Publicaciones Técnicas
pubtecia@astro.unam.mx

Instituto de Astronomía
UNAM

Reporte Técnico RT-2012-03

aceptación: 01-06-2012
recepción: 02-02-2012

Sistema de Inclinómetro Digital para Telescopios Astronómicos.

F. Lazo, B. Orozco, D. Hiriart, S. Zazueta, F. Murillo.
Instituto de Astronomía. Universidad Nacional Autónoma de México.
Km. 103 Carretera Tijuana-Ensenada, Ensenada, B. C., México.

RESUMEN:

Este trabajo presenta el diseño y construcción de un sistema para medir la inclinación y la aceleración angular en dos ejes ortogonales, además del ángulo de giro en un eje perpendicular a estos.

El sistema desarrollado se aplicó a determinar la inclinación y la aceleración de los ejes de ascensión recta y declinación de telescopios astronómicos con montura ecuatorial.

Contenido

1. INTRODUCCIÓN ---- 2
2. OBJETIVOS ---- 2
3. DESCRIPCIÓN DEL SISTEMA ---- 2
3.1 INCLINÓMETRO ADIS16209 ---- 3
3.1.1 TEORÍA DE OPERACIÓN DEL ADIS16209 ---- 3
3.2 MICROCONTROLADOR RCM3700 ---- 5
4. INTERFAZ GRÁFICA ---- 6
5. PROGRAMA DE ADQUISICIÓN DE DATOS ---- 7
6. PRUEBAS Y RESULTADOS ---- 8
7. CONCLUSIONES ---- 11
8. REFERENCIAS ---- 11
9. PROVEEDORES ---- 11
APÉNDICE A. DIAGRAMA ELÉCTRICO DEL MEDIDOR DE POSICIÓN DIGITAL ---- 12
APÉNDICE B. PROTOCOLO DE COMUNICACIÓN ---- 13
APÉNDICE C. PROGRAMA DE LA INTERFAZ GRÁFICA IMPLEMENTADO EN LENGUAJE PYTHON ---- 14
APÉNDICE D. PROGRAMA QUE LEE EL INCLINÓMETRO Y MANDA LOS DATOS POR LA RED ---- 17

1. INTRODUCCIÓN

El conocimiento de la posición angular de los ejes de declinación y ascensión recta es muy importante en la operación de un telescopio astronómico para su apuntado y en los sistemas de seguridad. Un sistema auxiliar que permita conocer estos ángulos en el sistema local de referencia es de gran utilidad para inicializar la posición de referencia del telescopio antes de utilizar el más preciso sistema de codificación angular del sistema de control del telescopio.

Este trabajo presenta el diseño y construcción de un sistema para medir la inclinación y la aceleración angular en dos ejes ortogonales, además del ángulo de giro en un eje perpendicular a estos. El sistema desarrollado se aplicó a determinar la inclinación y la aceleración de los ejes de ascensión recta y declinación de telescopios astronómicos con montura ecuatorial.

Además del uso astronómico presentado en este documento, el inclinómetro digital tiene otras aplicaciones generales que incluyen: la estabilización, alineación y control de plataformas ópticas; el sensado de inclinación y como medidor de nivel; mediciones de posición y movimiento angulares en general; como dispositivo monitor o de alarma en aplicaciones de seguridad y medicina; en sistemas de navegación que no requieran precisiones mayores a las que puede entregar el instrumento.

Este documento presenta la información técnica del equipo utilizado, los programas para la adquisición de datos del inclinómetro-acelerómetro y la interfaz gráfica del usuario. El sistema es de relativo bajo costo comparado con los equipos comerciales similares y de gran versatilidad pues se comunica con otros sistemas a través de la red Ethernet.

2. OBJETIVOS

- Diseñar y construir un inclinómetro digital para los ejes de ascensión recta y declinación de telescopios astronómicos con montura ecuatorial.
- Que la comunicación de este sistema sea a través de Ethernet para que pueda ser utilizado por los sistemas computarizados de control de telescopios ya existentes.
- Que la información pueda ser desplegada en una interfaz gráfica de usuario para facilitar su consulta y uso.

3. DESCRIPCIÓN DEL SISTEMA

La *Figura 1* muestra un diagrama a cuadros del equipo de medición implementado para cumplir los objetivos propuestos. El inclinómetro es leído por una computadora dedicada que además sirve de servidor de datos para ser consultado, vía Ethernet, por una computadora externa que contenga la interfaz gráfica de usuario.

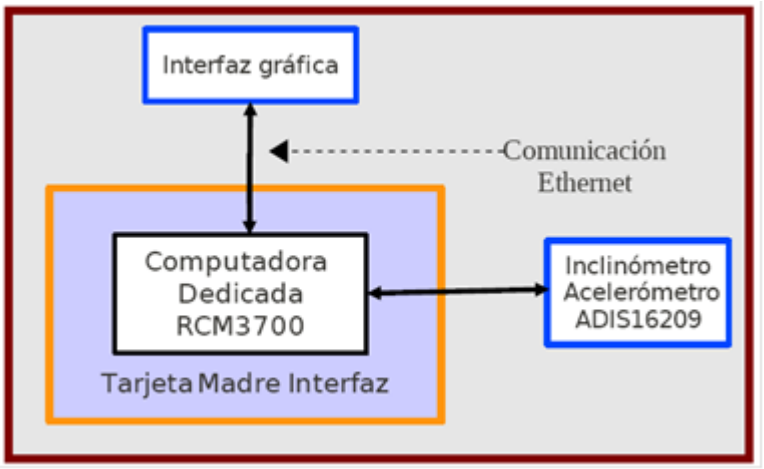


Figura 1: *Diagrama a cuadros del equipo de medición.*

3.1 INCLINÓMETRO ADIS16209

Para la medición de los ángulos se seleccionó un inclinómetro y acelerómetro modelo ADIS 16209 de la empresa *Analog Devices* el cual provee la medición en dos ejes ortogonales (x, y) con respecto a la horizontal, con un intervalo de medición de ±90°, con precisión de 0.1°, resolución de 0.025°; medición de rotación en un solo eje de ± 180° y una sensibilidad en la aceleración de 0.244e-3 g en los dos ejes (ver *Figura 2*) [1].

