemas mecánicos

Ramón Quiza, Teresa Pérez Sosa, Yarens J. Cruz Hernández

Centro de Estudio de Fabricación Avanzada y Sostenible (CEFAS)

Universidad de Matanzas

Autopista a Varadero km 3½, Matanzas 44740, Cuba

Teléf.: +(53)45261432, Web: http://cefas.umcc.cu, Email: ramon.quiza@umcc.cu

Resumen: El trabajo forma parte del desarrollo de una arquitectura de monitoreo para sistemas y procesos mecánicos. La misma está especialmente diseñada para pequeñas y medianas empresas, por lo que se ha implementado a base de componentes de bajo costo. No obstante, su concepción ha estado dirigida a hacerla robusta y flexible, además de incorporar técnicas de inteligencia artificial para la detección y predicción de estados. En el presente trabajo se describe el diseño, implementación y validación de la aplicación de servidor, la cual constituye el núcleo de la arquitectura. La misma realiza cuatro funciones fundamentales: recibe los datos de los módulos de captura, envía los datos a petición de los módulos de visualización, escribe y lee la información de la base de datos, y gestiona la seguridad de todo el sistema. La aplicación ha sido implementada utilizando herramientas de software libre y código abierto. Tiene, además, carácter multiplataforma, pudiendo ser ejecutada tanto sobre sistema operativo Microsoft Windows como sobre Linux. La aplicación ha sido validada mediante la interacción con los otros módulos de la arquitectura, ya existentes, para el monitoreo de sistemas, a escala de laboratorio.

Palabras claves: Arquitectura de monitoreo, Servidor, Pequeñas y medianas empresas, Software libre.

1. Introducción

El uso de sistemas de monitoreo es una de las condiciones indispensables para lograr procesos de fabricación eficientes, capaces de insertarse en la economía contemporánea, caracterizada por el dinamismo y la competitividad de los mercados (Choi *et al*. 2012). El monitoreo permite conocer el estado de un sistema, identificando las condiciones potencialmente peligrosas y permitiendo tomar las acciones correctivas pertinentes (Badonnel *et al*. 2005; Chammas *et al*. 2012).

No obstante, los altos precios de los sistemas de monitoreo los hacen económicamente prohibitivos para muchas empresas, especialmente las pequeñas y medianas (Wang *et al*. 2009). Sin embargo, esta situación se ha revertido, en gran medida, con la introducción en el mercado de productos baratos, tales como, los ordenadores de placa reducida y los sensores de bajo costo (Gutierrez-Aguado *et al*. 2016). El uso de estos componentes, permite una reducción significativa del precio del hardware de un sistema de monitoreo, pero, para ser efectivo, requiere del software correspondiente (Georges 2012).

Un sistema de monitoreo efectivo, requiere de módulos, o aplicaciones, para desempeñar sus diversas funciones, entre los que se encuentran la captura de datos, el manejo de las bases de datos, la visualización de los datos monitoreados y el control y la seguridad (Immonen, y Pakkala 2014). Como es lógico, dichos módulos deben basarse en el uso de protocolos de comunicación que garanticen una correcta trasmisión e interpretación de los datos intercambiados (Kim et al. 2016).

En el presente trabajo se muestra el diseño e implementación de un módulo de servidor, para una arquitectura de monitoreo de bajo costo. Después de esta introducción, se presenta una descripción general de la arquitectura propuesta. En la tercera sección, se definen los protocolos de comunicación y los aspectos de seguridad. La cuarta sección explica el diseño y la implementación del módulo. Finalmente, se presentan las conclusiones del trabajo y se desbroza el camino para su desarrollo futuro.

2. Descripción de la arquitectura

La arquitectura de monitoreo propuesta pretende reunir las siguientes características:

- robustez: será capaz de trabajar en las condiciones agresivas y ruidosas que caracterizan al ambiente industrial;
- flexibilidad: será capaz de adaptarse a diferentes sistemas, con un mínimo de cambios o acciones de configuración;
- asequibilidad: estará formado por componentes de hardware de bajo costo y módulos basados en software libre y de código abierto;
- cognitividad: emplea técnicas de inteligencia artificial, autoadaptativas, para modelar y predecir el comportamiento de variables de monitoreo indirecto.

