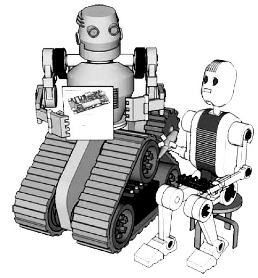
BCHS Advanced Computer Programming
Introduction to Robotics
By Daniel Johnson



Understanding Sensors

Sensors used to measure just about everything you can imagine.

> A sensor is a device that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument.

We are only interested in the case where the signal can be read by a computer. The connection between a computer and sensor is called the interface.

1 Sensor Interfaces
Sensor interfaces are either analog or digital.

1.1 Analog Sensor Interfaces
Because computers are digital in nature we must use a ADC (Analog to Digital Converter) when we read analog sensors. For that reason many uC's including our PICs have one or more ADCs included on the chip

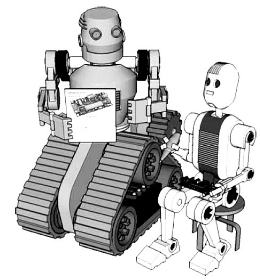
1.1.1 The Analog to Digital Converter
An ADC can directly read voltage, and indirectly read current and resistance. Ohm's law is used to convert current and resistance to voltage. The range of voltage measured by an ADC is generally limited to the range from Vss to Vdd which is zero to 5 volt in our case.

We will be using the Microchip C18 Compiler (MCC18) peripheral library to control the ADC. C:\MCC18\doc\periph-lib\AD Converter.htm

The library includes the following calls.

```
•  void OpenADC(              // Configure the A/D convertor.(ADC_V3, 4, 5, 6)
    unsigned char config,
    unsigned char config2,
    unsigned char portconfig);
•  void SetChanADC( uChar channel ); // Select input channel for A/D converter.
•  void ConvertADC( void );  // Starts the A/D conversion process.
•  char BusyADC( void );     // Is the A/D converter performing a conversion?
•  int ReadADC( void );      // ead the result of an A/D conversion.
•  void CloseADC( void );    // Disable the A/D converter.
```

Keep in mind that the OpenADC call is not the same for all PIC processor types. When you use this code on another processor you may have to rewrite the OpenADC call to conform to that used by the new processor.



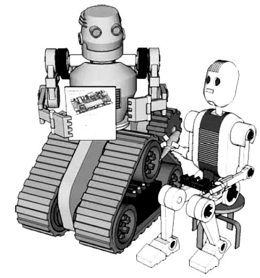
The following program demonstrates a simple ADC conversion on the Junebug.

```
// SENSOR1: simple 1 channel ADC conversion
#pragma config OSC = INTIO2, WDT = OFF, LVP = OFF, DEBUG = ON
#include <p18f1320.h>
#include <adc.h>
#include <delays.h>

void main( void )
{
  int result1, result3;

  // crank up the internal clock to 8MHz
  OSCCONbits.IRCF0=1; OSCCONbits.IRCF1=1; OSCCONbits.IRCF2=1;
  TRISA = 0xFF; // all inputs just to be sure

  // configure A/D convertor
  OpenADC( ADC_FOSC_32      &
           ADC_RIGHT_JUST   &
           ADC_12_TAD,             // wait this long for conversion
           ADC_CH1          &      // use RA1/AN1
           ADC_REF_VDD_VSS  &      // range is Vss to Vdd
           ADC_INT_OFF,ADC_4ANA ); // RA0-RA3 are digital

  Delay10TCYx( 50 );     // Delay for 50TCY after open
  while(1)
  {
    SetChanADC( ADC_CH1 ); // Paranoid, already done
    Delay10TCYx( 50 );     // Delay for 50TCY after changing channel
    ConvertADC();          // Start conversion
    while( BusyADC() );    // Wait for completion
    result1 = ReadADC();   // Read result
    Nop();  // BREAKPOINT here and see LOCALS window!
    Nop();
  }
  CloseADC();              // Disable A/D converter
  while(1);
}
```

With the while(1) loop will keep CloseADC() from executing. *Why did I include it ?*

The only complexity here is in the OpenADC command that configures the ADC. For now it is enough to accept most of the settings without question.

The first setting of interest in ADC_CH1 in the first parameter. It sets the ADC to use AN1 which is the RA1 pin on the 18F1320. The third paramager ADC_4ANA causes RA0-RA3 to



read analog values instead of digital on input. For these bits output is till digital. This is good because we can still use RA0 to charlieplex the Junebug LEDs.
The actual conversion process only requires 3 calls but we will use a few more.

```
SetChanADC( ADC_CH1 ); // Paranoid, already done
Delay10TCYx( 50 );     // Delay for 50TCY after changing channel
ConvertADC();          // Start conversion
while( BusyADC() );    // Wait for completion
result1 = ReadADC();   // Read result
```

SetChanADC is not needed here but it is included to illustrate that it is possible to change ADC channels without closing and opening the ADC.

