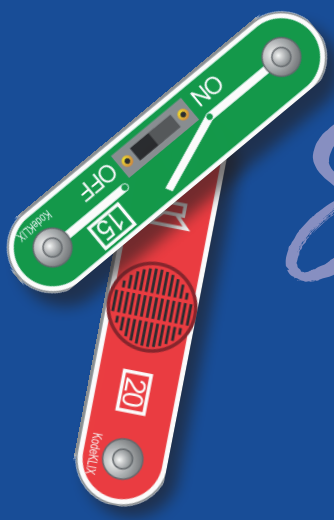
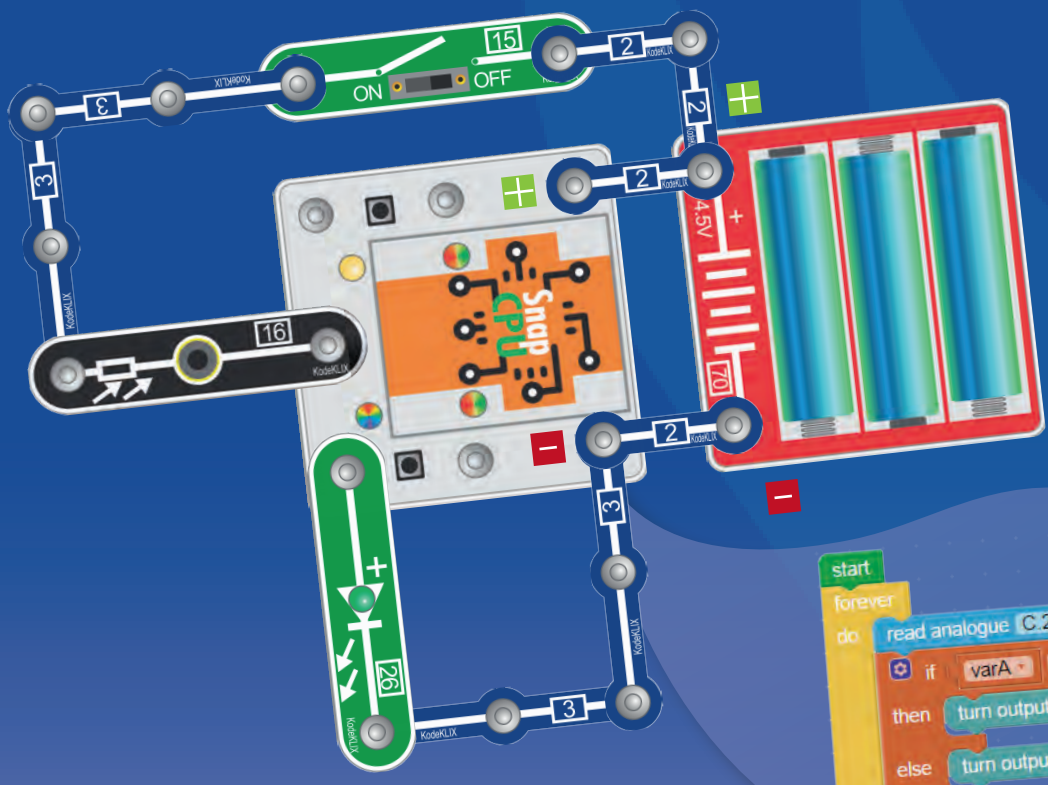
CORE ACTIVITIES

