Actas de las XXXVI Jornadas de Automática, 2 - 4 de septiembre de 2015. Bilbao
ISBN 978-84-15914-12-9 © 2015 Comité Español de Automática de la IFAC (CEA-IFAC)

Hacia un *framework* basado en ROS para la implementación de Sistemas Ciberfísicos

Pablo González-Nalda[1] Isidro Calvo[2] Ismael Etxeberria-Agiriano[1] Ekaitz Zulueta[2] José Manuel López-Guede[2]
{pablo.gonzalez,isidro.calvo,ismael.etxeberria,ekaitz.zulueta,jm.lopez}@ehu.eus
[1] Lenguajes y Sistemas Informáticos
[2] Ingeniería de Sistemas y Automática
Escuela Universitaria de Ingeniería de Vitoria-Gasteiz, Universidad del País Vasco, UPV/EHU

Resumen

Este artículo describe un framework para su uso en Sistemas Ciberfísicos (CPS), con el objetivo de facilitar la implementación de un sistema distribuido de adquisición de datos de sensores y otras mediciones, su computación y envío de órdenes a actuadores.

Esta arquitectura se puede implementar en diferentes dominios de aplicación. Cada uno de los elementos de esta arquitectura distribuida es un nodo del middleware libre ROS (Robotic Operating System). De esta forma se obtiene modularidad y evolución incremental, redundancia y flexibilidad. Se utilizan nodos de bajo coste, que cuentan con una gran comunidad de desarrolladores y abundante información por ser hardware y software libres. Finalmente se detalla un caso de aplicación consistente en un robot móvil que resalta algunas de las posibilidades de la arquitectura.

Palabras clave: Arquitectura, Sistemas Ciberfísicos (CPS), ROS middleware, Teleoperación; Robótica móvil; Hw y Sw libre; Raspberry Pi; Arduino.

1. INTRODUCCIÓN

Este artículo plantea un *framework* para su uso en Sistemas Ciberfísicos (CPS), con el objetivo de facilitar la implementación de un sistema distribuido de adquisición de datos de sensores y otras mediciones, su computación y envío de instrucciones a actuadores.

Esta arquitectura se puede implementar en diferentes entornos, por ejemplo para el control de un edificio industrial, de una vivienda o de un objeto móvil (vehículo, maquinaria, robótica autónoma). Cada uno de los elementos de esta arquitectura distribuida (o clúster) es un nodo del middleware ROS (Robotic Operating System). Las comunicaciones son por TCP/IP y se pueden hacer seguras implementando un servidor VPN.

Las ventajas de esta red de nodos ROS son varias: la modularidad de la arquitectura permite adaptarla a sistemas ya implantados para mejorar su evolución de forma incremental, la redundancia es posible por el esquema publicador/suscriptor de ROS, es libre (tiene licencia BSD) y multiplataforma.

Tras la definición de dicha arquitectura se explica la elección de componentes para cada tipo de nodo, en función de las tareas asignadas y buscando el coste mínimo y la máxima flexibilidad y modularidad. Los dispositivos hardware elegidos, de bajo coste, se encuentran entre los más utilizados en sus respectivas categorías, y por tanto una gran comunidad de desarrolladores que los utilizan. Se pueden tener muchos dispositivos replicados para desarrollar en paralelo cada subsistema.

En los dispositivos elegidos el hardware y el software son libres. Por ello es muy sencillo acceder a foros en los que encontrar abundante información relativa a los mismos, de forma que se puedan enriquecer los aspectos técnicos de la arquitectura. También se pueden establecer puentes para controlar hardware externo o no libre (a través de EtherCAT) u otro tipo de redes (OpenHab).

El resto del artículo se organiza de la siguiente manera: en la Sección 2 se hace una breve revisión de algunos trabajos relacionados con este trabajo; en la Sección 3 se proponen las ideas generales de la arquitectura presentada; en la Sección 4 se eligen los componentes que cumplen los requisitos; en la Sección 5 se demuestra la aplicabilidad del modelo mediante una implementación con un coche de radiocontrol y en la Sección 6 se presentan las conclusiones.

2. DOMINIOS DE APLICACIÓN

En esta sección se presentan los sistemas ciberfísicos y trabajos que tienen relación con el presentado.

