Simulate the States of Matter	Grade 8, Strand C Matter and Energy: Fluids	
# Lesson Plan	Coding Tool	Makecode and Micro:bits
	Cross-curricular	Coding

Big Ideas	**Specific Expectations**
Students will design and code their own "state of matter generator" using Micro:bits and MakeCode. This tool will randomly assign a student a state of matter (solid, liquid, or gas) which students will then model through a hands-on particle simulation. As part of the coding task, students will also program their Micro:bit to visually represent particles and their movement using the built-in LED matrix, illustrating how particle behaviour changes across different states. During the physical simulation, students will act as particles to demonstrate key principles of the particle theory of matter: 1. All matter is made of tiny particles. 2. Particles of a pure substance are identical. 3. Particles are always in motion. 4. There is space between particles. 5. Particles attract one another.	**C2.2** demonstrate an understanding of the relationship between mass, volume, and density **C2.3** explain the difference between solids, liquids, and gases in terms of their density, using the particle theory of matter **A2.1** write and execute code in investigations and when modelling concepts, with a focus on automating large systems in action

SCIENCE NORTH SCIENCE NORD

Materials	**Computational Thinking Skills**
1. Laptop or Chromebook with access to the internet 2. Micro:bits (It is preferred if you have microbits, but this lesson can be done using only the browser and in-software microbit simulator) 3. Optional: a. Masking tape to define areas for the different states.	Your text here is in Times New Roman 12pt and the heading is 12pt. Adjust the spacing in all the cells to fit on first page without losing text or going into footer.

Introduction

I In this lesson, students will explore how particles behave in the three states of matter (solid, liquid, and gas) using the particle theory of matter. This theory helps us understand what matter is made of and why different materials behave the way they do. It includes five key principles. Below, each principle is explained along with how it appears in solids, liquids, and gases.

This theory has 5 principles:

1. **All matter is made of tiny particles.**

Everything around us, from the air to the chair you are sitting on, plastic, metal, water, even you are made up of particles. These particles are too small for us to see, even with a light microscope! The basic building blocks of these are atoms of each element on the periodic table.

Solid: the particles are packed closely together, forming a definite shape. The particles stay in a fixed position, however there is still some movement (see number 3).

Liquid: The particles are still close, but they are not locked in place, allowing liquids to flow and take the shape of their container.

Gas: The particles are far apart and move freely, expanding to fill any space available.

2. **Particles of a pure substance are identical.**

A pure substance is made of only one kind of particle, which gives that substance consistent properties.

Solid: In a pure solid (like ice), all particles are identical and arranged in a structured pattern.

Liquid: When the same substance melts (in this case liquid water), the particles remain identical, even though their arrangement becomes more flexible.

Gas: As the substance becomes a gas (in this case water vapour), the particles remain the same, they simply spread out more.

3. **Particles are always in motion.**

Particles never stop moving (even in solids). Their motion changes with temperature and state. The warmer something is, the faster the particles are moving.

Solid: Particles vibrate in place but do not move past one another, giving solids a fixed shape.

Liquid: Particles slide past each other, allowing liquids to flow.

Gas: Particles move very quickly in all directions, colliding and spreading out.

4. **There is space between particles.**

The amount of space between particles is different in each state, and this affects density and compressibility.

Solid: Particles are very close together with little space between them, making solids dense and difficult to compress.

Liquid: There is more space between particles than in solids, allowing liquids to flow but still making them relatively difficult to compress.

Gas: Particles are very far apart, with large spaces between them, making gases much less dense and easy to compress.

5. **Particles attract one another.**

Particles are held together by attractive forces; the strength of these attractions changes with state.

Solid: Attractive forces are strong, keeping the particles locked in a rigid structure.

Liquid: The forces are weaker, allowing particles to move but still stay close.

Gas: Attraction is extremely weak, so particles spread out and move independently.

Action

Have each student visit the Micro:bit coding website, https://makecode.microbit.org , and start a new project.

