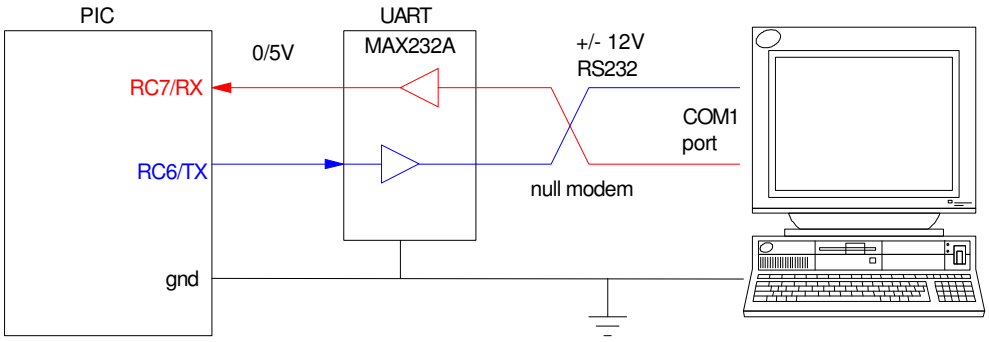
Data Collection & Calibration

Objective

- Design a circuit which outputs 0 .. 5V for a sensor (instrumentation amplifier)
- Determine a calibration function so that a PIC can determine temperature from the A/D reading

SPI Communications

Your PIC board communicates to the PC using SPI communications (serial peripheral interface). This for of communication uses a single wire to transmit data to the PC (TX) and a single wire to receive data (RX).

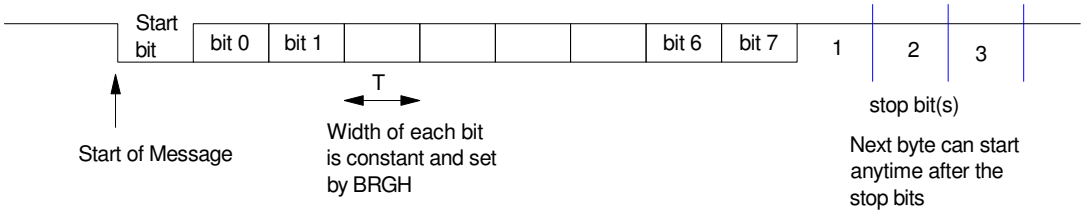


SPI communication with a PC (and a USB serial port)

This is how programs are download using a USB serial port. You can also use the serial port to send data back to the PC for data collection.

The way the data is encode is as follows:

- The data line idles high.
- When you want to send a byte, the data line goes low for one count.
- Eight data bits are then sent, least significant bit first.
- The data line then goes back to 5V until the next byte is to be sent.



Initializing the SPI Port: With SPI communication, timing is critical: the computer which sends the data and the computer which receives the data must know the time duration (T) of each bit. If these don't match, the data will look like gibberish.

The baud rate defines how fast data is sent out:

- 9600 baud means you send 9600 bits per second (or one bit is 1/9600 seconds long)

Other baud rates are supported as well:

Baud Rate	SPBRG	BRGH	BRG16	SYNC	Error (%)
2,400	255	0	1	0	-1.70%
4,800	129	0	1	0	-0.16%
9,600	255	1	1	0	-1.70%
19,200	129	1	1	0	-0.16%
38,400	64	1	1	0	-0.16%
57,600	42	1	1	0	-0.95%
115,200	21	1	1	0	+1.44%

Assuming you want to set up the serial port for 9600 baud, the following code will initialize the serial port:

```
// Turn on the serial port for 9600 baud

   TRISC = TRISC | 0xC0;
   TXIE = 0;
   RCIE = 0;
   BRGH = 1;
   BRG16 = 1;
   SYNC = 0;
   SPBRG = 255;
   TXSTA = 0x22;
   RCSTA = 0x90;
```

Once you initialize the serial port, you can send data to the PC (in ascii) using the following commands:

- Note: This routine is already in the file LCD_PORTD.c

```
void SCI_CRLF(void)
{
   while(!TRMT);       // wait until the serial port is free
   TXREG = 13;         // send a carriage return (ascii 13)
   while(!TRMT);       // wait until the serial port is free
   TXREG = 10;         // send a line feed (ascii 10)
   }
```

To send data to the serial port, the following subroutine can be used (similar to LCD_Write)

