Michael A. Covington
www.covingtoninnovations.com

This PDF file was prepared in 2005 from an article that originally appeared in *PC Techniques*, 1994, and is reproduced here by permission.
This software was written for DOS 5.0 and PROBABLY WILL NOT WORK under current versions of Windows. The author cannot provide any support or further information unless you want to employ him as a paid consultant.

Joystick-Port Interfacing

by Michael Covington

A joystick port (game port) can interface with more than just joysticks. Now that most PCs have joystick ports -- but comparatively few PC owners are serious game players -- it's time to put the neglected joystick port to use to monitor CPU temperature, line voltage, the weather, or a host of other real-world quantities. In this article I'll show you how. This article is a sequel to one I wrote for PC Tech Journal in 1985, and about which I'm still getting inquiries.

Inside the Joystick Port

IBM's design for a pair of joysticks contains four variable resistors and four switches ("fire buttons," Figure 1). Accordingly, one joystick port can measure four analog quantities and four switch closures. Note that the switches are returned to ground but the resistors are returned to +5V. Be

sure not to short +5V and ground together, unless you want to reboot the computer in a hurry, and possibly melt some wire in the process.

The switch-closure inputs are simply TTL inputs pulled up to +5V by 1000-ohm resistors. The analog inputs are more interesting; Figure 2 shows what's inside them. The crucial component is the capacitor, which normally cannot charge because a switching transistor is holding a short circuit across it. The computer takes a reading by removing the short, and then timing how long it takes the capacitor to charge to 3.3 volts.

This is the original 1981 IBM PC joystick port circuit, and all subsequent PC joystick ports have worked the same way, with two exceptions. First, a few of the cheapest ports only support one joystick. Second, many newer ports have a "sensitivity adjustment" that sets the charging threshold to values other than 3.3V. The adjustment is usually a switch or knob, but on some Kraft joysticks, it is a digital-to-analog converter under software control.

Reading the Sensors

Although the hardware is well-standardized, the software interface definitely isn't. The most reliable way to read the analog inputs is through Microsoft BASIC (i.e., BASIC.COM, BASICA.COM, QBASIC, or QuickBasic; not Visual Basic). Microsoft has done their best to get the STICK() function to return an integer close to the actual resistance in kilohms, regardless of the CPU speed, BIOS version, or anything else.

Access is simple: in BASIC, a statement such as

```
X = STICK(0)
```

takes readings from all four sensors, and stores the first one into **X**. Then **STICK(1)**, **STICK(2)**, and **STICK(3)** retrieve the values that were read from the other three sensors. Listing 1 demonstrates this technique.

Another way to read the sensors is through the BIOS, using **INT 15h** with AH = 84h and DX = 1. The four sensor values come back in AX, BX, CX, and DX respectively. Listing 4 shows how to do this in Turbo Pascal. While experimenting, you can also use Microsoft Diagnostics (MSD.EXE) to read the analog sensors through the BIOS.

The problem is, there's a lot of variation among BIOSes. This variation affects only the scale of the numbers: a 100-kilohm resistance may give a reading of 40 on one machine and 180 on another. The original PC, XT, and PCjr do not have BIOS joystick support at all.

The third way to read the joystick port is, of course, to manipulate the hardware itself -- in effect, implement for yourself what BASIC and the BIOS are trying to do. I haven't tried this. To do it, output any value (it doesn't matter what) to port 201h, then repeatedly read that same port, timing how long it takes each of the lower 4 bits to switch from 1 back to 0. For a range of 0 to 200 kilohms, the time will range from 24 to 2200 microseconds. For high accuracy, use a hardware timer.

Reproducibility

The laws of physics ensure that the time taken to charge the capacitor will always be proportional to the resistance (including the internal 2.2k resistor). Thus, this simple circuit gives excellent linearity -- just what game players need.

The problem is with reproducibility from machine to machine and port to port. The values of the capacitors can vary as much as 10%, so you won't get exactly the same numbers from different game cards, or even different analog inputs on the same card.

More importantly, precision timing is never easy on the PC, and both BASIC and the BIOS routines are affected by minor changes in the software environment. In particular, a program running in a DOS box under Windows will not give the same readings as when running under DOS by itself, because of differences in timing.

