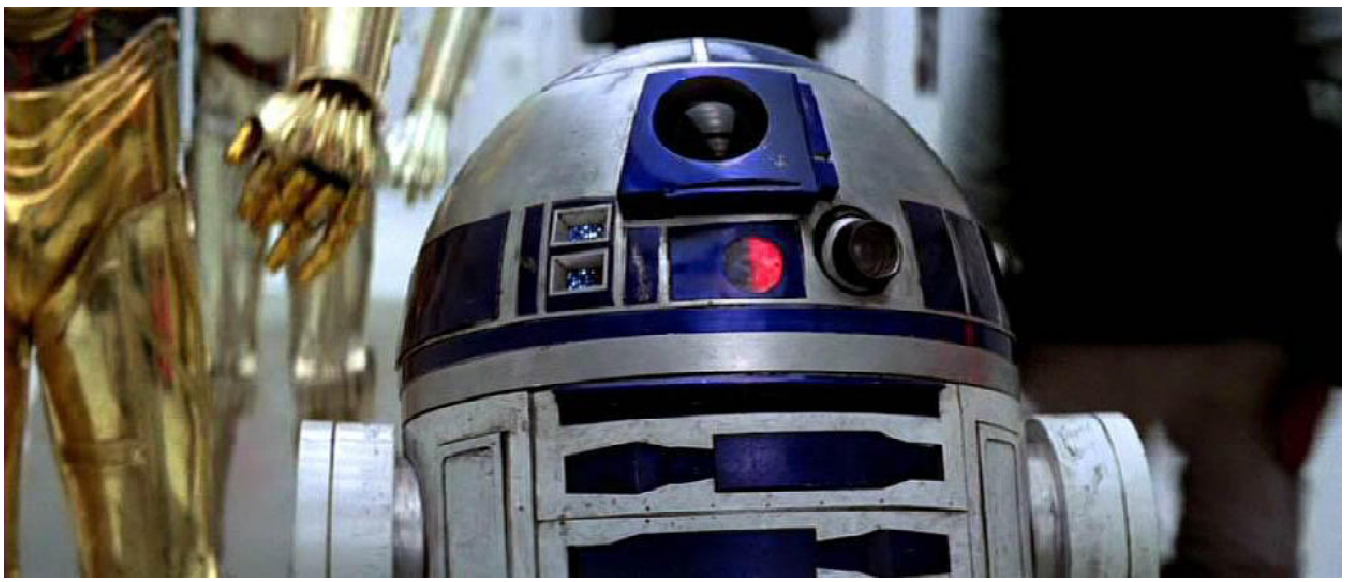
# Picaxe Electronics For Astromechs



By Murray Jones

Version 2.5 November 27th 2007

# Contents



***Note:*** *These are just the electronic control for these devices, no mechanical connection ideas are included for connecting servos to doors or holoprojectors.*

# Introduction To The Picaxe System

## What is the Picaxe System?

The Picaxe is a Microcontroller system which is programmed from your computer with simple programming system to tell the board what you want it to do. The programming uses simple commands such as:

**Servo 1,150**

In this case the command is telling the servo connected to pin 1 to move to position 150. Simple huh?

## Why Use Picaxe?

There are a number of reasons for using the Picaxe System some of them include -

* Quite cheap
* Boards already assembled (always a plus)
* Very flexable since YOU program what you want them to do.

## Where Can I Get Them?

The Picaxe systems are available online from the following places -

**USA - SparkFun:**
**www.sparkfun.com**

**UK - Rev-Ed:**
**www.rev-ed.co.uk/picaxe/**

**Australia - Microzed:**
**www.microzed.com.au/**


*NOTE: A Picaxe 18X "Power" board is available but it is mostly for powering small motors up to 1.5amps so not needed for projects included here.*

## How Much?

The following starter pack I have found most useful for Astromech use -

| | |
|---|---|
| Picaxe 18X | $24.95 USD (SparkFun) |
| | £15.10 Inc Vat (Rev Ed) |
| | $50.70 AUSD (Microzed) |

Price includes Board, Picaxe Chip & Programming Software CD. Price for Rev Ed and Microzed also includes programming cable. Programming cable sold separately for SparkFun packs:

| | |
|---|---|
| Programming Cable - Serial | $6.95 |
| Programming Cable - USB | $25.95 |

The cheaper serial cable is sufficient and the only reason you would need the USB cable is if your computer does not have a serial port.

A slightly cheaper option is to buy the board and the chip only, as well as the download cable. The software can be downloaded from the Rev-Ed site for free (it is around a 27mb download). If you are buying multiple Picaxe systems, or buying extra Picaxe systems after the starter pack, you are better off buying this way.

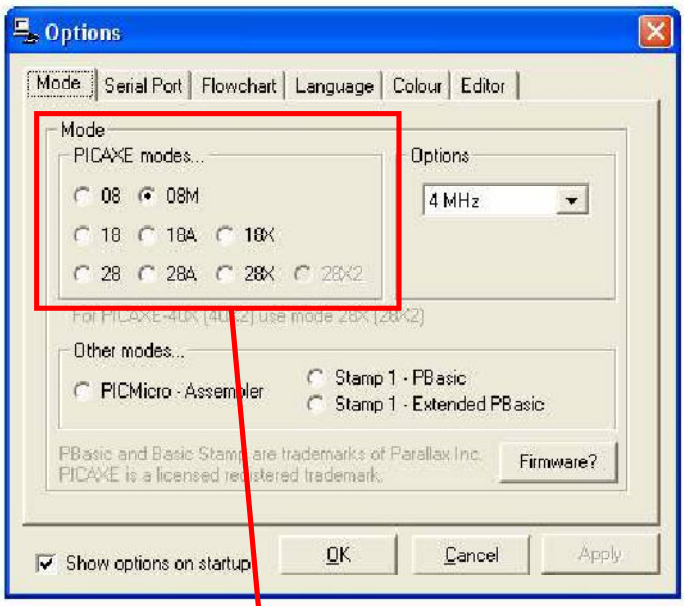
One potential problem with the Picaxe 18 board is that it doesn't have any mounting holes, how you get around this is up to you. But a number of ideas include slots to slide the board in your Astromech or putting the board in a plastic container and attaching the container to the frame.

Other Picaxe chips and board combinations are available, such as Picaxe 8M and Picaxe 14M, but they are slightly more limited and they are supplied as a "kit" which you must solder together. I for one would rather have a fully assembled board. Also the price difference with the 14M and the 18X is not much different, so the convenience of an already assembled board makes it a good buy.

# Picaxe Programming Editor Basics

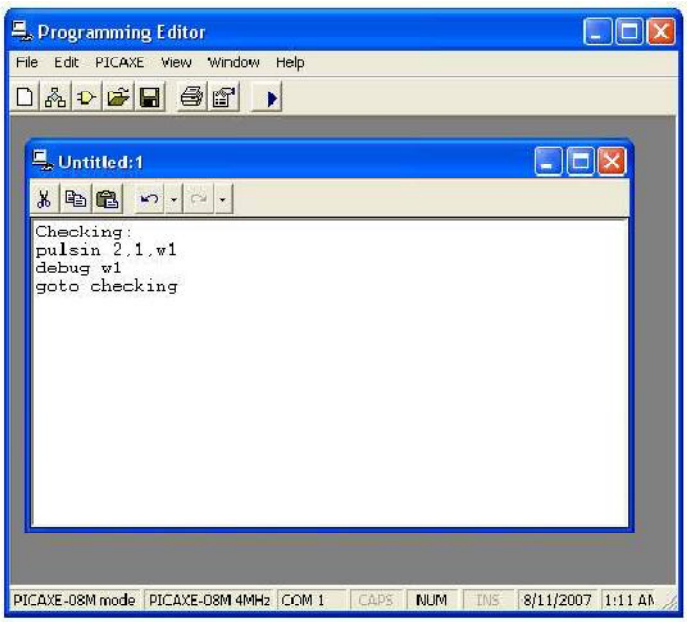
When you first startup the programming editor you will see this startup screen to select your options.

The only critical option you must change is the **"Picaxe Mode"** option to the type of chip you are using. We will be using the 18X.

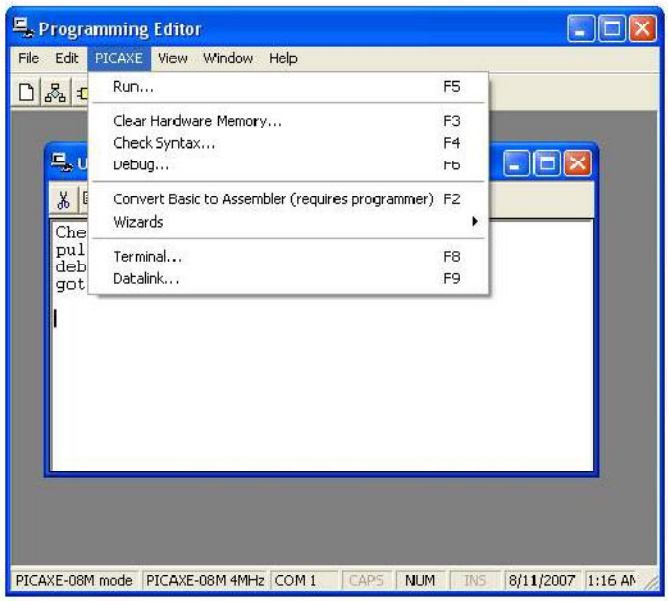



*Change Mode to the type of Picaxe chip you have*

After you click on the OK button you get to the programming editor. Type in your program, such as below:




When you are happy with your program and wish to run it on the Picaxe system simply plug in the programming cable, and make sure the Picaxe has power. In the menu select **PICAXE > RUN** and your program will be downloaded to your Picaxe.