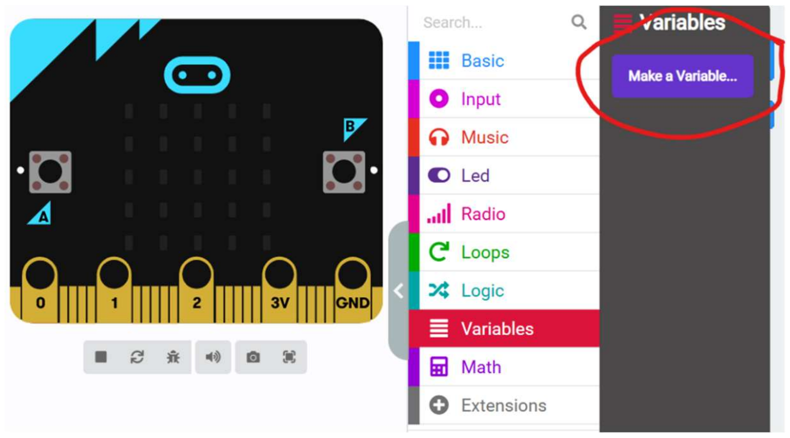
If you have not done so already, introduce the states of matter and their basic properties. You will explore the particle theory of matter in greater depth during the simulation. See the introduction section for additional details.

Explain to students that they will be completing a simulation to demonstrate the particle theory of matter. As part of this activity, they will need to randomly assign themselves a state of matter. There are many ways to do this; however, a fun and meaningful approach is to practise their coding skills by creating a state-assigning tool using the Micro:bit.

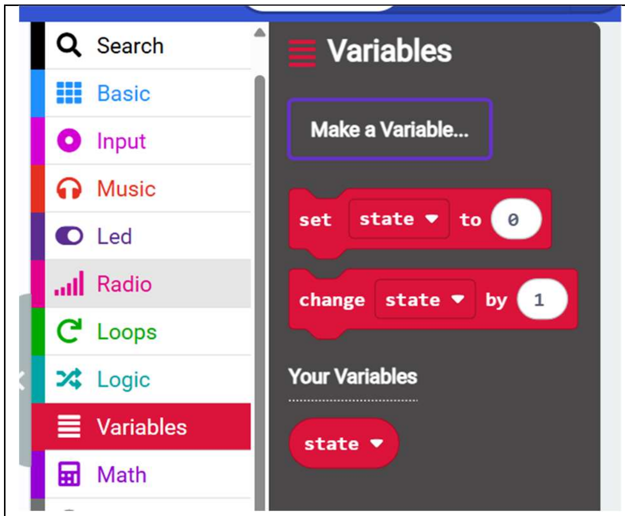
Step 1: Identifying and creating our variables

First, in order to assign a state, we need to set it as a variable because this is the value (solid, liquid, or gas) that the program will store and change throughout the simulation.

Click the *Red* Variables tab and select the "make a variable" block. You will need to create one variable named "state."



Once the variable is created, you should see it listed in the *Variables* tab under "Your Variables."



Step 2: Set your variables on start

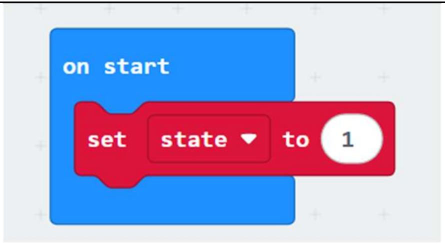
We will need to tell our program which state to assign when the program starts. We could introduce a text "block" (search *Text* and select the "block" with quotations) and name the states "solid", "liquid", or "gas," or choose to represent each state using a number variable. Using numbers is good practice, so we will use this approach. It also makes it easier for our program to generate a random state because there is a block designed to generate random numbers.

For our purposes:

- 1 will represent solid
- 2 will represent liquid
- 3 will represent gas

If there is not one already, drag in an "on start" block from the *Blue* Basic tab and nestle in a "set variable to" block found in the *Red* Variables tab. Ensure the "state" variable is selected from the drop-down menu and set the "state" to 1.

