Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de Septiembre de 2018
https://doi.org/10.17979/spudc.9788497497565.0546

CONSIDERACIONES SOBRE LA UTILIZACION DEL MICROCONTROLADOR ARDUINO COMO SISTEMA DE ADQUISICIÓN DE DATOS

Antonio José Calderón Godoy, Isaías González Pérez
Escuela de Ingenierías Industriales, Universidad de Extremadura, ajcalde@unex.es, igonzp@unex.es

Resumen

Los sistemas de adquisición de datos juegan un papel fundamental en todos los ámbitos de la ciencia y de la tecnología. En este trabajo se presentan y analizan tres soluciones basadas en el microcontrolador Arduino como tarjeta de adquisición de datos de bajo coste y carácter open source. Se describen y comparan las diferentes configuraciones analizadas, y se muestran y discuten los resultados obtenidos.

Palabras Clave: Arduino, adquisición de datos, open source.

1 INTRODUCCIÓN

Los Sistemas de Adquisición de Datos (SAD) tienen como funciones básicas la toma de señales del mundo real (analógico) y el tratamiento de los mismos de forma que puedan ser manejados por un computador u otro sistema digital. Generalmente, el proceso de conversión analógico/digital de estas señales recorre tres etapas: muestreo, cuantificación y codificación.

En el mercado existe desde hace muchos años una gran oferta de tarjetas/sistemas, abarcando todo tipo de campos y de propósitos. Desde los primeros dispositivos controlados por un panel propio hasta los sofisticados y flexibles sistemas de hoy en día, tenemos una gran variedad de posibilidades de elección y configuración. La mayor parte de los equipos actuales están preparados para trabajar en un entorno gráfico, el cual es muy útil para visualizar y analizar las señales registradas.

Uno de los principales inconvenientes de estos sistemas de adquisición es su elevado precio y la necesidad de adquirir licencias para la utilización de las herramientas software asociadas. En muchos casos, la finalidad para la que son requeridos no es de tal importancia o tan exigente como para hacer un desembolso de elevada magnitud. En otras, se prefiere utilizar un programa conocido y ya adquirido para el procesamiento de datos.

El objetivo del presente trabajo es diseñar un SAD de bajo coste capaz de recoger gran cantidad de señales procedentes de un proceso exterior a éste. La información a almacenar se registra desde cualquier sensor o circuito objeto de interés y dentro de un rango de voltaje. Estos valores se obtendrán junto a una marca de tiempo de adquisición, así podrán ser tratados posteriormente mediante un programa informático en un computador.

El SAD debe ser a su vez económico, eficiente y portable. Estas condiciones son muy importantes puesto que el cumplimiento de todas hará del sistema una herramienta muy útil y accesible a cualquier usuario.

En este trabajo se recogerán todos los pasos llevados a cabo para realizar un SAD basado en la plataforma Open Source Hardware Arduino [1].

En general, la tecnología open source está recibiendo cada vez más atención desde los ámbitos científico y tecnológico durante los últimos años [12]. La plataforma Arduino es el ejemplo más ilustrativo de las tendencias open source en cuanto al caso de equipos hardware. Otros populares dispositivos de carácter abierto y bajo coste son Raspberry Pi, Intel Edison, Phidget, OpenDAQ o BeagleBone. La filosofía de código abierto/open source conlleva que se dispone de una amplia variedad de información, ejemplos y tutoriales compartida por una comunidad colaborativa a través de Internet, promoviendo desarrollos basados en este tipo de equipos. Así, el bajo coste, la facilidad de uso y la gran disponibilidad han abierto una puerta a la democratización de la electrónica [11]. A nivel científico y académico, Arduino actúa tanto como componente nuclear como auxiliar en múltiples aplicaciones en campos como la automática, la adquisición de datos y la ingeniería en general [4]. Su versatilidad queda demostrada en numerosas y recientes publicaciones. Por ejemplo, fuera de la esfera de la adquisición de datos, se usa para robótica humanoide [6], sistemas de laboratorio para educación en ingeniería [7, 15, 20, 24], comunicaciones en el protocolo ZigBee [18], sistemas ciber-físicos [10, 16], control de invernaderos [22] o como interfaz entre autómatas

programables y sensores [13]. Respecto a la adquisición de datos, se utiliza en instrumentación para agricultura [14, 19], monitorización medioambiental [2, 5, 8], y para monitorizar sistemas energéticos como paneles fotovoltaicos [9, 21], producción de biogás [3], o pilas de combustible de hidrógeno [4, 23].