Figura 2: *Inclinómetro y acelerómetro ADIS 16209, foto Analog Device [1].*

3.1.1 TEORÍA DE OPERACIÓN DEL ADIS16209

El inclinómetro digital ADIS16209 utiliza la gravedad como su único estímulo y un acelerómetro construido en un Sistema Micro-Electromecánico (MEMS por sus siglas en inglés de "*Micro-Electro-Mecánica Systems*") como su elemento sensor. Los MEMS, es una tecnología que en su forma más general se puede definir como elementos mecánicos y electromecánicos miniaturizados; es decir, dispositivos y estructuras hechos utilizando las técnicas de micro-fabricación. Las dimensiones físicas críticas de los dispositivos MEMS pueden variar desde por debajo de una micra en el extremo inferior del espectro bidimensional hasta varios milímetros.

Del mismo modo, los tipos de dispositivos MEMS pueden variar de estructuras relativamente simples sin elementos móviles hasta sistemas electromecánicos extremadamente complejos, con múltiples elementos móviles bajo el control de microelectrónica integrada.

Los acelerómetros MEMS típicamente emplean un pequeño resorte pre-cargado que se entrelaza con una estructura digital en forma de peine (Ver *Figura 3*). La constante del resorte de la estructura flotante determina el desplazamiento cuando se la somete a una fuerza. Esta estructura responde a las fuerzas dinámicas asociadas con la aceleración y a fuerzas estáticas, tales como la gravedad.

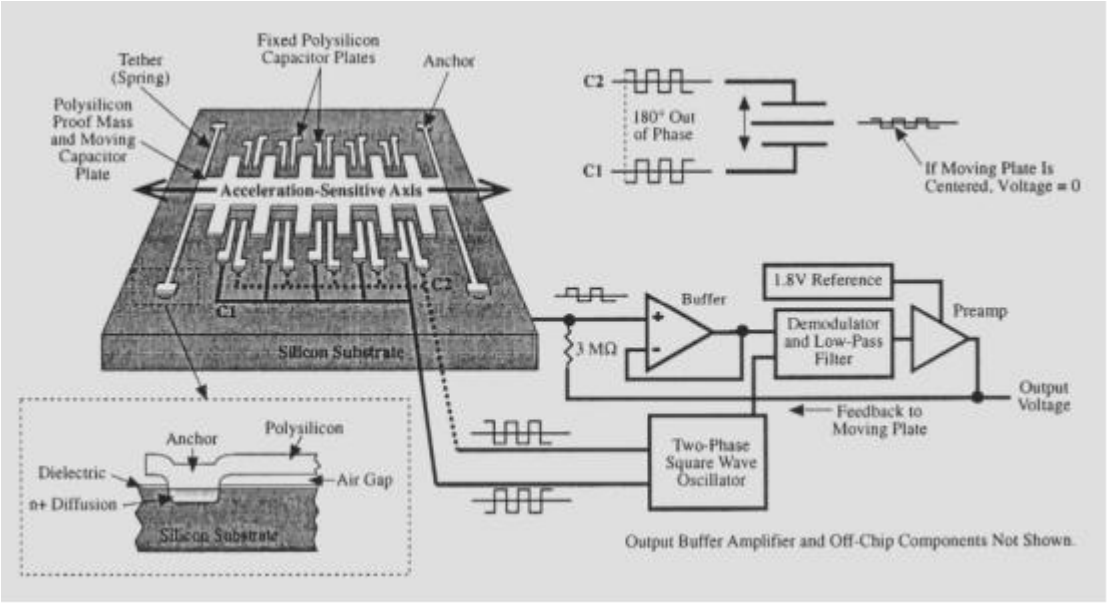


Figura 3: *Estructura MEM típica de un acelerómetro e inclinómetro. Acelerómetro de Analog Devices ADXL-50, (Tomada de "E.T.S. de Ingenieros de Telecomunicación de Vigo Departamento de Tecnología Electrónica") [3].*

Las *Figuras 4* y *5* muestran cómo el acelerómetro responde a la gravedad, según sea su orientación, con respecto al plano de la tierra. La *Figura 4* muestra la configuración de los ángulos de salida para inclinaciones en el plano del dispositivo.

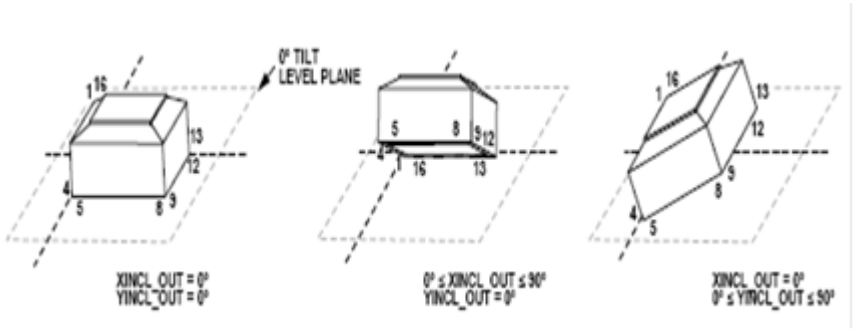
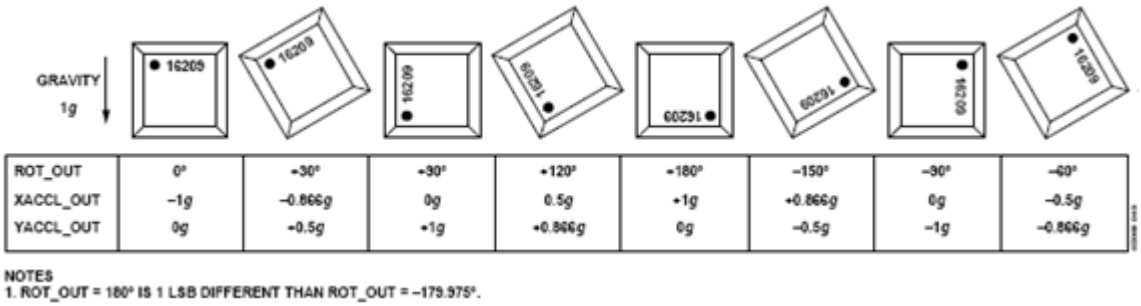


Figura 4: *Mediciones de la inclinación del plano del dispositivo que permiten una medición en el intervalo de ±90° en dos ejes ortogonales (X, Y) [1].*

La *Figura 5* muestra la configuración utilizada para la rotación de la posición angular. Esta configuración proporciona un mayor intervalo de medida de un solo eje con una mayor precisión en la medición angular.