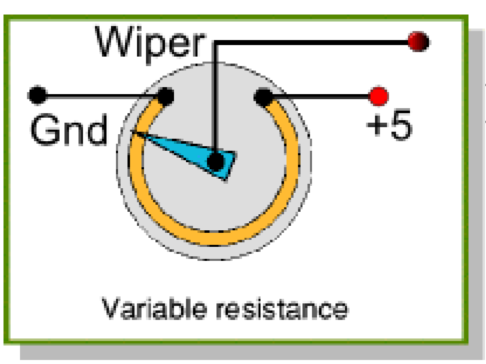
ConvertADC tells the ADC that we want to start a conversion. A conversion takes some period of time so we wait for the ADC to tell us it is done by repeatedly checking BusyADC(). When the ADC is finished we call ReadADC to get the value and save it in result1.

There is a lot more going on behind the scenes but we can make use of the ADC without knowing everything.

1.1.1 Sensing Voltage (for sensors that vary the voltage)

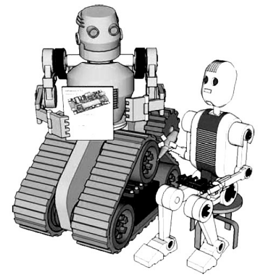
The easiest analog sensors to read are the ones which vary a voltage. If the voltage range falls between Vss and Vdd we can read the voltage directly using the on board ADC. If the sensor ouput voltage is not suitable to be directly read by the ADC there are methods that can be used to scale it.

To preform our experiment we need a variable voltage source. The Junebug is equipped with two variable resistors (potentiometers) label VR1 and VR2. The bottom switch on the DIP switch connects VR1 to AN1/RA1. The position above bottom connects VR2 to AN3/RA3. These are clearly marked to the left of the 8 position DIP switch.



How does the variable resistor work? The orange arc in this illustration is a strip of carbon. The resistance between the GND and +5 terminals is 10K. The resistance between GND and the WIPER depends on its position on the carbon arc. The resistance is directly proportional to its position. If the wiper if fully to +5 resistance is 10K. Move it ¼ of the way in the direction of GND and the resistance is 7.5K. In the center position it is 5K.

We are not interested in measuring the resistance here. We are going to apply power and sense the voltage between the wiper and ground. When the wiper is at +5 the ADC will sense



5V. Move it ¼ of the way in the direction of GND and it will sense (3/4)*5 or 3.75V. In the center position it will sense 2.5V.

Exercise 1: Create a project for the Junebug called sensor1 and include SENSOR1.c as a source file. Be sure to add the linker file 18F1320.lkr. Build the project without errors.

Turn on the VR1 dip switch.

Place a breakpoint at the position indicated in SENSOR1.c.

Open the LOCALS window and watch how the value of result1 changes as you rotate VR1.

Questions:

1. What do the values associated with result1 indicate?

   ____

2. What could we do to these values more meaningful?

   ____

3. Can you think of more then one possible meaning for the values?

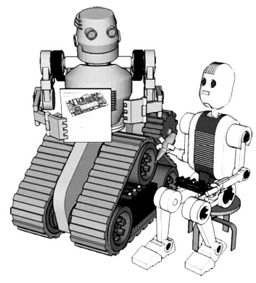
   ____

4. If the VR1 switch was OFF and we connected AN1 to a sensor with a analog voltage how would we interpret the values of result1?

   ____

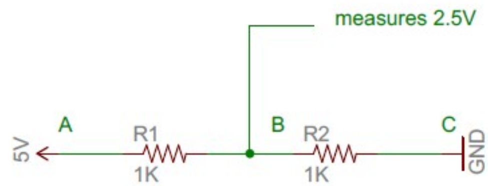
   ____

   ____



1.1.2 Sensing Resistance (for sensors with variable resistance)

To read the resistance of a sensor we need to create a voltage related to it by ohms law. This is commonly done using a voltage divider. From our earlier studies we know that we can determine the voltage drop of each resistor in series between any voltage and ground. The schematic to the right shows two 1K resistors in series between points A and C. The voltage between A and B is 5V. What is the voltage between A and B?

