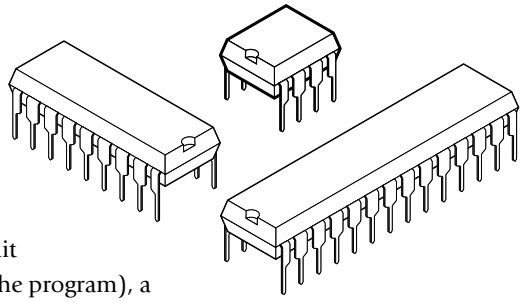
ALARM SYSTEMS

What is a microcontroller?

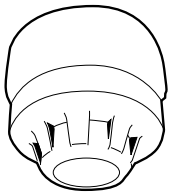


A microcontroller is often described as a 'computer-on-a-chip'. It can be used as an 'electronic brain' to control a product, toy or machine.

The microcontroller is an integrated circuit ("chip") that contains memory (to store the program), a processor (to process and carry out the program) and input/ output pins (to connect switches, sensors and output devices like motors).

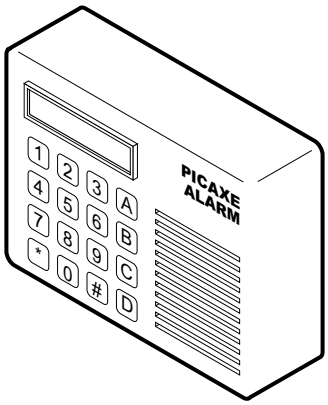
Microcontrollers are purchased 'blank' and then programmed with a specific control program. This program is written on a computer and then 'downloaded' into the microcontroller chip. Once programmed the microcontroller is built into a product to make the product more intelligent and easier to use.

Example use of a microcontroller.



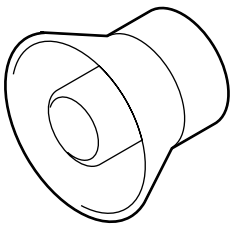
Almost all modern buildings are fitted with some type of alarm. For instance a fire detection system may have a number of smoke sensors to detect the smoke from a fire.

However many alarm systems are also safety systems - for instance an alarm system on an oil rig may monitor the temperature and pressure of the crude oil as it is being extracted and automatically shut the system down if a fault is detected. This ensures the safety of both the workers and the environment around the oil rig.



All systems are made up of input and output devices. Often these devices are connected to a microcontroller that interprets the information from the sensors and switches the outputs on and off at the correct time.

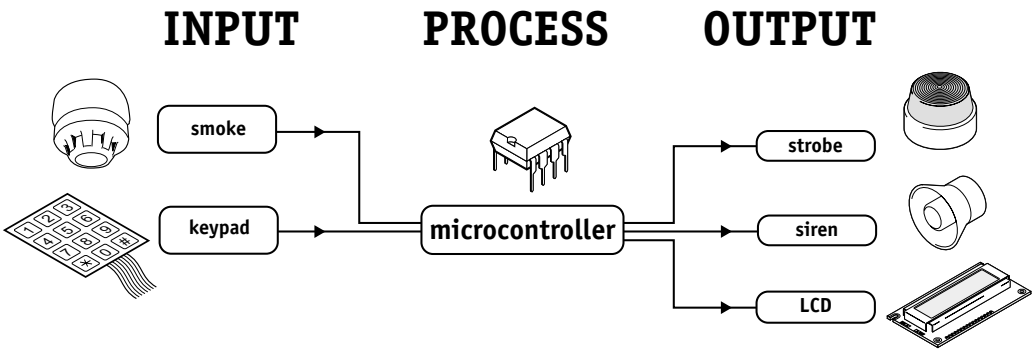
In the case of a fire alarm system the inputs may be smoke sensors and the keypad on the front of the control panel. The output devices are the display on the control panel as well as the external siren and strobe light. The microcontroller is the 'brain' of the system.



Microcontrollers are powerful electronic components that have a memory and can be programmed to switch things on and off in a special sequence. The microcontroller in the fire alarm, for instance, has been programmed to switcyh the siren on and off when the smoke sensor has detected fire.

Block Diagrams

The electronic system that makes up the alarm system be drawn as a 'block diagram'.



The smoke sensor and keypad provide information to the microcontroller, and so these are known as 'inputs'. The microcontroller then 'decides' how to behave and may then operate the outputs e.g. make the siren and strobe switch on or display a message on the Liquid Crystal Display (LCD).

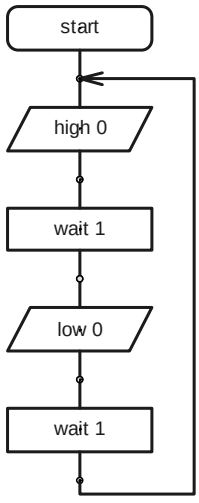
What is the PICAXE System?

The microcontrollers used in devices such as the alarm can be difficult to program, as they generally use a complicated programming language called 'assembler code', which can be quite difficult to learn.

The PICAXE system makes the microcontrollers much easier to program. The control sequence can be drawn (and simulated) on the computer as a flowchart, or written in a simpler programming language called BASIC. This makes it much easier to use the microcontroller as the complicated 'assembler code' does not need to be learnt.

A sample BASIC program and flowchart are shown here. In this case both programs do the same thing - flash a light (connected to output 0) on and off every second.

```
main:
        high 0
        wait 1
        low 0
        wait 1
        goto main
```



Building your own Alarm

Design Brief

Design and make an alarm system. The alarm must be programmed to react to inputs and sensors.

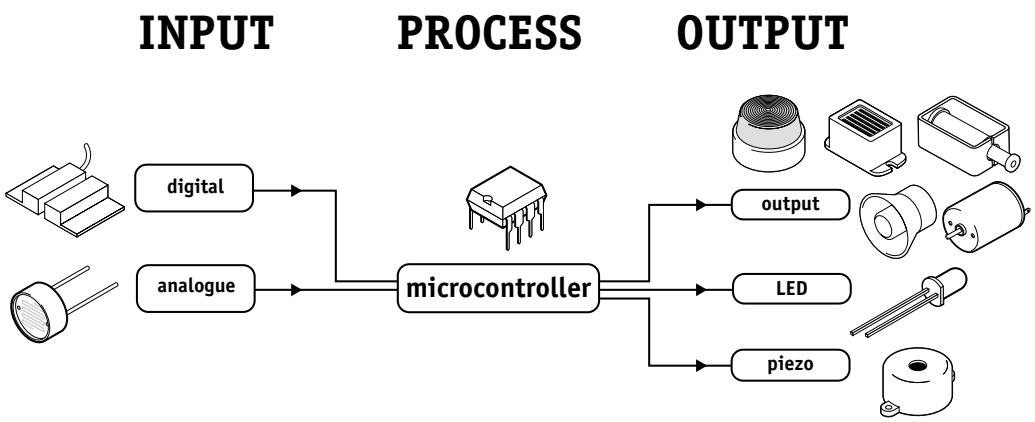
Design Specification Points

1) The design will use a PICAXE-08 microcontroller as it's brain.
2) The design will include an LED indicator, a piezo sounder to generate noises and an alarm output that could be a siren or motor..
3) The design will be also be able to react to analogue sensors such as light sensors.

.

Block Diagram

The block diagram for your alarm may look like this:



Designing Your Alarm

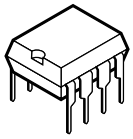
Your alarm may be for any purpose you choose. Some examples are given below.

1) A fire alarm. A light sensor is used to detect smoke. When smoke is detected a siren sounds.
2) A burglar alarm. When a tripwire is activated a flashing stobe light is activated. However the alarm is disabled during daylight by a light sensor.
3) A bank safety lock. When a 'panic' alarm switch is pushed an electronic solenoid bolt locks a bank safe door.
4) A baby monitor for a bedroom. When movement or sounds are NOT detected, a warning buzzer is activated.

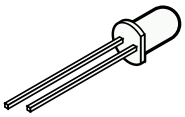
Electronic Components

The main electronic components you may need for your alarm are shown here. The next few pages describe each of these components in more detail, and also provide some programming ideas that may be useful when you are later programming your alarm.

PICAXE-08 microcontroller

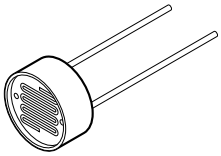


light emitting diode (LED)

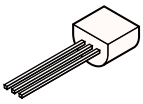


piezo sounder

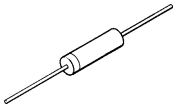
light dependent resistor (LDR)



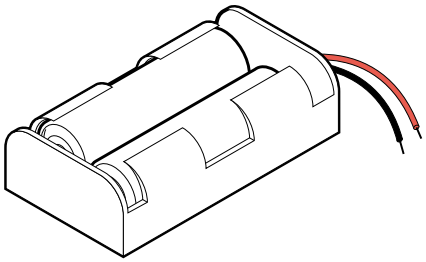
transistor



and diode

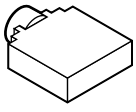


batteries

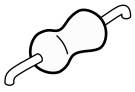


and you will also need

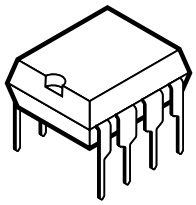
picaxe download socket



resistors



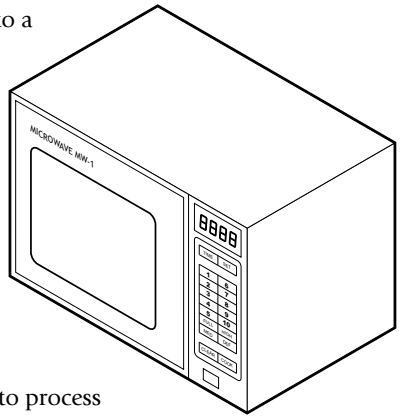
Section 2 - Electronic Components
Microcontrollers



What is a microcontroller?

A microcontroller is often described as a 'computer-on-a-chip'. It is an integrated circuit that contains memory, processing units, and input/ output circuitry in a single unit.

Microcontrollers are purchased 'blank' and then programmed with a specific control program. Once programmed the microcontroller is built into a product to make the product more intelligent and easier to use.