Se expondrán distintas configuraciones del hardware, así como sus ventajas e inconvenientes. Se comenzará por la adquisición de datos en la memoria interna del microcontrolador, siguiendo por el almacenamiento en una memoria externa, para finalizar con una configuración Maestro – Esclavo y así mejorar el periodo de muestreo de la toma de datos.

Una vez que la tarjeta esté conectada a las señales que se desean muestrear, el usuario podrá configurar los parámetros del muestreo obteniendo los resultados en un archivo .txt. Después, con la ayuda de un programa como puede ser Matlab o Excel, se podrá trabajar sobre esos valores.

El resto de la contribución se organiza como sigue. La sección 2 describe los elementos hardware y software empleados. En la sección 3 se analizan las tres soluciones propuestas. Finalmente, se presentan las conclusiones del trabajo.

2 HARDWARE/SOFTWARE UTILIZADO

El hardware utilizado para el desarrollo de este trabajo se basa en la placa Arduino Mega 2560 R3 junto con la shield Arduino Ethernet R3, Arduino Ethernet V2.0.

Arduino Mega 2560 R3 es una placa basada en el microcontrolador ATmega2560 que trabaja a una frecuencia de 16 MHz. Sus características más destacables son: 54 pines de entrada/salida digitales (de los cuales 15 pueden ser usados como salidas analógicas PWM), 16 entradas analógicas y 4 receptores/transmisores serie TTL-UART. Consta de una memoria flash de 256 KB de los cuales 8 KB son utilizados por el gestor de arranque o bootloader, 8 KB de SRAM y 4 KB de memoria EEPROM en la cual se puede leer y escribir mediante el uso de una librería.

Arduino shield Ethernet R3 añade la capacidad de conectar la placa Arduino a una red cableada TCP/IP. Está provisto de un chip microcontrolador W5100 que se configura con la librería de programación Ethernet, incluida por defecto en la plataforma de programación. Otro punto de interés de esta shield desde el punto de vista de este trabajo es el puerto para tarjetas microSD. Este puerto tiene el mismo bus de comunicación que la aplicación Ethernet por lo que sólo es posible tener activo uno de ellos. El uso de la tarjeta microSD es posible cargando la librería SD. Para el correcto funcionamiento de la tarjeta, hay que dejar libre el pin digital 4, para la línea SS (Slave Select) que es la encargada de elegir el dispositivo de comunicación entre todos los esclavos conectados.
Como hardware adicional se han utilizado tarjetas de memoria microSD Kingston. La capacidad de la tarjeta SD utilizada es de 8GB. Se han elegido de clase 10, lo que significa que la velocidad de escritura está limitada a 10Mb/s.

El paquete software utilizado para la programación y configuración de las placas Arduino es el Entorno de Desarrollo Integrado IDE (Integrated Development Enviroment). IDE consiste en un conjunto de herramientas software que permite a los programadores poder desarrollar sus programas (escribir y editar los programas o sketch). El programa contiene las librerías necesarias para la utilización de los módulos descritos. Además de las librerías antes mencionadas, en el presente trabajo se ha empleado la librería SPI, que implementa la comunicación entre Arduino (que actúa como maestro) y otros dispositivos externos (que actúan como esclavos).

3 SOLUCIONES ANALIZADAS

3.1 SAD CON ALMACENAMIENTO EN LA MEMORIA INTERNA

El punto de partida para el desarrollo de la propuesta es realizar un muestreo almacenando los datos obtenidos en la memoria interna de la placa. Con ello se pretende realizar el muestreo de la manera más rápida posible y con la menor inversión económica, puesto que sólo emplearemos una placa Arduino. El modelo Arduino MEGA dispone de 16 entradas analógicas, señaladas como “A0”, “A1” … ”A15”. El voltaje al que pueden ser conectadas las entradas tiene que estar dentro del rango 0 – 5V. Cuenta con un conversor analógico digital A/D con una resolución de 10 bits. Por lo tanto, la precisión en las muestra será 5/1024=0.049V.

