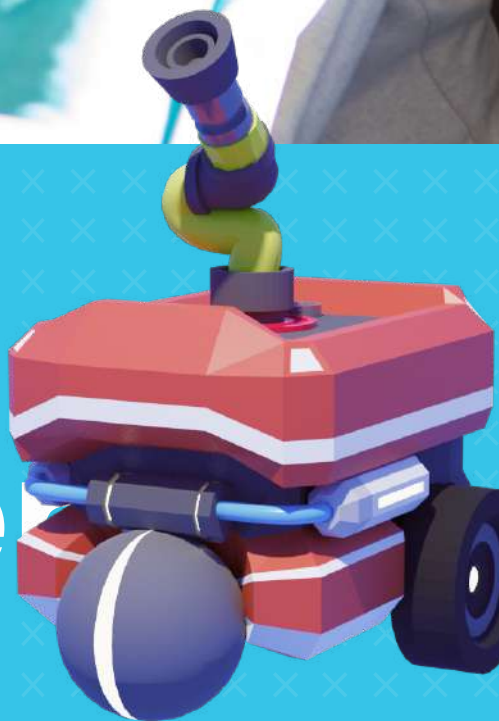
CreateBase

Educator Guide

# Heat Seeker



# Project Overview

## Background

An overloaded electrical circuit has resulted in a fire inside a warehouse! The warehouse stores badly protected explosive hydrogen fuel cells. If we don't put out the fire quickly, the resulting heat from the fires will cause the hydrogen fuel cells to explode! The overloaded circuit and resulting fire have fried most of the factories wiring, so the power is out. Without lights, we have limited visibility. Sending human firefighters into the warehouse would be extremely dangerous with the risk that an explosion could occur at any time. Luckily, this warehouse utilises line-following robots to move its items around.

## Synopsis

In this Project, learners will create an algorithm to guide a line-following robot to a series of fires within a warehouse, putting them out safely before they spread to the nearby hydrogen fuel cells! Learners will not only create their own control algorithm, but they will also learn about some of the basics of fire safety, warehouse automation, and the advantage that robots have over humans when operating in hazardous situations.

## This Project is divided into nine lesson plans

**01.** **Define** the situation by playing and discussing a video with your class. You will then help them to accurately **Define** the problem that they will be solving in the Project.

**02.** Learners will **Imagine** their own solutions to the problem before being introduced to the solution that they will be implementing. You will also describe the approach to implementing this solution that students will be taking in this Project (solving five seperate sub-problems).

**03.** Learners will now use the platform to start to **Create** their solutions. They will begin by researching and then planning their solution to the first sub-problem.

**04.** Learners will continue to **Create**, coding their solution to the first sub-problem before researching the second.

**05.** Learners will continue to **Create**, planning and then coding their solution to the second sub-problem.

**06.** Learners will continue to **Create**, planning and then coding their solution to the third sub-problem.

**07.** Learners will continue to **Create**, researching, planning and then coding their solution to the fourth sub-problem.

**08.** Learners will **Create** their solution to the complete problem. They will plan how they will combine their solutions to each of the sub-problems together before writing and testing their code.

**09.** Learners will be given an opportunity to **Improve** their solution, before concluding the Project by spending time to **Review** their journey.

# Project Overview

## Subject Areas

- Technology
- Engineering
- Computer Science
- Line Following
- Fire Safety

## Learning Outcomes

By the end of this Project, **learners** will be able to:

- Explain how and why line-following is used for navigation.
- Understand why we decompose problems.
- Understand the fundamentals of flow-based coding and write code using the Flow editor.
- Describe DC motors and outline how they work (optional).
- Describe IR sensors and outline how they work.
- Define comparators and conditionals.
- Write programs that use sensor data to control an output.
- Use logical reasoning to derive simple and more complex algorithms.
- Write programs that use conditional statements to control an output.
- Differentiate between different types of fires and how to deal with them.
- Define while loops and use them to solve problems.
- Combine the solutions to different sub-problems to solve a larger problem.
- Explore ways a solution can be optimised.

## Equipment List

**Learners** require:

- Access to our online platform through a Chromebook, laptop or PC
- Access to our learning journal through either Google docs or Microsoft Word

**Educators** require:

- The situation video (link included in lesson plan)
- Access to this lesson plan, either printed or digitally
- Easy access to the answer documents, printed or digitally (file included in lesson plan)

# Table of Contents

# Table of Contents

# Project Overview

## Adapting the lessons

The times given for each step of the Creation Process are only there to give you an estimate for how long they could take. You have the final say on how long to spend on each activity, depending on how much time you have available and how in-depth you want to discuss each topic. We recommend using the times as a minimum estimate to help you with your own planning. You are free and encouraged to adjust, skip and/or repeat steps depending on the engagement and aptitude of your class and your desired goals for the lesson.

The content included in each lesson plan serves as an example of how you could deliver the Project contents to your learners. It is your choice whether you want to follow it to the letter, use it as a rough guide or do your own thing. You may also want to allow excelling learners to move fowards at a faster pace and start working on the (nearly endless) Improve step before the other learners.

## Online learning

These lesson plans can be used for online learning.

If your school runs lessons through video meeting platforms, we recommend using breakout rooms for class discussions. All other tasks can be completed in the lesson as per the lesson plan or as homework activities.

If an individual learner needs additional help or guidance, you can ask them to temporarily share their screen (potentially in a breakout room) so that you can see what they are doing and guide them more closely.

Suppose under special circumstances, such as during a lockdown, it becomes difficult for students to have regular attendance. In that case, you can skip over the class discussion parts of the lesson plans and assign learners individual tasks (e.g. complete the first sub-problem in the Create step individually by DD/MM/YYYY).

We recommend that you use the learning journal with checkpoints to set completion targets for your class. Have them share their progress with you by making their learning journals available through email or Google Classroom.

# Overview 1

## Define (45 mins)

Introduce and discuss the Project with your learners with a video that showcases the situation. The goal is to have learners identify what has happened then work as a class to define the problem that they will be solving in this Project through the development of a Project brief.

## Learning Outcomes

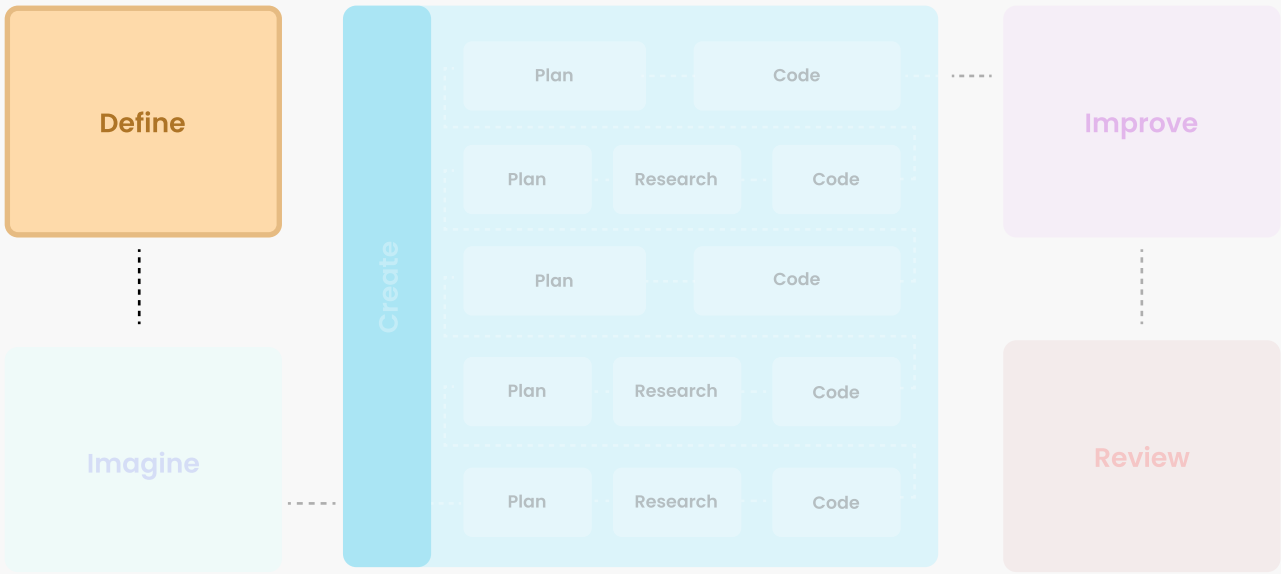
By the end of this lesson, learners will:

- Understand the situation.
- Understand the purpose of using line-following in warehouses.
- Be able to define the problem that they need to solve.

## Lesson Resources

Situation cutscene

## Project Progress



## Glossary

1. **Automation:**
   Automation is all about using technology to carry out tasks that a human would have traditionally carried out. Examples include robots in car manufacturing assembly lines and self-driving trains. Humans are good at critical thinking, whereas most technology is great at repeatedly performing the exact same action. This is why routine tasks (tasks that involve performing the same action repeatedly) are usually the first to be automated as little or no critical thinking is needed.

   **E.g. robots in car manufacturing assembly lines, self-driving trains and self check-in machines at airports.**

2. **Autonomous:**
   A device or vehicle that is autonomous is able to operate without any direct human control. A device or vehicle is normally made autonomous by writing code that is able to take in information, use that information to make a decision, and then automatically perform a designated action for the outcome of that decision.

   **E.g. traffic light that only turns green when there are cars waiting in that lane. A "dumb" autonomous vehicle or device would skip the information processing and decision-making steps and simply perform a repetitive task. For example, a traffic light that changes colours on a timer.**

3. **Line-following:**
   Line following refers to the process of directing a robot by making it follow a line (usually a black line on the floor). Line following requires the use of a series of sensors (normally infrared) to detect when the robot has strayed off the line and an algorithm that uses this sensor data to determine how to control the robot to keep it on the line. A lot of autonomous robots that always follow the same set of paths use line following to navigate as it is relatively cheap and easy to set up. The downside of using line following for navigation is that you are restricted to the predetermined paths set out with the physical lines.