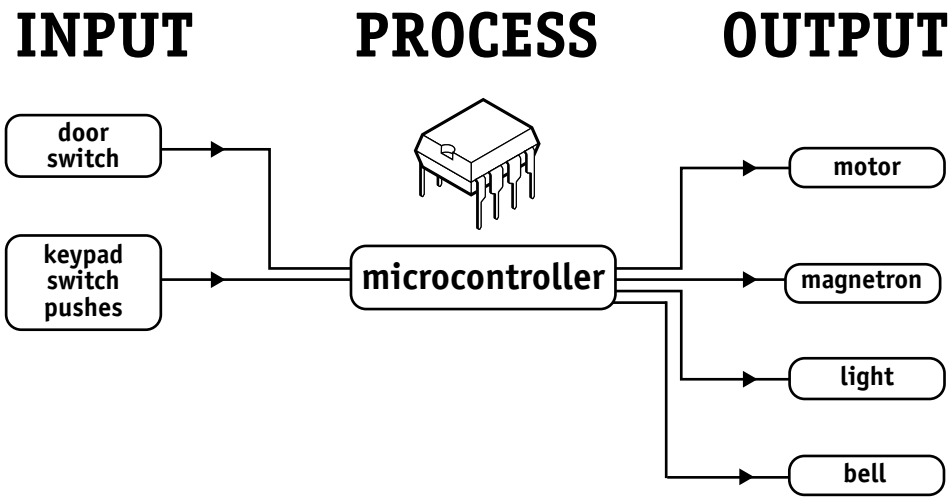
Where are microcontrollers used?

Applications that use microcontrollers include household appliances, alarm systems, medical equipment, vehicle subsystems, and electronic instrumentation. Some modern cars contain over thirty microcontrollers - used in a range of subsystems from engine management to remote locking!

As an example, a microwave oven may use a single microcontroller to process information from the keypad, display user information on the seven segment display, and control the output devices (turntable motor, light, bell and magnetron).

How are microcontrollers used?

Microcontrollers are used as the 'brain' in electronic circuits. These electronic circuits are often drawn visually as a 'block diagram'. For instance a simplified block diagram for the microwave above could be drawn like this:



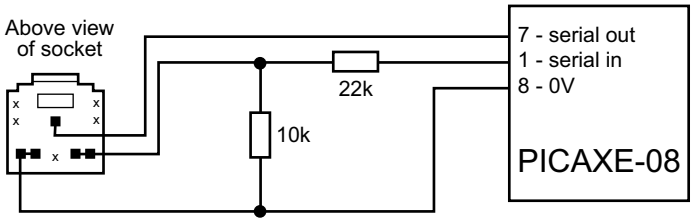
The program for the microcontroller is developed (and tested) on the computer and then downloaded into the microcontroller. Once the program is in the microcontroller it starts to 'run' and carries out the instructions.

How are programs written?

Programs are drawn as flowcharts or typed as 'BASIC' listings. This is is explained in the programming section (section 3) later in this booklet.

How is the program transferred to the microcontroller?

The PICAXE-08 microcontroller is programmed by connecting a cable from the serial port at the back of the computer to a socket on the printed circuit board (PCB) beside the microcontroller. This socket (which looks like a headphone socket as found on a portable CD player) connects to two legs of the microcontroller and to 0V from the battery. This allows the computer and the microcontroller to 'talk' to allow a new program to be downloaded into the microcontroller's memory.



The socket and interfacing circuit is included on every PCB designed to be used with the PICAXE-08 microcontroller. This enables the PICAXE microcontroller to be re-programmed without removing the chip from the PCB - simply connect the cable whenever you want to download a new program!

The circuit diagrams of PICAXE circuits often do not include the components above to make it easier to understand the input/output connections. However the two resistors and the socket are always built onto every PICAXE project board!

Output 0

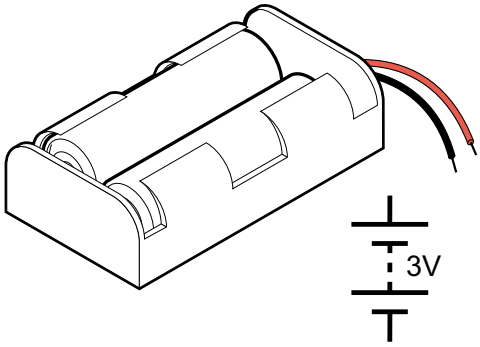
With the PICAXE-08 system leg 7 has two functions - when a program is being run the leg is known as output 0 and can control outputs like LEDs and motors.
When a program is being downloaded the same leg acts as the 'serial out' pin, 'talking' to the computer. Therefore if you also have an output such as an LED connected to the leg, you will find that the LED will flicker on and off as the program download takes place.

Note:
Most modern computers have two serial ports, normally labelled COM1 and COM2. The Programming Editor software used to create the programs must be configured for the correct serial port – select **View>Options>Serial Port** to select the correct serial port for your machine.

If you are using a new laptop computer it may only have the newer 'USB' type connector. In this case you must buy a USB to serial adapter to use the PICAXE system. These are available from most high street computer stores or online from www.tech-supplies.co.uk (part USB010).

BATTERIES

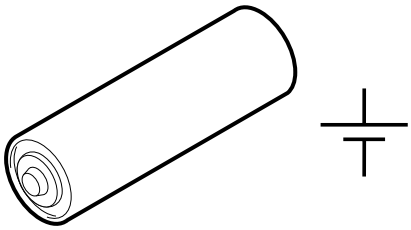
What is a battery?



A battery is a self-contained source of electronic energy. It is a portable power supply.
Batteries contain chemicals that store energy. When connected into a circuit this chemical energy is converted to electrical energy that can then power the circuit.

Which battery size should I use?

Batteries come in all sorts of types and sizes. Most battery packs are made up of a number of 'cells', and each cell provides about 1.5V. Therefore 4 cells will generate a 6V battery and 3 cells a 4.5V battery.



As a general rule, the larger the battery the longer it will last (as it contains more chemicals and so will be able to convert more energy). A higher voltage battery does not last longer than a lower voltage battery. Therefore a 6V battery pack made up of 4 AA cells will last much longer than a 9V PP3 battery, as it contains a larger total amount of chemical energy as it is physically larger. Therefore items that require more power to work (e.g. a CD walkman which contains a motor and laser to read the CD's) will always use AA cells rather than PP3 batteries.

Microcontrollers generally require 3 to 6V to work, and so it is better to use a battery pack made up of three or four AA size cells. Never use a 9V PP3 battery as the 9V supply will damage the microcontroller.

Which battery type should I use?

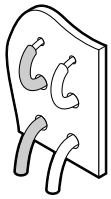
Different batteries are made of different chemicals. Zinc-carbon batteries are the cheapest, and are quite suitable for many microcontroller circuits. Alkaline batteries are more expensive, but will last much longer when driving devices like motors that require larger currents. Lithium batteries are much more expensive but have a long life, and so are commonly used in computer circuits to provide a clock backup.

Rechargeable batteries can be recharged when they 'run-down'. They are generally made up of nickel and cadmium (Ni-cad) or nickel metal hydroxide (NiMH) chemicals.

Safety!

Never 'short circuit' any battery. Alkaline and rechargeable batteries can provide a very large current, and can get so hot that they will actually melt the battery box if you short circuit them! Always make sure you connect the battery around the correct way (red positive (V+) and black negative (0V or ground)). The microcontroller chip will get hot and be damaged if the battery is connected the wrong way around.

Using battery snaps.



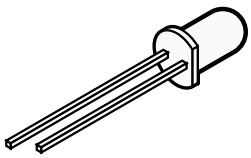
Battery packs are often connected to electronic printed circuit boards by battery snaps. Always ensure you get the red and black wires the correct way around. It is also useful to thread the battery snap through holes on the board before soldering it in place - this provides a much stronger joint that is less likely to snap off.

Never accidentally connect a 9V PP3 battery to the battery snap - this will damage the microcontroller, which only works between 3 and 6V.

Soldering to battery boxes.

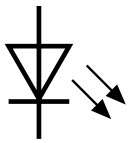
Some small battery boxes require wires to be soldered to metal contacts on the battery box. In this case you must be very careful not to overheat the metal contact. If the contacts gets very hot they will melt the plastic and fall off. A good way of stopping this happening is to ask a friend to hold the metal contact with a pair of small pliers. The pliers will act as a 'heat-sink' and help stop the plastic melting.

Light Emitting Diode (LED)



What is an LED?

A Light Emitting Diode (LED) is an electronic component that gives out light when current passes through it. An LED is a special type of diode. A diode is a component that only allows current to flow in one direction. Therefore when using a diode, it must always be connected the correct way around.



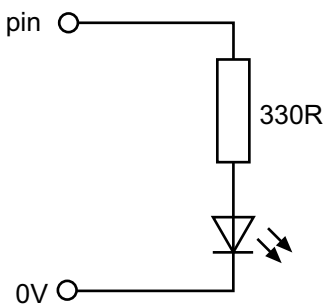
The positive (anode) leg of an LED is longer than the negative (cathode) leg (shown by the bar on the symbol). The negative leg also has a flat edge on the plastic casing of the LED.

What are LEDs used for?

LEDs are mainly used as indicator lights. Red and green LEDs are commonly used on electronic appliances like televisions to show if they are switched on or in 'standby' mode. LEDs are available in many different colours, including red, yellow, green and blue. Special 'ultrabright' LEDs are used in safety warning devices such as the 'flashing lights' used on bicycles. Infra-red LEDs produce infra-red light that cannot be seen by the human eye but can be used in devices such as video remote-controls.

Using LEDs.

LEDs only require a small amount of current to work, which makes them much more efficient than bulbs (this means, for instance, that if powered by batteries the LEDs will light for a much longer time than a bulb would). If too much current is passed through an LED it will be damaged, and so LEDs are normally used together with a 'series' resistor that protects the LED from too much current.



The value of the resistor required depends on the battery voltage used. For a 4.5V battery pack a 330R resistor can be used, and for a 3V battery pack a 120R resistor is appropriate.

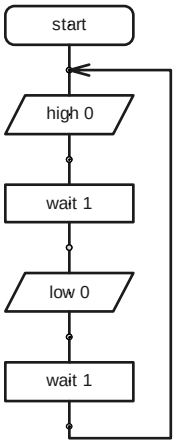
Connecting the LED to a microcontroller.

Because the LED only requires a small amount of current to operate, it can be directly connected between the microcontroller output pin and 0V (with the series protection resistor).

Testing the LED connection.

After connecting the LED it can be tested by a simple program like this:

```
main:
        high 0
        wait 1
        low 0
        wait 1
        goto main
```



This program would switch the LED (connected to output pin 0) on and off every second. If the LED does not work check:

1) the LED is connected the correct way around
2) the correct resistor is used
3) the correct output pin number is being used in the program
4) all the solder joints are good

This program flashes the LED connected to output pin 0 on and off 15 times using a BASIC programming technique called a for...next loop (this technique cannot be used with flowcharts). The number of times the code has been repeated is stored in the memory of the PICAXE chip using a 'variable' called b1 (the PICAXE contains 14 variables labelled b0 to b13). A variable is a 'number storage position' inside the microcontroller than the microcontroller can use to store numbers as the program is carried out.