2.1. SISTEMAS CIBERFÍSICOS

Los sistemas ciber-físicos, o CPS (Cyber-Physical Systems), se corresponden con una disciplina considerada fundamental en investigación en los últimos diez años [1, 2]. Son sistemas computacionales que controlan entidades físicas cuyo diseño requiere la aplicación transversal de conocimientos avanzados en diversas disciplinas tal y como se puede ver en [3]. Estas disciplinas se pueden

ISBN 978-84-15914-12-9 © 2015 Comité Español de Automática de la IFAC (CEA-IFAC)

agrupar en las siguientes áreas de conocimiento: (1) redes de comunicación: (2) computación embebida y sistemas de tiempo real; y (3) sistemas de control (ver Fig. 1) [4, 5]. El término CPS está frecuentemente relacionado con entornos de redes ubicuas de sensores para conectar elementos físicos a computadores y robótica autónoma [3, 6, 7]. Los elementos que componen este tipo de sistemas son cada vez más comunes en la vida cotidiana por lo que hace tiempo que han dejado de sorprendernos. Además, la popularización de los teléfonos inteligentes (o *smartphones*) ha facilitado el desarrollo y generalización de este tipo de aplicaciones, ya que este tipo de dispositivos proporciona una interfaz de usuario para las aplicaciones CPS.

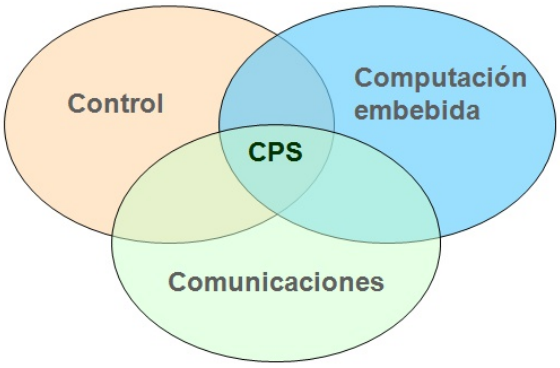

Figura 1: Áreas de trabajo principales de los CPS

Con respecto a los dominios de aplicación, los CPS abarcan extensas áreas como la domótica, agricultura, infraestructuras civiles, asistencia sanitaria, automoción, transporte, energía, automatización industrial, ingeniería aeroespacial, entretenimiento y electrónica de consumo. Crear una arquitectura estándar, libre, de bajo coste, de fácil manejo y con gran cantidad de documentación simplifica la implantación de estos sistemas.

Por otro lado, hay que tener en cuenta que hoy en día se dispone de potentes plataformas baratas y de bajo consumo para un uso cotidiano, como son las populares placas Raspberry Pi o las Arduino que permiten el prototipado e incluso construcción de sistemas finales de manera modular [10, 11]. La disponibilidad de gran cantidad de información técnica en Internet ha facilitado la popularidad de estos dispositivos.

2.2. ÁMBITO DE APLICACIÓN

Los CPS son unos sistemas muy adecuados para utilizarlos en el desarrollo de sistemas versátiles en la automatización industrial, infraestructuras, la robótica, la mecatrónica y los sistemas embebidos. Para esta tarea es ventajoso disponer de esquemas de bajo coste, estándar y libres.

El software ROS (Sistema Operativo Robótico) [16] es en realidad un *middleware* creado para simplificar las comunicaciones y facilitar la escritura de módulos en el control de robots. En 2012 se constituyó una iniciativa de empresas denominada ROS-Industrial [20] que tiene como objetivo impulsar el desarrollo de módulos de ROS para la automatización industrial.

Algunos autores como Valera, Soriano y Vallés [12] realizan un análisis de las diferentes soluciones en el ámbito educativo de la automatización, control por computador, robótica y sistemas mecatrónicos. A pesar de no ser soluciones pensadas para la automatización industrial, en la que la robustez y fiabilidad son imprescindibles, la redundancia de una red que incorpora gran número de dispositivos asequibles puede compensar dicho déficit.

Entre otras plataformas, en [12] se analizan la plataforma LEGO Mindstorms y sus entornos de programación. Posteriormente, los autores describen en detalle tres de las tarjetas de control embebido (también denominadas ordenadores de placa única, Single-Board Computers o SBC) más habituales: Arduino Uno, Raspberry Pi y BeagleBone Black. Estos SBC son hardware libre, ejecutan software libre [19], tienen mucha documentación y una amplia comunidad de usuarios.

