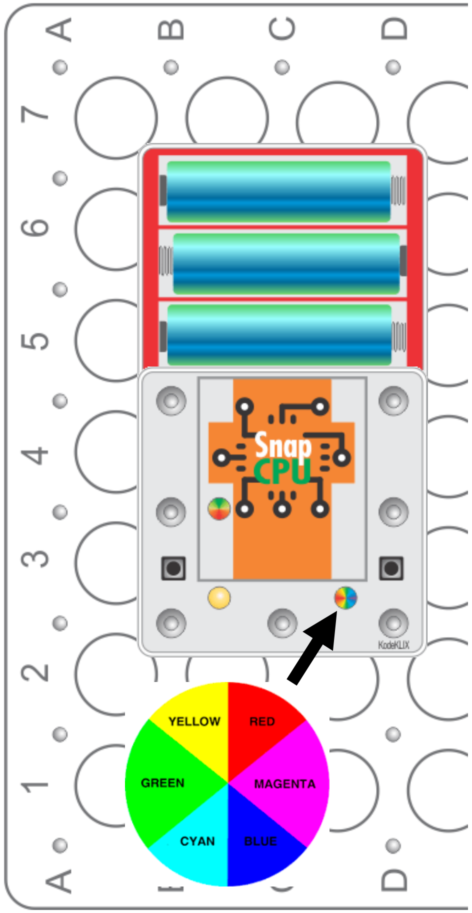
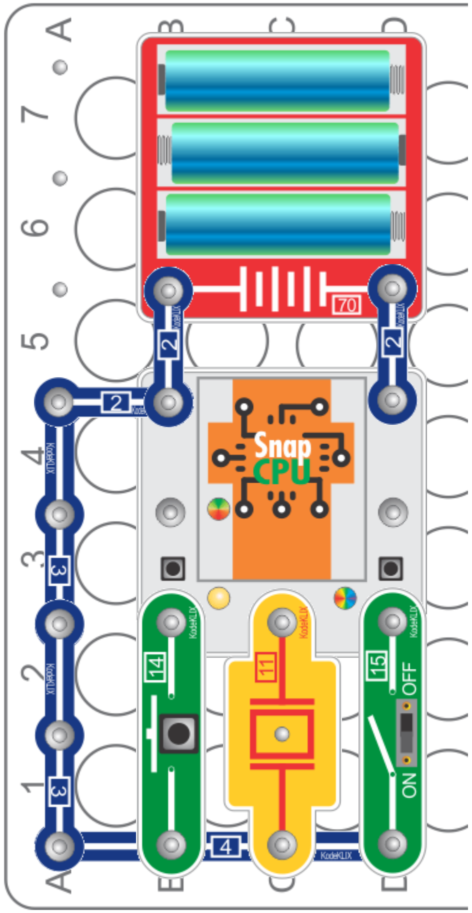
Example KodeKLIX® Circuits

Build these circuits to use with the pre-installed* code

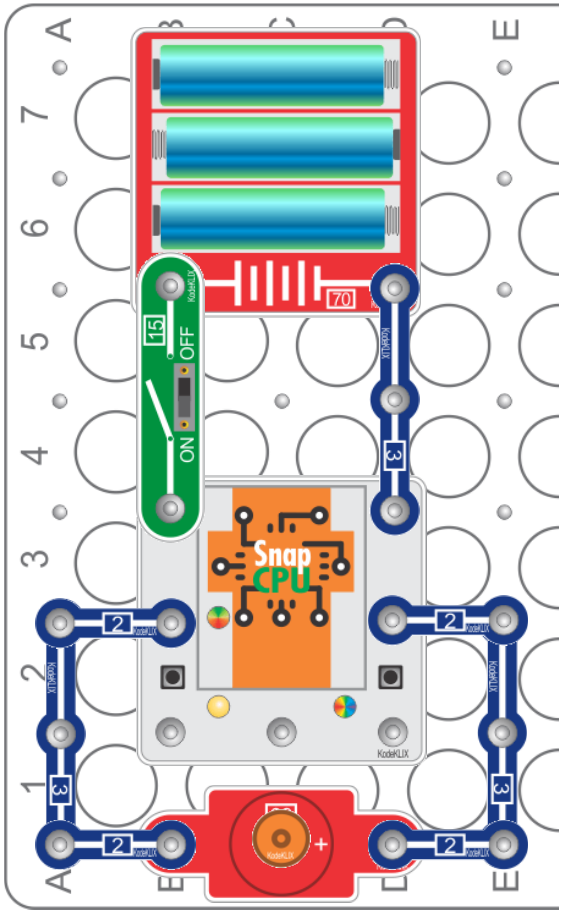
* The code is available can be re-downloaded to the SnapCPU™ at any time.