Para poder establecer el periodo mínimo de muestreo se ha de determinar el tiempo que tarda la placa en guardar el valor de la entrada analógica junto al tiempo de la adquisición. Para esta tarea se programa un sketch para guardar la lectura de una entrada analógica junto a su marca de tiempo antes y después de la lectura analógica. Se comprobará el tiempo de adquisición, así como el desfase temporal entre las muestras consecutivas y se establecerá el periodo mínimo de muestreo. Se ha realizado un programa

para la toma de 200 muestras de una única entrada analógica (A0) cuyo código se muestra a continuación.

```
for(int i = 0; i <= MUESTRAS - 1; i++) {
    for(int j = 0; j <= MAXENTRADAS - 1;
j++) {
        tiemposAnt[i][j] = micros();
        valores[i][j] = analogRead(j);
        tiemposPos[i][j] = micros();
        }
        }
```

Los tiempos se almacenan en matrices con formato *unsigned long*, que es el tipo de dato en el que devuelve el tiempo la instrucción *micros()*. De esta manera se puede determinar el tiempo que tarda en adquirir un dato analógico y el que transcurre entre dos lecturas consecutivas de una entrada. Los resultados se analizan en una tabla Excel siendo los datos más característicos los recogidos en la tabla 1.

Tabla 1: Tiempos de lectura de la solución con almacenamiento en memoria interna.

T_1	Va	T_2	T_{LEC}	ΔT
56	409	268	212	
272	409	388	116	120
392	409	508	116	120
5432	400	5548	116	120
5552	400	5668	116	120
5672	400	5788	116	120
5792	400	5908	116	120
5912	399	6028	116	120
6032	399	6152	120	124
6156	399	6268	112	116
6272	399	6388	116	120
6392	399	6508	116	120
8072	395	8188	116	120
8200	395	8316	116	128

Siendo:

T_1 : Tiempo antes de la lectura
V_a : Valor de la lectura analógica
T_2 : Tiempo después de la lectura
T_{LEC} : Tiempo de lectura de una entrada
ΔT : Tiempo entre dos lecturas consecutivas

Del análisis de la tabla 1 se desprende que el tiempo entre lecturas se mantiene entre 116µs y 120µs. Este desfase de 4µs es debido a la tolerancia que tiene la ejecución de la instrucción de tiempo *micros()* en la placa Arduino. El número de muestras que se pueden almacenar en la memoria SRAM depende tanto de la tarjeta empleada, como de la memoria consumida para el resto de variables del programa. Simplificando el Sketch para que únicamente tome una marca de tiempo por muestra, se pueden realizar aproximadamente 1300 muestras en la tarjeta Arduino MEGA, repartidas entre las entradas seleccionadas y ocupando un 97% de la memoria dinámica (limitaciones).

Para comprobar la viabilidad del método se recurre a un caso práctico sencillo, como el circuito RC de la figura 1, para visualizar el proceso de carga/descarga de un condensador.

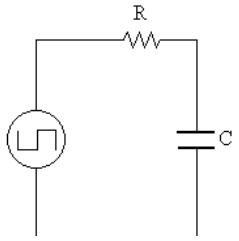

Figura 1: Circuito de carga/descarga de un condensador

El circuito se alimenta con una onda cuadrada de periodo T = 2ms y amplitud V = 5V. Se ha empleado un condensador de C = 10nF y una resistencia de R = 22k Ω. Con estos valores se obtiene una constante de tiempo τ = 0.22 ms. El periodo estacionario, tanto en el proceso de carga como en el de descarga del condensador, se alcanzará al en un tiempo de 5 τ = 1.1ms. Configurando la tarjeta con los siguientes valores:

- MUESTRAS: 50.
- MAXENTRADAS: 1.
- PERIODO: 200.

se obtienen los valores expuestos en la tabla 2.

Tabla 2: Valores de muestreo.

T_1	Va	ΔT
260	1023	
472	1023	212
656	687	184
860	303	204
1060	141	200
1260	67	200
1460	32	200
1660	15	200
1860	7	200
2060	4	200
2260	2	200
2460	0	200

2660	294	200
2860	739	200
3060	941	200
3260	1023	200
3460	1023	200

Con ayuda de la herramienta informática Matlab, se puede representar los datos anteriores en una gráfica como la mostrada en la figura 2.