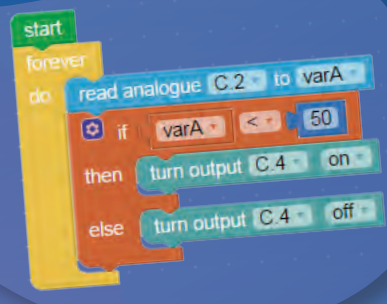
# STEM in Practice

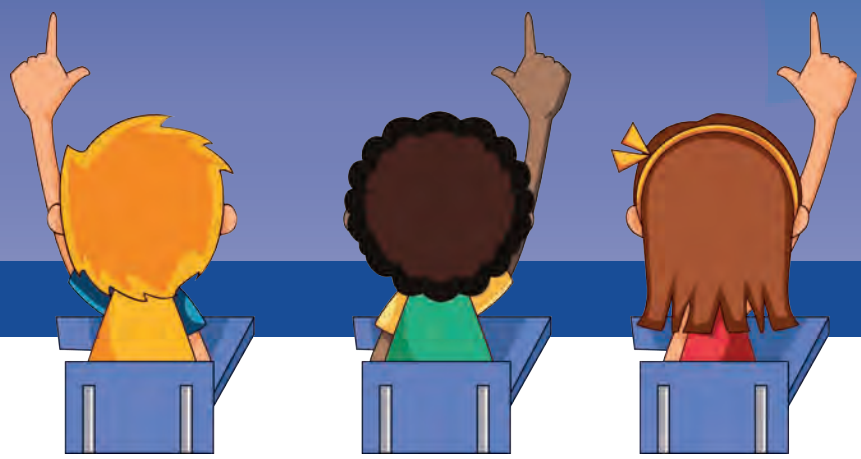
## with KodeKLIX®

- ✓ Define
- ✓ Plan
- ✓ Model
- ✓ Test
- ✓ Reflect
- ✓ Improve



AISWA SAMPLE

AISWA
www.ais.wa.edu.au

KODEKLIX
kodeklix.com

Peter Crosbie • Jan Clarke

## 1 SECTION 1: CORE CIRCUITS

Introduction to core electricity concepts, basic circuits and code



## 2 SECTION 2: SENSOR CIRCUITS

Basic electronic and code circuits using various sensor inputs



## 3 SECTION 3: ADVANCED CIRCUITS

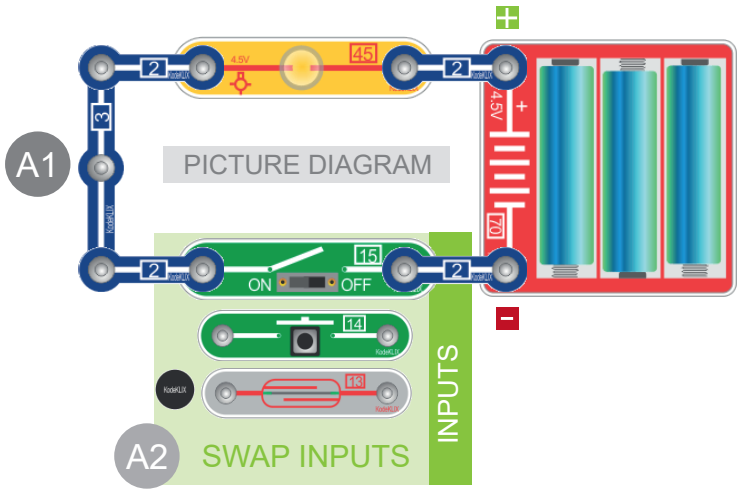
Advanced code circuits

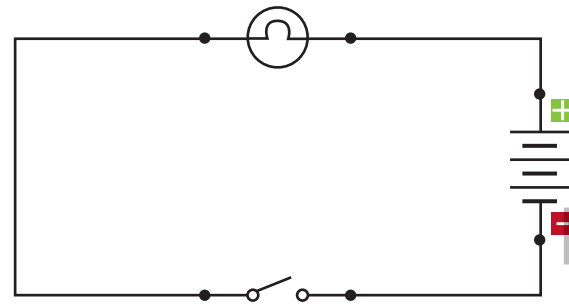
2 3 15 45 70
4 1 1 1 1

A1 BASIC CIRCUIT COMPONENTS

14 13
1 1

A2 ADDITIONAL COMPONENTS



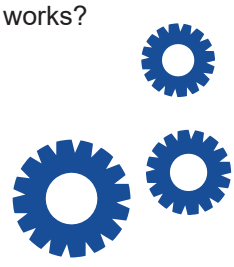
CIRCUIT DIAGRAM



## MAIN IDEA

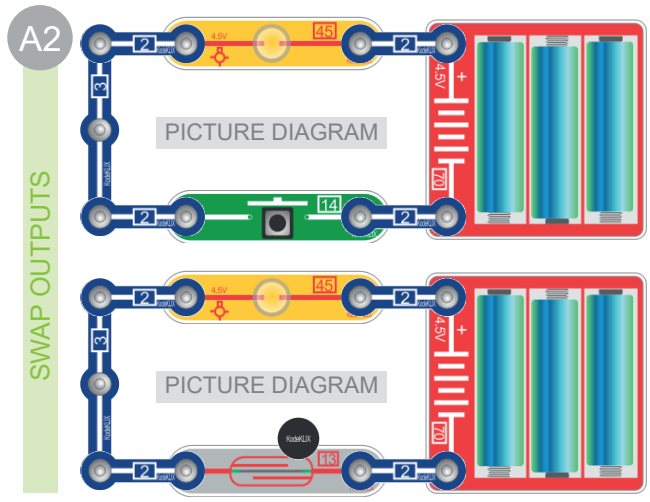