Fortunately, you can compensate for these variations in timing -- or at least detect them -- by putting known resistances across some of the analog inputs, and comparing them to the unknown value, rather than trying to take absolute readings. That's also a good way to deal with BIOSes that use different numeric scales for the same range of resistances.

Industry uses precision analog-to-digital converters -- not joystick ports -- for serious measurement work. But there are plenty of applications for which the limited capabilities of the joystick port are quite adequate.

Measuring Resistance and Voltage

Listing 1 shows a program for measuring resistance. Essentially, it solves the equation

Resistance = (Reading x Factor) + Offset

where **Factor** and **Offset** are unknown until calibration is performed for a particular game card and software environment. Naturally, you can do the calibration once, then save the values for future use on the same machine. The resistance to be measured is connected in place of one of the variable resistors in Figure 1.

The joystick port can also measure voltage, by charging the capacitor from the unknown voltage through a known resistance. The charging time is inversely proportional to the voltage, and the algorithms for making measurements are shown in Listing 2.

If the voltage to be measured is between 5 and 15 volts, the only external circuitry needed is a single resistor (Figure 3). For voltages outside this range, start with the circuit in Figure 4, which covers 0 to +5 volts with 0.1-volt resolution, and divide the voltage down as necessary at the input. Notice that this circuit requires a separate 9-volt battery for the op-amp. If the battery is inconvenient, you can steal +12V from inside the computer, or replace the LM324 op-amp with a Maxim MAX-418 or Texas Instruments TLC1079IN, which draws so little current that the battery will last for months. Do not substitute other op-amps (TL084, 741, etc.) whose input common-mode range does not include V-.

Measuring Temperature

Temperature is especially easy to measure with the joystick port. The trick is to use a negative-temperature-coefficient (NTC) thermistor in series with a resistor whose resistance is about 70% of that of the thermistor at the temperature where best performance is needed. A good combination is a 100-kilohm thermistor with a 68k resistor (Figure 5),

which gives 1°F resolution over a range of several tens of degrees. With considerable loss of resolution, you can use Radio Shack's 10-kilohm thermistor with a 4.7k resistor (allowing for 2.2k already inside the joystick port).

The resistor "linearizes" the thermistor so that 1/R is close to linearly proportional to temperature. You can then use the algorithm in Listing 3 to calibrate for Fahrenheit, Celsius, or Kelvin temperature. (Or even Rankine or Réaumur, if your tastes run toward the exotic.)

To calibrate the thermistor, bind it tightly to the bulb of an accurate thermometer by wrapping with aluminum foil; then apply an outer wrap of waterproof plastic, and immerse the whole assembly in hot and cold water.

What's this good for? Figure 6 shows one example: a graph of the air temperature in my computer room, measured once per minute by a program similar to Listing 2, then graphed with Quattro and post-edited with Corel Draw. This room has a rather large oscillation in temperature as the thermostat clicks on and off. At 7:00, the heat in another part of the house was turned on, changing the waveform but not the amplitude.

Another obvious use for a thermistor is to monitor the temperature of an 80486 or Pentium CPU. In this case, calibration may not be necessary; all you need is a relative reading, and a warning if the CPU suddenly gets hotter than normal.

Other measurements

The joystick port can measure anything that translates into resistance, voltage, or current -- and that means practically anything at all.

Figure 7 shows how to measure light level with a CdS photocell; the parallel resistor keeps the overall resistance in the measurable range even though the photocell, by itself, would have a resistance of several megohms in the dark. Calibration is up to you. Humidity sensors, barometric pressure sensors, and even potentiometers with weights (to measure position or acceleration) are other possibilities.

References

Covington, Michael A. "Joystick Metrics: Measuring Physical Properties through the PC's Joystick Port." PC Tech Journal, May, 1985, pp. 100-109.

Technical Reference. IBM Personal Computer Hardware Reference Library. IBM, 1983.

Parts suppliers

Most of the parts mentioned in this article are available at Radio Shack, which can special-order hard-to-find items. Thermistors are available from Digi-Key (1-800-344-4539, 701 Brooks Avenue South, Thief River Falls, MN 56701), which will probably also stock the MAX-418 IC in the future.

(Sidebar)

Joysticks under Windows

There are two ways to read the joystick port under Windows. The low road is to use the BIOS, as shown in Listing 4. This is more complex than simply issuing an INT instruction in the middle of a Windows program, but Turbo Pascal and Turbo C++ provide routines that do the housekeeping for you (**intr** and **int86** respectively).