There are over fifty commands in Picaxe Basic so the PDF Datasheets on the Rev-Ed site should be printed out and studied if you want to get the most out of your Picaxe system. The most important are:

Picaxe Manual 2 - Basic Commands
Picaxe Manual 3 - Electronic Interfacing Circuits

Other PDF Datasheets are also important, such as the datasheets for the individual boards etc.

Note: You can find the programs together in a zip file. The name of the program is to the side. Program "Bolt1" shown here for example.

**Bolt1**

```
main:
Readac 1,b0
Debug b0
Goto main
```

# Beginning With Breadboards

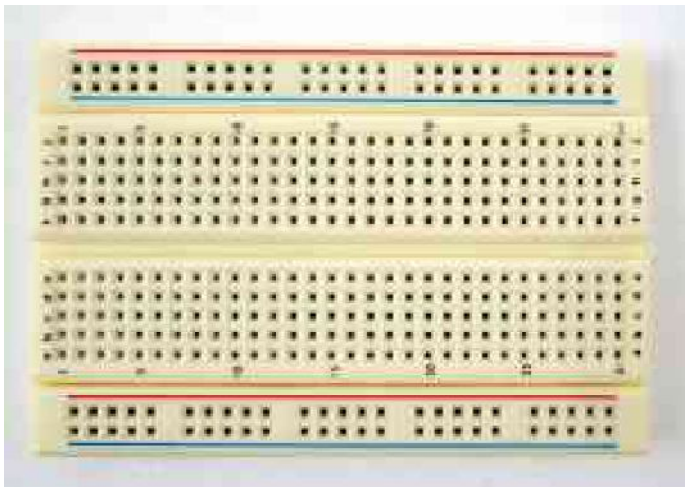
By far the easiest way to try out the projects described in this booklet is to use a breadboard. Any changes to the circuit require no soldering, far better when trying things out.

The following items are needed:

* Picaxe Chip
* Breadboard
* 22k Resistor
* 10k Resistor
* 4.7k Resistor
* 3.5mm Stereo Line Socket
* Assorted single strand hookup wire

That's all that is needed to get the Picaxe chip running. Of course you will need the items described in the following projects to get individual projects running. Such as LED's, Servo's, extra resistors etc.

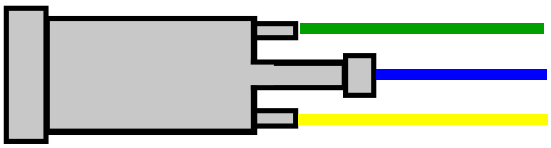
Below is a typical breadboard, although you can get many different types, too large is better than too small.



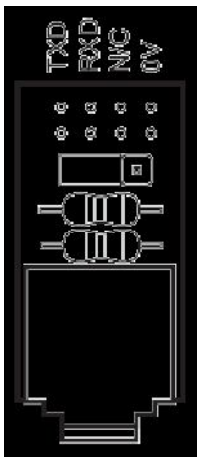
To plug your download cable in you will need to make up a connector from a 3.5mm stereo line socket. It has to be a stereo, not mono socket such as below.

When you unscrew the black plastic cover there will be three connection points. Solder different coloured single strand hookup wire to the three points. I have used blue, green and yellow here.

In the picture below, the middle large connector is at the top:

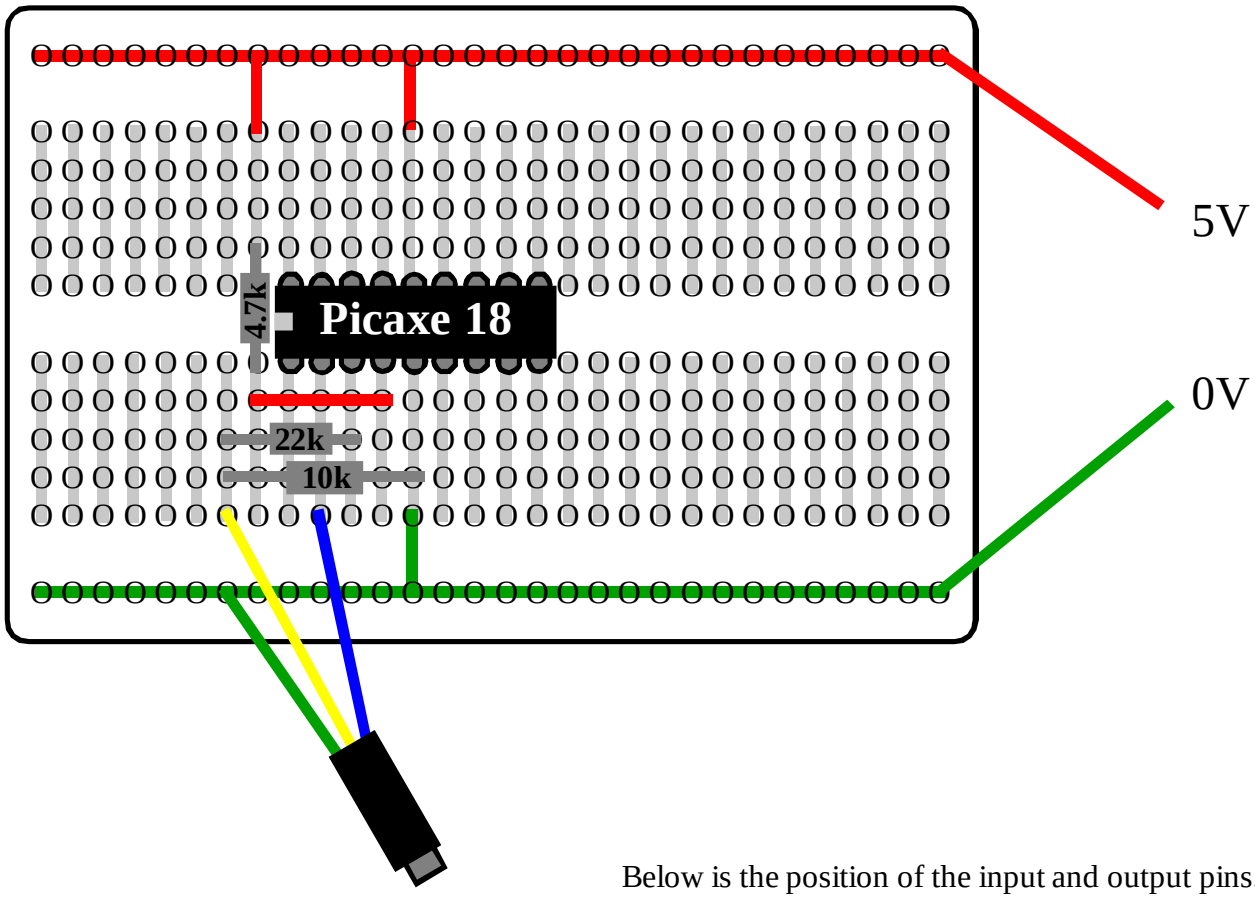


Sparkfun, Microzed and Rev-Ed also also sell a breadboard adaptor. At Sparkfun they are $4.95. You will have to solder it together, but it replaces the 3.5mm stereo line socket, as well as the 10k and 22K resistors in the download circuit on the next page. The breadboard adaptor looks like this -

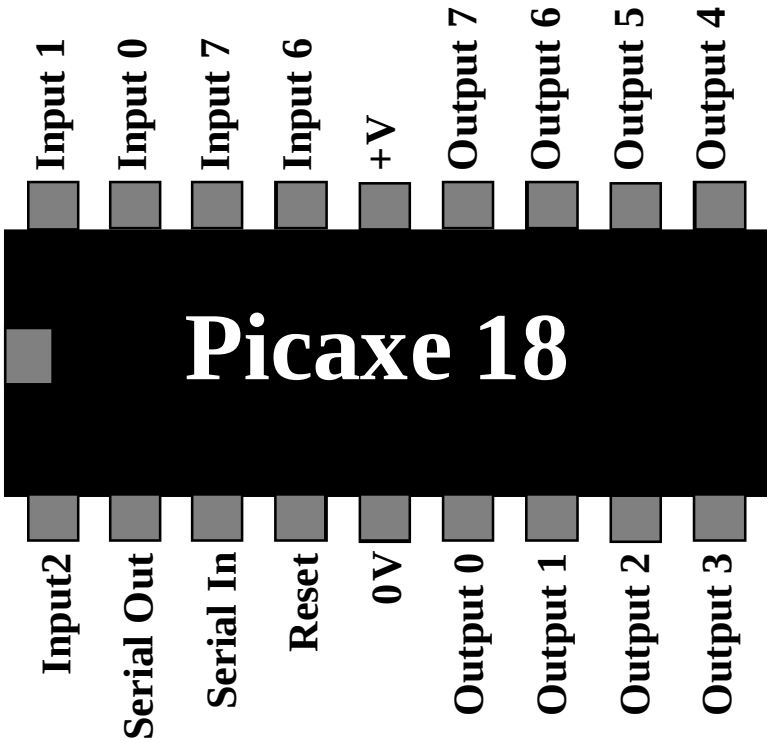



You will also need a power source. Anything from 4.5v to 5.5v DC. You might use 4 rechargable batteries, or a plug pack that outputs 5v DC.

