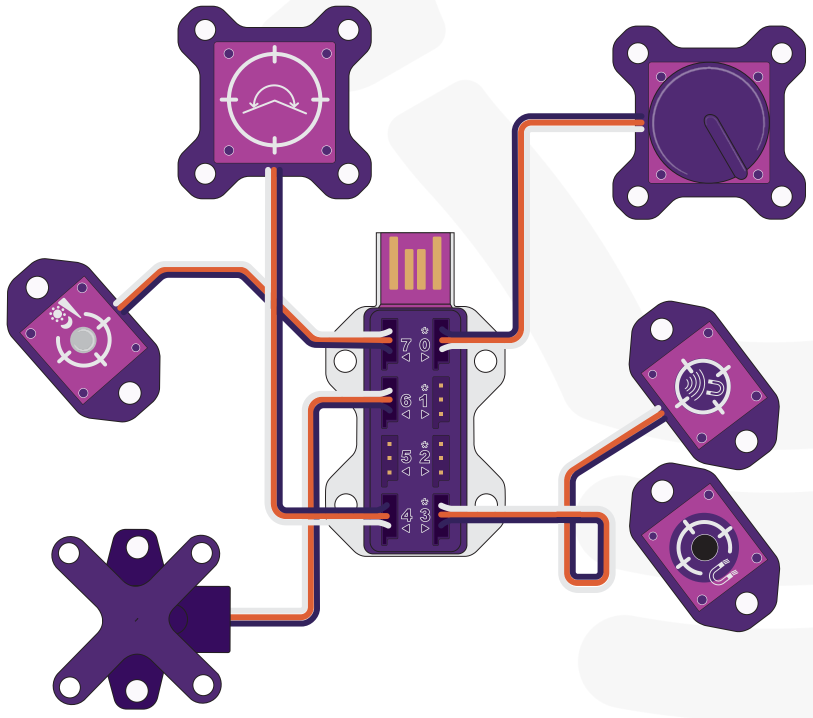
Invention Engine Lessons – Unit 2

Engineer to invent



Invention Engine

The fun of inventing is the journey

Document source number: 3.1.4.8.2.1

Contents

Invention Engine Lessons - Unit 2
by

microbric
motivate • create • educate

Illustrated by
Emma Dewar

Contributions from
Whymaker

The invention cycle

How does your idea for a world-changing invention become reality?

The answer: **Engineering!**

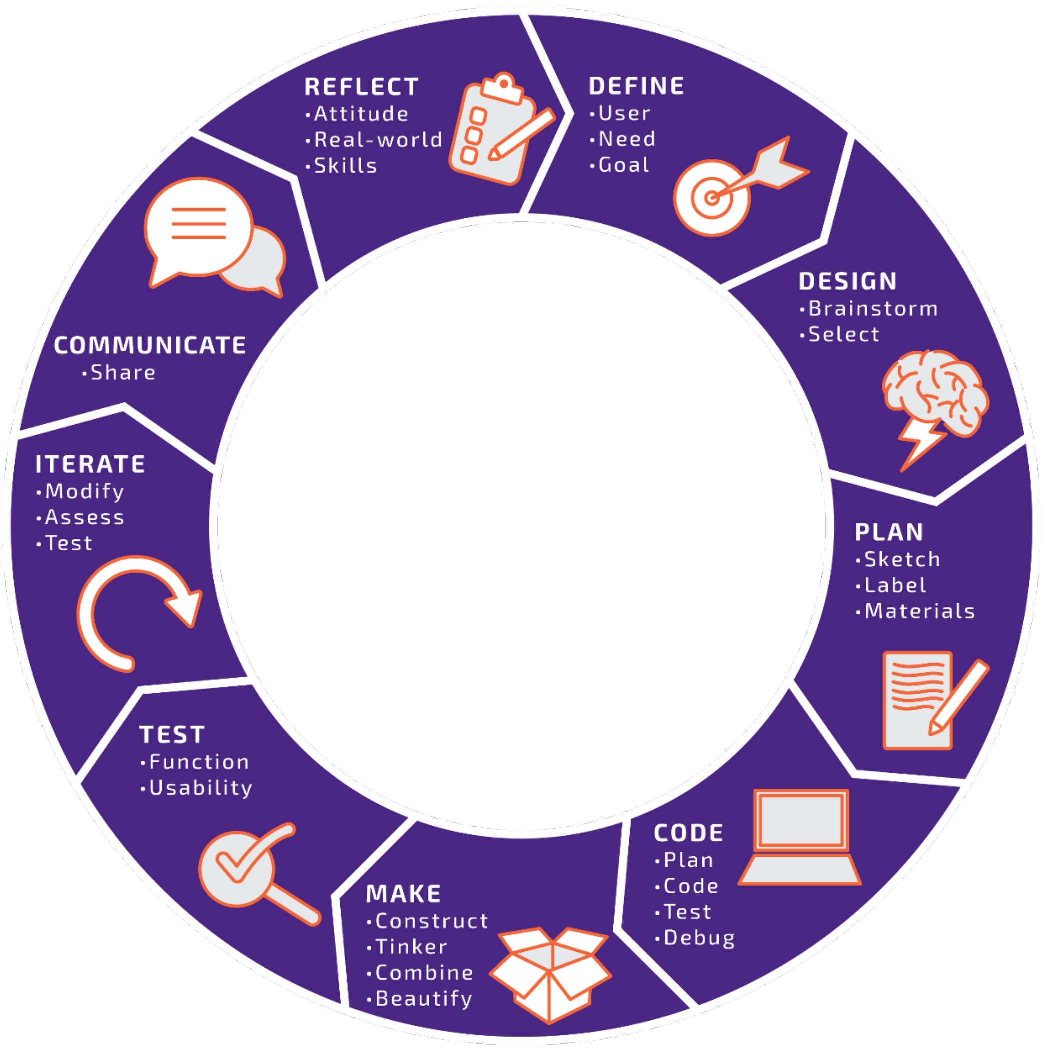
Engineering is the application of scientific principles to design, build, and invent a solution in response to a problem.