# Define Part I 10 mins



## Use these steps for guidance

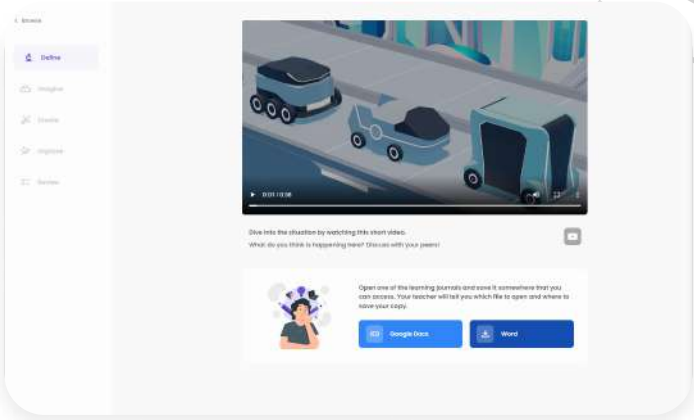
1. Bring all of your learners to the front of the class if possible.

2. Welcome the class, for example, by telling them that they will be completing a **Project** over the following nine lessons where they will get to plan, build, test, and improve a solution to a problem.

3. Tell the class that we will begin by playing a video of the situation to set the scene. You can then play the 68-second video on a projector for the class to watch. Access the video through the link on the right or the lesson resource.

4. Discuss with the class what happened in the video.

   Discussion points could be based on the setting, the first problem that occurs, and then how this results in a chain of events leading up to the final (dangerous) situation.



https://app.createbase.co.nz/project/heat-seeker/define

If they understand the problem, you might want to ask learners what they would do in the same situation.

### Tips for Educators

For your reference, here is a quick explanation of what is happening in the video:

- An electrical fire starts in a warehouse. This warehouse uses line following robots to move items around.
- Firefighters arrive at the warehouse to put out the fire.
- A dangerous explosion occurs as the fire spreads to nearby hydrogen fuel cells, almost injuring the firefighters.

# Helpful Examples



If you are finding it difficult to get your class to talk about it, replay the video to the class with pauses and ask questions about what is happening in specific scenes.

**Here are some example discussion points to get you started:**

### 0:06-0:10 seconds

**Q** "Where are we?"

**A** Inside a building/warehouse.

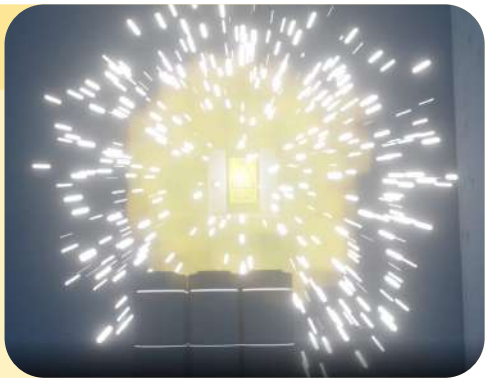
**Q** "What can we see inside the warehouse? What stands out as being unique?"

**A** There are white lines on the floor and a pair of robots that appear to be following the lines, each carrying an item.

### 0:11-0:27 seconds

**Q** "What has happened here?"

**A** A lightning strike has overloaded an electric circuit, resulting in a fire breaking out.



### 0:30-0:36 seconds

**Q** "What will happen next if we don't put out the fire?"

**A** The fire might spread to the other items in the warehouse (hydrogen fuel cells)."

# Helpful Examples



**0:37-0:50 seconds**

**Q** "What happened?"

**A** Firefighters arrived to put out the fire, but an explosion occurred when they tried to enter.

**Q** "Why was there an explosion? "

**A** The heat from the fire caused the hydrogen fuel cells to explode.

**Q** "Why is this a problem?"

**A** If the firefighters had entered the building slightly earlier, they would have been caught in the explosion!

**Summary**

**Q** "What has happened?"

**A** A lightning strike resulted in a fire starting inside a warehouse that utilises line-following robots. When firefighters arrived to try and put out the fire, explosive hydrogen fuel cells stored within the warehouse ignited, almost injuring the firefighters.

# Define Part II 10 mins



Now that you have watched the video with your class and you are happy that they understand what has happened, have an **in-depth discussion** about the exact problem that has occurred. The direction of this discussion should be dynamic and driven by learners' curiosity. Make sure to ask follow up questions, and if the learners lead you away from the prompts below, then don't be afraid to go off-script!

**Here are some examples of questions that you could ask to start discussions:**

1. We know from the situation that a fire is spreading throughout the warehouse!
    a. **What do you think could have been done to avoid the fire?** - don't place any flammable/explosive items near open flames or sources of electricity.
    b. **What do most buildings have to detect a fire?** - smoke alarms.
    c. **What do most buildings have to automatically put out a fire?** - sprinklers.

2. We saw that there was an explosion when the firefighters tried to enter the warehouse.
    a. **Why did this occur?** - the heat from the fires increased the temperature and pressure of the hydrogen fuel cells until they exploded.
    b. **How could this situation have been avoided?** - safety/warning signs outside the warehouse that indicated the presence of dangerous/explosive items inside the building to stop the firefighters from entering. Use an automatic or human-free system for putting out the fire.

2. We don't want to purposefully send human firefighters into highly dangerous situations (at least, more dangerous than normal fires).
    **a. Should we just leave the fire alone? Why or why not?**
        i. **Why:** won't risk any more human lives.
        ii. **Why not:** the fire could spread to nearby buildings and if we manage to put out the fire early, we may save some of the equipment inside the warehouse.
    **b. If we can't send a human into the warehouse to put out the fire, what could we send instead?** - a robot.

# Define Part II 10 mins



3. In order for a robot to replace one of our firefighters, what is it going to need to be able to do?



1. **Detect each fire**



2. **Move to the location of each fire inside the warehouse**



3. **Put each fire out**

4. Bring the discussion to a close by linking it back to the situation video and clearly stating the problem and the brief:

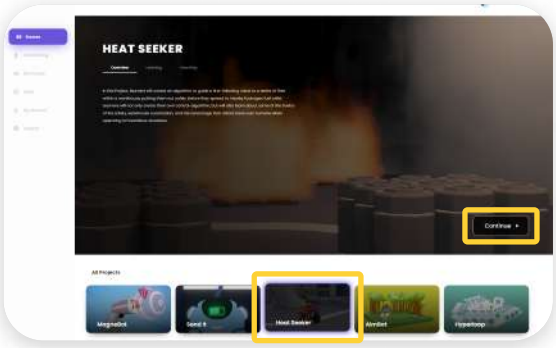
> **"In this Project, we have a fire inside a warehouse that we need to put out. Because the warehouse contains items that are at risk of exploding due to the fire, is too dangerous to send human firefighters inside. We are going to have to come up with another method of finding and putting out the fires that does not require human presence."**

# Define Part III 15 mins

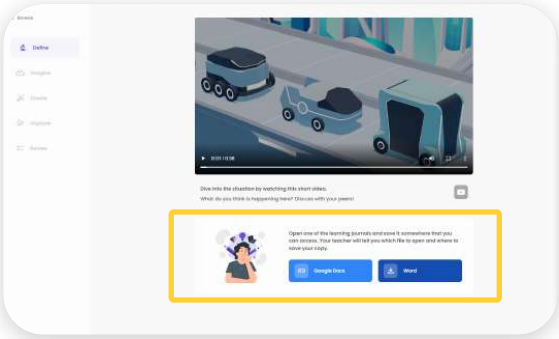


If your learners aren't already on the platform, then now is the time to get them to log in to their accounts and access the Heat Seeker Project.

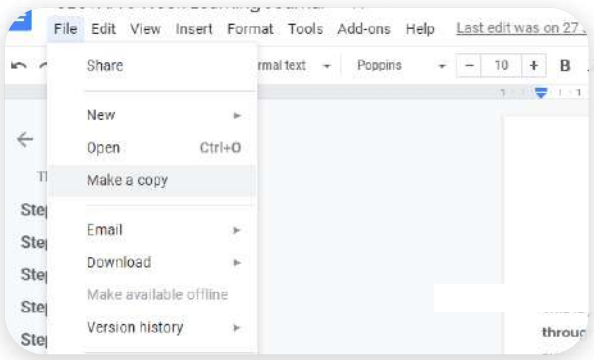
1. Tell your learners to visit [https://app.createbase.co.nz/browse/heat-seeker](https://app.createbase.co.nz/browse/heat-seeker) and click continue.




2. Ask them to rewatch the situation video if they need a refresher and tell them to download either the Google Docs **or** Microsoft Word **learning journal** using either of the buttons underneath the video.



If you want your learners to work from Google Docs, they must choose the Google Docs option and then click ***File*** -> ***Make a copy*** and save it somewhere that they can access.




3. Once downloaded, the learners should spend the remainder of the lesson answering all of the questions inside **Step 1: Define**.

3. If any students finish early you can ask them to move on to the Define step on the platform (the entire class will be going through this during the next lesson).

### Tips for Educators

To reduce confusion, it is recommended that you tell your learners exactly where to save their files. If you want to access these files later to check on their progress, ask them to save them in a location that you can also access.

7. At the end of the lesson, make sure that each learner has their copy of the **learning journal** saved somewhere that they can easily access at the start of the next class.

# Overview 2

## Imagine (45 mins)

Learners get the chance to brainstorm their own solutions before being introduced to the solution they will be building and the approach they will be taking to implement it.