An INPUT SWITCH can control the ELECTRICAL CURRENT (flow of electricity) in a simple circuit.

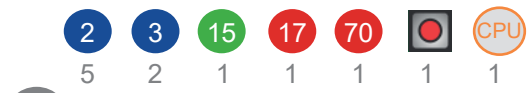
1. Look carefully at the component list and the coloured picture diagram. See how the components fit together.
2. Make the circuit.
3. Use the **SLIDE SWITCH** to control the circuit. OBSERVE the **LAMP** lighting up. Test this switch a few times.
4. Swap **BUTTON SWITCH** into the circuit. Test it.
5. Swap **REED SWITCH** into the circuit. Use the **MAGNET** to test it.
6. How do you think each switch works?



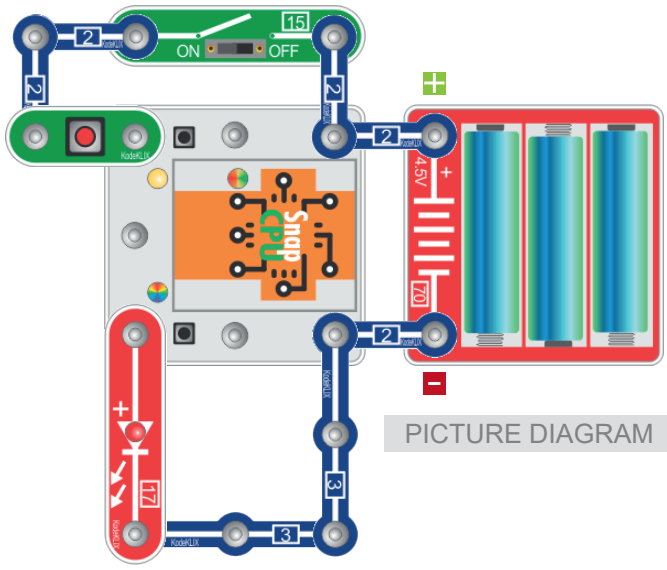
PICTURE DIAGRAMS and CIRCUIT DIAGRAMS show the same information in different ways.

1. Compare the components in the picture diagram and the circuit diagram.
   How do they look the same and different? What do the symbols mean?
2. Work out how to draw the simple circuit diagram for a different switch that you tested.