To create the best possible solution, engineers follow steps in the design cycle. There are many versions of the design cycle, and you may in the future develop your own cycle!

When using Invention Engine, we will follow the Invention Engine Cycle. We will work through each of the steps of the cycle to solve the problems in this Unit.

Grab your Invention Journal and let's start!

The Invention Engine Invention Cycle:



The Invention Engine Cycle

1. Define

When we approach a problem, we want to solve with an invention, it is a good idea to consider the problem in detail. To do this, we can use **empathy** to help us understand the person's (user's) difficulty or problem.

We can ask ourselves:

- Why is this a problem for them? Is it a need or want?
- What would a great outcome for the user look like? What is the goal?

Fuel your vocabulary

Empathy is the ability to understand and share the feelings of another.

Engineers use **empathy** to understand the problem they are trying to solve and to anticipate how people might use a product. This perspective often produces a great solution to a problem.

You can learn more about your users by talking to and observing them. The information you learn will help you put yourself in your user's situation to design a great solution.

Some tools to help you cultivate your empathy are:

- Be curious about the user
- 'Walk' in their shoes
- Keep questions open ended
- Seek feedback
- Keep your personal biases in check

2. Design

A design is a plan or drawing to show the look, and workings of your invention before it is made. In the design phase we brainstorm, select and plan.

Brainstorming:
Brainstorming is a technique that engineers and inventors use to generate, express, and organise ideas. It simply means thinking freely and suggesting as many ideas as possible, no matter how different the ideas may seem.

When brainstorming it may be helpful to:

- Stay focused! Brainstorming is a creative activity but keep your eye on the problem you are trying to solve.
- Produce ideas. Capture all ideas; at this stage, there are no 'bad' ideas.
- Respect ideas. If you are working as a team, remember that everyone's approach to solving a problem may be very different. This is what makes a team great!
- Combine and build on ideas. You can add, build, mix and match ideas to make an idea whole.

Select:
How do you choose just ONE idea?

- Cross out any idea that is unethical, illegal, or impossible.
- Does the idea solve the problem?
- Do you want to do it? Are you committed to it?
- Can you do it? Cross out any idea outside the project's **constraints.**
- Cross out ideas that don't meet the projects **specifications.**
- If your solution already exists, can your design improve the existing solution?
- Choose the most promising solution!

Fuel your vocabulary

Constraints are the 'things' that limit your design, like money, time, materials, etc.

For example, the constraints of using Invention Engine include: the bits can only be used with cardboard and the size of the bits means that you can't make micro-devices.

Specifications are the requirements that a solution, product or invention must meet. For example, be sturdy, be waterproof, fit in an adult's hand, etc.

3. Plan

A plan helps us think through our solution. This step will help you plan your physical invention. To plan your hardware, it helps to visualise the outcome of your invention.

Consider:

- What attributes is your solution going to have?
- The aesthetics - what is the invention going to look like?
- What decorations or instructions will be on it?
- Electronics - which bits will you use and where will they attach? Remember to include the Battery bit and the Hub.
- Materials - which materials will you use? Are they strong enough? Flexible enough?

In your plan, provide enough detail for someone else to be able to look at your sketch and build the final project.

4. Code

To learn to code using Invention Engine, revisit Unit 1.
Use the following checklist as a guide.

Coding checklist

- Plan your program
- Write your pseudocode
- Use the bit map to plug in your bits
- Setup blocks
- Start blocks
- Download your program
- Save your program
- Test your program
- Debug your program

5. Make

When we make inventions using Invention Engine, we follow the checklist below.

Make checklist

Construct

- Choose your cardboard
- Draw on cardboard
- Measure twice, cut once

Combine

- Create a grid
- Use the bit guide
- Perforate the external bit shape
- Perforate the rivet holes
- Secure the bits
- Fix and tinker

Beautify

- Make it awesome!

Construct with cardboard:

- Choose a cardboard box that is right for your invention. Study the size, robustness, and thickness of the box.
- Draw the outline of the project on cardboard.
- Cut out the outline of the project if needed (measure twice, cut once).
- Remember to consider the specifications and constraints for the invention.

Combine: See the Getting Started with Invention Engine guide for detailed instructions on how to add the bits to cardboard.

Fix and tinker: You may find yourself in the test – fix loop a few times, and that is ok! This loop is the same when you are fixing the code, the hardware or when you when the electronics and the carboard are combined.

Focus on the learning and enjoy the process.

- Take a deep breath! Remember that tinkering and fixing are both essential!
- Find the problem, then fix it. Start by narrowing down the source of the fault by decomposing the problem. Is the problem in the hardware or the code? If it is the code, where in the code is it? Etc.
- Track changes. Update the design and sketch to include the modifications made to your plan.
- Ask fellow inventors or teachers for their feedback and/or help.

Keep tinkering and adjust it until you feel satisfied.