La arquitectura estará formada por cuatro módulos (ver Fig. 1). En primer lugar, los módulos de captura toman las señales de los sensores, mediante el uso de interfaces, de ser necesario, y las envían al módulo central o de servidor.

El módulo de modelado se encarga de calcular, a petición del módulo central, de calcular los valores de las variables indirectas, a partir de las medidas directamente. También es capaz de modificar dichos modelos, periódicamente, a partir de nuevos datos suministrados.

Por su parte, el módulo de visualización, o interfaz hombre-máquina, tiene como objetivo mostrar el comportamiento de las variables monitoreadas a los usuarios de la arquitectura.

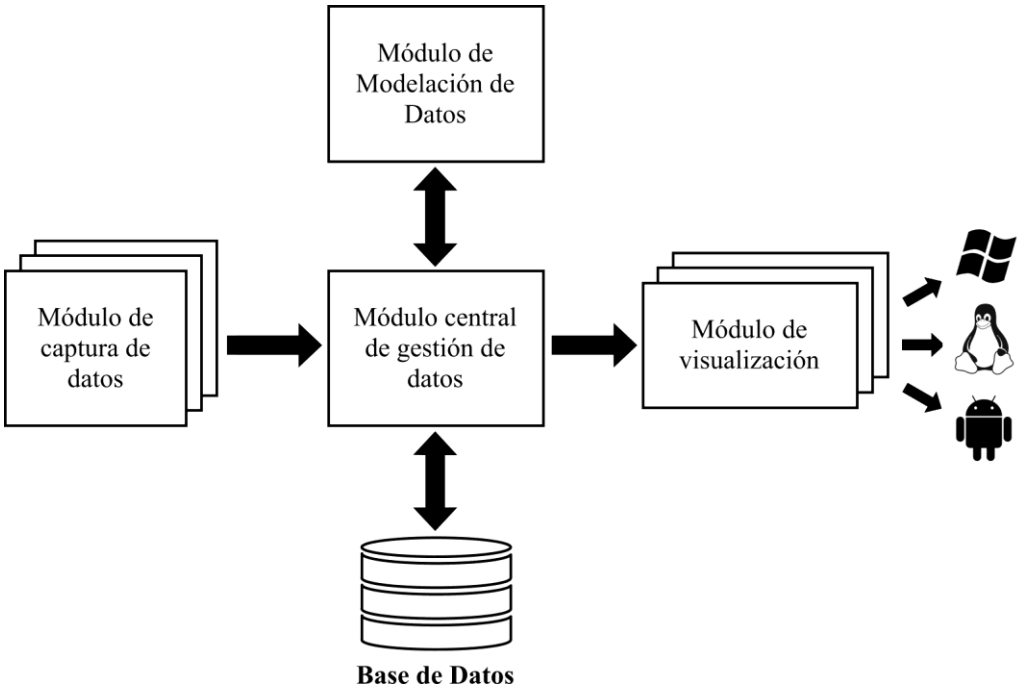

Figura 1: Esquema de la arquitectura de monitoreo propuesta (Quiza et al. 2019)

Finalmente, el módulo de servidor se encarga de interactuar con el sistema de base de datos, para escribir y leer los valores de las diferentes variables. También tiene como funciones comunicarse con el resto de los módulos y controlar las cuestiones de seguridad.

3. Protocolos de comunicación

Con el objetivo de garantizar la efectividad y seguridad de la comunicación entre los diversos módulos, se han establecido los correspondientes protocolos de comunicación.