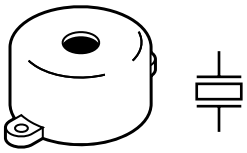
```
main:   for b1 = 1 to 15        ' start a for...next loop
                high 0          ' switch pin 0 high
                pause 500       ' wait for 0.5 second
                low 0           ' switch pin 0 low
                pause 500       ' wait for 0.5 second
        next b1                 ' end of for...next loop

        end                     ' end program
```

Buzzers and Piezo-Transducers

What is a piezo transducer?

A piezo transducer is a low-cost 'mini-speaker' that can used to make sounds. The sound that the piezo makes can be changed by altering the electronic signals provided by the microcontroller.

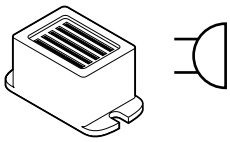


Where are piezos used for?

Piezos are used in many different consumer goods to provide 'feedback' to the user. A good example is a vending machine which will 'beep' whenever a keypad switch is pressed to select a drink or snack. The 'beep' provides the user with feedback to tell them their switch push has been successful. Uncased piezos are also often used in musical birthday cards to play a tune when the card is opened.

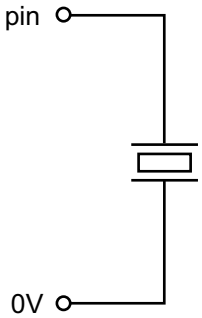
What is the difference between a piezo and a buzzer?

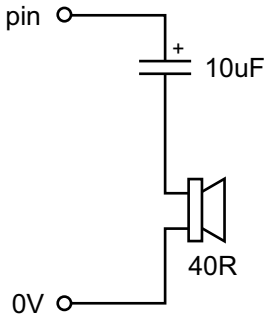
A buzzer contains a small electronic circuit that generates the electronic signal needed to make a noise. Therefore when a buzzer is connected to a battery it will always make the same sound. A piezo does not contain this circuit, and so therefore needs an external signal. This signal can be supplied by the output pin of a microcontroller. A piezo also requires less current to operate and so will last longer in battery powered circuits.



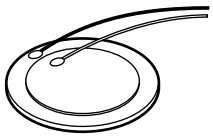
Using piezos.

A piezo is very simple to connect. Simply connect the red wire to the microcontroller output pin and the black wire to 0V (ground).



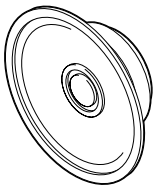


Note that the cheapest piezos do not have a plastic casing to them. In this case it is necessary to mount the piezo on a piece of board (with a sticky pad) to create a noise that can be heard. The board acts as a 'sound-box' to amplify the sound made by the piezo. Make sure the sticky pad is stuck on the correct side of the piezo (the brass side without the wires!).



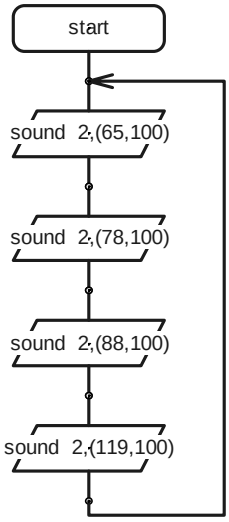
Making More Noise.

Some times you might want to make a louder noise. In this case it is possible to use a speaker instead of the piezo. When using a speaker it is also necessary to use a capacitor (e.g. 10uF electrolytic capacitor) to prevent damage to the microcontroller.
Remember that, like the piezo, a speaker only works correctly when mounted in a 'sound-box'.



Testing the piezo connection.

After connecting the piezo it can be tested by a simple program like this:




```
main:
        sound 2,(65,100)
        sound 2,(78,100)
        sound 2,(88,100)
        sound 2,(119,100)
        goto main
```

This program would make the piezo (connected to output pin 2) make 4 different sounds (value 65, 78, 88 and 119). If the piezo does not work check:

1) the sound value is between 0 and 127
2) the correct output pin number is being used in the program
3) all the solder joints are good

With the sound command, the first number provides the pin number (on projects pin2 is often used). The next number is the tone, followed by the duration. The higher the tone, the higher the pitch of the sound (note that some sounders cannot produce very high tones and so number greater than 127 may not be heard).

When using multiple sounds you may also include them all on the same line e.g.

```
sound 2,(65,100,78,100,88,100,119,100)
```

The following table shows some common notes with the appropriate sound value:
A(49), As(51), B(54), C(57), Cs(61), D(65), Ds(71), E(78), F(88), Fs(101), G(119)

The following BASIC program uses a for…next loop to produce 120 different sounds, using variable b1 to store the sound value.

```
main:
        for b1 = 1 to 120       ' start a for...next loop
           sound 2,(b1,50)      ' make a sound, freq value from b1
        next b1                 ' next loop
        end
```

The number stored in variable b1 is increased by 1 in every loop (1-2-3 etc.) Therefore by using the variable name b1 in the tone position, the note can be changed on each loop.

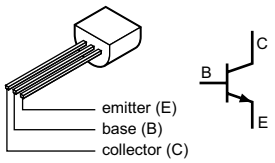
The following program does the same task but backwards (counting down instead of up).

```
main:
        for b1 = 120 to 1 step -1 ' start a for...next loop
           sound 2,(b1,50)        ' make a sound, freq value from b1
        next b1                   ' next loop
        end
```

TRANSISTOR

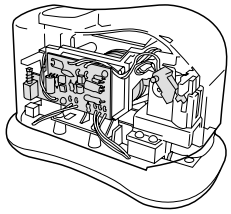
What is a transistor?

A transistor is a component that controls current flow in a circuit. A transistor acts as an 'electronic switch' so that a small current can control a large current. This allows low-current devices, like a microcontroller, to control large current devices (like motors).



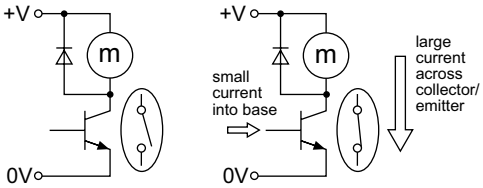
Where are transistors used?

Transistors are used in radios and electronic toys and many other devices such as this electronic stapler.



Using transistors.

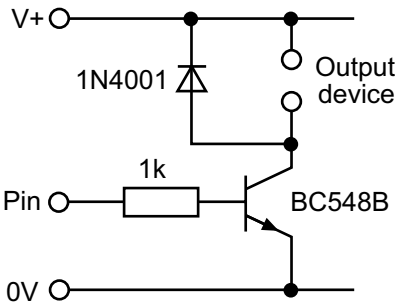
A transistor has three legs. These are labelled base, collector and emitter. The base connection is the leg that is used to activate the 'electronic switch'. When a small current is passed through the base connection, it allows a much larger current to flow down between the collector and emitter. This larger current can be used to switch on devices such as motors, lamps and buzzers.



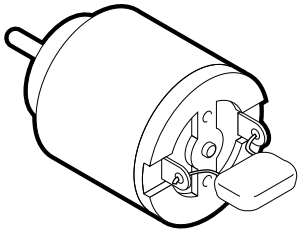
A common transistor is the BC548B type. This has a plastic can with a flat edge. The flat edge enables the base, collector and emitter legs to be correctly identified.

Using motors.

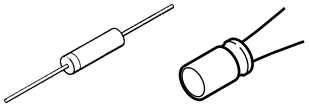
Motors can generate 'electrical noise' as they turn. This is because the magnets and electric coils inside the motor generate electrical signals as the motor rotates. These signals (the electronic noise) can disrupt the operation of the microcontroller. Some motors, like solar motors, produce very little noise whilst others can create a lot of noise.



To prevent electrical noise affecting the micro-controller circuit a 220nF capacitor should always be soldered across the motor terminals before it is used.

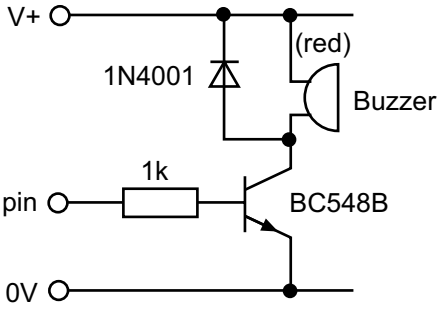


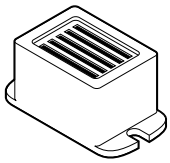
In addition, a diode (e.g. 1N4001 diode) should be connected alongside the motor. This prevents damage to the transistor as the motor slows down after the transistor switches off (for a short period of time (as it slows down and stops) the motor is acting as a 'dynamo' and generating electric current!). When connecting a diode make sure the 'band' is connected the correct way around. It is also a good idea to connect a 100uF electrolytic capacitor across the battery supply to help 'suppress' the the electrical noise.



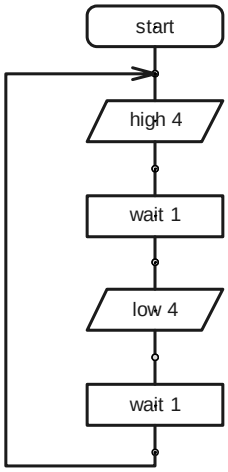
Testing the transistor using a buzzer

A buzzer could be connected as an output device as in this cicruit.