Your code should look like this:

 on start set state to 1	This means that when we start our program, our state will be set to 1, meaning solid.

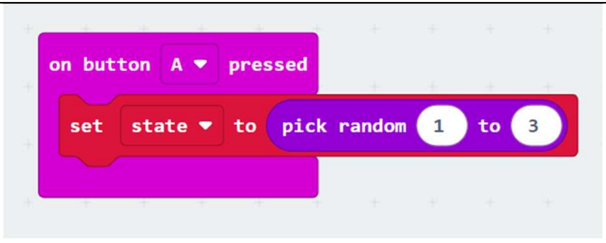
Step 3: Code to select a random state

We will need to code for generating a random state. To do this, we will select an input. You may use a different input, like “on shake”, however a button input is best so the state does not accidentally switch while moving around.

Therefore, we suggest dragging in an “on button A pressed” block from the *Pink* Input tab into your workspace. Inside that block, place a “set variable to” block found in the *Red* Variables tab. Again, ensure “state” is selected from the drop-down menu in the block.

Next, go to the *Purple* Math tab and locate the round “pick random _ to _” block, then place it inside the “set variable to” block. Since our states are represented by the numbers 1 to 3, we will use those numbers as our minimum and maximum random values.

Your code should look like this:

 on button A ▾ pressed / set state ▾ to pick random 1 to 3	This means that when we press the A button, our “state” variable will be set to a random value between 1 to 3, with 1 representing solid, 2 representing liquid and 3 representing gas.

Step 4: Coding a state indicator

Now we need to code an indicator to show which state has been selected, because currently there is no way to tell. This step also gives students an opportunity to be creative in how they represent the states of matter using coding. We will provide one example here, but there are many valid approaches.

First, if there is not one already, drag in a “forever” block from the *Blue* Basic tab, as we want this code to run continuously during the program.

Since we want the display to change depending on the selected state, we will introduce a conditional statement using the “If/then/else” block found in the *Teal* Logic tab.

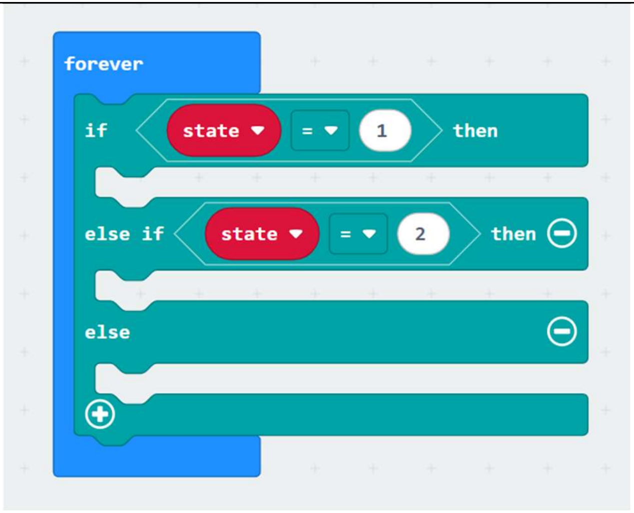
Nestle this conditional block inside the “forever” block. The condition we want to check is whether the state variable matches a certain value, so place a hexagonal comparison statement (also found in the *Teal* Logic tab) into the “if” section of the block. Make sure the equal sign is selected in the comparison. Then go to the *Red* Variables tab and drag the “state” variable into the left side of the comparison. On the right side, type 1 to represent the solid state. Your statement should read: “If state = 1”.

Click the plus button on the “else” portion of the “If/then/else” block to add another branch to the block, since we need to code for all three states.

For the first “else if” section, drag in another comparison block (with the equal sign) from the *Teal* Logic tab. Place the “state” variable on the left side again and type 2 on the right side to represent liquid. Your statement should read: “else if state = 2”.