En todos los casos, la comunicación se realiza en formato JSON, como cadena de octetos. Cada comunicación incluirá un identificador del dispositivo que la envía, así como el código Sha1 (cadena de 20 octetos), de la concatenación de su dirección IP con una palabra de seguridad (cadena de texto de 32 dígitos) (ver Fig. 2).

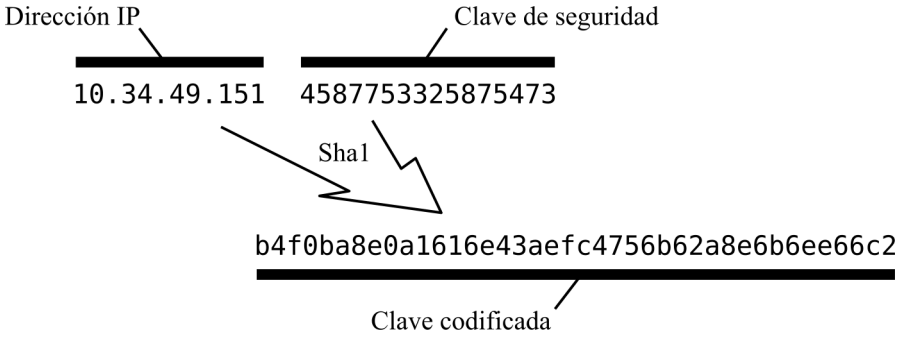

Figura 2: Codificación de la seguridad

El protocolo de comunicación entre los módulos de captura de datos y el modulo central se muestra en la Figura 3. Un aspecto importante es la marca del tiempo (*time_stamp*), que se pasa como texto con formato *yyyy*:*MM*:*dd hh*:*mm*:*zzz*, donde *yyyy* es el año, *MM* es el mes, *dd* es el día, *HH* es la hora (en formato 00…24), *mm* son los minutos, *ss* son los segundos y *zzz* son los milisegundos.

SOLICITUD

```
{
   string key;
   string time_stamp;
   array {
      {
         string id;
         double value;
         integer quality;
      }
      ...
   }
}
```

RESPUESTA

```
{
   string key;
   integer result;
   string msg;
}
```

Figura 3: Protocolo de comunicación para el envío de datos

Se incluye, además, un arreglo conteniendo las variables medidas, para cada una de las cuales se incluyen el identificador de la variable, el valor medido, y el indicador de calidad (entero en el rango 1: muy bueno a 5: muy malo),

La respuesta incluye la misma clave de seguridad pasada en la petición, un entero codificando el resultado y una cadena de texto con un mensaje explicando el resultado obtenido en la operación.

SOLICITUD

```
{
   string key;
   string id;
   array dataset {
      array {
         double x_11;
         ...
         double x_1n;
      }
      ...
      array {
         double x_m1;
         ...
         double x_mn;
      }
   }
}
```

RESPUESTA

```
{
   string key;
   integer result;
   string msg;
   array dataset {
      double y1;
      ...
      double ym;
   }
}
```

Figura 4: Protocolo de comunicación de la modelación

La relación entre el módulo central y el de modelación, se realiza en dos modalidades. La primera de ellas, es la modelación, cuyo protocolo se muestra en la Figura 4.

En la solicitud de dicho protocolo se envía un arreglo donde se incluyen las *m n*-tuplas de valores de las variables independientes. Por su parte, en la respuesta, se incluye, además del resultado, un arreglo de valores numéricos con los valores evaluados de la variable dependiente.

SOLICITUD

```
{
   string key;
   string id;
   array dataset {
      array {
         double x_11;
         ...
         double x_1n;
         double y_1;
      }
      ...
      array {
         double x_m1;
         ...
         double x_mn;
         double y_m;
      }
   }
   array parameters {
      object parameter {
         string name1;
         double value;
      }
   }
}
```

RESPUESTA

```
{
   string key;
   integer result;
   string msg;
}
```

Figura 5: Protocolo de comunicación del ajuste del modelo

El protocolo de ajuste de modelo (Fig. 5), además de los valores de las variables independientes, los valores de la variable dependiente con la que se ajusta el modelo. La respuesta, incluye solamente, el resultado de la operación.