After connecting the buzzer it can be tested by a simple program like this:



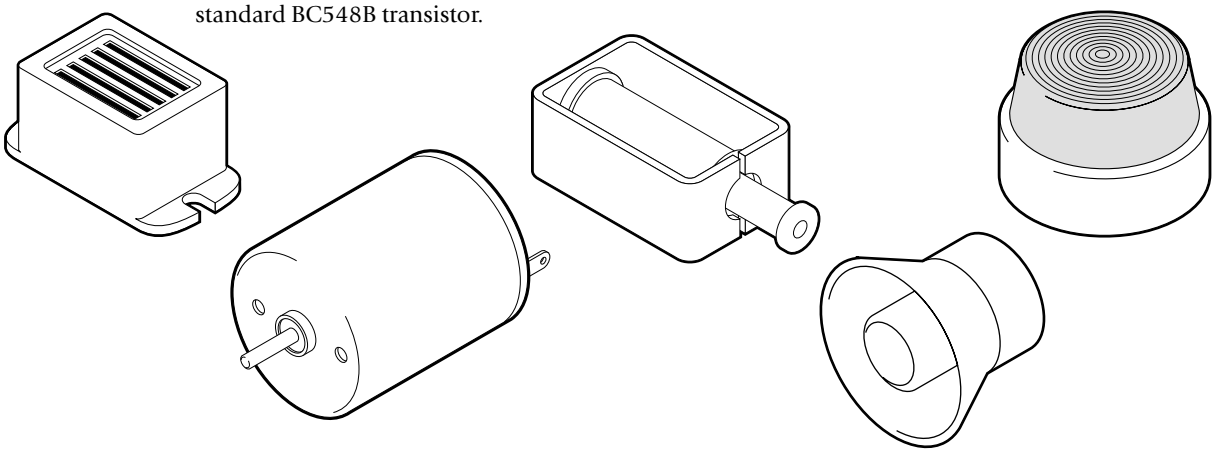

```
main:
        high 4
        wait 1
        low 4
        wait 1
        goto main
```

This program would switch the buzzer (connected to output pin 4) on and off every second. If the buzzer does not sound check that:

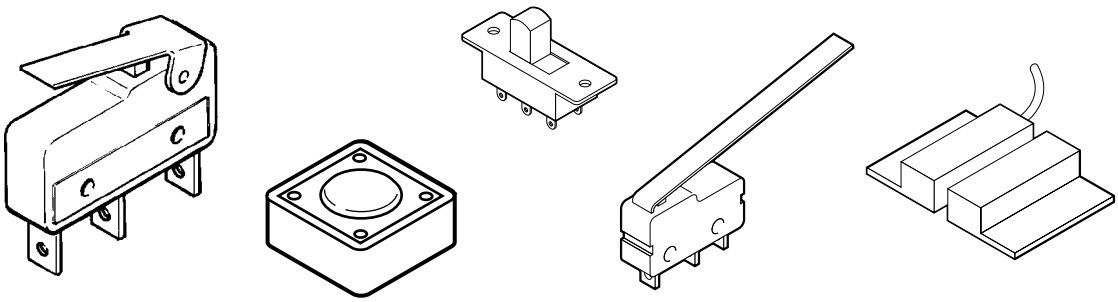
1) the diode is connected the correct way around
2) the correct resistors are used
3) the transistor is connected the correct way around
4) the buzzer red wire is connected the correct way around
5) the correct output pin number is being used in the program
6) all the solder joints are good

Output Devices

Output devices that can be connected via a transistor include buzzers, motors, solenoids, sirens and flashing strobe lights. However some devices may need a higher power transistor - in this case the Darlington transistor BCX38B can be used instead of the standard BC548B transistor.

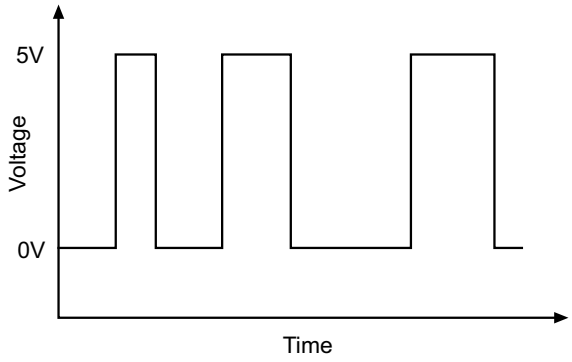


Digital Sensors (Switches)



What are switches?

A digital sensor is a simple 'switch' type sensor that can only be 'on' or 'off'. If a graph is drawn of the on-off signals as the switch is pushed it will look like this:



Switches are electronic components that detect movement. There are a large number of different types of switches e.g:

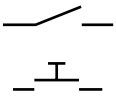
push switches that detect a momentary 'push'
micro-switches with long levers that detect small movements
tilt-switches that detect jolting
reed-switches that detect a magnet being moved

What are switches used for?

Push switches are commonly used on device like keypads. Micro-switches are used in burglar alarms to detect if the cover is removed from the alarm box. Reed switches are used to detect doors and windows being opened and tilt switches are often used to detect movement in devices such as toys, hair-dryers and tool-box alarms.

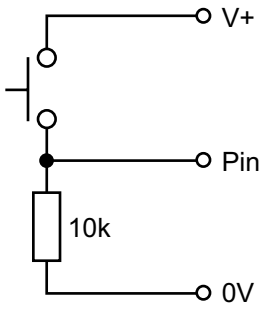
Switch Symbols.

The symbols for a slide switch and a push switch are shown here.



Using switches

A switch is used with a resistor as shown in the diagram. The value of the resistor is not that important, but a 10k resistor is often used. When the switch is 'open' the 10k resistor connects the microcontroller input pin down to 0V, which gives an off (logic level 0) 0V signal to the microcontroller input pin.



When the switch is activated, the input pin is connected to the positive battery supply (V+). This provides an on (logic level 1) signal to the microcontroller.

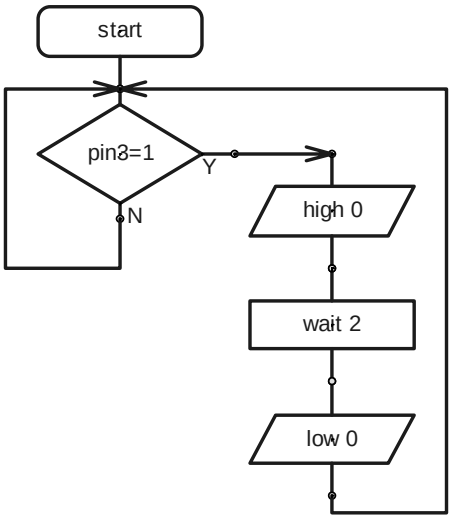
Testing the switch

After connecting the switch it can be tested by a simple program like this. This program will switch an output on and off according to if the switch is pushed or not.

```
main:                               ' make a label called 'main'
       if input3 is on then flash   ' jump if the input is on
       goto main                    ' else loop back around

flash:                              ' make a label called 'flash'
       high 0                       ' switch output 0 on
       wait 2                       ' wait 2 seconds
       low 0                        ' switch output 0 off
       goto main                    ' jump back to start
```

In this program the first three lines make up a continuous loop. If the input is off the program just loops around time and time again.



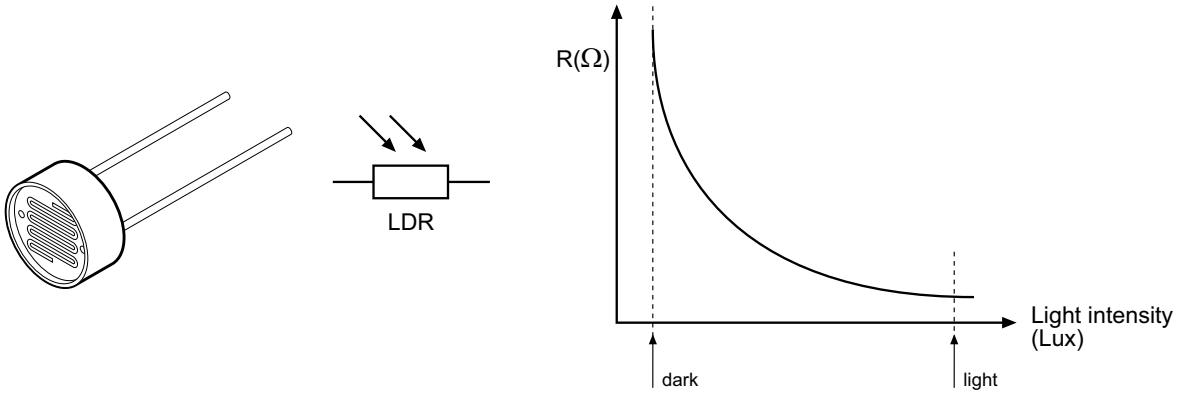
If the switch is then pushed the program jumps to the label called **'flash'**. The program then flashes output 0 on for two seconds before returning to the main loop.

Note carefully the spelling in the **if…then** line – **input3** is all one word (without a space). You can use the word **pin3** or **input3** to mean the same thing. Note also that only the label is placed after the command **then** – no other words apart from a label are allowed at this point.

Light Dependent Resistor (LDR)

What is an LDR?

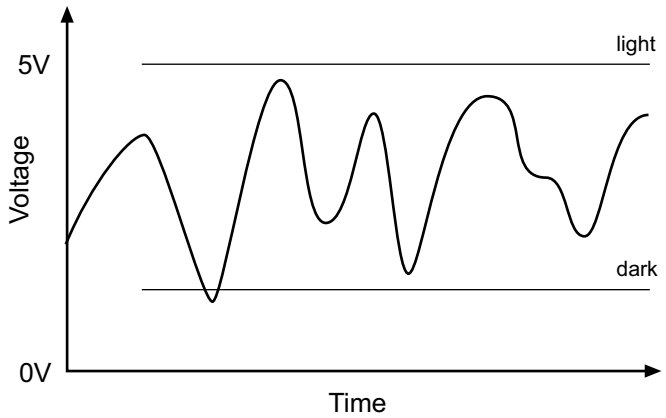
A Light Dependent Resistor (LDR) is special type of resistor that reacts to changes in light level. The resistance of the LDR changes as different amounts of light fall on the top 'window' of the device. This allows electronic circuits to measure changes in light level.



What are LDRs used for?

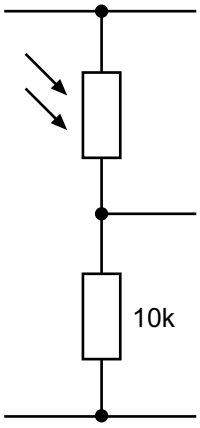
LDRs are used in automatic street lamps to switch them on at night and off during the day. They are also used within many alarm and toys to measure light levels.

The LDR is a type of **analogue sensor**. An analogue sensor measures a continuous signal such as light, temperature or position (rather than a digital on-off signal like a switch). The analogue sensor provides a varying voltage signal. This voltage signal can be represented by a number in the range 0 and 255 (e.g. very dark = 0, bright light = 255).



Using LDRs.

A LDR can be used in two ways. The simplest way to use an LDR is as a simple on-off ("digital") switch - when the light level is above a certain value (called the 'threshold value') the LDR will provide an on signal, when the light level is below a certain value the LDR will provide an off signal.



In this case the LDR is used in a potential divider with a standard resistor. The value of the standard resistor sets the 'threshold value'. For miniature LDRs a suitable value is 10k or 1k, for larger ORP12 type LDRs 10k is more appropriate. If desired the fixed resistor can be replaced by a variable resistor so that the threshold value can be 'tuned' to different light values.

A more versatile way of using the LDR is to measure a number of different light values, so that decisions can be made at varying light levels rather than just one fixed threshold value. A varying value is known as an 'analogue' value, rather than a digital 'on-off' value. To measure analogue values the microcontroller must contain an 'analogue to digital converter (ADC)' and the programming software must support use of this ADC. Most microcontrollers only contain ADC on certain input pins, and so the input pin connection must be carefully selected. With the 8 pin microcontroller only pin1 can be used.