ROT_OUT	0°	+30°	+90°	+120°	+180°	−150°	−90°	−60°
XACCL_OUT	−1g	−0.866g	0g	0.5g	+1g	+0.866g	0g	−0.5g
YACCL_OUT	0g	+0.5g	+1g	+0.866g	0g	−0.5g	−1g	−0.866g

NOTES
1. ROT_OUT = 180° IS 1 LSB DIFFERENT THAN ROT_OUT = −179.975°.

Figura 5: *Configuración para mediciones de la rotación en el intervalo de ±180°. Note la dirección de la gravedad [1].*

3.2 MICROCONTROLADOR RCM3700

En base a las especificaciones señaladas en el objetivo del trabajo y buscando en la literatura de instrumentación se optó por utilizar un micro controlador del tipo “RabbitCore RCM3700 C-Programmable Core Module with Ethernet” (ver *Figura 6*) [2]. Los programas en este microncontrolador son codificables en lenguaje de programación C.

Figura 6: *Microcontrolador RabbitCore RCM3700 (foto de hoja de datos Rabbit Core RCM3700) [2].*

Para poder intercambiar señales entre el microcontrolador y el inclinómetro se diseñó y construyó una tarjeta interfaz, (ver Apéndice A), donde se conecta el microcontrolador. La tarjeta interfaz cuenta además con reforzadores y acondicionadores de señal para la comunicación al exterior.

En la tarjeta madre interfaz se inserta el microcontrolador en un conector de 40 contactos, y el inclinómetro-acelerómetro, a través de cable plano.

Una de las consideraciones que se deben tomar en cuenta al instalar el sensor es que el eje de inclinación norte-sur coincida con el valor proporcionado en DEC de la interfaz de usuario (Interfaz Gráfica) y que el eje este-oste coincida con la posición indicada en AR de la interfaz (*Figura 7*).

4. INTERFAZ GRÁFICA

En la interfaz gráfica se muestran los valores de AR, DEC y de aceleración en los ejes de AR y DEC (ver *Figura 7*). La interfaz se construyó utilizando la aplicación *Glade* y el programa que adquiere y despliega la información en la interfaz gráfica del usuario está realizado en lenguaje *Python* (ver Apéndice C). El programa hace uso del método de hilos o hebras ("*threads*") para solicitar la posición y la aceleración en intervalos de tiempo de 200 milisegundos para posteriormente mostrar la información. La razón de emplear hilos para solicitar la información y no hacerlo en forma secuencial es para poder realizar otras funciones simultáneamente.

La función que solicita la información al microcontrolador se ejecuta cada 200 milisegundos independientemente de lo que esté realizando el programa. La cadencia de 200 milisegundos es arbitraria y puede ser modificada. La razón del valor seleccionado (200 milisegundos), es para no tener ocupada la línea de la red todo el tiempo.

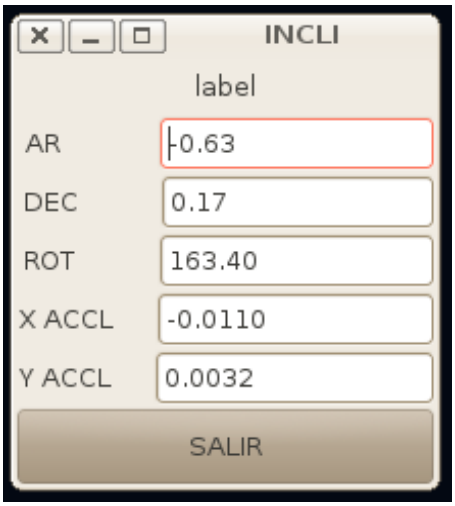


Figura 7: *Interfaz gráfica que muestra los valores de AR, DEC, rotación y las aceleraciones en los ejes de AR y DEC.*

5. PROGRAMA DE ADQUISICIÓN DE DATOS

El programa del microcontrolador que obtiene los datos del inclinómetro y acelerómetro está en un kernel multitareas (ver Apéndice D). Son tres las funciones que constantemente se están ejecutando:

La primera de ellas siempre está obteniendo la posición y aceleración del inclinómetro y actualizando la información en variables globales, disponibles para ser proporcionadas a la interfaz de usuario en el momento que la información sea solicitada. La información es proporcionada de la siguiente manera: posición AR, posición DEC, ROTACION, aceleración AR, aceleración DEC.

La segunda función se encarga de ver que el programa no se quede en un lazo infinito; si esto llega a suceder por cualquier situación, la función interrumpe al microntrolador para generar un ciclo de apagado y encendido del microcontrolador.

La tercera función se encarga de atender a la comunicación Ethernet para recibir los mandos y dar la respuesta a ellos (ver Apéndice B).

Para evitar que el control se quede en un lazo infinito, las rutinas se comunican entre sí a través de banderas. Al estar el programa en un ambiente multitareas, las rutinas son atendidas en una rebanada de tiempo de dos milisegundos aproximadamente. El procedimiento de atención a las rutinas es el siguiente: el sistema operativo va a la primer rutina, y de una tabla de control, toma el estado de donde ésta se había quedado, ejecuta instrucciones durante su rebanada de tiempo, posteriormente guarda su estado en la misma tabla y continúa con la siguiente rutina haciendo lo mismo que con la anterior. Este proceso se repite secuencialmente hasta volver a atender a la primera rutina.

Si la rutina que está atendiendo el sistema operativo no tiene nada qué hacer en ese momento, ésta cede su tiempo para que se ejecute la siguiente rutina. Esto ocasiona que las rutinas sean atendidas más rápidamente.

La función que obtiene los datos del inclinómetro-acelerómetro (las posiciones de AR, DEC, rotación, aceleración AR, aceleración DEC) se denomina *envia_datos* (arg) la cual hace una llamada a una función que escribe la dirección de donde va a tomar el dato, posteriormente llama a otra función que lee el dato de dirección indicada, la lectura/escritura es dígito a dígito del inclinómetro-acelerómetro, después se procesa el dato obtenido y éste se actualiza en su variable global correspondiente.