Below is all you need to get the Picaxe18 chip working on a breadboard



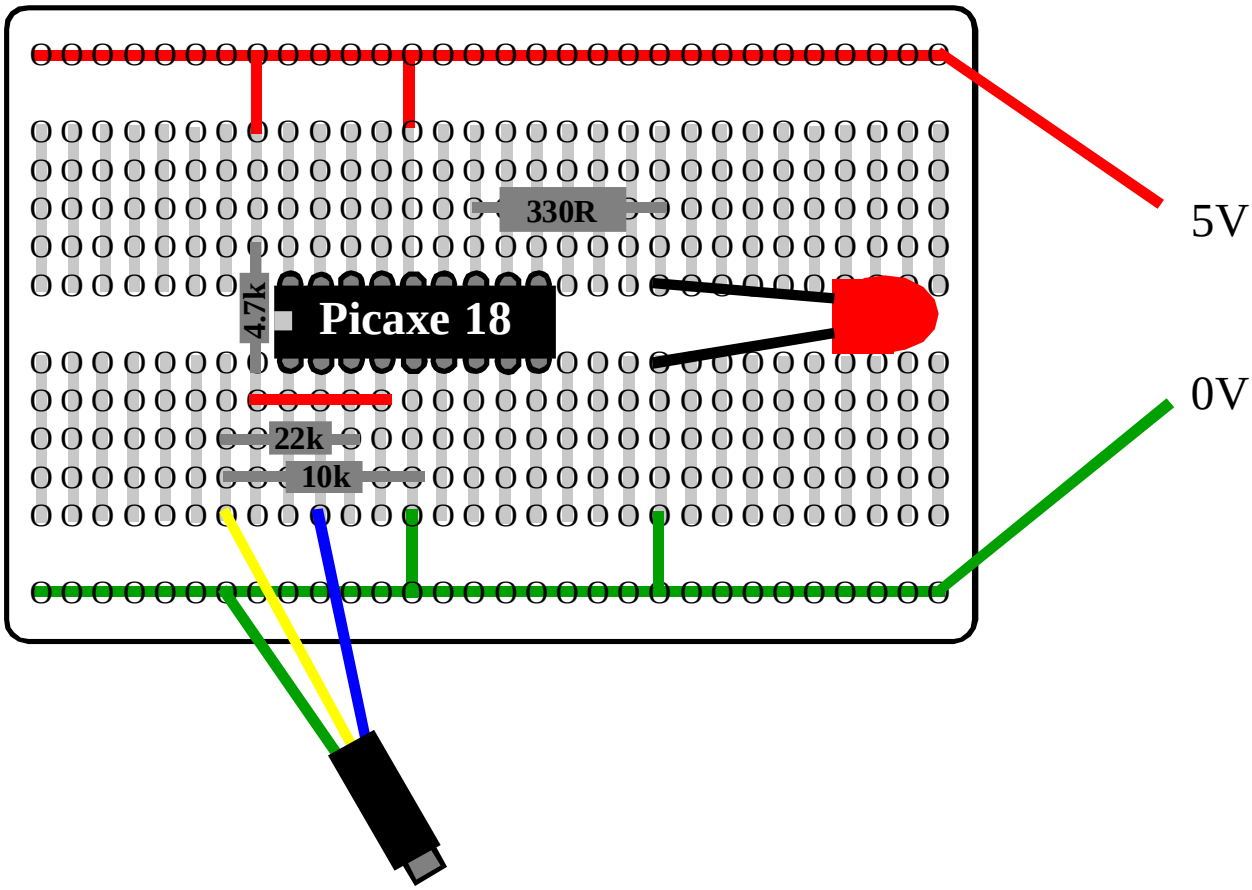

Below is the position of the input and output pins.
Remember outputs 0 - 7 and inputs are 0, 1, 2, 6, 7.




Whether you use the above download circuit or buy the Breadboard Adaptor, you will still need the 4.7k resistor (in some places it is written as 4K7 - they mean the same thing). Without the 4.7k resistor, which is connected to the reset pin, it will not work.

The Picaxe Manual 3 - Microcontroller Interfacing Circuits, available on the Rev-Ed site contains many simple circuit ideas to try, including servos, switches & leds. Once you get the hang of the breadboard you can try the circuit ideas presented in this booklet and without soldering and unsoldering wires every time you want to change the circuit or try a new idea.

Below is the hookup diagram for flashing led's. The Picaxe on it's own can only output 20ma per output pin, so if you want to try a string of led's on a single output you need to put the ULN2803A chip on the Breadboard. The string of led's is mentioned in the logic display section. The ULN chip can be a bit tricky, because unlike the led circuit below you must connect the output of the Picaxe to the ULN chip and then from the ULN to 5V, not 0V.




The only items added to the standard download circuit on the previous page is a 330R resistor, a led and a short wire link.

The LED is connected to output 6 in the above. A simple program to flash the led on and off would be:

Anything the standard Picaxe 18x board can do, you can also do on the breadboard. Everything from servos, leds, CF music system, switches etc. You will find it much easier to try new ideas and programs when you dont have to get your soldering iron out to change the circuit.

Board1

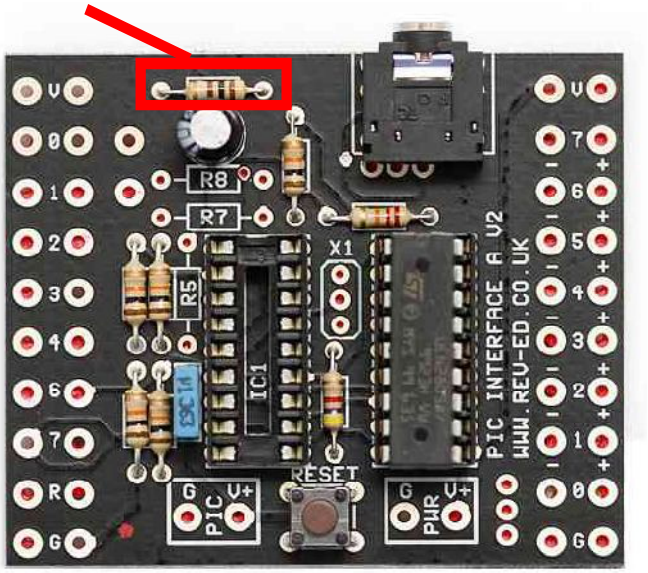
```
main:
high 6          'led on
wait 2          'wait 2 seconds
low 6           'led off
wait 2          'wait 2 seconds
goto main       'go back to start
```

# 8 Channel Front or Rear Logic Display

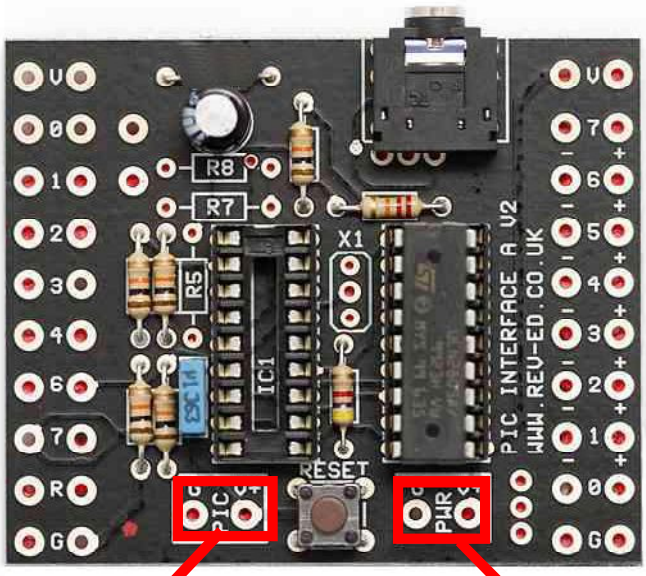
## Modifications to the Picaxe 18 Board

To use the Picaxe-18 board for LED Logic displays the following resistor has to be removed:

**Cut off this resistor**



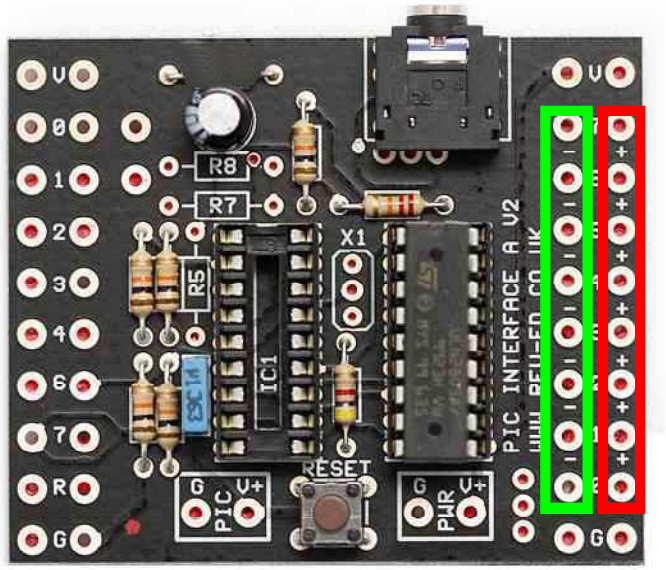

This will now seperate the power supplies so that for the Logic display it will allow a larger voltage to be used.



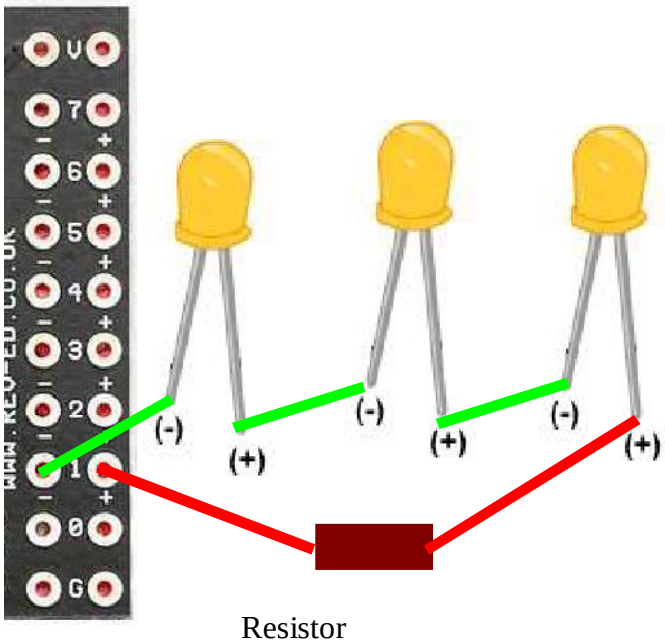

**Picaxe Power 5V**

**Power for LED's**

## Wiring Your LED's




To give you a better idea the above shows **GREEN** for LED out from the Picaxe connection ( - ) & **RED** for the other side of the LED string (+).