Beautify: and make it awesome! Decorate and add instructions to your invention. Use creativity to make the invention intuitive to use and add your personal touch to make it yours.

6. Test

Testing helps us understand if our invention:

- Helps the user
- Addresses the need or want intended
- Delivers a good outcome.

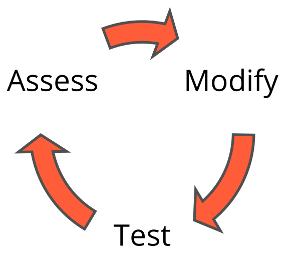
Two types of testing help us understand our invention:

Functional testing: This testing type ensures that your invention is doing exactly what it's meant to do. Functional testing requires that you look at the specifications and constrains in the design stage and test against them.

Usability testing: Also known as User Experience or UX. Is your invention easy to use and user friendly?

7. Iterate

The Engineering cycle is an iterative process. That means that it relies on repeated cycles of adjustments to improve.



When we iterate, we choose only one variable (or thing) to change at a time. If we modify more than one variable at the time, we will not be able to understand which variable is having an impact on our design.

Once we modify, we test the impact of that change on our invention. We then assess if the change has made our invention better. If it has, then we can choose if we want to keep the change permanently in our design or discard the change.

8. Communicate and share

Engineers need to be able to communicate to different audiences. For example:

- To users: explain how their invention solves the problem. The instructions for the use, and the care for the invention.
- To the marketing team: what the **benefits** and **features** are for the invention.
- To other engineers: how we arrived on our chosen invention.
- To the manufacturing team: explaining how to make the invention.
- To the legal team: to engage the team to work on the patents for the invention.

Fuel your vocabulary

Features are characteristics that your invention does or has.
For example, your invention may use rechargeable batteries.
Benefits are the outcomes or results that a user will experience by using your invention.
Continuing with our example, a benefit answers the question: How are rechargeable batteries a feature?
It means that you do not have to spend additional money buying new batteries, and rechargeable batteries are better for the planet, etc.

9. Reflect

A reflection is a thoughtful and intentional analysis of an experience.

Thinking about an experience or project, understanding what happened, identifying the parts you thought were meaningful and important, and making plans about how that may impact future experiences is an important part of the process.

Why? Because it helps you learn from the experience.

There are many approaches to reflection, but common themes are:

- Reflect about your personal growth
- Reflect about your invention
- Reflect about the process

When reflecting it is important to ask yourself a few questions.

What? Think about a summary of the experience to set the stage. What was the experience? How did I feel about it?

So what? What was learned that was surprising? What new knowledge or skills did I gain? What previous knowledge did I use? Did it go well? Why or why not?
What would I do different?

Now what? How does this change my future actions? What will I do in similar situations in the future?

Activity 1: A solution for drowsy driving

John started his job as a truck driver. He is concerned about drowsy driving.

Drowsy driving occurs when a person operates a vehicle, and they are too tired or sleepy to stay alert. Drowsy driving can lead to slower reaction times, erratic speed control, sloppy steering, and attention difficulties. Drowsy driving is responsible for 40% of truck crashes!

While on the job, 64% of truck drivers regularly experience fatigue and about 15% report having microsleeps behind the wheel.

Task

Using Invention Engine and the Tilt sensor bit, design and build an invention that solves John's problem.

Work through the steps in the Invention Engine Invention Cycle and record your progress in your journal.

The Tilt sensor bit

The Tilt sensor bit is an input bit that detects when it is tilted passed a certain point.

Much like the Button bit, the Tilt sensor bit has two states; tilted and not tilted.

The red LED on the Tilt sensor bit indicates the state of the sensor. The red LED is 'on' when the sensor is tilted.

You can find the programming blocks to control the Tilt sensor bit in the Sensing category.

Activity 2 – Mapping and divergent thinking

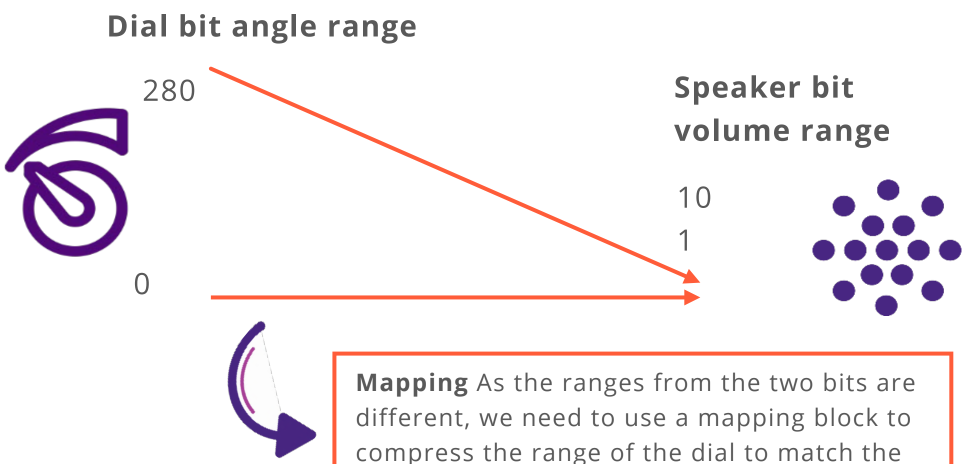
Have you ever wondered how a light dimmer, or a volume control works?
Let's find out.

Digital bits have only two states; on or off. In contrast, analogue bits have range in their output. For example, the Speaker bit has a range in volume and frequency.