## Learning Outcomes

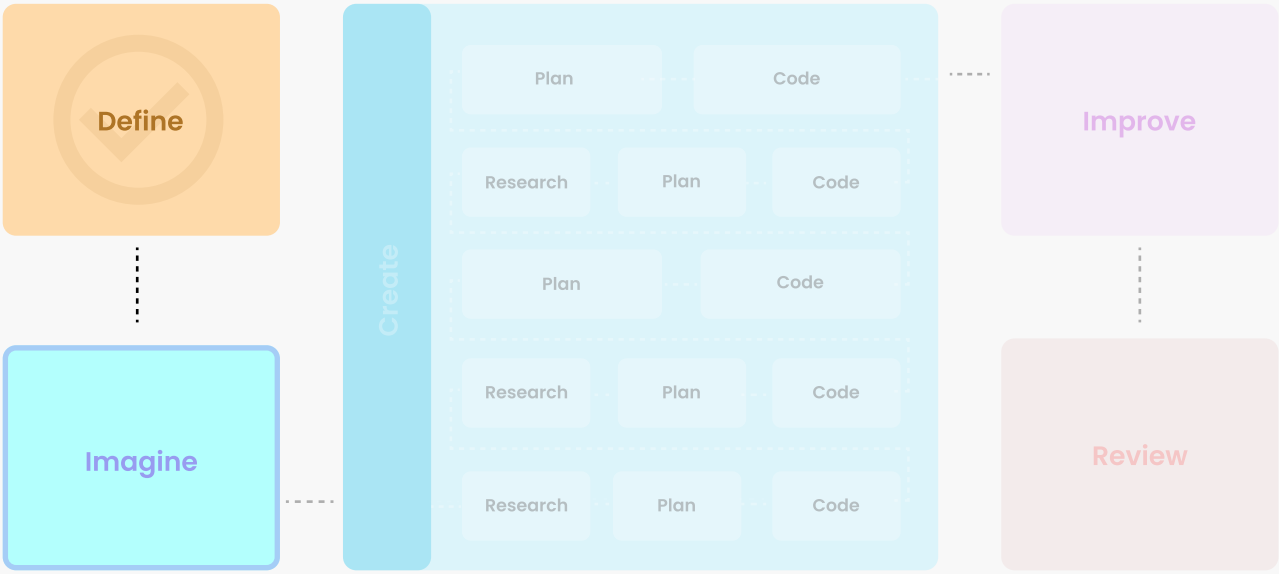
By the end of this lesson, learners will:

- Describe their own solution to the problem.
- Understand the approach that they will be undertaking to solve the problem.
- Understand why we decompose problems.

## Lesson Resources

- Line Following
- DC Motors

## Project Progress



## Glossary

1. **Line-following:**
   Line following refers to the process of directing a robot by making it follow a line (usually a black line on the floor). Line following requires the use of a series of sensors (normally infrared) to detect when the robot has strayed off the line and an algorithm that uses this sensor data to determine how to control the robot to keep it on the line. A lot of autonomous robots that always follow the same set of paths use line following to navigate as it is relatively cheap and easy to set up. The downside of using line following for navigation is that you are restricted to the predetermined paths set out with the physical lines.

2. **Flow control/branching:**
   In programming, the flow of code refers to the order in which code is run, and actions are performed. Think about it like walking down a tile path, where each tile represents a line of code. When you stand on a tile, that line of code is run. By looking ahead at the upcoming tiles, you can see the order that each line of code will be run. If your path splits into two, you can no longer tell exactly which code will be run in advance until you decide which path to walk down. Flow control is all about how we write our code to make these decisions. Branching is the simplest form of flow control, where the code "path" splits into multiple paths.

3. **Algorithm:**
   An algorithm is a sequence of rules or instructions. Computers usually follow algorithms to perform a calculation or make a decision. Good algorithms are well-defined, meaning that they have a rule or an instruction for every relevant possibility.

4. **Loops:**
   In programming, a loop refers to a piece of code that is run multiple times. The number of times that the code repeats itself depends on how the loop is written.

# Imagine Part I 20 mins



If some time has passed since the previous lesson, then go through the following questions with your class:

1. **What is the problem?** - There is a fire in a warehouse with explosive items.
2. **What needs to be done?** - We need to find and put out the fire without using any humans.

Now that you have defined the problem with your class and you are happy they understand the problem, we could move on to the Imagine step. The Imagine step is divided into 3 stages as below:

### 1. Imagine a solution:

Within this stage, your students should work in groups or individually to come up with their own potential solutions. They should be using the problems that they outlined in the Define step to determine if their solution would be fit-for-purpose. Their learning journals contain questions that help them to start thinking about potential solutions. You may ask your learners to simply complete the Imagine questions in their learning journals or you might want to carry out your own activity. Once the students have a solution in mind, try and get them to present their solution to the class explaining how it solves the problem. Ask your learners follow-up questions according to their answers to test their understanding of the problem.

### 2. Explore our solution:

In this stage, you have the option to either explain to the student or let the students read the information within our platform regarding the solution that we are going to go through. The students should also try out the manual game and try to solve the problem using keyboard controls. At the end, try and have a discussion about what they observed within the environment and walk through their thinking process as they were playing the game: what decisions were they making, and what information were they using to make those decisions?

## Main Highlights

**The solution:** A fire extinguisher is attached to a robot. The robot is programmed to follow a line, navigating between and putting out each fire.

- **Observations:** There are a few observations to look at:
  - It is challenging to navigate through the smoke with limited visibility. This may make a solution that involves navigating using cameras very difficult.
  - There is a possibility of human error occurring during manual control.

- **Process of thinking:** The learner should be able to identify how they reacted differently to each section of the track. Example sections include:
  - Straight lines
  - Slight curves
  - Sharp turns
  - FIre being close

### 3. Divide the problem:

This stage aims to introduce your learners to the approach that they will be taking to solve the problem.

We are breaking our main problem down into a series of sub-problems that we can build individual solutions for, and then combine them at the end to solve the complete problem. This makes our problem more approachable by breaking it down into bite-sized chunks that we can focus on one at a time.

Students can read through this process using the third page of the Imagine pdf on the platform (https://app.createbase.co.nz/project/heat-seeker/imagine) and/or you could explain it to them briefly.

## Main Highlights

**The reason behind dividing of the problem**

- Start implementing and testing at an earlier stage
- Easier to identify and fix any issues that occur along the way

- **The different subproblems**
  a. We need to program our robot to move.
  b. We need to program our robot to detect small curves in the line and adjust its movement accordingly.
  c. We need to program our robot to detect significant turns in the line and adjust its movement accordingly.
  d. We need to program our robot to detect the presence of a fire and automatically put it out.
  e. We will then combine each of these smaller programs to solve the complete problem.

After explaining the approach, you can conclude the lesson by letting your learners know that they will be able to start creating their solutions to the first sub-problem during the next lesson.

# Overview 3

## Research I (30 mins)

Learners will learn about line following algorithms as well as get a chance to research the tools available on our platform to construct their solutions.

## Plan I (15 mins)

Learners will also plan their approach for solving the first sub-problem.

## Learning Outcomes

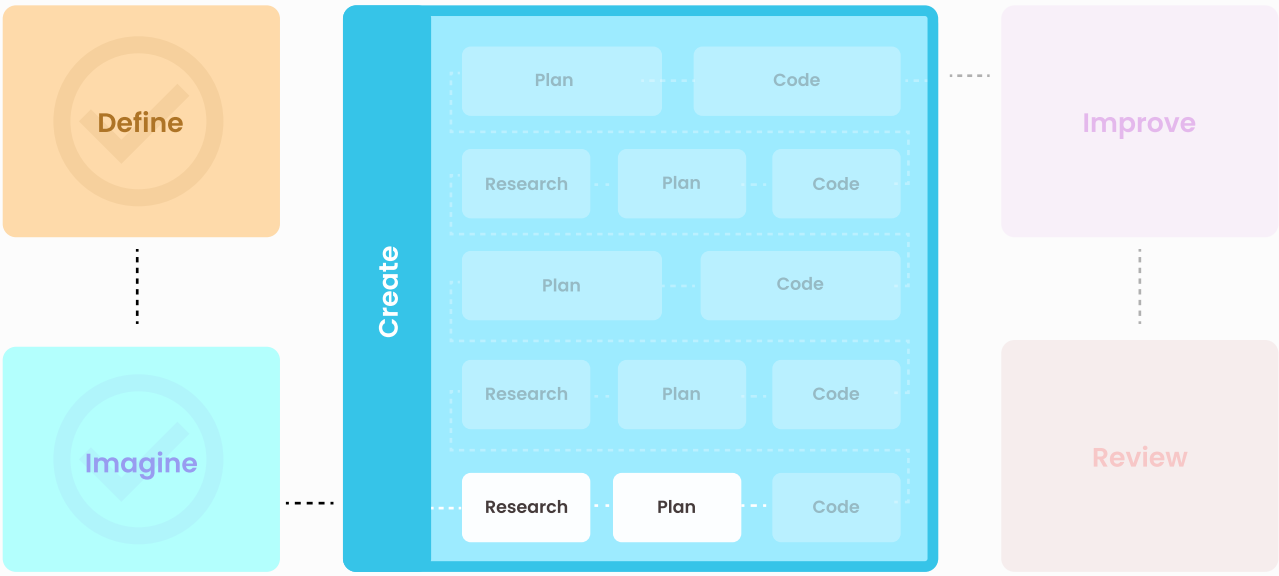
By the end of this lesson, learners will:

- Understand the fundamentals of flow based coding.
- Describe DC motors and outline how they work (optional).

## Lesson Resources

Subproblem 1 Plan Solutions

## Project Progress



## Glossary

1. **Comparison:**
   In programming, a comparison is a statement where we compare two different values with an operator. The operator determines what characteristic we are comparing and is usually a mathematical symbol like equals, less than or greater than. The statement that we write will always be either TRUE or FALSE. For example, we might write a statement that says, "20 is less than 10." This statement would be FALSE. We might instead write "20 minus 10 equals 10." This statement would be TRUE. Comparisons are important in coding because we can use the TRUE/FALSE result to decide which branch of code we should run next.

2. **Flow control/branching:**
   In programming, the flow of code refers to the order in which code is run, and actions are performed. Think about it like walking down a tile path, where each tile represents a line of code. When you stand on a tile, that line of code is run. By looking ahead at the upcoming tiles, you can see the order that each line of code will be run. If your path splits into two, you can no longer tell exactly which code will be run in advance until you decide which path to walk down. Flow control is all about how we write our code to make these decisions. Branching is the simplest form of flow control, where the code "path" splits into multiple paths.

3. **Algorithm:**
   An algorithm is a sequence of rules or instructions. Computers usually follow algorithms to perform a calculation or make a decision. Good algorithms are well-defined, meaning that they have a rule or an instruction for every relevant possibility.

4. **Loops:**
   In programming, a loop refers to a piece of code that is run multiple times. The number of times that the code repeats itself depends on how the loop is written.

# Create 5 mins



1. Welcome the class back to the Heat Seeker Project. If there has been a significant delay between the current and previous lesson, then you might want to give a quick recap to your learners of what you covered in your class in the last lesson.