El siguiente ejemplo describe las rutinas antes mencionadas:

```
While (verdadero)
{
  Función: Watchdog (), si es verdadero da una inicialización a todo el sistema
  Función: Lee posición y aceleración del telescopio ()
  Función: Atiende la comunicación ()
}
```

6. PRUEBAS Y RESULTADOS

Para probar el sistema en laboratorio, se utilizaron inclinómetros analógicos para verificar la medición de los ángulos. Se probó también la capacidad de trabajo dejándolo funcionar de manera continua durante varios días y observando su comportamiento. El sistema se comportó de manera satisfactoria y trabajó de acuerdo a los requerimientos establecidos.

Para probar el sistema en campo, el sistema se instaló en el Radio Telescopio de 5m (RT5), localizado en Tonantzintla, Puebla. El RT5 tiene una montura mecánica de tipo ecuatorial. En colaboración con el grupo de instrumentación del Instituto Nacional de Astrofísica Óptica y Electrónica (INAOE), se incorporó a la interfaz de usuario del RT5 el despliegue de los valores de los ángulos de inclinación de los ejes de ascensión recta y declinación. También se elaboró una rutina para la detección automática del cenit (ver *Figura 8*). Además, el inclinómetro se utilizó para verificar que las variables de los ángulos de los ejes de ascensión recta y declinación del RT5 que guarda el sistema de control del telescopio, corresponden a los valores reales.
Ver Tablas 1 y 2.

TABLA 1

Valores proporcionados por el inclinómetro en Declinación.

ADIS 16209	Coordenadas en DEC	Goniómetro
0.025°	19° 02' 1.2"	0°
15.050°	34° 02' 1.2"	15°
30.075°	49° 02' 1.2	30°
45.100°	64° 02' 1.2"	45°
60.125°	79° 02' 1.2"	60°
-14.975°	4° 02' 1.2"	-15°
-29.950°	-11° 02' 1.2"	-30°
-44.925°	-26° 02' 1.2"	-45°
-59.900°	-41° 02' 1.2"	-60°
-74.875°	-56° 02' 1.2"	-75°

TABLA 2

Valores proporcionados por el inclinómetro en Ascensión Recta.

ADIS 16209	Coordenadas en AH	Goniómetro
0.025°	0h 00' 0.0"	0°
15.050°	1h 00' 0.0"	15°
30.075°	2h 00' 0.0"	30°
45.100°	3h 00' 0.0"	45°
60.125°	4h 00' 0.0"	60°
-14.975°	-1h 00' 0.0"	-15°
-29.950°	-2h 00' 0.0"	-30°
-44.925°	-3h 00' 0.0"	-45°
-59.900°	-4h 00' 0.0"	-60°

De las tablas de valores se deduce que el valor de incertidumbre es de ± 0.05° aproximadamente.

El sistema de medición funcionó correctamente y proporcionó al RT5 la capacidad de iniciar las observaciones con mayor rapidez. Anteriormente, la medición del cenit se realizaba por prueba y error mediante la colocación de niveles de gota en cada eje. La implementación en el

RT5 del sistema descrito aquí redujo el tiempo de detección del cenit de aproximadamente 20 minutos a menos de 1 minuto. Además, utilizando el sistema de niveles de gota, el error de posición era sumamente alto y al buscar con el telescopio un objeto estelar era muy difícil su localización. Ahora, con la ayuda del inclinómetro digital implementado, el error es muy pequeño y los objetos astronómicos se encuentran mucho más rápido.

El mismo tipo de inclinómetros digitales fueron instalados y probados en los telescopios de 84 cm y 1.5 m del OAN como medios auxiliares y complementarios para encontrar, de una manera automática, la posición al cenit de estos telescopios.

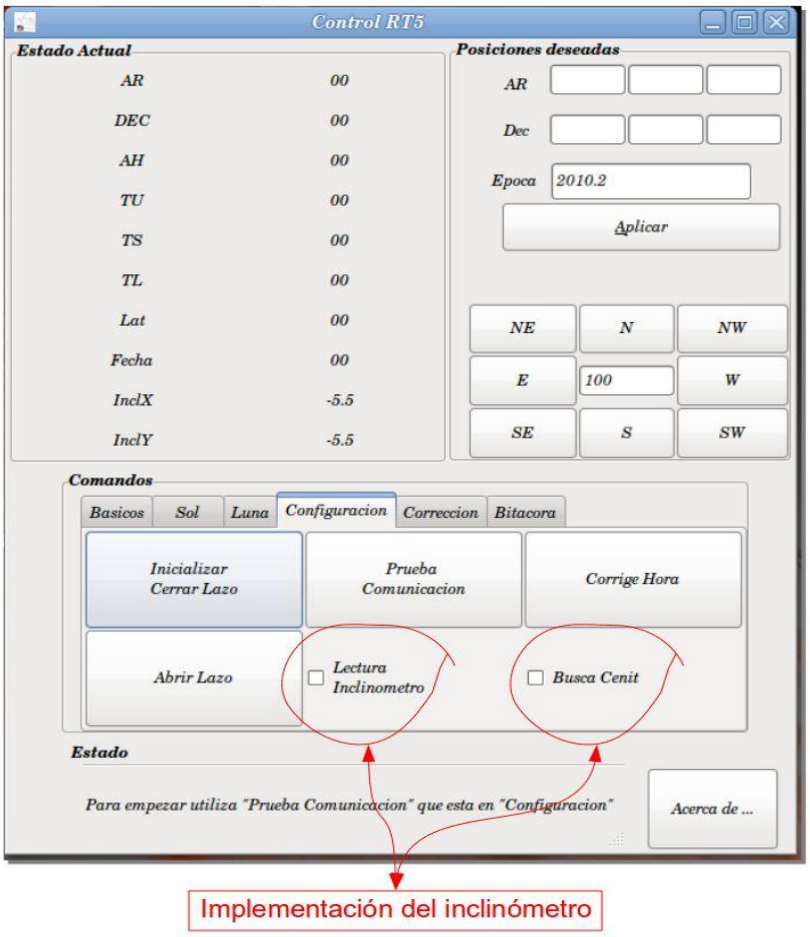


Figura 8: *Interfaz gráfica de usuario del radio telescopio RT5 donde se implementó la lectura del inclinómetro y la detección automática del cenit.*

7. CONCLUSIONES

El inclinómetro digital desarrollado es sencillo, confiable, robusto y de muy bajo costo. Este sistema puede ser de gran ayuda para coadyuvar con los sistemas de control y seguridad de telescopios astronómicos.