The RGB LED will cycle through 6 colours



Pressing [14] will buzz, sliding [15] will play tune



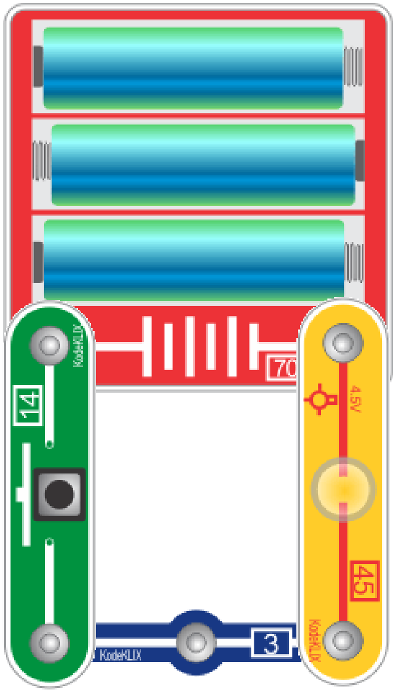
Pressing button C.3a or C.3b will spin motor

Completed □

Project 0.3

Let there be Light! (no SnapCPU required)

- Assemble the snap components as shown in "Circuit 0.3"
- For the Lamp [45] to light, the press switch [14] needs to be activated
- Rotate the lamp 180°; does it still light when the switch is pressed?

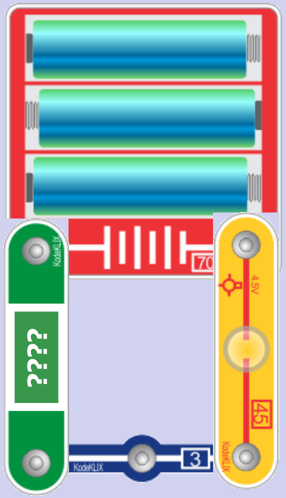


Circuit 0.3

Challenge 1: Lamps are a high power device. To work, the need a lot of electricity compared to an LED.

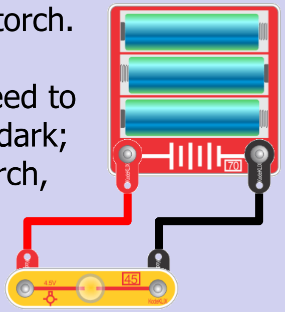
Modify the circuit to use different input s parts for [?] and identify which are able to allow this high power to flow.

Do the parts that work have high or low resistance; good or bad conductivity?



Challenge 2 :
Build your own torch.

Some project need to sense light and dark; if you need a torch, why not build it!

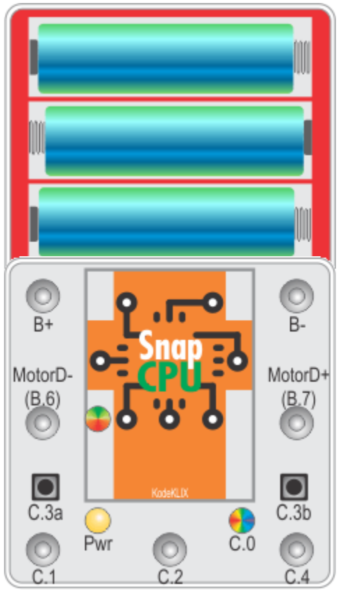


Completed □

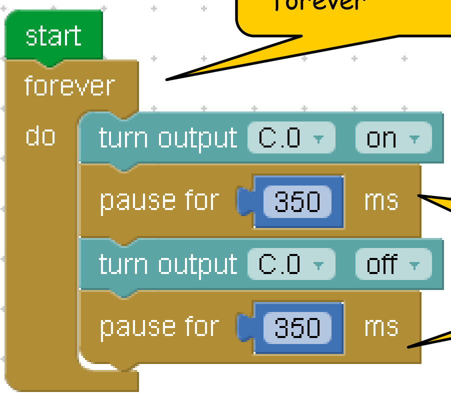
Project 1.1

Blinking LED

- Assemble the snap components as shown in Circuit 1.1
- Construct the following BLOCKLY code and download to the SnapCPU^{TM}
- Observe the LED marked C.0 blinking



Circuit 1.1



```
start
forever
do  turn output C.0 ▾  on ▾
    pause for 350 ms
    turn output C.0 ▾  off ▾
    pause for 350 ms
```

This code will loop "forever"

This code controls the default flash rate. 350ms is 0.35seconds "ON" followed by 0.35s "OFF".

