

Análisis sistema embebido para monitoreo de velocidad a bordo de un vehículo baja SAE®

Jesús Martin Silva Aceves¹, Jorge Arturo Pérez Venzor¹, Eduardo D. Medrano Saucedo¹, David García Chaparro¹, Francisco Javier Enríquez Aguilera¹, Abel Eduardo Quezada Carreón¹

¹ Universidad Autónoma de Ciudad Juárez

Resumen

El presente artículo tiene como finalidad un sistema embebido en dos distintas plataformas para monitoreo de revoluciones, así como la velocidad, para un vehículo de competencia BAJA SAE. Se detalla de modo secuencial la creación de dos códigos para cada una de las presentaciones, por ende, este desarrollo se traduce en dos programas distintos que poseen una función similar. El lenguaje de programación usado es C y se amolda de acuerdo a lo que cada compilador requiere. Las implementaciones de este trabajo de investigación son tarjetas listas para recibir pulsos externos previamente acondicionados. Uno sistemas tiene un carácter portátil y se maximizan sus funciones al momento de ser conectado a una computadora, el segundo tiene una orientación para permanecer a bordo del vehículo.

Palabras Clave: BAJA SAE, PIC, micros controladores, Lenguaje C.

Introducción

En el presente artículo se muestra una propuesta para un sistema embebido de monitoreo de velocidad lineal y revoluciones para un vehículo todo terreno BAJA SAE® (Society of Automotive Engineers, por sus siglas en inglés) de la Universidad Autónoma de Ciudad Juárez. La parte medular es el desarrollo de dos códigos en lenguaje C que se trabajan en dos distintas plataformas. Se detallan a fondo el desarrollo del código y su justificación en cada fase de funcionamiento del microcontrolador utilizado.

Se entiende por sistema embebido a todo aquel que se comunica con el mundo exterior por medio de datos digitales provenientes de un

CPU y que realiza una o pocas tareas específicas que se llevan a cabo en tiempo real [1]. La comunicación puede abarcar los siguientes puntos:

1. Interfaz directa con el usuario, ya sea por medio de botones o pantallas.
2. Mediciones de entrada, esto puede presentarse a partir de sensores.
3. Control de salidas, como el control de motores o actuadores.
4. Transferencia de datos o comunicación entre varios sistemas.

Método y Resultados

Simulación

Para realizar la implementación se utilizó PIC18 y para la simulación se utilizó al programa ISIS® del paquete Proteus®. Al realizar la compilación, se toma el archivo generado con extensión *.hex* y se realiza la simulación en el programa ISIS. Este archivo se encuentra en la carpeta donde se ha creado el código del programa realizado en MPLAB X. Previamente es necesario realizar las conexiones pertinentes, así como la selección del PIC utilizado.

Los elementos necesarios que se incluyen:

- Una pantalla LCD 20x4 caracteres y el PIC18F4620.
- Un generador de señales que funcionará para los cambios de pulso en los pines que realizan los conteos
- Botones para las funciones asignadas a cada uno de ellos.