8. REFERENCIAS

[1] ADIS16209: High Accuracy, Dual-Axis Digital Inclinometer and Accelerometer http://www.analog.com/static/imported-files/data_sheets/ADIS16209.pdf.

[2] Rabbit Core RCM3700, C-Programmable Core Module with Ethernet User's Manual 019–0136 030910–A http://www.rabbit.com/redirect.jsp

[3] http://www.marcombo.com/Descargas/9788426715753/SENSORES/TEMAS/SA%20Tema%2012%20Microsensores.pdf

9. PROVEEDORES

Componentes electrónicas:
http://www.mouser.com/

Inclinómetro:
http://search.digikey.com/us/en/cat/programmers-development-systems/eval-boards-sensors/2622557?k=adis%2016209

Rabbit:
http://search.digikey.com/us/en/products/20-101-1305/316-1184-ND/2410703

APÉNDICE A. DIAGRAMA ELÉCTRICO DEL MEDIDOR DE POSICIÓN DIGITAL

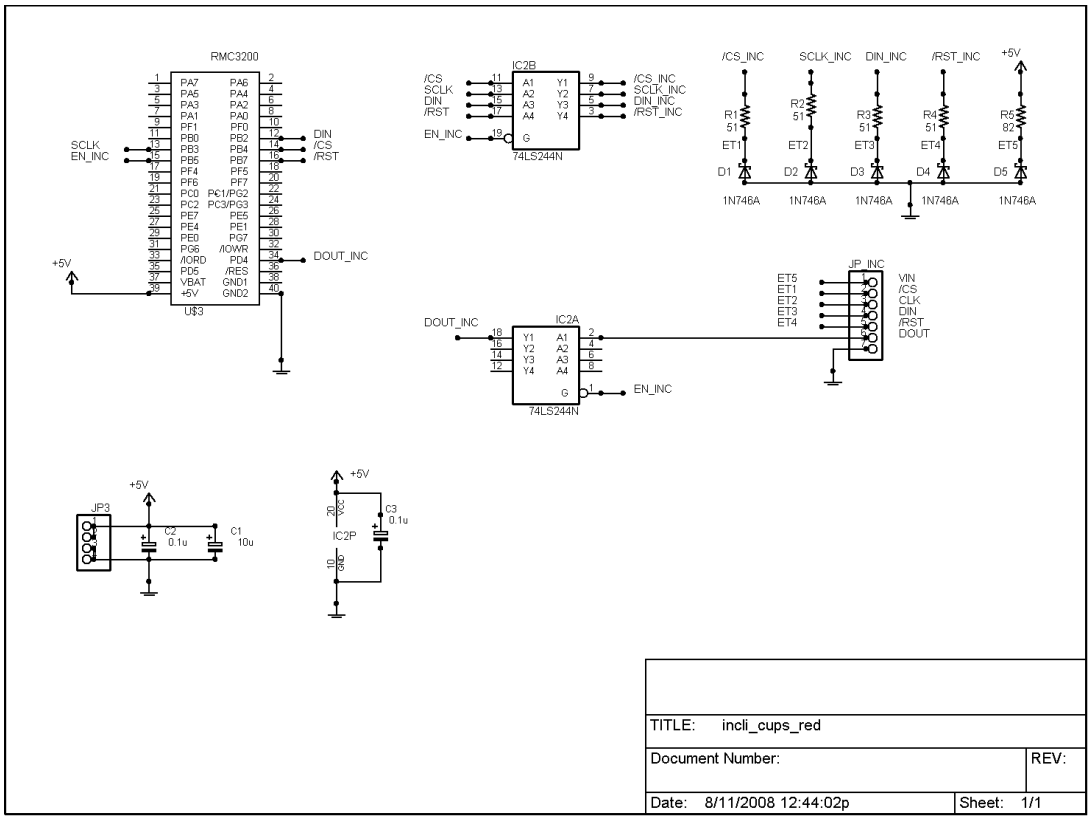


Figura A1: *Diagrama eléctrico del medidor de posición-aceleración*

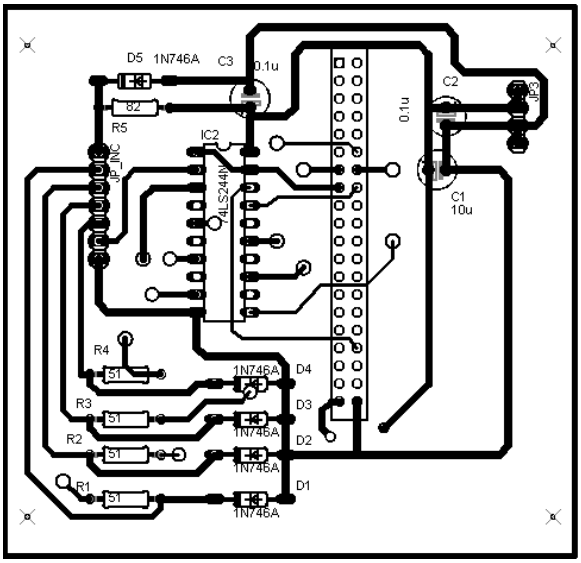


Figura A2: *Mascarilla PCB, lado soldadura*

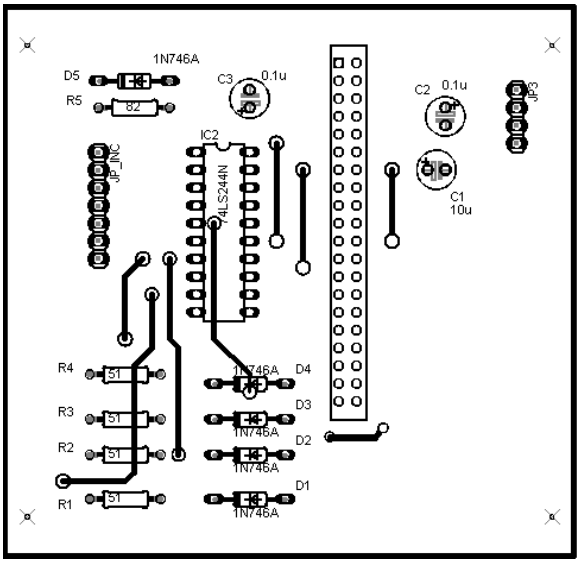


Figura A3: *Mascarilla PCB, lado componentes*

APÉNDICE B. PROTOCOLO DE COMUNICACIÓN

Comunicación Ethernet

Dirección del inclinómetro-acelerómetro: 192.168.3.199 & Puerto: 4545

Mando	Descripción
:P;	Proporciona la posición AR, DEC, ROTACION, XACCL, Y ACCL
:S;	Proporciona el voltaje del sensor y el promedio de lecturas (X^2)
:Z;	Pone en 8 el promedio de lecturas 8^2= 256 lecturas del sensor.
:XIN;	Solicita la posición en AR
:YIN;	Solicita la posición en DEC.
:ROT;	Solicita las poción de la rotación.
:XAC;	Solicita la aceleración en AR.
:YAC;	Solicita la aceleración en DEC.
:R;	Apaga y prende al microntrolador.

Ejemplo:

```
DESDE LINUX:      echo ":P;" |nc 192.168.3.109 4545
DESDE WINDOWS:    echo  :P;  |nc 192.168.3.109 4545
```

```
>echo :P; |nc 192.168.3.199 4545          // mando
>:-0.78 0.10 173.72 -0.0134 0.0015;       // Respuesta
```

APÉNDICE C. PROGRAMA DE LA INTERFAZ GRÁFICA IMPLEMENTADO EN LENGUAJE PYTHON

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-
import socket
import sys
from threading import Thread
import time
# Importamos el módulo pygtk y le indicamos que use la versión 2
import pygtk
pygtk.require("2.0")

# Luego importamos el módulo de gtk y el gtk.glade, este último que nos sirve
# para poder llamar/utilizar al archivo de glade
import gtk
import gtk.glade

adios = 0

def pide_pos() :
    global adios
    mi_MainWin = MainWin()
    while True :
        mi_MainWin.pide_pos()
        time.sleep(.2)
        if adios == 1 :
            break

#Creamos la clase de la ventana principal del programa
class MainWin( Thread ):
#class MainWin:

   def __init__(self):
        #self.widgets = gtk.glade.XML("GladeEjemplo.glade")
        #self.widgets = gtk.glade.XML("primer_glade.glade")
        self.widgets = gtk.glade.XML("incli_14_v2.glade")

        signals = {
                    "on_button1_salir_clicked"   : self.salir,
                    "on_window1_destroy"         : self.salir,
                  }

        self.widgets.signal_autoconnect(signals)

        self.entry1_AR = self.widgets.get_widget("entry1_AR")
        self.entry2_DEC = self.widgets.get_widget("entry2_DEC")