Estos SBC también resultan muy interesantes en el ámbito de la investigación en robótica móvil ya que facilita construir vehículos móviles de bajo coste. Oros y Krichmar [13, 14] plantean la utilización de teléfonos móviles para educación, investigación y entretenimiento. Estos autores proponen la "robótica basada en móviles" como un sistema barato y flexible. Basan el sistema en un móvil Android como unidad de cómputo junto a sensores (cámara, GPS y otros) y conectividad (WiFi y 3G), y una placa IOIO como interfaz ante los servos (comunicación por modulación de anchura de pulsos o PWM) y varios tipos de sensores. Sus conclusiones en cuanto a la mecánica son que un vehículo de radiocontrol es muy adecuado por ser muy rápido, tiene una buena amortiguación para terrenos irregulares y son resistentes.

En el área de los sistemas de control valoran muy positivamente que los móviles Android tienen generalmente gran potencia y baterías, gran conectividad, cámara incorporada, multitud de sensores y un buen sistema de desarrollo de software. También comparan el coste de su sistema robótico frente a otros. Plantean que su estructura es modular y que se pueden añadir más sensores y diferentes medios de locomoción de forma más flexible. En su comparación afirman que su sistema es de menor coste que otros para la misma potencia computacional. Por ello lo ven beneficioso para el desarrollo de enjambres de robots (swarm robotics) y analizan la implementación de librerías que requieren

grandes necesidades de recursos como OpenCV [15] o ROS [16], además de técnicas de control como las Redes Neuronales.

3. ARQUITECTURA PROPUESTA

En este trabajo se presenta un software marco o *framework*, altamente modular y distribuido con el objetivo de usarse en todo tipo de sistemas ciberfísicos que requieran adquirir gran variedad de mediciones, la ejecución de algoritmos de procesamiento, seguimiento y decisión, y el envío de instrucciones a diferentes maquinarias, robots y cualquier tipo de actuadores. Esta arquitectura se puede implementar para llevar a cabo el control de un edificio *inteligente* (como iluminación, aire acondicionado, control de presencia y vigilancia), de un edificio industrial, de una planta industrial o un almacén, de una vivienda o de un objeto móvil (vehículo, maquinaria, robótica autónoma).

Para mostrar sus posibilidades vamos a definir una arquitectura para un problema concreto de robótica móvil. Es de especial interés que el hardware utilizado sea de bajo consumo energético y ligero en peso.

La Fig. 2 muestra la visión general de la arquitectura para este problema. Se pueden observar los módulos de computación y decisión (se describe en la subsección 4.1), de interacción con componentes mecatrónicos (subsección 4.2), de sistemas de audio y vídeo (subsección 4.3) y módulo de visualización y control (subsección 4.4).

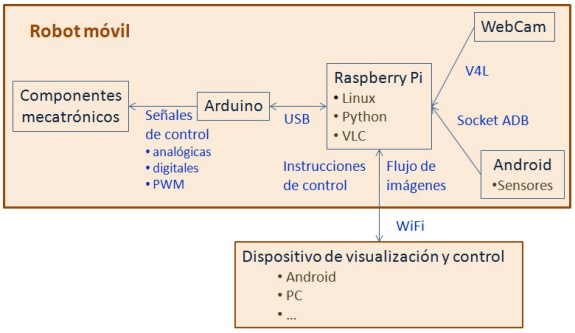

Figura 2: Esquema de la arquitectura propuesta.

3.1. ROS

ROS [16] es un *middleware* orientado a la construcción del control de robots. Se instala sobre un sistema operativo GNU/Linux y permite incorporar todo tipo de dispositivos a una red modular y distribuida que en su conjunto lleva el control del sistema. Un nodo es el nodo maestro que permite al resto de los nodos que se encuentren para comunicarse por TCP/IP. En la Fig. 3 se muestran los diferentes medios para comunicar los nodos: esquema de comunicación por mensajes o *tópicos* (publicador-suscriptor) y servicio-cliente.