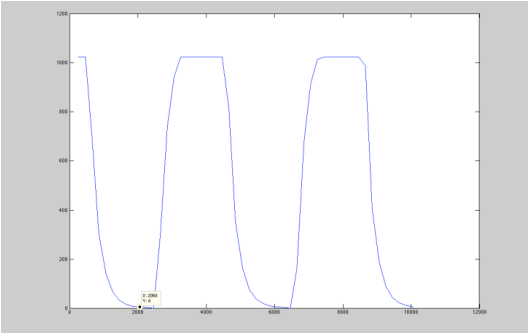

Figura 2: Ciclo de carga/descarga de un condensador

3.2 SAD CON ALMACENAMIENTO EN LA MEMORIA MICRO-SD

En este apartado se aborda la inclusión de una memoria microSD en el sistema de adquisición para el almacenamiento masivo de datos. El objetivo de esta propuesta es incrementar el volumen de datos almacenados. Además, se dotará a la tarjeta de la posibilidad de configuración mediante una página web. Para la materialización de esta solución se utiliza la shield Ethernet comentada en apartados anteriores. El módulo Ethernet admite distintas configuraciones. En esta aplicación se configurará la shield Ethernet para que la tarjeta actúe como servidor web dentro de la propia red. Para habilitar la comunicación Ethernet se deben incluir las líneas de código mostradas en el siguiente Sketch.

```
// Configuración de la conexión Ethernet
Ethernet.begin(mac, ip);
server.begin();
Serial.print("server is at ");
Serial.println(Ethernet.localIP());
```

Arduino leerá el formulario HTML procedente de la web, en el cual estarán los valores de configuración del sistema de adquisición. Estos valores serán los mismos que en la solución anterior en lo referente a periodo de muestreo, número de entradas y muestras. Una vez que la tarjeta microSD se acopla al zócalo de la shield Ethernet se inicializa. Este proceso consiste en asignar el pin 4 a la tarjeta microSD por medio de la función *SD.begin(4)*. Si esta función no devuelve ningún error (valor devuelto igual a 0), se pueden utilizar las instrucciones específicas para esta tarjeta. Antes de la creación del fichero en la tarjeta microSD, se comprueba si existe uno con el mismo nombre y, si es así, lo elimina para poder crear uno nuevo y trabajar sobre él. A continuación se crea un fichero con el nombre deseado y se especifica que éste va a ser de lectura/escritura (FILE_WRITE). Este procedimiento se lleva a cabo con las siguientes líneas de código:

```
// Coprobar si el fichero existe en la SD y
borrarlo en caso afirmativo
if(SD.exists("almacen.txt") != 0) {
    SD.remove("almacen.txt");
}
// Crear un fichero de lectura / escritura de
nombre "almacen"  miarchivo =
SD.open("almacen.txt", FILE_WRITE);
```

Para determinar el mínimo periodo de muestreo alcanzable con esta opción, se han analizado dos tipos de tarjetas microSD, a saber, de clase 4 y clase 10. En ambos casos se ha seguido el mismo procedimiento para esta tarea. La tabla 3 muestra los resultados obtenidos con una tarjeta microSD de clase 10.

Tabla 3: Valores de muestreo para tarjeta microSD clase 10.

Nº	T_1	Va	ΔT
1	3601448	383	
2	4064264	342	462816
3	4064936	333	672
43	4091992	299	680
44	4232012	289	140020
45	4232688	298	676
86	4260328	243	680
87	4402872	249	142544
88	4403552	237	680
89	4404228	230	676
128	4430496	197	668
129	4431168	209	672
130	4453672	212	22504
131	4454344	212	672

Se observa que cada 40 – 43 muestras, aparece un retardo debido a la escritura en la tarjeta microSD. También se puede comprobar que los tiempos de escritura alcanzan valores comprendidos entre 668 - 680µs, muy inferiores a los logrados con la tarjeta de clase 4 (entre 1252µs y 1718µs). Por tanto, el análisis

de esta metodología se realizará utilizando tarjetas de clase 10.

