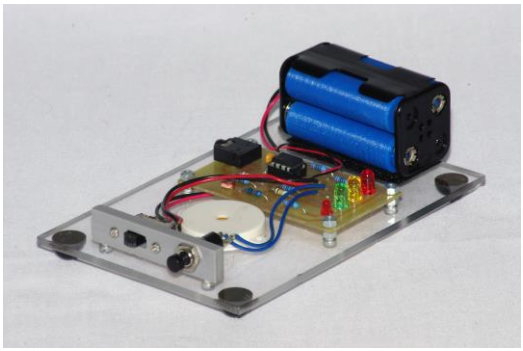
BEEPA

CONTENTS:

DESCRIPTION

The BEEPA project provides an introduction to microcontrollers, programming and electronics. BEEPA can be programmed to light three LEDs, produce sound from a piezo transducer, respond to pushbutton presses and respond to the light level.



SECTION 1: GENERAL AND PLANNING INFORMATION

1. DESIGN CONSIDERATIONS

1.1 GENERAL

Before construction, plan and lay out all the components using a suitable computer program or on a sheet of paper. Look at BEEPA as a complete unit, and not just as separate parts. Use our drawings as a starting point for your design.

1.2 ITEMS FOR INVESTIGATION

The tasks to complete the project include assembling and testing the circuit board and performing a range of investigations.

Investigations contained in this Teaching Unit include the following:

- Flashing LED
- Flickering LED
- LED chaser
- Three LED blinker
- Binary counter
- Traffic lights
- Heads or tails
- Light to flash rate
- Light activated flashing LED
- Light meter
- Shadow detector
- LED Dimmer
- Cricket
- Music using play command
- Music using tune command
- Shower or egg timer
- Tone using pwmout command
- Theremin
- Courtesy light

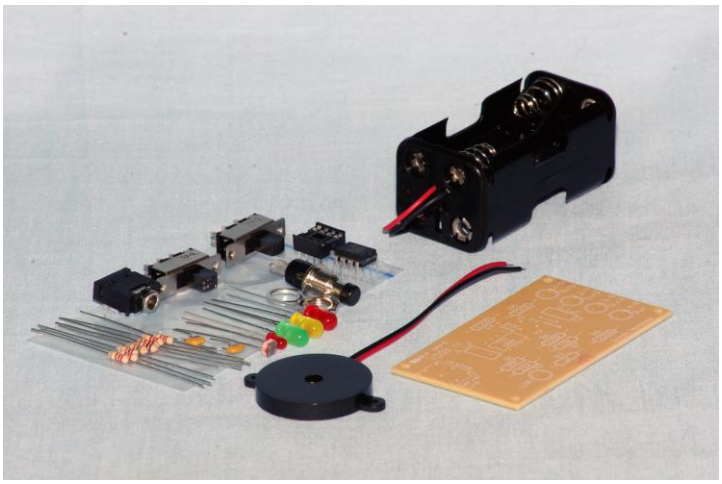
Additional areas of investigation could include:

- Investigate the operational requirements of a typical toy or domestic appliance, such as an alarm clock or a microwave oven, that use a microcontroller (embedded control system).
- Student's own project, such as a game or a "cyberpet".
- Investigate adding other output devices, such as lamps, motors and solenoids.

SECTION 2: COMPONENTS & MATERIAL REQUIRED

2.1 COMPONENTS SUPPLIED

The following components are supplied in the kit:



2.1 ADDITIONAL REQUIREMENTS

The following items are required and are available from Scorpio Technology:

- Battery – AA, 4 required (BATTAA)
- AXE027 - PICAXE USB Download Cable, which can be ordered from us. The "AXE027 USB Cable Driver" can be downloaded for free from www.picaxe.com.

The following material is to be supplied by the student / designer:

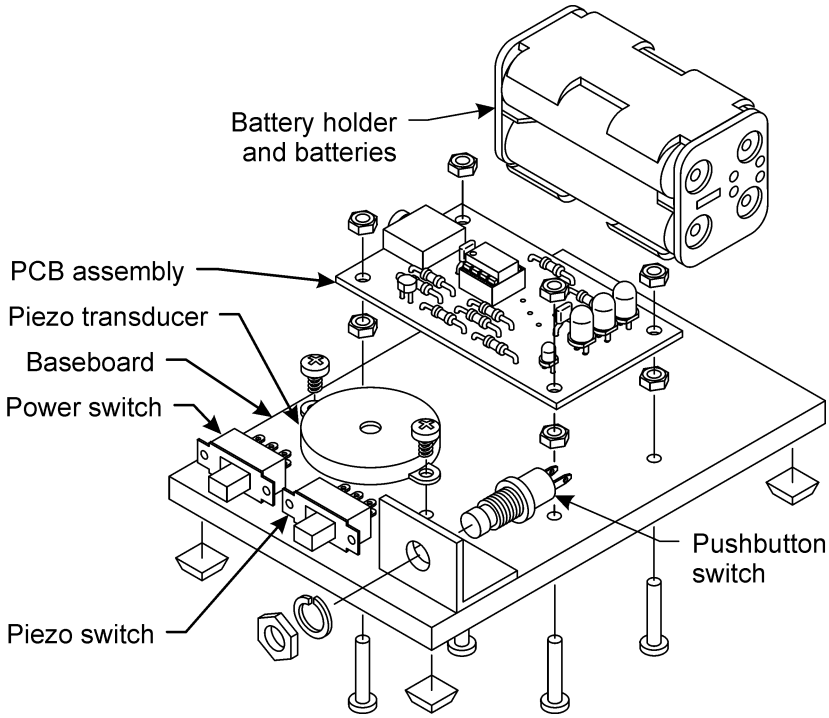
- Electric hook-up wire – Multi-strand in assorted colours
- Material for the platform (PVC or acrylic sheet, plywood, etc.)
- Assorted screws, nuts and washers
- The PICAXE editor, which can be downloaded for free from www.picaxe.com. The PICAXE programming editor software requires a PC running Windows XP or later.

