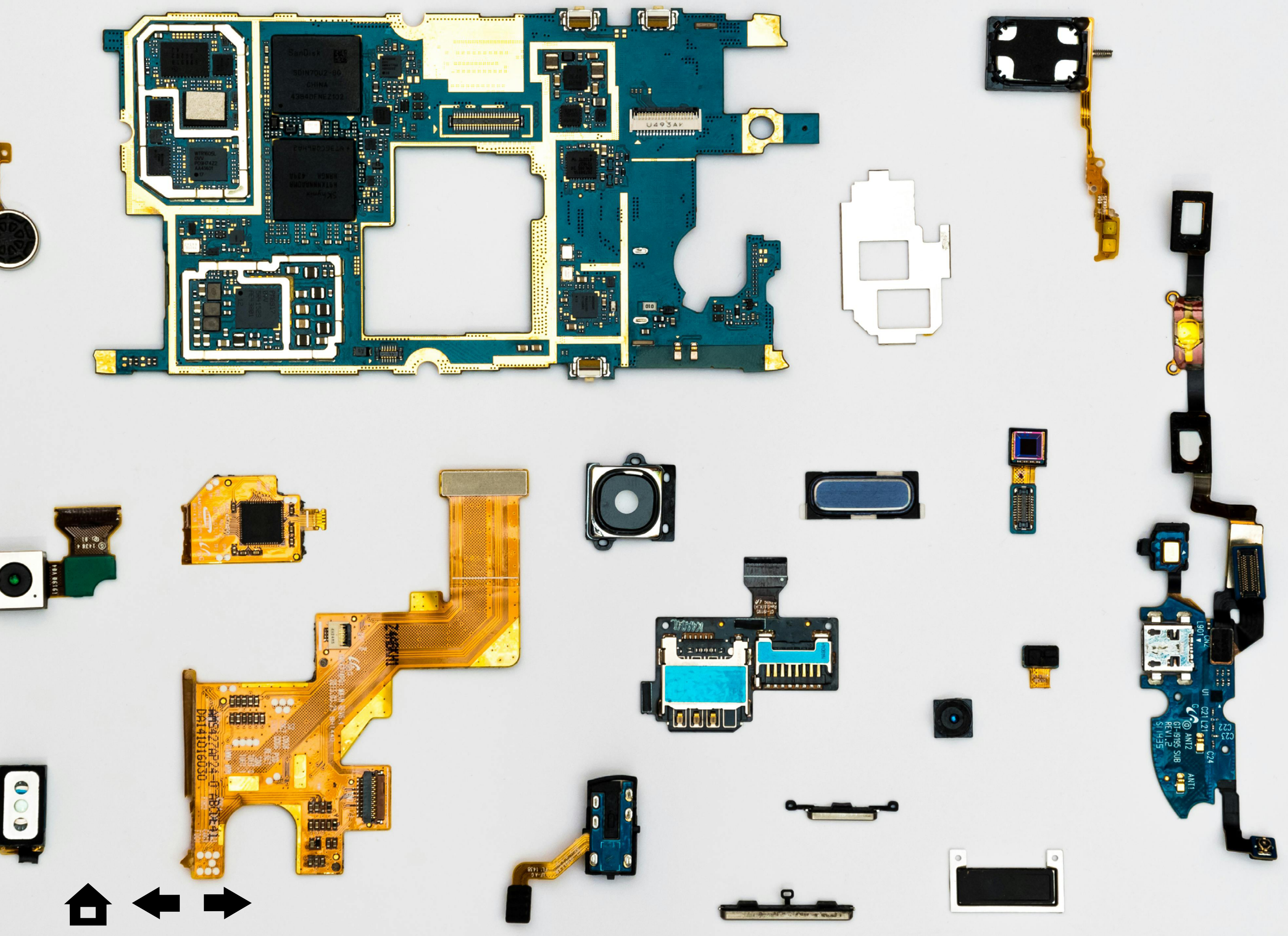
Robotics Year 1

SECTION 6

ROBOT CONSTRUCTION AND PROGRAMMING



ROBOT CONSTRUCTION & PROGRAMMING

Higher Order Design Thinking

Introduction

Welcome to Higher Order Thinking (HOT) in the design of programs for robotics! In this section, you will use critical thinking skills like analysis, synthesis, and evaluation to solve programming problems. You will focus on using flowcharts to create algorithms, making it easier to see the steps needed to solve robotics challenges.