Se han realizado pruebas del funcionamiento del programa para distintos casos, con periodos de muestreo comprendidos entre 1000 y 10000µs, entre 1 y 12 entradas y 1000 muestras por entrada. A partir de los resultados obtenidos se puede concluir que se producen retardos cada cierto tiempo en la escritura de valores en la memoria. Los instantes de muestreo en los que se producen dichos retardos dependen del tiempo de muestreo, es decir, de la cantidad de información que se guarda en la memoria en un tiempo determinado. Por ello, no se puede asegurar un periodo de muestro fiable para un tiempo menor de 100ms. Esto hace que esta solución sea poco idónea para su utilización como SAD.

3.3 SAD EN CONFIGURACIÓN MAESTRO – ESCLAVO

En este apartado se propone una solución alternativa para la adquisición de datos y su almacenamiento masivo en una memoria micro SD. Esta solución consiste en una configuración Maestro – Esclavo. Para ello se requieren tres Arduinos equipados con su correspondiente shield Ethernet y a su vez con una memoria microSD insertada en estos módulos. El uso de este material hace que la aplicación incremente su coste al triple lo que puede ser poco atractivo, pero ante las necesidades de grabar una gran cantidad de datos es una solución muy útil.

La idea de esta configuración surge del análisis de los datos obtenidos con el planteamiento de la subsección anterior. Se observa que aproximadamente cada 40 muestras la adquisición de datos sufre un retardo. Este retardo se puede enmascarar haciendo trabajar dos Arduinos (los dos esclavos) de forma alternativa. Así se tendrá una tarjeta tomando datos durante un número determinado de ciclos y a continuación ésta deja de trabajar entrando en funcionamiento la segunda, repitiendo el proceso cíclicamente. La activación de las tarjetas se realizará mediante interrupciones generadas por el maestro.

La figura 3 muestra las conexiones para llevar a cabo la interrupción y las comunicaciones Serial para la recogida y ordenación final de datos. El Maestro genera un pulso que es recogido por el pin de interrupción 0 de los Arduinos Esclavos. El Esclavo 1 tiene habilitada la interrupción mediante un flanco de subida y el Esclavo 2 mediante una de bajada. La comunicación para la fusión de los datos de los Esclavos se realiza mediante una conexión Serial. Estas fases de adquisición cíclica se esquematizan en la figura 4.

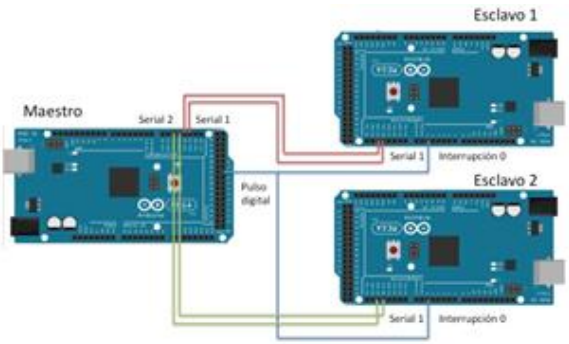

Figura 3: Esquema de conexionado Maestro - Esclavo

Figura 4: Fases de la adquisición cíclica de los dos esclavos

Durante el semiperiodo alto, el primer esclavo realiza el número de muestreos que tenga programados mientras el segundo esclavo almacena los valores de su fase de muestreo anterior en la tarjeta microSD. En el semiperiodo bajo se intercambian los roles; esto es, el segundo esclavo realiza el número de muestreos que tenga programados mientras el primer esclavo almacena los valores de su fase de muestreo anterior en la tarjeta microSD. De esta forma el muestreo es realizado por ambos esclavos de forma alterna. El primer factor a establecer es el número de muestras que se ha de tomar en cada ciclo de interrupción (ciclo de muestreo). Para la determinación de este número se ha de tener en cuenta el número de entradas analógicas seleccionadas, ya que no se puede finalizar un ciclo de muestreo hasta no leer el total de entradas previstas.

Puesto que los datos del muestreo estarán repartidos en dos tarjetas de memoria y a su vez separados dentro de la misma tarjeta un número de filas igual a las entradas habilitadas durante el proceso, al final del programa se procede al ensamblado de estos resultados en un mismo fichero y en el orden adecuado. Para sincronizar correctamente este proceso de fusionado, el Maestro genera una referencia de tiempo común a los dos esclavos (figura 5).