ROS presenta muchas ventajas:

- No hace falta reinventar la rueda: es un software optimizado para las comunicaciones entre elementos de una red.
- ROS es Software Libre (tiene licencia BSD [19]) y multiplataforma.
- ROS es un estándar en Robótica: 18 de los 23 robots de las finales en la competición DARPA Robotics Challenge usan ROS [18].
- Hay un entorno de desarrollo para crear aplicaciones Android que se integren en sistemas ROS [23].
- Se integran multitud de sistemas y sensores [24].
- Se puede usar cualquier tipo de hardware de comunicaciones por TCP/IP entre los nodos del middleware ROS, por ejemplo por Ethernet o WiFi.
- Se pueden simular robots en Gazebo [26].
- Se pueden usar protocolos como EtherCAT [30].
- Puede integrarse con plataformas utilizadas en automatización y robótica como OROCOS [21, 22].
- Se pueden integrar componentes de diferentes marcas siempre que se pueda crear un *puente* o sotfware que traduzca el protocolo utilizado a ROS.
- Hay software-puente con el servidor de dómotica OpenHab [28] mediante el puente [29] y todo tipo de *drivers* [25].
- Todo el código tiene licencias libres y está disponible en GitHub [27].

Figura 3: Esquema de ROS [16].

En resumen, ROS es la elección correcta por sus posibilidades, es el mejor documentado, es muy usado y tiene una gran comunidad de desarrollo y de comunidad [17].

ISBN 978-84-15914-12-9 © 2015 Comité Español de Automática de la IFAC (CEA-IFAC)

Una estructura modular proporciona evidentes ventajas:

- El crecimiento de la red es flexible, dado que la estructura publicador/suscriptor permite separar problemas de una manera P2P. No obstante, existe la posibilidad de crear también estructuras cliente/servidor con el sistema *Services*.
- La migración de un sistema implantado a otro se puede producir de forma gradual, incremental.
- Se obtiene redundancia fácilmente ya que se pueden integrar más elementos que dupliquen la estructura.

Las comunicaciones con el nodo maestro de ROS se pueden hacer seguras implementando un servidor VPN en uno de los nodos de computación al que se pueden conectar el resto de forma encriptada.

En definitiva, disponemos de una red de nodos que se conectan directamente para ofrecerse mensajes y servicios en la que el maestro actúa para resolver nombres. Los nodos comparten el estándar de comunicaciones pero su hardware es heterogéneo, como lo son las funciones que deben cumplir.

4. ELECCIÓN DE COMPONENTES

Una vez definida esta arquitectura genérica basada en ROS y detalladas las ventajas que proporciona para cualquier red de adquisición de datos y control de actuadores, pasamos a discutir la elección de unos componentes que mantengan las ventajas y faciliten la implementación de una red de este tipo buscando el coste mínimo y la máxima flexibilidad y modularidad.

4.1. MÓDULO DE COMPUTACIÓN

De entre los posibles, la opción más interesante es la basada en el SBC Raspberry Pi (abreviaremos con RPi). Concretamente, la recientemente lanzada Raspberry Pi 2, proporciona una gran potencia de cálculo y grandes posibilidades de conectividad por un precio muy asequible (en torno a los 50€).

La RPi opera con muchos sistemas operativos y lenguajes. Una opción muy habitual es usar un SO GNU/Linux como Debian o Ubuntu, con las múltiples ventajas que conlleva usar un sistema operativo tan extendido en superordenadores y servidores en Internet y en empresas. Es típica la utilización para la programación de C o Python. Desarrollar en estos sistemas con estos lenguajes (especialmente C) permite acercarse al hardware y facilitar un mejor tiempo de respuesta frente al entorno Java del dúo Android/IOIO planteado en [14]. Además se puede optimizar fácilmente la plataforma configurando adecuadamente para mejorar la eficiencia y rendimiento del sistema operativo, lo cual permite ahorrar en cálculo y por lo tanto en baterías. Este enfoque permite resolver uno de los problemas del ecosistema Android, ya que es un sistema muy pesado computacionalmente por ofrecer una gran variedad de servicios, es difícil de configurar y por ello es muy complejo.