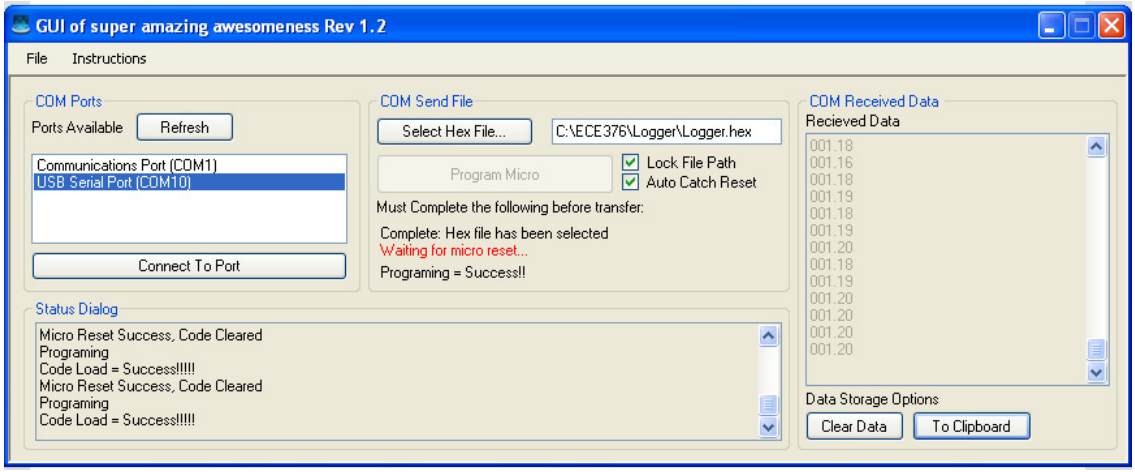
- Note: Also in file LCD_PORTD.c

```
void SCI_Out(long int DATA, unsigned char D, unsigned char N)
{
   unsigned char A[10], i;

   while(!TRMT);
   if(DATA < 0) {
      TXREG = '-';
      DATA = -DATA;
      }
   else TXREG = ' ';
   for (i=0; i<10; i++) {
      A[i] = DATA % 10;
      DATA = DATA / 10;
      }
   for (i=D; i>0; i--) {
      if (i == N) { while(!TRMT); TXREG = '.'; }
      while(!TRMT);  TXREG = A[i-1] + 48;
      }
}
```

With this code, you ca record the voltage every 100ms:

- Every 100ms, a new voltage appears on the COM Received Data window
- Clear Data to clear the data
- Once you have enough data, copy To Clipboard
- In Matlab, type in

```
DATA = [
    < paste data here (control V) >
   ]
```



Example 1: Six-Sided Die

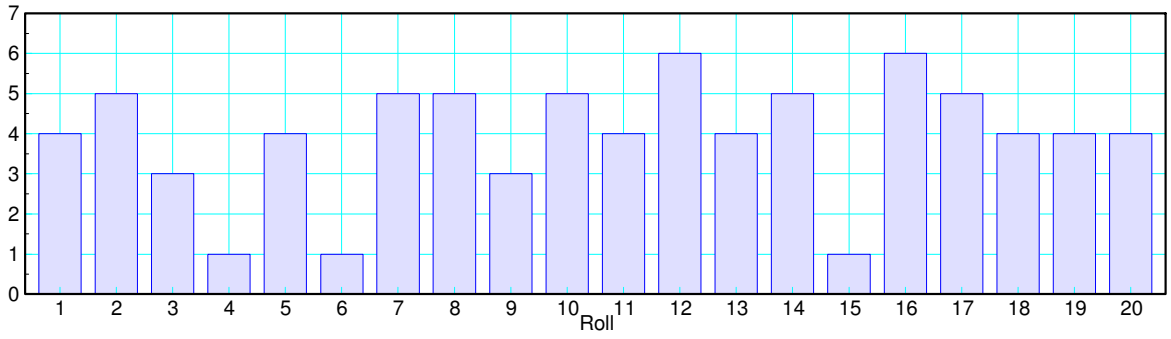
Generate random numbers from 1..6. Use the serial port to record the results of 100+ rolls.

Code (Main routine):

```
// Main Routine

void main(void)
{
   unsigned int DIE;
   unsigned int i, j;
   TRISB = 0xFF;
   ADCON1 = 0x0F;

// Turn on LCD display
   LCD_Init();

// Turn on the serial port for 9600 baud
   TRISC = TRISC | 0xC0;
   TXIE = 0;
   RCIE = 0;
   BRGH = 1;
   BRG16 = 1;
   SYNC = 0;
   SPBRG = 255;
   TXSTA = 0x22;
   RCSTA = 0x90;

   while(1) {
      while(!RB0);
      while(RB0) DIE = (DIE + 1) % 6;
      LCD_Move(1,0);  LCD_Write(DIE, 1, 0);
      SCI_Out(DIE, 4, 3);
      SCI_CRLF();
      }
   }
```

Result: After 129 rolls (129 presses of RB0), this is the number of times each number appears is:

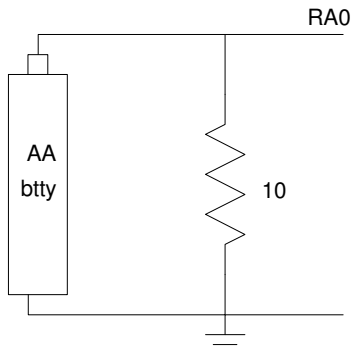
Number	1	2	3	4	5	6
Frequency	23	16	22	24	29	15