        self.entry3_ROT = self.widgets.get_widget("entry3_ROT")
        self.entry4_XACCL = self.widgets.get_widget("entry4_XACCL")
        self.entry5_YACCL= self.widgets.get_widget("entry5_YACCL")
```

```
    def pide_pos(self):
        s = socket.socket()
        s.connect(("192.168.3.199", 4545))
        mensaje = "\":P;\""
        s.send(mensaje)
        recibido =s.recv(1024)
        s.close()

        #time.sleep(.2)
        s = socket.socket()
        s.connect(("192.168.3.199", 4545))
        mensaje = "\":XIN;\""
        s.send(mensaje)
        recibido1 = s.recv(1024)
        s.close()

        #time.sleep(.2)
        s = socket.socket()
        s.connect(("192.168.3.199", 4545))
        mensaje = "\":YIN;\""
        s.send(mensaje)
        recibido2 = s.recv(1024)
        s.close()

        #time.sleep(.2)
        s = socket.socket()
        s.connect(("192.168.3.199", 4545))
        mensaje = "\":ROT;\""
        s.send(mensaje)
        recibido3 = s.recv(1024)
        s.close()

        #time.sleep(.2)
        s = socket.socket()
        s.connect(("192.168.3.199", 4545))
        mensaje = "\":XAC;\""
        s.send(mensaje)
        recibido4 = s.recv(1024)
        s.close()

        #time.sleep(.2)
        s = socket.socket()
        s.connect(("192.168.3.199", 4545))
        mensaje = "\":YAC;\""
        s.send(mensaje)
        recibido5 = s.recv(1024)
        s.close()

        dato = recibido1
        print dato
        self.entry1_AR.set_text ("%s" % dato)
```

```
        dato = recibido2
        print dato[1:5]
        self.entry2_DEC.set_text ("%s" % dato)

        dato = recibido3
        print dato
        self.entry3_ROT.set_text ("%s" % dato)

        dato = recibido4
        print dato
        self.entry4_XACCL.set_text ("%s" % dato)

        dato = recibido5
        print dato
        self.entry5_YACCL.set_text ("%s" % dato)

    def salir(self, widget):
            adios = 1
            gtk.main_quit()

def main():
        global adios
        gtk.gdk.threads_init()

        t = Thread(target=pide_pos)
        t.start()
        #MainWin()
        gtk.gdk.threads_enter()
        gtk.main()
        gtk.gdk.threads_leave()
        adios = 1


# Para terminar iniciamos el programa
if __name__ == "__main__":
    main()
```

APÉNDICE D. PROGRAMA QUE LEE EL INCLINÓMETRO Y MANDA LOS DATOS POR LA RED

```
#class auto
#define TCPCONFIG 1
#use "dcrtcp.lib"
#define PORT1 4545    // Standard Telnet port

#undef  SOCK_BUF_SIZE
#define SOCK_BUF_SIZE       2048

//C:\netcat>echo ":P;"   | nc 192.168.3.199 4545     direccion de este modulo

/*
 * We're only going to use to 1 sockets
 * Note: The default setting is 4 sockets
 */
#undef  MAX_SOCKETS
#define MAX_SOCKETS                 1

#define DIN             2       // bit PB2   SALIDA   (entrada del sensor)
#define SCLK    3                       // bit PB3   SALIDA
#define CS      4                       //          bit PB4   SALIDA
#define RST     7       // bit PB7   reset  del inclinometro
#define EN_INC   5        // bit PB5   Habilta Inclinometro
#define DOUT_INC 5                      // bit PD5   ENTRADA    (sale del Inclinometro)