Tenemos la posibilidad de instalar software de forma automática mediante las utilidades de gestión de software de Debian/Ubuntu en la RPi. Podemos también compilar todo tipo de software compatible con estas distribuciones, además de desarrollar en la propia tarjeta nuestro software. La flexibilidad de la RPi por su conectividad (varios USBs, Ethernet, GPIO, I2C, SPI, UART) nos permite que además de modular, sea fácilmente ampliable y permita añadir nuevos modos de comunicación. Si se necesitara menos potencia de cálculo para menos consumo, existen alternativas dentro de la familia Raspberry Pi.

4.2. MÓDULO DE INTERACCIÓN CON MECATRÓNICA

Con respecto al módulo de interacción con los componentes mecatrónicos, se optó por introducir en la arquitectura la placa Arduino Uno por ser un estándar y ser hardware libre: sus especificaciones son abiertas y se pueden crear nuevos dispositivos a partir de ellas o modificarlas. Nuestra elección es de muy bajo coste, en torno a los 20€ frente a IOIO que cuesta casi el doble, porque Arduino Uno es más popular y porque podemos prescindir de costosas funciones añadidas que tiene IOIO (programación con Java).

La tarjeta Arduino Uno proporciona prácticamente la misma funcionalidad (salvo la interfaz con sistemas Java) por lo que se conecta a sensores y motores de la misma forma. Otra ventaja es que se ha convertido en un estándar de facto con más documentación, vigencia y un gran número de sensores y actuadores comerciales disponibles. La comunicación entre la RPi y la Arduino Uno se realiza a través de sus respectivas librerías de comunicación Serie sobre USB y por supuesto y preferentemente con ROS.

Se podría simplificar el sistema reduciéndolo a una sola placa usando el miniordenador BeagleBone Black (BBB), que dispone de más flexibilidad en E/S hardware aunque tenga menos capacidad con señales analógicas que una Arduino. Con un BBB no renunciamos a la potencia del GNU/Linux puesto que también usa estas distribuciones. Sin embargo, estimamos muy potente mantener la dualidad de usar un SBC RPi para procesamiento y una o varias tarjetas Arduino por cada RPi para la interacción con los componentes mecatrónicos puesto que están diseñadas para controlar todo tipo de señales analógicas y digitales.

ISBN 978-84-15914-12-9 © 2015 Comité Español de Automática de la IFAC (CEA-IFAC)

4.3. MÓDULO DE SISTEMAS DE AUDIO Y VÍDEO

Si se necesitan una o más cámaras se puede ampliar el sistema de una forma sencilla, ya que el GNU/Linux instala automáticamente las cámaras web compatibles por USB con el estándar V4L2 (vídeo para Linux V2). Es el método habitual para que el software capture imagen, por ejemplo OpenCV. Existen puentes para comunicar OpenCV con ROS. Lo mismo ocurre con el sonido, que es fácil de capturar en ROS e incluso se puede sintetizar el habla.

Todas estas placas permiten añadir módulos hardware (llamados *shield* en Arduino o *cape* en BeagleBone) que pueden controlar otros tipos de señales. Por ejemplo, se puede incluir otro hardware como ZigBee en las Arduino a través de un *shield*.

4.4. MÓDULO DE SISTEMAS DE VISUALIZACIÓN Y CONTROL REMOTO

Este diseño de arquitectura no excluye el uso de un dispositivo Android en el sistema, pero un diseño centrado en un ordenador mono-placa basado en GNU/Linux evita la dificultad del desarrollo de aplicaciones en Android y su sistema de gestión de ejecución, orientado a la interactividad. Se pueden aprovechar las capacidades gráficas de los dispositivos Android para utilizarlos como interfaz de usuario del sistema, bien para monitorizar el sistema o para proporcionar información a las RPi. El módulo `rosjava` permite desarrollar aplicaciones Java dentro de ROS. De esta forma ya existen aplicaciones Android que visualizan datos de ROS.

Hay muchas aplicaciones de software libre de ejemplo para Android que permiten enviar con ROS los datos que recolectan (sensores, cámara) para que puedan ser integrados y procesados en el dispositivo principal. De hecho, no es necesario usar un móvil de alta gama, incluso puede utilizarse un móvil con taras (p.e. la pantalla parcialmente rota), por lo que se podría adaptar para pesar menos.