Una vez hechas las consideraciones previas y bajo la premisa de alcanzar un periodo de muestreo de 10 ms, se desarrolla esta propuesta. El sistema consta de una placa que actúa como Maestro la cual recoge la configuración introducida en los campos de una página web. Después calcula el número de muestras

del que consta cada ciclo de muestreo y transfiere todos estos datos a los Esclavos. Los Esclavos configuran sus variables con la información procedente del Maestro y pasan al siguiente estado que es la espera de la activación de la interrupción para comenzar con el muestreo. El muestreo de las entradas analógicas por parte de los Esclavos está dividido en dos fases: una primera en la que se recogen los datos del muestreo en la memoria interna y la segunda que transfiere estos valores a la memoria microSD. La transferencia de datos de la memoria interna a la microSD se produce en el tiempo en el que el otro Esclavo realiza el muestreo de las entradas. Así se consigue el doble de tiempo para el muestreo y almacenamiento de la información. Una vez finalizado el muestreo, el Maestro activa el último paso que es la recogida y adaptación en tiempo de los valores procedentes de los Esclavos. Esta información es almacenada en la memoria microSD del Maestro pudiéndose analizar los datos de manera similar a la expuesta en la anterior solución.

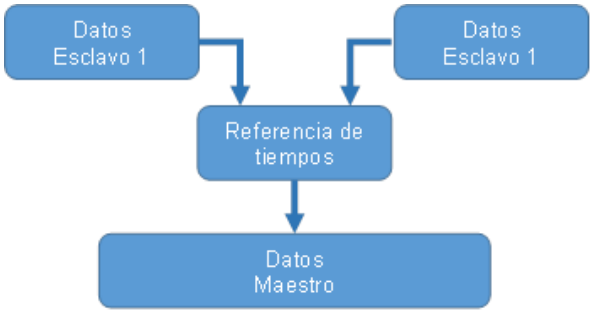

Figura 5: Procedimiento de adquisición cíclica

Con objeto de comprobar la viabilidad como SAD de esta propuesta, se muestrea una señal cuadrada de periodo y amplitud conocidos.
La representación de los datos obtenidos utilizando una gráfica obtenida con Matlab se muestra en la figura 6.

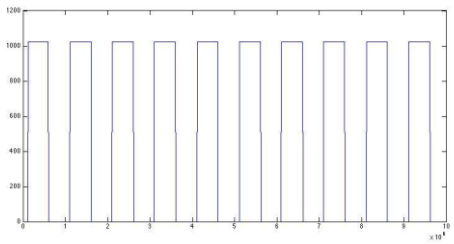

Figura 6: Onda cuadrada obtenida

Se observa que para una señal del periodo adecuado para que nuestra tarjeta pueda tomar las suficientes muestras, el resultado es aceptable. Existen pequeñas diferencias de tiempos entre las muestras de cambio de Esclavo, pero el muestreo vuelve al periodo establecido en no más de dos muestras. Es preciso un tiempo de muestreo mínimo de 10ms (seleccionando una entrada), lo cual constituye un hándicap a la hora de muestrear señales de frecuencia elevada. Otro punto a mejorar es la transferencia de datos de los Esclavos al Maestro ya que se ha observado que en ocasiones no finaliza completamente este proceso.

4 CONCLUSIÓN

Una vez analizadas todas las propuestas para el SAD basado en Arduino se puede determinar lo siguiente:

- La configuración que permite un menor tiempo de muestreo es la primera (200µs), la cual almacena los valores en la memoria interna. La segunda SAD "almacenamiento en microSD" tiene un tiempo mínimo de 100ms. La tercera aplicación "Maestro - Esclavo" por el contrario tiene un tiempo mínimo de muestreo de aproximadamente 10ms.
- Si se atiende al número de muestras que se puede adquirir, el SAD que obtiene los peores resultados es el primero (1300 muestras). Por el contrario la segunda y tercera dependen de la memoria microSD instalada. La capacidad máxima es de 32GB y el tamaño de una muestra es de 10Byte. Por lo tanto el número máximo de muestras es: 32x233 / 10x23 = 3,2x230 muestras = 1073741824 muestras.
- En precio, el SAD más económico es la primera que consta de un Arduino Mega 2560 con un precio de 41,75€. Para la segunda es necesario un Arduino Mega 2560, una shield Ethernet V2 y una tarjeta microSD de 8 GB clase 10; teniendo un coste de 74.02€. Por último, la tercera aplicación compuesta de tres Arduinos Mega 2560, tres shields Ethernet V2 y tres tarjetas microSD de 8 GB clase 10 tiene un precio de 222.06€.