2.2 TOOLS REQUIRED

The following tools are required:

- Assorted hand tools
- Soldering equipment and solder

SECTION 3: MECHANICAL DESIGN



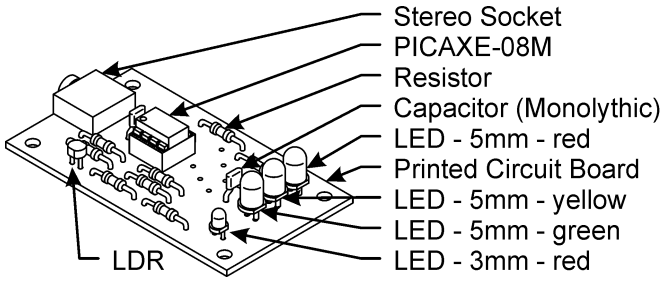
- Determine size and material for the baseboard. We used 75x130mm plastic sheet (acrylic or PVC).
- Determine a suitable position and attachment for the PCB assembly. We used nuts and bolts.
- Determine a suitable position and attachment for the battery holder. We have used hot-melt glue, double-sided tape and hook-and-loop tape ("Velcro" or equivalent).
- Determine a suitable position and attachment for the power switch, piezo transducer, piezo switch and pushbutton switch. We attached the switches using hot-melt glue.

Determine a suitable method to prevent the screws underneath the baseboard scratching the surface on which it is placed. We used self-adhesive feet.

SECTION 4: ASSEMBLING BEEPA

4.1 PRINTED CIRCUIT BOARD ASSEMBLY

CAUTION: Take care in orienting the IC socket and LEDs. Unsoldering and replacing damaged or wrongly positioned components will waste time. Do not overheat the PCB and components.

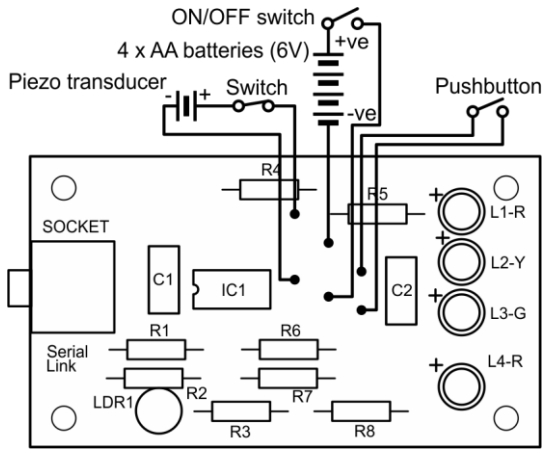


PCB COMPONENT TERMINOLOGY

- Solder the components to the PCB in the following order: resistors, capacitors, LDR, IC socket (notched end indicates leg 1), stereo socket and LEDs (flat is negative).
- Trim the component leads as required.
- Do not insert the IC into its socket yet. This will be inserted just prior to electrical testing.

4.2 WIRING AND MECHANICAL ASSEMBLY

NOTE: For wiring, use wires of different colours to assist in tracing wires during fault finding. When soldering wires, strip a short piece of insulation from the end of the wire, twist the strands and "tin" them with solder.



WIRING DIAGRAM

- Solder the red (positive) wire from the battery holder to the power switch. Solder the black (negative) wire from the battery holder to "0v" on the PCB. Solder a length of red wire between the switch to "V+".
- Solder the wires from the piezo transducer and switch to the PCB. Connect red to "+" and black to "-".
- Solder two length of wire between the pushbutton switch and "Switch" connections.
- Attach the PCB assembly, battery holder, piezo transducer, switches and pushbutton.

SECTION 5: ELECTRICAL TESTING

- Inspect soldering for short circuits and poor "wetting" of component leads or pads.
- Insert four 1.5Volt AA batteries into the battery holder. Move the power switch to "on". Check that the LED (L4) illuminates (this shows that power is available and is the correct polarity).
- If the LED does not illuminate:
- Check that the batteries are properly inserted in the battery holder.
- Check that the battery voltage is above 5.5 volts. (If low, replace the batteries.)
- Check that battery voltage is present between legs 1 and 8 on IC1. Leg 1 should be positive.
- Check the wiring against the wiring diagram.
- Check the values of the resistors against the circuit diagram.
- Check that the LED is the right way around.
- Check that the LED is working by using a 220 Ohm resistor and 6Volt (battery) power.
- Move the power switch to "off". Check the orientation of the IC - the end with leg 1 is identified with a notch or dimple at one end. Line up the legs of the IC with its IC socket holes and press down firmly. Do not use the letters/numbers on the IC to identify leg numbers.

NOTE: It may be necessary to bend the IC legs slightly to line them up with the socket holes.

CAUTION: ICs will be damaged if they are installed in the wrong direction or if power supply (battery) connections are reversed.

SECTION 6: PROGRAMMING THE PICAXE

6.1 INSTALL PICAXE EDITOR

NOTE: PICAXE editor needs to be installed only once.

IMPORTANT: Always connect the AXE-2027 USB Download Cable to the same USB port.

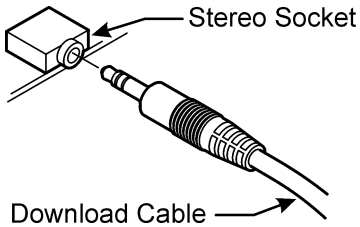
- Start up and log into your PC. (Some PCs require that you log in as the 'Administrator' to install software. See your systems administrator if you do not have administrative rights.)
- From the website www.picaxe.com, select "Free Software" > "Picaxe Programming Editor" > "Prog. Editor Installer". The installer will be downloaded to your computer.
- Select "Free Software" > "AXE027 USB Cable Driver". Download "Driver Installation Instructions". Download "Windows USB Driver (self extracting preinstaller format)".
- Install the PICAXE editor.
- Install the AXE027 USB Download Cable driver. Follow the installation instructions.
- Insert the AXE-027 USB Download Cable into an available USB port.

6.2 START PICAXE EDITOR

- Click Start>Programs>Revolution Education>Programming Editor to start the software.
- If the Options screen does not automatically appear, click the View>Options menu. On the 'Mode' tab select PICAXE-08M2 mode. (The PICAXE-08M, with "12F683" written on it, is also compatible with BEETLE, but instead use the PICAXE-08M mode.) On the 'Serial Port' tab select the serial COM port allocated to the "AXE-027 PICAXE USB".
- Select Help>About. Check that the software version is 5.5.1 or later.
- The PICAXE programming editor software is ready to use.

6.3 EDIT PROGRAM

- Use the PICAXE programming editor software to create your programs. Save each version with a different file name.