Otro posible uso de un dispositivo Android o cualquier otro sistema fijo o móvil es el control remoto o la supervisión con los mecanismos estándares de ROS:

- RVIZ: visualización 3D
- BAGS: registro y visualización de datos
- PLOT: dibujo de datos
- GRAPH: grafo de los procesos y conexiones

El cliente puede ser de cualquier tipo, incluyendo una interfaz web. Como se ha indicado anteriormente, hay un compromiso entre funcionalidad y gasto de batería, en este caso de transmisión Bluetooth o WiFi al cliente.

La estructura distribuida permite integrar cualquier otro dispositivo conectado por una gran variedad de hardware, por lo que la evolución de un sistema ya en uso está asegurada.

En resumen, los dispositivos hardware elegidos, de bajo coste, se encuentran entre los más utilizados en sus respectivas categorías, y disfrutan por tanto de una gran comunidad de desarrolladores que los utilizan. Se pueden tener muchos dispositivos replicados para desarrollar en paralelo cada subsistema. Además, en los dispositivos elegidos el hardware y el software son libres, y por ello es muy sencillo acceder a foros en los que encontrar abundante información relativa a los mismos, de forma que se puedan enriquecer los aspectos técnicos de la arquitectura.

5. UN CASO DE APLICACIÓN

Como un ejemplo de la arquitectura descrita, se implementa un caso de aplicación consistente en una instanciación de la arquitectura y una implementación de la misma. Para validar la arquitectura propuesta se ha implementado sobre un coche de radiocontrol LRP S10 Twister 2WD Short Course. Este vehículo puede alcanzar gran velocidad por lo que se necesitan buenos tiempos de reacción para evitar obstáculos. La autonomía de la batería NiMH de 7,2V y 1600mAh ronda la media hora (ver Fig. 4).

Figura 4: Vehículo radiocontrolado (LRP.cc).

El esquema de la Fig. 2 se particulariza con los diferentes componentes en la Fig. 5.

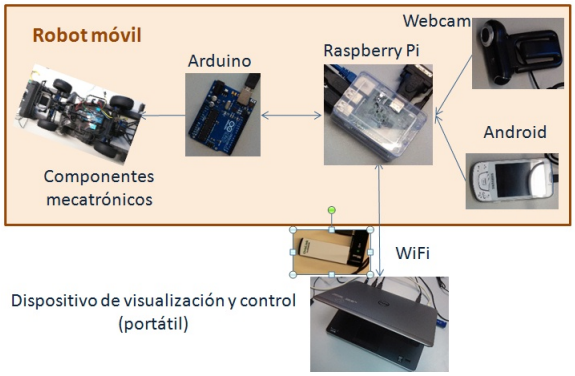

Figura 5: Aplicación práctica del modelo.

ISBN 978-84-15914-12-9 © 2015 Comité Español de Automática de la IFAC (CEA-IFAC)

La alimentación de la RPi se realiza a través de microUSB. En el prototipo propuesto se conecta a una batería USB estándar de carga de móviles. La RPi2 consume 900mA según carga de proceso y periféricos USB, mientras que la batería usada puede suministrar hasta 2A y tiene una capacidad de 6800mAh con lo que se estima una autonomía de siete horas. La configuración de los diferentes subsistemas se describe a continuación.

A. Conexión RPi-Arduino

Esta conexión une la RPi con una Arduino Uno. Mediante ROS sobre USB se envían a la Arduino las órdenes de activación de los motores y se pueden recoger las lecturas de los sensores conectados.

La Arduino modula el servo de dirección y el motor de tracción mediante PWM mediante un circuito ESC (Electronic Speed Control) que controla el motor de corriente continua. El ESC está preparado para adaptarse al receptor de control remoto con un método denominado AI (inteligente), que asociará valores entre 0 y 180 como activación nula del motor. A partir de ese valor acelera o frena, y si se vuelve al cero y se le envía un valor de frenado se mueve hacia atrás.

B. Webcam

Se le ha conectado una cámara web Logitech QuickCam Pro 9000 al módulo central mediante USB. La imagen se recoge mediante el controlador V4L2 del núcleo Linux. El retardo experimentado podrá reducirse con la RPi2, permitiendo más trabajo en el procesamiento de la señal con librerías como OpenCV.

C. Control Remoto

Para el control remoto del vehículo se conecta un módulo WiFi USB a la RPi. Para ello se precisa de un hub USB alimentado. Se ha definido una red local a la que se conectan la RPi y el cliente como en [8, 9], controlando desde un Android. Además se puede ver la imagen de la cámara web.