The high road is to implement a device driver for the joystick port, supporting the joystick API documented in the Windows Multimedia Reference. A sample is included with the Microsoft Windows Device Driver Kit and has also been distributed separately as IBMJOY.ZIP. As supplied, this driver isn't quite suitable, because it recalibrates itself periodically under the assumption that a real joystick is attached. Still, it could easily be modified to read resistance directly, storing machine-specific calibration information in WIN.INI.

(Sidebar)

Still More Uses for a Joystick Port

Besides what's already been mentioned, you could use a joystick port to...

• detect switch closures, tracking the state of a thermostat or burglar alarm, the movement of a model train, or the activities of a trained hamster;

• accept 4 bits of parallel TTL-level data directly, through the switch-closure inputs;

• use the **ON STRIG(n)** statement in BASIC to make switch closures trigger event-driven routines;

• steal 5-volt power from the computer for some other accessory (after all, the joystick port brings out both +5V and ground).

LISTING 1

```
100 ' R.BAS  (M. Covington 1994)
110 ' Measuring RESISTANCE through joystick port
120 '
130 SENSOR = 0 ' possibilities: 0, 1, 2, 3
140 '
150 CLS
160 PRINT "RESISTANCE PROGRAM"
170 PRINT
180 '
190 ' Calibration
200 '
210 PRINT "Calibration needs 2 known values,"
220 PRINT "preferably near ends of range."
230 '
240 OPTION BASE 1
250 DIM RDG(2), VALUE(2)
260 FOR I = 1 TO 2
270   PRINT
280   PRINT "Connect known value "; I; " and press any key..."
290   WHILE INKEY$ = ""
300     RDG(I) = STICK(0)          ' initialize all 4 sensors
310     IF SENSOR > 0 THEN RDG(I) = STICK(SENSOR)
320     LOCATE CSRLIN, 1
330     PRINT "Reading: "; RDG(I); "      ";
340   WEND
350   PRINT
360   INPUT "Actual value"; VALUE(I)
370 NEXT I
380 '
390 FACTOR = (VALUE(2) - VALUE(1)) / (RDG(2) - RDG(1))
400 OFFSET = .5 * (VALUE(2)-FACTOR*RDG(2)+VALUE(1)-FACTOR*RDG(1))
410 '
420 PRINT
430 PRINT "Factor =", FACTOR, "Offset =", OFFSET
440 '
450 ' Taking readings
460 '
470 PRINT
480 PRINT "Taking readings continuously.  Exit with Ctrl-Break."
490 PRINT
500 '
510 WHILE 1
520   RDG = STICK(0)               ' initialize all 4 sensors
530   IF SENSOR > 0 THEN RDG = STICK(SENSOR)   ' take reading
540   VALUE = FACTOR * RDG + OFFSET
550   LOCATE CSRLIN, 1, 0
560   PRINT USING "#### ####.##"; RDG; VALUE;
570 WEND
```

LISTING 2

```
' Same as Listing 1 except for the following lines:
'
100 ' V.BAS  (M. Covington 1994)
110 ' Measuring VOLTAGE or CURRENT through joystick port
'
160 PRINT "VOLTAGE/CURRENT PROGRAM"
'
390 FACTOR = (1/VALUE(2) - 1/VALUE(1)) / (RDG(2) - RDG(1))
400 OFFSET = .5*(1/VALUE(2)-FACTOR*RDG(2)+1/VALUE(1)-FACTOR*RDG(1))
'
540   VALUE = 1 / (FACTOR * RDG + OFFSET)
```

LISTING 3

```
' Same as Listing 1 except for the following lines:
'
100 ' T.BAS  (M. Covington 1994)
110 ' Measuring TEMPERATURE with linearized thermistor
'
160 PRINT "TEMPERATURE PROGRAM"
'
390 FACTOR = (VALUE(2) - VALUE(1)) / (1/RDG(2) - 1/RDG(1))
400 OFFSET = .5*(VALUE(2)-FACTOR/RDG(2)+VALUE(1)-FACTOR/RDG(1))
'
540   VALUE = FACTOR / RDG + OFFSET
```

