



Programming & Embedded Systems KS2 & KS3



A TEACHER'S HANDBOOK FOR THE NATIONAL CURRICULUM

B P Smith & G W Asquith

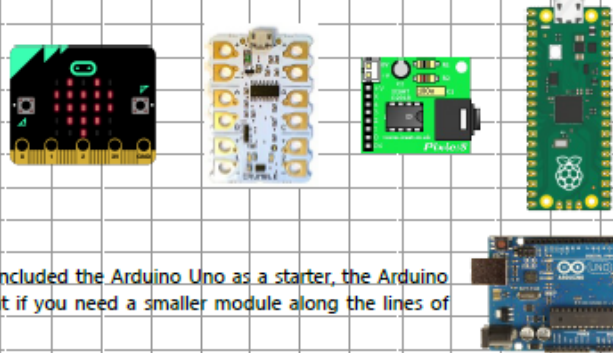
CONTENTS

0. Preface	14. Appendix
1. Introduction	14.1 What's on the CD
2. Coding/programming requirements at KS2 & KS3	14.2 Systems information
3. Embedded System & Electronics	14.3 Making enclosures
4. What is a Programmable system?	14.4 Hints & Tips
5. What is a microcontroller?	14.5 Useful suppliers
6. What systems are available for KS2 & KS3?	
7. Embedded systems - The basics	Symbols used in the book:
7.1 Inputs	NC symbol
7.1.1 Basic sensors	
7.2 Outputs	NC requirements
7.2.1 LEDs	
7.2.2 Motors	
7.2.3 Sound	
7.3 Power supply	Red number blobs are years
8. Programming methods	
9. How to design an embedded project	Curriculum links indicated by the jigsaw icon
10. Getting started with programming	Support materials on CD
10.1 Setting up your IDE	
10.2 How to connect your microcontroller to your computer	
10.3 How to connect input and output devices to your microcontroller	
10.4 Some simple examples to get you up and running	The 3S's reminder logo
11. Which commands do I use for....	
12. Working projects	
12.1 Mood light	
12.2 Automatic light	
12.3 Touch wire game	
12.4 Temperature controlled fan	
12.5 Celebration lamp	
12.6 Collecting / Charity Money Box	
12.7 Animated POS display / Sign	
13. Glossary	

1. INTRODUCTION

This book provides support on how to implement programming for KS2/3 DT, it describes the most common systems available along with their strengths and weaknesses:

- BBC MicroBit
- Crumble
- ICSAT's Pixie:8 system
- Raspberry Pi Pico



In addition for KS3 we have included the Arduino Uno as a starter, the Arduino Nano is a good starting point if you need a smaller module along the lines of Raspberry Pi Pico.



How to develop programming solutions for each of the above is given along with the key instructions you will need to use, when developing your own and pupil's solutions. A companion CD with supporting materials, code etc, is also provided.

The book illustrates eight different projects that develop programming within a D&T context and the 3S's methodology. Each project demonstrates how the required coding/programming can be achieved using each of the 4 programmable systems and describes any other electronics knowledge necessary.

The seven projects are:

1. Mood lamp – RGB LED
2. Automatic lighting – house, garden, LDR sensor
3. Touch wire game – sound, light, and lives
4. Temperature controlled fan – Heat sensor, dc motor
5. Celebration lamp – RGB LED, sound/tune, tilt switch
6. Money box – Vibration, sound, light
7. Animated signage – sound/tune, LED, LDR sensor, servo motor



7. EMBEDDED SYSTEMS – THE BASICS

The development of the microcontroller has revolutionised the electronics industry and the products it produces. The microcontroller has moved products from being all **hardwired** based where the functionality is determined by the circuit itself to one that is **softwired**, that is the functionality of the circuit is now determined by software (program code). This change has made electronics much easier and flexible to use.

Any electronic system in a product can be separate it into three parts:



Inputs – Electrical/electronic sensors, which take signals from the physical world (in the form of temperature, pressure, etc.) and convert them into electronic signals.

Processing circuits - These consist of electronic components connected together to manipulate, interpret and transform the information contained in the signals.

Outputs - Actuators or other devices that transform electronic signals back into human readable form – light, sound, movement etc.

This systems approach is a useful design tool for the creation of new circuits for new products.