¹ Instituto de Ingeniería y Tecnología, Universidad Autónoma de Ciudad Juárez

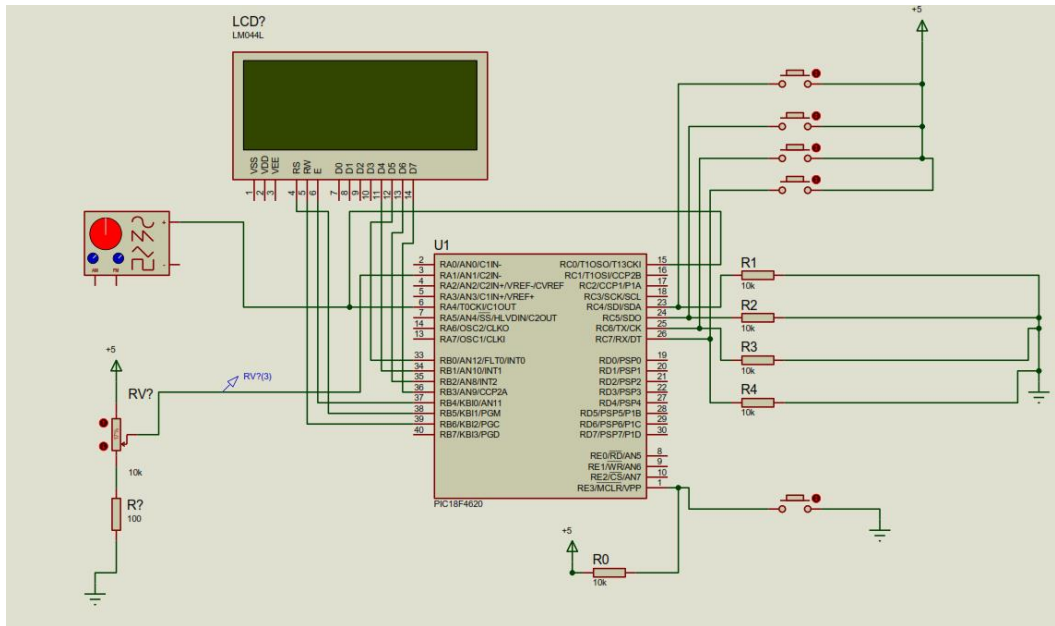


Figura 1. Tablas y cálculos

En la figura 2 se ilustra la primera pantalla que tiene una duración de 1 segundo y que ocurre al momento de iniciar el sistema.



Figura 2. Pantalla de inicio

Una vez transcurrido el tiempo de la pantalla de inicio, continua con la pantalla principal en la cual se despliegan la velocidad, las revoluciones de la polea primaria, las revoluciones de la polea secundaria y el reloj que demuestra el tiempo traspasado total. Para generar las cantidades que se ven en pantalla se utiliza un generador de señales que viene implementado dentro de la misma aplicación. De tal forma que el generador emite una señal cuadrada a un nivel de 5 V y con 370 hz.



Figura 3. Pantalla principal

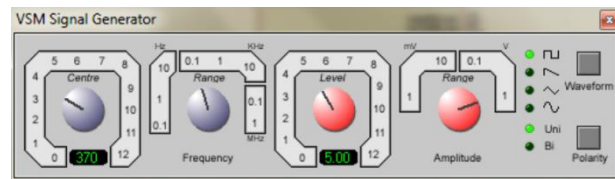


Figura 4. Generador de señales

Como método de comprobación de resultados, se consideran los 6 pulsos por revolución que se programan el radio de la llanta para el cálculo de las revoluciones y la velocidad lineal. Para el vehículo BAJA en el que se implementa el sistema donde se tiene: una rueda con un radio de 11.5 in, que equivale a .2921 m y una relación de transmisión de 9 (entrada con respecto a salida del reductor).

Para calcular las RPM se tiene que:

$$370 \frac{\text{ciclos}}{s} \left(\frac{60s}{1\text{min.}} \right) = 22,200 \quad (1)$$

$$\frac{22,200}{6} \text{RPM} = 3,700 \text{RPM} \quad (2)$$

El cálculo de la velocidad lineal a partir de la velocidad angular, que a su vez se expresa

$$v = 3,700 \frac{\text{rev.}}{\text{min}} \left(\frac{60 \text{ min}}{1h} \right) \left(\frac{2\pi(0.291m)}{\text{rev}} \right) \left(\frac{1km}{1000m} \right) 407.44 \frac{km}{h}$$

$$v = \frac{407.44 \frac{km}{h}}{9} = 45.27 \frac{km}{h}$$

de la siguiente manera: $\omega = 2\pi f$ y $v = \omega r$. Donde ω es la velocidad angular f es la frecuencia del sistema y r es el radio de la rueda.