2. Tell the class that today they will start creating their solutions to the first sub-problem. For each sub-problem, your learners will need to **research** the required tools and knowledge, **plan** how they will utilise these tools and this knowledge, and then write some **code** to solve the sub-problem in a 3D simulation. For your reference, here is a more in-depth breakdown of how the Create step works:

**Research**
Within this step, your learners will mainly go through two different contents. The first part will research the real life implementations of the subproblem and the thought process behind it. The next part will explain any new Flow blocks that they will need to utilise in order to solve the subproblem.

**Plan**
Within the plan, they will go through different questions in their learning journal with the purpose of understanding how they could formulate an initial algorithm that they could then implement with the code step.

**Code**
Within the code step, your learners will be able to convert their initial algorithm into code blocks and test it. They will make small changes within the algorithm to make sure that the sub-problem is solved.

This process is repeated once for each sub-problem

Lesson 3 •

# Research I 30 mins

3

### Tips for Educators

In this Project, research is divided into two parts:

- Line following research
- Coding research

For the line following research, they should read the line following document on the platform, conduct their own research, and then discuss it in groups to try and answer the questions within their learning journals.

In this Project, your learners have the option to code their answers in either a visual programming language called Flow, or a text programming language called JavaScript (JS). The platform contains Research content for both languages, so if you want your learners to code in only one language, you should let them know so that they can skip the Research content for the other language.

1. Tell your learners to return to the platform and start the first sub-problem (https://app.createbase.co.nz/project/heat-seeker/create/Speed%20control/research).

2. For the line-following research, they should read the Line Following document on the platform, conduct their own research and discuss it in groups to try and answer the corresponding questions in their **learning journal**.

   Every learner in your class should be able to complete their research. To help you assist your learners, you can find example answers on the following page.

# Research I 30 mins



1. **Describe what a line following robot is in one or two sentences.**
   a. https://www.elprocus.com/line-follower-robot-basics-controlling/
   b. A line follower robot is a robot that follows a fixed path, usually a physical line.

2. **List two examples where a robot may use line-following for navigation?**
   a. https://www.elprocus.com/line-follower-robot-basics-controlling/
   b. **Industrial Applications:** These robots can be used as automated equipment carriers in industries replacing traditional conveyor belts.
   c. **Automobile applications:** These robots can also be used as automatic cars running on roads with embedded magnets.
   d. **Domestic applications:** These can also be used at homes for domestic purposes like floor cleaning etc.
   e. **Guidance applications:** These can be used in public places like shopping malls, museums etc., to provide path guidance.

3. **How does a line-following robot work?**
   a. https://www.elprocus.com/line-follower-robot-basics-controlling/
   b. The line-following robot is an example of a self-operating machine that detects and follows a marked line. The line is normally indicated by a white line on a black surface or a black line on a white surface to produce a high level of contrast. Here we are using three sensors for path detection purposes, all IR sensors. One of the sensors is mounted at the front end of the robot, while the other two are at the back, just in front of the wheels. The IR sensor can be used to detect the line and make conclusions on the heading of the line. We could then make decisions based on how the line is going by adjusting the speed of the wheels to follow the same direction as the line.

**4. Discuss the importance of using lines for navigation for robots. Why do these robots use lines? What makes lines useful to them? Why would you use line following over other methods of navigation?**

a. https://www.elprocus.com/line-follower-robot-basics-controlling/
b. Lines are a simple way to physically lay out a path for robots to automatically follow. If you did not use lines for navigation, you would have to either use complex information from a potentially changing environment or have a 'dumb' pre-programmed environment that does not react to changes.
   i. They are quite simple to build and implement: containing simple sensors and basic information processing.
   ii. Fit and forget system: Once you implement it properly, not much further development is required other than basic maintenance.

**5. What do you think would happen if suddenly there was no line? This may be the case if the lines are poorly maintained or incorrectly created.**

a. It will stop following the line and will perform actions according to its programming. This could cause issues with safety if it is not handled appropriately.

**6. Can you find any real-life examples for the previous question?**

# Research I 30 mins



**(Optional)**: This Project requires learners to research and find information online to assist with problem solving. Questions are ordered in increasing levels of difficulty, with each one exploring more depth and detail about the topic.

1. **What is a DC Motor?** A DC motor is an electric motor that runs on direct current.
2. **Where are DC Motors used?** There are different types of DC Motors which each have very specific usages. However, in general, DC Motors are used anywhere where low powered rotational motion is needed.
3. **How does a DC Motor work?**
   https://www.elprocus.com/dc-motor-basics-types-application/
   The DC motor working principle is that when a current-carrying conductor is located within the magnetic field, then it experiences a mechanical force. This force direction can be decided through Flemming's left-hand rule as well as its magnitude.
   If the first finger is extended, the second finger, as well as the left hand's thumb, will be vertical to each other & the primary finger signifies the magnetic field's direction, the next finger signifies the current direction & the third finger-like thumb signifies the force direction which is experienced through the conductor.
   **F** = BIL Newtons
   Where,
   **'B'** is the magnetic flux density,
   **'I'** is current
   **'L'** is the conductor's length in the magnetic field.
   Whenever an armature winding is given toward a DC supply, then the flow of current will be set up within the winding. Field winding or permanent magnets will provide the magnetic field. So, armature conductors will experience a force because of the magnetic field based on the above-stated principle.

Next, learners will now use the platform to learn what tools they have available to solve the defined problem and how these tools work. This would mean going through the Flow tutorial video, the Introduction to flow blocks and the Action blocks. Once they go through these documents, they should have a go at trying to solve the questions within their learning journal.

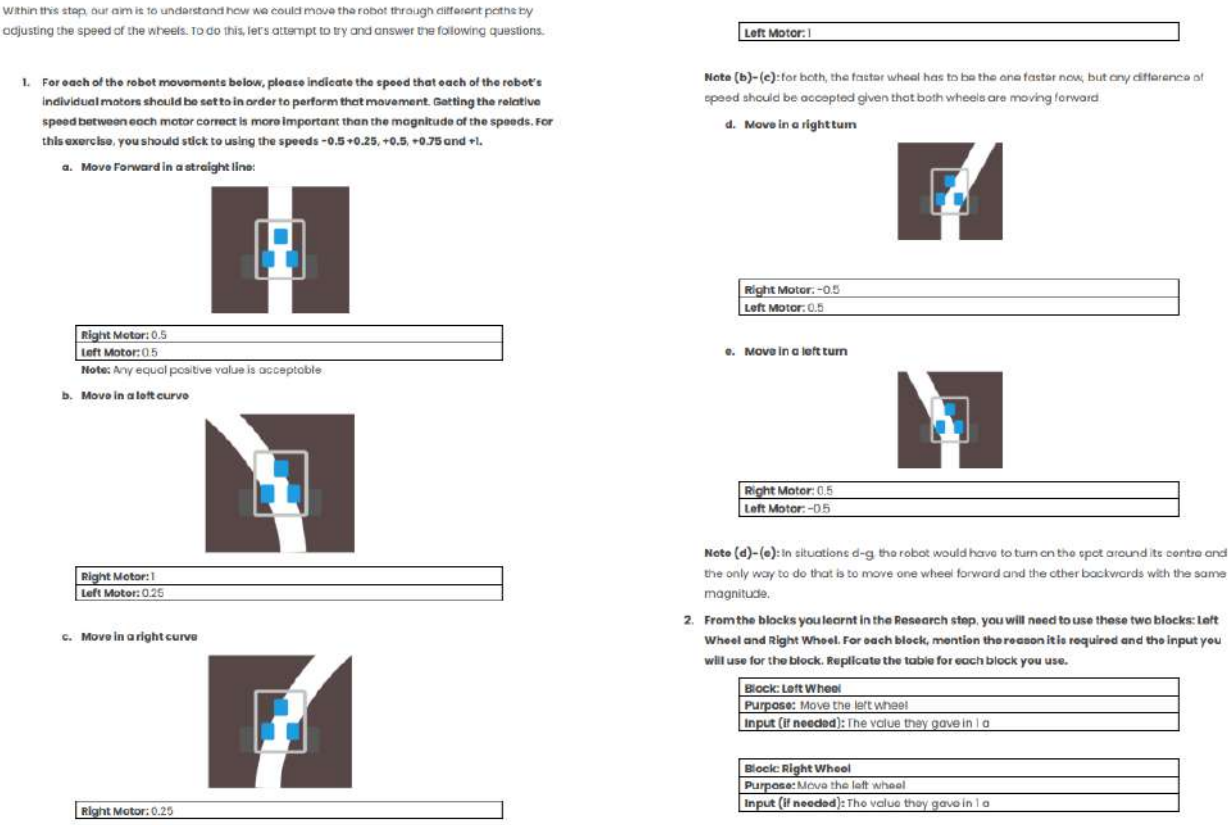
1. **What action blocks are available?** Left Wheel, and Right Wheel
2. **For each of these blocks, what action do they make our robot do? If the block has an input, how does this input change the resulting action?** Move the wheels forwards or backwards
3. **What actuators would be required to perform this action? (optional)** A DC motor to turn the wheels.

# Plan I 10 mins



Your learner's final task for the remainder of the lesson is to create a plan for how they will use Flow blocks to operate a line-following robot To do this, they should answer the **Subsystem I Plan** questions within their **learning journals**. Example answers for this plan step are available in this document for you to assist your learners if required.

It is not necessary for every learner to finish this lesson with a perfect plan. As long as they have something detailed out, they can improve it through trial and error once they start coding.



Within this step, our aim is to understand how we could move the robot through different paths by adjusting the speed of the wheels. To do this, let's attempt to try and answer the following questions.

1. **For each of the robot movements below, please indicate the speed that each of the robot's individual motors should be set to in order to perform that movement. Getting the relative speed between each motor correct is more important than the magnitude of the speeds. For this exercise, you should stick to using the speeds -0.5 +0.25, +0.5, +0.75 and +1.**

   a. **Move Forward in a straight line:**

   | **Right Motor:** 0.5 |
   |---|
   | **Left Motor:** 0.5 |

   **Note:** Any equal positive value is acceptable

   b. **Move in a left curve**

   | **Right Motor:** 1 |
   |---|
   | **Left Motor:** 0.25 |

   c. **Move in a right curve**

   | **Right Motor:** 0.25 |
   |---|
   | **Left Motor:** 1 |

   **Note (b)-(c):** for both, the faster wheel has to be the one faster now, but any difference of speed should be accepted given that both wheels are moving forward

   d. **Move in a right turn**

   | **Right Motor:** -0.5 |
   |---|
   | **Left Motor:** 0.5 |

   e. **Move in a left turn**

   | **Right Motor:** 0.5 |
   |---|
   | **Left Motor:** -0.5 |

   **Note (d)-(e):** In situations d-g, the robot would have to turn on the spot around its centre and the only way to do that is to move one wheel forward and the other backwards with the same magnitude.