Resistor

This is how each string of Led's is attached to the board. Their is no room on the board for the resistor so it must be attached to the LED's off the board. Their are eight outputs you can use numbered 0 to 7, the above string of LED's is attached to output 1. Just do the same for the other outputs.

You will have to work out which resistor you will need for each string of LED's. Their are many online LED calculators to help you. One I have used is LED Wizard

**http://led.linear1.org/led.wiz**

You will need to know Source Voltage, LED Voltage, LED current, and number of LED's in each string.

When you have everything wired up it is time to download a program to the picaxe and try it out -

Logic1

```
main:
random w0          'get random number
let pins = b0      'make outputs high or low
pause 1000         'wait 1 second
goto main          'go back to start
```

The program simply picks a random number and using that number turns on the string of LED's randomly. The entire program is only 5 lines, but if you want to expand on the program the Picaxe 18x can have up to 600 lines of code, so lots of range for experimentation.

If you don't want the LED's to appear randomly but want a particular pattern, then other commands can be used.

For example the command:

```
high 1
```

would turn on output 1, and

```
low 1
```

would turn it off.

If you wish to turn on outputs at the same time you need the Let Pins command.

The command below would turn on all outputs -

```
let pins = %11111111
```

And to turn all outputs off -

```
let pins = %00000000
```

**% 0 0 0 0 0 0 0 0**

This position represents output 7

This position represents output 0

Simply put a "1" in the position of the output you want to go on, and a"0" for any you want to be off.

Combine this with a pause command and in no time at all you will have the Logic Display the way you want it. Any time you want the display to look different, just change the program.

With the pause command

```
pause 500      = half a second
pause 1000     = 1 second
pause 2000     = 2 seconds
```

Use whatever amount looks good to you.

An example of a non random program -

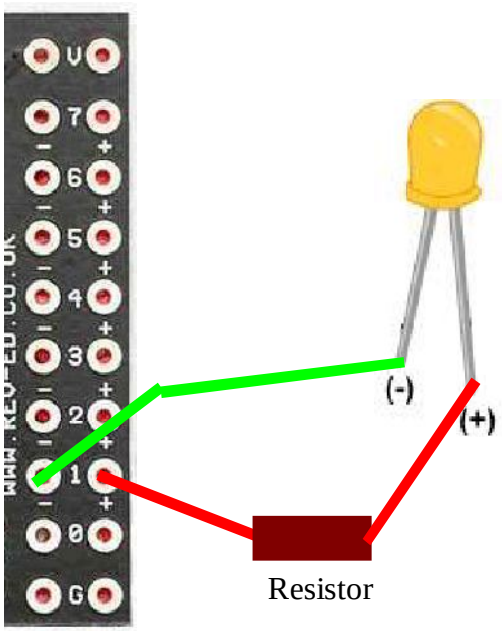
Logic2

```
main:
let pins = %10100110
pause 500
let pins = %01011001
pause 300
let pins = %11100100
pause 600
let pins = %00101001
pause 500
goto main
```

Remember you can have up to 600 lines of code so there is plenty of room for experimentation.

# Process State Indicators (PSI)

The LED's are connected exactly the same as the Logic Displays except only one LED is uses for each output:




Calculate the resistor value as before using an online LED calculator.

http://led.linear1.org/1led.wiz

The program is similar to the example given for the Logic Displays except you would only need to use 4 of the outputs (Front - Red/Blue & Rear - Green/Yellow).

In this example program -

Front Red - Output 0

Blue - Output 1

Rear Yellow - Output 2

Green - Output 3

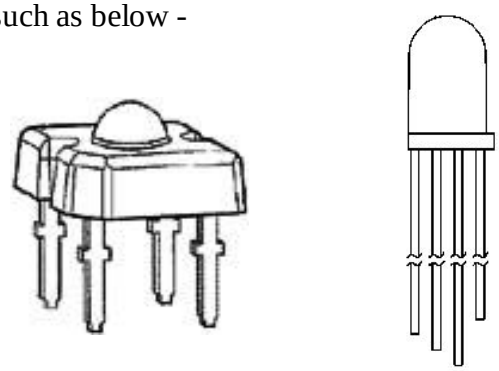
PSI1

```
main:
high 0
low 1
high 2
low 3
pause 1000
low 0
high 1
low 2
high 3
pause 500
goto main
```

You can still use unused pins 4 - 7 for something else if you wish.

## RGB LED's

If you can spare six outputs, you might like to use two RGB LED's. Unlike normal LED's, they have 4 legs, one for Red, one for Green and one for Blue, plus the common. They can vary in looks but should have four pins such as below -



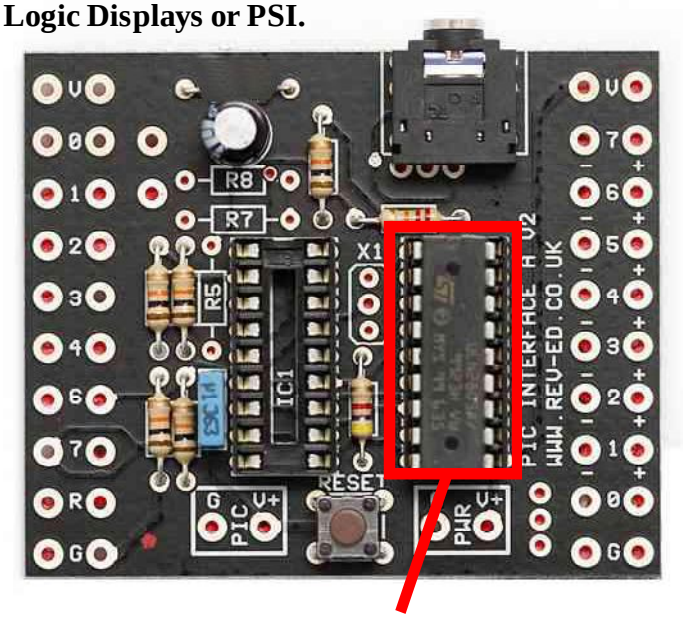
They come in two varieties, Common Cathode and Common Anode. For use with the Picaxe 18 you need to buy Common Cathode. Dont forget the Resistors on each output leg, each color uses different amounts of power so they need to be calculated as if they were three seperate LED's.

You can have more colours than Red Green and Blue though, by combining colours you can have the following: (1 = On, 0 = Off)

| | Outputs | | |
|---|---|---|---|
| | **Red** | **Green** | **Blue** |
| **Red** | **1** | **0** | **0** |
| **Green** | **0** | **1** | **0** |
| **Blue** | **0** | **0** | **1** |
| **Yellow** | **1** | **1** | **0** |
| **Light Blue** | **0** | **1** | **1** |
| **Pink** | **1** | **0** | **1** |
| **White** | **1** | **1** | **1** |

As you can see they can produce the required Red/Blue & Green/Yellow, but if you want something a little different, the option is there, and you can always go back to the standard colours just by changing the Picaxe program.

For Servo use a small change to the board is needed. The resistor described in the Logic Display section needs to be removed as normal to split the power supplies. Another change is needed for servo use. The ULN2803A chip needs to be removed and a 16 Pin, 8 x 330R resistor DIL pack is put in its place. **Do not make this change if you are using the board for Logic Displays or PSI.**

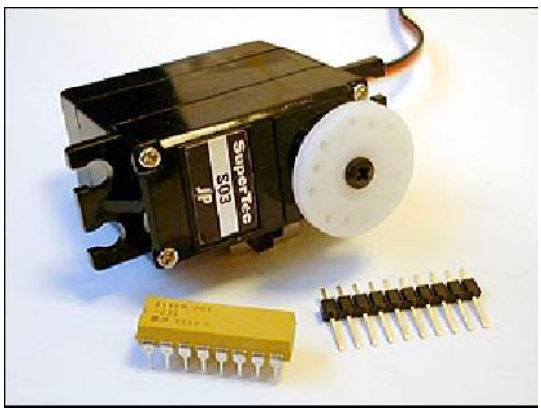



**ULN2803A Chip pictured**
**Replace with -**
**16 Pin, 8 x 330k resistor DIL pack**

The ULN2803A is an 18 pin Chip, the replacement Resistor pack is 16 pin, this is correct, just leave the bottom row of pins of the socket empty.

You can buy a servo upgrade pack includes one servo, DIL resistor pack and 10 pin header strip.