Por lo tanto, se sustituye a partir de las revoluciones obtenidas y tomando en cuenta la relación de transmisión a la salida del reductor con un valor de 9:

A continuación, se procede con las funciones que los botones realizan:

- Botón 1. Cambio de pantalla

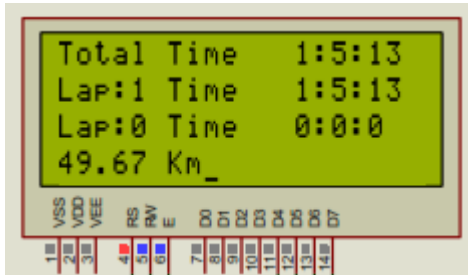


Figura 5. Segunda pantalla

- Botón 2. Reiniciar todos los tiempos.

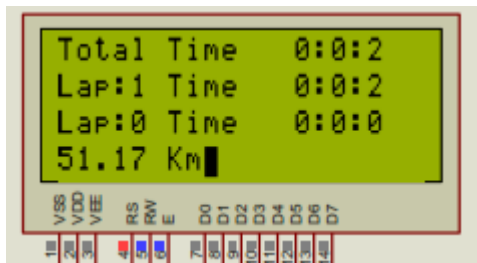


Figura 6. Tiempos reinicializado

- Botón 3. Incrementar la vuelta actual y pasar a línea inferior. Esta opción sirve para tener

una comparativa de la vuelta anterior con respecto a la actual.



Figura 7. Incrementar vueltas.

- Botón 4. Por último, la función para reiniciar el odómetro.

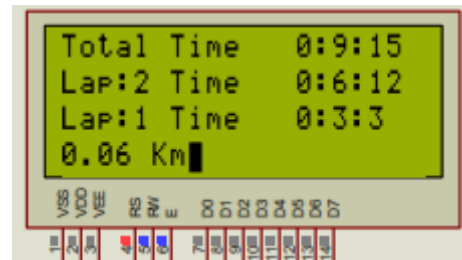


Figura 8. Odómetro reinicializado

En la figura 9 se observa la colocación de componentes para la realización final. Dicha

tarea se realiza en un software adicional llamado Circuit Wizard®

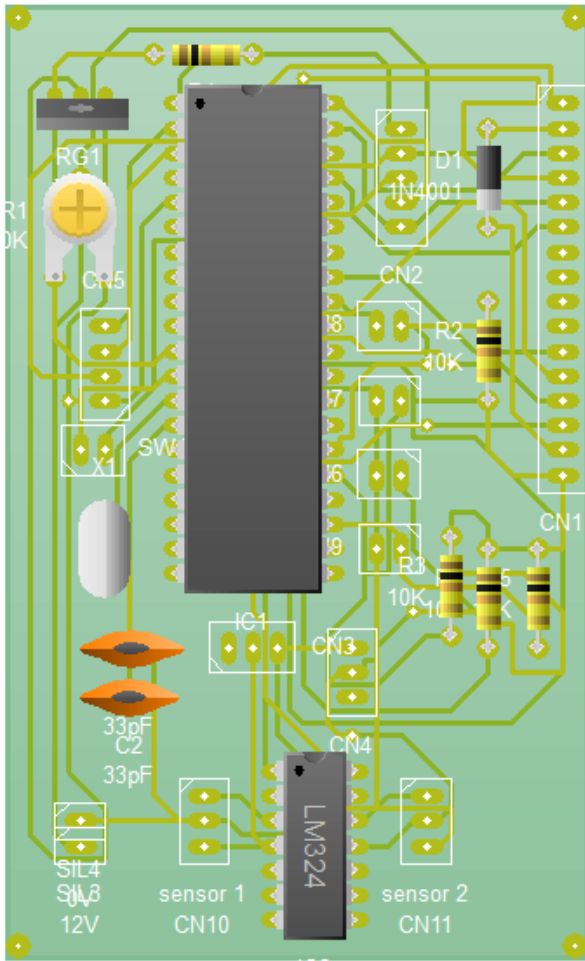


Figura 9. Tablilla final en circuit wizard.
Salida en las pantallas físicamente.



Figura 10. Pantalla de inicio. (Implementación física)



Figura 11. Pantalla principal (implementación física)



Figura 12. Segunda pantalla (implementación física).

