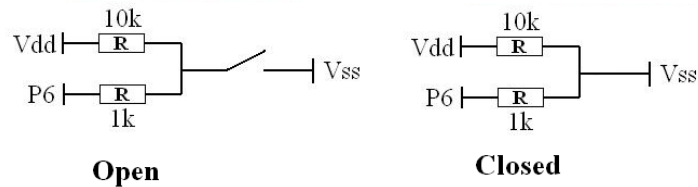


# Buttons

The next piece of hardware we will use is a button. These can be used to turn things on, or to tell your robot to do something more complex. The circuit we will be using to read the buttons state into the BS1 is pictured below. The trick when designing this circuit is to make sure the pin of the BS1 is connected to either 5V or ground at all times. Looking at the circuit you can see that when the button is not pressed, the circuit is open, which connects the pin to 5 volts. Yet when the button is pressed, the circuit is closed, and the pin is connected to ground. The resistors are placed in the circuit to once again, limit the current, since the I/O pins cannot handle too much power.



To read in the state of a pin, we need to set a previously declared variable equal to the pin. The variable will then either be a 1 for 5 volts, or 0 for ground. In the next program we will read on the state of the button, and display its state to the screen.

**SYMBOL** X = B0

Main:

X = PIN6

**DEBUG** CLS, X

**GOTO** Main

The next important thing to learn with the BS1 is *IF Statements*. This is what allows your robot to make decisions based on sensor input and other things. The If statements are formatted like the following:

**IF** *condition* **Then** *address*

The condition is where you compare a variable to a value, for example  $x = 0$ . When using the If statements, the condition can take the following comparators:

- = Equal
- <> Not equal
- < Less then
- > Greater then
- <= Less then or equal to
- >= Greater then or equal to

The address is a location in your program you want to run when the condition is true. The next program will read in the button, and when it is pressed, display the work

“pressed”, and when it is not, display “not pressed.” You will notice that in order for the If statements to work there must be a sub function named the same as the address. Then at the end of this sub function, you can use the Goto command to move back to the main program.

**SYMBOL X = B0**

Main:

**X = PIN6**

**IF X = 0 THEN Pressed**

**IF X = 1 THEN NotPressed**

**GOTO Main**

Pressed:

**DEBUG CLS, "Pressed"**

**GOTO Main**

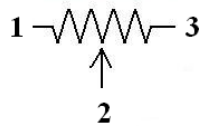
NotPressed:

**DEBUG CLS, "Not Pressed"**

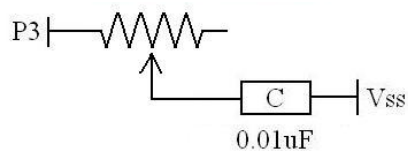
**GOTO Main**

## Variable Resistors

A variable resistor is exactly what the name infers. It is a device that has varying resistance when something is done too it. The most common variable resistor is a potentiometer. The resistance of the device changes when a knob is turned. When looking at a potentiometer, you will see three “legs.” The resistance between the leg 1 and 3 is fixed, and the resistance of 1 to 2 and 2 to 3 changes when the knob is turned.



The command we use to read these potentiometers in, is *POT*. This command was deigned specifically for measuring devices of varying resistance. The schematic we us is as follows:



Leg 1 of the potentiometer is connected to our input pin. The center leg of the potentiometer is connected to a capacitor and then to ground. Capacitors are used to store energy in the system to be used at later time. The Pot command is formatted as follows:

**POT Pin, Scale, variable**

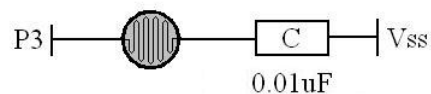
The pin expression is equivalent to the input pin in use. The scale is a number from 0 to 255 and based on the properties of the potentiometer and capacitor. The best way to find the correct scale to use is by experimentation. The variable expression is the variable you would like to store the results to. The result will be a value between 0 and 255 so a byte variable will suffice. The next program will read in the potentiometer and display its value to the screen.

**SYMBOL X = B0**

Main:

```
POT 3, 111, X  
DEBUG CLS, #X  
GOTO Main
```

Another type of variable resistor we will use very often is a photocell. This device's resistance changes when the lighting conditions change. The darker it is, the higher the resistance. So we can use a photocell like the potentiometer, hooking it up like the schematic shown below. Now if we run the program below you should see the darker you make it, the higher the number goes.



**SYMBOL X = B0**

Main:

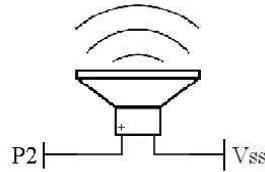
```
POT 4, 255, X  
DEBUG CLS, #X  
GOTO Main
```

## Sound

The next thing we will learn about is sound. The BS1 has a command designed specifically for this purpose name *Sound*. You can send out 128 different frequencies to a speaker. The sound command is formatted as below:

**SOUND** *pin, ( note , duration )*

The pin expression is simply the pin that is connected to the speaker. The note expression is a number from 0 to 255 corresponding to the note you would like to play. Numbers 0 to 127 are actual tones, and numbers 128 to 255 are white noise. The duration expression is how long you would like the note to play dimensioned in units of 12ms. We hook up the speaker like shown below. We can then run the program below which will beep repeatedly



**Main:**  
**SOUND 2, (120, 42)**  
**PAUSE 500**  
**GOTO Main**

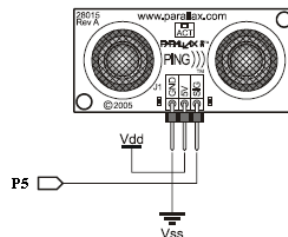
## Distance

The next thing we will learn is how to take distance measurements. To do this we use an ultrasound sensor. This sensor sends out an ultrasound ping (about 1130 ft/sec), it then waits until it hears it back then tells you how long it took, which we can equal to distance. The two main things you need to know to work with one of these sensors is how to trigger it (ask it for a reading), and how to receive information from it. To trigger it we simply send it a 5 volt pulse for 10  $\mu$ s. We can use our simple *pulsout* command for this. The sensor will then respond to us with another 5 volt pulse proportional to the distance. We can interpret this pulse using our *pulsin* command. The format is as follows:

**PULSIN** *pin, state, variable*

The pin expression is once again the pin in use. The state expression is the 1 or 0 telling the BS1 which type of pulse we are looking for, a 1 indicates we are looking for a 5 volt pulse, and 0 indicates we are looking for a 0 volt pulse.

The ultrasound sensor is hooked up exactly like a servo. 1 pin is connected to 5 volts, 1 pin is connected to ground, and the last is connected to an I/O pin. Using the following schematic, we can then run the program below to display distance measurements in inches.



**SYMBOL X = W0**

**Main:**  
**PULSOUT 5, 1**  
**PULSIN 5, 1, X**  
**X = X/15**  
**DEBUG CLS, #X**  
**GOTO Main**