CONNECTING DOWNLOAD CABLE

6.4 TRANSFER PROGRAM TO PICAXE

- Move the power switch to "off".
- Connect the AXE-027 USB Download Cable between your PC and the stereo socket on the PCB.
- Move the power switch to "on".
- Run the PICAXE Programming Editor software and transfer your program to the PCB. The program will automatically run once it has been successfully downloaded.
- If the Programming Editor software gives an error message stating that program cannot be transferred to the PICAXE:
  - Check that the power LED on the PCB is on.
  - Try downloading the program again.
  - Check that the cable is fully inserted into the stereo socket.
  - Check that the AXE027 USB Download Cable is inserted into the correct USB port.

- Check that the PICAXE Programming Editor software is set to the appropriate COM port.
- Carry out the checks listed in the Electrical Testing section.

- Disconnect the AXE027 USB Download Cable from the stereo socket on the PCB.

SECTION 7: INVESTIGATION

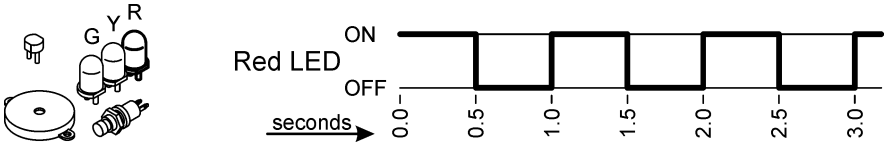
For programming language details (help), from the PICAXE "Programming Editor" help menu, open "PICAXE Manual 2 - BASIC Commands".

After checking that each program operates, you are encouraged to change the program to investigate the result. In some cases, a valid range for a parameter is specified.

The Picaxe-08M2 will not "break" if you do something "wrong" in a program. If you make a mistake, (a) the Programming Editor will give you an error message or (b) the Picaxe-08M2 will do something unexpected. In either case, check your typing for mistakes and think through what you have done and modify your program accordingly.

7.1 FLASHING LED

The red LED is switched on (high) for 0.5 seconds (500 milliseconds) and off (low) for 0.5 seconds.

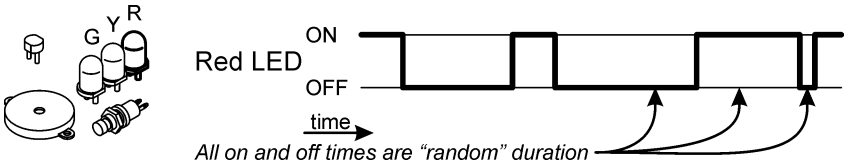



```
main:
        high 0
        pause 500
        low 0
        pause 500
        goto main
```

- Change the program to flash the yellow LED (output 2).
- Change the program to flash the green LED (output 4).
- Change the flashing speed (valid range for pause is 0 to 65535 milliseconds.).

7.2 FLICKERING LED

The internal random number generator is used to create an erratically flashing LED.

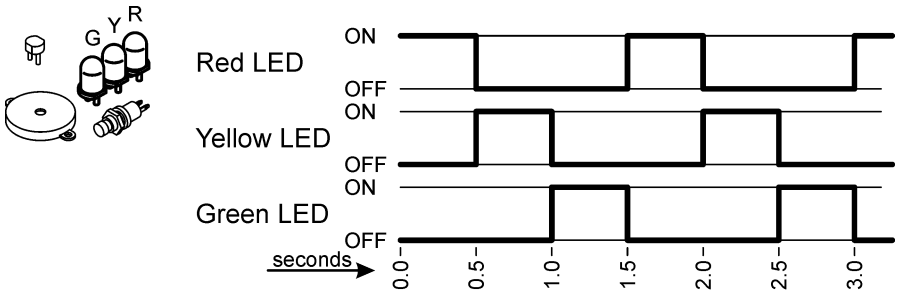



```
main:
        random w0
        high 0
        pause b1
        low 0
        random w0
        pause b1
        goto main
```

- Is the same sequence used each time the Picaxe is turned on? Why? (Refer to help.)
- What is the relation between the variables w0 and b1? (Refer to help.)

7.3 LED CHASER

An LED is illuminated for 0.5 seconds, then the next one is illuminated for 0.5 seconds, etc.

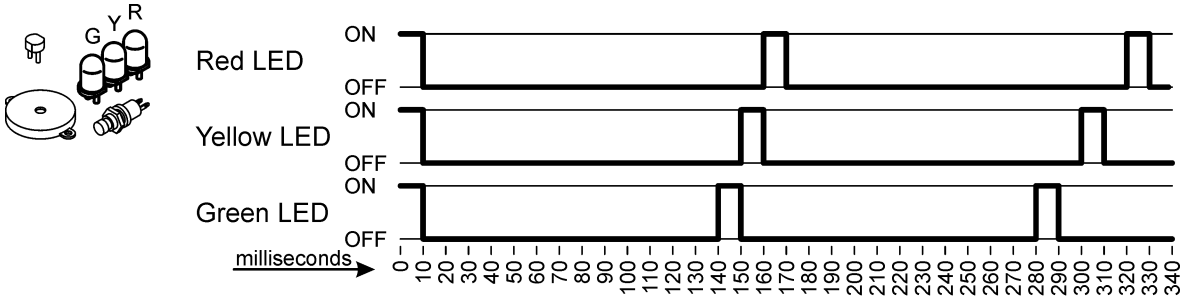



```
main:
        high 0
        pause 500
        low 0
        high 2
        pause 500
        low 2
        high 4
        pause 500
        low 4
        goto main
```

- Change the LED chaser so that it runs at twice the speed.
- Reverse the direction of the sequence.

7.4 THREE LED BLINKER

The three LEDs are blinked at different rates.




```
'Note 16 x 15 x 14 = 3360
main:
        for w0 = 1 to 3360
                b2 = w0 % 16
                if b2 <> 0 then skip1
                high 0
skip1:
                b2 = w0 % 15
                if b2 <> 0 then skip2
                high 2
skip2:
                b2 = w0 % 14
                if b2 <> 0 then skip3
                high 4
skip3:
                pause 10
                low 0
                low 2
                low 4
        next
        goto main
```

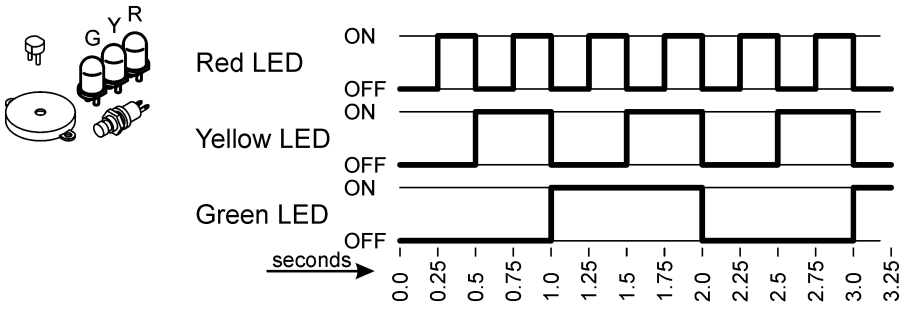
- What is the function of the following lines of code?

```
main:
skip1:
skip3:
skip2:
```

- Why is the number 3360 used in the program?
- Why is the modulus (%) calculation used? What other symbol can be used? (Refer to help.)