Example 2: Voltage of a AA Battery

Record the voltage of a AA battery as it discharges across a 10 Ohm resistor

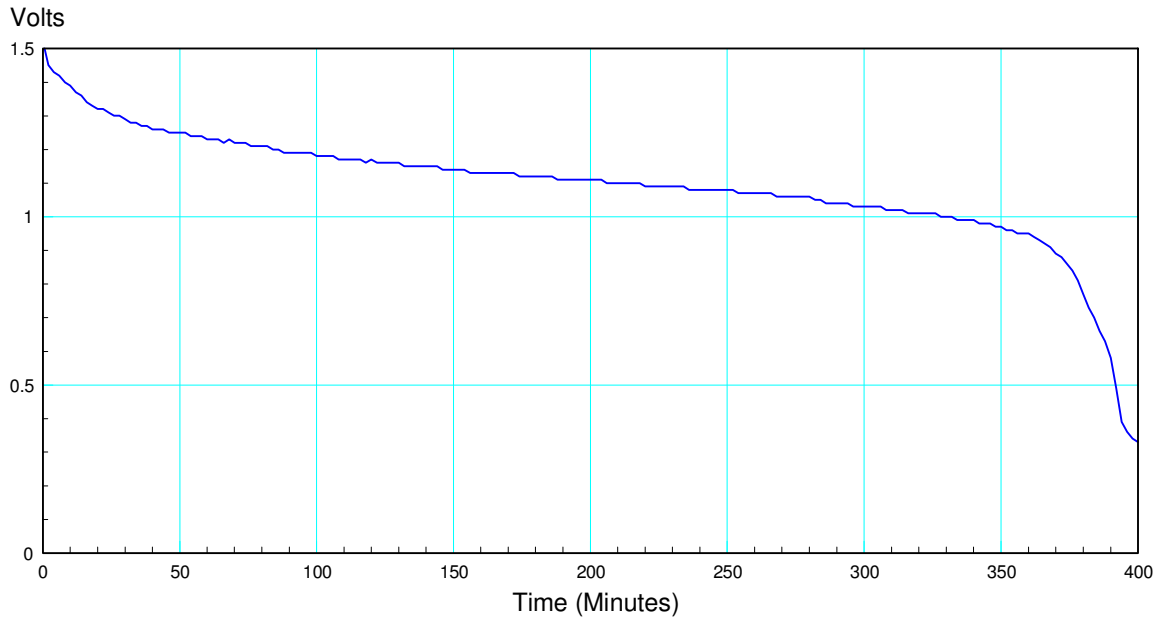
Hardware Setup:



Software: Previous Code to display the voltage every 120 seconds

Results:

minutes	volts
0	1.520
2	1.450
4	1.430
6	1.420

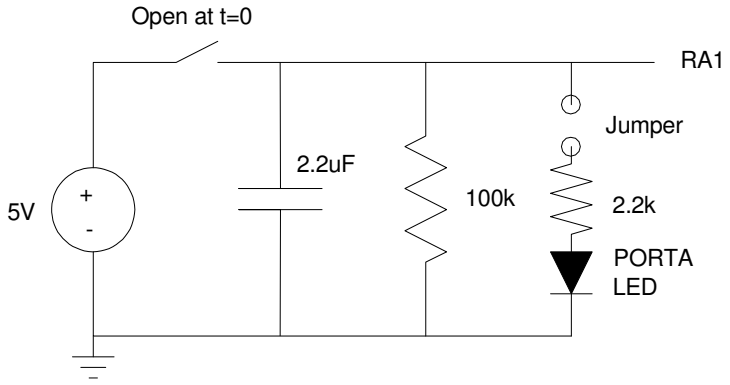


Example 3: Voltage Across a Capacitor as it Discharges

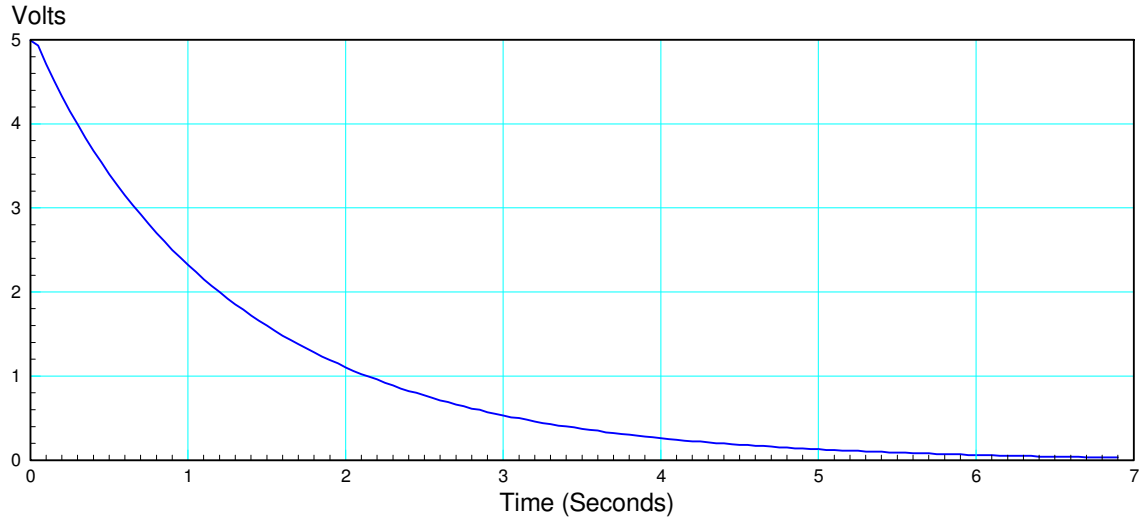
Charge a capacitor up to +5V

Measure the voltage as it discharges across a 100k resistor

- Remove the LED jumper on your PIC board for PORTA. This disconnects the 2.2k resistor and LED from PORTA.
- A 100k pull-down resistor is still connected
- Add a 2.2uF capacitor from RA1 to ground.
- Charge up to +5V
- Disconnect the +5V source and measure the voltage vs. time