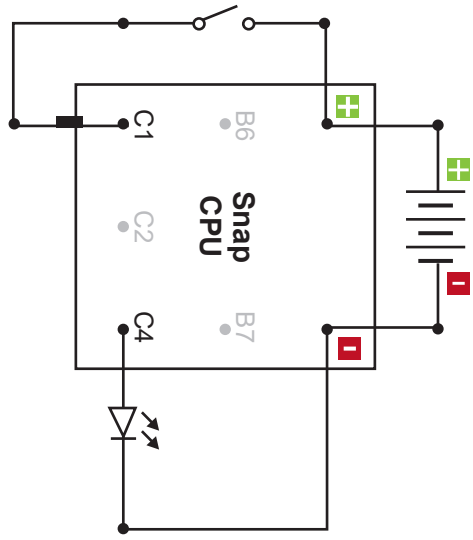





## B BASIC CIRCUIT COMPONENTS




PICTURE DIAGRAM




CIRCUIT DIAGRAM

### MAIN IDEA

The SnapCPU can control a variety of INPUTS using code downloaded from a computer. The button switch is connected to pin C2 and is the input for the circuit. The LED is connected to pin C4 and is the OUTPUT for the circuit.

#### The circuit

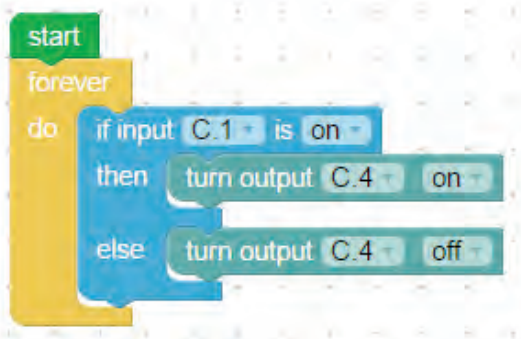
1. The SnapCPU requires power from batteries to work and a master switch for circuit protection.
2. The **BUTTON SWITCH** is connected to pin C1.
3. The LED is connected to C4.
4. Build the circuit. Make sure the master switch is turned **off**.

#### The code

1. The code is based on a loop command, and an '**IF**', '**THEN**', '**ELSE**' decision command block. The loop will run forever. **If** the switch is pressed (C1) the LED (C4) will turn **on**. **If** not, the LED will be **off**.
2. Simulate the code before downloading to the SnapCPU.

#### Test the circuit with the code

1. Download the code to the SnapCPU.
2. Turn the master switch to **on**.
3. Press and then release the button switch and observe the LED.
4. The LED should repeatedly switch **on** and **off** as you press\not press the switch.
5. If the circuit does **not** work, check your circuit very carefully. Are the power connections correct? Are any components in the wrong place or wrong way around? If the circuit **still** does not work, ask your teacher for help.