Price -
Rev-Ed: Order Code - AXE030 - £13.00
Microzed: Order Code - AXE030 - $45.00
SparkFun: Not available



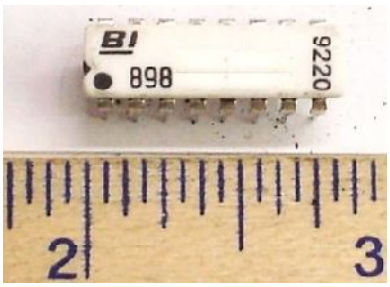
The Resistor DIL pack can be bought seperately if you already have enough servos. A quick Google brought up a few places to buy -

## www.electronicplus.com

Search by Part Number: **898-3-R330**

Description:
330k ohm 16 pin dip (dual inline package) resistor network-consists of 8 individual resistors each isolated from each other (individual resistors are connected to pins 1 & 16, pins 2 & 15, 3 & 14, etc

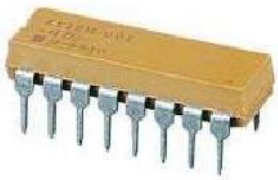
Price: $1.75




*Note: The search will produce two results. The first with 330 OHM in description is the one you want. NOT the 330K.*

## www.rapidonline.com

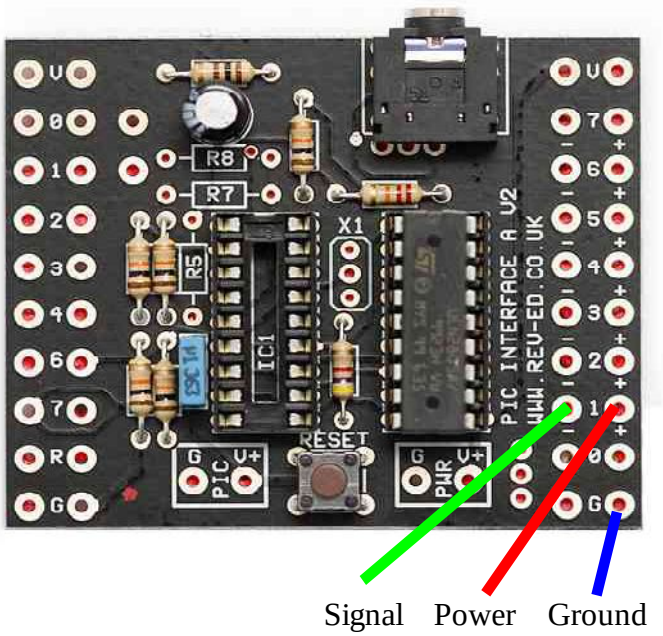
Search Order Code - **63-0635**



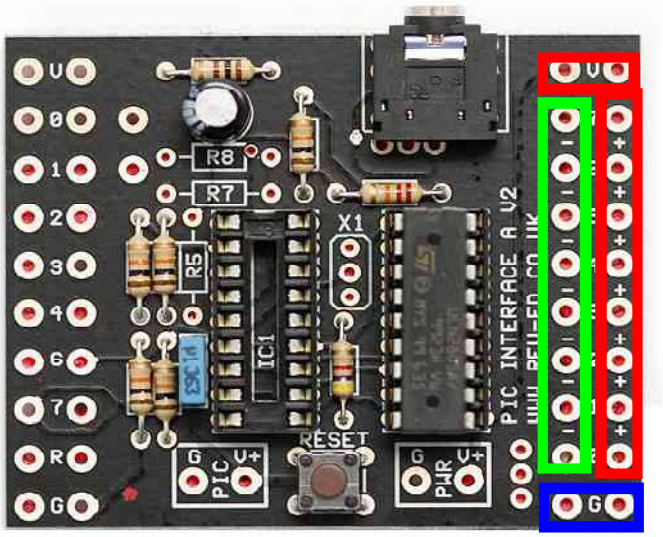
## www.techsupplies.co.uk

Search Order code - **RES034**

This is by no means the only places to buy this resistor pack, but it is not a common item. Just remember you need - **16 pin, 8 x 330k ohm Resistor DIL pack.**




The above example shows the connection wires go to on the servo. The ground wire will have to be shared between servos.




To give you a better idea the above shows **GREEN** for the connection for the Servos signal wire, **RED** for the Power wire, and **Blue** for the Ground connection wire.

On my Futaba Servos they are wired -

Red - Power
Black - Earth
White - Signal

But this can vary between manufacturers.

## Example Programs

The servo command is simply Servo, Output Pin Number, Servo Position. The position is a number between 75 and 225. The servos command can go outside this but if you use a number outside your servos capability it could damage it, so the numbers here are conservative.

Servo Position 75 = Fully Left
Servo Position 150 = Middle
Servo Position 225 = Fully Right

Example program to move a single servo connected to output 1 -

Holo1

```
main:
servo 1,75          'move servo to one end
pause 2000          'wait 2 seconds
servo 1,150         'move servo to centre
pause 2000          'wait 2 seconds
servo 1,225         'move servo to other end
pause 2000          'wait 2 seconds
goto main           'loop back to start
```

If you want random movement the random number generator has to be used -

Holo2

```
main:
random w0           'get a random number
w1 = w0 // 151 + 75 'range convert 75 - 225
servo 1,w1          'move servo
pause 3000          'pause 3 seconds
goto main           'go back to the start
```

To move two servos -

Holo3

```
main:
random w0           'get a random number
random w2           'get a random number
w1 = w0 // 151 + 75 'range convert 75 - 225
w3 = w2 // 151 + 75 'range convert 75 - 225
servo 1,w1          'move servo 1
servo 2,w3          'move servo 2
pause 3000          'pause 3 seconds
goto main           'go back to the start
```

It is very easy to experiment until you get your holoprojectors moving the way you want.

# Using A Servo To Open A Door

Just the same as the Random Holoprojector movement you need to remove the resistor and replace the ULN2803A chip with the DIL resistor pack and the servos are wired exactly the same.

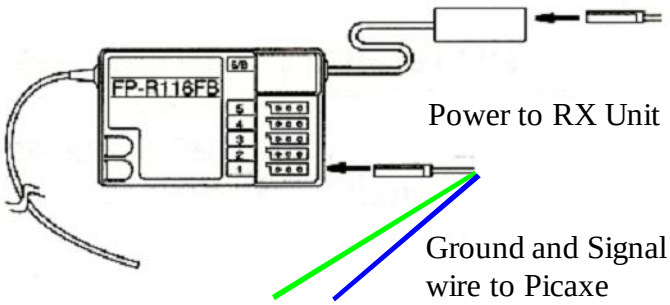
An example program -

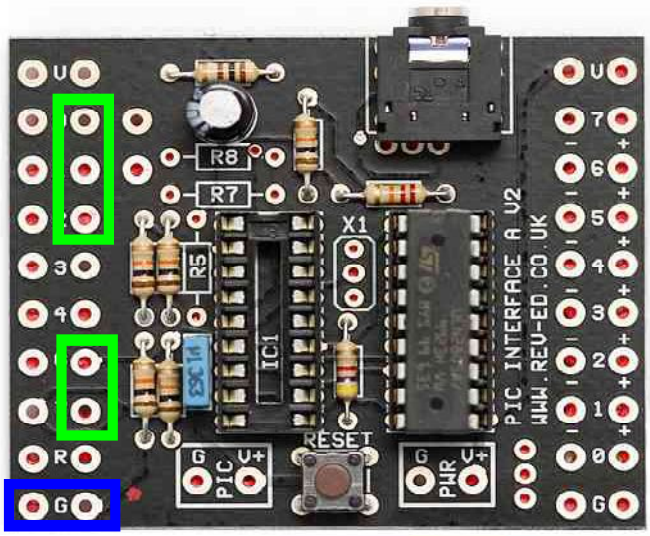
Door1

```
main:
servo 1,75          'move servo to one end
pause 2000          'wait 2 seconds
servo 1,225         'move servo to other end
pause 2000          'wait 2 seconds
goto main           'loop back to start
```

But this would just make the door continuously open and close so you need some way to tell the servo when to open or close. For that you need to receive your RC Radios RX Signal or use a separate 12 Channel Remote. The 12 Channel Remote is described later, I will focus on using the RC system.

Your Radio Control Receiver should look similar to below.




This time the board shows **GREEN** for the connection for the RC signal wire (Input) and **BLUE** for the Ground connection wire (Earth), the power wire doesn't need to be connected to receive the signal.

A Quirk of the Picaxe 18 system is that there is no pin 5 input and although pins 3 and 4 are included on the board, they cannot be used. So only 5 inputs - 0,1,2,6,7.

Plug in your Download cable from your computer to the Picaxe system and try the following program to see if everything is hooked up correctly –

*Note: You must leave the download cable plugged in so that the debug command can send info back to the computer -*

Door2

```
main:
Pulsin 1,1,b0
Debug b0
Goto main
```

Simply move the stick connected to the channel you have connected to the Picaxe and watch the numbers change when the stick is moved.

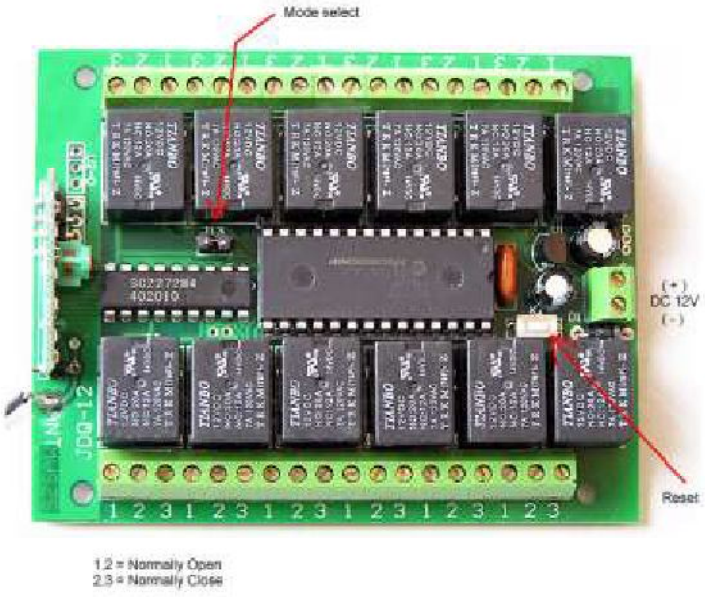
If everything is working correctly, wire in a servo as previously shown in the random Holoprojector movement and leave the RX unit connected. When you have done all that, try the following program -