The map function in Invention Engine Blocks only works for analogue bits.

The ranges within which the different bits operate do not match one to one. For example, the Dial bit has a range between 0 and 280 degrees whilst the Speaker bit has a range in volume between 0 and 10.

That is where mapping comes in! To 'match' the ranges in which the bits operate so that the two bits can interact in a proportionate way.



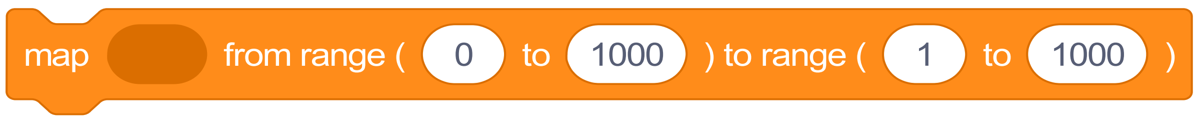
Mapping As the ranges from the two bits are different, we need to use a mapping block to compress the range of the dial to match the range of the speaker.

The mapping function

How to use the mapping function in Invention Engine Blocks:

Create a variable for the range you want to map.

Add the variable block here.



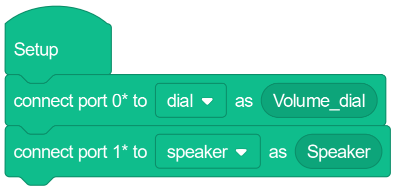
```
map (   ) from range ( 0 to 1000 ) to range ( 1 to 1000 )
```

Add the first bit's variable range.

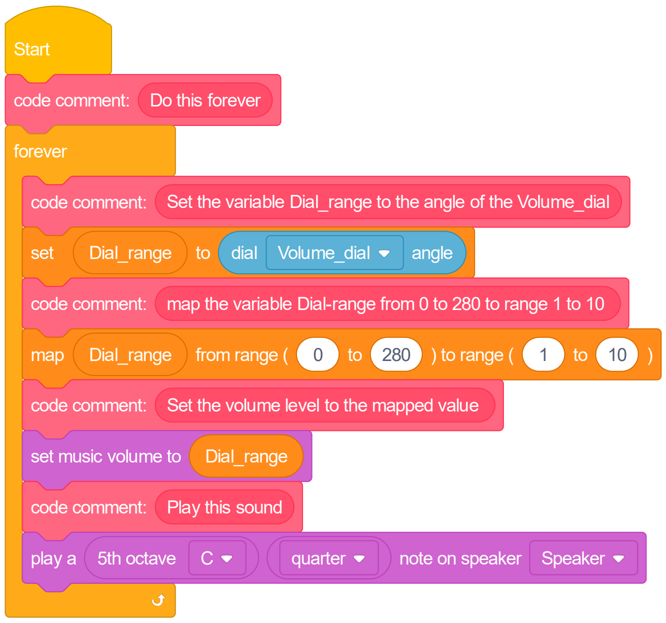
The lower limit in the first space and the highest limit on the second space.

Add the range of the second bit.

What do you think Invention Engine will do if you run this program?



```
Setup
connect port 0* to  dial ▾  as  Volume_dial
connect port 1* to  speaker ▾  as  Speaker
```



```
Start
code comment:  Do this forever
forever
    code comment:  Set the variable Dial_range to the angle of the Volume_dial
    set  Dial_range  to  dial  Volume_dial ▾  angle
    code comment:  map the variable Dial-range from 0 to 280 to range 1 to 10
    map  Dial_range  from range (  0  to  280  ) to range (  1  to  10  )
    code comment:  Set the volume level to the mapped value
    set music volume to  Dial_range
    code comment:  Play this sound
    play a  5th octave  C ▾   quarter ▾  note on speaker  Speaker ▾
↺
```

To map other bits, you can find the ranges for additional bits in the table below.

Name	Input range
Move servo (degrees)	0 -> 180
Rotate motor (SPEED)	0 -> 10
Set LED to level	0 -> 100
Display number on digital display	-9999 -> 30000
Display temperature on digital display	-99 -> 999
Display level on digital	0 -> 5
Send IR data	0 -> 255
Play a tone (Hz)	150 -> 5000
Set music tempo	60->250
Set music volume	1 -> 10

Let's Invent Activity 2: A serendipitous invention using the dial bit

Divergent thinking involves acquiring skills and considering creative and abstract ways to use them to develop multiple solutions to a problem. We like to call it 'thinking outside the box'!

Divergent thinking is great for inventing and thinking of ideas. Thinking divergently allows us to arrive to **serendipitous** or unplanned fortunate discoveries.

Task

Use your divergent thinking skills to use the dial and the map block. Think of as many ideas as possible. Look around you – can you see or think of any dials being used? Can they be used differently?

Work in your journal through the steps of the Invention Engine Invention Cycle to make your invention.

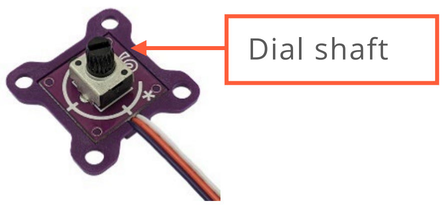
The Dial bit

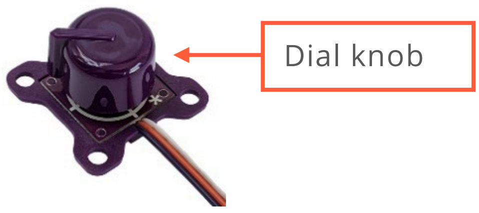
The Dial bit is a potentiometer. As the knob is turned, the output voltage changes in proportion.