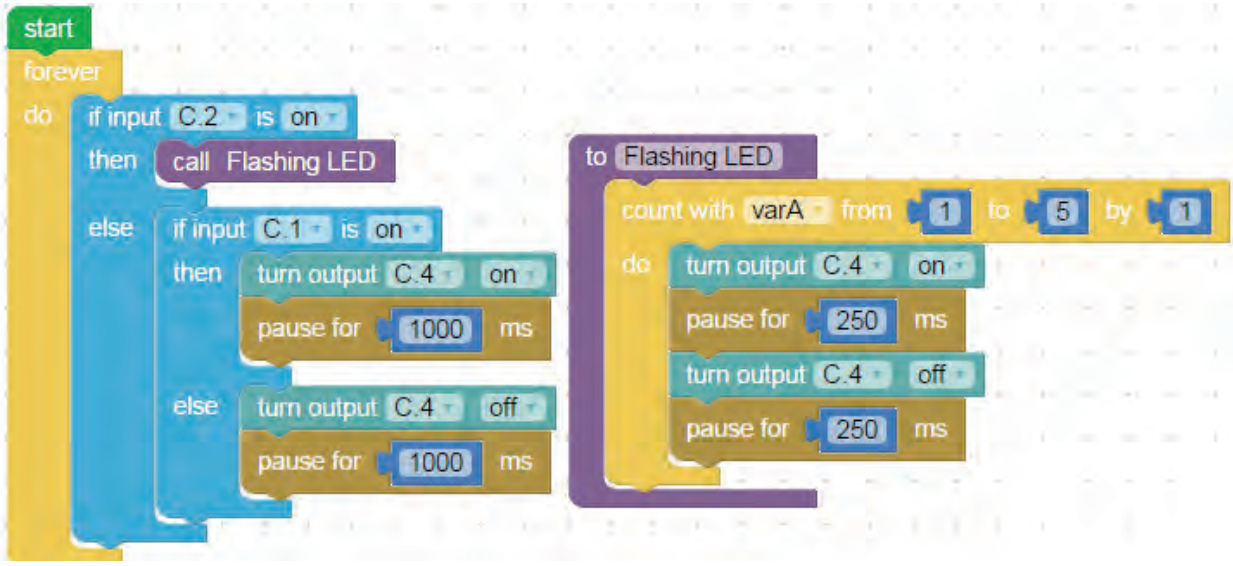

*Activity continues next page >>*

### The code

1. The code is based on a loop command, a procedure ***FlashingLED***, and two **if, then, else** decision command blocks and a COUNT WITH varA value.
2. The procedure is called ***FlashingLED*** and will only work when the switch pin C2 is pressed. When the switch is pressed the LED will flash 5 TIMES. This is because in the procedure code is a VARIABLE (VarA) which will count the loops from 1 to 5.
3. **If** pin C1 is pressed **then** LED is turned **on** for 1000ms then **off** when the switch is **not** pressed.
4. The loop will run FOREVER.
5. Simulate the code before downloading to the SnapCPU.