At the end of this section, you will be able to:

- Determine the Inputs, Processes and Outputs required to solve a particular problem.
- Define solutions to basic automated and robotic problems using algorithms, pseudocodes, and flowchart diagrams.

Key Ideas:

- **Higher Order Thinking (HOT):** This simply means using your brain in more advanced ways than just remembering facts. It includes skills like analyzing, evaluating, creating, and inventing new ideas and approaches to solve problems. In robotics programming, HOT helps solve tough problems, create good algorithms, and make smart choices. For example, if a robot faces a new challenge, HOT allows the programmer to look at the problem from different perspectives and come up with a clever solution.
- **Algorithm:** An algorithm is a step-by-step procedure or set of rules for solving a specific problem or accomplishing a task. It is a logical sequence of actions that leads to a desired outcome. Algorithms are fundamental to programming robots, as they provide clear instructions for the robot to follow. For example, a line-following robot will use an algorithm to decide how to move based on sensor input.
- **Flowchart:** A flowchart is a visual representation of an algorithm or process. It uses standardised symbols (such as rectangles, diamonds, and arrows) to depict different actions, decisions, and the flow of control within the algorithm. Flowcharts are used to map out the logic of a robotic program before writing the pseudocode or final computer code. They help programmers picture how the robot will execute tasks step-by-step, making it easier to identify potential issues and optimise the design.
- **Pseudocodes:** Pseudocode is a way of describing an algorithm using plain English (or simplified code-like language). It focuses on the logic and sequence of steps rather than specific programming language. Pseudocode helps bridge the gap between an algorithm and a computer code. It allows programmers to picture the solution simply before writing the final program in code.

- **Debugging:** Debugging is the process of identifying, isolating, and fixing errors or bugs in a sequence of steps leading to an output. It helps to maximise the performance of the robot. It involves testing to find where the logic or code is failing and making necessary corrections. Debugging is important for ensuring that a robot behaves as expected. Errors in algorithms, pseudocode, or computer code can cause a robot to malfunction or not perform a task correctly.

HIGHER ORDER DESIGN THINKING: USING FLOWCHART DIAGRAMS TO CREATE ALGORITHMS

The following text will focus on using flowchart diagrams to create algorithms for solving basic problems in robotics. Flowcharts provide visual representations of the problem-solving process, making it easier to determine the inputs, processes, and outputs required for specific challenges in robotics programming.

Introduction to Flowchart Diagrams:

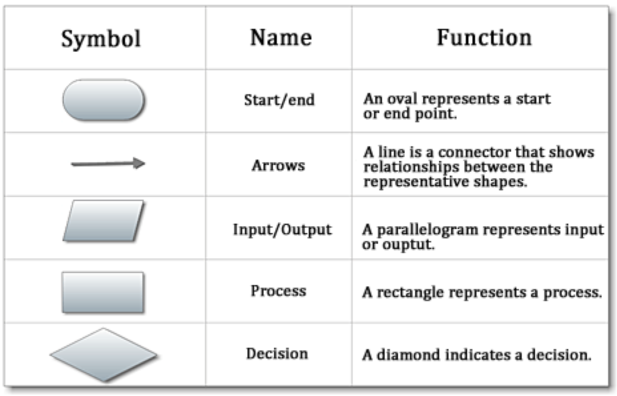
Flowchart diagrams use symbols and arrows to illustrate the sequence of steps in solving a problem. By using flowcharts, programmers can visualise the logic of an algorithm, making it easier to understand, debug, and optimise the resulting code. (Anderson et al 2002)

Creating Flowcharts for Algorithm Implementation:

To solve a programming problem using flowcharts, follow these steps:

a. **Define the Problem**: Clearly explain or state the problem you want to solve.

b. **Identify Inputs, Processes, and Outputs**: Decide on the data inputs, processes required, and the desired outputs for the solution.



Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or ouptut.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.

Fig. 6.1: Basic Flowchart Symbols and their Meanings

c. **Develop Flowchart**: Use the appropriate flowchart symbols to represent each step of the solution Connect the symbols with arrows to show the flow from input to output.

d. **Test the Flowchart**: Walk through the flowchart step-by-step, verifying the logic and ensuring that it produces the desired outputs for different scenarios.

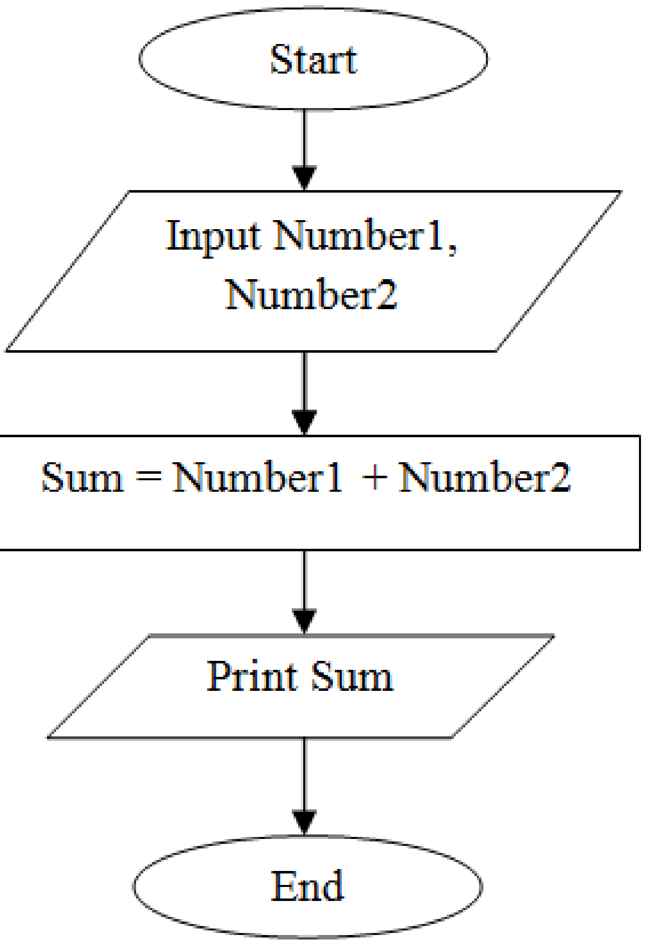


Fig. 6.2: Flowchart example for adding two numbers

The algorithm that can be written from the flow chart in fig. 6.2 is as follows:

```
Start
Input: Read two numbers, a and b.
Process: Calculate the sum of a and b and store it in a variable sum.
Output: Display the value of sum.
End
```

Applying Flowchart Diagrams in Robotics Programming:

Flowchart diagrams are used in robotics programming to:

a. **Design Navigation Algorithms**: Create flowcharts to plan a moving robot's path, help it make decisions based on sensor inputs, and control its movements efficiently.

b. **Implement Control Loops**: Use flowcharts to design return loops that adjust inputs to control robot actions based on feedback from sensors.

c. **Solve Logical Problems**: flowchart diagrams are used to solve logical challenges, such as obstacle avoidance, decision making, and pattern recognition.