2. **From the blocks you learnt in the Research step, you will need to use these two blocks: Left Wheel and Right Wheel. For each block, mention the reason it is required and the input you will use for the block. Replicate the table for each block you use.**

   | **Block: Left Wheel** |
   |---|
   | **Purpose:** Move the left wheel |
   | **Input (if needed):** The value they gave in 1 a |

   | **Block: Right Wheel** |
   |---|
   | **Purpose:** Move the left wheel |
   | **Input (if needed):** The value they gave in 1 a |

## Homework

Educators can optionally ask learners to complete their planning at home so that they can start coding right away at the start of the next lesson.

# Overview 4

## Code I (15 mins)

Your learners will use our platform to create and test their solutions using our 3D simulation tool.

## Research II (30 mins)

Learners will get a chance to research the tools available on our platform that they will need to use to construct their solution to the second sub-problem.

## Learning Outcomes

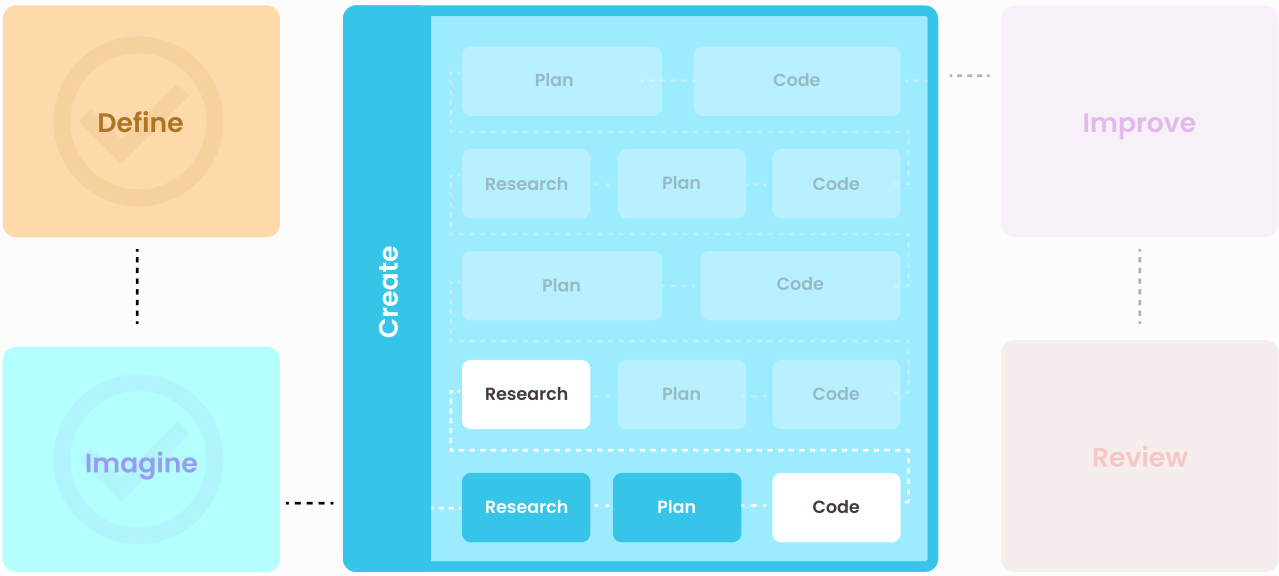
By the end of this lesson, learners will:

- Write simple code using Flow.
- Describe IR sensors and outline how they work.
- Define comparators and conditionals.

## Lesson Resources

- Subproblem 1 Plan Solutions
- Subproblem 1 Code Solutions
- IR Sensors

## Project Progress



## Glossary

1. **Comparison:**
   In programming, a comparison is a statement where we compare two different values with an operator. The operator determines what characteristic we are comparing and is usually a mathematical symbol like equals, less than or greater than. The statement that we write will always be either TRUE or FALSE. For example, we might write a statement that says, "20 is less than 10." This statement would be FALSE. We might instead write "20 minus 10 equals 10." This statement would be TRUE. Comparisons are important in coding because we can use the TRUE/FALSE result to decide which branch of code we should run next.

2. **Flow control/branching:**
   In programming, the flow of code refers to the order in which code is run, and actions are performed. Think about it like walking down a tile path, where each tile represents a line of code. When you stand on a tile, that line of code is run. By looking ahead at the upcoming tiles, you can see the order that each line of code will be run. If your path splits into two, you can no longer tell exactly which code will be run in advance until you decide which path to walk down. Flow control is all about how we write our code to make these decisions. Branching is the simplest form of flow control, where the code "path" splits into multiple paths.

3. **Algorithm:**
   An algorithm is a sequence of rules or instructions. Computers usually follow algorithms to perform a calculation or make a decision. Good algorithms are well-defined, meaning that they have a rule or an instruction for every relevant possibility.

4. **Loops:**
   In programming, a loop refers to a piece of code that is run multiple times. The number of times that the code repeats itself depends on how the loop is written.

# Code I 20 mins



1. Briefly explain to the class how the **Code** step will work:

**Build**
Learners will implement their solution by starting with individual elements and building up to the final solution. They should be referring to their plan as both a guide when they first get started and as a refernce point when they get stuck.

**Test**
Learners should constantly be testing their code to see how well their individual components or final solution solves the problem.

**Iterate**
Learners will continue building and testing until the problem has been completely solved.

**Share**
When learners think that they finished, encourage them to share their solution with other classmates who have finished and/or an educator to get their feedback. Make sure that the learners explain how they arrived at their solution. If necessary, they might need to start iterating again, either because a problem was identified or an educator/classmate gave them an idea for how to improve their solution.

2. Tell your learners to return to the platform and start the **Code** step for the first subsystem (https://app.createbase.co.nz/project/heat-seeker/create/Speed%20control/code).

### Tips for Educators

In this Project, your learners have the option to code their answers in a visual programming language called Flow, or a text programming language called JavaScript (JS). Learners can switch between the two languages by clicking on the FLOW and TEXT tabs in the bottom right corner of their screen in the simulation.

When you write code using Flow in the FLOW tab, the text equivalent is automatically generated in the TEXT tab when you click the TO TEXT button (next to the COMPILE button)! This can be a good tool for helping your learners transition from coding with a visual language to a text language.

### Tips for Educators

Your learners can save their Flow code using the save icon in the bottom left corner of the Flow editor or the SAVE button in the bottom right corner of the text editor.

If learners save their code before they leave the simulation, they will be able to continue where they left off when they return by pressing the restore icon or RESTORE button for the Flow and text editors respectively.

3. Attached as a lesson resource is a help sheet with explanations that is for the educator's viewing only. You can use the help sheet to assist struggling learners and give additional prompts to excelling learners to make them think more about their solution.

**Solution**

1. Connect both the left and the right wheel blocks to the start block. The order of the blocks does not matter.


2. For the input of each block, students should use the numbers that they had in the plan step. The values do not matter, except that they should both be positive and equal to each other. If one value is higher than the other, the robot will turn towards the side with the lower value.


3. Press Compile. The solution should work.

4. If any learners finish early and you do not see any more ways for them to work on their solution in the Code step, then you could either ask them to help their other classmates or let them move onto the next sub-problem early.

5. Next, learners will now use the platform to learn what tools they have available to solve the defined problem and how these tools work. Learners should read the cards inside Research for the second sub-problem and answer any questions in the corresponding section in their learning journals. Here are some example answers to help you guide your students:

**What sensing blocks are available?**

Left, right and middle line Sensors

**What do they measure?**

These sensors detect how much over the line these sensors are. If they are right in the middle the value is 1, 0 if it's not on the line at all, and some value in between if it's partially on the line.

**What sensors would be required to get this information? (Optional)**

IR sensors to detect the colour of the line and how much of the line is being detected. More information below.

**What is an IR sensor?**

https://www.elprocus.com/infrared-ir-sensor-circuit-and-working/

Gas warning devices, gas analysers, medical gas measurement technology, flame detectors, contactless precision temperature measurement, and colour measurement.

**Where are IR sensors used?**

https://www.elprocus.com/infrared-ir-sensor-circuit-and-working/

Gas warning devices, gas analysers, medical gas measurement technology, flame detectors and in contactless precision temperature measurement, colour measurement.

# Research II 25 mins



### How does an IR sensor work?

https://www.elprocus.com/infrared-ir-sensor-circuit-and-working/

IR LED is one kind of transmitter that emits IR radiation. This LED looks similar to a standard LED and the radiation which is generated by this is not visible to the human eye. Infrared receivers mainly detect the radiation using an infrared transmitter. These infrared receivers are available in photodiodes form. IR Photodiodes are dissimilar as compared with usual photodiodes because they detect simply IR radiation. Different kinds of infrared receivers mainly exist depending on the voltage, wavelength, package, etc.

Once it is used as the combination of an IR transmitter & receiver, then the receiver's wavelength must equal the transmitter. Here, the transmitter is IR LED whereas the receiver is IR photodiode. The infrared photodiode is responsive to the infrared light that is generated through an infrared LED. The resistance of the photodiode & the change in output voltage is in proportion to the infrared light obtained. This is the IR sensor's fundamental working principle.

Once the infrared transmitter generates emission, then it arrives at the object & some of the emission will reflect back toward the infrared receiver. The sensor output can be decided by the IR receiver depending on the intensity of the response.

**Tips for Educators**

For any learners that finish early, you could either come up with additional questions for them to research on the interenet, find answers to and document in their learning journals or simply allow them to move on to Plan early.

**Homework**

Educators can optionally ask learners to complete their research at home. For assessment, educators might only let learners start the Create step when they have shown evidence of some kind of research.

# Overview 5

## Plan II (20 mins)

Learners will get to detail their approach to solving the second sub-problem.

## Code II (25 mins)

Learners will use our platform to create and test their solutions in our simulation.