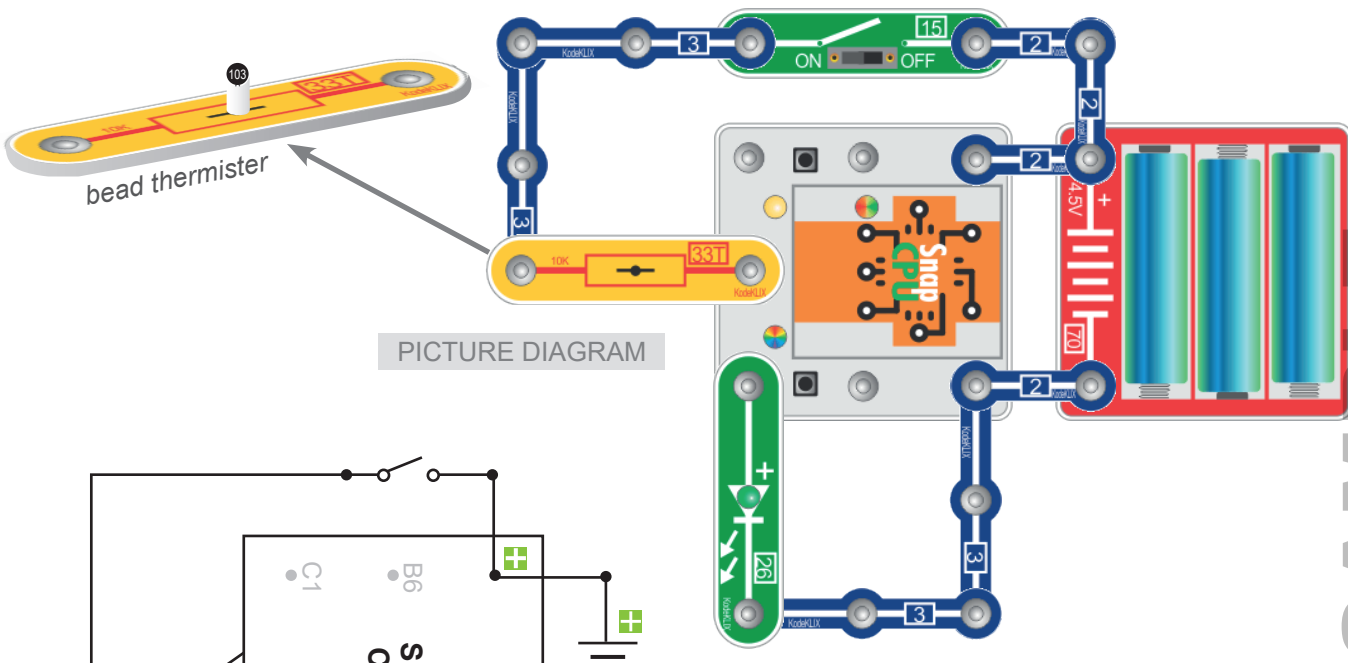
### Test the circuit with the code

1. Download the code to the SnapCPU.
2. Turn the master switch to **on**.
3. Press and then release the pin C1 button switch and observe the LED. The LED should repeatedly switch **on** for 1000ms and **off** as you press\not press the switch.
4. Press and then release the pin C2 button switch and observe the LED. The LED should turn **on** for 250ms and **off** for 250ms and repeats 5 times.
5. If the circuit does **not** work, check your circuit very carefully. Are the power connections correct? Are any components in the wrong place or wrong way around? If the circuit **still** does not work, ask your teacher for help.

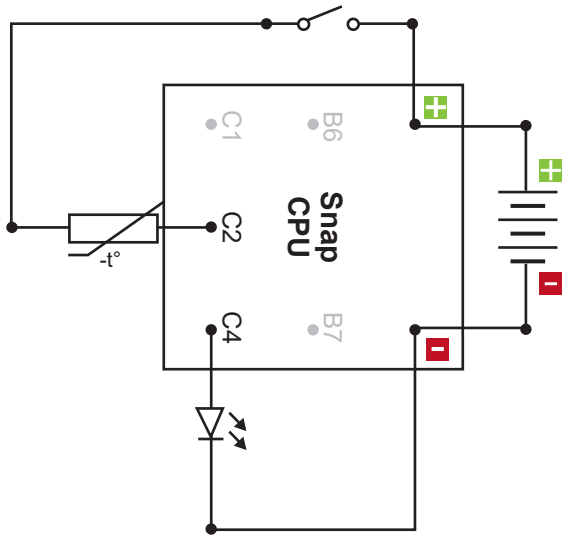


| 2 | 3 | 33T | 15 | 26 | CPU | 70 |
|---|---|---|---|---|---|---|
| 4 | 4 | 1 | 1 | 1 | 1 | 1 |

## B BASIC CIRCUIT COMPONENTS




PICTURE DIAGRAM




CIRCUIT DIAGRAM



### MAIN IDEA

The SnapCPU can control a variety of inputs using code downloaded from a computer. A heat sensor (thermistor) is used to switch on a piezo sounder when it senses heat. The code is based on loop and **if, then, else** commands and a *read analogue* variable settings. The number value you add will set the thermistor heat sensitivity. The number value can be found using the debug command block. The loop will run forever.

### The circuit

1. The SnapCPU requires power from batteries to work and a 'master switch' for circuit protection.
2. The **THERMISTOR** is connected to pin C2.
3. The **GREEN LED** is connected to pin C4.
4. Build the circuit. Make sure the master switch is turned **off**.
5. The LED will not light until the correct code is downloaded from the computer to the SnapCPU and the thermistor senses a change in light.

### The code

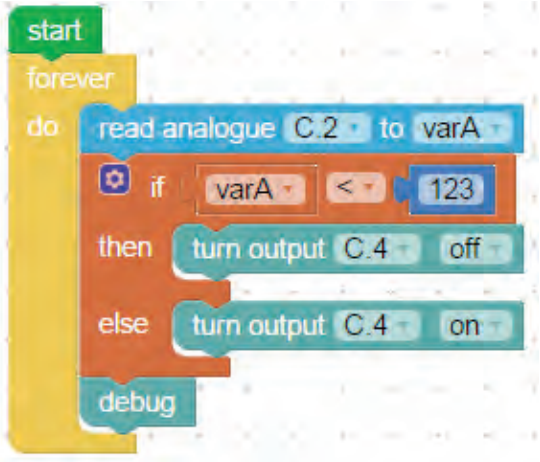
1. The code is based on a loop and *read analogue* block. The number value you add for *read analogue* will measure the heat on the thermistor.
2. The varA number value (<123) relates to the amount of heat needed to turn **on** the LED.
3. Look carefully at the coding. **If** the varA is less than the number value **then** LED is **off**, **else** the LED is **on**.
4. Write the code.
5. Simulate the code before downloading to the SnapCPU.