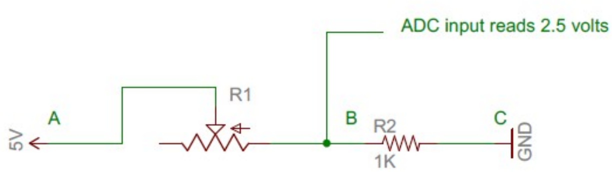


R1 and R2 are both 1K resistors. We know that for series resistances $R_{total} = R_1 + R_2 + \ldots R_n$

In this case we have the two resistors

$R_{total} = R_1 + R_{2.}$
$R_{total} = 1K + 1K$
$R_{total} = 2K.$

By simple inspection we can see that each resistor must provide an equal voltage drop so the voltage between A and B must be 2.5V.



In this schematic we have replaced R_1 with a sensor that varies its resistance with temperature. But for the purpose of this discussion we can ignore that and treat it like a simple variable resistor.

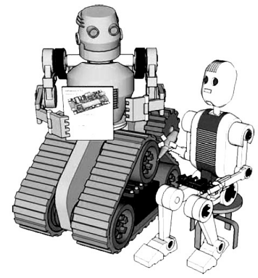
Reading the sensor is done by determining the resistance of R_1. In the case where X is 2.5 volts we know that R_1 must be be 1K ohms.

It would be useful to know how to determine the resistance mathematically because we will not always be able to solve for it by inspection.

Recall that Ohm's law can be used to determine voltage(V), current(I), or resistance(R). To find any one of these you must first know the other two. To find the resistance R_1 we first have to determine the voltage across R_1 and the current flowing through R_1.

What is the voltage across R_1?

We know that the voltage across $R_{2.}$ (B to C) is 2.5V. It is easy to see that the voltage across R_1 (A to B) must be $V_{total} - (V_{B\ to\ C})$ which is 5V – 2.5V or 2.5V.



$V_{R1} = V_{total} – V_{R2}$

What is the current flowing through R_1 ?
We do not know enough about R_1 to find its current.
But if we look at the schematic it is clear that the current flowing through R_1 is that same as the current flowing through R_2..
Happily we do have the V and R for R_2 so we will calculate the current through it and substitute it in for R_1 when needed. Lets calculate the current through R2.

$I_{R2} = V_{R2}/R_{R2}$
I_{R2} = 2.5V/1000R
I_{R2} = 0.0025 amp or 2.5 ma

The current through R_1 is the same as that in R_2. So the current through R_1 is .0025 amp.

This is the last bit of our puzzle. Now that we know V and I for R_1 we can again use Ohms law and calculate R.

$R_1 = V_{R1}/I_{R1}$
R_1 = 2.5V/.0025 amp
R_1 = 1000 Ohms

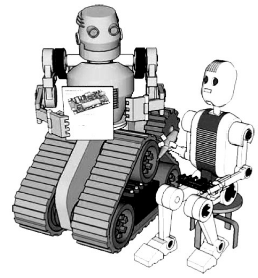
That was a lot of work to calculate a number that we already knew! But using the same method we can determine any resistance with our voltage divider.

Suppose that our ADC is reading 1V at B (voltage across R2 is 1V). We can find the resistance for R_1 if we first find V_{R1} and I_{R1} as we did in the previous section.

We know V_{R1} must be

$V_{R1} = V_{total} – V_{R2}$
V_{R1} = 5 V– 1V
V_{R1} = 4V

We can not find I_{R1} directly but it is the same as I_{R2} so we solve for I_{R2}.



$I_{R2} = V_{R2}/R_{R2}$
$I_{R2} = 1V/1000R$
$I_{R2} = 0.001$ amp or 1 ma
and
$I_{R1} = I_{R2}$

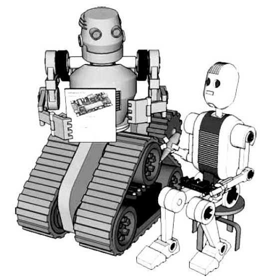
Now that we have V_{R1} and I_{R1} we can solve for R_1.

$R_1 = V_{R1}/I_{R1}$
$R_1 = 4V/.001$ amp
$R_1 = 4K$ Ohms

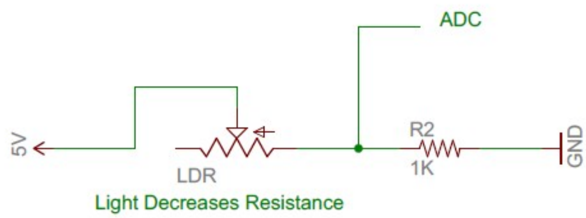
We can do a sanity check. We know that R_1 is 4K and R_2 is 1K so R_{total} is 5K. We know the voltage drop for R_{total} must be 5V and the current is .001 amp.