Challenge: modify code to change blink rate of the LED

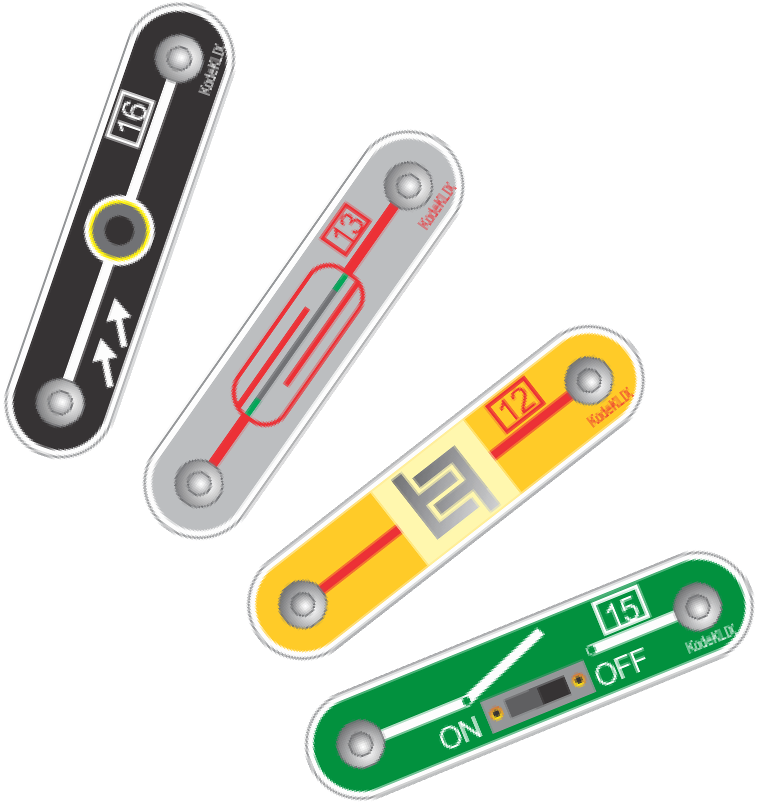
* In order to download the coding changes, your SnapCPU^{TM} will need to be powered-up and connected to a computer via its download link cable – refer download guide

* The SnapCPU^{TM} is designed so that it can be directly connected to the terminals of the battery box and the polarity of the connections is correct.

Note: this will immediately power-up SnapCPU^{TM} and start the code running.

Level 2: Single Input Circuits

- Flash LED C.0 when Input C.1 triggered by components connected to it. Test this using each of these input components in sequence
  - Push button
  - Slide switch
  - Magnet sensor
  - Light sensor

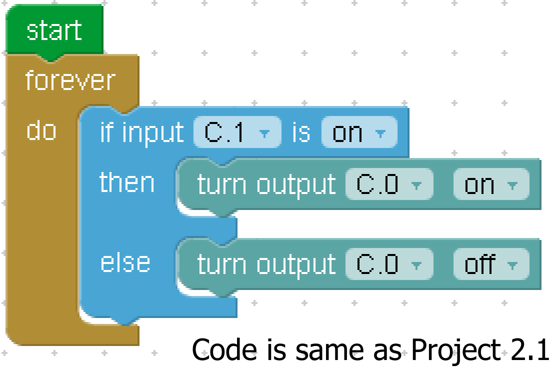


Completed □

Project 2.3

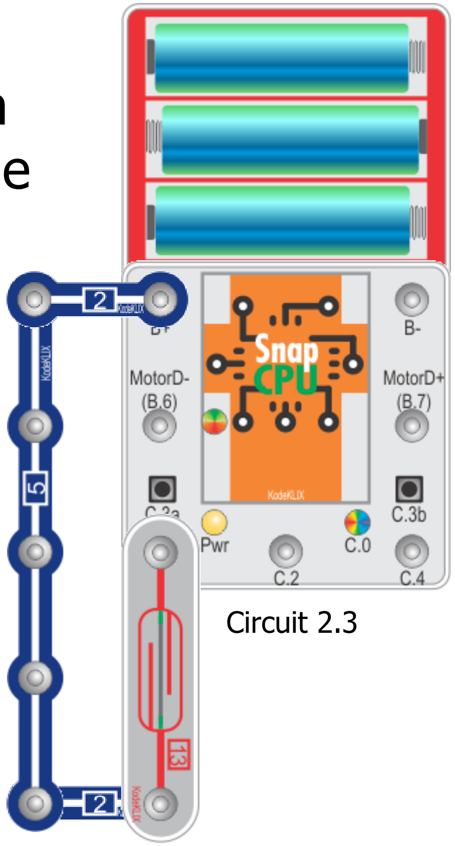
Magnet Sensor input to SnapCPU™

- Assemble the snap components as shown in circuit 2.3; Construct the BLOCKLY code below and download to the SnapCPU™
- Observe the LED C.0 light up whilst input C.1 is connected to "high" via [13]
  - "high" is activated by the disc magnet
- Review the following BLOCKLY code to understand what is happening



```
start
forever
do  if input C.1 is on
    then  turn output C.0 on
    else  turn output C.0 off
```

Code is same as Project 2.1



Circuit 2.3

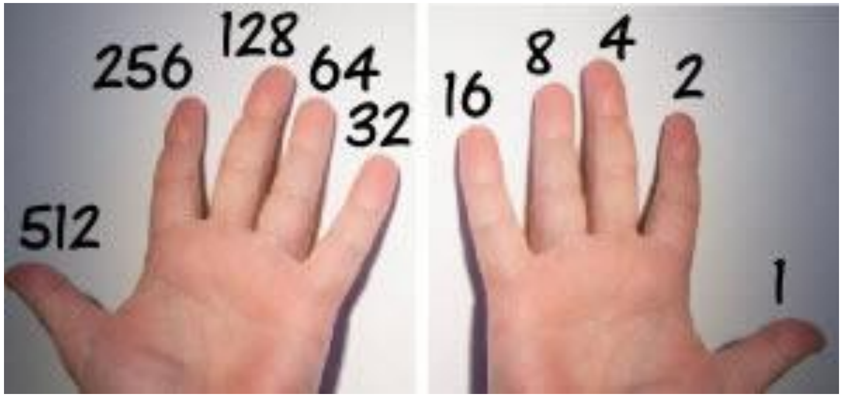
A magnet switch is momentary (only valid when magnet is present) so needs to be held until the code "sees" it. Lay the magnet flat, and ou may also need to move the magnet to find where the contacts in the sensor are.