## Learning Outcomes

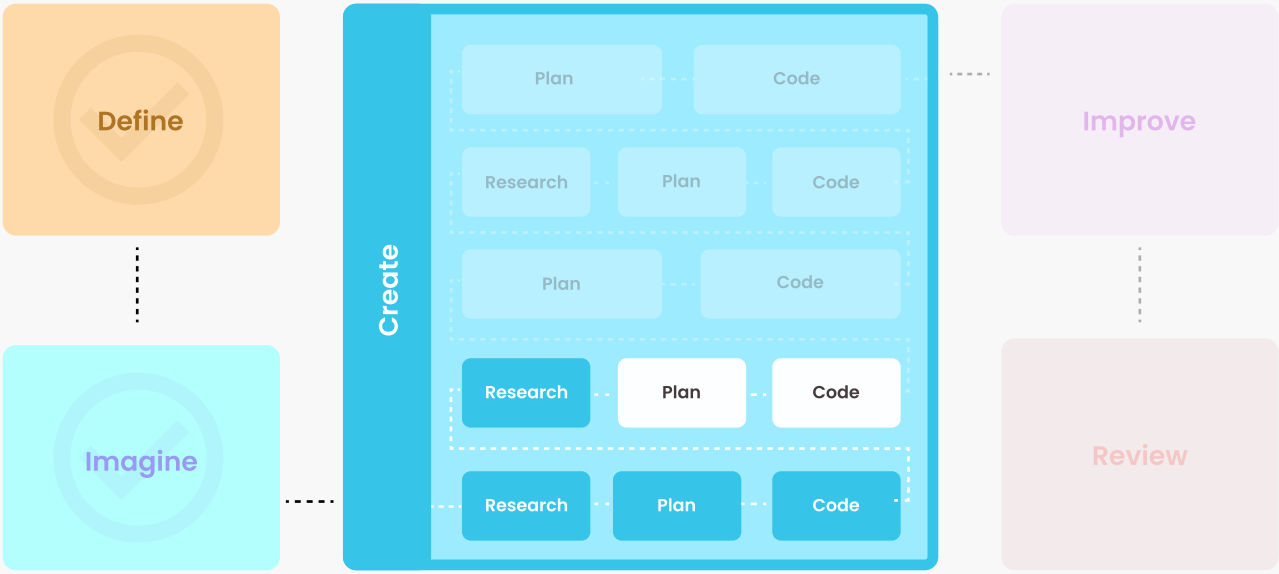
By the end of this lesson, learners will:

- Use logical reasoning to derive simple algorithms.
- Write programs that use sensor data to control an output.

## Lesson Resources

- Subproblem 2 Plan Solutions
- Subproblem 2 Code Solutions

## Project Progress



## Glossary

1. **Comparison:**
   In programming, a comparison is a statement where we compare two different values with an operator. The operator determines what characteristic we are comparing and is usually a mathematical symbol like equals, less than or greater than. The statement that we write will always be either TRUE or FALSE. For example, we might write a statement that says, "20 is less than 10." This statement would be FALSE. We might instead write "20 minus 10 equals 10." This statement would be TRUE. Comparisons are important in coding because we can use the TRUE/FALSE result to decide which branch of code we should run next.

2. **Flow control/branching:**
   In programming, the flow of code refers to the order in which code is run, and actions are performed. Think about it like walking down a tile path, where each tile represents a line of code. When you stand on a tile, that line of code is run. By looking ahead at the upcoming tiles, you can see the order that each line of code will be run. If your path splits into two, you can no longer tell exactly which code will be run in advance until you decide which path to walk down. Flow control is all about how we write our code to make these decisions. Branching is the simplest form of flow control, where the code "path" splits into multiple paths.

3. **Algorithm:**
   An algorithm is a sequence of rules or instructions. Computers usually follow algorithms to perform a calculation or make a decision. Good algorithms are well-defined, meaning that they have a rule or an instruction for every relevant possibility.

4. **Loops:**
   In programming, a loop refers to a piece of code that is run multiple times. The number of times that the code repeats itself depends on how the loop is written.

# Plan II 20 mins

5

In this step, learners aim to create a plan for navigating our robot around multiple curves in a line. To do this, they should answer the **Subsystem II Plan** questions within their **learning journal**.

Example answers for this plan step are available in this document for you to assist your learners if required.

It is not necessary for every learner to finish this lesson with a perfect plan. As long as they have something detailed out, they can improve it through trial and error once they start coding.

1. Briefly reiterate to the class how the **Code** step works, emphasising the iterative 'trial and error' nature:

**Build**
Learners will implement their solution by starting with individual elements and building up to the final solution. They should be referring to their plan as both a guide when they first get started and as a refernce point when they get stuck.

**Test**
Learners should constantly be testing their code to see how well their individual components or final solution solves the problem.

**Iterate**
Learners will continue building and testing until the problem has been completely solved.

**Share**
When learners think that they finished, encourage them to share their solution with other classmates who have finished and/or an educator to get their feedback. Make sure that the learners explain how they arrived at their solution. If necessary, they might need to start iterating again, either because a problem was identified or an educator/classmate gave them an idea for how to improve their solution.

2. Tell your learners to return to the platform and start the **Code** step for the second subsystem (https://app.createbase.co.nz/project/heat-seeker/create/Navigating%20curves/code).

3. Attached as a lesson resource is a help sheet with explanations that is for the educator's viewing only. You can use the help sheet to assist struggling learners and give additional prompts to excelling learners to make them think more about their solution.

4. If any learners finish early and you do not see any more ways for them to work on their solution in the Code step, then you could either ask them to help their other classmates or let them move onto the next sub-problem early.

**Homework**

For homework, educators could ask learners to complete their code at home as every student should have a working answer before moving on to the next sub-problem.

# Overview 6

## Plan III (20 mins)

Learners will get to detail their approach to solving the third sub-problem.

## Code III (25 mins)

Learners will use our platform to create and test their solutions in our simulation.

## Learning Outcomes

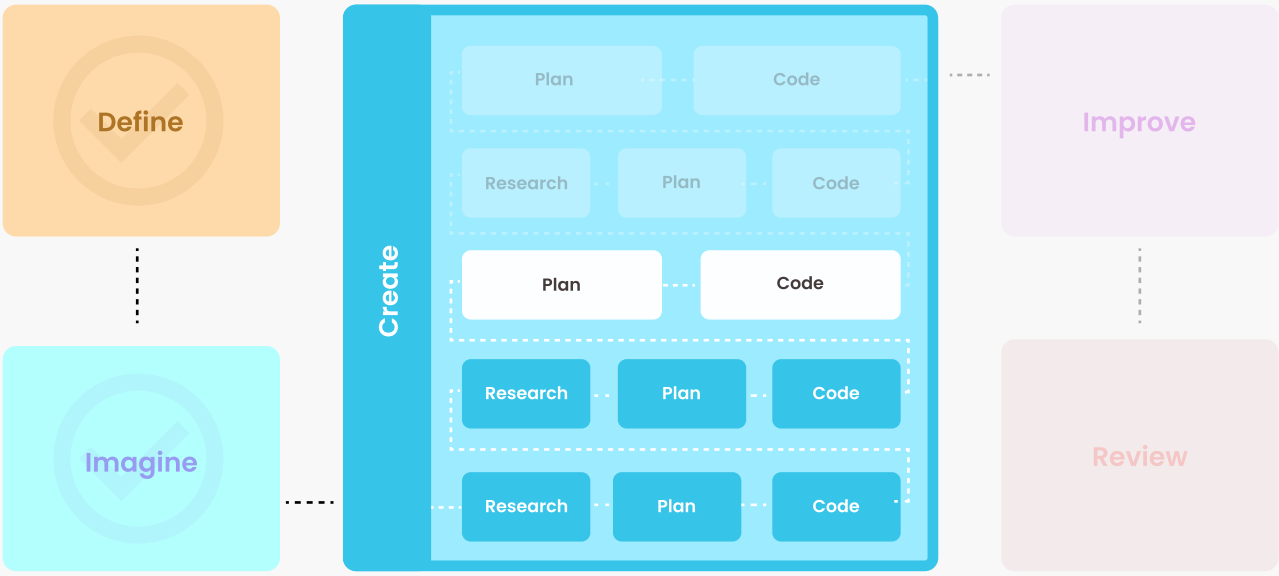
By the end of this lesson, learners will:

- Use logical reasoning to derive more complex algorithms.
- Write programs that use conditional statements to control an output.

## Lesson Resources

- Subproblem 3 Plan Solutions
- Subproblem 3 Code Solutions

## Project Progress



## Glossary

1. **Comparison:**
   In programming, a comparison is a statement where we compare two different values with an operator. The operator determines what characteristic we are comparing and is usually a mathematical symbol like equals, less than or greater than. The statement that we write will always be either TRUE or FALSE. For example, we might write a statement that says, "20 is less than 10." This statement would be FALSE. We might instead write "20 minus 10 equals 10." This statement would be TRUE. Comparisons are important in coding because we can use the TRUE/FALSE result to decide which branch of code we should run next.

2. **Flow control/branching:**
   In programming, the flow of code refers to the order in which code is run, and actions are performed. Think about it like walking down a tile path, where each tile represents a line of code. When you stand on a tile, that line of code is run. By looking ahead at the upcoming tiles, you can see the order that each line of code will be run. If your path splits into two, you can no longer tell exactly which code will be run in advance until you decide which path to walk down. Flow control is all about how we write our code to make these decisions. Branching is the simplest form of flow control, where the code "path" splits into multiple paths.

3. **Algorithm:**
   An algorithm is a sequence of rules or instructions. Computers usually follow algorithms to perform a calculation or make a decision. Good algorithms are well-defined, meaning that they have a rule or an instruction for every relevant possibility.

4. **Loops:**
   In programming, a loop refers to a piece of code that is run multiple times. The number of times that the code repeats itself depends on how the loop is written.

# Plan III 20 mins

6

In this step, learners aim to create a plan for navigating our robot around multiple turns in a line. Additionally, they should be able to find a way to differentiate between a curve and a turn.

To do this, they should answer the **Subsystem III Plan** questions within their **learning journal**.

Example answers for this plan step are available in this document for you to assist your learners if required.