ADDENDUM:
A Win32 C program that *may* be useful. This is from my files from 1994, apparently ran under Windows 3.1, and I have no other information about it, or any recollection of whether it worked. – M. C., 2005.

```
/* Joystick access under Windows */
/* Requires IBMJOY.DRV installed */

#include <stdio.h>          /* for printf() */
#include <conio.h>          /* for kbhit()  */
#include <stdlib.h>         /* for exit()   */
#include <mmsystem.h>       /* for joystick */


UINT WINAPI joySetCalibration(UINT uJoyID /*,
      UINT a, UINT b, UINT c, UINT d, UINT e, UINT f*/ );


UINT    joycal[6] = {0x0ba2,0x0ba2,0x0a3d,0x0a3d,0,0};
      /* undoc! */


JOYCAPS cap;
JOYINFO joy;
UINT    status;

void joyerror(int errcode){
  /* Writes message for joystick error */
  /* and terminates the program        */
  if (errcode == JOYERR_NOERROR)
    puts("Nothing wrong");
  else if (errcode == MMSYSERR_NODRIVER)
    puts("No joystick driver installed");
  else if (errcode == JOYERR_PARMS)
    puts("No such joystick ID");
  else if (errcode == JOYERR_UNPLUGGED)
    puts("Joystick not plugged in");
  else if (errcode == JOYERR_NOCANDO)
    puts("Required resource already in use");
  exit(errcode);
}

main(){
  /* Determine capabilities */
  status = joyGetDevCaps(JOYSTICKID1,&cap,sizeof(cap));
  if (status != JOYERR_NOERROR) joyerror(status);
  printf("Joystick driver: %s\n",cap.szPname);
  printf("X range:         %u-%u\n",cap.wXmin,cap.wXmax);
  printf("Y range:         %u-%u\n",cap.wYmin,cap.wYmax);

  status = joySetCalibration(JOYSTICKID1 /*
,3000,3000,2000,2000,0,0 */);
  printf("Calibration set.  Status = %d\n");
/*
  status = joySetCalibration(JOYSTICKID2,3000,3000,2000,2000,0,0);
  printf("Calibration set.  Status = %d\n");
*/
  printf("Press Return...\n");
  getchar();

  /* Repeatedly show position */
```

```
  do {
    Yield();
    status = joyGetPos(JOYSTICKID1,&joy);
    if (status == JOYERR_NOERROR)
      printf("%u %u   ", joy.wXpos, joy.wYpos);
    else joyerror(status);
    status = joyGetPos(JOYSTICKID2,&joy);
    if (status == JOYERR_NOERROR)
      printf("%u %u\n", joy.wXpos, joy.wYpos);
    else joyerror(status);
  }
  while(status==JOYERR_NOERROR && !kbhit());
}
```

LISTING 4

```
PROGRAM read_joystick;
  { BIOS joystick access, Turbo Pascal }

{$IFDEF WINDOWS}
USES WinDos,WinCrt;
VAR  r: TRegisters;
{$ELSE}
USES Dos,Crt;
VAR  r: Registers;
{$ENDIF}

BEGIN
  clrscr;
  WHILE NOT keypressed DO
    BEGIN
      gotoxy(1,1);
      r.ax := $8400;
      r.dx := 1;
      intr($15,r);
      writeln(r.ax:5,r.bx:5,r.cx:5,r.dx:5)
    END
END.
```

(Biography)

Michael Covington does research on computational linguistics and manages the artificial intelligence lab at the University of Georgia. His other interests include electronics, ham radio, and astronomy. He is author of Natural Language Processing for Prolog Programmers (Prentice-Hall, 1994) and co-author of the Cambridge Eclipse Photography Guide (Cambridge University Press, 1994).