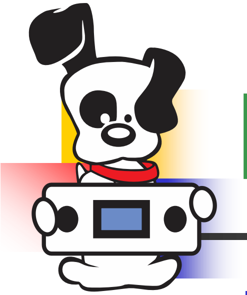
* The SnapCPU™ connections C.1, C.2, and C.4 are normally "pulled" low, or electrically speaking connected to –ve via a large resistance component; this gives the SnapCPU™ some definition when nothing is connected.

About Binary Numbers…

- Binary numbers are how computers count
  - Humans count in “tens” because we have 10 digits on our hands
  - Computers count with digits 0 and 1 only

Decimal	Binary	
0	0	000
1	1	001
2	10	010
3	11	011
4	100	100
5	101	101
6	110	110
7	111	111





Level 3: Quick Quiz

- Complete these logic truth tables for inputs A and B; you want check with a circuit you built earlier…

Hint: NAND is NOT AND; NOR is NOT OR

A	B	AND
0	0	
0	1	
1	0	
1	1	

A	B	OR
0	0	
0	1	
1	0	
1	1	

A	NOT
0	
0	
1	
1	

A	B	NAND
0	0	
0	1	
1	0	
1	1	

A	B	NOR
0	0	
0	1	
1	0	
1	1	

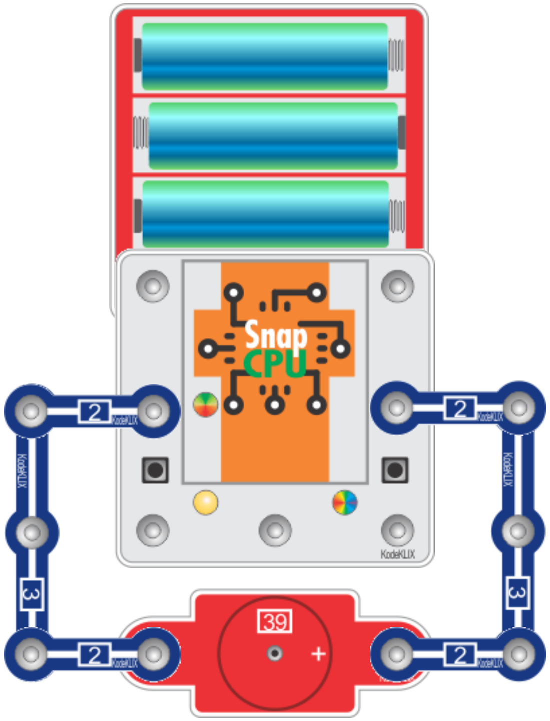
A	B	XOR
0	0	
0	1	
1	0	
1	1	

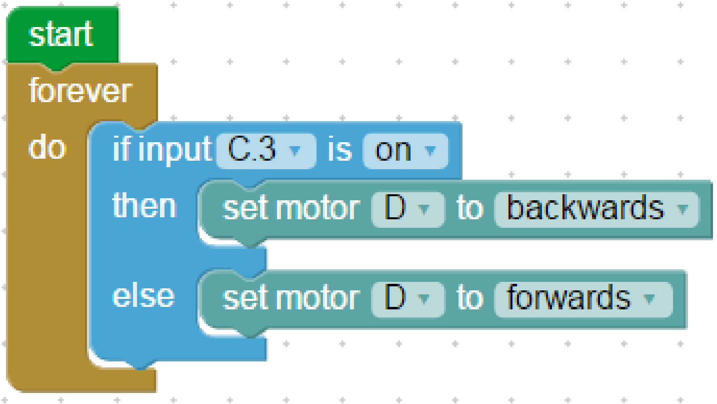
Completed □

Project 4.8

Directional Motor Control via the SnapCPU™

- Assemble the snap components as shown in circuit 4.6
- Construct the BLOCKLY code below and download to the SnapCPU™
- Pressing the built-in C.3 button will reverse the direction that the motor spins





```
start
forever
do   if input C.3 is on
     then  set motor D to backwards
     else  set motor D to forwards
```