#define SUPPLY_OUT          0x02
#define XACCL_OUT             0x04
#define YACCL_OUT             0x06
//#define AUX_ADC             0x08
#define TEMP_OUT    0x0A
#define XINCL_OUT   0x0C
//#define XINCL_SCALE  0x1C
//#define YINCL_SCALE  0x1E
#define YINCL_OUT   0x0E
#define AVG_CNT     0x38
#define ROT_OUT             0x10       // rotacion en grados +- 180

char buffer1[2048];
static tcp_Socket Socket_1;
static char mover;
static float   XINC, YINC, ROTACION, XACCL, YACCL, TEMP;

void
si_amo(void);

int
envia_datos(int direccion);

int
lee_incl(void);
void
```

```
escribe_incl(int DirData);

void
retardo_conv(int retardo);

nodebug
void msDelay(unsigned int delay);

void
inicializa(void);
void main()
{
  static int wd;              // ID for a virtual watchdog
      const static int runTime = 6;
      static unsigned long tmo;
      int dato;
  float VOLTAJE;
  inicializa();
  sock_init();
  tcp_reserveport(PORT1);

      while (1)
      {

     costate
     {
      tmo = SEC_TIMER;
      wd  = VdGetFreeWd(161); // wd activated, 9 virtual watchdogs now available
                              // wd must be hit at least every (33 - 1)/16 = 2 seconds
                              // (161-1)/16 =10 segundos
       //printf("\n This program will terminate normally with exit code 0 \n");
      //printf("in %d seconds if no virtual watchdog times out.", runTime);

      while(SEC_TIMER - tmo < runTime) { // let it run for a little while
// comment out VdHitWd(wd) to observe the fatal error message after 2 seconds of run time
      VdHitWd(wd);       // decrementing counter corresponding to wd reset to 33
         yield;
      }
       VdReleaseWd(wd);      // now all 10 virtual watchdogs are available
      //printf("\n YA TERMINE\n");
     }

     costate{
      msDelay(1000);
      yield;
//printf("\n X =%.2f Y=%.2f ROT=%.2f X_A=%.4f Y_A=%.4f",XINC,YINC,ROTACION,XACCL,YACCL);
       //printf("\n ROT=%.2f X=%.4f Y=%.4f",ROTACION,XACCL,YACCL);
       //printf("\n ROT=%.2f",ROTACION);
     }   */
```

```
    costate

{
      msDelay(1000);
     XINC =  0.025 * envia_datos(XINCL_OUT);
      printf( "\nXINC= %f",XINC);
      msDelay(1);
      yield;
      YINC =  0.025 * envia_datos(YINCL_OUT);
      printf( "\nYINC= %f",YINC);
      msDelay(1);
      yield;
      ROTACION =  0.025 * envia_datos(ROT_OUT);
      printf( "\nROT= %f",ROTACION);
      msDelay(1);
      yield;
      XACCL = 0.24414e-3  * envia_datos (XACCL_OUT);
      printf( "\nXACCL= %f",XACCL);
      msDelay(1);
      yield;
      YACCL = 0.24414e-3  * envia_datos (YACCL_OUT);
      printf( "\nYACCL= %f\n\n",YACCL);
      msDelay(1);
      yield;

    }

    costate
     {
      si_amo(); // Go do TCP/IP part of the application programM
    }
     } // while (1)
}   // main ()

void si_amo(void)
{
     char wr_buffer[128];
  char buf[25];
      int tcp_status;
      unsigned int len, i;
      static int connected;
  int dato, nc;
  float VOLTAJE;

  #GLOBAL_INIT {connected = FALSE;}

      costate
      {
        // Need to do continuously to handle the TCP processing on all sockets.
              tcp_tick(NULL);
      }

      costate
```

```
{
        // Setup the controller to listen on PORT1
        tcp_listen(&Socket_1,PORT1,0,0,NULL,0);
        sock_mode(&Socket_1,TCP_MODE_BINARY);
        /*
         * Use sock_dataready() in-conjunction with sock_established() to handle the case of
         * a socket being opened, written to, and closed between two sock_established checks.
         */
        waitfor(sock_established(&Socket_1));
        connected = TRUE;
        while (connected)
        {
// Using tcp_tick() because to detect when the socket is closed.
                waitfor((tcp_status = tcp_tick(&Socket_1)) == 0 ||  sock_dataready(&Socket_1) > 0);
                // Check if the socket is closed, if not get the data
                if (tcp_status)
                {
  len = sock_fastread(&Socket_1,buffer1,sizeof(buffer1)-1);
  if (len<=0 )continue;
                        buffer1[len] = '\0';

  //sock_fastwrite(&Socket_1,buffer1, strlen(buffer1));
  if(strstr (buffer1,":S;") != NULL){

    escribe_incl(AVG_CNT);
    dato=lee_incl();
    nc= dato & 0x0ff;
    printf("\n\n\t\t\t AVG_CNT = %x\n\n\n",nc);

    escribe_incl(SUPPLY_OUT);
    dato=lee_incl();
    dato= dato & 0x3fff;
    printf(" \n\n dato trae %d\n\n", dato);
    VOLTAJE= dato*0.30518e-3 ;
    printf("\n\n\t\t\t:%f;\n\n\n",VOLTAJE);

    sprintf(wr_buffer,":AVG_CNT = %x VOLTAJE = %.2f;\n",nc, VOLTAJE );
   sock_fastwrite(&Socket_1, wr_buffer, strlen(wr_buffer));
  }
  if(strstr (buffer1,":Z;") != NULL){  //pone el sensor AVG_CNT =8
        escribe_incl(0xB8);
        dato=lee_incl();
    dato= dato & 0x0ff;
    printf("\n\n\t\t\t:%x;\n\n\n",dato);
  }
  if(strstr (buffer1,":P;") != NULL){    //envia el dato solo una vez
  //printf("\n X =%.2f Y=%.2f ROT=%.2f X_A=%.4f Y_A=%.4f ",XINC,YINC,ROTACION,XACCL,YACCL);
    sprintf(wr_buffer,":%.2f %.2f %.2f %.4f %.4f;\n", XINC,YINC,ROTACION,XACCL,YACCL);
    sock_fastwrite(&Socket_1, wr_buffer, strlen(wr_buffer));
  }
  if(strstr (buffer1,":XIN;") != NULL){
        sprintf(wr_buffer,"%.2f", XINC);
                                sock_fastwrite(&Socket_1, wr_buffer, strlen(wr_buffer));
  }
```