Figure 1

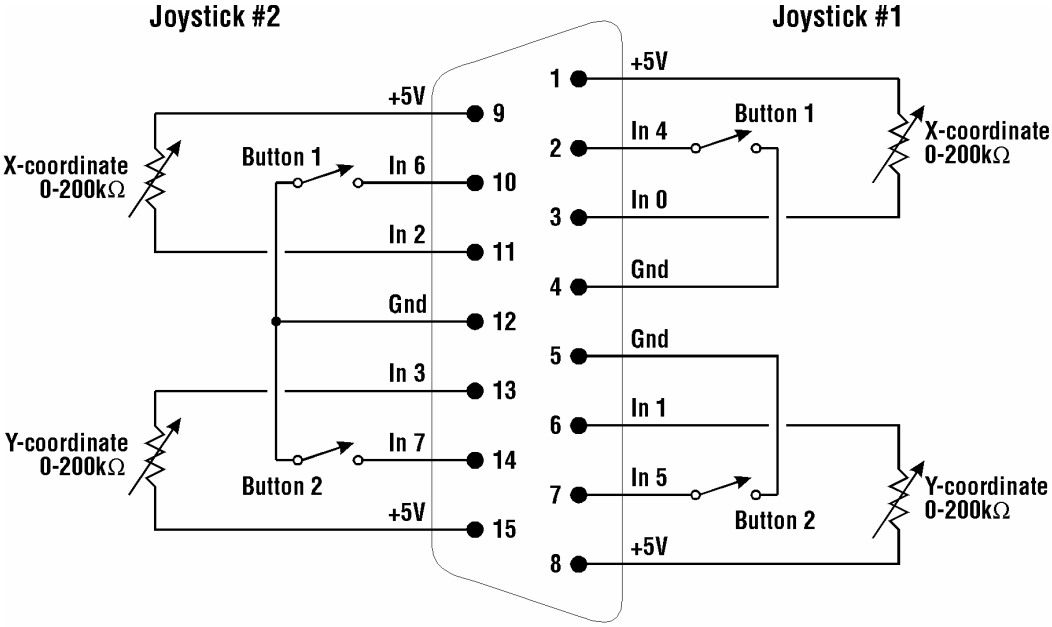


Figure 2

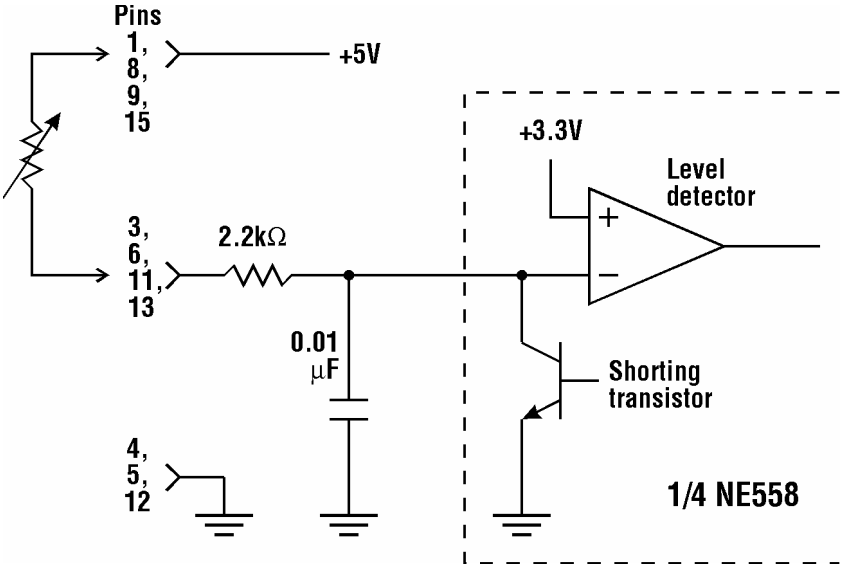


Figure 3

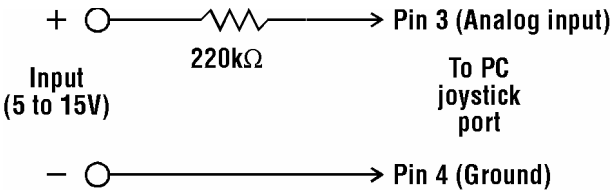


Figure 4

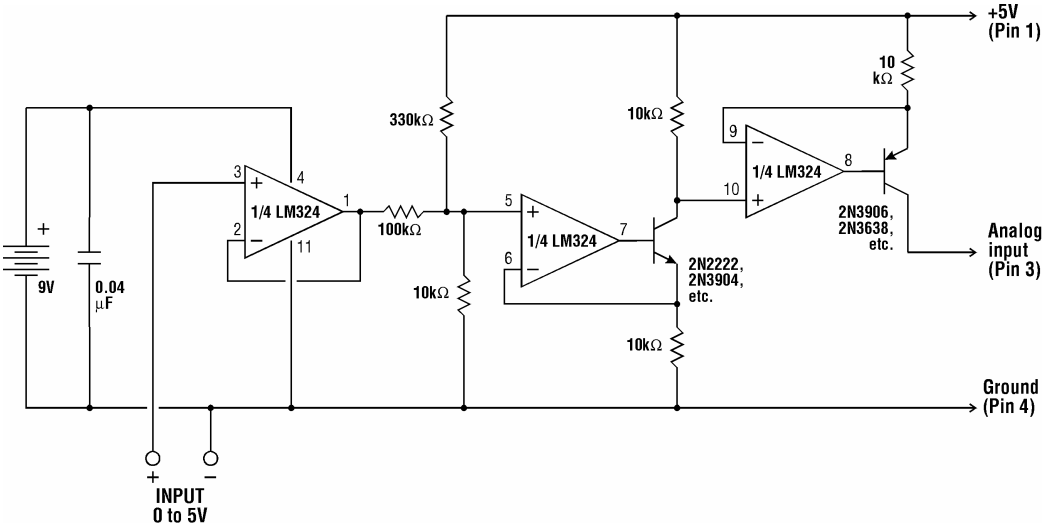


Figure 5