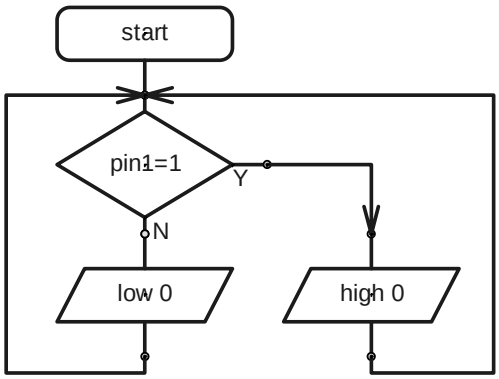
The electronic circuit for using the ADC is a potential divider identical to the circuit above. The analogue 'measurement' is carried out within the microcontroller itself.

Testing the LDR (digital)

After connecting the LDR it can be tested as a digital switch by a simple program like this:

```
main:
        if input1 is on then dohigh
        low 0
        goto main

dohigh:
        high 0
        goto main
```



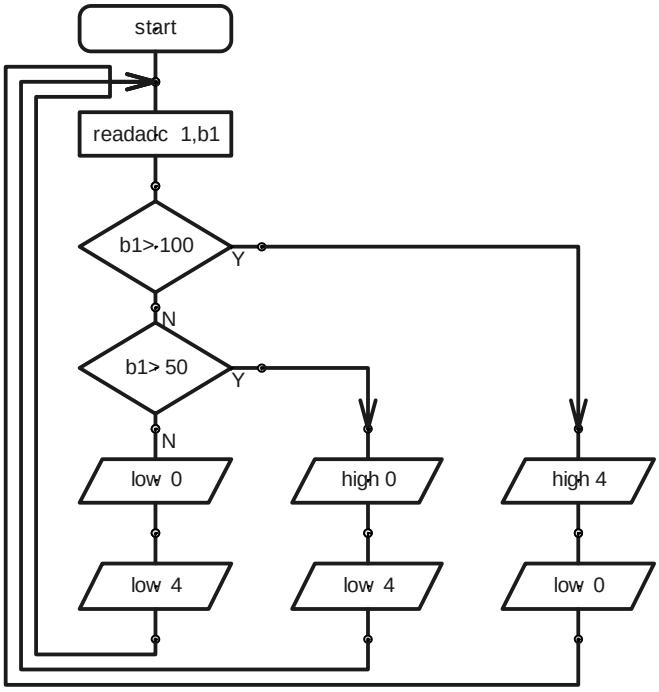
This program will switch output 0 on and off according to the light level.

Testing the LDR (analogue)

After connecting the LDR it can be tested as an analogue sensor by a program like this:

```
main:
        readadc 1,b1
        if b1 > 100 then do4
        if b1 > 50 then do0
        low 0
        low 4
        goto main

do4:
        high 4
        low 0
        goto main
do0:
        high 0
        low 4
        goto main
```



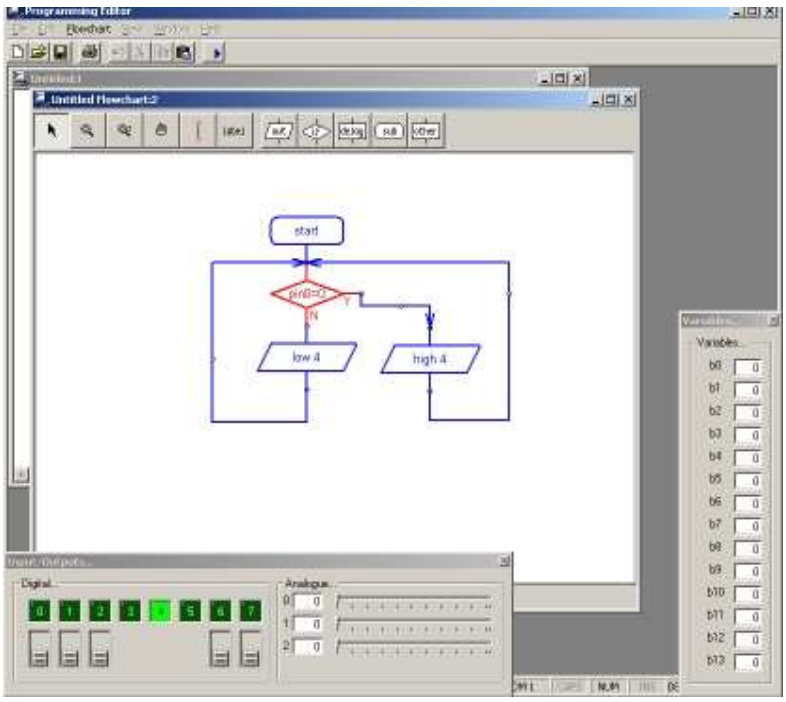
The 'readadc' command is used to read the analogue value (a number between 0 and 255) into variable b1. Once the number is in variable b1 it can be tested to see if it is greater than 100 or greater than 50. If it is greater than 100 output 4 is switched on, if it is between 50 and 100 output 0 is switched on, and if it is less than 50 both outputs are switched off.

SECTION 3
PROGRAMMING - DRAWING FLOWCHARTS

Flowcharts are a useful tool that allow programs to be drawn graphically to make them easier to understand. The Programming Editor software includes a flowchart editor that allows flowcharts to be drawn on screen. These flowcharts can then be converted to BASIC listings for download into the PICAXE. The flowcharts can also be printed or exported as graphics files for inclusion within project portfolios.

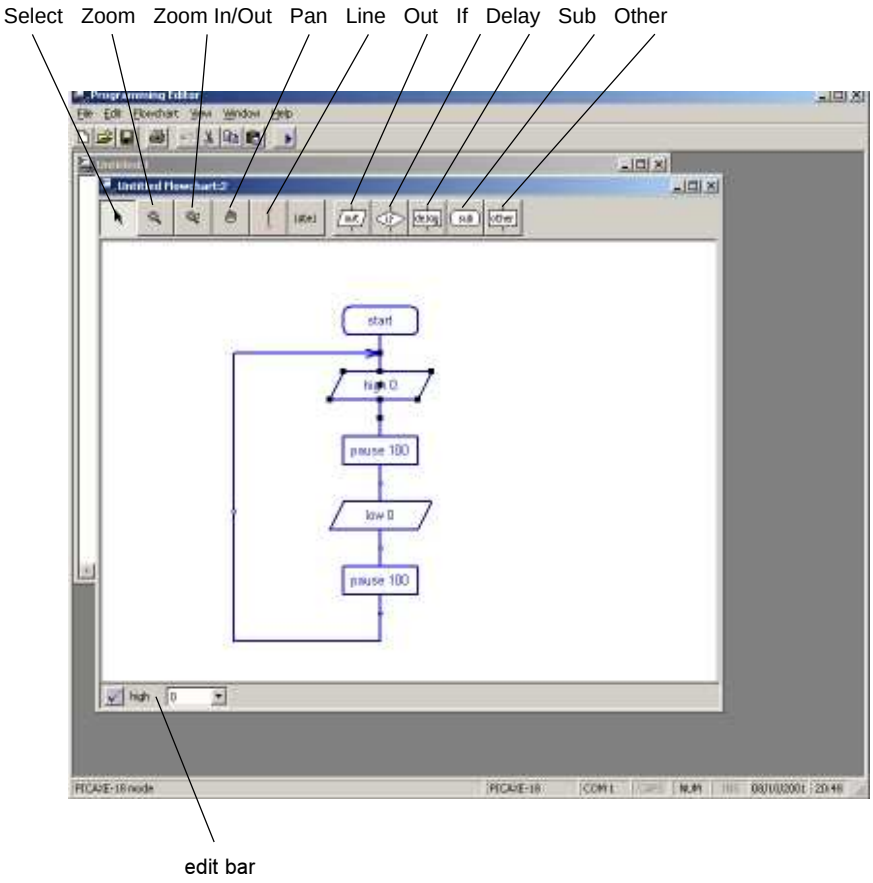
Detailed instructions for drawing/downloading a flowchart:

1. Connect the PICAXE cable to the computer serial port. Note which port it is connected to (normally labelled COM1 or COM2).
2. Start the Programming Editor software.
3. Select View>Options to select the Options screen (this may automatically appear).
4. Click on the 'Mode' tab and select PICAXE-08
5. Click on the 'Serial Port' tab and select the serial port that the PICAXE cable is connected to. Click 'OK'
6. Start a new flowchart by clicking the File>New Flowchart menu.
7. Draw the flowchart by dragging the correct symbols onto the screen, and then using the mouse to draw arrows between the symbols.
8. Once the flowchart is complete it can be converted into a BASIC program by selecting Flowchart>Convert Flowchart to BASIC. The BASIC program can then be downloaded into the PICAXE by clicking the PICAXE>Run menu.
9. To print or save the flowchart, use the File menu options. To export the flowchart as a graphic file, use the File>Export menu. To publish the image in a Word document select file type EMF. To publish the flowchart on an internet web page use the GIF file type.



Flowchart Screen

The Flowchart Editor allows flowcharts to be drawn and simulated on-screen. The flowchart can then be automatically converted into a BASIC program for downloading into the microcontroller.



Flowchart Screen

Select Tool
Use this to select and move shapes. When a single shape is selected it's BASIC code can be edited in the edit bar at the bottom of the window.

Zoom
Use to zoom in to an area of the graph. Right click to zoom out.

Zoom In/Out
To zoom in click and move the mouse up. To zoom out click and move the mouse down.

Pan
Use this tool to move around the flowchart.

Line Tool
Use this tool to draw lines between shapes. Corners can be added by clicking once. When the line is near to a shape it will 'snap' to the connection point.

Label Tool
Use this tool to add descriptive labels or titles to the flowchart.

Out / If / Delay / Sub / Other
Click on these buttons to move to the command sub-menu to select commands.

Drawing Flowcharts

To draw a flowchart click on one of the command menu buttons (out / if / delay / sub / other) on the toolbar to move to the appropriate command sub-menu. Select the appropriate command and then click on the screen where the shape is required. Do not try to locate the shape precisely at first – just drop it in the general area and then use the select tool to move the shape to the correct position.

Once the shape is in position click on it so that it is highlighted. The BASIC code for the shape will then appear in the edit bar at the bottom of the screen. Edit the code as required.

For further information about each command see the 'BASIC Commands' help file.