7.5 BINARY COUNTER

The LEDs are used to display binary numbers. The weighting for the LEDs are: red=1, yellow=2, green = 4. The range of binary numbers that can be displayed is from 0 (000) to 8 (111).

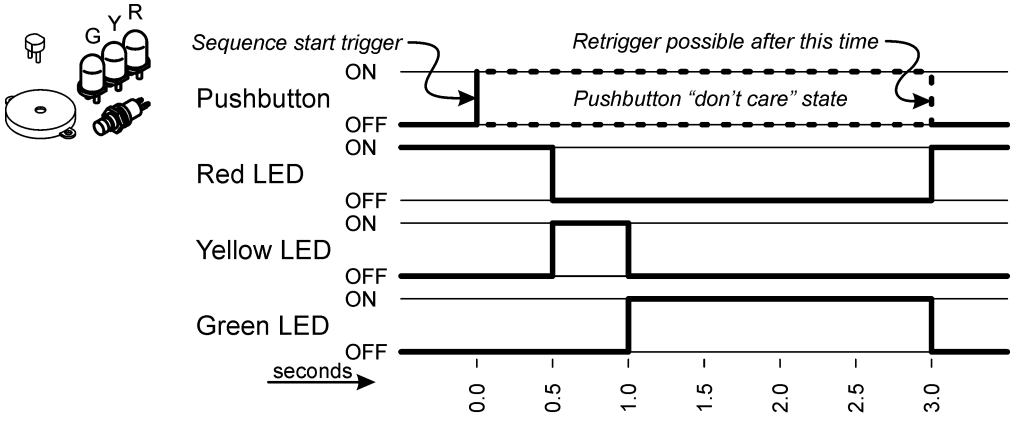



```
        b0 = 0
main:
        b1 = b0 and 1  'check 1st bit
        low 0
        if b1 = 0 then skipred
        high 0
skipred:
        b1 = b0 and 2  'check 2nd bit
        low 2
        if b1 = 0 then skipyellow
        high 2
skipyellow:
        b1 = b0 and 4  'check 3rd bit
        low 4
        if b1 = 0 then skipgreen
        high 4
skipgreen:
        pause 250
        b0 = b0 +1
        goto main
```

- Why is the binary number system important in digital computers?
- Why do the numbers 255 and 65535 appear frequently in this document? (Hint: power of 2.)

7.6 TRAFFIC LIGHTS

When the pushbutton is pressed, the yellow LED is displayed, then the green and back to the red.

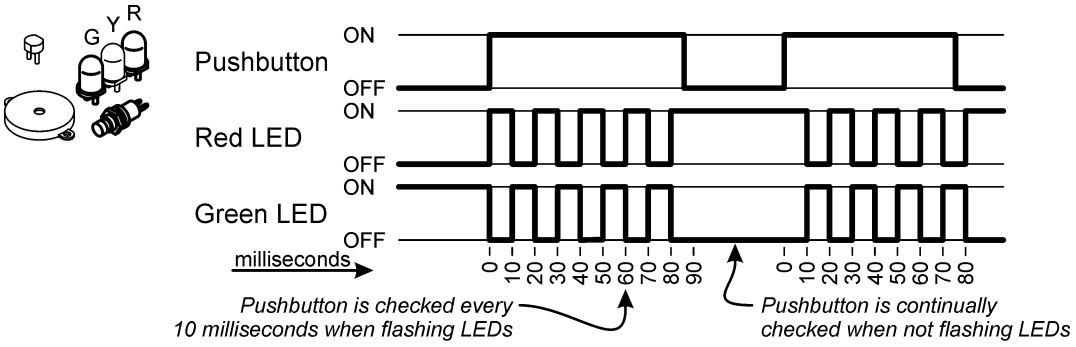



```
        high 0
        low 2
        low 4
main:
        if pin3 = 0 then main
        pause 500
        low 0
        high 2
        pause 500
        low 2
        high 4
        pause 2000
        low 4
        high 0
        goto main
```

- What is the function of the three program lines before the "main:" label?
- Change the program so that the sequence will not restart if the pushbutton is kept on.

7.7 HEADS OR TAILS

When the pushbutton is pressed, the red and green LEDs quickly alternate and stop when the pushbutton is released.

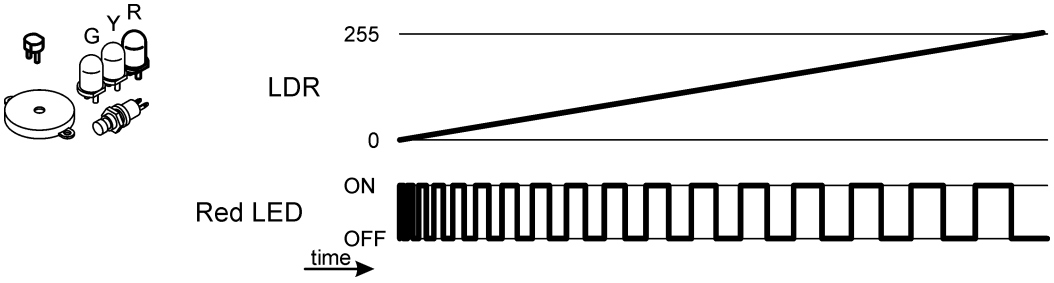



```
        high 0
        low 4
main:
        if pin3 = 0 then main
flashLED:
        high 0
        low 4
        pause 10
        if pin3 = 0 then main
        low 0
        high 4
        pause 10
        if pin3 = 0 then main
        goto flashLED
```

- Count the results for 100 pushbutton presses. Did you get an equal number of heads and tails?