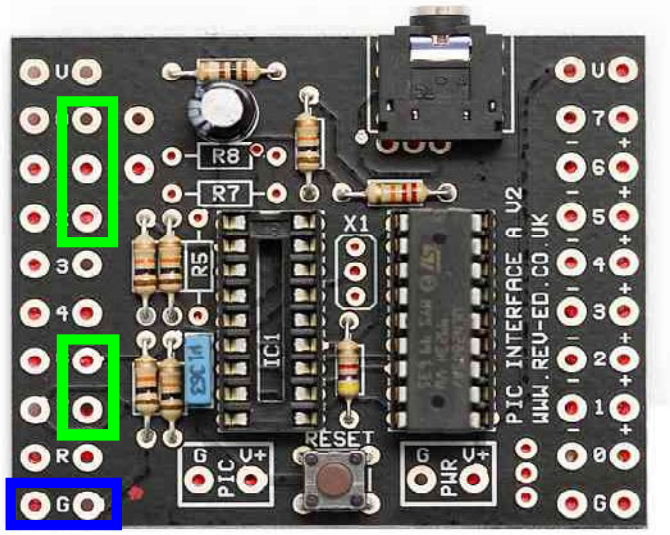
Door3

```
b1 = 0
main:
pulsin 1,1,b0
if b0 > 100 then goto main
b1 = b1 + 1
if b1 = 1 then goto open
if b1 = 2 then goto close
open:
servo 2,225
pause 1000
goto main
close:
servo 2,75
pause 1000
b1 = 0
goto main
```

# 12 Channel RF Relay Board Connection

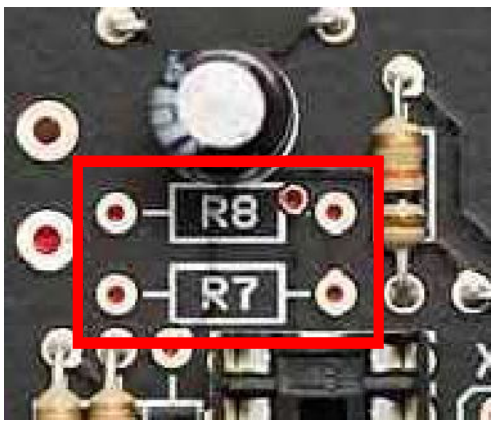
Your 12 Channel Relay Board should look similar to below -




Just wire the NO connection to one of the Picaxe input pins circled in Green, and the COM connection to the Ground on the Picaxe board circled in Blue. Don't forget the power for the Picaxe and the 12CH Receiver board. Also remove the jumper on the 12CH receiver board to turn on momentary mode. Don't forget to put in the Resistor DIL pack for Servo use as described in the random servo example.

Input pins 0 and 1 dont have the required pulldown resistors to be used in this way, but the required place on the board is left for them, R7 and R7. Just solder 10K resistors where shown below if you wish to use them.



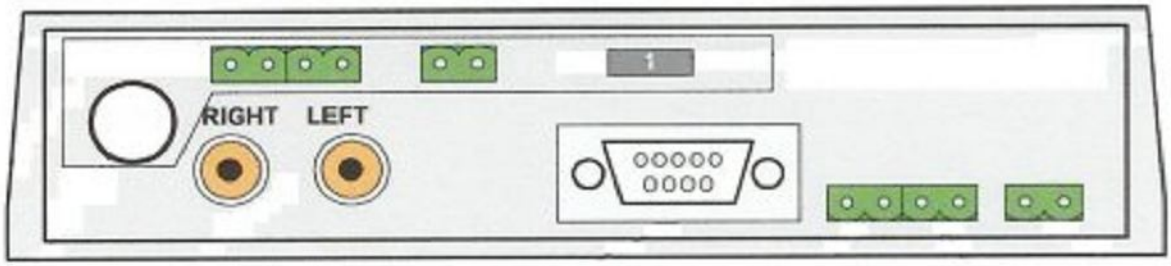

In this example Picaxe Program I will assume that only Relay 1 is wired up to input 2 on the Picaxe board and that you have a servo wired to output 3.

RFRelay1

```
b1 = 0
main:
if pin 2 = 1 then goto door
goto main
door:
b1 = b1 + 1
if b1 = 1 then goto open
if b1 = 2 then goto close
open:
servo 3,225
pause 1000
goto main
close:
servo 3,75
pause 1000
b1 = 0
goto main
```

If everything is working correctly, every press of button 1 on the remote will trigger the relay to move between two positions. Handy for doors on your Astromech.

# CF Sound II & III Control




For the following you need a CF-III Sound System, or for those that have the older CF-II, I will show how to hook that up also.

The CF III Sound System – available from:

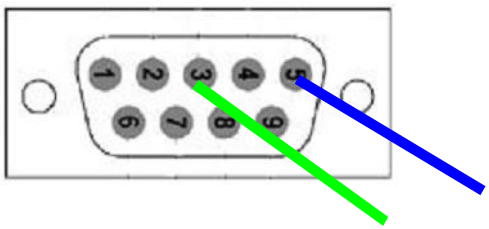
**www.cfsound.com/index_CFSound.asp**

You also need a Compact Flash Card – Not LEXAR brand, they are not compatible. Formatted to Fat 16. Also needed is a Speaker – 4-8 ohms, 15 watt max and a 9 Pin Female Serial Connector shown below.

NOTE: The extra contact sense boards are not needed for this setup.

Make sure that the Picaxe board is set up the same as for the servo connection with the resister DIL package and no more than 5v to the picaxe.

Below is a view from the **rear** of the 9 pin Serial Connector, connect the Ground wire to pin 5 and serial in to pin 3 as shown This is then plugged into the CF Sound II or III.




Connect the other ends of the above wires as shown, Ground to Ground BLUE Wire connection and the other wire to an output pin GREEN wire.

You will then need some R2 Sounds. They need to be converted to the following formats -

**CF II - WAV, Mono 8 bit 22 KHZ – PCM format.**

**CF III - WAV, Mono or Stereo 16 bit 44.1 KHZ – PCM format.**

Sounds can be converted with the sound recorder program that comes with windows.

Copy these sounds to your Compact Flash card, making sure they are renamed to a number plus a ”C” after the number so the sound will play when the contact is closed. i.e. – 01C.wav, 02C.wav up to 99C.wav if you have that many sounds.

Power both the CF II or III and the Picaxe system making sure you connect the speaker to the CF Sound, then try the following program to see if everything is hooked up correctly. In this example the output pin 1 of the Picaxe is connected to the CF Sound.

Sound1

```
main:
serout 1,n2400,($01,”P+01”,$03)
wait 4
goto main
```

This program will play sound 01 every four seconds, just substitute the 01in the inverted commas “ ” in the Serout command to play anyone of the 99 sounds.

Using the 12 Channel Remote to connect with the CF Sound and you can play random sounds.

# CF Sound + 12 Channel Relay Board Connection

For the following program connect the 12 Channel Relay board Relay outputs 1,2,3,4,5 to input pins 0,1,2,6,7 and the output pin 1 of the Picaxe is connected as shown in the previous CF Sound example.

Sound2

```
main:
if pin0 = 1 then but1
if pin1 = 1 then but2
if pin2 = 1 then but3
if pin6 = 1 then but4
if pin7 = 1 then but5
goto main

but1:
random b1
b2 = b1 // 20 + 1
SEROUT 1,N2400,($01,"p+",b2,$03)
wait 1
goto main

but2:
random b1
b2 = b1 // 20 + 21
SEROUT 1,N2400,($01,"p+",b2,$03)
wait 1
goto main

but3:
random b1
b2 = b1 // 20 + 41
SEROUT 1,N2400,($01,"p+",b2,$03)
wait 1
goto main

but4:
random b1
b2 = b1 // 20 + 61
SEROUT 1,N2400,($01,"p+",b2,$03)
wait 1
goto main

but5:
random b1
b2 = b1 // 19 + 81
SEROUT 1,N2400,($01,"p+",b2,$03)
wait 1
goto main
```

The previous example program is handy when you have banks of different "mood" sounds. so sounds 1-20 are happy sounds, sounds 21 - 40 are sad sounds, sounds 41 - 60 are chirps and beeps etc.

If instead you want a particular sound to play when a button is pressed - program below plays sound 1 when button 1 is pressed etc.

Sound3

```
main:
if pin0 = 1 then but1
if pin1 = 1 then but2
if pin2 = 1 then but3
if pin6 = 1 then but4
if pin7 = 1 then but5
goto main

but1:
SEROUT 1,N2400,($01,"p+01",$03)
wait 1
goto main

but2:
SEROUT 1,N2400,($01,"p+02",$03)
wait 1
goto main

but3:
SEROUT 1,N2400,($01,"p+03",$03)
wait 1
goto main

but4:
random b1
SEROUT 1,N2400,($01,"p+04",$03)
wait 1
goto main

but5:
SEROUT 1,N2400,($01,"p+05",$03)
wait 1
goto main
```

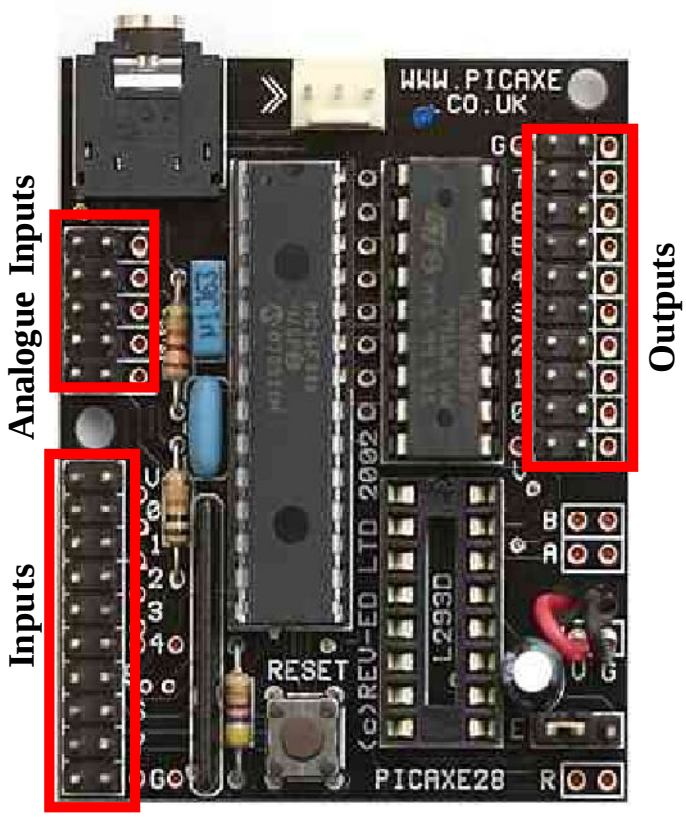
Or you can mix the above to have some random and some particular sounds. You could also have a sequence of particular sounds.