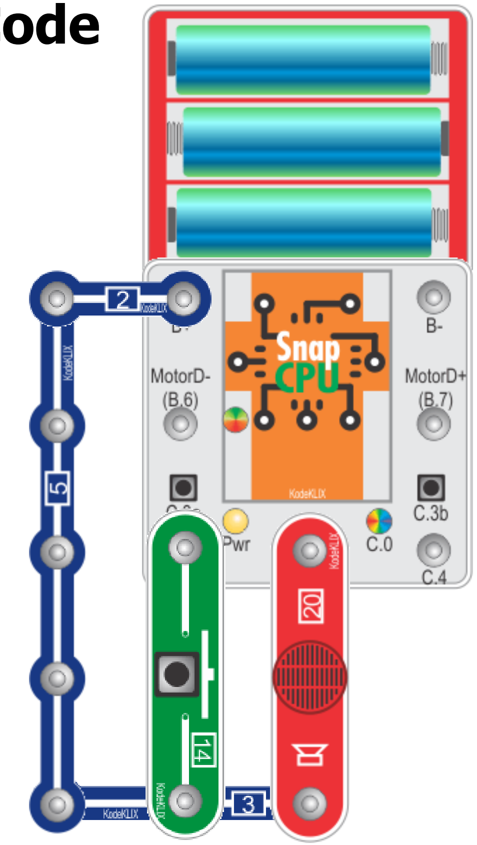
Challenge: investigate alll the other "set motor" options. Why does the motor with a fan take a while to "stop" after the command is given. Use a voltmeter to measure the voltage and polarity with each command.

Completed □

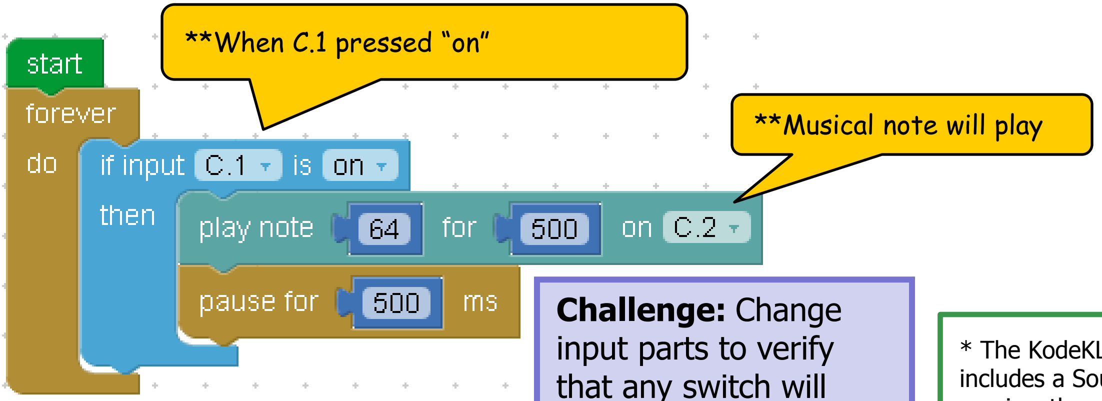
Project 5.3

User control of sound output, eg Morse Code

- Assemble the snap components as shown in Circuit 5.3; Construct the BLOCKLY code below and download to the SnapCPU^{TM}
- Press button [14] and observe response
- Review the following BLOCKLY code** to understand what is happening
- Try pulsing button [14] so as to send a Morse code message via the speaker [20]



Circuit 5.3



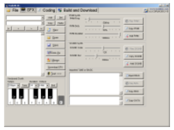
```
start
forever
do   if input C.1 is on
     then   play note 64 for 500 on C.2
            pause for 500 ms
```