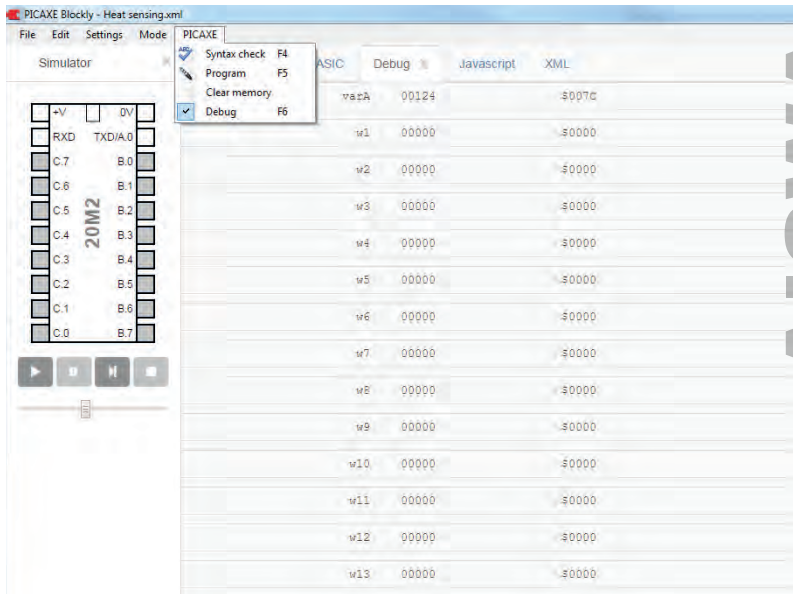
### Test the circuit with the code

1. Download the code to the SnapCPU.
2. Turn the master switch to **on** and observe the LED.
3. Heat the thermistor with your finger and thumb and the LED should turn **on**.
4. If the circuit does **not** work, check your circuit very carefully. Are the power connections correct? Are any components in the wrong place or wrong way around? Do you need to change the number value for ***read analogue*** (thermistor)?
5. If the circuit does **not** work, the varA number value may need to be changed. Go to 'debug process' (below). If the circuit **still** does not work, ask your teacher for help.

### Debug process

1. If the LED is always **on**, the varA number value needs to change. This is because the varA number <123 is too low.
2. Click on the PICAXE down tab and click 'Debug F6'. A table of data should appear and in the first line is varA number (00124 in example above). This number is a numeric value of room temperature.
3. Write down the number and click the 'Blocks' tab to return to your code.
4. Remember the code is based on **if, then, else**. **If** the varA number is less than (the debug varA number), **then** output (LED) is off. Reduce the debug varA number by 1 and change **if** varA < to this number.
5. Test the circuit with the code as described.