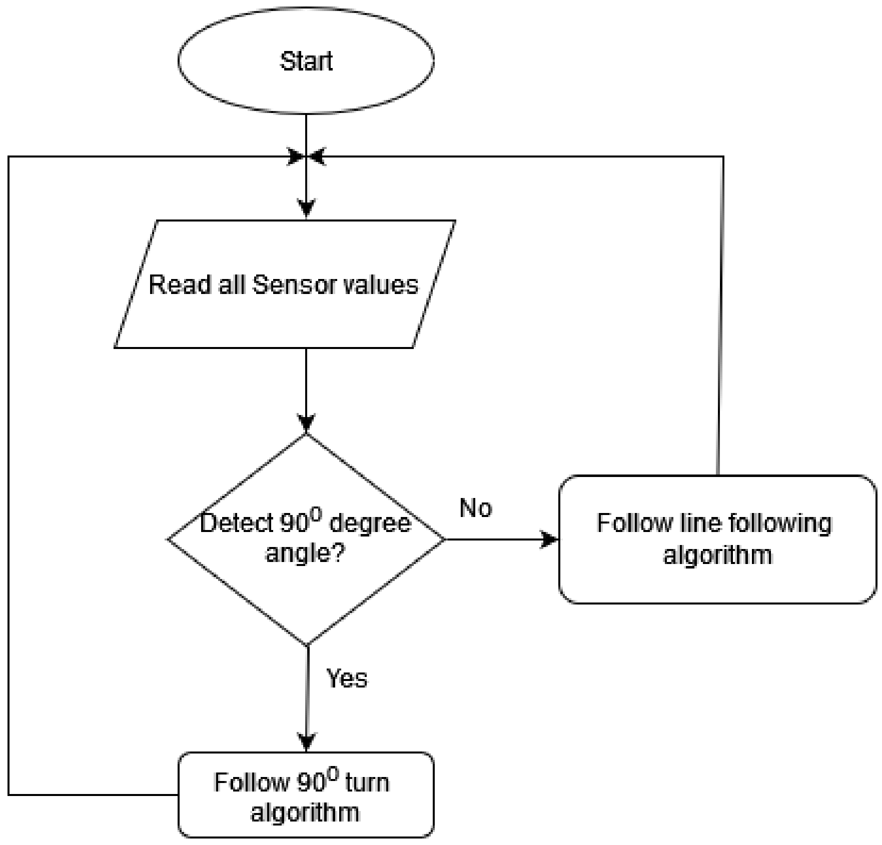


Fig. 6.3: Flowchart for line following robot

The algorithm that can be written from the flow chart in fig. 17.3 is as follows:

Start

Initialize: Set up the sensors and motors.

Input: Read values from the left and right sensors.

Process:

If both sensors detect the line, move forward.

If only the left sensor detects the line, turn left.

```
If only the right sensor detects the line, turn right.
If neither sensor detects the line, stop.
Output: Control the motors based on sensor readings.
Repeat: Continuously read sensors and adjust motors.
End
```

Activity 6.1

Choose one routine you follow in class or at home (e.g., sharpening a pencil or making breakfast porridge) and create a simple flowchart on paper. Identify the steps as inputs (e.g., Ingredients for porridge or tools for sharpening the pencil), processes and outputs. Break down each step if necessary.

Once you have your written instructions (your algorithm) then create your flowchart.

Use the correct symbols for start, stop, inputs, processes and outputs. Indicate the flow for your chart using lines to connect each symbol.

Activity 6.2

In a small group, complete the following Robotics Challenge: An EV3 robotic car has been given the task of moving forward in a straight line until it detects an obstacle, then turns 90° to avoid it.

1. Write down the steps in terms of inputs, processes, and outputs for controlling the robot's movements (your algorithm).
2. Create a flowchart representing your solution to the challenge. Ensure you use the correct symbols and logically organise the steps. Add the flow lines to connect each symbol.
3. Share your flowchart with a friend. Be ready for questions and to accept suggestions on how your flowchart might be improved.

ALGORITHMIC PROBLEM-SOLVING IN ROBOTICS: PSEUDOCODES AND FLOWCHART DIAGRAMS

You will learn how to use algorithms for problem-solving in robotics. You have already learnt that flowcharts can be drawn from algorithms, and vice versa.