7.8 LIGHT TO FLASH RATE

The LDR light level is stored in variable "b0". The red LED flashes quickly at low light levels and slows down at higher light levels.




```
main:
        readadc 1, b0
        high 0
        pause b0
        low 0
        pause b0
        goto main
```

- Insert the following line after the "readadc" statement to double the flash rate.

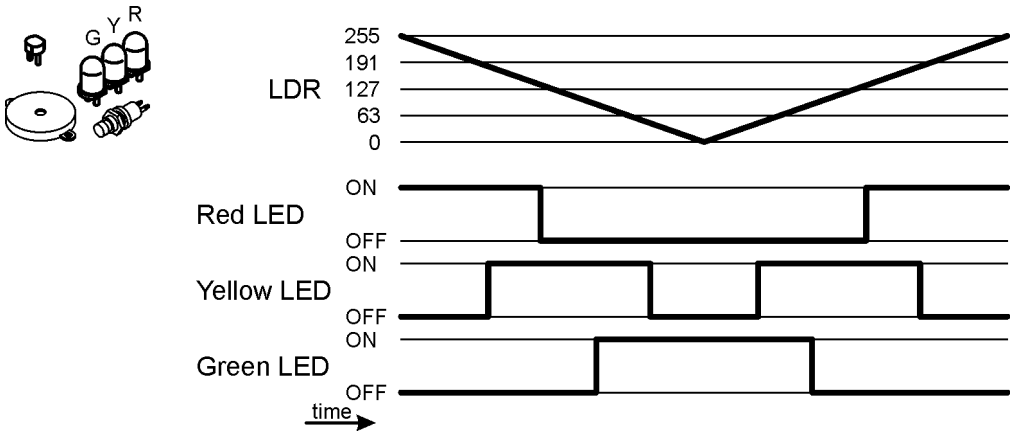
```
        b0 = b0 / 2
```

- Insert the following line after the "readadc" statement to have a slow flash at low light levels and fast flash at high light levels.

```
        b0 = 255 - b0
```

7.9 LIGHT METER

The LEDs indicate the current light level. Red is high, yellow is medium and green is low.

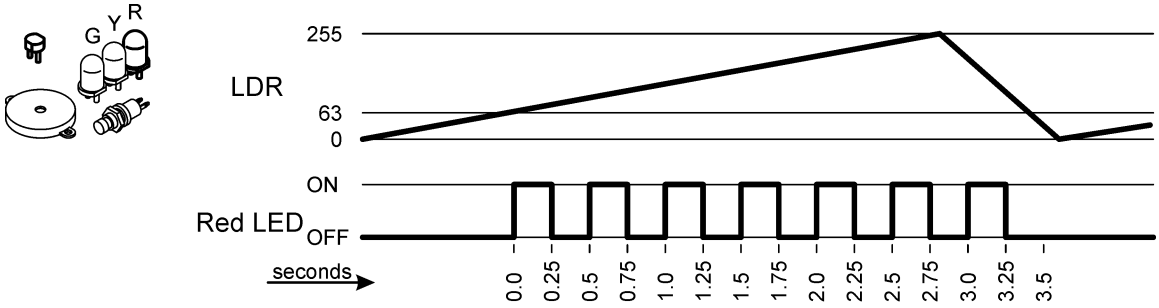



```
main:
        pause 10
        readadc 1, b0
        if b0 > 160 then LEDred
        if b0 > 120 then LEDredyellow
        if b0 > 80 then LEDyellow
        if b0 > 40 then LEDyellowgreen
        goto LEDgreen
LEDred:
        high 0
        low 2
        low 4
        goto main
LEDredyellow:
        high 0
        high 2
        low 4
        goto main
LEDyellow:
        low 0
        high 2
        low 4
        goto main
LEDyellowgreen:
        low 0
        high 2
        high 4
        goto main
LEDgreen:
        low 0
        low 2
        high 4
        goto main
```

- Change the light level triggers so that they suit your environment.

7.10 LIGHT ACTIVATED FLASHING LED

The program reads the LDR light level (input 1) and stores it in variable "b0". The red LED flashes if the light level is higher than the value in the program ("if" statement).



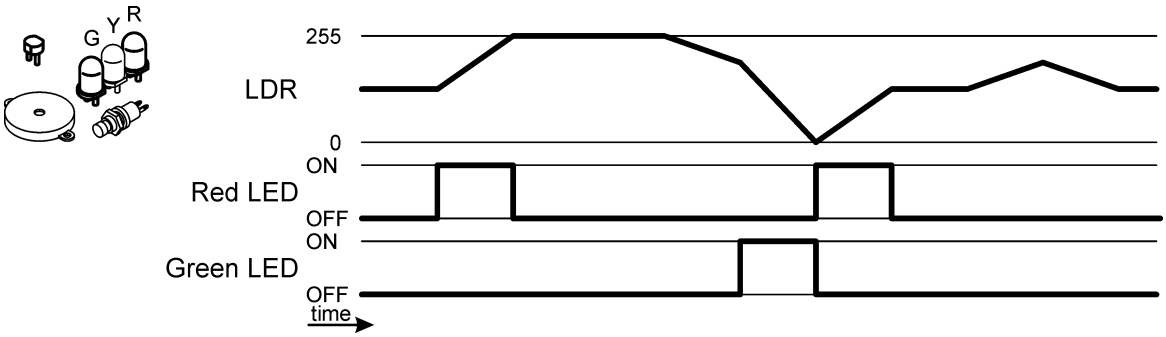

```
main:
        readadc 1, b0
        if b0 < 63 then main
        high 0
        pause 250
        low 0
        pause 250
        goto main
```

- What is the effect of changing "<" to ">" in the "if" statement?

- Change the trigger point to suit your environment.

7.11 SHADOW DETECTOR

The red LED is flashed for a rapid light increase and the green LED for a rapid light decrease. The average light reading is maintained using the EWMA (Exponentially Weighted Moving Average) method.