Por último, el protocolo utilizado para comunicarse entre los datos y el visualizador (Fig. 6), La solicitud incluye el intervalo de tiempo a considerar, dado por los valores numéricos, *from_time* and *to_time*, así como un arreglo de valores numéricos, dado por los identificadores de las variables a devolver. La respuesta incluye, además del resultado, las *m n*-tuplas, con los valores de las variables en el intervalo considerado.

SOLICITUD

```
{
   string key;
   string from_time;
   string to_time;
   array ids {
      string id1;
      ...
      string idn;
   }
}
```

RESPUESTA

```
{
   string key;
   integer result;
   string msg;
   array dataset {
      {
         double x11;
         ...
         double x1n;
      }
      ...
      {
         double xm1;
         ...
         double xmn;
      }
   }
}
```

Figura 6: Protocolo de comunicación para la visualización

4. Diseño e implementación de la aplicación

La aplicación de diseño con una arquitectura orientada a servicios. En la misma, se implementó un servidor de TCP/IP, que escucha y responde las peticiones de los módulos de captura y visualización. La comunicación con el módulo de modelación, se produce, asincrónicamente, cuando arriban datos de variables independientes.

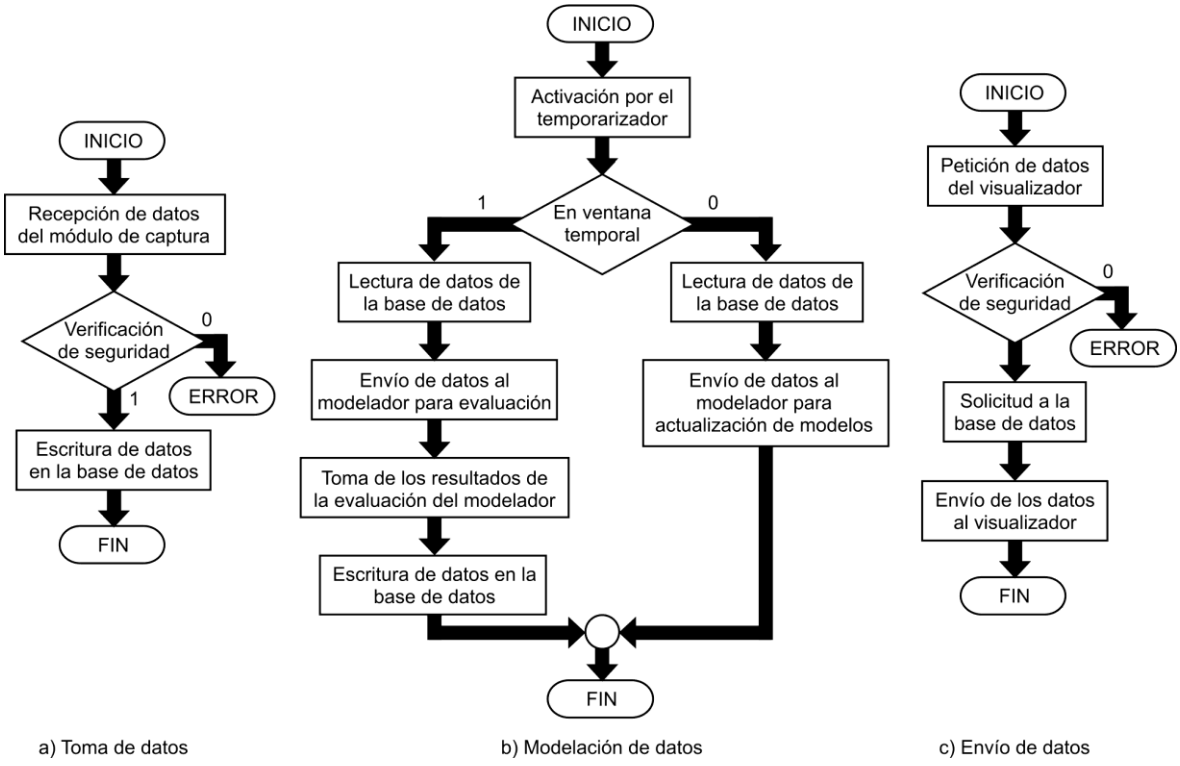

Figura 7: Procesos del módulo central

El servidor incluye tres procesos (Fig. 7). Los procesos de toma y de envío de datos se ejecutan a petición de los respectivos clientes. Ambos realizan la verificación de seguridad, en base al código de seguridad y del IP del cliente. En caso de ser positiva, realizan las acciones respectivas de escritura o lectura de la base de datos.

Por su parte, el proceso de modelación se realiza, a intervalos regulares, marcados por un temporizador. En dependencia de que el instante esté o no en la ventana temporal considerada, se ejecuta o bien la modelación o bien el reajuste del modelo.

Como gestor de base de datos se utiliza MySQL Server, que también se comunica mediante protocolo TCP/IP con el módulo del servidor.

5. Conclusiones

El módulo central diseñado para la arquitectura de monitoreo, reúne los requisitos para ejecutar las funciones previstas en el mismo. El uso de protocolos basados en JSON, permiten el intercambio rápido y efectivo de información con los otros módulos. El sistema está totalmente basado en herramientas de software libre, lo cual garantiza su distribución sin trabas de licencias.