The Invention Engine firmware reads the Dial bit's position in degrees. The range is 0 to 280 degrees.

NOTE: The Dial bit has an analogue output, hence it can only connect to the Hub in ports 0, 1, 2 and 3.

The Dial bit uses the four-hole housing, and it has a knob that attaches to the Dial bit's shaft.





Activity 3 – Snippets and movement

Snippets

Snippets are smaller chunks of code that can be reused in your code. Snippets are helpful when you need to repeat the same code multiple times. They can also make your program easier to read and follow.

Make a snippet:

- Set up blocks as usual
- Click on the snippets category block (orange)
- Select 'make a snippet'
- Name your snippet, in the example, the snippet is called 'LevelUp'.

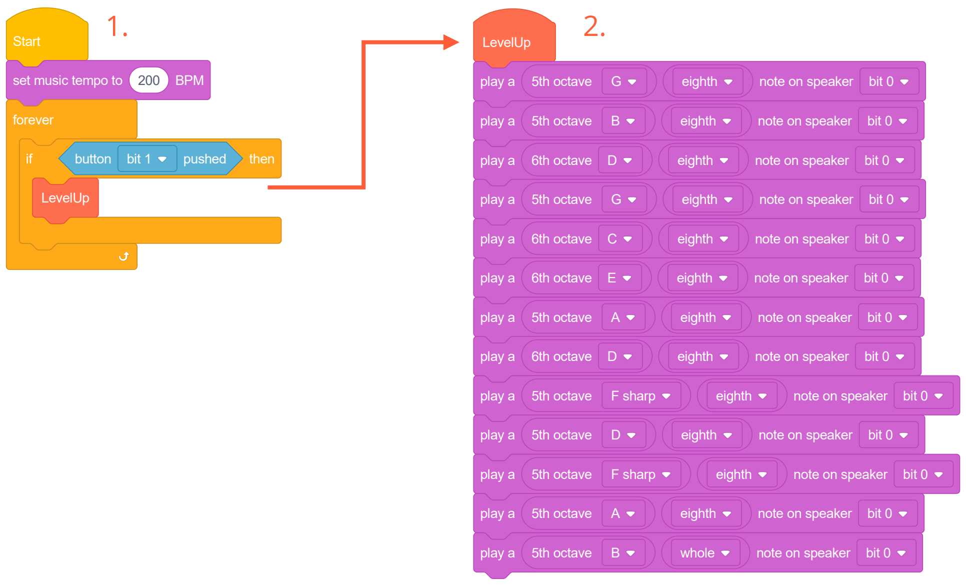
Tip: Don't use numbers, spaces or special characters when naming snippets.

- Drag the snippet block into the programming area as below
- Under the snippet block in the programming area, add the code blocks corresponding to what you want the snippet to do

In the example, the snippet will play the LevelUp song.

Start blocks:
Under the Start Blocks, code your program.
Once you arrive to the part of the sequence where the LevelUp song should play, insert the LevelUp snippet block.
When the program is being executed and the sequence arrives to the snippet block, it will perform the sequence under the corresponding snippet and then go back to the sequence under the start blocks.

You can use snippets as many times in a single program as you need to. You can also create more than one snippet and use all of them in the same program.

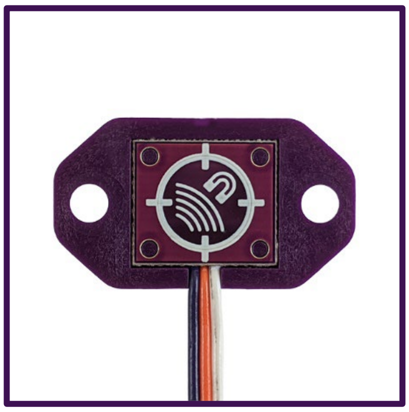


```
Start
set music tempo to 200 BPM
forever
    if button bit 1 pushed then
        LevelUp
```

```
LevelUp
play a 5th octave G eighth note on speaker bit 0
play a 5th octave B eighth note on speaker bit 0
play a 6th octave D eighth note on speaker bit 0
play a 5th octave G eighth note on speaker bit 0
play a 6th octave C eighth note on speaker bit 0
play a 6th octave E eighth note on speaker bit 0
play a 5th octave A eighth note on speaker bit 0
play a 6th octave D eighth note on speaker bit 0
play a 5th octave F sharp eighth note on speaker bit 0
play a 5th octave D eighth note on speaker bit 0
play a 5th octave F sharp eighth note on speaker bit 0
play a 5th octave A eighth note on speaker bit 0
play a 5th octave B whole note on speaker bit 0
```

The Magnet bits

There are two Magnet bits, one is a Magnet sensor bit and the other is a Magnet bit.



The Magnet sensor bit has a red LED which turns on when the Magnet bit is in close proximity (approx. 10mm). Larger and more powerful magnets are detected at greater distances.

The Magnet bit does not require a connection to the Invention Engine Hub.

The Magnet sensor bit and Magnet bit use the two-hole housing. There are no protruding parts, so there is no need to cut out the bit shape in the carboard.

The Magnet bit has only two states. Either a magnet is detected, or it is not detected. Coding using the Magnet sensor bit is like coding with the Button bit.

A use of the Magnet sensor bit is to detect the status of a door, window, drawer, or lid (opened or closed).