The final section represents the only remaining possibility, when the state variable equals 3, representing gas.

Your code should loo like this:



```
forever
  if state = 1 then
  else if state = 2 then
  else
```

This means that while our code is running, if the random number generator selected a 1, representing solid, something will happen (we have not coded for this yet). Else if the random number generator selected a 2, representing liquid, something else will happen. Else neither of those conditions are true, the number generator must have selected a 3, representing gas, and something different will happen.

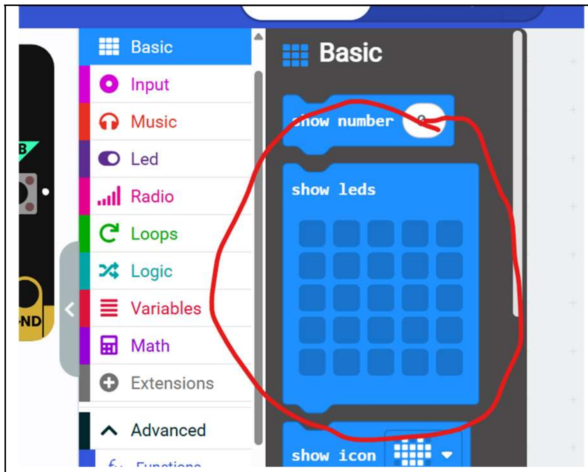
Step 5: Represent the states of matter using Microbit LEDs

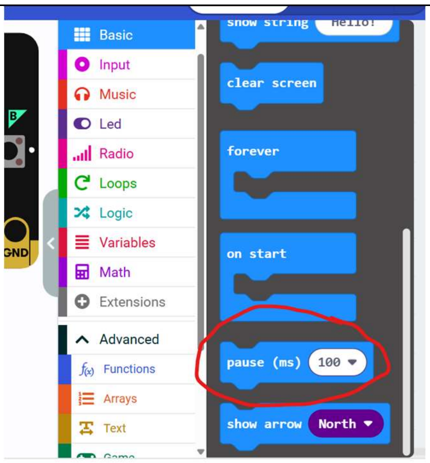
Now you will encourage students to pictorially represent the states of matter using the built-in LEDs on the Micro:bit screen. A turned-on LED will represent a particle. Remind students that they can consider both the spacing of the particles and how to code for movement. There are many valid ways to do this, ranging from simple flipbook-style animations to more complex designs. We will provide a simple example below.

Be sure to have a class discussion about the choices students make and their reasoning. This helps reinforce the core science concepts of the particle theory of matter and ensures that everyone is on the right track.

We recommend encouraging students to use the following blocks if they are stuck:

- The "show LEDs" block found within the *Blue* Basic tab, which allows them to choose which LEDs to light up.
- The "pause (ms)" block, also found in the *Blue* Basic tab, which can help them adjust timing when switching between different LED configurations.



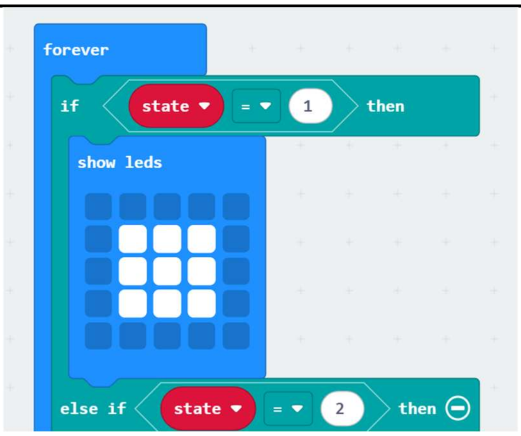


If students are struggling to come up with ideas, you can walk them through the following example.

For solid (state = 1), we chose to represent the particles by lighting up the middle nine LEDs in a tight cluster. We did this by clicking the LEDs we wanted to turn on inside the "show LEDs" block. Sinse the LEDs on the Micro:bit have space between them, this still effectively represents the concept that even in solids, particles, while very close together, still have a small amount of space between them.