**When C.1 pressed "on"

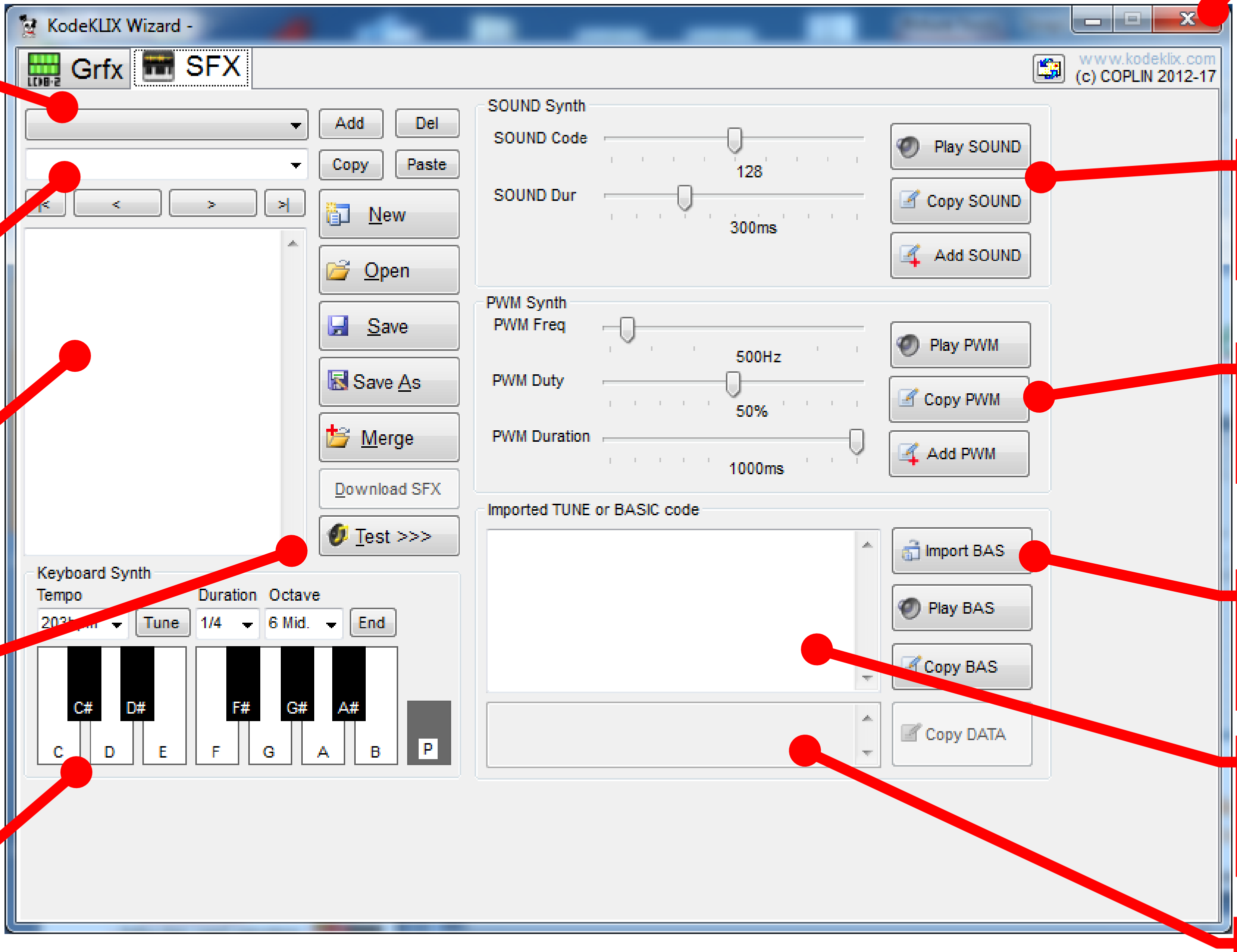
**Musical note will play

Challenge: Change input parts to verify that any switch will work with the code.

* The KodeKLIX SnapCPU^{TM} software includes a Sound FX simulator so you can preview the sound samples as you write your code.



KodeKLIX® SFX; CloseUP Guide



SOUND Synth

SOUND Code 128

SOUND Dur 300ms

PWM Synth

PWM Freq 500Hz

PWM Duty 50%

PWM Duration 1000ms

Imported TUNE or BASIC code

Keyboard Synth

Tempo 203

Duration 1/4

Octave 6 Mid.