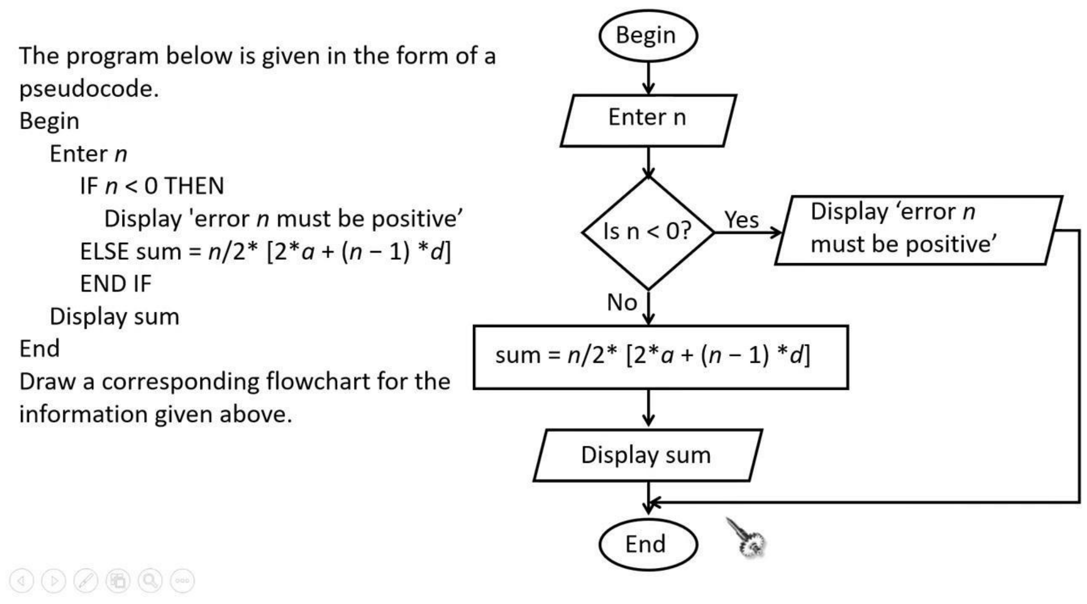
Understanding Algorithms and Pseudocodes in Robotics:

If you have an algorithm that describes the inputs and the steps that must be taken to achieve an output within a robotic system, changing direction to avoid and obstacle for example then this is the first stage in developing a program which will enable the robot to do this.

The next stage is to write the algorithm in a basic language, known as the Pseudocode.

Pseudocode can be described as a step-by-step description of an algorithm using plain English. It does not obey the rules of grammar. It may even make use of symbols or abbreviations. It bridges the gap between natural language and programming language. Once the pseudocode has been written then coding can begin.

The pseudocode can also be translated to a flowchart diagram before it is inputted into the robot control module using a selected computer programming language. By using pseudocode and creating a flowchart it helps to understand the sequence of steps and the decision-making process within the original algorithm.



The program below is given in the form of a pseudocode.

```
Begin
    Enter $n$
        IF $n < 0$ THEN
            Display 'error $n$ must be positive'
        ELSE sum = $n/2* [2*a + (n - 1) *d]$
        END IF
    Display sum
End
```

Draw a corresponding flowchart for the information given above.

Begin
Enter n
Is n < 0?
Yes
Display 'error n must be positive'
No
sum = n/2* [2*a + (n − 1) *d]
Display sum
End

Fig. 6.4: Example of an algorithm written in pseudocode and translated into a flowchart

Scan the QR code below for more on algorithms and how they are derived

Link	QR Code
What exactly is an algorithm? Algorithms explained | BBC Ideas : https://www.youtube.com/watch?v=ZnBF2GeAKbo	

Before you can write your own pseudocode, you need to learn the basic rules. The use of keywords is important. Keywords are commonly used to represent various programming constructs and control flow elements. Here are some of the most frequently used keywords:

BEGIN, END: Used to denote the start and end of the pseudocode.

INPUT, GET, READ: Used for input operations.

OUTPUT, PRINT, DISPLAY: Used to output data.