Se implementa en la tarjeta ethernet una tablilla para acondicionar las señales introducidas por los pines 2 y 3. Los elementos necesarios para las tareas que realiza el sistema se ilustran en la imagen 13.



Figura 13. Arduino ethernet, microSD y tablilla

Las peculiaridades de este sistema son el hecho de almacenar todos los datos de entrada en una microSD y adicionado a esto un monitoreo en vivo por medio de una computadora. Para comprobar este hecho se recurrió a un generador de señales figura 14. El sistema recibe cambios de pulso por dos pines distintos, para el caso del primero se considera que se recibían 3 pulsos por revolución y para el segundo solamente se considera un pulso por revolución. En la figura 14 se generan dos pulsos cuadrados a un nivel de 5 V y con distintas frecuencias (30hz y 60hz). En la tabla 1 se muestran los datos guardados en la microSD que ha pasado por los cálculos necesarios para la obtención de RPM a partir de la frecuencia que se presenta en cada uno de los pines. Para fines prácticos, la misma señal se hubo introducido por ambos pines con el fin de comprobar que los cálculos son correctos.



Figura 14. Frecuencias de generador de Señales.

Tiempo	Primario	Secundario
0:00:09	180	540
0:00:10	220	660
0:00:11	420	1260
0:00:12	600	1800
0:00:13	600	1800
0:00:14	600	1800
0:00:15	600	1800
0:00:16	600	1800
0:00:17	600	1800
0:00:18	600	1800
0:00:19	600	1800
0:00:20	600	1800
0:00:21	600	1800
0:00:22	940	2820
0:00:23	1200	3600
0:00:24	1200	3600
0:00:25	1200	3600
0:00:26	1200	3600
0:00:27	1200	3600

Tabla 1. Datos guardados en microSD

Para comprobar que los datos presentados son correctos solo es preciso realizar una sencilla operación considerando los pulsos por revolución para cada pin.

Si se sabe que se para los datos del primario se programó con un divisor de 3 y que para el secundario no existe ningún divisor, se tiene para el caso de 30Hz que:

$$30 \frac{\text{ciclos}}{s} \left(\frac{60s}{1 \text{ min.}} \right) = 1800 \text{RPM} .$$

$$\frac{1800 \text{RPM}}{3} \text{RPM} = 600 \text{RPM}$$

Y para el caso de 60 Hz, de manera similar se comprueban los datos que se despliegan en la tabla 1.

$$60 \frac{\text{ciclos}}{s} \left(\frac{60s}{1 \text{ min.}} \right) = 3600 \text{RPM} .$$

$$\frac{3600 \text{RPM}}{3} \text{RPM} = 1200 \text{RPM}$$

Los datos intermedios de 940 y 2820 se deben al momento es que se incrementó la frecuencia en el generador de señales al ajustar el control. Pero se puede ver que en ese momento el sistema detectó:

$$940RPM \cdot \left(\frac{1 \text{ min}}{60s}\right)(3) = 47Hz.$$

$$2820RPM \left(\frac{1 \text{ min}}{60s}\right) = 47Hz.$$

Se graficaron los datos con el programa MegunoLink, el cual tiene la funcionalidad de crear gráficas en vivo al momento de tener conectada la tarjeta en la computadora. Los datos son meramente representativos, se pretende demostrar que el sistema adiciona esta característica.