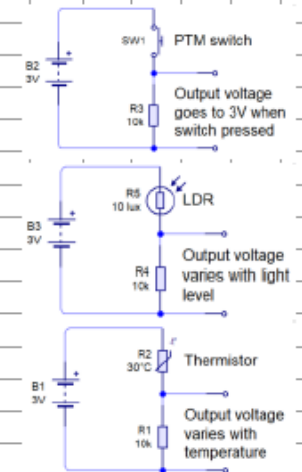
What are inputs?

Inputs to a system are usually sensors of some type, the most common sensors are:

1. Switch based, most common are PTM (Push to make), microswitch or tilt switches that produce a voltage when they are pressed or tilted.
2. Light sensing using an LDR (Light dependent resistor) to detect light levels.
3. Thermistor for heat sensing, which can be used to measure temperature.

They are used to input information into the system, so that it can make decisions based upon their values and turn outputs and output devices on or off

In most cases the battery voltage would be between 3V and 5V, and be provided by a suitable battery pack or USB power pack.



8. PROGRAMMING METHODS

To program an embedded system using a microcontroller we need to use a piece of software known as an IDE (Integrated Development Editor), within this software you create your code using several different methods:

Block-based systems

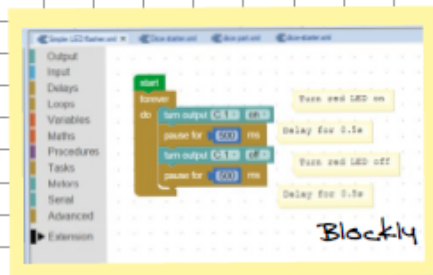
These types of coding methods are based on Scratch developed at the Massachusetts Institute of Technology (MIT), they are essentially a graphical method using coloured blocks, which contain the coding commands, they click together to create your code. The blocks are colour-coded so that all like commands are grouped in the same colour, such as inputs, outputs, etc.

Also, where values are required, you are guided in what you can enter, generally, they won't let you enter values (*parameters*), etc, which don't make sense, for example, use connections (pins) that don't exist or don't have that function.

This type of coding method makes it quite easy to transfer to a script-based system, this is because if you 'remove' the coloured graphics you essentially have a coding script, which is very similar to Python, MicroPython, CircuitPython, and BASIC.

The most common examples of this type are:

- Scratch
- MakeCode
- Blockly



9. HOW TO DESIGN AN EMBEDDED SYSTEMS PROJECT

The next step is to select the most appropriate input and output devices, you may only have one card per block, but you may likely have more than one. In this case, you will have several possible solutions which, tried out, evaluated and the best solution identified during your code development and testing.

Now you have an idea of what the components of your system are, it's time to write the code. The big question is how do I start! The easiest method is to use a process known as pseudocode, all you have to do is write a step-by-step list of the things the system needs to do, in a simple English sentence, to operate as you require, this is known as an algorithm.

For example, a simple night light system using an LED and light sensor (LDR):

1. Is it dark?
2. If the answer is yes then turn on the LED
3. If the answer is no then turn off the LED
4. Repeat steps 1 to 3

Once you have a simple step-by-step list as above, we then start to expand each step to make it more precise in terms of what has to happen:

1. Read the value of the light sensor
2. If the light sensor value is dark then set the LED output high
3. If the light sensor value is light then set the LED output low
4. Repeat steps 1 to 3

Let's do another iteration:

1. Read the value of the light sensor and store the value into a variable
2. If the light sensor value is dark then set the LED output high
3. If the light sensor value is light then set the LED output low
4. Repeat steps 1 to 3

The above process only needs to be repeated 2/3 times, then you should have enough information to allow you to convert the pseudocode into proper code.



12.1 MOOD LIGHT

Mood Light project

Context:

A common Mood light circuit only uses a switch, resistor and colour cycling LED. This has limitations in that it cannot be changed in any way, there is no means of altering its function.

Using a microcontroller system, you can take the Mood light concept even further; you could make it:

- Cycle faster / slower
- Do a different set of colours
- Do a specific colour sequence
- Do a different colour sequence each time
- Respond to environmental conditions such as light or heat

3S's

Description:

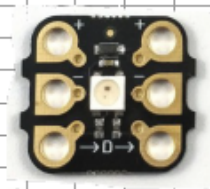
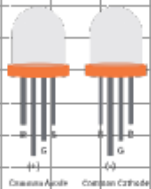
This project uses an RGB LED, the LED's colour will be cycled through a range of colours at random. The range of colours is determined by the coder and its target use.