IF, ELSE, ENDIF: Used for conditional statements.

WHILE, ENDWHILE: Used for while loops.

FOR, ENDFOR: Used for loops.

REPEAT, UNTIL: Used for repeat-until loops.

COMPUTE, CALCULATE: Used for performing calculations.

ADD, SUBTRACT, MULTIPLY, DIVIDE: Used for arithmetic operations.

//: Used to denote comments

This example shows how some of the keywords are used to write pseudocode:

```
BEGIN
  INPUT number
  IF number > 0 THEN
    PRINT «Positive»
  ELSE
    PRINT «Non-positive»
  END IF
END
```

The example gives the steps, in pseudocode, to check if a number is positive. Learn more about pseudocode by watching the following clip :

https://www.youtube.com/watch?v=HhBrkpTqzqg

Defining Solutions to Basic Robotic Problems:

In this part of section 6 you will study some basic robotic problems and create solutions using algorithms, pseudocodes, and flowchart diagrams. Some examples of problems you might meet when designing robots include:

a. **Obstacle Avoidance**: Create algorithms to enable the robot to navigate around obstacles in its path.

b. **Line Following**: Develop algorithms that allow the robot to follow a line using sensors and motors.

c. **Sumo Robot Strategy**: Design algorithms for a sumo robot to detect and push opponents out of a sumo ring. (Sumo wrestling is a sport where two opponents try to push one another out of a large circular area)

This first example shows a flow chart which represents the action a robot should take if it meets something in its path. The robot continues moving forward until it meets an obstacle then stops.

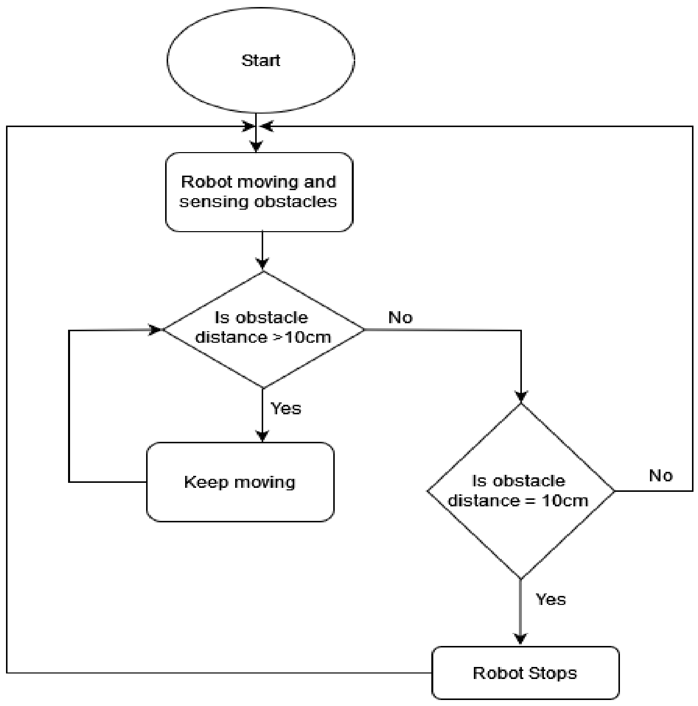

Start
Robot moving and sensing obstacles
Is obstacle distance >10cm
No
Yes
Keep moving
Is obstacle distance = 10cm
No
Yes
Robot Stops

Fig. 6.5: Flowchart diagram for a robot for obstacle avoidance

This second example shows the development of Pseudocode and a Flowchart from an algorithm for a robot which must follow a line until it reaches a destination.

a. Algorithm

Initialisation:

i. Start the robot and initialise the sensors (e.g., light sensors or infrared sensors) to detect the line.

ii. Calibrate the sensors to recognise the difference between the line (usually dark) and the floor (usually light).

Sensor Input: Continuously read the sensor values to determine the robot's position relative to the line.

Decision Making:

i. If the sensor on the left detects the line, steer the robot to the left.

ii. If the sensor on the right detects the line, steer the robot to the right.

iii. If both sensors detect the line (centred), move forward.

iv. If neither sensor detects the line, stop or search for the line by rotating.