Como paso siguiente del trabajo se impone su implementación para un caso de estudio concreto, con vistas a su validación en condiciones reales.

Agradecimientos

El presente trabajo ha sido llevado a cabo gracias al financiamiento del proyecto “ROFLEXIN/LC: Sistema robusto, flexible e inteligente, de bajo costo, para monitoreo de sistemas y procesos mecánicos”, asociado al Programa de Automatización de Procesos Tecnológicos.

Referencias

Badonnel, R., State, R., & Festor, O. (2005). Self-organized monitoring in ad-hoc networks. *Telecommunication Systems*, 30(1), 143-160. doi: 10.1007/s11235-005-4322-3

Chammas, A., Traore, M., Duviella, E., Sayed-Mouchaweh, M., & Lecoeuche, S. (2012). Condition monitoring architecture for maintenance of dynamical systems with unknown failure modes. *IFAC Proceedings Volumes*, 45(31), 36-41. doi: 10.3182/20121122-2-ES-4026.00042

Choi, S., Hwang, H., Song, B., & Cha, H. (2012). Hardware-assisted energy monitoring architecture for micro sensor nodes. *Journal of Systems Architecture*, 58(2), 73-85. doi: 10.1016/j.sysarc.2011.12.001

Georges, D. (2012). Optimal design of a PMU-based monitoring architecture for power systems. *IFAC Proceedings Volumes*, 45(21), 478-483. doi: 10.3182/20120902-4-FR-2032.00084

Gutierrez-Aguado, J., Alcaraz Calero, J. M., & Diaz Villanueva, W. (2016). IaaSMon: Monitoring architecture for public cloud computing data centers. *Journal of Grid Computing*, 14(2), 283-297. doi: 10.1007/s10723-015-9357-4

Immonen, A., & Pakkala, D. (2014). A survey of methods and approaches for reliable dynamic service compositions. *Service Oriented Computing and Applications*, 8(2), 129-158. doi: 10.1007/s11761-013-0153-3

Kim, H., Yoon, S., Jeon, H., Lee, W., & Kang, S. (2016). Service platform and monitoring architecture for network function virtualization (NFV). *Cluster Computing*, 19(4), 1835-1841. doi: 10.1007/s10586-016-0640-3

Wang, Y. M., Yin, H. L., Xiao, N. F., & Jiang, Y. R. (2009). Internet-based remote manipulation and monitoring of an industry robot in advanced manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, 43(9), 907-913. doi: 10.1007/s00170-008-1768-y