Task:

Your task is to create a product along with a software solution that uses a microcontroller system and a RGB LED to create a Mood light, which can form the basis of a new product, using one or more ideas from the above list of possible functions this type of system could offer.

Background information:

An RGB LED is an LED that contains a red, green and blue LED in the same package, although it has 4 leads. In the case of Crumble, it uses a Sparkle LED, which is an intelligent LED as it contains its own controller, as shown below:

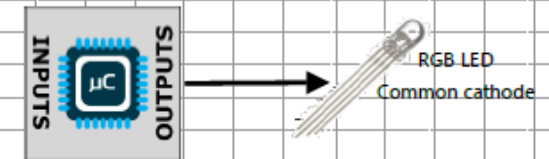


RGB LED's as with all LED's need a series resistor to protect each coloured LED from too much current, this can be calculated using Ohms law, in most cases, a 330Ω resistor will be ok.

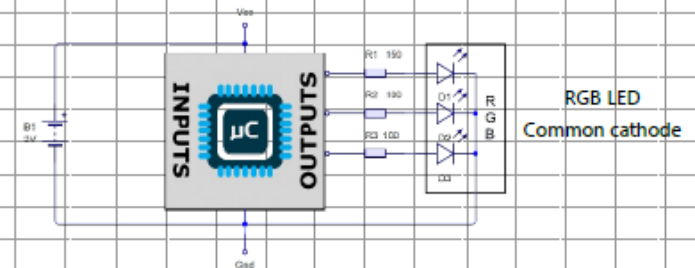
More details can be found in the Design & Technology KS2 book in the *Using Electronic Systems* section.

12.1 MOOD LIGHT

Block diagram:



Circuit diagram:



Take Note

The MicroBit and RPI Pico both use a 3V power supply, the Pixie8 can use 3V - 4.5V and Crumble uses 4.5V.

The red LED requires a 150Ω resistor, the green and blue LED's 100Ω resistors.

Pseudocode:

Begin

Setup variables

Setup IO pins

Do forever

Select a random number

Set colour of LED based on random number

Wait for 1s

EndDo

End

Remember pseudocode is a simple description of the program in plain English for each step required.



Blank area for drawing or notes.

121 MOOD LIGHT

Crumble

Setup
1 x Sparkle



Code

```

program start
do forever
  let x = random 0 to 7
  if x = 0 then
    set sparkle 0 to 0 0 0
  and if
  if x = 1 then
    set sparkle 0 to 0 0 255
  end if
  if x = 2 then
    set sparkle 0 to 255 0 0
  end if
  if x = 3 then
    set sparkle 0 to 255 0 255
  end if
  if x = 4 then
    set sparkle 0 to 0 255 0
  end if
  if x = 5 then
    set sparkle 0 to 0 255 255
  end if
  if x = 6 then
    set sparkle 0 to 255 255 0
  end if
  if x = 7 then
    set sparkle 0 to 255 255 255
  end if
  wait 250 milliseconds
loop
    
```

The Sparkles colour is set by the 3 values for red green and blue in that order. A value of 255 is maximum brightness, 0 is off and 128 is 1/2 brightness.

You might like to try this version, with more colours and a different method.

```

program start
do forever
  let R = random 0 to 255
  let G = random 0 to 255
  let B = random 0 to 255
  set sparkle 0 to R G B
  wait 250 milliseconds
loop
    
```

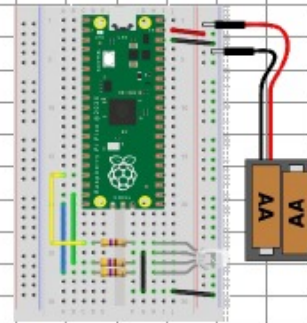


Supporting code cards are on the CD.