# Introduction to the Picaxe 28X

Now that you have your 12 Channel RF Relay Board working, the five input limit of the Picaxe 18 system might be a problem if you wish to control more doors, sounds etc than you have buttons. Their are two solutions.

The first presented here is to upgrade to a Picaxe 28. This larger system has 8 outputs, 8 inputs and 4 extra analogue inputs. Another advantage is that with a little bit of extra soldering a pin header to the output area, servos can be plugged straight in without removing the plug on the servo lead.

A Picaxe 28X Board is pictured below with inputs and outputs marked:



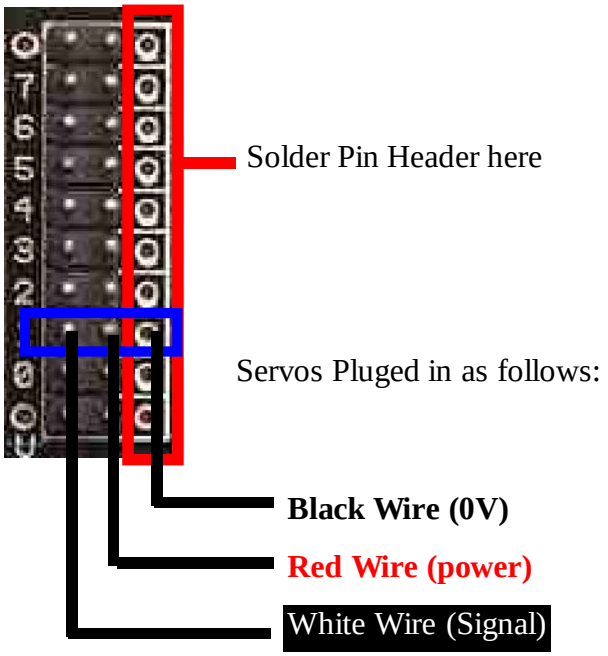

Prices:

| | |
|---|---|
| Sparkfun: | $37.95 |
| Rev Ed: | £21.87 |
| Microzed: | $77.00 |

Starter pack includes Board, Picaxe 28X Chip & Software CD. As always with Sparkfun, cable sold separately, the prices for Rev Ed and Microzed include Serial download cable, and the pack with USB cable is more expensive.

The smaller chip next to the Output pins is the ULN2803A, the same as the 18X board, and it can be replaced with the Resistor DIL Pack the same as the 18X board for servo use.

Also with the outputs you can see that their is a row of holes next to the pins, if you solder a row of header pins you can plug your servos directly on the board as previously mentioned.




With this board you can also separate the power supplies as with the 18X, this time you do not remove any resistors, just remove the jumper link at the bottom right corner of the board, and use the pins there as your external power as shown below.

# RF Relay Board Modes Setup

Another solution to get more out of your RF 12 Channel Remote would be to use MODES in software. How it would work would be that you use 1 button as a mode button.

For example if you are using the Picaxe 18X, you have 5 inputs. If you connect them to the RF Board as previously shown you have buttons 1 to 5 for Picaxe use. You can still use the others for non Picaxe applications but for Picaxe18X use 5 is the limit.

If you use one of the buttons for mode though, you have unlimited buttons, although too many modes would be confusing, 2 or 3 would be easier to remember.

In the following example buttons 1 to 5 are connected to the Picaxe, buttons 1 to 4 are as normal, but button 5 is Mode button, every press of the mode button would mean buttons 1 to 4 could be used for something else. In this example 3 modes are used and after each button press of the modes (5) button, the mode increases by 1, if mode is greater than 3, mode goes back to 1.

You could play a short sound from the CF Sound Board so that you could tell which mode was active. ie play sound once - mode 1, twice for mode 2 etc, and example is included near the end of the program.

The sound section of the program is this -

```
for b2 = 1 to b0
serout  1,n2400,($01,"P+01",$03)
wait 1
next b2
```

Simply change the 01 in the "P+01" section to a different number to use a different sound. Also the wait 1 command might need changing depending on how long the sound takes to play, the shorter the sound the better.

This example is unfinished, it is up to you to write what you want the Picaxe to do.

Modes1

```
b0 = 1
main:
if pin0 = 1 then but1
if pin1 = 1 then but2
if pin2 = 1 then but3
if pin6 = 1 then but4
if pin7 = 1 then but5
goto main

but1:
if b0 = 1 then .....
if b0 = 2 then .....
if b0 = 3 then.....

goto main

but2:
if b0 = 1 then .....
if b0 = 2 then .....
if b0 = 3 then.....

goto main

but3:
if b0 = 1 then .....
if b0 = 2 then .....
if b0 = 3 then.....

goto main

but4:
if b0 = 1 then .....
if b0 = 2 then .....
if b0 = 3 then.....

goto main

but5:
b0 = b0 + 1
if b0 > 3 then b0 = 1
for b2 = 1 to b0
serout  1,n2400,($01,"P+01",$03)
wait 1
next b2
goto main
```

# Serial Connection Between Picaxe Boards

If you decide to use more than one Picaxe board, it would be a good idea to get them to communicate, especially if you are using a slip ring and powering your dome electronics from the main batteries. Why would you need this? To control the electronics in the dome from your RF Remote.

If the two Picaxes share a common 0V rail from a battery, all you need is a single wire from an output of one Picaxe to an input of another to control whatever is connected to that Picaxe. If not then the 0V rail must be connected together. The Picaxe sending the information (Serout) has to have the servo setup as described with the 330R resistor DIL. Also make sure the picaxe that is the Send board, that the power is at no more than 5V. The send board would probably be used for servos as well so depending on what servos you are using, you might have the output power at 6V, this is to high for the serout to another board, you either have to lower the voltage to 5V or use a resistor in the connection to lower the voltage to 5V.

The command you would use is Serout and Serin.

```
Serout 1,2400,b1
Serin 1,2400,b1
```

In the above the 1 is the pin you are sending to or receiving from, the 2400 is the speed of the serial connection, and b1 the variable you wish to send or receive.

The following needs two programs, one for each board. In the example the Send Picaxe waits for a button press on the RF remote, then sends a command to turn on or off the holoprojector light on the Receive Board -

Program 1 - Send Board

Serial1

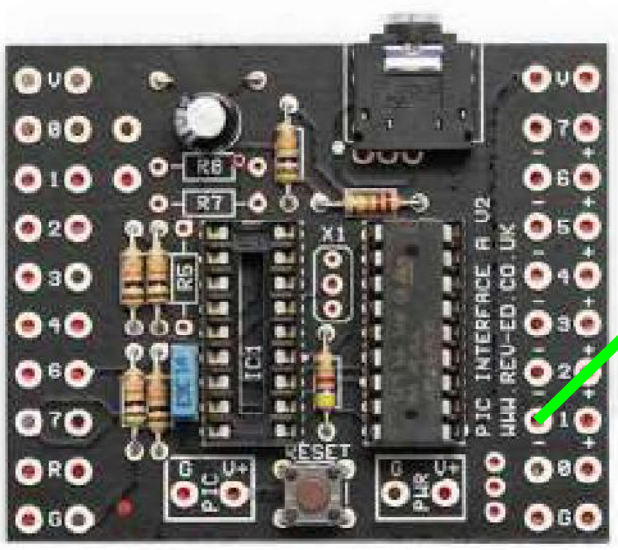
```
main:
if pin1 = 1 then holo
goto main
holo:
b0 = b0 + 1
if b0 > 2 then b0 = 1
serout 1,2400,b0
goto main
```

Program 2 - Receive Board

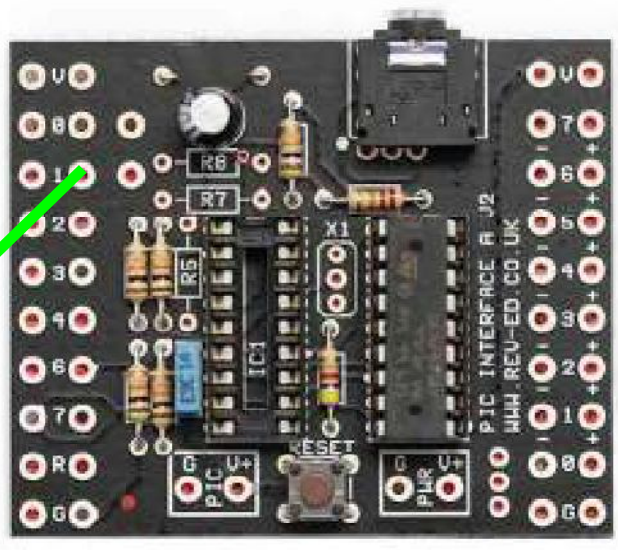
Serial2

```
main:
serin 1,2400,b0
if b0 = 1 then high 1
if b0 = 2 then low 1
goto main
```

You could have two way communication if needed by having another wire go from an Output on the Receive board to an Input of the Send Board.