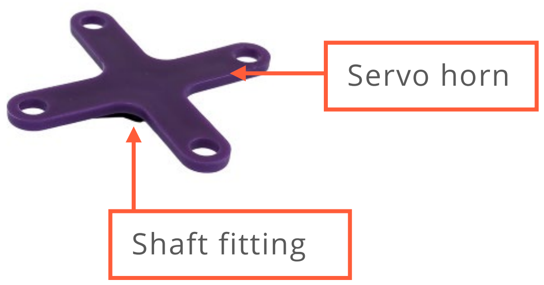
The Servo bit

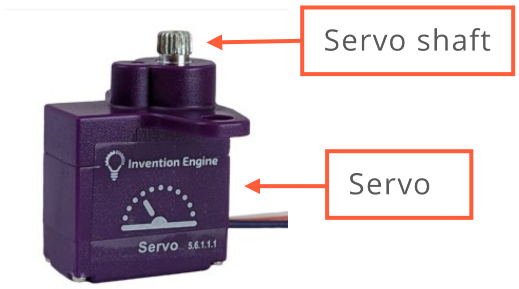
The Servo bit has two parts. The servo and the horn.

The horn attaches to the servo shaft by press fitting the shaft into the servo horn fitting (gently push it in).

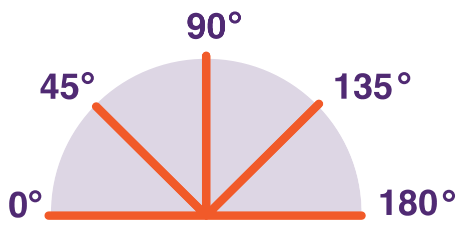
Use the stencil to punch the outline of the Servo bit's housing. The Servo bit uses the two-hole housing, and the servo horn uses four.

Attach the servo horn to the part of the cardboard you want to move and the servo to the part of the cardboard that will not move.

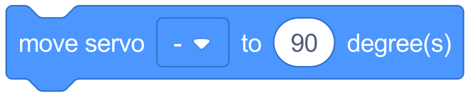




The Servo bit rotates from 0° to 180°.
To change the angle of rotation of the servo, change the degrees parameter. You can use the angle location in the picture to the right as reference.



The Servo bit does not make full rotations like the Motor bit (360°).

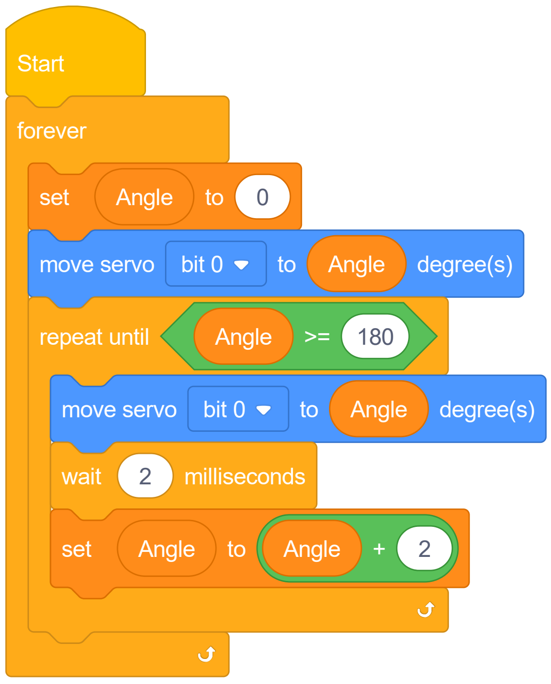


```
move servo [- ▾] to (90) degree(s)
```

There are no blocks to control the speed at which the servo will move. But it can be coded to move in incremental steps to manage the speed.

For example, in this code we want:

- the servo to move from 0° to 180° in incremental steps of 2° at a time.
- the action to continue until the servo has moved to the 180° position, then
- the servo to move back to the 0° position.

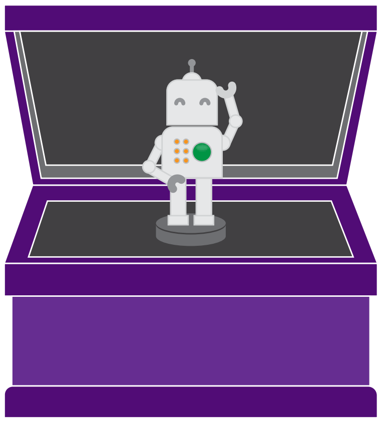


```
Start
forever
    set (Angle) to (0)
    move servo [bit 0 ▾] to (Angle) degree(s)
    repeat until <(Angle) >= (180)>
        move servo [bit 0 ▾] to (Angle) degree(s)
        wait (2) milliseconds
        set (Angle) to ((Angle) + (2))
    ↺
↺
```

Experiment with the code! Try:

- changing the number of degrees, the servo moves at each step.
- the number of milliseconds in the wait block.

Let's Invent Activity 3: A music box using the Servo and the Magnet bit



In the 18th century, there were no radios, or inventions to play music. Sensors, electronics, and electricity were all yet to be invented.

Music boxes were often carried by gentlemen in their waist coat to play music and delight their friends.

Then, music boxes were powered by clockwork. Therefore, restricted in size, quiet and very basic. Modern music boxes continue to delight us in different ways. Think of music jewellery boxes, baby mobiles and alarms.

Task

Using Invention Engine, let's invent an electronic music box to delight our friends with.

You must use: Snippets, the Magnet bits, and the servo in your invention.