# Code III 25 mins

6

1. Tell your learners to return to the platform and start the **Code** step for the third subsystem (https://app.createbase.co.nz/project/heat-seeker/create/Navigating%20turns/code).

2. Attached as a lesson resource is a help sheet with explanations that is for the educator's viewing only. You can use the help sheet to assist struggling learners and give additional prompts to excelling learners to make them think more about their solution.

3. If any learners finish early and you do not see any more ways for them to work on their solution in the Code step, then you could either ask them to help their other classmates or let them move onto the next sub-problem early.

### Homework

For homework, educators could ask learners to complete their code at home as every student should have a working answer before moving on to the next sub-problem.

# Overview 7

## Research IV (15 mins)

Learners will get a chance to research the tools available on our platform that they will need to use to construct their solution.

## Plan IV (15 mins)

Learners will get to detail their approach to solving the fourth sub-problem.

## Code IV (15 mins)

Your learners will use our platform to create and test their solutions in our simulation.

## Learning Outcomes

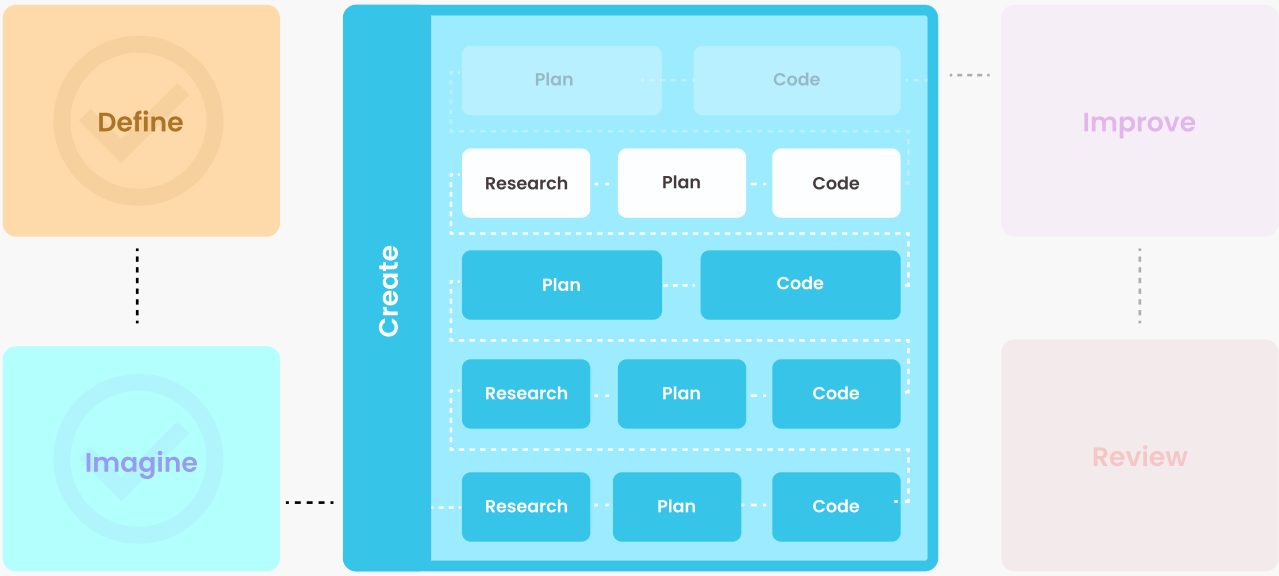
By the end of this lesson, learners will:

- Differentiate between different types of fires and how to deal with them.
- Define a while loop.
- Use while loops to solve problems.

## Lesson Resources

- Subproblem 4 Plan Solutions
- Subproblem 4 Code Solutions
- Phototransistors

## Project Progress



## Glossary

1. **Comparison:**
   In programming, a comparison is a statement where we compare two different values with an operator. The operator determines what characteristic we are comparing and is usually a mathematical symbol like equals, less than or greater than. The statement that we write will always be either TRUE or FALSE. For example, we might write a statement that says, "20 is less than 10." This statement would be FALSE. We might instead write "20 minus 10 equals 10." This statement would be TRUE. Comparisons are important in coding because we can use the TRUE/FALSE result to decide which branch of code we should run next.

2. **Flow control/branching:**
   In programming, the flow of code refers to the order in which code is run, and actions are performed. Think about it like walking down a tile path, where each tile represents a line of code. When you stand on a tile, that line of code is run. By looking ahead at the upcoming tiles, you can see the order that each line of code will be run. If your path splits into two, you can no longer tell exactly which code will be run in advance until you decide which path to walk down. Flow control is all about how we write our code to make these decisions. Branching is the simplest form of flow control, where the code "path" splits into multiple paths.

3. **Algorithm:**
   An algorithm is a sequence of rules or instructions. Computers usually follow algorithms to perform a calculation or make a decision. Good algorithms are well-defined, meaning that they have a rule or an instruction for every relevant possibility.

4. **Loops:**
   In programming, a loop refers to a piece of code that is run multiple times. The number of times that the code repeats itself depends on how the loop is written.

# Research IV 15 mins



This research is divided into two parts:

1. **Fire Fighting research**
2. **Blocks coding research**

For the **fire fighting research**, learners should read the first card on the platform under **Research**, discuss the answers to the questions inside, and add them to their **learning journal.**

Learners will now use the platform to learn what tools they have available to solve the defined problem and how these tools work. Students only need to research and answer the following questions for any new blocks that were introduced in this sub-problem.

1. **What sensing blocks are available?** Fire sensing block, Water tank level.
2. **What do they measure?** How close the car is to the fire and how much water is available to put out the fires.
3. **What sensors would be required to get this information? (Optional)** Photoresistor/ Light detector for detecting fires and a water level sensor for measuring the level of the tank. More Details are below.
4. **What action blocks are available?** Turn on/off the hose.
5. **What do they do?** Lets out water from the robot
6. **What actuators would be required to perform this action? (Optional)** A valve to open and close the hose. More Details are below.

You can also find some optional Information below that might be useful for a class discussion:

**What is a photoresistor/light sensor?**

https://www.elprocus.com/photoresistor-working-types-and-applications

A photoresistor is a type of light-dependent resistor that varies its resistance values based on the light falling on it.

**Where are photoresistors/light sensors used?**

https://www.elprocus.com/photoresistor-working-types-and-applications/

These are used as light sensors.

These are used to measure the intensity of light.

Night light and photography light meters use photoresistors.

Their latency property is used in audio compressors and outside sensing.

Photoresistors can also be found in Alarm clocks, outdoor clocks, solar street lamps, etc.

Infrared astronomy and Infrared Spectroscopy also use photoresistors for measuring mid-infrared spectral regions.

**How does a photoresistor/light sensor work?**

https://www.elprocus.com/photoresistor-working-types-and-applications/

These photoresistors tend to decrease their resistance values with an increase in the intensity of the light. When light is incident on the photoresistor, photons get absorbed by the semiconductor material. The energy from the photon gets absorbed by the electrons. When these electrons acquire sufficient energy to break the bond, they jump into the conduction band. Due to this, the resistance of the photoresistor decreases. With the decrease in resistance, conductivity increases.

Depending upon the type of semiconductor material used for photoresistor, their resistance range and sensitivity differs. In the absence of light, the photoresistor can have resistance values in megaohms. And during the presence of light, its resistance can decrease to a few hundred ohms.

**What is a water level sensor?**

https://www.elprocus.com/different-types-level-sensors-applications/

A level sensor is one kind of device used to determine the liquid level that flows in a system.

**Where are water level sensors used?**

https://www.elprocus.com/different-types-level-sensors-applications/

They are commonly used in applications like Tank level monitoring in chemical, water treatment, food, battery industries.

# Research IV 15 mins



**How does a water level sensor work?**

https://www.elprocus.com/different-types-level-sensors-applications/

There are multiple different types of water level sensors that all work differently. Some use distance while others check the pressure at the bottom of the liquid to determine how much is there.

**What is a valve?**

A valve controls the flow of air or a liquid through a pipe/duct/hose etc.

**Where are valves used?**

Valves have many uses, including controlling water for irrigation, industrial uses for controlling processes, residential uses such as on/off and pressure control to dish and clothes washers and taps in the home.

**How does a valve work?**

A valve is a mechanical stop that will physically stop any fluids flowing through a medium, normally a pipe. An electro-mechanical valve means that an electronic signal can control this valve. A signal is sent to a motor or other mechanical device which will move the physical obstruction, creating a gap. When opened, the higher pressure fluid is then able to flow through this gap.

# Plan IV 15 mins

7

In this step, your learners will plan how they will program their robot to detect and put out any fires that it encounters while following a line. Additionally, they should be able to find a way to differentiate between a curve and a turn.

To do this, they should answer the **Subsystem IV Plan** questions in their **learning journal.**

Answers for this plan step are available in this document for you to assist your learners if required.

# Code IV 20 mins



1. Tell your learners to return to the platformand start the **Code** step for the fourth subsystem (https://app.createbase.co.nz/project/heat-seeker/create/Extinguishing%20fires/code).

2. Attached as a lesson resource is a help sheet with explanations that is for the educator's viewing only. You can use the help sheet to assist struggling learners and give additional prompts to excelling learners to make them think more about their solution.

3. If any learners finish early and you do not see any more ways for them to work on their solution in the Code step, then you could either ask them to help their other classmates or let them move onto the next sub-problem early.

### Homework

For homework, educators could ask learners to complete their code at home as every student should have a working answer before moving on to the next sub-problem.

# Overview 8

## Plan V (20 mins)

Learners will get to detail their approach to solving the final problem.

## Code V (25 mins)

Your learners will use our platform to create and test their solutions in our simulation.