Movement: Adjust the motors to control the robot's direction based on the sensor input.

Repeat: Continuously repeat the above steps to keep the robot following the line.

End: The robot stops when it reaches the destination (e.g., a stop marker at the end of the line).

b. Pseudocode and Flowchart

```
START

INITIALISE robot
INITIALISE sensors

CALIBRATE sensors

WHILE destination not reached DO
    READ sensor values

IF left sensor detects line THEN
    STEER robot left
ELSE IF right sensor detects line THEN
    STEER robot right
ELSE IF both sensors detect line THEN
    MOVE forward
ELSE
    STOP or SEARCH for line
END IF

    ADJUST motors based on sensor
    input
END WHILE

STOP robot at destination
```

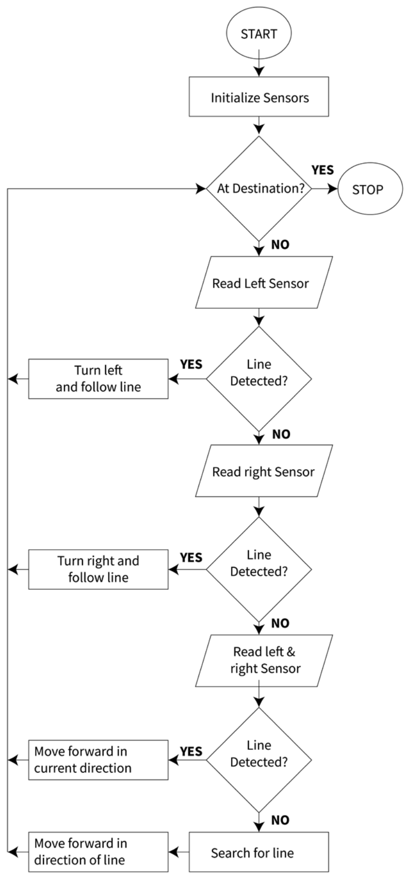


Fig. 6.6: Pseudocode and flowchart for line following algorithm

Scan the QR Code below for videos on how to create flowcharts from problem statements

Link	QR Code
https://www.youtube.com/watch?v=NdS8J9lHWgE	

Practice Problems:

1. **Problem: Design a Robot Maze Solver**
   Task: Create an algorithm, pseudocode, and flowchart diagram for a robot to solve a simple maze autonomously. The robot should explore the maze, avoiding dead-ends, and find its way to the centre.

2. **Problem: Navigate a Square Path**
   Task: Program a robot to navigate a square path and return to the starting point. Create an algorithm, pseudocode, and flowchart diagram for a robot.

3. **Problem: Obstacle Avoidance**
   Task: Create an algorithm and pseudocode for the obstacle avoidance flowchart in **Fig. 6.5.**

Activity 6.3

Select either problem one (1) or two (2) to solve. Write an algorithm to solve the chosen problem.

a. Convert the algorithm into pseudocode. Use plain English to describe each step.
b. Turn your pseudocode into a visual flowchart. Use the correct symbols to represent each decision point and action

Reflect on your solution to the problem. What did you find challenging? How did you ensure the algorithm was correct and worked with the minimum number of steps?

Activity 6.4

Study the obstacle avoiding pseudocode below. Go through it line by line and explain in a sentence what happens at each step. Try to spot any areas where the pseudocode could be simplified and rewrite it making the necessary changes.

```
START robot
INITIALISE sensors
CALIBRATE sensors
WHILE (robot is running)
    READ front sensor
IF (front sensor detects obstacle)
    STOP
    READ left and right sensors
IF (left sensor is clear and right sensor is blocked)
            TURN left
            MOVE forward
ELSEIF (right sensor is clear and left sensor is blocked)
TURN right
            MOVE forward
ELSEIF (both sensors are clear)
CHOOSE a random direction
            MOVE forward
ELSE
ROTATE 180 degrees (U turn)
MOVE forward
ELSE
MOVE forward
STOP
```

How could you improve this pseudocode to make the robot react faster or handle more complex obstacles?