Work in your journal through the steps of the Invention Engine Invention Cycle to make your invention.

Activity 4 –Boolean expressions and exploring light sensitivity

Boolean expressions

In code, a 'Boolean expression' or an 'expression', is a question that can be evaluated and resolved as being either 'true' or 'false'. Expressions can be used with other code, especially conditional code like 'until' blocks or 'if' blocks, to control the flow of a program.

Expressions let us compare two numbers or bits of data to each other.

These blocks use mathematical symbols to compare the left side to the right side of the expression.

Expression	**Meaning**	**Invention Block App** (operators blocks)
A = B	Is A the same as B?	0 = 0
A ≠ B	Is A not equal to B?	0 != 0
A > B	Is A greater than B?	0 > 0
A >= B	Is A greater than or equal to B?	0 >= 0
A < B	Is A less than B?	0 < 0
A <= B	Is A less than or equal to B?	0 <= 0

You can replace the 'A' and 'B' in the expressions with any value. You can also do computations to those values. For example, '(A + 2) > B' means: A plus 2 greater than B' which you can then evaluate to be either true or false.

In code, expressions work in a specific order. When an expression includes computations, the expression will complete the computations first. It will then compare the left side of the expression to the right side and resolve to either 'true' or 'false.'

Computers first evaluate the expression by completing any computations on either side of the expression. Then, they resolve the expression by comparing the left side of the notation to the right side. In other words, expressions are either 'true' or 'false', not numeric.

Try this - For these expressions, if A = 2 and B = 4, what does each of the following expressions mean (in other words, what is it trying to state) and what does each expressional statement resolve to (true or false)?

(A*2) = B
Meaning: 2*2 =4 (because A=2 and B=4)
Resolves to: True (because the two values are equal)

A >= B

Meaning: ______________________________

Resolves to: ______________________________

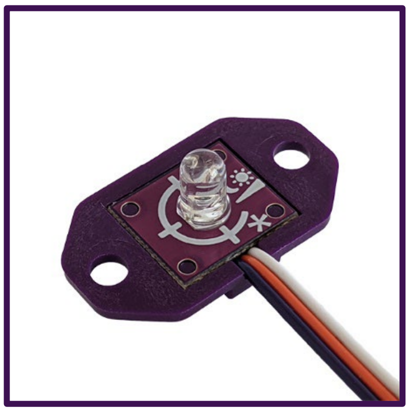
(A-1) < (B-3)
Meaning: ______________________________

Resolves to: ______________________________

Understanding what expressions mean and what they resolve to will help you follow a program just by 'reading' it. This important skill in programming is known as **tracing**.

The Light sensor bit

The Light sensor reads light levels between 0 to 100.



The Light sensor bit has an analogue output it can only be used at ports 0, 1, 2 and 3, which are the Invention Engine's analogue inputs.

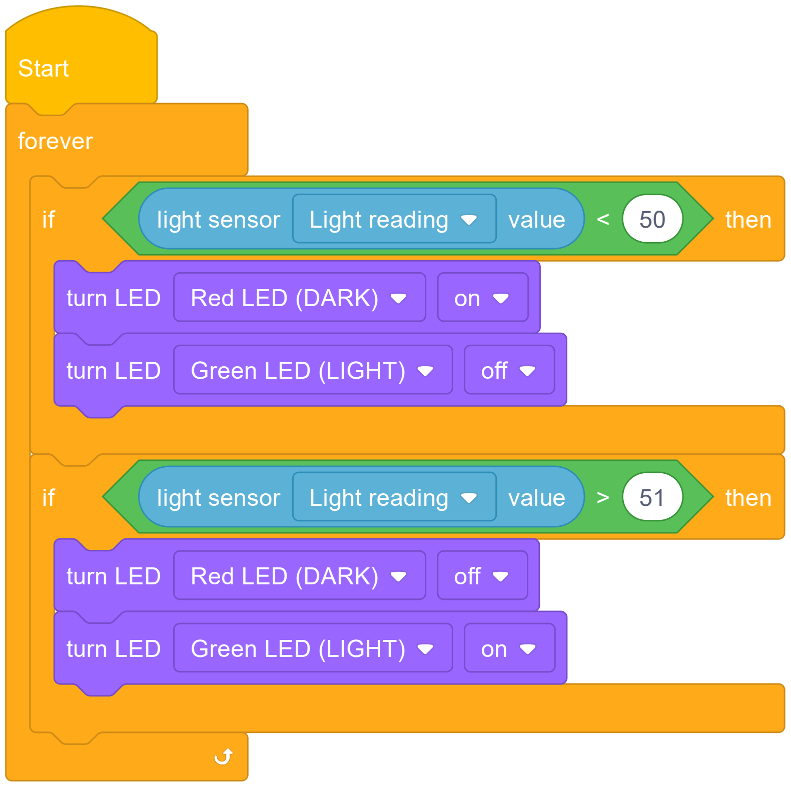
The Light sensor is used in inventions such as an alarm that detects that the fridge door has been opened, or an invention that reacts when the sun has gone down.

The Light sensor bit almost looks identical to the LED bits, however, the sensor called a 'phototransistor' has a water clear lens.

The Light sensor bit works by comparing the light levels. For example, if it gets darker than *this* number, then do *this* action. This comparison is possible by using Boolean expressions.