Los resultados obtenidos indican que la mejor opción dependerá del caso de aplicación particular, sirviendo este trabajo para facilitar la decisión a la hora de diseñar un SAD utilizando Arduino.

Agradecimientos

Este trabajo ha sido financiado con Fondos FEDER (Programa Operativo FEDER de Extremadura 2014-2020). Ayuda a Grupos de Investigación de la Junta de Extremadura (ref. GR18159.)

English summary

CONSIDERATIONS ABOUT THE USE OF ARDUINO MICROCONTROLLER AS DATA ACQUISITION SYSTEM

Abstract

Data acquisition systems play a paramount role in all scientific and technologic fields. This work presents and analyzes three solutions based on the Arduino microcontroller. This device acts as low-cost and open source data acquisition card. The different configurations are described and compared, and the achieved results are reported and discussed.

Keywords: Arduino, data acquisition, open source.

Referencias

[1] Arduino web: www.arduino.cc

[2] Arroyo, P., Lozano, J., Suarez, J.I., Herrero, J.L., Carmona, P., (2016) "Wireless Sensor Network for Air Quality Monitoring and Control", *Chemical Engineering Transactions*, vol. 54, pp. 217-222.

[3] Calderón, A.J., González, I., (2018) "Biogas Analyzer Based on Open Source Hardware: Design and Prototype Implementation", *Sensors & Transducers*, vol. 220, pp. 31-36.

[4] Calderón, A.J., González, I., Calderón, M., Segura, F., Andújar, J.M., (2016) "A New, Scalable and Low Cost Multi-Channel Monitoring System for Polymer Electrolyte Fuel Cells", *Sensors*, vol. 16(3), pp. 349. DOI: 10.3390/s16030349.

[5] Carland, J., Umeda, M., Wilkey, T., Oberbeck, A., Cumming, J., Parks, N., et al., (2013) "Meteorological Wireless Sensor Networks", *Sensors & Transducers*, vol. 160, pp. 118-124.

[6] Cela, A.; Yebes, J.J.; Arroyo, R.; Bergasa, L.M.; Barea, R.; López, E., (2013) "Complete low-cost implementation of a teleoperated control system for a humanoid robot", *Sensors*, 13(2), 1385-1401; DOI: 10.3390/s130201385.

[7] Chacón, J., Saenz, J., de la Torre, L., Díaz, J.M., Esquembre, F., (2017) "Design of a Low-Cost Air Levitation System for Teaching Control Engineering", *Sensors*, vol. 17, pp. 2321. DOI: 10.3390/s17102321.

[8] Ferdoush, S., Li, X., (2014) "Wireless Sensor Network System Design Using Raspberry Pi and Arduino for Environmental Monitoring Applications", *Procedia Computer Science*, vol. 34, pp. 103–110.

[9] Gad, H.E., Gad, H.E., (2015) "Development of a new temperature data acquisition system for solar energy applications", *Renewable Energy*, vol. 74, pp. 337-343. DOI: 10.1016/j.renene.2014.08.006.

[10] García, M.V., Irisarri, E., Pérez, F., Estévez, E., Marcos, M., (2016), "OPC-UA Communications Integration using a CPPS architecture", *Proceedings of IEEE Ecuador Technical Chapters Meeting, Guayaquil, Ecuador*. DOI: 10.1109/ETCM.2016.7750838.

[11] Garrigós, A., Marroquí, D., Blanes, J.M., Gutiérrez, R., Blanquer, I., Cantó, M., (2017) "Designing Arduino electronic shields: experiences from secondary and university courses", *Proceedings of IEEE Global Engineering Education Conference (EDUCON)*. DOI: 10.1109/EDUCON.2017.7942960.