## Learning Outcomes

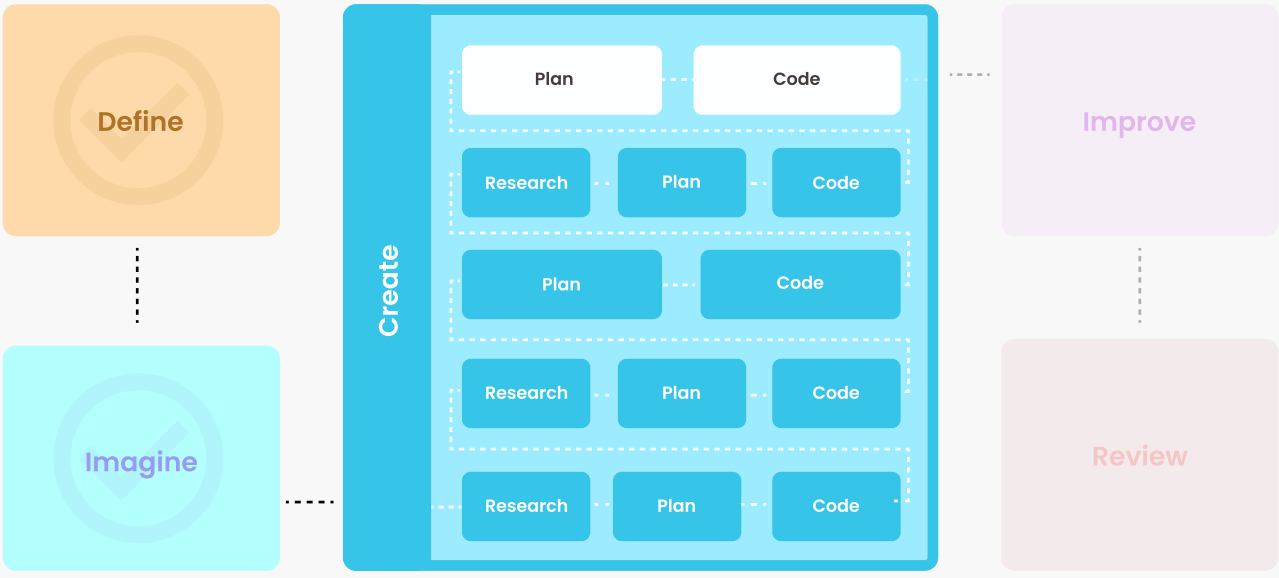
By the end of this lesson, learners will:

- Combine the solutions to different sub-problems to solve a larger problem.
- Explore ways a solution can be optimised.

## Lesson Resources

- Subproblem 5 Plan Solutions
- Subproblem 5 Code Solutions

## Project Progress



## Glossary

1. **Comparison:**
   In programming, a comparison is a statement where we compare two different values with an operator. The operator determines what characteristic we are comparing and is usually a mathematical symbol like equals, less than or greater than. The statement that we write will always be either TRUE or FALSE. For example, we might write a statement that says, "20 is less than 10." This statement would be FALSE. We might instead write "20 minus 10 equals 10." This statement would be TRUE. Comparisons are important in coding because we can use the TRUE/FALSE result to decide which branch of code we should run next.

2. **Flow control/branching:**
   In programming, the flow of code refers to the order in which code is run, and actions are performed. Think about it like walking down a tile path, where each tile represents a line of code. When you stand on a tile, that line of code is run. By looking ahead at the upcoming tiles, you can see the order that each line of code will be run. If your path splits into two, you can no longer tell exactly which code will be run in advance until you decide which path to walk down. Flow control is all about how we write our code to make these decisions. Branching is the simplest form of flow control, where the code "path" splits into multiple paths.

3. **Algorithm:**
   An algorithm is a sequence of rules or instructions. Computers usually follow algorithms to perform a calculation or make a decision. Good algorithms are well-defined, meaning that they have a rule or an instruction for every relevant possibility.

4. **Loops:**
   In programming, a loop refers to a piece of code that is run multiple times. The number of times that the code repeats itself depends on how the loop is written.

# Plan V 20 mins



No new research is required in this lesson, so your students can jump right into their **planning**.

In this step, your learners will combine all of the logic and solutions that they have built so far to solve the full problem: navigating around the entire warehouse and putting out any fires in our way.

To do that they will have to answer the questions within their **learning journals**.

Answers for this plan step are available in this document for you to assist your learners if required.

# Code V 20 mins



1. Tell your learners to return to the platform and start the **Code** step for the fith and final subsystem (https://app.createbase.co.nz/project/heat-seeker/create/Putting%20it%20all%20together/code).

2. Attached as a lesson resource is a help sheet with explanations that is for the educator's viewing only. You can use the help sheet to assist struggling learners and give additional prompts to excelling learners to make them think more about their solution.

3. If any learners finish early and you do not see any more ways for them to work on their solution in the Code step, then you could either ask them to help their other classmates or let them move onto the Improve step early.

### Homework

For homework, educators could ask learners to complete their code at home as every student should have a working answer before moving on to the next sub-problem.

# Overview 9

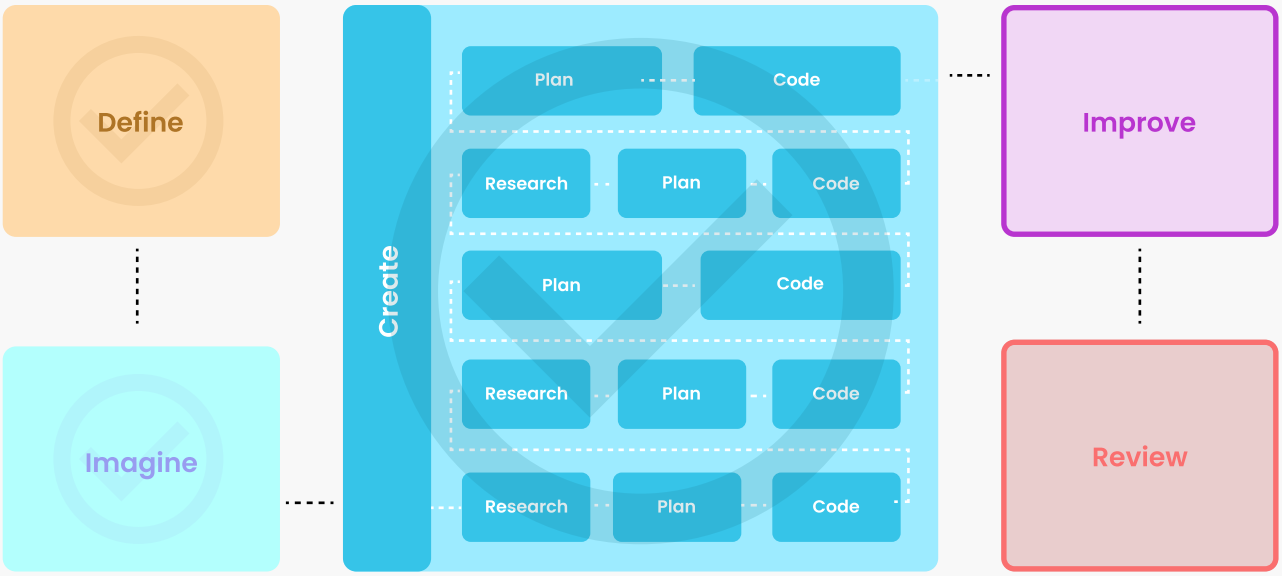
## Improve (30 mins)

Your learners will get a chance to apply the skills they have learnt and the knowledge they have gained. Learners will start to time their solutions and then try to optimise their code to increase the speed at which they can put out all of the fires.

## Review (15 mins)

Learners will then review their work. Options for review include having members of the class share their unique solutions and the decision-making that got them there and/or conducting self-assessments using their learning journal.

## Project Progress



# Improve

30 mins

9

1. To start the lesson, confirm the number of learners who have completed the **Create** step. Tell all of the learners who have a working solution that their next task is to move on to the **Improve** step.

2. If any learners have yet to complete the **Create** step, make sure they complete the **Create** step first and post a screenshot of their now working solution in their **learning journal**.

3. Let the learners who have finished the **Create** step try to complete the **Improve** step with minimal assistance. The **Improve** step helps reinforce the learners' learning. The purpose of the **Improve** step in this Project is to allow students to time their run and try and optimise their solution by speeding up the robot during the operation and tweaking the limits they set for fire and line following.

   **Some things that learners could experiment with to improve their solutions:**

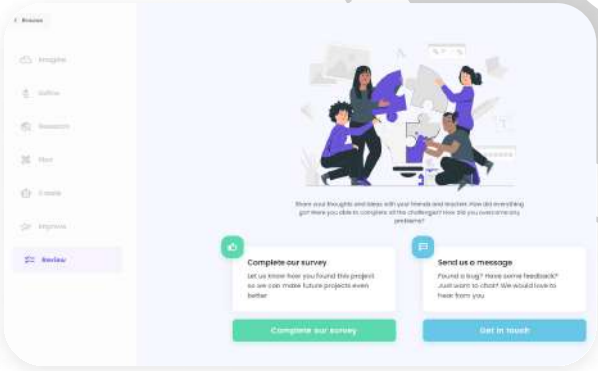
   1. **Base speed sent to the motors:** Whenever they add a left or right wheel block with a stated speed, they could change the values to speed the robot up. They might have to make sure it is not too fast that the robot starts turning left and right a lot.
   2. **Sensitivity to sensor changes:** Within the create step they will have conditions such as whether the middle line sensor is on the line or not (see code step 3 solution). They would have compared the sensor value with a certain value. The value will decide when the robot will decide to turn or not. If they tend to turn when they are not meant to, increasing the value there would be a good way to speed up the solution. Similarly, If they are turning too late, increasing the limit will be a good way to speed up.

5. At the end of the **Improve** step (when you feel like the learners have progressed far enough), tell the class to wrap up their work by taking a screenshot of their final code and inserting it into their learning journal. They should do this regardless of if they have a working solution or not.

# Review 15 mins



The final part of this Project is the **Review** step. We recommend doing one or both of the following: Share and Reflect. Feel free to do one or the other or both, depending on the time that you have available.




## Share

1. Encourage your learners to talk to the person next to them about their final solutions and their decisions at each step to their classmates.

   This helps learners remember what they have done and get new ideas by listening to how others solved the problem differently, which may help with learner reflection in the next task.

2. During this process, walk around the classroom and ask learners who do not have much to say prompts to get them talking.

## Reflect

1. Ask your learners to complete the Reflection section of the learning journal.

2. When every learner has completed their learning journal, you could ask some learners to share its contents and ask them questions as they do so.

3. **Optional**: At the end of the lesson, ask each learner to submit their completed learning journals somewhere where you have access in case you want to review them.

## Homework

To give your students an additional task, you could challenge your students to keep working on the Improve step at home and see who in the class can get the fastest time.