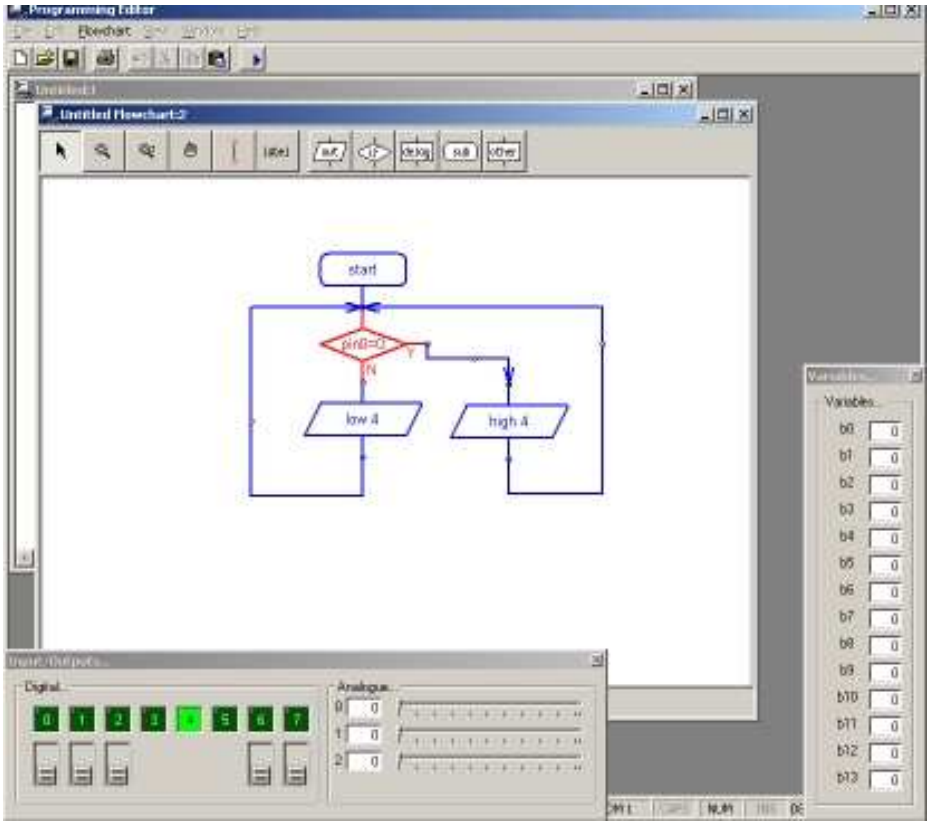
Joining Shapes

Shapes are joined by moving them close together until they 'snap' together. Alternately lines can be drawn between the shapes using the 'line tool' from the main toolbar. Note that it is only possible to join the bottom (side) of shapes to the top of other shapes (you cannot connect lines to lines). Only one line is allowed out of the bottom of each shape.

To enable neat diagrams, corners to the lines can be added by clicking with the mouse. When a line moves close to a connection point it will snap into position and then a click will finish the line.

Lines cannot be moved. If you try to move a line it will be deleted and a new line must be created.

On Screen Simulation



To simulate the flowchart, click 'Simulate' from the Flowchart menu. The program will then start to run on-screen.

As the program runs each cell is highlighted red as it is carried out. The 'Inputs/Outputs' and 'Variables' windows also appear when a simulation is being carried out. To adjust the input values click the on-screen switch (shown beneath the output LED) or slide the analogue input slider.

The time delay between shapes can be adjusted via the Flowchart options (View>Options>Flowchart menu).

Note that certain commands have no on-screen simulation equivalent feature. In this case the command is simply ignored as the flowchart runs.

Downloading Flowcharts

Flowcharts are not directly downloaded to the microcontroller. First the flowchart is converted into a BASIC program, which is then downloaded.

To convert a program select 'Convert' from the Flowchart menu. The BASIC program for downloading will then be created.

Shapes that are not connected to the 'start' or 'sub' shapes in the flowchart are ignored when the conversion takes place. The conversion will stop if an unconnected shape is found. Therefore always use a 'stop' shape or line to complete the flowchart before simulation or conversion.

Note that it is possible to quickly convert and then download a flowchart by pressing the shortcut key <F5> twice.

Using Symbols

Inputs, Outputs and Variables can all be renamed using the 'Symbol Table' from the Flowchart menu. When a symbol is renamed the new name appears in the drop-down menus on the edit bar. Note that you should not use commands (e.g. switch or sound) as a symbol as this will generate errors in your converted BASIC program.

Saving and Printing Flowcharts

Flowcharts can be saved, printed and exported as graphic files (for adding to word processor documents) via the File menu. Flowcharts can also be copied to the Windows clipboard (for pasting into other applications) via the Edit menu.

SECTION 4
PROGRAMMING - BASIC

Programming in BASIC is more powerful than using flowcharts. This is because BASIC contains more commands, eg. for...next loops, which cannot be used with the graphical flowchart methods. However you have to be more accurate in your 'typing' as no spelling mistakes are allowed!

The following program is a sample BASIC program which switches output 0 on and off every second. When you download the program an LED connected to output 0 would flash on and off every second.

```
main:
        high 0
        pause 1000
        low 0
        wait 1
        goto main
```

This program uses the **high** and **low** commands to control output pin 0, and uses the **pause** and wait commands to make a delay. Wait uses whole second units, whilst pause uses 1 millisecond (ms) units (1000 ms = 1 second). Therefore in this program both the delays are the same, just written in different ways.

The last **goto main** command makes the program 'jump' back to the label **main:** at the start of the program. This means the program loops forever. Note that the first time the label is used it must be followed by the colon (:) symbol. This tells the computer the word is a new label.

Detailed instructions:

1. Connect the PICAXE cable to the computer serial port. Note which port it is connected to (normally labelled COM1 or COM2).
1. Start the Programming Editor software.
2. Select View>Options to select the Options screen (this may automatically appear).
3. Click on the 'Mode' tab and select PICAXE-08
4. Click on the 'Serial Port' tab and select the serial port that the PICAXE cable is connected to. Click 'OK'
5. Type in the following program:

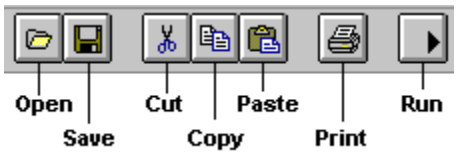
```
main:
        high 0
        pause 1000
        low 0
        wait 1
        goto main
```

   (NB note the colon (:) directly after the label 'main' and the spaces between the commands and numbers)

6. Make sure the PICAXE circuit is connected to the serial cable, and that the batteries are connected.
7. Select PICAXE>Run. A download bar should appear as the program downloads. When the download is complete the program should start running automatically – the LED on output 0 should flash on and off every second.

Programming Editor Software Reminders:

Toolbar short-cuts:



To download/run a BASIC program:

1. Check the download cable is connected to the PICAXE and the computer's serial port
2. Check that the battery is connected to the PICAXE
3. Make sure the Programming Editor software is in the correct mode (look for 'PICAXE-08' in the statusbar at the bottom left of the screen).
4. Click **PICAXE>Run** (or the toolbar icon) (or press the shortcut key F5)

To save a program/flowchart:

1. Click **File - Save As...** (or the toolbar icon)
2. Type in a filename
3. Click <OK>

To open a saved program/flowchart:

1. Click **File - Open...** (or the toolbar icon)
2. Select the file type (BASIC or flowchart)
3. Select a filename from the list by clicking on it
4. Click <OK>

To start a new BASIC program:

1. Click **File - New**

To start a new flowchart:

1. Click **File - New Flowchart** (or the toolbar icon)

To on-screen simulate a flowchart:

1. Click **Flowchart - Simulate...** (or the toolbar icon)
2. Click on the flowchart to stop the simulation

To convert a flowchart to BASIC:

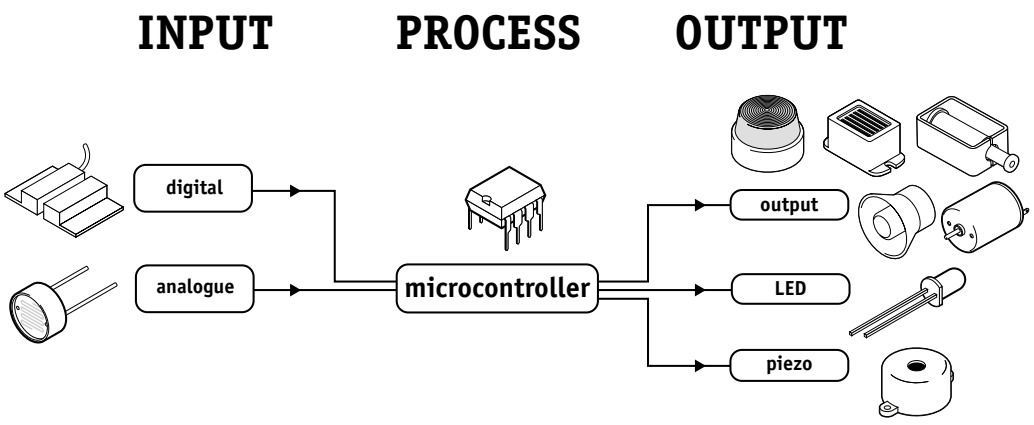
1. Click **Flowchart - Convert to BASIC...** (or press the shortcut key F5)

To print a program/flowchart:

1. Click **File - Print...** (or the toolbar icon)
2. If you want each program line printed in A BASIC program to have a number, make sure the 'Print Line Numbers' box is checked
3. Click <OK>

Section 5 - The Alarm PCB

The Alarm project uses a PICAXE-08 microcontroller with an LED and piezo sounder as user feedback devices, and a user selectable 'output' device (e.g. siren or strobe light).