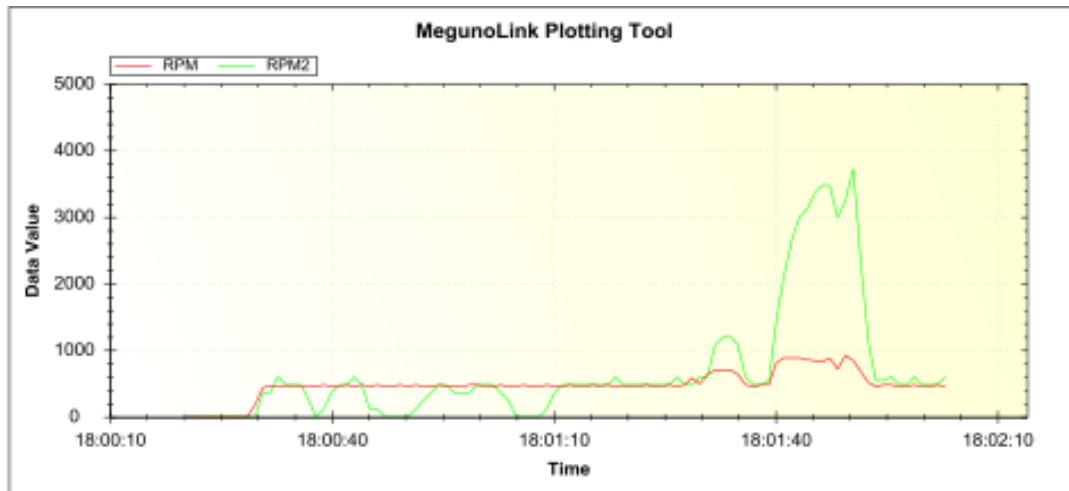


Figura 15. Prueba en tiempo real 1

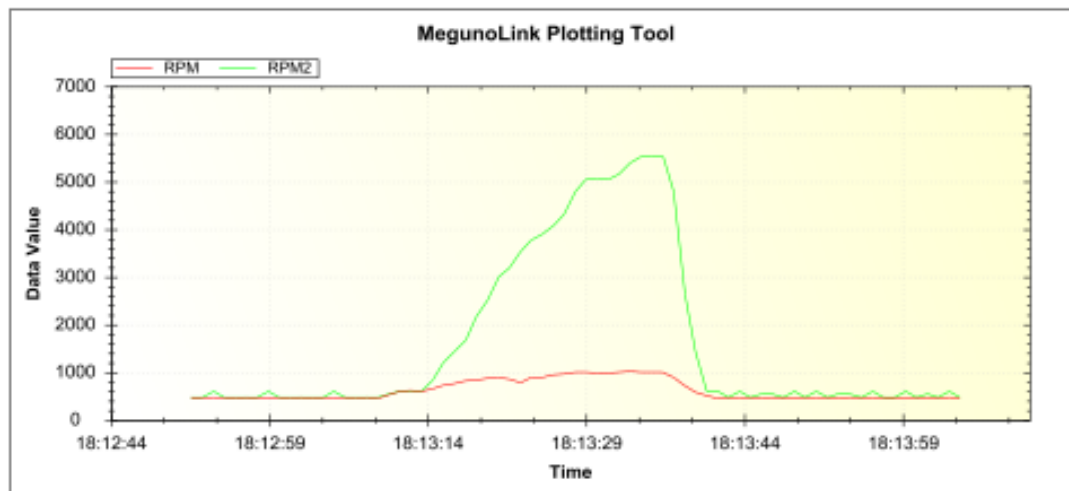


Figura 16. Prueba de tiempo 2.

Estos datos fueron obtenidos cuando el vehículo se encontraba con la ruedas suspendidas en el aire, por lo que no se encontraba con ninguna fricción, de tal forma que se pueda ver un crecimiento en la gráfica de las revoluciones.

Conclusiones

Al comparar dos plataformas para distinguir las ventajas que cada una presentaba se puede tomar una decisión cual es la más robusta para la aplicación. Todas las conclusiones que se buscan están enmarcadas en un ámbito didáctico. Éstas se enfocan en tres aspectos principales: la implementación física, la programación y el mercado.

La plataforma Arduino resulta bastante sencilla al momento de ser programada, además se ve muy favorecida por la gran proliferación de códigos en Internet. De esta forma, cualquier duda que llegue a surgir durante el proceso de reconocimiento puede ser resuelta con solo hacer una corta búsqueda en línea. Otra ventaja que muestra Arduino, es el hecho de ser una tarjeta lista para usarse, que no requiere de ningún otro aditamento para funcionar; es decir, solo se precisa conectarla a una computadora y el usuario está habilitado para comenzar a programar.