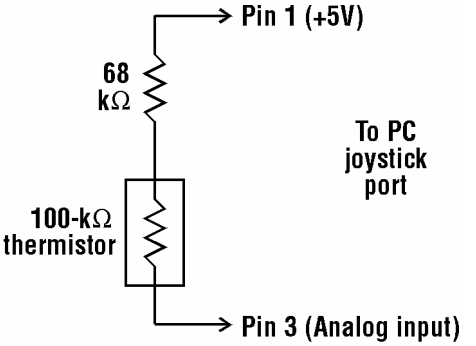


Figure 6

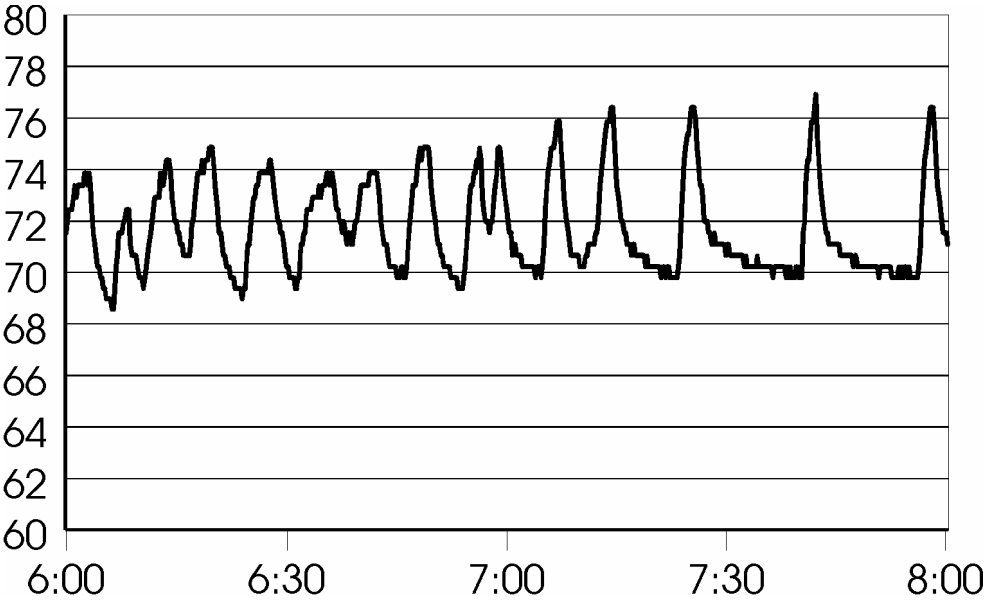
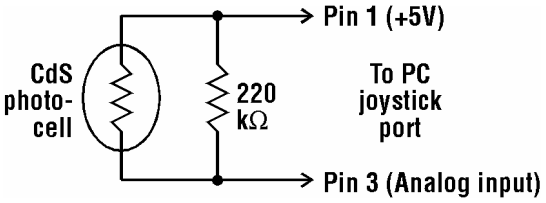


Figure 7


Pin 1 (+5V)
CdS photo-cell
220 kΩ
To PC joystick port
Pin 3 (Analog input)

-end-