Activity 6.5

Refer to the flowchart in **Fig. 6.5**.

1. Write an algorithm which outlines the step-by-step procedure that the robot would follow based on the flowchart.

   Guiding Questions:

   a. What does the robot do when it starts?
   b. What happens when the robot meets an obstacle?
   c. What decisions does the robot make at each step?
2. Convert the algorithm you deduced from the flowchart into pseudocode. Use plain English and simple instructions to describe each step.
3. Based on the algorithm and pseudocode you developed, look for ways to refine or optimise the original flowchart. Consider if the logic can be made simpler, more efficient, or if additional decision points or processes are needed.
4. Use flowchart symbols to redraw the refined version of the original flowchart. Make sure the new flowchart accurately reflects your improved algorithm and pseudocode.

Review Questions

Review Questions 6.1: Using Flowchart Diagrams To Create Algorithms

Choose the correct answer:

What is the purpose of a flowchart?

a) To write the final code

b) To visually represent an algorithm

c) To document software requirements

d) To test hardware components

Add the correct missing word:

1. The three main components of a problem-solving process in programming are inputs, ___________, and outputs.
2. A flowchart uses ___________ to show the flow of control from one step to another.
3. ___________ diagrams provide a visual representation of algorithms.

State whether the sentence is true or false:

4. Flowcharts are only used in robotics programming.
5. The synthesis of information is one of the Higher Order Thinking skills.
6. In a flowchart, the parallelogram symbol is used to represent input/output operations.
7. Testing a flowchart step-by-step is unnecessary once it has been created.

Extended writing:

8. Describe the purpose of using flowchart diagrams in robotics programming.
9. a. Outline three ways in which flowcharts can be used in the programming of robots.

   b. Create a flow chart for the following algorithm using the correct flowchart symbols.

   Start

   Step 1 Input two numbers: (a) and (b).

   Step 2 Calculate the product: (product = a*b).

   Step 3 Divide the product by 2: (a*b)/2

   Step 4 Display result

   End

Review Questions 6.2: Algorithmic Problem-solving in Robotics

1. Which of the following best describes the purpose of a flowchart in robotics programming?
   A. To provide a written description of a robot's behaviour
   B. To represent the robot's structure
   C. To visually represent the sequence of steps in an algorithm
   D. To debug robot hardware issues
2. What is the main role of pseudocode in problem-solving?
   A. To provide a visual representation of a problem
   B. To execute the robot's program directly
   C. To bridge the gap between an algorithm and the actual code
   D. To specify the physical components of a robot
3. In a line-following robot, which component is typically responsible for detecting the line?
   A. Motors
   B. Sensors
   C. Actuators
   D. Microcontroller
4. Pseudocode uses specific programming language syntax to ensure correct implementation.
   A. True
   B. False
5. A flowchart is a useful tool for debugging the logical flow of an algorithm.
   A. True
   B. False
6. The purpose of an algorithm in robotics is to provide a sequence of steps to achieve a specific task.
   A. True
   B. False
7. A flowchart uses ____________ which represent decisions, actions, and the flow of control.
8. Briefly explain the difference between an algorithm and pseudocode.

9. How can flowcharts help in the design and development of robotic programs?
10. Why is it important to refine a flowchart after creating the initial version?

REFERENCES

1. Cross, N. (2023). *Design thinking: Understanding how designers think and work*. Bloomsbury Publishing.
2. Andersen, B., Fagerhaug, T., & Henriksen, B. (2002). *Mapping work processes*. Quality Press
3. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. MIT press.

ACKNOWLEDGEMENTS

Ghana Education Service (GES)

(GAST)

List of Contributors

Name	Institution
Isaac Nzoley	Wesley Girls' High School
Kwame Oteng Gyasi	Kwame Nkrumah University of Science and Technology
Kwame Owusu Opoku	Prempeh College