[12] González, I., Calderón, A.J., Andújar, J.M., (2017) "Novel Remote Monitoring Platform for RES-Hydrogen based Smart Microgrid", *Energy Conversion and Management*, vol. 148, pp. 489-505. DOI: 10.1016/j.enconman.2017.06.031.

[13] Kayande, U. B., Jaspreetkaur, P., Kulkarni, R. M., Mahajan, S. B., Sanjeevikumar, P., (2018) "Digitally Controlled Hybrid Liquid Level Detection System Using Programmable Logic Controller and Microcontroller", *Lecture Notes in Electrical Engineering*, vol. 442, pp. 365-374. DOI: 10.1007/978-981-10-4762-6_35.

[14] Lampson, B.D., Han, Y.J., Khalilian, A., Greene, J.K., Degenhardt, D.C., Hallstrom, J.O., (2014) "Development of a portable electronic nose for detection of pests and plant damage", *Computers and Electronics in Agriculture*, vol. 108, pp. 87-94. DOI: /10.1016/j.compag.2014.07.002.

[15] Mejías, A., Reyes, M., Márquez, M.A., Calderón, A.J., González, I., Andújar, J.M., (2017) "Easy handling of sensors and actuators over TCP/IP Networks by Open Source Hardware/Software", *Sensors*, vol. 17, 94, 2017. DOI: 10.3390/s17010094.

[16] Müller, M., Wings, E., Bergmann, L., (2017) "Developing open source cyber-physical systems for service-oriented architectures using OPC UA", *Proceedings of IEEE 15th International Conference on Industrial Informatics (INDIN)*. DOI: 10.1109/INDIN.2017.8104751.

[17] Papageorgas, P., Piromalis, D., Antonakoglou, K., Vokas, G., Tseles, D., Arvanitis, K.G., (2013) "Smart Solar Panels: In-situ monitoring of photovoltaic panels based on wired and wireless sensor networks", *Energy Procedia*, vol. 36, pp. 535-545.

[18] Pereira, R., Figueiredo, J., Melicio, R., Mendes, V.M.F., Martins, J., Quadrado, J.C., (2015) "Consumer energy management system with integration of smart meters", *Energy Reports*, vol. 1, pp. 22-29. DOI:10.1016/j.egyr.2014.10.001.

[19] Piromalis, D., Arvanitis, K., (2016) "SensoTube: A scalable hardware design architectures for wireless sensors and actuators network nodes in the agricultural domain", *Sensors*, vol. 16, pp. 1227. DOI: 10.3390/s16081227.

[20] Prada, M.A., Reguera, P., Alonso, S., Morán, A., Fuertes, J.J., Domíngez, M., (2016) "Communication with resource-constrained devices through MQTT for control education", *IFAC-PapersOnLine*, vol. 49(6), pp. 150-155. DOI: 10.1016/j.ifacol.2016.07.169.

[21] Rahman, M.M., Selvaraj, J., Rahim, N.A., Hasanuzzaman, M., (2018) "Global modern monitoring systems for PV based power generation: A review", *Renewable and Sustainable Energy Reviews*, vol. 82(3), pp. 4142-4158. DOI: 10.1016/j.rser.2017.10.111.

[22] Robles, C., Callejas, J., Polo, A., (2017) "Low-Cost Fuzzy Logic Control for Greenhouse Environments with Web Monitoring", *Electronics*, vol. 6, pp. 71. DOI: 10.3390/electronics6040071.

[23] Segura, F., Bartolucci, V., Andújar, J.M., (2017) "Hardware/Software Data Acquisition System for Real Time Cell Temperature Monitoring in Air-Cooled Polymer Electrolyte Fuel Cells", *Sensors*, vol. 17(7), pp. 1600. DOI: 10.3390/s17071600.

[24] Tejado, I., Serrano, J., Pérez, E., Torres, D., Vinagre, B., "Low-cost Hardware-in-the-loop Testbed of a Mobile Robot to Support Learning in Automatic Control and Robotics", *IFAC-PapersOnLine*, vol. 49(6), pp. 242-247, 2016. DOI: 10.1016/j.ifacol.2016.07.184.