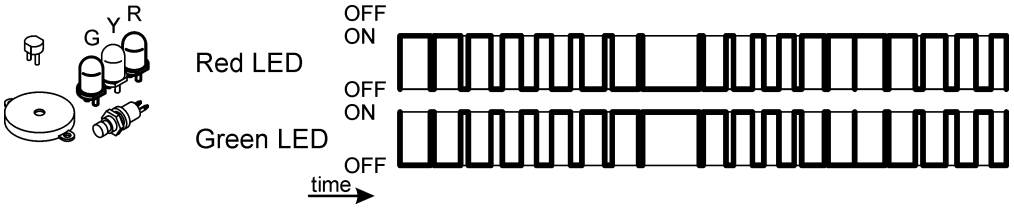



```
        readadc 1, b0       'initial average = b0
main:
        pause 10
        low 0
        low 4
        readadc 1, b1  'current value = b1
        w1 = b0 + b1   'temporary variable = w1
        b0 = w1 / 2    'store average in b0
        if b1 > b0 then check_green
        goto check_red
check_green:
        b5 = b1 - b0   'temporary variable = b5
        if b5 < 2 then main
        high 4
        goto main
check_red:
        b5 = b0 - b1   'temporary variable = b5
        if b5 < 2 then main
        high 0
        goto main
```

- Adjust the sensitivity to light level changes so that they are more suitable for your environment.
- Is there limitations to light intensity for this program? If so, then how could this be overcome?

7.12 LED DIMMER

The LEDs are dimmed and then made brighter using PWM (Pulse Width Modulation). The varying on/off ratio gives the perception of changing intensity.



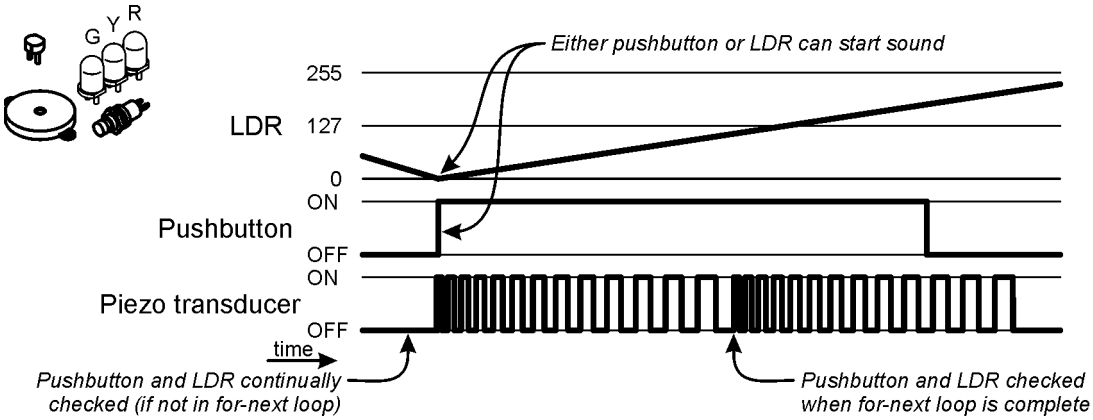

```
main:
        for b0 = 0 to 15
                b1 = 15 - b0
                low 0
                high 4
                pause b0
                high 0
                low 4
                pause b1
        next
        for b0 = 0 to 15
                b1 = 15 - b0
                low 0
                high 4
                pause b1
                high 0
                low 4
```

```
        pause b0
    next
    goto main
```

- Why are there two for-next loops in the program?
- Change one of the outputs to the yellow LED and listen to the sound from the piezo transducer. What does this tell you about program operation?
- What is the minimum frequency of the LEDs to minimise flickering?
- What are some applications that take advantage of persistence of vision?

7.13 CRICKET

When the pushbutton is pressed or the light level is low, the piezo transducer is rapidly switched on and off causing a sound to be produced. The switching rate is changed during each cycle.

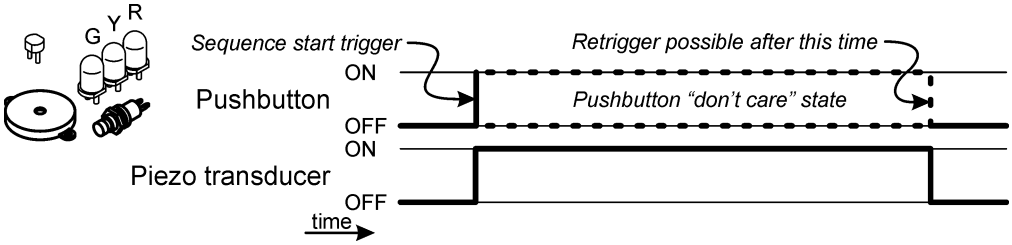



```
main:
        readadc 1, b1
        if b1 <= 4 then cricket
        if pin3 = 1 then cricket
        goto main
cricket:
        for b0 = 0 to 32
                high 2
                pause b0
                low 2
                pause b0
        next
        goto main
```

- What is the effect of changing the values in the "for" statement (valid range 0 to 255)?

7.14 MUSIC USING PLAY COMMAND

When the pushbutton is pressed, a pre-programmed internal tune is played on the piezo transducer.

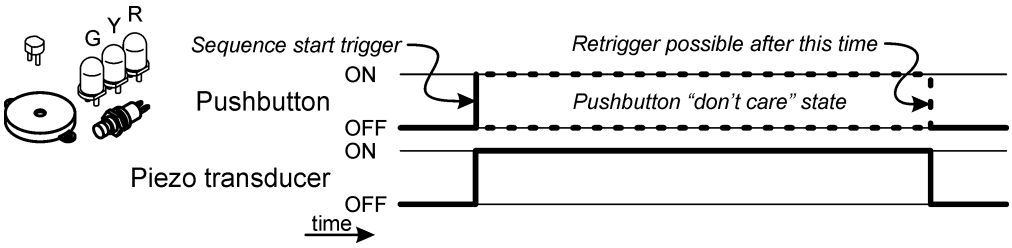



```
main:
        if pin3 = 0 then main
        play 0, 3
        goto main
```

- In the "play" command, change first parameter from 0 to 1, 2, and then 3. What changes?
- In the "play" command, change the second parameter from 3 to 0, 1 and then 2. What changes?

7.15 MUSIC USING TUNE COMMAND

When the pushbutton is pressed, a tune is played on the piezo transducer.



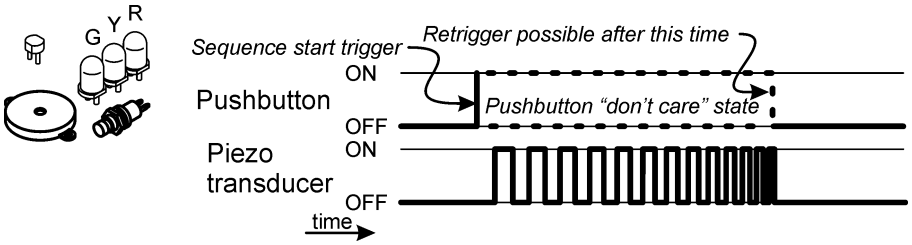

```
main:
        if pin3 = 0 then main
        tune 0,4,($EA,$C5,$43,$42,$40,$CA,$05,$43,$42,$40,$CA,$05,$43,$42,$43,$C0)
        goto main
```

Create a different tone sequence using the "Ring Tone Tunes" wizard in the Picaxe editor.