Close

SFX ID

Descriptive Name

BASIC is shown here

Test the SFX Code

Keyboard Synthesizer

PICAXE Sounds

PWM Sounds

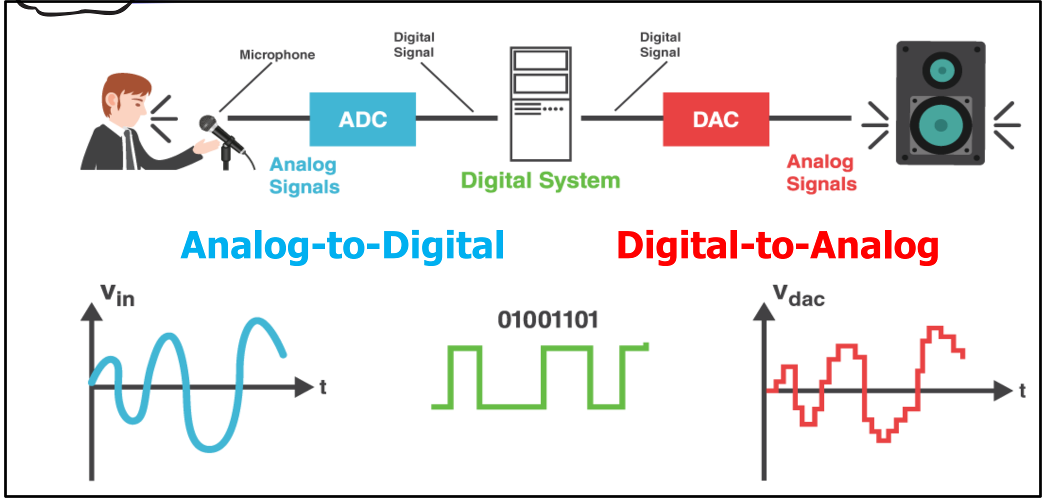
Import TUNEs

TUNE Format

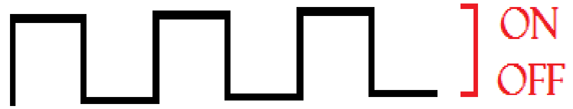
EEPROM Format

Tune Editor included with KodeKLIX®

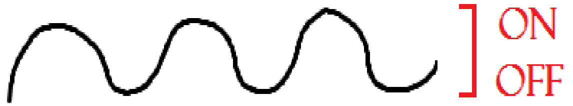
Analog-to-Digital Concepts



DIGITAL 0 OR 1, ABSOLUTE



ANALOG 0-255, FADES IN/OUT

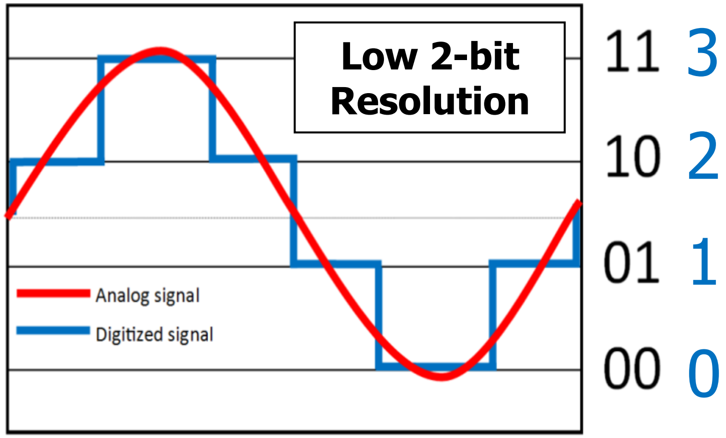


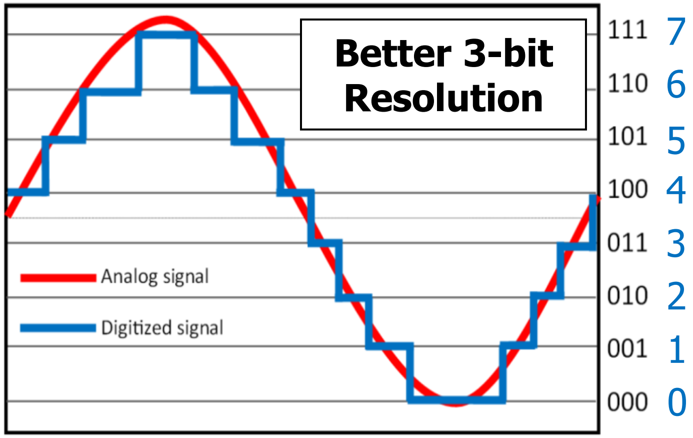
0-255 is 8-bit Resolution

Digitising

Analog signals are understood by digital systems only after they have been digitised, Digitising involves measuring the analog signal and storing those values. The scale used to measure determines the accuracy of the digitised version.

BINARY: 1's and 0's only; DECIMAL regular 0-9 digits used





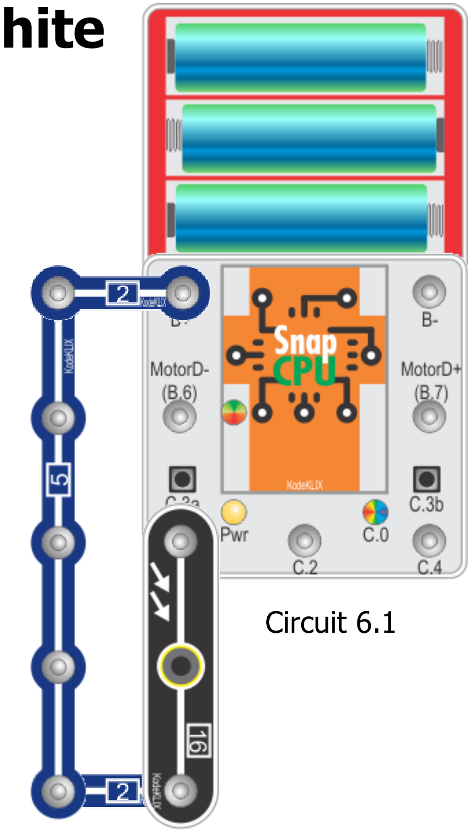
KODEKLIX™

Completed □

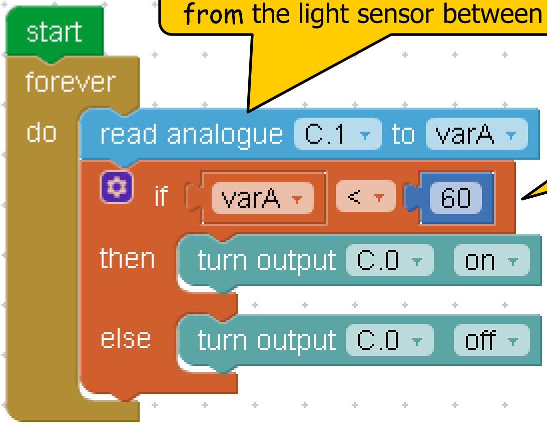
Project 6.1

Making decisions that are not black and white

- Assemble the snap components as shown in Circuit 6.1; Construct the BLOCKLY code below and download to the SnapCPU™
- In a dark room, LED C.0 should be lit up
- In a bright room, LED C.0 should be unlit
- Review the following BLOCKLY code to understand what is happening



Circuit 6.1



```
start
forever
do  read analogue C.1 to varA
    if varA < 60
    then turn output C.0 on
    else turn output C.0 off
```