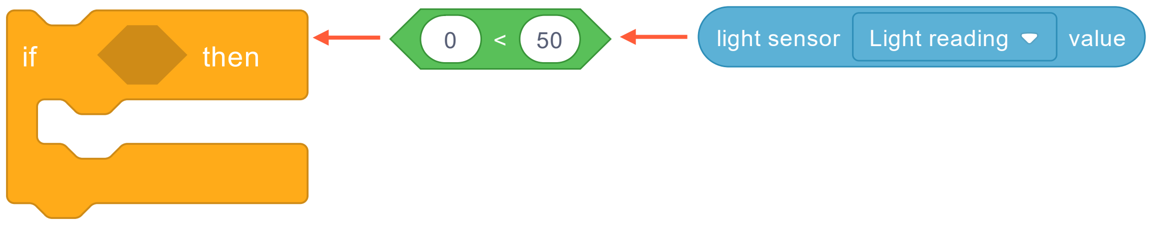
Trace the code

What do you think the following program does? Try it! Play with the code.



```
Start
forever
    if light sensor Light reading value < 50 then
        turn LED Red LED (DARK) on
        turn LED Green LED (LIGHT) off
    if light sensor Light reading value > 51 then
        turn LED Red LED (DARK) off
        turn LED Green LED (LIGHT) on
```

Remember to look at the block's shape.
The shape gives you a hint on where the block needs to go.



Let's Invent Activity 4: A nocturnal or diurnal creature

Living organisms thrive at different light levels. Some are nocturnal (active during night-time) and some diurnal (active during daytime), and they have evolved features to thrive in either low or high light environments.

For example, some animals are photophobic. Do you know the movie Gremlins? The creatures in it are photophobic, or light fearing. On the other hand, most plants are photophilic, or light loving.

Can you think of one animal that is either? Think of their behaviour and their reaction to light. How have they adapted to their environment?

Task

Using Invention Engine, let's invent an electronic creature that reacts to light. It can be a companion animal or pet, or an animal that you like.

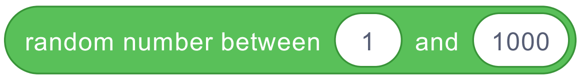
Work in your journal through the steps of the Invention Engine Invention Cycle to make your invention.

Activity 5: Random numbers and the Interactive book

Random numbers

Random numbers are chosen by chance, they are randomly selected from a set of numbers. All the numbers in the set of numbers have the same probability of being chosen.

Find the random block in the Operators category in Invention Engine Blocks, and you can use it to ask Invention Engine to choose a random number.



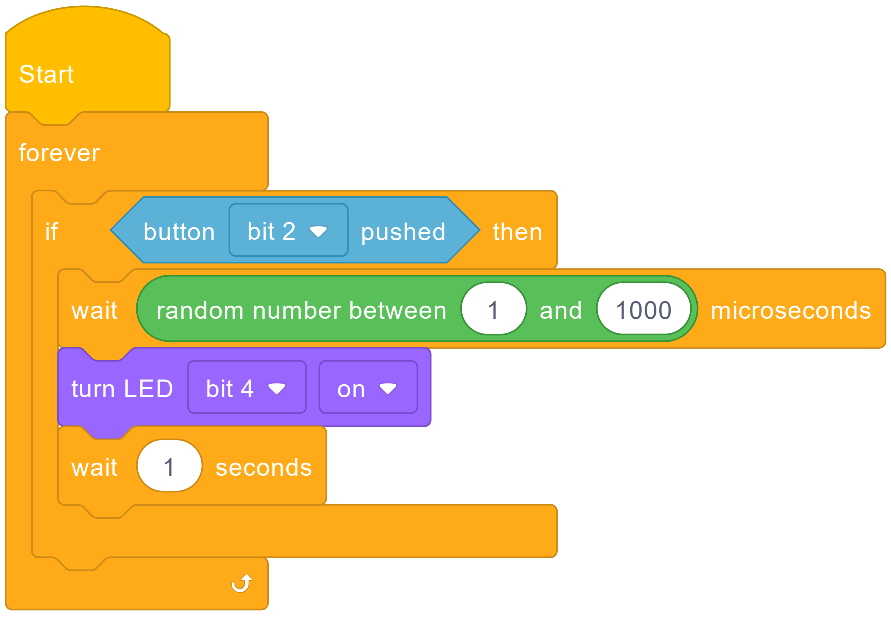
```
random number between (1) and (1000)
```

You can change the parameters of the random number by editing the lower and upper parameter in the block.

Random numbers are used to select lottery winners, sometimes in games or simulations, in random sampling, in medical trials, etc.

Random numbers can also be fun. For example, a jack-in-the-box is fun because we can't predict when the action will happen.

The code below uses random numbers in the number of milliseconds it will wait after the button push to turn on the light. So, we do not know how long it will be after we push the button for the light to come on.



```
Start
forever
    if <button [bit 2] pushed> then
        wait (random number between (1) and (1000)) microseconds
        turn LED [bit 4] [on]
        wait (1) seconds
```

Let's invent Activity 5: Interactive book

This activity will support you in practicing all the skills you have developed in Unit 2 of Invention Engine!

Interactive books require the reader to do something, thus providing the reader with the opportunity to be part of the story. Interactive books contain sounds, actions, special effects and ultimately make the experience of reading a book an exciting one.

Task

In this task you will create an interactive book. The story will be supported by the Invention Engine bits, which will make up the interactive part of the book.

For this task, push yourself! See how many bits you can add to your design. And enjoy the journey of creation.

Work in your journal through the steps of the Invention Engine Invention Cycle to make your invention.