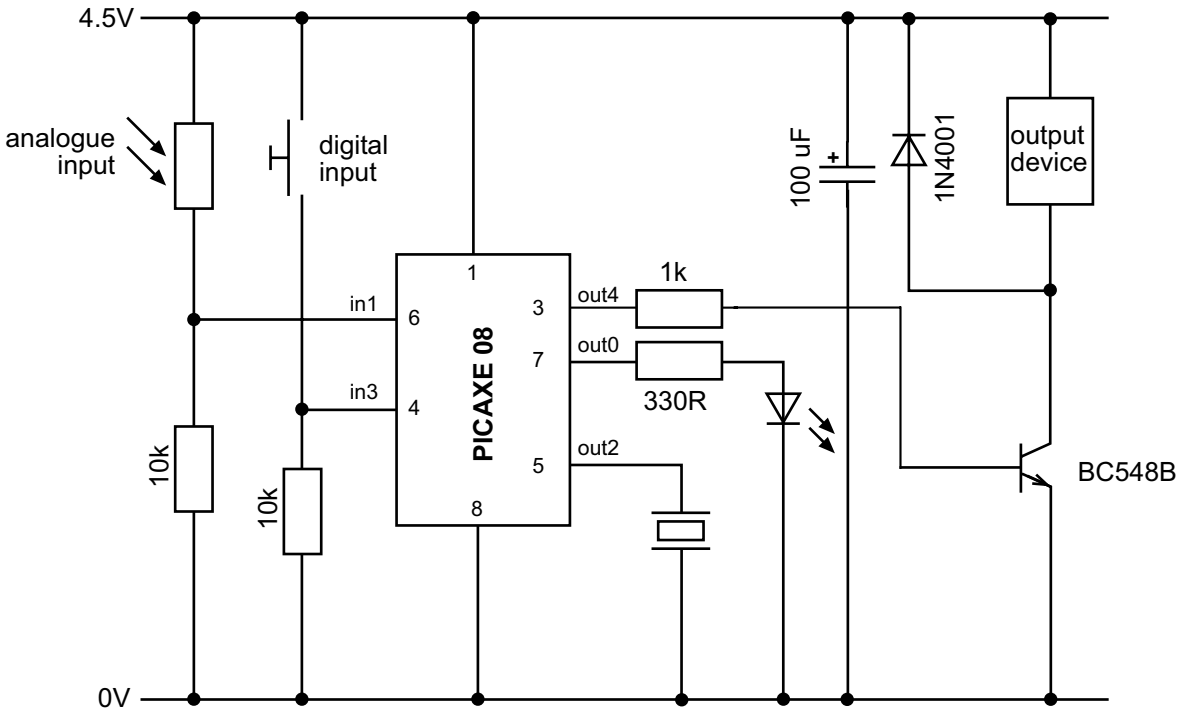


The project can also react to digital and/or analogue sensors (e.g. an LDR light sensor).
.
The electronic block diagram is shown below.
output - pin0 is connected to the LED
output - pin2 is connected to the piezo sounder
output - pin4 controls the output devices
input - pin1 is connected to the LDR
input - pin3 is connected to the push switch

Remember not to confuse the chip 'leg' number with the input/output pin number!

Circuit Diagram

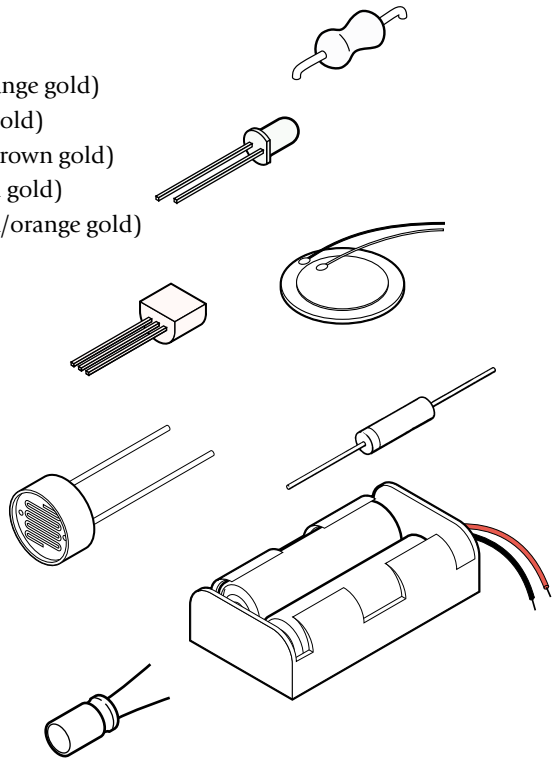
The circuit diagram for the alarm project is shown below:



Building the Alarm PCB

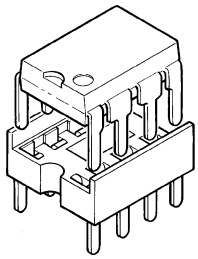
What you will need:

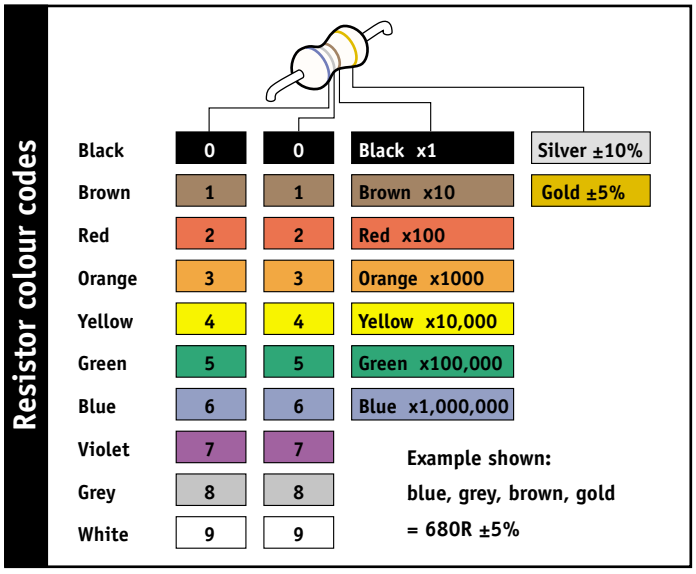
R1 and R2	10k resistor	(brown black orange gold)
R3	22k resistor	(red red orange gold)
R4	330R resistor	(orange orange brown gold)
R5	1k resistor	(brown black red gold)
R6	1k/10k resistor	(brown black red/orange gold)
(the value of R6 depends on the type of LDR used)		
LED1	5mm red LED	
PZ	piezo sounder	
TR1	BC548B Transistor	
D1	1N4001 diode	
C1	100uF electrolytic capacitor	
IC1	8 pin IC socket	
IC1	PICAXE-08 microcontroller	
CT1	PICAXE download 3.5mm socket	
BT1	battery clip	
BT1	4.5V (3xAA) battery box	
PCB	printed circuit board	



wires and sensors (e.g. switch and LDR)
output device (e.g. buzzer or motor)

Tools:
soldering iron and solder
side cutters





Soldering the PCB.

The printed circuit board (PCB) is specially manufactured with a 'solder resist' layer to make it simpler to solder. This is the green 'lacquer' layer that covers the tracks so that the solder does not stick to these tracks. However for successful assembly the PCB must be carefully assembled and soldered.

When soldering always make sure the solder iron tip is hot and clean. To test if it is hot enough try to melt a piece of solder on the tip. The solder should melt almost instantly. Then clean off the melted solder by wiping the tip on a damp sponge.

Remember that solder will only 'stick' to hot surfaces. Therefore never melt the solder on the soldering iron tip and then try to 'drop' it onto the joint – this won't work as the joint will be cold and so the solder won't stick.

To successfully solder you must hold the soldering iron in one hand and the solder in the other. Therefore make sure the board is held on the table so it won't move (e.g. use a bulldog clip or get someone else to hold it for you).

Steps to soldering:

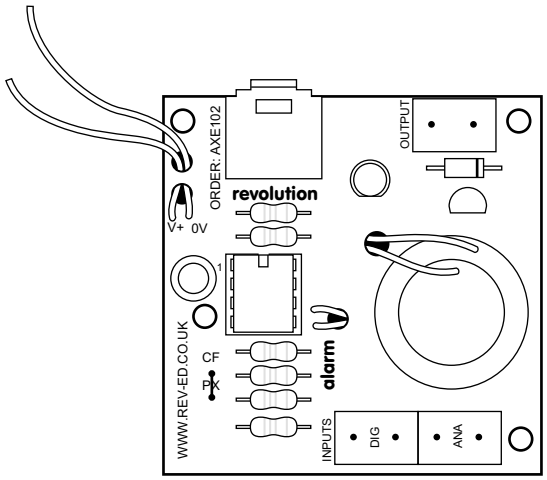
1) Clean the soldering iron tip on the damp sponge
2) Press the soldering iron tip against the pad on the PCB AND the leg of the component. Count to 3 to give the joint time to warm up.
3) Keep the soldering iron in position and touch the solder against the joint. Allow enough solder to melt to cover the joint.
4) Take the solder away first, then the soldering iron
5) Allow the solder to cool for about 5 seconds before trying to move the board.

After each joint is made make sure it does not accidentally 'bridge' across to other joints. However be aware that some solder joints (e.g. on the two sides of the PICAXE download socket) have two wires very close together that are already connected by a track (line) on the PCB. In this case it does not matter if the solder joins together.

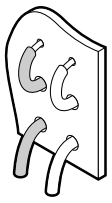
Tips!

1) Always start with the smallest components like the resistors. Then move onto larger components like the IC socket and then finish with the tall components like capacitors and transistors. Do not try to put all the components in position at once, only do two or three at a time.
2) Always make sure that the components lie flat on the board before they are soldered. When using components with long legs like resistors and LEDs, bend the legs so that the component is held firmly in position before soldering.
3) Make sure the PICAXE stereo download socket 'snaps' into position flat on the board before it is soldered.
4) Make sure that the components that only work one way around (LEDs, diodes, transistors and capacitors) are correctly aligned before soldering (see the marks on the PCB).
5) Piezo sounder wires are very thin. Make sure you do not overheat them or they may melt.
6) Always thread the battery snap wires down and up through the two thread holes before soldering. This helps make a much stronger joint which is less likely to snap off.

These instructions presume you are soldering all the components directly on the board. The instructions are identical if you are using longer wires to join some components (e.g. the LED), although you must be even more careful you get the wires around the correct way on the LED.



1) Place the 1k (brown black red gold) and 22k (red red orange gold) resistors in position. The bottom resistor marked 1k may be a 1k or 10k resistor depnding on the type of analogue sensor (LDR) used. Bend the legs to hold the resistors in position and then solder.
2) Place the two 10k (brown black orange gold) and 330 (orange orange brown gold) resistors in position. Bend the legs to hold the resistors in position and then solder.
3) Using an off cut resistor leg, make a wire loop over the letters PX marked beside the 330R resistors. Solder in position. (Ignore the hole above the holes marked CF).
4) Solder the 1N4001 diode is position D1, making sure the silver bar is pointing towards the right hand side of the PCB.
5) Push the PICAXE stereo download socket onto the PCB and make sure it clicks into position (so that it lies flat on the board). Solder the five metal square contacts (the five round plastic support post holes do not have to be soldered). Do not worry if the solder joins on the two metal contacts either side of the socket as they are supposed to be joined anyway.
6) Push the IC socket into position. Make sure the notch at one end points up towards the socket. Fold the legs over to hold the socket in position and then solder.
7) Solder the LED into position (or wires if the LED is to be connected by longer wires). Make sure the flat on one side of the LED aligns with the flat marked on the PCB.
8) Solder the BC548B transistor in position, making sure the flat side aligns with the marking on the PCB.