**rather than on/off, get a scale value from the light sensor between 0 and 255

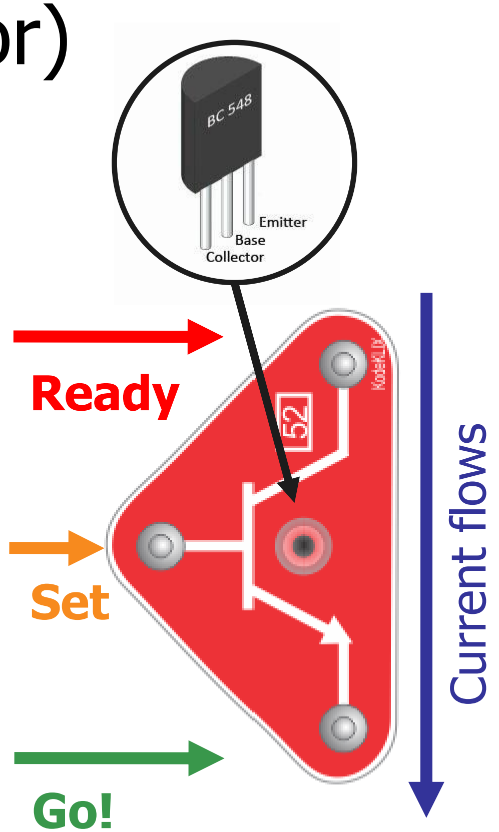
**adjusting the threshold will determine when the action will occur

Challenge: Try different settings to understand the sensitivity to light conditions. Part [16] has high resistance in the dark, and low resistance in bright light. Low resistance gives high varA values. Why? Because the low sensor resistance makes SnapCPU™ input closer to +ve voltage of the battery!

KODEKLIX™

Introducing the Transistor

- A transistor is a switch which needs a little energy to turn things ON/OFF that require a lot of electricity (like a motor)
- Transistors are easy to use if you understand the way electricity is connected to the motor, etc



Connected to power source; eg. battery

Ready

This controls the switching (by SnapCPU)

Set

This connects to the "load"; eg a motor

Go!

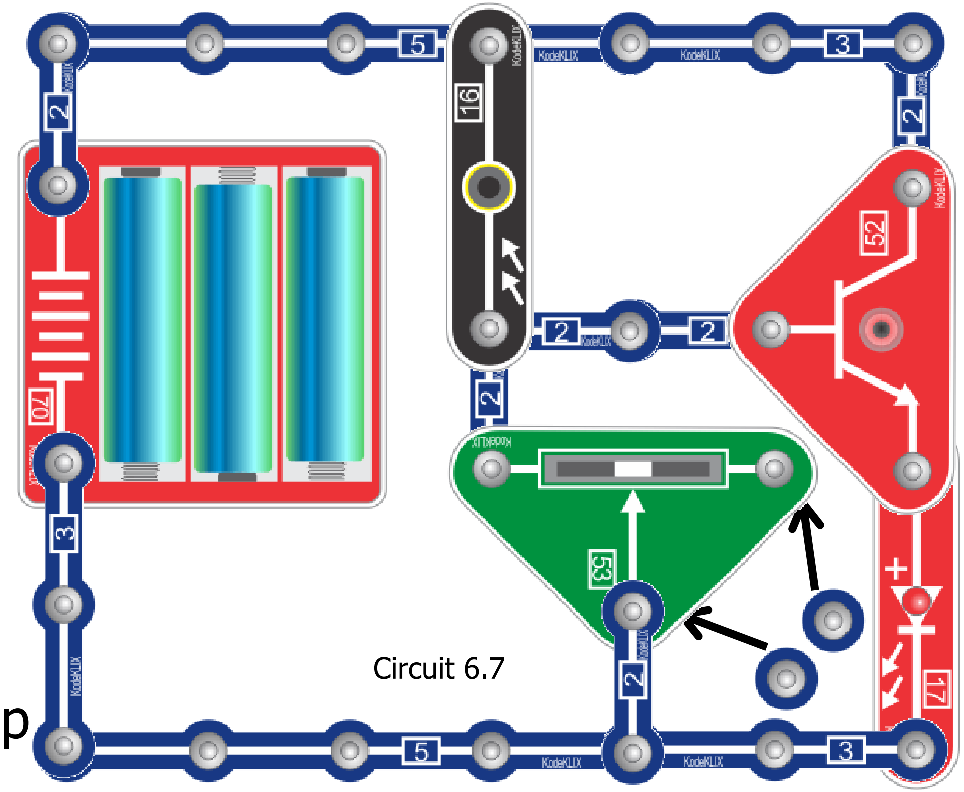
KodeKLIX® transistors components are modified internally so as to respond without the need for additional external components. The connections Ready, Set, Go! have technical names in the electronic industry like Collector, Base and Emitter (or Source, Gate, Drain for a more efficient FET device).

Completed □

Project 6.7

Adjustable Light Sensing without a SnapCPU™

- Assemble the snap components as shown in Circuit 6.7
- The adjustable resistor [53] can be used to set the “sensitivity” of the light sensor [16]
- Try this with other analog sensor such as temperature [33T] and moisture/touch [12]
- Replace the output LED with other output devices, eg motor or lamp



Circuit 6.7

Challenge: Try to research how this circuit works. To get you started, here are some clues. The resistance the light sensor [16] has and how it changes. The fixed resistor [34] has a value of 100k; how does the voltage to the transistor change and what value does it need to reach to switch.