121 MOOD LIGHT

RPI Pico

Setup
RGB LED CC
R1,2,3 = 470Ω



IO Type	Name	Pin
Output	redLED	GP13
Output	greenLED	GP14
Output	blueLED	GP15

Code

```

1 from machine import Pin
2 from utime import sleep
3 from random import randint
4
5 redLED = Pin(13, Pin.OUT)
6 greenLED = Pin(14, Pin.OUT)
7 blueLED = Pin(15, Pin.OUT)
8
9 def setBlack():
10 redLED.off()
11 greenLED.off()
12 blueLED.off()
13
14 def setBlue():
15 redLED.off()
16 greenLED.off()
17 blueLED.on()
18
19 def setRed():
20 redLED.on()
21 greenLED.off()
22 blueLED.off()
23
24 def setMagenta():
25 redLED.on()
26 greenLED.off()
27 blueLED.on()
28
29 def setGreen():
30 redLED.off()
31 greenLED.on()
32 blueLED.off()
33
34 def setCyan():
35 redLED.on()
36 greenLED.on()
37 blueLED.off()
38
39 def setYellow():
40 redLED.on()
41 greenLED.on()
42 blueLED.off()
43
44 def setWhite():
45 redLED.on()
46 greenLED.on()
47 blueLED.on()
48
49 colour = 0
50
51 while True:
52 colour = randint(0, 7)
53 if colour == 0:
54 setBlack()
55 if colour == 1:
56 setBlue()
57 if colour == 2:
58 setRed()
59 if colour == 3:
60 setMagenta()
61 if colour == 4:
62 setGreen()
63 if colour == 5:
64 setCyan()
65 if colour == 6:
66 setYellow()
67 if colour == 7:
68 setWhite()
69 sleep(0.25)
70
    
```

here's a version which is smaller by using the .value instead of .off / .on here 0 / 1 are used and these are sent to the function to turn an LED on (1) or off (0).

```

1 from machine import Pin
2 from utime import sleep
3 from random import randint
4
5 redLED = Pin(13, Pin.OUT)
6 greenLED = Pin(14, Pin.OUT)
7 blueLED = Pin(15, Pin.OUT)
8
9 def setColour(R, G, B):
10 redLED.value(R)
11 greenLED.value(G)
12 blueLED.value(B)
13
14 colour = 0
15
16 while True:
17 colour = randint(0, 7)
18 if colour == 0:
19 setColour(0,0,0)
20 if colour == 1:
21 setColour(0,0,1)
22 if colour == 2:
23 setColour(1,0,0)
24 if colour == 3:
25 setColour(1,0,1)
26 if colour == 4:
27 setColour(0,1,0)
28 if colour == 5:
29 setColour(0,1,1)
30 if colour == 6:
31 setColour(1,1,0)
32 if colour == 7:
33 setColour(1,1,1)
34 sleep(0.25)
35
    
```

Take Note

In programming there can be a number of ways to solve a problem. Some will be less elegant than others, this gives you differentiation, progress and assessment opportunities.



Supporting code cards are on the CD.

12.1 MOOD LIGHT

Example enclosure/artefact

Here are some design examples along with photos of an actual solution.



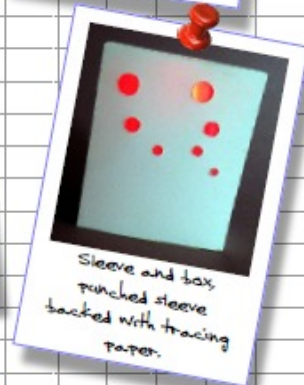
Showing parts of the Mood light apparatus



Die-cut box and lid with printed translucent paper.



Die-cut hexagonal prism with die-cut images backed with tracing paper as a diffuser.



Sleeve and box, punched sleeve backed with tracing paper.



Extending the project

Here are some further ideas that you could use to extend the project:

- Try adding a light sensor (LDR) to make it only operate when it is dark.
- You could add a heat sensor to make the colour sequence reflect the temperature.
- Could you make it do different colour sequences due to its orientation?
- You could set a new context such as a bedside lamp for a child's bedroom, which stays on for a set time before turning off, or has a controllable brightness or colour sequences.
- You could set a new context such as an outdoor garden lamp, that uses a light sensor (LDR) to make it operate only when it is dark and for a set length of time.

14 APPENDIX

- What's on the CD
- Systems information
 - BBC MicroBit
 - Crumble
 - Pixie:8 (Plus Picaxe & Genie)
 - RPi Pico
 - Arduino
- Making enclosures
 - Die-cutting
 - Found materials
 - Self-created nets
 - Fabrics
 - EVA foam
 - 3D Printing
 - Vacuum forming
 - Laser cutting
- Hints & Tips
 - General
 - BBC MicroBit
 - Crumble
 - Pixie:8 (Picaxe & Genie versions)
 - RPi Pico
 - Arduino Nano