La plataforma Arduino resulta bastante sencilla al momento de ser programada, además se ve muy favorecida por la gran proliferación de códigos en Internet. De esta forma, cualquier duda que llegue a surgir durante el proceso de reconocimiento puede ser resuelta con solo hacer una corta búsqueda en línea. Otra ventaja que muestra Arduino, es el hecho de ser una tarjeta lista para usarse, que no requiere de ningún otro aditamento para funcionar; es decir, solo se precisa conectarla a una computadora y el usuario está habilitado para comenzar a programar.

Hay que señalar que Microchip ha liberado al mercado gran número de tarjetas listas para usarse, pero no resultan igual de amigables que Arduino, sino que trabajan con el mismo compilador proporcionado por la compañía. Además de que su propuesta de venta

es la difusión de gran variedad de micro-controladores individuales con una amplia gama de prestaciones. En el mismo tenor, es significativo señalar un evento reciente en Microchip. La compañía de la conocida M ha introducido al mercado tarjetas con esencia similar a Arduino, que funcionan con el mismo tipo de sentencias simples en código C que se implementan en la *plataforma azul*. Dicho acto ha demostrado la gran presión que ejerce en el mercado la sencillez y portabilidad de Arduino.

Sin embargo, cuando se requiere hacer un control preciso y secuencial sobre los datos de entrada, así como el manejo de funciones internas, la plataforma PIC mantiene una vasta ventaja en conjunto con la interfaz MPLAB. La etapa de depuración, que es una tarea realizada con mucha frecuencia al momento de programar en PIC's, se traduce en una ventaja abismal para el programador que requiere saber lo que está llevándose a cabo en todo momento dentro del integrado. Estos argumentos pueden ser respaldados con varias tareas que la plataforma Arduino no proporciona en su interfaz, por ejemplo: el monitoreo de los bits de los diversos registros, la activación de banderas para interrupciones, los incrementos en los registros de los temporizadores, las pausas intencionales del flujo normal de tareas, entre otros.

A partir de los análisis anteriores, se puede deducir que no existe una plataforma superior a otra, sino que depende de los alcances que el programador desea al momento de la implementación. Por lo que para fines que sean rápidos y prácticos, Arduino es sin duda la mejor opción. Por otra parte, cuando se trate de un asunto que requiera un monitoreo detallado del sistema y de los sucesos internos del integrado, PIC presenta todas las herramientas necesarias.

Referencias

- [1] Arístides Silva Otero, Mariela Mata de Gossi. “*La llamada revolución industrial*”, Caracas: Universidad Católica Andrés Bello, (2205) pp. 15-20
- [2] Andrés García Higuera, Fernando J. Castillo García. “CIM, el computador en la automatización de la producción”. Ediciones de la universidad de Castilla, La mancha (2007) pp. 70-79
- [3] T. Moriwaki. Multi-functional machine tool. *CIRP Annals - Manufacturing Technology*, 57 (2008) pp. 736–749.
- [4] Mikell P. Groover. Fundamentos de manufactura moderna, Pearson Prentice Hall (1997), p. 19
- [5] Chávez Conde E., Beltrán Carbajal F., Aguilar Valdez A., López Bobadilla L., Guardado Ruiz M. E. En el diseño de máquinas-herramientas automáticas tipo fresadora, *8º congreso Nacional de mecatrónica*. (2009).
- [6] Daniel Samper Martínez, Diseño y construcción de una máquina de control numérico con control en tiempo real mediante dispositivo apuntador, Tesis de licenciatura, Universidad Politécnica de Valencia, Valencia, septiembre del 2011.
- [7] Chin-Chung Wei, Ruei-Syuan Lai. “Kinematical analyses and transmission efficiency of a preloaded ball screw operating at high rotational speeds”. Elsevier. *Mechanism and machine theory* 46 (2011) pp. 890-898
- [8] Directorios México (2012), Disponible [en línea]: <http://www.directoriosmexico.net/preparatorias-ciudad-juarez-chih/>
- [9] MSC. Bigbook. Disponible [En línea]: <http://www1.mscdirect.com>