9) Solder the 100uF capacitor is position, making sure the + and – legs are around the correct way.
10) Thread the battery clip down through the large hole by the letters DER. Thread it back up through the large hole by the letters ORD. Then solder the black wire in the hole marked 0V and the red wire in the hole marked V+.
11) Use a sticky pad to stick the piezo sounder (brass side) to the top of the PCB. Thread the wires down through the hole below LED1 and back up through the hole marked PZ. Solder the red wire into the bottom hole and the black wire into the top hole. It does not matter if the red wire solder joint joins pin5 of the IC socket as they are supposed to join anyway. The black wire should NOT join pin 6 of the IC socket.
12) Solder wires to the OUTPUT, DIG & ANA INPUT connections.
13) Carefully check the board to make sure there are no missed joints or accidental solder bridges.
14) Insert the microcontroller into the socket, ensuring pin1 faces the stereo socket.

Testing your circuit.

Step 1 – Check the solder joints.

Check that all the joints are connected to both the pad and the wire, and that the wire is held firmly so that it does not 'wobble' when pulled. Also check that the solder does not accidentally bridge between two pads. This is most likely to happen on the LED and on the piezo. On the stereo socket the two square pads close together on each side can be joined as they are already joined by a track on the board. However they must not be joined to the central round hole.

Step 2 - Check the components.

1) Check that the black battery clip wire is in the hole marked '0V' and the red battery clip wire is in the hole marked 'V+'
2) Check that the PICAXE-08 chip is in the socket correctly, with the dent (showing pin1) closest to the stereo socket.
3) Check that the flat edge of the LED is connected to the correct hole on the PCB.
4) Make sure you have not forgotten the wire link over the holes marked PX at the bottom left of the board.
5) Make sure the brass side of the piezo is stuck down with a sticky pad.
6) Check that the socket is correctly soldered, including the middle square pad which is often forgotten by mistake.

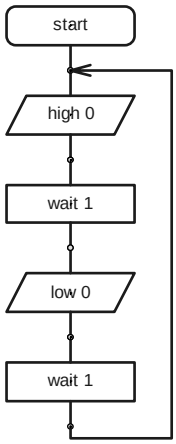
Step 3 - Connect the battery.

Check the 3 AA batteries are in the battery box correctly. Connect the battery box to the battery snap and put your finger on the PICAXE chip. If it starts to get hot remove the battery box immediately as there is a problem – most likely that the chip or the battery snap wires are around the wrong way.

Step 4 – Download a program to test LED 0.

Connect the cable to the back of the computer and to the PICAXE socket on the PCB. Make sure the cable is pushed fully into the socket on the PCB.

Make sure the software is in the PICAXE-08 mode and the correct serial port is selected (see section 4 of this booklet for more information).



Type in and download the following program:
program like this:

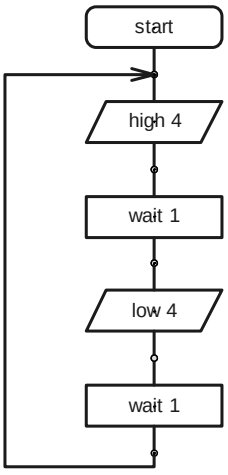
```
main:
        high 0
        wait 1
        low 0
        wait 1
        goto main
```

The LED should flicker as the program downloads. After the download is complete the LED should flash on and off every second. If the LED does not flash check that it is around the correct way and that the 330R resistors are in the correct positions on the PCB.

If the program does not download check that the 22k, 10k, socket and IC socket are all soldered correctly. Use a multimeter to make sure you are getting 4.5V across the top legs (1 and 8) of the microcontroller. Check that the cable is pushed firmly into the socket and that the correct serial port is selected within the software.

Step 5 – Test the output

Connect an output device (e.g. buzzer) to the output wires and then type in and download the following program:

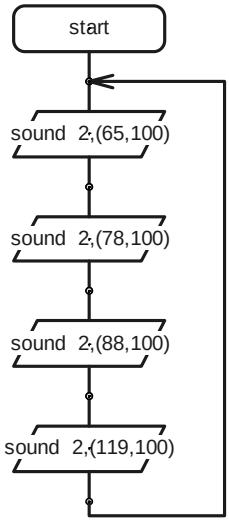



```
main:
        high 4
        wait 1
        low 4
        wait 1
        goto main
```

The buzzer should sound on and off every second. If it does not check that the transistor and diode and buzzer wires are all around the correct way.

Step 6 – Test the piezo

Type in and download the following program:




```
main:
        sound 2,(65,100)
        sound 2,(78,100)
        sound 2,(88,100)
        sound 2,(119,100)
        goto main
```

The piezo should make 4 different noises. If it does not make sure the wires are correctly soldered, that it is stuck on the brass side with a sticky pad (it will not work if 'hanging loose') and that the wire link over the letters PX is on the board.

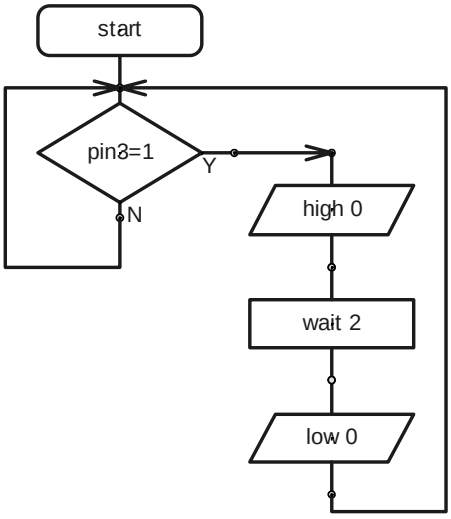
Step 7 – Test the switch

Connect a switch to the digital input.
Type in and download the following program.

```
main:                                   ' make a label called 'main'
        if input3 is on then flash      ' jump if the input is on
        goto main                       ' else loop back around

flash:                                  ' make a label called 'flash'
        high 0                          ' switch output 0 on
        wait 2                          ' wait 2 seconds
        low 0                           ' switch output 0 off
        goto main                       ' jump back to start
```

The LED on output 0 should light whenever the switch is pushed. If it does not check that the switch and 10k resistors are correctly soldered.

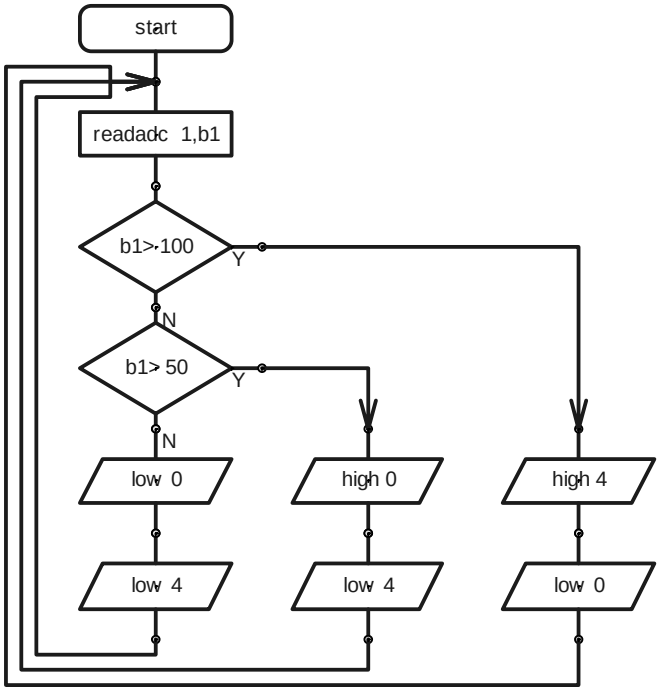


Step 8 – Test the LDR

Connect an LDR to the analogue input. Type in and download the following program.

```
main:
        readadc 1,b1
        if b1 > 100 then do4
        if b1 > 50 then do0
        low 0
        low 4
        goto main

do4:
        high 4
        low 0
        goto main
do0:
        high 0
        low 4
        goto main
```



Note: it may be necessary to alter the threshold values if the room is dark e.g. try 60 and 30 instead of 100 and 50.

The LED and output should come on at different times as you raise and lower your hand over the LDR (so that different amounts of light fall on the LDR). If they do not check that the LDR and 1k resistor are correctly soldered.

If all these tests pass, you can be congratulated as you have correctly built and assembled your alarm! It is now time to develop and test your own program to operate your alarm system.

Section 6 - Program Ideas.

Now that you have assembled and tested your alarm, it is time to develop your own program. This program can make the alarm react in different ways to the digital and analogue sensors.

Included on the next pages are two example programs. These are designed to give you a starting point for your program. You may choose to modify them or to start a completely new program if you prefer.

Program 1 Explanation

This general purpose program has a main loop which flashes the LED on and off, and also checks the analogue sensor (fitted with a LDR) and digital input (fitted with a switch). When the push switch is pressed an alarm is sounded for two seconds.

If the LDR light sensor is covered the piezo will make a awarning beep sound until the light level rises again.

Program 2 Explanation

This program is designed as a fire alarm system, where the alarm is activated when smoke is detected over the light sensor (ie the light sensor gives a lower value). Once triggered the alarm is held on and cannot be switched off until power is removed from the system.

The digital input is used as an 'anti-tamper' input. Whenever the case of the alarm is closed, the switch will be held on - this is the normal condition. If the case is opened the switch will open and so this triggers the piezo alarm until the case is closed again.

Program 1

```
' ***** main loop *****
' loop here flashing lights
' and checking sensors

main:
' LED full on and read light value
       high 0
       readadc 1,b1

' if analogue value low then make sound
       if b1 < 40 then beep

' if switch pushed do alarm
       if pin3 = 1 then alarm

' do a delay
       pause 500

' LED full off and check sensor again
       low 0
       readadc 1,b1

' if analogue value low then make sound
       if b1 < 40 then beep

' if switch pushed do alarm
       if pin3 = 1 then alarm

' do a delay
       pause 500

       goto main

' ***** make sound *****
beep:
       sound 2,(120,50,80,50,120,50)
       pause 200
       goto main

' ***** alarm output on ****
alarm:
       high 4
       pause 2000
       low 4
       goto main
```

Program 2

```
' ***** main loop *****
' loop here checking sensors

main:
' LED off
       low 0

' read light value
       readadc 1,b1

' if analogue value low then activate alarm
       if b1 < 40 then alarm

' if switch off do tamper
       if pin3 = 0 then tamper

       goto main

' ***** tamper - beep until switch shut *****
tamper:
       high 0
       sound 2,(120,100)
       if pin3 = 1 then main
       goto tamper

' ***** alarm output on forever ****
alarm:
       high 4
       goto alarm
```

ACKNOWLEDGEMENT

This project development was funded by the UK Offshore Oil and Gas Industry.
www.oilandgas.org.uk/education/