7.16 TONE USING PWMOUT COMMAND

Using an internal PWM generator is used to create sound. Because the PWM operates in the background, the program can do other tasks if required.

NOTE: The valid parameter range for the "PWMOUT wizard" is 3899Hz (pwmout 2, 255, x) to 14000Hz (approximate limit of hearing) (pwmout 2, 70, x), where x is the second parameter multiplied by between 0 (0% duty cycle) and 4 (100% duty cycle). (Refer to help for more detail.)

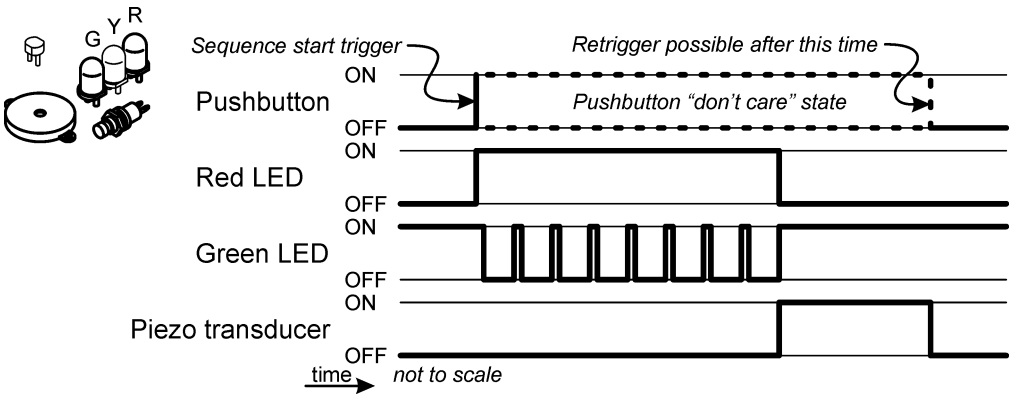



```
main:
        if pin3 = 0 then main
        for w0 = 255 to 70 step -1
                w1 = w0 * 2     'set up for approximately 50% duty cycle
                pwmout 2, w0, w1
                pause 10
        next
        pwmout 2, off
        goto main
```

- Change the program so that the sound goes from a high frequency to a low frequency.
- Change the program so that the sound goes from a low frequency to a high frequency and back to a low frequency.

7.17 SHOWER OR EGG TIMER

When the pushbutton is pressed, a timer is started. When the timing period has elapsed, a tune is played. LEDs indicate the timer status. (For testing purposes, the timing interval is set for 20 seconds – this can be changed as required.)



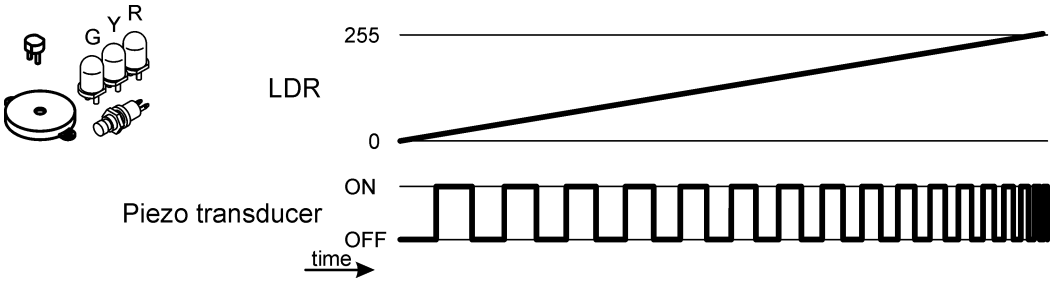

```
        low 0
        high 4
main:
```

```
        if pin3 = 0 then main
        high 0
        for b0 = 1 to 20
                high 4
                pause 10
                low 4
                pause 990
        next
        low 0
        high 4
        tune 0,4,($EA,$C5,$43,$42,$40,$CA,$05,$43,$42,$40,$CA,$05,$43,$42,$43,$C0)
        goto main
```

- From the user's point of view, what are the functions of the red and green LEDs?
- Change the program to increase the potential timing period above 255 seconds.
- Change the timing period in the "for" command line to suit your application.

7.18 THEREMIN

The theremin is a musical instrument that can be played without touching it. This version converts light intensity to frequency.

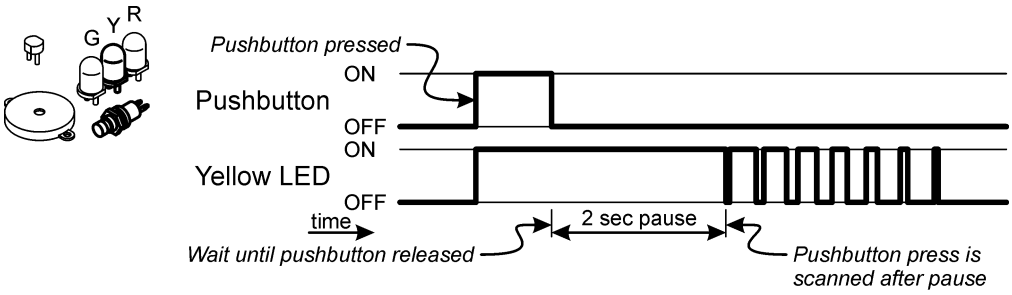



```
main:
        w0 = 0
        readadc 1, b0  'b1 is low byte of w0
        b0 = 255 - b0  'invert LDR reading
        w0 = w0 * 185  'convert LDR range to 255 - 70 = 185
        w0 = w0 / 256
        w0 = w0 + 70   'move base by 70
        w1 = w0 * 2            'set up for approximately 50% duty cycle
        pwmout 2, w0, w1
        pause 1n
        goto main
```

- What is the function of the "b0 = 256 – b0" command?
- What effect of deleting the pause statement?