V= IR
V=.001amp * 5000R
V= 5V

Now that we have R_1 what does it mean. That depends on the sensor. Often manufactures will provide a formula to convert R_1 to the temperature or whatever the sensor was monitoring.



Exercise 2:
This exercise uses the same program used to sense voltage, sensor1.c.



Connect a Light Dependent Resistor (LDS) and 1K resistor to form a voltage divider.

Questions: *(Answer these prior to class on Wednesday)*

5. How do we know where to connect the wire marked ADC to on the Junebug?

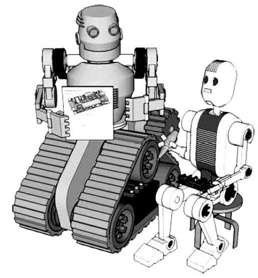
6. Why can we use the same program to read resistance as we used to read voltage ?

7. What do the values associated with result1 indicate?

8. What could we do to these values more meaningful?

9. As the light level increases how will the value of result1 change ?

10. What would happen if we choose to use a 1 Ohm resistor for R2 ? (Do not try it !)



1.1.2 Sensing Current

We can not sense current directly so we must use Ohm's law to convert it to voltage.

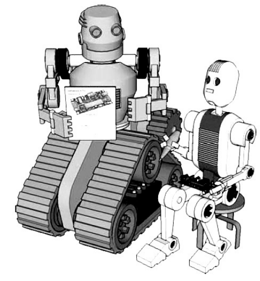
The current through a resistor of know value can be determined by sensing the voltage drop associated with the resistor.

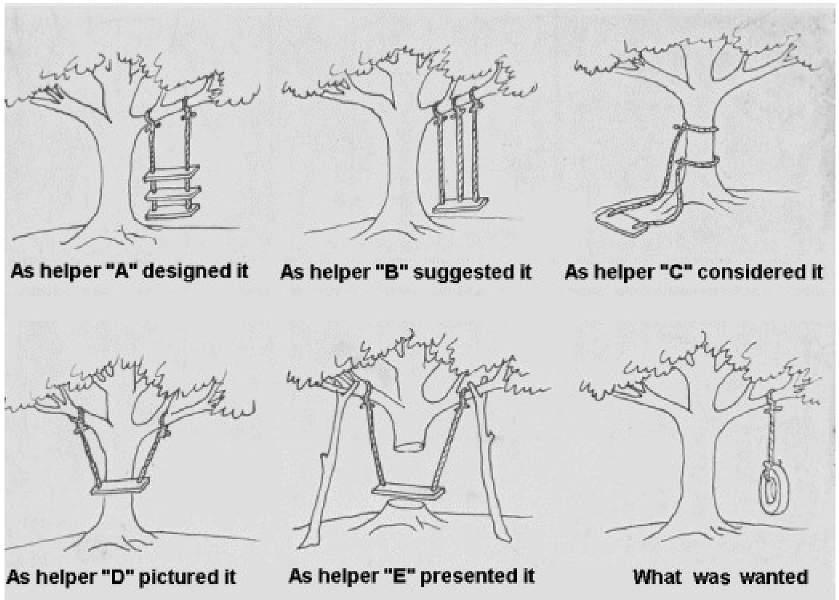
$I = V_{sensed}/R$

When determining the current being used by a motor or other device it is best to keep R as small a possible to prevent a unacceptable voltage drop. It is easy to see how that can happen with a simple example. Suppose we have a 10V battery delivering 1 Amp to a motor and we want to sense the current. If we use a 1 Ohm resistor it will create a voltage drop of

V = I R
V = 1amp 1ohm
V = 1V

For each additional ohm of resistance the voltage will drop by one volt. A 5 ohm resistor would reduce the available voltage by ½. Resistors with values that are a fraction of an ohm can be purchased for this purpose. The catch behind using low resistor values is that they generate a small voltage drop. If we use .001ohm resistor in the above example the voltage drop will only be .001 or 1 mv (millivolt). We can use a properly configured operational amplifer (opamp) to amplify the voltage by 5000 to get 5V. That is a topic for another time.





Schmitt trigger: When the input is higher than a certain chosen threshold, the output is high; when the input is below another (lower) chosen threshold, the output is low; when the input is between the two, the output retains its value. The trigger is so named because the output retains its value until the input changes sufficiently to trigger a change. This dual threshold action is called hysteresis, and implies that the Schmitt trigger has some memory