You may also encourage students to find a way to represent particle vibration. For simplicity, our example does not include it. However, you can demonstrate how to add this by creating a second "show LEDs" block with the particles shifted slightly. Placing the two blocks in sequence within a loop will cause the display to toggle between the patterns, giving the illusion of particle vibration.

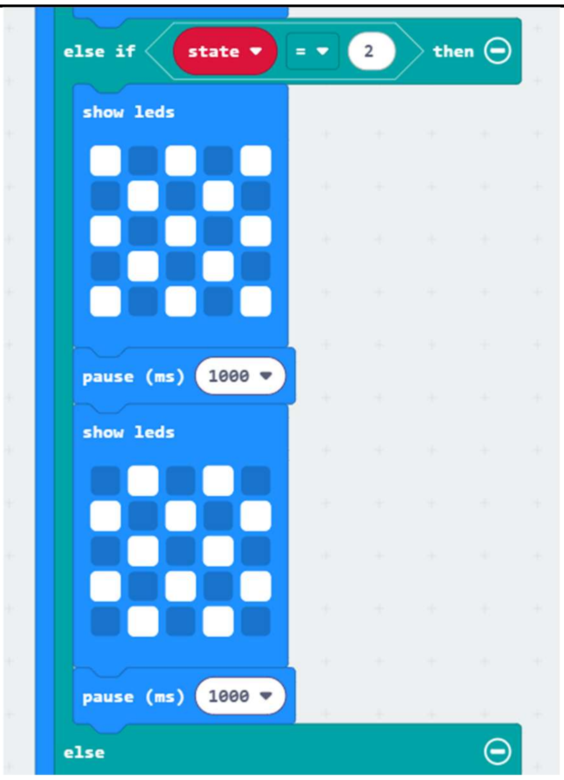
Our code looks like this:

 forever / if state = 1 then / show leds / else if state = 2 then	This means that if our variable has been randomly set to 1, meaning solid state, our microbit will always display this custom icon.

For liquid (state = 2), we chose to represent this by selecting alternating LEDs to be on, leaving a space between each one. To display particle motion, we added an additional "show LEDs" block with the alternate LEDs selected.

Students may notice that when they code their icons for gas, the toggling between the custom icons appears at the same speed as the liquid animation. To show the difference in particle speed between liquids and gases, we included a "pause" block after each icon in the liquid state. We found that a 1-second delay is enough to clearly show that liquid particles move more slowly than gas particles.

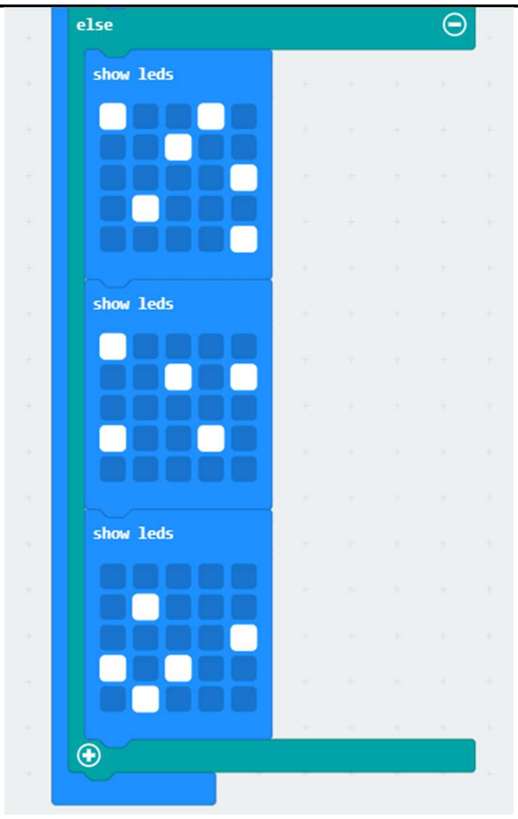
Our code looks like this:


`else if state = 2 then`
`show leds`
`# . # . #`
`. # . # .`
`# . # . #`
`. # . # .`
`# . # . #`
`pause (ms) 1000`
`show leds`
`. # . # .`
`# . # . #`
`. # . # .`
`# . # . #`
`. # . # .`
`pause (ms) 1000`
`else`
	This means that if the random number generator set our state to 2, representing liquid, our program will forever toggle between these two custom icons with a 1 second pause between each. This gives the illusion of particle motion, and the delay will differentiate the speed of the particles compare to that of the gas particles.

For gas (state = 3), we chose to represent this by placing three “show LEDs” blocks and randomly selecting five pixels in each one. This demonstrates that gases have a much lower density compared to solids and liquids, and the random placement of LEDs creates the illusion of fast, unpredictable particle motion. Because we did not include a “pause” block, the program cycles through the “show LEDs” blocks more quickly than it does in the liquid state, representing the much faster motion of gas particles.

Students may choose to add more “show LEDs” blocks to create additional movement frames. We recommend using between three and five blocks to achieve smooth, dynamic motion without over-complicating the code.

Our code looks like this:

	This means that if the random number generator did not set our "state" variable to 1 or 2, it must be 3, representing the gas state. Our program will then, forever, cycle through these three custom icons giving the illusion of fast, random particle motion.

Step 6: Download code and simulate!

If you have physical Micro:bits, download your code onto them. If not, you can still complete the activity using the in-browser digital Micro:bit simulator.

Setting up the simulation:

- Define areas in your room to represent the particle spaces for each of the three states of matter. You could use masking tape, string, or other materials to outline the boundaries.
    - Note: Create a smaller area for solids and a larger area for gases.
    - Note: Using square or rectangular spaces makes it easier for students to calculate density if you choose to extend the lesson.

Running the Simulation:

Have students press the A button on their Micro:bit to assign themselves a state of matter. Once a state appears, they move to the designated area and work with their peers to model how particles behave in that state. Their movement and arrangement should reflect the five key principles of the particle theory of matter:

1. All matter is made of tiny particles.
2. Particles of a pure substance are identical.
3. Particles are always in motion.
4. There is space between particles.
5. Particles attract one another.

Give students ~5 minutes to plan and coordinate their particle behaviour. Afterward, each group presents their motion to the class.

Encourage discussion throughout this process to reinforce understanding. Invite students to compare approaches, offer feedback, and refine their representations based on what they observe.

If time and interest allow, repeat the activity so students can experience behaving as different particle types.

Optional Gamified Variation

Once students understand the states well, you can introduce a fast-paced variation inspired by *Captain's Deck*. Call out "Change state!" and have students:

1. Press the A button to generate a new state.
2. Locate others in the same state.
3. Demonstrate the correct particle behaviour as a group.

The last group to organize themselves and properly demonstrate their motion is eliminated. Continue until there is a winner.

Before beginning, it is helpful to agree as a class on:

- How many students are required for each group
- What specific action each state must perform
-

Example rule set:

- Solid: Groups of 4. Students stand in a tight cluster and sway gently to represent vibration.
- Liquid: Groups of 3. Students spread out in a line and move together in one direction at a walking pace.
- Gas: Groups of 2. Students move quickly within arm's length of each other, high-fiving as they pass to simulate rapid collisions.

Any student without a group, or the last group to complete their action, is eliminated. Continue until a winner is determined or no further groups can be formed.

Consolidation/Extension

During the simulation, you may choose to have each group calculate their particle density. If you plan to do this, it is helpful to create square areas for each state to make the calculations easier. For example, you might designate the solid area as 1 m × 1 m, the liquid area as 2 m × 2 m, and the gas area as 3 m × 3 m.

Because the states are assigned randomly, you may end up with an uneven distribution of students. For this extension, it is most effective to evenly distribute your “particles” among the three areas so students can clearly compare densities across states.