7.19 COURTESY LIGHT

This program simulates the operation of a car's interior light. When the pushbutton is pressed (door opened), the yellow LED (door light) is turned on. When the pushbutton is released, the yellow LED stays on for a short time and then slowly fades off.



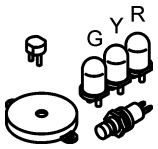

```
main:
        if pin3 = 0 then main
        high 2
buttonpressed:
        if pin3 = 1 then buttonpressed
        pause 2000
        for w0 = 1023 to 0 step -1
                pwmout 2, 100, w0
                pause 1
                if pin3 = 1 then exit
        next
        low 2
        pwmout 2, off
        low 2
```

```
goto main
```

- What is the function of the command "if pin3 = 0 then exit"?
- Change the program to use the green LED. Because the pwmout command will only function on output 2, you will need to change the dimmer routine to use a similar approach as used in "LED Dimmer". What changes are needed to make the program work satisfactorily? (Hint: Use nested for-next loops and the "setfreq" command.)

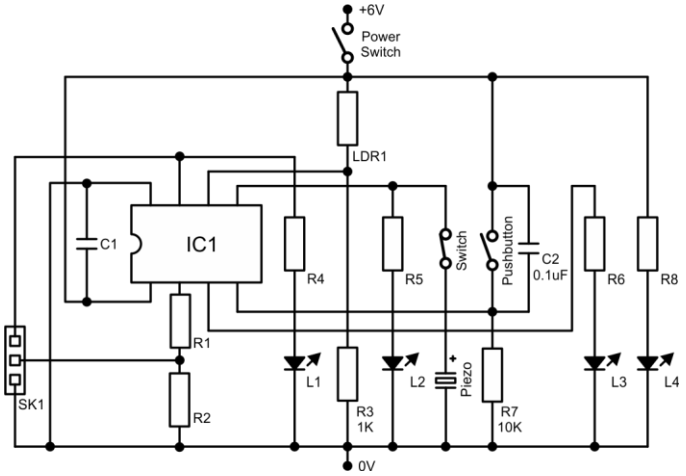
7.20 YOUR OWN PROJECT

Create and test your own application program using ideas presented in this section and other sources. Some ideas are listed below:



- Create your own cyberpet with LED eyes, touch sensor, light level detection, sound output and intelligence that you give to it. Use insulated wires to relocate components inside a suitable toy. If required, some Picaxe-08M2 pins can be reallocated to become inputs or outputs.
- Create your own game. Use the LDR and switch as inputs, the LEDs and/or piezo transducer as outputs. If required, some Picaxe-08M2 pins can be reallocated to become inputs or outputs.
- Change the input and/or output devices that are used in your project. For example, outputs could be used to switch a lamp, a motor or a solenoid. (For high current devices, such as these, you will need to add some interface circuitry – refer to Picaxe documentation.)

SECTION 8: HOW THE CIRCUIT WORKS (THEORY)



CIRCUIT DIAGRAM

8.1 CIRCUIT OVERVIEW

- PICAXE documentation refers to "Input Pins" and "Output Pins", which are not the same as the physical pins on a device. To avoid confusion, in this document "leg" means a physical pin of an integrated circuit and "pin" means a logical input or output.
- A program can be downloaded into the PICAXE-08M2 memory to control the three LEDs and piezo transducer in response to input from the switch and/or the light sensor (LDR).

8.2 ABOUT PICAXE MICROCONTROLLERS

- The PICAXE* is a type of IC (Integrated Circuit) called a microcontroller, which is another name for a single chip computer. The PICAXE has similar features to a normal PC: CPU (central processing unit), RAM (random access memory), ROM (read only memory), I/O (input/output) lines, timers and A/D (analogue to digital) converters.

* PICAXE is a trademark of Revolution Education Ltd.

- The Flash memory (EEPROM - Electrically Erasable Programmable Read Only Memory) in a PICAXE allows it to be reprogrammed many times (typically 100,000). This means that you can develop a program and constantly check the effects of changes.

- A PICAXE program is created using an easy to learn version of the BASIC programming language (our preferred method) or using flowcharting software.
- The PICAXE is supplied containing 'bootstrap' code that enables you to download your program using the serial cable. Do not substitute the PICAXE with a blank PIC microcontroller or any other integrated circuit.

8.3 PICAXE-08M2 (IC1)

- Leg 1 is connected to the positive terminal (+6V) of the power supply (batteries).
- Leg 2 is used only when transferring a program from a serial port on your PC to the PICAXE. The 22k Ohm (R1) and 10k Ohm (R2) resistors must be present for reliable operation.
- Leg 3 (pin4 in the program) is connected to the green LED.
- Leg 4 (pin3 in the program) is connected to the pushbutton switch.
- Leg 5 (pin2 in the program) is connected to the yellow LED and piezo tranducer.
- Leg 6 (pin1 in the program) is connected to the LDR to measure light intensity.
- Leg 7 (pin0 in the program) is connected to the red LED and is used during program download.
- Leg 8 is connected to the negative terminal (0V) of the power supply (batteries).

8.4 POWER

- Power switch SW1 is used to control power to the circuit.

8.5 CAPACITORS

- Capacitor C1 filters the power supply close to the IC.
- Capacitor C2 reduces switch bounce from the pushbutton switch connected to leg 4.

8.6 PUSHBUTTON SWITCH

- The value of the pushbutton switch status is accessible using the variable "pin3". When the pushbutton is released, 0V is present at leg 4 through 10k Ohm resistor (R7). When the pressed, leg 4 is connected to positive. Capacitor (C2) across the switch decreases electrical noise.
- If pressing the switch causes the PICAXE to reset (program starts from the beginning), shorten the wires connecting the pushbutton to the printed circuit board and/or remove capacitor C2.

8.7 LIGHT DEPENDANT RESISTOR (LDR)

- The LDR changes its resistance depending on the amount of light falling on the sensor. The LDR and a 1k Ohm resistor (R3) form a voltage divider across the battery. The voltage at the junction of the components connected to leg 5 and is accessible using the "readadc" command.

8.8 LIGHT EMITTING DIODE (LED)

- A LED (L4), through 220 Ohm current limiting resistor (R8), indicates when power is on.
- Three LEDs (L1, L2, L3) (red, yellow and green) are connected to the Picaxe-08M2 outputs via 220 Ohm resistors (R4, R5, R6). If required, the LED colours may be swapped or changed.

8.9 PIEZO TRANSDUCER

- The piezo transducer is used to generate sound and may be switched on when required.