D. Lectura de sensores de Android

Otra línea de evaluación de la arquitectura propuesta conecta al sistema un antiguo móvil Android para suministrar las lecturas de sensores utilizando una aplicación libre llamada Sensor Readout: giróscopo, acelerómetro lineal y angular, campos magnéticos y otros valores de interés según pueda integrarlos el programa principal. La comunicación se lleva a cabo a través de `rosjava`.

E. División modular

Como ejemplo de modularidad se propone dividir el proyecto en dos grupos de trabajo:

- Grupo principal: une una RPiB+ con una Arduino Uno.
- Grupo auxiliar: combina una RPiA, cámara, WiFi-USB y un móvil Android Galaxy i7500 con una ROM GAOSP Froyo.

Otras posibilidades planteadas son utilizar dispositivos móviles o portátiles con Ubuntu. Esta división muestra otra ventaja de este modelo: la facilidad de crear desarrollos en paralelo; el bajo coste del hardware permite disponer de varias unidades, además de diferentes versiones. Como consecuencia el sistema permite comprobar la robustez intercambiando las diferentes unidades para ajustar capacidades, valorar debilidades y minimizar el consumo.

F. Prototipo disponible

El montaje completo del prototipo del sistema descrito en esta sección se muestra en la Fig. 6. Un elemento a mejorar es la voracidad en el consumo de batería, probablemente debido a la cámara Web y el dispositivo WiFi.

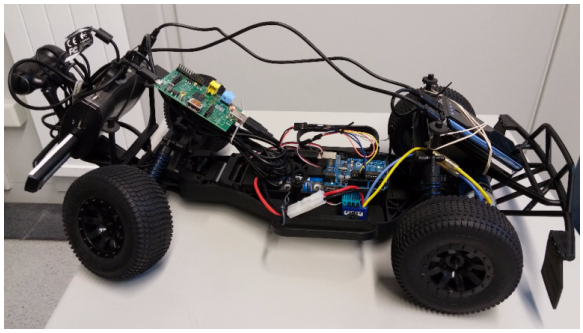

Figura 6: Robot móvil resultante con sus dispositivos.

6. CONCLUSIONES

Se ha presentado en este artículo un *sistema marco* o *framework* para estructurar y mejorar la implementación de un sistema distribuido de adquisición de datos de sensores, su computación, toma de decisiones y envío de instrucciones a actuadores.

Esta arquitectura se puede implementar en diferentes entornos y es especialmente adecuada para su uso en Sistemas Ciberfísicos (CPS).

Cada uno de los elementos de esta arquitectura distribuida es un nodo de ROS (Robotic Operating System). Las ventajas de esta red de nodos ROS son varias: es un software muy extendido y bien documentado, potente y extendido a gran variedad de hardware, la modularidad de la arquitectura permite adaptarla a sistemas ya implantados para mejorar su evolución de forma incremental, la redundancia

es posible, es software libre y multiplataforma. Las comunicaciones son por TCP/IP y se pueden hacer seguras implementando un servidor VPN.

Se han elegido componentes para los diferentes nodos. Los dispositivos hardware elegidos, de bajo coste, son los más utilizados por ser hardware y software libre, y con una gran comunidad de desarrolladores. Por ello se encuentra abundante información relativa a los mismos. También se pueden establecer puentes para controlar hardware externo.

Agradecimientos

Este trabajo ha sido subvencionado por el Ministerio de Economía y Competitividad (MINECO) bajo el proyecto DPI2012-37806-C02-01, por el Gobierno Vasco/Eusko Jaurlaritza bajo el proyecto CPS4PSS ETORTEK14/10 y por la Universidad del País Vasco/Euskal Herriko Unibertsitatea (UPV/EHU) bajo el proyecto EHU13/42 y la UFI11/28.

Referencias

[1] W. G. Gilroy, "NSF funds Cyber-Physical Systems Project," http://newsinfo.nd.edu/news/17248-nsf-funds-cyber-phys/

[2] National Science Foundation Program Solicitation, http://www.nsf.gov/pubs/2008/nsf08611/nsf08611.pdf.