```
        if(strstr (buffer1,":YIN;") != NULL){
          sprintf(wr_buffer,"%.2f", YINC);
                                    sock_fastwrite(&Socket_1, wr_buffer, strlen(wr_buffer));
        }
        if(strstr (buffer1,":ROT;") != NULL){
          sprintf(wr_buffer,"%.2f", ROTACION);
                                    sock_fastwrite(&Socket_1, wr_buffer, strlen(wr_buffer));
        }
        if(strstr (buffer1,":XAC;") != NULL){    //envia el dato solo una vez
          sprintf(wr_buffer,"%.4f", XACCL);
                                    sock_fastwrite(&Socket_1, wr_buffer, strlen(wr_buffer));
        }
        if(strstr (buffer1,":YAC;") != NULL){    //envia el dato solo una vez
          sprintf(wr_buffer,"%.4f", YACCL);
                                    sock_fastwrite(&Socket_1, wr_buffer, strlen(wr_buffer));
        }

        connected = FALSE;
                        }  // if tcp_status
        //connected = FALSE;
                sock_flush(&Socket_1);
        sock_close(&Socket_1);
        if(strstr (buffer1, ":R;")){ //Reset se queda en un lazo infinito
                getchar();              // y se da un reset virtual.
          getchar();
                }
                }                                   //while conectado
    }                       // costate
}                                       // fin de la funcion

void
inicializa(void)
{
  WrPortI(PDCR, &PDCRShadow, 0x00);      //clear all bits to pclk/2
  WrPortI(PDFR, &PDFRShadow, 0x00);      //clear all bits to normal function
  WrPortI(PDDCR, &PDDCRShadow, 0x00);    //clear all bits to drive high and low
  WrPortI(PDDR, &PDDRShadow, 0x11);      //set bits 4,0 high
  WrPortI(PDDDR, &PDDDRShadow, 0x11);    //set bits 4,0 to output, rest inputs
  //WrPortI(PDDDR, &PDDDRShadow, 0x01);    //set bits 4,0 to output, rest inputs

  WrPortI(PCFR, &PCFRShadow, PCFRShadow&0xEA);                    //clear bit 4,2,0 to normal function
                                  //bits 5,3,1 normally inputs
WrPortI(PCDR, &PCDRShadow, PCDRShadow|0x15);    //set bits 4,2,0 high
WrPortI(PBDR, &PBDRShadow, PBDRShadow|0xfd);//set all bits high, except bit 1
WrPortI(PBDDR,&PBDDRShadow,PBDDRShadow|0xfd);          //set all bits to output, except bit 1

  BitWrPortI (PBDR, &PBDRShadow, 1, RST);
}



int
envia_datos(int direccion)
{
```

```
  int dato, dato_temp;
  float valor;
      escribe_incl(direccion);
  msDelay(1);
  dato=lee_incl();
  dato = dato & 0x3fff;
  dato_temp= dato & 0x02000;
  if (dato_temp==0x2000){
     dato=~dato;
     dato=dato+1;
  }
  dato=dato & 0x1fff;
  if (dato_temp==0x2000)
      dato=-dato;
  return dato;
}

int
lee_incl (void)
{
      int temp, i,j ;
  int valor;
  char dato[2];

BitWrPortI (PBDR, &PBDRShadow, 1, SCLK);
  retardo_conv(100);
  BitWrPortI (PBDR, &PBDRShadow, 1, CS); // inicia ciclo lectura
  msDelay(1);
  BitWrPortI (PBDR, &PBDRShadow, 0, CS); // habilto para leer  INCLINOMETRO
  retardo_conv(10);
      //msDelay(1);
   j=15;
   valor = temp = 0;
   //printf("\n\n Dato inclinometro");
   for( i=0; i<=15; i++)
      {
      BitWrPortI (PBDR, &PBDRShadow, 0, SCLK);
      retardo_conv(10);                                    // espero dato estable
      temp =(BitRdPortI(PDDR, DOUT_INC));   // Toma dato del inclinometro
      //printf(" %d",temp);
      valor= valor | temp;
      valor = valor << 1;
      BitWrPortI (PBDR, &PBDRShadow, 1, SCLK);
      retardo_conv(10);
     }
     valor = valor >>1;
     retardo_conv(40);
     BitWrPortI (PBDR, &PBDRShadow, 1, CS); // listo para iniciar de nuevo
     return valor;
}
void
escribe_incl(int DirData)
{
      int i, j, temp;
```

```
  char dato[2];

  dato[1]= DirData;
  dato[0]=0x08;
     BitWrPortI (PBDR, &PBDRShadow, 1, SCLK);
  retardo_conv(100);
  BitWrPortI (PBDR, &PBDRShadow, 1, CS); // inicia ciclo lectura
  msDelay(1);
  BitWrPortI (PBDR, &PBDRShadow, 0, CS); // habilto para leer  INCLINOMETRO
  retardo_conv(50);
  //msDelay(1);
  //printf("\n\n inicio\n\n");
  j=15;
  for( i=0; i<=15; i++)
      {
      BitWrPortI (PBDR, &PBDRShadow, 0, SCLK);
      retardo_conv(10);                                    // espero dato estable
      temp= bit(dato,j);
      //printf("%d",temp);
      j--;
      BitWrPortI (PBDR, &PBDRShadow, temp, DIN); // pongo bit de direccion
      retardo_conv(10);
      BitWrPortI (PBDR, &PBDRShadow, 1, SCLK);
      retardo_conv(10);
     }
     retardo_conv(40);
     BitWrPortI (PBDR, &PBDRShadow, 1, CS); // listo para iniciar de nuevo
}

void
retardo_conv(int retardo)
{
      int j;
  for( j = 0; j < retardo; j++ )    // counting loop
  {
                #asm
      nop
      nop
      #endasm
      }
}

nodebug
void msDelay(unsigned int delay)
{
      auto unsigned long done_time;
      done_time = MS_TIMER + delay;
  while( (long) (MS_TIMER - done_time) < 0 );
}
```