Results: Collecting data every 50ms gives the following voltage vs. time



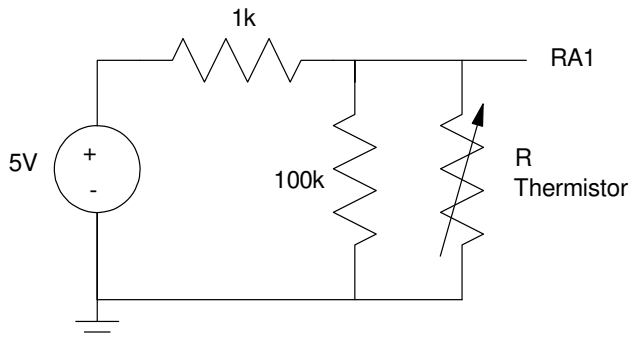
Example 4: Temperature of a Coffee Cup

The thermistor in your lab kit has a temperature vs. resistance relationship of

$$R = 1000 \exp\left(\frac{3905}{T+273} - \frac{3905}{298}\right) \Omega$$

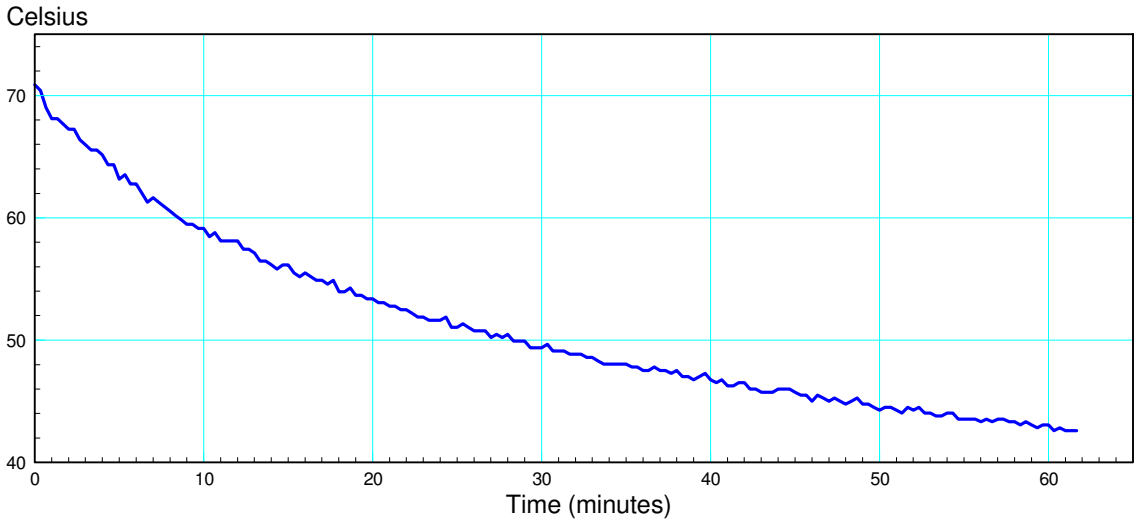
If you build a voltage divider with a 1k resistor and a thermistor,

- You can measure the voltage
- From this, you can compute the resistance of the thermistor (R),
- From this, you can compute the temperature.



Procedure:

- Add hot water to a coffee cup
- Measure the voltage every 10 seconds (from which you can compute the temperature every 10 seconds)
- Note that temperature behaves the same as the voltage across a capacitor for an RC circuit



Example 5: Temperature of a Pot of Water on the Stove

Procedure

- Place a quart of cold tap water in a pot of water on the stove.
- Set the stove temperature on high
- Record the temperature every 6 seconds

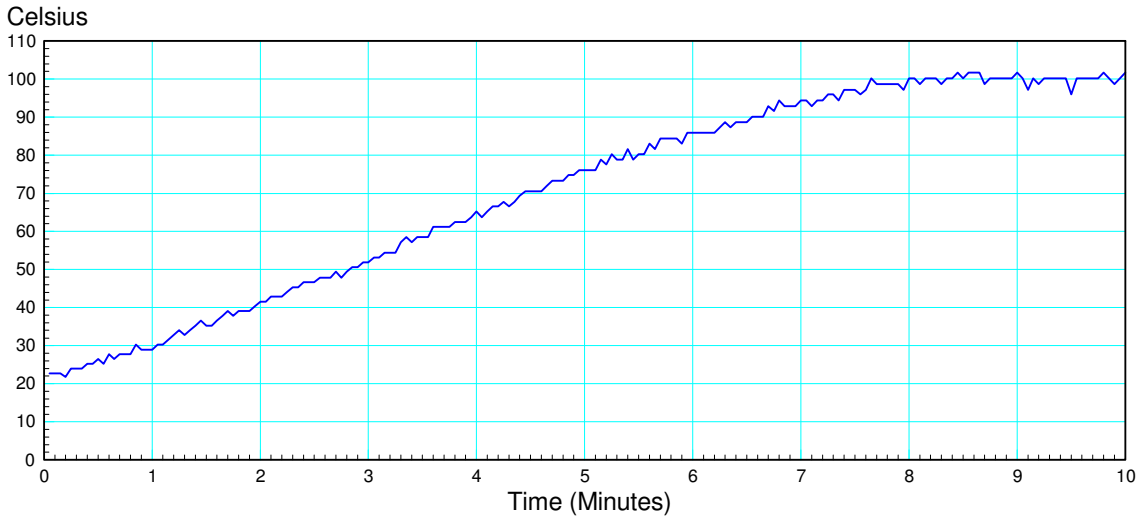
Result: Graph below

Discussion:

- You can see the water heating up over time
- The slope of the curve (0.19 degrees / second) tells you how many Watts the stove is putting out
  - Water has a specific heat of 4182 J/kg°C
  - 1 quart of water is 0.9464kg
  - 1 quart of water stores 3957J / C of thermal energy

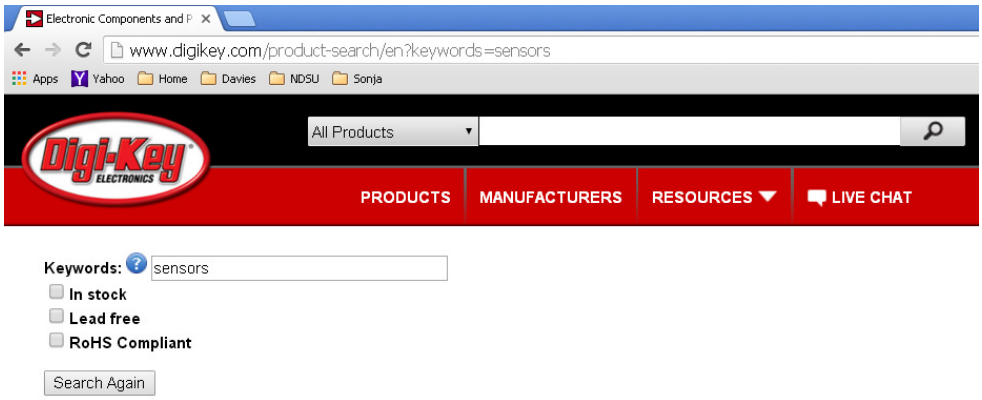
$$P = \left(0.19\frac{\text{degrees}}{\text{sec}}\right)\left(3957\frac{\text{Joules}}{\text{degree}}\right) = 751\ \text{Watts}$$

- The slope tapers off at the end. This is due to more heat loss through evaporation as the temperature goes up.
- The slope goes to zero at 100C. This is the boiling point of water.



Sensors

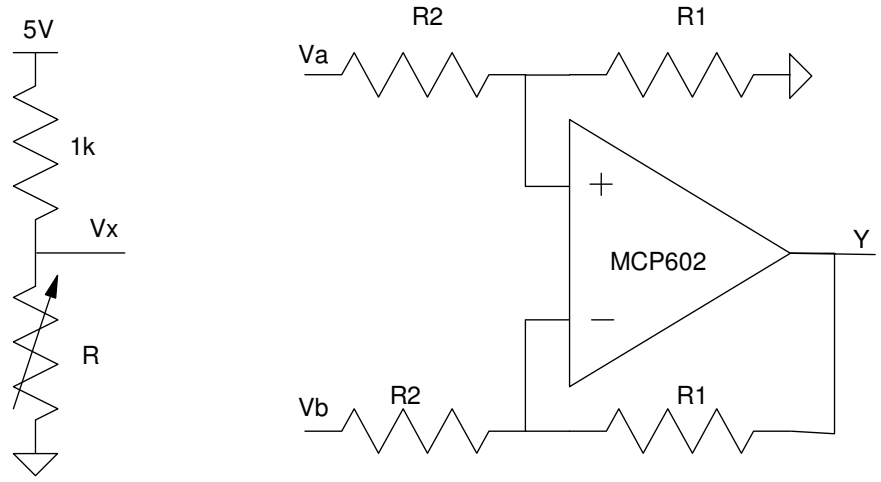
If you replace the thermistor in the previous circuit with a different sensor, a PIC can measure flow, flex, magnetic fields, etc. For example, Digikey sells over 19,000 different sensors:



Sensors, Transducers - 1017 New Products
Accelerometers (1135 items)
Accessories (3898 items)
Amplifiers (313 items)
Capacitive Touch Sensors, Proximity Sensor ICs (606 items)
Color Sensors (135 items)
Current Transducers (1503 items)
Dust Sensors (16 items)
Encoders (4364 items)
Flex Sensors (1 items)
Float, Level Sensors (622 items)
Flow Sensors (191 items)
Force Sensors (104 items)
Gas Sensors (78 items)
Gyroscopes (248 items)
Image Sensors, Camera (555 items)
Inclinometers (55 items)
IrDA Transceiver Modules (295 items)
LVDT Transducers (Linear Variable Differential Transformer) (8 items)
Magnetic Sensors - Compass, Magnetic Field (Modules) (24 items)
Magnetic Sensors - Hall Effect, Digital Switch, Linear, Compass (ICs) (3747 items)
Magnetic Sensors - Position, Proximity, Speed (Modules) (3288 items)
Magnets (145 items)
Moisture Sensors, Humidity (376 items)
Motion Sensors, Detectors (290 items)
Multifunction (147 items)
Optical Sensors - Ambient Light, IR, UV Sensors (736 items)
Optical Sensors - Distance Measuring (41 items)
Optical Sensors - Mouse (118 items)
Optical Sensors - Photo Detectors - CdS Cells (59 items)
Optical Sensors - Photo Detectors - Logic Output (134 items)
Optical Sensors - Photo Detectors - Remote Receiver (1188 items)
Optical Sensors - Photodiodes (1062 items)
Optical Sensors - Photoelectric, Industrial (11149 items)
Optical Sensors - Photointerrupters - Slot Type - Logic Output (1063 items)
Optical Sensors - Photointerrupters - Slot Type - Transistor Output (1171 items)
Optical Sensors - Phototransistors (808 items)
Optical Sensors - Reflective - Analog Output (332 items)
Optical Sensors - Reflective - Logic Output (134 items)
Position Sensors - Angle, Linear Position Measuring (1290 items)
Pressure Sensors, Transducers (26790 items)
Proximity Sensors (3762 items)
Proximity/Occupancy Sensors - Finished Units (240 items)
RTD (Resistance Temperature Detector) (87 items)
Shock Sensors (15 items)
Solar Cells (103 items)
Strain Gauges (22 items)
Temperature Regulators (Mechanical) (3947 items)
Temperature Sensors, Transducers (3457 items)
Temperature Switches (917 items)
Thermistors - NTC (5293 items)
Thermistors - PTC (1251 items)
Thermocouple, Temperature Probe (431 items)
Tilt Sensors (55 items)
Ultrasonic Receivers, Transmitters (96 items)
Vibration Sensors (58 items)

Instrumentation Amplifiers:

The basic circuit used to convert a signal to 0 .. 5V is an instrumentation amplifier:



Instrumentation Amplifier: $Y = \left(\frac{R_1}{R_2}\right)(V_a - V_b)$

Example: Assume R is a RTD (temperature sensitive resistor) with the following relationship:

$$R = 1000(1 + 0.004T)\,\Omega$$

Design a circuit which outputs

- 0V at 0C and
- 5V at +50C

Solution: First, determine the resistance at the two endpoints:

0C: R = 1000

50C: R = 1200

Step 2: Convert resistance to voltage. Assume a voltage divider with a 1k resistor. Then

$$V_x = \left(\frac{R}{R+1000}\right)5V$$

0C: Vx = 2.5V

50C: Vx = 2.7273V

Step 3: Determine whether Vx connects to the + input (Va) or the - input (Vb)

The output, Y, is to increase as Vx increases. Connect the + input (Va)

Step 4: Determine the gain.

$$Gain = \frac{\text{change in output}}{\text{change in input}}$$

$$Gain = \left(\frac{5V-0V}{2.7273V-2.5V}\right) = 22$$

Pick R1/R2 = 22. Let

- R1 = 10k
- R2 = 220k

Step 5: Determine the offset.

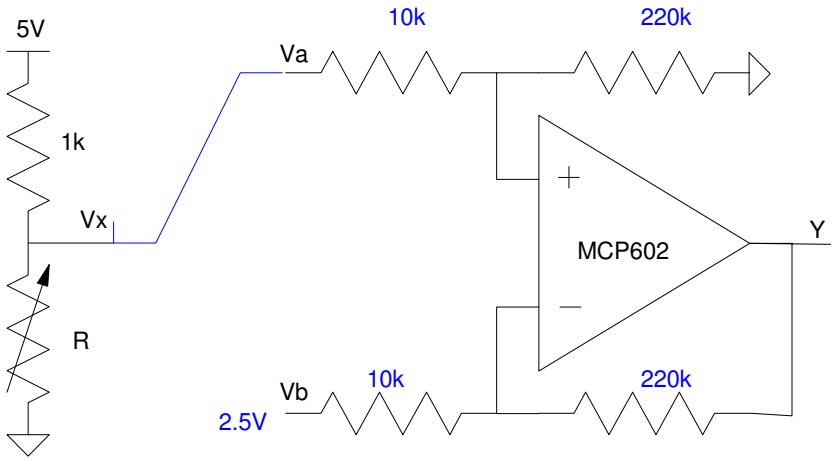
The output should be 0V when the input is 2.5V (0C)

$$V_y = gain(V_a - V_b)$$

$$0V = 22(2.5V - V_b)$$

$$V_b = 2.5V$$

The resulting circuit is as follows:

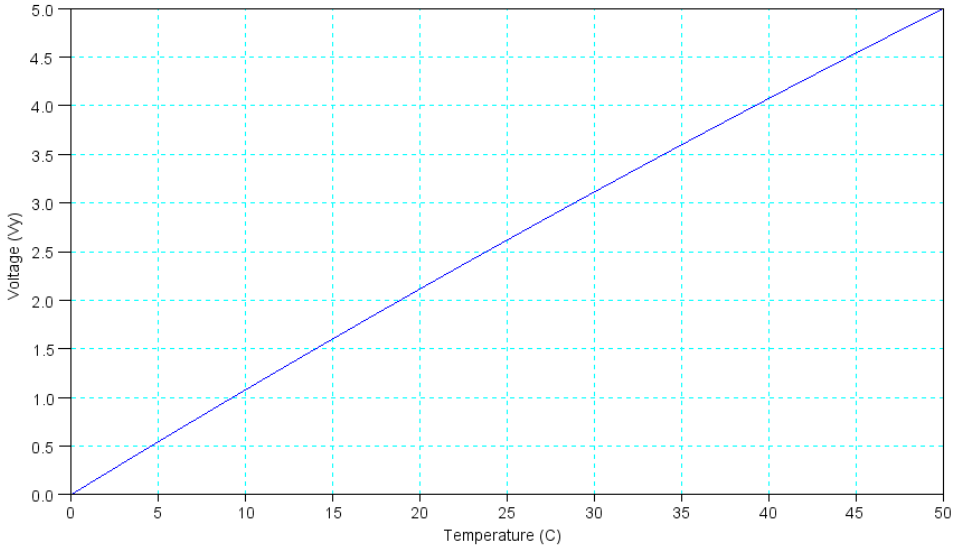


Instrumentation Amplifier: Output is 0V at 0C and 5V at +50C

The output voltage vs. temperature is then:

```
-->T = [0:0.1:50]';
-->R = 1000 * (1 + 0.004*T);

-->Vx = R ./ (1000+R) * 5;

-->min(Vx)
```

```
    2.5

-->max(Vx)

    2.7272727

-->gain = (5 - 0) / (max(Vx) - min(Vx))

    22.

-->Vb = min(Vx)

    2.5

-->Vy = gain*(Vx - 2.5);
-->plot(T,Vy)
-->xlabel('Temperature (C)');
-->ylabel('Voltage (Vy)');
```



Output of the Instrumentation Amplifier

Calibration

Assume the output of the instrumentation amplifier feeds an A/D input, such as RA1. Calibration takes the raw A/D reading (0 .. 1023) and converts this back to what you're trying to measure (temperature in this case). Essentially, calibration is curve fitting.

Assume you want to curve fit the above data with a straight line:

$$T \approx a \cdot A/D + b$$

where A/D is the 0..1023 raw A/D reading. Some types of calibration are:

- Endpoint Calibration: Pass a line through the endpoints
- Lease Squares: Come up with a line that minimizes the mean squared error

To find the least squares solution, rewrite the curve fit in matrix form where B contains your basis funcitons:

$$Y_{501x1} = B_{501x2} A_{2x1}$$

$$T = \begin{bmatrix} A/D & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

In the previous Matlab code, there are 501 data points, making B a 501x2 matrix. You cannot invert a 501x2 matrix. To solve, multiply both sides by B transpose:

$$B^T Y = B^T B A$$

B^TB is a 2x2 matrix, and can usually be inverted. A is then

$$A = (B^T B)^{-1} B^T Y$$

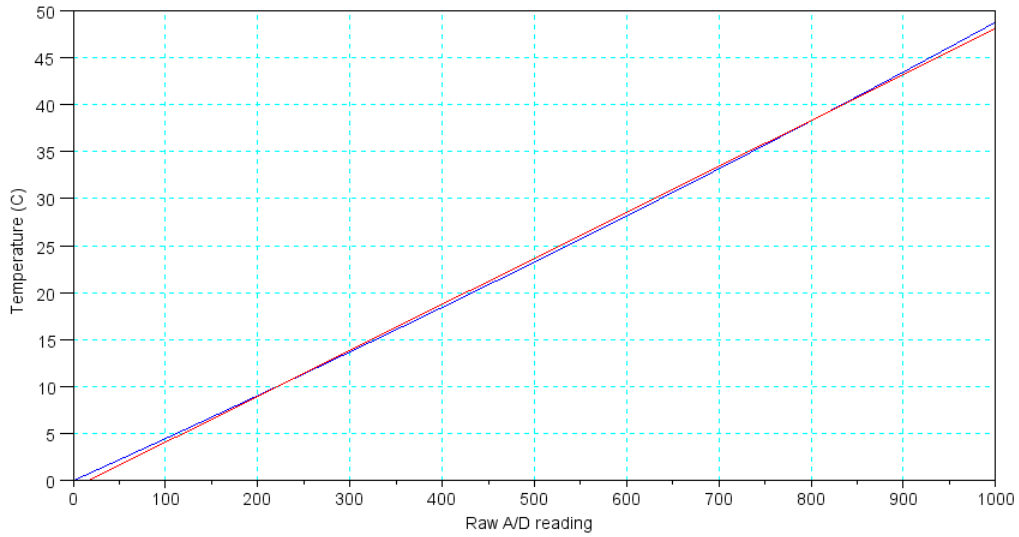
In Matlab:

```
A2D = Vy*1023/5;
A2D = round(A2D);

Y = T;
B = [A2D, A2D.^0];

A = inv(B'*B)*B'*Y

    0.0488903
  - 0.8000303
```

$$T \approx 0.0488903 \cdot A/D - 0.80003$$

Plotting this in Matlab:

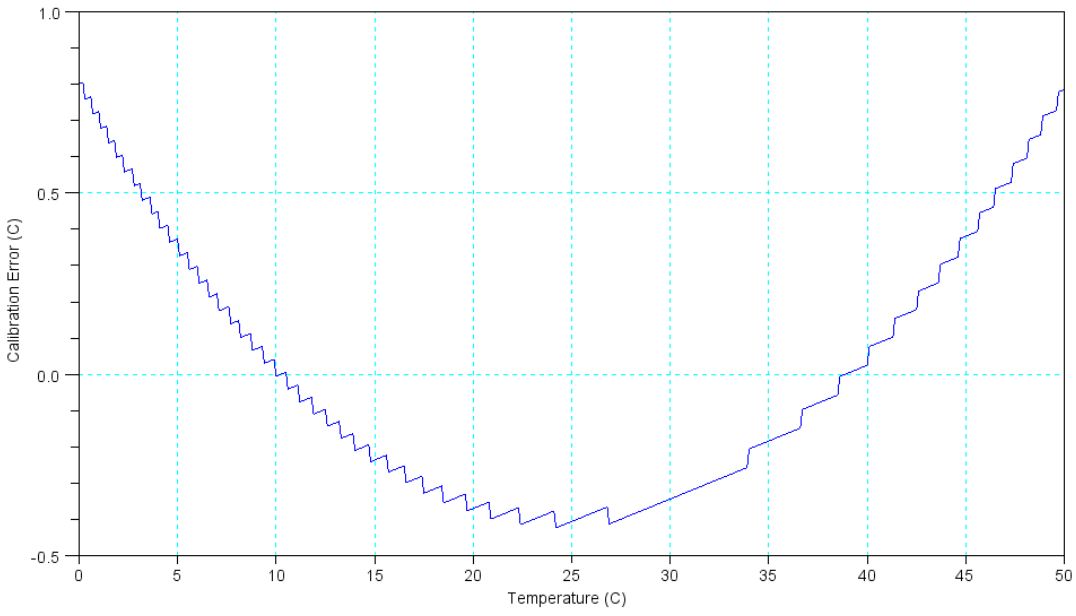
```
plot(A2D, T, A2D, B*A)
xlabel('Raw A/D reading');
ylabel('Temperature (C)');
```



Actual A/D vs Temperature Relationship (blue) and Linear Curve Fit (blue)

To determine the calibration error, plot the difference (the residual)

```
plot(T, T-B*A)
xlabel('Temperature (C)');
ylabel('Calibration Error (C)');
```



Calibration Error for a Linear Curve Fit for Temperature

The estimated temperature based upon the previous calibration function is off by 0.8 degrees.

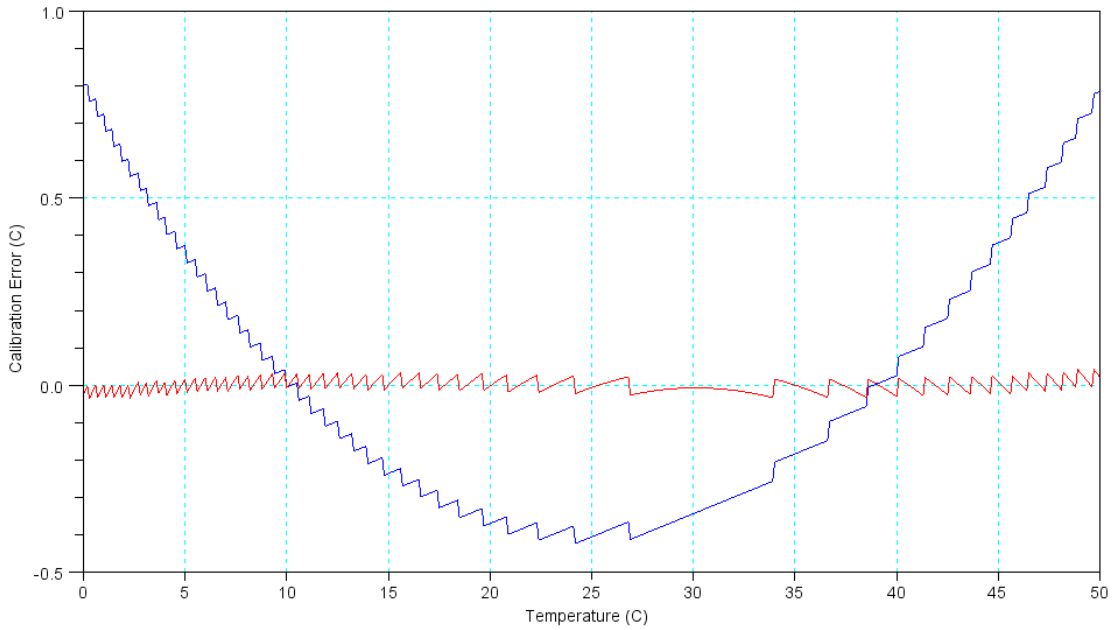
Note that the residual looks like a parabola. You can reduce the error by using a parabolic curve fit. To do this, change the basis:

$$T \approx a \cdot A/D^2 + b \cdot A/D + c$$

```
B = [A2D.^2, A2D, A2D.^0];

A = inv(B'*B)*B'*Y
Warning :
matrix is close to singular or badly scaled. rcond =    0.0000D+00

 A  =

a =     0.0000046
b =     0.0441556
c =     0.0258837
```

The resulting residual error is then within 0.04C

```
max(T - B*A)

 ans  =    0.0424377

min(T - B*A)

 ans  =  -0.0351971
```



Calibration Error for a Linear Curve Fit (blue) and Parabolic Curve Fit (red)

At this point, the quantization noise (the steps above) are dominating the error. There is no point in adding more terms.