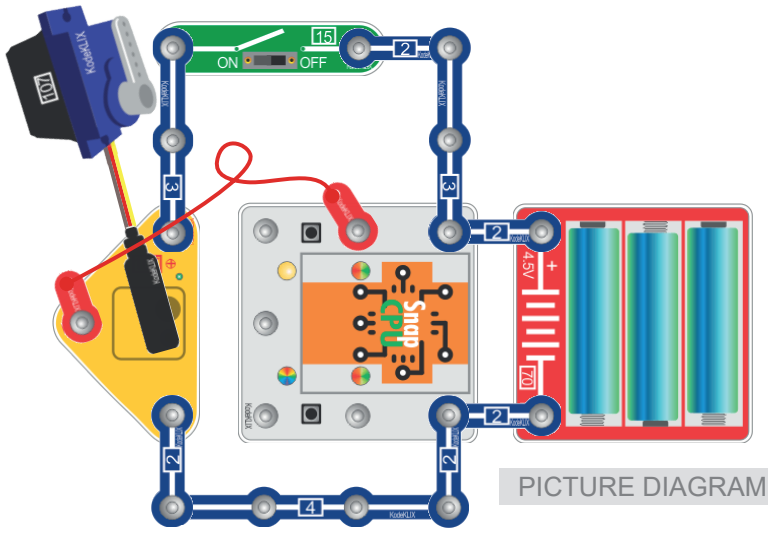


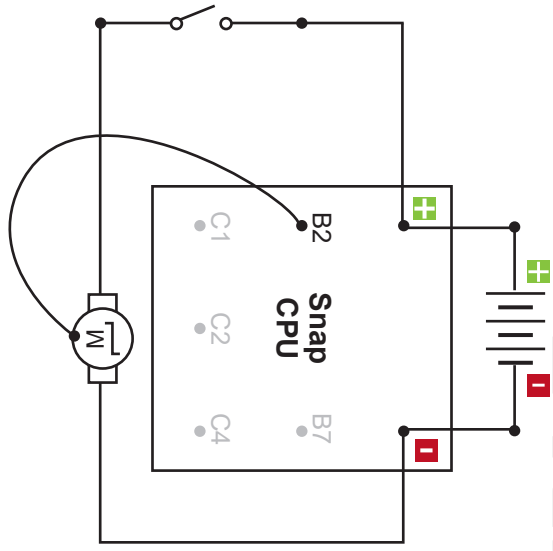

## EXPERIMENTING

Exchange the bead thermistor (33T) with temperature probe (33T) and exchange the LED for Motor (39). Use pin B7 for the motor and write code.
**Note:** You will need to download the code every time a change is made.


## A BASIC CIRCUIT COMPONENTS




PICTURE DIAGRAM




CIRCUIT DIAGRAM



### MAIN IDEA 1

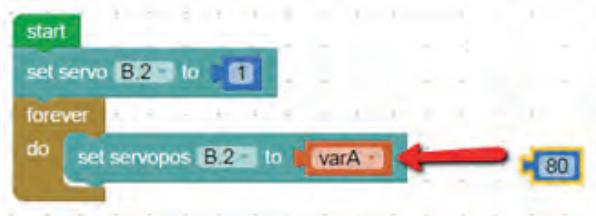
The SnapCPU can control servo motors using code downloaded from a computer. The servo is INITIALISED (woken up) with the SET command block. The servo is connected to pin B6 (coded as B2 because B2 is a special code just for servo motors).

#### The circuit

1. The SnapCPU requires power from batteries to work and a master switch for circuit protection.
2. The **SERVO POT** (107) has one of its pins connected to pin B2 (B6) on the SnapCPU by a flying lead.
3. Build the circuit. Make sure the master switch is turned **off**.
4. The servo will not move until the correct code is downloaded.

#### The code

1. The code uses a loop command and variable setting. The loop will run forever. The number value you add will set the POSITION of the servo motor.
2. Two commands control the servo motor. Set servo initialises (wakes up) the servo motor. Set servopos moves the motor to a position.
3. Write the code. The set servopos value is 80.
4. Simulate the code in the software before downloading it to the SnapCPU. On the simulator, pin B2 will change colour just once.




#### Test the circuit with the code

1. Download the code to the SnapCPU.
2. Turn the master switch to **on** and observe the servo motor working. It should move once to a position.
3. If it does **not** work, check your circuit very carefully. Are the power connections correct? Are any components in the wrong place or wrong way around? If the circuit **still** does not work, ask for help.

*Activity continues next page >>*