Send Board




Receive Board

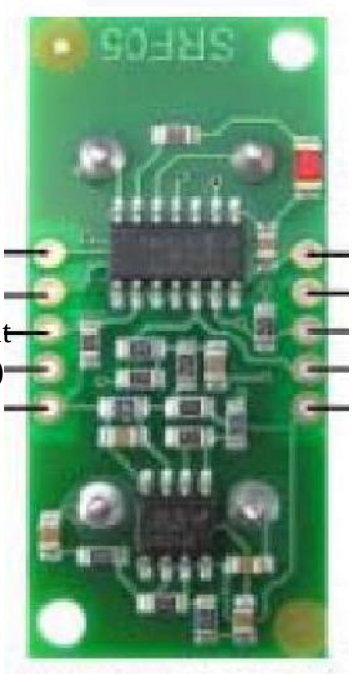
# Ultrasonic Rangefinder

Below is the SRF005 Ultrasonic Rangefinder.

Available at:
Rev-Ed: Order Code SRF005 £11.99
Microzed: Order Code SRF005 $42.00

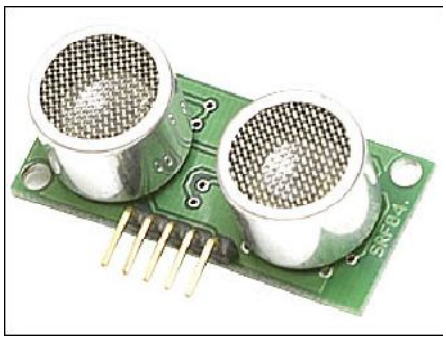
Minimum detection range 3cm, Maximum range 3m.

The rear of the Rangefinder -




5v Supply
To Picaxe Input
To Picaxe Output
(No Connection)
0v Ground

Do not connect to these pins



Ultrasonic Rangefinder

In this program the output pin of the Picaxe is 1 and the input pin is 2. Leave the download cable plugged into the Picaxe system and the debug window on the computer screen will show the variables, in this case W1 will show the distance in centimetres.

Range1

```
main:
pulsout 1,2
pulsin 2,1,w1
pause 10
let w1 = w1 * 10 / 58
debug w1
goto main
```

Combine that with the previous CF Sound board example and the Astromech can play a random sound when someone comes within range.

A more complex idea is to use two of these rangefinders and be able to turn the dome to face where someone is standing and the dome to follow them when they move.

In this program outputs 1 and 2 and inputs 1 and 2 are used for the Ultrasonic Rangefinder and a speed controller with motor to dome is connected to the output 3.

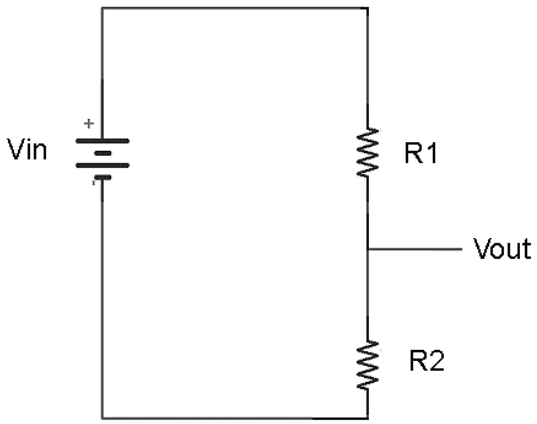
Range2

```
main:
pulsout 1,2
pulsin 1,1,w0
pause 10
pulsout 2,2
pulsin 2,1,w1
pause 10
let w0 = w0 * 10 / 58
let w1 = w1 * 10 / 58
if w0 < 300 then left
if w1 < 300 then right
servo 3,150
goto main
left:
servo 3,80
goto main
right:
servo 3,180
goto main
```

# Battery Voltage Monitoring

Battery voltage monitoring is someting the Picaxe is capable of. The Readadc or in otherwords anologue reading is available from input pins 0 and 1. As mentioned earlier you could damage your Picaxe if you get a reading much over the 5v the Pixace runs at. This means for your 12v batteries you need a voltage divider to lower the voltage to a level the picaxe can handle.

A Voltage Divider circuit looks like this:




The formula to work out what resistors you need is as follows:

$$Vout = Vin \times \frac{R2}{R1+R2}$$

A voltage divider calculator is also available at -

www.daycounter.com/Calculators/Voltage-Divider-Calculator.phtml

Or a program can be downloaded from -

www.yadtel.net/~ccp/potdesign/
(Use the simple divider calculation)

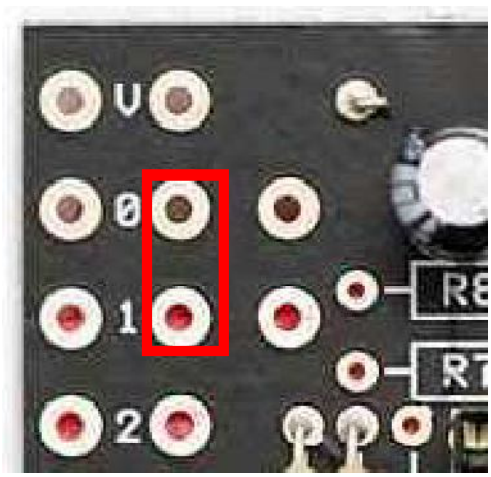
One thing to remember is that some fully charged batteries have a higher inital battery voltage than their rating, so for 12v batteries they may be over 13v even close to 14v. So be concervative in your calculations and use 14v.

For 14v the two resistors are as follows -

R1 = 18k & R2 = 10k

Just change for other voltages.

Closeup of Inputs 0 and 1 on the Picaxe 18X Board




Just connect the Vout to either Input 0 or input 1, you dont want the extra resistors soldered on to positions R7 & R8 for this application. As always the 0V from the battery and the 0v from the Picaxe has to be connected as well.

For this example, leave the download cable connected to receive the debug reading.

Voltage1

```
main:
Readac 1,b0
Debug b0
Goto main
```

The reading is not in volts though, 255 = high reading, 125 = mid reading, 0 = low reading (0V), for batteries they only drop a few volts so the numbers might only drop from 250 on a charged battery to 220 for a flat battery. High drain devices such as motors can make the reading unreliable so when checking all motors should be stationary, including servos.

Now you need to somehow display the reading. One ide is to use the RGB led's for you process state indicators, and when checking, they show by colour the current battery level, ie green for full charge, yellow for partially drained battery and red for low battery. I'm sure their are other ideas you could use, including playing a certain sound file when the battery is getting low, it is up to you.

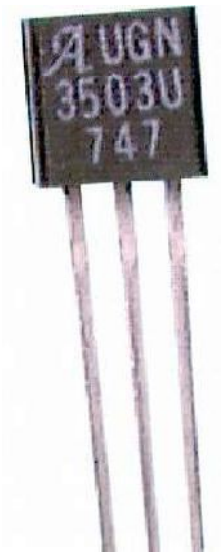
# Working Droid Restraining Bolt

**Warning:** The info here us more of a "idea" than a fully working example. I don't really know if it will work or even if it would damage your speed controller. It's just an idea I had that I though you might like.

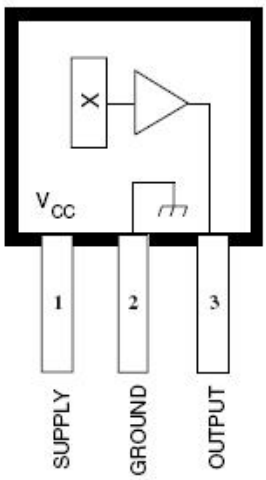
A lot of Restraining Bolts use a magnet to attach them to the door of the Astromech. In the past I have used a small Hall Effect sensor to detect magnetic fields, so put them together and you could have a working restraining bolt.

The Hall Effect Sensor is a UGN3503, very small the same size and shape as a small transistor and priced at around $4.00. I got mine in Australia at Jaycar (www.jaycar.com.au), but I think they are a reasonably common part.

Picture of the part:




Pin Wiring
(writing to front)




Supply is 5V, ground to a Picaxe ground, output goes to a Picaxe analoge input pin so either Input 0 or Input 1. Just attach the sensor to the inside of the door that the restraing bolt goes on.

**Note:** after a web search on restraining bolts, I came across a webpage of parts by Daniel Deutsch. It seems this is not a new idea, he seems to have had the same idea long before I thought of it.

For this Resistor positions R7 & R8 need to be left without the resistors.

This program is the same as the battery voltage monitor.

Bolt1

```
main:
Readac 1,b0
Debug b0
Goto main
```

The reading of b0 is a little bit different this time. With no magnet present the output should be in the middle of the range - around 127. If a south end a magnet is brought near, that reading will increase, and the north end of a magnet, the reading will go down.

So the idea is to attach an output of the Picaxe to a relay. The outputs of the relay are connected to the power of your speed controller. Obviously, large amp rating relays would be need, such as automotive horn relays. When the Restrinaing bolt is attached, the power to the speed controller is cut off, and so your Droid is not going anywere. Something to watch out for is other magnetic sources, such as the speaker for your sound system and motors. If they are too close it could interfere with this system.

Bolt2

```
main:
Readac 1,b0
if b0 > 200 then bolton
if b0 < 190 then boltoff
Goto main
bolton:
if b1 = 1 then goto main
low 1
b1 = 1
goto main
boltoff:
if b1 = 0 then goto main
high 1
b1 = 0
goto main
```