[3] Cyber-Physical Systems, http://cyberphysicalsystems.org/

[4] I. Etxeberria-Agiriano, I. Calvo, A. Noguero and E. Zulueta, "Towards middleware-based cooperation topologies for the next generation of CPS" International Journal of Online Engineering, 8 (Special Issue 2), pp. 20-27, 2012.

[5] I. Calvo, I. Etxeberria-Agiriano and A. Noguero, "Distribution Middleware Technologies for Cyber Physical Systems", Proc. of the Remote Engineering & Virtual Instrumentation (REV-2012), pp. 298-301, July 2012.

[6] W. Wolf, "Cyber-physical Systems", Computer, vol. 42, no. 3, pp. 88-89, March 2009. http://dx.doi.org/10.1109/MC.2009.81

[7] R. Rajkumar, I. Lee, L. Sha, J. Stankovic, "Cyber-physical systems: The next computing revolution", Design Automation Conference (DAC), 2010 47th ACM/IEEE , vol., no., pp.731-736, 13-18 June 2010

[8] P. González-Nalda, I. Calvo, I. Etxeberria-Agiriano, A. García-Ruíz, S. Martínez-Lesta, D. Caballero-Martín, "Building a CPS as an Educational Challenge" International Journal of Online Engineering, 10 (4), pp. 52-58, 2014.

[9] P. González-Nalda, I. Calvo, I. Etxeberria-Agiriano, A. García-Ruiz, S. Martínez-Lesta, D. Caballero-Martín, "The Challenge of Building a Cyber Physical System as an Educational Experience" 9ª Conferencia Ibérica de Sistemas y Tecnologías de la Información (CISTI), pp. 327-332, 18-21 Junio 2014

[10] E. A. Lee and S. A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach, http://LeeSeshia.org, ISBN978-0-557-70857-4,2011

[11] P. Marwedel, "Embedded and cyber-physical systems in a nutshell", DAC.COM Knowledge Center Article, 2010.

[12] A. Valera, A. Soriano, M. Vallés. "Plataformas de Bajo Coste para la Realización de Trabajos Prácticos de Mecatrónica y Robótica," Revista Iberoamericana de Automática e Informática industrial 11 (2014) 363-376

[13] Oros N., Krichmar J.L. 2013. "Smartphone Based Robotics: Powerful, Flexible and Inexpensive Robots for Hobbyists, Educators, Students and Researchers." CECS Technical Report 13-16. November 26, 2013. Center for Embedded Computer Systems, University of California, Irvine.

[14] Oros N., Krichmar J.L. 2013. "Android Based Robotics: Powerful, Flexible and Inexpensive Robots for Hobbyists, Educators, Students and Researchers." Cognitive Anteater Robotics Laboratory. University of California, Irvine, 2013. Web. November 2013. http://www.socsci.uci.edu/~jkrichma/ABR/index.html

[15] OpenCV. Bliblioteca libre de Visión Artifical. http://opencv.org

[16] ROS. Robot Operating System. http://www.ros.org

[17] ROS, comunidad. http://answers.ros.org/questions/

[18] OSRF Blog: ROS & Gazebo at the DRC Finals http://www.osrfoundation.org/ros-gazebo-at-the-drc-finals/

[19] Free Software Foundation, https://www.fsf.org

[20] ROS-Industrial. http://rosindustrial.org/

[21] OROCOS. http://www.orocos.org/

[22] Orocos RTT and ROS integrated. http://www.willowgarage.com/blog/2009/06/10/orocos-rtt-and-ros-integrated

ISBN 978-84-15914-12-9 © 2015 Comité Español de Automática de la IFAC (CEA-IFAC)

[23] ROS en Android http://wiki.ros.org/android?distro=indigo y https://github.com/ros-android

[24] Sensores para ROS http://wiki.ros.org/Sensors

[25] *Drivers* para ROS https://github.com/ros-drivers

[26] Gazebo en ROS http://wiki.ros.org/gazebo_ros_pkgs

[27] GitHub de ROS http://wiki.ros.org/RecommendedRepositoryUsage/CommonGitHubOrganizations

[28] OpenHab http://www.openhab.org/

[29] Puente entre OpenHab y ROS: iot_bridge http://wiki.ros.org/iot_bridge

[30] Simple Open EtherCAT Master or SOEM http://docs.ros.org/hydro/api/ethercat_soem/html/index.html